Chapter 1

Batch interface

Details about the algorithms used for data processing are given in the other sections of this manual. This section explains how a sequence of processing steps can be run at once without MATLAB programming. SPM8 includes `matlabbatch`[**?**] which has been derived from the SPM5 batch system, but is also available as a separate package.

In `matlabbatch`, each data processing step is called "module". There are e.g. modules for spatial processing of MRI data (realignment, normalisation, smoothing), statistics (fMRI or factorial design specification, model estimation, contrast specification). A batch describes which modules should be run on what kind of data and how these modules depend on each other.

Compared to running each processing step interactively, batches have a number of advantages:

Documentation Each batch can be saved as a MATLAB script. All parameters (including default settings) are included in this script. Thus, a saved batch contains a full description of the sequence of processing steps and the parameter settings used.

Reproducibility Batches can be saved, even if not all parameters have been set. For a multi-subject study, this allows to create template batches. These templates contain all settings which do not vary across subjects. For each subject, they can be loaded and only subject-specific parts need to be completed.

Unattended execution Instead of waiting for a processing step to complete before entering the results in the next one, all processing steps can be run in the specified order without any user interaction.

Multiple batches Multiple batches can be loaded and executed together.

1.1 Batch tutorial

In this tutorial we will develop a batch for spatial processing and fMRI statistics of a single subject of the "Face" example dataset (see chapter **??**). To follow this tutorial, it is not necessary to download the example dataset, except for the last step (entering subject dependent data).

To create a batch which can be re-used for multiple subjects in this study, it is necessary to collect/define

- study specific input data (e.g. MRI measurement parameters, time constants of the functional experiment, number of sessions),
- necessary processing steps,
- data flow between processing steps.

Subject specific input data (original functional and structural MRI data, subject specific experiment parameters) should be entered after the batch template has been saved.

1.1.1 Study specific input data

This dataset consists of fMRI data acquired in a single session and a structural MRI. See section 1.2 to learn how to deal efficiently with multi-session data. MRI parameters and experiment details are described in chapter **??**.

1.1.2 Necessary processing steps

Helper modules

Some SPM modules produce graphics output which is captured in a PostScript file in the current working directory. Also, a new directory needs to be created for statistics. The "BasicIO" menu provides a collection of modules which are useful to organise a batch. We will need the following modules:

- Named directory selector
- Change directory
- Make directory

SPM processing

For a classical SPM analysis, the following processing steps are necessary:

- Realignment
- Slice timing correction
- Coregistration
- Segmentation
- Normalisation
- Smoothing
- fMRI design
- Model estimation
- Contrasts
- Results report

1.1.3 Add modules to the batch

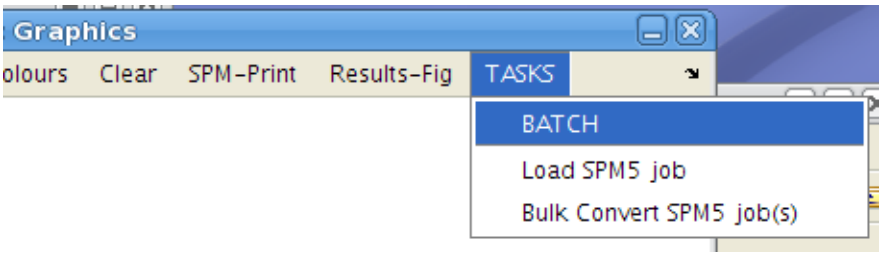

Figure 1.1: TASKS menu in "Graphics" window

The helper modules and the SPM processing modules can be assembled using the GUI. Locate the Graphics window and open the batch editor by selecting "BATCH" from the "TASKS" menu (fig. 1.1). First, add the helper modules, followed by the SPM modules in the order listed above. Do not configure any details until you have selected all modules.

1.1.4 Configure subject-independent data

Now, go through the batch and configure all settings that are subject-independent (e.g. the name of the analysis directory, slice timing parameters) as described in chapter **??**. Do not enter any data that is specific for a certain subject. The values that need to be entered are not repeated here, please refer to the corresponding sections in chapter **??**.

The file `man/batch/face_template_single_subject_nodeps.m` contains the batch after all modules have been added and subject-independent data has been entered.

Named Directory Selector

Input Name Give this selection a name (e.g. "Subject directory") - this name will be shown in the dependency list of this batch.

Directories Add a new directory selector, but do not enter a directory itself.

Change Directory

Nothing to enter now.

Make Directory

New Directory Name "categorical" - the name of the analysis directory. This directory will be created at batch run-time in the subject directory.

Realign: Estimate & Reslice

Data Add a new "Session" item. Do not enter any files for this session now.

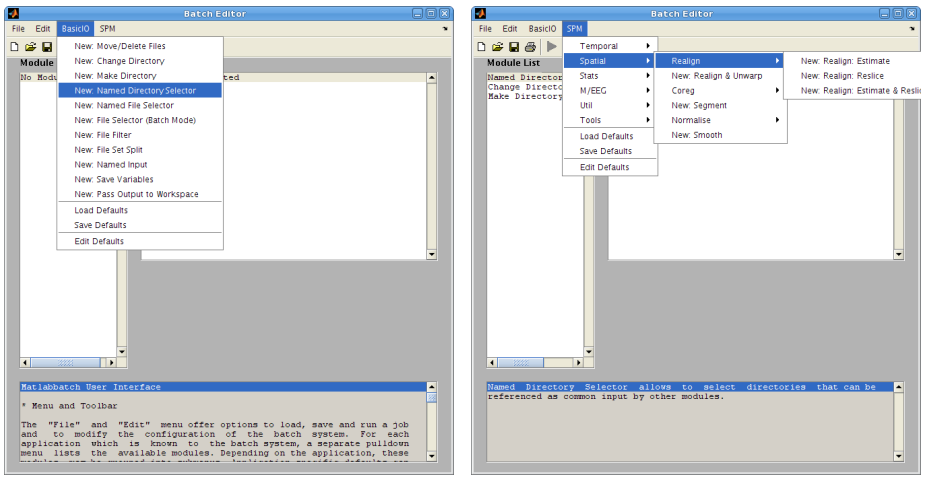

Figure 1.2: BasicIO and SPM application menus

Slice Timing

Data Add a new "Session" item. Do not enter any files for this session now.

Timing options Enter data for "Number of slices", "TR", "TA", "Slice order", "Reference slice".

Coreg: Estimate

Nothing to enter now.

Segment

Nothing to enter now.

Normalise: Write

Data Add a new "Subject". Do not enter any files now.

Writing Options Adjust bounding box, voxel sizes, interpolation

Smooth

FWHM Enter FWHM

fMRI model specification

Enter all data which is constant across subjects.

Timing parameters Enter values for "Units for design", "Interscan interval", "Microtime resolution", "Microtime onset"

Data & Design Add a new "Session" item. Do not enter scans, conditions or regressors yet. They will be added as dependencies or subject specific inputs. If you want to make sure to remember this, you can highlight

"Multiple conditions" and select "Clear Value" from the "Edit" menu. Do the same for "Multiple regressors". This will mark both items with an `<-X`, indicating that something must be entered there.

Factorial design Enter the specification for both factors.

Basis functions Select the basis function and options you want to use.

Model estimation

Nothing to be entered yet for classical estimation.

Contrast manager

If you have selected the "Factorial design" option as described above, SPM will automatically create some contrasts for you. Here, you can create additional T- or F-contrasts. As an example, we will add an "Effects of interest" F-contrast.

Contrast session Add a new "F-contrast" item.

Name Enter a name for this contrast, e.g. "Effects of interest".

Contrast vectors Add a new "Contrast vector" item. F-contrasts can have multiple rows. You can either enter a contrast matrix in an "F contrast vector" entry, or enter them row by row. To test for the effects of interest (1 basis function and 2 derivatives for each of the four conditions) enter `eye(12)` as F contrast vector.

Replicate over sessions This design does not have multiple sessions, so it is safe to say "No" here.

Results report

Reviewing individual results for a large number of subjects can be very time consuming. Results report will print results from selected contrasts to a PostScript file.

Contrast(s) Enter `Inf` to print a report for each of the defined contrasts.

1.1.5 Data flow

In chapter **??**, each processing step was performed on its own. In most cases, output data was simply passed on from one module to the next. This scheme is illustrated in figure 1.3. Only the coloured items at the top of the flow chart are subject specific and need to be entered in the final batch. All arrow connections are subject-independent and can be specified in the batch template.

Add dependencies

Based on the data flow in figure 1.3, modules in the batch can now be connected.
The batch containing all dependencies can be found in `man/batch/face_template_single_subject.m`.

Again, start editing at the top of the batch:

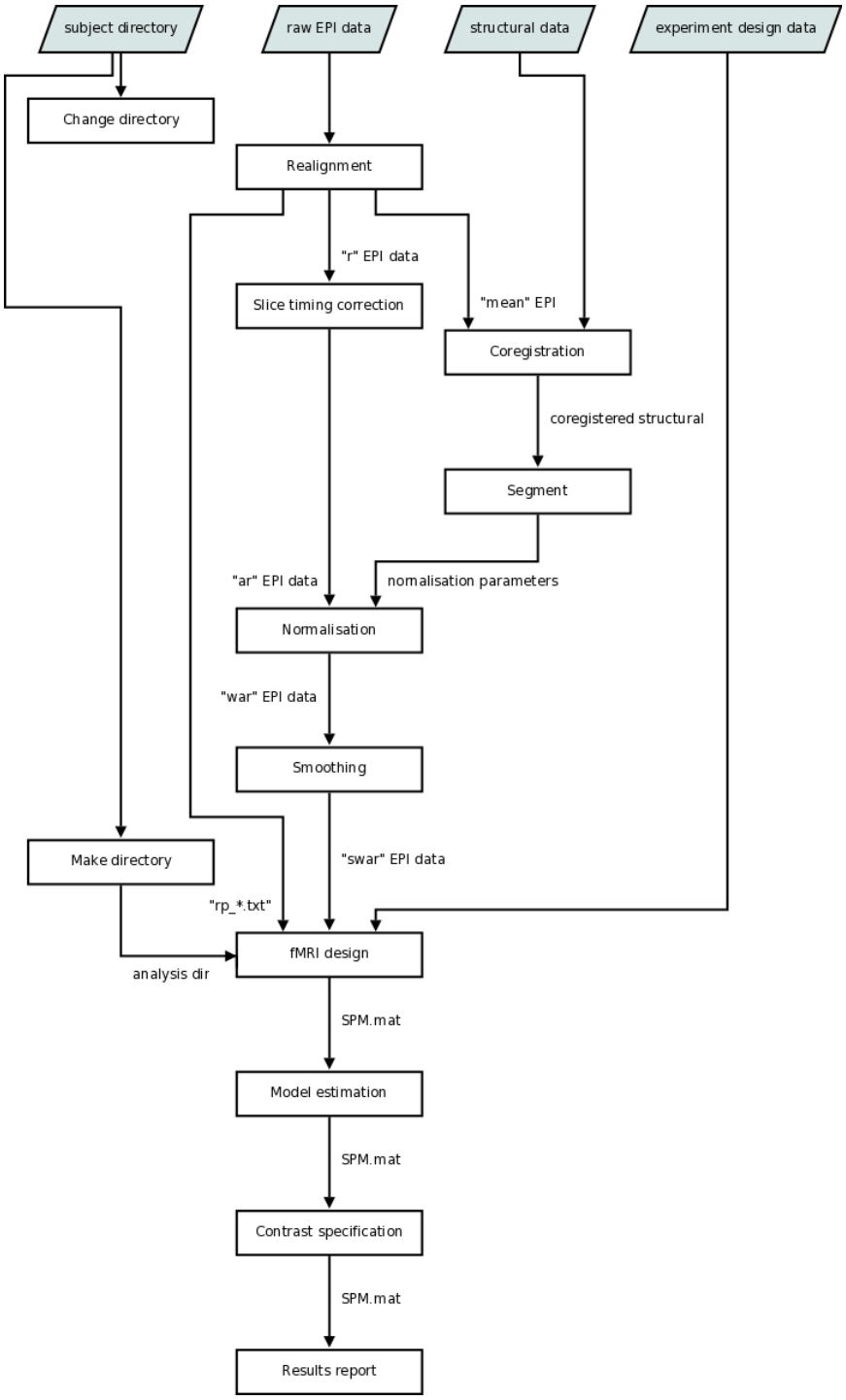

Figure 1.3: Flow chart for batch

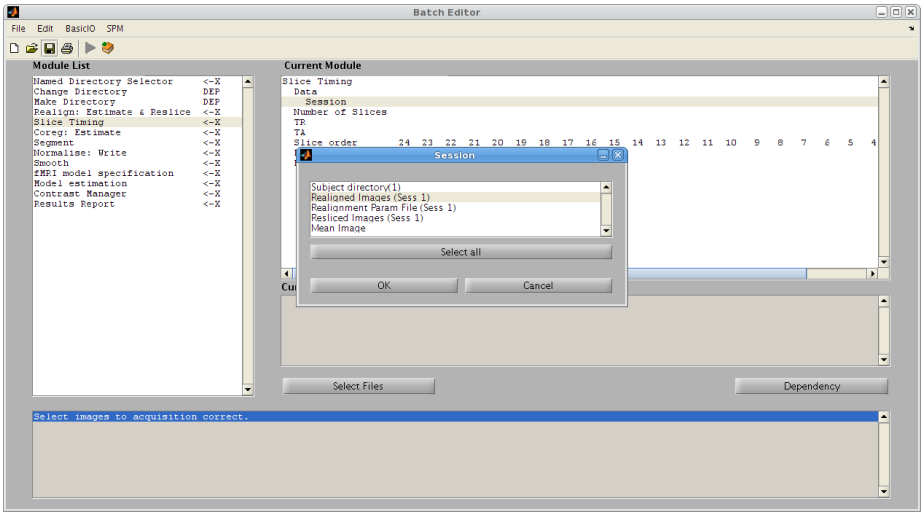

Figure 1.4: Dependency selection

Named Directory Selector

Nothing to enter now.

Change Directory

Directory Press "Dependency" and select "Subject directory(1)". At run time, SPM will change to this directory before batch processing continues.

Make Directory

Parent Directory Press "Dependency" and select "Subject directory(1)". The "categorial" directory will be created in this directory.

Realign: Estimate & Reslice

Nothing to enter now.

Slice Timing

Session Press "Dependency" and select "Resliced Images (Sess 1)".

Coreg: Estimate

Reference Image Press "Dependency" and select "Mean Image".

Segment

Data Press "Dependency" and select "Coregistered Images". At run time, this will resolve to the coregistered anatomical image.

Normalise: Write

Parameter File Press "Dependency" and select "Norm Params File Subj→MNI (Subj 1)".

Images to Write Press "Dependency" and select "Slice Timing Corr. Images (Sess 1)".

Smooth

Images to Smooth Press "Dependency" and select "Normalised Images (Subj 1)"

fMRI model specification

Directory Press "Dependency" and select "Make Directory 'categorical' "

Scans Press "Dependency" and select "Smoothed Images". Note: this works because there is only one session in our experiment. In a multisession experiments, images from each session may be normalised and smoothed using the same parameters, but the smoothed images need to be split into sessions again. See section 1.2 how this can be done.

Multiple regressors Press "Dependency" and select "Realignment Param File (Sess 1)".

Model estimation

Select SPM.mat Press "Dependency" and select "SPM.mat File (fMRI Design&Data)".

Contrast manager

Select SPM.mat Press "Dependency" and select "SPM.mat File (Estimation)".

Results report

Select SPM.mat Press "Dependency" and select "SPM.mat File (Contrasts)".

1.1.6 Entering subject-specific data

Now, only 4 modules should have open inputs left (marked with `<-X`). These inputs correspond to data which vary over the subjects in your study:

Named Directory Selector Subject directory

Realign: Estimate & Reslice Raw EPI data for the fMRT session

Coreg: Estimate Anatomical image to be coregistered to mean EPI

fMRI model specification Names, conditions and onsets of your experimental conditions, specified in a multiple conditions .mat file.

Using the GUI, you can now perform these steps for each subject:

1. load the template batch
2. enter subject-specific data
3. save batch under a subject specific name.

After that, all batches for all subjects can be loaded and run at once.

This process can be automated using some basic MATLAB scripting. See section 1.2.2 for details.

1.2 Advanced features

1.2.1 Multiple sessions

If an fMRI experiment has multiple sessions, some processing steps need to take this into account (slice timing correction, realignment, fMRI design), while others can work on all sessions at once (normalisation, smoothing).

Two modules in BasicIO help to solve this problem:

Named File Selector Files can be entered here session by session. Note that this file selector selects all files (not restricted to images) by default. To select only images, set the filter string to something like `.*nii$` or `.*img$`.

File Set Split This module splits a list of files based on an index vector. Named file selector provides such an index vector to split the concatenation of all selected images into individual sessions again.

1.2.2 Command line interface

Complete and run a pre-specified job

`cfg_serial(guifcn, job[, input1, input2 ...])`

This interface is called the "serial" interface. It takes a job, and asks for the input to any open configuration items one after another. If a list of inputs is supplied, these will be filled in (if they are appropriate). After all inputs are filled, the job will be run. The first argument, `guifcn` is a function handle that will be called to run a GUI if there are insufficient inputs or inputs that do not match a configuration item specification. For unattended mode, this may be left empty.

The following MATLAB code snippet can be used to fill in and run the `face_template_single_subject.m` batch. Instead of using `cfg_getfile` as file selector, files could be read from e.g. a MATLAB variable:

```
% Collect missing inputs
subjdir = cellstr(cfg_getfile([1 1],'dir','Subject Dir'));
subjepi = cellstr(cfg_getfile([1 inf],'image','Raw EPI images'));
subjana = cellstr(cfg_getfile([1 1],'image','Anatomy image'));
subjcon = cellstr(cfg_getfile([1 1],'mat','Multiple conditions'));
% Run batch, assuming face_template_single_subject.m is in your
% MATLAB path or working directory
cfg_serial([], 'face_template_single_subject', ...
           subjdir, subjepi, subjana, subjcon);
```

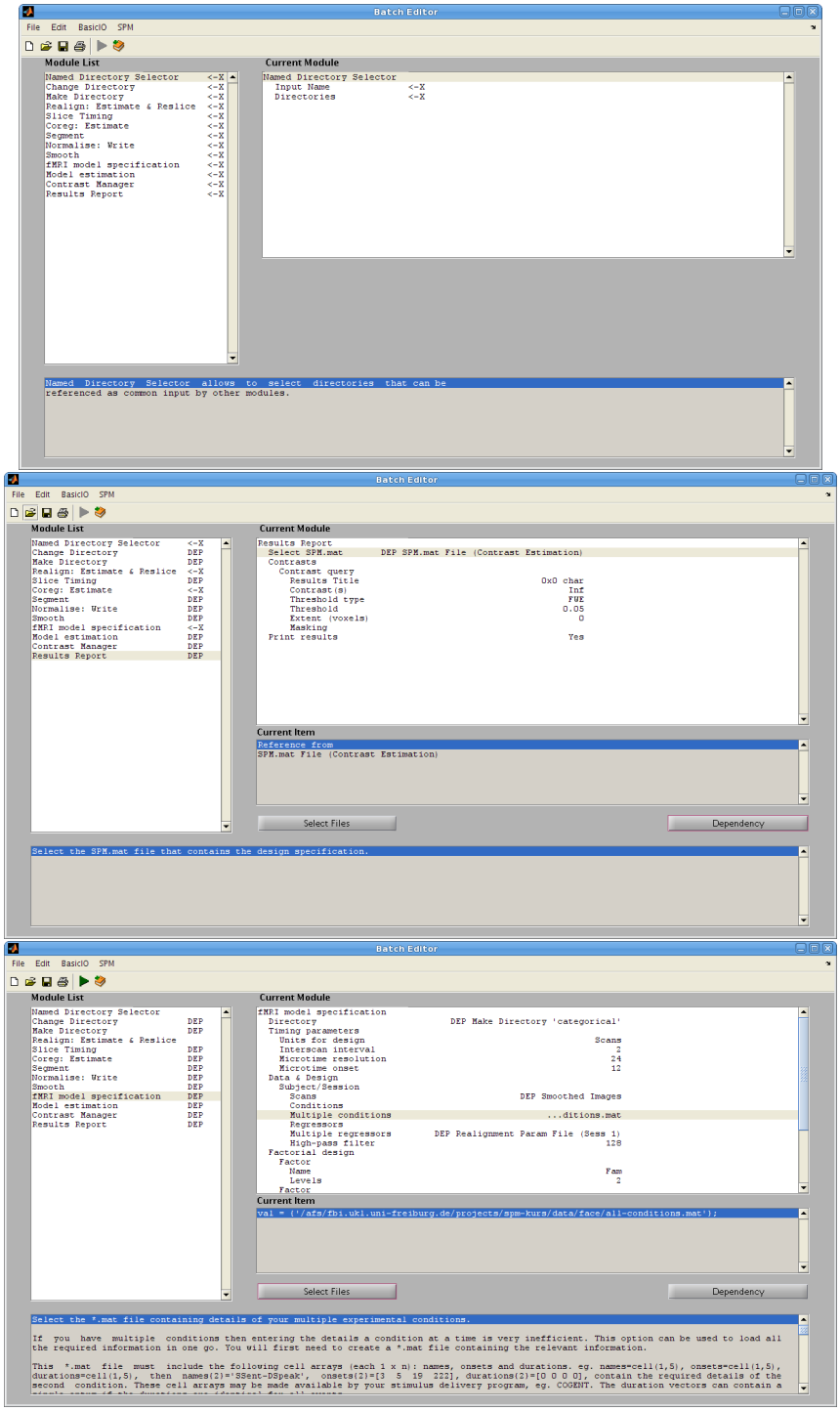

Figure 1.5: All stages of batch entry

1.3 SPM5 to matlabbatch transition guide

This is a short overview to describe code organisation and interfaces between SPM and the batch system.

1.3.1 Code Reorganisation

The following paths have changed:

- `fullfile(spm('dir'),'matlabbatch')` Core batch system.
- `fullfile(spm('dir'),'config')` New SPM config files.
- `fullfile(spm('dir'),'oldconfig')` Old SPM config files (unused)
- `spm_jobman.m` and `spm_select.m` replaced with compatibility code
- `spm_Menu.fig` Callbacks adapted

Configuration code has been generated automatically from the existing SPM configuration using `cfg_struct2cfg` and `gencode`. This sometimes results in redundant/duplicate code. Also, conditional constructs like `if`, `case` may not have been considered.

Some assignments to configuration items are guarded by validity checks. Usually, there will be a warning issued if a wrong value is supplied. Special care needs to be taken for `.prog`, `.vfiles`, `.vout`, `.check` functions or function handles. The functions referenced here must be on MATLAB path before they are assigned to one of these fields. For toolboxes, this implies that toolbox paths must be added at the top of the configuration file.

For details, see section 1.4.

1.3.2 Interfaces between SPM and Matlabbatch

Unchanged harvested job structure.

Changed Top-level node in SPM config now called `spmjobs` instead of `jobs`. New overall top-level node `matlabbatch`. `spm_jobman` will convert and load SPM5 style batch jobs into the new batch system.

Changed Configuration file syntax - instead of structs, configuration items are now objects. Structs of type `<type>` are now represented as objects of class `cfg_<type>`. Existing SPM5 configuration can be imported using `cfg_struct2cfg`. There is a new class `cfg_exbranch` which is used for branches that have a `.prog` field.

Deprecated Virtual files have been replaced by dependencies. These require computations to return a single output argument (e.g. a cell, struct). Parts of this output argument can be passed on to new inputs at run-time. Virtual files are treated as a special output argument.

Added Interface to the batch system

- `cfg_util` Configuration management, job management, job execution

- `cfg_serial` A utility to fill missing inputs and run a job (optionally with a GUI input function)
- `cfg_ui` GUI - inspired by `spm_jobman`, but modified to work around some MATLAB GUI "features" (like input widgets loosing focus before editing has finished).

1.4 Configuration Code Details

Configuration code has been split into two files per configuration:

spm_cfg_*.m Configuration classes, `.check`, `.vout` subfunctions

spm_run_*.m Run-time code, takes job structure as input and returns output structure as specified in `.vout`.

In a few cases (where there was no additional magic in the code), run-time code has been integrated into the main SPM code. This may be useful to run test batches without using the configuration/batch system.

1.4.1 Virtual Outputs

Virtual outputs are described by arrays of `cfg_dep` objects. These objects contain a "source" and a "target" part. Functions may have more than one virtual output (e.g. one output per session, a collection of results variables). One `cfg_dep` object has to be created for each output.

Only two fields in the "source" part need to be set in a `.vout` callback:

sname A display name for this output. This will appear in the dependencies list and should describe the contents of this dependency.

src_output A subscript reference that can be used to address this output in the variable returned at run-time.

tgt_spec (optional) A description on what kind of inputs this output should be displayed as dependency. This is not very convenient yet, the `match` and `cfg_findspec` methods are very restrictive in the kind of expressions that are allowed.

The `.vout` callback will be evaluated once the configuration system thinks that enough information about the *structure* of the outputs is available. This condition is met, once all in-tree nodes `cfg_(ex)branch`, `cfg_choice`, `cfg_repeat` have the required number of child nodes.

The `.vout` callback is called with a job structure as input, but its code *should not rely* on the evaluation of any contents of this structure (or at least provide a fallback). The contents of the leaf nodes may not be set or may contain a dependency object instead of a value during evalution of `.vout`.

The "target" part will be filled by the configuration classes, the `src_exbranch` field is set in `cfg_util`.

1.4.2 SPM Startup

The top level configuration file for SPM is `spm_cfg.m`. It collects SPM core configuration files and does toolbox autodetection. If a toolbox directory contains `*_cfg_*.m` files, they will be loaded. Otherwise, if there are only SPM5-style `*_config_*.m` files, the configuration will be converted at run-time using `cfg_struct2cfg`.

1.4.3 Defaults Settings

Due to automatic code generation from the previous configuration, some defaults are coded in the configuration files. This code will be removed, all defaults will be read from `spm_def.m` in the configuration folder. This file currently links to `spm_defaults.m` and contains code to include toolbox-specific defaults in a way similar to toolbox autodetection (i.e. they appear under `defaults.tools.<toolbox_tag>` in the defaults structure.

1.4.4 Toolbox Migration

In the `fullfile(spm('dir'),'toolbox')` folder there exists a migration utility `spm_tbx_config2cfg.m`. This utility will create a `*_cfg_*.m` and a `*_def_*.m` file based on the configuration tree given as input argument.

1.5 Utilities

1.5.1 Batch Utilities

Matlabbatch is designed to support multiple applications. A standard application "BasicIO" is enabled by default. Among other options, it contains file/file selection manipulation utilities which can be used as as dependency source if multiple functions require the same set of files as input argument. For debugging purposes, "Pass Output to Workspace" can be used to assign outputs of a computation to a workspace variable.

The `cfg_confgui` folder contains an application which describes all configuration items in terms of configuration items. It is not enabled by default, but can be added to the batch system using `cfg_util('addapp'...)`. This utility can be used generate a batch configuration file with the batch system itself.

1.5.2 Matlab Code Generation

The `gencode` utility generates MATLAB `.m` file code for any kind of MATLAB variable. This is used to save batch files as well as to generate configuration code.

1.5.3 Configuration Management

The backend utility to manage the configuration data is `cfg_util`. It provides callbacks to add application configurations, and to load, modify, save or run jobs. These callbacks are used by two frontends: `cfg_ui` is a MATLAB GUI, while `cfg_serial` can be used both as a GUI and in script mode. In script mode, it

will fill in job inputs from an argument list. This allows to run predefined jobs with e.g. subject dependent inputs without knowing the exact details of the job structure.