IBM SPSS Modeler 17 - Guide de génération de scripts Python et d'automatisation

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations figurant à la section «Remarques», à la page 319.

Informations produit

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.ibm.com/ca/fr (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

Cette édition s'applique à la version 17, édition 0, modification 0 d'IBM(r) SPSS(r) Modeler et à toutes les éditions et modifications ultérieures sauf mention contraire dans les éditions suivantes.

Table des matières

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	**IBM Canada**
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

France	Canada	Etats-Unis
(Pos1)		Home
Fin	Fin	End
(PgAr)		PgUp
(PgAv)		PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
(Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

Chapitre 1. Génération de scripts et langage de script

Génération de scripts - Présentation

La génération de scripts IBM® SPSS Modeler est un outil performant pour automatiser les processus dans l'interface utilisateur. Les scripts permettent d'effectuer les mêmes opérations qu'avec la souris ou le clavier. Vous pouvez les utiliser pour automatiser les tâches dont l'exécution manuelle s'avère très répétitive et très longue.

Vous pouvez utiliser les scripts pour :

- Imposer un ordre spécifique d'exécution des noeuds dans un flux.
- Définir des propriétés pour un noeud et effectuer des calculs en utilisant un sous-ensemble du CLEM (Control Language for Expression Manipulation).
- Définir l'exécution automatique d'une séquence d'actions qui nécessite normalement l'intervention de l'utilisateur (création et test d'un modèle, par exemple).
- Mettre en place des processus complexes dans lesquels l'intervention de l'utilisateur est importante (procédures de validation croisée qui exigent la création et le test de nombreux modèles, par exemple).
- Mettre en place des processus de gestion des flux (exécution d'un flux d'apprentissage de modèle et production automatique du flux de test modèle correspondant, par exemple).

Ce chapitre offre des descriptions et des exemples de haut niveau de scripts de flux, de scripts autonomes et de scripts dans les super noeuds dans l'interface IBM SPSS Modeler. Des informations sur la syntaxe, les commandes et le langage de script vous sont fournies dans les des sections suivantes.

Remarque : vous ne pouvez pas importer et exécuter des scripts créés dans IBM SPSS Statistics avec IBM SPSS Modeler.

Types de script

IBM SPSS Modeler utilise trois types de script :

- **Les scripts de flux** sont stockés en tant que propriété de flux et sont, par conséquent, sauvegardés et chargés avec un flux spécifique. Par exemple, vous pouvez écrire un script de flux qui automatise le processus de formation et d'application d'un nugget de modèle. Vous pouvez également préciser que lorsqu'un script particulier s'exécute, ce script doit être exécuté à la place du contenu de l'espace de travail de flux.
- Les**scripts autonomes** ne sont pas associés à un flux en particulier et sont sauvegardés dans des fichiers texte externes. Par exemple, vous pouvez utiliser un script autonome pour manipuler plusieurs flux ensemble.
- Les**scripts de super noeud** sont stockés en tant que propriété de flux de super noeud. Les scripts de super noeud sont uniquement accessibles dans les super noeuds terminaux. Vous pouvez utiliser un script de super noeud pour contrôler la séquence d'exécution des contenus de super noeuds. Pour les super noeuds qui ne sont pas terminaux (source ou processus), vous pouvez définir les propriétés du super noeud ou des noeuds qu'il contient directement dans votre script de flux.

Scripts de flux

Les scripts peuvent être utilisés pour personnaliser les opérations réalisées au sein d'un flux particulier, et ils peuvent être enregistrés avec ce flux. Les scripts de flux peuvent être utilisés pour spécifier un ordre d'exécution particulier des noeuds terminaux dans un flux. La boîte de dialogue du script de flux permet d'éditer le script qui est enregistré avec le flux en cours.

Pour accéder à l'onglet du script de flux dans la boîte de dialogue Propriétés du flux :

1. Dans le menu Outils, sélectionnez :

   Propriétés du flux > **Exécution**
2. Cliquez sur l'onglet **Exécution** pour utiliser les scripts du flux en cours.

Les icônes de la barre d'outils se trouvant en haut de la boîte de dialogue du script de flux vous permettent d'effectuer les opérations suivantes :

- Importer les contenus d'un script autonome préexistant dans la fenêtre.
- Sauvegarder un script comme fichier texte.
- Imprimer un script.
- Ajouter le script par défaut.
- Editer un script (annuler, couper, copier, coller et autres fonctions d'édition courantes).
- Exécuter entièrement le script en cours.
- Exécuter les lignes sélectionnées d'un script.
- Arrêter un script pendant l'exécution. (Cette icône est activée uniquement lorsqu'un script est en cours d'exécution.)
- Vérifier la syntaxe d'un script et, en cas d'erreurs, les afficher dans le panneau inférieur de la boîte de dialogue.

A partir de la version 16.0, SPSS Modeler utilise le langage de script Python. Toutes les versions antérieures utilisaient un langage de script propre à SPSS Modeler, appelé désormais "scriptage existant". En fonction du type de script que vous utilisez, dans l'onglet **Exécution**, sélectionnez le mode d'exécution **Par défaut (script facultatif)**, puis **Python** ou **Existant**.

De plus, vous pouvez spécifier si ce script doit ou non être exécuté lorsque le flux est exécuté. Vous pouvez sélectionner **Exécuter ce script** pour exécuter le script chaque fois que le flux est exécuté en respectant l'ordre d'exécution du script. Ce paramètre permet l'automatisation au niveau du flux et l'accélération de la vitesse de création des modèles. Cependant, le paramètre par défaut est d'ignorer ce script pendant l'exécution du flux. Même si vous sélectionnez l'option **Ignorer ce script**, vous pouvez quand même exécuter ce script directement depuis la boîte de dialogue.

L'éditeur de script inclut les fonctions suivantes qui facilitent la création de script :

- Mise en évidence de la syntaxe ; les mots-clés, les valeurs littérales (telles que les chaînes et les nombres) et les commentaires sont mis en évidence.
- Numérotation des lignes.
- Mise en correspondance des blocs ; lorsque le curseur est placé au début d'un bloc de programme, le bloc de fin correspondant est également mis en évidence.
- Suggestion de saisie automatique.

Vous pouvez personnaliser les couleurs et les styles de texte utilisés par le surligneur de syntaxe via les préférences d'affichage IBM SPSS Modeler. Pour accéder aux préférences d'affichage, sélectionnez **Outils** > **Options** > **Options utilisateur**, puis cliquez sur l'onglet **Syntaxe**.

Vous pouvez accéder à une liste de suggestions syntaxiques. Pour cela, sélectionnez **Suggestion automatique** dans le menu contextuel ou appuyez sur Ctrl + Espace. Utilisez les touches du curseur pour vous déplacer vers le haut ou vers le bas dans la liste, puis appuyez sur Entrée pour insérer le texte sélectionné. Appuyez sur la touche Echap pour quitter le mode Suggestion automatique sans modifier le texte existant.

L'onglet **Débogage** affiche des messages de débogage et peut servir à évaluer l'état du script après l'exécution du script. L'onglet **Débogage** est composé d'une zone de texte en lecture seule et d'un champ de texte d'entrée de ligne unique. La zone de texte affiche le texte qui est envoyé soit à la sortie standard,

soit à l'erreur standard par les scripts, par exemple via le texte de message d'erreur. Le champ de texte d'entrée utilise l'entrée de l'utilisateur. Cette entrée est ensuite évaluée dans le contexte du script le plus récemment exécuté dans la boîte de dialogue (appelé *contexte de génération de scripts*). La zone de texte contient la commande et le résultat de sortie pour que l'utilisateur ait une trace des commandes. Le champ de texte d'entrée contient toujours l'invite de commande (`-->` pour le scriptage existant).

Un nouveau contexte de génération de scripts est créé dans les circonstances suivantes :

- Un script est exécuté à l'aide du bouton “Exécuter ce script” ou “Exécuter les lignes sélectionnées”.
- Le langage de script est modifié.

Si un nouveau contexte de génération de scripts est créé, le contenu de la zone de texte est effacé.

Remarque : L'exécution d'un flux en dehors du panneau de script ne modifiera pas le contexte de script du panneau de script. Les valeurs des variables créées dans le cadre de cette exécution ne seront pas visibles dans la boîte de dialogue du script.

Exemple de script de flux : apprentissage d'un réseau de neurones

Lorsqu'il est exécuté, un flux peut être utilisé pour entraîner un modèle de réseau de neurones. La procédure de test normale consiste à exécuter le noeud de modélisation pour ajouter le modèle au flux, définir les connexions appropriées et exécuter un noeud Analyse.

L'utilisation d'un script IBM SPSS Modeler permet d'automatiser le processus de test du nugget de modèle après sa création. Par exemple, le script de flux suivant pour tester le flux de démo *druglearn.str* (disponible dans le dossier */Demos/streams/* dans votre installation IBM SPSS Modeler) peut être exécuté dans la boîte de dialogue Propriétés du flux (**Outils > Propriétés du flux > Script**) :

```
stream = modeler.script.stream()
neuralnetnode = stream.findByType("neuralnetwork", None)
results = []
neuralnetnode.run(results)
appliernode = stream.createModelApplierAt(results[0], "Drug", 594, 187)
analysisnode = stream.createAt("analysis", "Drug", 688, 187)
typenode = stream.findByType("type", None)
stream.linkBetween(appliernode, typenode, analysisnode)
analysisnode.run([])
```

Les puces suivantes décrivent chaque ligne dans cet exemple de script.

- La première ligne définit une variable qui pointe vers le flux en cours.
- Ligne 2, le script recherche le noeud de création de réseau de neurones.
- Ligne 3, le script crée une liste dans laquelle les résultats d'exécution peuvent être stockés.
- Ligne 4, le nugget de modèle Réseau de neurones est créé. Il est stocké dans la liste définie à la ligne 3.
- Ligne 5, un noeud d'application de modèle est créé pour le nugget de modèle et placé dans le canevas de flux.
- Ligne 6, un noeud Analyse appelé `Drug` est créé.
- Ligne 7, le script recherche le noeud Type.
- Ligne 8, le script connecte le noeud d'application de modèle créé ligne 5 entre le noeud Type et le noeud Analyse.
- Finalement, le noeud Analyse est exécuté pour produire le rapport Analyse.

Il est possible d'utiliser un script pour créer intégralement un flux à partir d'un espace de travail vierge, et exécuter ce flux. Pour en savoir plus sur le langage de script en général, consultez Langage de script. Pour en savoir plus sur des commandes de script spécifiques, consultez Commandes de script.

Scripts autonomes

Dans la boîte de dialogue Script autonome, vous pouvez créer ou éditer un script sauvegardé en tant que fichier texte. Cette boîte de dialogue indique le nom du fichier et fournit des fonctions pour charger, enregistrer, importer et exécuter des scripts.

Pour accéder à la boîte de dialogue du script autonome :

Dans le menu principal, sélectionnez :

Outils > **Script autonome**

Les scripts autonomes disposent de la même barre d'outils et des mêmes options de vérification de syntaxe de script que les scripts de flux. Pour plus d'informations, voir la rubrique «Scripts de flux», à la page 1.

Exemple de script autonome : Enregistrement et chargement d'un modèle

Les scripts autonomes permettent de gérer des flux. Imaginez que vous avez deux flux : l'un qui crée un modèle et l'autre qui utilise les graphiques pour explorer l'ensemble de règles généré à partir du premier flux contenant des champs de données existants. Le script correspondant à ce cas de figure pourrait être semblable au suivant :

```
taskrunner = modeler.script.session().getTaskRunner()

# Modify this to the correct Modeler installation Demos folder.
# Note use of forward slash and trailing slash.
installation = "C:/Program Files/IBM/SPSS/Modeler/16/Demos/"

# First load the model builder stream from file and build a model
druglearn_stream = taskrunner.openStreamFromFile(installation + "streams/druglearn.str", True)
results = []
druglearn_stream.findByType("c50", None).run(results)

# Save the model to file
taskrunner.saveModelToFile(results[0], "rule.gm")

# Now load the plot stream, read the model from file and insert it into the stream
drugplot_stream = taskrunner.openStreamFromFile(installation + "streams/drugplot.str", True)
model = taskrunner.openModelFromFile("rule.gm", True)
modelapplier = drugplot_stream.createModelApplier(model, "Drug")

# Now find the plot node, disconnect it and connect the
# model applier node between the derive node and the plot node
derivenode = drugplot_stream.findByType("derive", None)
plotnode = drugplot_stream.findByType("plot", None)
drugplot_stream.disconnect(plotnode)
modelapplier.setPositionBetween(derivenode, plotnode)
drugplot_stream.linkBetween(modelapplier, derivenode, plotnode)
plotnode.setPropertyValue("color_field", "$C-Drug")
plotnode.run([])
```

Remarque : Pour en savoir plus sur le langage de script en général, consultez Présentation du langage de script. Pour en savoir plus sur des commandes de script spécifiques, consultez Commandes de script.

Exemple de script autonome : Génération d'un modèle Sélection de fonction

A partir d'un espace de travail vierge, cet exemple crée un flux qui génère un modèle Sélection de fonction, applique le modèle et crée un tableau répertoriant les 15 champs les plus importants pour la cible indiquée.

```
stream = modeler.script.session().createProcessorStream("featureselection", True)

statisticsimportnode = stream.createAt("statisticsimport", "Statistics File", 150, 97)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/customer_dbase.sav")

typenode = stream.createAt("type", "Type", 258, 97)
typenode.setKeyedPropertyValue("direction", "response_01", "Target")

featureselectionnode = stream.createAt("featureselection", "Feature Selection", 366, 97)
featureselectionnode.setPropertyValue("top_n", 15)
featureselectionnode.setPropertyValue("max_missing_values", 80.0)
featureselectionnode.setPropertyValue("selection_mode", "TopN")
featureselectionnode.setPropertyValue("important_label", "Check Me Out!")
featureselectionnode.setPropertyValue("criteria", "Likelihood")

stream.link(statisticsimportnode, typenode)
stream.link(typenode, featureselectionnode)
models = []
featureselectionnode.run(models)

# Assumes the stream automatically places model apply nodes in the stream
applynode = stream.findByType("applyfeatureselection", None)
tablenode = stream.createAt("table", "Table", applynode.getXPosition() + 96,
applynode.getYPosition())
stream.link(applynode, tablenode)
tablenode.run([])
```

Le script crée un noeud source permettant la lecture des données, utilise un noeud type pour affecter le rôle (direction) `Target` au champ `response_01`, puis crée et exécute un noeud Sélection de fonction. Le script connecte également les noeuds et les positionne sur l'espace de travail de flux afin d'offrir une présentation lisible. Le nugget de modèle résultant est ensuite connecté à un noeud Table qui répertorie les 15 champs les plus importants, en fonction des propriétés `selection_mode` et `top_n`. Pour plus d'informations, voir la rubrique «Propriétés de featureselectionnode», à la page 187.

Scripts de super noeud

Vous pouvez créer et sauvegarder des scripts dans tous les super noeuds terminaux avec le language de script de IBM SPSS Modeler. Ces scripts sont uniquement accessibles pour les super noeuds terminaux et sont souvent utilisés lors de la création de modèles de flux ou pour imposer un ordre d'exécution particulier aux contenus des super noeuds. Les scripts de super noeud permettent également d'exécuter plusieurs scripts dans un même flux.

Par exemple, imaginons que vous avez besoin de préciser l'ordre d'exécution d'un flux complexe et que votre super noeud contient plusieurs noeuds qui comprennent un noeud v. globales qui doit être exécuté avant de calculer un nouveau champ utilisé dans un noeud Tracé. Dans ce cas, vous pouvez créer un script de super noeud qui exécute d'abord le noeud v. globales. Les valeurs calculées par ce noeud, telles que la moyenne ou l'écart-type, peuvent être utilisées lorsque le noeud Tracé est exécuté.

Dans le script de super noeud, vous pouvez spécifier les propriétés du noeud de la même façon que les autres scripts. Ou alors, vous pouvez modifier et définir les propriétés de n'importe quel super noeud ou de ses noeuds encapsulés directement depuis un script de flux. Pour plus d'informations, voir la rubrique Chapitre 19, «Propriétés du super noeud», à la page 303. Cette méthode fonctionne pour les super noeuds de source et de processus ainsi que pour les super noeuds terminaux.

Remarque : puisque seuls les super noeuds terminaux peuvent exécuter leurs propres scripts, l'onglet Scripts de la boîte de dialogue Super noeud est uniquement accessible pour les super noeuds terminaux.

Pour ouvrir la boîte de dialogue de script des super noeuds depuis l'espace de travail principal :

Sélectionnez un super noeud terminal dans l'espace de travail de flux et, dans le menu Super noeud, choisissez :

Script Super noeud...

Pour ouvrir la boîte de dialogue de script des super noeuds depuis l'espace de travail de super noeud en zoom avant :

Cliquez avec le bouton droit de la souris sur l'espace de travail de super noeud et, dans le menu contextuel, sélectionnez :

Script Super noeud...

Exemple de script de super noeud

Le script de super noeud suivant définit l'ordre dans lequel les noeuds terminaux doivent être exécutés à l'intérieur du super noeud. Cet ordre permet que le noeud v. globales soit exécuté en premier afin que les valeurs calculées par ce noeud puissent ensuite être utilisées lorsqu'un autre noeud est exécuté.

```
execute 'Set Globals'
execute 'gains'
execute 'profit'
execute 'age v. $CC-pep'
execute 'Table'
```

Exécution en boucle et conditionnelle dans les flux

A compter de la version 16.0, SPSS Modeler vous permet de créer des scripts de base à partir d'un flux en sélectionnant des valeurs dans diverses boîtes de dialogue au lieu de devoir écrire des instructions directement dans le langage de script. Les deux principaux types de scripts que vous pouvez créer de cette façon sont les boucles simples et l'exécution de noeuds si une condition prédéfinie est remplie.

Vous pouvez combiner les règles d'exécution en boucle et les règles d'exécution conditionnelle dans un flux. Par exemple, vous pouvez avoir des données relatives aux ventes de voitures des fabricants du monde entier. Vous pouvez définir une boucle visant à traiter les données dans un flux, en identifiant les informations détaillées par pays de fabrication, et obtenir des résultats sous forme de graphiques montrant des informations détaillées telles que les volumes de ventes par modèle, les niveaux d'émissions par constructeur et par taille de moteur, etc. Si vous ne souhaitez analyser que les données européennes, vous pouvez aussi ajouter au bouclage des conditions empêchant la création de graphiques pour les constructeurs basées en Amérique et en Asie.

Remarque : Etant donné que l'exécution conditionnelle et l'exécution en boucle sont toutes deux basées sur des scripts d'arrière-plan, elles sont uniquement appliquées à un flux complet lors de son exécution.

- **Bouclage** Vous pouvez utiliser le bouclage (exécution en boucle) pour automatiser les tâches répétitives. Par exemple, cela peut signifier ajouter un nombre de noeuds donné à un flux et modifier un paramètre de noeud à chaque fois. Vous pouvez aussi contrôler l'exécution d'un flux ou créer des branches successives un certain nombre de fois, comme dans les exemples suivants :
  - Exécuter le flux un certain nombre de fois et modifier la source à chaque fois.
  - Exécuter le flux un certain nombre de fois en modifiant la valeur d'une variable à chaque fois.
  - Exécuter le flux un certain nombre de fois en entrant un champ supplémentaire à chaque exécution.
  - Générer un modèle un certain nombre de fois et modifier son paramétrage à chaque fois.
- **Exécution conditionnelle** Cette option vous permet de contrôler l'exécution de noeuds terminaux en fonction de conditions prédéfinies ; voici des exemples :
  - Vous pouvez définir si un noeud sera exécuté ou non en fonction de la valeur true ou false d'une valeur définie.

– Vous pouvez définir si l'exécution en boucle des noeuds sera effectuée en mode parallèle ou séquentiel.

Vous définissez l'exécution en boucle et l'exécution conditionnelle dans l'onglet Exécution de la boîte de dialogue Propriétés de flux. Les noeuds qui sont utilisés dans les exécutions en boucle ou conditionnelles sont affichés associés à un symbole supplémentaire dans le canevas de flux afin d'indiquer qu'ils prennent part à l'exécution en boucle ou conditionnelle.

Vous pouvez accéder à l'onglet Exécution de trois manières :

- En utilisant les menus situés en haut de la boîte de dialogue principale :
  1. Dans le menu Outils, sélectionnez :

     Propriétés du flux > **Exécution**
  2. Cliquez sur l'onglet Exécution pour utiliser les scripts du flux en cours.
- A partir d'un flux :
  1. Cliquez avec le bouton droit de la souris sur un noeud et sélectionnez **Exécution en boucle/conditionnelle**.
  2. Sélectionnez l'option de sous-menu appropriée.
- A partir de la barre d'outils graphiques située en haut de la boîte de dialogue principale, cliquez sur l'icône des propriétés de flux.

Si c'est la première fois que vous définissez les détails d'une exécution en boucle ou conditionnelle, dans l'onglet Exécution, sélectionnez le mode d'exécution **Exécution en boucle/conditionnelle** puis sélectionnez le sous-onglet **Conditionnel** ou **Bouclage**.

Bouclage dans les flux

Le bouclage (ou exécution en boucle) vous permet d'automatiser les tâches répétitives dans les flux ; par exemple :

- Exécuter le flux un certain nombre de fois et modifier la source à chaque fois.
- Exécuter le flux un certain nombre de fois en modifiant la valeur d'une variable à chaque fois.
- Exécuter le flux un certain nombre de fois en entrant un champ supplémentaire à chaque exécution.
- Générer un modèle un certain nombre de fois et modifier son paramétrage à chaque fois.

Vous définissez les conditions à remplir dans le sous-onglet **Bouclage** de l'onglet d'exécution du flux. Pour afficher ce sous-onglet, sélectionnez le mode d'exécution **Exécution en boucle/conditionnelle**.

Les modalités d'exécution en boucle que vous définissez prendront effet lors de l'exécution du flux, si le mode d'exécution **Exécution en boucle/conditionnelle** a été défini. Vous pouvez facultativement générer le code de script des modalités d'exécution en boucle et le coller dans l'éditeur de script en cliquant sur **Coller...** dans l'angle inférieur droit du sous-onglet Bouclage. L'affichage de l'onglet principal Exécution change afin de présenter le mode d'exécution **Par défaut (script facultatif)** avec le script dans la partie supérieure de l'onglet. Cela signifie que vous pouvez définir une structure de bouclage en utilisant les diverses options de la boîte de dialogue de bouclage avant de générer un script que vous pouvez personnaliser ultérieurement dans l'éditeur de script. Notez que lorsque vous cliquez sur **Coller...**, les conditions conditionnelle que vous avez définies s'affichent aussi dans le script généré.

Important : Les variables de bouclage que vous définissez dans un flux SPSS Modeler peuvent être remplacées si vous exécutez le flux dans un travail IBM SPSS Collaboration and Deployment Services. En effet, l'entrée de l'éditeur de travail IBM SPSS Collaboration and Deployment Services se substitue à l'entrée SPSS Modeler. Par exemple, si vous définissez une variable de bouclage dans le flux pour créer un nom du fichier de sortie différent pour chaque boucle, les fichiers sont nommés correctement dans SPSS Modeler mais sont remplacés par l'entrée fixe saisie dans l'onglet Résultat de IBM SPSS Collaboration and Deployment Services Deployment Manager.

Pour définir une boucle

1. Créez une clé d'itération pour définir la structure de bouclage principale à utiliser dans un flux. Pour plus d'informations, voir Créer une clé d'itération.
2. Lorsque cela est nécessaire, définissez une ou plusieurs variables d'itération. Pour plus d'informations, voir Créer une variable d'itération.
3. Les itérations et les variables que vous avez créées sont affichées dans le corps principal du sous-onglet. Par défaut, les itérations sont exécutées dans l'ordre dans lequel elles apparaissent. Pour déplacer une itération vers le haut ou le bas de la liste, cliquez dessus pour la sélectionner puis utilisez les flèches vers le haut ou le bas situées dans la colonne de droite du sous-onglet pour modifier l'ordre.

Création d'une clé d'itération destinée au bouclage dans les flux

Vous utilisez une clé d'itération pour définir la principale structure de bouclage à utiliser dans un flux. Par exemple, si vous analysez des ventes de voitures, vous pouvez créer un paramètre de flux *Pays de fabrication* et l'utiliser comme clé d'itération. Lorsque le flux est exécuté, cette clé est définie successivement sur chaque valeur de pays lors de chaque itération. Utilisez la boîte de dialogue Définir la clé d'itération pour configurer la clé.

Pour ouvrir cette boîte de dialogue, sélectionnez le bouton **Clé d'itération...** dans l'angle inférieur gauche du sous-onglet Bouclage ou cliquez avec le bouton droit de la souris sur un noeud du flux et sélectionnez **Exécution en boucle/conditionnelle** > **Définir la clé d'itération (champs)** ou **Exécution en boucle/conditionnelle** > **Définir la clé d'itération (valeurs)**. Si vous ouvrez la boîte de dialogue à partir du flux, certains des champs peuvent avoir été renseignés automatiquement (par exemple, le nom du noeud).

Pour définir une clé d'itération, renseignez les champs suivants :

Itérer sur. Vous pouvez sélectionner l'une des options suivantes :

- **Paramètre de flux - Champs**. Utilisez cette option pour créer une boucle qui définisse la valeur d'un paramètre de flux existant successivement sur chaque champ spécifié.
- **Paramètre de flux - Valeurs**. Utilisez cette option pour créer une boucle qui définisse la valeur d'un paramètre de flux existant successivement sur chaque valeur spécifiée.
- **Propriété de noeud - Champs**. Utilisez cette option pour créer une boucle qui définisse la valeur d'une propriété de noeud successivement sur chaque champ spécifié.
- **Propriété de noeud - Valeurs**. Utilisez cette option pour créer une boucle qui définisse la valeur d'une propriété de noeud successivement sur chaque valeur spécifiée.

Elément à définir. Sélectionnez l'élément dont la valeur sera définie à chaque exécution de la boucle. Vous pouvez sélectionner l'une des options suivantes :

- **Paramètre**. Uniquement disponible si vous sélectionnez **Paramètre de flux - Champs** ou **Paramètres de flux - Valeurs**. Sélectionnez le paramètre requis dans la liste disponible.
- **Noeud**. Uniquement disponible si vous sélectionnez **Propriété de noeud - Champs** ou **Propriété de noeud - Valeurs**. Sélectionnez le noeud pour lequel vous souhaitez définir une boucle. Cliquez sur le bouton Parcourir pour ouvrir la boîte de dialogue de sélection de noeud et choisissez le noeud souhaité ; si la liste contient trop de noeuds, vous pouvez la filtrer afin de n'afficher que certains types de noeuds en sélectionnant l'une des catégories suivantes : Source, Processus, Graphique, Modélisation, Sortie, Exporter ou Appliquer les noeuds modèle.
- **Propriété**. Uniquement disponible si vous sélectionnez **Propriété de noeud - Champs** ou **Propriété de noeud - Valeurs**. Sélectionnez la propriété du noeud dans la liste disponible.

Champs à utiliser. Uniquement disponible si vous sélectionnez **Paramètre de flux - Champs** ou **Propriété de noeud - Champs**. Sélectionnez le ou les champs (dans un noeud) à utiliser pour fournir les valeurs d'itération. Vous pouvez sélectionner l'une des options suivantes :

- **Noeud**. Uniquement disponible si vous sélectionnez **Paramètre de flux - Champs**. Sélectionnez le noeud qui contient les détails pour lesquels vous souhaitez définir une boucle. Cliquez sur le bouton Parcourir pour ouvrir la boîte de dialogue de sélection de noeud et choisissez le noeud souhaité ; si la liste contient trop de noeuds, vous pouvez la filtrer afin de n'afficher que certains types de noeuds en sélectionnant l'une des catégories suivantes : Source, Processus, Graphique, Modélisation, Sortie, Exporter ou Appliquer les noeuds modèle.
- **Liste de champs**. Cliquez sur le bouton de listage dans la colonne de droite pour afficher la boîte de dialogue Sélectionner les champs, au sein de laquelle vous sélectionnez les champs du noeud qui fourniront les données d'itération. Pour plus d'informations, voir «Sélection de champs pour les itérations», à la page 10.

Valeur à utiliser. Uniquement disponible si vous sélectionnez **Paramètre de flux - Valeurs** ou **Propriété de noeud - Valeurs**. Sélectionnez la ou les valeurs du champ sélectionné à utiliser comme valeurs d'itération. Vous pouvez sélectionner l'une des options suivantes :

- **Noeud**. Uniquement disponible si vous sélectionnez **Paramètre de flux - Valeurs**. Sélectionnez le noeud qui contient les détails pour lesquels vous souhaitez définir une boucle. Cliquez sur le bouton Parcourir pour ouvrir la boîte de dialogue de sélection de noeud et choisissez le noeud souhaité ; si la liste contient trop de noeuds, vous pouvez la filtrer afin de n'afficher que certains types de noeuds en sélectionnant l'une des catégories suivantes : Source, Processus, Graphique, Modélisation, Sortie, Exporter ou Appliquer les noeuds modèle.
- **Liste de champs**. Sélectionnez le champ du noeud qui fournira les données d'itération.
- **Liste de valeurs**. Cliquez sur le bouton de listage dans la colonne de droite pour afficher la boîte de dialogue Sélectionner les valeurs, au sein de laquelle vous sélectionnez les valeurs du noeud qui fourniront les données d'itération.

Création d'une variable d'itération destinée au bouclage dans les flux

Vous pouvez utiliser les variables d'itération pour modifier les valeurs des paramètres ou propriétés de flux des noeuds sélectionnés dans un flux à chaque fois exécution d'une boucle. Par exemple, si la boucle de flux analyse les données de ventes de voitures et utilise *Pays de fabrication* comme clé d'itération, vous pouvez obtenir en sortie un graphique montrant les ventes par modèle et un autre graphique montrant les informations sur les émissions de gaz d'échappement. Dans cet exemple, vous pouvez créer des variables d'itération créant de nouveaux titres pour les graphiques résultants, tels que *Emissions des véhicules suédois* et *Ventes de voitures japonaises par modèle*. Utilisez la boîte de dialogue Définir la variable d'itération pour configurer les variables dont vous avez besoin.

Pour ouvrir cette boîte de dialogue, sélectionnez le bouton **Ajouter une variable...** dans l'angle inférieur gauche du sous-onglet Bouclage ou cliquez avec le bouton droit de la souris sur un noeud du flux et sélectionnez **Exécution en boucle/conditionnelle** > **Définir la variable d'itération**.

Pour définir une variable d'itération, renseignez les champs suivants :

Modifier. Sélectionnez le type d'attribut à modifier. Vous avez le choix entre **Paramètre de flux** et **Propriété de noeud**.

- Si vous sélectionnez **Paramètre de flux**, choisissez le paramètre qui vous convient, puis, à l'aide de l'une des options suivantes (si elle est disponible dans le flux), définissez la valeur que devra avoir le paramètre avec chaque itération de la boucle :
  - **Variable globale**. Sélectionnez la variable globale que le paramètre de flux devra avoir comme valeur.
  - **Cellule de table de résultats**. Pour définir le paramètre de flux qui devra constituer la valeur d'une cellule de table de résultats, sélectionnez la table dans la liste et entrez les **Ligne** et **Colonne** à utiliser.
  - **Saisir manuellement**. Sélectionnez cette option si vous souhaitez entrer manuellement une valeur que ce paramètre devra prendre à chaque itération. Lorsque vous revenez au sous-onglet Bouclage, une nouvelle colonne est créée et vous permet de saisir le texte requis.

- Si vous sélectionnez **Propriété de noeud**, sélectionnez le noeud requis et l'une de ses propriétés, puis définissez la valeur à utiliser pour cette propriété. Définissez la nouvelle valeur de propriété à l'aide de l'une des options suivantes :
  - **Seul**. La valeur de propriété utilisera la valeur de la clé d'itération. Pour plus d'informations, voir «Création d'une clé d'itération destinée au bouclage dans les flux», à la page 8.
  - **Comme préfixe à la racine**. Utilise la valeur de la clé d'itération comme préfixe de ce que vous entrez dans le champ **Racine**.
  - **Comme suffixe à la racine**. Utilise la valeur de la clé d'itération comme suffixe de ce que vous entrez dans le champ **Racine**.

  Si vous sélectionnez l'option de préfixe ou de suffixe, vous êtes invité à ajouter un texte supplémentaire dans le champ **Racine**. Par exemple, si la valeur de la clé d'itération est *Pays de fabrication* et que vous sélectionnez **Comme préfixe à la racine**, vous pouvez entrer - *ventes par modèle* dans ce champ.

Sélection de champs pour les itérations

Lorsque vous créez des itérations, vous pouvez sélectionner un ou plusieurs champs à l'aide de la boîte de dialogue Sélectionner les champs.

Trier par Vous pouvez trier les champs disponibles à afficher en sélectionnant l'une des options suivantes :

- **Naturel** Permet d'afficher l'ordre dans lequel les champs ont été transmis via le flux de données dans le noeud actuel.
- **Nom** Permet de trier les champs à afficher dans l'ordre alphabétique.
- **Type** Permet d'afficher les champs triés par niveau de mesure. Cette option est utile lorsque vous sélectionnez des champs avec un niveau de mesure particulier.

Sélectionnez les champs dans la liste un par un ou utilisez les méthodes Maj+clic et Ctrl+clic pour sélectionner plusieurs champs. Vous pouvez aussi utiliser les boutons situés sous la liste pour sélectionner des groupes de champs en fonction de leur niveau de mesure ou sélectionner ou désélectionner tous les champs de la table.

Notez que les champs disponibles en sélection sont filtrés pour n'afficher que ceux qui conviennent au paramètre de flux ou à la propriété de noeud que vous utilisez. Par exemple, si vous utilisez un paramètre de flux possédant un type de stockage Chaîne, seuls les champs possédant le même type de stockage sont affichés.

Exécution conditionnelle dans les flux

L'exécution conditionnelle vous permet de contrôler l'exécution des noeuds terminaux en définissant des conditions pour le contenu du flux ; voici des exemples :

- Vous pouvez définir si un noeud sera exécuté ou non en fonction de la valeur true ou false d'une valeur définie.
- Vous pouvez définir si l'exécution en boucle des noeuds sera effectuée en mode parallèle ou séquentiel.

Vous définissez les conditions à remplir dans le sous-onglet **Conditionnel** de l'onglet d'exécution du flux. Pour afficher ce sous-onglet, sélectionnez le mode d'exécution **Exécution en boucle/conditionnelle**.

Les modalités d'exécution conditionnelle que vous définissez prendront effet lors de l'exécution du flux, si le mode d'exécution **Exécution en boucle/conditionnelle** a été défini. Vous pouvez facultativement générer le code de script des modalités d'exécution conditionnelle et le coller dans l'éditeur de script en cliquant sur **Coller...** dans l'angle inférieur droit du sous-onglet Conditionnel. L'affichage de l'onglet principal Exécution change afin de présenter le mode d'exécution **Par défaut (script facultatif)** avec le script dans la partie supérieure de l'onglet. Cela signifie que vous pouvez définir les conditions en utilisant les diverses options de la boîte de dialogue de bouclage avant de générer un script que vous

pouvez personnaliser ultérieurement dans l'éditeur de script. Notez que lorsque vous cliquez sur **Coller...**, les conditions de bouclage que vous avez définies s'affichent aussi dans le script généré.

Pour définir une condition :

1. Dans la colonne de droite du sous-onglet Conditionnel, cliquez sur le bouton Ajouter une nouvelle condition  pour ouvrir la boîte de dialogue Ajout d'une instruction d'exécution conditionnelle. Cette boîte de dialogue vous permet de spécifier la condition qui doit être remplie pour que le noeud soit exécuté.
2. Dans la boîte de dialogue Ajout d'une instruction d'exécution conditionnelle, indiquez les informations suivantes :
   a. **Noeud**. Sélectionnez le noeud pour lequel vous souhaitez définir une exécution conditionnelle. Cliquez sur le bouton Parcourir pour ouvrir la boîte de dialogue de sélection de noeud et choisissez le noeud souhaité ; si la liste contient trop de noeuds, vous pouvez la filtrer afin d'afficher les noeuds en fonction de l'une des catégories suivantes : Exporter, Graphique, Modélisation ou Noeud sortie.
   b. **Condition basée sur**. Indiquez la condition qui doit être remplie pour que le noeud soit exécuté. Vous avez le choix entre quatre options : **Paramètre de flux**, **Variable globale**, **Cellule de table de résultats** ou **Toujours vraie**. Les détails que vous entrez dans la moitié inférieure de la boîte de dialogue sont définis par la condition que vous choisissez.
      - **Paramètre de flux**. Sélectionnez le paramètre dans la liste disponible, puis sélectionnez l'**opérateur** de ce paramètre ; par exemple, l'opérateur peut être Supérieur à, Egal, Inférieur à, Entre, etc. Vous entrez ensuite la **valeur** ou les valeurs minimale et maximale, selon l'opérateur.
      - **Variable globale**. Sélectionnez la variable dans la liste disponible ; il peut s'agir par exemple de Moyenne, Somme, Valeur minimale, Valeur maximale ou Ecart type. Sélectionnez ensuite l'**opérateur** et les valeurs requises.
      - **Cellule de table de résultats**. Sélectionnez le noeud table dans la liste disponible, puis sélectionnez la **ligne** et la **colonne** dans la table. Sélectionnez ensuite l'**opérateur** et les valeurs requises.
      - **Toujours vraie**. Sélectionnez cette option si le noeud doit toujours être exécuté. Si vous sélectionnez cette option, il n'y a aucun autre paramètre à sélectionner.
3. Répétez les étapes 1 et 2 aussi souvent que nécessaire jusqu'à ce que vous ayez configuré toutes les conditions souhaitées. Le noeud que vous avez sélectionné et la condition à remplir pour que le noeud soit exécuté sont affichés dans le corps principal du sous-onglet dans les colonnes **Exécuter un noeud** et **Si cette condition est vraie**.
4. Par défaut, les noeuds et conditions sont exécutés dans l'ordre dans lequel ils apparaissent. Pour déplacer un noeud et une condition vers le haut ou le bas de la liste, cliquez dessus pour le sélectionner puis utilisez les flèches vers le haut ou le bas situées dans la colonne de droite du sous-onglet pour modifier l'ordre.

En outre, vous pouvez définir les options suivantes au bas du sous-onglet Conditionnel :

- **Evaluer tout dans l'ordre**. Sélectionnez cette option pour évaluer les conditions dans l'ordre dans lequel elles apparaissent dans le sous-onglet. Les noeuds pour lesquels les conditions ont la valeur "True" sont tous exécutés une fois toutes les conditions évaluées.
- **Exécuter un à un**. Uniquement disponible si **Evaluer tout dans l'ordre** est sélectionné. Lorsque cette option est sélectionnée, si une condition est évaluée comme ayant la valeur "True", le noeud associé à cette condition est exécuté avant que la condition suivante ne soit évaluée.
- **Evaluer jusqu'au premier résultat**. Lorsque vous sélectionnez cette option, seul le premier noeud qui renvoie une évaluation "True" à partir des conditions spécifiées s'exécute.

Exécution et interruption de scripts

Plusieurs manières d'exécuter les scripts sont disponibles. Par exemple, dans la boîte de dialogue du script de flux ou du script autonome, le bouton « Exécuter ce script » permet d'exécuter l'intégralité du script :

Figure 1. Bouton Exécuter ce script

Le bouton « Exécuter les lignes sélectionnées » exécute une ligne unique ou un bloc de lignes adjacentes que vous avez sélectionnées dans le script :

Figure 2. Bouton Exécuter les lignes sélectionnées

Pour exécuter un script, utilisez l'une des méthodes suivantes :

- Cliquer sur le bouton « Exécuter ce script » ou sur le bouton « Exécuter les lignes sélectionnées » dans la boîte de dialogue du script de flux ou du script autonome.
- Exécutez un flux dans lequel la méthode d'exécution par défaut est **Exécuter ce script**.
- Utilisez l'indicateur `-execute` au démarrage en mode interactif. Pour plus d'informations, voir la rubrique «Utilisation d'arguments de ligne de commande», à la page 61.

Remarque : un script associé à un super noeud est exécuté en même temps que ce dernier si vous avez sélectionné **Exécuter ce script** dans la boîte de dialogue de script du super noeud.

Interruption de l'exécution d'un script

Dans la boîte de dialogue du script de flux, le bouton rouge d'arrêt est activé pendant l'exécution du script. Ce bouton permet d'abandonner l'exécution du script et de tout flux actuel.

Rechercher et remplacer

La boîte de dialogue Rechercher/Remplacer est disponible aux emplacements où vous modifier un script ou le texte d'une expression, par exemple l'éditeur de script ou le générateur d'expression CLEM, ou encore lors de la définition d'un modèle dans le noeud Rapport. Lorsque vous éditez du texte dans l'un de ces champs, appuyez sur `Ctrl+F` pour accéder à la boîte de dialogue, en vous assurant que le curseur est centré sur une zone de texte. Si vous travaillez dans un noeud Remplacer, par exemple, vous pouvez accéder à la boîte de dialogue depuis toute zone de texte de l'onglet Paramètres, ou depuis le champ de texte du générateur d'expression.

1. Lorsque le curseur se trouve sur une zone de texte, appuyez sur Ctrl+F pour accéder à la boîte de dialogue Rechercher/Remplacer.
2. Entrez le texte que vous souhaitez rechercher ou faites un choix dans la liste déroulante des éléments récemment consultés.
3. Saisissez éventuellement un texte de remplacement.
4. Cliquez sur **Suivant** pour lancer la recherche.
5. Cliquez sur **Remplacer** pour remplacer la sélection courante, ou **Remplacer tout** pour mettre à jour certaines instances ou les instances sélectionnées.

6. La boîte de dialogue se ferme après chaque opération. Appuyez sur F3 depuis toute zone de texte pour répéter la dernière opération de recherche ou sur Ctrl+F pour accéder de nouveau à la boîte de dialogue.

Options de recherche

Respecter la casse. Spécifie si l'opération de recherche est sensible à la casse ; par exemple, si *myvar* correspond à *myVar*. Le texte de remplacement est toujours inséré exactement tel qu'il a été saisi, quel que soit le réglage de ce paramètre.

Mot entier. Spécifie si l'opération de recherche doit porter sur le texte inséré dans des mots. Si cette option est sélectionnée, par exemple, une recherche portant sur *spider* ne produira pas la réponse *spiderman* ou *spider-man*.

Caractères génériques. Spécifie si la syntaxe des caractères génériques est utilisée (voir la section suivante). Lorsqu'elle est sélectionnée, l'option **Mot entier** est désactivée et sa valeur est ignorée.

Texte sélectionné. Contrôle la portée de la recherche lorsque vous utilisez l'option **Remplacer tout**.

Syntaxe des caractères génériques

Les caractères génériques vous permettent de rechercher des caractères spéciaux tels que les tabulations ou les sauts de ligne, des classes ou des intervalles de caractères telles que *a* à *d*, toute valeur numérique ou non et les limites telles que le début ou la fin d'une ligne. Les types d'expression suivants sont pris en charge :

Tableau 1. Correspondances de caractères.

Caractères	Correspondances
x	Le caractère x
\\	Le caractère barre oblique inversée
\0n	Le caractère présentant la valeur octale 0n (0 <= n <= 7)
\0nn	Le caractère présentant la valeur octale 0nn (0 <= n <= 7)
\0mnn	Le caractère présentant la valeur octale 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	Le caractère présentant la valeur hexadécimale 0xhh
\uhhhh	Le caractère présentant la valeur hexadécimale 0xhhhh
\t	Le caractère tabulation ('\u0009')
\n	Le caractère saut de ligne (retour à la ligne) ('\u000A')
\r	Le caractère retour chariot ('\000D')
\f	Le caractère alimentation de formulaire ('\u000C')
\a	Le caractère alerte (sonnerie) ('\u0007')
\e	Le caractère d'échappement('\u001B')
\cx	Le caractère de contrôle correspondant à x

Tableau 2. Classes de caractères correspondantes.

Classes de caractères	Correspondances
[abc]	a, b ou c (classe simple)
[^abc]	Tout caractère excepté a, b ou c (soustraction)
[a-zA-Z]	Caractères de a à z ou de A à Z compris (intervalle)

Tableau 2. Classes de caractères correspondantes (suite).

Classes de caractères	Correspondances
[a-d[m-p]]	Caractères de a à d ou de m à p (union). Cette option peut aussi être spécifiée comme [a-dm-p]
[a-z&&[def]]	Caractères de a à z ainsi que d, e ou f (intersection)
[a-z&&[^bc]]	Caractères de a à z à l'exception de b et c (soustraction). Cette option peut aussi être spécifiée comme [ad-z]
[a-z&&[^m-p]]	Caractères de a à z, mais pas de m à p (soustraction). Cette option peut aussi être spécifiée comme [a-lq-z]

Tableau 3. Classes de caractères prédéfinies.

Classes de caractères prédéfinies	Correspondances
.	Tout caractère (peut correspondre ou non aux terminaisons de ligne)
\d	Tout chiffre : [0-9]
\D	Un caractère non numérique : [^0-9]
\s	Un espace blanc : [\t\n\x0B\f\r]
\S	Un espace non blanc : [^\s]
\w	Un mot : [a-zA-Z_0-9]
\W	Un caractère autre qu'un mot : [^\w]

Tableau 4. Correspondances de limites.

Correspondances de limites	Correspondances
^	Le début d'une ligne
$	La fin d'une ligne
\b	Une limite de mot
\B	Une limite autre que celle d'un mot
\A	Le début de la saisie
\Z	La fin de la saisie à l'exception de la terminaison finale éventuelle
\z	La fin de la saisie

Chapitre 2. Langage de script

Présentation du langage de script

La fonction de scriptage (génération de scripts) pour IBM SPSS Modeler vous permet de créer des scripts qui fonctionnent sur l'interface utilisateur SPSS Modeler, manipulent des objets de sortie et exécutent la syntaxe de commande. Vous pouvez exécuter des scripts directement depuis SPSS Modeler.

Les scripts d'IBM SPSS Modeler sont écrits dans le langage de script Python. L'implémentation Java de Python qui est utilisée par IBM SPSS Modeler est appelée Jython. Le langage de script est composé des fonctions suivantes :

- un format pour le référencement des noeuds, des flux, des projets, des sorties, ainsi que d'autres objets IBM SPSS Modeler ;
- Un ensemble d'instructions ou de commandes de script qui permettent de manipuler ces objets.
- Un langage d'expression de script pour le paramétrage des valeurs des variables, paramètres et autres objets.
- Une prise en charge des commentaires, des lignes incomplètes et des blocs de texte littéral.

Les sections suivantes décrivent le langage de script Python, l'implémentation Jython de Python et la syntaxe de base pour s'initier au scriptage dans IBM SPSS Modeler. Des informations sur les propriétés et commandes spécifiques vous sont fournies dans les sections suivantes.

Python et Jython

Jython est une implémentation du langage de script Python, qui est écrit dans le langage Java et intégré à la plateforme Java. Python est un puissant langage de script orienté objet. Jython est utile car il offre les fonctions de productivité d'un langage de script abouti et, contrairement à Python, s'exécute dans tout environnement prenant en charge une machine virtuelle Java (JVM). Ainsi, les bibliothèques Java de la machine virtuelle Java peuvent être utilisées lorsque vous écrivez des programmes. Avec Jython, vous pouvez tirer profit de cette différence et utiliser la syntaxe ainsi que la plupart des fonctions du langage Python.

En tant que langage de script, Python (et son implémentation Jython) est facile à comprendre, permet un codage efficace et ne requiert qu'une structure minimale pour créer un programme d'exécution. Le code peut être saisi de manière interactive, c'est-à-dire une ligne à la fois. Python est un langage de script interprété ; il n'existe aucune étape de précompilation, comme c'est le cas dans Java. Les programmes Python sont simplement des fichiers textes interprétés lors de leur entrée (après l'analyse syntaxique recherchant les erreurs de syntaxe). Les expressions simples, telles que les valeurs définies, ainsi que les actions plus complexes, telles que les définitions de fonctions, sont immédiatement exécutées et utilisables. Les modifications apportées au code peuvent être testées rapidement. Toutefois, l'interprétation de script présente certains inconvénients. Par exemple, l'utilisation d'une variable non définie n'est pas une erreur de compilateur. Ainsi, elle est détectée uniquement si (et lorsque) l'instruction dans laquelle est utilisée la variable est exécutée. Dans ce cas, le programme peut être modifié et exécuté pour déboguer l'erreur.

Python perçoit tout, y compris l'ensemble du code et des données, comme un objet. Par conséquent, vous pouvez manipuler ces objets avec des lignes de code. Certains types, tels que les nombres et les chaînes, sont plus aisément considérés comme des valeurs, et non comme des objets ; Python assure cette prise en charge. Une seule valeur nulle est prise en charge. Cette valeur nulle porte le nom réservé `None` (aucun).

Pour une introduction plus approfondie des scripts Python et Jython et pour obtenir des exemples de scripts, voir http://www.ibm.com/developerworks/java/tutorials/j-jython1 et http://www.ibm.com/developerworks/java/tutorials/j-jython2.

Scripts Python

Ce guide relatif au langage de script Python est une introduction aux composants les plus susceptibles d'être utilisés lors de la génération de scripts (scriptage) dans IBM SPSS Modeler, notamment les concepts et les notions de base de la programmation. Ce guide vous apportera les connaissances nécessaires pour commencer à développer vos propres scripts Python à utiliser dans IBM SPSS Modeler.

Opérations

L'affectation s'effectue en utilisant le signe égal (=). Par exemple, pour affecter la valeur "3" à une variable appelée "x", utilisez l'instruction suivante :

```
x = 3
```

Le signe égal permet également d'affecter des données de type chaîne à une variable. Par exemple, pour affecter la valeur "a string value" à la variable "y", utilisez l'instruction suivante :

```
y = "a string value"
```

Le tableau suivant répertorie certaines des opérations numériques et de comparaison courantes ainsi que leurs descriptions.

Tableau 5. Opérations numériques et de comparaison courantes

Opération	Description
`x < y`	x est-il inférieur à y ?
`x > y`	x est-il supérieur à y ?
`x <= y`	x est-il inférieur ou égal à y ?
`x >= y`	x est-il supérieur ou égal à y ?
`x == y`	x est-il égal à y ?
`x != y`	x est-il différent de y ?
`x <> y`	x est-il différent de y ?
`x + y`	Ajoute y à x
`x - y`	Soustrait y de x
`x * y`	Multiplie x par y
`x / y`	Divise x par y
`x ** y`	Elève x à la puissance y

Listes

Les listes sont des séquences d'éléments. Une liste peut contenir tout nombre d'éléments, et les éléments de la liste peuvent correspondre à tout type d'objet. Les listes peuvent également être considérées comme des tableaux. Le nombre d'éléments dans une liste peut augmenter ou diminuer au fur et à mesure que des éléments sont ajoutés, supprimés ou remplacés.

Exemples

`[]`	Une liste vide.
`[1]`	Une liste avec un seul élément : un entier.

`["Mike", 10, "Don", 20]`	Une liste avec quatre éléments : deux éléments chaînes et deux éléments entiers.
`[[],[7],[8,9]]`	Une liste de listes. Chaque sous-liste est soit une liste vide soit une liste d'éléments entiers.
`x = 7; y = 2; z = 3;` `[1, x, y, x + y]`	Une liste d'entiers. Cet exemple illustre l'utilisation de variables et d'expressions.

Vous pouvez affecter une liste à une variable, par exemple :

```
mylist1 = ["one", "two", "three"]
```

Vous pouvez ensuite accéder à des éléments spécifiques de la liste, par exemple :

```
mylist[0]
```

La sortie suivante est générée :

```
one
```

Le nombre entre crochets (`[]`) est appelé *index* et renvoie à un élément particulier de la liste. Les éléments d'une liste sont indexés à partir de 0.

Vous pouvez également sélectionner une plage d'éléments d'une liste ; on appelle cela le *tranchage* (slicing). Par exemple, `x[1:3]` sélectionne les deuxième et troisième éléments de `x`. L'index de fin est celui après la sélection.

Chaînes

Une *chaîne* est une séquence de caractères non modifiable considérée comme une valeur. Les chaînes prennent en charge l'ensemble des fonctions et opérateurs de séquence non modifiable qui donnent lieu à une nouvelle chaîne. Par exemple, `"abcdef"[1:4]` génère la sortie `"bcd"`.

En Python, les caractères sont représentés par des chaînes de longueur un.

Les littéraux de chaîne sont définis par l'utilisation de guillemets simples ou triples. Les chaînes définies à l'aide de guillemets simples ne peuvent pas s'étendre sur plusieurs lignes, alors que les chaînes définies à l'aide de guillemets triples le peuvent. Une chaîne peut figurer entre guillemets simples (`'`) ou guillemets doubles (`"`). Un type de guillemet peut contenir l'autre type de guillemet sans caractère d'échappement, ou le même guillemet avec caractère d'échappement, c'est-à-dire précédé de la barre oblique inversée (\).

Exemples

```
"This is a string"
'This is also a string'
"It's a string"
'This book is called "Python Scripting and Automation Guide".'
"This is an escape quote (\") in a quoted string"
```

Plusieurs chaînes séparées par un blanc sont automatiquement concaténées par l'analyseur Python. Cela facilite la saisie des chaînes longues et permet de mélanger les types de guillemets dans une même chaîne. Par exemple :

```
"This string uses ' and " 'that string uses ".'
```

La sortie suivante est générée :

```
This string uses ' and that string uses ".
```

Les chaînes prennent en charge plusieurs méthodes utiles. Certaines de ces méthodes sont présentées dans le tableau suivant.

Tableau 6. Méthodes de chaîne

Méthode	Utilisation
`s.capitalize()`	Majuscule initiale pour `s`
`s.count(ss {,start {,end}})`	Compte les occurrences de `ss` dans `s[start:end]`
`s.startswith(str {, start {, end}})` `s.endswith(str {, start {, end}})`	Test pour savoir si `s` commence par `str` Test pour savoir si `s` se termine par `str`
`s.expandtabs({size})`	Remplace les tabulations par des espaces, la taille (`size`) par défaut est 8
`s.find(str {, start {, end}})` `s.rfind(str {, start {, end}})`	Recherche le premier index de `str` dans `s` ; s'il est introuvable, le résultat est -1. `rfind` recherche de droite à gauche.
`s.index(str {, start {, end}})` `s.rindex(str {, start {, end}})`	Recherche le premier index de `str` dans `s` ; s'il est introuvable, renvoie l'erreur `ValueError`. `rindex` recherche de droite à gauche.
`s.isalnum`	Test pour savoir si la chaîne est alphanumérique
`s.isalpha`	Test pour savoir si la chaîne est alphabétique
`s.isnum`	Test pour savoir si la chaîne est numérique
`s.isupper`	Test pour savoir si la chaîne est tout en majuscules
`s.islower`	Test pour savoir si la chaîne est tout en minuscules
`s.isspace`	Test pour savoir si la chaîne n'est constituée que de blancs
`s.istitle`	Test pour savoir si la chaîne est une séquence de chaînes alphanumériques avec majuscule initiale

`s.lower()`
`s.upper()`
`s.swapcase()`
`s.title()`
	Convertit tout en minuscules
	Convertit tout en majuscules
	Convertit tout dans la casse opposée
	Convertit tout en casse titre

`s.join(seq)`	Joint les chaînes dans `seq` avec `s` comme séparateur
`s.splitlines({keep})`	Fractionne `s` en lignes, si keep a la valeur `true`, conserve les nouvelles lignes
`s.split({sep {, max}})`	Fractionne `s` en "mots" avec `sep` (par défaut, `sep` est un blanc), le nombre de fois indiqué par `max`

`s.ljust(width)`
 	
`s.rjust(width)`
 	
`s.center(width)`
`s.zfill(width)`
	Justifie à gauche la chaîne dans un champ de largeur `width`
	Justifie à droite la chaîne dans un champ de largeur `width`
	Centre la chaîne dans un champ de largeur `width`
	Remplit avec 0.

`s.lstrip()`
`s.rstrip()`
`s.strip()`
	Supprime l'espace blanc de début
	Supprime l'espace blanc de fin
	Supprime les espaces blancs de début et de fin

`s.translate(str {,delc})`	Convertit `s` à l'aide de la table, après avoir supprimé les caractères de `delc`. `str` doit être une chaîne de longueur `== 256`.
`s.replace(old, new {, max})`	Remplace toutes les occurrences (ou le nombre d'occurrences indiqué par `max`) de la chaîne `old` par la chaîne `new`

Remarques

Les remarques sont des commentaires qui commencent par le signe dièse (#). L'ensemble du texte qui suit le signe dièse sur la même ligne est considéré comme appartenant à la remarque et est ignoré. Une remarque peut commencer dans n'importe quelle colonne. L'exemple suivant illustre l'utilisation des remarques :

```
#The HelloWorld application is one of the most simple
print 'Hello World' # print the Hello World line
```

Syntaxe des instructions

La syntaxe des instructions pour Python est très simple. En général, chaque ligne source correspond à une instruction unique. A l'exception des instructions `expression` et `assignment`, chaque instruction est introduite par un nom de mot-clé, tel que `if` ou `for`. Des lignes vierges ou des lignes de remarques peuvent être insérées partout entre les instructions du code. Si une ligne comporte plusieurs instructions, les instructions doivent être séparées par un point-virgule (`;`).

Les très longues instructions peuvent s'étendre sur plusieurs lignes. Dans ce cas, l'instruction qui doit se poursuivre sur la ligne suivante doit se terminer par une barre oblique inversée (\), par exemple :

```
x = "A looooooooooooooooooong string" + \
     "another looooooooooooooooooong string"
```

Lorsqu'une structure figure entre parenthèses (()), crochets ([]) ou accolades ({}), l'instruction peut se poursuivre sur une nouvelle ligne après toute virgule, sans nécessiter l'insertion d'une barre oblique inversée, par exemple :

```
x = (1, 2, 3, "hello",
      "goodbye", 4, 5, 6)
```

Identificateurs

Les identificateurs sont utilisés pour nommer des variables, des fonctions, des classes et des mots-clés. Les identificateurs peuvent être de n'importe quelle longueur, mais doivent commencer soit par un caractère alphabétique en majuscule ou en minuscule, soit par le caractère de soulignement (_). Les noms commençant par un trait de soulignement sont généralement réservés aux noms internes ou privés. Au delà du premier caractère, l'identificateur peut contenir n'importe quel nombre et n'importe quelle combinaison de caractères alphabétiques, nombres de 0 à 9, et le caractère de soulignement.

Il existe des mots réservés dans Jython qui ne peuvent pas être utilisés pour nommer les variables, les fonctions ou les classes. Ils appartiennent aux catégories suivantes :

- **Préfixes d'instruction :** `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `pass`, `print`, `raise`, `return`, `try` et `while`
- **Préfixes de paramètre :** `as`, `import` et `in`
- **Opérateurs :** `and`, `in`, `is`, `lambda`, `not` et `or`

L'utilisation d'un mot-clé incorrect entraîne généralement une erreur `SyntaxError`.

Blocs de code

Les blocs de code sont des groupes d'instructions utilisés où des instructions uniques sont attendues. Les blocs de code peuvent respecter les instructions suivantes : `if`, `elif`, `else`, `for`, `while`, `try`, `except`, `def` et `class`. Ces instructions introduisent le bloc de code avec le caractère deux-points (`:`), par exemple :

```
if x == 1:
    y = 2
    z = 3
elif:
    y = 4
    z = 5
```

L'indentation (mise en retrait) est utilisée pour délimiter les blocs de code (plutôt que les accolades utilisées dans Java). Toutes les lignes d'un bloc de code doivent être mises en retrait à la même position. En effet, un changement d'indentation indique la fin d'un bloc de code. Il est courant d'effectuer une mise en retrait de quatre espaces par niveau. Il est recommandé d'utiliser des espaces plutôt que des tabulations pour mettre en retrait les lignes. Vous ne devez pas mélanger les espaces et les tabulations. Les lignes du bloc le plus vers l'extérieur d'un module doivent commencer à la colonne un ; sinon, une erreur `SyntaxError` survient.

Les instructions qui constituent un bloc de code (et suivent le caractère deux-points) peuvent également figurer sur une seule ligne et être séparées par des points-virgules, par exemple :

```
if x == 1: y = 2; z = 3;
```

Transmission d'arguments à un script

La transmission d'arguments à un script est utile car cela signifie qu'un script peut être utilisé de manière répétée sans modification. Les arguments transmis sur la ligne de commande sont transmis en tant que valeurs dans la liste `sys.argv`. Vous pouvez obtenir le nombre de valeurs transmises à l'aide de la commande `len(sys.argv)`. Par exemple :

```
import sys
print "test1"
print sys.argv[0]
print sys.argv[1]
print len(sys.argv)
```

Dans cet exemple, la commande `import` importe toute la classe `sys` de manière à ce que les méthodes qui existent pour cette classe, telles que `argv`, puissent être utilisées.

Le script de cet exemple peut être appelé à l'aide de la ligne suivante :

```
/u/mjloos/test1 mike don
```

La sortie suivante est générée :

```
/u/mjloos/test1 mike don
test1
mike
don
3
```

Exemples

Le mot-clé `print` imprime les arguments qui le suivent immédiatement. Si l'instruction est suivie d'une virgule, aucune nouvelle ligne n'est incluse dans la sortie. Par exemple :

```
print "This demonstrates the use of a",
print " comma at the end of a print statement."
```

La sortie suivante est générée :

```
This demonstrates the use of a comma at the end of a print statement.
```

L'instruction `for` est utilisée pour itérer un bloc de code. Par exemple :

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    print lv
    continue
```

Dans cet exemple, trois chaînes sont affectées à la liste `mylist1`. Les éléments de la liste sont ensuite imprimés, avec un élément de chaque ligne. La sortie suivante est générée :

```
one
two
three
```

Dans cet exemple, l'itérateur `lv` prend la valeur de chaque élément dans la liste `mylist1` tour à tour tandis que la boucle for implémente le bloc de code pour chaque élément. Un itérateur peut être n'importe quel identificateur valide, de n'importe quelle longueur.

L'instruction `if` est une instruction conditionnelle. Elle évalue la condition et renvoie true ou false, selon le résultat de l'évaluation. Par exemple :

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    if lv == "two"
        print "The value of lv is ", lv
    else
        print "The value of lv is not two, but ", lv
    continue
```

Dans cet exemple, la valeur de l'itérateur `lv` est évaluée. Si la valeur de `lv` est `two`, la chaîne renvoyée est différente de la chaîne renvoyée si la valeur de `lv` n'est pas `two`. La sortie suivante est générée :

```
The value of lv is not two, but one
The value of lv is two
The value of lv is not two, but three
```

Méthodes mathématiques

A partir du module `math`, vous avez accès à des méthodes mathématiques utiles. Certaines de ces méthodes sont présentées dans le tableau suivant. Sauf indication contraire, toutes les valeurs sont renvoyées en tant que variables flottantes.

Tableau 7. Méthodes mathématiques

Méthode	Utilisation
`math.ceil(x)`	Renvoie la partie entière par excès de `x` en tant que variable flottante, c'est-à-dire le plus petit entier supérieur ou égal à `x`.
`math.copysign(x, y)`	Renvoie `x` avec le signe de `y`. `copysign(1, -0.0)` renvoie `-1`.
`math.fabs(x)`	Renvoie la valeur absolue de `x`.
`math.factorial(x)`	Renvoie la factorielle de `x`. Si `x` est négatif ou n'est pas un entier, une erreur `ValueError` est renvoyée.
`math.floor(x)`	Renvoie la partie entière par défaut de `x` en tant que variable flottante, c'est-à-dire le plus grand entier inférieur ou égal à `x`.
`math.frexp(x)`	Renvoie la mantisse (`m`) et l'exposant (`e`) de `x` en tant que paire `(m, e)`. `m` est une variable flottante et `e` est un entier, tel que `x == m * 2**e` exactement. Si `x` est égal à zéro, renvoie `(0.0, 0)`, sinon `0.5 <= abs(m) < 1`.
`math.fsum(iterable)`	Renvoie une somme de valeurs à virgule flottante précise dans `iterable`.
`math.isinf(x)`	Vérifie si la variable flottante `x` est un infinitif positif ou négatif.
`math.isnan(x)`	Vérifie si la variable flottante `x` n'est pas un nombre, à savoir `NaN` (not a number).
`math.ldexp(x, i)`	Renvoie `x * (2**i)`. Il s'agit pour l'essentiel de l'inverse de la fonction `frexp`.
`math.modf(x)`	Renvoie les parties fractionnelle et entière de `x`. Les deux résultats portent le signe de `x` et sont des variables flottantes.

Tableau 7. Méthodes mathématiques (suite)

Méthode	Utilisation
`math.trunc(x)`	Renvoie la valeur réelle (`Real`) `x`, qui a été tronquée en intégrale (`Integral`).
`math.exp(x)`	Renvoie `e**x`.
`math.log(x[, base])`	Renvoie le logarithme de `x` à la valeur donnée de `base`. Si `base` n'est pas spécifié, le logarithme naturel de `x` est renvoyé.
`math.log1p(x)`	Renvoie le logarithme naturel de `1+x` (`base e`).
`math.log10(x)`	Renvoie le logarithme de base 10 de `x`.
`math.pow(x, y)`	Renvoie `x` élevé à la puissance `y`. `pow(1.0, x)` et `pow(x, 0.0)` renvoient toujours 1, même lorsque `x` est égal à zéro ou n'est pas un nombre (NaN).
`math.sqrt(x)`	Renvoie la racine carrée de `x`.

Outre les fonctions mathématiques, il existe certaines méthodes trigonométriques utiles. Ces méthodes sont présentées dans le tableau suivant.

Tableau 8. Méthodes trigonométriques

Méthode	Utilisation
`math.acos(x)`	Renvoie le cosinus inverse de `x` en radians.
`math.asin(x)`	Renvoie le sinus inverse de `x` en radians.
`math.atan(x)`	Renvoie la tangente inverse de `x` en radians.
`math.atan2(y, x)`	Renvoie `atan(y / x)` en radians.
`math.cos(x)`	Renvoie le cosinus de `x` en radians.
`math.hypot(x, y)`	Renvoie la norme euclidienne `sqrt(x*x + y*y)`. Il s'agit de la longueur du vecteur de l'origine jusqu'au point `(x, y)`.
`math.sin(x)`	Renvoie le sinus de `x` en radians.
`math.tan(x)`	Renvoie la tangente de `x` en radians.
`math.degrees(x)`	Convertit l'angle `x` de radians en degrés.
`math.radians(x)`	Convertit l'angle `x` de degrés en radians.
`math.acosh(x)`	Renvoie le cosinus hyperbolique inverse de `x`.
`math.asinh(x)`	Renvoie le sinus hyperbolique inverse de `x`.
`math.atanh(x)`	Renvoie la tangente hyperbolique inverse de `x`.
`math.cosh(x)`	Renvoie le cosinus hyperbolique de `x`.
`math.sinh(x)`	Renvoie le sinus hyperbolique de `x`.
`math.tanh(x)`	Renvoie la tangente hyperbolique de `x`.

Il existe également deux constantes mathématiques. La valeur de `math.pi` est la constante mathématique pi. La valeur de `math.e` est la constante mathématique e.

Utilisation des caractères non ASCII

Pour pouvoir utiliser des caractères non ASCII, Python requiert un codage et un décodage explicite des chaînes en Unicode. Dans IBM SPSS Modeler, les scripts Python sont supposés être codés en UTF-8, ce qui est un codage Unicode standard prenant en charge les caractères non ASCII. Le script suivant peut être compilé car le compilateur Python a été configuré en UTF-8 par SPSS Modeler.

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "テストノード", 96, 64)
```

Le noeud résultant aura un libellé incorrect.

Figure 3. Libellé de noeud contenant des caractères non ASCII affiché incorrectement

Le libellé est incorrect car le littéral chaîne a lui-même été converti en une chaîne ASCII par Python.

Python permet de spécifier des littéraux chaîne Unicode en ajoutant un préfixe `u` devant le littéral chaîne :

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", u"テストノード", 96, 64)
```

Une chaîne Unicode est créée et le libellé s'affiche correctement.

Figure 4. Libellé de noeud contenant des caractères non ASCII affiché correctement

L'utilisation de Python et de l'Unicode est un vaste sujet qui dépasse la portée du présent document. De nombreux ouvrages et de nombreuses ressources en lignes traitant en détail de ce sujet sont à votre disposition.

Programmation orientée objet

La programmation orientée objet repose sur la notion de création d'un modèle du problème cible dans vos programmes. La programmation orientée objet réduit les erreurs de programmation et favorise la réutilisation du code. Python est un langage orienté objet. Les objets définis dans Python sont dotés des caractéristiques suivantes :

- **Identité :** Chaque objet doit être distinct et vous devez pouvoir le tester. A cet effet, il existe des tests `is` et `is not`.

- **Etat :** Chaque objet doit pouvoir stocker un état. Des attributs, tels que les champs et les variables d'instance, existent à cet effet.
- **Comportement :** Chaque objet doit pouvoir manipuler son état. Il existe des méthodes pour cela.

Python inclut les fonctions suivantes pour prendre en charge la programmation orientée objet :

- **Création d'objet basée sur les classes :** Les classes sont des modèles pour la création d'objets. Les objets sont des structures de données avec le comportement associé.
- **Héritage avec polymorphisme :** Python prend en charge l'héritage unique et l'héritage multiple. Toutes les méthodes d'instance Python sont polymorphes et peuvent être remplacées par des sous-classes.
- **Encapsulation avec masquage des données :** Python permet de masquer les attributs. Lorsque les attributs sont masqués, ils sont accessibles depuis l'extérieur de la classe uniquement via des méthodes de la classe. Les classes implémentent des méthodes pour modifier les données.

Définition d'une classe

Dans une classe Python, les variables et les méthodes peuvent être définies. A la différence de Java, avec Python, vous pouvez définir n'importe quel nombre de classes publiques par fichier source (ou *module*). Par conséquent, on peut considérer un module dans Python comme semblable à un package dans Java.

Dans Python, les classes sont définies à l'aide de l'instruction `class`. L'instruction `class` présente la forme suivante :

```
class name (superclasses): statement
```

ou

```
class name (superclasses):
    assignment
    .
    .
    function
    .
    .
```

Lorsque vous définissez une classe, vous avez la possibilité de fournir plusieurs instructions d'*affectation* ou aucune. Ces dernières créent des attributs de classe partagés par toutes les instances de la classe. Vous pouvez également fournir plusieurs définitions de *fonction* ou aucune. Ces définitions de fonction créent des méthodes. La liste de superclasses est facultative.

Le nom de classe doit être unique dans un même périmètre, c'est-à-dire au sein d'un module, d'une fonction ou d'une classe. Vous pouvez définir plusieurs variables pour faire référence à la même classe.

Création d'une instance de classe

Les classes permettent de stocker des attributs de classe (ou partagés) ou de créer des instances de classe. Pour créer une instance de classe, vous appelez la classe comme s'il s'agissait d'une fonction. Prenons l'exemple de la classe suivante :

```
class MyClass:
    pass
```

Dans ce cas, l'instruction `pass` est utilisée car une instruction est requise pour terminer la classe, mais aucune action à l'aide d'un programme n'est nécessaire.

L'instruction suivante crée une instance de la classe `MyClass` :

```
x = MyClass()
```

Ajout d'attributs à une instance de classe

A la différence de Java, avec Python, les clients peuvent ajouter des attributs à l'instance d'une classe. Une seule instance est modifiée. Par exemple, pour ajouter des attributs à une instance `x`, définissez de nouvelles valeurs sur cette instance :

```
x.attr1 = 1
x.attr2 = 2
    .
    .
x.attrN = n
```

Définition d'attributs de classe et de méthodes

Toute variable liée à une classe est un *attribut de classe*. Toute fonction définie dans une classe est une *méthode*. Les méthodes reçoivent une instance de la classe, appelée de manière conventionnelle `self`, comme premier argument. Par exemple, pour définir certains attributs de classe et méthodes, vous pouvez saisir le code suivant :

```
class MyClass
    attr1 = 10        #class attributes
    attr2 = "hello"

    def method1(self):
        print MyClass.attr1   #reference the class attribute

    def method2(self):
        print MyClass.attr2   #reference the class attribute

    def method3(self, text):
        self.text = text        #instance attribute
        print text, self.text   #print my argument and my attribute

    method4 = method3   #make an alias for method3
```

Au sein d'une classe, vous devez qualifier toutes les références à des attributs de classe avec le nom de classe ; par exemple, `MyClass.attr1`. Toutes les références à des attributs d'instance doivent être qualifiées avec la variable `self` ; par exemple, `self.text`. En dehors de la classe, vous devez qualifier toutes les références à des attributs de classe avec le nom de classe (par exemple `MyClass.attr1`) ou avec une instance de la classe (par exemple `x.attr1`, où `x` est une instance de la classe). En dehors de la classe, vous devez qualifier toutes les références à des variables d'instance avec une instance de la classe ; par exemple, `x.text`.

Variables masquées

Vous pouvez masquer des données en créant des variables *Private* (privées). Les variables privées sont accessibles uniquement par la classe elle-même. Si vous déclarez des noms de la forme `__xxx` ou `__xxx_yyy`, c'est-à-dire précédés de deux traits de soulignement, l'analyseur Python ajoutera automatiquement le nom de classe au nom déclaré, créant des variables masquées, par exemple :

```
class MyClass:
    __attr = 10   #private class attribute

    def method1(self):
        pass

    def method2(self, p1, p2):
        pass

    def __privateMethod(self, text):
        self.__text = text    #private attribute
```

A la différence de Java, avec Python, toutes les références à des variables d'instance doivent être qualifiées avec `self` ; il n'y a pas d'utilisation implicite de `this`.

Héritage

La capacité à hériter de classes est fondamentale à la programmation orientée objet. Python prend en charge aussi bien l'héritage unique que l'héritage multiple. L'*héritage unique* signifie qu'il ne peut exister qu'une seule superclasse. L'*héritage multiple* signifie qu'il peut exister plusieurs superclasses.

L'héritage est mis en oeuvre par la sous-classification d'autres classes. Un nombre quelconque de classes Python peuvent être des superclasses. Dans l'implémentation Jython de Python, il n'est possible d'hériter directement ou indirectement que d'une seule classe Java. Il n'est pas nécessaire de fournir une superclasse.

Tout attribut ou toute méthode d'une superclasse se trouve également dans une sous-classe et peut être utilisé par la classe elle-même, ou par un client dans la mesure où l'attribut ou la méthode n'est pas masqué. Il est possible d'utiliser une instance de sous-classe partout où une instance de superclasse peut être utilisée ; c'est un exemple de *polymorphisme*. Ces fonctionnalités permettent la réutilisation et facilitent l'extension.

Exemple

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass     #single inheritance

class Class4(Class3, Class2): pass     #multiple inheritance
```

Chapitre 3. Génération de scripts dans IBM SPSS Modeler

Types de scripts

Dans IBM SPSS Modeler, il existe trois types de scripts :

- Les *scripts de flux* sont utilisés pour contrôler l'exécution d'un flux unique et sont stockés dans le flux.
- Les *scripts de super noeud* sont utilisés pour contrôler le comportement des super noeuds.
- Les *scripts autonomes ou de session* peuvent être utilisés pour coordonner l'exécution sur plusieurs flux différents.

Diverses méthodes peuvent être utilisées dans les scripts dans IBM SPSS Modeler et permettent d'accéder à un large éventail de fonctionnalités SPSS Modeler. Ces méthodes sont également utilisées dans Chapitre 4, «API de scriptage», à la page 37 pour créer des fonctions plus avancées.

Flux, flux super noeud et diagrammes

La plupart du temps, le terme *flux* signifie la même chose, qu'il s'agisse d'un flux chargé à partir d'un fichier ou utilisé dans un super noeud. Il signifie généralement une collection de noeuds connectés ensemble et pouvant être exécutés. Toutefois, dans le cadre du scriptage, toutes les opérations ne sont pas forcément prises en charge dans tous les emplacements. Cela signifie qu'un auteur de script doit connaître les variantes de script qu'il utilise.

Flux

Un flux est le principal type de document IBM SPSS Modeler. Il peut être enregistré, chargé, édité et exécuté. Les flux peuvent aussi être associés à des paramètres, des valeurs globales, un script et d'autres d'informations.

Flux super noeud

Le *flux super noeud* est le type de flux utilisé dans un super noeud. Tout comme un flux normal, il contient des noeuds qui sont liés ensemble. Les flux super noeud présentent cependant des différences par rapport à un flux normal :

- Les paramètres et les éventuels scripts sont associés au super noeud propriétaire du flux super noeud, et non au flux super noeud lui-même.
- Les flux super noeud possèdent des noeuds de connecteurs d'entrée et de sortie supplémentaires, selon le type de super noeud. Ces noeuds de connecteur sont utilisés pour faire circuler les informations en entrée et en sortie du flux super noeud et sont créés automatiquement lors de la création de ce dernier.

Diagrammes

Le terme *diagramme* couvre les fonctions qui sont prises en charge par les flux normaux et les flux super noeud, telles que l'ajout et la suppression de noeuds ou la modification des connexions entre noeuds.

Exécution d'un flux

L'exemple suivant exécute tous les noeuds exécutables dans le flux, et constitue le type de script de flux le plus simple :

```
modeler.script.stream().runAll(None)
```

L'exemple suivant exécute également tous les noeuds exécutables dans le flux :

```
stream = modeler.script.stream()
stream.runAll(None)
```

Dans cet exemple, le flux est stocké dans une variable appelée `stream` (flux). Le stockage du flux dans une variable est utile car un script sert généralement à modifier soit le flux soit les noeuds au sein d'un flux. La création d'une variable qui stocke le flux permet d'obtenir un script plus concis.

Contexte de génération de scripts

Le module `modeler.script` fournit le contexte dans lequel un script est exécuté. Le module est automatiquement importé dans un script SPSS Modeler au moment de l'exécution. Le module définit quatre fonctions qui permettent au script d'accéder à son environnement d'exécution :

- La fonction `session()` renvoie la session pour le script. La session définit des informations telles que les paramètres régionaux et le backend SPSS Modeler (soit un processus local soit un SPSS Modeler Server en réseau) utilisé pour exécuter les flux.
- La fonction `stream()` peut être utilisée avec les scripts de flux et de super noeud. Cette fonction renvoie le flux propriétaire du script de flux ou de super noeud en cours d'exécution.
- La fonction `diagram()` peut être utilisée avec les scripts de super noeud. Cette fonction renvoie le diagramme au sein du super noeud. Pour les autres types de script, cette fonction renvoie les mêmes éléments que la fonction `stream()`.
- La fonction `supernode()` peut être utilisée avec les scripts de super noeud. Cette fonction renvoie le super noeud propriétaire du script en cours d'exécution.

Les quatre fonctions et leurs sorties sont résumées dans le tableau suivant.

Tableau 9. Récapitulatif des fonctions `modeler.script`

Type de script	`session()`	`stream()`	`diagram()`	`supernode()`
Autonome	Renvoie une session.	Renvoie le flux en cours de gestion au moment de l'appel du script (par exemple, le flux transmis via l'option `-stream` du mode de traitement par lots), ou `None`.	Identique à `stream()`	Sans objet
Flux	Renvoie une session.	Renvoie un flux.	Identique à `stream()`	Sans objet
Super noeud	Renvoie une session.	Renvoie un flux.	Renvoie un flux de super noeud.	Renvoie un super noeud.

Le module `modeler.script` définit également une méthode d'arrêt du script avec un code d'exit. La fonction `exit(exit-code)` arrête l'exécution du script et renvoie le code d'exit entier fourni.

L'une des méthodes définies pour un flux est `runAll(List)`. Cette méthode exécute tous les noeuds exécutables. Tous les modèles ou sorties générés par l'exécution des noeuds sont ajoutés à la liste fournie.

Il est courant pour une exécution de flux de générer des sorties telles que des modèles, des graphiques et autres résultats. Pour capturer cette sortie, un script peut fournir une variable initialisée en liste, par exemple :

```
stream = modeler.script.stream()
results = []
stream.runAll(results)
```

Lorsque l'exécution est terminée, tous les objets générés par l'exécution sont accessibles à partir de la liste `results`.

Référence aux noeuds existants

Un flux est souvent préconfiguré avec certains paramètres qui doivent être modifiés avant l'exécution du flux. La modification de ces paramètres implique les tâches suivantes :

1. Localisation des noeuds dans le flux approprié.
2. Modification des paramètres de noeud et/ou de flux.

Recherche de noeuds

Les flux permettent de localiser un noeud existant de différentes manières. Ces méthodes sont résumées dans le tableau suivant.

Tableau 10. Méthodes de localisation d'un noeud existant

Méthode	Type de retour	Description
`s.findAll(type, label)`	Collection	Renvoie une liste de tous les noeuds avec le type et le libellé spécifiés. Le type ou le libellé peut avoir la valeur `None`, auquel cas l'autre paramètre est utilisé.
`s.findAll(filter, recursive)`	Collection	Renvoie l'ensemble de tous les noeuds acceptés par le filtre spécifié. Si l'indicateur récursif est `True`, tous les super noeuds appartenant au flux spécifié font également l'objet de la recherche.
`s.findByID(id)`	Noeud	Renvoie le noeud avec l'ID fourni ou `None` si ce noeud n'existe pas. La recherche se limite au flux en cours.
`s.findByType(type, label)`	Noeud	Renvoie le noeud avec le type et/ou le libellé fourni. Le type ou le nom peut avoir la valeur `None`, auquel cas l'autre paramètre est utilisé. Si plusieurs noeuds génèrent une correspondance, un noeud arbitraire est choisi et renvoyé. Si aucun noeud ne génère une correspondance, la valeur renvoyée est `None`.
`s.findDownstream(fromNodes)`	Collection	Recherche dans la liste de noeuds fournie et renvoie l'ensemble de noeuds en aval des noeuds fournis. La liste renvoyée inclut les noeuds initialement fournis.
`s.findUpstream(fromNodes)`	Collection	Recherche dans la liste de noeuds fournie et renvoie l'ensemble de noeuds en amont des noeuds fournis. La liste renvoyée inclut les noeuds initialement fournis.

Par exemple, si un flux contient un seul noeud Filtrer auquel doit accéder le script, le noeud Filtrer peut être localisé à l'aide du script suivant :

```
stream = modeler.script.stream()
node = stream.findByType("filter", None)
...
```

Sinon, si l'ID du noeud (tel qu'il apparaît dans l'onglet Annotations de la boîte de dialogue du noeud) est connu, il peut être utilisé pour rechercher le noeud, par exemple :

```
stream = modeler.script.stream()
node = stream.findByID("id32FJT71G2")  # the filter node ID
...
```

Définition des propriétés

Les noeuds, les flux, les modèles et les sorties sont tous dotés de propriétés qui sont accessibles et qui peuvent, dans la plupart des cas, être définies. Les propriétés sont généralement utilisées pour modifier le comportement ou l'apparence de l'objet. Les méthodes disponibles pour accéder aux propriétés d'objet et pour les définir sont résumées dans le tableau suivant.

Tableau 11. Méthodes permettant d'accéder aux propriétés d'objet et de les définir

Méthode	Type de retour	Description
`p.getPropertyValue(propertyName)`	Objet	Renvoie la valeur de la propriété nommée, ou `None` s'il n'existe aucune propriété de ce genre.
`p.setPropertyValue(propertyName, value)`	Sans objet	Définit la valeur de la propriété nommée.
`p.setPropertyValues(properties)`	Sans objet	Définit les valeurs des propriétés nommées. Chaque entrée dans la carte de propriétés est composée d'une clé qui représente le nom de la propriété et la valeur qui doit être affectée à cette propriété.
`p.getKeyedPropertyValue( propertyName, keyName)`	Objet	Renvoie la valeur de la propriété nommée et la clé associée, ou `None` s'il n'existe aucune propriété ou clé de ce genre.
`p.setKeyedPropertyValue( propertyName, keyName, value)`	Sans objet	Définit la valeur de la propriété nommée et de la clé.

Par exemple, si vous souhaitez définir la valeur d'un noeud Délimité au début d'un flux, vous pouvez utiliser le script suivant :

```
stream = modeler.script.stream()
node = stream.findByType("variablefile", None)
node.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
...
```

Vous pouvez aussi souhaiter filtrer un champ à partir d'un noeud Filtrer. Dans ce cas, la valeur est également saisie sur le nom du champ, par exemple :

```
stream = modeler.script.stream()
# Locate the filter node ...
node = stream.findByType("filter", None)
# ... and filter out the "Na" field
node.setKeyedPropertyValue("include", "Na", False)
```

Création de noeuds et modification de flux

Dans certains cas, vous souhaiterez peut-être ajouter de nouveaux noeuds à des flux existants. L'ajout de noeuds à des flux existants implique généralement les tâches suivantes :

1. Création de noeuds.
2. Création de liens entre les noeuds dans le flux existant.

Création de noeuds

Les flux permettent de créer des noeuds de différentes manières. Ces méthodes sont résumées dans le tableau suivant.

Tableau 12. Méthodes de création de noeuds

Méthode	Type de retour	Description
`s.create(nodeType, name)`	Noeud	Crée un noeud du type spécifié et l'ajoute au flux spécifié.
`s.createAt(nodeType, name, x, y)`	Noeud	Crée un noeud du type spécifié et l'ajoute au flux spécifié à l'emplacement spécifié. Si x < 0 ou y < 0, l'emplacement n'est pas défini.
`s.createModelApplier(modelOutput, name)`	Noeud	Crée un noeud application de modèle dérivé de l'objet de sortie de modèle fourni.

Par exemple, pour créer un nouveau noeud type dans un flux, vous pouvez utiliser le script suivant :

```
stream = modeler.script.stream()
# Create a new type node
node = stream.create("type", "My Type")
```

Création et suppression de liens entre les noeuds

Lorsqu'un nouveau noeud est créé au sein d'un flux, il doit être connecté dans une séquence de noeuds pour pouvoir être utilisé. Les flux fournissent plusieurs méthodes permettant de créer et de supprimer des liens entre les noeuds. Ces méthodes sont résumées dans le tableau suivant.

Tableau 13. Méthodes de création et de suppression de liens entre les noeuds

Méthode	Type de retour	Description
`s.link(source, target)`	Sans objet	Crée un nouveau lien entre les noeuds source et cible.
`s.link(source, targets)`	Sans objet	Crée de nouveaux liens entre le noeud source et chaque noeud cible dans la liste fournie.
`s.linkBetween(inserted, source, target)`	Sans objet	Connecte un noeud entre deux autres instances de noeud (les noeuds source et cible) et définit la position du noeud inséré pour qu'il soit entre eux. Tout lien direct entre les noeuds source et cible est d'abord supprimé.
`s.linkPath(path)`	Sans objet	Crée un nouveau chemin d'accès entre les instances de noeud. Le premier noeud est lié au deuxième, le deuxième est lié au troisième, et ainsi de suite.
`s.unlink(source, target)`	Sans objet	Supprime tout lien direct entre les noeuds source et cible.
`s.unlink(source, targets)`	Sans objet	Supprime tout lien direct entre le noeud source et chaque objet dans la liste des cibles.
`s.unlinkPath(path)`	Sans objet	Supprime tout chemin d'accès existant entre des instances de noeud.

Tableau 13. Méthodes de création et de suppression de liens entre les noeuds (suite)

Méthode	Type de retour	Description
`s.disconnect(node)`	Sans objet	Supprime tout lien entre le noeud fourni et les autres noeuds du flux spécifié.
`s.isValidLink(source, target)`	*booléen*	Renvoie `True` s'il est possible (valide) de créer un lien entre les noeuds source et cible spécifiés. Cette méthode vérifie que les deux objets appartiennent au flux spécifié, que le noeud source peut fournir un lien, que le noeud cible peut recevoir un lien, et que la création d'un tel lien ne provoquera pas de circularité dans le flux.

L'exemple de script ci-dessous effectue les cinq tâches suivantes :

1. Crée un noeud d'entrée Délimité, un noeud Filtrer et un noeud de sortie Table.
2. Connecte les noeuds.
3. Définit le nom de fichier sur le noeud d'entrée Délimité.
4. Filtre le champ "Drug" dans le résultat.
5. Exécute le noeud Table.

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "My File Input ", 96, 64)
filternode = stream.createAt("filter", "Filter", 192, 64)
tablenode = stream.createAt("table", "Table", 288, 64)
stream.link(filenode, filternode)
stream.link(filternode, tablenode)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
filternode.setKeyedPropertyValue("include", "Drug", False)
results = []
tablenode.run(results)
```

Importation, remplacement et suppression de noeuds

Outre la création et la connexion de noeuds, il est souvent nécessaire de remplacer et de supprimer des noeuds du flux. Les méthodes disponibles pour importer, remplacer et supprimer des noeuds sont résumées dans le tableau suivant.

Tableau 14. Méthodes d'importation, de remplacement et de suppression de noeuds

Méthode	Type de retour	Description
`s.replace(originalNode, replacementNode, discardOriginal)`	Sans objet	Remplace le noeud spécifié dans le flux spécifié. Le noeud d'origine et le noeud de remplacement doivent tous deux appartenir au flux spécifié.

Tableau 14. Méthodes d'importation, de remplacement et de suppression de noeuds (suite)

Méthode	Type de retour	Description
`s.insert(source, nodes, newIDs)`	Liste	Insère des copies des noeuds dans la liste fournie. On suppose que tous les noeuds de la liste fournie sont inclus dans le flux spécifié. L'indicateur `newIDs` indique si de nouveaux ID doivent être générés pour chaque noeud ou si l'ID existant doit être copié et utilisé. Il est admis que tous les noeuds d'un flux ont un ID unique ; cet indicateur doit donc être défini sur `True` si le flux source est identique au flux spécifié. La méthode renvoie la liste des noeuds récemment insérés, où l'ordre des noeuds est non défini (autrement dit, l'ordre n'est pas nécessairement le même que celui des noeuds dans la liste d'entrée).
`s.delete(node)`	Sans objet	Supprime du flux spécifié le noeud spécifié. Le noeud doit appartenir au flux spécifié.
`s.deleteAll(nodes)`	Sans objet	Supprime du flux spécifié tous les noeuds spécifiés. Tous les noeuds de la collection doivent appartenir au flux spécifié.
`s.clear()`	Sans objet	Supprime tous les noeuds du flux spécifié.

Traversée des noeuds d'un flux

L'une des conditions requises courantes est d'identifier les noeuds qui se trouvent en amont ou en aval d'un noeud particulier. Le flux fournit plusieurs méthodes qui peuvent être utilisées pour identifier ces noeuds. Ces méthodes sont résumées dans le tableau suivant.

Tableau 15. Méthodes d'identification des noeuds en amont et en aval

Méthode	Type de retour	Description
`s.iterator()`	Itérateur	Renvoie un itérateur sur les objets de noeud contenus dans le flux spécifié. Si le flux est modifié entre les appels de la fonction `next()`, le comportement de l'itérateur est non défini.
`s.predecessorAt(node, index)`	Noeud	Renvoie le prédécesseur immédiat spécifié du noeud fourni ou `None` si l'index est en dehors des limites.
`s.predecessorCount(node)`	*Entier (int)*	Renvoie le nombre de prédécesseurs immédiats du noeud fourni.
`s.predecessors(node)`	Liste	Renvoie les prédécesseurs immédiats du noeud fourni.
`s.successorAt(node, index)`	Noeud	Renvoie le successeur immédiat spécifié du noeud fourni ou `None` si l'index est en dehors des limites.

Tableau 15. Méthodes d'identification des noeuds en amont et en aval (suite)

Méthode	Type de retour	Description
`s.successorCount(node)`	*Entier (int)*	Renvoie le nombre de successeurs immédiats du noeud fourni.
`s.successors(node)`	Liste	Renvoie les successeurs immédiats du noeud fourni.

Effacement ou suppression d'éléments

La fonction de scriptage existante prend en charge différentes utilisations de la commande `clear`, par exemple :

- `clear outputs` Pour supprimer tous les éléments de sortie de la palette du gestionnaire.
- `clear generated palette` Pour effacer tous les nuggets de modèle dans la palette Modèles.
- `clear stream` Pour supprimer le contenu d'un flux.

Le scriptage Python prend en charge un ensemble similaire de fonctions ; la commande `removeAll()` permet d'effacer les gestionnaires de flux, de sorties et de modèles, par exemple :

- Pour effacer le gestionnaire de flux :

```
session = modeler.script.session()
session.getStreamManager.removeAll()
```

- Pour effacer le gestionnaire de sorties :

```
session = modeler.script.session()
session.getDocumentOutputManager().removeAll()
```

- Pour effacer le gestionnaire de modèles :

```
session = modeler.script.session()
session.getModelOutputManager().removeAll()
```

Informations sur les noeuds

Les noeuds appartiennent à différentes catégories telles que les noeuds d'importation et d'exportation de données, les noeuds de génération de modèle et d'autres types de noeuds. Pour chaque noeud, plusieurs méthodes peuvent être utilisées pour trouver des informations sur le noeud.

Les méthodes disponibles pour obtenir l'ID, le nom et le libellé d'un noeud sont résumées dans le tableau suivant.

Tableau 16. Méthodes permettant d'obtenir l'ID, le nom et le libellé d'un noeud

Méthode	Type de retour	Description
`n.getLabel()`	*chaîne*	Renvoie le libellé d'affichage du noeud spécifié. Le libellé correspond à la valeur de la propriété `custom_name` uniquement si cette propriété est une chaîne non vide et si la propriété `use_custom_name` n'est pas définie ; sinon, le libellé correspond à la valeur de `getName()`.

Tableau 16. Méthodes permettant d'obtenir l'ID, le nom et le libellé d'un noeud (suite)

Méthode	Type de retour	Description
`n.setLabel(label)`	Sans objet	Définit le libellé d'affichage du noeud spécifié. Si le nouveau libellé est une chaîne non vide, il est affecté à la propriété `custom_name` et `False` est affecté à la propriété `use_custom_name` afin que le libellé indiqué ait la priorité ; sinon, une chaîne vide est affectée à la propriété `custom_name` et `True` est affecté à la propriété `use_custom_name`.
`n.getName()`	*chaîne*	Renvoie le nom du noeud spécifié.
`n.getID()`	*chaîne*	Renvoie l'ID du noeud spécifié. Un nouvel ID est créé chaque fois qu'un nouveau noeud est créé. L'ID est conservé avec le noeud lorsqu'il est sauvegardé dans le cadre d'un flux, de sorte que lorsque le flux est ouvert, les ID de noeud sont préservés. Toutefois, si un noeud sauvegardé est inséré dans un flux, le noeud inséré est considéré comme un nouvel objet et un nouvel ID lui sera attribué.

Les méthodes disponibles pour obtenir d'autres informations sur un noeud sont résumées dans le tableau suivant.

Tableau 17. Méthodes permettant d'obtenir des informations sur un noeud

Méthode	Type de retour	Description
`n.getTypeName()`	*chaîne*	Renvoie le nom de génération de script de ce noeud. Il s'agit du même nom qui peut être utilisé pour créer une nouvelle instance de ce noeud.
`n.isInitial()`	*Booléen*	Renvoie `True` s'il s'agit d'un noeud *initial*, c'est-à-dire un noeud qui apparaît au début d'un flux.
`n.isInline()`	*Booléen*	Renvoie `True` s'il s'agit d'un noeud *en ligne*, c'est-à-dire un noeud qui apparaît au milieu d'un flux.
`n.isTerminal()`	*Booléen*	Renvoie `True` s'il s'agit d'un noeud *terminal*, c'est-à-dire un noeud qui apparaît à la fin d'un flux.
`n.getXPosition()`	*Entier (int)*	Renvoie le décalage de la position x du noeud dans le flux.
`n.getYPosition()`	*Entier (int)*	Renvoie le décalage de la position y du noeud dans le flux.
`n.setXYPosition(x, y)`	Sans objet	Définit la position du noeud dans le flux.
`n.setPositionBetween(source, target)`	Sans objet	Définit la position du noeud dans le flux de manière à le positionner entre les noeuds fournis.

Tableau 17. Méthodes permettant d'obtenir des informations sur un noeud (suite)

Méthode	Type de retour	Description
`n.isCacheEnabled()`	*Booléen*	Renvoie `True` si le cache est activé ; sinon, renvoie `False`.
`n.setCacheEnabled(val)`	Sans objet	Active ou désactive le cache pour cet objet. Si le cache est saturé et que la mise en cache se trouve désactivée, le cache est vidé.
`n.isCacheFull()`	*Booléen*	Renvoie `True` si le cache est saturé ; sinon, renvoie `False`.
`n.flushCache()`	Sans objet	Vide le cache de ce noeud. N'a aucune incidence si le cache est désactivé ou n'est pas saturé.

Chapitre 4. API de scriptage

Introduction à l'API de scriptage

L'API de scriptage (génération de scripts) donne accès à un large éventail de fonctionnalités SPSS Modeler. Toutes les méthodes décrites jusqu'ici font partie de l'API et sont accessibles de manière implicite au sein du script, sans autre importation. Toutefois, si vous souhaitez faire référence aux classes de l'API, vous devez importer explicitement l'API avec l'instruction suivante :

```
import modeler.api
```

Cette instruction d'importation est requise par de nombreux exemples de l'API de scriptage.

Vous trouverez un guide complet sur les classes, les méthodes et les paramètres qui sont disponibles par le biais de l'API de scriptage dans le document *IBM SPSS Modeler 17 Python Scripting API Reference Guide*.

Exemple : recherche de noeuds à l'aide d'un filtre personnalisé

La section «Recherche de noeuds», à la page 29 inclut un exemple de recherche de noeud dans un flux et utilise le nom de type du noeud comme critère de recherche. Dans certains cas, une recherche plus générique est requise et peut être implémentée en utilisant la classe `NodeFilter` et la méthode `findAll()` du flux. Ce type de recherche implique les deux étapes suivantes :

1. Création d'une nouvelle classe qui étend `NodeFilter` et qui implémente une version personnalisée de la méthode `accept()`.
2. Appel de la méthode `findAll()` du flux avec une instance de cette nouvelle classe. Tous les noeuds correspondant aux critères définis dans la méthode `accept()` sont ainsi renvoyés.

L'exemple suivant indique comment rechercher dans un flux des noeuds pour lesquels le cache de noeud est activé. La liste de noeuds renvoyée peut être utilisée pour vider ou désactiver les caches de ces noeuds.

```
import modeler.api

class CacheFilter(modeler.api.NodeFilter):
 """A node filter for nodes with caching enabled"""
 def accept(this, node):
  return node.isCacheEnabled()

cachingnodes = modeler.script.stream().findAll(CacheFilter(), False)
```

Métadonnées : informations sur les données

Etant donné que les noeuds sont connectés entre eux dans un flux, des informations sur les colonnes ou les champs disponibles à chaque noeud sont également disponibles. Par exemple, dans l'interface utilisateur Modeler, vous pouvez ainsi sélectionner les champs en fonction desquels effectuer un tri ou une agrégation. Ces informations sont appelées modèle de données.

Les scripts peuvent également accéder au modèle de données en consultant les champs entrant ou sortant dans un noeud. Pour certains noeuds, les modèles de données de sortie et d'entrée sont les mêmes (par exemple, un noeud Trier réorganise simplement les enregistrements mais ne modifie pas le modèle de données). Certains noeuds, comme le noeud Dériver, ajoutent de nouveaux champs. D'autres, tels que le noeud Filtrer, peuvent renommer ou supprimer des champs.

Dans l'exemple suivant, le script utilise le flux standard IBM SPSS Modeler druglearn.str et, pour chaque champ, génère un modèle avec l'un des champs d'entrée supprimés. Il effectue cette opération en :

1. accédant au modèle de données de sortie à partir du noeud Type ;
2. effectuant une boucle dans chaque champ du modèle de données de sortie ;
3. modifiant le noeud Filtrer pour chaque champ d'entrée ;
4. modifiant le nom du modèle généré ;
5. exécutant le noeud génération du modèle.

Remarque : Avant d'exécuter le script dans le flux druglean.str, pensez à définir le langage de scriptage Python (le flux a été créé dans une version antérieure de IBM SPSS Modeler ; le langage de scriptage du flux est donc défini par Legacy (existant)).

```
import modeler.api

stream = modeler.script.stream()
filternode = stream.findByType("filter", None)
typenode = stream.findByType("type", None)
c50node = stream.findByType("c50", None)
# Toujours utiliser un nom de modèle personnalisé
c50node.setPropertyValue("use_model_name", True)

lastRemoved = None
fields = typenode.getOutputDataModel()
for field in fields:
    # S'il s'agit du champ cible, l'ignorer
    if field.getModelingRole() == modeler.api.ModelingRole.OUT:
        continue

    # Activer de nouveau le champ le plus récemment supprimé
    if lastRemoved != None:
        filternode.setKeyedPropertyValue("include", lastRemoved, True)

    # Supprimer le champ
    lastRemoved = field.getColumnName()
    filternode.setKeyedPropertyValue("include", lastRemoved, False)

    # Définir le nom du nouveau modèle, puis exécuter le modèle généré
    c50node.setPropertyValue("model_name", "Exclude " + lastRemoved)
    c50node.run([])
```

L'objet Modèle de données offre un certain nombre de méthodes pour accéder aux informations relatives aux champs ou aux colonnes du modèle de données. Ces méthodes sont résumées dans le tableau suivant.

Tableau 18. Méthodes de l'objet Modèle de données pour accéder aux informations sur les champs ou les colonnes

Méthode	Type de retour	Description
d.getColumnCount()	*Entier (int)*	Renvoie le nombre de colonnes du modèle de données.
d.columnIterator()	Itérateur	Renvoie un itérateur qui renvoie chaque colonne dans l'ordre d'insertion "naturel". L'itérateur renvoie les instances de colonne.
d.nameIterator()	Itérateur	Renvoie un itérateur qui renvoie le nom de chaque colonne dans l'ordre d'insertion "naturel".

Tableau 18. Méthodes de l'objet Modèle de données pour accéder aux informations sur les champs ou les colonnes (suite)

Méthode	Type de retour	Description
`d.contains(nom)`	*Booléen*	Renvoie `True` s'il existe une colonne avec le nom indiqué dans ce modèle de données et `False` dans le cas contraire.
`d.getColumn(nom)`	Colonne	Renvoie la colonne portant le nom indiqué.
`d.getColumnGroup(nom)`	Groupe de colonnes	Renvoie le groupe de colonne désigné ou `None` si ce groupe n'existe pas.
`d.getColumnGroupCount()`	*Entier (int)*	Renvoie le nombre de groupes de colonnes du modèle de données.
`d.columnGroupIterator()`	Itérateur	Renvoie un itérateur qui renvoie chaque groupe de colonnes l'un après l'autre.
`d.toArray()`	Colonne[]	Renvoie le modèle de données sous forme de tableau de colonnes. Les colonnes sont classées dans leur ordre d'insertion "naturel".

Chaque champ (objet de colonne) inclut un certain nombre de méthodes pour accéder aux informations relatives à la colonne. Le tableau ci-après présente une sélection de ces méthodes.

Tableau 19. Méthodes de l'objet de colonne pour accéder aux informations sur la colonne

Méthode	Type de retour	Description
`c.getColumnName()`	*chaîne*	Renvoie le nom de la colonne.
`c.getColumnLabel()`	*chaîne*	Renvoie le libellé de la colonne ou une chaîne vide si aucun libellé n'est associé à la colonne.
`c.getMeasureType()`	Type de mesure (MeasureType)	Renvoie le type de mesure de la colonne.
`c.getStorageType()`	Type de stockage (StorageType)	Renvoie le type de stockage de la colonne.
`c.isMeasureDiscrete()`	*Booléen*	Renvoie `True` si la colonne est discrète. Les colonnes qui sont un ensemble ou un indicateur sont considérées comme discrètes.
`c.isModelOutputColumn()`	*Booléen*	Renvoie `True` si la colonne est une colonne de sortie de modèle.
`c.isStorageDatetime()`	*Booléen*	Renvoie `True` si le stockage de la colonne est une valeur d'heure, de date ou d'horodatage.
`c.isStorageNumeric()`	*Booléen*	Renvoie `True` si le stockage de la colonne est un entier ou un nombre réel.
`c.isValidValue(valeur)`	*Booléen*	Renvoie `True` si la valeur spécifiée est valide pour ce stockage et `valid` lorsque les valeurs de colonne valides sont affichées.

Tableau 19. Méthodes de l'objet de colonne pour accéder aux informations sur la colonne (suite)

Méthode	Type de retour	Description
`c.getModelingRole()`	Rôle de modélisation (ModelingRole)	Renvoie le rôle de modélisation de la colonne.
`c.getSetValues()`	Objet[]	Renvoie un tableau de valeurs valides pour la colonne ou `None` si les valeurs ne sont pas affichées ou si la colonne n'est pas un ensemble.
`c.getValueLabel(valeur)`	*chaîne*	Renvoie le libellé de la valeur dans la colonne.
`c.getFalseFlag()`	Objet	Renvoie l'indicateur "false" pour la colonne ou `None` si la valeur n'est pas affichée ou si la colonne n'est pas un indicateur.
`c.getTrueFlag()`	Objet	Renvoie l'indicateur "true" pour la colonne ou `None` si la valeur n'est pas affichée ou si la colonne n'est pas un indicateur.
`c.getLowerBound()`	Objet	Renvoie la valeur limite inférieure des valeurs de la colonne ou `None` si la valeur n'est pas affichée ou si la colonne n'est pas continue.
`c.getUpperBound()`	Objet	Renvoie la valeur limite supérieure des valeurs de la colonne ou `None` si la valeur n'est pas affichée ou si la colonne n'est pas continue.

Notez que la plupart des méthodes qui accèdent aux informations relatives à une colonne possèdent des méthodes équivalentes définies dans l'objet de modèle de données lui-même. Par exemple, les deux instructions suivantes sont équivalentes :

```
dataModel.getColumn("Nom").getModelingRole()
dataModel.getModelingRole("Nom")
```

Accès aux objets générés

L'exécution d'un flux implique généralement la génération d'objets de sortie supplémentaires. Ces objets supplémentaires peuvent être un nouveau modèle ou une partie de sortie fournissant des informations à utiliser lors des exécutions ultérieures.

Dans l'exemple ci-dessous, le flux `druglearn.str` est de nouveau utilisé comme point de départ pour le flux. Dans cet exemple, tous les noeuds du flux sont exécutés et les résultats sont stockés dans une liste. Ensuite, le script parcourt les résultats en boucle, toute sortie du modèle résultant de l'exécution est enregistrée en tant que fichier de modèle IBM SPSS Modeler (`.gm`), et le modèle est exporté en PMML.

```
import modeler.api

stream = modeler.script.stream()

# Set this to an existing folder on your system.
# Include a trailing directory separator
modelFolder = "C:/temp/models/"

# Execute the stream
models = []
stream.runAll(models)
```

```
# Save any models that were created
taskrunner = modeler.script.session().getTaskRunner()
for model in models:
    # If the stream execution built other outputs then ignore them
    if not(isinstance(model, modeler.api.ModelOutput)):
        continue

    label = model.getLabel()
    algorithm = model.getModelDetail().getAlgorithmName()

    # save each model...
    modelFile = modelFolder + label + algorithm + ".gm"
    taskrunner.saveModelToFile(model, modelFile)

    # ...and export each model PMML...
    modelFile = modelFolder + label + algorithm + ".xml"
    taskrunner.exportModelToFile(model, modelFile, modeler.api.FileFormat.XML)
```

La classe d'exécution de tâche permet d'exécuter facilement diverses tâches courantes. Les méthodes disponibles dans cette classe sont résumées dans le tableau suivant.

Tableau 20. Méthodes de la classe d'exécution de tâche pour effectuer des tâches courantes

Méthode	Type de retour	Description
`t.createStream(name, autoConnect, autoManage)`	Flux	Crée et renvoie un nouveau flux. Notez que le code chargé de créer des flux de manière privée, sans qu'ils soient visibles pour l'utilisateur, doit définir l'indicateur `autoManage` sur `False`.
`t.exportDocumentToFile( documentOutput, filename, fileFormat)`	Sans objet	Exporte la description du flux vers un fichier en utilisant le format de fichier spécifié.
`t.exportModelToFile(modelOutput, filename, fileFormat)`	Sans objet	Exporte le modèle vers un fichier en utilisant le format de fichier spécifié.
`t.exportStreamToFile(stream, filename, fileFormat)`	Sans objet	Exporte le flux vers un fichier en utilisant le format de fichier spécifié.
`t.insertNodeFromFile(filename, diagram)`	Noeud	Lit et renvoie un noeud à partir du fichier spécifié, en l'insérant dans le diagramme fourni. Vous pouvez l'utiliser pour lire des objets de noeud et de super noeud.
`t.openDocumentFromFile(filename, autoManage)`	DocumentOutput	Lit et renvoie un document à partir du fichier spécifié.
`t.openModelFromFile(filename, autoManage)`	ModelOutput	Lit et renvoie un modèle à partir du fichier spécifié.
`t.openStreamFromFile(filename, autoManage)`	Flux	Lit et renvoie un flux à partir du fichier spécifié.
`t.saveDocumentToFile( documentOutput, filename)`	Sans objet	Enregistre le document à l'emplacement de fichier spécifié.
`t.saveModelToFile(modelOutput, filename)`	Sans objet	Enregistre le modèle à l'emplacement de fichier spécifié.
`t.saveStreamToFile(stream, filename)`	Sans objet	Enregistre le flux à l'emplacement de fichier spécifié.

Traitement des erreurs

Le langage Python fournit un traitement des erreurs via le bloc de code `try...except`. Celui-ci peut être utilisé dans les scripts pour intercepter les exceptions et gérer les problèmes qui, sans cela, entraîneraient l'arrêt du script.

Dans l'exemple de script ci-dessous, on tente d'extraire un modèle d'un IBM SPSS Collaboration and Deployment Services Repository. Cette opération peut provoquer une exception ; par exemple, il se peut que les données de connexion au référentiel n'aient pas été correctement configurées ou que le chemin de référentiel soit incorrect. Dans le script, cela peut entraîner l'émission d'une exception `ModelerException` (toutes les exceptions générées par IBM SPSS Modeler sont dérivées de `modeler.api.ModelerException`).

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
```

Remarque : certaines opérations de script peuvent provoquer l'émission d'exceptions Java standard ; celles-ci ne sont pas dérivées de `ModelerException`. Pour intercepter ces exceptions, un bloc except supplémentaire peut être utilisé pour intercepter toutes les exceptions Java, par exemple :

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
except java.lang.Exception, e:
    print "A Java exception occurred:", e.getMessage()
```

Paramètres de flux, de session et de super noeud

Les paramètres sont utiles pour transmettre des valeurs au moment de l'exécution, plutôt que de les coder en dur directement dans un script. Les paramètres et leurs valeurs sont définis de la même manière que pour les flux, à savoir comme des entrées dans la table de paramètres d'un flux ou super noeud, ou comme des paramètres sur la ligne de commande. Les classes de flux (Stream) et de super noeud (SuperNode) implémentent un ensemble de fonctions définies par l'objet ParameterProvider, comme indiqué dans le tableau suivant. La session fournit un appel `getParameters()` qui renvoie un objet définissant ces fonctions.

Tableau 21. Fonctions définies par l'objet ParameterProvider

Méthode	Type de retour	Description
`p.parameterIterator()`	Itérateur	Renvoie un itérateur des noms de paramètres pour cet objet.

Tableau 21. Fonctions définies par l'objet ParameterProvider (suite)

Méthode	Type de retour	Description
`p.getParameterDefinition( parameterName)`	ParameterDefinition	Renvoie la définition de paramètre pour le paramètre avec le nom spécifié, ou `None` s'il n'existe aucun paramètre de ce genre dans ce fournisseur. Le résultat peut être un instantané de la définition au moment de l'appel de la méthode et ne doit pas forcément refléter les modifications apportées ultérieurement au paramètre via ce fournisseur.
`p.getParameterLabel(parameterName)`	*chaîne*	Renvoie le libellé du paramètre nommé, ou `None` s'il n'existe aucun paramètre de ce genre.
`p.setParameterLabel(parameterName, label)`	Sans objet	Définit le libellé du paramètre nommé.
`p.getParameterStorage( parameterName)`	ParameterStorage	Renvoie le stockage du paramètre nommé, ou `None` s'il n'existe aucun paramètre de ce genre.
`p.setParameterStorage( parameterName, storage)`	Sans objet	Définit le stockage du paramètre nommé.
`p.getParameterType(parameterName)`	ParameterType	Renvoie le type du paramètre nommé, ou `None` s'il n'existe aucun paramètre de ce genre.
`p.setParameterType(parameterName, type)`	Sans objet	Définit le type du paramètre nommé.
`p.getParameterValue(parameterName)`	Objet	Renvoie la valeur du paramètre nommé, ou `None` s'il n'existe aucun paramètre de ce genre.
`p.setParameterValue(parameterName, value)`	Sans objet	Définit la valeur du paramètre nommé.

Dans l'exemple suivant, le script agrège certaines données Telco pour rechercher la région dont le revenu moyen (average income) est le plus faible. Un paramètre de flux est ensuite défini avec cette région. Ce paramètre de flux est par la suite utilisé dans un noeud Sélectionner (Select) afin d'exclure cette région des données, avant qu'un modèle d'attrition soit généré sur le reste.

Cet exemple est fictif car le script génère le noeud Sélectionner (Select) lui-même et aurait donc pu générer directement la valeur correcte dans l'expression du noeud Sélectionner (Select). Toutefois, les flux sont généralement préconfigurés ; la définition des paramètres de cette manière permet donc de donner un exemple utile.

La première partie de l'exemple de script crée le paramètre de flux qui contiendra la région dont le revenu moyen est le plus faible. Le script crée également les noeuds dans la branche d'agrégation et la branche de génération de modèle, et les connecte.

```
import modeler.api

stream = modeler.script.stream()

# Initialize a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)
```

```
# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)
```

L'exemple de script crée le flux suivant.

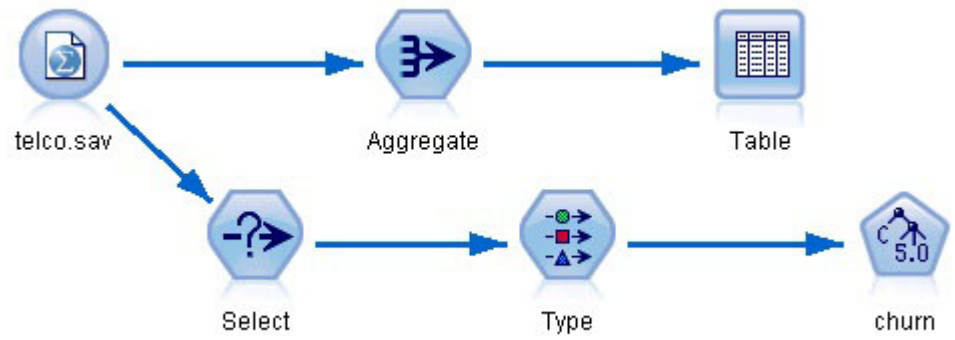

Figure 5. Flux résultant de l'exemple de script

La partie suivante de l'exemple de script exécute le noeud Table à la fin de la branche d'agrégation.

```
# First execute the table node
results = []
tablenode.run(results)
```

La partie suivante de l'exemple de script accède à la sortie de table générée par l'exécution du noeud Table. Ensuite, le script parcourt les lignes de la table, à la recherche de la région présentant le revenu moyen le plus faible.

```
# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None
```

```
# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1
```

La partie suivante du script utilise la région présentant le revenu moyen le plus faible pour définir le paramètre de flux "LowestRegion" créé précédemment. Le script exécute ensuite le générateur de modèle en excluant la région spécifiée des données d'apprentissage.

```
# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

L'exemple de script complet est présenté ci-dessous.

```
import modeler.api

stream = modeler.script.stream()

# Create a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)


# First execute the table node
results = []
tablenode.run(results)

# Running the table node should produce a single table as output
table = results[0]
```

```
# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1

# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

Valeurs globales

Les valeurs globales sont utilisées pour calculer diverses statistiques récapitulatives pour des champs spécifiés. Ces valeurs récapitulatives sont accessibles partout dans le flux. Les valeurs globales sont semblables aux paramètres de flux en ce sens qu'elles sont accessibles par nom via le flux. Elles diffèrent des paramètres de flux dans la mesure où les valeurs associées sont mises à jour automatiquement lorsqu'un noeud V. globales (Valeurs globales) est exécuté, plutôt que d'être affectées par le scriptage ou depuis la ligne de commande. Il est possible d'accéder aux valeurs globales pour un flux en appelant la méthode `getGlobalValues()` du flux.

L'objet GlobalValues définit les fonctions indiquées dans le tableau suivant.

Tableau 22. Fonctions définies par l'objet GlobalValues

Méthode	Type de retour	Description
`g.fieldNameIterator()`	Itérateur	Renvoie un itérateur pour chaque nom de champ avec au moins une valeur globale.
`g.getValue(type, fieldName)`	Objet	Renvoie la valeur globale pour le type et le nom de champ spécifiés, ou `None` si aucune valeur n'est trouvée. La valeur renvoyée est généralement un nombre, mais il se peut que des fonctionnalités futures renvoient des types de valeur différents.
`g.getValues(fieldName)`	Carte	Renvoie une carte contenant les entrées connues pour le nom de champ spécifié, ou `None` s'il n'existe aucune entrée pour le champ.

`GlobalValues.Type` définit le type des statistiques récapitulatives disponibles. Les statistiques récapitulatives suivantes sont disponibles :

- `MAX` : valeur maximum du champ.
- `MEAN` : valeur moyenne du champ.

- MIN : valeur minimum du champ.
- STDDEV : écart type du champ.
- SUM : somme des valeurs dans le champ.

Par exemple, le script suivant accède à la valeur moyenne du champ "income", qui est calculée par un noeud V. globales (Valeurs globales) :

```
import modeler.api

globals = modeler.script.stream().getGlobalValues()
mean_income = globals.getValue(modeler.api.GlobalValues.Type.MEAN, "income")
```

Utilisation de plusieurs flux : scripts autonomes

Pour utiliser plusieurs flux, il convient d'utiliser un script autonome. Le script autonome peut être édité et exécuté dans l'interface utilisateur IBM SPSS Modeler ou transmis comme paramètre de ligne de commande en mode de traitement par lots.

Le script autonome suivant ouvre deux flux. L'un de ces flux génère un modèle, et le deuxième trace la distribution des valeurs prédites.

```
# Change to the appropriate location for your system
demosDir = "C:/Program Files/IBM/SPSS/Modeler/16/DEMOS/streams/"

session = modeler.script.session()
tasks = session.getTaskRunner()

# Open the model build stream, locate the C5.0 node and run it
buildstream = tasks.openStreamFromFile(demosDir + "druglearn.str", True)
c50node = buildstream.findByType("c50", None)
results = []
c50node.run(results)

# Now open the plot stream, find the Na_to_K derive and the histogram
plotstream = tasks.openStreamFromFile(demosDir + "drugplot.str", True)
derivenode = plotstream.findByType("derive", None)
histogramnode = plotstream.findByType("histogram", None)

# Create a model applier node, insert it between the derive and histogram nodes
# then run the histgram
applyc50 = plotstream.createModelApplier(results[0], results[0].getName())
applyc50.setPositionBetween(derivenode, histogramnode)
plotstream.linkBetween(applyc50, derivenode, histogramnode)
histogramnode.setPropertyValue("color_field", "$C-Drug")
histogramnode.run([])

# Finally, tidy up the streams
buildstream.close()
plotstream.close()
```

Chapitre 5. Conseils pour la génération de scripts

Cette section présente les différents conseils et techniques pour l'utilisation des scripts : modification de l'exécution du flux, utilisation d'un mot de passe codé dans un script et accès aux objets du IBM SPSS Collaboration and Deployment Services Repository.

Modification de l'exécution du flux

Lorsqu'un flux est exécuté, ses noeuds terminaux sont exécutés dans un ordre optimisé pour la situation par défaut. Dans certains cas, vous pouvez choisir un ordre d'exécution différent. Pour modifier l'ordre d'exécution d'un flux, réalisez les opérations suivantes dans l'onglet Exécution de la boîte de dialogue Propriétés du flux :

1. Commencez par un script vide.
2. Cliquez sur le bouton **Ajouter le script par défaut** de la barre d'outils pour ajouter le script de flux par défaut.
3. Modifiez à votre convenance l'ordre des instructions qu'il contient.

Bouclage dans les noeuds

Vous pouvez utiliser une boucle `for` pour parcourir en boucle tous les noeuds d'un flux. Ainsi, les deux exemples de script suivants parcourent en boucle tous les noeuds et mettent les noms de champ de tous les noeuds Filtrer en majuscules.

Ces scripts peuvent être utilisés dans n'importe quel flux comportant un noeud, même si aucun champ n'est en réalité filtré. Ajoutez simplement un noeud Filtrer qui transmet tous les champs afin de mettre les noms de champ en majuscules sur l'ensemble du système.

```
# Alternative 1: using the data model nameIterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # nameIterator() returns the field names
        for field in node.getInputDataModel().nameIterator():
            newname = field.upper()
            node.setKeyedPropertyValue("new_name", field, newname)
```

```
# Alternative 2: using the data model iterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # iterator() returns the field objects so we need
        # to call getColumnName() to get the name
        for field in node.getInputDataModel().iterator():
            newname = field.getColumnName().upper()
            node.setKeyedPropertyValue("new_name", field.getColumnName(), newname)
```

Le script parcourt en boucle tous les noeuds du flux actuel et vérifie si chaque noeud est un noeud Filtrer. Si tel est le cas, le script parcourt en boucle chaque champ dans le noeud et utilise la fonction `field.upper()` ou `field.getColumnName().upper()` pour mettre le nom en majuscules.

Accès aux objets du IBM SPSS Collaboration and Deployment Services Repository

Si vous disposez d'une licence pour le référentiel IBM SPSS Collaboration and Deployment Services Repository, vous pouvez stocker, verrouiller et déverrouiller des objets dans le référentiel et en extraire à l'aide des commandes de script. Le référentiel permet de gérer le cycle de vie des modèles d'exploration de données et des objets prédictifs associés dans le contexte d'applications, de solutions et d'outils professionnels.

Connexion au IBM SPSS Collaboration and Deployment Services Repository

Pour pouvoir accéder au référentiel, vous devez dans un premier temps paramétrer une connexion valide avec ce dernier, en utilisant le menu Outils de l'interface utilisateur IBM SPSS Modeler ou avec une ligne de commande. (Pour plus d'informations, voir la rubrique «IBM SPSS Collaboration and Deployment Services Repository Arguments de connexion», à la page 65.)

Stockage et extraction des objets

Au sein d'un script, les commandes `retrieve` et `store` vous permettent d'accéder à différents objets, notamment les flux, modèles, sorties, noeuds et projets. La syntaxe est la suivante :

```
store object as REPOSITORY_PATH {label LABEL}
store object as URI [#l.label]
```

```
retrieve object REPOSITORY_PATH {label LABEL | version VERSION}
retrieve object URI [(#m.marker | #l.label)]
```

`REPOSITORY_PATH` fournit l'emplacement de l'objet dans le référentiel. Le chemin doit être placé entre guillemets et délimité par des barres obliques. Il ne distingue pas les majuscules des minuscules.

```
store stream as "/folder_1/folder_2/mystream.str"
store model Drug as "/myfolder/drugmodel"
store model Drug as "/myfolder/drugmodel.gm" label "final"
store node DRUG1n as "/samples/drug1ntypenode"
store project as "/CRISPDM/DrugExample.cpj"
store output "Data Audit of [6 fields]" as "/my folder/My Audit"
```

Il est également possible d'inclure dans le nom de l'objet une extension telle que *.str* ou *.gm*, mais cela n'est pas obligatoire tant que le nom est cohérent. Par exemple, si un modèle est stocké sans extension, il doit être extrait sur la base du même nom :

```
store model "/myfolder/drugmodel"
retrieve model "/myfolder/drugmodel"
```

et

```
store model "/myfolder/drugmodel.gm"
retrieve model "/myfolder/drugmodel.gm" version "0:2005-10-12 14:15:41.281"
```

Lors de l'extraction d'objets, la version la plus récente de l'objet est toujours renvoyée, sauf si vous indiquez une version ou un libellé en particulier. Lors de l'extraction d'un objet de noeud, le noeud est automatiquement inséré dans le flux actuel. Lors de l'extraction d'un objet de flux, vous devez utiliser un script autonome. Vous ne pouvez pas extraire un objet de flux à partir d'un script de flux.

Verrouillage et déverrouillage d'objets

A partir d'un script, vous pouvez verrouiller un objet pour empêcher les autres utilisateurs de mettre à jour ses versions existantes ou de créer de nouvelles versions. Vous pouvez également déverrouiller un objet que vous avez verrouillé.

La syntaxe pour verrouiller et déverrouiller un objet est la suivante :

```
lock REPOSITORY_PATH
lock URI

unlock REPOSITORY_PATH
unlock URI
```

Tout comme avec le stockage et la récupération d'objets, le `REPOSITORY_PATH` vous donne l'emplacement de l'objet dans le référentiel. Le chemin doit être placé entre guillemets et délimité par des barres obliques. Il ne distingue pas les majuscules des minuscules.

```
lock "/myfolder/Stream1.str"

unlock "/myfolder/Stream1.str"
```

Vous pouvez également utiliser un URI (Uniform Resource Identifier) plutôt qu'un chemin de référentiel pour fournir l'emplacement de l'objet. L'URI doit inclure le préfixe `spsscr:` et doit être entièrement entre guillemets. Seules les barres obliques sont autorisées en tant que délimiteurs de chemin ; les espaces, quant à eux, doivent être codés. En d'autres termes, utilisez `%20` à la place d'un espace dans le chemin. L'URI ne distingue pas les majuscules des minuscules. Voici quelques exemples :

```
lock "spsscr:///myfolder/Stream1.str"

unlock "spsscr:///myfolder/Stream1.str"
```

Veuillez noter que le verrouillage d'objets s'applique à toutes les versions d'un objet. Vous ne pouvez pas verrouiller ou déverrouiller des versions individuelles.

Génération d'un mot de passe codé

Dans certains cas, vous pouvez être amené à ajouter un mot de passe à un script, par exemple, pour accéder à une source de données protégée par un mot de passe. Vous pouvez utiliser les mots de passe codés dans :

- Les propriétés des noeuds source et de sortie SGBD
- Les arguments de ligne de commande permettant la connexion au serveur
- Les propriétés de connexion à la base de données stockées dans un fichier *.par* (fichier de paramètres généré depuis l'onglet Publier d'un noeud Export)

L'interface utilisateur dispose d'un outil qui permet de générer des mots de passe codés à partir de l'algorithme Blowfish (pour plus de détails, consultez le site *http://www.schneier.com/blowfish.html*). Une fois le mot de passe codé, vous pouvez le copier et le stocker dans des fichiers de script et des arguments de ligne de commande. La propriété de noeud `epassword`, utilisée pour `databasenode` et `databaseexportnode`, stocke le mot de passe codé.

1. Pour générer un mot de passe codé, choisissez l'option suivante dans le menu Outils :
   Coder le mot de passe...
2. Entrez un mot de passe dans la zone de texte Mot de passe.
3. Cliquez sur **Coder** pour générer le codage aléatoire de votre mot de passe.
4. Cliquez sur le bouton Copier pour copier le mot de passe codé dans le Presse-papiers.
5. Collez le mot de passe dans le script ou le paramètre souhaité.

Vérification du script

Vous pouvez procéder à une vérification rapide de la syntaxe de tous les types de script en cliquant sur le bouton de vérification rouge dans la barre d'outils de la boîte de dialogue Script autonome.

Figure 6. Icônes de la barre d'outils du script de flux

La vérification du script vous prévient de toute erreur se produisant dans votre code et vous propose des recommandations d'amélioration. Pour visualiser la ligne comportant des erreurs, cliquez sur le commentaire dans la partie inférieure de la boîte de dialogue. L'erreur est surlignée en rouge.

Génération de scripts à partir de la ligne de commande

La génération de scripts vous permet d'exécuter des opérations généralement effectuées dans l'interface utilisateur. Il vous suffit d'indiquer et d'exécuter un flux autonome dans la ligne de commande lors du lancement de IBM SPSS Modeler. Par exemple :

```
client -script scores.txt -execute
```

L'indicateur `-script` charge le script spécifié et l'indicateur `-execute` exécute toutes les commandes du fichier de script.

Compatibilité avec les versions précédentes

Les scripts créés dans les versions antérieures de IBM SPSS Modeler doivent en général fonctionner tels quels dans la nouvelle version. Cependant, les nuggets de modèle peuvent désormais être insérés automatiquement dans le flux (réglage par défaut), et peuvent soit remplacer soit compléter un nugget existant de ce type dans le flux. Ceci dépend des réglages des options **Ajout de modèle à un flux** et **Remplacer le modèle précédent** (**Outils > Options > Options utilisateur > Notifications**). Vous pouvez, par exemple, être amené à modifier le script d'une version précédente dans laquelle le remplacement de nugget est géré en supprimant le nugget existant et en insérant le nouveau.

Il se peut que des scripts créés dans la version actuelle ne fonctionnent pas dans les versions antérieures.

Si un script créé dans une version antérieure utilise une commande qui a été remplacée depuis (ou qui est obsolète), l'ancienne forme reste prise en charge mais un message d'avertissement apparaît. Par exemple, l'ancien mot-clé `generated` a été remplacé par `model`, et `clear generated` par `clear generated palette`. Les scripts qui utilisent les anciennes formes continuent de s'exécuter mais un avertissement apparaît.

Accès aux résultats d'exécution de flux

De nombreux noeuds IBM SPSS Modeler produisent des objets en sortie tels que des modèles, des graphiques et des données tabulaires. Une grande partie de ces sorties contiennent des valeurs utiles qui peuvent être utilisées par des scripts pour guider l'exécution ultérieure. Ces valeurs sont regroupées dans des conteneurs de contenu (appelés simplement conteneurs) auxquels on peut accéder à l'aide de balises ou d'ID qui identifient chaque conteneur. Le mode d'accès aux valeurs dépends du format ou "modèle de contenu" utilisé par ce conteneur.

Par exemple, de nombreuses sorties de modèle prédictif utilisent une variante de XML appelée PMML pour représenter les informations sur le modèle, par exemple, les champs utilisés par un arbre de décisions à chaque scission ou la manière dont les neurones d'un réseau de neurones sont connectés et avec quelle force. Les sorties de modèle qui utilisent PMML fournissent un modèle de contenu XML qui permet d'accéder à ces informations. Par exemple :

```
stream = modeler.script.stream()
# Assume the stream contains a single C5.0 model builder node
# and that the datasource, predictors and targets have already been
# set up
modelbuilder = stream.findByType("c50", None)
```

```
results = []
modelbuilder.run(results)
modeloutput = results[0]

# Now that we have the C5.0 model output object, access the
# relevant content model
cm = modeloutput.getContentModel("PMML")

# The PMML content model is a generic XML-based content model that
# uses XPath syntax. Use that to find the names of the data fields.
# The call returns a list of strings match the XPath values
dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
```

IBM SPSS Modeler prend en charge les modèles de contenu suivants dans le scriptage :

- Le **modèle de contenu de table** donne accès à des données tabulaires simples représentées sous la forme de lignes et de colonnes.
- Le **modèle de contenu XML** donne accès au contenu stocké au format XML.
- Le **modèle de contenu JSON** donne accès au contenu stocké au format JSON.
- Le **modèle de contenu de colonne de statistiques** donne accès à des statistiques récapitulatives à propos d'un champ spécifique.
- Le **modèle de contenu de colonne de statistiques par paire** donne accès à des statistiques récapitulatives entre deux champs ou les valeurs de deux champs distincts.

Modèle de contenu de table

Le modèle de contenu de table fournit un modèle simple permettant d'accéder à des données de ligne ou de colonne simples. Les valeurs d'une colonne particulière doivent avoir le même type de stockage (par exemple, des chaîne ou des entiers).

API

Tableau 23. API

Retour	Méthode	Description
`Entier (int)`	`getRowCount()`	Renvoie le nombre de lignes de cette table.
`Entier (int)`	`getColumnCount()`	Renvoie le nombre de colonnes de cette table.
`Chaîne`	`getColumnName(int columnIndex)`	Renvoie le nom de la colonne à l'index de colonne spécifié. L'index de colonne démarre à 0.
`Type de stockage (StorageType)`	`getStorageType(int columnIndex)`	Renvoie le type de stockage de la colonne à l'index spécifié. L'index de colonne démarre à 0.
`Objet`	`getValueAt(int rowIndex, int columnIndex)`	Renvoie la valeur figurant à l'index de ligne et de colonne spécifié. Les index de ligne et de colonne démarrent à 0.
`vide`	`reset()`	Vide la mémoire interne associée à ce modèle de contenu.

Noeuds et sorties

Ce tableau répertorie les noeuds qui génèrent des sorties qui incluent ce type de modèle de contenu.

Tableau 24. Noeuds et sorties

Nom du noeud	Nom de la sortie	ID conteneur
table	table	"table"

Exemple de script

```
stream = modeler.script.stream()
from modeler.api import StorageType

# Set up the variable file import node
varfilenode = stream.createAt("variablefile", "DRUG Data", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")

# Next create the aggregate node and connect it to the variable file node
aggregatenode = stream.createAt("aggregate", "Aggregate", 192, 96)
stream.link(varfilenode, aggregatenode)

# Configure the aggregate node
aggregatenode.setPropertyValue("keys", ["Drug"])
aggregatenode.setKeyedPropertyValue("aggregates", "Age", ["Min", "Max"])
aggregatenode.setKeyedPropertyValue("aggregates", "Na", ["Mean", "SDev"])

# Then create the table output node and connect it to the aggregate node
tablenode = stream.createAt("table", "Table", 288, 96)
stream.link(aggregatenode, tablenode)

# Execute the table node and capture the resulting table output object
results = []
tablenode.run(results)
tableoutput = results[0]

# Access the table output's content model
tablecontent = tableoutput.getContentModel("table")

# For each column, print column name, type and the first row
# of values from the table content
col = 0
while col < tablecontent.getColumnCount():
     print tablecontent.getColumnName(col), \
     tablecontent.getStorageType(col), \
     tablecontent.getValueAt(0, col)
     col = col + 1
```

The output in the scripting Debug tab will look something like this:

```
Age_Min Integer 15
Age_Max Integer 74
Na_Mean Real 0.730851098901
Na_SDev Real 0.116669731242
Drug String drugY
Record_Count Integer 91
```

Modèle de contenu XML

Le modèle de contenu XML donne accès au contenu XML.

Le modèle de contenu XML prend en charge la capacité à accéder aux composants basés sur des expressions XPath. Les expressions XPath sont des chaînes qui définissent les éléments ou les attributs qui sont requis par l'appelant. Le modèle de contenu XML masque les détails de la génération de divers objet et de différentes expressions de compilation qui sont généralement nécessaires pour la prise en charge de XPath. Cela simplifie l'appel depuis les scripts Python.

Le modèle de contenu XML inclut une fonction qui renvoie le document XML sous forme d'une chaîne. Cela permet à des utilisateurs de script Python d'utiliser leur bibliothèque Python préférée pour analyser le code XML.

API

Tableau 25. API

Retour	Méthode	Description
`chaîne`	`getXMLAsString()`	Renvoie le code XML sous forme de chaîne.
`nombre`	`getNumericValue(String xpath)`	Renvoie le résultat de l'évaluation du chemin avec un type de retour numérique (par exemple, comptage du nombre d'éléments qui correspondent à l'expression de chemin).
`booléen`	`getBooleanValue(String xpath)`	Renvoie le résultat booléen de l'évaluation de l'expression de chemin spécifiée.
`Chaîne`	`getStringValue(String xpath, String attribute)`	Renvoie la valeur d'attribut ou la valeur de noeud XML qui correspond au chemin spécifié.
`Liste de chaînes`	`getStringValues(String xpath, String attribute)`	Renvoie la liste de toutes les valeurs d'attribut ou les valeurs de noeud XML qui correspondent au chemin spécifié.
`Liste de listes de chaînes`	`getValuesList(String xpath, <Liste de chaînes> attributes, boolean includeValue)`	Renvoie la liste de toutes les valeurs d'attribut qui correspondent au chemin spécifié avec la valeur de noeud XML si nécessaire.
`Table de hachage (key:string, value:liste de chaînes)`	`getValuesMap(String xpath, String keyAttribute, <Liste de chaînes> attributes, boolean includeValue)`	Renvoie une table de hachage qui utilise l'attribut de clé ou la valeur de noeud XML en tant que clé, et la liste des valeurs d'attribut spécifiées en tant que valeurs de table.
`booléen`	`isNamespaceAware()`	Indique si les analyseurs syntaxiques XML doivent prendre en compte les espaces de nom. La valeur par défaut est `False`.
`vide`	`setNamespaceAware(valeur booléenne)`	Définit si les analyseurs syntaxiques XML doivent prendre en compte les espaces de nom. Appelle également `reset()` pour s'assurer que les modifications sont sélectionnées par les appels ultérieurs.
`vide`	`reset()`	Vide la mémoire interne associée à ce modèle de contenu (par exemple, un objet DOM mis en cache).

Noeuds et sorties

Ce tableau répertorie les noeuds qui génèrent des sorties qui incluent ce type de modèle de contenu.

Tableau 26. Noeuds et sorties

Nom du noeud	Nom de la sortie	ID conteneur
La plupart des générateurs de modèle	La plupart des modèles générés	"PMML"
"autodataprep"	n/a	"PMML"

Exemple de script

Le code de scriptage Python permettant d'accéder au contenu peut ressembler à ceci :

```
results = []
modelbuilder.run(results)
modeloutput = results[0]
cm = modeloutput.getContentModel("PMML")

dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
predictedNames = cm.getStringValues
("//MiningSchema/MiningField[@usageType='predicted']", "name")
```

Modèle de contenu JSON

Le modèle de contenu JSON est utilisé pour assurer la prise en charge du contenu au format JSON. Il fournit une interface de programme d'application (API) de base pour permettre aux appelants d'extraire des valeurs dans l'hypothèse où ils savent à quelles valeurs accéder.

API

Tableau 27. API

Retour	Méthode	Description
Chaîne	getJSONAsString()	Renvoie le contenu JSON sous forme de chaîne.
Objet	getObjectAt(<Liste d'objets> path, JSONArtifact artefact) throws Exception	Renvoie l'objet dans le chemin spécifié. L'artefact racine fourni peut être NULL auquel cas la racine du contenu est utilisée. La valeur renvoyée peut être une chaîne littérale, un entier, un nombre réel ou une valeur booléenne, ou un artefact JSON (un objet JSON ou un tableau JSON).
Table de hachage (key:object, value:object>	getChildValuesAt(<Liste d'objets> path, JSONArtifact artefact) throws Exception	Renvoie les valeurs enfant du chemin spécifié si le chemin mène à un objet JSON, sinon renvoie NULL. Les clés dans la table sont des chaînes tandis que la valeur associée peut être une chaîne littérale, un entier, un nombre réel ou une valeur booléenne, ou un artefact JSON (un objet JSON ou un tableau JSON).

Tableau 27. API (suite)

Retour	Méthode	Description
`Liste d'objets`	`getChildrenAt(<Liste d'objets> path chemin, JSONArtifact artefact) throws Exception`	Renvoie la liste d'objets du chemin spécifié si le chemin mène à un tableau JSON, sinon renvoie NULL. Les valeurs renvoyées peuvent être une chaîne littérale, un entier, un nombre réel ou une valeur booléenne, ou un artefact JSON (un objet JSON ou un tableau JSON).
`vide`	`reset()`	Vide la mémoire interne associée à ce modèle de contenu (par exemple, un objet DOM mis en cache).

Exemple de script

S'il existe un noeud générateur de sortie basé sur le format JSON, le code suivant peut être utilisé pour accéder à des informations sur un ensemble de livres :

```
results = []
outputbuilder.run(results)
output = results[0]
cm = output.getContentModel("jsonContent")

bookTitle = cm.getObjectAt(["books", "ISIN123456", "title"], None)

# Alternatively, get the book object and use it as the root
# for subsequent entries
book = cm.getObjectAt(["books", "ISIN123456"], None)
bookTitle = cm.getObjectAt(["title"], book)

# Get all child values for aspecific book
bookInfo = cm.getChildValuesAt(["books", "ISIN123456"], None)

# Get the third book entry. Assumes the top-level "books" value
# contains a JSON array which can be indexed
bookInfo = cm.getObjectAt(["books", 2], None)

# Get a list of all child entries
allBooks = cm.getChildrenAt(["books"], None)
```

Modèle de contenu de colonne de statistiques et modèle de contenu de statistiques par paire

Le modèle de contenu de colonne de statistiques donne accès à des statistiques qui peuvent être calculées pour chaque champ (statistiques univariées). Le modèle de contenu de statistiques par pair permet d'accéder à des statistiques qui peuvent être calculées entre des paires de champs ou de valeurs dans un champ.

Les mesures de statistiques possibles sont les suivantes :

- `Effectif`
- `UniqueCount`
- `ValidCount`
- `Moyenne`
- `Somme`
- `Min`

- Max
- Range
- Variance
- StandardDeviation
- StandardErrorOfMean
- Asymétrie
- SkewnessStandardError
- Kurtosis
- KurtosisStandardError
- Médiane
- Mode
- Pearson
- Covariance :
- TTest
- FTest

Certaines valeurs ne conviennent que pour les statistiques de colonne unique alors que d'autres ne sont appropriées que pour les statistiques par paire.

Les noeuds qui génèrent ces statistiques sont les suivants :

- Le **noeud Statistiques** génère des statistiques de colonne et peut produire des statistiques par paire lorsque des champs de corrélation sont spécifiés.
- Le **noeud Audit données** génère des statistiques de colonne et peut produire des statistiques par paire lorsqu'un champ de superposition est spécifié.
- Le **noeud Moyennes** génère des statistiques par paire lors de la comparaison des valeurs d'un champ avec d'autres récapitulatifs de champ.

Les modèles de contenu et les statistiques qui sont disponibles dépendent des capacités du noeud et des paramètres au sein de ce noeud.

API ColumnStatsContentModel

Tableau 28. API ColumnStatsContentModel.

Retour	Méthode	Description
List<Type de statistiques>	getAvailableStatistics()	Renvoie les statistiques disponibles dans ce modèle. Tous les champs n'auront pas forcément des valeurs pour toutes les statistiques.
List<Chaîne>	getAvailableColumns()	Renvoie les noms des colonnes pour lesquelles des statistiques sont calculées.
Nombre	getStatistic(String column, StatisticType statistic)	Renvoie les valeurs statistiques associées à la colonne.
vide	reset()	Vide la mémoire interne associée à ce modèle de contenu.

API PairwiseStatsContentModel

Tableau 29. API PairwiseStatsContentModel.

Retour	Méthode	Description
List<Type de statistiques>	getAvailableStatistics()	Renvoie les statistiques disponibles dans ce modèle. Tous les champs n'auront pas forcément des valeurs pour toutes les statistiques.
List<Chaîne>	getAvailablePrimaryColumns()	Renvoie les noms des colonnes primaires pour lesquelles des statistiques ont été calculées.
List<Objet>	getAvailablePrimaryValues()	Renvoie les valeurs de la colonne primaire pour laquelle des statistiques ont été calculées.
List<Chaîne>	getAvailableSecondaryColumns()	Renvoie les noms des colonnes secondaires pour lesquelles des statistiques ont été calculées.
Nombre	getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)	Renvoie les valeurs statistiques associées aux colonnes.
Nombre	getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)	Renvoie les valeurs statistiques associées à la colonne primaire et à la colonne secondaire.
vide	reset()	Vide la mémoire interne associée à ce modèle de contenu.

Noeuds et sorties

Ce tableau répertorie les noeuds qui génèrent des sorties qui incluent ce type de modèle de contenu.

Tableau 30. Noeuds et sorties.

Nom du noeud	Nom de la sortie	ID conteneur	Remarques
"means" (Noeud Moyennes)	"means"	"columnStatistics"	
"means" (Noeud Moyennes)	"means"	"pairwiseStatistics"	
"dataaudit" (Noeud Audit données)	"means"	"columnStatistics"	
"statistics" (Noeud Statistiques)	"statistics"	"columnStatistics"	Généré uniquement lorsque des champs spécifiques sont examinés.
"statistics" (Noeud Statistiques)	"statistics"	"pairwiseStatistics"	Généré uniquement lorsque des champs sont corrélés.

Exemple de script

```
from modeler.api import StatisticType
stream = modeler.script.stream()

# Set up the input data
varfile = stream.createAt("variablefile", "File", 96, 96)
varfile.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
```

```
# Now create the statistics node. This can produce both
# column statistics and pairwise statistics
statisticsnode = stream.createAt("statistics", "Stats", 192, 96)
statisticsnode.setPropertyValue("examine", ["Age", "Na", "K"])
statisticsnode.setPropertyValue("correlate", ["Age", "Na", "K"])
stream.link(varfile, statisticsnode)

results = []
statisticsnode.run(results)
statsoutput = results[0]
statscm = statsoutput.getContentModel("columnStatistics")
if (statscm != None):
 cols = statscm.getAvailableColumns()
 stats = statscm.getAvailableStatistics()
 print "Column stats:", cols[0], str(stats[0]), " = ", statscm.getStatistic(cols[0], stats[0])

statscm = statsoutput.getContentModel("pairwiseStatistics")
if (statscm != None):
 pcols = statscm.getAvailablePrimaryColumns()
 scols = statscm.getAvailableSecondaryColumns()
 stats = statscm.getAvailableStatistics()
 corr = statscm.getStatistic(pcols[0], scols[0], StatisticType.Pearson)
 print "Pairwise stats:", pcols[0], scols[0], " Pearson = ", corr
```

Chapitre 6. Arguments de ligne de commande

Appel du logiciel

Vous pouvez utiliser la ligne de commande de votre système d'exploitation pour lancer IBM SPSS Modeler comme suit :

1. Dans le cas d'un ordinateur sur lequel est installé IBM SPSS Modeler, ouvrez une fenêtre DOS ou une invite de commande.
2. Pour lancer l'interface IBM SPSS Modeler en mode interactif, tapez la commande `modelerclient` suivie des arguments souhaités, par exemple :

```
modelerclient -stream report.str -execute
```

Les arguments disponibles (indicateurs) vous permettent de vous connecter à un serveur, de charger des flux, d'exécuter des scripts ou d'indiquer les autres paramètres nécessaires.

Utilisation d'arguments de ligne de commande

Vous pouvez ajouter des arguments de ligne de commande (également appelés *indicateurs*) à la commande `modelerclient` initiale pour modifier l'appel de IBM SPSS Modeler.

Plusieurs types d'arguments de ligne de commande sont disponibles et décrits plus loin dans cette section.

Tableau 31. Types d'arguments de ligne de commande.

Type d'argument	Description
Arguments système	Pour plus d'informations, voir la rubrique «Arguments système», à la page 62.
Arguments de paramètre	Pour plus d'informations, voir la rubrique «Arguments de paramètre», à la page 63.
Arguments de connexion au serveur	Pour plus d'informations, voir la rubrique «Arguments de connexion au serveur», à la page 64.
Arguments de connexion au IBM SPSS Collaboration and Deployment Services Repository	Pour plus d'informations, voir la rubrique «IBM SPSS Collaboration and Deployment Services Repository Arguments de connexion», à la page 65.
Arguments de connexion au IBM SPSS Analytic Server	Pour plus d'informations, reportez-vous à la rubrique «Arguments de connexion à IBM SPSS Analytic Server», à la page 65.

Vous pouvez par exemple utiliser les indicateurs `-server`, `-stream` et `-execute` pour vous connecter à un serveur, puis charger et exécuter un flux, comme indiqué ci-dessous :

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

Notez que lors d'une exécution en parallèle avec l'installation d'un client local, les arguments de connexion au serveur ne sont pas obligatoires.

Il est possible de placer entre guillemets doubles les valeurs de paramètre qui contiennent des espaces. Par exemple :

```
modelerclient -stream mystream.str -Pusername="Joe User" -execute
```

Vous pouvez également exécuter des états et des scripts IBM SPSS Modeler de la même manière, en utilisant respectivement les indicateurs `-state` et `-script` .

Remarque : Si vous utilisez un paramètre structuré dans une commande, vous devez faire précéder les guillemets par une barre oblique inversée. Ainsi, les guillemets ne seront pas supprimés lors de l'interprétation de la chaîne.

Débogage d'arguments de ligne de commande

Pour déboguer une ligne de commande, utilisez la commande `modelerclient` afin de lancer IBM SPSS Modeler avec les arguments souhaités. Vous avez ainsi la possibilité de vérifier que les commandes s'exécutent comme souhaité. Vous pouvez également vérifier les valeurs de paramètre transmises depuis la ligne de commande dans la boîte de dialogue Paramètres de session (menu Outils, Définir les paramètres de session).

Arguments système

Le tableau suivant décrit les arguments système disponibles pour l'appel de la ligne de commande de l'interface utilisateur.

Tableau 32. Arguments système

Argument	Comportement/description
`@` <commandFile>	Le caractère `@` suivi d'un nom de fichier indique une liste de commandes. Lorsque `modelerclient` détecte un argument commençant par le caractère `@` , il agit sur les commandes du fichier correspondant comme si elles s'étaient trouvées dans la ligne de commande. Pour plus d'informations, voir la rubrique «Combinaison de plusieurs arguments», à la page 66.
`-directory` <dir>	Définit le répertoire de travail par défaut. En mode local, ce répertoire est utilisé pour les données et la sortie. Exemple : `-directory c:/` ou `-directory c:\\`
`-server_directory` <dir>	Définit le répertoire du serveur par défaut pour les données. Le répertoire de travail, défini à l'aide de l'indicateur `-directory` est utilisé pour la sortie.
`-execute`	Après le démarrage, exécute tout flux, état ou script chargé au démarrage. Si un script est chargé en plus d'un flux ou d'un état, seul le script est exécuté.
`-stream` <flux>	Au démarrage, charge le flux spécifié. Vous pouvez spécifier plusieurs flux, mais le dernier flux sera défini comme flux en cours.
`-script` <script>	Charge, au démarrage, le script autonome spécifié. Vous pouvez le spécifier en plus d'un flux ou d'un état comme décrit ci-dessous, mais un seul script peut être chargé au démarrage.
`-model` <modèle>	Au démarrage, charge le modèle généré (fichier au format *.gm*) indiqué.
`-state` <état>	Au démarrage, charge l'état enregistré indiqué.
`-project` <projet>	Charge le projet spécifié. Vous ne pouvez charger qu'un seul projet au démarrage.
`-output` <sortie>	Au démarrage, charge l'objet de sortie enregistré (fichier au format *.cou*).
`-help`	Affiche la liste des arguments de ligne de commande. Lorsque cette option est précisée, tous les autres arguments sont ignorés et l'écran Aide apparaît.
`-P` <name>=<value>	Sert à définir un paramètre de démarrage. Peut également être utilisé pour définir les propriétés de noeud (paramètres de propriété).

Remarque : vous pouvez également définir les répertoires par défaut dans l'interface utilisateur. Pour accéder aux options, dans le menu Fichier, sélectionnez **Définir le répertoire de travail** ou **Définir le répertoire du serveur**.

Chargement de plusieurs fichiers

A partir de la ligne de commande, vous pouvez charger plusieurs flux, états et sorties au démarrage en répétant l'argument applicable pour chaque objet chargé. Par exemple, pour charger et exécuter deux flux nommés *report.str* et *train.str*, utilisez la commande suivante :

```
modelerclient -stream report.str -stream train.str -execute
```

Chargement d'objets du IBM SPSS Collaboration and Deployment Services Repository

Etant donné que vous pouvez charger certains objets à partir d'un fichier ou du IBM SPSS Collaboration and Deployment Services Repository (si vous disposez de la licence correspondante), le préfixe de nom de fichier `spsscr:` et éventuellement `fichier:` (pour les objets sur disque) indique à IBM SPSS Modeler l'emplacement de l'objet. Le préfixe fonctionne avec les indicateurs suivants :

- `-stream`
- `-script`
- `-output`
- `-model`
- `-project`

Vous utilisez le préfixe pour créer un URI qui indique l'emplacement de l'objet. Par exemple : `-stream "spsscr:///folder_1/scoring_stream.str"`. Lorsque le préfixe `spsscr:` est présent, vous devez définir dans la même commande une connexion valide au IBM SPSS Collaboration and Deployment Services Repository. Ainsi, par exemple, la commande complète serait semblable à ce qui suit :

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

A partir de la ligne de commande, vous *devez* utiliser un URI. L'élément simple `REPOSITORY_PATH` n'est pas pris en charge. (Il fonctionne uniquement au sein des scripts.) Pour plus de détails sur les URI relatifs aux objets du IBM SPSS Collaboration and Deployment Services Repository, voir la rubrique «Accès aux objets du IBM SPSS Collaboration and Deployment Services Repository», à la page 50.

Arguments de paramètre

Les paramètres peuvent être utilisés en tant qu'indicateurs pendant l'exécution de la ligne de commande de IBM SPSS Modeler. Dans les arguments de ligne de commande, l'indicateur `-P` sert à indiquer un paramètre du type `-P` *<nom>=<valeur>*.

Il peut s'agir des paramètres suivants :

- Des **paramètres simples** (ou des paramètres utilisés directement dans les expressions CLEM).
- Des **paramètres de propriété**, également appelés **propriétés de noeud**. Ces paramètres servent à modifier les paramètres des noeuds du flux. Pour plus d'informations, voir la rubrique «Présentation des propriétés de noeud», à la page 69.
- Des **paramètres de ligne de commande**, utilisés pour modifier l'appel de IBM SPSS Modeler.

Par exemple, vous pouvez fournir les noms d'utilisateur et les mots de passe d'une source de données sous la forme d'un indicateur de ligne de commande, comme l'illustre l'exemple suivant :

```
modelerclient -stream response.str -P:databasenode.datasource={"ORA 10gR2", user1, mypsw, true}
```

Le format est identique à celui du paramètre `datasource` de la propriété du noeud `databasenode`. Pour plus d'informations, voir «Propriétés de databasenode», à la page 81.

Remarque : Si le noeud est nommé, vous devez placer son nom entre guillemets et utiliser une barre oblique inversée comme caractère d'échappement pour les guillemets. Par exemple, si le noeud de source de données de l'exemple précédent porte le nom *Source_ABC*, l'entrée doit se présenter comme suit :

```
modelerclient -stream response.str -P:databasenode.\"Source_ABC\".datasource={"ORA 10gR2",
 user1, mypsw, true}
```

Une barre oblique inversée est également obligatoire devant les guillemets qui identifient un paramètre structuré, comme dans l'exemple de source de données TM1 suivant :

```
clemb -server -hostname 9.115.21.169 -port 28053 -username administrator
    -execute -stream C:\Share\TM1_Script.str -P:tm1import.pm_host="http://9.115.21.163:9510/pmhub/pm"
    -P:tm1import.tm1_connection={\"SData\",\"\",\"admin\",\"apple\"}
    -P:tm1import.selected_view={\"SalesPriorCube\",\"salesmargin%\"}
```

Arguments de connexion au serveur

L'indicateur `-server` indique à IBM SPSS Modeler de se connecter à un serveur public et les indicateurs `-hostname`, `-use_ssl`, `-port`, `-username`, `-password`, et `-domain` sont utilisés pour indiquer à IBM SPSS Modeler comment se connecter au serveur public. Si aucun argument -server n'est spécifié, le serveur par défaut ou le serveur local est utilisé.

Exemples

Pour vous connecter à un serveur public :

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

Pour vous connecter à un cluster de serveurs :

```
modelerclient -server -cluster "QA Machines" \
-spsscr_hostname pes_host -spsscr_port 8080 \
-spsscr_username asmith -spsscr_epassword xyz
```

Veuillez noter que la connexion à un groupe de serveur nécessite le coordinateur de processus avec IBM SPSS Collaboration and Deployment Services. Par conséquent, l'argument `-cluster` doit être utilisé conjointement avec les options de connexion du référentiel (`spsscr_*`). Pour plus d'informations, voir la rubrique «IBM SPSS Collaboration and Deployment Services Repository Arguments de connexion», à la page 65.

Tableau 33. Arguments de connexion au serveur.

Argument	Comportement/description
`-server`	Exécute IBM SPSS Modeler en mode serveur, en se connectant à un serveur public à l'aide des indicateurs `-hostname`, `-port`, `-username`, `-password` et `-domain`.
`-hostname <name>`	Nom d'hôte du serveur. Disponible en mode serveur uniquement.
`-use_ssl`	Indique que la connexion doit utiliser le protocole SSL (Secure Socket Layer). Cet indicateur est facultatif ; par défaut, le protocole SSL n'est *pas* utilisé.
`-port <number>`	Numéro de port du serveur spécifié. Disponible en mode serveur uniquement.
`-cluster <name>`	Spécifie une connexion à un cluster de serveurs plutôt qu' à un serveur nommé ; cet argument est une alternative aux arguments `hostname`, `port` et `use_ssl` . Le nom est le nom de groupe ou une URI unique qui identifie le groupe dans IBM SPSS Collaboration and Deployment Services Repository. Le cluster de serveurs est géré par le coordinateur de processus dans IBM SPSS Collaboration and Deployment Services. Pour plus d'informations, voir la rubrique «IBM SPSS Collaboration and Deployment Services Repository Arguments de connexion», à la page 65.
`-username <name>`	Nom d'utilisateur utilisé pour la connexion au serveur. Disponible en mode serveur uniquement.
`-password <password>`	Mot de passe utilisé pour la connexion au serveur. Disponible en mode serveur uniquement. *Remarque* : si l'argument `-password` n'est pas utilisé, le système vous invite à entrer un mot de passe.

Tableau 33. Arguments de connexion au serveur (suite).

Argument	Comportement/description
`-epassword` <encodedpasswordstring>	Mot de passe codé utilisé pour la connexion au serveur. Disponible en mode serveur uniquement. *Remarque* : vous pouvez créer un mot de passe codé à partir du menu Outils de l'application IBM SPSS Modeler.
`-domain` <name>	Domaine utilisé pour la connexion au serveur. Disponible en mode serveur uniquement.
`-P` <name>=<value>	Sert à définir un paramètre de démarrage. Peut également être utilisé pour définir les propriétés de noeud (paramètres de propriété).

IBM SPSS Collaboration and Deployment Services Repository Arguments de connexion

Pour pouvoir stocker ou extraire des objets du IBM SPSS Collaboration and Deployment Services via la ligne de commande, vous devez indiquer une connexion valide au IBM SPSS Collaboration and Deployment Services Repository. Par exemple :

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

Le tableau suivant répertorie les arguments qu'il convient d'utiliser pour paramétrer la connexion.

Tableau 34. Arguments de connexion au IBM SPSS Collaboration and Deployment Services Repository

Argument	Comportement/description
`-spsscr_hostname` <nom d'hôte ou adresse IP>	Nom d'hôte ou adresse IP du serveur sur lequel est installé IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_port` <numéro>	Numéro de port sur lequel IBM SPSS Collaboration and Deployment Services Repository accepte les connexions (8080 par défaut).
`-spsscr_use_ssl`	Indique que la connexion doit utiliser le protocole SSL (Secure Socket Layer). Cet indicateur est facultatif ; par défaut, le protocole SSL n'est *pas* utilisé.
`-spsscr_username` <nom>	Nom d'utilisateur utilisé pour la connexion au IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_password` <mot de passe>	Mot de passe utilisé pour la connexion au IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_epassword` <mot de passe codé>	Mot de passe codé utilisé pour la connexion au IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_domain` <nom>	Domaine utilisé pour la connexion au IBM SPSS Collaboration and Deployment Services Repository. Cet indicateur est facultatif ; ne l'utilisez que si vous vous connectez via LDAP ou Active Directory.

Arguments de connexion à IBM SPSS Analytic Server

Pour pouvoir stocker ou extraire des objets de IBM SPSS Analytic Server via la ligne de commande, vous devez indiquer une connexion valide à IBM SPSS Analytic Server.

Remarque : L'emplacement d'Analytic Server est obtenu à partir de SPSS Modeler Server et ne peut pas être modifié sur le client.

Le tableau suivant répertorie les arguments qu'il convient d'utiliser pour paramétrer la connexion.

Tableau 35. Arguments de connexion au IBM SPSS Analytic Server

Argument	Comportement/description
`-analytic_server_username`	Nom d'utilisateur utilisé pour la connexion à IBM SPSS Analytic Server.
`-analytic_server_password`	Mot de passe utilisé pour la connexion à IBM SPSS Analytic Server.
`-analytic_server_epassword`	Mot de passe codé utilisé pour la connexion à IBM SPSS Analytic Server.
`-analytic_server_credential`	Données d'identification utilisées pour la connexion à IBM SPSS Analytic Server.

Combinaison de plusieurs arguments

Utilisez le symbole @ suivi du nom du fichier pour combiner plusieurs arguments dans un fichier de commande unique défini lors de l'appel de la commande. Vous pouvez ainsi raccourcir l'appel via la ligne de commande et remédier aux éventuelles limites appliquées à la longueur des commandes par les systèmes d'exploitation. Par exemple, la commande de démarrage suivante utilise les arguments spécifiés dans le fichier référencé par `<commandFileName>`.

```
modelerclient @<commandFileName>
```

Si vous devez utiliser des espaces, placez le nom et le chemin d'accès du fichier entre guillemets, comme indiqué ci-dessous :

```
modelerclient @ "C:\Program Files\IBM\SPSS\Modeler\nn\scripts\my_command_file.txt"
```

Le fichier de commande peut contenir tous les arguments qui étaient auparavant spécifiés séparément au démarrage, avec un argument par ligne. Par exemple :

```
-stream report.str
-Porder.full_filename=APR_orders.dat
-Preport.filename=APR_report.txt
-execute
```

Lorsque vous rédigez des fichiers de commande et que vous y faites référence, veillez à respecter les contraintes suivantes :

- N'inscrivez qu'une commande par ligne.
- N'intégrez pas un argument `@CommandFile` au sein d'un fichier de commande.

Chapitre 7. Référence sur les propriétés

Présentation des références sur les propriétés

Vous pouvez indiquer un certain nombre de propriétés pour les noeuds, les flux, les super noeuds et les projets. Certaines propriétés sont communes à tous les noeuds, telles que le nom, l'annotation et l'info-bulle, alors que d'autres sont propres à certains types de noeud. D'autres propriétés font référence aux opérations de flux de haut niveau, comme la mise en cache ou le comportement du super noeud. Vous pouvez accéder aux propriétés via l'interface utilisateur standard (par exemple, lorsque vous ouvrez une boîte de dialogue pour modifier les options d'un noeud) et les utiliser de plusieurs manières.

- Vous pouvez modifier les propriétés via des scripts, comme l'explique cette section. Pour plus d'informations, voir rubrique «Syntaxe des propriétés».
- Les propriétés de noeud peuvent être utilisées dans les paramètres du super noeud.
- Les propriétés de noeud peuvent également être utilisées dans le cadre d'une option de ligne de commande (en utilisant l'indicateur `-P`) lors du démarrage de IBM SPSS Modeler.

Dans le cadre de la génération de scripts dans IBM SPSS Modeler, les propriétés de noeud et de flux sont souvent appelées **paramètres de propriété**. Dans cette aide, elles sont appelées propriétés de noeud ou de flux.

Pour plus d'informations sur le langage de script, voir Langage de script.

Syntaxe des propriétés

Les propriétés peuvent être définies à l'aide de la syntaxe suivante :

```
OBJECT.setPropertyValue(PROPERTY, VALUE)
```

ou :

```
OBJECT.setKeyedPropertyValue(PROPERTY, KEY, VALUE)
```

La valeur des propriétés peuvent être extraites à l'aide de la syntaxe suivante :

```
VARIABLE = OBJECT.getPropertyValue(PROPERTY)
```

ou :

```
VARIABLE = OBJECT.getKeyedPropertyValue(PROPERTY, KEY)
```

Où `OBJECT` est un noeud ou une sortie, `PROPERTY` est le nom de la propriété de noeud à laquelle votre expression fait référence et `KEY` est la valeur de clé pour des propriétés saisies. Par exemple, la syntaxe suivante est utilisée pour trouver le noeud filtre, puis définir la valeur par défaut pour inclure tous les champs et filtrer le champ `Age` des données en aval :

```
filternode = modeler.script.stream().findByType("filter", None)
filternode.setPropertyValue("default_include", True)
filternode.setKeyedPropertyValue("include", "Age", False)
```

Tous les noeuds utilisés dans IBM SPSS Modeler peuvent être recherchés à l'aide de la fonction `findByType(TYPE, LABEL)` de flux. Au moins `TYPE` ou `LABEL` doit être spécifié.

Propriétés structurées

Pour obtenir une plus grande clarté lors de l'analyse, la génération de scripts peut utiliser les propriétés structurées de deux façons différentes :

- Attribuer une structure aux noms des propriétés des noeuds complexes, tels que les noeuds types, Filtrer ou Equilibrer.

- Fournir un format afin de spécifier plusieurs propriétés en une opération.

Structuration des interfaces complexes

Les scripts des noeuds contenant des tableaux ou d'autres interfaces complexes (par exemple, les noeuds types, Filtrer et Equilibrer) doivent suivre une structure particulière afin de procéder à une analyse correcte. Ces propriétés ont besoin d'un nom plus complexe que le nom correspondant à un identificateur unique. Ce nom est appelé "clé". Par exemple, dans un noeud Filtrer, chaque champ disponible (en amont) peut être activé ou désactivé. Pour faire référence à ces informations, le noeud Filtrer stocke une information par champ (indiquant s'il est vrai ou faux). Cette propriété peut avoir, ou se voir affecter, la valeur `True` ou `False` (vrai ou faux). Prenons l'exemple d'un noeud Filtrer appelé `mynode` et comportant, en amont, un champ appelé `Age`. Pour désactiver ce champ, définissez la propriété `include`, avec la clé `Age`, sur la valeur `False` comme suit :

```
mynode.setKeyedPropertyValue("include", "Age", False)
```

Structuration permettant de définir plusieurs propriétés

Pour de nombreux noeuds, vous pouvez affecter plusieurs propriétés de noeud ou de flux à la fois. Cette opération est appelée **définition globale** ou **définition en bloc**. Pour plus d'informations, voir la rubrique Commande set.

Les propriétés structurées peuvent parfois être très complexes. En voici un exemple :

```
sortnode.setPropertyValue("keys", [["K", "Descending"], ["Age", "Ascending"],
["Na", "Descending"]])
```

Les propriétés structurées offrent également la possibilité de définir plusieurs propriétés d'un noeud avant que le noeud soit stable. Par défaut, une définition globale définit toutes les propriétés d'un bloc avant d'agir en fonction du paramétrage d'une propriété spécifique. Par exemple, lors de la définition d'un noeud Fixe, la spécification des propriétés de champs en deux étapes génère des erreurs dans la mesure où le noeud n'est pas cohérent tant que les deux paramètres ne sont pas valides. Le fait de définir les propriétés en utilisant une définition globale permet d'éviter ce problème dans la mesure où les deux propriétés sont définies avant que le modèle de données ne soit mis à jour.

Abréviations

Les abréviations standard sont utilisées tout au long de la syntaxe des propriétés de noeud. La connaissance des abréviations est utile lors de la rédaction de scripts.

Tableau 36. Abréviations standard utilisées tout au long de la syntaxe

Abréviation	Signification
abs	Absolute value
len	Longueur
min	Minimum
max	Maximum
correl	Corrélation
covar	Covariance :
num	Nombre ou numérique
pct (pourcentage)	Percent (pour cent) ou (percentage) pourcentage
transp	Transparence
xval	Cross-validation (validation croisée)
var	Variance ou variable (dans les noeuds source)

Exemple de propriétés de noeud et de flux

Dans IBM SPSS Modeler, les propriétés de noeud et de flux peuvent être utilisées de différentes façons. Ces propriétés sont le plus souvent utilisées dans le cadre d'un script, que ce soit un **script autonome,** pour automatiser plusieurs flux ou opérations, ou un **script de flux,** pour automatiser des processus au sein d'un flux unique. Vous pouvez également indiquer des paramètres de noeud à l'aide des propriétés de noeud du super noeud. Au niveau le plus basique, les propriétés peuvent aussi être utilisées sous forme d'une option de ligne de commande pour démarrer IBM SPSS Modeler. Si vous utilisez le commutateur `-p` dans le cadre de l'appel d'une ligne de commande, vous pouvez modifier un paramètre dans le flux à l'aide d'une propriété du flux.

Tableau 37. Exemples de propriétés de noeud et de flux

Propriété	Signification
`s.max_size`	Désigne la propriété `max_size` du noeud `s`.
`s:samplenode.max_size`	Désigne la propriété `max_size` du noeud `s`, qui doit être un noeud Echantillonner.
`:samplenode.max_size`	Désigne la propriété `max_size` du noeud Echantillonner du flux actuel (il ne peut exister qu'un noeud de ce type).
`s:sample.max_size`	Désigne la propriété `max_size` du noeud `s`, qui doit être un noeud Echantillonner.
`t.direction.Age`	Désigne le rôle du champ *Age* du noeud type `t`.
`:.max_size`	*** NON AUTORISE *** Vous devez indiquer le nom ou le type du noeud.

Comme l'illustre l'exemple `s:sample.max_size`, vous n'avez pas besoin d'écrire les types de noeud en entier.

L'exemple `t.direction.Age` montre que certains noms de propriété peuvent être structurés lorsque les attributs d'un noeud sont trop complexes pour pouvoir être définis par des propriétés aux valeurs individuelles. Ces propriétés sont appelées propriétés **structurées** ou **complexes**.

Présentation des propriétés de noeud

Pour chaque type de noeud, un certain nombre de propriétés sont autorisées. Chaque propriété comporte un type. Ce type peut être « général » (nombre, indicateur ou chaîne), auquel cas le type approprié est attribué automatiquement aux paramètres de la propriété. Si le type correct ne peut pas être défini, une erreur est renvoyée. Un paramètre de propriété peut également spécifier un intervalle de valeurs autorisées, telles que `Supprimer`, `ApparierEtSupprimer` et `InclureCommeTexte`. Dans ce cas, une erreur est renvoyée si une autre valeur est utilisée. Les propriétés indicateurs doivent être lues ou définies avec les valeurs `True` et `False` (vrai et faux). (Les variations possibles, y compris `Off (désactivé)`, `OFF`, `off`, `No (Non)`, `NO`, `no`, `n`, `N`, `f`, `F`, `false (faux)`, `False`, `FALSE` ou `0`, sont également reconnues pendant la définition des valeurs, mais peuvent dans certains cas générer des erreurs lors de la lecture des valeurs de propriété. Toute autre valeur sera interprétée comme étant True (vrai). Utiliser `true` et `false` de façon cohérente permet d'éviter toute confusion.) Dans les tableaux de référence du présent document, les propriétés structurées sont indiquées telles quelles dans la colonne *Description de la propriété* et sont accompagnées de la syntaxe à utiliser.

Propriétés communes des noeuds

Dans IBM SPSS Modeler, un certain nombre de propriétés sont communes à tous les noeuds (y compris les super noeuds).

Tableau 38. Propriétés communes des noeuds.

Nom de la propriété	Le type de données	Description de la propriété
use_custom_name	*indicateur*	
name	*chaîne*	Propriété en lecture seule qui lit le nom (automatique ou personnalisé) d'un noeud de l'espace de travail.
custom_name	*chaîne*	Indique le nom personnalisé du noeud.
tooltip	*chaîne*	
annotation	*chaîne*	
keywords	*chaîne*	Propriété structurée indiquant la liste de mots-clés associés à l'objet (par exemple, ["Mot-clé1" "Mot-clé2"]).
cache_enabled	*indicateur*	
node_type	source_supernode process_supernode terminal_supernode tous les noms de noeud indiqués pour la génération de scripts	Propriété en lecture seule utilisée pour faire référence à un noeud par type. Par exemple, au lieu de faire référence à un noeud uniquement par son nom, tel que revenu_réel, vous pouvez également indiquer son type (noeudutilisateur ou noeudfiltrer).

Les propriétés propres aux super noeuds sont abordées séparément, comme pour tous les autres noeuds. Pour plus d'informations, voir la rubrique Chapitre 19, «Propriétés du super noeud», à la page 303.

Chapitre 8. Propriétés de flux

La génération de scripts permet de contrôler différentes propriétés de flux. Pour référencer les propriétés de flux, vous devez définir la méthode d'exécution pour utiliser des scripts :

```
stream = modeler.script.stream()
stream.setPropertyValue("execute_method", "Script")
```

Exemple

La propriété de noeud est utilisée pour faire référence aux noeuds du flux en cours. Le script de flux suivant fournit un exemple :

```
stream = modeler.script.stream()
annotation = stream.getPropertyValue("annotation")

annotation = annotation + "\n\nThis stream is called \"" + stream.getLabel() + "\" and
 contains the following nodes:\n"

for node in stream.iterator():
    annotation = annotation + "\n" + node.getTypeName() + " node called \"" + node.getLabel()
 + "\""

stream.setPropertyValue("annotation", annotation)
```

Dans l'exemple ci-dessus, la propriété de noeud est utilisée pour créer la liste de tous les noeuds du flux et pour écrire cette liste dans les annotations du flux. L'annotation générée ressemble à l'annotation suivante :

```
This stream is called "druglearn" and contains the following nodes:

type node called "Define Types"
derive node called "Na_to_K"
variablefile node called "DRUG1n"
neuralnetwork node called "Drug"
c50 node called "Drug"
filter node called "Discard Fields"
```

Les propriétés de flux sont décrites dans le tableau suivant.

Tableau 39. Propriétés du flux.

Nom de la propriété	Le type de données	Description de la propriété
execute_method	Normal Script	

Tableau 39. Propriétés du flux (suite).

Nom de la propriété	Le type de données	Description de la propriété
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
date_baseline	*nombre*	
date_2digit_baseline	*nombre*	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
time_rollover	*indicateur*	
import_datetime_as_string	*indicateur*	
decimal_places	*nombre*	
decimal_symbol	Default Period Comma	
angles_in_radians	*indicateur*	
use_max_set_size	*indicateur*	
max_set_size	*nombre*	
ruleset_evaluation	Voting FirstHit	

Tableau 39. Propriétés du flux (suite).

Nom de la propriété	Le type de données	Description de la propriété
refresh_source_nodes	*indicateur*	Permet de rafraîchir les noeuds source automatiquement dès l'exécution du flux.
script	*chaîne*	
annotation	*chaîne*	
name	*chaîne*	**Remarque :** Cette propriété est en lecture seule. Si vous souhaitez modifier le nom d'un flux, vous devez l'enregistrer sous un autre nom.
parameters		Utilisez cette propriété pour mettre à jour les paramètres de flux à partir d'un script autonome.
nodes		Reportez-vous aux informations détaillées ci-dessous.
encoding	SystemDefault "UTF-8"	
stream_rewriting	*booléen*	
stream_rewriting_maximise_sql	*booléen*	
stream_rewriting_optimise_clem_exécution	*booléen*	
stream_rewriting_optimise_syntax_exécution	*booléen*	
enable_parallelism	*booléen*	
sql_generation	*booléen*	
database_caching	*booléen*	
sql_logging	*booléen*	
sql_generation_logging	*booléen*	
sql_log_native	*booléen*	
sql_log_prettyprint	*booléen*	
record_count_suppress_input	*booléen*	
record_count_feedback_interval	*entier*	
use_stream_auto_create_node_paramètres	*booléen*	Si la valeur est définie sur true (vrai), les paramètres spécifiques au flux sont utilisés ; sinon, les préférences utilisateur sont utilisées.
create_model_applier_for_new_modèles	*booléen*	Si la valeur est définie sur true (vrai), lorsqu'un générateur de modèle crée un nouveau modèle et qu'il ne contient aucun lien de mise à jour actif, un nouvel applicateur de modèle est ajouté. **Remarque :** Si vous utilisez IBM SPSS Modeler Batch version 15, vous devez ajouter explicitement l'applicateur de modèle dans votre script.

Tableau 39. Propriétés du flux (suite).

Nom de la propriété	Le type de données	Description de la propriété

`create_model_applier_update_links`	`createEnabled`
	`createDisabled`
	`doNotCreate`
		Définit le type de lien créé lorsqu'un noeud applicateur de modèle est automatiquement ajouté.

`create_source_node_from_builders`	*booléen*	Si la valeur est définie sur true (vrai), lorsqu'un générateur de source crée une nouvelle sortie source et qu'elle ne contient aucun lien de mise à jour actif, un nouveau noeud source est ajouté.

`create_source_node_update_links`	`createEnabled`
	`createDisabled`
	`doNotCreate`
		Définit le type de lien créé lorsqu'un noeud source est automatiquement ajouté.

`has_coordinate_system`	*booléen*	Si la valeur est true, applique un système de coordonnées à la totalité du flux.
`coordinate_system`	*chaîne*	Nom du système de coordonnées projetées sélectionné.

Chapitre 9. Propriétés des nœuds source

Propriétés communes aux nœuds source

Les propriétés communes à tous les nœuds source sont répertoriées ci-dessous, avec des informations sur certains nœuds dans les rubriques suivantes.

Exemple 1

```
varfilenode = modeler.script.stream().create("variablefile", "Var. File")
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
varfilenode.setKeyedPropertyValue("check", "Age", "None")
varfilenode.setKeyedPropertyValue("values", "Age", [1, 100])
varfilenode.setKeyedPropertyValue("type", "Age", "Range")
varfilenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Exemple 2

Ce script suppose que le fichier de données spécifié contient un champ appelé `Region` qui représente une chaîne multiligne.

```
from modeler.api import StorageType
from modeler.api import MeasureType

# Create a Variable File node that reads the data set containing
# the "Region" field
varfilenode = modeler.script.stream().create("variablefile", "My Geo Data")
varfilenode.setPropertyValue("full_filename", "C:/mydata/mygeodata.csv")
varfilenode.setPropertyValue("treat_square_brackets_as_lists", True)

# Override the storage type to be a list...
varfilenode.setKeyedPropertyValue("custom_storage_type", "Region", StorageType.LIST)
# ...and specify the type if values in the list and the list depth
varfilenode.setKeyedPropertyValue("custom_list_storage_type", "Region", StorageType.INTEGER)
varfilenode.setKeyedPropertyValue("custom_list_depth", "Region", 2)

# Now change the measurement to indentify the field as a geospatial value...
varfilenode.setKeyedPropertyValue("measure_type", "Region", MeasureType.GEOSPATIAL)
# ...and finally specify the necessary information about the specific
# type of geospatial object
varfilenode.setKeyedPropertyValue("geo_type", "Region", "MultiLineString")
varfilenode.setKeyedPropertyValue("geo_coordinates", "Region", "2D")
varfilenode.setKeyedPropertyValue("has_coordinate_system", "Region", True)
varfilenode.setKeyedPropertyValue("coordinate_system", "Region",
"ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tableau 40. Propriétés communes aux noeuds source.

Nom de la propriété	Le type de données	Description de la propriété

`direction`	`Input`
	`Target`
	`Both`
	`None`
	`Partition`
	`Split`
	`Frequency`
	`RecordID`
		Propriété saisie (rôles des champs).
		
		Syntaxe :
		`NODE.direction.FIELDNAME`
		Remarque : Les valeurs `In` et `Out` sont désormais obsolètes. Leur prise en charge pourrait être supprimée dans une version ultérieure.

Tableau 40. Propriétés communes aux noeuds source (suite).

Nom de la propriété	Le type de données	Description de la propriété
type	Range Indicateur Set Sans type Discrete Ordered Set Default	Type de champ. Si vous paramétrez cette propriété sur *Par défaut*, toute définition de propriété de values est effacée. Si la valeur value_mode est paramétrée sur *Spécifier*, elle est redéfinie sur *Lire*. Si la propriété value_mode est paramétrée sur *Transférer* ou *Lire*, la définition du type n'a aucune incidence sur elle. Syntaxe : NODE.type.FIELDNAME
storage	Inconnu String Integer Real Time Date Timestamp	Propriété saisie en lecture seule pour le type de stockage de champ. Syntaxe : NODE.storage.FIELDNAME
check	None Nullify Coerce Discard Warn Abort	Propriété saisie (vérification du type et de l'intervalle des champs). Syntaxe : NODE.check.FIELDNAME
values	[*valeur valeur*]	Pour un champ continu (intervalle), la première valeur représente la valeur minimale et la dernière valeur, la valeur maximale. Pour des champs nominaux (ensemble), spécifiez toutes les valeurs. Pour les champs indicateurs, la première valeur représente *false* (faux) et la dernière valeur, *true* (vrai). La définition automatique de cette propriété paramètre la propriété value_mode sur *Specify*. Le stockage est déterminé en fonction de la première valeur de la liste. Par exemple, si la première valeur est une *chaîne*, le stockage est défini sur Chaîne (String). Syntaxe : NODE.values.FIELDNAME
value_mode	Read Pass Lire + Current Specify	Détermine la façon dont les valeurs sont déterminées pour un champ lors du passage suivant des données. Syntaxe : NODE.value_mode.FIELDNAME Remarque : vous ne pouvez pas paramétrer directement cette propriété sur *Spécifier*. Pour utiliser des valeurs spécifiques, paramétrez la propriété values.
default_value_mode	Read Pass	Indique la méthode par défaut de définition des valeurs de tous les champs. Syntaxe : NODE.default_value_mode Ce paramètre peut être ignoré pour certains champs à l'aide de la propriété value_mode.

Tableau 40. Propriétés communes aux noeuds source (suite).

Nom de la propriété	Le type de données	Description de la propriété
extend_values	*indicateur*	S'applique lorsque la propriété value_mode est paramétrée sur *Read*. Paramétrez cette valeur sur *T* pour ajouter des valeurs qui viennent d'être lues aux valeurs existantes du champ. Paramétrez cette valeur sur *F* pour supprimer des valeurs existantes en faveur des valeurs qui viennent d'être lues. Syntaxe : NODE.extend_values.FIELDNAME
value_labels	*chaîne*	Permet d'utiliser un libellé de valeur. Vous devez d'abord indiquer les valeurs.
enable_missing	*indicateur*	Lorsque cette propriété est paramétrée sur *T*, elle active le suivi des valeurs manquantes du champ. Syntaxe : NODE.enable_missing.FIELDNAME
missing_values	[*valeur valeur ...*]	Spécifie les valeurs de données qui indiquent les données manquantes. Syntaxe : NODE.missing_values.FIELDNAME
range_missing	*indicateur*	Lorsque cette propriété est définie sur *T*, spécifie si un intervalle de valeurs manquantes (vides) est défini pour un champ. Syntaxe : NODE.range_missing.FIELDNAME
missing_lower	*chaîne*	Lorsque le paramètre range_missing a pour valeur true (vrai), indique la limite inférieure de l'intervalle des valeurs manquantes. Syntaxe : NODE.missing_lower.FIELDNAME
missing_upper	*chaîne*	Lorsque le paramètre range_missing a pour valeur true, indique la limite supérieure de l'intervalle des valeurs manquantes. Syntaxe : NODE.missing_upper.FIELDNAME
null_missing	*indicateur*	Lorsque cette propriété est paramétrée sur *T*, les valeurs nulles (valeurs non définies affichées sous la forme $null$ dans le logiciel) sont considérées comme des valeurs manquantes. Syntaxe : NODE.null_missing.FIELDNAME
whitespace_missing	*indicateur*	Lorsque cette propriété est paramétrée sur *T*, les valeurs composées uniquement d'espaces blancs (espaces, tabulations et caractères de nouvelle ligne) sont considérées comme des valeurs manquantes. Syntaxe : NODE.whitespace_missing.FIELDNAME

Tableau 40. Propriétés communes aux noeuds source (suite).

Nom de la propriété	Le type de données	Description de la propriété
description	*chaîne*	Permet d'indiquer un libellé de champ ou une description.
default_include	*indicateur*	Propriété saisie permettant d'indiquer si le comportement par défaut consiste à transmettre ou à filtrer les champs : NODE.default_include Exemple : set mynode:filternode.default_include = false
include	*indicateur*	Propriété saisie utilisée pour déterminer pour chaque champ s'il doit être inclus ou filtré : NODE.include.FIELDNAME.
new_name	*chaîne*	
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Cette propriété saisie est similaire à type dans la mesure où elle peut être utilisée pour définir la mesure associée au champ. La différence est que dans un script Python, l'une des valeurs de MeasureType peut également être transmise à la fonction de méthode d'accès set (setter) alors que la méthode d'accès get (getter) est toujours renvoyée sur les valeurs de MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Pour les champs de résumé (listes avec une profondeur de 0), cette propriété saisie définit le type de mesure associé au valeurs sous-jacentes.
geo_type	Point MultiPoint Chaîne MultiChaînes Polygone MultiPolygone	Pour les champs géospatiaux, cette propriété saisie définit le type d'objet géospatial représenté par ce champ. Elle doit être cohérente avec la profondeur de liste des valeurs.
has_coordinate_system	*booléen*	Pour les champs géospatiaux, cette propriété spécifie si le champ comporte un système de coordonnées
coordinate_system	*chaîne*	Pour les champs géospatiaux, cette propriété saisie définit le système de coordonnées pour ce champ.

Tableau 40. Propriétés communes aux noeuds source (suite).

Nom de la propriété	Le type de données	Description de la propriété
custom_storage_type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	Cette propriété saisie est similaire à custom_storage dans la mesure où elle peut être utilisée pour définir le stockage de substitution pour le champ. La différence est que dans un script Python, l'une des valeurs de StorageType peut également être transmise à la fonction de méthode d'accès set (setter) alors que la méthode d'accès get (getter) est toujours renvoyée sur les valeurs de StorageType.
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Pour les champs de liste, cette propriété saisie spécifie le type de stockage des valeurs sous-jacentes.
custom_list_depth	*entier*	Pour les champs de liste, cette propriété saisie spécifie la profondeur du champ.

Propriétés de asimport

La source Analytic Server vous permet d'exécuter un flux sur le système de fichiers HDFS (Hadoop Distributed File System).

Exemple

```
node = stream.create("asimport", "My node")
node.setPropertyValue("data_source", "Drug1n")
```

Tableau 41. propriétés de asimport.

Propriétés de asimport	Type de données	Description de la propriété
data_source	*chaîne*	Nom de la source de données.

Propriétés du noeud cognosimport

Le noeud source IBM Cognos BI importe des données depuis les bases de données Cognos BI.

Exemple

```
node = stream.create("cognosimport", "My node")
node.setPropertyValue("cognos_connection", ["http://mycogsrv1:9300/p2pd/servlet/dispatch",
 True, "", "", ""])
```

```
node.setPropertyValue("cognos_package_name", "/Public Folders/GOSALES")
node.setPropertyValue("cognos_items", ["[GreatOutdoors].[BRANCH].[BRANCH_CODE]",
"[GreatOutdoors]
.[BRANCH].[COUNTRY_CODE]"])
```

Tableau 42. Propriétés du noeud cognosimport.

Propriétés du noeud cognosimport	**Le type de données**	**Description de la propriété**
mode	Data Rapport	Spécifie s'il faut importer les données Cognos BI (par défaut) ou les rapports.
cognos_connection	*{"chaîne",indicateur,"chaîne", "chaîne" ,"chaîne"}*	Une propriété de liste contenant les détails de la connexion du serveur Cognos. Le format est : {URL_serveur_Cognos", mode_connexion, "espace de nom", "nom utilisateur", "mot de passe"} où : URL_serveur_Cognos est l'URL du serveur Cognos contenant la source. mode_connexion indique si la connexion anonyme est utilisée et est soit true soit false ; si la valeur est true, les champs suivants doivent être définis sur "". espace de nom spécifie le fournisseur de sécurité pour l'authentification utilisé pour se connecter au serveur. nom utilisateur et mot de passe sont ceux utilisés pour la connexion au serveur Cognos. Au lieu de mode_connexion, les modes suivants sont également disponibles : • anonymousMode. Par exemple : {'URL_serveur_Cognos', 'anonymousMode', "espace de nom", "nom utilisateur", "mot de passe"} • credentialMode. Par exemple : {'URL_serveur_Cognos', 'credentialMode', "espace de nom", "nom utilisateur", "mot de passe"} • storedCredentialMode. Par exemple : {'URL_serveur_Cognos', 'storedCredentialMode', "nom_données_identification_stockées"} Où nom_données_identification_stockées est le nom de données d'identification stockées Cognos dans le référentiel.
cognos_package_name	*chaîne*	Le chemin d'accès et le nom du package Cognos depuis lequel vous importez les objets de donnés, par exemple : /Public Folders/GOSALES Remarque : seules les barres obliques sont valides.

Tableau 42. Propriétés du noeud cognosimport (suite).

Propriétés du noeud **cognosimport**	Le type de données	Description de la propriété
cognos_items	*{"champ","champ", ... ,"field"}*	Le nom d'un ou plusieurs objets de données à importer. Le format de *champ* est [*namespace*].[*query_subject*].[*query_item*]
cognos_filters	*zone*	Le nom d'un ou plusieurs filtres à appliquer avant d'importer les données.
cognos_data_parameters	*liste (list)*	Valeurs des paramètres d'invite des données. Les paires nom/valeur sont entre accolades et les paires multiples sont séparées par des virgules et la chaîne entière est entre crochets. Format : [{"*param1*", "*value*"},...,{"*paramN*", "*value*"}]
cognos_report_directory	*zone*	Le chemin Cognos d'un dossier ou d'un package depuis lequel importer les rapports, par exemple : /Public Folders/GOSALES Remarque : seules les barres obliques sont valides.
cognos_report_name	*zone*	Le chemin et le nom dans l'emplacement d'un rapport à importer.
cognos_report_parameters	*liste (list)*	Valeurs des paramètres de rapport. Les paires nom/valeur sont entre accolades et les paires multiples sont séparées par des virgules et la chaîne entière est entre crochets. Format : [{"*param1*", "*value*"},...,{"*paramN*", "*value*"}]

Propriétés de databasenode

Le nœud SGBD peut être utilisé pour importer des données provenant de nombreux autres logiciels utilisant la connectivité ODBC (Open Database Connectivity), tels que Microsoft SQL Server, DB2, Oracle, etc.

Exemple

```
import modeler.api
stream = modeler.script.stream()
nnode = stream.create("database", "My node")
node.setPropertyValue("mode", "Table")
node.setPropertyValue("query", "SELECT * FROM drug1n")
node.setPropertyValue("datasource", "Drug1n_db")
node.setPropertyValue("username", "spss")
node.setPropertyValue("password", "spss")
node.setPropertyValue("tablename", ".Drug1n")
```

Tableau 43. propriétés de databasenode.

Propriétés de databasenode	Le type de données	Description de la propriété
mode	Table Query	Spécifiez *Table* pour établir la connexion à une table de base de données à l'aide des commandes de la boîte de dialogue ou indiquez *Requête* pour interroger la base de données sélectionnée en utilisant SQL.
datasource	*chaîne*	Nom de la base de données (reportez-vous à la remarque ci-dessous).
username	*chaîne*	Détails de la connexion à la base de données (reportez-vous aussi à la remarque ci-dessous).
password	*chaîne*	
credential	*chaîne*	Nom des données d'identification stockées dans IBM SPSS Collaboration and Deployment Services. Peut être utilisé au lieu des propriétés username et password. Le nom d'utilisateur et le mot de passe des données d'identification doivent correspondre à ceux qui sont requis pour accéder à la base de données.
use_credential		Défini sur True ou False.
epassword	*chaîne*	Indique un mot de passe codé à la place du codage en dur d'un mot de passe dans un script. Pour plus d'informations, voir la rubrique «Génération d'un mot de passe codé», à la page 51. Cette propriété est en lecture seule au cours de l'exécution.
tablename	*chaîne*	Nom de la table à laquelle vous souhaitez accéder.
strip_spaces	None Left Right Both	Options permettant de supprimer des espaces situés en début et en fin de chaîne.
use_quotes	AsNeeded Always Never	Indiquez si les noms des tables et des colonnes sont placés entre guillemets lors de l'envoi des requêtes à la base de données (par exemple, s'ils contiennent des espaces ou des signes de ponctuation).
query	*chaîne*	Indique le code SQL de la requête à soumettre.

Remarque : Si le nom de base de données (dans la propriété datasource) contient un ou plusieurs espaces, points ou traits de soulignement, vous pouvez utiliser le format "barre oblique inversée guillemet" pour le traiter en tant que chaîne. Par exemple : `\"db2v9.7.6_linux\"` ou : `"\"TDATA 131\""`.

Remarque : Si le nom de la base de données (dans la propriété de datasource) comporte des espaces, au lieu des propriétés individuelles de datasource, username et password, vous pouvez également utiliser une propriété de datasource unique au format suivant :

Tableau 44. Propriétés de databasenode - datasource spécifique.

Propriétés de databasenode	Le type de données	Description de la propriété
datasource	*chaîne*	Format : {database_name,username,password[,true | false]} Le dernier paramètre sert aux mots de passe codés. S'il est défini sur true, le mot de passe est décodé avant usage.

Utilisez aussi ce format si vous modifiez la source de données ; cependant, si vous souhaitez simplement changer le nom d'utilisateur ou le mot de passe, vous pouvez utiliser les propriétés username ou password.

Propriétés de datacollectionimportnode

Le noeud d'importation des données IBM SPSS Data Collection importe des données d'enquête en fonction du modèle IBM SPSS Data Collection Data Model utilisé par les produits IBM Corp. dédiés aux études de marché. Pour pouvoir utiliser ce nœud, vous devez avoir installé avant la bibliothèque de données IBM SPSS Data Collection.

Figure 7. Nœud d'importation des données Dimensions

Exemple

```
node = stream.create("datacollectionimport", "My node")
node.setPropertyValue("metadata_name", "mrQvDsc")
node.setPropertyValue("metadata_file", "C:/Program Files/IBM/SPSS/DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("casedata_name", "mrQvDsc")
node.setPropertyValue("casedata_source_type", "File")
node.setPropertyValue("casedata_file", "C:/Program Files/IBM/SPSS/DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("import_system_variables", "Common")
node.setPropertyValue("import_multi_response", "MultipleFlags")
```

Tableau 45. propriétés de datacollectionimportnode.

Propriétés de `datacollectionimportnode`	**Le type de données**	**Description de la propriété**

`metadata_name`	*chaîne*	Nom du MDSC. La valeur spéciale `DimensionsMDD` indique que le document de métadonnées IBM SPSS Data Collection standard doit être utilisé. Autres valeurs possibles :
		`mrADODsc`
		`mrI2dDsc`
		`mrLogDsc`
		`mrQdiDrsDsc`
		`mrQvDsc`
		`mrSampleReportingMDSC`
		`mrSavDsc`
		`mrSCDsc`
		`mrScriptMDSC`
		
		La valeur spéciale `none` (aucun) indique qu'il n'existe aucun MDSC.

`metadata_file`	*chaîne*	Nom du fichier de stockage des métadonnées.

`casedata_name`	*chaîne*	Nom du CDSC. Valeurs possibles :
		`mrADODsc`
		`mrI2dDsc`
		`mrLogDsc`
		`mrPunchDSC`
		`mrQdiDrsDsc`
		`mrQvDsc`
		`mrRdbDsc2`
		`mrSavDsc`
		`mrScDSC`
		`mrXmlDsc`
		
		La valeur spéciale `none` (aucun) indique qu'il n'existe aucun CDSC.

`casedata_source_type`	`Inconnu`
	`File`
	`Folder`
	`UDL`
	`DSN`
		Indique le type de source du CDSC.

`casedata_file`	*chaîne*	Quand la propriété `casedata_source_type` est paramétrée sur *Fichier*, cette propriété indique le fichier contenant les données d'observation.
`casedata_folder`	*chaîne*	Quand la propriété `casedata_source_type` est paramétrée sur *Dossier*, cette propriété indique le dossier contenant les données d'observation.
`casedata_udl_string`	*chaîne*	Quand la propriété `casedata_source_type` est paramétrée sur *UDL*, cette propriété indique la chaîne de connexion OLE-DB correspondant à la source de données contenant les données d'observation.
`casedata_dsn_string`	*chaîne*	Quand la propriété `casedata_source_type` est paramétrée sur *DSN*, cette propriété indique la chaîne de connexion ODBC correspondant à la source de données.

Tableau 45. propriétés de datacollectionimportnode (suite).

Propriétés de `datacollectionimportnode`	**Le type de données**	**Description de la propriété**
`casedata_project`	*chaîne*	Lorsque vous lisez des données d'observation provenant d'une base de données IBM SPSS Data Collection, vous pouvez fournir le nom du projet. Pour tous les autres types de données d'observation, ce paramètre doit rester vide.

`version_import_mode`	`All`
	`Latest`
	`Specify`
		Définit le mode de traitement des versions.

`specific_version`	*chaîne*	Quand la propriété `version_import_mode` est paramétrée sur *Spécifier*, cette propriété définit la version des données d'observation à importer.
`use_language`	*chaîne*	Définit si les libellés d'une langue spécifique doivent être utilisées.
`language`	*chaîne*	Si la propriété `use_language` est paramétrée sur True (vrai), cette propriété définit le code langue à utiliser pour l'importation. Il doit s'agir de l'un des codes disponibles dans les données d'observation.
`use_context`	*chaîne*	Définit si un contexte spécifique doit être importé. Les contextes sont utilisés pour varier la description associée aux réponses.
`context`	*chaîne*	Si la propriété `use_context` est paramétrée sur True (vrai), cette propriété définit le contexte à importer. Il doit s'agir de l'un des contextes disponibles dans les données d'observation.
`use_label_type`	*chaîne*	Définit si un type de libellé spécifique doit être importé.
`label_type`	*chaîne*	Si la propriété `use_label_type` est paramétrée sur True (vrai), cette propriété définit le type de libellé à importer. Il doit s'agir de l'un des types de libellé disponibles dans les données d'observation.
`user_id`	*chaîne*	Pour les bases de données exigeant une connexion explicite, indiquez l'ID et le mot de passe utilisateur nécessaires pour accéder à la source de données.
`password`	*chaîne*	

`import_system_variables`	`Commun`
	`None`
	`All`
		Spécifie les variables système qui sont importées.

`import_codes_variables`	*indicateur*	
`import_sourcefile_variables`	*indicateur*	
`import_multi_response`	`MultipleFlags` `Single`	

Propriétés de excelimportnode

Le nœud Import Excel permet d'importer des données issues de n'importe quelle version de Microsoft Excel. Aucune source de données ODBC n'est requise.

Exemples

```
#Pour utiliser un intervalle nommé :
node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xls")
node.setPropertyValue("use_named_range", True)
node.setPropertyValue("named_range", "DRUG")
node.setPropertyValue("read_field_names", True)

#Pour utiliser un intervalle explicite :
node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xls")
node.setPropertyValue("worksheet_mode", "Name")
node.setPropertyValue("worksheet_name", "Drug")
node.setPropertyValue("explicit_range_start", "A1")
node.setPropertyValue("explicit_range_end", "F300")
```

Tableau 46. propriétés de excelimportnode.

Propriétés de excelimportnode	Le type de données	Description de la propriété
excel_file_type	Excel2003 Excel2007	
full_filename	*chaîne*	Nom du fichier complet, y compris son chemin d'accès.
use_named_range	*Booléen*	Si vous voulez utiliser un intervalle nommé. Si cette propriété est définie sur True (vrai), la propriété named_range est utilisée pour indiquer l'intervalle à lire. Les autres paramètres de feuille de calcul et d'intervalle de données sont ignorés.
named_range	*chaîne*	
worksheet_mode	Index Name	Indique si la feuille de calcul est définie par un index ou par un nom.
worksheet_index	*entier*	Index de la feuille de calcul à lire : 0 pour la première, 1 pour la deuxième, etc.
worksheet_name	*chaîne*	Nom de la feuille de calcul à lire.
data_range_mode	FirstNonBlank ExplicitRange	Indique le mode de détermination de l'intervalle.
blank_rows	StopReading ReturnBlankRows	Quand la propriété data_range_mode est paramétrée sur *PremièreLigneRenseignée*, cette propriété indique comment traiter les lignes non renseignées.
explicit_range_start	*chaîne*	Quand la propriété data_range_mode est paramétrée sur *IntervalleExplicite*, cette propriété indique le point de départ de l'intervalle à lire.

Tableau 46. propriétés de excelimportnode (suite).

Propriétés de excelimportnode	Le type de données	Description de la propriété
explicit_range_end	*chaîne*	
noms_champ_lecture	*Booléen*	Indique si la première ligne de l'intervalle spécifié doit être utilisée pour les noms de champ (de colonne).

Propriétés de evimportnode

Le nœud Enterprise View crée une connexion à un IBM SPSS Collaboration and Deployment Services Repository, vous permettant de lire des données Enterprise View dans un flux et de regrouper un modèle dans un scénario accessible depuis le référentiel par d'autres utilisateurs.
Remarque : Le noeud Enterprise View a été remplacé dans SPSS Modeler 16.0 par le noeud Vue de données. Pour les flux sauvegardés dans des éditions précédentes, le noeud Enterprise View reste pris en charge. Toutefois, lors de la mise à jour ou de la création de flux, il est recommandé d'utiliser le noeud Vue de données.

Exemple

```
node = stream.create("evimport", "My node")
node.setPropertyValue("connection", ["Training data","/Application views/Marketing","LATEST",
"Analytic", "/Data Providers/Marketing"])
node.setPropertyValue("tablename", "cust1")
```

Tableau 47. propriétés de evimportnode.

Propriétés de evimportnode	Le type de données	Description de la propriété
connection	*liste*	Propriété structurée : liste de paramètres constituant une connexion Enterprise View. Syntaxe : `evimportnode.connection = [description,app_view_path, app_view_version_label, environment,DPD_path]`
tablename	*chaîne*	Nom d'un tableau dans la Vue d'application.

Propriétés de fixedfilenode

Le nœud Fixe permet d'importer les données de fichiers texte de longueur fixe, c'est-à-dire les fichiers dont les champs ne sont pas délimités, mais commencent au même endroit et sont de longueur fixe. Les données générées automatiquement ou héritées sont souvent stockées au format de longueur fixe.

Exemple

```
node = stream.create("fixedfile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("record_len", 32)
node.setPropertyValue("skip_header", 1)
```

```
node.setPropertyValue("fields", [["Age", 1, 3], ["Sex", 5, 7], ["BP", 9, 10], ["Cholesterol",
 12, 22], ["Na", 24, 25], ["K", 27, 27], ["Drug", 29, 32]])
node.setPropertyValue("decimal_symbol", "Period")
node.setPropertyValue("lines_to_scan", 30)
```

Tableau 48. Propriétés de fixedfilenode.

Propriétés de **fixedfilenode**	Le type de données	Description de la propriété
record_len	*nombre*	Indique le nombre de caractères dans chaque enregistrement.
line_oriented	*indicateur*	Ignore le caractère de nouvelle ligne figurant à la fin de chaque enregistrement.
decimal_symbol	Default Comma Period	Type de séparateur décimal utilisé dans votre source de données.
skip_header	*nombre*	Indique le nombre de lignes à ignorer au début du premier enregistrement. Utile pour ignorer les en-têtes de colonne.
auto_recognize_datetime	*indicateur*	Spécifie si les dates ou les heures sont automatiquement identifiées dans les données source.
lines_to_scan	*nombre*	
fields	*liste*	Propriétés structurées.
full_filename	*chaîne*	Nom complet (répertoire compris) du fichier à lire.
strip_spaces	None Left Right Both	Les espaces situés en début et en fin de chaîne sont supprimés lors de l'importation.
invalid_char_mode	Discard Replace	Supprime les caractères non valides (null, 0 ou tout caractère inexistant dans le codage en cours) de l'entrée de données ou remplace les caractères non valides par le symbole représentant un caractère.
invalid_char_replacement	*chaîne*	
use_custom_values	*indicateur*	
custom_storage	Inconnu String Integer Real Time Date Timestamp	

Tableau 48. Propriétés de fixedfilenode (suite).

Propriétés de **fixedfilenode**	Le type de données	Description de la propriété
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "JJ-MM-AA" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	Cette propriété est applicable uniquement si un stockage personnalisé est indiqué.
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Cette propriété est applicable uniquement si un stockage personnalisé est indiqué.
custom_decimal_symbol	*champ*	Applicable uniquement si un stockage personnalisé est indiqué.
encoding	StreamDefault SystemDefault "UTF-8"	Indique la méthode de codage de texte.

Propriétés du noeud gsdata_import

Utilisez le noeud source Géospatial pour ajouter des données de carte ou spatiales dans votre session d'exploration de données.

Tableau 49. Propriétés du noeud gsdata_import

Propriétés du noeud gsdata_import	Type de données	Description de la propriété
full_filename	*chaîne*	Entrez le chemin d'accès au fichier .shp à charger.
map_service_URL	*chaîne*	Entrez l'URL du service de carte auquel se connecter.
map_name	*chaîne*	Uniquement si map_service_URL est utilisé. Contient la structure des dossiers de niveau supérieur du service de carte.

Propriétés de sasimportnode

Le nœud SAS permet d'importer des données SAS dans IBM SPSS Modeler.

Exemple

```
node = stream.create("sasimport", "My node")
node.setPropertyValue("format", "Windows")
node.setPropertyValue("full_filename", "C:/data/retail.sas7bdat")
node.setPropertyValue("member_name", "Test")
node.setPropertyValue("read_formats", False)
node.setPropertyValue("full_format_filename", "Test")
node.setPropertyValue("import_names", True)
```

Tableau 50. propriétés de sasimportnode.

Propriétés de sasimportnode	Le type de données	Description de la propriété
format	Windows UNIX Transport SAS7 SAS8 SAS9	Format du fichier à importer.
full_filename	*chaîne*	Nom de fichier complet que vous entrez, y compris son chemin d'accès.
member_name	*chaîne*	Indique le membre à importer du fichier Transport SAS spécifié.
read_formats	*indicateur*	Lit les formats de données (tels que les libellés de variable) du fichier de format spécifié.
full_format_filename	*chaîne*	
import_names	NamesAndLabels LabelsasNames	Spécifie la méthode de mappage des noms de variable et des libellés lors de l'importation.

Propriétés de simgennode

Le noeud Génération de simulation permet de générer facilement des données simulées, à partir de zéro en utilisant des distributions statistiques spécifiées par l'utilisateur ou automatiquement en utilisant les distributions issues de l'exécution d'un noeud Ajustement de simulation sur des données historiques existantes. Cela peut être utile si vous souhaitez évaluer le résultat d'un modèle prédictif en cas d'incertitude dans les entrées du modèle.

Tableau 51. Propriétés de simgennode.

Propriétés de simgennode	Type de données	Description de la propriété
fields	Propriété structurée	Voir l'exemple
correlations	Propriété structurée	Voir l'exemple
max_cases	*entier*	La valeur minimum est de 1000 ; la valeur maximum est de 2 147 483 647.
create_iteration_field	*booléen*	
iteration_field_name	*chaîne*	
replicate_results	*booléen*	
random_seed	*entier*	
overwrite_when_refitting	*booléen*	
parameter_xml	*chaîne*	Renvoie le paramètre Xml sous forme de chaîne.
distribution	Bernoulli Bêta Binomial Catégorielle Exponential Fixe Gamma Lognormal NegativeBinomialFailures NegativeBinomialTrials Normal Poisson Plage Triangulaire Uniforme Weibull	distribution est une déclaration du nom de distribution suivie par une liste contenant des paires de noms d'attribut et de valeurs. Exemple simgennode.setKeyedPropertyValue ("distribution", "Age", "Gamma") **Remarque :** Vous ne pouvez pas définir directement la distribution ; vous l'utilisez conjointement avec la propriété fields. Voir l'exemple pour fields qui suit ce tableau.
bernoulli_prob	*nombre*	0 ≤ bernoulli_prob ≤ 1
beta_shape1	*nombre*	Doit être ≥ 0
beta_shape2	*nombre*	Doit être ≥ 0
beta_min	*nombre*	Facultatif. Doit être inférieur à beta_max.
beta_max	*nombre*	Facultatif. Doit être supérieur à beta_min.
binomial_n	*entier*	Doit être > 0
binomial_prob	*nombre*	0 ≤ binomial_prob ≤ 1
binomial_min	*nombre*	Facultatif. Doit être inférieur à binomial_max.
binomial_max	*nombre*	Facultatif. Doit être supérieur à binomial_min.
exponential_scale	*nombre*	Doit être > 0

Tableau 51. Propriétés de simgennode (suite).

Propriétés de simgennode	Type de données	Description de la propriété
exponential_min	*nombre*	Facultatif. Doit être inférieur à exponential_max.
exponential_max	*nombre*	Facultatif. Doit être supérieur à exponential_min.
fixed_value	*chaîne*	
gamma_shape	*nombre*	Doit être ≥ 0
gamma_scale	*nombre*	Doit être ≥ 0
gamma_min	*nombre*	Facultatif. Doit être inférieur à gamma_max.
gamma_max	*nombre*	Facultatif. Doit être supérieur à gamma_min.
lognormal_shape1	*nombre*	Doit être ≥ 0
lognormal_shape2	*nombre*	Doit être ≥ 0
lognormal_min	*nombre*	Facultatif. Doit être inférieur à lognormal_max.
lognormal_max	*nombre*	Facultatif. Doit être supérieur à lognormal_min.
negative_bin_failures_threshold	*nombre*	Doit être ≥ 0
negative_bin_failures_prob	*nombre*	0 ≤ negative_bin_failures_prob ≤ 1
negative_bin_failures_min	*nombre*	Facultatif. Doit être inférieur à negative_bin_failures_max.
negative_bin_failures_max	*nombre*	Facultatif. Doit être supérieur à negative_bin_failures_min.
negative_bin_trials_threshold	*nombre*	Doit être ≥ 0
negative_bin_trials_prob	*nombre*	0 ≤ negative_bin_trials_prob ≤ 1
negative_bin_trials_min	*nombre*	Facultatif. Doit être inférieur à negative_bin_trials_max.
negative_bin_trials_max	*nombre*	Facultatif. Doit être inférieur à negative_bin_trials_min.
normal_mean	*nombre*	
normal_sd	*nombre*	Doit être > 0
normal_min	*nombre*	Facultatif. Doit être inférieur à normal_max.
normal_max	*nombre*	Facultatif. Doit être supérieur à normal_min.
poisson_mean	*nombre*	Doit être ≥ 0
poisson_min	*nombre*	Facultatif. Doit être inférieur à poisson_max.
poisson_max	*nombre*	Facultatif. Doit être supérieur à poisson_min.
triangular_mode	*nombre*	triangular_min ≤ triangular_mode ≤ triangular_max
triangular_min	*nombre*	Doit être inférieur à triangular_mode.
triangular_max	*nombre*	Doit être supérieur à triangular_mode.
uniform_min	*nombre*	Doit être inférieur à uniform_max.

Tableau 51. Propriétés de simgennode (suite).

Propriétés de simgennode	Type de données	Description de la propriété
uniform_max	*nombre*	Doit être supérieur à uniform_min.
weibull_rate	*nombre*	Doit être ≥ 0
weibull_scale	*nombre*	Doit être ≥ 0
weibull_location	*nombre*	Doit être ≥ 0
weibull_min	*nombre*	Facultatif. Doit être inférieur à weibull_max.
weibull_max	*nombre*	Facultatif. Doit être supérieur à weibull_min.

Exemple pour fields

Il s'agit d'un paramètre de propriété structurée dont la syntaxe est :

```
simgennode.setPropertyValue("fields", [
    [champ1, storage, locked, [distribution1], min, max],
    [champ2, storage, locked, [distribution2], min, max],
    [champ3, storage, locked, [distribution3], min, max]
])
```

Chaque distribution est définie de la manière suivante :

```
[distributionname, [{par1}, {par2}, {par3}]]
```

```
simgennode = modeler.script.stream().createAt("simgen", u"Sim Gen", 726, 322)
simgennode.setPropertyValue("fields", [["Age", "nteger", False, ["Uniform"], 15, 74]])
```

Par exemple, pour créer un noeud qui génère un champ unique avec une distribution binomiale, vous pouvez utiliser le script suivant :

```
simgen_node1 = modeler.script.stream().createAt("simgen", u"Sim Gen", 200, 200)
simgen_node1.setPropertyValue("fields", [["Education", "Real", False, ["Binomial", [["n", 32],
 ["prob", 0.7]]], "", ""]])
```

La distribution binomiale (Binomial) utilise deux paramètres : n et prob. Comme Binomial ne prend pas en charge les valeurs de minimum et de maximum, ces valeurs sont fournies en tant que chaîne vide.

Exemple pour correlations

Il s'agit d'un paramètre de propriété structurée dont la syntaxe est :

```
simgennode.setPropertyValue("correlations", [
    [champ1, champ2, correlation],
    [champ1, champ3, correlation],
    [champ2, champ3, correlation]
])
```

La corrélation peut correspondre à n'importe quel nombre compris entre +1 et -1. Vous pouvez indiquer autant ou aussi peu de corrélations que vous le souhaitez. Toutes les corrélations non spécifiées sont définies sur zéro. Si des champs sont inconnus, la valeur de corrélation doit être définie sur la matrice (ou table) de corrélation et apparaît sous forme de texte rouge. Lorsque des champs sont inconnus, il est impossible d'exécuter le noeud.

Propriétés de statisticsimportnode

Le noeud Fichier IBM SPSS Statistics lit les données du format de fichier *.sav* utilisé par IBM SPSS Statistics ainsi que des fichiers cache enregistrés dans IBM SPSS Modeler, qui utilisent le même format.

Les propriétés de ce nœud sont décrites dans «Propriétés de statisticsimportnode», à la page 299.

Propriétés du noeud tm1import

Le noeud source IBM Cognos TM1 importe des données depuis les bases de données Cognos TM1.

Tableau 52. Propriétés du noeud tm1import.

Propriétés du noeud tm1import	Type de données	Description de la propriété
pm_host	*chaîne*	Nom d'hôte. Par exemple : `TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
tm1_connection	*{"champ","champ", ... ,"champ"}*	Une propriété de liste contenant les détails de la connexion du serveur TM1. Le format est le suivant : `[ "nom_serveur_TM1", "nom_utilisateur_tm1", "mot_de_passe_tm1"]` Par exemple : `TM1_import.setPropertyValue("tm1_connection", ['Planning Sample', "admin", "apple"])`
selected_view	*{"champ" "champ"}*	Propriété de liste contenant les détails du cube TM1 sélectionné et le nom de la vue de cube depuis laquelle les données seront importées dans SPSS. Par exemple : `TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])`

Propriétés de userinputnode

Le noeud Utilisateur représente une façon simple de créer des données synthétiques (à partir de zéro ou en modifiant des données existantes). Ceci est utile, par exemple, si vous souhaitez créer un jeu de données de test pour la modélisation.

Exemple

```
node = stream.create("userinput", "My node")
node.setPropertyValue("names", ["test1", "test2"])
node.setKeyedPropertyValue("data", "test1", "2, 4, 8")
node.setKeyedPropertyValue("custom_storage", "test1", "Integer")
node.setPropertyValue("data_mode", "Ordered")
```

Tableau 53. propriétés de userinputnode.

Propriétés de userinputnode	Le type de données	Description de la propriété
data		
names		Propriété structurée définissant ou renvoyant une liste de noms de champ générés par le nœud.
custom_storage	Inconnu String Integer Real Time Date Timestamp	Propriété saisie définissant ou renvoyant le stockage d'un champ.
data_mode	Combiné Ordered	Si la propriété est définie sur Combined, les enregistrements sont générés pour chaque combinaison de valeurs définies et de valeurs min/max. Le nombre d'enregistrements générés est égal au produit du nombre de valeurs dans chaque champ. Si la propriété est définie sur Ordered, une valeur est copiée à partir des colonnes de chaque enregistrement pour générer une ligne de données. Le nombre d'enregistrements générés est égal au plus grand nombre de valeurs associées à un champ. Tous les champs comportant moins de valeurs de données seront remplis à l'aide de valeurs nulles.
values		**Remarque :** Cette propriété a été remplacée par userinputnode.data et ne doit plus être utilisée.

Propriétés de variablefilenode

Le noeud Délimité lit les données de fichiers texte de longueur variable, c'est-à-dire les fichiers dont les enregistrements contiennent un nombre fixe de champs et un nombre variable de caractères. Ce nœud est également utile pour les fichiers contenant des textes d'en-tête de longueur fixe et certains types d'annotation.

Exemple

```
node = stream.create("variablefile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("read_field_names", True)
node.setPropertyValue("delimit_other", True)
node.setPropertyValue("other", ",")
node.setPropertyValue("quotes_1", "Discard")
node.setPropertyValue("decimal_symbol", "Comma")
node.setPropertyValue("invalid_char_mode", "Replace")
node.setPropertyValue("invalid_char_replacement", "|")
node.setKeyedPropertyValue("use_custom_values", "Age", True)
node.setKeyedPropertyValue("direction", "Age", "Input")
node.setKeyedPropertyValue("type", "Age", "Range")
node.setKeyedPropertyValue("values", "Age", [1, 100])
```

Tableau 54. propriétés de variablefilenode.

Propriétés de variablefilenode	Le type de données	Description de la propriété
skip_header	*nombre*	Indique le nombre de caractères à ignorer au début du premier enregistrement.
num_fields_auto	*indicateur*	Détermine automatiquement le nombre de champs dans chaque enregistrement. Les enregistrements doivent être terminés par un caractère de retour à la ligne.
num_fields	*nombre*	Indiquez manuellement le nombre de champs dans chaque enregistrement.
delimit_space	*indicateur*	Indique le caractère utilisé pour délimiter les champs dans le fichier.
delimit_tab	*indicateur*	
delimit_new_line	*indicateur*	
delimit_non_printing	*indicateur*	
delimit_comma	*indicateur*	Si la virgule sert à la fois de délimiteur de champs et de séparateur décimal pour les flux, paramétrez delimit_other sur *true* (vrai), puis indiquez la virgule comme délimiteur à l'aide de la propriété other.
delimit_other	*indicateur*	Vous permet d'indiquer un délimiteur personnalisé à l'aide de la propriété other.
other	*chaîne*	Indique le délimiteur utilisé quand la propriété delimit_other est paramétrée sur *true* (vrai).
decimal_symbol	Default Comma Period	Indique le séparateur décimal utilisé dans la source de données.
multi_blank	*indicateur*	Considère plusieurs délimiteurs non renseignés non adjacents comme un seul délimiteur.
noms_champ_lecture	*indicateur*	Considère la première ligne du fichier de données comme des libellés de colonne.
strip_spaces	None Left Right Both	Les espaces situés en début et en fin de chaîne sont supprimés lors de l'importation.
invalid_char_mode	Discard Replace	Supprime les caractères non valides (null, 0 ou tout caractère inexistant dans le codage en cours) de l'entrée de données ou remplace les caractères non valides par le symbole représentant un caractère.
invalid_char_replacement	*chaîne*	
break_case_by_newline	*flag*	Indique que le délimiteur de ligne est le caractère de retour à la ligne.
lines_to_scan	*nombre*	Indique le nombre de lignes à analyser pour les types de données spécifiés.
auto_recognize_datetime	*indicateur*	Spécifie si les dates ou les heures sont automatiquement identifiées dans les données source.

Tableau 54. propriétés de variablefilenode (suite).

Propriétés de variablefilenode	Le type de données	Description de la propriété
quotes_1	Discard PairAndDiscard IncludeAsText	Indique la manière dont les guillemets simples sont traités lors de l'importation.
quotes_2	Discard PairAndDiscard IncludeAsText	Indique la manière dont les guillemets doubles sont traités lors de l'importation.
full_filename	*chaîne*	Nom complet (répertoire compris) du fichier à lire.
use_custom_values	*indicateur*	
custom_storage	Inconnu String Integer Real Time Date Timestamp	
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "JJ-MM-AA" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	Applicable uniquement si un stockage personnalisé est indiqué.

Tableau 54. propriétés de variablefilenode (suite).

Propriétés de variablefilenode	Le type de données	Description de la propriété
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Applicable uniquement si un stockage personnalisé est indiqué.
custom_decimal_symbol	*champ*	Applicable uniquement si un stockage personnalisé est indiqué.
encoding	StreamDefault SystemDefault "UTF-8"	Indique la méthode de codage de texte.

Propriétés de xmlimportnode

Le noeud source XML importe des données au format XML dans le flux. Vous pouvez importer un fichier ou tous les fichiers dans un répertoire. Vous pouvez aussi spécifier un fichier de schéma à partir duquel lire la structure XML.

Exemple

```
node = stream.create("xmlimport", "My node")
node.setPropertyValue("full_filename", "c:/import/ebooks.xml")
node.setPropertyValue("records", "/author/name")
```

Tableau 55. propriétés de xmlimportnode.

Propriétés de xmlimportnode	Le type de données	Description de la propriété
read	single directory	Lit un seul fichier de données (par défaut), ou tous les fichiers XML d'un répertoire.
recurse	*indicateur*	Spécifie s'il faut lire des fichiers XML supplémentaires dans tous les sous-répertoires du répertoire spécifié.
full_filename	*chaîne*	(requis) Chemin complet et nom de fichier du fichier XML à importer (si read = single).
directory_name	*chaîne*	(requis) Chemin complet et nom du répertoire à partir duquel importer les fichiers XML (si read = directory).
full_schema_filename	*chaîne*	Chemin complet et nom du fichier XSD ou DTD à partir duquel la structure XML est lue. Si vous omettez ce paramètre, la structure est lue à partir du fichier source XML.

Tableau 55. propriétés de xmlimportnode (suite).

Propriétés de xmlimportnode	Le type de données	Description de la propriété
records	*chaîne*	Expression XPath (par exemple. /auteur/nom) pour définir la limite de l'enregistrement. À chaque fois que cet élément est rencontré dans le fichier source, un nouvel enregistrement est créé.
mode	read specify	Lire toutes les données (par défaut), ou spécifier les éléments à lire.
fields		Liste des éléments (éléments et attributs) à importer. Chaque élément de la liste est une expression XPath.

Propriétés de dataviewimport

Le noeud Vue de données permet d'importer des données de vue de données dans IBM SPSS Modeler.

Exemple

```
stream = modeler.script.stream()

dvnode = stream.createAt("dataviewimport", "Data View", 96, 96)
dvnode.setPropertyValue("analytic_data_source",
["","/folder/adv", "LATEST"])
dvnode.setPropertyValue("table_name", ["","com.ibm.spss.Table"])
dvnode.setPropertyValue("data_access_plan",
["","DataAccessPlan"])
dvnode.setPropertyValue("optional_attributes",
[["","NewDerivedAttribute"]])
dvnode.setPropertyValue("include_xml", True)
dvnode.setPropertyValue("include_xml_field", "xml_data")
```

Tableau 56. Propriétés de dataviewimport

Propriétés de dataviewimport	Type de données	Description de la propriété
analytic_data_source	*chaîne*	Objet vue de données analytiques stocké dans IBM SPSS Collaboration and Deployment Services. Nom de chemin et libellé de version pour la version à utiliser. ["Object ID","Full path", "Version"]
table_name	*chaîne*	Table de vue de données utilisée dans la vue de données analytiques. Le nom de table doit être qualifié par le package. Vous pouvez obtenir le package en exportant le modèle objet métier (BOM) depuis le client Deployment Manager d'IBM SPSS Collaboration and Deployment Services et en recherchant le fichier default.bom dans l'archive ZIP exportée. Le nom de package doit toujours être le même sauf si le modèle objet métier a été importé depuis IBM Operational Decision Management (iLOG). ["Object ID", "Name"]

Tableau 56. Propriétés de dataviewimport (suite)

Propriétés de **dataviewimport**	Type de données	Description de la propriété
`data_access_plan`	*chaîne*	Plan d'accès aux données utilisé pour fournir les données pour la vue de données analytiques. `["Object ID","Name"]`
`optional_attributes`	*chaîne*	Liste d'attributs dérivés à inclure. `[["ID1","Name1"], ["ID2", "Name2"]]`
`include_xml`	*booléen*	`True` si un champ avec des données d'instance de modèle d'objet d'exécution (XOM) doit être inclus. Sauf si des noeuds IBM Analytical Decision Management iLOG sont utilisés, la valeur recommandée est `false`. L'activation de ce paramètre peut ajouter un traitement supplémentaire important.
`include_xml_field`	*chaîne*	Le nom du champ à ajouter lorsque `include_xml` est défini sur `true`.

Chapitre 10. Propriétés des noeuds d'opérations sur les lignes

Propriétés de appendnode

Le noeud Ajouter réalise la concaténation d'ensembles d'enregistrements. Il permet de combiner des jeux de données dont les structures sont similaires, mais les données différentes.

Exemple

```
node = stream.create("append", "My node")
node.setPropertyValue("match_by", "Name")
node.setPropertyValue("match_case", True)
node.setPropertyValue("include_fields_from", "All")
node.setPropertyValue("create_tag_field", True)
node.setPropertyValue("tag_field_name", "Append_Flag")
```

Tableau 57. propriétés de appendnode.

Propriétés de appendnode	Le type de données	Description de la propriété
match_by	Position Name	Vous pouvez ajouter des jeux de données sur la base de la position des champs dans la source de données principale, ou du nom des champs dans les jeux de données d'entrée.
match_case	*indicateur*	Active la distinction des majuscules/minuscules lors de la mise en correspondance des noms de champ.
include_fields_from	Principal All	
create_tag_field	*indicateur*	
tag_field_name	*chaîne*	

Propriétés de aggregatenode

Le noeud Agréger remplace une séquence d'enregistrements d'entrée par des enregistrements de sortie abrégés et agrégés.

Exemple

```
node = stream.create("aggregate", "My node")
# dbnode is a configured database import node
stream.link(dbnode, node)
node.setPropertyValue("contiguous", True)
node.setPropertyValue("keys", ["Drug"])
node.setKeyedPropertyValue("aggregates", "Age", ["Sum", "Mean"])
```

```
node.setPropertyValue("inc_record_count", True)
node.setPropertyValue("count_field", "index")
node.setPropertyValue("extension", "Aggregated_")
node.setPropertyValue("add_as", "Prefix")
```

Tableau 58. propriétés de aggregatenode.

Propriétés de aggregatenode	Le type de données	Description de la propriété
keys	*liste (list)*	Affiche les champs qui peuvent être utilisés comme clés pour l'agrégation. Par exemple, si vous avez choisi les champs-clés Sexe et Région, chaque combinaison unique de M et F avec les régions N et S (quatre combinaisons uniques) est associée à un enregistrement agrégé.
contiguous	*indicateur*	Sélectionnez cette option si vous savez que tous les enregistrements présentant les mêmes valeurs clés sont regroupés dans l'entrée (par exemple, si l'entrée est triée dans les champs clés). Ainsi, vous améliorez les performances.
aggregates		Propriété structurée affichant les champs numériques dont les valeurs seront agrégées, ainsi que les modes d'agrégation sélectionnés.
aggregate_exprs		Propriété saisie qui entre le nom de champ calculé avec l'expression agrégée utilisée pour le calcul. Par exemple : `aggregatenode.setKeyedPropertyValue ("aggregate_exprs", "Na_MAX", "MAX('Na')")`
extension	*chaîne*	Indique un préfixe ou un suffixe pour les champs agrégés dupliqués (exemple ci-dessous).
add_as	Suffix Prefix	
inc_record_count	*indicateur*	Crée un champ supplémentaire qui indique le nombre d'enregistrements d'entrée agrégés pour constituer chaque enregistrement agrégé.
count_field	*chaîne*	Indique le nom du champ de nombre d'enregistrements.
allow_approximation	*Booléen*	Autorise une approximation des statistiques d'ordre lorsqu'une agrégation est exécutée dans Analytic Server
bin_count	*entier*	Spécifie le nombre de casiers à utiliser dans l'approximation

Propriétés de balancenode

Le noeud Equilibrer corrige les déséquilibres survenant dans un jeu de données, de manière à respecter une condition précise. La règle d'équilibrage ajuste la proportion d'enregistrements présentant une condition True (vrai) par rapport au facteur indiqué.

Exemple

```
node = stream.create("balance", "My node")
node.setPropertyValue("training_data_only", True)
node.setPropertyValue("directives", [[1.3, "Age > 60"], [1.5, "Na > 0.5"]])
```

Tableau 59. Propriétés de balancenode.

Propriétés de balancenode	Le type de données	Description de la propriété
directives		Propriété structurée permettant d'équilibrer la proportion de valeurs de champ en fonction du nombre spécifié (reportez-vous à l'exemple ci-dessous).
training_data_only	*indicateur*	Spécifie que seules des données d'apprentissage devraient être équilibrées. Si aucun champ de partitionnement n'est présent dans le flux, cette option n'est pas prise en compte.

Cette propriété de noeud utilise le format suivant :

[[*nombre, chaîne*] \ [*nombre, chaîne*] \ ... [*nombre, chaîne*]].

Remarque : Si des chaînes (utilisant des guillemets doubles) sont intégrées à l'expression, elles doivent être précédées du caractère d'échappement " \ ". Le caractère " \ " est également le caractère de continuation de ligne, qui vous permet d'aligner les arguments pour plus de clarté.

Propriétés de derive_stbnode

Le noeud Cases-Espace-Heure dérive la valeur Cases-Espace-Heure des champs de latitude, de longitude et d'horodatage. Vous pouvez également identifier les valeurs Cases-Espace-Heure fréquentes comme des arrêts ou des ralentissements.

Exemple

```
node = modeler.script.stream().createAt("derive_stb", "My node", 96, 96)

# Mode enregistrements individuels
node.setPropertyValue("mode", "IndividualRecords")
node.setPropertyValue("latitude_field", "Latitude")
node.setPropertyValue("longitude_field", "Longitude")
node.setPropertyValue("timestamp_field", "OccurredAt")
node.setPropertyValue("densities", ["STB_GH7_1HOUR", "STB_GH7_30MINS"])
node.setPropertyValue("add_extension_as", "Prefix")
node.setPropertyValue("name_extension", "stb_")

# Mode des blocages
node.setPropertyValue("mode", "Hangouts")
node.setPropertyValue("hangout_density", "STB_GH7_30MINS")
node.setPropertyValue("id_field", "Event")
node.setPropertyValue("qualifying_duration", "30MINUTES")
node.setPropertyValue("min_events", 4)
node.setPropertyValue("qualifying_pct", 65)
```

Tableau 60. Propriétés du noeud Space-Time-Boxes

Propriétés de derive_stbnode	Type de données	Description de la propriété
mode	Enregistrements individuels : blocages	
latitude_field	*zone*	
longitude_field	*zone*	
timestamp_field	*zone*	

Tableau 60. Propriétés du noeud Space-Time-Boxes (suite)

Propriétés de derive_stbnode	Type de données	Description de la propriété
hangout_density	*densité*	Densité unique. Pour obtenir les valeurs de densité valides, voir densities.
densities	[*densité*,*densité*,..., *densité*]	Chaque densité est une chaîne ; par exemple STB_GH8_1DAY. **Remarque :** Il existe des limites de validité des densités. Pour geohash, les valeurs doivent être comprises entre GH1 et GH15. Pour la partie temporelle, les valeurs suivantes peuvent être utilisées : JAMAIS 1AN 1MOIS 1JOUR 12HEURES 8HEURES 6HEURES 4HEURES 3HEURES 2HEURES 1HEURE 30MINS 15MINS 10MINS 5MINS 2MINS 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SEC
id_field	*zone*	
qualifying_duration	1JOUR 12HEURES 8HEURES 6HEURES 4HEURES 3HEURES 2HEURES 1HEURE 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	Doit être une chaîne.
min_events	*entier*	La valeur entière minimale valide est 2.
qualifying_pct	*entier*	Doit être compris entre 1 et 100.
add_extension_as	Préfixe Suffixe	
name_extension	*chaîne*	

Propriétés de distinctnode

Le noeud Distinguer supprime les enregistrements en double, soit en incluant le premier enregistrement dans le flux de données, soit en le supprimant et en incluant ses doublons dans le flux de données.

Exemple

```
node = stream.create("distinct", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("fields", ["Age" "Sex"])
node.setPropertyValue("keys_pre_sorted", True)
```

Tableau 61. propriétés de distinctnode.

Propriétés de distinctnode	Le type de données	Description de la propriété
mode	Include Discard	Vous pouvez enlever le premier enregistrement distinct du flux de données, ou l'isoler et transmettre les éventuels enregistrements dupliqués au flux de données.
grouping_fields	*liste (list)*	Affiche les champs utilisés pour détecter les enregistrements identiques. **Remarque :** cette propriété est obsolète à partir de IBM SPSS Modeler 16.
composite_value	Propriété structurée	Reportez-vous à l'exemple ci-dessous.
composite_values	Propriété structurée	Reportez-vous à l'exemple ci-dessous.
inc_record_count	*flag*	Crée un champ supplémentaire qui indique le nombre d'enregistrements d'entrée agrégés pour constituer chaque enregistrement agrégé.
count_field	*chaîne*	Indique le nom du champ de nombre d'enregistrements.
sort_keys	Propriété structurée.	**Remarque :** cette propriété est obsolète à partir de IBM SPSS Modeler 16.
default_ascending	*flag*	
low_distinct_key_count	*indicateur*	Spécifie que vous avez seulement un petit nombre d'enregistrements et/ou un petit nombre de valeurs uniques du/des champ(s)-clé(s).
keys_pre_sorted	*indicateur*	Spécifie que tous les enregistrements avec les mêmes valeurs-clés sont regroupés dans l'entrée.
disable_sql_generation	*indicateur*	

Exemple de propriété composite_value

La propriété composite_value a le format général suivant :

```
node.setKeyedPropertyValue("composite_value", FIELD, FILLOPTION)
```

FILLOPTION a le format [FillType, Option1, Option2, ...].

Exemples :

```
node.setKeyedPropertyValue("composite_value", "Age", ["First"])
node.setKeyedPropertyValue("composite_value", "Age", ["last"])
node.setKeyedPropertyValue("composite_value", "Age", ["Total"])
node.setKeyedPropertyValue("composite_value", "Age", ["Average"])
node.setKeyedPropertyValue("composite_value", "Age", ["Min"])
node.setKeyedPropertyValue("composite_value", "Age", ["Max"])
node.setKeyedPropertyValue("composite_value", "Date", ["Earliest"])
node.setKeyedPropertyValue("composite_value", "Date", ["Latest"])
node.setKeyedPropertyValue("composite_value", "Code", ["FirstAlpha"])
node.setKeyedPropertyValue("composite_value", "Code", ["LastAlpha"])
```

Les options personnalisées ont besoin de plusieurs arguments qui sont ajoutés sous forme de liste, par exemple :

```
node.setKeyedPropertyValue("composite_value", "Name", ["MostFrequent", "FirstRecord"])
node.setKeyedPropertyValue("composite_value", "Date", ["LeastFrequent", "LastRecord"])
node.setKeyedPropertyValue("composite_value", "Pending", ["IncludesValue", "T", "F"])
node.setKeyedPropertyValue("composite_value", "Marital", ["FirstMatch", "Married", "Divorced", "Separated"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Space"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Comma"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "UnderScore"])
```

Exemple de propriété `composite_values`

La propriété `composite_values` a le format général suivant :

```
node.setPropertyValue("composite_values", [
                        [FIELD1, [FILLOPTION1]],
                        [FIELD2, [FILLOPTION2]],
.
.
])
```

Exemple :

```
node.setPropertyValue("composite_values", [
    ["Age", ["First"]],
    ["Name", ["MostFrequent", "First"]],
    ["Pending", ["IncludesValue", "T"]],
    ["Marital", ["FirstMatch", "Married", "Divorced", "Separated"]],
    ["Code", ["Concatenate", "Comma"]]
])
```

Propriétés de mergenode

Le noeud Fusionner permet de créer, à partir de plusieurs enregistrements d'entrée, un seul enregistrement de sortie contenant tout ou partie des champs d'entrée. Il sert notamment à fusionner des données provenant de différentes sources, telles que les données client internes et les données démographiques acquises.

Exemple

```
node = stream.create("merge", "My node")
# assume customerdata and salesdata are configured database import nodes
stream.link(customerdata, node)
stream.link(salesdata, node)
node.setPropertyValue("method", "Keys")
node.setPropertyValue("key_fields", ["id"])
node.setPropertyValue("common_keys", True)
node.setPropertyValue("join", "PartialOuter")
node.setKeyedPropertyValue("outer_join_tag", "2", True)
node.setKeyedPropertyValue("outer_join_tag", "4", True)
node.setPropertyValue("single_large_input", True)
```

```
node.setPropertyValue("single_large_input_tag", "2")
node.setPropertyValue("use_existing_sort_keys", True)
node.setPropertyValue("existing_sort_keys", [["id", "Ascending"]])
```

Tableau 62. propriétés de mergenode.

Propriétés de mergenode	Le type de données	Description de la propriété
method	Order Keys condition Rankedcondition	Indique si les enregistrements font l'objet d'une fusion dans l'ordre dans lequel ils sont répertoriés dans les fichiers de données, si des champs-clés sont utilisés pour fusionner les enregistrements comportant des valeurs identiques dans ces champs-clés, si des enregistrements feront l'objet d'une fusion si une condition particulière est satisfaite ou si chaque ligne appariée dans le jeu de données principal et tous les jeux de données secondaires doit être fusionnée, en utilisant l'expression de classement pour trier les correspondances multiples par ordre croissant.
condition	*chaîne*	Si method est défini sur Condition, spécifie la condition d'inclusion ou d'exclusion des enregistrements.
key_fields	*liste (list)*	
common_keys	*indicateur*	
join	Interne FullOuter PartialOuter Anti	
outer_join_tag.n	*indicateur*	Dans cette propriété, *n* est le nom de la balise tel qu'il apparaît dans la boîte de dialogue de sélection du jeu de données. Remarque : vous pouvez indiquer plusieurs noms de balise, puisque n'importe quel nombre de jeux de données peut générer des enregistrements incomplets.
single_large_input	*indicateur*	Indique si une fonction d'optimisation est utilisée lorsqu'une entrée est plus volumineuse que les autres.
single_large_input_tag	*chaîne*	Indique le nom de la balise tel qu'il apparaît dans la boîte de dialogue de sélection du jeu de données volumineux. L'utilisation de cette propriété diffère légèrement de celle de la propriété outer_join_tag (indicateur/chaîne) car un seul jeu de données d'entrée peut être spécifié.
use_existing_sort_keys	*indicateur*	Indique si les entrées sont déjà triées en fonction d'un ou de plusieurs champs-clés.
existing_sort_keys	[{*'string'*, 'Ascending'} \ {*'string''*, 'Descending'}]	Indique les champs déjà triés ainsi que le sens du tri.
primary_dataset	*chaîne*	Si method est Rankedcondition, sélectionnez le jeu de données principal dans la fusion. Ce jeu de données peut être considéré comme la partie de gauche d'une fusion de jointure externe.

Tableau 62. propriétés de mergenode (suite).

Propriétés de mergenode	Le type de données	Description de la propriété
add_tag_duplicate	*Booléen*	Si method est Rankedcondition et si Y est défini, si le jeu de données fusionné résultant contient plusieurs champs avec le même nom issus de sources de données différentes, les balises respectives provenant des sources de données sont ajoutées au début des en-têtes de colonne de champ.
merge_condition	*chaîne*	
ranking_expression	*chaîne*	
Num_matches	*entier*	Nombre de correspondances à renvoyer en fonction de merge_condition et ranking_expression. Minimum 1, maximum 100.

Propriétés de rfmaggregatenode

Le noeud agrégé Recency, Frequency, Monetary (RFM) vous permet de prendre les données de l'historique des transactions d'un client, d'en éliminer les éventuelles données inutilisées et de combiner le reste des données de transaction sur une seule ligne qui indique la date de la dernière consultation, le nombre de transactions réalisées et la valeur monétaire totale de ces transactions.

Exemple

```
node = stream.create("rfmaggregate", "My node")
node.setPropertyValue("relative_to", "Fixed")
node.setPropertyValue("reference_date", "2007-10-12")
node.setPropertyValue("id_field", "CardID")
node.setPropertyValue("date_field", "Date")
node.setPropertyValue("value_field", "Amount")
node.setPropertyValue("only_recent_transactions", True)
node.setPropertyValue("transaction_date_after", "2000-10-01")
```

Tableau 63. propriétés de rfmaggregatenode.

propriétés de rfmaggregatenode	Le type de données	Description de la propriété
relative_to	Fixed Today	Spécifie la date à partir de laquelle la récence des transactions sera calculée.
reference_date	*date*	Disponible uniquement si Fixed est choisi dans relative_to.
contiguous	*indicateur*	Si vos données sont pré-triées de façon à ce que tous les enregistrements avec le même ID apparaissent ensemble dans le flux de données, la sélection de cette option accélère le traitement.
id_field	*champ*	Spécifie le champ à utiliser pour identifier le client et ses transactions.
date_field	*champ*	Spécifie le champ de date à utiliser pour calculer la récence.
value_field	*champ*	Spécifie le champ à utiliser pour calculer la valeur monétaire.
extension	*chaîne*	Indique un préfixe ou un suffixe pour les champs agrégés en double.

Tableau 63. propriétés de rfmaggregatenode (suite).

propriétés de rfmaggregatenode	Le type de données	Description de la propriété
add_as	Suffix Prefix	Spécifie si l'extension doit être ajoutée comme suffixe ou comme préfixe.
discard_low_value_records	*indicateur*	Active l'utilisation du paramètre discard_records_below.
discard_records_below	*nombre*	Spécifie une valeur minimale sous laquelle les éventuels détails de la transaction ne seront pas utilisés pour calculer les totaux RFM. Les unités de valeur font référence au champ de value sélectionné.
only_recent_transactions	*indicateur*	Active l'utilisation des paramètres specify_transaction_date ou transaction_within_last.
specify_transaction_date	*indicateur*	
transaction_date_after	*date*	Disponible uniquement si specify_transaction_date est sélectionné. Spécifie la date de transaction après laquelle les enregistrements seront inclus dans votre analyse.
transaction_within_last	*nombre*	Disponible uniquement si transaction_within_last est sélectionné. Spécifie le nombre et le type de périodes (jours, semaines, mois ou années) provenant de la valeur de calcul de récence par rapport à la date après laquelle les enregistrements seront inclus dans votre analyse.
transaction_scale	Days Weeks Mois Years	Disponible uniquement si transaction_within_last est sélectionné. Spécifie le nombre et le type de périodes (jours, semaines, mois ou années) provenant de la valeur de calcul de récence par rapport à la date après laquelle les enregistrements seront inclus dans votre analyse.
save_r2	*indicateur*	Affiche la date de l'avant-dernière transaction pour chaque client.
save_r3	*indicateur*	Disponible uniquement si save_r2 est sélectionné. Affiche la date de la troisième transaction la plus récente pour chaque client.

Propriétés de Rprocessnode

Le noeud de processus R vous permet d'utiliser les données d'un flux IBM(r) SPSS(r) Modeler et de modifier ces données à l'aide de votre propre script R personnalisé. Une fois que les données sont modifiées, elles sont renvoyées au flux.

Exemple

```
node = stream.create("rprocess", "My node")
node.setPropertyValue("custom_name", "my_node")
node.setPropertyValue("syntax", """day<-as.Date(modelerData$dob, format="%Y-%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
```

```
var1<-c(fieldName="Next day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
node.setPropertyValue("convert_datetime", "POSIXct")
```

Tableau 64. Propriétés de Rprocessnode.

Propriétés de Rprocessnode	Type de données	Description de la propriété
syntax	*chaîne*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*flag*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*flag*	

Propriétés de samplenode

Le noeud Echantillonner sélectionne un sous-ensemble d'enregistrements. Divers types d'échantillons sont pris en charge, notamment les échantillons stratifiés, en cluster et non aléatoires (structurés). L'échantillonnage peut être utile pour améliorer les performances et pour sélectionner des groupes d'enregistrements associés ou des transactions pour analyse.

Exemple

```
/* Crée deux noeuds Echantillonner pour extraire
   différents échantillons des mêmes données */

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Simple")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("sample_type", "First")
node.setPropertyValue("first_n", 500)

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Complex")
node.setPropertyValue("stratify_by", ["Sex", "Cholesterol"])
node.setPropertyValue("sample_units", "Proportions")
node.setPropertyValue("sample_size_proportions", "Custom")
node.setPropertyValue("sizes_proportions", [["M", "High", "Default"],
["M", "Normal", "Default"], ["F", "High", 0.3], ["F", "Normal", 0.3]])
```

Tableau 65. propriétés de samplenode.

Propriétés de samplenode	Le type de données	Description de la propriété
method	Simple Complexe	
mode	Include Discard	Enlève ou isole les enregistrements qui correspondent à la condition indiquée.
sample_type	First OneInN RandomPct	Indique la méthode d'échantillonnage.
first_n	*entier*	Les enregistrements jusqu'au point de césure indiqué sont inclus ou exclus.

Tableau 65. propriétés de samplenode (suite).

Propriétés de samplenode	Le type de données	Description de la propriété
one_in_n	*nombre*	Inclut ou exclut un enregistrement tous les *n* enregistrements.
rand_pct	*nombre*	Indique le pourcentage d'enregistrements à inclure ou à exclure.
use_max_size	*indicateur*	Active l'utilisation du paramètre maximum_size.
maximum_size	*entier*	Indiquez la taille maximale des échantillons à inclure dans le flux de données ou à exclure du flux. Cette option est redondante et par conséquent désactivée si les options First et Include sont sélectionnées.
set_random_seed	*indicateur*	Active l'utilisation du paramètre de valeur de départ aléatoire.
random_seed	*entier*	Indique la valeur utilisée en tant que valeur de départ aléatoire.
complex_sample_type	Random Systematic	
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*nombre*	
fixed_counts	*entier*	
variable_proportions	*champ*	
variable_counts	*champ*	
use_min_stratum_size	*indicateur*	
minimum_stratum_size	*entier*	Cette option s'applique uniquement lorsqu'un échantillon Complexe est utilisé avec Sample units=Proportions.
use_max_stratum_size	*indicateur*	
maximum_stratum_size	*entier*	Cette option s'applique uniquement lorsqu'un échantillon Complexe est utilisé avec Sample units=Proportions.
clusters	*champ*	
stratify_by	*[champ1 ... champN]*	
specify_input_weight	*indicateur*	
input_weight	*champ*	
new_output_weight	*chaîne*	
sizes_proportions	[{string *valeur de chaîne*}{string *valeur de chaîne*}...]	Si sample_units=proportions et sample_size_proportions=Custom, spécifie une valeur pour chaque combinaison possible de valeurs pour les champs de stratification.
default_proportion	*nombre*	

Tableau 65. propriétés de samplenode (suite).

Propriétés de samplenode	Le type de données	Description de la propriété
sizes_counts	[{string *valeur de chaîne*}{string *valeur de chaîne*}...]	Spécifie une valeur pour chaque combinaison de valeurs possible pour les champs de stratification. L'utilisation est similaire à celle de sizes_proportions, mais en spécifiant un entier au lieu d'une proportion.
default_count	*nombre*	

Propriétés de selectnode

Le noeud Sélectionner permet de sélectionner ou d'exclure des sous-ensembles d'enregistrements d'un flux de données sur la base d'une condition spécifique. Par exemple, vous pouvez sélectionner les enregistrements qui appartiennent à un secteur de ventes particulier.

Exemple

```
node = stream.create("select", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("condition", "Age < 18")
```

Tableau 66. propriétés de selectnode.

Propriétés de selectnode	Le type de données	Description de la propriété
mode	Include Discard	Indique si les enregistrements sélectionnés sont à enlever ou à isoler.
condition	*chaîne*	Condition d'inclusion ou d'exclusion des enregistrements.

Propriétés de sortnode

Le noeud Trier trie les enregistrements par ordre croissant ou décroissant, en fonction de la valeur d'un ou de plusieurs champs.

Exemple

```
node = stream.create("sort", "My node")
node.setPropertyValue("keys", [["Age", "Ascending"], ["Sex", "Descending"]])
node.setPropertyValue("default_ascending", False)
node.setPropertyValue("use_existing_keys", True)
node.setPropertyValue("existing_keys", [["Age", "Ascending"]])
```

Tableau 67. Propriétés de sortnode.

Propriétés de sortnode	Le type de données	Description de la propriété
keys	*liste (list)*	Spécifie les champs servant de base au tri. Si aucune direction n'est indiquée, la valeur utilisée est celle par défaut.
default_ascending	*indicateur*	Indique l'ordre de tri par défaut.
use_existing_keys	*indicateur*	Indique si le tri est optimisé à l'aide de l'ordre de tri précédent des champs déjà triés.

Tableau 67. Propriétés de sortnode (suite).

Propriétés de sortnode	Le type de données	Description de la propriété
existing_keys		Indique les champs déjà triés ainsi que le sens du tri. Utilise le même format que la propriété keys.

Propriété de streamingts

Le noeud Flux TS (Streaming TS) crée et évalue des modèles de séries chronologiques en une seule étape, sans avoir besoin d'un noeud Intervalles de temps.

Exemple

```
node = stream.create("streamingts", "My node")
node.setPropertyValue("deployment_force_rebuild", True)
node.setPropertyValue("deployment_rebuild_mode", "Count")
node.setPropertyValue("deployment_rebuild_count", 3)
node.setPropertyValue("deployment_rebuild_pct", 11)
node.setPropertyValue("deployment_rebuild_field", "Year")
```

Tableau 68. Propriété de streamingts.

Propriété de streamingts	Type de données	Description de la propriété
custom_fields	*flag*	Si custom_fields=false, on utilise les paramètres provenant d'un noeud Type en amont. Si custom_fields=true, vous devez spécifier targets et inputs.
targets	[*champ1...champN*]	
inputs	[*champ1...champN*]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	*flag*	
conf_limit_pct	*real*	
use_time_intervals_node	*flag*	Si use_time_intervals_node=true, on utilise les paramètres provenant d'un noeud Intervalle de temps en amont. Si use_time_intervals_node=false, interval_offset_position, interval_offset et interval_type doivent être spécifiés.
interval_offset_position	LastObservation LastRecord	LastObservation fait référence à **dernière observation valide**. LastRecord fait référence à **Compter depuis le dernier enregistrement**.
interval_offset	*nombre*	
interval_type	Periods Years Quarters Months WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
événements	*champs*	

Tableau 68. Propriété de streamingts (suite).

Propriété de **streamingts**	Type de données	Description de la propriété
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*flag*	
detect_outliers	*flag*	
expert_outlier_additive	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_innovational	*flag*	
expert_outlier_transient	*flag*	
expert_outlier_seasonal_additive	*flag*	
expert_outlier_local_trend	*flag*	
expert_outlier_additive_patch	*flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles
arima_d	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles
arima_q	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles
arima_sp	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles
arima_sd	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles
arima_sq	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles
arima_transformation_type	None SquareRoot NaturalLog	Même propriété que pour le noeud de modélisation des séries temporelles
arima_include_constant	*flag*	Même propriété que pour le noeud de modélisation des séries temporelles
tf_arima_p.*nomchamp*	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles. Pour les fonctions de transfert.
tf_arima_d.*nomchamp*	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles. Pour les fonctions de transfert.
tf_arima_q.*nomchamp*	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles. Pour les fonctions de transfert.

Tableau 68. Propriété de streamingts (suite).

Propriété de **streamingts**	Type de données	Description de la propriété
tf_arima_sp.*nomchamp*	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles. Pour les fonctions de transfert.
tf_arima_sd.*nomchamp*	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles. Pour les fonctions de transfert.
tf_arima_sq.*nomchamp*	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles. Pour les fonctions de transfert.
tf_arima_delay.*nomchamp*	*entier*	Même propriété que pour le noeud de modélisation des séries temporelles. Pour les fonctions de transfert.
tf_arima_transformation_type.*fieldname*	None SquareRoot NaturalLog	
arima_detect_outlier_mode	None Automatic	
arima_outlier_additive	*flag*	
arima_outlier_level_shift	*flag*	
arima_outlier_innovational	*flag*	
arima_outlier_transient	*flag*	
arima_outlier_seasonal_additive	*flag*	
arima_outlier_local_trend	*flag*	
arima_outlier_additive_patch	*flag*	
deployment_force_rebuild	*flag*	
deployment_rebuild_mode	Count Percent	
deployment_rebuild_count	*nombre*	
deployment_rebuild_pct	*nombre*	
deployment_rebuild_field	*<zone>*	

Chapitre 11. Propriétés des nœuds d'opérations sur les champs

Propriétés de anonymizenode

Le noeud Anonymiser transforme la façon dont les noms et les valeurs des champs sont représentés en aval, masquant ainsi les données d'origine. Cela peut s'avérer utile si vous souhaitez permettre à d'autres utilisateurs de générer des modèles utilisant des données confidentielles, par exemple des noms de clients ou autre.

Exemple

```
stream = modeler.script.stream()
varfilenode = stream.createAt("variablefile", "File", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
node = stream.createAt("anonymize", "My node", 192, 96)
# Anonymize node requires the input fields while setting the values
stream.link(varfilenode, node)
node.setKeyedPropertyValue("enable_anonymize", "Age", True)
node.setKeyedPropertyValue("transformation", "Age", "Random")
node.setKeyedPropertyValue("set_random_seed", "Age", True)
node.setKeyedPropertyValue("random_seed", "Age", 123)
node.setKeyedPropertyValue("enable_anonymize", "Drug", True)
node.setKeyedPropertyValue("use_prefix", "Drug", True)
node.setKeyedPropertyValue("prefix", "Drug", "myprefix")
```

Tableau 69. propriétés de anonymizenode

Propriétés de anonymizenode	**Le type de données**	**Description de la propriété**
enable_anonymize	*flag*	Lorsque cette propriété est définie sur True, elle active l'anonymisation des valeurs de champ (revient à sélectionner **Oui** pour ce champ dans la colonne Anonymiser des valeurs).
use_prefix	*flag*	Lorsque cette propriété est définie sur True, un préfixe personnalisé sera utilisé, s'il en existe un. S'applique aux champs qui seront anonymisés grâce à la méthode de hachage et revient à sélectionner la case d'option **Personnalisé** dans la boîte de dialogue Remplacer les valeurs pour ce champ.
prefix	*chaîne*	Revient à taper un préfixe dans la zone de texte de la boîte de dialogue Remplacer les valeurs. Le préfixe par défaut est la valeur par défaut si rien d'autre n'a été indiqué.
transformation	Random Fixed	Détermine si les paramètres de transformation pour un champ anonymisé par la méthode de transformation seront aléatoires ou fixes.
set_random_seed	*flag*	Lorsque cette propriété est définie sur True, la valeur de départ indiquée sera utilisée, si la propriété transformation est également définie sur Random (Aléatoire).
random_seed	*entier*	Lorsque set_random_seed est défini sur True, il s'agit de la valeur de départ pour le nombre aléatoire.
échelle	*nombre*	Lorsque la propriété transformation est définie sur Fixed (Fixe), cette valeur sert pour la mise à l'échelle ("scale by"). La valeur d'échelle maximale est normalement 10, mais elle peut être diminuée pour éviter tout dépassement.

Tableau 69. propriétés de anonymizenode (suite)

Propriétés de anonymizenode	Le type de données	Description de la propriété
translate	*nombre*	Lorsque la propriété transformation est définie sur Fixed (Fixe), cette valeur est utilisée pour "translate." La valeur maximale de conversion est normalement 1 000, mais elle peut être diminuée pour éviter tout dépassement.

Propriétés autodataprepnode

Le noeud de préparation automatisée de données (ADP) peut analyser vos données, identifier des corrections et filtrer des champs qui sont problématiques ou qui sont peu susceptibles d'être utiles. Il peut aussi créer de nouveaux attributs le cas échéant et améliorer la performance au moyen de techniques de filtrage et d'échantillonnage intelligentes. Vous pouvez utiliser le noeud de manière totalement automatisée, en laissant le noeud choisir et appliquer les corrections, ou vous pouvez prévisualiser les modifications avant qu'elles ne soient mises en place et les accepter, les rejeter ou les modifier selon les besoins.

Exemple

```
node = stream.create("autodataprep", "My node")
node.setPropertyValue("objective", "Balanced")
node.setPropertyValue("excluded_fields", "Filter")
node.setPropertyValue("prepare_dates_and_times", True)
node.setPropertyValue("compute_time_until_date", True)
node.setPropertyValue("reference_date", "Today")
node.setPropertyValue("units_for_date_durations", "Automatic")
```

Tableau 70. Propriétés autodataprepnode

Propriétés de autodataprepnode	Le type de données	Description de la propriété
objective	Balanced Vitesse Accuracy Personnalisé	
custom_fields	*flag*	Si la valeur est définie sur true (vrai), elle permet de spécifier les champs cible, entrée et d'autres champs pour le nœud actuel. Si elle est définie sur false (faux), les paramètres actuels provenant d'un noeud type en amont sont utilisés.
target	*zone*	Spécifie un champ cible unique.
inputs	[*champ1 ... champN*]	Champs d'entrée ou prédicteur utilisés par le modèle.
use_frequency	*flag*	
frequency_field	*zone*	
use_weight	*flag*	
weight_field	*zone*	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	

Tableau 70. Propriétés autodataprepnode (suite)

Propriétés de autodataprepnode	Le type de données	Description de la propriété
prepare_dates_and_times	*flag*	Contrôle l'accès à tous les champs date et heure
compute_time_until_date	*flag*	
reference_date	Today Fixed	
fixed_date	*date*	
units_for_date_durations	Automatique Fixed	
fixed_date_units	Years Mois Jours	
compute_time_until_time	*flag*	
reference_time	CurrentTime Fixed	
fixed_time	*heure*	
units_for_time_durations	Automatique Fixed	
fixed_date_units	Hours Minutes Secondes	
extract_year_from_date	*flag*	
extract_month_from_date	*flag*	
extract_day_from_date	*flag*	
extract_hour_from_time	*flag*	
extract_minute_from_time	*flag*	
extract_second_from_time	*flag*	
exclude_low_quality_inputs	*flag*	
exclude_too_many_missing	*flag*	
maximum_percentage_missing	*nombre*	
exclude_too_many_categories	*flag*	
maximum_number_categories	*nombre*	
exclude_if_large_category	*flag*	
maximum_percentage_category	*nombre*	
prepare_inputs_and_target	*flag*	
adjust_type_inputs	*flag*	
adjust_type_target	*flag*	
reorder_nominal_inputs	*flag*	
reorder_nominal_target	*flag*	
replace_outliers_inputs	*flag*	
replace_outliers_target	*flag*	
replace_missing_continuous_inputs	*flag*	
replace_missing_continuous_target	*flag*	
replace_missing_nominal_inputs	*flag*	

Tableau 70. Propriétés autodataprepnode (suite)

Propriétés de autodataprepnode	Le type de données	Description de la propriété
replace_missing_nominal_target	*flag*	
replace_missing_ordinal_inputs	*flag*	
replace_missing_ordinal_target	*flag*	
maximum_values_for_ordinal	*nombre*	
minimum_values_for_continuous	*nombre*	
outlier_cutoff_value	*nombre*	
outlier_method	Replace Suppr	
rescale_continuous_inputs	*flag*	
rescaling_method	MinMax ZScore	
min_max_minimum	*nombre*	
min_max_maximum	*nombre*	
z_score_final_mean	*nombre*	
z_score_final_sd	*nombre*	
rescale_continuous_target	*flag*	
target_final_mean	*nombre*	
target_final_sd	*nombre*	
transform_select_input_fields	*flag*	
maximize_association_with_target	*flag*	
p_value_for_merging	*nombre*	
merge_ordinal_features	*flag*	
merge_nominal_features	*flag*	
minimum_cases_in_category	*nombre*	
bin_continuous_fields	*flag*	
p_value_for_binning	*nombre*	
perform_feature_selection	*flag*	
p_value_for_selection	*nombre*	
perform_feature_construction	*flag*	
transformed_target_name_extension	*chaîne*	
transformed_inputs_name_extension	*chaîne*	
constructed_features_root_name	*chaîne*	
years_duration_ name_extension	*chaîne*	
months_duration_ name_extension	*chaîne*	
days_duration_ name_extension	*chaîne*	
hours_duration_ name_extension	*chaîne*	
minutes_duration_ name_extension	*chaîne*	
seconds_duration_ name_extension	*chaîne*	
year_cyclical_name_extension	*chaîne*	
month_cyclical_name_extension	*chaîne*	

Tableau 70. Propriétés autodataprepnode (suite)

Propriétés de autodataprepnode	Le type de données	Description de la propriété
day_cyclical_name_extension	*chaîne*	
hour_cyclical_name_extension	*chaîne*	
minute_cyclical_name_extension	*chaîne*	
second_cyclical_name_extension	*chaîne*	

Propriétés astimeintervalsnode

Le noeud Intervalles de temps d'origine n'est pas compatible avec Analytic Server (AS). Le noeud Intervalles de temps AS (nouveau dans SPSS Modeler édition 17.0) contient un sous-ensemble des fonctions du noeud Intervalles de temps existant qui peuvent être utilisées avec Analytic Server.

Utilisez le noeud Intervalles de temps AS pour spécifier des intervalles et calculer un nouveau champ Heure pour l'estimation ou la prévision. Un ensemble complet d'intervalles de temps, allant des secondes aux années, est pris en charge.

Tableau 71. Propriétés astimeintervalsnode

Propriétés astimeintervalsnode	Type de données	Description de la propriété
time_field	*zone*	Peut uniquement accepter un seul champ continu. Ce champ est utilisé pour le noeud comme clé d'agrégation pour la conversion de l'intervalle. Si un champ de type entier est utilisé ici, il est considéré comme un index de temps.
dimensions	*[champ1 champ2 ... champn]*	Ces champs sont utilisés pour créer des séries temporelles individuelles en fonction des valeurs des champs.
fields_to_aggregate	*[champ1 champ2 ... champn]*	Ces champs sont agrégés dans le cadre du changement de période du champ d'heure. Les champs qui ne sont pas inclus dans ce sélecteur sont filtrés et exclus des données qui quittent le noeud.

Propriétés de binningnode

Le noeud Discrétiser crée automatiquement des champs nominaux (ensemble) sur la base des valeurs d'un ou de plusieurs champs continus (intervalle numérique) existants. Par exemple, vous pouvez transformer un champ continu de revenus en un nouveau champ catégoriel contenant des groupes de revenus comme écarts par rapport à la moyenne. Une fois les intervalles du nouveau champ créés, vous pouvez générer un noeud Dériver à partir des points de césure.

Exemple

```
node = stream.create("binning", "My node")
node.setPropertyValue("fields", ["Na", "K"])
node.setPropertyValue("method", "Rank")
node.setPropertyValue("fixed_width_name_extension", "_binned")
node.setPropertyValue("fixed_width_add_as", "Suffix")
```

```
node.setPropertyValue("fixed_bin_method", "Count")
node.setPropertyValue("fixed_bin_count", 10)
node.setPropertyValue("fixed_bin_width", 3.5)
node.setPropertyValue("tile10", True)
```

Tableau 72. propriétés de binningnode

Propriétés de binningnode	Le type de données	Description de la propriété
fields	*[champ1 champ2 ... champN]*	Champs continus (intervalle numérique) en attente de transformation. Vous pouvez créer des intervalles pour plusieurs champs simultanément.
method	FixedWidth EqualCount Rank SDev Optimale	Méthode utilisée pour déterminer les points de césure des nouveaux intervalles de champ (catégories).
rcalculate_bins	Always IfNecessary	Indique si les intervalles sont recalculés et les données placées dans l'intervalle approprié à chaque exécution du nœud ou si les données sont uniquement ajoutées aux intervalles existants et aux nouveaux intervalles éventuellement ajoutés.
fixed_width_name_extension	*chaîne*	L'extension par défaut est *_BIN*.
fixed_width_add_as	Suffix Préfixe	Indique si l'extension est ajoutée à la fin (suffixe) ou au début (préfixe) du nom du champ. L'extension par défaut est *income_BIN*.
fixed_bin_method	Width Effectif	
fixed_bin_count	*entier*	Désigne l'entier déterminant le nombre d'intervalles de largeur fixe (catégories) des nouveaux champs.
fixed_bin_width	*real*	Valeur (entier ou réel) permettant de calculer la largeur de l'intervalle.
equal_count_name_ extension	*chaîne*	L'extension par défaut est *_TILE*.
equal_count_add_as	Suffix Préfixe	Indique une extension, suffixe ou préfixe, utilisée pour le nom de champ généré à l'aide de centiles standard. L'extension par défaut est *_TILE* plus *N*, *N* étant le numéro du quantile.
tile4	*flag*	Génère quatre intervalles de quintile, chacun contenant 25 % des observations.
tile5	*flag*	Génère cinq intervalles de quintiles.
tile10	*flag*	Génère 10 intervalles de déciles.
tile20	*flag*	Génère 20 intervalles de vingtiles.
tile100	*flag*	Génère 100 intervalles de centiles.
use_custom_tile	*flag*	
custom_tile_name_extension	*chaîne*	L'extension par défaut est *_TILEN*.
custom_tile_add_as	Suffix Préfixe	
custom_tile	*entier*	

Tableau 72. propriétés de binningnode (suite)

Propriétés de binningnode	Le type de données	Description de la propriété
equal_count_method	RecordCount ValueSum	La méthode RecordCount vise à affecter un nombre égal d'enregistrements à chaque intervalle, tandis que la méthode ValueSum affecte les enregistrements de sorte que la somme des valeurs de chaque intervalle soit égale.
tied_values_method	Suivant Current Aléatoire	Spécifie l'intervalle dans lequel les données de valeur ex æquo doivent être placées.
rank_order	Ascending Décroissant	Cette propriété peut avoir la valeur Ascending (la valeur la plus faible est marquée 1) ou Descending (la valeur la plus élevée est marquée 1).
rank_add_as	Suffix Préfixe	Cette option s'applique au rang, au rang fractionnaire et au rang de pourcentage.
rank	*flag*	
rank_name_extension	*chaîne*	L'extension par défaut est *_RANK*.
rank_fractional	*flag*	Permet de classer les observations dans lesquelles la valeur du nouveau champ équivaut au rang divisé par la somme des pondérations des observations non manquantes. Les rangs fractionnaires sont compris dans l'intervalle 0–1.
rank_fractional_name_ extension	*chaîne*	L'extension par défaut est *_F_RANK*.
rank_pct	*flag*	Chaque rang est divisé par le nombre d'enregistrements avec valeurs valides et multiplié par 100. Les rangs fractionnaires de pourcentage sont compris dans l'intervalle 1–100.
rank_pct_name_extension	*chaîne*	L'extension par défaut est *_P_RANK*.
sdev_name_extension	*chaîne*	
sdev_add_as	Suffix Préfixe	
sdev_count	One Two Trois	
optimal_name_extension	*chaîne*	L'extension par défaut est *_OPTIMAL*.
optimal_add_as	Suffix Préfixe	
optimal_supervisor_field	*zone*	Champ choisi comme champ de superviseur et auquel les champs sélectionnés pour la création d'intervalles sont associés.
optimal_merge_bins	*flag*	Indique que tous les intervalles présentant peu d'observations seront ajoutés à un intervalle voisin plus grand.
optimal_small_bin_threshold	*entier*	
optimal_pre_bin	*flag*	Indique que la pré-création d'intervalles du jeu de données va avoir lieu.

Tableau 72. propriétés de binningnode (suite)

Propriétés de binningnode	Le type de données	Description de la propriété
optimal_max_bins	*entier*	Définit une limite supérieure afin d'éviter de créer un nombre d'intervalles trop important.
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	
optimal_last_bin	Unbounded Bounded	

Propriétés de derivenode

Le noeud Dériver modifie les valeurs de données ou crée des nouveaux champs à partir d'un ou de plusieurs champs existants. Il crée des champs de type formule, indicateur, ensemble, nominal, statistiques, comptage et conditionnel.

Exemple 1

```
# Créer et configurer un noeud de champ indicateur Calculer
node = stream.create("derive", "My node")
node.setPropertyValue("new_name", "DrugX_Flag")
node.setPropertyValue("result_type", "Flag")
node.setPropertyValue("flag_true", "1")
node.setPropertyValue("flag_false", "0")
node.setPropertyValue("flag_expr", "'Drug' == \"drugX\"")

# Créer et configurer un noeud de champ Calculer conditionnel
node = stream.create("derive", "My node")
node.setPropertyValue("result_type", "Conditional")
node.setPropertyValue("cond_if_cond", "@OFFSET(\"Age\", 1) = \"Age\"")
node.setPropertyValue("cond_then_expr", "(@OFFSET(\"Age\", 1) = \"Age\" >< @INDEX")
node.setPropertyValue("cond_else_expr", "\"Age\"")
```

Exemple 2

Ce script suppose qu'il existe deux colonnes numériques appelées `XPos` et `YPos` qui représentent les coordonnées X et Y d'un point (par exemple, le moment où un événement a eu lieu). Le script crée un noeud Calculer qui calcule une colonne géospatiale depuis les coordonnées X et Y représentant ce point dans un système de coordonnées spécifique :

```
stream = modeler.script.stream()
# Other stream configuration code
node = stream.createAt("derive", "Location", 192, 96)
node.setPropertyValue("new_name", "Location")
node.setPropertyValue("formula_expr", "['XPos', 'YPos']")
node.setPropertyValue("formula_type", "Geospatial")
# Now we have set the general measurement type, define the
# specifics of the geospatial object
node.setPropertyValue("geo_type", "Point")
node.setPropertyValue("has_coordinate_system", True)
node.setPropertyValue("coordinate_system", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tableau 73. propriétés de derivenode

Propriétés de derivenode	Le type de données	Description de la propriété
new_name	*chaîne*	Nom du nouveau champ.
mode	Single Multiple	Spécifie un ou plusieurs champs.
fields	*liste (list)*	Utilisé en mode Multiple uniquement pour sélectionner plusieurs champs.
name_extension	*chaîne*	Indique l'extension des nouveaux noms de champ.
add_as	Suffix Préfixe	Ajoute l'extension du nom de champ en tant que préfixe (au début) ou en tant que suffixe (à la fin).
result_type	Formula Indicateur Set Etat Count Conditionnel	Les six types de nouveau champ que vous pouvez créer.
formule_expr	*chaîne*	Expression de calcul de la valeur du nouveau champ dans un noeud Dériver
flag_expr	*chaîne*	
flag_true	*chaîne*	
flag_false	*chaîne*	
set_default	*chaîne*	
set_value_cond	*chaîne*	Propriété structurée (définition de la condition associée à une valeur donnée).
state_on_val	*chaîne*	Indique la valeur du nouveau champ si la condition Activé est vérifiée.
state_off_val	*chaîne*	Indique la valeur du nouveau champ si la condition Désactivé est vérifiée.
state_on_expression	*chaîne*	
state_off_expression	*chaîne*	
state_initial	On Désactivé	Affecte à chaque enregistrement du nouveau champ la valeur initiale On ou Off. Cette valeur peut changer au fur et à mesure que les conditions sont respectées.
count_initial_val	*chaîne*	
count_inc_condition	*chaîne*	
count_inc_expression	*chaîne*	
count_reset_condition	*chaîne*	
cond_if_cond	*chaîne*	
cond_then_expr	*chaîne*	
cond_else_expr	*chaîne*	

Tableau 73. propriétés de derivenode (suite)

Propriétés de derivenode	Le type de données	Description de la propriété
formula_measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Cette propriété peut être utilisée pour définir la mesure associée au champ calculé. La fonction de méthode d'accès set (setter) peut être transmise sous la forme de chaîne ou de l'une des valeurs MeasureType. La fonction de méthode d'accès get (getter) renvoie toujours les valeurs MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Pour les champs de résumé (listes avec une profondeur de 0), cette propriété définit le type de mesure associé au valeurs sous-jacentes.
geo_type	Point MultiPoint Chaîne MultiChaînes Polygone MultiPolygone	Pour les champs géospatiaux, cette propriété définit le type d'objet géospatial représenté par ce champ. Elle doit être cohérente avec la profondeur de liste des valeurs.
has_coordinate_system	*booléen*	Pour les champs géospatiaux, cette propriété spécifie si le champ comporte un système de coordonnées
coordinate_system	*chaîne*	Pour les champs géospatiaux, cette propriété définit le système de coordonnées pour ce champ.

Propriétés de ensemblenode

Le noeud Ensemble combine deux ou plusieurs nuggets de modèles pour obtenir des prévisions plus précises que celles acquises à partir d'un modèle quelconque.

Exemple

```
# Create and configure an Ensemble node
# Use this node with the models in demos\streams\pm_binaryclassifier.str
node = stream.create("ensemble", "My node")
node.setPropertyValue("ensemble_target_field", "response")
node.setPropertyValue("filter_individual_model_output", False)
node.setPropertyValue("flag_ensemble_method", "ConfidenceWeightedVoting")
node.setPropertyValue("flag_voting_tie_selection", "HighestConfidence")
```

Tableau 74. propriétés de ensemblenode.

Propriétés de ensemblenode	Le type de données	Description de la propriété
ensemble_target_field	*champ*	Indique le champ cible pour tous les modèles utilisés dans l'ensemble.
filter_individual_model_output	*indicateur*	Indique si les résultats de scoring de modèles individuels doivent être supprimés.

Tableau 74. propriétés de ensemblenode (suite).

Propriétés de ensemblenode	Le type de données	Description de la propriété
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	Indique la méthode utilisée pour déterminer le score de l'ensemble. Ce paramètre s'applique uniquement si la cible sélectionnée est un champ indicateur.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Indique la méthode utilisée pour déterminer le score de l'ensemble. Ce paramètre s'applique uniquement si la cible sélectionnée est un champ nominal.
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	Si une méthode de vote est sélectionnée, indique la manière dont les ex æquo sont résolus. Ce paramètre s'applique uniquement si la cible sélectionnée est un champ indicateur.
set_voting_tie_selection	Random HighestConfidence	Si une méthode de vote est sélectionnée, indique la manière dont les ex æquo sont résolus. Ce paramètre s'applique uniquement si la cible sélectionnée est un champ nominal.
calculate_standard_error	*indicateur*	Si le champ cible est continu, un calcul d'erreur standard est exécuté par défaut pour calculer la différence entre les valeurs mesurées ou estimées et les valeurs réelles, et pour montrer la correspondance proche de ces évaluations.

Propriétés de fillernode

Le noeud Remplacer permet de remplacer les valeurs de champ et de modifier le type de stockage. Vous pouvez décider de remplacer les valeurs reposant sur une condition CLEM, telle que `@BLANK(@FIELD)`. Vous pouvez également choisir de remplacer tous les blancs ou toutes les valeurs nulles par une valeur précise. Un noeud Remplacer est souvent associé à un noeud type pour remplacer les valeurs manquantes.

Exemple

```
node = stream.create("filler", "My node")
node.setPropertyValue("fields", ["Age"])
node.setPropertyValue("replace_mode", "Always")
node.setPropertyValue("condition", "(\"Age\" > 60) and (\"Sex\" = \"M\"")
node.setPropertyValue("replace_with", "\"old man\"")
```

Tableau 75. propriétés de fillernode

Propriétés de fillernode	Le type de données	Description de la propriété
fields	*liste (list)*	Champs du jeu de données dont les valeurs sont examinées et remplacées.

Tableau 75. propriétés de fillernode (suite)

Propriétés de fillernode	Le type de données	Description de la propriété
replace_mode	Always Conditional Blanc Valeur nulle BlankAndNull	Vous pouvez remplacer toutes les valeurs, les valeurs vides ou les valeurs nulles, ou effectuer le remplacement en fonction d'une condition définie.
condition	*chaîne*	
replace_with	*chaîne*	

Propriétés de filternode

Le noeud Filtrer filtre (supprime) les champs, les renomme et les mappe entre un noeud source et un autre.

Exemple

```
node = stream.create("filter", "My node")
node.setPropertyValue("default_include", True)
node.setKeyedPropertyValue("new_name", "Drug", "Chemical")
node.setKeyedPropertyValue("include", "Drug", False)
```

Utilisation de la propriété default_include. Le paramétrage de la valeur de la propriété `default_include` n'implique pas l'inclusion ou l'exclusion automatique de tous les champs ; il détermine simplement la valeur par défaut pour la sélection actuelle. Ceci équivaut, d'un point de vue fonctionnel, à cliquer sur le bouton **Inclure les champs par défaut** dans la boîte de dialogue du noeud Filtrer. Supposons, par exemple, que vous exécutiez le script suivant :

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["Age", "Sex"]:
    node.setKeyedPropertyValue("include", f, True)
```

Dans ce cas, le noeud transmet les champs *Age* et *Sexe* et ignore les autres. Supposons maintenant que vous exécutiez le même script avec deux autres champs :

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["BP", "Na"]:
    node.setKeyedPropertyValue("include", f, True)
```

Deux champs supplémentaires sont ajoutés au filtre ; par conséquent, quatre champs au total sont transmis (*Age*, *Sexe*, *TA*, *Na*). En d'autres termes, le fait de paramétrer de nouveau la valeur de la propriété `default_include` sur `False` (faux) ne réinitialise pas automatiquement tous les champs.

D'autre part, si vous paramétrez maintenant la propriété `default_include` sur `True` (vrai), par le biais d'un script ou de la boîte de dialogue du noeud Filtrer, le résultat est inversé et les quatre champs répertoriés ci-dessus sont ignorés plutôt qu'inclus. Si vous avez un doute, manipulez les commandes de la boîte de dialogue du noeud Filtrer pour comprendre cette interaction.

Tableau 76. propriétés de filternode

Propriétés de filternode	Le type de données	Description de la propriété
default_include	*flag*	Propriété saisie permettant d'indiquer si le comportement par défaut consiste à transmettre ou à filtrer les champs : Le paramétrage de cette propriété n'implique pas l'inclusion ou l'exclusion automatique de tous les champs ; il détermine simplement si les champs sélectionnés sont inclus ou exclus par défaut. Pour consulter d'autres commentaires, reportez-vous à l'exemple ci-dessous.
include	*flag*	Propriété saisie (inclusion et suppression de champs).
new_name	*chaîne*	

Propriétés de historynode

Le noeud Historiser crée des champs contenant des données provenant de champs d'enregistrements antérieurs. Les noeuds Historiser sont souvent utilisés pour les données séquentielles, telles que les séries temporelles. Avant d'utiliser un noeud Historiser, vous pouvez trier les données à l'aide d'un noeud Trier.

Exemple

```
node = stream.create("history", "My node")
node.setPropertyValue("fields", ["Drug"])
node.setPropertyValue("offset", 1)
node.setPropertyValue("span", 3)
node.setPropertyValue("unavailable", "Discard")
node.setPropertyValue("fill_with", "undef")
```

Tableau 77. propriétés de historynode

Propriétés de historynode	Le type de données	Description de la propriété
fields	*liste (list)*	Champs pour lesquels vous souhaitez un historique.
décalage	*nombre*	Indique le dernier enregistrement (avant l'enregistrement actuel) à partir duquel extraire les valeurs de champ historiques.
span	*nombre*	Indique le nombre d'enregistrements précédents desquels extraire des valeurs.
unavailable	Discard Leave Fill	Propriété destinée à la gestion des enregistrements qui n'ont aucune valeur historique ; fait généralement référence aux premiers enregistrements (situés au début du jeu de données) et pour lesquels il n'existe aucun enregistrement précédent à utiliser en tant qu'historique.
fill_with	Chaîne Nombre	Indique la valeur ou la chaîne à utiliser pour les enregistrements dans lesquels aucune valeur d'historique n'est disponible.

Propriétés de partitionnode

Le noeud Partitionner génère un champ de partition qui répartit les données dans des sous-ensembles distincts pour les étapes d'apprentissage, de test et de validation de la création d'un modèle.

Exemple

```
node = stream.create("partition", "My node")
node.setPropertyValue("create_validation", True)
node.setPropertyValue("training_size", 33)
node.setPropertyValue("testing_size", 33)
node.setPropertyValue("validation_size", 33)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 123)
node.setPropertyValue("value_mode", "System")
```

Tableau 78. propriétés de partitionnode

Propriétés de partitionnode	Le type de données	Description de la propriété
new_name	*chaîne*	Nom du champ de partition généré par le nœud.
create_validation	*flag*	Indique si une partition de validation doit être créée.
training_size	*entier*	Pourcentage des enregistrements (0 à 100) à allouer à la partition d'apprentissage.
testing_size	*entier*	Pourcentage des enregistrements (0 à 100) à allouer à la partition de test.
validation_size	*entier*	Pourcentage des enregistrements (0 à 100) à allouer à la partition de validation. Ignoré si aucune partition de validation n'est créée.
training_label	*chaîne*	Libellé de la partition d'apprentissage.
testing_label	*chaîne*	Libellé de la partition de test.
validation_label	*chaîne*	Libellé de la partition de validation. Ignoré si aucune partition de validation n'est créée.
value_mode	System SystemAndLabel Libellé	Indique les valeurs utilisées pour représenter chaque partition dans les données. Par exemple, l'échantillon d'apprentissage peut être représenté par l'entier système 1, le libellé Training ou une combinaison des deux : 1_Training.
set_random_seed	*Booléen*	Indique si vous devez utiliser une valeur de départ aléatoire définie par l'utilisateur.
random_seed	*entier*	Valeur de départ aléatoire définie par l'utilisateur. Pour utiliser cette valeur, set_random_seed doit avoir la valeur True (vrai).
enable_sql_generation	*Booléen*	Spécifie s'il faut utiliser la répercussion SQL pour affecter des enregistrements à des partitions.
unique_field		Spécifie le champ d'entrée utilisé pour vérifier que des enregistrements sont attribués à des partitions de manière aléatoire mais répétitive. Pour utiliser cette valeur, enable_sql_generation doit avoir la valeur True (vrai).

Propriétés de reclassifynode

Le noeud Recoder permet de transformer un ensemble de valeurs catégorielles en un autre. La recodification est utile pour réduire des catégories ou regrouper des données à analyser.

Exemple

```
node = stream.create("reclassify", "My node")
node.setPropertyValue("mode", "Multiple")
node.setPropertyValue("replace_field", True)
node.setPropertyValue("field", "Drug")
node.setPropertyValue("new_name", "Chemical")
node.setPropertyValue("fields", ["Drug", "BP"])
node.setPropertyValue("name_extension", "reclassified")
node.setPropertyValue("add_as", "Prefix")
node.setKeyedPropertyValue("reclassify", "drugA", True)
node.setPropertyValue("use_default", True)
node.setPropertyValue("default", "BrandX")
node.setPropertyValue("pick_list", ["BrandX", "Placebo", "Generic"])
```

Tableau 79. propriétés de reclassifynode

Propriétés de reclassifynode	Le type de données	Description de la propriété
mode	Single Multiple	La propriété Single recodifie les catégories d'un champ. La propriété Multiple active les options permettant de transformer plusieurs champs simultanément.
replace_field	*flag*	
zone	*chaîne*	Utilisé en mode Simple uniquement.
new_name	*chaîne*	Utilisé en mode Simple uniquement.
fields	*[champ1 champ2 ... champN]*	Utilisé en mode Multiple uniquement.
name_extension	*chaîne*	Utilisé en mode Multiple uniquement.
add_as	Suffix Préfixe	Utilisé en mode Multiple uniquement.
recoder	*chaîne*	Propriété structurée (valeur des champs).
use_default	*flag*	Utilisez la valeur par défaut.
default	*chaîne*	Spécifiez une valeur par défaut.
pick_list	*[string string ... string]*	Permet à l'utilisateur d'importer la liste des nouvelles valeurs connues pour remplir la liste déroulante de la table.

Propriétés de reordernode

Le noeud Re-trier définit l'ordre naturel utilisé pour afficher les champs situés en aval. Cet ordre a une incidence sur l'affichage des champs en différents endroits : tableaux, listes et sélecteur de champs. Cette opération est utile lorsque vous utilisez des jeux de données volumineux pour rendre plus visibles les champs intéressants.

Exemple

```
node = stream.create("reorder", "My node")
node.setPropertyValue("mode", "Custom")
node.setPropertyValue("sort_by", "Storage")
node.setPropertyValue("ascending", False)
node.setPropertyValue("start_fields", ["Age", "Cholesterol"])
node.setPropertyValue("end_fields", ["Drug"])
```

Tableau 80. propriétés de reordernode

Propriétés de reordernode	Le type de données	Description de la propriété
mode	Custom Automatique	Vous pouvez trier les valeurs automatiquement ou indiquer un ordre personnalisé.
sort_by	Name Type Stockage	
ascending	*flag*	
start_fields	*[champ1 champ2 ... champn]*	Les nouveaux champs sont insérés après ces champs.
end_fields	*[champ1 champ2 ... champn]*	Les nouveaux champs sont insérés avant ces champs.

Propriétés reprojectnode

Dans SPSS Modeler, les éléments tels que les fonctions spatiales du générateur de formules, le noeud Spatio-Temporal Prediction (STP) et le noeud Visualisation de carte utilisent le système de coordonnées projetées. Utilisez le noeud Reprojeter pour changer le système de coordonnées des données que vous importez et qui utilisent un système de coordonnées géographiques.

Tableau 81. Propriétés reprojectnode

Propriétés reprojectnode	Type de données	Description de la propriété
reproject_fields	*[champ1 champ2 ... champn]*	Liste de tous les champs à reprojeter.
reproject_type	Streamdefault Specify	Sélectionnez le mode de reprojection des champs.
coordinate_system	*chaîne*	Nom du système de coordonnées à appliquer aux champs. Exemple : `set reprojectnode.coordinate_system = "WGS_1984_World_Mercator"`

Propriétés de restructurenode

Le noeud Restructurer convertit un champ nominal ou un champ indicateur en un groupe de champs renseignés à partir des valeurs d'un autre champ. Par exemple, si l'on considère un champ nommé *type de paiement*, qui comporte les valeurs *crédit*, *liquide* et *débit*, trois champs sont alors créés (*crédit*, *liquide*, *débit*), chacun contenant la valeur du paiement réel effectué.

Exemple

```
node = stream.create("restructure", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("include_field_name", True)
node.setPropertyValue("value_mode", "OtherFields")
node.setPropertyValue("value_fields", ["Age", "BP"])
```

Tableau 82. propriétés de restructurenode

Propriétés de restructurenode	Le type de données	Description de la propriété
fields_from	[*catégorie catégorie catégorie*] toutes	
include_field_name	*flag*	Indique si le nom de champ doit être utilisé dans le nom de champ restructuré.
value_mode	OtherFields Indicateurs	Indique le mode de spécification des valeurs pour les champs restructurés. Avec OtherFields, vous devez indiquer les champs à utiliser (voir ci-dessous). Avec Flags, les valeurs sont des indicateurs numériques.
value_fields	*liste (list)*	Obligatoire si la propriété value_mode a la valeur OtherFields. Indique les champs à utiliser en tant que champs de valeur.

Propriétés de rfmanalysisnode

Le noeud Analyse RFM (Récence, Effectif, Monétaire) permet de déterminer de façon quantitative les clients susceptibles d'être les meilleurs par l'étude de leur dernier achat (récence), l'effectif de leurs achats (effectif), et la somme dépensée lors de toutes les transactions (monétaire).

Exemple

```
node = stream.create("rfmanalysis", "My node")
node.setPropertyValue("recency", "Recency")
node.setPropertyValue("frequency", "Frequency")
node.setPropertyValue("monetary", "Monetary")
node.setPropertyValue("tied_values_method", "Next")
node.setPropertyValue("recalculate_bins", "IfNecessary")
node.setPropertyValue("recency_thresholds", [1, 500, 800, 1500, 2000, 2500])
```

Tableau 83. propriétés de rfmanalysisnode

Propriétés de rfmanalysisnode	Le type de données	Description de la propriété
recency	*zone*	Indiquez le champ de récence. Il peut s'agir d'une date, d'un horodatage ou d'un simple nombre.
frequency	*zone*	Indiquez le champ de fréquence.
monetary	*zone*	Spécifiez le champ monétaire.
recency_bins	*entier*	Indiquez le nombre d'intervalles de récence à générer.
recency_weight	*nombre*	Indiquez la pondération à appliquer aux données de récence. La valeur par défaut est 100.
frequency_bins	*entier*	Indiquez le nombre d'intervalles de fréquence à générer.

Tableau 83. propriétés de rfmanalysisnode (suite)

Propriétés de rfmanalysisnode	Le type de données	Description de la propriété
frequency_weight	*nombre*	Indiquez la pondération à appliquer aux données de fréquence. La valeur par défaut est 10.
monetary_bins	*entier*	Indiquez le nombre d'intervalles monétaires à générer.
monetary_weight	*nombre*	Indiquez la pondération à appliquer aux données monétaires. La valeur par défaut est 1.
tied_values_method	Suivant Current	Spécifiez l'intervalle dans lequel les données de valeur ex æquo doivent être placées.
recalculate_bins	Always IfNecessary	
add_outliers	*flag*	Disponible uniquement si recalculate_bins est défini sur IfNecessary. Si cette valeur est définie, les enregistrements situés sous l'intervalle inférieur seront ajoutés à celui-ci et les enregistrements situés au-dessus de l'intervalle supérieur seront ajoutés à ce dernier.
binned_field	Recency Frequency Monétaire	
recency_thresholds	*valeur valeur*	Disponible uniquement si recalculate_bins est défini sur Always. Spécifiez les seuils supérieur et inférieur pour les intervalles de récence. Le seuil supérieur d'un intervalle est aussi le seuil inférieur de l'intervalle suivant ; par exemple, [10 30 60] définirait deux intervalles, le premier présentant des seuils supérieur et inférieur de 10 et 30, les seuils du deuxième intervalle étant de 30 et 60.
frequency_thresholds	*valeur valeur*	Disponible uniquement si recalculate_bins est défini sur Always.
monetary_thresholds	*valeur valeur*	Disponible uniquement si recalculate_bins est défini sur Always.

Propriétés de settoflagnode

Le noeud Binariser calcule plusieurs champs indicateurs en fonction des valeurs catégorielles définies pour un ou plusieurs champs nominaux.

Exemple

```
node = stream.create("settoflag", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("true_value", "1")
node.setPropertyValue("false_value", "0")
node.setPropertyValue("use_extension", True)
```

```
node.setPropertyValue("extension", "Drug_Flag")
node.setPropertyValue("add_as", "Suffix")
node.setPropertyValue("aggregate", True)
node.setPropertyValue("keys", ["Cholesterol"])
```

Tableau 84. propriétés de settoflagnode

Propriétés de settoflagnode	Le type de données	Description de la propriété
fields_from	[*catégorie catégorie catégorie*] toutes	
true_value	*chaîne*	Indiquez la valeur true (vrai) utilisée par le nœud lors de la définition d'un indicateur. La valeur par défaut est T.
false_value	*chaîne*	Indique la valeur false (faux) utilisée par le nœud lors de la définition d'un indicateur. La valeur par défaut est F.
use_extension	*flag*	Utilisez une extension en tant que suffixe ou préfixe pour le nouveau champ indicateur.
extension	*chaîne*	
add_as	Suffix Préfixe	Indique si l'extension est ajoutée en tant que suffixe ou préfixe.
aggregate	*flag*	Regroupe des enregistrements sur la base des champs-clés. Tous les champs indicateurs d'un groupe sont activés si un enregistrement est paramétré sur true (vrai).
keys	*liste (list)*	Champs-clés.

Propriétés statisticstransformnode

Le noeud Transformation exécute une sélection de commandes de syntaxe IBM SPSS Statistics en fonction des sources de données dans IBM SPSS Modeler. Ce noeud requiert une copie avec licence de IBM SPSS Statistics.

Les propriétés de ce nœud sont décrites dans «Propriétés statisticstransformnode», à la page 299.

Propriétés de timeintervalsnode

Le noeud Intervalle de temps définit des intervalles et crée, si nécessaire, des libellés pour la modélisation des séries temporelles. Si les valeurs ne sont pas espacées de manière égale, ce noeud peut les remplir ou les agréger, selon les besoins, pour générer un intervalle uniforme entre les enregistrements.

Exemple

```
node = stream.create("timeintervals", "My node")
node.setPropertyValue("interval_type", "SecondsPerDay")
node.setPropertyValue("days_per_week", 4)
node.setPropertyValue("week_begins_on", "Tuesday")
node.setPropertyValue("hours_per_day", 10)
node.setPropertyValue("day_begins_hour", 7)
node.setPropertyValue("day_begins_minute", 5)
```

```
node.setPropertyValue("day_begins_second", 17)
node.setPropertyValue("mode", "Label")
node.setPropertyValue("year_start", 2005)
node.setPropertyValue("month_start", "January")
node.setPropertyValue("day_start", 4)
node.setKeyedPropertyValue("pad", "AGE", "MeanOfRecentPoints")
node.setPropertyValue("agg_mode", "Specify")
node.setPropertyValue("agg_set_default", "Last")
```

Tableau 85. propriétés de timeintervalsnode.

Propriétés de timeintervalsnode	**Le type de données**	**Description de la propriété**
interval_type	None Periods CyclicPeriods Years Quarters Mois DaysPerWeek DaysNonPeriodic HoursPerDay HoursNonPeriodic MinutesPerDay MinutesNonPeriodic SecondsPerDay SecondsNonPeriodic	
mode	Label Create	Indique si vous souhaitez étiqueter les enregistrements de manière consécutive ou créer la série sur la base d'un champ de date, d'horodatage ou d'heure spécifié.
field	*champ*	Lorsque la série est créée à partir des données, indique le champ qui détermine la date ou l'heure de chaque enregistrement.
period_start	*entier*	Indique l'intervalle de début pour les périodes ou les périodes cycliques.
cycle_start	*entier*	Cycle de début pour les périodes cycliques.
year_start	*entier*	Pour les types d'intervalle concernés, année qui inclut le premier intervalle.
quarter_start	*entier*	Pour les types d'intervalle concernés, trimestre qui inclut le premier intervalle.
month_start	Janvier Février Mars Avril Mai Juin Juillet Août Septembre Octobre Novembre Décembre	
day_start	*entier*	
hour_start	*entier*	
minute_start	*entier*	
second_start	*entier*	

Tableau 85. propriétés de timeintervalsnode (suite).

Propriétés de timeintervalsnode	Le type de données	Description de la propriété
periods_per_cycle	*entier*	Pour les périodes cycliques, nombre de périodes au sein de chaque cycle.
fiscal_year_begins	Janvier Février Mars Avril Mai Juin Juillet Août Septembre Octobre Novembre Décembre	Pour les intervalles trimestriels, indique le mois au cours duquel l'exercice commence.
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday Sunday	Pour les intervalles périodiques (jours par semaine, heures par jour, minutes par jour et secondes par jour), indique le jour au cours duquel la semaine commence.
day_begins_hour	*entier*	Pour les intervalles périodiques (heures par jour, minutes par jour, secondes par jour), indique l'heure au cours de laquelle le jour commence. Il est possible d'utiliser cette propriété en combinaison avec day_begins_minute et day_begins_second pour indiquer une heure exacte telle que *8:05:01*. Reportez-vous à l'exemple de syntaxe ci-dessous.
day_begins_minute	*entier*	Pour les intervalles périodiques (heures par jour, minutes par jour, secondes par jour), indique la minute au cours de laquelle le jour commence (par exemple, *5* dans *8:05*).
day_begins_second	*entier*	Pour les intervalles périodiques (heures par jour, minutes par jour, secondes par jour), indique la seconde au cours de laquelle le jour commence (par exemple, *17* dans *8:05:17*).
days_per_week	*entier*	Pour les intervalles périodiques (jours par semaine, heures par jour, minutes par jour et secondes par jour), indique le nombre de jours par semaine.
hours_per_day	*entier*	Pour les intervalles périodiques (heures par jour, minutes par jour et secondes par jour), indique le nombre d'heures par jour.

Tableau 85. propriétés de timeintervalsnode (suite).

Propriétés de **timeintervalsnode**	Le type de données	Description de la propriété
interval_increment	1 2 3 4 5 6 10 15 20 30	Pour les minutes par jour et les secondes par jour, indique le nombre de minutes ou de secondes qui fait l'objet de l'incrément pour chaque enregistrement.
field_name_extension	*chaîne*	
field_name_extension_as_prefix	*indicateur*	
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	

Tableau 85. propriétés de timeintervalsnode (suite).

Propriétés de timeintervalsnode	Le type de données	Description de la propriété
aggregate	Mean Sum Mode Min Max First Last TrueIfAnyTrue	Indique la méthode d'agrégation pour un champ.
pad	Blanc MeanOfRecentPoints True False	Indique la méthode de remplissage pour un champ.
agg_mode	All Specify	Indique si l'agrégation ou le remplissage de tous les champs doit avoir lieu à l'aide des fonctions par défaut (si nécessaire) ou si les champs et les fonctions à utiliser doivent être spécifiés.
agg_range_default	Mean Sum Mode Min Max	Indique la fonction par défaut à utiliser lors de l'agrégation de champs continus.
agg_set_default	Mode First Last	Indique la fonction par défaut à utiliser lors de l'agrégation de champs nominaux.
agg_flag_default	TrueIfAnyTrue Mode First Last	
pad_range_default	Blanc MeanOfRecentPoints	Indique la fonction par défaut à utiliser lors du remplissage de champs continus.
pad_set_default	Blanc MostRecentValue	
pad_flag_default	Blanc True False	
max_records_to_create	*entier*	Indique le nombre maximal d'enregistrements à créer lors du remplissage de la série.
estimation_from_beginning	*indicateur*	
estimation_to_end	*indicateur*	
estimation_start_offset	*entier*	
estimation_num_holdouts	*entier*	
create_future_records	*indicateur*	
num_future_records	*entier*	
create_future_field	*indicateur*	
future_field_name	*chaîne*	

Propriétés de transposenode

Le noeud Transposer fait passer les données des lignes vers les colonnes (et réciproquement) de sorte que les enregistrements deviennent des champs et les champs des enregistrements.

Exemple

```
node = stream.create("transpose", "My node")
node.setPropertyValue("transposed_names", "Read")
node.setPropertyValue("read_from_field", "TimeLabel")
node.setPropertyValue("max_num_fields", "1000")
node.setPropertyValue("id_field_name", "ID")
```

Tableau 86. propriétés de transposenode

Propriétés de transposenode	Le type de données	Description de la propriété
transposed_names	Prefix Lire	Les nouveaux noms de champ peuvent être générés automatiquement à partir d'un préfixe défini ou ils peuvent être lus à partir d'un champ existant dans les données.
prefix	*chaîne*	
num_new_fields	*entier*	Lorsqu'un préfixe est utilisé, indique le nombre maximal de champs à créer.
read_from_field	*zone*	Champ à partir duquel les noms sont lus. Il doit s'agir d'un champ instancié ; sinon, une erreur apparaît lorsque le noeud est exécuté.
max_num_fields	*entier*	Lorsque des noms sont lus à partir d'un champ, indique une limite supérieure afin d'éviter de créer un nombre de champs trop important.
transpose_type	Numérique Chaîne Personnalisé	Par défaut, seuls les champs continus (intervalle numérique) sont transposés, mais vous pouvez choisir un sous-ensemble personnalisé de champs numériques ou choisir de transposer tous les champs de type chaîne.
transpose_fields	*liste (list)*	Indique les champs à transposer lorsque l'option Custom est utilisée.
id_field_name	*zone*	

Propriétés de typenode

Le noeud type définit les propriétés et métadonnées de champ. Par exemple, vous pouvez indiquer un niveau de mesure (continu, nominal, ordinal ou indicateur) pour chaque champ, définir des options pour la gestion des valeurs manquantes et des valeurs système nulles, spécifier le rôle d'un champ en vue de la modélisation, définir des libellés de champ et de valeur, et indiquer les valeurs d'un champ.

Exemple

```
node = stream.createAt("type", "My node", 50, 50)
node.setKeyedPropertyValue("check", "Cholesterol", "Coerce")
node.setKeyedPropertyValue("direction", "Drug", "Input")
node.setKeyedPropertyValue("type", "K", "Range")
node.setKeyedPropertyValue("values", "Drug", ["drugA", "drugB", "drugC", "drugD", "drugX",
```

```
 "drugY", "drugZ"])
node.setKeyedPropertyValue("null_missing", "BP", False)
node.setKeyedPropertyValue("whitespace_missing", "BP", False)
node.setKeyedPropertyValue("description", "BP", "Blood Pressure")
node.setKeyedPropertyValue("value_labels", "BP", [["HIGH", "High Blood Pressure"],
 ["NORMAL", "normal blood pressure"]])
```

Dans certains cas, le noeud type doit être complètement instancié de façon à ce que d'autres noeuds fonctionnent correctement, notamment la propriété `fields from` du noeud Binariser. Vous pouvez simplement connecter un noeud Table et l'exécuter pour instancier les champs :

```
tablenode = stream.createAt("table", "Table node", 150, 50)
stream.link(node, tablenode)
tablenode.run(None)
stream.delete(tablenode)
```

Tableau 87. propriétés de typenode.

Propriétés de typenode	Le type de données	Description de la propriété

`direction`	`Input`
	`Target`
	`Both`
	`None`
	`Partition`
	`Split`
	`Frequency`
	`RecordID`
		Propriété saisie (rôles des champs).
		Remarque : Les valeurs `In` et `Out` sont désormais obsolètes. Leur prise en charge pourrait être supprimée dans une version ultérieure.

`type`	`Range`
	`Indicateur`
	`Set`
	`Sans type`
	`Discrete`
	`OrderedSet`
	`Default`
		Niveau de mesure du champ (auparavant appelé "type" du champ).
		La définition de `type` sur `Default` effacera toute valeur du paramètre `values` et si `value_mode` a la valeur `Specify`, ce paramètre sera réinitialisé à `Read`.
		Si la propriété `value_mode` est paramétrée sur `Pass` ou `Read`, la définition du `type` n'a aucune incidence sur `value_mode`.
		Remarque : Les types de données utilisés en interne diffèrent de ceux visibles dans le noeud de type. La correspondance est comme suit : Range -> Continuous Set -> Nominal OrderedSet -> Ordinal Discrete- > Categorical

`storage`	`Inconnu`
	`String`
	`Integer`
	`Real`
	`Time`
	`Date`
	`Horodatage`
		Propriété saisie en lecture seule pour le type de stockage de champ.

`check`	`None`
	`Nullify`
	`Coerce`
	`Discard`
	`Warn`
	`Abandonner`
		Propriété saisie (vérification du type et de l'intervalle des champs).

`valeurs`	[*valeur valeur*]	Pour les champs continus , la première valeur représente la valeur minimale et la dernière valeur, la valeur maximale. Pour des champs nominaux, spécifiez toutes les valeurs. Pour les champs indicateurs, la première valeur représente *false* (faux) et la dernière valeur, *true* (vrai). La définition automatique de cette propriété paramètre la propriété `value_mode` sur `Specify`.

`value_mode`	`Read`
	`Pass`
	`Read+`
	`Current`
	`Spécifier`
		Détermine le mode de définition des valeurs. Remarque : vous ne pouvez pas paramétrer directement cette propriété sur `Spécifier`. Pour utiliser des valeurs spécifiques, paramétrez la propriété `values`.

Tableau 87. propriétés de typenode (suite).

Propriétés de typenode	Le type de données	Description de la propriété
extend_values	*flag*	S'applique lorsque la propriété value_mode est paramétrée sur Read. Paramétrez cette valeur sur T pour ajouter des valeurs qui viennent d'être lues aux valeurs existantes du champ. Paramétrez cette valeur sur F pour supprimer des valeurs existantes en faveur des valeurs qui viennent d'être lues.
enable_missing	*flag*	Lorsque cette propriété est paramétrée sur T, elle active le suivi des valeurs manquantes du champ.
missing_values	[*valeur valeur ...*]	Spécifie les valeurs de données qui indiquent les données manquantes.
range_missing	*flag*	Indique si un intervalle de valeurs manquantes (non renseignées) a été défini pour un champ.
missing_lower	*chaîne*	Lorsque le paramètre range_missing a pour valeur true (vrai), indique la limite inférieure de l'intervalle des valeurs manquantes.
missing_upper	*chaîne*	Lorsque le paramètre range_missing a pour valeur true, indique la limite supérieure de l'intervalle des valeurs manquantes.
null_missing	*flag*	Lorsque cette propriété est paramétrée sur T, les valeurs *nulles* (valeurs non définies affichées sous la forme $null$ dans le logiciel) sont considérées comme des valeurs manquantes.
whitespace_missing	*flag*	Lorsque cette propriété est paramétrée sur T, les valeurs composées uniquement de blancs (espaces, tabulations et caractères de nouvelle ligne) sont considérées comme des valeurs manquantes.
description	*chaîne*	Définit la description d'un champ.
value_labels	*[{Valeur ChaîneLibellé} { Valeur ChaîneLibellé} ...]*	Permet d'attribuer des libellés aux paires de valeurs.
display_places	*entier*	Définit le nombre de décimales du champ lorsqu'il est affiché (s'applique uniquement aux champs dont le stockage est REAL (REEL)). La valeur –1 utilise le flux par défaut.
export_places	*entier*	Définit le nombre de décimales du champ lorsqu'il est exporté (s'applique uniquement aux champs dont le stockage est REAL (REEL)). La valeur –1 utilise le flux par défaut.
decimal_separator	DEFAULT PERIOD COMMA	Définit le séparateur décimal du champ (s'applique uniquement aux champs dont le stockage est REAL (REEL)).
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	Définit le format de date du champ (s'applique uniquement aux champs dont le stockage est DATE ou TIMESTAMP).

Tableau 87. propriétés de typenode (suite).

Propriétés de typenode	Le type de données	Description de la propriété
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Définit le format d'heure du champ (s'applique uniquement aux champs dont le stockage est TIME ou TIMESTAMP).
number_format	DEFAULT Standard SCIENTIFIC CURRENCY	Définit le format d'affichage des nombres du champ.
standard_places	*entier*	Définit le nombre de décimales du champ lorsqu'il est affiché au format standard. La valeur –1 utilise le flux par défaut. La propriété display_places existante modifie également ce nombre, mais a été remplacée.
scientific_places	*entier*	Définit le nombre de décimales du champ lorsqu'il est affiché au format scientifique. La valeur –1 utilise le flux par défaut.
currency_places	*entier*	Définit le nombre de décimales du champ lorsqu'il est affiché au format monétaire. La valeur –1 utilise le flux par défaut.
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Définit le symbole de regroupement du champ.
column_width	*entier*	Définit la largeur des colonnes du champ. La valeur –1 définit la largeur des colonnes sur Auto.
justify	AUTO CENTER LEFT RIGHT	Définit la justification des colonnes du champ.
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Cette propriété saisie est similaire à type dans la mesure où elle peut être utilisée pour définir la mesure associée au champ. La différence est que dans un script Python, l'une des valeurs de MeasureType peut également être transmise à la fonction de méthode d'accès set (setter) alors que la méthode d'accès get (getter) est toujours renvoyée sur les valeurs de MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Pour les champs de résumé (listes avec une profondeur de 0), cette propriété saisie définit le type de mesure associé au valeurs sous-jacentes.
geo_type	Point MultiPoint Chaîne MultiChaînes Polygone MultiPolygone	Pour les champs géospatiaux, cette propriété saisie définit le type d'objet géospatial représenté par ce champ. Elle doit être cohérente avec la profondeur de liste des valeurs.
has_coordinate_system	*booléen*	Pour les champs géospatiaux, cette propriété spécifie si le champ comporte un système de coordonnées
coordinate_system	*chaîne*	Pour les champs géospatiaux, cette propriété saisie définit le système de coordonnées pour ce champ.

Tableau 87. propriétés de typenode (suite).

Propriétés de typenode	Le type de données	Description de la propriété
custom_storage_type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	Cette propriété saisie est similaire à custom_storage dans la mesure où elle peut être utilisée pour définir le stockage de substitution pour le champ. La différence est que dans un script Python, l'une des valeurs de StorageType peut également être transmise à la fonction de méthode d'accès set (setter) alors que la méthode d'accès get (getter) est toujours renvoyée sur les valeurs de StorageType.
custom_list_storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Pour les champs de liste, cette propriété saisie spécifie le type de stockage des valeurs sous-jacentes.
custom_list_depth	*entier*	Pour les champs de liste, cette propriété saisie spécifie la profondeur du champ.

Chapitre 12. Propriétés des noeuds Graphiques

Propriétés communes aux noeuds Graphiques

Cette section décrit les propriétés disponibles pour les noeuds Graphiques, notamment les propriétés communes, ainsi que celles propres à chaque type de noeud.

Tableau 88. Propriétés communes aux noeuds Graphiques

Propriétés communes aux noeuds Graphiques	**Le type de données**	**Description de la propriété**
`titre`	*chaîne*	Spécifie le titre. Exemple : « Ceci est un titre ».
`caption`	*chaîne*	Spécifie la légende. Exemple : « Ceci est une légende ».
`output_mode`	`Screen` `Fichier`	Indique si la sortie d'un noeud Graphique doit être affichée ou écrite dans un fichier.

`output_format`	`BMP`
	`JPEG`
	`PNG`
	`HTML`
	`output` (*.cou*)
		Indique le type de sortie. Le type exact de sortie autorisée varie pour chaque nœud.

`full_filename`	*chaîne*	Indique le chemin et le nom de fichier cible de la sortie générée à partir du noeud Graphique.
`use_graph_size`	*flag*	Vérifie si le graphique est dimensionné de manière explicite, à l'aide des propriétés de largeur et de hauteur définies ci-dessous. Ne concerne que les graphiques affichés à l'écran. Non disponible pour le noeud distribution.
`graph_width`	*nombre*	Lorsque `use_graph_size` prend la valeur `True` (vrai), définit la largeur du graphique en pixels.
`graph_height`	*nombre*	Lorsque `use_graph_size` prend la valeur `True` (vrai), définit la hauteur du graphique en pixels.

Désactivation de champs facultatifs

Vous pouvez désactiver les champs facultatifs (par exemple, un champ de superposition pour les nuages) en paramétrant la valeur de la propriété sur " " (chaîne vide), comme le montre l'exemple suivant.

```
plotnode.setPropertyValue("color_field", "")
```

Spécification de couleurs

Vous pouvez indiquer les couleurs des titres, légendes, arrière-plans et libellés à l'aide des chaînes hexadécimales commençant par le symbole dièse (#). Par exemple, pour mettre l'arrière-plan du graphique en bleu ciel, utilisez l'instruction suivante :

```
mygraphnode.setPropertyValue("graph_background", "#87CEEB")
```

Les deux premiers caractères, 87, indiquent le contenu rouge, les deux caractères du milieu, CE, indiquent le contenu vert, et les deux derniers caractères, EB, indiquent le contenu bleu. Chaque caractère peut correspondre à une valeur comprise entre 0–9 ou A–F. Si elles sont associées, ces valeurs peuvent indiquer une couleur RVB (rouge, vert, bleu).

Remarque : Lorsque vous spécifiez des couleurs en RVB, vous pouvez utiliser le sélecteur de champs de l'interface utilisateur pour déterminer le code couleur correct. Déplacez la souris au-dessus de la couleur pour activer l'info-bulle contenant les informations souhaitées.

Propriétés de collectionnode

Le noeud Résumé fournit la proportion de valeurs d'un champ numérique par rapport aux valeurs d'un autre champ. (Il génère des graphiques semblables aux histogrammes.) Il est utile pour illustrer une variable ou un champ dont les valeurs changent avec le temps. Grâce à la représentation graphique en 3D, vous pouvez en outre inclure un axe symbolique affichant les proportions par catégorie.

Exemple

```
node = stream.create("collection", "My node")
# Onglet Tracé
node.setPropertyValue("three_D", True)
node.setPropertyValue("collect_field", "Drug")
node.setPropertyValue("over_field", "Age")
node.setPropertyValue("by_field", "BP")
node.setPropertyValue("operation", "Sum")
# Section Superposer
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# Onglet Options
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1)
node.setPropertyValue("range_max", 100)
node.setPropertyValue("bins", "ByNumber")
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 5)
```

Tableau 89. propriétés de collectionnode

Propriétés de collectionnode	Le type de données	Description de la propriété
over_field	*zone*	
over_label_auto	*flag*	
over_label	*chaîne*	
collect_field	*zone*	
collect_label_auto	*flag*	
collect_label	*chaîne*	
three_D	*flag*	
by_field	*zone*	
by_label_auto	*flag*	
by_label	*chaîne*	
operation	Sum Mean Min Max Ecart type	
color_field	*chaîne*	
panel_field	*chaîne*	

Tableau 89. propriétés de collectionnode (suite)

Propriétés de collectionnode	Le type de données	Description de la propriété
animation_field	*chaîne*	
range_mode	Automatic UserDefined	
range_min	*nombre*	
range_max	*nombre*	
bins	ByNumber ByWidth	
num_bins	*nombre*	
bin_width	*nombre*	
use_grid	*flag*	
graph_background	*Couleur*	Les couleurs de graphique standard sont décrites au début de cette section.
page_background	*Couleur*	Les couleurs de graphique standard sont décrites au début de cette section.

Propriétés de distributionnode

Le noeud distribution fournit l'occurrence des valeurs symboliques (catégorielles), comme un type de prêt hypothécaire ou le sexe d'un individu. Ce noeud est souvent utilisé pour montrer les déséquilibres des données, déséquilibres que vous pouvez rectifier à l'aide d'un noeud Equilibrer avant la création d'un modèle.

Exemple

```
node = stream.create("distribution", "My node")
# Onglet Tracé
node.setPropertyValue("plot", "Flags")
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("normalize", True)
node.setPropertyValue("sort_mode", "ByOccurence")
node.setPropertyValue("use_proportional_scale", True)
```

Tableau 90. propriétés de distributionnode

Propriétés de distributionnode	Le type de données	Description de la propriété
plot	SelectedFields Indicateurs	
x_field	*zone*	
color_field	*zone*	Champ de superposition.
normalize	*flag*	
sort_mode	ByOccurence Alphabétique	
use_proportional_scale	*flag*	

Propriétés de evaluationnode

Le noeud Evaluation permet d'évaluer et de comparer des modèles prédictifs. Le graphique d'évaluation montre l'aptitude des modèles à prédire des résultats spécifiques. Il trie les enregistrements en fonction de la valeur prédite et de la confiance dans cette prévision. Il scinde les enregistrements en groupes de taille égale (**quantiles**), puis reporte la valeur du critère traité pour chaque quantile, du plus élevé au plus faible. Les divers modèles apparaissent sous forme de lignes dans le graphique.

Exemple

```
node = stream.create("evaluation", "My node")
# Onglet Tracé
node.setPropertyValue("chart_type", "Gains")
node.setPropertyValue("cumulative", False)
node.setPropertyValue("field_detection_method", "Name")
node.setPropertyValue("inc_baseline", True)
node.setPropertyValue("n_tile", "Deciles")
node.setPropertyValue("style", "Point")
node.setPropertyValue("point_type", "Dot")
node.setPropertyValue("use_fixed_cost", True)
node.setPropertyValue("cost_value", 5.0)
node.setPropertyValue("cost_field", "Na")
node.setPropertyValue("use_fixed_revenue", True)
node.setPropertyValue("revenue_value", 30.0)
node.setPropertyValue("revenue_field", "Age")
node.setPropertyValue("use_fixed_weight", True)
node.setPropertyValue("weight_value", 2.0)
node.setPropertyValue("weight_field", "K")
```

Tableau 91. propriétés de evaluationnode.

Propriétés de evaluationnode	Le type de données	Description de la propriété
chart_type	Gains Réponse Lift Profit ROI ROC	
inc_baseline	*indicateur*	
field_detection_method	Metadata Name	
use_fixed_cost	*indicateur*	
cost_value	*nombre*	
cost_field	*chaîne*	
use_fixed_revenue	*indicateur*	
revenue_value	*nombre*	
revenue_field	*chaîne*	
use_fixed_weight	*indicateur*	
weight_value	*nombre*	
weight_field	*champ*	

Tableau 91. propriétés de evaluationnode (suite).

Propriétés de evaluationnode	Le type de données	Description de la propriété
n_tile	Quartiles Quintles Deciles Vingtiles Percentiles 1000-tiles	
cumulative	*indicateur*	
style	Line Point	
point_type	Rectangle Point Triangle Hexagone Plus Pentagone Etoile NoeudPapillon TraitHorizontal TraitVertical CroixDeFer Fabrique Maison Cathédrale Bulbe TriangleConcave GlobeAplati OeilDeChat OreillerAQuatreCôtés RectangleArrondi Eventail	
export_data	*indicateur*	
data_filename	*chaîne*	
delimiter	*chaîne*	
new_line	*indicateur*	
inc_field_names	*indicateur*	
inc_best_line	*indicateur*	
inc_business_rule	*indicateur*	
business_rule_condition	*chaîne*	
plot_score_fields	*indicateur*	
score_fields	*[champ1 ... champN]*	
target_field	*champ*	
use_hit_condition	*indicateur*	
hit_condition	*chaîne*	
use_score_expression	*indicateur*	
score_expression	*chaîne*	
caption_auto	*indicateur*	

Propriétés de graphboardnode

Le noeud Représentation Graphique offre de nombreux types de graphiques différents dans un seul noeud. Ce noeud permet de choisir les champs de données que vous souhaitez explorer puis de sélectionner un graphique parmi ceux disponibles pour les données sélectionnées. Le noeud filtre automatiquement tous les types de graphiques ne fonctionnant pas avec les sélections de champs.

Remarque : Si vous définissez une propriété non valide pour le type de graphique (par exemple, si vous spécifiez `y_field` pour un histogramme), cette propriété est ignorée.

Remarque : Dans l'interface utilisateur, sur l'onglet Détaillé de nombreux types de graphique différents, il existe un champ **Récapitulatif** (Summary). Vous ne pouvez pas définir ce champ à l'aide du scriptage. Les types de graphique qui ne prennent pas en charge le champ récapitulatif sont les suivants : 3DArea, 3DBar, 3DHistogram, 3DPie, Area, Bar, BarCounts. BarMap, Histogram, Line, LineChartMap, Pie, PieCounts, Ribbon et Scatterplot.

Exemple

```
node = stream.create("graphboard", "My node")
node.setPropertyValue("graph_type", "Line")
node.setPropertyValue("x_field", "K")
node.setPropertyValue("y_field", "Na")
```

Tableau 92. propriétés de graphboardnode

propriétés de graphboard	**Le type de données**	**Description de la propriété**
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Aire ArrowMap Barre BarCounts BarCountsMap BarMap BinnedScatter Boxplot Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram Line LineChartMap LineOverlayMap Parallel Path Graphique circulaire PieCountMap PieCounts PieMap PointOverlayMap PolygonOverlayMap Ribbon Graphique en nuage de points SPLOM Surface	Identifie le type de graphique.
x_field	*zone*	Spécifie un libellé personnalisé pour l'axe *x*. Disponible uniquement pour les libellés.
y_field	*zone*	Spécifie un libellé personnalisé pour l'axe *y*. Disponible uniquement pour les libellés.
z_field	*zone*	Utilisé dans certains graphiques en 3D
color_field	*zone*	Utilisé dans les cartes de zones de chaleurs.

Tableau 92. propriétés de graphboardnode (suite)

propriétés de graphboard	**Le type de données**	**Description de la propriété**
size_field	*zone*	Utilisé dans les graphiques à bulles.
categories_field	*zone*	
values_field	*zone*	
rows_field	*zone*	
columns_field	*zone*	
fields	*zone*	
start_longitude_field	*zone*	Utilisé avec les flèches sur une carte de référence.
end_longitude_field	*zone*	
start_latitude_field	*zone*	
end_latitude_field	*zone*	
data_key_field	*zone*	Utilisé dans diverses cartes.
panelrow_field	*chaîne*	
panelcol_field	*chaîne*	
animation_field	*chaîne*	
longitude_field	*zone*	Utilisé avec les coordonnées sur les cartes.
latitude_field	*zone*	
map_color_field	*zone*	

Propriétés de histogramnode

Le noeud Histogramme montre l'occurrence des valeurs des champs numériques. Il est souvent utilisé pour explorer les données avant toute création de modèles ou manipulation. Semblable au noeud distribution, le noeud Histogramme sert souvent à montrer les déséquilibres des données.

Exemple

```
node = stream.create("histogram", "My node")
# Onglet Tracé
node.setPropertyValue("field", "Drug")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# Onglet Options
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1.0)
node.setPropertyValue("range_max", 100.0)
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 10)
node.setPropertyValue("normalize", True)
node.setPropertyValue("separate_bands", False)
```

Tableau 93. propriétés de histogramnode

Propriétés de histogramnode	**Le type de données**	**Description de la propriété**
zone	*zone*	
color_field	*zone*	

Tableau 93. propriétés de histogramnode (suite)

Propriétés de histogramnode	Le type de données	Description de la propriété
panel_field	*zone*	
animation_field	*zone*	
range_mode	Automatic UserDefined	
range_min	*nombre*	
range_max	*nombre*	
bins	ByNumber ByWidth	
num_bins	*nombre*	
bin_width	*nombre*	
normalize	*flag*	
separate_bands	*flag*	
x_label_auto	*flag*	
x_label	*chaîne*	
y_label_auto	*flag*	
y_label	*chaîne*	
use_grid	*flag*	
graph_background	*Couleur*	Les couleurs de graphique standard sont décrites au début de cette section.
page_background	*Couleur*	Les couleurs de graphique standard sont décrites au début de cette section.
normal_curve	*flag*	Indique si la courbe de proportion normale doit apparaître sur la sortie.

Propriétés de multiplotnode

Le noeud Courbes génère un graphique qui affiche plusieurs champs *Y* pour un seul champ *X*. Les champs *Y* sont représentés par des lignes colorées. Chacun équivaut à un noeud Tracé dont le style est défini sur **Ligne** et le mode X sur **Trier**. Les graphiques Courbes sont utiles lorsque vous souhaitez étudier la fluctuation de plusieurs variables au fil du temps.

Exemple

```
node = stream.create("multiplot", "My node")
# Onglet Tracé
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("y_fields", ["Drug", "BP"])
node.setPropertyValue("panel_field", "Sex")
# Section Superposer
node.setPropertyValue("animation_field", "")
node.setPropertyValue("tooltip", "test")
node.setPropertyValue("normalize", True)
node.setPropertyValue("use_overlay_expr", False)
node.setPropertyValue("overlay_expression", "test")
node.setPropertyValue("records_limit", 500)
node.setPropertyValue("if_over_limit", "PlotSample")
```

Tableau 94. propriétés de multiplotnode

Propriétés de multiplotnode	Le type de données	Description de la propriété
x_field	*zone*	
y_fields	*liste (list)*	
panel_field	*zone*	
animation_field	*zone*	
normalize	*flag*	
use_overlay_expr	*flag*	
overlay_expression	*chaîne*	
records_limit	*nombre*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*flag*	
x_label	*chaîne*	
y_label_auto	*flag*	
y_label	*chaîne*	
use_grid	*flag*	
graph_background	*Couleur*	Les couleurs de graphique standard sont décrites au début de cette section.
page_background	*Couleur*	Les couleurs de graphique standard sont décrites au début de cette section.

Propriétés de plotnode

Le noeud Tracé montre les relations existant entre les champs numériques. Vous pouvez créer un tracé à l'aide de points (nuage de points) ou de courbes.

Exemple

```
node = stream.create("plot", "My node")
# Onglet Tracé
node.setPropertyValue("three_D", True)
node.setPropertyValue("x_field", "BP")
node.setPropertyValue("y_field", "Cholesterol")
node.setPropertyValue("z_field", "Drug")
# Section Superposer
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("size_field", "Age")
node.setPropertyValue("shape_field", "")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "BP")
node.setPropertyValue("transp_field", "")
node.setPropertyValue("style", "Point")
# Onglet Sortie
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "JPEG")
node.setPropertyValue("full_filename", "C:/temp/graph_output/plot_output.jpeg")
```

Tableau 95. propriétés de plotnode.

Propriétés de plotnode	Le type de données	Description de la propriété
x_field	*champ*	Spécifie un libellé personnalisé pour l'axe *x*. Disponible uniquement pour les libellés.
y_field	*champ*	Spécifie un libellé personnalisé pour l'axe *y*. Disponible uniquement pour les libellés.
three_D	*indicateur*	Spécifie un libellé personnalisé pour l'axe *y*. Disponible uniquement pour les libellés des graphiques en 3D.
z_field	*champ*	
color_field	*champ*	Champ de superposition.
size_field	*champ*	
shape_field	*champ*	
panel_field	*champ*	Sélectionne un champ nominal ou un champ indicateur pour créer un graphique distinct pour chaque catégorie. Les graphiques apparaissent tous sous forme de panneaux dans une fenêtre de sortie.
animation_field	*champ*	Spécifie un champ nominal ou un champ indicateur pour mettre en évidence les catégories des valeurs de données en créant une série de graphiques animés apparaissant les uns après les autres.
transp_field	*champ*	Désigne un champ destiné à mettre en évidence les catégories des valeurs de données en utilisant un degré de transparence différent pour chaque catégorie. Non disponible pour les graphiques de répartition.
overlay_type	None Smoother Function	Indique si une fonction de superposition ou un lissage LOESS est affiché.
overlay_expression	*chaîne*	Définit l'expression utilisée lorsque la valeur Function est affectée à overlay_type.
style	Point Line	
point_type	Rectangle Point Triangle Hexagone Plus Pentagone Etoile NoeudPapillon TraitHorizontal TraitVertical CroixDeFer Fabrique Maison Cathédrale Bulbe TriangleConcave GlobeAplati OeilDeChat OreillerAQuatreCôtés RectangleArrondi Eventail	

Tableau 95. propriétés de plotnode (suite).

Propriétés de plotnode	Le type de données	Description de la propriété
x_mode	Sort Overlay AsRead	
x_range_mode	Automatique UserDefined	
x_range_min	*nombre*	
x_range_max	*nombre*	
y_range_mode	Automatique UserDefined	
y_range_min	*nombre*	
y_range_max	*nombre*	
z_range_mode	Automatique UserDefined	
z_range_min	*nombre*	
z_range_max	*nombre*	
jitter	*indicateur*	
records_limit	*nombre*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*indicateur*	
x_label	*chaîne*	
y_label_auto	*indicateur*	
y_label	*chaîne*	
z_label_auto	*indicateur*	
z_label	*chaîne*	
use_grid	*indicateur*	
graph_background	*couleur*	Les couleurs de graphique standard sont décrites au début de cette section.
page_background	*couleur*	Les couleurs de graphique standard sont décrites au début de cette section.
use_overlay_expr	*indicateur*	Désormais obsolète et remplacé par overlay_type.

Propriétés de timeplotnode

Le noeud Tracé horaire affiche un ou plusieurs jeux de données temporelles. En règle générale, vous utilisez un noeud Intervalles de temps, en premier lieu, pour créer un champ *TimeLabel* qui servira de libellé à l'axe *x*.

Exemple

```
node = stream.create("timeplot", "My node")
node.setPropertyValue("y_fields", ["sales", "men", "women"])
node.setPropertyValue("panel", True)
node.setPropertyValue("normalize", True)
```

```
node.setPropertyValue("line", True)
node.setPropertyValue("smoother", True)
node.setPropertyValue("use_records_limit", True)
node.setPropertyValue("records_limit", 2000)
# Paramètres d'apparence
node.setPropertyValue("symbol_size", 2.0)
```

Tableau 96. propriétés de timeplotnode.

Propriétés de **timeplotnode**	Le type de données	Description de la propriété
plot_series	Series Models	
use_custom_x_field	*indicateur*	
x_field	*champ*	
y_fields	*liste (list)*	
panel	*indicateur*	
normalize	*indicateur*	
line	*indicateur*	
points	*indicateur*	
point_type	Rectangle Point Triangle Hexagone Plus Pentagone Etoile NoeudPapillon TraitHorizontal TraitVertical CroixDeFer Fabrique Maison Cathédrale Bulbe TriangleConcave GlobeAplati OeilDeChat OreillerAQuatreCôtés RectangleArrondi Eventail	
smoother	*indicateur*	Pour pouvoir ajouter des fonctions de lissage au graphique, paramétrez panel sur True (vrai).
use_records_limit	*indicateur*	
records_limit	*entier*	
symbol_size	*nombre*	Définit une taille de symbole.
panel_layout	Horizontal Vertical	

Propriétés de webnode

Le noeud Relations illustre la force de la relation existant entre les valeurs de plusieurs champs symboliques (catégoriels). Le graphique utilise des lignes d'épaisseur différente pour représenter les forces de connexion. Par exemple, vous pouvez utiliser un noeud Relations pour explorer la relation avec l'achat d'un ensemble d'articles sur un site de commerce électronique.

Exemple

```
node = stream.create("web", "My node")
# Onglet Tracé
node.setPropertyValue("use_directed_web", True)
node.setPropertyValue("to_field", "Drug")
node.setPropertyValue("fields", ["BP", "Cholesterol", "Sex", "Drug"])
node.setPropertyValue("from_fields", ["BP", "Cholesterol", "Sex"])
node.setPropertyValue("true_flags_only", False)
node.setPropertyValue("line_values", "Absolute")
node.setPropertyValue("strong_links_heavier", True)
# Onglet Options
node.setPropertyValue("max_num_links", 300)
node.setPropertyValue("links_above", 10)
node.setPropertyValue("num_links", "ShowAll")
node.setPropertyValue("discard_links_min", True)
node.setPropertyValue("links_min_records", 5)
node.setPropertyValue("discard_links_max", True)
node.setPropertyValue("weak_below", 10)
node.setPropertyValue("strong_above", 19)
node.setPropertyValue("link_size_continuous", True)
node.setPropertyValue("web_display", "Circular")
```

Tableau 97. propriétés de webnode

Propriétés de webnode	Le type de données	Description de la propriété
use_directed_web	*flag*	
fields	*liste (list)*	
to_field	*zone*	
from_fields	*liste (list)*	
true_flags_only	*flag*	
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	*flag*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*nombre*	
links_above	*nombre*	
discard_links_min	*flag*	
links_min_records	*nombre*	
discard_links_max	*flag*	
links_max_records	*nombre*	
weak_below	*nombre*	
strong_above	*nombre*	
link_size_continuous	*flag*	
web_display	Circular Réseau Directed Grid	

Tableau 97. propriétés de webnode (suite)

Propriétés de webnode	Le type de données	Description de la propriété
graph_background	*Couleur*	Les couleurs de graphique standard sont décrites au début de cette section.
symbol_size	*nombre*	Définit une taille de symbole.

Chapitre 13. Propriétés des noeuds de modélisation

Propriétés communes des noeuds de modélisation

Les propriétés suivantes sont communes à certains ou à tous les noeuds de modélisation. Les éventuelles exceptions sont notées dans la documentation des noeuds de modélisation individuels, le cas échéant.

Tableau 98. Propriétés communes des noeuds de modélisation

Propriété	Valeurs	Description de la propriété
`custom_fields`	*flag*	Si la valeur est définie sur true (vrai), elle permet de spécifier les champs cible, entrée et d'autres champs pour le nœud actuel. Si elle est définie sur false (faux), les paramètres actuels provenant d'un noeud type en amont sont utilisés.

`target`
ou
`targets`
	champ
	
	ou
	[*champ1 ... champN*]
		Spécifie un ou plusieurs champs cibles selon le type de modèle.

`inputs`	[*champ1 ... champN*]	Champs d'entrée ou prédicteur utilisés par le modèle.
`partition`	*zone*	
`use_partitioned_data`	*flag*	Si un champ de partition est défini, cette option assure que seules les données provenant de la partition d'apprentissage sont utilisées pour générer le modèle.
`use_split_data`	*flag*	
`splits`	*[champ1 ... champN]*	Indique le champ ou les champs à utiliser pour la modélisation découpée. Activé uniquement si `use_split_data` est défini sur `True`.
`use_frequency`	*flag*	Les champs de pondération et de fréquence sont utilisés par des modèles spécifiques comme indiqué pour chaque type de modèle.
`frequency_field`	*zone*	
`use_weight`	*flag*	
`weight_field`	*zone*	
`use_model_name`	*flag*	
`model_name`	*chaîne*	Nom personnalisé du nouveau modèle.
`mode`	`Simple` `Expert`	

Propriétés de anomalydetectionnode

Le noeud Détection des anomalies identifie les observations inhabituelles, ou valeurs éloignées, qui ne se conforment pas aux motifs de données “normales”. Il vous permet d'identifier les valeurs éloignées même si celles-ci ne correspondent pas aux motifs connus précédemment et même si vous ne savez pas exactement ce que vous recherchez.

Exemple

```
node = stream.create("anomalydetection", "My node")
node.setPropertyValue("anomaly_method", "PerRecords")
node.setPropertyValue("percent_records", 95)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("peer_group_num_auto", True)
node.setPropertyValue("min_num_peer_groups", 3)
node.setPropertyValue("max_num_peer_groups", 10)
```

Tableau 99. propriétés de anomalydetectionnode

Propriétés de **anomalydetectionnode**	Valeurs	Description de la propriété
inputs	*[champ1 ... champN]*	Les modèles Détection des anomalies filtrent les enregistrements en fonction des champs d'entrée spécifiés. Ils n'utilisent pas de champ cible. Les champs de pondération et de fréquence également ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	Indique la méthode utilisée pour déterminer la valeur de césure de signalement des enregistrements jugés comme irréguliers.
index_level	*nombre*	Indique la valeur de césure limite pour le signalement des anomalies.
percent_records	*nombre*	Définit le seuil de signalement des enregistrements, basé sur le pourcentage d'enregistrements dans les données d'apprentissage.
num_records	*nombre*	Définit le seuil de signalement des enregistrements, basé sur le nombre d'enregistrements dans les données d'apprentissage.
num_fields	*entier*	Nombre de champs à signaler pour chaque enregistrement irrégulier.
impute_missing_values	*flag*	
adjustment_coeff	*nombre*	Valeur utilisée pour équilibrer la pondération relative attribuée aux champs continus et aux champs catégoriels pour les calculs de distance.
peer_group_num_auto	*flag*	Calcule automatiquement le nombre de groupes d'homologues.
min_num_peer_groups	*entier*	Indique le nombre minimal de groupes d'homologues utilisés quand la propriété peer_group_num_auto est définie sur True (vrai).
max_num_per_groups	*entier*	Indique le nombre maximal de groupes d'homologues.

Tableau 99. propriétés de anomalydetectionnode (suite)

Propriétés de anomalydetectionnode	Valeurs	Description de la propriété
num_peer_groups	*entier*	Indique le nombre de groupes d'homologues utilisés quand la propriété peer_group_num_auto est définie sur False (faux).
noise_level	*nombre*	Détermine le mode de traitement des valeurs éloignées au cours de la classification non supervisée. Indiquez une valeur comprise entre 0 et 0,5.
noise_ratio	*nombre*	Indique la quantité de mémoire allouée au composant qui doit être utilisée pour la mise en mémoire tampon du bruit. Indiquez une valeur comprise entre 0 et 0,5.

Propriétés de apriorinode

Le noeud Apriori extrait des données un ensemble de règles et retient les règles contenant la plus grande quantité d'informations. Le noeud Apriori fournit cinq méthodes de sélection de règles et utilise un modèle d'indexation sophistiqué pour traiter efficacement les volumes de données importants. Pour les problèmes importants, l'apprentissage du noeud Apriori est généralement plus rapide ; il n'existe aucune limite quant au nombre de règles pouvant être conservées et il peut prendre en charge des règles faisant l'objet de 32 pré-conditions. Le noeud Apriori exige que les champs d'entrée et de sortie soient tous catégoriels, mais fournit de meilleures performances car il est optimisé de ce type de données.

Exemple

```
node = stream.create("apriori", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("partition", "Test")
# For non-transactional
node.setPropertyValue("use_transactional_data", False)
node.setPropertyValue("consequents", ["Age"])
node.setPropertyValue("antecedents", ["BP", "Cholesterol", "Drug"])
# For transactional
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("content_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Apriori_bp_choles_drug")
node.setPropertyValue("min_supp", 7.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_antecedents", 7)
node.setPropertyValue("true_flags", False)
node.setPropertyValue("optimize", "Memory")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("evaluation", "ConfidenceRatio")
node.setPropertyValue("lower_bound", 7)
```

Tableau 100. propriétés de apriorinode

Propriétés de apriorinode	Valeurs	Description de la propriété
consequents	*zone*	Les modèles Apriori utilisent les Conséquences et les Antécédents à la place des champs cible et entrée standard. Les champs de pondération et de fréquence ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
antecedents	[*champ1 ... champN*]	
min_supp	*nombre*	
min_conf	*nombre*	
max_antecedents	*nombre*	
true_flags	*flag*	
optimize	Vitesse Memory	
use_transactional_data	*flag*	
contiguous	*flag*	
id_field	*chaîne*	
content_field	*chaîne*	
mode	Simple Expert	
évaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*nombre*	
optimize	Vitesse Memory	Permet d'indiquer si la création du modèle doit être optimisée en vitesse ou en mémoire.

Propriétés associationrulesnode

Le noeud Règles d'association est similaire au noeud Apriori ; toutefois, à la différence de celui-ci, il peut traiter les données de liste. Par ailleurs, le noeud Règles d'association peut être utilisé avec IBM SPSS Analytic Server pour traiter des données volumineuses et exploiter un traitement parallèle, lequel est plus rapide.

Tableau 101. Propriétés associationrulesnode

Propriétés associationrulesnode	Type de données	Description de la propriété
prévisions	*zone*	Les champs de cette liste peuvent uniquement apparaître en tant que prédicteur d'une règle.
conditions	*[champ1...champN]*	Les champs de cette liste peuvent uniquement apparaître en tant que condition d'une règle.
max_rule_conditions	*entier*	Nombre maximal de conditions pouvant être incluses dans une même règle. Minimum 1, maximum 9.

Tableau 101. Propriétés associationrulesnode (suite)

Propriétés associationrulesnode	Type de données	Description de la propriété
max_rule_predictions	*entier*	Nombre maximal de prévisions pouvant être incluses dans une même règle. Minimum 1, maximum 5.
max_num_rules	*entier*	Nombre maximal de règles pouvant être prises en compte dans la construction de règles. Minimum 1, maximum 10000.
rule_criterion_top_n	Confidence Rulesupport Lift Conditionsupport Deployability	Critère de règle qui détermine la valeur selon laquelle les "N" premières règles sont sélectionnées dans le modèle.
true_flags	*Booléen*	Si vous indiquez *Y*, seules les valeurs true pour les champs indicateur sont prises en compte lors de la génération de règle.
rule_criterion	*Booléen*	Si vous indiquez *Y*, les valeurs de critère de règle sont utilisées pour exclure des règles lors de la génération de modèle.
min_confidence	*nombre*	0.1 à 100 - valeur de pourcentage du niveau de confiance minimum requis pour une règle générée par le modèle. Si le modèle génère une règle avec un niveau de confiance inférieur à la valeur spécifiée ici, la règle est ignorée.
min_rule_support	*nombre*	0.1 à 100 - valeur de pourcentage de niveau de prise en charge de la règle minimum requis pour une règle générée par le modèle. Si le modèle génère une règle avec un niveau de prise en charge de la règle inférieur à la valeur spécifiée, la règle est ignorée.
min_condition_support	*nombre*	0.1 à 100 - valeur de pourcentage de niveau de prise en charge de la condition minimum requis pour une règle générée par le modèle. Si le modèle génère une règle avec un niveau de prise en charge de la condition inférieur à la valeur spécifiée, la règle est ignorée.
min_lift	*entier*	1 à 10 - représente le lift minimum requis pour une règle générée par le modèle. Si le modèle génère une règle avec un niveau de lift inférieur à la valeur spécifiée, la règle est ignorée.
exclude_rules	*Booléen*	Permet de sélectionner une liste de champs associés que le modèle ne doit pas utiliser pour la création. Exemple : set :gsarsnode.exclude_rules = [{[champ1,champ2, champ3]},{[champ4, champ5]}] - où chaque liste de champs séparée par {} correspond à une ligne dans la table.
num_bins	*entier*	Définit le nombre de casiers automatiques dans lesquels les champs continus sont regroupés. Minimum 2, maximum 10.

Tableau 101. Propriétés associationrulesnode (suite)

Propriétés associationrulesnode	Type de données	Description de la propriété
max_list_length	*entier*	S'applique aux champs de liste pour lesquels la longueur maximale n'est pas connue. Les éléments de la liste dans la limite du nombre spécifié ici sont inclus dans la génération du modèle ; les autres éléments sont ignorés. Minimum 1, maximum 100.
output_confidence	*Booléen*	
output_rule_support	*Booléen*	
output_lift	*Booléen*	
output_condition_support	*Booléen*	
output_deployability	*Booléen*	
rules_to_display	upto toutes	Nombre maximal de règles à afficher dans les tables de sortie.
display_upto	*entier*	Si upto est défini dans rules_to_display, stipule le nombre de règles à afficher dans les tables de sortie. Minimum 1.
field_transformations	*Booléen*	
records_summary	*Booléen*	
rule_statistics	*Booléen*	
most_frequent_values	*Booléen*	
most_frequent_fields	*Booléen*	
word_cloud	*Booléen*	
word_cloud_sort	Confidence Rulesupport Lift Conditionsupport Deployability	
word_cloud_display	*entier*	Minimum 1, maximum 20
max_predictions	*entier*	Nombre maximal de règles pouvant être appliquées à chaque entrée dans le score.
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	Sélectionnez la mesure utilisée pour déterminer la puissance des règles.
allow_repeats	*Booléen*	Détermine si des règles avec la même prévision sont incluses dans le score.
check_input	NoPredictions Predictions NoCheck	

Propriétés autoclassifiernode

Le noeud Discriminant automatique crée et compare les résultats binaires de plusieurs modèles différents (oui ou non, avec ou sans attrition, etc.), ce qui vous permet de choisir la meilleure approche pour une analyse donnée. Plusieurs algorithmes de modélisation sont pris en charge. Vous pouvez alors sélectionner les méthodes que vous souhaitez utiliser, les options spécifiques pour chacune d'elles et le critère de comparaison des résultats. Le noeud génère un ensemble de modèles basé sur les options spécifiées et classe les meilleurs candidats en fonction des critères indiqués.

Exemple

```
node = stream.create("autoclassifier", "My node")
node.setPropertyValue("ranking_measure", "Accuracy")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_accuracy_limit", True)
node.setPropertyValue("accuracy_limit", 0.9)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("use_costs", True)
node.setPropertyValue("svm", False)
```

Tableau 102. propriétés autoclassifiernode.

Propriétés autoclassifiernode	Valeurs	Description de la propriété
target	*champ*	Pour les cibles indicateur, le noeud Discriminant automatique requiert un seul champ cible et un ou plusieurs champs d'entrée. Il est également possible de spécifier des champs de pondération et de fréquence. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
ranking_measure	Accuracy Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	*entier*	Nombre de modèles à inclure dans le nugget de modèle. Indiquez un entier compris entre 1 et 100.
calculate_variable_importance	*indicateur*	
enable_accuracy_limit	*indicateur*	
accuracy_limit	*entier*	Entier compris entre 0 et 100.
enable_area_under_curve_limit	*indicateur*	
area_under_curve_limit	*nombre*	Nombre réel compris entre 0,0 et 1,0.
enable_profit_limit	*indicateur*	
profit_limit	*nombre*	Entier supérieur à 0.
enable_lift_limit	*indicateur*	
lift_limit	*nombre*	Nombre réel supérieur à 1.0.
enable_number_of_variables_limit	*indicateur*	

Tableau 102. propriétés autoclassifiernode (suite).

Propriétés autoclassifiernode	Valeurs	Description de la propriété
number_of_variables_limit	*nombre*	Entier supérieur à 0.
use_fixed_cost	*indicateur*	
fixed_cost	*nombre*	Nombre réel supérieur à 0.0.
variable_cost	*champ*	
use_fixed_revenue	*indicateur*	
fixed_revenue	*nombre*	Nombre réel supérieur à 0.0.
variable_revenue	*champ*	
use_fixed_weight	*indicateur*	
fixed_weight	*nombre*	Nombre réel supérieur à 0,0.
variable_weight	*champ*	
lift_percentile	*nombre*	Entier compris entre 0 et 100.
enable_model_build_time_limit	*indicateur*	
model_build_time_limit	*nombre*	Entier défini sur le nombre de minutes devant être utilisé pour limiter le temps nécessaire à la création de chaque modèle.
enable_stop_after_time_limit	*indicateur*	
stop_after_time_limit	*nombre*	Nombre réel défini sur le nombre d'heures servant à limiter le temps global passé pour une exécution de Discriminant automatique.
enable_stop_after_valid_model_produced	*indicateur*	
use_costs	*indicateur*	
<algorithm>	*indicateur*	Active ou désactive l'utilisation d'un algorithme particulier.
<algorithm>.<property>	*chaîne*	Définit une valeur de propriété pour un algorithme spécifique. Pour plus d'informations, voir la rubrique «Définition des propriétés de l'algorithme».

Définition des propriétés de l'algorithme

Pour les noeuds Discriminant automatique, Numérisation automatique et Cluster automatique, les propriétés d'algorithmes spécifiques utilisées par le noeud peuvent être définies à l'aide de la forme générale :

```
autonode.setKeyedPropertyValue(<algorithme>, <propriété>, <valeur>)
```

Par exemple :

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

Les noms d'algorithme pour le noeud Discriminant automatique sont `cart`, `chaid`, `quest`, `c50`, `logreg`, `decisionlist`, `bayesnet`, `discriminant`, `svm` et `knn`.

Les noms d'algorithme pour le noeud Numérisation automatique sont `cart`, `chaid`, `neuralnetwork`, `genlin`, `svm`, `regression`, `linear` et `knn`.

Les noms d'algorithme pour le noeud Cluster automatique sont `twostep`, `k moyenne` et `kohonen`.

Les noms de propriété sont standard (reportez-vous aux informations relatives à chaque noeud d'algorithme).

Les propriétés d'algorithme qui contiennent des points ou d'autres signes de ponctuation doivent être placés entre guillemets simples (' '), par exemple :

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

Plusieurs valeurs peuvent également être affectées à une propriété, par exemple :

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up", "Down"])
```

Pour activer ou désactiver l'utilisation d'un algorithme particulier :

```
node.setPropertyValue("chaid", True)
```

Remarque : Lorsque certaines options d'algorithme ne sont pas disponibles dans le noeud Discriminant automatique ou qu'une seule valeur peut être indiquée plutôt qu'un intervalle de valeurs, les mêmes limites s'appliquent à la génération de scripts que lorsque l'on accède au noeud de la manière standard.

Propriétés du noeud de cluster automatique

Le noeud Cluster automatique évalue et compare les modèles de classification identifiant des groupes d'enregistrements ayant des caractéristiques similaires. Le noeud fonctionne de la même manière que les autres noeuds de modélisation automatiques, vous permettant de tester plusieurs combinaisons d'options en une seule modélisation. Les modèles peuvent être comparés à l'aide de mesures de bases permettant d'essayer de filtrer et de classer l'utilité des modèles de classification et de fournir une mesure en fonction de l'importance de champs particuliers.

Exemple

```
node = stream.create("autocluster", "My node")
node.setPropertyValue("ranking_measure", "Silhouette")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_silhouette_limit", True)
node.setPropertyValue("silhouette_limit", 5)
```

Tableau 103. Propriétés du noeud de cluster automatique

Propriétés de **autoclusternode**	Valeurs	Description de la propriété
`évaluation`	*zone*	**Remarque :** Noeud de cluster automatique uniquement : Identifie le champ pour lequel une valeur d'importance sera calculée. Il peut aussi être utilisé pour identifier la façon dont un cluster différencie la valeur de ce champ et, par conséquent, la façon dont le modèle pourra prédire ce champ.

`ranking_measure`	`Silhouette`
	`Num_clusters`
	`Size_smallest_cluster`
	`Size_largest_cluster`
	`Smallest_to_largest`
	`Importance`	

`ranking_dataset`	`Training` `Test`	

Tableau 103. Propriétés du noeud de cluster automatique (suite)

Propriétés de autoclusternode	Valeurs	Description de la propriété
summary_limit	*entier*	Nombre de modèles devant figurer dans le rapport. Indiquez un entier compris entre 1 et 100.
enable_silhouette_limit	*flag*	
silhouette_limit	*entier*	Entier compris entre 0 et 100.
enable_number_less_limit	*flag*	
number_less_limit	*nombre*	Nombre réel compris entre 0,0 et 1,0.
enable_number_greater_limit	*flag*	
number_greater_limit	*nombre*	Entier supérieur à 0.
enable_smallest_cluster_limit	*flag*	
smallest_cluster_units	Pourcentage Comptages	
smallest_cluster_limit_percentage	*nombre*	
smallest_cluster_limit_count	*entier*	Entier supérieur à 0.
enable_largest_cluster_limit	*flag*	
largest_cluster_units	Pourcentage Comptages	
largest_cluster_limit_percentage	*nombre*	
largest_cluster_limit_count	*entier*	
enable_smallest_largest_limit	*flag*	
smallest_largest_limit	*nombre*	
enable_importance_limit	*flag*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*nombre*	Entier compris entre 0 et 100.
importance_limit_less_than	*nombre*	Entier compris entre 0 et 100.
<algorithm>	*flag*	Active ou désactive l'utilisation d'un algorithme particulier.
<algorithm>.<property>	*chaîne*	Définit une valeur de propriété pour un algorithme spécifique. Pour plus d'informations, voir la rubrique «Définition des propriétés de l'algorithme», à la page 168.

Propriétés de autonumericnode

Le noeud Numérisation automatique évalue et compare des modèles pour des résultats d'intervalle numérique continus par le biais de différentes méthodes. Le noeud fonctionne de la même manière que le noeud Discriminant automatique, vous permettant ainsi de choisir les algorithmes à utiliser et à tester avec différentes combinaisons d'options en un seul passage de modélisation. Les algorithmes pris en charge comprennent les réseaux de neurones, l'arborescence C&R Tree, CHAID, la régression linéaire, la régression linéaire généralisée et support vector machines (SVM). Les modèles peuvent être comparés selon la corrélation, l'erreur relative ou le nombre de variables utilisées.

Exemple

```
node = stream.create("autonumeric", "My node")
node.setPropertyValue("ranking_measure", "Correlation")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_correlation_limit", True)
node.setPropertyValue("correlation_limit", 0.8)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("neuralnetwork", True)
node.setPropertyValue("chaid", False)
```

Tableau 104. propriétés de autonumericnode

Propriétés de autonumericnode	Valeurs	Description de la propriété
custom_fields	*flag*	Si cette valeur est définie sur True (vrai), les paramètres de champ personnalisés seront utilisés à la place des paramètres du noeud type.
target	*zone*	Le noeud Numérisation automatique requiert un seul champ cible et un ou plusieurs champs d'entrée. Il est également possible de spécifier des champs de pondération et de fréquence. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
inputs	*[champ1 ... champ2]*	
partition	*zone*	
use_frequency	*flag*	
frequency_field	*zone*	
use_weight	*flag*	
weight_field	*zone*	
use_partitioned_data	*flag*	Si un champ de partition est défini, seules les données d'apprentissage sont utilisées pour la création du modèle.
ranking_measure	Correlation NumberOfFields	
ranking_dataset	Test Formation	
number_of_models	*entier*	Nombre de modèles à inclure dans le nugget de modèle. Indiquez un entier compris entre 1 et 100.
calculate_variable_importance	*flag*	
enable_correlation_limit	*flag*	
correlation_limit	*entier*	
enable_number_of_fields_limit	*flag*	
number_of_fields_limit	*entier*	
enable_relative_error_limit	*flag*	
relative_error_limit	*entier*	
enable_model_build_time_limit	*flag*	
model_build_time_limit	*entier*	
enable_stop_after_time_limit	*flag*	

Tableau 104. propriétés de autonumericnode (suite)

Propriétés de autonumericnode	Valeurs	Description de la propriété
stop_after_time_limit	*entier*	
stop_if_valid_model	*flag*	
<algorithm>	*flag*	Active ou désactive l'utilisation d'un algorithme particulier.
<algorithm>.<property>	*chaîne*	Définit une valeur de propriété pour un algorithme spécifique. Pour plus d'informations, voir la rubrique «Définition des propriétés de l'algorithme», à la page 168.

Propriétés de bayesnetnode

Le noeud Réseau Bayésien permet de créer un modèle de probabilité en combinant les preuves observées et enregistrées avec les connaissances réelles pour établir la probabilité des occurences. Le noeud est axé sur le Tree Augmented Naïve Bayes (TAN) et sur les réseaux Couverture de Markov qui servent principalement à la classification.

Exemple

```
node = stream.create("bayesnet", "My node")
node.setPropertyValue("continue_training_existing_model", True)
node.setPropertyValue("structure_type", "MarkovBlanket")
node.setPropertyValue("use_feature_selection", True)
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("independence", "Pearson")
```

Tableau 105. Propriétés de bayesnetnode

Propriétés de bayesnetnode	Valeurs	Description de la propriété
inputs	*[champ1 ... champN]*	Les modèles de réseau Bayésien utilisent un seul champ cible et un ou plusieurs champs d'entrée. Les champs continus sont automatiquement mis en intervalles. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
continue_training_existing_model	*flag*	
structure_type	TAN MarkovBlanket	Sélectionne la structure à utiliser lors de la création du réseau Bayésien.
use_feature_selection	*flag*	
parameter_learning_method	Likelihood Bayes	Spécifie la méthode utilisée pour estimer les tableaux de probabilités conditionnelles entre les noeuds où les valeurs des parent sont connues.
mode	Expert Simple	
missing_values	*flag*	
all_probabilities	*flag*	

Tableau 105. Propriétés de bayesnetnode (suite)

Propriétés de bayesnetnode	Valeurs	Description de la propriété
independence	Likelihood Pearson	Spécifie la méthode utilisée pour déterminer si des observations par paire sur deux variables sont indépendantes l'une de l'autre.
significance_level	*nombre*	Spécifie la valeur de césure permettant de déterminer l'indépendance.
maximal_conditioning_set	*nombre*	Définit le nombre maximum de variables de conditionnement à utiliser pour les essais d'indépendance.
inputs_always_selected	*[champ1 ... champN]*	Spécifie les champs du jeu de données qui doivent toujours être utilisés lors de la création du réseau Bayésien. **Remarque :** Le champ cible est toujours sélectionné.
maximum_number_inputs	*nombre*	Spécifie le nombre maximum de champs d'entrée à utiliser pour créer le réseau Bayésien.
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés de buildr

Le noeud de création R vous permet d'entrer un script R personnalisé pour procéder à la création de modèle et au scoring de modèle déployés dans IBM SPSS Modeler.

Exemple

```
node = stream.create("buildr", "My node")
node.setPropertyValue("score_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
var1<-c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tableau 106. Propriétés de buildr.

Propriétés de buildr	Valeurs	Description de la propriété
build_syntax	*chaîne*	Syntaxe de script R pour la création de modèle.
score_syntax	*chaîne*	Syntaxe de script R pour le scoring de modèle.
convert_flags	StringsAndDoubles LogicalValues	Option permettant de convertir des champs indicateurs.

Tableau 106. Propriétés de buildr (suite).

Propriétés de buildr	Valeurs	Description de la propriété
convert_datetime	*flag*	Option permettant de convertir des variables au format de date ou date-heure en format de date/heure R.
convert_datetime_class	POSIXct POSIXlt	Options permettant d'indiquer dans quel format sont converties les variables au format de date ou date-heure.
convert_missing	*indicateur*	Option permettant de convertir des valeurs manquantes en valeur R NA.
output_html	*indicateur*	Option permettant d'afficher les graphiques sur un onglet du nugget de modèle R.
output_text	*flag*	Option permettant d'écrire la sortie texte de console R sur un onglet du nugget de modèle R.

Propriétés de c50node

Le noeud C5.0 crée un arbre décision ou un ensemble de règles. Le fonctionnement de ce modèle repose sur un découpage de l'échantillon basé sur le champ qui fournit le gain d'informations le plus important à chaque niveau. Le champ cible doit être catégoriel. Les divisions multiples en plus de deux sous-groupes sont autorisées.

Exemple

```
node = stream.create("c50", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "C5_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("output_type", "DecisionTree")
node.setPropertyValue("use_xval", True)
node.setPropertyValue("xval_num_folds", 3)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("favor", "Generality")
node.setPropertyValue("min_child_records", 3)
# "Costs" tab
node.setPropertyValue("use_costs", True)
node.setPropertyValue("costs", [["drugA", "drugX", 2]])
```

Tableau 107. propriétés de c50node

Propriétés de c50node	Valeurs	Description de la propriété
target	*zone*	Les modèles C50 utilisent un seul champ cible et un ou plusieurs champs d'entrée. Un champ poids peut aussi être spécifié. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
output_type	DecisionTree RuleSet	
group_symbolics	*flag*	
use_boost	*flag*	
boost_num_trials	*nombre*	

Tableau 107. propriétés de c50node (suite)

Propriétés de c50node	Valeurs	Description de la propriété
use_xval	*flag*	
xval_num_folds	*nombre*	
mode	Simple Expert	
favor	Accuracy Généralité	Privilégiez l'exactitude ou la généralité.
expected_noise	*nombre*	
min_child_records	*nombre*	
pruning_severity	*nombre*	
use_costs	*flag*	
coûts	*structurées*	Il s'agit d'une propriété structurée.
use_winnowing	*flag*	
use_global_pruning	*flag*	Activé (True(vrai)) par défaut.
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés de carmanode

Le modèle CARMA extrait un ensemble de règles des données sans que vous ayez à définir les champs d'entrée ou les champs cible. Au contraire du noeud Apriori le noeud CARMA offre des paramètres de création pour la prise en charge de la règle (à la fois pour les antécédents et les conséquences), plutôt qu'une simple prise en charge d'antécédents. Cela signifie que les règles générées peuvent être utilisées dans un grand nombre d'applications, par exemple pour rechercher une liste des produits ou des services (antécédents) dont la conséquence correspond à l'élément que vous souhaitez promouvoir à l'occasion de cette période de congés.

Exemple

```
node = stream.create("carma", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Drug"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "age_bp_drug")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 10.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_size", 5)
# Expert Options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 300)
```

```
node.setPropertyValue("vary_support", True)
node.setPropertyValue("estimated_transactions", 30)
node.setPropertyValue("rules_without_antecedents", True)
```

Tableau 108. propriétés de carmanode

Propriétés de carmanode	Valeurs	Description de la propriété
inputs	*[champ1 ... champN]*	Les modèles CARMA utilisent une liste de champs d'entrée, mais pas de cible. Les champs de pondération et de fréquence ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
id_field	*zone*	Champ utilisé en tant que champ d'ID pour la création du modèle.
contiguous	*flag*	Permet d'indiquer si les ID du champ ID sont contigus.
use_transactional_data	*flag*	
content_field	*zone*	
min_supp	*nombre (pourcentage)*	Est lié à la prise en charge de la règle plutôt qu'à celle des antécédents. La valeur par défaut est 20 %.
min_conf	*nombre (pourcentage)*	La valeur par défaut est 20 %.
max_size	*nombre*	La valeur par défaut est 10.
mode	Simple Expert	La valeur par défaut est Simple.
exclude_multiple	*flag*	Exclut les règles comportant plusieurs conséquences. La valeur par défaut est False.
use_pruning	*flag*	La valeur par défaut est False.
pruning_value	*nombre*	La valeur par défaut est 500.
vary_support	*flag*	
estimated_transactions	*entier*	
rules_without_antecedents	*flag*	

Propriétés de cartnode

Le noeud Arbre Classification et Régression, (C&R), génère un arbre décision qui vous permet de prévoir ou de classifier les observations futures. La méthode utilise la technique de partition récursive afin de diviser les données d'apprentissage en segments en réduisant l'index d'impureté à chaque étape, un noeud de l'arbre étant considéré comme “pur” si 100 % de ses observations appartiennent à une catégorie spécifique du champ cible. Les champs cible et les champs d'entrée peuvent être des champs d'intervalle numériques ou des champs catégoriels numériques (nominal,ordinal ou indicateur). Toutes les divisions sont binaires (deux sous-groupes uniquement).

Exemple

```
node = stream.createAt("cart", "My node", 200, 100)
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "BP", "Cholesterol"])
```

```
# "Build Options" tab, "Objective" panel
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", """Grow Node Index 0 Children 1 2
Grow Node Index 2 Children 3 4""")
# "Build Options" tab, "Basics" panel
node.setPropertyValue("prune_tree", False)
node.setPropertyValue("use_std_err_rule", True)
node.setPropertyValue("std_err_multiplier", 3.0)
node.setPropertyValue("max_surrogates", 7)
# "Build Options" tab, "Stopping Rules" panel
node.setPropertyValue("use_percentage", True)
node.setPropertyValue("min_parent_records_pc", 5)
node.setPropertyValue("min_child_records_pc", 3)
# "Build Options" tab, "Advanced" panel
node.setPropertyValue("min_impurity", 0.0003)
node.setPropertyValue("impurity_measure", "Twoing")
# "Model Options" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Cart_Drug")
```

Tableau 109. propriétés de cartnode

Propriétés de cartnode	Valeurs	Description de la propriété
target	*zone*	Les modèles d'arbre C&R requièrent un seul champ cible et un ou plusieurs champs d'entrée. Un champ de fréquence peut aussi être spécifié. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
continue_training_existing_model	*flag*	
objective	Standard Boosting Bagging psm	psm est destinée aux jeux de données très volumineux et nécessite une connexion à un serveur.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*chaîne*	Définissez les directives de développement de l'arbre. Vous pouvez placer les directives entre guillemets triples pour ne pas tenir compte des caractères d'insertion de ligne ou des guillemets doubles. Notez que les directives sont parfois très sensibles aux modifications, même mineures, apportées aux données ou aux options de modélisation. Elles peuvent ne pas s'étendre aux autres jeux de données.
use_max_depth	Default Personnalisé	
max_depth	*entier*	Profondeur maximale de l'arbre, comprise entre 0 et 1000. Utilisée uniquement si use_max_depth = Custom
prune_tree	*flag*	Elagage de l'arbre pour éviter le surajustement.

Tableau 109. propriétés de cartnode (suite)

Propriétés de cartnode	Valeurs	Description de la propriété
use_std_err	*flag*	Utiliser la différence maximale pour le risque (dans Erreurs standard).
std_err_multiplier	*nombre*	Différence maximale.
max_surrogates	*nombre*	Nombre maximal de substitutions.
use_percentage	*flag*	
min_parent_records_pc	*nombre*	
min_child_records_pc	*nombre*	
min_parent_records_abs	*nombre*	
min_child_records_abs	*nombre*	
use_costs	*flag*	
coûts	*structurées*	Propriétés structurées.
priors	Data Equal Personnalisé	
custom_priors	*structurées*	Propriétés structurées.
adjust_priors	*flag*	
trails	*nombre*	Nombre des modèles de composant pour le boosting ou le bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Règles de combinaison par défaut pour les cibles catégorielles.
range_ensemble_method	Mean Médiane	Règles de combinaison par défaut pour les cibles continues.
large_boost	*flag*	Appliquer le boosting aux jeux de données très volumineux.
min_impurity	*nombre*	
impurity_measure	Gini Twoing Ordonné	
train_pct	*nombre*	Ensemble de prévention de surajustement.
set_random_seed	*flag*	Dupliquer l'option des résultats.
seed	*nombre*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés de chaidnode

Le noeud CHAID génère des arbres décision à l'aide des statistiques du khi-deux pour identifier les séparations optimales. Contrairement aux noeuds C&R Tree et QUEST, CHAID peut générer des arbres non binaires, ce qui implique que certaines divisions possèdent plusieurs branches. Les champs cibles et les champs d'entrée peuvent être d'intervalle numérique (continu) ou catégoriques. La méthode Exhaustive CHAID correspond à une modification du CHAID qui examine plus en détail toutes les divisions possibles, mais dont les calculs sont plus longs.

Exemple

```
filenode = stream.createAt("variablefile", "My node", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("chaid", "My node", 200, 100)
stream.link(filenode, node)

node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "CHAID")
node.setPropertyValue("method", "Chaid")
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", "Test")
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("merge_alpha", 0.04)
node.setPropertyValue("chi_square", "Pearson")
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("epsilon", 0.003)
node.setPropertyValue("max_iterations", 75)
node.setPropertyValue("split_merged_categories", True)
node.setPropertyValue("bonferroni_adjustment", True)
```

Tableau 110. propriétés de chaidnode

Propriétés de chaidnode	Valeurs	Description de la propriété
target	*zone*	Les modèles CHAID requièrent un seul champ cible et un ou plusieurs champs d'entrée. Un champ de fréquence peut aussi être spécifié. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
continue_training_existing_model	*flag*	
objective	Standard Boosting Bagging psm	psm est destinée aux jeux de données très volumineux et nécessite une connexion à un serveur.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*chaîne*	
method	Chaid ExhaustiveChaid	

Tableau 110. propriétés de chaidnode (suite)

Propriétés de chaidnode	Valeurs	Description de la propriété
use_max_depth	Default Personnalisé	
max_depth	*entier*	Profondeur maximale de l'arbre, comprise entre 0 et 1000. Utilisée uniquement si use_max_depth = Custom
use_percentage	*flag*	
min_parent_records_pc	*nombre*	
min_child_records_pc	*nombre*	
min_parent_records_abs	*nombre*	
min_child_records_abs	*nombre*	
use_costs	*flag*	
coûts	*structurées*	Propriétés structurées.
trails	*nombre*	Nombre des modèles de composant pour le boosting ou le bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Règles de combinaison par défaut pour les cibles catégorielles.
range_ensemble_method	Mean Médiane	Règles de combinaison par défaut pour les cibles continues.
large_boost	*flag*	Appliquer le boosting aux jeux de données très volumineux.
split_alpha	*nombre*	Niveau de signification pour la division.
merge_alpha	*nombre*	Niveau de signification pour la fusion.
bonferroni_adjustment	*flag*	Ajuster les valeurs de signification à l'aide de la méthode Bonferroni.
split_merged_categories	*flag*	Autoriser une nouvelle division des catégories fusionnées.
chi_square	Pearson LR	Méthode utilisée pour calculer les statistiques du khi-carré : Pearson ou Likelihood Ratio
epsilon	*nombre*	Modification minimale dans les prévisions de fréquence de cellule..
max_iterations	*nombre*	Itérations maximales pour convergence.
set_random_seed	*entier*	
seed	*nombre*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*entier*	

Propriétés de coxregnode

Le noeud de régression de Cox vous permet de créer un modèle de survie pour les données de durée jusqu'à l'événement en présence d'enregistrements censurés. Ce modèle produit une fonction de survie qui prédit la probabilité que l'événement en question se soit produit à un moment (*t*) pour des valeurs données des variables d'entrée.

Exemple

```
node = stream.create("coxreg", "My node")
node.setPropertyValue("survival_time", "tenure")
node.setPropertyValue("method", "BackwardsStepwise")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("removal_criterion", "Conditional")
node.setPropertyValue("survival", True)
```

Tableau 111. propriétés de coxregnode

Propriétés de coxregnode	Valeurs	Description de la propriété
survival_time	*zone*	Les modèles de régression de Cox requièrent un champ unique contenant les durées de survie.
target	*zone*	Les modèles de régression de Cox requièrent un seul champ cible et un ou plusieurs champs d'entrée. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
method	Enter Stepwise BackwardsStepwise	
groups	*zone*	
model_type	MainEffects Personnalisé	
custom_terms	[*"BP*Sex" "BP*Age"*]	
mode	Expert Simple	
max_iterations	*nombre*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	

Tableau 111. propriétés de coxregnode (suite)

Propriétés de coxregnode	Valeurs	Description de la propriété
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
removal_criterion	LR Wald Conditionnel	
probability_entry	*nombre*	
probability_removal	*nombre*	
output_display	EachStep LastStep	
ci_enable	*flag*	
ci_value	90 95 99	
correlation	*flag*	
display_baseline	*flag*	
survival	*flag*	
hazard	*flag*	
log_minus_log	*flag*	
one_minus_survival	*flag*	
separate_line	*zone*	
value	*nombre* ou *chaîne*	Si aucune valeur n'est spécifiée pour un champ, l'option par défaut, « Moyenne » sera utilisée pour ce champ.

Propriétés de decisionlistnode

Le noeud Liste de décision identifie les sous-groupes, ou les segments, qui présentent une probabilité plus élevée ou plus faible d'un résultat binaire donné par rapport à la population globale. Vous pouvez, par exemple, rechercher les clients qui ont une faible probabilité d'attrition ou ceux qui ont une plus forte probabilité de répondre favorablement à une campagne. Vous pouvez incorporer vos connaissances métier dans le modèle en ajoutant vos propres segments personnalisés et en prévisualisant des modèles alternatifs côte à côte de façon à comparer les résultats. Les modèles Liste de décision se composent d'une liste de règles dans laquelle chaque règle présente une condition et un résultat. Les règles sont appliquées dans l'ordre et la première règle correspondante détermine le résultat.

Exemple

```
node = stream.create("decisionlist", "My node")
node.setPropertyValue("search_direction", "Down")
node.setPropertyValue("target_value", 1)
node.setPropertyValue("max_rules", 4)
node.setPropertyValue("min_group_size_pct", 15)
```

Tableau 112. Propriétés de decisionlistnode

Propriétés de decisionlistnode	Valeurs	Description de la propriété
target	*zone*	Les modèles Liste de décision utilisent un seul champ cible et un ou plusieurs champs d'entrée. Un champ de fréquence peut aussi être spécifié. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
model_output_type	Model InteractiveBuilder	
search_direction	Up Down	Est lié à la recherche de segments ; où Up est l'équivalent de Forte probabilité et Down est l'équivalent de Faible probabilité.
target_value	*chaîne*	Si cette propriété n'est pas définie, la valeur True (vrai) est sélectionnée pour les champs indicateurs.
max_rules	*entier*	Nombre maximal de segments, sauf le reste.
min_group_size	*entier*	Taille minimale de segment.
min_group_size_pct	*nombre*	Taille minimale de segment, en pourcentage.
confidence_level	*nombre*	Seuil minimal dont dispose un champ d'entrée pour améliorer (augmenter) la probabilité d'une réponse, pour que son ajout à une définition de segment soit utile.
max_segments_per_rule	*entier*	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	*nombre*	
max_models_per_cycle	*entier*	Largeur de recherche des listes.
max_rules_per_cycle	*entier*	Largeur de recherche des règles de segment.
segment_growth	*nombre*	
include_missing	*flag*	
final_results_only	*flag*	
reuse_fields	*flag*	Permet aux attributs (champs d'entrée qui apparaissent dans les règles) d'être réutilisés.
max_alternatives	*entier*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés de discriminantnode

L'analyse discriminante crée des hypothèses plus strictes que la régression logistique mais peut constituer une alternative ou un complément précieux à une analyse de régression logistique lorsque ces hypothèses sont réunies.

Exemple

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

Tableau 113. Propriétés de discriminantnode

Propriétés de discriminantnode	Valeurs	Description de la propriété
target	*zone*	Les modèles Discriminant requièrent un seul champ cible et un ou plusieurs champs d'entrée. Les champs de pondération et de fréquence ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
method	Enter Etape par étape	
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
moyennes	*flag*	Options Statistiques dans la boîte de dialogue Sortie avancée.
univariate_anovas	*flag*	
box_m	*flag*	
within_group_covariance	*flag*	
within_groups_correlation	*flag*	
separate_groups_covariance	*flag*	
total_covariance	*flag*	
fishers	*flag*	
unstandardized	*flag*	
casewise_results	*flag*	Options Classification dans la boîte de dialogue Sortie avancée.
limit_to_first	*nombre*	La valeur par défaut est 10.
summary_table	*flag*	
leave_one_classification	*flag*	
combined_groups	*flag*	
separate_groups_covariance	*flag*	Option Matrices, **Covariance par groupes distincts**
territorial_map	*flag*	

Tableau 113. Propriétés de discriminantnode (suite)

Propriétés de discriminantnode	Valeurs	Description de la propriété
combined_groups	*flag*	Option Tracé **Groupes regroupés**.
separate_groups	*flag*	Option Tracé **Groupes distincts**
summary_of_steps	*flag*	
F_pairwise	*flag*	
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*nombre*	
criteria	UseValue UseProbability	
F_value_entry	*nombre*	La valeur par défaut est 3,84.
F_value_removal	*nombre*	La valeur par défaut est 2,71.
probability_entry	*nombre*	La valeur par défaut est 0.05.
probability_removal	*nombre*	La valeur par défaut est 0.10.
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés de factornode

Le noeud ACP/Analyse factorielle propose des techniques de factorisation puissantes qui vous permettent de réduire la complexité de vos données. L'analyse en composantes principales (ACP) recherche les combinaisons linéaires des champs d'entrée qui permettent de capturer au mieux la variance dans l'ensemble de champs, où les composantes sont orthogonales (perpendiculaires) les unes par rapport aux autres. L'analyse factorielle a pour but d'identifier les facteurs sous-jacents qui expliquent la tendance des corrélations dans un ensemble de champs observés. Quelle que soit l'approche choisie, le but consiste à trouver un nombre limité de champs dérivés récapitulant les informations contenues dans l'ensemble de champs d'origine.

Exemple

```
node = stream.create("factor", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["BP", "Na", "K"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Factor_Age")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "GLS")
# Expert options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", True)
```

```
node.setPropertyValue("matrix", "Covariance")
node.setPropertyValue("max_iterations", 30)
node.setPropertyValue("extract_factors", "ByFactors")
node.setPropertyValue("min_eigenvalue", 3.0)
node.setPropertyValue("max_factor", 7)
node.setPropertyValue("sort_values", True)
node.setPropertyValue("hide_values", True)
node.setPropertyValue("hide_below", 0.7)
# "Rotation" section
node.setPropertyValue("rotation", "DirectOblimin")
node.setPropertyValue("delta", 0.3)
node.setPropertyValue("kappa", 7.0)
```

Tableau 114. propriétés de factornode

Propriétés de **factornode**	Valeurs	Description de la propriété
inputs	[*champ1 ... champN*]	Les modèles APC/Facteur utilisent une liste de champs d'entrée, mais pas de cible. Les champs de pondération et de fréquence ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Expert	
max_iterations	*nombre*	
complete_records	*flag*	
matrix	Correlation Covariance :	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*nombre*	
max_factor	*nombre*	
rotation	None Varimax DirectOblimin Equamax Quartimax Promax	
delta	*nombre*	Si vous sélectionnez DirectOblimin comme type de données de rotation, vous pouvez indiquer une valeur de delta. Si vous ne le faites pas, la valeur par défaut pour delta est utilisée.

Tableau 114. propriétés de factornode (suite)

Propriétés de factornode	Valeurs	Description de la propriété
Kappa	*nombre*	Si vous sélectionnez Promax comme type de données de rotation, vous pouvez indiquer une valeur de kappa. Si vous ne le faites pas, la valeur par défaut pour kappa est utilisée.
sort_values	*flag*	
hide_values	*flag*	
hide_below	*nombre*	

Propriétés de featureselectionnode

Le noeud Sélection de fonction filtre les champs d'entrée en vue de leur suppression, en fonction d'un ensemble de critères donné (tel que le pourcentage de valeurs manquantes) ; il classe ensuite les entrées restantes selon leur importance par rapport à la cible indiquée. Si l'on prend, par exemple, un jeu de données comportant des centaines d'entrées potentielles, quelles sont celles susceptibles d'être les plus utiles dans la modélisation des résultats de patients ?

Exemple

```
node = stream.create("featureselection", "My node")
node.setPropertyValue("screen_single_category", True)
node.setPropertyValue("max_single_category", 95)
node.setPropertyValue("screen_missing_values", True)
node.setPropertyValue("max_missing_values", 80)
node.setPropertyValue("criteria", "Likelihood")
node.setPropertyValue("unimportant_below", 0.8)
node.setPropertyValue("important_above", 0.9)
node.setPropertyValue("important_label", "Check Me Out!")
node.setPropertyValue("selection_mode", "TopN")
node.setPropertyValue("top_n", 15)
```

Pour un exemple plus détaillé de création et d'application d'un modèle Sélection de fonction, voir «Exemple de script autonome : Génération d'un modèle Sélection de fonction», à la page 4.

Tableau 115. propriétés de featureselectionnode

Propriétés de featureselectionnode	Valeurs	Description de la propriété
target	*zone*	Les modèles Sélection de fonction classent les prédicteurs par rapport à la cible spécifiée. Les champs de pondération et de fréquence ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
screen_single_category	*flag*	Si cette propriété est définie sur True (vrai), elle filtre les champs qui comportent, par rapport au nombre total d'enregistrements, trop d'enregistrements relatifs à une même catégorie.

Tableau 115. propriétés de featureselectionnode (suite)

Propriétés de **featureselectionnode**	Valeurs	Description de la propriété
max_single_category	*nombre*	Indique le seuil utilisé quand la propriétéscreen_single_category est définie sur True (vrai).
screen_missing_values	*flag*	Si cette propriété est définie sur True (vrai), elle filtre les champs qui comportent un nombre trop important de valeurs manquantes, exprimé en pourcentage du nombre total d'enregistrements.
max_missing_values	*nombre*	
screen_num_categories	*flag*	Si cette propriété est définie sur True (vrai), elle filtre les champs comportant trop de catégories par rapport au nombre total d'enregistrements.
max_num_categories	*nombre*	
screen_std_dev	*flag*	Si cette propriété est définie sur True (vrai), elle filtre les champs dont l'écart-type est inférieur ou égal à la valeur minimale indiquée.
min_std_dev	*nombre*	
screen_coeff_of_var	*flag*	Si cette propriété est définie sur True (vrai), elle filtre les champs dont le coefficient de la variance est inférieur ou égal à la valeur minimale spécifiée.
min_coeff_of_var	*nombre*	
criteria	Pearson Likelihood CramersV Lambda	Lors du classement des prédicteurs indépendants par rapport à une cible catégorielle, indique la mesure sur laquelle est basée la valeur d'importance.
unimportant_below	*nombre*	Indique les valeurs *p* du seuil, utilisées pour classer les variables comme étant importantes, marginales ou non significatives. Accepte des valeurs de 0,0 à 1,0.
important_above	*nombre*	Accepte des valeurs de 0,0 à 1,0.
unimportant_label	*chaîne*	Indique le libellé du classement non significatif.
marginal_label	*chaîne*	
important_label	*chaîne*	
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*flag*	Quand la propriété selection_mode est définie sur ImportanceLevel, indique si les champs importants doivent être sélectionnés.
select_marginal	*flag*	Quand la propriété selection_mode est définie sur ImportanceLevel, indique si les champs marginaux doivent être sélectionnés.

Tableau 115. propriétés de featureselectionnode (suite)

Propriétés de featureselectionnode	Valeurs	Description de la propriété
select_unimportant	*flag*	Quand la propriété selection_mode est définie sur ImportanceLevel, indique si les champs non significatifs doivent être sélectionnés.
importance_value	*nombre*	Quand la propriété selection_mode est définie sur ImportanceValue, indique la valeur de césure à utiliser. Accepte des valeurs de 0 à 100.
top_n	*entier*	Quand la propriété selection_mode est définie sur TopN, indique la valeur de césure à utiliser. Accepte des valeurs de 0 à 1000.

Propriétés de genlinnode

La procédure Modèles linéaires généralisés développe le modèle linéaire général de sorte que la variable dépendante soit linéairement reliée aux facteurs et covariables via une fonction de lien précise. En outre, le modèle permet à la variable dépendante de suivre une distribution non normale. Il couvre les fonctionnalités d'un grand nombre de modèles statistiques, notamment le modèle de régression linéaire, le modèle de régression logistique, le modèle log-linéaire pour les données d'effectif et le modèle de survie avec censure par intervalle.

Exemple

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

Tableau 116. Propriétés de genlinnode

Propriétés de genlinnode	Valeurs	Description de la propriété
target	*zone*	Les modèles linéaires généralisés requièrent un seul champ cible, qui doit être un champ nominal ou indicateur, et un ou plusieurs champs d'entrée. Un champ poids peut aussi être spécifié. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
use_weight	*flag*	
weight_field	*zone*	Le champ peut uniquement être de type continu.
target_represents_trials	*flag*	
trials_type	Variable FixedValue	
trials_field	*zone*	Le champ est de type Continu, Indicateur ou Ordinal.
trials_number	*nombre*	La valeur par défaut est 10.
model_type	MainEffects MainAndAllTwoWayEffects	

Tableau 116. Propriétés de genlinnode (suite)

Propriétés de genlinnode	Valeurs	Description de la propriété
offset_type	Variable FixedValue	
offset_field	*zone*	Le champ peut uniquement être de type continu.
offset_value	*nombre*	Doit être un nombre réel.
base_category	Last First	
include_intercept	*flag*	
mode	Simple Expert	
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: Gaussienne inverse. NEGBIN : Binomiale négative.
negbin_para_type	Specify Estimate	
negbin_parameter	*nombre*	La valeur par défaut est 1. Doit contenir un nombre réel non négatif.
tweedie_parameter	*nombre*	
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG : Log-log complémentaire. LOGC : complément log. NEGBIN : Binomiale négative. NLOGLOG : Log-log négatif. CUMCAUCHIT : Cauchit cumulé. CUMCLOGLOG : Log-log complémentaire cumulé. CUMLOGIT : Logit cumulé. CUMNLOGLOG : Log-log négatif cumulé. CUMPROBIT : Probit cumulé.
power	*nombre*	La valeur doit être un nombre réel autre que zéro.
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	*nombre*	La valeur par défaut est 1 ; seuls les entiers positifs sont autorisés.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	

Tableau 116. Propriétés de genlinnode (suite)

Propriétés de genlinnode	Valeurs	Description de la propriété
scale_value	*nombre*	La valeur par défaut est 1 ; elle doit être supérieure à 0.
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	*nombre*	La valeur par défaut est 100 ; entiers non négatifs uniquement.
max_step_halving	*nombre*	La valeur par défaut est 5 ; entiers positifs uniquement.
check_separation	*flag*	
start_iteration	*nombre*	La valeur par défaut est 20 ; seuls les entiers positifs sont autorisés.
estimates_change	*flag*	
estimates_change_min	*nombre*	La valeur par défaut est 1E-006 ; seuls les nombres positifs sont autorisés.
estimates_change_type	Absolute Relatif	
loglikelihood_change	*flag*	
loglikelihood_change_min	*nombre*	Seuls les nombres positifs sont autorisés.
loglikelihood_change_type	Absolute Relatif	
hessian_convergence	*flag*	
hessian_convergence_min	*nombre*	Seuls les nombres positifs sont autorisés.
hessian_convergence_type	Absolute Relatif	
case_summary	*flag*	
contrast_matrices	*flag*	
descriptive_statistics	*flag*	
estimable_functions	*flag*	
model_info	*flag*	
iteration_history	*flag*	
goodness_of_fit	*flag*	
print_interval	*nombre*	La valeur par défaut est 1 ; il doit s'agir d'un entier positif.
model_summary	*flag*	
lagrange_multiplier	*flag*	
parameter_estimates	*flag*	
include_exponential	*flag*	
covariance_estimates	*flag*	
correlation_estimates	*flag*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistiques	Wald LR	

Tableau 116. Propriétés de genlinnode (suite)

Propriétés de genlinnode	Valeurs	Description de la propriété
citype	Wald Profile	
tolerancelevel	*nombre*	La valeur par défaut est 0,0001.
confidence_interval	*nombre*	La valeur par défaut est 95.
loglikelihood_function	Complet Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending Descending DataOrder	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés glmmnode

Un modèle mixte linéaire généralisé (MMLG) élargit le modèle linéaire de sorte que la cible puisse avoir une distribution non normale, qu'elle soit liée linéairement aux facteurs et covariables via une fonction de lien spécifiée, et que les observations puissent être corrélées. Les modèles mixtes linéaires généralisés couvrent une large variété de modèles, depuis les modèles de régression linéaire simple aux modèles multi-niveaux complexes destinés aux données longitudinales non normales.

Tableau 117. propriétés glmmnode.

Propriétés glmmnode	Valeurs	Description de la propriété
residual_subject_spec	*structurée*	La combinaison des valeurs des champs catégoriels spécifiés qui définissent de manière unique les sujets dans le jeu de données
repeated_measures	*structurée*	Champs à utiliser pour identifier les observations répétées.
residual_group_spec	[*champ1 ... champN*]	Champs qui définissent les ensembles indépendants des paramètres de covariance d'effets répétés.
residual_covariance_type	Diagonale AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	Spécifie la structure de la covariance des résidus.

Tableau 117. propriétés glmmnode (suite).

Propriétés glmmnode	Valeurs	Description de la propriété
custom_target	*indicateur*	Indique s'il faut utiliser la cible définie dans le noeud en amont (false) ou la cible personnalisée spécifiée par target_field (true).
target_field	*champ*	Champ à utiliser comme cible si custom_target a la valeur true.
use_trials	*indicateur*	Indique si un champ supplémentaire ou une valeur spécifiant le nombre d'essais doit être utilisé lorsque la réponse cible est un nombre d'événements se produisant dans un ensemble d'essais. La valeur par défaut est false.
use_field_or_value	Field Valeur	Indique si un champ (par défaut) ou une valeur est utilisé pour spécifier le nombre d'essais.
trials_field	*champ*	Champ à utiliser pour spécifier le nombre d'essais.
trials_value	*entier*	Valeur à utiliser pour spécifier le nombre d'essais. Si spécifiée, la valeur minimum est 1.
use_custom_target_reference	*indicateur*	Indique si la catégorie de référence personnalisée doit être utilisée pour une cible catégorielle. La valeur par défaut est false.
target_reference_value	*chaîne*	Catégorie de référence à utiliser si use_custom_target_reference a la valeur true.
dist_link_combination	Nominal Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	Modèles standard pour la distribution des valeurs de la cible. Choisissez Custom pour spécifier une distribution à partir de la liste fournie par target_distribution.
target_distribution	Normal Binomial Multinomial Gamma Inverse NegativeBinomial Poisson	Distribution des valeurs de la cible quand dist_link_combination a la valeur Custom.

Tableau 117. propriétés glmmnode (suite).

Propriétés glmmnode	Valeurs	Description de la propriété
link_function_type	Identité LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	Fonction de lien pour associer les valeurs cibles aux prédicteurs. Si target_distribution a la valeur Binomial, vous pouvez utiliser n'importe laquelle des fonctions de lien listées. Si target_distribution a une valeur Multinomial, vous pouvez utiliser CLOGLOG, CAUCHIT, LOGIT, NLOGLOG ou PROBIT. Si target_distribution a une valeur autre que Binomial ou Multinomial, vous pouvez utiliser IDENTITY, LOG ou POWER.
link_function_param	*nombre*	Valeur de paramètre de fonction de lien à utiliser. S'applique uniquement si normal_link_function ou link_function_type a la valeur POWER.
use_predefined_inputs	*indicateur*	Indique si les champs d'effets fixes doivent être ceux définis en amont comme champs d'entrée (true) ou ceux de fixed_effects_list (false). La valeur par défaut est false.
fixed_effects_list	*structurée*	Si use_predefined_inputs est false, spécifie les champs d'entrée à utiliser comme champs d'effets fixes.
use_intercept	*indicateur*	Si true (par défaut), inclut la constante dans le modèle.
random_effects_list	*structurée*	Liste des champs à spécifier comme effets aléatoires.
regression_weight_field	*champ*	Champ à utiliser comme champ de pondération d'analyse.
use_offset	None offset_value offset_field	Indique comment le décalage est spécifié. La valeur None signifie qu'aucun décalage n'est spécifié.
offset_value	*nombre*	Valeur à utiliser pour le décalage si use_offset est défini sur offset_value.
offset_field	*champ*	Champ à utiliser pour la valeur de décalage si use_offset est défini sur offset_field.
target_category_order	Ascending Descending Data	Ordre de tri des cibles catégorielles. La valeur Data spécifie qu'il faut utiliser l'ordre de tri trouvé dans les données. La valeur par défaut est Ascending.
inputs_category_order	Ascending Descending Data	Ordre de tri pour les prédicteurs indépendants. La valeur Data spécifie qu'il faut utiliser l'ordre de tri trouvé dans les données. La valeur par défaut est Ascending.
max_iterations	*entier*	Nombre maximum d'itérations que l'algorithme effectuera. Un entier positif ; la valeur par défaut est 100.

Tableau 117. propriétés glmmnode (suite).

Propriétés glmmnode	Valeurs	Description de la propriété
confidence_level	*entier*	Niveau de confiance utilisé pour calculer les estimations d'intervalle des coefficients de modèle. Un entier positif ; le nombre maximum est 100, la valeur par défaut est 95.
degrees_of_freedom_method	Fixed Varied	Spécifie comment les degrés de liberté sont calculés pour le test de signification.
test_fixed_effects_coeffecients	Model Robust	Méthode de calcul de la matrice de covariance des estimations de paramètre.
use_p_converge	*flag*	Option pour la convergence de paramètres.
p_converge	*nombre*	Vide ou valeur positive.
p_converge_type	Absolute Relative	
use_l_converge	*flag*	Option pour la convergence du log de vraisemblance.
l_converge	*nombre*	Vide ou valeur positive.
l_converge_type	Absolute Relative	
use_h_converge	*flag*	Option pour la convergence hessienne.
h_converge	*nombre*	Vide ou valeur positive.
h_converge_type	Absolute Relative	
max_fisher_steps	*entier*	
singularity_tolerance	*nombre*	
use_model_name	*indicateur*	Indique s'il faut spécifier un nom personnalisé pour le modèle (true) ou utiliser le nom généré par le système (false). La valeur par défaut est false.
model_name	*chaîne*	Si use_model_name a la valeur true (vrai), indique le nom de modèle à utiliser.
confidence	onProbability onIncrease	Base de calcul de la valeur de confiance de scoring : probabilité prédite la plus élevée ou différence entre les probabilités prédites les plus élevées et les deuxièmes plus élevées.
score_category_probabilities	*indicateur*	Si true, produit des probabilités prédites pour les cibles catégorielles. La valeur par défaut est false.
max_categories	*entier*	Si score_category_probabilities a la valeur true, spécifie le nombre maximum de catégories à enregistrer.
score_propensity	*indicateur*	Si true, produit des scores de propension pour les champs cibles indicateurs qui indiquent la probabilité du résultat « true » pour le champ.
emeans	*structure*	Pour chaque champ catégoriel de la liste des effets fixes, spécifie s'il faut produire des moyennes marginales estimées.

Tableau 117. propriétés glmmnode (suite).

Propriétés glmmnode	Valeurs	Description de la propriété
covariance_list	*structure*	Pour chaque champ continu de la liste des effets fixes, spécifie s'il faut utiliser la moyenne ou une valeur personnalisée lors du calcul des moyennes marginales estimées.
mean_scale	Original Transformation	Spécifie s'il faut calculer les moyennes marginales estimées en fonction de l'échelle originale de la cible (par défaut) ou de la transformation de la fonction de lien.
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	Méthode d'ajustement à utiliser lors des tests d'hypothèse avec contrastes multiples.

Propriétés de kmeansnode

Le noeud k moyenne classe le jeu de données dans différents groupes (ou clusters). La méthode définit un nombre de clusters fixe, affecte à plusieurs reprises des enregistrements à des clusters et ajuste les centres de cluster, jusqu'à ce que le modèle ne puisse plus être amélioré. Au lieu de tenter de prédire un résultat, le modèle *k*-means utilise un processus connu sous le nom d'apprentissage non supervisé pour découvrir des tendances dans l'ensemble de champs d'entrée.

Exemple

```
node = stream.create("kmeans", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Cholesterol", "BP", "Drug", "Na", "K", "Age"])
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Kmeans_allinputs")
node.setPropertyValue("num_clusters", 9)
node.setPropertyValue("gen_distance", True)
node.setPropertyValue("cluster_label", "Number")
node.setPropertyValue("label_prefix", "Kmeans_")
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("stop_on", "Custom")
node.setPropertyValue("max_iterations", 10)
node.setPropertyValue("tolerance", 3.0)
node.setPropertyValue("encoding_value", 0.3)
```

Tableau 118. propriétés de kmeansnode

Propriétés de kmeansnode	Valeurs	Description de la propriété
inputs	[*champ1 ... champN*]	Les modèles k moyenne procèdent à une analyse des clusters sur un ensemble de champs d'entrée, mais n'utilisent pas de champ cible. Les champs de pondération et de fréquence ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.

Tableau 118. propriétés de kmeansnode (suite)

Propriétés de kmeansnode	Valeurs	Description de la propriété
num_clusters	*nombre*	
gen_distance	*flag*	
cluster_label	String Nombre	
label_prefix	*chaîne*	
mode	Simple Expert	
stop_on	Default Custom	
max_iterations	*nombre*	
tolerance	*nombre*	
encoding_value	*nombre*	
optimize	Vitesse Memory	Permet d'indiquer si la création du modèle doit être optimisée en vitesse ou en mémoire.

Propriétés knnnode

Le noeud *k*-Voisin le plus proche (KNN) associe une nouvelle observation à la catégorie ou à la valeur des objets *k* les plus proches dans l'espace du prédicteur, où *k* est un entier. Les observations semblables sont proches l'une de l'autre et les observations dissemblables sont éloignées l'une de l'autre.

Exemple

```
node = stream.create("knn", "My node")
# Objectives tab
node.setPropertyValue("objective", "Custom")
# Settings tab - Neighbors panel
node.setPropertyValue("automatic_k_selection", False)
node.setPropertyValue("fixed_k", 2)
node.setPropertyValue("weight_by_importance", True)
# Settings tab - Analyze panel
node.setPropertyValue("save_distances", True)
```

Tableau 119. propriétés knnnode

Propriétés de knnnode	Valeurs	Description de la propriété
analysis	PredictTarget IdentifyNeighbors	
objective	Equilibrer Vitesse Accuracy Personnalisé	
normalize_ranges	*flag*	
use_case_labels	*flag*	Cochez la case pour activer l'option suivante.
case_labels_field	*zone*	

Tableau 119. propriétés knnnode (suite)

Propriétés de knnnode	Valeurs	Description de la propriété
identify_focal_cases	*flag*	Cochez la case pour activer l'option suivante.
focal_cases_field	*zone*	
automatic_k_selection	*flag*	
fixed_k	*entier*	Activé uniquement si automatic_k_selectio est False
minimum_k	*entier*	Activé uniquement si automatic_k_selectio est True
maximum_k	*entier*	
distance_computation	Euclidean CityBlock	
weight_by_importance	*flag*	
range_predictions	Mean Médiane	
perform_feature_selection	*flag*	
forced_entry_inputs	[*champ1 ... champN*]	
stop_on_error_ratio	*flag*	
number_to_select	*entier*	
minimum_change	*nombre*	
validation_fold_assign_by_field	*flag*	
number_of_folds	*entier*	Activé uniquement si validation_fold_assign_by_field est False
set_random_seed	*flag*	
random_seed	*nombre*	
folds_field	*zone*	Activé uniquement si validation_fold_assign_by_field est True
all_probabilities	*flag*	
save_distances	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés de kohonennode

Le noeud Kohonen génère un type de réseau de neurones qui peut être utilisé pour classer les données en groupes distincts. Lorsque l'apprentissage du réseau est terminé, les enregistrements similaires doivent être regroupés dans la connexion de sortie, tandis que les enregistrements différents sont à l'opposé. Vous pouvez étudier le nombre d'observations capturées par chaque unité du nugget de modèle afin d'identifier les unités fortes. Vous pouvez ainsi vous faire une idée du nombre de clusters approprié.

Exemple

```
node = stream.create("kohonen", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Symbolic Cluster")
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("time", 1)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("width", 3)
node.setPropertyValue("length", 3)
node.setPropertyValue("decay_style", "Exponential")
node.setPropertyValue("phase1_neighborhood", 3)
node.setPropertyValue("phase1_eta", 0.5)
node.setPropertyValue("phase1_cycles", 10)
node.setPropertyValue("phase2_neighborhood", 1)
node.setPropertyValue("phase2_eta", 0.2)
node.setPropertyValue("phase2_cycles", 75)
```

Tableau 120. propriétés de kohonennode

Propriétés de kohonennode	**Valeurs**	**Description de la propriété**
inputs	[*champ1 ... champN*]	Les modèles Kohonen utilisent une liste de champs d'entrée, mais pas de cible. Les champs de fréquences et de pondération ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
continue	*flag*	
show_feedback	*flag*	
stop_on	Default Heure	
heure	*nombre*	
optimize	Vitesse Memory	Permet d'indiquer si la création du modèle doit être optimisée en vitesse ou en mémoire.
cluster_label	*flag*	
mode	Simple Expert	
width	*nombre*	
longueur	*nombre*	
decay_style	Linear Exponential	
phase1_neighborhood	*nombre*	
phase1_eta	*nombre*	
phase1_cycles	*nombre*	
phase2_neighborhood	*nombre*	
phase2_eta	*nombre*	
phase2_cycles	*nombre*	

Propriétés de linearnode

Les modèles de régression linéaire prédisent une cible continue en fonction de relations linéaires entre la cible et un ou plusieurs prédicteurs.

Exemple

```
node = stream.create("linear", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Model Selection panel
node.setPropertyValue("model_selection", "BestSubsets")
node.setPropertyValue("criteria_best_subsets", "ASE")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Tableau 121. Propriétés de linearnode.

Propriétés de linearnode	Valeurs	Description de la propriété
target	*champ*	Spécifie un champ cible unique.
inputs	[*champ1 ... champN*]	Champs prédicteurs utilisés par le modèle.
continue_training_existing_model	*indicateur*	
objective	Standard Bagging Boosting psm	psm est destinée aux jeux de données très volumineux et nécessite une connexion à un serveur.
use_auto_data_preparation	*indicateur*	
confidence_level	*nombre*	
model_selection	ForwardStepwise BestSubsets None	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquared ASE	
probability_entry	*nombre*	
probability_removal	*nombre*	
use_max_effects	*indicateur*	
max_effects	*nombre*	
use_max_steps	*indicateur*	
max_steps	*nombre*	
criteria_best_subsets	AICC AdjustedRSquared ASE	
combining_rule_continuous	Mean Median	
component_models_n	*nombre*	
use_random_seed	*indicateur*	

Tableau 121. Propriétés de linearnode (suite).

Propriétés de linearnode	Valeurs	Description de la propriété
random_seed	*nombre*	
use_custom_model_name	*indicateur*	
custom_model_name	*chaîne*	
use_custom_name	*indicateur*	
custom_name	*chaîne*	
tooltip	*chaîne*	
keywords	*chaîne*	
annotation	*chaîne*	

Propriétés de logregnode

La régression logistique est une technique statistique de classification des enregistrements sur la base des valeurs des champs d'entrée. Excepté le fait qu'elle utilise un champ cible catégoriel et non pas numérique, cette régression est similaire à la régression linéaire.

Exemple multinomial

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Log_reg Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("logistic_procedure", "Multinomial")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("model_type", "FullFactorial")
node.setPropertyValue("custom_terms", [["BP", "Sex"], ["Age"], ["Na", "K"]])
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("max_steps", 3)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
node.setPropertyValue("delta", 0.03)
# "Output..." section
node.setPropertyValue("summary", True)
node.setPropertyValue("likelihood_ratio", True)
node.setPropertyValue("asymptotic_correlation", True)
node.setPropertyValue("goodness_fit", True)
node.setPropertyValue("iteration_history", True)
```

```
node.setPropertyValue("history_steps", 3)
node.setPropertyValue("parameters", True)
node.setPropertyValue("confidence_interval", 90)
node.setPropertyValue("asymptotic_covariance", True)
node.setPropertyValue("classification_table", True)
# "Stepping" options
node.setPropertyValue("min_terms", 7)
node.setPropertyValue("use_max_terms", True)
node.setPropertyValue("max_terms", 10)
node.setPropertyValue("probability_entry", 3)
node.setPropertyValue("probability_removal", 5)
node.setPropertyValue("requirements", "Containment")
```

Exemple binomial

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Cholesterol")
node.setPropertyValue("inputs", ["BP", "Drug", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Log_reg Cholesterol")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("binomial_method", "Forwards")
node.setPropertyValue("logistic_procedure", "Binomial")
node.setPropertyValue("binomial_categorical_input", "Sex")
node.setKeyedPropertyValue("binomial_input_contrast", "Sex", "Simple")
node.setKeyedPropertyValue("binomial_input_category", "Sex", "Last")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
# "Output..." section
node.setPropertyValue("binomial_output_display", "at_each_step")
node.setPropertyValue("binomial_goodness_of_fit", True)
node.setPropertyValue("binomial_iteration_history", True)
node.setPropertyValue("binomial_parameters", True)
node.setPropertyValue("binomial_ci_enable", True)
node.setPropertyValue("binomial_ci", 85)
# "Stepping" options
node.setPropertyValue("binomial_removal_criterion", "LR")
node.setPropertyValue("binomial_probability_removal", 0.2)
```

Tableau 122. propriétés de logregnode.

Propriétés de logregnode	Valeurs	Description de la propriété
target	*champ*	Les modèles de régression logistique requièrent un seul champ cible et un ou plusieurs champs d'entrée. Les champs de fréquences et de pondération ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
logistic_procedure	Binomial Multinomial	
include_constant	*indicateur*	
mode	Simple Expert	
method	Enter Stepwise Forwards Backwards BackwardsStepwise	
binomial_method	Enter Forwards Backwards	
model_type	MainEffects FullFactorial Custom	Lorsque le type de modèle indiqué est FullFactorial, les méthodes d'analyse pas à pas ne sont pas exécutées, même si elles sont spécifiées. La méthode utilisée sera la méthode Enter. Si le type de modèle est paramétré sur Custom, mais qu'aucun champ personnalisé n'est indiqué, un modèle Effets principaux est créé.
custom_terms	[*{BP Sex}{BP}{Age}*]	
multinomial_base_category	*chaîne*	Indique le mode de détermination de la catégorie de référence.
binomial_categorical_input	*chaîne*	
binomial_input_contrast	Indicator Simple Difference Helmert Repeated Polynomial Deviation	Propriété saisie pour l'entrée catégorielle qui indique la façon dont le contraste est déterminé.
binomial_input_category	First Last	Propriété saisie pour l'entrée catégorielle qui indique la façon dont la catégorie de référence est déterminée.
scale	None UserDefined Pearson Deviance	
scale_value	*nombre*	

Tableau 122. propriétés de logregnode (suite).

Propriétés de logregnode	Valeurs	Description de la propriété
all_probabilities	*indicateur*	
tolerance	1.0E-5 1.0E-6 1.0E-7 1.0E-8 1,0E-9 1.0E-10	
min_terms	*nombre*	
use_max_terms	*indicateur*	
max_terms	*nombre*	
entry_criterion	Score LR	
removal_criterion	LR Wald	
probability_entry	*nombre*	
probability_removal	*nombre*	
binomial_probability_entry	*nombre*	
binomial_probability_removal	*nombre*	
requirements	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	*nombre*	
max_steps	*nombre*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
delta	*nombre*	
iteration_history	*indicateur*	
history_steps	*nombre*	
summary	*indicateur*	
likelihood_ratio	*indicateur*	
asymptotic_correlation	*indicateur*	
goodness_fit	*indicateur*	
parameters	*indicateur*	
confidence_interval	*nombre*	
asymptotic_covariance	*indicateur*	
classification_table	*indicateur*	

Tableau 122. propriétés de logregnode (suite).

Propriétés de logregnode	Valeurs	Description de la propriété
stepwise_summary	*indicateur*	
info_criteria	*indicateur*	
monotonicity_measures	*indicateur*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*indicateur*	
binomial_parameters	*indicateur*	
binomial_iteration_history	*indicateur*	
binomial_classification_plots	*indicateur*	
binomial_ci_enable	*indicateur*	
binomial_ci	*nombre*	
binomial_residual	outliers all	
binomial_residual_enable	*indicateur*	
binomial_outlier_threshold	*nombre*	
binomial_classification_cutoff	*nombre*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*indicateur*	
calculate_raw_propensities	*indicateur*	

Propriétés de neuralnetnode

Attention : une nouvelle version du noeud de modélisation Réseau de neurones, avec des caractéristiques améliorées, est disponible dans cette version et est décrite dans la section suivante (*neuralnetwork*). Bien que vous soyez encore en mesure de créer et d'évaluer un modèle avec la version précédente, nous vous conseillons de procéder à la mise à jour de vos scripts afin d'utiliser la nouvelle version. Les détails de la version précédentes sont conservées ici pour référence.

Exemple

```
node = stream.create("neuralnet", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("targets", ["Drug"])
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
# "Model" tab
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Dynamic")
node.setPropertyValue("train_pct", 30)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("accuracy", 95)
node.setPropertyValue("cycles", 200)
node.setPropertyValue("time", 3)
node.setPropertyValue("optimize", "Speed")
```

```
# "Multiple Method Expert Options" section
node.setPropertyValue("m_topologies", "5 30 5; 2 20 3, 1 10 1")
node.setPropertyValue("m_non_pyramids", False)
node.setPropertyValue("m_persistence", 100)
```

Tableau 123. propriétés de neuralnetnode

Propriétés de **neuralnetnode**	Valeurs	Description de la propriété
targets	[*champ1 ... champN*]	Le noeud Réseau de neurones attend un ou plusieurs champs cible et un ou plusieurs champs d'entrée. Les champs de fréquence et de pondération sont ignorés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	*flag*	
train_pct	*nombre*	
set_random_seed	*flag*	
random_seed	*nombre*	
mode	Simple Expert	
stop_on	Default Accuracy Cycles Heure	Mode d'arrêt.
exactitude	*nombre*	Exactitude d'arrêt.
cycles	*nombre*	Cycles d'apprentissage.
heure	*nombre*	Durée d'apprentissage (minutes).
continue	*flag*	
show_feedback	*flag*	
binary_encode	*flag*	
use_last_model	*flag*	
gen_logfile	*flag*	
logfile_name	*chaîne*	
alpha	*nombre*	
initial_eta	*nombre*	
high_eta	*nombre*	
low_eta	*nombre*	
eta_decay_cycles	*nombre*	
hid_layers	One Two Trois	
hl_units_one	*nombre*	
hl_units_two	*nombre*	

Tableau 123. propriétés de neuralnetnode (suite)

Propriétés de neuralnetnode	Valeurs	Description de la propriété
hl_units_three	*nombre*	
persistance	*nombre*	
m_topologies	*chaîne*	
m_non_pyramids	*flag*	
m_persistence	*nombre*	
p_hid_layers	One Two Trois	
p_hl_units_one	*nombre*	
p_hl_units_two	*nombre*	
p_hl_units_three	*nombre*	
p_persistence	*nombre*	
p_hid_rate	*nombre*	
p_hid_pers	*nombre*	
p_inp_rate	*nombre*	
p_inp_pers	*nombre*	
p_overall_pers	*nombre*	
r_persistence	*nombre*	
r_num_clusters	*nombre*	
r_eta_auto	*flag*	
r_alpha	*nombre*	
r_eta	*nombre*	
optimize	Vitesse Memory	Permet d'indiquer si la création du modèle doit être optimisée en vitesse ou en mémoire.
calculate_variable_importance	*flag*	Remarque : la propriété sensitivity_analysis utilisée dans les versions précédentes est remplacée par cette propriété. L'ancienne propriété est toujours prise en charge, mais il est recommandé d'utiliser calculate_variable_importance.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés de neuralnetworknode

Le noeud R. neurones est un modèle simplifié de la manière dont le cerveau humain traite les informations. Le fonctionnement de ce modèle repose sur la simulation d'un grand nombre d'unités de traitement simples interconnectées, qui sont en quelque sorte des versions abstraites de nos neurones. Les réseaux de neurones sont de puissants estimateurs de fonctions qui ne requièrent qu'une connaissance limitée en matière de statistiques ou de mathématiques.

Exemple

```
node = stream.create("neuralnetwork", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Tableau 124. propriétés de neuralnetworknode

Propriétés de neuralnetworknode	Valeurs	Description de la propriété
targets	[*champ1 ... champN*]	Spécifie des champs cible.
inputs	[*champ1 ... champN*]	Champs prédicteurs utilisés par le modèle.
splits	*[champ1 ... champN*	Indique le champ ou les champs à utiliser pour la modélisation découpée.
use_partition	*flag*	Si un champ de partition est défini, cette option assure que seules les données provenant de la partition d'apprentissage sont utilisées pour générer le modèle.
continue	*flag*	Poursuivre le modèle d'apprentissage existant.
objective	Standard Bagging Boosting psm	psm est destinée aux jeux de données très volumineux et nécessite une connexion à un serveur.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*flag*	
first_layer_units	*nombre*	
second_layer_units	*nombre*	
use_max_time	*flag*	
max_time	*nombre*	
use_max_cycles	*flag*	
max_cycles	*nombre*	
use_min_accuracy	*flag*	
min_accuracy	*nombre*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	
combining_rule_continuous	Mean Median	
component_models_n	*nombre*	
overfit_prevention_pct	*nombre*	
use_random_seed	*flag*	
random_seed	*nombre*	
missing_values	listwiseDeletion missingValueImputation	

Tableau 124. propriétés de neuralnetworknode (suite)

Propriétés de neuralnetworknode	Valeurs	Description de la propriété
use_model_name	*booléen*	
model_name	*chaîne*	
confiance	onProbability onIncrease	
score_category_probabilities	*flag*	
max_categories	*nombre*	
score_propensity	*flag*	
use_custom_name	*flag*	
custom_name	*chaîne*	
tooltip	*chaîne*	
keywords	*chaîne*	
annotation	*chaîne*	

Propriétés de questnode

Le noeud QUEST est une méthode de classification binaire permettant de créer des arbres décision, développée pour réduire le temps de traitement nécessaire aux analyses C&R Tree importantes, tout en limitant la tendance, observée parmi les méthodes d'arbre de classification, à favoriser les entrées autorisant un nombre supérieur de divisions. Les champs d'entrée peuvent être des intervalles numériques (continues) mais les champs cible doivent être catégoriels. Toutes les divisions sont binaires.

Exemple

```
node = stream.create("quest", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("max_surrogates", 5)
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("prune_tree", True)
node.setPropertyValue("use_std_err", True)
node.setPropertyValue("std_err_multiplier", 3)
```

Tableau 125. propriétés de questnode

Propriétés de questnode	Valeurs	Description de la propriété
target	*zone*	Les modèles QUEST requièrent un seul champ cible et un ou plusieurs champs d'entrée. Un champ de fréquence peut aussi être spécifié. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
continue_training_existing_model	*flag*	

Tableau 125. propriétés de questnode (suite)

Propriétés de questnode	Valeurs	Description de la propriété
objective	Standard Boosting Bagging psm	psm est destinée aux jeux de données très volumineux et nécessite une connexion à un serveur.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*chaîne*	
use_max_depth	Default Personnalisé	
max_depth	*entier*	Profondeur maximale de l'arbre, comprise entre 0 et 1000. Utilisée uniquement si use_max_depth = Custom
prune_tree	*flag*	Elagage de l'arbre pour éviter le surajustement.
use_std_err	*flag*	Utiliser la différence maximale pour le risque (dans Erreurs standard).
std_err_multiplier	*nombre*	Différence maximale.
max_surrogates	*nombre*	Nombre maximal de substitutions.
use_percentage	*flag*	
min_parent_records_pc	*nombre*	
min_child_records_pc	*nombre*	
min_parent_records_abs	*nombre*	
min_child_records_abs	*nombre*	
use_costs	*flag*	
coûts	*structurées*	Propriétés structurées.
priors	Data Equal Personnalisé	
custom_priors	*structurées*	Propriétés structurées.
adjust_priors	*flag*	
trails	*nombre*	Nombre des modèles de composant pour le boosting ou le bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Règles de combinaison par défaut pour les cibles catégorielles.
range_ensemble_method	Mean Médiane	Règles de combinaison par défaut pour les cibles continues.
large_boost	*flag*	Appliquer le boosting aux jeux de données très volumineux.
split_alpha	*nombre*	Niveau de signification pour la division.
train_pct	*nombre*	Ensemble de prévention de surajustement.
set_random_seed	*flag*	Dupliquer l'option des résultats.
seed	*nombre*	
calculate_variable_importance	*flag*	

Tableau 125. propriétés de questnode (suite)

Propriétés de questnode	Valeurs	Description de la propriété
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Test Validation	

Propriétés de regressionnode

La régression linéaire est une technique statistique couramment utilisée dans le domaine de la synthèse de données et de la prévision. Cette technique établit une ligne droite ou une surface afin de réduire les écarts entre les valeurs de sortie prévues et observées.

Remarque : le noeud Régression sera remplacé par le noeud Linéaire dans une prochaine version. Nous recommandons d'ores et déjà d'utiliser les modèles linéaires pour la régression linéaire.

Exemple

```
node = stream.create("regression", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Age")
node.setPropertyValue("inputs", ["Na", "K"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_weight", True)
node.setPropertyValue("weight_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Regression Age")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", False)
node.setPropertyValue("tolerance", "1.0E-3")
# "Stepping..." section
node.setPropertyValue("stepping_method", "Probability")
node.setPropertyValue("probability_entry", 0.77)
node.setPropertyValue("probability_removal", 0.88)
node.setPropertyValue("F_value_entry", 7.0)
node.setPropertyValue("F_value_removal", 8.0)
# "Output..." section
node.setPropertyValue("model_fit", True)
node.setPropertyValue("r_squared_change", True)
node.setPropertyValue("selection_criteria", True)
node.setPropertyValue("descriptives", True)
node.setPropertyValue("p_correlations", True)
node.setPropertyValue("collinearity_diagnostics", True)
node.setPropertyValue("confidence_interval", True)
node.setPropertyValue("covariance_matrix", True)
node.setPropertyValue("durbin_watson", True)
```

Tableau 126. propriétés de regressionnode

Propriétés de regressionnode	Valeurs	Description de la propriété
target	*zone*	Les modèles Régression requièrent un seul champ cible et un ou plusieurs champs d'entrée. Un champ poids peut aussi être spécifié. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
method	Enter Stepwise Backwards Ascendante	
include_constant	*flag*	
use_weight	*flag*	
weight_field	*zone*	
mode	Simple Expert	
complete_records	*flag*	
tolerance	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 1,0E-9 1.0E-10 1,0E-11 1,0E-12	Pour les arguments, utilisez des guillemets doubles.
stepping_method	useP useF	useP : utiliser la probabilité de F useF : utiliser la valeur F
probability_entry	*nombre*	
probability_removal	*nombre*	
F_value_entry	*nombre*	
F_value_removal	*nombre*	
selection_criteria	*flag*	
confidence_interval	*flag*	
covariance_matrix	*flag*	
collinearity_diagnostics	*flag*	
regression_coefficients	*flag*	
exclude_fields	*flag*	
durbin_watson	*flag*	
model_fit	*flag*	
r_squared_change	*flag*	
p_correlations	*flag*	
descriptives	*flag*	
calculate_variable_importance	*flag*	

Propriétés de sequencenode

Le noeud Séquence recherche des règles d'association dans des données dotées d'une dimension temporelle. Une séquence est une liste de jeux d'éléments ayant tendance à survenir dans un ordre prévisible. Par exemple, un client qui achète un rasoir et une lotion après-rasage achètera vraisemblablement de la crème à raser. Le noeud Séquence est basé sur l'algorithme de règles d'association CARMA, qui utilise une méthode efficace de double lecture pour rechercher des séquences.

Exemple

```
node = stream.create("sequence", "My node")
# "Fields" tab
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("use_time_field", True)
node.setPropertyValue("time_field", "Date1")
node.setPropertyValue("content_fields", ["Drug", "BP"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Sequence_test")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 15.0)
node.setPropertyValue("min_conf", 14.0)
node.setPropertyValue("max_size", 7)
node.setPropertyValue("max_predictions", 5)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_max_duration", True)
node.setPropertyValue("max_duration", 3.0)
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 4.0)
node.setPropertyValue("set_mem_sequences", True)
node.setPropertyValue("mem_sequences", 5.0)
node.setPropertyValue("use_gaps", True)
node.setPropertyValue("min_item_gap", 20.0)
node.setPropertyValue("max_item_gap", 30.0)
```

Tableau 127. propriétés de sequencenode

Propriétés de sequencenode	Valeurs	Description de la propriété
id_field	*zone*	Pour créer un modèle Séquence, vous devez renseigner un champ ID, un champ de temps facultatif et au moins un champ d'analyse. Les champs de pondération et de fréquence ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
time_field	*zone*	
use_time_field	*flag*	
content_fields	[*champ1 ... champn*]	
contiguous	*flag*	
min_supp	*nombre*	
min_conf	*nombre*	

Tableau 127. propriétés de sequencenode (suite)

Propriétés de sequencenode	Valeurs	Description de la propriété
max_size	*nombre*	
max_predictions	*nombre*	
mode	Simple Expert	
use_max_duration	*flag*	
max_duration	*nombre*	
use_gaps	*flag*	
min_item_gap	*nombre*	
max_item_gap	*nombre*	
use_pruning	*flag*	
pruning_value	*nombre*	
set_mem_sequences	*flag*	
mem_sequences	*entier*	

Propriétés de slrmnode

Le noeud Modèle de réponse en auto-apprentissage (SLRM) vous permet de créer un modèle dans lequel une nouvelle observation unique, ou un petit nombre de nouvelles observations, peuvent être utilisés pour réestimer un modèle sans qu'un recyclage de toutes les données soit nécessaire.

Exemple

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

Tableau 128. Propriétés de slrmnode

Propriétés de slrmnode	Valeurs	Description de la propriété
target	*zone*	Le champ cible doit être nominal ou indicateur. Un champ de fréquence peut aussi être spécifié. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
target_response	*zone*	Type doit être indicateur.
continue_training_existing_model	*flag*	
target_field_values	*flag*	Use all : Utiliser toutes les valeurs source. Specify : Sélectionner les valeurs nécessaires.
target_field_values_specify	*[champ1 ... champN]*	
include_model_assessment	*flag*	
model_assessment_random_seed	*nombre*	Doit être un nombre réel.
model_assessment_sample_size	*nombre*	Doit être un nombre réel.

Tableau 128. Propriétés de slrmnode (suite)

Propriétés de slrmnode	Valeurs	Description de la propriété
model_assessment_iterations	*nombre*	Nombre d'itérations.
display_model_evaluation	*flag*	
max_predictions	*nombre*	
randomization	*nombre*	
scoring_random_seed	*nombre*	
sort	Ascending Décroissant	Indique si les premières offres affichées sont celles dont les scores sont les plus élevés ou les moins élevés.
model_reliability	*flag*	
calculate_variable_importance	*flag*	

Propriétés statisticsmodelnode

Le noeud Modèle Statistics vous permet d'analyser et de travailler avec vos données en exécutant des procédures IBM SPSS Statistics qui produisent un PMML. Ce noeud requiert une copie avec licence de IBM SPSS Statistics.

Les propriétés de ce nœud sont décrites dans «Propriétés statisticsmodelnode», à la page 300.

Propriétés de stpnode

Le noeud de prévisions spatio-temporelles (STP) utilise des données contenant des informations d'emplacement, des champs d'entrée pour la prévision (prédicteurs), une zone temporelle et une zone cible. Chaque emplacement correspond à plusieurs lignes dans les données correspondant aux valeurs de chaque prédicteur à chaque mesure. Une fois les données analysées, elles peuvent être utilisées pour anticiper des valeurs cibles à n'importe quel emplacement dans les données de forme utilisées dans l'analyse.

Tableau 129. Propriétés de stpnode

Propriétés de stpnode	Type de données	Description de la propriété
Onglet **Champs**		
target	*zone*	Il s'agit du champ cible.
emplacement	*zone*	Champ d'emplacement pour le modèle. Seuls les champs géospatiaux sont autorisés.
location_label	*zone*	Champ catégoriel à utiliser pour libeller les emplacements sélectionnés dans location
time_field	*zone*	Champ de date/heure pour le modèle. Seuls les champs dont le niveau de mesure est continu sont autorisés et le type de stockage doit être de type heure, date, horodatage ou entier.
inputs	*[champ1 ... champN]*	Liste de champs d'entrée.
Onglet **Intervalles de temps**		

Tableau 129. Propriétés de stpnode (suite)

Propriétés de stpnode	Type de données	Description de la propriété
interval_type_timestamp	Years Quarters Mois Weeks Days Hours Minutes Secondes	
interval_type_date	Years Quarters Mois Weeks Jours	
interval_type_time	Hours Minutes Secondes	Limite le nombre de jours par semaine pris en compte lors de la création de l'index de date/heure utilisé par STP pour le calcul
interval_type_integer	Periods (Champs d'index de date/heure uniquement, Stockage d'entier)	Intervalle auquel le jeu de données sera converti. La sélection disponible dépend du type de stockage du champ sélectionné comme champ de date/heure (time_field) pour le modèle.
period_start	*entier*	
start_month	Janvier Février Mars Avril Mai Juin Juillet Août Septembre Octobre Novembre Décembre	Mois à partir duquel le modèle démarre l'indexation (par exemple, si la valeur est March (mars), mais que le premier enregistrement du jeu de données est January (janvier), le modèle ignorera les deux premiers enregistrements et commencera l'indexation à Mars.
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	Point de départ de l'index de date/heure créé par STP à partir des données
days_per_week	*entier*	Minimum 1, maximum 7, par incréments de 1
hours_per_day	*entier*	Nombre d'heures que le modèle prend en compte dans une journée. Si la valeur est 10, le modèle démarre l'indexation à l'heure de day_begins_at, continue l'indexation pendant 10 heures, puis passe à la valeur suivante correspondant à la valeur de day_begins_at, etc.

Tableau 129. Propriétés de stpnode (suite)

Propriétés de stpnode	Type de données	Description de la propriété
day_begins_at	00:00 01:00 02:00 03:00 ... 23:00	Définit la valeur d'heure à partir de laquelle le modèle démarre l'indexation.
interval_increment	1 2 3 4 5 6 10 12 15 20 30	Ce paramètre d'incrémentation correspond aux minutes ou aux secondes. Il détermine à quel moment le modèle crée des index depuis les données. Ainsi, avec un incrément de 30 et un type d'intervalle seconds, le modèle crée un index à partir des données toutes les 30 secondes.
data_matches_interval	*Booléen*	Si la valeur est N, la conversion des données en interval_type classique a lieu avant la création du modèle. Si les données sont déjà au format correct, et si interval_type et les paramètres associés correspondent à vos données, indiquez Y pour empêcher la conversion ou l'agrégation de vos données. Si vous indiquez Y pour cette propriété, tous les contrôles d'agrégation sont désactivés.
agg_range_default	Sum Mean Min Max Médiane 1stQuartile 3rdQuartile	Détermine la méthode d'agrégation par défaut utilisée pour les champs continus. Les champs continus qui ne sont pas inclus spécifiquement dans l'agrégation personnalisée seront agrégés à l'aide de la méthode spécifiée ici.
agg_set_default	Mode Min Max	Identique à agg_range_default, mais pour les champs nominaux/ catégoriels.
agg_flag_default	TrueIfAnyTrue FalseIfAnyFalse	Méthode d'agrégation par défaut à appliquer à tous les champs continus qui ne sont pas spécifiés individuellement.

Tableau 129. Propriétés de stpnode (suite)

Propriétés de stpnode	Type de données	Description de la propriété
custom_agg	[{champ, méthode d'agrégation},{}..] Démonstration : [{'x5' 'FirstQuartile'}{'x4' 'Sum'}]	Propriété structurée : Paramètre de script : custom_agg Par exemple : set :stpnode.custom_agg = [{champ1 fonction} {champ2 fonction}] Où fonction est la fonction d'agrégation à utiliser avec ce champ.
Onglet **Bases**		
include_intercept	*flag*	
max_autoregressive_lag	*entier*	Minimum 1, maximum 5, par incréments de 1. Il s'agit du nombre d'enregistrements précédents requis pour une prévision. Par exemple, si la valeur est 5, les 5 enregistrements précédents sont utilisés pour créer une prévision. Le nombre d'enregistrements spécifié ici depuis les données de création sont incorporés dans le modèle. Ainsi, l'utilisateur n'a pas besoin de fournir à nouveau les données lors du scoring du modèle.
estimation_method	Parametric Nonparametric	Méthode de modélisation de la matrice de covariance spatiale
parametric_model	Gaussian Exponential PoweredExponential	Paramètre d'ordre pour le modèle de covariance spatiale Parametric
exponential_power	*nombre*	Niveau de puissance pour le modèle PoweredExponential. Minimum 1, maximum 2.
Onglet **Avancé**		
max_missing_values	*entier*	Pourcentage maximal d'enregistrements avec des valeurs manquantes autorisé dans le modèle.
significance	*nombre*	Niveau de signification pour les tests d'hypothèses dans la construction du modèle Spécifie la valeur d'importance pour tous les tests d'estimation du modèle STP, y-compris deux tests de qualité d'ajustement, tests d'effets F et tests de coefficient T.
Onglet **Sortie**		
model_specifications	*flag*	
temporal_summary	*flag*	

Tableau 129. Propriétés de stpnode (suite)

Propriétés de stpnode	Type de données	Description de la propriété
location_summary	*flag*	Détermine si la table récapitulative des emplacements (Location Summary) est incluse dans la sortie du modèle.
model_quality	*flag*	
test_mean_structure	*flag*	
mean_structure_coefficients	*flag*	
autoregressive_coefficients	*flag*	
test_decay_space	*flag*	
parametric_spatial_covariance	*flag*	
correlations_heat_map	*flag*	
correlations_map	*flag*	
location_clusters	*flag*	
similarity_threshold	*nombre*	Seuil auquel les clusters de sortie sont considérés comme étant suffisamment similaires pour être fusionnés en un seul cluster.
max_number_clusters	*entier*	Nombre maximal de clusters pouvant être inclus dans la sortie du modèle.
Onglet **Options de modèle**		
use_model_name	*flag*	
model_name	*chaîne*	
uncertainty_factor	*nombre*	Minimum 0, maximum 100. Détermine l'augmentation d'incertitude (erreur) appliquée aux prévisions à l'avenir. Il s'agit des limites supérieure et inférieure pour les prévisions.

Propriétés de svmnode

Le noeud Support Vector Machine (SVM) vous permet de classer les données dans l'un de deux groupes sans surajustement. SVM fonctionne bien avec les grands jeux de données, comme ceux qui disposent d'un très grand nombre de champs d'entrée.

Exemple

```
node = stream.create("svm", "My node")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("kernel", "Polynomial")
node.setPropertyValue("gamma", 1.5)
```

Tableau 130. propriétés de svmnode.

Propriétés de svmnode	Valeurs	Description de la propriété
all_probabilities	*indicateur*	

Tableau 130. propriétés de svmnode (suite).

Propriétés de svmnode	Valeurs	Description de la propriété
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (valeur par défaut) 1.0E-4 1.0E-5 1.0E-6	Détermine le moment de l'arrêt de l'algorithme d'optimisation.
regularization	*nombre*	Egalement appelé Paramètre C
precision	*nombre*	Option uniquement utilisée si le niveau de mesure du champ cible est Continuous.
kernel	RBF(valeur par défaut) Polynomial Sigmoid Linear	Type de fonction de noyau utilisée pour la transformation.
rbf_gamma	*nombre*	Utilisé uniquement si le kernel est RBF.
gamma	*nombre*	Utilisé uniquement si le kernel est Polynomial ou Sigmoid.
bias	*nombre*	
degree	*nombre*	Utilisé uniquement si le kernel est Polynomial.
calculate_variable_importance	*indicateur*	
calculate_raw_propensities	*indicateur*	
calculate_adjusted_ propensities	*indicateur*	
adjusted_propensity_partition	Test Validation	

Propriétés de tcmnode

La modélisation de causalité temporelle tente de découvrir des relations de causalité clés dans les données de séries temporelles. Dans la modélisation de causalité temporelle, vous spécifiez un ensemble de séries cibles et un ensemble d'entrées candidates sur ces cibles. La procédure construit alors un modèle de série temporelle autorégressive pour chaque cible et n'inclut que les entrées avec la relation de causalité la plus significative avec la cible.

Tableau 131. Propriétés de tcmnode

Propriétés de tcmnode	Valeurs	Description de la propriété
custom_fields	*Booléen*	
dimensionlist	[*dimension1 ... dimensionN*]	
data_struct	Multiple Single	
metric_fields	*champs*	
both_target_and_input	[*c1 ... cN*]	
targets	[*c1 ... cN*]	
candidate_inputs	[*c1 ... cN*]	
forced_inputs	[*c1 ... cN*]	

Tableau 131. Propriétés de tcmnode (suite)

Propriétés de tcmnode	Valeurs	Description de la propriété
use_timestamp	Timestamp Period	
input_interval	None Inconnu Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*chaîne*	
period_start_value	*entier*	
num_days_per_week	*entier*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*entier*	
start_hour_of_day	*entier*	
timestamp_increments	*entier*	
cyclic_increments	*entier*	
cyclic_periods	*liste (list)*	
output_interval	None Year Quarter Mois Week Day Hour Minute Second	
is_same_interval	Identique Notsame	
cross_hour	*Booléen*	
aggregate_and_distribute	*liste (list)*	
aggregate_default	Mean Somme Mode Min Max	
distribute_default	Mean Somme	

Tableau 131. Propriétés de tcmnode (suite)

Propriétés de **tcmnode**	Valeurs	Description de la propriété
group_default	Mean Somme Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend None	
k_mean_param	*entier*	
k_median_param	*entier*	
missing_value_threshold	*entier*	
conf_level	*entier*	
max_num_predictor	*entier*	
max_lag	*entier*	
epsilon	*nombre*	
seuil	*entier*	
is_re_est	*Booléen*	
num_targets	*entier*	
percent_targets	Rootmean Bic Rsquare	
fields_display	*liste (list)*	
series_display	*liste (list)*	
network_graph_for_target	*Booléen*	
sign_level_for_target	*nombre*	
fit_and_outlier_for_target	*Booléen*	
sum_and_para_for_target	*Booléen*	
impact_diag_for_target	*Booléen*	
impact_diag_type_for_target	Effet Cause Both	
impact_diag_level_for_target	*entier*	
series_plot_for_target	*Booléen*	
res_plot_for_target	*Booléen*	
top_input_for_target	*Booléen*	
forecast_table_for_target	*Booléen*	
same_as_for_target	*Booléen*	
network_graph_for_series	*Booléen*	
sign_level_for_series	*nombre*	
fit_and_outlier_for_series	*Booléen*	
sum_and_para_for_series	*Booléen*	

Tableau 131. Propriétés de tcmnode (suite)

Propriétés de tcmnode	Valeurs	Description de la propriété
impact_diagram_for_series	*Booléen*	
impact_diagram_type_for_series	Effet Cause Both	
impact_diagram_level_for_series	*entier*	
series_plot_for_series	*Booléen*	
residual_plot_for_series	*Booléen*	
forecast_table_for_series	*Booléen*	
outlier_root_cause_analysis	*Booléen*	
causal_levels	*entier*	
root_cause_analysis_method	First Tous	
outlier_table	Interactif Croisé Both	
rmsp_error	*Booléen*	
norm_bic	*Booléen*	
r_square	*Booléen*	
outliers_over_time	*Booléen*	
series_transormation	*Booléen*	
use_estimation_period	*Booléen*	
estimation_period	Heures Observation	
observations	*liste (list)*	
observations_type	Latest La plus ancienne	
observations_num	*entier*	
observations_exclude	*entier*	
extend_records_into_future	*Booléen*	
forecastperiods	*entier*	

Propriétés de timeseriesnode

Le noeud Séries temporelles estime les modèles de lissage exponentiel, d'ARIMA (Autoregressive Integrated Moving Average) univariable et d'ARIMA multivariable (ou fonction de transfert) pour les données de séries temporelles et génère des prévisions d`une performance future. Un noeud Séries temporelles doit toujours être précédé d'un noeud Intervalles de temps.

Exemple

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

Tableau 132. Propriétés de timeseriesnode

Propriétés de **timeseriesnode**	Valeurs	Description de la propriété
targets	*zone*	Le noeud Séries temporelles prévoit une ou plusieurs cibles, utilisant éventuellement un ou plusieurs champs d'entrée en tant que prédicteurs. Les champs de fréquences et de pondération ne sont pas utilisés. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
continue	*flag*	
method	ExpertModeler Exsmooth Arima Reuse	
expert_modeler_method	*flag*	
consider_seasonal	*flag*	
detect_outliers	*flag*	
expert_outlier_additive	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_innovational	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_transient	*flag*	
expert_outlier_seasonal_additive	*flag*	
expert_outlier_local_trend	*flag*	
expert_outlier_additive_patch	*flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*entier*	
arima_d	*entier*	
arima_q	*entier*	
arima_sp	*entier*	
arima_sd	*entier*	
arima_sq	*entier*	
arima_transformation_type	None SquareRoot NaturalLog	

Tableau 132. Propriétés de timeseriesnode (suite)

Propriétés de timeseriesnode	Valeurs	Description de la propriété
arima_include_constant	*flag*	
tf_arima_p. *fieldname*	*entier*	Pour les fonctions de transfert.
tf_arima_d. *fieldname*	*entier*	Pour les fonctions de transfert.
tf_arima_q. *fieldname*	*entier*	Pour les fonctions de transfert.
tf_arima_sp. *fieldname*	*entier*	Pour les fonctions de transfert.
tf_arima_sd. *fieldname*	*entier*	Pour les fonctions de transfert.
tf_arima_sq. *fieldname*	*entier*	Pour les fonctions de transfert.
tf_arima_delay. *fieldname*	*entier*	Pour les fonctions de transfert.
tf_arima_transformation_type. *fieldname*	None SquareRoot NaturalLog	Pour les fonctions de transfert.
arima_detect_outlier_mode	None Automatique	
arima_outlier_additive	*flag*	
arima_outlier_level_shift	*flag*	
arima_outlier_innovational	*flag*	
arima_outlier_transient	*flag*	
arima_outlier_seasonal_additive	*flag*	
arima_outlier_local_trend	*flag*	
arima_outlier_additive_patch	*flag*	
conf_limit_pct	*real*	
max_lags	*entier*	
événements	*champs*	
scoring_model_only	*flag*	Utilisation pour les modèles comportant un grand nombre (des dizaines de milliers) de séries temporelles.

Propriétés de twostepnode

Le noeud TwoStep utilise une méthode de classification non supervisée en deux étapes. La première étape consiste en une exploration des données visant à compresser les données d'entrée brutes en sous-clusters plus faciles à manipuler. Au cours de la seconde étape, l'utilisation d'une méthode de classification hiérarchique permet de fusionner progressivement les sous-clusters en clusters de plus en plus importants. La technique TwoStep a l'avantage d'évaluer automatiquement le nombre de clusters optimal pour les données d'apprentissage. Il peut prendre en charge de manière efficace des types de champ mixtes et des jeux de données volumineux.

Exemple

```
node = stream.create("twostep", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Age", "K", "Na", "BP"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "TwoStep_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("exclude_outliers", True)
node.setPropertyValue("cluster_label", "String")
node.setPropertyValue("label_prefix", "TwoStep_")
node.setPropertyValue("cluster_num_auto", False)
node.setPropertyValue("max_num_clusters", 9)
node.setPropertyValue("min_num_clusters", 3)
node.setPropertyValue("num_clusters", 7)
```

Tableau 133. propriétés de twostepnode

Propriétés de twostepnode	Valeurs	Description de la propriété
inputs	[*champ1 ... champN*]	Les modèles TwoStep utilisent une liste de champs d'entrée, mais pas de cible. Les champs de pondération et de fréquence ne sont pas reconnus. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds de modélisation», à la page 161.
standardize	*flag*	
exclude_outliers	*flag*	
percentage	*nombre*	
cluster_num_auto	*flag*	
min_num_clusters	*nombre*	
max_num_clusters	*nombre*	
num_clusters	*nombre*	
cluster_label	Chaîne Nombre	
label_prefix	*chaîne*	
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

Propriétés de twostepAS

Le cluster TwoStep est un outil d'exploration conçu pour dévoiler des regroupements naturels (ou encore clusters) dans un jeu de données, regroupements qui autrement ne seraient pas apparents. L'algorithme employé par cette procédure dispose de certaines fonctionnalités intéressantes qui le différencient des techniques de mise en cluster traditionnelles, par exemple, des fonctions de gestion des données catégorielles et continues, de sélection automatique du nombre de clusters et d'évolutivité.

Tableau 134. Propriétés de twostepAS

Propriétés de twostepAS	Valeurs	Description de la propriété
inputs	[*c1 ... cN*]	Les modèles TwoStepAS utilisent une liste de champs d'entrée, mais pas de cible. Les champs de pondération et de fréquence ne sont pas reconnus.

Tableau 134. Propriétés de twostepAS (suite)

Propriétés de **twostepAS**	Valeurs	Description de la propriété
use_predefined_roles	Booléen	Valeur par défaut = True
use_custom_field_assignments	Booléen	Valeur par défaut = True
cluster_num_auto	Booléen	Valeur par défaut = True
min_num_clusters	entier	Valeur par défaut = 2
max_num_clusters	entier	Valeur par défaut = 15
num_clusters	entier	Valeur par défaut = 5
clustering_criterion	AIC BIC	
automatic_clustering_method	use_clustering_criterion_setting Distance_jump Minimum Maximum	
feature_importance_method	use_clustering_criterion_setting effect_size	
use_random_seed	Booléen	
random_seed	entier	
distance_measure	Euclidean Loglikelihood	
include_outlier_clusters	Booléen	Valeur par défaut = True
num_cases_in_feature_tree_leaf_is_less _than	entier	Valeur par défaut = 10
top_perc_outliers	entier	Valeur par défaut = 5
initial_dist_change_threshold	entier	Valeur par défaut = 0
leaf_node_maximum_branches	entier	Valeur par défaut = 8
non_leaf_node_maximum_branches	entier	Valeur par défaut = 8
max_tree_depth	entier	Valeur par défaut = 3
adjustment_weight_on_measurement_level	entier	Valeur par défaut = 6
memory_allocation_mb	nombre	Valeur par défaut = 512
delayed_split	Booléen	Valeur par défaut = True
fields_to_standardize	[*c1 ... cN*]	
adaptive_feature_selection	Booléen	Valeur par défaut = True
featureMisPercent	entier	Valeur par défaut = 70
coefRange	nombre	Valeur par défaut = 0,05
percCasesSingleCategory	entier	Valeur par défaut = 95
numCases	entier	Valeur par défaut = 24
include_model_specifications	Booléen	Valeur par défaut = True
include_record_summary	Booléen	Valeur par défaut = True
include_field_transformations	Booléen	Valeur par défaut = True
excluded_inputs	Booléen	Valeur par défaut = True
evaluate_model_quality	Booléen	Valeur par défaut = True
show_feature_importance bar chart	Booléen	Valeur par défaut = True
show_feature_importance_ word_cloud	Booléen	Valeur par défaut = True
show_outlier_clusters interactive_table_and_chart	Booléen	Valeur par défaut = True
show_outlier_clusters_pivot_table	Booléen	Valeur par défaut = True
across_cluster_feature_importance	Booléen	Valeur par défaut = True
across_cluster_profiles_pivot_table	Booléen	Valeur par défaut = True
withinprofiles	Booléen	Valeur par défaut = True
cluster_distances	Booléen	Valeur par défaut = True
cluster_label	Chaîne Nombre	

Tableau 134. Propriétés de twostepAS (suite)

Propriétés de **twostepAS**	Valeurs	Description de la propriété
label_prefix	Chaîne	

Chapitre 14. Propriétés du noeud de nugget de modèle

Les noeuds de nugget de modèle partagent les mêmes propriétés que les autres noeuds. Pour plus d'informations, voir la rubrique «Propriétés communes des noeuds», à la page 69.

Propriétés de applyanomalydetectionnode

Les noeuds de modélisation Détection des anomalies peuvent être utilisés pour générer un nugget de modèle Détection des anomalies. Le nom de génération de script de ce nugget de modèle est *applyanomalydetectionnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de anomalydetectionnode», à la page 161.

Tableau 135. Propriétés de applyanomalydetectionnode.

Propriétés de applyanomalydetectionnode	Valeurs	Description de la propriété
anomaly_score_method	FlagAndScore FlagOnly ScoreOnly	Détermine les sorties créées pour le scoring.
num_fields	*entier*	Champs à signaler.
discard_records	*indicateur*	Indique si les enregistrements sont supprimés de la sortie.
discard_anomalous_records	*indicateur*	Indique s'il faut supprimer les enregistrements irréguliers ou *réguliers*. Cette propriété est désactivée par défaut, ce qui signifie que les enregistrements *réguliers* sont supprimés. Si elle est on, les enregistrements irréguliers sont supprimés. Cette propriété n'est activée que si la propriété discard_records l'est également.

Propriétés de applyapriorinode

Les noeuds de modélisation Apriori peuvent être utilisés pour générer un nugget de modèle Apriori. Le nom de génération de script de ce nugget de modèle est *applyapriorinode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de apriorinode», à la page 163.

Tableau 136. propriétés de applyapriorinode.

Propriétés de applyapriorinode	Valeurs	Description de la propriété
max_predictions	*nombre (entier)*	
ignore_unmatached	*indicateur*	
allow_repeats	*indicateur*	
check_basket	NoPredictions Predictions NoCheck	
criterion	Confidence Support RuleSupport Lift Deployability	

Propriétés de applyautoclassifiernode

Les noeuds de modélisation de Discriminant automatique peuvent être utilisés pour générer un nugget de modèle de Discriminant automatique. Le nom de script de ce nugget de modèle est *applyautoclassifiernode*. Pour de plus amples informations sur la génération de scripts du noeud de modélisation lui-même, voir «Propriétés autoclassifiernode», à la page 167.

Tableau 137. Propriétés de applyautoclassifiernode.

Propriétés de applyautoclassifiernode	Valeurs	Description de la propriété
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	Indique la méthode utilisée pour déterminer le score de l'ensemble. Ce paramètre s'applique uniquement si la cible sélectionnée est un champ indicateur.
flag_voting_tie_selection	Random HighestConfidence RawPropensity	Si une méthode de vote est sélectionnée, indique la manière dont les ex æquo sont résolus. Ce paramètre s'applique uniquement si la cible sélectionnée est un champ indicateur.
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Indique la méthode utilisée pour déterminer le score de l'ensemble. Ce paramètre s'applique uniquement si la cible sélectionnée est un champ d'ensemble.
set_voting_tie_selection	Random HighestConfidence	Si une méthode de vote est sélectionnée, indique la manière dont les ex æquo sont résolus. Ce paramètre s'applique uniquement si la cible sélectionnée est un champ nominal.

Propriétés de applyautoclusternode

Les noeuds de modélisation de cluster automatique peuvent être utilisés pour générer un nugget de modèle de cluster automatique. Le nom de génération de script de ce nugget de modèle est *applydecisionlistnode*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés du noeud de cluster automatique», à la page 169.

Propriétés de applyautonumericnode

Les noeuds de modélisation de numérisation automatique peuvent être utilisés pour générer un nugget de modèle de numérisation automatique. Le nom de script de ce nugget de modèle est *applyautonumericnode*. Pour de plus amples informations sur la génération de scripts du noeud de modélisation lui-même, voir «Propriétés de autonumericnode», à la page 170.

Tableau 138. propriétés de applyautonumericnode.

Propriétés de applyautonumericnode	Valeurs	Description de la propriété
calculate_standard_error	*indicateur*	

Propriétés de applybayesnetnode

Les noeuds de modélisation Réseau Bayésien peuvent être utilisés pour générer un nugget de modèle Réseau Bayésien. Le nom de génération de script de ce nugget de modèle est *applybayesnetnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de bayesnetnode», à la page 172.

Tableau 139. propriétés de applybayesnetnode.

Propriétés de applybayesnetnode	Valeurs	Description de la propriété
all_probabilities	*indicateur*	
raw_propensity	*indicateur*	
adjusted_propensity	*indicateur*	
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés de applyc50node

Les noeuds de modélisation C5.0 peuvent être utilisés pour générer un nugget de modèle C5.0. Le nom de génération de script de ce nugget de modèle est *applyc50node*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de c50node», à la page 174.

Tableau 140. Propriétés de applyc50node.

Propriétés de applyc50node	Valeurs	Description de la propriété
sql_generate	Never NoMissingValues	Permet de définir les options de génération SQL au cours de l'exécution des ensembles de règles.
calculate_conf	*indicateur*	Disponible lorsque la génération SQL est activée, cette propriété inclut des calculs de confiance dans l'arbre généré.
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés de applycarmanode

Les noeuds de modélisation CARMA peuvent être utilisés pour générer un nugget de modèle CARMA. Le nom de génération de script de ce nugget de modèle est *applycarmanode*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de carmanode», à la page 175.

Propriétés de applycartnode

Des noeuds de modélisation d'arbre C&R peuvent être utilisés pour générer un nugget de modèle d'arbre C&R. Le nom de génération de script de ce nugget de modèle est *applycartnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de cartnode», à la page 176.

Tableau 141. propriétés de applycartnode.

Propriétés de applycartnode	Valeurs	Description de la propriété
sql_generate	Never MissingValues NoMissingValues	Permet de définir les options de génération SQL au cours de l'exécution des ensembles de règles.
calculate_conf	*indicateur*	Disponible lorsque la génération SQL est activée, cette propriété inclut des calculs de confiance dans l'arbre généré.
display_rule_id	*indicateur*	Ajoute un champ à la sortie de scoring, indiquant l'ID du noeud terminal auquel chaque enregistrement est affecté.
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés de applychaidnode

Les noeuds de modélisation CHAID peuvent être utilisés pour générer un nugget de modèle CHAID. Le nom de génération de script de ce nugget de modèle est *applychaidnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de chaidnode», à la page 179.

Tableau 142. Propriétés de applychaidnode.

Propriétés de applychaidnode	Valeurs	Description de la propriété
sql_generate	Never MissingValues	
calculate_conf	*indicateur*	
display_rule_id	*indicateur*	Ajoute un champ à la sortie de scoring, indiquant l'ID du noeud terminal auquel chaque enregistrement est affecté.
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés de applycoxregnode

Les noeuds de modélisation de Cox peuvent être utilisés pour générer un nugget de modèle de Cox. Le nom de génération de script de ce nugget de modèle est *applycoxregnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de coxregnode», à la page 181.

Tableau 143. propriétés de applycoxregnode.

Propriétés de applycoxregnode	Valeurs	Description de la propriété
future_time_as	Intervals Fields	
time_interval	*nombre*	
num_future_times	*entier*	
time_field	*champ*	
past_survival_time	*champ*	
all_probabilities	*indicateur*	
cumulative_hazard	*indicateur*	

Propriétés de applydecisionlistnode

Les noeuds de modélisation Liste de décision peuvent être utilisés pour générer un nugget de modèle Liste de décision. Le nom de génération de script de ce nugget de modèle est *applydecisionlistnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de decisionlistnode», à la page 182.

Tableau 144. Propriétés de applydecisionlistnode.

Propriétés de **applydecisionlistnode**	Valeurs	Description de la propriété
enable_sql_generation	*indicateur*	Lorsque la valeur est true (vrai), IBM SPSS Modeler essaie de répercuter le modèle Liste de décision dans SQL.
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés de applydiscriminantnode

Les noeuds de modélisation Discriminant peuvent être utilisés pour générer un nugget de modèle Discriminant. Le nom de génération de script de ce nugget de modèle est *applydiscriminantnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de discriminantnode», à la page 184.

Tableau 145. propriétés de applydiscriminantnode.

Propriétés de **applydiscriminantnode**	Valeurs	Description de la propriété
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés de applyfactornode

Les noeuds de modélisation APC/Facteur peuvent être utilisés pour générer un nugget de modèle APC/Facteur. Le nom de génération de script de ce nugget de modèle est *applyfactornode*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de factornode», à la page 185.

Propriétés de applyfeatureselectionnode

Les noeuds de modélisation Sélection de fonction peuvent être utilisés pour générer un nugget de modèle Sélection de fonction. Le nom de génération de script de ce nugget de modèle est *applyfeatureselectionnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de featureselectionnode», à la page 187.

Tableau 146. Propriétés de applyfeatureselectionnode.

Propriétés de **applyfeatureselectionnode**	Valeurs	Description de la propriété
selected_ranked_fields		Indique les champs classés sélectionnés dans le navigateur de modèle.
selected_screened_fields		Indique les champs filtrés sélectionnés dans le navigateur de modèle.

Propriétés de applygeneralizedlinearnode

Les noeuds de modélisation linéaire généralisée (genlin) peuvent être utilisés pour générer un nugget de modèle linéaire généralisé. Le nom de génération de script de ce nugget de modèle est *applygeneralizedlinearnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de genlinnode», à la page 189.

Tableau 147. propriétés de applygeneralizedlinearnode.

Propriétés de `applygeneralizedlinearnode`	Valeurs	Description de la propriété
`calculate_raw_propensities`	*indicateur*	
`calculate_adjusted_propensities`	*indicateur*	

Propriétés de applyglmmnode

Les noeuds de modélisation MMLG peuvent être utilisés pour générer un nugget de modèle MMLG. Le nom de génération de script de ce nugget de modèle est *applyglmmnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés glmmnode», à la page 192.

Tableau 148. Propriétés de applyglmmnode.

Propriétés de `applyglmmnode`	Valeurs	Description de la propriété
`confidence`	`onProbability` `onIncrease`	Base de calcul de la valeur de confiance de scoring : probabilité prédite la plus élevée ou différence entre les probabilités prédites les plus élevées et les deuxièmes plus élevées.
`score_category_probabilities`	*indicateur*	Si elle est définie sur `True`, produit des probabilités prédites pour les cibles catégorielles. Un champ est créé pour chaque modalité. La valeur par défaut est `False`.
`max_categories`	*entier*	Nombre maximal de catégories pour lesquelles prédire des probabilités. Utilisée uniquement si `score_category_probabilities` a la valeur `True`.
`score_propensity`	*indicateur*	Si elle est définie sur `True`, produit des scores de propension brute (probabilité du résultat "True") pour les modèles avec des cibles indicateur. Si les partitions sont activées, produit également des scores de propension ajustée basés sur la partition de test. La valeur par défaut est `False`.

Propriétés de applyassociationrulesnode

Le noeud modélisation Règles d'association peut être utilisé pour générer un nugget de modèle de règles d'association. Le nom de génération de script de ce nugget de modèle est *applyassociationrulesnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés associationrulesnode», à la page 164.

Tableau 149. Propriétés de applyassociationrulesnode

Propriétés de **applyassociationrulesnode**	Type de données	Description de la propriété
max_predictions	*entier*	Nombre maximal de règles pouvant être appliquées à chaque entrée dans le score.
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	Sélectionnez la mesure utilisée pour déterminer la puissance des règles.
allow_repeats	*Booléen*	Détermine si des règles avec la même prévision sont incluses dans le score.
check_input	NoPredictions Predictions NoCheck	

Propriétés de applykmeansnode

Les noeuds de modélisation k moyenne peuvent être utilisés pour générer un nugget de modèle k moyenne. Le nom de génération de script de ce nugget de modèle est *applykmeansnode*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de kmeansnode», à la page 196.

Propriétés applyknnnode

Les noeuds de modélisation KNN peuvent être utilisés pour générer un nugget de modèle KNN. Le nom de génération de script de ce nugget de modèle est *applyknnnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés knnnode», à la page 197.

Tableau 150. Propriétés applyknnnode.

Propriétés de **applyknnnode**	Valeurs	Description de la propriété
all_probabilities	*indicateur*	
save_distances	*indicateur*	

Propriétés de applykohonennode

Les noeuds de modélisation Kohonen peuvent être utilisés pour générer un nugget de modèle Kohonen. Le nom de génération de script de ce nugget de modèle est *applykohonennode*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de c50node», à la page 174.

Propriétés de applylinearnode

Les noeuds de modélisation linéaire peuvent être utilisés pour générer un nugget de modèle linéaire. Le nom de génération de script de ce nugget de modèle est *applylinearnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de linearnode», à la page 200.

Tableau 151. Propriétés de applylinearnode.

Propriétés **linear**	Valeurs	Description de la propriété
use_custom_name	*indicateur*	

Tableau 151. Propriétés de applylinearnode (suite).

Propriétés linear	Valeurs	Description de la propriété
custom_name	*chaîne*	
enable_sql_generation	*indicateur*	

Propriétés de applylogregnode

Les noeuds de modélisation Régression logistique peuvent être utilisés pour générer un nugget de modèle Régression logistique. Le nom de génération de script de ce nugget de modèle est *applylogregnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de logregnode», à la page 201.

Tableau 152. Propriétés de applylogregnode.

Propriétés de applylogregnode	Valeurs	Description de la propriété
calculate_raw_propensities	*indicateur*	
calculate_conf	*flag*	
enable_sql_generation	*flag*	

Propriétés de applyneuralnetnode

Les noeuds de modélisation Réseau de neurones peuvent être utilisés pour générer un nugget de modèle Réseau de neurones. Le nom de génération de script de ce nugget de modèle est *applyneuralnetnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de neuralnetnode», à la page 205.

Attention : une nouvelle version du nugget Réseau de neurones, avec des fonctions améliorées, est disponible dans cette version et est décrite dans la section suivante (*applyneuralnetwork*). Bien que la version précédente soit encore disponible, nous vous conseillons de procéder à la mise à jour de vos scripts afin d'utiliser la nouvelle version. Les détails de la version précédente sont préservés ici à titre de référence, mais la prise en charge sera supprimée dans une version ultérieure.

Tableau 153. propriétés de applyneuralnetnode.

Propriétés de applyneuralnetnode	Valeurs	Description de la propriété
calculate_conf	*indicateur*	Disponible lorsque la génération SQL est activée, cette propriété inclut des calculs de confiance dans l'arbre généré.
enable_sql_generation	*indicateur*	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés de applyneuralnetworknode

Les noeuds de modélisation Réseau de neurones peuvent être utilisés pour générer un nugget de modèle Réseau de neurones. Le nom de génération de script de ce nugget de modèle est *applyneuralnetworknode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de neuralnetworknode», à la page 207.

Tableau 154. Propriétés de applyneuralnetworknode

Propriétés de applyneuralnetworknode	Valeurs	Description de la propriété
use_custom_name	*flag*	
custom_name	*chaîne*	
confiance	onProbability onIncrease	
score_category_probabilities	*flag*	
max_categories	*nombre*	
score_propensity	*flag*	

Propriétés de applyquestnode

Les noeuds de modélisation QUEST peuvent être utilisés pour générer un nugget de modèle QUEST. Le nom de génération de script de ce nugget de modèle est *applyquestnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de questnode», à la page 209.

Tableau 155. Propriétés de applyquestnode.

Propriétés de applyquestnode	Valeurs	Description de la propriété
sql_generate	Never MissingValues NoMissingValues	
calculate_conf	*indicateur*	
display_rule_id	*indicateur*	Ajoute un champ à la sortie de scoring, indiquant l'ID du noeud terminal auquel chaque enregistrement est affecté.
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés de applyr

Les noeuds de création R peuvent être utilisés pour générer un nugget de modèle R. Le nom de génération de script de ce nugget de modèle est *applyr*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de buildr», à la page 173.

Tableau 156. propriétés applyr

Propriétés de applyr	Valeurs	Description de la propriété
score_syntax	*chaîne*	Syntaxe de script R pour le scoring de modèle.
convert_flags	StringsAndDoubles LogicalValues	Option permettant de convertir des champs indicateurs.
convert_datetime	*flag*	Option permettant de convertir des variables au format de date ou date-heure en format de date/heure R.
convert_datetime_class	POSIXct POSIXlt	Options permettant d'indiquer dans quel format sont converties les variables au format de date ou date-heure.

Tableau 156. propriétés applyr (suite)

Propriétés de applyr	Valeurs	Description de la propriété
convert_missing	*flag*	Option permettant de convertir des valeurs manquantes en valeur R NA.

Propriétés de applyregressionnode

Les noeuds de modélisation Régression linéaire peuvent être utilisés pour générer un nugget de modèle Régression linéaire. Le nom de génération de script de ce nugget de modèle est *applyregressionnode*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de regressionnode», à la page 211.

Propriétés de applyselflearningnode

Les noeuds de modélisation Réponse en auto-apprentissage (SLRM) peuvent être utilisés pour générer un nugget de modèle SLRM. Le nom de génération de script de ce nugget de modèle est *applyselflearningnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de slrmnode», à la page 214.

Tableau 157. propriétés de applyselflearningnode.

Propriétés de applyselflearningnode	Valeurs	Description de la propriété
max_predictions	*nombre*	
randomization	*nombre*	
scoring_random_seed	*nombre*	
sort	ascending descending	Indique si les premières offres affichées sont celles dont les scores sont les plus élevés ou les moins élevés.
model_reliability	*indicateur*	Option Prendre en compte la fiabilité du modèle dans l'onglet Paramètres.

Propriétés de applysequencenode

Les noeuds de modélisation Séquence peuvent être utilisés pour générer un nugget de modèle Séquence. Le nom de génération de script de ce nugget de modèle est *applysequencenode*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de sequencenode», à la page 213.

Propriétés de applysvmnode

Les noeuds de modélisation SVM peuvent être utilisés pour générer un nugget de modèle SVM. Le nom de génération de script de ce nugget de modèle est *applysvmnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de svmnode», à la page 219.

Tableau 158. propriétés de applysvmnode.

Propriétés de applysvmnode	Valeurs	Description de la propriété
all_probabilities	*indicateur*	
calculate_raw_propensities	*indicateur*	
calculate_adjusted_propensities	*indicateur*	

Propriétés applystpnode

Le noeud modélisation STP peut être utilisé pour générer un nugget de modèle associé qui affiche la sortie du modèle dans le visualiseur de sortie. Le nom de génération de script de ce nugget de modèle est *applystpnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de stpnode», à la page 215.

Tableau 159. Propriétés applystpnode

Propriétés `00applystpnode`	Type de données	Description de la propriété
`uncertainty_factor`	*Booléen*	Minimum 0, maximum 100.

Propriétés de applytimeseriesnode

Les noeuds de modélisation Séries temporelles peuvent être utilisés pour générer un nugget de modèle Séries temporelles. Le nom de génération de script de ce nugget de modèle est *applytimeseriesnode*. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de timeseriesnode», à la page 223.

Tableau 160. propriétés de applytimeseriesnode.

Propriétés de `applytimeseriesnode`	Valeurs	Description de la propriété
`calculate_conf`	*indicateur*	
`calculate_residuals`	*indicateur*	

Propriétés de applytwostepnode

Les noeuds de modélisation TwoStep peuvent être utilisés pour générer un nugget de modèle TwoStep. Le nom de génération de script de ce nugget de modèle est *applytwostepnode*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de twostepnode», à la page 225.

Propriétés de applytwostepAS

Les noeuds de modélisation TwoStep AS peuvent être utilisés pour générer un nugget de modèle TwoStep AS. Le nom de génération de script de ce nugget de modèle est *applytwostepAS*. Aucune autre propriété n'existe pour ce nugget de modèle. Pour plus d'informations à propos de la génération de script pour le noeud de modélisation lui-même, voir «Propriétés de twostepAS», à la page 226.

Chapitre 15. Propriétés du nœud de modélisation SGBD

IBM SPSS Modeler prend en charge l'intégration des outils d'exploration de données et de modélisation disponibles auprès des fournisseurs de base de données, notamment Microsoft SQL Server Analysis Services, Oracle Data Mining, IBM DB2 InfoSphere Warehouse et IBM Netezza Analytics. Vous pouvez créer et déterminer le score des modèles à l'aide d'algorithmes natifs de base de données, qui proviennent tous de l'application IBM SPSS Modeler. Les modèles de base de données peuvent également être créés et manipulés via des scripts à l'aide des propriétés décrites dans cette section.

Par exemple, l'extrait de script suivant illustre la création d'un modèle d'arbre de décisions Microsoft via l'interface de génération de scripts de IBM SPSS Modeler :

```
stream = modeler.script.stream()
msbuilder = stream.createAt("mstreenode", "MSBuilder", 200, 200)

msbuilder.setPropertyValue("analysis_server_name", 'localhost')
msbuilder.setPropertyValue("analysis_database_name", 'TESTDB')
msbuilder.setPropertyValue("mode", 'Expert')
msbuilder.setPropertyValue("datasource", 'LocalServer')
msbuilder.setPropertyValue("target", 'Drug')
msbuilder.setPropertyValue("inputs", ['Age', 'Sex'])
msbuilder.setPropertyValue("unique_field", 'IDX')
msbuilder.setPropertyValue("custom_fields", True)
msbuilder.setPropertyValue("model_name", 'MSDRUG')

typenode = stream.findByType("type", None)
stream.link(typenode, msbuilder)
results = []
msbuilder.run(results)
msapplier = stream.createModelApplierAt(results[0], "Drug", 200, 300)
tablenode = stream.createAt("table", "Results", 300, 300)
stream.linkBetween(msapplier, typenode, tablenode)
msapplier.setPropertyValue("sql_generate", True)
tablenode.run([])
```

Propriétés du nœud pour la modélisation Microsoft

Propriétés des nœuds de modélisation Microsoft

Propriétés communes

Les propriétés suivantes sont communes aux noeuds de modélisation de base de données Microsoft.

Tableau 161. Propriétés communes des noeuds Microsoft

Propriétés communes des noeuds Microsoft	Valeurs	Description de la propriété
analysis_database_name	*chaîne*	Nom de la base de données Analysis Services.
analysis_server_name	*chaîne*	Nom de l'hôte Analysis Services.
use_transactional_data	*flag*	Spécifie si les données d'entrée sont au format tabulaire ou transactionnel.
inputs	*liste (list)*	Champs d'entrée pour les données tabulaires.

Tableau 161. Propriétés communes des noeuds Microsoft (suite)

Propriétés communes des noeuds Microsoft	Valeurs	Description de la propriété
`target`	*zone*	Champ prédit (ne s'applique pas au nœuds Classification non supervisée MS ou Classification de séquences).
`unique_field`	*zone*	Champ-clé.
`msas_parameters`	*structurées*	Paramètres d'algorithme. Pour plus d'informations, voir la rubrique «Paramètres d'algorithme», à la page 243.
`with_drillthrough`	*flag*	Option Avec extraction

Arbre de décision MS

Aucune propriété particulière n'est définie pour les noeuds du type `mstreenode`. Reportez-vous aux propriétés Microsoft communes au début de cette section.

Classification non supervisée MS

Aucune propriété particulière n'est définie pour les noeuds du type `msclusternode`. Reportez-vous aux propriétés Microsoft communes au début de cette section.

Règles d'association MS

Les propriétés particulières suivantes sont disponibles pour les noeuds du type `msassocnode`.

Tableau 162. propriétés de msassocnode

Propriétés de `msassocnode`	Valeurs	Description de la propriété
`id_field`	*zone*	Identifie chaque transaction dans les données.
`trans_inputs`	*liste (list)*	Champs d'entrée pour les données transactionnelles.
`transactional_target`	*zone*	Champ variable indépendante (données transactionnelles).

MS Naive Bayes

Aucune propriété particulière n'est définie pour les noeuds du type `msbayesnode`. Reportez-vous aux propriétés Microsoft communes au début de cette section.

Régression linéaire MS

Aucune propriété particulière n'est définie pour les noeuds du type `msregressionnode`. Reportez-vous aux propriétés Microsoft communes au début de cette section.

Réseau neuronal MS

Aucune propriété particulière n'est définie pour les noeuds du type `msneuralnetworknode`. Reportez-vous aux propriétés Microsoft communes au début de cette section.

Régression logistique MS

Aucune propriété particulière n'est définie pour les noeuds du type `mslogisticnode`. Reportez-vous aux propriétés Microsoft communes au début de cette section.

Séries temporelles MS

Aucune propriété particulière n'est définie pour les noeuds du type `mstimeseriesnode`. Reportez-vous aux propriétés Microsoft communes au début de cette section.

Mise en cluster de séquences MS

Les propriétés particulières suivantes sont disponibles pour les noeuds du type `mssequenceclusternode`.

Tableau 163. Propriétés de mssequenceclusternode

Propriétés de mssequenceclusternode	Valeurs	Description de la propriété
id_field	*zone*	Identifie chaque transaction dans les données.
input_fields	*liste (list)*	Champs d'entrée pour les données transactionnelles.
sequence_field	*zone*	Identificateur de séquence.
target_field	*zone*	Champ variable indépendante (données tabulaires).

Paramètres d'algorithme

Chaque type de modèle de base de données Microsoft a des paramètres spécifiques pouvant être définis à l'aide de la propriété `msas_parameters`, par exemple :

```
stream = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters", [["MAXIMUM_INPUT_ATTRIBUTES", 255],
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

Ces paramètres sont issus de SQL Server. Pour visualiser les paramètres pertinents de chaque noeud, procédez comme suit :

1. Placez un noeud source de base de données dans l'espace de travail.
2. Ouvrez le noeud source de base de données.
3. Sélectionnez une source valide dans la liste déroulante **Source de données**.
4. Sélectionnez une table valide dans la liste **Nom de la table**.
5. Cliquez sur **OK** pour fermer le noeud source de base de données.
6. Reliez le noeud de modélisation de base de données Microsoft dont vous voulez répertorier les propriétés.
7. Ouvrez le noeud de modélisation de base de données.
8. Cliquez sur l'onglet **Expert**.

Les propriétés `msas_parameters` disponibles pour ce noeud apparaissent.

Propriétés du nugget de modèle Microsoft

Les propriétés suivantes s'appliquent aux nuggets de modèle créés à l'aide des noeuds de modélisation de la base de données Microsoft.

Arbre de décision MS

Tableau 164. Propriétés de l'arbre de décision MS.

Propriétés de applymstreenode	Valeurs	Description
analysis_database_name	*chaîne*	Ce noeud peut être directement évalué dans un flux. Cette propriété est utilisée pour identifier le nom de la base de données Analysis Services.
analysis_server_name	*chaîne*	Nom de l'hôte du serveur Analysis.

Tableau 164. Propriétés de l'arbre de décision MS (suite).

Propriétés de applymstreenode	Valeurs	Description
datasource	*chaîne*	Nom de la source de données (DSN) ODBC du serveur SQL.
sql_generate	*indicateur*	Active la génération SQL.

Régression linéaire MS

Tableau 165. Propriétés de la régression linéaire MS.

Propriétés de applymsregressionnode	Valeurs	Description
analysis_database_name	*chaîne*	Ce noeud peut être directement évalué dans un flux. Cette propriété est utilisée pour identifier le nom de la base de données Analysis Services.
analysis_server_name	*chaîne*	Nom de l'hôte du serveur Analysis.

Réseau neuronal MS

Tableau 166. Propriétés du réseau neuronal MS.

Propriétés de applymsneuralnetworknode	Valeurs	Description
analysis_database_name	*chaîne*	Ce noeud peut être directement évalué dans un flux. Cette propriété est utilisée pour identifier le nom de la base de données Analysis Services.
analysis_server_name	*chaîne*	Nom de l'hôte du serveur Analysis.

Régression logistique MS

Tableau 167. Propriétés de la régression logistique MS.

Propriétés de applymslogisticnode	Valeurs	Description
analysis_database_name	*chaîne*	Ce noeud peut être directement évalué dans un flux. Cette propriété est utilisée pour identifier le nom de la base de données Analysis Services.
analysis_server_name	*chaîne*	Nom de l'hôte du serveur Analysis.

Séries temporelles MS

Tableau 168. Propriétés des séries temporelles MS.

Propriétés de applymstimeseriesnode	Valeurs	Description
analysis_database_name	*chaîne*	Ce noeud peut être directement évalué dans un flux. Cette propriété est utilisée pour identifier le nom de la base de données Analysis Services.
analysis_server_name	*chaîne*	Nom de l'hôte du serveur Analysis.
start_from	new_prediction historical_prevision	Spécifie s'il faut faire des prévisions futures ou des prévisions historiques.

Tableau 168. Propriétés des séries temporelles MS (suite).

Propriétés de applymstimeseriesnode	Valeurs	Description
new_step	*nombre*	Définit la période de commencement des prévisions futures.
historical_step	*nombre*	Définit la période de commencement des prévisions historiques.
end_step	*nombre*	Définit la période de fin des prévisions.

Mise en cluster de séquences MS

Tableau 169. Propriétés de mise en cluster de séquences MS.

Propriétés de applymssequenceclusternode	Valeurs	Description
analysis_database_name	*chaîne*	Ce noeud peut être directement évalué dans un flux. Cette propriété est utilisée pour identifier le nom de la base de données Analysis Services.
analysis_server_name	*chaîne*	Nom de l'hôte du serveur Analysis.

Propriétés du noeud pour la modélisation Oracle

Propriétés du noeud de modélisation Oracle

Les propriétés suivantes sont communes aux noeuds de modélisation de base de données Oracle.

Tableau 170. Propriétés communes aux noeuds Oracle.

Propriétés communes des noeuds Oracle	Valeurs	Description de la propriété
target	*champ*	
inputs	*Liste des champs*	
partition	*champ*	Champ utilisé pour partitionner les données en échantillons distincts pour les étapes d'apprentissage, de test et de validation de la création d'un modèle.
datasource		
username		
password		
epassword		
use_model_name	*indicateur*	
model_name	*chaîne*	Nom personnalisé du nouveau modèle.
use_partitioned_data	*indicateur*	Si un champ de partition est défini, cette option assure que seules les données provenant de la partition d'apprentissage sont utilisées pour générer le modèle.
unique_field	*champ*	
auto_data_prep	*indicateur*	Active ou désactive la fonction de préparation de données automatique d'Oracle (Bases de données 11g uniquement).

Tableau 170. Propriétés communes aux noeuds Oracle (suite).

Propriétés communes des noeuds Oracle	Valeurs	Description de la propriété
costs	*structurée*	Propriété structurée utilisant la syntaxe : [{drugA drugB 1.5} {drugA drugC 2.1}]où les arguments entre {} sont les coûts réels prédits.
mode	Simple Expert	Certaines propriétés sont ignorées si ce paramètre est défini sur Simple, comme indiqué dans les propriétés de noeud individuelles.
use_prediction_probability	*indicateur*	
prediction_probability	*chaîne*	
use_prediction_set	*indicateur*	

Oracle Naive Bayes

Les propriétés suivantes sont disponibles pour les noeuds du type Oracle Naive Bayes.

Tableau 171. propriétés de oranbnode.

Propriétés de oranbnode	Valeurs	Description de la propriété
singleton_threshold	*nombre*	0,0–1,0.*
pairwise_threshold	*nombre*	0,0–1,0.*
priors	Data Equal Custom	
custom_priors	*structurée*	Propriété structurée utilisant la syntaxe : set :oranbnode.custom_priors = [{drugA 1}{drugB 2}{drugC 3}{drugX 4}{drugY 5}]

* Propriété ignorée si mode est défini sur Simple.

Oracle Adaptive Bayes

Les propriétés suivantes sont disponibles pour les noeuds du type Oracle Adaptive Bayes.

Tableau 172. propriétés de oraabnnode.

Propriétés de oraabnnode	Valeurs	Description de la propriété
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*indicateur*	*
execution_time_limit	*entier*	La valeur doit être supérieure à 0.*
max_naive_bayes_predictors	*entier*	La valeur doit être supérieure à 0.*
max_predictors	*entier*	La valeur doit être supérieure à 0.*
priors	Data Equal Custom	
custom_priors	*structurée*	Propriété structurée utilisant la syntaxe : set :oraabnnode.custom_priors = [{drugA 1}{drugB 2}{drugC 3}{drugX 4}{drugY 5}]

* Propriété ignorée si mode est défini sur Simple.

Oracle Support Vector Machines

Les propriétés suivantes sont disponibles pour les noeuds du type Oracle Support Vector Machines.

Tableau 173. propriétés de orasvmnode.

Propriétés de orasvmnode	Valeurs	Description de la propriété
active_learning	Activer Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	*entier*	Noyau gaussien uniquement. La valeur doit être supérieure à 0.*
convergence_tolerance	*nombre*	La valeur doit être supérieure à 0.*
use_standard_deviation	*indicateur*	Noyau gaussien uniquement.*
standard_deviation	*nombre*	La valeur doit être supérieure à 0.*
use_epsilon	*indicateur*	Modèles de régression uniquement.*
epsilon	*nombre*	La valeur doit être supérieure à 0.*
use_complexity_factor	*indicateur*	*
complexity_factor	*nombre*	*
use_outlier_rate	*indicateur*	Variante à classe unique seulement.*
outlier_rate	*nombre*	Variante à classe unique seulement. 0,0–1,0.*
weights	Data Equal Custom	
custom_weights	*structurée*	Propriété structurée utilisant la syntaxe : set :orasvmnode.custom_weights = [{drugA 1}{drugB 2}{drugC 3}{drugX 4}{drugY 5}]

* Propriété ignorée si mode est défini sur Simple.

Modèles linéaires généralisés d'Oracle

Les propriétés suivantes sont disponibles pour les noeuds du type oraglmnode.

Tableau 174. propriétés oraglmnode.

Propriétés de oraglmnode	Valeurs	Description de la propriété
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	

Tableau 174. propriétés oraglmnode (suite).

Propriétés de oraglmnode	Valeurs	Description de la propriété
use_row_weights	*indicateur*	*
row_weights_field	*champ*	*
save_row_diagnostics	*indicateur*	*
row_diagnostics_table	*chaîne*	*
coefficient_confidence	*nombre*	*
use_reference_category	*indicateur*	*
reference_category	*chaîne*	*
ridge_regression	Auto Off On	*
parameter_value	*nombre*	*
vif_for_ridge	*indicateur*	*

* Propriété ignorée si mode est défini sur Simple.

Arbre décision Oracle

Les propriétés suivantes sont disponibles pour les noeuds du type oradecisiontreenode.

Tableau 175. propriétés de oradecisiontreenode.

Propriétés de oradecisiontreenode	Valeurs	Description de la propriété
use_costs	*indicateur*	
impurity_metric	Entropy Gini	
term_max_depth	*entier*	2–20.*
term_minpct_node	*nombre*	0,0–10,0.*
term_minpct_split	*nombre*	0,0–20,0.*
term_minrec_node	*entier*	La valeur doit être supérieure à 0.*
term_minrec_split	*entier*	La valeur doit être supérieure à 0.*
display_rule_ids	*indicateur*	*

* Propriété ignorée si mode est défini sur Simple.

O-Cluster Oracle

Les propriétés suivantes sont disponibles pour les noeuds du type oraoclusternode.

Tableau 176. propriétés de oraoclusternode.

Propriétés de oraoclusternode	Valeurs	Description de la propriété
max_num_clusters	*entier*	La valeur doit être supérieure à 0.
max_buffer	*entier*	La valeur doit être supérieure à 0.*
sensitivity	*nombre*	0,0–1,0.*

* Propriété ignorée si mode est défini sur Simple.

KMeans Oracle

Les propriétés suivantes sont disponibles pour les noeuds du type orakmeansnode.

Tableau 177. propriétés de orakmeansnode.

Propriétés de orakmeansnode	Valeurs	Description de la propriété
num_clusters	*entier*	La valeur doit être supérieure à 0.
normalization_method	zscore minmax none	
distance_function	Euclidean Cosine	
iterations	*entier*	0–20.*
conv_tolerance	*nombre*	0,0–0,5.*
split_criterion	Variance Size	La valeur par défaut est Variance.*
num_bins	*entier*	La valeur doit être supérieure à 0.*
block_growth	*entier*	1–5.*
min_pct_attr_support	*nombre*	0,0–1,0.*

* Propriété ignorée si mode est défini sur Simple.

NMF Oracle

Les propriétés suivantes sont disponibles pour les noeuds du type oranmfnode.

Tableau 178. propriétés de oranmfnode.

Propriétés de oranmfnode	Valeurs	Description de la propriété
normalization_method	minmax none	
use_num_features	*indicateur*	*
num_features	*entier*	0–1. La valeur par défaut est estimée à partir des données par l'algorithme.*
random_seed	*nombre*	*
num_iterations	*entier*	0–500.*
conv_tolerance	*nombre*	0,0–0,5.*
display_all_features	*indicateur*	*

* Propriété ignorée si mode est défini sur Simple.

Apriori Oracle

Les propriétés suivantes sont disponibles pour les noeuds du type oraapriorinode.

Tableau 179. propriétés de oraapriorinode.

Propriétés de oraapriorinode	Valeurs	Description de la propriété
content_field	*champ*	

Tableau 179. propriétés de oraapriorinode (suite).

Propriétés de oraapriorinode	Valeurs	Description de la propriété
id_field	*champ*	
max_rule_length	*entier*	2–20.
min_confidence	*nombre*	0,0–1,0.
min_support	*nombre*	0,0–1,0.
use_transactional_data	*indicateur*	

Description de longueur minimale d'Oracle (MDL)

Aucune propriété particulière n'est définie pour les noeuds du type MDL Oracle. Reportez-vous aux propriétés Oracle courantes au début de cette section.

Importance de l'attribut d'Oracle (IA)

Les propriétés suivantes sont disponibles pour les noeuds du type oraainode.

Tableau 180. propriétés de oraainode.

Propriétés de oraainode	Valeurs	Description de la propriété
custom_fields	*indicateur*	Si la valeur est définie sur true (vrai), elle permet de spécifier les champs cible, entrée et d'autres champs pour le nœud actuel. Si elle est définie sur false (faux), les paramètres actuels provenant d'un noeud type en amont sont utilisés.
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*indicateur*	Quand la propriété selection_mode est définie sur ImportanceLevel, indique si les champs importants doivent être sélectionnés.
important_label	*chaîne*	Indique le libellé du classement « significatif ».
select_marginal	*indicateur*	Quand la propriété selection_mode est définie sur ImportanceLevel, indique si les champs marginaux doivent être sélectionnés.
marginal_label	*chaîne*	Indique le libellé du classement « marginal ».
important_above	*nombre*	0,0–1,0.
select_unimportant	*indicateur*	Quand la propriété selection_mode est définie sur ImportanceLevel, indique si les champs non significatifs doivent être sélectionnés.
unimportant_label	*chaîne*	Indique le libellé du classement « non significatif ».
unimportant_below	*nombre*	0,0–1,0.
importance_value	*nombre*	Quand la propriété selection_mode est définie sur ImportanceValue, indique la valeur de césure à utiliser. Accepte des valeurs de 0 à 100.
top_n	*nombre*	Quand la propriété selection_mode est définie sur TopN, indique la valeur de césure à utiliser. Accepte des valeurs de 0 à 1000.

Propriétés du nugget de modèle Oracle

Les propriétés suivantes s'appliquent aux nuggets de modèle créés à l'aide des modèles Oracle.

Oracle Naive Bayes

Aucune propriété particulière n'est définie pour les noeuds du type `applyoranbnode`.

Oracle Adaptive Bayes

Aucune propriété particulière n'est définie pour les noeuds du type `applyoraabnnode`.

Oracle Support Vector Machines

Aucune propriété particulière n'est définie pour les noeuds du type `applyorasvmnode`.

Arbre décision Oracle

Les propriétés suivantes sont disponibles pour les noeuds du type `applyoradecisiontreenode`.

Tableau 181. propriétés de applyoradecisiontreenode

Propriétés de `applyoradecisiontreenode`	Valeurs	Description de la propriété
`use_costs`	*flag*	
`display_rule_ids`	*flag*	

O-Cluster Oracle

Aucune propriété particulière n'est définie pour les noeuds du type `applyoraoclusternode`.

KMeans Oracle

Aucune propriété particulière n'est définie pour les noeuds du type `applyorakmeansnode`.

NMF Oracle

La propriété suivante est disponible pour les noeuds du type `applyoranmfnode` :

Tableau 182. propriétés de applyoranmfnode

Propriétés de `applyoranmfnode`	Valeurs	Description de la propriété
`display_all_features`	*flag*	

Apriori Oracle

Ce nugget de modèle ne peut pas être appliqué dans la génération de script.

MDL Oracle

Ce nugget de modèle ne peut pas être appliqué dans la génération de script.

Propriétés de nœud pour la modélisation IBM DB2

Propriétés du noeud de modélisation IBM DB2

Les propriétés suivantes sont communes aux noeuds de modélisation de base de données IBM InfoSphere Warehouse (ISW).

Tableau 183. Propriétés communes du noeud ISW.

Propriétés communes du noeud ISW	Valeurs	Description de la propriété
inputs	*Liste des champs*	
datasource		
username		
password		
epassword		
enable_power_options	*indicateur*	
power_options_max_memory	*entier*	La valeur doit être supérieure à 32.
power_options_cmdline	*chaîne*	
mining_data_custom_sql	*chaîne*	
logical_data_custom_sql	*chaîne*	
mining_settings_custom_sql		

Arbre décision ISW

Les propriétés suivantes sont disponibles pour les noeuds du type Arbre décision IM DB2.

Tableau 184. propriétés de db2imtreenode.

Propriétés de db2imtreenode	Valeurs	Description de la propriété
target	*champ*	
perform_test_run	*indicateur*	
use_max_tree_depth	*indicateur*	
max_tree_depth	*entier*	Valeur supérieure à 0.
use_maximum_purity	*indicateur*	
maximum_purity	*nombre*	Nombre compris entre 0 et 100.
use_minimum_internal_cases	*indicateur*	
minimum_internal_cases	*entier*	Valeur supérieure à 1.
use_costs	*indicateur*	
costs	*structurée*	Propriété structurée utilisant la syntaxe : [{drugA drugB 1.5} {drugA drugC 2.1}]où les arguments entre {} sont les coûts réels prédits.

Association ISW

Les propriétés suivantes sont disponibles pour les noeuds du type db2imassocnode.

Tableau 185. propriétés de db2imassocnode.

Propriétés de db2imassocnode	Valeurs	Description de la propriété
use_transactional_data	*indicateur*	
id_field	*champ*	
content_field	*champ*	
data_table_layout	basic limited_length	
max_rule_size	*entier*	La valeur doit être supérieure à 2.
min_rule_support	*nombre*	0–100%
min_rule_confidence	*nombre*	0–100%
use_item_constraints	*indicateur*	
item_constraints_type	Include Exclude	
use_taxonomy	*indicateur*	
taxonomy_table_name	*chaîne*	Nom de la table DB2 dans laquelle stocker les détails de la taxonomie.
taxonomy_child_column_name	*chaîne*	Nom de la colonne enfant de la table de taxonomie. La colonne enfant comporte les noms des éléments ou des catégories.
taxonomy_parent_column_name	*chaîne*	Nom de la colonne parent de la table de taxonomie. La colonne parent contient les noms des catégories.
load_taxonomy_to_table	*indicateur*	Permet de savoir si les informations sur la taxonomie stockées dans IBM SPSS Modeler doivent être envoyées à la table de taxonomie lors de la création du modèle. La table de taxonomie est supprimée si elle existe déjà. Les informations sur la taxonomie sont stockées avec le noeud de création du modèle et peuvent être modifiées via les boutons **Modifier les catégories** et **Modifier la taxonomie**.

Séquence ISW

Les propriétés suivantes sont disponibles pour les noeuds du type db2imsequencenode.

Tableau 186. propriétés de db2imsequencenode.

Propriétés de db2imsequencenode	Valeurs	Description de la propriété
id_field	*champ*	
group_field	*champ*	
content_field	*champ*	
max_rule_size	*entier*	La valeur doit être supérieure à 2.
min_rule_support	*nombre*	0–100%
min_rule_confidence	*nombre*	0–100%
use_item_constraints	*indicateur*	
item_constraints_type	Include Exclude	
use_taxonomy	*indicateur*	
taxonomy_table_name	*chaîne*	Nom de la table DB2 dans laquelle stocker les détails de la taxonomie.

Tableau 186. propriétés de db2imsequencenode (suite).

Propriétés de db2imsequencenode	Valeurs	Description de la propriété
taxonomy_child_column_name	*chaîne*	Nom de la colonne enfant de la table de taxonomie. La colonne enfant comporte les noms des éléments ou des catégories.
taxonomy_parent_column_name	*chaîne*	Nom de la colonne parent de la table de taxonomie. La colonne parent contient les noms des catégories.
load_taxonomy_to_table	*indicateur*	Permet de savoir si les informations sur la taxonomie stockées dans IBM SPSS Modeler doivent être envoyées à la table de taxonomie lors de la création du modèle. La table de taxonomie est supprimée si elle existe déjà. Les informations sur la taxonomie sont stockées avec le noeud de création du modèle et peuvent être modifiées via les boutons **Modifier les catégories** et **Modifier la taxonomie**.

Régression ISW

Les propriétés suivantes sont disponibles pour les noeuds du type db2imregnode.

Tableau 187. propriétés de db2imregnode.

Propriétés de db2imregnode	Valeurs	Description de la propriété
target	*champ*	
regression_method	transform linear polynomial rbf	Consultez le tableau suivant pour les propriétés s'appliquant uniquement si regression_method est définie sur rbf.
perform_test_run	*champ*	
limit_rsquared_value	*indicateur*	
max_rsquared_value	*nombre*	Valeur comprise entre 0,0 et 1,0.
use_execution_time_limit	*indicateur*	
execution_time_limit_mins	*entier*	Valeur supérieure à 0.
use_max_degree_polynomial	*indicateur*	
max_degree_polynomial	*entier*	
use_intercept	*indicateur*	
use_auto_feature_selection_method	*indicateur*	
auto_feature_selection_method	normal adjusted	
use_min_significance_level	*indicateur*	
min_significance_level	*nombre*	
use_min_significance_level	*indicateur*	

Les propriétés suivantes ne s'appliquent que si regression_method est définie sur rbf.

Tableau 188. Propriétés db2imregnode si regression_method est définie sur rbf.

Propriétés de db2imregnode	Valeurs	Description de la propriété
use_output_sample_size	*indicateur*	Si true (vrai), définissez automatiquement la valeur sur la valeur par défaut.

Tableau 188. Propriétés db2imregnode si regression_method est définie sur rbf (suite).

output_sample_size	*entier*	La valeur par défaut est 2. Le minimum est 1.
use_input_sample_size	*indicateur*	Si true (vrai), définissez automatiquement la valeur sur la valeur par défaut.
input_sample_size	*entier*	La valeur par défaut est 2. Le minimum est 1.
use_max_num_centers	*indicateur*	Si true (vrai), définissez automatiquement la valeur sur la valeur par défaut.
max_num_centers	*entier*	La valeur par défaut est 20. Le minimum est 1.
use_min_region_size	*indicateur*	Si true (vrai), définissez automatiquement la valeur sur la valeur par défaut.
min_region_size	*entier*	La valeur par défaut est 15. Le minimum est 1.
use_max_data_passes	*indicateur*	Si true (vrai), définissez automatiquement la valeur sur la valeur par défaut.
max_data_passes	*entier*	La valeur par défaut est 5. Le minimum est 2.
use_min_data_passes	*indicateur*	Si true (vrai), définissez automatiquement la valeur sur la valeur par défaut.
min_data_passes	*entier*	La valeur par défaut est 5. Le minimum est 2.

Classification ISW

Les propriétés suivantes sont disponibles pour les noeuds du type db2imclusternode.

Tableau 189. propriétés de db2imclusternode.

Propriétés de db2imclusternode	**Valeurs**	**Description de la propriété**
cluster_method	demographic kohonen birch	
kohonen_num_rows	*entier*	
kohonen_num_columns	*entier*	
kohonen_passes	*entier*	
use_num_passes_limit	*indicateur*	
use_num_clusters_limit	*indicateur*	
max_num_clusters	*entier*	Valeur supérieure à 1.
birch_dist_measure	log_likelihood euclidean	La valeur par défaut est log_likelihood.
birch_num_cfleaves	*entier*	La valeur par défaut est 1000.
birch_num_refine_passes	*entier*	La valeur par défaut est 3, la valeur minimum est 1.

Tableau 189. propriétés de db2imclusternode (suite).

Propriétés de db2imclusternode	Valeurs	Description de la propriété
use_execution_time_limit	*indicateur*	
execution_time_limit_mins	*entier*	Valeur supérieure à 0.
min_data_percentage	*nombre*	0–100%
use_similarity_threshold	*indicateur*	
similarity_threshold	*nombre*	Valeur comprise entre 0,0 et 1,0.

Naive Bayes ISW

Les propriétés suivantes sont disponibles pour les noeuds du type db2imnbsnode.

Tableau 190. propriétés de db2imnbnode.

Propriétés de db2imnbnode	Valeurs	Description de la propriété
perform_test_run	*indicateur*	
probability_threshold	*nombre*	La valeur par défaut est 0.001. La valeur minimum est de 0 ; la valeur maximum est de 1,000.
use_costs	*indicateur*	
costs	*structurée*	Propriété structurée utilisant la syntaxe : [{drugA drugB 1.5} {drugA drugC 2.1}]où les arguments entre {} sont les coûts réels prédits.

Régression logistique ISW

Les propriétés suivantes sont disponibles pour les noeuds du type db2imlognode.

Tableau 191. propriétés de db2imlognode.

Propriétés de db2imlognode	Valeurs	Description de la propriété
perform_test_run	*indicateur*	
use_costs	*indicateur*	
costs	*structurée*	Propriété structurée utilisant la syntaxe : [{drugA drugB 1.5} {drugA drugC 2.1}]où les arguments entre {} sont les coûts réels prédits.

Séries temporelles ISW

Remarque : le paramètre des champs d'entrée n'est pas utilisé pour ce noeud. Si le paramètre des champs d'entrée se trouve dans le script, un avertissement apparaît et signale que le noeud contient les champs entrants *heure* et *cibles* mais pas de champs d'entrée.

Les propriétés suivantes sont disponibles pour les noeuds du type db2imtimeseriesnode.

Tableau 192. propriétés de db2imtimeseriesnode.

Propriétés de db2imtimeseriesnode	Valeurs	Description de la propriété
time	*champ*	Entier, heure ou date autorisé.
targets	*liste des champs*	

Tableau 192. propriétés de db2imtimeseriesnode (suite).

Propriétés de db2imtimeseriesnode	Valeurs	Description de la propriété
forecasting_algorithm	arima exponential_ smoothing seasonal_trend_ decomposition	
forecasting_end_time	auto integer date time	
use_records_all	*booléen*	Si false (faux), use_records_start et use_records_end doivent être définis.
use_records_start	*entier / heure / date*	Dépend du type du champ Temps
use_records_end	*entier / heure / date*	Dépend du type du champ Temps
interpolation_method	none linear exponential_splines cubic_splines	

Propriétés du nugget de modèle IBM DB2

Les propriétés suivantes s'appliquent aux nuggets de modèle créés à l'aide des modèles IBM DB2 ISW.

Arbre décision ISW

Aucune propriété particulière n'est définie pour les noeuds du type applydb2imtreenode.

Association ISW

Ce nugget de modèle ne peut pas être appliqué dans la génération de script.

Séquence ISW

Ce nugget de modèle ne peut pas être appliqué dans la génération de script.

Régression ISW

Aucune propriété particulière n'est définie pour les noeuds du type applydb2imregnode.

Classification ISW

Aucune propriété particulière n'est définie pour les noeuds du type applydb2imclusternode.

Naive Bayes ISW

Aucune propriété particulière n'est définie pour les noeuds du type applydb2imnbnode.

Régression logistique ISW

Aucune propriété particulière n'est définie pour les noeuds du type applydb2imlognode.

Séries temporelles ISW

Ce nugget de modèle ne peut pas être appliqué dans la génération de script.

Propriétés du noeud pour la modélisation IBM Netezza Analytics

Propriétés des nœuds de modélisation Netezza

Les propriétés suivantes sont communes aux noeuds de modélisation de base de données IBM Netezza.

Tableau 193. Propriétés communes aux noeuds Netezza.

Propriétés communes des noeuds Netezza	Valeurs	Description de la propriété
custom_fields	*indicateur*	Si la valeur est définie sur true (vrai), elle permet de spécifier les champs cible, entrée et d'autres champs pour le nœud actuel. Si elle est définie sur false (faux), les paramètres actuels provenant d'un noeud type en amont sont utilisés.
inputs	*[champ1 ... champN]*	Champs d'entrée ou prédicteur utilisés par le modèle.
target	*champ*	Champ cible (continu ou qualitatif).
record_id	*champ*	Champ à utiliser comme identifiant d'enregistrement unique.
use_upstream_connection	*indicateur*	Si true (vrai) (valeur par défaut), les détails de connexion spécifiés dans un noeud en amont. Non utilisé si move_data_to_connection est spécifié.
move_data_connection	*indicateur*	Si true (vrai), déplace les données vers la base de données mentionnée par connection. Non utilisé si use_upstream_connection est spécifié.
connection	*structurée*	La chaîne de connexion à la base de données Netezza où le modèle est stocké. Propriété structurée utilisant la syntaxe : ['odbc' '<dsn>' '<username>' '<psw>' '<catname>' '<conn_attribs>' {true|false}] où : <dsn> est le nom de la source de données <username> et <psw> sont le nom d'utilisateur et le mot de passe de la base de données <catname> est le nom du catalogue <conn_attribs> sont les attributs de connexion true | false indique si un mot de passe est requis.
table_name	*chaîne*	Le nom de la table de la base de données où le modèle est stocké.
use_model_name	*indicateur*	Si true (vrai), utilise le nom spécifié par model_name comme nom du modèle ; sinon, le nom du modèle est créé par le système.
model_name	*chaîne*	Nom personnalisé du nouveau modèle.
include_input_fields	*indicateur*	Si true (vrai), passe tous les champs d'entrée en aval ; sinon, passe uniquement record_id et les champs générés par le modèle.

Arbre décision Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type netezzadectreenode.

Tableau 194. propriétés de netezzadectreenode.

Propriétés de netezzadectreenode	Valeurs	Description de la propriété
impurity_measure	Entropy Gini	La mesure d'impureté, utilisée pour évaluer le meilleur endroit où diviser l'arbre.
max_tree_depth	*entier*	Nombre de niveaux maximum que l'arbre peut atteindre. La valeur par défaut est 62 (le nombre maximum possible).
min_improvement_splits	*nombre*	Amélioration minimum de l'impureté pour que la division ait lieu. La valeur par défaut est de 0,01.
min_instances_split	*entier*	Nombre minimum d'enregistrements non divisés restants avant que la division n'ait lieu. La valeur par défaut est 2 (le nombre minimum possible).
weights	*structurée*	Pondérations relatives des classes. Propriété structurée utilisant la syntaxe : set :netezza_dectree.weights = [{drugA 0.3}{drugB 0.6}] La valeur par défaut est une pondération de 1 pour toutes les classes.
pruning_measure	Acc wAcc	La valeur par défaut est Acc (exactitude). wAcc (exactitude pondérée) tient compte des pondérations de classe tout en appliquant l'élagage.
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	La valeur par défaut est d'utiliser allTrainingData pour estimer l'exactitude du modèle. Utilisez partitionTrainingData pour spécifier un pourcentage de données d'apprentissage à utiliser ou useOtherTable pour utiliser un jeu de données d'apprentissage d'une table de base de données spécifique.
perc_training_data	*nombre*	Si prune_tree_options est définie sur partitionTrainingData, spécifie le pourcentage de données à utiliser pour l'apprentissage.
prune_seed	*entier*	Valeur de départ aléatoire à utiliser pour dupliquer les résultats d'analyse quand prune_tree_options est définie sur partitionTrainingData ; la valeur par défaut est 1.
pruning_table	*chaîne*	Nom de la table d'un le jeu de données d'élagage distinct pour estimer l'exactitude du modèle.
compute_probabilities	*indicateur*	Si true (vrai), produit un champ de niveau de confiance (probabilité) ainsi que le champ de prévision.

k moyenne Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type `netezzakmeansnode`.

Tableau 195. propriétés de netezzakmeansnode.

Propriétés de `netezzakmeansnode`	Valeurs	Description de la propriété

`distance_measure`	`Euclidean`
	`Manhattan`
	`Canberra`
	`maximum`
		Méthode à utiliser pour mesurer la distance entre les points de données.

`num_clusters`	*entier*	Le nombre de clusters à créer ; la valeur par défaut est 3.
`max_iterations`	*entier*	Le nombre d'itérations d'algorithme après lequel arrêter l'apprentissage du modèle ; la valeur par défaut est 5.
`rand_seed`	*entier*	Valeur de départ aléatoire à utiliser pour dupliquer les résultats d'analyse ; la valeur par défaut est 12345.

Bayes Net Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type `netezzabayesnode`.

Tableau 196. propriétés de netezzabayesnode.

Propriétés de `netezzabayesnode`	Valeurs	Description de la propriété
`base_index`	*entier*	Identifiant numérique attribué au premier champ d'entrée pour la gestion interne ; la valeur par défaut est 777.
`sample_size`	*entier*	Taille de l'échantillon à prélever si le nombre d'attributs est très grand ; la valeur par défaut est 10 000.
`display_additional_information`	*indicateur*	Si true (vrai), affiche les informations de progression supplémentaires dans une boîte de dialogue.

`type_of_prediction`	`best`
	`neighbors`
	`nn-neighbors`
		Type d'algorithme de prévision à utiliser : best (meilleur ; voisin le plus corrélé), neighbors (voisins ; prévision pondérée des voisins) ou nn-neighbors (voisins nn ; voisins non nuls).

Naive Bayes Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type `netezzanaivebayesnode`.

Tableau 197. propriétés de netezzanaivebayesnode.

Propriétés de `netezzanaivebayesnode`	Valeurs	Description de la propriété
`compute_probabilities`	*indicateur*	Si true (vrai), produit un champ de niveau de confiance (probabilité) ainsi que le champ de prévision.
`use_m_estimation`	*indicateur*	Si true (vrai), utilise la technique de m-estimation pour éviter les probabilités zéro pendant l'estimation.

KNN Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type `netezzaknnnode`.

Tableau 198. propriétés de netezzaknnnode.

Propriétés de netezzaknnnode	Valeurs	Description de la propriété
weights	*structurée*	Propriété structurée utilisée pour attribuer des pondérations aux classes individuelles. Exemple : set :netezzaknnnode.weights = [{drugA 0.3}{drugB 0.6}]
distance_measure	Euclidean Manhattan Canberra Maximum	Méthode à utiliser pour mesurer la distance entre les points de données.
num_nearest_neighbors	*entier*	Nombre de voisins les plus proches pour un cas particulier ; la valeur par défaut est 3.
standardize_measurements	*indicateur*	Si true (vrai), standardise les mesures des champs d'entrée continus avant de calculer les valeurs de distance.
use_coresets	*indicateur*	Si true (vrai), utilise l'échantillonnage d'ensembles principaux pour accélérer le calcul de grands jeux de données.

Classification par division Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type netezzadivclusternode.

Tableau 199. propriétés de netezzadivclusternode.

Propriétés de netezzadivclusternode	Valeurs	Description de la propriété
distance_measure	Euclidean Manhattan Canberra Maximum	Méthode à utiliser pour mesurer la distance entre les points de données.
max_iterations	*entier*	Nombre maximum d'itérations d'algorithme à effectuer avant l'arrêt de l'apprentissage du modèle ; la valeur par défaut est 5.
max_tree_depth	*entier*	Nombre maximum de niveaux dans lesquels les jeux de données peuvent être divisés ; la valeur par défaut est 3.
rand_seed	*entier*	Valeur de départ aléatoire utilisée pour dupliquer les analyses ; la valeur par défaut est 12345.
min_instances_split	*entier*	Nombre minimum d'enregistrements pouvant être divisés, la valeur par défaut est 5.
level	*entier*	Niveau hiérarchique selon lequel les enregistrements sont évalués ; la valeur par défaut est -1.

ACP Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type netezzapcanode.

Tableau 200. propriétés de netezzapcanode.

Propriétés de netezzapcanode	Valeurs	Description de la propriété
center_data	*indicateur*	Si true (vrai) (par défaut), effectue un centrage des données (également nommé « soustraction moyenne ») avant l'analyse.

Tableau 200. propriétés de netezzapcanode (suite).

Propriétés de netezzapcanode	Valeurs	Description de la propriété
perform_data_scaling	*indicateur*	Si true (vrai), effectue la mise à l'échelle des données avant l'analyse. Ceci peut rendre l'analyse moins arbitraire quand différentes variables sont mesurées dans différentes unités.
force_eigensolve	*indicateur*	Si true (vrai), utilise une méthode moins précise mais plus rapide pour trouver les composantes principales.
pc_number	*entier*	Nombre de composantes principales auquel le jeu de données doit être réduit ; la valeur par défaut est 1.

Arbre de régression Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type netezzaregtreenode.

Tableau 201. propriétés de netezzaregtreenode.

Propriétés de netezzaregtreenode	Valeurs	Description de la propriété
max_tree_depth	*entier*	Nombre maximum de niveaux auxquels l'arbre peut grandir sous le noeud racine ; la valeur par défaut est 10.
split_evaluation_measure	Variance	Mesure d'impureté de la classe, utilisée pour évaluer le meilleur endroit pour diviser l'arbre ; la valeur par défaut (et actuellement la seule option) est Variance.
min_improvement_splits	*nombre*	Quantité minimale pour réduire l'impureté avant de créer une nouvelle division dans l'arbre.
min_instances_split	*entier*	Le nombre minimal d'enregistrements pouvant être divisés.
pruning_measure	mse r2 pearson spearman	Méthode à utiliser pour l'élagage.
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	La valeur par défaut est d'utiliser allTrainingData pour estimer l'exactitude du modèle. Utilisez partitionTrainingData pour spécifier un pourcentage de données d'apprentissage à utiliser ou useOtherTable pour utiliser un jeu de données d'apprentissage d'une table de base de données spécifique.
perc_training_data	*nombre*	Si prune_tree_options est définie sur PercTrainingData, spécifie le pourcentage de données à utiliser pour l'apprentissage.
prune_seed	*entier*	Valeur de départ aléatoire à utiliser pour dupliquer les résultats d'analyse quand prune_tree_options est définie sur PercTrainingData ; la valeur par défaut est 1.

Tableau 201. propriétés de netezzaregtreenode (suite).

Propriétés de netezzaregtreenode	Valeurs	Description de la propriété
pruning_table	*chaîne*	Nom de la table d'un le jeu de données d'élagage distinct pour estimer l'exactitude du modèle.
compute_probabilities	*indicateur*	Si true (vrai), spécifie que les variances des classes attribuées doivent être incluses dans la sortie.

Régression linéaire Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type netezzalineregressionnode.

Tableau 202. propriétés netezzalineregressionnode.

Propriétés de netezzalineregressionnode	Valeurs	Description de la propriété
use_svd	*indicateur*	Si true (vrai), utilise la matrice de décomposition en valeurs singulières à la place de la matrice d'origine, pour améliorer la vitesse et l'exactitude numérique.
include_intercept	*indicateur*	Si true (vrai) (défaut), augmente l'exactitude générale de la solution.
calculate_model_diagnostics	*indicateur*	Si true (vrai), calcule le diagnostic sur le modèle.

Séries temporelles Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type netezzatimeseriesnode.

Tableau 203. propriétés netezzatimeseriesnode.

Propriétés de netezzatimeseriesnode	Valeurs	Description de la propriété
time_points	*champ*	Champ d'entrée contenant les valeurs de date et d'heure des séries temporelles.
time_series_ids	*champ*	Un champ d'entrée contenant des ID de séries temporelles à utiliser si l'entrée contient plusieurs séries temporelles.
model_table	*champ*	Nom de la table de la base de données dans laquelle le modèle de séries temporelles Netezza sera stocké.
description_table	*champ*	Nom de la table d'entrée qui contient les noms et descriptions des séries temporelles.
seasonal_adjustment_table	*champ*	Nom de la table de sortie dans laquelle les valeurs ajustées de manière saisonnière calculées par des algorithmes de lissage exponentiel ou de décomposition des tendances saisonnières seront stockées.

Tableau 203. propriétés netezzatimeseriesnode (suite).

Propriétés de **netezzatimeseriesnode**	Valeurs	Description de la propriété
algorithm_name	SpectralAnalysis ou spectral ExponentialSmoothing ou esmoothing ARIMA SeasonalTrendDecomposition ou std	Algorithme à utiliser pour la modélisation des séries temporelles.
trend_name	N A DA M DM	Type de tendance du lissage exponentiel : N - none (aucun) A - additive (additif) DA - damped additive (additif amorti) M - multiplicative (multiplicatif) DM - damped multiplicative (Multiplicatif amorti)
seasonality_type	N A M	Type de saisonnalité pour le lissage exponentiel : N - none (aucun) A - additive (additif) M - multiplicative (multiplicatif)
interpolation_method	linear cubicspline exponentialspline	Méthode d'interpolation à utiliser.
timerange_setting	ET SP	Paramètre pour l'intervalle de temps à utiliser : SD - system-determined (déterminé par le système) (utilise toute la plage de séries temporelles) SP - spécifié par l'utilisateur via earliest_time et latest_time

earliest_time
latest_time
	entier
	date
	heure
	timestamp
		Valeurs de début et de fin, si timerange_setting est SP.
		
		Le format doit suivre la valeur de time_points.
		
		Par exemple, si le champ time_points contient une date, cette valeur doit également être une date.
		
		Exemple :
		`set NZ_DT1.timerange_setting = 'SP'`
		`set NZ_DT1.earliest_time = '1921-01-01'`
		`set NZ_DT1.latest_time = '2121-01-01'`

Tableau 203. propriétés netezzatimeseriesnode (suite).

Propriétés de **netezzatimeseriesnode**	Valeurs	Description de la propriété
arima_setting	ET SP	Paramètre pour l'algorithme ARIMA (utilisé uniquement si algorithm_name est défini sur ARIMA) : SD - system-determined (déterminé par le système) SP - spécifié par l'utilisateur Si arima_setting = SP, utilisez les paramètres suivants pour définir les valeurs saisonnières et non saisonnières. Exemple (non saisonnier uniquement) : set NZ_DT1.algorithm_name = 'arima' set NZ_DT1.arima_setting = 'SP' set NZ_DT1.p_symbol = 'lesseq' set NZ_DT1.p = '4' set NZ_DT1.d_symbol = 'lesseq' set NZ_DT1.d = '2' set NZ_DT1.q_symbol = 'lesseq' set NZ_DT1.q = '4'
p_symbol d_symbol q_symbol sp_symbol sd_symbol sq_symbol	less eq lesseq	ARIMA - opérateur pour les paramètres p, d, q, sp, sd et sq : less - inférieur à eq - égal à lesseq - inférieur ou égal à
p	*entier*	ARIMA - degrés d'autocorrélation non saisonniers.
q	*entier*	ARIMA - valeur de calcul non saisonnier.
d	*entier*	ARIMA - nombre d'ordres de moyenne mobile non saisonniers dans le modèle.
sp	*entier*	ARIMA - degrés d'autocorrélation saisonniers.
sq	*entier*	ARIMA - valeur de dérivation saisonnière.
sd	*entier*	ARIMA - nombre d'ordres de moyenne mobile saisonniers dans le modèle.

Tableau 203. propriétés netezzatimeseriesnode (suite).

Propriétés de **netezzatimeseriesnode**	Valeurs	Description de la propriété
advanced_setting	ET SP	Détermine comment traiter les paramètres avancés : SD - system-determined (déterminé par le système) SP - spécifié par l'utilisateur via period, units_period et forecast_setting. Exemple : set NZ_DT1.advanced_setting = 'SP' set NZ_DT1.period = 5 set NZ_DT1.units_period = 'd'
period	*entier*	Longueur du cycle saisonnier spécifié en association avec units_period. Ne s'applique pas à l'analyse spectrale.
units_period	ms s min h d wk q y	Unités dans lesquelles period est exprimé : ms - millisecondes s - secondes min - minutes h - heures d - jours wk - semaines q - trimestres y - années Par exemple, pour une série temporelle hebdomadaire, utilisez 1 pour period et wk pour units_period.
forecast_setting	forecasthorizon forecasttimes	Indique comment doivent être effectuées les prévisions.
forecast_horizon	*entier* *date* *heure* *horodatage*	Si forecast_setting = forecasthorizon, indique la valeur de point final des prévisions. Le format doit suivre la valeur de time_points. Par exemple, si le champ time_points contient une date, cette valeur doit également être une date.
forecast_times	*entier* *date* *heure* *horodatage*	Si forecast_setting = forecasttimes, indique les valeurs à utiliser pour les prévisions. Le format doit suivre la valeur de time_points. Par exemple, si le champ time_points contient une date, cette valeur doit également être une date.
include_history	*indicateur*	Indique si les valeurs historiques doivent être incluses dans la sortie.

Tableau 203. propriétés netezzatimeseriesnode (suite).

Propriétés de netezzatimeseriesnode	Valeurs	Description de la propriété
include_interpolated_values	*indicateur*	Indique si les valeurs interpolées doivent être incluses dans la sortie. Ne s'applique pas si include_history est false.

Linéaire généralisé Netezza

Les propriétés suivantes sont disponibles pour les noeuds du type netezzaglmnode.

Tableau 204. propriétés netezzaglmnode.

Propriétés de netezzaglmnode	Valeurs	Description de la propriété
dist_family	bernoulli gaussian poisson negativebinomial wald gamma	Type de distribution ; le type par défaut est bernoulli.
dist_params	*nombre*	Valeur de paramètre de distribution à utiliser. S'applique uniquement si distribution est Negativebinomial.
trials	*entier*	S'applique uniquement si distribution est Binomial. Lorsque la réponse cible est un nombre d'événements se produisant dans un ensemble d'essais, le champ target contient le nombre d'événements et le champ trials contient le nombre d'essais.
model_table	*champ*	Nom de la table de la base de données dans laquelle le modèle linéaire généralisé Netezza sera stocké.
maxit	*entier*	Nombre maximum d'itérations que l'algorithme doit exécuter ; la valeur par défaut est de 20.
eps	*nombre*	Valeur du nombre maximal d'erreurs (en notation scientifique) à laquelle l'algorithme doit s'arrêter de rechercher le modèle le plus adapté. La valeur par défaut est de -3, soit 1E-3 ou 0,001.
tol	*nombre*	Valeur (en notation scientifique) sous laquelle les erreurs sont traitées comme ayant une valeur de zéro. La valeur par défaut est de -7, ce qui signifie que les valeurs d'erreurs inférieures à 1E-7 (ou 0,0000001) sont considérées comme non significatives.

Tableau 204. propriétés netezzaglmnode (suite).

Propriétés de netezzaglmnode	Valeurs	Description de la propriété
link_func	identity inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	Fonction de lien à utiliser ; la fonction par défaut est logit.
link_params	*nombre*	Valeur de paramètre de fonction de lien à utiliser. S'applique uniquement si link_function est power ou oddspower.
interaction	[{[*colnames1*],[*levels1*]}, {[*colnames2*],[*levels2*]}, ...,{[*colnamesN*],[*levelsN*]},]	Spécifie des interactions entre les champs. *colnames* est une liste de champs d'entrée et *level* est toujours 0 pour chaque champ. Exemple : [{["K","BP","Sex","K"],[0,0,0,0]}, {["Age","Na"],[0,0]}]
intercept	*indicateur*	Si true, inclut la constante dans le modèle.

Propriétés du nugget de modèle Netezza

Les propriétés suivantes sont communes aux noeuds de modèle de base de données Netezza.

Tableau 205. Propriétés communes du nugget de modèle Netezza

Propriétés communes du nugget de modèle Netezza	Valeurs	Description de la propriété
connexion	*chaîne*	La chaîne de connexion à la base de données Netezza où le modèle est stocké.
table_name	*chaîne*	Le nom de la table de la base de données où le modèle est stocké.

Les autres propriétés du nugget de modèle sont les mêmes que celles du noeud de modélisation correspondant.

Les noms de script des nuggets de modèle sont les suivants.

Tableau 206. Noms de script des nuggets de modèle Netezza

Nugget de modèle	**Nom du script**
Arbre de décision	`applynetezzadectreenode`
k moyenne	`applynetezzakmeansnode`
Bayes Net	`applynetezzabayesnode`
Naive Bayes	`applynetezzanaivebayesnode`
KNN	`applynetezzaknnnode`
Classification par division	`applynetezzadivclusternode`
ACP	`applynetezzapcanode`
Arbre de régression	`applynetezzaregtreenode`
Régression linéaire	`applynetezzalineregressionnode`
Séries temporelles	`applynetezzatimeseriesnode`
Linéaire généralisé	`applynetezzaglmnode`

Chapitre 16. Propriétés des noeuds de sortie

Les propriétés de nœud de sortie diffèrent légèrement de celles des autres types de nœud. Au lieu de faire référence à une option de nœud particulière, les propriétés du nœud de sortie stockent une référence à l'objet de sortie. Cela peut être utile lorsque vous choisissez une valeur dans un tableau pour la définir en tant que paramètre de flux.

Cette section décrit les propriétés de génération de scripts disponibles pour les noeuds de sortie.

Propriétés de analysisnode

Le noeud Analyse évalue la capacité des modèles prédictifs à générer des prévisions précises. Les noeuds Analyse comparent les valeurs prédites et les valeurs réelles d'un ou de plusieurs nuggets de modèle. Ils peuvent également comparer entre eux les modèles prédictifs.

Exemple

```
node = stream.create("analysis", "My node")
# Onglet Analyse
node.setPropertyValue("coincidence", True)
node.setPropertyValue("performance", True)
node.setPropertyValue("confidence", True)
node.setPropertyValue("threshold", 75)
node.setPropertyValue("improve_accuracy", 3)
node.setPropertyValue("inc_user_measure", True)
# Définir les mesures de l'utilisateur
node.setPropertyValue("user_if", "@TARGET = @PREDICTED")
node.setPropertyValue("user_then", "101")
node.setPropertyValue("user_else", "1")
node.setPropertyValue("user_compute", ["Mean", "Sum"])
node.setPropertyValue("by_fields", ["Drug"])
# Onglet Sortie
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/analysis_out.html")
```

Tableau 207. propriétés de analysisnode.

Propriétés de analysisnode	Le type de données	Description de la propriété
output_mode	Screen File	Permet d'indiquer l'emplacement cible pour les sorties générées à partir du noeud de sortie.
use_output_name	*indicateur*	Indique si un nom de sortie personnalisé est utilisé.
output_name	*chaîne*	Si use_output_name a la valeur true (vrai), indique le nom à utiliser.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Permet d'indiquer le type de sortie.
by_fields	*liste (list)*	
full_filename	*chaîne*	Si vous indiquez Disque, Données ou HTML : nom du fichier de sortie utilisé.

Tableau 207. propriétés de analysisnode (suite).

Propriétés de analysisnode	Le type de données	Description de la propriété
coincidence	*indicateur*	
performance	*indicateur*	
evaluation_binary	*flag*	
confidence	*indicateur*	
seuil	*nombre*	
improve_accuracy	*nombre*	
inc_user_measure	*indicateur*	
user_if	*expression*	
user_then	*expression*	
user_else	*expression*	
user_compute	[Mean Sum Min Max SDev]	

Propriétés de dataauditnode

Le noeud Audit données fournit un premier aperçu complet des données, notamment des statistiques récapitulatives, des histogrammes et distributions pour chaque champ, ainsi que des informations sur les valeurs éloignées, les valeurs manquantes et les valeurs extrêmes. Les résultats sont affichés dans une matrice facile à lire pouvant être triée et utilisée pour générer les noeuds de préparation des données et des graphiques grandeur nature.

Exemple

```
filenode = stream.createAt("variablefile", "File", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("dataaudit", "My node", 196, 100)
stream.link(filenode, node)
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("fields", ["Age", "Na", "K"])
node.setPropertyValue("display_graphs", True)
node.setPropertyValue("basic_stats", True)
node.setPropertyValue("advanced_stats", True)
node.setPropertyValue("median_stats", False)
node.setPropertyValue("calculate", ["Count", "Breakdown"])
node.setPropertyValue("outlier_detection_method", "std")
node.setPropertyValue("outlier_detection_std_outlier", 1.0)
node.setPropertyValue("outlier_detection_std_extreme", 3.0)
node.setPropertyValue("output_mode", "Screen")
```

Tableau 208. propriétés de dataauditnode.

Propriétés de dataauditnode	Le type de données	Description de la propriété
custom_fields	*indicateur*	
fields	*[champ1 ... champN]*	
overlay	*champ*	
display_graphs	*indicateur*	Permet d'activer ou de désactiver l'affichage des graphiques dans la matrice de sortie.
basic_stats	*indicateur*	

Tableau 208. propriétés de dataauditnode (suite).

Propriétés de **dataauditnode**	Le type de données	Description de la propriété
advanced_stats	*indicateur*	
median_stats	*indicateur*	
calculate	Count Breakdown	Utilisée pour calculer les valeurs manquantes. Sélectionnez une méthode de calcul, les deux ou bien aucune.
outlier_detection_method	std iqr	Utilisé pour indiquer la méthode de détection des valeurs éloignées et des valeurs extrêmes.
outlier_detection_std_outlier	*nombre*	Si outlier_detection_method est défini sur std, indique le nombre à utiliser pour définir les valeurs éloignées.
outlier_detection_std_extreme	*nombre*	Si outlier_detection_method est défini sur std, indique le nombre à utiliser pour définir les valeurs extrêmes.
outlier_detection_iqr_outlier	*nombre*	Si outlier_detection_method est défini sur iqr, indique le nombre à utiliser pour définir les valeurs éloignées.
outlier_detection_iqr_extreme	*nombre*	Si outlier_detection_method est défini sur iqr, indique le nombre à utiliser pour définir les valeurs extrêmes.
use_output_name	*indicateur*	Indique si un nom de sortie personnalisé est utilisé.
output_name	*chaîne*	Si use_output_name a la valeur true (vrai), indique le nom à utiliser.
output_mode	Screen File	Permet d'indiquer l'emplacement cible pour les sorties générées à partir du noeud de sortie.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Permet d'indiquer le type de sortie.
paginate_output	*indicateur*	Lorsque output_format est défini sur HTML, cette propriété divise la sortie en pages.
lines_per_page	*nombre*	Lorsque cette propriété est utilisée avec paginate_output, elle indique le nombre de lignes par page de sortie.
full_filename	*chaîne*	

Propriétés de matrixnode

Le noeud Matrice permet de créer un tableau dans lequel les relations entre les champs sont indiquées. Il s'agit généralement de deux champs symboliques, mais il peut également s'agir de champs indicateurs ou numériques.

Exemple

```
node = stream.create("matrix", "My node")
#  Onglet Paramètres
node.setPropertyValue("fields", "Numerics")
node.setPropertyValue("row", "K")
node.setPropertyValue("column", "Na")
node.setPropertyValue("cell_contents", "Function")
node.setPropertyValue("function_field", "Age")
node.setPropertyValue("function", "Sum")
#  Onglet Apparence
node.setPropertyValue("sort_mode", "Ascending")
node.setPropertyValue("highlight_top", 1)
node.setPropertyValue("highlight_bottom", 5)
node.setPropertyValue("display", ["Counts", "Expected", "Residuals"])
node.setPropertyValue("include_totals", True)
# Onglet Sortie
node.setPropertyValue("full_filename", "C:/output/matrix_output.html")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tableau 209. propriétés de matrixnode.

Propriétés de matrixnode	Le type de données	Description de la propriété
fields	sélectionnées Flags Numérique	
row	*champ*	
column	*champ*	
include_missing_values	*indicateur*	Indique si les valeurs utilisateur (blancs) et système (valeurs nulles) manquantes sont incluses dans la sortie des lignes et des colonnes.
cell_contents	CrossTabs Function	
function_field	*chaîne*	
function	Sum Mean Min Max SDev	
sort_mode	Unsorted Ascending Descending	
highlight_top	*nombre*	Si non égal à zéro, alors True (vrai).
highlight_bottom	*nombre*	Si non égal à zéro, alors True (vrai).

Tableau 209. propriétés de matrixnode (suite).

Propriétés de matrixnode	Le type de données	Description de la propriété
display	[Counts Expected Résiduels RowPct ColumnPct TotalPct]	
include_totals	*indicateur*	
use_output_name	*indicateur*	Indique si un nom de sortie personnalisé est utilisé.
output_name	*chaîne*	Si use_output_name a la valeur true (vrai), indique le nom à utiliser.
output_mode	Screen File	Permet d'indiquer l'emplacement cible pour les sorties générées à partir du noeud de sortie.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Permet d'indiquer le type de sortie. Les formats Formatted et Delimited peuvent utiliser le modificateur transposed, qui transpose les lignes et les colonnes dans le tableau.
paginate_output	*indicateur*	Lorsque output_format est défini sur HTML, cette propriété divise la sortie en pages.
lines_per_page	*nombre*	Lorsque cette propriété est utilisée avec paginate_output, elle indique le nombre de lignes par page de sortie.
full_filename	*chaîne*	

Propriétés de meansnode

Le noeud Moyennes compare les moyennes de groupes indépendants ou de paires de champs associés, afin de détecter toute différence sensible. Par exemple, vous pouvez comparer les revenus moyens avant et après l'application d'une augmentation, ou comparer les revenus des personnes ayant obtenu une augmentation avec ceux des personnes qui n'en ont pas eu.

Exemple

```
node = stream.create("means", "My node")
node.setPropertyValue("means_mode", "BetweenFields")
node.setPropertyValue("paired_fields", [["OPEN_BAL", "CURR_BAL"]])
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("output_view", "Advanced")
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/means_output.html")
```

Tableau 210. propriétés de meansnode.

Propriétés de meansnode	Le type de données	Description de la propriété
means_mode	BetweenGroups BetweenFields	Indique le type de statistique de moyenne à exécuter sur les données.

Tableau 210. propriétés de meansnode (suite).

Propriétés de meansnode	Le type de données	Description de la propriété
test_fields	[champ1 ... champN]	Indique le champ de test lorsque means_mode est paramétré sur BetweenGroups.
grouping_field	*champ*	Indique le champ de regroupement.
paired_fields	[{field1 field2} {field3 field4} ...]	Indique les paires de champs à utiliser lorsque means_mode est paramétré sur BetweenFields.
label_correlations	*indicateur*	Indique si les libellés de corrélation apparaissent dans la sortie. Ce paramètre ne s'applique que si means_mode est paramétré sur BetweenFields.
correlation_mode	Probabilité Absolute	Indique si les corrélations doivent être étiquetées par probabilité ou valeur absolue.
weak_label	*chaîne*	
medium_label	*chaîne*	
strong_label	*chaîne*	
weak_below_probability	*nombre*	Lorsque correlation_mode est paramétré sur Probability, cette propriété indique la valeur de césure utilisée pour les corrélations faibles. Il doit s'agir d'une valeur comprise entre 0 et 1 (par exemple, 0,90).
strong_above_probability	*nombre*	Valeur de césure utilisée pour les corrélations fortes.
weak_below_absolute	*nombre*	Lorsque correlation_mode est paramétré sur Absolute, cette propriété indique la valeur de césure utilisée pour les corrélations faibles. Il doit s'agir d'une valeur comprise entre 0 et 1 (par exemple, 0,90).
strong_above_absolute	*nombre*	Valeur de césure utilisée pour les corrélations fortes.
unimportant_label	*chaîne*	
marginal_label	*chaîne*	
important_label	*chaîne*	
unimportant_below	*nombre*	Valeur de césure utilisée pour une importance de champ faible. Il doit s'agir d'une valeur comprise entre 0 et 1 (par exemple, 0,90).
important_above	*nombre*	
use_output_name	*indicateur*	Indique si un nom de sortie personnalisé est utilisé.
output_name	*chaîne*	Nom à utiliser.

Tableau 210. propriétés de meansnode (suite).

Propriétés de meansnode	Le type de données	Description de la propriété
output_mode	Screen File	Indique l'emplacement cible pour les sorties générées à partir du noeud de sortie.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Indique le type de sortie.
full_filename	*chaîne*	
output_view	Simple Advanced	Indique si la vue simple ou avancée est affichée dans la sortie.

Propriétés de reportnode

Ce noeud permet de créer des rapports formatés contenant du texte fixe et des données, ainsi que des expressions calculées à partir de ces dernières. Le format du rapport est déterminé par des modèles texte définissant la structure du texte fixe et de la sortie de données. Vous pouvez définir un formatage de texte personnalisé en utilisant des balises HTML dans le modèle et en définissant des options dans l'onglet Sortie. Vous pouvez inclure des valeurs de données et d'autres sorties conditionnelles à l'aide des expressions CLEM du modèle.

Exemple

```
node = stream.create("report", "My node")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/report_output.html")
node.setPropertyValue("lines_per_page", 50)
node.setPropertyValue("title", "Report node created by a script")
node.setPropertyValue("highlights", False)
```

Tableau 211. propriétés de reportnode.

Propriétés de reportnode	Le type de données	Description de la propriété
output_mode	Screen File	Permet d'indiquer l'emplacement cible pour les sorties générées à partir du noeud de sortie.
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	Permet d'indiquer le type de sortie.
use_output_name	*indicateur*	Indique si un nom de sortie personnalisé est utilisé.
output_name	*chaîne*	Si use_output_name a la valeur true (vrai), indique le nom à utiliser.
text	*chaîne*	
full_filename	*chaîne*	
highlights	*indicateur*	
title	*chaîne*	
lines_per_page	*nombre*	

Propriétés de routputnode

Le noeud de sortie R vous permet d'analyser des données et les résultats du scoring de modèle à l'aide de votre propre script R personnalisé. La sortie de l'analyse peut être textuelle ou graphique. La sortie est ajoutée à l'onglet **Sortie** du panneau du gestionnaire ; autrement, la sortie peut être redirigée vers un fichier.

Tableau 212. Propriétés de routputnode

Propriétés de routputnode	Type de données	Description de la propriété
syntax	*chaîne*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*flag*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*flag*	
output_name	Automatique Custom	
custom_name	*chaîne*	
output_to	Screen File	
output_type	Graph Text	
full_filename	*chaîne*	
graph_file_type	HTML COU	
text_file_type	HTML TEXT COU	

Propriétés de setglobalsnode

Le nœud V. globales (Valeurs globales) analyse les données et calcule des valeurs récapitulatives pouvant être utilisées dans des expressions CLEM. Par exemple, vous pouvez utiliser ce noeud pour calculer les statistiques d'un champ *âge*, puis utiliser la moyenne globale du champ *age* dans des expressions CLEM en insérant la fonction `@GLOBAL_MEAN(age)`.

Exemple

```
node = stream.create("setglobals", "My node")
node.setKeyedPropertyValue("globals", "Na", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "K", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
node.setPropertyValue("clear_first", False)
node.setPropertyValue("show_preview", True)
```

Tableau 213. propriétés de setglobalsnode.

Propriétés de setglobalsnode	Le type de données	Description de la propriété
globals	[Sum Mean Min Max SDev]	Propriété structurée où les champs à définir doivent être référencés avec la syntaxe suivante : node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
clear_first	*indicateur*	
show_preview	*indicateur*	

Propriétés de simevalnode

Le noeud Evaluation de simulation évalue un champ cible prévu et spécifié, et présente les informations de distribution et de corrélation concernant le champ cible.

Tableau 214. Propriétés de simevalnode.

Propriétés de simevalnode	Type de données	Description de la propriété
target	*zone*	
itération	*zone*	
presorted_by_iteration	*booléen*	
max_iterations	*nombre*	
tornado_fields	*[champ1...champN]*	
plot_pdf	*booléen*	
plot_cdf	*booléen*	
show_ref_mean	*booléen*	
show_ref_median	*booléen*	
show_ref_sigma	*booléen*	
num_ref_sigma	*nombre*	
show_ref_pct	*booléen*	
ref_pct_bottom	*nombre*	
ref_pct_top	*nombre*	
show_ref_custom	*booléen*	
ref_custom_values	*[numéro1...numéroN]*	
category_values	Catégorie Probabilities Both	
category_groups	Catégories Iterations	
create_pct_table	*booléen*	
pct_table	Quartiles Intervalles Custom	
pct_intervals_num	*nombre*	

Tableau 214. Propriétés de simevalnode (suite).

Propriétés de simevalnode	Type de données	Description de la propriété
pct_custom_values	*[numéro1...numéroN]*	

Propriétés de simfitnode

Le noeud Ajustement de simulation examine la distribution statistique des données dans chaque champ et génère (ou met à jour) un noeud Génération de simulation, avec la distribution la plus appropriée affectée à chaque champ. Le noeud Génération de simulation peut ensuite être utilisé pour générer les donnés simulées.

Tableau 215. Propriétés de simfitnode.

Propriétés de simfitnode	Type de données	Description de la propriété
build (génération)	Node XMLExport Both	
use_source_node_name	*booléen*	
source_node_name	*chaîne*	Nom personnalisé du noeud source qui est soit en cours de génération soit en cours de mise à jour.
use_cases	Tous LimitFirstN	
use_case_limit	*entier*	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	*entier*	
parameter_xml_filename	*chaîne*	
generate_parameter_import	*booléen*	

Propriétés de statisticsnode

Le noeud Statistiques fournit des informations récapitulatives de base sur les champs numériques. Ill calcule les statistiques récapitulatives des champs individuels et des corrélations entre les champs.

Exemple

```
node = stream.create("statistics", "My node")
#  Onglet Paramètres
node.setPropertyValue("examine", ["Age", "BP", "Drug"])
node.setPropertyValue("statistics", ["Mean", "Sum", "SDev"])
node.setPropertyValue("correlate", ["BP", "Drug"])
# "Correlation Labels..." section
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("weak_below_absolute", 0.25)
node.setPropertyValue("weak_label", "lower quartile")
node.setPropertyValue("strong_above_absolute", 0.75)
node.setPropertyValue("medium_label", "middle quartiles")
```

```
node.setPropertyValue("strong_label", "upper quartile")
# Onglet Sortie
node.setPropertyValue("full_filename", "c:/output/statistics_output.html")
node.setPropertyValue("output_format", "HTML")
```

Tableau 216. propriétés de statisticsnode.

Propriétés de **statisticsnode**	Le type de données	Description de la propriété
use_output_name	*indicateur*	Indique si un nom de sortie personnalisé est utilisé.
output_name	*chaîne*	Si use_output_name a la valeur true (vrai), indique le nom à utiliser.
output_mode	Screen File	Permet d'indiquer l'emplacement cible pour les sorties générées à partir du noeud de sortie.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Permet d'indiquer le type de sortie.
full_filename	*chaîne*	
examine	*liste (list)*	
correlate	*liste (list)*	
statistics	[Count Mean Sum Min Max Range Variance SDev SErr Median Mode]	
correlation_mode	Probabilité Absolute	Indique si les corrélations doivent être étiquetées par probabilité ou valeur absolue.
label_correlations	*indicateur*	
weak_label	*chaîne*	
medium_label	*chaîne*	
strong_label	*chaîne*	
weak_below_probability	*nombre*	Lorsque correlation_mode est paramétré sur Probability, cette propriété indique la valeur de césure utilisée pour les corrélations faibles. Il doit s'agir d'une valeur comprise entre 0 et 1 (par exemple, 0,90).
strong_above_probability	*nombre*	Valeur de césure utilisée pour les corrélations fortes.
weak_below_absolute	*nombre*	Lorsque correlation_mode est paramétré sur Absolute, cette propriété indique la valeur de césure utilisée pour les corrélations faibles. Il doit s'agir d'une valeur comprise entre 0 et 1 (par exemple, 0,90).
strong_above_absolute	*nombre*	Valeur de césure utilisée pour les corrélations fortes.

Propriétés de statisticsoutputnode

Le noeud Sortie Statistics vous permet d'appeler une procédure IBM SPSS Statistics pour analyser les données IBM SPSS Modeler. De nombreuses procédures d'analyses IBM SPSS Statistics sont disponibles. Ce noeud requiert une copie avec licence de IBM SPSS Statistics.

Les propriétés de ce nœud sont décrites dans «Propriétés de statisticsoutputnode», à la page 300.

Propriétés de tablenode

Le noeud Table affiche les données au format tabulaire (ces données peuvent également être écrites dans un fichier). Ainsi, vous pouvez passer en revue les valeurs de données ou les exporter dans un format facilement lisible.

Exemple

```
node = stream.create("table", "My node")
node.setPropertyValue("highlight_expr", "Age > 30")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("transpose_data", True)
node.setPropertyValue("full_filename", "C:/output/table_output.htm")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tableau 217. Propriétés de tablenode.

Propriétés de tablenode	Le type de données	Description de la propriété
full_filename	*chaîne*	Si vous indiquez Disque, Données ou HTML : nom du fichier de sortie utilisé.
use_output_name	*indicateur*	Indique si un nom de sortie personnalisé est utilisé.
output_name	*chaîne*	Si use_output_name a la valeur true (vrai), indique le nom à utiliser.
output_mode	Screen File	Permet d'indiquer l'emplacement cible pour les sorties générées à partir du noeud de sortie.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Permet d'indiquer le type de sortie.
transpose_data	*indicateur*	Transpose les données avant l'exportation, afin que les lignes et les colonnes représentent respectivement les champs et les enregistrements.
paginate_output	*indicateur*	Lorsque output_format est défini sur HTML, cette propriété divise la sortie en pages.
lines_per_page	*nombre*	Lorsque cette propriété est utilisée avec paginate_output, elle indique le nombre de lignes par page de sortie.
highlight_expr	*chaîne*	

Tableau 217. Propriétés de tablenode (suite).

Propriétés de tablenode	Le type de données	Description de la propriété
output	*chaîne*	Propriété en lecture seule qui comporte une référence à la dernière table créée par le nœud.
value_labels	*[{Valeur ChaîneLibellé} {Valeur ChaîneLibellé} ...]*	Permet d'attribuer des libellés aux paires de valeurs.
display_places	*entier*	Définit le nombre de décimales du champ lorsqu'il est affiché (s'applique uniquement aux champs dont le stockage est Réel). La valeur –1 utilise le flux par défaut.
export_places	*entier*	Définit le nombre de décimales du champ lorsqu'il est exporté (s'applique uniquement aux champs dont le stockage est Réel). La valeur –1 utilise le flux par défaut.
decimal_separator	DEFAULT PERIOD COMMA	Définit le séparateur décimal du champ (s'applique uniquement aux champs dont le stockage est Réel).
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	Définit le format de date du champ (s'applique uniquement aux champs dont le stockage est DATE ou TIMESTAMP).

Tableau 217. Propriétés de tablenode (suite).

Propriétés de tablenode	Le type de données	Description de la propriété
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Définit le format d'heure du champ (s'applique uniquement aux champs dont le stockage est TIME ou TIMESTAMP).
column_width	*entier*	Définit la largeur des colonnes du champ. La valeur –1 définit la largeur des colonnes sur Auto.
justify	AUTO CENTER LEFT RIGHT	Définit la justification des colonnes du champ.

Propriétés de transformnode

Le noeud Transformation vous permet de sélectionner et de prévisualiser les résultats des transformations avant de les appliquer aux champs sélectionnés.

Exemple

```
node = stream.create("transform", "My node")
node.setPropertyValue("fields", ["AGE", "INCOME"])
node.setPropertyValue("formula", "Select")
node.setPropertyValue("formula_log_n", True)
node.setPropertyValue("formula_log_n_offset", 1)
```

Tableau 218. propriétés de transformnode.

Propriétés de transformnode	Le type de données	Description de la propriété
fields	[*champ1... champn*]	Champs à utiliser dans la transformation.
formula	All sélectionner	Indique si toutes les transformations ou seules les transformations sélectionnées doivent être calculées.
formula_inverse	*indicateur*	Indique si la transformation inverse doit être utilisée.
formula_inverse_offset	*nombre*	Indique un décalage de données à utiliser pour la formule. Elément paramétré sur 0 par défaut, sauf s'il est défini par l'utilisateur.

Tableau 218. propriétés de transformnode (suite).

Propriétés de transformnode	Le type de données	Description de la propriété
formula_log_n	*indicateur*	Indique si la transformation par le logarithme$_n$ doit être utilisée.
formula_log_n_offset	*nombre*	
formula_log_10	*indicateur*	Indique si la transformation par le logarithme$_{10}$ doit être utilisée.
formula_log_10_offset	*nombre*	
formula_exponential	*indicateur*	Indique si la transformation exponentielle (e^x) doit être utilisée.
formula_square_root	*indicateur*	Indique si la transformation par la racine carrée doit être utilisée.
use_output_name	*indicateur*	Indique si un nom de sortie personnalisé est utilisé.
output_name	*chaîne*	Si use_output_name a la valeur true (vrai), indique le nom à utiliser.
output_mode	Screen File	Permet d'indiquer l'emplacement cible pour les sorties générées à partir du noeud de sortie.
output_format	HTML (*.html*) Output (*.cou*)	Permet d'indiquer le type de sortie.
paginate_output	*indicateur*	Lorsque output_format est défini sur HTML, cette propriété divise la sortie en pages.
lines_per_page	*nombre*	Lorsque cette propriété est utilisée avec paginate_output, elle indique le nombre de lignes par page de sortie.
full_filename	*chaîne*	Indique le nom du fichier à utiliser pour la sortie de fichier.

Chapitre 17. Propriétés du nœud d'exportation

Propriétés communes des nœuds Exportation

Les propriétés suivantes sont communes à tous les nœuds d'exportation :

Tableau 219. Propriétés communes des noeuds d'exportation

Propriété	Valeurs	Description de la propriété
`publish_path`	*chaîne*	Saisissez le nom racine à utiliser pour les fichiers d'images et de paramètres publiés.
`publish_metadata`	*flag*	Spécifie si un fichier de métadonnées décrit les entrées et les sorties de l'image et de leurs modèles de données.
`publish_use_parameters`	*flag*	Spécifie si des paramètres de flux sont inclus dans le fichier *.par.
`publish_parameters`	*liste de chaîne*	Spécifie les paramètres à inclure.
`execute_mode`	`export_data` `publier`	Spécifie si le nœud s'exécute sans publier le flux, aussi le flux est automatiquement publié lorsque le nœud est exécuté.

Propriétés de asexport

L'exportation Analytic Server vous permet d'exécuter un flux sur le système de fichiers HDFS (Hadoop Distributed File System).

Exemple

```
node = stream.create("asexport", "My node")
node.setPropertyValue("data_source", "Drug1n")
node.setPropertyValue("export_mode", "overwrite")
```

Tableau 220. propriétés de asexport.

Propriétés de asexport	Type de données	Description de la propriété
`data_source`	*chaîne*	Nom de la source de données.
`export_mode`	*chaîne*	Indique s'il faut ajouter (**`append`**) les données exportées à la source de données existante ou s'il faut remplacer (**`overwrite`**) la source de données existante.

Propriétés cognosexportnode

Le noeud IBM Cognos BI Export exporte des données dans un format qui peut être lu par les bases de données Cognos BI.

Pour ce nœud, vous devez définir une connexion Cognos et une connexion ODBC.

Connexion Cognos

Les propriétés de la connexion Cognos sont les suivantes.

Tableau 221. Propriétés cognosexportnode

Propriétés de cognosexportnode	Le type de données	Description de la propriété
cognos_connection	*{"field","field", ... ,"field"}*	Une propriété de liste contenant les détails de la connexion du serveur Cognos. Le format est : {URL_serveur_Cognos", mode_connexion, "espace de nom", "nom utilisateur", "mot de passe"} où : URL_serveur_Cognos est l'URL du serveur Cognos contenant la source. mode_connexion indique si la connexion anonyme est utilisée et est soit true soit false ; si la valeur est true, les champs suivants doivent être définis sur "". espace de nom spécifie le fournisseur de sécurité pour l'authentification utilisé pour se connecter au serveur. nom utilisateur et mot de passe sont ceux utilisés pour la connexion au serveur Cognos. Au lieu de mode_connexion, les modes suivants sont également disponibles : • anonymousMode. Par exemple : {'URL_serveur_Cognos', 'anonymousMode', "espace de nom", "nom utilisateur", "mot de passe"} • credentialMode. Par exemple : {'URL_serveur_Cognos', 'credentialMode', "espace de nom", "nom utilisateur", "mot de passe"} • storedCredentialMode. Par exemple : {'URL_serveur_Cognos', 'storedCredentialMode', "nom_données_identification_stockées"} Où nom_données_identification_stockées est le nom de données d'identification stockées Cognos dans le référentiel.
cognos_package_name	*chaîne*	Le chemin d'accès et le nom du package Cognos vers lequel vous exportez les donnés, par exemple : /Public Folders/MyPackage
cognos_datasource	*chaîne*	
cognos_export_mode	Publish ExporterFichier	
cognos_filename	*chaîne*	

connexion ODBC

Les propriétés de la connexion ODBC sont identiques à celles répertoriées pour databaseexportnode dans la prochaine section, à l'exception de la propriété datasource qui n'est pas valide.

Propriétés de databaseexportnode

Le noeud exportation de base de données écrit des données dans une source de données relationnelles compatible ODBC. Pour que cette opération puisse être effectuée, la source de données ODBC doit exister et vous devez y avoir accès en écriture.

Exemple

```
'''
Use this sample with fraud.str from demo folder
Assumes a datasource named "MyDatasource" has been configured
'''
stream = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByType("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)

# Export tab
db_exportnode.setPropertyValue("username", "user")
db_exportnode.setPropertyValue("datasource", "MyDatasource")
db_exportnode.setPropertyValue("password", "password")
db_exportnode.setPropertyValue("table_name", "predictions")
db_exportnode.setPropertyValue("write_mode", "Create")
db_exportnode.setPropertyValue("generate_import", True)
db_exportnode.setPropertyValue("drop_existing_table", True)
db_exportnode.setPropertyValue("delete_existing_rows", True)
db_exportnode.setPropertyValue("default_string_size", 32)

# Schema dialog
db_exportnode.setKeyedPropertyValue("type", "region", "VARCHAR(10)")
db_exportnode.setKeyedPropertyValue("export_db_primarykey", "id", True)
db_exportnode.setPropertyValue("use_custom_create_table_command", True)
db_exportnode.setPropertyValue("custom_create_table_command", "My SQL Code")

# Indexes dialog
db_exportnode.setPropertyValue("use_custom_create_index_command", True)
db_exportnode.setPropertyValue("custom_create_index_command", "CREATE BITMAP INDEX <index-name>
 ON <table-name> <(index-columns)>")
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields", ["id", "region"]])
```

Tableau 222. propriétés de databaseexportnode.

Propriétés de databaseexportnode	Le type de données	Description de la propriété
datasource	*chaîne*	
username	*chaîne*	
password	*chaîne*	
epassword	*chaîne*	Cette propriété est en lecture seule au cours de l'exécution. Pour générer un mot de passe codé, sélectionnez Outil pour mot de passe dans le menu Outils. Pour plus d'informations, voir la rubrique «Génération d'un mot de passe codé», à la page 51.
table_name	*chaîne*	

Tableau 222. propriétés de databaseexportnode (suite).

Propriétés de databaseexportnode	Le type de données	Description de la propriété
write_mode	Create Append Merge	
map	*chaîne*	Mappe un nom de champ de flux sur un nom de colonne de base de données (valide uniquement si write_mode est sur Merge). Pour une fusion, tous les champs doivent être mappés afin d'être exportés. Les noms de champ qui n'existent pas dans la base de données sont ajoutés en tant que nouvelles colonnes.
key_fields	*liste (list)*	Spécifie le champ de flux utilisé pour la clé ; la propriété map indique à quoi cela correspond dans la base de données.
join	Database Add	
drop_existing_table	*indicateur*	
delete_existing_rows	*indicateur*	
default_string_size	*entier*	
type		Propriété structurée utilisée pour définir le type de schéma.
generate_import	*indicateur*	
use_custom_create_table_command	*indicateur*	Utilisez la propriété *custom_create_table* pour modifier la commande SQL standard CREATE TABLE.
custom_create_table_command	*chaîne*	Indique la commande de chaîne à utiliser à la place de la commande SQL standard CREATE TABLE.
use_batch	*indicateur*	Les propriétés suivantes sont les options avancées du chargement en masse dans la base de données. La valeur True (vrai) pour Use_batch désactive les validations ligne par ligne soumises à la base de données.
batch_size	*nombre*	Indique le nombre d'enregistrements à envoyer à la base de données avant validation dans la mémoire.
bulk_loading	Off ODBC External	Indique le type de chargement en masse. Les options supplémentaires relatives à ODBC et à External sont répertoriées ci-dessous.
not_logged	*flag*	
odbc_binding	Ligne Column	Indiquez le lien par ligne ou par colonne pour le chargement en masse via ODBC.

Tableau 222. propriétés de databaseexportnode (suite).

Propriétés de databaseexportnode	Le type de données	Description de la propriété
loader_delimit_mode	Tabulation Space Other	Indiquez le type de délimiteur pour le chargement en masse via un programme externe. Sélectionnez Other avec la propriété loader_other_delimiter pour indiquer les délimiteurs, tels que la virgule (,).
loader_other_delimiter	*chaîne*	
specify_data_file	*indicateur*	Un indicateur ayant la valeur True (vrai) active la propriété data_file ci-dessous, vous permettant d'indiquer le nom et le chemin du fichier où écrire lors du chargement en masse dans la base de données.
data_file	*chaîne*	
specify_loader_program	*indicateur*	Un indicateur ayant la valeur True (vrai) active la propriété loader_program ci-dessous, vous permettant d'indiquer le nom et l'emplacement d'un programme ou d'un script de module de chargement externe.
loader_program	*chaîne*	
gen_logfile	*indicateur*	Un indicateur ayant la valeur True (vrai) active la propriété logfile_name ci-dessous, vous permettant d'indiquer le nom d'un fichier se trouvant sur le serveur pour générer un journal des erreurs.
logfile_name	*chaîne*	
check_table_size	*indicateur*	Un indicateur ayant la valeur True (vrai) permet de vérifier les tables pour garantir que l'augmentation de la taille des tables de la base de données correspond au nombre de lignes exportées à partir de IBM SPSS Modeler.
loader_options	*chaîne*	Indiquez des arguments supplémentaires (comme -comment et -specialdir) pour le programme du module de chargement.
export_db_primarykey	*indicateur*	Indique si un champ donné est une clé primaire.
use_custom_create_index_command	*indicateur*	Si la valeur true (vrai) est utilisée, active le code SQL personnalisé pour tous les index.
custom_create_index_command	*chaîne*	Indique la commande SQL utilisée pour créer des index lorsque le code SQL personnalisé est activé. (Cette valeur peut être ignorée pour certains index comme indiqué ci-dessous.)

Tableau 222. propriétés de databaseexportnode (suite).

Propriétés de databaseexportnode	Le type de données	Description de la propriété
indexes.INDEXNAME.fields		Crée l'index indiqué si nécessaire et répertorie les noms de champ à inclure dans cet index.
INDEXNAME "use_custom_create_⌂index_command"	*indicateur*	Permet d'activer/de désactiver le code SQL personnalisé pour un index spécifique. Reportez-vous aux exemples ci-dessous.
INDEXNAME "custom_create_index_command"	*chaîne*	Indique le code SQL personnalisé utilisé pour l'index spécifié. Reportez-vous aux exemples ci-dessous.
indexes.INDEXNAME.remove	*indicateur*	Si la valeur True (vrai) est utilisée, supprime l'index spécifié de l'ensemble d'index.
table_space	*chaîne*	Spécifie l'espace Table qui sera créé.
use_partition	*indicateur*	Spécifie que le champ de distribution par hachage sera utilisé.
partition_field	*chaîne*	Spécifie le contenu du champ de distibution par hachage.

Remarque : Pour certaines bases de données, vous pouvez spécifier que les tables de base de données doivent être créées pour l'exportation avec compression (par exemple, l'équivalent de `CREATE TABLE MYTABLE (...) COMPRESS YES;` dans SQL). Les propriétés `use_compression` et `compression_mode` sont fournies pour prendre en charge cette fonctionnalité, comme suit.

Tableau 223. databaseexportnode properties using compression features.

Propriétés de databaseexportnode	Le type de données	Description de la propriété
use_compression	*Booléen*	Si défini sur True, crée des tables à exporter avec la compression.
compression_mode	Ligne Page	Définit le niveau de compression des bases de données SQL Server.
	Default Direct_Load_Operations All_Operations Basique OLTP Query_High Query_Low Archive_High Archive_Low	Définit le niveau de compression des bases de données Oracle. Veuillez noter que les valeurs OLTP, Query_High, Query_Low, Archive_High et Archive_Low requièrent Oracle 11gR2 au minimum.

Exemple montrant comment modifier la commande CREATE INDEX pour un index spécifique

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["use_custom_create_index_command",
 True])db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["custom_create_index_command",
 "CREATE BITMAP INDEX <index-name> ON <table-name> <(index-columns)>"])
```

Vous pouvez également exécuter cette opération via une table de hachage

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", {"fields":["id", "region"],
 "use_custom_create_index_command":True, "custom_create_index_command":"CREATE INDEX
<index-name> ON
 <table-name> <(index-columns)>"})
```

Propriétés de datacollectionexportnode

Le noeud Export IBM SPSS Data Collection génère des données au format utilisé par les logiciels d'étude de marché IBM SPSS Data Collection. Pour pouvoir utiliser ce nœud, vous devez avoir installé avant la bibliothèque de données IBM SPSS Data Collection.

Exemple

```
stream = modeler.script.stream()
datacollectionexportnode = stream.createAt("datacollectionexport", "Data Collection", 200, 200)
datacollectionexportnode.setPropertyValue("metadata_file", "c:\\museums.mdd")
datacollectionexportnode.setPropertyValue("merge_metadata", "Overwrite")
datacollectionexportnode.setPropertyValue("casedata_file", "c:\\museumdata.sav")
datacollectionexportnode.setPropertyValue("generate_import", True)
datacollectionexportnode.setPropertyValue("enable_system_variables", True)
```

Tableau 224. propriétés de datacollectionexportnode

Propriétés de datacollectionexportnode	Le type de données	Description de la propriété
metadata_file	*chaîne*	Nom du fichier de métadonnées à exporter.
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	*flag*	Spécifie si le fichier *.mdd* exporté devrait inclure les variables système IBM SPSS Data Collection.
casedata_file	*chaîne*	Le nom du fichier *.sav* vers lequel les données d'observation sont exportées.
generate_import	*flag*	

Propriétés de excelexportnode

Le noeud Export Excel génère une sortie de données au format Microsoft Excel (*.xls*). Si vous le souhaitez, vous pouvez choisir de lancer Excel automatiquement et d'ouvrir le fichier exporté lors de l'exécution du noeud.

Exemple

```
stream = modeler.script.stream()
excelexportnode = stream.createAt("excelexport", "Excel", 200, 200)
excelexportnode.setPropertyValue("full_filename", "C:/output/myexport.xls")
excelexportnode.setPropertyValue("excel_file_type", "Excel2007")
excelexportnode.setPropertyValue("inc_field_names", True)
excelexportnode.setPropertyValue("inc_labels_as_cell_notes", False)
excelexportnode.setPropertyValue("launch_application", True)
excelexportnode.setPropertyValue("generate_import", True)
```

Tableau 225. Propriétés de excelexportnode

Propriétés de **excelexportnode**	Le type de données	Description de la propriété
full_filename	*chaîne*	
excel_file_type	Excel2003 Excel2007	
export_mode	Create Ajout	
inc_field_names	*flag*	Indique si les noms de champ doivent être inclus dans la première ligne de la feuille de calcul.
start_cell	*chaîne*	Spécifie la cellule de démarrage de l'export.
worksheet_name	*chaîne*	Nom de la feuille de calcul à écrire.
launch_application	*flag*	Indique si Excel doit être appelé pour le fichier obtenu. Notez que le chemin de lancement d'Excel doit être spécifié dans la boîte de dialogue Programmes externes (menu Outils, Programmes externes).
generate_import	*flag*	Indique si un nœud Import Excel doit être généré pour lire le fichier de données exporté.

Propriétés de outputfilenode

Le noeud Export Fichier plat génère des données dans un fichier texte délimité. Elles peuvent ainsi être lues par d'autres logiciels d'analyse ou par des tableurs.

Exemple

```
stream = modeler.script.stream()
outputfile = stream.createAt("outputfile", "File Output", 200, 200)
outputfile.setPropertyValue("full_filename", "c:/output/flatfile_output.txt")
outputfile.setPropertyValue("write_mode", "Append")
outputfile.setPropertyValue("inc_field_names", False)
outputfile.setPropertyValue("use_newline_after_records", False)
outputfile.setPropertyValue("delimit_mode", "Tab")
outputfile.setPropertyValue("other_delimiter", ",")
outputfile.setPropertyValue("quote_mode", "Double")
outputfile.setPropertyValue("other_quote", "*")
outputfile.setPropertyValue("decimal_symbol", "Period")
outputfile.setPropertyValue("generate_import", True)
```

Tableau 226. Propriétés de outputfilenode

Propriétés de **outputfilenode**	Le type de données	Description de la propriété
full_filename	*chaîne*	Nom du fichier de sortie.
write_mode	Overwrite Ajout	

Tableau 226. Propriétés de outputfilenode (suite)

Propriétés de outputfilenode	Le type de données	Description de la propriété
inc_field_names	*flag*	
use_newline_after_records	*flag*	
delimit_mode	Comma Tabulation Space Autre	
other_delimiter	*char*	
quote_mode	None Single Double Autre	
other_quote	*flag*	
generate_import	*flag*	
encodage	StreamDefault SystemDefault "UTF-8"	

Propriétés de sasexportnode

Le noeud Export SAS permet d'obtenir des données de sortie au format SAS afin qu'elles puissent être lues par SAS ou par un logiciel compatible. Trois formats de fichier SAS sont disponibles : SAS pour Windows/OS2, SAS pour UNIX ou SAS Version 7/8.

Exemple

```
stream = modeler.script.stream()
sasexportnode = stream.createAt("sasexport", "SAS Export", 200, 200)
sasexportnode.setPropertyValue("full_filename", "c:/output/SAS_output.sas7bdat")
sasexportnode.setPropertyValue("format", "SAS8")
sasexportnode.setPropertyValue("export_names", "NamesAndLabels")
sasexportnode.setPropertyValue("generate_import", True)
```

Tableau 227. propriétés de sasexportnode

Propriétés de sasexportnode	Le type de données	Description de la propriété
format	Windows UNIX SAS7 SAS8	Champs étiquette de propriétés de variante.
full_filename	*chaîne*	
export_names	NamesAndLabels NamesAsLabels	Permet de mapper les noms de champ de IBM SPSS Modeler lors de l'exportation vers des noms de variable IBM SPSS Statistics ou SAS.
generate_import	*flag*	

Propriétés de statisticsexportnode

Le noeud Export Statistics génère des données au format IBM SPSS Statistics *.sav* ou *.zsav*. Les fichiers *.sav* ou *.zsav* peuvent être lus par IBM SPSS Statistics Base et d'autres produits. Ce format est également utilisé pour les fichiers cache IBM SPSS Modeler.

Les propriétés de ce nœud sont décrites dans «Propriétés de statisticsexportnode», à la page 301.

Propriétés du noeud tm1export

Le noeud IBM Cognos TM1 Export exporte des données dans un format qui peut être lu par les bases de données Cognos TM1.

Tableau 228. Propriétés du noeud tm1export.

Propriétés du noeud tm1export	Type de données	Description de la propriété
pm_host	*chaîne*	Nom d'hôte. Par exemple : `TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
tm1_connection	*{""champ"",""champ"", ... ,""champ""}*	Une propriété de liste contenant les détails de la connexion du serveur TM1. Le format est le suivant : `[ "nom_serveur_TM1", "nom_utilisateur_tm1", "mot_de_passe_tm1"]` Par exemple : `TM1_export.setPropertyValue("tm1_connection", ['Planning Sample', "admin" "apple"])`
selected_cube	*champ*	Nom du cube vers lequel vous exportez des données. Par exemple : `TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")`
spssfield_tm1element_mapping	*liste*	L'élément tm1 vers lequel le mappage a lieu doit faire partie de la dimension de colonne pour la vue de cube sélectionnée. Le format est le suivant : `[{"param1", "valeur"},...,{"paramN","valeur"}]` Par exemple : `TM1_export.setPropertyValue("spssfield_ tm1element_mapping", [["plan_version","plan_version"], ["plan_department","plan_department"]])`

Propriétés de xmlexportnode

Le noeud Export XML génère une sortie de données dans un fichier au format XML. Vous pouvez également créer un noeud source XML pour lire de nouveau les données exportées dans le flux.

Exemple

```
stream = modeler.script.stream()
xmlexportnode = stream.createAt("xmlexport", "XML Export", 200, 200)
xmlexportnode.setPropertyValue("full_filename", "c:/export/data.xml")
xmlexportnode.setPropertyValue("map", [["/catalog/book/genre", "genre"],
["/catalog/book/title", "title"]])
```

Tableau 229. Propriétés de xmlexportnode

Propriétés de xmlexportnode	**Le type de données**	**Description de la propriété**
full_filename	*chaîne*	(requis) Chemin complet et nom de fichier du fichier d'exportation XML.
use_xml_schema	*flag*	Spécifiez s'il faut utiliser un schéma XML (fichier XSD ou DTD) pour contrôler la structure des données exportées.
full_schema_filename	*chaîne*	Chemin complet et nom de fichier et du fichier XSD ou DTD à utiliser. Requis si use_xml_schema est défini sur true (vrai).
generate_import	*flag*	Génère un nœud source XML pour lire le fichier de données exporté dans le flux.
enregistrements	*chaîne*	Expression XPath qui indique la limite de l'enregistrement.
map	*chaîne*	Mappe le nom de fichier dans une structure XML.

Chapitre 18. Propriétés de noeuds IBM SPSS Statistics

Propriétés de statisticsimportnode

Le noeud Fichier Statistics lit les données du format de fichier *.sav* ou *.zsav* utilisé par IBM SPSS Statistics, ainsi que des fichiers cache enregistrés dans IBM SPSS Modeler, qui utilisent le même format.

Exemple

```
stream = modeler.script.stream()
statisticsimportnode = stream.createAt("statisticsimport", "SAV Import", 200, 200)
statisticsimportnode.setPropertyValue("full_filename", "C:/data/drug1n.sav")
statisticsimportnode.setPropertyValue("import_names", True)
statisticsimportnode.setPropertyValue("import_data", True)
```

Tableau 230. propriétés de statisticsimportnode.

Propriétés de statisticsimportnode	Le type de données	Description de la propriété
full_filename	*chaîne*	Nom du fichier complet, y compris son chemin d'accès.
password	*chaîne*	Mot de passe. Le paramètre password doit être défini avant le paramètre file_encrypted.
file_encrypted	*indicateur*	Indique si le fichier est protégé par mot de passe ou non.
import_names	NamesAndLabels LabelsAsNames	Méthode de gestion des noms de variable et des libellés.
import_data	DataAndLabels LabelsAsData	Méthode de gestion des valeurs et des libellés.
use_field_format_for_storage	*Booléen*	Spécifie s'il convient d'utiliser les informations de format de champ IBM SPSS Statistics lors de l'importation.

Propriétés statisticstransformnode

Le noeud Transformation exécute une sélection de commandes de syntaxe IBM SPSS Statistics en fonction des sources de données dans IBM SPSS Modeler. Ce noeud requiert une copie avec licence de IBM SPSS Statistics.

Exemple

```
stream = modeler.script.stream()
statisticstransformnode = stream.createAt("statisticstransform", "Transform", 200, 200)
statisticstransformnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticstransformnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed Drugs")
statisticstransformnode.setPropertyValue("check_before_saving", True)
```

Tableau 231. propriétés statisticstransformnode

Propriétés de statisticstransformnode	Le type de données	Description de la propriété
syntax	*chaîne*	
check_before_saving	*flag*	Valide la syntaxe saisie avant d'enregistrer les entrées. Affiche un message d'erreur si la syntaxe n'est pas valide.
default_include	*flag*	Pour plus d'informations, voir la rubrique «Propriétés de filternode», à la page 128.
include	*flag*	Pour plus d'informations, voir la rubrique «Propriétés de filternode», à la page 128.
new_name	*chaîne*	Pour plus d'informations, voir la rubrique «Propriétés de filternode», à la page 128.

Propriétés statisticsmodelnode

Le noeud Modèle Statistics vous permet d'analyser et de travailler avec vos données en exécutant des procédures IBM SPSS Statistics qui produisent un PMML. Ce noeud requiert une copie avec licence de IBM SPSS Statistics.

Exemple

```
stream = modeler.script.stream()
statisticsmodelnode = stream.createAt("statisticsmodel", "Model", 200, 200)
statisticsmodelnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticsmodelnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed Drugs")
```

Propriétés de statisticsmodelnode	Le type de données	Description de la propriété
syntax	*chaîne*	
default_include	*flag*	Pour plus d'informations, voir la rubrique «Propriétés de filternode», à la page 128.
include	*flag*	Pour plus d'informations, voir la rubrique «Propriétés de filternode», à la page 128.
new_name	*chaîne*	Pour plus d'informations, voir la rubrique «Propriétés de filternode», à la page 128.

Propriétés de statisticsoutputnode

Le noeud Sortie Statistics vous permet d'appeler une procédure IBM SPSS Statistics pour analyser les données IBM SPSS Modeler. De nombreuses procédures d'analyses IBM SPSS Statistics sont disponibles. Ce noeud requiert une copie avec licence de IBM SPSS Statistics.

Exemple

```
stream = modeler.script.stream()
statisticsoutputnode = stream.createAt("statisticsoutput", "Output", 200, 200)
statisticsoutputnode.setPropertyValue("syntax", "SORT CASES BY Age(A) Sex(A)
BP(A) Cholesterol(A)")
statisticsoutputnode.setPropertyValue("use_output_name", False)
statisticsoutputnode.setPropertyValue("output_mode", "File")
statisticsoutputnode.setPropertyValue("full_filename", "Cases by Age, Sex and Medical History")
statisticsoutputnode.setPropertyValue("file_type", "HTML")
```

Tableau 232. propriétés de statisticsoutputnode

Propriétés de **statisticsoutputnode**	Le type de données	Description de la propriété
mode	Dialog Syntax	Sélectionne l'option « Boîte de dialogue IBM SPSS Statistics » ou l'Éditeur de syntaxe.
syntax	*chaîne*	
use_output_name	*flag*	
output_name	*chaîne*	
output_mode	Screen Fichier	
full_filename	*chaîne*	
type_fichier	HTML SPV SPW	

Propriétés de statisticsexportnode

Le noeud Export Statistics génère des données au format IBM SPSS Statistics *.sav* ou *.zsav*. Les fichiers *.sav* ou *.zsav* peuvent être lus par IBM SPSS Statistics Base et d'autres produits. Ce format est également utilisé pour les fichiers cache IBM SPSS Modeler.

Exemple

```
stream = modeler.script.stream()
statisticsexportnode = stream.createAt("statisticsexport", "Export", 200, 200)
statisticsexportnode.setPropertyValue("full_filename", "c:/output/SPSS_Statistics_out.sav")
statisticsexportnode.setPropertyValue("field_names", "Names")
statisticsexportnode.setPropertyValue("launch_application", True)
statisticsexportnode.setPropertyValue("generate_import", True)
```

Tableau 233. propriétés de statisticsexportnode.

Propriétés de **statisticsexportnode**	Le type de données	Description de la propriété
full_filename	*chaîne*	
type_fichier	Standard compressé	Enregistrez le fichier au format *sav* ou *zsav*.
encrypt_file	*indicateur*	Indique si le fichier est protégé par mot de passe ou non.
password	*chaîne*	Mot de passe.
launch_application	*indicateur*	

Tableau 233. propriétés de statisticsexportnode (suite).

Propriétés de **statisticsexportnode**	Le type de données	Description de la propriété
`export_names`	`NamesAndLabels` `NamesAsLabels`	Permet de mapper les noms de champ de IBM SPSS Modeler lors de l'exportation vers des noms de variable IBM SPSS Statistics ou SAS.
`generate_import`	*indicateur*	

Chapitre 19. Propriétés du super noeud

Les propriétés propres aux super noeuds sont décrites dans les tableaux suivants : Remarque : les propriétés de noeud communes s'appliquent également aux super noeuds.

Tableau 234. Propriétés de super noeud terminal

Nom de la propriété	Type de propriété/Liste de valeurs	Description de la propriété
execute_method	Script Normal	
script	*chaîne*	

Paramètres du super noeud

Vous pouvez utiliser des scripts pour créer ou définir les paramètres du super noeud en utilisant le format général :

```
mySuperNode.setParameterValue("minvalue", 30)
```

Vous pouvez extraire la valeur de paramètre avec :

```
value mySuperNode.getParameterValue("minvalue")
```

Recherche des super noeuds existants

Vous pouvez rechercher des super noeuds dans des flux à l'aide de la fonction findByType() :

```
source_supernode = modeler.script.stream().findByType("source_super", None)
process_supernode = modeler.script.stream().findByType("process_super", None)
terminal_supernode = modeler.script.stream().findByType("terminal_super", None)
```

Définition des propriétés des noeuds encapsulés

Vous pouvez paramétrer les propriétés de noeuds encapsulés spécifiques dans un super noeud en accédant au diagramme enfant au sein du super noeud. Par exemple, supposons que vous ayez un super noeud source avec un noeud Délimité encapsulé pour lire les données. Vous pouvez transmettre le nom du fichier à lire (indiqué par la propriété full_filename) en accédant au diagramme enfant et en recherchant le noeud pertinent comme suit :

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

Création de super noeuds

Pour créer un super noeud et son contenu à partir de zéro, vous pouvez opérer de manière similaire en créant le super noeud, en accédant au diagramme enfant et en créant les noeuds souhaités. Vous devez également vérifier que les noeuds au sein du diagramme de super noeud sont aussi liés aux noeuds de connecteurs d'entrée et/ou de sortie. Par exemple, si vous souhaitez créer un super noeud de processus :

```
process_supernode = modeler.script.stream().createAt("process_super", "My SuperNode", 200, 200)
childDiagram = process_supernode.getChildDiagram()
filternode = childDiagram.createAt("filter", "My Filter", 100, 100)
childDiagram.linkFromInputConnector(filternode)
childDiagram.linkToOutputConnector(filternode)
```

Annexe A. Référence des noms de noeuds

Cette section fournit une référence pour les noms de scriptage des noeuds dans IBM SPSS Modeler.

Noms des nuggets de modèle

Les nuggets de modèle (également appelés modèles générés) peuvent être référencés par type, tout comme les objets de noeud et de sortie. Le tableau ci-dessous reprend les noms de référence d'objet de modèle.

Ces noms sont utilisés spécialement pour le référencement des nuggets de modèle figurant dans la palette Modèles (dans l'angle supérieur droit de la fenêtre IBM SPSS Modeler). Pour référencer les noeuds de modèle qui ont été ajoutés à un flux à des fins de scoring, le système utilise un ensemble différent de noms commençant par `apply...`. Pour plus d'informations, voir la rubrique Propriétés du noeud de nugget de modèle.

Remarque : en règle générale, le référencement des modèles à la fois par nom *et* par type est recommandé car il permet d'éviter toute confusion.

Tableau 235. Noms des nuggets de modèle (Palette Modélisation).

Nom du modèle	**Modèle**
anomalydetection	Anomalie
apriori	Apriori
autoclassifier	Discriminant automatique
autocluster	Classification non supervisée automatique
autonumeric	Numérisation automatique
bayesnet	Réseau Bayésien
c50	C5.0
carma	Carma
cart	Arbre C&RT
chaid	CHAID
coxreg	Régression de Cox
decisionlist	Liste de décision
discriminant	Analyse discriminante
facteur	ACP/Facteur
featureselection	Sélection de fonction
genlin	Régression linéaire généralisée
glmm	GLMM
kmeans	k moyenne
knn	*k*-Voisin le plus proche
kohonen	Kohonen
linear	Linéaire
logreg	Régression logistique
neuralnetwork	Réseau de neurones

Tableau 235. Noms des nuggets de modèle (Palette Modélisation) (suite).

Nom du modèle	Modèle
quest	QUEST
régression	Régression linéaire
séquence	Séquence
slrm	Modèle de réponse en auto-apprentissage
statisticsmodel	Modèle IBM SPSS Statistics
svm	Support vector machine
timeseries	Séries temporelles
twostep	TwoStep

Tableau 236. Noms des nuggets de modèle (Palette Modélisation de base de données).

Nom du modèle	Modèle
db2imcluster	Classification ISW IBM
db2imlog	Régression logistique ISW IBM
db2imnb	Naive Bayes ISW IBM
db2imreg	Régression ISW IBM
db2imtree	Arbre de décision ISW IBM
msassoc	Règles d'association MS
msbayes	MS Naive Bayes
mscluster	Classification non supervisée MS
mslogistic	Régression logistique MS
msneuralnetwork	Réseau neuronal MS
msregression	Régression linéaire MS
mssequencecluster	Mise en cluster de séquences MS
mstimeseries	Séries temporelles MS
mstree	Arbre de décision MS
netezzabayes	Bayes Net Netezza
netezzadectree	Arbre décision Netezza
netezzadivcluster	Classification par division Netezza
netezzaglm	Linéaire généralisé Netezza
netezzakmeans	k moyenne Netezza
netezzaknn	KNN Netezza
netezzalineregression	Régression linéaire Netezza
netezzanaivebayes	Naive Bayes Netezza
netezzapca	ACP Netezza
netezzaregtree	Arbre de régression Netezza
netezzatimeseries	Séries temporelles Netezza
oraabn	Oracle Adaptive Bayes
oraai	Oracle AI
oradecisiontree	Arbre décision Oracle
oraglm	Oracle GLM

Tableau 236. Noms des nuggets de modèle (Palette Modélisation de base de données) (suite).

Nom du modèle	Modèle
orakmeans	*k* moyenne Oracle
oranb	Oracle Naive Bayes
oranmf	NMF Oracle
oraocluster	O-Cluster Oracle
orasvm	Oracle SVM

Pour éviter les noms de modèle en double

Lorsque vous utilisez des scripts pour manipuler les modèles générés, gardez à l'esprit que les noms de modèle en double peuvent donner lieu à des références ambiguës. Pour éviter ce problème, il s'avère judicieux, lors de la génération de scripts, d'utiliser des noms uniques pour les modèles générés.

Pour définir les options des noms de modèle en double, procédez comme suit :

1. A partir des menus, sélectionnez :
   Outils > **Options d'utilisateur**
2. Cliquez sur l'onglet **Notifications**.
3. Sélectionnez **Remplacer le modèle précédent** afin de limiter les noms en double pour les modèles générés.

Le comportement de l'exécution du script peut varier entre SPSS Modeler et IBM SPSS Collaboration and Deployment Services lorsqu'il existe des références de modèle ambigües. Le client SPSS Modeler inclut l'option « Remplacer le modèle précédent » qui remplace automatiquement les modèles qui ont le même nom (par exemple, lorsqu'un script parcourt une boucle pour produire un modèle différent à chaque fois). Cependant, cette option n'est pas disponible lorsque le même script est exécuté dans IBM SPSS Collaboration and Deployment Services. Vous pouvez éviter cette situation en renommant le modèle généré dans chaque itération pour éviter des références ambiguës aux modèles ou en effaçant le modèle actuel (par exemple, en ajoutant une instruction `clear generated palette`) avant la fin de la boucle.

Nom des types de sortie

Le tableau suivant répertorie tous les types d'objet de sortie et les noeuds qui les génèrent. Pour obtenir la liste complète des formats d'exportation disponibles pour chaque type d'objet de sortie, reportez-vous à la description des propriétés propres au noeud qui génère le type de sortie en question, dans Propriétés communes aux noeuds Graphiques et Propriétés des noeuds de sortie.

Tableau 237. Types d'objet de sortie et noeuds qui les génèrent.

Type d'objet de sortie	Noeud
analysisoutput	Analyse
collectionoutput	Collecte
dataauditoutput	Audit données
distributionoutput	Proportion
evaluationoutput	Evaluation
histogramoutput	Histogramme
matrixoutput	Matrice
meansoutput	Moyennes
multiplotoutput	Courbes

Tableau 237. Types d'objet de sortie et noeuds qui les génèrent (suite).

Type d'objet de sortie	**Noeud**
plotoutput	Tracé
qualityoutput	Qualité
reportdocumentoutput	Ce type d'objet n'est pas issu d'un noeud ; il s'agit en fait de la sortie créée par un rapport de projet.
reportoutput	Rapport
statisticsprocedureoutput	Sortie Statistiques
statisticsoutput	Statistiques
tableoutput	Table
timeplotoutput	Tracé horaire
weboutput	Web

Annexe B. Migration du scriptage existant au scriptage Python

Présentation de la migration de script existant

Cette section présente un récapitulatif des différences entre le scriptage Python et le scriptage existant dans IBM SPSS Modeler. Elle fournit également des informations sur la façon de migrer vos scripts existants vers des scripts Python. Dans cette section, vous trouverez une liste des commandes existantes SPSS Modeler standard et les commandes Python équivalentes.

Différences générales

Le design du scriptage existant doit beaucoup aux scripts de commande OS. Le scriptage existant est orienté ligne et, bien qu'il y ait quelques structures de bloc (par exemple `if...then...else...endif` et `for...endfor`), l'indentation n'est généralement pas significative.

Dans le scriptage Python, l'indentation est significative et les lignes appartenant au même bloc logique doivent être mises en retrait au même niveau.

Remarque : Vous devez faire attention lorsque vous copiez et collez du code Python. Une ligne qui est mise en retrait à l'aide de tabulations peut avoir la même présentation dans l'éditeur qu'une ligne mise en retrait à l'aide d'espaces. Toutefois, le script Python générera une erreur car les lignes ne sont pas considérées comme ayant la même indentation.

Contexte de scriptage)

Le contexte de scriptage (ou génération de scripts) définit l'environnement dans lequel le script est exécuté (par exemple, le flux ou le super noeud qui exécute le script). Dans le scriptage existant, le contexte est implicite, ce qui signifie par exemple que toute référence de noeud dans un script de flux est supposé se trouver dans le flux qui exécute le script.

Dans le scriptage Python, le contexte de scriptage est fourni explicitement via le module `modeler.script`. Par exemple, un script de flux Python peut accéder au flux qui exécute le script via le code suivant :

```
s = modeler.script.stream()
```

Les fonctions liées au flux peuvent ensuite être invoquées via l'objet renvoyé.

Commandes et fonctions

Le scriptage existant est orienté commande. Cela signifie que chaque ligne de script commence généralement par la commande à exécuter, cette dernière étant suivie des paramètres ; par exemple :

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

Python utilise des fonctions qui sont généralement invoquées via un objet (module, classe ou objet) définissant la fonction ; par exemple :

```
stream = modeler.script.stream()
typenode = stream.findByType("type", "Type)
filternode = stream.findByType("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

Littéraux et commentaires

Certaines commandes de littéraux et commentaires couramment utilisées dans IBM SPSS Modeler présentent des commandes équivalentes dans le scriptage Python. Cela peut vous aider à convertir vos scripts SPSS Modeler existants en scripts Python afin de les utiliser dans IBM SPSS Modeler 17.

Tableau 238. Mappage du scriptage existant au scriptage Python pour les littéraux et commentaires.

Scriptage existant	Scriptage Python
Entier, par exemple `4`	Identique
Valeur flottante, par exemple `0.003`	Identique
Chaînes entre guillemets simples, par exemple `'Hello'`	Identique **Remarque :** Les littéraux chaîne qui contiennent des caractères non-ASCII doivent être précédés d'un `u` afin d'être sûr qu'ils soient représentés en Unicode.
Chaînes entre guillemets doubles, par exemple `"Hello again"`	Identique **Remarque :** Les littéraux chaîne qui contiennent des caractères non-ASCII doivent être précédés d'un `u` afin d'être sûr qu'ils soient représentés en Unicode.
Chaînes longues, par exemple `"""This is a string that spans multiple lines"""`	Identique
Listes, par exemple `[1 2 3]`	`[1, 2, 3]`
Référence de variable, par exemple `set x = 3`	`x = 3`
Continuation de ligne (\), par exemple `set x = [1 2 \ 3 4]`	`x = [ 1, 2,\ 3, 4]`
Commentaire de bloc, par exemple `/* This is a long comment over a line. */`	`""" This is a long comment over a line. """`
Commentaire de ligne, par exemple `set x = 3 # make x 3`	`x = 3 # make x 3`
`undef`	`None`
`true`	`True`
`false`	`False`

Opérateurs

Certaines commandes d'opérateur couramment utilisées dans IBM SPSS Modeler présentent des commandes équivalentes dans le scriptage Python. Cela peut vous aider à convertir vos scripts SPSS Modeler existants en scripts Python afin de les utiliser dans IBM SPSS Modeler 17.

Tableau 239. Mappage du scriptage existant au scriptage Python pour les opérateurs.

Scriptage existant	Scriptage Python

`NUM1 + NUM2`
`LIST + ITEM`
`LIST1 + LIST2`
	`NUM1 + NUM2`
	`LIST.append(ITEM)`
	`LIST1.extend(LIST2)`

`NUM1 – NUM2` `LIST - ITEM`	`NUM1 – NUM2` `LIST.remove(ITEM)`
`NUM1 * NUM2`	`NUM1 * NUM2`

Tableau 239. Mappage du scriptage existant au scriptage Python pour les opérateurs (suite).

Scriptage existant	Scriptage Python
NUM1 / NUM2	NUM1 / NUM2
= ==	==
/= /==	!=
X ** Y	X ** Y
X < Y X <= Y X > Y X >= Y	X < Y X <= Y X > Y X >= Y
X div Y X rem Y X mod Y	X // Y X % Y X % Y
et ou not(EXPR)	et ou not EXPR

Commandes conditionnelles et de bouclage

Certaines commandes conditionnelles et de bouclage couramment utilisées dans IBM SPSS Modeler présentent des commandes équivalentes dans le scriptage Python. Cela peut vous aider à convertir vos scripts SPSS Modeler existants en scripts Python afin de les utiliser dans IBM SPSS Modeler 17.

Tableau 240. Mappage du scriptage existant au scriptage Python pour les commandes conditionnelles et de bouclage.

Scriptage existant	Scriptage Python

for VAR from INT1 to INT2
...
endfor
	for VAR in range(INT1, INT2):
	...
	ou
	VAR = INT1
	while VAR <= INT2:
	...
	VAR += 1

for VAR in LIST ... endfor	for VAR in LIST: ...
for VAR in_fields_to NODE ... endfor	for VAR in NODE.getInputDataModel(): ...
for VAR in_fields_at NODE ... endfor	for VAR in NODE.getOutputDataModel(): ...

if...then
...
elseif...then
...
else
...
endif
	if ...:
	...
	elif ...:
	...
	else:
	...

with TYPE OBJECT ... endwith	Pas d'équivalent

Tableau 240. Mappage du scriptage existant au scriptage Python pour les commandes conditionnelles et de bouclage (suite).

Scriptage existant	Scriptage Python
`var VAR1`	La déclaration de variable n'est pas requise

Variables

Dans le scriptage existant, les variables sont déclarées avant d'être référencées ; par exemple :

```
var mynode
set mynode = create typenode at 96 96
```

Dans le scriptage Python, les variables sont déclarées lors de leur premier référencement ; par exemple :

```
mynode = stream.createAt("type", "Type", 96, 96)
```

Dans le scriptage existant, les références aux variables doivent être explicitement supprimées via l'opérateur ^ ; par exemple :

```
var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

Comme dans la plupart des langages de scriptage, cette opération n'est pas nécessaire dans Python ; par exemple :

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction","Age","Input")
```

Types de noeuds, de sorties et de modèles

Dans le scriptage existant, les différents types d'objets (noeud, sortie et modèle) sont généralement ajoutés à l'objet concerné. Par exemple, le noeud Dériver possède le type `derivenode` :

```
set feature_name_node = create derivenode at 96 96
```

L'API IBM SPSS Modeler dans le langage Python n'inclut pas le suffixe `node` ; le noeud Dériver possède donc le type `derive` ; par exemple :

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

La seule différence entre les noms de type dans le scriptage existant et le scriptage Python réside dans l'absence de suffixe de type.

Noms de propriétés

Les noms de propriétés sont identiques dans le scriptage existant et le scriptage Python. Par exemple, dans le noeud Délimité, la propriété qui définit l'emplacement du fichier est `full_filename` dans les deux environnements de scriptage.

Références de noeud

De nombreux scripts existants utilisent une recherche implicite pour rechercher le noeud à modifier et y accéder. Par exemple, les commandes suivantes recherchent dans le flux en cours un noeud Type ayant le libellé "Type", puis définissent la direction (ou le rôle de modélisation) du champ "Age" sur Input et du champ "Drug" sur Target (qui constitue la valeur à prévoir) :

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

Dans le scriptage Python, les objets de noeud doivent être localisés de façon explicite avant d'appeler la fonction en vue de définir la valeur de la propriété ; par exemple :

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

Remarque : Dans ce cas, "Target" doit être entre guillemets.

Les scripts Python peuvent aussi utiliser l'énumération `ModelingRole` dans le pack `modeler.api`.

Bien que la version du scriptage Python puisse être plus prolixe, elle conduit à des meilleures performances d'exécution car la recherche du noeud n'est généralement effectuée qu'une seule fois. Dans l'exemple de scriptage existant, la recherche du noeud est effectuée pour chaque commande.

La recherche de noeuds par ID est également prise en charge (l'ID de noeud est visible dans l'onglet Annotations de la boîte de dialogue de noeud). Par exemple, dans le scriptage existant :

```
# id65EMPB9VL87 est l'ID d'un noeud Type
set @id65EMPB9VL87.direction."Age" = Input
```

Le script suivant présente le même exemple en Python :

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Extraction et définition de propriétés

Le scriptage existant utilise la commande `set` pour affecter une valeur. Le terme suivant la commande `set` peut être une définition de propriété. Le script suivant présente deux formats de script possibles pour définir une propriété :

```
set <référence_noeud>.<propriété> = <valeur>
set <référence_noeud>.<propriété_saisie>.<clé> = <valeur>
```

Dans le scriptage Python, on obtient le même résultat en utilisant les fonctions `setPropertyValue()` et `setKeyedPropertyValue()` ; par exemple :

```
objet.setPropertyValue(propriété, valeur)
objet.setKeyedPropertyValue(propriété_saisie, clé, valeur)
```

Dans le scriptage existant, on peut accéder aux valeurs de propriété en utilisant la commande `get` ; par exemple :

```
var n v
set n = get node :filternode
set v = ^n.name
```

Dans le scriptage Python, on obtient le même résultat en utilisant la fonction `getPropertyValue()` ; par exemple :

```
n = stream.findByType("filter", None)
v = n.getPropertyValue("name")
```

Edition de flux

Dans le scriptage existant, la commande `create` est utilisée pour créer un noeud ; par exemple :

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

Dans le scriptage Python, les flux possèdent diverses méthodes de création de noeuds ; par exemple :

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

Dans le scriptage existant, la commande `connect` est utilisée pour créer des liens entre les noeuds ; par exemple :

```
connect ^agg to ^select
```

Dans le scriptage Python, c'est la méthode `link` qui est utilisée pour créer des liens entre les noeuds ; par exemple :

```
stream.link(agg, select)
```

Dans le scriptage existant, la commande `disconnect` est utilisée pour supprimer des liens entre les noeuds ; par exemple :

```
disconnect ^agg from ^select
```

Dans le scriptage Python, c'est la méthode `unlink` qui est utilisée pour supprimer des liens entre les noeuds ; par exemple :

```
stream.unlink(agg, select)
```

Dans le scriptage existant, la commande `position` est utilisée pour positionner les noeuds dans les canevas de flux ou entre d'autres noeuds ; par exemple :

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

Dans le scriptage Python, on obtient le même résultat en utilisant deux méthodes distinctes : `setXYPosition` et `setPositionBetween`. Par exemple :

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

Opérations de noeud

Certaines commandes d'opération de noeud couramment utilisées dans IBM SPSS Modeler présentent des commandes équivalentes dans le scriptage Python. Cela peut vous aider à convertir vos scripts SPSS Modeler existants en scripts Python afin de les utiliser dans IBM SPSS Modeler 17.

Tableau 241. Mappage du scriptage existant au scriptage Python pour les opérations de noeud.

Scriptage existant	Scriptage Python
`create` *nodespec* `at x y`	*stream*`.create(`*type*`, `*nom*`)` *stream*`.createAt(`*type*`, `*nom*`, x, y)` *stream*`.createBetween(`*type*`, `*nom*`, preNode, postNode)` *stream*`.createModelApplier(`*modèle*`, `*nom*`)`
`connect` *noeud_source* `to` *noeud_cible*	*stream*`.link(`*noeud_source*`, `*noeud_cible*`)`
`delete` *noeud*	*stream*`.delete(`*noeud*`)`
`disable` *noeud*	*stream*`.setEnabled(`*noeud*`, False)`
`enable` *noeud*	*stream*`.setEnabled(`*noeud*`, True)`
`disconnect` *noeud_source* `from` *noeud_cible*	*stream*`.unlink(`*noeud_source*`, `*noeud_cible*`)` *stream*`.disconnect(`*noeud*`)`
`duplicate` *noeud*	*noeud*`.duplicate()`
`execute` *noeud*	*stream*`.runSelected(`*noeuds*`, `*résultats*`)` *stream*`.runAll(`*résultats*`)`
`flush` *noeud*	*noeud*`.flushCache()`
`position` *noeud* `at x y`	*noeud*`.setXYPosition(x, y)`

Tableau 241. Mappage du scriptage existant au scriptage Python pour les opérations de noeud (suite).

Scriptage existant	Scriptage Python
position *noeud* between *noeud1* and *noeud2*	*noeud*.setPositionBetween(*noeud1*, *noeud2*)
rename *noeud* as *nom*	*noeud*.setLabel(*nom*)

Bouclage

Dans le scriptage existant, il existe principalement deux options de bouclage prises en charge :

- Les boucles *comptabilisées* dans lesquelles une variable d'index varie entre deux limites entières.
- Les boucles *en séquence* qui forment une boucle via une séquence de valeurs, en liant la valeur en cours à la variable de boucle.

Le script suivant est un exemple de boucle comptabilisée dans le scriptage existant :

```
for i from 1 to 10
 println ^i
endfor
```

Le script suivant est un exemple de boucle en séquence dans le scriptage existant :

```
var items
set items = [a b c d]

for i in items
 println ^i
endfor
```

Il existe également d'autres types de boucles pouvant être utilisés :

- Itération par les modèles de la palette de modèles ou par les sorties de la palette de sorties.
- Itération par les champs entrant dans un noeud ou en sortant.

Le scriptage Python prend également en charge d'autres types de boucles. Le script suivant est un exemple de boucle comptabilisée dans le scriptage Python :

```
i = 1
while i <= 10:
 print i
 i += 1
```

Le script suivant est un exemple de boucle en séquence dans le scriptage Python :

```
items = ["a", "b", "c", "d"]
for i in items:
 print i
```

La boucle en séquence est très flexible et lorsqu'elle est combinée avec les méthodes API IBM SPSS Modeler, elle peut prendre en charge la majorité des cas d'utilisation du scriptage existant. L'exemple suivant montre comment utiliser une boucle en séquence dans le scriptage Python pour itérer via les champs sortant d'un noeud :

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
 print column.getColumnName()
```

exécution des flux

Durant l'exécution d'un flux, les objets de modèle ou de sortie générés sont ajoutés à l'un des gestionnaires d'objets. Dans le scriptage existant, le script doit soit localiser les objets générés dans le gestionnaire d'objets, soit accéder à la dernière sortie générée à partir du noeud ayant généré cette sortie.

L'exécution d'un flux est différente dans Python : tout objet de modèle ou de sortie généré par l'exécution est renvoyé dans une liste transmise à la fonction d'exécution. Cela simplifie l'accès aux résultats de l'exécution de flux.

Le scriptage existant prend en charge trois commandes d'exécution de flux :

- `execute_all` exécute tous les noeuds terminaux exécutables du flux.
- `execute_script` exécute le script de flux quelle que soit la configuration de l'exécution de script.
- `execute` *`noeud`* exécute le noeud spécifié.

Le scriptage Python prend en charge une série de fonctions similaire :

- *`stream`*`.runAll(`*`liste_résultats`*`)` exécute tous les noeuds terminaux exécutables du flux.
- *`stream`*`.runScript(`*`liste_résultats`*`)` exécute le script de flux quelle que soit la configuration de l'exécution de script.
- *`stream`*`.runSelected(`*`grappe_de_noeuds`*`, `*`liste_résultats`*`)` exécute l'ensemble de noeuds spécifié dans l'ordre dans lequel ils sont fournis.
- *`stream`*`.run(`*`liste_résultats`*`)` exécute le noeud spécifié.

Dans le scriptage existant, une exécution de flux peut être arrêtée via la commande `exit` avec un code entier facultatif ; par exemple:

```
exit 1
```

Dans le scriptage Python, on obtient le même résultat à l'aide du script suivant :

```
modeler.script.exit(1)
```

Accéder aux objets via le système de fichiers et le référentiel

Dans le scriptage existant, vous pouvez ouvrir un objet de flux, de modèle ou de sortie existant à l'aide de la commande `open` ; par exemple :

```
var s
set s = open stream "c:/my streams/modeling.str"
```

Dans le scriptage Python, il existe une classe `TaskRunner` qui est accessible à partir de la session et qui peut être utilisée pour effectuer des tâches similaires ; par exemple :

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

Pour enregistrer un objet dans le scriptage existant, vous pouvez utiliser la commande `save` ; par exemple :

```
save stream s as "c:/my streams/new_modeling.str"
```

L'approche de script Python équivalente consisterait à utiliser la classe `TaskRunner` ; par exemple :

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

Les opérations basées sur IBM SPSS Collaboration and Deployment Services Repository sont prises en charge dans le scriptage existant via les commandes `retrieve` et `store` ; par exemple :

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

Dans le scriptage Python, l'utilisateur accéderait à la fonctionnalité équivalente via l'objet Référentiel associé à la session ; par exemple:

```
session = modeler.script.session()
repo = session.getRepository()
s = repo.retrieveStream("/my repository folder/my_stream.str", None, None, True)
repo.storeStream(s, "/my repository folder/my_stream_copy.str", None)
```

Remarque : L'accès via l'objet Référentiel requiert que la session ait été configurée avec une connexion de référentiel valide.

Opérations de flux

Certaines commandes d'opération de flux couramment utilisées dans IBM SPSS Modeler présentent des commandes équivalentes dans le scriptage Python. Cela peut vous aider à convertir vos scripts SPSS Modeler existants en scripts Python afin de les utiliser dans IBM SPSS Modeler 17.

Tableau 242. Mappage du scriptage existant au scriptage Python pour les opérations de flux.

Scriptage existant	Scriptage Python
`create stream NOMFICHIER_PAR_DEFAUT`	*taskrunner*.createStream(*name*, *autoConnect*, *autoManage*)
`close stream`	*stream*.close()
`clear stream`	*stream*.clear()
`get stream stream`	Pas d'équivalent
`load stream path`	Pas d'équivalent
`open stream path`	*taskrunner*.openStreamFromFile(*path*, *autoManage*)
`save stream as path`	*taskrunner*.saveStreamToFile(*stream*, *path*)
`retreive stream path`	*repository*.retreiveStream(*path*, *version*, *label*, *autoManage*)
`store stream as path`	*repository*.storeStream(*stream*, *path*, *label*)

Opérations de modèle

Certaines commandes d'opération de modèle couramment utilisées dans IBM SPSS Modeler présentent des commandes équivalentes dans le scriptage Python. Cela peut vous aider à convertir vos scripts SPSS Modeler existants en scripts Python afin de les utiliser dans IBM SPSS Modeler 17.

Tableau 243. Mappage du scriptage existant au scriptage Python pour les opérations de modèle.

Scriptage existant	Scriptage Python
`open model chemin`	*taskrunner*.openModelFromFile(*path*, *autoManage*)
`save model as path`	*taskrunner*.saveModelToFile(*model*, *path*)
`retrieve model path`	*repository*.retrieveModel(*path*, *version*, *label*, *autoManage*)
`store model as path`	*repository*.storeModel(*model*, *path*, *label*)

Opérations de sortie de document

Certaines commandes d'opération de sortie de document couramment utilisées dans IBM SPSS Modeler présentent des commandes équivalentes dans le scriptage Python. Cela peut vous aider à convertir vos scripts SPSS Modeler existants en scripts Python afin de les utiliser dans IBM SPSS Modeler 17.

Tableau 244. Mappage du scriptage existant au scriptage Python pour les opérations de sortie de document.

Scriptage existant	Scriptage Python
`open output path`	*taskrunner*.openDocumentFromFile(*path*, *autoManage*)

Tableau 244. Mappage du scriptage existant au scriptage Python pour les opérations de sortie de document (suite).

Scriptage existant	Scriptage Python
`save output as path`	`taskrunner.saveDocumentToFile(output, path)`
`retrieve output path`	`repository.retrieveDocument(path, version, label, autoManage)`
`store output as path`	`repository.storeDocument(output, path, label)`

Autres différences entre le scriptage existant et le scriptage Python

Les scripts existants prennent en charge la manipulation des projets IBM SPSS Modeler. Les scripts Python ne prennent pas en charge cette fonction.

Le scriptage existant offre une prise en charge du chargement des objets d'*état* (combinaison de flux et de modèles). Ce type d'objet est devenu obsolète depuis la version 8.0 de IBM SPSS Modeler. Le scriptage Python ne prend pas en charge les objets d'état.

Le scriptage Python offre les fonctionnalités supplémentaires suivantes, qui ne sont pas disponibles dans le scriptage existant :

- Définitions de classe et de fonction
- Gestion des erreurs
- Support plus sophistiqué des entrées et sorties
- Modules externes et tiers

Remarques

Ces informations ont été développées pour les produits et services offerts dans le monde.

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, programme ou service IBM n'implique pas que seul ce produit, programme ou service IBM puisse être utilisé. Tout produit, programme ou service fonctionnellement équivalent peut être utilisé s'il n'enfreint aucun droit de propriété intellectuelle d'IBM. Cependant l'utilisateur doit évaluer et vérifier l'utilisation d'un produit, programme ou service non IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. L'octroi de ce document n'équivaut aucunement à celui d'une licence pour ces brevets. Vous pouvez envoyer par écrit des questions concernant la licence à :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Pour le Canada, veuillez adresser votre courrier à :
IBM Director of Commercial Relations
IBM Canada Ltd.
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7
Canada

Pour toute demande au sujet des licences concernant les jeux de caractères codés sur deux octets (DBCS), contactez le service Propriété intellectuelle IBM de votre pays ou adressez vos questions par écrit à :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japon

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales. LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE. Certains états n'autorisent pas l'exclusion de garanties explicites ou implicites lors de certaines transactions, par conséquent, il est possible que cet énoncé ne vous concerne pas.

Ces informations peuvent contenir des erreurs techniques ou des erreurs typographiques. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Toute référence dans ces informations à des sites Web autres qu'IBM est fournie dans un but pratique uniquement et ne sert en aucun cas de recommandation pour ces sites Web. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation à votre égard, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Software Group
ATTN: Licensing
200 W. Madison St.
Chicago, IL; 60606
U.S.A.

Ces informations peuvent être disponibles, soumises à des conditions générales, et dans certains cas payantes.

Le programme sous licence décrit dans le présent document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions du Livret Contractuel IBM, des Conditions internationales d'utilisation des Logiciels IBM ou de tout autre contrat équivalent.

Toutes les données sur les performances contenues dans le présent document ont été obtenues dans un environnement contrôlé. Par conséquent, les résultats obtenus dans d'autres environnements d'exploitation peuvent varier de manière significative. Certaines mesures peuvent avoir été effectuées sur des systèmes en cours de développement et il est impossible de garantir que ces mesures seront les mêmes sur les systèmes commercialisés. De plus, certaines mesures peuvent avoir été estimées par extrapolation. Les résultats réels peuvent être différents. Les utilisateurs de ce document doivent vérifier les données applicables à leur environnement spécifique.

les informations concernant les produits autres qu'IBM ont été obtenues auprès des fabricants de ces produits, leurs annonces publiques ou d'autres sources publiques disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Aucune réclamation relative à des produits non IBM ne pourra être reçue par IBM. Les questions sur les capacités de produits autres qu'IBM doivent être adressées aux fabricants de ces produits.

Toutes les déclarations concernant la direction ou les intentions futures d'IBM peuvent être modifiées ou retirées sans avertissement préalable et représentent uniquement des buts et des objectifs.

Ces informations contiennent des exemples de données et de rapports utilisés au cours d'opérations quotidiennes standard. Pour les illustrer le mieux possible, ces exemples contiennent des noms d'individus, d'entreprises, de marques et de produits. Tous ces noms sont fictifs et toute ressemblance avec des noms et des adresses utilisés par une entreprise réelle ne serait que pure coïncidence.

Si vous consultez la version papier de ces informations, il est possible que certaines photographies et illustrations en couleurs n'apparaissent pas.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines dans de nombreux pays. Les autres noms de produits et de services peuvent être des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web "Copyright and trademark information" à l'adresse www.ibm.com/legal/copytrade.shtml.

Intel, le logo Intel, Intel Inside, le logo Intel Inside, Intel Centrino, le logo Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium, et Pentium sont des marques commerciales ou des marques déposées de Intel Corporation ou de ses filiales aux Etats-Unis et dans d'autres pays.

Linux est une marque déposée de Linus Torvalds aux Etats-Unis et/ou dans d'autres pays.

Microsoft, Windows, Windows NT et le logo Windows sont des marques commerciales de Microsoft Corporation aux Etats-Unis et/ou dans d'autres pays.

UNIX est une marque déposée de The Open Group aux Etats-Unis et dans d'autres pays.

Java ainsi que tous les logos et toutes les marques incluant Java sont des marques d'Oracle et/ou de ses sociétés affiliées.

Les autres noms de produits et de services peuvent être des marques d'IBM ou d'autres sociétés.

Index

M

N

O

P

R

S

T

V