Mrtrix APOE pipeline:

The following steps are performed in the Mrtrix APOE pipeline:

1. We start by reading the necessary 4D, anatomical, bvec, and bval files.
2. Using the output of dwigrdcheck, we swap the x and y components of the bvec file and flip the signs of all the components.
3. We convert the anatomical and 4D image to the mif format for Mrtrix usage, resulting in T1\_mif and out\_mif files.
4. By utilizing the "dwi2tensor" and "tensor2metric" commands, we generate several diffusion tensor-related metrics, including dt, fa, dk, mk, md, ad, and rd.
5. We create subject-specific label images using samba, and these labels are employed to create a mask. In the creation of the mask, we exclude the cerebrospinal fluid (CSF) regions based on the atlas. The resulting mask is referred to as mask\_of\_label. Note that the dwigradcheck step, which determines the correct bvec transformation, can only be performed at this stage since it requires the mask. Once the transformation is determined, it is applied in step 2 and remains constant for all other subjects.
6. The dwi2response command, utilizing the dhollander algorithm, estimates the basis functions for white matter (wm), gray matter (gm), and CSF.
7. Applying the estimated basis functions to the diffusion data, we compute the fiber orientation distribution (fod) files, namely wmfod\_mif, gmfod\_mif, and csffod\_mif, using the "dwi2fod" command. These fod files are then normalized to wmfod\_norm\_mif using the "mtnormalise" command.
8. We create 10 million tracts using the gmwmSeed\_coreg\_mif as the seed image (representing the entire masked brain). From these 10 million tracts, we subset and select 2 million tracts for each subject, with a fractional anisotropy (fa) cutoff of 0.1 and a maximum length of 1000 mm. During the subsetting process, a minimum length of 0.1 is set to avoid excessive noise. The following system commands are executed:  
   1. os.system('tckgen -backtrack -seed\_image '+ gmwmSeed\_coreg\_mif + ' -maxlength 1000 -cutoff 0.1 -select 10000000 ' + wmfod\_norm\_mif + ' ' + tracks\_10M\_tck + ' -force')
   2. os.system('tckedit '+ tracks\_10M\_tck + ' -number 2000000 -minlength 0.1 ' + smallerTracks + ' -force')
9. Using the subject-specific labels we make parcel\_mif to plug in to tck2connectome and creates 5 types of connectomes for each subject:
   1. sift node means I used the -scale\_invnodevol -tck\_weights\_in options of the tck2connectome commandand and fed sift\_1M\_txt to it.
      1. -scale\_invnodevol scale each contribution to the connectome edge by the inverse of the two node volumes
      2. -tck\_weights\_in specify a text scalar file containing the streamline weights
      3. -sift\_1M\_txt comes from tcksift2 command that Optimises per-streamline cross-section multipliers to match a whole-brain tractogram to fixel-wise fibre densities and it must be the weighting factor for each streamline
   2. sift is simimilar to the above without -scale\_invnodevol
   3. distances connectome made by using -scale\_length stat\_edge  mean
      1. -scale\_length scale each contribution to the connectome edge by the length of the streamline
      2. stat\_edge  mean: statistic for combining the values from all streamlines in an edge into a single scale value for that edge (options are: sum, mean, min, max; default=sum)
   4. FA\_connectome are made by using -scale\_file ' and feeding mean\_FA\_per\_streamline plus -stat\_edge mean like above.
      1. -scale\_filescale each contribution to the connectome edge according to the values in a vector file
      2. mean\_FA\_per\_streamline is made using tcksample command with  -stat\_tck mean option  
         1. this command samples values of an associated image along tracks
         2. -stat\_tck mean computes some statistic from the values along each streamline
   5. Plain connectome only uses -symmetric -zero\_diagonal that all other 4 commands are already using as an option when creating the connectomes.