TBSS - Tract-Based Spatial Statistics

We will now run a TBSS analysis of a small dataset - 3 young people (mean age 25) and 3 older people (mean age 65). We will attempt to find where on the tract skeleton the two groups of subjects have significantly different FA.

Additional informations

http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/TBSS

http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Randomise

---

0 - creating FA data from a diffusion study

We have already created the FA images from the 6 subjects, using dtifit.

```
cd DATA_TP/tbss
```

Remember that all of the main scripts (until you reach the `randomise` stage near the end) need to be run from within this directory.

LOOK AT YOUR DATA:

- Open one image with fslview to get a feel for the raw data quality and resolution. You may need to adjust the intensity display range. Look at the image histogram.
- Run `slicesdir *.nii.gz` and open the resulting web page report; this is a quick-and-dirty way of checking through all the original images.

---

1 - preparing your FA data for TBSS

You are now nearly ready to run the first TBSS script, which will erode your FA images slightly to remove brain-edge artefacts and zero the end slices (again to remove likely outliers from the diffusion tensor fitting). Type:

```
tbss_1_preproc *.nii.gz
```

The script places its output in a newly-created sub-directory called `FA`. It will also create a sub-directory called `origdata` and place all your original images in there for posterity.

At this point you can move onto the next script, but first it is probably worth checking the images as created in FA. LOOK AT YOUR DATA - view the output of `slicesdir`

---

2 - registering all the FA data

The next TBSS script runs the nonlinear registration, aligning all the FA data across subjects. The recommended approach is to align every FA image to the FMRIB58_FA template. This process can

take a long time, as each registration takes around 10 minutes. You can easily speed this up if you have multiple computers running cluster software such as SGE (Sun Grid Engine). To save time, we have pre-computed all the alignments for you, so instead of running the TBSS script `tbss_2_reg`, make sure you are still in the directory one level above `FA`, and run

```
cp precomputed_registration/* FA
```

This puts the necessary registration output files into FA, as if you had run the registration yourself.

---

3 - post-registration processing

Before reading this section, start the next script running so that you can read what's going on while it runs (it will take about 5 minutes):

```
tbss_3_postreg -S
```

The previous script only got as far as registering all subjects to the chosen template. The `tbss_3_postreg` script applies these registrations to take all subjects into 1x1x1mm standard space.

Next, all subjects' standard space nonlinearly aligned images are merged into a single 4D image file called `all_FA`, created in a new subdirectory called `stats`. The mean of all FA images is created, called `mean_FA`, and this is then fed into the FA skeletonisation program to create `mean_FA_skeleton`.

Once the script has finished running, check that the mean FA image looks reasonable, and is well aligned with the MNI152 image:

```
cd stats
fslview $FSLDIR/data/standard/MNI152_T1_1mm mean_FA -l Red-Yellow -b .2,.6
```

As you move around in the image you should see that the mean FA image is indeed well aligned to standard space and corresponds to white matter in the MNI152 image. Now kill FSLView and restart it, loading the 4D aligned FA data underneath the skeleton:

```
fslview all_FA mean_FA_skeleton -l Green -b .2,.6
```

Now turn on the movie loop; you will see the mean FA skeleton on top of each different subject's aligned FA image. If all the processing so far has worked, the skeleton should look like the examples shown in the lecture. If the registration has worked well you should see that in general each subject's major tracts are reasonably well aligned to the relevant parts of the skeleton. If you set the skeleton threshold (in FSLView, the lower of the display range settings) much lower than 0.2, it will extend away towards extremes where there is too much cross-subject variability and where the nonlinear registration has not been able to give good alignments. In this dataset, with very few subjects, the mean FA image is quite noisy, so you probably want the threshold around 0.3.

---

4 - projecting all pre-aligned FA data onto the skeleton

The last TBSS script carries out the final steps necessary before you run the voxelwise cross-subject stats. It thresholds the mean FA skeleton image at the chosen threshold:

If you're still in the `stats` directory

```
cd ..
```

Then

```
tbss_4_prestats 0.3
```

This takes 4-5 minutes to run; read the rest of this section while it's running (if you can't make much sense of it, don't worry - it's decribed in more detail in the paper!).

The thresholding creates a binary skeleton mask that defines the set of voxels used in all subsequent processing.

Next a "distance map" is created from the skeleton mask. This is used in the projection of each subject's FA onto the skeleton; when searching outwards from a skeleton voxel for the local tract centre, the search only continues while the distance map values keep increasing - this means that the search knows to stop when it has got more than halfway between the starting skeleton point and another separate part of the skeleton.

Finally, the script takes the 4D pre-aligned FA images in `all_FA` and, for each "timepoint" (subject ID), projects the FA data onto the mean FA skeleton. This results in a 4D image file containing the (projected) skeletonised FA data. It is this file that you will feed into voxelwise statistics in the next section.

Once the script has finished, `cd` into `stats` and have a look at `all_FA_skeletonised` in fslview - turn on movie mode to see the different timepoints of the skeletonised data.

---

5 - voxelwise statistics on the skeletonised FA data

The previous step resulted in the 4D skeletonised FA image. It is this that you now feed into voxelwise statistics, that, for example, tells you which FA skeleton voxels are significantly different between two groups of subjects.

The recommended way of doing the stats is to use the `randomise` tool. For more detail see the randomise manual. Before running `randomise` you will need to generate design matrix and contrast files (e.g., `design.mat` and `design.con`). We will use the **Glm** GUI to generate these. Note that the order of the entries (rows) in your design matrix *must* match the alphabetical order of your original FA images, as that determines the order of the aligned FA images in the final 4D file `all_FA_skeletonised`. In this case the order was: 3 young subjects and then 3 older subjects. Start the GUI:

```
cd stats
Glm
```

Change `Timeseries design` to `Higher-level / non-timeseries design`. Change the number of inputs to 6 (you may have to press `Return` after typing in `6`) and then use the wizard to setup the two-group unpaired t-test. Reduce the number of contrasts to 2 (we're not interested in the group means on their own). Finally, save the design as filename `design`, and in the terminal use `more` to look at the `design.mat` and `design.con` files.

You are now ready to run the stats. Because this reduced dataset only contains 6 subjects, only 20 distinct permutations are possible, so by default randomise will run just these 20. Again, because this tiny dataset has so few subjects the raw t-stat values will not be very significant - so let's try setting a cluster-forming t threshold of 1.5 (in "real" datasets we would normally recommend using the --T2 option for TFCE instead of cluster-based thresholding):

```
randomise -i all_FA_skeletonised -o tbss -m mean_FA_skeleton_mask \
-d design.mat -t design.con -c 1.5
```

Contrast 1 gives the young>older test. The raw unthresholded tstat image is `tbss_tstat1` and the corresponding (p-values corrected for multiple comparisons) cluster image is `tbss_clustere_corrp_tstat1`.

Thresholding clusters at 0.949 (corresponding to thresholding the p-values at 0.05, because randomise outputs p-values as 1-p for convenience of display - so that higher values are more significant) shows a few areas of reduced FA in the older subjects. The following shows unthresholded t-stats in red-yellow and thresholded clusters in blue:

```
fslview $FSLDIR/data/standard/MNI152_T1_1mm mean_FA_skeleton -l Green \
 -b .3,.7 tbss_tstat1 -l Red-Yellow -b 1.5,3 tbss_clustere_corrp_tstat1 \
 -l Blue-Lightblue -b 0.949,1
```

Look at the histogram of tstat1; it is clearly shifted to the right, suggesting a global decrease in FA with aging (note: you may need to change the number of histogram bins to see this easily).

If you prefer to display "fattened" results (that are not just the one-voxel-thick skeleton, but "fill out" into the tracts as seen in mean FA image), whilst still masking with the mean FA image, there is a simple script which will do this for you, then run:

```
tbss_fill tbss_clustere_corrp_tstat1 0.949 mean_FA
tbss_clustere_corrp_tstat1_filled
```

and then load this output into FSLView and choose an appropriate colourmap. This script smooths the thresholded input image (by a typical tract width according to the local tract structure), multiples by the mean FA image (in order to constrain the width) and then adds the original input back in so that the skeleton can still be seen within the thickened output.
