seam Documentation

Release 0.0

Scott Burns

April 08, 2014

Contents

1 Contents **3**
- 1.1 Installation . . . 3
- 1.2 Philosophy . . . 3
- 1.3 Contributing . . . 4
- 1.4 Tools . . . 4

2 Support **9**

3 License **11**

Python Module Index **13**

Seam is a simple layer between neuroimaging tools and your data. It makes no decisions about how to organize your data or execute data analyses. It simply provides commands to intelligently call standard neuroimaging tools.

CHAPTER 1

Contents

1.1 Installation

Install the latest version with pip:

```
$ pip install seam
```

Seam is also distributed in the wheel format, so you can install it as a wheel:

```
$ pip install --use-wheel seam
```

Note, this requires the wheel package along with an up-to-date pip.

To install the bleeding edge from the github repo, use the following:

```
$ pip install -e git+https://github.com/VUIIS/seam.git#egg=seam
```

Either way, it has no dependencies.

1.2 Philosophy

Software is best written in layers. Each layer should encapsulate knowledge about how to best use the next lower layer. Its functionality should be exposed through as simple an API as possible.

Seam extends this rationale to neuroimaging data analysis. It is up to you to organize your data and run analyses, but seam will generate commands for you to apply your data against standard neuroimaging tools.

Simply put, it builds commands around common neuroimaging tools. You're free to do with these commands whatever you wish.

Importantly, seam is oblivious to these two otherwise-important factors:

- Data organization
- Command execution

Seam has no dependencies and requires minimal effort to use it. Seam can easily integrate into any application ranging from a single script to something much more complicated.

1.2.1 Recipes vs Functions

Where possible, functionality provided by seam is organized in two layers:

- Functions: low-level, highly configurable functions to produce commands or scripts.
- Recipes: high-level, less configurable functions to produce complete workflows around a tool.

Recipes will be exposed as both binaries that can be executed in a shell and as importable functions in python code. Users should first try to use recipes to accomplish their goals as these expose best practices around the tool.

If more customization is needed, the low-level functions are also available. Understandably, more effort is required to piece together the low-level functions into a meaningful processing stream.

1.2.2 Backwards Compatiblity

Seam makes every effort to ensure backward compatibility for the commands it generates. Because neuroimaging projects can last for very long periods, the commands generated by seam will be versioned so you can upgrade seam and still produce the same commands for a given project. Newer projects, though, can use newer versions of commands using the same seam.

1.3 Contributing

If you would like to contribute commands or find errors and/or better ways to build commands, please consider contributing to seam. To do so, please use the following workflow:

- Fork the repository to your own account.
- Checkout an aptly-named branch and commit your changes.
- Please add tests (and documentation) and make sure they pass. You can use `$ make test` to run the suite.
- Push your commits to your fork and submit a pull-request.

1.4 Tools

1.4.1 DTI_QA

This interface assists in executing Bennett Landman's DTI_QA toolkit.

TBD: More documentation here.

Because DTI_QA is built upon matlab, the generated commands are m-code.

Functions

seam.dti_qa.**dtiqa_mcode**(*images*, *basedir*, *dtiqa_path*, *n_b0=1*)

Returns m-code that can be executed in matlab to run DTI_QA

Parameters

- **images** (*str,list*) – path to single raw DTI image or list of multiple
- **basedir** (*str*) – base directory to write results
- **dtiqa_path** (*str*) – path to DTI_QA installation
- **n_b0** (*int*) – # of b0 averages (default is 1)

Usage:

```
>>> from seam.dti_qa.v1 import dtiqa_mcode
>>> images = ['/path/to/first.nii', '/path/to/second.nii']
>>> basedir, n_b0, dtiqa_path = '/path/to/output', 6, '/path/to/dtiqa'
>>> f = open('dti_qa.m', 'w')
>>> f.write(dtiqa_mcode(images, basedir, dtiqa_path, n_b0))
>>> f.close()
```

Versioning:

- V1 defines the following:
  - `seam.dti_qa.v1.dtiqa_mcode()` that generates m-code to run DTI_QA.

1.4.2 Freesurfer

Freesurfer is a set of software tools for the study of cortical and subcortical anatomy. In the cortical surface stream, the tools construct models of the boundary between white matter and cortical gray matter as well as the pial surface. Once these surfaces are known, an array of anatomical measures becomes possible, including: cortical thickness, surface area, curvature, and surface normal at each point on the cortex. The surfaces can be inflated and/or flattened for improved visualization. The surfaces can also be used to constrain the solutions to inverse optical, EEG and MEG problems. In addition, a cortical surface-based atlas has been defined based on average folding patterns mapped to a sphere. Surfaces from individuals can be aligned with this atlas with a high-dimensional nonlinear registration algorithm. The registration is based on aligning the cortical folding patterns and so directly aligns the anatomy instead of image intensities. The spherical atlas naturally forms a coordinate system in which point-to-point correspondence between subjects can be achieved. This coordinate system can then be used to create group maps (similar to how Talairach space is used for volumetric measurements).

Most of the FreeSurfer pipeline is automated, which makes it ideal for use on large data sets.

Recipes

`seam.freesurfer.v1.recipe.`**`build_recipe`**(*subject_id*, *input_data*, *script_dir*, *use_xvfb=False*, *recon_flags=None*)

This function builds a complete pipeline around Freesufer.

It does the following:

•Imports data using `recon-all -i`

•Runs the main `recon-all` command with the following flags:

- `-qcache`
- `-measure thickness`
- `-measure curv`
- `-measure sulc`
- `-measure area`
- `-measure jacobian_white`

•Takes volumetric snapshots using `tkmedit`

•Per hemisphere:

- Converts the aparc.a2009s.annot file to labels in the subject's `label` directory
- Takes screenshots of the inflated surface with and without the advanced labels.

Parameters

- **subject_id** (*str*) – subject identifier
- **input_data** (*str,list*) – list of paths or string to subject's T1 images
- **script_dir** (*str*) – directory to write scripts & screenshots
- **use_xvfb** (*boolean*) – Wrap `tksurfer` & `tkmedit` commands in xvfb-run, useful if running in a non-graphical (ie cluster) environment.
- **recon_flags** (*list*) – other flags to pass to `recon-all`

Return type tuple

Returns paths to recon script, tkmedit script and lh & rh tksurfer scripts

Note the main script is set as executable

Note This function is exposed on the command line through `build-recon-v1`

Functions

These are lower-level functions to be used when more customization is needed than provided by the above recipes.

`seam.freesurfer.v1.core.`**`recon_input`**(*subject_id*, *data*)

The function supplies the `recon-all -i` command. This command initializes the Freesurfer directory for a subject and converts the raw data into an internal format for use by the rest of the `recon-all` pipeline.‘

Parameters

- **subject_id** (*str*) – Subject identifier
- **data** (*str,list*) – path(s) to input data

Returns command that will execute `recon-all -i` command

Return type str

Usage:

```
>>> from seam.freesurfer import recon_input
>>> recon_input('sub0001', '/full/path/to/data.nii')
'recon-all -s sub0001 -i /full/path/to/data.nii'
>>> # or with multiple inputs...
>>> recon_input('sub0001', ['/path/first.nii', '/path/second.nii'])
'recon-all -s sub0001 -i /path/first.nii -i /path/second.nii'
```

`seam.freesurfer.v1.core.`**`recon_all`**(*subject_id*, *flags=None*)

This function supplies the `recon-all -all` command. This command will run the entire anatomical analysis suite of Freesurfer.

Note Use `seam.freesurfer.recon_input()` to setup this subject

Parameters

- **subject_id** (*str*) – Subject identifier on which to run `recon-all`
- **flags** (*list*) – command-line flags to pass to `recon-all`

Returns command that will execute `recon-all -all`

Return type str

Usage:

```
>>> from seam.freesurfer import recon_all
>>> recon_all('sub0001', flags=['-use-gpu'])
'recon-all -s sub0001 -all -qcache -measure thickness -measure curv -measure sulc -measure area
```

seam.freesurfer.v1.core.**tkmedit_screenshot_tcl**(*basepath*, *beg=5*, *end=256*, *step=10*)

Supplies a tcl string that can be used to take screenshots of a volume using tkmedit

Images are written to `*basepath*/tkmedit-$i.tiff` where *beg* <= i <= *end* in increments of *step*

Parameters

- **basepath** – base directory to write images in
- **beg** (*int*) – Beginning slice where screenshots begin
- **end** (*int*) – End slice where screenshots end
- **step** (*int*) – Value to increment successive screenshots

Usage:

```
>>> from seam.freesurfer import tkmedit_screenshot_tcl
>>> f = open('tmedit_screenshots.tcl', 'w')
>>> f.write(tkmedit_screenshot_tcl('/path/to/image_dir'))
>>> f.close()
$ tkmedit sub0001 brain.finalsurfs.mgz -aseg -surfs -tcl tkmedit_screenshots.tcl
```

seam.freesurfer.v1.core.**tkmedit_screenshot_cmd**(*subject_id*, *volume*, *tcl_path*, *flags=None*)

Supplies a command to execute a tcl script in `tkmedit` for *subject_id*'s *volume*

Parameters

- **subject_id** (*str*) – subject identifier
- **volume** (*str*) – Volume for tkmedit to load
- **tcl_path** (*str*) – Path to tcl script
- **flags** (*list*) – Flags to pass to `tkmedit`

Usage:

```
>>> from seam.freesurfer import tkmedit_screenshot_cmd
>>> tkmedit_screenshot_cmd('sub0001', 'brain.finalsurfs.mgz', '/path/tkmedit.tcl', ['-aseg', '-s
'tkmedit sub0001 brain.finalsurfs.mgz -aseg -surfs -tcl /path/tkmedit.tcl'
```

seam.freesurfer.v1.core.**tksurfer_screenshot_tcl**(*basepath*, *annot='aparc.a2009s.annot'*)

Supplies a tcl command to take screenshots of a surface using tksurfer

Four screenshots are taken:

•lateral view (default view when `tksurfer` opens) saved to *basepath*-lateral.tiff

•medial view saved to *basepath*-medial.tiff

•Lateral view with an annotation loaded (given by *annot*) saved to *basepath*-annot-lateral.tiff

•Medial view with an annotation loaded (given by *annot*) saved to *basepath*-annot-medial.tiff

Parameters

- **basepath** (*str*) – prefix for images to be saved
- **annot** (*str*) – annotation file to load for overlay on the surface

Usage:

```
>>> from seam.freesurfer import tksurfer_screenshot_tcl
>>> f = open('tksurfer.lh.tcl', 'w')
>>> f.write(tksurfer_screenshot_tcl('/path/to/screenshots/lh'))
>>> f.close()
```

seam.freesurfer.v1.core.**tksurfer_screenshot_cmd**(*subject_id*, *hemi*, *surface*, *tcl_path*, *flags=None*)

Supply a command that will run `tksurfer` using the *surface* from *subject_id*'s *hemi* hemisphere and execute a tcl script.

Parameters

- **subject_id** (*str*) – subject identifier
- **hemi** (*str*) – 'lh' or 'rh', hemisphere to open in tksurfer
- **surface** (*str*) – surface to view
- **tcl_path** (*str*) – path to tcl script to execute
- **flags** (*list*) – flags to pass into `tksurfer`

Usage:

```
>>> from seam.freesurfer import tksurfer_screenshot_cmd
>>> tksurfer_screenshot_cmd('sub0001', 'lh', 'inflated', '/path/tksurfer.lh.tcl', ['-gray'])
'tksurfer sub0001 lh inflated -gray -tcl /path/tksurfer.lh.tcl'
```

seam.freesurfer.v1.core.**annot2label_cmd**(*subject_id*, *hemi*, *annot_path*, *outdir*, *surface='white'*)

Build the mri_annotation2label commandline string.

Parameters

- **subject_id** (*str*) – subject identifier
- **hemi** (*str*) – 'lh' or 'rh', hemisphere to use
- **annot_path** (*str*) – path to annotation file
- **outdir** (*str*) – output directory to place labels
- **surface** (*str*) – surface to use when generating coords in labels

Versions: V1 defines the following recipes:

- `seam.freesurfer.v1.build_recipe()` for building a complete script for executing the recon-all pipeline.

V1 defines the following functions:

- `recon-all -all` exposed through `seam.freesurfer.v1.recon_all()`
- `recon-all -i` exposed through `seam.freesurfer.v1.recon_input()`
- `seam.freesurfer.v1.tkmedit_screenshot_tcl()` for generating tcl to take screenshots of a volume loaded in `tkmedit`.
- `seam.freesurfer.v1.tkmedit_screenshot_cmd()` for supplying a command to execute `tkmedit` with a tcl script.
- `seam.freesurfer.v1.tksurfer_screenshot_tcl()` for generating a tcl script to take screenshots of a hemisphere using `tksurfer`

- `seam.freesurfer.v1.tksurfer_screenshot_cmd()` for supplying a command to run `tksurfer` and generate screenshots.
- `seam.freesufer.v1.annot2label_cmd()` for building a `mri_annotation2label` command.

CHAPTER 2

Support

If you are having problems, please raise an issue on Github. I can't guarantee support but promise to help where I can.

- Issue Tracker: https://github.com/VUIIS/seam/issues
- Source Code: https://github.com/VUIIS/seam

License

The project is licensed under the MIT license.

S

seam.dti_qa, 4
seam.dti_qa.v1, 4
seam.freesurfer.v1, 7