CONTENTS

ビジュアルヘルプ – データのインポートとエクスポート（2）

一般的なテキストファイルの読み込み

ここでは、「一般的なテキスト」という用語は、１つまたは複数の数値データのブロックで構成されるテキストファイルを指すために使います。
ブロックとは、行と列の数字のセットです。
行内の数字は、１つ以上のタブまたはスペースで区切られています。
また、１つ以上の連続したカンマも空白文字として扱われます。
行は、改行文字（CR）、ラインフィード（LF）、改行/ラインフィード（CRLF）によって終了します。

Load General Text ルーチンは、数値データのみを処理し、日付、時刻、日付/時刻、テキストは処理しません。
これらの形式には、Load Delimited Text または Load Fixed Field Text を使います。
Load General Text では、1D だけでなく、2D の数値データも処理できます。

最初のデータブロックには、ヘッダー情報が先行している場合がありますが、これは Load General Text ルーチンが自動的にスキップします。

２つ目のブロックがある場合、通常は１つ以上の空行で最初のブロックと区別されます。
また、２番目のブロックの前にヘッダー情報があれば、Igor はこれもスキップします。

1D データをロードする時、Load General Text ルーチンは、各ブロックの各列を個別のウェーブにロードします。
列ラベルは、タブやカンマだけでなく、スペースも区切り文字として受けられることを除いて、Load Delimited Text ルーチンで説明するとおりに処理されます。
2D データを読み込む時には、すべての列を１つの 2D ウェーブに読み込みます。

Load General Text ルーチンは、行の数値の数を数えることで、ブロックの開始と終了の位置を決定します。
同じ数字の行を２つ見つけると、これをブロックの始まりとみなします。
ブロックは、異なる数字の個数を持つ行が現れるまで続きます。

一般的なテキストの例

一般的なテキストファイルに含まれるテキストの例を紹介します。

単純な一般的なテキスト

```
ch0             ch1             ch2             ch3       （ラベル行はオプション）
2.97055         1.95692         1.00871         8.10685
3.09921         4.08008         1.00016         7.53136
3.18934         5.91134         1.04205         6.90194
```

Load General Text ルーチンは、３つのポイントを持つ４つのウェーブを作成するか、または読み込みを行列として指定した場合は、３行４列の１つのウェーブを作成します。

ヘッダーを持つ一般的なテキスト

```
Date: 3/2/93
```

```
Sample: P21-3A
ch0           ch1           ch2           ch3      （ラベル行はオプション）
2.97055       1.95692       1.00871       8.10685
3.09921       4.08008       1.00016       7.53136
3.18934       5.91134       1.04205       6.90194
```

Load General Text ルーチンは、ヘッダー行（`Date:` と `Sample:`）を自動的にスキップし、３つのポイントを持つ４つのウェーブを作成するか、または読み込みを行列として指定した場合は、３行４列の１つのウェーブを作成します。

ヘッダーと複数ブロックを持つ一般的なテキスト

```
Date: 3/2/93
Sample: P21-3A
ch0_1         ch1_1         ch2_1         ch3_1    （ラベル行はオプション）
2.97055       1.95692       1.00871       8.10685
3.09921       4.08008       1.00016       7.53136
3.18934       5.91134       1.04205       6.90194

Date: 3/2/93
Sample: P98-2C
ch0_2         ch1_2         ch2_2         ch3_2    （ラベル行はオプション）
2.97055       1.95692       1.00871       8.10685
3.09921       4.08008       1.00016       7.53136
3.18934       5.91134       1.04205       6.90194
```

Load General Text ルーチンは、ヘッダー行を自動的にスキップし、３つのポイントのウェーブを８つ作成するか、または読み込みを行列として指定した場合は、３行４列の２つのウェーブを作成します。

一般的なテキスト、固定フィールドテキスト、区切り記号付きテキストの比較

Load General Text ルーチン、Load Fixed Field ルーチン、Load Delimited Text ルーチンのどれを使うべきか、迷うかもしれません。
ほとんどの商用プログラムは、これらのルーチンで処理できるシンプルなタブ区切りファイルを生成できます。
科学機器、メインフレームプログラム、カスタムプログラムで作成されたファイル、またはスプレッドシートからエクスポートされたファイルは、より多様です。
より効率的な方法を見つけるには、いくつかのルーチンを試してみる必要があるかもしれません。
最初にどれを試すべきかを決めるのには次の比較が役立つでしょう。

Load Fixed Field Text と Load Delimited Text と比較した Load General Text のメリット：

- 自動的にヘッダーテキストをスキップできる
- １つのファイルから複数のブロックをロードできる
- 列間に複数のタブまたはスペース文字を許容できる

Load Fixed Field Text と Load Delimited Text と比較した Load General Text のデメリット：

- ブランク（欠損値）を処理できない
- 数値列内に数値以外のテキストや値の列を許容できない
- テキスト値、日付、時刻、日付/時刻をロードできない

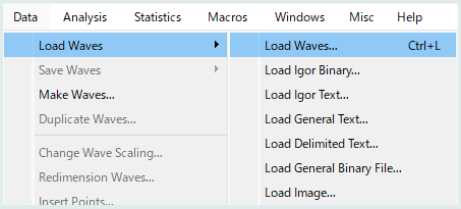
- 小数点としてカンマを処理できない（ヨーロッパの数字形式）

Load General Text ルーチンは、ファイルに「NaN」（Not a Number）として明示的に表されている場合は、欠損値を読み込むことができます。
数値のブロックがどこからどこまでかを判断する処理を混乱させるため、欠損値を空白で表したファイルは処理できません。

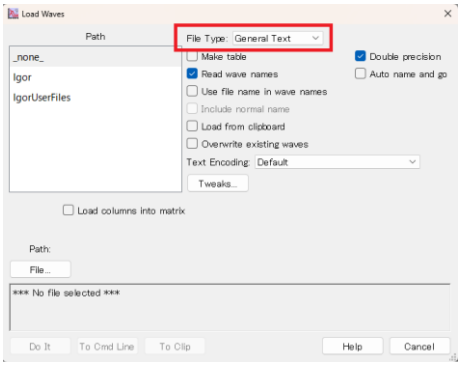
一般的なテキストに対する Load Wave ダイアログ – 1D

一般的なテキストファイルからデータを読み込む基本的なプロセスは次の通りです。

1. メニュー Data → Load Waves → Load Waves を選択し、Load Waves ダイアログを表示します。

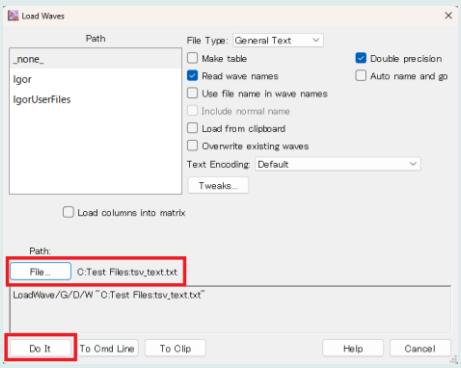


2. File Type ポップアップメニューから General Text を選択します。



3. File ボタンをクリックして、データを含むファイルを選択します。

Do It をクリックします。



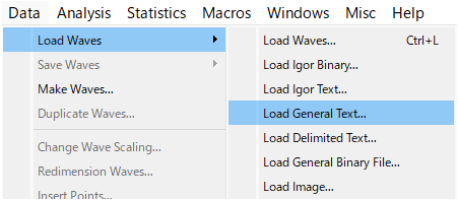
Do It をクリックすると、LoadWave コマンドが起動します。
Load General Text ルーチンが次のステップで実行されます。

1. 連続する行の数を数える処理を行って、データのブロックの開始位置を見つける
   このステップでは、ヘッダーがある場合はそれをスキップし、ブロックの列数を決定する
2. オプションで、数値のブロックの直前に列ラベルの行があるかどうかを決定する
3. オプションで、ウェーブの名前の確認または変更ができる別のダイアログを表示する
4. ウェーブを作成する
5. ファイルの最後まで、または数値の数が異なる行が見つかるまで、データをウェーブに読み込む

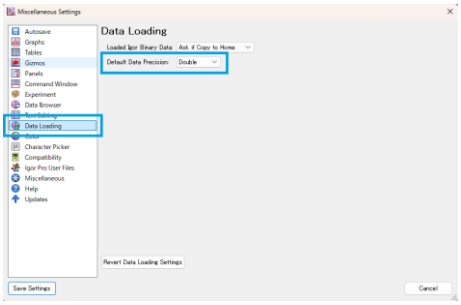
6. ファイルの終わりでない場合は、ステップ 1 に戻って別のブロックを探す

「Read wave names」オプションが有効な場合のみ、Igor が列ラベルの行を探します。
データブロックの直前の行を調べます。
ラベルが見つかり、ラベルの数がブロックの列の数と一致する場合、これらのラベルをウェーブ名として使います。
それ以外の場合には、Igor が自動的に wave0、wave1 などの形式のウェーブ名を生成します。

メニュー Data → Load Waves → Load Waves ではなく、Data → Load Waves → Load General Text を選択すると、Open File ダイアログが表示され、直接ロードする一般的なテキストファイルを選択できるようになります。
これは、Load Waves ダイアログをスキップし、デフォルトのオプションを使ってロードするショートカットです。
これは常に 1D ウェーブを読み込み、行列は読み込みません。

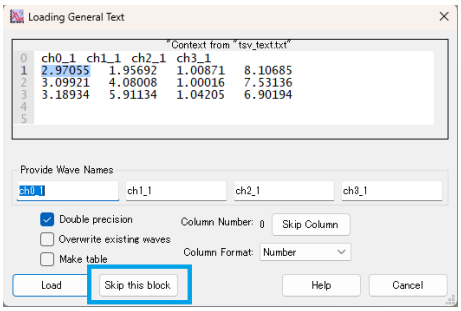


数値ウェーブの精度は、Miscellaneous Settings ダイアログ（Misc メニュー）の Data Loading セクションにある Default Data Precision 設定によってコントロールされます。
このショートカットを使う前に、Load Waves ダイアログで使うことができるオプションを確認してください。



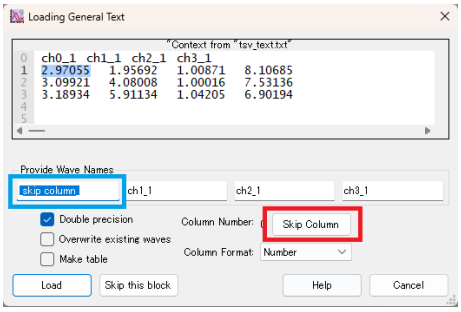
ブロックに対するウェーブ名の編集

前のセクションのステップ 3 では、Load General Text ルーチンが、ウェーブ名を変更できるダイアログを表示します。
これは、Load Delimited Text ルーチンとまったく同じように動作しますが、「Skip this block」という追加のボタンがある点が異なります。



「Skip this block」を使うと、複数ブロックの一般的なテキストファイルの 1 つ以上のブロックをスキップできます。

Skip Column ボタンをクリックすると、選択した名前ボックスに対応する列の読み込みをスキップできます。
選択した列以外のすべての列をスキップするには、ボタンを Shift キーを押しながらクリックします。



一般的なテキストに対する Load Wave ダイアログ – 2D

Igor は、Load General Text ルーチンを使って 2D ウェーブをロードできます。
ただし、Load General Text では、行/列ラベルと位置の読み込みはサポートされていません。
ファイルにそのような行と列がある場合は、区切り記号付きテキストファイルとして読み込む必要があります。

行列の読み込みに Load Delimited Text ルーチンではなく、Load General Text ルーチンを使う主な理由は、Load General Text ルーチンが数値以外のヘッダー情報を自動的にスキップできることです。
また、Load General Text は、カンマ１つだけでなく、スペースやタブを任意の数だけ１つの区切り記号として扱うため、あまり厳密でないフォーマットにも対応できます。

一般的なテキストデータの読み込み後にスケーリングを設定する

1D データが X 軸方向に等間隔で並んでいる場合、ウェーブフォームデータ用に設計された多くのコマンドや機能を使うことができます。
ウェーブを読み込んだ後、（メニュー Data → Change Wave Scaling）Change Wave Scaling ダイアログを使って、ウェーブの X スケーリングを設定する必要があります。

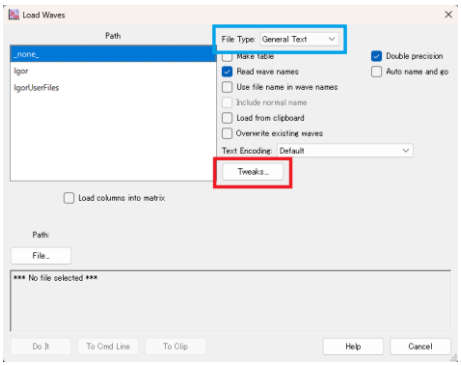
注記： データが等間隔の場合、ウェーブの X スケーリングを設定することが非常に重要です。
Igor の多くのコマンドは、正しい結果を得るために X スケーリング情報に依存しています。

1D データが等間隔ではない場合は、XY ペアを使うため、X スケーリングを変更する必要はありません。
データの単位を設定するには、Change Wave Scaling を使うとよいでしょう。

一般的なテキストの微調整（Tweaks）

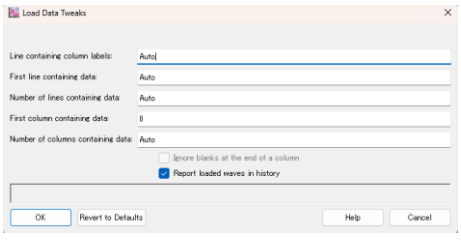
Load General Text ルーチンでは、ファイルを読み込む時に、処理を指示するための微調整（Tweak）ができるようになっています。

これを行うには、LoadWaves ダイアログの Tweaks ボタンを使います。



Load Data Tweaks ダイアログでは、Load Delimited Text ルーチンにのみ適用されるいくつかの項目が非表示になっています。
Load General Text は常に値間のタブやスペースをスキップし、カンマも１つスキップします。
「小数点」の文字は常にピリオドで、日付の処理には対応していません。

列ラベル、データ行、データ列に関連する項目には、２つの用途が考えられます。
ファイルの一部だけをロードしたり、自動的なデータブロックの検出方法が不正確な結果を出す場合に、動作を指示するためにこれらを使うことができます。



Tweaks ダイアログ内の行と列はゼロから始まる番号が振られています。

Line containing column labels と First line containing data 設定について、一般的なテキストファイルと区切り記号付きテキストファイルで異なる解釈をします。
区切り記号付きテキストの場合、ゼロは「最初の行」を意味します。
一般的なテキストの場合、これらのパラメーターのゼロは「自動」を意味します。

一般的なテキストにおける「自動」の意味は次の通りです。

First containing data が自動の場合
行をスキップすることなく、ファイルの先頭からデータの検索を開始します。
自動でない場合は、指定された行にスキップし、そこでデータの検索を開始します。
このようにすると、ファイルの先頭にあるデータのブロックをスキップすることができます。

Line containing column label が自動の場合
データ検索で見つかった行の直前の行で列ラベルを探します。
自動でない場合は、指定された行の列ラベルを探します。

Number of lines containing data が自動ではない場合
指定された行数に達するか、最初のブロックの終わりに到達した時点で読み込みを停止します。
この動作は、複数のブロックを含むファイルから、１つのブロックまたはブロックの一部を抽出できるようにするために必要です。

一般的なテキストファイルに複数のデータブロックが含まれ、Number of lines containing data が「自動」に設定されている場合、最初のブロック以降のブロックでは、列ラベルを含む行と最初のデータを含む行の間の関係を最初のブロックと同じ処理を維持します。
したがって、最初のブロックの列ラベルがデータの最初の行の１行前にある場合、それ以降のブロックでも同じことが当てはまると想定します。

First column containing data と Number of columns containing data の微調整（Tweaks）機能を使って、ブロック内のサブセットを読み込むことができます。
Number of columns containing data が「自動」または「0」に設定されている場合、ブロックの最後の列に到達するまで、すべての列を読み込みます。

一般的なテキストファイルでのトラブルシューティング

Load General Text ルーチンによって作成されたウェーブは、テーブルを使って確認することができます。
期待通りの結果が得られない場合は、Igor が処理できる形式になるまでテキストファイルを調べて編集することができます。
以下の点を忘れないようにしてください：

- Load General Text は、日付、時刻、日付/時刻、小数点として使われているカンマ、数値列ではない列を持つデータのブロックを処理できません。
  代わりに、Load Delimited Text を試してください。

- 数値間のタブやスペースはすべてスキップされ、カンマ１つもスキップされます。

- 列ラベルがある場合は、別の設定をしていない限り、数値データの前に現れる最初の行に位置すると想定しています。
  ラベルもタブ、カンマ、スペースで区切られていることを想定しています。
  Read Wave Names オプションを有効にしない限り、ラベルは検索されません。

- これは、連続する行の値の数を数えることで機能します。
  一部の特殊なフォーマット（例：1234.56 ではなく 1,234.56）は、このカウントを狂わせ、早く新しいブロックを始めてしまうことがあります。

- 空白や数値以外の値を含む列は処理できません。
  これらのいずれも、新しいデータのブロックを開始する原因となります。

- 列数の変化を検出すると、新しいブロックを新しいウェーブセットに読み込み始めます。

ファイルを調べるだけでは問題が特定できない場合は、データのサブセットを作って読み込むという手法を試してみるべきです。
これは、「区切り記号付きテキストファイルのトラブルシューティング」のセクションで説明されており、問題の解決に繋がることもあります。

LoadWave のウェーブ名の生成

Igor バイナリファイルまたは Igor テキストファイルをロードする時、LoadWave は、ロードするファイルに保存されているウェーブ名を使います。

区切り記号付きテキスト（/J）、固定フィールドテキスト（/F）、一般的なテキスト（/G）としてファイルを読み込む場合、ウェーブ名は /A、/N、/W、/B、/NAME フラグによって決定されます。
このセクションでは、これらの命名フラグの動作について説明します。

命名フラグをすべて省略すると、LoadWave は wave0、wave1、wave2 のようなウェーブ名を生成しますが、これらのウェーブが既に存在する場合は、wave3、wave4、wave5 のようなユニークな名前を生成します。
LoadWave は、名前を編集できるダイアログを表示します。

/A フラグ

すべてを省略したときと同じ動作をしますが、名前を編集できるダイアログをスキップする「Auto name & go」（Load Waves ダイアログ）が有効になります。
/A=baseName は、/A と同じですが、'wave' 以外のベース名を指定できる点が異なります。

/N フラグ

常にゼロから始まるサフィックス番号を使い、ファイルから読み込まれる各ウェーブごとに 1 つずつインクリメントされることを除いて /A フラグと同じです。
生成された名前が既存のウェーブと競合する場合、既存のウェーブが上書きされます。
例えば、/N=wave と指定すると、wave0、wave1、wave2 というウェーブ名が生成されます。

/W フラグ

ファイル自体からウェーブ名を読み込みます。
デフォルトでは、LoadWave ウェーブ名がファイルの最初の行にあることを想定していますが、/L フラグを使うと、別の行を指定することができます。
ファイル内の名前が既存のウェーブ名と競合し、上書き指定（/O）した場合、既存のウェーブは上書きされます。
上書きを指定しない場合、LoadWave は固有の名前を入力できるダイアログを表示します。

/B フラグ

LoadWave をユーザー定義関数から呼び出すときに使う /B フラグにより、各列に明示的な名前を指定することができます。
詳細は、「個々の列の特性の指定」のセクションを参照してください。

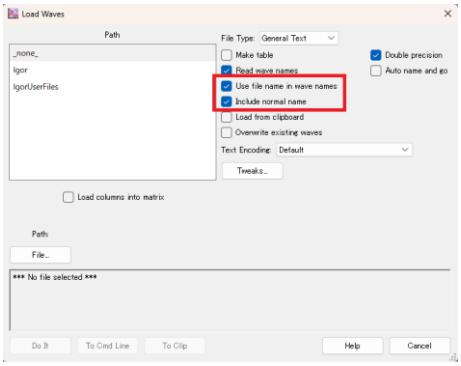
/NAME フラグ

ファイル名をウェーブ名に簡単に組み込む方法を提供します。
詳細は次のセクションを参照してください。

/NAME は /B を上書きし、/B は /W を上書きし、/W は /N を上書きし、 /N は /A を上書きします。

ウェーブ名にファイル名を使う

LoadWave /NAME フラグは、主にウェーブ名にファイル名を簡単に組み込む方法を提供するために、Igor Pro 9.0 で追加されました。



Load Wave ダイアログ（メニュー Data → Load Waves → Load Waves）では、Use File Name in Wave Names と Include Normal Name チェックボックスで /NAME フラグをサポートしています。
このダイアログでは、/NAME のすべての機能にアクセスできるわけではありませんが、ほとんどの一般的な用途には十分です。

このセクションでは、/NAME フラグの一般的な説明を行います。
続きのセクションでは、使い方を明確にするための例を示します。

フラグのフォーマットは次のようになります：

`/NAME={`*`namePrefix`*`, `*`nameSuffix`*`, `*`nameOptions`*`}`

一般的なウェーブ名は次の要素で構成されます：

`<`*`namePrefix`*`><normal name><`*`nameSuffix`*`><suffix number>`

`namePrefix` と `nameSuffix` は、空（“”）、リテラルテキスト、特別なパターン ”:filename:” または”:filename:_” のようなリテラルテキストと特別なパターンの組み合わせにすることができます。
LoadWave は、特別なパターン “:filename:” を、読み込まれるファイルの名前から拡張子を除いたものに置き換えます。
`namePrefix` と `nameSuffix` の両方が空の場合、LoadWave は /NAME フラグが省略された場合と同様に動作します。

<normal name> は、/NAME が省略された場合に使われるウェーブ名を指します。

このセクションの残りの部分では、さまざまなシナリオで柔軟な命名を可能にする名前オプションのビット単位パラメーターについて説明します。
抽象的な表現である nameOptions は分かりにくいかもしれません。
その意味と用法については、次に示す例で明確になると思います。

nameOptions の bit	説明
0	設定されていれば、LoadWave が <normal name> を含みます。 設定されていなければ、<normal name> を省きます。
1, 2, 3	サフィックスの数字の使用をコントロールします。 サフィックスの数字とは、0、1、2 などの数字で、ウェーブ名をユニークにするために使われます。 LoadWave で単一のウェーブをロードする時、nameOptions の bit 1 が設定されてい

る場合、次の bit 3 で抑制されていない限り、<suffix number> が含まれます。
複数のウェーブをロードする時、nameOptions の bit 2 が設定されている場合、LoadWave は次の bit 3 で抑制されていない限り、<suffix number> が含まれます。
複数のウェーブを読み込む時には、読み込むウェーブの名前を区別するためにサフィックスの数字が必要となるため、サフィックスの数字を含めたい場合が多いです。
しかし、１つのウェーブを読み込む時には、サフィックスの数字を除外したい場合もあると思います。
その場合は、bit 1 はクリアしたままにし、bit 2 と 3 を設定します。

3 bit 3 は、bit 1 と 2 を上書きし、名前の競合を防ぐために必要ではないときにはサフィックスの番号を追加しないようにします。
単一のウェーブを読み込む時、bit 3 が bit 1 を上書きし、名前の競合がないときにはサフィックスの番号を追加しないようにします。
複数のウェーブを読み込む時、bit 3 が bit 2 を上書きし、名前の競合がないときにはサフィックスの番号を追加しないようにします。

4 設定されていれば、LoadWave は、既存のウェーブや他のオブジェクトとの衝突を避けるために、サフィックス（有効になっている場合）の番号を選択します。
クリアされている場合は、サフィックス番号は 0 から始まり、ロードされるウェーブごとにインクリメントされます。

5 クリアされている場合は、LoadWave はウェーブ名を標準名に変更します。
そうでない場合は、自由な名前を使用できます。
自由な名前でのプログラミングは難しいため、標準の名前を使うことを推奨します。
詳細は、ヘルプ Object Names を参照してください。

ファイル名を使って単一のウェーブをロードする

このセクションでは、「Data.txt」という名前のファイルをロードし、そのファイルから１つのウェーブをロードすると仮定します。

```
// nameOptions=0 で normal name を省略する
LoadWave/NAME={":filename:","",0}
```

LoadWave は、まだ存在していない場合、Data という名前のウェーブを作成します。
すでにその名前が存在していて、/O（上書き）フラグを付けている場合は、Data は上書きされます。
LoadWave に /O フラグを付けずに実行すると、ダイアログボックスが表示され、ユニークな名前を入力することができます。

```
// nameOptions=26 でユニークなサフィックス番号を含める
// 名前が衝突している場合に限る
LoadWave/NAME={":filename:","",26}          // 26 = 2 | 8 | 16 （bits 1, 3, 4 を設定）
```

LoadWave は、まだ存在していない場合、Data という名前のウェーブを作成します。
すでにその名前が存在する場合、LoadWave は、Data0、Data1 という名前を作成します。
サフィックスの数字は、結果として得られるウェーブ名がユニークになるように選択されます。

ファイル名を使って複数のウェーブを読み込む

このセクションでは、「Data.txt」という名前のファイルをロードし、そのファイルから３つのウェーブをロードすると仮定します。

```
// nameOptions=0 で normal name を省略する
LoadWave/NAME={":filename:","",0}
```

この場合、サフィックス番号を要求していないため、生成されるウェーブ名のすべてが「Data」となり、
LoadWave はユニークな名前を入力するよう、ダイアログを表示します。

```
// nameOptions=4 で常に順番にサフィックスを付ける
LoadWave/NAME={":filename:","",4}          // bit 2 を設定
```

LoadWave は、Data0、Data1、Data2 という名前のウェーブ名を生成します。
これらのウェーブが既に存在し、/O（上書き）フラグが指定されている場合、それらは上書きされます。
もし、/O を省略した場合、LoadWave はユニークな名前を入力できるダイアログを表示します。

```
// nameOptions=20 で常に順番にサフィックス番号を付ける
LoadWave/NAME={":filename:","",20}         // 20 = 4 | 16 （bits 2, 4 を設定）
```

LoadWave は、Data0、Data1、Data2 などのウェーブ名を生成します。
サフィックスの番号は、名前をユニークにするために選択されます。
同じファイルに対して同じコマンドを２回実行すると、LoadWave は Data3、Data4、Data5 というウェーブ名を生成します。

ファイル名と通常の名前を使って複数のウェーブを読み込む

このセクションでは、「Data.txt」という名前のファイルをロードし、そのファイルから３つのウェーブをロードすると仮定します。
さらに、ファイルには ColumnA、ColumnB、ColumnC の列名が含まれていると想定し、ファイルから列名を読み込むために /W フラグを使います。

```
// nameOptions=1 で通常の名前を含める
LoadWave/W/NAME={":filename:_","",1}
```

これは、Data_ColumnA、Data_ColumnB、Data_ColumnC というウェーブ名を生成します。
これらのいずれかがすでに存在し、/O（上書き）フラグが含まれている場合、それらは上書きされます。
もし存在し、/O を省略した場合、LoadWave はユニークな名前を入力できるダイアログを表示します。

```
// nameOptions=21 で常に通常の名前とユニークなサフィックス番号を付ける
LoadWave/W/NAME={":filename:_","",21}      // 20 = 1 | 4 | 16 （bits 0, 2, 4 を設定）
```

LoadWave は、Data_ColumnA0、Data_ColumnB0、Data_ColumnC0 というウェーブ名を生成します。
サフィックスの番号は、名前をユニークにするために選択されます。
同じファイルに対して同じコマンドを２回実行すると、LoadWave は Data_ColumnA1、Data_ColumnB1、Data_ColumnC1 というウェーブ名を生成します。

このテクニックは、/W フラグの代わりに /B を使って、ファイル名と /B で明示的に指定された追加の名前を組み合わせたウェーブ名を作成する時にも使えます。
機能的な例は、「データファイルを読み込む時にウェーブ名を設定する」のセクションを参照してください。

LoadWave のその他の機能

このセクションでは、テキストデータファイルの読み込みに関するその他の項目について説明します。

カスタムの日付フォーマットを読み込む

このセクションは、区切り記号付きテキスト（/J）、固定フィールドテキスト（/F）、一般的なテキスト（/G）の読み込みに適用されます。
以下に、カスタムの日付フォーマットの例と、LoadWave /R フラグを使ってそれらを指定する方法を示します。

October 11, 1999	`/R={English, 2, 4, 1, 1, "Month DayOfMonth, Year", 40}`
Oct 11, 1999	`/R={English, 2, 3, 1, 1, "Month DayOfMonth, Year", 40}`
11 October 1999	`/R={English, 2, 4, 1, 1, "DayOfMonth Month Year", 40}`
11 Oct 1999	`/R={English, 2, 3, 1, 1, "DayOfMonth Month Year", 40}`
10/11/99	`/R={English, 1, 2, 1, 1, "Month/DayOfMonth/Year", 40}`
11-10-99	`/R={English, 1, 2, 2, 1, "DayOfMonth-Month-Year", 40}`
11-Jun-99	`/R={English, 1, 3, 2, 1, "DayOfMonth-Month-Year", 40}`
991011	`/R={English,1,2,2,1,"YearMonthDayOfMonth", 40}`

区切り記号付きテキストとしてデータを読み込む時、「October 11, 1999」のようなカンマを含む日付形式を使う場合は、LoadWave がカンマを区切り記号として扱わないように、/V フラグを使う必要があります。

991011 のように、すべて数字で構成される日付形式を読み込む時には、LoadWave/B フラグを使って、そのデータが日付であることを LoadWave に伝える必要があります。
そうしないと、LoadWave は通常の数値として処理します。

各列の特性を指定する

LoadWave /B=columnInfoStr フラグは、区切り記号付きテキスト（/J）、固定フィールドテキスト（/F）、一般的なテキスト（/G）の各列に関する情報を LoadWave に提供します。
このフラグは、LoadWave の通常の動作を上書きします。
ほとんどの場合、このフラグを使う必要はありません。
/B は、ユーザー定義関数で追加のコントロールが必要な場合に便利です。

columnInfoStr は次の形式で表されます。

```
"<column info>;<column info>; . . .;<column info>;"
```

`<column info>` は、以下の１つ以上を使うことができます。

C=<number>　この列情報の指定でコントロールされる列の数です。
<number> は１以上の整数です。

F=<format>　列のデータ型を指定するコードです。
<format> は、-2 から 10 の整数です。

-2　テキスト。列はテキストウェーブにロードされる
-1　フォーマット不明。Igor がフォーマットを推測
0～5　数値
6　日付
7　時刻
8　日付/時刻
9　８進数
10　１６進数

F= フラグは、区切り記号付きテキストファイルと固定フィールドテキストファイルのみで使われます。
一般的なテキストファイルでは無視されます。

N=<name>　列に使う名前です。
<name> は標準の名前（例：wave0）またはクオート付きの自由な名前（例：'Heart Beat'）にすることができます。
<name> が '_skip_' の場合、LoadWave はその列をスキップします。

N= フラグは、区切り記号付きテキスト、固定フィールドテキスト、一般的なテキストファイルで機能します。

詳細は、「LoadWave のウェーブ名の生成」のセクションを参照してください。

T=<numtype>　その列の数値の型を指定する数字です。
このフラグは LoadWave/D フラグを上書きします。
フォーマットがテキストである列には影響しません。
<numtype> は以下のどれかである必要があります。

2　32-bit float
4　64-bit float
8　8-bit singed integer
16　16-bit signed integer
32　32-bit signed integer
72　8-bit unsigned integer
80　16-bit unsigned integer
96　32-bit unsigned integer

W=<width>　固定フィールドファイルの列フィールドの幅です。
<width> は 1 以上の整数です。
固定幅ファイルとは、FORTRAN 形式のファイルで、各列に固定バイト数が割り当てられ、スペースがパディングとして使われるファイルです。

W= フラグは、固定フィールドテキストでのみ使われます。

/B=columnInfoStr フラグの例としては、次のようなものです。

```
/B="C=1,F=-2,T=2,W=20,N=Factory; C=1,F=6,W=16,T=4,N=MfgDate;
    C=1,F=0,W=16,T=2,N=TotalUnits; C=1,F=0,W=16,T=2,N=DefectiveUnits;"
```

この例は２行で示されていますが、実際のコマンドは１行です。
プロシージャでは、次のように書くことができます。

```
String columnInfoStr = ""
columnInfoStr += "C=1,F=-2,T=2,W=20,N=Factory;"
columnInfoStr += "C=1,F=6,T=4,W=16,N=MfgDate;"
columnInfoStr += "C=1,F=0,T=2,W=16,N=TotalUnits;"
columnInfoStr += "C=1,F=0,T=2,W=16,N=DefectiveUnits;"
```

クオートで囲まれた文字列内の各フラグは、カンマまたはセミコロンで終わることに注意してください。
カンマは、特定の列情報の指定内のフラグを区切ります。
セミコロンは列情報の指定の終了を意味します。
最後のセミコロンは必須です。
文字列内ではスペースとタブが許可されています。

この例では、４つの列を含むファイルに関する情報を提供します。

最初の列情報の指定は、`"C=1,F=-2,T=2,W=20,N=Factory;"` です。
これは、この指定が
・１つの列に適用
・列のフォーマットがテキスト
・数値フォーマットが単精度浮動小数点（ただし、これはテキスト列には影響しない）
・列データが 20 バイトの固定フィールド幅
・作成されたウェーブの名前は「Factory」
であることを示します。

２番目の列情報の指定は、`"C=1,F=6,T=4,W=16,N=MfgDate;"` です。
これは、この指定が
・１つの列に適用
・列のフォーマットが日付
・数値フォーマットが倍精度浮動小数点（倍精度は常に日付にに使われる）
・列データが 16 バイトの固定フィールド幅
・作成されたウェーブの名前は「MfgDate」
であることを示します。

３番目の列情報の指定は、`"C=1,F=0,T=2,W=16,N=TotalUnits;"` です。
これは、この指定が
・１つの列に適用
・列のフォーマットが数値
・数値フォーマットが単精度浮動小数点
・列データが 16 バイトの固定フィールド幅
・作成されたウェーブの名前は「TotalUnits」
であることを示します。

４番目の列情報の指定は、ウェーブ名が DefectUnits であることを除いて、３番目と同じです。

列指定の項目はすべて任意です。列情報の指定の各項目のデフォルト値は次の通りです。

C=<number>	C=1 列情報が 1 つの列を記述していることを示します。
F=<format>	F=0 /K フラグで指定されたフォーマットを決定します。 /K=0 が使われた場合、LoadWave は自動的に列のフォーマットを決定します。
N=<name>	N=_auto_ /B フラグが省略された場合と同様に、ウェーブ名を生成します。
T=<numtype>	LoadWave/D フラグが使われている場合は T=4（倍精度） /D フラグが省略されている場合は、T=2（単精度）
W=<width>	W=0 固定幅のファイルの場合、W=<width> を使って 0 より大きな明確なフィールド幅を指定しない限り、LoadWave は /F フラグで指定されたデフォルトのフィールド幅を使います。

デフォルト値を利用すれば、先の例は次のように短縮できます。

```
/B="F=-2,W=20,N=Factory; F=6,T=4,W=16,N=MfgDate;
W=16,N=TotalUnits; W=16,N=DefectiveUnits;"
```

もしファイルが固定フィールドのテキストファイルでなかった場合、W= を省略し、例は次のようになります。

```
/B="F=-2,N=Factory; F=6,T=4,N=MfgDate; N=TotalUnits; N=DefectiveUnits;"
```

以下に、/B=*columnInfoStr* フラグの使用例と説明をいくつか示します。

この例では、/B フラグは、ファイル内の列から作成されるウェーブに使う名前を指定する目的のみに使われています。

```
/B="N=WaveLength; N=Absorbance;"
```

先の例のウェーブ名は標準名です。
スペースやドットを含む名前など、自由な名前を使いたい場合は、シングルクオートを使う必要があります。
例えば、

```
/B="N='Wave Number'; N='Reflection Angle';"
```

上書きがオフになっていて、すでにこの名前のウェーブが存在する場合、またはマクロ、関数、演算子、変数と名前が競合する場合には、N= で指定した名前を使うことはできません。
このような場合、LoadWave は、該当する列の N= フラグで指定した名前に 1 つ以上の数字を追加して、ユニークな名前を生成します。
別のウェーブ名との競合問題を回避するには、上書き（/O）フラグを使うか、または新しく作成したデータフォルダーにデータを読み込むことになります。
あいまいな名前を避けることで、関数、コマンド、変数による名前の衝突の可能性を最小限に抑えることができます。

2 つの N= フラグに同じ名前を指定すると、LoadWave はエラーとなるため、名前がユニークであることを確認してください。
ただし、指定された名前が _skip_ である場合を除き、C= フラグを使って複数の列を指定した場合でも、N= フラ

グは 1 つの列のみの名前を生成します。
次の例を考えてみてください。

```
/B="C=10,N=Test;"
```

これは表面上、10 列に対して「Test」という名前を使っています。
しかし、ウェーブ名はユニークでなければならないため、LoadWave ではこのようなことは行いません。
これは、最初の列のみ「Test」という名前を使い、他の列にはデフォルト名前が割り当てられます。

/L フラグを使うと、ファイル内の列の一部だけを読み込むことができます。
この場合でも、/B フラグで指定する列情報の指定は、読み込む最初の列ではなく、ファイル内の最初の列から開始されます。
例えば、/L を使って列 0 と 1 をスキップする場合、列情報の指定で列 0 と 1 をスキップする必要があります。

```
// 列 0 と 1 をスキップし、続く列に名前を付ける
/L={0,0,0,2,0} /B="C=2;N=Column2;N=Column3;
```

「C=2;」の部分は、列 0 と 1 のデフォルトの設定を受け入れ、その後の指定はそれ以降の列に適用されます。
/B を使うと、/L を使わなくても、次のように同じ結果を得ることができます。

```
/B="C=2,N='_skip_';N=Column2;N=Column3;"
```

また、LoadWave は行列ウェーブにデータを読み込む時に、 1 つの名前のみ使います。
複数の名前を指定した場合、最初の名前のみが使われます。
行列にデータを読み込み、列をスキップする場合、スキップに関する上記の説明が適用されます。

次の例では、/B フラグはファイル内の各列のフォーマットのみを指定します。
対象のファイルは、テキスト列で始まり、日付列、 3 つの数値列が続きます。

```
/B="F=-2; F=6; C=3,F=0"
```

ほとんどの場合、LoadWave はフォーマットを自動的に推測できるため、F= フラグを使う必要はありません。
このフラグは、列のフォーマットを誤って推測するような場合に使います。
また、LoadWaves は 8 進数や 1 6 進数を自動的に推測できないため、これらを強制的に解釈させる時にも役立ちます。

F= フラグで使われる数字コード（0…10）は、ModifyTable コマンドで使うコードと同じです。
/E フラグを使ってテーブルを作成した場合、F= フラグはテーブル列の数値フォーマットをコントロールします。

コード -1 は、実際の列フォーマットのコードではありません。
特定の列に対して F=-1 を使うと、LoadWave は列テキストからその列のフォーマットを推測します。

次の例では、/B フラグは固定フィールドファイル内の各列の幅を指定する目的でのみ使われています。
このファイルには、20 バイトの列が 1 つ、その後 16 バイトの列が 10 個、その後 24 バイトの列が 3 つ続きます。

```
/B="C=1,W=20; C=10,W=16; C=3,W=24"
```

W= で指定されたフィールド幅は、/F フラグで指定されたデフォルトのフィールド幅を上書きします。
ファイル内のすべての列のフィールド幅が同じである場合は、/F フラグだけを使うことができます。

/L フラグを使うと、ファイル内の列の一部だけを読み込むことができます。
この場合でも、/B フラグで設定する列情報の指定は、読み込む最初の列ではなく、ファイル内の最初の列から始まります。

LoadWave のその他の問題

このセクションでは、LoadWave コマンドに関するその他の問題について説明します。

LoadWave のテキストエンコーディングの問題

このセクションでは、上級ユーザーにとって関心のある LoadWave テキストエンコーディングの問題について説明します。
テキストエンコーディングの一般的なトピックについては、ヘルプ Text Encodings で説明しているため、それを理解していることを前提としています。

Igor はすべてのテキストを UTF-8 として内部的に保存しているため、読み込んだテキストをソーステキストのエンコーディングから UTF-8 に変換する必要があります。

Igor のバイナリファイルをロードする時には、LoadWave は /ENCG=textEncoding フラグを無視します。
ロードされたウェーブのテキストエンコーディングは、ヘルプ LoadWave Text Encodings for Igor Binary Wave Files の説明に従って決定されます。

このセクションの残りの部分は Igor バイナリウェーブファイルではなく、プレーンテキストファイルからのデータ読み込みに適用されます。

テキストデータファイルを読み込む時には、/ENCG=textEncoding フラグを使って、そのテキストのエンコード方式を Igor に指示することができます。
textEncoding の指定可能な値の一覧の詳細は、ヘルプ Text Encoding Names and Codes を参照してください。

テキストエンコーディングのコードの一覧（日本でよく使うと思われるもの）

テキストエンコーディング	コード
なし	0
UTF-8	1
Macintosh, MacRoman, x-macroman	2
Windows-1252	3
Shift_JIS	4
MacJapanese, x-mac-japanese	4
Windows-932	4
EUC-JP	5
UTF-16BE	100
UTF-16LE	101
UTF-32BE	102
UTF-32LE	103
ISO-8859-1, Latin1	120

Symbol	150
Binary	255

LoadWave は、/ENCG で指定されたテキストエンコードと、ヘルプ Determining the Text Encoding for a Plain Text File で説明されている、テキストファイルのデータを UTF-8 に変換するためのソーステキストエンコーディングを決定するルールを使います。
/ENCG を省略するか、/ENCG=0 を指定すると、指定されたテキストエンコーディングは不明となり、テキストエンコーディングの決定には考慮されません。
ファイルのテキストを UTF-8 に変換するときに有効なテキストエンコーディングが、ルールに従っても特定できない場合、Choose Text Encoding ダイアログを表示します。

ファイルが ASCII 文字のみで構成されている場合によくあることですが、バイト指向のテキストエンコーディングであればどれでも機能し、/ENCG フラグを使う必要はありません。

巨大なファイル（例えば、数百 MB）を読み込む場合、ソーステキストの有効なエンコーディングを見つけるために、ファイルの読み込みに要する時間が大幅に増える可能性があります。
ファイルがすべて ASCII であるか、有効な UTF-8 であることがわかっている場合は、次のように、オプションのパラメーターを使って、テキストのエンコード変換を完全にスキップするように LoadWave に指示することができます：

```
/ENCG={1,4}
```

「1」は、テキストが UTF-8 として有効であることを LoadWave に伝えます。
つまり、テキストがすべて ASCII 文字であるか、または非 ASCII 文字が含まれている場合はそれらがすべて UTF-8 として適切にエンコーディングされていることを意味します。

「4」は、LoadWave にテキストが UTF-8 として有効であると想定し、すべての検証と変換をスキップするように指示するものです。

注記： このフラグを使っても、ファイルが有効な UTF-8 ではなく、データをテキストウェーブに読み込む場合、テキストウェーブは無効なデータがあるところで終了し、後でウェーブを使おうとするとエラーが発生します。

先に説明したとおり、ファイルのテキストを UTF-8 に変換するときに有効なテキストエンコーディングをルールに従って特定できない場合、Choose Text Encoding ダイアログを表示します。
自動化されたプロシージャで多数のファイルを読み込む場合、このダイアログが表示されると、プロシージャが完全に停止します。
次のように、別のオプションフラグを使うことで、これを防ぐことができます：

```
/ENCG={1,8}
```

このフラグを使うと、LoadWave がファイルのテキストエンコーディングを決定できない場合、エラーが返されます。
他のファイルでプロシージャ処理を継続したい場合には、GetRTError を使ってエラーをチェックし、処理する必要があります。

非常に大きなファイルをロードする

テキストファイルをロードする時に LoadWave が処理できるウェーブ（列）またはポイント（行）の数は、使うことができるメモリ容量のみに制限されます。

/L フラグの numLines パラメーターを使うと、非常に大きなファイル（5 万行以上のデータ）の読み込み速度と効率を向上することができます。
通常、このパラメーターは、ファイル全体ではなく、ファイルの一部を読み込むために使われます。
ただし、区切り記号付きの一般的なテキストや固定フィールドテキストのロードの場合、numLines パラメーターは、ウェーブが最初に持つべき行数を指定します。
したがって、必要なメモリはすべてロードの開始時に割り当てられ、データ行がロードされるたびに増やすのではありません。
非常に大きなファイルを読み込む場合、ファイル内の正確なデータ行数がわかっている場合は、/L フラグの numLines パラメーターを使います。
正確な行数がわからない場合は、予想して、より大きな数値を指定することができます。

/L フラグを省略した場合、numLines パラメーターがゼロの場合、または、500,000 バイトを超える場合、LoadWave は自動的にファイル内のデータ行数をカウントし、データ読み込みが始まる前にウェーブ全体を割り当てることができるようにします。
これは、/L を使い、numLines を正確に正しい値に設定した場合と同じ動作になり、通常、読み込み処理が大幅に高速化されます。
この機能は、/V フラグを使い、loadFlags パラメーターの Bit 2 を 1 に設定することで無効にすることができます。

エスケープシーケンス

エスケープシーケンスとは、プレーンテキストで特殊文字を表すために使われる 2 文字のシーケンスです。
エスケープシーケンスは、バックスラッシュ文字（\）で始まります。

デフォルトでは、テキスト列において、LoadWave は次のエスケープシーケンスを解釈します：
\t（タブ）、\n（ラインフィード）、\r（キャリッジリターン）、\"（ダブルクオート）、\'（シングルクオート）
これは、Igor の「Save」コマンドとうまく連携し、これらの最初の 4 文字をエンコードするためにエスケープシーケンスを使います。

エスケープシーケンスを含まないが、バックスラッシュを含んでいるファイルを読み込む場合は、/V フラグの loadFlags パラメーターの Bit 3 を設定することで、これらのエスケープシーケンスの解釈を無効にすることができます。
これは主に、エスケープされていない Windows ファイルシステムのパスを含むテキストファイルを読み込む時に役立ちます。

Igor テキストファイルの読み込み

Igor テキストファイルは、キーワード、データ、および Igor コマンドで構成されます。
データは数値、テキスト、またはその両方であり、１次元から４次元まで可能です。

多くの Igor ユーザーが、独自のカスタムプログラムから Igor へデータをエクスポートする時に、この形式が簡単で強力であると感じています。

Igor テキストファイルのファイル名拡張子は「.itx」です。
古いバージョンの Igor では「.awav」が使われていましたが、これは現在も有効です。

Igor テキストの例

以下は、Igor テキストファイルで見かける可能性のあるテキストの例です。

シンプルな Igor テキスト

```
IGOR
WAVES/D unit1, unit2
BEGIN
        19.7    23.9
        19.8    23.7
        20.1    22.9
END
X SetScale x 0,1, "V", unit1; SetScale d 0,0, "A", unit1
X SetScale x 0,1, "V", unit2; SetScale d 0,0, "A", unit2
```

これをロードすると、unit1 と unit2 という名前の 2 つの倍精度ウェーブが作成され、それらの X スケーリング、X 単位、データ単位が設定されます。

追加のコマンドを持つ Igor テキスト

```
IGOR
WAVES/D/O xdata, ydata
BEGIN
        98.822          486.528
        109.968         541.144
        119.573         588.21
        133.178         654.874
        142.906         702.539
END
X SetScale d 0,0, "V", xdata
X SetScale d 0,0, "A", ydata
X Display/N=TempGraph ydata vs xdata
X ModifyGraph mode=2,lsize=5
X CurveFit line ydata /X=xdata /D
X Textbox/A=LT/X=0/Y=0 "ydata= \\{W_coef[0]}+\\{W_coef[1]}*xdata"
X PrintGraphs TempGraph
X KillWindow TempGraph                                  // Kill the graph
X KillWaves xdata, ydata, fit_ydata                     // Kill the waves
```

これをロードすると、２つの倍精度ウェーブが生成され、データ単位が設定されます。
その後、グラフを作成し、カーブフィッティングを行い、グラフに注釈を付け、グラフを出力します。
最後の２行は後処理を行います。

Igor テキストのファイルフォーマット

Igor テキストファイルはキーワード IGOR で始まります。
ファイルの残りの部分は、ウェーブに読み込まれるデータブロックや実行される Igor コマンドを含み、必ず空白行で終わらなければなりません。

Igor テキストファイル内のデータブロックは、読み込むウェーブの宣言を先頭に置かなければなりません。
この宣言はキーワード WAVES で始まり、オプションのフラグと読み込むウェーブ名を続けます。
次にキーワード BEGIN がデータブロックの開始を示します。
キーワード END がデータブロックの終了を示します。

ファイルには任意の数のデータブロックを含めることができ、各ブロックには宣言が先行します。
ウェーブが１次元の場合、ブロックには任意の数のウェーブを含めることができますが、特定のブロック内のウェーブはすべて同じデータ型でなければなりません。
多次元ウェーブは、１ブロックあたり１つのウェーブとして出現しなければなりません。

ブロック内のデータ行は、１つ以上の数値またはテキスト項目で構成され、数値はタブで区切られ、行末に終端記号が付加されます。
終端記号は CR、LF、または CRLF のどれかです。
各行の項目数は同一でなければなりません。

Igor テキストファイルでは、空白、日付、時刻、または日付時刻を使用できません。
数値列の欠損値を表すには、「NaN」（not-a-number）を使ってください。
日付や時刻を表すには、標準の Igor 日付形式（1904 年 1 月 1 日からの秒数）を使ってください。

ウェーブの数やポイントの数に制限はありませんが、すべてのデータが利用可能なメモリに収まる必要があります。

WAVES キーワードは次のオプションフラグを受け付けます：

フラグ	**効果**
/N=(...)	多次元ウェーブにおける各次元のサイズを指定します。
/O	既存のウェーブを上書きします。
/R	ウェーブを実数にします（デフォルト）。
/C	ウェーブを複素数にします。
/S	ウェーブを単精度浮動小数点にします（デフォルト）。
/D	ウェーブを倍精度浮動小数点にします。
/I	ウェーブを 32 ビット整数にします。
/W	ウェーブを 16 ビット整数にします。
/B	ウェーブを 8 ビット整数にします。
/U	整数ウェーブを符号なしにします。
/T	テキストデータ形式を指定します。

通常は単精度または倍精度の浮動小数点ウェーブを作成すべきです。
整数ウェーブは通常、外部操作で取得した生データを格納するためにのみ使われます。
画像データの保存にも適しています。

/N フラグはデータが多次元の場合にのみ必要ですが、１次元データでも使用可能です。
次元数にかかわらず、次元サイズのリストは常に括弧内に記述する必要があります。
例：

```
WAVES/N=(5) wave1D
WAVES/N=(3,3) wave2D
WAVES/N=(3,3,3) wave3D
```

整数ウェーブは、/U フラグを使って符号なしにする場合を除き、符号付きです。

/C フラグを使う場合、行内の数値ペアは結果のウェーブにおける１つのポイントの実数部と虚数部を指定します。

既に使われているウェーブの名前を指定し、上書きオプションを使わない場合、Igor は競合を解決するためのダイアログを表示します。

/T フラグは数値ウェーブではなくテキストウェーブを生成します。詳細は
「Igor テキストファイルからテキストウェーブを読み込む」のセクションを参照してください。

Igor テキストファイル内のコマンドは、キーワード「X」とそれに続くスペースで導入されます。
コマンドは X の直後に同じ行に記述されます。
Igor テキストファイルを読み込む時にこれを検出すると、そのコマンドを実行します。

X の後に、Igor のコマンドラインから実行できるものは何でも使用できます。
コメントは「X //」で開始してください。
条件分岐やループ処理を行う方法はありません。
ただし、組み込みプロシージャまたは補助プロシージャウィンドウで定義された Igor プロシージャを呼び出すことは可能です。

X で導入されたコマンドは、コマンドラインで入力されたか、Execute コマンドによって実行されたかのように実行されます。
このようなコマンドの実行はスレッドセーフではありません。
したがって、Igor のスレッドからコマンドを含む Igor テキストファイルを読み込むことはできません。

Igor テキストファイルのスケーリングの設定

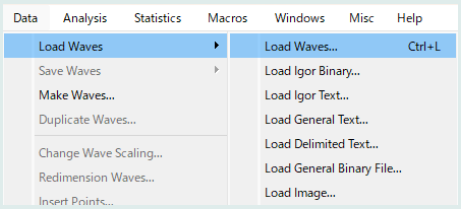
Igor が Igor テキストファイルを作成する時には、常に各ウェーブのスケール、単位、次元ラベルを設定するコマンドを含みます。
また、各ウェーブのノートも設定します。

Igor テキストファイルを生成するプログラムを作成する場合は、少なくともスケーリングと単位を設定する必要があります。
１次元データが X 次元で等間隔である場合は、SetScale コマンドを使ってウェーブの X スケーリング、X 単位、データ単位を設定する必要があります。
データが等間隔でない場合は、データ単位のみを設定してください。
多次元ウェーブの場合、必要に応じて SetScale を使って Y、Z、T の単位を設定してください。

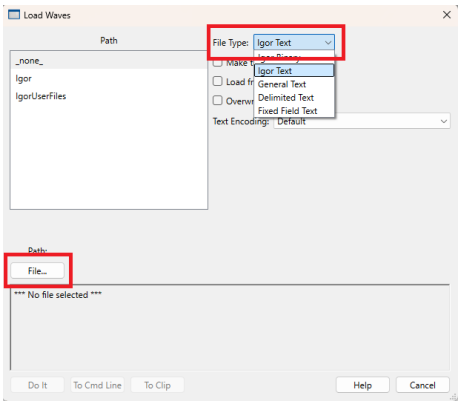
Igor テキスト用の Load Waves ダイアログ

Igor テキストファイルからデータを読み込む基本的な手順は以下の通りです。

1. **メニュー Data → Load Waves → Load Waves を選択し、Load Waves ダイアログを表示します。**



2. **File Type ポップアップメニューから Igor Text を選択します。**
3. **File ボタンをクリックして、データを含むファイルを選択します。**
4. **Do It をクリックします。**



Do It をクリックすると、Igor の LoadWave コマンドが実行されます。
これは Load Igor Text ルーチンを実行し、ファイルを読み込みます。

Data→Load Waves→Load Waves を選択する代わりに、Data→Load Waves→Load Igor Text を選択すると、Igor は Open File ダイアログを表示し、読み込む Igor テキストファイルを直接選択できます。

Igor テキストファイルから複数次元ウェーブを読み込む

Igor テキストファイルでは、ウェーブデータのブロックの前に WAVES 宣言が置かれます。
多次元データの場合、各ウェーブごとに個別のブロックを使う必要があります。
以下は 2 次元ウェーブを定義する Igor テキストファイルの例です：

```
IGOR
WAVES/D/N=(3,2) wave0
BEGIN
        1       2
        3       4
        5       6
END
```

「/N=(3,2)」フラグは、ウェーブが 3 行 2 列であることを指定します。
データの最初の行（1 と 2）は、ウェーブの最初の行のデータを格納します。
このデータレイアウトは、明瞭さのために推奨されますが、必須ではありません。
同じウェーブは以下のように作成することも可能です：

```
IGOR
WAVES/D/N=(3,2) wave0
```

```
BEGIN
        1       2               3       4               5       6
END
```

Igor は単に連続する値を読み取り、それらをウェーブに格納します。
最初の行の各列に値を格納してから、次の行へ移動します。
すべての空白文字（スペース、タブ、改行、改行コード）は同じように扱われます。

3D ウェーブを読み込む時、Igor はデータが列/行/レイヤー順であることを想定しています。
可読性のためにレイヤー間に空白行を挿入しても構いませんが、必須ではありません。

以下は 3 行×2 列×2 レイヤーのウェーブの例です：

```
IGOR
WAVES/D/N=(3,2,2) wave0
BEGIN
        1       2
        3       4
        5       6

        11      12
        13      14
        15      16
END
```

最初の 6 つの数値は、3D ウェーブの最初のレイヤーの値を定義します。
次の 6 つの数値は、 2 番目のレイヤーの値を定義します。
空白行は可読性を高めますが、必須ではありません。

4D ウェーブを読み込む時、Igor はデータが列/行/レイヤー/チャンクの順で格納されていることを想定します。
可読性のためにレイヤー間に 1 行、チャンク間に 2 行の空白行を挿入できますが、これは必須ではありません。

多次元ウェーブを読み込む場合、Igor は /N フラグで指定された次元サイズが正確であることを前提とします。
ファイル内のデータが想定より多い場合、Igor は余分なデータを無視します。
データが想定より少ない場合、結果のウェーブの一部値は未定義となります。
いずれの場合も、Igor は履歴エリアにメッセージを出力し、不一致を警告します。

Igor テキストファイルからテキストウェーブを読み込む

Igor テキストファイルからテキストウェーブを読み込む操作は、区切り文字付きテキストファイルからの読み込みと類似していますが、Igor テキストファイルではウェーブ名とウェーブタイプを宣言する必要があります。
また、Igor テキストファイルでは、Igor コマンドラインと同様にテキスト文字列は引用符で囲まれます。
以下はテキストウェーブを定義する Igor テキストの例です：

```
IGOR
WAVES/T textWave0, textWave1
BEGIN
        "This"          "Hello"
        "is"            "out"
        "a test"        "there"
END
```

Igor テキストファイルのブロック内のすべてのウェーブは、同じポイント数とデータ型を持つ必要があります。
したがって、同じブロック内で数値ウェーブとテキストウェーブを混在させることはできません。
１つの Igor テキストファイルには、任意の数のブロックを含めることができます。

この例が示すように、テキストデータブロック内の各文字列にはダブルクオートを使う必要があります。
１つのテキスト値内に引用符、タブ、キャリッジリターン、またはラインフィードを埋め込む場合は、エスケープシーケンス \", \t, \r または \n を使います。
バックスラッシュを埋め込むには \\ を使います。
あまり一般的ではないエスケープシーケンスについては、ヘルプ Escape Sequences in Strings を参照してください。

Igor バイナリデータの読み込み

このセクションでは、Igor バイナリデータをメモリに読み込む方法について説明します。

Igor はバイナリデータを２つの方法で保存します。
パックされていないエクスペリメントファイルでは１つの Igor バイナリウェーブファイルにつき１つのウェーブ、パックされたエクスペリメントファイル内では複数のウェーブを格納します。

エクスペリメントを開くと、Igor は自動的に Igor バイナリデータをロードし、エクスペリメントのウェーブを再構築します。
Igor バイナリウェーブファイルを上書きロードする主な理由は、複数の Igor エクスペリメントから同じデータにアクセスしたい場合です。
別のエクスペリメントからデータをロードする最も簡単な方法は、データブラウザーを使うことです（ヘルプ Data Browser を参照）。

注意： ２つの Igor エクスペリメントが同じ Igor バイナリウェーブファイルからデータを読み込むと問題が発生する可能性があります。詳細は「Igor バイナリウェーブファイルの共有とコピー」のセクションを参照してください。

現在のエクスペリメントデータに Igor バイナリデータをメモリから読み込む方法はいくつかあります。
以下に要約します。
ほとんどのユーザーにとって、最初の２つの方法（シンプルで使いやすい）で十分です。

エクスペリメントを開く

パックされたファイルおよびパックされていないファイルを読み込みます。

エクスペリメントを最後に保存された状態に復元します。

データブラウザー

パックされたファイルおよびパックされていないファイルを読み込みます。

あるエクスペリメントから別のエクスペリメントへデータをコピーします。

詳細はヘルプ The Browse Expt Button を参照してください。

デスクトップでのドラッグ＆ドロップ

パックされていないファイルのみ読み込みます。

あるエクスペリメントから別のエクスペリメントへデータをコピーするか、エクスペリメント間でデータを共有します。

Load Waves ダイアログ

パックされていないファイルのみ読み込みます。

あるエクスペリメントから別のエクスペリメントへデータをコピーするか、エクスペリメント間でデータを共有します。

LoadWaves コマンド

パックされていないファイルのみ読み込みます。

あるエクスペリメントから別のエクスペリメントへデータをコピーするか、エクスペリメント間でデータを共有します。

詳細はヘルプ LoadWave コマンドを参照してください。

LoadData コマンド

パックされていないファイルのみ読み込みます。

あるエクスペリメントから別のエクスペリメントへデータをコピーするか、エクスペリメント間でデータを共有します。

詳細はヘルプ LoadData コマンドを参照してください。

Igor バイナリウェーブファイル

Igor バイナリウェーブファイル形式は、Igor がウェーブを保存するためのネイティブ形式です。
この形式は、 1 ファイルに 1 つのウェーブを非常に効率的に保存します。
このファイルには、ウェーブの数値コンテンツ（テキストウェーブの場合はテキストコンテンツ）に加え、次のような補助情報もすべて含まれています。
Igor のパックされたエクスペリメントファイルでは、任意の数の Igor バイナリウェーブファイルを 1 つのファイルにパックすることができます。

Igor バイナリウェーブファイルのファイル名拡張子は「.ibw」です。
古いバージョンの Igor では「.bwav」が使われていましたが、これは現在も有効です。

ウェーブ名は Igor バイナリウェーブファイル内に保存されます。
ファイル名から取得されるものではありません。
例えば、wave0 は「wave0.ibw」というファイルに保存される場合があります。
ファイル名は任意に変更できます。
ただし、ファイル内に保存されているウェーブ名は変更されません。

Igor バイナリウェーブファイル形式は、Igor エクスペリメントの一部であるウェーブを保存するために設計されました。
パックされていないエクスペリメントの場合、ウェーブ用の Igor バイナリウェーブファイルはエクスペリメントフォルダーに保存され、LoadWave コマンドを使って読み込むことができます。

パックされたエクスペリメントの場合、Igor バイナリ形式のデータはエクスペリメントファイルにパックされ、LoadData コマンドを使って読み込むことができます。

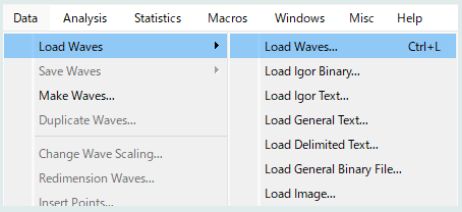
.ibw ファイルは 20 億要素を超えるウェーブをサポートしていません。
代わりに、SaveData コマンドまたはデータブラウザーの Save Copy ボタンを使って、非常に大きなウェーブをパックされたエクスペリメントファイル（.pxp）に保存できます。

一部の Igor ユーザーは、カスタムプログラムを作成し、Igor バイナリウェーブファイルを書込み、エクスペリメントに読み込んでいます。
Igor Technical Note #003「Igor Binary Format」には、プログラマーがこれを実現するために必要な詳細が記載されています。
Igor Pro Technical Note PTN003 も参照してください。

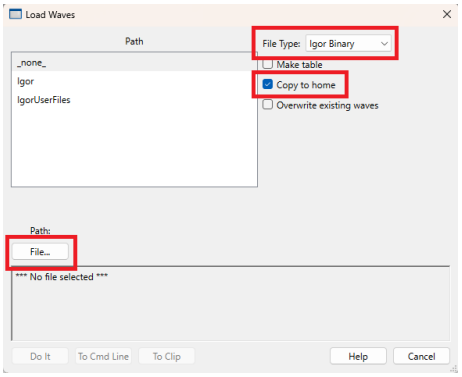
Igor バイナリ用の Load Waves ダイアログ

Igor バイナリウェーブファイルからデータを読み込む基本的な手順は以下の通りです。

1. **メニュー Data → Load Waves → Load Waves を選択し、Load Waves ダイアログを表示します。**



2. **File Type ポップアップメニューから Igor Binary を選択します。**
3. **File ボタンをクリックして、データを含むファイルを選択します。**
4. **Copy to home チェックボックスをチェックします。**
5. **Do It をクリックします。**



Do It をクリックすると、Igor の LoadWave コマンドが実行されます。
これは Load Igor Binary ルーチンを実行し、ファイルを読み込みます。
読み込もうとしているウェーブが、既存のウェーブまたは他の Igor と同じ名前である場合、Igor は競合を解決するためのダイアログを表示します。

「Load Waves」ダイアログ内の「Copy to home」チェックボックスに注意してください。
これは非常に重要です。
このチェックボックスがオンの場合、Igor はウェーブを現在のエクスペリメントに読み込んだ後、そのウェーブをソースファイルから切り離します。
次にエクスペリメントを保存すると、Igor はウェーブの新しいコピーを現在のエクスペリメントとともに保存します。
このエクスペリメントは元のソースファイルを参照しません。
これを、ウェーブを現在のエクスペリメントに「コピーする」と呼びます。

「Copy to home」がオフの場合、Igor はウェーブと読み込まれたファイル間の接続を維持します。
エクスペリメントを保存すると、ソースファイルへの参照が含まれます。
これをエクスペリメント間でウェーブを「共有する」と呼びます。

ウェーブファイルは共有するのではなく、コピーすることを強く推奨します。
詳細は「Igor バイナリウェーブファイルの共有とコピー」のセクションを参照してください。

Data→Load Waves→Load Waves ではなく、Data→Load Waves→Load Igor Binary を選択すると、Igor は Open File ダイアログを表示し、ロードする Igor バイナリウェーブファイルを直接選択できます。
これは Load Waves ダイアログをスキップするショートカットです。
このショートカットを使うと、「Copy to home」チェックボックスを設定する機会が失われます。
したがって、読み込み操作中に、Igor はウェーブをコピーするか共有するかを選択できるダイアログを表示します。

LoadData コマンド

LoadData コマンドは、Igor プログラマーがパックされた Igor エクスペリメントファイル、またはパックされていない Igor バイナリウェーブファイルを含むファイルシステムフォルダーからデータを自動的に読み込む手段を提供します。
ウェーブだけでなく、数値変数や文字列変数、さらにはウェーブと変数を含むデータフォルダーの階層構造も読み込むことができます。

Data Browser の Browse Expt ボタンは、LoadData コマンドへの対話型アクセスを提供し、ある Igor エクスペリメントから現在のメモリ内のエクスペリメントへデータ階層をドラッグすることを可能にします。
Igor プロシージャで同様の機能を実現するには、LoadData コマンドを直接使う必要があります。
LoadData コマンドのヘルプを参照してください。

LoadData は、コマンドラインまたは Data Browser からアクセスでき、既存のウェーブ、変数、データフォルダーを上書きする機能があります。
Igor は、上書きされたウェーブを表示するグラフやテーブルを自動的に更新します。
これにより、実験の連続実行によって生成されるような、同一構造のデータセットを非常に強力かつ簡単に表示できます。
まず、最初のデータセットを読み込み、それを表示するグラフやテーブルを作成します。
次に、同じ名前が付けられた連続したウェーブデータセットを読み込みます。
それらは前のデータセットを上書きし、すべてのグラフやテーブルが自動的に更新されます。

Igor バイナリウェーブファイルの共有とコピー

別の Igor エクスペリメントの一部として作成されたバイナリファイルを読み込む理由は２つあります。
現在のエクスペリメントで他のエクスペリメントとデータを共有したい場合、あるいは他のエクスペリメントから現在のエクスペリメントへデータをコピーしたい場合です。

２つのエクスペリメントが１つのファイルを共有すると、深刻な問題が発生する可能性があります。
ファイルは同時に２つの場所に存在できません。
したがって、ファイルは作成したエクスペリメントに保存されますが、他のエクスペリメントとは分離されます。

問題は、ファイルやフォルダーを移動または名前変更すると、２つ目のエクスペリメントがバイナリファイルを見つけられなくなることです。

この問題がどのように厄介な結果をもたらすかの例を以下に示します。

1. 職場でエクスペリメントを作成し、それをハードディスク上にパックされていないエクスペリメントファイルとして保存するとします。
   これを「experiment A」と呼びましょう。
   experiment A のウェーブデータは、エクスペリメントフォルダー内に個別の Igor バイナリウェーブファイルとして保存されています。

2. 新しいエクスペリメントを作成します。
   これを「experiment B」と呼びましょう。
   Load Igor Binary ルーチンを使って、experiment A から experiment B へウェーブを読み込みます。
   ウェーブを共有することを選択します。
   experiment B をハードディスクに保存します。
   experiment B には、experiment A のホームフォルダー内にあるファイルへの参照が含まれるようになりました。

3. 次に、experiment B を別のコンピューターで使用しようと決めたため、それをコピーしました。
   experiment B を開こうとすると、Igor は共有ウェーブを読み込むために必要なファイルが見つからないと通知します。
   このファイルは元のコンピューターのハードディスク上に残っているためです。

experiment B を別のコンピューターに移動する代わりに、experiment A のフォルダーの名前や場所を変更した場合にも同様の問題が発生します。
experiment B は依然として、古い名前や古い場所にある共有ファイルを探し続けるため、experiment B を開いた時に Igor はそのファイルを読み込めなくなります。

この問題のため、ファイル共有は可能な限り避けることをお勧めします。
バイナリファイルを共有する必要がある場合は、上記のような状況を避けるよう細心の注意を払う必要があります。

Data Browser は、ディスクからメモリへデータを転送する時、常にコピーを行います。

ファイル共有の問題に関する詳細については、ヘルプ References to Files and Folders を参照してください。

Copy or Share Wave ダイアログ

Igor バイナリウェーブファイルを対話的に読み込む場合（コマンド経由ではない場合）、デフォルトでは Copy or Share Wave ダイアログが表示され、ウェーブを現在のエクスペリメントにコピーするか、他のエクスペリメントと共有するかを選択できます。
Miscellaneous Settings ダイアログの Data Loading セクションで、常にコピーまたは常に共有するようにデフォルト動作を変更できます。

複数の Igor バイナリウェーブファイルを一度にインタラクティブに読み込む場合、デフォルトでは読み込むファイルごとに Copy or Share Wave ダイアログが１回ずつ表示されます。
Igor Pro 9 以降では、Apply to All Igor Binary Wave Files in the Batch Currently Being Loaded チェックボックスをオンにすることで、選択内容をすべてのファイルに適用できます。
この機能は、以下の条件を満たす場合にのみ利用可能です：

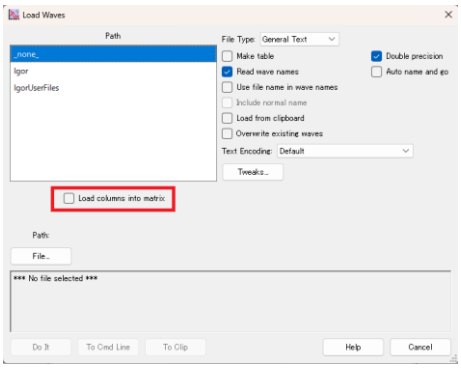
- Data→Load Waves→Load Igor Binary を選択した
- 複数の Igor バイナリウェーブファイルを Igor コマンドウィンドウにドラッグした
- 複数の Igor バイナリウェーブファイルをクリックして、Igor フレームウィンドウにドラッグした
- 複数の Igor バイナリウェーブファイルを Data Browser にドラッグした

Data Browser の Browse Expt ボタンを使って Igor バイナリウェーブファイルを読み込む場合、Copy or Share Wave ダイアログは表示されません。
この場合、ウェーブは常に現在のエクスペリメントにコピーされます。

複数の Igor バイナリウェーブファイルを読み込む時、出力変数 V_Flag、S_waveNames、S_path、S_fileName は最後に読み込まれたファイルのみを反映します。

画像ファイルの読み込み

Load Image ダイアログ（メニュー Data → Load Waves → Load Image）を使って、JEPG、PNG、TIFF、BMP、Sun Raster の画像ファイルを読み込むことができます。



画像データを含む数値プレーンテキストファイルは、Data メニューから Load Waves → Load Waves を選択して、LoadWaves ダイアログを使って読み込むことができます。
Load columns into matrix チェックボックスにチェックを入れます。

HDF5 ファイルから画像を読み込むことができます。
ヘルプを表示するには、コマンドラインで次を実行します。

```
DisplayHelpTopic "HDF5 in Igor Pro"
```

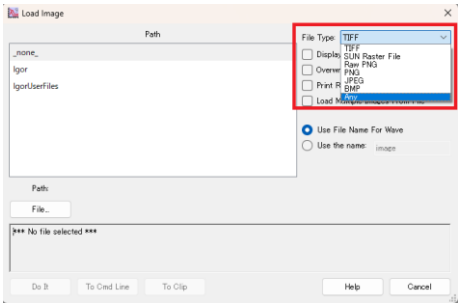
HDF4 ファイルから画像を読み込むことができます。
ヘルプを表示するには、コマンドラインで次を実行します。

```
DisplayHelpTopic "HDF Loader XOP"
```

フレームを囲んで画像を読み込むこともできます。
NewCamera コマンドのヘルプを参照してください。

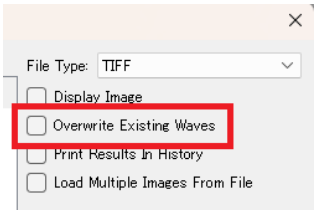
Load Image ダイアログ

ウェーブに画像ファイルをロードするには、メニュー Data → Load Waves → Load Image を選択して、Load Image ダイアログを表示します。



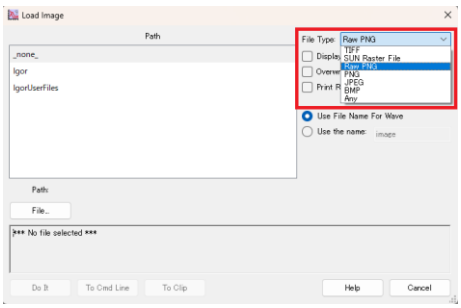
File Type ポップアップメニューから、特定の種類の画像ファイルを選択すると、画像ファイル選択ダイアログを表示するときに使われるファイルフィルターが設定されます。
画像ファイルのファイル拡張子が正しいかどうかわからない場合は、フィルターが選択を制限しないように File Type ポップアップメニューから Any を選択します。

読み込まれたウェーブの名前は、ファイル名または指定した名前にすることができます。
ダイアログボックスに既存のウェーブ名と競合する名前を入力し、Overwrite Existing Waves チェックボックスにチェックを入れない場合、新しいウェーブ名に数字のサフィックスが付けられます。



PNG ファイルを読み込む

PNG 形式には、Raw PNG と PNG の 2 つのメニューオプションがあります。
Raw PNG が選択された場合、データはファイルから直接ウェーブに読み込まれます。
PNG が選択されると、ファイルはメモリにロードされ、オフスクリーン画像が作成され、ウェーブデータはオフスクリーン画像を読み込むことで設定されます。
ほとんどの場合、Raw PNG を選択すべきです。



PNG ファイルを読み込む時には、画像データは、レイヤー 0、1、2 の符号なしバイト RGB 要素を含む 3D Igor RGB ウェーブに読み込まれます。
画像ファイルにアルファチャンネルが含まれている場合は、生成される 3D RGBA ウェーブにはアルファレイヤーが含まれます。

RGB 画像を含む 3D ウェーブをグレースケール画像に変換するには、ImageTransform コマンドで rgb2gray キーワードを使います。

JPEG ファイルを読み込む

JPEG ファイルを読み込む時には、画像データは、レイヤー 0、1、2 の符号なしバイト RGB 要素を含む 3D Igor RGB ウェーブに読み込まれます。
JPEG はアルファをサポートしていません。

RGB 画像を含む 3D ウェーブをグレースケール画像に変換するには、ImageTransform コマンドで rgb2gray キーワードを使います。

BMP ファイルを読み込む

BMP ファイルを読み込む時には、画像データは、レイヤー 0、1、2 の符号なしバイト RGB 要素を含む 3D Igor RGB ウェーブに読み込まれます。
BMP はアルファをサポートしていません。

RGB 画像を含む 3D ウェーブをグレースケール画像に変換するには、ImageTransform コマンドで rgb2gray キーワードを使います。

TIFF ファイルを読み込む

TIFF ファイルは、多くのフォーマットで１つ以上の画像を保存できます。
もっとも一般的なものは次です：

- ２値 — 各ピクセルが黒または白を表現できる、１つの平面データで構成されます。
  Igor は２値画像を 2D ウェーブに読み込みます。
- グレースケール — 各ピクセルが強度の範囲を表現できる、１つの平面データで構成されます。
  Igor はグレースケール画像を 2D ウェーブに読み込みます。
- パレットカラー — グレースケール画像に似ていますが、カラーパレットを含んでいます。
  Igor はグレースケール画像を 2D ウェーブに読み込み、さらに「_CMap」というサフィックスのついたカラーマップウェーブを作成します。
- フルカラー（RGB、RGBA、CMYK） — ３レイヤーまたは４レイヤーの 3D ウェーブに読み込みます。
  各レイヤーには１つの色要素のピクセルが格納されます。

複数の画像を含む TIFF ファイルは TIFF スタックと呼ばれます。
読み込みには２つの方法があります。

- 画像を１つの 3D ウェーブに読み込みます。
  これはグレースケール画像のみで動作します。
  各グレースケール画像は、3D 出力ウェーブのレイヤーに読み込まれます。
- 各画像をそれぞれのウェーブに読み込みます。
  これはあらゆる種類の画像で機能します。
  グレースケール画像はそれぞれ別の 2D ウェーブに読み込まれます。
  RGB、RGBA、CMYK 画像は、それぞれ別の 3D ウェーブに読み込まれます。

マルチイメージ TIFF ファイルから読み込む特定の画像、または画像の範囲を指定することができます。
Load Image ダイアログで、読み込む最初の画像のゼロから始まるインデックスと、TIFF スタックから読み込む画像の数を入力します。

TIFF 画像は NewImage コマンドで表示でき、ImageTransform コマンドで画像ウェーブを他の形式に変換できます。

RGB 画像を含む 3D ウェーブをグレースケール画像に変換するには、ImageTransform コマンドで rgb2gray キーワードを使います。

ImageTransform コマンドで stackImages キーワードを使うと、複数の 2D 画像ウェーブを 3D スタックに変換することができます。

Sun Raster ファイルを読み込む

Sun Raster ファイルは 2D ウェーブに読み込まれます。
Sun Raster ファイルにカラーマップが含まれている場合、画像ウェーブに加えて、サフィックス「_CMap」のついたカラーマップウェーブを作成します。

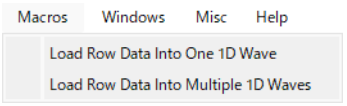
行指向テキストデータの読み込み

組み込みのテキストローダーはすべて列指向です。
つまり、ファイル内のデータ列を 1D ウェーブにロードします。
かなり一般的に、行指向のフォーマットがあります。
このフォーマットでは、ファイルは１つのウェーブのデータを表しますが、複数の列に書き込まれます。
次がその一例です。

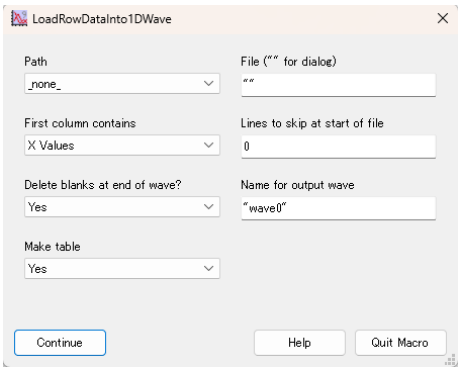
```
350     2.97    1.95    1.00    8.10    2.42
351     3.09    4.08    1.90    7.53    4.87
352     3.18    5.91    1.04    6.90    1.77
```

この列では、最初の列には X の値が含まれ、残りの列にはデータ値が含まれ、行/列の順序で書かれています。

Igor には、この形式を処理するファイルローダーエクステンションはありませんが、この目的のための WaveMetrics 製のプロシージャファイルがあります。
これを使うには、WaveMetrics Procedures:File Input Output フォルダー内の Load Row Data プロシージャファイルを使います（Load Row Data.ipf ファイルを Igor Procedures フォルダーにコピーします）。



これを使うと、Macro メニューに Load Row Data という項目が追加されます。
これを選択すると、Igor はいくつかのオプションを提供するダイアログを表示します。
オプションの１つは、最初の列を X 値またはデータとして扱います。
列を X 値として指定すると、最初の列の値が等間隔であると仮定して、出力ウェーブの X スケーリングを決定するためにそれを使います。
通常、これが一般的です。



Excel ファイルの読み込み

Excel ファイルからデータを読み込むには、XLLoadWave コマンドを直接使うか、メニュー Data → Load Waves → Load Excel File を選択して、Load Excel File ダイアログを表示します。

XLLoadWave は、数値、テキスト、日付、時刻、日付/時刻データを Excel ファイルからウェーブに読み込みます。
.xls と .xlsx ファイルからデータを読み込むことができます。
.xlsb（大きなファイル用のバイナリ形式）ファイルには対応していません。
また、パスワードで保護された Excel ファイルも読み込めません。

Igor にロードする前に、Excel のワークシートを閉じておく必要があります。

一部のプログラムではタブ区切りやその他の Excel 以外の形式のファイルを .xls という拡張して保存してしまうことがあります。

これらのファイルのいずれかをロードしようとすると、XLLoadWave はそれが Excel のバイナリファイルではないことを表示します。

XLLoadWave が読み込むもの

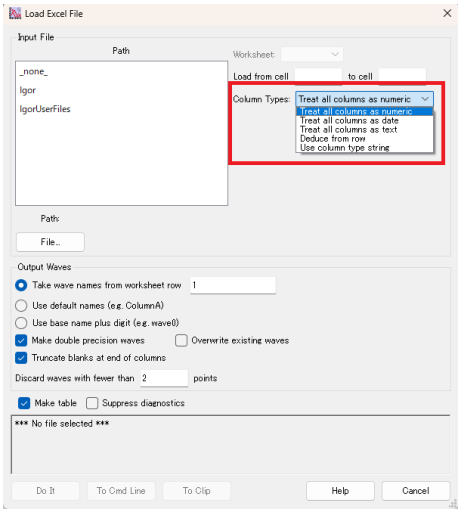
ワークシートは、長方形の数字ブロックだけで構成される非常にシンプルなものでも、数値、文字列、数式のブロックが任意の方法で混在する非常に複雑なものでも構いません。
XLLoadWave は、ワークシートから矩形のセルブロックを抜き出して、列をウェーブに変換するように設計されています。

XLLoadWave は数値データとテキスト（文字列）データの両方を読み込むことができます。
Excel の列には、数値セルとテキストセルを混在させることができます。
ウェーブは、すべて数値またはすべてテキストでなければなりません。
Excel の列をウェーブに読み込む場合、数値ウェーブまたはテキストウェーブのどちらにデータを読み込むかを決定する必要があります。
XLLoadWave は、日付、時刻、日付/時刻データを数値ウェーブにロードすることもできます。

列とウェーブの形式

XLLoadWave は、指定された列に対して作成するウェーブの形式を決定する次の方法を提供します。
これらの方法は、Load Excel File ダイアログに表示され、XLLoadWave コマンドの /C および /COLT フラグによってコントロールされます。

Treat all columns as numeric（すべての列を数値として扱う）

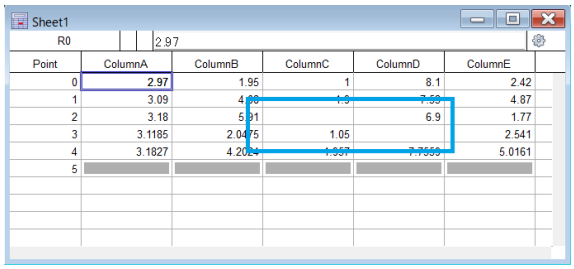


これがデフォルトの方法です。
ウェーブに読み込みたい単純な数字のブロックがある場合は、この方法を使います。

XLLoadWave は、読み込む Excel の各列に対して数値ウェーブを作成します。
列に数値セルが含まれている場合、それらの値はウェーブの対応するポイントに格納されます。
列にテキストセルが含まれている場合、XLLoadWave はウェーブの対応するポイントに NaN（空白）を格納します。

読み込むデータ（Excel：左）と読み込み後（Igor Pro：右）

	A	B	C	D	E	F
1	2.97	1.95	1.00	8.10	2.42	
2	3.09	4.08	1.90	7.53	4.87	
3	3.18	5.91	AAAA	6.90	1.77	
4	3.12	2.05	1.05	BBBB	2.54	
5	3.18	4.26	1.96	7.76	5.02	
6						



Treat all columns as date（すべての列を日付として扱う）

これは、XLLoadWave が Excel の日付/時刻フォーマットの数値データを Igor の日付/時刻フォーマットに変換する点を除いては、前述の方法と同じです。
詳細は、数セクション後の「Excel Date/Time vs. Igor Date/Time」を参照してください。

読み込むデータ（Excel：左）と読み込み後（Igor Pro：右）

	A	B	C
1	2024/12/10 8:00	2024/12/10 11:00	
2	2024/12/11 8:00	2024/12/11 11:00	
3	2024/12/12 8:00	2024/12/12 13:00	
4	2024/12/13 8:00	2024/12/13 13:00	
5	2024/12/14 8:00	2024/12/14 20:00	
6			

Sheet1
R0 2024/12/10 08:00:00

Point	ColumnA	ColumnB
0	2024/12/10 08:00:00	2024/12/10 11:00:00
1	2024/12/11 08:00:00	2024/12/11 11:00:00
2	2024/12/12 08:00:00	2024/12/12 13:00:00
3	2024/12/13 08:00:00	2024/12/13 13:00:00
4	2024/12/14 08:00:00	2024/12/14 20:00:00
5		

XLLoadWave が日付や時刻を保存する数値ウェーブを作成する場合、日付を正確に保存するには倍精度が必要であるため、常に倍精度ウェーブが作成されます。
また、XLLoadWave はウェーブのデータ単位を「dat」に設定します。
グラフで XY ペアの X 軸としてウェーブを使う場合、「dat」を日付および/または時刻を含むことを示す記号として認識します。

この方法では、XLLoadWave がテーブルにウェーブを表示する場合、テーブルの列に日付/時刻フォーマットが使われます。
ModifyTable コマンドを使って、列のフォーマットを日付のみ、または時刻のみに変更することができます。

Treat all columns as text（すべての列をテキストとして扱う）

XLLoadWave は、すべての列をテキストウェーブにロードします。

数値セルを含む列をテキストウェーブに読み込むと、数値セルの値をテキストに変換し、その結果のテキストをウェーブに格納します。

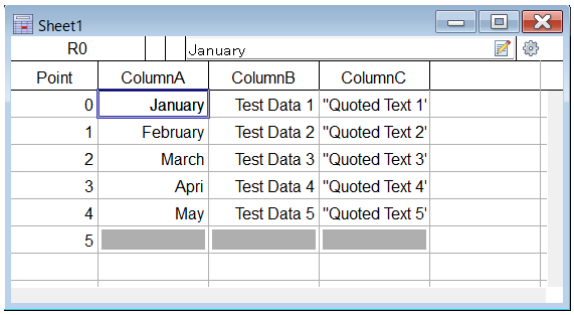
デフォルトの設定は、１行目を列ラベルとして使います。
つまり結果は次のようになります。
読み込むデータ（Excel：左）と読み込み後（Igor Pro：右）

	A	B	C	D
1	January	Test Data 1	"Quoted Text 1"	
2	February	Test Data 2	"Quoted Text 2"	
3	March	Test Data 3	"Quoted Text 3"	
4	Apri	Test Data 4	"Quoted Text 4"	
5	May	Test Data 5	"Quoted Text 5"	
6				

Sheet1
R0 February

Point	January	Test_Data_1	X_Quoted_Text_1
0	February	Test Data 2	"Quoted Text 2"
1	March	Test Data 3	"Quoted Text 3"
2	Apri	Test Data 4	"Quoted Text 4"
3	May	Test Data 5	"Quoted Text 5"
4			

このような場合、Load Excel File ダイアログで、Use default names (e.g. ColumnA) または Use base name plus digit (e.g. wave0) を選択して 1 行目をデータとして処理するように設定します。



Point	ColumnA	ColumnB	ColumnC
0	January	Test Data 1	"Quoted Text 1'
1	February	Test Data 2	"Quoted Text 2'
2	March	Test Data 3	"Quoted Text 3'
3	Apri	Test Data 4	"Quoted Text 4'
4	May	Test Data 5	"Quoted Text 5'
5			

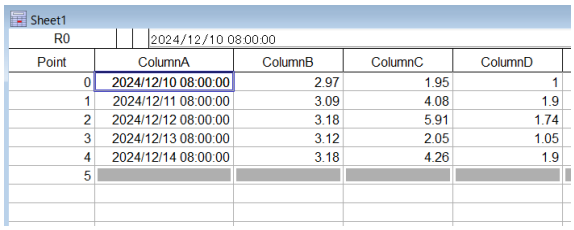
Deduce from row（行から推測する）

これは、異なるタイプの列（数値、日付、テキスト）の混合を読み込むのに適した方法です。

XLLoadWave にどの行を調べるかを指定します。
XLLoadWave はその行のセルを調べます。
指定された列について、セルが数値の場合、XLLoadWave は数値ウェーブを作成し、セルがテキストの場合、テキストウェーブを作成します。

読み込むデータ（Excel：左）と読み込み後（Igor Pro：右）

	A	B	C	D	E	F	G
1	2024/12/10 8:00	2.97	1.95	1.00	8.10	2.42	
2	2024/12/11 8:00	3.09	4.08	1.90	7.53	4.87	
3	2024/12/12 8:00	3.18	5.91	1.74	6.90	1.77	
4	2024/12/13 8:00	3.12	2.05	1.05	8.21	2.54	
5	2024/12/14 8:00	3.18	4.26	1.90	7.70	5.02	
6							



Point	ColumnA	ColumnB	ColumnC	ColumnD
0	2024/12/10 08:00:00	2.97	1.95	1
1	2024/12/11 08:00:00	3.09	4.08	1.9
2	2024/12/12 08:00:00	3.18	5.91	1.74
3	2024/12/13 08:00:00	3.12	2.05	1.05
4	2024/12/14 08:00:00	3.18	4.26	1.9
5				

数値セルに Excel の組み込みの日付、時刻、日付/時刻フォーマットが使われている場合、XLLoadWave は Excel の日付/時刻フォーマットの数値データを Igor の日付/時刻フォーマットに変換します。
XLLoadWave は、カスタムのセルフォーマットによってコントロールされているセルの日付と時刻のフォーマットを推測することはできません。
この場合、手動での変換方法の詳細については、後のセクション「Excel Date/Time vs. Igor Date/Time」を参照してください。

XLLoadWave がこの方法で列タイプを推測する場合、Excel ファイル内の対応する列のビルトインセルのフォーマットに応じて、日付/時刻ウェーブ用の Igor のテーブルの列形式を日付、時刻、日付/時刻のいずれかに設定します。

Use column type string（列形式文字列を使う）

異なるタイプの列（数値、日付、テキスト）が混在していて、「deduce from row」の方法で正しい推測ができない場合に、この方法を使います。
例えば、一部のファイルでは、列タイプを推測するのに適した行が 1 つもない場合があります。

この方法では、読み込む各列の形式を識別する文字列を指定します。
例えば、「1T1D3N」という文字列は、
　1T：最初の列をテキストウェーブに読み込む
　1D：次の列を日付/時刻ウェーブに読み込む
　3N：次の 3 列を数値ウェーブに読み込む
ことを意味します。
文字列でカバーされているよりも多くの列をロードした場合、余分な列は数値としてロードされます。
また、
　N：すべての列が数値である

D：すべての列が日付/時刻である
T：すべての列がテキストである
ことを意味します。
文字列には空白やその他の余分な文字を含めてはいけません。

正しい文字列の例を次に示します。

N	すべての列が数値
T	すべての列がテキスト
1T1D3N	１列目がテキスト、２列目が日付/時刻、次の３列が数値
1T1N3T25N	１列目がテキスト、２列目が数値、次の３列がテキスト、その次の 25 列が数値

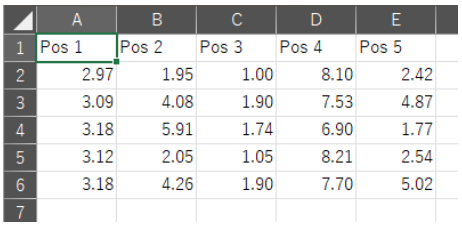
テキスト	数値	テキスト	テキスト	テキスト	数値	数値	以降数値

treat all columns as numeric 設定では、数値列内のテキストセルは空白として扱われます。
この動作は、XLLoadWave が導入される以前のバージョンと互換性があります。
use column type string 設定では、数値列内のテキストセルを検出すると、そのテキストセルを数値に変換します。
テキストが有効な数値を表している場合（例：”1.234”）、ウェーブに有効な数値が生成されます。
テキストが有効な数値を表していない場合（例：”January”）、ウェーブには空白が生成されます。
これは、不注意で数値列にテキストセルを含むファイルがある場合に便利です。

XLLoadWave とウェーブ名

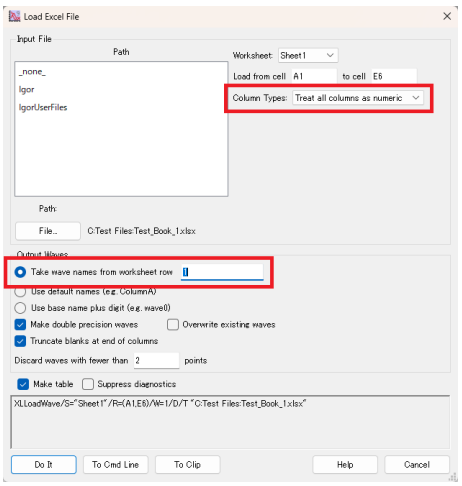
Load Excel File ダイアログで見ることができるように、XLLoadWave は作成するウェーブ名を生成するときに、３つの方法のうちの１つを使います。

1）ワークシートで指定した行からウェーブ名を取得することができます。
この場合、XLLoadWave は行に文字列値が含まれていることを想定しています。

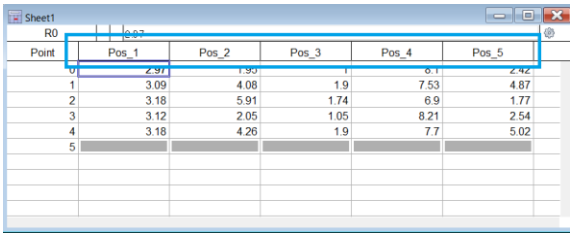


	A	B	C	D	E
1	Pos 1	Pos 2	Pos 3	Pos 4	Pos 5
2	2.97	1.95	1.00	8.10	2.42
3	3.09	4.08	1.90	7.53	4.87
4	3.18	5.91	1.74	6.90	1.77
5	3.12	2.05	1.05	8.21	2.54
6	3.18	4.26	1.90	7.70	5.02
7					

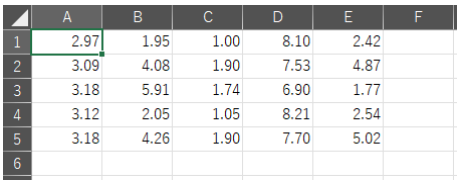
Load Excel File ダイアログで、Take wave names from worksheet row を選択し、名前を取得する行番号を入力します。



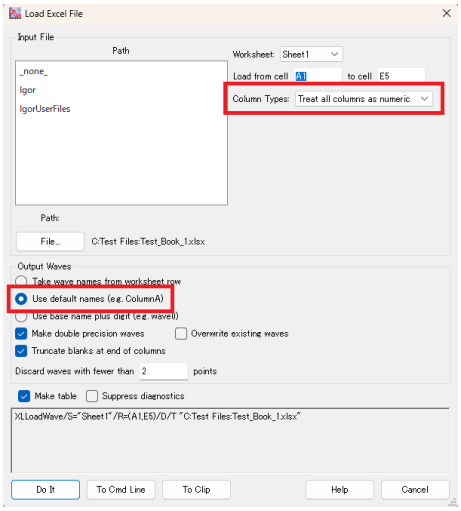
結果は次のようになります。



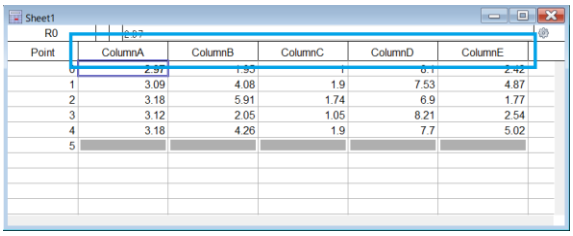
2）デフォルトのウェーブ名として、ColumnA、ColumnB などの形式で生成することができます。
名前の末尾の文字は、そのウェーブが作成されたワークシートの列を示します。



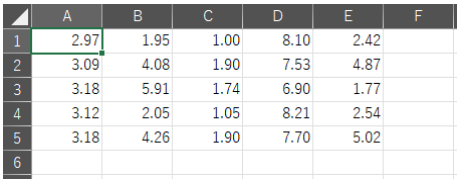
Load Excel File ダイアログで、Use default names (e.g. ColumnA) を選択します。



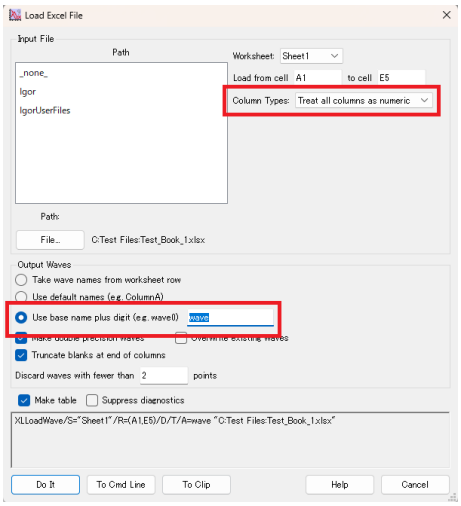
結果は次のようになります。



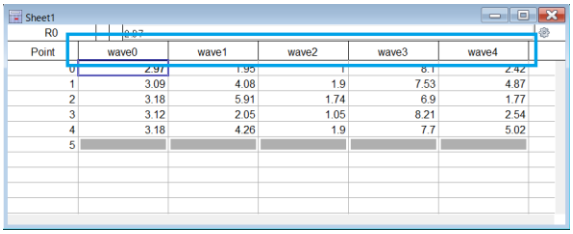
3）XLLoadWave は、この場合「wave」というベース名を使って、wave0、wave1 などの形式のウェーブ名を生成することができます。



Load Excel File ダイアログで、Use base name plus digit (e.g. wave0) を選択し、ベースとなる名前（ここでは wave）を入力します。



結果は次のようになります。



XLLoadWave コマンドは、ダイアログでは使うことができない４番目のウェーブの命名方法、/NAME フラグをサポートしています。
このフラグを使うと、セミコロンで区切られた文字列のリストを使って、各列の名前を指定することができます。

XLLoadWave が作成するウェーブの名前が、以下に説明するいくつかの状況により、予想とは異なるものになる場合があります。
このような場合、XLLoadWave は履歴エリアに元の名前と新しい名前を表示します。
読み込み後、必要であれば、Rename コマンドを使って、別の名前を指定することができます。

ワークシート内の名前が長すぎる場合
XLLoadWave はそれを適切な長さに切り詰めます。

標準のウェーブ名で許可されていない文字が名前に含まれている場合
XLLoadWave はその文字をアンダースコアに置き換えます。

ワークシート内の２つの名前が競合する場合
XLLoadWave は「D_」などのプリフィックスを追加することで、２番目の名前をユニークなものにします。
この「D_」は、ウェーブに読み込まれる Excel の列を示しています。

ワークシート内の名前が既存のウェーブの名前と競合する場合
上書きオプションを使わない限り、XLLoadWave は１つ以上の数字を追加することで、読み込むウェーブの名前をユニークにします。
上書きオプションがオンの場合、読み込まれたデータは既存のウェーブを上書きします。

XLLoadWave が名前をユニークにするために１つ以上の数字を追加する必要があるが、名前の長さが既にウェーブ名の限界に達している場合
名前の途中から１つ以上の文字を削除します。

ワークシートのセルから取得した名前が、コマンド、関数、マクロの名前と競合する場合があります。
例えば、Date と Time は組み込み関数であるため、ウェーブにこれらの名前を付けることはできません。
このような競合が発生した場合、XLLoadWave は名前を変更し、履歴エリアに元の名前と新しい名前を示すメッセージを表示します。

XLLoadWave の出力変数

XLLoadWave は、ファイルローダーの出力変数である V_flag、S_path、S_fileName、S_waveNames を標準として設定します。
さらに、S_worksheetName をワークブックファイル内の、ロードされたワークシートの名前に設定します。

Excel Date/Time vs. Igor Date/Time

Excel は、1900 年 1 月 1 日または 1904 年 1 月 1 日からの日数単位で日付/時刻情報を保存します。
Windows では 1900 年（MacOS では 1904 年）がデフォルトです。
Igor は、1904 年 1 月 1 日からの秒単位で日付を保存します。

列タイプを決定するときに、Treat all columns as date、Deduce from row、Use column type string を使うと、XLLoadWave が Excel 形式から Igor 形式に自動的に変換します。
Treat all columns as numeric を使う場合は、Excel 形式から Igor 形式に手動で変換する必要があります。

Excel ファイルが 1904 年を基準年として使っている場合（MacOS）は、変換は次のようになります。

```
wave *= 24*3600                 // 日を秒に変換（wave = wave*24*3600）
```

Excel ファイルが 1900 年を基準年として使っている場合（Windows）は、変換は次のようになります。

```
wave *= 24*3600                 // 日を秒に変換（wave = wave*24*3600）
wave -= 24*3600*365.5*4         // ４年の差分を処理
```

365 ではなく 365.5 を使っているのは、閏年を考慮したためです。
Microsoft の 1900 の日付システムは 1900 年 1 月 1 日を 0 日ではなく、1 日としています（うるう年の日と開始日の日の 2 日分が必要）。

時刻データを表形式で表示する場合、Excel と Igor で 1 秒の誤差が生じる場合があります。
例えば、Excel では「9:00:30」と表示される場合でも、Igor では「9:00:29」と表示されることがあります。
これは、Excel のデータが基準時間（Nominal time）にわずかに満たないためです。
この例では、Excel のセルには「9:00:30」からミリ秒を引いた値が含まれます。
Excel が時刻を表示するときには、四捨五入されます。
Igor が時刻を表示するときには切り捨てられます。
これが気になる場合は、Igor のウェーブのデータを丸めることができます。

```
wave = round(wave)
```

この四捨五入を行うことで、データ内の小数点以下の秒数はすべて削除されます。
そのため、XLLoadWave では自動的に丸め処理は行いません。

Excel Data を 2D ウェーブに読み込む

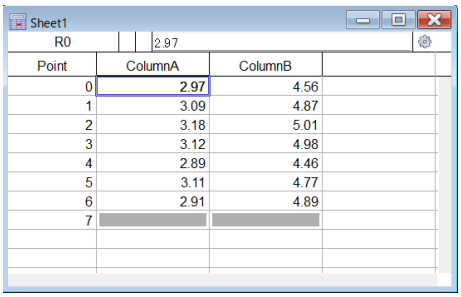
XLLoadWave は 1D ウェーブを作成します。1D ウェーブを 2D ウェーブに変換する関数を以下に示します。

```
Function LoadExcelNumericDataAsMatrix(pathName, fileName, worksheetName,
                        startCell, endCell)
        String pathname         // Igor シンボリックパスまたは "" でダイアログを表示
        String filename         // 読み込むファイル名または "" でダイアログを表示
        String worksheetName
        String startCell        // 例： "B1"
        String endCell          // 例： "J100"

        if ((strlen(pathName)==0) || (strlen(fileName)==0))
                // ファイルを指定するダイアログを表示
                Variable refNum
                String filters = "Excel Files (*.xls,*.xlsx,*.xlsm):.xls,.xlsx,.xlsm;"
                filters += "All Files:.*;"
                Open/D/R/P=$pathName /F=filters refNum as filename
                fileName = S_fileName           // S_fileName は Open/D によって設定される
                if (strlen(fileName) == 0)      // ユーザーがキャンセル？
                        return -2
                endif
        endif

        // 行 1 を数値ウェーブに読み込む
        XLLoadWave/S=worksheetName/R=($startCell,$endCell)/COLT="N"/O/V=0/K=0/Q filename
        if (V_flag == 0)
                return -1                       // ユーザーがキャンセル
        endif

        String names = S_waveNames              // S_waveNames は XLLoadWave によって作成される
        String nameOut = UniqueName("Matrix", 1, 0)
        Concatenate /KILL /O names, $nameOut    // マトリックスを作成し、1D ウェーブをキルする
```

```
        String format = "Created numeric matrix wave %s containing cells %s to %s in
                                worksheet \"%s\"\r"
        Printf format, nameOut, startCell, endCell, worksheetName
End
```

1D ウェーブとして読み込んだ場合：読み込むデータ（Excel：左）と読み込み後（Igor Pro：右）

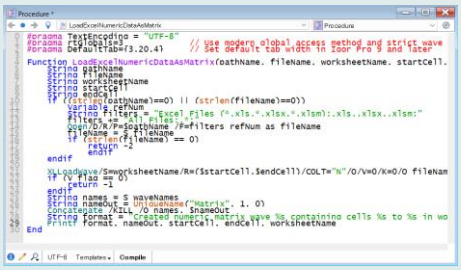
	A	B	C
1	2.97	4.56	
2	3.09	4.87	
3	3.18	5.01	
4	3.12	4.98	
5	2.89	4.46	
6	3.11	4.77	
7	2.91	4.89	
8			



上記の関数を使うステップは次の通りです。

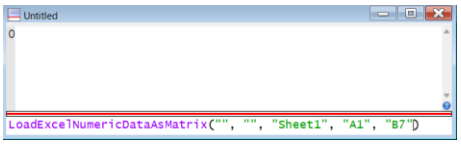
1. **メニュー Windows → Procedure Windows → Procedure Window を選択し、上記のコードを貼り付けます（改行やコメントなどを整理しておくとよい）。**

画面下の Compile ボタンを押します。



2. **今回は、ファイルはダイアログから選択できるような形にするため、最初の２つの引数は "" として、コマンドラインに次を入力して実行します。**

```
LoadExcelNumericDataAsMatrix("", "",
        "Sheet1", "A1", "B7")
```



3. **ファイルを選択するダイアログが開くので、Excel ファイルを選択します。**

Data Browser には１つのウェーブのみが表示され、ダブルクリックしてテーブルを表示させると 2D ウェーブになっていることがわかります。