**NIMB = NeuroImaging My Brain**

Pipeline has 2 stages:

1. Processing stage
2. Check Data stage
3. Statistical Analysis Stage

In the PROCESSING STAGE:

* Performs processing of the provided MRI files
* If multiple time points are provided, a longitudinal analysis will be performed
* extracts stats for individual participants

CHECK DATA STAGE:

* Checks the processed data/ results and potential outlining values in the individual stats, compared to other participants and previous experience (AI, ML)

STATISTICAL STAGE:

After processing is done, if the file with groups is provided, App performs statistical analysis.

If FreeSurfer is installed on the local computer, the FreeSurfer GLM analysis will be done.

Pipeline starts with the ***exe.pyw*** file (*python exe.pyw*).

**GUI-main**

Button *project name* (default = *not set*) (each project has a different name)

Button *do processing*

temporary-Button *copy subjects to cluster*

Delimiting line

Button *cluster name* (default = *not set*)

2 change: (temporarily, until everything works)

Add button: retrieving data from the remote/cluster

Add button: extract stats

**GUI-project name** (after pressing the button *project name* for each project)

button MRI folder:

* Version 1: receives a .json file with the corresponding addresses of the MRI files or the path to the folder with the MRI data (.dcm or .nii data)
* Version 2: receives the folder that has a BIDS structure
* Version n3: receives the folder with the csv files that pre-classified the MRI folder or baseline: Main\_File = ENTRY: (address of the main file with data (may have the with extension “\_tp1”)); tp1: Longitudinal-file\_tp1 = ENTRY: (address of the file with data tp 1); Longitudinal-file\_tp2 = address of the file with data tp 2

button Results folder: path to the folder that will have the results

button GLM folder: folder used to send the statistical results after the GLM analysis. GLM analysis can be performed ONLY on a virtual Linus machine.

button File with clinical data and groups: path to the excel file that has the names of the subjects and the groups. *One excel/ csv file is provided that has the id’s and groups (‘group’)*

commented: Button *Change data* – change data for the existing project, manually + update sqlite database

Button *Add cluster* – add a new project to the sqlite database

**GUI-cluster name** (after pressing the button *cluster name*)

Username: username used to access the cluster

Cluster address: address to access the cluster (*e.g., beluga.computecanada.ca*)

Home folder: is the folder on the cluster that is used to install the NIMB scripts (in the folder …/a), install the anaconda.sh and the FreeSurfer software (if needed).

Scratch/Working folder: the folder on the cluster that will be used to perform the freesurfer processing (*e.g., /scratch)*

App\_DIR:folder where the NIMB scripts on the remote computer will be copied, by default it will be located in /home/USER/a or /home/USER/nimb, but for beluga this folder must be /home/USER/projects/def-SUPERVISOR/nimb

Subjects\_raw\_DIR: is the folder on the cluster that is used to receive the raw .dcm or .nii or .nii.gz files that will further be used for processing. By default it is located in the /home/USER/subjects.

Processed\_SUBJECTS\_DIR: is the folder that will store the processed subjects, after finishing the FreeSurfer pipeline. All subjects send to this folder are either processed or have an error in the processing and were removed from the processing stream. By default it is located in /scratch/USER/processed (“*fs-processed” at the moment*)

Supervisor CCRI: is the ComputeCanada CCRI number of the supervisor (mandatory for Compute Canada accounts)

Button *Change data* – change data for the existing cluster+ update sqlite database

Button *Add cluster* – add a new cluster to the sqlite database

**GUI- cluster name-Change data – DATA ENTRY**

**GUI- cluster name-Add cluster**

Cluster name: name of the cluster (at the moment defaults are: *beluga, mammouth, niagara, helios)*

Username: username used to access the cluster (*e.g., user*)

Cluster address: address to access the cluster (*e.g., beluga.computecanada.ca*)

Cluster SCRATCH folder: the folder on the cluster that will be used to perform the freesurfer processing (*e.g., /scratch)*

Cluster HOME folder: is the folder on the cluster that is used to install the NIMB scripts (in the folder …/a), install the anaconda.sh and the FreeSurfer software (if needed). For beluga cluster the address is */home/USER/project/def-USER*

Password: user enters the password to access the cluster. Password is displayed in stars.

Supervisor CCRI: is the ComputeCanada CCRI number of the supervisor (mandatory for Compute Canada accounts) and a script must be writte to check if the CCRI is mandatory, and if yes – the USER must be asked to provide it.

Button *Submit* – triggers the module and saves the data to the sqlite database

Button *Delete* – triggers the module and removes the cluster data from the sqlite database

Save data in a **SQL** file stored on local and encripted

**PROCESSING STAGE:**

**BUTTON-do processing**

(after pressing the button *do processing* on GUI-main)

CHECK\_CLUSTER\_SETUP

If everything OK:

MRI\_DIR\_LOOP:

If an MRI folder is provided:

CLASSIFY\_FOLDERS

JSON\_LOOP

JSON\_LOOP:

if .json file is provided:

read file, for each \_id in the file:

if \_id is NOT present in the *Results folder*:

CHECK\_REMOTE

COPY\_to\_REMOTE

START\_REMOTE\_PROCESSING

GET\_PROCESSED

else:

CHECKING DATA STAGE:

If True: STATISTICAL ANALYSIS STAGE

**MODULES/ SCRIPTS on LOCAL**

Class **CHECK\_REMOTE**:

check how much space is available on the remote for in the Subjects\_raw\_DIRfolder.

Use the available space to compute how many subjects can be copied to the remote.

Return: the nr of subjects that can be copied

Class **COPY\_to\_REMOTE**:

If CHECK\_REMOTE is >0:

-> start copying the MRI data for each \_id from the .json file to the remote Subjects\_raw\_DIR

-> copy folder with .dcm for the \_id to *Sujects\_raw\_DIR* using the structure folder: \_id+\_+ session+\_+MRI-type (MRI\_type = \_t1 or \_flair) (e.g.: pls\_hc000\_ses-1\_t1/). folder contains all .dcm files or the .nii.gz file

-> create another new\_subjects.json file that will have same BIDS structure: \_id -> session -> anat -> MRI-type -> path to the folder on the remote with the .dcm or .nii files

-> for each \_id that is copied, update the new\_subjects.json file

-> when all \_id are copied based on the available space, send the new\_subjects.json file to remote folder App\_DIR

-> start the analysis on the rmeote with START\_REMOTE\_PROCESSING

Class **START\_REMOTE\_PROCESSING**:

Makes the connection with the remote

If remote has a scheduler (slurm, pbs):

Sends the App\_DIR/run.pbs to the scheduler using the corresponding “run” command (sbatch or other)

If remote does NOT has a scheduler:

Check if tmux or screen is installed

If True:

Send the running command to a new tmux or screen

Class **GET\_PROCESSED:** (retrieves the processed folder from the remote)

Check if there are folders in the remote folder:

If True: Copy from the Remote processed\_SUBJECTS\_DIR

For each folder:

If the ‘long’ name is present in the folder:

Copy the folder to:

PATH\_to\_save results/processed/freesurfer\_longitudinal

Check that the copy was performed correctly.

If copy is correct:

Remove the copied folder from the remote folder

Else:

Remove copied folder from the local, restart copy. Try 3 times.

If there are no more folders with the “long” name in the remote folder:

If there are folder with the name ‘base’ in their name:

Copy those folders to:

PATH\_local/processed/freesurfer\_longitudinal\_base

Check if copy is correct as in the previous step.

If there are no more folders with the ‘base’ name in their names:

Copy each remaining folder to:

PATH\_local/processed/freesurfer

verify that all files were copied

If True: delete the cluster files/folder (in the ‘processed’ and the ‘subjects’ folders)

Start EXTRACT\_STATS

Start STATISTICAL\_ANALYSIS

Class **CLASSIFY\_FOLDERS**: (Data Classifier)

Files are being classified for each participant as presented in the examples in the example files in the “a/classification/Data”

Script a/classification/DataClassifier/DataClassifier

*script extract id’s*

*based on the id - searches in the ‘address\_of\_the\_DCM\_folder’ folders for the specified id*

*classifies time points based on the criteria defined by the user.*

*Criteria - will be provided in the ‘entry data’; e.g., user will be asked to type the parameter that differentiates different time points (e.g, \_tp), script will search for ‘\_tp’ in folder’s name and classify accordingly.*

**Data Classification excel/csv based**

pandas, access the Main\_File file (excel, csv) usually time point 1 (i.e., “tp1”)

extract column ‘id’

for each main\_id:

extract name of t1/flair/t2 for MAIN\_FILE

for each file (t1/ flair/ t2):

PATH\_FILE = check\_if\_file\_is\_present():

if PATH\_FILE not null:

sent\_to\_dictionary(PATH\_FILE)

if there are additional tp files (based on the Longitudinal-TP-names value):

if main\_id in list of id’s for the tp file:

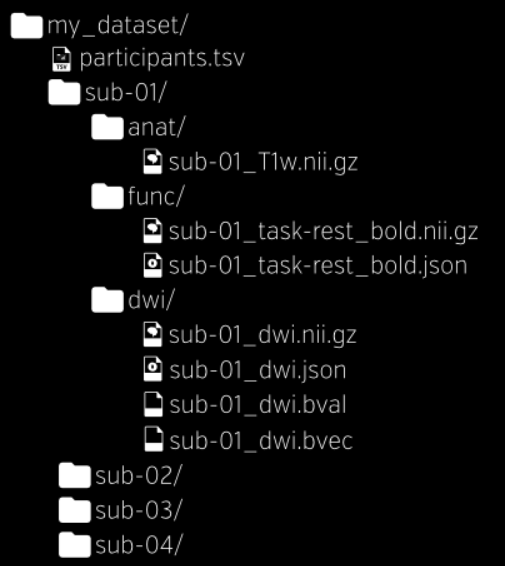
extract the name of t1/flair/t2 DIR or NII.FILE for the that specific tp

for each file (t1/ flair/ t2):

if check\_if\_file\_is\_present():

sent\_to\_dictionary()

def check\_if\_file\_is\_present(DIR, PATH):



list\_dcm\_files = list()

if path.isdir(PATH+DIR):

if False:

print(“ ERROR no folder”, folder name)

pass

if ‘.dcm’ in FILE\_NAME in listdir(PATH+DIR):

if False:

print(“ ERROR no dcm files in the folder”)

pass

if len(listdir(PATH+DIR))>15:

PATH + ‘/’ DIR+’/’+sorted(listdir(PATH+DIR)[0])

elif path.isfile(PATH+DIR):

if ‘.nii’ in FILE\_NAME:

PATH+’/’+FILE\_NAME\_if\_nii

return PATH\_FILE

def sent\_to\_dictionary(PATH\_FILE):

note: structure of the **subjects** folder

send to dict() “classification\_dict”:

{‘subj\_id1’:

{*tp1*:

{*‘anat’*:

{*‘t1’*:

[PATH\_FILE]

*‘flair’*:

[file flair for tp0 for id]

*‘t2’*:

[file t2 for tp0 for id]

}

*‘tp2’*:

{*‘anat’*:

{*‘t1’*:

[PATH\_FILE]

*‘flair’*:

[file flair for tp0 for id]

*‘t2’*:

[file t2 for tp0 for id]

}

-”-

}

}

‘subj\_id2’:

}

id, files\_types for each tp (e.g., tp1\_flair, tp1\_t1, tp2\_t1, etc)

open classification\_dict:

for each id:

for ‘anat’:

if there >1 tp:

go longitudinal analysis:

else:

go cross-sectional analysis

if cross-sectional:

create batch file for cross-sectional

if longitudinal:

create batch file for longitudinal

**Data Classification BIDS-folder based**

*STEP - NOT READY.*

**MODULES/ SCRIPTS on REMOTE**

CLUSTER

Crun.py

Read the a/subjtext.txt

Based on the subjects will search for all subjects in the $HOME/projects/def-hanganua/subjects/

Will use the cmd: “recon-all –I subj ID”

Will create the new subjects ID in the $HOME/projects/def-hanganua/fs-subejcsts

Will start the run() loop.

Until all subjects are processed and Moved to “processed\_SUBJECTS\_DIR” folder

Send To Cluster:

for each id:

send all batch files to cluster

send ald DCM folders to cluster

LOGGING STAGE:

initiating cmd on cluster

extracting data

verifying processing stage

LOG file

file (probably csv) that will be updated by the local pipeline and will show:

current status for each id for FS analysis

columns with ‘OK’ answer for:

recon

hip

bstem

stats

*“””*

*One excel/ csv file is provided that had the id’s and groups (‘group’)*

*script extract id’s*

*based on the id - searches in the ‘address\_of\_the\_DCM\_folder’ folders for the specified id*

*classifies time points based on the criteria defined by the user. Criteria - will be provided in the ‘entry data’ in the browser; e.g., user will be asked to type the parameter that differentiates different time points (e.g, \_tp), script will search for ‘\_tp’ in folder’s name and classify accordingly.*

*“””*

Send To Cluster:

for each id:

send all batch files to cluster

send ald DCM folders to cluster

LOGGING STAGE:

initiating cmd on cluster

extracting data

verifying processing stage

LOG file

file (probably csv) that will be updated by the local pipeline and will show:

current status for each id for FS analysis

columns with ‘OK’ answer for:

recon

hip

bstem

stats

**CHECK DATA STAGE:**

check if \_id processed completely and all files present

**STATISTICAL ANALYSIS STAGE: (on local)**

Class **EXTRACT\_STATS**

Extract statistical data for each \_id

Create the excel file with data.

Class **STATISTICAL\_ANALYSIS**:

Perform statistical analysi based on the File\_with\_clinical\_data\_and\_groups

Class **DO\_FREESURFER\_GLM**:

If VM with FS is provided – local app will perform the GLM.

* + saves FDR corrected images
  + save MonteCarlo corrected images => python stats on data
  + distribution (seaborn.distplot; seaborn.jointplot)
  + group descriptions (scipy.stats: ttest\_ind, f\_oneway, bartlett, mannwhitneyu, kruskal)
  + correlations (pandas.DataFrame.corr(pearson, spearman, kendall))
  + anova (statsmodels.formula.api.ols)
  + linear regression (seaborn.lmplot)
  + logistic regression (sklearn.linear\_model.LogisticRegression())