CONTENTS

ビジュアルヘルプ – データフォルダー

データフォルダー

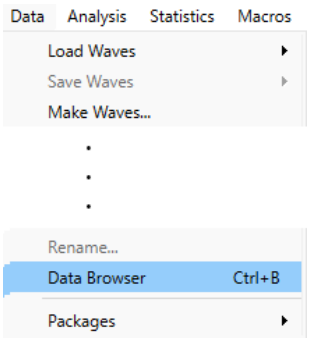
データフォルダーを使うと、エクスペリメント内のデータを階層的に保存することができます。
階層的な保存は、類似したデータセットが複数ある場合に役立ちます。
各データセットを個別のデータフォルダーに保存することで、データを整理しやすくなるだけでなく、名前の競合も防ぐことができます。

データフォルダーには、４種類のデータオブジェクトを含めることができます。

- ウェーブ
- 数値変数
- 文字列変数
- 他のデータフォルダー

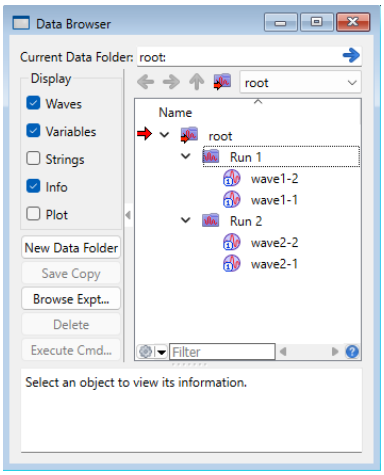
Igor Pro のデータフォルダーは、コンピューターの階層型ディスクファイルシステムと非常によく似ていますが、ディスク上ではなく、完全にメモリ内に存在するという点が異なります。
この類似点は、データフォルダーという概念を理解する上で役立ちますが、コンピューターのフォルダーやファイルと混同しないよう注意が必要です。

データフォルダーは、エクスペリメントを複数回実行する場合に特に役立ちます。
各実行分のデータを、それぞれ別のデータフォルダーに保存することができます。
データフォルダーには「run1」、「run2」などの名前を付けることができ、各データフォルダー内のウェーブ（wave）や変数（variable）の名前は、他のフォルダーと同じにして構いません。
つまり、実行に関する情報はデータフォルダー名にエンコードされるため、データオブジェクト自体の名前はすべての実行で同じにすることができます。
これを利用して、どの実行を処理しているかに関係なく、同じウェーブや変数名を使うプロシージャを作成することができます。



データフォルダーは、プロシージャの実行間においてプライベートデータを保存する必要があるプログラマーにとって非常に便利です。
データフォルダーのこの用途については、ヘルプ Managing Package Data で説明しています。

Data Browser ウィンドウでは、データフォルダーの階層構造を確認できます。
このウィンドウが表示されていない場合は、Data→Data Browser を選択します。



１つのデータフォルダーが「現在の」データフォルダーとして指定されます。
特定のデータフォルダーを明示的に指定しないコマンドは、現在のデータフォルダーに対して実行されます。
現在のデータフォルダーは、ブラウザーを使うか、GetDataFolder、GetDataFolderDFR 関数、SetDataFolder コマンドを使って確認、設定できます。

Data Browser は、階層を確認したり、現在のデータフォルダーを設定したりするだけでなく、次のような用途にも使用できます。

- 新しいデータフォルダーを作成

- オブジェクトの移動、複製、名前の変更、削除
- 他の Igor Pro エクスペリメントファイルを参照し、そこからデータをメモリに読み込む
- 現在のエクスペリメントのデータのコピーを、ディスク上のエクスペリメントファイルまたはフォルダーに保存する
- オブジェクトを選択し、情報ペインで変数、文字列、ウェーブの内容を表示、編集
- プロットペインが表示されている状態で、メインリストからウェーブを１つずつ選択し、1D または 2D のウェーブのシンプルなプロットを確認する
- 他の Igor Pro のエクスペリメントを閲覧しながら、ウェーブの簡単なグラフを確認する
- それぞれのアイコンをダブルクリックして、変数、文字列、ウェーブの内容を確認する
- １つのウェーブごとに、シンプルなヒストグラムやウェーブの統計情報を表示する

ダイアログでのウェーブの選択にも、同様のブラウザーが使われます。
詳細は、ヘルプ Dialog Wave Browser を参照してください。

データフォルダーを使う前に、必ず「データフォルダーの使用方法」のセクションをお読みください。

プログラマーは、ヘルプ Programming with Data Folders を読むべきです。

データフォルダーのシンタックス

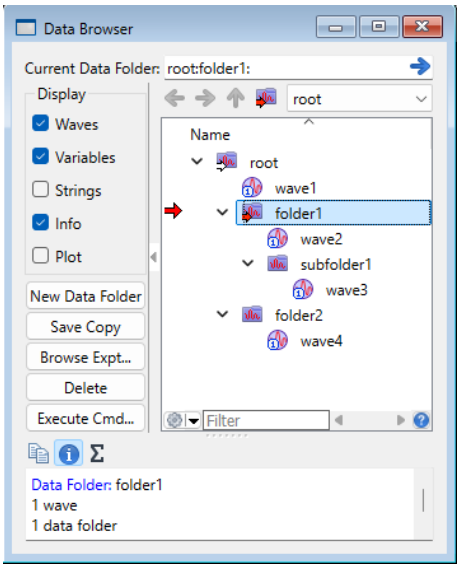
データフォルダーは、ウェーブや変数といった他のオブジェクトと同様に、名前付きのオブジェクトです。
データフォルダーの名前は、ウェーブ名と同じ規則に従います。
ヘルプ Liberal Object Names を参照してください。

Igor Pro のデータフォルダーでは、オブジェクトへのパスの構成要素を区切るためにコロン（：）が使われます。

「root」という名前のデータフォルダーは常に存在し、他のすべてのデータフォルダーが含まれています。

コマンド内で特定のオブジェクトを指定するには、次のようにします。

- フルパスを指定する
- 部分パスを使う
- オブジェクト名のみを使う



オブジェクト名のみを使用できるのは、そのオブジェクトが現在のデータフォルダー内にある場合に限られます。

フルパスは「root」で始まり、現在のデータフォルダーには依存しません。
部分パスは「:」で始まり、現在のデータフォルダーを基準としています。

右のようなデータフォルダーの構造があると仮定します。

また、現在のデータフォルダーが folder1（→ 記号で示されている）であると仮定します。
以下は、すべて有効な Display コマンドです。

```
Display wave2
Display :subfolder1:wave3
```

```
Display root:folder1:subfolder1:wave3
Display ::folder2:wave4
```

最後の例は、新しいルールを示しています。
コロンを複数回使うことで、階層を遡ることができます。
例えば、有効ではあるものの、ちょっとばかばかしい例を示します。

```
Display root:folder1:subfolder1:::folder2:wave4
```

場合によっては、データフォルダー内のオブジェクトではなく、データフォルダーそのものを指定する必要があることがあります。
その場合は、オブジェクト名を省略してください。
従って、パス指定の末尾にはコロンを付ける必要があります。
ただし、末尾のコロンを省略しても、通常、意図を正しく理解してくれます。

現在のデータフォルダーを指定する必要がある場合は、コロンを１つだけ使います。
例えば：

```
KillDataFolder :
```

これにより、現在のデータフォルダーとその中のすべての内容が削除され、「現在の」データフォルダーが現在のフォルダーの親フォルダーに設定されます。
プログラミングの知識がない方は、データブラウザーを使ってデータフォルダーを削除する方がよいかもしれません。

$ 演算子は、文字列式を単一の名前に変換することを思い出してください（ヘルプ String Substitution Using $ を参照）。
データフォルダーには名前が付けられているため、次のような記述は有効です。

```
String df1 = "folder1", df2="subfolder1"
Display root:$(df1):$(df2):wave3
```

これは些細な例ですが、もし df1 と df2 がプロシージャのパラメーターだった場合、役に立つでしょう。

この種の式では、必ず括弧を使う必要があることに注意してください。
これは、$ と : の演算子の優先順位の違いによるものです。

パスの先頭で使う場合、$ 演算子は特別な動作をするため、パス全体に対して使う必要があります。

```
String path1 = "root:folder1:subfolder1:wave3"
Display $path1
```

パス内でリベラル名を使う場合は、一重引用符で囲む必要があります。
例えば：

```
Display root:folder1:'subfolder 1':'wave 3'
String path1 = "root:folder1:'subfolder 1':'wave 3'"
Display $path1
```

ただし、単純な名前を文字列として渡す場合は、一重引用符を使ってはいけません。

```
Make 'wave 1'
String name
name = "'wave 1'"           // 正しくない
name = "wave 1"             // 正しい
Display $name
```

データフォルダーのコマンドと関数

ほとんどのユーザーは、データフォルダーを作成、表示、および操作するために Data Browser を使います。
以下の操作は主にプログラマーが使うものであり、プログラマーの方はヘルプ Programming with Data Folders を参照してください。

NewDataFolder *path*
SetDataFolder *path*
KillDataFolder *path*
DuplicateDataFolder *srcPath*, *destPath*
MoveDataFolder *srcPath*, *destPath*
MoveString *srcPath*, *destPath*
MoveVariable *srcPath*, *destPath*
MoveWave *wave*, *srcPath* [*newname*]
RenameDataFolder *path*, *newname*
Dir

以下は、データフォルダーで使われる関数です。

GetDataFolder(*mode* [, *dfr*])
CountObjects(*pathStr*, *objtype*)
GetIndexedObjName(*pathStr*, *type*, *index*)
GetWavesDataFolder(*wave*, *mode*)
DataFolderDir(*mode*)
DataFolderExists(*pathStr*)

データフォルダーの参照関数

プログラマーは、パスの代わりにデータフォルダー参照を利用することができます。
データフォルダー参照は、データフォルダーを直接参照する軽量なオブジェクトですが、一方、一連の名前で構成されるパスは、実際のターゲットフォルダーを見つけるために検索を行う必要があります。

以下は、データフォルダーへの参照を扱う関数です。

GetDataFolder(*mode* [, *dfr*])
GetDataFolderDFR( )
GetIndexedObjNameDFR(*dfr*, *type*, *index*)
GetWavesDataFolderDFR(*wave*)
CountObjectsDFR(*dfr*, *type*)
DataFolderRefChanges(*dfr*, *changeType*)
DataFolderRefStatus(*dfr*)
DataFolderRefsEqual(*dfr1*, *dfr2*)
NewFreeDataFolder( )

データフォルダー参照を使ったプログラミングの詳細は、ヘルプ Data Folder References を参照してください。

データフォルダーとコマンド

Igor Pro は通常、現在のデータフォルダーのコンテキストに基づいてコマンドを評価します。
つまり、特定のデータフォルダーへのパスが明示されていない限り、オブジェクト名は現在のデータフォルダー内の

オブジェクトを指します。
例えば：

```
Function Test()
        Make wave1
        Variable/G myGlobalVariable
End
```

この関数は、現在のデータフォルダー内に「wave1」と「myGlobalVariable」を作成します。
同様に、コマンド：

```
WaveStats wave1
```

は、現在のデータフォルダー内の wave1 に対する操作です。

データフォルダーとユーザー定義関数

ユーザー定義関数からグローバル変数やウェーブにアクセスするには、まず NVAR、SVAR、または WAVE 参照を作成する必要があります。
これらの参照は、グローバルオブジェクトを指すローカルオブジェクトです。
詳細については、ヘルプ Accessing Global Variables And Waves を参照してください。

データフォルダーとウィンドウ再作成マクロ

ウィンドウ再作成マクロは、キーワード「Window」で始まります。
これらは、グラフ、テーブル、その他の Igor Pro ウィンドウを再作成するために使われ、ヘルプ Saving a Window as a Recreation Macro で説明しています。

ウィンドウ再作成マクロは、ルートデータフォルダーをコンテキストとして評価されます。
つまり、Igor はウィンドウマクロの開始時に現在のデータフォルダーをルートに設定し、ウィンドウマクロの終了時に元の状態に戻します。

Macro または Proc キーワードで始まるマクロは、現在のデータフォルダーをコンテキストとしてコマンドを評価します。

この方法でウィンドウ再作成マクロを評価することで、現在のデータフォルダーに関係なくウィンドウが正しく再作成されることが保証され、データフォルダー機能がなかった以前のバージョンの Igor Pro で作成されたウィンドウマクロとの互換性もある程度確保されます。

つまり、ウィンドウ再作成マクロ内で、データフォルダーのパスを明示的に含まないオブジェクト名は、ルートデータフォルダー内のオブジェクトを指すことになります。

データフォルダーと代入文

ウェーブと変数の代入文は、その文の左辺にあるウェーブまたは変数が含まれるデータフォルダーのコンテキスト内で評価されます。

```
root:subfolder:wave0 = wave1 + var1
```

これは、次と同じ意味を簡潔に表したものです。

```
root:subfolder:wave0 = root:subfolder:wave1 + root:subfolder:var1
```

このルールは、代入演算子として “=” の代わりに “:=” を使う依存関係式にも適用されます。

データフォルダーとコントロール

ValDisplay コントロールは、ルートデータフォルダーのコンテキストで値式を評価します。

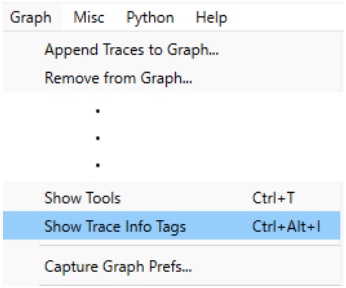
SetVariable コントロールは、コントロール対象のグローバル変数が存在するデータフォルダーを記憶しており、現在のデータフォルダーがコントロール対象の変数とは異なる場合でも、正常に機能し続けます。

コントロール の詳細は、ヘルプ Controls and Control Panels を参照してください。

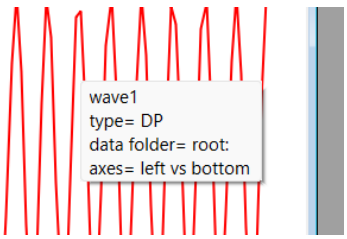
データフォルダーとトレース

グラフ上のトレースを見ただけでは、それがどのデータフォルダーにあるのかは判別できません。

トレースのウェーブがどのデータフォルダーにあるかを確認する最も簡単な方法は、トレース情報のヘルプを利用することです。
メニュー Graph→Show Trace Info Tags を選択し、トレースの上にマウスを合わせると、トレース情報が表示されます。



もう 1 つの方法は、Modify Trace Appearance ダイアログを使うことです。
トレースをダブルクリックすると、このダイアログが表示されます。
ダイアログ内の Traces リストには、各トレースのデータフォルダーのフルパスが表示されます。



最後に、グラフウィンドウの再表示マクロを作成して確認することができます。
詳細は、ヘルプ Saving a Window as a Recreation Macro を参照してください。

データフォルダーの使い方

データフォルダーはさまざまな用途に利用できます。
ここでは、データフォルダーの代表的な 2 つの使い方を紹介します。

ウェーブ、文字列、変数を隠す

高度なプロシージャでは、ユーザーが直接アクセスすることを意図していないグローバル変数、文字列、ウェーブを多数必要とする場合があります。
これらのプロシージャを作成するプログラマーは、他のデータフォルダー名と重複しないよう一意の名前を付けて作成したデータフォルダー内に、そのような項目をすべて格納しておく必要があります。

これらのプロシージャを使うユーザーは、現在のデータフォルダーを、生データと最終結果が保存されているデータフォルダーのままにしておく必要があります。
そうすることで、プロシージャのグローバル変数やウェーブがダイアログのリストを乱雑にするのを防ぐことができます。

プロシージャを作成するプログラマーは、ヘルプ Managing Package Data を参照してください。

類似したデータの分離

繰り返しテストを行う時に生じる問題の１つとして、各テスト実行のデータを他の実行データと区別して管理する必要があるという点が挙げられます。
多くの場合、各実行のデータは他の実行のデータと非常に似ており、同じ名前が付けられていることさえあります。データフォルダーがない場合、最初の実行以降は新しい名前を付ける必要が出てきます。

テスト実行ごとに１つのデータフォルダーを作成することで、１回の実行に関連するすべてのデータをそれぞれのフォルダーにまとめることができます。
他の同名のデータは別のデータフォルダーにあるため、データに同じ名前を付けても問題ありません。

データフォルダーを使うことで、各ダイアログに表示されるデータや、データフォルダー名を指定せずにコマンドで使用できるデータは、現在のデータフォルダー内のデータのみとなるため、異なる実行結果のデータが誤って混在することを防ぐことができます。

WaveMetrics が提供する「Multipeak Fitting」のエクスペリメントのプロシージャは、次のように機能します。個別のピーク曲線カーブフィッティングの実行結果とグローバルな状態情報を格納するためのデータフォルダーを作成します。

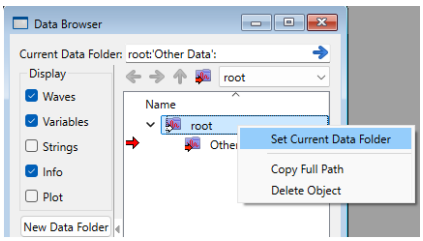
データフォルダーの使い方の例

この例では、データフォルダーを使って以下の手順を追ってみます。

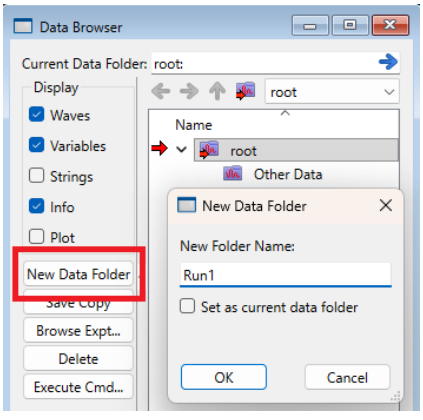
- ２回のテスト実行から取得したデータを、それぞれ別のデータフォルダーに読み込む
- 各テストの実行結果を個別に示すグラフを作成する
- ２回のテスト実行結果を比較するグラフを作成する

まず、Data Browser を使って、各テスト実行ごとにデータフォルダーを作成します。

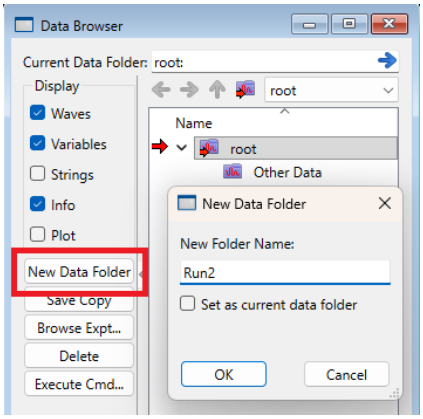
1. Data Browser を開き、root アイコンを右クリックして Set as Current Data Folder を選択し、現在のデータフォルダーをルートに設定します。
（ほかにデータフォルダーがない場合は、メニューは選択できなくなっています）



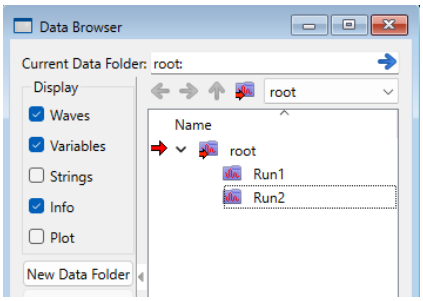
2. ルートデータフォルダーをクリックし、New Data Folder ボタンをクリックします。
新しいデータフォルダーの名前に「Run1」と入力し、OK をクリックします。



3. もう一度 New Data Folder をクリックします。
新しいデータフォルダーの名前に「Run2」と入力し、OK をクリックします。

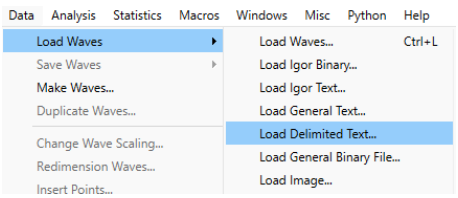


4. Data Browser ウィンドウは、右のような表示になるはずです。

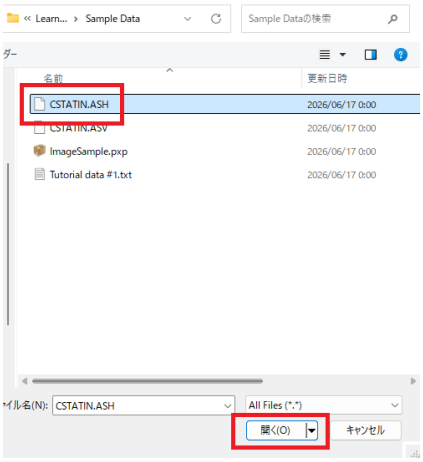


それでは、Run1 から順に、各データフォルダーにサンプルデータを読み込んでいきます。

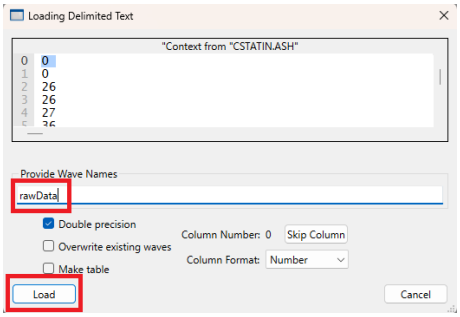
5. 現在のデータフォルダーを「Run1」に設定し、メニュー Data→Load Waves→Load Delimited Text を選択します。



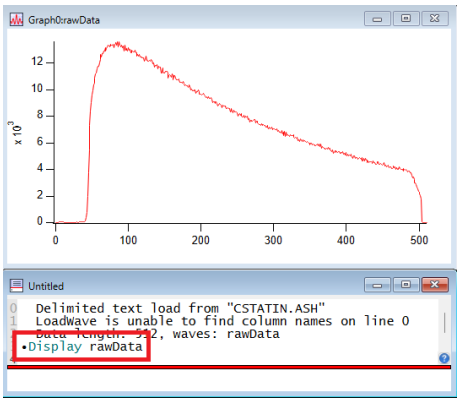
6. 「Learning Aids」フォルダー内の「Sample Data」サブフォルダーから「CSTATIN.ASH」ファイルを選択し、「開く」をクリックします。



7. 表示される Loading Delimited Text ダイアログで、読み込むウェーブに「rawData」という名前を付け、「Load」をクリックします。
このデータを Run 1 の結果であると仮定します。

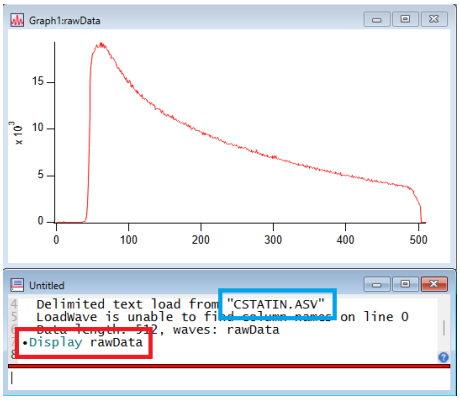


8. コマンドラインに「Display rawData」と入力して、データをグラフ化します。

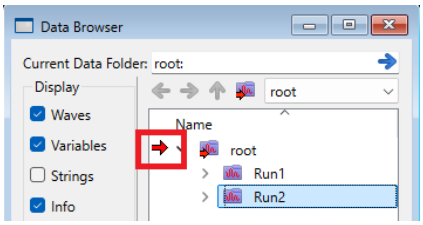


9. 現在のデータフォルダーを「Run2」に設定し、ウェーブ読み込みの手順を繰り返して、代わりに CSTATIN.ASV ファイルを選択します。
表示される Loading Delimited Text ダイアログで、読み込んだウェーブに「rawData」という名前を付けます。
このデータを Run 2 の結果であると仮定します。
「Display rawData」コマンドを繰り返して、このデータのグラフを作成します。

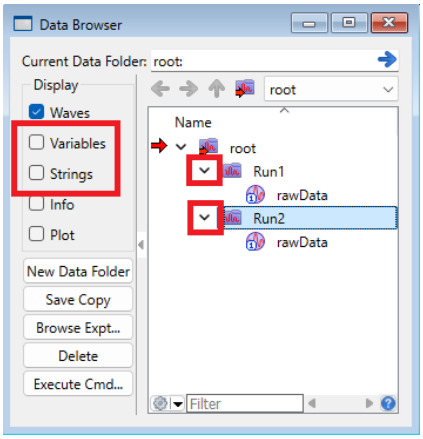
   読み込んだデータには同じ名前を使っている点に注目してください。
   もう一方の `rawData` ウェーブは別のデータフォルダーにあるため、競合は発生しません。



10. Data Browser で、現在のデータフォルダーをルートに設定します。

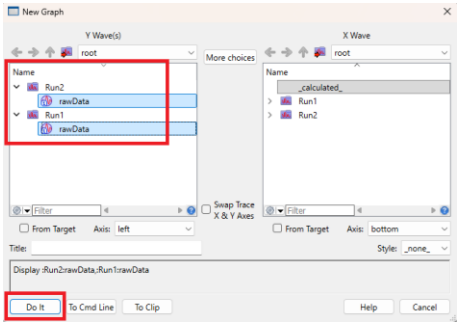


11. Data Browser の Display セクションで、Variables と Strings のチェックボックスのチェックを外します。
「Run1」と「Run2」のアイコンの横にある展開アイコンをクリックして、各アイコンを展開します。
この時点で、Data Browser は右のような表示になっているはずです。

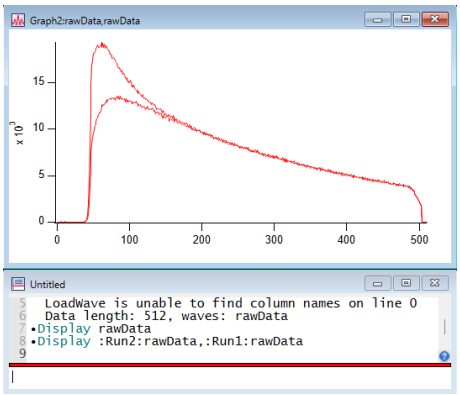


12. rawData のウェーブを両方とも表示したグラフを簡単に作成し、より詳しく比較することができます。

    メニュー Windows→New Graph を選択して、New Graph ダイアログを表示します。
    ダイアログのウェーブブラウザーコントロール（ヘルプ Dialog Wave Browser を参照）を使って、「root:Run1:rawData」と「root:Run2:rawData」の両方を選択します（Shift キーを使います）。
    「Do It」をクリックします。



13. 右のようなグラフが表示されます。

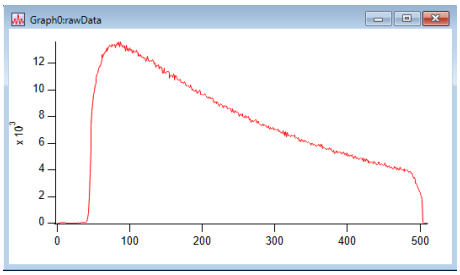


現在のデータフォルダーを任意のものに変更しても、グラフには引き続き同じデータが表示されます。
グラフは、各ウェーブがどのデータフォルダーに属しているかを記憶しており、グラフ再作成マクロも同様です。
これは多くの場合望ましい動作ですが、必ずしもそうとは限りません。

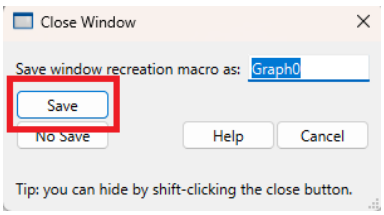
エクスペリメントに多数のテスト実行があり、それぞれが個別のデータフォルダーにきちんと整理されているとします。
そこで、各テスト実行のデータフォルダーからのデータのみを表示する単一のグラフを使って、各テスト実行を確認したいとします。
別のテスト実行を確認する時には、その別のデータフォルダーからのデータのみがグラフに表示されるようにしたいのです。

さらに、グラフの特性を同じにしたい場合（軸のラベル、注釈、線のスタイルや色などが同じになるように）は、次のようにすることができます。

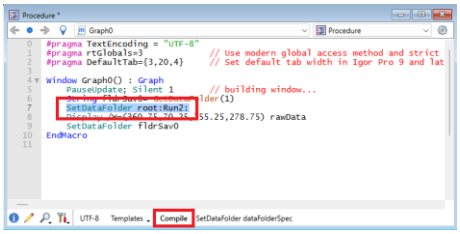
14. 最初のテスト実行のグラフを作成します。



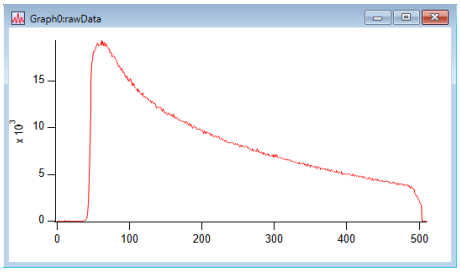
15. ウィンドウを Kill し、グラフウィンドウのマクロを保存します。



16. メニュー Windows→Procedure Windows→Procedure Window を選択し、別のデータフォルダー内のデータを参照するように、再作成マクロを編集します（詳細は下記）。
Compile ボタンをクリックします。



17. メニュー Windows→Graph Macros→Graph0 を選択し、編集した再生マクロを実行します。



再作成されたグラフは見た目は同じですが、別のデータフォルダー内のデータが使われます。
編集の時は、通常、次のようなコマンドを変更することになります。

```
SetDataFolder root:Run1:
```

を

```
SetDataFolder root:Run2:
```

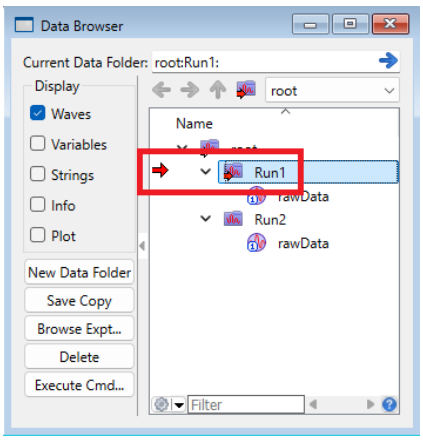
に変更します。

グラフに複数のデータフォルダーからのウェーブが表示される場合は、次のようにコマンドを編集する必要があるかもしれません。

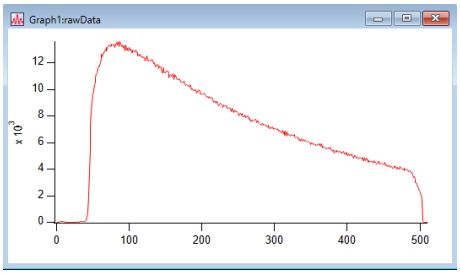
```
Display rawData,::Run1:rawData
```

ただし、再作成マクロを編集する必要のない別の方法もあります。
それは、ReplaceWave コマンドを使って、グラフ内のウェーブを別のフォルダーにあるウェーブに置き換える方法です。
その手順は次のようになります（Run1 から Run2 へ置き換え）。

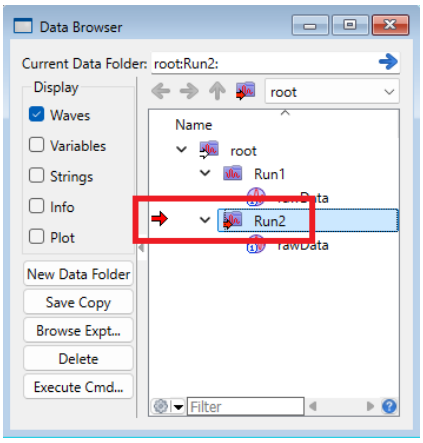
1. 表示したいウェーブファイルが含まれているフォルダーを、現在のデータフォルダーとして設定します。



2. 目的のグラフを有効にします。
（右のグラフは Run1 のグラフ）

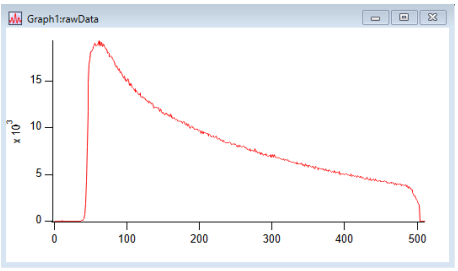


3. 表示したいウェーブファイルが含まれているフォルダーを、現在のデータフォルダーとして設定します。

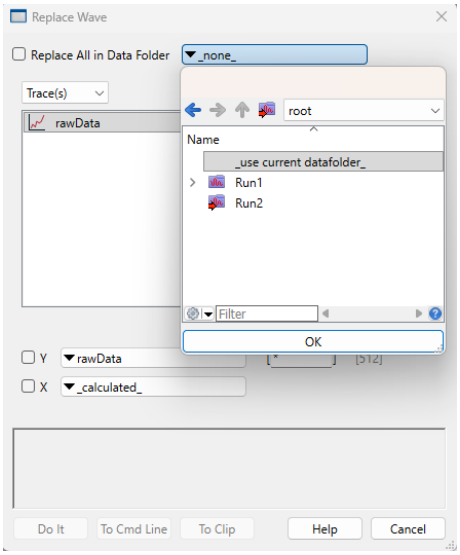


4. コマンドラインで次のコマンドを実行します：

```
ReplaceWave allinCDF
```



これにより、グラフ内のすべてのウェーブが、現在のデータフォルダー内に同名のウェーブが存在する場合、そのウェーブに置き換えられます。
詳細は、ReplaceWave コマンドを参照してください。
Graph→Replace Wave を選択すると、ダイアログが表示され、そこで対話形式で ReplaceWave コマンドを生成することができます。



データフォルダーごとに１つのウェーブしかありませんが、試してみることはできます：

① 現在のデータフォルダーを「Run1」に設定します。

② Run2 のデータのみが表示されたグラフを選択します（CSTATIN.ASV）。

③ コマンドラインで次のコマンドを実行します：

```
ReplaceWave allinCDF
```

グラフが更新され、Run1 の rawData のウェーブが表示されるようになりました。

データフォルダーの使用例を例えばもう１つ確認するには、File→Example Experiments→Tutorials→Data Folder Tutorial を選択してください。

Data Browser

Data Browser を使うと、データフォルダーの階層を閲覧したり、ウェーブのプロパティや数値変数・文字列変数の値を確認したり、他のエクスペリメントからデータオブジェクトを読み込んだり、現在のエクスペリメントのデータのコピーをディスク上のエクスペリメントファイルやフォルダーに保存したりすることができます。

ブラウザーを開くには、Data メニューから Data Browser を選択してください。

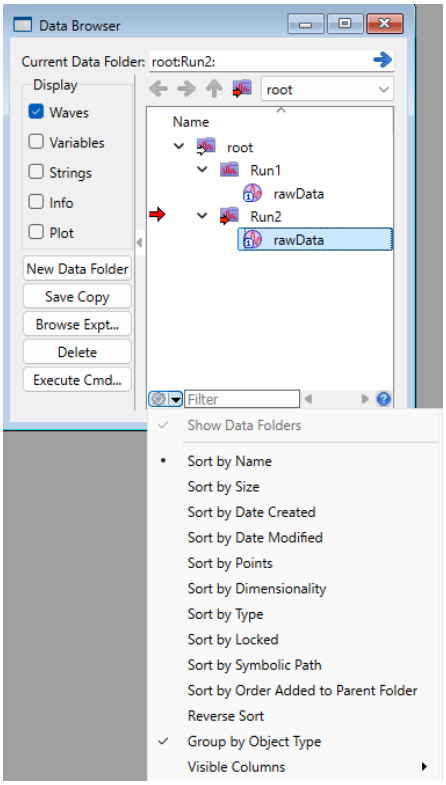
Data Browser のインターフェイスは、コンピューターのデスクトップと似ています。
Igor Pro の基本的なデータオブジェクト（変数、文字列、ウェーブ、データフォルダー）はアイコンで表され、現

在のエクスペリメントにおける階層構造に基づいてメインリストに配置されます。
また、このブラウザーには、ユーザーに追加機能を提供するいくつかのボタンも備わっています。

Data Browser ウィンドウの主な構成要素は次のとおりです。

- データフォルダー、ウェーブ、変数を表すアイコンを表示するメインリスト
- メインリストに表示されるオブジェクトの種類をコントロールする Display チェックボックス
- データ階層を操作するためのボタン
- 選択した項目に関する情報を表示する情報ペイン
- 選択したウェーブのグラフを表示するプロットペイン

メインリスト



Data Browser を最初に起動した時、メインリストがその画面の大部分を占めます。

データツリーの最上位には root データフォルダーがあり、デフォルトでは展開された状態で表示されます。
データフォルダーのアイコンをダブルクリックすると、ツリーの表示を変更し、root ではなく選択したフォルダーを最上位のデータフォルダーとして表示させることができます。
メインリストの上にあるポップアップメニューを使うと、現在の最上位データフォルダーを、階層内の別のデータフォルダーに変更することができます。

最上位のデータフォルダーの下には、そのフォルダーに含まれるすべてのデータオブジェクトが表示されます。
オブジェクトはタイプごとにグループ化されており、デフォルトでは作成順に一覧表示されます。
歯車アイコンをクリックすると、並べ替え順を変更できます。
また、フィルター入力欄を使用して、一覧を絞り込むこともできます。

マウスを使ってデータオブジェクトを選択できます。
Shift キーを押しながらクリックすると、連続した複数のデータオブジェクトを選択できます。
Ctrl キーを押しながらクリックすると、連続していない複数のデータオブジェクトを選択できます。

オブジェクトは、折りたたまれたデータフォルダーの中に隠れていても、選択されたままになります。
あるウェーブを選択し、そのデータフォルダーを折りたたんだ状態で、Shift キーを押しながら別のウェーブを選択し、それを別のデータフォルダーにドラッグすると、両方のウェーブがそこに移動します。
ただし、関連する Display チェックボックスの選択を解除して選択されたオブジェクトを非表示にした場合、そのオブジェクトに対しては削除や複製などの操作は行われません。

データオブジェクトの名前を変更するには、そのオブジェクトの名前をクリックして、名前を編集してください。

Data Browser では、アイコンをドラッグしてデータオブジェクトをあるデータフォルダーから別のデータフォルダーへ移動またはコピーすることもできます。
データオブジェクトをドラッグすることで、あるデータフォルダーから別のデータフォルダーへ移動できます。
ドラッグ中に Alt キーを押すと、データオブジェクトをあるデータフォルダーから別のデータフォルダーへコピーできます。

データフォルダー内のデータオブジェクトは、Edit メニューから Duplicate を選択するか、Ctrl+D キーを押すことで複製できます。

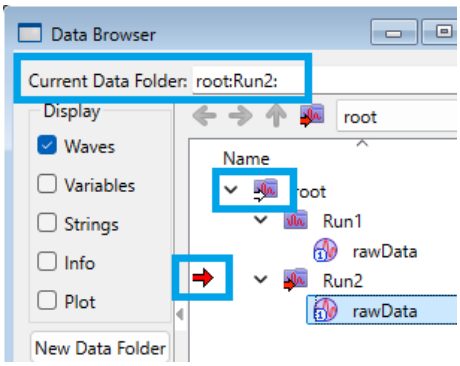
選択したすべてのデータオブジェクトのフルパス（必要に応じて引用符で囲んだもの）を、コマンドライン上にオブジェクトをドラッグすることで、コマンドラインにコピーすることができます。

リストから１つまたは複数のウェーブを選択し、既存のグラフまたはテーブルウィンドウにドラッグすることで、対象のウィンドウにそれらを追加することができます。
詳細は、ヘルプ Appending Traces by Drag and Drop を参照してください。

現在のデータフォルダー

「現在のデータフォルダー」とは、Igor Pro がデフォルトで、新しく作成された変数、文字列、ウェーブ、その他のデータフォルダーを保存するために使うデータフォルダーのことです。
データフォルダーを作成またはアクセスするコマンドは、データフォルダーのパスを指定して別のデータフォルダーを指定しない限り、現在のデータフォルダー内で実行されます。

メインリストの上部には、現在のデータフォルダーのフルパスを表示するテキストボックスがあります。
メインリストには、現在のデータフォルダーのアイコンの上に赤い矢印が表示されます。
現在のデータフォルダーが別のデータフォルダー内に含まれている場合、現在のデータフォルダーの親フォルダーのアイコンの上に白い矢印インジケーターが表示されます。



現在のデータフォルダーを設定するには、任意のデータフォルダーを右クリックし、Set Current Data Folder を選択します。
また、赤い矢印をドラッグするか、Alt キーを押しながらデータフォルダーのアイコンをクリックすることで、現在のデータフォルダーを設定することもできます。

Display チェックボックス

Display チェックボックスグループを使うと、メインリストに表示されるオブジェクトの種類をコントロールできます。
データフォルダーは常に表示されます。
また、このグループでは、情報ペインやプロットペインの表示／非表示を切り替えることもできます。

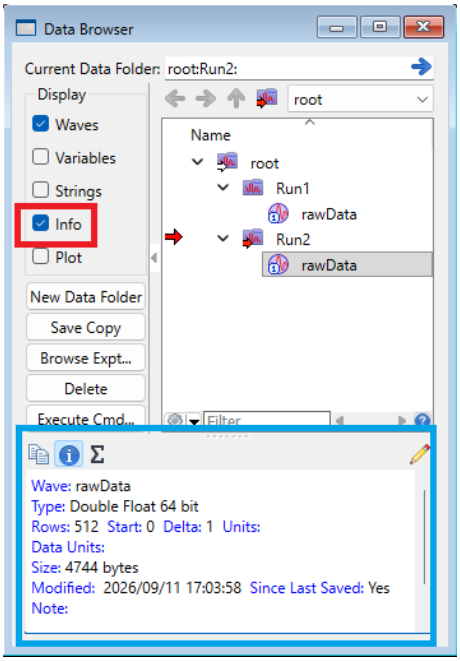
情報ペイン

情報ペインを表示するには、Info チェックボックスをオンにします。

情報ペインはメインリストの下に表示されます。
メインリストでオブジェクトを選択すると、そのプロパティや内容が情報ペインに表示されます。
例えば、変数を選択すると、その名前と値が情報ペインに表示されます。

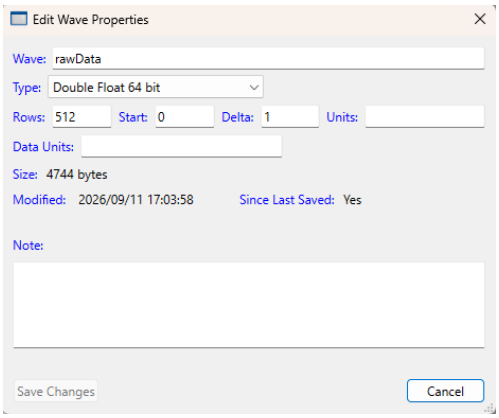
メインリストでウェーブを選択すると、情報ペインに、そのウェーブのタイプ、サイズ、次元、単位、開始 X 座標、デルタ X、注釈など、さまざまなプロパティが表示されます。
これらの各フィールドは、青いラベルとその後にプレーンテキストの値が続く形式で表示されます。
表示するプロパティは、Miscellaneous Settings ダイアログの Data Browser カテゴリで設定できます。

データフォルダーを選択すると、情報ペインにそのデータフォルダーに関するメモが表示される場合があります。
詳細は、「データフォルダーのメモ」のセクションを参照してください。



情報ペインの右上隅にある Edit Object Properties ボタンをクリックすると、数値変数や文字列変数の名前や値、あるいはウェーブの名前やプロパティを編集できます。
このボタンは、１つのオブジェクトのみが選択されている場合にのみ表示されます。
クリックすると、Edit Object Properties ダイアログが表示されます。

情報ペインでは、選択したウェーブの統計情報を表示することもできます。
ウェーブの統計情報を表示するには、情報ペインの上部にある「Σ」アイコンをクリックします。
通常モードに戻すには、「i」アイコンをクリックします。

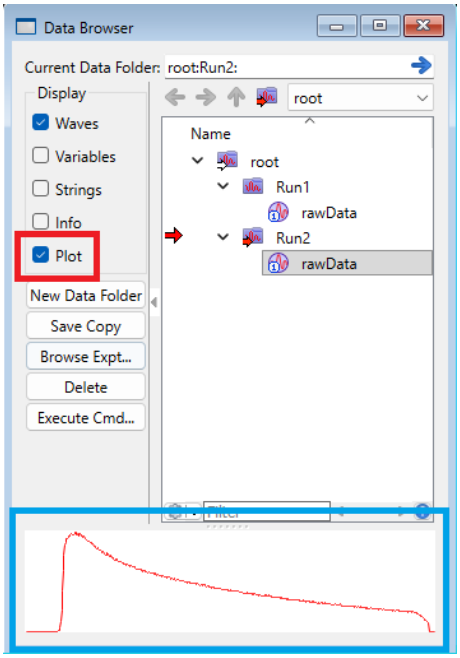


情報ペインの上部にあるクリップボードボタンをクリックすると、情報ペインに表示されているテキストをクリップボードにコピーできます。

情報ペインの表示に関するさまざまな設定は、右クリックして Settings サブメニューから項目を選択することで変更できます。
設定項目には、空白の表示方法、括弧の対応付け、構文の色付け、行の折り返しなどが含まれます。

プロットペイン

プロットペインを表示するには、Plot チェックボックスをオンにしてください。



プロットペインには、選択された１つのウェーブがグラフィカルに表示されます。
プロットペインは、メインリストと情報ペインの下に配置されています。
その上にあるメインリストで選択されたウェーブの小さなグラフまたは画像が表示されます。
プロットペインには、テキストウェーブや複数のウェーブが同時に表示されることはありません。

プロットペインのさまざまな設定は、右クリックして表示されるポップアップメニューから変更できます。

ポップアップメニューから Show Axes を選択すると、軸の表示／非表示を切り替えることができます。
また、Show Plot または Show Histogram を選択することで、選択したウェーブのプロットと１次元ヒストグラムのプロットを切り替えることもできます。

単純な 1D 実数ウェーブは、白い背景の上に赤色で描画されます。
複素数ウェーブは２本のトレースとして描画され、実部は赤色、虚部は青色で表示されます。
トレースのモード、線のスタイル、線の太さ、マーカー、マーカーのサイズ、色は、すべてポップアップメニューを使って設定できます。

2D ウェーブは画像として表示され、デフォルトではプロットペインのサイズに合わせて拡大・縮小され、Rainbow カラーテーブルが使われます。
アスペクト比やカラーテーブルは、ポップアップメニューから変更できます。

デフォルトでは、画像は左軸が反転して表示されます。
つまり、NewImage コマンドの挙動と同様に、Y 値は上から下に向かって増加します。
ポップアップメニューから Reverse Left Axis を選択することで、この表示を無効にすることができます。

メインリストで 3D または 4D のウェーブを選択すると、プロットペインにはそのウェーブのレイヤーが１つずつ表示されます。
プロットペイン上部のコントロールを使って、表示するレイヤーを選択できます。
再生ボタンを使ってプロットペインでレイヤーを順次切り替えるように設定でき、停止ボタンでその切り替えを停止できます。

選択したウェーブの次元が、RGB または RGBA データを含むと解釈できる場合、Automatically Detect RGB(A) チェックボックスが表示されます。
このチェックボックスをオンにすると、そのデータを RGB(A) 合成画像として表示します。
オンにしない場合は、2D データと同様に、カラーテーブルを使って画像データが表示されます。

New Data Folder ボタン

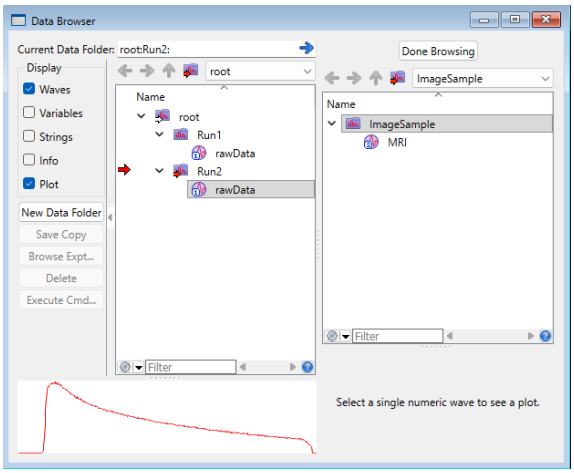
New Data Folder ボタンをクリックすると、現在のデータフォルダー内に新しいデータフォルダーが作成されます。
Data Browser には、新しいデータフォルダーの名前を入力するためのシンプルなダイアログが表示され、入力された名前が有効かどうかがチェックされます。
ダイアログに自由形式のオブジェクト名を入力する時は、名前の前後を単一引用符で囲まないでください。

Browse Expt ボタン

Browse Expt ボタンをクリックすると、パックされたエクスペリメントファイルまたはディスク上のフォルダーからデータオブジェクトが読み込まれ、現在のエクスペリメントに追加されます。

パックされたエクスペリメントファイルを閲覧するには、Browse Expt をクリックします。
Data Browser に Open File ダイアログが表示され、そこから閲覧したいパックされたエクスペリメントファイルを選択します。

ディスク上のフォルダー（パックされたエクスペリメントファイル、スタンドアロンの Igor Wave（.ibw）ファイル、その他のフォルダーが含まれている場合があります）を参照するには、Alt キーを押しながら Browse Expt をクリックしてください。
Data Browser に Choose Folder ダイアログが表示されるので、そこから参照するフォルダーを選択してください。



パックされたエクスペリメントファイルまたは参照するフォルダーを選択すると、Data Browser のメインリストの右側に追加のリストが表示されます。
右側のリストには、参照用に選択したファイルまたはフォルダー内のデータを表すアイコンが表示されます。
データを現在のエクスペリメントに読み込むには、右側のリストでアイコンを選択し、左側のリストにあるデータフォルダーにドラッグします。

Done Browsing ボタンをクリックすると、右側のリストが閉じられ、通常の操作モードに戻ります。

Save Copy ボタン

Save Copy ボタンをクリックすると、現在のエクスペリメントのデータオブジェクトが、ディスク上のパックされたエクスペリメントファイルに、あるいは個別のファイルとしてディスク上のフォルダーにコピーされます。
現在のエクスペリメントを保存するとデータも自動的に保存されるため、ほとんどのユーザーはこの操作を行う必要はありません。

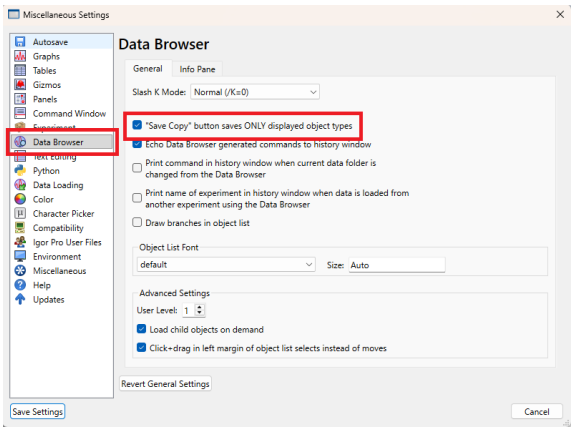
Save Copy をクリックする前に、保存したい項目を選択します。
Save Copy をクリックすると、Data Browser にダイアログが表示され、保存されたデータのコピーを含むパックされたエクスペリメントファイルの名前と保存先を指定します。

Save Copy をクリックする時に Alt キーを押すと、Data Browser にダイアログが表示され、データをパックされていない形式で保存するディスク上のフォルダーを選択できるようになります。
パックされていない形式は、特殊な用途を持つ上級ユーザーを対象としています。

展開せずに保存する時、Data Browser は「ミックスイン」モードを使います。
これは、保存されるアイテムが選択したディスク上のフォルダーに書き込まれるものの、ディスク上にすでに存在するアイテムには影響しないことを意味します。
ただし、保存されるアイテムと競合する場合は、既存のアイテムが上書きされます。

展開せずに保存する時、Data Browser でデータフォルダーを選択すると、そのフォルダーとその中身のすべてがディスクに書き込まれます。
ディスク上のフォルダー名は、選択したデータフォルダー名と同じになります。
ただし、ルートデータフォルダーについては「Data」という名前で書き込まれます。
データフォルダーを選択せず、いくつかのウェーブなどのアイテムのみを選択した場合、Data Browser はそれらのアイテムに対応するファイルを書き込みますが、ディスク上のフォルダーは作成しません。

デフォルトでは、Data Browser の Display セクションにある Waves、Variables、Strings の各チェックボックスの状態にかかわらず、オブジェクトが出力に書き込まれます。
ただし、Miscellaneous Settings ダイアログの Data Browser カテゴリには、この動作を変更するプレファレンスがあります。
Save Copy saves ONLY displayed object types チェックボックスをオンにすると、Save Copy は、対応する Display チェックボックスがオンになっている場合にのみ、特定のタイプのオブジェクトを書き込みます。



Data Browser には、ディスク上のパックされたエクスペリメントファイルに対してデータを追加または削除する機能はありません。
既存のファイルを上書きすることしかできません。
データを追加または削除するには、エクスペリメントファイルを開き、追加や削除を行った上で、エクスペリメントを保存する必要があります。

Delete ボタン

メインリストでデータオブジェクトが選択されているときは、Delete ボタンが有効になります。
このボタンをクリックすると、Data Browser に、削除される項目の数が記載された警告が表示されます。

警告を表示させないようにするには、Delete ボタンをクリックする時に Alt キーを押してください。

> **注意**
> このスキップ機能は、細心の注意を払って使ってください。誤って何かを削除してしまった場合、エクスペリメント全体を最後に保存された状態に復元する以外、その操作を取り消すことはできません。

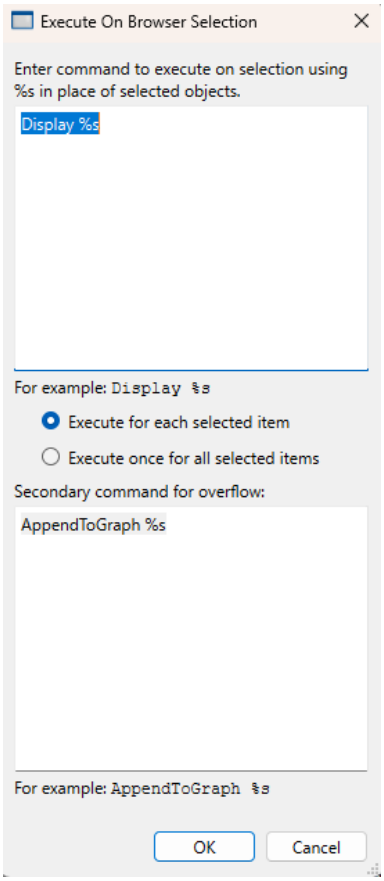
ルートデータフォルダーが選択されている状態で Delete ボタンをクリックすると、現在のエクスペリメント内のすべてのデータオブジェクトが削除されます。

> **注意**
> 誤って何かを削除してしまった場合、エクスペリメント全体を最後に保存された状態に戻す以外には、その操作を取り消すことはできません。

グラフやテーブルで使われているウェーブなど、削除できないオブジェクトを削除しようとすると、一部のオブジェクトを削除できなかったことを知らせる警告メッセージを表示します。
このダイアログの Show Details ボタンをクリックすると、削除されなかったオブジェクトの一覧を確認できます。

Execute Cmd ボタン

Execute Cmd ボタンは、データブラウザウィンドウで選択した項目に対してコマンドを実行するためのショートカットです。
Execute Cmd ボタンをクリックすると、実行するコマンドと実行モードを指定できるダイアログを表示します。
コマンドを設定したら、Alt キーを押しながらボタンをクリックすることで、ダイアログをスキップしてコマンドを直接実行することができます。



このコマンドの構文は、他の Igor Pro コマンドと同じですが、選択範囲を挿入する場所に「%s」を使う点が異なります。
例えば：

```
Display %s
```

実行するコマンドが Igor Pro コマンドの最大長を超える場合、セカンダリコマンドを指定することができます。
例えば：

```
AppendToGraph %s
```

「Execute for each selected item」が有効になっている場合、選択された各項目についてプライマリコマンドを 1 回ずつ実行し、%s の代わりにその項目のフルパスを代入します。

```
Display root:wave0
Display root:wave1
Display root:wave2
```

「Execute once for all selected items」が有効になっている場合、次のように、メインコマンドを 1 回実行します。

```
Display root:wave0, root:wave1, root:wave2
```

コマンドがコマンドラインの最大長を超える場合、次のように、プライマリコマンドとセカンダリコマンドを実行します。

```
Display root:wave0, root:wave1
AppendToGraph root:wave2
```

Data Browser のコマンド文字列に関する制限事項

Execute Cmd ダイアログで設定されたコマンド文字列、ModifyBrowser コマンドを介して設定されたコマンド文字列は、最大コマンド長である 2500 バイト未満でなければならず、printf、sprintf、または sscanf を含んではなりません。

選択したすべてのオブジェクトに対して一括でコマンドを実行する場合、プライマリおよびセカンダリのコマンド文字列には、「%s」トークンを最大 1 つまで含めることができます。

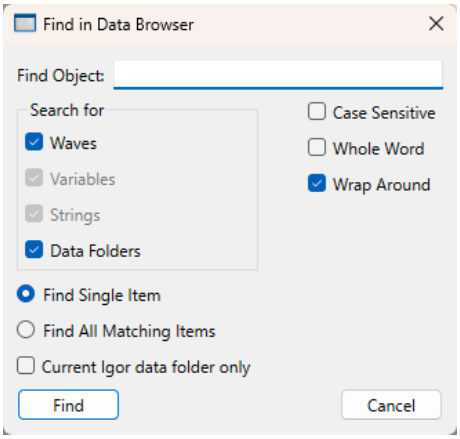
コマンドを実行しても、Data Browser ウィンドウを閉じてはなりません。

コマンド文字列の仕組みは便利な場合もありますが、やや不格好な面もあります。
GetBrowserSelection 関数を使って選択されたオブジェクトのリストを取得し、通常のリスト処理手法を用いてそのリストを処理する方が、よりすっきりとしていて透明性が高くなります。

Data Browser ダイアログの検索の使用方法

Edit→Find を選択すると、Data Browser は Find in Data Browser にダイアログを表示します。
このダイアログを使うと、サブデータフォルダーの中に埋もれている可能性のあるウェーブ、変数、データフォルダーを検索できます。
また、検索文字列に基づいて、複数の項目を一度に選択する便利な方法も提供されています。
その後、Execute Cmd ボタンを使って、選択した項目に対して操作を行うことができます。

Find in Data Browser ダイアログでは、名前の一致判定に使うテキストを指定できます。
名前に指定したテキストが含まれるオブジェクトはすべて、一致するとみなされます。



また、ワイルドカード文字 ”*” を使うと、文字の種類に関係なく、0 個以上の文字に一致させることができます。
例えば、” w*3” は “wave3” や “w3”、さらには “” にも一致します。

このダイアログでは、大文字と小文字を区別するかどうか、単語単位の検索を行うかどうか、および検索範囲をループさせるかどうかを指定することもできます。

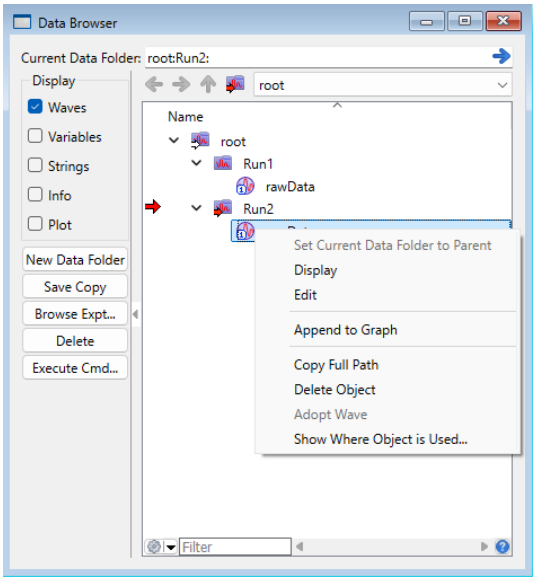
Data Browser のポップアップメニュー

データブラウザーでオブジェクトを選択し、右クリックして表示されるポップアップメニューから操作を選択することで、オブジェクトにさまざまな操作を適用できます。

Display と New Image のポップアップ項目を使うと、選択したウェーブを新しいグラフや画像として作成できます。
同じデータフォルダー内または異なるデータフォルダー内のウェーブを複数選択し、それらを同じグラフにまとめて表示することができます。

Copy Full Path は、選択したオブジェクトのデータフォルダーのフルパスを、必要に応じて引用符で囲んだ状態で、クリップボードにコピーします。

Show Where Object Is Used を選択すると、選択したオブジェクトが使われている依存関係の一覧、ウェーブの場合は、そのウェーブが使われているウィンドウの一覧が表示されるダイアログが表示されます。
この項目は、データオブジェクトが 1 つだけ選択されている場合にのみ利用可能です。



Adopt Wave は、共有されているウェーブを現在のエクスペリメントに取り込みます。

Data Browser のプレファレンス

Data Browser の各種設定は、Miscellaneous Settings ダイアログの Data Browser カテゴリでコントロールされます。
これにアクセスするには、Misc→Miscellaneous Settings を選択し、左側のリストにある Data Browser をクリックします。

Data Browser のプログラミング

Data Browser は、以下のコマンドと関数を使って、Igor Pro プロシージャからコントロールすることができます。

CreateBrowser, ModifyBrowser, GetBrowserSelection, GetBrowserLine

上級ユーザーは、GetBrowserSelection 関数を使って、Data Browser を入力デバイスとして利用できます。
例えば、File→Example Experiments→Tutorials→Data Folder Tutorial を参照してください。

Data Browser ダイアログをモーダルとして使う方法

Data Browser をモーダルダイアログとして使うことで、ユーザーがプロシージャで処理する 1 つまたは複数のオブジェクトを対話形式で選択できるようにすることができます。
このモーダル Data Browser は、Data→Data Browser を選択して呼び出す通常のデータブラウザーとは別のものです。
モーダル Data Browser は通常、プロシージャでユーザーからの入力を求める間だけ、短時間表示されます。

モーダル Data Browser を作成・表示するには、 2 つの方法があります。
具体的な例については、CreateBrowser コマンドのヘルプに記載されています。

最初で最も簡単な方法は、プロンプトキーワードを指定して CreateBrowser コマンドを使うことです。
必要に応じて、CreateBrowser でサポートされている他のキーワードを任意に使うこともできます。

２つ目の方法は、/M フラグを指定して CreateBrowser コマンドを使うことです。
これにより、モーダル Data Browser が作成されますが、表示はされません。
その後、/M フラグを指定して ModifyBrowser を呼び出し、モーダルブラウザーを任意の設定に構成します。
ブラウザーを表示する準備ができたら、/M フラグと showModalBrowser キーワードを指定して ModifyBrowser を呼び出します。

どちらの方法でも、ユーザーが「OK」または「キャンセル」をクリックするか、閉じるボタンを使ってモーダルブラウザーを閉じると、ブラウザーは V_Flag と S_BrowserList 変数を設定します。
ユーザーが「OK」をクリックすると、V_Flag は 1 に設定され、S_BrowserList には、選択された項目のフルパスがセミコロン区切りでリストとして格納されます。
ユーザーが「キャンセル」または閉じるボックスをクリックした場合、V_Flag は 0 に設定され、S_BrowserList は “” に設定されます。

Data Browser は、必要に応じて引用符で囲んだフルパスを S_BrowserList に格納します。
各フルパスの後にはセミコロンが続きます。
StringFromList 関数を使うと、フルパスを１つずつ抽出することができます。

プロシージャから CreateBrowser を実行した場合、出力変数はローカル変数となります。
コマンドラインからモーダル Data Browser を作成する理由はありませんが、万が一作成した場合、出力変数はグローバル変数となり、CreateBrowser コマンドが呼び出された時点の現在のデータフォルダー内に作成されます。

ユーザーは、モーダルブラウザーで現在のデータフォルダーを変更することができます。
ただし、ほとんどの場合、これは望ましくありません。
CreateBrowser コマンドの例では、現在のデータフォルダーを保存および復元する方法について説明しています。

ユーザーが「OK」ボタンをクリックすると、Data Browser は、CreateBrowser または ModifyBrowser に対して command1 と command2 キーワードを使って指定したコマンドを実行します。
これらのキーワードを省略した場合、コマンドは実行されません。

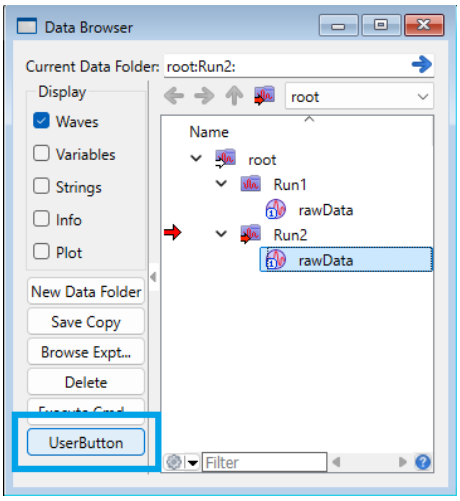
Data Browser のユーザー定義ボタンの管理

ユーザー定義ボタンを使うと、作業環境をカスタマイズしたり、頻繁に使う操作にすばやくアクセスすることができます。
ボタンの追加には ModifyBrowser appendUserButton を、ボタンの削除には ModifyBrowser deleteUserButton を使います。
appendUserButton キーワードを使うと、ボタン名や、ボタンをクリックしたときに実行するコマンドを指定することができます。

例えば、

```
ModifyBrowser appendUserButton={UserButton, "Display %s"}
```

を実行すると、右のようにボタンが追加されます。



ユーザーボタンコマンドを使うと、データブラウザーで現在選択されているオブジェクトに対してコマンドや関数を呼び出したり、選択内容とはまったく関係のないアクションを実行したりすることができます。
例えば、コマンド文字列 “Display %s” を実行すると、現在選択されている各ウェーブが表示され、コマンド文字列 “Print IgorInfo(0)” を実行すると、履歴ウィンドウに一般的な情報が表示されます。

ユーザーボタンをクリックした時、コマンド文字列に %s が含まれている場合、選択されている項目ごとにコマンドが１回ずつ実行されます。

それ以外の場合は、現在の選択状態に関係なく、コマンドは１回だけ実行されます。
選択範囲全体に対して１回だけ操作を行いたい場合は、%s を使わず、代わりに関数内から GetBrowserSelection を呼び出す必要があります。

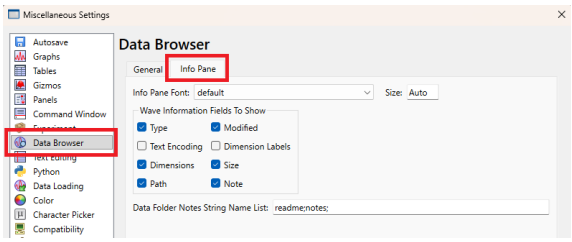
ユーザーボタンは、ウィンドウに追加された順序で描画されます。
位置を変更したい場合は、それらを一度削除してから、希望する順序で再度追加する必要があります。

ボタンはエクスペリメントやプレファレンスに保存されないため、Igor Pro の起動時にデータブラウザーに追加する必要があります。
どのエクスペリメントでも利用できるようにボタンのセットを追加するには、IgorStartOrNewHook フック関数を記述する必要があります。
詳細は、ヘルプ User-Defined Hook Functions を参照してください。

データフォルダーに関するメモ

特定のデータフォルダーに関するメモを、そのデータフォルダー内の文字列変数に保存し、Data Browser でそれらのメモを表示することができます。

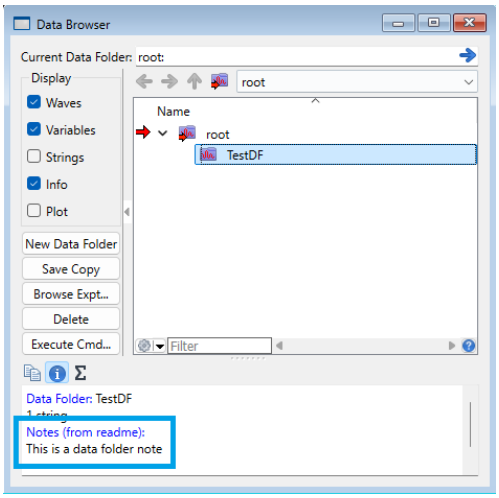
Data Browser で１つのデータフォルダーが選択され、情報ペインが表示されている場合、そのデータフォルダー内に、事前設定リストと名前が一致する文字列が含まれていると、その文字列の内容が情報ペインに表示されます。
文字列名のリストは、Miscellaneous Settings ダイアログの Data Browser カテゴリにある Info Pane タブで編集できます。
デフォルトのリスト名は「readme;notes;」です。



例えば、コマンドラインで次のコマンドを実行します。

```
NewDataFolder/O TestDF
String root:TestDF:readme = "This is a data folder note"
```

次に、Data Browser を開き、Info Pane のチェックボックスがオンになっていることを確認してから、メインリストから「TestDF」を選択します。
「root:TestDF:readme」の内容が情報ペインに表示されます。



データフォルダー内に、データフォルダーの注記として使う文字列名のリストと一致する文字列が複数含まれている場合、リスト内の最初の一致する文字列のみが使われます。

この機能は Igor Pro 9.0 で追加されました。
この機能を無効にするには、Miscellaneous Settings ダイアログで名前のリストを空文字列に設定してください。