バッチを用いた一括処理(2)

筑波大学医学医療系精神医学
根本清貴

Ver.20191110.01

この時間の作業ディレクトリ

- 先程に続いて、`nisg-201912/ex3` がこの時間のディレクトリです

  >> **pwd** %nisg-201912/ex3 のはず

SPMのバッチ機能

- SPMには、Batch Editorが準備されており、これを用いると、GUIでバッチ（一括処理）の下準備ができる
- バッチは、以下の流れで準備する
  - １人のデータに対して、一連の処理を行う
  - 複数人のデータでもできるようにする
- このセクションでは、複数人のデータでもできるようにする
- バッチに多用するコマンド群を知る

VBM Clinical（多数例対応版）

- 1人のデータを指定すると、健常者と比べて低下している領域を表示することを、一度に複数人分処理できるようなバッチを作成したい
- 自分がしたい一連の流れを洗い出す
  - ワーキングディレクトリを指定
  - 統計結果を保存するディレクトリを準備
  - Segmentation
  - Smoothing
  - 統計モデルの作成
  - コントラストの作成
  - 結果表示
  - Rendering（脳表上に投影）

バッチ作成の流れ

- 1例の処理を作成し、バッチ用に mファイルで保存（コアバッチ；中核となるバッチ）
- テンプレートを開き、別名で保存（テンプレート）
- テンプレートのバッチ挿入部分にコアバッチを挿入
- 2例でうまく動作するか試行

テンプレート

- 自分が（実際に）使っているテンプレート
- バッチに必要な部分を前もって準備してある

```
>> edit spm_batch_template.m
```

- 今後、バッチに記載してある内容は緑色で表示
- %: コメント
  - %以降は、コメントとして扱われる
  - %% と%が2個連続すると、「セクション」として扱われる

テンプレート:スクリプトの説明

```
%% batchname.m
% Here comes explanation of script
%
% Author Name Day/Month/Year
```

- 冒頭にスクリプトの説明をいれる
- うまく作れた際に、最後にこの部分を書いている

テンプレート：SPMの起動

```
%% Run SPM

spm('pet')

%spm('fmri')


%% Prepare the SPM window

%spm('CreateMenuWin','on'); %top-left window (Menu)

%spm('CreateIntWin','on');  %bottom-left window
(Interactive)

%spm_figure('Create','Graphics','Graphics','on'); %right
window (Graphics)
```

- SPMを起動
- 下の方は、SPMのウィンドウをひとつずつ起動したい場合のコマンド

テンプレート:バッチの初期化

```
%% Initialize batch
spm_jobman('initcfg');
matlabbatch = {};
```

- バッチの初期化(必須)
  - SPMのバッチを初期化
  - matlabbatchの構造体を初期化
- お約束事として書くものと考える

テンプレート:ファイルの選択

```
%% Select Image files

filelist =
spm_select(Inf,'image','Message...'
,{},pwd,'.*',1);
```

- SPMのファイル/ディレクトリ選択GUIを表示
- 選んだ画像を、変数 **filelist** に格納

テンプレート:ディレクトリの選択

```
%% Select directory

dir =
spm_select(1,'dir','Message...');
```

- SPMのファイル/ディレクトリ選択GUIを表示
- 選んだディレクトリを、変数 **dir** に格納

テンプレート:forループ

```
%% for loop
for i = 1:size(filelist,1)
   %% Batch
end
```

- バッチの内容を、forループの中に入れる
- forループの中身はインデントをかけておくとわかりやすい

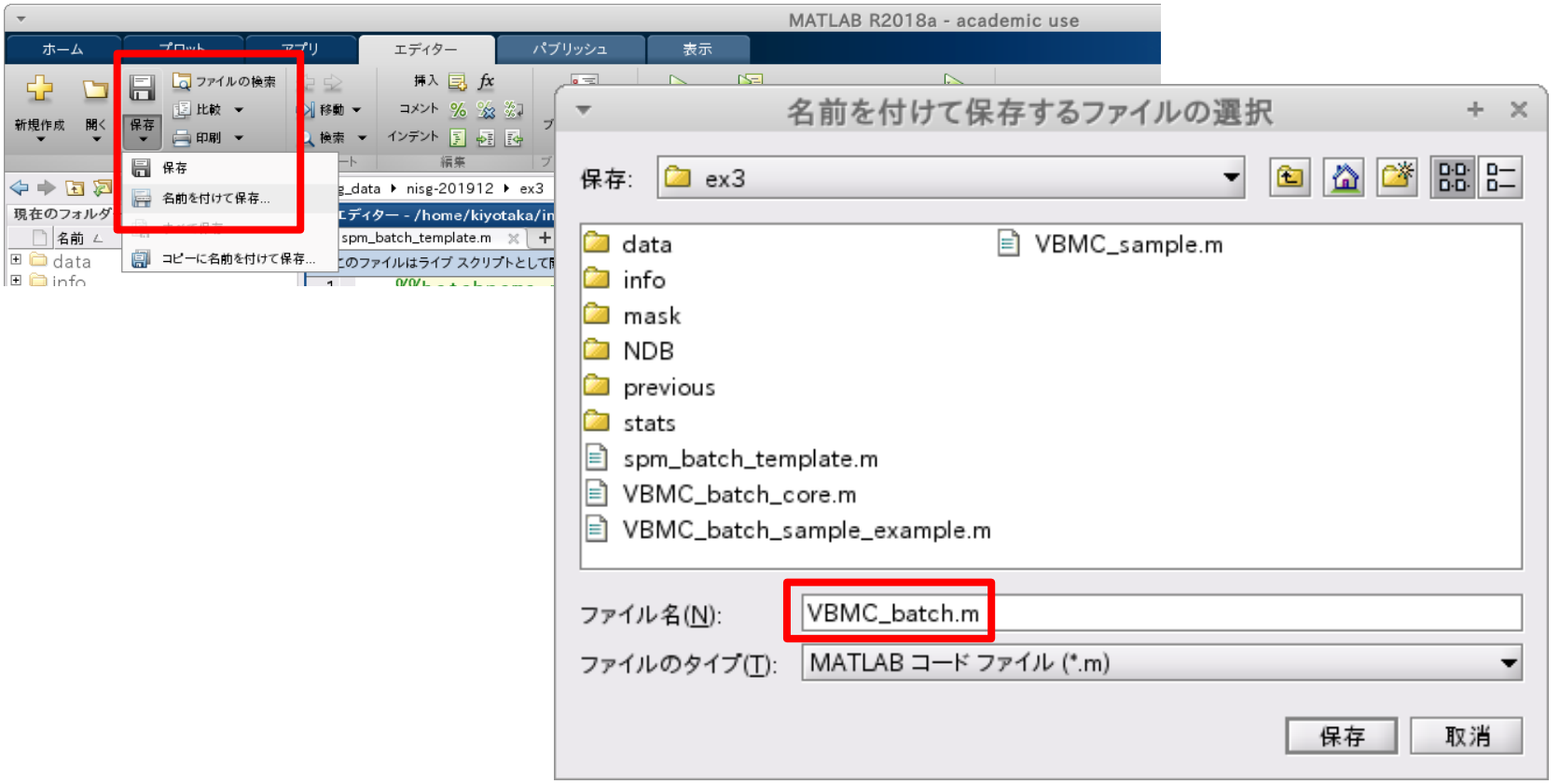
テンプレート:バッチの実行

```
%% Run batch
spm_jobman('interactive',matlabbatch);
%spm_jobman('run',matlabbatch);
```

- バッチを実行するために必要な部分
- これも必須

バッチの原型を作成

- 今開いているものを、Matlabのメニューから、「エディター」→「保存」→「名前をつけて保存...」とし、**VBMC_batch.m** として保存



コアバッチ

- 先程作成した **VBMC.m** から、最初のディレクトリ作成の部分だけ変更したものを、**VBMC_batch_core.m** として準備
- Matlabのエディタからこの中身を確認

```
>> edit VBMC_batch_core.m
```

SPMのバッチ構造

- **matlabbatch**という構造体を使用している
  - **matlabbatch{1}** → 1人目のワーキングディレクトリの設定
  - **matlabbatch{2}** → segmentation
  - **matlabbatch{3}** → smoothing
  - **matlabbatch{4}** → 統計モデル作成
  - **matlabbatch{5}** → パラメータ推定
  - **matlabbatch{6}** → コントラスト設定
  - **matlabbatch{7}** → 結果表示
  - **matlabbatch{8}** → 脳表への表示
  - **matlabbatch{9}** → PNG画像への保存

一括処理の発想

- forループを使って、matlabbatchを複製していく

  matlabbatch{1} 1人目ワーキングディレクトリ

  …

  matlabbatch{9} 1人目PNG作成

  matlabbatch{10} 2人目ワーキングディレクトリ

  …

  matlabbatch{18} 2人目PNG作成

  matlabbatch{19} 3人目ワーキングディレクトリ

これから行うこと

- **VBMC_batch.m**（テンプレート）の **matlabbatch** 部分に、**VBMC_batch_core.m**（コアバッチ）を挿入する
- forループでまわせるように、**matlabbatch** の番号を工夫する
- 汎用化するために、SPMのパスなどを修正する

作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

バッチに必要なコマンド

- SPMのコマンド
  - **spm_select**　　ファイル/ディレクトリの選択
  - **spm_fileparts**　パスを分割
- Matlabのコマンド
  - **mkdir**　　ディレクトリの作成
  - **pwd**　　現在のディレクトリ
  - **fullfile**　　ファイル名、ディレクトリからパスを作成
  - **deblank**　　行末の空白を削除

テンプレートへの処理バッチの挿入

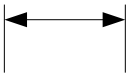
- **VBMC_batch_core.m** をすべて選択し、コピー
- **VBMC_batch.m** の35行目付近、**%Insert matlabbatch here** と **end** の間に、貼り付け

```
30
31      %% for loop
32 -    for i = 1:size(filelist,1)
33
34          %% Batch
35          % Insert matlabbatch here
36              ← ここに貼り付け
37 -    end
38
```

挿入した部分の整形

- forループの中は、インデントをつけておくと見やすい
- matlabbatch全体を選択し、Tabキーを押すと、インデントがつく

```
30
31      %% for loop
32 -    for i = 1:size(filelist,1)
33
34          %% Batch
35          % Insert matlabbatch here
36 -        matlabbatch{1}.cfg basicio.file dir.dir ops.cfg cd.dir = {'/hor
37 -        matlabbatch{2}.spm.spatial.preproc.channel.vols = {'/home/foo/
38 -        matlabbatch{2}.spm.spatial.preproc.channel.biasreg = 0.001;
39 -        matlabbatch{2}.spm.spatial.preproc.channel.biasfwhm = 60;
40 -        matlabbatch{2}.spm.spatial.preproc.channel.write = [0 0];
```



これがインデント

作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

SPMの起動（確認のみ）

- 今後、**VBMC_batch.m** のみで作業
  - 他のmファイルは閉じた方が無難
- 8行目

```
spm('pet')
```

- これでSPM (PET&VBM) が起動する
  - fMRIモードなら、**spm('pet')**

作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

matlabbatchの初期化（確認のみ）

- 18行目～20行目
- ここで、matlabbatchの初期化がなされる

```
%% Initialize batch
spm_jobman('initcfg');
matlabbatch = {};
```

作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

被験者データの選択

- **spm_select** を用いて、被験者データを選択
- 24行目を以下のように変更（すべて1行）

```
subj = spm_select(Inf,'image','Select
subject image(s)…',{},pwd,'.*',1);
```

```
filelist → subj
```

```
'Messages…' → 'Select subject image(s)…'
```

- **Select subject image(s)**は自分がわかればどのような文字列でもよい（任意）

ノーマルデータベースの選択

- 先程と同様に、spm_select を用いて、ノーマルデータベースとなる健常者データを選択
- 先程作成した24行目を25行目に複製し、以下のように変更（すべて1行）

```
ndb = spm_select(Inf,'image','Select NDB…',
{},pwd,'.*',1);
```

```
subj → ndb
```

```
'Select subject image(s)…' → 'Select NDB…'
```

- Select NDBも任意

spm_select

- ファイル/ディレクトリの選択

  spm_select(n,typ,mesg,sel,wd,filt,frames)

  - **n** ファイル数 [**Default: Inf**]
  - **typ** ファイルの種類 [**Default: 'any'**]

    'image' 画像ファイル (**".img" and ".nii"**)

    'dir' ディレクトリ
  - **mesg** プロンプト [**Default: 'Select files...'**]
  - **sel** 既に選択されたリスト [**Default: {}**]
  - **wd** 最初に選ぶディレクトリ [**Default: pwd**]
  - **filt** ファイル選択のためのフィルタ [**Default: '.*'**]
  - **frames** （4次元データの）フレーム番号 [**Default: '1'**]

Matlabから実行

```
>> subj = spm_select(Inf,'image');
```

data ディレクトリにある、V_subj[1-3].nii を選択

subj

```
>> subj

subj =

'img_data/nisg-201912/ex3/data/V_subj1.nii,1'

'img_data/nisg-201912/ex3/data/V_subj2.nii,1'

'img_data/nisg-201912/ex3/data/V_subj3.nii,1'
```

- 3行の行列
- 1例目を指定したかったら、deblank(subj(1,:))
  - 空白があった場合は、deblank で行末の空白を削除しないとエラーになる

作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

ワーキングディレクトリの選択

- ディレクトリの選択にも、**spm_select** を用いることができる
- 29行目あたりを以下のように変更（すべて1行）

```
cwd = spm_select(1,'dir','Select working directory…');
```

dir → cwd

Messages → Select working directory

作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

forループ

- 被験者の数だけ繰り返す
- 被験者の数を知る方法

```
size(subj,1) %変数 subj の行数
```

- forループは以下のようになる

```
for i = 1:size(subj,1)
   ループの中身
end
```

forループとmatlabbatch

- **forループ後のmatlabbatch**
  - **matlabbatch{1}** 1人目ワーキングディレクトリ
  - **…**
  - **matlabbatch{9}** 1人目PNG作成
  - **matlabbatch{10}** 2人目ワーキングディレクトリ
  - **…**
  - **matlabbatch{18}** 2人目PNG作成
  - **matlabbatch{19}** 3人目ワーキングディレクトリ
- 被験者〇人目から自動でmatlabbatchの番号を作る方法は？

被験者No.とmatlabbatch番号の関係

被験者番号 変数 i	matlabbatch番号 変数 j+1
1	1
	(2-9)
2	10
	(11-18)
3	19
	(20-27)
4	28
	(29-36)
5	37
	(38-45)
6	46
	(47-54)

- 左の表で、被験者番号の変数を i, バッチ番号の変数を j+1 とする
  - j+1 とする理由は、コアバッチが、{1} からはじまっており、番号の前に "j+" とだけつけたら楽だから。
- i と j+1 の間には、j+1 = ai+b (aとbは未知）という関係がある

```
 1 = a*1+b
10 = a*2+b
a = 9; b = -8
j+1 = 9*i-8
 j = 9*i-9
```

for ループの修正

- 33行目あたりを以下に修正

```
for i = size(subj,1)
```

```
filelist → subj
```

- その下に以下を挿入

```
j = 9*i-9;
```

- 36行目、37行目あたりにあるコメントはもう不要のため、削除

作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

spm_fileparts

- spm_select で得られたファイルに関する情報をディレクトリ、ファイル名、拡張子などに分解することができる
- subj の1行目を分解してみる

```
>> subj %3人分のデータがあるはず

>> [dir fname ext] = spm_fileparts(subj(1,:))

dir =

    '/自分のパス/img_data/nisg-201912/ex3/data'

fname =

    'V_subj1'

ext =

    '.nii'
```

fullfileを使ってディレクトリを作成

- `fullfile` を使って、現在のディレクトリの下にある`'stats'`ディレクトリの下に先程得られた `fname` を使ったパスを作成（まだこの時点ではディレクトリは作成されない）

```
>> subjstat = fullfile(pwd,'stats',fname)

subjstat =

'/自分のパス/img_data/nisg-201912/ex3/stats/V_subj1'
```

- `mkdir` を使って、実際にディレクトリを作成する

```
>> mkdir(subjstat)
```

ディレクトリ作成のスクリプト化

- 34行目の j = 9*i-9 の下、36行目あたりに以下を記載

```
%% make a stat directory for each subject
[dir fname ext] = spm_fileparts(subj(i,:));
subjstat = fullfile(cwd,'stats',fname);
mkdir(subjstat);
```

- spm_filepartsの中の subj(i,:)に注意
  - forループに対応させるために、subj(i,:)となっている

作り上げるバッチのイメージ

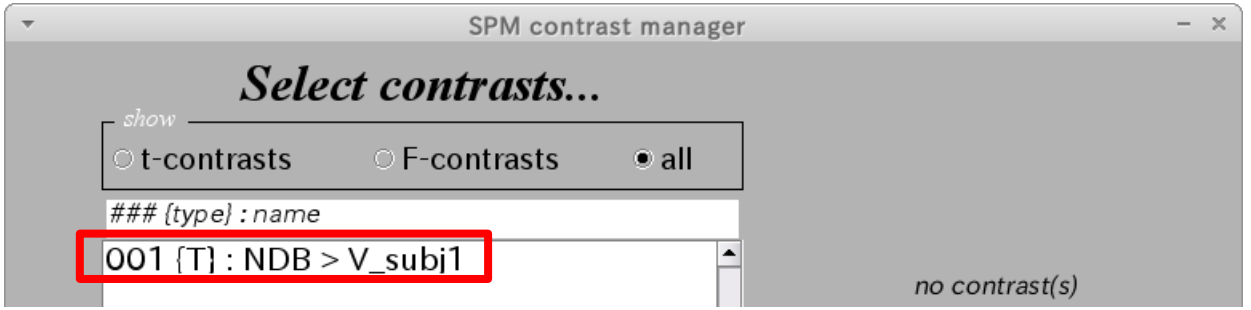
- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

コントラスト名

- SPMのコントラストにつける名前は、先程の fname を応用すればつけられる
- コントラストの名前は結果のタイトルにもなる
- Matlabは文字列は、行列でつなげれば表せる
- 今、"NDB > V_subj1" という風に記載したい
- 42行目あたりに以下を記載

```
%% Set Contrast name

conname = ['NDB > ' fname];  % 'NDB > 'は任意
```



作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

matlabbatchの行数確認

- 皆がエラーを起こさないため、matlabbatchの開始行数を揃える
- 45行目から開始
- 45行目が以下になっているように、適宜空白行を挿入/削除

```
matlabbatch{1}.cfg_basicio.file_dir
```

- ここでファイルを保存

matlabbatch番号の変更

- **matlabbatch{1} → matlabbatch{j+1}** のように変更
- **matlabbatch{1}**から **matlabbatch{9}** まで変更

- 「検索」を用いて、**batch{** を **batch{j+** に置換
  - **{1}**だけ**{j+1}**に置換すると、変更すべきでないところも置換してしまうリスクがある
- ひとつひとつ確認してから保存

Dependencyのバッチ番号の変更

- matlabbatchでは、dependencyが設定されているところは、**cfg_dep** となっている
- 最初の **substruct** の**4つ目**の引数がmatlabbatch番号に対応するため、ここも変更する必要がある
- **cfg_dep** を検索して見つける
- 81行目を確認

```
matlabbatch{j+3}.spm.spatial.smooth.data(1) =
cfg_dep('Segment: wc1 Images', substruct('.','val',
'{}',{2}, '.','val', '{}',{1}, '.','val', '{}',{1}),
substruct('.','tiss', '()',{1}, '.','wc', '()',
{':'}));
```

- ここの {2} を {j+2} に変更

Dependencyに関係する修正行

- 162行
```
matlabbatch{j+4}.spm.stats.factorial_design.des.t2.scans2(1) = cfg_dep('Smooth: Smoothed Images', substruct('.','val', '{}',{j+3}, '.','val', '{}',{1}, '.','val', '{}',{1}), substruct('.','files'));
```

Dependencyに関係する修正行

- 175行

```
matlabbatch{j+5}.spm.stats.fmri_est
.spmmat(1) = cfg_dep('Factorial
design specification: SPM.mat
File', substruct('.','val', '{}',
{j+4}, '.','val', '{}',{1},
'.','val', '{}',{1}),
substruct('.','spmmat'));
```

Dependencyに関係する修正行

- 178行

```
matlabbatch{j+6}.spm.stats.con.spmmat(1) = cfg_dep('Model estimation: SPM.mat File', substruct('.','val', '{}',{j+5}, '.','val', '{}',{1}, '.','val', '{}',{1}), substruct('.','spmmat'));
```

Dependencyに関係する修正行

- 183行

```
matlabbatch{j+7}.spm.stats.results.
spmmat(1) = cfg_dep('Contrast
Manager: SPM.mat File',
substruct('.','val', '{}',{j+6},
'.','val', '{}',{1}, '.','val',
'{}',{1}),
substruct('.','spmmat'));
```

Dependencyに関係する修正行

- 195行

```
matlabbatch{j+8}.spm.util.render.di
splay.conspec.spmmat(1) =
cfg_dep('Contrast Manager: SPM.mat
File', substruct('.','val', '{}',
{j+5}, '.','val', '{}',{1},
'.','val', '{}',{1}),
substruct('.','spmmat'));
```

バッチの個々の設定

- ここまでで一回保存

- これから matlabbatch の個々の設定を見ていく

matlabbatch{j+1}

- ディレクトリの指定
- ここは先程指定した **cwd** を設定すれば終わり
- 45行目を以下に修正

```
matlabbatch{j+1}.cfg_basicio.file_dir.dir_ops.cfg_cd.dir = {cwd};
```

matlabbatch{j+2}

- Segmentation
- 設定項目

  spm.spatial.preproc.channel.vols

  → **Segmentation**を行うデータ

  spm.spatial.preproc.tissue(?).tpm

  → **TPM**のパス

spm.spatial.preproc.channel.vol

- 46行目の{'/home/foo/img_data/nisg-201912/ex3/data/V_subj1.nii,1'} には、被験者データが入る
- ここの部分を、{'/home/.../V_subj1.nii,1'} を、{deblank(subj(i,:))} に修正

```
matlabbatch{j+2}.spm.spatial.preproc.channel.vols = {deblank(subj(i,:))};
```

spm.spatial.preproc.tissue(1).tpm

- tissue probability map (tpm) の指定
- SPMのインストールフォルダは個々で違うが、SPMのディレクトリを変数にできる方法がある
  - **`spm('dir')`**
- **`spm('dir')`** と **`fullfile`** を使うと、**`tpm.nii`** の場所を指定できる

fullfile の使い方

- Matlabから以下をタイプ

```
>> fullfile(spm('dir'),'tpm','TPM.nii,1')

ans =

'自分のSPMの場所/tpm/TPM.nii,1'
```

- パスの部分を指定することで、パスを作成できる

spm.spatial.preproc.tissue(?).tpm

- VBMC_batch.m の tissue(1).tpm ～ tissue(6).tpm を修正
- preproc.tissue(1).tpm = {fullfile(spm('dir'),'tpm','TPM.nii,1')};
- preproc.tissue(2).tpm = {fullfile(spm('dir'),'tpm','TPM.nii,2')};
- preproc.tissue(3).tpm = {fullfile(spm('dir'),'tpm','TPM.nii,3')};

  *コピー＆ペーストを上手に使う

  *行末の）を忘れない

matlabbatch{j+3}

- Smoothing
- 修正点はない
- **`matlabbatch{j+3}.spm.spatial.smooth`** はいじらなくてよい

matlabbatch{j+4}

- 統計モデルの作成
- 設定項目

```
stats.facorial_design.dir

stats.factorial_design.des.t2.scans1

stats.factorial_design.masking.em
```

統計ディレクトリの設定

- **stats.facorial_design.dir** には、統計結果を格納するディレクトリを指定する
- 先程作成した **subjstat** を指定すればよい
- 以下に変更
- **matlabbatch{j+4}.spm.stats.factorial_design.dir = {subjstat};**

ノーマルデータベース

- **spm.stats.factorial_design.des.t2.scans1** には、ノーマルデータベースの画像を指定する
- 先程設定した **ndb** を用いるが工夫が必要
- spm_selectで選んだ画像の情報は、文字列の行列だが、複数行にわたる情報の場合、matlabbatchに格納するには、これを「セル配列」に変更する必要がある
- （複数行の）文字列をセル配列に変更するコマンドは、**cellstr**
- 以下に変更し、71行にわたる画像のリストは削除

```
matlabbatch{j+4}.spm.stats.factorial_design.des.t2.
scans1 = cellstr(ndb);
```

- この行の前後のコメント %% は削除

マスク画像の指定

- **spm.stats.factorial_design.masking.em**　にはマスク画像を指定する
- **mask**ディレクトリにある **mask.nii**　を指定
- **fullfile**と**cwd**を用いて指定
- 97行目あたりを以下に変更

```
matlabbatch{j+4}.spm.stats.factorial_design.masking.em =
{fullfile(cwd,'mask','mask.nii,1')};
```

matlabbatch{j+5}

- パラメーターの推定
- ここは修正項目なし

matlabbatch{j+6}

- コントラスト設定
- 設定項目
  - `spm.stats.con.consess{1}.tcon.name`

- コントラスト名を自動で指定
  - 先程作成した connname を使う
- 105行目あたりを以下のように修正

```
matlabbatch{j+6}.spm.stats.con.consess{1
}.tcon.name = conname;
```

matlabbatch{j+7}

- 結果表示
- 設定項目
  - `spm.stats.results.conspec.titlestr`

- 結果に使う文字列
  - ここも先程作成した connname を使う
- 110行目あたりを以下のように修正

```
matlabbatch{j+7}.spm.stats.results.conspec.titlestr = conname;
```

matlabbatch{j+8}

- 脳表上へのレンダリング像を作成
- 設定項目
  - **spm.util.render.display.rendfile**

- **fullfile** と **spm('dir')** を利用

```
matlabbatch{j+8}.spm.util.render.display.rendfile
=
{fullfile(spm('dir'),'rend','render_spm96.mat')};
```

作り上げるバッチのイメージ

- SPMを起動
- matlabbatchの初期化
- データ/ディレクトリの選択
  - 被験者データの選択
  - ノーマルデータベース（NDB）の選択
  - ワーキングディレクトリの選択
- forループ
  - 被験者ごとの統計ディレクトリ作成
  - コントラスト名、結果表示のタイトルを設定
  - matlabbatch本体
- matlabbatchの実行

matlabbatchの実行

- バッチの最後に、matlabbatchを実行するコマンドを挿入する必要がある
- 2通りの方法がある
  - Batch Editorを立ち上げる

    ```
    spm_jobman('interactive',matlabbatch);
    ```

    - ユーザは確認してから実行
  - Batch Editorを立ち上げずにそのまま実行

    ```
    spm_jobman('run',matlabbatch);
    ```

    - 自動化したい場合はこちら

matlabbatchの実行（確認のみ）

- どちらも記載しておいて、使わない方をコメントアウトすると便利
- 135行目～137行目あたりに以下の記載を確認

```
%% Run batch
spm_jobman('interactive',matlabbatch);
%spm_jobman('run',matlabbatch);
```

- これでスクリプトの完成！
- さらに下にある情報はコメントのため、消しても消さなくてもよい
- 保存

バッチを実行する前に

- statsディレクトリの中をきれいにしておき、エラーが起きないようにする
- Matlabでのディレクトリの削除を学ぶ

Matlabでのディレクトリの削除

- Matlabでは、**rmdir** というコマンドでディレクトリを削除できる
  - ただし、ディレクトリが空の場合
- ディレクトリにデータが入っている場合は、**'rmdir** ディレクトリ名 **s'** と、ディレクトリ名の後に、半角スペース+s をつけることで、削除できる
- ワイルドカードにも対応している
  - V_subj1 V_subj2 を削除したい場合

```
rmdir V_* s
```

statsディレクトリのサブディレクトリを削除

- statsディレクトリ内にある、V_からはじまるサブディレクトリを一度に削除

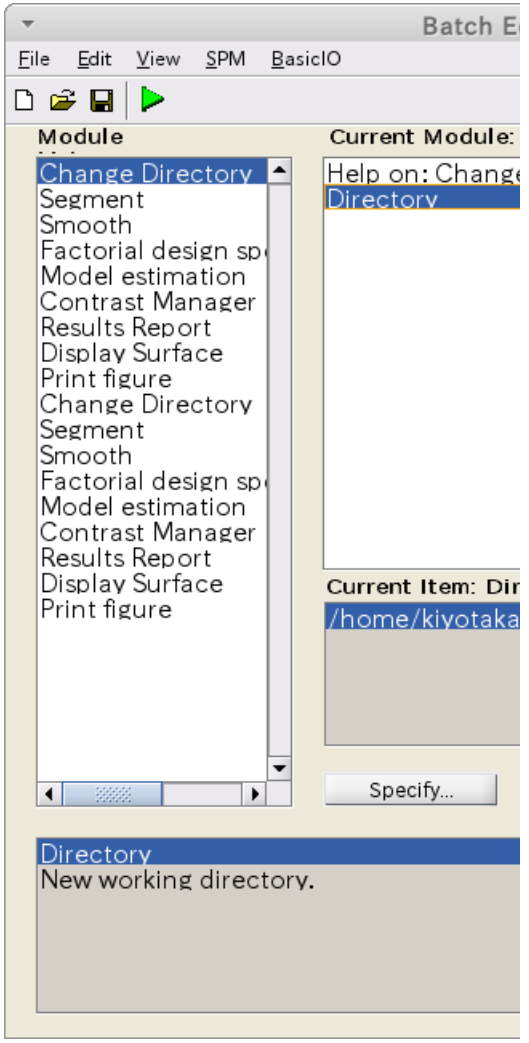
```
>> cd stats
>> ls   %ディレクトリ内のファイル・ディレクトリを確認
>> rmdir V_* s
```

- >> cd .. %元のディレクトリに戻る

バッチを実行

```
>> VBMC_batch
```

- 最初の画面で**V_subj1**と**V_subj2**を選択
- 次の画面で**NDB**内の**71**人を選択
- ワーキングディレクトリに現在のディレクトリを選択
- 右図が出ればうまくいっているはず
- 実行



エラーはつきもの

- 一度でうまくいったことはない
- 次のようなエラーが出る
  - エラー: ファイル: VBMC_batch.m 行:97 列:95
  - 式が無効です。関数の呼び出しまたは変数のインデックス付けにはかっこを使用してください。そうでない場合、区切り記号の不一致をチェックしてください。
- たいていはかっこ忘れ
- エラーの出た場所を探す
  - Matlabの右下に、カーソルのある行と列が表示される
- 修正した後に改めて **VBMC_batch** を実行

Questions?