Spatio-Temporal State-Space Analysis for fMRI **User Manual**

Firdaus Janoos

May 2, 2011

Contents

List of Figures

Chapter 1

Introduction

This manual describes the procedure for building and analyzing spatio-temporal state-space models of neural processes. Please refer to Janoos et al. [6]. A flowchart depiction of the algorithm developed in that paper is shown in Fig. 1.1

This algorithm has been implemented in a combination of MATLAB® codes, MATLAB® with Star-P®, and `C++`.

1.1 Single Session Analysis

The main steps involved in analyzing a single session[1] data-set are:

i Preprocessing for

  i.i Slice timing correction

  i.ii Head motion correction

  i.iii De-noising

  i.iv Head motion artifact removal

  i.v Other signal artifact removal (e.g. scanner drift, pulsatile signal, respiratory signal)

ii Building the feature space

iii Specifying the experimental design

iv Model estimation

[1]That is, to analyze one continuous fMRI time-series of one subject, one session. It may consists of one actual run, or multiples runs concatenated together. Essentially, one session is any fMRI time-series that shares the same anatomy and imaging characteristics.

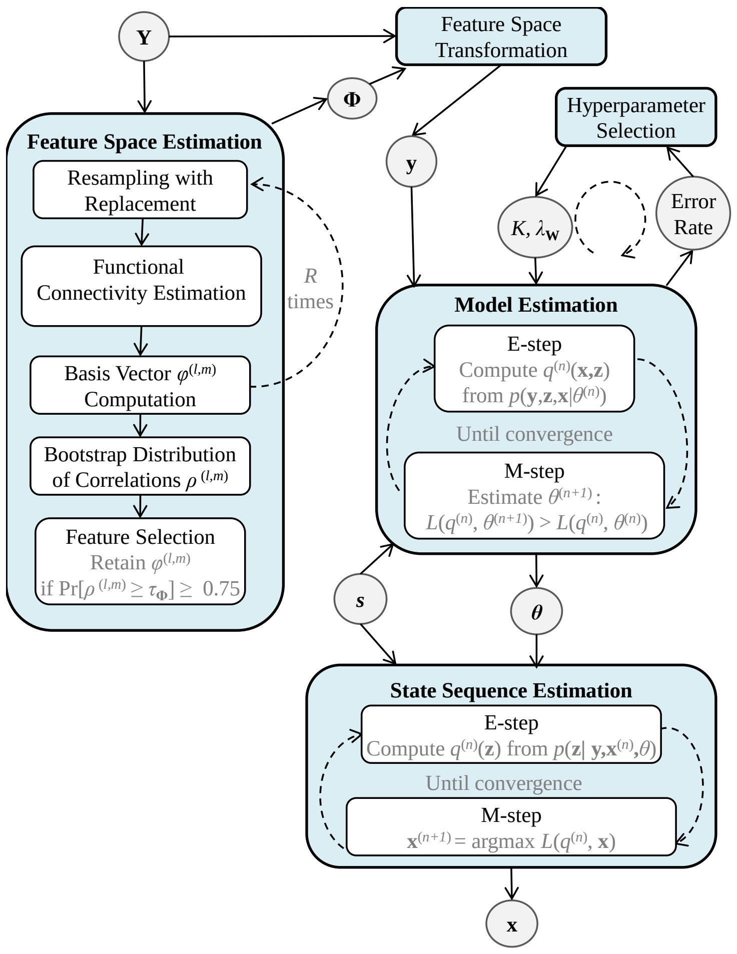

Y
Feature Space Transformation
Φ
Feature Space Estimation
Resampling with Replacement
Functional Connectivity Estimation
Basis Vector $\varphi^{(l,m)}$ Computation
Bootstrap Distribution of Correlations $\rho^{(l,m)}$
Feature Selection
Retain $\varphi^{(l,m)}$ if $\Pr[\rho^{(l,m)} \geq \tau_\Phi] \geq 0.75$
R times
y
Hyperparameter Selection
K, λ_w
Error Rate
Model Estimation
E-step
Compute $q^{(n)}(\mathbf{x},\mathbf{z})$ from $p(\mathbf{y},\mathbf{z},\mathbf{x}|\theta^{(n)})$
Until convergence
M-step
Estimate $\theta^{(n+1)}$: $L(q^{(n)}, \theta^{(n+1)}) > L(q^{(n)}, \theta^{(n)})$
s
θ
State Sequence Estimation
E-step
Compute $q^{(n)}(\mathbf{z})$ from $p(\mathbf{z}|\mathbf{y},\mathbf{x}^{(n)},\theta)$
Until convergence
M-step
$\mathbf{x}^{(n+1)}$ = argmax $L(q^{(n)}, \mathbf{x})$
x

Figure 1.1: Basis vectors of the feature–space Φ are computed from the functional connectivity of the brain estimated from the fMRI data $\mathbf{Y}$. Dimensionality reduction is performed by retaining stable basis vectors using bootstrapping. The data ($\mathbf{y}$), after projecting into the low dimensional feature–space are used to estimate model parameters θ through a generalized EM algorithm. Model hyper–parameters K and $\lambda_{\mathbf{w}}$ are selected to minimize the error of predicting the stimulus $\mathbf{s}$. Given a set of model parameters, the optimal state–sequence x^* is estimated using EM.

v Model exploration

1.2 Multi-Subject Analysis

In addition to the steps required for a single session analysis, during the preprocessing step data is spatially normalized into a reference coordinate system using non-linear registration.

Group level analysis is performed by comparing the models across subjects, in terms of state-space behavior and spatial activation patterns.

1.3 Prerequisites

1.3.1 Hardware

A really fast computer / workstation to examine and explore the results. Access to a kick-ass cluster to do the heavy duty computation.

1.3.2 Software

i MATLAB® on the work-station for design specification and results exploration.

ii MATLAB® with Star-P® installed and configured, for feature-space computation and model estimation.

iii SPM [8] version 5 or higher

iv Group ICA of fMRI Toolbox (GIFT) [2] version 1.2 or higher.

v An ISO Standard compatible C++ compiler.

vi ITK[4] version 2 or higher, installed and configured.

[2]`http://icatb.sourceforge.net/gift`

Chapter 2

Preprocessing

Each data-set will have different artifacts and characteristics and the pre-processing should be adapted to it. However, in general we have the following procedure to be effective in order for further spatio-temporal modeling.

In the discussion that follows, each session (run) is pre-processed individually.

2.1 Slice Timing and Motion Correction with SPM

Use the slice timing and motion correction modules from SPM5. For motion correction, apply the `Estimate and Reslice` operations (§ Fig. 2.1).

In order to remove residual motion artifacts from the time-series data (described in the next section), build a design matrix consisting only of the motion parameters estimated

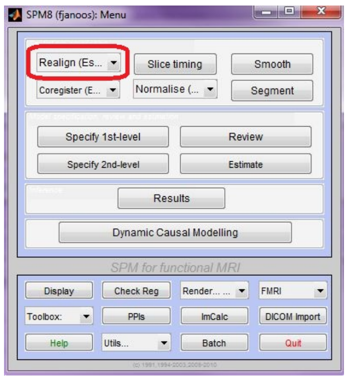


Figure 2.1: Estimate and Reslice Option of SPM8.

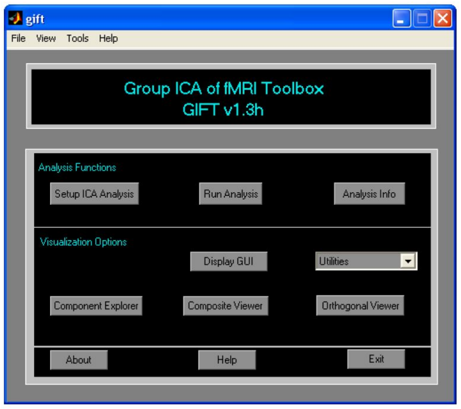


Figure 2.2: Group ICA for fMRI Toolbox (GIFT).

by SPM (`rp_*.txt`), and save it as an `SPM.mat` file[1].

For details, refer to the SPM Manual[8].

2.2 Motion Artifact Removal with GIFT

The motion correction operation itself tends to introduce strong artifacts Grootoonk et al. [3]. These artifacts are removed using ICA analysis with GIFT (§ Fig. 2.2). For this, first start up GIFT and configure it for analysis of an individual session(§ Fig. 2.4). Set the number of ICA to that automatically estimated by GIFT. Set ICA Algorithm to Infomax and Group ICA Analysis to Regular. Use the setup defaults for all other settings.

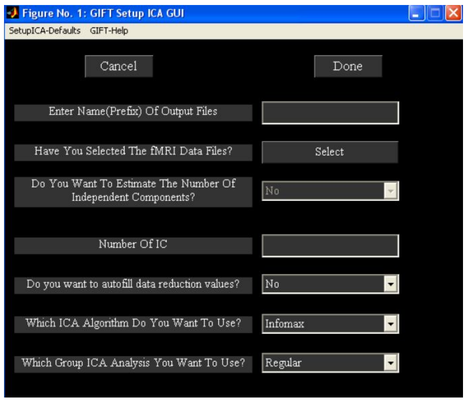
After configuring the GIFT analysis, run all the analysis steps

After the analysis is complete, open the `Component Explorer` and select the generated parameter file. Now, identify components of artifactual origin using one of two methods:

a Manual inspection of components for high concentration of intensity close to vessicles, tissue interfaces and image boundaries.

b Sorting component through multiple regression against motion parameters estimated in Section 2.1. This can be achieved the `Sort Components` feature of the visualization GUI and providing the design matrix of the SPM.mat file containing regressors built from the motion parameters (§ Fig. 2.5).

[1]By selecting the file in the Multiple Regressors field of the SPM Specify 1st-level GUI. Do not include regressors for any experimental conditions



Enter Name(Prefix) Of Output Files

Have You Selected The fMRI Data Files?

Do You Want To Estimate The Number Of Independent Components? No

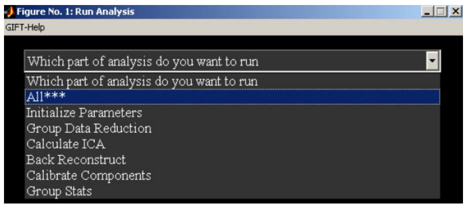
Number Of IC

Do you want to autofill data reduction values? No

Which ICA Algorithm Do You Want To Use? Infomax

Which Group ICA Analysis You Want To Use? Regular

Figure 2.3: Configuring a GIFT analysis session.



Which part of analysis do you want to run

Which part of analysis do you want to run
All***
Initialize Parameters
Group Data Reduction
Calculate ICA
Back Reconstruct
Calibrate Components
Group Stats

Figure 2.4: Group ICA for fMRI Toolbox (GIFT).

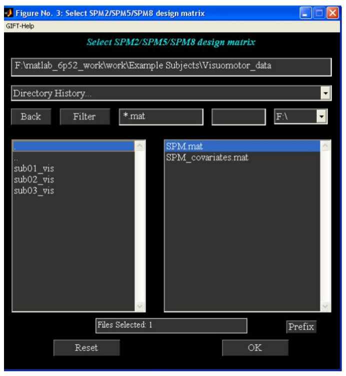


Figure No. 3: Select SPM2/SPM5/SPM8 design matrix
GIFT-Help
Select SPM2/SPM5/SPM8 design matrix
Back
Filter
Prefix
Reset
OK

F:\matlab_6p52_work\work\Example Subjects\Visuomotor_data

Directory History...

*.mat

F:\

.
..
sub01_vis
sub02_vis
sub03_vis

SPM.mat
SPM_covariates.mat

Files Selected: 1

Figure 2.5: Sorting components through regression in GIFT.

After the artifactual components have been identified, they can be effectively removed from the time-series data through the `Back Reconstruction` facility available in GIFT.

Please refer to the GIFT manual for more details.

2.3 De-noising

The preferred method of removing random $1/f$–type noise present in the fMRI time-course data is a wavelet based algorithm [1], that used a Wiener-type shrinkage for thresholding noise coefficient.

For this, the steps are as follows, in MATLAB[2]:

```
\% load the fmri dataset
vi = spm_vol( epi_series_copy_fnames );
preprocWaveletDenoise(vi, 'daubechies', 'wiener');
```

The options for

```
preprocWaveletDenoise( epi\_vol\_info\_struct,
 wavelet\_type, shrinkage\_type )
```

are:

- `epi\_vol\_info\_struct` - the array of `spm\_vol` structures of the EPI time-series. Note that the files pointed to by these structures are used both as input and output of the denoising routine.
- `wavelet\_type` - Valid choices are `daubechies`, `haar`, or `cdf97`.
- `shrinkage\_type` - Valid options are `hard`, `soft`, `sure`, `wiener`.

Next, the time–series data are high-pass filtered (0.5Hz) to remove artifacts due to breathing, blood pressure changes and scanner drift. Further polynomial trends are also eliminated through polynomial regression[3].

For this, use the same `spm\_vol` structure array obtained from above as input to

```
preprocDetrend( epi\_vol\_info\_struct,
freq\_cutoff, poly\_order )
```

are:

- `epi\_vol_info_struct` - the array of `spm\_vol` structures of the EPI time-series. Note that the files pointed to by these structures are used both as input and output of the denoising routine.

[2]Make sure to use a copy of the original data, as this routine will overwrite the volumes in the vi structure.

[3]The method uses a complex analytic representation of the signal and of a Legendre polynomial basis for regression.

- `freq\_cutoff` - The upper frequency for truncation.
- `poly\_order` - The highest order of the polynomial for detrending.

Chapter 3

Feature Space Computation

As seen in Fig. 1.1, computing the low-dimensional feature-space representation of the data involves the following steps:

i Generate a re-sample of the fMRI session.

ii Compute the functional connectivity for the resample.

iii Compute the orthogonal basis for the resample.

iv Retain or discard basis vectors using bootstrap analysis of stability[2].

v Project each fMRI volume (time-point) into the low dimensional feature space obtained above.

The algorithms that provide this functionality are extremely computation intensive though easily parallelized, and are therefore implemented in MATLAB® with Star-P®. To run this module, it essential to have Star-P® installed and configured with MATLAB. Also, it is recommended that a cluster or high-end high memory multi-processor computer be used. For an fMRI session with ≈ 500 volumes at 3mm resolution, a minimum of 32GB RAM is essential.

A example of feature-space computation and projection operation code fragment is:

```
% load the preprocessed fMRI session
vi_original = spm_vol( epi_series_preprocessed_fnames );
% load a mask for the ROI
vi_mask = spm_vol( roi_mask_fname);
basis_filename = {};
num_boot_iter = 1000;
% BOOTSTRAP : iterate over multiple resamples
for boot_iter = 1 : num_boot_iter
    % select a resample
    if boot_iter == 1
```

```
        % keep the first resample as the original sample
        vi_resample = vi_original;
    else
        [vi_resample] = fsResample( vi_original,
                                    ``block'', 10  );
    end
    % compute the connectivity matrix for the resample
    fsComputeConnectivity(vi_resample, vi_mask, 4,
                            0.25, connectivity_filename.bin);
    % compute the orthogonal basis vectors for the resample
    basis_filename{boot_iter} = [basis_filename_template,
            num2str(boot_iter)];
    fsOrthogonalize(connectivity_filename.bin,
                            basis_filename{boot_iter});
end
% perform the bootstrap analysis of stability
[basis_set_idx, num_basis] =
                            fsBASC(basis_filename, 0.8, 0.75);
% project original data on basis
[basis_coords]=  fsProjectBasis( vi_original, basis_set_idx
                                    basis_filename{1});
```

3.1 Resampling the fMRI Session

Function Syntax:

```
[vi_resample] = fsResample( vi_original, resamp_type,
option_1, option_2  );
```

Arguments:

- `vi_original` - The array of `spm_vol` structures of the original EPI time-series.
- `resamp_type` - Valid choices are ` ``block'' ` which selects the block bootstrap method described in [6] or ` ``wavelet'' ` that selects the wavelet based time-series bootstrapping method described in [7].
- `option_1` - For `resamp_type=``block''`, `option_1` is the length of a block in TR units. Typically `option_1=10` has been found to be suitable for different datasets. The reader is referred to [5] for more details on the tradeoffs involved. For `resamp_type=``wavelet''`, `option_1` is the size of the coarsest scale in the wavelet decomposition tree, in TR units. Wavelet coefficients over a support greater than this value are not computed for permutation.
- `option_2` - Not defined for `resamp_type=``block''`. For `resamp_type=``wavelet''`, `option_2` is the order (number of vanishing moments) of the wavelet basis. A

typical value is between 2 and 5.

Returns:

- `vi_resample` - The resample of the session.

3.2 Computing Functional Connectivity for a Resample

Function Syntax:

```
fsComputeConnectivity( vi_resample, fwhm,
                       hac_frac, output_filename)
```

Arguments:

- `fwhm` - The FWHM (in mm) of the Gaussian filter during the HAC step, typically set to 4.
- `hac_frac` - The number of HAC clusters as a fraction of the number original voxels.
- `output_filename` - The name of the output file in which to store the connectivity matrix. The matrix is stored row-wise in binary format (single precision float).

Notes: This function is implemented in `C++` with a MATLAB hook.

3.3 Computing the Orthogonal Basis Vectors

Function Syntax:

```
fsOrthogonalize(connectivity_filename_input,
                               basis_filename);
```

Arguments:

- `connectivity_filename_input` - The file containing the connectivity matrix.
- `basis_filename` - The file name (prefix) of the output files in which the basis vectors are stored (in zipped 4D NII format). Example, `/data/session/subject_basis_resample_4` will result in the N basis vectors corresponding to the 4-th resample of the session being stored in the file `/data/session/subject_basis_resample_4.nii.gz`.

3.4 Bootstrap Analysis of Stability

Function Syntax:

```
[basis_set_idx, num_basis] =
                      fsBASC(basis_filename_array, tau, p);
```

Arguments:

- `basis_filename_array` - A cell array of filenames of the 4D zipped NII (`.nii.gz`) each containing the basis vectors computed for a resample of the session.
- `tau` - The threshold at which correlations between the corresponding basis vectors across resamples are retained. Typically set between 0.6 and 0.9.
- `p` - The probability value of suprathreshold correlations, if when exceeded the corresponding basis vector is retained in the final low-dimensional set.

Returns:

- `basis_set_idx` - The indices of the basis vector retained.
- `num_basis` - The number of basis vectors retained

3.5 Project Session on to Basis

Function Syntax:

```
[fs_coords]=  fsProjectBasis( vi_original,
           basis_set_idx, basis_filename);
```

Arguments:

- `vi_original` - The array of `spm_vol` structures of the original EPI time-series.
- `basis_set_idx` - The indices of the basis vector retained.
- `basis_filename` - The filename of the 4D zipped NII (`.nii.gz`) containing the basis vectors computed for a particular re-sample of the session.

Returns:

- `fs_coords` - An number of scans times number of basis array giving the coordinates of each fMRI time-frame in the low dimensional feature space.

Chapter 4

Model Specification and Estimation

The state-space model (§. Fig. 4.1) requires as input:

i The fMRI data (projected into the low-dimensional feature-space), described in Chapter 3.

ii Information about the stimulus presentation timing, described next.

A certain amount of information about the experimental paradigm can be provided as input to the SSM estimation routines, in order to stabilize estimation and provide a reference for estimating the model hyper-parameters (using prediction error, measured through cross-validation). In general, if an experiment has multiple types (channels) of stimuli and subject responses prevalent during its course, only some of these channels may be included in the model estimation, as necessary. The effect of the remaining experimental parameters on the SSM can be tested post-hoc, i.e. after the model has been estimated. The reader is referred to [5, 6] for more details on this.

4.1 Specifying the Experimental Design

The experimental design, to be provided as input to the model-specification, is specified using the `Specify 1-st Level` feature of SPM (§. Fig. **??**. Please refer to the SPM Manual [8] for more details.

Of interest are the different conditions, their presentation timing and duration. The rest of the fields may be filled with filler or dummy values. Save the `Specify 1-st Level` batch job as a `.mat` file. This `.mat` is provided as input the SSM estimation routine. There is no need to run the job - as the `SPM.mat` containing the design matrix *is not used*.

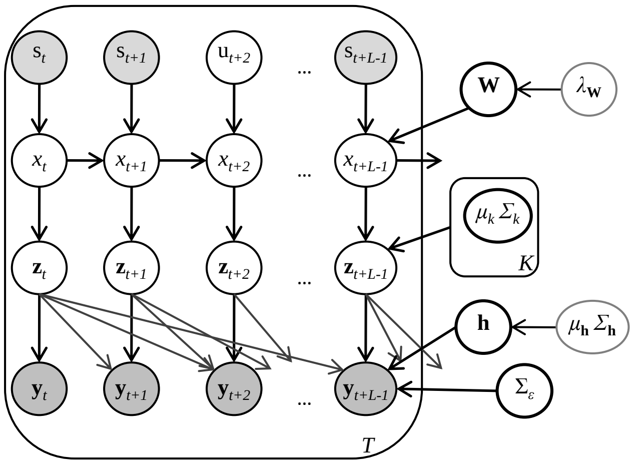
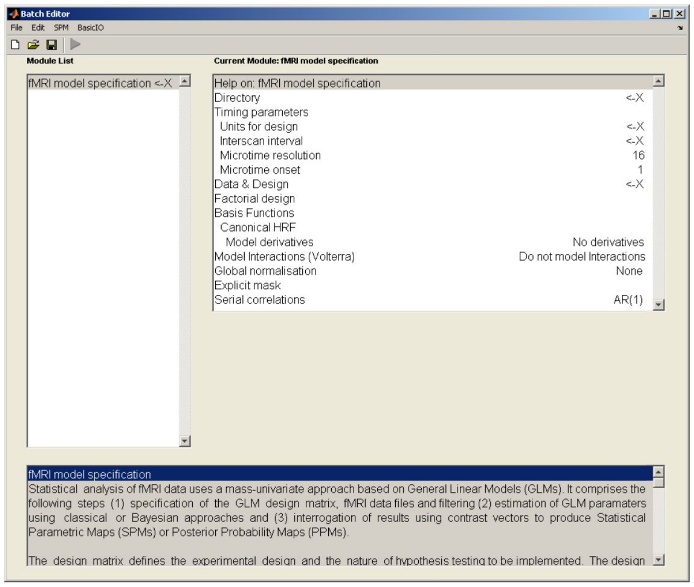


Figure 4.1: **The State–Space Model (SSM).** The experimental parameters are represented by $\mathbf{s}_t$, while the corresponding brain–state is x_t, and the instantaneous activation pattern is $\mathbf{z}_t$. The activation pattern is observed in the fMRI data $\mathbf{y}_t \ldots \mathbf{y}_{t+L-1}$ after convolution with the HRF $\mathbf{h}$.


Batch Editor
File Edit SPM BasicIO
Module List
fMRI model specification <-X
Current Module: fMRI model specification

Help on: fMRI model specification	
Directory	<-X
Timing parameters	
Units for design	<-X
Interscan interval	<-X
Microtime resolution	16
Microtime onset	1
Data & Design	<-X
Factorial design	
Basis Functions	
Canonical HRF	
Model derivatives	No derivatives
Model Interactions (Volterra)	Do not model Interactions
Global normalisation	None
Explicit mask	
Serial correlations	AR(1)

fMRI model specification

Statistical analysis of fMRI data uses a mass-univariate approach based on General Linear Models (GLMs). It comprises the following steps (1) specification of the GLM design matrix, fMRI data files and filtering (2) estimation of GLM paramaters using classical or Bayesian approaches and (3) interrogation of results using contrast vectors to produce Statistical Parametric Maps (SPMs) or Posterior Probability Maps (PPMs).

The design matrix defines the experimental design and the nature of hypothesis testing to be implemented. The design

Figure 4.2: **Specifying the Experimental Design.** Use the “Specify 1-st Level” of SPM to specify the experimental conditions and the timing.

4.2 Estimating the Model

To estimate the model, and do hyper-parameter selection, the following function is used:

Function Syntax:

```
[error_missing, log_likelihood, params_opt ]
    = ssmEstimateFull(fs_coords, specify_mat_file,
                      num_cvs, block_length,
                      tol);
```

Arguments:

- `fs_coords` - The feature-space embedding of the fMRI session
- `specify_mat_file` - The name of the `Specify 1-st spm Level` job batch file (`.mat` format) created in Section 4.1.
- `num_cvs` - (Optional, Default 5) Number of cross validation steps for hyper-parameter selection. Typically set between 5–10.
- `block_length` - (Optional, Default 5) The length of a block of missing stimuli (in TR units) for the cross validation method of hyper-parameter selection. Typically set between 4 – 8.
- `tol` - (Optional) The relative tolerance of the various parameter updates specifying the termination criteria for iterative optimization. Typically set at 0.001 (indicating termination for a 0.1% change in parameter value).

‘ Returns:

- `error_missing` - An array giving the distribution of the missing stimulus prediction error rate (over multiple cross validations) for the optimal model.
- `log_likelihood` - An array giving the distribution of the data log likelihood (over multiple cross validations) for the optimal model.
- `params_opt` - A structure of the SSM parameters averaged multiple CVs for the optimal model. It contains the fields:
  - `K_opt` - Optimal model size (number of states).
  - `lambda_opt` - Optimal value of the hyper-parameter controlling the estimation of the state-transition parameters.
  - `lambda_opt` - Optimal value of the hyper-parameter controlling the estimation of the state-transition parameters.
  - `w` - The 3-d array giving the state transition parameters $\mathbf{w}$, where `w(i,j,:)` = $\mathbf{w}_{i,j}$.
  - `omega` - The 2-d array giving the state transition parameters ω, where `omega(i,:)` = ω_i.

  - `mu` - An 2-d matrix where each column gives the mean value of the activation patterns corresponding to each state.
  - `Sigma` - An 3-d matrix where each 2-d sub-matrix gives the variance for each state.
  - `h` - An 2-d matrix where each column gives the estimated hemodynamic response filter for each element of the feature (data embedding) space.
  - `Sigma_eps` - A 2-d matrix giving the noise variance in the activation patterns.

It is also possible to estimate the model for a given setting of the hyper-parameters, and a given subset of the stimulus marked as hidden (unobserved) using the function described next: Function Syntax:

```
[error_missing, log_likelihood, params ]
    = ssmEstimate(fs_coords, specify_mat_file,
                        t_missing, K, lambda,  tol);
```

Arguments:

- `t_missing` - A vector of binary values the length of the session with 1 indicating which time-point to treat as missing / hidden stimulus.

‘

Chapter 5

Model Exploration

The spatio-temporal SSM encodes information about the abstract state of the subject, the spatial maps corresponding to the state, the state transition probabilities, and the hemodynamic response. In addition, the optimal state-sequence for a particular fMRI session (not necessarily the same as the one against which the SSM parameters were estimated) can also be computed.

The `params` object returned by `ssmEstimateFull` (or by `ssmEstimate` encapsulates all the parameters of the SSM, which can be explored by MATLAB code fragments. Addition, some helper functions have also been provided, listed below.

5.1 Computing Optimal State-Sequence

This function estimates the optimal state-sequence by a specific model for a given fMRI session.

Function Syntax:

```
[st_seq_opt, log_likelihood ]
    = ssmGetStateSeq(params, fs_coords_session);
```

Arguments:

- `params` - A structure of the SSM parameters as defined earlier.
- `fs_coords` - The feature-space embedding of the fMRI session

‘ Returns:

- `st_seq_opt` - An array
- `log_likelihood` - The log likelihood of the optimal state sequence

5.2 State Marginal Distribution

It is possible to compute the marginal probability distribution of each state conditioned on a specific stimulus $\Pr[x_t = k|\mathbf{s}_t]$ as follows

Function Syntax:

```
[log_marginal ]
    = ssmStateMarginal(params, cond);
```

Arguments:

- `params` - A structure of the SSM parameters as defined earlier.
- `cond` - A structure giving the value of each stimulus (experimental conditions) for which the marginal is desired. Stimuli omitted from this structure are marginalized out. For example, let the names of the conditions as given in the `.mat` file created during the `Specify 1-st Level` batch job of SPM, be `stimulus_1` ... `stimulus_n`. If the distribution of state probabilities for `stimulus_1`=5, `stimulus_3`=2 are required, regardless of the other stimuli, then `cond` will be

  ```
  cond.stimulus_1 = 5;
  cond.stimulus_3 = 2;
  ```

‘ Returns:

- `log_marginal` - An array giving the log marginal probabilities for each state, under the specified experimental conditions.

This function can be used to generate plots giving the distribution of each state during different experimental conditions, as shown in Fig. 5.1

5.3 Obtaining the State Transition Distribution

Similar to the method above, it is possible to compute the entire state transition probability matrix, conditioned on a specific stimulus $\Pr[x_t = j|x_{t-1} = i, \mathbf{s}_t]$ as follows

Function Syntax:

```
[log_transition]= ssmStateTransition(params, cond);
```

Arguments:

- `params` - A structure of the SSM parameters as defined earlier.
- `cond` - A structure giving the value of each stimulus (experimental conditions) for which the marginal is desired, as describe above.

Returns:

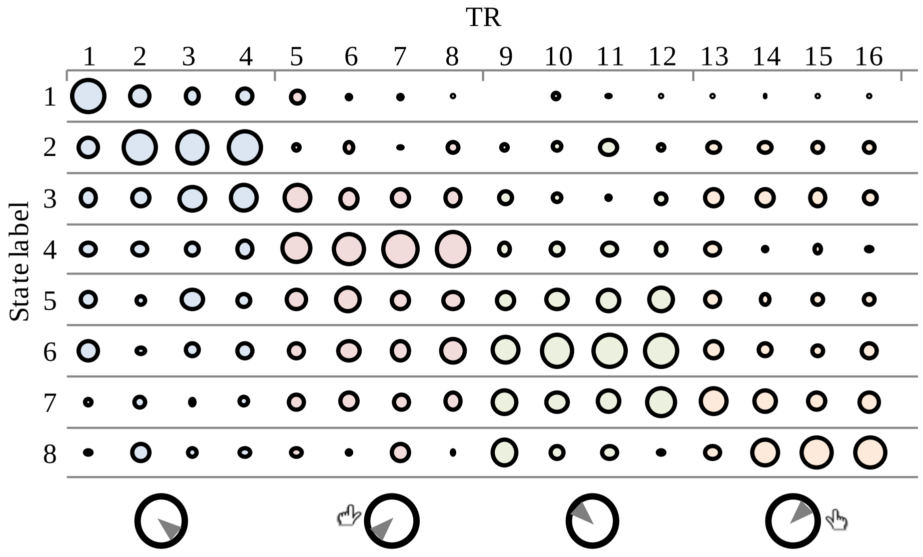


Figure 5.1: **Brain–state probabilities for one subject.** The size of the circles corresponds to the marginal probabilities of the states during the display of the wedge in lower right, lower left, upper left and upper right quadrants for 4TRs each. States have been relabeled for expository purposes and transition probabilities have been omitted for visual clarity.

- `log_transition` - An 2D array giving the log probabilities for a state transition from state i (row index) to state j (column index), under the specified experimental conditions.

5.4 The Activation Maps

The spatially distributed activation pattern for a specific value of the experimental variables along with its z–score map is computed using the next function.

Function Syntax:

```
[act_map_vi, z_map_vi ]  = ssmActivationMap(params, cond,
                           basis_set_idx, basis_filename);
```

Arguments: Arguments:

- `params` - A structure of the SSM parameters as defined earlier.
- `cond` - A structure giving the value of each stimulus (experimental conditions) for which the marginal is desired, as describe above.
- `basis_set_idx` - The indices of the basis vector of the feature-space that retained after dimensionality reduction (§Section 3.4)
- `basis_filename` - The filename of the 4D zipped NII (`.nii.gz`) containing the basis vectors computed for a particular re-sample of the session.

Returns:

- `act_map_vi` - An `spm_vol` structure giving the desired activation map. Filename format is
  `act_map_<cond_name_1>=<value>...<cond_name_n>=<value>.nii`
- `z_map_vi` - An `spm_vol` structure giving the desired z-score map. Filename format is
  `z_map_<cond_name_1>=<value>...<cond_name_n>=<value>.nii`

These z-score maps can be then examined and visualized either volumetrically, or on inflated cortical surfaces (using FreeSurfer, for example) as shown in Fig. 5.2.

5.5 The HRF Filters

The SSM estimates an HRF filter at each feature-space coordinate. These HRF estimates can be recombined and transformed back into the original (physical) space to give the HRF estimated at a specific location.

Function Syntax:

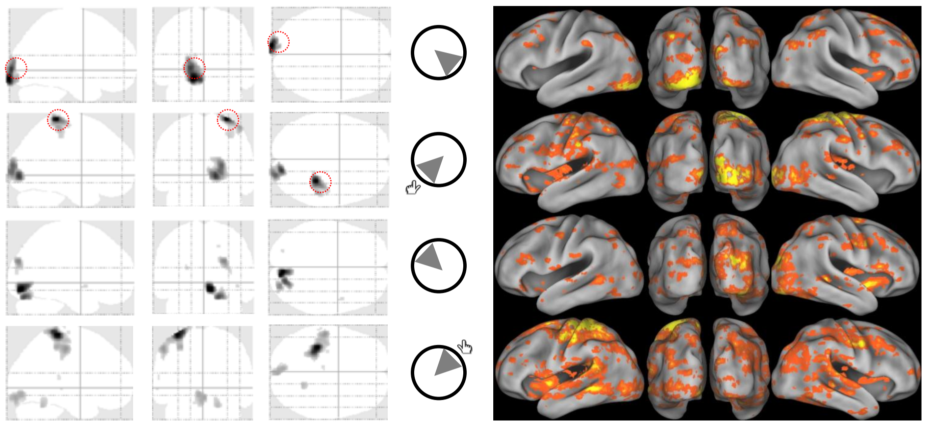


Figure 5.2: **Spatial Activation Maps for the Visuo-Motor Task from SPM8 and the State–Space Multivariate Analysis.** Fig. (a): Maximum intensity projections of significantly activated voxels ($p < 0.05$, FWE corrected) in a single subject for the four orientations of the wedge and the hand motor actions, computed using SPM8. The red circles indicate the ROIs for which the estimated HR FIR filters are displayed in Fig. 5.3. Fig. (b): Spatial (z–score) maps showing the distribution of activity for each orientation of the wedge computed from our state–space model displayed on an inflated surface of the brain. Displayed are the posterio-lateral and posterio-medial views of the left and right hemispheres respectively. Values of $z \leq 1$ have been masked out for visual clarity.

```
[hrfs]  = ssmGetHRF(params, spatial_ijk,
                              basis_set_idx, basis_filename);
```

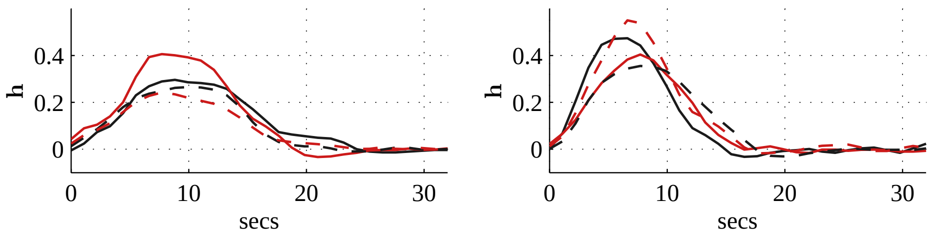
Arguments: Arguments:

- `params` - A structure of the SSM parameters as defined earlier.
- `spatial_ijk` - A $3 \times n$ array giving the voxel coordinates (in $i - j - k$) at which the HRF estimates are desired.
- `basis_set_idx` - The indices of the basis vector of the feature-space that retained after dimensionality reduction (§Section 3.4)
- `basis_filename` - The filename of the 4D zipped NII (`.nii.gz`) containing the basis vectors computed for a particular re-sample of the session.

Returns:

- `hrf` - An n item array of the HRF filter coefficients for the desired spatial locations.

The HRFs can then be visualized as in Fig. 5.3



(a) Motor Cortex ROI (b) Visual Cortex ROI

Figure 5.3: **Estimated hemodynamic FIR filter h.** The estimated FIR filter coefficients for each of the four subjects averaged in two ROIs selected in the motor and visual cortices.

REFERENCES

[1] Alexander, M., Baumgartner, R., Windischberger, C., Moser, E., and Somorjai, R. (2000). Wavelet domain de-noising of time-courses in MR image sequences. *Mag Res Imag*, 18(9):1129–1134.

[2] Bellec, P., Rosa-Neto, P., Lyttelton, O. C., Benali, H., and Evans, A. C. (2010). Multi-level bootstrap analysis of stable clusters in resting-state fMRI. *Neuroimage*, 51(3):1126–1139.

[3] Grootoonk, S., Hutton, C., Ashburner, J., Howseman, A. M., Josephs, O., Rees, G., Friston, K. J., and Turner, R. (2000). Characterization and correction of interpolation effects in the realignment of fMRI time series. *Neuroimage*, 11(1):49–57.

[4] Ibanez, L., Schroeder, W., Ng, L., and Cates, J. (2003). *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-10-6, http://www.itk.org/ItkSoftwareGuide.pdf, first edition.

[5] Janoos, F. (2011). *Spatio-Temporal Representations and Analysis of Brain Function from fMRI*. PhD thesis, Ohio State Univ.

[6] Janoos, F., Machiraju, R., Singh, S., and Mórocz, I. A. (2011). Spatio-temporal models of mental processes with fMRI. *Neuroimage, In Press*.

[7] Patel, R. S., Ville, D. V. D., and Bowman, F. D. (2006). Determining significant connectivity by 4d spatiotemporal wavelet packet resampling of functional neuroimaging data. *Neuroimage*, 31(3):1142–1155.

[8] The FIL Methods Group (2011). *SPM8 Manual*. Functional Imaging Laboratory, Wellcome Trust Centre for Neuroimaging, Institute of Neurology, UCL 12 Queen Square, London WC1N 3BG, UK.