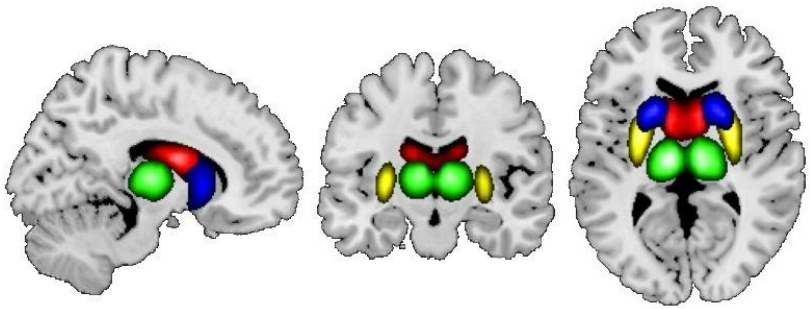
A walkthrough of GIFT for fMRI data

How to do ICA and like it

Elena A. Allen

University of Bergen
& the Mind Research Network

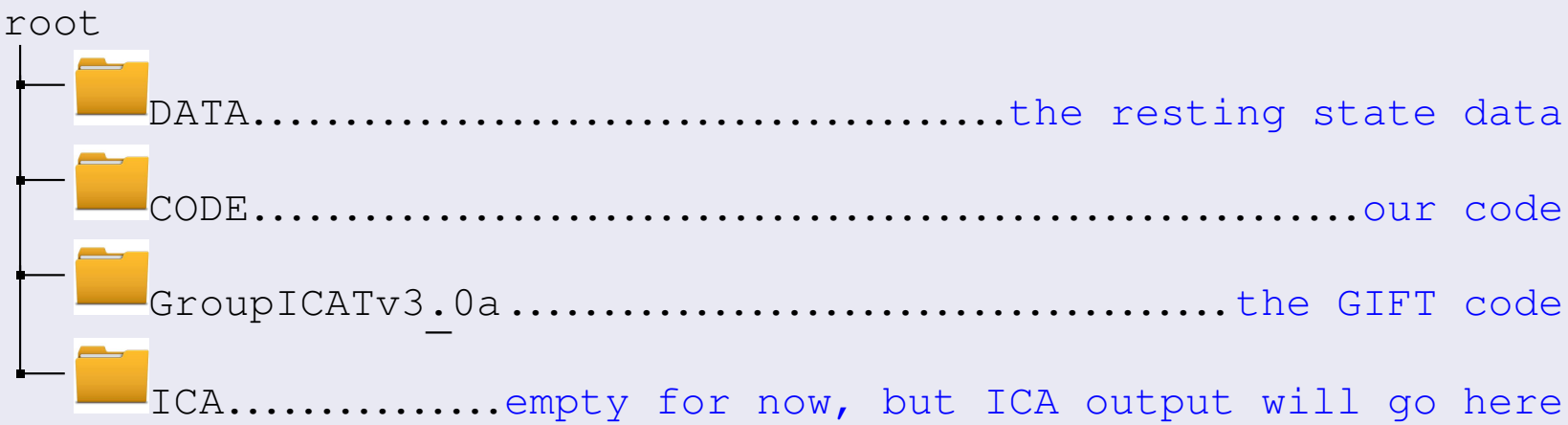
Sept 26, 2013



Group ICA of fMRI Toolbox (GIFT) is a MATLAB toolbox that provides all the necessary functions to perform ICA on single-subject and multi-subject fMRI and EEG datasets. Developed for users at all levels, GIFT has an extensive graphical-user interface (GUI) that offers full control over analysis parameters. Beyond the GUI, more advanced users can access GIFT functions directly, increasing flexibility. Today we'll work both with the GUI and at the command line to peer under the GIFT hood and work through examples that address the practical issues of ICA. All the examples and MATLAB code used in this presentation are provided in `gift_demo.m`. Please feel free to follow along there and run the examples yourself.

Overview

the following structure will be helpful



garbage in = garbage out

Just like every other analysis, the quality of your ICA results will depend on the quality of the data. ICA can do a good job of separating noise from sources of interest, but it is not magic; including some bad apples will adversely affect the decomposition.
For each dataset you should know:

- Was spatial normalization successful?
- Are there large regions of signal dropout?
- Is there excessive motion (several millimeters) or small but continuous movements throughout the scan?

Correct what you can and discard what you must.

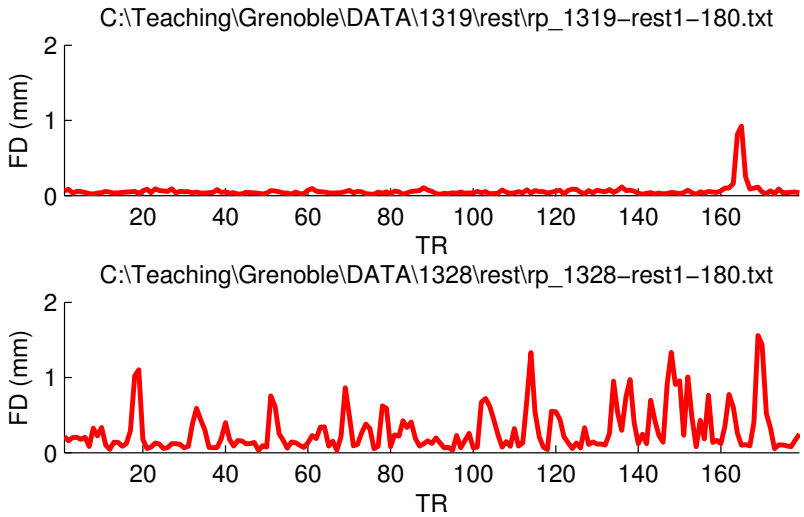
Example: assess motion

One of the many ways to assess motion is to compute the absolute displacement between TRs, often referred to as the framewise displacement (FD). Below is a little MATLAB code to compute the FD from the realignment parameters.

```
% load in the realignment parameters
% estimated in preprocessing
rp = load(motfiles{ii});
% rp is [180 TRs x 6 directions]
% (x, y, z, pitch, roll, yaw)

% convert rotations (radians) into translations (mm)
rad = 50; % assume a head radius of 50 mm
rp(:,4:6) = rad*tan(rp(:,4:6));

% compute the FD (Euclidean distance frame-to-frame)
FD{ii} = sqrt(sum(diff(rp).^2,2));
% sqrt(dx^2 + dy^2 + dz^2 + dphi^2...)
```



Behind the mask

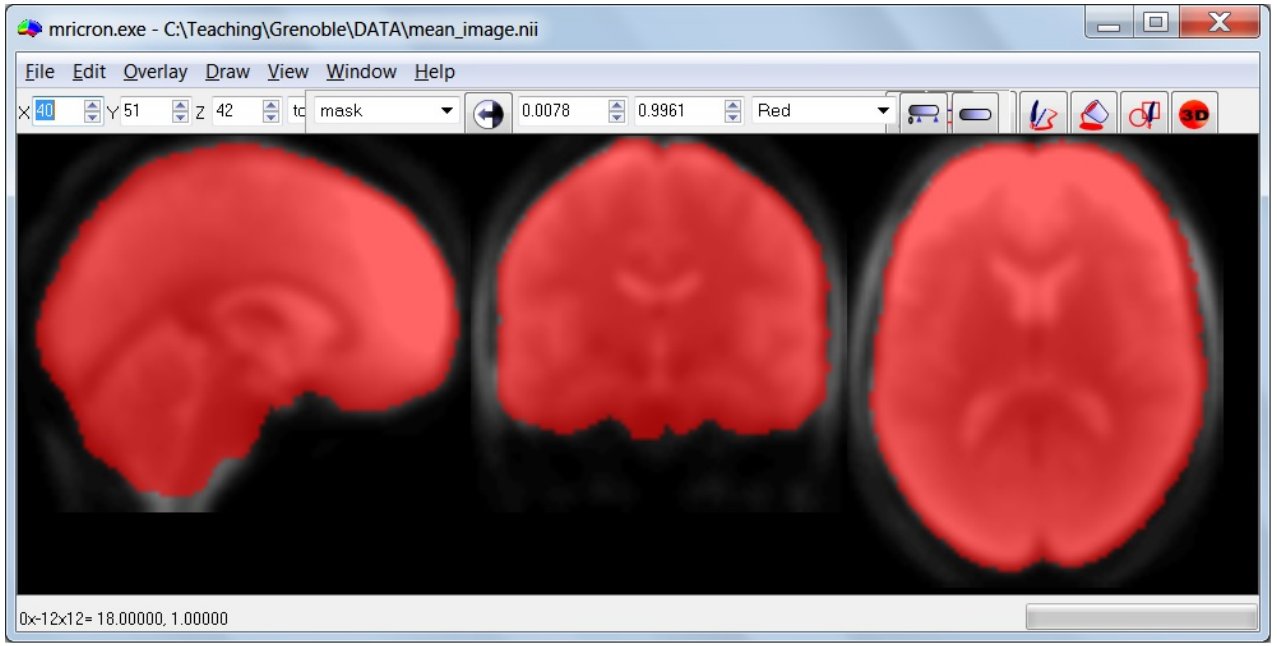
GIFT will generate a default mask if none is provided but it's a good idea to create your own ahead of time or at the very least **check the mask** after it's been generated.

The default GIFT's default mask uses the **intersection** of masks determined at the individual, based on keeping voxels whose intensity is greater than the average intensity. For large multi-subject datasets some individuals are certain to have low signal in areas of interest; by using the default mask you run risk of missing important areas (like the cerebellum or motor cortex).

Alternatives You can make your own mask based on a template image (perhaps used for registration), a mean image (the average brain in your sample), or use less stringent criteria than the intersection. AFNI's 3dAutomask or FSL's Brain Extraction Tool (BET, also included in MRICRON) can be helpful.

Example: check your mask

Below is an example of visually checking the mask by overlaying the binary mask on the mean image.



Model order

- The ICA algorithm will separate the data into as many components as you dictate. So what's the right number?
- There probably isn't a single "right" number; fMRI data can be sensibly decomposed into a relatively small number components (10 to 20) or a relativey large number (70 to 100) because of the brain's heirarchical organization. We can use theoretical or empirical approaches (both implemented in GIFT) to estimate a reasonable model order, but don't stress about it too much. Using 73 instead of 75 components will have little effect on the inferences we make.

The default GIFT will use the minimum description length (MDL, modified for spatial dependence) to estimate the number of components in each dataset, then will take the mean across datasets.

Alternatives You can use the GIFT function `icatb_estimate_dimension` to estimate the number of components using several different information theoretic criteria (ITC) ahead of time, and select a model order based on the distribution of values across datasets. Or, you can use empirical methods based on the stability of the decomposition.

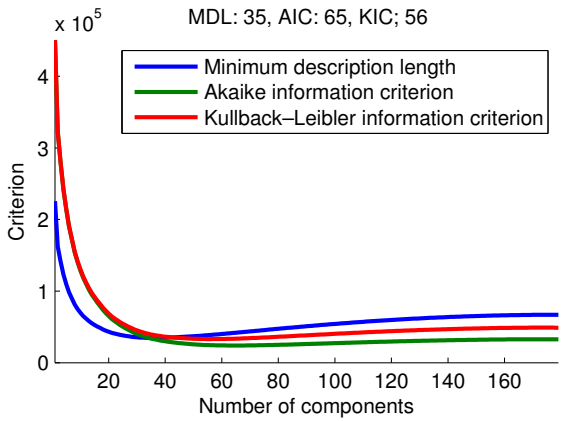
Example: estimate model order

Below is an example of how to estimate model order from one subject. To apply to all subjects, just wrap in a `for` loop.

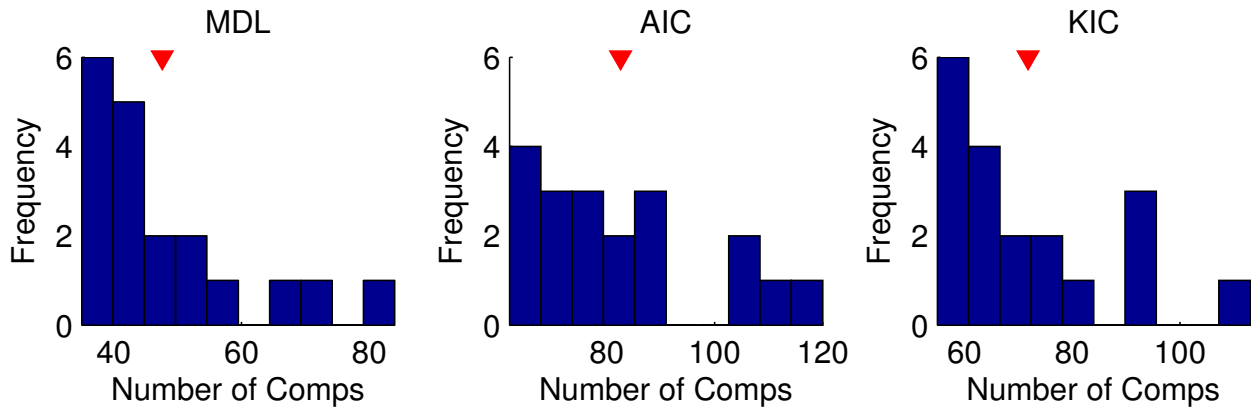
```
fnames = ...
     rdir(fullfile(DATADIR,'*', 'rest', 's5war*180.nii'));
% a generous mask made from this data earlier
maskfile = fullfile(DATADIR, 'mask.nii');

% For the first subject:
[comp_est, mdl, aic, kic] = ...
     icatb_estimate_dimension(fnames{1}, maskfile);
% number of components is the minimum of the curve
[mv,mdl_mind]=min(mdl);
[mv,aic_mind]=min(aic);
[mv,kic_mind]=min(kic);
```

for the first subject



ITC estimates for all subjects



Overview

Enough prep work.
Let's actually set up an ICA in GIFT using data provided by Tom Zeffiro.

This dataset

Resting-state scans collected from 19 healthy subjects	*small dataset, intersubject spatial variability may limit detection of smaller components*
Siemens 3T scanner with a 32 channel head coil	*should be excellent quality*
TR $=$ 3.34 s	*slow sampling (Nyquist frequency $=$0.15 Hz), frequency profiles will be limited*
180 volumes per subject (scan length of 10 minutes)	*fair estimation of single-subject maps and time courses*
2.5 mm isotropic voxels, resampled to 2 mm	*excellent spatial resolution*
spatial smoothing filter FWHM $=$ 5 mm	*good, won't blur spatial profiles though for such a small sample 7 mm is also reasonable*

The acquisition parameters, dataset dimensions, and preprocessing **will** affect the ICA decomposition.

Analysis steps

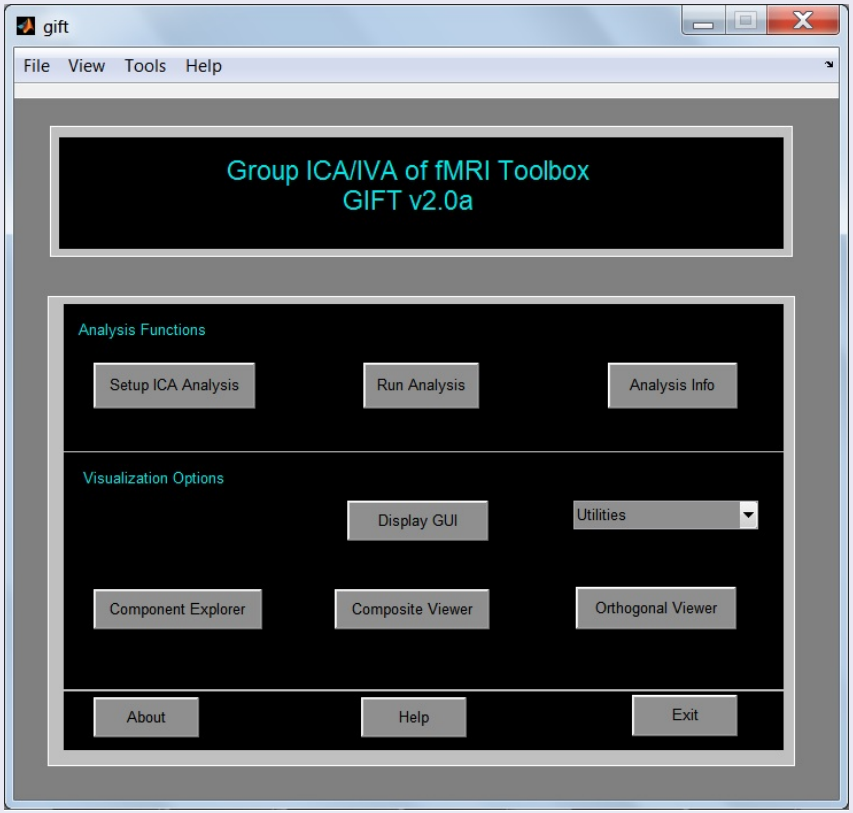
A typical analysis in GIFT comprises 6 steps:

1. Preprocessing
2. Single-subject PCA
3. Group PCA
4. ICA
5. Back-reconstruction of subject specific components
6. Scaling of subject components

To set up the analysis, we choose parameteters for each of these steps.

start the GUI

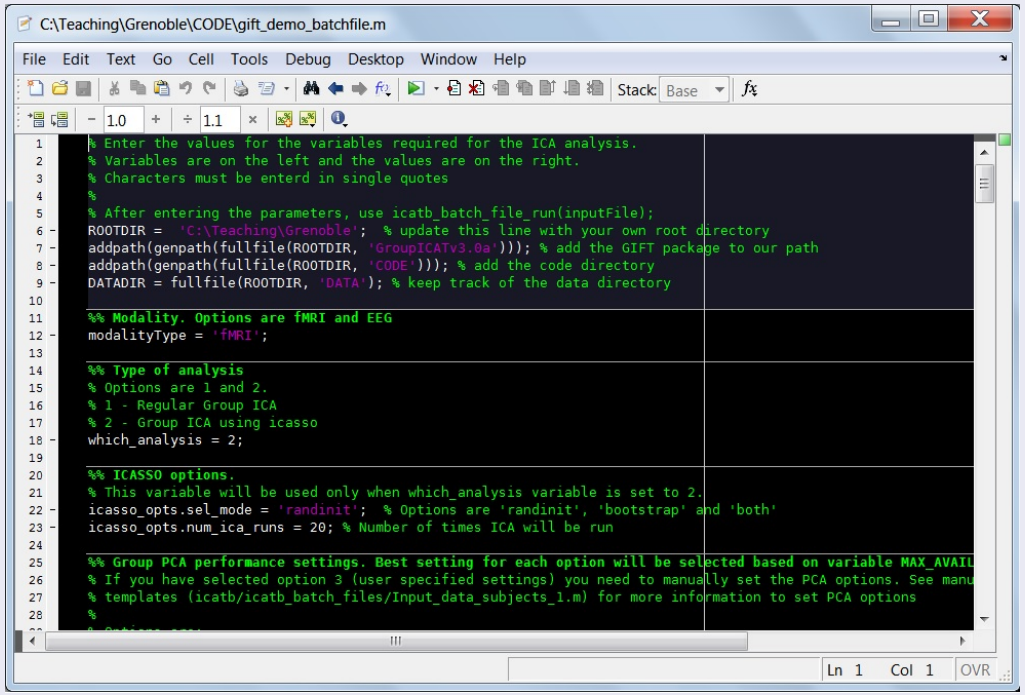
```
>> gift
```



working from the command line

Rather than push buttons, you can set up the analysis with a simple batch script.

```
>> edit('gift_demo_batchfile.m')
```



Critical parameters

Step	Name	My Choice	Explanation
Preprocessing	preproc_type	Variance normalization	Some voxels (e.g., in ventricles, along brain edges) will have very large signal changes, while others will have very small changes. Normalizing the variance across voxels (i.e., z-scoring the voxel timeseries) prior to analysis will increase the likelihood of retaining more interesting (i.e., non-artifact) signals in the PCA steps.
Single-subject PCA	numOfPC1	125	Close to the max of the ITC estimates. Should always be larger than the desired number of components. Maximally it is the number of time points (here, 180), which rotates & whitens subject data but does not reduce it.
Group PCA	numOfPC2	100	This is equal to the model order (number of components estimated in ICA). Here we use 100 as a starting point to assess stability.
ICA	algoType	InfoMax	The ICA algorithm (and various options) will affect the types of components you can decompose. InfoMax is very consistent from run to run and will do a good job of finding super-Gaussian components. See icatb_icaAlgorithm for more info.
Scaling	scaleType	Don't scale	A tricky one. Basically: how do you want to deal with amplitude information? For this demo I chose not to scale, but if I planned on comparing time course or spectral amplitudes between groups I would "normalize spatial maps using the maximum intensity value and multiply timecourses using the maximum intensity value".

one step at a time

You can run the entire analysis with a single line of code:

```
icatb_batch_file_run('gift_demo_batchfile');
```

However it can be informative and useful to run the analysis one step at a time.

```
% Read the parameters from the batch script:
param_file = icatb_setup_analysis('gift_demo_batchfile');
% Store the parameters properly:
sesInfo = icatb_runAnalysis(sesInfo, 2);
% take a look at 'sesInfo'.  Is everything correct?

% Data reduction:
% Runs both single-subject PCA and the group level PCA
sesInfo = icatb_runAnalysis(sesInfo, 3);

% calculate ICA:
sesInfo = icatb_runAnalysis(sesInfo, 4);
```

Understanding output

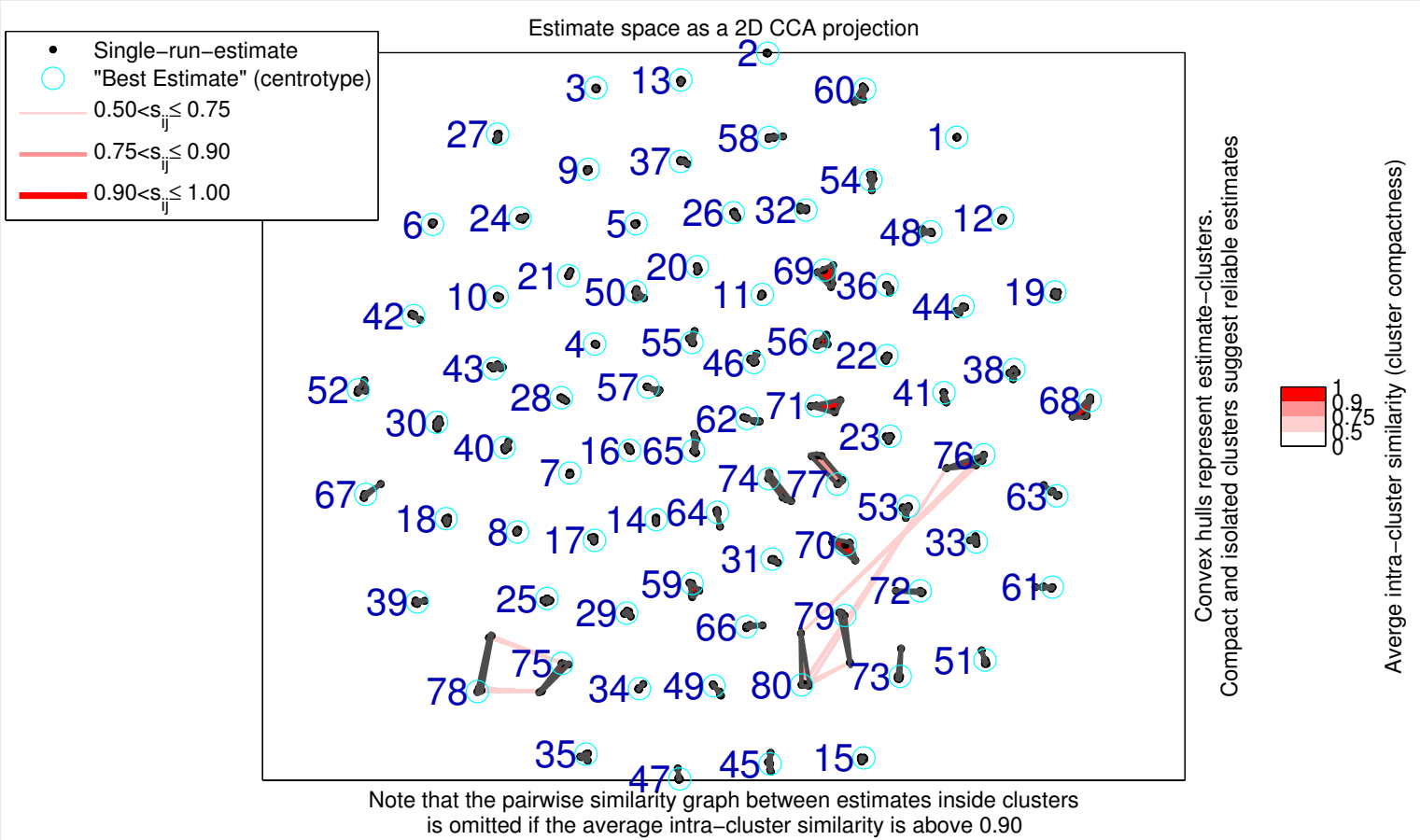
By now there should be files accumulating in your \ICA directory.

File name	Contents
rest_ica_parameter_info.mat	The parameter file. The Holy Grail. Holds all the answers.
rest_results.log	Log file that includes all screen output.
restMask.hdr (.img)	The mask. Should be identical to the mask we started out with.
restSubject.mat	Information about the files used for each subject.
rest_pca_r1-*.mat	The PCA-reduced data for each subject. Also contains the eigenvalues and reducing matrix.
rest_pca_r2-1.mat	The PCA-reduced data for the group data. Also contains the eigenvalues and reducing matrix.
rest_icasso_results.mat	ICASSO output assessing variability in the components from different ICA runs (more on this later).
rest_ica.mat	The ICA result. Contains the sources (as vectors), mixing matrix and unmixing matrix.
rest_agg__component_ica_.nii	The aggregate components (sources) in image space.
rest_agg__timecourses_ica.nii	The mixing matrix stored as a Nifti (not very useful).

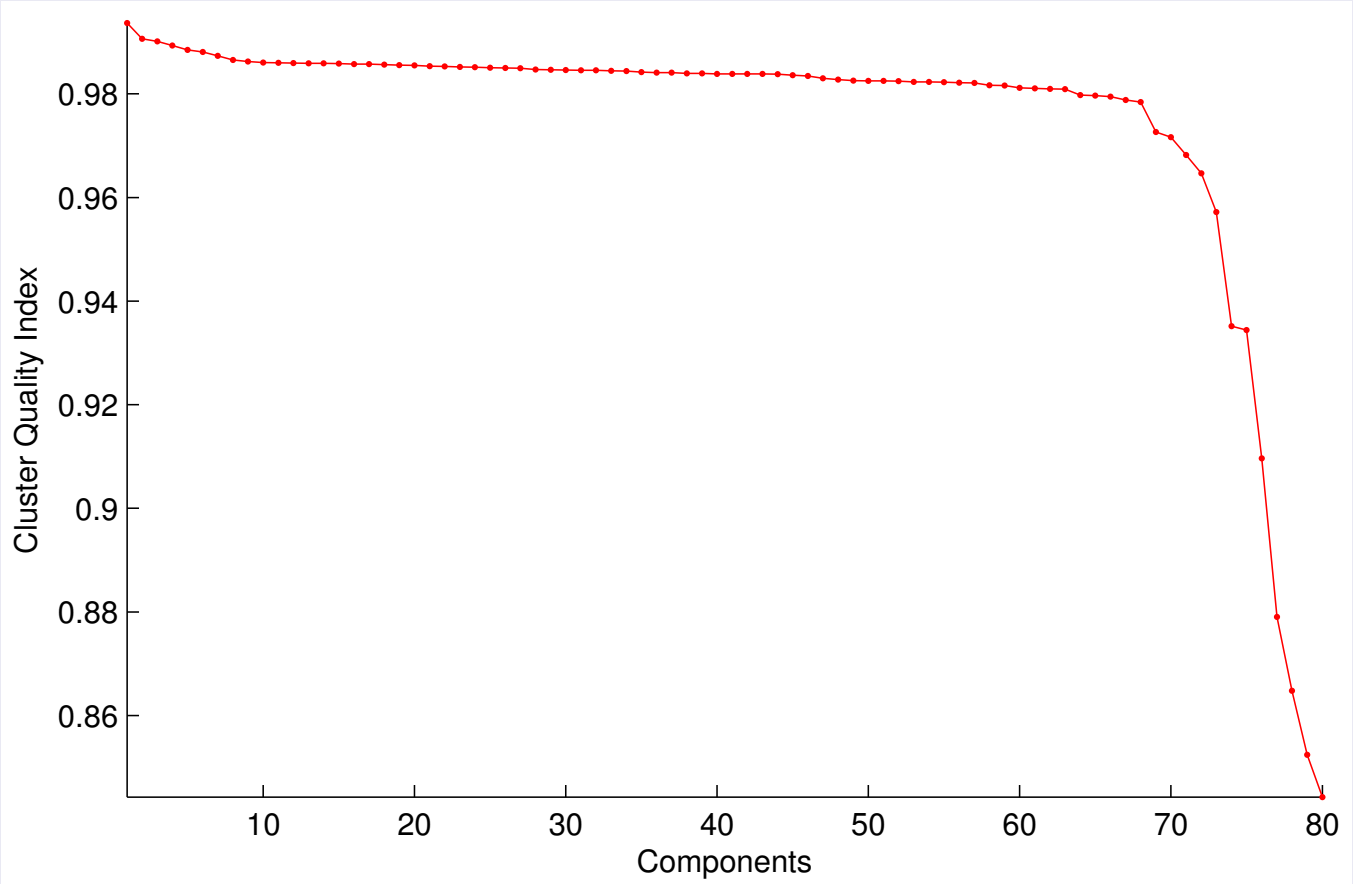
ICASSO

If you've run the analysis using ICASSO, you've run the ICA algorithm several times (here, 15) with different initial conditions or bootstrap resamples. The ICASSO toolbox compares the sources from the different runs and clusters them. If the solution is very consistent, you should get excellent separation of component clusters. If not, it may be difficult to separate components into clusters or these clusters may overlap. ICASSO provides some very nice visualizations to help us understand how reliable the decomposition is.

Component clustering



The knee in the curve



Model order revision

Based on ICASSO results and/or inspection of the components, we can revise our model order and re-run parts of the analyis. The lines below will allow you to start from the group PCA step. Alternatively you can edit the batch script and start from the beginning.

```
% Update the number of components to estimate
sesInfo.userInput.numOfPC2 = 70;

% Update some ICA options
nCind = find(strcmp(sesInfo.userInput.ICA_Options, 'ncomps'));
sesInfo.userInput.ICA_Options{nCind + 1} = sesInfo.userInput.numOfPC2;

% Store the parameters correctly
sesInfo = icatb_runAnalysis(sesInfo, 2);

% We don't need to rerun subject PCAs, just the group:
sesInfo.reductionStepsToRun = 2;
sesInfo = icatb_runAnalysis(sesInfo, 3);

% calculate ICA
sesInfo = icatb_runAnalysis(sesInfo, 4);
```

Final steps

Once you are satisfied with the model order, run the remaining steps.

```
% back-reconstruction
% estimation of components and timecourses for each subject
sesInfo = icatb_runAnalysis(sesInfo, 5);

% calibration (scale components)
sesInfo = icatb_runAnalysis(sesInfo, 6);

% group stats (mean and t-tests)
sesInfo = icatb_runAnalysis(sesInfo, 7);
```

Understanding output

Back-reconstruction, scaling, and group statistics will produce more output files.

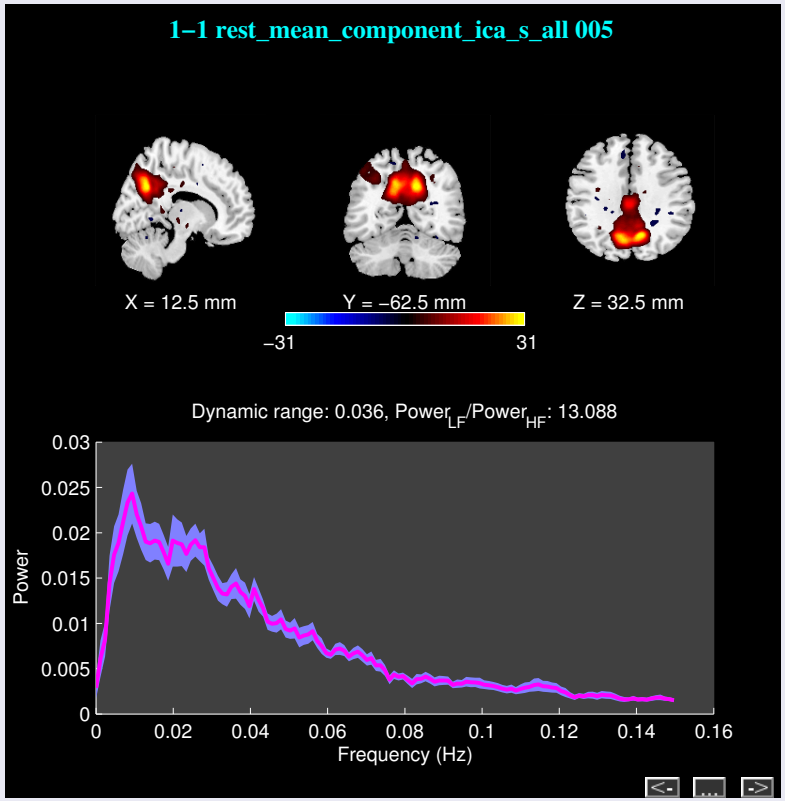
File name	Contents
rest_ica_br*.mat	The back-reconstructed components (as vectors) and timecourses for each subject.
rest_ica_c*-1.mat	The back-reconstructed and **calibrated** (scaled) components and timecourses for each subject.
rest_sub*_component_ica*.nii	The calibrated back-reconstructed components in image space for each subject.
rest_sub*_timecourses_ica*.nii	The calibrated back-reconstructed time courses for each subject.
rest_mean_timecourses_ica*.nii	The timecourses averaged across subjects (useful for task data). Separate files for each session and a mean of "all".
rest_mean_component_ica*.nii	The subject-specific components averaged across subjects in image space.
rest_std_timecourses_ica*.nii	The standard deviation of timecourses across subjects.
rest_std_component_ica*.nii	The standard deviation of components across subjects.
rest_tmap_timecourses_ica*.nii	The t-statistic of timecourses across subjects.
rest_tmap_component_ica*.nii	The t-statstic of components across subjects.

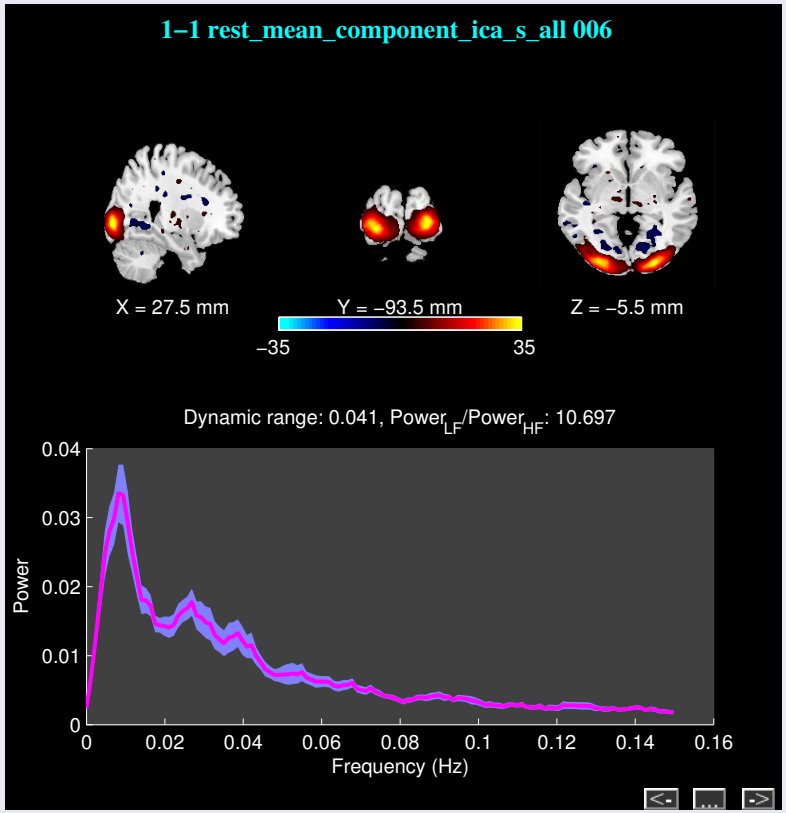
Overview

Visualization

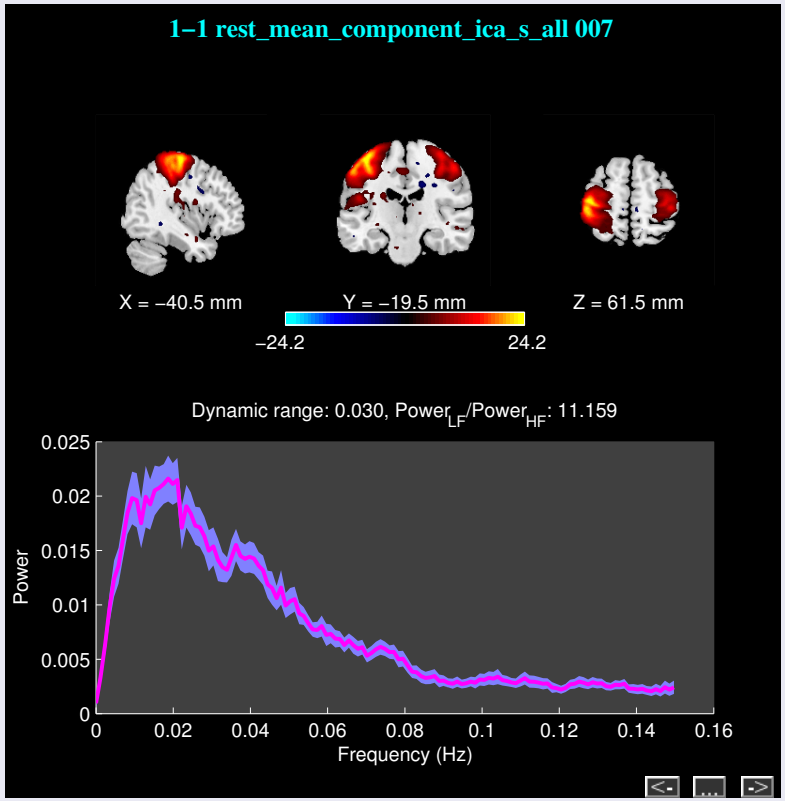
After all that hard work, it would be nice to actually see the components. Some very nice tools are available through GIFT's Display GUI. Alternatively, you can open the Nifti files in your preferred visualization program (e.g., MRIcroN, AFNI, etc.) that allows interactive use, overlays, and atlas integration.

Intrinsic Connectivity Networks

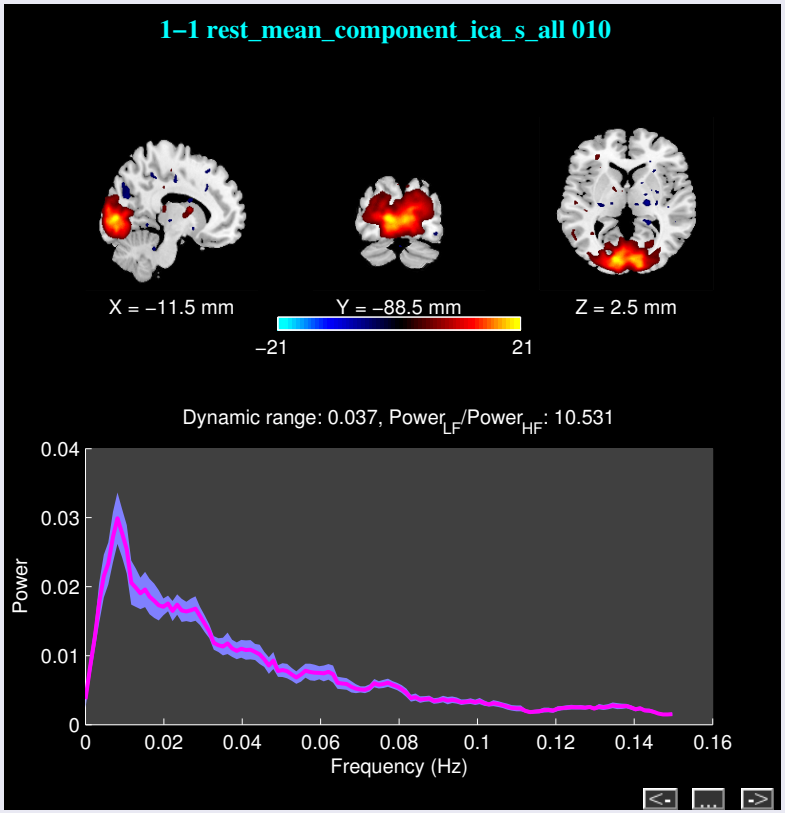


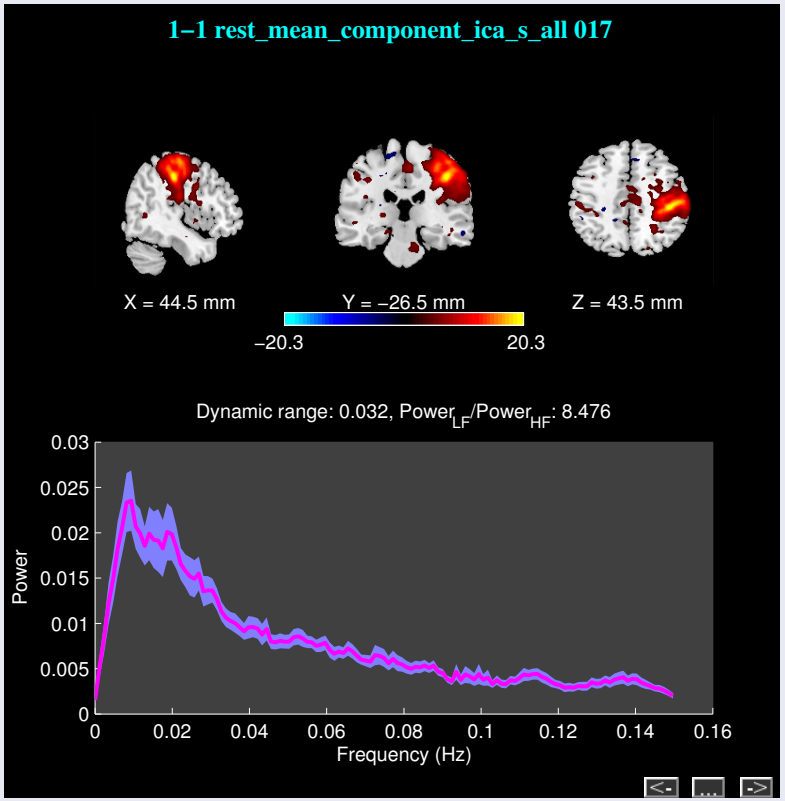


Intrinsic Connectivity Networks

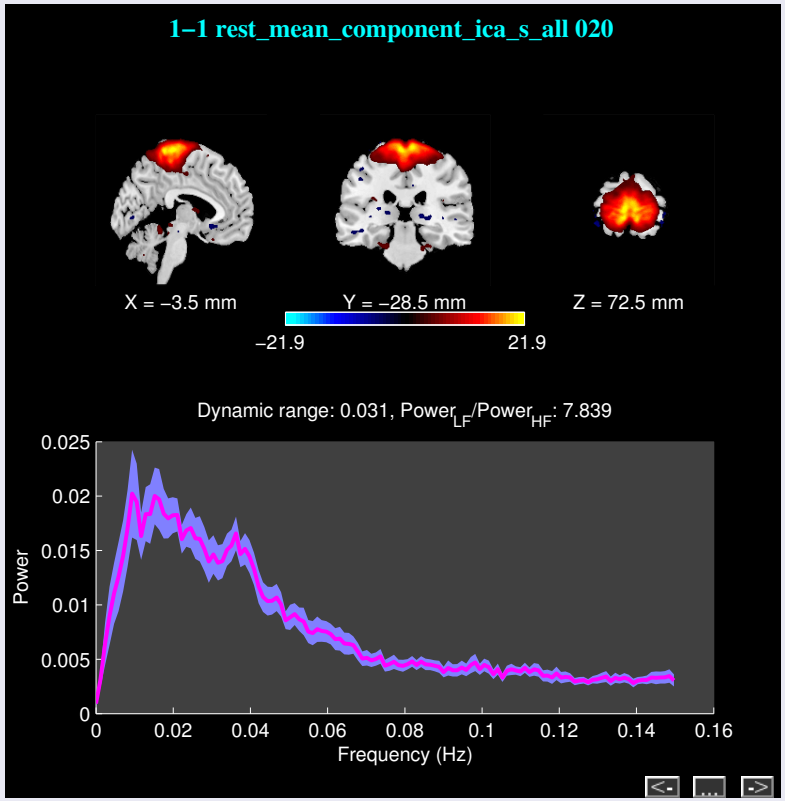


Intrinsic Connectivity Networks

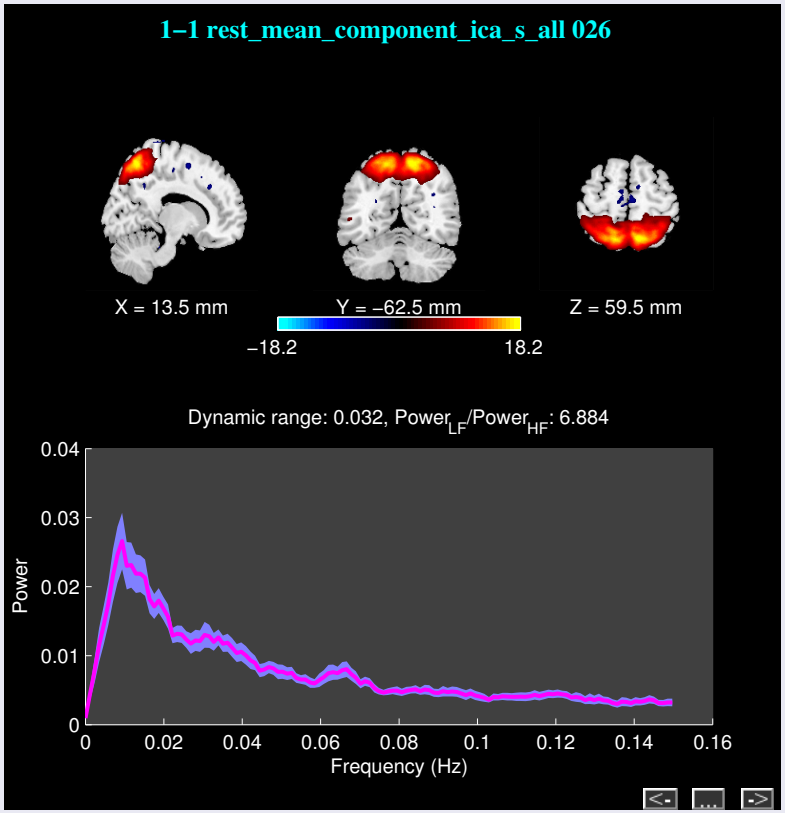


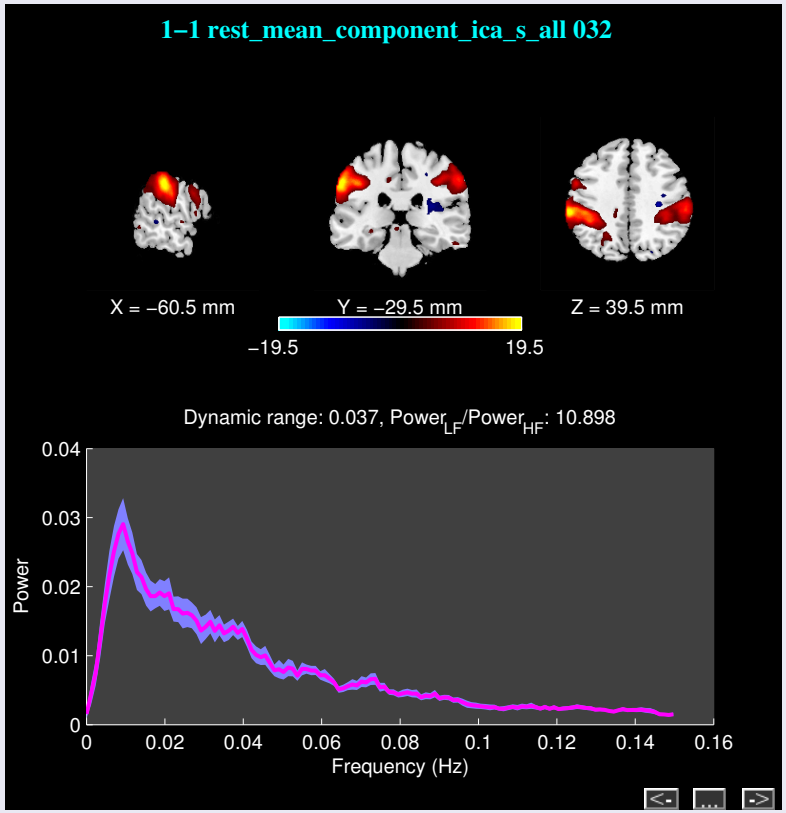


Intrinsic Connectivity Networks

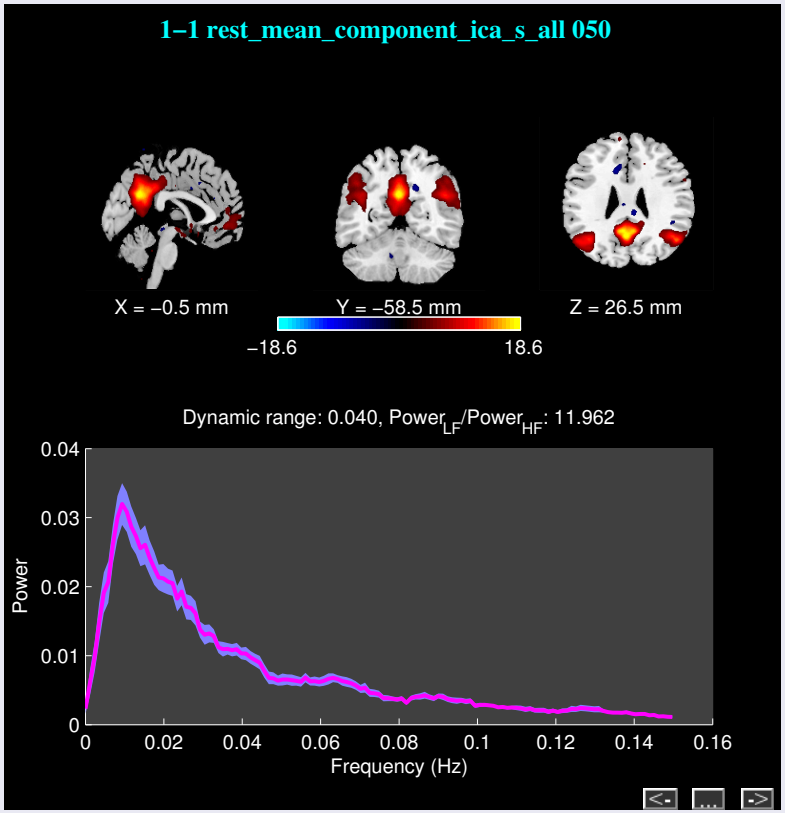


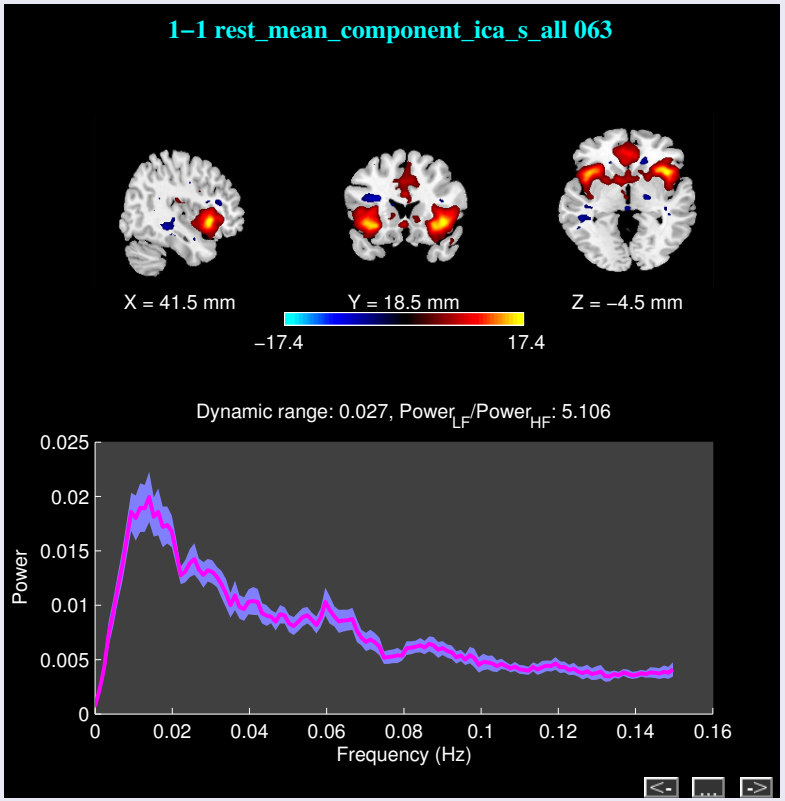
Intrinsic Connectivity Networks



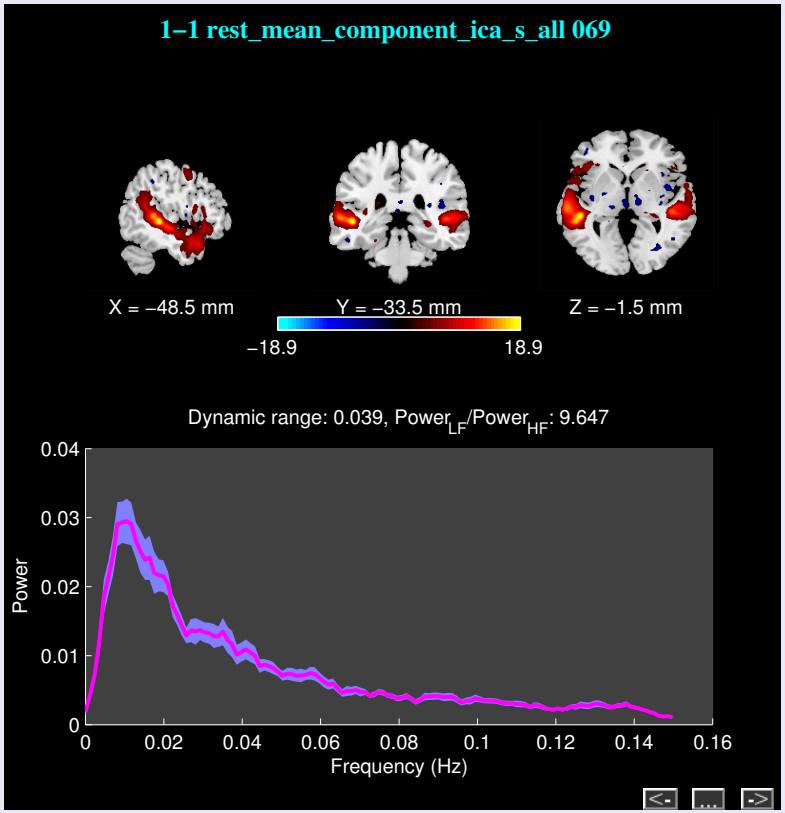


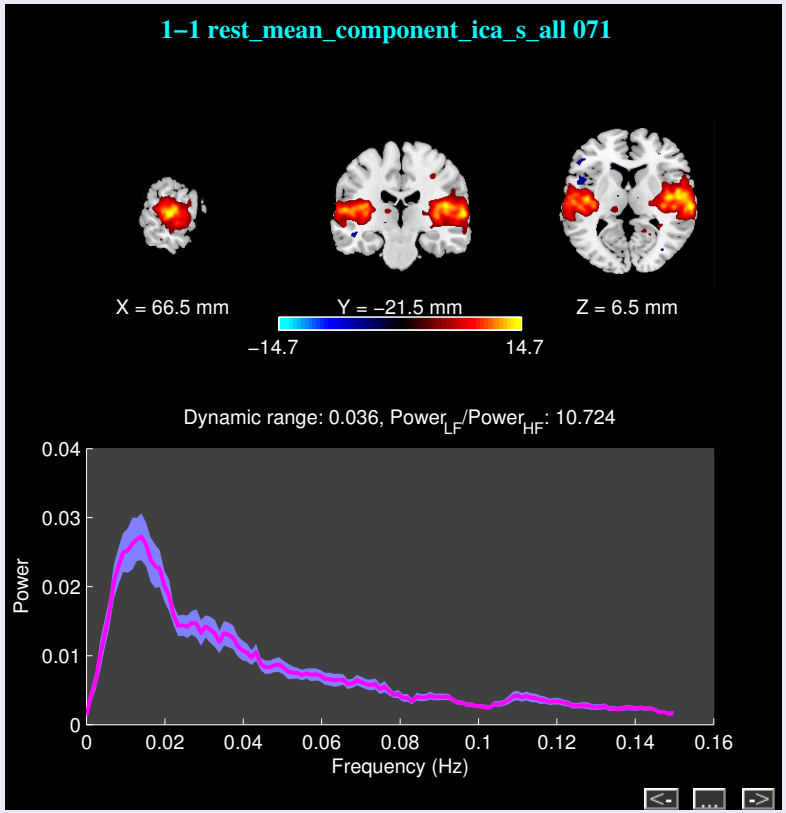
Intrinsic Connectivity Networks



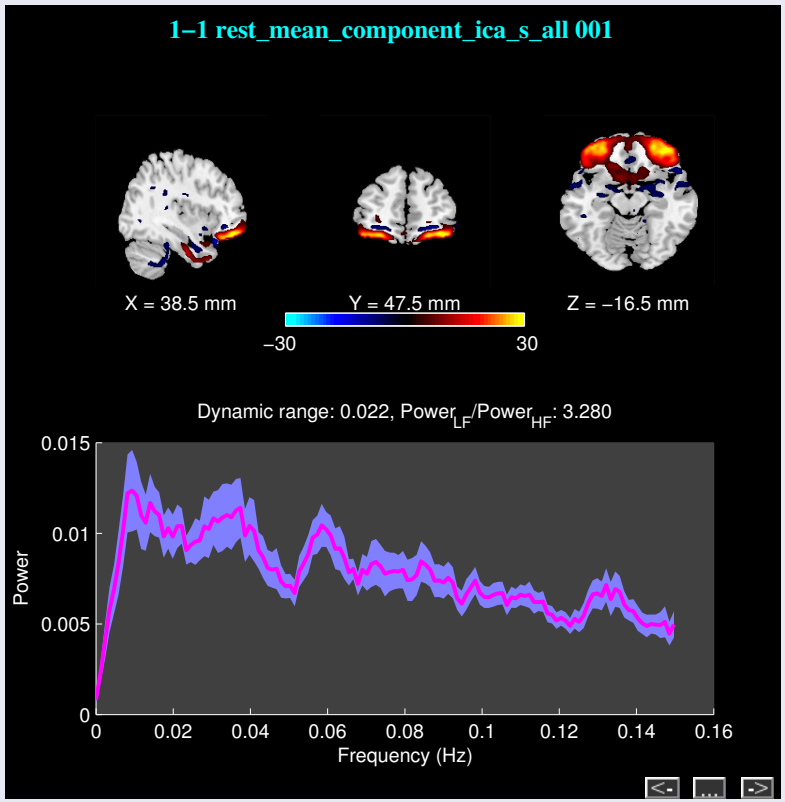


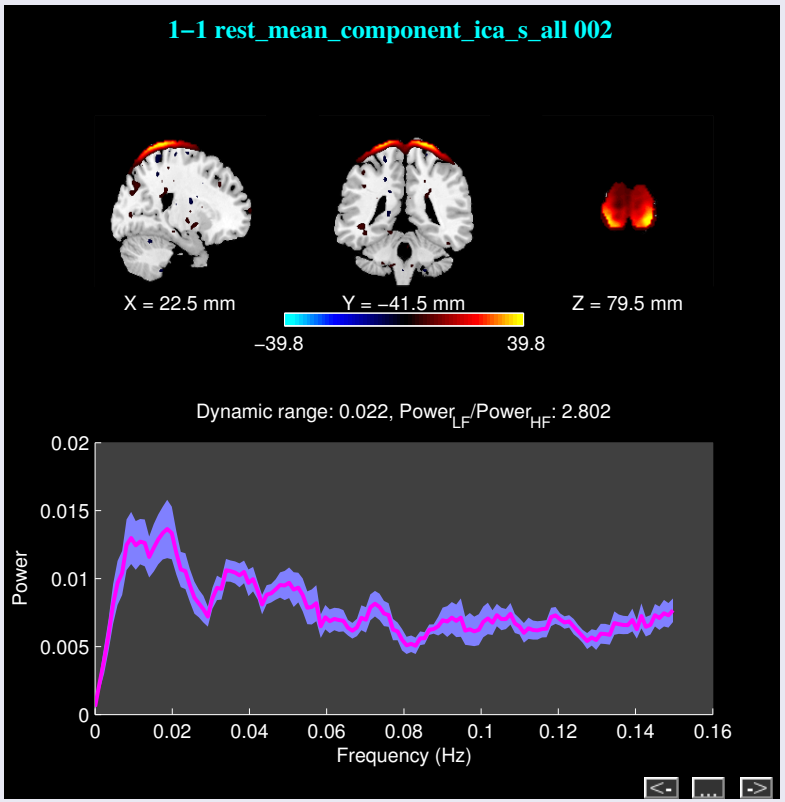
Intrinsic Connectivity Networks



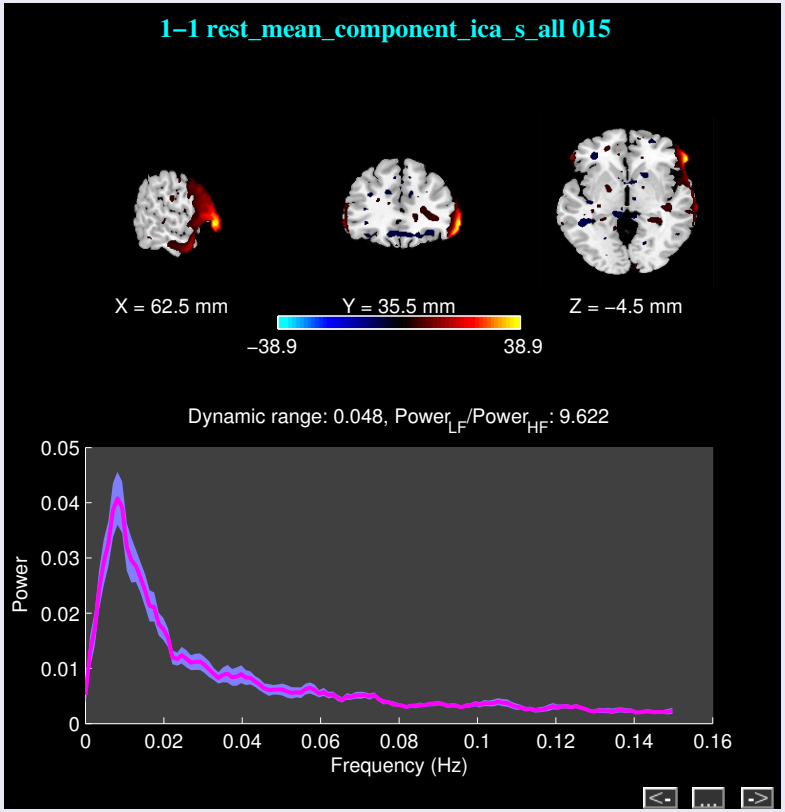


Motion

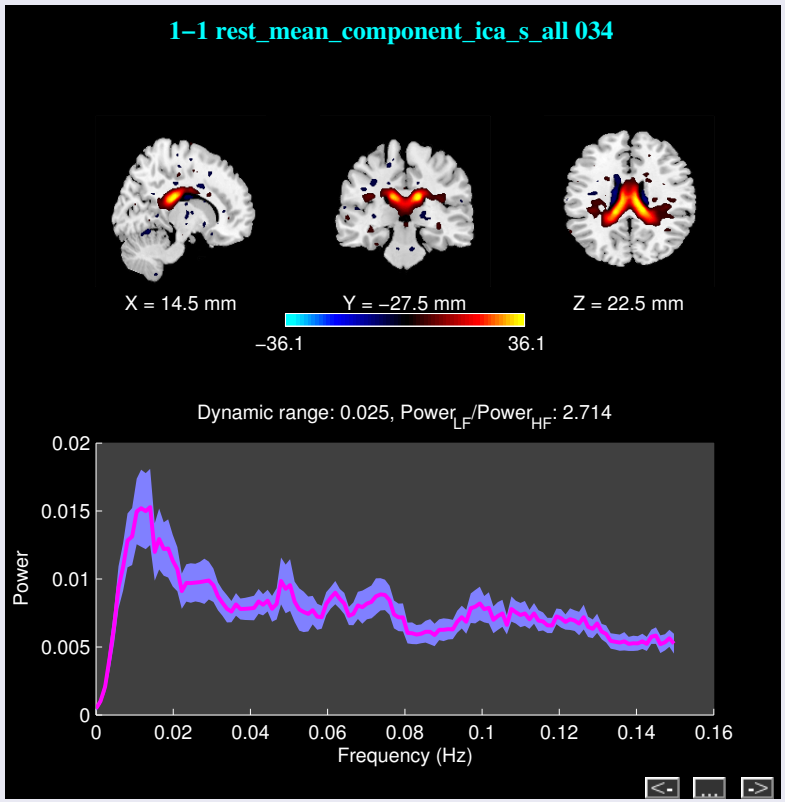

1–1 rest_mean_component_ica_s_all 001
X = 38.5 mm
Y = 47.5 mm
Z = −16.5 mm
−30
30
Dynamic range: 0.022, Power$_{LF}$/Power$_{HF}$: 3.280
Power
Frequency (Hz)

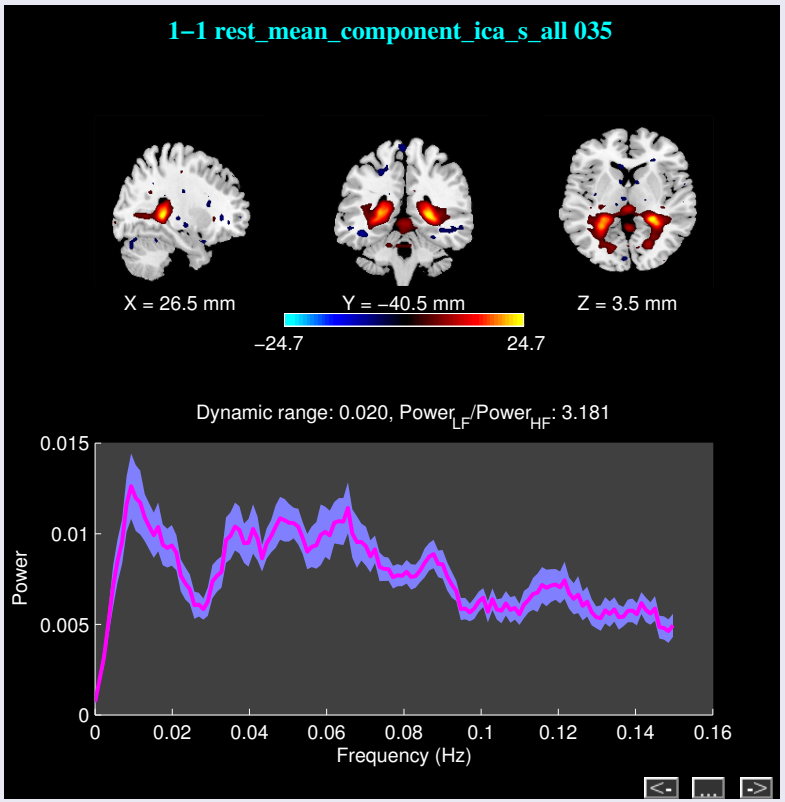

1–1 rest_mean_component_ica_s_all 002
X = 22.5 mm
Y = −41.5 mm
Z = 79.5 mm
−39.8
39.8
Dynamic range: 0.022, Power$_{LF}$/Power$_{HF}$: 2.802
Power
Frequency (Hz)

Motion

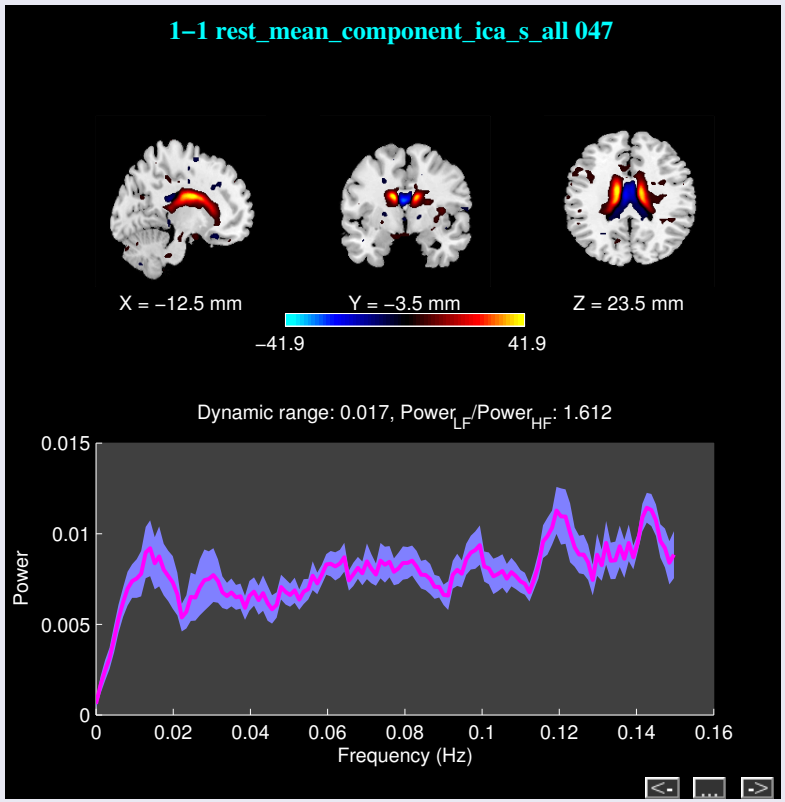

1–1 rest_mean_component_ica_s_all 015
X = 62.5 mm
Y = 35.5 mm
Z = −4.5 mm
−38.9
38.9
Dynamic range: 0.048, Power$_{LF}$/Power$_{HF}$: 9.622
Power
Frequency (Hz)
0.05
0.04
0.03
0.02
0.01
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
<-
...
->
1–1 rest_mean_component_ica_s_all 019
X = 57.5 mm
Y = 34.5 mm
Z = 5.5 mm
−15.6
15.6
Dynamic range: 0.020, Power$_{LF}$/Power$_{HF}$: 2.742
Power
Frequency (Hz)
0.02
0.015
0.01
0.005
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
<-
...
->

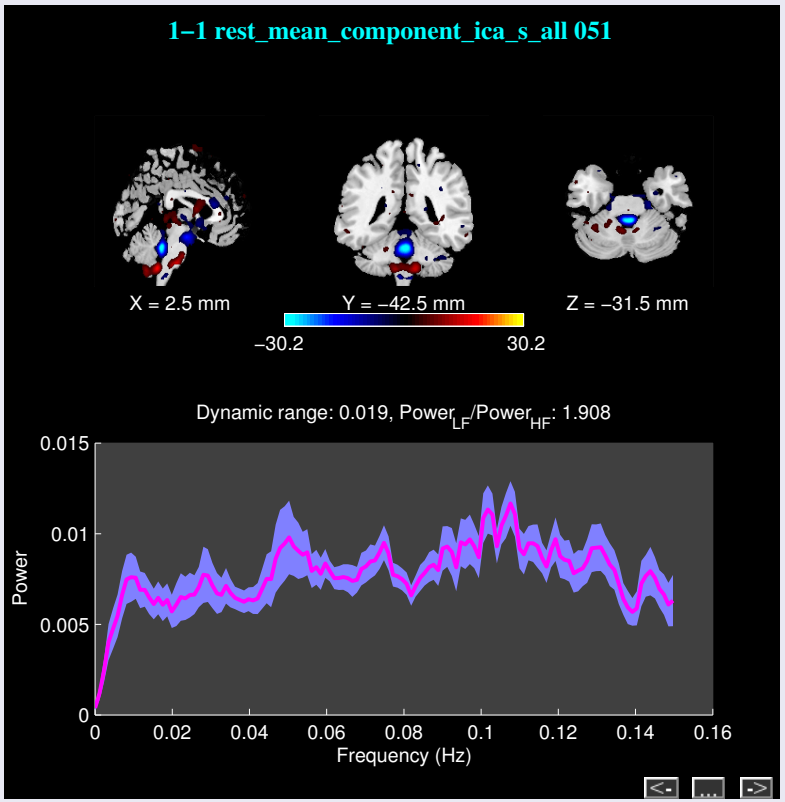
CSF


1–1 rest_mean_component_ica_s_all 034
X = 14.5 mm
Y = −27.5 mm
Z = 22.5 mm
−36.1
36.1
Dynamic range: 0.025, Power$_{LF}$/Power$_{HF}$: 2.714
Power
Frequency (Hz)

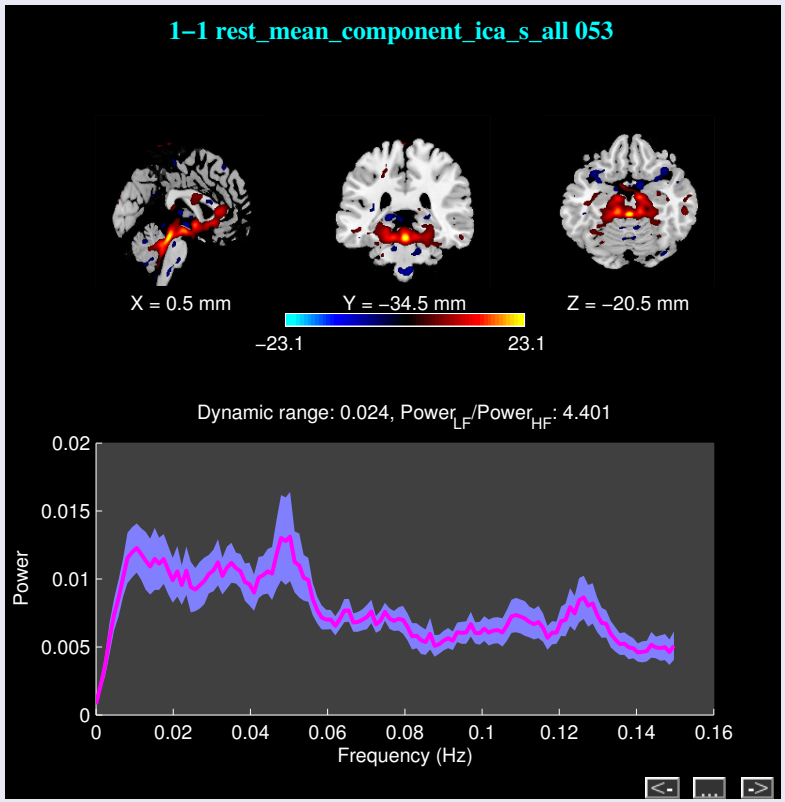

1–1 rest_mean_component_ica_s_all 035
X = 26.5 mm
Y = −40.5 mm
Z = 3.5 mm
−24.7
24.7
Dynamic range: 0.020, Power$_{LF}$/Power$_{HF}$: 3.181
Power
Frequency (Hz)

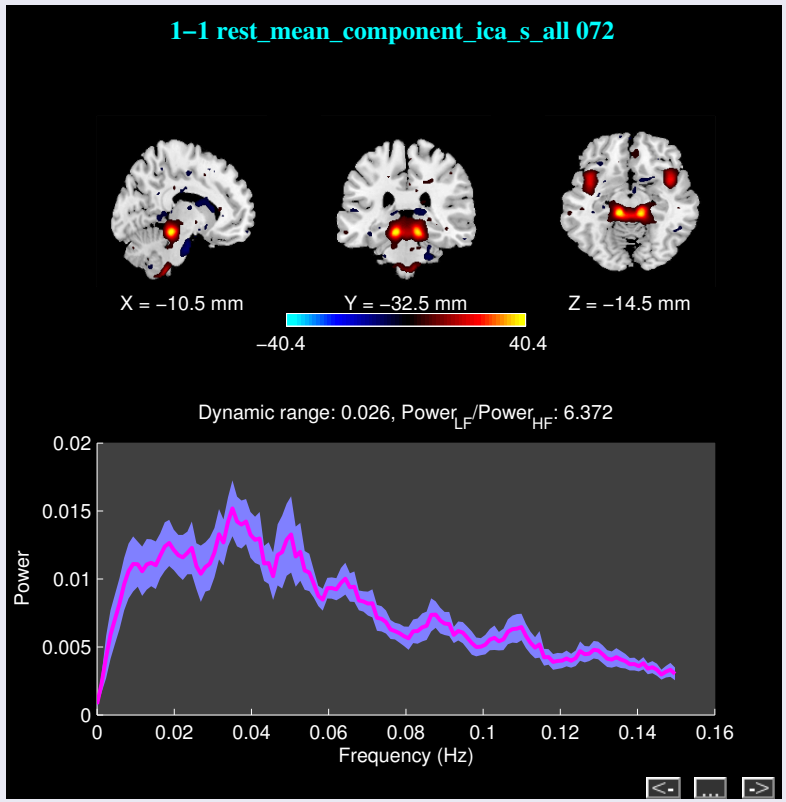
CSF


1–1 rest_mean_component_ica_s_all 047
X = −12.5 mm
Y = −3.5 mm
Z = 23.5 mm
−41.9
41.9
Dynamic range: 0.017, Power$_{LF}$/Power$_{HF}$: 1.612
Power
Frequency (Hz)

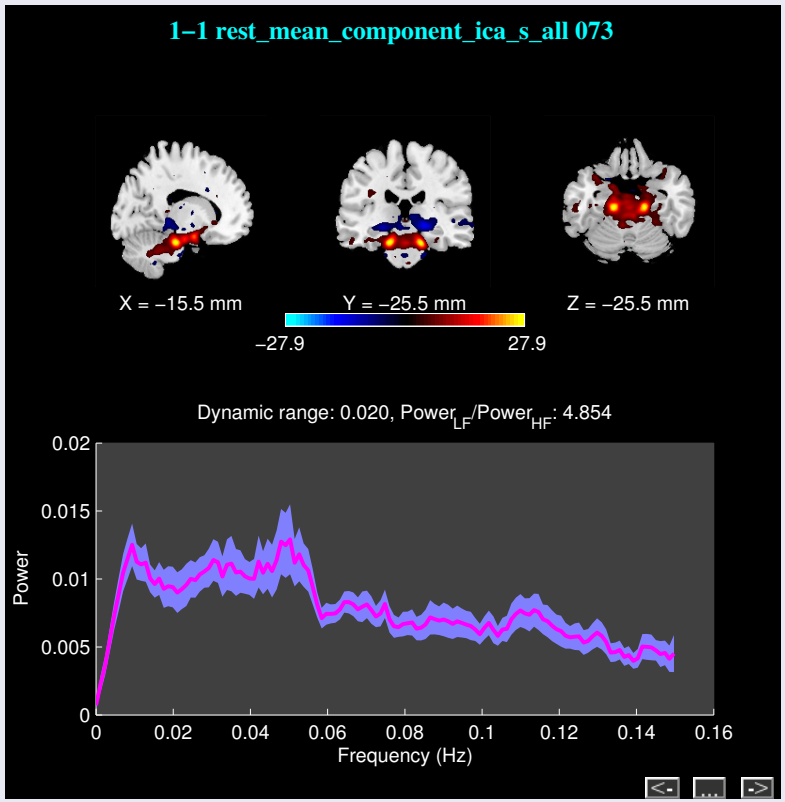

1–1 rest_mean_component_ica_s_all 051
X = 2.5 mm
Y = −42.5 mm
Z = −31.5 mm
−30.2
30.2
Dynamic range: 0.019, Power$_{LF}$/Power$_{HF}$: 1.908
Power
Frequency (Hz)

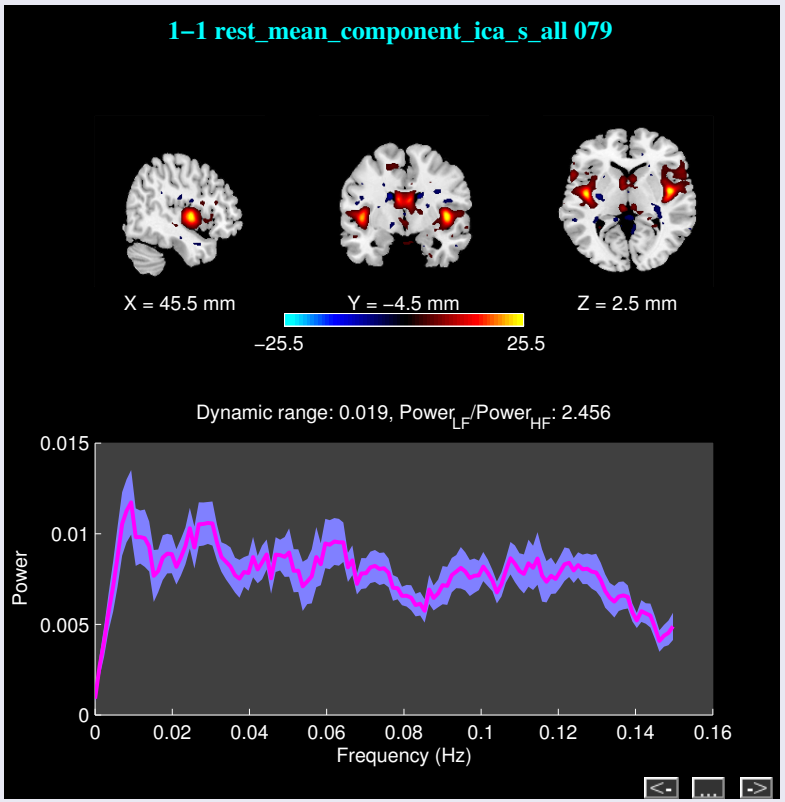
CSF


1–1 rest_mean_component_ica_s_all 053
X = 0.5 mm
Y = −34.5 mm
Z = −20.5 mm
−23.1
23.1
Dynamic range: 0.024, Power$_{LF}$/Power$_{HF}$: 4.401
Power
Frequency (Hz)

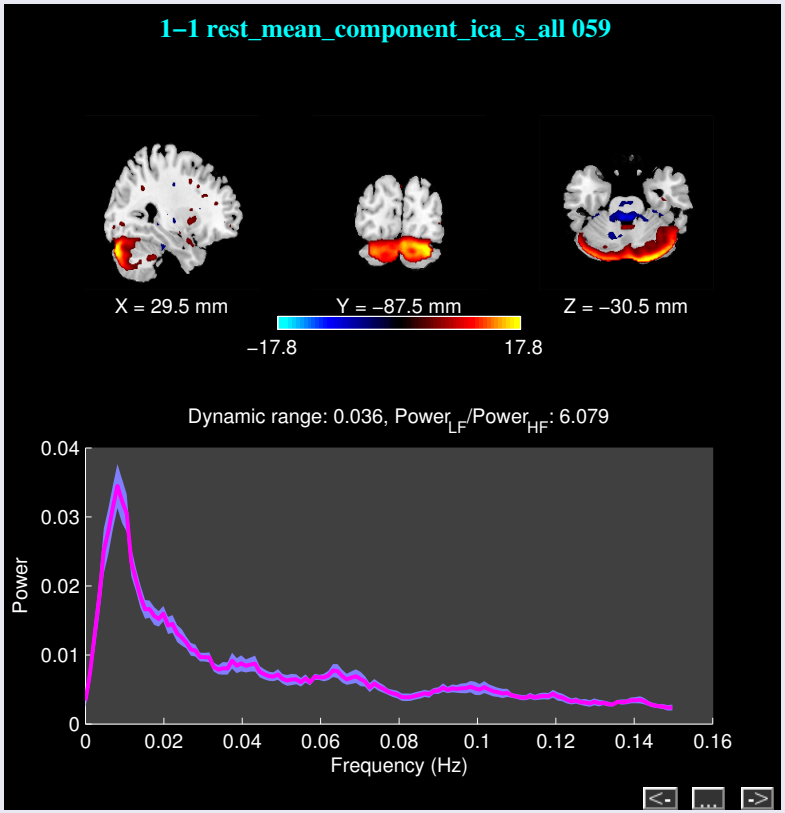

1–1 rest_mean_component_ica_s_all 072
X = −10.5 mm
Y = −32.5 mm
Z = −14.5 mm
−40.4
40.4
Dynamic range: 0.026, Power$_{LF}$/Power$_{HF}$: 6.372
Power
Frequency (Hz)

CSF


1–1 rest_mean_component_ica_s_all 073
X = −15.5 mm
Y = −25.5 mm
Z = −25.5 mm
−27.9
27.9
Dynamic range: 0.020, Power$_{LF}$/Power$_{HF}$: 4.854
Power
Frequency (Hz)


1–1 rest_mean_component_ica_s_all 079
X = 45.5 mm
Y = −4.5 mm
Z = 2.5 mm
−25.5
25.5
Dynamic range: 0.019, Power$_{LF}$/Power$_{HF}$: 2.456
Power
Frequency (Hz)

Sinuses


1–1 rest_mean_component_ica_s_all 059
X = 29.5 mm
Y = –87.5 mm
Z = –30.5 mm
–17.8
17.8
Dynamic range: 0.036, Power$_{LF}$/Power$_{HF}$: 6.079
Power
Frequency (Hz)

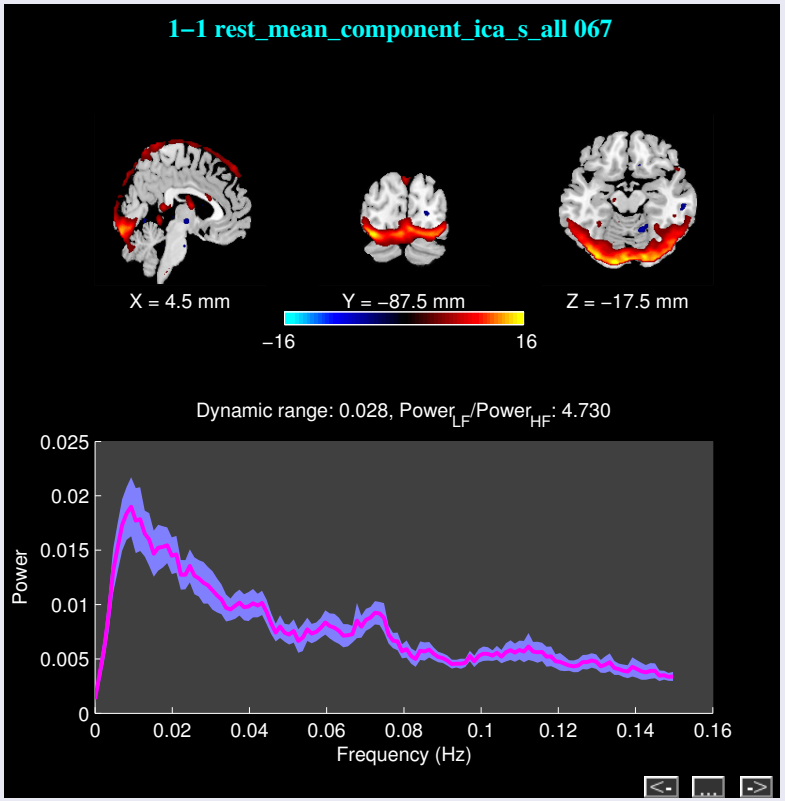

1–1 rest_mean_component_ica_s_all 067
X = 4.5 mm
Y = –87.5 mm
Z = –17.5 mm
–16
16
Dynamic range: 0.028, Power$_{LF}$/Power$_{HF}$: 4.730
Power
Frequency (Hz)

Image artifacts