バージョン18

スクリプト構文リファレンス

「真の発見の旅とは、新しい風景を探すことではなく、新たな視点を持つことである。」

マルセル・プルースト

JMP Statistical Discovery LLC
920 SAS Campus Drive
Cary, North Carolina 27513-2414

The correct bibliographic citation for this manual is as follows: JMP Statistical Discovery LLC 2024. *JMP® 18 JSL Syntax Reference*. Cary, NC: JMP Statistical Discovery LLC

JMP® 18 JSL Syntax Reference

JMP Statistical Discovery LLC, 920 SAS Campus Drive, Cary, North Carolina 27513-2414.

March 2024

目次
スクリプト構文リファレンス

1 JMP の概要 9
マニュアルとその他のリソース
JMP Pro 11
JMP オンラインヘルプ 11
ドキュメンテーション PDF アドイン 11
JMP ヘルプメニュー 19
JMP を習得するためのその他のリソース 20
JMP を使い始める 20
JMP の検索 20
サンプルデータテーブル 21
JSL の概要 21
JMP のツールヒント 21
JMP User Community 22
オンラインの統計的思考コース（無料） 22
JMP New User Welcome Kit 22
統計ナレッジポータル 22
JMP トレーニング 22
JMP 関連書籍 23
「JMP スターター」ウィンドウ 23
JMP テクニカルサポート 23

2 JSL の関数、演算子、およびメッセージ 25
コマンドの概要
割り当て関数 27
文字関数 30
文字パターン関数 44
コメント関数 54
比較関数 56
条件付き関数と論理関数 60
定数関数 72
日付と時間関数 73
離散型確率関数 81
表示関数 87

式の関数 ... 121
ファイル関数 ... 123
財務関数 ... 145
グラフ関数 ... 150
HTTP関数 ... 170
リスト関数 ... 170
MATLABインテグレーション関数 ... 177
MATLAB JSL関数インターフェース ... 177
行列関数 ... 184
数値関数 ... 210
最適化関数 ... 213
連続型確率関数 ... 216
プログラミング関数 ... 245
Pythonインテグレーション関数 ... 264
Rインテグレーション関数 ... 269
乱数関数 ... 275
行関数 ... 285
行の属性関数 ... 291
SQL関数 ... 295
統計関数 ... 297
超越関数 ... 312
三角関数 ... 318
ユーティリティ関数 ... 321

3 JSLメッセージ ... 357
オブジェクトおよびディスプレイボックスのメッセージの概要
アルファシェイプのメッセージ ... 360
連想配列のメッセージ ... 360
クラスのメッセージ ... 361
データコネクタのメッセージ ... 363
データコネクタメタデータのメッセージ ... 363
データコネクタレジストリのメッセージ ... 364
データテーブルのメッセージ ... 365
データテーブル全体のメッセージ ... 365
列のメッセージ ... 394
行のメッセージ ... 401
データフィルタのメッセージ ... 402
データフィードのメッセージ(Windowsのみ) ... 406
ディスプレイボックスのメッセージ ... 408
すべてのディスプレイボックスのメッセージ ... 408

Axis Box のメッセージ . . . 419
Border Box のメッセージ . . . 424
Data Browser Box のメッセージ . . . 425
Data Filter Source Box のメッセージ . . . 425
Frame Box のメッセージ . . . 425
Graph 3D Box のメッセージ . . . 427
Excerpt Box のメッセージ . . . 427
Filter Col Selector のメッセージ . . . 428
Global Box のメッセージ . . . 428
Hier Box のメッセージ . . . 428
Matrix Box のメッセージ . . . 428
Nom Axis Box のメッセージ . . . 429
Number Col Box のメッセージ . . . 429
Number Col Edit Box のメッセージ . . . 432
Number Edit Box のメッセージ . . . 433
Outline Box のメッセージ . . . 433
Panel Box のメッセージ . . . 434
Plot Col Box のメッセージ . . . 434
Slider Box／Range Slider Box のメッセージ . . . 435
String Col Box のメッセージ . . . 436
Tab Box のメッセージ . . . 438
Table Box のメッセージ . . . 438
Text Box のメッセージ . . . 442
Tree Node と Tree Box のメッセージ . . . 443
三角分割のメッセージ . . . 445
Window のメッセージ . . . 447
ダイナミックリンクライブラリ（DLL）のメッセージ . . . 449
イメージのメッセージ . . . 451
インタラクティブ HTML のメッセージ . . . 454
Web レポートのメッセージ . . . 454
JMP アプリケーションのメッセージ . . . 455
JMP App のメッセージ . . . 455
JMP App Module のメッセージ . . . 457
JMP App Module Instance のメッセージ . . . 458
MATLAB のメッセージ . . . 458
名前空間のメッセージ . . . 462
PI サーバーからの読み込みメッセージ . . . 464
PI サーバーのメッセージ . . . 464
Importer メッセージ . . . 465
Python インテグレーションのメッセージ . . . 467

Rインテグレーションのメッセージ 472
スケジュールのメッセージ 475
セグメントのメッセージ 476
ソケットのメッセージ 476
SQLのメッセージ 480
その他のオブジェクトのメッセージ 483
Zipアーカイブ 483
ジャーナル 484

A JMPクエリーで使用可能なSQL関数 487

SQLの数値関数 489
SQLの日付時間関数 490
SQLの文字列関数 493
SQLのシステム関数 494
SQLの集計関数 495

B 参考文献 497

第1章

JMPの概要
マニュアルとその他のリソース

「JMPの概要」マニュアル。JMP Pro限定機能、JMPドキュメンテーションアドイン、各JMPドキュメントの説明、ヘルプメニューのオプション、その他のサポートの場所などを紹介しています。

目次

JMP Pro 11
JMP オンラインヘルプ 11
ドキュメンテーション PDF アドイン 11
JMP ヘルプメニュー 19
JMP を習得するためのその他のリソース 20
JMP を使い始める 20
JMP の検索 20
サンプルデータテーブル 21
JSL の概要 21
JMP のツールヒント 21
JMP User Community 22
オンラインの統計的思考コース（無料） 22
JMP New User Welcome Kit 22
統計ナレッジポータル 22
JMP トレーニング 22
JMP 関連書籍 23
「JMP スターター」ウィンドウ 23
JMP テクニカルサポート 23

JMP Pro

JMP Proの限定機能には、JMP Proアイコン JMP PRO が付いています。JMP Proの機能の概要については https://www.jmp.com/software/pro/ をご覧ください。

JMP オンラインヘルプ

JMPオンラインヘルプでは、JMPの機能や統計手法、JMPスクリプト言語（**JSL**）に関する情報が検索できます。JMPオンラインヘルプは、次のような方法で開きます。

- Windowsで、［**ヘルプ**］**>**［**JMP オンラインヘルプ**］を選択します。
- macOSで、［**ヘルプ**］**>**［**JMP ヘルプ**］を選択します。
- WindowsでF1キーを押します。
- データテーブルやレポートウィンドウにおける特定の部分に関するヘルプを表示するには、［**ヘルプ**］**>**［**ヘルプツール**］を選択します。その後、データテーブルやレポートウィンドウの任意の箇所をクリックします。ヘルプツールを無効にするには、Escキーを押します。
- JMPウィンドウ内で［**ヘルプ**］ボタンをクリックします。

メモ: JMPヘルプを使用するにはインターネット接続が必要です。インターネット接続がない場合は、ドキュメンテーションアドインをインストールする方法があります。詳細については、「ドキュメンテーションPDFアドイン」を参照してください。

ドキュメンテーションPDFアドイン

JMPドキュメンテーションアドインをダウンロードし、インストールすることができます。このドキュメンテーションアドインには、JMPライブラリの個々のドキュメントをPDFファイルとして保存したものと、『JMPドキュメンテーションライブラリ』ファイルが含まれています。『JMPドキュメンテーションライブラリ』ファイルは、個々のドキュメントのPDFファイルを1つにまとめたPDFファイルです。JMPオンラインヘルプのように、1つのPDFファイルで全マニュアル内を検索することができます。

このアドインをインストールすると、PDFファイルがコンピュータにインストールされ、［ヘルプ］メニューに［ドキュメンテーションPDF］オプションが表示されます。［**ヘルプ**］**>**［**ドキュメンテーションPDF**］を選択して、ローカルコンピュータ上のドキュメンテーションを使用することが可能になります。ドキュメンテーションアドインは、https://www.jmp.com/doc-addin でダウンロードしてください。

以下の表は、ドキュメンテーションアドインに含まれている各マニュアルの目的および内容を要約したものです。

マニュアル	目的	内容
『JMPドキュメンテーションライブラリ』	個々のマニュアルのPDFファイルを1つにまとめたPDFファイルです。	1つのPDFにすべてのJMPドキュメンテーションが含まれています。
『はじめてのJMP』	JMPをあまりご存知ない方を対象とした入門ガイド	JMPを紹介し、データを作成・分析する方法や、結果を共有する方法を解説します。
『JMPの使用法』	JMPのデータテーブルと、基本操作を理解する	データの読み込み、列プロパティの変更、データの並べ替え、ワークフロービルダーの使い方など、JMP全体で使用する一般的な概念や機能を説明しています。
『基本的な統計分析』	このマニュアルを見ながら、基本的な分析を行う	［分析］メニューにある以下のプラットフォームについて説明しています。 • 一変量の分布 • 二変量の関係 • 表の作成 • テキストエクスプローラ ［分析］>［二変量の関係］で二変量分析、一元配置分散分析、分割表分析を実行する方法を説明しています。ブートストラップによる標本分布の近似方法や、乱数シミュレーションによるパラメトリックな標本再抽出の実行方法も取り上げています。

マニュアル	目的	内容
『グラフ機能』	データに合った理想的なグラフを見つける	［グラフ］メニューにある以下のプラットフォームについて説明しています。 • グラフビルダー • 三次元散布図 • 等高線図 • バブルプロット • パラレルプロット • セルプロット • 散布図行列 • 三角図 • ツリーマップ • チャート • 重ね合わせプロット このマニュアルには背景地図や独自の地図の作成方法も記載されています。
『プロファイル機能』	対話式のプロファイルツールの使い方を学ぶ。任意の応答曲面の断面を表示できるようになります。	［グラフ］メニューで用意されているすべてのプロファイルについて。誤差因子の分析が、ランダムなインプットを使用したシミュレーションの実行とともに含まれています。
『実験計画 (DOE)』	実験の計画方法と適切な標本サイズの決定方法を学ぶ	［実験計画 (DOE)］メニューで用意されているすべての機能について。

マニュアル	目的	内容
『基本的な回帰モデル』	「モデルのあてはめ」プラットフォームとその多くの手法について学ぶ	［分析］メニューの「モデルのあてはめ」プラットフォームで使用できる、以下の手法について説明しています。 • 標準最小2乗 • ステップワイズ法 • 一般化回帰 • 混合モデル • 一般化線形混合モデル • MANOVA • 対数線形-分散 • 名義ロジスティック • 順序ロジスティック • 一般化線形モデル

マニュアル	目的	内容
『予測モデルおよび発展的なモデル』	さらなるモデリング手法について学ぶ	［分析］>［予測モデル］メニューにある以下のプラットフォームについて説明しています。 • ニューラル • パーティション • ブートストラップ森 • ブースティングツリー • K近傍法 • 単純Bayes • サポートベクトルマシン • モデルの比較 • モデルのスクリーニング • 検証列の作成 • 計算式デポ ［分析］>［発展的なモデル］メニューにある以下のプラットフォームについて説明しています。 • 曲線のあてはめ • 非線形回帰 • 関数データエクスプローラ • Gauss過程 • 時系列分析 • 時系列予測 • 対応のあるペア ［分析］>［スクリーニング］メニューにある以下のプラットフォームについて説明しています。 • 外れ値を調べる • 欠測値を調べる • パターンを調べる • 応答のスクリーニング • 説明変数のスクリーニング • アソシエーション分析 • 工程履歴エクスプローラ

マニュアル	目的	内容
『多変量分析』	複数の変数を同時に分析する方法。	［分析］>［多変量］メニューにある以下のプラットフォームについて説明しています。 • 多変量の相関 • 主成分分析 • 判別分析 • PLS回帰 • 多重対応分析 • 構造方程式モデル • 因子分析 • 多次元尺度構成 • 多変量埋め込み • 項目分析 ［分析］>［クラスター分析］メニューにある以下のプラットフォームについて説明しています。 • 階層型クラスター分析 • K Means クラスター分析 • 正規混合 • 潜在クラス分析 • 変数のクラスタリング

マニュアル	目的	内容
『品質と工程』	工程を評価し、向上させるためのツールについて学びます。	［分析］>［品質と工程］メニューにある以下のプラットフォームについて説明しています。 • 管理図ビルダーと個々の管理図 • 測定システム分析（EMP、タイプ1ゲージ） • 計量値/計数値ゲージチャート • 工程のスクリーニング • 工程能力 • モデルに基づく多変量管理図 • 旧機能の管理図 • パレート図 • 特性要因図 • 品質に関する限界の設定 • OC曲線
『信頼性/生存時間分析』	製品やシステムにおける信頼性を評価し、向上させる方法、および人や製品の生存時間データを分析する方法について学ぶ	［分析］>［信頼性/生存時間分析］メニューにある以下のプラットフォームについて説明しています。 • 寿命の一変量 • 寿命の二変量 • 累積損傷 • 疲労モデル • 再生モデルによる分析 • 反復測定劣化 • 破壊劣化 • 信頼性予測 • 信頼性成長 • 信頼性ブロック図 • 修理可能システムのシミュレーション • 生存時間分析 • 生存時間(パラメトリック)のあてはめ • 劣化分析 • 比例ハザードのあてはめ

マニュアル	目的	内容
『消費者調査』	消費者の嗜好を調査して製品・サービスの改良につなげる方法について学びます。	［分析］>［消費者調査］メニューにある以下のプラットフォームについて説明しています。 • カテゴリカル • 選択モデル • MaxDiff • アップリフト • 多重因子分析
『遺伝学』	遺伝子データを分析して育種計画をシミュレートし、最適な遺伝子交雑を予測する方法を学びます。	［分析］>［遺伝学］メニューにある以下のプラットフォームについて説明しています。 • マーカーの統計量 • マーカーのシミュレーション
『スクリプトガイド』	パワフルなJMPスクリプト言語（JSL）について学びます。	スクリプトの作成やデバッグ、データテーブルの操作、ディスプレイボックスの構築、JMPアプリケーションの作成など。
『スクリプト構文リファレンス』	JSL関数の引数やメッセージについて学びます。	JSLコマンドの構文、例、および注意書き。
『キーボードショートカット』	キーボードを使ってJMP内を移動し、タスクを実行する方法を学びます。	コマンドとそれに対応するWindowsとmacOSのショートカットを紹介しています。
『メニューの説明』	JMPのメニュー項目について学びます。	WindowsとmacOSのメニューオプションについて説明しています。

JMPヘルプメニュー

JMP 18では、［ヘルプ］メニューが一新されています。

メニュー項目	説明
JMPの検索	統計検定などの機能をJMP内で検索する。詳細については、「JMPの検索」を参照してください。
JMPオンラインヘルプ	Webブラウザ内に最新バージョンのヘルプが表示される。
ヘルプツール	データテーブルやレポートウィンドウの任意の箇所をクリックしてヘルプを表示する。
クイックスタート	(「使い方ヒント」が「クイックスタート」に生まれ変わりました。) JMPの基本がすばやく学べるように、便利なヒントを表示する。詳細については、「JMPを使い始める」を参照してください。
ドキュメンテーションPDF	コンピュータ上にあるドキュメンテーションPDFファイルにアクセスできる。詳細については、「ドキュメンテーションPDFアドイン」を参照してください。 **メモ:** このメニューオプションは、ドキュメンテーションアドインをダウンロードしてインストールした場合のみ表示されます。
JMPの機能	Webブラウザを開いてJMPにあるツールや機能を一覧表示する。オンラインヘルプのリンクも表示され、詳細を確認することができます。
JMPの学習	Webブラウザを開いて、JMPについて学習するためのリソースを表示する。JMPの使い方を説明した短いビデオなどの教材が用意されています。
JMP User Community	Webブラウザが開き、他のJMPユーザとつながって問題の解決方法やJMPの改良案などを交換できる。詳細については、「JMP User Community」を参照してください。
JMPの新機能	Webブラウザが開き、JMPの最新リリースで導入された新機能を確認できる。
サンプルデータフォルダ	サンプルデータを使って、JMPでの分析について学ぶ。サンプルデータファイルを開いてスクリプトを実行すると、サンプルの分析を見ることができます。詳細については、「サンプルデータテーブル」を参照してください。
サンプルの索引	サンプルデータを分析の種類や産業別に検索できる。学習用サンプルなどへのリンクも用意されています。詳細については、「サンプルデータテーブル」を参照してください。
スクリプトの索引	JMPのスクリプトコマンドを検索し、その使い方を学ぶ。詳細については、「JSLの概要」を参照してください。

メニュー項目	説明
My JMP	my.jmp.comサイトを開く。
バージョン情報	使用しているJMPのバージョンが表示される。ソフトウェアアップデートの有無も確認できます。このオプションはWindowsでのみ利用できます。

JMPを習得するためのその他のリソース

JMPヘルプに加え、次のリソースもJMPを学習するのに役立ちます。

- 「JMPを使い始める」
- 「JMPの検索」
- 「サンプルデータテーブル」
- 「JSLの概要」
- 「JMPのツールヒント」
- 「JMP User Community」
- 「オンラインの統計的思考コース（無料）」
- 「JMP New User Welcome Kit」
- 「統計ナレッジポータル」
- 「JMPトレーニング」
- 「JMP関連書籍」
- 「「JMPスターター」ウィンドウ」

JMPを使い始める

JMPを起動すると、最初にJMPの使い方を説明する「クイックスタート」ウィンドウが表示されます。クイックスタートが表示されないようにするには、**［起動時にクイックスタートを表示］**チェックボックスをオフにします。再び表示されるようにするには、**［ヘルプ］＞［クイックスタート］**を選択します。また、「環境設定」ウィンドウでも表示するかどうかを設定できます。

JMPの検索

ある統計手法をどこから実行すればよいかわからない場合は、JMP全体で検索しましょう。検索を起動したウィンドウ（データテーブル、レポートなど）に合わせて、結果が表示されます。

1. **［ヘルプ］＞［JMPの検索］**をクリックします。または、Ctrlキーを押しながらカンマキーを押します。
2. 検索したいテキストを入力します。

3. 探していた統計手法が含まれている結果をクリックします。

   右側にその手法の説明と場所が表示されます。

4. 該当するボタンをクリックして分析を起動するか、ヘルプを開きます。

サンプルデータテーブル

JMPのマニュアルで取り上げる例は、すべてサンプルデータを使用しています。サンプルデータのディレクトリを開くには、**［ヘルプ］＞［サンプルデータフォルダ］**を選択します。

サンプルデータテーブルを文字コード順に並べた一覧を表示する、またはカテゴリごとにサンプルデータを表示するには、**［ヘルプ］＞［サンプルの索引］**を選択します。

サンプルデータテーブルは次のディレクトリにインストールされています。

Windowsの場合: C:¥Program Files¥JMP¥JMP¥18¥Samples¥Data

macOSの場合: /Library/Application Support/JMP/18/Samples/Data

JMP Proでは、サンプルデータが（JMPではなく）JMPPROディレクトリにインストールされています。

サンプルデータの使用例を見るには、**［ヘルプ］＞［サンプルの索引］**を選択して「Teaching Examples」に移動します。

JSLの概要

JSLスクリプトに関するヘルプやスクリプトの例を表示するには、**［ヘルプ］＞［スクリプトの索引］**を選択します。［スクリプトの索引］では、JMPスクリプト言語（JSL）の関数、オブジェクト、ディスプレイボックスに関する情報を検索できます。サンプルのスクリプトに編集を加えて実行したり、コマンドに関するヘルプを表示したりできます。

JMPのツールヒント

次のような項目の上にカーソルを置くと、その項目を説明するツールヒント（または**ホバーラベル**）が表示されます。

- メニューまたはツールバーのオプション
- グラフ内のラベル
- レポートウィンドウ内の結果（テキスト）（カーソルで円を描くと表示される）
- 「ホームウィンドウ」内のファイル名またはウィンドウ名
- スクリプトエディタ内のコード

ヒント: Windowsでは、JMPの「環境設定」ウィンドウでツールヒントを表示しないよう設定できます。**［ファイル］>［環境設定］>［一般］**を選択し、**［メニューのヒントを表示］**の選択を解除します。このオプションは、macOSでは使用できません。

JMP User Community

JMP User Communityでは、さまざまな方法でJMPをさらに学習したり、他のSASユーザとのコミュニケーションを図ったりできます。ラーニングライブラリには1ページガイド、チュートリアル、デモなどが用意されており、JMPを使い始める上でとても便利です。また、JMPのさまざまなトレーニングコースに登録して、自己教育を進めることも可能です。

その他のリソースとして、ディスカッションフォーラム、サンプルデータやスクリプトファイルの交換、Webcastセミナー、ソーシャルネットワークグループなども利用できます。

WebサイトのJMPリソースにアクセスするには、**［ヘルプ］>［JMP User Community］**を選択するか、https://community.jmp.com をご覧ください。

オンラインの統計的思考コース（無料）

この無料のオンラインコースでは、「探索的データ分析」、「品質手法」、「相関と回帰」といったトピックについて、実用的な統計技術を学習できます。各コースは、短い動画、デモ、練習問題などで構成されています。https://www.jmp.com/statisticalthinking をご覧ください。

JMP New User Welcome Kit

JMP New User Welcome Kitは、JMPの基本的な使用方法をすばやく習得するのに役立ちます。30の短いデモ動画と実習をこなし、ソフトウェアの使い方を身につけ、全世界のJMPユーザが集まった最大規模のオンラインコミュニティに参加できます。https://www.jmp.com/welcome をご覧ください。

統計ナレッジポータル

統計ナレッジポータルは、わかりやすい例とグラフを使って統計を簡潔に説明し、統計の基礎知識を提供しています。https://www.jmp.com/skp をご覧ください。

JMPトレーニング

JMPでは、経験豊かなJMPのエキスパートを講師とした各種トレーニングを提供しています。パブリックコース、ライブWebセミナー、オンサイトコースをご利用いただけます。また柔軟に学べるeラーニングも用意しています。https://www.jmp.com/training をご覧ください。

JMP関連書籍

JMP関連書籍は、次のJMP Webページで紹介されています。https://www.jmp.com/books をご覧ください。

「JMPスターター」ウィンドウ

JMPまたはデータ分析にあまり慣れていないユーザは、「JMPスターター」ウィンドウから始めるとよいでしょう。カテゴリ分けされた項目には説明がついており、ボタンをクリックするだけで該当の機能を起動できます。「JMPスターター」ウィンドウには、［分析］、［グラフ］、［テーブル］、および［ファイル］メニューで用意されている多くの項目があります。また、JMP Proの機能やプラットフォームのリストも含まれています。

- 「JMPスターター」ウィンドウを開くには、**［表示］**（macOSでは**［ウィンドウ］**）**>［JMPスターター］**を選びます。
- JMPの起動時に自動的に「JMPスターター」を表示するには、**［ファイル］>［環境設定］>［一般］**を選び、「開始時のJMPウィンドウ」リストから**［JMPスターター］**を選びます。macOSでは、**［JMP］>［環境設定］>［全般］>［起動時にJMPスターターウィンドウを表示する］**を選択します。

JMPテクニカルサポート

JMPのテクニカルサポートは、JMPのエンジニアが担当し、その多くは、統計学などの技術的な分野の知識を有しています。

https://www.jmp.com/support には、テクニカルサポートへの連絡方法などが記載されています。

第2章

JSLの関数、演算子、およびメッセージ
コマンドの概要

ここでは、JMPで用意されている多くの関数と演算子について、簡単に説明しています。また、一般的なオブジェクトメッセージについても簡単に説明しています。詳細は、JMPを起動した後、[**ヘルプ**] > [**スクリプトの索引**]で表示される「スクリプトの索引」を参照してください。

プラットフォームのメッセージの詳細については、『スクリプトガイド』を参照してください。

目次

割り当て関数 . . . 27
文字関数 . . . 30
文字パターン関数 . . . 44
コメント関数 . . . 54
比較関数 . . . 56
条件付き関数と論理関数 . . . 60
定数関数 . . . 72
日付と時間関数 . . . 73
離散型確率関数 . . . 81
表示関数 . . . 87
式の関数 . . . 121
ファイル関数 . . . 123
財務関数 . . . 145
グラフ関数 . . . 150
HTTP関数 . . . 170
リスト関数 . . . 170
MATLABインテグレーション関数 . . . 177
    MATLAB JSL関数インターフェース . . . 177
行列関数 . . . 184
数値関数 . . . 210
最適化関数 . . . 213
連続型確率関数 . . . 216
プログラミング関数 . . . 245
Pythonインテグレーション関数 . . . 264
Rインテグレーション関数 . . . 269
乱数関数 . . . 275
行関数 . . . 285
行の属性関数 . . . 291
SQL関数 . . . 295
統計関数 . . . 297
超越関数 . . . 312
三角関数 . . . 318
ユーティリティ関数 . . . 321

割り当て関数

JSLには、**割り当て関数**も用意されています。割り当て関数は、演算結果を変数に直接、代入します。関数の形式で指定した場合、最初のオペランドに演算結果が割り当てられます。最も基本的な割り当て演算子は、等号が1つの=演算子です（対応する関数はAssign関数）。たとえば、*a*が3のとき、「a+=4」を実行すると、*a*が7になります。

割り当て関数の最初のオペランドは、値を割り当てることができる変数でなければなりません。このような変数は、「左辺値（*L-Value*）」と呼ばれています。たとえば、「3+=4」といった式は、「3」が単なる数値なので、値を割り当てることはできません。そのため、この式はエラーとなります。しかし、「a+=4」といった式は、「*a*」が値を割り当てられる変数なので、実行できます。

Add To(*a*, *b*)

a+=*b*

説明

*a*と*b*を足して、その合計を*a*に代入する。

戻り値

合計

引数

a 変数でなければならない。

b 変数、リスト、数値、または行列。

メモ

第1引数の値は変更を受け入れる必要があるので、変数でなければなりません。第1引数を数値にすると、エラーが出ます。

Add To()という関数の形式で指定する場合、引数は2つしか指定できません。引数が1つまたはなしの場合、Add To()は欠測値を戻します。また、最初の2つの引数以外は無視されます。

a+=*b*という演算子の形式で指定する場合、3つ以上の引数を取ることができます。その場合、ペアを右から左へと評価し、それぞれの合計が左側の変数に代入されます。最後の引数を除いて、引数はすべて変数でなければなりません。

例

a+=b+=c

*b*と*c*を足して、その合計を*b*に代入します。さらに、*a*と*b*を足して、その合計を*a*に代入します。

Assign(*a*, *b*)

a=*b*

説明

*b*の値を*a*に代入する。

戻り値

*a*の新しい値

引数

a 変数でなければならない。

b 変数、数値、または行列。

メモ

*a*は値の変更を受け入れる必要があるので、変数でなければなりません。第1引数を数値にすると、エラーが出ます。*b*が何らかの式の場合、まずその式が評価され、その結果が*a*に代入されます。

Divide To(*a*, *b*)

a/=*b*

説明

*a*を*b*で割り、その結果を*a*に代入する。

戻り値

商

引数

a 変数でなければならない。

b 変数、数値、または行列。

Multiply To(*a*, *b*)

*a**=*b*

説明

*a*と*b*を掛けて、その積を*a*に代入する。

戻り値

積

引数

a 変数でなければならない。

b 変数、数値、または行列。

メモ

第1引数の値は変更を受け入れる必要があるので、変数でなければなりません。第1引数を数値にすると、エラーが出ます。

Multiply To() という関数の形式で指定する場合、引数は2つしか指定できません。引数が1つまたはなしの場合、Multiply To() は欠測値を戻します。また、最初の2つの引数以外は無視されます。

a=*b*で指定する場合、3つ以上の引数を取ることができます。その場合、ペアを右から左へと評価し、それぞれの合計が左側の変数に代入されます。最後の引数を除いて、引数はすべて変数でなければなりません。

例

a*=b*=c

*b*と *c*を掛けて、その積を *b*に代入します。さらに、*a*と *b*を掛けて、その積を *a*に代入します。

PostDecrement(*a*)

a--

説明

ポストデクリメント。*a*から1を引いて、差を *a*に代入する。

戻り値

a-1

引数

a 変数でなければならない。

メモ

a--または Post Decrement(*a*) が別の式の中にある場合、まずその式が評価され、次にデクリメント演算子が実行されます。この式は、主にループ制御に使用されます。

Post Increment(*a*)

a++

説明

ポストインクリメント。*a*に1を足して、合計を *a*に代入する。

戻り値

a+1

引数

a 変数でなければならない。

メモ

a++または Post Increment(*a*) が別の式の中にある場合、まずその式が評価され、次にインリメント演算子が実行されます。主にループ制御に使用されます。

Subtract To(*a*, *b*)

a-=*b*

説明

*a*から *b*を引いて、差を *a*に代入する。

戻り値

差

引数

a 変数でなければならない。

b 変数、数値、または行列。

メモ

第1引数の値は変更を受け入れる必要があるので、変数でなければなりません。第1引数を数値にすると、エラーが出ます。

Subtract To() という関数の形式で指定する場合、引数は2つしか指定できません。引数が1つまたはなしの場合、Subtract To() は欠測値を戻します。また、最初の2つの引数以外は無視されます。

a-=*b*で指定する場合、3つ以上の引数を取ることができます。その場合、ペアを右から左へと評価し、それぞれの合計が左側の変数に代入されます。最後の引数を除いて、引数はすべて変数でなければなりません。

例

```
a-=b-=c
```

*b*から *c*を引いて、その差を *b*に代入します。さらに、*a*から *b*を引いて、その差を *a*に代入します。

文字関数

大部分の文字関数においては、引数は文字列であり、戻り値も引用符付き文字列です（一部、数値であるものもあります）。引数に文字列定数を指定する場合、その文字列は引用符で囲む必要があります。

BLOB To Char(*blob*, <Encoding="*enc*">)

説明

指定のエンコーディングを使って、BLOBから引用符付きの文字列を作成する。

戻り値

引用符付き文字列

必須の引数

blob（binary large object）

オプションの引数

encoding エンコーディングを指定する引用符付き文字列。文字列のデフォルトのエンコーディングは "utf-8" です。また、"utf-16le"、"utf-16be"、"us-ascii"、"iso-8859-1"、"ascii~hex"、"shift-jis"、および "euc-jp" もサポートされています。

メモ

エンコーディングのオプションのうち、"ascii~hex" は、CR、LF、TABなどが含まれるBLOBデータを、特別な考慮をせずに、ASCII文字の文字列に単純に変換します。

BLOB To Matrix(*blob*, *type*, *bytes*, *endian*, <*nCols*>)

説明

blobのバイトを数値に変換して行列を作成する。

戻り値

BLOBを数値に変換した行列

必須の引数

blob BLOBまたはBLOBへの参照。

type 数値のデータ型を示す引用符付き文字列。"int"、"uint"、"float"のいずれか。

bytes BLOB内のデータのサイズをバイトで示したもの。1、2、4、8のいずれか。

endian 引用符付きのシステムのエンディアン: "Big"（上位から並べる）、"Little"（下位から並べる）、"Native"（コンピュータのネイティブ形式）。

オプションの引数

<nCols> 行列の列数。デフォルト値は1です。

Char(*x*, <*width*>, <*decimal*>, < <<Use Locale(*Boolean*)>)

説明

式または数値を引用符付き文字列に変換する。

戻り値

引用符付き文字列。

必須の引数

x 式または数値。式はExpr()で囲む必要があります。囲まない場合、その評価結果が引用符付き文字列に変換されます。

オプションの引数

width 引用符付き文字列の最大文字数を設定する数値。

decimal 引用符付き文字列に含まれる、小数点以下の最大桁数を設定する数値。

Use Locale(*Boolean*) ロケール固有の数値形式を維持するかどうかを指定する引数。

例

```
Char( Pi(), 10, 4)
   "3.1416"

Char( Pi(), 3, 4)
   "3.1"
```

メモ

引数*width*が引数*decimal*より優先されます。

Char To BLOB(*string*, <encoding="*enc*">)

説明

引用符付き文字列を、バイナリデータ（BLOB）に変換する。

戻り値

BLOBオブジェクト

必須の引数

string 引用符付き文字列、または文字列変数。

オプションの引数

encoding エンコーディングを指定する引用符付き文字列。blobのデフォルトのエンコーディングは"utf-8"です。また、"utf-16le"、"utf-16be"、"us-ascii"、"iso-8859-1"、"ascii~hex"、"shift-jis"、および"euc-jp"もサポートされています。

メモ

BLOBを印字可能な形式に変換する際、\（および~ " ！と、ASCIIの印字不能な範囲の文字）は、16進数表記に変換されます（バックスラッシュの場合は、~5C）。

```
x = Char To BLOB( "abc\def\!n" );
y = BLOB To Char( x, encoding = "ASCII~HEX" );
If(
   y == "abc~5Cdef~0A", "JMP 12.2以降のバージョンの動作",
   y == "abc\def~0A", "JMP 12.2より前のバージョンの動作"
);
   "JMP 12.2以降のバージョンの動作" // 出力
```

Char To Hex(*value*, <*integer*|encoding="*enc*">)

Hex(*value*, <*integer*|encoding="*enc*"|Base(*number*)|Pad To(*number*)>)

説明

指定された*value*（数値、引用符付き文字列、blob）とencodingに対応する16進数（または他の基数による値）のテキストを戻す。*value*が数値の場合は、*integer*または*Base*が指定されていない限り、IEEE 754の64ビット形式が使用されます。

必須の引数

value 任意の数値、引用符付き文字列、またはBLOB。

オプションの引数

integer valueが数値の場合、浮動小数点ではなく整数としての16進数を戻す。

encoding エンコーディングを指定する引用符付き文字列。デフォルトのエンコーディングは"utf-8"です。"utf-16le"、"utf-16be"、"us-ascii"、"iso-8859-1"、"ascii~hex"、"shift-jis"、"euc-jp"もサポートされています。

Base(*number*) 2～36の整数値。底が指定されている場合は、指定の数値が、16進数ではなくその底を使った数値に対応するテキストで戻されます。

Pad To(*number*) 16進数の出力の先頭にゼロを追加し、指定した桁数で戻す。

Collapse Whitespace(*text*)

説明

先頭および末尾の空白文字を削除し、文字列内で空白文字が連続している部分は重複を削除する。つまり、Collapse Whitespace関数で2つの連続したスペースを1つのスペースに置換できます。

戻り値

引用符付き文字列

必須の引数

text 引用符付き文字列。

Concat(*a*, *b*)

Concat(*A*, *B*)

a||*b*

A||*B*

説明

引用符付き文字列の場合: 文字列 *b* を文字列 *a* に追加する。どちらの引数も変更されません。

リストの場合: リスト *b* をリスト *a* に追加する。どちらの引数も変更されません。

行列の場合: 行列Aと行列Bを横に連結する。

戻り値

文字列の場合: 文字列 *a* の後に文字列 *b* を追加した引用符付き文字列

リストの場合: リスト *a* の後にリスト *b* を追加したリスト

行列の場合: 行列

引数

2つ以上の引用符付き文字列、引用符付き文字列変数、リスト、または行列。

例

```
a = "こんにちは"; b = " "; c = "世界"; a || b || c;
   "こんにちは 世界"
d = {"りんご", "バナナ"}; e = {"桃", "梨"}; Concat( d, e );
   {"りんご", "バナナ", "桃", "梨"}
A = [1 2 3]; B = [4 5 6]; Concat( A, B );
   [1 2 3 4 5 6]
```

メモ

3つ以上の引数を取ることができます。追加の引用符付き文字列は、左から右の順番に文字列の最後に追加されます。行列の場合は、左から右の順番で、横に結合されていきます。

Concat Items

「Concat Items({string1, string2, ...}, <delimiter>})」を参照してください。

Concat To(*a*, *b*)

Concat To(*a*, *b*)

a||=*b*

A||=*B*

説明

引用符付き文字列の場合: 文字列 *a* に文字列 *b* を追加して、新しくできた連結文字列を *a* に代入する。

行列の場合: 行列 *a* に行列 *b* を追加して、新しくできた行列を *a* に代入する。

リストの場合: リスト *a* にリスト *b* を追加して、新しくできたリストを *a* に代入する。

戻り値

引用符付き文字列の場合: 文字列 *a* の後に文字列 *b* を追加した文字列

行列の場合: 行列

リストの場合: リスト *a* の後にリスト *b* を追加したリスト

引数

2つ以上の引用符付き文字列、引用符付き文字列変数、行列、またはリスト。第1引数には変数を指定しなければならない。

メモ

3つ以上の引数を取ることができます。追加の引用符付き文字列、行列、またはリストは、左から右の順番に文字列の最後に追加されます。

例

```
a = "こんにちは"; b = " "; c = "世界"; Concat To( a, b, c ); Show( a );
   a = "こんにちは 世界";
A = [1 2 3]; B = [4 5 6]; Concat To( A, B ); Show( A );
   A = [1 2 3 4 5 6];
d = {"りんご", "バナナ"}; e = {"桃", "梨"}; Concat to(d,e); Show( d );
   d = {"りんご", "バナナ", "桃", "梨"};
```

Contains(*whole*, *part*, <*start*>)

説明

部分（*part*）が全体（*whole*）の中に含まれているかどうかを判断する。

戻り値

part が見つかったとき、リスト、引用符付き文字列、および名前空間の場合は *part* が最初に見つかった位置の数値を戻し、連想配列の場合は1を戻す。

*part*が見つからないときは、すべてのケースで0が戻されます。

必須の引数

whole 引用符付き文字列、リスト、名前空間、または連想配列。

part 引用符付き文字列または名前空間の場合、文字列*whole*の一部の文字列。リストの場合、リスト*whole*にある項目。連想配列の場合、マップ*whole*にあるキー。

オプションの引数

start 文字列*whole*内での検索開始位置を表す数値引数。全体（*whole*）の中。なお、*start*が負の場合、Containsは、*whole*の長さから*start*を差し引いた位置から、*whole*内の*part*を逆方向に検索します。連想配列の場合、*start*は意味がなく、無視されます。

例

```
nameList={"Katie", "Louise", "Jane", "Jaclyn"};
r = Contains(nameList, "Katie");
```

この例では、項目 "Katie" がリストの 1 番目にあるので、1 が戻されます。

Contains Item(*x*, <*item* | {*list*} | *pattern*>, <*delimiter*>)

説明

多重応答（複数回答）の文字列において、指定された項目、リスト、パターン、区切り文字を検索する。この関数は、多重応答の尺度または列プロパティを持つ列に対して使用できます。

戻り値

引数*x*のテキスト内にある単語のいずれかと、単語（*item*）、単語リスト（*list*）の中の1つ、またはパターン（*pattern*）がマッチするかどうかを、ブール値で戻す。テキスト内の単語は、オプションで指定された引用符付き区切り文字（*delimiter*）で区切られます。デフォルトの区切り文字はカンマです。なお、テキスト（*x*）から抽出された各単語の末尾にある空白は削除されます。

例

次の例では、*"pots"*およびそれに続くカンマを検索して、結果を出力します。

```
x = "Franklin Garden Supply is a leading online store featuring garden decor,
   statues, pots, shovels, benches, and much more.";
b = Contains Item( x, "pots", "," );
If( b,
   Write( "指定された項目が見つかりました。"  ), Write( "一致しません。" )
);
```

指定された項目が見つかりました。

Ends With(*string*, *substring*)

説明

文字列（string）の最後に引用符付きの部分文字列（substring）があるかどうかを判断する。

戻り値

引用符付き文字列（string）が引用符付き部分文字列（substring）で終わる場合は1、そうでない場合は0

必須の引数

string 引用符付き文字列、または引用符付き文字列変数。リストでも可。

substring 引用符付き文字列、または引用符付き文字列変数。リストでも可。

等価表現

Right(*string*, Length(*substring*)) == substring

Hex(*value*, <*integer*|encoding="*enc*"|Base(*number*)|Pad To(*number*)>)

「Char To Hex(value, <integer|encoding="enc">)」を参照してください。

Hex To BLOB(*string*)

説明

16進数の文字列（*string*）（空白を含む）から、BLOB（binary large object）を作成する。

例

```
Hex To BLOB( "4A4D50" );
   Char To BLOB("JMP", "ascii~hex")
```

Hex To Char(*string*, <*encoding*>)

説明

16進数の文字列（*string*）を、整数、もしくは、浮動小数点に変換します。

例

Hex To Char("30") の結果は“0”です。

メモ

引用符付き文字列のデフォルトのエンコーディングは"utf-8"です。"utf-16le"、"utf-16be"、"us-ascii"、"iso-8859-1"、"ascii~hex"、"shift-jis"、"euc-jp"もサポートされています。

Hex To Number(*string*, <Base(*number*)>)

説明

16進数（または他の数表現）のテキストに対応する数値を戻す。

必須の引数

string 引用符付きの16進数文字列。

オプションの引数

Base(*number*) 2～36の整数。baseが指定されている場合、テキストは、その基数による値を表す引用符付き文字列とみなされます。

例

```
Hex To Number( "80" );
```
128

メモ

- 入力値の16進数は、IEEE 754の64ビット形式の浮動小数点数として変換されます。それ以外の場合、入力値の16進数は、16進数の整数として、変換されます。
- バイト間（つまり数字のペア）およびバイトの中央に空白文字が含まれていてもかまいません（例："FF 1919"、"F F1919"）。

Insert

「Insert(source, item, <position>)」を参照してください。

Insert Into

「Insert Into(source, item, <position>)」を参照してください。

Item(*n*|[*first last*], *string*, <*delimiter*>, <Unmatched(*result string*)>, <Include Boundary Delimiters(*Boolean*)>)

説明

引用符付き文字列*delimiters*に従って、引用符付き*string*の*n*番目の項目、または*first*から*last*までの項目を戻す。複数の区切り文字を指定した場合、指定したすべての文字が区切り文字とみなされます。

必須の引数

n 抽出する単語の位置。

[*first last*] 抽出する単語の範囲の始めと終わりを指定する行列。

string 評価する引用符付き文字列。

オプションの引数

delimiter 区切りとして使用する文字。*delimiter*がない場合は、ASCII文字のスペースが区切り文字になります。*delimiter*を引用符付きの空白の文字列とした場合、1文字1文字がそれぞれ個別の項目とみなされます。

Unmatched(*result string*) マッチするものがない場合に戻される引用符付き文字列。

Include Boundary Delimiters(*Boolean*) 文字列の前後の区切り文字をどのように扱うかを指定する。デフォルト値は0（偽）で、文字列の前後の区切り文字は無視されます。値が1（真）の場合、空白の要素が生成されます（文字列内に連続する区切り文字がある場合と同様）。

例

区切り文字が連続している場合、区切り文字の間に単語が挟まれているものとみなします。この例では、区切り文字としてカンマとスペースを使用しています。

```
Item( 4,"the  quick, brown fox", ", " ); // quickの前に2つのスペース
```

式は次のように処理されます。

the<delim[space]><word2><delim[space]>quick<delim[comma]><word 4><delim[space]>brown<delim[space]>fox

Word4が空白であるため、式は空白の引用符付き文字列を戻します。

Item()は、区切り文字1つ1つが個別の区切り文字として使われる以外はWord()と同じ。Word()では、連続した複数の区切り文字が1つの区切り文字として扱われます。

```
Word( 4,"the  quick, brown fox", ", " ); // quickの前に2つのスペース
```

式は次のように処理されます。

the<delim[2 spaces]>quick<delim[comma + space]>brown<delim[space]>fox

式は、"fox"を戻します。

Left(*string*, *n*, <*filler*>)

Left({*list*}, *n*, <*filler*>)

説明

元の引用符付き文字列（*string*）から、左から*n*文字の文字列を取り出す。もしくは、元のリスト（*list*）から、最初の*n*個のリスト項目を取り出す。文字列（*string*）の長さが*n*個よりも短い場合は、*filler*に指定された文字列が右側に補充されます。

Length(*string*)

説明

指定した引用符付き文字列の長さ（文字数）、リスト（項目数）、連想配列（キー数）、BLOB（バイト数）、行列（要素数）、名前空間（関数と変数の数）、クラス（メソッド、関数、変数の数）を戻す。

Lowercase(*string*)

説明

引用符付き文字列（*string*）内に現れる文字をすべて小文字に変換する。

Matrix to BLOB(*matrix*, *type*, *bytesEach*, *endian(value)*)

説明

行列の要素を1バイト、2バイト、4バイトの符号付整数または符号なしの整数、4バイトまたは8バイトの浮動小数点数に変換することで、行列からBLOBを作成する。

必須の引数

matrix 行列。

type 引用符付きのBLOBの型: int、uint、float。

bytesEach int、uint、floatでのバイト数。整数（int）はそれぞれ1、2、または4バイト、浮動小数点（float）はそれぞれ4または8バイトになります。

endian(value) 引用符付きのシステムのエンディアン: "Big"（上位から並べる）、"Little"（下位から並べる）、"Native"（コンピュータのネイティブ形式）。

Munger(*string*, *start position*, *find|length*, <*replacement string*>)

説明

引用符付き文字列（*string*）に文字を挿入したり、削除したりして、新しい引用符付き文字列を作る。また、引数の指定方法により、文字列の一部を取り出す、また、ある文字列が何文字目に含まれているかを計算する、といった処理を実行できます。

必須の引数

start position 引用符付き文字列内での検索開始位置を指定する数式。なお、ある検索文字列において、その先頭位置より *start position*が大きい場合、その検索文字列は検索に含まれません。また、*start position*がstringの長さより大きい場合、Munger()は、*start position*を（*string*の長さ+1）に設定します。

find|length 検索する文字列または文字数を指定する。

オプションの引数

replacement string 引用符付きの置換文字列。*replacement string*を指定しなかった場合、*start position*と*position+length*の間にある部分文字列が戻されます。

Num(*string*)

説明

引用符付き文字列を数値に変換する。

Regex(*source*, *pattern*, (<*replacement string*>, <*format*, "GLOBALREPLACE", "IGNORECASE">>)

説明

引用符付きの *source*文字列内で引用符付き *pattern*を検索する。

戻り値

一致するテキストを引用符文字列または数値で戻す。一致するテキストが見つからない場合は欠測値を戻します。

必須の引数

source 引用符付き文字列。

pattern 引用符付き正規表現。

replacement string 置換文字列。

オプションの引数

format 一致したグループへの前方参照。デフォルトは、一致した引用符付き文字列全体である\0。\n は *n*番目にマッチしたものを戻します。

"IGNORECASE" "IGNORECASE"を指定しない場合、大文字と小文字が区別される。

"GLOBALREPLACE" 一致したものがすべて見つかるまで、引用符付きソース文字列（*source*）に正規表現を適用します。

Remove

「Remove(source, position, <n>)」を参照してください。

Remove From

「Remove From(source, position, <n>)」を参照してください。

Repeat(*source*, *a*)

Repeat(*matrix*, *a*, *b*)

説明

*source*を、*a*回、繰り返して連結した結果を戻す。または、*matrix*を*a*回だけ縦に結合し、それを*b*回だけ横に結合した行列を戻します。*source*はテキスト、またはリスト、*matrix*は行列です。

Reverse

「Reverse(source)」を参照してください。

Reverse Into

「Reverse Into(source)」を参照してください。

Right(*string*, *n*, <*filler*>)

Right({*list*}, *n*, <*filler*>)

説明

元の引用符付き文字列（*string*）から、右から*n*文字の文字列を取り出す。もしくは、元のリスト（*list*）から、最後の*n*個のリスト項目を取り出す。文字列（*string*）の長さが*n*個よりも短い場合は、オプションの*filler*に指定された文字列が左側に補充されます。

Shift

「Shift(source, <n>)」を参照してください。

Shift Into

「Shift Into(source, <n>)」を参照してください。

Starts With(*string*, *substring*)

説明

引用符付きの文字列（*string*）の最初に引用符付きの部分文字列（*substring*）があるかどうかを判断する。

戻り値

文字列（*string*）が部分文字列（*substring*）で始まっていれば1、そうでなければ0

引数

string 引用符付き文字列または文字列変数。リストでも可。

substring 引用符付き文字列または文字列変数。リストでも可。

等価表現

```
Left(string, Length("substring")) = = "substring"
```

Substitute

「Substitute(string, "substring", "replacementString", ...)」を参照してください。

Substitute Into

「Substitute Into(string, substring, replacementString, ...)」を参照してください。

Substr(*string*, *start*, *length*)

説明

第2引数（*start*）で指定した位置から第3引数（*length*）で指定した文字数の文字を、第1引数（*string*）の文字列から抽出する。第1引数には、文字列を指定してください。開始位置と文字数の引数は、整数です。

例

この例では、ファーストネームを抽出します。

```
Substr( "Katie Layman", 1, 5 );
```

1文字目から5文字を読み取り、残りの文字を無視して「Katie」を戻します。

Text Score(*text column*, *text-to-number*, *<weighting>*, *<{support vectors}>*, *<text explorer setup>*)

説明

テキストエクスプローラのスコア計算式に使用される。この関数は、[同じ語幹の単語] オプションをサポートしていません。

戻り値

スコアのベクトルを戻す。

必須の引数

text column データテーブルの列。

text-to-number 小文字の単語を数値にマッピングする連想配列。

オプションの引数

weighting 引用符付きの"Count"、"Binary"、"Ternary"、"LogCount"、"LCA"、または TFLogIDF の文書度数の逆数の配列。デフォルト値は、"Count" です。デフォルト値は、"Count" です。

support vectors テキストのスコアリングに使用されるベクトルのリスト。ベクトルの数と長さは、*weighting*引数に依存します。

text explorer setup テキストエクスプローラの設定情報を含む式。

Titlecase(*string*)

説明

引用符付き文字列（*string*）をタイトルケースに変換する。つまり、文字列内の各単語の先頭を大文字にし、残りを小文字にします。

戻り値

引用符付き文字列

引数

string 引用符付き文字列。

例

次の関数は、名前の頭文字を大文字にします。

```
Titlecase( "veronica layman" )
   "Veronica Layman"
```

Trim(*string*,<"Left"|"Right"|"Both">)

Trim Whitespace(*string*,<"Left"|"Right"|"Both">)

説明

指定の文字列から最初および最後のスペースを削除する。

戻り値

引用符付き文字列

必須の引数

string 引用符付き文字列。

オプションの引数

"Left"|"Right"|"Both" 文字列の左側、右側、または両側のいずれからスペースを削除するかを指定する引用符付き文字列。指定しなかった場合、両側から削除されます。

例

たとえば、次のコマンドは"John"を戻します。

```
Trim( "  John    ", Both )
   "John"
```

Uppercase(*string*)

説明

引用符付き文字列（*string*）内に現れる小文字をすべて大文字に変換する。

Word(*n*|[*first last*], *string*, <*delimiter*>, <Unmatched(*result string*)>)

説明

文字列の*n*番目の項目を戻す。*delimiter*引数にある文字が区切り文字となり、任意の数の区切り文字で区切られた部分文字列が単語として扱われます。

必須の引数

n 抽出する単語の位置。

[*first last*] 抽出する単語の範囲の始めと終わりを指定する行列。

string 評価する引用符付き文字列。

オプションの引数

delimiter 区切りとして使用する文字。*delimiter*がない場合は、ASCII文字のスペースが区切り文字になります。*delimiter*を空白の引用符付き文字列とした場合、1文字1文字がそれぞれ個別の項目とみなされます。

Unmatched(*result string*) マッチするものがない場合に戻される引用符付き文字列。

例

この例では、ラストネームを戻します。

```
Word( 2, "Katie Layman" );
   "Layman"
```

次も参照

Word()とItem()の違いを示す例については、「Item(n|[first last], string, <delimiter>, <Unmatched(result string)>, <Include Boundary Delimiters(Boolean)>)」を参照してください。

Words

「Words(string, <delimiter>)」を参照してください。

XPath Query(*xml*, *xpath_expression*)

説明

XMLドキュメントに対して、XPath式を実行する。

戻り値

リスト

必須の引数

xml 有効なXMLドキュメント。

xpath_expression 引用符付きXPath 1.0式。

例

JMPで検定結果のレポートを作成し、重要な詳細をXMLドキュメントに書き出したとしましょう。検定の結果は<result>タグに挟まれています。

次の式は、そのXMLドキュメントを変数内に保存します。XPath Query式は、XMLを解析して<result>に挟まれたテキストノードを見つけます。結果はリストで戻されます。

```
rpt =
"\[<?xml version="1.0" encoding="utf-8"?>
<JMP><report><title>Production Report</title>
<result>November 21st: Pass</result>
<result>November 22nd: Fail</result>
<note>Tests ran at 3:00 a.m.</note></report>
</JMP> ]\";
results = XPath Query( rpt, "//result/text()" );
   {"November 21st: Pass", "November 22nd: Fail"}
```

文字パターン関数

Pat Abort()

説明

パターンマッチを即座に終了する引用符付きパターンを生成する。バックアップも再試行も行いません。条件の割り当ては**行われません**。すでに行われた即座の割り当てが維持されます。

戻り値

マッチが終了したときは0

引数

なし

Pat Altern(*pattern1*, <*pattern2*, ...>)

説明

パターン引数のいずれかにマッチする引用符付きパターンを生成する。

戻り値

引用符付きパターン

引数

1つ以上のパターン。

Pat Any(*string*)

説明

引数のいずれか1文字にマッチする引用符付きパターンを生成する。

戻り値

引用符付きパターン

引数

pattern 引用符付き文字列。

Pat Arb()

説明

任意の引用符付き文字列にマッチする引用符付きパターンを生成する。始めは引用符付きヌル文字列とマッチします。バックアップが行われるたびにマッチする文字列が1文字ずつ長くなります。

戻り値

引用符付きパターン

引数

なし

例

```
p = "最後ではなく" + Pat Arb() >? stuffInTheMiddle + "出てきます";
Pat Match( "ストーリーの最後ではなく始めの方で、3匹のクマが出てきます", p );
Show( stuffInTheMiddle );
   stuffInTheMiddle = "始めの方で、3匹のクマが"
```

Pat Arb No(*pattern*)

説明

引数（*pattern*）に0回以上マッチする引用符付きパターンを生成する。

戻り値

引用符付きパターン

引数

pattern マッチ対象の引用符付きパターン。

例

```
adjectives = "Lサイズ" | "Mサイズ" | "Sサイズ" | "温かい" | "冷たい" | "熱い" | "甘
   め";
rc = Pat Match( "Mサイズの甘めの熱い紅茶を1つください",
                Pat Arbno( adjectives | Pat Any("の") ) >> adj +
```

```
                ("紅茶" | "コーヒー" | "ミルク") );
Show( rc, adj );
   rc = 1;
   adj = "M サイズの甘めの熱い";
```

Pat At(*varName*)

説明

引用符付きヌル文字列にマッチする引用符付きパターンを生成し、ソース文字列の現在の場所を、指定したJSL変数（*varName*）に代入する。割り当ては即座に行われます。値が割り当てられた変数をexpr()で使用することにより、残りのマッチを変更することができます。

戻り値

引用符付きパターン

引数

varName 結果を格納する変数の名前。

例

```
p = ":" + Pat At( listStart ) + Expr(
   If( listStart == 1,
      Pat Immediate( Pat Len( 3 ), early ),
      Pat Immediate( Pat Len( 2 ), late )
   )
);
early = "";
late = "";
Pat Match( ":123456789", p );
Show( early, late );
early = "";
late = "";
Pat Match( "   :123456789", p );
Show( early, late );
```

まず次が生成されます。

```
   early = "123"
   late = ""
```

その後、次が生成されます。

```
   early = ""
   late = "12"
```

Pat Break(*string*)

説明

引数に含まれていない0個以上の文字にマッチする引用符付きパターンを生成し、引数に含まれる文字の前でパターンマッチを停止させる。引数に含まれる文字が見つからない場合は失敗します（特に、停止のための文字が見つからないまま引用符付きソース文字列を最後まで検索し終わった場合）。

戻り値

引用符付きパターン

引数

string 引用符付き文字列。

Pat Concat(*pattern1*, *pattern2* <*pattern3*, ...>)

Pattern1 + *Pattern2* + ...

説明

引数で指定したパターンに順番にマッチする引用符付きパターンを生成する。

戻り値

引用符付きパターン

引数

2つ以上の引用符付きパターン。

Pat Conditional(*pattern*, *varName*)

説明

マッチが終了し成功した場合、引用符付きパターンマッチの結果を第2引数の変数（*varName*）に保存する。

戻り値

引用符付きパターン

引数

pattern マッチ対象の引用符付きパターン。

varName 結果を格納する変数の名前。

例

```
type = "未定義";
rc = Pat Match(
   "青りんご",
   Pat Conditional( "赤" | "青", type ) + "りんご"
);
Show( rc, type );
   rc = 1;
   type = "青";
```

Pat Fail()

説明

マッチングを先に進ませない引用符付きパターンを生成する。マッチングはバックアップし、別のマッチングを試行します。別のマッチングが残っていない場合、マッチングは失敗し、Pat Matchが0を戻します。

戻り値

マッチが失敗したときは0

引数

なし

Pat Fence()

説明

マッチングが先に進む場合は、成功し、引用符付きヌル文字列にマッチするパターンを生成する。バックアップが生じた場合は、失敗します。一部のマッチの最適化に使用できます。

戻り値

マッチが成功したときは1、そうでなければ0

引数

なし

Pat Immediate(*pattern*, *varName*)

説明

パターンマッチの結果を第2引数（*varName*）の名前の変数に即座に保存する。

戻り値

引用符付きパターン

引数

pattern マッチ対象の引用符付きパターン。

varName 結果を格納する変数の名前。

例

```
type = "未定義";
rc = Pat Match(
   "青りんご",
   ("赤" | "青") >> type + "なし"
);
Show( rc, type );
   rc = 0
   type = "青"
```

マッチが失敗した場合でも、割り当ては即座に行われます。

Pat Len(*int*)

説明

*n*文字にマッチする引用符付きパターンを生成する。

戻り値

引用符付きパターン

引数

int 文字数を指定する整数。

Pat Look Ahead(*pattern*, *Boolean*)

説明

現在の位置以後でのゼロ幅のパターンマッチ。

引数

pattern 引用符付きパターン。

Boolean 0（デフォルト）は、マッチを表します。1は、負のマッチまたは非マッチを表します。

Pat Look Behind(*pattern*, *Boolean*)

説明

現在の位置以前でのゼロ幅の引用符付きパターンマッチ。

引数

pattern 引用符付きパターン。

Boolean 0（デフォルト）は、マッチを表します。1は、負のマッチまたは非マッチを表します。

Pat Match(*source text*, *pattern*, <*replacement string*>, <"NULL">, <"ANCHOR">, <"MATCHCASE">, <"FULLSCAN">)

説明

Pat Matchは、*source text*に対して引用符付きパターン（*pattern*）を実行する。直接指定する方法か、または別のJSL変数へパターンを割り当てておく方法により、パターンを生成する必要があります。

戻り値

パターンが見つかれば1、そうでなければ0

必須の引数

source text 検索対象のテキストを示す引用符付き文字列または引用符付き文字列変数。

pattern 検索対象のテキストを示す引用符付きパターンまたはパターン変数。

オプションの引数

replacement string ソーステキスト内でマッチしたパターンを指定の引用符付き文字列に置換する。

"NULL" ANCHOR、MATCHCASE、またはFULLSCANが必要で、かつ置換テキストがない場合の第3引数のプレースホルダー。

"ANCHOR" 引用符付き文字列の冒頭でパターンマッチを開始する。次のパターンマッチは文字列の冒頭にパターン「cream」がないため、失敗となります。

```
Pat Match( "coffee with cream and sugar", "cream", NULL, ANCHOR );
```

"MATCHCASE"（オプション）パターンマッチで大文字／小文字を区別するためのオプション。デフォルトのPat Match()は大文字／小文字を区別しません。

"FULLSCAN"（オプション）Pat Matchにすべての候補を強制的に検索させるオプション。検索数が多いほど、多くのメモリが必要となります。デフォルトでは、Pat Match()はFULLSCANを使用せず、再帰を停止する、あるいはマッチングを継続するための前提を決めて動作します。

Pat Not Any(*string*)

説明

引数内に含まれていない1つの文字にマッチするパターンを生成する。

戻り値

引用符付きパターン

引数

string 引用符付き文字列。

Pat Pos(*int*)

説明

現在の位置が文字列の左端から*int*のとき、引用符付きヌル文字列にマッチするパターンを生成する。その他の場合は失敗します。

戻り値

引用符付きパターン

引数

int 引用符付き文字列内の位置を指定する整数。

Pat R Pos(*int*)

説明

現在の位置が文字列の右端から*int*のとき、引用符付きヌル文字列にマッチするパターンを生成する。その他の場合は失敗します。

戻り値

引用符付きパターン

引数

int 引用符付き文字列内の位置を指定する整数。

Pat R Tab(*int*)

説明

引用符付き文字列の最後から *int* の位置までにマッチする引用符付きパターンを生成する。0文字以上の文字とマッチできます。後ろに移動する場合、または文字列の最後を超える場合は失敗します。

戻り値

引用符付きパターン

引数

int 引用符付き文字列内での位置を指定する整数。

Pat Regex(*string*)

説明

指定された正規表現とマッチする引用符付きパターンを生成する。正規表現を記述する引数 *string* は、引用符付きの文字列で指定する必要があります。

戻り値

引用符付きパターン

引数

string 引用符付き文字列。

Pat Rem()

説明

引用符付き文字列の残りにマッチする引用符付きパターンを生成する。Pat R Tab(0) と等価です。

戻り値

引用符付きパターン

引数

なし

Pat Repeat(*pattern*, *minimum*, *maximum*, <"GREEDY"|"RELUCTANT">)

説明

指定された引用符付パターン（*pattern*）に最小（*minimum*）～最大（*maximum*）回マッチするパターンを生成する。

戻り値

引用符付きパターン

必須の引数

pattern マッチ対象のパターン。

minimum 最大回数。最小回数より小さい整数。

maximum 最小回数。最大回数より大きい整数。

オプションの引数

"GREEDY"|"RELUCTANT" GREEDYが指定されている場合、まず最大回数だけ試行してから最小回数まで戻る。RELUCTANTが指定されている場合、まず最小回数だけ試行してから最大回数まで進む。

メモ

- Pat Arbno(*p*)は、Pat Repeat(*p*, 0, infinity, RELUCTANT)と同じです。
- Pat Repeat(*p*)は、Pat Repeat(*p*, 1, infinity, GREEDY)と同じです。
- Pat Repeat(*p*, *n*)は、Pat Repeat(*p*, *n*, infinity, GREEDY)と同じです。
- Pat Repeat(*p*, *n*, *m*)は、Pat Repeat(*p*, *n*, *m*, GREEDY)と同じです。

Pat Span(*string*)

説明

引数内の文字で構成されている1文字以上（0ではなく）の文字列にマッチするパターンを生成する。これはGREEDYのため、常に、可能な限り長い引用符付き文字列とのマッチを行います。マッチする文字が0文字の場合は失敗します。

戻り値

引用符付きパターン

引数

string 引用符付き文字列。

Pat String(*string*)

説明

引用符付き文字列引数にマッチするパターンを生成する。

戻り値

引用符付きパターン

引数

string 引用符付き文字列。

Pat Succeed()

説明

バックアップが行われた場合でも常に成功するパターンを生成する。引用符付きヌル文字列とマッチします。

戻り値

マッチが成功したときは1

引数

なし

Pat Tab(*int*)

説明

引用符付きソース文字列の *int* の位置までマッチするパターンを生成する。0文字以上の文字とマッチできます。後ろに移動する場合、または文字列の最後を超える場合は失敗します。

戻り値

パターン

引数

int 引用符付き文字列内での位置を指定する整数。

Pat Test(*expr*)

説明

式（*expr*）が0以外のときは、成功し、引用符付きヌル文字列とマッチするパターンを生成する。そうでない場合は失敗します。

戻り値

引用符付きパターン

引数

expr 式。

メモ

通常、引数には expr() が使用されます。通常、テストは、Pat Immediate、Pat Conditional、および Pat Atから戻された現在の値に対して行います。expr() を使用しないと、テストは、パターンマッチを行っている最中ではなく、パターンを生成した時点の値によって行われます。この値により、パターンマッチは常に成功するか常に失敗するかのどちらかとなってしまうため、多くの場合、意図したとおりに動作しません。

例

```
nRaws = 0;
whichRaw = 3;
string = "生米生麦生卵";
rc = Pat Match(
   string,
   "生" + Pat Test(
      Expr(
         nRaws = nRaws + 1;
         nRaws == whichRaw;
      )
   ),
   "ゆで"
```

```
);
Show( rc, string, nRaws );
   rc = 1
   string = "生米生麦ゆで卵"
   nRaws = 3
```

Regex Match(*source*, *pattern*, <*replacement string*>|<"MATCHCASE">, <"NULL">)

説明

引用符付き文字列で指定されたソース（*source*）に対して、引用符付きパターン（*pattern*）のマッチを実行する。

戻り値

パターン

必須の引数

source 引用符付き文字列。

pattern 引用符付きパターン。

オプションの引数

replacement string 元の文字列を置き換える引用符付き文字列。

"MATCHCASE" MATCHCASEを指定しない場合、大文字と小文字は区別されない。

"NULL" MATCHCASEを指定しても、文字列の置換はしないことを示す。

例

```
Regex Match(
   "person=Fred id=77 friend= favorite=tea", // source
   "(\w+)=(\S*) (\w+)=(\S*) (\w+)=(\S*) (\w+)=(\S*)" // pattern
);
   {"person=Fred id=77 friend= favorite=tea", "person", "Fred", "id", "77",
      "friend", "", "favorite", "tea"}

// 大文字と小文字の区別なし、置換なし
Regex Match( "beliEve", "([aeiou])(.*?)(\1)" );
   {"eliE", "e", "li", "E"}
// 大文字と小文字の区別あり、置換なし
Regex Match( "beliEve", "([aeiou])(.*?)(\1)", NULL, MATCHCASE );
   {"eliEve", "e", "liEv", "e"}
```

コメント関数

// comment

説明

行の終わりまでコメント。

メモ

スクリプトの実行時、//から後はすべて無視されます。

/* comment */

説明

行の途中にも挿入できるコメント。

メモ

スクリプトの実行時、開始タグ/*と終了タグ*/の間のものはすべて無視されます。この形式のコメントは、引数のリスト内も含め、ほぼどこにでも挿入できます。ただし、引用符付き文字列の中にコメントを挿入した場合、そのコメントは文字列の一部として扱われ、コメントとはみなされません。また、コメントを演算子の真ん中に挿入することはできません。

例

```
+/*comment*/=
:/*comment*/name
```

これは無効で、エラーが出ます。最初のコメントは+=を妨害し、2番目のコメントは:nameを妨害しています。

```
sums = {(a+b /*comment*/), /*comment*/ (c^2)}
```

これは有効なJSLです。どちらのコメントも無視されます。

//!

説明

これをスクリプトの1行目に記述すると、そのスクリプトをJMPで開いたときに、スクリプトエディタに開かれずに、すぐにスクリプトが実行される。

メモ

このコマンドが指定されていても、次のいずれかの手順を踏めば、ファイルを実行しないで開くことができます。［**ファイル**］**>**［**開く**］を選択します。呼び出されたダイアログにて、JSLファイルを選択し、Ctrlキーを押しながら［**開く**］をクリックします。または、ホームウィンドウの「最近使ったファイル」に表示されているファイルを右クリックし、［**スクリプトの編集**］を選択します。これにより、スクリプトは実行されずに、スクリプトウィンドウに表示されます。

/*debug step*/

/*debug run*/

説明

これをスクリプトの1行目に記述した場合、そのスクリプトは、実行時にデバッガで開かれる。

メモ

文字はすべて小文字でなければなりません。debugとstepの間、またはdebugとrunの間にスペースを1つ挿入し、それ以外にはスペースを挿入しません。どちらか一方の行を、スクリプトの1行目に記述

します。1行目が空白で、2行目にこのコメントを記述した場合、デバッグコマンドは有効になりません。

比較関数

比較演算子（<、<=、>、>=）は、数値、引用符付き文字列、および行列に対して使用できます。行列の場合、各要素を比較した結果の行列が生成されます。文字列と数値や文字列と行列など、タイプが異なるオペランドを比較すると、結果は欠測値になります。リストを比較することはできず、その場合も欠測値が戻されます。

等価演算子（==と!=）は、数値、引用符付き文字列、行列、およびリストに対して使用できます。行列の場合、各要素が等しいかどうかを表す結果の行列が生成されます。リストの場合は、リスト全体として等しいかどうかを表す結果の値が1つ生成されます。文字列と数値や文字列と行列など、タイプが異なるオペランドが**等しいかどうかをテスト**すると、0（等しくない）という結果になります。

範囲をチェックする演算子を使用すると、2つの指定した値の範囲に入っているかどうかをチェックできます。

```
a = 1;
Show( 1 <= a < 3 );
b = 2;
Show( 2 < b <= 3 );
   1 <= a < 3 = 1;
   2 < b <= 3 = 0;
```

比較演算子が含まれた式は、順にではなく、すべて一度に評価される

比較演算子はすべて**省略演算子**（複数の演算を指定する際に省略ができる演算子）です。大部分の演算においては、一度に1つずつ演算子が評価されていきます。しかし、比較演算子で結合されているオペランドは1つの大きなまとまりとして取り扱われます。1つのまとまりとして評価すると、部分ごとに評価する通常の方法とは異なる結果が生成されます。たとえば、次の2つのステートメントはそれぞれ別のものです。

```
12 < a < 13;
(12 < a) < 13;
```

最初のステートメントは、3つの引数と両方の演算子すべてが読み取られて一緒に評価されるため、*a*が12と13の範囲にあるかどうかをチェックします。2つ目のステートメントでは、括弧を使って明示的に処理をグループに分けて、左から順に評価します。そのため、この式では、まず12が*a*より小さいかどうかをチェックし、真のとき1を、偽のとき0を戻します。次に、その結果（0または1）が13より小さいかどうかをチェックします。0も1も13より小さいので、この結果は必ず真になります。

同じ演算子の組み合わせまたは異なる演算子の組み合わせ（<... <=と <=... <）で使用すると、すべての比較演算子は一度に評価されます。つまり、一度に1つずつ比較演算子を評価する場合には、括弧( )を使って、明示的に操作の順序を制御する必要があることを意味します。

Equal(*a*, *b*, ...)

a==*b*==...

説明

指定されたすべての値を比較し、同じ値かどうかを判定する。

戻り値

すべての引数の結果が同じ値になるときは1（true）

そうでなければ0（false）

引数

2つ以上の変数、参照、行列、または数値。

メモ

– 3つ以上の引数が指定された場合は、すべての引数が同じ値である場合にだけ1が戻されます。この演算子は、主に、条件文や、ループの制御で使用されます。

– 引用符付き文字列の比較では、大文字と小文字が区別されます。

Greater(*a*, *b*, ...)

a>*b*>...

説明

指定された値を比較して、各ペアについて左の値が右の値よりも大きいかどうかを判定する。

戻り値

*a*が*b*より大きい（および*b*が*c*より大きい）ときは1（true）

そうでなければ0（false）

引数

2つ以上の変数、参照、行列、または数値。

メモ

3つ以上の引数が指定された場合は、すべての引数の値がそれぞれの右側の引数の値よりも大きい場合にだけ1が戻されます。この演算子は、主に、条件文や、ループの制御で使用されます。

（0 < x <= 5のように）Greater、Less、Greater Or Equal、およびLess Or Equalを同時に指定することもできます。括弧を使ってグループ化した場合を除き、各ペアは左から右へと評価されます。括弧を使って明確に式の評価順を示すこともできます。

Greater or Equal(*a*, *b*, ...)

a>=*b*>=...

説明

指定された値を比較して、各ペアについて左の値が右の値以上かどうかを判定する。

戻り値

*a*が*b*以上（および*b*が*c*以上）のときは1（true）

そうでなければ0（false）

引数

2つ以上の変数、参照、行列、または数値。

メモ

3つ以上の引数が指定された場合は、すべての引数の値がそれぞれの右側の引数の値以上である場合にだけ1が戻されます。この演算子は、主に、条件文や、ループの制御で使用されます。

（`0 < x <= 5`のように）`Greater`、`Less`、`Greater Or Equal`、および`Less Or Equal`を同時に指定することもできます。括弧を使ってグループ化した場合を除き、各ペアは左から右へと評価されます。括弧を使って明確に式の評価順を示すこともできます。

`Is Missing(`*expr*`)`

説明

評価後の引数が欠測値のときは1、そうでなければ0を戻す。

`Less(`*a*`, `*b*`, ...)`

a`<`*b*`<...`

説明

指定された値を比較して、各ペアについて左の値が右の値よりも小さいかどうかを判定する。

戻り値

*a*が*b*より小さい（および*b*が*c*より小さい）ときは1（true）

そうでなければ0（false）

引数

2つ以上の変数、参照、行列、または数値。

メモ

3つ以上の引数が指定された場合は、第1引数が第2引数以下で、そのほかのすべての引数の値がそれぞれの右側の引数未満である場合にだけ1が戻されます。この演算子は、主に、条件文や、ループの制御で使用されます。

（`0 < x <= 5`のように）`Greater`、`Less`、`Greater Or Equal`、および`Less Or Equal`を同時に指定することもできます。括弧を使ってグループ化した場合を除き、各ペアは左から右へと評価されます。括弧を使って明確に式の評価順を示すこともできます。

Less Less Equal(*a*, *b*, *c*, ...)

a<*b*<=*c*<=...

説明

範囲の照合。*b*が*a*より大きく*c*以下であるかを判定する。

戻り値

*b*が*a*より大きく*c*以下のときは1（true）

そうでなければ0（false）

引数

a，*b*，*c* 変数、参照、行列、または数値。

メモ

第1引数が第2引数より小さく、残りの各引数が右側の引数以下であるという、2つの条件が両方とも満たされた場合に1を返します。この演算子は、主に、条件文や、ループの制御で使用されます。

Less or Equal(*a*, *b*, ...)

a<=*b*<=...

説明

指定された値を比較して、各ペアについて左の値が右の値以下であるかどうかを判定する。

戻り値

*a*が*b*以下（および*b*が*c*以下）のときは1（true）

そうでなければ0（false）

引数

2つ以上の変数、参照、行列、または数値。

メモ

3つ以上の引数が指定された場合は、すべての引数の値がそれぞれの右側の引数の値以下である場合にだけ1が戻されます。この演算子は、主に、条件文や、ループの制御で使用されます。

（`0 < x <= 5`のように）`Greater`、`Less`、`Greater Or Equal`、および`Less Or Equal`を同時に指定することもできます。括弧を使ってグループ化した場合を除き、各ペアは左から右へと評価されます。括弧を使って明確に式の評価順を示すこともできます。

Less Equal Less(*a*, *b*, *c*, ...)

a<=*b*<*c*<...

説明

範囲の照合。*b*が*a*以上で*c*より小さいかを判定する。

戻り値

*b*が*a*以上で*c*より小さいときは1（true）

そうでなければ0（false）

引数

a，*b*，*c* 変数、参照、行列、または数値。

メモ

第1引数が第2引数以下で、残りの各引数が右側の引数より小さいという、2つの条件が両方とも満たされた場合に1を返します。この演算子は、主に、条件文や、ループの制御で使用されます。

`Not Equal(a, b)`

a!=*b*

説明

*a*と*b*を比較して、等しくないかどうか判定する。

戻り値

*a*と*b*が同値と評価されたときは0（false）

そうでなければ1（true）

引数

a，*b* 任意の変数または数値。

メモ

主に、条件文や、ループの制御で使用されます。

条件付き関数と論理関数

`And(a, b)`

a&*b*

説明

論理積。

戻り値

*a*と*b*の両方が真のときは1（true）

*a*または*b*のどちらか、または*a*と*b*の両方が偽のときは0（false）

*a*または*b*のどちらか、または*a*と*b*の両方が欠測値のときは欠測値

引数

2つ以上の変数または式。

メモ

3つ以上の引数を取ることができます。*a&b*は、すべての引数が真と評価されたときのみ、1（true）を戻します。

AndMZ(*a*, *b*)

a:&*b*

説明

すべての引数の論理積を戻す。欠測値はゼロとして扱われます。

戻り値

*a*と*b*の両方が真のときは1（true）

*a*または*b*のどちらか、または*a*と*b*の両方が偽のときは0（false）

*a*または*b*のどちらか、または*a*と*b*の両方が欠測値のときは0（false）

引数

2つ以上の変数または式。

メモ

3つ以上の引数を取ることができます。*a*:&*b*は、すべての引数が真と評価されたときのみ、1（true）を戻します。

Break()

説明

ループを完全に停止して、ループの後に続くステートメントの実行に移る。

メモ

Break()は、Forループ、Whileループ、およびFor Each Rowで使用できます。

Choose(*expr*, *r1*, *r2*, *r3*, ..., *resultElse*)

説明

式（*expr*）を評価し、*expr*の値が1のときは*r1*の値を戻し、2のときは*r2*の値を戻す。どれにも該当しない場合は最後の引数（*resultElse*）を戻します。

戻り値

引数リスト内でインデックスが*expr*とマッチする値、または最後の引数の値

引数

expr 式または値。

r1, *r2*, *r3*, ... 式または値。

resultElse どれにも該当しないときに戻される引数。

Continue()

説明

ループの現在の反復を終了して、次の反復を開始する。

メモ

Continue()は、Forループ、Whileループ、およびFor Each Rowで使用できます。

Filter Each(*names*, *container*, *locals*, *body*)

説明

リスト、連想配列、行列のいずれかであるコンテナ（*container*）で反復し、各反復におけるブール式の評価に基づいてコンテナ内の値のサブセットを戻す。値、キー、または要素、そして通し番号を、各反復において指定できます。

- コンテナが連想配列である場合は、キーと値をペアにしたリストを指定することで、それらキーと値の組み合わせを評価することもできます。
- コンテナが行列である場合は、デフォルトでは全要素に対する通し番号が与えられますが、行番号と列番号をペアにしたリストを2つ目の項目に指定することで、それら行番号と列番号を評価することができます。

それらのシンボルは、ループの本体（*body*）だけで使われます。ローカル変数のリストを指定することもできます。ローカル変数を定義した場合、それらのローカル変数は、最初の反復においてシンボルが設定された後に初期化されます。

戻り値

元のコンテナからのサブセット。元のコンテナと同じ種類のオブジェクトが戻されます。

引数

names ループ制御の変数名を、リストとして指定する。リストの形式は、コンテナの種類によって異なります。この最初の*names*引数はオプションであり、*locals*引数または*body*引数で参照する必要がある場合のみ指定します。要素や通し番号などを参照する必要がない場合、最初の*names*引数には、{}またはList()を使って空白のリストを指定してください。

コンテナが**リスト**である場合は、*names*引数として指定するリストには、コンテナとして指定したリスト内の各値を参照する変数と、リスト内の通し番号を参照する変数を含めます。

コンテナが**連想配列**である場合は、*names*引数として指定するリストには、連想配列のキーと値に対応したリストと、通し番号を参照する変数を含めます。

コンテナが**行列**である場合は、*names*引数として指定するリストには、行列の各要素の値を参照する変数と、行列内での通し番号を参照する第2引数を含めます。この第2引数には、全体での通し番号を参照したい場合には変数を指定してください。一方、行番号および列番号を参照したい場合には、それらをペアにしたリストを指定してください。

Across()キーワードを使って複数のコンテナを指定する場合は、*names*引数に指定するリストには、各コンテナの値を参照する変数を含めたリストを含めてください。その際、そのリストに含める項目の個数は、Across()キーワードで指定したコンテナの個数と一致しなければなりません。

container リスト、連想配列、または行列。コンテナ（*container*）は、引数で定義するか、前に定義したオブジェクトへの参照として指定できます。

この引数は、Across() キーワードを使うことにより、複数のコンテナを扱うこともできます。複数のコンテナは、個別の引数として、またはリスト内の項目として指定できます。Across() キーワードには、オプションの Count() 引数があります。この引数では、サイズが異なるコンテナの扱い方を指定できます。使用できる Count() のオプションは、"Shortest"、"Longest"、"Enforce Equal"、N です。ここで、N は数値です。数値を指定した場合、すべてのコンテナにおいて、N 回の反復が行われます。なお、その際、項目数が N より少ないコンテナは、項目数を超えると最初の項目に戻ります。

locals 関数に対してローカルである変数のリスト。これは、JSL での他のローカル変数の初期化と同じです。ローカル変数の初期化は、最初のループ制御変数が設定された後に行われますが、その後、二度と行われることはありません。

body 任意の数の有効な JSL 式。セミコロンで区切ることにより、複数の JSL 式を指定できます。その JSL 式の結果は、ブール値になるようにしてください。その JSL 式の結果が真の場合、現在の反復におけるコンテナの値が結果に含められます。その JSL 式の結果が真でない場合は、現在の反復におけるコンテナの値は結果に含まれません。Continue() 関数を使うと、「偽」が戻されて次の反復に進んだ場合と同じになります。また、Break() 関数を使ってループの反復を停止し、ループの後の式に進むこともできます。「Break 関数と Continue 関数」を参照してください。

例

```
values = Filter Each( {x}, {10, 20, 30}, x > 15 );
Show( values );
   values = {20, 30};
```

For(*init*, *while*, *increment*, *body*)

説明

*body*に指定されたスクリプトを、条件（*while*）が真である間、繰り返し実行する。*init*と*increment*によって反復を制御します。

戻り値

なし

引数

init ループ制御カウンタの初期化。

while ループの継続／終了の条件。条件文の*while*が真である限り、ループは再度反復される。*while*が偽になると、即座にループは終了する。

increment ループが実行されるたびに、*while*が評価された後、ループカウンタをインクリメント（またはデクリメント）する。

body 任意の数の有効な JSL 式。セミコロンで区切ることにより、複数の JSL 式を指定できます。

例

```
mysum = 0; myprod = 1;
For( i = 1, i <= 10, i++, mysum += i; myprod *= i; );
Show( mysum, myprod );
```

```
mysum = 55;
myprod = 3628800;
```

For Each(*names*, *container*, *locals*, *body*)

説明

リスト、連想配列、行列のいずれかであるコンテナ（*container*）で反復する。コンテナには、値、キー、または要素が含まれる。各反復で通し番号を使うこともできる。コンテナが連想配列である場合は、キーと値をペアにしたリストを指定することで、それらキーと値の組み合わせを評価することもできます。コンテナが行列である場合は、デフォルトでは全要素に対する通し番号が与えられますが、行番号と列番号をペアにしたリストを2つ目の項目に指定することで、それら行番号と列番号を評価することができます。それらのシンボルは、ループの本体（*body*）だけで使われます。ローカル変数のリストを指定することもできます。ローカル変数を定義した場合、それらのローカル変数は、最初の反復においてシンボルが設定された後に初期化されます。

引数

names ループ制御の変数名を、リストとして指定する。リストの形式は、コンテナの種類によって異なります。この最初の *names*引数はオプションであり、*locals*引数または *body*引数で参照する必要がある場合のみ指定します。要素や通し番号などを参照する必要がない場合、最初の *names*引数には、{}またはList()を使って空白のリストを指定してください。

コンテナが**リスト**である場合は、*names*引数として指定するリストには、コンテナとして指定したリスト内の各値を参照する変数と、リスト内の通し番号を参照する変数を含めます。

コンテナが**連想配列**である場合は、*names*引数として指定するリストには、連想配列のキーと値に対応したリストと、通し番号を参照する変数を含めます。

コンテナが**行列**である場合は、*names*引数として指定するリストには、行列の各要素の値を参照する変数と、行列内での通し番号を参照する第2引数を含めます。この第2引数には、全体での通し番号を参照したい場合には変数を指定してください。一方、行番号および列番号を参照したい場合には、それらをペアにしたリストを指定してください。

Across()キーワードを使って複数のコンテナを指定する場合は、*names*引数に指定するリストには、各コンテナの値を参照する変数を含めたリストを含めてください。その際、そのリストに含める項目の個数は、Across()キーワードで指定したコンテナの個数と一致しなければなりません。

container リスト、連想配列、または行列。コンテナ（*container*）は、引数で定義するか、前に定義したオブジェクトへの参照として指定できます。

この引数は、Across()キーワードを使うことにより、複数のコンテナを扱うこともできます。複数のコンテナは、個別の引数として、またはリスト内の項目として指定できます。Across()キーワードには、オプションのCount()引数があります。この引数では、サイズが異なるコンテナの扱い方を指定できます。使用できるCount()のオプションは、"Shortest"、"Longest"、"Enforce Equal"、Nです。ここで、Nは数値です。数値を指定した場合、すべてのコンテナにおいて、N回の反復が行われます。なお、その際、項目数がNより少ないコンテナは、項目数を超えると最初の項目に戻ります。

locals 関数に対してローカルである変数のリスト。これは、JSLでの他のローカル変数の初期化と同じです。ローカル変数の初期化は、最初のループ制御変数が設定された後に行われますが、その後、二度と行われることはありません。

body 任意の数の有効なJSL式。セミコロンで区切ることにより、複数のJSL式を指定できます。Continue()関数を使って次の反復に進むことができます。また、Break()関数を使ってループの反復を停止し、ループの後の式に進むこともできます。「Break関数とContinue関数」を参照してください。

例

```
For Each( {value, index}, {10, 20, 30}, Show( value, index ) );
   value = 10;
   index = 1;
   value = 20;
   index = 2;
   value = 30;
   index = 3;
```

For Each Row(<*dt*>, *script*)

説明

データテーブルの各行に対してスクリプト（*script*）を繰り返す。

戻り値

なし

必須の引数

script 任意の有効なJSL式。

オプションの引数

dt データテーブルへの参照である位置引数。この引数がデータテーブルを参照する変数でない場合は、データテーブルの式とみなされます。

例

次の例は、データテーブルへの参照を設定し、「Big Class.jmp」のすべての行に処理を適用します。ある行の「年齢」の値が15より大きい場合は、その年齢がログに出力されます。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
For Each Row( dt, If( :年齢 > 15, Show( :年齢 ) ) );
```

If(*condition1*, *result1*, <*condition2*, *result2*>, ..., <*elseResult*>)

説明

引数の各ペアにおいて、*condition*引数がゼロ以外の結果になるとき、結果（*result*）を戻す。*condition*引数は順番に評価されます。すべての*condition*引数がゼロになる場合は、オプションの*elseResult*を評価し、その結果を戻します。*elseResult*の指定がなく、*condition*のいずれも真でない場合は、欠測値を戻します。*condition*引数のすべてが欠測値になる場合は、欠測値を戻します。

IfMax(*expr1*, *result1*, *expr2*, *result2*, ... <*all missing results*>)

説明

引数の各ペアにおいて、1つ目の引数が最大値となっているペアの結果（各ペアの2つ目）を戻します。最大値となる式が複数ある場合には、最初の最大値に対応する結果を戻します。すべての式が欠測値で、最後の結果式が指定されていない場合、欠測値を戻します。すべての式が欠測値であり、最後の結果式が指定されている場合は、最後の結果式の評価を戻します。条件式は数値になる必要がありますが、結果式はどのような値でもかまいません。

戻り値

最大値に対する結果式の評価

IfMin(*expr1*, *result1*, *expr2*, *result2*, ... <*all missing results*>)

説明

引数の各ペアにおいて、1つ目の引数が最小値となっているペアの結果（各ペアの2つ目）を戻します。最小値となる式が複数ある場合には、最初の最小値に対応する結果を戻します。すべての式が欠測値で、最後の結果式が指定されていない場合、欠測値を戻します。すべての式が欠測値であり、最後の結果式が指定されている場合は、最後の結果式の評価を戻します。条件式は数値になる必要がありますが、結果式はどのような値でもかまいません。

戻り値

最小値に対する結果式の評価

IfMZ(*condition1*, *result1*, <*condition2*, *result2*>, ..., <*elseResult*>)

説明

引数の各ペアにおいて、*condition1*引数がゼロ以外の結果になるとき、結果（*result*）を戻す。*condition*引数は順番に評価されます。すべての *condition*引数がゼロまたは欠測値になる場合は、オプションの *elseResult*を評価し、その結果を戻します。*elseResult*の指定がなく、*condition*のいずれも真でない場合は、欠測値を戻します。

メモ

条件式（*condition*）の引数は、最初にゼロ以外の結果が出るまで順番に評価されます。すべての条件式がゼロまたは欠測値になる場合、*elseExpr*引数が評価されます。

IfMZ() は If() と等価ですが、*condition*引数の評価結果が欠測値になった場合はその結果をゼロとして扱います。

Interpolate(*x*|*xmatrix*|*xlist*, *x1*, *y1*, *x2*, *y2*, ...)

Interpolate(*x*|*xmatrix*|*xlist*, *xmatrix*, *ymatrix*)

Interpolate({*x*, *y*}, *xvector*, *yvector*, *zmatrix*)

説明

連続量であるデータの線形補間を実行する。この関数は、いろいろな方法で指定できます。

最も単純な例では、*xmatrix*と*ymatrix*で与えられた関数（もしくは、x, yの2変数のペアで与えられた関数）について、第1引数の*x*値に対応した*y*値を戻します。*x*値（*x1*、*x2*、...）または*xmatrix*は、昇順に並んでいる必要があります。

最初の引数に数値の行列またはリストを指定した場合、行列またはリストで補間した値が戻されます。

引数として4つを指定すると**双線形補間（bilinear interpolation）**を実行することができます。ここで、第1引数は2つの点を含むリストで、第2引数と第3引数はxとyの値のグリッドを定義するベクトル、第4引数はデータ点の行列です。関数は、*zmatrix*の該当する長方形領域内で補間された*z*値を求めます。

戻り値

補間された値。3つの引数を指定した場合、戻り値の種類は第1引数の種類と同じものです。4つの引数を指定した場合、戻り値は数値です。

Is Associative Array(*name*)

説明

評価後の引数が連想配列のときは1、そうでなければ0を戻す。

Is Empty(*global*)

Is Empty(*dt*)

Is Empty(*col*)

説明

*global*変数、データテーブル、またはデータ列が定義されていないか、Empty()値をもつときは1、そうでなければ0を戻す。

Is Expr(*x*)

説明

評価後の引数が式のときは1、そうでなければ0を戻す。

Is List

「Is List(x)」を参照してください。

Is Name(*x*)

説明

評価後の引数が名前のときは1、そうでなければ0を戻す。

Is Namespace(*namespace*)

説明

引数*namespace*が名前空間の場合は1、そうでなければ0を戻す。

Is Number(*x*)

説明

評価後の引数が数値か欠測値のときは1、そうでなければ0を戻す。

Is Scriptable(*x*)

説明

評価後の引数がスクリプト可能なオブジェクトのときは1、そうでなければ0を戻す。

Is String(*x*)

説明

引数が引用符付き文字列のときは1、そうでなければ0を戻す。

Match(*x*, *value1*, *result1*, *value2*, *result2*, ..., *resultElse*)

説明

*a*が値1（*value1*）に等しいときは結果1（*result1*）を戻し、*a*が値2（*value2*）に等しいときは結果2（*result2*）、・・・を戻す。

メモ

Match()関数は比較の値*x*が欠測値であるか、*value1*の値が欠測値であるかを明示的にチェックします。もし両方とも欠測値であればresult1の値を戻します。そうでなければ欠測値を無視して、式*x*と*valueN/resultN*の*valueN*との比較を続けます。もし式*x*が*valueN*値のいずれかと等しい場合は、対応する*resultN*値を戻します。もし等しい*valueN*値が見つからなければ、*resultElse*値を戻します。

MatchMZ(*x*, *value1*, *expr1*, *value2*, *expr2*, ..., *exprElse*)

説明

*x*引数を評価して*exprN*を戻すか、*x*と一致する*valueN*がない場合は*exprElse*引数を評価して戻す。

メモ

MatchMZ()関数は、Match()関数と同じように動作しますが、欠測値を0として扱います。

Not(*a*)

!*a*

説明

論理否定。

戻り値

*a*が0以外のときは0（false）

a=0 のときは1（true）

*a*が欠測値のときは欠測値

引数

a 任意の変数または数値。変数の場合、数値または行列でなければなりません。

メモ

主に、条件文や、ループの制御で使用されます。

Or(*a*, *b*)

a|*b*

説明

論理和。

戻り値

*a*と*b*のどちらかまたは両方が真のときは1（true）

そうでなければ0（false）

一方が欠測値で、もう一方が欠測値もしくは偽の場合には、欠測値を戻す。

引数

a, *b* 任意の変数または数値。

メモ

主に、条件文や、ループの制御で使用されます。

OrMZ(*a*, *b*)

a:|*b*

説明

欠測値を0とみなし、すべての引数の論理和（OR）を戻す。いずれかの引数が0以外であれば1、それ以外の場合には0を戻す。

戻り値

*a*と*b*のどちらかまたは両方が真のときは1（true）

そうでなければ0（false）

引数

a, *b* 任意の変数または数値。

メモ

- 主に、条件文や、ループの制御で使用されます。
- Or() は、いずれの引数の評価が欠測値である場合には、欠測値を戻します。OrMZ() は、いずれの引数の評価も欠測値である場合、0 を戻します。

Return(<*expr1*>, <*expr2*>, ..., <*exprN*>)

説明

ユーザ定義の関数から式の値を戻します。

例

次の例では、Return() 関数の2つの式の結果がリストで戻されます。Return() 関数には、2つ以上の引数を指定することもできます。引数が1つしかない場合は、その式の値が戻されます。引数が2つ以上ある場合は、すべての式の値がリストとして戻されます。

```
f = Function( {a, b},
   Return( a - b, a + b )
);
{lo, hi} = f( 10, 1 );
Show( lo, hi );
Show( f( 7, 15 ) );
   lo = 9;
   hi = 11;
   f(7, 15) = {-8, 22};
```

メモ

Return() を関数やメソッド、再帰的関数呼び出しの中でないところで使用すると、エラーとなります。

Step(*x0*, *x1*, *y1*, *x2*, *y2*, ...)

Step(*x0*, [*x1*, *x2*, ...], [*y1*, *y2*, ...])

説明

*x0*以下となっている引数 *x* のうち、最大の値に対応する引数 *y* を戻す。引数 *x* は小さい順に指定されていなければなりません。

Stop()

説明

実行中のスクリプトを直ちに停止します。

Transform Each(*names*, *container*, *<Output(type)>*, *<locals>*, *body*)

説明

リスト、連想配列、行列のいずれかであるコンテナ（*container*）で反復し、各反復でコンテナ内の値を更新する。値、キー、または要素、そして通し番号を、各反復において指定できます。コンテナが連想配列である場合は、キーと値をペアにしたリストを指定することで、それらキーと値の組み合わせを評価することもできます。コンテナが行列である場合は、デフォルトでは全要素に対する通し番号が与えられますが、行番号と列番号をペアにしたリストを2つ目の項目に指定することで、それら行番号と列番号を評価することができます。それらのシンボルは、ループの本体（*body*）だけで使われます。

ローカル変数のリストを指定することもできます。ローカル変数を定義した場合、それらのローカル変数は、最初の反復においてシンボルが設定された後に初期化されます。

戻り値

元のコンテナを更新したもの。Output() キーワードを使って別の種類のコンテナを戻すよう指定した場合を除き、元のコンテナと同じ種類のオブジェクトが戻されます。

引数

names ループ制御の変数名を、リストとして指定する。リストの形式は、コンテナの種類によって異なります。この最初の *names*引数はオプションであり、*locals*引数または *body*引数で参照する必要がある場合のみ指定します。要素や通し番号などを参照する必要がない場合、最初の *names*引数には、{} または List() を使って空白のリストを指定してください。

コンテナが**リスト**である場合は、*names*引数として指定するリストには、コンテナとして指定したリスト内の各値を参照する変数と、リスト内の通し番号を参照する変数を含めます。

コンテナが**連想配列**である場合は、*names*引数として指定するリストには、連想配列のキーと値に対応したリストと、通し番号を参照する変数を含めます。

コンテナが**行列**である場合は、*names*引数として指定するリストには、行列の各要素の値を参照する変数と、行列内での通し番号を参照する第2引数を含めます。この第2引数には、全体での通し番号を参照したい場合には変数を指定してください。一方、行番号および列番号を参照したい場合には、それらをペアにしたリストを指定してください。

Across() キーワードを使って複数のコンテナを指定する場合は、*names*引数に指定するリストには、各コンテナの値を参照する変数を含めたリストを含めてください。その際、そのリストに含める項目の個数は、Across() キーワードで指定したコンテナの個数と一致しなければなりません。

container リスト、連想配列、または行列。コンテナ（*container*）は、引数で定義するか、前に定義したオブジェクトへの参照として指定できます。

この引数は、Across() キーワードを使うことにより、複数のコンテナを扱うこともできます。複数のコンテナは、個別の引数として、またはリスト内の項目として指定できます。Across() キーワードには、オプションの Count() 引数があります。この引数では、サイズが異なるコンテナの扱い方を指定できます。使用できる Count() のオプションは、"Shortest"、"Longest"、"Enforce Equal"、Nです。ここで、Nは数値です。数値を指定した場合、すべてのコンテナにおいて、N回の反復が行われます。なお、その際、項目数がNより少ないコンテナは、項目数を超えると最初の項目に戻ります。

Output(type) 出力のタイプを指定する。選択肢には、"List"、"Matrix"、"Associative Array" があります。デフォルトでは、出力の種類は、入力に指定したコンテナの種類と同じものです。

locals 関数に対してローカルである変数のリスト。これは、JSLでの他のローカル変数の初期化と同じです。ローカル変数の初期化は、最初のループ制御変数が設定された後に行われますが、その後、二度と行われることはありません。

body 任意の数の有効なJSL式。セミコロンで区切ることにより、複数のJSL式を指定できます。各反復において、JSL式の結果が、出力のコンテナでの各要素の値に設定されます。Continue()関数を使用すると、反復においては値が戻されず、次の反復に進みます。また、Break()関数を使ってループの反復を停止し、ループの後の式に進むこともできます。「Break関数とContinue関数」を参照してください。

例

```
values = Transform Each( {x}, {10, 20}, x + 10 );
Show( values );
   values = {20, 30};
```

While(*expr*, *body*)

説明

条件式（*expr*）を評価し、真であれば本文の式（*body*）を実行する。これを条件式（*expr*）が真でなくなるまで繰り返します。

Zero Or Missing(*expr*)

説明

式（*expr*）が欠測値またはゼロを生成する場合に1を戻し、そうでない場合は0を戻す。

定数関数

JMPには、便利な定数の関数が2つあります。

メモ: これらの関数は、引数は不要ですが、括弧は必要です。

e()

説明

定数*e*の値（2.7182818284590451...）を戻す。

Pi()

説明

定数π（3.1415926535897931...）を戻す。

日付と時間関数

JMPでは、日付時間値を、1904年1月1日午前0時からの秒数で内部的に保持します。

「x=01Jan1904」という式を実行すると、指定されている日付がJMPでの基準日つまり「ゼロ日」のため、*x*はゼロに設定されます。通常、日付値の内部的な数値は、かなり大きな数値になります。(たとえば5oct1998は2990390400になります。)

Abbrev Date(*date*)

説明

指定された日付(*date*)を引用符付き文字列に変換する。

戻り値

日付の引用符付き文字列

引数

date 基準日(1904/01/01の午前零時)からの秒数、または任意の日付時間演算子。

例

```
Abbrev Date( 29Feb2004 );
```

2004/02/29

As Date(*x*)

説明

数値または式*x*の日付型の形式を戻す。戻り値は、テキストウィンドウに送られたときに、日付または期間として表示されるようになります。1年以上を示す値は日付として戻されます。1年未満を表す値は、時間の長さとして戻されます。

戻り値

指定された数値または式から計算された日付

引数

x 数値または式。

Date Difference(*datetime1*, *datetime2*, *interval name*, <*alignment*>)

説明

2つの日付時間値の差を、指定された単位で戻す。

戻り値

数値

必須の引数

datetime1, *datetime2* 日付時間値。

interval name 期間を示す引用符付き文字列。"month"、"day"、"hour"など。

オプションの引数

alignment 引用符付き文字列。

- "Start"は、2つの日付時間値の間隔内で、指定した期間が何度始まったかを数えます。
- "Actual"は、2つの日付時間値の間隔内に期間全体がいくつ含まれていたかを数えます。
- "Fractional"は、2つの日付時間値の間隔内の期間の数を、"year"、"quarter"、"month"の長さの平均値を使って小数部まで計算します。

Date DMY(*day*, *month*, *year*)

説明

引数から日付値を生成する。

戻り値

1904年1月1日午前0時から指定の日付までの秒数

引数

day 日付を示す1～31の数字。エラーのチェックは行われません。たとえば、2月31日も入力されてしまうので注意が必要。

month 月を示す1～12の数字。

year 年。

Date Increment(*datetime*, *interval name*, <*increment*>, <*alignment*>)

説明

期首の日付時間値に1または複数の期数を加算します。

戻り値

新しい日付時間値

必須の引数

datetime 開始の日付時間値。

interval name 期間を示す引用符付き文字列。"year"、"quarter"、"month"、"week"、"day"、"hour"、"minute"、"second"が使用できます。

オプションの引数

increment 間隔数を指定する数値。デフォルト値は1です。

alignment キーワードを含んだ引用符付き文字列。

- "Start"は、指定した日付時間値から*interval*で指定した単位より小さな部分を切り捨てます。たとえば、時間を切り捨てて、日付を出力します。"start"がデフォルト値です。
- "Actual"は、指定した日付時間値を切り捨てることなく加算を行います。
- "Fractional"は、日付時間値の間隔数を、"year"、"quarter"、"month"の長さの平均値を使って小数部まで計算します。

Date MDY(*month*, *day*, *year*)

説明

引数から日付値を生成する。

戻り値

1904年1月1日午前0時から指定の日付までの秒数

引数

month 月を示す1～12の数字。

day 日付を示す1～31の数字。エラーのチェックは行われません。たとえば、2月31日も入力されてしまうので注意が必要。

year 年。

Day(*datetime*)

説明

日付時間値（*datetime*）の日付の値を戻す。

戻り値

指定した日付（date）がその月の何日であるかを表す整数値を戻す。

引数

datetime 1904年1月1日午前0時から指定の日付までの秒数。または式でも可。

例

```
d1 = Date DMY( 12, 2, 2003 );
   3127852800

Day( 3127852800 );
   12
Day( d1 );
   12
```

Day Of Week(*datetime*)

説明

日付時間値（*datetime*）の曜日の値を戻す。

戻り値

指定の日付（date）が何曜日であるかを表す整数値を戻す。この関数では、週は、日曜日から始まり、土曜日で終わるとみなします。

引数

datetime 1904年1月1日午前0時から指定の日付までの秒数。または式でも可。

Day Of Year(*datetime*)

説明

日付時間値（*datetime*）の日付が、1年のうちで何日目かを戻す。

戻り値

指定された日付（date）が、その年の何日目であるかを表す整数値を戻す。

引数

datetime 1904年1月1日午前0時から指定の日付までの秒数。または式でも可。

Days In Month(*year*, *month*)

説明

指定の年（*year*）の指定の月（*month*）に含まれる日数を戻す。

Format(*x*, formatString, *width*|<*width*, *decimal places*>, <"Use Thousands Separator">)

Format(*x*, "Best", <*width*>, <"Use Thousands Separator">)

Format(*x*, ("Fixed Dec"|"Percent"), *width*|<*width*, *decimal places*>, <"Use Thousands Separator">)

Format(*x*, "Pvalue", <*width*>)

Format(*x*, ("Scientific"|"Engineering"|"Engineering SI"), <*width*>|<*width*, *decimal places*>)

Format(*x*,"Precision", *width*|<*width*, *decimal places*>, <"Use Thousands Separator">, <"Keep Trailing Zeroes">, <"Keep All Whole Digits">)

Format(*x*, "Currency", <*currency code*>, <*width*>|<*width*, *decimal places*>, <"Use Thousands Separator">, < <<Use Locale(*Boolean*) >)

Format(*x*, *datetime*, <*width*>)

Format(*x*, ("Latitude DDD"|"Latitude DDM"|"Latitude DMS"|"Longitude DDD"|"Longitude DDM"|"Longitude DDM"), *width*|<*width*, *decimal places*>, ("PUN"|"DIR"|"PUNDIR"))

Format(*x*, "Custom", Formula(), <*width*>)

説明

*x*の値を、後続の引数で指定された引用符付きの形式（"*format*"）に変換する。

戻り値

数値を指定された形式のテキストとして戻す。

引数

引数の詳細については、『JMPの使用法』を参照してください。引数は、データテーブルの列の「列情報」ウィンドウにも表示されます。

例

```
Format( x, "Fixed Dec", 10, 2, "Use Thousands Separator");
Format( x, "Currency", "EUR", 20, <<Use Locale(0)); // コンピュータのロケールを無視
Format( x, "m/d/y", 10 );
Format( x, "Precision", 10, 2, "Keep Trailing Zeroes", "Keep All Whole Digits" );
Format( x, "Latitude DDD", "PUNDIR"); // "PUN"はフィールド句読記号、"DIR"は方角、
   PUNDIRは両方
Format( x, "Custom", Formula( Abs( value ) ), 15 );
```

メモ

- 小数点以下の桁数を指定する場合には、その前に表示幅を指定する必要があります。
- 日付の形式が不明な場合は、ログにエラーが出力されます。

Format Date(*x*, *datetime*, <*width*>)

説明

*x*の値を、引用符付きの第2引数*datetime*で指定した形式*datetime*に変換する。選択できる形式は、データテーブルの列情報のウィンドウに表示されます。

戻り値

数値を指定された形式で戻す。

引数

引数の詳細については、『JMPの使用法』を参照してください。

例

```
Format Date( 22Feb2021, "yyyyQq" );
   "2021Q1"
```

Hour(*datetime*, <"12"|"24">)

説明

日付時間値（*datetime*）の時間の値を戻す。

戻り値

指定された日付時間値（*datetime*）が、24時間もしくは12時間のうちで何時間目になっているかを、整数で戻す。

引数

datetime 1904年1月1日午前0時から指定の日付までの秒数。または式でも可。

"12"|"24" 形式を、12時間形式にする（午前および午後における時間）。デフォルト値は24時間形式です。

HP Time()

説明

高精度の時間値（マイクロ秒単位）を戻す。この関数は、別のHP Time()値との比較で役立ちます。時間値は、JMPセッションの開始から経過したマイクロ秒数を表します。

メモ

これより精度の低い時間値については、Tick Seconds()を使用します。

In Days(*n*)

説明

日数*n*を秒数に変換して戻す。秒数をIn Days(1)で割ると、日数に変換できます。

In Hours(*n*)

説明

時間数*n*を秒数に変換して戻す。秒数をIn Hours(1)で割ると、時間数に変換できます。

In Minutes(*n*)

説明

分数*n*を秒数に変換して戻す。秒数をIn Minutes(1)で割ると、分数に変換できます。

In Weeks(*n*)

説明

週数*n*を秒数に変換して戻す。秒数をIn Weeks(1)で割ると、週数に変換できます。

In Years(*n*)

説明

年数*n*を秒数に変換して戻す。秒数をIn Years(1)で割ると、年数に変換できます。

Informat(*string*, *format*)

Parse Date(*string*, *format*)

説明

引用符付きの引数*format*で指定された形式で、引用符付きの引数*string*で指定された文字列を解析し、日付／時間の値を戻す。As Date()と同様に、日付の場合には"**ddMonyyyy**"形式で戻します。

例

```
Informat( "07152000", "MMDDYYYY" );
```

15Jul2000

メモ

- 形式のオプションを確認するには、データテーブルの列で「列情報」ウィンドウを開き、「表示形式」として日付または時間の項目を選択し、「入力形式」のリストを開きます。
- 日付の形式が不明な場合は、ログにエラーが出力されます。

次も参照

「As Date(x)」

Is Leap Year(*year*)

説明

指定の年（*year*）が閏年であれば1、そうでなければ0を戻す。

ISO Year(*datetime*)

説明

日付時間値（*datetime*）のISO年を戻す。ISO年は、ISO週に対応し、4日以上で構成される第1週の月曜日に始まる。

Long Date(*date*)

説明

与えられた日付（*date*）から、"2004年2月29日 日曜日"や"2011年11月9日 水曜日"のような、OSで指定されているロケールの日付表示を引用符付きで戻す。

MDYHMS(*date*)

説明

与えられた日付（*date*）から"2/29/04 00:02:20"のような形式の引用符付きの日付表示を戻す。

Minute(*datetime*)

説明

日付時間値（*datetime*）の分の値（0～59）を戻す。

戻り値

指定した日付時間値（*datetime*）の分を表す整数値

Month(*date*)

説明

指定された日付（*date*）の月を数値で戻す。

Nth Day of Week in the Month(*datetime*)

説明

日付時間値（*datetime*）の曜日の値と、日付時間値（*datetime*）の月にその曜日が何回起こったかを示す値を戻す。

戻り値

日付時間値（*datetime*）の曜日がその月の何度目であるかを示す整数を戻す。

Parse Date()

「Informat(string, format)」を参照してください。

Quarter(*datetime*)

説明

日付時間値（*datetime*）の四半期を示す整数（1～4）を戻す。

Second(*datetime*)

説明

日付時間値（*datetime*）の秒の値を戻す。

戻り値

指定した日付時間値（*datetime*）の秒を表す整数値

引数

datetime 1904年1月1日午前0時から指定の日付までの秒数。または式でも可。

Short Date(*date*)

説明

与えられた引用符付きの日付（*date*）をMM/DD/YYYYの形式で戻す。

Tick Seconds()

説明

スクリプトの実行にかかった時間を60分の1秒まで計測する。

メモ

これより高精度の時間値については（マイクロ秒など）、HP Time()関数を使用します。

Time Of Day(*datetime*)

説明

指定された日時（*datetime*）がその日の何秒目かを整数値で戻す。

Today()

説明

1904年1月1日午前0時から現在の日時までの秒数を戻す。引数はとりませんが、括弧は必要です。

Week Of Year(*date*, <*ruleN*>)

説明

指定された日時（date）がその年の何週目であるかを戻す。1年の第1週がいつ始まるかを決めるルール（rule）は3つあります。

- ルール1（デフォルト）では、週は日曜日から始まり、年の最初の日曜日が第2週となります。第1週は一部だけの週となるかまたは存在しません。
- ルール2では、最初の日曜日が第1週となり、その前にある日は第0週となります。
- ルール3は、ISO-8601方式の週番号を戻します。週は月曜日から始まり、その年に入ってからの4日間を含む最初の週が第1週となります。年の最初の3日間または最後の3日間が前年または翌年の週番号に属する場合があります。

Year(*date*)

説明

与えられた日付（*date*）の年を数値で戻す。

離散型確率関数

Beta Binomial Distribution(*k*, *p*, *n*, *delta*)

説明

ベータ二項分布の下側累積確率を戻す。ベータ二項分布に従う確率変数が*k*以下になる確率です。下側累積確率は、Xの値が0から*k*までの範囲において、ベータ二項分布の確率を累積した値です。

引数

k 確率を求めたい度数。*k*は整数でなければなりません。

p 各試行の成功確率。値は0～1の間でなければなりません。

n 試行回数。値は1より大きくなければなりません。

delta 過分散パラメータ。値の範囲は、Maximum[-*p*/(*n*-*p*-1), -(1-*p*)/(*n*-2+*p*)]～1です。過分散パラメータがゼロの場合、分布はBinomial(*n*, *p*)になります。

Beta Binomial Probability(*k*, *p*, *n*, *delta*)

説明

ベータ二項分布の確率を戻す。ベータ二項分布に従う確率変数が*k*と等しくなる確率を戻します。確率関数は、次式のとおりです。

$$P(X = k;p, n, \delta) = \binom{n}{k} \frac{\Gamma\left(\frac{1}{\delta}-1\right)\Gamma\left[k+p\left(\frac{1}{\delta}-1\right)\right]\Gamma\left[n-k+(1-p)\left(\frac{1}{\delta}-1\right)\right]}{\Gamma\left[p\left(\frac{1}{\delta}-1\right)\right]\Gamma\left[(1-p)\left(\frac{1}{\delta}-1\right)\right]\Gamma\left(n+\frac{1}{\delta}-1\right)}$$

引数

k 確率を求めたい度数。*k*は整数でなければなりません。

p 各試行の成功確率。値は0～1の間でなければなりません。

n 試行回数。値は1より大きくなければなりません。

delta 過分散パラメータδ。値の範囲は、Maximum[-*p*/(*n*-*p*-1), -(1-*p*)/(*n*-2+*p*)]～1です。過分散パラメータがゼロの場合、分布はBinomial(*n*, *p*)になります。

メモ

ベータ二項分布は、X|πがBinomial(*n*, π)に従い、πがBeta(*p*(1-δ)/δ, (1-*p*)(1-δ)/δ)に従うという仮定から導出できます。データが、異なる成功確率を持つ複数の二項分布の組み合わせである場合に有用です。

Beta Binomial Quantile(*p*, *n*, *delta*, *cumprob*)

説明

ベータ二項分布の分位点関数。パラメータが（*p, n, delta*）であるベータ二項分布の下側累積確率が*cumprob*以上となるような整数の分位点のうち、最小のものを戻す。

引数

p 各試行の成功確率。*p*は0～1の間でなければなりません。

n 試行回数。値は1より大きくなければなりません。

delta 過分散パラメータδ。値の範囲は、Maximum[-*p*/(*n*-*p*-1), -(1-*p*)/(*n*-2+*p*)]～1です。過分散パラメータがゼロの場合、分布はBinomial(*n*, *p*)になります。

cumprob 下側累積確率。*cumprob*は0～1の間でなければなりません。

Binomial Distribution(*p*, *n*, *k*)

説明

二項分布の下側累積確率を戻す。二項分布に従う確率変数が*k*以下になる確率です。下側累積確率は、Xの値が0から*k*までの範囲において二項分布の確率を累積した値です。

引数

p 各試行の成功確率。*p*は0～1の間でなければなりません。

n 試行回数。

k 成功回数。値は、*n*以下でなければなりません。

Binomial Probability(*p*, *n*, *k*)

説明

二項分布の確率を戻す。これは、二項分布に従う変数が*k*と等しくなる確率です。確率関数は、次式のとおりです。

$$P(X = k;p, n) = \binom{n}{k} p^k (1-p)^{n-k}$$

引数

p 各試行の成功確率。*p*は0～1の間でなければなりません。

n 試行回数。

k 成功回数。値は、*n*以下でなければなりません。

Binomial Quantile(*p*, *n*, *cumprob*)

説明

パラメータが（*p, n*）であるベータ二項分布の下側累積確率が*cumprob*以上となるような整数の分位点のうち、最小のものを戻す。

引数

p 各試行の成功確率。*p*は0～1の間でなければなりません。

n 試行回数。

cumprob 下側累積確率。*cumprob*は0～1の間でなければなりません。

Gamma Poisson Distribution(*k*, *lambda*, *sigma*)

説明

ガンマPoisson分布の下側累積確率を戻す。これは、ガンマPoisson分布に従う確率変数が*k*以下になる確率です。下側累積確率は、Xの値が0から*k*までの範囲において、ガンマPoisson分布の確率を累積した値です。

引数

k 確率を求めたい度数。*k*は整数でなければなりません。

lambda 形状パラメータλ。値は0より大きくなければなりません。これは、分布の平均です。

sigma 過分散パラメータσ。値は1以上でなければなりません。過分散パラメータが1の場合、分布はPoisson(λ)分布になります。

Gamma Poisson Probability(*k*, *lambda*, *sigma*)

説明

ガンマPoisson分布の確率を戻す。これは、ガンマPoisson分布に従う確率変数が*k*に等しくなる確率です。確率関数は、次式のとおりです。

$$P(X = k;\lambda, \sigma) = \frac{\Gamma\left(k + \frac{\lambda}{\sigma - 1}\right)}{\Gamma(k+1)\Gamma\left(\frac{\lambda}{\sigma - 1}\right)}\left(\frac{\sigma - 1}{\sigma}\right)^{k}\left(\sigma^{-\left(\frac{\lambda}{\sigma - 1}\right)}\right)$$

ここで、Γ(·)は、ガンマ関数です。

引数

k 確率を求めたい度数。*k*は整数でなければなりません。

lambda 形状パラメータλ。値は0より大きくなければなりません。これは、分布の平均です。

sigma 過分散パラメータσ。値は1以上でなければなりません。過分散パラメータが1の場合、分布はPoisson(λ)分布になります。

メモ

ガンマPoisson分布は、X|μが平均μのPoisson分布に従い、その平均がGamma(λ/(σ-1),σ-1)分布に従うという仮定から導出できます。データが、異なるμを持つ複数のPoisson(μ)分布の組み合わせである場合に有用です。

Gamma Poisson Quantile(*lambda*, *sigma*, *cumprob*)

説明

ガンマPoisson分布の分位点関数。パラメータが（*lambda*, *sigma*）であるガンマPoisson分布の下側累積確率が*cumprob*以上となるような整数の分位点のうち、最小のものを戻す。

引数

lambda 形状パラメータλ。値は0より大きくなければなりません。これは、分布の平均です。

sigma 過分散パラメータσ。値は1以上でなければなりません。過分散パラメータが1の場合、分布はPoisson(λ)分布になります。

cumprob 下側累積確率。*cumprob*は0～1の間でなければなりません。

Hypergeometric Distribution(*N*, *K*, *n*, *x*, <*r*>)

説明

超幾何分布の下側累積確率を戻す。これは、超幾何分布に従う確率変数が*x*以下になる確率です。下側累積確率は、Xの値が0から*x*までの範囲において、超幾何分布の確率を累積した値です。

必須の引数

N 母集団のサイズ。

k 母集団の中で対象となるカテゴリに属する個数。

n 標本サイズ。

x 対象となる度数。値は、*n*および*k*以下でなければなりません。

オプションの引数

r オッズ比。

Hypergeometric Probability(*N*, *k*, *n*, *x*, <*r*>)

説明

超幾何分布の確率を戻す。これは、超幾何分布に従う確率変数が*x*と等しくなる確率です。確率関数は、次式のとおりです。

$$P(X = x;N, n, k) = \frac{\binom{k}{x}\binom{N-k}{n-x}}{\binom{N}{n}}, n - x \le N - k$$

必須の引数

N 母集団のサイズ。

k 母集団の中で対象となるカテゴリに属する個数。

n 標本サイズ。

x 対象となる度数。値は、*n*および*k*以下でなければなりません。

オプションの引数

r オッズ比。

Neg Binomial Distribution(*p*, *n*, *k*)

説明

負の二項分布の下側累積確率を戻す。これは、成功確率が*p*のときに、*n*回成功するまでに失敗する回数が、*k*回以下である確率です。下側累積確率は、Xの値が0から*k*までの範囲において負の二項分布の確率を累積した値です。

引数

p 各試行の成功確率。*p*は0～1の間でなければなりません。

n 成功回数。

k *n*回目の成功の前にあった失敗の回数。

Neg Binomial Probability(*p*, *n*, *k*)

説明

負の二項分布の確率を戻す。これは、負の二項分布に従う確率変数が*k*と等しくなる確率です。確率関数は、次式のとおりです。

$$P(X = k;p, n) = \binom{n+k-1}{k} p^n (1-p)^k$$

引数

p 各試行の成功確率。*p*は0～1の間でなければなりません。

n 成功回数。

k *n*回目の成功の前にあった失敗の回数。

メモ

確率関数の戻り値は、*n*回成功するまでに*k*回失敗する確率です。

Poisson Distribution(*lambda*, *k*)

説明

Poisson分布の下側累積確率を戻す。これは、指定の平均（*lambda*）を持つPoisson分布に従う確率変数が、*k*以下になる確率です。下側累積確率は、Xの値が0から*k*までの範囲において、Poisson分布の確率を累積した値です。

引数

k 指定した時間間隔におけるイベントの発生回数。*k*は整数でなければなりません。

lambda 形状パラメータλ。値は0より大きくなければなりません。これは、分布の平均です。

Poisson Probability(*lambda*, *k*)

説明

Poisson分布の確率を戻す。これは、指定の平均（*lambda*）を持つPoisson分布に従う確率変数が、*k*に等しくなる確率です。確率関数は、次式のとおりです。

$$P(X = k;\lambda) = \frac{e^{-\lambda}\lambda^k}{k!}$$

引数

k 指定した時間間隔におけるイベントの発生回数。*k*は整数でなければなりません。

lambda 形状パラメータλ。正の値でなければなりません。これは、分布の平均です。

Poisson Quantile(*lambda*, *cumprob*)

説明

Poisson分布の分位点関数。パラメータが*lambda*であるPoisson分布の下側累積確率が*cumprob*以上となるような整数の分位点のうち、最小のものを戻す。

引数

lambda 形状パラメータλ。正の値でなければなりません。これは、分布の平均です。

cumprob 下側累積確率。*cumprob*は0～1の間でなければなりません。

表示関数

Alpha Shape(Triangulation())

説明

指定された三角分割に対して、そこから計算されるアルファシェイプを戻す。

Border Box(<Left(*pix*)>, <Right(*pix*)>, <Top(*pix*)>, <Bottom(*pix*)>, <Sides(*Boolean*)>, *db*)

説明

別のディスプレイボックスを含むことができる枠付きのディスプレイボックスを作成する。外側の枠線と、その中身との間隔を、Left（左）、Right（右）、Top（上）、Bottom（下）というオプションで指定できます。Sidesオプションによって、周りに描く枠、枠の色（黒もしくは強調色）、背景（透明もしくは白）、背景を消すかどうかを指定できます。

戻り値

ディスプレイボックス

必須の引数

db ディスプレイボックスオブジェクト（たとえば、テキストボックスや別のボーダーボックスなど）。

オプションの引数

Left(*pix*) ピクセル数を指定する整数。

Right(*pix*) ピクセル数を指定する整数。

Top(*pix*) ピクセル数を指定する整数。

Bottom(*pix*) ピクセル数を指定する整数。

Sides(*pix*) ディスプレイボックスの設定にマップする整数。

メモ

Sidesオプションには、1*top + 2*left + 4*bottom + 8*right + 16*highlightcolor + 32*whitebackground + 64*eraseという計算式で求められた整数を指定してください。たとえば、上（top）と下（bottom）に、黒色の枠線（hightlightcolor）を描きたい場合は、1+4 = 5を指定してください。同じものを、白の背景色（whitebackground）に変えて、描きたい場合は、5+32 = 37を指定してください。

Box Plot Seg(<*data*>, <*frequency*>, <*weight*>, <Vertical(*Boolean*)>)

説明

指定されたデータの箱ひげ図を描くディスプレイセグメントを戻す。

戻り値

ディスプレイボックス（箱ひげ図）

オプションの引数

data 箱ひげ図を作成するデータ値。

frequency オブザベーションの度数。

weight オブザベーションの重み。

Vertical(*Boolean*) 箱ひげ図の向き。縦（1）または横（0）を指定する。

例

```
win = New Window( "Box Plot Segの例",
Graph Box(
   Frame Size( 40, 180 ),
   Y Scale( 0, 100 ),
   Box Plot Seg(
      [20, 30, 40], // データ
      [1, 1, 3], // 度数
      [1, 1, 1], // 重み
      1 // 縦方向
      )
   )
);
```

Busy Light(< <<Automatic(*Boolean*)>, <Size(*x*, *y*)>, < <<Disable>)

説明

ビジー状態を示す、回転するイメージを作成する。

戻り値

回転するイメージ

オプションの引数とメッセージ

<<Automatic(*Boolean*) イメージを回転させる。

Size(*x*, *y*) イメージのサイズをピクセルで指定する。

<<Disable イメージを非表示にする。

例

```
win = New Window( "例",
   blb = Busy Light( <<Automatic( 1 ), Size( 50, 50 ) ) );
```

Button Box(*title*, *script*, < <<Set Icon(*path*)>, < <<Set Icon Location(left|right)>

説明

クリックするとスクリプト（*script*）を実行する、*title*という名前のボタンを作成する。

戻り値

ディスプレイボックス（ボタンボックス）

必須の引数

title 引用符付き文字列、または文字列変数。

script 有効なJSLスクリプトを指定する引用符付き文字列、または文字列変数。

オプションのメッセージ

<<Set Icon("*path*") 引用付きパス名にあるイメージをボタン上に表示する。GIF、JPG、PNG、BMP、TIFなどの一般的なグラフィック形式に対応しています。このオプションを指定しなかった場合、テキストだけのボタンが作成されます。titleオプションの引数に空の文字列を指定し、アイコンだけを指定した場合、アイコンだけが表示されたボタンが作成されます。両方のオプションを指定すると、テキストとアイコンの両方が表示されたボタンが作成されます。その場合、アイコンは、テキストの隣に表示されます。

<<Set Icon Location("left"|"right") ボタン上のアイコンをテキストの左または右に配置する。

例

次の例は、シンプルなボタンボックスを表示します。ユーザーがボタンをクリックすると、ログに「押されました」と出力されます。

```
win = New Window( "シンプルな例",
   ex = Button Box( "クリックしてください" )
);
ex << Set Script( Print( "押されました" ) );
```

メモ

Button Boxでは改行文字が無視されます。

Calendar Box(*title*, < <<Date>, < <<Min Date>, < <<Max Date>, < <<Show Time>)

説明

日時を選択できる、ポップアップカレンダーを作成する。

戻り値

ディスプレイボックス（カレンダーボックス）

必須の引数

title 引用符付き文字列、または文字列変数。

オプションのメッセージ

<<Date 現在選択されている日付。

<<Min Date 選択可能な最初の日付。

<<Max Date 選択可能な最後の日付。

<<Show Time 時間の選択を可能にする。

例

次の例は、1989年10月5日が選択されている状態のカレンダーを作成します。また、選択可能な最初の日付と最後の日付を指定し、ユーザが選択可能な期間を限定します。

```
New Window( "カレンダーボックスの例", cal = Calendar Box() );
date = Date MDY (10, 5, 1989);
cal << Date( date );
cal << Show Time( 0 ); // 時間は省略

/* 選択範囲の最初の日付は1989/10/5から60日前
```

```
"start"を指定して、時間の値を切り捨てる */
cal << Min Date( Date Increment(date, "Day", -60, "start" ) );

// 選択範囲の最後の日付は1989/10/05から60日後
cal << Max Date( Date Increment(date, "Day",  60, "start" ) );

cal << Set Function( Function( {this}, Print( Abbrev Date(this << Get Date()) ) )
  ); // 短い表示形式で日付をログに出力
```

Cell Plot(*Y(column(s))*, <*X(column)*>)

説明

セルプロットを表示する。

Check Box({*list*}, <*script*>, < <<Get(*n*)>, < <<Set(*n*, *Boolean*)>, < <<Get Selected>, < <<Enable Item(*n*, *Boolean*)>, < <<Item Enabled(*check box item*)>)

説明

1つまたは複数のチェックボックスを表示するディスプレイボックスを作成する。

戻り値

ディスプレイボックス（チェックボックス）

必須の引数

list 引用符付き文字列を含むリスト。

オプションの引数

script（オプション）JSLスクリプト。

オプションのメッセージ

<<Get(*n*) *n*で指定されたチェックボックス項目が選択されている場合は1、そうでない場合は0を戻す。

<<Set(*n*, *Boolean*) *n*で指定されたチェックボックス項目を選択（1）または選択解除（0）する。

<<Get Selected 選択されているチェックボックス項目の名前を含む引用符付き文字列リストを戻す。

<<Enable Item(*n*, *Boolean*) *n*で指定されたチェックボックス項目を有効（1）または無効（0）にする。無効になっているチェックボックスの状態は変更できません。

<<Item Enabled(*check box item*) 指定されたチェックボックス項目が有効の場合は1、そうでない場合は0を戻す。

例

「one」、「two」、「three」というラベルがついた3つのチェックボックスを作成します。その際、最初のチェックボックスを選択された状態にします。

```
New Window( "例", Check Box( {"one", "two", "three"}, <<Set( 1, 1 ) ) );
```

Col Box(*title*, *display boxes*)

説明

指定されたディスプレイボックスを含む列ボックスを戻す。

引数

title 列の引用符付きタイトル。

display boxes 列ボックス内に含めるディスプレイボックス。

例

```
win = New Window( "例",
   exx = 1;
   exy = 4;
   exz = 8;
   Table Box(
      String Col Box( "strings", {"x", "y", "z"} ),
      Col Box(
         "boxes",
         Slider Box( 0, 10, exx, Show( exx ) ),
         Slider Box( 0, 10, exy, Show( exy ) ),
         Slider Box( 0, 10, exz, Show( exz ) )
      )
   );
);
```

Col List Box(<Data Table(*name*)>, <"All"|"Character"|"Numeric">, <width(*pixels*)>, <"Grouped">, <MaxSelected(*n*)>, <nLines(*n*)>, <MaxItems(*n*)>, <MinItems(*n*)>, <On Change(*expr*)>, < <<Modeling Type("Any"|"Continuous"|"Ordinal"|"Nominal"|Multiple Response"|"Unstructured Text"|"Vector"|"None"|"Row State") >, < <<Set Data Type(Any|Numeric|Character)>, <*script*>)

説明

データテーブルの列を選択できるリストボックスを表示するディスプレイボックスを作成する。

戻り値

ディスプレイボックス（Col List Box）

オプションの引数

Data Table(*name*) データテーブルの引用符付きの名前。

"All" | "Character" | "Numeric" 現在のデータテーブルのすべての列をリストに追加する。"All"を省略すると、“オプション”ラベルの付いた空の列リストボックスが表示されます。“オプション(文字)”を表示する場合は"Character"を指定します。“オプション(数値)”を表示する場合は"Numeric"を指定します。

width(*pixels*) リストボックスの幅をpixelsで設定する。pixelsはピクセル数を示す数値。

"Grouped" グループ化されている列がある場合、Col List Boxの中でもグループ化して表示する。

MaxSelected(*n*) 選択できる項目が1つだけかどうかを設定する。*n*<1の場合、*n*は無視される。

nLines(*n*) リストボックスの長さを*n*行に設定する。*n*は整数。

script スクリプト。

MaxItems(*n*) リストに追加できる列の数を*n*個に制限する。

MinItems(*n*) リストに少なくとも*n*個の列を追加するように求める。たとえば、*n*=2の場合、Col List Boxの最初の2つの項目に、「必須」、「必須(文字)」、もしくは「必須(数値)」と表示される。

On Change(*expr*) リストの選択内容に変更が生じるたびに式を評価する。この引数を持つ2つの列リストボックスの間をドラッグすると、両方の式が評価されます。ドラッグ先の式がまず評価され、その後ドラッグ元の式が評価されます。

オプションのメッセージ

<<Set Tips({*Tip text 1*, *Tip text 2*, ...}) リストボックス内の項目にツールヒントを設定する引用符付き文字列。引用符付きヌル文字列または空のリストを指定した場合、ツールヒントは設定されません。リストボックス内の項目のリストよりも、指定したリストの方が短い場合、最後のツールヒントテキストが、リストボックス内の残りのリスト項目にも使用されます。

<<Set Tip(*Tip text*) Set Tips()関数で設定されたツールヒントを上書きする引用符付き文字列。ボックス自体にヒントが設定されている場合、個々のリスト項目のヒントを設定することはできません。

Set Tip()を引数未指定で使用すると、リストボックスのヒントが消去され、各リスト項目のツールヒントが表示されるようになります。

メモ

- MaxSelected(*n*)の引数は、項目を1つだけ選択できるか、または複数選択できるかだけを決定するものです。2つ以上の列を強制的に選択させるものではありません。
- 特殊な尺度の列は、その尺度の列を明示的に受け入れると指定している場合のみ割り当てることができます。

Col Span Box(*title*, *display box args*)

説明

テーブルボックスに、列見出しを作成する。最上部の列見出しは、2つの子列ヘッダにまたがります。

戻り値

ディスプレイボックス（Col Span Box）

引数

title ボックスに表示されるタイトル。

display box args ディスプレイボックス。

例

```
win = New Window( "Col Span Box",
   <<Modal,
   Table Box(
      Col Span Box(
         "信頼限界",
```

```
                neb = Number Col Edit Box( "上側限界", [0, 0] ),
                Number Col Edit Box( "下側限界", [0, 0] )
            )
        )
    );
```

Combo Box({*items <(tip string)>, ...*}, *<script>*)

説明

ドロップダウンリストを表示するためのディスプレイボックスを作成する。

戻り値

ディスプレイボックス（コンボボックス）

引数

items ユーザが選択できる項目。

tip string ツールチップのテキストを指定する引用符付き文字列。

script（オプション）JSLスクリプト。

Context Box(*display box*, ...)

説明

変数や式のスコープを限定するディスプレイボックスを定義する。各コンテキストボックスは、独立して、個別に評価・実行されます。

戻り値

ディスプレイボックス

引数

任意の数のディスプレイボックス

Contour Seg(Triangulation, [levels], <zColor([*colors*]|{*colors*}, <"Cycle Colors"|"Interpolate Colors">)>, <"Fill"|"Fill Between"|"Fill Below"|"Fill Above">, <Transparency([]|*t*)>)

説明

三角分割の等高線を表すディスプレイセグメントを戻す。

必須の引数

Triangulation 三角要素座標に含む列。

[levels] 等高線の値を示す行列。

オプションの引数

zColor([*colors*]|{colors} リストまたは行列として指定した各水準の色。

"Cycle Colors"|"Interpolate Colors" このうち、"Cycle Colors"は、色を交互に切り替える（たとえば、赤、緑、赤、緑）。一方、"Interpolate Colors"では、例えば、最初の等高線が赤、

最後の等高線が緑である場合には、その間にある等高線には、赤と緑を混ぜたグラデーションが使用される。

"Fill Below"|"Fill Between"|"Fill Above"|"Fill" このうち、"Fill Below"は、線の下の領域を塗りつぶす。"Fill Between"は、線の間の領域を塗りつぶす。"Fill Above"は、線の上の領域を塗りつぶす。"Fill"は、"Fill Above"と同様に動作しますが、Fillのオプションが指定されていない場合はデフォルトで線を表示します。

Transparency([]|*t*) 透明度を数値または行列で指定する。

例

```
dt = Open( "$SAMPLE_DATA/Cities.jmp" );
tri = Triangulation( X( :X, :Y ), Y( :POP ) );
{xx, yy} = tri << Get Points();
win = New Window( "Contour Segの例",
   g = Graph Box(
      X Scale( Min( xx ) - .1, Max( xx ) + .1 ),
      Y Scale( Min( yy ) - .1, Max( yy ) + .1 ),
      Contour Seg(
         tri,
         [0, 400, 1000, 2000, 9000],
         zColor( 5 + [64 32 0 16 48] ),
         Transparency( [1, 1, 1, 1, 1] )
      )
   )
);
```

メモ

三角分割は、2つのXを使って計算され、Yは、各位置で定義された連続尺度の変数です。この例の[levels]は、等高線を描く人口の値を設定しています。Fillの引数が指定されている場合、塗りつぶされる領域は、[level1, level2]、[level2, level3]...[level-n]です。

Current Journal()

説明

現在のジャーナルの最上位のディスプレイボックスへの参照を戻す。

戻り値

現在のジャーナルの最上位のディスプレイボックスへの参照

Current Report()

説明

現在のレポートウィンドウの最上位のディスプレイボックスへの参照を戻す。

戻り値

現在のレポートウィンドウの最上位のディスプレイボックスへの参照

Current Window()

説明

現在のウィンドウへの参照を戻す。

Data Filter Context Box(*display box*)

説明

表示ツリーに含まれるローカルデータフィルタの範囲を定義するディスプレイボックスを戻す。Data Filter Context Boxとデータフィルタを階層形式で配置することで、Data Filter Context Boxに含まれるプラットフォームまたはボックス間でフィルタを共有できます。

Data Filter Source Box(*display box*)

説明

どのグラフが選択フィルターの「ソース」であるかを定義する。Data Filter Source Boxの中あるレポートで選択された行が、同じData Filter Context Box内にある他のレポートの分析対象となります。

Data Table Box(*data table*)

説明

指定されたデータテーブルを表示するTable Boxを戻す。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
win = New Window( "例", Data Table Box( dt ) );
```

Data Table Col Box(*column*)

説明

指定されたデータテーブルを表示する列ボックスを戻す。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
win = New Window( "例",
   Table Box( Data Table Col Box( :名前 ), Data Table Col Box( :"身長(インチ)"n ) )
);
```

Dialog(*contents*)

説明

Dialogは廃止されました。Dialog関数の代わりに、Modal引数とともにNew Window()関数を使用してください。

Excerpt Box(*report*, *subscripts*)

説明

レポートから一部分を抜粋し、その抜粋したディスプレイボックスを戻す。レポートの番号を *report* で指定し、ディスプレイのリストを *subscripts* で指定します。*subscripts* に指定する番号は、抜粋したディスプレイを除いた後の番号。

Expr As Picture(*Expr*(...), <Width(*pixels*)>)

説明

計算式エディタに表示される式と同じ形式で、expr() をイメージに変換する。

戻り値

イメージへの参照

必須の引数

Expr(...) イメージとして表示できる有効なJSL式をexpr()の中に挿入。

オプションの引数

Width(*pixels*) ボックスの幅をpixに設定する。pixはピクセル数を示す数値。

Filter Col Selector(<Data Table(*name*)>, <Width(*pixels*)>, <nLines(*n*)>, <*script*>, <OnChange(*expr*)>)

説明

列名のリストを含むディスプレイボックスを戻す。列のフィルタリングを設定できる。

Get Project(*title*|*index*|*display box*|*window*)

説明

プロジェクトを戻す。

例

次の例は、さまざまなプロジェクトのウィンドウタイトルを取得する方法を示します。

```
Get Project( 1 ) << Get Window Title;
// 開いている最初のプロジェクト
Get Project( "マイプロジェクト" ) << Get Window Title;
// 「マイプロジェクト」という名前の最初のプロジェクト
Get Project( display box ) << Get Window Title;
// 指定したディスプレイボックスの親プロジェクト
```

Get Project List()

説明

現在開いているすべてのプロジェクトをリストで戻す。

例

```
Get Project List() << Get Window Title;
// 開いているすべてのプロジェクトのタイトルのリスト
```

Get Window(<Project(*title*|*index*|*display box*|*window*)>, <Type(*string*)>, *title*|*index*|*display box*)

説明

指定したタイトル、通し番号、またはディスプレイボックスを持つウィンドウへの参照を戻す。Get Window()をプロジェクトの中で実行した場合、現在のプロジェクトのウィンドウが戻されます。

オプションの引数

Project プロジェクトのタイトル（引用符付き文字列）、通し番号、ディスプレイボックス、またはウィンドウを指定する。

Type(*string*) 検索対象を特定のウィンドウタイプに限定するには、Type ()引数と、"Data Tables"、"Journals"、"Reports"、"Dialogs"のいずれかの引用符付き文字列を使用します。

例

次の例は、さまざまなウィンドウのウィンドウタイトルを取得する方法を示します。

```
Get Window( 1 ) << Get Window Title;
// 現在のプロジェクトの最初のウィンドウ

Get Window( "Big Class" ) << Get Window Title;
// 現在のプロジェクトの「Big Class」ウィンドウ

Get Window( ob ) << Get Window Title;
// 現在のプロジェクトにある指定のディスプレイボックスの親ウィンドウ

Get Window( Project(), 1 ) << Get Window Title;
// 最初のウィンドウ、プロジェクトなし（グローバルスコープ）

Get Window( Project( myProject ), "Big Class" << Get Window Title;
// 指定したプロジェクトの「Big Class」ウィンドウ
```

Get Window List(<Project(*title*|*index*|*display box*|*window*),><Type(*string*)>)

説明

現在開いているウィンドウのリストを戻す。プロジェクト内で実行された場合、Get Window List()は、現在のプロジェクトで開いているウィンドウへの参照を戻します。現在のプロジェクト以外のプロジェクトから、開いているウィンドウのリストを取得するには、Project()引数を使用します。検索対象を特定のウィンドウタイプに限定するには、Type ()引数と、"Data Tables"、"Journals"、"Reports"、"Dialogs"のいずれかの引用符付き文字列を使用します。

オプションの引数

Project プロジェクトのタイトル、通し番号、ディスプレイボックス、またはウィンドウを指定します。

Type 検索対象を特定のウィンドウタイプに限定するには、Type ()引数と、"Data Tables"、"Journals"、"Reports"、"Dialogs"のいずれかの引用符付き文字列を使用します。

例

```
Get Window List() << Get Window Title;
// 現在のプロジェクト内で開いているウィンドウのリスト
Get Window List( Type( "Reports" ) ) << Get Window Title;
// 現在のプロジェクト内で開いているレポートのタイトルのリスト
Get Window List( Project( 0 ), Type( "Reports" ) ); // 位置引数
// プロジェクトの外で開いているレポートのタイトルのリスト
Get Window List( 2 );
// 2番目のウィンドウリスト
```

Global Box(*global*)

説明

グローバル変数（*global*）の値を直接編集するボックスを作成する。

Graph()

「Graph Box(properties, script)」を参照してください。

Graph 3D Box(*properties*)

説明

3次元散布図を含むディスプレイボックスを作成する。

戻り値

ディスプレイボックス

引数

properties プロパティには、Frame Size(x, y)、Xname("title")、Yname("title")、Zname("title")がある。

メモ

この機能は試験段階です。

Graph Box(*properties*, *script*)

Graph(*properties*, *script*)

説明

X軸およびY軸のあるグラフを作成する。

戻り値

ディスプレイボックス（グラフボックス）

引数

properties 引数には、Title("title")、XScale(low, high)、YScale(low, high)、FrameSize(h, v)、XName("x")、YName("y")、SuppressAxesがある。

script グラフボックスで実行するスクリプト。

H Center Box(<*child box*>)

オプションの引数で指定した子ディスプレイボックスを含むディスプレイボックスを戻す。子ディスプレイボックスを横方向のスペースの中央に配置します。

H List Box(<Align("center"|"bottom")>, display box, ...)

説明

ディスプレイボックスを作成し、その中に別のディスプレイボックスを横に並べて表示する。

引数

Align("center"|"bottom") ディスプレイボックスの位置を、中央揃え（center）または下揃え（bottom）に指定する。デフォルトでは、下揃えで配置されます。

display box リストボックスに含める任意の数のディスプレイボックスを指定する。

H Scroll Box(<Size(*h*)>, *display box*)

説明

横方向のスクロールバーがついたディスプレイボックスを戻す。中により大きな子ボックスを配置することができます。

引数

Size(*h*) スクロールバーの横方向の長さ。

メモ

flexible引数は廃止されています。代わりに、Set Stretchを使用してください。例については、「V Scroll Box(<Size(v)>, display box)」を参照してください。

H Sheet Box(<<Hold(*report*), *display boxes*)

説明

引数に指定された複数のディスプレイボックスを横方向に配置したディスプレイボックスを戻す。<<Hold()メッセージは、抜粋元となるレポートをどのシートが保持するかを示す。

H Splitter Box(<Size(*h*, *v*)>, *display box*, ...)

説明

引数に指定された複数のディスプレイボックスを横方向（パネル）に配置し、分割線で仕切ったディスプレイボックスを戻す。ユーザは分割線をドラッグしてパネルのサイズを変更できます。

必須の引数

display box 分割線ボックスには任意の数のディスプレイボックスの引数を設定できる。

オプションの引数

Size(*h*, *v*) Splitter Boxのサイズをピクセルで指定する。このサイズは外側のSplitter Box用です。内側のディスプレイボックスのサイズは、外側のSplitter Boxの幅と高さに合わせて変更されます。

オプションのメッセージ

<<Size(*n*) 最後のパネルの比率を指定する。<<Size(.25)は、最後のパネルのサイズを分割線ボックスの高さ（H Splitter Boxの場合は幅）の25%に変更します。

<<Set Sizes({*n*, *n*}) 各パネルの比率を指定する。db<<Set Sizes({.75, .25})は、最初のパネルのサイズを分割線ボックスの高さ（H Splitter Boxの場合は幅）の75%に変更し、2番目のパネルのサイズを同25%に変更します。

<<Close Panel(*n*, *<Boolean>*) 指定したパネルを閉じる。<<Close Panel(2)は、2番目のパネルを閉じます。3つ以上のパネルがある場合は、2番目のブール値を含める必要があります。その値により、閉じたパネルによって余ったスペースを埋めるパネルを指定します。

- <<Close Panel(2,0)は、2番目のパネルを閉じ、その後のパネルが余ったスペースを埋めます。
- <<Close Panel(2,1)は、2番目のパネルを閉じ、その前のパネルが余ったスペースを埋めます。

<<Open Panel(*n*, *<Boolean>)* 指定したパネルを開く。3つ以上のパネルがある場合は、2番目のブール値を含める必要があります。前述の<<Close Panelと同様に機能します。パネルは初期状態ですべて開いているため、<<Open Panelは<<Close Panelを使用した後にのみ使用します。

<<Get Sizes() 各パネルの比率をリストで戻す。

Hier Box(text, Hier Box(...), ...)

説明

テキスト（text）が含まれているツリーのノードを作成する。これは、特性要因図プラットフォームの出力と同じです。Hier Boxは別のHier Boxを含めることができ、ツリーを作成できます。テキストは、引用符付き文字列またはText Edit Boxでもかまいません。

Hist Seg([*data*], <[*freq column*]>, <[*weight column*]>, <Vertical(*Boolean*)>, <Row States()>

説明

ヒストグラムセグメントを戻す。

必須の引数

data 行列形式のデータ。

オプションの引数

freq column 行列形式の度数列。

weight column 行列形式の重み列。

Vertical(*Boolean*) デフォルトで（または1に設定されている場合）、ヒストグラムを縦に表示する。値が0に設定されている場合は、ヒストグラムを横に表示する。

Row States データテーブル参照または行の属性を指定する。

Icon Box(*name*)

説明

アイコンを含むディスプレイボックスを作成する。引数で指定する名前（name）には、Popup 、Locked 、Labeled 、Sub 、Excluded 、Hidden 、Continuous 、Nominal 、Ordinal があります。引数には、イメージへのパスを指定することもできます。

引数

name JMPのアイコン名またはアイコンのパスを示す引用符付き文字列。

例

```
Icon Box( "Nominal" )
```
は、名義尺度アイコンを含むディスプレイボックスを作成します。

```
Icon Box( "$SAMPLE_IMAGES/pi.gif" )
```
は、pi.gifサンプルイメージを挿入します。

メモ

- WindowsとmacOSの両方で使用されているアイコンと、その他のアイコンは、それぞれのプラットフォームでのみ使用可能です。
- Built-In JMP Iconsというアドインがあり、いろいろなコンテキストでアイコンがどのように表示されるかを確認することができます。
  このアドインは、https://community.jmp.com/t5/JMP-Add-Ins/Built-In-JMP-Icons/ta-p/42251からダウンロードできます。

If Box(*Boolean, display boxes*)

説明

条件によって中身が表示されるディスプレイボックスを作成する。

引数

Boolean 1はIf Boxの内部にあるディスプレイボックスを表示し、0である場合は表示しない。

display boxes 任意のディスプレイボックスツリー。

If Seg(<State(*Boolean*)>)

説明

セグメントの子を表示または非表示にするディスプレイセグメントを戻す。

引数

State(*Boolean*) 子のディスプレイセグメントを表示する（1）か非表示にする（0）かを決定する。

例

```
lines = [30 20 80 70, 10 90 90 10, 40 20 60 30];
win = New Window( "Lines Segの例",
   g = Graph Box( If Seg( true, <<Append( Lines Seg( lines ) ) ) )
);
```

Journal Box(*string*)

説明

引用符付き文字列から生成されるジャーナルを表示するディスプレイボックスを作成する。手動でジャーナルテキストを生成することはお勧めしません。

Line Seg(*x*, *y*, <Row States(*dt*|*dt*, [*rows*]|dt, {{*rows*}, ...}|{*row states*})>, <Sizes(*s*)>)

説明

指定の*x*と*y*の値について、つながった線分を描くディスプレイセグメントを作成する。オプションの第3引数では、データテーブルの行属性を使用するか、またはそれとは別のものを使用するかを指定できます。

Lines Seg([*x1 y1 x2 y2*, ...])

説明

指定の*x*と*y*の値について、複数の線分を描くディスプレイセグメントを戻す。

Lineup Box(<nCol(*n*)>, <Spacing(*pixels*, <*vspace*>), *display boxes*, ...)

説明

*n*列にボックスを整列させて表示するディスプレイボックスを作成する。

List Box({*item*, ...}, <Width(*pixels*)>, <maxSelected(*n*)>, <nLines(*n*)>, <*script*>)

説明

複数の項目（引用符付き文字列）を含むリストボックスを表示するディスプレイボックスを作成する。引数*item*には、項目名の文字列を含むリストを指定します。もしくは、項目名の文字列と、尺度または並べ替え順序を示す引用符付き文字列を含むリストのリストを指定します（なお、項目名では、デフォ

ルトで大文字・小文字が区別されます)。後者の場合、リストボックス内の項目の横に、尺度または並べ替え順序のアイコンが表示されます。

Marker Seg(*x*, *y*, <Row States(*dt*|*dt*, [*rows*]|*dt*, {{*rows*}, ...}|{*row states*})>, <Sizes(*s*)>)

説明

指定された*x*と*y*のすべての値にマーカーを描くディスプレイセグメントを作成する。オプションの第3引数では、データテーブルの行属性を使用するか、またはそれとは別のものを使用するかを指定できます。

Matrix Box(*x*)

Matrix Box(*matrix*, < <<Column Names(*col1*, *col2*, ...)>, < <<Row Names(*row1*, *row2*, ...)>>)

説明

与えられた行列（*matrix*）を表示する。列と行の名前は引用符付き文字列です。

Mouse Box(*display box arguments*)

説明

ドラッグ&ドロップ、マーキング、クリック、トラックなどのマウス動作に対するJSLコールバックを作成するボックスを戻す。

引数

display box arguments Text BoxやButton Boxなど、ユーザが操作できるオブジェクトを指定する。［ヘルプ］メニューの［スクリプトの索引］を参照してください。

Move to Project(<Source(*project*)|Destination(*project*)>, <Windows({*list of windows to move*})>)

説明

1つまたは複数のウィンドウをプロジェクトから外へ（またはプロジェクトの中に）、またはプロジェクト間で移動させる。

引数

Source(*project*) 移動させたいウィンドウを含むプロジェクト。

Destination(*project*) 移動先のプロジェクト。

Windows({*list of windows to move*}) 移動するウィンドウのリスト。これを省略した場合、すべてのウィンドウが移動します。windows引数にデータテーブルを指定した場合、そのデータテーブルから作成されたすべてのレポートも移動します。レポートを指定した場合も、そのレポートのもとになったデータテーブルとそのテーブルから作成されたすべてのレポートが移動します。windows引数の中で指定するのはデータテーブルまたはレポートの名前です。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
report = dt << Run Script( "Bivariate" );
project = New Project();
// レポートとデータテーブルを新しいプロジェクトに移動させる
Move to Project( Destination( project ), Windows( {report} ) );
```

New Image()

New Image(*width*, *height*)

New Image(*filepath*)

New Image(Open(*url*))

New Image(*picture*)

New Image(*matrix*)

New Image(rgb|r|rgba, {*matrix*, ...})

説明

新しいイメージを作成する。このイメージは、JSLで編集できます。有効な形式は、png、bmp、jpeg、jpg、tiff、tif、gifです。

戻り値

イメージ

引数

すべての引数セットはオプションだが、それぞれのセット内ではすべての引数が必須。

width, height イメージの幅と高さをピクセルで設定する。

filepath イメージへの引用符付きファイルパス。

Open(*url*) 指定したURLパスにあるイメージを開きます。

picture JSLピクチャーオブジェクト。

matrix イメージをJSLカラーのピクセルの配列で示したもの。

rgb|r|rgba, {*matrix*, ...} チャネルを指定し（"rgb"、"r"、または"rgba"）、各チャネルの値の行列（0.0-1.0）を指定する。

例:

```
New Image( "r", [r matrix] );
New Image( "rgb", {[r matrix], [g matrix], [b matrix]} );
New Image( "rgba", {[r matrix], [g matrix], [b matrix], [a matrix]} );
```

New Project(*arguments*)

説明

新しいプロジェクトを作成する。

引数

<<Add Bookmarks({<File(*path*)>, <Folder(*path*, Expanded(*Boolean*)>)>, <Group(*name*, <Expanded(*Boolean*)>, {*contents*}>) プロジェクトで頻繁に使用されるファイルをブックマークに追加する。引数は、ブックマーク項目のリストで、それぞれをFile()、Folder()、またはGroup()を使って指定します。Group()には、File()、Folder()、およびGroup()を子として指定できます。

<<Reset Layout プロジェクトのウィンドウレイアウトをデフォルトの状態に戻す。

<<Run Script プロジェクトの中に表示するデータテーブルとレポートを指定する。

<<Save(<*path*>) プロジェクトを保存する。プロジェクトを特定の場所に保存するには、引用符付きのパスおよびファイル名を含めます。Save Asはエイリアスです。

<<Set Bookmarks({<File(*path*)>, <Folder(*path*, Expanded(*Boolean*)>)>, <Group(*name*, <Expanded(*Boolean*)>, {*contents*}>) プロジェクトのブックマークを設定する。引数は、ブックマーク項目のリストで、それぞれをFile()、Folder()、またはGroup()を使って指定します。Group()には、File()、Folder()、およびGroup()を子として指定できます。

<<Set Layout プロジェクトのウィンドウレイアウトを設定する。

<<Show Bookmarks ブックマークの表示／非表示を切り替える。

<<Show Log ログの表示／非表示を切り替える。

<<Show Window List ウィンドウリストの表示／非表示を切り替える。

例

次の例は、「BigClass.jmp」と2つのレポートを含むプロジェクトを作成します。

```
project = New Project();
project << Run Script(
   dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
   dt << Run Script( "二変量の関係(二変量) " );
   dt << Run Script( "一変量の分布" );
);
```

New Window(*title*, <optional *arguments*>, *displayBox* | *script*)

説明

指定されたタイトル（必須、引用符付き）を含む新しいウィンドウを作成する。

必須の引数

displayBox 任意のディスプレイボックス。

script ウィンドウに表示するスクリプト。Type("Script")においてのみ有効。

オプションの引数

<<Type *("Report" | "Dialog" | "Modal Dialog" | "Journal" | "Launcher" | "Script")*

指定されたタイプのウィンドウを作成する。タイプを指定しなかった場合、レポートウィンドウが作成されます。Type("Modal Dialog")のウィンドウは、閉じられるまでJMP内の他のアクションが実行できなくなります。

<<Return Result ウィンドウが閉じるときの戻り値を、廃止されたDialog()関数の戻り値と一致するよう変更する。Type("Modal Dialog")においてのみ有効。

<<On Open(*expr*) ウィンドウの作成時に式（expr）を実行する。Type("Modal Dialog")においてのみ有効。

<<On Close(*expr*) ウィンドウが閉じるときに式（expr）を実行する。式がFalseを戻した場合、ウィンドウは閉じません。ウィンドウを閉じることができない場合は0を返します。

<<On Validate(*expr*) [**OK**] ボタンがクリックされたときに式（expr）を実行する。式がTrueを戻した場合、ウィンドウは閉じます。それ以外の場合は、ウィンドウは開いたままです。Type("Modal Dialog")においてのみ有効。

<<Show Menu(*Boolean*) メニューバーの表示／非表示を切り替える。デフォルト値は1です。Type("Report")においてのみ有効。Windowsのみ。

<<Show Toolbars(*Boolean*) ツールバーの表示／非表示を切り替える。デフォルト値は1です。Type("Report")においてのみ有効。Windowsのみ。メニュー領域を右クリックすると、表示するツールバーをカスタマイズできます。指定した内容はセッションを越えて保持されます。

<<Suppress AutoHide(*Boolean*) メニューおよびツールバーの自動非表示機能の使用／不使用を切り替える。デフォルト値は1です。Windowsのみ。Type("Report")においてのみ有効。

<<Window View(*"Visible" | "Invisible"*) *"Invisible"*は、"Modal Dialog"を除く全種類のウィンドウに対して指定可能です。

<<Size Window(*width*, *height*) 新しいウィンドウの幅と高さをピクセルで指定する。

メモ

データテーブルでは、別のプログラムを実行するOn Open（またはOnOpen）スクリプトは、実行されません。このスクリプトが実行されるタイミングを制御する必要がある場合は、環境設定の［OnOpenスクリプトの評価］を設定してください。

Dialog()は廃止されました。代わりに、New Window()とType("Modal Dialog")を組み合わせて使用してください。

Number Col Box(*title*, *numbers*)

説明

引用符付き文字列で指定されたタイトル（*title*）をつけた列を作成し、リストまたは行列の形で与えられた数値を挿入する。

Number Col Edit Box(*title*, {*numbers*}|[*numbers*])

説明

引用符付き文字列で指定されたタイトル（*title*）をつけた列を作成し、リストまたは行列の形で与えられた数値を挿入する。この関数によって作成された列の数値は、編集できます。

Number Edit Box(*initial value*, <*width*>)

説明

*initial value*引数で指定された数値が入力された数値ボックスを作成する。

戻り値

ディスプレイボックスオブジェクト

引数

initial value 初期値として使用する数値。日付または時間の形式を使用すると、日付または時間のセレクタウィンドウが作成されます。

<*width*> ボックスの幅を文字数で設定する。

Outline Box(*title*, *display box*, ...)

説明

指定された複数のディスプレイボックスを含んだ、*title*（引用符付き文字列）という名前の新しいアウトラインを作成する。

Page Break Box()

説明

ウィンドウが印刷された場合、改ページを強制するディスプレイボックスを作成する。

Panel Box(*title*, *display box*)

説明

指定されたタイトル（*title*）のラベルをもつパネルのディスプレイボックスを作成する。そのパネルには、複数のディスプレイボックスを含めることができます。

Picture Box(Open(*picture*), *format*)

説明

グラフィックピクチャーオブジェクトを含むディスプレイボックスを作成する。

戻り値

ディスプレイボックスへの参照

引数

Open(*picture*) ピクチャを含むディレクトリを開く。

format グラフィックファイルの形式を指定する。JMPでピクチャーを開く形式を指定します。この引数を指定しないと、ピクチャーはデフォルトのグラフィックプログラムで開かれます。有効な形式は、引用符付き文字列の"png"、"bmp"、"jpeg"、"jpg"、"tiff"、"tif"、"gif"です。

例

```
New Window( "例",
```

```
Picture Box( Open( "$SAMPLE_IMAGES/pi.gif", gif ) ) );
```

Platform(*data table*, *script*)

説明

指定されたスクリプトを指定されたデータテーブルのコンテキストにおいて評価する。

戻り値

表示ツリーに埋め込むための結果のディスプレイボックス

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
win = New Window( "Platformの例",
   H List Box(
      Platform(
         dt,
         Bubble Plot(
            X( :"体重( ポンド )"n ),
            Y( :"身長( インチ )"n ),
            Sizes( :年齢 ),
            Title Position( 0, 0 )
         )
      ),
      Platform(
         dt,
         Bubble Plot(
            X( :"体重( ポンド )"n ),
            Y( :年齢 ),
            Sizes( :"身長( インチ )"n ),
            Title Position( 0, 0 )
         )
      )
   )
);
```

Plot Col Box(*title*, *numbers*)

説明

数字（*numbers*）をグラフ化したディスプレイボックスに、引用符付き文字列で指定されたタイトル（*title*）を付けて戻す。数字にはリストまたは行列を指定できます。

Poly Seg(*x values*, *y values*)

説明

指定された（*x values, y values*）座標を通るポリゴン（多角形）を描くディスプレイセグメントを戻す。

例

```
x = [10, 50, 90];
y = [10, 90, 10];
win = New Window( "Poly Segの例",
g = Graph Box( Poly Seg( x, y ) ) );
frame = g[FrameBox( 1 )];
seg = (frame << Find Seg( "Poly Seg" ));
```

Popup Box({*command1*, *script1*, *command2*, *script2*, ...})

説明

赤い三角形ボタンのメニューを作成する。引数のリストには、コマンドの文字列と実行されるスクリプトをペアで指定します。まず、引用符付きコマンド文字列を指定し、その後にそのコマンドを選択したときに評価される式を指定します。コマンドが空の場合は、区切り線が挿入されます。

メモ

Altキーを押しながら赤い三角ボタンのメニューを右クリックすると、コマンドのチェックボックスがあるウィンドウが表示されます。

Radio Box({*item*, ...}, <*script*>)

説明

一連のラジオボタンを表示するディスプレイボックスを作成する。項目は引用符付き文字列です。ラジオボタンが選択されるたびに、オプションのスクリプトが実行されます。

Range Slider Box(*minValue*, *maxValue*, *lowVariable*, *highVariable*, *script*)

説明

Range Slider Box()は、最小値（*minValue*）から最大値（*maxValue*）までの範囲スライダを表示したディスプレイボックスを戻す。2つのスライダの位置が変化すると、その値が*lowVariable*と*highVariable*に代入され、スクリプトが実行されます。

戻り値

ディスプレイボックス（範囲スライダボックス）

引数

minValue, *maxValue* スライダの最小値および最大値となる数値。

lowVariable 最小値側のスライダによって設定・変更される変数。

highVariable 最大値側のスライダによって設定・変更される変数。

script スライダボックスの移動に伴って実行される任意の有効なJSLコマンド。

Report(*obj*)

説明

プラットフォームオブジェクト（*obj*）の表示ツリーを戻す。同じ処理は、次のように、メッセージとしてプラットフォームに送っても行えます。

```
obj<<Report
```

Scene Box(*x size*, *y size*)

説明

3Dグラフィック用のシーンボックスを作成する。ボックスの横幅は*x*、高さは*y*に設定される。

Scene Display List

説明

3次元グラフィックのためのディスプレイリストを戻す。

例

```
ex = Scene Display List();
ex << Color( .9, .9, .9 );
ex << Text( center, middle, .3, "Hello World" );
exScene = Scene Box( 600, 600 );
exScene << Background Color( 0 );
exScene << Show Arcball( always );
New Window( "Samples/Scripts/Scene3Dにある「HelloWorld.jsl」を参照", exScene );
exScene << Perspective( 45, .2, 20 );
exScene << Translate( 0.0, 0.0, -4.5 );
exScene << Arcball( ex, 1.5 );
exScene << Update;
```

Script Box(<*script*>, <*language*>, <*width*>, <*height*>)

説明

引用符付き文字列で指定されたスクリプト（*script*）を含む編集ボックスを作成する。このボックスはスクリプトウィンドウの一種で、JSLを編集・実行できます。

オプションの引数

script スクリプトボックスに表示する引用符付き文字列を指定する。

language 指定した言語に対応した構文の強調表示を可能にする引用符付き文字列。有効な言語は、"JSL"、"Text"、"SAS"、"SAS Output"、"SASLog"、"R"、"MATLAB"、"JavaScript"、"C"、"SQL"、"Python"、"JSON"、"XML"です。

width スクリプトボックスの幅を指定する整数。

height スクリプトボックスの高さを指定する整数。

例

```
// JSON
New Window( "JSON",
   Script Box(
      "{\!"a\!":1,\!"b\!":\!"test\!"}",
      "JSON"
   )
);
```

Scroll Box(<Size(*h*,*v*)>, *display box*, ...)

説明

スクロールバーを使って表示する、より大きな子ボックスを配置したディスプレイボックスを作成する。

戻り値

スクロールボックスオブジェクトへの参照

必須の引数

display box スクロールボックスには任意の数のディスプレイボックスの引数を設定できる。

オプションの引数

Size(*h*,*v*) *h*引数および *v*引数で、ボックスのサイズをピクセル単位で指定する。

メモ

スクロールボックスオブジェクトに次のメッセージを送ることによって、背景色を設定できます。

```
<<Set Background Color( {R, G, B} | <color> )
```

Flexible引数は廃止されています。代わりに、Set Stretchメッセージを使用してください。例については、「V Scroll Box(<Size(v)>, display box)」を参照してください。

横方向（h）と縦方向（v）のスクロールについてブール値のフラグを設定し、スクロールバーを有効（1）または無効（0）にすることができます。所定の方向のスクロールが無効な場合、スクロールボックスはその方向については標準のコンテナ同様の働きをします。

```
<<Set Scrollers (h, v)
```

スクロールに関する現在のこれらのフラグを得るには、次のメッセージを使用します。

```
<<Get Scrollers
```

スクロールバーのスクローラーの横方向（h）と縦方向（v）の位置（ピクセル単位）を設定するには、次のメッセージを使用します。

```
<<Set Scroll Position (h,v)
```

現在のこれらのスクロール位置を得るには、次のメッセージを使用します。

```
<<Get Scroll Position
```

横方向と縦方向のスクロール範囲に関する現在の上限位置を得るには、次のメッセージを使用します。

```
<<Get Scroll Extents
```

例

次の例は、所定の設定のスクロールボックスを含むウィンドウを表示します。

```
win = New Window( "例",
   sb = Scroll Box(
      Size( 150, 75 ),
      List Box(
         {"First Item", "Second Item",
         "Third Item", "Fourth Item",
         "Fifth Item"},
         width( 200 ),
         max selected( 2 ),
         nLines( 6 )
      )
   )
);
win << Set Window Size( 300, 200 );
sb << Set Scrollers( 1, 1 ); // 両方向のスクロールバーを有効化
// スクロールバー上のスクローラーの位置を指定
sb << Set Scroll Position( 0, 20 );
```

Shape Seg({Path(*path*), ...}, <Row States(*dt*|*dt*,[*rows*]|dt,{{*rows*}, ...}|{*row states*})>)

説明

さまざまな形状を持つディスプレイセグメントを戻す。

必須の引数

Path Nx3行列または文字列でパスを指定する。行列の場合は、x座標、y座標、およびパス内の各点のフラグの3列の構成で指定する。フラグの値には、0（コントロール点）、1（移動）、2（線分）、3（3次ベジエ曲線）または負の値（点がパスの終点でもある場合）を指定できます。パスを文字列で指定する場合は、SVG構文を用います。

オプションの引数

Row States データテーブル参照、およびオプションで行または現在の行の属性を指定する。

例

```
win = New Window( "Shape Segの例",
   Graph Box(
      Shape Seg(
         {Path( [10 10 1, 10 70 0, 70 70 0, 70 10 -3] ),
         Path( "M20,20 C20,60 60,60 60,20 Z" )},
         Row States( {Selected State( 1 ), Color State( "red" )} )
      )
   )
);
```

Sheet Box(<<Hold(*rpt*), *display box*, ...)

説明

他のディスプレイボックスを縦または横に配置したディスプレイボックスを戻す。

Sheet Panel Box(*title*, *display box*)

説明

シートボックスを縦にするか、横にするかを指定する。

Slider Box(*minValue*, *maxValue*, *variable*, *script*, <Set Width(*n*)>, <Rescale Slider(*min*, *max*)>)

説明

インタラクティブなスライダコントロールを作成する。

戻り値

ディスプレイボックス（スライダボックス）

必須の引数

minValue, *maxValue* スライダの最小値および最大値となる数値。

variable スライダボックスによって設定・変更される変数。

script スライダボックスの移動に伴って実行される有効なJSLコマンド。

オプションの引数

Set Width(*n*) スライダボックスの幅をピクセルで指定する。

Rescale Slider(*minValue*, *maxValue*) スライダボックスの最大値と最小値をリセットする。

メモ

Set WidthとRescale Sliderをメッセージとしてスライダオブジェクトに送ることができます。

例:

```
ex = .6;
New Window( "例", mybox = Slider Box( 0, 1, ex, Show( ex ) ) );
mybox << Set Width( 200 ) << Rescale Slider( 0, 5 );
```

Spacer Box(<Size(*h*, *v*)>, <Color(*color*)>)

説明

他のディスプレイボックスとの間のスペースを維持するため、または、LineUp Box内のセルを埋めるために使用されるディスプレイボックスを作成する。

戻り値

ディスプレイボックスへの参照

オプションの引数

Size(*h*, *v*) *h*引数および*v*引数で、ボックスのサイズをピクセル単位で指定する。

Color(*color*) ボックスの色を、引数で指定されたJSLカラーに設定する。

Spin Box(*script*)

説明

上向き/下向きの矢印ボタンを持つディスプレイボックスを戻す。

引数

script クリックされた矢印の向きを示す引数と共に呼び出される。負の値は下向き、正の値は上向きを示します。1の値は1回のクリックを示し、2以上の値はクリックの反復を示します。

例

```
win = New Window( "例",
   Lineup Box(
      2,
      nb = Number Edit Box( 3 ),
      sb = Spin Box( Function( {value}, nb << Increment( value ) ) )
   )
);
nb << Set Increment( 1 );
```

Splitter Box(<Size(*x*, *y*)>, *display box*, ...)

説明

他のディスプレイボックスを横または縦に配列できるディスプレイボックスと、そのサイズをインタラクティブに調整するためのコントロールを戻す。子のサイズは、分割線ボックス（splitter box）の幅または高さの割合で指定します。オプションの*Size*引数は、最上位の分割線ボックスに対する指定で、子ボックスはそれに対する割合での指定となります。

H Splitter Box()またはV Splitter Box()を使用します。

String Col Box(*title*, {*string*, ...})

説明

テーブルボックス（Table Box）に列を作成し、引用符付きの文字列（*string*）の項目を保存する。引用符付きの*title*が列名になります。

String Col Edit Box(*title*, {*string*, ...}, <Set Width(*n*)>)

説明

テーブルボックス（Table Box）に列を作成し、引用符付きの文字列（*string*）の項目を保存する。この関数によって作成された列の文字列は、編集できます。引用符付きの*title*が列名になります。

オプションの引数

Set Width(*n*) 列の幅をピクセルで指定する。

例

```
Names Default To Here( 1 );
New Window( "例",
   scb =
   String Col Edit Box(
      "文字列",
      {"The", "quick", "brown", "fox", "jumps",
      "over", "the", "lazy", "dog"},
   )
);
scb << SetWidth( 300 );
```

メモ

データを取得するには、次のメッセージを使用します。

```
data = obj << Get;
```

Tab Box(Tab Page Box(Title(*page title 1*), <*options*>, *contents of page 1*), Tab Page Box(Title(*page title 2*), <*options*>, *contents of page 2*), ...);

説明

（Tab List Boxから名称変更。）タブ付きのウィンドウペインを作成する。引数には、タブページの名前とタブページの内容を交互に同数指定します。ページのタイトルは引用符で囲む必要があります。

引数

Tab Page Box タブボックスの中で使用できる、またはタイトルが付いた単独のコンテナとして使用できるディスプレイボックスを戻す。

- Title(*page title #*)は、1ページのタイトルを指定します。
- オプションには、<<Closeable(*Boolean*)（ボックスを閉じることができるかどうかを指定する）、<<Tip(*string*)（ツールチップを指定する引用符付き文字列）、<<Icon(*string*)（アイコンを指定する引用符付き文字列）があります。
- *contents of page #* は、タブのテキストを指定する引用符付き文字列。

例

```
New Window( "例",
   Tab Box(
      t1 = Tab Page Box( Title( "alpha" ), Panel Box( "panel", Text Box( "text" ) ) )
   ),
      t2 = Tab Page Box( Title( "beta" ), Popup Box( {"x", ex = 1, "y", ex = 2} ) ),
   )
);
```

メモ

Tab Page Boxに送ることのできるメッセージの名称が、以下のように変更になりました。

- Set Titleの現在の名称はTitleです。
- Set Tipの現在の名称はTipです。

- Set Iconの現在の名称はIconです。
- Set Closeableの現在の名称はCloseableです。

Tab List Box(*title*, *tabExpr1*, ...)

「Tab Box(Tab Page Box(Title(page title 1), <options>, contents of page 1), Tab Page Box(Title(page title 2), <options>, contents of page 2), ...);」を参照してください。

Tab Page Box(Title(*string*), <*options*>, *contents*)

説明

タブボックスの中で使用できる、またはタイトルが付いた単独のコンテナとして使用できるディスプレイボックスを戻す。

必須の引数

Title タブのタイトルを指定する引用符付き文字列。

オプションのメッセージ

<<Tip(*string*) ツールチップを指定する引用符付き文字列。

<<Closeable(*Boolean*) ページを閉じることができるかどうかを指定する。

<<Icon(*string*) アイコンを指定する引用符付き文字列。

<<Set Font(font name, <size>, <"bold"|"italic"|"underline"|strikeout">, <angle> フォント属性を指定する。

<<Set Font Name("*font name*") フォントの名前を指定する。

<<Set Font Scale(f) フォントの倍率を指定する。倍率は、基本フォントとポイントサイズで決められたサイズに適用されます。

<<Set Font Size(n) フォントサイズをピクセル数で指定する。

<<Set Font Style("plain"|"bold") フォントスタイルを指定する。

メモ

Tab Page Boxに送ることのできるメッセージの名称が、以下のように変更になりました。

- Set Titleの現在の名称はTitleです。
- Set Tipの現在の名称はTipです。
- Set Iconの現在の名称はIconです。
- Set Closeableの現在の名称はCloseableです。

Table Box(*display box*, ...)

説明

指定のディスプレイボックス（*display box*）を列としたレポートテーブル（表）を作成する。

Text Box(*text*, <*arguments*>)

説明

引用符付き文字列で指定されたテキスト（*text*）を含んだボックスを作成する。

引数

<<Justify Text(*position*) 引用符付き文字列で指定された位置（左“left”、中央“center”、または右“right”）にテキストを整列する。

<<Set Wrap(*pixels*) テキストを折り返す位置を設定する。

Text Edit Box(*text*, <*arguments*>)

説明

引用符付き文字列（*text*）を含む編集ボックスを作成する。

引数

<<Password Style(*Boolean*) パスワード入力用に、テキストを入力すると、アスタリスクを表示する。

<<Set Script テキストが編集された後、指定のスクリプトを実行する。

<<Set Width(*pixels*) テキストを折り返す位置を設定する。

This Project()

説明

プロジェクトの中でJSLスクリプトが実行された場合に、そのプロジェクトを取得する。

例

次の例は、現在のプロジェクトのウィンドウタイトルを取得します。

```
project = New Project();
project << Save( "$DOCUMENTS/Test Project.jmpprj" );
project << Run Script(
   New Window( "Project Title",
      Text Box(This Project() << Get Window Title())
   );
);
```

Tree Box(<{*rootnodes*}>, <Size(*width*, *height*)>, <MultiSelect(*Boolean*)>)

説明

ツリーノードで構成される階層構造のツリーを表示するボックスを作成する。

引数

{*rootnodes*} ボックス内のTree Node()によって作成されるルートノードの名前を指定する。

Size(*width*, *height*) ボックスの幅と高さを指定する（単位はピクセル）。

MultiSelect(*Boolean*) ツリー内で複数のアイテムの選択を可能とする。

Tree Node(<*data*>)

説明

ツリーボックスディスプレイ内に表示するノードを作成する。Tree Nodeは、親と子の両方のノードに使用されます。

メモ

macOSの場合、子ノードを持つルートノードにCollapseメッセージを定義するSet Node Select Scriptを送ると、スクリプトが2回実行されます。Windowsの場合、スクリプトは実行されません。macOSで2回実行されるのは増分だけではありません。すべてのスクリプトが2回実行されます。ログへの出力、列の作成、何らかの削除など、すべての処理が2回行われます。

Triangulation(<*dt*>, X(*col1*, *col1*), <Y(*col*)>)

説明

与えられたデータ点に対してDelaunayの三角分割を行い、その結果のオブジェクトを戻す。データ点の座標に重複があった場合、それらのデータ点は一つにまとめられ、オプション指定のY値には平均が使われる。

例

```
tri = Triangulation(
   X( [0 0 1 1], [0 1 0 1] ),
   Y( [0 1 2 3] )
);
dt = Open( "$SAMPLE_DATA/Cities.jmp" );
tri = Triangulation( X( :X, :Y ), Y( :POP ) );
```

V Center Box(*display box...*)

引数で指定した子ディスプレイボックスを含むディスプレイボックスを戻す。子ディスプレイボックスを縦方向のスペースの中央に配置します。

V List Box(<Align("Center"|"Right")>, *display box*, ...)

説明

ディスプレイボックスを作成し、その中に別のディスプレイボックスを縦に並べて表示する。

引数

Align ディスプレイボックスの位置を、右揃え（right）または中央揃え（center）に指定する。デフォルトでは、中央揃えで配置されます。

display box リストボックスに含める任意の数のディスプレイボックスを指定する。

V Scroll Box(<Size(*v*)>, *display box*)

説明

中身がスクロールボックスより大きい場合に右側と下側にスクロールバーがあるディスプレイボックスを戻す。

引数

Size(*v*) スクロールバーの縦の長さ。

display box スクロールボックスには任意の数のディスプレイボックスの引数を設定できる。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
New Window( "stretchable",
   H Splitter Box(
      Size( 400, 200 ),
      Scroll Box(
         Size( 200, 200 ),
         dt <<Run Script( "Distribution" ),
         <<Set Stretch( "Window", "Window" )
      ),
      Scroll Box(
         Size( 200, 200 ),
         dt << Run Script( "二変量の関係(二変量) " ),
         <<Set Stretch( "Window", "Window" )
      ),
      <<Set Stretch( "Window", "Window" )
   )
);
```

メモ

flexible引数は廃止されています。代わりに、Set Stretchを使用してください。

V Sheet Box(<<Hold(*report*), *display box, ...*)

説明

引数に指定された複数のディスプレイボックスを、縦方向に配置したディスプレイボックスを戻す。<<Hold()メッセージは、抜粋元となるレポートをどのシートが保持するかを示す。

V Splitter Box(<Size(*h*,*v*)>, *display box...*, <*arguments*>)

説明

引数に指定された複数のディスプレイボックスを縦方向（パネル）に配置し、分割線で仕切ったディスプレイボックスを戻す。ユーザは分割線をドラッグしてパネルのサイズを変更できます。

引数

Size(*v*) スクロールバーの縦の長さ。

display box 分割線ボックスには任意の数のディスプレイボックスの引数を設定できる。

オプションの引数

オプションの引数の詳細については、「H Splitter Box(<Size(h, v)>, display box, ...)」を参照してください。

Web Browser Box(*url*)

説明

Webページを表示するディスプレイボックスを作成する。WindowsのInternet Explorerでのみ使用できます。

戻り値

Webブラウザボックスオブジェクトへの参照

引数

url 表示するWebページのURLを示す引用符付き文字列。

例

次の例では、分割線で仕切られたディスプレイボックスを作成し、左にWebブラウザボックスを、右にバブルプロットを配置します。

```
dt = Open( "$SAMPLE_DATA/PopAgeGroup.jmp" );
New Window( "例",
   H Splitter Box(
      Size( 800, 300 ),
      wb = Web Browser Box( "http://www.jmp.com" ),
      dt << Run Script( "バブルプロット 地域" )
   )
);
wb << Set Stretch( "Window", "Window" ); // 自動的に縦横のサイズを調整する
wb << Set Max Size( 10000, 10000 ); // 最大サイズをピクセルで指定
```

メモ

targetが“_blank”である<a href>は、新しいInternet Explorerウィンドウに Webページを開きます。targeが“_new”である<a href>は、アクティブなInternet Explorerタブに Webページを開きます。

Window(<*string*|*int*>)

戻り値

開いているウィンドウすべてへの参照のリスト、または指定されたウィンドウへの参照

引数

string 開いている特定のウィンドウの名前を示す引用符付き文字列。

int 開いている特定のウィンドウを示す番号。

メモ

引数が指定されない場合は、開いているすべてのウィンドウのリストを戻します。

引数で指定されたウィンドウ名または数値が存在しない場合、空のリストを戻します。

Wrap List Box(*display box*, ...)

説明

複数のディスプレイボックスを横に並べて表示するリストボックスを作成する（ただし、印刷時には折り返して表示する）。

引数

display box リストボックスに含める任意の数のディスプレイボックスを指定する。

式の関数

Arg(*expr*, *i*)

Arg Expr(*expr*, *i*)

説明

指定された式から *i* 番目の引数を戻す。

戻り値

式 *expr* 内の *i* 番目の引数

i 番目の引数がない、または指定されていない場合は、Empty() を戻します。

引数

expr 式。

i どの引数を戻すかを示す整数。

メモ

Arg Expr() は廃止されました。代わりに、Arg() を使用してください。

Eval Expr(*expr*)

説明

外側の式は未評価のまま残して、*expr* に含まれている内側の式だけをすべて評価する。

戻り値

expr 内のすべての式を評価した結果の式

引数

expr 有効な式。

Expr(*expr, i*)

説明

引数を評価しないまま戻す（式のクォート）。

戻り値

未評価の引数

*i*番目の引数がない、または指定されていない場合は、Empty()を戻します。

引数

expr 式。

i どの引数を戻すかを示す整数。

Extract Expr(*expr, pattern*)

説明

パターン（*pattern*）に一致する式（*expr*）を見つける。

戻り値

指定した*pattern*とマッチするパターン

引数

expr 式。

pattern 任意のパターン。

Head(*exprArg*)

Head Expr(*exprArg*)

説明

評価された式の一番先頭の関数名を引数を付けないで戻す。

メモ

Head Expr()は廃止されました。代わりに、Head()を使用してください。

Head Name(*expr*)

Head Name Expr(*expr*)

説明

式の一番先頭の関数名を引用符付き文字列として戻す。

メモ

Head Name Expr()は廃止されました。代わりに、Head Name()を使用してください。

N Arg(*exprArg*)

説明

評価後の式の引数の個数を戻す。

N Arg Expr(*exprArg*)

説明

式の先頭部にある引数の個数を戻す。

メモ

N Arg Expr()は廃止されました。代わりに、N Arg()を使用してください。

Name Expr(*x*)

説明

*x*に格納されている式を評価せずに戻す。

ファイル関数

Close(<*dt*|*query*>, <*noSave*|*Save*(*path*)>)

説明

データテーブル、クエリー、またはJSONファイルを閉じる。引数が指定されていない場合は、現在のファイルを閉じます。ファイルが変更されている場合は、自動的に保存します。従属するウィンドウ（データテーブルを元に作成されたレポートウィンドウなど）もすべて閉じられます。

戻り値

なし

引数

dt データテーブル、クエリー、またはJSONファイルへの参照。

noSave|Save(*path*)（オプション）ファイルを閉じる前に指定の引用付きパスに保存するか、または保存しないで閉じるかを指定する。

Close All(<Project(*title*|*index*|*display box*|*window*)>, *type*, <"Invisible"|"Private">, <NoSave|Save>)

説明

指定されたタイプ（*type*）のウィンドウをすべて閉じる。

必須の引数

type 閉じるウィンドウのタイプ名を示す名前付き引数。使用できるタイプは、Data Tables、Reports、Journalsです。

オプションの引数

Project(*title*|*index*|*display box*|*window*) 指定のプロジェクトウィンドウを閉じる。

"Invisible" 非表示（invisible）のデータテーブルをすべて閉じる。

"Private" プライベート（private）に指定したデータテーブルをすべて閉じる。

NoSave または Save 指定されたタイプのウィンドウを閉じる前に保存するか、保存せずに閉じる。

Close Database Connection(*db connection handle*)

説明

Create Database Connection で戻されたデータベース接続を閉じる。

Close Log(*Boolean*)

説明

ログウィンドウを閉じる。

Convert File Path(*path*, <"Absolute"|"Relative">, <"POSIX"|"Windows">, <Base(*path*)>, <*Search*>)

説明

引数に従ってファイルパスを変換する。

戻り値

変換したパス

必須の引数

path WindowsまたはPOSIXの引用符付きパス名。

オプションの引数

"Absolute"|"Relative" 戻り値のパス名を絶対パスにするか相対パスにするかを指定する。デフォルト値は絶対パス（Absolute）。

"POSIX"|"Windows" 戻り値のパス名をWindows形式にするかPOSIX形式にするかを指定する。デフォルト値はPOSIX。

Base(*path*) 基準となるパス名を指定する。相対パス（Relative）を指定している場合に便利。デフォルト値はデフォルトのディレクトリ。「Set Default Directory(path)」を参照してください。

Search 指定のディレクトリの中で指定の文字列を検索する。次の例では、JMPは$SAMPLE_DATA/と$SAMPLE_DATA/Time Series/の中でAir.jmpを探します。「Air.jmp」が両方のディレクトリにある場合は、最初のインスタンスが戻されます。検索オプションがない場合、Convert File Path()は、代わりにデフォルトのディレクトリを使用します。

例

```
Set File Search Path(
  {Convert File Path( "$SAMPLE_DATA/" ),
  Convert File Path( "$SAMPLE_DATA/Time Series/" )}
```

```
);
Show( Get File Search Path() );
Show( Convert File Path( "Air.jmp", Search ) );
   Get File Search Path() = {"/C:/Program Files/JMP/JMPPRO/18/Samples/Data/",
      "/C:/Program Files/JMP/JMPPRO/18/Samples/Data/Time Series/"};
   Convert File Path("Air.jmp", search) = "/C:/Program
      Files/JMP/JMPPRO/18/Samples/Data/Time Series/Air.jmp";
```

Copy Directory(*from path*, *to path*, <Recursive(*Boolean*)>)

説明

ファイルをあるディレクトリから別のディレクトリにコピーする。オプションで、サブディレクトリもコピーできます。引数 *to path* で指定された場所にディレクトリが作成されるので、コピーするディレクトリの名前を引数 *to path* に含めてはいけません。

戻り値

ディレクトリがコピーされた場合は1、そうでない場合は0

必須の引数

from path 新しいディレクトリにコピーするファイルが含まれているディレクトリを指定する。*from path* は引用符付き文字列。

to path ファイルのコピー先となる新しいディレクトリの作成場所のパスを指定する。*to path* は引用符付き文字列。

オプションの引数

Recursive(*Boolean*)> コピー元ディレクトリ（*from path*）のサブディレクトリ構造をコピー先（*to path*）にコピーするかどうかを指定する。

メモ

Copy Directory(*from path*, *to path*, *Boolean*) を用いることは、推奨されません。

Copy File(*from path*, *to path*)

説明

ファイルを、元のファイルと同名または別名の新しいファイルにコピーする。

戻り値

ファイルがコピーされた場合は1、そうでない場合は0

引数

from path 新しいファイルにコピーするファイルのパスとファイル名を指定する。*from path* は引用符付き文字列。

to path 新しいファイルのパスとファイル名を指定する。*from path* は引用符付き文字列。

メモ

Copy File() ですでに存在するファイルを置き換えることはできません。

Create Database Connection(*dataSourceName*|"Connect Dialog", <DriverPrompt(*Boolean*)>);

説明

指定したデータベースへの接続を確立するか、ユーザにログイン情報を入力するよう促す。

戻り値

データベース接続のハンドラ

必須の引数

dataSourceName データソース名やユーザ名などの情報を含む引用符付きのサーバー接続文字列。

オプションの引数

"Connect Dialog"［データソースの選択］ウィンドウを開く。ユーザはこのウィンドウでデータベースを選択する。

Driver Prompt(*Boolean*) 必要に応じてODBCドライバが接続情報の入力を促せるようにする。

例

データソース名、ユーザ名、およびパスワードを指定する例:

```
Create Database Connection( "dsn=Books;UID=johnsmith;password=Christmas" );
```

引用符付きの接続の文字列にはパスワードを含めず、ユーザに接続情報の入力を促すウィンドウを表示するようODBCドライバに要求する例:

```
Create Database Connection( "dsn=Books;UID=johnsmith", Driver Prompt( 1 ) );
```

"Connect Dialog"を指定してユーザによるデータソースの選択を可能とする例:

```
Create Database Connection( "Connect Dialog" );
```

Create Directory(*path*)

説明

path引数に指定した場所に新しいディレクトリを作成する。

戻り値

ディレクトリが作成された場合は1、そうでない場合は0

引数

path 新しいディレクトリの作成場所の引用符付きパスを指定する。

Creation Date(*path*)

説明

指定したファイルまたはディレクトリの作成日を戻す。

戻り値

作成日

引数

path 作成日を調べる対象の引用符付きディレクトリのパスまたはファイル名。

Delete Directory(*path*, <Allow Undo(*Boolean*)>)

説明

指定したディレクトリとその内容、およびサブディレクトリを削除する。

戻り値

ディレクトリが削除された場合は1、そうでない場合は0

引数

path 削除するディレクトリのパスとディレクトリ名。

Allow Undo(*Boolean*) ごみ箱に移動するなどの「元に戻す」操作を許可する。

Delete File(*path*, <Allow Undo(*Boolean*)>)

説明

指定したファイルを削除する。

戻り値

ファイルが削除された場合は1、そうでない場合は0

必須の引数

path 削除するファイルの引用符付きパスとファイル名を指定する。

オプションの引数

Allow Undo(*Boolean*) ごみ箱に移動するなどの「元に戻す」操作を許可する。

Directory Exists(*path*)

説明

指定したディレクトリが存在するかどうかを調べる。

戻り値

ディレクトリが存在するときは1、そうでない場合は0

引数

path 調べるディレクトリの引用符付きパスとディレクトリ名。

File Exists(*path*)

説明

指定したファイル名が指定したパスに存在するかどうかを調べる。

戻り値

ファイルが存在するときは1、そうでない場合は0

引数

path 検証するファイルの引用符付きパスとファイル名を指定する。

File Size(*path*)

説明

指定されたパス内のファイルのサイズを特定する。

引数

path 引用符付きパスとファイル名を指定する。

例

```
File Size( "$SAMPLE_DATA/Big Class.jmp" );
   13142
```

Files In Directory(*path*, <Recursive(*Boolean*)>)

説明

*path*で指定されたディレクトリ内のファイル名を、リストにして戻す。

戻り値

ファイル名のリスト。Recursive(*Boolean*)を指定しなかった場合は、ディレクトリ名がリストに含まれます。

必須の引数

path 有効な引用符付きパス名。

オプションの引数

Recursive(*Boolean*) このオプションを指定すると、パスにある全フォルダ、およびそれらのフォルダに含まれる全サブフォルダから、ファイルが検索される。

メモ

Files In Directory(*path*, "Recursive"|*Boolean*)は廃止されました。

Find All(<Project(*title*|*index*|*display box*|*window*)>, *type*, <"Invisible"|"Private">)

説明

現在開いている特定のタイプのリソースをすべて戻す。

必須の引数

type 閉じるウィンドウのタイプ名を示す名前付き引数。使用できるタイプは、Data Tables、Reports、Journalsです。

オプションの引数

Project(*title*|index|*display box*|*window*) 指定のプロジェクトウィンドウを閉じる。

"Invisible" 非表示（invisible）のデータテーブルをすべて閉じる。

"Private" プライベート（private）に指定したデータテーブルをすべて閉じる。

例

次の例は、開いているすべてのデータテーブルを戻します。

```
exdt1 = Open( "$SAMPLE_DATA/Big Class.jmp" );
exdt2 = Open( "$SAMPLE_DATA/Animals.jmp" );
windows = Find All( Data Tables );
For( i = 1, i <= N Items( windows ), i++,
   Write( Char( windows[i] << Get Window Title ) || "\!N" )
);
   Big Class
   Animals
```

Get Default Directory()

説明

ユーザのデフォルトのディレクトリを取得する。このパスが相対パスに使用されます。

Set Default Directory()関数を使ってデフォルトのディレクトリが設定されている場合、Get Default Directory()がそのSet Default Directory()関数と同じスクリプト内にある限り、その指定されたパスを戻します。

「Set Default Directory(path)」を参照してください。

戻り値

引用符付き文字列で示した絶対パス名

引数

なし

メモ

Get Default Directory()も、保存されたアクティブなスクリプトウィンドウのパスを取得します。

Get Excel Worksheets(*absolute path*)

説明

指定されたMicrosoft Excelワークブックに含まれるワークシートのリストを取得する。ワークシートがない場合は、空のリストを戻します。パスは引用符付き文字列。

メモ

この関数は、.xlsxおよびExcel 1997以降のワークブックに対応しています。

Get File Search Path()

説明

現在、ファイルを開く時に検索されるディレクトリの一覧をリストで戻す。

このリストは、Set File Search Path()関数を使って設定されます。「Set File Search Path(path|{list of paths})」を参照してください。

戻り値

引用符付き文字列で示したパス名のリスト

Get Path Variable(*name*)

説明

nameに指定されたパス変数の値を取得する。

戻り値

引用符付き文字列で示した絶対パス名

引数

name パス変数を示す引用符付き文字列。(例: SAMPLE_DATA および SAMPLE_SCRIPTS)

Google Sheet Export(*dt*, *email*, *spreadsheet URL*|*spreadsheet ID*|*new spreadsheet name*, *sheet name*)

説明

データテーブルをGoogleスプレッドシートに書き出す。

戻り値

書き出しが正常に完了した場合は「1」を戻す。

引数

dt データテーブル。

email 引用符で囲んだGmailアドレス。@gmail.com は不要です。

spreadsheet URL|*ID* 引用符で囲んだスプレッドシートのURLまたはID(先頭に「spreadsheets/d/」が付く文字列)。

new spreadsheet name 新しく作成するスプレッドシートの引用符付きの名前。

sheet name スプレッドシート内にあるシート(タブ)の引用符付きの名前。

メモ

- 計算式や「リストチェック」列プロパティなどのJMP特有の機能は、Googleスプレッドシートではサポートされません。
- 書き出したスプレッドシートが空白になってしまった場合は、JMPログにエラーメッセージがないかを確認してください。Windowsの場合は、**[表示] > [ログ]** を選択します。macOSの場合は、**[ウィンドウ] > [ログ]** を選択します。

次も参照

セキュリティ、各国の制限事項などについては、『JMPの使用法』を参照してください。

Google Sheet Import(*email*, *spreadsheet URL*|*spreadsheet ID*, <*sheet name*|{*sheet name*, *sheet name*}, *Google Sheet Settings*>)

説明

Googleスプレッドシートからデータを読み込む。

戻り値

データテーブル（一度に複数のスプレッドシートを読み込む場合は最初に読み込まれたデータテーブル）

必須の引数

email 引用符で囲んだGmailアドレス。@gmail.com は不要です。

spreadsheet URL|*ID* 引用符で囲んだスプレッドシートのURLまたはID（先頭に「spreadsheets/d/」が付きます）。

オプションの引数

sheet name 読み込むスプレッドシートの引用符付きの名前。

Google Sheet Settings データの読み込み方法の設定。

次も参照

セキュリティ、各国の制限事項などについては、『JMPの使用法』を参照してください。

Is Directory(*path*)

説明

引用符付きpath引数がディレクトリのときは1、そうでなければ0を戻す。

Is Directory Writable(*path*)

説明

引用符付きpath引数に指定されたディレクトリが書き込み可能な場合は1、そうでなければ0を戻す。

Is File(*path*)

説明

引用符付きpath引数がファイルのときは1、そうでなければ0を戻す。

Is File Writable(*path*)

説明

引用符付きpath引数に指定されたファイルが書き込み可能な場合は1、そうでなければ0を戻す。

JSON To Data Table(*JSON string*, (<Private(*Boolean*)>|<Invisible(*Boolean*)>), <Guess(Stack(*Boolean*)|"Tall"|"Wide"|"Huge"|"Pandas")>)

説明

引用符付きのJSON文字列をデータテーブルに変換する。

戻り値

データテーブルへの参照。空白の値や、"" という文字列、欠測値、その他の無効な値を解析すると、空のデータテーブルが戻されます。

必須の引数

JSON string 引用符付きのJSON文字列。

オプションの引数

Private(Boolean) データテーブルを完全に非表示にする。データテーブルでインタラクティブな操作を行う必要がない場合は、この引数を指定します。

Invisible(Boolean) データテーブルを非表示にする。ただし、JMPホームウィンドウには表示されます。

Guess(Stack(*Boolean*)|"Tall"|"Wide"|"Huge"|"Pandas") 積み重ね（stack）は、行を作成する親ノード内で反復されるノードに適用されます。デフォルトでは、1つのテーブルセルに値がカンマで区切って保存されます。1を指定した場合は、繰り返す値は別の行となり積み重ねられます。データの積み重ねは、慎重に行ってください。見た目で判別しにくいデータエラーにつながる可能性があります。

"Tall"は、データを縦長のデータテーブルに読み込みます。JSONファイルの行数が多い場合は、このオプションを選択してください。これはデフォルトの設定です。

"Wide"は、データを横長のデータテーブルに読み込みます。JSONファイルの列数が多い場合は、このオプションを選択してください。

"Huge"は、データを縦長かつ横長のデータテーブルに読み込みます。

"Pandas"は、データをPandas形式で読み込みます。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt = JSON to Data Table(
   "[ { \!"name\!": \!"KATIE\!", \!"age\!": 12, \!"sex\!": \!"F\!", \!"height\!":
   59, \!"weight\!": 95 }, { \!"name\!": \!"LOUISE\!", \!"age\!": 12, \!"sex\!":
   \!"F\!", \!"height\!": 61, \!"weight\!": 123 }, { \!"name\!": \!"JANE\!",
   \!"age\!": 12, \!"sex\!": \!"F\!", \!"height\!": 55, \!"weight\!": 74 } ]"
);
```

メモ

データを積み重ねると、見た目では判別しにくいデータエラーにつながる可能性があります。同じ行にあるはずの値が別の行にある、などです。以下に、データの積み重ねの例を示します。

```
d =
"\[
{
```

```
        "おもちゃ": [{
                "車輪": 2,
                "色": "赤"
        },
        {
                "車輪": 4,
                "サイズ": "大"
        }]
}
]\";

JSON to Data Table( d, Guess( Stack( 1 ) ) );
```

図2.1 積み重ねたデータのエラー例

	車輪	色	サイズ
1	2	赤	大
2	4		

最初のおもちゃは、赤で2つの車輪が付いているはずです。2つ目のおもちゃは、サイズが大で、車輪が4つのはずです。

JSON To List(*JSON string*)

説明

引用符で囲んだJSON形式の文字列を、JSLのリストに変換する。戻されたリストの構造は、JSONテキストの構造を反映したものとなります。空白の値や、"" という文字列、欠測値、その他の無効な値を解析すると、{}（空のリスト）が戻されます。

例

```
l = JSON To List(
   "[ { \!"name\!": \!"KATIE\!", \!"age\!": 12, \!"sex\!": \!"F\!", \!"height\!":
   59, \!"weight\!": 95 }, { \!"name\!": \!"LOUISE\!", \!"age\!": 12, \!"sex\!":
   \!"F\!", \!"height\!": 61, \!"weight\!": 123 }, { \!"name\!": \!"JANE\!",
   \!"age\!": 12, \!"sex\!": \!"F\!", \!"height\!": 55, \!"weight\!": 74 } ]"
);
Show( l );
```

JSON Literal(*JSON string*)

説明

指定したパラメータによって、有効なJSONブールまたはNULLである定数の値を戻す。空白の値や、"" という文字列、欠測値、その他の無効な値を解析すると、Empty()が戻されます。

Last Modification Date(*path*)

説明

パスの最終更新日を戻す。

戻り値

最終更新日

引数

path 引用符付きのディレクトリ名またはファイル名。

Load Text File(*path*, <Charset(character set), Force("Throw"|"Alert"|"Silent")>)>, <Line Separator(character)>, <"XMLParse">|<"SASODSXML">|<"JSON">|<BLOB(<arguments>)>)>)

説明

指定されたパス（*path*）にあるテキストファイルを、JSL変数に読み込む。

戻り値

引用符付き文字列

必須の引数

path テキストファイルを指す引用符付きのパス名。URLでも可。

オプションの引数

Charset("charater set") 文字セットを指定する。有効な文字セットには、"Best Guess"、"utf-8"、"utf-16"、"us-ascii"、"windows-1252"、"x-max-roman"、"x-mac-japanese"、"shift-jis"、"euc-jp"、"utf-16be"、"gb2312"があります。

Force("Throw"|"Alert"|"Silent") 文字セットが検出できない場合の処理を指定します。

Line Separator(*character*) 引用符で囲んだ行の区切り文字を指定します。たとえば、"\!n"（ラインフィード文字）や、"\!t"（タブ）を指定します。

"XMLParse" XMLファイルをJSLに変換する。

"SASODSXML" SASのODSにおけるデフォルトのXML形式として、テキストファイルを解釈する。

"JSON" JSONをツリー構造に変換する。

BLOB(<*arguments*>) ファイル内のデータを、引用符付き文字列ではなくBLOBとして戻す。ファイルの一部を読み込むには、以下の引数を使用できます。

- ReadOffsetFromBegin(*n*) ファイルの読み込み開始位置を、ファイル先頭からのオフセット値（ゼロベース）で指定する。
- ReadOffsetFromEnd(*n*) ファイルの読み込み開始位置を、ファイル末尾からのオフセット値（ゼロベース）で指定する。
- ReadLength(*n*) ファイルから読み込むバイト数を指定する（ファイルの先頭またはオフセット値から開始）。

– Base64Compressed(*Boolean*) BLOBを印字可能な形式に変換する方法を指定する。0を指定すると、JMPのASCII~HEX表記を使用します（デフォルトかつ推奨）。1を指定すると、BLOBをbase64形式で圧縮・変換して印字します。

Move Directory(*from path*, *to path*)

説明

ディレクトリとその内容（サブディレクトリも含む）を引用符付きパスから、別の引用符付きパスに移動する。

戻り値

ディレクトリが移動した場合は1、そうでない場合は0

引数

from path 移動元ディレクトリのパスとディレクトリ名。

to path 移動先ディレクトリのパスとディレクトリ名。

Move File(*from path*, *to path*)

説明

ファイルを引用符付きパスから、別の引用符付きパスに移動する。ファイル名は同一のままでも変更することもできます。

戻り値

ファイルが移動した場合は1、そうでない場合は0

引数

from path 移動するファイルの引用符付きパスとファイル名を指定する。

to path 移動先の引用符付きパスとファイル名を指定する。

メモ

Windowsでは、ファイルを存在しないフォルダに移動しようとすると、フォルダが作成され、1が戻されます。macOSでは、フォルダは作成されず、エラーが戻されます。

Open(*path*, <*arguments*>)

説明

*path*引数で指定されたファイルから作成されたデータテーブルやその他のJMPファイル、またはオブジェクトを開く。path引数が未指定の場合は、「開く」ウィンドウが表示されます。JSONまたはHDF5ファイルも開けます。具体的なファイルタイプに適用される引数の詳細については、JMPの［ヘルプ］メニューにある「スクリプトの索引」で例を参照してください。

必須の引数

path 開くファイルの引用符付きのパス。

オプションの引数

Add to Recent Files(*Boolean*) ホームウィンドウの「最近使ったファイル」リストにファイルを追加するかどうかを指定する。

Charset(*character set*) テキストファイルの読み込みに使用できる文字セットのオプションは、"Best Guess"、"utf-8"、"utf-16"、"us-ascii"、"windows-1252"、"x-max-roman"、"x-mac-japanese"、"shift-jis"、"euc-jp"、"utf-16be"、および"gb2312"。

Column Names Start(*n*)|Column Names are on Line(*n*) 読み込み対象のテキストファイル内で、列名が入力されている最初の行の番号を指定する。テキストファイルでセル間に改行が使われている場合、列名が複数行に分かれていることがあります。

Columns(*colName*=*colType*(*colWidth*),...) テキストファイル内の列のうち、データテーブルに読み込む対象の列を名前で指定する。

- colName: 読み込まれたデータの列の名前を指定する。
- colType("Character"|"Numeric"): 指定した列を文字タイプとするか、数値タイプとするかを指定する。
- colWidth(*n*): 指定した列の幅を整数で指定する。

Columns(<*arguments*>) ESRIシェープファイル（.shp）の場合、この引数とその設定は以下のようになります。

- Shape=numeric(*n*): 読み込み対象のESRIシェープファイル内で、シェープ番号が記載されている列の番号を指定する。
- Part=numeric(*n*): 読み込み対象のESRIシェープファイル内で、パーツ番号が記載されている列の番号を指定する。
- X=numeric(*n*): 読み込み対象のESRIシェープファイル内で、10進数表記の経度（±180°の範囲）が記載されている列の番号を指定する。
- Y=numeric(*n*): 読み込み対象のESRIシェープファイル内で、10進数表記の緯度（±90°の範囲）が記載されている列の番号を指定する。

"Column Names Only" 列名のリストを戻す。

Compress Allow List Check(*Boolean*) 読み込んだテキストファイルから作成したデータテーブルをJMPで圧縮するかどうかを指定する。

Compress Character Columns(*Boolean*) テキストファイルから読み込んだ文字タイプの列をJMPで圧縮するかどうかを指定する。

Compress Numeric Columns(*Boolean*) テキストファイルから読み込んだ数値タイプの列を圧縮するかどうかを指定する。

Concatenate Worksheets(*Boolean*) 読み込み対象の複数のExcelワークシートを1つのデータテーブルに連結するかどうかを指定する。

Create Concatenation Column(*Boolean*) 読み込み対象のExcelファイルの列を1列にまとめるかどうかを指定する。

Data Starts(*n*)|Data Starts on Line(*n*) 読み込み対象のテキストファイル内で、データが始まっている行の番号を指定する。

Debug JSL(*Boolean*) 指定されたJSLスクリプトを開かずに、デバッガ内に開く。

End Of Field("Tab"|"Space"|"Comma"|"Semicolon"|"Other"|"None") 読み込み対象のテキストファイルでフィールドの区切り文字として使用されている引用符付き文字を指定する。複数の文字を指定するには、各文字をカンマで区切ってください。"Other"を指定した場合は、EOF Other()引数で区切り文字を指定してください。

End Of Line("CRLF"|"CR"|"LF"|"Semicolon"|"Other") 読み込み対象のテキストファイルで行の区切り文字として使用されている引用符付き文字を指定する。複数の文字を指定するには、各文字をカンマで区切ってください。"Other"を指定した場合は、EOL Other()引数で区切り文字を指定してください。

EOF Other(*char*) 読み込み対象のテキストファイルで、End of Field()で指定できる文字とは異なるフィールド区切り文字が使用されている場合は、使用する文字をこの引数で指定する。

EOL Other(*char*) 読み込み対象のテキストファイルで、End of Line()で指定できる文字とは異なる行の区切り文字が使用されている場合は、使用する文字をこの引数で指定する。

"Excel Wizard" Microsoft Excelワークシートを、Excel読み込みウィザードで開く。この引数を指定しないと、ワークシートはそのままデータテーブルとして開かれます。

"text"|"journal"|"sas"|"script"|"png"|"jmp"（オプション）開くファイルの種類を指定する引用符付き文字列。ファイルに拡張子がない場合や、ファイルの拡張子とファイルの内容が一致しない場合、JSL BLOBを読み込む場合などに便利です。この文字列を指定しない場合は、ファイル拡張子のデフォルトのプログラムでファイルが開きます。

メモ：zipアーカイブの場合は、path引数を指定します。拡張子（.zip）は必要ありません。zipアーカイブに送るメッセージについては、「Zipアーカイブ」を参照してください。基本的な機能は、zipアーカイブ内のファイルのリストを取得すること、zipアーカイブ内のファイルを引用符付き文字列またはBLOBに読み込むこと、ファイルをzipアーカイブに書き込むことです。zipアーカイブを読み取ると、内容が一時的にメモリに格納されます。非常に大きなzipアーカイブを読み取ると、エラーが発生する場合があります。

First(*n*) データテーブルの先頭の*n*行を読み込む。

"Force Refresh" 指定されたJMPファイル（.jrn、.jsl、.jrp、または.jmpappsource）を保存せずに閉じ、ディスクから再び開きます。ファイルを最後に開いてから変更していた場合、その変更内容は削除されます。

HTML Table(*n*, <ColumnNames(*n*)>, <DataStarts(*n*)>) ファイルパスに指定されたURLのHTML Webページから、表（テーブル）を読み込む。オプションの引数*n*で、Webページ上の開きたい表の番号*n*を指定します。値を省略した場合、ページ上の最初の表だけが読み込まれます。オプションの引数ColumnNames(*n*)で、列名となる行を指定できます。オプションの引数DataStarts(*n*)では、データが始まる行を指定できます。

ヒント：読み込もうとしているテーブルにイメージが含まれている場合、それらは最初テキストとして読み込まれます。JMPデータテーブルにイメージをロードするには、自動的に生成されるLoad Picturesというテーブルスクリプトを実行します。すると、イメージを含む新しい式の列が作成されます。式列の詳細については、『JMPの使用法』を参照してください。

Ignore Columns(*col1, ...*) JMPデータテーブルまたはその他のJMPファイル内の列のうち、データテーブルには含めない列の引用符付きの名前を指定する。

"Invisible" ファイルを非表示で開く。この引用符付き引数は、データテーブル、JMPファイル、外部ファイル、テキストファイル、Excelファイル、SASファイル、ESRIシェープファイル、およびHTMLファイルにのみ適用されます。データテーブルは、「JMPホームウィンドウ」と［ウィンドウ］メニューにのみ表示されます。

Labels(*Boolean*) 読み込み対象のテキストファイルの最初の行にラベルや列見出しが含まれているかどうかを指定する。デフォルト値は1（真）。

Last(*n*) データテーブルの末尾の*n*行を読み込む。

Lines to Read(*n*) テキストファイル内の読み込み対象とする行の数を指定する。JMPでは、列名が読み取られた行の次の行から行数のカウントを開始します。

Number of Columns(*n*) 読み込み対象のテキストファイルに含まれている列の数を指定する。

Random(*n*) データテーブルから無作為に抽出した*n*行を読み込む。引数が0～1の場合、その数値は割合として使用されます。たとえばRandom(0.1)は、データの10パーセントを無作為に抽出して読み込みます。

メモ： 0以下の値を*n*に入力した場合は、テーブル全体が読み込まれます。

Run JSL(*Boolean*) 指定されたJSLファイルを、開かずに実行する。ブール値またはブール値を含む式を引数として指定します。スクリプトが、スクリプトを自動的に実行する //! で始まっている場合、そのスクリプトを開くにはブール値（0）を指定します。

Password(*password*) パスワードで保護されたSASファイルの引用符付きパスワードを指定する。ここで指定しておくと、手動で入力する必要がなくなります。パスワードは暗号化されません。（パスワードで保護されたMicrosoft Excelファイルは読み込むことができません。）

"PDF Wizard" PDFファイルをPDF読み込みウィザードで開く。このウィザードで、データの読み込み方法を指定できます。

"Private" データテーブルを非表示で開き、JMPホームウィンドウにも［ウィンドウ］メニューにもテーブル名を表示しない。ユーザが見る必要のない一時的なデータテーブルを作成したいときに使用してください。この引用符付き引数はデータテーブル、JMPファイル、外部ファイル、テキストファイル、Excelファイル、SASファイル、ESRIシェープファイル、またはHTMLファイルにのみ適用されます。プライベートのデータテーブルを作成すると、データへのアクセスが高速化しますが、データテーブルのデータを保持するために必要なメモリが少なくなるわけではありません。

Quarantine Action("Allow Scripts"|"Block Scripts"|"Do Not Open"|"Show Dialog") ダウンロードしたデータテーブルを開いたときにスクリプトが実行されるかどうかを指定する。オプションにより、ユーザーに対し、データテーブルを確認するか、そのまま開くかを指定してもらうウィンドウを表示することもできます。ユーザーが［確認する］をクリックした場合、スクリプトは実行されません。

Scan Whole File(*Boolean*) 列のデータタイプを判断するためにどれくらいJMPがテキストファイルをスキャンするかを指定する。値はブール値。デフォルトの値は真で、データの種類が特定できるまでファイル全体をスキャンします。大きいファイルを読み込むときは、値を偽に設定すると、ファイルが5秒間だけスキャンされます。

Select Columns(*col1, ...*) JMPデータテーブルまたはその他のJMPファイル内の列のうち、データテーブルに含める列の引用符付きの名前を指定する。

"Strip Quotes"|Strip Enclosing Quotes(*Boolean*) テキストファイル内のフィールドが引用符で囲まれているとき、この引数が真の場合は引用符を削除し、偽の場合は引用符を削除しない。デフォルト値は1（真）。

Table Contains Column Headers(*Boolean*) 読み込み対象のテキストファイルの最初の行にラベルや列見出しが含まれているかどうかを指定する。デフォルト値は1（真）。

"Text Wizard" テキストファイルを、テキスト読み込みプレビューで開く。このウィンドウで、読み込みオプションを選択できます。このオプションを指定しない場合は、JMPの環境設定の［テキストデータファイル］の設定に従って、データテーブルとして自動的に読み込まれます。

Treat Empty Columns as Numeric(*Boolean*) テキストファイル内の欠測値データの列を文字タイプではなく数値タイプとするかどうかを指定する。ピリオド、Unicodeのドット、NaN、引用符付き空白文字列は欠測値と判断されます。デフォルト値は0（偽）。

Use Apostrophe as Quotation Mark(*Boolean*) テキストファイルの読み込み時にアポストロフィを引用符として使用することを宣言する。テキストデータが標準的な形式ではなく、フィールドが引用符ではなくアポストロフィで囲まれている場合以外、このオプションは推奨されません。デフォルト値は0（偽）。

Use Labels for Var Names(*Boolean*) SASデータセットの場合、SASラベルをJMPの列名として使用するかどうかを指定する。デフォルト値は0（偽）。

Use for All Sheets(*Boolean*) Worksheetsの設定を、データテーブルとして開くExcelファイルのすべてのワークシートで使用するかどうかを指定する。

Worksheet Settings(*Boolean*, <*options*>) ExcelファイルのJMPデータテーブルへの読み込みオプションを指定する。次のいずれかのオプションを指定できます。

- Has Column Headers(*Boolean*): Excelファイルの第1行目が列見出しであることを指定する。
- Number of Rows in Headers(*n*): Excelファイル内で列見出しとして使用されている行の数を指定する。
- Headers Start on Row(*n*): Excelファイル内で列見出しが入力されている最初の行の番号を指定する。デフォルト値は1です。
- Data Starts on Row(*n*): Excelファイル内でデータが入力されている最初の行の番号を指定する。
- Data Starts on Column(*n*): Excelファイル内でデータが入力されている最初の列の番号を指定する。
- Data Ends on Row(*n*): Excelファイル内でデータが入力されている最後の行の番号を指定する。
- Data Ends on Column(*n*): Excelファイル内でデータが入力されている最後の列の番号を指定する。
- Replicated Spanned Rows(*Boolean*): Excelファイル内にあるセルが結合された行のデータを複製して読み込むかどうかを指定する。
- Suppress Hidden Rows(*Boolean*): Excelファイルで非表示になっている行をJMPに読み込まないかどうかを指定する。

– Suppress Hidden Columns(*Boolean*): Excelファイルで非表示になっている列をJMPに読み込まないかどうかを指定する。

– Treat as Hierarchy(*Boolean*): JMPにExcelファイルを読み込むときに、複数の列見出しを階層として扱うかどうかを指定する。真の場合、Excelファイルは、複数の見出し行が名が階層構造になった状態で読み込まれます。

Worksheets("Sheet Name"|{"Sheet Name", "Sheet Name",...}|*n*) 名前が指定されたワークシート、名前リスト内のすべてのワークシート、またはインデックス番号が指定されたワークシートを開く。ワークシートが未指定の場合は、スプレッドシート内のすべてのワークシートが個別のデータテーブルとして開きます。

メモ: Worksheets引数を指定することで、Webサイトから.xlsワークシートを読み込むことができます。Open()を使ってWebサイトから.xlsxワークシートを読み込むことはできません。

"XML Wizard" XMLファイルをXML読み込みウィザードで開く。この引数を指定しないと、XMLファイルはデータテーブルとして開かれます。

Year Rule | Two Digit Year Rule ("1900-1999"|"1910-2009"|"1920-2019"|"1930-2029"|"1940-2039"|"1950-2049"|"1960-2059"|"1970-2069"|"1980-2079"|"1990-2089"|"2000-2099") 読み込み対象のテキストファイルで使用されている2桁の年の範囲を指定する。たとえば、一番古い日付が1979年の場合は、"1970-2069"と指定します。

Open Database(dataSourceName|"Connect Dialog", "SELECT ..."|"SQLFILE..."|tableName, <"Invisible"|"Private">, <outputTableName>)

説明

SELECT、*SQLFILE*、*tableName*などの引数を伴う*dataSourceName*で指定されたデータベースを開く。

戻り値

*outputTableName*で指定した名前のデータテーブル

必須の引数

dataSourceName|"Connect Dialog" *dataSourceName*は、データソースの名前を指定する。"Connect Dialog"は、データソースを選択するためのウィンドウを開きます。

オプションの引数

"Invisible" 非表示のデータテーブルを作成する。データテーブルは非表示になりますが、「JMPホームウィンドウ」や［ウィンドウ］メニューにはリストされます。

"Private" データテーブルを完全に非表示にする。プライベートのデータテーブルを作成すると、データへのアクセスが高速化しますが、データテーブルのデータを保持するために必要なメモリが少なくなるわけではありません。

outputTableName 読み込まれたJMPデータテーブルの名前。

例

```
Open Database(
   "DSN=dBASE Files;DBQ=C:\Program Files\JMP\JMPPRO\18\Samples\Import Data\;",
   // SQLステートメント
```

```
    "SELECT HEIGHT, WEIGHT FROM Bigclass", // 列の選択
    "hw" // 出力データテーブルの名前
);
```

Parse JSON(JSON string)

説明

JSON文字列を、その構造に従って連想配列またはリストに変換する。解析されたJSONオブジェクトに2つ以上のメンバーがある場合は、結果をリストに変換します。空白の値や、"" という引用符付き文字列、欠測値、その他の無効な値を解析すると、Empty()が戻されます。

例

次の例は、JSONをリストに変換します。

```
j = Parse JSON(
   "[ { \!"name\!": \!"KATIE\!", \!"age\!": 12, \!"sex\!": \!"F\!", \!"height\!":
   59, \!"weight\!": 95 }, { \!"name\!": \!"LOUISE\!", \!"age\!": 12, \!"sex\!":
   \!"F\!", \!"height\!": 61, \!"weight\!": 123 }, { \!"name\!": \!"JANE\!",
   \!"age\!": 12, \!"sex\!": \!"F\!", \!"height\!": 55, \!"weight\!": 74 } ]"
);
Show( j );
   j = {["age" => 12, "height" => 59, "name" => "KATIE", "sex" => "F", "weight" =>
      95], ["age" => 12, "height" => 61, "name" => "LOUISE", "sex" => "F", "weight"
      => 123], ["age" => 12, "height" => 55, "name" => "JANE", "sex" => "F", "weight"
      => 74]};
```

Pick Directory(<"Prompt">, <*path*>, <Show Files(*Boolean*)>)

説明

ユーザにディレクトリの選択を促し、ディレクトリパスを引用符付き文字列で戻す。

戻り値

ユーザが選択したディレクトリのパス

オプションの引数

"Prompt" 引用符付き文字列は「参照」ウィンドウのタイトル部分（Windows）またはFinderウィンドウ（macOS）に表示されます。

path ディレクトリの選択ウィンドウで最初に開く場所を、引用符付き文字列で指定する。

Show Files(*Boolean*) 1はディレクトリの選択ウィンドウにファイルを表示し、0はファイルを非表示にする。デフォルト値は0。

Pick File(<"Prompt">, <*initial directory*>, <{*filter list*}>, <*first filter*>, <Save Flag(*Boolean*)>, <default file>), <"Multiple">)

説明

「開く」ウィンドウを表示し、ユーザにファイルの選択を促す。

戻り値

ユーザが選択したファイルのパス

オプションの引数

"Prompt" 引用符付き文字列。指定した文字列は、Windows上の「開く」ウィンドウの最上部に表示されます。

initial directory フォルダへの有効なファイルパスである引用符付き文字列。指定したパスは、ファイルを開くウィンドウの開始ディレクトリとして使用されます。これを指定しない場合、または空の文字列を指定した場合は、JMPのデフォルトのディレクトリが使用されます。

filter list ファイルを開くウィンドウに表示するファイルの種類のリストである引用符付き文字列。構文については、次の例を参照してください。

first filter フィルタリスト内のどのフィルタを最初に使用するかを示す整数。大きすぎる整数（項目数が3のリストに対して4など）または小さすぎる整数を指定した場合、リスト内の最初のフィルタが使用されます。

Save Flag(*Boolean*)「開く」ウィンドウと「名前を付けて保存」ウィンドウのどちらを使用するかを指定するブール値。0を指定した場合、ユーザはJMPで開くファイルを選択できます。1を指定した場合、ユーザは、選択した種類の新しい空のファイルを、選択したフォルダに保存できます。デフォルト値は0です。

default file デフォルトでウィンドウに表示されるファイルの引用符付きの名前。

"Multiple" Save Flagが0の場合に、ユーザが複数のファイルを選択できるようにする引用符付きの名前。

メモ

すべての引数はオプションですが、位置に依存します。たとえば、caption（キャプション）と、initial directory（最初のディレクトリ）の両方を指定しないと、filter list（フィルタリスト）を指定することはできません。

コンピュータの物理メモリ内のバッファサイズは、ユーザが開くことのできるファイルの数に影響します。

例

次のスクリプトは、ウィンドウのタイトルを**「JMPファイルの選択」**としたファイルを開くウィンドウにおいて、JMPの「Samples/Data」ディレクトリを開き、ファイルの種類のリストに**「JMPファイル」**と**「すべてのファイル」**を含め、そのうち**「JMPファイル」**が選択された状態とし、ファイル名のフィールドに「Hollywood Movies.jmp」を表示します。

```
Pick File(
   "JMPファイルの選択",
   "$SAMPLE_DATA",
   {"JMP Files|jmp;jsl;jrn", "All Files|*"},
   1,
   0,
   "Hollywood Movies.jmp"
);
```

Rename Directory(*old path name*, *new directory name*)

説明

ディレクトリを移動またはコピーせずに名前を変更する。

戻り値

ディレクトリの名前を変更した場合は1、そうでない場合は0

引数

old path name 変更前のディレクトリのパスと名前を指定する引用符付き文字列。

new directory name 新しいディレクトリ名を指定する引用符付き文字列。

メモ

*new directory name*を指定するときは、フルパスではなくディレクトリ名だけを含めます。

Rename File(*old path name*, *new file name*)

説明

ファイルを移動またはコピーせずに名前を変更する。

戻り値

ファイル名を変更した場合は1、そうでない場合は0

引数

old path name 変更前のファイルのパスと名前を指定する。

new file name 新しいファイル名を指定する。

メモ

*new file name*を指定するときは、フルパスではなくファイル名だけを含めます。

Save Text File(*path*, *text*|BLOB, <Mode("Replace"|"Append")>)

説明

作成されたファイルのパス名を戻す。*text*は引用符付き文字列。

Set Default Directory(*path*)

説明

デフォルトのディレクトリを設定する。以降、このディレクトリが相対パスの基準となります。

次も参照

「Get Default Directory()」

Set File Search Path(*path*|{*list of paths*})

説明

ファイルを開く時に検索されるディレクトリを設定する。パスとして{"."}を設定すると、カレントディレクトリが使用されます。

例

```
Set File Search Path( {"C:\JMP\13\source", "C:\Program
  Files\JMP\JMPPRO\18\Samples"} );
```

次も参照

「Get File Search Path()」。

Set Path Variable(*name*, <*path*>)

説明

パス変数に、パスを格納する。

必須の引数

Name 変数の引用符付きの名前。

オプションの引数

path パス。

TripleS Import(*path*, <*arguments*>)

説明

指定されたTriple-S（SSS）形式のファイル（調査ファイル）を読み込む。SSS形式は、.xmlまたは.sssのいずれかと、.csv、.dat、または.ascのいずれかのファイルのペアで構成されます。ペアのファイルは拡張子以外同じ名前で、同じフォルダになければなりません。

必須の引数

path xmlまたはsssファイルのフルパスを指定する引用符付き文字列。

オプションの引数

"Invisible" テーブルを非表示にする引用符付きキーワード。データテーブルは、「JMPホームウィンドウ」と［ウィンドウ］メニューにのみ表示されます。なお、非表示のデータテーブルは、明示的に閉じるまでメモリ内にとどまるため、不要になったものは閉じるよう注意してください。非表示のテーブルを明示的に閉じるには、Close(*dt*)を実行します。ここで、*dt* はTripleS Import関数から戻されたデータテーブル参照の変数です。

Use Labels for Imported Column Names(*Boolean*) ラベル名を列見出しとして使用します。デフォルト値は1です。

例

```
dt = TripleS Import( "C:\Data\airlines.sss", "Invisible", Use Labels for Imported
  Column Names( 0 ) );
```

財務関数

Double Declining Balance(*cost*, *salvage*, *life*, *period*, <*factor*>)

説明

指定した期における減価償却費を戻します。この関数では、倍額定率法または他の償却率を使用します。

必須の引数

cost 初期費用。

salvage 減価償却終了後の価値。

life 寿命。減価償却の期間。

period 減価償却費を求めたい期。寿命と同じ単位で指定してください。

オプションの引数

factor 償却率を示す数値。デフォルト値は2です。

メモ

この関数は、ExcelのDDB関数に相当します。

Future Value(*rate*, *nper*, *pmt*, <*pv*>, <*Type(Boolean)*>)

説明

利率が一定な状況で、定期に定額支払をした場合の、投資の将来価値を戻します。

必須の引数

rate 利率。

nper 投資の期間。

pmt 定額支払における毎回の支払額。

オプションの引数

pv 現在価値。デフォルト値は0です。

Type(*Boolean*) 期末払いの場合は0、期首払いの場合は1に設定する。デフォルト値は0です。

メモ

この関数は、ExcelのFV関数に相当します。

Interest Payment(*rate*, *per*, *nper*, *pv*, <*fv*>, <*Type(Boolean)*>)

説明

利率が一定な状況で、定期に定額支払をした場合の、支払いにおける利子額を戻します。

必須の引数

rate 利率。

per 元金支払額を求める期。

nper 投資の期間。

pv 現在価値。

オプションの引数

fv 将来価値。デフォルト値は0です。

Type(Boolean) 期末払いの場合は0、期首払いの場合は1に設定する。デフォルト値は0です。

メモ

この関数は、ExcelのIPMT関数に相当します。

Interest Rate(*nper*, *pmt*, *pv*, <*fv*>, <*Type(Boolean)*>, <*guess*>)

説明

投資の1期あたりの利率を戻します。

必須の引数

nper 投資の期間。

pmt 定額支払における毎回の支払額。

pv 現在価値。

オプションの引数

fv 将来価値。デフォルト値は0です。

Type(Boolean) 期末払いの場合は0、期首払いの場合は1に設定する。デフォルト値は0です。

guess 予測される率。デフォルトは0.1（10%）。

メモ

この関数は、ExcelのRATE関数に相当します。

Internal Rate of Return(*values*, <*guess*>)

Internal Rate of Return(*guess*, *value1*, *value2*, ...)

説明

一連の定期的なキャッシュフローに対して、内部収益率を戻します。

引数

values 値を含んだベクトル（行列）。ただし、各値を、1つ1つの引数に指定する形式でも指定できます。

guess 予測されるおおよその結果を示す値。デフォルトの数は0.1（10%）です。値を指定する場合、この引数は必須です。個々の値を指定する場合は必須です。

メモ

この関数は、ExcelのIRR関数に相当します。

Modified Internal Rate of Return(*values*, *finance rate*, *reinvest rate*)

Modified Internal Rate of Return(*finance rate*, *reinvest rate*, *value1*, *value2*, ...)

説明

一連の定期的なキャッシュフローに対して、修正内部収益率を戻します。投資コストと、現金の再投資によって得た利子の両方が考慮されます。

引数

values 値を含んだベクトル（行列）。ただし、各値を、1つ1つの引数に指定する形式でも指定できます。

finance rate キャッシュフローの現金に対して支払う利率。

reinvest rate キャッシュフローの現金を再投資した場合に受け取る利率。

メモ

この関数は、ExcelのMIRR関数に相当します。

Net Present Value(*rate*, *values*)

Net Present Value(*rate*, *value1*, *value2*, ...)

説明

割引率および一連の将来の支払（負の値）と収入（正の値）を考慮して、投資の正味現在価値を戻します。

引数

rate 割引率。

values 値を含んだベクトル（行列）。ただし、各値を、1つ1つの引数に指定する形式でも指定できます。

メモ

この関数は、ExcelのNPV関数に相当します。

Number of Periods(*rate*, *pmt*, *pv*, <*fv*>, <*type(0|1)*>)

説明

利率が一定な状況で、定期に定額支払をした場合の、投資の期間を戻します。

必須の引数

rate 利率。

pmt 定額支払における毎回の支払額。

pv 現在価値。

オプションの引数

fv 将来価値。デフォルト値は0です。

type(0|1) 期末払いの場合は0、期首払いの場合は1に設定する。デフォルト値は0です。

メモ

この関数は、ExcelのNPER関数に相当します。

Payment(*rate*, *nper*, *pv*, <*fv*>, <*type(0|1)*>)

説明

利率が一定な状況で、定期に定額支払をした場合の、ローンの支払額を戻します。

必須の引数

rate 利率。

nper 投資の期間。

pv 現在価値。

オプションの引数

fv 将来価値。デフォルト値は0です。

type(0|1) 期末払いの場合は0、期首払いの場合は1に設定する。デフォルト値は0です。

メモ

この関数は、ExcelのPMT関数に相当します。

Present Value(*rate*, *nper*, *pmt*, <*fv*>, <*type(0|1)*>)

説明

投資の現在価値を戻します。

引数

rate 1期あたりの利率。

nper 投資の期間。

pmt 定額支払における毎回の支払額。

fv 将来価値。デフォルト値は0です。

type(0|1) 期末払いの場合は0、期首払いの場合は1に設定する。デフォルト値は0です。

メモ

この関数は、ExcelのPV関数に相当します。

Principal Payment(*rate*, *per*, *nper*, *pv*, <*fv*>, <*type(0|1)*>)

説明

利率が一定な状況で、定期に定額支払をした場合の、支払いにおける元金分を戻します。

必須の引数

rate 1期あたりの利率。

per 元金支払額を求める期。

nper 投資の期間。

pv 現在価値。

オプションの引数

fv 将来価値。デフォルト値は0です。

type(0|1)（オプション）期末払いの場合は0、期首払いの場合は1に設定する。デフォルト値は0です。

メモ

この関数は、ExcelのPPMT関数に相当します。

Straight Line Depreciation(*cost*, *salvage*, *life*)

説明

定額法によって、指定した期における減価償却額を戻します。

引数

cost 資産の初期費用。

salvage 減価償却終了後の価値。

life 寿命。減価償却の期間。

メモ

この関数は、ExcelのSLN関数に相当します。

Sum Of Years Digits Depreciation(*cost*, *salvage*, *life*, *per*)

説明

級数法によって、指定した期における減価償却額を戻します。

引数

cost 資産の初期費用。

salvage 減価償却終了後の価値。

life 寿命。減価償却の期間。

per 減価償却費を求めたい期。寿命と同じ単位で指定してください。

メモ

この関数は、ExcelのSYD関数に相当します。

グラフ関数

Add Color Theme({*name*, <*flags*>, {*color*}, <{*position*}>)

説明

マーカー、データテーブルの行、ツリーマップなどのコンポーネントに適用できる独自のカラーテーマを作成する。Preferences()の中にAdd Color Theme(...)を含めることにより、JMP環境設定にカラーテーマを追加できます。

戻り値

なし

必須の引数

name カラーテーマの引用符付きの名前。

color RGB値のリスト。これらの値により、カテゴリカルのカラーテーマ内のブロックと連続変数のカラーテーマ内のグラデーションが定義されます。RGB値の各リストは、それぞれ環境設定のカラーテーマウィンドウ内のスライダに対応しています。

position（オプション）各色の位置を示す0～1の数字のリスト。各位置は、それぞれ環境設定のカラーテーマウィンドウ内のスライダに対応しています。位置を指定しなかった場合、スライダは均等に配置されます。

オプションの引数

flags 連続変数またはカテゴリカル変数のカラーテーマリストおよびカラーのカテゴリのためのフラグ。

```
Continuous, <Continuous>, Sequential
Continuous, <Continuous>, Diverging
Continuous, <Continuous>, Chromatic
Categorical, <Continuous>, Sequential
Categorical, <Continuous>, Diverging
Categorical, Qualitative
Categorical, <Continuous>, Chromatic
```

デフォルトのJMPカラーテーマでは、順次（Sequential）は左から右または右から左へ徐々に色が変化します。発散（Diverging）は中央が薄めです。色彩（Chromatic）は明るい色のブロック（グラデーション）で構成されます。定性（Qualitative）以外のすべてのカテゴリは、連続変数とカテゴリカル変数の両方に使用できます。

このフラグを指定しなかった場合、色は連続変数の順次、およびカテゴリカルの順次カテゴリで表示されます。

例

次の例は、「青->紫」という名前の連続変数のカラーテーマを作成します。色は発散（Diverging）カテゴリです。RGB値は、4つのリストで指定されます。

```
Add Color Theme(
  {"青-> 紫", {"Continuous", "Diverging"}, {{0, 0, 255},
  {57, 108, 244}, "white", {128, 0, 100}}} );
```

メモ

定性（Qualitative）以外のすべてのカテゴリは、連続変数とカテゴリカル変数の両方に使用できます。たとえば、「寒色->暖色」の発散テーマは、連続変数とカテゴリカルの両方のテーマリストにあります。JMPで例を見るには、**［環境設定］ > ［グラフ］**を選択してください。

カラーテーマを削除するには、`Remove Color Theme()`を使用します。

次も参照

「`Remove Color Theme(name|{name, <flags>, {color, ...}, <{position, ...}>}>)`」

Arc(*x1*, *y1*, *x2*, *y2*, *startangle*, *endangle*)

説明

引数によって指定された長方形の中に内接する弧を描く。

戻り値

なし

引数

x1, *y1* 長方形の左上隅の座標。

x2, *y2* 長方形の右下隅の座標。

startangle, *endangle* 度単位の開始角度と終了角度で、*startangle*から*endangle*まで時計回りに円弧が描画されます。このとき、0度は時計の12時の位置とします。

Arrow(<*pixellength*>, {*x1*, *y1*}, {*x2*, *y2*})

説明

指定された点の間に、順に矢印を引いていく。オプションの第1引数は、矢じりの長さをピクセルで指定します。

戻り値

なし

必須の引数

{*x1*, *y1*}, {*x2*, *y2*} グラフ内の座標を示す2つの数値のリスト。

オプションの引数

pixellength 矢じりの長さをピクセルで指定する。

メモ

x座標が入った行列と、y座標が入った行列で指定することもできます。`Arrow(<pixellength>, [x1, x2], [y1, y2])`

Back Color(*name*)

説明

グラフの背景の色を設定する。

戻り値

なし

引数

name 引用符付き色名または色番号（たとえば、赤の場合は "red" または 3）。

Char To Path(*path*)

説明

パスの指定を引用符付きの文字形式から行列形式に変換する。

戻り値

行列

引数

path パス名を示す引用符付き文字列。

Circle({*x*, *y*}, *radius*|*PixelRadius*(*n*), <...>, <"Fill">)

説明

{x, y} を中心として、指定された半径の円を描く。

戻り値

なし

必須の引数

{*x*, *y*} グラフ内の座標を示す数値。

radius 円の半径。この半径は、縦軸の目盛りに基づくものです。縦軸のサイズを変更すると、円のサイズも変わります。

PixelRadius(*n*) 円の半径をピクセルで表した数値。このオプションで半径をした場合、縦軸のサイズを変更しても、円のサイズは**変わりません**。

オプションの引数

"Fill" 関数内に定義された円をすべて、現在 fill color で指定されている色で塗りつぶすよう指示する位置引数。"Fill" が省略された場合、円は塗りつぶされません。

メモ

中心点と半径は、任意の順序で指定できます。また、中心点と半径を複数指定すれば、1つの関数によって複数の円を作成できます。中心点を1つ、半径を複数指定した場合、結果は同心円になります。中心点を複数指定した場合、それまでの円がすべて描かれた後に、最後に指定された半径の円が追加されます。例として、次のような指定をしたとします。

```
Graph Box(circle({20, 30}, 5, {50, 50}, 15))
```

この場合、2つではなく、3つの円が描かれます。まず、(20, 30) を中心とした半径5の円、次に (50, 50) を中心とした半径5の円、最後に (50, 50) を中心とした半径15の円が描かれます。

Color To HLS(*color*)

説明

色（*color*）の引数（JMPの色番号を含む）をHLS値のリストに変換する。

戻り値

指定した色（*color*）の色調（H）、明度（L）、彩度（S）の成分を表したリスト。値の範囲は0～1です。

引数

color JMPの色番号。

例

ColorToHLS()の結果は、1つのリスト変数または3つのスカラー変数のリストに割り当てることができます。

```
hls = Color To HLS( 8 );
{h, l, s} = Color To HLS( 8 );
Show( hls, h, l, s );
   hls = {0.778005464480874, 0.509803921568627, 0.976};
   h = 0.778005464480874;
   l = 0.509803921568627;
   s = 0.976;
```

Color To RGB(*color*)

説明

色（*color*）の引数（JMPの色番号を含む）をRGB値のリストに変換する。

戻り値

指定した色（*color*）の赤（R）、緑（G）、青（B）の成分を表したリスト。値の範囲は0～1です。

引数

color JMPの色番号。

例

ColorToRGB()の結果は、1つのリスト変数または3つのスカラー変数のリストに割り当てることができます。

```
rgb = Color To RGB( 8 );
{r, g, b} = Color To RGB( 8 );
Show( rgb, r, g, b );
   rgb = {0.670588235294118, 0.0313725490196078, 0.988235294117647};
   r = 0.670588235294118;
   g = 0.0313725490196078;
   b = 0.988235294117647;
```

Contour(*xVector*, *yVector*, *zGridMatrix*, *zContour*, <*messages*>)

説明

指定のグリッド値で等高線を描く。

戻り値

なし

必須の引数

*xVector zGridMatrix*を表す*n*個の値。

*yVector zGridMatrix*を表す*m*個の値。

zGridMatrix 曲面を表す*n* x *m*行列。

zContour 等高線の値。

オプションのメッセージ

<<zColor(color, option) 等高線に使用する色。

<<"Fill"|"Fill Between"|"Fill Below"|"Fill Above" 等高線の塗りを表示するかどうかを指定する。

<<Transparency(vector) 塗りの透明度をベクトルで指定する。

Contour Function(*zExpr*, *xName*, *yName*, (*z*|*zMatrix*), < <<xGrid(*min*, *max*, *incr*)>, < <<yGrid(*min*, *max*, *incr*)>, < <<zColor(*color*, *option*)>, < <<zLabeled>, < <<"Filled">, < <<"FillBetween">, < <<"Ternary">, < <<Transparency(*number*|*matrix*))

説明

2変数に対する関数値の等高線図を描く。等高線の高さを指定する引数（*z*）は1つの数値でも行列でもかまいません。

戻り値

なし

引数

zExpr 任意の式（たとえば、Sine(y)+Cosine(x)）。

xName、*yName* 2変数の名前。

z|*zMatrix* z値またはz値の行列。

オプションのメッセージ

<<xGrid, <<yGrid グリッドの細かさを定義する名前付きオプション

<<zColor 色を定義する数値、もしくは行列。引数はスカラーでも行列でもかまいませんが、数値でなければなりません。

<<zLabeled 等高線にラベルをつける。

<<"Filled" 現在、zColorで指定されている色で、等高線以上の領域を塗りつぶす。

<<"FillBetween" 現在、zColorで指定されている色で、隣接する等高線間の領域を塗りつぶす。nz本の等高線が描かれている場合、このオプションは(nz-1)個の領域を塗りつぶします。<<Filledオプションでなく、このオプションを使用することをお勧めします。

<<"Ternary" 三角図内の三角座標系からはみ出る線を切り取る。

<<Transparency(*number*|*matrix*) 塗りの透明度を設定する。0～1の数値のベクトルで指定してください。等高線の各領域を塗りつぶすのに、指定された透明度が循環して使われます。このオプションは、<<FillBetweenオプションと組み合わせて使用する場合にのみ、使用してください。

Drag Line(*xMatrix*, *yMatrix*, <*dragScript*>, <*mouseUpScript*>)

説明

引数の行列（*xMatrix*、*yMatrix*）で与えられた座標上に、頂点がドラッグ可能な線分を描く。

戻り値

なし

必須の引数

xMatrix x座標の行列。

yMatrix y座標の行列。

オプションの引数

dragScript 任意の有効なJSLスクリプト。このスクリプトは、マウスをドラッグしたときに実行される。

mouseUpScript 任意の有効なJSLスクリプト。このスクリプトは、マウスボタンを放したときに実行される。

Drag Marker(*xMatrix*, *yMatrix*, <*dragScript*>, <*mouseUpScript*>)

説明

引数の行列（*xMatrix*、*yMatrix*）で与えられた座標上に、ドラッグ可能なマーカーを描く。

戻り値

なし

引数

xMatrix x座標の行列。

yMatrix y座標の行列。

dragScript 任意の有効なJSLスクリプト。このスクリプトは、マウスをドラッグしたときに実行される。

mouseUpScript 任意の有効なJSLスクリプト。このスクリプトは、マウスボタンを放したときに実行される。

Drag Polygon(*xMatrix*, *yMatrix*, <*dragScript*>, <*mouseUpScript*>)

説明

引数の行列（*xMatrix*、*yMatrix*）で与えられた座標上に、頂点がドラッグ可能な多角形を描く。

戻り値

なし

必須の引数

xMatrix *x*座標の行列。

yMatrix *y*座標の行列。

オプションの引数

dragScript 任意の有効なJSLスクリプト。このスクリプトは、マウスをドラッグしたときに実行される。

mouseUpScript 任意の有効なJSLスクリプト。このスクリプトは、マウスボタンを放したときに実行される。

Drag Rect(*xMatrix*, *yMatrix*, <*dragScript*>, <*mouseUpScript*>)

説明

引数の行列（*xMatrix*、*yMatrix*）で与えられた座標上に、頂点がドラッグ可能な塗りつぶされた長方形を描く。

戻り値

なし

必須の引数

xMatrix 2つの*x*座標の行列。

yMatrix 2つの*y*座標の行列。

オプションの引数

dragScript 任意の有効なJSLスクリプト。このスクリプトは、マウスをドラッグしたときに実行される。

mouseUpScript 任意の有効なJSLスクリプト。このスクリプトは、マウスボタンを放したときに実行される。

メモ

*xMatrix*と*yMatrix*には、値を2つずつ指定する必要があります。指定する座標ペアは、Rect()を描くための規則に従っていなければなりません。最初の点（*xMatrix*の最初の値と*yMatrix*の最初の値）は、長方形の左上の点となります。2番目の点（*xMatrix*の2番目の値と*yMatrix*の2番目の値）は、長方形の右下の点となります。

Drag Text(*xMatrix*, *yMatrix*, *string*, <*dragScript*>, <*mouseUpScript*>)

説明

引数の行列（*xMatrix*、*yMatrix*）で与えられた座標上に、文字列（*string*）を描く。または、リストが指定されている場合は、そのリストの項目を描く。

戻り値

なし

引数

xMatrix x座標の行列。

yMatrix y座標の行列。

string 引用符付き文字列。グラフに描画されるテキストとなる。

dragScript 任意の有効なJSLスクリプト。このスクリプトは、マウスをドラッグしたときに実行される。

mouseUpScript 任意の有効なJSLスクリプト。このスクリプトは、マウスボタンを放したときに実行される。

Fill Color(*name*|*color*|*rgbList*)

説明

領域を塗りつぶすときの色を設定する。

戻り値

なし

引数

name|*color*|*rgbList* 引用符付きの色の名前、色番号、またはRGB値のリスト。

Fill Pattern(*name*|*mask*|*image*)

説明

塗りつぶし領域のパターンを設定する。

引数

name|*mask*|*image* 引用符付きの色の名前、0～1の値の行列、またはイメージへのパス。

Get Color Theme Details(*name*)

説明

指定されたカラーテーマのスクリプトを戻す。

例

次の例は、JMP標準のカラーテーマのスクリプトを戻します。

```
Get Color Theme Details( "JMP Default" );
  {"JMP Default", 9221, {{213, 72, 87}, {57, 177, 67}, {64, 111, 223}...}}
```

Get Color Theme Names(<*kind*>)

説明

すべてのカラーテーマ名、または指定された種類のカラーテーマ名のリストを戻す。引数kindに指定できる種類には、"Continuous"（連続尺度）、"Categorical"（カテゴリカル）、"Sequential"（順序）、"Diverging"（発散）、"Qualitative"（定性）、"Chromatic"（色彩）があります。

例

次の例は、すべてのカラーテーマを戻します。

```
Get Color Theme Names();
```

{"緑 -> 黒 -> 赤", "緑 -> 白 -> 赤", "白 -> 黒"...}

次の例は、発散カテゴリのカラーテーマを戻します。

```
Get Color Theme Names( "diverging" );
```

{"緑 -> 黒 -> 赤", "緑 -> 白 -> 赤", "青 -> グレー -> 赤"...}

Gradient Function(*zExpr*, *xName*, *yName*, [*zLow*, *zHigh*], zColor([*colorLow*, *colorHigh*]), < <<xGrid(*min*, *max*, *incr*)>, < <<yGrid(*min*, *max*, *incr*)> < <<Transparency(*alpha*|*vector*))

説明

2変数に対する関数値を色で描いたグラフを作成する。グリッド上の長方形を、式の値に基づき、小さい値に対する色と大きい値に対する色を混合して塗りつぶす。

必須の引数

zExpr 任意の式（たとえば、Sine(y)+Cosine(x)）。

xName, *yName* 2変数の名前。

zLow, *zHigh* 最小値と最大値（2～10）の行列。

zColor([colorLow, colorHigh) 混合される2つの色を定義する（4は緑、6はオレンジ）。

オプションのメッセージ

<<xGrid, *<<yGrid* グラデーションを描く範囲をボックスとして指定する。

<<zColor 等高線の色。引数はスカラーでも行列でもかまいませんが、数値でなければなりません。

<<Transparency(*number*|*matrix*) 塗りの透明度。0～1の数値のベクトルで指定してください。等高線の各領域を塗りつぶすのに、指定された透明度が循環して使われます。

例

```
Gradient Function(Log(a * a + b * b),
a, b, [2 10],
zColor([4, 6]));
```

H Line(*x1*, *x2*, *y*)

H Line(*y*)

説明

*y*の位置に横線を描く。*x*座標の開始点および終了点（*x1*および*x2*）が指定された場合、*y*の位置に*x1*から*x2*まで線を引きます。引数*y*に行列を指定し、複数の線を引くこともできます。

H Size()

説明

グラフィックフレームの幅をピクセル単位で戻す。

Handle(*xPos*, *yPos*, *dragScript*, *mouseUpScript*)

説明

引数（*xPos*と*yPos*）で与えられた座標に、ドラッグ可能なマーカーを配置する。最初のスクリプト（*dragScript*）はドラッグしたときに、2番目のスクリプト（*mouseUpScript*）はマウスのボタンを放したときに実行されます。

Heat Color(*x*, < <<*theme*>)

Heat Color(*x*)

説明

指定されたカラーテーマ（*"theme"*; 彩色方法）において、*n*に対応する色の値を戻す。色の値は、JMPで定義されている色の番号で戻されます。

戻り値

JMPの色番号。整数

必須の引数

x 0～1までの数値。

オプションのメッセージ

theme セルプロットでサポートされている引用符付きカラーテーマ名。
デフォルト値は「青 -> グレー -> 赤」（Blue to Gray to Red）。

HLS Color(*h*, *l*, *s*)

HLS Color({*h*, *l*, *s*})

説明

指定された色調（h）、明度（l）、および彩度（s）の値をJMPの色番号に変換する。

戻り値

JMPの色番号。整数

引数

色調、明度、および彩度、またはそれら3つの値を含んだリスト。指定できる値の範囲は0～1。

In Path(*x*, *y*, (*pathMatrix*|*pathText*))

説明

*x*と*y*で指定された点が*path*内にあるかどうかを特定する。

戻り値

点（*x, y*）が指定されたパス内にあれば真（1）、そうでない場合は偽（0）を戻す。

引数

x and *y* 点の座標

pathMatrix|*pathText* パスを示す行列または引用符付き文字列

In Polygon(*x*, *y*, *xMatrix*, *yMatrix*)

In Polygon(*x*, *y*, *xyPolygon*)

説明

指定の点（*x, y*）が、ベクトル引数（*xMatrix*と*yMatrix*）で定義された多角形の内側にあるかどうかを示すブール値（1または0）を戻す。

*xMatrix, yMatrix*の代わりに、2列の行列（*xyPolygon*）を使用することもできます。また、*x*と*y*として、互いに対応したベクトルを指定することもでき、その場合、各（*x, y*）のペアが指定された多角形の中に入るかどうかに基づき、0と1から構成されるベクトルが戻されます。

Level Color(*i*)

Level Color(*i*, *n*)

Level Color(*i*, *n*, <*theme*>)

Level Color(*i*, <*theme*>)

説明

カテゴリカルデータに対してJMPのグラフで使われている色の番号を戻す。

戻り値

JMPの色番号。整数

必須の引数

i 1以上、かつ、*n*で指定されたカテゴリ数以下の整数。

n カテゴリ数。

オプションの引数

theme［列プロパティ］の「値の色」リストに表示されているカラーテーマのいずれか（引用符付き）。指定されていない場合、JMPのデフォルトのカラーテーマが適用されます。

メモ

第2引数が数値ではなく引用符付き文字列の場合、第2引数はカラーテーマとみなされます。

Line({*x1*, *y1*}, {*x2*, *y2*}, ...), <<ValueSpace(*Boolean*)

Line([*x1*, *x2*, ...], [*y1*, *y2*, ...]), <<ValueSpace(*Boolean*)

説明

点間に線を描く。

引数

{*x1*, *y1*}, {*x2*, *y2*}|[*x1*, *x2*, ...], [*y1*, *y2*, ...] 点の座標を示すペアを含むリスト、または *x* の行列と *y* の行列。

<<ValueSpace(*Boolean*) バブルプロットの軌跡のように、線がデータの変化を表している場合、その軌跡の線を描く。Boolean には定数または式を指定できます。

Line Style(*name*|*number*)

説明

グラフの線種を設定する。

引数

n 引用符付きの線種名または線種の番号

- 0／"Solid"（実線）
- 1／"Dotted"（点線）
- 2／"Dashed"（破線）
- 3／"DashDot"（一点鎖線）
- 4／"DashDotDot"（二点鎖線）

Marker(<*rowState*>, {*x1*, *y1*}, {*x2*, *y2*}, ...)

Marker(<*rowState*>, [*x1*, *x2*, ...], [*y1*, *y2*, ...})

説明

リストまたは行列で指定された座標にマーカーを描く。オプションの引数 *rowState* でマーカーの種類を設定できます。

Marker Size(*n*)

説明

マーカーのサイズを指定する。

Mousetrap(*dragScript*, *mouseUpscript*)

説明

クリックの座標を読み取り、グラフのプロパティを更新する。最初のスクリプト（*dragScript*）はドラッグしたときに、2番目のスクリプト（*mouseUpScript*）はマウスのボタンを放したときに実行されます。

New Heat Image(*matrix*, <*colorTheme*>)

説明

指定した行列とカラーテーマ、グラデーションに基づいてヒートマップのイメージを作成する。

引数

matrix 値の行列。指定した行列と同じ次元を持つヒートマップが作成されます。

colorTheme カラーテーマ、つまり値の色へのマッピングを定義するグラデーション属性一式。以下に、カスタムのグラデーションを使用する例を示します。

例

```
width = 256;
height = 256;
wstride = 16;
hstride = 16;
b = J( hstride, wstride, Random Normal() );
a = Transform Each( {z, {row, col}},
   J( height, width, 0 ),
   b[Floor( (row - 1) / hstride ) + 1,
   Floor( (col - 1) / wstride ) + 1]
);
E1 = New Heat Image(
   a,
   gradient(
      {Color Theme( "青->グレー->オレンジ" ),
      Scale Type( "標準偏差" ),
      Show Missing Color( "On" ),
      Reverse Gradient( 1 )}
   )
);
New Window( "test", Lineup Box( E1 ) );
```

Normal Contour(*prob*, *meanMatrix*, *stdMatrix*, *corrMatrix*, <*colorsMatrix*>, <*fill=x*>)

説明

*k*個の母集団の、2変量正規分布の等高線を描く。

必須の引数

prob 累積確率。スカラーまたは行列で指定。

meanMatrix k行2列の平均。行列で指定。

stdMatrix k行2列の標準偏差。行列で指定。

corrMatrix k行1列の相関係数。行列で指定。

オプションの引数

colorsMatrix k個の等高線の色を指定する。色は、JSLで用意されている標準色の整数値、またはRGB ColorやHLS Colorなどの色関数の戻り値で指定します。

fill=x 等高線の塗りの色の透明度を指定する。

Oval(*x1*, *y1*, *x2*, *y2*, <*Fill(Boolean)*>)

Oval({*x1*, *y1*}, {*x2*, *y2*}, <*Fill(Boolean)*>)

説明

対角線が（*x1, y1*）と（*x2, y2*）である長方形に内接する楕円を描く。*Fill*はブール値です。*Fill*が0の場合、楕円は塗りつぶされません。*Fill*が0以外の場合、現在fill colorで設定されている色で塗りつぶされます。*fill*のデフォルト値は0です。

Path((*pathMatrix*|*pathText*), <*Fill(Boolean)*>)

説明

指定されたパスに沿って線を描く。fillが指定されている場合は、パスが現在の塗りつぶし色で塗りつぶされます。

必須の引数

pathMatrix Nx3行列。*pathMatrix*を指定しない場合は、*pathText*を指定してください。

pathText SVG構文の引用符付き文字列。

オプションの引数

Fill(Boolean) 線を描く（0）かパスを塗りつぶすか（1）を指定する。デフォルト値は0です。

メモ

パスを行列で指定する場合は、x座標、y座標、および、パスの種類を表すフラグで構成します。フラグの値には、0（コントロール点）、1（移動）、2（線分）、3（3次ベジエ曲線）または負の値（パスによって閉じた領域を表す場合）を指定できます。

Path To Char(*path*)

説明

パスの指定を行列形式から引用符付きの文字形式に変換する。

戻り値

引用符付き文字列

引数

path Nx3のパスの行列。

メモ

パスを行列で指定する場合は、x座標、y座標、および、パスの種類を表すフラグで構成します。フラグの値には、0（コントロール点）、1（移動）、2（線分）、3（3次ベジエ曲線）または負の値（パスによって閉じた領域を表す場合）を指定できます。

Pen Color(*n*)

説明

線の色を指定する。

Pen Size(*n*)

説明

線の太さをピクセル値で指定する。

Pick Color(<*window title*>, <*name*|*index*|*RGBlist*>)

説明

カラーピッカーを作成する。カラーピッカーは、たとえば、ユーザーが色を選択して、グラフに適用したい場合に使えます。ユーザーは、オペレーティングシステムで用意されているカラーピッカーを使って事前に用意されている色を選択したり、独自の色を作成したりできます。スクリプトでデフォルトの色を指定しておくこともできます。デフォルトの色を指定しなかった場合、黒が選択されます。

戻り値

オペレーティングシステムのカラーピッカーからユーザーが選択した色

引数

window title カラーピッカーウィンドウのタイトルを引用符付きで指定する。

name デフォルトの色のJMP色名。

index デフォルトの色のJMP色番号。

RGBlist デフォルトの色のRGB値。

Pie(*left*, *top*, *right*, *bottom*, *startAngle*, *endAngle*)

説明

塗りつぶされた扇形を描く。2つの座標で表わされる長方形に内接する楕円形が仮に構成されます。そして、開始角度（*startangle*）と終了角度（*endangle*）で指定された扇形だけが実際に描画されます。

Pixel Line To(*h*, *v*)

説明

現在の位置から、ピクセル座標で指定された位置まで、幅が1ピクセルの線を引く。Pixel Origin()関数およびPixel Move To()関数を使って、現在のピクセル座標を設定します。

Pixel Move To(*h*, *v*)

説明

現在の位置を、ピクセル座標で指定された新しい位置に移動する。

Pixel Origin(x, *y*)

説明

後に続くPixel Line To()関数やPixel Move To()関数のために、グラフ座標の原点を設定する。

Polygon({*x1*, *y1*}, {*x2*, *y2*}, ...)

Polygon(*xMatrix*, *yMatrix*)

説明

座標のリストによって定義された多角形を描く。多角形は塗りつぶされます。

Polygon Area({*x1*, *y1*}, {*x2*, *y2*}, ...)

Polygon Area(*xMatrix*, *yMatrix*)

説明

指定した多角形の面積を計算する。

例

```
area = Polygon Area( {0, 0}, {0, 10}, {10, 10}, {10, 0} );
area = Polygon Area( [10 20 30], [10 30 20] );
```

Polygon Centroid({*x1*, *y1*}, {*x2*, *y2*}, ...)

Polygon Centroid(*xMatrix*, *yMatrix*)

説明

指定した多角形の重心を計算する。

例

```
{cx, cy} = Polygon Centroid( {0, 0}, {0, 10}, {10, 10}, {10, 0} );
centroid = Polygon Centroid( [10 20 30], [10 30 20] );
```

Pixel Path(*h*, *v*, (*path matrix*|*path text*), <fill=*Boolean*)>, <scale=*n*>, <orient={*n*, *n*})

説明

fill（塗りつぶし）が0の場合はピクセルで指定されたパスに沿って線を描き、0以外の場合はパスの内側を塗る。

必須の引数

h, *v* 横と縦の座標を指定する。

path matrix *h*座標、*v*座標、およびそれらの座標の各点に対応するフラグの3列の構成で指定する。フラグの値には、0（コントロール点）、1（移動）、2（線分）、3（3次ベジエ曲線）または負の値（点がパスの終点でもある場合）を指定できます。各フラグ（1点につき1つ）には、0、1、2、3（形状の終点の場合は負）の値を指定できます。

path matrixを指定しない場合は、path textを指定してください。

path text SVG構文をサポートする。パスは、オプションのパラメータ（fill、orient）に応じて、基点、方向および尺度（scale）が決められる。

オプションの引数

fill(*Boolean*) 0は形状を塗りつぶす。1は空白の形状を描く。

orient オブジェクトを回転させる。たとえば、向きを{Sqrt(2), Sqrt(2)}とした場合、45度の角度で形状が描かれます。

scale オブジェクトの尺度。たとえば、尺度が1の場合、形状は定義されたとおりの大きさで描かれます。尺度が2の場合は2倍の大きさ、0.5の場合は半分の大きさで描かれます。

Pixel Text(<properties>, {h, v}, text, ...)

説明

ピクセル位置{*h, v*}に移動し、引用符付きの*text*引数で指定されたテキストを描画する。

必須の引数

h 横方向のピクセル位置。

v 縦方向のピクセル位置。

text ピクセルの開始位置と終了位置に表示する引用符付き文字列。

オプションの引数

center justified テキストを中央揃えにする。

right justified テキストを右揃えにする。

erased テキストの背景の長方形領域を消去する。テキストを読みやすくするために長方形の内側部分が消去され、テキストは背景色の上に描画されます。

boxed テキストをボックスで囲む。

counterclockwise テキストを反時計回りに回転させる。

clockwise テキストを時計回りに回転させる。

Rect(*left*, *top*, *right*, *bottom*, <Fill(*Boolean*)>)

Rect({*left*, *top*}, {*right*, *bottom*}, <Fill(*Boolean*)>)

説明

対角線が（*left*, *top*）と（*right*, *bottom*）を結んだ直線である長方形を描く。*Fill*はブール値です。*Fill*が0の場合、長方形は塗りつぶされません。*Fill*が0以外の場合、Fill Color関数で設定されている色で塗りつぶされます。*fill*のデフォルト値は0です。

Remove Color Theme(name|{name, <*flags*>, {*color*, ...}, <{*position*, ...}>}>)

説明

グローバルリストから、名前またはフルカラーテーマオブジェクトとして指定されたカスタムカラーテーマを削除する。

必須の引数

name カラーテーマの引用符付きの名前。

color 色のRGB値。

オプションの引数

flags テーマが連続尺度のものかカテゴリカルなものかといったメタデータを表す数値。この数値を知るには、Get Color Theme Details()関数の引数に該当のカラーテーマ名を指定した時の戻り値を見てください。

position 0～1の数値。それぞれの色に、グラデーションのどこに位置するかを示す0～1の値が割り当てられます。

例

```
Remove Color Theme( {"Yellow Blue", 0, {{255, 255, 0}, {0, 0, 255}}, {0.0, 1.0}} );
```

RGB Color(*r*, *g*, b)

RGB Color({*r*, *g*, *b*})

説明

指定された赤（r）、緑（g）、および青（b）の値をJMPの色番号に変換する。

戻り値

JMPの色番号。整数

引数

赤、緑、および青、または3つのRGB値を含んだリスト。指定できる値の範囲は0～1。

Text(<*properties*>, ({*x*, *y*}|{*left*, *bottom*, *right*, *top*}), *text*)

説明

xy座標、軸の左、下、右、一番上のいずれか、指定された位置に、引用符付き文字列（*text*）を書き込む。

プロパティ（*properties*）には、Center Justified（中央揃え）、Right Justified（右揃え）、Erased（消去）、Boxed（囲み）、Clockwise（時計回り）、Counterclockwise（反時計回り）のいずれかを指定できます。位置、名前付き引数、および引用符付き文字列は、任意の順序で指定できます。位置と名前付き引数は、すべての文字列に適用されます。

Text Color(*n*)

説明

テキストの色を指定する。

Text Font(*fontName*, <*size*>, <bold italic underline strikeout>, <*angle*>)

説明

テキストのフォントを設定する。現在のフォント属性を取得するには、引数なしで使用します。Angleは、時計回りの角度。

フォント属性は引用符で囲みます。複数の属性を適用するには、"bold italic"のようにまとめて指定します。

Text Size(*n*)

説明

テキストのサイズをポイント数で設定する。

Transparency(*alpha*)

説明

現在の描画の透明度を0～1の範囲の*alpha*で指定する。0は透明（描画なし）、1は完全に不透明（デフォルト）です。

メモ

オペレーティングシステムの中には、透化をサポートしていないものもあります。

V Line(*x*, <*y1*, *y2*>)

説明

*x*の位置に縦線を描く。*y*座標の開始点および終了点（*y1*および*y2*）が指定された場合は、*x*の位置に*y1*から*y2*まで線を引きます。引数*x*に行列を指定し、複数の線を描くこともできます。

V Size()

説明

グラフフレームの縦のサイズをピクセル単位で戻す。

X Function(*zExpr*, *yName*, <Min(*min*), Max(*max*), Fill(*Boolean*), Inc(*upper bound*))

説明

*yName*の値をY軸上に沿って変化させた時の関数値をX座標にプロットする。

X Origin()

説明

グラフフレームの左端のx値を戻す。右端の*x*値は、XOrigin()+XRange()により得ることができます。

X Range()

説明

ディスプレイボックスの左端から右端までの距離を戻す。たとえば、XOrigin()+XRange()は右端を示します。

X Scale(*xMin*, *xMax*)

説明

水平方向のスケールの範囲を設定する。スケールの指定がない場合、デフォルト値の（0,100）が使われます。*xMin*のデフォルト値は0です。*xMax*のデフォルト値は100です。

XY Function(*x(t)*, *y(t)*, *t*, Min(*min*), Max(*max*), (Inc(*bound*)|Steps(*min*))

説明

式*x(t)*と式*y(t)*を組み合わせ、パラメータ*t*の指定範囲でx-y曲線を作成する。*t*が最小値（Min）と最大値（Max）で変化するたびに、*x*と*y*の式が現在の*t*の値を使って評価されます。

メモ: デフォルトの精度では曲線が滑らかでない場合は、*inc()*や*steps()*を指定する必要があります。

Y Function(*yExpr*, *xName*, <Min(*min*), Max(*max*), Fill(*Boolean*), Inc(*bound*)>)

説明

*xName*の値をX軸上に沿って変化させた時の関数値をY座標にプロットする。

Y Origin()

説明

グラフフレームの下端の*y*値を戻す。

Y Range()

説明

ディスプレイボックスの下端から上端までの距離を戻す。たとえば、YOrigin()+YRange()は上端を示します。

Y Scale(*yMin*, *yMax*)

説明

垂直方向のスケールの範囲を設定する。スケールの指定がない場合、デフォルト値の(0, 100)が使われます。

HTTP関数

Decode 64 Blob(*string*)

説明

引用符付き文字列をBase64エンコーディングでデコードする。

Encode 64 Blob(*string*)

説明

引用符付き文字列をBase64エンコーディングでエンコードする。

リスト関数

As List(*matrix*)

説明

行列をリストに変換する。行列に複数の行がある場合は、行ごとのリストを含むリストを戻します。

戻り値

リスト

引数

matrix 任意の行列。

Concat Items({*string1*, *string2*, ...}, <*delimiter*>})

説明

引用符付き文字列のリストを、1つの文字列に変換する。各文字列は、区切り文字で区切られます。区切り文字を指定しなかった場合は、スペースで区切られます。

戻り値

連結した引用符付き文字列

引数

string 任意の引用符付き文字列。

delimiter（オプション）各項目間に挿入される引用符付き文字列。*delimter*には、2文字以上指定することもできます。

例

```
str1 = "いち";
str2 = "に";
str3 = "さん";

comb = Concat Items({str1, str2, str3});
   "いち に さん"
comb = Concat Items({str1, str2, str3}, " : ");
   "いち : に : さん"
del = ",";
comb = Concat Items({str1, str2, str3}, del);
   "いち , に , さん"
```

Eval List({*list*})

説明

*list*内の式を評価する。

戻り値

評価後の式を含むリスト

引数

list 有効なJSL式のリスト。

Insert(*source*, *item*, <*position*>)

Insert(*source*, *key*, *value*)

説明

ソース（*source*）の指定箇所（*position*）に新しい項目（*item*）を挿入する。*position*の指定を省略したときは、*項目*は最後尾に挿入されます。

連想配列の場合、ソース（*source*）の連想配列にキー（*key*）を追加し、それに値（*value*）を割り当てる。*source*にすでに*key*がある場合は、新しい*value*で置き換えられます。

必須の引数

source 引用符付き文字列、リスト、ベクトル、式、または連想配列。

item|*key* *source*の中に挿入する任意の値。連想配列の場合、*source*内に*key*がない場合もあります。

*value key*に割り当てる値。

オプションの引数

position（オプション）*source*内の*item*の挿入位置を示す数値。

Insert Into(*source*, *item*, <*position*>)

Insert Into(*source*, *key*, *value*)

説明

Insert関数と同じだが、結果をソース（*source*）に格納する。*source*は、左辺値（L-value）でなければなりません。

引数

source 引用符付き文字列、リスト、ベクトル、ディスプレイボックス、式、または連想配列を含む変数。

item|*key* sourceの中に挿入する任意の値。連想配列の場合、*source*内に*key*がある場合もあります。

position（オプション）*source*内の*item*の挿入位置を示す数値。

*value key*に割り当てる値。

Is List(*x*)

説明

評価後の引数がリストのときは1、そうでなければ0を戻す。

Items(*string*, <*delimiter*>, <Include Boundary Delimiters(*Boolean*)>)

説明

引数*delimiter*のいずれかの1文字で区切られた（空白も含む）引用符付き文字列のリストを戻す。

引数

string 評価する引用符付き文字列。

Delimiter（オプション）区切りとして使用する文字。*delimiter*がない場合は、ASCII文字のスペースが区切り文字になります。*delimiter*を空白の引用符付き文字列とした場合、1文字1文字がそれぞれ個別の項目とみなされます。

Include Boundary Delimiters(Boolean)（オプション）文字列の始まりと区切り文字の間の空白の文字列を戻す。

例

```
Items( "http://www.jmp.com", ":/." );
   {"http", "", "", "www", "jmp", "com"}
Items(",toy,", ",");
   {"toy"}
Items(",toy,", ",", Include Boundary Delimiters( 1 ));
   {"", "toy", ""}
   /* 引用符付き文字列の始まりとカンマ（区切り文字）の間にテキストがないため、
```

空白の文字列が戻される。文字列の終わりにある区切り文字も同様。/*

```
List(a, b, c, ...)
{a, b, c, ...}
```

説明

一連の項目を持つリストを作成する。

```
N Items(source)
```

説明

指定されたソース（*source*）内の要素数を数える。

戻り値

リストまたはディスプレイボックスの場合は、リストまたはディスプレイボックスの中の項目の数。連想配列の場合は、キーの数。行列の場合は、行列の要素の数。名前空間の場合は、名前空間内の関数および変数の数。クラスオブジェクトの場合は、メソッド、関数、変数の数。

引数

source リスト、連想配列、行列、ディスプレイボックス、または名前空間。

```
Remove(source, position, <n>)
Remove(source, {items})
Remove(source, key)
```

説明

指定の場所（*position*）から数えて *n* 番目の項目を削除する。*n* が指定されていない場合、*position* にある1項目だけを削除します。*position* と *n* が指定されていない場合、最後の1項目だけを削除します。連想配列の場合、キー（*key*）およびその値を削除します。

戻り値

項目が削除されたソース（*source*）のコピー

引数

source 引用符付き文字列、リスト、ベクトル、式、または連想配列。

position または key リストまたは式における特定の項目の位置を指す整数（または整数のリスト）。

n （オプション）削除する項目の個数を指定する整数。

Remove From(source, position, <n>)

Remove From(source, key)

説明

Remove関数と同じだが、結果を元の変数に格納する。*n*が指定されていない場合、*position*にある1項目だけを削除します。*position*と*n*が指定されていない場合、最後の1項目だけを削除します。連想配列の場合、キー（*key*）およびその値を削除します。*source*は、左辺値（L-value）でなければなりません。

戻り値

ソース（*source*）から削除された項目のリスト

引数

source 引用符付き文字列、リスト、ベクトル、式、ディスプレイボックス、連想配列。

position または key リストまたは式における特定の項目の位置を指す整数（または整数のリスト）。

n （オプション）削除する項目の個数を指定する整数。

Reverse(source)

説明

ソース（*source*）の要素や項目の順序を逆にする。

引数

source 引用符付き文字列、リスト、行列、式。

Reverse Into(source)

説明

ソース（*source*）の要素や項目の順序を逆にし、結果を*source*に格納する。

引数

source 引用符付き文字列、リスト、行列、ディスプレイボックス、式。

Shift(source, <n>)

説明

ソース（*source*）の最初の1つまたは*n*個の項目を末尾に移動する。

引数

source 引用符付き文字列、リスト、行列、式。

n （オプション）移動する項目の個数を指定する整数。正の値を指定すると、項目をソース（*source*）の冒頭から末尾へ移動します。負の値を指定すると、項目をソース（*source*）の末尾から冒頭へ移動します。デフォルト値は1です。

Shift Into(source, <n>)

説明

Shift関数と同じだが、結果を元の変数に格納する。sourceは、左辺値（L-value）でなければなりません。

引数

source 引用符付き文字列、リスト、行列、ディスプレイボックス、式。

n （オプション）移動する項目の個数を指定する整数。正の値を指定すると、項目をソース（*source*）の冒頭から末尾へ移動します。負の値を指定すると、項目をソース（*source*）の末尾から冒頭へ移動します。デフォルト値は1です。

Sort List({list}|expr)

説明

リスト（*list*）または式（*expr*）の要素や項目を並べ替える。

Sort List Into({list}|expr)

説明

Sort List関数と同じだが、結果を元の変数に格納する。リスト（*list*）や式（*expr*）はL-value（左辺に指定できるもの）でなければなりません。

Substitute(*string*, "substring", "replacementString", ...)

Substitute({list}, listItem, replacementItem, ...)

Substitute(Expr(sourceExpr), Expr(findExpr), Expr(replacementExpr), ...)

説明

検索と置換を行う。第1引数で指定したソース内から、第2引数で指定した特定の部分を検索し、第3引数で指定した項目で置換します。

引用符付き文字列の場合、ソース文字列（*string*）内の部分文字列（*substring*）に一致する部分を、すべて置換文字列（*replacementString*）に置換します。

リストの場合、ソースリスト（*list*）内のリスト項目（*listItem*）に一致する部分を、すべて置換項目（*replacementItem*）に置換します。

式の場合、ソース式（*sourceExpr*）内の検索式（*findExpr*）に一致する部分を、すべて置換式（*replacementExpr*）に置換します。ただし、すべての式を、Expr()関数の中に含める必要があります。

引数

string, list, sourceExpr 置換を行う対象の引用符付き文字列、リスト、行列、式。

substring, listItem, findExpr 検索する引用符付き文字列、リスト項目、式。

replacementString, replacementItem, replacementExpr 置換後の引用符付き文字列、リスト項目、式。

Substitute Into(*string*, substring, replacementString, ...)

Substitute Into(list, listItem, replacementItem, ...)

Substitute Into(Expr(sourceExpr), Expr(findExpr), Expr(replacementExpr), ...)

説明

Substitute() と同様に検索と置換を行うが、結果を元の変数に格納する。第1引数で指定したソース内から、第2引数で指定した特定の部分を検索し、第3引数で指定した項目で置換します。第1引数はL-value（左辺に指定できるもの）でなければなりません。

引用符付き文字列の場合、ソース文字列（*string*）内の部分文字列（*substring*）に一致する部分を、すべて置換文字列（*replacementString*）に置換します。

リストの場合、ソースリスト（*list*）内のリスト項目（*listItem*）に一致する部分を、すべて置換項目（*replacementItem*）に置換します。

式の場合、ソース式（*sourceExpr*）内の検索式（*findExpr*）に一致する部分を、すべて置換式（*replacementExpr*）に置換します。ただし、すべての式を、Expr() 関数の中に含める必要があります。

引数

string, list, sourceExpr 置換を行う対象の引用符付き文字列、リスト、行列、式。

substring, listItem, findExpr 検索する引用符付き文字列、リスト項目、式。

replacementString, replacementItem, replacementExpr 置換後の引用符付き文字列、リスト項目、式。

Words(*string*, <delimiter>)

説明

引数の引用符付き文字列（*string*）から単語を抽出する。区切り文字（delimiters）を指定すると、文字列を各単語に区切る際に、その文字が使われます。デフォルトの区切り文字はASCII文字のスペースです。複数の区切り文字を指定した場合、指定したすべての文字が区切り文字とみなされます。*delim*を空白とした場合、1文字1文字がそれぞれ個別の項目とみなされます。

例

```
Words( "the quick brown fox" );
   {"the","quick","brown","fox"}
Words( "Doe, Jane P.",", ." );
   {"Doe","Jane","P"}
```

Where(<dt>, clause)

説明

指定された節にしたがって項目をフィルタリングし、インデックスを戻す。節には、比較関数または条件文を指定できます。同じ節の中で列・行列・リストを混ぜて照合することもできます。

Where()によって戻されるインデックスは、大量のリスト・行列・列を高速に処理できるよう最適化されています。

必須の引数

clause 比較関数または条件文。

オプションの引数

<dt> 評価時に現在のデータテーブルを変更する。

例

```
Names Default To Here( 1 );
xs = [10 20 30 .50];
xs[Where( xs >= 20 )];
xs[Where( !Is Missing( xs ) )];
ys = {10, 20, "30", ., 50};
ys[Where( ys >= 20 )];
```

MATLABインテグレーション関数

JMPには、MATLABへのインターフェースが関数として用意されています。基本的には、まず、MATLABへの接続を開始し、次に、何らかの処理をMATLAB上で実行し、最後にMATLABへの接続を解除します。これらの関数の多くは、MATLABの処理が正常に実行された場合は0、そうでない場合は、エラーコードを戻します。MATLABの処理が正常に実行されなかった場合は、「ログ」ウィンドウにメッセージが表示されます。ただし、MATLAB Get( )関数だけは、エラーコード以外の値を戻します。

MATLAB JSL関数インターフェース

MATLAB Connect(<named arguments>)

説明

現在のMATLAB接続オブジェクトを戻します。もし、現在、MATLABに接続されていない場合は、JMPからMATLABへの接続を初期化した後、MATLAB接続オブジェクトを戻します。

戻り値

スクリプト可能なMATLABオブジェクト

名前付き引数

Echo(Boolean) 実行したMATLABのプログラムコードを、JMPのログに出力する。デフォルト値は、1（真）。

MATLAB Control(<named arguments>)

説明

プログラムコードの表示などの外部イベントの制御命令をMATLABに送る。

戻り値

なし

引数

なし。

名前付き引数

Echo(Boolean) グローバル。実行したMATLABのプログラムコードを、JMPのログに出力する。

Visible(Boolean) グローバル。アクティブなMATLABワークスペースの表示／非表示を指定する。

MATLAB Execute({ list of inputs }, { list of outputs }, mCode, <named arguments>)

説明

現在のグルーバルなMATLAB接続に対して、第1引数のリストに指定されたJMP変数を送り、第3引数に指定されたMATLABコードをサブミットする。第2引数のリストに指定された変数が、JMPに戻されます。

戻り値

成功した場合は0、そうでない場合は0以外

引数

{ list of inputs } 位置固定、名前のリスト。入力としてMATLABに送られるJMP変数名のリスト。

{ list of outputs } 位置固定、名前のリスト。出力としてMATLABから取得されるJMP変数名のリスト。

mCode 位置固定、引用符付き文字列。サブミットされるMATLABコード。

名前付き引数

Expand(Boolean) MATLABコードのサブミット前に、コードに対してEval Insertを実行する。

Echo(Boolean) 実行したMATLABのプログラムコードを、JMPのログに出力する。デフォルト値は1（真）です。

例

次の例では、JMP変数xとyをMATLABに送り、MATLABステートメント *z = x * y*を実行して、MATLAB変数zを取得してJMPに戻します。

```
MATLAB Init();
x = [1 2 3];
y = [4 5 6];
MATLAB Execute( {x, y}, {z}, "z = x * y;" );
Show( z );
```

MATLAB Get(name)

説明

name引数で指定されたMATLABの変数を、JMPで取得する。

戻り値

name引数で指定された変数の値

引数

name 位置固定。MATLABに送るJMP変数の名前。

例

qbxという名前の行列と、dfという名前の構造体が、MATLAB上に存在するとします。

```
// MATLAB変数qbxの値を取得し、それをJMP変数qbxに代入
qbx = MATLAB Get( qbx );

/* MATLAB変数dfのデータフレームを、JMPの変数dfにて参照される
JMPデータテーブルとして取得*/
df = MATLAB Get( df );
```

表2.1に、MATLAB Get( )関数でMATLABからJMPに変数を取得した場合に、MATLABのデータタイプ（データ型）が、JMPにおいて、どのような型に変換されるかを示します。リストの場合は、リスト内の要素ごとにデータタイプをチェックして、変換します。なお、入れ子になっているリストも、サポートされています。

表2.1 MATLAB Get( )関数のJMPデータタイプとMATLABデータタイプ

MATLABデータタイプ	JMPデータタイプ
実数（double）	数値
論理	数値 (0 | 1)
文字列	文字列
整数	数値
日付／時間	数値
構造体	データテーブル
行列	行列
数値ベクトル	行列
文字列ベクトル	文字列のリスト
グラフ	ピクチャーオブジェクト

MATLAB Get Graphics(format)

説明

MATLABグラフ表示ウィンドウに書き込まれた最後のグラフオブジェクトを、format引数で指定されたグラフィック形式で取得する。

戻り値

JMPピクチャーオブジェクト

引数

format 位置固定。MATLABのグラフを取得するときの変換形式。有効な形式は、png、bmp、jpeg、jpg、tiff、tif、gifです。

MATLAB Get Version

説明

JMPのMATLABインターフェースで使用されているMATLABのバージョン番号を戻す。

MATLAB Init(<named arguments>)

説明

MATLAB接続インターフェースを初期化する。

戻り値

リターンコード

名前付き引数

Echo(Boolean)は、実行したMATLABのプログラムコードを、JMPのログに出力する。このオプションは、グローバルです。デフォルト値は、1（真）。

MATLAB Is Connected()

説明

MATLAB接続がアクティブかどうかを調べる。

戻り値

アクティブなMATLAB接続がある場合は1、そうでない場合は0を戻す。

MATLAB JMP Name To MATLAB Name(name)

説明

MATLABの命名規則に従い、JMP変数名を、対応するMATLAB変数名に変換する。

戻り値

マップされたMATLABの変数名を引用符付き文字列として戻す。

引数

name 位置固定。MATLABに送るJMP変数の名前。

MATLAB Send(name, <named arguments>)

説明

名前付き変数をJMPからMATLABに送る。

戻り値

成功した場合は0、そうでない場合は0以外

引数

name 位置固定。MATLABに送るJMP変数の名前。

名前付き引数

次のオプションの引数は、データテーブルにのみ適用されます。

Selected(Boolean) 参照先データテーブルの選択されている行をMATLABに送る。

Excluded(Boolean) 参照先データテーブルの除外されている行のみをMATLABに送る。

Labeled(Boolean) 参照先データテーブルのラベルのついている行のみをMATLABに送る。

Hidden(Boolean) 参照先データテーブルの非表示の行のみをMATLABに送る。

Colored(Boolean) 参照先データテーブルの色のついている行のみをMATLABに送る。

Markered(Boolean) 参照先データテーブルのマーカーのついている行のみをMATLABに送る。

Row States(Boolean, <named arguments>) 参照先データテーブルにおける行属性の情報を、「RowState」という列名のデータ列を追加して、MATLABに送る。個々の設定をあわせて追加すれば、複数選択も可能です。行属性の各設定には、それぞれ次の値が割り当てられています。

- Selected = 1
- Excluded = 2
- Labeled = 4
- Hidden = 8
- Colored = 16
- Markered = 32

Row Statesには、次の名前付き引数をオプションで指定できます。

Colors(Boolean) 行の色を送る。「RowStateColor」という名前のデータ列を追加します。

Markers(Boolean) 行のマーカーを送る。「RowStateMarker」という名前のデータ列を追加します。

例

```
// 行列を変数Xに代入し、その変数をMATLABに送る
X = [1 2 3];
ml = MATLAB Send( X );

/* データテーブルを開き、データテーブルへの参照をdtに割り当て、
```

```
データテーブルとその行の属性をMATLABに送る */
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
ml = MATLAB Send( dt, Row States( 1 ) );
```

表2.2に、MATLAB Send( )関数でJMPからMATLABに変数を送った場合に、JMPのデータタイプ（データ型）が、MATLABにおいて、どのような型に変換されるかを示します。リストの場合は、リスト内の要素ごとにデータタイプをチェックして、変換します。なお、入れ子になっているリストも、サポートされています。

表2.2 MATLAB Send( )関数のJMPデータタイプとMATLABデータタイプ

MATLABデータタイプ	JMPデータタイプ
実数（double）	数値
文字列	文字列
実数の行列	行列
構造体	データテーブル

例

```
MATLAB Init( );
X = 1;
MATLAB Send( X );
S = "Report Title";
MATLAB Send( S );
M = [1 2 3, 4 5 6, 7 8 9];
MATLAB Send( M );
MATLAB Submit( "
X
S
M
" );
MATLAB Term( );
```

MATLAB Send File(filename, <MATLAB Name(name)>)

説明

データファイルをMATLABに送る。

引数

filename MATLABに送るファイルのパス名を引用符付き文字列として指定する。

MATLAB Name MATLABでのファイル名を変更する場合に指定する。

MATLAB Submit File(pathname, <named arguments>)

説明

pathnameで指定されたファイルに含まれるMATLABコードを、MATLAB上で実行する。

戻り値

成功した場合は0、そうでない場合は0以外

引数

pathname 位置固定、引用符付き文字列。実行するMATLABプログラムコードを含むファイルのパス名。

名前付き引数

Expand(Boolean) MATLABコードのサブミット前に、コードに対してEval Insertを実行する。

Echo(Boolean) 実行したMATLABのプログラムコードを、JMPのログに出力する。デフォルト値は1（真）です。

MATLAB Submit(mCode, <named arguments>)

説明

アクティブなグローバルMATLAB接続に、MATLABコードをサブミットする。

戻り値

成功した場合は0、そうでない場合は0以外

引数

mCode 位置固定、引用符付き文字列。サブミットされるMATLABコード。

名前付き引数

Expand(Boolean) MATLABコードのサブミット前に、コードに対してEval Insertを実行する。

Echo(Boolean) 実行したMATLABのプログラムコードを、JMPのログに出力する。デフォルト値は1（真）です。

例

次の例は、2つの乱数ベクトルを作成し、それらをx変数とy変数にプロットします。

```
MATLAB Init();
mc = MATLAB Submit("/[
   x = rand(5);
   fprintf('%f/n', x);
   y = rand(5);
   fprintf('%f/n', x);
   z = plot(x, y);
]/" );
```

MATLAB Term();

説明

MATLABへの接続を終了する（アクティブなMATLAB接続インターフェースを終了する）。

戻り値

アクティブなMATLAB接続がある場合は1、そうでない場合は0

引数

なし

行列関数

All(A, ...)

戻り値

すべての行列のすべての要素が0以外のときは1、そうでなければ0

Any(A, ...)

戻り値

1つまたは複数の行列のある1要素が0以外のときは1、そうでなければ0

B Spline Coef(x, Internal Knot Grid, <degree=3>, <Knot End Points=min(x)||max(x)>)

説明

*x*引数に指定したデータのB-スプライン係数の行列を戻す。

戻り値

B-スプライン曲線の係数の行列。線形モデルの計画行列として使用できます。行列の最初の列に切片項が含まれます。

引数

x データを含む行ベクトルまたは列ベクトル。

Internal Knot Grid *x*のパーセント点に基づく、必要な節点の数か、または内部節点を指定するベクトル。内部節点の数は、1以上で、*x*内にある一意の要素の数から2を引いた値以下でなければなりません。

degree B-スプライン曲線の次数を示す数値。デフォルトの値は3です。

Knot End Points 節点の下側と上側の位置から成る2×1行列。この引数を指定しなかった場合、デフォルトの下側と上側の接点は*x*の最小値と最大値になります。

メモ

この関数は、「関数データエクスプローラ」プラットフォームで作成される計算式に使用されています。

CDF(Y)

説明

ベクトルまたはリストで指定された*Y*に対し、経験累積分布関数の値を戻す。経験累積分布関数は、*QuantVec*の値以下となっているデータの割合を表します。

構文

```
{QuantVec, CumProbVec} = CDF(YVec)
```

Chol Update(L, V, C)

説明

$n \times n$行列Aのコレスキー根をLとした場合、cholUpdateを呼び出すと、A+V*C*V'（Cは$m \times m$の対称行列、Vは$n \times m$行列）のコレスキー根で置き換わる。

Cholesky(A)

説明

半正定値行列の下コレスキー根（L）を戻す。Lは、L*L' = Aとなるような下三角行列。

戻り値

L（コレスキー根）

引数

A 対称行列。

Correlation(matrix, < <<"Pairwise">, < <<"Shrink">, < <<Freq(vector)>, < <<Weight(vector)>)

説明

行列引数*matrix*の相関係数行列を計算する。

戻り値

指定された行列の相関係数行列

引数

matrix データを示す行列。データがn行m列で構成されている場合、結果は$m \times m$の行列です。

"Pairwise" 欠測値に対してリストワイズ法ではなくペアワイズ法を使用する。

"Shrink" Schafer-Strimmerの縮小推定を実行する。

<<Freq(vector) 行列の行の度数を指定するベクトル。

<<Weight(vector) 行列の行の重みを指定するベクトル。

メモ

デフォルトでは、欠測値が1つでもある行は、行ごと除外されます。"Pairwise"オプションを指定した場合は、欠測値以外のすべてのペアを相関行列の計算に使用します。

この関数は、可能ならマルチスレッドを使用します。そのため、行数が多い大きなデータに適しています。

列が定数の場合、その相関係数は0で、対角要素も0です。

Covariance(matrix, < <<"Pairwise">, < <<"Shrink">, < <<Freq(vector)>, < <<Weight(vector)>)

説明

行列引数*matrix*の共分散行列を計算する。

戻り値

指定された行列の共分散行列

引数

matrix データを示す行列。データがm行n列で構成されている場合、結果は$m \times n$の行列です。

"Pairwise" 欠測値に対してリストワイズ法ではなくペアワイズ法を使用する。

"Shrink" Schafer-Strimmerの縮小推定を実行する。

<<Freq(vector) 行列の行の度数を指定するベクトル。

<<Weight(vector) 行列の行の重みを指定するベクトル。

メモ

デフォルトでは、欠測値が1つでもある行は、行ごと除外されます。"Pairwise"オプションを指定した場合は、欠測値以外の全ペアを共分散行列の計算に使用します。

この関数は、可能ならマルチスレッドを使用します。そのため、行数が多い大きなデータに適しています。

Design(vector, < levelsList | <<levels, <<ElseMissing >)

説明

ベクトル（*vector*）の一意の値ごとに、1と0の列でできた計画行列を作成する。

戻り値

引数の一意の値ごとに1と0の列を持つ計画行列、または計画行列と水準のリストを含むリスト

引数

vector ベクトル。

levelsList（オプション）戻される行列の水準を指定するリストまたは行列。

<<levels（オプション）これを指定すると、戻り値が計画行列と水準を含むリストになる。

<<ElseMissing（オプション）引数*vector*に、引数*levelsList*に含まれない値があった場合、その値の処理を指定する。この引数を指定すると、計画行列に欠測値が挿入されます。指定しない場合は、計画行列に0が挿入されます。

メモ

*levelsList*引数内の欠測値は無視されません。

例:

```
Show( Design ( ., [. 0 1] ),
Design( 0, [. 0 1] ),
Design( 1, [. 0 1] ),
Design( [0 0 1 . 1], [. 0 1] ),
Design( {0, 0, 1, ., 1}, [. 0 1] ) );
   Design(., [. 0 1]) = [1 0 0];
   Design(0, [. 0 1]) = [0 1 0];
   Design(1, [. 0 1]) = [0 0 1];
   Design([0 0 1 . 1], [. 0 1]) =
   [  0 1 0,
      0 1 0,
      0 0 1,
      1 0 0,
      0 0 1];
   Design({0, 0, 1, ., 1}, [. 0 1]) =
   [  0 1 0,
      0 1 0,
      0 0 1,
      1 0 0,
      0 0 1];
```

Design Last(vector, < levelsList, <<ElseMissing >)

説明

引数の一意の値ごとに、1と0の列を持つ計画行列を作成する。ただし、最後の水準は、0だけの行とします。

戻り値

フルランクの計画行列、または計画行列と水準のリスト

引数

vector ベクトル。

levelsList（オプション）戻される行列の水準を指定するリストまたは行列。この引数を指定すると、このリストまたは行列の最後の水準が、計画行列の最後の水準として扱われます。指定しない場合は、引数 *vector*の最大値が、最後の水準に設定されます。

<<ElseMissing（オプション）引数 *vector*に、引数 *levelsList*に含まれない値があった場合、その値の処理を指定する。この引数を指定すると、計画行列に欠測値が挿入されます。指定しない場合は、計画行列に0が挿入されます。

Design Nom(vector, < levelsList | <<levels, <<ElseMissing >)

DesignF(vector, < levelsList | <<levels, <<ElseMissing >)

説明

引数の一意の値ごとに、1と0の列を持つ計画行列を作成する。ただし、最後の水準は、-1の行としてコード化されます。

戻り値

フルランクの計画行列、または計画行列と水準のリスト

引数

vector ベクトル。

levelsList（オプション）戻される行列の水準を指定するリストまたは行列。この引数を指定すると、このリストまたは行列の最後の水準が、計画行列の最後の水準として扱われます。指定しない場合は、引数 *vector*の最大値が、最後の水準に設定されます。

<<levels（オプション）これを指定すると、戻り値が計画行列と水準を含むリストになる。

<<ElseMissing（オプション）引数 *vector*に、引数 *levelsList*に含まれない値があった場合、その値の処理を指定する。この引数を指定すると、計画行列に欠測値が挿入されます。指定しない場合は、計画行列に0が挿入されます。

メモ

*levelsList*引数内の欠測値は無視されません。

例:

```
Show( Design Nom( ., [. 0 1] ),
Design Nom( 0, [. 0 1] ),
Design Nom( 1, [. 0 1] ),
Design Nom( [0 0 1 . 1], [. 0 1] ),
Design Nom( {0, 0, 1, ., 1}, [. 0 1] ) );
   Design Nom(., [. 0 1]) = [1 0];
   Design Nom(0, [. 0 1]) = [0 1];
   Design Nom(1, [. 0 1]) = [-1 -1];
   Design Nom([0 0 1 . 1], [. 0 1]) = [0 1, 0 1, -1 -1, 1 0, -1 -1];
   Design Nom({0, 0, 1, ., 1}, [. 0 1]) = [0 1, 0 1, -1 -1, 1 0, -1 -1];
```

Design Ord(vector, < levelsList | <<levels, <<ElseMissing >)

説明

引数の一意の値ごとに列を持つ計画行列を作成する。最初の水準は、0の行としてコード化されます。引数 *levelsList*のそれ以降のn番目の水準は、（n-1）個の1およびそれ以外が0の行としてコード化されます。

戻り値

フルランクの計画行列、または計画行列と水準のリスト

引数

vector ベクトル。

levelsList（オプション）戻される行列の水準を指定するリストまたは行列。

<<levels（オプション）これを指定すると、戻り値が計画行列と水準を含むリストになる。

<<ElseMissing（オプション）引数*vector*に、引数*levelsList*に含まれない値があった場合、その値の処理を指定する。この引数を指定すると、計画行列に欠測値が挿入されます。指定しない場合は、計画行列に0が挿入されます。

メモ

*levelsList*引数内の欠測値は無視されません。

例:

```
Show( Design Ord( ., [. 0 1] ),
Design Ord( 0, [. 0 1] ),
Design Ord( 1, [. 0 1] ),
Design Ord( [0 0 1 . 1], [. 0 1] ),
Design Ord( {0, 0, 1, ., 1}, [. 0 1] ) );
   Design Ord(., [. 0 1]) = [0 0];
   Design Ord(0, [. 0 1]) = [1 0];
   Design Ord(1, [. 0 1]) = [1 1];
   Design Ord([0 0 1 . 1], [. 0 1]) = [1 0, 1 0, 1 1, 0 0, 1 1];
   Design Ord({0, 0, 1, ., 1}, [. 0 1]) = [1 0, 1 0, 1 1, 0 0, 1 1];
```

Det(A)

説明

正方行列の行列式。

戻り値

行列式

引数

A 正方行列。

Diag(A, <B>)

説明

正方行列やベクトルから対角行列を作成する。2つ以上の行列が指定された場合、それらの行列を対角線上に連結したブロック対角行列を戻します。

戻り値

行列

引数

A 行列またはベクトル。

Direct Product(A, B)

説明

行列の直積（Kronecker積）を戻す。

戻り値

行列の直積

引数

A, B 行列。

Distance(x1, x2, <scales>, <powers>)

説明

x1の行とx2の行との間の距離の行列を生成する。

戻り値

行列

引数

x1, x2 2つの行列。

scales（オプション）行列の尺度を設定する引数。

powers（オプション）行列のべき乗を設定する引数。

E Div(A, B)

A:/B

戻り値

2つの行列の要素ごとの割り算を示す行列

引数

A, B 2つの行列。両方の行列の行数および列数が同じでなければいけません。

E Max(A, B)

戻り値

複数の行列またはスカラー引数の要素ごとの最大値を示す行列

引数

A, B 2つ以上の行列またはスカラー。すべての行列の行数および列数が同じでなければいけません。

E Min(A, B)

戻り値

複数の行列またはスカラー引数の要素ごとの最小値を示す行列

引数

A, B 2つ以上の行列またはスカラー。すべての行列の行数および列数が同じでなければいけません。

E Mult(A, B)

A:*B

説明

2つの行列の要素ごとの掛け算。

戻り値

複数の行列またはスカラー引数の要素ごとの掛け算を示す行列

引数

A, B 2つ以上の行列またはスカラー。すべての行列の行数および列数が同じでなければいけません。

Eigen(A)

説明

固有値分解。

戻り値

対称行列Aに対して、A'=E * Diag(M) * EとなるようなMとEが、リスト{M, E}の形式で戻される。

引数

A 対称行列。

Estimate Bartlett Factor Score(dataRow, ManMeans, LatMeans, S, A)

説明

Bartlettの手法を用いて構造方程式モデル（SEM）の因子スコアを推定する。

戻り値

構造方程式モデルに基づいて推定した因子スコアの行ベクトル

引数

dataRow データ値の行ベクトル。

ManMeans モデルから求めた顕在変数の平均のベクトル。

LatMeans モデルから求めた潜在変数の平均のベクトル。

S 構造方程式モデルの対称なS行列。

A 構造方程式モデルの長方形のA行列。

Estimate Factor Score(dataRow, Covariance, ManMeans, LatMeans)

説明

構造方程式モデル（SEM）の因子スコアを推定する。

戻り値

構造方程式モデルに基づいて推定した因子スコアの行ベクトル

引数

dataRow データ値の行ベクトル。

Covariance モデルから求めた分散共分散行列。

ManMeans モデルから求めた顕在変数の平均のベクトル。

LatMeans モデルから求めた潜在変数の平均のベクトル。

メモ

この関数は、「構造方程式モデル」レポートの［因子スコアの保存］オプションで使用されています。

Fourier Basis Coef(x, Number Pairs, <Period=max(x)-min(x)+1>)

説明

引数 *x* に指定したデータの Fourier 基底の係数行列を戻す。

戻り値

Fourier 基底の係数行列。線形モデルの計画行列として使用できます。行列の最初の列に切片項が含まれます。残りの列には、基底のペアが含まれ、ペア *i* は、*i* * (2 * π / Period) * x の sin() および cos() として定義されます。

引数

x データを含む行ベクトルまたは列ベクトル。

Number Pairs Fourier 基底の sin() と cos() のペア。

Period Fourier 基底を構成する三角関数の周期。

メモ

この関数は、「関数データエクスプローラ」プラットフォームで作成される計算式に使用されています。

G Inverse(A)

説明

Moore-Penrose 型の一般逆行列。

H Direct Product(A, B)

説明

行数が等しい行列の横の直積。

Hough Line Transform(matrix, <NAngle(number)>, <NRadius(number)>)

説明

彩度の行列をとり、それを行列内の線を見つけやすいように変換する。角度を列、半径を行とした Hough Line Transform を含む行列を作成します。

引数

matrix イメージの彩度から得られた行列の場合もあるが、平坦化の装置によって筋状に問題が発生している可能性のある半導体ウエハーから得られたものである場合が多い。

NAngle(number) 異なるサイズで変換するための角度を入力する。デフォルト値は180度です。

NRadius(number) 異なるサイズで変換するための半径を入力する。デフォルトは sqrt(NRow*nRow+nCol*nCol) です。

Identity(n)

説明

n × *n* の単位行列を作成する。単位行列は、対角要素が1で、非対角要素が0である行列です。

戻り値

行列

引数

n 整数。

Index(i, j, <increment>)

i::j

説明

i から *j* までの整数を含む行ベクトルを作成する。

戻り値

行列

引数

i, j 範囲を定義する整数。*i* は範囲の始点、*j* は終点。

increment（オプション）これを指定することにより、増分をデフォルト値の+1から変更できる。

Inv()

「Inverse(A)」を参照してください。

Inv Update(A, X, 1|-1)

説明

A=Inv(**X'X**)がすでに計算されている場合、行列Xに行ベクトルxを追加／削除した後の、Inv(X'X)を効率的に計算する。

引数

A 更新する行列。

X 行列*A*に対して追加／削除する1つまたは複数の行。

1|-1 第3引数は、第2引数*x*で指定された行を行列*A*に追加／削除するかどうかを制御する。1は行を追加し、-1は行を削除します。

Inverse(A)

Inv(A)

説明

逆行列を戻す。行列は正則な正方行列でなければなりません。

Is Matrix(x)

説明

評価後の引数が行列のときは1、そうでなければ0を戻す。

J(nrows, <ncols>, <value>)

説明

すべての要素が同じ値（value）の行列を作成する。

戻り値

行列

引数

nrows 行列の行数。列数（ncols）が指定されていない場合、列数は行数と同じ数に設定されます。

ncols 行列の列数。

value 行列を埋める値。valueが指定されていない場合、1が使用されます。

KDTable(matrix)

説明

*k*次元のツリー（*k*-d木）アルゴリズムを使って作成したテーブルを戻す。このテーブルを使うと、近傍点を効率的に探すことができます。*k*-d木は、*k*次元空間内で点を探すためのデータ構造です。各次元について点を小さなサブセットに分割し、再帰的にツリーを構築します。アルゴリズムは、クエリー点を各ノードの分割値と比較して該当する部分木（サブツリー）を探索します。近傍点検索など、多次元

データを扱う場合に便利です。*k*-d 木の構造の計算方法については、Bentley (1975) を参照してください。

戻り値

K次元テーブルオブジェクト

引数

`matrix` *k*次元の点の行列。次元数または点数に制限はありません。行列の各列はデータの次元、各行はデータ点を表しています。

メッセージ

`<<Distance between rows(row1, row2)` K次元テーブル内の指定された2つの行の間の距離を戻す。この距離は、削除された行および挿入された行に対しても適用されます。

`<<K nearest rows(stop, <position>)` 行列を戻す。*<position>*には、*n*個の近傍点を求めたい点を、座標の行ベクトル、もしくは、行番号の整数で指定します。*<position>*が指定されていない場合は、すべての行に関する結果を戻します。*<position>*が指定されている場合は、指定された点または行に対する結果を戻します。引数*stop*には、*n*または{*n*, *limit*}を指定してください。これらの制限に関するパラメータは、結果として戻される行の行数を定義します。数値またはリストで指定できます。数値（たとえば5）を指定した場合、最も近傍にある5行だけが戻されます。また、リスト（たとえば{5, 10}）を指定した場合、距離が10を超えるまで、最大5つの近傍の行が戻されます。*limit*にリストを使用した場合、戻された最後の行の距離は10を超える可能性があります。*stop*の距離を超えた次の行が見つかるまでコマンドが続行されるため、この最後の点も戻されます。これは、特に*stop*の距離内に行がない場合に役立ちます。

`<<Remove rows(number | vector)` *number*で指定された行、または、*vector*で指定された行を、削除する。削除された行の行数を戻します。すでに削除されている行は無視されます。

`<<Insert rows(number | vector)` *number*で指定された行、または、*vector*で指定された行を、再度挿入する。挿入された行の行数を戻します。すでに挿入されている行は無視されます。

メモ

行が削除または挿入されても、行に対する通し番号は変更されません。削除または再挿入できるのは、既存のKDテーブルオブジェクトに含まれている行だけです。新しい行を含める必要がある場合は、別のKDテーブルを新たに作成してください。

`Least Squares Solve(y, X, < <<noIntercept, <<weights(OptionalWeightVector), <<method("Sweep"|"GInv")>)`

説明

線形モデル「y = X * beta + error」の推定値を最小2乗法で計算する。

戻り値

行列`Beta=Inverse(X'X')X'y`およびBetaの分散共分散行列の推定値を含むリスト

オプションの名前付き引数

`<<noIntercept` 切片のないモデルを指定する。

<<weights(optional weight vector) 重み（ベクトル）を指定して、重み付き最小2乗法を実行する。

<<method("Sweep" | "GInv") どの手法で正規方程式を解くかを指定する。Sweep法（デフォルト）の方が計算速度は速く、消費メモリも少ないですが、一般化逆行列（"GInv"）の方が数値的に安定しています。

Linear Regression(y, X, < <<noIntercept, <<printToLog, <<weight(OptionalWeightVector), <<freq(OptionalFreqVector)>)

説明

線形モデル「y = X * beta + error」の推定値を最小2乗法で計算し、いくつかの診断統計量も求める。

戻り値

推定値のベクトル、標準誤差のベクトル、および診断統計量のリストを含むリスト。診断統計量のリストには、推定値の*t*値と*p*値のベクトル、そして回帰のあてはめのR2乗と自由度調整R2乗の値が含まれます。

オプションの名前付き引数

<<noIntercept 切片のないモデルをあてはめる。

<<printToLog あてはめの要約をログに出力する。

<<weight(vector) 重み（ベクトル）を指定して、重み付き最小2乗法を実行する。

<<freq(vector) yとXの各行に対する、度数のベクトルを指定する。

例

```
n = 10;
x = J( n, 1, Random Normal() );
y = 1 + x * 3 + J( n, 1, Random Normal() );
{Estimates, Std_Error, Diagnostics} = Linear Regression( y, x, <<printToLog );
As Table( y || x );
Bivariate( Y( :Col1 ), X( :Col2 ), Fit Line( 1 ) );
```

Loc(A)

Loc(A, item)

説明

行列*A*の要素のうち、0でも欠測値でもない要素の位置を示す、添え字の行列を戻す。引数が2つ指定された場合は、*A*のうち要素*item*が見つかった位置の行列を戻します。最初の引数がリストの場合は、2つ目の引数が必須になります。

高速性と柔軟性を求める場合は、代わりにWhere()を使用してください。「Where(<dt>, clause)」を参照してください。

引数

A 行列またはリスト。

item 行列またはリスト*A*の中にある要素。

Loc Max(A)

説明

行列の要素のうち、最大値である要素の位置を戻す。

戻り値

最小値の位置を示す整数

引数

A 行列。

Loc Min(A)

説明

行列の要素のうち、最小値である要素の位置を戻す。

戻り値

最小値の位置を示す整数

引数

A 行列。

Loc NonMissing(matrix, ..., {list}, ...)

説明

指定された行列またはリスト内で、欠測値がない行の番号を戻す。引数がリストの場合、空でない文字列の番号も戻すことができます。

戻り値

新しい行列またはリスト

Loc Sorted(A, B)

説明

二分探索を通じて*A*の値が*B*の各要素以下である位置を、各要素ごとに探し、その位置の添え字の列ベクトルを作成する。*A*は、昇順で並べられた欠測値のない行列でなければなりません。

戻り値

*B*と同じ次元を持つ新しい行列。*B*にある値が*A*の最初の値より小さい場合、戻される添え字の値は1になります。

引数

A, B 行列

Matrix({{x11, ..., x1m}, {x21, ..., 2m}, {...}, {xn1, ..., xnm}})

Matrix({x1, ..., xn})

Matrix(n, m)

説明

n × *m*行列を作成する。次のような指定方法があります。

- それぞれが*m*個の要素を含んでいる*n*個のリストを指定した場合、各リストを縦に連結して行列が作成されます。各リストが行列の各行ベクトルになりますので、各リストの項目数は一致していなければなりません。
- *n*個の項目を持つ1つのリストを指定した場合、戻り値は、*n* × 1の列ベクトルです。リストの項目は、評価後数値にならなければなりません。
- 2つの整数引数を指定した場合、戻り値は、*n*行、*m*列から成るゼロの行列です。

例

```
Matrix({{1, 2, 3}, {4, [5 6]}, {7, 8, 9}});
   [1 2 3, 4 5 6, 7 8 9]
Matrix({{[1 2 3], 4, 5, 6, 7, 8, 9}});
   [1 2 3 4 5 6 7 8 9]
Matrix({2,3+7});
   [2, 10]
Matrix(2,3);
   [0 0 0, 0 0 0]
```

Matrix Mult(A, B)

C=A*B, ...

説明

行列の掛け算。

引数

A, B, ... 行数と列数が整合している2つ以上の行列（すべての乗算において、掛けられる行列の列数が掛ける行列の行数と一致している必要がある）。

メモ

Matrix Mult()で指定できる引数は2つだけですが、*演算子を使用すれば3つ以上の行列を掛けることができます。

Matrix Rank(A)

説明

行列Aのランク（階数）を戻す。

Mode({list} or matrix)

説明

数値または文字のリスト、あるいは数値行列から、最も頻繁に出現する項目を選択する。頻度の同じものがある場合は、最も小さい値が選択されます。複数の引数を指定する場合は、数値と引用符付き文字列を組み合わせることもできます。

引数

リストまたは行列を指定する。

Multivariate Normal Impute(yVec, meanYvec, symCovMat, colMin, colMax)

説明

応答ベクトルyVecにおける欠測値を平均と共分散に基づいて補完する。

引数

yVec いくつかの要素が欠測値となっているベクトル。

meanYvec 平均のベクトル。

symCovMat 共分散行列（対称行列）。共分散行列を指定しなかった場合は、平均で補完されます。

colMin 列の最小値のベクトル。補完の下限を指定します。

colMax 列の最大値のベクトル。補完の上限を指定します。

NChooseK Matrix(n, k)

説明

*n*個から*k*個を選んだ場合のすべての組み合わせを含む行列を戻す。

N Col(x)

N Cols(x)

説明

指定されたデータテーブルまたは行列の列数を戻す。

引数

x データテーブルまたは行列。

Ortho(A, <Centered(0)>, <Scaled(1)>)

説明

行列*A*の列をGram-Schmidt法で直交正規化する。Centered(0)を指定すると、直交化後の列の和がゼロに中心化されない。Scaled(1)を指定すると、長さが1に尺度化されない。

Ortho Poly(vector, order)

説明

説明変数の間隔を表すベクトル（*vector*）に対し、指定の次数（*order*）までの直交多項式を戻す。

P Spline Coef(x, Internal Knot Grid, <degree=3>)

説明

*x*引数に指定したデータの罰則付きB-スプライン（P-スプライン）係数の行列を戻す。

戻り値

P-スプライン曲線の基底を含む行列。P-スプライン曲線の基底は、指定された次数（*degree*）の切断べき関数で構成されています。k_1からk_Kまでの節点を持つ、P-スプライン曲線の基底は、次のように定義されます。

$$1, x, x^2, \ldots, x^p, (x-k_1)_+^p, \ldots, (x-k_K)_+^p$$

ここで、$(x - k_1)_+$は、$(x - k_1)$が正のときにはそのままの値、0以下の場合には0となる関数です。

引数

x データを含む行ベクトルまたは列ベクトル。

Internal Knot Grid *x*のパーセント点に基づく、必要な節点の数か、または内部節点を指定するベクトル。

degree P-スプライン曲線の次数を示す数値。デフォルトの値は3です。

メモ

この関数は、「関数データエクスプローラ」プラットフォームで作成される計算式に使用されています。

Parallel Assign({thread_local_var = global_var, ...}, matrix[a, b] = expression using a and b)

説明

複数のスレッドを使って行列に値を割り当てる。コンピュータのマルチコアが活用できます。関数には2つの引数があります。

- 最初の引数は、グローバル変数を各スレッドのローカル変数のリストにコピーする割り当てのリストです。
- 2番目の引数は、割り当ての式で、左辺が1つまたは2つの添え字を指定された行列、右辺がそれらの添え字とリストのローカル変数（およびJMPのグローバル変数）を使った任意のJSL式です。

例

次の例は、グローバル名前空間への読み取りアクセスを可能にします。

```
a = 42;
x = [1 2 3, 4 5 6, 7 8 9];
Show( Parallel Assign( {}, x[i, j] = global:a ) );
```

```
Show( x );
   Parallel Assign({}, x\[i,j] = global:a) = 1;
   x =
   [  42 42 42,
      42 42 42,
      42 42 42];
```

Print Matrix(M, <named arguments>)

説明

表示形式を整えた行列を含む引用符付き文字列を戻す。たとえば、この関数を使用して、行列をログに出力できます。

引数

M 行列。

オプションの名前付き引数

<<ignore locale(Boolean) 偽（0）の場合は、ロケールの小数点記号が使われる。真（1）の場合は、常にピリオド（.）を小数点記号とする。デフォルト値は0（偽）です。

<<decimal digits(n) 表示する小数桁数を指定する整数。

<<style("style name") Parseable、Latex、Otherのいずれかのスタイルを指定する。ParseableはJSLと同じ表示形式です。LatexはLaTeX用の形式です。Otherを指定した場合、次の3つの引数を指定する必要があります。

<<separate("character") 要素の区切り記号を定義する。

<<line begin("character") 開始行の文字を定義する。

<<line end("character") 終了行の文字を定義する。

QR(A)

説明

*A*のQR分解を戻す。通常は、{Q, R} = QR(A)のように使います。

Random SVD(X, <nSingularValues=min(nRow, nCol)>, <nOver=10>, <nIter=2>)

説明

乱択特異値分解を使用して、行列Xの特異値分解（SVD）を行う。従来の特異値分解より高速で、通常は近似が正確です。特に、行列が大きく、必要な特異値の数が少ない場合に高速です。この手法では、データを低次元空間にランダムに投影して低次元空間での特異値分解を行い、元のデータの特異値分解を推定します。計算に使用する空間の次元は、特異値の数に、オーバーサンプリング列の数を足したものに等しくなります。

戻り値

U*diag(**M**)***V**′が**X**と等しくなるようなリスト{**U**, **M**, **V**}を戻す。

引数

X 行列。

nSingularValues 戻す特異値の数。デフォルトでは、行列Xの行数と列数のいずれか小さい方になります。

nOver オーバーサンプリング列の数。

nIter 検出力の計算の反復回数。

Rank Index(vector)

Rank(vector)

説明

引数のベクトルを昇順に並べ替えるためのインデックスを戻す。結果のベクトルを、元のベクトル（*vector*）の添え字として使用すれば昇順に並べ替えることができる。なお、欠測値は結果から除外されます。行列に加えて、数値または引用符付き文字列のリストがサポートされています。

Ranking(vector)

説明

ベクトル（*vector*）の各要素の大きさの順位を、列ベクトルで戻す。1～*n*の順位が低位から高位に与えられますが、同じ値に対しては任意の順位が与えられます。行列に加えて、数値または引用符付き文字列のリストがサポートされています。

Ranking Tie(vector)

説明

ベクトル（*vector*）の各要素の大きさの順位を、列ベクトルで戻す。ただし、同じ値に対しては、平均順位が与えられます。行列に加えて、数値または引用符付き文字列のリストがサポートされています。

Scoring Impute(rowWithMissing, VMat, colMeanVec, colStdDevVec)

説明

Automated Data Imputation（ADI）アルゴリズムのストリーミング機能を実現する。

戻り値

欠測値を標準最小2乗法で補完した行ベクトルを戻す。

引数

rowwithMissing 欠測値を含む行ベクトル。

VMat ADIアルゴリズムで生成される負荷量行列。

colMeanVec 欠測値のセルを無視した列の平均のベクトル。

colStdDevVec 欠測のセルを無視した列の標準偏差のベクトル。

Shape(A, nrow, <ncol>, < <<bycol>)

説明

行列*A*の全行を指定の次元に再構成する。行列*A*の各値が、変形された行列に入れられます。デフォルトでは、1行ごとに順にデータが埋められます。

戻り値

再構成された行列

引数

A 行列。

nrow 新しい行列の行数。

ncol（オプション）新しい行列の列数。

<<bycol（オプション）値を、1行ごとではなく1列ごとに順にデータを埋めていく。

メモ

列数（*ncol*）が指定されていない場合は、元の行列の値をすべて挿入できるだけの列数が使用されます。

*nrow*に欠測値が指定されている場合、行数は、元の行列のすべての値を収めるのに必要な大きさに設定されます。

新しい行列が元の行列よりも小さい場合は、余った値が破棄されます。

新しい行列が元の行列よりも大きい場合は、新しい行列が完全に埋まるまで値が繰り返されます。

例

```
a = Matrix({ {1, 2, 3}, {4, 5, 6}, {7, 8, 9} });
   [  1 2 3,
      4 5 6,
      7 8 9]

Shape(a, 2);
   [  1 2 3 4 5,
      6 7 8 9 1]

Shape(a, 2, 2);
   [  1 2,
      3 4]

Shape(a, 4, 4);
   [  1 2 3 4,
      5 6 7 8,
      9 1 2 3,
      4 5 6 7]

Shape(a, 4, 4, <<bycol);
   [  1 5 9 4,
      2 6 1 5,
      3 7 2 6,
```

4 8 3 7]

Solve(A, b)

説明

線形システムの解を戻す。この解は、x=inverse(A)*bによって得られる解と同じです。

Sort Ascending(source)

説明

リスト（*source*）の要素を昇順に並べたリストを戻す。

Sort Descending(source)

説明

リスト（*source*）の要素を降順に並べたリストを戻す。

Sparse SVD(X, <nSingularValues=min(nRow, nCol)>, <tolerance=1e-10>)

説明

暗黙的なリスタートと、部分的な再直交化を用いたLanczos法で疎な行列Xの特異値分解（SVD）を行う。

戻り値

U*diag(**M**)***V**′が**X**と等しくなるようなリスト{**U**, **M**, **V**}を戻す。

引数

X 行列。

nSingularValues 戻す特異値の数。デフォルトでは、行列Xの行数と列数のいずれか小さい方になります。

tolerance 収束基準値。

Spline Coef(x, y, lambda)

説明

knots||a||b||c||d形式の5個の要素が含まれた列ベクトルを戻す。ここで、knotsは節点を表し、*x*の一意な値で構成されています。

xは説明変数のベクトル、yは応答変数のベクトル、lambdaは平滑化パラメータです。lambdaの値が大きくなるほどなめらかなスプライン曲線になります。

Spline Eval(x, coef)

説明

Spline Eval関数は、Spline Coef関数から戻される係数の行列（これは、3次スプラインの場合、knots||a||b||c||dという形式の行列です）から、スプライン曲線の予測値を計算します。説明変数を表す引数のxは、スカラーでも行列でもかまいません。引数のcoef行列には、2列以上の任意の列数を設定することができ、各列がそれぞれの次数の係数を表します。xの累乗を計算するときには、節点（knots）の値が引かれます。たとえば、knots||a||b||c||dのcoefを計算する場合、knots[j]がxより小さい最大の節点となるようなjを使います。

xx = x - knots[j]とすると、*x*に対する予測値は次のように計算されます。

```
result = a[j] + xx * (b[j] + xx * (c[j] + xx * d[j]))
```

変形すると、次のようになります。

```
result = a[j] + b[j] * xx + c[j] * xx ^ 2 + d[j] * xx ^ 3
```

Spline Smooth(x, y, lambda)

説明

スプライン曲線をあてはめた結果の予測値(平滑化された値)を戻す。

xは説明変数のベクトル、yは応答変数のベクトル、lambdaは平滑化パラメータです。lambdaの値が大きくなるほどなめらかなスプライン曲線になります。

SVD(A)

説明

特異値分解。

Sweep(A, <indices>)

説明

掃き出し法による行列の計算を行う。

Trace(A)

説明

トレース、つまり正方行列の対角要素の和。

Transpose(A)

説明

行列*A*の行と列を転置する。

戻り値

転置した行列

引数

A 行列。

等価表現

A`

V Concat(A, B, ...)

説明

2つ以上の行列を縦に連結する。

戻り値

行列

引数

2つ以上の行列。

V Concat To(A, B, ...)

説明

2つの行列を縦に連結し、その結果を元の変数に割り当てる。

戻り値

行列

引数

2つ以上の行列。

V Max(matrix)

説明

行列（*matrix*）の各列の最大値を含む行ベクトルを戻す。

V Mean(matrix)

説明

行列（*matrix*）の各列の平均を含む行ベクトルを戻す。

V Median(matrix)

説明

行列（*matrix*）の各列の中央値を含む行ベクトルを戻す。

V Min(matrix)

説明

行列（*matrix*）の各列の最小値を含む行ベクトルを戻す。

V Quantile(matrix, p)

説明

行列（*matrix*）の各列の第*p*分位点を含む行ベクトルを戻す。

V Standardize(matrix)

説明

行列の各列を、平均0、標準偏差1に標準化して戻す。

V Std(matrix)

説明

行列（*matrix*）の各列の標準偏差を含む行ベクトルを戻す。

V Sum(matrix)

説明

行列（*matrix*）の各列の合計を含む行ベクトルを戻す。

Varimax(matrix, <norm=1>)

説明

Varimax回転を実行する。

戻り値

回転後の行列と直交行列を含むリスト

引数

matrix 回転させる行列。

norm 正規化を伴う回転の場合は1、正規化を伴わない回転の場合は0を指定する。デフォルト値は1です。

Vec Diag(A)

説明

正方行列*A*の対角要素から成るベクトルを作成する。

戻り値

行列

引数

A 正方行列。

メモ

正方行列でない行列を指定した場合はエラーが生じます。

Vec Quadratic(symmetric matrix, rectangular matrix)

説明

n × *m*行列を作成する。ハット値などの計算に使われます。

戻り値

行列

引数

2つの行列。最初の行列は対称行列でなければならない。

等価表現

```
Vec Diag(X*Sym*X')
```

VPTree(matrix)

説明

VP木を使ったテーブルを戻す。このテーブルを使うと、近傍点を効率的に探すことができます。横長なデータにはVP木が特に適しています。VP木は、あるランダムな点をVPとして選択し、それ以外の点をVPからの距離に基づいて分割します。VP木は、再帰的に構築され、距離をチェックして部分木を探すことで効率的に探索できます。探索領域の大部分が除外されるため、近傍点検索など、多次元データを扱う場合に便利です。VP木の詳細については、Uhlmann（1991）を参照してください。

戻り値

VP木オブジェクト

引数

`matrix` *k*次元の点の行列。次元数または点数に制限はありません。行列の各列はデータの次元、各行はデータ点を表しています。

メッセージ

`<<K nearest rows(stop, <position>)` 行列を戻す。*<position>*には、*n*個の近傍点を求めたい点を、座標の行ベクトル、もしくは、行番号の整数で指定します。*<position>*が指定されていない場合は、すべての行に関する結果を戻します。*<position>*が指定されている場合は、指定された点または行に対する結果を戻します。引数*stop*には、*n*または{*n*, *limit*}を指定してください。これらの制限に関するパラメータは、結果として戻される行の行数を定義します。数値またはリストで指定できます。数値（たとえば5）を指定した場合、最も近傍にある5行だけが戻されます。また、リスト（たとえば{5, 10}）を指定した場合、距離が10を超えるまで、最大5つの近傍の行が戻されます。*limit*にリストを使用した場合、戻された最後の行の距離は10を超える可能性があります。*stop*の距

離を超えた次の行が見つかるまでコマンドが続行されるため、この最後の点も戻されます。これは、特に *stop* の距離内に行がない場合に役立ちます。

Wavelet Basis Coef(x, grid, coeff, <wavelet = "Harr" | "Biorthogonal" | "Coiflet" | "Daubechies" | "Symlet">, <param = 0>)

説明

指定したウェーブレットモデルのベクトル x の予測値を求める。

戻り値

予測値のベクトル

引数

x データを含む行ベクトルまたは列ベクトル。

grid データのグリッドを指定するベクトル。

coeff ウェーブレット係数のベクトル。

wavelet ウェーブレットモデルの名前。

param ウェーブレットモデルのオプションのパラメータ。

メモ

この関数は、「関数データエクスプローラ」プラットフォームで作成される計算式に使用されています。

Where(<dt>, clause)

説明

指定された節にしたがって項目をフィルタリングし、インデックスを戻す。節には、比較関数または条件文を指定できます。同じ節の中で列・行列・リストを混ぜて照合することもできます。

Where() によって戻されるインデックスは、大量のリスト・行列・列を高速に処理できるよう最適化されています。

必須の引数

clause 比較関数または条件文。

オプションの引数

<dt> 評価時に現在のデータテーブルを変更する。

例

```
Names Default To Here( 1 );
xs = [10 20 30 .50];
xs[Where( xs >= 20 )];
xs[Where( !Is Missing( xs ) )];
ys = {10, 20, "30", ., 50};
ys[Where( ys >= 20 )];
```

数値関数

Abs(n)

説明

*n*の絶対値を計算する。

戻り値

*n*の絶対値

引数

n 任意の数値。

Ceiling(n)

説明

*n*が整数でない場合、*n*を整数に切り上げた値を戻す。

戻り値

*n*以上の整数のうち最小のもの

引数

n 任意の数値。

Derivative(expr, {name, ...}, ...)

説明

*name*に対する *expr*式の導関数を計算する。

戻り値

導関数

引数

expr 任意の式。この引数は、Name Expr、Expr、Evalなどの関数を用いて、間接的に指定してもかまいません。

name 1つの変数または変数のリスト。

メモ

追加の変数をリスト内に指定すると（Derivative(expr, {name}, {name2})）、第2次導関数も求められます。

Floor(n)

説明

*n*が整数でない場合、*n*を整数に切り捨てた値を戻す。

戻り値

*n*以下の整数のうち最大のもの

引数

n　任意の数値。

例

```
Floor( 2.7 );
   2
Floor( -.5 );
   -1
```

Integrate(expr, varname, lowLimit, upLimit, <<Tolerance(1e-10), <<StoreInfo({list}), <<StartingValue(val))

説明

Gander and Gautschi (2000) の数値積分によって、1次元の積分を行います。

引数

expr 被積分関数の式。

varname 積分変数の名前。この変数に値が含まれる場合は、積分の精度を高めるための基準値（初期値）として使用されます。

lowLimit 積分の下限。負の無限大を指定するには、欠測値を指定する。

upLimit 積分の上限。正の無限大を指定するには、欠測値を指定する。

StoreInfo StoreInfo() の引数に数値計算を診断した情報が保存される。

StartingValue 積分の精度を高めるための基準値（初期値）。

Invert Expr(expr, name)

説明

名前（*name*）に関して、式（*expr*）の逆関数を求めることを試みる。

Mod()

「Modulo(number, divisor)」を参照してください。

Modulo(number, divisor)

Mod(number, divisor)

説明

数（*number*）を除数（*divisor*）で割った余りを戻す。

例

```
Modulo( 6, 5 );
```

1

Normal Integrate(muVector, sigmaMatrix, expr, x, nStrata, nSim)

説明

多変量の正規分布に従う変数の滑らかな関数を動径-球法で積分した結果を戻す。

引数

muVector ベクトル。

sigmaMatrix 行列。

expr 変数*x*で表される式。

x 式*expr*に使用される変数。

nStrata 層の数。

nSim シミュレーションの回数。

Num Deriv(f(x,...), <parnum=1>)

説明

関数*f(x,...)*を数値微分により偏微分した結果を戻す。*Num Deriv*関数の第2引数に、偏微分する引数を指定する。第2引数を指定しなかった場合、関数の最初の引数に対する偏微分が求められる。偏微分は*f(x,...)*の引数に指定されている数値の箇所で評価される。

メモ

関数Num Deriv()の結果は、以下の例のように、間違っているように見えることがあります。

```
x = 3;
n = Num Deriv( 3 * x ^ 2 );
    // 9.00000000001455
```

この使用法は正しくありません。この関数は、「非線形回帰」プラットフォームで、解析的な微分が求められない関数を数値微分するために開発されました。この関数の正しい使い方は以下のとおりです。

```
x = 3;
f = Function( {x}, 3 * x ^ 2 );
n = Num Deriv( f( x ), 1 );
   // 18.000029999854
```

Num Deriv2(f(x,...))

説明

関数*f(x,...)*を*x*に関して数値微分により2次微分した結果を戻す。偏微分は*f(x,...)*の引数に指定されている数値の箇所で評価される。

Round(n, places)

説明

*n*を四捨五入して、桁数（*places*）で指定された小数点以下の桁数に丸める。

Simplify Expr(expr(expression))

Simplify Expr(nameExpr(global))

説明

式を代数的に簡素化する。

最適化関数

Constrained Maximize(expr, {x1(low1, up1), x2(low2, up2), ...}, messages)

説明

式 *expr*を最大化する引数 *x*の値を求める。引数はリストで指定すること。各引数に対して下限と上限を括弧で囲んで指定するか、オプションのSet Variable Limit()メッセージを使用します。引数 *x*には、スカラーまたはベクトルを指定できます。

以下のメッセージにおいて、Aは係数からなる行列、x = [x1, x2, ...]は引数のベクトル、bは、式の右辺を成すベクトルとする。

メッセージ

<<Less than EQ({A, b}) 指定された値以下に制約する (A*x <= b)。

<<Greater Than EQ({A, b}) 指定された値以上に制約する (A*x >= b)。

<<Equal To({A, b}) 指定された値に制約する (A*x = b)。

<<Starting Values([x1Start, x2Start, ...]) 初期値を指定する。

<<Max Iter(int) 実行される反復の最大回数を指定する整数。

<<Tolerance(p) *p*は収束基準値。デフォルト値は 10^{-5}。

<<Show Details("true") 結果のリスト（目標値、反復回数、勾配、ヘッセ行列）を戻す。最適化のステップごとの結果をログに出力する。

<<SetVariableLimit({low,high}) 最適化に使用する変数の下限と上限のベクトルを指定する。

Constrained Minimize(expr, {x1(low1, up1), x2(low2, up2), ...}, messages)

説明

式 *expr*を最小化する引数 *x*の値を求める。引数はリストで指定すること。各引数に対して下限と上限を括弧で囲んで指定するか、オプションのSet Variable Limit()メッセージを使用します。引数 *x*には、スカラーまたはベクトルを指定できます。

以下のメッセージにおいて、Aは係数からなる行列、x = [x1, x2, ...]は引数のベクトル、bは、式の右辺を成すベクトルとする。

メッセージ

<<Less than EQ({A, b}) 指定された値以下に制約する (A*x <= b)。

<<Greater Than EQ({A, b}) 指定された値以上に制約する (A*x >= b)。

<<Equal To({A, b}) 指定された値に制約する (A*x = b)。

<<Starting Values([x1Start, x2Start, ...]) 初期値を指定する。

<<Max Iter(int) 実行される反復の最大回数を指定する整数。

<<Tolerance(p) *p*は収束基準値。デフォルト値は10^{-5}。

<<Show Details("true") 結果のリスト（目標値、反復回数、勾配、ヘッセ行列）を戻す。最適化のステップごとの結果をログに出力する。

<<SetVariableLimit({low,high}) 最適化に使用する変数の下限と上限のベクトルを指定する。

Desirability(yVector, desireVector, y)

説明

3点を通過するという条件のもと、応答変数（*y*）の満足度を決定する関数を定義する。*yVector*と*desireVector*は、満足度関数を決定する3点に対応するベクトルです。実際の関数は、満足度値が大きい方が良いか、小さい方が良いか、目標値に合わせるか、または目標値がないかのいずれであるかによって異なります。

戻り値

満足度関数

引数

yVector 3つの入力値。

desireVector 対応する3つの満足度の値。

y 満足度を計算する値。

LPSolve(A, b, c, L, U, neq, nle, nge, <slackVars(Boolean)>)

説明

戻り値のリストの第1要素は、決定変数の値（slackVarsに1が指定された場合にはスラック変数の値も含む）。第2要素は、目的変数の最適値（存在する場合のみ）。

引数

A 制約式の係数を表す行列。

b 制約式の右辺値を列にした行列。

c 目的関数のコスト係数のベクトル。

L, U 下限値と上限値を表す行列。

neq 等号制約式の数。

`nle`「以下」を示す不等号制約式の数。

`nge`「以上」を示す不等号制約式の数。

`slackVars(Boolean)`（オプション）決定変数に加えて、スラック変数も戻すかどうかを指定する。デフォルト値は0です。

メモ

制約式は、等号制約、「以下」を示す不等号制約、「以上」を示す不等号制約の順に指定しなければなりません。

Maximize(expr, {x1(low1, up1), x2(low2, up2), ...}, messages)

説明

式 *expr* を最大化する引数 *x* の値を求める。引数はリストで指定すること。各引数の下限および上限を、括弧内に指定できる。また、反復の最大回数、収束基準値、最適化の詳細表示を指定する引数もある。ヘッセ行列を解析的微分で求められる場合は、Newton-Raphson法が使用される。それ以外の場合は、対称ランクワン法（SR1）に基づく準Newton法が使用される。

メッセージ

`<<Max Iter(int)` 実行される反復の最大回数を指定する整数。デフォルトの最大反復回数は250です。

`<<Tolerance(p)` *p* は収束基準値。デフォルト値は 10^{-8}。

`<<Details("both" | "displaySteps" | "returnDetails")` 戻される出力を指定する。"displaySteps"を指定した場合、最適化のステップごとの結果がログに出力されます。"returnDetails"を指定した場合、関数は、結果のリスト（目標値、反復回数、勾配、ヘッセ行列）を戻します。"both"を指定した場合、結果とログへの出力の両方が取得できます。

`<<Gradient(exprList)` 最適化に使用する勾配を定義する式のリスト。リストに含まれる各式は、式 *expr* を微分したものです。

`<<Hessian(exprList)` 最適化に使用するヘッセ行列を定義する式のリスト。リストに含まれる各式は、ヘッセ行列の上三角部分を1行ずつ順に並べたものです。

`<<Method(NR | SR1)` 最適化をNewton-Raphson（NR）法とSymmetric-Rank One（SR1）法のどちらで行うかを指定する。

`<<UseNumericDeriv("true")` 最適化で数値近似を使用する。

Minimize(expr, {x1(low1, up1), x2(low2, up2), ...}, messages)

説明

式 *expr* を最小化する引数リスト *x* の値を求める。各引数の下限および上限を、括弧内に指定できる。また、反復の最大回数、収束基準値、最適化の詳細表示を指定する引数もある。ヘッセ行列を解析的微分で求められる場合は、Newton-Raphson法が使用される。それ以外の場合は、対称ランクワン法（SR1）に基づく準Newton法が使用される。

メッセージ

`<<Max Iter(int)` 実行される反復の最大回数を指定する整数。デフォルトの最大反復回数は250です。

<<Tolerance(p) *p*は収束基準値。デフォルト値は10^{-8}。

<<Details("both" | "displaySteps" | "returnDetails") 戻される出力を指定する。"displaySteps"を指定した場合、最適化のステップごとの結果がログに出力されます。"returnDetails"を指定した場合、関数は、結果のリスト（目標値、反復回数、勾配、ヘッセ行列）を戻します。"both"を指定した場合、結果とログへの出力の両方が取得できます。

<<Gradient(exprList) 最適化に使用する勾配を定義する式のリスト。リストに含まれる各式は、式*expr*を微分したものです。

<<Hessian(exprList) 最適化に使用するヘッセ行列を定義する式のリスト。リストに含まれる各式は、ヘッセ行列の上三角部分を1行ずつ順に並べたものです。

<<Method(NR | SR1) 最適化をNewton-Raphson（NR）法とSymmetric-Rank One（SR1）法のどちらで行うかを指定する。

<<UseNumericDeriv("true") 最適化で数値近似を使用する。

連続型確率関数

Beta Density(x, alpha, beta, <theta=0>, <sigma=1>)

説明

ベータ分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{B(\alpha, \beta)\sigma^{\alpha+\beta-1}}(x-\theta)^{\alpha-1}(\theta+\sigma-x)^{\beta-1}$$

この式で、B(·)はベータ関数です。

引数

x 密度を求めたい分位点。*x*は、*theta*と*theta* + *sigma*の間にある分位点。

alpha, beta 形状パラメータαおよびβ。両者とも正の値。

theta（オプション）閾値パラメータθ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

メモ

ベータ分布は、分位点xの定義域が無限であるような正規分布やガンマ分布とは異なり、限定された区間に対してだけ正の密度を持ちます。ベータ分布は、割合のような、範囲が0から1までに制限されている確率変数をモデル化する場合に役立ちます。

Beta Distribution(x, alpha, beta, <theta=0>, <sigma=1>)

説明

ベータ分布の*x*の下側累積確率を戻す。パラメータはBeta Density()関数と同じです。

引数

x 下側累積確率を求めたい分位点。*x*は、*theta*と*theta* + *sigma*の間にある分位点。

alpha, beta 形状パラメータαおよびβ。両者とも正の値。

theta（オプション）閾値パラメータθ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Beta Quantile(p, alpha, beta, <theta=0>, <sigma=1>)

説明

指定されたパラメータをもつベータ分布の下側累積確率*p*に対する分位点を戻す。分位点は、閉じた解析的な解としては求めることができません。

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

alpha, beta 形状パラメータαおよびβ。両者とも正の値。

theta（オプション）閾値パラメータθ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Cauchy Density(q, <center=0>, <scale=1>)

説明

Cauchy分布の*q*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(q) = \frac{1}{\pi\sigma}\frac{1}{1+\left(\frac{q-\mu}{\sigma}\right)^2}$$

引数

q 密度を求めたい分位点。

center（オプション）位置パラメータμ。デフォルト値は0。

scale（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Cauchy Distribution(q, <center=0>, <scale=1>)

説明

Cauchy分布に従う変数が*q*以下になる確率を戻す。累積分布関数は、次式のとおりです。

$$F(q) = \frac{1}{2} + \frac{1}{\pi}\arctan\left(\frac{x-\mu}{\sigma}\right)$$

引数

q 下側累積確率を求めたい分位点。

center（オプション）位置パラメータμ。デフォルト値は0。

scale（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Cauchy Quantile(p, <center=0>, <scale=1>)

説明

Cauchy分布の*p*に対する分位点を戻す。この分位点は、Cauchy分布の下側累積確率が*p*となる分位点です。分位点関数は、次式のとおりです。

$$F^{-1}(p) = \sigma \tan\left[\pi\left(p + \frac{1}{2}\right)\right] + \mu$$

引数

p　下側累積確率。*p*は0～1の間でなければなりません。

center（オプション）位置パラメータμ。デフォルト値は0。

scale（オプション）尺度パラメータσ。正の値。デフォルト値は1。

ChiSquare Density(q, df, <nc=0>)

説明

カイ2乗分布の*q*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(q) = \exp(-\lambda/2) \sum_{r=0}^{\infty} \frac{(\lambda/2)^r}{r!} f_{n+2r}(q)$$

この式で、$f_{n+2r}(q)$ は、自由度が$n+2r$のカイ2乗分布の密度です。

引数

q　密度を求めたい分位点。*q*は0以上でなければなりません。

df　自由度*n*。正の値。

nc（オプション）非心度パラメータλ。負でない値。デフォルト値は0。

ChiSquare Distribution(q, df, <nc=0>)

説明

自由度を*df*、オプションの非心度パラメータを*nc*としたカイ2乗分布の、分位点*x*における累積分布関数の値を戻す。累積分布関数は、次式のとおりです。

$$F(q) = \exp(-\lambda/2) \sum_{r=0}^{\infty} \frac{(\lambda/2)^r}{r!} F_{n+2r}(q),$$

この式で、$F_{n+2r}(q)$ は、自由度が$n+2r$のカイ2乗分布の累積分布です。

引数

q　下側累積確率を求めたい分位点。*q*は0以上でなければなりません。

df　自由度*n*。正の値。

nc（オプション）非心度パラメータλ。負でない値。デフォルト値は0。

ChiSquare Log CDistribution(x, df, <nc=0>)

説明

(1 - value)の対数を戻す。ここで、valueは、自由度を*df*、非心度パラメータを*nc*としたカイ2乗分布の*x*における下側累積確率。

ChiSquare Log Density(x, df, <nc=0>)

説明

自由度を*df*、非心度パラメータを*nc*としたカイ2乗分布の*x*における下側累積確率の対数を戻す。

ChiSquare Log Distribution(x, df, <nc=0>)

説明

自由度を*df*、非心度パラメータを*nc*としたカイ2乗分布の*x*における累積分布関数の値の対数を戻す。

ChiSquare Noncentrality(x, df, prob)

説明

以下の式を満たすカイ2乗分布の非心度パラメータ*nc*を戻す。

prob = ChiSquare Distribution(*x, df, nc*)

引数

x　下側累積確率を求めたい分位点。

df　自由度*n*。正の値。

prob　下側累積確率。*prob*は0～1の間でなければなりません。

ChiSquare Quantile(p, df, <nc=0>)

説明

自由度を*df*、オプションの非心度パラメータを*nc*としたカイ2乗分布の、下側累積確率*p*に対する分位点を戻す。分位点は、閉じた解析的な解としては求めることができません。

引数

p　下側累積確率。*p*は0～1の間でなければなりません。

df　自由度*n*。正の値。

nc（オプション）非心度パラメータλ。負でない値。デフォルト値は0。

Dunnett P Value(q, nTrt, dfe, <lambdaVec=.>)

説明

Dunnettの多重比較検定の*p*値を戻す。

引数

q 検定統計量の数値。

nTrt 対照群と比較する処置群の数。

dfe 誤差の自由度。

lambdaVec 相関構造を表すパラメータのベクトル。*lambdaVec*が欠測値（.）の場合、ベクトルのすべての要素は、1/Sqrt(2)に設定されます。

Dunnett Quantile(1-alpha, nTrt, dfe, <lambdaVec=.>)

説明

Dunnettの多重比較検定で使われる分位点を戻す。

引数

1-alpha 信頼水準を示す数値。

nTrt 対照群と比較する処置群の数。

dfe 誤差の自由度。

lambdaVec 相関構造を表すパラメータのベクトル。*lambdaVec*が欠測値（.）の場合、ベクトルのすべての要素は、1/Sqrt(2)に設定されます。

ExGaussian Density(x, mu, sigma, lambda)

説明

指数修正Gauss分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$\lambda \exp\left[\frac{\lambda(2\mu + \lambda\sigma^2 - 2x)}{2}\right]\Phi\left[\frac{x - \mu - \lambda\sigma^2}{\sigma}\right]$$

引数

x 密度を求めたい分位点。

mu 正規分布の平均。

sigma 正規分布の標準偏差。*sigma*は0より大きい値でなければなりません。

lambda 指数分布のλパラメータ。*lambda*は0より大きい値でなければなりません。

メモ： ExGaussian Density関数での指数分布に関するパラメータ表現は、Exponential Density関数で使用されているパラメータの逆数になっています。

ExGaussian Distribution(x, mu, sigma, lambda)

説明

指数修正Gauss分布に従う確率変数が*x*以下になる確率を戻す。

引数

x 下側累積確率を求めたい分位点。

mu 正規分布の平均。

sigma 正規分布の標準偏差。*sigma*は0より大きい値でなければなりません。

lambda 指数分布のλパラメータ。*lambda*は0より大きい値でなければなりません。

メモ: ExGaussian Distribution関数での指数分布に関するパラメータ表現は、Exponential Distribution関数で使用されているパラメータの逆数になっています。

ExGaussian Quantile(p, mu, sigma, lambda)

説明

指数修正Gauss分布の第*p*分位点を戻す。

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

mu 正規分布の平均。

sigma 正規分布の標準偏差。*sigma*は0より大きい値でなければなりません。

lambda 指数分布のλパラメータ。*lambda*は0より大きい値でなければなりません。

メモ: ExGaussian Quantile関数での指数分布に関するパラメータ表現は、Exponential Quantile関数で使用されているパラメータの逆数になっています。

Exp Density(x, <theta=1>)

Exponential Density(x, <theta=1>)

説明

指数分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{\theta}\exp(-x/\theta)$$

引数

x 密度を求めたい分位点。*x*は0以上でなければなりません。

theta（オプション）尺度パラメータθ。正の値。デフォルト値は1。

Exp Distribution(x, <theta=1>)

Exponential Distribution(x, <theta=1>)

説明

指数分布に従う確率変数が*x*以下になる確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = 1 - \exp(-x/\theta)$$

引数

x　下側累積確率を求めたい分位点。*x*は0以上でなければなりません。

theta（オプション）尺度パラメータθ。正の値。デフォルト値は1。

Exp Quantile(p, <theta=1>)

Exponential Quantile(p, <theta=1>)

説明

尺度パラメータ*theta*を持つ指数分布の下側累積確率が*p*となる分位点を戻す。分位点関数は、次式のとおりです。

$$F^{-1}(p) = -\theta\log(1-p)$$

引数

p　下側累積確率。*p*は0～1の間でなければなりません。

theta（オプション）尺度パラメータθ。正の値。デフォルト値は1。

F Density(x, dfnum, dfden, <nc=0>)

説明

分子の自由度を*dfnum*、分母の自由度を*dfden*、オプションの非心度パラメータを*nc*としたF分布の*x*における密度を戻す。

$$f(x) = \exp(-\lambda/2)\sum_{r=0}^{\infty}\frac{(\lambda/2)^r}{\mathrm{B}\left(\frac{v_2}{2},\frac{v_1}{2}+r\right)r!}\left(\frac{v_1}{v_2}\right)^{\frac{v_1}{2}+r}\left(1+\frac{v_1}{v_2}x\right)^{-\left(\frac{v_1+v_2}{2}+r\right)}x^{\frac{v_1}{2}-1+r}$$

この式で、B(·)はベータ関数です。

引数

x　密度を求めたい分位点。*x*は正の値でなければなりません。

dfnum　*F*分布の分子に使用されるカイ2乗分布の自由度、v_1。*dfnum*は、0より大きくなければなりません。

dfden　*F*分布の分母に使用されるカイ2乗分布の自由度、v_2。*dfden*は、0より大きくなければなりません。

nc（オプション）非心度パラメータλ。負でない値。デフォルト値は0。

F Distribution(x, dfnum, dfden, <nc=0>)

説明

分子の自由度を *dfnum*、分母の自由度を *dfden*、オプションの非心度パラメータを *nc* としたF分布の *x* における下側累積確率を戻す。

F Log CDistribution(x, dfnum, dfden, <nc=0>)

説明

(1 - value) の対数を戻す。ここで、valueは、分子の自由度を *dfnum*、分母の自由度を *dfden*、オプションの非心度パラメータを *nc* としたF分布の *x* における下側累積確率。

F Log Density(x, dfnum, dfden, <nc=0>)

説明

分子の自由度を *dfnum*、分母の自由度を *dfden*、オプションの非心度パラメータを *nc* としたF分布の *x* における確率密度関数の値の対数を戻す。

F Log Distribution(x, dfnum, dfden, <nc=0>)

説明

分子の自由度を *dfnum*、分母の自由度を *dfden*、オプションの非心度パラメータを *nc* としたF分布の *x* における累積分布関数の値の対数を戻す。

F Noncentrality(x, dfnum, dfden, prob)

説明

以下の式を満たすF分布の非心度パラメータ *n* を戻す。

prob = F Distribution(*x, dfnum, dfden, nc*)

次も参照

「F Distribution(x, dfnum, dfden, <nc=0>)」

F Power(alpha, dfh, dfm, d, n)

説明

与えられた引数から算出される *F* 検定や *t* 検定の検出力を戻す。

引数

alpha 検定の有意水準。*alpha* は、0～1の間でなければなりません。

dfh 仮説の自由度。*dfh* は、0より大きくなければなりません。

dfm モデル全体の自由度。*dfm*は、0より大きくなければなりません。

d 効果の大きさの2乗。Δ^2/σ^2と定義される。この式で、σ^2は誤差分散、Δ^2は次のように定義されます。

$\Delta^2 = (\bar{x} - \mu)^2$ 1標本の*t*検定

$\Delta^2 = \frac{(\bar{x}_1 - \bar{x}_2)^2}{4}$ 2標本の*t*検定

$\Delta^2 = \sqrt{\sum_{i=1}^{k} \frac{(\bar{x}_i - \bar{x})^2}{k}}$ *k*標本の*F*検定

n 観測（オブザベーション）の合計個数。*n*は、*dfm*より大きくなければなりません。

F Quantile(x, dfnum, dfden, <nc=0>)

説明

分子の自由度を*dfnum*、分母の自由度を*dfden*、オプションの非心度パラメータを*nc*としたF分布の下側累積確率*p*に対する分位点を戻す。

F Sample Size(alpha, dfh, dfm, d, power)

説明

与えられた引数から算出される*F*検定や*t*検定の標本サイズを戻す。

引数

alpha 検定の有意水準。*alpha*は、0～1の間でなければなりません。

dfh 仮説の自由度。*dfh*は、0より大きくなければなりません。

dfm モデル全体の自由度。*dfm*は、0より大きくなければなりません。

d 効果の大きさの2乗。Δ^2/σ^2と定義される。この式で、σ^2は誤差分散、Δ^2は次のように定義されます。

$\Delta^2 = (\bar{x} - \mu)^2$ 1標本の*t*検定

$\Delta^2 = \frac{(\bar{x}_1 - \bar{x}_2)^2}{4}$ 2標本の*t*検定

$\Delta^2 = \sqrt{\sum_{i=1}^{k} \frac{(\bar{x}_i - \bar{x})^2}{k}}$ *k*標本の*F*検定

power 検定の検出力。

Frechet Density(x, mu, sigma)

説明

Fréchet分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \exp\left[-\exp\left(-\frac{\log(x)-\mu}{\sigma}\right)\right]\exp\left(-\frac{\log(x)-\mu}{\sigma}\right)\frac{1}{x\sigma}$$

引数

x 密度を求めたい分位点。*x*は正の値でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Frechet Distribution(x, mu, sigma)

説明

Fréchet分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \exp\left[-\exp\left(-\frac{\log(x)-\mu}{\sigma}\right)\right]$$

引数

x 下側累積確率を求めたい分位点。*x*は正の値でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Frechet Quantile(p, mu, sigma)

説明

位置*mu*、尺度*sigma*のFréchet分布の、下側累積確率*p*に対する分位点を戻す。分位点関数は、次式のとおりです。

$$F^{-1}(p) = \exp[-\sigma\log\{-\log(p)\}+\mu]$$

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Gamma Density(x, <alpha=1>, <scale=1>, <threshold=0>)

説明

ガンマ分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}}(x-\theta)^{\alpha-1}\exp(-(x-\theta)/\beta)$$

引数

x 密度を求めたい分位点。*x*はθより大きくなければなりません。

alpha（オプション）形状パラメータα。正の値。デフォルト値は1。

scale（オプション）尺度パラメータβ。正の値。デフォルト値は1。

threshold（オプション）閾値パラメータθ。デフォルト値は0。

Gamma Distribution(x, <alpha=1>, <scale=1>, <threshold=0>)

IGamma(x, <alpha=1, scale=1, threshold=0>)

説明

形状パラメータ（*alpha*）、尺度パラメータ（*scale*）、閾値パラメータ（*threshold*）を持つガンマ分布の、分位点*x*における累積分布関数の値を戻す。

Gamma Log CDistribution(x, <alpha=1>, <scale=1>, <threshold=0>)

説明

計算できる範囲がはるかに大きいことを除けば、Log(1 - Gamma Distribution(x, alpha))と同じ。

Gamma Log Density(x, <alpha=1>, <scale=1>, <threshold=0>)

説明

計算できる範囲がはるかに大きいことを除けば、Log(GammaDensity(x, alpha))と同じ。

Gamma Log Distribution(x, <alpha=1>, <scale=1>, <threshold=0>)

説明

計算できる範囲がはるかに大きいことを除けば、Log(Gamma Distribution(x, alpha))と同じ。

Gamma Quantile(p, <alpha=1>, <scale=1>, threshold>)

説明

形状パラメータ（*alpha*）、尺度パラメータ（*scale*）、閾値（*threshold*）を持つガンマ分布の、下側累積確率*p*に対する分位点を戻す。

GenGamma Density(x, mu, sigma, lambda)

説明

拡張一般化ガンマ分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \begin{cases} \frac{|\lambda|}{x\sigma}\phi_{lg}[\lambda\omega + \log(\lambda^{-2}); \quad \lambda^{-2}] & \lambda \neq 0 \text{ の場合} \\ \frac{1}{x\sigma}\phi_{nor}(\omega) & \lambda = 0 \text{ の場合} \end{cases}$$

この式で、$\omega = [\log(x) - \mu]/\sigma$。次の式は、0より大きい形状パラメータ$\kappa$を持つ標準化した対数ガンマ分布の確率密度関数です。

$$\phi_{lg}(z;\kappa) = \frac{1}{\Gamma(\kappa)}\exp[\kappa z - \exp(z)]$$

$\phi_{nor}(\cdot)$は、標準正規分布の確率密度関数です。

引数

x 密度を求めたい分位点。*x*は正の値でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

lambda 形状パラメータλ。

GenGamma Distribution(x, mu, sigma, lambda)

説明

拡張一般化ガンマ分布の下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \begin{cases} \Phi_{lg}[\lambda\omega + \log(\lambda^{-2}); \quad \lambda^{-2}] & \lambda > 0 \text{ の場合} \\ \Phi_{nor}(\omega) & \lambda = 0 \text{ の場合} \\ 1 - \Phi_{lg}[\lambda\omega + \log(\lambda^{-2}); \quad \lambda^{-2}] & \lambda < 0 \text{ の場合} \end{cases}$$

この式で、$\omega = [\log(x) - \mu]/\sigma$。次の式は、0より大きい形状パラメータ$\kappa$を持つ標準化した対数ガンマ分布の累積分布関数です。

$$\Phi_{lg}(z;\kappa) = \Gamma_I[\exp(z);\kappa]$$

この式で、$\Gamma_I[\cdot]$は、不完全ガンマ関数を表します。$\Phi_{nor}(\cdot)$は、標準正規分布の累積分布関数です。

引数

x 下側累積確率を求めたい分位点。*x*は正の値でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

lambda 形状パラメータλ。

GenGamma Quantile(p, mu, sigma, lambda)

説明

パラメータ *mu*、*sigma*、*lambda* の拡張一般化ガンマ分布の下側累積確率が *p* となる分位点を戻す。分位点は、閉じた解析的な解としては求めることができません。

引数

p 下側累積確率。*p* は0～1の間でなければなりません。

mu 位置パラメータ μ。

sigma 尺度パラメータ σ。正の値。

lambda 形状パラメータ λ。

GLog Density(x, mu, sigma, lambda)

説明

一般化対数分布の *x* における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \phi\left\{\frac{1}{\sigma}\left[\log\left(\frac{x+\sqrt{x^2+\lambda^2}}{2}\right)-\mu\right]\right\}\frac{x+\sqrt{x^2+\lambda^2}}{\sigma(x^2+\lambda^2+x\sqrt{x^2+\lambda^2})}$$

上の式で、$\phi(\cdot)$ は標準正規分布の確率密度関数です。

引数

x 密度を求めたい分位点。

mu 位置パラメータ μ。

sigma 尺度パラメータ σ。正の値。

lambda 形状パラメータ λ。正の値。

メモ

この分布は、パラメータ lambda が0の場合、Lognormal(μ, σ) の対数正規分布になります。

GLog Distribution(x, mu, sigma, lambda)

説明

一般化対数分布の累積分布関数。一般化対数分布に従う確率変数が *x* 以下になる確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \Phi\left\{\frac{1}{\sigma}\left[\log\left(\frac{x+\sqrt{x^2+\lambda^2}}{2}\right)-\mu\right]\right\}$$

上の式で、$\Phi(\cdot)$ は標準正規分布の累積分布関数です。

引数

x 下側累積確率を求めたい分位点。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

lambda 形状パラメータλ。正の値。

GLog Quantile(p, mu, sigma, lambda)

説明

一般化対数分布の下側累積確率が*p*となる分位点を戻す。

IGamma()

「Gamma Distribution(x, <alpha=1>, <scale=1>, <threshold=0>)」を参照してください。

Johnson Sb Density(q, gamma, delta, theta, sigma)

説明

Johnson Sb分布の*q*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(q) = \phi\left[\gamma + \delta \ln\left(\frac{q-\theta}{\sigma-(q-\theta)}\right)\right]\left(\frac{\delta\sigma}{(q-\theta)(\sigma-(q-\theta))}\right)$$

上の式で、$\phi(\cdot)$は標準正規分布の確率密度関数です。

引数

q 密度を求めたい分位点。*q*は、*theta*～*theta*+*sigma*の区間内でなければなりません。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Johnson Sb Distribution(q, gamma, delta, theta, sigma)

説明

Johnson Sb分布の*q*における下側累積確率を戻す。確率密度関数は、次式のとおりです。

$$F(q) = \Phi\left[\gamma + \delta \ln\left(\frac{q-\theta}{\sigma-(q-\theta)}\right)\right]$$

上の式で、$\Phi(\cdot)$は標準正規分布の累積分布関数です。

引数

q 下側累積確率を求めたい分位点。*q*は、*theta*～*theta*+*sigma*の区間内でなければなりません。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Johnson Sb Quantile(p, gamma, delta, theta, sigma)

説明

Johnson Sb分布の下側累積確率が*p*となる分位点を戻す。

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Johnson Sl Density(x, gamma, delta, theta, sigma)

説明

Johnson Sl分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{\delta}{|x-\theta|}\phi\left[\gamma + \delta \ln\left(\frac{x-\theta}{\sigma}\right)\right]$$

上の式で、$\phi(\cdot)$は標準正規分布の確率密度関数です。

引数

x 密度を求めたい分位点。*x*は、*sigma*が1のときは*theta*より大きい値、*sigma*が-1のときは*theta*より小さい値でなければなりません。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 分布が正の方向に歪むか、負の方向に歪むかを示すパラメータσ。sigmaは、+1（正の方向）または-1（負の方向）のどちらかをとります。

Johnson Sl Distribution(q, gamma, delta, theta, sigma)

説明

Johnson Sl分布の*q*における下側累積確率を戻す。

$$F(x) = \begin{cases} \Phi\left[\gamma + \delta \ln\left(\frac{x-\theta}{\sigma}\right)\right], & \sigma = 1 \\ 1 - \Phi\left[\gamma + \delta \ln\left(\frac{x-\theta}{\sigma}\right)\right], & \sigma = -1 \end{cases}$$

上の式で、$\Phi(\cdot)$は標準正規分布の累積分布関数です。

引数

q 下側累積確率を求めたい分位点。*q*は、*sigma*が1のときは*theta*より大きい値、*sigma*が-1のときは*theta*より小さい値でなければなりません。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 分布が正の方向に歪むか、負の方向に歪むかを定義するパラメータσ。Sigmaは、+1（正の方向）または-1（負の方向）のどちらかをとります。

Johnson Sl Quantile(p, gamma, delta, theta, sigma)

説明

Johnson Sl分布の下側累積確率が*p*となる分位点を戻す。

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 分布が正の方向に歪むか、負の方向に歪むかを定義するパラメータσ。Sigmaは、+1（正の方向）または-1（負の方向）のどちらかをとります。

Johnson Su Density(x, gamma, delta, theta, sigma)

説明

Johnson Su分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{\delta}{\sigma}\left[1 + \left(\frac{x-\theta}{\sigma}\right)^2\right]^{-1/2} \phi\left[\gamma + \delta \sinh^{-1}\left(\frac{x-\theta}{\sigma}\right)\right]$$

上の式で、$\phi(\cdot)$は標準正規分布の確率密度関数です。

引数

x 密度を求めたい分位点。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Johnson Su Distribution(q, gamma, delta, theta, sigma)

説明

Johnson Su分布の*q*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \Phi\left[\gamma + \delta \sinh^{-1}\left(\frac{x-\theta}{\sigma}\right)\right]$$

上の式で、Φ(·)は標準正規分布の累積分布関数です。

引数

q 下側累積確率を求めたい分位点。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Johnson Su Quantile(p, gamma, delta, theta, sigma)

説明

Johnson Su分布の下側累積確率が*p*となる分位点を戻す。

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

LEV Density(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の最大極値分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{\sigma}\exp\left[-\frac{x-\mu}{\sigma} - \exp\left(-\frac{x-\mu}{\sigma}\right)\right]$$

引数

x 密度を求めたい分位点。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

LEV Distribution(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の最大極値分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \exp\left[-\exp\left(-\frac{x-\mu}{\sigma}\right)\right]$$

引数

x 下側累積確率を求めたい分位点。*x*は、*sigma*より大きくなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

LEV Quantile(p, mu, sigma)

説明

位置*mu*、尺度*sigma*の最大極値分布の下側累積確率が*p*となる分位点を戻す。分位点関数は、次式のとおりです。

$$F^{-1}(p) = -\sigma\log(-\log(p)) + \mu$$

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

LogGenGamma Density(x, mu, sigma, lambda)

説明

パラメータが*mu*、*sigma*、*lambda*の対数一般化ガンマ確率分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \begin{cases} \frac{|\lambda|}{\sigma}\phi_{\lg}[\lambda\omega + \log(\lambda^{-2});\quad \lambda^{-2}] & \lambda \neq 0 \text{ の場合} \\ \frac{1}{\sigma}\phi_{\mathrm{nor}}(\omega) & \lambda = 0 \text{ の場合} \end{cases}$$

この式で、$\omega = [x - \mu]/\sigma$です。次の式は、0より大きい形状パラメータκを持つ対数ガンマ分布の確率密度関数です。

$$\phi_{lg}(z;\kappa) = \frac{1}{\Gamma(\kappa)}\exp[\kappa z - \exp(z)]$$

$\phi_{nor}(\cdot)$は、標準正規分布の確率密度関数です。

引数

x 密度を求めたい分位点。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

lambda 形状パラメータλ。

LogGenGamma Distribution(x, mu, sigma, lambda)

説明

パラメータが*mu*、*sigma*、*lambda*の対数一般化ガンマ分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \begin{cases} \Phi_{lg}[\lambda\omega + \log(\lambda^{-2});\ \lambda^{-2}] & \lambda > 0 \text{ の場合} \\ \Phi_{nor}(\omega) & \lambda = 0 \text{ の場合} \\ 1 - \Phi_{lg}[\lambda\omega + \log(\lambda^{-2});\ \lambda^{-2}] & \lambda < 0 \text{ の場合} \end{cases}$$

この式で、$\omega = [x - \mu]/\sigma$です。次の式は、0より大きい形状パラメータκを持つ対数ガンマ分布の累積分布関数です。

$$\Phi_{lg}(z;\kappa) = \Gamma_I[\exp(z);\kappa]$$

この式で、$\Gamma_I[\cdot]$は、不完全ガンマ関数を表します。$\Phi_{nor}(\cdot)$は、標準正規分布の累積分布関数です。

引数

x 下側累積確率を求めたい分位点。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

lambda 形状パラメータλ。

LogGenGamma Quantile(p, mu, sigma, lambda)

説明

対数一般化ガンマ分布の第*p*分位点を戻す。

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

lambda 形状パラメータλ。

Logistic Density(x, mu, sigma)

説明

位置*mu*、尺度*sigma*のロジスティック分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{\sigma}\frac{\exp\left(-\frac{x-\mu}{\sigma}\right)}{\left[1+\exp\left(-\frac{x-\mu}{\sigma}\right)\right]^2}$$

引数

x 密度を求めたい分位点。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Logistic Distribution(x, mu, sigma)

説明

位置*mu*、尺度*sigma*のロジスティック分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \frac{1}{\left[1+\exp\left(-\frac{x-\mu}{\sigma}\right)\right]}$$

引数

x 下側累積確率を求めたい分位点。*x*は、σより大きくなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Logistic Quantile(p, mu, sigma)

説明

位置*mu*、尺度*sigma*のロジスティック分布の下側累積確率が*p*となる分位点を戻す。分位点関数は、次式のとおりです。

$$F^{-1}(p) = -\sigma\log\left(\frac{1}{p}-1\right)+\mu$$

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Loglogistic Density(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の対数ロジスティック分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{x\sigma}\frac{\exp\left(\frac{\log(x)-\mu}{\sigma}\right)}{\left[1+\exp\left(\frac{\log(x)-\mu}{\sigma}\right)\right]^2}$$

引数

x 密度を求めたい分位点。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Loglogistic Distribution(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の対数ロジスティック分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \frac{1}{1+\exp\left(-\frac{\log(x)-\mu}{\sigma}\right)}$$

引数

x 下側累積確率を求めたい分位点。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Loglogistic Quantile(p, mu, sigma)

説明

位置*mu*、尺度*sigma*の対数ロジスティック分布の下側累積確率が*p*となる分位点を戻す。分位点関数は、次式のとおりです。

$$F^{-1}(p) = \exp\left[-\sigma\log\left(\frac{1}{p}-1\right)+\mu\right]$$

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Lognormal Density(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の対数正規分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{x}\phi\left[\frac{\log(x)-\mu}{\sigma}\right]$$

上の式で、$\phi(\cdot)$は標準正規分布の確率密度関数です。

引数

x 密度を求めたい分位点。*x*は0以上でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Lognormal Distribution(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の対数正規分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \Phi\left[\frac{\log(x)-\mu}{\sigma}\right]$$

上の式で、$\Phi(\cdot)$は標準正規分布の累積分布関数です。

引数

x 密度を求めたい分位点。*x*は0以上でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

Lognormal Quantile(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の対数正規分布の下側累積確率が*p*となる分位点を戻す。

Normal Biv Distribution(x, y, r, <mu1>, <s1>, <mu2>, <s2>)

説明

2変量正規分布に従う確率変数(X, Y)が(*x, y*)以下になる確率を計算する。ここで、Xの周辺分布は平均*mu1*、標準偏差*s1*、Yの周辺分布は平均*mu2*、標準偏差*s2*の正規分布に従っているとします。また、2変量の相関関数は*r*とします。*mu1*、*s1*、*mu2*、*s2*が指定されていない場合、*mu1*=0、*s1*=1、*mu2*=0、*s2*=1の標準正規分布が使われます。

Normal Density(x, <mean=0>, <stddev=1>)

説明

平均（*mean*）、標準偏差（*stddev*）を持つ正規分布の、*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

引数

x 密度を求めたい分位点。

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

メモ

正規分布は、西洋の教会にあるベルの形状をした、左右対称な分布です。

Normal Distribution(x, <mean=0>, <stddev=1>)

説明

平均（*mean*）、標準偏差（*stddev*）を持つ正規分布の、*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$$

$\Phi(\cdot)$は標準正規分布の累積分布関数で、次のように定義されます。

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_0^x \exp\left(-\frac{t^2}{2}\right)dt$$

引数

x 密度を求めたい分位点。

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Normal Log CDistribution(x, <mean=0>, <std dev=1>)

説明

正規分布の分位点*x*での上側累積確率の対数を戻す。

Normal Log Density(x, <mean=0>, <stddev=1>)

説明

平均（*mean*）、標準偏差（*stddev*）を持つ正規分布の、分位点*x*における密度関数の値の対数を戻す。デフォルトの平均（*mean*）は0です。デフォルトの標準偏差（*stddev*）は1です。

Normal Log Distribution(x, <mean=0>, <std dev=1>)

説明

正規分布の分位点*x*での累積分布関数の値の対数を戻す。

Normal Mixture Density(q, mean, stdev, probability)

説明

正規混合分布の密度関数。グループ平均*mean*、グループ標準偏差*stdev*、グループ確率*probability*の*q*における密度を戻す。*mean*、*stdev*、および*probability*はすべて同じサイズのベクトルです。

Normal Mixture Distribution(q, mean, stdev, probability)

説明

正規混合分布の確率関数。グループ平均*mean*、グループ標準偏差*stdev*、グループ確率*probability*の正規混合分布に従う確率変数が*q*よりも小さい確率を戻す。*mean*、*stdev*、および*probability*はすべて同じサイズのベクトルです。

Normal Mixture Quantile(p, mean, stdev, probability)

説明

正規混合分布の下側累積確率が*p*となる分位点を戻す。*mean*、*stdev*、および*probability*はすべて同じサイズのベクトルです。

Normal Quantile(p, <mean=0>, <stddev=1>)

Probit(p, <mean=0>, <stddev=1>)

説明

平均（*mean*）、標準偏差（*stddev*）をもつ正規分布の下側累積確率が*p*となる分位点を戻す。デフォルトの平均（*mean*）は0です。デフォルトの標準偏差（*stddev*）は1です。

Probit()

「Normal Quantile(p, <mean=0>, <stddev=1>)」を参照してください。

SEV Density(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の最小極値分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{1}{\sigma}\exp\left[\frac{x-\mu}{\sigma} - \exp\left(\frac{x-\mu}{\sigma}\right)\right]$$

引数

x 密度を求めたい分位点。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

SEV Distribution(x, mu, sigma)

説明

位置*mu*、尺度*sigma*の最小極値分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = 1 - \exp\left[-\exp\left(\frac{x-\mu}{\sigma}\right)\right]$$

引数

x 下側累積確率を求めたい分位点。*x*は、σより大きくなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

SEV Quantile(p, mu, sigma)

説明

位置*mu*、尺度*sigma*の最小極値分布の下側累積確率が*p*となる分位点を戻す。分位点関数は、次式のとおりです。

$$F^{-1}(p) = \sigma\log[-\log(1-p)] + \mu$$

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

SHASH Density(x, gamma, delta, theta, sigma)

説明

sinh-arcsinh（SHASH）分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{\delta \cosh(w)}{\sqrt{\sigma^2 + (x-\theta)^2}} \phi[\sinh(w)]$$

ここで、

$\phi(\cdot)$は標準正規分布の確率密度関数です。

$$w = \gamma + \delta \sinh^{-1}\left(\frac{x-\theta}{\sigma}\right)$$

引数

x 密度を求めたい分位点。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

SHASH Distribution(x, gamma, delta, theta, sigma)

説明

sinh-arcsinh（SHASH）分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = \Phi\left[\sinh\left(\gamma + \delta \sinh^{-1}\left(\frac{x-\theta}{\sigma}\right)\right)\right]$$

上の式で、$\Phi(\cdot)$は標準正規分布の累積分布関数です。

引数

x 下側累積確率を求めたい分位点。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

SHASH Quantile(p, gamma, delta, theta, sigma)

説明

sinh-arcsinh（SHASH）分布の下側累積確率が*p*となる分位点を戻す。分布のパラメータは、*gamma*、*delta*、*theta*、*sigma*です。

引数

p 下側累積確率。*p*は0～1の間でなければなりません。

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Students t Density()

「t Density(x, df, <nc=0>)」を参照してください。

Students t Distribution()

「t Distribution(q, df, <nc=0>)」を参照してください。

Students t Quantile()

「t Quantile(p, df, <nc=0>)」を参照してください。

t Density(x, df, <nc=0>)

Students t Density(x, df, <nc=0>)

説明

指定された自由度（*df*）をもつStudentの*t*分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\nu\pi}} \left[1 + \frac{x^2}{\nu}\right]^{-\left(\frac{\nu+1}{2}\right)}$$

引数

x 密度を求めたい分位点。

df 自由度ν。値は1以上でなければなりません。

nc（オプション）非心度パラメータλ。負でない値。デフォルト値は0。

t Distribution(q, df, <nc=0>)

Students t Distribution(q, df, <nc=0>)

説明

Studentの*t*分布の累積分布関数。Studentの*t*分布に従う確率変数が*q*以下になる確率を戻します。*nc*のデフォルト値は0です。

t Log CDistribution(x, df, <nc=0>)

説明

*t*分布の分位点*x*での上側累積確率の対数を戻す。

t Log Density(x, df, <nc=0>)

説明

*t*分布の分位点*x*での密度関数の値の対数を戻す。

t Log Distribution(x, df, <nc=0>)

説明

*t*分布の分位点*x*での累積分布関数の値の対数を戻す。

t Noncentrality(x, df, prob)

説明

以下の式を満たす*t*分布の非心度パラメータ*nc*を戻す。

prob = T Distribution(*x, df, nc*)

t Quantile(p, df, <nc=0>)

Students t Quantile(p, df, <nc=0>)

説明

Studentの*t*分布の分位点関数。指定された自由度（*df*）をもつStudentの*t*分布の、下側累積確率が*p*となる分位点を戻す。*nc*のデフォルト値は0です。

Tukey HSD P Value(q, n, dfe)

説明

TukeyのHSD多重比較検定の*p*値を戻す。

引数

q 検定統計量。多重性の調整をされたTukey検定の棄却値であり、スチューデント化範囲の分位点を2の平方根で割った値です。

n 検定されるグループの数。

dfe すべての標本から計算される誤差の自由度。

Tukey HSD Quantile(1-alpha, n, dfe)

説明

TukeyのHSD多重比較検定に使用される分位点を戻す。戻される分位点は、多重性の調整をされたTukey検定の棄却値であり、スチューデント化範囲の分位点を2の平方根で割った値です。

引数

1-alpha 信頼水準。

n 検定されるグループの数。

dfe すべての標本から計算される誤差の自由度。

Weibull Density(x, shape, <scale=1>, <threshold=0>)

説明

Weibull分布の*x*における密度を戻す。確率密度関数は、次式のとおりです。

$$f(x) = \frac{\beta}{\alpha}\left(\frac{x-\theta}{\alpha}\right)^{\beta-1}\exp\left[-\left(\frac{x-\theta}{\alpha}\right)^{\beta}\right]$$

引数

x 密度を求めたい分位点。*x*は、*threshold*より大きくなければなりません。

shape 形状パラメータβ。正の値。

scale（オプション）尺度パラメータα。正の値。デフォルト値は1。

threshold（オプション）閾値パラメータθ。デフォルト値は0。

Weibull Distribution(x, shape, <scale=1>, <threshold=0>)

説明

Weibull分布の*x*における下側累積確率を戻す。累積分布関数は、次式のとおりです。

$$F(x) = 1 - \exp\left[-\left(\frac{x-\theta}{\alpha}\right)^{\beta}\right]$$

引数

x 密度を求めたい分位点。*x*は、*threshold*より大きくなければなりません。

shape 形状パラメータβ。正の値。

scale（オプション）尺度パラメータα。正の値。デフォルト値は1。

threshold（オプション）閾値パラメータθ。デフォルト値は0。

Weibull Quantile(p, shape, <scale=1>, <threshold=0>)

説明

指定されたパラメータをもつWeibull分布の下側累積確率が*p*となる分位点を戻す。分位点関数は、次のように計算されます。

$$F^{-1}(p) = \alpha[\ln(1-p)]^{\frac{1}{\beta}} + \theta$$

引数

p　下側累積確率。*p*は0～1の間でなければなりません。

shape　形状パラメータβ。正の値。

scale（オプション）尺度パラメータα。正の値。デフォルト値は1。

threshold（オプション）閾値パラメータθ。デフォルト値は0。

プログラミング関数

As Boolean(x)

説明

JSL式を評価し、ブール値を戻す。

例

```
x = 45;
b = As Boolean( x > 2 );
Show( b );
   b = true;
```

As C Expr(x)

説明

式をC言語の形式に変換する。

戻り値

引用符付き文字列

As Column(name)

As Column(dt, name)

:name

dt:name

説明

このスコープ演算子により、グローバル変数ではなく、現在のデータテーブル（または、オプションのデータテーブル参照引数 *dt* で指定したテーブル）内のデータテーブル列として変数（*name*）が評価される。

引数

name 変数名。

dt データテーブル参照。

メモ

:name は、現在のデータテーブルの列名を参照します。dt:name を使用して参照先データテーブルを指定することもできます。

As Constant(expr)

説明

計算後に変化しない値を生成するために、式を一度評価する。

戻り値

評価の結果

引数

expr 任意のJSL式。

メモ

データテーブルに予測式の列を保存できるいくつかのプラットフォームでは、As Constant() が使用されています。この関数は、計算式の一部がすべての行について一定である場合に挿入されます。1行目について引数を評価し、後続の行については、再評価せずに同じ結果を使用します。

As Global(name)

:name

説明

このスコープ演算子により、データテーブル列ではなく、グローバル変数として変数（*name*）が評価される。

引数

name 変数名。

As JavaScript Expr(x)

説明

式をJavaScriptの形式に変換する。

戻り値

引用符付き文字列

As JSON Expr(x)

説明

式をJSON（JavaScript Object Notation）の形式に変換する。

戻り値

引用符付き文字列

As List(matrix)

「As List(matrix)」を参照してください。

As Name(*string*)

説明

引数を引用符付き文字列として評価し、それを名前に変更する。

戻り値

名前

As Namespace(name)

説明

指定された名前空間にアクセスする。そのような名前空間が存在しない場合はエラーが戻されます。

戻り値

名前空間

引数

name 定義された引用符なしの名前空間。

As Python Expr(x)

説明

式をPythonの形式に変換する。

戻り値

引用符付き文字列

As SAS Expr(x)

説明

式を、SASのDATAステップで使用できる形式に変換し、文字列で戻す。コードは、PROC DS2のコールでラップする必要があります。リテラルな式を指定する場合には、Expr(...)で、その式を囲んでください。また、式が変数name内に格納されている場合には、NameExpr(name)と指定してください。このように指定しないと、式が評価された後の結果が渡されます。

戻り値

引用符付き文字列

As Scoped(namespace, variable)

namespace:variable

説明

指定の名前空間*namespace*にある指定の変数*variable*にアクセスする。

戻り値

変数の値、ただしスコープ変数が見つからない場合はエラー

引数

namespace 定義された名前空間の名前。

variable 名前空間*namespace*内の定義された変数。

Associative Array({key, value}, ...)

Associative Array(keys, values)

説明

連想配列（「辞書」または「ハッシュマップ」ともいう）を作成する。

戻り値

連想配列オブジェクト

引数

キー（key）と値（value）をペアとしたリスト。または、キーと値それぞれに、リスト、行列、またはデータテーブル列を指定することもできます。

Class Exists(class)

説明

指定されたクラスが、定義されているかどうかを示す値を戻す。

戻り値

0または1

引数

class 定義されたクラスを表す引用符付き文字列、または、インスタンスが作成されたクラスオブジェクトへの参照名。

Clear Globals(<name>, <name>, ...)

説明

すべてのグローバルシンボルの値をクリアする。グローバル以外の任意のスコープにおけるシンボルには影響しません。1つまたは複数の名前が指定されている場合は、それらのグローバルシンボルだけがクリアされます。

戻り値

なし

オプションの引数

name 任意のグローバル変数名。

次も参照

「Clear Symbols(<name>, <name>, ...)」

Clear Log()

説明

ログウィンドウの内容をクリアする。

Clear Symbols(<name>, <name>, ...)

説明

任意およびすべてのスコープにおけるすべてのシンボルの値をクリアする。1つまたは複数の名前が指定されている場合は、それらのシンボルだけがクリアされます。

戻り値

なし

オプションの引数

name 任意のグローバル変数名。

次も参照

「Clear Globals(<name>, <name>, ...)」

Close Log()

説明

ログを閉じる。

Define Class("class name", <Base Class("base class name", <"base class name", ...>),> <Show(All(Boolean)) | Show(<Members(Boolean),> <Methods(Boolean),> <Functions(Boolean)>),> <Assignment Statements>)

説明

新しいクラスオブジェクトを定義する。

例

```
Define Class(
   "aa",
   _init_ = Method( {} ); x = 1; m1 = Method( {a, b}, a * b )
);
```

Delete Classes(<Force(Boolean)>, < <class>, ...>)

説明

現在定義されているクラスをすべて削除する。

オプションの引数

Force(Boolean) クラスが使用中の場合でも削除する。

class 削除するクラスを指定する。複数のクラスを指定できます。この引数には、定義されたクラスを表す引用符付き文字列、または、インスタンスが作成されたクラスオブジェクトへの参照名を指定できます。

Delete Globals(<name>, <name>, ...)

説明

グローバルシンボルをすべて削除する（ロックされているもの以外）。グローバル以外の任意のスコープにおけるシンボルには影響しません。1つまたは複数の名前が指定されている場合は、それらのグローバルシンボルだけがクリアされます。

オプションの引数

name 任意のグローバル変数名。

次も参照

「Delete Symbols(<name>, <name>, ...)」

Delete Namespaces(<Force(Boolean expression)>, < <namespace reference>, ...>)

Delete(<Force(Boolean expression)>, < <namespace reference>, ...>)

説明

現在定義されているすべての名前空間、または指定された名前空間を削除する。

オプションの引数

Force(Boolean expression) 名前空間が使用されている場合でも削除する。

namespace reference 削除する名前空間を指定する。複数指定できます。

メモ

- ロックされた名前空間を含む名前空間を削除しようとした場合、ログにエラーが表示されます。ロックされた名前空間を削除するには、Force()引数を指定しなければなりません。
- 引数が何も指定されていない場合、Delete Namespaces()はロックされた名前空間を無視します。

Delete Symbols(<name>, <name>, ...)

説明

任意およびすべてのスコープにおけるすべてのシンボルを削除する。1つまたは複数の名前が指定されている場合は、それらのシンボルだけが削除されます。

オプションの引数

name 任意のグローバル変数名。

次も参照

「Delete Globals(<name>, <name>, ...)」

Eval(expr)

説明

exprを評価し、その評価の結果を戻す（つまり、アンクォートを行う）。

戻り値

評価の結果

引数

expr 任意のJSL式。

Eval Insert(*string*, <startDel>, <endDel>, < <<Use Locale(1) >)

説明

複数の置換を行う。

戻り値

結果

引数

string 置換したい式が途中に指定されている引用符付き文字列。

startDel（オプション）開始の区切り文字。デフォルトの値は^。

endDel（オプション）終了の区切り文字。デフォルトの終了文字は、開始文字と同じ文字。

Use Locale(1)（オプション）ロケール固有の数値形式を維持する引数。

Eval Insert Into(*string*, <startDel>, <endDel>)

説明

複数の置換を行う。Eval Insertと同じ処理が実行され、引用符付きの*string*に結果が割り当てられます。

戻り値

結果

引数

string 式が組み込まれている文字列を含む引用符付き文字列変数。

startDel（オプション）開始の区切り文字。デフォルトの値は^。

endDel（オプション）終了の区切り文字。デフォルトの終了文字は、開始文字と同じ文字。

Eval List

「Eval List({list})」を参照してください。

Exit(<NoSave>)

Quit(<NoSave>)

説明

JMPを終了する。

戻り値

なし

引数

NoSave（オプション）名前付きコマンド。開いているファイルを保存するかどうかを確認せずに、JMPを終了します。このオプション名も、大文字／小文字の区別は**なく**、また、スペースを含めてもかまいません。

First(expr, <expr>, ...)

説明

引数で指定されたすべての式を評価する。

戻り値

最初に評価された式の結果のみ

引数

expr 任意の有効なJSL式。

Function({arguments}, <{local variables}>, <Return(<expr>)>, script)

説明

*arguments*をローカル変数とした、スクリプト（*script*）を保存する。

戻り値

定義されたとおりの関数。Return()引数が指定されている場合は、指定の式を戻します。

呼ばれたときは、指定の引数（*arguments*）でスクリプト（*script*）を実行した結果を戻します。

引数

{arguments} 関数に渡す引数のリスト。オプションまたは必須の引数を指定できます。

{local variables} 関数に対してローカルである変数のリスト。ローカル変数は、次の3つの方法で宣言できます。

```
{var1, var2}
{var1=0, var1="a string"}
{Default Local}
```

最後のオプションは、関数内で使用されている変数のうち、適用範囲が指定されていないものがすべてローカルであることを宣言します。

Return(expr)（オプション）ユーザー定義関数から式を戻す引数。ヌルの式を使用した場合は、ピリオド（.）が戻されます。

script 任意の有効なJSLスクリプト。

Get Class Names(< <class>, ...>)

説明

すべてのクラス、または指定したクラスの名前を取得する。

引数

class 定義されたクラスを表す引用符付き文字列、または、インスタンスが作成されたクラスオブジェクトへの参照名。

戻り値

引数で指定されたクラスの名前のリスト

Get Classes(< <class>, ...>)

説明

すべてのクラス、または指定したクラスへの参照を取得する。

引数

class 定義されたクラスを表す引用符付き文字列、または、インスタンスが作成されたクラスオブジェクトへの参照名。

戻り値

引数で指定されたクラス参照のリスト

Get Environment Variable("variable")

説明

オペレーティングシステムの環境変数の値を取得する。

戻り値

指定された環境変数の値を示す引用符付き文字列。指定された変数が見つからない場合は空を戻します。

引数

"variable" 環境変数の名前を示す引用符付き文字列。

メモ

macOSでは、環境変数名に大文字と小文字の区別があります。Windowsでは区別がありません。

Get Locale Setting()

小数点記号などのロケール設定を取得する。

Get Log(<n>)

説明

ログの内容を、リストで戻す。

戻り値

引用符付き文字列のリスト。各文字列には1行分のログが含まれます。

引数

n （オプション）整数。引数が指定されなかった場合は、すべての行を戻します。正の数が指定された場合は、最初の*n*行を戻します。負の数が指定された場合、最後の*n*行を戻します。*n*=0の場合、どの行も戻しません（空のリストを戻します）。ログが空の場合、空のリストを戻します。

Get Namespace Names(< <namespace reference>,...>)

説明

現在定義されているすべての名前空間の名前をリストで戻す。

例

```
nsaa = New Namespace(
   "aa",
   {
      x = 1
   }
);
nsbb = New Namespace(
   "bb",
   {
      y = 1
   }
```

```
);
lns = Get Namespace Names();
Show( lns );
nsaa << Delete;
nsbb << Delete;
```

Get Namespaces(< <namespace reference>,...>)

説明

現在定義されている名前空間をリストで戻す。

例

```
nsaa = New Namespace(
   "aa",
   {
      x = 1
   }
);
nsbb = New Namespace(
   "bb",
   {
      y = 1
   }
);
lns = Get Namespaces();
```

Include("pathname", <named arguments>)

説明

引用符付き文字列として指定されたパス名（*pathname*）に対応するスクリプトファイルを開き、スクリプトを実行する。

戻り値

インクルードされたスクリプトが戻すものすべて。<<Parse Onlyオプションを使用した場合、Includeはスクリプトの中身を戻します。

名前付き引数

<<Parse Only スクリプトを解析するが、実行はしない。

<<New Context インクルードしたスクリプトを、独自の名前空間で実行する。親とインクルードされるスクリプトがグローバル名前空間を使用する場合は、<<Names Default to Hereを<<New Contextとともに指定します。

<<Allow Include File Recursion インクルード元のファイルを、自分自身からインクルードすることを許す。

メモ

パス名の末尾に空白文字が含まれる場合、Windowsでは空白文字が無視されます。macOSではスクリプトが動作しません。

Include File List()

説明

実行時にインクルードされているファイルのリストを戻す。

Is Class(class)

説明

引数で指定したものがクラスオブジェクトかどうかを示す値を戻す。

引数

インスタンスが作成されたクラスオブジェクトへのクラス参照。

戻り値

0または1

Is Log Open()

説明

ログウィンドウが開いているかどうかの結果を戻す。

Length

「Length(string)」を参照してください。

List

「List(a, b, c, ...)」を参照してください。

Local({name=value, ...}, script)

説明

変数（*name*）をローカル変数として定義し、スクリプト（script）を実行する。

Local Here(expression)

説明

ローカルのHere名前空間ブロックを作成する。複数のスクリプトが同じルートの名前空間から実行されるときの競合を回避したい場合に、この関数を使用します（たとえば、スクリプトが同じ変数を持つ2つのボタンスクリプトを実行する場合など）。引数には任意の有効なJSL式を使用できます。

Lock Namespaces(<*string*>,|< {*string*}, ...>)

説明

名前空間にあるすべての変数または指定した変数をロックし、追加や変更、削除ができないようにする。

例

```
ns = New Namespace(
   "aaa"
);
ns << Lock Namespaces;
Try( ns << Delete Namespaces, Show( exception_msg ) );
Delete Namespaces();
Try( Delete Namespaces( "aaa" ), Show( exception_msg ) );
```

Lock Globals(name1, name2, ...)

説明

1つ以上のグローバル変数を、変更されないようにロックする。

Lock Symbols(<name>, <name>, ...)

説明

指定のグローバル変数をロックする。これにより、変数が変更されたり消去されたりするのを防げます。変数が指定されていない場合は、すべてのグローバル変数をロックします。なお、変数が指定されていない場合でも、スクリプトで**Names Default To Here**モードがオンになっている場合は、すべてのローカル変数だけをロックします。

Log Capture(expr)

説明

引数exprを評価し、通常ログに送られるはずの出力を取得し、ログに出力する代わりに文字列で戻す。

戻り値

ログ出力を示す引用符付き文字列

引数

任意の有効なJSL式。

メモ

ログには何も出力されません。

Method({arg1 = val1, ...}, script)

説明

クラス内にメソッドを作成する。メソッドは、明示的に適用範囲が指定されていないすべての変数（クラスメンバー変数を除く）に、ローカルのスコープを使用します。

引数

{ arg1 = val1, ... }期待される引数とオプションの初期値の式のセット。呼び出し時にメソッドに渡されます。

script 任意の有効なJSLスクリプト。

N Items

「N Items(source)」を参照してください。

Names Default To Here(Boolean)

説明

未解決の名前を保持する場所を指定する。グローバルまたはローカルに保持する場合は*Boolean*を0、Hereスコープ内に保持する場合は*Boolean*を1に指定します。

Namespace(name)

説明

指定された名前（*name*）の名前空間への参照を戻す。

引数

name 名前空間名の引用符付き文字列、または、名前空間への参照。

Namespace Exists(name)

説明

指定された名前（*name*）の名前空間が存在する場合は1、そうでない場合は0を戻す。

New Namespace(<"name">, <{expr, ...}>)

説明

指定の名前（*name*）で新しい名前空間を作成する。名前が指定されていない場合は、匿名の名前が使用されます。

戻り値

名前空間への参照

引数

name（オプション）新しい名前空間の名前を示す引用符付き文字列。

{list of expressions}（オプション）名前空間内の式のリスト。

Open Log(<Boolean>)

説明

ログを開く。ログウィンドウがすでに開いている場合、ブール値の引数を指定するとウィンドウがアクティブになります。

New Object("class name"(constructor arguments))
New Object(class name(constructor arguments))
New Object(class reference(constructor arguments))

説明

クラスのインスタンスオブジェクトを作成する。

引数

"class name" インスタンスを作成するクラスの名前。

class name インスタンスを作成するクラスの、引用符なしの名前。

class reference 既存のクラスオブジェクトへの参照。これを使って、同じクラスの新しいオブジェクトのインスタンスが作成されます。

constructor arguments _init_ コンストラクタに渡される引数。

例

```
Define Class(
   "complex",
   real = 0; imag = 0;
   _init_ = Method( {a, b}, real = a; imag = b; );
   Add = Method( {y}, complex( real + y:real, imag + y:imag ) );
   Sub = Method( {y}, complex( real - y:real, imag - y:imag ) );
   Mul = Method( {y},
      complex( real * y:real - imag * y:imag, imag * y:real + real * y:imag )
   );
   Div = Method( {y},
      t = complex( 0, 0 );
      mag2 = y:Magsq();
      t:real = real * y:real + imag * y:imag;
      t:imag = imag * y:real + real * y:imag;
      t:real = t:real / mag2;
      t:imag = t:imag / mag2;
      t;
   );
   Magsq = Method( {}, real * real + imag * imag );
   Mag = Method( {}, Sqrt( real * real + imag * imag ) );
   To String = Method( {}, Char( real ) || " + " || Char( imag ) || "i" )
);
cl = New Object( complex( 1, 2 ) );
```

Parameter({name=value, ...}, model expression)

説明

「非線形回帰」プラットフォームで用いるモデルの計算式パラメータを定義する。

Parse(*string*)

説明

引用符付き文字列をJSL式に変換する。

Print(expr, expr, ...)

説明

指定された式（*expr*）の値をログに出力する。

Quit()

「Exit(<NoSave>)」を参照してください。

Recurse(function)

説明

定義した*関数*の再帰呼び出しを行う。

Save Log(pathname)

説明

指定のファイルにログの内容を書き込む。

Send(obj, message)

obj << message

説明

プラットフォームオブジェクト（*obj*）にメッセージ（*message*）を送る。

Set Environment Variable("variable", <"value">)

説明

環境変数およびその値に設定する。"value"引数が未指定または引用符付きの空文字列の場合、環境変数はJMPプロセスの環境変数テーブルから削除されます。

Show(expr, expr, ...)

説明

各式（*expr*）の名前と値をログに出力する。

Show Classes(< <class>,...>)

説明

ユーザ定義のクラスの内容をログに表示する。複数のクラスを指定できます。引数を指定しなかった場合、ユーザ定義のクラスがすべて表示されます。

例

```
Define Class(
   "aa",
   _init_ = Method( {} ); x = 1; m1 = Method( {a, b}, a * b )
);
Define Class(
   "bb",
   _init_ = Method( {} ); y = 1; m2 = Method( {a, b}, a / b )
);
Show Classes();
   // クラス aa

   _init_ = Method( {} );
   m1 = Method( {a, b}, a * b );
   x = 1;

   // クラス bb

   _init_ = Method( {} );
   m2 = Method( {a, b}, a / b );
   y = 1;
```

Show Globals()

説明

すべてのグローバルシンボルの値を表示する。グローバル以外の任意のスコープにおけるシンボルは表示されません。

次も参照

「Show Symbols()」

Show Namespaces(< <namespace reference>,...>)

説明

ユーザが定義した名前空間（名前付きおよび匿名の両方）の内容を表示する。名前空間の指定はオプションです。

Show Symbols()

説明

すべてのスコープにおけるすべてのシンボルの値を表示する。

次も参照

「Show Globals()」

Sort List

「Sort List({list}|expr)」を参照してください。

Sort List Into

「Sort List Into({list}|expr)」を参照してください。

Throw("text")

説明

例外をスローする。テキスト（*text*）を指定した場合、スローが実行されると、*exception_msg*という名前のグローバル変数にその*text*が格納されます。*text*が感嘆符（!）で始まり、Try()式の中に入っている場合は、どこで例外が発生したかを示すエラーメッセージを生成します。Try()の第2引数によってThrow()がキャッチされた場合でも、感嘆符（!）がスクリプトを停止します。

Try(expr1, expr2)

説明

最初の式（*expr1*）を評価して、例外がスローされた場合には、そこで停止し、2番目の式（*expr2*）を評価してその結果を戻す。例外がスローされなかった場合は、*expr2*は評価されない。

例

```
Try( Sqrt( "s" ), "invalid" );
   "invalid"

Try( Sqrt( "s" ), exception_msg );
   {"引数を数値（または行列）に変換できません。"(1, 2, "Sqrt", Sqrt/*###*/("s"))}
```

メモ

*Expr2*には、引用符付き文字列を指定することも、戻されたエラーに関する情報を含むグローバルな例外メッセージ（exception_msg）を指定することもできます。

Type(x)

説明

引数（*x*）のタイプを示す引用符付き文字列を戻す。戻されるタイプとしては、Unknown、List、DisplayBox、Picture、Column、TableVar、Table、Empty、Pattern、Date、Integer、Number、String、Name、Matrix、RowState、Expression、Associative Array、BLOBがあります。

Unlock Symbols(name1, name2, ...)

Unlock Globals(name1, name2, ...)

説明

Lock Symbols()またはLock Globals()コマンドでロックされた指定の変数のロックを解除する。

Wait(n)

説明

*n*秒待ってから、スクリプトの実行を継続する。デフォルトの設定は3秒。Wait(0)と設定すると、直前までのメッセージの処理を終えた後その先の処理に移ります。この関数を使用すると、ユーザインターフェースでボタンを押す操作を可能にすることなどができます。Waitで一時停止する時間として設定できる最小値はn = 0.01です。プロンプトを含むJMPダイアログを表示しない場合、一時停止できる時間の最大値は、n = 60*60*4秒です。

メモ

Wait(*n*)は、確認するのに十分な時間、何かを表示しておきたい場合や、プラットフォームの起動を完了させてから、そのプラットフォームを操作するスクリプトを実行する場合、スクリプトの実行中にユーザインターフェースでボタンを押す必要がある場合などに使用します。

Watch(all | name1, ...)

説明

グローバル、Here、ローカルの名前空間とその値をウィンドウに表示する。引数に"all"が指定されている場合、すべての変数がウィンドウに表示されます。

メモ

- 新しく追加した変数は、ウィンドウに追加されません。
- メッセージを使って変更された連想配列の表示はサポートされていません。

Wild()

説明

どんな式にも一致するワイルドカードの位置を表す。この関数は、Extract Expr()での式のマッチングにだけ使用できる。

Wild List()

説明

どんな式にも一致するワイルドカードの位置を表す。この関数は、Extract Expr()での式のマッチングにだけ使用できる。

Write("text")

説明

テキスト（*text*）から引用符を取り除いたものをログに書き込む。

Pythonインテグレーション関数

Python Connect()

説明

Pythonインテグレーションのアクティブな接続を、スクリプト可能なオブジェクトとして戻す。JMP 14～17でPython Connect()に使用できたパラメータは廃止されました。

Python Control()

説明

この関数は廃止されました。

Python Create JPIP CMD()

説明

Pythonのpipコマンドを使用する、jpipコマンドラインラッパースクリプトを作成する。生成されたスクリプトの保存先ディレクトリを選択するダイアログが開きます。このスクリプトによりpipの全機能が使用可能となり、JMPに組み込まれたPython環境に必要な環境変数が設定されます。

例

```
Names Default To Here( 1 );
Python Create JPIP CMD();
```

Python Disconnect()

この関数は廃止されました。

Python Execute({list of inputs}, {list of outputs}, Python_Code)

説明

Python環境に対して、第1引数のリストに指定されたJMP変数を送り、第3引数に指定されたPythonコードをサブミットする。第2引数のリストに指定された変数が、JMPに戻されます。

戻り値

成功した場合は0、そうでなければ1

位置引数

{list of inputs} 入力としてPythonに送られるJMP変数名のリスト。

{list of outputs} Pythonからの出力を格納するJMP変数名のリスト。

Python_Code Pythonで実行するコード。

例

以下の例は、文字変数、数値変数、行列をPythonに渡し、Pythonが、行列演算を行います。その後、Python Execute()により、行列演算によって作成された行列、および始めに渡された文字変数と数値変数の値を取得しています。

```
Names Default To Here( 1 );

a = "abcdef";
d = 3.141;
x = 0;
z = 0;
v = [1 0 0, 0 1 0, 0 0 1];
// pi, e, phi, c, Plank's, Faraday, 345 triangle
m = [3.141 2.718 1.618,
2.997 6.626 9.648,
3 4 5];
ml = Python Execute(
   {v, m, a, d},
   {x, z, a, d},
   "\[
import numpy as np
a = np.multiply(v, m) # 行列の積
d = np.divide(v, m) # 行列の除算
z = np.multiply(m, np.linalg.inv(v)) # m * inv(v) は左除算
x = np.multiply(np.linalg.inv(m), v) # inv(m) * v は右除算
]\"
);
Show( v, m, ml, x, z, a, d );
   v =
   [  1 0 0,
      0 1 0,
      0 0 1];
   m =
```

```
[  3.141 2.718 1.618,
   2.997 6.626 9.648,
   3 4 5];
ml = 0;
x =
[  -0.681183278459541 0 0,
   0 1.3532624962586 0,
   0 0 1.57966926070039];
z =
[  3.141 0 0,
   0 6.626 0,
   0 0 5];
a =
[  3.141 0 0,
   0 6.626 0,
   0 0 5];
d =
[  0.318369945877109 0 0,
   0 0.150920615756112 0,
   0 0 0.2];
```

Python Get(*name*)

説明

*name*引数で指定されたPythonの変数を、JMPで取得する。

戻り値

*name*引数で指定された変数の値

引数

name JMPに送るPython変数の名前。引数には、数値、引用符付き文字列、行列、リスト、データフレームのいずれかのデータタイプのPython変数を指定できます。

例

```
Names Default To Here( 1 );

x1 = {1, 2, 3};
Python Send( x1 );
x2 = Python Get( x1 );
Show( x1, x2 );
   x1 = {1, 2, 3};
   x2 = {1, 2, 3};
```

Python Get Graphics()

説明

この関数は廃止されたため、使用すると構文エラーが生じる。matplotlibのpyplotによって最後に書き込まれたグラフィックオブジェクトを、引数で指定されたグラフィック形式で戻していました。

例

次の例では、JMP 17以前のコード例をコメント部分で示し、代わりとなるJMP 18以降でのコード例を示しています。

```
Names Default To Here( 1 );

ml = Python Submit(
   "\[
# この例では、matplotlibパッケージがインストールされている必要がある。
import matplotlib.pyplot as plt
plt.clf()          # 必ずクリーンなプロットから開始する:
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.draw()

# 選択した場所にイメージを保存する
plt.savefig('/tmp/get_graphics_img.png')
]\"
);
// plot = Python Get Graphics( png );       // 廃止
plot = Open( "/tmp/get_graphics_img.png", png );
pngJMP = New Window( "Plot", Picture Box( plot ) );
Python Submit( "plt.close()" );
rc = Delete File( "/tmp/get_graphics_img.png" );
```

Python Get Version()

説明

JMPのPythonインターフェースで使用されているPythonのバージョン番号を戻す。

Python Init()

説明

この関数は廃止されました。JMP18では、Python Connect()で同じ処理が行えます。

例

```
Names Default To Here( 1 );

Python Init();
Python Submit( "\[
str = 'The quick brown fox jumps over the lazy dog';
]\" );
getStr = Python Get( str );
Show( getStr );
   getStr = "The quick brown fox jumps over the lazy dog";
```

Python Install Packages(*packages*)

説明

Pythonパッケージを JMPのsite-packagesディレクトリにインストールする。シンプルなパッケージのインストール以外の処理を行うには、Python Create JPIP CMD()を使用して、Directory Pick()で選択したディレクトリの中にコマンドラインラッパースクリプトを作成してください。または、JMPのPythonスクリプトウィンドウからインストールを実行することもできます。『スクリプトガイド』を参照してください。

例

```
Names Default To Here( 1 );
// numpyおよびpandasパッケージをインストール
Python Install Packages( "numpy pandas" );
```

Python Is Connected

この関数は廃止されました。常に 1 を戻します。

Python JMP Name to Python Name(*name*)

説明

Pythonの命名規則に従い、JMP変数名を、対応するPython変数名に変換する。

戻り値

変換後のPython名の引用符付き文字列

引数

name Pythonに送るJMP変数の名前。

Python Send(*name*, <Python Name(*name*)>)

説明

JMPからPythonに変数を送る。

戻り値

成功した場合は0を戻す。

引数

name Pythonに送るJMP変数の名前。

<Python Name(*name*)> Python環境での変数の名前を指定する。

Python Send File(*name*, <Python Name(*name*)>)

説明

データファイルをPythonに送る。filename引数は、Pythonに送るファイルのパス名の引用符付き文字列。

引数

name Pythonに送るファイルの名前。

<Python Name(*name*)> Python環境でのファイルの名前を指定する。

Python Submit(*Python_Code*)

説明

Pythonコードを Python環境にサブミットする。

戻り値

成功した場合は0、そうでなければ1

引数

Python_Code Pythonで実行するコード。ステートメントは、引用符付き文字列か文字列のリスト。

例

```
Names Default To Here( 1 );
Python Submit( "\[
str = 'The quick brown fox jumps over the lazy dog'
a = 200]\" );
getStr = Python Get( str );
getNum = Python Get( a );
Show( getStr, getNum );
   getStr = "The quick brown fox jumps over the lazy dog";
   getNum = 200;
```

Python Submit File(*path*)

説明

指定されたファイルのプログラムを、Pythonにサブミットする。

引数

path 実行するPythonのプログラムコードを含んだファイルのパス。

Python Term()

この関数は廃止されました。

Rインテグレーション関数

R Connect(<named_arguments>)

説明

現在のR接続オブジェクトを戻す。Rへの接続が確立されていない場合は、R接続インターフェースを初期化し、アクティブなRインターフェース接続をスクリプト可能なオブジェクトとして戻します。

戻り値

スクリプト可能なオブジェクト

引数

Echo(Boolean)（オプション）実行したRのプログラムコードを、JMPのログに出力する。このオプションは、グローバルです。デフォルト値は1（真）。

R Control(Interrupt|Async(Boolean)|Echo(Boolean))

説明

Rの制御オプションを変更する。

R Execute({ list of inputs }, { list of outputs }, "rCode", <named_arguments>)

説明

現在のグルーバルなR接続に対して、第1引数のリストに指定されたJMP変数を送り、第3引数に指定されたRコードをサブミットする。第2引数のリストに指定されたR変数が、JMPに戻されます。

戻り値

成功した場合は0、そうでない場合は0以外の値

引数

{ list of inputs } 入力としてRに送られるJMP変数名のリスト

{ list of outputs } Rから戻される出力を格納するJMP変数名のリスト

rCode サブミットするRコードを示す引用符付き文字列

Expand(Boolean)（オプション）この引数が1（真）の場合、Rにサブミットする前に、Rコードに対してEval Insert()を実行します。

Echo(Boolean)（オプション）この引数が1（真）の場合、実行したRのプログラムコードを、JMPのログに出力する。このオプションは、グローバルです。（オプション）デフォルト値は1（真）。

例

JMP変数xとyをRに送り、Rステートメントz <- x * yを実行し、そのR変数zをJMP側で取得します。

```
x = [1 2 3];
y = [4 5 6];
rc = R Execute( {x, y}, {z}, "z <- x * y" );
```

R Get(variable_name)

説明

Rの変数を、JMPで取得する。

戻り値

variable_nameで指定した変数の値

引数

name（必須）値を取得するR変数の名前。

例

qbxという名前の行列と、dfという名前のデータフレームが、R上に存在するとします。

```
// R変数qbxの値を取得し、それをJMP変数qbxに代入
qbx = R Get( qbx );

// R変数dfのデータフレームを、JMP側でデータテーブルとして取得し、参照をJMP変数dfに格納
df = R Get( df );
```

R Get Graphics("format")

説明

Rのグラフウィンドウに最後に出力されたグラフを、指定の形式で取得する。

戻り値

JMPピクチャーオブジェクト

引数

format（必須）使用するグラフィック形式。有効な形式は、png、bmp、jpeg、jpg、tiff、tif、gifです。

R Get Version

説明

JMPのRインターフェースに使用されているRのバージョン番号を戻す。

R Init(named_arguments)

説明

JMPからRへの接続を、初期化する。

戻り値

成功した場合は0、そうでない場合は0以外

引数

Echo(Boolean)（オプション）実行したRのプログラムコードを、JMPのログに出力する。このオプションは、グローバルです。（オプション）デフォルト値は1（真）。

R Is Connected()

説明

Rへの接続が存在するかどうかを特定する。

戻り値

アクティブなR接続がある場合は1、そうでない場合は0を戻す。

引数

なし

R JMP Name to R Name(name)

説明

Rの命名規則に従い、JMP変数名を、対応するR変数名に変換する。Rへのアクティブな接続が必要です。

引数

name Rに送るJMP変数の名前。

戻り値

Rでの命名規則に従った変数名を示す引用符付き文字列

R Send(name, <R Name(name)>)

説明

JMPからRに変数を送る。

戻り値

成功した場合は0、そうでない場合は0以外の値

引数

name（必須）Rに送るJMP変数の名前。

R Name(name)（オプション）Rに送る変数に別の名前をつけることができる。以下の例のように指定します。

```
R Send(Here:x, R Name("localx"))
```

以下は、データテーブルの場合のみ適用されます。

Selected(Boolean)（オプション）名前付き、ブール値。参照先データテーブルの選択された行のみをRに送ります。

Excluded(Boolean)（オプション）名前付き、ブール値。参照先データテーブルの除外された行のみをRに送ります。

Labeled(Boolean)（オプション）名前付き、ブール値。参照先データテーブルのラベルのついた行のみをRに送ります。

Hidden(Boolean)（オプション）名前付き、ブール値。参照先データテーブルの非表示の行のみをRに送ります。

Colored(Boolean)（オプション）名前付き、ブール値。参照先データテーブルの色のついた行のみをRに送ります。

Markered(Boolean)（オプション）名前付き、ブール値。参照先データテーブルのマーカーのついた行のみをRに送ります。

Row States(Boolean, <named arguments>)（オプション）名前付き。ブール値の引数とオプションの名前付き引数を指定する。JMPデータテーブルにおける行属性の情報を、「RowState」という列名のデータ列を追加して、Rに送ります。個別の設定をまとめて追加することによって、複数の行の属性を作成できます。個々の値は次のとおりです。

- Selected = 1
- Excluded = 2
- Hidden = 4
- Labeled = 8
- Colored = 16
- Markered = 32

Row States()引数の名前付き引数は次のとおりです。

Colors(Boolean)（オプション）名前付き、ブール値。行の色を送ります。「RowStateColor」という名前のデータ列を追加します。

Markers(Boolean)（オプション）名前付き、ブール値。行のマーカーを送ります。「RowStateMarker」という名前のデータ列を追加します。

例

行列を変数Xに代入します。そして、その変数をRに送ります。

```
X = [1 2 3];
rc = R Send( X );
```

データテーブルを開き、その参照を*dt*に割り当てます。そして、そのデータテーブルを、現在の行属性も含めてRに送ります。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
rc = R Send( dt, Row States(1) );
```

R Send File("pathname", <R Name("name")>)

説明

JMPからRに指定したデータファイルを送る。

戻り値

成功した場合は0、そうでない場合は0以外の値

引数

pathname（必須）ファイルのパス名を含む引用符付き文字列。

R Name(name)（オプション）Rに送るデータファイルに別の名前をつけることができる。

R Submit("rCode", <named_arguments>)

説明

指定されたRコードを、R上で実行する（現在のグローバルなR接続に、サブミットする）。

戻り値

成功した場合は0、そうでない場合は0以外の値

引数

rCode（必須）サブミットするRコードを示す引用符付き文字列。

Expand(Boolean)（オプション）この引数が1（真）の場合、Rにサブミットする前に、Rコードに対してEval Insert()を実行します。

Echo(Boolean)（オプション）この引数が1（真）の場合、実行したRのプログラムコードを、JMPのログに出力する。このオプションは、グローバルです。デフォルト値は1（真）。

Async(Boolean)（オプション）この引数が1（真）の場合、Escキーを押すか、またはR接続に対してrconn<<Control(Interrupt(1))のメッセージを送ると、処理の途中でもキャンセルできる。デフォルト値は0（偽）。

例

```
rc = R Submit("\[
   x <- rnorm(5)
   print(x)
   y <- rnorm(5)
   print(y)
   z = plot(x, y)
]\" );
```

R Submit File("pathname")

説明

*pathname*で指定されたファイルに含まれるRコードを、R上で実行する。

戻り値

成功した場合は0、そうでない場合は0以外の値

引数

pathname 実行するRコードを含むファイルのパス名を示す引用符付き文字列。

R Term()

説明

Rへの接続を終了する（アクティブなR接続インターフェースを終了する）。

戻り値

終了が成功した場合は0、そうでない場合は-1を戻す。

引数

なし

乱数関数

Col Shuffle(<By var,...>)

説明

列の計算式で使用した場合、現在のデータテーブルの行番号をランダムに並べる。

メモ: この関数は、一般に列の計算式で使用されます。

戻り値

1と現在のデータテーブルの行数の間の整数乱数

引数

By var（オプション）By変数により、By変数の値のグループ内で行をランダムに並べることが可能になる。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << New Column( "Shuffle", Numeric, Continuous, Set Formula( Col Shuffle() ) );
```

この例の計算式は、評価されるたびに行の番号（1～40）の順序をランダムに並べ替えます。どの番号も、1度ずつ使用されます。

Make KFold Formula(folds, Y Columns(cols), <<Stratification Columns(cols), <<Grouped Columns(cols))

説明

列の計算式で使用された場合、k水準の検証列を生成する。この関数は、主に「検証列の作成」プラットフォームによって使用されます。

引数

folds 計算式によって生成される分割の数。

Y Columns 1つ以上の数値列を割り当てる。

<<Stratification Columns 1つまたは複数の層別の列を割り当てます。

<<Grouped Columns 1つまたは複数のグループの列を割り当てます。

Make Validation Formula(rates, <<Stratification Columns(cols), <<Grouped Columns(cols), <<Cutpoint Column(col), <<Cutpoint Batch ID(col), <<Determine cutpoins using("Proportions"|"Numbers of Rows"|"Fixed Time or

Date"|"Elapsed Time"), <<Assign Extra Rows("To Training"|"To Validation"|"To Test"))

説明

列の計算式で使用された場合、2水準または3水準の検証列を生成する。この関数は、主に「検証列の作成」プラットフォームによって使用されます。

引数

rates 学習、検証、テストの割合を指定する3つの割合のベクトル。

<<Stratification Columns 1つまたは複数の層別の列を割り当てます。

<<Grouped Columns 1つまたは複数のグループの列を割り当てます。

<<Cutpoint Column 特定の閾値で分割するときの基準となる、数値列を1つ割り当てます。

<<Cutpoint Batch ID カットポイントの列を割り当てると、カットポイントバッチIDの列も割り当てることができます。これにより、カットポイントバッチIDの各水準内で、カットポイントの値を設定できます。

<<Determine cutpoints using カットポイントの設定方法を指定します。

<<Assign Extra Rows 余った行を、学習セット・検証セット・テストセットのいずれに追加するかを指定します。

Random Beta(alpha, beta, <theta=0>, <sigma=1>)

説明

*alpha*と*beta*の2つの形状パラメータ、およびオプションのパラメータ*theta*と*sigma*を持つベータ分布に従う乱数を戻す。

引数

alpha, beta 形状パラメータαおよびβ。両者とも正の値。

theta（オプション）閾値パラメータθ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Random Beta Binomial(n, p, <delta=0>)

説明

試行回数*n*、確率*p*、過分散パラメータ*delta*のベータ二項分布に従う乱数を戻す。

引数

n 試行回数。値は2以上でなければなりません。*n*に整数でない値を指定した場合、非整数部分は切り捨てられます。

p 各試行の成功確率。値は0～1の間でなければなりません。

delta 過分散パラメータδ。値の範囲は、Maximum[-*p*/(*n*-*p*-1), -(1-*p*)/(*n*-2+*p*)]～1です。デフォルト値は0。

Random Binomial(n, p)

説明

試行回数が*n*で、イベントの生起確率が*p*である二項分布に従う乱数を生成する。

引数

p 各試行の成功確率。値は0～1の間でなければなりません。

n 試行回数。

Random Category(probA, resultA, probB, resultB, <..., ...,> resultElse)

説明

指定された確率と結果の式のペアから、結果の式の1つをランダムに選んで戻す。一様乱数を生成し、*prob*引数と比較して、どの*result*引数を戻すかが決定されます。

引数

probA 戻す結果の式の確率を表す0～1の数値。

resultA *probA*に対応する式。

resultElse 前の式の結果が戻されない場合に戻される式。

Random Cauchy()

説明

中央値が0のCauchy分布に従う(擬似)乱数を戻す。

Random ChiSquare(df, <nc=0>)

説明

指定した*df*（自由度）とオプションの非心度パラメータを持つカイ2乗分布に従う乱数を戻す。

引数

df 自由度*n*。正の値。

nc（オプション）非心度パラメータλ。負でない値。デフォルト値は0。

Random ExGaussian(location, scale, shape)

説明

location、*scale*、*shape*のパラメータで指定された指数修正Gauss分布に従う乱数を戻す。指数修正Gauss分布は、正規分布と指数分布の和です。

引数

location 正規分布の平均。

scale 正規分布の標準偏差。

shape 指数分布のλパラメータ。

メモ: Random ExGaussian関数のパラメータ表現は、「一変量の分布」プラットフォームの［指数のあてはめ］オプションで使用されているパラメータの逆数になっています。

Random Exp()

説明

尺度パラメータが1の指数分布に従う乱数を戻す。また、-Log(Random Uniform ())（一様乱数の対数の符号を変えたもの）と等価です。

Random F(dfnum, dfden, <noncentral=0>)

説明

指定した*dfnum*、*dfden*、およびオプションの非心度パラメータを持つF分布に従う乱数を戻す。

引数

dfnum *F*分布の分子に使用されるカイ2乗分布の自由度、v_1。*dfnum*は、0より大きくなければなりません。

dfden *F*分布の分母に使用されるカイ2乗分布の自由度、v_2。*dfden*は、0より大きくなければなりません。

noncentral（オプション）非心度パラメータλ。負でない値。デフォルト値は0。

Random Frechet(<mu=0>, <sigma=1>)

説明

位置*mu*、尺度*sigma*のFréchet分布に従う乱数を戻す。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Random Gamma(alpha, <scale=1>)

説明

指定した*alpha*とオプションの*scale*を持つガンマ分布に従う乱数を戻す。

引数

alpha 形状パラメータα。正の値。

scale（オプション）尺度パラメータβ。正の値。デフォルト値は1。

Random Gamma Poisson(lambda, <sigma=1>)

説明

パラメータ*lambda*と*sigma*のガンマPoisson分布に従う乱数を生成する。

引数

lambda 形状パラメータλ。値は0より大きくなければなりません。

sigma（オプション）過分散パラメータσ。値は1以上でなければなりません。デフォルト値は1。過分散パラメータが1の場合、分布はPoisson(λ)分布になります。

Random GenGamma(<mu=0>, <sigma=1>, <lambda=0>)

説明

パラメータ*mu*、*sigma*、*lambda*の拡張一般化ガンマ分布に従う乱数を生成する。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

lambda（オプション）形状パラメータλ。デフォルト値は0。

Random Geometric(p)

説明

1回の試行における成功確率が*p*である幾何分布に従う乱数を生成する。生成された値は、成功するまでの失敗回数。

Random GLog(mu, sigma, lambda)

説明

パラメータ*mu*、*sigma*、*lambda*の一般化対数分布に従う乱数を生成する。

引数

mu 位置パラメータμ。

sigma 尺度パラメータσ。正の値。

lambda 形状パラメータλ。正の値。

Random Index(n, k)

説明

1～*n*までの重複しない整数の乱数を含む*k* x 1行列を戻します。

Random Integer(n)

Random Integer(k, n)

説明

1～*n*または*k*～*n*までの整数の乱数を生成する。

Random Johnson Sb(gamma, delta, theta, sigma)

説明

gamma、*delta*、*theta*、*sigma*のパラメータを持つJohnson Sb分布に従う乱数を戻す。

引数

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Random Johnson Sl(gamma, delta, theta, <sigma=1>)

説明

gamma、*delta*、*theta*、オプションの*sigma*のパラメータを持つJohnson Sl分布に従う乱数を戻す。

引数

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 分布が正の方向に歪むか、負の方向に歪むかを示すオプションのパラメータσ。sigmaは、+1（正の方向）または-1（負の方向）のどちらかをとります。デフォルトは+1です。

Random Johnson Su(gamma, delta, theta, sigma)

説明

gamma、*delta*、*theta*、*sigma*のパラメータを持つJohnson Su分布に従う乱数を戻す。

引数

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Random LEV(<mu=0>, <sigma=1>)

説明

位置*mu*、尺度*sigma*の最大極値分布に従う乱数を生成する。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Random LogGenGamma(<mu=0>, <sigma=1>, <lambda=0>)

説明

パラメータ*mu*、*sigma*、*lambda*の対数一般化ガンマ分布に従う乱数を生成する。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

lambda（オプション）形状パラメータλ。デフォルト値は0。

Random Logistic(<mu=0>, <sigma=1>)

説明

位置*mu*、尺度*sigma*のロジスティック分布に従う乱数を生成する。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Random Loglogistic(<mu=0>, <sigma=1>)

説明

位置*mu*、尺度*sigma*の対数ロジスティック分布に従う乱数を生成する。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Random Lognormal(<mu=0>, <sigma=1>)

説明

位置*mu*、尺度*sigma*の対数正規布に従う乱数を生成する。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Random Multivariate Normal(mean, covar, <nrows=1>)

説明

平均ベクトル*mean*と共分散行列*covar*を持つ多変量正規分布の乱数のベクトルを戻す。複数のベクトルが生成されるようにするには、*nrows*引数に1より大きい整数を指定します。*nrows*が1より大きい場合は、行列が戻されます。乱数のベクトルまたは行列に含まれる列の数は、*covar*引数の行の数と等しくなります。

引数

mean 多変量正規分布の平均ベクトル。

covar 多変量正規分布の共分散行列。この行列は、列の数が平均ベクトルと等しい、対称な正方行列でなければなりません。

nrows 戻される乱数のベクトルの数を指定するオプションの引数。デフォルトの行数は1。

Random Negative Binomial(n, p)

説明

成功確率が*p*、成功回数が*n*の負の二項分布に従う乱数を戻す。

Random Normal(<mu=0>, <sigma=1>)

説明

平均が*mu*、標準偏差が*sigma*の正規分布に従う乱数を戻す。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Random Normal Mixture(meanvec, sdvec, probabvec)

説明

引数で指定された正規混合分布に従う乱数を生成する。

引数

meanvec グループ平均を示すベクトル。

sdvec グループ標準偏差を示すベクトル。

probvec グループ確率を示すベクトル。

Random Poisson(lambda)

説明

形状パラメータ*lambda*を持つPoisson分布に従う乱数を戻す。

引数

lambda 形状パラメータλ。値は0より大きくなければなりません。

Random Reset(seed)

説明

指定されたシード値（*seed*；初期値）から乱数系列を再開する。後に同じ結果を再現したい場合には、乱数シード値として4,294,967,295以下の正の整数を指定します。

メモ

Random Reset関数を使うと、乱数シード値を設定し、乱数セットを再現することができます。JMPの乱数生成アルゴリズムは改良されることがあるため、異なるバージョンのJMPを使った場合には必ずしも同じ乱数が生成されない可能性があります。

Random Seed State(<seed state>)

説明

B乱数シード値の状態を戻します。もしくは、乱数シード値の状態を設定します。なお、乱数シード値の状態は、BLOBオブジェクトとして表されています。

Random SEV(<mu=0>, <sigma=1>)

説明

位置*mu*、尺度*sigma*の最小極値分布に従う乱数を生成する。

引数

mu（オプション）位置パラメータμ。デフォルト値は0。

sigma（オプション）尺度パラメータσ。正の値。デフォルト値は1。

Random SHASH(gamma, delta, theta, sigma)

説明

gamma、*delta*、*theta*、*sigma*のパラメータを持つsinh-arcsinh（SHASH）分布に従う乱数を戻す。

引数

gamma 形状パラメータγ。

delta 形状パラメータδ。正の値。

theta 位置パラメータθ。

sigma 尺度パラメータσ。正の値。

Random Shuffle(matrix)

説明

要素をランダムにシャッフルした行列を戻す。

Random t(df, <noncentral=0>)

説明

指定された*df*（自由度）のt分布に従う乱数を生成する。非心度引数の値は負でも正でもかまいません。*noncentral*のデフォルト値は0です。

Random Triangular(min, mode, max)

Random Triangular(mode, max)

Random Triangular(mode)

説明

0～1の値の三角分布に従う乱数を生成する（最頻値は*mode*で指定する）。三角分布は、通常、データ数の少ない母集団で使用されます。

引数

min 三角分布の下限値を指定する。デフォルト値は0です。

mode 三角分布の最頻値を指定する。

max 三角分布の上限値を指定する。デフォルト値は1です。

メモ

最頻値のみを指定した場合は、最小値は0、最大値は1になります。最頻値と最大値を指定した場合は、最小値はデフォルトで0となります。

Random Uniform()

Random Uniform(x)

Random Uniform(min, max)

説明

0～1の値の一様分布に従う乱数を生成する。Random Uniform(x)では0～xの間で乱数が生成されます。Random Uniform (min, max)では*min*～*max*の間で乱数が生成されます。生成された乱数データは、ほぼ均等に分布します。

Random Weibull(shape, <scale=1>)

説明

*shape*とオプションの*scale*のパラメータを持つWeibull分布に従う乱数を戻す。

引数

shape 形状パラメータβ。正の値。

scale（オプション）尺度パラメータα。正の値。デフォルト値は1。

Resample Freq(<rate=1, <column>>)

説明

復元抽出法（重複抽出法）に基づき度数の列を生成する。引数が指定されていない場合は、元データと同じ標本サイズの標本が抽出されます。

メモ: この関数は、一般に列の計算式で使用されます。

引数

rate（オプション）標本の再抽出率。デフォルト値は1です。負の*rate*は、小数点以下の値を含む度数が使用できることを意味します。

column（オプション）*column*を指定した場合は、*rate*も指定する必要があります。標本サイズは、指定された列の和に抽出率を掛けたものとなります。*rate*が負の場合、標本サイズは、指定された列の和に抽出率をかけ、符号を変えたものとなります。列を指定しなかった場合、生成される度数の合計は行数に等しくなります。

例

スクリプトを実行するたびに、度数列が同じ数値になるようにするには、As Constant()を使用します。As Constant()では、いったん式を評価して定数を求めると、その値は変化しません。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dc = dt << New Column( "column",
   Formula(
      As Constant(
         Random Reset( 123 );
         0;
      ) + Resample Freq()
   )
);
dc << Eval Formula;
```

メモ

- この関数を使用すると、0、1、2といった度数を含む列が生成されます（通常、度数が1である行が多い）。これらの度数は、元の標本から無作為に*n*行選択することにより作成されます。
- また、**既存の**度数列に対してこの関数を使用すると、元の度数列とほぼ同じ値（期待値は同じ値）の列が生成されます。（既存の度数列に比例する割合で）ランダムに選択するため、既存の度数列の値とは多少異なります。

行関数

As Table(matrix, <matrix 2, ...>, < <<invisible >, < <<private >, < <<Column Names({list}) >)

説明

行列*matrix*から新しいデータテーブルを作成する。

戻り値

新しいデータテーブル

引数

matrix 任意の行列。

<<invisible 非表示のデータテーブルを作成する。データテーブルは非表示になりますが、「JMPホームウィンドウ」や［ウィンドウ］メニューにはリストされます。

<<private テーブルを完全に非表示にする。プライベートのデータテーブルを作成すると、データへのアクセスが高速化しますが、データテーブルのデータを保持するために必要なメモリが少なくなるわけではありません。

<<Column Names(list) データの列名を指定したリスト。引数は引用符付きの列名のリストです。

Col Stored Value(<dt>, col, <row>)

説明

列プロパティ（「欠測値のコード」など）を使って割り当てられた値ではなく、実際に列に保存されているデータ値を戻す。

引数

dt（オプション）データテーブルへの参照。この値が指定されていない場合は、現在のデータテーブルが使用される。

col 列の名前。

row（オプション）行の名前または番号。この値が指定されていない場合は、現在の行が使用される。

例

たとえば、「欠測値のコード」列プロパティがx1列に割り当てられていて、「999」が欠測値として扱われるとします。また、別の列に、平均の計算式が含まれているとします。平均を計算するときに、欠測値ではなく「999」を使用するためには、式の中にCol Stored Value()を使用します。

```
Mean( Col Stored Value( :x1 ), :x2, :x3 )
```

Column(<dt>, "name", "formatted")

Column(<dt>, n)

説明

データテーブル列への参照を取得する。

引数

dt（オプション）データテーブルへの参照。指定されていない場合は、現在のデータテーブルが使用される。

name 列名を指定する引用符付き文字列。

formatted 設定されている表示形式でセルの値を戻すよう指定する引用符付き文字列。

n 列番号。

Column Name(n)

説明

nで指定された列の名前を取得する。

戻り値

第*n*列の名前を（引用符付き文字列ではなく）式として戻す。

引数

n 列番号。

Count(from, to, step, times)

説明

列計算式で使用される。*from*から *to*までインクリメントする値を各行に割り当てます。*steps*には、*from*と *to*も含めた、*from*から *to*までの値の個数を指定します。count関数の最初の3つの引数によって決められた等差数列の値が、指定された *times*の数だけ繰り返して作成されます。*to*値に達すると、countは再び *from*値から開始されます。*from*と *to*の引数がデータテーブル列名のときは、Countは最初の行の値を取り、それ以降の行の値は無視されます。

戻り値

最後の値

引数

from 数値、列参照、または式。Countはこの値からカウントを開始する。

to 数値、列参照、または式。Countはこの値でカウントを終了する。

step 数値または式。*from*から *to*まで（両者を含む）のカウントに使用するステップの数を指定する。

times 数値または式。次のステップに進むまでに同じ値を繰り返す回数を指定する。

例

```
/* colnameという名前の列は、0, 3, 6, 0, ...という数列ですべての行が埋められる */
For Each Row(:colname[row()]=count(0, 6, 3, 1))

/* colnameという名前の列は、0, 0, 3, 3, 6, 6, 0, ...という数列ですべての行が埋められる */
For Each Row(:colname[row()]=count(0, 6, 3, 2))
```

メモ

Count()はRow()に依存するので、主に列の計算式内で利用すると便利です。

Current Data Table(<dt>)

説明

引数が指定されていない場合、現在（最前面）のデータテーブルを取得する。引数が指定されている場合は、それを現在のデータテーブルとします。

戻り値

現在のデータテーブルへの参照

引数

dt（オプション）データテーブルの名前またはデータテーブルへの参照。

メモ

プライベートテーブルは、Current Data Table()を用いて現在のテーブルにすることはできません。

Data Table(n)

Data Table("name")

Get Data Table(<project(title|index|box|window),> name|index)

説明

開いている*n*番目のデータテーブル、または、引用符付き文字列で指定された名前（*name*）のデータテーブルへの参照を戻す。

戻り値

指定のデータテーブルへの参照

引数

n データテーブルの番号。

name データテーブル名を指定する引用符付き文字列。

Dif(col, n)

説明

列*col*において、現在行の値から、現在行の*n*行前の値を引いた差を計算する。

戻り値

差

引数

col 列名（たとえば:ageなど）。

n 数値。

Dim(<dt|matrix>)

説明

現在のデータテーブル、指定されたデータテーブル、または行列の次元を含む行ベクトルを戻す。次元とは、行数と列数を指し、この順序でリストされます。

引数

dt データテーブル。

matrix 行列。

メモ

引数が指定されなかった場合は、現在のデータテーブルの次元を戻します。

Get Data Table List(<Project(title|index|box|window>)

説明

現在開いているすべてのデータテーブルのリストを戻す。

メモ

プロジェクトの中で実行するスクリプトで、プロジェクトの外で開いているデータテーブルのリストを取得する場合は、Project(0)を指定します。

Lag(col, n)

説明

*n*行前の列(col)の値を戻す。

N Row(dt); NRow(matrix)

N Rows(dt); NRows(matrix)

説明

指定されたデータテーブル（*dt*）や行列（*matrix*）の行数を戻す。

N Table()

説明

開いているデータテーブルの数を戻す。プライベートのデータテーブルは含みません。

New Column("name", <"data type">, <"modeling type">, <Width(n)>, Format("format", width, precision), <Formula()>, <Set Values>, <Like(column reference)>, <actions>)

説明

データテーブル（*dt*）の最後に、指定された名前（*"name"*）で新しい列を追加する。特に指定をしない場合、列は数値タイプ、連続尺度で総桁数は12となります。

戻り値

列への参照

メモ

メッセージとしても使用可能です。dt<<New Column.

Like()引数は、データタイプ、尺度、形式、数式、その他の属性を参照列から新しい列にコピーします。

次も参照

「dt<<New Column(name, <data type>, <modeling type>, <Format(format, width)>, <Formula()>, <Set Values({..., ..., })>, <Set Property(properties)>)」

New Table("name", <visibility("invisible" | "private" | "visible")>, <actions>)

説明

指定された名前（*name*）で新しいデータテーブルを作成する。

引数

name 新しいテーブル名を示す引用符付き文字列。

visibility（オプション）引用符付きキーワード。invisibleを指定すると、データテーブルは、「JMPホームウィンドウ」と［ウィンドウ］メニューにのみリスト表示されます。privateを指定すると完全に非表示になります。visibleを指定すると、データテーブルは表示されます。"visible"がデフォルトです。

メモ: プライベートのデータテーブルを作成すると、データへのアクセスが高速化しますが、データテーブルのデータを保持するために必要なメモリが少なくなるわけではありません。

actions（オプション）新しいデータテーブルを定義するための引数。

Row()

Row() = y

説明

現在の行の番号を戻すか、設定する。引数はとりません。

Sequence(from, to, <step size>, <repeat times>)

説明

データテーブルの行に連続した数値を挿入する。*step size*引数と*repeat times*引数はオプションで、どちらもデフォルトの値は1です。

Subscribe to Data Table List(<subscriber name|"">, <OnOpen(function)|OnClose(function)|OnRename(function)>)

説明

データテーブルリストへの登録を行う。新しいデータテーブルが追加されたときや、閉じられたとき、名前が変更されたときに通知されます。

Subscript(a, b, c)

list[i]

matrix[b, c]

説明

リストや行列に対して、添え字を指定する。リスト（*list*）からは、*i*番目の項目が抽出されます。または、行列（*matrix*）からは、第*b*行、第*c*列の要素が抽出されます。

Suppress Formula Eval(Boolean)

説明

引数が1の場合、すべてのデータテーブルの計算式の自動評価をオフにする。引数が0の場合は、オンにする。

Unsubscribe to Data Table List(<subscriber name>, <"OnOpen"|"OnClose"|"All">)

説明

Subscribe to Data Table List()で追加されたデータテーブルリストへの登録を削除する。

Where(<dt>, clause)

説明

指定された節にしたがって項目をフィルタリングし、インデックスを戻す。節には、比較関数または条件文を指定できます。同じ節の中で列・行列・リストを混ぜて照合することもできます。

Where()によって戻されるインデックスは、大量のリスト・行列・列を高速に処理できるよう最適化されています。

必須の引数

clause 比較関数または条件文。

オプションの引数

<dt> 評価時に現在のデータテーブルを変更する。

例

```
Names Default To Here( 1 );
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Clear Select << Select Rows(
   Where( Col Max( :"身長(インチ)"n, :年齢 ) >= 68 )
);
dt << Clear Select << Select Rows(
   Where( :"身長(インチ)"n == Col Max( :"身長(インチ)"n, :年齢 ) )
);
```

行の属性関数

As Row State(i)

説明

整数*i*を、行属性に変換する。

戻り値

指定された整数*i*に対応した行属性

引数

i 整数。

Color Of(rowstate)

説明

色番号を戻すか、設定する。

戻り値

行の属性（*rowstate*）の色番号

引数

rowstate 行の属性。

例

5行目に赤を割り当てます。

```
Color Of( Rowstate( 5) ) = 3
```

Color State(i)

説明

色番号を *i* とする行属性を戻す。

戻り値

行の属性

引数

i JMPの色番号。

Combine States(rowstate, rowstate, ...)

説明

複数の行属性を組み合わせる。

戻り値

組み合わせた行属性

引数

rowstate 2つ以上の行属性。

Excluded(rowstate)

説明

「除外する/除外しない」の状態（1または0）を戻すか、もしくは設定する。

戻り値

除外の属性（0または1）

引数

rowstate 1つ以上の行属性。

Excluded State(num)

説明

除外の状態を指定された数値（*num*）に設定した行属性を戻す。

Hidden(rowstate)

説明

「表示しない/再表示」の状態（1または0）を戻す、もしくは、設定する。

Hidden State(num)

説明

非表示の状態を指定された数値（*num*）に設定した行属性を戻す。

Hue State(num)

説明

色相の状態を指定された数値（*num*）に設定した行属性を戻す。

Labeled(rowstate)

説明

「ラベルあり/ラベルなし」の状態（1または0）を戻す、もしくは設定する。

Labeled State(num)

説明

ラベルの状態を指定された数値（*num*）に設定した行属性を戻す。

Marker Of(rowstate)

説明

行の属性のマーカーインデックスを戻すか、設定する。

Marker State(num)

説明

マーカーの状態を指定された数値（*num*）に設定した行属性を戻す。

Row State(<dt,> <n>)

説明

アクティブな行または*n*行目において、初期値と異なる状態にある行属性を戻す。

引数

dt（オプション）位置指定引数。データテーブルへの参照。この引数がデータテーブルを参照する変数でない場合は、データテーブルの式とみなされます。

n 行番号。

例

次の例は、データテーブル参照を作成し、「Big Class.jmp」の1行目の行の属性を戻します。

```
dt1 = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt2 = Open( "$SAMPLE_DATA/San Francisco Crime.jmp" );
Row State( dt1, 1 );
```

Selected(rowstate)

説明

選択されている状態なのか、選択されていない状態なのか（1または0）を戻す、もしくは設定する。

Selected State(num)

説明

選択の状態を指定された数値（*num*）に設定した行属性を戻す。

Shade State(num)

説明

濃淡の状態を指定された数値（*num*）に設定した行属性を戻す。

Where(<dt>, clause)

説明

指定された節にしたがって項目をフィルタリングし、インデックスを戻す。節には、比較関数または条件文を指定できます。同じ節の中で列・行列・リストを混ぜて照合することもできます。

Where()によって戻されるインデックスは、大量のリスト・行列・列を高速に処理できるよう最適化されています。

必須の引数

clause 比較関数または条件文。

オプションの引数

<dt> 評価時に現在のデータテーブルを変更する。

例

```
Names Default To Here( 1 );
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Select Rows( [2 4 6] ) << Exclude( 1 );
Where( Excluded() );
Where( !Excluded() );
```

SQL関数

メモ: データベーステーブル名で、$# -+/%()&|;?の文字を含むものはサポートされていません。

As SQL Expr(x, <style>)

説明

式を、SQLのSelectステートメントで使用できるコードに変換し、文字列で戻す。リテラルな式を指定する場合には、Expr(...)で、その式を囲んでください。また、式が変数name内に格納されている場合には、NameExpr(name)と指定してください。このように指定しないと、式が評価された後の結果が渡されます。

戻り値

有効なSQL構文に変換された式を含む引用符付き文字列。構文は、SQLのSelectステートメントで使用できます。

New SQL Query(Connection ("ODBC:connection_string"), Select(Column("column", "t1")), From(Table("table", <Schema("schema")>, <Alias("t1")>)), <Options(JMP 12 Compatible(1)|JMP 13 Compatible(1)|Run on Open(1)))>

New SQL Query(Connection("ODBC:connection_string;"), Custom("SELECT col1, col2, col3 FROM table;")), <Options(JMP 12 Compatible(1)|JMP 13 Compatible(1)|Run on Open(1))>

指定した接続、列、データテーブル、または独自のSQLクエリーのSQLクエリーオブジェクトを作成する。

戻り値

クエリー対象のデータを含むデータテーブル。そのデータテーブルには、クエリーを変更および更新するための引用符付きSQLクエリー文字列とテーブルスクリプトが含まれています。

引数

Connection ODBC接続のための引用符付き文字列。

Select 選択したい列とその別名。

From クエリー対象のテーブル、および（オプションで）スキーマと列の別名。

Custom 指定したテーブルの列を選択するSQLステートメント。

Version クエリーを開くのに必要なJMPの最低バージョン。この条件が満たされない場合は、互換性に関するメッセージがログに書き込まれ、クエリーは開かれません。

Options ブール値。［クエリービルダー］の環境設定で、JMP 12との互換性を維持するオプションを選択すると（または、［クエリービルダー］の赤い三角ボタンのメニューで同等のオプションを選択すると）、生成されたスクリプトにJMP 12 Compatibleが挿入されます。このオプションを指定すると、互換性に問題があるJMP 13のクエリーも、JMP 12で実行することができます。クエリーを編集モードで開くのではなく、開くと同時に実行するには、Run on Open(1)を指定します。

例

```
New SQL Query(
   Connection(
      "ODBC Connection String..."
   ),
   QueryName( "g6_Movies" ),
   Select( Column( "ItemNo", "t1" ), Column( "LengthMins", "t1" ), Column( "Genre",
   "t1" ) ),
   From( Table( "g6_Movies", Schema( "SQBTest" ), Alias( "t1" ) ) )
) << Run Background( On Run Complete( dt = queryResult ) );

Show( dt );
```

メモ

- クエリービルダーは、On Run Complete()スクリプトのコンテキストにおいてqueryResultというシンボルを作成します。これは、クエリーによって読み込まれたデータテーブルへの参照です。queryResultにより、後で使用するためにデータテーブルをグローバル変数に割り当てることができます。
- New SQL Query()は、クエリーの実行後に必ずODBC接続を閉じます。
- New SQL Query()の実行後、作成したデータベース接続が［**データベース**］**>**［**テーブルを開く**］ウィンドウに使用可能な接続として表示されます。ただし、［**データベース**］**>**［**テーブルを開く**］ウィンドウの接続は、そのデータベースへの接続方法の追跡記録に基づいて表示されるだけで、ODBC接続はすべて閉じた状態になります。
- New SQL Query()は、クエリ時の実行後に必ず接続を閉じるため、New SQL Query()で確立されたODBC接続を閉じるコマンドはありません。
- New SQL Query()の実行後、データベース接続は［**データベース**］**>**［**テーブルを開く**］ウィンドウに使用可能な接続として表示されます。JMPは、そのデータベースへの接続方法を追跡し、記録しています。ODBC接続自体が開いているわけではありません。

Query(<<dt1|Table(dt1, alias1)>, ..., <dtN, aliasN)>>, <private | invisible>, <scalar>, sqlStatement)

説明

選択されたデータテーブルに対し、SQLクエリーを実行する。

戻り値

クエリーの結果（データテーブルまたは1つの値）

引数

dt1, dtN（オプション）データテーブルに割り当てられた変数。

Table（オプション）データテーブルへの参照を渡す。

alias1, aliasN データベーステーブルの別名を指定する。

private（オプション）結果のデータテーブルを表示しない。プライベートのデータテーブルを使用すると、データへのアクセスが高速化しますが、データテーブルのデータを保持するために必要なメモリが少なくなるわけではありません。

invisible（オプション）データテーブルを非表示にする。データテーブルは、「JMPホームウィンドウ」と［ウィンドウ］メニューにのみ表示されます。なお、非表示のデータテーブルは、明示的に閉じるまでメモリ内にとどまるため、不要になったものは閉じるよう注意してください。非表示のテーブルを明示的に閉じるには、Close(dt)を実行します。ここで、*dt*は、データテーブル参照の変数です。

scalar（オプション）クエリーが1つの値を戻すよう指定する。

sqlStatement（必須）SQLステートメント（通常はSELECTステートメント）。SQLステートメントは、必ず、最後の引数として指定します。

例

次の例では、15歳以上の生徒のデータをすべて選択します。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
result = Query( Table( dt, "t1" ), "SELECT * FROM t1 WHERE 年齢 > 14;" );
```

次も参照

「JMPクエリーで使用可能なSQL関数」

統計関数

Arc Finder(X(col), Y(col), Group(lot, wafer))

説明

点のデータにおいて円弧を見つけ、円弧を示す新しい列を作成する。

例

```
dt = Open( "$SAMPLE_DATA/Wafer Stacked.jmp" );
Arc Finder(
   Group( :Lot, :Wafer ),
   X( :X_Die ),
   Y( :Y_Die ),
   Min Distance( 12 ), // 弧を定義する3点の間の最小距離
   Min Radius( 15 ), // 許容できる弧の最小半径
   Max Radius( 2000 ), // 許容できる弧の最大半径
```

```
    Max Radius Error( 2 ), // 点を追加する近さ
    Min Arc Points( 5 ), // 何個の点で弧を定義するか
    Number of Searches( 500 ), // ランダムなプローブの数
    Max Number Arcs( 3 ) // 探す弧の数
);
dt << Color or Mark by Column( :Arc Number );
dt << Graph Builder(
    Size( 1539, 921 ),
    Variables( X( :X_Die ), Y( :Y_Die ), Wrap( :Lot_Wafer Label ), Color( :Arc
    Number ) ),
    Elements( Points( X, Y, Legend( 6 ) ) )
);
```

メモ

- この関数は、30～50個のユニットを持つデータを対象とします。
- この関数は、興味の対象となる不適合部分が記録されたデータを分析するのに適しています。
- 点の密度が高い場合には適していません。

ARIMA Forecast(column, length, model, estimates, from, to)

説明

指定のモデルと予測値を使って、指定の列にある指定の行の予測値を戻す。

戻り値

引数 *from* と *to* によって指定された範囲の、*column* 列に対する予測値のベクトル

引数

column データテーブルの列。

Length 使用する列内の行数。

model 時系列モデルオプションのメッセージ。

estimates 予測に用いるモデルの係数を表す名前付き値のリスト。時系列プラットフォームにて、ARIMAモデルをあてはめて、予測値を保存したときにも、このリストは生成されます。

from, to 値の範囲。通常、*from* には、1以上から *to* 以下の整数のいずれかを指定します。*from* が0以下かつ *to* 以下の場合、結果は、実測値に対する予測値になります。

Best Partition(xindices, yindices, <<Ordered, <<Continuous Y, <<Continuous X)

説明

最適なグループ分けを探す関数。試験的な関数。

戻り値

リスト

引数

xindices, yindices 同次元の行列。

Col Cumulative Sum(name, <By var, ...>)

Cumulative Sum(name)

説明

現在の行までの累積和を戻す。Col Cumulative Sumは、By列をサポートしていますが、事前にBy列で並べ替えをしておく必要はありません。

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

Col Maximum(name, <By var, ...>)

Col Max(name)

説明

指定された列の全行における最大値を計算する。複数の評価を迅速に行えるよう、結果は内部にキャッシュされます。

戻り値

列の最大値

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Mean(name, <By var, ...>)

説明

指定された列の全行における平均値を計算する。複数の評価を迅速に行えるよう、結果は内部にキャッシュされます。

戻り値

列の平均値

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Median(name, <By var, ...>)

説明

指定された列の全行における中央値を計算する。複数の評価を迅速に行えるよう、順序は内部にキャッシュされます。

戻り値

列の中央値

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Minimum(name, <By var, ...>)

Col Min(name)

説明

指定された列の全行における最小値を計算する。複数の評価を迅速に行えるよう、結果は内部にキャッシュされます。

戻り値

列の最小値

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Mode(name, <By var, ...>)

説明

指定された列の全行における最頻値を計算する。複数の評価を迅速に行えるよう、順序は内部にキャッシュされます。

戻り値

列の最頻値

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Moving Average(name, options, <By var, ...>)

Moving Average(name, options)

説明

指定された期間で、現在の行における移動平均を戻します。Col Moving AverageはBy列をサポートしています。

引数

name 列名。

Weighting(1|0|n) 必須の位置引数。値への重みの付け方を指定する。1の場合、すべての項に等しい重みを加える。0の場合、線形に増加する重みを加える。その他の値の場合は、その値を指数加重移動平均のパラメータとして使用する（EWMAまたはEMA）。

Before(1|0|n) 位置引数。現在の項のいくつ前からの項を平均の範囲（ウィンドウ）に含めるかを指定する（現在の項を数に入れて）。デフォルトの値は-1で、過去のすべての項をすべて含めます。

After(1|0|n) 位置引数。現在の項のいくつ後までの項を平均の範囲（ウィンドウ）に含めるかを指定する（現在の項を数に入れて）。デフォルトの値は0で、後の項をまったく含めません。

Partial Window is Missing 位置引数（ブール値）。欠測値の扱いを指定する。デフォルトでは、欠測値は無視されます。0は、欠測値のある期間の平均を計算します。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

例

```
// 5つの項の移動平均を、等しい重みで求める
Col Moving Average( x, 1, 4 );

// 過去のすべての項の移動平均を、線形に増加する重みを加えて求める
Col Moving Average( x, 0 );

// 現在の項に前後の2項つを含む5項目の三角移動平均を求める
Col Moving Average( x, 0, 2, 2 );

// 過去のすべての項の指数移動平均を求める
Col Moving Average( x, 0.25 );
```

Col N Missing(name, <By var, ...>)

説明

指定された列の全行における欠測値の個数を求める。複数の評価を迅速に行えるよう、結果は内部にキャッシュされます。

戻り値

列の欠測値の個数

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Number(name, <By var, ...>)

説明

指定された列の全行における非欠測値の個数を求める。複数の評価を迅速に行えるよう、結果は内部にキャッシュされます。

戻り値

列の非欠測値の個数

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Quantile(name, p, <ByVar>)

説明

指定された列の行全体における、下側累積確率*p*に対する分位点を求める。複数の評価を迅速に行えるよう、結果は内部にキャッシュされます。

戻り値

列の分位点

引数

name 列名。

p 分位点を求めたい下側累積確率*p*。0～1の範囲で指定します。

ByVar（オプション）Byグループ。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
Col Quantile( :Name("身長(インチ)"), .5 );
```

63

戻り値の「63」は、「身長(インチ)」列の50%点、つまり中央値（メディアン）です。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Rank(column, <ByVar, ...>, <<tie("average"|"arbitrary"|"row"|"minimum"))

説明

最小値を1位、最大値を最後の順位として、各行に順位をつける。デフォルトでは、同順位のデータ値には、恣意的な順位が与えられます。

引数

column 順位付けされる列。

ByVar（オプション）グループごとに統計量を計算するにはBy変数を指定する。

<<tie 同じ値が複数ある場合、順位の付け方を決定する。[33 55 77 55]というデータの場合、33が1位、77が4位となり、2つの55については、順位が定まらない。averageを指定すると、両方とも、平均順位の2.5位になる。arbitraryを指定すると、2位と3位を任意に割り当てる（JMP 12ではこの方法で処理されていた）。rowを指定すると、元のデータの順番に従う（1つ目の55が2位、2つ目の55が3位となる）。

minimumを指定すると、両方に上位の順位（2位）を割り当てる。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Simple Exponential Smoothing(column, alpha, <ByVar>)

説明

現在の行から*alpha*を平滑化の重みとした1重指数平滑化法の予測値を戻す。

引数

column 時系列の観測値の列。

alpha 平滑化の重み。

ByVar（オプション）グループごとに予測値を計算するにはBy変数を指定する。By変数の順序を整えておく必要はありません。

メモ

行*t*の予測値は、次のように、求められます。

予測値[t] = alpha * 観測値[t-1] + (1-alpha) * 予測値[t-1]

ただし、最初の予測値は、予測値][1] = 観測値[1]となります。

Col Standardize(name,<By var, ...>)

説明

指定された列の全行を対象に、平均値を引いて標準偏差で割った値を算出する。

戻り値

標準化したデータ値

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数を指定した場合、値は、対応するBy変数グループの平均と標準偏差で標準化されます。

メモ

標準化とは、データから平均を引いて、それを標準偏差で割ることです。そのため、次の2つのコマンドは同じ結果になります。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << New Column( "stdht", Formula( Col Standardize( :Name("身長(インチ)") ) ) );
dt << New Column( "stdht2",
   Formula( (:"身長(インチ)"n - Col Mean( :"身長(インチ)"n )) / Col Std Dev( :"身長
   (インチ)"n ) )
);
```

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Std Dev(name,<By var, ...>)

説明

指定された列の全行における標準偏差を計算する。複数の評価を迅速に行えるよう、結果は内部にキャッシュされます。

戻り値

列の標準偏差

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Col Sum(name,<By var, ...>)

説明

指定された列の全行における合計を計算する。全行が欠測値の場合、Col Sum関数は欠測値を戻します。複数の評価を迅速に行えるよう、結果は内部にキャッシュされます。

戻り値

列の合計

引数

name 列名。

By var（オプション）グループごとに統計量を計算するにはBy変数を指定する。By変数は、列の計算式またはFor Each Row()の中でこの関数を使用する場合にのみ指定できます。

メモ

データ値が列プロパティ（「欠測値のコード」など）によって割り当てられている場合、代わりに列に保存されている値の計算を基にするには、Col Stored Value()を使用します。

次も参照

「Col Stored Value(<dt>, col, <row>)」

Fit Censored(Distribution("name"), YLow(vector) | Y(Vector), <YHigh(vector)>, <Weight(vector)>, <X(matrix)>, <Z(matrix)>, <HoldParm(vector)>, <Use random sample to compute initial values(percent)>, <Use first N observations to compute initial values(nobs)>)

説明

打ち切りのあるデータに、指定された分布をあてはめる。

戻り値

パラメータ推定値、共分散行列、対数尤度、AICc、BIC、収束メッセージで構成されるリスト。『基本的な回帰モデル』を参照してください。

引数

Distribution("name") あてはめる分布の引用符付きの名前。

YLow(vector) | Y(Vector) 打ち切りがないデータの場合は、Yだけを指定し、YHighは指定しない。打ち切りがあるデータの場合は、YLowおよびYHighに、それぞれ、打ち切りの下限値と上限値を指定してください。

オプションの引数

YHigh(vector) 打ち切りの上限値を示すベクトル。打ち切りがある場合のみ、YLowとYHighの2つを指定してください。

Weight(vector) 重み値を示すベクトル。

X(matrix) 回帰モデルの位置に対する計画行列。

Z(matrix) 回帰モデルの尺度に対する計画行列。

HoldParm(vector) 固定するパラメータの配列。パラメータを固定する場合は非欠測値、自由パラメータとして推定する場合は欠測値を指定してください。このオプションは、「パラメータが、ゼロである」や「パラメータが、特定の値である」いう仮説に対する検定を、特定のパラメータに対して行いたいときに使ってください。

Use random sample to compute initial values(percent) 初期値の計算に使うオブザベーションの割合。データベクトルが大きい場合に指定します。

Use first N observations to compute initial values(nobs) 初期値の計算に使うオブザベーションの数。データベクトルの先頭から指定した数のオブザベーションを使用します。データベクトルが大きい場合に指定します。

Fit Circle(Xvec, Yvec)

説明

最小2乗法を使って、3つ以上の点を最適に通る円をあてはめる。点が3つしか指定されていない場合は、直接解が見つかるため、誤差平方和はゼロとなります。

戻り値

円の中心点のXおよびY座標、半径の長さ、誤差平方和を含むリスト

引数

Xvec 3つ以上の点のX座標のベクトル。

Yvec 3つ以上の点のY座標のベクトル。

構文

```
{Xcenter, yCenter, radius, SSE} = Fit Circle(Xvec, Yvec)
```

Hier Clust(x)

説明

データ行列xについて、Ward法により（データを標準化せずに）階層型クラスター分析を行った履歴を戻す。

引数

x データ行列。

IRT Ability(Q1, <Q2, Q3, ...Qn,> parmMatrix)

説明

項目反応理論のモデルで、*n*個の2値の項目と既知のパラメータを使用して、潜在変数のスコアを算出する。パラメータの行列はモデル内のパラメータと同じ数の行と、分析で使用する項目と同じ数の列を持っていなければなりません。

引数

Q1, Q2, ..., Qn *n*個の2値の項。

parmMatrix 項目反応理論モデルのパラメータの行列。

KDE(vector, <named arguments>)

説明

バンド幅を自動選択して、カーネル密度推定値を戻す。

引数

vector ベクトル。

オプションの名前付き引数

<<weights vectorと同じ長さのベクトル。負でない任意の実数を含めることができます。度数や重みなどを指定するときに用います。

<<bandwidth(n) 負でない実数。0を指定した場合、バンド幅は自動選択されます。

<<bandwidth scale(n) 正の実数。

<<bandwidth selection(n) バンド幅の自動選択方法として、0（Sheather and Jones）、1（正規分布参照）、2（Silvermanの経験則）、3（過平滑化）のいずれかを指定してください。

<<kernel(n) カーネル関数として、0（Gauss）、1（Epanechnikov）、2（双加重）、3（三角）、4（矩形）のいずれかを指定してください。

LenthPSE(x)

説明

ベクトルxの値からLenthの擬似標準誤差を求める。

引数

x ベクトル。

Max()

「Maximum(var1, var2, ...)」を参照してください。

Maximum(var1, var2, ...)

Max(var1, var2, ...)

説明

引数の最大値を戻す。または、引数として指定された1つの行列もしくはリストの中の最大値を戻す。

複数の引数を指定する場合は、すべてを数値またはすべてを引用符付き文字列にする必要があります。

Mean(var1, var2, ...)

説明

引数の算術平均を戻す。または、1つの行列もしくはリストの値の算術平均を戻す。

Median(var1, var2, ...)

説明

引数の中央値、または1つの行列もしくはリストの中央値を戻す。

Min()

「Minimum(var1, var2, ...)」を参照してください。

Minimum(var1, var2, ...)

Min(var1, var2, ...)

説明

引数の最小値を戻す。または、引数として指定された1つの行列もしくはリストの中の最小値を戻す。複数の引数を指定する場合は、すべてを数値にするか、すべてを引用符付き文字列にする必要があります。

N Missing(expression)

説明

指定された複数の変数における、欠測値の個数を戻す。

Number(var1, var2, ...)

説明

指定された複数の変数における、非欠測値の個数を戻す。

Product(i=initialValue, limitValue, bodyExpr)

説明

*limitValue*になるまで、すべての *i* について *bodyExpr* の結果を乗算し、積を戻す。

Quantile(p, arguments)

説明

引数の分位点 *p* を戻す。最初の引数には、0～1のスカラー値または行列を指定できます。arguments の引数も、1つの行列または1つのリストとして指定できます。

Range(var1, var2, ...)

説明

指定した引数における最小値と最大値を戻す。結果は、最小値と最大値を含む2要素の行ベクトルとして戻されます。

Robust PCA(X, <Lambda(2/sqrt(max(nrow, ncol)))>, <tolerance=1e-10>, <maxit(75)>, <Center(1)>, <Scale(1)>)

説明

一連の特異値分解と閾値処理を実行して、データの行列を低ランク近似行列と残差行列に分解する。

戻り値

A 低ランク近似行列

E 残差行列

S 特異値のベクトル

引数

X データ行列。

Lambda 残差行列の疎性（スパース性）を特定する値。0より大きい値を指定します。λの値が大きいほど、残差行列は疎になります。

tolerance 収束基準。

maxit 特異値分解の最大反復回数。

Center 特異値分解の反復を実行する前にデータを中央で揃える。

Scale 特異値分解の反復を実行する前にデータのスケールを設定する。

Std Dev(var1, var2, ...)

説明

指定された複数の変数における、標準偏差を戻す。

Sum(var1, var2, ...)

説明

指定された複数の変数における、合計を戻す。「Sum(.,.)」のように、すべての引数が欠測値の場合は、欠測値を戻します。

SSQ(x1, ...)

説明

すべての要素の平方和を戻します。引数には数値、行列、リストを指定できます。スカラー値が戻されます。欠測値は除外されます。

Summarize(<dt>, <by>, <count>, <sum>, <mean>, <min>, <max>, <stddev>, <corr>, <quantile>, <first>)

説明

データテーブルの要約統計量を求め、グローバル変数に格納する。

戻り値

なし

引数

dt（オプション）位置指定引数。データテーブルへの参照。この引数がデータテーブルを参照する変数でない場合は、データテーブルの式とみなされます。

その他の引数はすべてオプションで、任意の順序で指定できます。通常、各引数は変数に割り当てられるので、値の表示や、さらなる操作が可能です。

name=By(col | list | Eval) Byを指定すると、全体に対する1つの結果ではなく、Byに指定した列の各グループごとに結果が計算される。

Summarize YByX(X(<x columns>, Y (<y columns>), Group(<grouping columns>), Freq(<freq column>), Weight(<weight column>))

説明

大規模なデータセットに対し、すべての組み合わせで二変量の関係の統計量を計算する。

戻り値

YとXの組み合わせごとの*p*値と対数価値のデータテーブル

引数

X(col) あてはめるモデルで使用する因子列。

Y(col) あてはめるモデルで使用する応答列。

Group(gcol) あてはめるモデルで使用するグループ化列。

Freq(col) あてはめるモデルで使用する（各行の）度数列。

Weight(col) あてはめるモデルで使用する重要度（影響度）の列。

メモ

「応答のスクリーニング」プラットフォームと同じ働きをします。

Summation(init, limitvalue, body)

説明

*init*から*limitvalue*までのすべての整数について、指定された式（body）の結果を合計し、その値を戻す。

Tolerance Limit(1-alpha, p, n)

説明

標本サイズ*n*の標本から計算される平均のうち割合*p*だけが含まれるような区間を、信頼水準（*1-alpha*）で求める。

超越関数

Arrhenius(n)

説明

温度*n*を、Arrheniusモデルにおける説明変数の値に変換する。

戻り値

11604.5181215503/(n+273.15)

引数

n 温度（単位は摂氏）。

メモ

これは頻繁に使用される変換の1つです。

Arrhenius Inv(n)

説明

Arrenhius関数の逆関数。値*n*を摂氏の温度に変換する。

戻り値

(11604.5181215503/n)-273.15

引数

n Arrheniusモデルにおいて説明変数の値に変換された値。

メモ

これは頻繁に使用される変換の1つです。

Beta(a, b)

説明

$$\frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

戻り値

ベータ関数

引数

a, b 数値

Cytometry Logicle(x, T, W, M, A)

説明

サイトメトリー Logicle 変換を計算する。Logicle 変換の詳細については、Moore and Parks（2012）を参照してください。

Cytometry Logicle Inverse(y, T, W, M, A)

説明

逆サイトメトリー Logicle 変換を計算する。Logicle 変換の詳細については、Moore and Parks（2012）を参照してください。

Digamma(n)

説明

ガンマ関数（LGamma）の対数の導関数。

戻り値

*n*におけるディガンマ関数の値

引数

n　数値。

Exp(a)

説明

eを*a*乗する。

戻り値

e^a

引数

a　数値。

等価表現

e()^a

ExpM1(x)

説明

*Exp(x)-1*を戻す。*x*が非常に小さい場合に、より正確な計算結果を戻します。

Factorial(n)

説明

1から*n*までの全整数を掛ける。

戻り値

*n*の階乗

引数

n 任意の整数。

メモ

指定できる引数は1つだけです。

FFT({list}, <named arguments>)

説明

行列のリストに対して高速Fourier変換（FFT）を行う。

戻り値

複素数を表す1つまたは複数の行列のリストを引数とし、その最初の引数と同じ次元の、2つの行列のリストを戻す。

引数

List 1つまたは2つの行列を含むリスト。1つの行列で構成されている場合、その行列は実数とみなされます。2つの行列で構成されている場合、1番目は実数、2番目は虚数部分とみなされます。2つの行列は次元が同じで、どちらも行が2つ以上なければなりません。

オプションの名前付き引数

<<inverse(Boolean) 1（真）の場合、逆FFTが実行される。

<<multivariate(Boolean) 1（真）の場合、単変量FFTが1列ごとに実行される。0（偽）の場合、空間FFTが実行される。

<<scale(number) 戻り値に *number* で指定した乗数を掛け合わせる。

Fit Transform To Normal(Distribution("name"), Y(vector), <Freq(vector))

説明

データのベクトルに対し、正規分布へと変換するための分布をあてはめる。Johnson Sl、Johnson Sb、Johnson Su、一般化対数（GLog）といった分布をあてはめることができます。

戻り値

パラメータ推定値、共分散行列、対数尤度、AICc、収束メッセージ、変換値で構成されるリスト。『基本的な回帰モデル』を参照してください。

Gamma(t, <limit>)

説明

*x*に対するガンマ関数の値を戻す。

$$\Gamma(t) = \int_0^\infty x^{t-1} e^{-x} dx$$

戻り値

ガンマ関数の値

引数

`t` 数値または列。

`limit`（オプション）上限。デフォルトは∞。

メモ

`Gamma(t, limit)` は `Gamma(t)` と積分される式は同じですが、積分の範囲における上限を無限ではなくlimitとします。

LGamma(t)

説明

t の対数ガンマ関数（ガンマ関数の自然対数）を戻す。

Ln(n)

説明

n の自然対数（底 *e* の対数）を戻す。

Log(n, <base>)

説明

n の自然対数（底 *e* の対数）を戻す。オプションの第2引数を追加すると、別の底を指定できます。たとえば、`Log(n,3)` は、3を底とする *n* の対数です。Logの引数は、任意の数式をとることができます。式 `Log(e())` は1、`Log(32,2)` は5です。

Log10(n)

説明

n の常用対数（底は10）を戻す。

Log1P(n)

説明

x がきわめて小さいときに精度がより高くなることを除けば、Log(1 + *x*) と同じ。

Logist(x)

説明

1/(1+Exp(-x)) を戻す。定義域（-∞～+∞）が0～1に変換されます。この関数は、ロジスティック回帰に役立ちます。

Logist Percent(p)

説明

Logist()関数の結果を0～100のスケールで戻す。

Logit(p)

説明

log(p/(1-p))を戻す。

Logit Percent(p)

説明

Logit()関数と似ているが、引数を0～1ではなく0～100で指定する。

N Choose K(n, k)

説明

この関数は、一度にn個のうちからk個を選択する場合の組み合わせの数（「nCk」）を戻す。この組み合わせの数は、$n!/(k!(n-k)!)$のように階乗を使った式で計算することができます。たとえば、NChooseK(5,2)は10になります。$k > n$の場合は0になります。

引数

n 選択できる項目の数を戻す。この引数が0の場合、結果は0になります。

k 一度に選択される項目の数。この引数が0の場合、結果は1になります。

メモ

この関数の計算において、JMPではLGamma関数が内部的に使用されています。そのため、結果は必ずしも整数とは限りません。

Power(a, <b>)

a^b

説明

*a*を*b*乗する。

戻り値

*a*を*b*回掛け合わせた積

引数

a 変数、数値、または行列。

b （オプション）変数または数値。

メモ

Power()の場合、第2引数（*b*）はオプションで、デフォルト値は2です。Power(a)はa^2を戻します。

Root(n, <r>)

説明

*n*の*r*次の根を戻す。デフォルトでは*r*が2で、平方根を戻します。

SbInv(z, gamma, delta, theta, sigma)

説明

標準正規分布の変数を上下に有界なJohnson Sb分布の変数に変換する。

SbTrans(x, gamma, delta, theta, sigma)

説明

上下に有界なJohnson Sb分布の変数を標準正規分布の変数に変換する。

Scheffe Cubic(x1, x2)

説明

x1*x2*(x1-x2)の結果を戻す。3次の配合モデルの表記に対応しています。

SHASHInv(z, gamma, delta, theta, sigma)

説明

標準正規分布の変数をsinh-arcsinh（SHASH）分布の変数に変換する。変換は、σ*sinh((arcsinh(z)-γ)/δ)+θの式で計算されます。

SHASHTrans(x, gamma, delta, theta, sigma)

説明

sinh-arcsinh（SHASH）分布の変数を標準正規分布の変数に変換する。変換は、sinh(γ+δ*arcsinh((x-θ)/σ))の式で計算されます。

SlInv(z, gamma, delta, theta, sigma)

説明

標準正規分布の変数をJohnson Sl分布の変数に変換する。

SlTrans(x, gamma, delta, theta, sigma)

説明

Johnson Sl分布の変数を標準正規分布の変数に変換する。

Sqrt(n)

説明

*n*の平方根を戻す。

Squash(expr)

説明

関数1/ [1 + exp(*expr*)]を効率よく計算する。

Squish(expr)

説明

Squash(-expr)または $1/(1+e^{-\text{expr}})$ と同じ。

SuInv(z, gamma, delta, theta, sigma)

説明

標準正規分布の変数を有界でないJohnson Su分布の変数に変換する。

SuTrans(x, gamma, delta, theta, sigma)

説明

有界でないJohnson Su分布の変数を標準正規分布の変数に変換する。

Trigamma()

説明

*n*におけるトリガンマ関数の値を戻す。トリガンマ関数はディガンマ関数の導関数です。

三角関数

JMPの三角関数では、すべての角度引数をラジアンで入力します。

ArcCosH(x)

説明

逆双曲余弦関数の値を戻す。

戻り値

*x*に対する逆双曲余弦関数の値

引数

x 任意の数値、数値変数、または数値式。

ArcCosine(x)

ArCos(x)

説明

逆余弦関数の値を戻す。

戻り値

*x*に対する逆余弦関数の値。戻り値は、角度でラジアン単位

引数

x 任意の数値、数値変数、または数値式。

ArcSine(x)

ArSin(x)

説明

逆正弦関数の値を戻す。

戻り値

*x*に対する逆正弦関数の値。戻り値は、角度でラジアン単位

引数

x 任意の数値、数値変数、または数値式。

ArcSinH(x)

説明

逆双曲正弦関数の値を戻す。

戻り値

*x*に対する逆双曲正弦関数の値

引数

x 任意の数値、数値変数、または数値式。

ArcTangent(x1, <x2=1>)

ArcTan(x1 <x2=1>)

ATan(x1 <x2=1>)

説明

逆正接関数の値を戻す。

戻り値

正接関数（*x1/x2*）の逆関数を戻す（範囲は -Pi()/2 ～ Pi()/2）。

引数

x1 任意の数値、数値変数、または数値式。

x2=1 *atan2*を指定する。

ArcTanH(x)

説明

逆双曲正接関数の値を戻す。

戻り値

*x*に対する逆双曲正接関数の値

引数

x 任意の数値、数値変数、または数値式。

CosH(x)

説明

双曲余弦

戻り値

*x*の双曲余弦

引数

x 任意の数値、数値変数、または数値式。

Cosine(x)

Cos(x)

説明

余弦。

戻り値

*x*の余弦

引数

x 任意の数値、数値変数、または数値式。ラジアン単位での角度。

Sine(expr)

Sin(expr)

説明

正弦。

SinH(expr)

説明

双曲正弦。

Tangent(expr)

Tan(expr)

説明

正接。

TanH(expr)

説明

双曲線正接。

ユーティリティ関数

Add(a, b, ...)

a+b+...

説明

リストされた引数の値を加算する。どの引数も変更されません。

戻り値

列の合計

引数

Add(): カンマで区切った、複数の変数、数値、または行列。

a+b: 任意の変数、数値、または行列。

メモ

– 引数はいくつでも使えます。引数が指定されていない場合、Add() は 0 を戻します。

– Add()は、いずれかの引数の評価が欠測値の場合、欠測値を戻します。欠測値を無視するには、Sum()を使用してください。

次も参照

『スクリプトガイド』

「Sum(var1, var2, ...)」を参照してください。

Beep()

説明

警告音を鳴らす。

戻り値

なし

BLOB MD5(blob)

説明

引数*blob*から、16バイトのBLOBを生成する。

メモ

戻り値の16バイトのBLOBは、引数のBLOBから計算されたMD5チェックサム（ハッシュ値）です。

BLOB Peek(blob, offset, length)

説明

引数*blob*から指定されたバイトだけ抜き出し、新しいBLOBを作成する。

戻り値

BLOBオブジェクト

引数

blob BLOB（binary large object）。

offset BLOBの先頭から何バイト目以降を抜き出すかを指定する整数。最初のバイトはオフセットが0で、2番目のバイトはオフセットが1。

Length オフセットから何バイト目までを抜き出すかを指定する整数。

Build Information()

説明

ビルト日時、リリースビルトとデバッグビルトの区別、および製品名を、カンマで区切った引用符付き文字列で戻す。

Caption({*h*, *v*}, "text", <Delayed(seconds)>, <Font(font)>, <Font Size(size)>, <Text Color("color")>, <Back Color("color")>, <Spoken(Boolean)>)

説明

指定のテキスト（*text*）を含んだキャプションウィンドウを指定の位置 *{h,v}* に表示する。キャプションの表示を *seconds* だけ遅らせたり、音声を出したりすることもできます。フォントの種類、サイズ、色、および背景色を指定することもできます。

戻り値

なし

引数

{h, v} 2つの値のリスト。*h* は、モニタの左上からの横方向の位置をピクセルで表したもの。*v* は、モニタの左上からの縦方向の位置をピクセルで表したもの。

text キャプションウィンドウに表示される引用符付き文字列または文字列への参照。

Delayed(seconds)（オプション）*seconds* は、文字列をキャプションウィンドウに表示するまでの遅延時間。このオプションを設定すると、このキャプションおよび後続のすべてのキャプションは指定の秒数が経ってから表示されます。

Font(font) フォントの種類を指定する。

Font Size(size) フォントのサイズを指定する。

Text Color("color") テキストの色を指定する。

Back Color("color") 背景色を指定する。

Spoken(Boolean) テキスト（*text*）を音声で読み上げる。現在の設定（オンまたはオフ）は、Spoken設定を含む別のCaptionステートメントによって変更されるまで有効となります。

Column Dialog(<var = ColList("*label*", <Min Col(*n*)>, <Max Col(*n*)>, <Width(*n*)>, <Data Type(*"Numeric"*|*"Character"*|*"Any"*)>, <Modeling Type({*"Continuous"*, *"Nominal"*, *"Ordinal"*, *"None"*, *"Multiple Response"*, *"Unstructured Text"*, *"Vector"*})>)>, <var = Combo Box("*text*")>, <var = RadioButtons("*label*")>, <HList(*display box*)>, <VList(*display box*)>, <LineUp(*n*, *display box*)>, <Text Box("*label*")>, <var = EditText(*string*)>, <var = EditNumber(*n*)>, <var = Check Box("*label*", *boolean*), <Window Title("*title*")>, <Window Icon("*icon string*"|"*path*")>, <Dialog Description(*string*)>, <Help Script(script)>)

説明

カスタマイズ可能なダイアログを開き、現在のデータテーブルから列を選ぶよう促します。

引数

ColList("*label*", *arguments*) データテーブル列の選択に使用される列リストのフィールドを戻す。ColList()のパラメータは次の*引数*で定義されます：Min Col()、Max Col()、Width()、Data Type()、Modeling Type()。

Min Col(*n*) 列リストで選択できるデータ列の最小数を指定する。

Max Col(*n*) 列リストで選択できるデータ列の最大数を指定する。

Width(*n*) 列リストの幅をピクセルで設定する。

Data Type(*string*) ダイアログの列リストに指定できる列のデータタイプを指定する。オプションは、*"Numeric"*（数値）、*"Character"*（文字）、*"Any"*（すべて）。

Modeling Type({*string*}) 列リストに指定できる列の尺度を指定する。オプションは、*"Continuous"*（連続尺度）、*"Nominal"*（名義尺度）、*"Ordinal"*（順序尺度）、*"None"*（なし）、*"Multiple Response"*（多重応答）、*"Unstructured Text"*（非構造化テキスト）、*"Vector"*（ベクトル）。複数の尺度を指定できます。

HList(*display box*) 指定された引数の *display box*を横に並べたものを戻す。

VList(*display box*) 引数の *display box*を縦に並べて戻す。

LineUp(*n*, *display box*) 引数の *display box*を *n*列に並べて戻す。

Check Box("*label*", *boolean*) チェックボックスフィールドを戻す。*boolean*引数は、ダイアログの作成時にボックスにチェックが入った状態にするかどうかを定義します。

Combo Box("*text1*", "*text2*", ...) *text*引数で指定されたテキストを選択肢とするコンボボックスフィールドを戻す。

RadioButtons("*label1*", "*label2*", ...) *label* 引数で指定されたテキストをラベルとするラジオボタンを戻す。

Text Box("*label*") ユーザにテキストを入力してもらうためのテキストボックスを戻す。

EditText(*string*) Text Boxオブジェクトにウォーターマークとして表示されるテキストを指定する。

EditNumber(*n*) Text Boxオブジェクトにウォーターマークとして表示される数値を指定する。

Window Title("*title*") ダイアログウィンドウのタイトルを指定する。

Window Icon("*icon string*"|"*path*") ウィンドウのタイトルバーに表示されるアイコンを指定する。JMPのアイコン名またはアイコンのパスを示す文字列。GIF、JPG、PNG、BMP、TIFなどの一般的なグラフィック形式に対応しています。

Dialog Description(*string*) ダイアログウィンドウのタイトルバーの下に表示される文字列（string）を指定する。

Help Script(*script*) ヘルプボタンを表示する。このボタンがクリックされると、指定したスクリプト（script）が実行される。

例

```
Names Default To Here( 1 );
Open( "$SAMPLE_DATA/Consumer Preferences.jmp" );
Column Dialog(
   ex y = ColList( "Y",
      Min Col( 1 ),
      Max Col( 2 ),
      Data Type( "Numeric" )
   ),
   ex x = ColList( "X",
      Max Col( 1 ),
```

```
            Modeling Type( {"Continuous", "Multiple Response"} )
         ),
         Line Up( 2,
            Text Box( "Alpha" ), ex = EditNumber( .05 ),
            Text Box( "Beta" ), ey = EditText( "xyz" )
         ),
         HList( cb = Check Box( "check", 1 ) ),
         HList( combo = Combo Box( "option1", "option2" ) ),
         HList( rb = RadioButtons( "a", "b" ) ),
         Window Title( "Custom Launch Dialog" ),
         Window Icon( "RowState" ), //icon stringは画像ファイルのフルパスでもよい。
         Dialog Description(
            "The dialog before a groundbreaking discovery!"
         ),
         Help Script( Web( "http://www.jmp.com/" ) )
      );
```

Datafeed()

「Open Datafeed()」を参照してください。

Debug Break()

JSLデバッガが開いているとき、この関数はスクリプト内のその時点でJSLスクリプトの実行を停止します。この関数は、ユーザが指定した条件の下で、デバッガで追跡調査をしているときに便利です。JSLデバッガが起動していない場合、この関数は実行されません。

Decode64 BLOB(*string*)

説明

Base64でエンコードされた引用符付き文字列をBLOBにデコードする。

戻り値

BLOB

引数

`string` Base64エンコーディングでエンコードした引用符付きの文字列。

例

```
Decode64 BLOB( "dGhlIHF1aWNrIGJyb3duIGZveA==" );
   Char To BLOB( "the quick brown fox", "ascii~hex" )
```

Decode64 Double(*string*)

説明

Base64エンコーディングでエンコードした引用符付きの文字列から浮動小数点数を作成する。

戻り値

浮動小数点数

引数

`string` Base64エンコーディングでエンコードした引用符付きの文字列。

Disable JMP Live URL(url)

説明

ユーザが発行先として使用できないURLを指定する。この環境設定は、jmpStartAdmin.jslでのみ指定可能です。

jmpStartAdmin.jslの場所については、『スクリプトガイド』の「プログラム例の紹介」章を参照してください。

あるURLが`Disable JMP Server()`と`Enable JMP Server()`の両方のリストに入っている場合、そのURLには発行できません。

例

```
Disable JMP Live URL( "https://public.jmp.com" )
```

Divide(a, b)

Divide(x)

a/b

説明

*a*を*b*で割る。引数が1つだけの場合（`divide(x)`）、1を*x*で割ります。

戻り値

a/*b*の商、または、引数が1つだけの場合は、*x*の逆数(1/*x*)

引数

`a, b, x` 変数、数値、または行列。

メモ

両引数が行列のときは、行列の割り算を行います。

Empty()

説明

計算式エディタで空のボックスを作成するのに使用される。

戻り値

空の値

引数

なし

Enable JMP Live URL(url)

説明

ユーザが発行先として使用できるURLを指定する。この環境設定は、jmpStartAdmin.jslでのみ指定可能です。

jmpStartAdmin.jslの場所については、『スクリプトガイド』の「プログラム例の紹介」章を参照してください。

あるURLがEnable JMP Server()とDisable JMP Server()の両方のリストに入っている場合、そのURLには発行できません。

例

```
Enable JMP Live URL( "https://public.jmp.com" )
```

Encode64 BLOB(x)

説明

BLOBをBase64の引用符付き文字列にエンコードする。

戻り値

Base64エンコーディングでエンコードした引用符付きの文字列。

例

```
Encode64 BLOB( Char To BLOB( "the quick brown fox" ) );
   "dGhlIHF1aWNrIGJyb3duIGZveA=="
```

Encode64 Double(n)

説明

浮動小数点数からBase64エンコーディングでエンコードした引用符付き文字列を作成する。

戻り値

Base64エンコーディングでエンコードした引用符付きの文字列

引数

n　浮動小数点数。

Faure Quasi Random Sequence(nDim, nRow)

説明

Faure数列の準乱数を生成する。この準乱数は、Space Filling（空間充填）の分野で使われることがある。

Get Addin("id")

説明

*id*によって指定された登録済みアドインを取得する。

戻り値

アドイン用スクリプト可能オブジェクト。指定されたIDのアドインが見つからない場合は空を戻します。

引数

"id" インストールされたアドインのID。

Get Addins()

戻り値

登録されているアドインすべてのリスト

Get Addr Info("address", <port>)

説明

指定されたアドレス名を、数値アドレスに変換する。

戻り値

引用符付き文字列のリスト。最初の要素はコマンド名（Get Addr Info）。2番目は結果（たとえば、コマンドが成功した場合は"ok"）。3番目はいくつかの情報を表す文字列のリスト。この3番目のリストのなかに、指定した名前に対応する数値アドレスが含まれている。

引数

address 名前を指定する引用符付き文字列（たとえば、"www.jmp.com"）。

port アドレスのポート。

Get Clipboard()

説明

コンピュータのクリップボードからテキストを戻す。クリップボードの内容がテキストではない場合、結果はヌルです。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Copy Table Script( "一変量の分布" );
s = Get Clipboard();
nw = New Window( "Script", Script Box( s ) );
```

Get Name Info("address", <port>)

説明

指定された数値アドレスを、アドレス名に変換する。

戻り値

引用符付き文字列のリスト。最初の要素はコマンド名（GetNameInfo）。2番目は結果（たとえば、コマンドが成功した場合は"ok"）。3番目はいくつかの情報を表す文字列のリスト。この3番目のリストのなかに、指定した数値アドレスに対応する名前が含まれている。

引数

address 数値アドレスを指定する引用符付き文字列（たとえば、"149.173.5.120"）。

port アドレスのポート。

Get Platform Preferences(<platform <(option, ...)> ... >)

Get Platform Preference(<platform <(option, ...)> ... >)

説明

指定されたプラットフォームの環境設定を戻す。

戻り値

プラットフォームの環境設定のリスト

引数

platform（オプション）プラットフォーム名。これが指定されていない場合、すべてのプラットフォームのすべての環境設定オプションの値を戻します。各プラットフォームに対して、1つまたは複数の環境設定を指定できます。

option（オプション）オプションの値。これが指定されていない場合、すべてのプラットフォーム環境設定の値を戻します。

メモ

表2.3に、プラットフォーム環境設定を取得するための構文を示します。

表2.3 Get Platform Preferences()の構文

構文	説明
Get Platform Preferences( )	すべてのプラットフォーム環境設定の現在のオプションの値を戻す。
Get Platform Preferences(Platform)	指定されたプラットフォーム環境設定の現在のオプションの値を戻す。
Get Platform Preferences(Platform(Option))	指定されたプラットフォームの現在のオプションの値を戻す。

表2.3 Get Platform Preferences()の構文（続き）

構文	説明
Get Platform Preferences(<<Changed)	すべてのプラットフォーム環境設定について、変更されたオプションの現在の値を戻す。
Get Platform Preferences(Platform(<<Changed))	指定されたプラットフォーム環境設定について、変更されたオプションの現在の値を戻す。
Get Platform Preferences(Platform(Option (<<Changed)))	指定されたプラットフォーム環境設定について、変更されたオプションの現在の値を戻す。

例

JMPのプラットフォームウィンドウまたはスクリプトを通じて、ユーザが複数の環境設定を変更したとします。

```
Platform Preferences(
   Distribution( Set Bin Width( 2 ), Horizontal Layout( 1 ) ),
   Model Dialog( Keep Dialog Open( 1 ) ),
   Graph Builder( Legend Position( "Bottom" ) )
);
```

変更されたすべてのプラットフォーム環境設定を取得するには、次のようにGet Platform Preferences(<<Changed)を使用します。

```
Get Platform Preferences( <<Changed );
   Platform Preferences(
      Distribution( Horizontal Layout( 1 ), Set Bin Width( 2, <<On ) ),
      Graph Builder( Legend Position( "Bottom", <<On ) ),
      Model Dialog( Keep dialog open( 1 ) )
   );
```

Get Preferences(<preference_name>)

Get Preference(<preference_name>)

説明

指定された環境設定の設定を戻す。

戻り値

環境設定の設定のリスト

引数

preference_name（オプション）環境設定名。これが指定されていない場合、すべての環境設定を戻します。指定された場合は、その環境設定の設定を戻します。

メモ

テキストデータファイル、Windowsのみ、Mac OSの設定、フォント、通信、スクリプトエディタ、JMPアップデートの環境設定には、JSLを使ってアクセスすることはできません。プラットフォームの環境設定の取得について詳しくは、「Get Platform Preferences(<platform <(option, ...)> ... >)」を参照してください。

Glue(expr1, expr2, ...)

expr1; expr2

説明

各引数を順番に評価する。

戻り値

最後に評価された引数の結果

引数

1つ以上の有効なJSL式。

メモ

Glue()では、式の区切りとしてセミコロンの方が一般的に使用されます。

Gzip Compress(blob)

説明

BLOBデータをgzip BLOBに圧縮する。

例

```
Gzip Compress(
   Char To BLOB(
      "random data does not usually compress well and may get larger"
   )
);
   Char To BLOB(

      "~1F~8B~08~00~00~00~00~00~00~0A~0D~CA~C1~0D~00~21~08~04~C0V~B6~B5~CDA~FC~80~
      5C~00c~EC^~E7=~C9)~E1~106~21~A1~85~19~8DU~8Bf~07_~F8~9FZ~85~ADfx~13~CE~83~A1
      ~0Dc~0E~CD~0B~94*~16~1E=~00~00~00",
      "ascii~hex"
```

Gzip Uncompress(blob)

説明

BLOBのgzipデータをBLOBに解凍する。

例

```
Gzip Uncompress( /*このデータは普通はGzip Compress()のものと考えられるが、Load Text
   File()とBLOBオプションを使った.gzファイルのものである可能性もある*/
   Char To BLOB(

   "~1F~8B~08~00~00~00~00~00~00~0A~0D~CA~C1~0D~00~21~08~04~C0V~B6~B5~CDA~FC~80~5C~
   00c~EC^~E7=~C9)~E1~106~21~A1~85~19~8DU~8Bf~07_~F8~9FZ~85~ADfx~13~CE~83~A1~0Dc~0
   E~CD~0B~94*~16~1E=~00~00~00",
      "ascii~hex"
   )
);
   Char To BLOB(
      "random data does not usually compress well and may get larger",
      "ascii~hex"
   )
```

Host Is("Mac"|"Windows")

説明

ホスト環境が、指定のOSと一致しているかどうかを判断する。一度にテストできる引数は1つだけです。

戻り値

現在のOS環境が引数と一致するときは1（真）、そうでなければ0（偽）

引数

"Windows" ホスト環境がWindowsの場合に真を戻す。

"Mac" ホスト環境がMacの場合に真を戻す。

Is Alt Key()

説明

Altキーが押されているときは1、そうでなければ0を戻す。

メモ

macOSの場合、Is Alt Key()はoptionキーをテストします。

Is Command Key()

説明

コマンドキーが押されているときは1、そうでなければ0を戻す。

Is Context Key()

説明

コンテキストキーが押されているときは1、そうでなければ0を戻す。

Is Control Key()

説明

Ctrlキーが押されているときは1、そうでなければ0を戻す。

メモ

macOSの場合、Is Control Key()はcommandキーをチェックします。

Is Option Key()

説明

Optionキーが押されているときは1、そうでなければ0を戻す。

Is Shift Key()

説明

Shiftキーが押されているときは1、そうでなければ0を戻す。

JMP Product Name()

説明

JMPの製品ライセンスに応じて"Standard"、"Pro"、または"Student"を戻す。

JMP Version()

説明

使用しているJMPのバージョンを戻す。

戻り値

「メジャーリリース番号.マイナーリリース番号<.修正リリース番号>」という文字列

引数

なし

Load DLL("path" <,AutoDeclare(Boolean | Quiet | Verbose)>)

Load DLL("path" <, Quiet | Verbose>)

説明

パス（*path*）で指定されたDLLをロードする。

引数

path DLLをロードする場所のパス名。

AutoDeclare(Boolean | Quiet | Verbose)（オプション）AutoDeclare(1)およびAutoDeclare(Verbose)は、ログにverboseメッセージを書き込む。AutoDeclare(Quiet)は、ロ

グウィンドウのメッセージを無効にします。このオプションを省略すると、ログにverboseメッセージが書き込まれます。

`Quiet | Verbose`（オプション）`Declare Function()`を使用した場合、このオプションは、ログウィンドウへのメッセージ出力のオフ（`Quiet`）とオン（`Verbose`）を切り替える。

次も参照

DLLがロードされたら、DLLオブジェクトメッセージを送り、それを操作します。これらのメッセージの詳細については、「ダイナミックリンクライブラリ（DLL）のメッセージ」を参照してください。

`Mail("address"|"addresses", "subject", "message", <"attachment filepath" | {"attachment 1 filepath", "attachment 2 filepath", ...}>)`

説明

（Windows）*address*で指定されたメールアドレスに、指定のタイトル（*subject*）とメッセージ（*message*）から成る電子メールを送る（MAPIを使用）。オプションで*attachment*引数を指定すると、1つまたは複数の添付ファイルを送信できます。`attachment`引数には、引用符付き文字列または文字列のリストを指定できます。

（macOS）ユーザのメールアプリケーション内で電子メールを作成する。ユーザは、メールのウィンドウで［**送信**］をクリックしなければなりません。Microsoft Outlookの場合は、メールへの添付ファイルの追加は手動で行う必要があります。

例

Windowsで、複数のファイルを添付してメールを送信するには、次のように指定します。

```
Mail(
   "yourname@company.com",
   "最新データとスクリプト",
   "本日更新されたデータテーブルとスクリプトを添付します。",
   {"$DOCUMENTS/wd.jsl", "$DOCUMENTS/survey.jmp"}
);
```

もしくは、次のように指定します。

```
list = {"$DOCUMENTS/wd.jsl", "$DOCUMENTS/survey.jmp"};
Mail(
   "yourname@company.com",
   "最新データとスクリプト",
   "本日更新されたデータテーブルとスクリプトを添付します。",
   list
);
```

複数の受信者にメールを送信するには、次のように指定します。

```
Mail(
   {"hername@company.com", "hisname@company.com"},
   "データベースの更新",
   "今日の販売データベースには、先月の数値が含まれています。"
);
```

メモ

macOSの場合、Mail()はYosemiteおよびそれ以降のオペレーティングシステムでのみ動作します。

Main Menu(*string*, <*string*>)

説明

引用符付き文字列（*string*）で示されたJMPメニューのコマンドを実行する。

引数

string メニューエディタに表示される、項目の内部名。たとえば、［ファイル］メニューの［新規作成］コマンドに対応する内部名は、"NEW"です。

string（オプション）コマンドの送り先であるウィンドウの名前。

例

Main Menu()には、完全パスまた部分パスを指定できます。部分パス名に一致するメニュー項目が複数ある場合は、最初に見つかったメニュー項目が実行されます。検索は、上位メニュー（ファイル、テーブル、DOEなど）から下位メニューの順に行われます。

```
Main Menu( "File:New:Data Table" ); // 完全パス
Main Menu( "Data Table" ); // 部分パス
```

Minus(a)

-a

説明

*a*の符号を反転する。

戻り値

*a*が正のときは-Abs(*a*)（a=3; -a=-3; Minus(a)=-3）

*a*が負のときはAbs(*a*)（a=-3; -a=3; Minus(a)=3）

*a*が0のときは0（a=0; -a=0; Minus(a)=0）

*a*が割り当てられていないときは欠測値

引数

a 変数または数値。変数には数値または行列が含まれている必要がある。

Multiple File Import(arguments)

説明

1つまたは複数のファイルをデータテーブルに読み込みます。このJSLは、「複数ファイルの読み込み」ウィンドウから［スクリプトをスクリプトウィンドウに保存］を選択することでも生成できます。

戻り値

複数ファイルの読み込みのオブジェクトを作成する。オブジェクトは、フォルダを設定するメッセージ、ファイルをフィルタリングするメッセージ、読み込みオプションを指定するメッセージを受け入れます。

引数

<<Create Window（オプション）ウィンドウを開き、現在の設定を表示する。

<<Set Folder 読み込みたいファイルを含むフォルダを指定する。

<<Set Name Filter（オプション）ファイルのファイル名または拡張子を指定する。名前のフィルタでは、*で0個以上の文字を表し、?で1個の文字を表します。*と?はピリオドにも一致します。デフォルト設定は、*.*、ですべてのファイルを表します。セミコロンで区切られたリストでフィルタリングしてファイルを指定できるため、セミコロンや「|」を含んだファイル名を読み込むには、「?」または「*」のようなワイルドカード文字を使う必要がある。

<<Set Name Enable(Boolean) 名前フィルタを有効にする。デフォルトはオフ。

<<Set Show Hidden(Boolean) Windowsで通常非表示になっているファイルを含める。デフォルトはオフ。

<<Set Size Filter（オプション）ファイルリストをファイルのサイズでフィルタリングする。サイズは、kB（キロバイト、すなわち1000バイト）としてリストに指定します。デフォルトの値は、ファイルリスト内にあるファイルのサイズの範囲に基づきます。

<<Set Size Enable(Boolean)（オプション）サイズフィルタを有効にする。デフォルトではオフ。

<<Set Date Filter（オプション）ファイルリストを日時でフィルタリングする。日時は、秒数としてリストに指定します。デフォルトの値は、ファイルリスト内にあるファイルの日時の更新日時の範囲になります。

<<Set Date Enable(Boolean)（オプション）日付フィルタを有効にする。デフォルトではオフ。

<<Set Add File Name Column(Boolean)（オプション）読み込んだファイルの名前の列を追加する。デフォルトではオフ。

<<Set Add File Size Column(Boolean)（オプション）読み込んだファイルのサイズの列を追加する。デフォルトではオフ。

<<Set Add File Date Column(Boolean)（オプション）読み込んだファイルの更新日時の列を追加する。デフォルトではオフ。

<<Set Import Mode(Row Per File|Row Per Line|CSV Data)（オプション）ファイルの読み込み形式を指定する。ファイル全体を1行に読み込む、ファイルの1行を1行に読み込む、CSV設定を使用して読み込む、のいずれか。デフォルトはCSV設定を使用しての読み込み。

<<Set Charset(Best Guess|utf-8|utf-16|us~ascii|windows-1252|x-mac-roman|x-mac-japanese|shift-jis|euc-jp|utf-16be|gb2312)（オプション）読み込むファイルの文字コード。デフォルトでは、環境設定ので指定されている文字セット（開くテキストファイルの文字コード）に設定されています。

<<Set Stack Mode(Stack Similar|TablePerFile)（オプション）ファイルの連結方法を指定する。デフォルトでは、類似したファイルを連結する（Stack Similar）が設定されています。（同じ列を持つファイルが1つのデータテーブルに連結されます。）

<<Set Subfolders(Boolean)（オプション）サブフォルダ内のファイルを含める。デフォルトはオフ。

<<Set CSV Has Headers(Boolean)（オプション）CSVファイルに列見出しの行があるかどうかを指定する。デフォルトはオン。

<<Set CSV Allow Numeric(Boolean)（オプション）数値データの列のデータタイプを数値に設定する。デフォルトはオン。

<<Set CSV First Header Line(*n*)（オプション）列見出しの開始行を指定する。デフォルトは1。

<<Set CSV Number of Header Lines(*n*)（オプション）列見出しの行数を指定する。デフォルトは1。

<<Set CSV First Data Line(n)（オプション）データの開始行を指定する。デフォルトは2。

<<Set CSV EOF Comma(Boolean)（オプション）フィールドがカンマで区切られていることを指定する。デフォルトはオン。

<<Set CSV EOF Tab(Boolean)（オプション）フィールドがタブで区切られていることを指定する。

<<Set CSV EOF Space(Boolean)（オプション）フィールドがスペースで区切られていることを指定する。

<<Set CSV EOF Spaces(Boolean)（オプション）フィールドが複数のスペースで区切られていることを指定する。

<<Set CSV EOF Other("")（オプション）その他の区切り文字を指定する。

<<Set CSV EOF CRLF(Boolean)（オプション）行がキャリッジリターンとラインフィードで区切られていることを指定する。デフォルトはオン。

<<Set CSV EOF CR(Boolean)（オプション）行がキャリッジリターンで区切られていることを指定する。デフォルトはオン。

<<Set CSV EOF LF(Boolean)（オプション）行がラインフィードで区切られていることを指定する。

<<Set CSV Semicolon(Boolean)（オプション）行がセミコロンで区切られていることを指定する。デフォルトはオフ。

<<Set CSV EOL Other("")（オプション）その他の行の区切り文字を指定する。

<<Set CSV Quote("")（オプション）引用符として使用する文字を指定する。デフォルトは二重引用符（\!"）。

<<Set CSV Escape("")（オプション）エスケープ文字を指定する。

<<Set Excel Add Sheet Name Column(Boolean)（オプション）データの読み込み元であるスプレッドシートの名前を持つ列を追加する。

<<Set Excel Best Guess(Boolean)（オプション）各スプレッドシートのデータを調べ、列名を推測する。これを設定すると、<<Set Excel Add Sheet Name Column以外のExcelパラメータは使用されない。

<<Set Excel Column Headers As Hierarchies(Boolean)（オプション）結合されたセルの情報を見出しとして再編成し、生成されたテーブルに配置する。

<<Set Excel Column Name Separator("")（オプション）複数のセルを連結して列見出しの名前にするときに、列名の区切り文字として使用する文字列を設定する。

<<Set Excel First Data Column(n)（オプション）データとして読み込む最初の列の番号を指定する。（先頭部分にある空白の列は除いて数える。）

<<Set Excel First Data Line(n)（オプション）データとして読み込む最初の行の番号を指定する。（先頭部分にある空白の行は除いて数える。）

<<Set Excel First Header Line(n)（オプション）列見出しの定義に使用される、最初の行の番号を指定する。（先頭部分にある空白の行は除いて数える。）

<<Set Excel Has Headers(Boolean)（オプション）<<Set Excel Header Lineおよび<<Set Excel Number of Header Linesで指定された値にしたがって列見出しを定義する。デフォルトはオフ。

<<Set Excel Import Color Cells(Boolean) セルの背景色を読み込む。デフォルトはオフ。

<<Set Excel Limit Column Type Detection(Boolean) 列のデータタイプを検出する際に一部の行だけをチェックする。これにより検出は高速になるが、列の先頭と末尾でデータタイプが異なる場合は、間違ったデータタイプを選択する可能性がある。

<<Set Excel Multiple Series Stack(Boolean) <<Set Excel Column Headers As Hierarchiesも有効な場合に列を積み重ねる。デフォルトはオフ。

<<Set Excel Number of Header Lines(n) スプレッドシート内で列見出しとして読み込む行数を指定する。

<<Set Excel Suppress Empty Columns(Boolean)（オプション）空の列が読み込まれないようにする。

<<Set Excel Suppress Hidden Columns(Boolean)（オプション）非表示の列が読み込まれないようにする。

<<Set Excel Suppress Hidden Rows(Boolean)（オプション）非表示の行が読み込まれないようにする。

<<Set Excel Worksheet Filter("")（オプション）読み込むワークシートを指定する。ワークシートのフィルタでは、*で0個以上の文字を表し、?で1個の文字を表します。*と?はピリオドにも一致します。デフォルト設定は*で、すべてのファイルを表します。セミコロンで区切られたリストでフィルタリングしてワークシートを指定できるため、セミコロンや「|」を含んだワークシートを読み込むには、「?」または「*」のようなワイルドカード文字を使う必要があります。

<<Set JSON Guess("Tall"|"Wide"|"Huge"|"Pandas") JSONファイルのデータ値を推測する方法を指定する。

<<Set JSON Method("Guess"|"JSON Settings") JSON値の読み込みにビルトインの識別機能とカスタムJSLのどちらを使うかを指定する。

<<Set JSON Settings JSONデータの読み込みに使う読み込むカスタムJSLを指定する。

<<Set PDF Method("Guess"|"PDF Settings") PDF値の読み込みにビルトインの識別機能とカスタムJSLのどちらを使うかを指定する。

<<Set PDF Settings PDFデータの読み込みに使う読み込むカスタムJSLを指定する。

<<Set XML Guess("Tall"|"Wide"|"Huge") XMLファイルのデータ値を推測する方法を指定する。

<<Set XML Method("Guess"|"XML Settings") XMLファイルの読み込みにビルトインの識別機能とカスタムJSLのどちらを使うかを指定する。

<<Set XML Settings XMLデータの読み込みに使う読み込むカスタムJSLを指定する。

<<Set Import Callback("")（オプション）読み込みプロセスの最終ステップで実行するカスタムコールバック関数を指定する。Multiple File Import()関数は、コールバック関数にMultiple File Importオブジェクトと開いたデータテーブルのリストを渡します。

<<Import Data データを読み込む。

例

```
mfi = Multiple File Import(
   <<Set Folder( "$SAMPLE_IMPORT_DATA" ),
   <<Set Name Filter( "UN*.csv" ), // このファイル名をもつファイルを読み込む
   <<Set Name Enable( 1 ) // ファイル名に対するフィルタを使用
)
<<Import Data();
```

Multiply(a, b, ...)

a*b*...

説明

すべての値を掛ける。どの引数も変更されません。

戻り値

nの階乗

引数

任意の変数、数値、または行列。

メモ

引数はいくつでも使えます。引数が指定されていない場合、Multiply()は1を戻します。

Name(*string*)

説明

名前。「名前」とは、ある項目をどう呼ぶを決めるものです。

- アルファベット文字またはアンダースコアで始まり、アルファベット文字、スペース、数学記号（Unicode）、特定の句読点（アポストロフィ、パーセント記号、ピリオド、バックスラッシュ、アンダースコア）のみが使われている名前であれば、スクリプトでそのまま使えます。
- そうでない名前は、Name()キーワードで指定する必要があります。

例

"name"nを用いることを推奨します。Name(*string*)は推奨されません。

```
"my-var-name"n = "hello";
Length( "my-var-name"n )
   5
```

New OAuth2 Token(account(*string*), client ID(*string*), client secret(*string*), refresh token(*string*), token URL(*string*))

説明

さまざまなWeb APIでデータに安全にアクセスするための、OAuth2トークンを作成する。

引数（必須）

`account` アクセスするアカウントのユーザ名、メールアドレス、またはID。

`client ID` APIキーとして動作するパブリックID。

`client secret` Client IDに対応するプライベートID。

`refresh token` アクセストークンを得るために使用するトークン。

`token URL` アクセストークンの入手元のURL。サービスごとに一意で、APIまたはOAuthページを通じてアクセスします。

引数（認証コード付与）

`redirect URL( "https://app.getpostman.com/oauth2/callback" )` アクセスコードを送り返すURL。自身の会社やサービスがURLを用意しているのでない限り、無料のPostmanアカウントを作成してこのリダイレクトを利用することをお勧めします。

`client secret( "1aB893cdDeFf2D" )` Client IDに対応するプライベートID。

`request auth( ... )` 認証コードフローを使用することを示す追加のパラメータ。`Auth URL()`が必要になります。サービスによっては、スコープが必要です。Fieldsを使ってカスタムフィールドを追加できます。

`scope( "spreadsheets email docs" )` スペースで区切られた、スコープのリスト。サービスごとに一意で、APIまたはOAuthページを通じてアクセスします。`Request Auth()`でのみ使用できます。

`auth URL( "https://www.example.com/oauth2/v1/authorize" )` 認証をリクエストするためのURL。サービスごとに一意で、APIまたはOAuthページを通じてアクセスします。`Request Auth()`でのみ使用できます。

引数（インプリシット付与）

`redirect URL( "https://app.getpostman.com/oauth2/callback" )` アクセスコードを送り返すURL。自身の会社やサービスがURLを用意しているのでない限り、無料のPostmanアカウントを作成してこのリダイレクトを利用することをお勧めします。

引数（リソースオーナー）

`password( "wordpass123" )` ユーザ名に対応するパスワード。

`client secret( "1aB893cdDeFf2D" )` Client IDに対応するプライベートID。

引数（カスタムデータ）

`fields( fields )` HTTP RequestのForm(fields(fields))と等価なカスタムフィールド。`New OAuth2 Token()`と`Request Auth()`の両方で指定できます。サービスが、OAuth 2.0標準で定義されていない情報を求めている場合のみ必要です。

headers(headers) HTTP RequestのHeaders(headers)と等価なカスタムの見出し。New OAuth2 Token()でのみ指定できます。サービスが、OAuth 2.0標準で定義されていない情報を求めている場合のみ必要です。

例

```
token = New OAuth 2 Token (
   User( "yourgoogleaccount@gmail.com" ),
   Refresh Token( "1a2b3c4e5F" ),
   Token URL( "https://www.example.com/oauth2/token" ),
   Client ID( "12ab" ),
   Client Secret( "3456dEfG" )
);
```

次も参照

クライアントシークレットやトークンURLなどの値を取得する方法について詳しくは、APIのマニュアルを参照してください。

Open Datafeed()

Datafeed()

説明

データフィードオブジェクトおよびウィンドウを作成する。

戻り値

データフィードオブジェクトへの参照

引数

引数は必須ではありません。ただし、通常は、Open Datafeed()内の引数によって、データフィードの基本操作を設定します。

Open Help("Help"|"Scripting Index", ...)

説明

指定したヘルプウィンドウを開く。

Parse XML(*string*, On Element("tagname", Start Tag(expr), End Tag(expr), Text))

説明

On Elementで指定されたXMLタグによって、XMLを解析する。

例

```
XMLData = "
<Book name='食べ物'>大百科
       <Chapter num='1'>果物
             <kind>リンゴ</kind>
             <kind>サクランボ</kind>
```

```
                <ps>デザート大好き！ </ps>
            </Chapter>
            <Chapter num='2'>パン
                <kind>小麦</kind>
                <kind>コーン</kind>
                <ps>サンドイッチ大好き！ </ps>
            </Chapter>
            <Chapter num='3'>野菜
                <kind>カボチャ </kind>
                <kind>キャベツ</kind>
                <ps>これだけは勘弁！ </ps>
            </Chapter>
        完全版
        </Book> ";

        // テキストを連結できるように変数を初期化
        title = "";
        subtitle = "";
        chap = "";
        chapnum = "";
        ps = "";

        Parse XML( XMLData,
            On Element( "Book",
        // Bookの開始時にname属性をキャプチャする
                Start Tag( title = XML Attr( "name" ) ),
        /* このBookには分割されたサブタイトルがあるので、テキストを結合する必要がある。
        結合されたテキストはendTagのところで使用される。Text(...)にJSLを指定する。*/
                Text( subtitle = subtitle || " -- " || Trim( XML Text() ) ),
        /* endTagによる変数の処理が終わったら、これらを最初の状態に戻す。
        これは、同じXMLで処理するBookがもう1つあった場合のため。*/
                endTag( Write( "\!n", title, " ", subtitle ); title = ""; subtitle =
           ""; )
            ),
            On Element( "Chapter",
        // Chapterの開始時に章の番号をキャプチャする
                Start Tag( chapnum = XML Attr( "num" ) ),
        /*Chapterのテキストを結合し、改行と余分なスペースを削除し、
        個々のテキストを単一のスペースで区切る。<kind>タグはこのParseXML指定では無視される。
        <kind>テキストは、他のOn Elementで使われていないため、このText(...)によって処理される*/
                Text( chap = chap || Trim( XML Text() ) || " " ),
        /* endTagによる変数の処理が終わったら、それらを最初の状態に戻す。
        これは、他にも何もない状態で始めなければならないChapterがあるため。*/
                endTag( Write( "\!n", chapnum, " ", chap, " ps: ", ps ); chapnum = "";
           chap = ""; ps = ""; )
            ),
```

```
        On Element( "ps", End Tag( ps = XML Text() ) )
);
   1 果物 リンゴ サクランボ  ps: デザート大好き！
   2 パン 小麦 コーン  ps: サンドイッチ大好き！
   3 野菜 カボチャ キャベツ  ps: これだけは勘弁！
   食べ物  -- 大百科 -- 完全版
```

```
Platform Preferences(platform(option(value)), ...)
Platform Preference(platform(option(value)), ...)
Set Platform Preferences(platform(option(value)), ...)
Set Platform Preference(platform(option(value)), ...)
```

説明

プラットフォームの環境設定を設定または解除する。

引数

`platform` 環境設定を行うプラットフォーム。

`option` オプションの名前。

`value` オプションの値。

メモ

表2.4に、プラットフォームの環境設定を行うための構文を示します。

表2.4 Platform Preferences()の構文

構文	説明

`Platform Preferences( <<Default )`
`Platform Preferences( <<Factory Default )`
`Platform Preferences( Default )`
	すべてのプラットフォーム環境設定をデフォルトの値にリセットする。

`Platform Preferences( Platform( <<Default ) )`
`Platform Preferences( Platform( <<Factory Default ) )`
`Platform Preferences( Platform( Default ) )`
	指定されたプラットフォーム環境設定をデフォルトの値にリセットする。

`Platform Preferences( Platform( option ( <<Default ) ) )` `Platform Preferences( Platform( option ( <<Factory Default ) ) )`	指定されたプラットフォームオプションをデフォルトの値にリセットする。
`Platform Preferences( Platform( option( value, <<On ) ) )`	指定されたプラットフォームオプションをオンにし、値を設定する。
`Platform Preferences( Platform( option( value, <<Off ) ) )`	指定されたプラットフォームオプションをオフにし、値を設定する。

例

次の式は、「一変量の分布」環境設定の［棒の幅の設定］オプションを選択（オンに）して、値を2に設定します。

```
Platform Preferences( Distribution( Set Bin Width( 2 ) ) );
```

次の式は、［棒の幅の設定］の値を変更し、このオプションを無効にします。

```
Platform Preferences( Distribution( Set Bin Width( 2, <<Off ) ) );
```

次の式は、［棒の幅の設定］の値をデフォルト値にリセットして、環境設定の選択を解除します。

```
Platform Preferences( Distribution( Set Bin Width( <<Default ) ) );
```

Polytope Uniform Random(samples, A, b, L, U, neq, nle, nge, <nwarm=200>, <nstride=25>)

説明

凸多面体上に一様乱数の点を生成する。

引数

Samples 生成される乱数の個数。

A 制約式の係数の行列。

B 制約式の右辺値。

L, U 変数の下限および上限。

neq 等号制約式の数。

nle「以下」を示す不等号制約式の数。

nge「以上」を示す不等号制約式の数。

nwarm（オプション）予備反復の回数。結果を出力する前に、何回、反復を行うかを指定します。

nstride（オプション）結果を出力する間隔。何回の1回のペースで結果を出力するかを指定します。

メモ

制約式は、等号制約、「以下」を示す不等号制約、「以上」を示す不等号制約の順に指定しなければなりません。

Preferences(pref1(value1), ...)

Preference(pref1(value1), ...)

Pref(pref1(value1), ...)

Prefs(pref1(value1), ...)

Set Preferences(pref1(value1), ...)

Set Preference(pref1(value1), ...)

説明

JMPの環境設定を設定する。

引数

Add Files Opened by Scripts to the Recent Files List(Boolean) スクリプトが開いたファイルをホームウィンドウの「最近使ったファイル」リストに追加するかどうかを指定する。

Analysis Destination(window) 新しい分析の結果をどこに出力するかを指定する。

Annotation Font("font", size, "style") レポートに表示される注釈のフォントを指定する。

Axis Font("font", size, "style") 軸ラベルのフォントを指定する。

Axis Title Font("font", size, "style") 軸タイトルのフォントを指定する。

Background Color({R, G, B} | <color>) ウィンドウの背景色を指定する。

Calculator Boxing(Boolean) 計算式エディタの中にボックスを作成し、大きい式の中の各項の階層がわかるようにする。

Conditional Formatting Rules レポート内のテキストを条件付きの形式で表示するためのルールを作成する。「例」を参照してください。

Data Table Font("font", size, "style") データテーブルのフォントを指定する。

Data Table Title on Output(Boolean) レポートにデータテーブル名のタイトルをつける。

Date Title on Output(Boolean) レポートに現在の日付のタイトルをつける。

Evaluate OnOpen Scripts("always"|"never"|"prompt") ユーザがデータテーブルを開いたときにOn Openテーブルスクリプトを実行するかどうかを指定する。デフォルトでは、確認のメッセージが表示されます。1つのJMPセッションで再度同じデータテーブルを開いた場合は、前回のユーザーの選択が適用されます。ただし、別のプログラムを実行するスクリプトは実行されません。

Excel Has Labels(Boolean) オンにすると、データの先頭行が列見出しと解釈される。

Excel Selection(Boolean) オンにすると、Excelワークブックからどの（空白ではない）Excelワークシートを読み込むか指定するよう、プロンプトが表示される。

File Location Settings(<Directory Type>("<path>"<,"initial directory">)) 次のディレクトリを指定できます。

Data Files Directory: データファイルのデフォルトの保存場所を設定する。

Help Files Directory: ヘルプファイルの保存場所を設定する。

Installation Directory: Windowsでは、次の場所がJMPのデフォルトのインストールフォルダに設定されている。

"C:¥Program Files¥JMP¥JMP¥18" または "C:¥Program Files¥JMP¥JMPPro¥18"

License File Path: JMPライセンスファイルのデフォルトの保存場所を設定する。

Preferences File Directory: 環境設定ファイルの保存場所を設定する。

Save As Directory: 名前を付けて保存の操作で使われるデフォルトの保存場所を設定する。

Foreground Color(color) ウィンドウの前面の色を指定する。

Formula Font("font", size, "style") 計算式エディタに用いるフォントを指定する。

Graph Background Color(color) グラフのフレーム内の背景色を指定する。

Graph Marker Size(size) マーカーを描くときのデフォルトの大きさを指定する。

Heading Font("font", size, "style") レポート中の表の、列見出しに使われるフォントを指定する。

Initial JMP Starter Window(Boolean) 起動時の「JMPスターター」ウィンドウの表示／非表示を指定する。

Initial Splash Window(Boolean) 起動時スプラッシュ画面の表示／非表示を指定する。

Maximum JMP Call Depth(size) JMPにおける呼び出しの最大の深さ（スタックサイズ）のデフォルトを設定する。JSLビルトイン関数、ユーザ定義の関数、またはRecurse()関数をこの深さまで呼び出せます。設定は、デフォルトでは256KB。

JMPが作成する各スレッドは、デフォルトで2MBのスタックがあります。呼び出しの最大の深さを増やすと、物理的な実行スタックがオーバーフローを起こす可能性があるので、JSLスクリプトがうまく動作する最良の値を見つけるまで、この環境設定を少量ずつ増加させてください。

Marker Font("font", size, "style") プロットで使用されるマーカーのフォントを指定する。

Monospaced Font("font", size, "style") モノスペースフォントを指定する。

ODBC Suppress Internal Quoting(Boolean) SQLステートメントに含まれるテーブル名や変数名に大文字、小文字、スペースが混在している場合に、内部で引用符を追加しない。

Outline Connecting Lines(Boolean) 同じレベルのアウトラインノードのタイトルを線でつないで見やすくする。

Print Settings(option(value), ...)「ページ設定」ウィンドウの印刷オプションを変更する。

Margins(<n>, <n>, <n>, <n>): 左、上、右、下の余白を設定する。センチ単位で指定します。

Margins(<n>): すべての余白を同じ値（単位はセンチ）に設定する。

Orientation("portrait" | "landscape"): 用紙の向きを変更する。

Headers(<"char">, <"char">, <"char">): 左、中央、右のヘッダに表示されるテキストを指定する。

Headers(<"char">): ヘッダに表示するテキストのみを指定する。

Footers(<"char">, <"char">, <"char">): 左、中央、右のフッタに表示されるテキストを指定する。

Footers(<"char">): フッタに表示するテキストのみを指定する。

Scale(<n>): 印刷倍率を増減する。

Show Explanations(Boolean) 分析の種類によっては、出力について説明するテキストをオプションで表示する。

Show Menu Tips(Boolean) メニューのヒントの表示／非表示を指定する。

Show Status Bar(Boolean) ステータスバーの表示／非表示を指定する。

Small Font("font", size, "style") 小さなテキストに用いるフォントを指定する。

Text Font("font", size, "style") レポートの中の一般的なテキストのフォントを指定する。

Thin Postscript Lines(Boolean) macOSのみ。ポストスクリプトプリンタに出力する場合の線の太さを、それ以外の場合より細くするよう指定する。

Title Font("font", size, "style") タイトルのフォントを指定する。引数にはフォント名（例: "Times"）、ポイント数、スタイル（"bold"、"plain"、"underline"、"italic"）があります。

Use Triple-S Labels as Headings(Boolean) オンにすると、ラベル名が列見出しと解釈される。例: Pref(Name("Use Triple-S Labels as Headings")(0)); は、この環境設定をオフにします。

例

次の式は、すべての環境設定をデフォルトの値にリセットします。

```
Preferences( "Default" );
Preferences( "Factory Default" );
```

次のスクリプトは、レポート内のテキストの条件付き表示形式のルールを作成します。

```
Preferences(
    Conditional Formatting Rules(
        RuleSet(
            RuleName( "警告" ),

            // 値が0でない場合、テキストを80%のグレーで表示する。
            NotEqualTo( Value( 0 ), Format( TextAlpha( 0.8 ) ) )
        )
    )
);
```

メモ

テキストデータファイル、Windowsのみ、Mac OSの設定、フォント、通信、スクリプトエディタ、JMPアップデートの環境設定には、JSLを使ってアクセスすることはできません。

次も参照

「Platform Preferences(platform(option(value)), ...)」

Register Addin("unique_id", "home_folder", <named_arguments>)

説明

JMPアドインを登録し、正常に登録されたら、そのアドインをロードする。

戻り値

成功した場合は、登録されたアドインを示すスクリプト可能オブジェクトを戻す。成功しなかった場合は、空を戻す。

引数

unique_id アドインの一意の識別子を示す引用符付き文字列。最大64文字まで使用できます。文字列の最初は文字でなければならず、文字、数字、ピリオド、下線を使用できます。一意である確率を上げるために、DNSの逆の名前を使用することをお勧めします。

home_folder アドインファイルを含むフォルダのファイルパスを示す引用符付き文字列。ファイルパスは、ホストのオペレーティングシステムのファイルパス要件を満たすものでなければなりません。

DisplayName("name")（オプション）制限のある一意のID名ではなく、JMPユーザインターフェースでのアドインの表示に使われる、わかりやすい表示名。

JMPVersion("version")（オプション）JMPの特定のバージョンを示す引用符付き文字列。デフォルト値は"All"で、この場合、アドインに対応しているすべてのJMPバージョンでアドインのロードおよび実行ができます。"Current"を指定すると、アドインの使用が現在のバージョンのみに制限されます。引用符付きのバージョン番号（"7"や"9"など）を指定すると、アドインの使用が単一のJMPバージョンに制限されます。

LoadsAtStartup(Boolean)（オプション）ブール値。デフォルト値は1（真）で、この場合、JMPの起動時にアドインがロードされます。0（偽）を指定すると、アドインは自動的にはロードされません。

LoadNow(Boolean) アドインを即座にロードする。

メモ

指定されたホームフォルダに「addin.def」という名前のファイルがあれば、Register Addin()関数に含まれていないオプションの引数すべてに、そのファイルの値が使用されます。

例

次の例では、第1引数で一意の識別子を指定し、第2引数でアドインのインストール先を指定しています。第3引数では、アドイン名が通常表示される場所で使用する表示名を指定しています（たとえば、Windowsの［**表示**］**>**［**アドイン**］）。

```
Register Addin("com.company.lee.dan.MyAddIn","$DOCUMENTS/myaddin",
  displayname( "Calculator Addin" ));
```

第2引数は、$ADDIN_HOMEパス変数を設定します。このアドインスクリプトを参照する際は、パス変数の末尾に必ずスラッシュを含めます。

```
Include("$ADDIN_HOME(com.jmp.jperk.texttocols)/texttocols.jsl");
```

Revert Menu()

説明

JMPメニューを出荷時の状態に戻す。

Run Program(Executable("path/filename.exe"), Options({"/a", "/b", "..."}), Read Function(expression), Write Function(expression), Parameter(expression))

説明

Executable引数で指定された外部プログラムを、Options引数で指定されたコマンドライン引数を使って実行する。

結果

引用符付き文字列、BLOB、またはRun Programオブジェクトのいずれかを、Read Function引数による制御に従って戻す。

引数

Executable 実行ファイルのパス。macOSの場合は、実行ファイルのフルパスをタイプします。

Options 実行ファイルのコマンドライン引数。

Read Function Read Function("text")が指定された場合は引用符付き文字列を戻し、Read Function("blob")が指定された場合はBLOBを戻す。スクリプトは、外部プログラムがそのstdoutを閉じるまで待ちます。その後、Run Programは、外部プログラムが引用符付き文字列またはBLOBとしてstdoutに書き込んだすべてのデータを戻します。

Read Functionが指定されなかった場合は、Run Programオブジェクトを戻します。

Write Function（オプション）関数を値として受け入れる。"text"や"blob"は受け入れられません。

Parameter（オプション）Read Functionで式を読み／書きするための引数。

メモ:

- Run Program()が関数の中に含まれている場合、Run Programの中ではローカル変数ではなく、グローバル変数を使用してください。
- Read Functionが指定されていない場合に戻されるRun Programオブジェクトは、外部プログラムのstdoutからデータを読み取るための以下のメッセージを受け入れます。
  - <<Read: 読み取り可能なデータを引用符付き文字列として読み取る。読み取り可能なデータがない場合は、空の文字列を戻します。
  - <<Can Read: 読み取り可能なデータがある場合は1（真）を戻す。
  - <<Is ReadEOF: 外部プログラムが完了し、すべてのデータを読み取ったら、1（真）を戻す。

    これらのメッセージを使って、外部プログラムによってデータが生成されたときに、それらのデータをポーリングし、処理できます。
- Run Programオブジェクトは、データを外部プログラムのstdinに書き込むための以下のメッセージを受け入れます。
  - <<Write("text"): 外部プログラムのstdinにデータを送る。
  - <<Can Write: 外部プログラムが即時にデータを受け入れる場合は1（真）を戻す。そうでない場合、<<Writeを呼び出すとスクリプトが止まってしまいます。
  - <<WriteEOF: データ送信が終了したことを示す、外部プログラムへの信号。
- 戻されたRun Programオブジェクトにメッセージを送る代わりに、Read Function引数をインライン関数として指定できます。RPはRun Programオブジェクトです。

```
RP = Run Program(
   Executable( ... ),
   Read Function(
      Function( {RP},
          <ここにコードを入力>
         RP << Read
      )
   )
```

```
);
```

Parameter(optParm)引数は、Read Functionにおけるオプションの引数です。これが指定された場合、Read FunctionとWrite Functionに定義された関数は、第2引数（optParmの値）を受け取ることができます。

例

次のスクリプトは、Write Function引数の一例です。RPはRun Programオブジェクトです。この例では、<<Writeと<<WriteEOFのメッセージを受け取っています。

```
RP = Run Program(
   Executable( ... ),
   Write Function(
      Function( {RP},
         <ここにコードを入力>
         RP << Write( "Program finished." )
      )
   )
);
```

次のスクリプトは、Parameter(optParm)引数の一例です。

```
RP = Run Program(
   Executable( ... ),
   Parameter( x ),
   Read Function( Function( {RP, optParm},... ) )
);
```

Read Function内で、optParmは*x*の値を含んでいます。Parameter引数を指定していない場合は、関数内のoptParm引数にアクセスしようとしないでください。

SAS Open For Var Names("path")

説明

変数名の取得だけを目的としてSASデータセットを開き、変数名を引用符付き文字列のリストで戻す。

戻り値

ファイル内の変数名のリスト

引数

path SASデータセットのパス名を示す引用符付き文字列。

Schedule(n, script)

説明

*n*秒後に指定のスクリプト（*script*）を実行するというイベントをキューに入れる。

Set Clipboard(*string*)

説明

指定された引用符付き引数（"*string*"）のテキスト部分の情報をクリップボードに入れる。

例

```
Set Clipboard( "このテキストをコピー" );
```

Set Platform Preference()

Set Platform Preferences()

「Platform Preferences(platform(option(value)), ...)」を参照してください。

Set Preference()

Set Preferences()

「Preferences(pref1(value1), ...)」を参照してください。

Set Toolbar Visibility("toolbar name" | default | all, window type | all, "true" | "false")

説明

Windowsで、ウィンドウタイプに基づいて、またはすべてのウィンドウに対して、ツールバーを表示するか、非表示にする。

引数

`toolbar name | default | all` ツールバーの内部名（JMPの［**表示**］**>**［**ツールバー**］のリストを参照）、指定したウィンドウタイプのデフォルトのツールバー（`default`）、またはすべてのツールバー（`all`）。`"toolbar name"`には引用符が必要です。

`window type | all` データテーブル（Data table）、スクリプト（script）、レポート（report）、ジャーナル（journal）、またはすべてのウィンドウ（all）。

`true | false` ツールバーの表示または非表示を指定する引用符付き文字列。

Shortest Edit Script(A, B)

Shortest Edit Script(strings(A, B, <matrix(0|1)>, <limit(*number*)>))

Shortest Edit Script(lines(A, B, <matrix(0|1)>, <limit(*number*)>, <separators(*characters*)>, <ignore(*characters*)|ignore white space( )>))

Shortest Edit Script(sequences(nA, nB, Function({iA, iB}, adata[iA] == bdata[ib])))

説明

2つの引用符付き文字列、行、またはシーケンスを比較する。

戻り値

編集コマンドのリストまたは行列を戻す。比較対象のみを指定した場合は、リストが戻されます。引用符付きのstringsまたはlinesを指定した場合は、戻り値として行列またはリストのいずれかを指定できます。sequencesは行列を戻します。

利用できるコマンドは3つあります。両方の引用符付き文字列で共通のデータ、最初の文字列から削除するデータ、2つ目の文字列で維持するデータです。

オプションの引数

matrix 戻り値を行列にするかどうかを指定する。

limit 編集リストの挿入または削除項目が指定の数を超えたら、評価を停止する。比較したい文字列がランダムであっても、多くの個所に多くの共通な文字がありえます。そのため、比較する文字列がランダムな文字列である場合、最良の一致を探すために、処理時間がかかる場合があります。limitを指定すると、関数の実行時間が短くなります。

Linesのオプションの引数

matrix 戻り値を行列にするかどうかを指定する。

limit 編集リストの挿入または削除項目が指定の数を超えたら、評価を停止する。比較したい文字列がランダムであっても、多くの個所に多くの共通な文字がありえます。そのため、比較する文字列がランダムな文字列である場合、最良の一致を探すために、処理時間がかかる場合があります。limitを指定すると、関数の実行時間が短くなります。

separators 単語を区切る文字。

ignore 行の中にある指定された文字を無視する。

ignore white space 行の中にあるスペースを無視する。

Sequencesのオプションの引数

Function ユーザ定義関数。

例

次の例は、共通する文字シーケンスを3つ持つ、2つの引用符付き文字列を比較します。

```
Shortest Edit Script( "abcdef", "abdezgh" );
   {{"Common", "ab"}, {"Remove", "c"}, {"Common", "de"}, {"Insert", "zgh"},
      {"Remove", "f"}}
```

次の例は、aaとbbの各行を調べています。

```
aa = "this is
a test of
shortest
edit script
lines with several words";

bb = "this is
a test 2 of
shortest
edit ?., script
```

```
lineswithseveral words";

Shortest Edit Script( lines( aa, bb, separators( "\!n" ),

  // 行の区切り文字を引用符で囲んで指定
  ignore( "?., " ) ) ); // これらの文字とスペースを無視
  {{"Common", "this is // "this is" は aa と bb に共通する行
  "}, {"Remove", "a test of // aa のみ 2 行目にある
  "}, {"Insert", "a test 2 of // bb のみ 2 行目にある
  "},
  {"Common", "shortest
  edit script
  lines with several words"}}
  // aa と bb の両方に含まれる行:"shortest"、"edit script"、"lines with several words"
```

Show Addin Builder Dialog()

説明

カスタムアドインを作成するためのウィンドウを開く。

Show Addins Dialog()

説明

アドインのステータスウィンドウを開く（[**表示**] > [**アドイン**]）。

引数

なし

Show Commands()

説明

スクリプト可能なオブジェクトと演算子をすべて表示する。引数には、All、DisplayBoxes、Scriptables、Scriptable Objects、StatTerms、Translationsを指定できます。

Show Preferences(<"all">)

説明

現在の環境設定を表示する。引数が指定されていない場合、変更された環境設定を表示します。引数に"all"が指定されている場合は、すべての環境設定を表示します。

Show Properties(object)

説明

与えられたオブジェクト（*object*）が解釈できるメッセージと、その基本的な構文情報を表示する。

Sobol Quasi Random Sequence(nDim, nRow)

説明

Sobol数列の準乱数を生成する。最大4000次元まで。この準乱数は、Space Filling（空間充填）の分野で利用されることがある。

Socket(<STREAM | DGRAM>)

説明

ソケットを生成する。

戻り値

生成されたソケット

引数

STREAM | DGRAM（オプション）ソケットの種類がストリームかデータグラムかを指定する。引数が指定されていない場合、ストリームソケットが生成される。

Speak(text, <wait(Boolean)>)

説明

システムの音声出力機能を呼び出し、テキストを読み上げる。Waitがオンになっている場合、次のスクリプトは読み上げが終了してから実行されます。

Status Msg("message")

説明

引用符付き文字列（message）をステータスバーに表示する。

Subtract(a, b)

a-b-...

説明

リストされた引数の値を、左から右へと引く。どの引数も変更されません。

戻り値

差

引数

2つ以上の変数、数値、または行列。

メモ

2つ以上の引数が使えます。

Unregister Addin("unique_id")

説明

登録されたアドインの登録を解除（削除）する。

引数

unique_id 登録を解除するアドインの一意の識別子を示す引用符付き文字列。

Web(*string*, <JMP Window>)

説明

引用符付きの*string*で指定されたURLをデフォルトのWebブラウザで開く。

URL内のhttp://接頭辞は省略することも可能です。

例

```
url = "www.jmp.com"; // URLをデフォルトのWebブラウザで開く
Web( url );

Web( "www.jmp.com" ); // URLをデフォルトのWebブラウザで開く

Web( "www.jmp.com", JMP Window ); // URLをJMPのブラウザウィンドウで開く
```

XML Attr("attr name")

説明

Parse XMLコマンドの評価において、XML引数の引用符付き文字列を抽出する。

XML Decode("xml")

説明

XML内の記号を通常のテキストにデコードする。たとえば、&を&、<を<に変更します。

引数

xml XMLを含んだ引用符付き文字列。

XML Encode("text")

説明

XMLに埋め込むテキストを準備する。たとえば、&を&、<を<に変更します。

引数

xml プレーンテキストを含んだ引用符付き文字列。

XML Text()

説明

Parse XMLコマンドの評価において、XMLタグの本体（ボディ）の引用符付き文字列を抽出する。

第3章

JSL メッセージ
オブジェクトおよびディスプレイボックスのメッセージの概要

ここでは、JMP の一般的なオブジェクトメッセージについて簡単に説明しています。オブジェクトメッセージの詳細については、**[ヘルプ] > [スクリプトの索引]** で表示される「スクリプトの索引」を参照してください。

プラットフォームのメッセージの詳細については、『スクリプトガイド』を参照してください。

目次

アルファシェイプのメッセージ 360
連想配列のメッセージ 360
クラスのメッセージ 361
データコネクタのメッセージ 363
データコネクタメタデータのメッセージ 363
データコネクタレジストリのメッセージ 364
データテーブルのメッセージ 365
データテーブル全体のメッセージ 365
列のメッセージ 394
行のメッセージ 401
データフィルタのメッセージ 402
データフィードのメッセージ (Windows のみ) 406
ディスプレイボックスのメッセージ 408
すべてのディスプレイボックスのメッセージ 408
Axis Box のメッセージ 419
Border Box のメッセージ 424
Data Browser Box のメッセージ 425
Data Filter Source Box のメッセージ 425
Frame Box のメッセージ 425
Graph 3D Box のメッセージ 427
Excerpt Box のメッセージ 427
Filter Col Selector のメッセージ 428
Global Box のメッセージ 428
Hier Box のメッセージ 428
Matrix Box のメッセージ 428
Nom Axis Box のメッセージ 429
Number Col Box のメッセージ 429
Number Col Edit Box のメッセージ 432
Number Edit Box のメッセージ 433
Outline Box のメッセージ 433
Panel Box のメッセージ 434
Plot Col Box のメッセージ 434
Slider Box ／ Range Slider Box のメッセージ 435
String Col Box のメッセージ 436
Tab Box のメッセージ 438
Table Box のメッセージ 438
Text Box のメッセージ 442
Tree Node と Tree Box のメッセージ 443

三角分割のメッセージ 445
Window のメッセージ 447
ダイナミックリンクライブラリ（DLL）のメッセージ 449
イメージのメッセージ 451
インタラクティブ HTML のメッセージ 454
Web レポートのメッセージ 454
JMP アプリケーションのメッセージ 455
JMP App のメッセージ 455
JMP App Module のメッセージ 457
JMP App Module Instance のメッセージ 458
MATLAB のメッセージ 458
名前空間のメッセージ 462
PI サーバーからの読み込みメッセージ 464
PI サーバーのメッセージ 464
Importer メッセージ 465
Python インテグレーションのメッセージ 467
R インテグレーションのメッセージ 472
スケジュールのメッセージ 475
セグメントのメッセージ 476
ソケットのメッセージ 476
SQL のメッセージ 480
その他のオブジェクトのメッセージ 483
Zip アーカイブ 483
ジャーナル 484

アルファシェイプのメッセージ

以下のJSLメッセージで、ashapeはアルファシェイプまたはアルファシェイプへの参照を表します。

---

ashape <<Get Alpha

現在のアルファ値を戻す。

---

ashape <<Set Alpha(*alpha*)

アルファ（*alpha*）値を設定し、アルファシェイプによる三角分割を再計算する。

---

ashape <<Get Tri Alpha

各三角形のアルファ値を戻す。

連想配列のメッセージ

以下のJSLメッセージで、mapは連想配列または連想配列への参照を表します。

---

map<<First

*map*内の最初のキーを戻す。*map*にキーがない場合は、Empty()を戻します。キーは辞書式の順序で戻されます。

---

map<<Get Contents

*map*内におけるすべてのキーと値のペアを、リストで戻す。

---

map<<Get Keys

*map*内のすべてのキーを、リストで戻す。

---

map<<Get Default Value()

存在しないキーで参照した場合の値を設定する。値が設定されていない場合は、Empty()を戻します。

---

map<<Get Value(*key*)

*map*内の*key*に対応する値を戻す。

map<<Get Values(*<{keyList}>*)

引数が指定されない場合は、*map*内のすべての値を、リストで戻す。

キーがリストで引数に指定された場合、それらのキーに対応する値を、リストで戻します。

map<<Insert(*key*, *value*)

map にキー（*key*）を挿入し、それに値（*value*）を割り当てる。*map* にすでに *key* がある場合は、新しい *value* で置き換えられます。このメッセージは、関数 Insert Into と等価です。

map<<Next(*key*)

*map*内の指定のキー（*key*）の次のキーを戻す。*map*にキーがない場合は、Empty() を戻します。キーは辞書式の順序で戻されます。

map<<Remove(*key*)

*map*から、キー（*key*）とその値（*value*）を削除する。このメッセージは、関数Remove From と等価です。

map<<Set Default Value(*v*)

存在しないキーで参照した場合の値を設定する。存在しないキーにはすべて、デフォルトでこの値が割り当てられます。

クラスのメッセージ

obj<<Clone

obj クラスオブジェクトのコピーである新しいクラスオブジェクトへの参照を戻す。

obj<<Contains(*quoted string*)

obj クラスオブジェクトに指定の引用符付き文字列が含まれている場合は 1、そうでない場合は 0 を戻す。

obj<<Delete Class

obj クラスオブジェクトを削除する。

obj<<Equal(*classref*)

classref クラスオブジェクトが *obj* クラスオブジェクトと等しい場合は 1、そうでない場合は 0 を戻す。

obj<<First

obj クラスオブジェクトの最初のメンバー（項目）の名前を示す引用符付き文字列を戻す。クラスオブジェクトのメンバー（項目）はアルファベット順に並んでいます。

obj<<Get Contents

obj クラスオブジェクトのメンバー（項目）のリストを戻す。リスト内の各要素は、キーと関連する値から成る 2 項目のリストです。

obj<<Get Keys

obj クラス内のキーのリストを戻す。各キーは、*obj* クラスオブジェクト内のメンバー（項目）の名前を示す引用符付き文字列です。

obj<<Get Name

obj クラスオブジェクトの名前を示す引用符付き文字列を戻す。

obj<<Get Value(*key quoted string*)

obj クラスオブジェクト内の指定のメンバー（項目）の値を戻す。引用符付きの *key quoted string* 引数は、メンバー（項目）へのキーを指定します。

obj<<Get Values

obj クラスオブジェクトのメンバー（項目）の値のリストを戻す。リスト内の各要素は、クラス内の各メンバー（項目）の値を示す式です。

obj<<Insert(*quoted string*, *value*)

obj クラスオブジェクトにメンバー（項目）を挿入する。引用符付きの *string* 引数は、メンバー（項目）の名前。*value* 引数は、メンバー（項目）の式の値。

obj<<Lock Class(<*quoted string*|*{quoted stringList}*>)

obj クラスオブジェクトをロックするか、*obj* クラスオブジェクト内の特定のメンバー（項目）をロックする。クラスオブジェクトがロックされると、メンバー（項目）の追加や変更、削除ができなくなります。引用符付きの *string* または引用符付きの *stringlist* 引数は、ロックするメンバー（項目）を指定します。また、引用符付き文字列のリストを指定して複数のメンバー（項目）をロックすることもできます。

obj<<N Items

obj クラスオブジェクト内のメンバー（項目）の数を戻す。

obj<<Next(*quoted string*)

引用符付きの *string* で指定された *obj* 内のメンバー（項目）に続くメンバー（項目）の名前を、引用符付きの *string* で戻す。クラスオブジェクトのメンバー（項目）はアルファベット順に並んでいます。

obj<<Remove(<*string*|*{stringList}*>)

引用符付きの *string* または引用符付きの *stringlist* で指定されたメンバー（項目）を、*obj* クラスから削除する。引用符付き文字列のリストを使えば、複数のメンバー（項目）を削除することができます。

obj<<Show Contents

obj クラスオブジェクトの内容をログウィンドウに表示する。

obj<<Unlock Class(<*quoted string*|*{stringList}*>)

obj クラスオブジェクトのロックを解除するか、*obj* クラスオブジェクト内の特定のメンバー（項目）のロックを解除する。クラスオブジェクトのロックを解除すると、メンバー（項目）の追加や変更、削除ができるようになります。引用符付きの *string* または引用符付きの *stringlist* 引数は、ロックを解除するメンバー（項目）を指定します。また、引用符付き文字列のリストを指定して複数のメンバー（項目）のロックを解除することもできます。

データコネクタのメッセージ

この節では、データコネクタ用のJSLメッセージをご紹介します。

データコネクタメタデータのメッセージ

metadata<<Get Description()

データコネクタの説明を取得する。

metadata<<Get Driver()

データコネクタのドライバがある場合にそれを取得する。

metadata<<Get Name()

データコネクタ名を取得する。

metadata<<Get Path()

データコネクタのパスを取得する。

metadata<<Get Type()

データコネクタの種類を取得する。

metadata<<Set Description(*description*)

データコネクタの説明を設定する。

metadata<<Set Name(*name*)

データコネクタ名を設定する。

データコネクタレジストリのメッセージ

dc<<Get(*name*)

レジストリからデータコネクタを読み込む。

dc<<Get Available()

レジストリから使用可能なデータコネクタのリストを読み込む。

dc<<Get Metadata(*name*)

レジストリからデータコネクタのメタデータを取得する。

dc<<Register(Path(*path*), <Name(*name*)>, <Description(*description*)>)

説明

レジストリにデータコネクタを追加する。

必須の引数

path 登録するデータコネクタのパス。

name データコネクタの名前。

オプションの引数

description データコネクタの説明。

例

```
Names Default To Here( 1 );

Data Connector Registry() << Register(
   Path( "$DOCUMENTS/my connector.jmpdc" ),
   Name( "My Data Connector" )
);
```

dc<<Unregister(*name*)

レジストリからデータコネクタを削除する。

データテーブルのメッセージ

この節では、JMP データテーブル用の JSL メッセージをご紹介します。

データテーブル全体のメッセージ

dt<<Add Column Properties(*property argument*, ...)

選択されている列に指定のプロパティ（「値の表示順序」や「欠測値のコード」など）を追加する。

dt<<Add Multiple Columns(*column prefix*, *n*, <"Before First"|"After Last"|After(*column*)>, "Character"|"Numeric"|"Row State", <Field Width(*n*)>)

説明

データテーブル（*dt*）内の指定された位置に *n* 個の列を追加する。

必須の引数

column prefix 新しい列の列名に追加する接頭辞。

n 追加する列の数。

Character 新しい列を文字タイプとする。

Numeric 新しい列を数値タイプとする。

Row State 新しい列を「行の属性」列とする。

オプションの引数

Before First 列を最初の列の前に追加する。

After Last 列を最後の列の後に追加する。

After(*column*) 列を指定の列の後に追加する。

Field Width(*n*) 列の桁数を指定する。

メモ

引数を省略した場合、または引数を正しく指定しなかった場合、「複数の列を追加」ウィンドウが表示されます。

dt<<Add Rows(*n*, <"At Start"|"At End"|After(*row number*)>|
{*column name=value pairs*})

説明

データテーブルの先頭、末尾、または指定した行の後ろに行を追加する。このメッセージでは、列名と値のペアを指定して、追加した行に値を設定することもできます。その場合、新しい行は、データテーブルの末尾に追加されます。

メモ

引数を省略した場合、または引数を正しく指定しなかった場合、「行の追加」ウィンドウが表示されます。

dt<<Add Scripts to Table(*script*, ...)

dt<<Add Properties to Table(*script*, ...)

スクリプト（script）をデータテーブルに追加する。

dt<<Anonymize(<Columns*(column list(s))*>, <Output Table Name(*name*)>);

データ、一部の列プロパティ、およびテーブルスクリプトから識別可能な値を削除する。データテーブルまたは指定の列のリストに適用されます。新しいデータテーブルには、引用符付きの *name* 引数で指定された名前が付きます。

dt<<Begin Data Update

データテーブルのセルをすばやく更新できるように、表示の更新をオフにする。表示の更新をオンに戻すには、End Data Update を使います。

メモ

Begin Data Update は、データ値を変更する以外のテーブル操作によるデータの再描画には影響しません。たとえば、列を削除または追加する場合は、データテーブルが更新された後、データ値の更新が開始されます。

dt<<Clear Column Selection

選択されているすべての列の選択を解除する。

dt<<Clear Edit Lock(<"Modify Cells">, <"Add Rows">, <"Add Columns">,
<"Delete Rows">, <"Delete Columns">)

説明

データテーブルに対して、指定された操作を再び行えるようにする。

メモ

引数を指定しなかった場合、すべてのロックが解除されます。

dt<<Clear Row States

有効な行の属性をすべてクリアする。

dt<<Clear Select

現在の選択をクリアする。

dt<<Clone Formula Column(*column*, *n*, Substitute Column Reference(*column1*, *{list}*))

*n*個の新しい計算式列を作成し、*column1*への参照を *list*の列に置き換えて元の *column*の計算式に挿入する。

dt<<Close Data Grid(*Boolean*)

真の場合にデータテーブルグリッドを閉じる。

dt<<Close Side Panels(*Boolean*)

真の場合にデータテーブルのサイドパネルを閉じる。

dt<<Color or Mark by Column(*column*, *<named arguments>*)

dt<<Color by Column(*column*, *<named arguments>*)

dt<<Marker By Column(**column**, *<named arguments>*);

説明

データテーブルの列の値に従って色またはマーカーを割り当てる。オプションの引数が指定されていない場合は、デフォルトのカラーテーマに従って色が割り当てられます。

必須の引数

column 色またはマーカーを付ける列。

オプションの名前付き引数

Color(*n*) 指定のJMPカラーを使用する。

Add Marker(*Boolean*) データテーブル内のマーカーの表示/非表示を切り替える。

Color Theme(*color theme*) 引用符付きで指定されたカラーテーマを使用する。

Marker Theme(*marker theme*) 引用符付きで指定された*マーカーテーマ*を使用する。指定できるテーマは、"Standard"（標準）、"Hollow"（中抜き）、"Paired"（ペア）、"Classic"（クラシック）、および、"Alphanumeric"（英数字）。

Continuous Scale|Continuous Scale(*Boolean*) 指定の列の値に従って、グラデーションになった色を割り当てる。

Reverse Scale|Reverse 使用中の色を反転する。

Excluded Rows(*Boolean*) 真の場合は、除外されている列に行の属性を適用する。

"Make Window with Legend" 凡例のウィンドウを別に作成する。

dt<<Color Rows by Row State

行の属性による色の割り当てに基づき、データテーブルグリッド内の行に色をつける。行の色を無効にするには、再度このメッセージを送ります。

dt<<Combine Columns(Delimiter("*delim*"), Columns(*column1*, *column2*, etc.), Column Name(*quoted string*))

複数の列の値を 1 つに結合する。元の列の値は、引用符付きの *delim* 引数で指定された区切り文字で区切られます。

例

```
dt = Open( "$SAMPLE_DATA/Consumer Preferences.jmp" );
dt << Combine Columns(
   Delimiter( "," ),
   Columns(
      : 起床後に歯磨き,
      : 食後に歯磨き,
      : 就寝前に歯磨き,
      : 別のタイミングで歯磨き
   ),
   Column Name( "いつ歯磨きをするか" )
);
```

dt<<Compress File When Saved(*Boolean*)

データテーブルの保存時にファイルを圧縮する。

dt<<Compress Selected Columns(*{column1, ...}*)

リストされた列を可能な限り小さく圧縮する。文字データの列の場合、水準が 255 個未満であれば 1 バイトに圧縮できます。数字データの列の場合、数値が -127 ～ 127 の整数であれば 1 バイトに圧縮できます。

dt<<Concatenate(*dt2*|Data Table(*name*)|Multiple Data Table(*name*) arguments, (<"Private"|"Invisible">), <Output Table Name(*name*)>|"Append to First Table">, <"Keep Formulas">, <"Create Source Column">)

説明

データテーブル（*dt*）とデータテーブル 2（*dt2*）を連結し、新しいデータテーブル（*name*）を作成する。デフォルトでは、Concatenate は新しいデータテーブルを作成し、指定された各データテーブルの行を追加します。

戻り値

連結されたデータテーブルへの参照

必須の引数

dt2|Data Table(*name*)|Multiple Data Table(*name*) 結合したいデータテーブルのデータテーブル参照または名前。

オプションの引数

"Private" データテーブルウィンドウには表示せずにデータテーブルを開く引用符付きキーワード。

"Invisible" 出力のデータテーブルを非表示にする引用符付きキーワード。データテーブルを非表示で出力して、後に続くスクリプトで利用する場合、この引数を使用してください。このデータテーブルは、ホームウィンドウの「ウィンドウリスト」または［**ウィンドウ**］>［**再表示**］のリストに表示されます。

Output Table name(*name*) 最終的なデータテーブルの名前。名前を入力しない場合、データテーブルには「無題#」（たとえば、「無題1」）という名前が付きます。

"Append to First Table" 最初の引数にある最初のデータテーブル参照またはデータテーブル名に行を追加する。このオプションは、新しいデータテーブルを作成する代わりとなるものです。

"Keep Formulas" 最終的なデータテーブルに計算式を含める。

"Create Source Column" 新しいデータテーブルに元のテーブルという列を追加する。

メモ

"Private"と"Invisible"は、"Append to First Table"を使用して**いない**場合のみ適用されます。

dt<<Copy Column Properties

選択されているすべての列の列プロパティをコピーする。データテーブルで列を選択しておく代わりに、リストで列を指定することもできます。

例

```
dt = Open( "$SAMPLE_DATA/Tiretread.jmp" );
dt << Select Columns( :引張応力, :伸び);
dt << Copy Column Properties;
New Window( "Script", Script Box( "//ここに貼り付けてください。" ) );
```

または

```
dt = Open( "$SAMPLE_DATA/Tiretread.jmp" );
dt << Copy Column Properties( {:引張応力, :伸び} );
New Window( "Script", Script Box( "//ここに貼り付けてください。" ) );
```

dt<<Copy Selected Properties

説明

選択されているテーブルプロパティをクリップボードにコピーする。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
```

```
dt << Select Properties( {"一変量の分布", "二変量の関係(一元配置)"} );
proplist = dt << Copy Selected Properties();
New Window( "Script", Script Box( "//ここに貼り付けてください。" ) );
```

---

dt<<Copy Table Script("No Data")

データテーブルを再作成するためのスクリプトを、クリップボードにコピーする。"No Data"という引数を指定すると、スクリプトからデータ部分が省かれます。

---

dt<<Copy Table Scripts

dt<<Copy Selected Properties

説明

選択されているスクリプトをクリップボードにコピーする。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Select Properties( {"一変量の分布", "二変量の関係(一元配置)"} );
proplist = dt << Copy Table Scripts();
New Window( "Script", Script Box( "//ここに貼り付けてください。" ) );
```

---

dt<<Data Filter(<Location(*x*, *y*), <"Close Outline">, <"Local">, <Inverse(*Boolean*)>, <Show Columns Selector(*Boolean*)>, <Title(*quoted string*)>, <Save and Restore Current Row States(*Boolean*)>, <Conditional(*Boolean*)>, <Auto Clear(*Boolean*)>, <Group By AND(*Boolean*)>, <Show Histograms and Bars(*Boolean*)>, <Count Excluded Rows(*Boolean*)>, <Mode(...)>, <Add Filter((*cols*(...), <Where(...)>, <Display(...)>, <Select Missing(*cols*)>, <Order By Count(*cols*)>)>, <Favorites(...)>, <Animation(...)>)

データフィルタを作成する。引数が指定されていない場合は、「フィルタ列の追加」ウィンドウが表示されます。

オプションの引数

Location(*x*, *y*) データフィルタウィンドウを指定の場所に移動する。*x*と*y*はピクセル単位で指定します。0,0はディスプレイの左上隅を意味します。

"Close Outline" データフィルタアウトラインを閉じる。

"Local" レポートにフィルタを組み込み、他のレポートに影響を及ぼさずに1つまたは複数のプラットフォームをフィルタできるようにする。

Inverse(*Boolean*) すべてのフィルタに対し、指定の行を除くすべての行を選択する。

Show Columns Selector(*Boolean*) 真の場合は、フィルタに新しい列を追加するための列名リストを表示する。

Title(*quoted string*) アウトラインに表示するタイトル。

Save and Restore Current Row States(*Boolean*)「データフィルタ」ウィンドウを閉じるときに現在の行の属性を復元する。

Conditional(*Boolean*) 選択されているフィルタ列に対し、表示されるカテゴリを制限します。

Auto Clear(*Boolean*)「データフィルタ」で複数の名義尺度または順序尺度の列が選択されている場合、フィルタ基準を1つ選択する前に、前回選択した基準を選択解除する。

Group By AND(*Boolean*) デフォルトでは、フィルタグループを作成した場合、ORによって、1つまたは複数のフィルタが追加され、2つ目のフィルタグループが作成されます。Grouped By ANDを指定した場合は、動作が逆になり、ANDでグループ化されます。

Show Histograms and Bars(*Boolean*) データフィルタのヒストグラムと棒の表示／非表示を切り替える。

Count Excluded Rows(*Boolean*) 除外された行の数の表示/非表示を切り替える。

Mode フィルタには、次の3つのモードがあります。Select(*Boolean*)は、データテーブル内で選択された行を強調表示（または非表示）します。Show(*Boolean*)は、選択されていない行を表示または含め、［非表示］アイコンを表示します。Include(*Boolean*)は、選択されていない行を表示または含め、［除外］アイコンを表示します。

グローバルデータフィルタのデフォルトは、Select()、Show(0)、Include(0)です。ローカルデータフィルタのデフォルトは、Show(1)、Include(1)で、Select()は有効なオプションではありません。

Add Filter データフィルタを作成する。引数には、Columns()、Where()、Display()、Select Missing(*cols*)、Order By Count(*cols*)があります。Columns()は、カンマで区切った1つまたは複数の列名をとります。フィルタを定義する場合は、1つまたは複数のWhere節を追加します。

Where データのフィルタリングに使うWhere節を定義する。

Display(*column, size, display type*) 指定のカテゴリカル列の水準を、どのようにフィルタに表示するかを設定する。引数は、Blocks Display、List Display、Single Category Display、Check Box Display、Radio Box Display。カテゴリカルな列では、Find(Set Text(*quoted string*))引数を使って検索フィールドを含め、初期化することができます。連続尺度の列では、Displayを使い、size引数を含めることもできます。

Select Missing Cols(*cols*) 連続尺度の列で欠測値を選択する。

Order by Count(*cols*) このオプションは、カテゴリカルな列に対し、度数の降順に従って値を並べる。

Favorites 現在のデータフィルタの条件をお気に入りとして保存する。

Animation 指定されたアニメーションの向きで、行の選択と選択解除を交互に繰り返す。オプションの引数には、Animate Column(*col*)、Animate Rate(*number*)、"Forward"|"Backward"|"Bounce"があり、"Forward"|"Backward"|"Bounce"はアニメーションの向きを、最初から最後へと前進させます。"Backward"は、アニメーションの向きを、最後から最初へと後進させます。"Bounce"は、アニメーションの向きを、前進させた後、後進させます。

dt<<Data View(*<named arguments>*)

説明

新しいウィンドウにデータテーブルを複製する。次の引用符付き引数のいずれか 1 つを指定した場合は、該当する行だけが新しいデータテーブルに含まれます。

戻り値

データビューへの参照

オプションの名前付き引数

Excluded 新しいデータテーブルに、元のデータテーブルで除外されている行だけが含まれる。

Labeled|Labelled 新しいデータテーブルに、元のデータテーブルでラベルありとマークされている行だけが含まれる。

Hidden 新しいデータテーブルに、元のデータテーブルで表示しないとマークされている行だけが含まれる。

Selected 新しいデータテーブルに、元のデータテーブルで選択されている行だけが含まれる。

dt<<Delete Columns(*column1*, *column2*, ...)

dt<<Delete Column

説明

データテーブル *dt* から 1 つまたは複数の列を削除する。col には、削除する列を指定します。引数がない場合、選択されている列があれば、それを削除します。

dt<<Delete Rows(*<n>*)

dt<<Delete Rows(*{n, o, p, ...}*)

dt<<Delete Rows(*{n::q}*)

dt<<Delete Rows([n, o, p])

dt<<Delete Row(*preceding arguments*)

説明

現在選択されている行、または指定された行を削除する。削除された行の行数を戻します。

dt<<Delete Scripts(*table script name*|*{table script1, table script2, ...}*)

説明

引用符付きの名前で指定されたデータテーブルスクリプト、もしくはリストで指定された複数のデータテーブルスクリプトを削除する。

メモ

14より以前のバージョンのJMPでテーブルスクリプトを削除するには、Delete Table Propertyを使用してください。

dt<<Delete Table Property(*name*|*{property1, property2, ...}*)

引用符付きの *name* で指定されたテーブルプロパティ（スクリプトや変数など）を削除する。

dt<<Delete Table Variable(*name*)

引用符付きの *name* で指定されたテーブル変数を削除する。

dt<<Disable Undo(*Boolean*)

真の場合にデータテーブル内で「元に戻す」または「取り消し」操作を無効にする。

dt<<End Data Update

Begin Data Updateメッセージの後で、表示の更新を再開する。これらのコマンドは、変更箇所が多数ある場合に、データテーブルをすばやく更新するために使われます。表示の更新をオフにすると更新速度が上がります。

dt<<Exclude

dt<<Unexclude

データテーブル（*dt*）内で選択された行を「除外する」から「除外しない」、または**その逆**に切り替える。

dt<<Get All Columns As Matrix

データテーブル（*dt*）のすべての列の値を行列で戻す。文字型の列には、水準に従って1から順に番号が付けられます。

dt<<Get As Matrix(*<{list of columns by name}>*|*<{list of columns by number}>*, *<column range>*)

データテーブル（*dt*）の数値列の値を行列で戻す。デフォルトでは、すべての数値列が出力されます。

例

```
dt1 = Open( "$SAMPLE_DATA/Big Class.jmp" );
cols = dt1 << Get As Matrix(); // すべての数値列を戻す
Show( cols );
   cols =
   [ 12 59 95,
      12 61 123,
```

```
      12 55 74,...]
colnums = dt1 << Get As Matrix( {4, 5} ); // 4列目と5列目を戻す
Show( colnums );
   colnums = [ 59 95, 61 123, 55 74, 66 145, 52 64, 60 84, 61 128, ...]

dt2 = Open( "$SAMPLE_DATA/Probe.jmp" );
colrange = dt2 << Get As Matrix( 10::22 ); // 10~22列目を戻す
Show( colrange );
   colrange =
   [ -0.08818069845438 0.711340010166168 1.85904002189636 0.396923005580902
      4.50656986236572 7.86504983901978 1.53891003131866 -2.76178002357483
      0.0711032971739769 5.75577020645142 -3.62023997306824 -0.971698999404907
      -0.0525696985423565, ...]
```

dt<<Get As Report

データテーブルをレポートにして戻す。データテーブルで行および列が選択されている場合は、選択されている行および列のみがレポートに含まれます。

例

次のスクリプトは、「Big Class.jmp」をレポートにして、分布とともに1つのウィンドウ内に表示します。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dtRpt = dt << Get As Report;
distRpt = V List Box(
   dt << Distribution(
      Continuous Distribution( Column( :"体重(ポンド)"n ) ),
      Nominal Distribution( Column( :年齢 ) )
   )
);
New Window( "例", H List Box( dtRpt, distRpt ) );
```

dt<<Get Cell Height

データテーブルのセルの高さ（ピクセル）を戻す。

dt<<Get Column Names(*<data type>*, *<modeling type>*, *<string>*)

説明

指定したタイプの列の名前をリストで戻す。複数のタイプを指定できます。引数がない場合、すべての列名が戻されます。

オプションの引数

data type 指定可能な値: Numeric, Character, and RowState.

modeling type 指定可能な値: Continuous, Ordinal, and Nominal.

string 引用符付き文字列のリストを戻す。

dt<<Get Column Reference({*list of column names*}|[*matrix of column numbers*])

dt<<Get Column References({*list of column names*}|[*matrix of column numbers*])

リスト内で指定された引用符付き列名または行列内で指定された番号の、列に対する参照を戻す。リストまたは行列を使用しない場合、すべての列への参照のリストが戻されます。

dt<<Get Display Width

列の表示幅（単位はピクセル）を戻す。

dt<<Get Edit Lock

データテーブルで許可されていない操作（セルの編集、行の追加・削除、列の追加・削除など）のリストを戻す。

dt<<Get Excluded Columns

現在除外されている列を戻す。

dt<<Get Excluded Rows

データテーブルで現在除外されている行を戻す。Where()を使用すると、より柔軟に指定ができます。「JSLの関数、演算子、およびメッセージ」の章の「Where(<dt>, clause)」を参照してください。

dt<<Get Header Height

列見出しの高さを戻す（単位はピクセル）。

dt<<Get Hidden Columns

データテーブルで現在非表示の列を戻す。

dt<<Get Hidden Rows

データテーブルの現在非表示の行を戻す。Where()を使用すると、より高速に処理できます。また柔軟な指定も可能です。「JSLの関数、演算子、およびメッセージ」の章の「Where(<dt>, clause)」を参照してください。

dt<<Get Journal

ディスプレイボックスのジャーナルを生成するソースを引用符付き文字列として戻す。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
```

```
biv = dt << Bivariate( Y( :"体重(ポンド)"n ), X( :"身長(インチ)"n ) );
rbiv = biv << Report;
Print( rbiv << Get Journal );
```

dt<<Get Label Columns

dt<<Get Labeled Columns

dt<<Get Labelled Columns

データテーブルで現在ラベルのついている列を戻す。

例

「PopAgeGroup.jmp」では、「国」列と「年」列にラベルがついています。次のスクリプトは、ラベルのついた列の名前のリストを戻します。

```
dt = Open( "$SAMPLE_DATA/PopAgeGroup.jmp" );
dt << Get Labeled Columns;
   {:Country, :Year}
```

dt<<Get Labeled Rows

dt<<Get Labelled Rows

データテーブルの現在ラベルの付いている行を戻す。Where() を使用すると、より高速に処理できます。また柔軟な指定も可能です。「JSL の関数、演算子、およびメッセージ」の章の「Where(<dt>, clause)」を参照してください。

dt<<Get Name

データテーブルの名前を戻す。

オプションの引数

Ignore Extension データテーブルの名前を拡張子なしで戻す。

dt<<Get Path

JMP のデータテーブルの絶対パスを戻す。読み込んだ後にまだ保存していないデータの場合は、絶対パスは戻されません。

dt<<Get Property(*name*)

引用符付きプロパティ *name* のスクリプトを戻す。

dt<<Get Row Change Function

行が選択されたときに評価する式を戻す。

dt<<Get Row ID Width

行番号の領域の表示幅（単位はピクセル）を戻す。

dt<<Get Row States

データテーブルまたはデータフィルタの各行の行属性を含むベクトルを戻す。

dt<<Get Rows Where(*where clause*)

指定のWhere条件とマッチするデータテーブルの行を戻す。以下はその例です。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Get Rows Where( :性別 == "M" );
dt << Get Rows Where( :性別 == "M" & :年齢 < 15 );
```

dt<<Get Script(*<script name>*)

引用符付きの *script name* で指定されたスクリプトを戻す。*script name* が省略された場合、Get Script は、データテーブルを表すテキストを、データテーブルに保存されているすべてのスクリプトとともに戻します。

dt<<Get Script Group(*<group name>*)

説明

引用符付きの *group name* に含まれているテーブルスクリプトのスクリプト名を、リストで戻す。グループ名が指定されていない場合は、すべてのグループのすべてのテーブルスクリプトのリストを戻します。

dt<<Get Script Group Names

説明

複数のテーブルスクリプトグループのグループ名を、リストで戻す。

dt<<Get Scroll Locked Columns

スクロールロックされている列のリストを戻す。

dt<<Get Selected Columns(*<quoted string>*)

説明

選択されている列のリストを列参照として戻す。選択されている列の名前を文字列のリストとして戻すには、引用符付きの *string* 引数を指定してください。

dt<<Get Selected Properties(*<{list of properties}>*)

説明

選択されているテーブルプロパティをリストで戻す。

オプションの引数

list of properties 取得するプロパティを指定する。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Select Properties( {2, 4} );
proplist = dt << Get Selected Properties();
// 2番目と4番目のテーブルスクリプトを選択し、
// それらのスクリプトを戻す
```

dt<<Get Selected Rows()

選択されている行を戻す。

dt<<Get Table Script Names()

データテーブルのすべてのスクリプト名およびプロパティ名のリストを戻す。

dt<<Get Table Variable(*name*)

引用符付きの *name* で指定された変数の値を戻す。

dt<<Get Table Variable Names

すべてのテーブル変数の変数名を、リストで戻す。

dt<<Go To Row(*n*)

データテーブル（*dt*）の番号 *n* の行を選択する。

dt<<Group Columns(*{column1, column2, ...}*)

dt<<Group Columns(*group name*, *column*, *n*)

dt<<Group Columns(*first column*, *n*)

説明

列を、引用符付きの *group name* で指定されたグループ名でグループ化する。グループ化する列をリストの形で指定するか、最初の列の名前と列数という形で指定します。後者の場合、数字 *n* は、その列とそれに続く *n*-1 個の列をグループ化することを意味します。

dt<<Group Scripts(*{script1, script2, ...}*)

説明

引数のリストで指定されたテーブルスクリプトをグループ化する。

dt<<Hide

dt<<Unhide

データテーブル（*dt*）内で選択された行を「表示しない」から「表示する」、または**その逆**に切り替える。

dt<<Hide and Exclude

選択されている行を非表示かつ除外にする。

dt<<Invert Column Selection(*<{list of columns}>*)

説明

現在選択されていない列を選択し、選択されている列の選択を解除する。*list of columns* が指定されている場合、リストにない列が選択されます。

dt<<Invert Row Selection

現在選択されていない行を選択し、選択されている行の選択を解除する。

dt<<Is Dirty

テーブルが保存された状態から変更されている場合は 1、それ以外は 0 を戻す。

dt<<Join(With(*Data Table(name)*), (<"Private">|<"Invisible">), Select(*columns*), Select With(*columns*), (By Matching Columns(*column1=column2, ...*)|"Cartesian"|"By Row Number"), <"Merge Same Name Columns">, <"Match Flag">, <Copy Formula(*Boolean*)>, <Suppress Formula Evaluation(*Boolean*)>, <"Update">, <Drop Multiples(*Boolean*, *Boolean*)>, <Include Non Matches(*Boolean*, *Boolean*)>, <"Preserve Main Table Order">, <Output Table Name(*name*))>

説明

データテーブル（*dt* および *Data Table*）を横に並べて結合する。

戻り値

データテーブル

必須の引数

With(*Data Table(name)*) アクティブなデータテーブルと結合するデータテーブルを指定する。

"Private" データテーブルウィンドウには表示せずにデータテーブルを開く引用符付きキーワードを指定する。

"Invisible" 出力のデータテーブルを非表示にする引用符付きキーワードを指定する。データテーブルを非表示で出力して、後に続くスクリプトで利用する場合、この引数を使用してください。このデータテーブルは、ホームウィンドウの「ウィンドウリスト」または［**ウィンドウ**］>［**再表示**］のリストに表示されます。

Select(*columns*) アクティブなデータテーブルと結合するデータテーブルを選択する。

Select With(*columns*)

By Matching Columns(*column1=column2*) 両方のテーブルで値とデータタイプがマッチする列を選択する。

"Cartesian" 直積で結合すると、2つの元のデータテーブルの行の可能な組合せがすべて含まれた新しいデータテーブルが作成されます。最初のテーブルのデータを2番目のテーブルのデータと組み合わせて、値のすべての組合せがセットで表示されます。

"By Row Number" 2つのテーブルを横に結合します。

"Merge Same Name Columns" 2つ目のテーブルの列データで、元のテーブルにある同じ名前の列のデータが置き換えられます。最初のテーブルの欠測値は、2番目のテーブルの非欠測値で置き換えられます。

"Match Flag" 対応する列の値で結合する場合に、「対応フラグ」列を作成するかどうかを指定します。

Copy Formula(*Boolean*) 主テーブルまたは結合するテーブルの計算式を出力列に含めます。

Suppress Formula Evaluation(*Boolean*) 新しいテーブルの作成中に列の計算式が自動に評価されないようにします。

"Update" 2つ目のテーブルの列データで、元のテーブルにある同じ名前の列のデータが変更されます。更新された結果は、出力先の新しいデータテーブルに表示されます。次の点に注意してください。データは、欠測値では置き換えられません。また、出力テーブルは、元のテーブルと同じ列を持ちます。そのため、"Update"を使用するときは、列を選択してください。"Update"オプションは、行番号で結合する場合、または対応する列の値で結合する場合のみ使用できます。

Drop Multiples(*Boolean*, *Boolean*) 新しいデータテーブルに名前ごとに1行だけ含まれるよう指定する。対応する列の値で結合する場合のみ適用されます。

Include Non Matches(*Boolean*, *Boolean*) 主データテーブルと新しいデータテーブルで対応しない列を含める。対応する列の値で結合する場合のみ適用されます。

"Preserve Main Table Order" 対応する列を基準に並べ替えるのではなく、元のデータテーブルの順序を結合後のテーブルでも維持します。

Output Table Name(*name*) 結合後のテーブルの名前を指定します。名前を指定しない場合、データテーブルには「**無題#**」（たとえば、「無題1」）という名前が付きます。

dt<<Journal

データテーブルからジャーナルを作成する。データグリッドのみが含まれ、ノート、変数、またはスクリプトは含まれません。

メモ

JMP 14以降で作成されたジャーナルは大きな行列を圧縮した行列データを含んでいる場合があります。ジャーナルを開いてそこからデータを抽出するJSLスクリプトを使用するときは、Journalメッセージでジャーナルをディスクに保存するのではなく、（行列を圧縮しない）Get Journalメッセージの使用を検討してください。

次も参照

「dt<<Get Journal」を参照してください。

---

dt<<Journal Link(<Save(*path*)|Embed()>, <Button Name(*name*)>))

現在のジャーナルに、データテーブルへのリンクを追加する。ジャーナルが存在しない場合は、新しいジャーナルが作成されます。

オプションの引数

path テーブルの保存場所を引用符付きの*path*で指定する。データテーブルがすでにディスク上に保存されている場合は、この指定を省くことができます。保存されていない状態で指定を省くと、ジャーナルリンクが不完全になり、テーブルの再ロードができなくなります。

Embed データテーブルを再作成するJSLスクリプトを組み込む。

Button Name(name) ボタンに表示される名前。*name*引数は引用符付きです。ボタン名を指定しなかった場合、データテーブルの名前が使用されます。

---

dt<<Label

dt<<Unlabel

データテーブル（*dt*）内の選択された行を「ラベルあり」から「ラベルなし」、または**その逆**に切り替える。

---

dt<<Last Modified

データテーブルが最後に保存された日付を戻す。

---

dt<<Layout

Layoutは将来のリリースで削除される予定です。代わりにJournalを使用してください。

---

dt<<Lock Data Table

説明

データテーブルをロックして、データや列プロパティが追加または変更されないようにする。

次も参照

「dt <<Set Edit Lock(<"Modify Cells">, <"Add Rows">, <"Add Columns">, <"Delete Rows">, <"Delete Columns">)」

dt<<Make Indicator Columns(<Append Column Name(*Boolean*)>, <Include Missing(*Boolean*)>)

指定したカテゴリカル列について、0または1の値をとる指示変数の列を作成する。

例

次の例では、「性別」列の指示変数列を作成します。Append Column Nameを指定すると、作成される列名は「性別_F」と「性別_M」になります。指定しない場合は、水準値が列名となります（「F」と「M」）。Include Missingは、欠測値を含めます。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Make Indicator Columns(
   Columns( :性別 ),
   Append Column Name( 1 ),
   Include Missing( 1 )
);
```

dt<<Make RowState Handler

行の属性ハンドラの関数を作成する。関数の引数には、行の属性が変更された行をとる。

dt<<Make SAS Data Step

データテーブルと同じデータを作成するためのSASデータステップを生成する。

dt<<Make SAS Data Step Window

データテーブルと同じデータを作成するためのSASデータステップを生成し、それをSASスクリプトウィンドウに表示する。

dt<<Marker by Column(*column*)

説明

指定されたデータテーブルの列 *column* の値に従ってマーカーを割り当てる。

次も参照

「dt<<Marker By Column(column, <named arguments>);」

dt<<Markers(*n*)

選択されている行にマーカー番号 *n* のマーカーを割り当てる。

dt<<Maximize Display

このメッセージは廃止されます。代わりにOptimize Displayを使用してください。

すべての列のサイズを再調整して、データテーブルのウィンドウを最適なサイズに拡大する。

dt<<Move Script Group(*group name*, "To First"|"To Last"|After(table *script name*)|After(*group*))

説明

引用符付きの *group name* で指定されたテーブルスクリプトグループを並べ替える。

dt<<Move Selected Column(*name(s)*, "To First"|"To Last"|After(*name*))

dt<<Move Selected Columns(*name(s)*, "To First"|"To Last"|After(*name*))

説明

データテーブル内で選択されている列を、指定した位置に移動する。*name* 引数は引用符付きです。

例

次の例は、「Big Class.jmp」データテーブルの「年齢」列を全列の最後へと移動させます。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Go To( :年齢 );
dt << Move Selected Columns( "To Last" );
```

リストを使って列名を指定することもできます。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
list = {"名前", "性別"};
dt << Move Selected Columns( list, "To Last" );
```

dt<<Move Rows("At Start"|"At End"|After(*n*))

データテーブル内で選択されている行を、指定した位置に移動する。*n* は行番号を表します。

dt<<New Column(*name*, *<data type>*, *<modeling type>*, <Format(*format*, *width*)>, <Formula()>, <Set Values(*{..., ..., }*)>, <Set Property(*properties*)>)

説明

データテーブル（*dt*）の最後に、引用符付きで指定された名前（"name"）の新しい列を追加する。特に指定をしない場合、列は数値タイプ、連続尺度で総桁数は 12 となります。

戻り値

列への参照

必須の引数

name 新しい列の名前。

オプションの引数

data type データタイプを指定する引用符付き文字列。オプションには、"Numeric"（数値）、"Character"（文字）、"Row State"（行の属性）、"Expression"（式）があります。

modeling type 尺度を指定する引用符付き文字列（"Continuous"（連続尺度）、"Nominal"（名義尺度）、"Ordinal"（順序尺度）、"Multiple Response"（多重応答）、"Unstructured Text"（非構造化テキスト）、"None"（なし）、"Vector"（ベクトル））。

Format(*format*, *width*) 表示形式と列の幅を設定する。

Set Values({}) 列のデータ値を指定する。

Formula 列の計算式を指定する。

Set Property(*properties*) データテーブル列でサポートされるメッセージを指定する。「列の新規作成」ウィンドウの［列プロパティ］プロパティのリストを確認できます。AxisとLink Referenceは、プロパティです。

次も参照

数値の表示形式を設定する例については、「col<<Format(<width>, <decimal places>, <"Use Thousands Separator">)」を参照してください。

dt<<New Data Box()

ディスプレイボックスツリーに、データテーブルビューを作成する。データテーブルとレポートを1つのウィンドウに表示するのに便利です。データテーブルオブジェクトにNew Data Boxメッセージを送ると、データブラウザボックスが作成されます。

例

次のスクリプトは、データテーブルビューとレポートを作成し、1つのウィンドウに表示します。データテーブルは、データブラウザボックスに配置されます。ボックスの幅は、800ピクセルに設定されます。自動伸縮をオフにしているため、ウィンドウの右端を伸ばしても、データテーブルビューの幅は800ピクセルに保たれます。

```
dtA = Open( "$SAMPLE_DATA/Semiconductor Capability.jmp", invisible );
nw = New Window( "例",
   H List Box(
      V List Box( dtbox = dtA << New Data Box() ),
      dtA << Distribution(
         Continuous Distribution( Column( :NPN1 ) ),
         Continuous Distribution( Column( :PNP1 ) )
      )
   )
);
dtbox << Set Stretch( "Off", "Off" ) << Set Width( 800 );
```

dt<<New Data View

データテーブルの複製を開く。2番目のデータテーブルは元のデータテーブルと同一で、元のデータテーブルにリンクしています。そのため、どちらか一方が変更されると、他方にも反映されます。また、どちらか一方を閉じると他方も閉じられ、データテーブルへの参照はすべて削除されます。

これは、非表示のデータテーブルを表示する際に便利です。

dt<<New Script(*name*, *script*)

dt<<Set Property(*name*, *script*)

script で指定されたスクリプトを保存する新しいテーブルプロパティ（**「テーブルスクリプト」**ともいう）を、引用符付きの *name* を使って作成する。

今後サポートされなくなる New Property() および New Table Property() の代わりに New Script() または Set Property() を使用してください。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Set Property( "二変量の例", Bivariate( Y( :"体重(ポンド)"n ), X( :"身長(インチ)"n ), Fit Line ) );
```

dt<<New Table Variable(*name*, *number*)

dt<<Set Table Variable(*name*, *number*)

引用符付きの *name* と *number* を使って新しいテーブル変数を作成する。

dt<<Next Selected

次の選択行を表示する。次の選択行が画面の外にある場合は、データテーブルを下にスクロールします。

dt<<Optimize Display

すべての列のサイズを再調整して、データテーブルのウィンドウを最適なサイズに拡大する。

dt<<Original Order

データテーブル（*dt*）の列の順を保存されている順序に戻す。

dt<<Paste Column Properties

複数の列プロパティの情報を含むリストを、複数の列に貼り付ける。データテーブルで対象の列を選択しておく代わりに、列のリストを指定することもできます。

例

```
dt = Open( "$SAMPLE_DATA/Tiretread.jmp" );
dt << Copy Column Properties( {:引張応力, :伸び} );
dt2 = New Table( "テスト",
   New Column( "T1", numeric, continuous ),
   New Column( "T2", numeric, continuous ),
   New Column( "T3", numeric, continuous ),
   Add Rows( 10 )
);
dt2 << Paste Column Properties( {:T1, :T3} );
//引張応力と伸びの列プロパティをT1とT3に貼り付ける
```

dt<<Previous Selected

前の選択行を表示する。前の選択行が画面の外にある場合は、データテーブルを上にスクロールします。

dt<<Print Window(<"Show Dialog">)

ウィンドウを印刷する。オプションの名前付き引数 "Show Dialog" が指定されている場合は、印刷ウィンドウが表示されます。そうでない場合は、ウィンドウはデフォルトのプリンタに現在の設定を使って印刷され、印刷ウィンドウは表示されません。

dt<<Rename Script Group(*old name*, *new name*)

説明

テーブルスクリプトグループのグループ名を変更する。

例

```
dt << Rename Script Group( "地図", "ストリートマップ" );
```

dt<<Reorder By Data Type

データテーブル（*dt*）の列を、データタイプによって行の属性、文字、数値の順に並べ替える。

dt<<Reorder By Modeling Type

データテーブル（*dt*）の列を、尺度によって連続尺度、順序尺度、名義尺度の順に並べ替える。

dt<<Reorder By Name

データテーブル（*dt*）の列を名前の昇順に並べ替える。

dt<<Rerun Formulas

テーブル変数を含んだ、列の計算式を、すべて再計算する。再計算は適切な順序で実行されます。

dt<<Reverse Order

データテーブル（*dt*）の列順を逆にする。

dt<<Revert

最後に保存されたデータテーブル（*dt*）に戻す。

dt<<Row Selection(Select Where(condition), <current selection("Extend"|"Restrict"|"Clear")> <Dialog("Keep Dialog Open")>)

説明

指定の条件を満たすすべての行を選択する。

必須の引数

Select Where(condition) 行を選択する条件。

オプションの引数

current selection("Extend"|"Restrict"|"Clear") 現在の選択範囲を拡張するか、制限するか、解除する。デフォルトではClear（解除）が使用されます。

Dialog("Keep Dialog Open") ユーザがオプションを編集できるようにダイアログを表示する。

dt<<Run Formulas

他の計算式を評価した後で評価するために保留されている計算式も含め、データテーブルのすべての計算式の評価を実行する。

dt<<Run Script(*name*)

テーブルパネルにある、引用符付きの *name* で指定されたプロパティの JSL スクリプトを実行する。

dt<<Save(*path*)

dt<<Save As(*path*)

説明

テーブルを、引用符付きの *path* で指定されたパスに保存する。

dt<<Save Database(*connection information*, *table name*, <"Replace">)

データテーブルを、引用符付きの接続情報（*connection information*）と引用符付きのテーブル名（*table name*）で指定されたデータベースに保存する。"Replace"オプションを指定した場合は、既存のデータベースを置換します。

dt<<Save Script to Script Window

データテーブルを再作成するためのスクリプトをスクリプトエディタウィンドウに保存する。スクリプトウィンドウにスクリプトが入力されている場合は、その後ろにスクリプトを追加します。

dt<<Select All Rows

データテーブル（*dt*）のすべての行を選択する。

dt<<Select Columns(*<column1>, <column2>,...*|"All")

データテーブルの、指定された列（またはすべての列）を選択する。

dt<<Select Duplicate Rows

説明

値が重複している行のうち、2番目以降の行を選択する。列が選択または指定されている場合は、それらの列の値が重複した行を探します。重複しているかどうかの判断の際、大文字と小文字が区別されます。

dt<<Select Excluded

データテーブル（*dt*）で現在除外されている行だけを選択する。Where() を使用すると、より柔軟に指定ができます。「JSL の関数、演算子、およびメッセージ」の章の「Where(<dt>, clause)」を参照してください。

dt<<Select Hidden

データテーブル（*dt*）で現在非表示になっている行だけを選択する。Where() を使用すると、より高速に処理できます。また柔軟な指定も可能です。「JSL の関数、演算子、およびメッセージ」の章の「Where(<dt>, clause)」を参照してください。

dt<<Select Labeled

データテーブル（*dt*）で現在ラベルがついている行だけを選択する。Where() を使用すると、より柔軟に指定ができます。「JSL の関数、演算子、およびメッセージ」の章の「Where(<dt>, clause)」を参照してください。

dt<<Select Randomly(*n*|*p*|Sample Size(*n*)|Sampling Rate(*p*))

指定された割合（*p*）、もしくは行数（*n*）でデータテーブルの行を無作為に選択する。引数をキーワードで指定する場合、0 ～ 1 の数値は抽出率、1 より大きい数値は行数を表します。

dt<<Select Rows([*row1, row2, ...*])

行番号のリスト（list）で指定された行を選択する。

dt<<Select Script Group(*<group name|{group1, group2, ...}>*)

引用符付きのグループ名（*group name*）または引用符付きの文字列のリストで 指定されたテーブルスクリプトのグループを選択する。引数がない場合は、すべてのグループを選択します。

dt<<Select Where(*condition*, <Current Selection("Extend"|"Restrict"|"Clear")>)

説明

データテーブル（*dt*）で、条件（condition）が真になる行を選択する。

dt<<Set Dirty(*Boolean*)

変更されていない場合でも、データテーブルを変更済みとしてマークする。

dt <<Set Edit Lock(<"Modify Cells">, <"Add Rows">, <"Add Columns">, <"Delete Rows">, <"Delete Columns">)

説明

セルの変更や、行の追加・削除、列の追加・削除ができないようロックする。

dt <<Set Cell Height(*n*)

セルの高さをピクセル数で指定する。

dt<<Set Header Height(*n*)

セルの見出しの高さをピクセル数で指定する。

dt<<Set Label Columns(*column1, columns2, ...*)

指定の列をラベル列として割り当てる。

dt<<Set Matrix(*[matrix]*)

データテーブルに指定の行列を挿入し、必要に応じて新しい列と行を追加する。

dt<<Set Name(*name*)

説明

テーブルの名前を指定する。*name*引数は引用符付きです。

戻り値

データテーブル名の引用符付き文字列

メモ

Set Nameメッセージが、新しいテーブル名の引用符付き文字列を戻すように変更されました。以前のリリースでは、このメッセージは、スクリプト可能なデータテーブルオブジェクトを戻していました。この変更に伴い、JMPスクリプトを変更する必要がある場合があります。たとえば、以下のスクリプトは書き換える必要があります。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" ) << Set Name( "テスト" );
```

*dt*が「テスト」にならないように、メッセージを分けます。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
dt << Set Name( "テスト" );
```

これは以前のリリースと同じ結果になり、JMPの以前のバージョンと新しいバージョンの両方に対応します。

---

dt<<Set Property(*name*, *script*)

「dt<<New Script(name, script)」を参照してください。

---

dt<<Set Label Columns(*column(s), ...*)

dt<<Set Label Columns

指定した列のラベルの属性をオン（ラベルあり）にする。列がリストされていない場合は、ラベルの属性がオフになります。

---

dt<<Set Row ID Width(*n*)

行番号が表示されている領域の表示幅を指定のピクセル数にする。*n*を0に設定すると、行番号の表示幅が自動的に調整されます。

---

dt<<Set Row States(*[matrix]*)

データテーブル内のすべての行の属性を設定する。

---

dt<<Set Scroll Lock Columns(*column name, ...*)

引用符付きの文字列で指定された列に対し、スクロールをロックする。列が指定されていないときは、スクロールのロックを解除します。

---

dt<<Set Table Variable(*name*, *value*)

「dt<<New Table Variable(name, number)」を参照してください。

---

dt<<Sort(<"Private">|<"Invisible">, <"Replace Table">, By(*columns*), Order("Descending" | "Ascending"), <Output Table Name(*name*))>

説明

列（*columns*）の値に従って、データテーブル（*dt*）の行を並べ替え、新しいデータテーブル（"name"）を作成する。

戻り値

並べ替えたテーブルへの参照

dt<<Split(Split(*columns*), Split By(*column*), <Group(*column*)>, <"Private">|<"Invisible">, <Remaining Columns("Keep All"|"Drop All"|Keep(*columns*)|Drop(*columns*))>, <Copy Formula(*Boolean*)>, <Suppress Formula Evaluation(*Boolean*)>, <Sort by Column Property>, <Output Table (*name*)>)

説明

Split で指定した各列を Split by の列の値により複数の列に分割する。Split と Split by の引数は必須です。

戻り値

列の分割で作成されたデータテーブルへの参照

必須の引数

Split(*columns*) 分割する列。

Split By(*column*) 分割の基準となる列。

オプションの引数

Group 指定のグループ内でデータを分割する。

Remaining Columns("Keep All"|"Drop All"| Keep(*columns*)|Drop(*columns*)) 残りの列を結果のテーブル内に含めるか指定する。デフォルトでは Keep All（すべて保持）。

メモ: Keep All は、出力データテーブルにすべての列を含めます。ただし、各列の**値**は含めません。出力データテーブルでは、複数の行が畳んで1つの行にまとめられるため、保持した列の値の一部が省略されます。

Copy Formula(*Boolean*) ソーステーブルの列の計算式を結果のテーブルに含める。

Suppress Formula Evaluation(*Boolean*) コピーされた列の計算式を自動評価しない。デフォルトは1（真）。

Sort by Column Property 出力列を、基準となる列（Split by）に対して定義された並べ替え列プロパティで並べ替える。

Output Table(*name*) 出力テーブルの名前を指定する。

dt<<Stack(<"Private">|<"Invisible">, Columns(*columns*), <Source Label Column(*quoted string*)>, <Stacked Data Column(*quoted string*)>, <Copy Formula(*Boolean*)>, <Suppress Formula Evaluation(*Boolean*)>, <Drop All Other Columns(*Boolean*)|Name(*non-stacked columns*)(Keep(*column1, column2, ...*))|Name(*non-stacked columns*) (Drop(*column1, column2, ...*))>, <Output Table(*name*)>), <Number of Series(*n*)>, <"Contiguous">

説明

データテーブル（*dt*）内の複数列の値を 1 列（newcol）に積み重ねて、新しいテーブルを作成する。

戻り値

列の積み重ねで作成されたデータテーブルへの参照

dt<<Subscribe("keyname"(<"*client*">), On Delete Columns(<*function*>|<*script*>)|On Add Columns(<*function*>|<*script*>)|On Add Rows(<*function*>|<*script*>)|On Delete Rows(<*function*>|<*script*>)|On Rename Column(<*function*>|<*script*>)|On Close(<*function*>|<*script*>)|On Save(<*function*>|<*script*>)|On Rename(<*function*>|<*script*>))

説明

そのデータテーブルへの変更に関するメッセージを取得するため、データテーブルに登録する。

戻り値

キー名

引数

"keyname"(<*client*>) 登録を参照できるように登録名を指定する。データテーブルで閉じる操作が行われると、引用符付きのclientで指定されたクライアントが、そのデータテーブルに依存するウィンドウがあることを警告し、操作を実行するかどうかを確認します。

On Delete Columns(<*function*>|<*script*>) 列の削除時にキー名を戻す。

On Add Columns(<*function*>|<*script*>) 列の追加時にキー名を戻す。

On Add Rows(<*function*>|<*script*>) 行の追加時にキー名を戻す。

On Delete Rows(<*function*>|<*script*>) 行の削除時にキー名を戻す。

On Rename Column(<*function*>|<*script*>) 列名の変更時にキー名を戻す。

On Close(<*function*>|<*script*>) データテーブルを閉じるときにキー名を戻す。1つの引数（関数）を取ります。その関数は、引数としてデータテーブル名のみを必要とします。

On Save(<*function*>|<*script*>) データテーブルの保存時にキー名を戻す。

On Rename(<*function*>|<*script*>) データテーブル名の変更が試みられると、キー名を戻す。関数（function）は、前に定義した関数か、その関数自体の名前です。

メモ

それぞれのオプションは、解除するまで有効です。

dt<<Subset(<"Private">|<"Invisible">, <"Selected Columns">, <Columns(*column list*)>, <"Selected Rows">, <Rows(*[number, number, ...]*)>, <By(*column list*)>, <Sampling Rate(*fraction*)>, <Sample Size(*integer*)>, <Stratify(*column list*)>, <Link to Original Data Table(*Boolean*)>, <Copy Formula(*Boolean*)>, <Suppress Formula Evaluation(*Boolean*)>, <"Keep by Columns">)

説明

データテーブル（*dt*）から指定された行と列を抽出し、新しいデータテーブルを作成する。

戻り値

サブセットデータテーブルへの参照

dt<<Summary(<"Private">|<"Invisible">, <Group(*column*)>, <Subgroup(*column*)>, <N(*column*)>, <Mean(*column*)>, <Std Dev(*column*)>, <Min(*column*)>, <Max(*column*)>, <Range(*column*)>, <Sum(*column*)>, <CV(*column*)>, <Freq(*column*)>, <Weight(*column*)>, "Include Marginal Statistics", <Link to Original Data Table(*Boolean*)>, <Statistics Column Name Format(*Stat(column)*|*Column*|*Stat of Column*|*Column Stat*)>)

説明

指定した列の要約統計量を含むデータテーブルを（オプションで、グループやサブグループごとに）作成する。Statistics Column Name Format の値は引用符付きです。

戻り値

要約データテーブルへの参照

dt<<Suppress Formula Eval(*Boolean*)

引数が 1 の場合、データテーブル（*dt*）の計算式の自動評価をオフにする。引数が 0 の場合は、オンにする。

dt<<Text to Columns(Delimiters(<*separator*>, <"Tab">, <"Newline">), Columns(*column1, column2...*))

区切り文字で区切ったテキストから、テキストごとの列もしくは指示変数の列を作成する。"newline" には、\r、\n、\r\n の 3 つのいずれかを指定できます。区切り文字は引用符付きです。

例

```
dt = Open( "$SAMPLE_DATA/Consumer Preferences.jmp" );
dt << Text To Columns(
   delimiter( "," ),
   columns( :歯磨き カンマ区切り )
);
```

dt<<Transpose(Columns(*columns*), Rows([*matrix*]), Output Table Name(*name*))

説明

指定の行と列を転置し、（引用符付きの *name* で指定された名前の）新しいデータテーブルを作成する。

戻り値

転置したデータテーブルへの参照

dt<<Ungroup Columns(*{column1, column2, ...}*)

リスト引数で定義された列のグループを解除する。

dt<<Ungroup Scripts(*Name of Script Group*|*{script1, script2,...}*)

説明

指定したテーブルスクリプトまたはグループをグループから除外する。*Name of Script Group*引数は、引用符付きです。

dt<<Unsubscribe(*keyname*, "On Delete Columns"|"On Add Columns"|"On Add Rows"|"On Delete Rows"|"On Close"|"On Col Rename"|"All")

データテーブル（*dt*）への以前の登録を解除する。*keyname*引数は引用符付きです。

dt<<Update from Database

データテーブル（*dt*）内のデータを、データベースから再読み込みしたデータで更新する。

列のメッセージ

col<<Add Column Properties(*name*, *expression*)

指定の式（*expression*）を持つ（引用符付きの *name* で指定された）プロパティを追加する。標準の列プロパティまたはユーザ指定のプロパティを追加できます。

col<<Add From Row States

データテーブルの各行に設定されている「行の属性」を、列に含まれている「行の属性」に追加する。

col<<Add To Row States

列に保存されている行属性値をすべて、データテーブルで使われる行の属性に追加する。

col<<Color Cells(*color*)

説明

データテーブルグリッドの列のセルに色をつける。引用符付きで指定された任意の色名を使用します。色をクリアする場合は 0 を指定します。

col<<Color Cell by Value(*Boolean*)

説明

「値の色」プロパティに基づいて、データテーブルグリッドの列のセルに色をつける。

col<<Copy Column Properties

列プロパティをバッファ内にコピーする。

col<<Copy From Row States

現在のデータテーブルにおけるすべての行属性を列にコピーする。

col<<Copy to Row States

列に含まれている「行の属性」を、データテーブルの各行がもつ「行の属性」に設定する。

col<<Data Type(*type*, <Format(*format quoted string*)>, <Input Format(*format quoted string*)>, <*width*>)

col<<Set Data Type(*type*, <Format(*format quoted string*)>, <Input Format(*format quoted string*)>, <*width*>)

説明

データタイプ（*data type*）を列（*col*）に設定する。

必須の引数

type データタイプとして"Numeric"（数値）、"Character"（文字）、"Row State"（行の属性）、"Expression"（式）を指定する。

オプションの引数

Format(*format quoted string*) データの表示形式を指定する（たとえば、時間と分をh:m形式で表示する、など）。*format quoted string*引数は引用符付きです。

Input Format(*format quoted string*) データの出力形式を指定する。*format quoted string*引数は引用符付きです。

width（数値データ向けのオプション）1、2、または4（列のバイト数）を指定する。

col<<Delete Formula

列から計算式を削除する。

col<<Delete Property(*name*)

col<<Delete Column Property(*name*)

引用符付きで指定された名前（*name*）の列プロパティを削除する。

col<<Eval Formula

計算式を強制的に評価する。［自動評価しない］オプションが有効な場合、評価は行われません。

col<<Exclude(*Boolean*)

指定されたブール引数に従って、「除外する」の属性をオン／オフにする。

col<<Format(<*width*>, <*decimal places*>, <"Use Thousands Separator">)

col<<Format("Best", <*width*>, <"Use Thousands Separator">)

col<<Format(("Fixed Dec"|"Percent"), <*width*>, <*decimal places*>, <"Use Thousands Separator">)

col<<Format("Pvalue", <width>)

col<<Format(("Scientific"|"Engineering"|"Engineering SI"), <*width*>, <*decimal places*>)

col<<Format("Precision", <*width*>, <*decimal places*>, <"Use Thousands Separator">, <"Keep Trailing Zeroes">, <"Keep All Whole Digits">)

col<<Format("Currency", <"Currency Code">, <*width*>, <*decimal places*>, <"Use Thousands Separator">)

col<<Format("Datetime", <*width*>, <*input format*>)

col<<Format(("Latitude DDD"|"Latitude DDM"|"Latitude DMS"|"Longitude DDD"|"Longitude DDM"|"Longitude DDM"), <*width*>, <*decimal places*>, ("PUN"|"DIR"|"PUNDIR"))

col<<Format("Custom", Formula(...), <*width*>, <*input format*>)

説明

数値の表示形式を設定する。

引数

引数の詳細については、『JMPの使用法』を参照してください。

例

```
col<<Format( 10, 2, "Use Thousands Separator");
col<<Format( "Currency", "EUR", 20 );
col<<Format( "m/d/y", 10 );
col<<Format( "Precision", 10, 2, "Keep Trailing Zeroes", "Keep All Whole Digits"
   );
col<<Format( "Latitude DDD", "PUNDIR"); // "PUN"はpunctuation（フィールド句読記号）、
   "DIR"はdirection（方角）、PUNDIRは両方
col<<Format( "Custom", Formula( Abs( value ) ), 15 );
```

col<<Formula(*expression*)

col<<Set Formula(*expression*)

列に計算式を設定し、その計算式を評価する。

col<<Get Column Field Width

データを表示するのに使用されている、列のフィールド幅を戻す。

col<<Get Data Type

列（*col*）のデータタイプを戻す。

col<<Get Data Type Length

データ列のデータタイプと長さを戻す。文字タイプの列など、データの長さが固定されていない場合は、データタイプのみを戻します。

col<<Get Format

列の表示形式を戻す。

col<<Get Formula

計算式を戻す。

col<<Get Hidden

列が非表示の場合に 1 を戻す。

col<<Get Input Format

その列へのデータの入力および保存に適用されている形式を戻す。

col<<Get Labeled

列がラベルありに設定されている場合に 1 を戻す。

col<<Get List Check

リストチェックの定義を戻す。列にリストチェックが定義されていない場合は、そのことを示すメッセージがログに送られます。

col<<Get Lock

現在のロック設定を戻す。

col<<Get Modeling Type

列の尺度を戻す。

col<<Get Name

列の名前を戻す。

col<<Get Property("property name")

指定のプロパティ定義を戻す。列にそのプロパティが定義されていない場合は、そのことを示すメッセージがログに送られます。

col<<Get Range Check

範囲チェックの定義を戻す。列に範囲チェックが定義されていない場合は、そのことを示すメッセージがログに送られます。

col<<Get Role

列（*col*）に事前に割り当てられている役割を戻す。

col<<Get Script

列を再作成するスクリプトを戻す。

col<<Get Scroll Locked

列がスクロールロックされている場合に 1 を戻す。

col<<Get Selected

列が選択されている場合は 1、そうでない場合は 0 を戻す。

col<<Get Stored Values

「欠測値のコード」列プロパティを無視して、列の値をそのまま戻す。

col<<Get Value Labels

値ラベルの定義を戻す。列に値ラベルが定義されていない場合は、そのことを示すメッセージがログに送られます。

col<<Get Use Value Labels

列に値ラベルを使用するよう設定されている場合は 1、そうでない場合は 0 を戻す。

col<<Get Values

列の値を戻す。

col<<Hide(Boolean)

指定されたブール引数に従って、「表示しない」の属性をオン／オフにする。

col<<Ignore Errors

列内の計算式を評価中にエラーが発生した場合、セルの値を欠測値に設定する。

col<<Input Format(*format*)

その列へのデータの入力および保存に使用される形式を、引用符付きの *format* で指定された形式に設定する。引数は、任意のJMP形式の名前（日付値の列なら "ddmmyyyy" など）です。

date_col<<Is Transformed On SAS Export

データをSASに書き出してSASデータセットを作成する際、日付列のデータが変更される場合は真を戻す。

col<<Label(*Boolean*)

指定されたブール引数に従って、ラベル属性をオン／オフにする。

col<<Lock(*Boolean*)

col<<Set Lock(*Boolean*)

指定されたブール引数に従って、ロック属性をオン／オフにする。

col<<Preselect Role(*role*)

列（col）の役割（*role*）を事前に設定しておく。選択肢は、"Y"、"X"、"Weight"（重み）、"Freq"（度数）、"None"（なし）または "No Role"（役割なし）です。

col<<Reset Transform

変換列のキャッシュデータを削除する。列のデータにアクセスするとキャッシュが再び構築される。このオプションを利用することで、メモリを減らしたり、外部情報に依存する計算式の再計算を可能にしたりできる。

col<<Set Display Width(*n*)

列の表示幅を *n* に設定する（単位はピクセル）。*n* を0に設定すると、列の表示幅が自動的に調整されます。

col<<Set Each Value(*n*)

列のすべての値を *n* に設定する。

col<<Set Excluded

列を除外する。

col<<Set Field Width(*n*)

列のフィールド幅を*n*に設定する。

col<<Set Hidden

列を非表示にする。

col<<Set Labeled

col<<Set Labelled

列のデータ値をラベルに使用する。

col<<Set Modeling Type(*type*)

変数の尺度（type）を設定する。選択肢は、"Continuous"（連続尺度）、"Ordinal"（順序尺度）、"Nominal"（名義尺度）、"None"（なし）、"Row State"（行の属性）、"Unstructured"（非構造化テキスト）、"Multiple Response"（多重応答）、または "Vector"（ベクトル）です。

col<<Set Name(*name*)

列の名前を設定する。*name*引数は引用符付きです。

col<<Set Property(*name*, *expression*)

指定された式（*expression*）をプロパティ（引用符付きの *name*）に設定する。標準の列プロパティまたはユーザ指定のプロパティを設定できます。

例

次の例は、「性別」列に「値の色」列プロパティを追加し、「F」の値をピンク、「M」の値を青に設定します。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
Column( "性別" ) << Set Property( "Value Colors", {"F" = 78, "M" = 69} );
```

次の例は、**「身長(インチ)」**列に「記録日」という名前の独自の列プロパティを追加します。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
Column( "身長(インチ)" ) << Set Property( "記録日", 05Jan1990 );
```

次も参照

『スクリプトガイド』

『JMPの使用法』

col<<Set Scroll Locked(*Boolean*)

指定されたブール引数に従って、スクロールロック属性をオン／オフにする。

col<<Set Selected(*Boolean*)

列を、選択された状態または選択されていない状態に設定する。

col<<Set Use for Marker

col<<Use for Marker

列の値をグラフのマーカーとして使用する。式（画像）の列や、行ごとに一意の値を持つ文字タイプの列に適しています。「Big Class Families.jmp」サンプルデータテーブルでは、「写真」列がグラフのマーカーとして指定されています。バブルプロットではサポートされていません。

col<<Set Values(*[matrix]* or *{list}*)

col<<Values(*[matrix]* or *{list}*)

行列（数値変数の場合）またはリスト（文字変数の場合）の値を列の値として設定する。

col<<Suppress Eval(*Boolean*)

列の計算式に対する自動評価を、オフにする。

col<<Use For Marker(*Boolean*)

列の値をグラフのマーカーとして使用するか、オプションをオフにする。式（画像）の列や、行ごとに一意の値を持つ文字タイプの列に適しています。「Big Class Families.jmp」サンプルデータテーブルでは、「写真」列がグラフのマーカーとして指定されています。

行のメッセージ

row<<Colors(*n*)

選択されている行に番号 *n* の色を割り当てる。

row<<Exclude(*Boolean*)

row<<Unexclude(*Boolean*)

指定されたブール引数に従って、行の「除外する」の属性をオン／オフにする。引数を省略すると、現在の状態と逆に設定されます。

row<<Hide(*Boolean*)

row<<Unhide(*Boolean*)

指定されたブール引数に従って、「表示しない」の属性をオン／オフにする。引数を省略すると、現在の状態と逆に設定されます。

row<<Hide and Exclude

選択されている行を非表示かつ除外にするか、表示し、かつ計算に含める。

row<<Label(*Boolean*)

row<<Unlabel(*Boolean*)

指定されたブール引数に従って、ラベル属性をオン／オフにする。引数を省略すると、現在の状態と逆に設定されます。

row<<Markers(*marker*)

選択されている行に引用符付きの "*marker*" で指定されたマーカーを割り当てる。

row<<Next Selected

次の選択行を点滅させる。

row<<Previous Selected

前の選択行を点滅させる。

row<<Row Editor

選択されている行の編集ウィンドウを開く。

データフィルタのメッセージ

dtf<<Add Favorites(*name*)

説明

現在のフィルタの選択範囲を、引用符付きの *"name"* で指定された名前を付けて「お気に入り」リストに保存する。

戻り値

お気に入りを引用符付き文字列として戻す。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
```

```
df = dt << Data Filter(
   Add Filter(
      Columns( :年齢, :性別, :"身長(インチ)"n, :"体重(ポンド)"n ),
      Where( :性別 == "F" ),
      Where( :"身長(インチ)"n >= 55 & :"身長(インチ)"n <= 65 )
   ),
   Mode( Select (1) )
);
Wait( 1 ); // デモ用
fav1 = df << Add Favorites( "平均的な身長の女子" );
```

dtf<<Add Filter(Columns(*column1*, <*column2*>), <Where(*clause*)>)

1つまたは複数のフィルタ列をORの条件で追加する。

dtf<<Auto Clear(*Boolean*)

新しい選択項目を設定する前に、現在選択されている行をすべてクリアする。

dtf<<Clear

現在選択されている行をクリアする。

dtf<<Close

データフィルタウィンドウを閉じる。

dtf<<Columns(*column1, column2, ...*)

データフィルタで使用する列を設定する。

dtf<<Data Table Window

データフィルタウィンドウが使用しているデータテーブルを表示する。

dtf<<Delete All

設定されているすべてのフィルタを削除する。

dtf<<Delete(*column1, column2, ...*)

指定の列をデータフィルタから削除する。

dtf<<Display(*column*, <Size(*x, y*)>, "Blocks Display"|"List Display"|"Single Category Display"|"Checkbox Display")

指定のカテゴリカル列の水準を、どのようにフィルタに表示するかを設定する。

dtf<<Get Script

データフィルタのスクリプトをテキストとしてログに戻す。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
df = dt << Data Filter(
     Add Filter( Columns( :年齢, :性別 ), Where( :年齢 == 12 ) )
);
txt = df << Get Script;
Show( txt );
```

dtf<<Local Data Filter

指定のウィンドウにデータフィルタを組み込む。ローカルデータフィルタについて詳しくは、『スクリプトガイド』を参照してください。

dtf<<Location(*x*, *y*)

データフィルタウィンドウを指定の場所に移動する。*x*と*y*はピクセル単位で指定します。0,0はディスプレイの左上隅を意味します。

dtf<<Make Filter Change Handler(*function*)

フィルタが変わったという通知を扱うデータフィルタハンドラを作成する。フィルタされる行の数は、関数の引数の中で戻されます。

例

```
dt = Open( "$SAMPLE_DATA/PopAgeGroupSubset.jmp" );
dist = Distribution( Automatic Recalc( 1 ), Continuous Distribution( Column( :人口
  ) ) );
filter = dist << Local Data Filter( Add Filter( Columns( :地域 ) ) );
f = Function( {a}, Print( a ) );
rs = filter << Make Filter Change Handler( f );
```

dtf<<Make Subset

データフィルタで選択されている行を含む、新しいサブセットデータテーブルを作成する。

dtf<<Match(Filter Columns(*column1*, column2, ...), Where(*clause*)

各列のフィルタ条件を設定する。*Where節*は、リストされたすべての列で使用されます。列ごとに異なるWhere節を使用するには、Matchメッセージを各列に個別に送ります。

dtf<<Mode(Select(*Boolean*)|Show(*Boolean*)|Include(*Boolean*))

データフィルタを使って行が選択されたときのアクションまたはモードを設定する。

dtf<<Save and Restore Current Row States

データテーブルの現在の「行の属性」を保存しておき、その後でデータフィルタを閉じたときに、それらの属性を復元する。

dtf<<Show Columns Selector(*Boolean*)

フィルタの完了後、列セレクタを表示または非表示にする。

dtf<<To Clipboard

データフィルタの現在の状態からWhere節を作成し、クリップボードに置いて、別の場所に貼り付けることができるようにする。

dtf<<To Data Table

データフィルタの現在の状態からWhere節を作成し、プロパティとしてデータテーブルに保存する。

dtf<<To Journal

データフィルタの現在の状態からWhere節を作成し、現在のジャーナルに追加する。現在のジャーナルが存在しない場合、新しいジャーナルが開き、Where節はそれに追加されます。

dtf<<To Row State Column

計算式がWhere節である行の属性列を作成する。

dtf<<To Script Window

データフィルタの現在の状態からWhere節を作成し、現在のスクリプトウィンドウに追加する。現在のスクリプトウィンドウが存在しない場合、新しいスクリプトウィンドウが開き、Where節はそれに追加されます。

dtf<<Use Floating Window(*Boolean*)

データフィルタウィンドウを、関連するデータテーブルの上に配置するか、通常のウィンドウとして表示するかを設定する。

dtf<<Where(*clause*)

行選択の条件を設定する。

データフィードのメッセージ(Windowsのみ)

feed<<Close

データフィードオブジェクトとそのウィンドウを閉じる。

feed<<Connect(*port settings*)

デバイスに接続するために、ポートの設定を行う。

feed<<Disconnect

データフィードオブジェクトをアクティブにしたまま、データフィードのキューとデバイスとの接続を切断する。

feed<<EOL("CR", "LF", "CRLF")

入ってくるデータ行を解析する際に、区切り文字として使用する行の終わりの値を指定する。この値は、データを送る場合にもデータ行の終わりとして使用されます。

- "CR": ASCIIコード13（キャリッジリターン）
- "LF": ASCIIコード10（ラインフィード）
- "CRLF": CRとLFの両方を続けて使用。

feed<<Get Line

データフィードのキューの中から1ラインのデータを戻し、削除する。

feed<<Get Lines

データフィードのキューのデータすべてをリストにして戻し、削除する。

feed<<Print Queue

内部のメッセージのキューをログウィンドウに表示する。

feed<<Queue Line(*quoted string*)

引用符付きの文字列（*string*）またはライン（line）をデータフィードのキューの最後に送る。Queue Lineを使うと、デバイスを接続せずにスクリプトをテストすることができます。デバイスから届くデータをシミュレートし、実際にデバイスを接続したときに値を適切に処理できるかどうかを確認するのに役立ちます。

feed<<Restart

データラインへの処理を再開する。

feed<<Set Script(*script*)

データラインが届くたびに実行されるスクリプト（*script*）を設定する。

feed<<Stop

データラインへの処理を停止する。

feed<<Write(*quoted string*)

説明

データフィードの機器に引用符付きの文字列（*string*）を送る。

例

```
/* 例 - シリアルポートを介して外部デバイスにメッセージを送信する。センサーなど、接続したデバイスに制御メッセージを送るのに使用できる。*/
exfeed = Open Datafeed(
   Connect( Port( "com1" ), Baud rate( 4800 ), Parity( "even" ), DataBits( 8 ) ),
   Set Script(
      ex = exfeed << Get Line;
      Show( ex );
   )
);
exfeed << Write( "Ready" );
```

feed<<Write Line(*quoted string*)

説明

データフィードの機器に引用符付きの文字列（*string*）を送る。データフィードに EOL が設定されている場合、引用符付き文字列は、指定した EOL 値で終わります。EOL が設定されていない場合、行は CRLF で終わります。

例

```
/* シリアルポートを介して外部デバイスにメッセージを送信する。センサーなど、接続したデバイスに制御メッセージを送るのに使用できる。*/
exfeed = Open Datafeed(
   Connect( Port( "com1" ), Baud rate( 4800 ), Parity( "even" ), DataBits( 8 ) ),
   Set Script(
      ex = exfeed << Get Line;
      Show( ex );
   )
);
```

```
exfeed << Write Line( "Ready" );
```

feed<<Write Lines(*{quoted string1, quoted string2, quoted string3}*)

説明

データフィードの機器に引用符付き文字列（*"strings"*）のリストを送る。データフィードにEOLが設定されている場合、引用符付き文字列は、指定したEOL値で終わります。EOLが設定されていない場合、行はCRLFで終わります。

例

```
/* シリアルポートを介して外部デバイスにメッセージを送信する。センサーなど、接続したデバイスに制
   御メッセージを送るのに使用できる。*/
exfeed = Open Datafeed(
   Connect( Port( "com1" ), Baud rate( 4800 ), Parity( "even" ), DataBits( 8 ) ),
   Set Script(
      ex = exfeed << Get Line;
      Show( ex );
   )
);
exfeed << Write Lines( {"Ready", "Set", "Go"} );
```

ディスプレイボックスのメッセージ

この節では、ディスプレイボックスに使用できるJSLメッセージをご紹介します。他の例については、JMPの［スクリプトの索引］を参照してください。

すべてのディスプレイボックスのメッセージ

db<<Add Text Annotation(Text(*quoted string*), Text Box(*<x1, y1, x2, y2>*))

ピクセル単位で指定した位置に、文字列（引用符付きの *string*）を含むテキスト注釈ボックスを描画する。Text Box引数によって、テキスト注釈ボックスを描く位置を、ウィンドウにおける相対的な位置（左上から右下）で指定します。

x1、*y1*、*x2*、*y2* には、グラフの軸の値ではなく、ウィンドウ内のピクセル単位の位置を指定します。テキストボックスが表示される位置は、ユーザーのウィンドウサイズやディスプレイの解像度などにより異なる可能性があります。

db<<Append(*db2*)

db2 を *db* の最後の子として追加する。

db<<Child

ボックスの子を戻す。

db<<Class Name

ディスプレイボックスの表示クラス名を戻す。

db<<Clone Box

ディスプレイボックスの新しいコピーを作成する。

db<<Close Window

ディスプレイボックスが表示されているウィンドウを閉じる。

db<<Copy Picture

ディスプレイボックスのイメージをクリップボード上に置く。

db<<Delete

ディスプレイボックスを削除する。

db<<Enable(*Boolean*)

ディスプレイボックスを操作可能にするかどうかを指定する。0 の場合はディスプレイボックスを操作できず、1 だとディスプレイボックスを操作できます。

db<<Get HTML

ディスプレイボックスの HTML ソースを含んだ引用符付き文字列を戻す。

db<<Get Journal

ディスプレイボックスのジャーナルソースを含んだ引用符付き文字列を戻す。

db<<Get Menu Item State(*index*)

指定されたポップアップメニュー項目の状態を戻す。状態は、通常（0）、選択されている（1）、または選択不可（-1）のいずれか。

db<<Get Menu Items

説明

ボタンがクリックされたときに呼び出されるポップアップメニューの項目を戻す。メニュー項目はリストで戻されます。

次も参照

サブメニューについては、「db<<Get Submenu(index)」を参照してください。

db<<Get Menu Script

オブジェクトに指定されているメニュースクリプトを戻す。

db<<Get Page Setup()

ページ設定の設定内容を戻す。

例

下の例では、新しいウィンドウを作成し、ページ設定の設定内容を戻します。

```
w = New Window( "Window",
   Text Box( "Page Setup Test" )
);
w << Get Page Setup();
```

メッセージの結果は次のとおりです。

```
{Margins( {0.75, 0.75, 0.75, 0.75} ), Scale( 1 ), Portrait( 1 ),
Paper Size( "Letter" )}
```

db<<Get Picture(<Scale(*n*)>)

ディスプレイボックス（*db*）をイメージオブジェクトとして取得する。Scale(*n*) 引数は、元のイメージサイズが基準となります。たとえば、Scale(2) を指定すると、イメージオブジェクトのサイズが 2 倍になります。

db<<Get RTF

ディスプレイボックスの RTF ソースを含んだ引用符付き文字列を戻す。

db<<Get Script

ディスプレイボックスを再作成するためのスクリプトを戻す。

db<<Get Size

{ *x, y* } または { *h, v* } をピクセルで戻す。

```
xy = DisplayBox << Get Size;
```

xとyをピクセルで戻す。

```
{ x, y } = DisplayBox << Get Size;
```

db<<Get Submenu(*index*)

指定されたメニュー項目の下位にあるサブメニュー項目の数を戻す。

例

下の例は、*"A"*、*"B"*、*"C"*というメニュー項目があるメニューを作成します。*"A"*にはサブメニュー項目*"A1"*と*"A2"*を、*"B"*にはサブメニュー項目*"B1"*、*"B2"*、*"B3"*を設定しています。<<Get Submenu(*inc*)は、インデックスを指定した各メニュー項目の下位にあるサブメニュー項目の数を戻します。

```
New Window( "Title",
obj = Outline Box( "title" ) );
submenus = { };
obj << Set Menu Script(
   {"A", "", "A1", Print( "A1" ), "A2", Print( "A2" ),
   "B", "", "B1", Print( "B1" ), "B2", Print( "B2" ), "B3", Print( "B3" ),
   "C", Print( "C" )}
);
obj << Set Submenu( 1, 2 ); // メニューA、サブメニューにA1とA2の2項目
obj << Set Submenu( 4, 3 ); // メニューB、サブメニューにB1、B2、B3の3項目
For( inc = 1, inc <= N Items( Words( obj << Get Menu Script, "," ) )/2, inc++,
   Insert Into( submenus, obj << Get Submenu( inc ) );
);
submenus;
   {2, 0, 0, 3, 0, 0, 0, 0}
```

このログ出力は、インデックス（1）にサブメニュー項目が2つ、インデックス（3）にサブメニュー項目が3つあることを示しています。

db<<Get Text

ディスプレイボックスのテキストを含んだ引用符付き文字列を戻す。

db<<Horizontal Alignment(*position*)

ディスプレイボックスが入れ子になっている場合、子ディスプレイボックスの位置を*position*で指定する。デフォルト値は"Left"（左揃え）です。"Center"（中央揃え）、または"Right"（右揃え）を指定することもできます。

例

```
New Window( "例",
   Outline Box( "親ディスプレイボックス",
      Button Box( "OK", <<Horizontal Alignment( "Center" ) )
   )
);
```

db<<Inval

ウィンドウ内のディスプレイボックス領域を無効にする。ウィンドウは、次回、オペレーティングシステムによってウィンドウが更新される際（たとえば、ユーザがディスプレイボックスのサイズを変更したとき）に、更新されます。

メモ

Wait(0)を使う代わりに、メッセージ<<Update Windowの使用を検討してください。Wait(*n*)を使う場合は、*n*をどの程度大きい値にするか決めておく必要がある点が問題です。

ディスプレイボックスの多くのメッセージは（<<Set Textなど）、ボックスを自動的に無効としてマークするため、<<Invalメッセージは通常不要です。スライダとJSLコールバックを使うインタラクティブなスクリプトでは、ディスプレイの各所をスライダと同期させるために、<<Update Windowが必要になる場合があります。

db<<Is Enabled

コントロールが有効になっているかどうかの状態を戻す。このメッセージは、Busy Light Box()、Button Box()、Calendar Box()、Check Box()、Col List Box()、Combo Box()、Completion Box()、Filter Col Selector()、gtext()、List Box()、Number Edit Box()、Popup Box()、Radio Box()、Range Slider Box()、Slider Box()、Spin Box()、Text Edit Box()、Tree Box()、Tree Map Box()、Tree Map Seg()でサポートされています。

db<<Journal

ディスプレイボックスをジャーナルに追加する。

db<<Journal Window

ディスプレイボックスが表示されているウィンドウ全体をジャーナルに追加する。Journalと比較してください。

db<<Move Window(*x*, *y*)

ウィンドウをスクリーン上の (*x*, *y*) の位置に移動する。

db<<Page Break

ディスプレイボックスの前にページ区切りを挿入する。

db<<Parent

ディスプレイボックスの親を戻す。

db<<Prepend(*db2*)

db2 を表示ツリーの *db* の前に追加する。

db<<Prev Sib

ディスプレイボックスの前の兄弟（同レベルのもの）を戻す。

db<<Reshow

ウィンドウ内のディスプレイボックスの領域を無効にし、ウィンドウの内容を更新する。

db<<Save Capture(<*path*>, <*format*>, <Add Sibling(*n*)>)

ディスプレイボックスをスクリーンキャプチャーし、指定のパス（引用符付きの *path*）に、指定の形式（引用符付きの *format*）のグラフィックで保存する。オプションの引数（Add Sibling）は、取り込む兄弟ディスプレイボックスの数を指定します。デフォルト値は 1 で、この場合、指定のディスプレイボックスだけを取り込みます。ディスプレイボックスが、別のウィンドウに隠されていたり、スクロールしないと表示されなくなっていたりする時には注意が必要です。ディスプレイボックスが画面に表示されていない場合は、期待どおりの結果にはならないかもしれません。

パスを省略した場合、パスの実行時にファイルに名前を付けて保存するよう促されます。

db<<Save HTML(<*path*>, <*format*>)

HTML ソースファイルとグラフィックファイルのフォルダを、指定のパス（引用符付きの *path*）に指定の形式（引用符付きの *format*）で保存する。*path* 引数を省略した場合、スクリプトを実行した時点で、ファイルに名前を付けて保存するよう促されます。

db<<Save Interactive HTML(<*path*>, "Is Static")

ディスプレイボックスを、インタラクティブ HTML 機能を使った Web ページとして指定のパス（引用符付きの *path*）に保存する。こうすれば、JMP を使用していないユーザーでも、データを見ることができます。データは Web ページに組み込まれます。

引数

path Web ページの保存場所を示すオプションの引用符付きのパス（*path*）。

"Is Static" Web ページからデータを削除し、そのページの静的なコピーを保存する。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
biv = dt << Bivariate( y( :"体重(ポンド)"n ), x( :"身長(インチ)"n ) );
rbiv = (biv << Report);
rbiv << Save Interactive HTML( "$DOCUMENTS/MyInteractiveHTML.htm" );
```

db<<Save Journal(<*path*>)

ボックスのジャーナルソースを指定のパス（引用符付きの *path*）に保存する。引数を省略した場合、グラフィックに名前を付け、タイプを指定するよう促されます。

db<<Save MSWord(<*path*>)

（Windows のみ）ディスプレイボックスを Microsoft Word 文書として指定のパス（引用符付きの *path*）に保存する。*path* 引数を省略した場合、スクリプトを実行した時点で、ファイルに名前を付けて保存するよう促されます。

db<<Save PDF(<*path*>, <Show Page Setup(*Boolean*)>, <Portrait(*Boolean*)>)

説明

ディスプレイボックスをPDFファイルとして指定のパス（引用符付きの *path*）に保存する。

オプションの引数

path ファイルを指定のパス（引用符付きの *path*）に保存する。この引数を省略した場合、スクリプトを実行した時点で、ファイルに名前を付けて保存するよう促されます。

Show Page Setup(*Boolean*)（Windowsのみ）用紙の向きやヘッダ、フッタ、余白、ページの倍率、用紙サイズなどが指定できる「ページ設定」ウィンドウを表示する。

Portrait(*Boolean*) 内容を縦または横に表示する。

メモ

PDFファイルには、ヘッダとフッタが含まれます。ヘッダやフッタを省略したい場合は、Save Picture を使用してください。

db<<Save Picture(<*path*>, <*format*>)

説明

ディスプレイボックスの画像を指定のパス（引用符付きの *path*）に指定の形式（引用符付きの *format*）で保存する。

メモ

- 引用符付きの *path* 引数を省略した場合、スクリプトを実行した時点で、ファイルに名前を付けて保存するよう促されます。
- 有効な形式には、"PDF"、"PNG"、"GIF"、"JPG"、"JPEG"、"EPS"、"SVG"、"PICT"（macOS）、"EMF"（Windows）があります。
- Windowsの場合は、環境設定の［Windowsのみ］カテゴリで解像度（DPI）を指定できます。または、次のスクリプトを使用できます。

  ```
  Pref( Save Image DPI( number ) );
  ```

- macOSでは、オペレーティングシステムによってDPIが決められます。

- レポートを、ヘッダやフッタのないPDFファイルとしてエクスポートするには、Save Pictureを使用します。ヘッダやフッタを含めるには、Save PDFを使用してください。

---

db<<Save Presentation(<*path*>, <Template(*path*)>, <Insert("Begin"|"End"|*n*)|Replace("Begin"|"End"|*n*)|Append>, <Outline Titles(*title location*)>, <*format*>)

Microsoft PowerPointファイルにディスプレイボックスを保存する。このファイルは、任意のプレゼンテーションソフトウェアプログラムで開くことができます。

オプションの引数

path ファイルを指定のパス（引用符付きの*path*）に保存する。ファイル名には拡張子.pptxを含める必要があります。*path*引数を省略した場合、スクリプトを実行した時点で、ファイルに名前を付けて保存するよう促されます。

Template(path) カスタムのPowerPointテンプレートのパス（引用符付きの*path*）を指定する。この引数が指定されていない場合は、インストールディレクトリ内のpptxフォルダにあるデフォルトのテンプレートが使用されます。

テンプレートには簡単なテーブルを含めるようにしてください。そうでない場合は、レポートテーブルにデフォルトのテーブル形式が適用されます。Windowsでの例については、JMPインストールフォルダの¥pptx¥JMPExportTemplate.pptxを参照してください。

Insert 既存のプレゼンテーション内でスライドを挿入する位置を指定する。

– *n*は、スライドを*n*番目のスライドとして挿入することを意味します。

– "Begin"は、スライドをプレゼンテーションの冒頭に挿入します。

– "End"は、スライドをプレゼンテーションの末尾に挿入します。

Replace 既存のプレゼンテーション内で交換するスライドを指定する。引数は、Insertの場合と同様に、*n*、"Begin"、"End"です。

Append スライドを既存のプレゼンテーションの末尾に挿入する。

Outline Titles スライドのアウトラインタイトルと親アウトラインタイトルの位置。デフォルトでは、スライドの内容の上に1つ上の親アウトラインタイトルがスライドタイトルとして表示され、他にも親アウトラインタイトルがある場合はスライドの左下角に表示されます。

– "None"は、グラフィックの上のスライドタイトルとアウトラインタイトルを省略します。

– "Hide"は、アウトラインタイトルを省略します。

– "TopLeft"（左上）、"TopRight"（右上）、"BottomLeft"（左下）、"BottomRight"（右下）により、スライド上の親アウトラインタイトルの位置が決まります。

format 埋め込みグラフィックの形式。オプションは、"Native"、"EMF"、"PNG"、"JPG"、"BMP"、"GIF"、"TIF"です。Windowsの場合、ネイティブ形式はEMFです。macOSの場合、ネイティブ形式はPDFです。互換性の問題については、「メモ」を参照してください。この引数を指定しなかった場合は、［一般］環境設定の［PowerPointのイメージ形式］での設定が適用されます。

メモ

Windowsは、macOSで作成したネイティブのPDFグラフィックをサポートしていません。macOSは、Windowsで作成したネイティブのEMFグラフィックをサポートしていません。クロスプラットフォームの互換性を保つためには、"PNG"、"JPG"、"GIF"、または"TIF"を指定してください。

引数を指定しなかった場合、ユーザは、ファイルに名前を付けて保存するよう促されます。

---

db<<Save RTF(*<path>*, *format*)

ファイルを、指定のパス（引用符付きの *path*）に指定の形式（引用符付きの *format*）で保存する。*path*引数を省略した場合、スクリプトを実行した時点で、ファイルに名前を付けて保存するよう促されます。

---

db<<Save Text(*<path>*, *format*)

ボックスのテキストを含むファイルを、指定のパス（引用符付きの *path*）に指定の形式（引用符付きの *format*）で保存する。*path*引数を省略した場合、スクリプトを実行した時点で、ファイルに名前を付けて保存するよう促されます。

---

db<<Scroll Window(*Display Box*|*relative-vertical-pixels*|*relative-horizontal-pixels*, *relative-vertical-pixels*|{*absolute-vertical-pixels*, *absolute-horizontal-pixels*})

ディスプレイボックスが表示されているウィンドウをスクロールする。

---

db<<Select

db<<Deselect

ディスプレイボックスを選択（強調表示）、または選択解除する。

---

db<<Set Menu Item State(*index*, 0|1|-1)

*index*で指定されたポップアップメニュー項目の状態を設定する。通常（0）、選択されている（1）、選択不可（-1）のいずれか。

---

db<<Set Page Setup<Margins(*left*, *right*, *top*, *bottom*)>, <Scale(*s*)>, <Portrait(*Boolean*)>, <Paper Size(*paper size*)>)

db<<Set Page Setup<Margins({*left*, *right*, *top*, *bottom*})>, <Scale(*s*)>, <Portrait(*Boolean*)>, <Paper Size(*paper size*)>)

ページ設定を行う。margins（余白）はセンチ単位で指定します。scale変数*s*は、10（1000%）から0.2（20%）の範囲の数値で、デフォルト値は1（100%）です。portraitが真（1）の場合、ページは縦向き、それ以外の場合は横向きです。paper sizeには、用紙サイズ（"Letter"、"Legal"など）を指定します。

例

下の例では、新しいウィンドウを作成し、ページ設定の各項目を設定します。

```
w = New Window( "Window",
   Text Box( "Page Setup Test" )
);
w << Set page setup(
   margins( 1, 1, 1, 1 ),
   scale( 1 ),
   portrait( 1 ),
   paper size( "Letter" )
);
```

---

db<<Set Print Headers(*left header*, *center header*, *right header*)

説明

印刷する際にヘッダの左・中央・右に記す情報を指定する。

例

```
w = New Window( "ウィンドウ", Text Box( "ヘッダの例" ) );
w << Set Print Headers(
   "日付: &d;", // 左
   "&wt;", // 中央
   "ページ &pn; / &pc;" // 右
);
w << Print Window;
```

---

db<<Set Print Footers(*left footer*, *center footer*, *right footer*)

説明

印刷する際にフッタの左・中央・右に記す情報を指定する。

例

```
w = New Window( "ウィンドウ", Text Box( "フッタの例" ) );
w << Set Print Footers(
   "日付: &d;", // 左
   "&wt;", // 中央
   "ページ &pn; / &pc;" // 右
);
w << Print Window;
```

---

db<<Set Stretch(x,y)

説明

ディスプレイボックスの縦方向と横方向の伸縮動作を設定する。

引数

"Window" ボックスは、ウィンドウのサイズに応じて変わる。サイズはボックスの最小値と最大値のプロパティによって決まります。

"Fill" ボックスは、コンテナいっぱいのサイズになる。

"Off" ボックスは伸縮しない。

"Neutral" 多くのボックスは最初ニュートラルな状態にある。最下位のボックス（子を持たないボックス）がニュートラルな場合は、"Off" となります。ほとんどのコンテナボックスでは、子ボックスの伸縮動作（および最小値と最大値）によって動作が決まります。

db<<Set Submenu (*index*, *submenu count*)

説明

indexで指定した番号のメニュー項目に対して、指定した数のサブメニュー項目を設定する。

例

下の例は、"A"、"B"、"C" というメニュー項目があるメニューを作成します。"A" にはサブメニュー項目 "A1" と "A2" を、"B" にはサブメニュー項目 "B1"、"B2"、"B3" を設定しています。

```
New Window( "title", ob = Outline Box( "title" ) );
ob << Set Menu Script(
   {"A", "", "A1", Print( "A1" ), "A2", Print( "A2" ),
   "B", "", "B1", Print( "B1" ), "B2", Print( "B2" ), "B3", Print( "B3" ),
   "C", Print( "C" )}
);
ob << Set Submenu(1, 2); // メニューA、サブメニューにA1とA2の2項目
ob << Set Submenu(4, 3); // メニューB、サブメニューにB1、B2、B3の3項目
```

db<<Set Report Title(*title*)

レポートの新しいタイトルを設定する。*title* は引用符付きです。

Show Properties(*db*)

与えられたディスプレイボックスで利用できるメッセージを表示する。

db<<Sib

ディスプレイボックスの兄弟（同レベルのもの）を戻す。

db<<Sib Append(*db2*)

ディスプレイボックス *db* と同レベルにディスプレイボックス *db2* を追加する。引数は、評価結果がディスプレイボックスへの参照である必要があります。

db<<Size Window(*x*, *y*)

ディスプレイボックスが表示されているウィンドウのサイズを変更する。

db<<Update Window

ディスプレイボックスを（オペレーティングシステムによっては、他のウィンドウも同時に）保持するウィンドウに無効化された領域がある場合に、そのウィンドウを更新する。無効化されたボックス領域には、新しいコンテンツが再描画されます。

メモ

スライダと JSL コールバックを組み合わせた一部のインタラクティブな JSL スクリプトでは、<<Update Window を使用して、ディスプレイの各部をスライダと同期させる必要があります。

db<<Zoom Window

内容がすべて表示されるようにウィンドウのサイズを拡大する。

Axis Boxのメッセージ

Axis Box<<Axis Settings(*<named arguments>*)

「軸の指定」ウィンドウを開くか、または、目盛りや軸ラベルなどの軸の設定を指定する。

引数が指定されていない場合は、「軸の指定」ウィンドウを表示します。

そうでない場合は、各軸に名前付き引数を指定します。

- Y軸はAxis Box(1)として指定します。
- X軸はAxis Box(2)として指定します。

オプションの名前付き引数

すべての軸

Scale("Linear"|"Log"|"Power"|"Geodesic"|"Geodesic US"|"Custom Scale"|"Normal Probability|Weibull Probability|Frechet Probability|Logistic Probability|Exponential Probability|Gamma Probability|Beta Probability|Mixture of 2 Normals Probabilities|Mixture of 3 Normals Probabilities) 軸のスケールを指定する。タイプ（type）にCustom Scaleを指定する場合、Scale to Internal(*expr*)とScale to External(*expr*)の2つの名前付き引数が必要になります。

Min(*n*) 軸に表示される最小値を変更する。

Max(*n*) 軸に表示される最大値を変更する。

Reverse Order(*Boolean*) 軸の最小値と最大値を逆にする。

Inc(*n*) 指定の目盛り間隔で数値を表示する。

Set Font(*font*) 数値に指定のフォント（引用符付きの *font*）を適用する。デフォルトのフォントはJMPの［フォント］の環境設定で決まります。

Set Font Size(*points*) 数値に指定のフォントサイズを適用する。デフォルトのフォントはJMPの［フォント］の環境設定で決まります。

Set Font Style("Strikeout"|"Underline") 数値に指定のスタイル（引用符付き）を適用する。

Automatic Font Size(*Boolean*) デフォルトのサイズではラベルが軸にすべて収まらない場合に、ラベルが自動的に（特定の最小サイズまで）縮小される。0の場合、フォントサイズは縮小されません。

Automatic Tick Marks(*Boolean*) 目盛りがオフになっている場合でも、（スペースが足りないために）一部のラベルが表示されない場合に、目盛りがオンになる。

Label Orientation("Automatic"|"Horizontal"|"Vertical"|"Perpendicular"|"Parallel"|"Angled") 軸ラベルの向きを指定する。デフォルトの値は、"Automatic"で、ラベルの幅に応じて向きが決まります。

Lower Frame(*Boolean*) ラベルの下にフレームを表示する。デフォルト値はオフです。

Value Labels データ値に代えて指定のラベルを表示する。

Inside Ticks(*Boolean*) 目盛りを軸の内側または外側に表示する。

Add Ref Line({Label Row Nesting(*n*), *begin range*, <*end range*>, <"Solid"|"Dotted"|"Dashed"|"DashDot"|"DashDotDot">, <*color*>, <*label*>, <width(*n*)>, <*opacity*(%)>}) 参照線の範囲、線のパターン、色、ラベル、幅、透明度を定義する。デフォルトの設定は、色が黒、幅が1ピクセルの実線です。Label Row Nesting(*n*)は、軸のラベルの階層数を指定します。*color*引数と*label*引数は引用符付きです。

カテゴリカル軸

Wrap Lines(*n*) 複数行（*n*）にまたがる長いラベルを折り返す

数値軸

Format(*arguments*) 数値軸のデータの形式を指定する。引数については、数値軸の列プロパティにある「形式」リストを参照してください。日付時間形式を指定する場合は、次に挙げるInterval引数のいずれかも指定してください: "Numeric"、"Year"、"Quarter"、"Month"、"Week"、"Day"、"Hour"、"Minute"、または"Second"。

Minor Ticks(*number*) 目盛りの間に刻む補助目盛りの数（*number*）を指定する。

Tick Offset(*number*) 目盛りの開始点を指定する。

Major Ticks(*Boolean*) 各数値の間に目盛りを表示する、または削除する。

Minor Ticks(*Boolean*) 各数値の間に補助目盛りを表示する、または削除する。

Show Major Grid(*Boolean*) 目盛りの位置にグリッド線を表示する、または削除する。

Show Minor Grid(*Boolean*) 補助目盛りの位置にグリッド線を表示する、または削除する。

Major Grid Line Color(*color*) 目盛りを指定の色（引用符付きの *color*）で表示する。

Minor Grid Line Color(*color*) 補助目盛に引かれたグリッド線の色を指定する。

例

次の例は、二変量に対する散布図を作成し、X 軸と Y 軸の基本設定を定義します。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
biv = dt << Bivariate( X( :"身長(インチ)"n ), Y( :"体重(ポンド)"n ), FitLine );
rbiv = biv << Report;
xaxis = rbiv[Axis Box( 2 )];
yaxis = rbiv[Axis Box( 1 )];
xaxis << Axis Settings( Show Major Grid( 1 ) );
yaxis << Axis Settings( Decimal( 10, 3 ) );
```

Axis Box<<Add Axis Label(*quoted string*)

軸に、指定の文字列（引用符付きの *string*）を表示したラベルをつける。

Axis Box<<Add Ref Line(*number*, *linestyle*, *<color>*, *<label>*, *<width>*)

指定の数値（*number*）の位置に、指定の線種（*linestyle*）("Solid"|"Dashed"|"Double")、色（引用符付きの *color*）、ラベル（引用符付きの *label*）、幅（*width*）（単位はピクセル）で参照線を引く。

Axis Box<<Decimal(*width*, *decimal places*)

軸の数値の表示形式を変更する。

Axis Box<<Format(*name*)

指定の名前（引用符付きの *name*）で与えられた表示形式に変更する。

Axis Box<<Get Inc(*n*)

軸の目盛りの間隔を取得する。

Axis Box<<Inc(*n*)

目盛りの間隔を設定する。

Axis Box<<Interval(*format*)

Inc()（目盛り間隔）で使われる値の単位を、日付／時間データの形式で指定する。"Numeric"、"Year"、"Quarter"、"Month"、"Week"、"Day"、"Hour"、"Minute"、または "Second"。

Axis Box<<Label Orientation(*format*)

軸のラベルを、次のいずれかの向きに回転させる。"Automatic"（ラベルの幅に基づいたデフォルト設定）、"Horizonal"（横）、"Vertical"（縦）、"Perpendicular"（垂直）、"Parallel"（平行）、"Angled"（角度）。

Axis Box<<Major Grid Line Color(*color*)

目盛りを指定の色（引用符付きの *color*）で表示する。

Axis Box<<Max(*maximum*)

軸に表示される最大値を変更する。

Axis Box<<Minor Grid Line Color(*color*)

補助目盛りを指定の色（引用符付きの *color*）で表示する。

Axis Box<<Min(*minimum*)

軸に表示される最小値を変更する。

Axis Box<<Minor Ticks(*number*)

目盛りの間に刻む補助目盛りの数（*number*）を指定する。

Axis Box<<Remove Axis Label

Add Axis Label でつけられたラベルをすべて削除する。

Axis Box<<Reverse Scale(*Boolean*)

通常のスケールの方向を逆にして、最大値が左または下に来るようにする。

Axis Box<<Revert Axis

軸を元の設定（作成時の設定）に戻す。

Axis Box<<Scale(*type*)

軸のスケールを指定のタイプ（*type*）に変更する。選択肢は、"Linear"|"Log"|"Exp Prob"|"Weibull Prob"|"Logistic Prob"|"Frechet Prob"|"Normal"|"Cube Root"|"Johnson Su Scale"|"Geodesic"|"Geodesic US"|"Custom Scale"|"Power"|"Gamma Prob"|"Beta Prob"|"Mixture of 2 Normals Prob"|"Mixture of 3 Normals Prob"。

タイプ（type）に Custom Scale を指定する場合、Scale to Internal(expr) と Scale to External(expr) の 2 つの名前付き引数が必要になります。

Axis Box<<Tick Font(*name*, *<size>*, *<style|style style...>*, *<angle>*)

目盛りのフォント（引用符付きの *name*）、サイズ（*size*）、プロパティ（引用符付き）を設定する。複数のスタイルを指定するには、各スタイルの間にスペースを挿入し、スタイルを引用符付きで指定します。

Axis Box<<Show Labels(*Boolean*)

軸上の値のラベルを表示する、または表示しない。

Axis Box<<Show Major Grid(*Boolean*)

目盛りに合わせてグリッド線を引く、または削除する。

Axis Box<<Show Major Ticks(*Boolean*)

目盛りを表示する、または削除する。

Axis Box<<Show Minor Grid(*Boolean*)

補助目盛りに合わせてグリッド線を引く、または削除する。

Axis Box<<Show Minor Ticks(*Boolean*)

補助目盛りを表示する、または削除する。

Axis Box<<Tick Label List(*<i>*, *{text1,text2, ...}*,*<{n1, n2, ...}>*)

軸の目盛りラベルの値と位置を設定する。

メモ: 目盛りの位置を指定しないと、間隔は、自動的に 1.0 に設定されます。

必須の引数

{text1,text2, ...} ラベルのタイトルを引用符付き文字列で指定する。

オプションの引数

i ラベル行のインデックスを指定する。指定しない場合は、既存のラベル行がクリアされ、指定のとおりに新しく作成されます。指定すると、特定のラベル行が上書きされます。現在のラベル行数より大きいインデックスを使用すると、末尾に新しいラベル行が追加されます。

{n1, n2, ...} 各ラベルに対応する値を指定する。値のリストを指定しなかった場合、ラベルは、1 を始点にして整数の上に配置されます。

Border Boxのメッセージ

この節では、Border Box（境界ボックス）に使用できるJSLメッセージをご紹介します。

メモ: 境界ボックスで引数に指定できるディスプレイボックスの個数は、1つだけです。

Border Box<<Set Background Color(*{r, g, b}*|<*color*>)

境界ボックスの背景色を設定する。RGB値、または色（*color*）のオプションのリストを引用符付きで指定する。

例:

```
border box<<Set Background Color("red");
```

または

```
border box<<Set Background Color( {255, 192, 3} );
```

Border Box<<Set Color(*{r, g, b}*|<*color*>)

境界ボックスの境界線の色を設定する。RGB値のリスト、または引用符付きの色（*color*）を指定してください。

例:

```
border box<<Set Color("red");
```

Border Box<<Get Color

境界ボックスの境界線の色を取得する。

Border Box<<Set Style(*style*)

境界ボックスの境界線の線種を設定する。線種（*style*）は、次の番号またはキーワードを使って指定してください。0／"Solid"（実線）、1／"Dotted"（点線）、2／"Dashed"（破線）、3／"DashDot"（一点鎖線）、または4／"DashDotDot"（二点鎖線）。

例:

```
border box<<Set Style("Dotted");
```

Border Box<<Get Style

境界ボックスの境界線の線種を取得する。

Data Browser Boxのメッセージ

dbb<<Set Data Table(*<data table>*)

データブラウザボックスのデータテーブルを設定する。

Data Filter Source Boxのメッセージ

dfsb<<Set Row States(*dt*, *rs*)

フィルタの中で指定されたデータテーブルに行の属性を設定する。行の属性での選択はデータテーブルにはリンクしませんが、選択フィルタにリンクしているレポートには反映されます。

次も参照

Frame Boxのメッセージ

Frame Box<<Add Graphics Script(*<order>*,*<description>*, *<script>*)

説明

フレームボックス内にグラフィックを描くためのスクリプトを追加する。

オプションの引数

order グラフィック要素を描く順序を指定する。指定できる値は、"Back"または"Forward"、あるいは複数のグラフィック要素の描画順を示す整数です。1は、そのオブジェクトを最初に描くことを示します。

description「グラフをカスタマイズ」ウィンドウのグラフィックスクリプトの横に表示される引用符付き文字列。*description*引数は引用符付きです。

script JSLスクリプト。

例

次の例では、グラフィックスクリプトはまず線を描き、次に二変量の散布図に必要なその他のグラフィック要素（グリッド線、参照線、マーカー）を描きます。1の順序引数を指定しなかった場合は、線が最後に描かれてマーカーを隠してしまいます。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
obj = dt << Bivariate( Y( :"体重(ポンド)"n ), X( :"身長(インチ)"n ) );
Report( obj )[FrameBox( 1 )] <<
Add Graphics Script(
  1, // 最初に線を描く
  Description( "Pen Script" ),
  Pen Color( "red" );
  Pen Size( 5 );
  Y Function( 60 + 120 / 2 * (1 + Sine( (2 * Pi() * (x - 50)) / 22.5 )), x );
```

```
);
```

Frame Box<<Append Seg

特定のフレームボックスにディスプレイセグメントを追加する。

Frame Box<<Background Color({*RGB values*}|<*color*>)

背景の色を変更する。RGB 値のリスト、または引用符付きの色（*color*）を指定してください。

Frame Box<<Child Seg

フレームボックスのディスプレイセグメントの子を戻す。

Frame Box<<Edit Graphics Script

現在のグラフィックスクリプトの編集や削除を行うためのダイアログボックスを表示する。

Frame Box<<Find Seg

指定の引数を持つディスプレイセグメントを戻す（たとえば、セグメントの名前）。

Frame Box<<Frame Size(*x, y*)

ピクセル値をもとに、フレームのサイズを変更する。

Frame Box<<Make Table of Graphs Like This

グラフを含むデータテーブルを作成する。

Frame Box<<Marker Size(*size*)

マーカーのサイズを変更する。値は、0（ドット）、1（小）、2（中）・・・です。

Frame Box<<Row Colors(*color*)

Frame Box<<Row Markers(*marker*)

Frame Box<<Row Exclude(*Boolean*)

Frame Box<<Row Hide(*Boolean*)

Frame Box<<Row Label(*Boolean*)

レポートに関連するデータテーブルにコマンドを送る。これらのメッセージにより、選択された行の属性が変更されます。Row Exclude（行の除外）、Row Hide（行を表示しない）、Row Label（行ラベル）を引数なしで使用すると、そのオプションが切り替わります。オプションがオフなら、メッセージがオンになります。オプションがオンなら、メッセージがオフになります。

frame box<<Set Background Fill(*Boolean*)

グラフの背景を背景色で塗りつぶす機能を有効または無効にする。透明な背景でグラフを貼り付けたい場合に使用できます。

例

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
biv = Bivariate( y( :"体重(ポンド)"n ), x( :"身長(インチ)"n ) );
rbiv = biv << Report;
framebox = rbiv[Frame Box( 1 )];
// 背景色を設定する
framebox << Background Color( "red" );
// デモ用: 色の変化を観察する
Wait( 1 );
// 背景色の塗りつぶしをオフにする
framebox << Set Background Fill( 0 );
```

frame box<<X Axis(<Min(*minimum*)>, <Max(*maximum*)>, <Inc(*n*)>, *<named arguments>*)

X 軸のスケールを設定する。

frame box<<Y Axis(*<Min(min)>*, *<Max(max)>*, <Inc(*n*)>, *<named arguments>*)

Y 軸のスケールを設定する。

Graph 3D Box のメッセージ

Graph 3D Box()

表示コマンドを 3D プロットに送信する。

Excerpt Box のメッセージ

Excerpt Box(*rptnum*, *lstSubscripts*)

レポートから一部分を抜粋し、その抜粋したディスプレイボックスを戻す。レポートの番号を rptnum で指定し、ディスプレイのリストを lstSubscripts で指定します。subscripts に指定する番号は、抜粋したディスプレイを除いた後の番号。

Filter Col Selectorのメッセージ

Filter Col Selector(<Data Table*(name)*>, <width*(pixels)*>, <nLines*(n)*>, *<script*>, <onChange*(expr)*>)

列名のリストを含むディスプレイボックスを戻す。列のフィルタリングを設定できます。

Global Boxのメッセージ

Global Box(*value*)

グローバル変数の値を表示するディスプレイボックスを作成する。

Hier Boxのメッセージ

Hier Box(*title*, Hier Box(...), Hier Box(...), ...)

引用符付き文字列の階層を含んだディスプレイボックスに指定のタイトル（引用符付きの *title*）を付けて戻す。

Matrix Boxのメッセージ

Matrix Box<<Get

行列の要素を戻す。

Matrix Box<<Make Into Data Table(<Invisible(*Boolean*)|Private(*Boolean*)>)

説明

行列から新しいデータテーブルを作成する。Invisible(1) は、データテーブルを非表示にします。非表示のデータテーブルでも、「JMP ホームウィンドウ」や［ウィンドウ］メニューから開くことができます。Private(1) は、データテーブルウィンドウには表示せずにデータテーブルを開きます。プライベートのデータテーブルは、一般的な使用の対象としないようスクリプトのみで使用する場合に適しています。

戻り値

新しいデータテーブルへの参照

Matrix Box<<Set Format(*<width>*, *<decimal places>*, <"Use Thousands Separator">)

説明

行列の要素の表示形式を指定する。

引数

Matrix box では、桁数以外にも、多数の表示形式を指定することができます。構文の詳細については、「Number Col Box<<Set Format(<width>|<width, decimal places>, <"Use Thousands Separator">)」を参照してください。

Matrix Box<<Sort(*column number*, *ascending*)

column_num で指定された列番号に基づいて行列の行を並べ替える。デフォルトの順序は昇順です。

column number が 0 の場合、並べ替えは削除されます。

ascending はブール値です。*ascending* が "True" の場合は昇順、"False" の場合は降順に並べ替えられます。

Nom Axis Box のメッセージ

Nom Axis Box<<Divider Lines(*Boolean*)

軸ボックスのラベルを区切る分割線を引く、または消す。

Nom Axis Box<<Lower Frame(*Boolean*)

軸周辺の下のフレームを追加または削除する。

Nom Axis Box<<Rotated Tick Labels(*Boolean*)

各目盛りの値のラベルを回転させる、または回転させない。

Number Col Box のメッセージ

Number Col Box<<Add Element(*item*)

項目（*item*）を数値列ボックスに追加する。項目（*item*）は 1 つの数字、数字のリスト、または数字の行列です。

Number Col Box<<Bootstrap(*nsample*, Random Seed(*number*), Fractional Weights(*Boolean*), Split Selected Column(*Boolean*), Discard Stacked Table if Split Works(*Boolean*)

説明

異なる抽出の重みを使って何度も分析を繰り返し、選択されたテーブルを収集して分析をブートストラップする。

引数

nsample データからの無作為抽出と、そこからの統計量の計算といった一連の処理を反復する回数を設定します。回数を増やせば、それだけ統計量の状態をより正確に推定することができます。デフォルトでは、2,500に設定されています。

Random Seed(*number*) 後でブートストラップ分析を再実行するときに、同じ結果を再現したい場合は、同じ乱数シード値を入力してください。デフォルトでは、乱数シード値は設定されていません。

Fractional Weights(*Boolean*) ベイズ流のブートストラップ分析を実行します。ブートストラップを実行するたびに、各データ行に対する重みが計算されます（重みの計算方法については、『基本的な統計分析』を参照してください）。この重みを使って、目的の統計量が計算されます。デフォルトでは、このオプションは選択されておらず、通常の単純なブートストラップが実行されます。

Split Selected Column(*Boolean*) ブートストラップで得られた統計量を、結果のデータテーブルにおいて列ごとに分割します。このデータテーブルでは、各行が、ブートストラップ標本1組に対応しています（ただし、最初の1行目は、観測されたデータに対するものです）。

このオプションをオフにすると、積み重ねた形式のデータテーブルだけが作成されます。この形式では、ブートストラップにおける抽出ごとに、元の表全体が縦に積み重ねられていきます。この形式では、元の表における各行が、結果のデータテーブルにおける複数の各行に対応しています。そして、元の表における各列が、結果のデータテーブルにおける各列と対応しています。

Discard Stacked Table if Split Works(*Boolean*)（Split Selected Columnオプションを含めた場合のみ選択可能）このオプションによって、作成されるデータテーブルの数が決まります。Discard Stacked Table if Split Worksオプションを選択しなかった場合は、以下の2つのデータテーブルが作成されます。1つは、積み重ねた形式のデータテーブルです。このデータテーブルは、元の表を積み重ねた形式で、表のすべての列に対する結果を含んでいます。もう1つは、分割した形式のデータテーブルです。このデータテーブルは、積み重ねた形式のデータテーブルを分割したもので、元の表で選択した列に対する結果だけを含んでいます。

Number Col Box<<Get

Number Col Box<<Get(*i*)

値すべてをリストで取得する。または、*i* 番目の値を取得する。

Number Col Box<<Get As Matrix

値を行列（列ベクトル）の形で取得する。

Number Col Box<<Get Format

現在の形式を戻す。

Number Col Box<<Get Heading

列見出しのテキストを戻す。

Number Col Box<<Remove Element(*row number*)

列から、指定した位置にある要素を削除する。

Number Col Box<<Set Format(*<width>*|*<width, decimal places>*, <"Use Thousands Separator">)

Number Col Box<<Set Format("Best", *<width>*, <"Use Thousands Separator">)

Number Col Box<<Set Format(("Fixed Dec"|"Percent"), *<width>*|*<width, decimal places>*, <"Use Thousands Separator">)

Number Col Box<<Set Format("Pvalue", *<width>*)

Number Col Box<<Set Format(("Scientific"|"Engineering"|"Engineering SI"), *<width>*|*<width, decimal places>*)

Number Col Box<<Set Format("Precision", *<width>*|*<width, decimal places>*, <"Use Thousands Separator">, <"Keep Trailing Zeroes">, <"Keep All Whole Digits">)

Number Col Box<<Set Format("Currency", *<currency code>*, *<width>*|*<width, decimal places>*, <"Use Thousands Separator">)

Number Col Box<<Set Format(*datetime*, *<width>*, *<input format>*)

Number Col Box<<Set Format(("Latitude DDD"|"Latitude DDM"|"Latitude DMS"|"Longitude DDD"|"Longitude DDM"|"Longitude DDM"), *<width>*|*<width, decimal places>*, ("PUN"|"DIR"|"PUNDIR"))

Number Col Box<<Set Format("Custom", Formula(...), *<width>*, *<input format>*)

説明

列の表示形式を設定する。

引数

『JMPの使用法』に引数の説明があります。Matrix Box()、Number Col Box()、Number Col Edit Box()、Number Edit Box()は、同じSet Format構文を使用します。

例

```
<<Set Format( 10, 2, "Use Thousands Separator");
```

```
<<Set Format( "Currency", "EUR", 20, );
<<Set Format( "m/d/y", 10 );
<<Set Format( "Precision", 10, 2, "Keep Trailing Zeroes", "Keep All Whole Digits"
  );
<<Set Format( "Latitude DDD", "PUNDIR"); // "PUN"は punctuation（フィールド句読記号)、
  "DIR"はdirection（方角)、PUNDIRは両方
<<Set Format( "Custom", Formula( Abs( value ) ), 15 );
```

メモ

- 通貨コードのリストは、『スクリプトガイド』を参照してください。通貨コードを省いた場合、ロケールに基づいた通貨コードが使用されます。
- 表示形式を指定しない場合、日付時間値であれば *decimal places* に100より大きな値、*p* 値であれば97を設定します。
- 小数点以下の桁数を指定する場合には、その前に表示幅を指定する必要があります。
- オプションは、リストまたは変数で指定するか、または Function() で指定することができます。

```
ncbFunc = Function({}, {"Fixed", 12, 5});
```

number col box<<Set Heading(*quoted string*)

列見出しのテキストを変更する。

Number Col Edit Boxのメッセージ

Number Col Edit Box<<Set Format(*<width>*, *<decimal places>*, <"Use Thousands Separator">|*<other options>*)

説明

列の表示形式を設定する。

引数

Number Col Edit Boxでは、桁数以外にも多数の形式を指定することができます。

次も参照

「Number Col Box<<Set Format(<width>|<width, decimal places>, <"Use Thousands Separator">)」

Number Col Edit Box<<Remove Element(*x position*, *y position*, *i*)

列から、指定した位置にある要素を削除する。

Number Edit Box のメッセージ

Number Edit Box<<Set Format(*<width>*, *<decimal places>*, <"Use Thousands Separator">|*<other options>*)

説明

列の表示形式を設定する。

引数

Number Edit Box では、桁数以外にも多数の形式を指定することができます。

次も参照

「Number Col Box<<Set Format(<width>|<width, decimal places>, <"Use Thousands Separator">)」

Outline Box のメッセージ

Outline Box<<Close(*Boolean*)

アウトラインボックスを閉じる。

Outline Box<<Close All Below

ノードの子ノードをすべて閉じる。

Outline Box<<Close All Like This

このアウトラインボックスと同様のノードをすべて閉じる。

Outline Box<<Close Where No Outlines

子ノードを持たないノードをすべて閉じる。

Outline Box<<Get Title

アウトラインボックスのタイトルを取得する。

Outline Box<<Horizontal(*Boolean*)

ノードの子ノードを横に並べる。

Outline Box<<Open All Below

ノードの子ノードをすべて開く。

Outline Box<<Open All Like This

このアウトラインボックスと同様のノードをすべて開く。

Outline Box<<Set Menu Script(*{quoted string1, script1, quoted string2, script2, ...}*)

赤い三角ボタンをクリックしたときのメニューに項目を追加する。

Outline Box<<Set Title(*title*)

アウトラインボックスのタイトル（引用符付きの *title*）を指定する。

Panel Boxのメッセージ

Panel Box<<Get Title

パネルボックスのタイトルを取得する。

Panel Box<<Set Title(*title*)

パネルボックスのタイトル（引用符付きの *title*）を指定する。

Plot Col Boxのメッセージ

Plot Col Box<<Get As Matrix

値を行列（列ベクトル）の形で取得する。

Plot Col Box<<Get Labels

各行のラベルを取得する。

Plot Col Box<<Remove Element(*row number*)

列から、指定した位置にある要素を削除する。

Plot Col Box<<Set Labels(*{list}*)

各行のラベルを設定する。

Plot Col Box<<Set Scale(*minimum*, *maximum*, <"format">, *<width>*, *<decimal places>*, *<*"Use Thousands Separator">)

説明

横軸の最小値と最大値を指定する。

引数

表示形式の引数は、最初の 2 つの引数が 0 と 1 の場合に、使用する形式を指定するものです。設定可能な形式は多数あります。構文の詳細については、「Number Col Box<<Set Format(<width>|<width, decimal places>, <"Use Thousands Separator">)」を参照してください。

Plot Col Box<<Set Values(*[matrix]* or *{list}*)

行列（数値変数の場合）またはリスト（文字変数の場合）の値を列の値として設定する。

Slider Box ／ Range Slider Box のメッセージ

Slider Box<<Get(*<index>*)

Range Slider Box<<Get Lower(*<index>*)

Range Slider Box<<Get Upper(*<index>*)

スライダの現在の値を戻す。

Slider Box<<Get Max()

範囲スライダとスライダに使用できる最大値を戻す。

Slider Box<<Get Min()

範囲スライダとスライダに使用できる最小値を戻す。

Slider Box<<Get Var

Range Slider Box<<Get Lower Var

Range Slider Box<<Get Upper Var

スライダの値が入る変数の名前を戻す。

Slider Box<<Set(*float*, <*index*>, <Run Script(*Boolean)*>)

Range Slider Box<<Set Lower(*float*, <*index*>, <Run Script(*Boolean*)>)

Range Slider Box<<Set Upper(*float*, <*index*>, <Run Script(*Boolean*)>)

スライダの値を設定する。Run Script(*Boolean*) は、スライダが変化したときのスクリプトを、Set、Set Lower、または Set Upper メッセージの後で実行するかどうかを制御します。

Slider Box<<Set Max(*float*, <*index*>)

範囲スライダとスライダに使用できる最大値を設定する。

Slider Box<<Set Min(*float*, <*index*>)

範囲スライダとスライダに使用できる最小値を設定する。

Slider Box<<Set Script(<*script*>)

範囲スライダとスライダの更新時に実行するスクリプトを設定する。

Slider Box<<Set Var(*slider variable*)

Range Slider Box<<Set Lower Var(*slider variable*)

Range Slider Box<<Set Upper Var(*slider variable*)

スライダの値が入る変数の名前を設定する。

String Col Boxのメッセージ

String Col Box<<Add Element(*item*)

項目（item）を引用符付き文字列ボックスに追加する。項目は 1 つの引用符付き文字列、または引用符付き文字列のリストです。

String Col Box<<Get

string Col Box<<Get(*i*)

リスト内の値すべて、または *i* 番目の値を取得する。

String Col Box<<Get Heading

列見出しのテキストを戻す。

String Col Box<<Remove Element(*row number*)

列から、指定した位置にある要素を削除する。

String Col Box<<Set Allow Text Search(*Boolean*)

説明

このメッセージにより、行を選択できる Table Box において、列の値がキーボードで入力した文字で始まっている行が選択されるようになる。

例

```
// 例を実行する。
// K2 を選択する。
// g の文字をタイプすると、最後の行が選択される。
// ki の文字をタイプすると、3 番目の行が選択される。
New Window( "Mountains",
   tb = Table Box(
      sb =
      String Col Box( "Mountain",
         {"K2", "Delphi", "Kilimanjaro",
         "Grand Teton"}
      ),
      Number Col Box( "Elevation (meters)",
         {8611, 681, 5895, 4199}
      ),
      Plot Col Box( "", {8611, 681, 5895, 4199} )
   )
);
tb << Set Selectable Rows( 1 );
sb << Set Allow Text Search( 1 );
```

String Col Box<<Set Heading(*title*)

列の見出し（引用符付きの *title*）を変更する。

String Col Box<<Set Justify(*justification*)

文字列ボックス内の引用符付き文字列の位置を右揃え（"Right"）、左揃え（"Left"）、中央揃え（"Center"）のいずれかに指定する。

Tab Boxのメッセージ

Tab Box<<Get Tab Margin()

タブボックスの現在のマージン（単位はピクセル）をリストで戻す。戻り値のリストは、{左，上，右，下}の順番で、現在のマージンを含んでいます。

Tab Box<<Set Style("Tab"|"Combo" |"Outline"|"Vertical Spread"|"Horizontal Spread"|"Minimize Size")

タブボックスの表示形式をタブからコンボボックスまたはアウトラインノードに変更する。

"Vertical Spread" と "Horizontal Spread" は、タブタイトルの表示の向きを変更します。

"Minimize Size" はタブのスタイルがタイトルの幅に応じて決まります。

Tab Box<<Set Tab Margin(*n*|{...})

タブボックスのタブのマージンを設定する。数字が1つだけ指定された場合は、4つのマージンすべてをこのピクセル数に設定します。2つの数字のリストが指定された場合は、左右のマージンを最初の数字、上下のマージンを2番目の数字のピクセル数に設定します。4つの数字のリストが指定された場合は、マージンを{左，上，右，下}の順序で設定します。

Tab Box<<Show Tabs(*Boolean*)

タブボックスのタブの表示／非表示を切り替える。タブを非表示にした場合、タブを選択したり表示したりする別の方法が必要になります。たとえば、タブへの参照リストを含んだリストボックスなどです。デフォルト値は1です。

Table Boxのメッセージ

Table Box<<Bootstrap(*nsample*, Random Seed(*number*), Fractional Weights(*Boolean*), Split Selected Column(*Boolean*), Discard Stacked Table if Split Works(*Boolean*)

説明

異なる抽出の重みを使って何度も分析を繰り返し、選択されたテーブルを収集して分析をブートストラップする。

次も参照

「Number Col Box のメッセージ」

Table Box<<Get

表に含まれている値をリストの形で取得する。

Table Box<<Get As Matrix(<"Visible">)

表に含まれている数値を行列の形で取得する。"Visible" は、表示されている列のみを含めます。

Table Box<<Get Click Sort

列見出しをクリックするとテーブルが並べ替えられる場合は 1、そうでない場合は 0 を戻す。

Table Box<<Get Locked Columns

手のひらカーソルでドラッグできない列の数を戻す。それらの列の前に列をドロップすることはできない。

Table Box<<Get Row Change Function

行が選択されたときに評価する式を戻す。

Table Box<<Get Selectable Rows

テーブルボックスの行が現在、選択可能な場合は真（1）を戻す。

Table Box<<Get Selected Row Color

テーブルボックス内の選択された行の背景色のインデックス番号を戻す。

Table Box<<Make Combined Data Table

データテーブルへの参照を戻す。Make Data Table と同じ。ただし、同じ列を持つレポートテーブルを検索し、これらをすべて組み合わせて新しいデータテーブルを作成します。

Table Box<<Make Data Table(*name*)

データテーブルへの参照を戻す。テーブルの内容を新しいデータテーブルに出力し、そのデータテーブルに引用符付きの *name* 引数で指定された名前を付ける。

Table Box<<Reorder Columns(*from column index*, *to column index*)

from column index で指定された列が、*to column index* で指定された位置にくるよう列の順序を変更する。インデックス（*index*）は、0 を起点とします。最初の列は「0」、2 番目の列は「1」と指定してください。

Table Box<<Set Cell Changed Function(Function({*this, col box, row*}, *<script>*))

説明

テーブル内の列のセルをユーザーが編集したときに必ず呼び出される関数を設定する。

例

この例では、変更されたセルの変更後の値をログに出力します。

```
New Window( "Mountains",
   tb = Table Box(
      String Col Edit Box(
         "Mountain",
         {"K2", "Delphi", "Kilimanjaro",
         "Grand Teton"}
      ),
      Number Col Edit Box(
         "Elevation (meters)",
         {8611, 681, 5895, 4199}
      ),
      Plot Col Box( "", {8611, 681, 5895, 4199} )
   )
);
tb <<
Set Cell Changed Function(
   Function( {this, col, row},
      Print(
         (col << Get Heading) || ": row:" ||
         Char( 3 ) || " is now " ||
         Char( col << Get( row ) )
      )
   )
);
```

Table Box<<Set Click Sort(*Boolean*)

列見出しをクリックすることでテーブルが並べ替えられるようにするかどうかを指定する。

Table Box<<Set Column Borders(*Boolean*)

列の境界線を表示する。

Table Box<<Set Heading Column Borders(*Boolean*)

列見出しの境界線を表示する。

Table Box<<Set Locked Columns(*n*)

最初の*n*列をロックする。ロックされた列は、手のひらカーソルでドラッグしたり、前に列をドロップしたりできません。

Table Box<<Set Row Borders(*Boolean*)

行の境界線を表示する。

Table Box<<Set Row Change Function(*function*)

行が選択されたときに評価する式を設定する。

Table Box<<Set Selectable Rows(*Boolean*)

テーブルボックスの行を選択可能または選択不可能にする。

Table Box<<Set Selected Row Color(*color*)

テーブルボックスの行が選択可能な場合（Set Selectable Rows(True)）、選択された行の背景色（引用符付きの *color* 引数）を設定する。

Table Box<<Set Shade Cells(*Boolean*)

テーブル内のすべてのセルの背景を網掛けする。

Table Box<<Set Shade Alternate Rows(*Boolean*)

テーブル内の行を交互に濃淡表示する。

Table Box<<Set Shade Heading(*Boolean*)

列見出しの背景を網掛けする。

Table Box<<Set Underline Headings(*Boolean*)

列見出しの下に線を表示する。

Table Box<<Sort By Column(*<column number|column title>*, <Ascending(*Boolean*)>

すべての行を、列番号、または列見出し（引用符付きの *column title*）で指定された列の値に基づいて並べ替える。デフォルトの順序は、降順です。

Text Box のメッセージ

Text Box<<Font Color(*n*)

Text の文字列の色を指定する。

Text Box<<Get Hidden State

テキストボックスの現在の状態を戻す。

Text Box<<Get Text

ボックス内の文字列を戻す。

Text Box<<Get Tip

テキストボックス（またはテキスト編集ボックス）のツールヒントを戻す。

Text Box<<Markup

指定した HTML タグによりフォーマットされたテキストを戻す。HTML は完全な形でなければなりません。入れ子状のタグは適切に閉じてください。

次の例では、太字、下線、斜体のテキストが戻されます。

```
w = New Window( "書式付きテキスト",
   Text Box( "これは <b> 太字 </b> のテキストです。これは <b><i> 太字斜体 </i></b> のテキストです。これは <u> 下線付き </u> のテキストです。",
   <<Markup) );
```

Text Box<<Rotate Text(*direction*)

テキストを左（"Left"）または右（"Right"）に 90 度回転するか、水平に戻す。

Text Box<<Set Font(*name*, *<size>*, *<style|style style...>*, *<angle>*)

引用符付きの *name* 引数で指定されたフォントと、テキストの引用符付き文字列のプロパティを設定する。複数のスタイルを指定するには、各 *style* の間にスペースを挿入し、スタイルを引用符付きで指定します。

Text Box<<Set Font Size(*n*)

引用符付きテキストのサイズをポイント数で設定する。

Text Box<<Set Script(*script*)

スクリプトをテキストボックスに関連付ける。ユーザがEnterキーを押すと（または、テキスト編集ボックスがフォーカスを失ったときに）、スクリプトが実行されます。

Text Box<<Set Text(*quoted string*)

引用符付きの *string* 引数で指定された文字列をボックス内のテキストとする。

Text Box<<Set Tip(*quoted string*)

引用符付きの *string* 引数で指定された文字列をテキストボックス（またはテキスト編集ボックス）のツールヒントとする。

Text Box<<Set Wrap(*n*)

改行ポイントをピクセル（*n*）で設定する。

Tree NodeとTree Boxのメッセージ

以下のJSLメッセージで、nodeはツリーノードまたはツリーノードへの参照、rootはツリーボックスまたはツリーボックスへの参照を表します。

注意： macOSの場合、子ノードを持つルートノードにCollapseメッセージを定義するSet Node Select Scriptを送ると、スクリプトが2回実行されます。Windowsの場合、スクリプトは実行されません。macOSで2回実行されるのは増分だけではありません。すべてのスクリプトが2回実行されます。ログへの出力、列の作成、何らかの削除など、すべての処理が2回行われます。

node<<Append(*<node>*)

このノードの子の後ろにツリーノードを挿入する。

node<<Collapse

ノードを閉じる。ノードの親が折りたたまれている場合、この動作が実行されない場合もあります。

node<<Expand

ノードを開く。ノードの親が折りたたまれている場合、この動作が実行されない場合もあります。

node<<Get Dimmed(*<node>*)

ノードのテキストを暗くする（透明度を低くする）オプションを取得する。

node<<Get Font Style(*<node>*)

ノードのフォントスタイルを取得する。

node<<Get Tip

ツリーノードのツールヒントを戻す。

node<<Prepend(*<node>*)

このノードの子の前に、ツリーノードを挿入する。

node<<Remove

ツリーからノードとそのすべての子を削除する。

node<<Set Dimmed(*Boolean*)

ノードのテキストを暗くする（透明度を低くする）オプションを設定する。

node<<Set Font Style("Plain"|"Bold")

ノードのフォントスタイルを指定する。

node<<Set Selected(*<node>*)

指定したノードを選択する。ノードの親が折りたたまれている場合、この動作が実行されない場合もあります。

node<<Set Tip(*tooltip*)

ツリーノードのツールヒントを設定する。*tooltip*引数は引用符付きです。

root<<Collapse(*<node>*)

ツリー内のノードを折りたたむ。

root<<Expand(*<node>*)

ツリー内のノードを展開する。

root<<Get Selected(*<node>*)

現在選択されているツリーノードを取得する。

- 項目が1つのツリーでは、現在選択されているツリーノードまたはEmptyが戻されます。
- MultiSelect引数を含むTree Box()の結果は、表3.1に示すとおりです。

表3.1 複数選択のツリーの結果

ツリー内の選択項目数	結果
選択項目なし	empty
1つの項目を選択	1つのツリーノードのリスト
複数の項目を選択	選択された複数のツリーノードのリスト

`root<<Is Multiselect`

複数選択ツリーの場合は1、単一選択のツリーの場合は0を戻す。

`root<<Set Selected(`*node*`|`*{nodes}*`, <`*Boolean*`>)`

指定されたツリーノードを、ツリーディスプレイボックス内で選択する。ツリーノードのリストが指定された場合、複数選択のツリーではリスト内のすべてのノードが選択されます。そうでない場合は、リストの最初のノードが選択されます。ノードの選択、または非選択は、ブール引数で指定します。デフォルト値は1で、ノードが選択されます。

メモ

- Windowsの場合、`Set Selected`メッセージは、選択されたノードとツリーのルートとの間のすべてのノードを展開し、ツリーの奥深くで選択されている項目をすべて表示します。この展開の状態は、すでに選択されていたノードには影響しません。
- macOSの場合、`Set Selected`メッセージはツリーの展開状態に変化を与えません。

三角分割のメッセージ

以下のJSLメッセージで、`tri`は三角分割または三角要素座標への参照を表します。

`tri<<Get N Points`

三角分割において、一意な点の個数を戻す。

`tri<<Get Points`

三角分割において、一意な点の座標を戻す。

`tri<<Get Y`

三角分割において、一意な各座標に対するY値を戻す。

tri<<Get N Hull Points

三角分割において、凸包を構成する点の個数を戻す。

tri<<Get Hull Points

三角分割の境界の点のインデックスを戻す。

tri<<Get N Hull Edges

三角分割において、凸包を構成する辺の数を戻す。

tri<<Get Hull Edges

三角分割において、凸包を構成する辺の通し番号を戻す。

tri<<Get Hull Path

三角分割において、凸包の情報をパス形式で戻す。

tri<<Get N Triangles

三角形の個数を戻す。

tri<<Get Triangles

三角分割において、各三角形の情報を Nx3 の行列で戻す。

tri<<Get N Edges

三角分割の辺の数を戻す。

tri<<Get Edges

三角分割において、辺のインデックスを Nx2 の行列で戻す。

tri<<Subset(*{indices}*)

指定された点の部分集合に対して、その三角分割を戻す。

tri<<Peel

現在の三角分割から、凸包を構成する辺（一番外側の辺）を削除する。

Windowのメッセージ

window<<Bring Window to Front

ウィンドウを最前面に移動する。

window<<Close Window(*<nosave>*)

ウィンドウを閉じる。オプションの引数*nosave*を指定した場合は、保存したり、確認のメッセージが表示されたりしないまま、ウィンドウ（ジャーナルやレポートなど）が閉じます。

window<<Get Content Size

ウィンドウの内容のサイズを戻す。

window<<Get Window Icon

ウィンドウのアイコンの名前を戻す。

window<<Get Window Position

ウィンドウの位置を戻す。

window<<Get Window Size

ウィンドウのサイズを戻す。

window<<Get Window Title

ウィンドウのタイトルを戻す。

window<<Inval

ディスプレイボックスを無効にする。ウィンドウは、<<Update Windowメッセージが送られた際、または、次回、オペレーティングシステムによってウィンドウが更新される際に更新されます。他の方法については、「window<<Reshow」を参照してください。

window<<Maximize Display

ウィンドウを最大化する。このメッセージは廃止されます。

window<<Maximize Window(*Boolean*)

ウィンドウを最大化する。このメッセージは廃止されます。

window<<Minimize Window(*Boolean*)

ウィンドウを最小化する。

window<<Move Window(*x*, *y*)

ウィンドウを指定の位置に移動させる。

window<<On Close(*script*)

ウィンドウが閉じられたときにスクリプトを実行する。

window<<Pad Window(*Boolean*)

ウィンドウの内容に合わせてパディング（余白）をオン（1）またはオフ（0）にする。デフォルト値はオフです。

window<<Print Window

ウィンドウをデフォルトのプリンタに印刷する。「印刷」ウィンドウは開かれず、ユーザによる入力が不要です。

window<<Reshow

ディスプレイボックスを無効にして、ウィンドウを新しいコンテンツで更新する。更新のタイミングをより具体的に制御する必要がある場合は、<<Inval メッセージおよび <<Update Window メッセージを参照してください。

window<<Save Window to Report(*pathname*, <Embed Data(*Boolean*)>)

現在のレポートウィンドウをJMPレポートファイル（.jrp）に保存する。

window<<Set Main Window

指定のウィンドウを、JMPの実行時に表示されるデフォルトのウィンドウとして設定する。

window<<Set Window Icon(*icon name*)

引用符付きの *icon name* 引数で指定されたアイコンをウィンドウのアイコンとして設定する。

window<<Set Window Size(*x*, *y*)

ウィンドウのサイズを変更する。

window<<Show Window(*Boolean*)

1はウィンドウを表示し（ウィンドウが現在開かれていない場合）、0はウィンドウを非表示にする。ウィンドウが（Windowsで）最小化されているか（macOSで）固定されている場合は、ウィンドウを表示すると通常の状態に戻され、最前面に表示されます。

window<<Size Window(*x*, *y*)

ウィンドウのサイズを変更する。

window<<Update Window

ディスプレイボックスに無効になった領域がある場合、そのディスプレイボックスを保持するウィンドウを更新または再描画する。その他の方法については、「window<<Inval」および「window<<Reshow」を参照してください。

window<<Window Class Name

ディスプレイボックスのウィンドウのクラス名を戻す。有効な応答は、DataTable、FormulaEditor、Starter、Journal、Launcher、Report、Dialog、DialogWithMenu、ModalDialog、FindReplace、User、Generic、ToolWindow、FindReplace、AppBuilder、Debuggerです。

window<<Zoom Window

内容がすべて表示されるようにウィンドウのサイズを拡大する。

ダイナミックリンクライブラリ（DLL）のメッセージ

dll object<<Call DLL(*function name*, *signature*, *arguments*)

指定のシグニチャおよび引数で、DLL 内の指定の関数を呼び出す。

dll object<<Declare Function(*name*, *<named arguments>*)

説明

指定の関数における戻り値と引数の型を宣言する。この宣言により、DLL 内の関数を正しく呼び出すことができます。Convention には、名前付き引数として "STDCALL"、"PASCAL"、または "CDECL" のいずれかを使用します。また、Return および Arg において、戻り値と引数の*型*を指定します。*name* 引数は引用符付きです。

オプションの名前付き引数

Alias(*name*) 引用符付きの *name* で指定された名前を、DLL でエンコードされた名前に代えて含める。

Arg(*type*, *<description>*, *<access mode>*, *<array>*) Argは、関数がもつ引数ごとに1回ずつ、複数回使用できます。

typeには、次のいずれかのキーワードによって、引数の型を指定します。"Int8"、"UInt8"、"Int16"、"UInt16"、"Int32"、"UInt32"、"Int64"、"UInt64"、"Float"、"Double"、"AnsiString"、"UnicodeString"、"Struct"、"IntPtr"、"UIntPtr"、"ObjPtr"。

descriptionは、引数に対する何らかのコメントを記載する引用符付きの文字列です。

access mode（オプション）は、引数の渡し方を指定するキーワードです。"input"は、値渡しを意味します。引数の値を渡す時に指定します。"output"は、アドレス渡しを意味します。値が定義されていない引数のアドレスを渡す時に指定します。"update"は、参照渡しを意味します。引数への参照を渡す時に指定します（この時の引数の値は、JSL変数の値となります）。デフォルト値は、"input"です。

arrayはオプションのキーワードです。このオプションは、*type*が"Double"で、access modeが"input"または"update"に指定されている場合のみ有効です。関数の引数が、Doubleの配列であることを指定します。

Convention(*calling convention*) 呼び出しの規則を指定する。"STDCALL"、"PASCAL"、または"CDECL"のいずれか。デフォルト値は"STDCALL"です。STDCALLとPASCALは等価です。

MaxArgs(*n*) 使用可能な引数の最大数を整数で指定する。

MinArgs(*n*) 使用可能な引数の最小数を整数で指定する。

Returns(*type*) 関数からの戻り値の型を指定する。"Int8"、"UInt8"、"Int16"、"UInt16"、"Int32"、"UInt32"、"Int64"、"UInt64"、"Float"、"Double"、"AnsiString"、"UnicodeString"、"Struct"、"IntPtr"、"UIntPtr"、"ObjPtr"。

StackOrder(*order*) 関数の呼び出し時における、スタックへの引数の格納順序を指定するキーワード。有効な値は"L2R"（左から右）、"R2L"（右から左）です。デフォルト値は"R2L"です。

StackPop(*pop*) エクスポートされた関数が戻った後にスタックをクリアする際の方法を指定するキーワード。有効な値は"CALLER"および"CALLEE"です。デフォルト値は"CALLEE"です。

StructArg(*Arg(...)*, *<Arg(...)>*, ..., *<access mode>*, *<pack mode>*, *<description>*) 複数回使用できる。エクスポートしたDLL関数において、構造体の引数を引数として渡す必要がある場合は、StructArgを使用して構造体メンバーを宣言します。Arg引数の構文は、前述のDeclare FunctionにおけるArg引数と同じです。他の引数として、access modeおよびpack modeがあります。

access mode（オプション）は、構造体引数が値（input）によって渡されるか、または参照（update）によって渡されるかを指定するキーワードです。

pack mode（オプション）は、構造体のパック方法を指定するための整数です。有効な値は1、2、4、8、16です。デフォルト値は8です。

"description"（オプション）は、構造体に対する何らかのコメントを記載するための引用符付き文字列です。

dll object<<Get Declaration JSL

DLL内に用意されたJSLによる関数宣言をログに出力する。

dll object<<Load DLL(*path*, <AutoDeclare(*Boolean*|"Quiet"|"Verbose")>)

dll object<<Load DLL(*path*, <"Quiet"|"Verbose">)

説明

path で指定された DLL をロードする。

必須の引数

path DLL をロードする場所のパス。引用符付きで指定します。

オプションの名前付き引数

AutoDeclare(*Boolean*|"Quiet"|"Verbose") AutoDeclare(1) および AutoDeclare("Verbose") は、ログに verbose メッセージを書き込む。AutoDeclare("Quiet") は、ログウィンドウのメッセージを無効にします。このオプションを省略すると、ログに verbose メッセージが書き込まれます。

Quiet|Verbose Declare Function を使用した場合、このオプションは、ログウィンドウへのメッセージ出力のオフ（"Quiet"）とオン（"Verbose"）を切り替える。

dll object<<Show Functions

DLL オブジェクトに対して宣言された関数をログに送る。

dll object<<Unload DLL

DLL をアンロードする。

イメージのメッセージ

この節では、イメージに使用できる JSL メッセージをご紹介します。

関連情報

- イメージ処理の例については、［スクリプトの索引］を参照してください。JMP のメニューで、**［ヘルプ］＞［スクリプトの索引］**を選ぶと、この対話的なヘルプを参照できます。
- その他のリソースは、JMP File Exchange（https://community.jmp.com/community/file-exchange）で入手できます。

img<<Crop(Left(*pix*), Right(*pix*), Top(*pix*), Bottom(*pix*))

既存のイメージから、指定された大きさ（単位はピクセル）の新しいイメージを作成する。

img<<Filter(*name*, *<n>*)

指定されたアルゴリズムに基づいてイメージをフィルタリングする。フィルタリングはイメージ内のノイズを除去するのに便利です。

メモ: JMPのイメージフィルタはすべて、オペレーティングシステムレベルでサポートされます。Windowsで処理されたイメージとmacOSで処理されたイメージは異なる場合があります。

引数

name 引用符付きでJMPイメージフィルタの名前を指定する。使用可能なフィルタは次のとおりです。

- "Despeckle"は、スキャンまたはキャプチャされたイメージからしみ（引っかき傷やごみなど）を取り除く。
- "Edge"は、明度が極端に変化する部分のピクセルを特定し、それらを暗くして極端な変化を抑える。エッジの検出は、表面、深さ、素材、およびライティングにおける変化の検出に使用されます。
- "Enhance"は、ノイズが多いイメージでピクセル間のコントラストを低減する。
- "Median"は、各ピクセルの明度を近接のピクセルと比較することで、ノイズ（ランダムな変動）を低減する。値が大きく異なるときは、近接ピクセルの平均値で置き換えます。
- "Negate"は、各ピクセルの色をそれぞれ補色に変えることで、逆の色またはグレースケールのイメージを作成する。
- "Normalize"は、カラーイメージのピクセルを、ファイル形式の数値システムの範囲全体を使用するよう変更する。この処理によって、イメージの色がより強調されます。
- "Sharpen"は、ピクセルのエッジを目立たせることで、ぼけを低減する。
- "Contrast", *n*は、イメージを明るくまたは暗くする。0.0より大きい値を指定するとイメージが明るくなり、0.0より小さい値を指定すると暗くなります。
- "Gamma", *n*は、イメージの視覚的な表示（明度と彩度）を、モニターの違いを考慮して補正する。1.0より大きい値を指定するとイメージが明るくなり、1.0より小さい値を指定すると暗くなります。
- "Reduce Noise", *n*は、ISO感度が大きいときや露出時間が長いときに発生するイメージ内のランダムな変動（ノイズ）を低減する。
- "Gaussian Blur", radius, sigmaは、イメージのノイズやディテールを低減し、よりスムーズなイメージにする。半径（radius）は各ピクセルの周囲のぼかし半径で、sigmaはGauss分布の標準偏差です。このGaussぼかしは、通常、サイズ変更やエッジ検出の実行の際に使用されます。

img<<Flip Both

イメージの上下左右を反転する。

img<<Flip Horizontal

イメージの左右を反転する。

img<<Flip Vertical

イメージの上下を反転する。

img<<Get EXIF

イメージに埋め込まれたEXIFデータ（シャッター速度や絞り値など）を連想配列として戻す。

img<<Get N Frames

説明

マルチフレームTIFファイルまたはアニメーションGIFファイルに含まれるフレーム数を返す。フレーム番号は、0から開始する。

例

次の例では、4つのフレームからなるTIFファイルを新しいウィンドウに配置し、フレーム1のイメージを表示する。

```
img = New Image( "$DOWNLOADS/Multiframe.tif" );
nframes = img << Get N Frames(); // 4を戻す
img << Set Current Frame( 1 ); // イメージ1を表示する
win = New Window( "Multi-Frame TIFF", img );
```

img<<Get Size

img<<Size

イメージの幅と高さ（単位はピクセル）を含んだリストを戻す。

img<<Rotate(*degrees*)

指定した角度だけイメージを回転させる。

img<<Save Image(*path*)

イメージを指定のパス（引用符付きの *path*）に保存する。

img<<Scale(*scale|xscale, yscale*)

イメージのサイズを指定の倍率で変更する。引数を1つ指定した場合は、幅と高さの両方が同じ倍率でサイズ変更されます。引数を2つ指定した場合は、幅と高さがそれぞれの倍率でサイズ変更されます。

例

```
img = New Image( "$SAMPLE_IMAGES/tile.jpg" );
xs = 2;
img << Scale( xs );
New Window( "Tilex 2", img );

img = New Image( "$SAMPLE_IMAGES/tile.jpg" );
img << Scale( 2, 0.5 ); // イメージの幅を2倍、高さを1/2倍にする
New Window( "Tile squished", img );
```

メモ

イメージのサイズを取得し、倍率を掛け、サイズを設定するスクリプトの代わりにScaleを使用できます。

`img<<Set Current Frame`

説明

マルチフレーム TIFF ファイルまたはアニメーション GIF ファイルにおいて、表示したいフレーム番号を設定する。最初のフレーム番号は 0、最後のフレーム番号はフレーム数から 1 を引いた値となります。たとえば、フレームが 4 つある場合は、フレーム 0 ～フレーム 3 を指定できます。

次も参照

「img<<Get N Frames」

`img<<Set Size(`*`width`*`, `*`height`*`)`

イメージのサイズを指定の大きさ（単位はピクセル）に変更する。イメージの縦横比を固定したい場合は、元のイメージの縦横比と同じ比率の高さと幅を指定します。

`img<<Transparency(`*`fraction`*`)`

イメージの透明度を設定する。*fraction*には、0.0（完全に透明）～ 1.0（透明度なし）の値を指定します。

インタラクティブHTMLのメッセージ

この節では、インタラクティブHTML Web レポート用のJSL メッセージをご紹介します。

Web レポートのメッセージ

`webreport<<Publish(<Add Image(...)>, <Add Report(...)>, <Add Reports(...)>, <Public(`*`Boolean`*`)>, <Index(...), <User Name(...)>, <Password(...)|`*`password function`*`)>, <Prompt("IfNeeded"|"Always"|"Never")>, <URL(...)>, <Publish Data(`*`Boolean`*`)>, Replace(<`*`id`*`>, Prompt("IfNeeded"|"Always"|"Never"))>)`

説明

Web レポートを JMP サーバーに発行する。

戻り値

処理が成功したときは、発行されたレポートのURL

オプションの引数

Add Image インデックスページの冒頭にイメージを挿入する。有効な形式は、"png"、"bmp"、"jpeg"、"jpg"、"tiff"、"tif"です。TitleとDescriptionはオプションです。Titleはイメージの上、Descriptionはイメージの下に表示されます。File(*filepath*)、または引用符付きの文字列を使用してください。以下はその例です。

```
webrpt << Add Image( File( "C:\Users\Public\JMP\Projects\WebJMP\atlas.jpg" ),
   Title( "Atlas" ), Description( "Holding up the world as always." ) );
```

Add Report Webレポート内に、発行するレポートを追加する。

Public(*Boolean*) レポートが公開されるかどうかを指定する。デフォルトではレポートは非公開です。

Index 複数のレポートを含むインデックスページの名前。説明文を指定することもできます。

User Name JMPサーバーに登録されているユーザ名を指定する。

Password ユーザのパスワードを指定する。パスワード関数を定義することもできます。

Prompt サーバーのURL、ユーザ名、パスワードを入力するためのウィンドウを表示する。

URL 公開先の場所。

Publish Data(*Boolean*) データをHTMLに含める。データをHTMLに含めない場合は、レポートに含まれるイメージは、インタラクティブではなく静的になります。無制限に公開するレポートでは、データを共有したくない場合もあるでしょう。

Replace レポートを置換する。そのページが表示されているブラウザのアドレスフィールドからURLを取得する。

JMPアプリケーションのメッセージ

この節では、JMPアプリケーションビルダーとJMPダッシュボードビルダーに使用できるJSLメッセージをご紹介します。これら2つの機能は、アプリケーションやダッシュボードの設計および実行に同じ基礎構造を使用しています。ダッシュボードは、アプリケーションの一形態です。そこで、この節では、ダッシュボードも**アプリケーション**と呼ぶことにします。

例については、［ヘルプ］メニューの［スクリプトの索引］を参照してください。

JMP Appのメッセージ

JMP Appオブジェクトは、アプリケーションビルダーまたはダッシュボードビルダーによって作成されるJMPアプリケーションの主なコントローラです。JMPアプリケーションの中のスクリプトでも外のスクリプトでも、JMP Appオブジェクトを使用できます。

JMPアプリケーションには、初期（エディタがなく、アプリケーションが実行されていない）、実行中、編集中の3つの状態があります。JMP Appオブジェクトは、一度に1つの状態でしか存在しません。あるJMPアプリケーションを編集していて、それを実行することを選択した場合、実行前にそのJMPアプリケーションのコピーが作成されます。

app<<Combine Windows({*reports or data tables*})

説明

指定のプラットフォームレポートまたはデータテーブルのリストを新しいモジュールにまとめる。このメッセージを送信するときは、アプリケーションが初期状態になければなりません。

app<<Debug

JMP アプリケーションに対して JSL デバッガを呼び出す。アプリケーションスクリプトが先に実行されます。その後、モジュールが作成されるたびにデバッガが中断し、モジュールスクリプトを呼び出します。デバッガにおいて、アプリケーションやモジュールと関連付けられたスクリプトをデバッグするためには、ブレークポイントを設定してください。

app<<Edit

初期状態にある JMP アプリケーションに対してアプリケーションビルダーを開始する。エディタはなく、アプリケーションは実行されていません。

app<<Get Modules

アプリケーションに関連付けられているモジュールのリストを取得する。アプリケーションビルダーでは、ワークスペースにおける 1 つのタブが各モジュールに対応しています。モジュールごとに、アプリケーション内にあるウィンドウについて、レイアウトや動作を記述します。

app<<Get Namespace

JMP App() オブジェクトは、アプリケーションスクリプト内で作成された変数に対し、自動的に匿名名前空間を作成します。このメッセージを使うと、この名前空間へのハンドルを取得して変数を確認または変更することができます。この名前空間には、thisApplication というデフォルトの記号があり、アプリケーション自体への参照を含みます。

app<<Get Windows

JMP アプリケーションモジュールのインスタンスとして作成されたすべてのウィンドウのリストを取得する。モジュールの中には、2 つ以上のインスタンスを作成するものもあります。すべてのウィンドウが同時に存在するわけではないので、ウィンドウの数は一定でない可能性があり、また、モジュールの数とは異なる場合もあります。

app<<Open File(*path*)

.jmpapp ファイルまたは .jmappsource ファイルから既存のアプリケーションの状態をリセットする。*path* は引用符付きです。

app<<Relaunch Analysis

実行中のアプリケーションの新しいコピーを作成し、新しいインスタンスを実行する。

app<<Run

アプリケーションを実行する。アプリケーションスクリプトが先に実行され、設定によっては、1つまたは複数のJMPアプリケーションモジュールンのインスタンスオブジェクトが自動的に作成される場合があります。

app<<Save Script to Add-In

app<<Save Script to Data Table

app<<Save Script to Journal

app<<Save Script to Script Window

アプリケーションのスクリプトを指定の場所に保存する。アプリケーションスクリプトは、アプリケーションの定義を含むJMP App()オブジェクトで構成されます。アドインやデータテーブル、ジャーナルに保存されるスクリプトには、アプリケーションを実行するRunメッセージが含まれます。スクリプトウィンドウに保存されるスクリプトには、アプリケーションビルダーを開くためのEditメッセージが含まれます。

JMP App Moduleのメッセージ

JMPアプリケーションモジュールとは、JMPアプリケーションまたはダッシュボード内の1つのコンポーネントについて、ディスプレイボックスのレイアウトと動作を定義したものです。モジュールの種類によって、コンポーネントは、アプリケーション内のウィンドウであったり、ウィンドウの一部であったりします。

module<<Create Instance

JMPアプリケーションモジュールのインスタンスを作成するには、JMP App()スクリプトまたはJMP App Module()スクリプトの中でCreate Instanceを使用します。デフォルトでは、JMPアプリケーションを実行するたびに各アプリケーションモジュールのインスタンスが1つ作成されます。起動ウィンドウやレポートウィンドウなど、複数のウィンドウを使用する複雑なアプリケーションの場合は、デフォルトの設定に変更を加え、モジュールインスタンスの作成を制御する必要があります。

module<<Get Instance

モジュールが属するアプリケーションへのハンドルを戻す。

JMP App Module Instanceのメッセージ

JMPアプリケーションモジュールインスタンスとは、JMPアプリケーションモジュール、画面上のウィンドウ、または別のウィンドウに挿入できるディスプレイボックス要素一式を具現化したものです。

inst<<Create Objects

このメッセージは、JMP Appのモジュールスクリプトにおいて、デフォルトで指定されます。このメッセージは、ディスプレイボックスやウィンドウのオブジェクトをどの時点で作成するかを指定するときに使用します。スクリプトの作成者は、オブジェクトが作成される前に特定の設定を行って、その後にこのメッセージを使用できます。たとえば、アプリケーションのパラメータを特定の値に設定した後に、このメッセージを使用してください。

inst<<Get Box

モジュールインスタンスに関連付けられている最上位のディスプレイボックスへのハンドルを戻す。これは、Save to PDFメッセージやClose Windowメッセージなど、表示コマンドまたはウィンドウコマンドを実行する際に便利です。

inst<<Get Namespace

JMP App()と同様に、各JMPアプリケーションモジュールインスタンスは、モジュールスクリプト内で作成されるすべての変数に対して匿名名前空間を作成します。名前空間には、モジュール内のディスプレイボックスを表す変数もすべて含まれます。この名前空間に含まれるthisModuleInstanceという名前のデフォルトの記号は、自身を参照します。

inst<<Get User Data

inst<<Set User Data

JMPアプリケーションモジュールインスタンスでJSL値を保存し、取得する。値は、数値、引用符付き文字列、リスト、連想配列、およびType()関数で戻されるJSLのタイプのいずれかです。

MATLABのメッセージ

MATLAB接続オブジェクトを使用して、MATLABとのインターフェースをスクリプトで記述できます。MATLAB Connect()というJSL関数を使用して、スクリプト可能なMATLAB接続オブジェクトを取得できます。

mlconn<<Control(<Echo(*Boolean*)>, <Visible(*Boolean*)>)

MATLABの実行を制御する。

戻り値

なし

オプションのグローバル引数

Echo(*Boolean*) 実行したMATLABのプログラムコードを、JMPの「ログ」ウィンドウに出力する。

Visible(*Boolean*) アクティブなMATLABワークスペースの表示／非表示を指定する。

---

mlconn<<Disconnect()

説明

MATLAB接続インターフェースを接続解除する。

---

mlconn<<Execute({*list of inputs*}, {*list of outputs*}, *mCode*, <Expand(*Boolean*)>, <Echo(*Boolean*)>)

MATLABに対して、第1引数のリストに指定されたJMP変数を送り、第3引数に指定されたMATLABコードをサブミットする。第2引数のリストに指定されたMATLAB変数が、JMPに戻されます。

戻り値

成功した場合は0、そうでない場合は0以外

必須の引数

{*list of inputs*} 位置固定、名前のリスト。入力としてMATLABに送られるJMP変数名のリスト。

{*list of outputs*} 位置固定、名前のリスト。出力としてMATLABから取得されるJMP変数名のリスト。

mCode 位置固定、引用符付き文字列。サブミットされるMATLABコード。

オプションの名前付き引数

Expand(*Boolean*) MATLABコードのサブミット前に、コードに対してEval Insertを実行する。

Echo(*Boolean*) 実行したMATLABのプログラムコードを、JMPの「ログ」ウィンドウに出力する。デフォルト値は1（真）です。

---

mlconn<<Get Graphics(*format*)

MATLABのグラフウィンドウに最後に出力されたグラフを、指定の形式で取得する。*format*引数で指定されたグラフィック形式で取得する。

戻り値

JMPピクチャーオブジェクト

オプションの引数

format 位置固定。MATLABのグラフを取得するときの変換形式（引用符付き）。有効な形式は、"png"、"bmp"、"jpeg"、"jpg"、"tiff"、"tif"です。

mlconn<<Get Version()

インストールされているMATLABの現在のバージョンを取得する。

戻り値

行列。MATLABのバージョン番号を示す長さ3のベクトル

mlconn<<Get(*name*)

説明

*name*引数で指定されたMATLABの変数を、JMPで取得する。

戻り値

*name*引数で指定された変数の値

必須の引数

name MATLABから取得するJMP変数の名前。

mlconn<<Is Connected()

説明

接続がアクティブ化どうかを判断する。

戻り値

アクティブなMATLAB接続がある場合は1、そうでない場合は0を戻す。

mlconn<<JMP Name To MATLAB Name(*jmp name*)

説明

MATLABの命名規則に従い、JMP変数名を、対応するMATLAB変数名に変換する。

戻り値

引用符付き文字列。マップされたMATLAB名

必須の引数

jmp name 位置固定。MATLABに送るJMP変数の名前。

mlconn<<Send(*name*, *<named arguments>*)

説明

名前付き変数をJMPからMATLABに送る。

戻り値

成功した場合は0、そうでない場合は0以外

必須の引数

name 位置固定。MATLABに送るJMP変数の名前。

名前付き引数

次の引数はデータテーブルにのみ指定可能です。

`Selected(Boolean)` 参照先データテーブルの選択されている行をMATLABに送る。

`Excluded(Boolean)` 参照先データテーブルの除外されている行のみをMATLABに送る。

`Labeled(Boolean)` 参照先データテーブルのラベルのついている行のみをMATLABに送る。

`Hidden(Boolean)` 参照先データテーブルの非表示の行のみをMATLABに送る。

`Colored(Boolean)` 参照先データテーブルの色のついている行のみをMATLABに送る。

`Markered(Boolean)` 参照先データテーブルのマーカーのついている行のみをMATLABに送る。

`Row States (Boolean, <named arguments>)` 参照先データテーブルにおける行属性の情報を、「RowState」という列名のデータ列を追加して、MATLABに送る。個々の設定をあわせて追加すれば、複数選択も可能です。行属性の各設定には、それぞれ次の値が割り当てられています。

- Selected = 1
- Excluded = 2
- Labeled = 4
- Hidden = 8
- Colored = 16
- Markered = 32

オプションの名前付き Row State 引数

Row Statesには、次の名前付き引数をオプションで指定できます。

`Colors(Boolean)` 行の色を送る。「RowStateColor」という名前のデータ列を追加します。

`Markers(Boolean)` 行のマーカーを送る。「RowStateMarker」という名前のデータ列を追加します。

`mlconn<<Submit(mCode, <named arguments>)`

説明

アクティブなグローバルMATLAB接続に、MATLABコードをサブミットする。

戻り値

成功した場合は0、そうでない場合は0以外

必須の引数

mCode 位置固定、引用符付き文字列。サブミットされるMATLABコード。

名前付き引数

`Expand(Boolean)` MATLABコードのサブミット前に、コードに対してEval Insertを実行する。

`Echo(Boolean)` 実行したMATLABのプログラムコードを、JMPのログに出力する。デフォルト値は1（真）。

mlconn<<Submit File(*path*)

説明

指定のパス（引用符付きの*path*）を使ってステートメントをMATLABにサブミットする。

戻り値

成功した場合は0、そうでない場合は0以外

引数

path 位置固定、引用符付き文字列。実行するMATLABプログラムコードを含むファイルのパス。

名前空間のメッセージ

ns<<Contains(*string*)

引用符付きの string で指定された文字列が名前空間内に存在すれば 1、そうでなければ 0 を戻す。

ns<<Delete Namespace

内部のグローバルリストからこの名前空間を削除する。

名前空間内の変数を削除するには、Remove(*variable name*) メッセージを使用します。

ns<<First

名前空間内で使用されている最初の変数名を引用符付き文字列として戻す。

ns<<Get Contents

名前と値のペアのリストを、それぞれ含んだリストとして戻す。名前は変数名を含んだ引用符付き文字列で、各値は変数に含まれている式（未評価）です。

ns<<Get Keys

名前空間内の変数名のリストを戻す。

ns<<Get Name

名前空間の名前を戻す。

ns<<Get Value(*variable name*);

この名前空間内の指定の変数（引用符付きの *variable name*）に含まれる式を、評価せずに戻す。

ns<<Get Values

名前空間内の各変数に含まれている式を、評価せずにリストとして戻す。

ns<<Get Values(*{variable name1, variable name2, ... }*);

リスト引数で指定された、名前空間内の各変数（引用符付きの *variable name*）に含まれている式を、評価せずにリストとして戻す。指定の変数名が見つからない場合、エラーを戻します。

ns<<Insert(*variable name*, *expr*);

この名前空間に指定の式（*expr*）を含む指定の変数（引用符付きの *variable name*）を挿入する。

ns<<Lock Namespace(*<variable name, ...>*)

名前空間内の指定の変数をロックし、変数が追加または削除されるのを防ぐ。変数が指定されていない場合、名前空間の変数すべてのロックを解除します。

ns<<N Items

名前空間に含まれている変数の数を戻す。

ns<<New Namespace(*name*, *<{list of expressions}>*)

作成されるすべての関数と変数がオプションの引用符付き *name* 引数の中でのみ定義されるような名前空間を作成する。

ns<<Next(*variable name*)l

指定の変数（引用符付きの *variable name*）に続く変数の名前を戻す。

ns<<Remove(*variable name*, ...)

指定の変数（引用符付きの *vaiable name*）または変数のリストを削除する。

ns<<Show Contents

名前空間の内容をログに表示する。

ns<<Unlock Namespace(*variable name*, ...);

名前空間内の指定の変数（引用符付きの *variable name*）のロックを解除する。変数が指定されていない場合、すべての変数のロックが解除されます。

PIサーバーからの読み込みメッセージ

PIサーバーのメッセージ

---

obj<<Authentication Method(*"none"*|*"kerberos"*|*"basic"*)

説明

認証方法を指定する。

---

obj<<Importer(AF Path(Asset Framework path), <Series(string)>, <Start Time(PI time string)>, <End Time(PI time string)>, <UTC(boolean)>, <Boundary Type(string)>, <Max Count(integer)>, <Filter(string)>, <Retrieve Attribute Status(boolean)>, <Intervals(integer)>, <Sync Time(PI time string)>, <Sync Time Boundary Type(string)>, <Interval(duration in PI AFTimeSpan format)>, <Timeout(integer)>)

説明

生データを読み込む新たなインスタンスを作成する。

例

```
Names Default To Here( 1 );
/* 生データと属性のステータスを読み込む */
client = New PI Client(
   URL( "https://myserver.com/piwebapi" ),
   Authentication Method( "basic" ),
   Username( "myuserid" ),
   Password( "mypassword" )
);
importer = client << Importer(
   AF Path(     /* Asset Framework paths */
      "\\myserver\PIData\Atlanta Data Center\Server Rack1\ION 6200 Power Meter1|I A",
      "\\myserver\PIData\Atlanta Data Center\Server Rack1\ION 6200 Power Meter1|I B",
      "\\myserver\PIData\Atlanta Data Center\Server Rack1\ION 6200 Power Meter1|I C"
   ),
   Series( "raw" ),
   Start Time( "*-1d" ), /* PIの時間文字列 */
   End Time( "*" ),      /* PIの時間文字列 */
   UTC( 0 ), /* 指定の開始/終了時刻がUTCに基づいているかどうか - デフォルトはゼロ */
   Boundary Type( "inside" ), /* 選択肢は"inside"（内側）、"outside"（外側）、
   "interpolated"（内挿値） */
   Max Count( 5000 ), /* 取得する値の個数の最大値 - デフォルトは5000 */
   Filter( "" ), /* オプションのフィルタ */
   Retrieve Attribute Status( 1 ), /* 読み込んだ各列の属性のステータスを取得し、対応する"I
   A status"、"I B status"、"I C status"の各列に入れる */
```

```
    Timeout( 120 ) /* サーバーが応答するまで待つ時間 */
);
importer << Run;
```

---

obj<<Modify

説明

最初に指定されたパラメータを使用して、新しいテーブルを読み込むためのウィンドウを起動し、対話式セッションを開始する。

---

obj<<Password(*text*)

説明

認証方法が「基本」のときのみ使用。

---

obj<<Run

説明

読み込みを実行してテーブルを作成する。

---

obj<<URL(*text*)

説明

PIサーバー REST APIエンドポイントのURL。

---

obj<<Username(*text*)

説明

認証方法が「基本」のときは必須。

Importerメッセージ

---

obj<<AF Path(*path1, path2, ...*)

説明

読み込む属性のAsset Frameworkパス。単一のパスでも複数のパスでもよい。パスをJSLリストに入れ、リストの変数を指定することもできる。

---

obj<<Boundary Type(*"inside"*|*"outside"*|*"interpolated"*)

説明

開始時刻と終了時刻付近の値をどのように扱うかを指定する。

obj<<End Time(*text*)

説明

読み込む時系列の終了時刻（オプション）。デフォルトは「*」（現在）。

obj<<Filter(*text*)

説明

PIサーバーのフィルタ式の構文を使ったオプションのフィルタ式。

obj<<Interval(*text*)

説明

データを取る間隔をAFTimeSpan形式で指定する。デフォルトは「1h」。

obj<<Intervals(*n*)

説明

取得する間隔の個数。デフォルト値は、24。

obj<<Max Count(*n*)

説明

取得する値の個数の最大値を設定する。デフォルト値は5000。

obj<<Retrieve Attribute Status(state=*0*|*1*)

説明

別の列に保存されている属性のステータスを取得するかどうか。この「<attribute-name> ステータス」という列は、多重応答プロパティと値ラベルを使って、状態を「良好」、「不確実」、「置換済み」、「注釈済み」の値で表示しています。デフォルトでは、このオプションは無効になっています。

obj<<Series(*"raw"*|*"plot"*|*"interpolated"*)

説明

時系列のタイプを指定する。デフォルト値は「raw」（生データ）。

obj<<Stack(state=*0*|*1*)

説明

同じ名前を持つ属性を1つの列に積み重ね、それらのパスに共通する要素をまとめた列を作成する。

obj<<Start Time(*text*)

説明

（オプション）読み込未対象の開始時刻。デフォルト値は、「*-1d」（24 時間前）。

obj<<Sync Time(*text*)

説明

（オプション）同期時間。間隔の境界の計算を開始するスタートポイントを設定する。

obj<<Sync Time Boundary Type("*inside*"|"*outside*")

説明

開始時刻と終了時刻付近の値をどのように扱うかを指定する。デフォルト値は「inside」（内側）。

obj<<Timeout(*n*)

説明

指定した秒数が経過してもデータが受信されなかった場合に読み込み処理をキャンセルする。負の値を指定した場合、デフォルトの秒数が使用される。

obj<<UTC(state=*0*|*1*)

説明

（オプション）開始時刻と終了時刻のパラメータが UTC に基づいているかどうかを示すフラグ。デフォルト値は 0（偽）。

Python インテグレーションのメッセージ

JMP での Python へのインターフェース機能は、Python 接続オブジェクトに対してメッセージを送ることによって、スクリプトで記述していきます。Python Connect() 関数によって、Python 接続オブジェクトへの参照を取得できます。「Python Connect()」を参照してください。

pythconn<<Control

この関数は廃止されました。

pythconn<<Create JPIP CMD

説明

Python の pip コマンドを使用する、jpip コマンドラインラッパースクリプトを作成する。生成されたスクリプトの保存先ディレクトリを選択するダイアログが開きます。このスクリプトにより pip の全機能が使用可能となり、JMP に組み込まれた Python 環境に必要な環境変数が設定されます。

例

```
Names Default To Here( 1 );
Python Create JPIP CMD();
```

pythconn<<Disconnect

この関数は廃止されました。

pythconn<<Execute(*{list of inputs}*, *{list of outputs}*, *Python_code*)

説明

Python環境に対して、第1引数のリストに指定されたJMP変数を送り、第3引数に指定されたPythonコードをサブミットする。第2引数のリストに指定された変数が、JMPに戻されます。

戻り値

成功した場合は0、そうでなければ1を戻す。実行結果は*list of outputs*として戻されます。出力のJMPリストの各要素に、Pythonの変数値が戻されます。

位置引数

{list of inputs} 入力としてPythonに送られるJMP変数名のリスト。

{list of outputs} Pythonからの出力を格納するJMP変数名のリスト。

Python code サブミットするPythonコード（引用符付き）。

pythconn<<Get(*name*)

説明

*name*引数で指定されたPythonの変数を取得する。

戻り値

*name*引数で指定された変数の値

引数

name Pythonから取得するJMP変数の名前。引数には、数値、引用符付き文字列、行列、リスト、データフレームのいずれかのデータタイプのPython変数を指定できます。

pythconn<<Get Graphics(*format*)

説明

この関数は廃止されました。matplotlibのpyplotによって最後に書き込まれたグラフィックオブジェクトを、引数で指定されたグラフィック形式で戻していました。

例

次の例では、JMP 17以前のコード例をコメント部分で示し、代わりとなるJMP 18以降でのコード例を示しています。

```
Names Default To Here( 1 );
```

```
ml = Python Submit(
   "\[
# この例では、matplotlibパッケージがインストールされている必要がある。
import matplotlib.pyplot as plt
plt.clf()          # 必ずクリーンなプロットから開始する:
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.draw()

# 選択した場所にイメージを保存する
plt.savefig('/tmp/get_graphics_img.png')
]\"
);
// plot = Python Get Graphics( png );       // 廃止
plot = Open( "/tmp/get_graphics_img.png", png );
pngJMP = New Window( "Plot", Picture Box( plot ) );
Python Submit( "plt.close()" );
rc = Delete File( "/tmp/get_graphics_img.png" );
```

pythconn<<Get Version

説明

インストールされているPythonの現在のバージョンを取得する。

戻り値

バージョン番号の5つの構成要素（メジャーバージョン、マイナーバージョン、マイクロバージョン、リリースレベル、シリアル）を含む長さ5のリストを戻す。リリースレベルの値は引用符付き文字列です。

pythconn<<Install Packages

説明

Pythonパッケージを JMP の site-packages ディレクトリにインストールする。シンプルなパッケージのインストール以外の処理を行うには、Python Create JPIP CMD() を使用して、Directory Pick() で選択したディレクトリの中にコマンドラインラッパースクリプトを作成してください。または、JMP の Python スクリプトウィンドウからインストールを実行することもできます。『スクリプトガイド』を参照してください。

例

```
Names Default To Here( 1 );
// numpyおよびpandasパッケージをインストール
conn = Python Connect();
conn << Install Packages( "numpy pandas" );
```

pythconn<<Is Connected

説明

この関数は廃止されました。

戻り値

常に1を戻す。

pythconn<<JMP Name to Python Name(*name*)

説明

Python の命名規則に従い、JMP 変数名を、対応する Python 変数名に変換する。

引数

name Pythonに送るJMP変数の名前。JMPで使用できる変数名の中には、Pythonでサポートされていないものもあります。SendメッセージでJMPからPythonに送る際にサポートされていない変数名を使った場合、変数名が変更されます。どの変数名が変更されたかを確認するには、JMP Name to Python Nameを使用します。

pythconn<<Send(*name*, <Python Name(*name*)>)

説明

JMPからPythonに変数を送る。

戻り値

成功した場合は0、そうでなければ1

引数

name Pythonに送るJMP変数の名前。

<Python Name(*name*)> Python環境での変数の名前を指定する。

pythconn<<Send File(*name*, <Python Name(*name*)>)

説明

JMP データテーブルを Python に送る。pythconn<<Send File() は将来のリリースで廃止される予定です。代わりに pythconn<<Send() を使用してください。

引数

name Pythonに送るデータテーブルの名前。

<Python Name(*name*)> Python環境でのデータテーブルの名前を指定する。

例

```
Names Default To Here( 1 );
PythonConnection = Python Connect();
PythonConnection << Send File( "$SAMPLE_DATA/Big Class.jmp" );
dtname = "$SAMPLE_DATA/Baseball.jmp";
```

```
PythonConnection << Send File( dtname );
PythonConnection << Submit( "print(Big_Class)" );
PythonConnection << Submit( "print(Baseball)" );
  //:*/
  print(Big_Class)
  /*:

  Big Class (5 columns x 40 rows)
  //:*/
  print(Baseball)
  /*:

  Baseball (2 columns x 43 rows)
  0
```

pythconn<<Set(*name*)

説明

Python 環境で変数を割り当てる。

引数

name Python 変数として設定されるオブジェクトの名前を文字列で指定する。

例

```
Names Default To Here( 1 );
PythonConnection = Python Connect();
x = [1, 2, 3];
PythonConnection << Set( x );
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
PythonConnection << Set( dt );
PythonConnection << Submit( "print(x)" );
PythonConnection << Submit( "print(dt)" );
  //:*/
  print(x)
  /*:

  [[1.]
   [2.]
   [3.]]
  //:*/
  print(dt)
  /*:

  Big Class (5 columns x 40 rows)
  0
```

pythconn<<Submit(*Python_code*)

説明

Pythonコードを Python環境にサブミットする。

戻り値

成功した場合は0、そうでなければ1を戻す。

必須の引数

Python_code サブミットする Pythonコード（引用符付き）。

pythconn<<Submit File(*path*)

説明

指定のパス名（引用符付きの *path*）を使ってステートメントを Pythonにサブミットする。

引数

path 実行する Pythonのプログラムコードを含んだファイルのパス（引用符付き）。

Rインテグレーションのメッセージ

R接続の機能は、JSL関数としてだけではなく、オブジェクトへのメッセージとしても用意されています。R接続オブジェクトへの参照は、R Connect() 関数によって取得できます。以下で述べるメッセージをこのR接続オブジェクトに送ることができます。

rconn<<Control(Interrupt|Async(*Boolean*)|Echo(*Boolean*))

Rの制御オプションを変更する。R Submit() における Asyncオプションに1（真）が設定されているとき、このメッセージを送ると、サブミットされたRコードの処理がただちに中止される。

rconn<<Disconnect()

R接続を解除します。

rconn<<Is Connected()

R接続がアクティブな場合は1、そうでない場合は0を戻す。

rconn<<Send File(*name*, *<named arguments>*)

指定のJMP変数をRに送る。

戻り値

成功した場合は0、そうでない場合は0以外

必須の引数

name Rに送るJMP変数の名前を示した引用符付き文字列

オプションのデータテーブルへの名前付き引数

Selected(*Boolean*) 真の場合、参照先データテーブルで選択されている行のみをRに送ります。

Excluded(*Boolean*) 真の場合、参照先データテーブルで除外されている行のみをRに送ります。

Labeled(*Boolean*) 真の場合、参照先データテーブルでラベルがついている行のみをRに送ります。

Hidden(*Boolean*) 真の場合、参照先データテーブルで非表示になっている行のみをRに送ります。

Colored(*Boolean*) 真の場合、参照先データテーブルで色がついている行のみをRに送ります。

Markered(*Boolean*) 真の場合、参照先データテーブルでマーカーがついている行のみをRに送ります。

Row States(*Boolean*, *<named arguments>*) ブール値の引数とオプションの名前付き引数を指定する。「RowState」という名前のデータ列を追加し、参照先データテーブルの行の属性情報をRに送ります。行の属性値は、表3.2に示す値を持った個別の設定で構成されます。

表3.2 行の属性

個別の設定をまとめて追加することによって、複数の行の属性を作成できます。	Selected = 1 Excluded = 2 Labeled = 4 Hidden = 8 Colored = 16 Markered = 32
引数	Colors(*Boolean*)（オプション）真の場合、行の色を送り、「RowStateColor」という名前のデータ列を追加します。 Markers(*Boolean*)（オプション）真の場合、行のマーカーを送り、「RowStateMarker」という名前のデータ列を追加します。

rconn<<Send(*name*, *<R Name(name)>*)

引用符付きのnameで指定されたJMPデータファイルをRに送る。*name*引数には、数値・引用符付き文字列・行列・リスト・データテーブルのいずれかのデータタイプを指定できます。

rconn<<Get(*name*)

Rのデータを戻す。*name*引数には、数値・引用符付き文字列・行列・リスト・データテーブルのいずれかのデータタイプを指定できます。

戻り値

指定の変数の値

引数

name Rから取得するJMP変数の名前を引用符付きで指定する。

rconn<<Get Graphics(*type*)

Rのグラフウィンドウに最後に出力されたグラフを、指定の形式で取得する。グラフィックオブジェクトは指定のグラフィック形式で戻されます。

戻り値

JMPピクチャーオブジェクト

必須の引数

type Rのグラフを取得するときの変換形式。有効な形式は、"png"、"bmp"、"jpeg"、"jpg"、"tiff"、"tif"です。

rconn<<Submit(*R code*, Expand(*Boolean*), Echo(*Boolean*))

引用符付きで指定されたRコードをサブミットする。

戻り値

成功した場合は0、そうでない場合は0以外

必須の引数

code サブミットするRコードを引用符付きで指定する。

オプションの名前付き引数

Expand(*Boolean*) コードをサブミットする前に、Rコードに対してEval Insert()を実行します。

Echo(*Boolean*) 実行したRのプログラムコードを、JMPのログに出力します。デフォルト値は、1（真）。

Rconn<<Submit File(*path*)

引用符付きの*path*で指定されたファイルにあるプログラムを、Rでサブミットする。

引数

path 実行するRのコードが含まれているファイルのパスを引用符付きで指定する。

rconn<<Execute(*{list of inputs}*, *{list of outputs}*, *R code*, *<named arguments>*)

Rに対して、第1引数のリストに指定されたJMP変数を送り、第3引数に引用符付きで指定されたRコードをサブミットする。第2引数のリストに指定されたR変数が、JMPに戻されます。

戻り値

成功した場合は0、そうでない場合は0以外

必須の引数

R code サブミットするRコードを引用符付きで指定する。

{list of inputs} 入力としてRに送られるJMP変数名のリスト

{list of outputs} 出力としてRから取得されるJMP変数名のリスト

オプションの名前付き引数

「rconn<<Submit(R code, Expand(Boolean), Echo(Boolean))」を参照してください。

rconn<<Control(<Echo(*Boolean*)>)

Rの実行を制御する。

戻り値

なし

オプションの名前付き引数

Echo(*Boolean*) 実行したRのプログラムコードを、JMPのログに出力します。

rconn<<Get Version()

現在インストールされているRのバージョンを取得する。

戻り値

Rのバージョン番号を示す行列

rconn<<JMP Name To R Name(*name*)

Rの命名規則に従い、引用符付きで指定されたJMP変数名を、対応するR変数名に変換する。

戻り値

Rでの命名規則に従った変数名を示す引用符付き文字列

引数

name R変数名に変換するJMP変数名を示す引用符付き文字列

スケジュールのメッセージ

この節では、スケジュールに使用できるJSLメッセージをご紹介します。

関連情報

- 「SASデータセットのパス名を示す引用符付き文字列。」も参照してください。

sch<<Clear Schedule()

スケジュールが設定されているイベントをすべてキャンセルする。

sch<<Close()

スケジューラを閉じる。

sch<<Restart()

スケジュール設定されたすべてのイベントの実行を停止した後で、スケジューラを再起動する。

sch<<Show Schedule()

スケジュールが設定されたすべてのイベントのリストを表示する。

sch<<Stop()

スケジュール設定されたすべてのイベントのスケジューラによる実行を停止する。

セグメントのメッセージ

Pie Seg(*<style>*, {*x*, *y*}, *radius*, [*values*])

説明

指定の値を使って、指定の原点の位置に指定の半径で円グラフのセグメントを作成する。

必須の引数

{*x*, *y*} 円グラフのセグメントを表示する位置をx座標とy座標で指定する。

radius 半径を指定する。

values 値を行列形式で指定する。

オプションの引数

style スタイルを指定する引用符付き文字列。"Pie"（円グラフ。扇形のサイズが要約統計量によって決まる従来型の円グラフ）、"Ring"（ドーナツ。層別化変数の変数または水準が、それぞれ同心のドーナツ形状で表されます）、"Coxcomb"（鶏頭図。中央の角度はすべての扇形で同じです）。

ソケットのメッセージ

skt<<Accept(*<callback, timeout>*)

説明

サーバーソケットに、接続を受け入れて、新しく接続されたソケットを戻すよう伝える。

戻り値

最大4項目のリスト。最初の引用符付き文字列はコマンド（"accept"）。2番目の引用符付き文字列は"ok"またはエラー。3番目の引用符付き文字列は接続したコンピュータの名前を指定するもの。4番目の文字列は、さらにメッセージを送信するためのソケットへの参照。

オプションの引数

callback データを受信する関数の名前を指定。

*timeout callback*を使用する場合は、*timeout*で関数が応答するまでの待機時間を指定。サーバーソケットに対しては、しばらくの間接続が行われていなくてもサーバーをシャットダウンするべきではないので、0を設定することもあるでしょう。

skt<<bind(*localhost*, *port*)

説明

ローカルコンピュータのポートをソケットに関連付ける。

戻り値

2つの引用符付き文字列のリスト。最初の文字列はコマンド名（"bind"）、2番目の文字列は、成功した場合は"ok"、そうでなければエラー。

必須の引数

localhost ローカルコンピュータを引用符付きで指定する。別のコンピュータにバインドすることはできない。

port 使用するポートを指定する。

skt<<Close()

説明

ソケットを閉じる。

戻り値

2つの引用符付き文字列のリスト。最初の文字列はコマンド名（"close"）、2番目の文字列は、成功した場合は"ok"。

skt<<Connect(*socketname*, *port*)

説明

リスニングソケットに接続します。

戻り値

2つの引用符付き文字列のリスト。最初の文字列はコマンド名（"connect"）で、2番目の文字列は、成功した場合は"ok"、そうでない場合は相手側のソケットから返されるエラー。

引数

socketname 相手側のソケットの名前を指定する。Webサーバーに接続する場合は、（IPアドレスではなく）Webアドレスを指定する。

port 相手側のソケットで接続に使用されるポートを指定する。

skt<<GetPeerName()

説明

接続先のソケットのアドレスとポートを取得する。

戻り値

4つの引用符付き文字列のリスト。最初の文字列はコマンド（"getpeername"）。2番目の文字列は"ok"またはエラー。3番目がアドレスで、4番目がポート。

skt<<Get Sock Name()

説明

接続元（こちら側）のソケットのアドレスとポートを取得する。

戻り値

4つの引用符付き文字列のリスト。最初の文字列はコマンド（"getsockname"）。2番目の文字列は"ok"またはエラー。3番目がアドレスで、4番目がポート。

skt<<ioctl(FIONBIO, *Boolean*)

説明

ソケットのブロック動作をコントロールする。

戻り値

2つの引用符付き文字列のリスト。最初の文字列はコマンド名（"ioctl"）、2番目の文字列は、成功した場合は"ok"、そうでなければエラー。

引数

FIONBIO, 1 FIONBIOはNon-Blocking I/O（非ブロッキング入出力）を意味する。1（真）の場合、この機能および引数をオンにする。

skt<<Listen()

説明

接続をlisten状態にするようサーバーに伝える。

戻り値

2つの引用符付き文字列のリスト。最初の文字列はコマンド名（"listen"）で、2番目の文字列は、成功した場合は"ok"、そうでなければエラー。

skt<<recv(*n*, <*callback*, *timeout*>)

skt<<recvfrom(*n*, <*callback*, *timeout*>)

説明

相手側のソケットからストリームメッセージ（recv）またはデータグラムメッセージ（recvfrom）のいずれかを受信する。2つのオプションの引数が使用された場合、データは即座に受信されず、*callback*関数が呼ばれたときに受信されます。

戻り値

3つの引用符付き文字列のリスト。最初の文字列はコマンド名（"recv"または"recvto"）。2番目は、成功した場合は"ok"、そうでなければエラー。3番目の文字列は受信したデータ。*callback*関数が使用された場合は、4番目の要素として元のrecvまたはrecvfromメッセージで使用されたソケットを戻す。

必須の引数

n 相手側のソケットから受信するバイト数を指定する。

オプションの引数

callback データを受信する関数の名前を指定。

*timeout callback*を使用する場合は、*timeout*で関数が応答するまでの待機時間を指定。

skt<<Send(*stream*)

skt<<SendTo(*dgram*)

説明

引数内のデータを、相手側のソケットに送る。Sendはストリームを送り、Sendtoはデータグラムを送ります。

戻り値

3つの引用符付き文字列のリスト。最初の文字列はコマンド名（"send"または"sendto"）。2番目は、成功した場合は"ok"、そうでなければエラー。3番目の文字列は送信できなかったストリームの一部、または、すべてのデータが正常に送信された場合は空白。

引数

stream 相手側のソケットに送信するコマンドを指定する。

dgram 相手側のソケットに送信するコマンドを指定する。

メモ

引数にバイナリデータが必要な場合があります。JMPでは、印刷できないASCII文字を波形符号（~）と16進数で示します。たとえば、次のようになります。

```
skt<<send(("GET / HTTP/1.0~0d~0a~0d~0a");
```

これは、HTTPサーバーにGET要求を送信します。

SQLのメッセージ

obj<<Custom SQL(*sql*)

説明

クエリーをカスタムSQLクエリーに変更し、SQLを設定する。

必須の引数

`Sql` 引用符付きのSQLクエリー。

obj<<Generate SQL

クエリーを実行するときに生成されるSQLを戻す。

obj<<Modify

クエリーをクエリービルダで開く。

obj<<PostQueryScript(*script as text*)

クエリーが実行された後に実行されるJSLスクリプトを設定する。*script as text*は、引用符付きのJSLコード。

obj<<Query Name(<*new name*>)

クエリーの名前を取得（*new name*引数を省略）または設定（*new name*引数を指定）する。クエリーの名前は、そのクエリーを実行した結果のデータテーブル名に使用される。

obj<<Run(<"Private"|"Invisible">, <Update Table(*table*)>, <OnRunComplete(*script*)>, <OnRunCanceled(*script*)>, <OnError(*script*)>)

説明

［クエリービルダー］環境設定の［可能な限りクエリーをバックグラウンドで実行］の選択にしたがって、SQLクエリーをバックグラウンドまたはフォアグラウンドで実行する。

戻り値

クエリーがバックグラウンドで実行された場合はなし。クエリーがフォアグラウンドで実行された場合はデータテーブル。

オプションの名前付き引数

`"Private"` クエリーにより生成されたデータテーブルをデータテーブルウィンドウに表示せずに開く、引用符付きのキーワード。`"Private"`は、スクリプトに`OnRunComplete`が含まれている場合のみ使用可能です。

`"Invisible"` クエリーが生成したデータテーブルを非表示にする引用符付きのキーワード。クエリーの結果を非表示にして、後に続くクエリーで使用する場合、この引数を使用してください。このデー

タテーブルは、ホームウィンドウの「ウィンドウリスト」または［**ウィンドウ**］**>**［**再表示**］のリストに表示されます。

Update Table 指定されたデータテーブルを更新する。クエリーはフォアグラウンドで実行される。

OnRunComplete クエリーが完了した後に実行するスクリプトを指定する。結果のデータテーブルを取得するには、OnRunCompleteを含めます。OnRunCompleteスクリプトは、次の例の二重コロンが示すようにグローバル名前空間で定義します。

```
Names Default To Here( 1 );
::onComplete = Function( {dt},
   {default local},
   Write(
      "\!NQuery is complete!Result name: \!"",
      dt << Get Name,
      "\!", Number of rows: ",
      N Rows( dt )
   )
);

query = Include( "rentals_fam_romcom.jmpquery" );
query << Run Background( On Run Complete( ::onComplete ) );
```

OnRunCanceled ユーザがクエリーをキャンセルした後に実行するスクリプトを指定する。

OnError エラーが起こったときに実行するスクリプトを指定する。

メモ

バックグラウンドクエリーの結果のデータテーブルを取得するには、オプションのOnRunComplete引数を使用します。クエリーが完了したときに実行するスクリプトを記述し、さらに結果のデータテーブルへのデータテーブル参照を割り当てます。または、データテーブルを最初の引数として受け入れる関数に引数として渡すこともできます。クエリーが完了した後、その関数が呼び出されます。

例

次の例は、クエリービルダーで保存したクエリーを開きます。クエリーはプライベートに（クエリービルダーを開かずに）開きます。クエリーが実行され、結果のデータテーブルが開きます。

```
query = Open( "C:\My Data\Movies.jmpquery", "Private");
dt = query << Run();
```

スクリプトに.jmpqueryファイルを指定して、このクエリーをバックグラウンドで実行するには、<<Run Backgroundメッセージを使用します。

```
query = Include( "C:\Queries\movies.jmpquery");
query <<Run Background();
```

次の例は、データベースにクエリーを送り、結果のデータテーブルを開き、データテーブルの行数をログに出力します。

```
confirmation = Function( {dtResult},
   Write( "\!Nクエリーの結果の行数: ", N Rows( dtResult ) )
);
query = New SQL Query(
```

```
      Connection(
         "ODBC:DSN=SQL
      Databases;APP=MYAPP;TrustedConnection=yes;WSID=D79255;DATABASE=SQB;"
      ),
      QueryName( "movies_to_update" ),
      Select( Column( "YearMade", "t1" ), Column( "Rating", "t1" ) ),
      From( Table( "g6_Movies", Schema( "SQB" ), Alias( "t1" ) ) ),

   );
   query << Run( OnRunComplete( confirmation ) );
```

Run Background(<OnRunComplete(*script*), <"Private"|"Invisible">>, <OnRunCanceled(*script*)>, <OnError(*script*)>)

説明

SQLクエリーをバックグラウンドで実行する。実行中のクエリーは表示されません。

戻り値

なし（または、OnRunCompleteが含まれている場合はデータテーブルオブジェクト）

オプションの名前付き引数

OnRunComplete(*script*) クエリーが完了した後に実行するスクリプトを指定する。結果のデータテーブルを取得するには、OnRunCompleteを含めます。OnRunCompleteスクリプトは、次の例の二重コロンが示すようにグローバル名前空間で定義します。

```
Names Default To Here( 1 );
::onComplete = Function( {dt},
   {default local},
   Write(
      "\!NQuery is complete!Result name: \!"",
      dt << Get Name,
      "\!", Number of rows: ",
      N Rows( dt )
   )
);

query = Include( "rentals_fam_romcom.jmpquery" );
query << Run Background( On Run Complete( ::onComplete ) );
```

"Private" 結果のデータテーブルを開かない。必ずOnRunCompleteとともに使用します。バックグラウンドで実行するクエリーにprivateを指定すると、データテーブルは開かれたうえで非表示（invisible）になります。

"Invisible" データテーブルを非表示にする。クエリーの結果を非表示にして、後に続くクエリーで使用する場合、この引数を使用してください。このデータテーブルは、ホームウィンドウの「ウィンドウリスト」または［**ウィンドウ**］**>**［**再表示**］のリストに表示されます。

OnRunCanceled ユーザがクエリーをキャンセルした後に実行するスクリプトを指定する。

OnError エラーが起こったときに実行するスクリプトを指定する。

メモ

クエリーはすべてバックグラウンドで実行されます。これは、クエリービルダー環境設定の［可能な限りクエリーをバックグラウンドで実行］（デフォルトで選択されている）に基づいています。

スクリプトに .jmpquery ファイルを指定して、このクエリーをバックグラウンドで実行するには、Run Background メッセージを使用します。

```
query = Include( "C:\Queries\movies.jmpquery");
query <<Run Background();
```

Run Foreground(<OnRunComplete(*script*), <"Private"|"Invisible">>, <OnRunCanceled(*script*)>, <OnError(*script*)>)

説明

SQL クエリーをフォアグラウンドで実行する。

戻り値

クエリーが完了したときに開くデータテーブル

次も参照

「Run Background(<OnRunComplete(script), <"Private"|"Invisible">>, <OnRunCanceled(script)>, <OnError(script)>)」

---

obj<<Save

クエリーを、それが関連付けられているファイルに保存する。関連付けられているファイルがまだない場合、保存に失敗します。

---

obj<<Save As(*path*, <Replace Existing(*Boolean*))

クエリーを、指定されたファイルに保存する。ファイルがすでに存在している場合、Replace Existing が真でなければ保存に失敗します。

その他のオブジェクトのメッセージ

この節では、Zip アーカイブやジャーナルなど、その他のオブジェクトに使用できる JSL メッセージをご紹介します。

Zip アーカイブ

この節では、Zip アーカイブに使用できる JSL メッセージをご紹介します。

---

list = za<<Dir

メンバー名のリストを戻す。

data = za<<Read(*member name*, <Format("*blob*")>)

引用符付きの*member name*全体を含む引用符付き文字列を戻す。ファイル名（または*"member name"*）で構成されるzipファイル。

メモ

OPEN関数でURLを指定して、zipデータを開いた場合、JMPは、そのリモートファイルをローカルディスクにコピーします。zipデータが使用されなくなると、ローカルのファイルは削除されます。

actualname = za<<Write(*member name*, *member data*, *<"replace">*)

zipアーカイブのメンバファイルに引用符付き文字列または引用符付きのBLOBを書き込む。引用符付きの*member name*で指定したファイルが現在のzipファイル内にない場合、*member name*と同じ名前のactualnameが戻されます。既存のメンバーファイルが上書きされないように、メンバーファイル名は変更され、実際に使用される名前が戻されます。引用符付きの*member data*引数は、その名前を持つzipファイルのメンバに書き込まれるデータです。*replace*は、一時的な名前を付けたファイルを作成し、前のファイルを削除してから一時ファイルの名前を既存の名前に変更します。

ジャーナル

この節では、JMPジャーナルに使用できるJSLメッセージをご紹介します。

jnl<<Save HTML(*<path>*, *<format>*)

ジャーナルをHTMLとして保存する。

オプションの引数

"path" 保存されるHTMLファイルのパスを引用符付きで指定する（たとえば、"C:\myFile.html"）。

format グラフィックファイルの形式。引用符付きで指定します。JPG、PNG、TIFFの各形式がサポートされています。グラフィックはgfxという名前のサブディレクトリに保存されます。

jnl<<Save RTF(*<path>*, <format>)

ジャーナルをRTFファイルとして保存する。

オプションの引数

"path" 保存されるRTFファイルのパスを引用符付きで指定する（たとえば、"C:\myFile.rtf"）。

"format" 埋め込みグラフィックのファイル形式。引用符付きで指定します。Windowsでは、"JPG"、"PNG"、"EMF"の各形式がサポートされています。macOSでは、デフォルトですべてのジャーナルがPDFファイルとして保存されます。

メモ

*path*や*format*が指定されていない場合、Windowsではファイル名と形式を指定するよう促されます。macOSでは、ファイル名を入力するよう促されます。ファイルは、デフォルトでPDFファイルとして保存されます。

---

jnl<<Save PDF(<*path*>, <Show Page Setup(*Boolean*)>, <Portrait(*Boolean*)>)

ジャーナルをPDFファイルとして保存する。

オプションの引数

"path" 保存されるPDFファイルのパスを引用符付きの*path*で指定する（たとえば、"C:\myFile.pdf"）。

Show Page Setup 1（真）に設定すると、「ページ設定」ウィンドウが開く。このウィンドウで、余白、倍率、その他のページレイアウトオプションを変更できます。

Portrait 用紙の向きを縦または横のいずれかに設定する。ここでの設定内容は、Show Page Setupで表示されるウィンドウでユーザが選択した内容よりも優先されます。

付録 A

JMP クエリーで使用可能な SQL 関数

JSL 関数の Query() は、選択したテーブルに対して SQL クエリーを実行し、データをデータテーブルに書き出します。次の例では、「Big Class.jmp」に t1 という別名を割り当て、t1 テーブルから名前、年齢（14 歳以上）、身長(インチ)を選択します。

```
dt = Open( "$SAMPLE_DATA/Big Class.jmp" );
Query( Table( dt, "t1" ),
       "SELECT t1.名前, t1.年齢, t1.\!"身長(インチ)\!"  FROM t1
WHERE t1.年齢 > 13" );
```

クエリーでは、SQL の関数を使用できます。たとえば、SELECT CURRENT_TIMESTAMP は、現在の日時（UTC/GMT）を SQLite の日付時間文字列として戻します。

```
Query( Scalar, "SELECT CURRENT_TIMESTAMP;" );
```

この付録には、SQL クエリーで使用できる数値関数、日付時間関数、文字列関数、SQL システム関数、集計関数の一覧を示します。ネイティブの SQLite 関数には、該当欄に「○」と記します。詳細については、SQLite のオンラインマニュアル https://www.sqlite.org/lang.html を参照してください。

目次

SQL の数値関数 489
SQL の日付時間関数 490
SQL の文字列関数 493
SQL のシステム関数 494
SQL の集計関数 495

SQLの数値関数

以下に、SQLの数値関数について説明します。

数値関数	SQLite ネイティブ	説明
ABS(*number*)		指定された数値（*number*）の絶対値を戻す。
ACOS(*cosine*)		指定された余弦（*cosine*）の角度をラジアンで戻す。
ASIN(*sin*)		指定された正弦（*sin*）の角度をラジアンで戻す。
ATAN(*tangent*)		指定された正接（*tangent*）の角度をラジアンで戻す。
ATAN2(*x*, *y*)		逆正接関数（引数を2つとる）。
CEILING(*number*) CEIL(*number*)		指定された数値（*number*）より大きい最小の整数を戻す。
COS(*radians*)		指定された角度（*radians*）の余弦を戻す。
COT(*radians*)		指定された角度（*radians*）の余接を戻す。
DEGREES(*radians*)		角度（*radians*）をラジアンから度に変換する。
EXP(*number*)		定数eを、指定された値（*number*）でべき乗した結果を戻す。
FLOOR(*number*)		指定された数値（*number*）より小さい最大の整数を戻す。
LN(*number*) LOG(*number*)		指定された数値（*number*）の自然対数を戻す。
LOG10(*number*)		指定された数値（*number*）の常用対数を戻す。
MAX(*n1*, *n2*, ...)	○	指定された数値のうち、最大値を戻す。2つ以上の数値を指定する必要があります。
MIN(*n1*, *n2*, ...)	○	指定された数値のうち、最小値を戻す。2つ以上の数値を指定する必要があります。
MOD(*dividend*, *divisor*)		被除数（*dividend*）を除数（*divisor*）で割った余りを戻す。小数部のある数を指定した場合は、小数点以下を切り捨てて整数にしたうえで、計算を実行します。
PI()		定数piの値を戻す。
POWER(*number*, *power*) POW(*number*, *power*)		数値（*number*）を、指定された値（*power*）でべき乗した結果を戻す。
RADIANS(*degrees*)		角度（*degrees*）をラジアンに変換する。

数値関数	SQLite ネイティブ	説明
RANDOM() RANDOM(*max*) RANDOM(*min*, *max*)		乱数を戻す。RANDOM()は、0～1の値を戻します。RANDOM(max)は、0～*max*の値を戻します。RANDOM(min, max)は、*min*～*max*の値を戻します。この関数は、JSL関数のRandom Uniform()と同じです。シード値は、JSL関数のRandom Reset()を使って制御することができます。RANDOMは、RANDと略記することもできます。
RANDOMBLOB(*length*)	○	NバイトのBLOBの乱数を戻す。詳細については、SQLiteのオンラインマニュアルhttps://www.sqlite.org/lang.htmlを参照してください。
ROUND(*number*, *<precision>*)		数値（*number*）を、指定された小数点以下の桁数（*precision*）になるように四捨五入する。*precision*のデフォルト値は0です。また、*precision*には負の値を指定することもできます。
SIGN(*number*)		*number*が正の数である場合は1を、*number*が負の数である場合は-1を、*number*が0である場合は0を戻す。
SIN(*radians*)		指定された角度（*radians*）の正弦を戻す。
SQRT(*number*)		*number*の平方根を戻す。
TAN(*radians*)		指定された角度（*radians*）の正接を戻す。
TRUNCATE(*number*, *<precision>*)		数値（*number*）を、指定された小数点以下の桁数（*precision*）になるように切り捨てる。*precision*のデフォルト値は0です。また、*precision*には負の値を指定することもできます。TRUNCATE()は、TRUNC()と略記することもできます。

SQLの日付時間関数

JMPのクエリーで日付時間関数を使用する場合は、SQLエンジン（SQLite）とJMPの日付格納形式が異なるため、注意が必要です。SQLiteは、日付時間値を文字列として格納しますが、JMPは、1904年1月1日からの秒数で格納します。テーブルに日付時間値の列がある場合は、この変換は自動的に行われますが、日付時間値を戻す関数を使用した場合は、必要に応じてJMPで変換する必要があります。

CURRENT_TIMESTAMP関数を例にとって説明します。CURRENT_TIMESTAMPはビルトインのSQLite関数で、現在の日時（UTC/GMT）をSQLiteの日付時間文字列として戻します。

```
Query( Scalar, "SELECT CURRENT_TIMESTAMP;" );
```

この式は、次の値を戻します。

```
"2016-02-16 15:44:42"
```

この文字列を解析し、JMPの日付時間値として戻すこともできますが、その手間を省くためには、以下のようにCURRENT_TIMESTAMP関数をJMPDATE()関数に挿入します。

```
Query( Scalar, "SELECT JMPDATE( CURRENT_TIMESTAMP );" );
```

この式は、次の値を戻します。

3538482531

これは、JMPの日付時間値（表示形式を設定していない状態）です。SQLiteの日付時間文字列を、別のSQL日付時間関数に渡す場合は、自動的にJMPの日付時間値に変換されるので、JMPDate()を使用する必要はありません。以下はその例です。

```
Query( Scalar, "SELECT EXTRACT('YEAR', CURRENT_TIMESTAMP);" );
```

SQLiteネイティブの日付時間関数（date()、time()、datetime()、julianday()、strftime()）は、JMPの日付時間値と互換でないため、JMPクエリーではこれらの関数を使用しないでください。

日付時間関数	SQLite ネイティブ	説明
CURRENT_DATE	○	現在の日付（UTC/GMT）を、SQLiteの日付時間文字列として戻す。
CURRENT_TIME	○	現在の時間（UTC/GMT）を、SQLiteの日付時間文字列として戻す。
CURRENT_TIMESTAMP	○	現在の日時（UTC/GMT）を、SQLiteの日付時間文字列として戻す。
DATEDIFF(*date1*, *date2*, *interval*, <*alignment* = "*Start*">)		2つの日付の差を計算する（単位を*interval*、基準を*alignment*で指定する）。この関数は、JSL関数のDate Difference()と同じように動作します。*interval*には、"Year"、"Quarter"、"Month"、"Week"、"Day"、"Hour"、"Minute"、"Second"のいずれかを指定できます。*alignment*には、"Start"、"Actual"、または"Fractional"を指定できます。*alignment*が指定されていない場合、"Start"が使用されます。

日付時間関数	SQLite ネイティブ	説明	
EXTRACT(*datepart*, *datetime*, *<use_locale = 1>)*		日付値または日付時間値の一部を抽出する。*datetime*には、JMPの日付時間値か、SQLiteの日付時間文字列を指定できます。*use_locale*（オプション）は、日付時間値の一部を名前で戻す関数において（月名（"MonthName"）や曜日（"DayName"）など）、現在の言語と英語のどちらを使用するかを指定します。*datepart*には、以下の値を指定できます。	
		"Year"	年を数字で戻す。
		"Month"	月を数字で戻す（1-12）。
		"MonthName"	月名を、現在の言語（*use_locale = 1*）または英語（*use_locale = 0*）で戻す。
		"Mon", "MMM"	月名を略称で戻す。
		"Day"	日にち（月単位）を戻す（1-31）。
		"DayName"	曜日を戻す。
		"DayOfWeek"	曜日を数字で戻す（1-7）。
		"DayOfYear"	日にち（年単位）を戻す（1-366）。
		"Quarter"	四半期の値を戻す（1-4）。
		"Hour"	時間を戻す（0-23）。
		"Minute"	分を戻す（0-59）。
		"Second"	秒を戻す（小数点を含む）。
		"Date"	日付の部分だけ抽出して、JMPの日付時間値として戻す。
		"Time"	時間の部分だけ抽出して、JMPの日付時間値として戻す。
JMPDATE(*SQLite time string*)		SQLiteの日付時間文字列を、JMPの日付時間値に変換する。	
NOW()		TODAY()と同じ。	
TODAY()		現在のJMP日付時間値を**ローカルの**時間で戻す（JMPのToday()関数と同じ）。	

SQLの文字列関数

以下に、SQLの文字列関数について説明します。

関数	SQLite ネイティブ	説明
HEX(*binary*)	○	BLOBを16進数文字列に変換するビルトインのSQLite関数。RANDOMBLOB() 関数と組み合わせて使用すると便利です。
JLEFT(*string*, *len*, *<pad>*)		JSLの Left() 関数と同じ。文字列（*string*）の先頭から *len* 文字を抽出して戻す。*pad*が指定され、文字列（*string*）の長さが *len* より短い場合は、*pad*を末尾に追加して、長さを *len*にして戻します)。
JRIGHT(*string*, *len*, *<pad>*)		JSLの Right() 関数と同じ。文字列（*string*）の末尾から、*len*文字を抽出して戻す。*pad*が指定され、文字列（*string*）の長さが *len*より短い場合は、*pad*を先頭に追加して、長さを *len*にして戻します)。
LENGTH(*string*)	○	ANSI標準の CHAR_LENGTH() に対応するSQLite関数。文字列（*string*）の文字数を戻します。
LOCATE(*string1*, *string2*) POSITION(*string1*, *string2*)		文字列 *string2*の中から文字列 *string1*を探し、その開始位置を戻す（先頭位置を1とする）。見つからなかった場合は、0を戻す。
LOWER(*string*)		文字列（*string*）に含まれる大文字をすべて小文字に変換して戻す。
LTRIM(*string*, *<trimchars>*)	○	文字列（*string*）の先頭が、*trimchars*に指定された文字のいずれかと一致する場合、これを削除した結果を戻す。*trimchars*を省略した場合は、スペースが削除される。
PRINTF(*format*, *<arg1, ..., argN>*)	○	表記と引数を指定して、文字列を生成する。詳細については、SQLiteのオンラインマニュアル https://www.sqlite.org/lang.html を参照してください。
REPLACE(*string*, *find*, *replace*)	○	文字列（*string*）の中から文字列（*find*）をすべて検索し、文字列（*replace*）で置き換える。*replace*に数値を指定した場合は、文字列に変換されます。
REVERSE(*string*)		文字列（*string*）を反転して戻す。
RTRIM(*string*, *<trimchars>*)	○	文字列（*string*）の末尾が、*trimchars*に指定された文字のいずれかと一致する場合、これを削除した結果を戻す。*trimchars*を省略した場合は、スペースが削除される。
SPACE(*length*)		*length*個のスペースからなる文字列を戻す。

関数	SQLite ネイティブ	説明
SUBSTR(*string*, *start*, *<length>*)	○	文字列（*string*）の指定された位置（*start*）から *length* で指定した数の文字を戻す（先頭位置を 1 とする）。*length* を省略した場合は、指定された点（*start*）から末尾までが戻されます。
TRIM(*string*, *<trimchars>*)		文字列（*string*）の末尾が、*trimchars* に指定された文字のいずれかと一致する場合、これを削除した結果を戻す。*trimchars* を省略した場合は、スペースが削除される。
UPPER(*string*)		文字列（*string*）に含まれる小文字をすべて大文字に変換して戻す。

SQL のシステム関数

以下に、SQL のシステム関数について説明します。

関数	SQLite	説明
COALESCE(*arg1*,..., *argN*)	○	渡された引数のうち、ヌルでない最初の値を戻す。すべての引数がヌルである場合は、ヌルを戻す。引数を 2 つ以上指定する必要があります。
IFNULL(*arg1*, *arg2*)	○	*arg1* がヌルでない場合は *arg1* を、ヌルの場合は *arg2* を戻す。IFNULL は、基本的に、COALESCE() に引数を 2 つ指定するのと同じです。
NULLIF(*arg1*, *arg2*)	○	*arg1* と *arg2* が異なる場合は *arg1* を、等しい場合はヌルを戻す。データベースに含まれるヌルでない特定の値を、ヌルとして扱いたい場合に使用します。

SQL の集計関数

集計関数に 1 つの引数を渡す場合は、その前に DISTINCT というキーワードを挿入して、重複する値を省くことができます。

COUNT(*) を除くすべての集計関数では、ヌルと欠測値は無視されます。

関数	SQLite	説明
AVG(*num_expr*)		グループに含まれる行の、*num_expr* の列の平均値を計算する。*num_expr* の列は数値タイプでなければなりません。
COUNT(*expr*) COUNT(*)		グループの中で、*expr* の列がヌルでない行数を戻す。COUNT(*) は、グループに含まれる全行数を戻します。
GROUP_CONCAT(*expr*, *<separator = ','>*)	○	*expr* の列のうち、ヌルでない値をすべて連結して、文字列として戻す。*expr* の列が数値である場合は、文字に変換されます。区切り文字（*separator*）を指定した場合は、値の間に挿入されます。デフォルトの区切り文字はカンマです。DISTINCT は、GROUP_CONCAT() で *separator* が指定されていない場合のみ、使用できます。
MAX(*expr*)		グループの中で、*expr* の列の最大値を戻す（*expr* は、文字または数値）。
MIN(*expr*)		グループの中で、*expr* の列の最小値を戻す（*expr* は、文字または数値）。
STDDEV_POP(*num_expr*)		グループに含まれる行の、*num_expr* の列の母標準偏差を計算する。
STDDEV_SAMP(*num_expr*)		グループに含まれる行の、*num_expr* の列の標本標準偏差を計算する。
SUM(*num_expr*)		グループに含まれる行の、*num_expr* の列の合計値を戻す。すべてヌルの場合、SUM() はヌルを戻します。
TOTAL(*num_expr*)	○	SUM(*num_expr*) と同じだが、TOTAL() は、すべてヌルの場合、0.0 を戻す。
VAR_POP(*num_expr*)		グループに含まれる行の、*num_expr* の列の母分散を計算する。
VAR_SAMP(*num_expr*)		グループに含まれる行の、*num_expr* の列の標本分散を計算する。

付録 B

参考文献

『スクリプト構文リファレンス』では、以下の文献を参考にしています。

Bentley, Jon Louis. (1975). "Multidimensional binary search trees used for associative searching." *Communications of the ACM* 18, 9:509–517.

Gander, W., and Gautschi, W. (2000). “Adaptive Quadrature–Revisited.” *BIT Numerical Mathematics* 40:84–101.

Moore, W. A., and Parks, D. R. (2012). “Update for the logicle data scale including operational code implementations.” *Cytometry Part A* 81A:273–277.

Uhlmann, Jeffrey (1991). "Satisfying General Proximity/Similarity Queries with Metric Trees". *Information Processing Letters*. 40 (4): 175–179.