FSL MELODIC ANALYSIS USER GUIDE

2013

Instructions, rationale and example scripts to perform subject preprocessing and Melodic, Seed-based functional connectivity and Tractography analyses with FSL package.
Rev. 1.1, April 2013.

Table of Contents

FSL ANALYSIS: USER GUIDE

OVERVIEW

This guide is intended to explain all the processing steps necessary to perform i) subject preprocessing and ii) melodic iii), seed-based functional connectivity (SBFC) and iv) tractography analyses.

The first step will convert and store subjects' images data and perform the first pre-processing at the subject level. It will perform a set of operations thought to ease the further processing, like scalping, segmentation, main co-registrations among different sequences, single subject melodic and confound signal removal from epi data for SBFC.
The last three sections of this guide will instead focus on the three techniques (melodic, sbfc and tracto), explaining all the steps to perform single-subject and group-level and the final statistical analyses

Naming conventions

Images file and their processed derivate, will be stored in a fixed and conventional way. This important assumption which involves the creation of a standard folder and file names scheme, will allow the present coding architecture to automatically process data through a set of automatized and highly parameterized scripts. Each deviation from the standard scheme will prevent the correct functioning of the analyses scripts. Project and subject fixed file system schema will be extensively explained in the following paragraphs.

Script architecture: global and projects scripts and variables

To perform all the analysis two classes of bash scripts will be available: global and projects scripts. The former is a collection of functions which perform several steps. They are predetermined and *users* **do not have** to modify them. They accept several parameters according to investigated subjects' information and *user* settings. The project scripts must be instead edited by the *user* according to the desired analysis. In the folder $GLOBAL_SCRIPT_DIR/examples/sbfc an example of all the project scripts used for melodic analysis can be found.

Multi-threaded scripting

Some global script can be invoked in a multi-threaded manner. In that case, in the calling project script you can define the number of CPU to be used, providing the list of multiple cases (subjects, folders, glm files) to an intermediary script that will automatically implement the multi-threading process.

ANALYSIS TOOLS

Global variables

Each project script must start with a specific header

```
GLOBAL_SCRIPT_DIR=/homer/home/dati/fsl_global_scripts
PROJ_DIR=/media/Iomega_HDD/CAB/STUDY_AD_FTD         # change according to your project path
. use_fsl 4                                         # change according to desired FSL version
#=============================================

. $GLOBAL_SCRIPT_DIR/init_vars.sh $PROJ_DIR
#=============================================
```

It will call the init_vars.sh script which will initialize some project variables, defining the paths to important global/project paths and filename as listed here:

[**init_vars.sh**]

```
GLOBAL_GROUP_SCRIPT_DIR=$GLOBAL_SCRIPT_DIR/process_group
GLOBAL_SUBJECT_SCRIPT_DIR=$GLOBAL_SCRIPT_DIR/process_subject
GLOBAL_GLM_SCRIPT_DIR=$GLOBAL_SCRIPT_DIR/glm

GLOBAL_DATA_TEMPLATES=$GLOBAL_SCRIPT_DIR/data_templates
MULTICORE_SCRIPT_DIR=$GLOBAL_SCRIPT_DIR/multicore_scripting

FSL_BINS=$FSLDIR/bin
FSL_DATA_STANDARD=$FSLDIR/data/standard
FSL_STANDARD_MNI_2mm=$FSL_DATA_STANDARD/MNI152_T1_2mm_brain.nii.gz

# projects dirs
PROJ_SCRIPT_DIR=$PROJ_DIR/script
PROJ_GROUP_SCRIPT_DIR=$PROJ_SCRIPT_DIR/group
PROJ_GROUP_ANALYSIS_DIR=$PROJ_DIR/group_analysis

SUBJECTS_DIR=$PROJ_DIR/subjects
```

Subjects variables

Moreover, in order to ease the processing of subjects data, a second script (**subject_init_vars.sh**) can be invoked to define subjects file names and path and be used in own project scripts. The list of those variables is later summarized. Before calling this script user must call the init_vars script and define a SUBJ_NAME variable which must correspond to the folder name containing those subject images. Considering that the project file system schema is implicitly longitudinal, that is each subject folder will have one subfolder for each MRI longitudinal session, unless specified all the variables will point to the first session images. To specify the current longitudinal session the variable SESS_ID is available. When its values will not be specified it will set by default to 1.

[**subject_init_vars.sh**]

```
if [ -z $SUBJ_NAME ]; then echo "SUBJ_NAME not defined..exiting"; exit; fi
if [ -z $SESS_ID ]; then      SESS_ID=1; fi

SUBJECT_DIR=$SUBJECTS_DIR/$SUBJ_NAME/s$SESS_ID
if [ ! -d $SUBJECT_DIR ]; then echo "ERROR: subject dir ($SUBJECT_DIR) not present !!!.....exiting"; fi

T1_IMAGE_LABEL=m3DT1
T1_DIR=$SUBJECT_DIR/mpr
```

```
T1_DATA=$T1_DIR/$T1_IMAGE_LABEL
T1_BRAIN_DATA=$T1_DIR/$T1_IMAGE_LABEL"_brain"
FAST_DIR=$T1_DIR/fast
FIRST_DIR=$T1_DIR/first
SIENAX_DIR=$T1_DIR/$T1_IMAGE_LABEL"_sienax"
T1_SEGMENT_GM_PATH=$T1_DIR/c13DT1
T1_SEGMENT_WM_PATH=$T1_DIR/c23DT1
T1_SEGMENT_CSF_PATH=$T1_DIR/c33DT1

DTI_IMAGE_LABEL=dw
DTI_EC_IMAGE_LABEL=dw_aligned_m1
DTI_ROTATED_BVEC=dw_aligned.bvecs
DTI_BVAL=dw_aligned.bvals
DTI_DIR=$SUBJECT_DIR/dti
DTI_DATA=$DTI_DIR/$DTI_IMAGE_LABEL
DTI_FIT_LABEL=dti
BEDPOSTX_DIR=$DTI_DIR/bedpostx
PROBTRACKX_DIR=$DTI_DIR/probtrackx

RS_IMAGE_LABEL=resting
RS_DIR=$SUBJECT_DIR/resting
RS_DATA=$RS_DIR/$RS_IMAGE_LABEL
SBFC_DIR=$RS_DIR/fc

DE_DIR=$SUBJECT_DIR/de
DP_IMAGE_LABEL=dp
DP_DATA=$DE_DIR/$DP_IMAGE_LABEL
DP_BRAIN_DATA=$DE_DIR/$DP_IMAGE_LABEL"_brain"

T2_IMAGE_LABEL=t2
T2_DATA=$DE_DIR/$T2_IMAGE_LABEL
T2_BRAIN_DATA=$DE_DIR/$T2_IMAGE_LABEL"_brain"

ROI_DIR=$SUBJECT_DIR/roi
```

Global script input parameters

Global scripts perform different operations and can be thus invoked with several parameters which differ according to the scope of the script. Nevertheless, a set of conventions were created for the most common parameters usage. Users are asked to respect as much as possible these conventions while defining new global scripts parameters.
As a general rule the following letters correspond to:

f=files, d=directory, i=input, o=output, n=name, p=path

for example:
-ifn: input file name
-ofn: output fine name
-ifp: input file path
-ofp: output fine path

-idn: input directory name
-idn2: second input directory name (usually idn=resting, idn2=rs.ica)
-odn: output directory name
-idp: input directory path
-odp: output directory path

-model: fsf full path
-modeln: fsf file name

-son: series output name

-stdimg: alternative standard image
-std4img: alternative standard image at 4mm
-seed, stop, target : are subjects related variables, full path is obtained by first adding a subject dependent directory (e.g. $ROI_DIR/reg_dti)
-seedp, stop, targetp : are instead full paths, e.g. when used over images registered in the standard space.

Generally speaking, thus is not a strict rule, names corresponds to relative paths (ifn, ofn, idn, odn, model) and are used for subject-level analyses, while absolute paths are used for group analyses.

Multi-threading scripting

There is a mechanism which lets you run a list of script simultaneously. You define a parameters list, the number of CPU to use and the name of a script, and a special script will run those script in N CPU simultaneously, passing each element of the list as parameter.

E.g1: you can define a "single_subject_melodic.sh" and define an array of subjects label and run N melodic simultaneously.
Eg2: you can have a script which run a randomize with a specific con/mat couple. You can define a list of GLM model and run all of them
Eg3: you may calculate a same GLM model over N resting-state folders simultaneously.

This is the command:

```
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_cases" $PROJ_DIR  $extra
```

The first 4 parameters are mandatory and must be the following

NUM_CPU: number of simultaneous processes
EXECUTE_SH global script to be executed
arr_cases string-equivalent of an array of script parameters iteratively passed to the script
PROJ_DIR path of current project
EXTRA is a possible list of further parameters which will be passed "as-is"

arr_subj can be declared as a normal array

```
declare -a array_cases=(a b c d e f …..)
```

but you have to pass it as a string by calling this command

```
str_arr_cases=`echo ${arr_cases[@]}`
```

and then pass it as parameter put in bracket

```
. /../define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$str_arr_cases" $PROJ_DIR $EXTRA
```

The script called in this way will receive only the 3rd and 4th variable (the single case and the project path) and thus will have to be designed accordingly (fist parameter, the “case” variable, the second parameter is the project path).

Subjects list

In order to define the subjects participating in a study, user must create a bash file with the following path: $PROJ_SCRIPT_DIR/subjects_list.sh. It will define one array for each study population, that is, one for controls (which are often located under a different $PROJ_DIR) and one for each patients subgroups. Subjects lists are represented as an array, and can be cycled through a *for* statement as here specified:

```
. $PROJ_SCRIPT_DIR/subjects_list.sh
SESS_ID=X
for SUBJ_NAME in ${arr_controls[@]}
do
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
  ….. do whatever you want ….......
done
```

Note: SUBJ_NAME is a mandatory label as the following script use it to define its variables

SUBJECT PREPROCESSING

The processing pipeline include: i) project file system generation, ii) images conversion and renaming, iii) images preprocessing

1) Project File system creation

Standard Folders:

```
PROJ_DIR=/bender/home2/dati/PRJ1
mkdir $PROJ_DIR
mkdir -p $SUBJECTS_DIR
mkdir -p $PROJ_SCRIPT_DIR/docs
mkdir -p $PROJ_SCRIPT_DIR/glm/template
mkdir -p $PROJ_SCRIPT_DIR/utility
mkdir -p $PROJ_GROUP_ANALYSIS_DIR
```

Optional folder (according to available sequences)

[melodic]

```
mkdir -p $ PROJ_GROUP_ANALYSIS_DIR/melodic/group_templates
mkdir -p $ PROJ_GROUP_ANALYSIS_DIR/melodic/dr
mkdir -p $ PROJ_SCRIPT_DIR/melodic
```

[seed-based functional connectivity]

```
mkdir -p $PROJ_GROUP_ANALYSIS_DIR/sbfc
mkdir -p $PROJ_SCRIPT_DIR/sbfc
```

[dti / tbss]

```
mkdir -p $PROJ_GROUP_ANALYSIS_DIR/tbss
mkdir -p $PROJ_SCRIPT_DIR/tbss
```

[dti / bedpostx]

```
mkdir -p $PROJ_GROUP_ANALYSIS_DIR/probtrackx
mkdir -p $PROJ_SCRIPT_DIR/probtrackx
```

2) Subjects File system creation and data preprocessing

Subject's data arrive at CAB as a zip file containing hundreds/thousands of DICOM files. The present Standard Operating Procedure asks user to:

a) run the 2nii.sh bash script; it calls the dcm2nii program (converting DICOM to nifty) which:
- recognize the MRI sequence each file belongs to
- convert each DICOM image to NIFTI format and split them to several folders, one for each sequence.

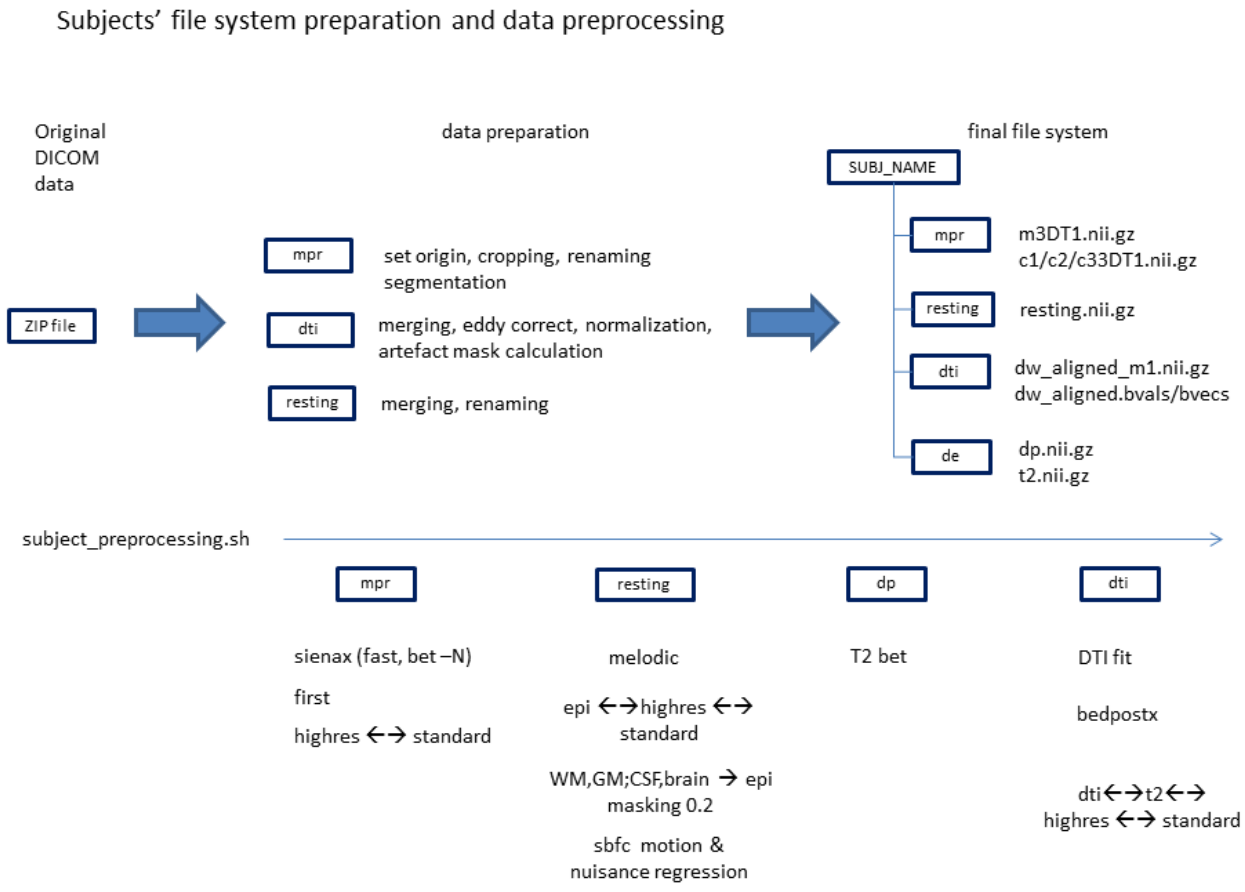
b) open Matlab and run a list of command that reorient all the sequences, merge dti and resting state files in 4D files,
c) Rename resting file as resting.nii.gz
d) Manually reorder of the sequence obtained, in order to remove undesired sequences and move subjects' file to PROJ_DIR/subjects/ SUBJ_NAME
e) T1 preprocessing:

Re-orienting, cropping, set origin, segment with SPM, expected output is: m3DT1.nii, c13DT1.nii.gz, c23DT1.nii.gz, c33DT1.nii.gz

f) DW preprocessing:

eddy_current, vectors aligning, normalization, expected output is dw_aligned.nii.gz. Then an exclusion mask is created and applied to the data, creating a file called dw_aligned_m1.nii.gz

g) DE …..



3) Subject-level processing

Then a bash script (subject_preprocessing.sh) must be called in order to start the subject-level preprocessing which involves these steps:

T1:

- sienax without deleting intermediate files, that is, preserving BET and FAST output data. Inner BET is call by default with the –B option and an f value of 0.3
- linear and non-linear registration to and from standard MNI template

- first (optional). It's possible to indicate which structures it will segment and where (below the ROI_DIR/reg_t1 folders) it will store them.

Resting
- Subject level melodic (standard output folder name : $RS_DIR/resting.ica)
- Copying of reg folder to proper $ROI_DIR subfolders.
- linear and non-linear registration to and from standard MNI template

Sbfc over resting state data
- Registration of fast images to epi space
- Extraction of WM,CSF,BRAIN time series
- Motion and nuisance signal regression

T2
- Bet

DTI
- dtifit
- Bedpostx: it's possible to defind a subfolder of $DTI_DIR as output directory name
- Linear and non-linear registration to and from standard MNI template / T2 / T1

ROI directory

In addition to these preprocessing operations, the welcome script also calculate all the possible (linear and non linear) co-registration matrices and warp files among all the modalities. Transformation files and some output images are saved in distinct subfolder of $ROI_DIR folder according to the destination space:
e.g. : dti2highres is stored in ROI_DIR/reg_t1 folder
while highres2epi is stored in ROI_DIR/reg_epi.

These folders should be also used to store ROI images used in the study. For example subcortical structures derived from first, mask of csf and gray and white matter, etc….

Script parameters

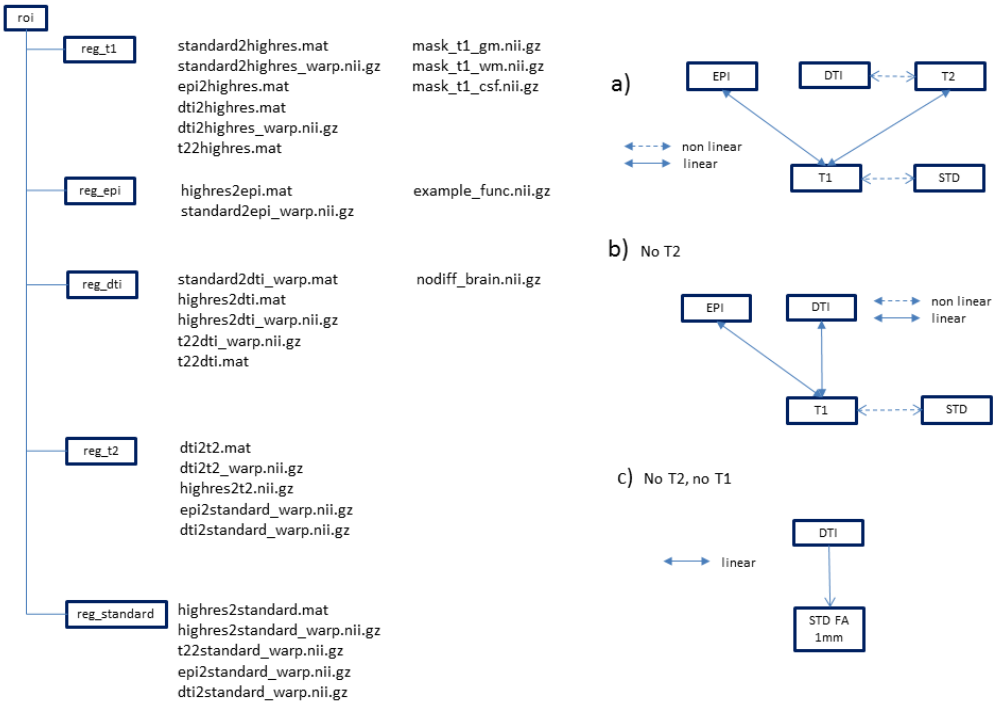
The preprocessing steps can be selected with the following parameters.

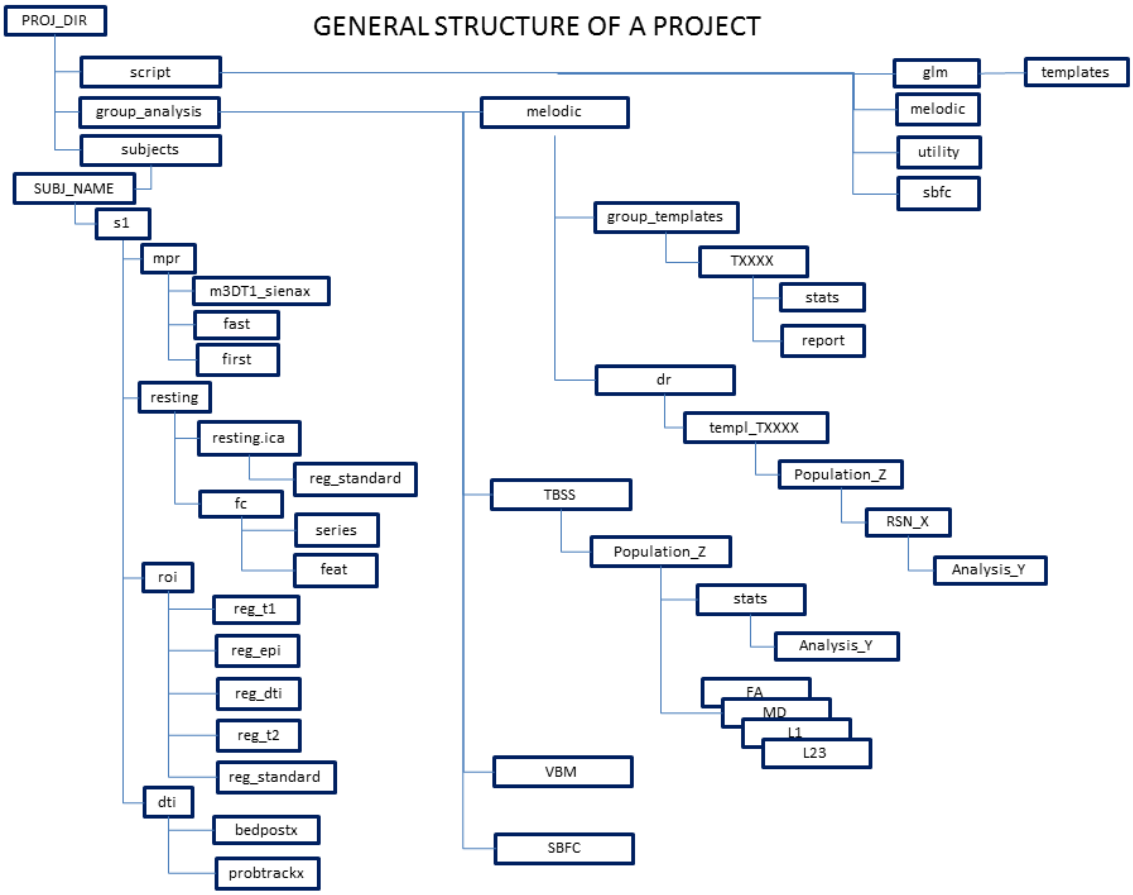
```
echo "         -sienax)              bet param string e.g. "-B -f 0.3"
echo "         -firststructs)        structs_list"
echo "         -firstodn)            output dir name"
echo "         -mel)                 output dir name"
echo "         -bedx)                output dir name"
echo "         -dtifit"
echo "         -sbfcpre)"
echo "         -fsl5)"
```

An example of script call is this:

```
echo "usage: $0 SUBJ_LABEL PROJ_DIR -sienax "-SNB –f 0.25" -firststructs L_Thal,R_Thal –firstodn first
-mel resting -dtifit -sbfcpre –bedx bedpostx"
```

Linear & non-linear registration among all sequences





MELODIC

The melodic package allows you to evaluate the **within-network** functional connectivity. It decomposes the brain rest activity in resting-state networks (RSN) and then, independently for each RSN, assess the functional connectivity of each voxel contained within the RSN to the other voxel of the same network. This measure can be then compared between groups or correlated with clinical, behavioral and demographic variables.

The processing pipeline include: o) subject welcome, i) single-subject analysis, ii) template creation, iii) dual-regression of subjects data to the template, iv) statistical analysis, v) data visualization and vi) final packaging for publication.

1: subjects_single_melodic:

GUI usage

The analysis is normally performed by the GUI application called Melodic. This tool requires that you define:

- subject RS image: resting.nii.gz
- the anatomical scalped brain image m3DT1.nii.gz
- the TR and TE values of the sequence
- Information over the normalization to standard anatomical template
    - anatomical template path
    - spatial re-sampling dimension in mm. (usually 4 mm)
    - registration approach (linear vs non-linear)
- the spatial smoothing (5 or 6 mm),
- the high-pass filter cutoff (between 100-150)
- optionally the output folder if different from the default one (resting.ica)

NOTE: the TE values must be edited manually in the fsf file produced by the GUI application. Hence it is advisable to create a study template file for 1st-level melodic analysis, load that file and modify subject-dependent information.

Scripted usage

The global script for this step is a multi-threated script; in the project script you must define this information:

- number of CPU to be used
- the global script ($GLOBAL_SUBJECT_SCRIPT_DIR/execute_subject_melodic.sh)
- fsf template
- subjects list string name (variable defined in $PROJ_SCRIPT_DIR/subjects_list.sh)
- custom output folder (optionally)

```
. $PROJ_SCRIPT_DIR/subjects_list.sh

EXECUTE_SH=$GLOBAL_SUBJECT_SCRIPT_DIR/execute_subject_melodic.sh
melodic_fsf_template=$PROJ_SCRIPT_DIR/glm/singlesubj_melodic
declare -i NUM_CPU=2
postfix_output_folder_name="non-default_name"  # optional string appended to the label : $SUBJ_NAME-rs
#================================================================

# standard call....read: SUBJ_NAME/resting/resting.nii.gz,  create a folder:
SUBJ_NAME/resting/resting.ica
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -model
$melodic_fsf_template

# non-standard input file....  read: SUBJ_NAME/resting/resting_skip4vol.nii.gz, create a folder
SUBJ_NAME/resting/resting.ica
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -ifn
resting_skip4vol -model $melodic_fsf_template

# non-standard input file and folder....  read: SUBJ_NAME/rs2/resting_skip4vol.nii.gz, create a folder
SUBJ_NAME/rs2/resting.ica
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -ifn
resting_skip4vol -idn rs2 -model $melodic_fsf_template

# non-standard input file and folder and output dir....  read: SUBJ_NAME/rs2/resting_skip4vol.nii.gz,
create a folder SUBJ_NAME/rs2/resting _denoised.ica
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -ifn
resting_skip4vol -idn rs2 -model $melodic_fsf_template -odn resting_denoised
```

The output of such analysis is a folder (SUBJECTS_DIR/SUBJ_NAME/resting/resting.ica) containing the subject-level analysis of resting state data. The most important files created are the

- resting.ica /filtered_func_data.nii.gz
- resting.ica/reg_standard/filtered_func_data.nii.gz (registered to the anatomical template)

The latter is of special interest as it will be later used by the dual regression process. They represent the filtered data.
This step is used to verify the quality of subjects' data, how his movement affected the RSN identification and if he needs a specific denoising. As later discussed

1a: denoising (optional)

It is possible to remove specific artifacts from the original data after a preliminary melodic analysis. The procedure is realized by visually inspecting the melodic output, take note of the artefactual components id and invoke the fsl_regfilt command which remove those components from the signal and create a denoised file.

<u>There are two approaches:</u>

- simply correct the rs data without changing its final name (substitute the original file which is in turn renamed as filtered_func_data_original.nii.gz)
- preserve original file name and create a denoised version (filtered_func_data_denoised)

The former is used when you plan to analyze original data and you had just to correct the data of few subjects. On the contrary, the latter approach is used when you plan to denoise all subjects data, thus it creates a reg_standard_denoised folder, later used by dual-regression analyses on denoised data.

Few subjects correction

```
declare -a arr_subjects2denoise=(DYT_B_prsic_svetislav)
declare -a arr_ic2remove=("1,2,3,4,5,6,7,8,9,10, 12,13,14,15, 17,18,19, 22,23,25,26,27,28")

declare -i cnt=0
for SUBJ_NAME in ${arr_subjects2denoise[@]}
do
  echo "$SUBJ_NAME"
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
    mv $RS_DATA.ica/reg_standard $RS_DATA.ica/reg_standard_original
    mv $RS_DATA.ica/filtered_func_data.nii.gz $RS_DATA.ica/filtered_func_data_original.nii.gz
    $FSLDIR/bin/fsl_regfilt -i $RS_DATA.ica/filtered_func_data_original.nii.gz -o
$RS_DATA.ica/filtered_func_data.nii.gz -d $RS_DATA.ica/filtered_func_data.ica/melodic_mix -f
"${arr_ic2remove[cnt]}"
    $FSLDIR/bin/featregapply $RS_DATA.ica
    $FSLDIR/bin/fslroi $RS_DATA.ica/filtered_func_data.nii.gz
$RS_DATA.ica/filtered_func_data_skip4vol.nii.gz 4 196
    $FSLDIR/bin/fslroi $RS_DATA.ica/reg_standard/filtered_func_data.nii.gz
$RS_DATA.ica/reg_standard/filtered_func_data_skip4vol.nii.gz 4 196
     cnt=$cnt+1
done
```

Whole population correction

```
declare -a arr_ic2remove=("1,2,3,4,5,6,7,8,9,10, 12,13,14,15, 17,18,19, 22,23,25,26,27,28" "…..” “……”
“……” ……)

declare -i cnt=0
for SUBJ_NAME in ${arr_patients[@]}    #variable found in subjects_list.sh
do
  echo "$SUBJ_NAME"
  .   $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
  mv $RS_DATA.ica/reg_standard $RS_DATA.ica/reg_standard_original
  mv $RS_DATA.ica/filtered_func_data.nii.gz $RS_DATA.ica/filtered_func_data_original.nii.gz

  $FSLDIR/bin/fsl_regfilt -i $RS_DATA.ica/filtered_func_data_original.nii.gz -o
$RS_DATA.ica/filtered_func_data.nii.gz -d
  $RS_DATA.ica/filtered_func_data.ica/melodic_mix -f "${arr_ic2remove[cnt]}"
   $FSLDIR/bin/featregapply $RS_DATA.ica
   $FSLDIR/bin/fslroi $RS_DATA.ica/filtered_func_data.nii.gz
$RS_DATA.ica/filtered_func_data_skip4vol.nii.gz 4 196
   $FSLDIR/bin/fslroi $RS_DATA.ica/reg_standard/filtered_func_data.nii.gz
$RS_DATA.ica/reg_standard/filtered_func_data_skip4vol.nii.gz 4 196
    cnt=$cnt+1
done
```

Note1

In this phase is it advisable to be very conservative: in case of doubt keep the IC, it may hide some good signal, and delete only those IC related to subjects movements, which highly differs between subjects. For instance, the artifact related to cardiac impulse and blood flow is present in each subjects and have similar spatio-frequency pattern, thus the group melodic algorithm is perfectly able to find it in every subjects and associate it to a common IC which will not be considered in the group template. The criteria to define artefactual IC are out of the scope of the present guide.

Note2

In the two examples, the first 4 volumes of the data were removed from the filtered_func_data. This approach is used by several researchers, particularly when their scanners are old or not perfectly set-up. The signal present in the first volumes can be in fact altered by the not perfect signal stabilization.

It is advisable to view some subject melodic in order to decide if this procedure is necessary and how many volumes should be removed.

2: template definition

This step creates the group template representing the RSN spatial pattern that will be later investigated with randomise.
The subjects used to create the template must belong to one of the following populations:

- healthy controls, better if age-matched, **different from those included** in the study
- entire population of the study (controls + patients)

Otherwise you can use the templates located at the following paths.

$GLOBAL_SCRIPT_DIR/data_templates/rsn/fsl_20/rsn20_444.nii.gz
$GLOBAL_SCRIPT_DIR/data_templates/rsn/bishwal/metaICA_2mm.nii.gz

After having created the group template (a) at
$PROJ_GROUP_ANALYSIS_DIR/melodic/group_templates/XXX,
you must inspect the IC by opening the web page at /.../XXX/report/00index.html and then (b) create the template script file, an sh file containing the variables which defines the template characteristics.

Template creation

It is performed using the following script, where user must define i) the TR of the sequence, ii) the name of subjects resting-state image input folder (rs), iii) the name of the RS images (rs), iv) the template name and v) use the proper subjects list (whom corresponding variables are defined in the subjects_list.sh file)

```
TR_VALUE=3.0
SUBJECTS_INPUT_RS_DIR_NAME=resting
SUBJECTS_INPUT_RS_NAME=resting

output_template_name=belgrade_dyt_controls21_patients45_skip4vol
arr_ctrl=${arr_controls21[@]}
arr_patients=${arr_patients45[@]}
input_file_name=filtered_func_data_skip4vol


CTRL_SUBJECTS_DIR=/gnappo/home2/dati/.../HC/subjects
MELODIC_OUTPUT_DIR=$PROJ_GROUP_ANALYSIS_DIR/melodic/group_templates/$output_template_name
mkdir -p $MELODIC_OUTPUT_DIR

filelist=$MELODIC_OUTPUT_DIR/.filelist_$template_name

echo "creating file lists"
bglist=""
masklist=""

for SUBJ_NAME in ${arr_ctrl[@]}
do
reg_standard_dir=$CTRL_SUBJECTS_DIR/$SUBJ_NAME/s$SESS_ID/$SUBJECTS_INPUT_RS_DIR_NAME/$SUBJECTS_INPUT_RS
_NAME.ica/reg_standard
  bglist="$bglist $reg_standard_dir/bg_image"
  masklist="$masklist $reg_standard_dir/mask"
```

```
  echo "$reg_standard_dir/$input_file_name" >> $filelist
done

for SUBJ_NAME in ${arr_patients[@]}
do
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
  reg_standard_dir=$SUBJECT_DIR/$SUBJECTS_INPUT_RS_DIR_NAME/$SUBJECTS_INPUT_RS_NAME.ica/reg_standard
  bglist="$bglist $reg_standard_dir/bg_image"
  masklist="$masklist $reg_standard_dir/mask"
  echo "$reg_standard_dir/$input_file_name" >> $filelist
done

echo "merging background image"
$FSLDIR/bin/fslmerge -t $MELODIC_OUTPUT_DIR/bg_image $bglist
$FSLDIR/bin/fslmaths $MELODIC_OUTPUT_DIR/bg_image -inm 1000 -Tmean $MELODIC_OUTPUT_DIR/bg_image -odt
float
echo "merging mask image"
$FSLDIR/bin/fslmerge -t $MELODIC_OUTPUT_DIR/mask $masklist

echo "start group melodic !!"
$FSLDIR/bin/melodic -i $filelist -o $MELODIC_OUTPUT_DIR -v --nobet --bgthreshold=10 --tr=$TR_VALUE --
report --guireport=$MELODIC_OUTPUT_DIR/report.html --bgimage=$MELODIC_OUTPUT_DIR/bg_image -d 0 --
mmthresh=0.5 --Ostats -a concat
```

Template script file

User must define the parameters of a group template by creating a proper file which declares some variables used by the subsequent processing steps. These file must be stored in the folder:

$GLOBAL_SCRIPT_DIR/melodic_templates/
by calling : . $GLOBAL_SCRIPT_DIR/melodic_templates/fsl_rsn20_444.nii.gz
you have all these variables available in your project script.

```
template_name=fsl_rsn20

TEMPLATE_MELODIC_IC                 /..path to/melodic_IC.nii.gz or which ever 4D images
MASK_IMAGE                          / path to / mask image
BG_IMAGE                            / path to / bg_image.nii.gz
TEMPLATE_STATS_FOLDER               / path to thresh_tstatsXXX images
TEMPLATE_MASK_FOLDER                / path to / folder containing single RSN mask.

str_pruning_ic_id="0,1,3,4,6,7,12,15"              /  0-based ids of RSN networks of interest
str_arr_IC_labels="SM,AUDIO,DMN,..."               / labels of RSN networks of interest
declare -a arr_IC_labels=(SM AUDIO DMN .. .. ..)   / same elements of previous parameters but as an
array
```

VERY IMPORTANT !!!!:
In *str_pruning_id* variable, RSN ID are 0-based. That is, when you select your RSN of interest in the report folder, you must subtract 1 from its component number (DMN is at component 6, you must note its ID as 5)

Template RSN masks

If you want to restrict randomize analyses to the RSN mask you have two options. One is to use the thresh_zstat maps calculated by group-melodic step (located in: group_templates/templ_name/stats folder). The second consists in calculating the mean map of each component with randomize. A script is available to perform this step: it performs a randomize, then it masks them with a threshold of 0.998 creating a mask for each RSN previously defined as network of interests.

```
SINGLE_IC_PRUNING_SCRIPT=$GLOBAL_GROUP_SCRIPT_DIR/dual_regression_split2singleIC.sh
EXECUTE_STATS_SH=$GLOBAL_GROUP_SCRIPT_DIR/dual_regression_randomize_singleIC_multiple_folders_mean_mask
.sh

NUM_PERM=5000
NUM_CPU=3
NUM_SUBJECTS=78                      # preserved for backward compatibility

. $GLOBAL_SCRIPT_DIR/melodic_templates/belgrade_controls.sh       # load template-related variables
TEMPLATE_DIR=$PROJ_GROUP_ANALYSIS_DIR/melodic/group_templates/$template_name
filelist=$TEMPLATE_DIR/.filelist_$template_name
DR_DIR=$TEMPLATE_DIR/dr

echo "start DR SORT !!"
. $GLOBAL_GROUP_SCRIPT_DIR/dual_regression_sort.sh $TEMPLATE_MELODIC_IC 1 $DR_DIR `cat $filelist`

echo "start DR SPLIT 2 SINGLE ICs !!"
. $GLOBAL_GROUP_SCRIPT_DIR/dual_regression_split2singleIC.sh $TEMPLATE_MELODIC_IC $DR_DIR $DR_DIR
$NUM_SUBJECTS "$str_pruning_ic_id" "$str_arr_IC_labels"

str_folders="$DR_DIR/${arr_IC_labels[0]}"
for ic in ${arr_IC_labels[@]:1}
do
  str_folders="$str_folders $DR_DIR/$ic"
done

. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_STATS_SH "$str_folders" $PROJ_DIR
-nperm $NUM_PERM -maskf $GLOBAL_DATA_TEMPLATES/gray_matter/mask_T1_gray_4mm.nii.gz

mkdir -p $TEMPLATE_DIR/mask

for ic in ${arr_IC_labels[@]}
do
  input_file=$DR_DIR/$ic/mean/$ic"_mean_mask_tfce_corrp_tstat1.nii.gz"
  output_file=$TEMPLATE_DIR/mask/"mask_"$RSN_LABEL.nii.gz
  fslmaths $input_file -thr 0.998 -bin $output_file
done

rm -rf $DR_DIR
```

The last parameter is represented by a mask of gray-matter only in standard space resampled to 4mm. This in order to obtain mask representing the mean RSN map limited to gray-matter

NOTE, differences between the two masks: The latter mask corresponds to the group-melodic RSN image seen in the corresponding web page. The former instead is larger, including areas which are not usually described in the literature as belonging to that network. Nevertheless, some reviewer might not appreciate the smaller mask as you limit your analysis running the risk to lose some unexpected activation. The larger map should be accepted universally.

3: dual regression and RSN splitting

Note: All the following analyses are template-dependent. You must repeat each of the following steps for every template used.

This step is the core process of all the analysis. It basically analyzes each filtered_func_data image trying to reconstruct the subject version of the independent components found in the specified template. Then it sorts these subjects component in the same order as the template and creates a 4D image (the fourth dimension is the subjects' one) for each component. Additionally to normal dual

regression script, as defined by fsl, here components are explicitly splitted into RSN of interest (chosen by the user and defined in the template script file) and stored in specific folders.
$PROJ_GROUP_ANALYSIS_DIR/melodic/dr/templ_XXX/population_XX/RSN1

In the project script you must define these variables:

```
SESS_ID=1
. $PROJ_SCRIPT_DIR/subjects_list.sh                          # load subjects list variables

SUBJECTS_INPUT_RS_DIR_NAME=resting
SUBJECTS_INPUT_RS_NAME=resting

#===== operations ========================================
DO_SORT=1
DO_SPLIT=1

. $GLOBAL_SCRIPT_DIR/melodic_templates/belgrade_controls.sh  # load template-related variables
NUM_SUBJECTS=78                                              # preserved for backward compatibility

out_population_name=controls_md_skip4vol
CTRL_SUBJECTS_DIR=/. . . /belgrade_controls/subjects # when using subjects from different projects
#=========================================
```

a) dual regression over a specific population (sort)

In this stage you also define which kind of analysis you will do, mainly which population(s) you will investigate and/or compare. For example consider a 3 groups study, here you define if compare groups A vs B vs C, A vs B, A vs C or B vs C. for each of this comparison you will create a specific folder

PROJ_GROUP_ANALYSIS_DIR/melodic/dr/templ_XXX/population_XX

This step basically sorts subjects IC according to template schema. It creates #IC files dr_stage2_ic00XX.nii.gz containing one volume for each subject.

```
DR_DIR=$PROJ_GROUP_ANALYSIS_DIR/melodic/dr/templ_$template_name/$out_population_name

if [ $DO_SORT -eq 1 ]
then
  mkdir -p $DR_DIR
  filelist=$DR_DIR/.filelist_$out_population_name

  echo "creating file lists"
  for SUBJ_NAME in ${arr_controls_md[@]}
  do
    echo
"$CTRL_SUBJECTS_DIR/$SUBJ_NAME/s$SESS_ID/$SUBJECTS_INPUT_RS_DIR_NAME/$SUBJECTS_INPUT_RS_NAME.ica/reg_st
andard/filtered_func_data_skip4vol" >> $filelist
  done

  echo "creating file lists"
  for SUBJ_NAME in ${arr_md_corr[@]}
  do
    . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
    echo
"$SUBJECT_DIR/$SUBJECTS_INPUT_RS_DIR_NAME/$SUBJECTS_INPUT_RS_NAME.ica/reg_standard/filtered_func_data_s
kip4vol" >> $filelist
  done

  echo "start DR SORT !!"
  . $GLOBAL_GROUP_SCRIPT_DIR/dual_regression_sort.sh $TEMPLATE_MELODIC_IC 1 $DR_DIR `cat $filelist`
fi
```

b) components split

Within a population folder (eg. /../templ_XXX/A_B), it creates a specific folder for each RSN defined in the arr_IC_labels, variable defined in the template script file, and merge all subjects RSN in a 4D file.

PROJ_GROUP_ANALYSIS_DIR/melodic/dr/templ_XXX/population_XX /RSN1/dr_stage2_ic0000.nii.gz

which will be later used by randomise to perform statistical analysis

```
if [ $DO_SPLIT -eq 1 ]
then
  echo "start DR SPLIT 2 SINGLE ICs !!"
  . $GLOBAL_GROUP_SCRIPT_DIR/dual_regression_split2singleIC.sh $TEMPLATE_MELODIC_IC $DR_DIR $DR_DIR
    $NUM_SUBJECTS "$str_pruning_ic_id" "$str_arr_IC_labels"
fi
```

4: statistical analysis

This step consists in performing the statistical analysis, matching the 4D subject data of each RSN against a specific General Linear Model. Hence, you need to define:

- a GLM model
- which RSN networks investigate
- the population folder previously created
- the number of CPU and permutations.
- the mask

The processing steps are:

a) create GLM models, and verify that model.con and model.mat are present. These two files will be searched for by the script once you provide the model file path and name (**without extension**).
b) (**optional**) you can calculate the gray matter 4D file used to correct for gray matter differences
c) do randomize in selected networks

If you want to execute the same GLM to different RSN folder, call the following script

$GLOBAL_GROUP_SCRIPT_DIR/dual_regression_randomize_singleIC_multiple_folders.sh

You will obtain your results in => input_folder/analysis_name

```
NUM_CPU=2
NUM_PERM=5000
EXECUTE_STATS_SH=$GLOBAL_GROUP_SCRIPT_DIR/dual_regression_randomize_singleIC_multiple_folders.sh
#-------------------------------------------------------------------------------------------------
. $GLOBAL_SCRIPT_DIR/melodic_templates/belgrade_controls.sh # define templates variables

in_population_name=dyt_b_st_wc_skip4vol             # as defined in step3 (sorting) $out_population_name
in_GLM_FILE=$PROJ_SCRIPT_DIR/glm/b_st_wc_x_age      # as defined by GLM program
out_analysis_name=dyt_b_st_wc_maskrsn               # group_analysis/melodic/dr/$in_population_name
                                                      /RSN_LABEL/$out_analysis_name
```

```
ANALYSIS_OUTPUT_DIR_ROOT=$PROJ_GROUP_ANALYSIS_DIR/melodic/dr/templ_$template_name/$in_population_name
# do STATS !!
#arr_IC_labels=(DMN)   # remove the "#" to restrict analysis to some networks, or change analysis order

str_folders="$ANALYSIS_OUTPUT_DIR_ROOT/${arr_IC_labels[0]}"
for ic in ${arr_IC_labels[@]:1}
do
  str_folders="$str_folders $ANALYSIS_OUTPUT_DIR_ROOT/$ic"
done

# use default dual regression derived mask
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_STATS_SH "$str_folders" $PROJ_DIR
-model $GLM_FILE -nperm $NUM_PERM -odn $out_analysis_name
# specify a file mask
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_STATS_SH "$str_folders" $PROJ_DIR
-model $GLM_FILE -nperm $NUM_PERM -odn $out_analysis_name -maskf $path_to_specific_mask
# specify a folder which must  contain several files called   mask_RSNLABEL.nii.gz
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_STATS_SH "$str_folders" $PROJ_DIR
-model $GLM_FILE -nperm $NUM_PERM -odn $out_analysis_name -maskd $TEMPLATE_MASK_FOLDER
# GM correction... insert GLM column and path to gm demeaned 4d_file.  N.B. if 2 want to use a default
mask: write "mask"
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_STATS_SH "$str_folders" $PROJ_DIR
-model $GLM_FILE -nperm $NUM_PERM -odn $out_analysis_name -vxl 3 -vxf $path_to_gm_demeaned_4d_file
wait
```

It assumes that:

1) input_folder (#1) contains the input path with the last folder corresponding to the RSN label.
thus extract last folder name and use it as prefix for creating output dr_stage3 files.

2) by default it masks the analysis using $INPUT_DIR/mask.nii.gz . Then it's possible to define:

-maskf: uses a specific file as mask
-maskd: specify a folder that **must** contain "mask_$RSN_LABEL.nii.gz

The output of this step is a set of corrected and uncorrected zstat image for each of the contrasts present in the GLM. The results file name is composed by:
RSN_LABEL"_"GLM_name_masktype

5: results visualization

A proper global script called: show_dr_results_in_singleIC_subfolders.sh

is available to search and visualize all the analysis results, contained within a RSN folder, which pass the requested level of significance. It also calculates the activation parameters by mean of the FSL *cluster* command, which calculate the position and the Z score of both maxima and centre-of-gravity. The script needs:

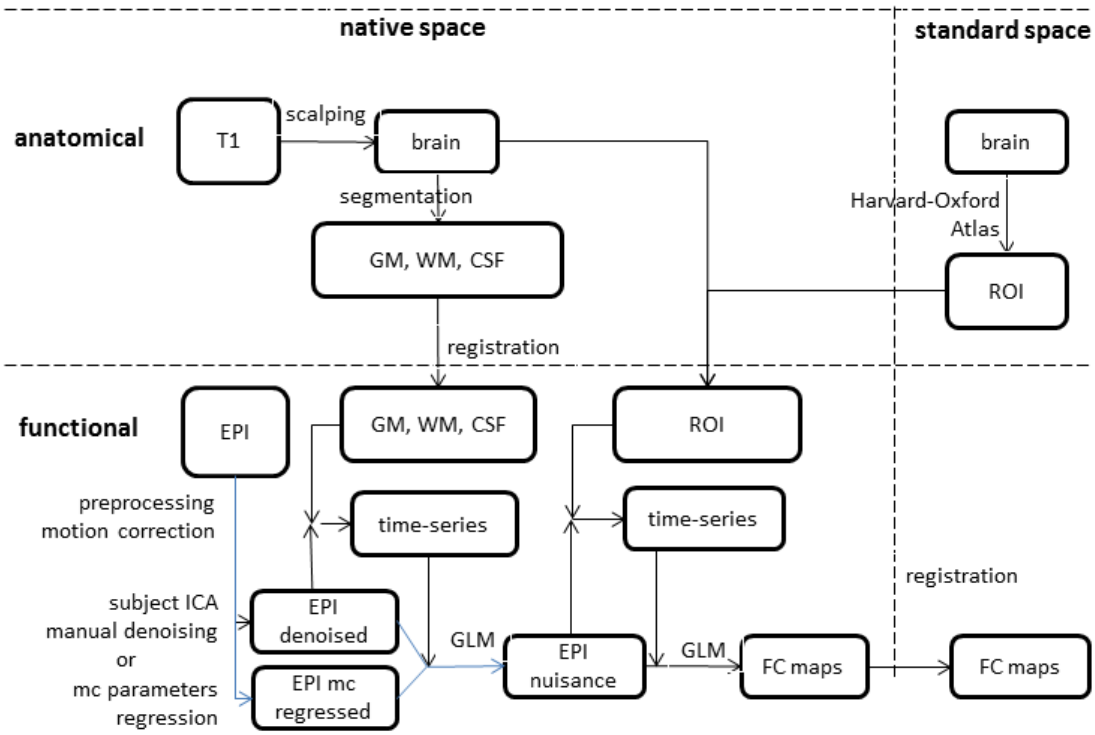
- the input folder
- the type of searched images (corrected or uncorrected)
- the name of the output text file where it stores the results.
- the background image
- the requested significance value
- the type of searched images.

```
#================================================
SHOW_IMAGE=1
WRITE_ONLY_SIGNIFICANT=1
THRESH_VALUE="0.95"
stats_type="corrp"

. $GLOBAL_SCRIPT_DIR/melodic_templates/controls_belgrade18_cab45_earlypd_skip4vol.sh
POPULATION_DIR=controls28_pd66_denoised

# results file is written by default in $MELODIC_POPULATION_DIR/results/RSN_$ICLABEL
#=========================================================================
MELODIC_POPULATION_DIR=$PROJ_GROUP_ANALYSIS_DIR/melodic/dr/templ_$template_name/$POPULATION_DIR
for IC_NAME in ${arr_IC_labels[@]}
do
  INPUT_FOLDER=$MELODIC_POPULATION_DIR/$IC_NAME
  . $GLOBAL_SCRIPT_DIR/utility/show_dr_results_in_singleIC_subfolders.sh $INPUT_FOLDER -bgimg
$TEMPLATE_BG_IMAGE -ifn "$stats_type" -shimg $SHOW_IMAGE -wrsign $WRITE_ONLY_SIGNIFICANT -thr
$THRESH_VALUE
done
```

SBFC : Seed-based functional connectivity with FEAT

The melodic package allows you to evaluate the **whole-brain** functional connectivity among one or more region-of-interest (ROI) and the rest of the brain. The output of this method are functional connectivity maps (FC), at either individual or group level, representing those voxels whom bold signal time-series (their temporal evolution) are correlated with the ROI's one. There are two possible approaches to pre-process the data, a) calculate motion correction and regress out motion effect over the data using the FEAT module, or b) perform a subject-level MELODIC analysis and manually denoise movement (and non-movement) related components. Moreover, before calculating the functional connectivity of a ROI with the rest of the brain, it is necessary to regress out the confound signals generated by white matter, csf and the whole brain, then subject-level FC maps can be generated.

The processing pipeline include: o) subject welcome, i) subject pre-processing with either FEAT or MELODIC, ii) confounds signals regression, iii) roi creation, iv) single/multiple roi(s) functional connectivity, v) group-level statistical analysis.



1-2: motion pre-processing and nuisance signal regression

The step 1 and 2 described in the analysis pipeline are performed by a single script. Nevertheless, there are two different approaches for doing this step, which regards the way the movement-related artifacts are removed from the analysis. In the conventional approach (as used for example by the fc1000 projects) you can use FEAT to calculate motion-correction and add the calculated motion parameters to multiple regression GLM that remove their effect from the data. The second approach involves instead the execution of a subject-level MELODIC and a manual denoising using the regfilt function. Both these steps require the presence of two template fsf files which contain some general settings valid for all the subjects.
After having corrected for motion, the effect of the nuisance signals must be regressed out by the data. This can be accomplished with the FEAT module using the output of the previous step. Since such correction is better performed in the native space, WM, CSF, and whole brain masks derived from T1 segmentation must be coregistered to epi native space. A proper script has been designed to perform these steps. The final output of this analysis is a residual 4D files which contains the bold signal after having regressed out the effect of movement artifacts and confound signals. Such file will be stored in $RSFC_DIR and be called (by default) nuisance_10000.nii.gz

a: Motion parameters regression with FEAT

Template creation:

Open a FEAT window and select First-level analysis & Pre-stats + Stats

Data tab:

- the TR value of the sequence
- the high-pass filter cutoff (between 100-150)

“Select 4D data” and “Output directory” will be overwritten by the script.

Pre-stats tab:

you must select:
Motion correction: MCFLIRT
BET brain extraction
Spatial Smoothing FWHM: 5/6 mm
Temporal Filtering: Highpass

Stats tab:

Select:
Use FILM prewhitening
Add motion parameters to model

Press: “Full model setup”, Create a model with a single dummy EV, selecting as Basic shape: Empty (all zeros) and one contrast with filled with an “1”.

Registration:

Keep unchecked.

Usage

A global multi-threaded script, called rsfc_motion_nuisance_feat.sh, is available to perform such analysis.
You have to define the number of CPU, the script name, the subjects array (in string version). The script will i) perform a FEAT to regress out the motion parameters calculated with MCFLIRT, ii) extract their mean time-series, writing the corresponding text files in the `$RSFC_DIR/series` folder.

```
. $PROJ_SCRIPT_DIR/subjects_list.sh

NUM_CPU=1
EXECUTE_SH=$GLOBAL_SCRIPT_DIR/process_subject/rsfc_motion_nuisance_feat.sh
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$str_arr_subj" $PROJ_DIR
```

Optionally you can decide to process a different *data* from the standard one. In order to do this you can add a further parameter. The script will use the $RS_DIR/$INPUT_NAME.

```
ALTERNATIVE_INPUT_NAME=”resting_skip4vol”
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$str_arr_subj" $PROJ_DIR
$ALTERNATIVE_INPUT_NAME
```

b: Melodic denoising

Template creation:

Open a Melodic window

Data / Pre-stats / Registration tabs :

Same as method a).

Stats tab:

Select:
Variance-normalize timecourses
Automatic dimensionality estimation.
Single-session ICA

Post-stats tab:

Select:
Threshold IC maps 0.5
Background image: Mean highres

NOTE: After having saved the fsf template, user must manually edit such file, modifying the TE value of the sequence.

1b: manual denoising after MELODIC

Instead of regressing out the motion parameters, for example when you suspect the presence of strong artifact, either related or not to head movements, it is possible to perform a deeper artifact removal procedure using the MELODIC package. The procedure is realized by executing a single-subject melodic processing, visually inspecting its output, taking note of the artefactual components id and invoking the fsl_regfilt command which remove those components from the signal and create a denoised file.

Subject-level melodic is implemented through a multi-threated global script, in the project script you must define these information:
- number of CPU to be used
- the global script ($GLOBAL_SUBJECT_SCRIPT_DIR/execute_subject_melodic.sh)
- fsf template previously created
- subjects' list string name (variable defined in $PROJ_SCRIPT_DIR/subjects_list.sh)

```
. $PROJ_SCRIPT_DIR/subjects_list.sh
EXECUTE_SH=$GLOBAL_SUBJECT_SCRIPT_DIR/execute_subject_melodic.sh
melodic_fsf_template=$PROJ_SCRIPT_DIR/glm/singlesubj_melodic
declare -i NUM_CPU=2
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -model
$melodic_fsf_template
```

The output of such analysis is a folder (SUBJECTS_DIR/SUBJ_NAME/resting/resting.ica) containing the subject-level analysis of resting state data. In the file ./filtered_func_data.ica/report.html you find the html page showing the calculated independent component.
More detail on melodic analysis can be found in the "melodic_methods.doc" user guide.
In order to denoise your subjects there are two approaches:

- simply correct the rs data without changing its final name (substitute the original file which is in turn renamed as filtered_func_data_original.nii.gz)
- preserve original file name and create a denoised version (filtered_func_data_denoised)

The former is used when you plan to analyze original data and you had just to correct the data of few subjects. On the contrary, the latter approach is used when you plan to denoise all subjects data, thus it creates a reg_standard_denoised folder, later used by dual-regression analysis, for each subject.

Note
When you will correct just few subjects, you should be very conservative: in case of doubt keep the IC, it may hide some good signal, and delete only those IC related to subjects movements, which highly differs between subjects. If you instead plan to denoise all the subjects, you can decide to remove other structured noise like those related to cardiac impulse, blood flow and scanner artifact, which are present in each subjects. In fact, here data won't be analyzed with group melodic (able to individuate IC pattern present in all the subjects), so such kind of artifact might be here removed

- **Usage**

In order to proceed with such analysis user must use a different function respect to normal SBFC pre-processing. You can define the input Compared to the previous mode, in order to compose the input file name, you must specify the melodic ICA output dir (`ICA_DIR_NAME`) and the denoised file name (`INPUT_IMAGE_NAME`). That will be used as follows:

```
ICA_DIR=$RS_DIR/$ICA_DIR_NAME
INPUT_IMAGE=$ICA_DIR/$INPUT_IMAGE_NAME.nii.gz
```

Moreover, a third parameter must be specified (`OUTPUT_POSTFIX_NAME`) in order to:

1) select a different FEAT output folder name to discriminate this analysis from standard one (e.g. not involving denoised data)
2) append the output WM, CSF, BRAIN time series names.

   ```
   csf/wm/global"_"$OUTPUT_POSTFIX_NAME"_ts.txt"
   ```

3) append output nuisance file (`$RSFC_DIR/nuisance"_"$OUTPUT_POSTFIX_NAME"_10000".nii.gz`)

An example of this call is

```
. $PROJ_SCRIPT_DIR/subjects_list.sh
SESS_ID=1
NUM_CPU=1
EXECUTE_SH=$GLOBAL_SCRIPT_DIR/process_subject/rsfc_nuisance_from_feat.sh

# reads /SUBJ_NAME/resting/resting.ica/filtered_func_data_denoised.nii.gz, writes
$RSFC_DIR/nuisance_denoised_10000.nii.gz
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$str_arr_subj" $PROJ_DIR -idn
resting.ica -ifn filtered_func_data_denoised -odn "denoised"
```

3: ROI creation

There are basically two situations: a) roi derives from group analysis results, b) they are obtained in the subjects native space (most commonly the anatomical one). In both cases, ROI must be registered to subject native space.
The subject preprocessing step created all kinds of cross-modal linear and nonlinear registration, so user just have to simply apply those transformation to starting rois.

4: Functional connectivity maps

This is the final subject-level step, which is performed again with a FEAT analysis. The global script that will implement this process needs a template fsf file that can be created as following.

Usage

There are 2 possible multi-threaded scripts that allows to

1) calculate R-roi FEATs of one roi over S-subjects => rsfc_multiple_subject_several_1roi_feat
2) calculate one FEAT of R-rois over S-subjects => rsfc_multiple_subject_1multiroi_feat

Each of them are multi-threaded scripts, which extracts the ROI timeseries from the input image (usually $RSFC_DIR/nuisance_10000) using the input roi as binary masks. Input roi are expected to be stored in a subfolder of $ROI_DIR. Although this process is usually done in the EPI space, in order to be more versatile, the reg_epi subfolder must be specified in the project script. The script will append the relative path of the input roi mask (reg_epi/mask_t_thal_epi.nii.gz) to $ROI_DIR

calculate R-roi FEATs of one roi over S-subjects

"str_arr_subjects" usual string containing the list of subjects
$PROJ_DIR: usual project directory
-model: full path of alternative model, the default one used by the script is normally the right one
-ifn: ALTERNATIVE_INPUT_FILE_NAME. file stored in $RSFC_DIR (e.g. nuisance_denoised)
-son: OUTPUT_SERIES_POSTFIX_NAME, name appended to rois timeseries
<….> At the end of these parameters, you must specify all the ROIs names.

The final OUTPUT feat folder name is defined in this way:
feat_$ROINAME"_"$OUTPUT_SERIES_POSTFIX_NAME

This is an example.

```
SESS_ID=1
NUM_CPU=2
EXECUTE_SH=$GLOBAL_SCRIPT_DIR/process_subject/rsfc_multiple_subject_several_1roi_feat.sh

# base name of ROI: final name used by the script will be $ROI_DIR/reg_epi/mask_ROINAME_epi.nii.gz
declare -a arr_roi=(l_caudate_hos_fsl l_pallidum_hos_fsl l_putamen_hos_fsl l_thalamus_hos_fsl)

ALTERNATIVE_TEMPL_FSF=$PROJ_SCRIPT_DIR/glm/templates/template_feat_roi.fsf  # can be omitted...as it is
already the default template used by the script

# alternative call: define input file name and output series postfix name

OUTPUT_DIR_NAME2=roi_left_caud_pall_put_thal_ortho_denoised
ALTERNATIVE_INPUT_NUISANCE_FILE="nuisance_denoised_10000 "
OUTPUT_SERIES_POSTFIX_NAME="denoised" # "skip4vol"
#================================================================================
declare -a final_roi=()
declare -i cnt=0

for roi in ${arr_roi[@]};
do
  final_roi[cnt]=reg_epi/mask_$roi"_epi.nii.gz"
```

```
  cnt=$cnt+1
done

## !!!!! the OUTPUT feat folder name is defined in this way:
feat_$ROINAME"_"$OUTPUT_SERIES_POSTFIX_NAME

# default call: read $RSFC_DIR/nuisance_10000.nii.gz and use template_feat_roi.fsf
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR
${final_roi[@]} # -model $ALTERNATIVE_TEMPL_FSF if u want a special feat setup
# default call but with a custom template : read $RSFC_DIR/nuisance_10000.nii.gz and use
ALTERNATIVE_TEMPL_FSF
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -model
$ALTERNATIVE_TEMPL_FSF ${final_roi[@]}
# alternative call: read $RSFC_DIR/nuisance_denoised_10000.nii.gz
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -ifn
$ALTERNATIVE_INPUT_NUISANCE_FILE -son $OUTPUT_SERIES_POSTFIX_NAME ${final_roi[@]}
wait
```

calculate one FEAT of R-rois over S-subjects

"str_arr_subjects" usual string containing the list of subjects
$PROJ_DIR: usual project directory
-model: full path of model templates
-odn: OUTPUT_DIR_NAME, name appended to $PROJ_GROUP_ANALYSIS_DIR/sbfc/
-ifn: ALTERNATIVE_INPUT_FILE_NAME. file stored in $RSFC_DIR (e.g. nuisance_denoised)
-son: OUTPUT_SERIES_POSTFIX_NAME, name appended to rois timeseries
<….> At the end of these parameters, you must specify all the ROIs names.

This is an example.

```
SESS_ID=1
NUM_CPU=2
EXECUTE_SH=$GLOBAL_SCRIPT_DIR/process_subject/rsfc_multiple_subject_1multiroi_feat.sh

# base name of ROI: final name used by the script will be $ROI_DIR/reg_epi/mask_ROINAME_epi.nii.gz
declare -a arr_roi=(l_caudate_hos_fsl l_pallidum_hos_fsl l_putamen_hos_fsl l_thalamus_hos_fsl)

TEMPL_FSF=$PROJ_SCRIPT_DIR/glm/templates/template_feat_4roi_ortho

# standard call: define output dir name
OUTPUT_DIR_NAME=roi_left_caud_pall_put_thal_ortho

# alternative call: define output dir name, input file name and output series postfix name
OUTPUT_SERIES_POSTFIX_NAME="denoised"
ALTERNATIVE_OUTPUT_DIR_NAME=roi_left_caud_pall_put_thal_ortho_denoised
ALTERNATIVE_INPUT_NUISANCE_FILE="nuisance_denoised_10000 "
OUTPUT_SERIES_POSTFIX_NAME="denoised"
#================================================================================
declare -a final_roi=()
declare -i cnt=0

for roi in ${arr_roi[@]};
do
  final_roi[cnt]=reg_epi/mask_$roi"_epi.nii.gz"
  cnt=$cnt+1
done

# default call: read $RSFC_DIR/nuisance_10000.nii.gz
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -model
$TEMPL_FSF -odn $OUTPUT_DIR_NAME ${final_roi[@]}
# default call: read $RSFC_DIR/nuisance_denoised_10000.nii.gz
```

```
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$arr_subj" $PROJ_DIR -ifn
$ALTERNATIVE_INPUT_NUISANCE_FILE -model $TEMPL_FSF -odn $ALTERNATIVE_OUTPUT_DIR_NAME -son
$OUTPUT_SERIES_POSTFIX_NAME ${final_roi[@]}
wait
```

5: Group level analysis

Group analysis is performed with another FEAT analysis. User must first create one or more FEAT fsf template files, indicating the threshold masking option and the GLM group models.
Then two global group scripts are available performing:

- Test multiple GLM models over one 1st-level analysis
- Execute the same GLM model over N 1st-level analyses

Test multiple GLM models over one 1st-level analysis

User must define an array of FEAT fsf file (containing masking options and group GLMs), provide an 1st level input folder name and an output group folder name,

"str_arr_fsf" string containing the full path of several fsf files.
$PROJ_DIR: usual project directory
-odp: OUTPUT_DIR, full path of output folder
-ncope: number of copes contained in the first level folders.
<….> At the end of these parameters, you must specify the full path of all the first level analyses

```
SESS_ID=1
NUM_CPU=1
EXECUTE_SH=$GLOBAL_SCRIPT_DIR/process_group/rsfc_multiple_model_group_feat.sh

. $PROJ_SCRIPT_DIR/subjects_list.sh

INPUT_1stlevel_DIR="roi_right_caud_pall_put_thal_ortho_denoised"

OUTPUT_DIR=$PROJ_GROUP_ANALYSIS_DIR/sbfc/$INPUT_1stlevel_DIR
declare -a arr_fsf_templates=($PROJ_SCRIPT_DIR/glm/templates/groupfeat_ctrl28_treated45_naive21_maskgm)
str_arr_fsf_templates=`echo ${arr_fsf_templates[@]}`

CONTROLS_SUBJ_DIR=/media/data/MRI/projects/CAB/fsl_resting_belgrade_controls/subjects

# create 1st level feat dir list
first_level_feat_paths=""

for SUBJ_NAME in ${arr_controls28[@]}
do
  first_level_feat_paths="$first_level_feat_paths
$CONTROLS_SUBJ_DIR/$SUBJ_NAME/s$SESS_ID/resting/fc/feat/$INPUT_1stlevel_DIR"
done

for SUBJ_NAME in ${arr_treated45[@]}
do
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh first_level_feat_paths="$first_level_feat_paths
$RSFC_DIR/feat/$INPUT_1stlevel_DIR"
done

for SUBJ_NAME in ${arr_naive21[@]}
do
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
   first_level_feat_paths="$first_level_feat_paths $RSFC_DIR/feat/$INPUT_1stlevel_DIR"
```

```
done

#============================================================================
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$str_arr_fsf_templates"
$PROJ_DIR -odp $OUTPUT_DIR -ncope 8 $first_level_feat_paths
wait
```

Execute the same GLM model over N 1st-level analyses

"str_arr_1stlvl_feat_name" string containing the name of the 1st level feat analysis
$PROJ_DIR: usual project directory
-odp: OUTPUT_DIR, full path of output folder
-ncope: number of copes contained in the first level folders.
-model: full path of the template fsf file
<….> At the end of these parameters, you must specify the subjects' directory that contains the 1st level analyses subfolders

```
SESS_ID=1
NUM_CPU=1
EXECUTE_SH=$GLOBAL_SCRIPT_DIR/process_group/rsfc_multiple_roi_group_feat.sh

. $PROJ_SCRIPT_DIR/subjects_list.sh

declare -a arr_1stlevel_input_roi=(roi_right_caud roi_right_pall roi_right_put roi_right_thal)
str_arr_1stlevel_input_roi=`echo ${arr_1stlevel_input_roi[@]}`

OUTPUT_DIR=$PROJ_GROUP_ANALYSIS_DIR/rsfc/ctrl_treated_naive
fsf_template=$PROJ_SCRIPT_DIR/glm/templates/groupfeat_ctrl28_treated45_naive21_maskgm

CONTROLS_SUBJ_DIR=/media/data/MRI/projects/CAB/fsl_resting_belgrade_controls/subjects

# create 1st level feat roots list
first_level_feat_roots=""

for SUBJ_NAME in ${arr_controls28[@]}
do
  first_level_feat_roots="$first_level_feat_roots
$CONTROLS_SUBJ_DIR/$SUBJ_NAME/s$SESS_ID/resting/fc/feat"
done

for SUBJ_NAME in ${arr_treated45[@]}
do
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
  first_level_feat_roots="first_level_feat_roots $RSFC_DIR/feat"
done

for SUBJ_NAME in ${arr_naive21[@]}
do
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
  first_level_feat_roots="$first_level_feat_roots $RSFC_DIR/feat"
done

#============================================================================
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $EXECUTE_SH "$str_arr_1stlevel_input_roi"
$PROJ_DIR -model $fsf_template -odp $OUTPUT_DIR -ncope 8 $first_level_feat_roots
wait
```

Tractography

Probtrackx

A main multi-threaded script, called dti_multiple_subject_probrtackx.sh is available to perform probtrackx analysis. Through its input parameters is possible to define all (most of) the standard operations, that is, user can define seed and stop masks, define several waypoints and the file containing target images for Classification Targets approach.

The script allows user to define absolute or relative (to $ROI_DIR) paths separately for all the involved images. That is you can set some path as absolute, some other as relative. For example, if dti images are in the standard space, you can have co-registered the seed using subject's T1 and T2 and thus store them in reg_dti folder but you may use a common image for stop mask.

Three further operations are included:

a) If Classification Target procedure is selected, the function "find_the_biggest" is automatically called.
b) NORMALIZATION
   By default, the script create a normalized version of fdt_paths file (fdt_paths_norm) by dividing the original file by the number of tracts contained in the waypoint file
c) THRESHOLDING
   If you provide a further parameters (-thrP) followed by a comma-separated list of N values, the script perform N thrP thresholding of fdt_paths_norm file and move the resulting N masks in ROI_DIR/reg_dti folder.

The input parameters of the script are coded as follows:

```
-idn)          INPUT_MERGED_DIR, appended to $BEDPOSTX_DIR
-odn)          OUTPUT_DIR_NAME, appended to $PROBTRACKX_DIR
-mask)         mask file path relative to $ROI_DIR
-maskp)        full path of mask file
-seed)         seed file path relative to $ROI_DIR
-seedp)        full path of seed file
-stop)         stop file path relative to $ROI_DIR
-stopp)        full path of stop file
-target)       TARGET_FILE=$2
-targetp)      full path of target
-wp)           define that waypoints file listed must be considered full paths
-thrP)         "20,30,40" list of -thrP values to be applied to fdt_paths_norm

All the remaining parameters are the list of waypoints file to be used, if -wp is set, they are full
paths, otherwise they are path relative to $ROI_DIR

declare -a WP_FILES=( "$@" )
```

Here follow an example of Classification target analysis:

```
SESS_ID=1
. $PROJ_SCRIPT_DIR/subjects_list.sh
BASH_SCRIPT=$GLOBAL_SUBJECT_SCRIPT_DIR/dti_multiple_subject_probtrackx.sh
NUM_CPU=1
#================================================

DO_OVERWRITE_TARGET_FILE=0

SEED_IMAGE_r=reg_dti/mask_R_Thal_dti.nii.gz
SEED_IMAGE_l=reg_dti/mask_L_Thal_dti.nii.gz

STOP_IMAGE_r=$GLOBAL_DATA_TEMPLATES/gray_matter/MNI152_T1_2mm_brain_left.nii.gz
STOP_IMAGE_l=$GLOBAL_DATA_TEMPLATES/gray_matter/MNI152_T1_2mm_brain_right.nii.gz

OUTPUT_DIR_NAME_r=r_thalamus_to_8lobes
OUTPUT_DIR_NAME_l=l_thalamus_to_8lobes

# ----- target list ------------------------------------------------------------------------------
input_roi_dir=$GLOBAL_SCRIPT_DIR/data_templates/roi/2mm/lobes
target_list_file_r=$input_roi_dir/r_8lobes_list.txt
target_list_file_l=$input_roi_dir/l_8lobes_list.txt
declare -a target_image_list_r=(r_mask_1_pfc r_mask_2_premotor r_mask_3_precentral r_mask_4_postcentral
r_mask_5_parietal_lobes r_mask_6_temporal_lobes r_mask_7_tempoccip r_mask_8_occipital_lobes)
declare -a target_image_list_l=(l_mask_1_pfc l_mask_2_premotor l_mask_3_precentral l_mask_4_postcentral
l_mask_5_parietal_lobes l_mask_6_temporal_lobes l_mask_7_tempoccip l_mask_8_occipital_lobes)

if [ ! -f $target_list_file_r -o $DO_OVERWRITE_TARGET_FILE -eq 1 ]; then
  echo "$input_roi_dir/${target_image_list_r[0]}.nii.gz" > $target_list_file_r
  for f in ${target_image_list_r[@]:1}; do echo "$input_roi_dir/$f.nii.gz" >> $target_list_file_r; done
fi

if [ ! -f $target_list_file_l -o $DO_OVERWRITE_TARGET_FILE -eq 1 ]; then
  echo "$input_roi_dir/${target_image_list_l[0]}.nii.gz" > $target_list_file_l
  for f in ${target_image_list_l[@]:1}; do echo "$input_roi_dir/$f.nii.gz" >> $target_list_file_l; done
fi
#============================================================================================================
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $BASH_SCRIPT "$str_arr_pd65" $PROJ_DIR -odn
$OUTPUT_DIR_NAME_r -maskp "mask" -seed $SEED_IMAGE_r -targetp $target_list_file_r -stopp $STOP_IMAGE_r
wait
. $MULTICORE_SCRIPT_DIR/define_thread_processes.sh $NUM_CPU $BASH_SCRIPT "$str_arr_pd65" $PROJ_DIR -odn
$OUTPUT_DIR_NAME_l -maskp "mask" -seed $SEED_IMAGE_l -targetp $target_list_file_l -stopp $STOP_IMAGE_l
wait

declare -i cnt=0
for SUBJ_NAME in ${arr_pd65[@]}
do
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
  cnt=1
  for ROI_NAME in ${target_image_list_r[@]}
  do
    $FSLDIR/bin/fslmaths $PROBTRACKX_DIR/r_thalamus_to_8lobes/biggest -thr $cnt -uthr $cnt -bin
$ROI_DIR/reg_dti/$ROI_NAME
    cnt=$cnt+1
  done
  cnt=1
  for ROI_NAME in ${target_image_list_l[@]}
  do
    $FSLDIR/bin/fslmaths $PROBTRACKX_DIR/l_thalamus_to_8lobes/biggest -thr $cnt -uthr $cnt -bin
$ROI_DIR/reg_dti/$ROI_NAME
    cnt=$cnt+1
  done
done
```

Calculate mean tract dtifit values

One of two main application of tractography is the possibility to use the reconstructed tract to calculate its mean FA, MD etc values.

```
. $PROJ_SCRIPT_DIR/subjects_list.sh

thrPvalue=20
OUTPUT_DIR_NAME_l=l_cst_ped2mi_2wp_1s
OUTPUT_DIR_NAME_r=r_cst_ped2mi_2wp_1s
mask_thrp_file_name_r=mask_$OUTPUT_DIR_NAME_r"_P"$thrPvalue.nii.gz
mask_thrp_file_name_l=mask_$OUTPUT_DIR_NAME_l"_P"$thrPvalue.nii.gz

for SUBJ_NAME in ${arr_ela_dti[@]}
do
  echo $SUBJ_NAME
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh

  # calculate mean FA/MD in CST
  [ ! -f $ROI_DIR/reg_dti/$mask_thrp_file_name_l ] && $FSLDIR/bin/fslmaths
$PROBTRACKX_DIR/$OUTPUT_DIR_NAME_l/fdt_paths_norm.nii.gz -thrP $thrPvalue
$ROI_DIR/reg_dti/$mask_thrp_file_name_l

  $FSLDIR/bin/fslmeants -i $DTI_DIR/$DTI_FIT_LABEL"_FA.nii" -m $ROI_DIR/reg_dti/$mask_thrp_file_name_l
-o $ROI_DIR/reg_dti/FA_meants_$OUTPUT_DIR_NAME_l"_P"$thrPvalue.txt
       $FSLDIR/bin/fslmeants -i $DTI_DIR/$DTI_FIT_LABEL"_MD.nii" -m
$ROI_DIR/reg_dti/$mask_thrp_file_name_l -o
$ROI_DIR/reg_dti/MD_meants_$OUTPUT_DIR_NAME_l"_P"$thrPvalue.txt

  [ ! -f $ROI_DIR/reg_dti/$mask_thrp_file_name_r ] && $FSLDIR/bin/fslmaths
$PROBTRACKX_DIR/$OUTPUT_DIR_NAME_r/fdt_paths_norm.nii.gz -thrP $thrPvalue
$ROI_DIR/reg_dti/$mask_thrp_file_name_r

  $FSLDIR/bin/fslmeants -i $DTI_DIR/$DTI_FIT_LABEL"_FA.nii" -m $ROI_DIR/reg_dti/$mask_thrp_file_name_r
-o $ROI_DIR/reg_dti/FA_meants_$OUTPUT_DIR_NAME_r"_P"$thrPvalue.txt
       $FSLDIR/bin/fslmeants -i $DTI_DIR/$DTI_FIT_LABEL"_MD.nii" -m
$ROI_DIR/reg_dti/$mask_thrp_file_name_r -o
$ROI_DIR/reg_dti/MD_meants_$OUTPUT_DIR_NAME_r"_P"$thrPvalue.txt
done
#=================================================================================================
# collect subjects' FA & MD means and store in a single file for statistical analysis
mkdir -p $PROJ_GROUP_ANALYSIS_DIR/results
group_fa_means=$PROJ_GROUP_ANALYSIS_DIR/results/group_fa_means.txt
group_md_means=$PROJ_GROUP_ANALYSIS_DIR/results/group_md_means.txt

echo "subj     r_fa    l_fa" > $group_fa_means
echo "subj     r_md    l_md" > $group_md_means

for SUBJ_NAME in ${arr_ela_dti[@]}
do
  . $GLOBAL_SCRIPT_DIR/subject_init_vars.sh
  wr=$(cat $ROI_DIR/reg_dti/FA_meants_$OUTPUT_DIR_NAME_l"_P20".txt | tr -d ' ')
  wl=$(cat $ROI_DIR/reg_dti/FA_meants_$OUTPUT_DIR_NAME_r"_P20".txt | tr -d ' ')
  echo "$SUBJ_NAME    $wr     $wl" >> $group_fa_means

  wr=$(cat $ROI_DIR/reg_dti/MD_meants_$OUTPUT_DIR_NAME_r"_P20".txt | tr -d ' ')
  wl=$(cat $ROI_DIR/reg_dti/MD_meants_$OUTPUT_DIR_NAME_l"_P20".txt | tr -d ' ')
  echo "$SUBJ_NAME    $wr     $wl" >> $group_md_means
done
```