Data Preparation for Neuroimagers: BIDS, mriqc, and fmriprep

Dr. Kyle Kurkela
kkurkela@bu.edu
Research Computing Services
Information Services & Technology
Boston University

Resources

- Tutorial Slides and a Recording of today's tutorial will be made available
- Code and example data are available here:
  - https://rcs.bu.edu/examples/imaging/tut_dataprep_scc/
- Questions? Comments? Concerns?
  - help@scc.bu.edu
  - kkurkela@bu.edu
  - mcmains@bu.edu

Assumptions

- This tutorial is aimed at a beginner level.
- However, I will assume that you have a basically level of comfort within a Linux environment.
- The Research Computing Services team has many resources for learning Linux:
  - A 20 min Introduction to Linux:
    - https://www.bu.edu/tech/support/research/training-consulting/rcs-tutorial-videos-and-third-party-tutorials/intro-linux-20min/
  - A 2 hr Introduction to Linux and the SCC:
    - https://www.bu.edu/tech/support/research/training-consulting/rcs-tutorial-videos-and-third-party-tutorials/intro-scc/
  - Linux on the SCC Cheat Sheet:
    - http://scv.bu.edu/documents/Linux_SCC_CheatSheet.pdf

Learning Objectives

- By the end of this tutorial, you should …

  1. have a basic understanding of the two major file formats for neuroimaging data: DICOM and NIFTI.
  2. have a basic understanding of how neuroimaging data is organized in BIDS.
  3. be able to convert DICOM files to NIFTI files using *dcm2niix*.
  4. be able to convert DICOM files to BIDS formatted NIFTI files using *dcm2bids*.
  5. Have a basic understanding of batch jobs on the SCC
  6. be able to run a quality assurance routine using *MRIQC* on the SCC.
  7. be able to run a preprocessing routine using *FMRIPREP* on the SCC.

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Workshop Overview

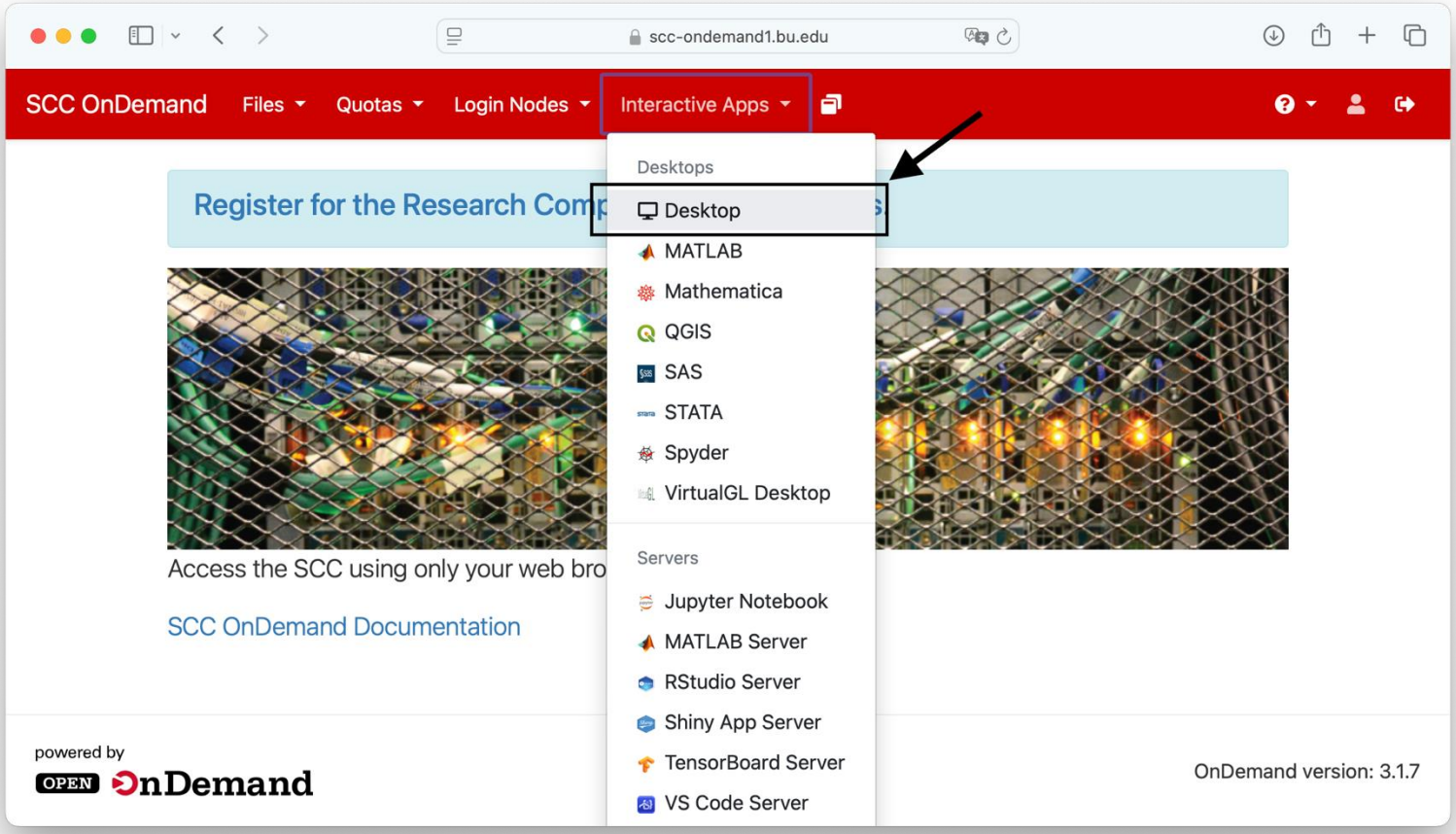
1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Getting an SCC Account

- Today we will be using tutorial accounts. These accounts are temporary and will be inaccessible after today's workshop
- All users of the SCC must be on a Research Project headed by a full-time BU Faculty member with some exceptions:
  - Some academic classes
  - 3 Month Trial Accounts for "trying out" the SCC
- Please see for more detailed information:
  - https://www.bu.edu/tech/support/research/account-management/

Accessing Tutorial Accounts

- Open a web browser
- Go to scc-ondemand-tutorial.bu.edu
- Username: tuta#
- Password: ******
- Click “Interactive Apps”
- Click “Desktop”



Copy the tutorial data

- Open up a terminal window and follow along with me.
- NOTE: all commands used in today's tutorial will be in a publicly accessible text file *notes.txt*
  - https://rcs.bu.edu/examples/imaging/tut_dataprep_scc/notes.txt

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

MRI Data 101 -- DICOM

- Magnetic Resonance Imaging (MRI) data largely stored in DICOM format
- DICOM
  - Standard medical image format used by most human imaging scanners.
  - Very rich meta data.
  - Supports both 2D and 3D images (Some 3D images are stored as a sequence of 2D images).
  - Not very conducive for image processing.
  - Relatively difficult to compress.
  - See for more information:
    - https://www.dicomstandard.org
    - https://dicom.innolitics.com/ciods/mr-image

Explore the DICOM files

- Let's return to SCC OnDemand and explore the data a little bit, inspecting the headers of one of the DICOM files. Follow along with me…
- NOTE: all commands used in today's tutorial will be in a publicly accessible text file *notes.txt*
  - https://rcs.bu.edu/examples/imaging/tut_dataprep_scc/notes.txt

MRI Data 101 -- NIFTI

- Raw DICOM image files are usually converted to one of several different file formats for further processing and analysis.
- NIFTI
  - Most popular – the research standard format
  - Relatively limited meta-data
  - Represents 3D volumes (or 4D – i.e., 3-D volumes over time) in a single file
  - Easily compressed to reduce storage space
  - Examples: sub_001.nii, subject_001.nii.gz
  - Freesurfer, FSL, AFNI, SPM, CONN

DICOMs -> NIFTIs

- Let’s convert the DICOM files we just looked at to NIFTI files and explore them a bit closer. Follow along with me…
- NOTE: all commands used in today’s tutorial will be in a publicly accessible text file *notes.txt*
  - https://rcs.bu.edu/examples/imaging/tut_dataprep_scc/notes.txt

MRI Data Organization

- MRI data is numerous and complex – you need an organization system!
- In the good-old-days, every research group, institution, lab would have their own idiosyncratic organization systems. It was a nightmare.
- BIDS came to the rescue!

The Brain Imaging Data Structure (BIDS)

- BIDS – a universal organization system for brain imaging data
- Organizes data into subfolders for data types nested within sessions nested within subjects with specific meta data files required at different stages of a hierarchy depending on data type
- Please see the BIDS data format manual, associated website, and associated paper for further details
  - **Manual**: https://bids-specification.readthedocs.io/en/stable/
  - **Website**: https://bids.neuroimaging.io/specification.html
  - **Paper**: Gorgolewski, K. J., Auer, T., Calhoun, V. D., Craddock, R. C., Das, S., Duff, E. P., ... & Poldrack, R. A. (2016). The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific data*, *3*(1), 1-9.

The Brain Imaging Data Structure (BIDS)

- What does a BIDS formatted dataset look like? What are the basics?
- Let’s take a look at some publicly available ones on OpenNeuro
- https://openneuro.org

Software Spotlight: *dcm2bids*

- How do I get my data into BIDS format?
- dcm2bids
  - One of a vast array of software packages designed to organize your MRI data into a bids format
  - Relies on the very popular *dcm2nix* software package to convert DICOM → NIFIT AND performs data organization all in one go
  - You need to carefully design a config.json file to sort your scans into the appropriate BIDS datatypes

DICOMs -> NIFTIs -> BIDS

- Let's convert our example DICOM data into a bids formatted dataset. Follow along with me…
- NOTE: all commands used in today's tutorial will be in a publicly accessible text file *notes.txt*
  - https://rcs.bu.edu/examples/imaging/tut_dataprep_scc/notes.txt

Creating a dcm2bids config file

- The trickiest part about using dcm2bids is creating the config file.
- The config file is a json text file that tells dcm2bids how to map header information from the DICOM file to the appropriate BIDS information
  - For example, which DICOM files represent an ANAT image? A BOLD images? A FMAP image?

Does your dataset meet the BIDS guidelines?

- How do you know if your datasets meets BIDS guidelines?
- You can use the *BIDS validator*.
  - Available as a locally installable utility and as an online web browser utility
  - Web Utility: https://bids-standard.github.io/bids-validator/

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Two Types of Jobs on the SCC

Interactive Jobs

- Have a user interface, allow you to point, click, and type to interact with software, data, etc.
- You create them with SCC OnDemand

powered by

OPEN OnDemand

Batch Jobs

- AKA “Non-interactive”
- Scripts that run without your active input.

Batch Job on the SCC – 101

- All batch jobs require a ***batch script*** -- a text file that has all the commands that you want to run during your non-interactive session.
- Batch jobs are submitted by using the **qsub command**

```
#!/bin/bash -l

# Set SCC project
#$ -P my_project

# Specify hard time limit for the job.
#   The job will be aborted if it runs longer than this time.
#   The default time is 12 hours
#$ -l h_rt=12:00:00

# Send an email when the job finishes or if it is aborted (by default no email is sent).
#$ -m ea

# Give job a name
#$ -N example

# Combine output and error files into a single file
#$ -j y

# Keep track of information related to the current job
echo "============================================================="
echo "Start date : $(date)"
echo "Job name : $JOB_NAME"
echo "Job ID : $JOB_ID  $SGE_TASK_ID"
echo "============================================================="

module load python3/3.8.10
python -V
```

Example batch script

- Batch scripts are a type of *bash script* – a text file with commands you would normally run in the terminal.

```
#!/bin/bash -l

# Set SCC project
#$ -P my_project

# Specify hard time limit for the job.
#   The job will be aborted if it runs longer than this time.
#   The default time is 12 hours
#$ -l h_rt=12:00:00

# Send an email when the job finishes or if it is aborted (by default no email is sent).
#$ -m ea

# Give job a name
#$ -N example

# Combine output and error files into a single file
#$ -j y

# Keep track of information related to the current job
echo "============================================================="
echo "Start date : $(date)"
echo "Job name : $JOB_NAME"
echo "Job ID : $JOB_ID  $SGE_TASK_ID"
echo "============================================================="

module load python3/3.8.10
python -V
```

Example batch script

- Bash scripts always start with this **opening.**
- Importantly, batch scripts also need to add the -l flag.
- Let's the computer know that this is bash code.

```
#!/bin/bash -l

# Set SCC project
#$ -P my_project

# Specify hard time limit for the job.
#   The job will be aborted if it runs longer than this time.
#   The default time is 12 hours
#$ -l h_rt=12:00:00

# Send an email when the job finishes or if it is aborted (by default no email is sent).
#$ -m ea

# Give job a name
#$ -N example

# Combine output and error files into a single file
#$ -j y

# Keep track of information related to the current job
echo "============================================================"
echo "Start date : $(date)"
echo "Job name : $JOB_NAME"
echo "Job ID : $JOB_ID  $SGE_TASK_ID"
echo "============================================================"

module load python3/3.8.10
python -V
```

Example batch script

- These are **comments**. Otherwise ignored by the computer.
- Comments always begin with “#”.
- Supposed to help the programmer understand what is going on.

```
#!/bin/bash -l

# Set SCC project
#$ -P my_project

# Specify hard time limit for the job.
#   The job will be aborted if it runs longer than this time.
#   The default time is 12 hours
#$ -l h_rt=12:00:00

# Send an email when the job finishes or if it is aborted (by default no email is sent).
#$ -m ea

# Give job a name
#$ -N example

# Combine output and error files into a single file
#$ -j y

# Keep track of information related to the current job
echo "============================================================="
echo "Start date : $(date)"
echo "Job name : $JOB_NAME"
echo "Job ID : $JOB_ID  $SGE_TASK_ID"
echo "============================================================="

module load python3/3.8.10
python -V
```

Example batch script

- These are **job directives.**
- They are options sent to the job scheduler.

```
#!/bin/bash -l

# Set SCC project
#$ -P my_project

# Specify hard time limit for the job.
#   The job will be aborted if it runs longer than this time.
#   The default time is 12 hours
#$ -l h_rt=12:00:00

# Send an email when the job finishes or if it is aborted (by default no email is sent).
#$ -m ea

# Give job a name
#$ -N example

# Combine output and error files into a single file
#$ -j y

# Keep track of information related to the current job
echo "============================================================="
echo "Start date : $(date)"
echo "Job name : $JOB_NAME"
echo "Job ID : $JOB_ID  $SGE_TASK_ID"
echo "============================================================="

module load python3/3.8.10
python -V
```

Example batch script

- This is the **code that you want to run** on the SCC.
- Typically, you want to load a software package with a module load command and then do something with it.

General Batch Job Directives

- This table lists some of the most common batch job directives
- Many should be straight forward

General Directives	
Directive	Description
-l h_rt=*hh:mm:ss*	Hard run time limit in *hh:mm:ss* format. The default is 12 hours.
-P *project_name*	Project to which this jobs is to be assigned. This directive is **mandatory** for all users associated with any Med.Campus project.
-N *job_name*	Specifies the job name. The default is the script or command name.
-o *outputfile*	File name for the stdout output of the job.
-e *errfile*	File name for the stderr output of the job.
-j y	Merge the error and output stream files into a single file.
-m *b|e|a|s|n*	Controls when the batch system sends email to you. The possible values are – when the job begins (b), ends (e), is aborted (a), is suspended (s), or never (n) – default.
-M *user_email*	Overwrites the default email address used to send the job report.
-V	All current environment variables should be exported to the batch job.
-v *env=value*	Set the runtime environment variable *env* to *value*.
-hold_jid *job_list*	Setup job dependency list. *job_list* is a comma separated list of job ids and/or job names which must complete before this job can run. See Advanced Batch System Usage for more information.

Resource Batch Job Directives

- This table lists batch job directives related to computational resources.
- Tells the job scheduled what type of computer you want
- For example, one with lots of memory? One with lots of cores? One with a gpu? A particular cpu chip?

Directives to request SCC resources	
Directive	**Description**
-l h_rt=*hh:mm:ss*	Hard run time limit in *hh:mm:ss* format. The default is 12 hours.
-l mem_per_core=*#G*	Request a node that has at least this amount of memory per core. Recommended choices are: 3G, 4G, 6G, 8G, 12G, 16G, 18G and 28G
-pe omp *N*	Request multiple slots for Shared Memory applications (OpenMP, pthread). This option can also be used to reserve a larger amount of memory for the application. *N* can vary. Currently, to request multiple cores on SCC's shared nodes, we recommend selecting 1-4, 8, 16, 28, or 36 cores.
-pe mpi_#_tasks_per_node *N*	Select multiple nodes for an MPI job. Number of tasks can be 4, 8, 12, 16, or 28 and *N* must be a multiple of this value. See Running Parallel Batch Jobs for more information.
-t *N*	Submit an Array Job with *N* tasks. N can be up to 75,000. For more information see Array Jobs
-l cpu_arch=*ARCH*	Select a processor architecture (broadwell, ivybridge, cascadelake…). See Technical Summary for all available choices.
-l cpu_type=*TYPE*	Select a processor type (X5670, X5675, Gold-6132 etc.) See Technical Summary for all available choices.
-l gpus=*G*	Requests a node with GPUs. *G* is the number of GPUs. See GPU Computing for more information.
-l gpu_type=*GPUMODEL*	To see the current list of available GPU models, run *qgpus* command. See GPU Computing for more information.
-l gpu_c=*CAPABILITY*	Specify minimum GPU capability. Current choices for *CAPABILITY* are 3.5, 5.0, 6.0, 7.0, and 8.6
-l gpu_memory=#G	Request a node with a GPU that has 12G, 16G, 24G, 32G, or 48G of memory.
-l avx	Request a node that supports AVX and newer CPU instructions. A small number of modules require support for these instructions.

Resource Batch Job Directives

- The most important ones for our purposes will be **mem_per_core, pe omp**

Directives to request SCC resources	
Directive	Description
-l h_rt=*hh:mm:ss*	Hard run time limit in *hh:mm:ss* format. The default is 12 hours.
-l mem_per_core=*#G*	Request a node that has at least this amount of memory per core. Recommended choices are: 3G, 4G, 6G, 8G, 12G, 16G, 18G and 28G
-pe omp *N*	Request multiple slots for Shared Memory applications (OpenMP, pthread). This option can also be used to reserve a larger amount of memory for the application. *N* can vary. Currently, to request multiple cores on SCC's shared nodes, we recommend selecting 1-4, 8, 16, 28, or 36 cores.
-pe mpi_#_tasks_per_node *N*	Select multiple nodes for an MPI job. Number of tasks can be 4, 8, 12, 16, or 28 and *N* must be a multiple of this value. See Running Parallel Batch Jobs for more information.
-t *N*	Submit an Array Job with *N* tasks. N can be up to 75,000. For more information see Array Jobs
-l cpu_arch=*ARCH*	Select a processor architecture (broadwell, ivybridge, cascadelake…). See Technical Summary for all available choices.
-l cpu_type=*TYPE*	Select a processor type (X5670, X5675, Gold-6132 etc.) See Technical Summary for all available choices.
-l gpus=*G*	Requests a node with GPUs. *G* is the number of GPUs. See GPU Computing for more information.
-l gpu_type=*GPUMODEL*	To see the current list of available GPU models, run *qgpus* command. See GPU Computing for more information.
-l gpu_c=*CAPABILITY*	Specify minimum GPU capability. Current choices for *CAPABILITY* are 3.5, 5.0, 6.0, 7.0, and 8.6
-l gpu_memory=#G	Request a node with a GPU that has 12G, 16G, 24G, 32G, or 48G of memory.
-l avx	Request a node that supports AVX and newer CPU instructions. A small number of modules require support for these instructions.

Environmental Variables

- Not specific to batch jobs, **environmental variables** are variables that are defined within a terminal that often change how programs function.
- The table to the right lists the environmental variables that the batch job system defines for us

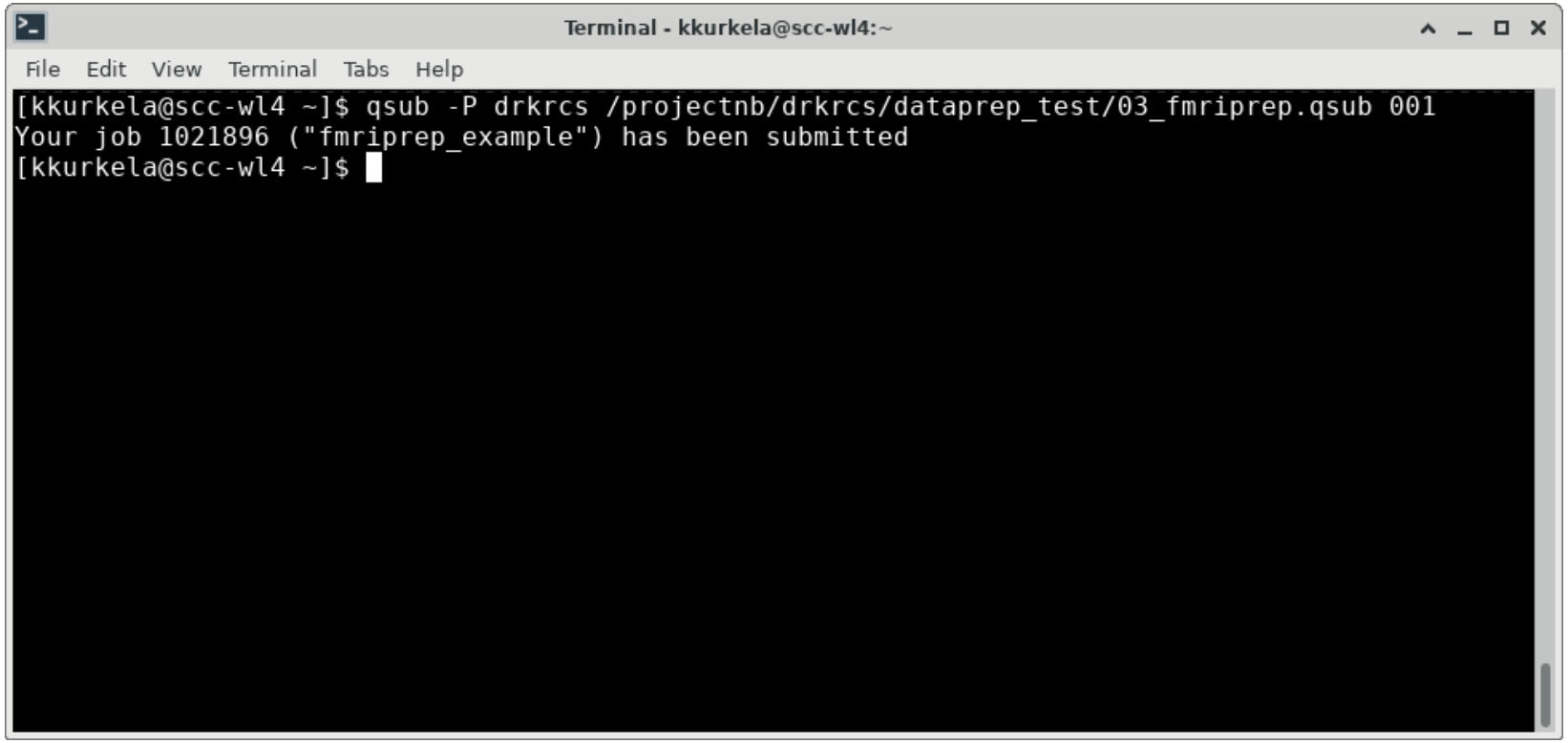
Batch System Environment	
Environment Variable	Description
JOB_ID	Current job ID
JOB_NAME	Current job name
NSLOTS	The number of slots requested by a job
HOSTNAME	Name of execution host
SGE_TASK_ID	Array Job task index number
SGE_TASK_STEPSIZE	The step size of the array job specification
SGE_TASK_FIRST	The index number of the first array job task
SGE_TASK_LAST	The index number of the last array job task
TMPDIR	The absolute path to the job's temporary working directory

Environmental Variables

- Useful ones for us will be the **NSLOTS** and **TMPDIR** variables.

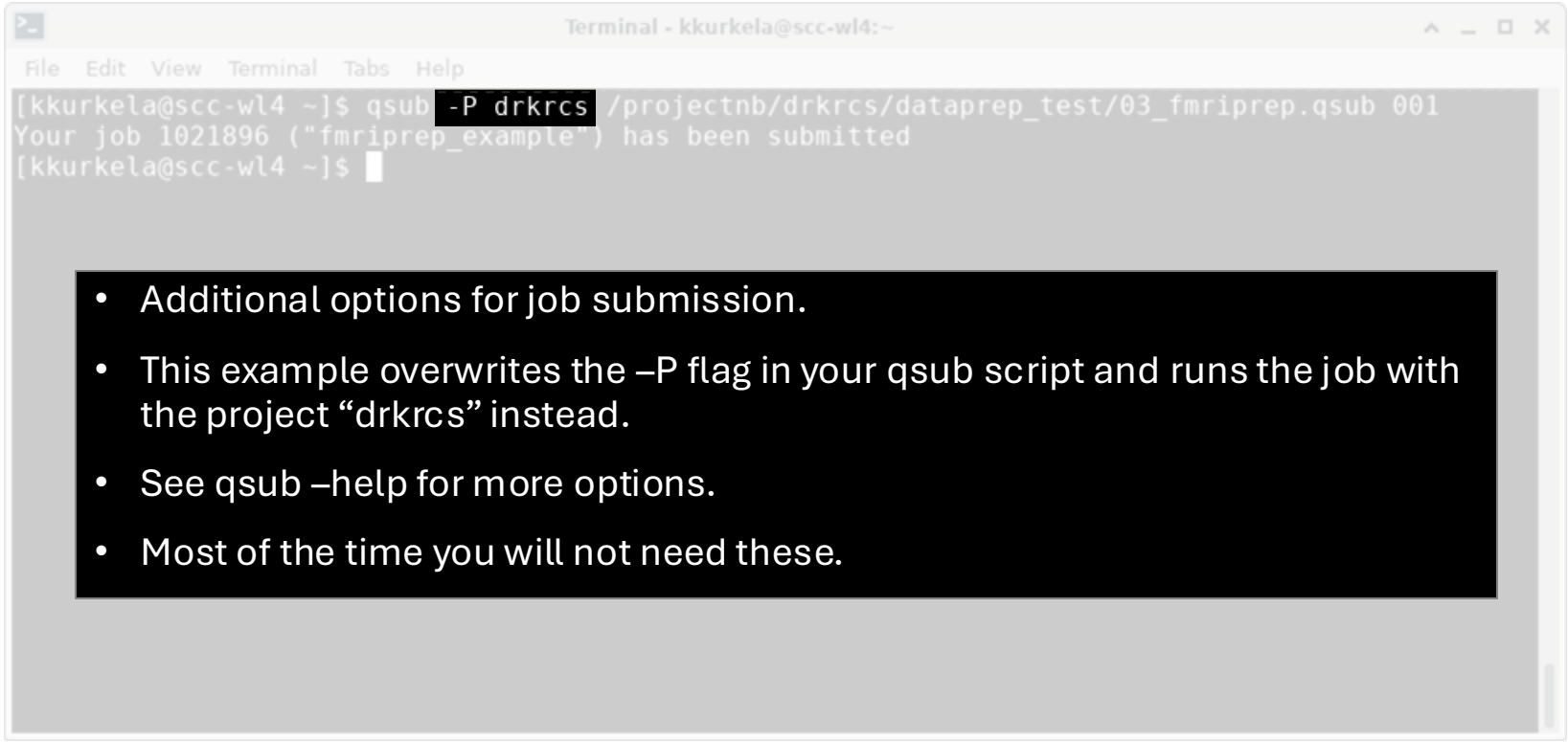
Batch System Environment	
Environment Variable	Description
JOB_ID	Current job ID
JOB_NAME	Current job name
NSLOTS	The number of slots requested by a job
HOSTNAME	Name of execution host
SGE_TASK_ID	Array Job task index number
SGE_TASK_STEPSIZE	The step size of the array job specification
SGE_TASK_FIRST	The index number of the first array job task
SGE_TASK_LAST	The index number of the last array job task
TMPDIR	The absolute path to the job's temporary working directory

SCC Batch Job on the SCC – qsub



qsub [options] [batch_script.qsub] [arguments sent to batch_script.qsub]

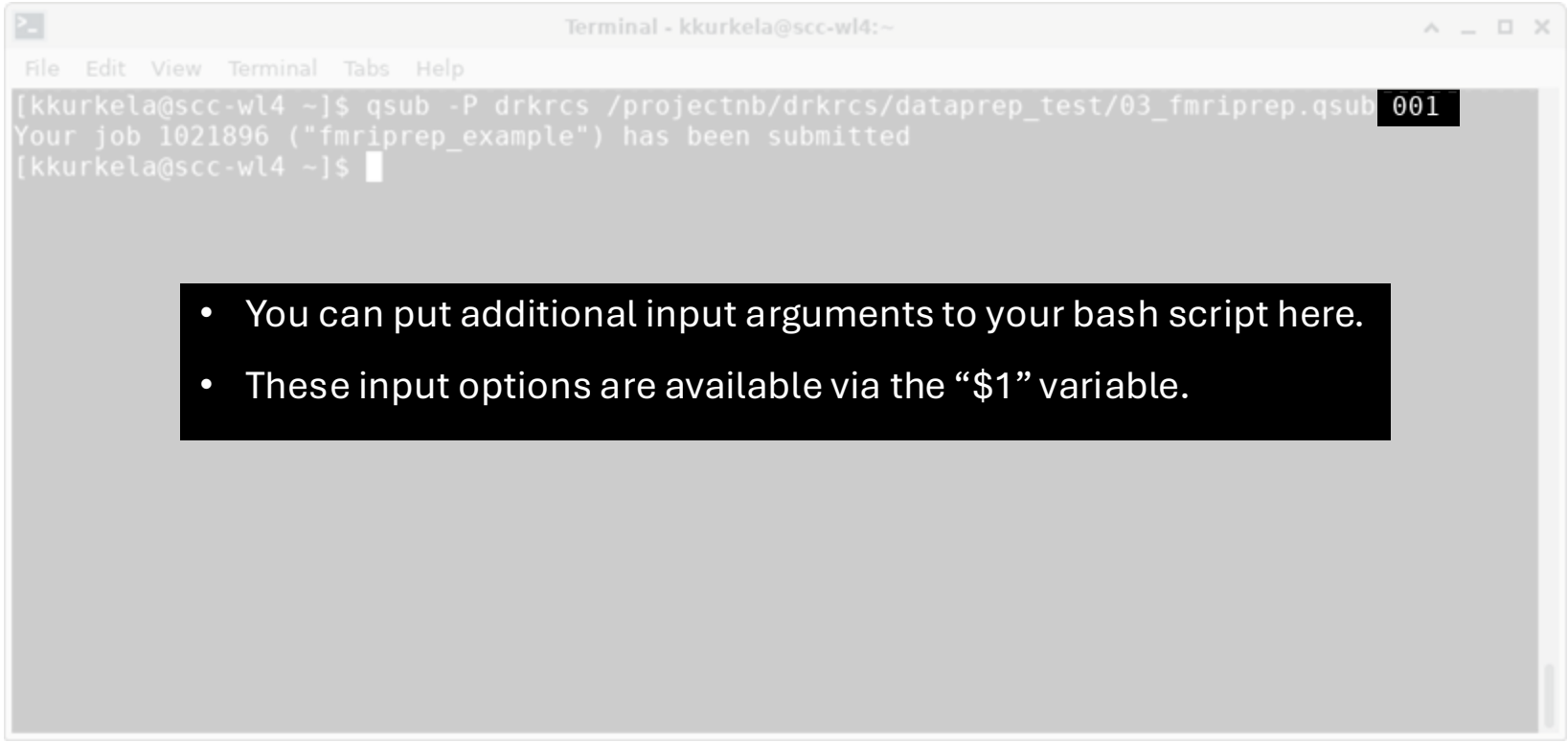
SCC Batch Job on the SCC – qsub



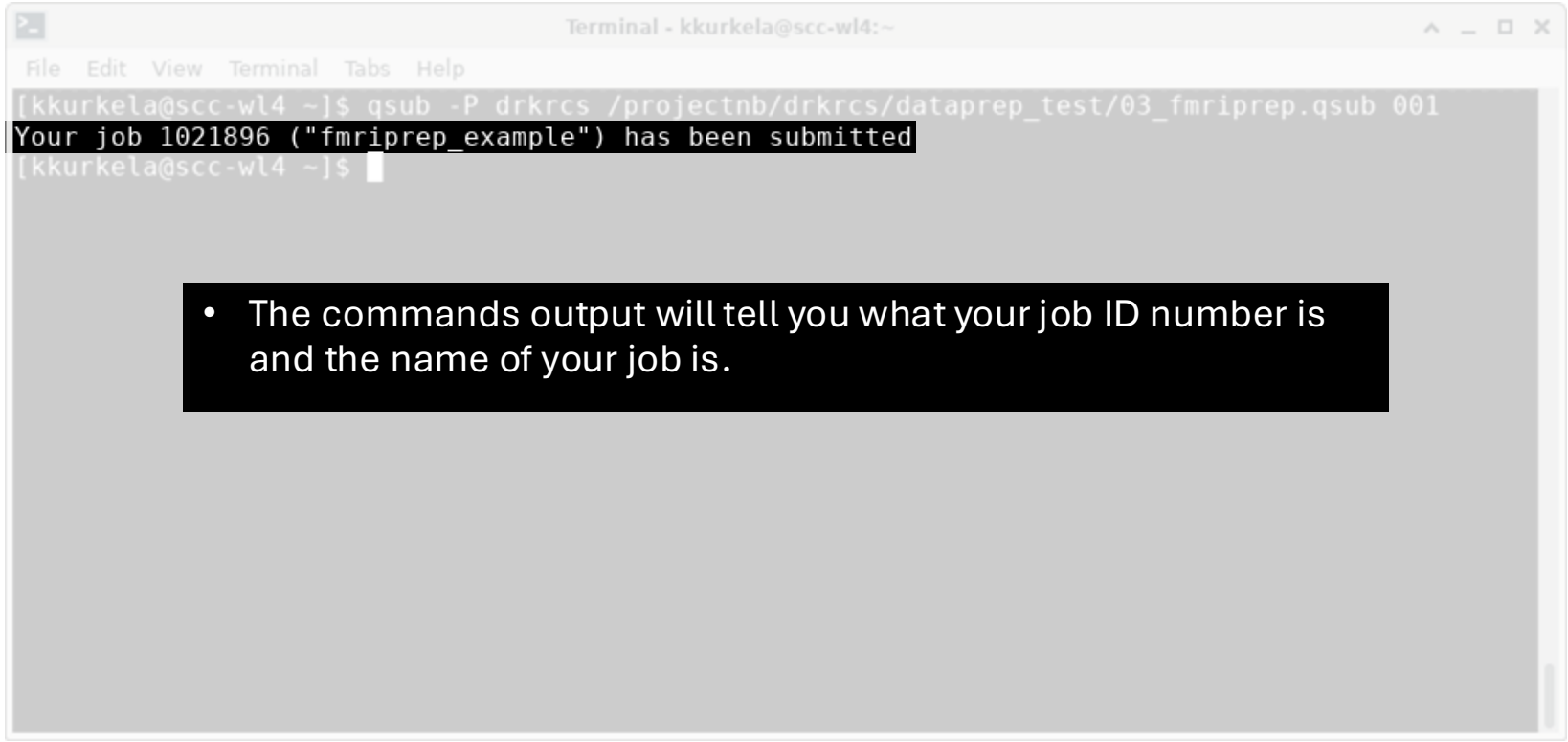
- Additional options for job submission.
- This example overwrites the –P flag in your qsub script and runs the job with the project “drkrcs” instead.
- See qsub –help for more options.
- Most of the time you will not need these.

SCC Batch Job on the SCC – qsub

SCC Batch Job on the SCC – qsub

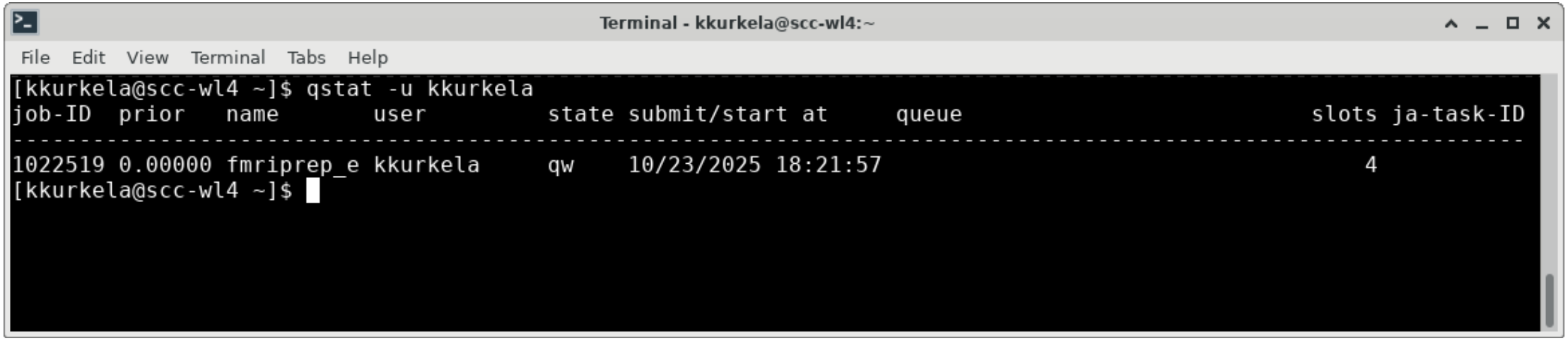


SCC Batch Job on the SCC – qsub



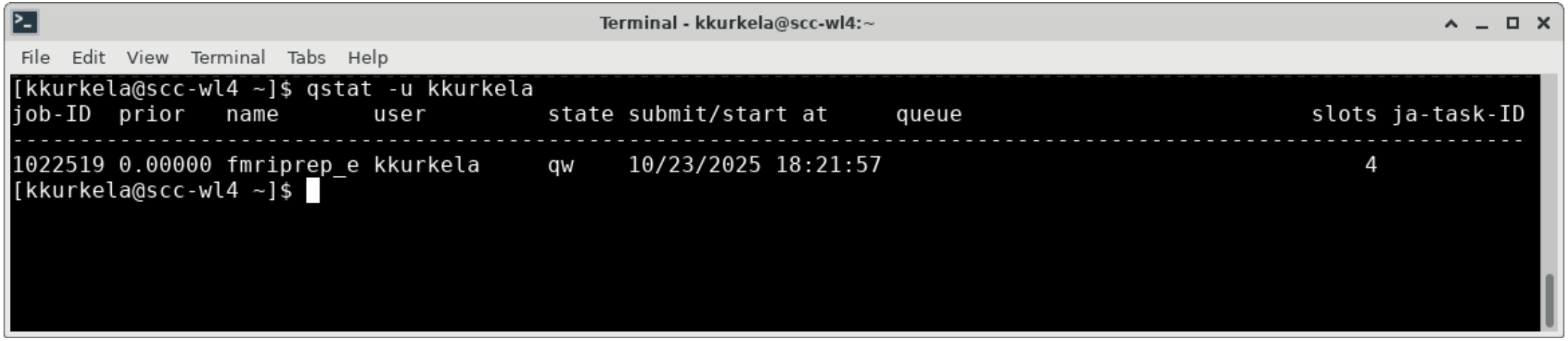
- The commands output will tell you what your job ID number is and the name of your job is.

Checking the status of your batch job



qstat –u yourusername

Checking the status of your batch job



Output lists an entry for each of your jobs.
The most common job statuses are queued (qw), running (), or complete ().
It will also list the queue and the computer where your job is currently running.

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Assessing your Data

- Now that you have your data organized, how do you know if it is any good?
- In neuroimaging, there are several common data quality concerns including but not limited to:
  - Singal dropout, typically near the sinuses and ear canals
  - Magnetic Field Inhomogeneities
  - Participant motion

Software Spotlight – *MRIQC*

- MRIQC is a BIDS compatible software package that will create a nicely formatted html report giving a snapshot overview of the quality of an MRI scan
- MRIQC is designed to use the best performing tools from other MRI software packages with minimal data processing time
- Once your data in compliance with BIDS, the running of the software package is straightforward!
- https://mriqc.readthedocs.io/en/latest/

MRIQC

- Let's take a closer look at an example batch script for running MRIQC on the SCC. Follow along with me…
- NOTE: all commands used in today's tutorial will be in a publicly accessible text file *notes.txt*
  - https://rcs.bu.edu/examples/imaging/tut_dataprep_scc/02_mriqc.qsub
  - https://rcs.bu.edu/examples/imaging/tut_dataprep_scc/notes.txt

An Example MRIQC Individual Report

- Let’s take a closer look at an example output report:
  - T1 and T2 Images:
    - https://mriqc.readthedocs.io/en/latest/reports/smri.html
  - BOLD Images:
    - https://mriqc.readthedocs.io/en/latest/reports/bold.html

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Workshop Overview

1. Connecting to the SCC
2. Organizing Data on the SCC with *dcm2bids*
   - The Brain Imaging Data Structure (BIDS)
3. Batch Job 101
4. Assessing the quality of your data with *mriqc*
5. Preprocessing your data with *fmriprep*

Preprocessing your Data

- Now that your data are organized and are of high quality, how do you prepare the data for analysis?
- Typically, in MRI studies you want to perform a set of data processing steps prior to analyzing your data. They include:
  - Brain extraction and defacing
  - Motion correction
  - Slice timing correction
  - Smoothing
  - Registration and Normalization
- There are many different preprocessing pipelines each with their own algorithms
  - How do you choose which software package to use? Which ones are the best?

Software Spotlight – *FMRIPREP*

- FMRIPREP is a BIDS compatible software package that creates a preprocessing pipeline from all the best performing parts of other commonly used MRI processing software
  - For example: a smoothing algorithm from SPM, a normalization algorithm from AFNI, etc, etc.
- FMRIPREP offers a ton of options for tinkering with its internals
- Like MRIQC, creates nicely formatted html reports on the success (or failure) of the preprocessing pipeline
- Once your data is following BIDS, running FMRIPREP is easy!

FMRIPREP

- Let's return once again to the SCC and examine a batch script designed to run fmriprep. Follow along with me…
- NOTE: all commands used in today's tutorial will be in a publicly accessible text file *notes.txt*
  - https://rcs.bu.edu/examples/imaging/tut_dataprep_scc/notes.txt

An Example FMRIPREP Report

- Lets take a look at an example output report for FMRIPREP:
  - https://fmriprep.org/en/stable/outputs.html

Additional Web Resources

- Research Computing Support Pages
  - www.bu.edu/tech/support/research/
- Software Packages Available on the SCC
  - www.bu.edu/tech/support/research/software-and-programming/software-and-applications/
- Brain Imaging Data Standard
  - bids.neuroimaging.io
- *dcm2bids*
  - unfmontreal.github.io/Dcm2Bids/3.2.0/
- *MRIQC*
  - mriqc.readthedocs.io/en/latest/
- *FMRIPREP*
  - fmriprep.org/en/stable/