CONTENTS

Volume II User's Guide: Part1 （Importing and Exporting Data #2）

Loading General Text（一般的なテキストファイルの読み込み）

Igor Pro マニュアル：II-138 ページ以降をもとに編集

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 を使用します。
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 ルーチンで説明した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、連続する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。
同じ数字の列を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：

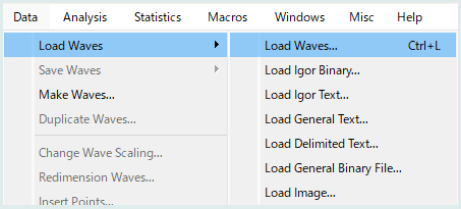
- ブランク（欠損値）を処理できない
- 数値列内に数値以外のテキストや値の列を許容できない
- テキスト値、日付、時刻、日付/時刻をロードできない
- 小数点としてカンマを処理できない（ヨーロッパの数字形式）

一般的なテキストの読み込みルーチンは、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。

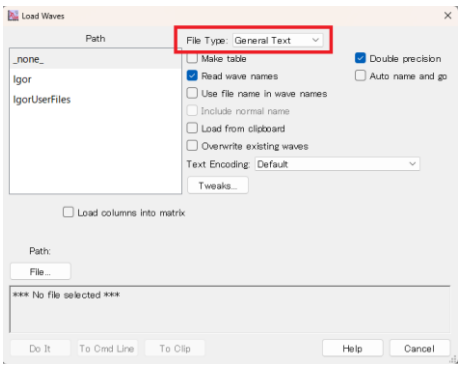
一般的なテキストに対する Load Wave ダイアログ – 1D

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。**

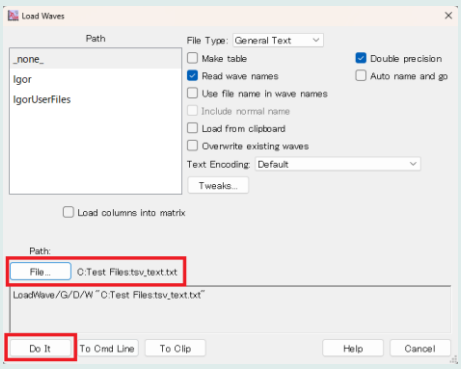


2. **File Type ポップアップメニューから General Text を選択します。**



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。

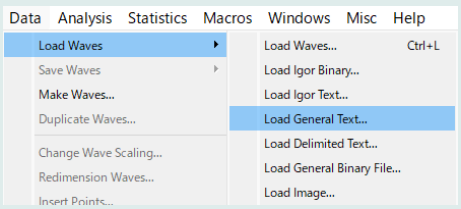


Do It をクリックすると、LoadWave コマンドが起動します。
Load General Text ルーチンが次のステップで実行されます。

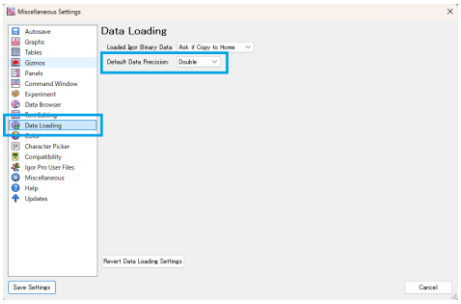
1. 連続する行の数を数える処理を行って、データのブロックの開始位置を見つける
   このステップでは、ヘッダーがある場合はそれをスキップし、ブロックの列数を決定する
2. オプションで、数値のブロックの直前に列ラベルの行があるかどうかを決定する
3. オプションで、ウェーブの名前の確認または変更ができる別のダイアログを表示する
4. ウェーブを作成する
5. ファイルの最後まで、または数値の数が異なる行が見つかるまで、データをウェーブに読み込む
6. ファイルの終わりでない場合は、ステップ 1 に戻って別のブロックを探す

「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。



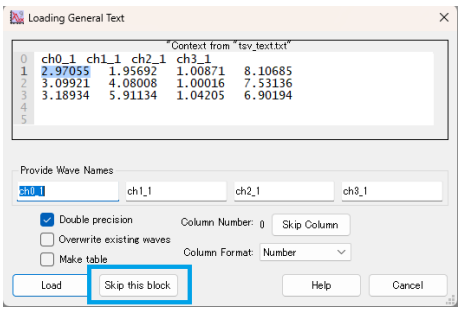
数値ウェーブの精度は、Miscellaneous Settings ダイアログ（Misc メニュー）の Data Loading セクションにある Default Data Precision 設定によってコントロールされます。
このショートカットを使う前に、Load Waves ダイアログで使うことができるオプションを確認してください。



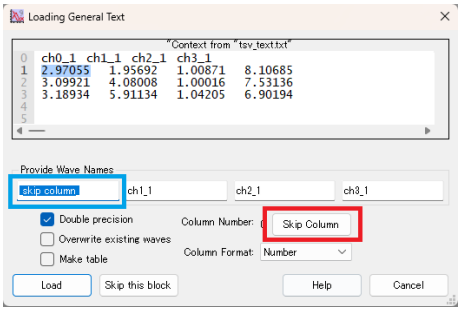
ブロックに対するウェーブ名の編集

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。



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。

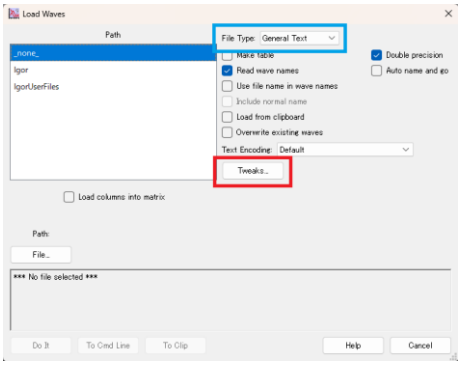
注記： データが等間隔の場合、ウェーブの X スケーリングを設定することが非常に重要です。
Igor の多くのコマンドは、正しい結果を得るために X スケーリング情報に依存しています。

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）

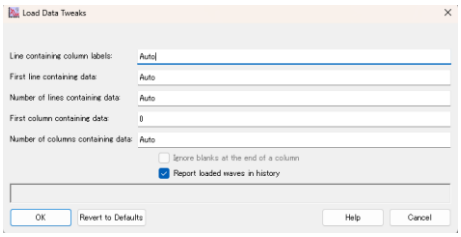
Load General Text ルーチンでは、ファイルを読み込む時に、処理を指示するための微調整（Tweak）ができるようになっています。

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、マニュアル II-137 Troubleshooting Delimited Text Files 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 Pro マニュアル：II-142 ページ以降をもとに編集

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、マニュアル II-145 Specifying Characteristics of Individual Columns を参照してください。

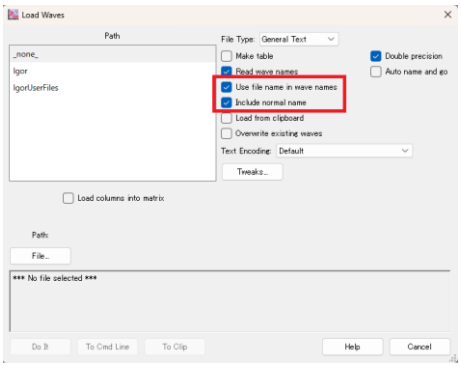
/NAME フラグ

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、マニュアル III-501 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 normal name 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  normal name 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 normal name 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、マニュアル II-176 Setting Wave Names When Loading Data Files を参照してください。

LoadWave のその他の機能

Igor Pro マニュアル：II-144 ページ以降をもとに編集

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、マニュアル II-142 LoadWave Generation of Wave Names 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用されています。

```
/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１６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

Igor Pro マニュアル：II-149 ページ以降をもとに編集

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、マニュアル III-459 Text Encoding 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、マニュアル III-475 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、マニュアル II-490 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、マニュアル III-467 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 Dialog 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。