Temporal reliability of ultra-high field resting-state MRI for single-subject motor and language mapping

Paulo Branco, Daniela Seixas and São Luís Castro

Supporting material v1.0. Last updated on 29 October 2016.

The following document provides step-by-step instruction for all processing steps included in the manuscript, including example command-line codes (FSL) and Matlab scripts.
All Matlab scripts are referred in the text and available for download at
https://github.com/brancop/Brancoetal_7tesla_testretest

For conciseness, we illustrate the processing steps for the sensorimotor network. The same steps can be replicated for the language network by replacing the code with the appropriate filenames. Finally, we provide step-by-step instructions on how to prepare files in individual subject/protocols for improved flexibility. Automated processing pipelines (e.g., more than one subject/protocol) can be achieved with minor changes to the code. Throughout this document "< >" denotes a filename. Thus, please keep in mind this field should be replaced by the name of variables/filenames used if changes are made.

These scripts may be optimized and debugged in the future - please download the most recent versions. Contact brancop@me.com if any questions arise.

All analyses were performed using FSL 5.0.9, Matlab 8.3.0 (R2014a) and in Mac OS X, v10.8.5.

1. Preparation of the supporting files

A few key files should be prepared before the actual preprocessing can take place: the skull-stripped structural image, the white-matter (WM) and cerebral spinal fluid (CSF) masks, and the fieldmaps files.

1.1 Skull-stripping

Skull-stripping MP2RAGE structural images is quite challenging. For convenience, all skull-stripped structural images are made available at gibhub.com, inside the folder "Skullstripped_images". The following steps should be used:

First, multiply the uniform contrast image (UNI) by the second inversion image (INV2). This will result in a biased image with a conventional T1-weighed contrast, but without noise outside of the skull.

fslmaths <mp2rage_UNI> –mul <mp2rage_INV2> <structural> -odt float

Second, use the FSL Brain Extraction Tool (BET) with the following parameters:

Select the option "*Bias field & neck cleanup*"; activate output brain-extracted image; and activate output binary brain mask image.

The fractional intensity threshold will vary on a subject-to-subject basis; the range 0.15 to 0.35 worked best. We further recommend using positive threshold gradients to improve brain extraction in inferior parts of the brain.

Third, manually edit the mask so that it closely resembles the brain image. This can be achieved using the tool FSLview by superimposing the mask with the UNI image and using the FSLview edit functions. It may be convenient to erode this mask to remove the outer layer of skull so that it fits better with the skull image (this needs to be inspected on a subject-by-subject basis):

> *fslmaths <structural_brain_mask> -ero <structural_brain_mask_eroded>*

Finally, use this mask to extract the brain from the unbiased uniform contrast image

> *fslmaths <mp2rage_uni> –mul <structural_brain_mask_eroded> <structural_brain>*

1.2 Segmentation of white-matter (WM) and cerebral spinal fluid (CSF)

To segment the WM and CSF masks, use the FSL tool FAST with the following command:

> *fast –g <structural_brain>*

This will output 3 binary masks for each tissue type. "*structural_brain_seg*_0" for CSF, "*structural_brain*_seg_1" for gray-matter and "*structural_brain*_seg_2" for WM. Erode the masks to avoid mixing signal from different tissue types.

> *fslmaths <structural_brain_seg_0> -ero <csf_eroded>*
> *fslmaths <structural_brain_seg_2> -ero <wm_eroded>*

1.3 Preparing the fieldmap files to work in FSL

Fieldmap files should be prepared using the FSL built-in fsl_prepare_fieldmap tool. The following steps are required:

First, split the magnitude image using the fslsplit. Two files will be generated: vol0000.niigz and vol0001.nii.gz.

> *fslsplit <magnitude_image>*

Second, perform BET on the first magnitude image (vol0000.nii.gz; it typically has the best image contrast) using the default options. A fractional intensity threshold in the range of 0.4-0.5 worked best. For this step, it is critical that the brain extraction is flawless and that no skull voxels are left behind. Thus, we recommend eroding the brain extracted magnitude image, and if needed, to edit the mask manually using FSLview and the edit options.

> *fslmaths <vol0000_brain> -ero <vol0000_brain>*

Finally, use the built-in FSL tool to prepare the fieldmap files. The Delta TE value is 1.02 (cf. acquisition protocols, http://openscience.cbs.mpg.de/7t_trt/).

> *fsl_prepare_fieldmap SIEMENS <phase_image> <vol0000_brain> fmap_rads 1.02*

2. Pre-processing

2.1. Unwarping, smoothing and registration

Pre-processing is carried out using the FSL FEAT tool and supporting command-line tools. The FEAT GUI is used for simplicity.

Open the FEAT GUI and select the resting-state 4D image. Select the following options:

Data Tab. Remove the initial 3 volumes to exclude T1 saturation effects. Make sure the TR value is set to the correct value (3 seconds).

Pre-stats tab. Use MCFLIRT for motion correction; activate the b0 unwarping option. The fieldmap file is the *fmap_rads* file obtained from step 1.2. The fieldmap magnitude image is the brain extracted magnitude image also from step 1.2, *vol0000_brain*.

The effective EPI echo spacing is 0.33, the EPI TE = 17 ms, and the unwarp direction –y (cf. acquisition protocols, http://openscience.cbs.mpg.de/7t_trt/). We used the default 10% signal loss threshold option.

Select the option to perform brain extraction on the functional images and use a 3 mm FHWM smoothing kernel. Turn off the high-pass temporal filter option, as this will only be performed after denoising the data.

Registration tab. Registration to the structural image is achieved by using FLIRT with the Boundary Based Registration (BBR) option. For the main structural image, use the brain extracted structural image. Make sure it has the same base filename as the original, non-skull-stripped image as BBR requires this file to successfully perform the registration. Normalization is performed using FNIRT (turn on the option "Nonlinear transformation") with a warp resolution of 6 mm.

Click "Go".

2.2 ICA denoising

Data were denoised using ICA-AROMA (Pruim et al., 2015, https://github.com/rhr-pruim/ICA-AROMA). To perform this step, please download the package and follow the instructions provided by the authors to setup all software dependencies. Apply the automated cleaning procedure using the following command-line code:

> python 2.7 ICA_AROMA.py –feat <feat_directory> -o <feat_clean>.

We recommend visual inspection of the removed ICs for quality assurance (we did not find any problem in our analyses). The denoised 4D file will be within the selected output folder, under the filename "denoised_func_data_nonaggr.nii.gz".

2.3 Removal of WM and CSF confounds

To remove WM and CSF confounds from the data, first the WM and CSF masks need to be warped to functional space. This can be achieved by using FLIRT with the following command-line codes (the higres2example_func registration matrix and the *example_func* are inside the FEAT output folder from step 2.1, subfolder /reg):

> flirt –in <csf_ero> -ref <example_func> -init <highres2example_func.mat> –out <csf_functional_space> -applyxfm

> flirt –in <csf_ero> -ref <example_func> -init <highres2example_func.mat> –out <wm_functional_space> -applyxfm

Next, threshold and binarize both masks:

> fslmaths <csf_functional_space> –thr 0.9 –bin <csf_functional_space>
> fslmaths <wm_functional_space> –thr 0.9 –bin <wm_functional_space>

And finally, extract the average timecourse for each confound (the denoised_func_data_noaggr file is within the folder created by step 2.2):

> fslmeants –i <denoised_func_data_noaggr.nii.gz> -m <csf_functional_space> -o csf.txt
> fslmeants –i <denoised_func_data_noaggr.nii.gz> -m <wm_functional_space> -o wm.txt

Each text file will contain the timecourse of each confound. Merge the two text files such that a unique text file with two columns is created. The following Matlab code will combine the columns in a way that works in FSL:

ParametersMatrix(:,1) = load('csf.txt');
ParametersMatrix(:,2) = load('wm.txt');
save confounds.txt ParametersMatrix –ASCII

Finally, these confounds can be removed from the data through multiple regression. There are several ways of achieving this yet for convenience, we recommend using the FSL command-line tool fsl_regfilt:

> fsl_regfilt –i <*denoised_func_data_noaggr.nii.gz*> *-d <confounds.txt> -o <clean.nii.gz> -f "1,2"*

This will output the residuals of the multiple regression model, now without signal from WM and CSF.

3. Extraction of the independent components (IC)

Extraction of resting-state networks is achieved through the FSL MELODIC tool. Again, the GUI is used for simplicity.

Add the residuals from the previous step (filename *clean.nii.gz*) to the MELODIC GUI with the following options:

Data tab. Use the defaults options, including a high-pass 100 s filter cutoff.

Pre-stats tabs. Turn off all options (these have been performed previously), except high-pass temporal filtering.

Registration tab. Turn off all options, registration was already performed.

Stats tab. Leave all settings as default. Make sure Single-session ICA is selected.

Post-stats tab. Use the default threshold value (0.5 for mixture modeling), enable the option "Output full stats folder".

Click "Go".

4. Template matching procedure

The masks used for the template-matching procedure were collected from FINDLAB (Shirer et al. 2012, http://findlab.stanford.edu/functional_ROIs.html; sensorimotor network mask and the language network mask).
To perform the template matching procedure, these masks need to be warped from standard space to functional space. This can be achieved with the following command-line codes:

First, create an inverted warp registration file (standard space to structural space):

> invwarp –-ref <highres> --warp <highres2standard_warp> --out <highres2standard_warp_inv>

Then, apply this warp on the template masks:

> applywarp --ref=<example_func> --in=<sensorimotor_network> --warp= <highres2standard_warp_inv> --postmat=highres2example_func.mat –out= <sensorimotor_network_func>

Finally, threshold and binarize the mask:
> fslmaths <sensorimotor_network_func > -thr 0.5 –bin <sensorimotor_network_func>

Next, all IC maps need to be thresholded and binarized. To do this, go to the "stats" folder inside the MELODIC output folder. The provided Matlab script will binarize all images within the stats folder and output them in a newly created folder (/ICs):

[Please see matlab script Orepare_ICS_Templatematching.m]

Finally, Dice coefficients between the network masks and each resulting IC are computed to rank the ICs. Copy the template mask in functional space into the ICs folder, and run the provided Matlab script:

[Please see matlab script Template_Matching.m]

This will output the file "*classification_motor*" with 5 volumes, ordered by the spatial similarity to the template mask. These need to be inspected to select which ICS are part of a given network and which are not.
After choosing which ICs are related to a given network, split the file and multiply the masks so that a unique mask is created. This can be achieved through the following command-line commands:

> fslsplit classification_motor

This will generate files vol0000.nii.gz, vol0001.nii.gz, vol0002.nii.gz, vol0003.nii.gz and vol0004.nii.gz, corresponding to volumes 1 to 5.

After selecting the ICs, merge the masks using fslmaths. A threshold of z = 4 was used to constrain the size of the mask to a subset of the highest significant voxels. For this example, we will stack 3 masks: vol0000, vol0002 and vol0004, and the resulting mask will be called input_DR.

> fslmaths vol0000.nii.gz –thr 4 –bin temp1
> fslmaths vol0002.bii.gz –thr 4 –bin temp2
> fslmaths vol0004nii.gz –thr 4 –bin temp3
> fslmaths temp1 –add temp2 –add temp3 –bin input_DR
> rm temp1 temp2 temp3

The output from this step is the combined network mask that will be fed to dual regression so that the whole-network is re-estimated without ICA overdescompositions.

5. Dual regression

The final step to obtain the resting-state network is dual regression. This can be achieved by performing two consecutive fsl_glm calls in the command-line:

First, a spatial regression is performed to obtain the network timecourse:

> fsl_glm -i <clean.nii.gz> -d <input_DR.nii.gz> -o <sensorimotor.txt> -–demean

Next, a temporal regression is performed using the timecourse from the previous step:

> fsl_glm -i <clean.nii.gz> –d <sensorimotor.txt> -o <sensorimotor_beta> --out_z=<sensorimotor_z> –-demean

6. Preparation of files to calculate reliability metrics

Before calculating reliability metrics, all maps need to be in structural space, but at original spatial resolution (1.5 mm^3). To achieve this, use flirt to create a 1.5 mm^3 structural image:

flirt –in <structural_brain> -ref <structural_brain> –out <structural_func> -applyisoxfm 1.5 --noresampblur

Next, warp the network to low-resolution structural space:

flirt –in <sensorimotor_z> -ref <structural_func > -init <example_func2highres.mat –out < sensorimotor_zmap> --applyxfm

For the ROI statistics, two new masks were used. For the sensorimotor network, we used the Oxford-Harvard Cortical Strucutal Atlas. The masks of the Precentral and Postcentral were combined, after thresholding and binarizing the image to a minimum probability of 0.20. For the language network, we used the thirteen language ROIs from the Fedorenko et al. (2010), avaliable for download at (https://evlab.mit.edu/), combined into a single mask. These ROIs should be warped into structural space:

applywarp –ref=<structural_func> --in=<sensorimotor_ROI> --warp=<highres2standard_warp_inv>–out=<sensorimotor_ROI_struct>

6.1 Preparing files for Dice, fixed z-scores

To calculate Dice at fixed z scores, the network should be thresholded at multiple values. In this study we used thresholds in the range between 2 and 20, in steps of 0.5. To achieve this, use the provided Matlab script:

[Please see matlab script Preparefiles_dicefixedz]

6.2 File preparation for Dice, fixed size

In this manuscript, we also examined Dice coefficient when networks were restricted at a fixed size between 1000 and 20000 voxels, in steps of 1000. To perform the Dice calculation at these fixed network sizes, first we need to calculate what thresholds are required to achieve mask of a given size. There is no built-in way to perform this with FSL. The following provided Matlab script calculates the thresholds for each size, applied this threshold, and outputs the files in a new folder names "size".

[Please see matlab script Findthr_Dicesize]

Important note: due to numerical precision, in a few isolated cases it might not be possible to automatically find the correct threshold. Hence, we implemented a quality control step to ensure all masks are of the expected size. Use the provided Matlab script:

[Please see matlab script Findthr_Qualitycontrol]

Inspect the variable "matrix_qualcontrol" to make sure the mask sizes are correct (they should be ordered from 1000 to 20000 in steps of 1000). We recommend that in situations where the algorithm did not find the correct threshold, to override it manually by searching the correct threshold with the following command-line code:

fslstats <sensorimotor_zmap> -l threshold –V

The final step is to prepare the files for ROI analyses. To do so, output maps from the previous steps should be masked by the template ROI. To do this, enter the newly created size folder, copy the template mask into this folder and run the provided Matlab script:

[Please see matlab script Findthr_ROIs]

6.3 File preparation for the ICC calculation

There are several ways to perform the ICC calculation; for convenience we will report one way that works in Matlab but can also be used with any conventional statistical package.

First, extract a text file with all voxel intensities. This can be achieved by the following command:

fsl2ascii <sensorimotor_zmap> <sm_network>

This will output an ascii file with intensity values for each voxel. Due to filename conventions, FSL will output 5 zeros to the ascii file name (hence, the output name should be sm_network00000).

Next, a brain mask is required to constrain the ICC analysis to the whole-brain. Please keep in mind this structural image should be the lowres (1.5mm3) one created in the step 5.1. This can be obtained as in the previous step.

fsl2ascii <structural_func> <strucfunc>

Finally, to perform ICC restricted at ROIs, obtain an ascii file from the ROI mask:

fsl2ascii <sensorimotor_ROI_struct> <motorROI>

7. Calculating reliability metrics

So far, we showed all steps required to provide all files for a single-subject and protocol. To calculate the reliability metrics, files from all sessions must be obtained. For this next section, we will assume all files are in individual folders, organized first by the reliability metric, and further separated into protocols, as in the table below:

Main Folder	Files	Subfolder 1	Subfolder 2	Files
Dice_fixedz		Protocol 1	WB	sensorimotor_1 to _20
			ROI	sensorimotor_1 to _20
		Protocol 2	WB	sensorimotor_1 to _20
			ROI	sensorimotor_1 to _20
		Protocol 3	WB	sensorimotor_1 to _20
			ROI	sensorimotor_1 to _20
Dice_fixed_size		Protocol 1	WB	sensorimotor_1 to _37
			ROI	sensorimotor_1 to _37
		Protocol 2	WB	sensorimotor_1 to _37
			ROI	sensorimotor_1 to _37
		Protocol 3	WB	sensorimotor_1 to _37
			ROI	sensorimotor_1 to _37
ICC	motorROI00000 strucfunc00000	Protocol	WB	sm_network00000
			ROI	sm_network00000
		Protocol2	WB	sm_network00000
			ROI	sm_network00000
		Protocol3	WB	sm_network00000
			ROI	sm_network00000

Note: to avoid file duplications, for the ICC calculation place the structural and ROI files in the main directory

7.1 Calculating Dice at fixed z scores

To obtain the Dice matrices for fixed z scores, run the provided Matlab script:

[Please see matlab script Calculate_Dice_fixedz]

The matrices intrasession and intersession will have the Dice coefficient values for each combination of thresholds, for each comparison type (intrasession and intersession reliability, respectively). For ROI statistics, the variable names are intrasession_ROI and intersession_ROI.

7.2 Calculating Dice at fixed mask sizes

To calculate the Dice coefficients at fixed mask sizes, run the provided Matlab script:

[Please see matlab script Calculate_Dice_fixedsize]

Again, the matrices intrasession and intersession will have the Dice coefficient values for each size, from 1 to 20 (1000 to 20000 voxels), and for each comparison type (intrasession and intersession reliability, respectively). For ROI statistics, the variable names are intrasession_ROI and intersession_ROI.

7.3 Calculating ICC

To perform ICC, we used two matlab functions available at (https://www.mathworks.com/matlabcentral/fileexchange/22099-intraclass-correlation-coefficient--icc-).

These functions provide 6 ICC alternatives. The ICC variant used in this paper (two-way random ICC for absolute agreement) is implemented in these functions as the "A-1" option. Download the functions and make sure they are properly set in the Matlab directory path.

Finally, to perform the ICC calculation, use the following Matlab script.

[Please see matlab script Calculate_ICC]

The intrassession and intersession variables will contain the ICC value (column 1), the F value (column 2) and the corresponding p-value (column 3) for intrasession and intersession reliability, respectively. For ROI statistics, the variable names are intrasession_ROI and intersession_ROI.