2020-01-07 10:43:11

DTI

Starting with DTI, we'll use FSL FDT:

```
cd /scratch/sudregp/dti_fdt
mkdir A01
cd "dMRI_dir107_PA - 21"/
dcm2niix_afni -z y -f PA *
mv PA* ../A01/
cd "../dMRI_dir107_AP - 19"/
dcm2niix_afni -z y -f AP *
mv AP* ../A01/
```

Let's re-orient each file and rename them to make sure we're in the correct space to begin with:

```
cd /scratch/sudregp/dti_fdt/A01/
for i in AP PA; do
    immv ${i} ${i}_old;
    fslreorient2std ${i}_old ${i};
done
```

Now it's mostly following directions from here:

https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FDT/UserGuide

```
# running TOPUP
fslroi AP nodif 0 1
fslroi PA nodif_PA 0 1
fslmerge -t AP_PA_b0 nodif nodif_PA
```

By looking at the JSON and then here:

https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/topup/Faq

I think our parameters are .68 and 140. So, we do:

```
echo "0 -1 0 0.095" > acqparams.txt
echo "0 1 0 0.095" >> acqparams.txt
# this takes a while
topup --imain=AP_PA_b0 --datain=acqparams.txt --config=b02b0.cnf \
    --out=topup_AP_PA_b0 --iout=topup_AP_PA_b0_iout --
fout=topup_AP_PA_b0_fout
```

To check the TOPUP results, load topup_AP_PA_b0_iout and compare its two volumes to those in AP_PA_b0.nii.gz.

If that looks good, we will only need to send it to eddy, instead of applying it beforehand. In other words, if using eddy, no need to use applytopup!

Here's a curious step: from then on FDT only makes use of the first acquisition from according to the documentation. In other words, it only uses AP and PA for TOPUP, but the rest of the analysis is only conducted in one of the datasets? That's odd. I can see the rationale, as structural data shouldn't change, but what about data quality and movement? Should one be making use of both datasets to begin with?

That doesn't make sense to me. So, let's apply TOPUP first, and use that instead:

```
applytopup --imain=AP,PA --topup=topup_AP_PA_b0 --datain=acqparams.txt \
    --inindex=1,2 --out=topup_corrected
```

Next step is applying eddy. Note that bval and bvec are the same in AP and PA!

```
fslmaths topup_AP_PA_b0_iout -Tmean topup_b0_mean
bet topup_b0_mean topup_b0_brain -m -f 0.2
idx=''; for i in {1..108}; do
    a=$a' '1;
done;
echo $a > index.txt
eddy_openmp --imain=topup_corrected --mask=topup_b0_brain_mask \
    --index=index.txt --acqp=acqparams.txt --bvecs=PA.bvec --bvals=PA.bval
\
    --fwhm=0 --flm=quadratic --out=eddy_unwarped_images --data_is_shelled
```

If you have a fancier computer, it's worth running outlier correction in eddy. But because we put together two different sequences here, I'm not going to use slice-to-volume correction.

```
eddy_openmp --imain=topup_corrected --acqp=acqparams.txt --index=index.txt
\
    --mask=topup_b0_brain_mask --bvals=PA.bval --bvecs=PA.bvec \
    --out=eddy_ol_unwarped_images --niter=8 --fwhm=10,6,4,2,0,0,0,0 \
    --repol --ol_type=sw --flm=quadratic  --cnr_maps --data_is_shelled \
    --ol_nstd=4
```

Regardless of what is used, it's good to run some eddy QC:

https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/eddyqc/UsersGuide

```
eddy_quad eddy_unwarped_images -idx index.txt -par acqparams.txt -m
topup_b0_brain_mask.nii.gz -b PA.bval
```

2020-01-08 13:47:52

Then, it's just a matter of checking the quality of the results and running autoPtx and TBSS:

```
# copying over some files to their correct names for bedpostX
cd /scratch/sudregp/dti_fdt/A01
cp eddy_unwarped_images.nii.gz data.nii.gz;
cp PA.bval bvals;
cp PA.bvec old_bvecs;
cp eddy_unwarped_images.eddy_rotated_bvecs bvecs;
cp topup_b0_brain_mask.nii.gz nodif_brain_mask.nii.gz;

cd /scratch/sudregp/dti_fdt
/data/NCR_SBRB/software/autoPtx/autoPtx_1_preproc A01/data.nii.gz
# this takes a while...
```

Now, the next step would be to run step 2 of autoPtx, which is simply running /data/NCR_SBRB/software/autoPtx/autoPtx_2_launchTractography. However, I find that script a big buggy... so, I came up with my own version. For a single subject, all you need to do is:

```
cd /scratch/sudregp/dti_fdt

execPath=/data/NCR_SBRB/software/autoPtx
structures=$execPath/structureList
track=$execPath/trackSubjectStruct

while read structstring; do
    struct=`echo $structstring | awk '{print $1}'`
    nseed=`echo $structstring | awk '{print $2}'`
    $track A01 $struct $nseed 2>&1 > /dev/null &
done < $structures
```

If you're also interested in doing TBSS (voxelwise) analysis, you'd need to also do:

```
# this only needs the output from autoPtx_1
cd /scratch/sudregp/dti_fdt/preproc/A01
tbss_1_preproc dti_FA.nii.gz
```

Let's generate some more QC files, just because...

```
cp origdata/dti_FA.nii.gz ./
cp FA/dti_FA_FA.nii.gz dti_FA_eroded.nii.gz

# make directionality encoded QC pictures, checking that eroded FA looks
fine.
# Note that the dtifit results, and the std alignment were run by autoPtx!
fat_proc_decmap -in_fa dti_FA_eroded.nii.gz -in_v1 dti_V1.nii.gz \
    -mask nodif_brain_mask.nii.gz -prefix DEC

# apply the transform calculated by autoPtx to a few maps. code copied
from
# tbss_non_fa
for f in FA L1 L2 L3 MD MO; do
    echo Warping $f;
    applywarp -i dti_${f} -o ${f}_in_FMRIB58_FA_1mm \
        -r $FSLDIR/data/standard/FMRIB58_FA_1mm -w nat2std_warp
done

# make transformation QC figure: FSL template as the edges
@snapshot_volreg FA_in_FMRIB58_FA_1mm.nii.gz \
    $FSLDIR/data/standard/FMRIB58_FA_1mm.nii.gz \
    QC/FA_transform;

# make QC images for standard errors. Here we set our color scale to have
95th
# percentile of all errors. Meaning, more red = bigger error.
@chauffeur_afni                             \
    -ulay  data.nii.gz                       \
    -olay  dti_sse.nii.gz                          \
    -opacity 5                              \
    -pbar_posonly   \
    -cbar Spectrum:red_to_blue              \
    -set_subbricks 0 0 0     \
    -prefix   QC/sse              \
    -montx 6 -monty 6                       \
    -set_xhairs OFF                         \
    -label_mode 1 -label_size 3             \
    -thr_olay 0 \
    -func_range_perc_nz 95 \
    -do_clean
```

fMRI

Here the approach will largely process AP and PA independently, and then towards the end either concanetante both time series or just average the connectivity metrics of the two runs.

fmriprep needs data in BIDS format, so let's do that:

```
m=A01;
cd /scratch/sudregp/
```

```
mkdir fmri
cd fmri
jo -p "Name"="test BIDS" "BIDSVersion"="1.0.2" >>
dataset_description.json;
mkdir -p sub-${m}/anat;
dcm2niix_afni -o sub-${m}/anat/ -z y -f sub-${m}_T1w "../T1w_MPR - 16"/
mkdir -p sub-${m}/func;
dcm2niix_afni -o sub-${m}/func/ -z y \
    -f sub-${m}_task-rest_acq-PA_bold "../rfMRI_REST_PA - 15/"
dcm2niix_afni -o sub-${m}/func/ -z y \
    -f sub-${m}_task-rest_acq-AP_bold "../rfMRI_REST_AP - 13/"
```

Then, just run fmriprep. Note that this will be different in Biowulf, but the actual command you'll run is likely the same:

```
export TMPDIR=/lscratch/$SLURM_JOBID;
m=A01;
mkdir -p $TMPDIR/out $TMPDIR/wrk;
fmriprep /scratch/sudregp/fmri/ $TMPDIR/out \
    participant --participant_label sub-${m} -w $TMPDIR/wrk --use-aroma \
    --nthreads 32 --mem_mb 50000 --notrack \
    --fs-license-file /usr/local/apps/freesurfer/license.txt --fs-no-
reconall;
mv $TMPDIR/out/* /scratch/sudregp/fmriprep_output
```

The result will be in /scratch/sudregp/fmriprep_output.

2020-01-14 08:29:50

Returning to this, let's see how to run two different files in xcpengine, if at all possible:

```
export TMPDIR=/lscratch/$SLURM_JOBID;
m=A01;
echo id0,id1,img > ${TMPDIR}/${m}.csv;
cp /data/NCR_SBRB/fc-36p_despike.dsn $TMPDIR/;
echo sub-${m},AP,fmriprep/sub-${m}/func/sub-${m}_task-rest_acq-AP_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz >> ${TMPDIR}/${m}.csv;
echo sub-${m},PA,fmriprep/sub-${m}/func/sub-${m}_task-rest_acq-PA_space-
MNI152NLin2009cAsym_desc-preproc_bold.nii.gz >> ${TMPDIR}/${m}.csv;
xcpEngine -c $TMPDIR/${m}.csv -d $TMPDIR/fc-36p_despike.dsn -i $TMPDIR/wrk
\
    -o $TMPDIR/out -r /scratch/sudregp/fmriprep_output/;
mv $TMPDIR/out/* /scratch/sudregp/xcpengine_output/;
```

Result will be in /scratch/sudregp/xcpengine_output/.

Then, you have two options: average whatever connectivity metrics you want to use (e.g. correlation, ReHo, ALLF), which were computed separately for AP and PA, between the two runs. If doing that, it might be good to also check which connections have a big difference. For example, computing the signed difference across subjects and doing a t-test.

The alternative is to concatenate both processed runs over time (assuming both processed without errors), and re-running xcpengine on the concatenated files. It would go something like:

```
export TMPDIR=/lscratch/$SLURM_JOBID;
m=A01;
cd /scratch/sudregp/xcpengine_output/sub-${m}
3dTcat -prefix sub-A01_concat.nii.gz \
    AP/regress/sub-A01_AP_residualised.nii.gz \
    PA/regress/sub-A01_PA_residualised.nii.gz
cp /data/NCR_SBRB/fc-template.dsn $TMPDIR/;
echo id0,id1,img > ${TMPDIR}/${m}.csv;
echo sub-${m},concat,sub-${m}_concat.nii.gz >> ${TMPDIR}/${m}.csv;

xcpEngine -c $TMPDIR/${m}.csv -d $TMPDIR/fc-template.dsn -i $TMPDIR/wrk \
    -o $TMPDIR/out -r /scratch/sudregp/xcpengine_output/sub-${m}/;
mv $TMPDIR/out/* /scratch/sudregp/xcpengine_output/;
```

Not sure what the differences between the two methods would be. The concatenation method introduces a discontinuity between the two runs, which will affect measures like Reho and ALFF, but not Pearson correlation. The latter would have more degrees of freedom, and that way increase certainty.

There will be some difference in the results because the concatenation is done in template space, while the within-run metrics are computed in subject space.