CENTER FOR FUNCTIONAL & MOLECULAR IMAGING
GEORGETOWN UNIVERSITY MEDICAL CENTER

Brain Morphometry Analyses in FreeSurfer I: Reconstructing Structural MRI Scans

Made for CFMI training

4 November 2015
(updated 15 June 2016)

1. INTRODUCTION

Freesurfer (FS) is a powerful (and freely available!) software package developed at the Martinos Center for Biomedical Imaging for automatized analyses of brain anatomy. A non-inclusive list of its capabilities includes:

- Performing morphometric analysis of the brain
  (i.e. cortical thickness, curvature, surface area, volume)
- Creating study-specific anatomical templates
- Optimal alignment of cortical folding patterns across individuals
- Processing of functional imaging data using the FSFAST software suite
- Segmenting and aligning individual brain volumes to an anatomical atlas
- Rendering 3D models of the pial surface, white-matter and user-defined masks

Although FS possesses remarkable capabilities that greatly reduce the effort involved in analyzing structural MRI data, there are crucial caveats the user must be aware of. This document is the first in a series of training guides intended to bring a novice user up to speed on using and trouble-shooting FS. In this guide, we will describe how to install and configure your system for FS and then walk you through the steps of reconstructing a brain scan from raw data and viewing the final results. Part II in this series, "Trouble Shooting & Manual Editing" will outline steps for artifact correction and manual intervention at various steps along the FS anatomical analysis workflow. Part III will cover statistical methods for morphometric analyses.

The materials presented here are collected from miscellaneous notes taken from the FreeSurfer Wiki, list-serv user postings, published papers and independent experimentation. If a question comes up that is not addressed in this guide be sure to consult these resources first! If all else fails, ask questions! *(Also see Appendix)*

1.1. Background

Cortical Surface Reconstruction FS initially emerged from efforts to constrain the inverse problem for EEG/MEG source localization using anatomically accurate cortical surface models [1]. Dale & Sereno's major innovation was to use the gray/white boundary instead of the pial surface to reconstruct the cortex – effectively curtailing the challenge of conforming a continuous spherical surface to a highly irregular and non-continuous cortical surface. However, these early attempts did not produce accurate surface models and required extensive user intervention to correct topological defects. The accuracy of surface deformation techniques is constrained by three considerations:

1. **Iso-Intensity Surfaces** – the assumption that the MRI intensity of gray/white and pial surfaces are constant. However, this assumption is not accurate. Cortical tissue varies in microanatomy and thus the signal intensity is quite variable across cortex. For example, myelin-rich regions will appear much brighter than myelin-poor regions.

2. **Topological Constraints** – surfaces must be constrained to not self intersect. This problem is made exceptionally difficult since the cortex is tightly packed making the demarcation between adjacent sulcal banks a hard computational problem for planar Eulerian methods.

3. **Curvature minimization** – smooth surfaces are generated through curvature minimization, however cortex contains many regions with high curvature.

Fischl & Dale addressed concern #1 by adaptively determining MR intensity thresholds for a boundary at each point in an image, concern #2 by avoiding curvature minimization and using a second-order polynomial patch instead of a plane surface and concern #3 by applying fast triangle-triangle intersection borrowed from the computer graphics literature [4]. This method proved to produce accurate and sensitive (up to $\frac{1}{4}$ mm) measurements of cortical thickness that withstood considerable variation in sequence parameters, acquisition artifacts and tissue variability [5]. Many groups have now validated Fischl & Dale's reconstruction algorithm with histological and manual measures across many scanner platforms, in the presence of subject motion, different acquisition parameters, correlations with behavioral measures and analysis types [3, 8, 13, 12, 17].

Whole Brain Segmentation Once the cortical surface reconstruction approach had been tackled, Fischl et al. turned their attention to whole brain segmentation. The major difficulty in performing algorithmic volumetric segmentation of the gray matter is the high variability of image intensity across brain structures. A Bayesian approach was adopted to solve this problem using a different model to compute the image likelihood at each position in space. A non-stationary, anistrophic Markov Random Field model was used to define the prior on structure identity given relative spatial location to other macro-structures. The technique was validated using a training set of manually labeled images, achieving comparable performance to extensively trained personnel and exhibited excellent performance with pathological samples [6, 7].

Other Contributions Many other groups have contributed improvements and additional modules to the FS software suite:

- Tractography [18]
- Alignment of Cortical Folding Patterns and Subcortical/Ventricular Structures [10]
- Analysis of Cortical Curvature Development [9]
- Skull Stripping [15]
- Statistical Models for Longitudinal Analyses [11]
- Gyral Based Cortical Parcellation [2]

- Insensitivity to Pulse Sequences [8]
- High Resolution Segmentation of Hippocampal Subfields [16]
- Probabilistic Label Segmentation [14]

Thanks to the efforts of those that validated and built-upon the software, cognitive neuroscientists and clinicians can accurately measure a variety of morphometrics in relation to brain function. Now that you are stoked after perusing this exciting scientific historical backdrop, continue on to learn how to install FS and run your first cortical reconstruction.

2. Installation & System Configuration

The following installation instructions are tailored to OS X/Linux users with some familiarity of command line usage. Please see the FS instructions if you are an unfortunate Microsoft user. Procedural hints for these instructions are included in the Appendix. If anything is confusing at all don't hesitate to ask!

1. Download FreeSurfer
2. Add the FS start-up routine to your `.bash_profile`. To do this open a terminal, cd into your home directory and open `.bash_profile` in a text editing program such as emacs, vi or nano

```
cd ~
emacs .bash_profile
```

   Edit your .bash_profile to include the commands below. Make sure the she-bang (`#!/bin/bash`) is included at the top if you are creating the file from scratch. This tells your terminal what program is used to run these commands.

```
#!/bin/bash
# FREESURFER SETUP
export FREESURFER_HOME="/Applications/freesurfer"
source $FREESURFER_HOME/SetUpFreeSurfer.sh
export SUBJECTS_DIR="/Applications/freesurfer/subjects"
```

   After verifying everything looks ok, type the `control` key then `x`, the `control` key again, then `c` to exit and save (for emacs). You will be prompted whether you want to save the file before exiting. Type `y` then hit `enter` to do so.

   The commands you added to your `.bash_profile` put the FS tools at your disposal each time you open a terminal window. Note that you may eventually want to change your `SUBJECTS_DIR` to the path of your own data directory:

```
SUBJECTS_DIR="/exports/home/<cfmiuser>/ADS/ads.subjects"
```

3. Next open up the `.dmg` file you downloaded in step 1 and go through the installer.
4. Then register on the FS website to get a license.

5. When it is emailed to you, use `emacs` to create a license file then copy and paste in the key. You will need an administrator password for this since you must use super-user do:

```
sudo emacs /Applications/freesurfer/.license
```

6. IMPORTANT: be sure to source your `.bash_profile` so you can use FS at the terminal

```
cd ~
. .bash_profile
```

7. Run your first test of the installation by typing the following into a terminal

```
tkmedit $FREESURFER_HOME/subjects/bert orig.mgz
```

3. Reconstruct them all!

3.1. Quick-Start Tutorial

The most attractive feature of FS is that it makes cortical surface reconstruction and subcortical segmentation for a single subject as easy as typing in a single command and coming back the next day to check your results. To get started, you must set up your environment and import your imaging data.

Environment Setup You may clone a repository containing all of the tools you will require for this tutorial by typing the following into a terminal window

```
cd ~
git clone https://github.com/seldamat/Surfer-gems.git
```

The `ls` command can be used to view the contents of this repository

```
ls ~/Surfer-gems
FS-stats
FSQC-check
FSQC-makehtml
FSQC-summary
README.md
doc
nextract.batch
nextractomatic
recon-all-go
ship-data
```

General information about the repository can be found within the `README` file. The `doc` directory contains the LaTeX files used to generate this document. Note that this repository is updated frequently. The git command line tool can keep you up to date with the newest versions.

Usage information about any of the programs can be obtained by calling the program by name with an appended `-h` option. For example,

```
cd ~/Surfer-gems
ship-data -h

ship-data - A wrapper for the rsync function to quickly sync large data directories.

  Usage   :: ship-data DATASOURCE DESTINATIONFOLDER

  Example :: ship-data ~/mydata ~/projectfolder

  Extra Info ::

    !! the datasource will be placed inside the project folder

    !! If the project folder contains a directory with the same
      name as the data source, then the folder will not be
      overwritten but rater updated.  Only newer files will be
      kept.  Older files are overwritten.

    !! Do not use slashes at the end of directory paths

    !! sync information is written to shipping.log, a file
    located in the project folder
```

IMPORTING RAW DATA At the CFMI, the original `.IMA` DICOM files are usually downloaded or copied from the raw data directory on the server and converted to nifty (`.nii`) or analyze (`.img/.hdr`) format. A program for performing automatic conversion and data duplication can be found at:

```
~/Surfer-gems/nextractomatic
```

This program uses the `dcm2niix` conversion software and is written specifically to handle the CFMI mySQL database. Software such as MRICONVERT or MRICRON may also be used for this purpose. An example call of `nextractomatic` appears below,

```
nextractomatic \
    -r /exports/home/se394/ADS/data/raw/w1 \  # Path to raw dicom .IMA data
    -s "149959" \                             # Subject ID
    -dm 2012-6-14 -dx 2012-6-15 \             # Date range of scans to extract
    -rf .nii \                                # Conversion format
    -t "MPRAGE Rest" \                        # Scan names to extract
    -ra "Anat Rest"                           # New names for scans
```

Sometimes pre-existing data may have already been converted and stored in a central project directory on the server. If this is the case, it is more efficient to copy the data over to your own local user project directory, perform your analyses then ship the data back to the central project directory. To accomplish this, you can use the `ship-data` program found within the `Surfer-gems` repository.

Once the environment for FS has been set, you are ready to analyze your first subject. First you must import the raw image files to your `SUBJECTS_DIR` by typing the following into a terminal:

```
recon-all -s <subjectname> -i <pathtorawdata1.nii> \
                           -i <pathtorawdata2.nii>
```

The `-i` option copies the raw image files to the directory structure required for running the `recon-all` command. Multiple input images can be specified if more than one run exists for a given subject. These runs can be averaged together to generate cleaner images.

After importing your data to the FreeSurfer directory structure, take a look in `SUBJECTS_DIR` and you should see a folder for the subject with the following directories:

```
ls $SUBJECTS_DIR/<subjectname>
bem
label
mri
scripts
src
stats
surf
temp
touch
trash
```

We will explain the purpose of some of these folders later. For now, you can view the original raw mri

```
tkmedit <subjectname> orig/001.mgz
```

Once it is confirmed that the subject has been successfully imported, we can begin the processing stream

```
recon-all -s <subjectname> -all
```

The pre-processing can take up to 24 or as little as 4 hours depending on your machine. Optimizing this work flow is absolutely crucial when performing a large study with many subjects.

3.2. Batch Processing

Batches of subjects can be run by writing bash scripts to loop across a list of subjects, invoking recon-all for each one. As previously mentioned, each individual subject can take up to 24 hours to complete reconstruction thus it is highly desirable to run your batches in parallel across as many nodes as possible. A user can run as many recon-all sessions on a single system as there are processor cores or nodes on that system.

LockFiles & Adaptive Iteration One way to make parallel FS processing more efficient is to prevent different instances of `recon-all` from processing the same subject by using a lock-file. The lock-file is a temporary file that is created as a particular subject is currently being processed so that any new invocations of `recon-all` know to move on to the next unprocessed subject.

The problem becomes much more complicated when multiple users are running `recon-all` with large subject databases hosted on a server. In this instance, we avoid re-processing already completed subjects by cross-checking each `recon-all` call with a list of completed subjects. In conjunction with a lock-file, this prevents different users from processing or re-processing the same subject.

SERIAL ITERATION These considerations are implemented in the batch processing program `recon-all-go`. This executable requires a path to the raw data directory, image format, subject grouping suffix as input and list of subjects. The subject group is null,", by default (i.e. if not provided). For example:

```
recon-all-go -rd /Volumes/CFMI/FreeSurfer/data/raw -rf .nii -g wl -s subjects
```

Be sure to inspect the program usage with the `-h` option before running your first batch with `recon-all-go`. The program will automatically read the default subjects directory, however you can input a different directory using the `-sd` option. This program will output each step of the reconstruction stream to the terminal and will append total run time and time of completion to a list in the subject directory.

PARALLEL ITERATION If you have access to a multicore machine you may be interested in running many invocations of`recon-all` at the same time. A quick and dirty shortcut for parallelizing `recon-all` is to run the command in as many terminal prompts as there are processors. Unfortunately, since the quick and dirty method parallelizes at the highest level, system resources are not efficiently utilized.

The GNU parallel tool is an excellent way for efficiently parallelizing any job on a Mac or Linux machine. You can install this tool using the homebrew package installer for OS X,

```
brew install gnuparallel
```

Once installed, a quick way to get started is to put all of your subjects' anatomical scans in a single folder and typing the following command to import and run in a single call,

```
cd ~/Projects/raw
ls -d *.nii | parallel recon-all -s {.} -i {} -all
```

4. PERUSING THE RESULTS

FS comes with multiple image visualization programs for editing, viewing and overlaying anatomical/-functional brain scans. `Tksurfer` and `Tkmedit` are older, C-based legacy programs whereas `Freeview` is much newer and has a user-friendly design. To view a subject's T1, white matter mask, brain mask subcortical segmentation and surface reconstruction in `Freeview`, open a terminal and type:

```
subjid=<subjectid>
freeview \
-v $SUBJECTS_DIR/$subjid/mri/brainmask.mgz \
$SUBJECTS_DIR/$subjid/mri/aseg.mgz:colormap=lut:opacity=0.2 \
$SUBJECTS_DIR/$subjid/mri/T1.mgz \
$SUBJECTS_DIR/$subjid/mri/wm.mgz:colormap=heat:opacity=0.4:visible=0 \
-f $SUBJECTS_DIR/$subjid/surf/lh.white:edgecolor=blue \
$SUBJECTS_DIR/$subjid/surf/lh.pial:edgecolor=red \
$SUBJECTS_DIR/$subjid/surf/rh.white:edgecolor=blue \
$SUBJECTS_DIR/$subjid/surf/rh.pial:edgecolor=red \
$SUBJECTS_DIR/$subjid/surf/lh.smoothwm.nofix:visible=0 \
$SUBJECTS_DIR/$subjid/surf/rh.smoothwm.nofix:visible=0 \
$SUBJECTS_DIR/$subjid/surf/lh.inflated:visible=0 \
$SUBJECTS_DIR/$subjid/surf/rh.inflated:visible=0
```

The -v flag is used to open volume data and the -f flag is used to load surfaces. You can indicate which colormap, color, opacity and other plotting options by indicating them with a colon following the file name. We can create an alias, or a shortcut, to avoid typing this very long command every time you want to visualize a new subject. To define an alias, open your .bash_profile and include the following commands.

```
# Check to see if subject is defined, if not set to default bert
if [ -z $subjid ]; then
  export subjid="bert"
fi
# Define your command shortcut using alias
alias launchfv="freeview \
 -v $SUBJECTS_DIR/$subjid/mri/brainmask.mgz \
$SUBJECTS_DIR/$subjid/mri/aseg.mgz:colormap=lut:opacity=0.2\
$SUBJECTS_DIR/$subjid/mri/T1.mgz \
$SUBJECTS_DIR/$subjid/mri/wm.mgz:colormap=heat:opacity=0.4:visible=0 \
-f $SUBJECTS_DIR/$subjid/surf/lh.white:edgecolor=blue \
$SUBJECTS_DIR/$subjid/surf/lh.pial:edgecolor=red \
$SUBJECTS_DIR/$subjid/surf/rh.white:edgecolor=blue \
$SUBJECTS_DIR/$subjid/surf/rh.pial:edgecolor=red \
$SUBJECTS_DIR/$subjid/surf/lh.smoothwm.nofix:visible=0 \
$SUBJECTS_DIR/$subjid/surf/rh.smoothwm.nofix:visible=0 \
$SUBJECTS_DIR/$subjid/surf/lh.inflated:visible=0 \
$SUBJECTS_DIR/$subjid/surf/rh.inflated:visible=0"
```

The alias can be invoked from a terminal after you source your .bash_profile. Note that you must define your subjects directory, $SUBJECTS_DIR, and the specific subject, $subjid, you want to see. Once you add the alias, you may use it like-so:

```
subjid="149959"
. .bash_profile #source the new commands, do this for every new subjid definition
launchfv
```

You can check to see your command is loading the appropriate data by typing,

```
type launchfv
```

Freeview gives you many options to view and manipulate your subject's processed results. To learn more about the Freeview graphical interface and key-mappings to functions please read this tutorial.

5. UNDER-THE-HOOD

While you are running your first subject you may notice that the terminal will quickly begin filling up with text describing the processing status, any errors that may have been encountered and details regarding individuals steps taken. You can get more information about each of these steps and the usage of recon-all by opening a new terminal and typing:

```
recon-all -help
```

The help command also provides you with information regarding optional arguments you can flag each recon-all call with. Of particular importance are the autorecon flags:

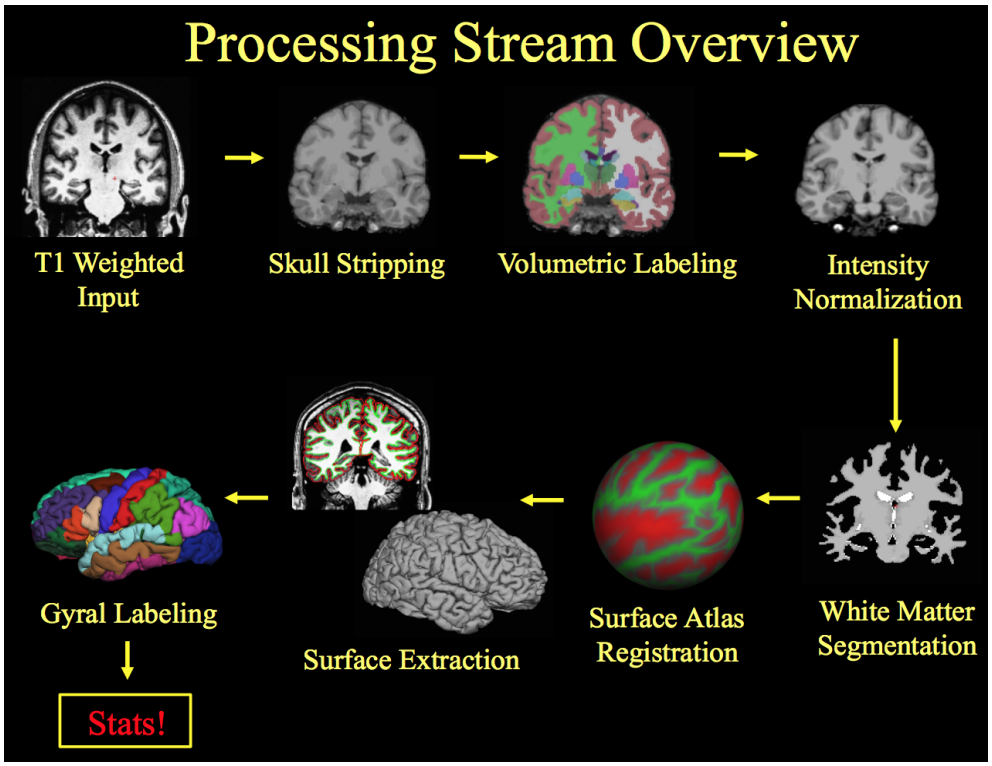

Figure 5.1: **FreeSurfer Processing Stream.**

```
-autorecon1   :   process stages 1-5
-autorecon2   :   process stages 6-23
-autorecon3   :   process stages 24-34
```

Each of these flags can take additional options for algorithm parameters or to only run a subset of the processing stages. The individual stages and some optional parameters of the FS pre-processing pipeline are summarized below.

RECON-ALL CONSISTS OF 34 PROCESSING STAGES:

1. **Motion Correction and Conform** – Multiple scans from the same subject are averaged together to minimize motion artifacts. The input are the volumes found in file(s) `mri/orig/XXX.mgz`. The output will be the volume `mri/orig.mgz`. If no runs are found, then it looks for a volume in `mri/orig`. If that volume is there, then it is used in subsequent processes as if it was the motion corrected volume. If no volume is found, then the process exits with errors.

2. **NU (Non-Uniform intensity normalization)** – Non-parametric Non-uniform intensity Normalization (N3), corrects for intensity non-uniformity in MR data, making relatively few assumptions about the data. This runs the MINC tool `nu_correct`. By default, four iterations of `nu_correct` are run. The flag `-nuiterations` specification of some other number of iterations.

3. **Talairach transform computation** – The affine matrix required to register the scan to the MNI305 space is computed at this step. You can/should check the talairach registration using `tkregister2 -s subjid -fstal`. `tkregister2` allows you to compare the original volume against the talairach volume resampled into the original space. Run `tkregister2 -help` for more information. This step creates the files `mri/transform/talairach.auto.xfm` and `talairach.xfm`.

4. **Intensity Normalization 1** – Renormalization of intensity. Performs intensity normalization of the `orig` volume and places the result in `mri/T1.mgz`. Attempts to correct for fluctuations in intensity that would otherwise make intensity-based segmentation much more difficult. *Intensities for all voxels are scaled so that the mean intensity of the white matter is 110.* If there are problems with the normalization, users can add control points. (See part II for more info)

5. **Skull Strip** – The skull and brainstem are removed from `mri/T1.mgz` and stores the result in `mri/brainmask.auto.mgz` and `mri/brainmask.mgz`. Runs the `mri_watershed` program. *If the strip fails, users can specify seed points* (`-wsseed`) *or change the threshold* (`-wsthresh, -wsmore, -wsless`). The `-autorecon1` stage ends here.

6. **EM Register (linear volumetric registration)** – Computes transform to align the `mri/nu.mgz` volume to the default GCA atlas found in `$FREESURFER_HOME/average` (see `-gca` flag for more info). Creates the file `mri/transforms/talairach.lta`. The `-autorecon2` stage starts here.

7. **CA Intensity Normalization** – Further normalization, based on GCA model. Creates `mri/norm.mgz`.

8. **CA Non-linear Volumetric Registration** –Computes a nonlinear transform to align with GCA atlas. Creates the file `mri/transform/talairach.m3z`.

9. **Remove neck** – The neck region is removed from the NU-corrected volume `mri/nu.mgz`. Makes use of transform computed from prior CA Register stage. Creates the file `mri/nu_noneck.mgz`.

10. **EM Register, with skull** –Computes transform to align volume `mri/nu_noneck.mgz` with GCA volume possessing the skull. Creates `mri/transforms/talairach_with_skull.lta`.

11. **CA Label (Aseg: Volumetric Labeling) and Statistics** – Labels subcortical structures, based in GCA model. Creates the files `mri/aseg.auto.mgz` and `mri/aseg.mgz`.

12. **Intensity Normalization 2 (start here for control points)** – Performs a second (major) intensity correction using only the brain volume as the input (so that it has to be done after the skull strip). Intensity normalization works better when the skull has been removed. Creates a new `brain.mgz volume`. The `-autorecon2-cp` stage begins here. If `-noaseg` flag is used, then `aseg.mgz` is not used by `mri_normalize`.

13. **White matter segmentation** – Attempts to separate white matter from everything else. The input is `mri/brain.mgz`, and the output is `mri/wm.mgz`. Uses intensity, neighborhood, and smoothness constraints. *This is the volume that is edited when manually fixing defects.* Calls `mri_segment, mri_edit_wm_with_aseg, and mri_pretess`. To keep previous edits, run with `-keepwmedits`. If `-noaseg` is used, them `mri_edit_wm_aseg` is skipped.

14. **Edit White Matter with Aseg** – Any user edits to the segmentation is processed here.

15. **Fill/Cut Into Brainstem, and Hemispheres** – This creates the subcortical mass from which the `orig` surface is created. The mid brain is cut from the cerebrum, and the hemispheres are cut from each other. The left hemisphere is binarized to 255. The right hemisphere is binarized to 127. The input is `mri/wm.mgz` and the output is `mri/filled.mgz`. Calls `mri_fill`. If the cut fails, then seed points can be supplied (see `-cc-crs, -pons-crs, -lh-crs, -rh-crs`). The actual points used for the cutting planes in the corpus callosum and pons can be found in `scripts/ponscc.cut.log`. The stage `-autorecon2-wm` begins here. This is the last stage of volumetric processing. If `-noaseg` is used, then `aseg.mgz` is not used by `mri_fill`.

16. **Tessellation (begins per-hemisphere operations)** – This is the step where the `orig` surface (ie, `surf/?h.orig.nofix`) is created. The surface is created by covering the filled hemisphere with triangles. Runs `mri_tessellate`. The places where the points of the triangles meet are called vertices. Creates the file `surf/?h.orig.nofix`. *Note: the topology fixer will create the surface ?h.orig.*

17. **Smooth1** – After tesselation, the `orig` surface is very jagged because each triangle is on the edge of a voxel face and so are at right angles to each other. The vertex positions are adjusted slightly here to reduce the angle. This is only necessary for the inflation processes. Creates `surf/?h.smoothwm(.nofix)`. Calls `mris_smooth`. Smooth1 is the step just after tessellation, and smooth2 is the step just after topology fixing.

18. **Inflate1** – Inflation of the `surf/?h.smoothwm(.nofix)` surface to `create surf/?h.inflated`. The inflation attempts to minimize metric distortion so that distances and areas are preserved (ie, the surface is not stretched). In this sense, it is like inflating a paper bag and not a balloon. Inflate1 is the step just after tessellation, and inflate2 is the step just after topology fixing. Calls `mris_inflate`. Creates `?h.inflated, ?h.sulc, ?h.curv`, and `?h.area`.

19. **QSphere** – This is the initial step of automatic topology fixing. It is a quasi-homeomorphic spherical transformation of the inflated surface designed to localize topological defects for the subsequent automatic topology fixer. Calls `mris_sphere`. Creates `surf/?h.qsphere.nofix`.

20. **Automatic Topology Fixer** – Finds topological defects (ie, holes in a filled hemisphere) using `surf/?h.qsphere.nofix`, and changes the original surface (`surf/?h.orig.nofix`) to remove the defects. Changes the number of vertices. All the defects will be removed, but the user should check the `orig` surface in the volume to make sure that it looks appropriate. Calls `mris_fix_topology`. Creates `surf/?h.orig` (by iteratively fixing `surf/?h.orig.nofix`).

21. **White Matter Surfaces** Creates the `?h.white surface`. The white surface is created by "nudging" the `orig` surface so that it closely follows the white-gray intensity gradient as found in the T1 volume.

22. **Smooth2** Smooth the surfaces again (repeat Smooth 1).

23. **Inflate2** – Inflates the `orig` surface into a sphere while minimizing metric distortion. This step is necessary in order to register the surface to the spherical atlas. (also known as the spherical morph). Calls `mris_sphere`. Creates `surf/?h.sphere`. The -autorecon3 stage begins here.

24. **Spherical Registration** Registers the `orig` surface to the spherical atlas through `surf/?h.sphere`. The surfaces are first coarsely registered by aligning the large scale folding patterns found in `?h.sulc` and then fine tuned using the small-scale patterns as in `?h.curv`. Calls `mris_register`. Creates `surf/?h.sphere.reg`.

25. **Spherical Registration, Contralateral hemisphere** Same as ipsilateral but registers to the contralateral atlas. Creates `lh.rh.sphere.reg` and `rh.lh.sphere.reg`.

26. **Spherical Mapping**. Maps sphere to atlas labels.

27. **Map average curvature to subject** – Resamples the average curvature from the atlas to that of the subject. Allows the user to display activity on the surface of an individual with the folding pattern (ie, anatomy) of a group. Calls `mrisp_paint`. Creates `surf/?h.avg_curv`.

28. **Cortical Parcellation (Labeling)** – Assigns a neuroanatomical label to each location on the cortical surface. Incorporates both geometric information derived from the cortical model (sulcus and curvature), and neuroanatomical convention. Calls `mris_ca_label`. `-cortparc` creates `label/?h.aparc.annot`, and `-cortparc2` creates `/label/?h.aparc.a2005s.annot`.

29. **Cortical Parcellation Statistics** – Runs `mris_anatomical_stats` to create a summary table of cortical parcellation statistics for each structure, including 1. structure name 2. number of vertices 3. total surface area (mm2) 4. total gray matter volume (mm3) 5. average cortical thickness (mm) 6. standard error of cortical thicknessr (mm) 7. integrated rectified mean curvature 8. integrated rectified Gaussian curvature 9. folding index 10. intrinsic curvature index. For

-parcstats, the file is saved in stats/?h.aparc.stats. For -parcstats2, the file is saved in stats/?h.aparc.a2005s.stats.

30. **Pial Surfs** Creates the thickness file (?h.thickness) and curvature file (?h.curv). The pial surface is created by expanding the white surface so that it closely follows the gray-CSF intensity gradient as found in the T1 volume. Calls mris_make_surfaces.

31. **WM/GM Contrast** – ? No information available on this step

32. **Cortical Ribbon Mask** – Creates binary volume masks of the cortical ribbon, ie, each voxel is either a 1 or 0 depending upon whether it falls in the ribbon or not. Saved as ?h.ribbon.mgz. Uses mgz regardless of whether the -mgz option is used. The -autorecon2 stage ends here.

33. **Cortical Parcellation mapped to ASeg** – Maps the cortical parcellation to the subcortical segmentation.

34. **Brodmann and ex vivo EC labels** – Generate labels to map Brodmann areas and Entorhinal cortex.

Appendices

A. Command Line Cheat Sheet

cd – change directory.

```
cd ~ #move into home directory
cd . #stay in current directory
cd ../ #move up one directory in the tree
```

ls – List directory contents. Useful flags :

- **-a** list all directory contents (including hidden files)
- **-l** long form, show extra information
- **-d** show only directories, useful for getting lists to iterate over
- **-G** output with colors
- **-S** sort files by size
- **-s** display number of file system blocks used by each file in units of 512 bytes

```
ls -alGS ~/Projects/ADS/data/Subjects
```

mv – Move or rename a file/directory. Useful flags :

- **-f** overwrite files without asking permission
- **-i** return an error if attempting to overwrite existing file
- **-n** do not overwrite an existing file
- **-v** verbose output, shows files after they are moved

```
mv -nv ~/Projects/ADS/data/Subjects/149959 ~/Projects/ADS/data/BadSubjects/149959
```

cp – Copy contents of source file/directory to a target file/directory. Useful flags :

- **-i** return an error if attempting to overwrite existing file
- **-n** do not overwrite an existing file
- **-v** verbose output, shows files after they are moved
- **-R** copy a source directory and all of it's subdirectories. If the source file ends in a /, the contents of the directory are copied rather than the directory itself.

```
cp -ivR ~/Projects/ADS/data/Subjects/149959 ~/Projects/ADS/data/Subjects.BackUp/149959
```

rm – Delete a file or directory. Useful flags :

- **-f** overwrite files without asking permission

-i return an error if attempting to overwrite existing file

-d delete directories

-R/-r delete directories and all of it's sub directories (use this instead of -d)

-v verbose output, shows files after they are moved

```
rm -rv  ~/Projects/ADS/data/BadSubjects/149959
```

sudo – Use before a command to over come permissions. USE WITH CAUTION. THIS WILL BREAK YOUR SYSTEM IF YOU ARE NOT CAREFUL TO TRIPLE-CHECK WHAT YOU ARE MOVING/DELETING.

```
sudo rm -rv  ~/Projects/ADS/data/BadSubjects/149959
```

cat – concatenate and print files. This is a very useful command for viewing the contents of any file.

```
cat ~/.bash_profile #print your .bash_profile to the screen
cat ~/file1 > ~/file2 #overwrite file 2 with file 1
cat ~/file1 >> ~/file2 #append file 1 to file 2
```

echo – print a variable or a string to the terminal. Useful flags :

-e allows string formatting, i.e.) \n gives a newline

Simple usage:

```
echo "Hello World"
```

Define a variable and print it. Note that you have to use $ to refer to the variable after you define it.

```
subjid="149959"
echo $subjid
echo "The subject ID is " $149959
```

Advanced usage. Get a list of subjects by using `ls` and string formatting to print each subject name on a new line. You must use parentheses around the output of a command with a leading $ to store it in a variable. This is very useful if used properly.

```
subjects=$(ls -d ~/Projects/ADS/data/Subjects/*)
echo -e "These are all the subjects in our study with data\n"
echo -e "\n\t\%s" $subjects
```

touch – create a new file

```
touch .cool_script
newcommand="freeview -v $SUBJECTS_DIR/149959/mri/T1.mgz"
echo $newcommand >> .cool_scirpt
. .cool_script
```

date – get the date. You can use this to measure elapsed time of a script.

```
begin=$(date -u "+%s")
sleep 10 #sleep 10s, or run script here
end=$(date -u "+%s")
elapsed=$((END-START)) #double parenthesis to do arithmetic
echo "$(($elapsed %60)) seconds elapsed"
```

B. Useful Hints

How can I overlay my data on a study average for analysis? Qdec needs each subject to have pre-computed smoothed data for the target surface (fsaverage is the default) for each measure (thickness, sulc, area, curv, etc.). Your SUBJECTS_DIR should contain either a link or a copy of the 'fsaverage' subject found in your $FREESURFER_HOME/subjects directory. Presmoothing the data onto the target surface is not part of the normal recon processing stream, but you can easily create this data with recon-all, using the command:

```
recon-all -s <subjid> -qcache
```

How do I make a study specific average? Use `make_average_subject`. See details here.

How can I easily print all of the results of my Freesurfer segmentation? Try the `FSQC-makehtml` program. It will create images of all of your subjects and then make an html file you can view in a browser.

How to run a command in the background? You may notice that if you run a command in a terminal window that the terminal becomes non-responsive to new commands. A command can be run in the background by adding an ampersand at the end. For example,

```
freeview &
```

You'll notice that the terminal is still available for commands, however that the output of the program are still dumped to the screen. To redirect the output of a program while running it into the background you can type the following:

```
recon-all-go -rd ./raw -g w1 > recon.log &
```

This command will print all of the output of the program `recon-all-go` to the log file, `recon.log`. This is very useful if you would like to save the output of a program for later inspection.

How to run a program after logging out? Use the `nohup` command to run a script or program in the background that is disconnected from the terminal. `nohup` allows the system to take over the job and continues running the program even after you log out. This solution is perfect for systems plagued by power failures or constant network connectivity issues.

```
nohup recon-all-go -rd ./raw -g wl > recon.log &
```

Can I script Freeview to look at a bunch of subjects iteratively? Sure you can! In fact you can do lots of cool stuff with Freeview such as taking automated screenshots, rendering 3D brains, etc. See the help section for Freeview to learn more. An example of a script for viewing subjects iteratively appears below. This presupposes you have a list of subjects `wave1.subjects.artifact` that you iterate through. You can easily create one by hand. Note that it must be a single column list.

```
#! /bin/bash
# This script loads a list of subjects with artifacts and then launches each subject for
    viewing surfaces and volumes
subwave=wave1
subs=$(cat wave1.subjects.artifact)
echo '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * '
echo 'View all subjects with artifacts'
echo '* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * '
echo 'The following Subjects will be loaded: ' $subwave
$subs
for k in $subs
do
# View Volumes
echo -------------------------------------------
echo 'Running Subject: ' $k
freeview -v \
$k/mri/T1.mgz \
$k/mri/wm.mgz \
$k/mri/brainmask.mgz \
$k/mri/aseg.mgz:colormap=lut:opacity=0.2 \
-f $k/surf/lh.white:edgecolor=blue \
$k/surf/lh.pial:edgecolor=red \
$k/surf/rh.white:edgecolor=blue \
$k/surf/rh.pial:edgecolor=red
echo -------------------------------------------
# View Surfaces
freeview -f $k/surf/lh.pial:annot=aparc.annot:name=pial_aparc:visible=0 \
$k/surf/lh.inflated:overlay=lh.thickness:overlay_threshold=0.1,3::name=inflated_thickness:
    visible=0 \
$k/surf/lh.inflated:visible=0 \
$k/surf/lh.white:visible=0 \
$k/surf/lh.pial \
--viewport 3d
freeview -f $k/surf/rh.pial:annot=aparc.annot:name=pial_aparc:visible=0 \
$k/surf/rh.inflated:overlay=rh.thickness:overlay_threshold=0.1,3::name=inflated_thickness:
    visible=0 \
$k/surf/rh.inflated:visible=0 \
$k/surf/rh.white:visible=0 \
$k/surf/rh.pial \
--viewport 3d
done
```

C. Guides & Tutorials

- Excellent UNIX tutorial
- Inspection of Output
- Group Analysis Tutorial
- Multiple Comparisons Tutorial

- Multiple Comparisons Continued
- ROI Analysis

D. Additional Resources

- Stack Overflow is a great resource to ask programming questions if you get stuck.

References

[1] A. M. Dale and M. I. Sereno, *Improved localization of cortical activity by combining EEG and MEG with MRI cortical surface reconstruction: A linear approach.*, Journal of Cognitive Neuroscience, 5 (1993), pp. 162–176.

[2] R. S. Desikan, F. Segonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, M. S. Albert, and R. J. Killiany, *An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest*, Neuroimage, 31 (2006), pp. 968–80.

[3] B. C. Dickerson, E. Fenstermacher, D. H. Salat, D. A. Wolk, R. P. Maguire, R. Desikan, J. Pacheco, B. T. Quinn, A. Van der Kouwe, D. N. Greve, D. Blacker, M. S. Albert, R. J. Killiany, and B. Fischl, *Detection of cortical thickness correlates of cognitive performance: Reliability across MRI scan sessions, scanners, and field strengths*, Neuroimage, 39 (2008), pp. 10–8.

[4] B. Fischl and A. M. Dale, *Measuring the thickness of the human cerebral cortex from magnetic resonance images*, Proc Natl Acad Sci U S A, 97 (2000), pp. 11050–5.

[5] B. Fischl, A. Liu, and A. M. Dale, *Automated manifold surgery: constructing geometrically accurate and topologically correct models of the human cerebral cortex*, IEEE Trans Med Imaging, 20 (2001), pp. 70–80.

[6] B. Fischl, D. H. Salat, E. Busa, M. Albert, M. Dieterich, C. Haselgrove, A. van der Kouwe, R. Killiany, D. Kennedy, S. Klaveness, A. Montillo, N. Makris, B. Rosen, and A. M. Dale, *Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain*, Neuron, 33 (2002), pp. 341–55.

[7] B. Fischl, D. H. Salat, A. J. van der Kouwe, N. Makris, F. Segonne, B. T. Quinn, and A. M. Dale, *Sequence-independent segmentation of magnetic resonance images*, Neuroimage, 23 Suppl 1 (2004), pp. S69–84.

[8] X. Han, J. Jovicich, D. Salat, A. van der Kouwe, B. Quinn, S. Czanner, E. Busa, J. Pacheco, M. Albert, R. Killiany, P. Maguire, D. Rosas, N. Makris, A. Dale, B. Dickerson, and B. Fischl, *Reliability of MRI-derived measurements of human cerebral cortical thickness: the effects of field strength, scanner upgrade and manufacturer*, Neuroimage, 32 (2006), pp. 180–94.

[9] R. Pienaar, B. Fischl, V. Caviness, N. Makris, and P. E. Grant, *A methodology for analyzing curvature in the developing brain from preterm to adult*, Int J Imaging Syst Technol, 18 (2008), pp. 42–68.

[10] G. Postelnicu, L. Zollei, and B. Fischl, *Combined volumetric and surface registration*, IEEE Trans Med Imaging, 28 (2009), pp. 508–22.

[11] M. Reuter, N. J. Schmansky, H. D. Rosas, and B. Fischl, *Within-subject template estimation for unbiased longitudinal image analysis*, Neuroimage, 61 (2012), pp. 1402–18.

[12] M. Reuter, M. D. Tisdall, A. Qureshi, R. L. Buckner, A. J. W. van der Kouwe, and B. Fischl, *Head motion during mri acquisition reduces gray matter volume and thickness estimates*, NeuroImage, 107 (2014), pp. 107–115.

[13] H. D. Rosas, A. K. Liu, S. Hersch, M. Glessner, R. J. Ferrante, D. H. Salat, A. van der Kouwe, B. G. Jenkins, A. M. Dale, and B. Fischl, *Regional and progressive thinning of the cortical ribbon in huntington's disease*, Neurology, 58 (2002), pp. 695–701.

[14] M. R. Sabuncu, B. T. Yeo, K. Van Leemput, B. Fischl, and P. Golland, *A generative model for image segmentation based on label fusion*, IEEE Trans Med Imaging, 29 (2010), pp. 1714–29.

[15] F. Segonne, A. M. Dale, E. Busa, M. Glessner, D. Salat, H. K. Hahn, and B. Fischl, *A hybrid approach to the skull stripping problem in MRI*, Neuroimage, 22 (2004), pp. 1060–75.

[16] K. Van Leemput, A. Bakkour, T. Benner, G. Wiggins, L. L. Wald, J. Augustinack, B. C. Dickerson, P. Golland, and B. Fischl, *Automated segmentation of hippocampal subfields from ultra-high resolution in vivo MRI*, Hippocampus, 19 (2009), pp. 549–57.

[17] C. von Economo, *he Cytoarchitectonics of the Human Cerebral Cortex.*, Oxford University Press„ 1929.

[18] A. Yendiki, P. Panneck, P. Srinivasan, A. Stevens, L. Zollei, J. Augustinack, R. Wang, D. Salat, S. Ehrlich, T. Behrens, S. Jbabdi, R. Gollub, and B. Fischl, *Automated probabilistic reconstruction of white-matter pathways in health and disease using an atlas of the underlying anatomy*, Front Neuroinform, 5 (2011), p. 23.