Pythong Virtual Environment?

You might want to do all of the following in a python virtual environment

```
conda create --name jessie_asl python=3.10
```

This will take a bit for it to install some basic python libraries

```
conda activate jessie_asl
```

now you're starting fresh!

Let's install some of the libraries we'll be using:

```
python3 -m pip install dcm2bids -U
```

BIDS

Let's setup a directory:

```
mkdir $PWD/Jessie_ASL && cd Jessie_ASL
dcm2bids_scaffold
```

the `dcm2bids_scaffold` creates some of our basic directories

let's make a directory in sourcedir (which is where we will keep our master par/rec files), and unzip the files you gave me there

```
mkdir sourcedata/sub-004
unzip -d sourcedata/sub-004 ASL_sub-004_Copy.zip
```

dcm2bids helper

I'm not as familiar with Philips data, so let's see what we get when we convert these to nifti. We can use dcm2bids to help:

```
dcm2bids_helper -d sourcedata/sub-004/
```

This is what I get in the folder tmp_dcm2bids/helper:

> 002_sub-004_WIPVBRAIN_3DT1_0.8mm_20210720122519.json 002_sub-004_WIPVBRAIN_3DT1_0.8mm_20210720122519.nii.gz 004_sub-004_WIP3D_pCASL_4mm_20210720122519_ph.json 004_sub-004_WIP3D_pCASL_4mm_20210720122519_ph.nii.gz 004_sub-004_WIPSOURCE-3D_pCASL_4mm_20210720122519.json 004_sub-004_WIPSOURCE-3D_pCASL_4mm_20210720122519.nii.gz 005_sub-004_WIP3D_pCASL_REF_4mm_20210720122519.json 005_sub-004_WIP3D_pCASL_REF_4mm_20210720122519.nii.gz 006_sub-004_WIPFast_3DT1_20210720122519.json 006_sub-004_WIPFast_3DT1_20210720122519.nii.gz 009_sub-004_WIPFast_3DT1_20210720122519.json 009_sub-004_WIPFast_3DT1_20210720122519.nii.gz 011_sub-004_WIPFast_3DT1_20210720122519.json 011_sub-004_WIPFast_3DT1_20210720122519.nii.gz 013_sub-004_WIPFast_3DT1_20210720122519.json 013_sub-004_WIPFast_3DT1_20210720122519.nii.gz

I think we can ignore the WIPFAST ones (006 to 013)

002 seems to be our T1w

004_sub-004_WIP3D_pCASL_4mm_20210720122519_ph looks like maybe the CBF file that the scanner creates for us. You could use that, or you could create (perhaps) a better one using ASLprep

005_sub-004_WIP3D_pCASL_REF_4mm_20210720122519 looks like maybe the proton density (PD) scan. I'm not sure we need it, because if you look at 004_sub-004_WIPSOURCE-3D_pCASL_4mm_20210720122519 (using FSLeyes, for exapmle), it has what look like PD scans at the beginning and the middle.

dcm2bids

ok, let's see if we can get dcm2bids to convert our Philips DICOM files into nifty and BIDS in one go.

First we need to create a `config.json` file that tells dcm2bids what file to convert and place in anat and perf folders. We can place it in the subject's sourcedata folder

```
{
    "descriptions": [
        {
            "dataType": "anat",
            "modalityLabel": "T1w",
            "criteria": {
                "SidecarFilename": "002*"
            }
        },
        {
            "dataType": "perf",
            "modalityLabel": "asl",
            "criteria": {
                "SidecarFilename": "*SOURCE*"
```

```
            },
            "sidecarChanges": {
                "ArterialSpinLabelingType": "PCASL",
                "PostLabelingDelay": 2,
                "BackgroundSuppression": true,
                "M0Type": "Included",
                "RepetitionTimePreparation": 4.15728,
                "LabelingDuration": 1.8,
                "MagneticFieldStrength": 3,
                "MRAcquisitionType": "3D"
            }
        }
    ]
}
```

You'll want to double check the above!

Here I used the 002* to tell dcm2bids that the T1 files start with 002 (this might be different for each subject)

I used *SOURCE* to tell it which was the 4D file that has our PD and control and labels. I also filled out a lot of info w.r.t. how the ASL was acquired. You will want to double check these values with your MRI tech.

```
dcm2bids -d sourcedata/sub-004/ -p 004 -c sourcedata/sub-004/config.json -o
. --forceDcm2niix
```

Here, -d is the directory with the DICOM files, -p is the name you are giving this subject, -c is the path to the config file, -o is the output folder (main BIDS directory), and --forceDcm2niix is not necessary, but if you have run this once before, it will give an error saying these files already exist.

now we have a folder called sub-004 in our main BIDS folder. It looks like this:

```
└── sub-004
    ├── anat
    │   ├── sub-004_T1w.json
    │   └── sub-004_T1w.nii.gz
    └── perf
        ├── sub-004_asl.json
        └── sub-004_asl.nii.gz
```

Next ASLprep needs a .tsv file that tells it how the 4D asl file is organized. It looks like your ASL images have M0 (PD) scans at the beginning and middle, and then control/label scans in between:

```
volume_type
m0scan
m0scan
control
```

```
label
control
label
control
label
control
label
```

save this file as `sub-004_aslcontext.tsv` in the perf folder

.bidsignore

In order to make sure ASLprep works without errors, let's add some files we don't want to be part of BIDS so that they are ignored:

```
echo "tmp_dcm2bids" > .bidsignore
```

For example. If you want to add any other files or folders:

```
echo "ASL_sub-004_Copy.zip" >> .bidsignore
```

The >> appends to the file

bids validator

Now let's check if your data is in BIDS format. ASLprep will crash if it isn't (which is another way of checking, actually)

https://bids-standard.github.io/bids-validator/

Upload your BIDS folder there.

Hopefully you just get Warnings, not Errors. The warnings can be ignored, but if you can fix them, you should!

Docker

Hopefully you have docker installed and working. Otherwise: https://docs.docker.com/desktop/install/mac-install/ (assuming you're on a mac)

You'll also need a Freesurfer license from https://surfer.nmr.mgh.harvard.edu/registration.html

Place the license somewhere (mine is just in my home folder)

Also, make a temporary work directory. This is where the files aslprep creates will be stored. If aslprep crashes, it will also look here to continue from where it left off

```
mkdir ~/tmp_work
```

Now we can try and run aslprep from docker! Note I am in the main BIDS folder:

```
docker run -ti --rm -v ~/license.txt:/license/license.txt -v $PWD:/data:ro
-v $PWD/derivatives:/out:rw -v ~/tmp_work:/work:rw pennlinc/aslprep /data
/out participant --participant-label sub-004 --fs-license
/license/license.txt -w /work --output-spaces T1w
```

There are a lot more options to run. Check them out here:
https://aslprep.readthedocs.io/en/latest/usage.html