

---

# Open Platform of Transparent Analysis Tools for fNIRS

## 付録:FAQ

国立研究開発法人 産業技術総合研究所

---

よくある質問と回答をまとめます。

### 解析

Questions	Answer
複数被験者間で T 検定をしたい	Multi Analysis で T 検定します
複数被験者の平均信号を取得したい	Multi Analysis で File I/O Move Average を実施
Group 解析 Correlation Analysis を行いたい	Developer モードで 1 <sup>st</sup> , 2 <sup>nd</sup> 解析を行います

### 表示

Questions	Answer
描画結果の Line プロパティを変更したい	メニューで設定してください
描画する Line プロパティを変更したい	Layout を変更してください

### データ管理

Questions	Answer
解析結果を CSV ファイルに出力したい	Export WS 機能を使ってください
データを追加したい	

## 1. 解析

### 1.1. 複数被験者間で T 検定をしたい

#### 1.1.1. Answer

Multi Analysis で T 検定します。

但し、被験者間要因を無視するので、必ずしも正確な解析ではありません。

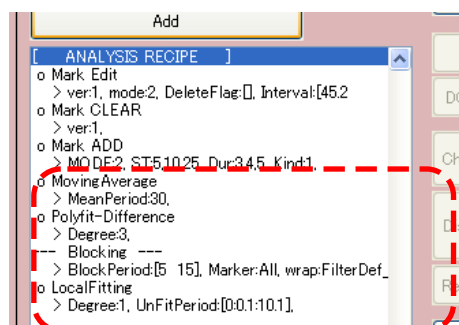
#### 1.1.2. 実行例

解析は POTATo の Developer モードで行います。

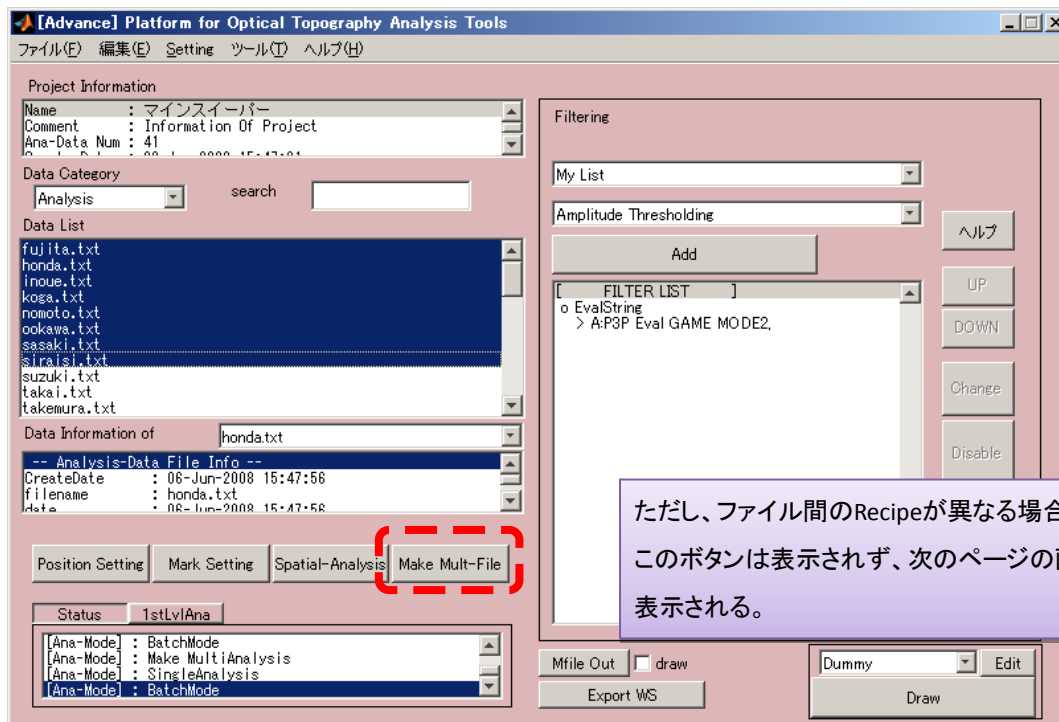
最初に解析データに対し、前処理として Blocking や LocalFitting を設定します。

このとき、Blocking はそれぞれのファイルに対し同じ値を設定してください。

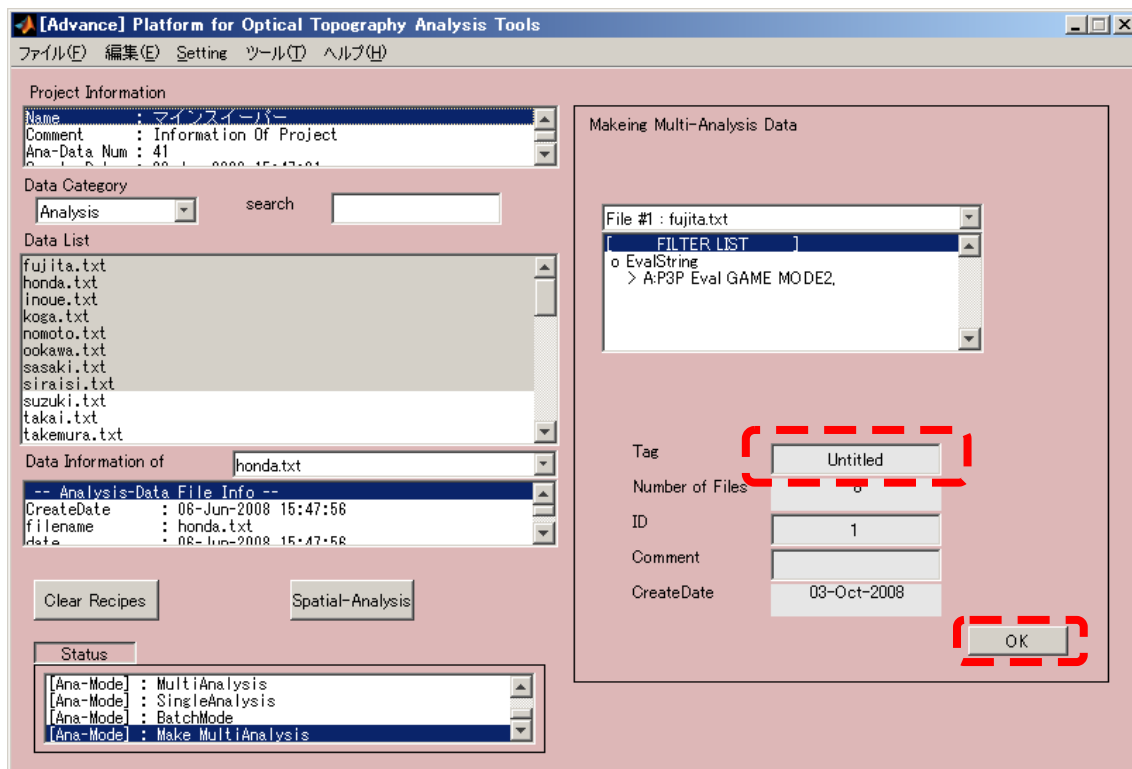
(右の例では、マーカーの設定が含まれています)



目的の解析データを選択し、Make Multi-File ボタンを押し Multi-File を作成します。

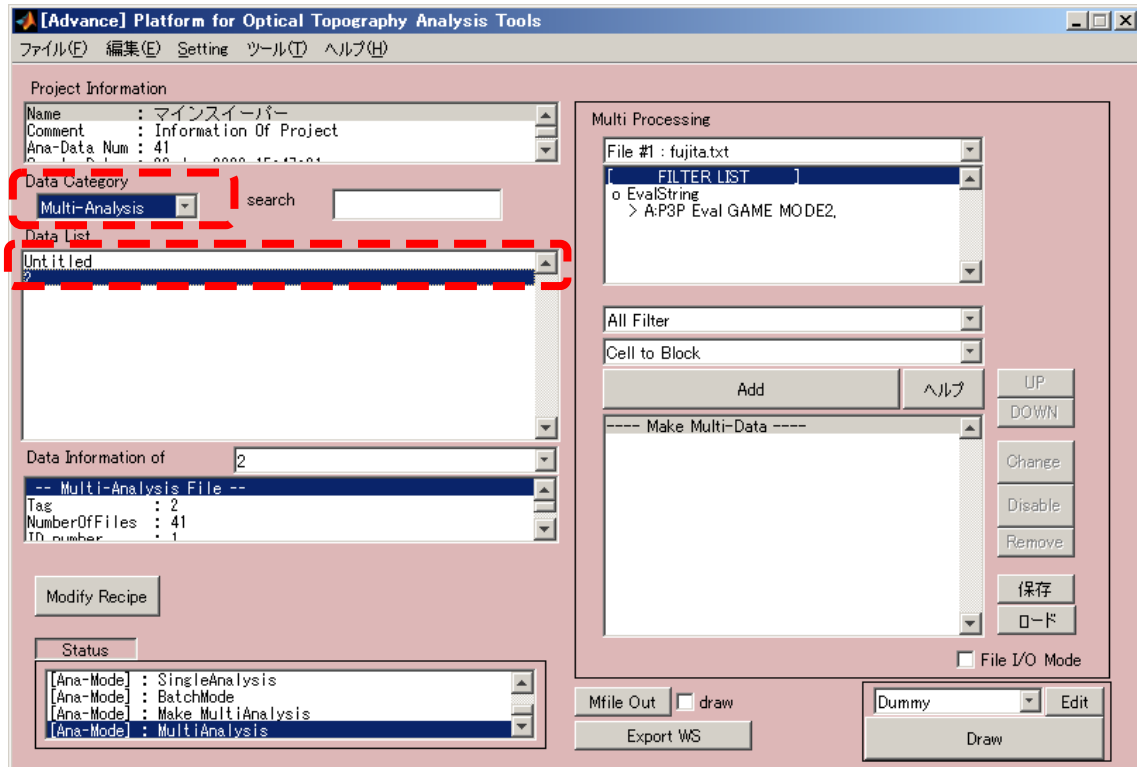


そうするとMulti-Analysisデータ設定画面になりますので、任意の名前を入力しOKを押します。



ここでは Untitled を入力します。

この時 Data Category から「Multi-Analysis」が選択できるようになりますので、作成した Data(Untitled)を選択します。

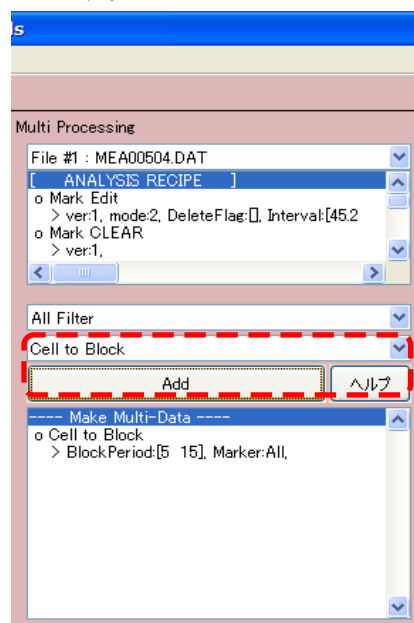


では、MultiAnalysis でt検定のための Recipe を作成していきます。

まず関数「Cell to Block」を追加します。

設定画面は「Blocking」と同様の画面が出ます。

このとき、それぞれのファイルで設定した Blocking と同じ値、もしくは、それらよりも Block 長さが短くなるように設定してください。



詳細説明:

関数「Cell to Block」を行う前は、内部データは各ファイルの「セル」として保存しています。

この関数によって、各ファイルの Block データは、ファイル間の区別がなくなり、一つの「配列」に変換されます。

留意点:

Block のサイズが被験者間で異なる場合、短い Block に長い Block が合わせられます。

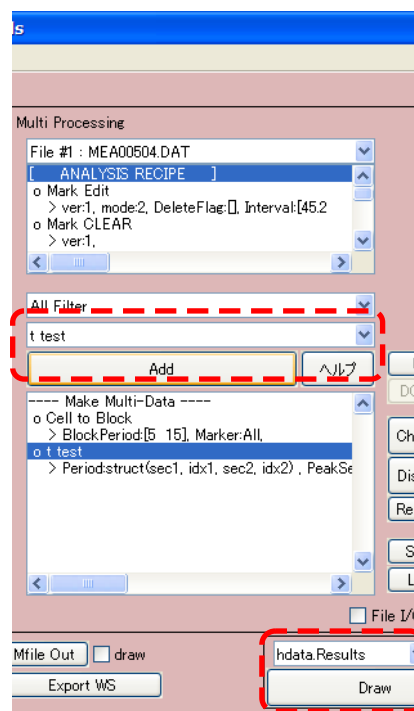
File I/O Mode ではメモリの使用量を減らすために、ディスク領域からデータを読み込んでいます。そのため、作業に時間がかかる場合があります。

次に、関数「t test」を追加します。

設定画面が出ますが、単一ファイルの場合での設定と同様です。設定方法の詳細はフィルタプラグインをご参照ください。

最後に、描画レイアウト「Map for hdata.Results」を選択し、「Draw」ボタンを押し、結果を表示させます。

詳細な値を取得する場合には、「Export WS」を押し、Matlab のコマンドラインから、「hdata.Results.ttest」を実行すると、変数の内容が表示されます



留意点:

ただし、この解析方法は十分に汎用的ではありません。  
利用は全体的な傾向の確認に留めるべきかもしれません。  
なぜなら、一般的に光トポグラフィ信号値を被験者間で等価に扱い、  
比較することは、ほとんどの場合、正確さを欠くと思われます。

この辺りの解析方法についてはまだまだ議論が必要だと思います。

## 1.2. 複数被験者の平均信号を取得したい

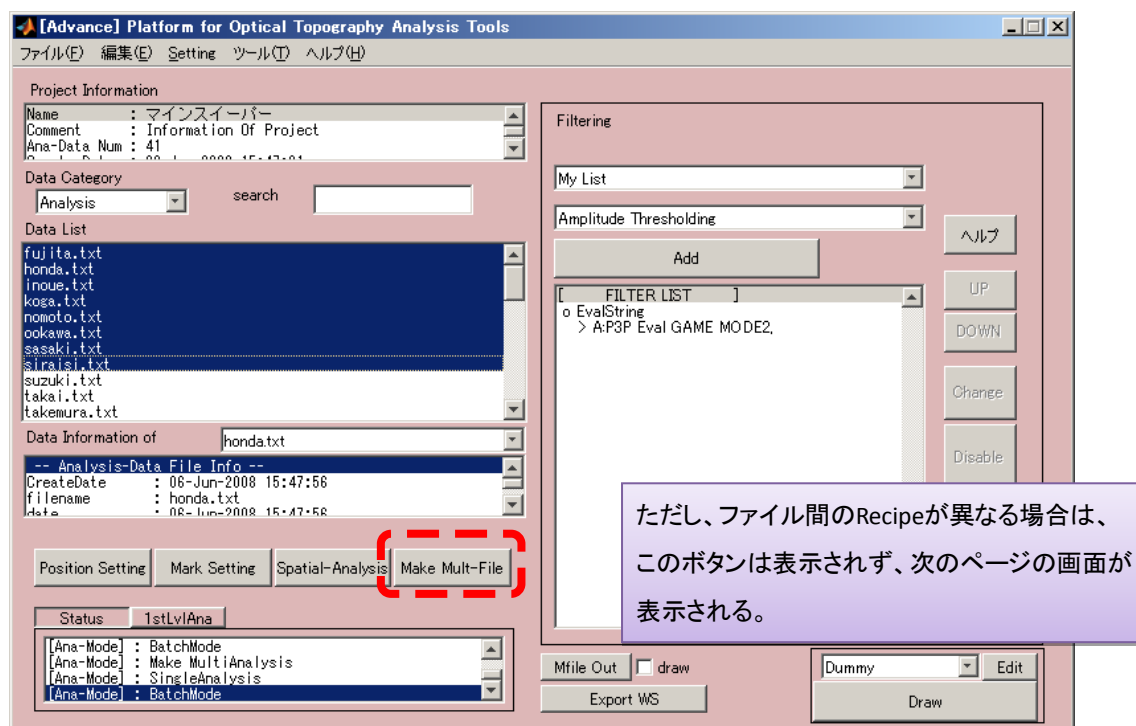
### 1.2.1. Answer

Multi Analysis で File I/O Move Average を実施してください。

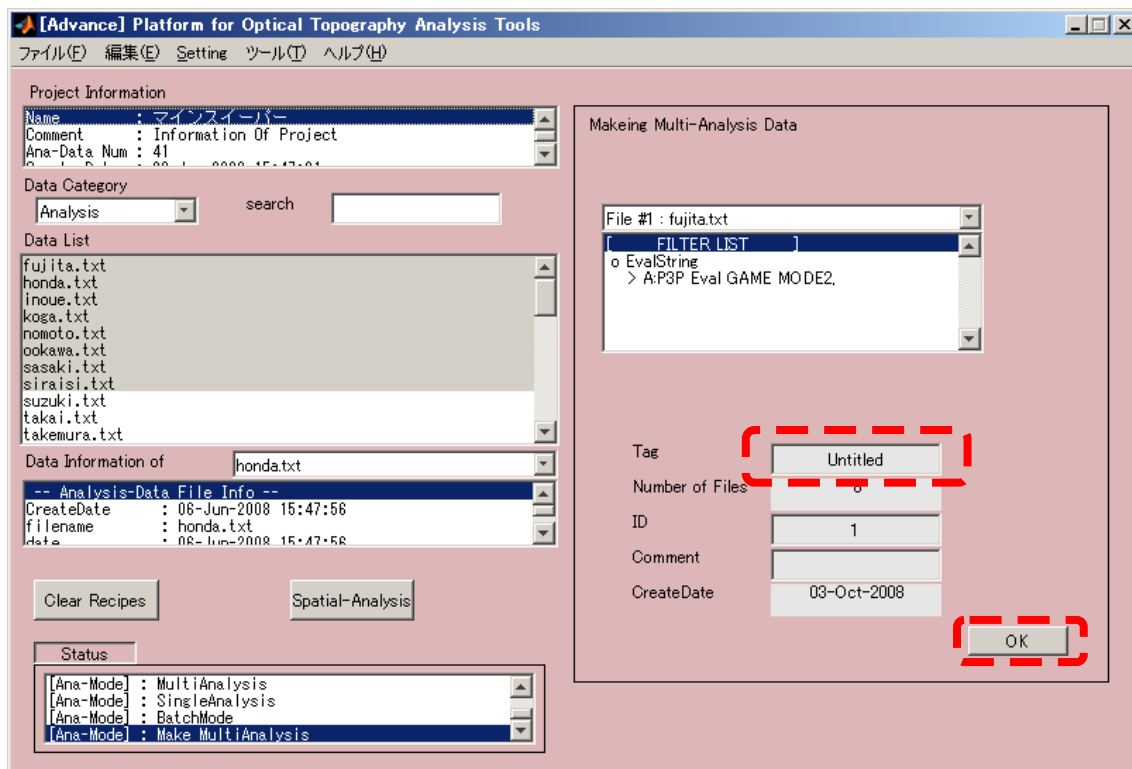
### 1.2.2. 実行例

解析は POTATo の Developer モードで行います。

目的の解析データを選択し、Make Multi-File ボタンを押し Multi-File を作成します。



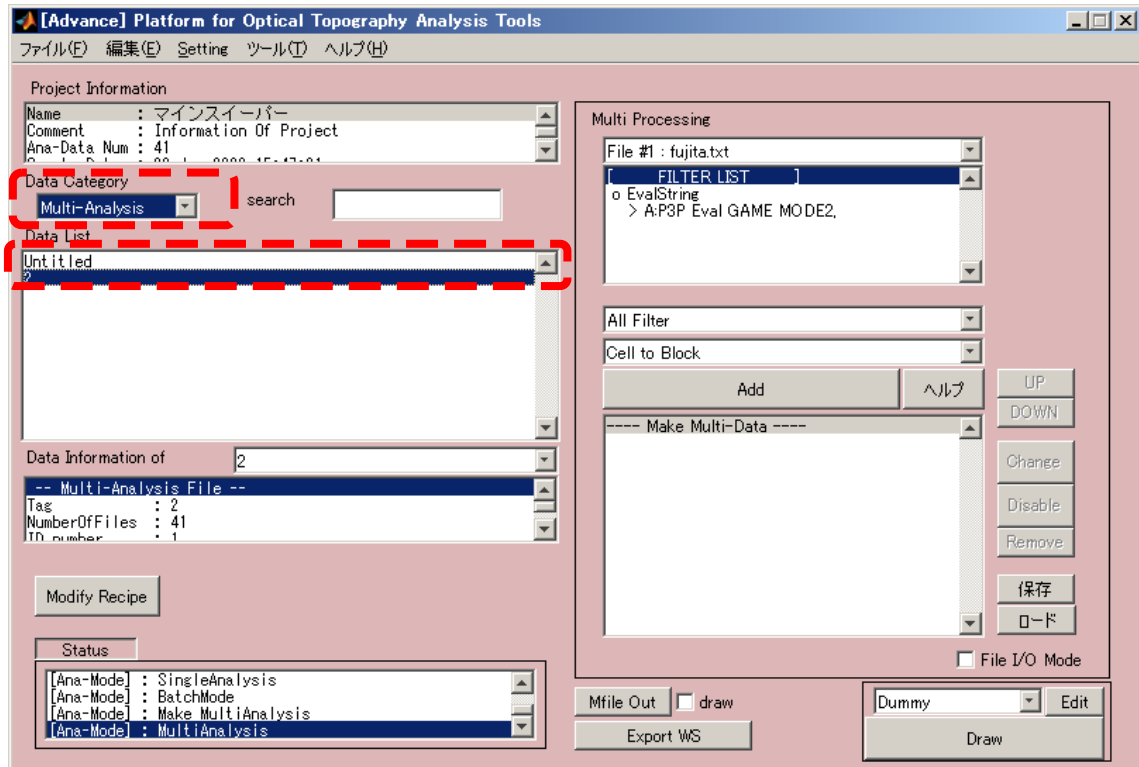
そうするとMulti-Analysisデータ設定画面になりますので、任意の名前を入力しOKを押します。



ここでは Untitled を入力します。

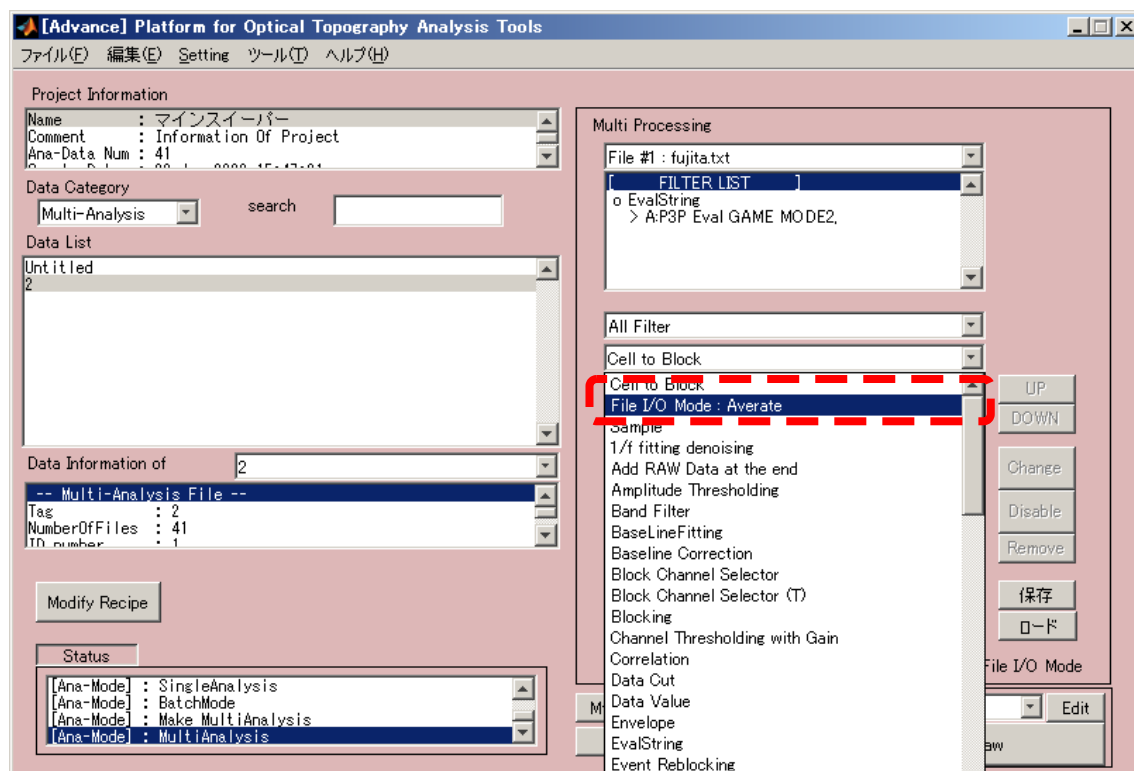


この時 Data Category から「Multi-Analysis」が選択できるようになりますので、作成した Data(Untitled)を選択します。

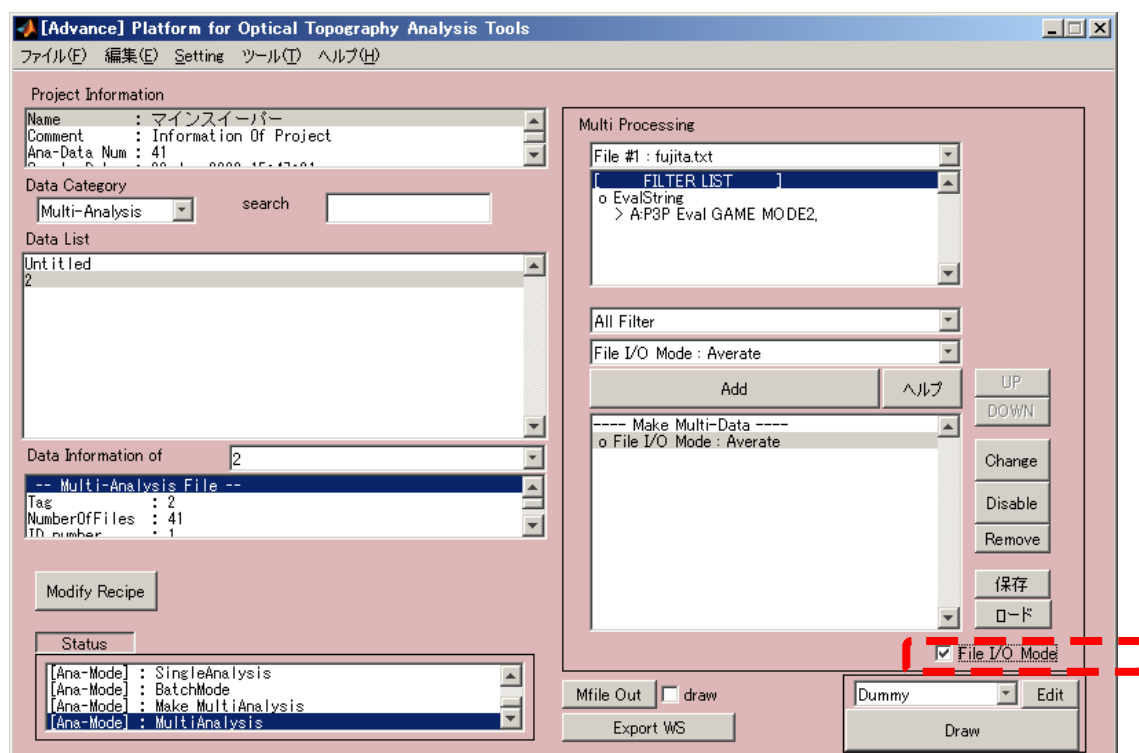


## 解析

Multi-Analysis データを作成し、モードを起動し、File I/O Move Average を選択し、Add します。



File I/O Mode にチェックを入れます。



最後に、ExportWS を押すと、ワークスペースに平均化データが出力されます。  
結果は data に記載されています。

留意点：

Block のサイズが被験者間で異なる場合、短い Block に長い Block が合わせられます。

File I/O Mode ではメモリの使用量を減らすために、ディスク領域からデータを読み込んでいます。そのため、作業に時間がかかる場合があります。

なお、“File I/O Mode: Standard deviation”の場合は、hdata.Results.FileIO\_SD に結果が入ります。

### 1.3. Group 解析 Correlation Analysis を行いたい

#### 1.3.1. Answer

Developer モードの 1st 解析で”[?] Duplicate Blocks for Correlation analysis”、” [?] save Re-sults for Correlation analysis”を行い、2nd 解析で“Correlation Analysis”を実行する。

#### 1.3.2. 実行例

解析は POTATo の Developer モードで行います。

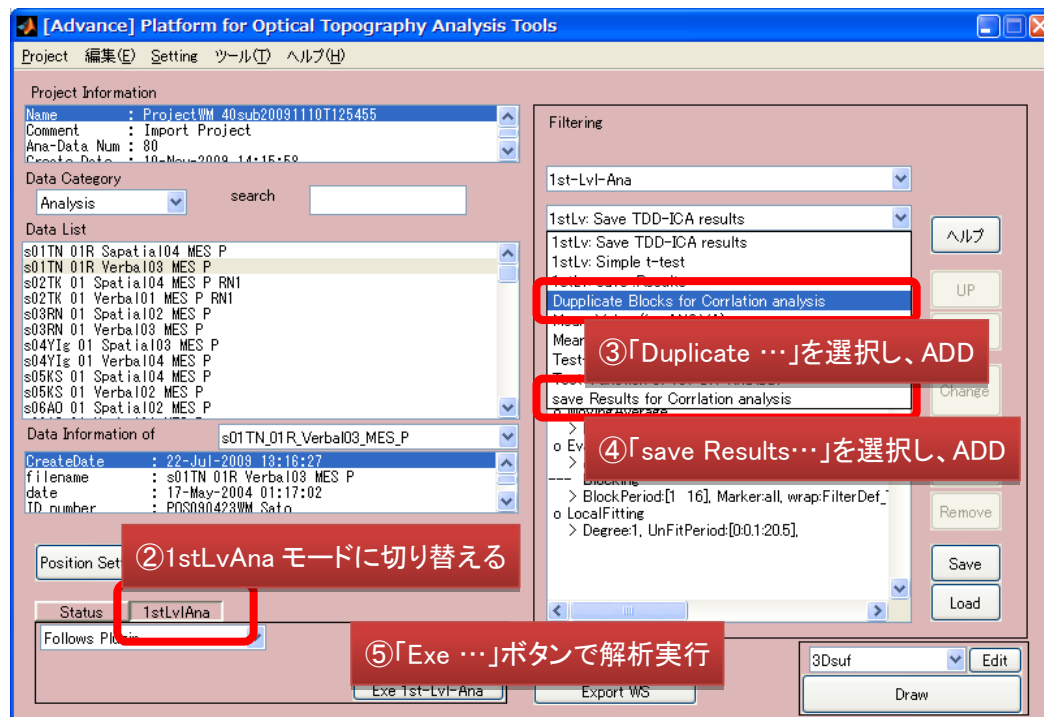
##### 1. 関連データの読み込み

ツール→Extended Search を起動し、検索キーとして関連データを読み込ませます。  
具体的な方法は、マニュアル「拡張検索機能」を参照してください。

##### 2. 代表データ作成 (1st level analysis)

①各データに対し前処理設定を行います。このとき、Block 化は必須です。

次に以下の通り、2つの 1st-Level-Analysis フィルタを追加し、実行してください。

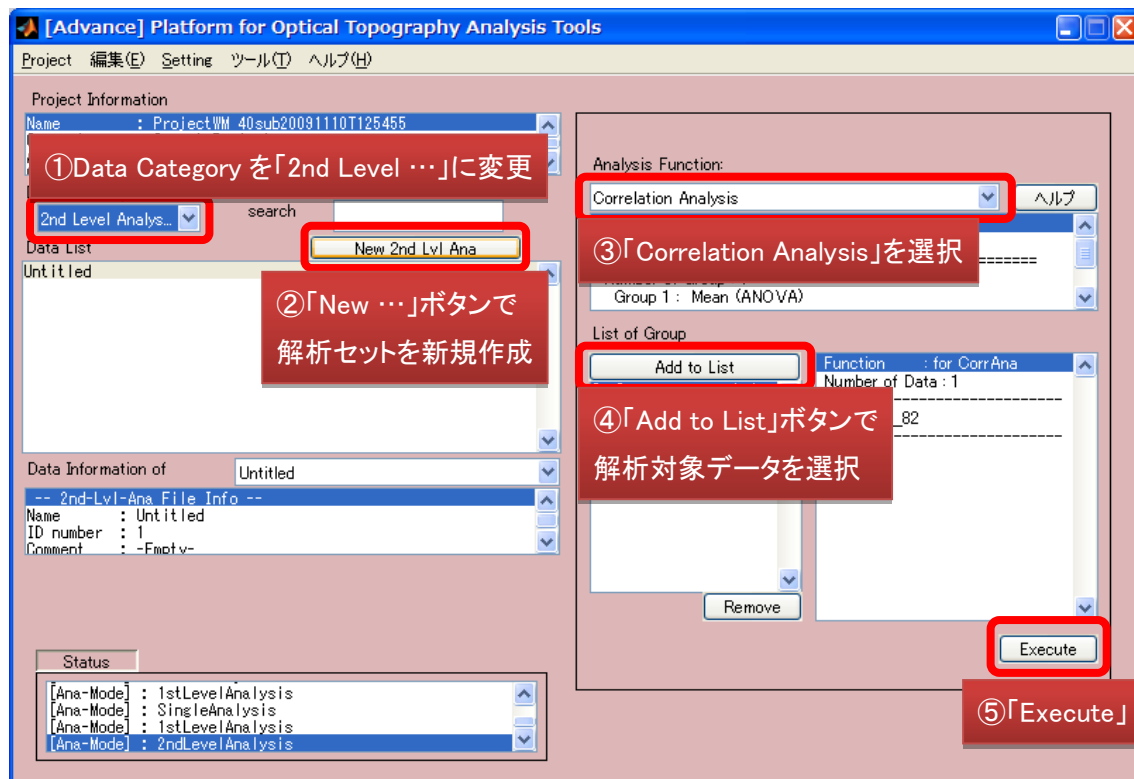


\* すべてのファイルを選択し同時に実行してください

Data Category を「1st Level ...」にして、データが作成されていることを念のため確認してください。

### 3. 2ndlvl の実行

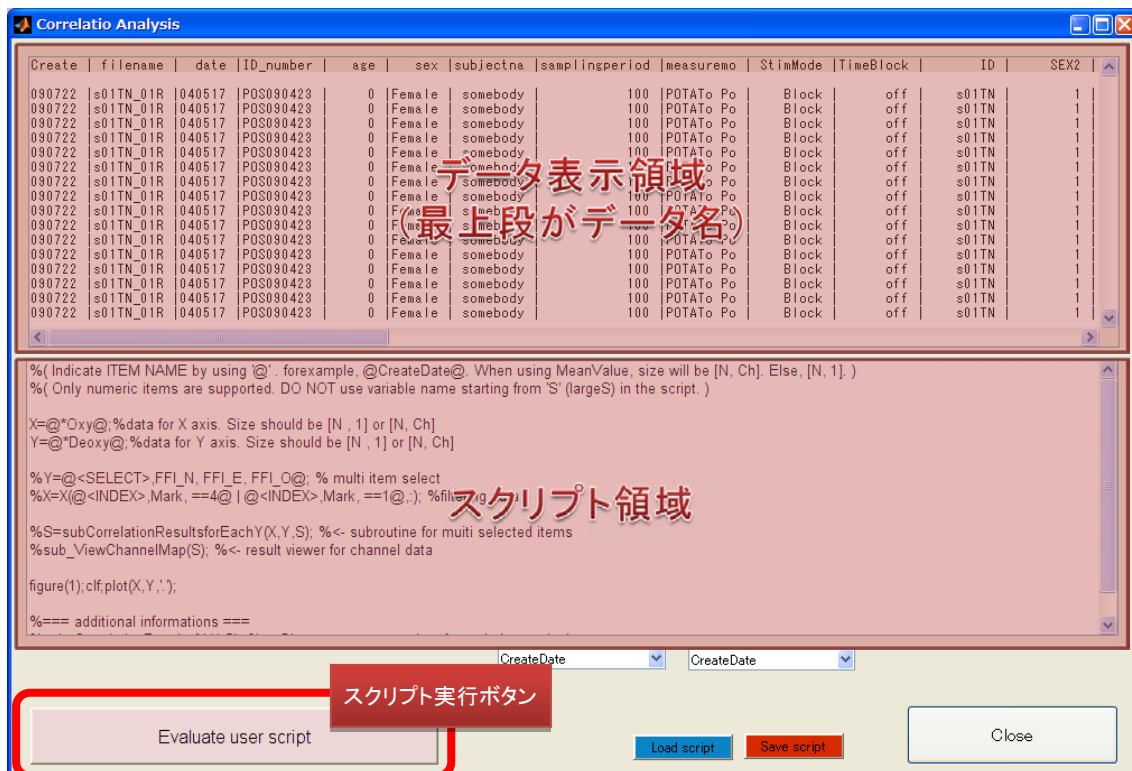
次に 2nd-Level 解析を行います。



このとき、Add to List で選択する解析対象データは"for CorrAna"から選びます。

## 4. Correlation Analysis

最後の実際に Correlation Analysis を実施します。



スクリプトの記述を説明します。

各行は Matlab 関数「eval」で実行されるので、Matlab 文法で記述してください。

以下に、「Correlation Analysis Tool」に特有の文法を説明します。

### **@選択 (@name@)**

画面上部の表からデータを選択する場合は、そのデータ名を「@」で囲み指定します。例えば、変数 X に「age」を代入する場合、

```
X=@age@;
```

となります。以降、この方式による選択を「@選択」と呼びます。

なお、内部処理では@で囲まれた部分をデータ選択関数に置き換えた後、eval で実行します。

例えば、先ほどの文字列は、

```
X=subFunction_Cell2Num(S.g(:,5));
```

のように変換されます。

### **複数選択 (@<SELECT>, name1, name2, name3@)**

通常、@選択により得られるデータのサイズは、[N, 1]である(ここで、N はデータ数、表の縦の数)。

<SELECT>オプションでは、複数のデータ列を選択し[N,M]とすることができます。

<SELECT>では2つ以上のデータ列を指定可能である。各データ列名は「,」で区切る必要があります。例えば、

```
X=@<SELECT>, age, samplingperiod@
```

とすると、X のサイズは、[N, 2]となる。ここで、@選択は配列を出力するので、数値と文字列などを同時に指定することはできません。

### **インデックス作成 (@<INDEX>, name, 条件式@)**

name で指定されたデータに対し、「条件式」を適用し INDEX を得ます。

```
@<INDEX>, age, >20@
```

では、データ「age」に対し、「age>20」を適用した結果を返します。これは例えば、

```
IDX=@<INDEX>, age, >20@;
```

```
X=@name1@;
```

```
X=X(IDX,:);
```

のように用います。インデックス同士は論理演算が可能です。例えば、

```
IDX=@<INDEX>, age, >20@ && @<INDEX>, age, <30@
```

とすることができます。

### サブルーチンについて

Correlation Analysis プラグイン内部に組み込まれたサブルーチンを利用することができます。以下にそれらの詳細を説明します。

#### subSummarize: subSummarize(x, condition, string)

(x: データ、condition: データ、string: eval 可能な文字列)

この関数では、データ「condition」のなかで重複しているもののインデックスを導出し、それに基づきデータ「x」を縮約することを目的とします。例えば、各「filename」について複数のブロックデータがあり、それを平均化したい場合、

```
x=@name1@;
CND=@filename@;
x1=subSummarize(x, CND, 'M=mean(M);');
```

とします。この関数の内部処理では、CND からユニークな値を抽出し、それぞれについてのループのなかで、その値をもつ複数のインデックスから選択した x を変数 M に代入します。そして、引数 string でしたいされた文字列を eval 関数で実行します。

この文字列では変数名 M に対して作業しなければなりません。

#### subSummarizeST: subSummarizeST(S)

(S: 構造体。

S.Target: 処理対象の変数を格納するフィールド。複数可。

S.Condition: 重複項目を選択するためのコンディションデータ

S.chloop: ループ数

S.String: eval 可能な文字列。セル配列可能。

この関数は subSummarize の機能を拡張したもので、縮約対象のデータを複数設定可能としたものです。引数は構造体として渡します。構造体のフィールドには上記のものが必須です。縮約した結果は M.Return フィールドに記録されます。

S.String では、S.Target の代わりに変数名 T を指定します。また、最終結果は変数名 X に代入しなければなりません。

```
M.Target.x1=@name1@;
M.Target.x2=@name2@;
M.Condition=@age@;
M.chloop=47;
M.String{1}=[h,p,c,stat]=ttest(T.x1,T.x2);
M.String[end+1]='X=stat.tstat;';
M=subSummarizeST(M);
```



## 解析

```
x=M.Return;
```

### **sub\_ViewChannelMap: sub\_ViewChannelMap(S)**

解析結果を Map として表示します。レイアウトは LAYOUT「resultLayout\_CorrAna.mat」を用いています。エディタで編集可能です。このサブルーチンでは、引数 S から、S.tmp\_hdata に基づき結果を表示します。1\*channel のデータがあれば、

```
S.tmp_hdata.Results.test = data;
```

```
sub_ViewChannelMap(S);
```

とすると結果が表示されます。

## 2. 表示機能

### 2.1. 描画結果の Line プロパティを変更したい

#### 2.1.1. Answer

描画図の Axes-Control メニューから、変更したい Line のある Axes を選び、Axes-Editor を起動します。

その後、Axes-Control メニューの Property から Line、All から他の Axes に結果を反映します。

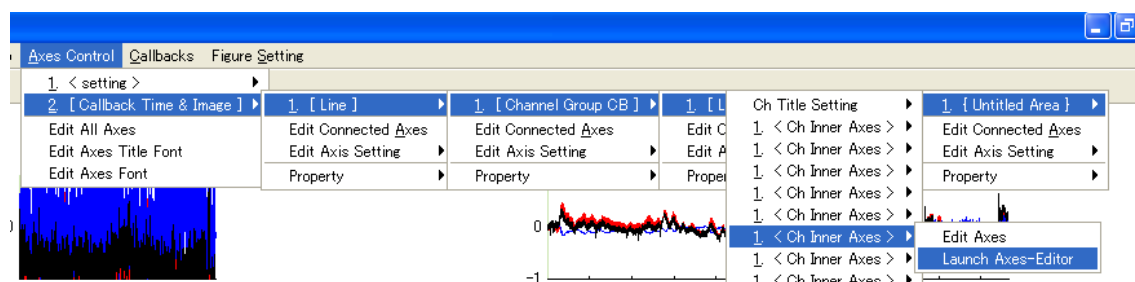
#### 2.1.2. 関連質問

今回は1回のみの設定です。

毎回、描画方法を変更したい場合は、描画用の設定ファイルである「Layout」ファイルを変更することで可能です。

#### 2.1.3. 実行例

描画後のフィギュアの Axes-Control メニューからの、“Launch Axes-Editor”を選択します。



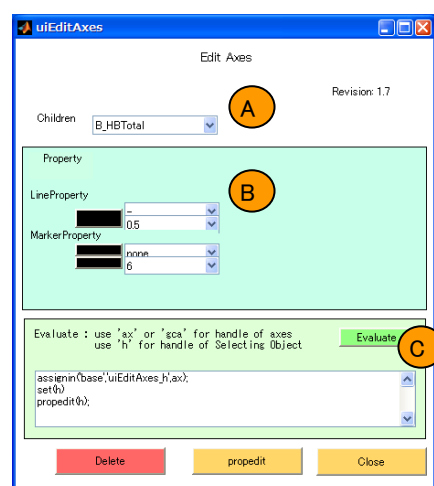
Axes の位置はレイアウトにより異なります。

すると、右図のようなウィンドウが立ち上がります。

変更したいデータ種を“Children”ポップアップメニュー (A) から選択し、Property (B) でラインやマーカーの種類・色を変更します。

最後に Close (C) ボタンで終了します。

これで、1つだけのプロットの表示が変更されます。



次に、この変更を図中の全てのプロットに反映させます。

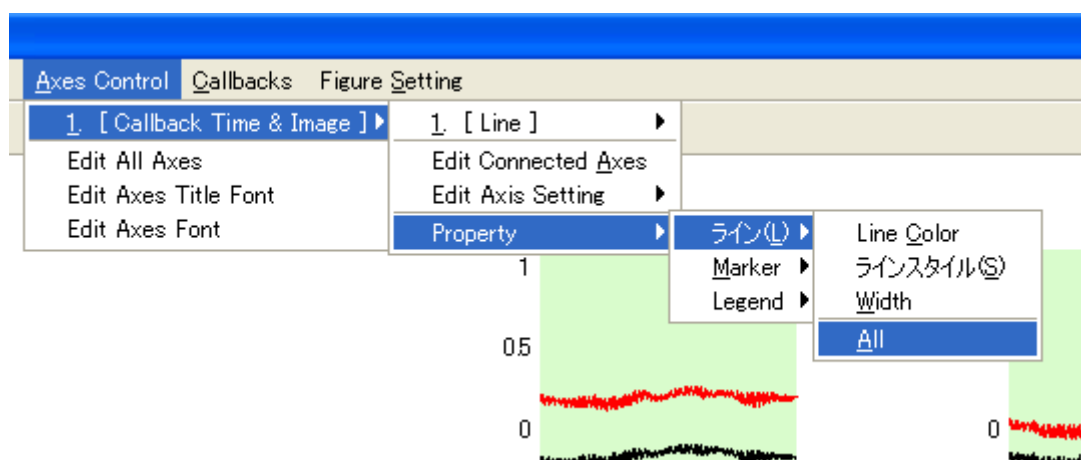
注意:

先ほどの変更後には、他のプロットをクリックしてはいけません。  
 なぜなら、Matlab 内部での“フォーカス”が変更したプロットにあります。  
 このフォーカスをいかしたまま、次の作業を行います。

図のメニューから今度は、

Axes Control -> 1.[ ] -> Property -> ライン -> All

を選択します。すると、すべてのプロットにフォーカスのあったプロットの“ライン”に関するプロパティがコピーされます。



同様に、マーカーに関しても、

Axes Control -> 1.[ ] -> Property -> Marker-> All

とすると、反映されます。

## 2.2. 描画する Line プロパティを変更したい

### 2.2.1. Answer

Layout Editor の Area 内の Primary 設定で、Line プロパティを設定してください。

### 2.2.2. 関連質問

描画済みの図を変更するにはメニューから対応可能です。

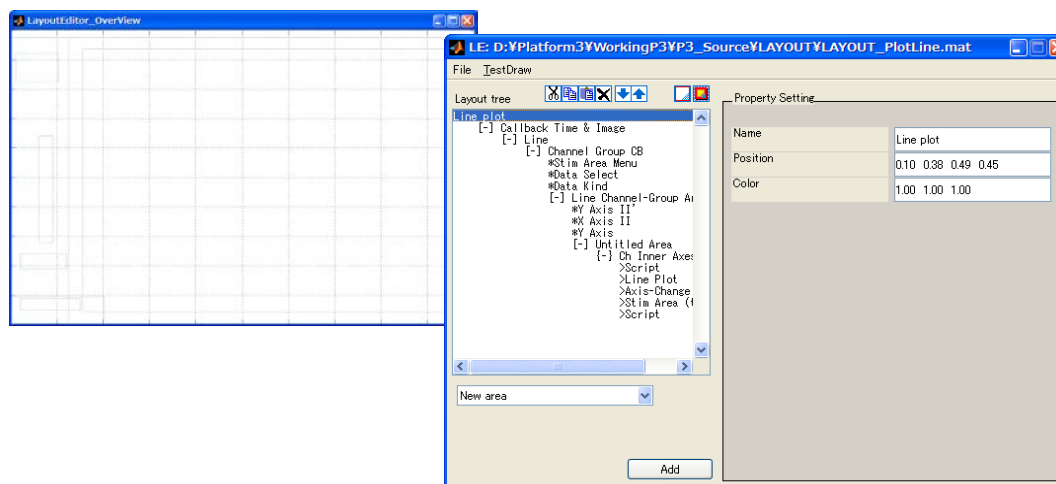
### 2.2.3. 実行例

レイアウトを変更することにより、描画の Line プロパティを常に変更する方法を説明します。

Layout Editor という機能の、Line プロパティの変更に関する部分に特化して説明します。Layout Editor の詳細はマニュアル『表示機能』の Layout Editor の使い方をご参照ください。



編集したレイアウトをポップアップメニュー(A)から選択し、Edit ボタン(B)を押し、Layout Editor を起動します。すると、以下のような2つのウィンドウが表示されます。



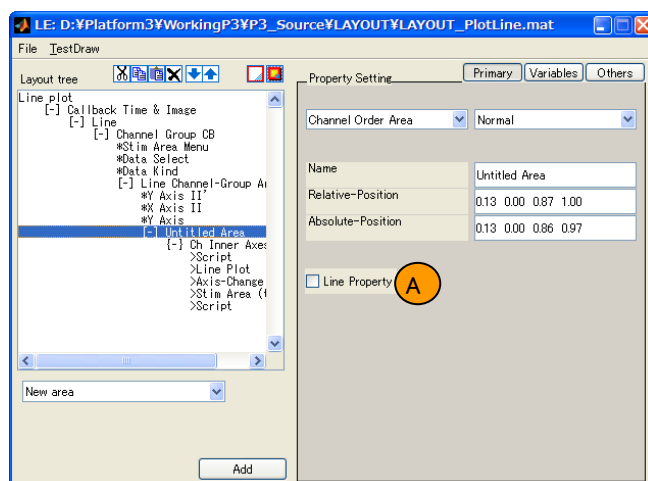
表示される、“Layout tree”リストから、変更したい Line を描画する AO を探します。そして、この“Line Plot”が含まれる“Area”([-])が表示されます)を選択します。

この例では、図のように“Line Plot”AO の上位 Area、“Untitled Area”を選択します。

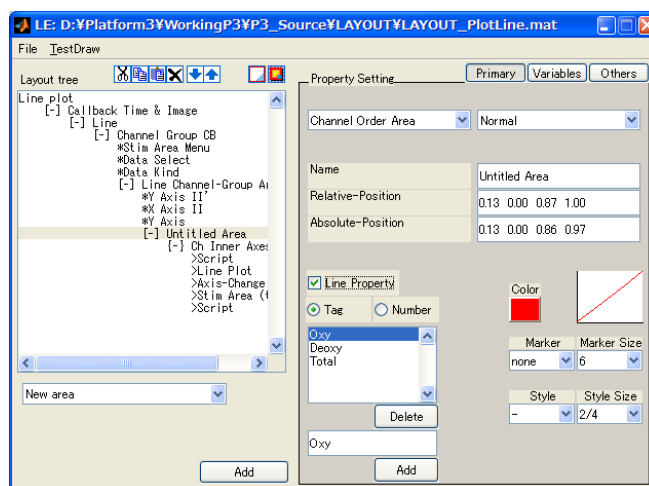
```
#Data Select
#Data Kind
[-] Line Channel-Group A
  #Y Axis II
  #X Axis II
  #Y Axis
  [-] Untitled Area
    [-] Ch Inner Axes
      >Script
      >Line Plot
      >Axis-Change
```

Area を選択すると、右画面が右図のようになります。

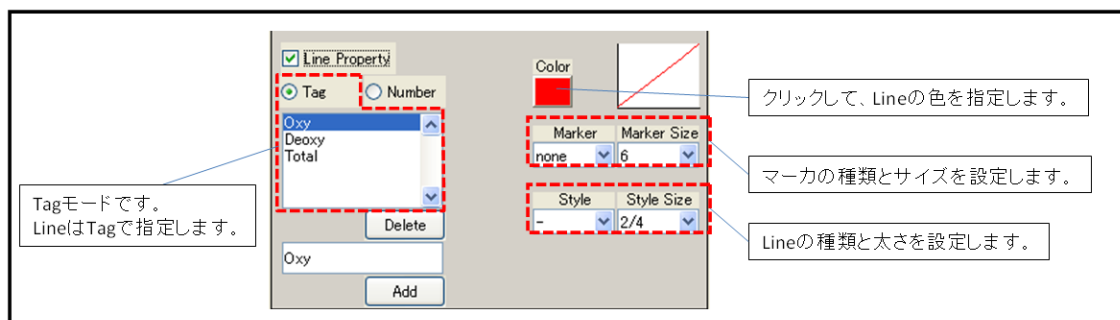
この画面で、“Line Property”チェックボックス(A)をチェックします。



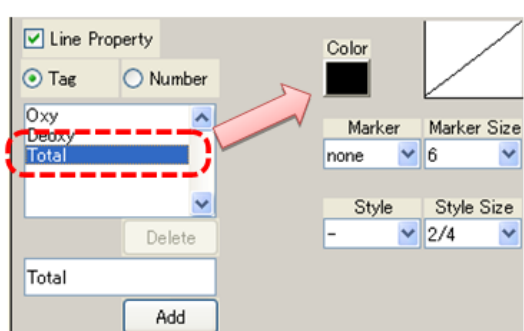
すると、右図のような表示になります。  
この設定画面から Line プロパティを変更します。



Line プロパティは描画する Line の Tag で指定します。設定方法は以下になります。



例として Total を紫に変更します。



Totalを選択、Colorをクリック。



紫を選択し、OKボタン。

これで、設置完了です。

あとは、このレイアウトファイルを上書き保存します。上書き保存は File メニューの save から実行可能です。

レイアウトエディタを閉じ、POTATo のメイン画面に戻って描画してみてください。

次は、少し上級者向けの説明です。

自作のプラグインで Data Kind を追加した場合に表示プロパティを設定する方法です。Tag モードでの設定方法を説明します。

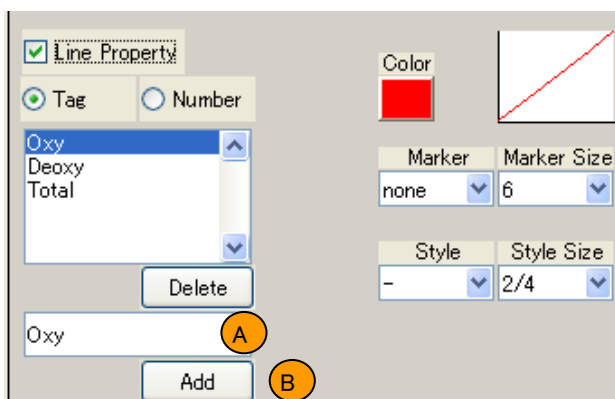
Data Kind に追加する DataTag を左下のテキストボックス(A)に入力し、Add(B)ボタンを押します

例として、"test\_kind"を追加しました。

test\_kind にはデフォルトで Line プロパティが設定されます。

色や、マーカーをお好みに設定し直し、

最後にレイアウトファイルを上書き保存します。



このレイアウトファイルで描画するときに、もしデータに"test\_kind"があればこの設定で描画されます。ない場合には、無視されます。

ここで、Data-Kind に関する補足です。

Data-Kind は POTATo データの種類です。

連続データでは data(time, channel, kind) の第3次元目のデータになります。

データのタグはヘッダ部(hdata)の TAGs.DataTag フィールドにセル配列で記載されています。

詳細はマニュアル『基本操作』の POTATo のデータ形式をご参照ください。

\* data(:, :, 4)に test\_kind に関するデータを入れた、TAGs.DataTag[4]='test\_kind' と設定した場合に結果が反映されます。

### 3. データ管理

#### 3.1. 解析結果を CSV ファイルに出力したい

##### 3.1.1. Answer

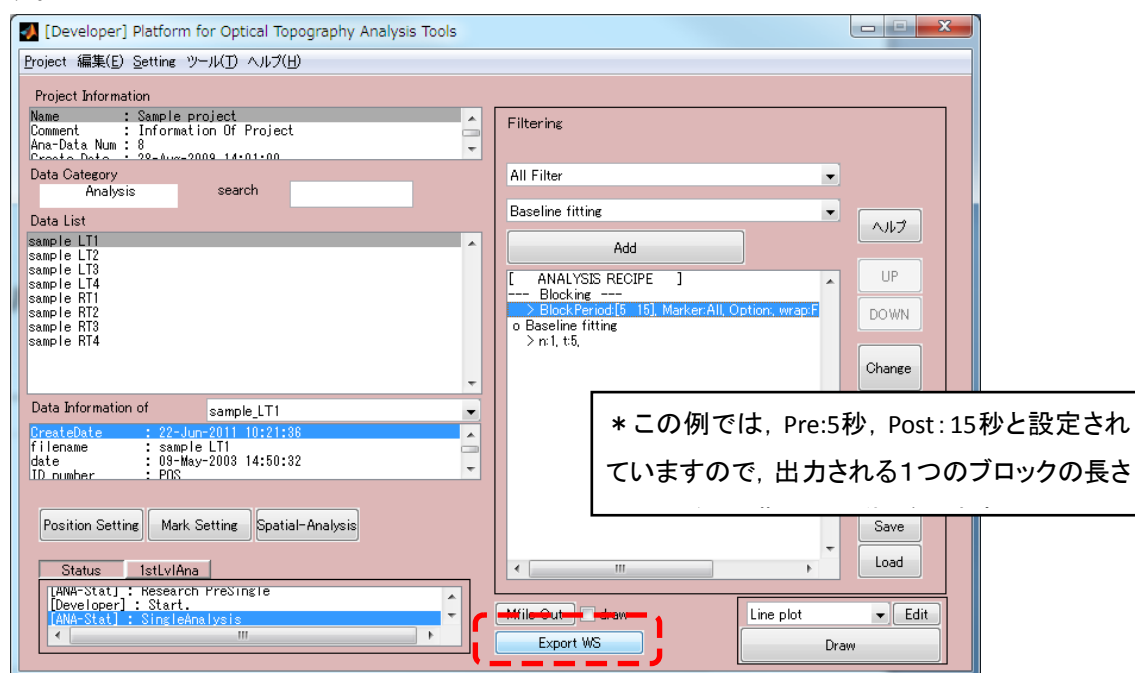
Export WS ボタンでデータをワークスペースに出力し、CVSWRITE 関数を利用してファイルに出力してください。

##### 3.1.2. 関連項目

『MATLAB の基本操作』にデータの操作、ファイル I/O を説明しています。

##### 3.1.3. 実行例

Developer モードもしくは Research モードの Preprocess で、解析 Recipe 中で Block データの作成パラメータを適切に設定し、「Export WS」ボタンを押し、Matlab ワークスペースへデータを出力します。



その結果、ワークスペースに POTATo 区間データとして data, hdata が出力されます。

注意:

データ名は Export WS ボタンを実行したモードやサブ状態により異なります。



Matlab コマンドウィンドウから、以下のようにコマンドを実行します。

```
csvwrite('temp.csv', permute(data(:, :, 1, 1), [2 1 3 4]));;
```

コマンド: csvwrite は変数をファイルに保存する関数です。例では, temp.csv というファイルに, data(:, :, 1, 1)を出力します。

CSV ファイルで縦の列を時間, 横の行をブロック番号とするために, コマンド: permute を用い, 出力するデータ配列の次元を修正しています。

なお、POTATo 区間データは 4 次元配列となっており, 各次元はそれぞれブロック番号, 時間, チャンネル数, データ種 (Oxy, Deoxy, Total) となっています。例では, チャンネル 1 番の Oxy 信号を出力します。

出力したいデータに併せて出力データを調整してください。

例えばチャンネル 2 番を出力する場合には data(:, :, 2, 1)と設定します。

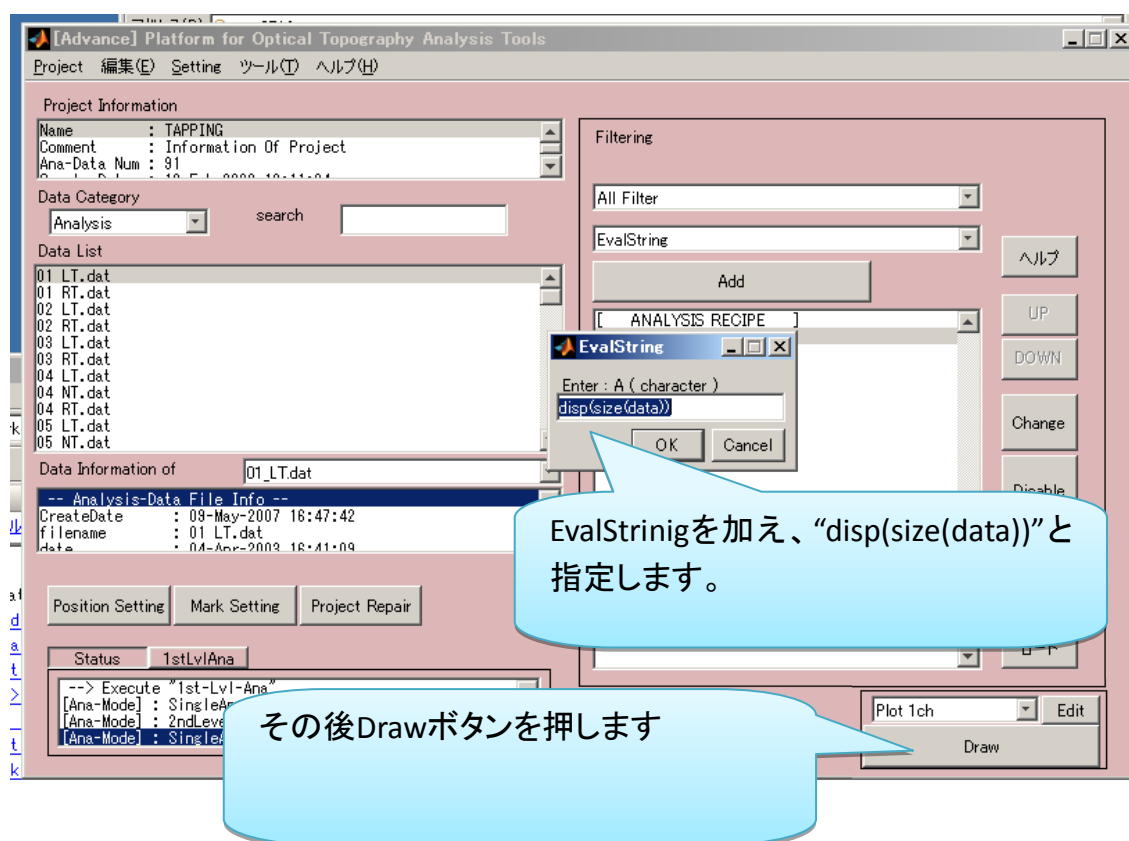
## 3.2. データを追加したい

### 3.2.1. Answer

POTATo データのヘッダ部の TAGs.DataTag の最後にデータ種類名を記載し、データ部の3次元目にデータを追加してください。

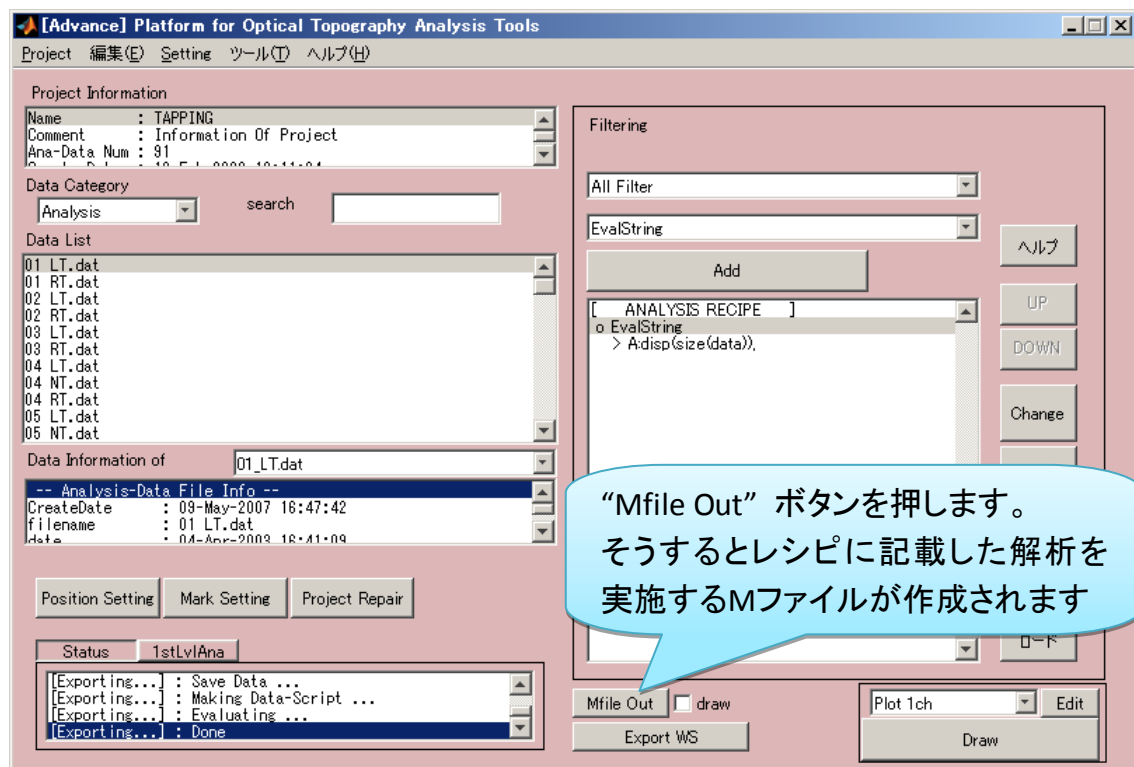
### 3.2.2. 実行例

EvalString フィルタを使った作成例を示します。



結果、コマンドウィンドウ上に data のサイズが表示されます。

ここで、解析内容を見ます。



作成された M ファイルは以下ようになります。

```

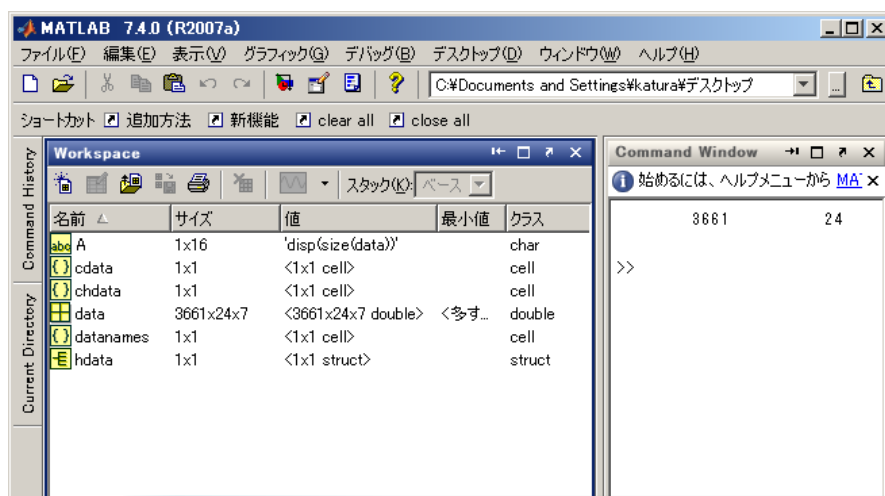
1  % ANA_20090129T140943, make POTATo data.
2  %
3  % == Analysis-Data-Information ==
4  % Project : D:\Platform3\WP3Project
5  %       TAPPING
6  % Operator: P3 User
7  % Name   : 01_LT.dat
8  %
9  % Date : 29-Jan-2009 14:09:43
10 %
11 % == Recipe Information ==
12 % * EvalString
13
14 % -----
15 % Platform for Optical Topography Analysis Tools III
16 % Function : GroupData2Mfile : $Revision: 1.24 $
17 % -----
18
19 %*****
20 % Continuous Data Setting
21 %*****
22 datanames = { ...
23     'D:\Platform3\WP3Project\TAPPING\RAW_01_LT.dat.mat'};
24
25 %*****
26 % Continuous Data Operation
27 %*****
28 % Load Signal-Data, See also UC_DATALOAD
29 [data, hdata] = uc_dataLoad(datanames{1});
30
31 %=====
32 % == EvalString ==
33 % EvalString v1.0
34 % =====
35
36 try
37     A = 'disp(size(data))';
38     [data, hdata] = uc_EvalString(data, hdata, A);
39 catch
40     error('lasterr');
41 end
42
43 % =====
44
45
46
47 cdata = {data}; chdata = {hdata};
48
49

```

**データ読込**  
POTATo データ  
hdata, data が読み込まれます。

ここは“EvalString”により作成されたスクリプトです。  
関数“uc\_EvalString”内では  
文字列“A”が MATLAB の関数 eval を  
用いて実行されます。

このスクリプトを実行した際のワークスペースを見ます。



POTATOデータ“data”と“hdata”に注目します。

“data” は時刻、チャンネル、データの種類の3次元配列です。

“hdata”は“data”に関する情報を提供するヘッダです。

このフィールドにある“hdata.TAGs.DataTag”には各データの種類の名前が記載されています。

この EvalString でデータの種別を追加します。

データの種別を追加するスクリプト M ファイルを“testAddData.m”という名前で作ります。

関数例は以下の通りです。

```
s=size(data);  
add1=ones(s([1 2])); % 実際には追加したいデータを記載  
add2=ones(s([1 2])); % 実際には追加したいデータを記載  
data(:, :, end+1)=add1;  
data(:, :, end+1)=add2;  
hdata.TAGs.DataTag{end+1}= 'new_kind1' ;  
hdata.TAGs.DataTag{end+1}= 'new_kind2' ;
```

最後に EvalString の設定を変更します。

