Instructions for Running fNIRS data through Image Recon Pipeline using NeuroDOT

11 March 2021

Before you begin…

* Download the latest ‘MRI-NIRS\_Pipeline’ from github and link the scripts/matlab folder into the matlab path
* Download the latest ‘NeuroDOT\_Beta’ from github and link this repository into the matlab path (<https://github.com/WUSTL-ORL/NeuroDOT_Beta>)
* Copy the ANTS folder (see ‘misc’) to the applications folder on your computer.
* You will also need working versions of Homer2 and AtlasViewer (<https://homer-fnirs.org/>). We use the development version for this. Alternatively, you can use Homer3 and AtlasViewer (<https://github.com/BUNPC/Homer3>). Whichever you use needs to be linked into the matlab path.
* We don’t discuss the handling of Polhemus digitisations in this document. See our *digitizeR* package for that (<https://github.com/samhforbes/digitizeR>).

# Rotation and segmentation of MRI images

## Rotation

* We use the *sub\_neurologicalHighRes.nii* files for each subject
* This step can be skipped if the orientation is fairly canonical
* This first needs to be rotated so that it lands on close to an axial orientation– the *autosegment.sh* script does the final rotation after that.
* *MRI\_Rotate.sh* does this first rotation, but you need to change three things:
  1. The subject list
  2. The working directory
  3. The -rotate x y z amount. It can be negative to allow rotation in the other direction.
* This will need to be done on an individual basis for all participants, then we can move to the next step.

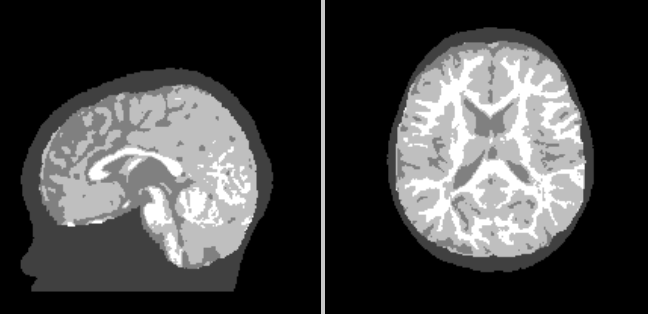
## Segmentation

* If you have a presegmented template (anatomical folder with headvol.vox) you can skip this whole step.
* If you are using a template MRI for every kid which needs to be segmented, or you have a series of already clean MRIs, you can just follow steps 1, 2 and 6.
* We need to do this in two steps (for noisy images) - one is to remove the background noise, the other is to strip the skull
* These can be run as a group if the files are similar.
* Open the *MRI\_cleanup\_Template.sh* (set this up in the same folder as *autoSegment.sh* if not running from the scripts folder – it’s often easier to run this locally because of changes you will want to make)
  1. Subject list needs to be updated
  2. Change the working directory
  3. First argument changes the step amount. We probably want step(a-350), but this can be adjusted if needed.
  4. The next two arguments shouldn’t need adjustment - they remove the outside noise, and create ${subj}\_final\_Masked.nii.
  5. We then can multiply the image by itself if required. We find this helpful for very noisy images, but unnecessary when the image is relatively clean.
  6. The last line then needs adjustment - shown below. Make sure the working directory is correct, then there are two changes you can make: One is the 0.7 below can be adjusted up or down - this is the segmentation parameter (see the first example below which uses a segmentation parameter of 0.7, whereas the second example uses 0.6). The other is you can add a -a at the end - this uses the alternate segmentation method (the alternate approach can be more aggressive for removing the skull if this was being left in).

autoSegment.sh -t /Users/administrator/Desktop/test/${subj}\_enhanced.nii -o  
//Users/administrator/Desktop/test/${subj}/ -s 0.7 -c 3 -b -m

autoSegment.sh -t /Users/administrator/Desktop/test/${subj}\_enhanced.nii -o  
//Users/administrator/Desktop/test/${subj}/ -s 0.6 -c 3 -b -m -a

* A picture containing text, invertebrate

  Description automatically generatedThe end result of this should have no skull, and needs to be checked that it has not cut away part of the brain. Equally there needs to be padded skull all around the brain (the latest version of *autosegment.sh* should have fixed this issue entirely).
* You can see the below image is a poor segmentation, as despite the white matter/grey matter definition looking good, the back of the brain has been cut off.
* In contrast this is a much better one as the brain is complete but the skull has been removed.

1. Running MCs in AtlasViewer

# Photon migration simulation preparation

## Creating anatomical folder

(Note, if you are using the latest version of AV there’s a possible OS related bug which leads to tissue properties being incorrectly passed through. This will be noticeable if you don’t have the option to input all the optical properties at the end of this step. This has also been flagged with the Homer/AV team)

* You will need to do this step unless you already have a folder named ‘anatomical’ which you are using as a template
* Once segmented (see above), you need to open MATLAB and navigate to the folder with all the segmented images for that one participant in. You do not need a NIRS or a digpts at this stage.
* Open *AtlasViewerGUI* (by typing the same into MATLAB)
* It will popup with a dialogue box - just hit close or cancel.
* When it opens AV, hit file -> import MRI anatomy
* By default, it should populate the fields - make sure hseg is with segmented head, WM is with white matter, brain is with grey matter, and CSF is with CSF.
* Hit import
* Once this is done, it will ask you to select reference points - hit yes
* These should be the same as for a digitization – e.g., NZ, IZ, CZ, LP, RP
* You *MUST* check during this step that the head is aligned with the cardinal left right as given. If not, the rotation and segmentation was not right.
* When this is done it will ask if you want the 10-20 EEG points, hit yes
* When this is done, you can close AV

## Move files

* If you already have an atlas or anatomical (ie template), start here.
* Make a new subfolder for the participant entitled *digitization*
* Copy the anatomical folder, the digpts, a NIRS file (see note below), and the HeliumScripts folder into there. An example HeliumScripts can be seen in the GitHub repository: misc/HeliumScripts\_example. This is only necessary if running on an HPC architecture, and the exact details will depend on your HPC specification.
* Use the digpts from that session - if it doesn’t exist, use a template with the same capsize (for discussion, see *digitizeR*)
* The NIRS file does not need to be from that subject, just the right number of sources/detectors. You need to check if this is right regarding short sources as well.

## Set up probes

* In MATLAB navigate to the digitization folder and open AV
* Go to tools -> register atlas to Digpoints
* Then at the bottom: Register Probe to Surface
* Go to Forward model -> set MC Parameters
* Check scattering values to use. The ones we have used are found in the optical properties excel sheet and can be copied across, but these may require investigation. Change these to what’s specified in the right hand columns of the *OpticalProperties.xlsx* if using our optical properties.
* Number of photons: 100000000 (thats *8* zeros - count them!)
* Forward model -> Generate MC Input
* Then cancel (don’t run locally on your computer)
* This gives fw …..inp files within the fw folder - check you have the right number
* IF you are running locally to test, you can say ‘file exists, try running now,’ and wait for it to finish. Once that is done you can move to complete steps, below.

# Running MCs

## Transfer files to HPC

* This section just provides a sample of how you would move these files across to run on a server using our experiences.
* Copying folder from local destination to HPC, using your ID:

scp -r /Users/administrator/Desktop/sub1002  
taw15kfu@hpc.uea.ac.uk:/gpfs/home/taw15kfu/

* I would then advise using linux to see them directly (if this is made available through your HPC provider). If you prefer to do it remotely, you sign in with:

ssh -XY (id)

* After copying homer2 over, (if not there already - make sure version is the same as the setup computer), set permissions, this allows homer to access the tMCimg file for monte carlo:

chmod 700 /gpfs/home/taw15kfu/homer2/PACKAGES/tMCimg/bin/Linux/tMCimg

* You may also need to make the tMCimg first for the HPC. This is usually done through the terminal, by navigating to the correct OS subfolder and running “make”

## Changing paths in files

* We have the fw folder and the HeliumScripts folder (again for HPC use).
* Make changes to the pathnames with the two files, and the subject lists (can do multiple at once).
* Every single inp file in the path needs to be changed
* HeliumScripts folder will be there with all the bsub files
* Also in HeliumScripts, the bsub files need paths to the Headvol to be updated
* *change\_inp.sh* (see ‘scripts’) does this, use the backslash vs forward slashes to separate out words, and the second part tells you what each one gets replaced with
* *change\_bsub.sh* (see ‘scripts’) does the same thing for the bsub files
* Check that everything is right before proceeding
* If working remotely - use nano to see it in editor instead of bash
* Run these two scripts and check the output.

bash change\_bsub.sh

bash change\_inp.sh

## Running MC

* To run, first edit the subject list in *ind\_sub\_job.sh*

bash ind\_sub\_job.sh

* Check bjobs, if it has run suspiciously quickly then there is a problem
* Once it’s done, bjobs will show no active jobs
* Then we can transfer everything back to the local destination

## Move back and complete steps

* Once everything finishes, copy folder from HPC to local destination (reverse of the above):

scp -r taw15kfu@hpc.uea.ac.uk:/gpfs/home/taw15kfu/NIRS\_Project/  
/Users/administrator/Desktop

* Do the register atlas to digpts and register probe to surface again.
* After that, you need to cd to the *digitization/fw/* folder of that participant in the terminal and run the following (unless run locally):

touch \*

* This updates the files so they are modified recently. Without this it might say input is newer than output, and wipe the output, or ask you to rerun it.
* If you try to run with Forward model -> Generate MC Input, it will say there is already an input, so hit OK
* *IMPORTANT* if it says anything else hit cancel and check that the fw folder contains the new MC files
* Tick Forward model -> enable sensitivity matrix volume
* Then go to Forward model -> Generate load sensitivity profile
* These will take a while to load. It may ask if the head needs downsampling. If it asks, say yes, and accept the downsampled head.
* This generates a coloured picture which is your sensitivity profile. Check this to see that it matches the input. If there is little to no sensitivity around the optodes, check that the tissue properties were input correctly.
* Before closing, check that the *Adot* files are in the *digitization/fw* folder

1. **Create Input File for NeuroDOT pipeline**

* There is a template in the ‘files’ folder called ‘Sample\_SubjListGroup.xlsx’. It is best to start creating the input file in Excel to allow for easy updating.
  + The Excel template uses data in the subjectID column to fill in the correct path details for each column. **So be sure to keep the formulas intact.**
  + To edit the file, paste in your subject list in column A and delete any extra rows.
  + Next, update the paths in each column in row 1 and ‘fill down’. Note that data might be in different places, so it is easy to just ‘fill down’ the relevant rows with different paths as needed. For instance, if some kids use templates, you can group them together and then point to a single template (headvol) file in column F and fill down from that row.
  + Note: we have included the full path to each folder location to allow maximal flexibility, so you can sort your data in any preferred structure.
* Once the file has been updated, save it as a .csv file. Then open the .csv with a text editor like BBEdit or TextWrangler. Find the commas and replace with space. Then save as a .prn file. **Be sure to specify that line breaks are set in Unix format.** 
  + Note: we’ve tried saving a space-delimited file with Excel but have found it unreliable.
* Column specification:
  + SubjectID
  + Full path to template .nirs file for the project. This was created by saving one example .nirs file with a common name. This is just needed to pull header info from the .nirs file (e.g., number of channels). Note: this can be a .snirf file if you prefer.
  + Full path to subject-specific MC folder (where you put the output from AtlasViewer for each subject)
  + Full path to the NIRS folder containing the processed fNIRS data
  + Full path to the desired ‘subject’ output folder (where you want the ImageRecon files to be written)
  + Full path to the headvol file for this subject (e.g., headvol.vox – could be subject-specific or a template)
  + Full path to the group T1 template to use for group registration
  + Full path to the group T1 brain mask used for group registration
  + Full path to the ‘group’ output folder used for group registration
  + Type of registration for this subject. Options are:
    - Subject (i.e., subject-specific MRI)
    - AtlasR (i.e., register this subject to the group template)
    - AtlasW (i.e., warp this subject using the atlas specified in the next column)
  + Name of the registration files. Mapping to the options above, an example would be:
    - The subject ID (from column 1)
    - The name of the atlas you are creating with the AtlasR option
    - The name of the atlas you want to use for warping
  + Flag to indicate if the subject-specific T1 registration file is an hseg file
  + Flag to indicate if the atlas-specific T1 registration file is an hseg file
  + The subject T1 image used for registration
  + The subject T1 brain mask used for registration
  + The output folder to be used post-group analysis (see steps 16/17)
* Sample data structure (see image below):
  + Child = base folder. Contains the Subject\_list\_Child\_final\_1Subj.prn input file and the sample .nirs file for the project.
  + 32HWB\_ChildNIRS: location of the .nirs files
  + Child\_MCs: the AtlasViewer output for each subject (32HWB036 is an example)
  + GroupAnalysis: will contain the analysis output created in steps 16/17
  + ImageRecon\_Child: will hold the subject-specific image-reconstructed fNIRS files
  + ImageRecon\_Child\_Group: will hold the group-registered beta maps and transformations from subject-space to group-space
  + MNI152\_T1\_2mm: the folder that contains the group T1 and brain mask used for group registration

****

1. **Creating sensitivity profiles in niftii format for NeuroDOT Light Model**

* Run **transformSensProfileToAnat\_NeuroDOT.sh**.
* This script takes a subject list file (see input file above).
* Comments:
  + This script does not trim the sensitivity data to the brain. Rather, it uses the entire profile including the scalp and surface tissues.
  + The script packs all of the sensitivity data into a single .nii file with sub-bricks by channel (in an order corresponding to the measurement list in the .nirs file). HbO is the first set; HbR is second. Note that channels are listed consecutively to keep them distinctive (e.g., HbO might be channels 1-20; HbR channels 21-40).
  + A light thresholding is applied to the sensitivity volumes (0.000001). This was useful because a few volumes fill the entire brain if left un-thresholded.
  + We also downsample to 2mm resolution to keep the file size reasonable.
* Note: the channel information is obtained from the template .nirs file you provide (see above). This information is pulled from the SD data structure.
* To run the script, do the following in a terminal window:
  + Navigate to the ‘Data’ folder for the project in terminal (in the example above, this would be the ‘Child’ folder). This is where your subject list file should be located.
  + Link in the Github scripts: example…

export PATH=${PATH}:/Users/DDLab/Documents/GitHub/MRI-NIRS\_Pipeline/scripts

* + Link in matlab: example…

export PATH=${PATH}:/Applications/MATLAB\_R2018b.app/bin

* + Run: transformSensProfileToAnat\_NeuroDOT.sh {filename} – example…

transformSensProfileToAnat\_NeuroDOT.sh SubjectList.prn

* Note: Running this code twice on the same set of files results in twice the number of channels in the sub-bricks even though the output in the terminal window says that the file has not been over-written. So, make sure to either delete or rename the old files (headvol.nii, Adot.nii, etc.) before re-running.
* In some of our files, the orientation of the files coming out of AtlasViewer was off. This was caused by earlier versions of AtlasViewer not saving the ‘2ras’ transformation. Thus, we created a utility to move the files into a canonical orientation. To run this fix (should it be needed), do the following:
  + Link the utilities path:

export PATH=${PATH}:/Users/nfb15zpu/Documents/GitHub/MRI-NIRS\_Pipeline/scripts/Utilities

registerCommonDriver\_FixOrientation.sh [input file name]

1. **NIRS Processing in EasyNIRS / Homer2 / Homer3:**

* First step in NIRS processing is to sort the stim marks to mark events for the GLM.
* Next, analyse the .nirs files using EasyNIRS (or analyse your .snirf files using Homer3). This is done using a configuration file (.cfg file) that loads the specific processing options desired. Here’s a typical sequence:
  + hmrIntensity2OD [converts to optical density]
  + enPruneChannels SD [prunes channels with low signal-to-noise]
  + hmrMotionCorrectPCArecurse [motion correction using targeted PCA]
  + hmrMotionArtifactByChannel [flags epochs that have uncorrected motion artefact]
  + enStimRejection [turns ‘off’ stim marks where HRF window falls within a motion epoch]
  + hmrBandpassFilt [band pass filter]
  + hmrOD2Conc [converts optical density to concentration data] – this last step is useful for viewing data in EasyNIRS but it technically not needed by NeuroDOT (which only uses the procResult.dod data)
  + Note: we also often run a GLM in EasyNIRS using hmrDeconvHRF\_DriftSS. This returns beta values for each channel. Again, NeuroDOT doesn’t use this info, but it can be useful for sanity checking the GLM run in NeuroDOT.
  + Note 2: NeuroDOT has an option to use Scott Prahl’s optical extinction coefficients (see below). These differ from the default values in Homer2/EasyNIRS. Be careful here if you want to compare the Homer2 GLM with the values coming out of NeuroDOT. In the steps below, you can select ‘0’ to avoid using the Prahl values. In this case, our code assumes you wrote the desired extinction coefficients into the .nirs file. This is not an option if you are using a .snirf file as there is no extinction coefficients field. Thus, if you are using .snirf, our pipeline will default to the Prahl values.

1. **Image Reconstruction using NeuroDOT**

* Open matlab
* Navigate to Data folder (e.g., the ‘Child’ folder in the example above)
* Make sure the Github ‘scripts’ folder and subfolders are linked into the matlab path (check the ‘set path’ toolbar).
* Make sure the Github ‘NeuroDOT\_Beta’ folder and subfolders are linked into the matlab path.
* Run **ImageRecon\_NeuroDOT.m** in matlab. Example…

ImageRecon\_NeuroDOT('SubjectList.prn', 25,10,20,40,30,0.05,1,1)

* Comments on params:
  + 1: input file
  + 2: ‘25’ specifies the old sampling frequency (25Hz)
  + 3: 10 is the new sampling frequency for the final 4D image recon file. If matlab crashes because it runs out of memory, you’ll have to downsample the data from, say, 25Hz to 10Hz.
  + 4: 20 is padding start -- to make the image recon data as trim as possible, we only reconstruct data from, say, 20s before the first stim event. If you are running into problems with finding the perfect padding values that fit all your subject datasets, just replace with a high number and the whole times series will be included.
  + 5: 40 is padding end – reconstruct from last stim + 40s. This is longer than front-end padding to allow, for instance, 18s of neural response after the last stim event. If you are running into problems with finding the perfect padding values that fit all your subject datasets, just replace with a high number and the whole times series will be included.
  + 6: baseSDmm – the ‘base’ separation between source and detector. This is not currently used below, but could be used in some NeuroDOT functions.
  + 7: FFRproportion – threshold the data based on values > FFRproportion (a proportion of the flat field reconstruction)
  + 8: usePrahl : Use 1 if you want to use Prahl’s extinction coefficients; only use 0 if you have coefficients stored in the .nirs file and you want to use those.
    - Coefficients should be stored in SD.extCoef in the .nirs file. We assume the input coefficients are in 1/molar (the default in Homer2), and we convert to 1/millimolar.
  + 9: GSR: Use 1 if you want to run global signal regression; Use 0 is you do not want to do this.
* Note: it is important to carefully check that the data in the .nirs file is correctly mapped to the NeuroDOT data structures. To facilitate this mapping, here is what NeuroDOT uses from the .nirs file. Note: the mapping to .snirf and associated .mat files has been sorted, so if you are using .snirf and Homer3, all is good.
  + SD.MeasList (the list of channels)
  + SD.Lambda (the wavelengths)
  + procResult.SD.MeasListAct (which channels are included [1] vs. pruned [0])
  + procResult.s (the stim timings)
  + procResult.dod (the processed optical density data)

1. **Running Individual-level GLMs using NeuroDOT**

* Open matlab
* Navigate to Data folder
* Make sure the Github ‘scripts’ folder and subfolders are linked into the matlab path (check the ‘set path’ toolbar).
* Make sure the Github ‘NeuroDOT\_Beta’ folder and subfolders are linked into the matlab path.
* Run **RunGLM\_NeuroDOT.m** in matlab. Example…

RunGLM\_NeuroDOT('SubjectList.prn', [1,2,4,5], 100, ‘WLRN’, 10, ‘’)

* Comments:
  + Input 1: subject list input file
  + Input 2: regressor list you want included in this particular GLM
    - Regressors are specified in the ‘s’ matrix in the .nirs file. We typically code up many more events than is included in any particular GLM. The regressor list specifies which ones to include.
  + Input 3: the duration of each event (the duration of the boxcar in the GLM). This is specified in time steps (so duration \* sampling frequency). As an example, the duration of a trial in the VWM task is 10 s, so we would enter 100 (10\*10Hz). In other tasks, we might have, for instance, a brief naming event, so we would enter 10 (1s\*10Hz) – as 10Hz is the new sampling frequency. Note: if the data has been downsampled, the duration should be in the downsampled frequency.
  + Input 4: a tag for each output filename to indicate the ‘name’ of the GLM you are running (e.g., ‘WLRN’ for ‘word learning GLM’).
  + Input 5: the sampling frequency used during image recon (e.g., 25Hz if you didn’t have to downsample; or 10Hz if you did)
  + Input 6: if empty (‘’), use the default NeuroDOT HRF (see ‘hrf\_DOT3.mat’ in NeuroDOT\_Beta/Support\_Files/GLM). Otherwise, specify the HRF file to use. If path is not specified, be sure this is located in your main folder. Also be sure the size is specified correctly (the default NeuroDOT HRF is 30s long, sampled at 1s resolution…)
* This script will output betamaps for each condition for oxy and deoxy in the headvol space (i.e., subject space).

1. **Move to Group Space (e.g., CustomMNI)**

[Vince: could you add some details here about the files we are using; good options, etc.]

* Navigate to Data folder in terminal
* Link in the Github scripts: example…

export PATH=${PATH}:/Users/DDLab/Documents/GitHub/MRI-NIRS\_Pipeline/scripts

* Link ANTS tools (a copy can be found in the ‘misc’ folder): example…

export PATH=${PATH}:/Applications/ANTS/bin

* Run registerCommonDriver.sh. This takes the input file as input:

registerCommonDriver.sh [inputfile name]

1. **Create a Group Mask**

Once all the data are in a common space, you’ll want to find the common voxels that should be included in the group analysis. For instance, if some voxels only have non-zero beta values for a few participants, you might want to exclude these voxels due to low N. We usually include all voxels that have non-zero beta values for, say, 60-70% of participants.

* Github ‘scripts/AFNI\_Examples’ has an example called ‘NIRS\_Mask\_ANOVA\_Example.sh’ that can be edited. The goal of this script is to sum up how many subjects contribute at least some data (across any condition) to each voxel. This can then be thresholded in the final step based on a percentage of subjects (e.g., 60-70%).
* Once the mask has been created, it can then be multiplied with all the group-level images to filter out any ‘fringy’ voxels with sparse data.

1. **Group Analysis**

The next step is to analyse your group-level data. We often using ANOVA or linear mixed-effects models for this. AFNI examples of both can be found below.

* Sample script: 3dMVM\_Example2.txt (runs a within-subject ANOVA)
* Include -GES option to get effect size estimates
* Include -resid option to estimate the spatial autocorrelation (ACF)
* If one or more within-subjects variables, include -wsMVT to get multi-variate estimates of the within-subjects effects
* Include -SS\_type 2 (a bit more powerful)
* Example:

3dMVM -GES -prefix GatesPLAgeSSHb\_gesConHb3 -jobs 2 \

-resid GatesPLResidConHb3.nii \

-mask Infants\_IntersecMask.nii \

-bsVars 'Age' \

-wsVars "SS\*Hb" \

-wsMVT \

-SS\_type 2 \

-dataTable \

* There is also a 3dLME\_Example.sh which runs a linear mixed-effects model in AFNI.

1. **Concatenate ANOVA results**

Once the group-analysis is complete, you’ll have results showing clusters with significant effects. Typically, you’ll want to extract the beta values from each cluster for each participant so you can, for instance, make plots of the data. A first step toward this end in AFNI is to concatenate the data files.

* Sample script in “AFNI\_Examples’: conCat\_ANOVA\_Example.sh
* This prepares the data for the next steps
* Basically, packs the beta images into a single file for each beta map you are contrasting in the ANOVA and each chromophore.
* NOTE: if some participants don’t have a betamap for a condition, you’ll need to fill in an empty image for these sub-bricks. Otherwise, subsequent steps (e.g., ROIstats) will throw an error because the concat code won’t write an output file if it can’t find the relevant cond image.
  + An example of how to concat with empty images is in ‘conCat\_MissingCond.sh’
  + To create an empty image, you can run:

3dcalc -a 09IND390B\_India\_cond1\_Unmasked\_oxy\_ND\_To\_Atlas\_ClipToBrain.nii.gz -expr 'a\*0' -prefix Empty.nii.gz

1. **Determine thresholds for family-wise error correction**

* Now we need to correct for multiple comparisons. For us, this involves two steps.
* First, you need to estimate the spatial noise in the data. Do this by running an analysis on the residuals from the ANOVA.

3dFWHMx -input GatesPLResid.nii -mask Infants\_IntersecMask.nii -acf

* This computes the spatial auto-correlation in the data (residuals) and fits parameters to estimate the ACF (see Cox et al., 2017). The result is a nice plot of the ACF and 3 output parameters estimating the ACF.
* Next, use the ACF parameters to run 3dClustSim. This will set a cluster size threshold. Only clusters >= the size threshold can be considered significant after controlling for family-wise error.
* 3dClustSim requires setting a voxelwise p value and an alpha level. Based on Cox et al., 2017, we’ll use p = 0.01 and alpha = 0.05. Example:

3dClustSim -mask Infants\_IntersecMask.nii -acf 0.736355 6.45424 2.94415 -pthr 0.01 -athr 0.05 -iter 10000 -nodec -quiet

* This command will run simulations estimating the cluster size threshold for different ways to define the cluster. See the AFNI help for details. As an example, we might select NN1 (first nearest neighbor clustering = voxels cluster if faces touching), two-sided (non-directional hypotheses), and not bi-sided

1. **Threshold the ANOVA results**

* Sample script: 3dClusterize\_Child\_WLCT.sh
* This applies criteria from 3dClustSim. The example code shows how to loop through all the effects of interest.
* 3dclusterize takes the input image, specifies the sub-brick to threshold (see ithr), takes the mask input, NN desired, number of voxels specified by 3dClustSim (clust\_nvox), and saves the data as clusters with increasing numbers (pref\_map).
* -1sided RIGHT\_Tail : since we are using F stats, this should be the F critical determined by using the cluster threshold in the AFNI gui. It is 1sided here because F is always positive, but the critical F is conceptually two-sided (this was confirmed with Gang Chen at AFNI).
* The code also writes the GES stats (when relevant)

3dROIstats -mask Infants\_cPL\_gesConHb3/0.01/${effect}/clust\_order\_${effect}\_l1\_01.nii GatesPLAgeSSHb\_gesConHb3+tlrc'[16]' > \

Infants\_cPL\_gesConHb3/0.01/${effect}/${effect}\_l1\_01\_ges.1D

* Additional helpful commands. Whereami gives labels from the MNI atlas if you have group data in that space. This reads in the values in columns 13, 14, 15 from the 1D file created above (note: indices start at 0 in AFNI, so columns 14, 15, 16).

whereami -coord\_file Infants\_cPL\_gesConHb3/0.01/${effect}/${effect}\_l1\_01.1D'[13,14,15]' -tab -space MNI > Infants\_cPL\_gesConHb3/0.01/${effect}/${effect}\_l1\_01\_wai.1D

* Finally, it’s useful to copy the underlay image to your directory so you can view the clusterized .nii file.

cp MNI\_avg152T1+tlrc.BRIK Infants\_cPL\_gesConHb3/0.01/${effect}/

cp MNI\_avg152T1+tlrc.HEAD Infants\_cPL\_gesConHb3/0.01/${effect}/

1. **ROIStats on ANOVA results**

* Sample script: ROI\_stats\_Example2.sh
* This computes the average beta values for each cluster for each participant. These data can then be used to run, for instance, correlations with behavioural variables.
* The example code loops through all participants (in each age group or overall, depending on how you’ve set up the data). It also loops through all the effects included in the previous step.
* The key call is to 3dROIstats. A key parameter here is the list of bricks that need analysing. The brick list is [0,1,2,3,4,5] below. This list was created when we concatenated the file in step 11. So in the example below, I’m averaging the beta values for 0 = oxy load 1, 1 = deoxy load 1, 2 = oxy load 2, 3 = deoxy load 2, 4 = oxy load 3, 5 = deoxy load 3.

3dROIstats -mask Infants\_cPL\_gesConHb3/0.01/${region}/clust\_order\_${region}\_l1\_01.nii Infants\_cPL\_gesConHb3/Concat\_4mo\_${names}.nii'[0,1,2,3,4,5]' > \

Infants\_cPL\_gesConHb3/0.01/${region}/ROIstats/${region}\_4mo\_${names}.1D

* Note: we generally use the ‘nzmean’ option when pulling out ROIstats (see the script ‘ROI\_stats\_ANOVA\_Example2.sh’). This option outputs both regular means (column 1) and the mean of all non-zero voxels in the mask (i.e., nzmeans; see column 2) in the resultant 1D files. Thus, if you have one cluster, you’ll get two output columns; two clusters, will yield 4 output columns, etc.
* If you then continue to the ‘CombinedROIStats\_Example2.m’ file (see next step), this code will pull together all your data, including both sets of means (basically, this code was written to pull out all columns from the 1D files). Importantly, each column will be labelled as a cluster. Thus, if you have two significant clusters and you use the nzmean option…
  + cluster 1 = mean for cluster 1
  + cluster 2 = nzmean for cluster 1
  + cluster 3 = mean for cluster 2
  + cluster 4 = nzmean for cluster 2
* If you then move on to the R code to plot your ROIstats data (‘NIRS\_ROIstats\_Plots.R’), you need to take care when selecting your clusters to plot, selecting, for example, clusters 2 and 4 in the example above (assuming you want to plot the nzmean data).

1. **CombinedROIStats**

* Sample script: ROI\_stats\_ANOVA\_Example2.m
* This organizes the output of ROIstats into a file with the average beta values for each cluster for each effect for each participant. These data can then be used to run, for instance, correlations with behavioural variables.
* The example code loops through all participants and all effects. The output is a long form .csv file that can be read into R for follow-up analyses.
* For details on plotting, see ‘scripts/R\_Examples’ folder: NIRS\_ROIstats\_Plots.R
* Note: if you have effects with no clusters, it is useful to change the effect folder name, adding ‘\_NoClusters’. Otherwise, this code with open the files created and throw an error because there are no data in the subject-specific 1D files.

1. **Extract Time Series Data from Image Recon Files from significant clusters**

After group analysis, it can be useful to extract individual time series from the significant clusters to show a block average time series that NIRS folks are used to. Here’s how to do this.

1. Start with the clust\_order\*.nii file for the effect you want to examine

* e.g., clust\_order\_6Hb\_l1\_01.nii
* copy these ‘clust’ files to the folder that will store the output of the current analysis (this must be the final column in your input file)

1. Run registerInvertCluster.sh

* Then it creates a subject-specific clust file by inverting the original affine transformation applied to the data to get to group space. So the end result is a mask with the clusters in each ‘clust’ file moved to subject-specific space.

bash registerInvertCluster.sh SubjList.prn

1. run ExtractHbFromMask.m

* To be used for extracting a weighted average time series for each subject for each effect from each cluster for each regressor for each chromophore
* Input example in matlab. First navigate to your base folder (e.g., ‘Child’). Then run…

ExtractHbFromMask('SubjList.prn',[1,2,3],'Test',10,18,3,0,0,3)

* Input file name (.prn file)
* Regressor List – the conditions from the original GLM that you want to extract block averages for
* A name for the analysis (for the output file). The name is also a column in the .csv file which can be useful for tagging each run through this code or tagging a specific type of subject (e.g., if you ultimately want to combine data across years)
* newSamplingFreq: the sampling frequency of the image recon files
* HRFDuration: the duration of the HRF window (in seconds)
* MaxMaskValue = the max mask cluster value across all clust\_order files to analyse (i.e., some effects might have multiple clusters in them)
* checkAlignment – flag to display the headvol, the cluster mask, and the ‘GoodVox’ as overlays. Useful for checking a handful of subjects initially, but set to 0 for full runs. When the image comes up, you can move the crosshairs around and ‘click’ to update each image. Hit ‘q’ when hovering over the window to move out of the interactive display and keep processing.
* showHRF – flag to plot the resultant time series data for each subject. Again, most useful for initial explorations, but set to 0 for full runs.
* BaselineDuration – subtract the mean of the first X seconds from the block average time series. This is useful for fast event-related designs. If 0, this has no effect.
* This matlab file will output a TsHb file containing the time series data for each chromophore, for each condition, for each cluster, for each effect including the number of trials (N), the time series position (time), the weighted mean (weighted by the number of trials per run), the sum, and the standard error.
* You can read the TsHb .csv file into R to make pretty plots. An example can be found in the ‘R\_Examples’ folder. Copy ‘NIRS\_ROITimeSeries\_Plots.R’ to your analysis folder. This will make a plot of your data once you set the desired input file name, cluster and effect.

1. **Check correlation between channel-based fNIRS and image-based reconstruction**

In our paper validating this pipeline, one of the things we did was to look at the correlation between the channel-based NIRS time series and the image-reconstructed time series. To do this, we computed the maximum of the sensitivity volume for each channel in the brain (i.e., the max was constrained to be in the cortex) for each sample subject. We then placed a 1cm radius sphere at the max value and clipped this sphere to the cortex. Next, we plotted the channel-based fNIRS data against the image-reconstructed fNIRS data from each sphere (i.e., from each channel in image space). In particular, we extracted the weighted mean across the sphere. Finally, we computed the correlation for each channel/sphere pair.

1. To run this analysis, you can create a sphere for each channel using:

bash registerInvertCreateSphere.sh SubjectList.prn

* This will copy the AdotVol and headvol files to the analysis folder for each subject.
* Then it will write a ‘Mvalues.1D’ file with the max value in cortex for each sensitivity volume in subject-space
* Next it will write ‘clust\_order\_Peaks.nii’ files for each channel. These images contain the spherical ROI for each channel that it then clipped to the brain (see ‘BrainOnly’ files).
* Finally, the script will create dummy ‘clust\_order’ files that are needed in the next step.

1. run ExtractHbFromMaskByChannel.m

* This code will plot the channel-based fNIRS data (red) against the image-based fNIRS data (black) for each channel and each subject
* It will also write a ‘CorrelationsByChannel.csv’ file with the correlation per channel for each chromophore for each subject

ExtractHbFromMaskByChannel('SubjList.prn',25,10,20,40,18,1,0,1,3)

* Input file name (.prn file)
* oldSamplingFrequency (e.g., 25Hz)
* newSamplingFrequency (e.g., 10Hz)
* paddingStart (e.g., 20s)
* paddingEnd (e.g., 40s)
* HRFDuration (e.g., 18s)
* MaxClustValue – this is the maximum value in all the clust\_order files you want to process. You might, for instance, have one clust\_order file that has 3 significant clusters. The others might have only 1 cluster. In this case, enter 3 here.
* checkAlignment – plots the sensitivity volume and then the ROI to ensure everything lines up in subject space
* GeneratePlots – flag [0,1] to indicate if you want to plot the data for each channel and chromophore per subject. Note that the number of plots can be quite large if you are running this analysis on a large number of subjects. So only use this option if you are running it on a handful of subjects.
* nPlotsPerFig – number of plots per figure. 3 is a good value if you want to view the data in detail.
* This matlab script will create Figures for each comparison for each chromophore for each .nirs run (X per plot – see flag). It will also create a ‘CorrelationsByChannel.csv’ file.
* The .csv file can be read into R using ‘NIRS\_Compare\_ChannelvImage.R’. This will make the pretty plots we included in the paper. Hopefully, most of the correlation values are high!