ENIGMA functional processing script

This repository contains processing scripts for internal usage in the ENGIMA functional project. Goal is the improved reproducibility of processing steps across different computer as site in order to facility improved data quality.

- `scripts/` holds processing scripts for AFNI
- `tools/` holds tools used by these processing scripts

The processing environment for running the scripts is provided through private Docker images in DockerHub which is the recommended way of script execution.

Installing Docker

Running a Docker container first requires the installation of the Docker engine. Below a list of instructions for installation on Ubuntu, macOS and Windows is given

Installing Docker on Ubuntu

Docker can be installed form the standard repositories via

```
sudo apt install docker.io
```

In order to verify docker is working we can run

```
docker run hello-world
```

Usually if we want to run a command with `docker` we always need to type `sudo` to obtain the proper rights. On Linux this can be annoying in some cases and can be changed by typing

```
sudo usermod -aG docker $USER
```

Download for macOS

When using macOS, Docker can be installed using the official installer available on the Docker website under macOS

Download for Windows

When using macOS, Docker can be installed using the official installer available on the Docker website under Windows

Loading the Container from DockerHub

The container image is centrally provisioned via the DockerHub which is a central registry for handling preconfigured images for Docker. Currently the pipeline is a private image, which means it is not publicly accessible. Therefore it is required to login with DockerHub.

This can be done by `docker login --username bepipelinepub` , you will then be prompted for your password for that account

Pulling the image from DockerHub

In order to use the image locally, we first need to pull it from DuckerHub. Now that your machine has access to the image after login, the latest version of the image can be pull by typing:

```
docker pull bepipelinepub/pipeline_afni_ants_scripts
```

The output will look similar to

```
$ docker run bepipeline/pipeline
Unable to find image 'bepipelinepub/pipeline_afni_ants_scritps:latest' locally
latest: Pulling from bepipeline/pipeline
b3e1c725a85f: Pull complete
4daad8bdde31: Pull complete
63fe8c0068a8: Pull complete
4a70713c436f: Pull complete
bd842a2105a8: Pull complete
b32a3aa7eb66: Pull complete
607235f51720: Pull complete
6136beb049b5: Pull complete
dd169488eced: Pull complete
8e0dc559149f: Pull complete
37d9df5a4039: Pull complete
Digest: sha256:aca7cc7a17494b617924a386d21cad38ae9c27c24d4286a23f6edee820fa9cf8
Status: Downloaded newer image for bepipelinepub/pipeline_afni_ants_scripts:latest
```

Please note that in the current version of the container (August 2017) with AFNI, MRtrix and several support tools installed, the total image size is about **5.2GB**. However, if only the scripts are updated it is not required to download the whole image again but only some layers in which case the download size will be only about 200MB.

Running commands from within the container

After pulling the **latest** version of the image, we can run it by typing for instance:

```
docker run -it bepipelinepub/pipeline_afni_ants_scripts 3dcalc
```

This will download the image called `pipeline_afni_ants_scripts` provided by the account `bepipelinepub` and

This command will invoke `3dcalc` in the Docker container `pipeline_afni_ants_scripts` which we just pulled from DockerHub. When you see output from the tool on your shell, you know the pipeline is ready for use! :)

Using the AFNI preprocessing pipeline

Expected input files

The processing script was tested on files generated by Charité Berlin Walter lab. The only expected input file for each subject is a EPI for the recording of the task. Registration is done on a shared template (see folder `scripts/afni_ants/templates/`).

The pipeline was tested and verified on input files using the image size of `64x64x32 voxels`. However, there is no general limitation on image size.

The directory structure is as follows:

```
$ tree
.
   afni_pipeline.cfg
   pp01
       pp01_task.nii.gz
   pp02
       pp02_task.nii.gz
...
   pp99
       pp99_task.nii.gz
   templates
       ENIGMA_Template.tlrc.nii.gz
       vent.erode.18.nii.gz
       vent.nii
       wm.erode.18.EPI.nii.gz
       wm.erode.18.nii.gz
       wm.nii
```

The file properties of the input file can be inspected for instance using `3dinfo` from the AFNI package:

```
$ 3dinfo pp01_task.nii.gz
++ 3dinfo: AFNI version=AFNI_17.2.04 (Jul 21 2017) [64-bit]

Dataset File:    pp01_task.nii.gz
```

```
Identifier Code: NII_g3m7Jf31LfekLDbwWarRhA  Creation Date: Tue Aug 15 20:17:03 2017
Template Space:  ORIG
Dataset Type:    Echo Planar (-epan)
Byte Order:      LSB_FIRST {assumed} [this CPU native = LSB_FIRST]
Storage Mode:    NIFTI
Storage Space:   92,274,688 (92 million [mega]) bytes
Geometry String: "MATRIX(3.110602,0.290954,0.096193,-108.6852,0.296439,-3.097027,-0.39453,55
Data Axes Tilt:  Oblique (7.672 deg. from plumb)
Data Axes Approximate Orientation:
  first  (x) = Right-to-Left
  second (y) = Posterior-to-Anterior
  third  (z) = Inferior-to-Superior   [-orient RPI]
R-to-L extent:  -108.685 [R] -to-    88.190 [L] -step-     3.125 mm [ 64 voxels]
A-to-P extent:  -141.137 [A] -to-    55.738 [P] -step-     3.125 mm [ 64 voxels]
I-to-S extent:   -27.892 [I] -to-   102.308 [S] -step-     4.200 mm [ 32 voxels]
Number of time steps = 352  Time step = 2.07000s  Origin = 0.00000s
  -- At sub-brick #0 '?' datum type is short
  -- At sub-brick #1 '?' datum type is short
  -- At sub-brick #2 '?' datum type is short
** For info on all 352 sub-bricks, use '3dinfo -verb' **
```

Invoking the script on data

The full script can be invoked via:

```
docker run -v /home/hlaubish/Desktop/DATA_DIR:/usr/share/data  \
           -it bepipelinepub/pipeline_afni_ants_scripts \
           run_afni_pipeline.py
```

Parameters for AFNI processing script

Configuration of the pipeline is implemented via a configuration script which holds all the possible configuration values in `afni_pipeline.cfg`. Blocks in the processing pipeline can be enabled or disabled as shown in the excerpt below

```
[afni-pipeline-config]
subjectID           = pp01
data_path           = /usr/share/data
run_denoise         = true

; valid options for time shift patters see : ...
time_shift_pattern  = altplus

; export the final files from AFNI to NIFTI-format
run_export_to_nifti = true
```

```
cleanup_files       = true
```

Viewing and Changing processing scripts

The processing scripts are distributed as part of a prepared Docker container image which ensure both a standardized code basis as well as easy distribution of updates. Updates which are integrated into this repository (https://github.com/engima-functional/neuro_processing_scripts) are integrated into the Docker image available on DockerHub under the account/image : `bepipelinepub/pipeline_afni_ants_scripts`