Command-line genomics

Jelmer Poelstra, MCIC

Apr 4, 2023

What I'll talk about today

Today, I will give an overview of the **typical computational environment infrastructure used for genomics projects**.

Most of this can be summarized as "**command-line genomics**", and I will explain what this entails, and why you need the command line.

Of course, I won't have time to teach you the different components, but I would like to orient you on this topic so that it's not as much of a black box.

This will hopefully give you a starting point for learning more – and I will point you to some specific resources as well.

What kind of research is this relevant for?

What kind of research is this relevant for?

- Any project in which you generate *high-throughput sequencing data*, e.g.:

  → **Whole-genome sequencing** – de novo assembly, pangenomics, "resequencing"

  → **Reduced-representation sequencing** (GBS, etc) for population genomics

  → **Microbiomics** – both shotgun metagenomics and amplicon metabarcoding

  → **Transcriptomics** with RNAseq

What kind of research is this relevant for?

- Any project in which you generate *high-throughput sequencing data*, e.g.:

  → **Whole-genome sequencing** – de novo assembly, pangenomics, "resequencing"

  → **Reduced-representation sequencing** (GBS, etc) for population genomics

  → **Microbiomics** – both shotgun metagenomics and amplicon metabarcoding

  → **Transcriptomics** with RNAseq

- Other genomics projects like **comparative genomics** with publicly available genomes.

What kind of research is this relevant for?

- Any project in which you generate *high-throughput sequencing data*, e.g.:

  → **Whole-genome sequencing** – de novo assembly, pangenomics, "resequencing"

  → **Reduced-representation sequencing** (GBS, etc) for population genomics

  → **Microbiomics** – both shotgun metagenomics and amplicon metabarcoding

  → **Transcriptomics** with RNAseq

- Other genomics projects like **comparative genomics** with publicly available genomes.

- But only to some extent to *proteomics* and *metabolomics*.

Overview

1. Introduction
2. **Command-line genomics?**
3. Ohio Supercomputer Center (OSC) overview
4. Command-line software
5. The VS Code editor and the whole game

A typical computational infrastructure for genomics

A typical computational infrastructure for genomics

- A **supercomputer** – in our case, the Ohio Supercomputer Center (OSC)
  (Cloud computing is an alternative, but won't be covered today)

A typical computational infrastructure for genomics

- A **supercomputer** – in our case, the Ohio Supercomputer Center (OSC)
  (Cloud computing is an alternative, but won't be covered today)
- The **Unix shell** (terminal)

A typical computational infrastructure for genomics

- A **supercomputer** – in our case, the Ohio Supercomputer Center (OSC)
  (Cloud computing is an alternative, but won't be covered today)
- The **Unix shell** (terminal)
- A **text editor** – I recommend and will demonstrate *VS Code*

A typical computational infrastructure for genomics

- A **supercomputer** – in our case, the Ohio Supercomputer Center (OSC)
  (Cloud computing is an alternative, but won't be covered today)
- The **Unix shell** (terminal)
- A **text editor** – I recommend and will demonstrate *VS Code*

Command-line genomics

For the purposes of this talk, I will refer to working with the above elements by running command-line programs as "batch jobs" (non-interactively) as **command-line genomics**.

A typical computational infrastructure for genomics

- A **supercomputer** – in our case, the Ohio Supercomputer Center (OSC)
  (Cloud computing is an alternative, but won't be covered today)
- The **Unix shell** (terminal)
- A **text editor** – I recommend and will demonstrate *VS Code*

Command-line genomics

For the purposes of this talk, I will refer to working with the above elements by running command-line programs as "batch jobs" (non-interactively) as **command-line genomics**.

- **R** or perhaps *Python* for interactive statistical analysis and visualization.
  (I won't talk more about this other than showing you how you can use R at OSC)

What is a supercomputer?

A highly interconnected set of many computer processors and storage units.
You can think of it simply as **a network of computers**.

Supercomputers are also commonly referred to as *High-Performance Computing (HPC) clusters* or simply *compute clusters*.

Why do I need a supercomputer?

- Your genomics **dataset is often too large** to be handled efficiently, or even at all, by a laptop or desktop computer.
- To **speed up long-running analyses** by using more computing power.
- To **speed up repeated analyses**, like the independent mapping of reads for different samples to a reference genome: these can be run in parallel on a supercomputer.
- It's also a great place to **store large amounts of data**

What is the Unix shell?

A computer's **shell** is also referred to as a *Terminal* or "*the command line*", and allows you to interact with your computer by **typing commands** rather than pointing-and-clicking.

The *Unix* shell is the shell of Unix-based computers, which include Mac and Linux (but not Windows) operating systems.[1]

```
Host: pitzer.osc.edu
[jelmer@pitzer-login04 ~]$ mkdir mynewfolder
```

1. And the most common Unix shell is the **Bash shell**, which runs the Bash language

Why do I need to work in the Unix shell?

Why do I need to work in the Unix shell?

- A genomics project usually involves sequentially running a whole array of bioinformatics programs (or “tools”).

  For instance, an RNAseq project may include:
  raw read QC => raw read trimming => trimmed read mapping => gene counting

Why do I need to work in the Unix shell?

- A genomics project usually involves sequentially running a whole array of bioinformatics programs (or "tools").

  For instance, an RNAseq project may include:
  raw read QC => raw read trimming => trimmed read mapping => gene counting

- **Many of these tools can only be run through a "command-line interface" (CLI)**

  Even those that have a "graphical user interface" (GUI) are more efficiently and reproducibly run through a CLI:

  → **Efficiency** — A CLI allows you to write a simple loop to run it in the same way for many samples. (In combination with the computing power of a supercomputer, this in turn allows you to process those hundreds of samples *in parallel*.)

  → **Reproducibility** — You can easily save all commands and scripts which would allow you to rerun a project rather straightforwardly.

How/when can I *avoid* all of this?

If you will often do genomics projects like the ones mentioned above, it's hard to avoid command line genomics as described.

But here are some conditions in which you might reasonably avoid it:

- You're doing a single genomics project, your main research focus is elsewhere
- You have data which can be analyzed with no or a relatively small command-line-based part, such as proteomics/metabolomics/metabarcoding/RNAseq.

In such cases, you might be able to get someone else to do the command line work, or you could try ***Galaxy***, a cloud-based bioinformatics platform with a web browser interface and no coding.

Overview

1. Introduction
2. Command-line genomics?
3. **Ohio Supercomputer Center (OSC) overview**
4. Command-line software
5. The VS Code editor and the whole game

The Ohio Supercomputer Center (OSC)

Ohio Supercomputer Center

An OH·TECH Consortium Member

The Ohio Supercomputer Center (OSC)

- The **Ohio Supercomputer Center (OSC)** provides computing resources to researchers (and others) across Ohio.

  OSC has two individual supercomputers/clusters (named Owens and Pitzer), and lots of infrastructure for their usage.

Ohio Supercomputer Center
An OH·TECH Consortium Member

The Ohio Supercomputer Center (OSC)

- The **Ohio Supercomputer Center (OSC)** provides computing resources to researchers (and others) across Ohio.

  OSC has two individual supercomputers/clusters (named Owens and Pitzer), and lots of infrastructure for their usage.

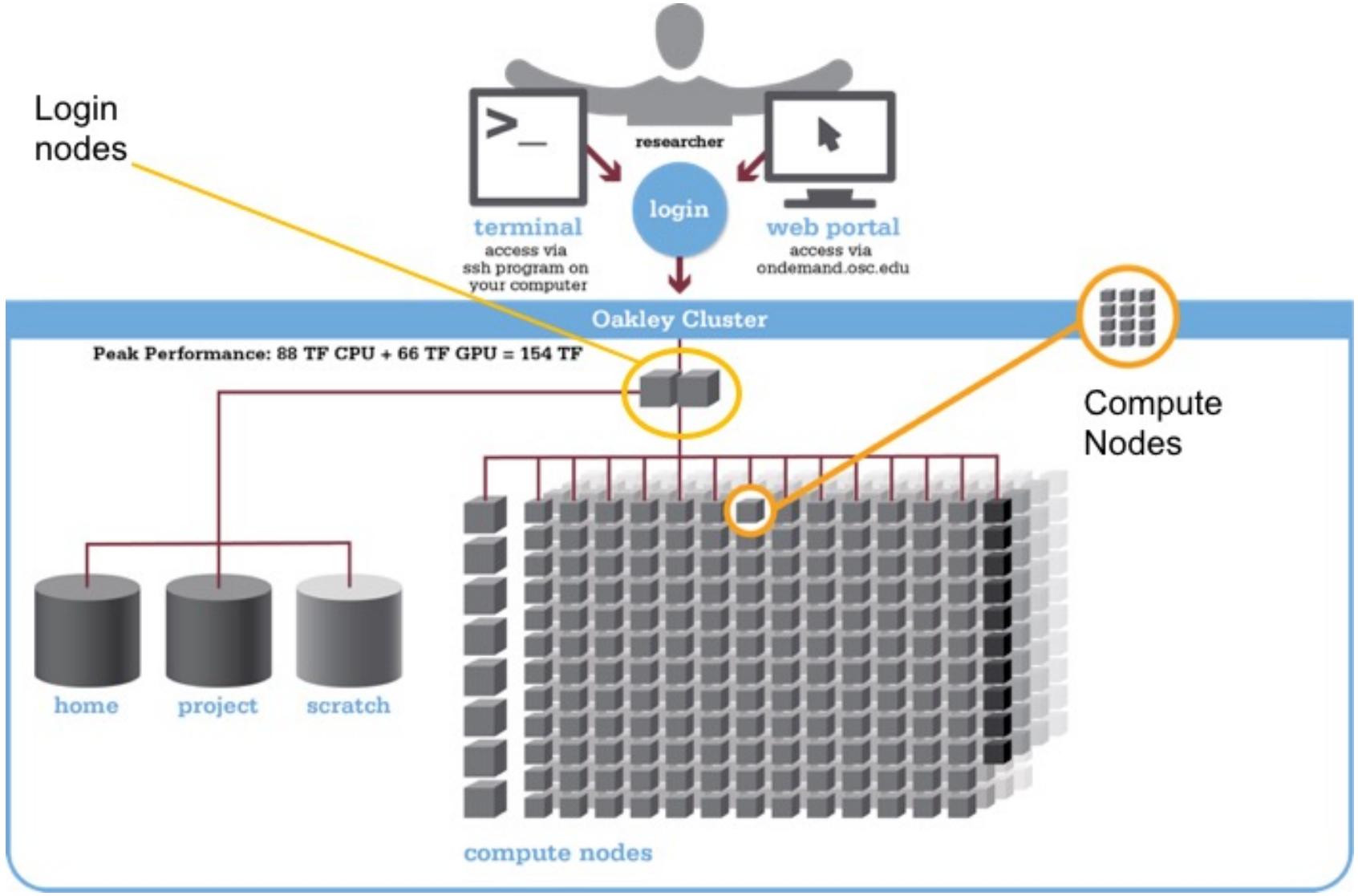
- Research usage is charged but at heavily subsidized rates, and most or all OSU colleges absorb these costs at the college level (!)

  Educational usage is entirely free, like for the `PAS2250` project you have been added to for this lecture.

Ohio Supercomputer Center

An OH·TECH Consortium Member

Supercomputer overview



OSC OnDemand

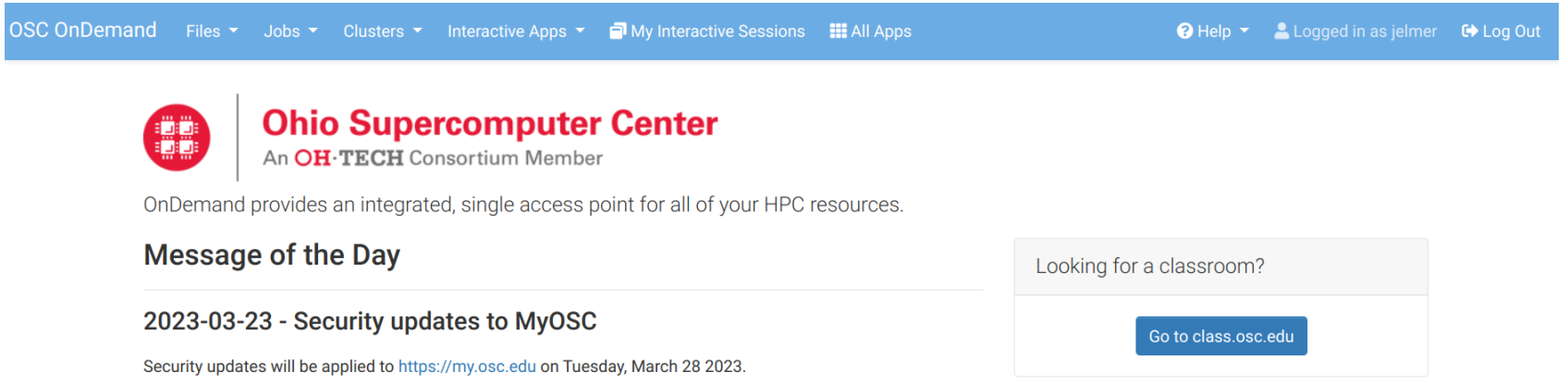
The OSC OnDemand web portal allows you to use a web browser to access OSC resources such as:

- A file browser where you can also create and rename folders and files, and download and upload files. [1]
- A Unix shell
- More than a dozen different "Interactive Apps", or programs with a GUI, such as RStudio, Jupyter, QGIS, and more.

1. But for large transfers, use a different interface: *Globus*

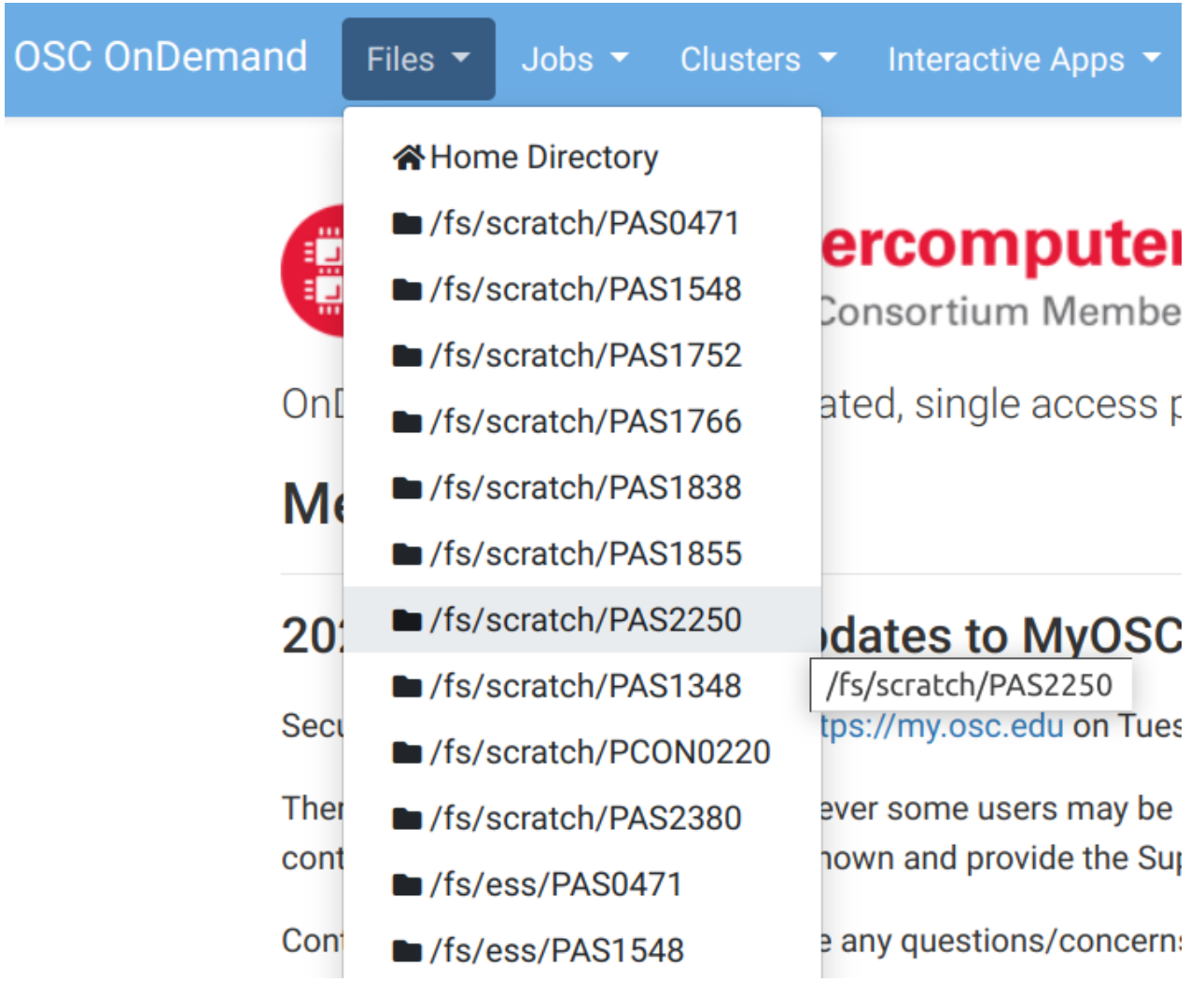
Connecting to OSC with OnDemand

- Go to https://ondemand.osc.edu, and log in with your OSC credentials.



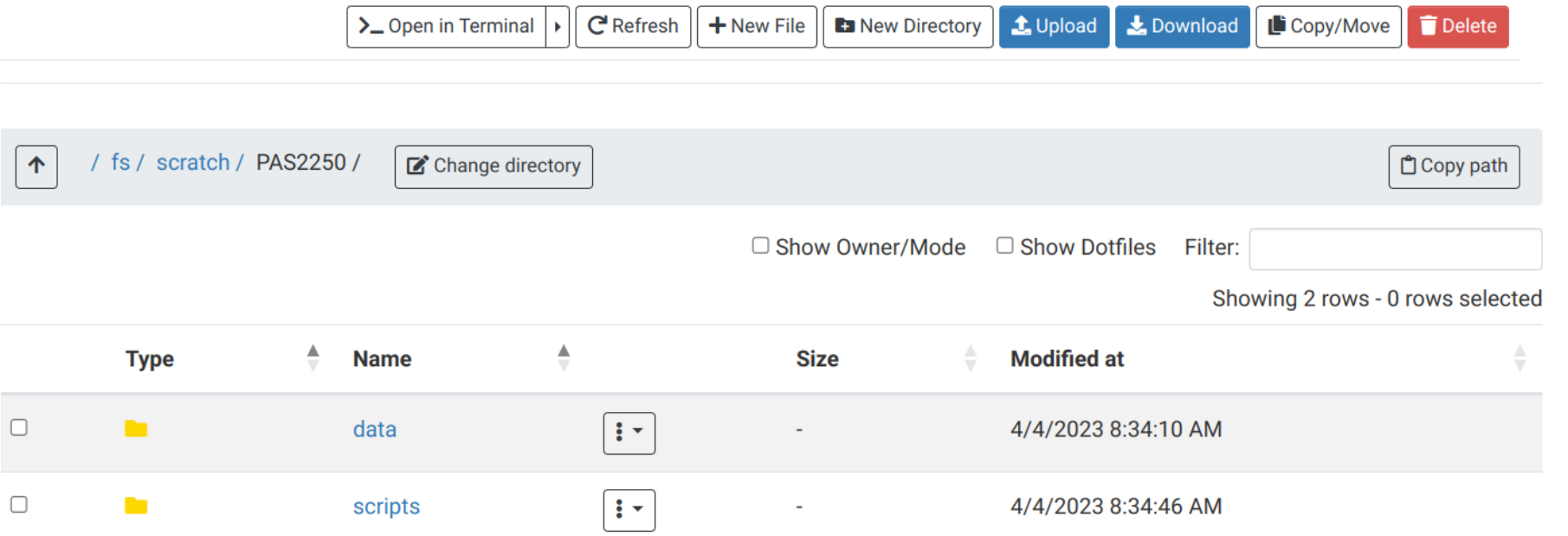
OnDemand "Files" menu

Choose a folder as a starting point for the file browser:



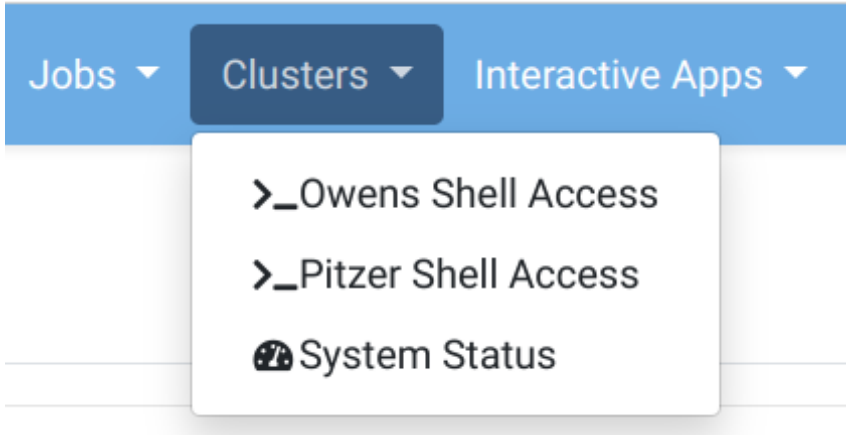
OnDemand "Files" menu

Here you can view, create and rename folders and files, and download and upload files:



OnDemand "Clusters" menu

Here you can access a Unix shell on either of the two clusters:



(Since the two clusters share the file systems, and they have fairly similar capabilities, it generally doesn't matter which cluster you connect to).

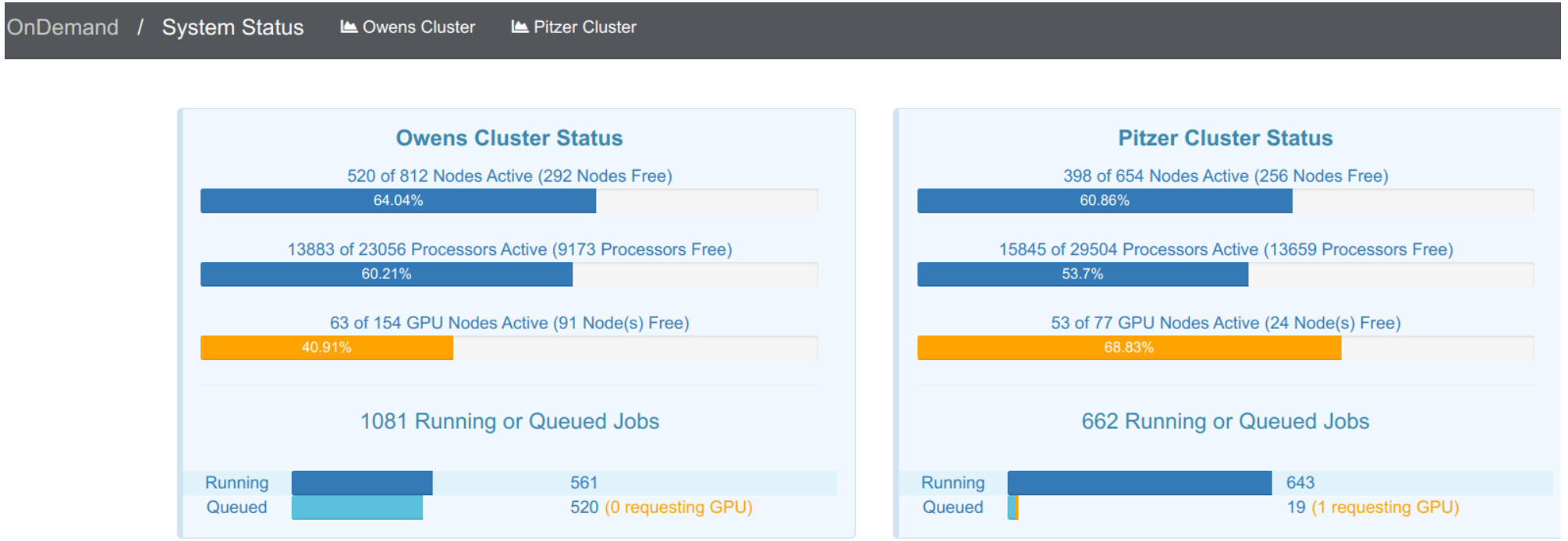
OnDemand "Clusters" menu: Shell

When you click on one of the shell options, a new browser tab with a shell will open. There are some welcome messages, and some storage usage/quota info, and then you get a prompt to type commands:

```
Host: pitzer.osc.edu                                                                    Themes: Default

with applications, such as MATLAB, RStudio, or Jupyter Notebook.

As an example, an OSU undergrad statistics class recently used iPads
to remotely access RStudio on OSC systems. We can provide online
demonstrations or evaluations and potentially add additional software
packages.

Please contact OSCHelp@osc.edu to talk to OSC about distance-learning
support options available to you.

*********************************************************************
2023/03/23
--- Security updates to MyOSC

Security updates will be applied to https://my.osc.edu on Tuesday,
March 28 2023.

There should be no interruption, however some users may be mistakenly
blocked. Should this occur, please contact OSC support using the link
shown and provide the Support ID shown.

Contact oschelp@osc.edu if there are any questions/concerns.

*********************************************************************


Your primary project is the only group quota reported here.
See http://osc.edu/check-quotas to learn how to check the group storage quotas for your other projects.

As of 2023-04-02T17:55:01.000000 userid jelmer on /fs/ess/PAS1752 used 0 GiB of quota 0 GiB and 22 files of quota 0 files
As of 2023-04-02T17:55:01.000000 userid jelmer on /fs/ess/PAS0471 used 14920 GiB of quota 0 GiB and 3445648 files of quota 0 files
As of 2023-04-02T17:55:01.000000 userid jelmer on /fs/ess/PAS1838 used 0 GiB of quota 0 GiB and 31 files of quota 0 files
As of 2023-04-02T17:55:01.000000 userid jelmer on /fs/ess/PAS1855 used 0 GiB of quota 0 GiB and 414 files of quota 0 files
As of 2023-04-02T17:55:01.000000 userid jelmer on /fs/ess/PAS2380 used 4305 GiB of quota 0 GiB and 742710 files of quota 0 files
As of 2023-04-02T17:55:01.000000 project/group PAS0471 on /fs/ess used 55434 GiB of quota 66560 GiB and 14058407 files of quota 20000000 files
As of 2022-10-12T07:50:01.000000 userid jelmer on /fs/project/PAS0471 used 17191 GiB of quota 0 GiB and 3284217 files of quota 0 files
As of 2022-10-12T07:50:01.000000 project/group PAS0471 on /fs/project used 57136 GiB of quota 66560 GiB and 11392591 files of quota 15000000 files
As of 2023-04-02T17:55:01.000000 userid jelmer on /fs/scratch used 16780 GiB of quota 102400 GiB and 447307 files of quota 26214400 files
As of 2023-04-02T17:51:07.000000 userid jelmer on /users/PAS0471 used 167.09 GiB of quota 500 GiB and 882372 files of quota 1000000 files

[jelmer@pitzer-login04 ~]$
```

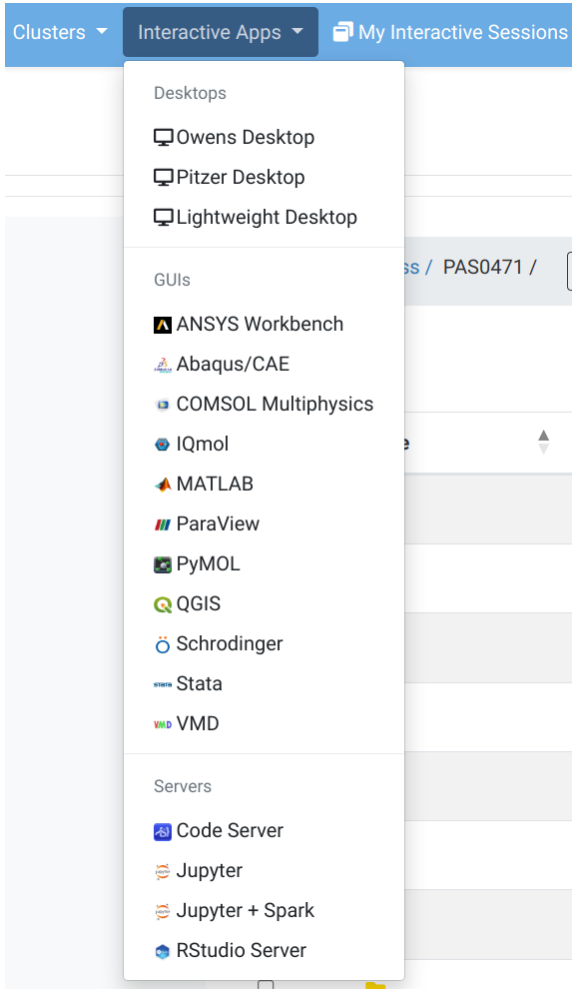
OnDemand "Clusters" menu: System Status

If you click on "System status", you'll get an overview of the current usage of the two clusters:



OnDemand "Interactive Apps" menu

Here you can mostly access programs with a GUI that will run on a compute node. We'll try `RStudio Server` now (and `Code Server` a little later).



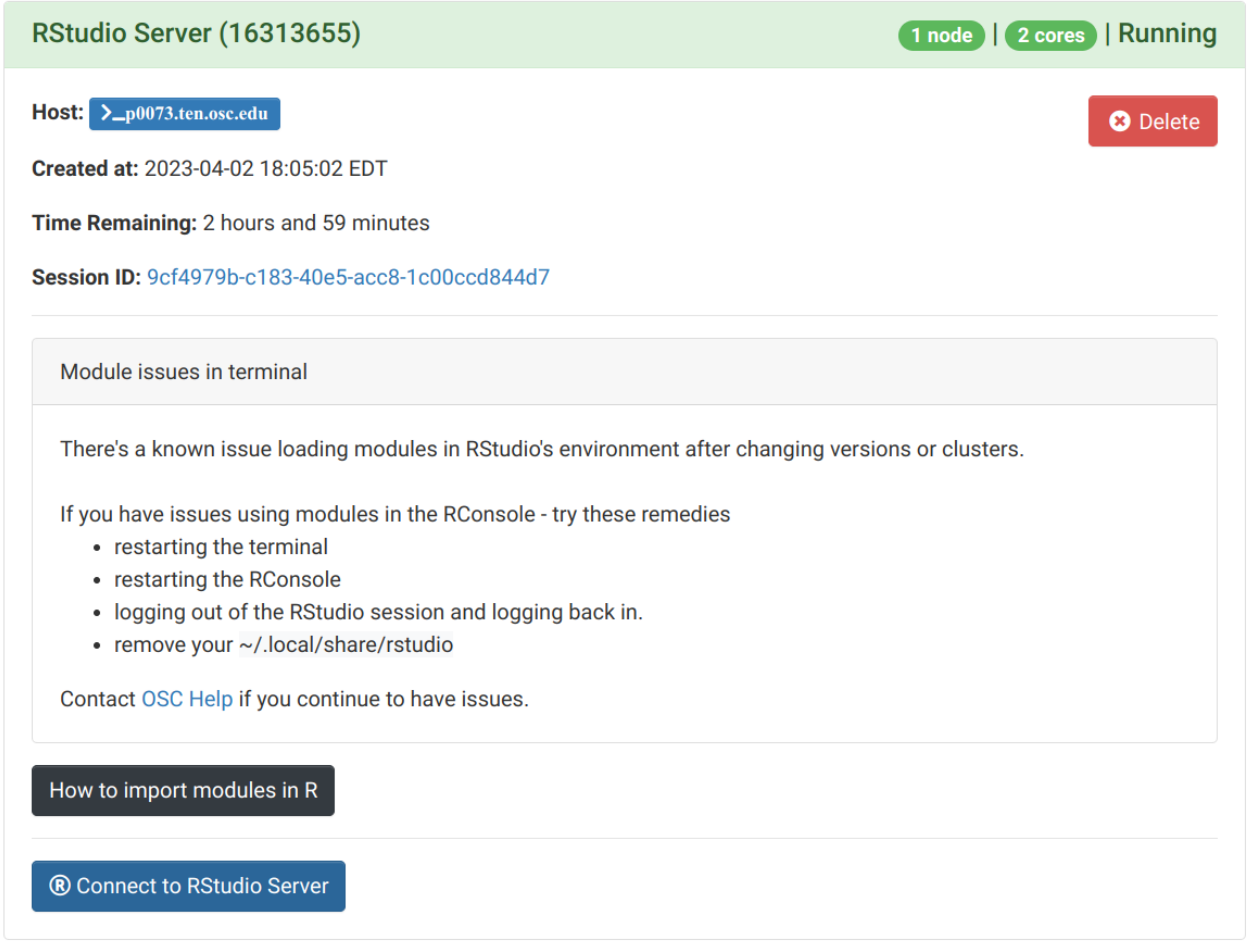
OnDemand "Interactive Apps": RStudio Form

Fill out this form to start an RStudio session.

This will run on a compute node and is therefore charged: for that reason, it needs the OSC account number so as to bill the correct account.

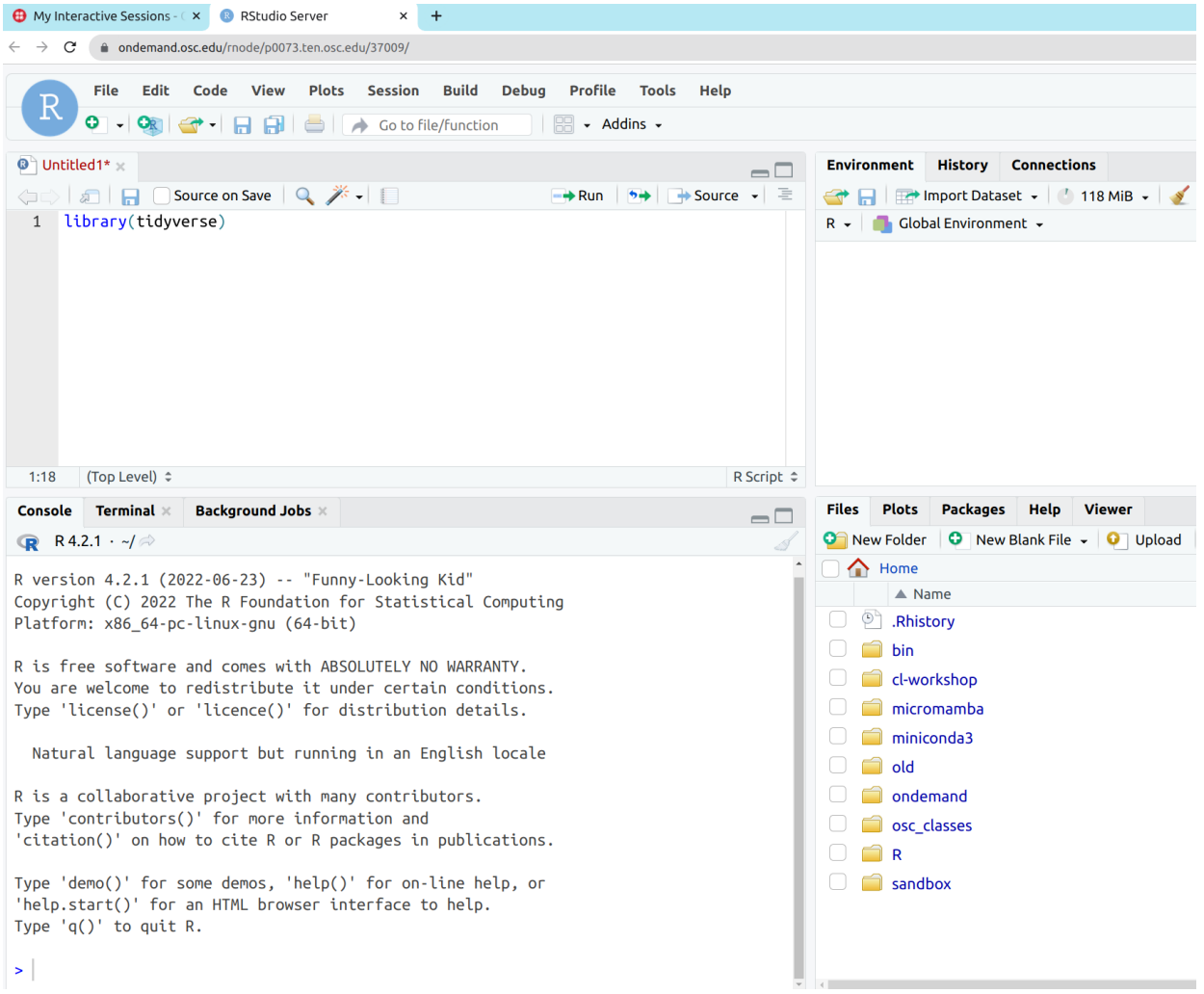
OnDemand "Interactive Apps": RStudio

Once the top bar of the box like the one shown below turns green and says "Running", you can click "**Connect to RStudio Server**" way at the bottom:



OnDemand "Interactive Apps": RStudio

Now, you'll have RStudio running in your browser!
It looks just like the desktop app version you may be familiar with:



What is different at OSC / on a supercomputer?

What is different at OSC / on a supercomputer?

- **Software**
  Because you don't have administrator rights, and because the system is shared by so many people, you can't install and use software "the regular way".

  → For system-wide installed software, **load it** with `module` commands.

  → If something is not installed, ask OSC or use *Conda* or containers.

What is different at OSC / on a supercomputer?

- **Software**
  Because you don't have administrator rights, and because the system is shared by so many people, you can't install and use software "the regular way".

  → For system-wide installed software, **load it** with `module` commands.

  → If something is not installed, ask OSC or use *Conda* or containers.

- **Login versus compute nodes** (node ≈ computer)
  "Login nodes", the nodes you end up on after logging in, are not meant for heavy computing and you have to *request access to "compute nodes"* to run most analyses.

What is different at OSC / on a supercomputer?

- **Software**
  Because you don't have administrator rights, and because the system is shared by so many people, you can't install and use software "the regular way".

  → For system-wide installed software, **load it** with `module` commands.

  → If something is not installed, ask OSC or use *Conda* or containers.

- **Login versus compute nodes** (node ≈ computer)
  "Login nodes", the nodes you end up on after logging in, are not meant for heavy computing and you have to *request access to "compute nodes"* to run most analyses.

- **"Non-interactive" usage is common, using a job scheduler (SLURM)**
  You submit your scripts to the SLURM queue and monitor the resulting jobs.

Overview

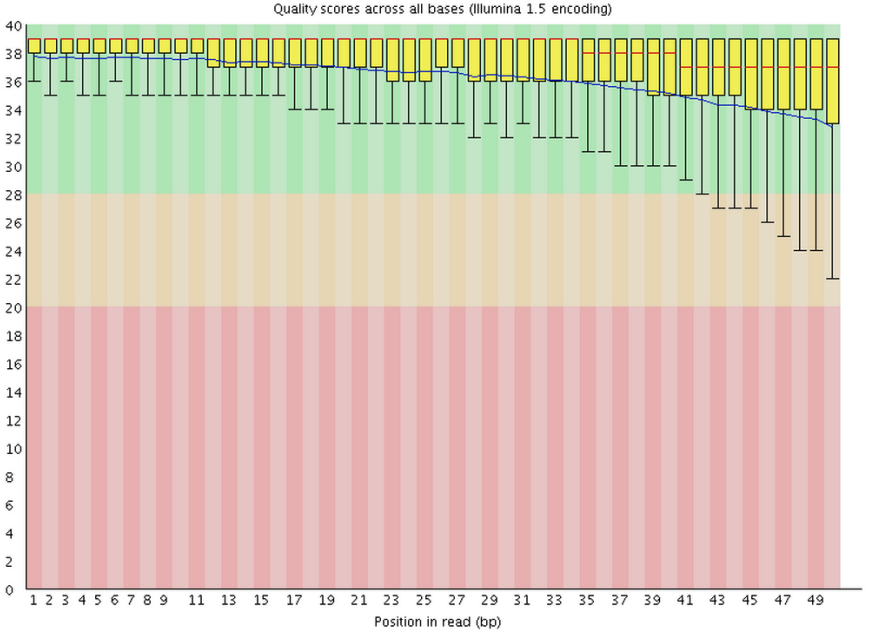
1. Introduction
2. Command-line genomics?
3. Ohio Supercomputer Center (OSC) overview
4. **Command-line software**
5. The VS Code editor and the whole game

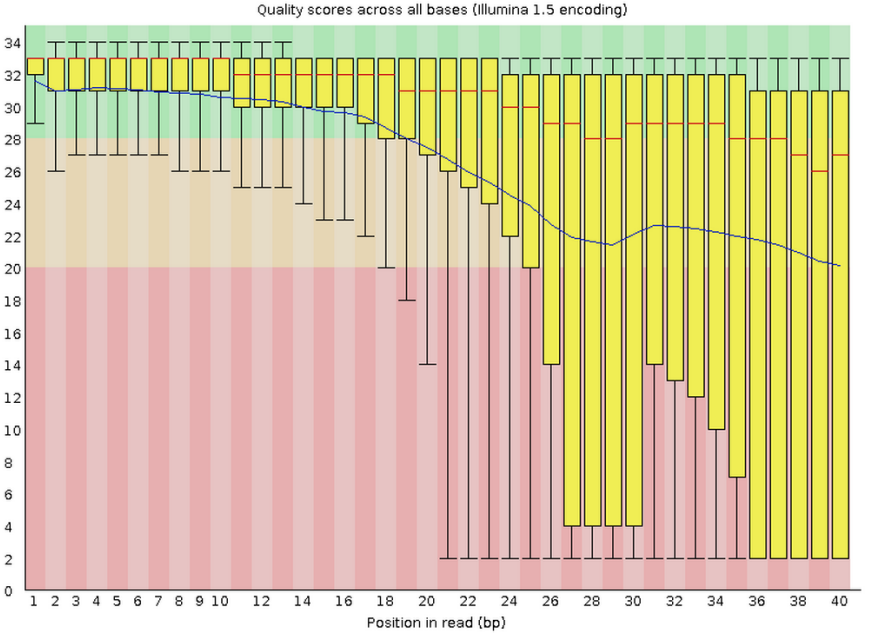
An example of a command-line program

A useful example of a genomics tool with a CLI is **FastQC, a program for quality control of FASTQ files.**

It is ubiquitous because nearly all high-throughput sequencing data comes in FASTQ files, and your first step is always to check the quality of the reads.

FastQC produces visualizations and assessments of aspects of your reads such as adapter content, and, as shown below, **mean base quality along the read**:





Running FastQC

Running FastQC

- To run FastQC, you use the command `fastqc`.

  Command-line programs are typically run non-interactively, so we don't fire up the program first, and tell it what to do later, like we would with a program with a GUI.

  Instead, we at once issue a complete set of instructions for the program to do what we would like it to.

Running FastQC

- To run FastQC, you use the command `fastqc`.

  Command-line programs are typically run non-interactively, so we don't fire up the program first, and tell it what to do later, like we would with a program with a GUI.

  Instead, we at once issue a complete set of instructions for the program to do what we would like it to.

- For example, say we want to analyze one of the FASTQ files that I put in `/fs/scratch/PAS2250/data`, with default FastQC settings.

  A complete FastQC command would be:

```
[jelmer@owens-login04 ~]$ fastqc /fs/scratch/PAS2250/data/sample1.fastq.gz
```

  So, it is simply `fastqc` followed by a space and the name of the file!

Loading the FastQC "module" at OSC

1. full list of installed software: https://www.osc.edu/resources/available_software/software_list

Loading the FastQC "module" at OSC

- FastQC is available to us at OSC[1], but we first have to **load it**. Here is what happens when we try to run the program in a fresh shell session at OSC:

```
[jelmer@owens-login04 ~]$ fastqc /fs/scratch/PAS2250/data/sample1.fastq.gz
#> -bash: fastqc: command not found
```

1. full list of installed software: https://www.osc.edu/resources/available_software/software_list

Loading the FastQC "module" at OSC

- FastQC is available to us at OSC[1], but we first have to **load it**. Here is what happens when we try to run the program in a fresh shell session at OSC:

```
[jelmer@owens-login04 ~]$ fastqc /fs/scratch/PAS2250/data/sample1.fastq.gz
#> -bash: fastqc: command not found
```

- We can load the software as follows:

```
[jelmer@owens-login04 ~]$ module load fastqc
```

1. full list of installed software: https://www.osc.edu/resources/available_software/software_list

Loading the FastQC "module" at OSC

- FastQC is available to us at OSC[1], but we first have to **load it**. Here is what happens when we try to run the program in a fresh shell session at OSC:

```
[jelmer@owens-login04 ~]$ fastqc /fs/scratch/PAS2250/data/sample1.fastq.gz
#> -bash: fastqc: command not found
```

- We can load the software as follows:

```
[jelmer@owens-login04 ~]$ module load fastqc
```

- Now, let's try again:

```
[jelmer@owens-login04 ~]$ fastqc /fs/scratch/PAS2250/data/sample1.fastq.gz
#> Started analysis of sample1.fastq.gz
#> Approx 5% complete for sample1.fastq.gz
#> Approx 10% complete for sample1.fastq.gz
#> Approx 15% complete for sample1.fastq.gz
#> [truncated]
```

Success!

1. full list of installed software: https://www.osc.edu/resources/available_software/software_list

Something is missing here

I mentioned earlier that one benefit of running programs at the command-line is reproducibility – but how do we save the commands that we run?

- We need to not just save them, but to keep a detailed digital notebook that will enable us to redo our analysis.
- We also need to wrap these commands in little scripts, so that we can run programs non-interactively and in parallel.

For all of this, we will need a good **text editor**.

Overview

1. Introduction
2. Command-line genomics?
3. Ohio Supercomputer Center (OSC) overview
4. Command-line software
5. **The VS Code editor and the whole game**

The VS Code text editor

VS Code (in full, "Visual Studio Code") is a nice modern GUI-based text editor. [1]
We can use a version of this editor (often referred to as *Code Server*) in our browser through OSC OnDemand.

Because it also has an integrated terminal to access a Unix shell, this setup effectively combines the 3 aspects of command-line genomics:

- Supercomputer
- Unix shell
- Text editor

(And while I personally prefer RStudio for R, you can also run that through VS Code).

1. While there are also command-line text editors, these make the learning curve for command-line genomics even steeper

Starting up VS Code in OnDemand

1. Log in to OSC's OnDemand portal at https://ondemand.osc.edu.
2. In the blue top bar, select `Interactive Apps` and then near the bottom of the dropdown menu, click `Code Server`.
3. In the form that appears on a new page:
   - Select project `PAS2250`
   - No need to change "Number of hours" and "Working Directory"
   - Make sure the "Codeserver Version" is `4.8`.
4. On the next page, once the top bar of the box has turned green and says `Runnning`, click `Connect to VS Code`.

Starting up VS Code in OnDemand

Code Server version: v0.7.0

This app will launch a VS Code server using Code Server on the Pitzer cluster.

Project

PAS2250

Number of hours

1

Working Directory

Select your project directory; defaults to $HOME

Select Path

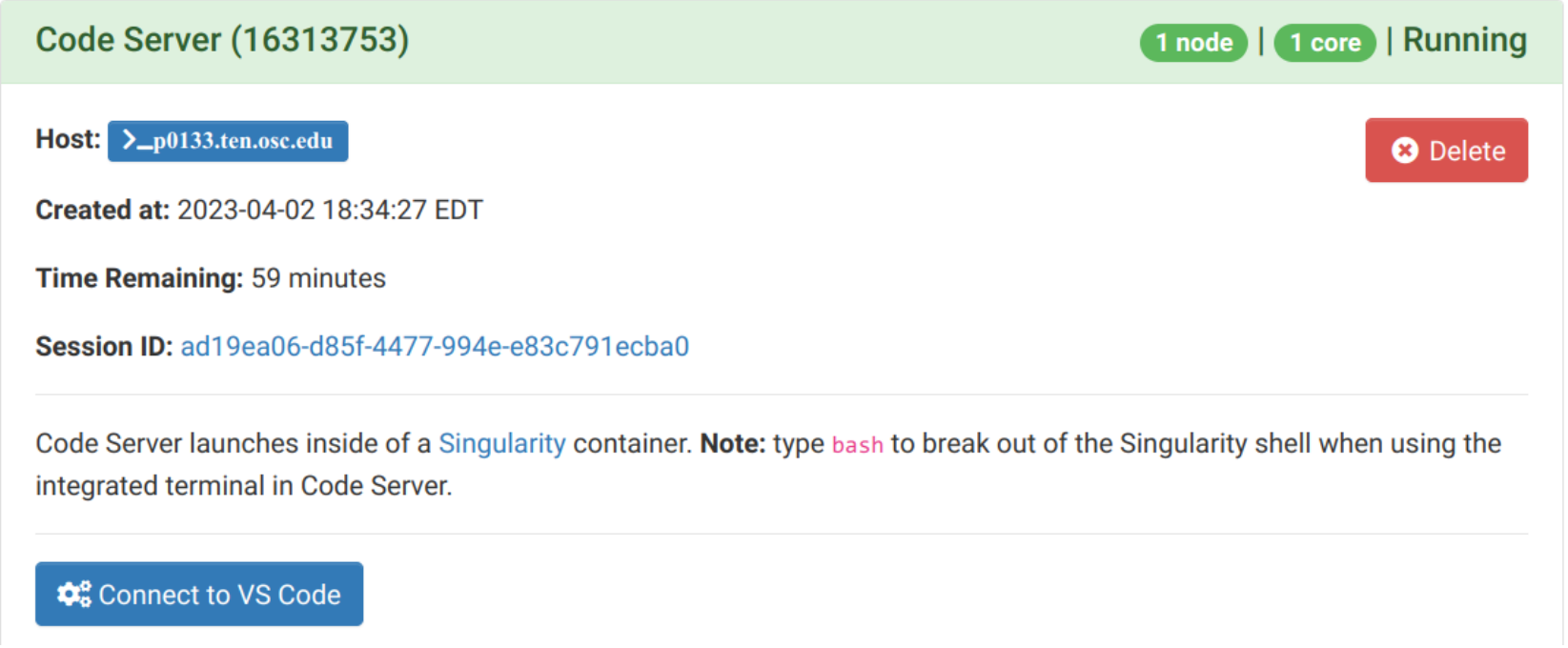
Codeserver Version

4.8

Launch

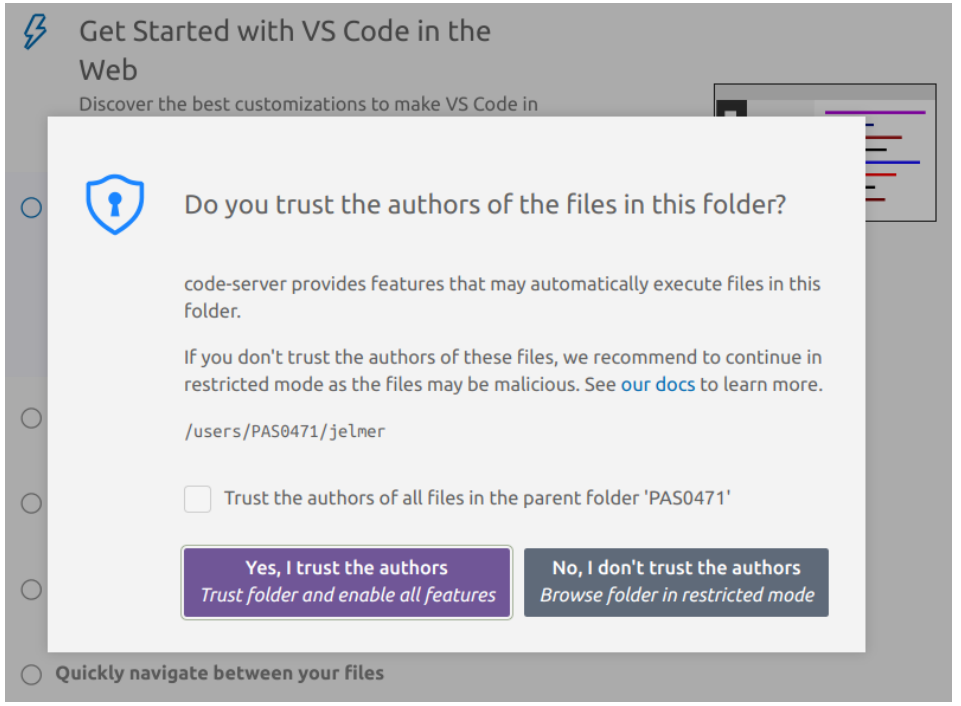
Starting up VS Code in OnDemand

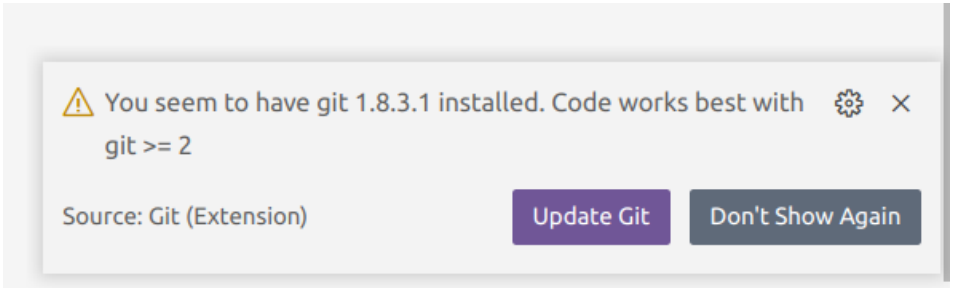
Once the session is running, you can click “Connect to VS Code”:



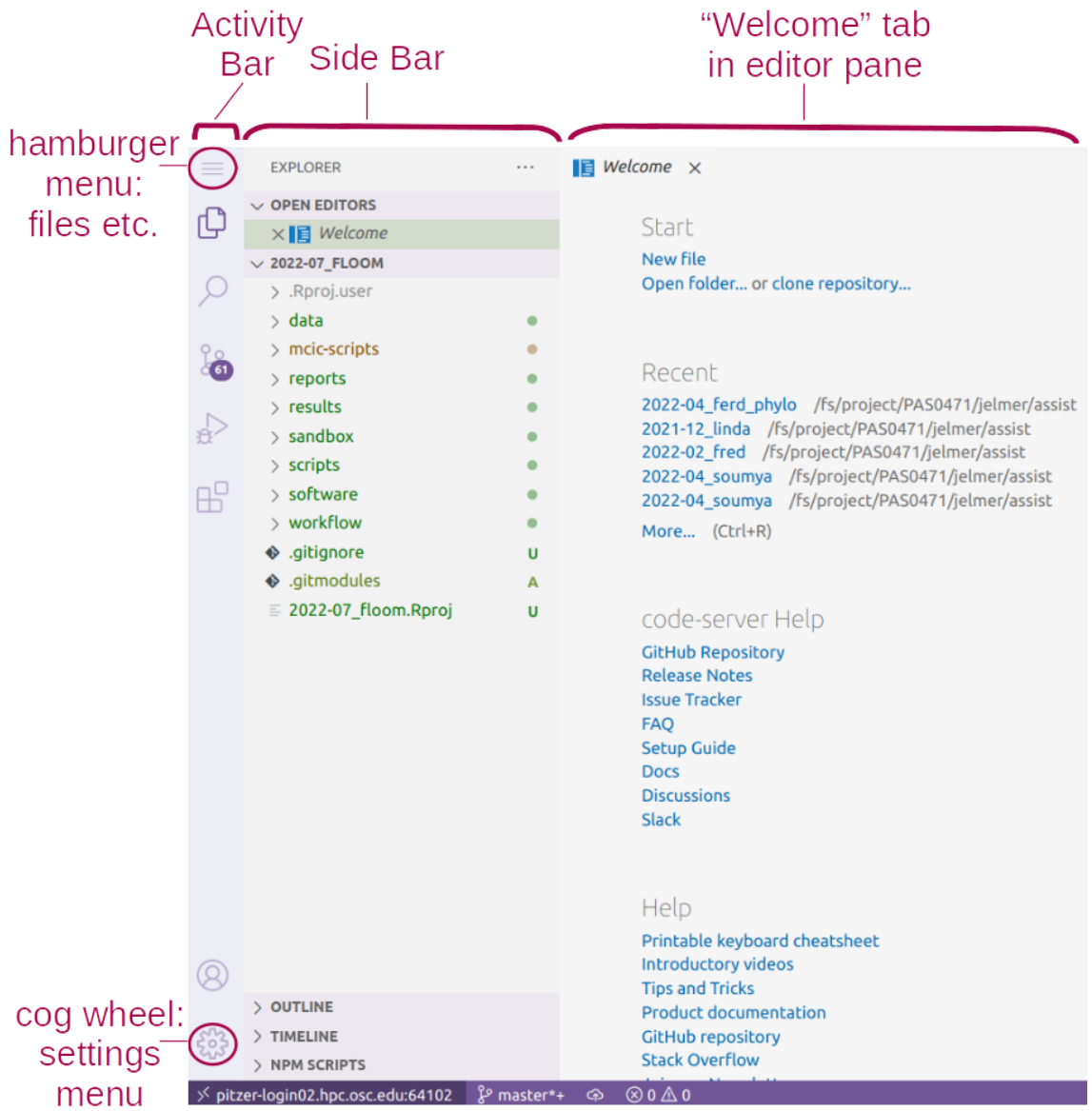
Starting up VS Code in OnDemand

You'll likely get these two pop-ups – click "Yes" and "Don't Show Again", respectively:





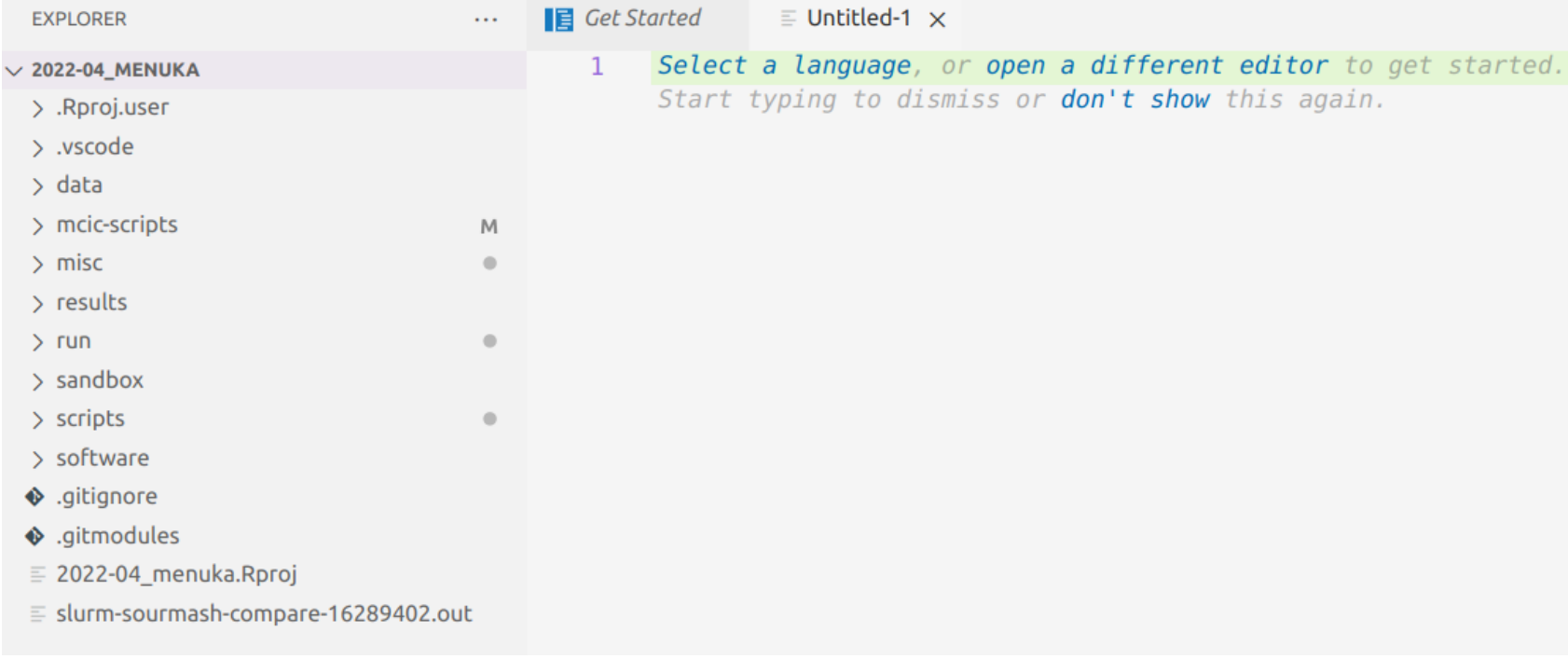
VS Code Basics: Side bar with file browser



VS Code Basics: Editor pane

The main part of the VS Code is the *editor pane*.

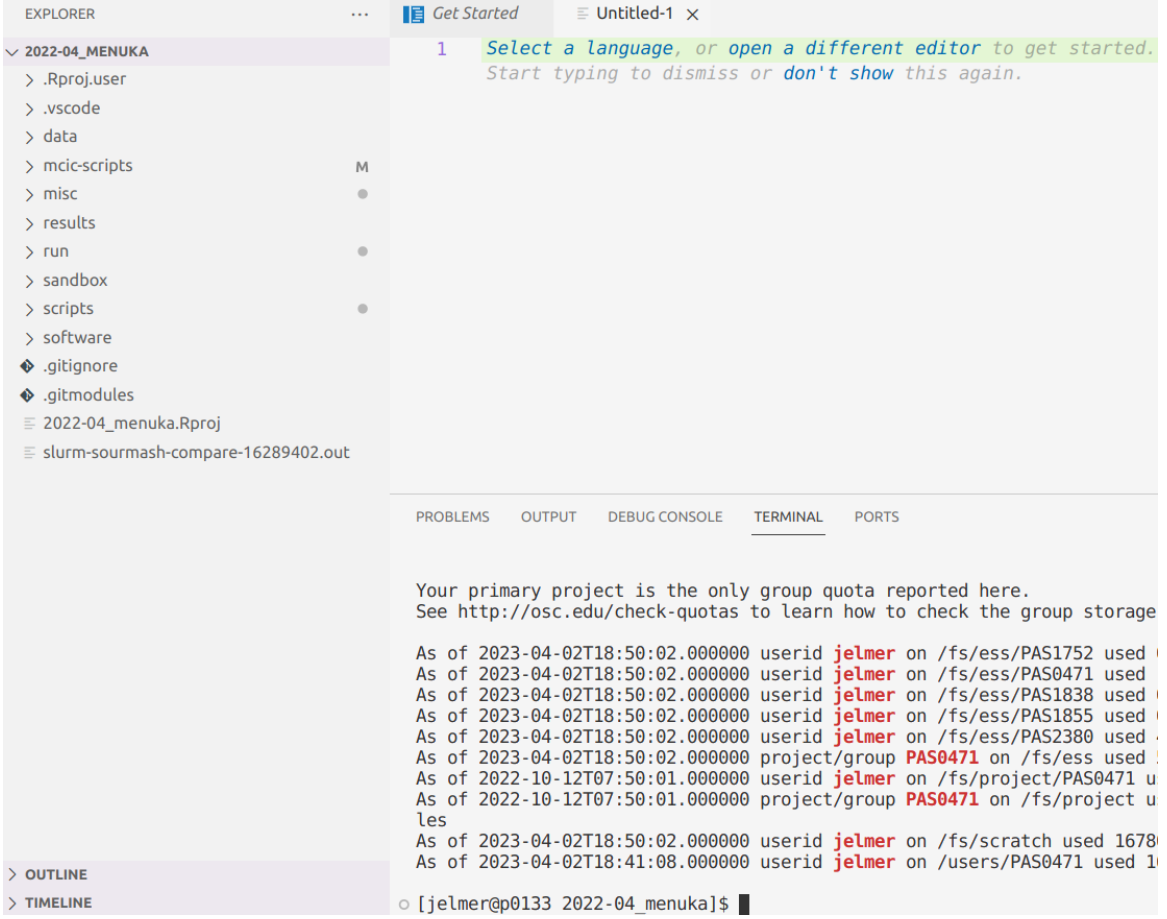
Whenever you open VS Code, a editor pane tab with a `Get Started` document is automatically opened. We can use this document to open a new text file by clicking `New file` below `Start`, which opens as a second tab in the editor pane.



VS Code Basics: Terminal

Open a terminal with a Unix shell by clicking ☰ => `Terminal` => `New Terminal`.

The great thing with this setup is that we can keep notes and write shell scripts in the same window as our shell and a file browser!



The whole game: building on the FastQC example

I've shown you the main pieces of the computational infrastructure for "**command-line genomics**". We've seen a very basic example of loading and running a command-line tool at OSC.

The missing pieces for a fuller example of how such tools are run in the context of an actual genomics project are (if we stay with FastQC):

The whole game: building on the FastQC example

I've shown you the main pieces of the computational infrastructure for "**command-line genomics**". We've seen a very basic example of loading and running a command-line tool at OSC.

The missing pieces for a fuller example of how such tools are run in the context of an actual genomics project are (if we stay with FastQC):

- Putting the command to run FastQC in a **"shell script"**.
  (This used the same language (Bash) as the commands you'd type interactively, so at its most basic this involves pasting those commands into a text file.)

The whole game: building on the FastQC example

I've shown you the main pieces of the computational infrastructure for "**command-line genomics**". We've seen a very basic example of loading and running a command-line tool at OSC.

The missing pieces for a fuller example of how such tools are run in the context of an actual genomics project are (if we stay with FastQC):

- Putting the command to run FastQC in a **"shell script"**.
  (This used the same language (Bash) as the commands you'd type interactively, so at its most basic this involves pasting those commands into a text file.)
- **Submitting the script** to the SLURM scheduler queue as a "batch job".
  (At its most basic, this involves putting `sbatch` in front of the script name.)

The whole game: building on the FastQC example

I've shown you the main pieces of the computational infrastructure for "**command-line genomics**". We've seen a very basic example of loading and running a command-line tool at OSC.

The missing pieces for a fuller example of how such tools are run in the context of an actual genomics project are (if we stay with FastQC):

- Putting the command to run FastQC in a **"shell script"**.
  (This used the same language (Bash) as the commands you'd type interactively, so at its most basic this involves pasting those commands into a text file.)
- **Submitting the script** to the SLURM scheduler queue as a "batch job".
  (At its most basic, this involves putting `sbatch` in front of the script name.)
- To make use of the capabilities of the supercomputer and speeding up our analysis, we can **submit multiple jobs *in parallel* using a loop**.

So what computational skills should I learn?

So what computational skills should I learn?

- **The core skills**:

  → Unix shell basics – the commonly used commands

  →Some shell *scripting* basics

  → SLURM basics to submit and manage your batch jobs

  → R for “downstream”, statistical and visualization tasks

So what computational skills should I learn?

- **The core skills**:

  → Unix shell basics – the commonly used commands

  →Some shell *scripting* basics

  → SLURM basics to submit and manage your batch jobs

  → R for "downstream", statistical and visualization tasks

- **When you start doing genomics projects more often**:

  → Using `conda` or containers for software

  → Unix data tools (`grep`, `sed`, `awk`, etc)

So what computational skills should I learn?

- **The core skills**:

  → Unix shell basics – the commonly used commands

  →Some shell *scripting* basics

  → SLURM basics to submit and manage your batch jobs

  → R for "downstream", statistical and visualization tasks

- **When you start doing genomics projects more often**:

  → Using `conda` or containers for software

  → Unix data tools (`grep`, `sed`, `awk`, etc)

- **When you want to become proficient in applied bioinformatics**:

  → Version control with `git`

  → More advanced: formal workflow/pipeline management tools (e.g. `Nextflow`)

  → More advanced: Python (or advanced R) for custom data processing

Resources for further learning: OSC

- Tutorials and courses!
  - https://khill42.github.io/OSC_IntroHPC/
  - https://osc.catalog.instructure.com/

- OSC regularly has online introductory sessions, both overviews and more hands-on sessions – see the OSC Events page: https://www.osc.edu/events.
  They also have weekly office hours.

- There is also some good introductory material at their *Getting Started* pages (https://www.osc.edu/resources/getting_started), as well as more specific *"HOWTO" pages* (https://www.osc.edu/resources/getting_started/howto).

Resources for further learning: OSU

- OSU courses and workshops
  - Jonathan Fresnedo Ramirez's "Genome Analytics" course (`HCS 7004`)
  - Microbiome Informatics (`MICRBIO 8161`)
  - The online materials for the workshop "Command line basics for genomic analysis at OSC" that myself and Mike Sovic gave last August: https://mcic-osu.github.io/cl-workshop-22/
  - I have a course "Practical Computing Skills for Omics Data" (`PLNTPTH 5006`) that I am planning to teach in in Spring 2024. All materials for the 2021 version of this course ("Practical Computing Skills for Biologists") are at: https://mcic-osu.github.io/pracs-sp21/

Resources for further learning: OSU

Some particularly useful books:

- The Linux Command Line (William Shotts, 2019)
- Bioinformatics Data Skills (Vince Buffalo, 2015)
- Computing Skills for Biologists: A Toolbox (Wilmes & Allesino, 2019)
- A Primer for Computational Biology (Shawn T. O' Neil, 2019)
  https://open.oregonstate.education/computationalbiology/

MCIC bioinformatics assistance

MCIC bioinformatics assistance

- I mainly provide **research support** — mostly to grad students and postdocs.

  I work on many different types of projects, and by far the most common are *RNAseq* and *microbial metabarcoding*.

MCIC bioinformatics assistance

- I mainly provide **research support** — mostly to grad students and postdocs.

  I work on many different types of projects, and by far the most common are *RNAseq* and *microbial metabarcoding*.

- I typically **help you do your analysis** rather than do it for you. I also help with getting people up to speed with the above-mentioned skills.

  In some cases, I do run analyses – for instance, with RNAseq and metabarcoding I can run a **standardized, nearly automated workflow** to get count data such that you can skip the "command-line genomics" part and work with R on your laptop.

MCIC bioinformatics assistance

- I mainly provide **research support** — mostly to grad students and postdocs.

  I work on many different types of projects, and by far the most common are *RNAseq* and *microbial metabarcoding*.

- I typically **help you do your analysis** rather than do it for you. I also help with getting people up to speed with the above-mentioned skills.

  In some cases, I do run analyses – for instance, with RNAseq and metabarcoding I can run a **standardized, nearly automated workflow** to get count data such that you can skip the "command-line genomics" part and work with R on your laptop.

- What the MCIC gets in return is a *bioinformatics membership fee* and in case of significant contributions, *authorship*.

MCIC bioinformatics assistance

- I mainly provide **research support** — mostly to grad students and postdocs.

  I work on many different types of projects, and by far the most common are *RNAseq* and *microbial metabarcoding*.

- I typically **help you do your analysis** rather than do it for you. I also help with getting people up to speed with the above-mentioned skills.

  In some cases, I do run analyses – for instance, with RNAseq and metabarcoding I can run a **standardized, nearly automated workflow** to get count data such that you can skip the "command-line genomics" part and work with R on your laptop.

- What the MCIC gets in return is a *bioinformatics membership fee* and in case of significant contributions, *authorship*.

- You can contact me at poelstra.1@osu.edu – though I will be away most of the rest of this month.

Any questions?

Bonus: Submitting FastQC batch jobs

Due to time constraints, I have skipped over the details of these steps. But an example shell script to run FastQC can be found at `/fs/scratch/PAS2250/scripts/fastqc.sh`, which contains the following code:

```bash
#!/bin/bash
#SBATCH --account=PAS2250

# Load the software
module load fastqc

# Bash strict settings
set -euo pipefail

# Copy the placeholder variables
input_file=$1
output_dir=$2

# Create the output dir if needed
mkdir -p "$output_dir"

# Run FastQC
fastqc --outdir="$output_dir" "$input_file"
```

- Indicate it's a Bash script and tell SLURM which OSC account to use
- Strict settings make the script stop on failure
- The script takes "arguments", which are stored as placeholder variables `$1` and `$2`. This allows us to run the script for different files

Bonus: Submitting FastQC batch jobs

Here is how I would submit that script as a batch job to analyze one FASTQ file:

```
fastq_file=/fs/scratch/PAS2250/data/sample1.fastq.gz
sbatch /fs/scratch/PAS2250/scripts/fastqc.sh "$fastq_file" results_jelmer
#> Submitted batch job 16323144
```

And how you can loop over all FASTQ files to submit as many jobs in parallel as you have FASTQ files:

```
for fastq_file in /fs/scratch/PAS2250/data/*.fastq.gz; do
    sbatch /fs/scratch/PAS2250/scripts/fastqc.sh "$fastq_file" results_jelmer
done
#> Submitted batch job 16323145
#> Submitted batch job 16323146
#> Submitted batch job 16323147
#> Submitted batch job 16323148
#> Submitted batch job 16323149
#> Submitted batch job 16323110
#> Submitted batch job 16323111
#> Submitted batch job 16323112
```