SPM programming

How to batch and programme using SPM functions

Cyril Pernet, PhD – BRIC Edinburgh

Overview

- Level 1: the batch interface and job manager
- Level 2: SPM functions to pipeline your analyses (not reviewed here)
- Level 3: using SPM functions to do your own stuff

Before we start

- Whenever using SPM functions, this is a good idea to load the defaults

- global Defaults; % using global you make sure it is there whatever function, workspace is used

- Defaults = spm_get_defaults

Programming using the job manager

spm8/matlabbatch/...
&
spm_jobman.m

Job manager

- The easiest way to loop through many subjects is to 1 – create a batch for 1 subject and 2 – write a simple ‘for subject1=x’ loop calling the batch and updating the information

- you can also create few different batch if you want to do stuff in between, e.g. 1 batch to preprocess then 2 load log file, analyse onsets etc and 3 stat batch

Job manager

- spm_jobman('initcfg')
- load batch
- output_list = spm_jobman('run',job)

Toy example

- Say I want to realign 10 images and compute an inclusive mask (sum images > 0)
- Via the interface I can create this job, save it and then look at how it is stored
- Create a script that does the same but loop trough my images

Batch Editor

File Edit View SPM BasicIO

Module List

Realign: Estimate & Reslice

Current Module: Realign: Estimate & Reslice

Help on: Realign: Estimate & Reslice	
Data	
. Session	10 files
Estimation Options	
. Quality	0.9
. Separation	4
. Smoothing (FWHM)	5
. Num Passes	Register to mean
. Interpolation	2nd Degree B-Spline
. Wrapping	No wrap
. Weighting	
Reslice Options	
. Resliced images	Mean Image Only
. Interpolation	4th Degree B-Spline
. Wrapping	No wrap
. Masking	Mask images
. Filename Prefix	r

Current Item: Resliced images

All Images (1..n)
Images 2..n
All Images + Mean Image
* Mean Image Only

Edit Value

Resliced images

All Images (1..n) : This reslices all the images – including the first image selected – which will remain in its original position.

Images 2..n : Reslices images 2..n only. Useful for if you wish to reslice (for example) a PET image to fit a structural MRI, without creating a second identical MRI volume.

All Images + Mean Image : In addition to reslicing the images, it also creates a mean of the resliced image.

Batch Editor

File Edit View SPM BasicIO

Module List

Realign: Estimate & Reslice
Image Calculator DEP

Current Module: Image Calculator

Help on: Image Calculator
Input Images DEP Realign: Estimate & Reslice: Realigned Images (Sess 1)
Output Filename std.img
Output Directory /home/cyril/spm8/data_set/Face_expe/
Expression (i1+i2+i3+i4+i5+i6+i7+i8+i9+i10) > 0
Options
. Data Matrix No - don't read images into data matrix
. Masking No implicit zero mask
. Interpolation Trilinear
. Data Type INT16 - signed short

Current Item: Expression

(i1+i2+i3+i4+i5+i6+i7+i8+i9+i10) > 0

Edit Value

f = 'i1>100'
* Make a mask from one image and apply to another
f = 'i2.*(i1>100)'
- here the first image is used to make the mask, which is applied to the second image
* Sum of n images
f = 'i1 + i2 + i3 + i4 + i5 + ...'
* Sum of n images (when reading data into a data-matrix - use dmtx arg)
f = 'sum(X)'
A String is entered.
The string must have at least 2 characters.

Matlab batch

- load toy_batch

```
matlabbatch{1}.spm.spatial.realign.est
write

ans =

      data: {{10x1 cell}}
   eoptions: [1x1 struct]
   roptions: [1x1 struct]
```

Data in (this is what we can change)

Estimate options (happy with that)

Write options (happy with that too)

Matlab batch

```
matlabbatch{2}.spm.util.imcalc

ans =

        input: [1x1 cfg_dep]
       output: ‘XXX.img'
       outdir: {'/home/cyril/spm8/data_set/Face_expe/'}
   expression: '(i1+i2+i3+i4+i5+i6+i7+i8+i9+i10) > 0'
      options: [1x1 struct]
```

Dependency worked out for me :-)

Jobman

```
output_list = spm_jobman('run',matlabbatch)
```

```
>> output_list{1}

ans =

     sess: [1x1 struct]
    rmean:
{'/home/cyril/spm8/data_set/Face_expe/RawEPI/meanasM03953_0005_0006.img,1'}

>> output_list{2}

ans =

    files: {'/home/cyril/spm8/data_set/Face_expe/std.img,1'}
```

Script matlabbatch/jobman

- clc; clear all
- global Defaults;
- Defaults = spm_get_defaults
- spm_jobman('initcfg')
- load toy_batch
- matlabbatch{2}.spm.util.imcalc.outdir = {loc};
- cd data_set/Face_expe/RawEPI
- all_images = dir('sM*.img');

- Note in the batch everything need to be a cell string

Script matlabbatch/jobman

```
for i=1:10:length(all_images)-10

    for j=i:i+9
        names{j,:} = [pwd '/' all_images(j).name];
    end

    matlabbatch{1}.spm.spatial.realign.estwrite.data{1} = names;

    matlabbatch{2}.spm.util.imcalc.output
    = sprintf(‘mask_%g.img',i);

    output_list = spm_jobman('run',matlabbatch);
end
```

When to use it?

- Works for everything but particularly useful to test and run several statistical models on several subjects, as there is not easy to use functions to set up the fMRI model

- Preprocessing, by contrast, is easy to set up even without the batch.

Programming using SPM functions

Which functions do what- 1st level

1st level analysis

- spm_slice_timing
- spm_coreg
- spm_preproc / spm_preproc8
- spm_fmri_design
- spm_spm

Going further with SPM

Using basic spm tools to create your own analyses

Loading and reading files

- [P,filter]=spm_select(Inf,'.*\.img$','Select Images ');
- V = spm_vol(P);
- Images = spm_read_vols(V);

Get data

- [Y] = spm_get_data(V,XYZ);

Move between spaces

```
C = [15 20 15]' % a voxel of interest

coordinate_in_mm = V.mat(1:3,:)*[C;1];

Inv = inv(V.mat);
coordinate_in_voxel = Inv(1:3,:)*[coordinate_in_mm ;1];
```

Write images

- V = spm_create_vol(V,varargin)
- spm_write_vol(Info_img,matrix_to_write);

Toy example

- Load a stat image, get max, get the coordinate, write a mask around the max

- global defaults
- defaults = spm_get_defaults;

- % load stat image
- [P,filter]=spm_select(1,'.*\.img$','Select Stat image ');
- V = spm_vol(P);
- Image = spm_read_vols(V);

- % find max
- M = max(Image(:));
- [x,y,z]=ind2sub(size(Image),find(Image == M));

Toy example

- % mni
- mni_coordinates = V.mat(1:3,:)* [x y z 1]';

- % create a mask
- mask = zeros(V.dim);
- mask(x-2:x+2, y-2:y+2, z-2:z+2) = 1;

- V.fname = 'my_mask.img';
- V.descrip = 'binary mask around the max';
- spm_write_vol(V,mask);