Part 2 of Onboarding Challenge

Task: *When given the output/working directories from C-PAC's minimal-preproc configuration, show us the complete configuration used and its differences from the default pipeline*

- Complete configuration of CPAC's minimal-preproc pipeline can be found here: https://github.com/FCP-INDI/C-PAC/blob/main/CPAC/resources/configs/pipeline_config_preproc.yml
  The difference between preproc pipeline and default pipeline is that it doesn't calculate the outputs and data derivatives which include brain extraction, tissue segmentation, registration to template, slice-timing correction, motion estimation and correction, co-registration to structural, nuisance correction and filtering, and registration to template (resource).

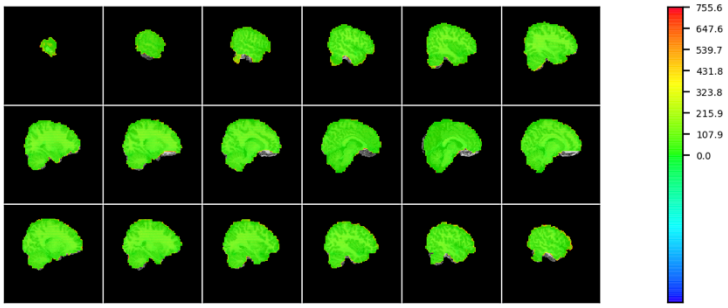
Task: *The a) longest and b) most memory-consuming tasks in the pipeline*

- Most memory consuming task: `.nuisance_regressors_Regressor-1_136.aCompCor_DetrendPC`. Memory run time: 4.548511504882812 GB. This can be found in `/log/pipeline_cpac_preproc/sub-0025429_ses-1/callback.log.resource_overusage.txt`

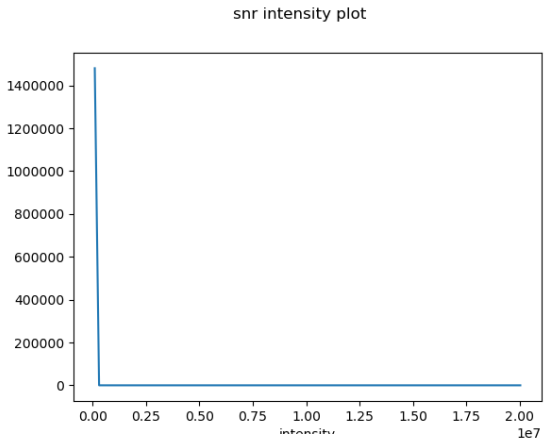
Task: *All quality control (QC) plots generated throughout executions*

- QC images:

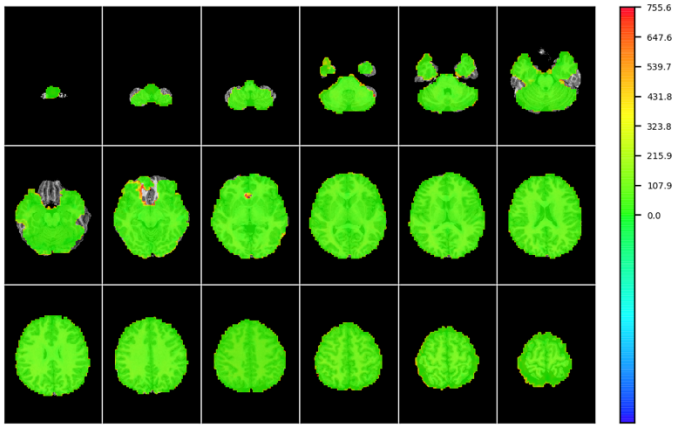
SNR sagittal



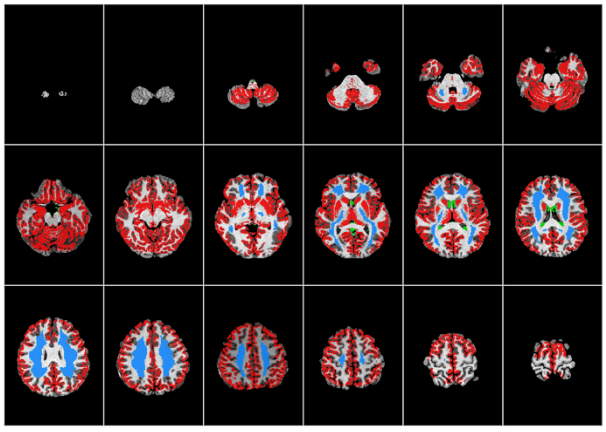
SNR histogram



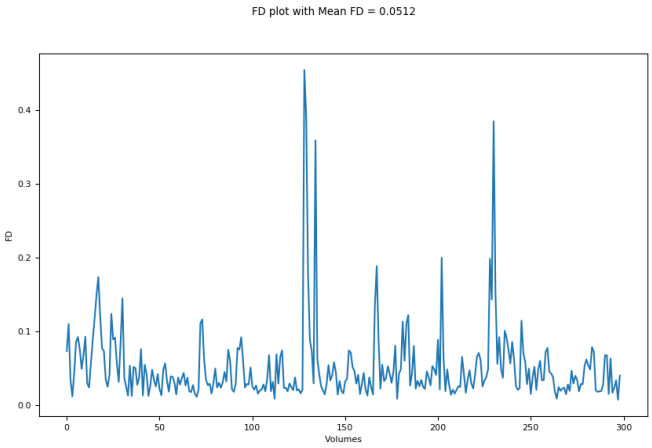
SNR axial



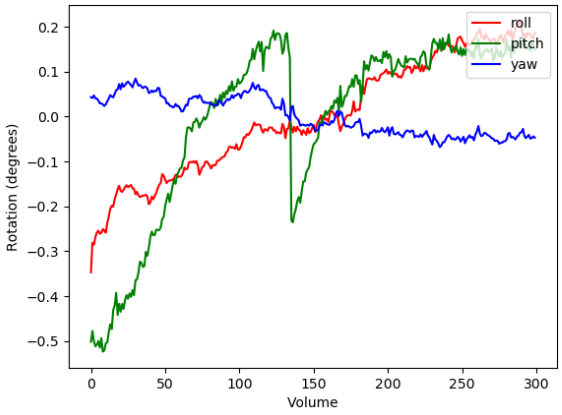
DSEG axial



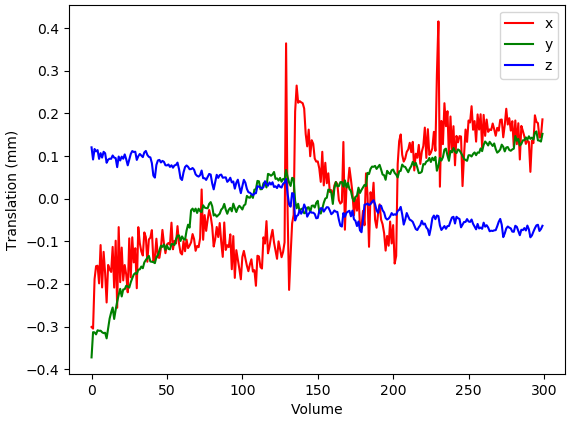
FD_J Plot



Movement parameters rotation Plot



Movement parameters translation Plot



Abbreviations:
SNR: signal-to-noise-ratio
DSEG: diffusion tensor image segmentation
FD_J: framewise displacement Jenkinson

Task: *When given voxelwise time series data, show us Voxelwise connectomes*

Used code here:
https://github.com/amygutierrez/CPAC_Onboarding/blob/main/conn_matrix.py

Will need to use the --mask tag and input the brain mask sub-0025429_ses-1_task-rest_run-1_space-template_desc-bold_mask.nii.gz as well as the functional time series file sub-0025429_ses-1_task-rest_run-1_space-template_desc-preproc-1_bold.nii.gz.

This code takes a while to run and may not execute.

Task: *Parcellated time series data with each of the CC200 and Schaefer 200 atlases*

Can use bash command `3dNetCorr`, which will save the files as `corr_matrix_CC200_000.niml.dset`. this file will have the parcellated values.

If you use this code, this line will also output the parcellated values. Just modify the code to output the variable.

Bash commands:

CC200

```
3dresample -master /cpac_templates/CC200.nii.gz -prefix
/home/agutierrez/cpac-preproc/C200_timeseries.nii.gz -input
/home/agutierrez/cpac-preproc/sub-0025429_ses-
1/Derivatives/func/sub-0025429_ses-1_task-rest_run-1_space-
template_desc-preproc-1_bold.nii.gz

3dNetCorr -prefix /home/agutierrez/cpac-
preproc/corr_matrix_CC200 -inset /home/agutierrez/cpac-
preproc/C200_timeseries.nii.gz -in_rois
/cpac_templates/CC200.nii.gz
```

Shafer 1000

```
3dresample -master /cpac_templates/Schaefer2018_space-
FSLMNI152_res-2mm_desc-1000Parcels17NetworksOrder.nii.gz -prefix
/home/agutierrez/cpac-preproc/Schaefer100_timeseries.nii.gz -
input /home/agutierrez/cpac-preproc/sub-0025429_ses-
1/Derivatives/func/sub-0025429_ses-1_task-rest_run-1_space-
template_desc-preproc-1_bold.nii.gz

3dNetCorr -prefix /home/agutierrez/cpac-
preproc/corr_matrix_Schaefer100 -inset /home/agutierrez/cpac-
preproc/Schaefer100_timeseries.nii.gz -in_rois
/cpac_templates/Schaefer2018_space-FSLMNI152_res-2mm_desc-
1000Parcels17NetworksOrder.nii.gz -push_thru_many_zero
```

3dresample was used to resample the time series to dimensions that match the atlas

Task: *Parcellated connectomes for the above atlases*

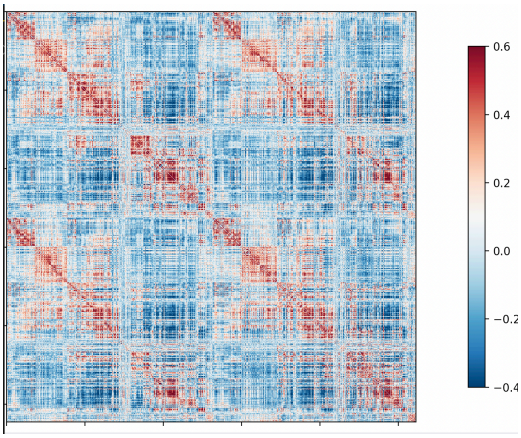
Use code: https://github.com/amygutierrez/CPAC_Onboarding/blob/main/conn_matrix.py

Will use --atlas tag and input the atlas file. The functional time series file will be the resampled time series created in the previous task. Will look something like this:

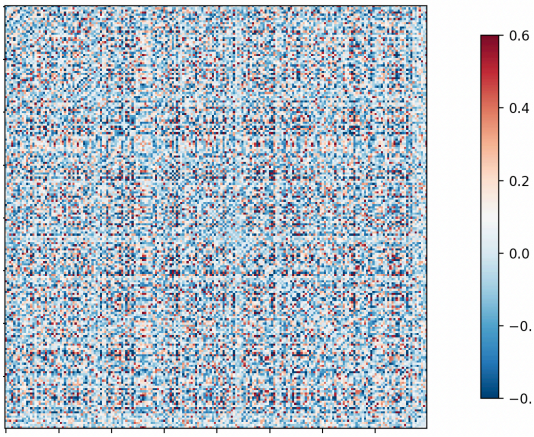
```
python3 ~/conn_matrix.py -ts ~/Schaefer100_timeseries.nii.gz -a
~/Schaefer2018_space-FSLMNI152_res-2mm_desc-
1000Parcels17NetworksOrder.nii.gz
```

Plots from code:

Parcellated connectivity matrix: Schaeffer 1000 (1064 X 1064)



Parcellated connectivity matrix: CC200 (200 X 200)



Task: *When given vertexwise time series and surface data, show us average values across time for each vertex, visualized on the surface*

To crate average time series, can use this command:

```
wb_command -cifti-reduce /data3/cnl/data/example_data/cpac-abcd-options/sub-NDARAD481FXF_ses-1/Derivatives/func/sub-NDARAD481FXF_1_task-task-rest_space-fsLR_den-32k_bold-dtseries.nii MEAN mean_ts.dscalar.nii
```

To view the average time series on the brain, can use this:

```
wb_view {surface files} mean_ts.dscalar.nii
```

Task*: Parcellated average time series data using the Schaefer 200 parcellation*

```
wb_command -cifti-parcellate /data3/cnl/data/example_data/cpac-abcd-options/sub-NDARAD481FXF_ses-1/Derivatives/func/sub-NDARAD481FXF_1_task-task-rest_space-fsLR_den-32k_bold-dtseries.nii /post_freesurfer_189/MNINonLinear/fsaverage_LR32k/1019436_1.aparc.32k_fs_LR.dlabel.nii COLUMN parcellation.ptseries.nii

wb_command -cifti-reduce parcellation.ptseries.nii MEAN mean_parcellated_ts.dscalar.nii
```

Task: *When given a BIDS input dataset, show us the corresponding output data for the default configuration of C-PAC v1.8.4*

```
git checkout v1.8.4

docker run --security-opt=apparmor:unconfined -v
/home/agutierrez/Documents/C-PAC:/code -v
/home/agutierrez/Documents/C-
PAC/dev/docker_data/run.py:/code/run.py -v
/home/agutierrez/Documents/C-PAC/dev/docker_data/run-with-
freesurfer.sh:/code/run-with-freesurfer.sh -v
/home/agutierrez/Documents/C-
PAC/dev/docker_data/default_pipeline.yml:/cpac_resources/default
_pipeline.yml -v /home/agutierrez/DATA/HBN/HNU_1:/data -v
/home/agutierrez/Documents/cpac_v1.8.4:/output fcpindi/c-
pac:nightly /data /output participant --save_working_dir --
skip_bids_validator --n_cpus 6 --mem_gb 25 --participant_label
sub-0025429
```

Outputs are in: `/home/agutierrez/cpac_v1.8.4`

Task: *The corresponding output data for a pipeline that is similar to the default, but that does not perform component correction (CompCor) in the nuisance regressor that includes global signal regression (GSR)*

YAML file used as the pipeline config can be found here:
https://github.com/amygutierrez/CPAC_Onboarding/blob/main/no_CompCorr.yml

```
docker run --security-opt=apparmor:unconfined -v
/home/agutierrez/Documents/C-PAC:/code -v
/home/agutierrez/Documents/C-
PAC/dev/docker_data/run.py:/code/run.py -v
/home/agutierrez/Documents/C-PAC/dev/docker_data/run-with-
freesurfer.sh:/code/run-with-freesurfer.sh -v
/home/agutierrez/Documents/C-
PAC/dev/docker_data/default_pipeline.yml:/cpac_resources/default
_pipeline.yml -v /home/agutierrez/DATA/HBN/HNU_1:/data -v
/home/agutierrez/Documents/C-
PAC/edited_pipelines/no_CompCorr.yml:/no_CompCorr.yml -v
/home/agutierrez/Documents/cpac_v1.8.4_noCompCorr:/output
fcpindi/c-pac:nightly /data /output participant --
save_working_dir --skip_bids_validator --n_cpus 6 --mem_gb 25 --
participant_label sub-0025429 --pipeline /no_CompCorr.yml
```