NHP Awake fMRI Preprocessing Pipeline User Guide

Software Requirements

UNIX / Mac OS

Setting up environment variables like $PATH

- $PATH is a standard environment variable in operating systems that specifies directories in which executable files can be found and thereby allows applications to be easily accessible from the command line without specifying the full path each time
    - https://en.wikipedia.org/wiki/PATH_(variable)
    - e.g., it allows you to run `matlab` in the command line without the full path to the MATLAB application folder
    - Editing the $PATH variable is required for setting up MATLAB, ANTs, FSL etc. to be used in the scripts/command line
- In addition to $PATH, you can define any arbitrary environment variable that points to any arbitrary bit of information, that will be accessible to all scripts run from your terminal
    - E.g., we will define $MRCATDIR to point to the local MrCat directory, thereby allowing all scripts to use $MRCATDIR without having to re-define its pointer every time
- This is all done in your Shell config file (.bash_profile or other config file of choice)
- Your .bash_profile file (or similar) normally exists within the main user directory (e.g., /Users/Matthew) but is normally hidden in the Finder/terminal (due to the .)
    - These files can be made visible: https://ianlunn.co.uk/articles/quickly-showhide-hidden-files-mac-os-x-mavericks/
- Fastest way to edit the .bash_profile is using “nano”, a command line editor, directly in the command line:

```
nano .bash_profile
```

- this will open up nano in the terminal and allow you to edit the file – for example, to set up the ability to call MATLAB from the command line:

```
PATH=/Applications/MATLAB_R2014b.app/bin:${PATH}
export PATH
```

    - Here we are pre-appending the MATLAB application bin folder (application executables are commonly stored in the respective “bin” folder) to the $PATH variable, and exporting it
- After any edits:
    - to save a file in nano, press ctrl+o (to write out) → it will ask you if you want to write it to the same filename (.bash_profile), press enter to confirm → press ctrl+x to exit nano
- Nano can be tricky to use
    - See here for tips: https://wiki.gentoo.org/wiki/Nano/Basics_Guide
- You can also avoid using nano by:
    - opening .bash_profile in Sublime (e.g., through the Finder)
    - this requires that hidden files (prefixed with a .) appear in your Finder (see above)
- After any saved changes to the .bash_profile, to see the effects of your edits:
    - you need to reload the .bash_profile in your current terminal (by running `. .bash_profile`)
    - or open a new terminal → the .bash_profile is meant to set up the config for a given instance of a terminal and is therefore automatically loaded every time a new terminal is opened

MATLAB 2013 or later

- Check with the department IT staffs, or
- If you are a student, get your licence from:
- https://register.it.ox.ac.uk/self/software
- Set up MATLAB command line by adding the MATLAB application bin folder to your $PATH in your .bash_profile file. For example:

```
PATH=/Applications/MATLAB_R2014b.app/bin:${PATH}
export PATH
```

- Set up $MATLABBIN environment variable (used in MrCat toolbox scripts)
    - Add the following code to your .bash_profile file:

```
matlabdir=($(echo /Applications/MATLAB_R20*))
matlabdir=${matlabdir[${#matlabdir[@]}-1]}
[[ -z $MATLABBIN ]] && MATLABBIN=$matlabdir/bin
export MATLABBIN
```

MrCat toolbox

- Obtaining/working through Git is strongly recommended (see Git guide)
- Accessing the toolbox requires getting user access from Rogier Mars, who administrates the MrCat Gitlab server hosted in the Netherlands
    - Once an account is granted, login here: https://gitlab.socsci.ru.nl/users/sign_in
- After the MrCat repository is available on your computer, we need to add the $MRCATDIR environment variable to your .bash_profile, for example:

```
MRCATDIR=/Users/Matthew/code/MrCat-dev/
export MRCATDIR
```

Advanced Normalization Tools (ANTs)

- Installation instructions:
    - https://github.com/ANTsX/ANTs/wiki/Compiling-ANTs-on-Linux-and-Mac-OS
    - NOTE: though instructions assume Git usage (preferred), ANTs can be manually downloaded from https://github.com/stnava/ANTs.git
- Note that installation requires:
    - **xcode command line tools ("developer tools")**
        - The easiest way to get this is to make a free developer account with Apple by using your Apple ID (create one if don't already have one)
        - https://developer.apple.com/download/
        - NOTE: make sure you get the version compatible with your current OSX version
    - **cmake**
        - once installed, add to $PATH:

```
PATH=/Applications/CMake.app/Contents/bin:${PATH}
export PATH
```

- Ensure ANTs works from the command line (i.e., set and update $PATH in .bash_profile)

FMRIB Software Library (FSL)

- http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslInstallation

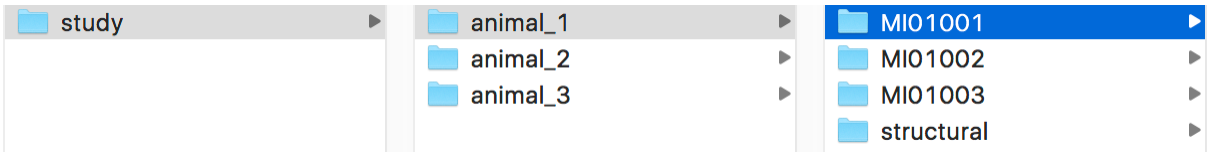
Your favourite text editor

- Sublime Text is a good one (https://www.sublimetext.com/) → the trial version is essentially free forever, but "pay for me" messages pop up every so often (tolerable)
- Provides syntax highlighting and code completion for many languages

- See also Atom

Setting up general folder structure

Here we set up the general folder structure – here this is done manually, but scripting is possible. The assumed general folder structure is:



where MI01001, MI01002…etc. are the session folders containing the raw fMRI data and structural is a folder containing that animal's structural image.

Setting up the session folder structure

Here we set up the specific folder structure for each session upon which the rest of the pipeline depends. This is also when it is determined which specific EPI (ep2d) file will be preprocessed for any given session folder (important if each session consists of multiple acquired EPIs, e.g., different runs or task + resting state data).

Required files:

- WrapperPrepareRawDataSample.sh (in MrCat/in_vivo/PreprocFunc_macaque/)
- PrepareRawData.sh (in MrCat/in_vivo/PreprocFunc_macaque/)

Steps:

- copy WrapperPrepareRawDataSample.sh from your local MrCat directory into a separate working directory (especially if using Git)
    - given that you will be editing the wrapper for your own needs, this will avoid any future version conflicts when MrCat is updated and pulled
- Edit the wrapper to your needs
    - this wrapper calls PrepareRawData.sh which sets up the folder structure within each session folder
- This script requires user input
    - Occurs in cases when there is more than one EPI (ep2d) file present in a session folder (which is often!)
    - recommended to run this script for all sessions at one time (i.e, in a loop) to quickly provide user input when necessary
- NOTE: PrepareRawData.sh considers EPI file sizes to decide whether to skip an EPI file for preprocessing or to ask for user input.
    - These file size thresholds were based on commonly acquired session lengths and may not apply to your study!
    - Check the script and edit accordingly
- The full preprocessing pipeline assumes that each session folder contains only ONE EPI (ep2d) subfolder
    - if multiple runs or task + resting state data were acquired in one session, the easiest approach is to split each session into multiple pseudo-session folders (e.g., MI01001_run1, MI01002_run2…etc.)

    - or else the pipeline edited accordingly…
- After running the script, each final session folder should have:
    - a raw folder (with all raw EPI/GRE files)
    - a folder for each GRE (with symlinks to raw GRE files)
    - ONE ep2d folder (with a symlink to raw EPI file)
- The script does NOT delete any data
    - even if it "skips" certain EPI files, it only re-organises it
- NOTE: if you run the script a second time on the same session:
    - It will not find any useable files because they would have all been moved into the "raw" subfolder
    - To run the script a 2nd time (for whatever reason), you will first have to manually or otherwise re-organise the session folder into its original state

Preprocessing the structural images

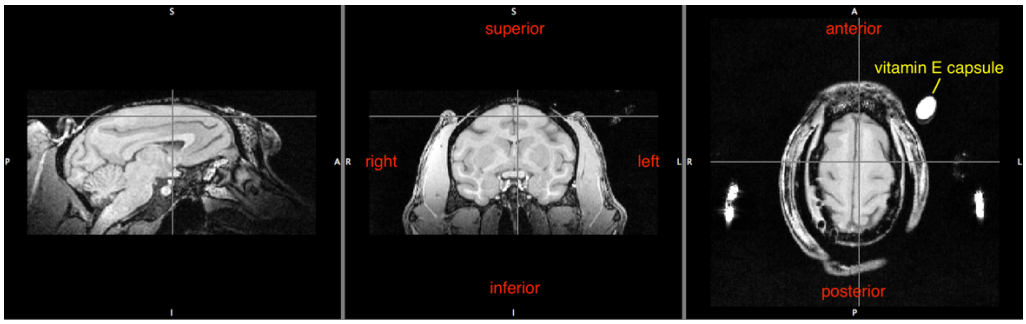
Here we manually check and preprocess the structurals for each animal.

Required files:

- structural images (.nii.gz format)
- wrapper_struct_macaque.sh (in MrCat/in_vivo/struct_macaque/)
- struct_macaque.sh (in MrCat/in_vivo/struct_macaque/)
- reference image (e.g., McLaren template in F99 space; in MrCat/ref_macaque/)

Steps:

- double-check that all structural images are oriented/labelled correctly:



    - if images are NOT oriented or labelled correctly, you will have to fix it manually using FSL tools fslorient and fslswapdim: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Orientation%20Explained
    - NOTE: finding the right parameters to correct this may be unintuitive and can require several attempts, so create a backup copy of your structural image…you will need it!
    - Example bash code (adapt rotations and qform codes to your own data):

```
fslorient -deleteorient structural.nii.gz
fslswapdim structural.nii.gz x z -y structural.nii.gz
fslorient -setqformcode 1 structural.nii.gz
```

    - NOTE: double-check that left/right is not swapped as it can be tricky to tell by eye → a vitamin E capsule is often used during acquisition to mark the left side, so check for that if possible

- If multiple structural images for an animal are collected together:
    - Concatenate and average these using fslmerge and fslmaths tools:

    - https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Fslutils
- copy wrapper_struct_macaque.sh from your local MrCat directory into a separate working directory (especially if using Git)
    - given that you will be editing the wrapper for your own needs, this will avoid any future version conflicts when MrCat is updated and pulled
    - set it up according to your needs to call struct_macaque.sh
    - note: script can take several hours per animal
- struct_macaque.sh, for each structural:
    - if FOV too big, crops the FOV around the head, to save processing time later on
    - performs bias correction
    - performs brain extraction

    input:
        - animal directory
        - structural image
        - reference space (e.g., F99, in MrCat/ref_macaque/F99)
        - reference image (e.g., McLaren in F99 space, MrCat/ref_macaque/F99/McLaren)
        - NOTE: input images should all be g-zipped niftis (.nii.gz)
            - Use `gzip yourimage.nii` to gzip your images from the command line

    output:
        - bias-corrected image, suffixed "_restore" (structural_restore.nii.gz)
        - brain-extracted image, suffixed "_brain" (structural_restore_brain.nii.gz)
        - various masks (e.g., structural_brain_mask.nii.gz)

Creating a group template

This step creates a group template for your animals to be used as a standard image for higher-level FEAT analyses. This may slightly improve anatomical localisation of results, or at least improve visualisation (e.g., the 112RM McLaren template can be too blurry for certain purposes).

Required files:
- structural images (.nii.gz format)
- reference image (e.g., McLaren template in F99 space; in MrCat/ref_macaque/)
- groupTemplate.sh (in MrCat/ref_macaque/group_template/)
- groupTemplate_core.sh (in MrCat/ref_macaque/group_template/)
- config folder with config files (e.g., config_1.cnf) for initial+6 iterations MrCat/ref_macaque/group_template/config)
- wrapperGroupTemplate.sh (in MrCat/ref_macaque/group_template/)

Steps:
- copy wrapperGroupTemplate.sh from your local MrCat directory into a separate working directory (especially if using Git)
    - given that you will be editing the wrapper for your own needs, this will avoid any future version conflicts when MrCat is updated and pulled
    - Edit the wrapper to your needs and use it to call groupTemplate.sh
- groupTemplate.sh:
    - calls groupTemplate_core.sh to do the core registration work
    - the script can create up to 6 iterations of templates:

        - with each iteration, the template become closer to the reference image, and less like the input animals
        - using the 2nd iteration is recommended → it strikes a balance between providing a good general template in line with the reference image and preserving the anatomical features of the input s7tructural
        - by default, the script runs only 2 iterations, and provides a template for each one
    - after the group template is created:
        - the script uses ANTs to register each structural to the group template (these ANTs-format transformations will later be concatenated with the functional-to-structural ANTs transformations)
        - it also warps each animal's structural brain mask to template space, and sums across these masks to make a good template brain mask
            - this is necessary because the brain mask produced by the main template script is far too strict (needs to be fixed!)
            - see also output section below!

Input:

- Group directory (containing a folder for each animal)
- Space-separated list of monkey folder names
- Output directory
- Reference image (e.g., McLaren template in F99 space; in MrCat/ref_macaque/)

Output:

- "Transform" folder
    - FSL-compatible transforms between each animal's structural and the group template, for each iteration
    - in general, these transforms are valid to use but will NOT be used in the current form of the FEAT pipeline as they cannot be easily combined with the other ANTs transforms that will be created (FSL-ANTs warp incompatibility…)
- "Group" folder (group template images for each iteration)
- Reference space folder, e.g., "F99_McLaren" (each animal's structural in reference space)
- "ants" folder (created only for the desired iteration – 2nd by default)
    - "image" subfolder
        - each animal's structural in template space
        - group template brain mask (better than the one provided in "group" folder)
        - group template brain (groupavg_2_brain.nii.gz, if you want to use a brain-extracted template image for analysis, use this one – NOT the one in the "group" folder)
    - "transform" subfolder
        - for each animal, ANTs-compatible transform between the structural and the group template (structural_to_groupavg_2_warp.nii.gz, *these will be combined with func-to-struct warps later on*)
        - the other files are the component affine and non-linear warps which make up the integrated warp above

Running main preprocessing pipeline

Here we run the main chunk of preprocessing for the fMRI data. Broadly, this involves choosing the best GRE to be used for image reconstruction, reconstructing and ghost-correcting the data, motion correction

(i.e., slice alignment), functional to structural image registration, and manual checking of the results of this pipeline (ESSENTIAL!).

Required files:

- properly set up session folders with ep2d and GRE folders (using PrepareRawData.sh)
- bias-corrected structural image, structural brain mask (.nii.gz format)
- WrapperSample.sh (MrCat/in_vivo/PreprocFunc_macaque/)
- Reconstruction_macaque.sh (MrCat/in_vivo/PreprocFunc_macaque/)
- PickBestGre.sh (MrCat/in_vivo/PreprocFunc_macaque/)
- countVol MATLAB functions (MrCat/in_vivo/PreprocFunc_macaque/countVol/)
- offlineSENSE MATLAB functions (MrCat/in_vivo/PreprocFunc_macaque/offlineSENSE/)
- MotionCorrection_macaque.sh (MrCat/in_vivo/PreprocFunc_macaque/)
- CheckMotionCorrection.sh (MrCat/in_vivo/PreprocFunc_macaque/)
- RegisterFuncStruct_macaque.sh (MrCat/in_vivo/PreprocFunc_macaque/)

Steps:

- copy WrapperSample.sh from your local MrCat directory into a separate working directory (especially if using Git)
    - given that you will be editing the wrapper for your own needs, this will avoid any future version conflicts when MrCat is updated and pulled
- Edit the wrapper to your needs – by default, this wrapper calls:
    - PrepareRawData.sh
        - if not already run – comment out otherwise
    - Reconstruction_macaque.sh
        - chooses the GRE, reconstructs the image, along with correcting ghosting noise
    - MotionCorrection_macaque.sh
        - performs slice alignment of the data to correct acquisition artefacts caused by motion-induced field distortion (hence “motion correction”)
    - RegisterFuncStruct_macaque.sh
        - registers the mean functional image to the structural
- Parallelise your scripts
    - the pipeline takes several hours to run, so it is recommended to run it in parallel
    - can be done “manually” by running 4 wrappers in 4 separate terminals in parallel
    - can also be done using special function calls which can distribute processing across multiple threads, and may optimise the parallelisation (see Alessandro’s example script)

Reconstruction_macaque.sh – offline SENSE reconstruction of the EPI image

- *NOTE: A full description of the parameters is available if you call the function without any input.*
- The function first counts the total number of volumes in the EPI, if this isn’t pre-specified
- Next, it deals with the GREs:
    - the GRE is used to estimate the location of the brain within the EPI image to suppress ghosting noise during reconstruction
    - if more than 1 GRE is present, it will do automatic GRE selection by comparing noise levels in each GRE and in the 1st volume EPI reconstructions using each GRE
    - If you have no GRE for a given session, set this using `--usegre=0`
    - If you have only 1 GRE for a given session or know that only 1 is usable, set this `--greid=<greID>` to avoid the automated GRE picking step
    - If you have no GRE or a bad one, you may need to adjust the `--lambda=<0 to 1>` setting, which adjusts the weighting of the GRE in the reconstruction → The default

with a normal GRE is 0.5. Without a GRE, more ghosting is present. Lower lambda values (e.g., 0.25) will decrease ghosting, while also adding overall noise, so find the trade-off the works for you.

- Next, it will unpack all of the volumes in k-space, and perform offline SENSE reconstruction
  - this step relies on Hauke's offline_SENSE functions
- Lastly, it will correct the orientation/labels of the output images and gzip them
- This function can take a while depending on the number of volumes and system specs (e.g., ~1 hour for 1200 volumes on the late 2015 iMacs)

Input:

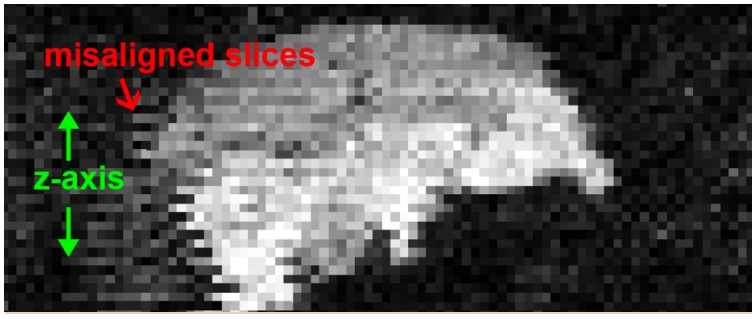
- Session folder (assumes it will find one ep2d subfolder, and at least 1 GRE subfolder, unless GRE parameter is turned off)

Output:

- Reconstructed 4D image (/MI01001/ep2d/f.nii)
- 1st volume reconstructed with each GRE (/MI01001/gre_1/fe.nii)
- reconstructed GRE (/MI01001/gre_1/fg.nii)
- extra plots of reconstruction process for 1st volume (calibration, unpacking, reconstruction with each GRE)
- summary of noise levels in each GRE/EPI (/MI01001/GRE_comparison_summary.txt)
- other miscellaneous files (e.g., phase images) that aren't relevant here

MotionCorrection_macaque.sh – correcting motion-induced slice misalignment

- This step corrects the slice misalignment that is caused by motion-induced field distortions, as the animals' bodies move inside the scanner. Note that with severe enough distortions, in addition to misalignment, volumes can show signal dropout, which is irrecoverable.



Example of misaligned slices. Because the data is acquired in an interleaved fashion, misalignments/distortions also follow this pattern. Because the phase-encoding direction is anterior-posterior, the misalignments/distortions occur in this direction as well and are most noticeable from slice to slice when looking across the z-axis.

- This function first creates a reference EPI image to which all other volumes will be aligned
  - This reference is an average of the 20% best volumes (with best defined using slice-alignment and signal intensity metrics, with a 66-33% weighting)
- Next, for each slice in each volume, using ANTs, it applies a combination of linear and non-linear (if necessary) transformations to register each slice to the corresponding slice in the reference volume
- After alignment, it calls CheckMotionCorrection.sh:
  - This function uses a slice-alignment-based and signal-intensity-based metric to detect "bad volumes" that exceed 3 standard deviations above the best volume in the session (best defined as most similar to the reference image)

    - this is done using a detrended version of the aligned EPI image to avoid rejecting volumes at the beginning and end of a session that will deviate from the mean reference image due to signal intensity drifts caused by the scanner heating (rather than motion-induced distortions)
    - this "bad volume" detection is not perfect
        - it should only be used as a rough measure to check that alignment has been done successfully (e.g., if 50% of volumes are "bad", alignment output should be checked thoroughly to ensure nothing has gone wrong)
        - more refined "bad volume" detection is described below after the main preprocessing
- This process can take 6-7 hours for a 1200-volume session
- **WARNING: if a session contains more than ~2000 volumes:**
    - As of now, the standard scripts will crash
    - See Alessandro's scripts to deal with this issue
    - Standard scripts robust to this issue may be released in the future
- *NOTE: see function for more detailed explanation*

Input:

- reconstructed EPI image (f.nii)
- bias-corrected structural
- structural brain mask

Output:

- Aligned 4D functional (/MI01001/ep2d/f_aligned.nii) and a detrended version used to find bad volumes (/MI01001/ep2d/f_aligned_detrend.nii)
- Warp that registers original 4D image to the reference EPI (/MI01001/transform/motionWarpField.nii)
- Displacement field that linearly registers 4D image to the reference EPI (/MI01001/transform/motionDisplacementField.nii) and its PCA component files
    - Along with PCA components of distance (i.e., absolute displacement)
- Rigid and affine transform parameters per slice/volume (/MI01001/transform/yScale.txt, yTranslate.txt)
- reference EPI image (/MI01001/ep2d/f_ref.nii) and associated files
- "Good" mean EPI image (/MI01001/ep2d/f_mean.nii) and associated files
    - mean of best volumes after correction
- "report" folder (/MI01001/ep2d/report/) containing several slice and volume-based measures (used later for nuisance regressors)

RegisterFuncStruct_macaque.sh – registration of the functional EPI image to the structural

- Registers the mean EPI image (created above) to the structural
- This function first creates a good brain mask for the EPI image:
    - Bias correction of the EPI image
    - Linear and non-linear registration of the EPI to the structural
    - Back-transformation of the structural brain mask into EPI space
    - Repeat iteratively until a good mask is attained
- The EPI and structural are registered using linear and non-linear approaches
- *see function for more detailed process explanation*
- takes ~30 min per session

Input:

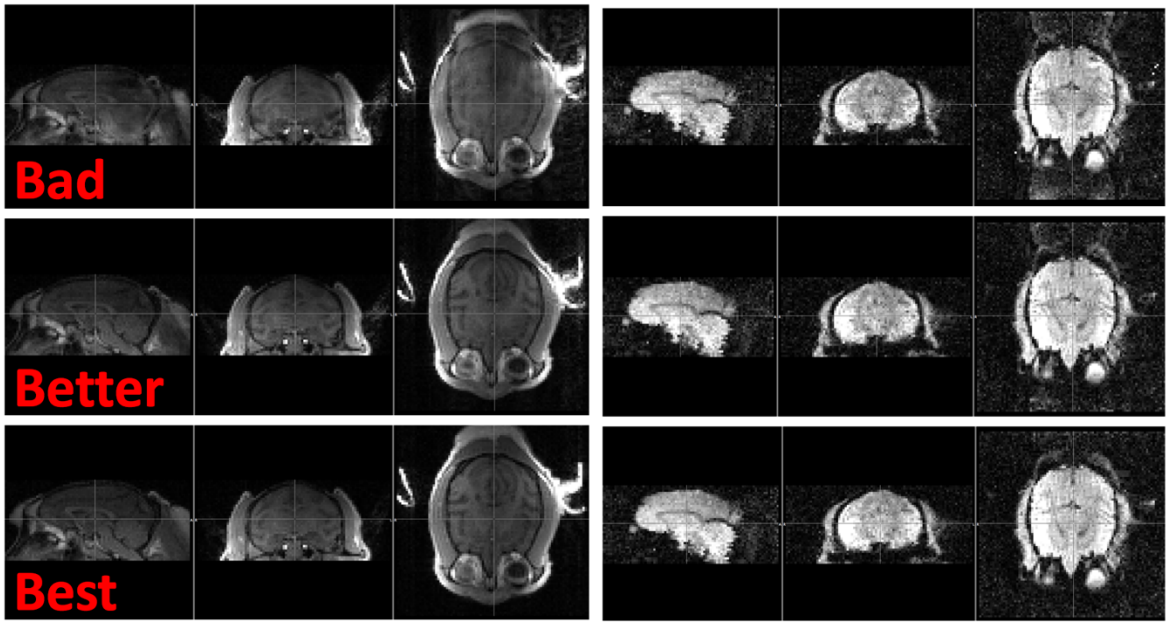
- Mean EPI image created above (f_mean.nii)
- Bias-corrected structural
- Structural brain mask

Output:

- Warps for func-to-struct and struct-to-func (in /MI01001/transform/ f_mean_to_structural_warp.nii.gz)
- Images/masks for all above warps
  - all functional-space images are in the ep2d folder, and all structural-space images are in a new folder called structural (/MI01001/structural)

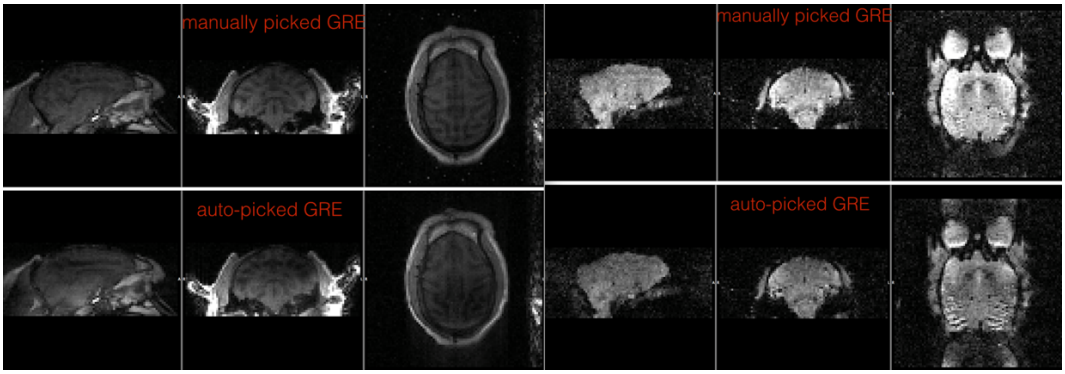
Checking the output:

- Check the unpacking plots (in /MI01001/plots_unpack1stVol/) to make sure that it looks like a brain, the 4 coil images roughly match, and that quality is reasonable
- Check that the 1st volume reconstructed EPI image (fe.nii) with the selected GRE (fg.nii), is not worse than the alternatives
  - each pair of EPI and GREs is found in the respective GRE folders(/MI01001/gre_###).



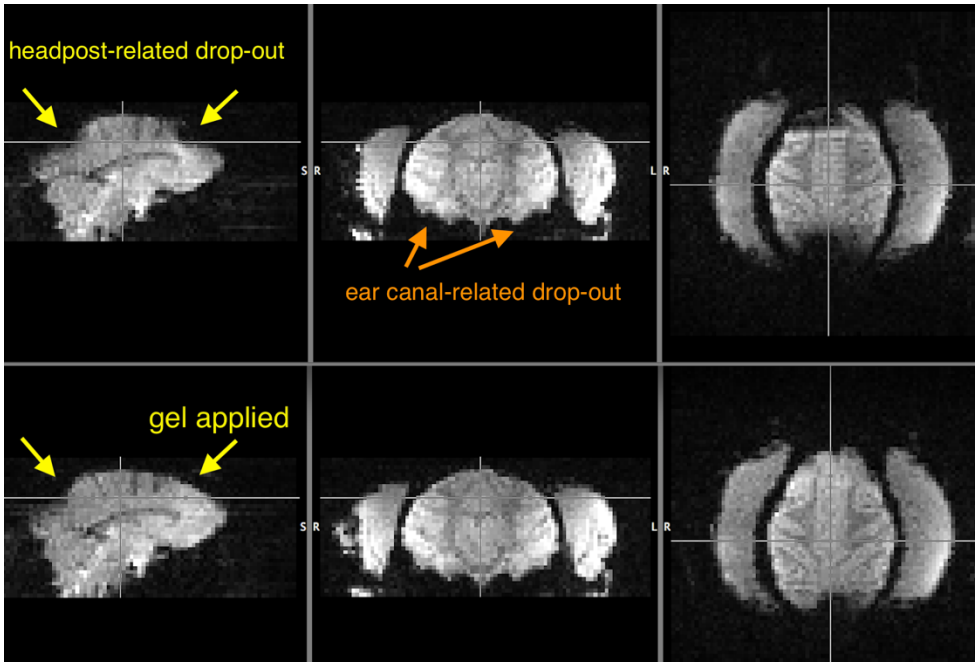
Example of 3 GREs for a session (left) and reconstructed EPI volumes (right). Note the reduction in ghosting in the EPI, as the GRE quality improvements.

- Sometimes (so far only 1/98 sessions) the selection algorithm gets fooled:



left image: GRE images, right image: reconstructed EPIs. For whatever reason the auto-picked GRE has some drop-out inside the brain leading to poor EPI reconstruction with overall lower signal intensity and more ghosting. The manually picked GRE results in far better reconstruction.

- When checking the reconstructed EPI image (f.nii), pay attention to:
  - The alignment (in the sagittal z-plane) between the EPI image and GRE image → normally, these should be obviously well-aligned because they are collected together with the animal in the same position. If they are not aligned, you will need adjust the `--grezoffset` parameter to adjust this.
  - Noise, especially near/inside the brain
  - Ghosting (copies of the head on the top and bottom of the image – see image above)
  - Signal dropouts inside the head
    - Around inferior temporal lobes is common due to ear canal in-homogeneities
    - Posterior visual cortex may also show some signal loss if motion-distortions are severe enough, though these will vary from volume to volume (easier to see in aligned image below)
    - there may also be drop-out in motor/parietal areas caused by air pocket in-homogeneities around the headpost-scalp transition, *though these should be prevented during acquisition with application of water-based Intrasite gel around the headpost!*

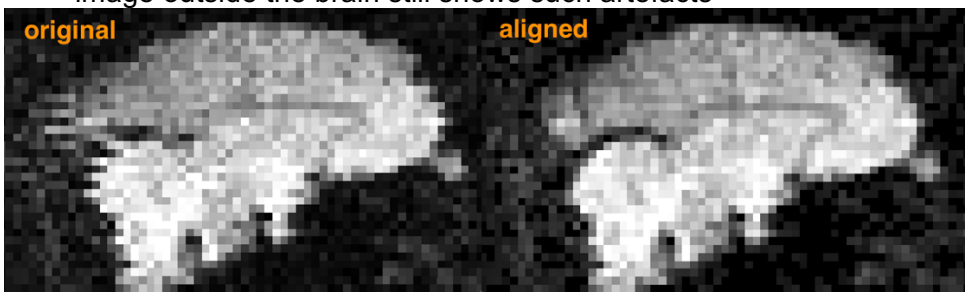


The top row is functional data from an animal showing headpost-related drop-out (yellow arrows), acquired without gel. The bottom row shows the same but with gel applied during acquisition. Also notice the ear canal-related drop-out (orange arrows).

  - *Check across all volumes* (movie mode), to make sure that the head doesn't jump around dramatically (e.g., half the FOV distance) or disappear in certain volumes (though it will certainly jump around from volume to volume before alignment)

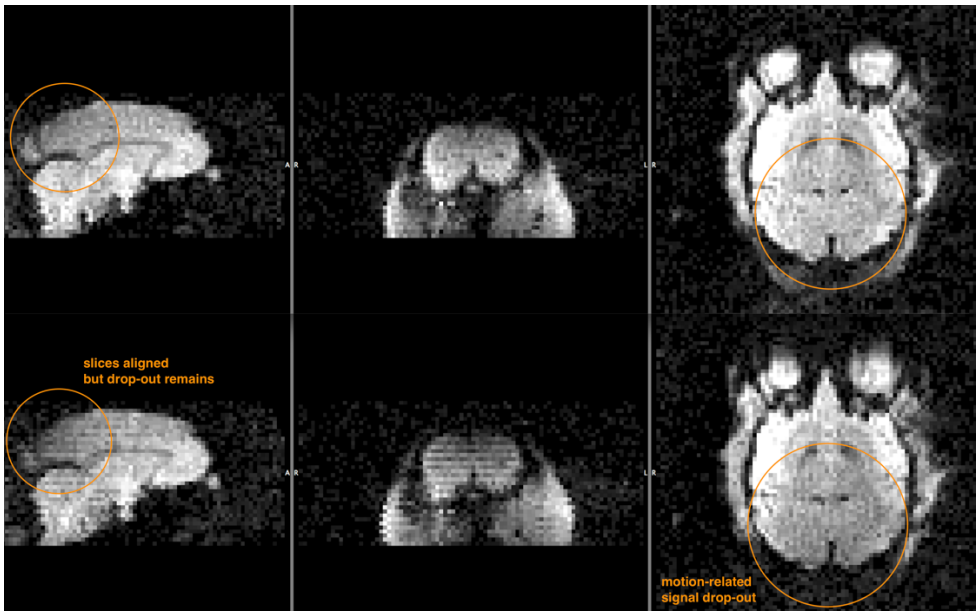
- Check the motion corrected (slice aligned) EPI image (f_aligned.nii):
  - The brain should show few or no motion artefacts (slices shifting back and forth across volumes) → the best way to check is to look at the occipital lobe in sagittal view or the axial view

- The alignment procedure is done only on the brain so it is normal that the rest of the image outside the brain still shows such artefacts



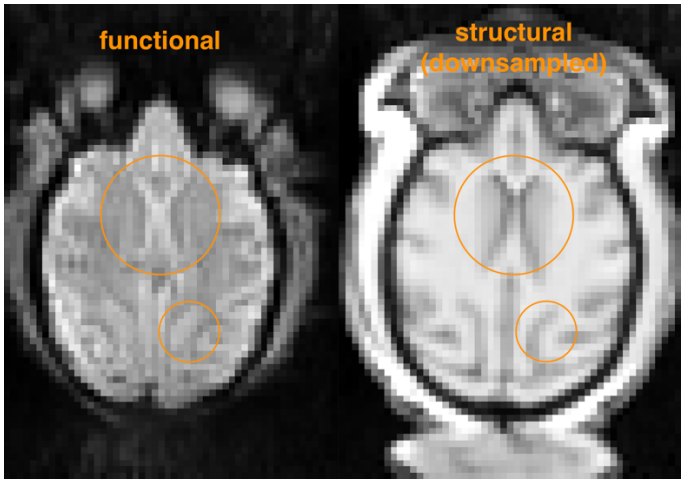
Comparison of original volume showing motion-induced slice misalignment, and the same volume after alignment.

- For some volumes, if distortions are severe enough, there will be signal drop-out even though the slices themselves are aligned. This drop-out is irrecoverable and these "bad" volumes will be automatically detected later on and regressed out during FEAT



Top row: a volume that has been well-aligned and showing decent signal strength in posterior cortex. Bottom row: a volume that has been well-aligned but shows signal drop-out in at least half of the acquired slices (the striped pattern remains even though slices are aligned).

- Check the func-to-struct registration
  - Open functional and structural images in FSLview/eyes
    - Note that you can open the functional image in upsampled "structural" resolution (MI01001/structural/f_mean_restore.nii.gz) together with the native resolution structural (structural_restore.nii.gz from your main structural folder), which may make it easier to see details
    - OR the native resolution functional image (MI01001/ep2d/f_mean_restore.nii.gz) together with the structural image in downsampled "functional" resolution (MI01001/ep2d/structural_restore.nii.gz)
    - use the opacity slider to switch back and forth between the two images
    - sulci/gyri and ventricles are good landmarks that can be checked for registration (remember that the contrast is flipped between functional and structural images)
    - signal drop-out in the functional image will become apparent
      - normally in the temporal lobes, and in the posterior cortex if motion-distortion is bad enough



Registered functional and structural images (in downsampled structural space). Ventricles and sulci are good landmarks to pay attention to. Note that it's easier to compare when images are overlaid!

Preparing for FEAT 1st level analysis (or other analysis)

Here we run a few final steps to prepare all files necessary for FEAT analysis (or other analysis).

Required files:

- Session folders with preprocessed data from previous step
- PrepareFeat_1stLevel.sh
- WrapperPrepareFeat_1stLevel.sh
- CreateNuisanceReg.m

Steps:

- Set up WrapperPrepareFeat_1stLevel.sh to call PrepareFeat_1stLevel.sh in a loop over your animals/sessions, and then call CreateNuisanceReg.m for all sessions at once.

PrepareFeat_1stLevel.sh, for each session:

- creates a new "proc" folder for final data
  - this can (optionally) be placed entirely separately from the main session folder (otherwise it just creates a "proc" folder in the main session folder)
    - useful if you want to store the heavy intermediate files on an external HD, and only work with the final data locally
- copies over the aligned 4D functional image (f_aligned.nii.gz), applies a brain mask to it, and edits the header to indicate the correct TR
- creates a concatenated ANTs func-to-struct-to-standard warp
  - combines ANTs func-to-struct warp (from RegisterFuncStruct.sh) and ANTs struct-to-standard warp (from group_template.sh)
  - this can be applied prior to calling FEAT 2nd level analysis to bring all sessions into a common space
  - This combination of warps is desirable because it reduces the number of times the functional image is resampled (and information degraded) during transformations
- it optionally deletes the f_aligned_detrend.nii.gz image from the main session folder as it takes up a lot of space and was used temporarily during CheckMotionCorrection.sh

Input:

- the session folder
- the output folder (optional)
- flag to make a concatenated func-to-struct-to-stand warp
- the standard image (e.g., group template created earlier, groupavg_2.nii.gz)
- ANTs struct-to-stand warp (e.g., created in group_template.sh earlier, structural_to_groupavg_2_warp.nii.gz)
- *See function for other options*

Output:

- "proc" folder containing:
  - masked 4D func (f_aligned_brain.nii.gz)
  - func-to-struct-to-stand warp (f_mean_to_structural_to_groupavg_2_warp.nii.gz)

CreateNuisanceReg.m, for each session:

- creates a nuisance regressor design matrix text file that can be used in FEAT analysis
- it uses 3 measures to detect "bad volumes", which can then be entirely regressed out in FEAT or otherwise
  - yScale.txt
    - for each slice in each volume, the linear transform (in the y-plane) from that slice to the corresponding slice in the mean reference image
    - averaged across slices to produce 1 measure per volume
  - similarityMetricWarp.txt
    - for each slice in each volume, the normalized correlation between that slice and the corresponding slice in the mean reference image
    - averaged across slices to produce 1 measure per volume
  - rvalueSlices.txt
    - for each volume, the correlation between that volume (mean-filtered across z-slices) and the mean reference image after correction
    - mean-filtering averages neighbouring z-slices, and so a well-aligned volume would show a good match to the reference, thereby emphasizing the slice alignment (rather than merely motion-induced signal drop-out)
  - all 3 measures were produced in the MotionCorrection_macaque.sh script, and outputted as .txt files in the session folder
  - volumes are marked as "bad" when they exceed a desired number of SDs above the median of each measure
  - this rejection process can be done recursively
    - can be useful if a few REALLY bad volumes are skewing the SD
    - better used with a high SD threshold to stay conservative with volume rejection
  - note: many bad volume regressors can significantly slow down 1st level FEAT analysis – maybe worth rejecting conservatively at least for the first analysis run
- it also adds, as parametric regressors, 13 PCA components that describe, for each volume, the warping from that volume to the mean reference image when correcting motion artefacts in MotionCorrection_macaque.sh (i.e., they capture signal variability associated with motion-induced distortion artefacts)
  - note: because these parameters capture motion-related variability, they will also correlate with other potentially task-relevant regressors that also correlate with motion (e.g., responses), so ensure that they are not completely competing with your regressor of interest

Input:

- space-separated list of input sessions (single session also works)
  - this list approach is used to avoid re-opening MATLAB for each session…
- standard deviation threshold
  - how many deviations above median to use as rejection threshold for bad volumes
- space-separated list of output directories (single session also works)
- flag to run the rejection recursively until no more bad volumes detected
- *See function for other options*

Output:

- when calling from a shell script (without any outputs defined), it produces a nuisance.txt regressor file
  - the file contains a design matrix of size: *nRegressor columns x nVolume Rows*
  - with binary columns corresponding to bad volume regressors (1 column per regressor)
  - parametric columns corresponding to the 13 slice-alignment (motion correction) components
- you can optionally call the function using the MATLAB GUI and then specify additional outputs which can provide basic summary stats of bad volumes for each session (*see function for details)*

Running 1st level FEAT analysis

See FEAT user guide for more details about analysis setup (or the FMRIB Graduate Programme)
https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FEAT/UserGuide

- Motion correction: OFF
- Slice timing correction: OFF
- Brain extraction: OFF
- Spatial smoothing FWHM (mm): 3 mm
- B0 unwarping: OFF
- Intensity normalization: ON (normally recommended off but seems to help with our monkey data)
- High-pass filtering: ON
- High-pass filter cutoff = 100 Hz
- MELODIC: OFF (as of yet, no principled way to discriminate noise and signal component)
- HRF:
  - Convolution: gamma
  - Phase (s): 0
  - Stddev (s): 1.5
  - Delay (s): 3
  - Temporal filtering: ON
  - Temporal derivative: OFF
    - though see temporal_derivatives_discussion.pdf for a detailed discussion…
- Registration: OFF
  - With the current pipeline, 1st level FEAT analysis is run on functional data in its own “native” session-space (i.e., not yet registered to the structural or to a standard template).

    - Because we use ANTs for all of our registrations (functional → structural, structural → standard template), instead of FLIRT/FNIRT, we later apply our ANTs transforms to the 1st level stats using a separate script prior to running 2nd level analysis (see below)
- Include the nuisance.txt regressors as "additional confound EVs" in the FEAT stats section
- To additionally capture motion-related variability:
    - Include a pair of *unconvolved* response regressors with the duration of 1 TR (2.28 s), in addition to the typical convolved response regressors
    - Ensure that these response regressors capture ALL responses during the task (e.g., during the ITI), NOT just those that are task-relevant

Preparing for FEAT 2nd level analysis (or other analysis)

Here we bring 1st level FEAT data into a common space, by applying our ANTs func-to-struct-to-stand warps to our COPEs and all other relevant files created during FEAT 1st level analysis.

Required files:

- FEAT folders with 1st level analysis output
- PrepareFeat_2ndLevel.sh
- WrapperPrepareFeat_2ndLevel.sh

Steps:

- Set up WrapperPrepareFeat_2ndLevel.sh to call PrepareFeat_2ndLevel.sh in a loop over your FEAT folders.

  PrepareFeat_2ndLevel.sh for each FEAT folder:
    - Creates backup copies of all COPES, VARCOPEs, mean_func, mask images
        - prefixed with "orig_"
    - applies the ANTs func-to-struct-to-stand warp to all above images
    - creates a dummy reg folder (feat.feat/reg) with:
        - symbolic links (example_func2standard.mat, standard2example_func.mat) pointing to the FSL identity transform to let FEAT 2nd level apply a dummy transform on the data (this hack is required for FEAT to run)
        - standard template image

  Input:
    - FEAT directory
    - Standard template image (e.g., group template created earlier, groupavg_2.nii.gz)
    - ANTS func-to-struct-to-standard warp image (f_mean_to_structural_to_groupavg_2_warp.nii.gz)

  Output:
    - Updated FEAT directory with all relevant files warped into standard space

Running 2nd level FEAT analysis

See FEAT user guide for more details about analysis setup (or the FMRIB Graduate Programme)
https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FEAT/UserGuide

What kind of hierarchy?

- Currently, no consensus on whether to run 2-level hierarchy (sessions from all animals pooled into one pseudo-group) or 3-level hierarchy (sessions → animal → group)
- though with enough data per animal, and enough animals, a 3-level hierarchy is best
- with a 2-level hierarchy, with all sessions pooled together, it might help to turn on “automatic outlier de-weighting”, since there may be enough data points to estimate outliers

Fixed or mixed effects?

- Mixed-effects is best
- FLAME1 is faster for initial analysis, while FLAME1+2 improves results but takes longer to run