Laminar fMRI pipeline

Documentation

Tommy Clausner

last updated on May 4, 2018

Contents

List of Figures

List of Tables

1 Before starting

1.1 Pre- requisites

- MRIcron (https://www.nitrc.org/projects/mricron)
- FSL (https://fsl.fmrib.ox.ac.uk/fsl/fslwiki)
- FreeSurfer (https://surfer.nmr.mgh.harvard.edu/)
- SPM (http://www.fil.ion.ucl.ac.uk/spm/)
- analysePRF (http://kendrickkay.net/analyzePRF/)
- knkutils (https://github.com/kendrickkay/knkutils/)
- Open fMRI Analysis (https://github.com/TimVanMourik/OpenFmriAnalysis)
- MATLAB R2015a or later (https://nl.mathworks.com/products/matlab.html)
- Tommy's Scriptinator 3000 TM (optional; https://github.com/TommyClausner/Scriptinator)

Use e.g. MRIcron's dicom2nii to convert raw MRI data to NifTi files.

1.2 Folder structure

The general folder structure is set up by a call to setupfolders.sh (see also Figure 2)

- Functional data must be located in niftis/**functionals**/
- Anatomical data must be located in niftis/**t1**/
- Inverted functional data must be located in niftis/**inverted**/
- Proton density data must be located in niftis/**pd**/
- Inverted proton density data must be located in niftis/**pdinverted**/

1.3 How things work

Parts of the analysis run in Bash shell scripts. Those can be called from their root directory using "sh scriptname.sh" (excl. quotation marks). Each step within the analysis has a different subfolder containing the respective results of this step. Leading numbers indicate in which logical order the corresponding shell scripts should be executed. Note that all files created by each respective step are stored in the coresponding folder. However files that need to be preset (e.g. config files) must be located in A_helperfiles. Note Further, that the entire analysis can be broken down to a few sub-pipelines. Those are

explained in a individual section.

Since most operations can be grouped into bigger chunks or "pipelines", all subsections below are wrapped into pipeline scripts, calling them in the respective order. Addtionally a file having the extension .pipe stores the pipeline information in a format readable with Tommy's Scriptinator 3000 TM. This is a java based tool, that can be used to represent and store scripts and their dependencies including a graphical representation. See Section 7.2 of this document for more information.

2 Suggested order of operations

This document builds upon the analysis rational used during out own analyses. It is build upon the technically facilities of the DCCN (Radboud University Nijmegen, 2018). Hence a copy-paste workflow will only work on a very similar infrastructure.

- makenewsubject.sh
- copy NifTi files to the correct location (see above)
- open terminal (the following are terminal commands)
- cd PATH/SUBJECTFOLDER/B_scripts
- sh runonqsub.sh 32gb do_preparefunctionals.sh

2.1 Pre-processing

Data will be motion and field distortion corrected. An inplane template for the functional scans to be used for the co-registration will be created.

maximum memory required	approx time required
64GB	$\approx 6h$

- sh runonqsub.sh 32gb do_realignment.sh
- sh runonqsub.sh 32gb do_distcorr.sh
- sh runonqsub.sh 64gb do_applysimpledistcorr.sh
- sh runonqsub.sh 16gb do_preparecoregistration.sh
- sh do_correctavgdiff.sh
- using pipeline: sh preprocessing.sh

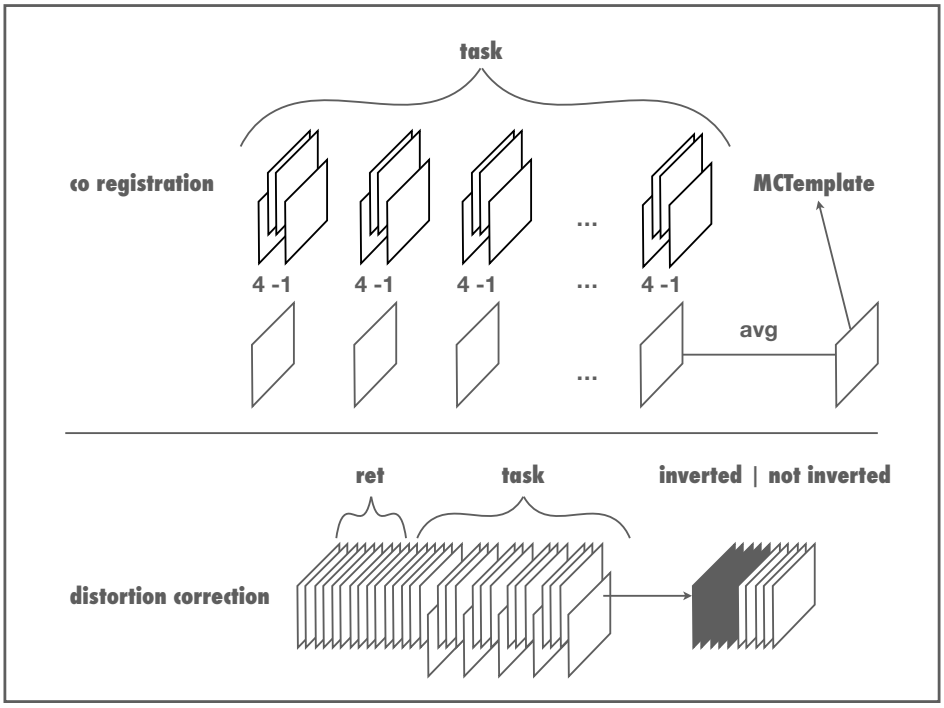


Figure 1: Coregistration and distortion correction visual depiction

2.2 Co-registration and masks

Data will be co-registered to the anatomical T1 scan based on the gray and white matter boundaries. Furthermore partial and full brain masks will be created, including gray and white matter and brain.

maximum memory required	approx time required
16GB	$\approx 6.5h$

- sh runonqsub.sh 16gb do_fsrecon.sh
- sh do_coregistration.sh
- sh runonqsub.sh 16gb do_makemasksandlabels.sh
- using pipeline: sh CoregistrationAndAnatMasks.sh

2.3 Retinotopy

Population receptive field mapping. Estimates retinotopic mapping from screen to cortical locations. This is to be done in order to obtain boundaries for visual areas such

as V1, V2 and V3. First the fMRI time series will be interpolated (cubic) to match the numeber of frames presented during the stimulation. Afterwards pRF mapping will be analyzed and paramters will be written to volume files (NiFTI) and surface files for use with FreeSurfer.

maximum memory required	approx time required
64GB (480GB)	$\approx 8h$

- sh do_tseriesinterpolation.sh
- sh do_split_analyzePRF.sh
- sh combine_split_PRF_results.sh
- sh make_PRF_overlays.sh
- sh runonqsub.sh 16gb makeOverlays.sh
- using pipeline: sh retinotopy.sh

2.4 laminar separation

Gray matter boundaries will be used to compute cortical layer separation. First tksurfer is used to "draw" occipital ROIs based on the retinotopic mapping. Those regions will be transformed into gray matter masks such that an individual masks for V1, V2, V3 for each hemisphere exists. Cortical layer separation yields a probability map for each voxel to lay within a specifcially spaced area between gray matter boundaries. Thus a layer specific ROI specific map is created for each ROI by multiplying the ROI map with each of the layer probability maps. The results are 4D mask volumes separating the layer probabilies in the 4th dimension.

maximum memory required	approx time required
16GB	$\approx 0.5h + user$

- sh GUI2ROIs.sh
- sh labels2masks.sh
- sh expandROIs.sh
- sh do_getLayers.sh
- sh do_getLayerWeights.sh
- using pipeline: sh laminar.sh

3 qsub

Scripts that are flagged with the (Q) can be run as a qsub job.
When running the computation on the cluster use the wrapper function runonqsub.sh

Note that some scripts send a MATLAB script to the qsub cluster. Thus they cannot be run using runonqsub.sh, but instead create their own job.

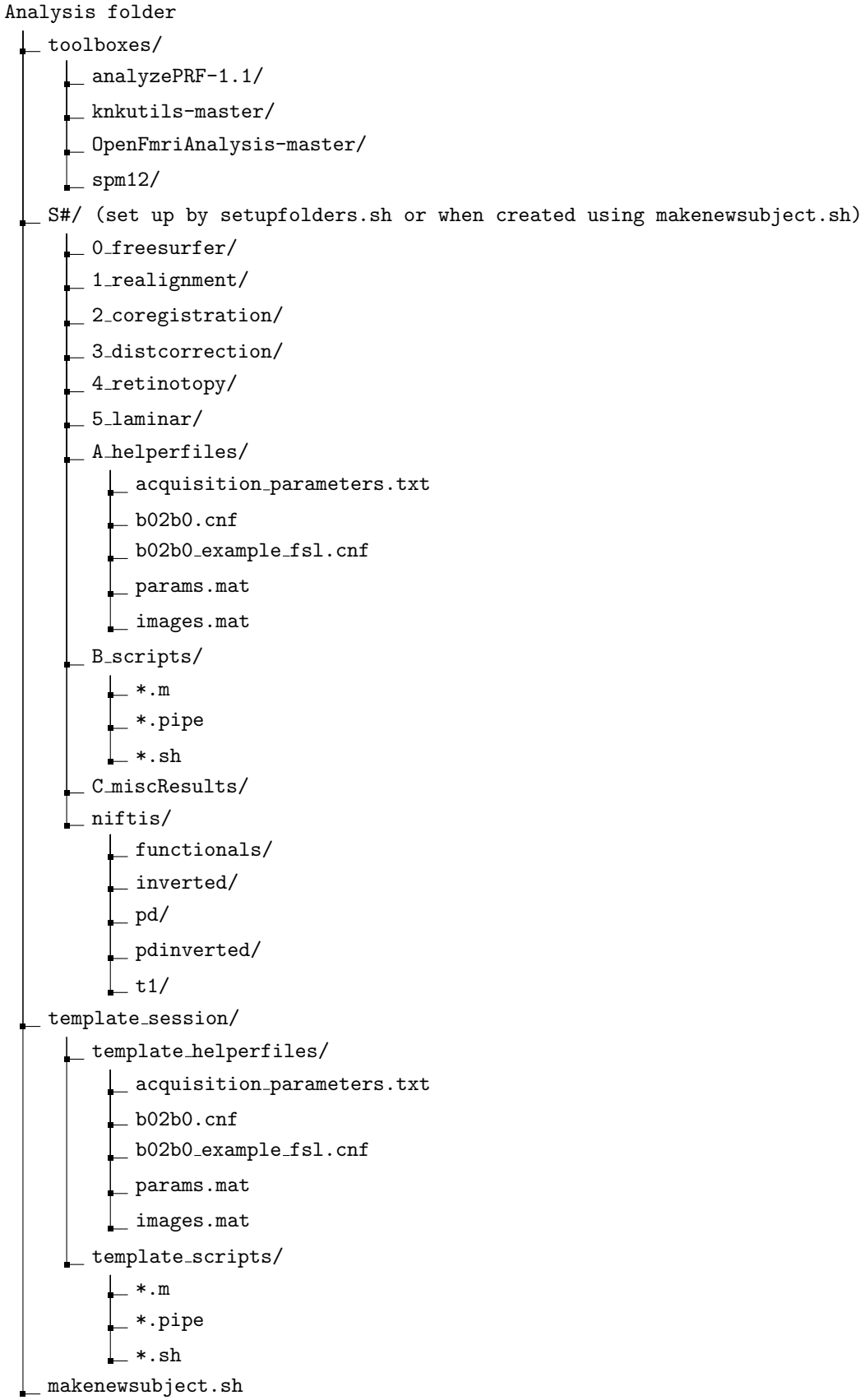
Some scripts are present as scriptnameQsub.sh Those scripts contain an internal Qsub-wrapper and do the same as sh runonqsub.sh XXgb scriptname.sh

the following requirements on memory and computation time can be expected:

script	memory required	time required
do_preparefunctionals.sh	32GB	$\approx 10min$
do_realignment.sh	32GB	$\approx 30min$
do_distcorr.sh	32GB	$\approx 15min$
do_applysimpledistcorr.sh	32GB	$\approx 30min$
do_preparecoregistration.sh	16GB	$\approx 4.5h$
do_correctavgdiff.sh	4GB	$\approx 3sec$
do_fsrecon.sh	16GB	$\approx 6h$
do_coregistration.sh	16GB	$\approx 30min$
do_makemasksandlabels.sh	16GB	$\approx 1min$
do_tseriesinterpolation.sh	64GB	$\approx 20min$
do_split_analyzePRF.sh	480GB	$\approx 7h$
combine_split_PRF_results.sh	64GB	$\approx 10min$
make_PRF_overlays.sh	16GB	$\approx 10min$
makeOverlays.sh	16GB	$\approx 5min$
GUI2ROIs.sh	8GB	user dependent
labels2masks.sh	8GB	$\approx 10s$
expandROIs.sh	16GB	$\approx 3min$
do_getLayers.sh	16GB	$\approx 20min$
do_getLayerWeights.sh	16GB	$\approx 5min$

Table 1: Approximated time and memory requirements for the respective script to run. Note that everything that is more than 4GB should be run on the cluster. Use runonqsub.sh for that purpose.

Figure 2: Initial folder structure as required for the analysis



4 Helper files

There are several helper files that are needed in order to perform the analysis:

4.1 acquisition_parameters.txt

indicating the respective acquisition parameters per volume needed in order to perform the distortion correction. One line indicates the respective parameter setting for the respective volume (1st line corresponds to 1st volume, etc.)

e.g.:
0 1 0 0.042
0 -1 0 0.042

The first 3 columns indicate the respective phase coding direction the last column some weird value, that is only important if it changes. Otherwise it's fine to use any value as long as it is the same.

4.2 b02b0.cnf

Settings for distortion correction.

Needed in order to set the parameters for the distortion correction (example provided by FSL).

4.3 params.mat

Parameters used for retinotopy scans as obtained from VistaDisp.

4.4 images.mat

Stimuli used for retinotopy scans. Stimuli file must be such that each frame is represented as a binary image. The matrix must be of shape $Y \times X \times T$ where X and Y are the image dimensions in pixel and T is the respective number of frames.

5 Scripts explained (support for runonqsub.sh: Q)

5.1 runonqsub.sh

Initiates a qsub session and sends it to the cluster. This is necessary, since scripts running on the cluster are copied to a different directory, but within scripts all directories are relative. runonqsub.sh must hence be run in a "normal" session not using a non-interactive qsub, giving the respective script that was intended to run as an argument. runonqsub will then set everything up correctly. Further you have to specify the amount

of memory needed.

example: sh runonqsub.sh 32gb myscript.sh

Additional arguments can be passed after the script.

example: sh runonqsub.sh 32gb myscript.sh myargument

Under the hood:

- a qsub session is initiated like so: qsub -l walltime=24:00:00,mem=$1 -F "$DIR ${@:3}" $DIR/$2
- $DIR is the current absolute path to B_scripts
- $1 is the amount of memory used (e.g. 32gb)
- $2 is the respective script to run (e.g. myscript.sh)
- ${@:3} is all following arguments that are parsed to the script
- every time runonqsub.sh is called the absolute path is forwarded to the script (-F $DIR) in order to keep the relative dependencies clear

5.2 cleanscriptsfolder.sh

Since qsub puts a output + error log into /B_ scripts cleanscriptsfolder.sh can be used to move all qsub outputs to /B_scripts/qsuboutput

5.3 liveupdateqsub.sh

Submits a qstat -u user request every second.

example: sh liveupdateqsub.sh tomcla

exit command by hitting ctrl+c

5.4 waitForJobs.sh

Checks all currently running jobs (qstat -u user) and waits until the last script up until this point has finished.

5.5 makenewsubject.sh

executable script (double click to execute)

Creates a new subject (S#) folder structure in order to get all folder dependencies right. It automatically detects folders called "S*number*" and creates a new folder incrementing *number* by one.

5.6 setupfolders.sh (Q)

Sets up the necessary folder structure for the analysis (within a subject folder). This function is automatically called when using makenewsubject.sh

5.7 do_preparefunctionals.sh (Q)

Prepares functional data, i.e. removes the first 4 and the last x volumes of every set of functionals, where x is the number of volumes acquired after the stimulation ended. If files were modified (i.e. if volumes were deleted), the original file will remain in /niftis/functionals/old

5.8 do_realignment.sh (Q)

Merges and motion corrects all files in /niftis/functionals using the average volume of all task volumes.

numberofvolumes.txt is written to /A_helperfiles giving information about which sets had how many volumes. The first column is coresponds to dim4 from fslinfo

Under the hood:

- fslmerge is called to create a combined .nii for all files in niftis/functionals/*sparse* - files are merged allong the 4th dimension
- a .txt file is created saving which files were merged and how many volumes were in there
- mcflirt is used on the combined data doing the motion correction based on the average volume across all task blocks
- all results are stored in /1_realignment
- results have the extension "_mcf"

5.9 do_distcorr.sh (Q)

Uses the averge volume of the inverted set of images in /niftis/inverted as reference for *inverted*

Uses as many volumes n as there were in /niftis/inverted. Respective volumes are selected starting from the last going in $4\ x\ n$ steps backward. Hence the resulting average of this set was computed over the 4th volume of each of n last sets in the normal images. Simplified (all 1 are selected):

for n=5

0,0,0,0,...,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1

Both (normalavg and invertedavg) will be used to esimate the distortion correction.

Under the hood:

- reference volumes (normal + inverted) are selected in order to do the field distortion estimate (selection: see above).
- selected reference volumes are merged into a single file (/3_distcorrection/all_b0.nii.gz)
- topup is called using the merged (normal + inverted) .nii and performs field mapping accoriding to the specifications in /A_helperfiles/b02b0.cnf using the acquisition parameters specified in acquisition_parameters.txt
- all results are stored in /3_distcorrection

5.10 do_applysimpledistcorr.sh (Q)

Applies distortion correction to all functional data.

Under the hood:

- distortion correction is applied to the full set of functionals using the OUTPUT of topup (topup [...] –out=OUTPUT) as well as acquisition_parameters.txt (Using –method=jac)
- all results are stored in /3_distcorrection
- all results have the prefix "corrected_"

5.11 do_preparecoregistration.sh (Q)

Computes a reference volume (MCTemplate) used as the basis to co-register with the anatomical image. This image is computed by splitting the entire set of task volumes into chunks of 4. In each chunk the 1st volume is substracted from the 4th. This procedure yields a pseude-T1 contrast. All chunk differences are then averaged to form "MCTemplate"

5.12 do_correctavgdiff.sh (Q)

Due to the subtraction of the 1st from the 4th volume the area outside the brain has the highest value. In order to correct this the lowest value will be shifted to zero the a certain part of the higher values are cut (e.g. 0.975).

The result should be checked using "fsleyes ../2_coregistration/Inplane/MCTemplateThr*.nii.gz" to ensure that the area outside the brain is nulled. Note that it can happen that some areas within the brain are nulled as well. If they are not too wide spread they do not affect later co-registration.

5.13 do_fsrecon.sh (Q)

Performs segmentation using Freesurfer.

If running on OSX the script assumes:
FREESURFER_HOME=/Applications/freesurfer

If running on linux the script assumes:
FREESURFER_HOME=/opt/freesurfer/version

The target image that is used for the segmentation must be located in /niftis/t1

Under the hood:

- calls recon-all -i /niftis/t1/* -subjid 0_freesurfer -all
- all results are stored in /0_freesurfer

5.14 do_coregistration.sh

Will submit a qsub session using MATLAB, that in turn will be using FreeSurfer (bbregister) and OpenFmriAnalysis to perform a linear and non-linear boundary registration. Movie files of the respective co-registration are stored in /C_miscResults
The file used to execute MATLAB commands is /B_scripts/do_coregistration.m Note that it will be modified using the correct folder settings, which yields tmp.m that will be called and removed after MATLAB was closed.

5.15 do_coregistration.m

The script doing the actual co-registration. Uses wrapper functions from OpenFmriAnalysis to evoke FreeSurfers bbregister and computes a recursive boundary based registration.

The results are seevral files storing the boundaries as a point cloud, and the respective transformation in diferent formats.
Under the hood:

- calls bbregister
- results are stored in /2_coregistration
- yields boundaries.mat, matrix.mat, bbregister.dat, transmat.txt, transmatinv.txt

5.16 do_makemasksandlabels.sh (Q)

example: sh do_makemasksandlabels.sh

Uses FreeSurfer reconstruction to create masks for CSF, gray matter and white matter and creates labeled volume for use within retinotopy containing left / right + gray / white matter. All masks will be created as full-brain and partial-brain masks.

If the orientation needs to be changed, parse the respective FreeSurfer compatible orientation parameter (https://surfer.nmr.mgh.harvard.edu/pub/docs/html/mri_convert.help.xml.html)

Under the hood:

- moves anatomical volumes to functional space calling do_movesinglevolume.sh InputVolume.nii.gz transmat.txt OutputVolume.nii.gz
- results are stored in /2_coregistration
- coregistered functional masks have the prefix "fct" and the suffix "coreg"

5.17 do_movesinglevolume.sh

Simple wrapper function to apply a transformation from one volumetric space to another, given a transformation matrix.

Under the hood:

- flirt -applyxfm -in $1 -ref $1 -init $2 -out $3
- $1: volume to move
- $2: transformation matrix in form of a .txt file containing a 4×4 transformation matrix

5.18 do_tseriesinterpolation.sh

Wrapper function for do_tseriesinterpolation.m Variables used by do_tseriesinterpolation.m can be defined in the header section of the script.
Under the hood:

- submits a MATLAB job using qsub calling do_tseriesinterpolation.m using the below settings
- default number of blocks: 3
- default TR: 2.7s
- default Mask Threshold: 0.01
- default rescaled stimulus size: 100px

5.19 do_tseriesinterpolation.m

Interpolates time series for retinotopy scans to match the number of frames during the stimulation (cubic interpolation). The new TR will be $TR_{new} = \frac{TR_{old} N_{volumes}}{N_{imageframes}}$. The function implements "tseriesinterp" from the analyzePRF toolboxe. The data will be normalized to percent signal change.

Furthermore all images will be downsampled to the predefined size (e.g. 100px)

Under the hood:

- interpolates time series of functional data to match the numer of frames during the stimulation
- results: tIntData.mat (interpolated time series)
- results: mask.mat (volume mask)
- results: voxelindices.mat (indices of mask voxels, ratio between $N_{imageframes}$ and $N_{volumes}$, mask threshold)
- results: downsampled images

5.20 do_split_analyzePRF.sh

Wrapper function to call do_analyzePRF.sh The function calls a predefined number of instances of do_analyzePRF.sh to run on the cluster, submitted using qsub. Per default 80 jobs requesting 6 GB of memory each (see Section 5.21), will be spawned. This procedure speeds up the population receptive field estimation significantly and would be otherwise impractical on high resolution functional data.

5.21 do_analyzePRF.sh

Wrapper function to call do_analyzePRF.m Spawns a qsub job using 6GB of memory (default) and runs a specific part of a set of parts. The first input argument is the part that is to be computed and the second input argument is the sum of all parts.

example: sh do_analyzePRF.sh 1 5

example 2: sh do_analyzePRF.sh 1 1

5.22 do_analyzePRF.m

Computes pRF mapping implemeting "analyzePRF" from the analyzePRF toolbox. It can be decided whether to average across blocks or if the estimates shall be done for each block separately and averaging the results. Note, that running the analysis on separate blocks slows down computation time, but to my experience yields slightly better results.

Under the hood:

- estimates pRFs
- computes pRF mapping for part x of X by splitting the list of indices and selecting the corresponding set
- set avgdata=1 to average data before the analysis (otherwise it will be averaged afterwards)
- result: structure containing pRF parameters of part x of X

5.23 combine_split_PRF_results.sh

Wrapper function for combine_split_PRF_results.m Combines split results from do_split_analyzePRF.sh (see Section 5.20). Note that the exact amount of parts that the data was split into must be defined here (default: 80).

5.24 combine_split_PRF_results.m

Combines split results from do_split_analyzePRF.sh (see Section 5.20) into a single MATLAB file. Note that given the respective numerical naming of all parts (x_of_X) parts will be concatenated along values of x. After this operation split parts will be deleted.

5.25 make_PRF_overlays.sh

Wrapper function for make_PRF_overlays.m

5.26 make_PRF_overlays.m

Takes result of pRF mapping and creates .nii files for separate parameters (ang, ecc, expt, r2, rfsize, xpos, ypos). The data will be saved in ../4_retinotopy as parameterName_map.nii

Under the hood:

- angle (θ) will be devided by 180 and multiplied by π
- eccentricity (r) will be devided by the image size (in px) and multiplied by the original value range (° visual angle; default: 7). Only ecc values that have a positive R^2 and are smaller than the defined field of view, will be written to file. All other values will be set to NaN
- X position (xpos) $\cos(\theta) \cdot r$
- Y position (ypos) $\sin(\theta) \cdot r$

5.27 makeOverlays.sh (Q)

Makes FreeSurfer compatible overlays from NiFTI files. Calls FreeSurfer's "mri_vol2surf" in order to transform a volumetric .nii file into a surface file. This will be done for both hemispheres separately. Parameters transformed to surface files are: ang, ecc, xpos, ypos, r2

Under the hood:

- mri_vol2surf –src $DIR/4_retinotopy/ang_map.nii –srcreg $DIR/2_coregistration/bbregister.dat –hemi ?h –surf pial –out $DIR/4_retinotopy/?h.ang.mgh –out_type paint
- –src: NiFTI to be transformed
- –srcreg: result from "bbregister" of do_coregistration.sh (default: bbregister.dat)
- –hemi: respective hemisphere (lh for left hemisphere or rh for right hemisphere)
- –surf: surface used as reference (obtained from FreeSurfer in ../0_freesurfer/surf/)
- –out: output file name
- –out_type: influences how the file is outputted (see "mri_vol2surf –help" for more information)

5.28 GUI2ROIs.sh

This script spawns an instance for tksurfer (FreeSurfer)

5.29 labels2masks.sh (Q)

5.30 expandROIs.sh

5.31 expandROIs.m

5.32 tc_expandVoxelSelection.m

5.33 do_getLayers.sh

5.34 do_getLayers.m

5.35 do_getLayerWeights.sh

5.36 do_getLayerWeights.m

6 How to: Selecting visual areas based on pRF mapping

7 How to: Pipelines

7.1 Scripts

7.2 using Tommy's Scriptinator 3000 TM

Tommy's Scriptinator 3000 TM is a graphical user interface that allows drag and drop based arrangements of scripts on a canvas. The code of the script as well as defined variables can directly be viewed by double clicking one of the icons representing a script (examples below). This way analysis pipelines can easily be comprehended and shared. Tommy's Scriptinator 3000 TM can be downloaded from:

https://github.com/TommyClausner/Scriptinator/

It requires at least JDK9 or newer to be installed:

http://www.oracle.com/technetwork/java/javase/downloads/

After downloading the repository from GitHub, one can start the program by double clicking Scriptinator.jar. All further help will be provided from there.

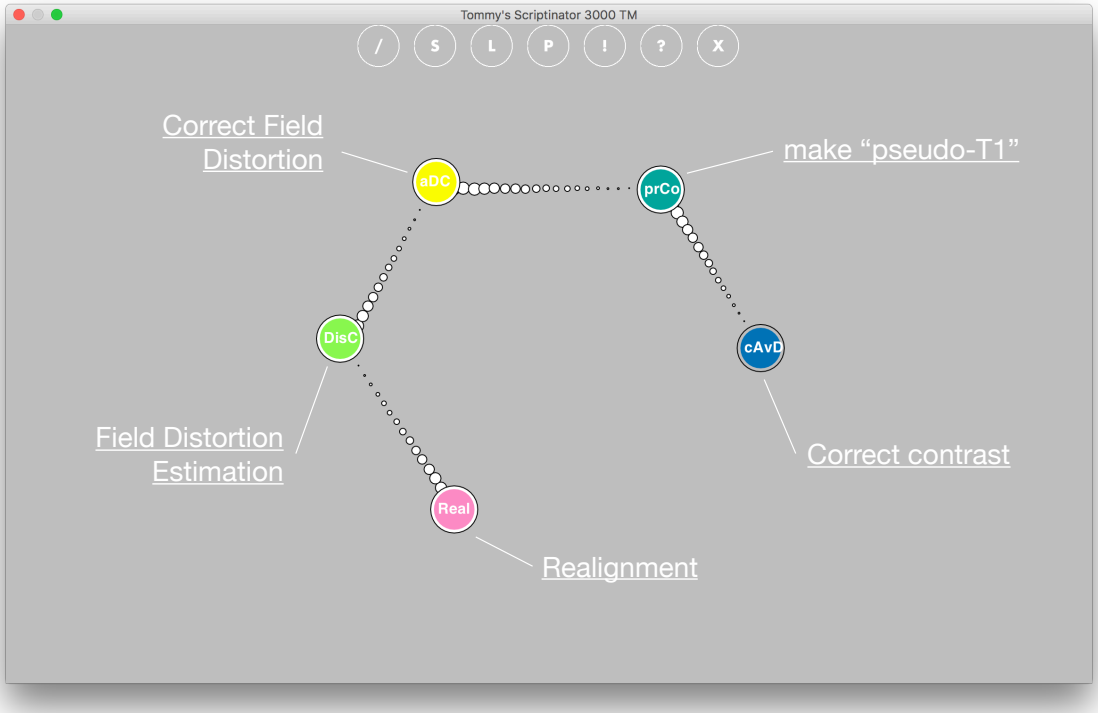


Figure 3: Example pipeline for the pre-processing, using Tommy's Scriptinator 3000 TM: preprocessing.pipe

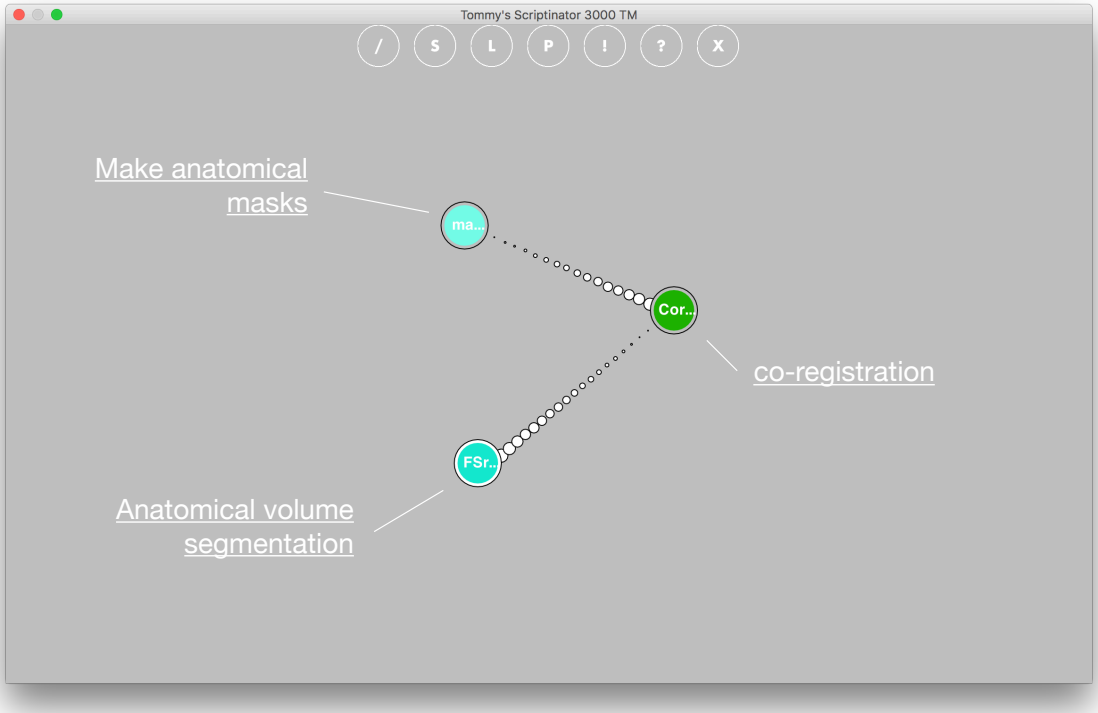


Figure 4: Example pipeline for the anatomical segmentation, using Tommy's Scriptinator 3000 TM: CoregistrationAndAnatMasks.pipe

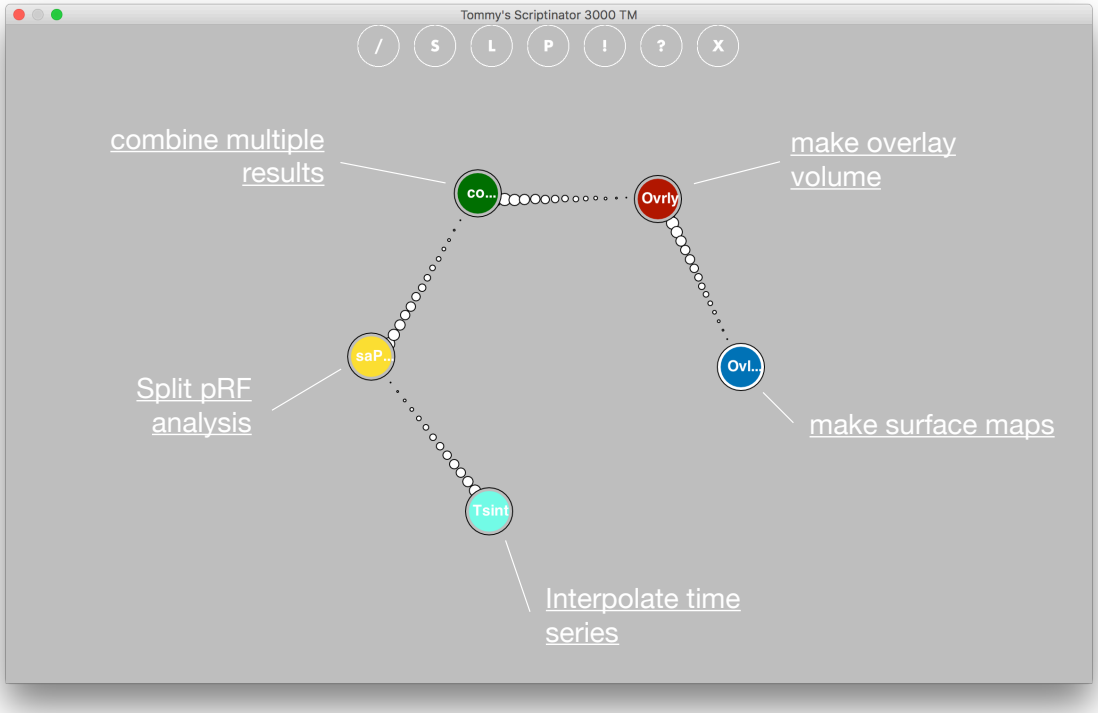


Figure 5: Example pipeline for the retinotopy, using Tommy's Scriptinator 3000 TM: retinotopy.pipe

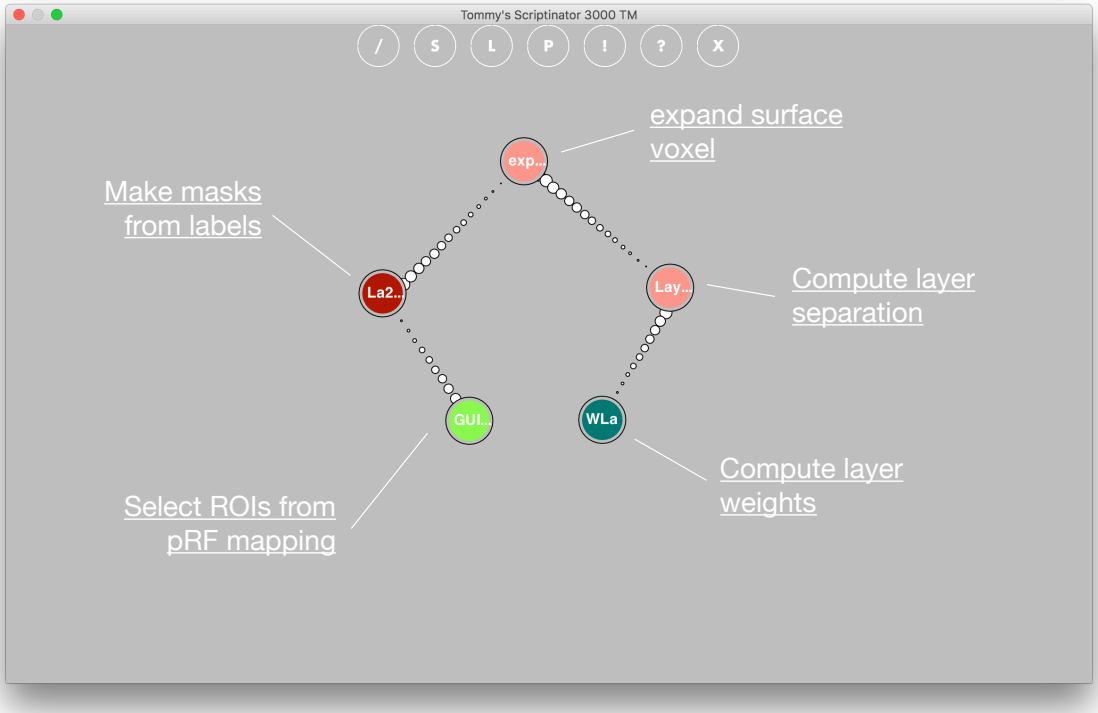


Figure 6: Example pipeline for the cortical layer definition, using Tommy's Scriptinator 3000 TM: laminar.pipe