

canlab\_spm\_fmri\_model\_job.m & SPM Gui

|  |
| --- |
| function matlabbatch = canlab\_spm\_fmri\_model\_job(subjoutputdir, TR, hpfilterlen, images\_by\_run, conditions\_per\_run, onsets, durations, names, multi\_nuisance\_matfilenames, varargin) |

canlab\_spm\_fmri\_model\_job.m 는 spm fmri 를 실행시켰을 때, 설정들을 gui에서 마우스와 키보드를 이용하여 하나씩 입력하지 않고 코드로 한번에 입력할 수 있도록 만든 것이다. 이 코드는 참가자 한 명에 대해서 각각 실행된다.

[specify 1st-level]의 설정을 gui로 지정한 다음 batch를 저장하면,   
matlabbatch{1}.spm.stats.fmri\_spec 라는 structure안에 저장된다.  
[Estimate]의 설정은 matlabbatch{2}.spm.stats.fmri\_est에 저장된다.

* Current Module > Directory

>> matlabbatch{1}.spm.stats.fmri\_spec.dir = {subjoutputdir};  
subjoutputdir 는 본 model의 결과가 저장되는 directory 주소.

* Current Module > Timing parameters

>> matlabbatch{1}.spm.stats.fmri\_spec.timing.units = 'secs';  
Units for design > seconds : TR의 단위

>> matlabbatch{1}.spm.stats.fmri\_spec.timing.RT = TR;  
Interscan interval > 0.46 : TR을 입력하면 됨. 예시로 460ms

>> matlabbatch{1}.spm.stats.fmri\_spec.timing.fmri\_t = fmri\_t;  
Microtime resolution > 16 : 기본 설정대로

>> matlabbatch{1}.spm.stats.fmri\_spec.timing.fmri\_t0 = fmri\_t0;  
Microtime onset > 1 : 기본 설정대로

* Current Module > Data & Design > Subject/Session > Scans  
  이것은 fMRI 데이터를 위한 design matrix를 만드는 과정이다.

|  |
| --- |
| % Images (scans)  for j = 1:length(images\_by\_run) % from run1 to the last run    imgs = images\_by\_run{j};    if is4d || size(imgs, 1) == 1  imgs = expand\_4d\_filenames(imgs); % expand to volumes if necessary  end    if iscell(imgs)  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).scans = imgs;  elseif ~iscell(imgs)  imgs\_cell = mat2cell(imgs, ones(size(imgs, 1), 1), size(imgs, 2));  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).scans = imgs\_cell;  end    % Make sure there are no trailing blanks - will prevent from loading correctly  for i = 1:length(matlabbatch{1}.spm.stats.fmri\_spec.sess(j).scans)  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).scans{i} = deblank(matlabbatch{1}.spm.stats.fmri\_spec.sess(j).scans{i});  end  (continued) |

images\_by\_run은 전체 run에 해당하는 fmri image의 주소이다.

예를 들어 run 4개를 분석하는 경우에 images\_by\_run는 1x4 cell이며, 하나의 성분이 한 run의 image주소로 ‘data/FAST/imaging/preprocessed/sub-fast001/func/swdcrsub-fast001\_task-fastfmri\_run-03\_bold.nii’가 된다. 파일 이름을 보면 유추할 수 있겠지만, 이 이미지는 preprocessed data로 smoothing, warping, distortion correction, registration이 이루어진 이미지이다.

for images\_by\_run의 개수(=run 개수)만큼 for 문이 돌아간다. imgs는 imaegs\_by\_run 에서 한 성분을 뽑아낸 것. 즉, matlabbatch{1}.spm.stats.fmri\_spec.sess(j)에서 j 는 몇 번째 run인지를 나타낸다(아래에 계속 적용).

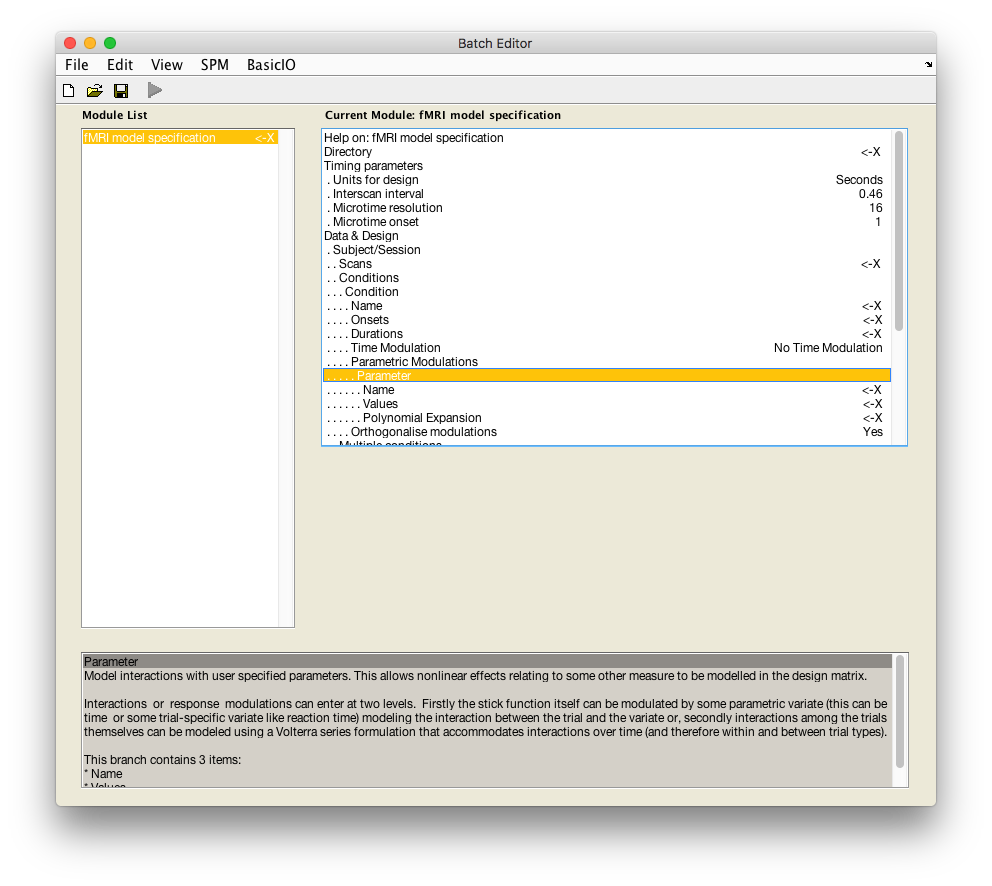
is4d는 본 함수의 character형 varagin으로 들어가는 옵션인데, images\_by\_run에 들어있는 파일 이름들을 4D로 지정하고 이들을 3D로 확장한다. 만약 run당 이미지가 하나라면 그들을 4D로 지정하고 자동적으로 확장하는 것을 시도한다. (…. 이해 더 필요)  
if is4d 옵션을 넣으면 is4d는 logical 1 이 된다. is4d가 ture(1)거나 imgs의 행이 하나일 때, imgs가 expand\_4d\_filenames(imgs)로 바뀌게 된다. 이 함수는 CanlabCore에 들어있는 함수이다. 이를 거치면 한 이미지 파일에 들어있는 TR 개수만큼 행이 여러 개가 생긴다. 예를 들어 imgs는 1990x125 인 char가 된다. 1990은 한 run의 TR 개수이며, 125는 char 길이를 의미한다.

if imgs가 cell일 경우 matlabbatch{1}.spm.stats.fmri\_spec.sess(j).scans에 imgs를 바로 넣고, cell이 아닐 경우 mat2cell을 이용하여 (TR 크기)x 1 크기의 cell의 imgs\_cell로 바꿔준 뒤, scans에 넣는다.

for imgs\_cell에 빈 공백이나 null을 제거해주기 위해 deblank 함수를 이용한다.

* Current Module > Data & Design > Subject/Session > Conditions
* Current Module > Data & Design > Multiple regressors

|  |
| --- |
| % Conditions and regressors  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).cond = struct('name', {}, 'onset', {}, 'duration', {}, 'tmod', {}, 'pmod', {}); % conditions    matlabbatch{1}.spm.stats.fmri\_spec.sess(j).multi\_reg = multi\_nuisance\_matfilenames(j); % is a cell % nuisance mat e.g. = '/.../reg.mat'; |

Conditions는 event-related, epoch-related response에 대한 정보를 넣는 곳이다. Conditions 설정 안에는 아래 그림과 같이 Name, Onsets, Durations, Time modulation, Parametric Modulations, Orthogonalise modulations라는 옵션들이 존재하는데, 이들은 matlabbatch상에서 name, onset, duration, tmod, pmod, orth(위 코드보다 아랫부분에 있음) 라는 이름으로 structure을 이루고 있다.   


Multiple regressors인 multi\_reg에는 j번째 run의 multi\_nuisance\_matfilenames을 넣어준다. 그런데 (j)라고 했기 때문에, multi\_reg 안에는 cell 형식이 들어가게 된다. multi\_nuisan…은 preprocessed 폴더의 참가자별로 이미 저장되어 있는 nuisance\_run#.mat의 주소이다. 이 파일은 변수 R을 저장하며, R의 열은 하나의 regressor을 의미한다.

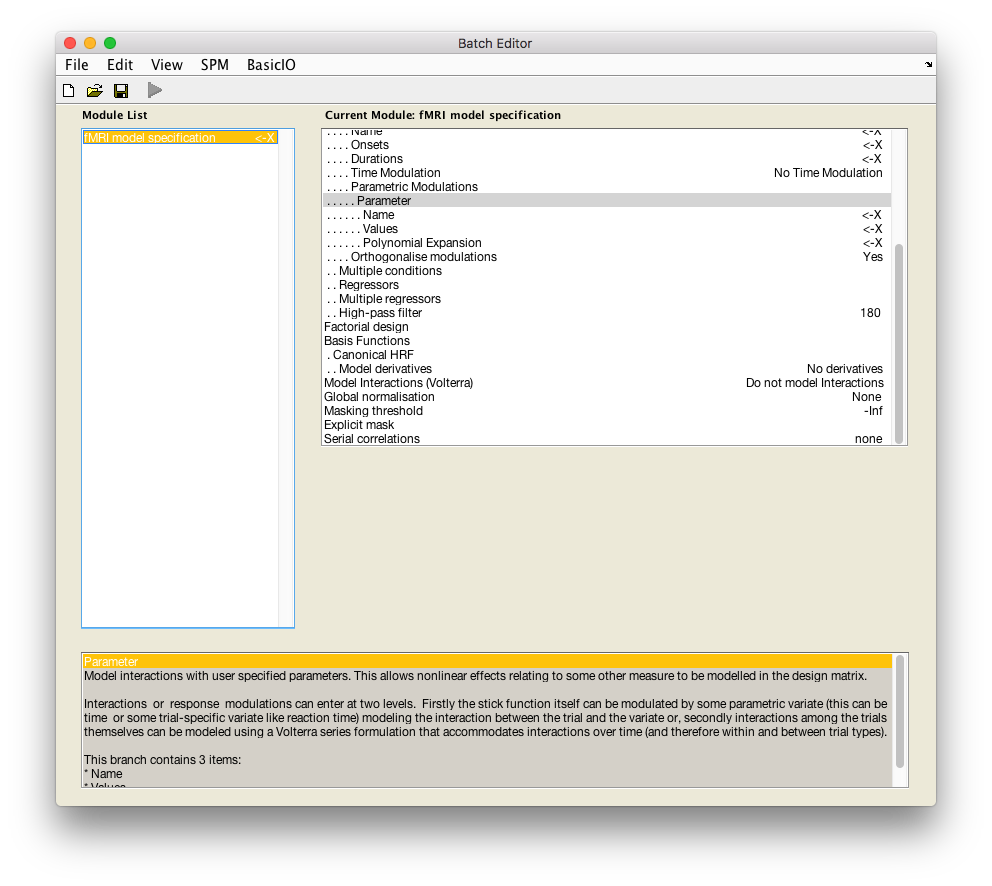
* Current Module > Data & Design > Subject/Session > High-pass filter  
  >> matlabbatch{1}.spm.stats.fmri\_spec.sess(j).hpf = hpfilterlen;  
  이것은 high-pass filter의 기준을 정해주는 것으로 기본은 128 seconds이다. 즉 주기가 128보다 긴, short frequency 신호는 사라지게 된다. Canlab의 기본은 180이다.
* Current Module > Data & Design > Subject/Session > Conditions 의 세부 설정

|  |
| --- |
| c = c;  % add time modulation by default  for j = 1:length(images\_by\_run) % from run1 to the last run  for k = 1:conditions\_per\_run(j)  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).cond(k).name = names{c};  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).cond(k).onset = onsets{c};  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).cond(k).duration = durations{c};  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).cond(k).tmod = timemod; % time mod – default  if length(pmodstruct) == 0  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).cond(k).pmod = pmodstruct;  else  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).cond(k).pmod = pmodstruct{c};  end  % 'tmod', {}, 'pmod', {}); % conditions    c = c + 1;  if isequal(spm('ver'), 'SPM12')  matlabbatch{1}.spm.stats.fmri\_spec.sess(j).cond(k).orth = orth;  end  end  end |

for j = 1:length(images\_by\_run) 을 통해 j는 run대로 for문을 돌아간다. 이어서  
for k = 1:conditions\_per\_run(j)을 통해 j번째 run의 condition 개수만큼 for 문을 돌아가게 된다. 즉 ~sess(j).cond(k)은 j번째 run의 k번째 condition에 대한 설정이 된다.

1. Name, Onset, Duration  
   이제 ~sess(j).cond(k)에 각 condition에 해당하는 name, onset, duration, time modulation에 대한 설정을 넣어준다. 이때 c는 1부터 max(j) x mak(k) = length(images\_by\_run) x conditions\_per\_run(j) 만큼 커지게 된다. 예를 들어 fast\_model1의 경우, conditions\_per\_run이 [41 41 41 41]로 한 run당 41개의 condtion이 존재하고, 4개의 run이 있다. 이때 names, onsets, durations 라는 변수는 164x1 cell이 된다. c는 두 개 for 문을 돌면서 하나씩 커지기 때문에 1에서 164까지 커지게 된다. 동시에 총 condition의 개수가 164개다.
2. Time Modulation  
   Time modulation은 시간의 효과를 model하는 것이다. linear 또는 nonlinear한 time effect를 model 할 수 있다. timemod를 1로 하면, linear한 1st order modulation이 적용된다. 본 코드에서 timemod는 기본이 1이지만, 옵션에서 ‘notimemod’를 선택하면 timemod가 0이 된다. nuisance에 linear time effect가 들어간 경우 timemod = 0으로 한다.
3. Parameteric Modulations

|  |
| --- |
| pmodstruct = struct('name', [], 'param', [], 'poly', []);  pmodstruct = pmodstruct([]);  switch(varargin{i})  case {'pmod'}  clear pmodstruct;  i=i+1; pmodstruct = varargin{i}; |

parametric modulation은 각 시행에서 brain 데이터와 어떤 변수와의 상호작용을 모델링하기 위해 stick function을 parametric variate(시간이나 RT와 같은 trial-specific variate)으로 modulate하는 것이다. (The stick function itself can be modulated by some parametric variate (this can be time or some trial-specific variate like reaction time) modeling the interaction between the trial and the variate.)  
pmod를 옵션으로 넣지 않으면 pmodstruct는 빈 structure가 되어 sess(j).cond(k).pmod 에 들어가고, 옵션으로 넣으면 'pmod' 옵션과 함께 varargin부분에 parametric cell(pmod)을 넣어주어야 한다. 그러면 varargin 부분의 입력이 pmodstruct가 된다. pmod 변수는 (run 수)x(condition 수) 개수의 cell이 들어있다. model 4의 경우 conditions\_per\_run이 [2 2 2 2]이기 때문에 pmod의 크기는 8x1 cell이었다. 각 cell안에는 name, param, poly라는 하위 구조가 들어있고 param에는 parameter variate이 들어있는 것이다. model 4의 경우에는 view와 rating으로 2가지 condition을 나눈 뒤 view에 해당하는 survey rating 값을 parameter로 넣어주었다.   


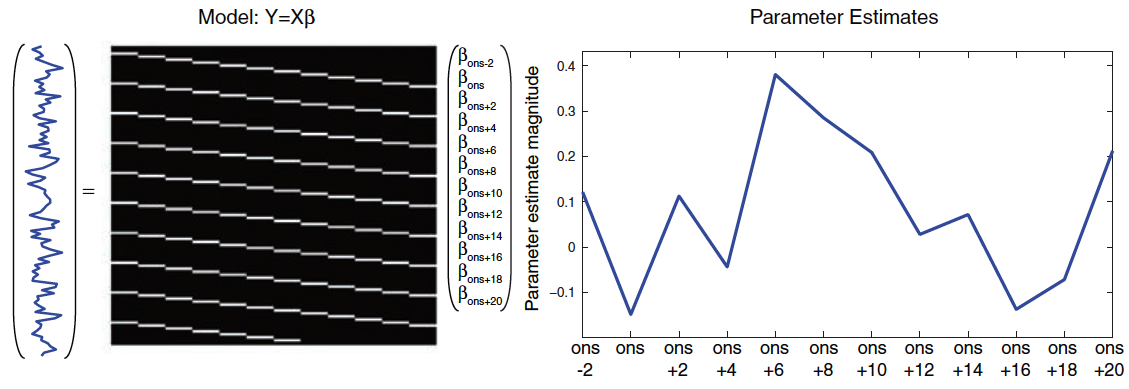
1. Orthogonalise modulations   
   무엇인지 잘 모르겠지만 기본 orth의 값은 1이고, 옵션 ‘orth\_off’를 하면 orth = 0.

* Current Module > Factorial design



Factorial design을 사용하지 않으면 name과 levels만 넣는다.  
>> matlabbatch{1}.spm.stats.fmri\_spec.fact = struct('name',{}, 'levels', {});

* Current Module > Basis Functions  
  Basis functions는 hemodynamic response를 모델하기 위한 함수이다. 일반적인 선택은 ‘Canonical HRF’이다. 하위 선택의 ‘Model derivatives’은 canonical HRF에 시간과 dispersion derivatives를 결합한 것이다. 이렇게 하면 참가자 간, voxel 간의 반응이 달라질 수 있다. ‘hrf’ 옵션은 이 기능을 활성화하고 싶을 때 사용한다.

‘fir’ 옵션은 HRF 대신에 conditions에 finite impulse response convolution을 사용하고 싶을 때의 옵션이다. 이때 varargin으로 length와 order을 넣어줘야 한다. FIR은 canonical HRF 대체 방법으로 length만큼의 time window에서 order 수 만큼을 쪼갠 뒤 각 order에서의 signal을 합하여 구한 파형을 HRF 대신 사용하는 것이다. 아래 그림을 보면 가장 왼쪽은 아래 방향을 시간의 흐름인 신호이다. Model에는 12개의 column, design matrix가 있고 첫 column을 보면 10개의 boxcar가 있다. 이 boxcar는 각각 한 stimulus 구간을 담당한다. 오른쪽 column으로 갈수록 각 stimulus의 length에서 order가 늘어나는 것(아래쪽으로 밀려진 모양)을 볼 수 있다. 여기서 length는 order의 수와 연속된 order간의 간격을 곱한 값이며 아래의 예시에서는 order가 12개이며 order간의 간격은 2초이다. 오른쪽의 그래프를 보면 첫번째 order의 경우는 stimulus가 제시되기 2초 전, 두번째 order는 stimulus가 제시된 순간, 마지막 order는 stimulus가 제시되고 20초 후의 신호를 다루고 있다. 오른쪽 그래프에는 각 order에 해당하는 신호들의 합을 보여주고 있다.   
  
fast의 model2의 경우에는 order는 40개, length는 40 x TR(0.46s)로 약 20초가 되었다.

|  |
| --- |
| case 'hrf'  bases.hrf.derivs = [varargin{i+1} varargin{i+2}];  i=i+2;  case 'fir'  i=i+1; bases.fir.length = varargin{i};  i=i+1; bases.fir.order = varargin{i};  if ~exist('bases','var')  bases.hrf.derivs = [0 0];  end  Matlabbatch{1}.spm.stats.fmri\_spec.bases = bases; % LUKA – left CANLAB default (hrf, no derivs), but allows hrf+derivatives or fir |

* Current Module > Model Interactions (Volterra)  
  matlabbatch{1}.spm.stats.fmri\_spec.volt = 1; % CANLAB – WE DO NOT WANT VOLTERRA; 1:none, 2:interactions  
  hemodynamic convolution을 first-order나 second order로 모델할 때
* Current Module > Global normalisation

|  |
| --- |
| matlabbatch{1}.spm.stats.fmri\_spec.global = 'None'; % CANLAB - WE DO NOT WANT GLOBAL |

Global normalisation은 fMRI data를 normalise하는 방법을 지정한다. 먼저 average within-brain fMRI signal 을 각 scan(n), session(s)마다 계산한다. ‘Scaling’ 옵션을 선택할 경우 각 scan, session마다 계산한 값을 100에서 나눠준다.   
‘None’을 선택할 경우 grand mean value 을 구하는데, 이것은 한 session의 모든 시간과 voxel들을 통틀어 fMRI signal의 평균이다. 우리는 ‘None’으로 사용.

* Current Module > Explicit mask

|  |
| --- |
| mask = which('brainmask.nii'); % CANLAB - WE WANT WHOLE-BRAIN EXPLICIT MASK, NOT IMPLICIT  matlabbatch{1}.spm.stats.fmri\_spec.mask = {[mask,',1']}; |

mask를 설정하면 그 안에 있는 voxel들만 분석된다.

* Current Module > Serial correlations

|  |
| --- |
| matlabbatch{1}.spm.stats.fmri\_spec.cvi = cvi;  % CANLAB DEFAULT - WE DO NOT WANT AR MODEL (options allow AR(1)) |

none / AR(1) / FAST 세가지 옵션이 존재.  
fMRI 신호에는 drift, motion, physiological noise등 여러 가지 noise가 존재한다.   
이러한 autocorrelation(non-independence of the observations over time)를 제거하기 위한 방법 중에 하나가 autoregressive process다.