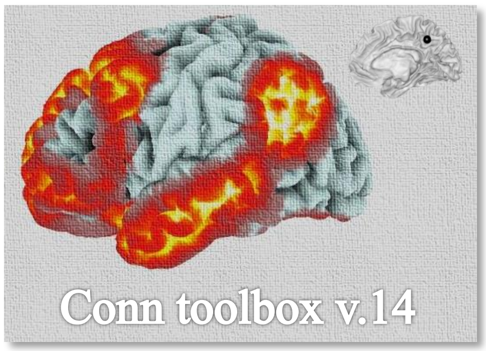
CONN – fMRI functional connectivity toolbox

BATCH processing manual



Gabrieli Lab. McGovern Institute for Brain Research
Massachusetts Institute of Technology
http://www.nitrc.org/projects/conn
Susan Whitfield-Gabrieli
Alfonso Nieto-Castanon

Table of contents

General information on batch process

This document describes batch processing in the CONN – fMRI connectivity toolbox (v.12.k). Batch processing permits the user to define and run connectivity analyses using matlab scripts (instead or in combination with the GUI). All of the definitions / analyses that can be performed from the conn GUI can also be performed using the batch functionality (or at least this was our intention, please let us know any additional functionality that we might be missing and/or that you would like added to the *conn_batch* processes).

The main function implementing batch processing is:

>> conn_batch(BATCH);

which executes the commands specified |by the BATCH structure.

The typical usage of *conn_batch* uses the following structure:

>> *clear BATCH;*

>> ... here one or several *BATCH.fieldnames=fieldvalues* definitions …

>> conn_batch(BATCH);

The BATCH structure can contain any number of different fields specifying different commands (e.g. it can contain a *Setup* field specifying the experiment information, and also simultaneously a *Preprocessing* field specifying the preprocessing steps). The different commands will be run in the natural order of the toolbox (first all the *New* commands, then *Setup*, then *Preprocessing*, then *Analysis*, and last *Results*). Additionally if BATCH is a cell array of BATCH structures, each of the elements in this cell array will be run sequentially.

Example:

Set the TR (inter-scan time) to 3 seconds and specifies the ROI files (two subject-independent ROIs) for the current experiment.

>> clear BATCH;

>> BATCH.Setup.RT=3;

>> BATCH.Setup.roi.names={‘Source_1’,’Source_2’};

>> BATCH.Setup.roi.files={‘/mypath/source1.img’,’/mypath/source2.img’};

>> conn_batch(BATCH);

The following sections describe the valid fields in the *BATCH* structure and their functionality in detail.

BATCH.filename

Description: Specifies a conn_*.mat project to work with. If this field is left empty or unspecified *conn_batch* will work on the currently open project.

> Note that when a BATCH command is issued specifying the *filename* field, the conn_*.mat project is first loaded, then modified with any additional changes indicated in the same BATCH structure, and last the conn_*.mat project file is saved again. In contrast, if a BATCH command is issued without specifying a filename field, the changes to the currently open project will not be saved to the current conn_*.mat project file after the modifications are made. You can always save a currently open conn_*.mat project by issuing a "*>> conn save;"* command.
> Also note that if the gui is open, the gui always shows the information regarding the currently open project, so *BATCH.filename* commands can alter the gui contents (you need to refresh the gui –clicking some button there- to see the changes).

Example:
>> BATCH.filename='/mypath/conn_foo.mat';

BATCH.New

Description: Performs spatial preprocessing steps (realignment, coregistration, segmentation, normalization, smoothing) on the functional and/or anatomical volumes. This process can also initialize a conn_*.mat project with the post-processed volumes as the original dataset. This step is optional on the typical connectivity toolbox processing pipeline. For example, if your data is already spatially pre-processed you can safely skip this step and start directly from the BATCH.Setup step. Additionally you might prefer to follow a different spatial-preprocessing pipeline (e.g. normalize your functional volumes to an EPI template) than the one implemented here. Keep in mind that the main goal of spatially preprocessing your data is to have all the data (functional, anatomical, and ROI volumes) coregistered to a common space. See '*help spm_jobman'* for a much extended batch implementation of arbitrary spatial-preprocessing steps of functional data.

Example: (realigns functional volumes of one subject –one session-)

```
>> clear BATCH;
>> BATCH.New.steps={'realignment'};
>> BATCH.New.functionals{1}{1}='mypath/subject1/session1/foo4d.nii';
>> conn_batch(BATCH);
```

The following sections describe the specific sub-fields of the *BATCH.New* structure.

BATCH.New.functionals

Description: Specifies the source functional volumes for spatial preprocessing. This field contains a double cell ranging over each subject and session, where the *BATCH.New.functionals{nsub}{nses}* element contains a char array pointing to the source functional volume(s) for subject *nsub* and session *nses*.

Example: (realigns functional volumes for two subjects, the first subject contains three scanning sessions, and the second subject only one session)

```
>> clear BATCH;
>> BATCH.New.steps={'realignment'};
>> BATCH.New.functionals{1}{1}='mypath/subject1/session1/foo4d.nii';
>> BATCH.New.functionals{1}{2}='mypath/subject1/session2/foo4d.nii';
>> BATCH.New.functionals{1}{3}='mypath/subject1/session3/foo4d.nii';
>> BATCH.New.functionals{2}{1}='mypath/subject2/session1/foo4d.nii';
>> conn_batch(BATCH);
```

BATCH.New.structurals

Description: Specifies the source anatomical volumes for spatial preprocessing. This field contains a cell array ranging over each subject, where the *BATCH.New.structural{nsub}* element contains a char array pointing to the source anatomical volume for subject *nsub*.

Example: (coregisters and reslices the functional volumes to the anatomical volume of one subject)

```
>> clear BATCH;
>> BATCH.New.steps={'coregistration};
>> BATCH.New.functionals{1}{1}='mypath/subject1/session1/rfoo4d.nii';
>> BATCH.New.structurals{1}='mypath/subject1/anatomical/foo.img';
>> conn_batch(BATCH);
```

BATCH.New.steps

Description: Specifies the spatial preprocessing steps. This field contains a cell array listing the steps to be performed (the order of entry is disregarded). Valid field values are: '*slicetiming*', *'realignment','coregistration','segmentation','normalization','smoothing',*and*'initialization'.* If this field is left unspecified *conn_batch* will perform all these pre-processing steps. All the spatial preprocessing steps are performed separately for each subject. The order in which these steps are performed is the following: a) rough affine coregistration of anatomical volumes to a template; b) segmentation of anatomical volumes; c) slice-timing correction of functional volumes; d) realign functional volumes; e) coregistration of functional to anatomical volumes; f) normalization of anatomical and functional volumes; and g) smoothing of functional volumes. Each of these steps can set on (or off) by including (or not) the following strings in the *BATCH.New.steps* cell array:

- **'slicetiming'**: Slice-timing correction of the functional volumes. The functional volumes are time-aligned to the acquisition time of the middle-slice during each acquisition, and the output file names are prefixed with the value 'a'.
- **'realignment'**: Realigns the functional volumes to the first scan of the first session. The functional volumes are resliced to the new orientation and the output file names are prefixed with the value 'r'.
- **'coregistration'**: Coregisters the functional volumes to the anatomical volumes. If 'normalization' is not included the functional volumes are resliced to the new orientation (and voxel size specified by the *BATCH.New.VOX* value) and the output file names are prefixed with the value 'r', otherwise the coregistration only affects the header information of the original functional volumes.
- **'segmentation'**: Segments the anatomical volumes into gray matter, white matter, and CSF areas. This procedure creates a set of output mask volumes with the anatomical volume file names prefixed with the values 'c1','c2', and 'c3' (corresponding to the gray, white, and CSF masks, respectively). It also creates a 'c0' mask file containing the skull-stripped anatomical volumes which is used in the 'coregistration' step, and a set of _seg_sn.mat files which are used in the 'normalization' step.
- **'normalization'**: Normalizes the anatomical volumes and applies the same transformation to the functional volumes. This step applies the _seg_sn.mat file transformation estimated during segmentation to the functional and anatomical volumes. The output files are resliced and resampled to the voxel size specified in the *BATCH.New.VOX* value, and prefixed with the value 'w'.
- **'smoothing'**: Spatially smooths the functional volumes. The output files are prefixed with the value 's'.
- **'initialization'**: Initializes a new conn_*.mat project (specified in the *BATCH.filename* field) with the post-processed anatomical, functional, and segmented volumes as the original dataset. This process will initialize the following setup information: a) **number of subjects**; b) **functional volumes** (set to the spatially-preprocessed functional volumes); c) **anatomical volumes** (set to the spatially-preprocessed anatomical volumes); d) **ROIs** defining the grey, white, and CSF areas (set to the output of the segmentation step); e) **conditions** ('Session' condition set to model the entire session data); f) **first-level covariates** ('realignment' covariate set to output of the realignment step - 6 estimated subject motion parameters- for each subject/session); and g) **second-level covariates** ('All' covariate set to model the entire group of subjects).

Example: (performs all spatial-preprocessing steps for one subject –one session- data, and initializes a *conn_foo.mat* project with this dataset)

```
>> clear BATCH;
>> BATCH.filename='/mypath/conn_foo.mat';
>> BATCH.New.steps={'slicetiming','realignment','coregistration','segmentation',…
'normalization','smoothing','initialization'};
>> BATCH.New.functionals{1}{1}='mypath/subject1/session1/rfoo4d.nii';
>> BATCH.New.structurals{1}='mypath/subject1/anatomical/foo.img';
>> BATCH.New.FWHM=8;
>> BATCH.New.VOX=2;
>> BATCH.New.sliceorder=[1:2:47,2:2:47];
>> conn_batch(BATCH);
```

BATCH.New.FWHM

Description: Specifies the Gaussian filter width of the spatial smoothing step (FWHM in mm). Only relevant when the value '*smoothing'* is contained in *BATCH.New.Steps*.

Example:
>> BATCH.New.FWHM=8;

BATCH.New.VOX

Description: Specifies the voxel size (in mm) after normalization or coregistration of the data. Only relevant when the value '*coregistration' or 'normalization'* is contained in *BATCH.New.Steps*. This value represents the voxel size that the connectivity analyses will be performed on.

Example:
>> BATCH.New.VOX=2;

BATCH.New.RT

Description: Defines the repetition time (inter-scan acquisition time) in seconds of the functional volumes

Example:

```
>> clear BATCH;
>> BATCH.New.RT=3;
>> conn_batch(BATCH);
```

BATCH.New.sliceorder

Description: Defines the acquisition order of each slice. A vector of integers, each integer referring the slice number in the image file (1=first), and the order of integers representing their temporal acquisition order. Only relevant when the value '*slicetiming'* is contained in *BATCH.New.Steps*.

Example:

```
>> clear BATCH;
>> BATCH.New.sliceorder=[1:2:47,2:2:47];
>> conn_batch(BATCH);
```

BATCH.New.template_structural

Description: Defines the anatomical template for approximate coregistration. By default it uses SPM’s *spm/template/T1.nii*.

Example:
```
>> clear BATCH;
>> BATCH.New.template_structural='/spm/template/T1.nii';
>> conn_batch(BATCH);
```

BATCH.New.template_functional

Description: Defines the functional template for functional normalization. By default it uses SPM’s *spm/template/EPI.nii*.

Example:

```
>> clear BATCH;
>> BATCH.New.template_structural='/spm/template/EPI.nii';
>> conn_batch(BATCH);
```

BATCH.Setup

Description: Defines experiment information and optionally performs initial data extraction steps.

Example: (creates a new project for the analysis of 10 subjects)

```
>> clear BATCH;
>> BATCH.filename='/mypath/conn_foo.mat';
>> BATCH.Setup.isnew=1;
>> BATCH.Setup.nsubjects=10;
>> conn_batch(BATCH);
```

The following sections describe the specific sub-fields of the *BATCH.Setup* structure.

BATCH.Setup.RT

Description: Defines the repetition time (inter-scan acquisition time) in seconds of the functional volumes

Example:

```
>> clear BATCH;
>> BATCH.Setup.RT=3;
>> conn_batch(BATCH);
```

BATCH.Setup.nsubjects

Description: Specifies the total number of subjects in the experiment (see ‘overwrite’ fields to learn more on how to add new subjects to a current experiment).

Example:

```
>> clear BATCH;
>> BATCH.Setup.nsubjects=20;
>> conn_batch(BATCH);
```

BATCH.Setup.spmfiles

Description: Specifies the location of SPM.mat files (cell array, one file path per subject pointing to the subject-level SPM.mat file) from which the toolbox may extract the location of the functional volumes, design specification, as well as subject motion parameters.

Example:

```
>> clear BATCH;
>> BATCH.Setup.acquisitiontype=1;
>> conn_batch(BATCH);
```

BATCH.Setup.acquisitiontype

Description: Specifies the functional volumes acquisition type. This field defaults to a value of 1 (specifying continuous acquisition). Set to 0 if you want to specify sparse acquisition instead (this will skip the convolution with the hemodynamic response function when estimating the scans associated with each condition).

Example:

```
>> clear BATCH;
>> BATCH.Setup.acquisitiontype=1;
>> conn_batch(BATCH);
```

BATCH.Setup.analyses

Description: Vector of integers specifying the type of analyses that you wish to run on this dataset (1 for ROI-to-ROI analyses, 2 for seed-to-voxel analyses, and 3 for voxel-to-voxel analyses).

Example:

```
>> clear BATCH;
>> BATCH.Setup.analyses=[1,2];
>> conn_batch(BATCH);
```

BATCH.Setup.voxelmask

Description: Specifies the analysis mask type (for voxel-based analyses only, this limits analyses only to voxels within this mask). Set to 1 for using a fixed template mask (by default brainmask.nii or that specified by BATCH.Setup.voxelmaskfile), or 2 for using a subject-specific analysis mask (SPM implicit-mask procedure)

Example:

```
>> clear BATCH;
>> BATCH.Setup.voxelmask=1;
>> conn_batch(BATCH);
```

BATCH.Setup.voxelmaskfile

Description: Specifies the explicit mask filename (for voxel-based analyses only, when BATCH.Setup.voxelmask is set to 1).

Example:

```
>> clear BATCH;
>> BATCH.Setup.voxelmaskfile= fullfile(fileparts(which('spm')),'apriori','brainmask.nii');
>> conn_batch(BATCH);
```

BATCH.Setup.voxelresolution

Description: Specifies the space of voxel-level analyses (e.g. volume- or surface- based analyses). For volume-based analyses this option also specifies the resolution/registration/bounding-box of voxel-based analyses. Set to 1 for volume-based analyses using a fixed template resolution (by default 2mm isotropic voxels or same as BATCH.Setup.voxelmaskfile if specified); 2 for volume-based analyses using the same resolution/registration as the structural volumes; 3 for volume-based analyses using the same resolution/registration as the functional volumes; and 4 for surface-based analyses.

Example:

```
>> clear BATCH;
>> BATCH.Setup.voxelresolution=1;
>> conn_batch(BATCH);
```

BATCH.Setup.analysisunits

Description: Specifies the BOLD signal units: 1 for default percent-signal-change (PSC) units; 2 for raw units.

Example:
```
>> clear BATCH;
>> BATCH.Setup.analysisunits=1;
>> conn_batch(BATCH);
```

BATCH.Setup.surfacesmoothing

Description: Specifies the level of spatial smoothing in surface-based analyses (number of discrete diffusion steps)

Example:

```
>> clear BATCH;
>> BATCH.Setup.surfacesmoothing=16;
>> conn_batch(BATCH);
```

BATCH.Setup.outputfiles

Description: Specifies optional output files to be created during the analyses. Set outputfiles(1) to 1 for creating nifty volumes containing the beta estimates associated with each of the confounding effects during preprocessing (one volume per session/subject/confounding-effect). Set outputfiles(2) to 1 for creating nifty volumes containing the confound-corrected BOLD timeseries (one 4d volume per session/subject). Set outputfiles(3) to 1 for creating nifty volumes containing the Pearson correlation r-coefficients for each seed (for seed-to-voxel analyses that use correlation measures only; one volume per subject/condition/seed). Set outputfiles(4) to 1 for creating nifty volumes containing the Pearson correlation p-values for each seed (for seed-to-voxel analyses that use correlation measures only; one volume per subject/condition/seed). Set outputfiles(5) to 1 for creating nifty volumes containing the Pearson correlation FDR-corrected p-values for each seed (for seed-to-voxel analyses that use correlation measures only; one volume per subject/condition/seed). Set outputfiles(6) to 1 for creating REX files (see help REX) containing additional information of the ROI-extraction step (one file per each Subject/Session/ROI)

Example:

```
>> clear BATCH;
>> BATCH.Setup.outputfiles=[0,1,0,0,0,0];
>> conn_batch(BATCH);
```

BATCH.Setup.functionals

Description: Specifies the source functional volumes for connectivity analyses. This field contains a double cell ranging over each subject and session, where the *BATCH.Setup.functionals{nsub}{nses}* element contains a char array pointing to the source functional volume(s) for subject *nsub* and session *nses*.

Example: (specifies the functional volumes in the current experiment; two subjects, the first subject containing three scanning sessions, and the second subject only one session)

```
>> clear BATCH;
>> BATCH.Setup.functionals{1}{1}='mypath/subject1/session1/swrfoo4d.nii';
>> BATCH.Setup.functionals{1}{2}='mypath/subject1/session2/swrfoo4d.nii';
>> BATCH.Setup.functionals{1}{3}='mypath/subject1/session3/swrfoo4d.nii';
>> BATCH.Setup.functionals{2}{1}='mypath/subject2/session1/swrfoo4d.nii';
>> conn_batch(BATCH);
```

BATCH.Setup.structurals

Description: Specifies the source anatomical volumes for connectivity analyses. This field contains a cell array ranging over each subject, where the *BATCH.Setup.structural{nsub}* element contains a char array pointing to the source anatomical volume for subject *nsub.*

Example: (specifies the anatomical volumes for the current experiment -3 subjects-)

```
>> clear BATCH;
>> BATCH.Setup.structurals{1}='mypath/subject1/anatomical/foo.img';
>> BATCH.Setup.structurals{2}='mypath/subject2/anatomical/foo.img';
>> BATCH.Setup.structurals{3}='mypath/subject3/anatomical/foo.img';
>> conn_batch(BATCH);
```

BATCH.Setup.roiextract

Description: Specifies the source of functional data for ROI timeseries extraction (typically unsmoothed BOLD signal volumes); 1: same as 'functionals' field; 2: same as 'functionals' field after removing leading 's' from filename; 3: other (define rule programmatically in *roiextract_rule* field); 4: other (different set of functional volume files specified in *roiextract_functionals* field)

Example:

```
>> clear BATCH;
>> BATCH.Setup.roiextract=1;
>> conn_batch(BATCH);
```

BATCH.Setup. roiextract_rule

Description: Specifies the source of functional data for ROI timeseries extraction programmatically (when roiextract is set to 3).

roiextract_rule should be a cell array with three elements:
  roiextract_rule{1}: 1 for rules considering the file-name only; 2 for rules on full file-path
  roiextract_rule{2}: search expression
  roiextract_rule{3}: replace expression

The new filesnames used for ROI extraction will be defined as: *regexprep(filename, roiextract_rule{2}, roiextract_rule{3})*, where filename is the original functional file names (full paths or only file names depending on the roiextract_rule{1} value).

Example: (removes leading 's' from functional names)
>> clear BATCH;
>> BATCH.Setup.roiextract=3;
>> BATCH.Setup.roiextract_rule={1,'^s',''};
>> conn_batch(BATCH);

BATCH.Setup. roiextract_functional

Description: Specifies the source of functional data for ROI timeseries extraction explicitly (when roiextract is set to 4). *Setup.roiextract_functional* has the same structure as the *Setup.funtionals* field.

Example:
```
>> clear BATCH;
>> BATCH.Setup.roiextract=4;
>> BATCH.Setup.roiextract_functional{1}{1}='mypath/subject1/session1/swrfoo4d.nii';
>> BATCH.Setup.roiextract_functional{1}{2}='mypath/subject1/session2/swrfoo4d.nii';
>> BATCH.Setup.roiextract_functional{1}{3}='mypath/subject1/session3/swrfoo4d.nii';
>> BATCH.Setup.roiextract_functional{2}{1}='mypath/subject2/session1/swrfoo4d.nii';
>> conn_batch(BATCH);
```

BATCH.Setup.masks

Description: Specifies the Grey/White/CSF ROIs for each subject. If any of these fields are left unset the conn toolbox will perform segmentation on the anatomical volumes (for each subject) in order to define the missing masks.

BATCH.Setup.masks.Grey
BATCH.Setup.masks.White
BATCH.Setup.masks.CSF

Each of these three fields is characterized as a structure with sub-fields *files* and *dimensions,* characterizing the Grey matter, White matter, and CSF masks, respectively, for each subject.

BATCH.Setup.masks.[Grey/White/CSF].files

This field contains a cell array ranging over each subject, where the *BATCH.Setup.masks.Grey.files{nsub}* element contains a char array pointing to the grey matter mask for subject *nsub* (typically a c1*.img file obtained from segmentation of the anatomical volumes), the *BATCH.Setup.masks.White.files{nsub}* element contains a char array pointing to the white matter mask for subject *nsub* (typically a c2*.img file obtained from segmentation of the anatomical volumes), and the *BATCH.Setup.masks.CSF.files{nsub}* element contains a char array pointing to the CSF mask for subject *nsub* (typically a c3*.img file obtained from segmentation of the anatomical volumes).

BATCH.Setup.masks.[Grey/White/CSF].dimensions

This field contains a single number specifying the number of component dimensions to extract from each mask area. By default the toolbox will extract 1 component (the mean BOLD signal) from the Grey matter area, and 16 components (PCA decomposition) from the White and CSF areas.

Example: (specifies the Grey/White/CSF mask volumes for the current experiment -2 subjects-, and extracts 32 time-series components from each area)

```
>> clear BATCH;
>> BATCH.Setup.masks.Grey.files{1}='mypath/subject1/anatomical/c1foo.img';
>> BATCH.Setup.masks.Grey.files{2}='mypath/subject2/anatomical/c1foo.img';
>> BATCH.Setup.masks.Grey.dimensions=32;
>> BATCH.Setup.masks.White.files{1}='mypath/subject1/anatomical/c2foo.img';
>> BATCH.Setup.masks.White.files{2}='mypath/subject2/anatomical/c2foo.img';
>> BATCH.Setup.masks.White.dimensions=32;
>> BATCH.Setup.masks.CSF.files{1}='mypath/subject1/anatomical/c3foo.img';
>> BATCH.Setup.masks.CSF.files{2}='mypath/subject2/anatomical/c3foo.img';
>> BATCH.Setup.masks.CSF.dimensions=32;
>> conn_batch(BATCH);
```

BATCH.Setup.rois

Description: Specifies the ROI files for connectivity analyses. If this field is left unset the toolbox will by default use all the ROI files in the conn/rois directory.

BATCH.Setup.rois.names

This field contains a cell array ranging over each ROI, where the *BATCH.Setup.rois.names{nroi}* element contains a char array with the name of the ROI *nroi*. This value, if unspecified, defaults to the name of each ROI file (without path or extension information)

BATCH.Setup.rois.files

This field contains a cell array ranging over each ROI, where the *BATCH.Setup.rois.files{nroi}* element contains a char array pointing to the ROI file for the *nroi* ROI.
Alternatively, if subject-specific ROIs are being used, the *BATCH.Setup.rois.files{nroi}{nsub}* element contains a char array pointing to the subject-specific ROI file for the *nroi* ROI for subject *nsub*.

BATCH.Setup.rois.dimensions

This field contains a cell array ranging over each ROI, where the *BATCH.Setup.rois.dimensions{nroi}* element contains an integer describing the number of component time-series to be extracted from each ROI. This value, if unspecified, defaults to the value of 1 for every ROI (where only the mean timeseries is extracted from each ROI).

BATCH.Setup.rois.mask

Array where BATCH.Setup.rois.mask(nroi) specifies whether to mask (1) or not (0) each ROI with gray-matter voxels (defaults to 0)

BATCH.Setup.rois.regresscovariates

Array where BATCH.Setup.rois.regresscovariates(nroi) specifies whether to regress (1) or not (0) covariates before PCA decomposition of ROI BOLD timecourses (only applies when Setup.rois.dimensions>1) (defaults to 1)

BATCH.Setup.rois.roiextract

Array where BATCH.Setup.rois.unsmoothedvolumes(nroi) specifies, separately for each ROI, whether to use the ROI-specific BOLD timeseries specified in the *BATCH.Setup.roiextract* field (1) or use the original functional BOLD timeseries specified in the *BATCH.Setup.functionals* field (0) when extracting ROI BOLD timecourses. (defaults to 1)

Example: (specifies two subject-independent ROIs for the current experiment)

```
>> clear BATCH;
>> BATCH.Setup.rois.names={'ROI1','ROI2'};
>> BATCH.Setup.rois.dimensions={1,1};
>> BATCH.Setup.rois.files{1}='mypath/rois/roifile1.img';
>> BATCH.Setup.rois.files{2}='mypath/rois/roifile2.img';
>> conn_batch(BATCH);
```

BATCH.Setup.conditions

Description: Specifies the (taks or rest) conditions characterizing the block-design acquisition of the functional volumes. Note that if the *BATCH.New* initialization step has been run this field will default to defining a single condition encompassing the entire functional data. Alternatively if *BATCH.Setup.spmfiles* has been specified this field will be defined by the modeled conditions of interest in the corresponding SPM.mat files. Otherwise this field needs to be defined (e.g. use the example below for rest-state functional data)

BATCH.Setup.conditions.names

This field contains a cell array ranging over each condition, where the *BATCH.Setup.conditions.names{ncondition}* element contains a char array with the name of the condition *ncondition*.

BATCH.Setup.conditions.onsets

This field contains a triple cell array ranging over each condition, subject, and session, where the *BATCH.Setup.conditions.onsets{ncondition}{nsub}{nses}* element contains a vector with the condition onsets (in seconds) of the condition *ncondition,* for subject *nsub,* during session *nses*.

BATCH.Setup.conditions.durations

This field contains a triple cell array ranging over each condition, subject, and session, where the *BATCH.Setup.conditions.durations{ncondition}{nsub}{nses}* element contains a vector with the condition durations (in seconds) of the condition *ncondition,* for subject *nsub,* during session *nses*.

BATCH.Setup.conditions.param

Optional temporal weighting. This field contains an array where *BATCH.Setup.conditions.param(ncondition)* is either an index to a first-level covariate (see *BATCH.Setup.covariates*), or 0 for no temporal weighting.

BATCH.Setup.conditions.filter

Optional temporal filtering. This field contains a cell array where *BATCH.Setup.conditions.filter{ncondition}* is: *[]* for no filtering; *[a b]* (two-values vector) for band-pass filter between a and b (in Hz); or *n* (single scalar) for a filter bank with n filters.

Example: (specifies one rest-state condition –encompassing the entire functional data- for the current experiment –two subjects, one session each-)

```
>> clear BATCH;
>> BATCH.Setup.conditions.names={'rest'};
>> BATCH.Setup.conditions.onsets{1}{1}{1}=[0];
>> BATCH.Setup.conditions.onsets{1}{2}{1}=[0];
>> BATCH.Setup.conditions.durations{1}{1}{1}=[inf];
>> BATCH.Setup.conditions.durations{1}{2}{1}=[inf];
>> conn_batch(BATCH);
```

BATCH.Setup.covariates

Description: Specifies the first-level covariates (timeseries possibly associated with the BOLD signal). Note that if the *BATCH.New* initialization step has been run or if *BATCH.Setup.spmfiles* has been specified this field will default to a single covariate representing the estimated subject motion for each subject/session.

BATCH.Setup.covariates.names

This field contains a cell array ranging over each covariate, where the *BATCH.Setup.covariates.names{ncovariate}* element contains a char array with the name of the covariate *ncovariate*.

BATCH.Setup.covariates.files

This field contains a triple cell array ranging over each covariate, subject, and session, where the *BATCH.Setup.covariate.files{ncovariate}{nsub}{nses}* element contains a char array pointing to a .txt or .mat file defining the time-series first-level covariate *ncovariate*, for subject *nsub*, during session *nses*. Note that each covariate can in turn be multivariate (e.g. the 'movement' covariate contains 6 time-series, each representing one aspect of the estimated subject movement – translation and rotation parameters-). Valid first-level covariate files are .txt files with a row of numbers for each scan, and .mat files containing a single two-dimensional variable (arbitrarily named) with as many rows as scans.

Example: (specifies one realignment covariate for the current experiment –two subjects, one session each-)

```
>> clear BATCH;
>> BATCH.Setup.covariates.names={'motion'};
>> BATCH.Setup.covariates.onsets{1}{1}{1}='/mypath/subject1/sessions1/rp_foo.txt';
>> BATCH.Setup.covariates.onsets{1}{2}{1}='/mypath/subject2/sessions1/rp_foo.txt';
>> conn_batch(BATCH);
```

BATCH.Setup.subjects

Description: Specifies the second-level covariates (subject-level variables, such as behavioral or demographic measures as well as subject group definitions). By default a single second-level covariate is defined named ‘All’ characterizing a single group that includes all subjects. Note that second-level covariates can be redundant as different subsets of second-level covariates can be used in different second-level analyses (see *BATCH.Results* for more details)

BATCH.Setup.subjects.effect_names

This field contains a cell array ranging over each second-level covariate, where the *BATCH.Setup.subjects.effect_names{ncovariate}* element contains a char array with the name of the second-level covariate *ncovariate*.

BATCH.Setup.subjects.effects

This field contains a cell array ranging over each second-level covariate, where the *BATCH.Setup.subjects.effects{ncovariate}* element contains a column vector (with one element per subject) characterizing the second-level covariate *ncovariate*.

Example: (defines two groups and one behavioral measure for the current experiment -6 subjects-)

```
>> clear BATCH;
>> BATCH.Setup.subjects.effect_names={’group_1’,’group_2’,’behavioral’};
>> BATCH.Setup.subjects.effects{1}=[1;1;1;0;0;0];
>> BATCH.Setup.subjects.effects{2}=[0;0;0;1;1;1];
>> BATCH.Setup.subjects.effects{3}=[1.0;-.5;-.5;-1.2;0.4;0.8];
>> conn_batch(BATCH);
```

BATCH.Setup.done

Description: 1/0 variable specifying whether to run the initial data extraction steps (the steps performed in the gui when pressing ‘Done’ on the ‘Setup’ window). These steps include: a) segmentation of structural volumes if Grey/White/CSF masks have not been specified; and b) extraction of functional and ROI data (average or principal component decomposition of BOLD timeseries within each ROI).

Example: (runs the initial data extraction steps)

```
>> clear BATCH;
>> BATCH.Setup.done=1;
>> conn_batch(BATCH);
```

BATCH.Setup.overwrite

Description: ‘Yes’/’No’ variable specifying whether to overwrite existing files if they already exist when performing the data extraction steps. This field is only relevant when *BATCH.Setup.done* is set to 1. The default value is ‘Yes’. Set *overwrite* to ‘No’ if you are adding new subjects (or ROIs) to an already processed experiment, and want to perform the data extraction only for the new subjects (or ROIs).

Example: (runs the initial data extraction steps after new subjects have been added without overwriting the data already extracted for the original set of subjects)

```
>> clear BATCH;
>> BATCH.Setup.done=1;
>> BATCH.Setup.overwrite=’No’;
>> conn_batch(BATCH);
```

BATCH.Setup.isnew

Description: 1/0 variable specifying whether the conn_*.mat project specified in BATCH.filename is a new project. The default value is 0. Set *isnew* to 1 if you want to create a new conn_*.mat project (or you want to overwrite an existing project file)

Example: (creates a new conn_example.mat project)

```
>> clear BATCH;
>> BATCH.filename='conn_example.mat';
>> BATCH.Setup.isnew=1;
>> conn_batch(BATCH);
```

BATCH.Preprocessing

Description: Defines and run preprocessing steps (possible confounding effects and filtering).

Example: (defines a frequency filter for the current project)

```
>> clear BATCH;
>> BATCH.Preprocessing.filter=[.01,.1];
>> conn_batch(BATCH);
```

BATCH.Preprocessing.filter

Description: Two element vector defining the band-pass filter cutoff frequencies (in Hz). This value defaults to [0,inf] (inf specifies in this context the Nyquist cutoff frequency depending on your inter-scan acquisition time)

Example: (defines a frequency filter between 0.01Hz and 0.1Hz for the current project)

```
>> clear BATCH;
>> BATCH.Preprocessing.filter=[.01,.1];
>> conn_batch(BATCH);
```

BATCH.Preprocessing.detrending

Description: 1/0. Performs temporal detrending of the BOLD timeseries (separately for each session).

Example:

```
>> clear BATCH;
>> BATCH.Preprocessing.detrending=1;
>> conn_batch(BATCH);
```

BATCH.Preprocessing.despiking

Description: 1/0. Performs temporal despiking of the BOLD timeseries with a hyperbolic tangent squashing function.

Example:
>> clear BATCH;
>> BATCH.Preprocessing.despiking=1;
>> conn_batch(BATCH);

BATCH.Preprocessing.confounds

Description: Specifies confounding effects for connectivity analyses (these effects will be removed from the BOLD timeseries before computing connectivity measures). If this field is left unset the toolbox will by default use an aCompCor strategy (removing noise signals from White matter and CSF areas) together with realignment first-level covariates, and task-related effects.

BATCH.Preprocessing.confounds.names

This field contains a cell array ranging over each confounding effect, where the *BATCH.Preprocessing.confounds.names{neffect}* element contains a char array with the name of the confounding effect *neffect*. Valid confounding effect names can be: 'Grey Matter', 'White Matter', 'CSF', any ROI name (defined in *BATCH.Setup.rois.names*), any covariate name (defined in *BATCH.Setup.covariates.names*), or 'Effect of *' where * represents any condition name (defined in *BATCH.Setup.conditions.names*). Note that each of the confounding effects defined here can be multivariate (defined by multiple time-series, see below)

BATCH. Preprocessing.confounds.dimensions

This field contains a cell array ranging over each confounding effect, where the *BATCH.Preprocessing.confounds.dimensions{neffect}* element contains an integer characterizing the number of components (time-series) characterizing the confounding effect *neffect*. Valid dimension numbers range between 1 and the number of dimensions associated with the original effect (e.g. for White matter it will be limited to the number of dimensions specified in *BATCH.Setup.masks.White.dimensions,* for an arbitrary ROI it will be limited to the number of dimensions specified in *BATCH.Setup.rois.dimensions,* etc.)

BATCH. Preprocessing.confounds.deriv

This field contains a cell array ranging over each confounding effect, where the *BATCH.Preprocessing.confounds.deriv{neffect}* element contains an integer describing the order of additional time-derivatives to be used as additional confounding effects for the original effect *neffect*. This value defaults to the value of 0 for signals derived from mask areas, and 1 for first-level covariates or task-related effects (e.g. 6 motion parameters plus their temporal derivatives for subject motion parameters)
.

Example: (specifies five confounding effects to be removed from the BOLD signal before computing connectivity measures)

```
>> clear BATCH;
>> BATCH.Preprocessing.confounds.names={'White','CSF','ROI1','ROI2','realignment'};
>> BATCH.Preprocessing.confounds.dimensions={3,3,1,1,6};
>> BATCH.Preprocessing.confounds.deriv={0,0,0,0,1};
>> conn_batch(BATCH);
```

BATCH.Preprocessing.done

Description: 1/0 variable specifying whether to run the data preprocessing steps (the steps performed in the gui when pressing ‘Done’ on the ‘Preprocessing’ window). These steps include: a) removal of confounding effects from the BOLD signal at each voxel and for each ROI; and b) frequency filtering of the residual BOLD signal.

Example: (runs the initial preprocessing steps using all default values)

```
>> clear BATCH;
>> BATCH.Preprocessing.done=1;
>> conn_batch(BATCH);
```

BATCH.Preprocessing.overwrite

Description: ‘Yes’/’No’ variable specifying whether to overwrite existing files if they already exist when performing the preprocessing steps. This field is only relevant when *BATCH.Preprocessing.done* is set to 1. The default value is ‘Yes’. Set *overwrite* to ‘No’ if you are adding new subjects (or ROIs) to an already processed experiment, and want to perform the data preprocessing only for the new subjects (or ROIs).

Example: (runs the initial preprocessing steps after new subjects have been added without overwriting the data already preprocessed for the original set of subjects)

```
>> clear BATCH;
>> BATCH.Preprocessing.done=1;
>> BATCH.Preprocessing.overwrite=’No’;
>> conn_batch(BATCH);
```

BATCH.Analysis (for ROI-to-ROI or seed-to-voxel analyses)

Description: Defines and run first-level connectivity analyses (estimation of seed-to-voxel and ROI-to-ROI connectivity measures for each subject).

Example: (estimates default connectivity measures for MPFC and PCC sources)

```
>> clear BATCH;
>> BATCH.Analysis.sources={‘MPFC’,’PCC’};
>> conn_batch(BATCH);
```

BATCH.Analysis.analysis_number

Description: Index uniquely identifying a set of connectivity analyses. This value defaults to 1. Use sequential integer numbers to define multiple first-level connectivity analyses.

Example: (runs the second set of first-level analyses defined)

```
>> clear BATCH;
>> BATCH.Analysis.analysis_number=2;
>> BATCH.Analysis.done=1;
>> conn_batch(BATCH);
```

BATCH.Analysis.type

Description: Specifies the type of analyses to be performed . Use 1 for ROI-to-ROI analyse, 2 for seed-to-voxel analyses, or 3 for both ROI-to-ROI and seed-to-voxel analyses.

Example:

```
>> clear BATCH;
>> BATCH.Analysis.analysis_number=2;
>> BATCH.Analysis.type=1;
>> conn_batch(BATCH);
```

BATCH.Analysis.measure

Description: Integer identifying the connectivity measure used. Current valid values are:

1 – Correlation (bivariate) [default]
2 – Correlation (semipartial)
3 – Regression (bivariate)
4 – Regression (multivariate)

By default bivariate or Pearson correlation measures between two BOLD timeseries are used as connectivity measures. Note that (1) and (3) are bivariate measures, so the results for each source are independent on the number of sources used. In contrast, (2) and (4) are multivariate measures (estimating the unique contribution of each source) so the results for a given source will vary depending on the other sources included in a given first-level analysis.

Example: (defines semipartial-correlations as the connectivity measure of interest for the current experiment)

```
>> clear BATCH;
>> BATCH.Analysis.measure=2;
>> conn_batch(BATCH);
```

BATCH.Analysis.weight

Description: Integer identifying the type of intra-block weighting performed:

1 – none
2 – hrf [default]
3 – hanning

By default the scans within each block (or session if no task conditions are defined) is weighted using a hrf-shaped weighting function. This effectively weights down the contribution of the first few scans on the estimation of connectivity measures. Alternatively you can define no weighting (1) or a hanning weighting function (3) that weights more heavily only in the middle scans within each block.

Example: (defines hrf-shaped intra-block weighting for the current experiment)

```
>> clear BATCH;
>> BATCH.Analysis.weight=2;
>> conn_batch(BATCH);
```

BATCH.Analysis.sources

Description: Specifies ROI sources for connectivity analyses (connectivity measures will be estimated between these sources and every voxel –for seed-to-voxel analyses-, and between these sources and every ROI –for ROI-to-ROI analyses-). If this field is left unset the toolbox will by default use as sources all the ROIs that have not been defined as confounds.

BATCH.Analysis.sources.names

This field contains a cell array ranging over each source ROI, where the *BATCH.Analysis.sources.names{nsource}* element contains a char array with the name of the source *nsource*. Valid source names can be any ROI name (defined in *BATCH.Setup.rois.names*). Note that each of the sources defined here can be multivariate (defined by multiple time-series, see below)

BATCH. Analysis.sources.dimensions

This field contains a cell array ranging over each source ROI, where the *BATCH.Analysis.sources.dimensions{nsource}* element contains an integer characterizing the number of components (time-series) characterizing the source *nsource*. Valid dimension numbers range between 1 and the number of dimensions extracted for the original source (specified in *BATCH.Setup.rois.dimensions;* typically just one characterizing the average BOLD signal within this ROI)

BATCH. Analysis.sources.deriv

This field contains a cell array ranging over each source ROI, where the *BATCH.Analysis.sources.deriv{nsource}* element contains an integer describing the order of additional time-derivatives to be used as additional source time-series for source *nsource* (typically 0 to indicate the raw BOLD signal, and no time-derivatives used)

BATCH. Analysis.sources.fbands

This field contains a cell array ranging over each source ROI, where the *BATCH.Analysis.sources.fbands{nsource}* element contains an integer describing the number of frequency bands (partitioning the preprocessing band-pass filter) used to compute spectral decomposition of the functional connectivity measures (typically 1 to indicate no spectral decomposition)

Example: (specifies two ROI sources for the current experiment)

```
>> clear BATCH;
>> BATCH.Analysis.sources.names={’ROI3’,’ROI4’};
>> BATCH.Analysis.sources.dimensions={1,1};
>> BATCH.Analysis.sources.deriv={0,0};
>> conn_batch(BATCH);
```

BATCH.Analysis.done

Description: 1/0 variable specifying whether to run the first-level analysis steps (the steps performed in the gui when pressing ‘Done’ on the ‘Analyses’ window). These steps estimate seed-to-voxel and ROI-to-ROI connectivity measures for each subject and condition.

Example: (runs the first-level analyses using all default values)

```
>> clear BATCH;
>> BATCH.Analysis.done=1;
>> conn_batch(BATCH);
```

BATCH.Analysis.overwrite

Description: ‘Yes’/’No’ variable specifying whether to overwrite existing files if they already exist when performing the first-level analysis steps. This field is only relevant when *BATCH.Analysis.done* is set to 1. The default value is ‘Yes’. Set *overwrite* to ‘No’ if you are adding new subjects (or ROIs) to an already processed experiment, and want to estimate the first-level connectivity measures only for the new subjects (or ROIs).

Example: (runs the first-level analysis steps after new subjects have been added without overwriting the data already analyzed for the original set of subjects)

```
>> clear BATCH;
>> BATCH.Analysis.done=1;
>> BATCH.Analysis.overwrite=’No’;
>> conn_batch(BATCH);
```

BATCH.Analysis (for voxel-to-voxel analyses)

Description: Defines and run first-level voxel-to-voxel connectivity analyses.

Example: (estimates ILC measure for all subjects)

```
>> clear BATCH;
>> BATCH.Analysis.measures={‘ILC (Integrated Local Correlation)’};
>> conn_batch(BATCH);
```

BATCH.Analysis.analysis_number

Description: This field must be set to zero for voxel-to-voxel analyses (unless the *BATCH.Analysis.measures* field is defined, which would also uniquely identify this instance of *BATCH.Analysis* as voxel-to-voxel analysis as well).

Example: (computes all of the voxel-to-voxel measures defined)

```
>> clear BATCH;
>> BATCH.Analysis.analysis_number=0;
>> BATCH.Analysis.done=1;
>> conn_batch(BATCH);
```

BATCH.Analysis.measures

Description: Specifies voxel-to-voxel measures (measures derived from the matrix of voxel to voxel correlations). If this field is left unset the toolbox will use as measures all the default measures (type *conn_v2v measures* for a list of default measures).

$$\sum_{\mathbf{y}\in\Omega} k(\mathbf{y}-\mathbf{x})\cdot\mathbf{r}(\mathbf{x},\mathbf{y})$$

BATCH.Analysis.measures.names

This field contains a cell array ranging over each voxel-to-voxel measure, where the *BATCH.Analysis.measures.names{nmeasure}* element contains a char array with the name of the measure *nmeasure*. If a name matches (exactly) one of the default measure names, the fields below will be filled from the corresponding default measure definitions.

BATCH.Analysis.measures.type

BATCH.Analysis.measures.kernelsupport

BATCH.Analysis.measures.type.kernelshape

BATCH.Analysis.measures.type.dimensions

These fields contain each a cell array ranging over each measure. *type{nmeasure}* characterizes the type of connectivity matrix used when computing this measure (1 for using the matrix of voxel-to-voxel correlation values, 2 for using the voxel-to-voxel matrix of similarity values); *kernelsupport{nmeasure}* characterizes the size of the spatial convolution kernel (FWHM in mm); *kernelshape{nmeasure}* characterizes the shape of the convolution kernel (0 for a Gaussian filter, 1 for its gradient, or 2 for its laplacian); and *dimensions{nmeasure}* characterizes the number of SVD dimensions to consider from the voxel-to-voxel matrix of connectivity values (set to inf to skip dimensionality reduction).

Example: (specifies two ROI measures for the current experiment)

```
>> clear BATCH;
>> BATCH.Analysis.measures.names={'ILC'};
>> BATCH.Analysis.measures.type={1};
>> BATCH.Analysis.measures.kernelsupport={24};
>> BATCH.Analysis.measures.kernelshape={0};
>> BATCH.Analysis.measures.dimensions={16};
>> conn_batch(BATCH);
```

BATCH.Analysis.done

Description: 1/0 variable specifying whether to run the first-level analysis steps (the steps performed in the gui when pressing ‘Done’ on the ‘Analyses’ window). These steps estimate the defined voxel-to-voxel connectivity measures for each subject and condition.

Example: (computes all of the voxel-to-voxel measures defined)

```
>> clear BATCH;
>> BATCH.Analysis.analysis_number=0;
>> BATCH.Analysis.done=1;
>> conn_batch(BATCH);
```

BATCH.Analysis.overwrite

Description: ‘Yes’/’No’ variable specifying whether to overwrite existing files if they already exist when performing the first-level analysis steps. This field is only relevant when *BATCH.Analysis.done* is set to 1. The default value is ‘Yes’. Set *overwrite* to ‘No’ if you are adding new subjects (or ROIs) to an already processed experiment, and want to estimate the first-level connectivity measures only for the new subjects (or measures).

Example: (runs the first-level analysis steps after new subjects have been added without overwriting the data already analyzed for the original set of subjects)

```
>> clear BATCH;
>> BATCH.Analysis.analysis_number=0;
>> BATCH.Analysis.done=1;
>> BATCH.Analysis.overwrite=’No’;
>> conn_batch(BATCH);
```

BATCH.Results (for ROI-to-ROI or seed-to-voxel analyses)

Description: Defines and run second-level connectivity analyses (general linear model between-subjects analyses).

Example: (runs second-level paired t-tests comparing the connectivity with MPFC between task1 blocks vs. during task2 blocks)

```
>> clear BATCH;
>> BATCH.Results.between_subjects.effect_names={'All'};
>> BATCH.Results.between_subjects.contrast=[1];
>> BATCH.Results.between_conditions.effect_names={task1','task2'};
>> BATCH.Results.between_conditions.contrast=[1,-1];
>> BATCH.Results.between_sources.effect_names={'MPFC_1_1'};
>> BATCH.Results.between_sources.contrast=[1];
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.analysis_number

Description: Index uniquely identifying a set of connectivity analyses (the same as defined in *BATCH.Analysis.analysis_number*). This value defaults to 1. Use sequential integer numbers to define multiple first-level connectivity analyses.

Example: (runs second-level analyses on the the second set of first-level analyses defined)

```
>> clear BATCH;
>> BATCH.Results.analysis_number=2;
>> BATCH.Results.between_subjects.effect_names={'All'};
>> BATCH.Results.between_subjects.contrast=1;
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.foldername

Description: Local folder name where second-level results will be stored. A folder with this name will be created inside the /results/secondlevel/ folder. Within this folder a different folder will be created for each source ROI. This field defaults to a folder named defined by combining the analysis number, subject effects, and condition of interest of the second level analysis. For example, when performing a t-test comparing two groups named ‘A’ and ‘B’ during the ‘rest’ condition, the default *foldername* value will be:
  ANALYSIS_01.SUBJECT_EFFECTS_A(1)_B(-1).CONDITIONS_rest
Within this folder the second-level results will be stored in source-specific sub-folders (named after each source)

Example: (runs second-level analyses on the 2nd set of first-level analyses defined)

```
>> clear BATCH;
>> BATCH.Results.analysis_number=2;
>> BATCH.Results.between_subjects.effect_names={’All’};
>> BATCH.Results.between_subjects.contrast=1;
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.between_subjects

Description: Define the second-level between-subject model and contrast of interest. A general linear model will be fit using the subject effects specified in the effect_names field (see below) as regressors. The rest of subject effects defined in the *BATCH.Setup.subjects* field will be disregarded for this analysis.

BATCH.Results.between_subjects.effect_names

This field contains a cell array ranging over the subjects effects to be included in this second level analysis, where the *BATCH.Results.between_subjects.effect_names{neffect}* element contains a char array with the name of the subject effect *neffect*. Valid effect names can be any subject-effect name (as defined in *BATCH.Setup.subjects.effect_names*).

BATCH.Results.between_subjects.contrast

This field contains a vector ranging over the subjects effects defined above, where the *BATCH.Results.between_subjects.contrast(neffect)* element represents the contrast value on the effect *neffect.*

Example: (runs a second-level two-sample t-test comparing group A to group B)

```
>> clear BATCH;
>> BATCH.Results.between_subjects.effect_names={'Group_A','Group_B'};
>> BATCH.Results.between_subjects.contrast=[1,-1];
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.between_conditions

Description: Define the second-level between-conditions contrast of interest. If this field is left unspecified a separate second-level analysis will be performed separately for each condition defined in *BATCH.Setup.conditions*.

BATCH.Results.between_conditions.effect_names

This field contains a cell array ranging over the conditions to be included in this second level analysis, where the *BATCH.Results.between_conditions.effect_names{ncondition}* element contains a char array with the name of the condition *ncondition*. Valid condition names can be any condition name defined in *BATCH.Setup.conditions.names*.

BATCH.Results.between_conditions.contrast

This field contains a vector ranging over the conditions defined above, where the *BATCH.Results.between_conditions.contrast(ncondition)* element represents the contrast value on the condition *ncondition.*

Example: (runs a second-level paired t-test comparing condition A to condition B)

```
>> clear BATCH;
>> BATCH.Results.between_subjects.effect_names={'All'};
>> BATCH.Results.between_subjects.contrast=[1];
>> BATCH.Results.between_conditions.effect_names={'Condition A','Condition B'};
>> BATCH.Results.between_conditions.contrast=[1,-1];
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.between_sources

Description: Define the second-level between-sources contrast of interest. If this field is left unspecified a separate second-level analysis will be performed separately for each source defined in *BATCH.Analysis.sources*.

BATCH.Results.between_sources.effect_names

This field contains a cell array ranging over the sources to be included in this second level analysis, where the *BATCH.Results.between_sources.effect_names{nsource}* element contains a char array with the name of the source *nsource*. Valid source names can be any source name defined in *BATCH.Analysis.sources* and appended with *'_1_1'* (to specify the zero-order derivative of the first component) or appended with *'_N_M'* (to specify the N-1th order derivative of the Mth component, if multivariate sources have been defined).

BATCH.Results.between_sources.contrast

This field contains a vector ranging over the sources defined above, where the *BATCH.Results.between_sources.contrast(nsource)* element represents the contrast value on the condition *nsource.*

Example: (runs a second-level paired t-test comparing the connectivity with ROI1 to the connectivity with ROI2)

```
>> clear BATCH;
>> BATCH.Results.between_subjects.effect_names={'All'};
>> BATCH.Results.between_subjects.contrast=[1];
>> BATCH.Results.between_sources.effect_names={'ROI1_1_1','ROI2_1_1'};
>> BATCH.Results.between_sources.contrast=[1,-1];
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.done

Description: 1/0 variable specifying whether to run the specified second-level analysis.

Example: (runs second-level one-sample t-test including all subjects separately for each of the sources included in first-level analyses, and for each task- or rest- condition defined)

```
>> clear BATCH;
>> BATCH.Results.between_subjects.effect_names={'All'};
>> BATCH.Results.between_subjects.contrast=1;
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results (for voxel-to-voxel analyses)

Description: Defines and run second-level connectivity analyses (general linear model between-subjects analyses).

Example: (runs second-level paired t-tests comparing the ILC measure between task1 blocks vs. during task2 blocks)

```
>> clear BATCH;
>> BATCH.Results.between_subjects.effect_names={'All'};
>> BATCH.Results.between_subjects.contrast=[1];
>> BATCH.Results.between_conditions.effect_names={task1','task2'};
>> BATCH.Results.between_conditions.contrast=[1,-1];
>> BATCH.Results.between_measures.effect_names={' ILC (Integrated Local Correlation)'} ;
>> BATCH.Results.between_measures.contrast=[1];
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.analysis_number

Description: This field must be set to zero for voxel-to-voxel analyses (unless the *BATCH.Results.between_measures* field is defined, which would also uniquely identify this instance of *BATCH.Results* as voxel-to-voxel analysis as well).

Example: (runs second-level analyses on all of the voxel-to-voxel first-level analyses defined)

```
>> clear BATCH;
>> BATCH.Results.analysis_number=0;
>> BATCH.Results.between_subjects.effect_names={’All’};
>> BATCH.Results.between_subjects.contrast=1;
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.foldername

Description: Local folder name where second-level results will be stored. A folder with this name will be created inside the /results/secondlevel/ folder. Within this folder a different folder will be created for each voxel-to-voxel measure. This field defaults to a folder named defined by combining the analysis number, subject effects, and condition of interest of the second level analysis. For example, when performing a t-test comparing two groups named ‘A’ and ‘B’ during the ‘rest’ condition, the default *foldername* value will be:
ANALYSIS_01.SUBJECT_EFFECTS_A(1)_B(-1).CONDITIONS_rest
Within this folder the second-level results will be stored in measure-specific sub-folders (named after each voxel-to-voxel measure)

BATCH.Results.between_subjects

Description: Define the second-level between-subject model and contrast of interest. A general linear model will be fit using the subject effects specified in the effect_names field (see below) as regressors. The rest of subject effects defined in the *BATCH.Setup.subjects* field will be disregarded for this analysis.

BATCH.Results.between_subjects.effect_names

This field contains a cell array ranging over the subjects effects to be included in this second level analysis, where the *BATCH.Results.between_subjects.effect_names{neffect}* element contains a char array with the name of the subject effect *neffect*. Valid effect names can be any subject-effect name (as defined in *BATCH.Setup.subjects.effect_names*).

BATCH.Results.between_subjects.contrast

This field contains a vector ranging over the subjects effects defined above, where the *BATCH.Results.between_subjects.contrast(neffect)* element represents the contrast value on the effect *neffect.*

Example: (runs a second-level two-sample t-test comparing group A to group B)

```
>> clear BATCH;
>> BATCH.Results.analysis_number=0;
>> BATCH.Results.between_subjects.effect_names={'Group_A','Group_B'};
>> BATCH.Results.between_subjects.contrast=[1,-1];
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.between_conditions

Description: Define the second-level between-conditions contrast of interest. If this field is left unspecified a separate second-level analysis will be performed separately for each condition defined in *BATCH.Setup.conditions*.

BATCH.Results.between_conditions.effect_names

This field contains a cell array ranging over the conditions to be included in this second level analysis, where the *BATCH.Results.between_conditions.effect_names{ncondition}* element contains a char array with the name of the condition *ncondition*. Valid condition names can be any condition name defined in *BATCH.Setup.conditions.names*.

BATCH.Results.between_conditions.contrast

This field contains a vector ranging over the conditions defined above, where the *BATCH.Results.between_conditions.contrast(ncondition)* element represents the contrast value on the condition *ncondition.*

Example: (runs a second-level paired t-test comparing condition A to condition B)

```
>> clear BATCH;
>> BATCH.Results.analysis_number=0;
>> BATCH.Results.between_subjects.effect_names={'All'};
>> BATCH.Results.between_subjects.contrast=[1];
>> BATCH.Results.between_conditions.effect_names={'Condition A','Condition B'};
>> BATCH.Results.between_conditions.contrast=[1,-1];
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.between_measures

Description: Define the second-level between-measures contrast of interest. If this field is left unspecified a separate second-level analysis will be performed separately for each measure defined in *BATCH.Analysis.measures*.

BATCH.Results.between_measures.effect_names

This field contains a cell array ranging over the measures to be included in this second level analysis, where the *BATCH.Results.between_measures.effect_names{nmeasure}* element contains a char array with the name of the measure *nmeasure*. Valid measure names can be any measure name defined in *BATCH.Analysis.measures*.

BATCH.Results.between_measures.contrast

This field contains a vector ranging over the measures defined above, where the *BATCH.Results.between_measures.contrast(nmeasure)* element represents the contrast value on the condition *nmeasure.*

Example: (runs a second-level paired t-test comparing two ILC measures with different kernel sizes)

```
>> clear BATCH;
>> BATCH.Results.between_subjects.effect_names={’All’};
>> BATCH.Results.between_subjects.contrast=[1];
>> BATCH.Results.between_measures.effect_names={’ILC_a’,’ILC_b’};
>> BATCH.Results.between_measures.contrast=[1,-1];
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

BATCH.Results.done

Description: 1/0 variable specifying whether to run the specified second-level analysis.

Example: (runs second-level one-sample t-test including all subjects separately for each of the measures included in first-level analyses, and for each task- or rest- condition defined)

```
>> clear BATCH;
>> BATCH.Results.analysis_number=0;
>> BATCH.Results.between_subjects.effect_names={'All'};
>> BATCH.Results.between_subjects.contrast=1;
>> BATCH.Results.done=1;
>> conn_batch(BATCH);
```

One batch example script explained

This batch file (*conn_batchexample_nyu.m*) is included in the conn toolbox and it performs standard functional connectivity analyses on a publicly available dataset (NYU_CSC_TestRetest dataset, available at http://www.nitrc.org/projects/nyu_trt/ , see additional information below). This dataset includes 3 rest-state runs obtained during two different scanning sessions for 25 subjects. The batch steps in this example perform: a) complete spatial preprocessing steps; b) aCompCor confound removal (noise signals from White and CSF areas) plus removal of realignment and session effects; c) first-level and second-level analyses for each of 88 sources (Brodmann areas + 4 Fox ROIs commonly used in the resting-state literature).

The batch script is listed below with additional clarifications to help interpret it.

```
%
% batch process the NYU_CSC_TestRetest dataset (published in Shehzad et al., 2009,
The Resting Brain: Unconstrained yet Reliable. Cerebral Cortex.
doi:10.1093/cercor/bhn256)
%
% Steps:
% 1. Download the dataset from: http://www.nitrc.org/projects/nyu_trt/ (6
NYU_TRT_session*.tar.gz files)
% 2. Run this script. The script will:
%       a) Decompress the NYU_TRT_session*.tar.gz files into NYU_TRT_session*
folders
%          Decompress the *.nii.gz files
%       b) Spatial preprocessing of the anatomical and functional volumes
%         (normalization & segmentation of anatomical volumes; realignment,
%         coregistration, normalization, and smooting of the functional
%         volumes)
%       c) Estimate first-level seed-to-voxel connectivity maps for each of
%         the default seeds (located in the conn/rois folder), separately
%         for each subject and for of the three test-retest sessions.
%         Estimate second-level seed-to-voxel results for each of the default
%         seeds, collapsed across the three sessions.
%
%
%
```

The following section locates the original *.tar.gz files in the distribution and decompresses them.

```
%% UNTAR .tar.gz files
a=dir('NYU_TRT_session*.tar.gz');
for n1=1:length(a),

[a_path,a_name,a_ext]=fileparts(a(n1).name);[nill,a_name2,a_ext2]=fileparts(a_name
);
    dirname=fullfile(a_path,a_name2);
    if ~isdir(dirname),
        disp(['extracting contents from file ',a(n1).name]);
        untar(a(n1).name,dirname);
    end
end
```

The following section decompresses the resulting *nii.gz files. Note: the command *conn_dir* is very similar to the matlab command *dir* but it performs the directory listing recursively.

```
%% UNZIP .nii.gz files
a=strvcat(conn_dir('lfo.nii.gz'),conn_dir('mprage_skullstripped.nii.gz'));
```

```
for n1=1:size(a,1),
    [a_path,a_name,a_ext]=fileparts(a(n1,:));
    if isempty(dir(fullfile(a_path,a_name))),
        disp(['unzipping file ',a(n1,:)]);
        gunzip(deblank(a(n1,:)));
    end
end
```

This section locates the functional and anatomical volumes. Functional volumes are always named *lfo.nii* and they are stored in session-specific and subject-specific folders, anatomical volumes are named *mprage_skullstripped.nii* and they are included in each session-specific and subject-specific folders (here we are keeping only the anatomical volumes defined for the first session). The script below is adapted in case someone decides to download only a subset of the three available sessions (they can be downloaded separately).

```
%% FIND functional/structural files
cwd=pwd;
FUNCTIONAL_FILE=cellstr(conn_dir('lfo.nii'));
STRUCTURAL_FILE=cellstr(conn_dir('mprage_skullstripped.nii'));
nsubjects=25;
if rem(length(FUNCTIONAL_FILE),nsubjects),error('mismatch number of functional
files');end
if rem(length(STRUCTURAL_FILE),nsubjects),error('mismatch number of anatomical
files');end
nsessions=length(FUNCTIONAL_FILE)/nsubjects;
FUNCTIONAL_FILE=reshape(FUNCTIONAL_FILE,[nsubjects,nsessions]);
STRUCTURAL_FILE={STRUCTURAL_FILE{1:nsubjects}};
disp([num2str(size(FUNCTIONAL_FILE,1)),' subjects']);
disp([num2str(size(FUNCTIONAL_FILE,2)),' sessions']);
TR=2; % Repetition time = 2 seconds
```

Here goes the BATCH definition. The entire experiment is defined with a single BATCH structure.

```
%% PREPARES connectivity analyses (using default values for all parameters, see
help conn_batch to define non-default values)
clear batch;
% CONN New experiment
(realignment/coregistration/segmentation/normalization/smoothing)
```

This points to the conn_NYU.mat project file (to be created in the current folder). This project is going to be created by the BATCH.New step (so there is no need to use *isnew* in the BATCH.*Setup* step)

```
batch.filename=fullfile(cwd,'conn_NYU.mat');            % New conn_*.mat
experiment name
```

These lines define the BATCH.New process. We are leaving the *BATCH.New.steps* unspecified so that all of the spatial preprocessing steps are included. We are simply defining the *functionals* and *structurals* fields to make them point to the appropriate files (one functional .nii file per subject/session, and one anatomical .nii files per subject).

```
batch.New.FWHM=8;                                       % 8mm FWHM smoothing
batch.New.VOX=2;                                        % 2mm voxel size for
analyses
batch.New.functionals=repmat({{}},[nsubjects,1]);       % Point to functional
volumes for each subject/session
for nsub=1:nsubjects,for
nses=1:nsessions,batch.New.functionals{nsub}{nses}{1}=FUNCTIONAL_FILE{nsub,nses};e
nd; end %note: each subject's data is defined by three sessions and one single
(4d) file per session
batch.New.structurals=STRUCTURAL_FILE;                  % Point to anatomical
volumes for each subject
```

This defines the *BATCH.Setup* step. Since the *BATCH.New* step has already defined the first-level covariates (realignment parameters), and subject effects (all subjects treated as a single group), we only need here to re-define the conditions of interest. Note that the *BATCH.New* step in reality has already defined one single condition encompassing all sessions. If we did not care about between-session differences in connectivity, and given that this is a resting-state design, we would not need to change the default condition definitions. Yet this particular dataset was created specifically to look at the robustness of connectivity analyses across conditions so we decided that we might want to look at session-specific effects here. To that end we create three conditions (named 'Session1', 'Session2', and 'Session3'). The *onsets* and *durations* fields for the condition 'Session1' are set to *0* and *inf*, respectively, during the first scanning session, and these fields are left empty (set to *[]*) during the second and third scanning sessions to indicate that the condition 'Session1' does not occur during the second and third sessions. The same logic applies to the definition of the 'Session2' and 'Session3' session-specific conditions. Last, regarding the ROIs, we leave the *BATCH.Setup.rois* field unspecified here so that all of the ROIs in the conn/rois/ folder will be included in the analyses. Note that we set the field *done* to 1 to indicate that we do want to perform all the *Setup* (data extraction) steps.

```
% CONN Setup                                            % Default options (uses
all ROIs in conn/rois/ directory); see conn_batch for additional options
batch.Setup.RT=TR;                                      % TR (seconds)
nconditions=nsessions;                                  % treats each session as a
different condition (comment the following three lines and lines 84-86 below if
you do not wish to analyze between-session differences)
batch.Setup.conditions.names=cellstr([repmat('Session',[nconditions,1]),num2str((1
:nconditions)')]);
for ncond=1:nconditions,for nsub=1:nsubjects,for nses=1:nsessions,
batch.Setup.conditions.onsets{ncond}{nsub}{nses}=[];batch.Setup.conditions.duratio
ns{ncond}{nsub}{nses}=[]; end;end;end
for ncond=1:nconditions,for nsub=1:nsubjects,for nses=ncond,
batch.Setup.conditions.onsets{ncond}{nsub}{nses}=0;
batch.Setup.conditions.durations{ncond}{nsub}{nses}=inf;end;end;end
batch.Setup.overwrite='Yes';
batch.Setup.done=1;
```

These lines define the *BATCH.Preprocessing* step. We want to perform the default analyses (removal of aCompCor, realignment, and task- or session-related effects) so we leave the BATCH.Preprocessing.confounds field unspecified here and only define the *filter* field to the 0.01Hz<f<0.1Hz window of interest.

```
% CONN Preprocessing                                    % Default options (uses
White Matter+CSF+realignment+conditions as confound regressors); see conn_batch
for additional options
batch.Preprocessing.filter=[0.01, 0.1];                 % frequency filter (band-
pass values, in Hz)
batch.Preprocessing.done=1;
batch.Preprocessing.overwrite='Yes';
```

Again the first-level analyses are standard so the *BATCH.Analysis* step can use all the default values (specifically we are leaving the *BATCH.Analysis.sources* field unspecified so that all of the ROIs –not defined as confounds- are used as possible sources; we are also interested in obtaining bivariate correlation measures and apply the standard *hrf* weighting so there is no need to specify these fields)

```
% CONN Analysis                                         % Default options (uses
all ROIs in conn/rois/ as connectivity sources); see conn_batch for additional
options
batch.Analysis.done=1;
batch.Analysis.overwrite='Yes';
```

We want to leave most second-level analyses to the user to define and view through the gui but just as a starting point we are performing here a second-level one-sample t-tests across the entire group of 25 subjects and across the three conditions of interest. To that end the *between_conditions* contrast is defined as [1/3,1/3,1/3] (and pointing to the three session-specific conditions defined), the *between_subjects* contrast as [1] (pointing to the only subject-level effect defined, identifying the entire group of subjects), and the *between_sources* field is left unspecified so that the analyses will be performed separately for each source ROI analyzed.

```
% CONN Results                                        % Default options (compute
second-level results for all sources); see conn_batch for additional options
batch.Results.between_subjects.effect_names={'All'};    % Second-level analyses
for all subjects
batch.Results.between_subjects.contrast=[1];
batch.Results.between_conditions.effect_names=...       % All sessions, average
effect across sessions
    cellstr([repmat('Session',[nconditions,1]),num2str((1:nconditions)')]);
batch.Results.between_conditions.contrast=ones(1,nconditions)/nconditions;
batch.Results.done=1;
batch.Results.overwrite='Yes';
```

After all these definitions it is time to run the *conn_batch* command, which will performed the definitions and analysis steps sequentially

```
%% RUNS analyses
conn_batch(batch);
```

After all the analyses are done (which will take several hours) the following lines will launch the conn GUI, then load the CONN_NYU.mat file, and switch the gui to the *results* tab.

```
%% CONN Display
% launches conn gui to explore results
conn
conn('load',fullfile(cwd,'conn_NYU.mat'));
conn gui_results
% Use conn_display to display any of the already computed whole-brain second-level
results
```