junaid — -bash — 136×49

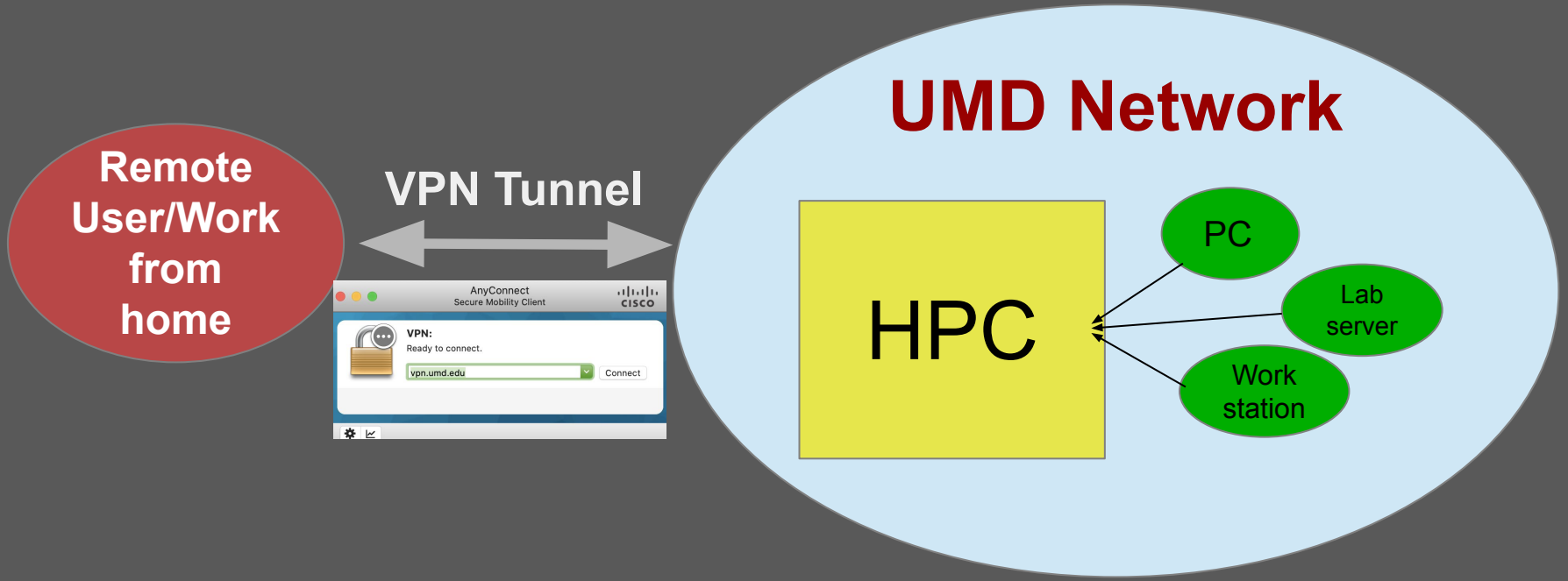
```
DingoMach:~ junaid$
```

Getting Started: Shelling into a server & copying files

Outline

- Access UMD network from home
- Launching the terminal/command prompt
- Connecting to your HPC
- Copying files to/from HPC
- Introducing SLURM job scheduling software

Accessing your HPC: get on UMD Network through VPN



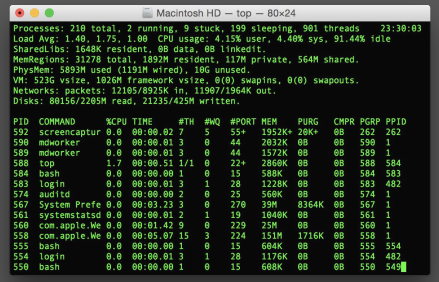
Accessing UMD Network

Virtual Private Network (VPN) allows you to work from off campus

- Secure connection from your computer at home to UMD network, with all the access you would expect from actually being on-site!

https://terpware.umd.edu/Windows/title/1840

https://it.umd.edu/spotlight/connections/what-vpn



Accessing your HPC: Using the terminal

Terminal AKA Command line is a non-graphical interface to use your computer

- Mac has Terminal pre-installed in Utilities folder in Applications
- Windows has Command Prompt or CMD, but requires some set up to connect to HPC
  - Can use PowerShell or 3rd party App MobaXterm
  - https://mobaxterm.mobatek.net/
- Linux has Terminal pre-installed

Getting started: connecting to your server

Once you have an account, you can connect to your HPC or “shell in” through the terminal using the following syntax:

```
ssh <user name>@<server address>
```

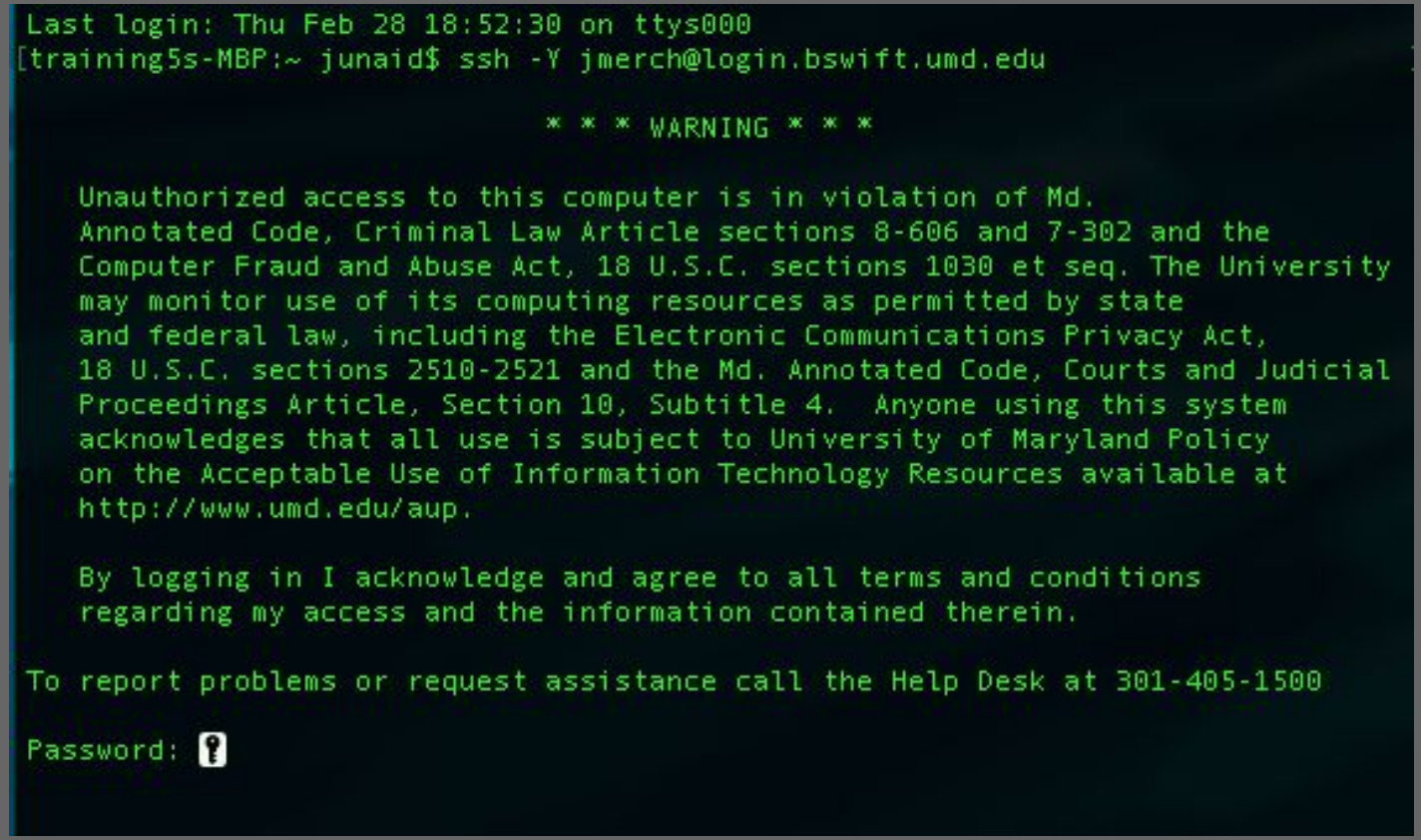
```
ssh jmerch@login.bswift.umd.edu
```

Enter your password when prompted (you won’t see the cursor move!!)

Now you’re on the head node of the HPC, where you can run basic commands, explore the directory structure, and submit jobs

Do NOT run big jobs on the head node

Getting started: connecting to your server



Getting started: copying files to HPC

Here are some ways to move data between your local computer and server:

scp – secure copy; copy files to/from the HPC server.

From local to HPC:

```
scp /local/path/file.ext <user name>@<server address>:/path/to/folder
```

```
scp /Users/junaid/IHeartHpc.txt jmerch@login.bswift.umd.edu:/data/dir
```

From HPC to local:

```
scp <user name>@<server address>:/path/to/file /local/path/directory
```

```
scp jmerch@login.bswift.umd.edu:/data/Results.txt /Users/junaid/Desktop
```

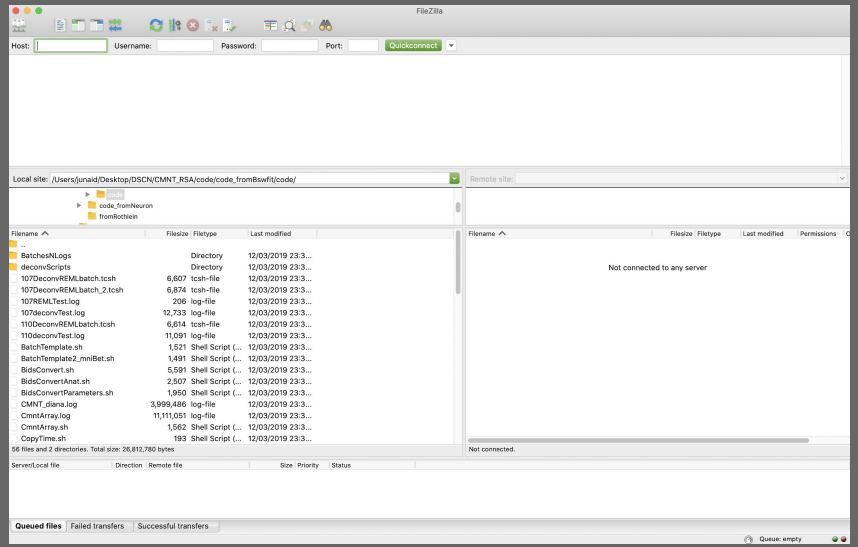
To copy entire directories, you can use the -r option after the scp command

More info: https://linuxize.com/post/how-to-use-scp-command-to-securely-transfer-files/

Getting started: copying files to/from HPC

You can use FileZilla for a graphical interface transfer:
https://filezilla-project.org/

I've occasionally gotten corrupted files using FileZilla.



slurm
workload manager

SLURM

Simple Linux Utility for Resource Management (SLURM)

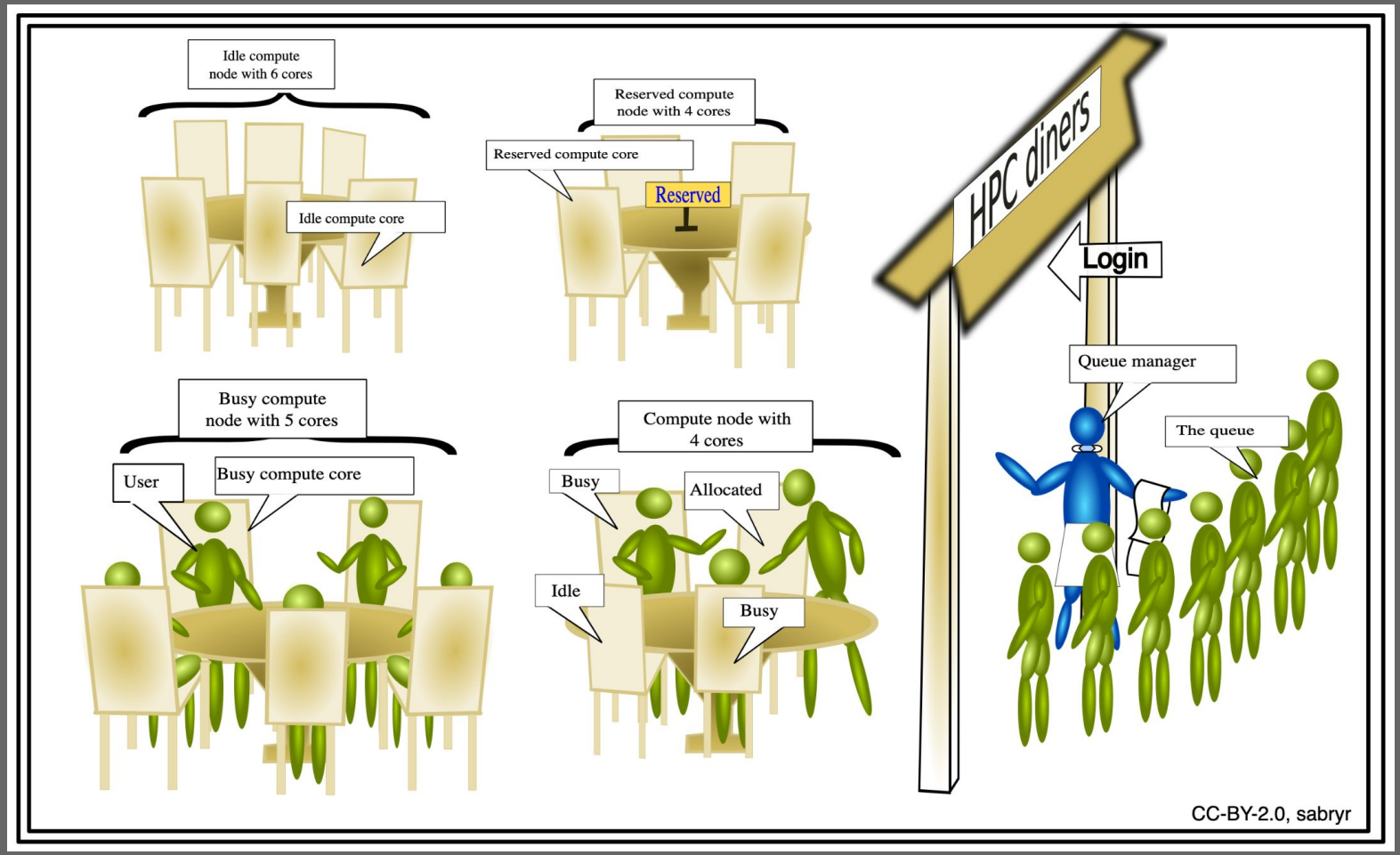
Workload manager, job scheduler, & queuing system for running processes on an HPC

Figures out what specifications you need, finds the available node that meets the requirements, and allocates your job to that node!

Runs multiple jobs in parallel at once.

Use in conjunction with the job script you want to run

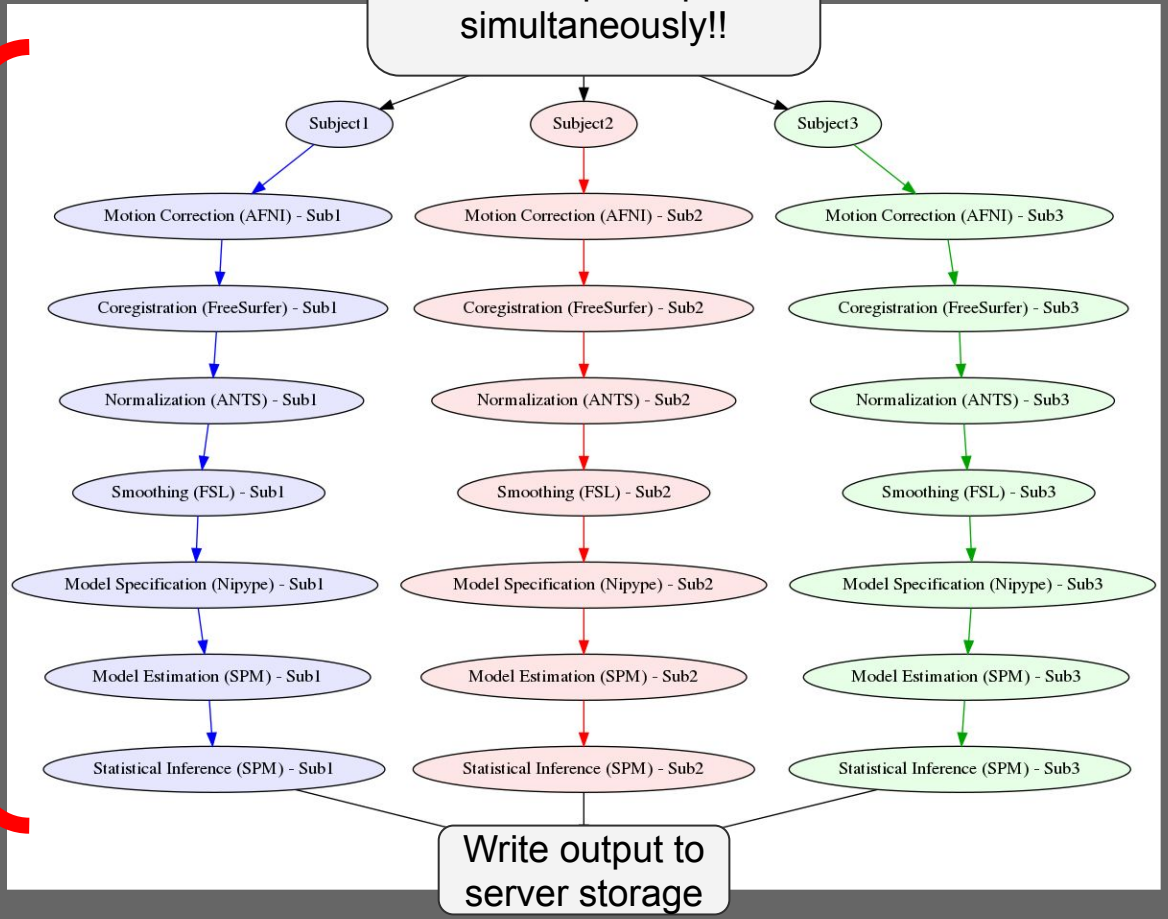
https://slurm.schedmd.com/quickstart.html



https://sabryr.github.io/hpc-intro/13-scheduler/index.html

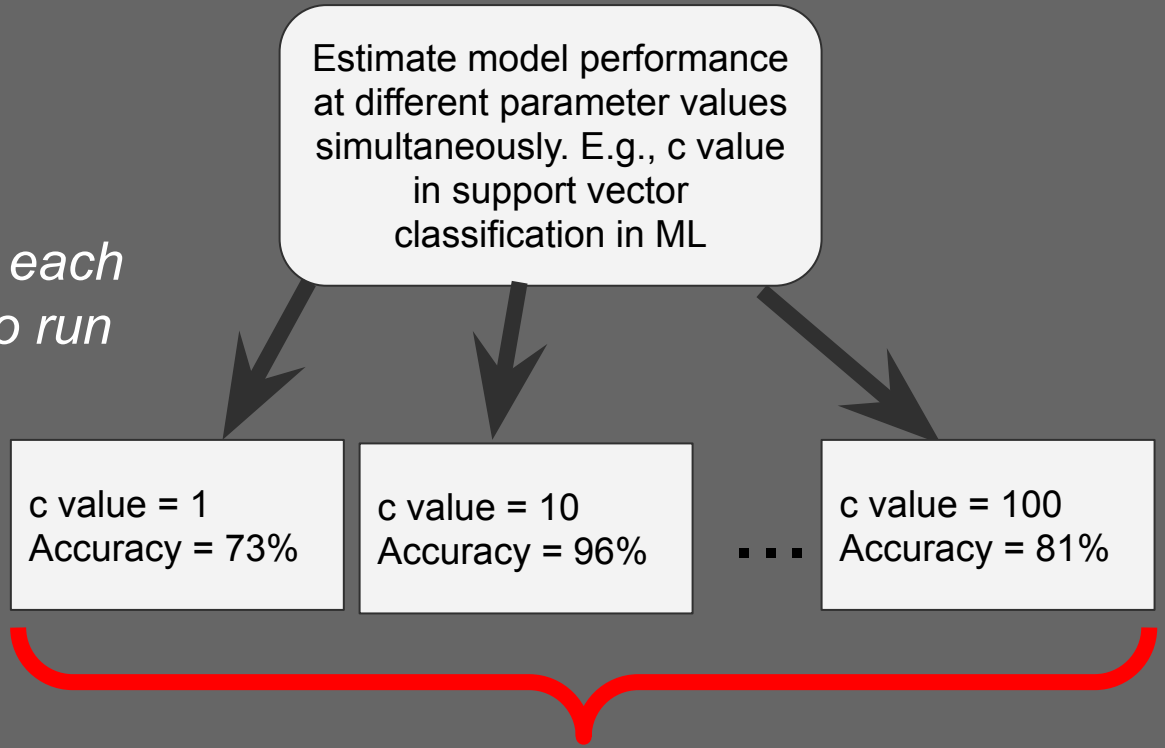
Neuroimaging Example:

The data processing pipeline for each participant can be a series of data analysis steps



Parameter Tuning Example

Remember that each has to be able to run independently!



After running the model at all values of parameter, you can compare to choose optimal value

SLURM Commands: sbatch

sbatch – probably the most important thing in this entire presentation

Allows you to submit a job to the appropriate node

Use with job script for whatever sort of process you want to run

Things you absolutely MUST specify when using sbatch to submit a job:

- # of Nodes
- Amount of RAM
- Time to process the job

You can specify a lot of other things: https://slurm.schedmd.com/sbatch.html

SLURM Commands: sbatch

sbatch https://slurm.schedmd.com/sbatch.html

Usage: `sbatch [options] <job script>`

For example: `sbatch RunSimulation.sh`

This will give you a job number that you can check in on later

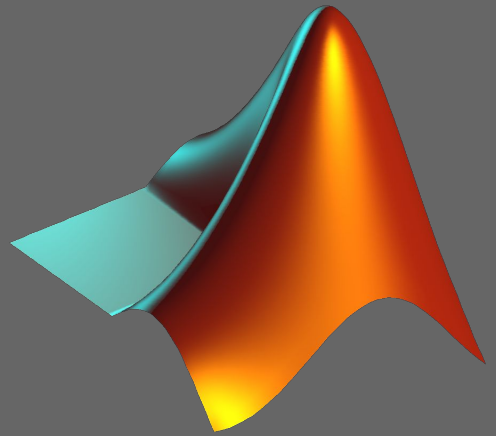
```
[login-1:~: sbatch example.sh
 Submitted batch job 82581
 login-1:~:
```

Any questions so far?

Things are about rev up really quickly in the next presentation...

Presentation 5:

How to write & submit code for HPC w/examples in MATLAB & R



Outline

- Basic approach of writing code for HPC
- SLURM options
- A MATLAB example with introduction to wrappers
- An example of an R script (translates to Python)

Write stand-alone scripts

The easiest way to implement code on a HPC is to write stand alone code in whatever programming language you like and create a “wrapper” script for it

A wrapper script is basically a way to tell the shell to launch a computing node, and run whatever script you have set it to run

The wrapper function contains syntax/options that are specific to the job submission software you are using, i.e., SLURM

And gives instructions that are specific to the programming language you are using

Important considerations

The script that you write (in whatever programming language) has to be able to be run non-interactively.

- You should be able to run it without having to do anything once you start.
- It can take an input, but once it has started, you cannot interact with it

No graphical interfaces/GUIs and you cannot generate figures!

- When you submit a job, it is running on a node with no graphics potential

Solution/output of your code either needs to printed to screen or save to file

SLURM Job Scripts

Before going too much further, we should look inside a job script

```
#!/bin/bash
#
#
#SBATCH --time=144:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=sub-JAM014
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
module load singularity
#
echo Starting fMRIprep at:
echo working on sub-JAM014
date
echo ----------------------------------------------------------------------
```

SLURM Job Scripts

Before going too much further, we should look inside a job script

```
#!/bin/bash
#
#
#SBATCH --time=144:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=sub-JAM014
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
module load singularity
#
echo Starting fMRIprep at:
echo working on sub-JAM014
date
echo ----------
```

First line is called the ‘shebang’ and indicates what type of script it is.
In this case it is bash code, which is very general purpose

You can do python this way:
#!/bin/python

Or R:
#!/usr/bin/env Rscript
Matlab is a little different (more on this later)

SLURM Job Scripts

Before going too much further, we should look inside a job script

```
#!/bin/bash
#
#
#SBATCH --time=144:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=sub-JAM014
#SBATCH --mail-user=jmerch@terp[illegible]ail.umd.edu
#SBATCH --mail-type=ALL
#
module load singularity
#
echo Starting fMRIprep at:
echo working on sub-JAM014
date
echo ----------
```

The next few lines are sbatch options. These options can be specified within the job script, or outside of the script as I described before:

```
sbatch [options] <job script>
```

Required:

-time is the amount of time you want allocated in hours:minutes:seconds

-nodes is # of nodes

-mem is amount of RAM in megabytes

Optional:

-output - name of the output log file that contains what would be printed to terminal and any errors that occurred

-mail - give your umd email so it sends a message when the job starts and ends!

SLURM Job Scripts

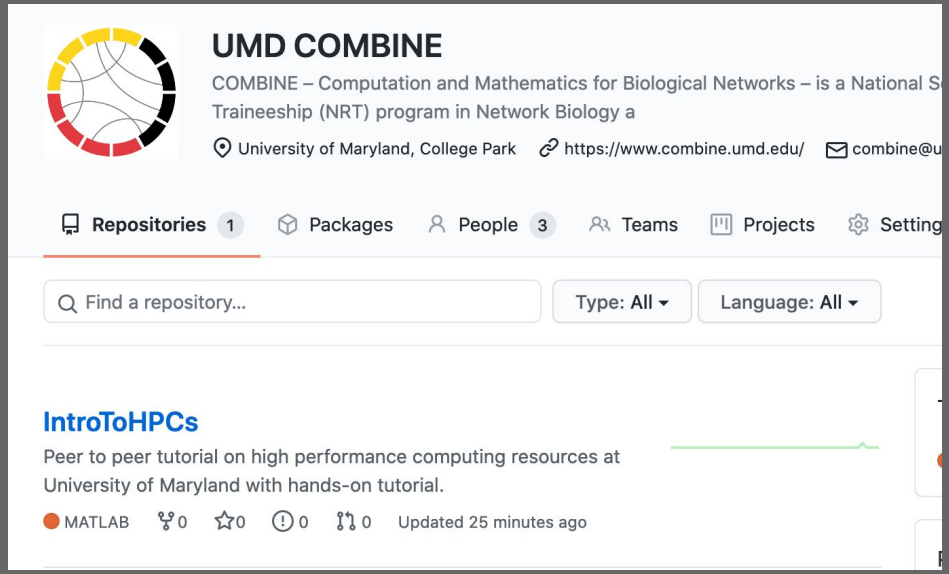
Before going too much further, we should look inside a job script

```
#!/bin/bash
#
#
#SBATCH --time=144:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=sub-JAM014
#SBATCH --mail-user=jmerch@terpmail.umd.ed
#SBATCH --mail-type=ALL
#
module load singularity
#
echo Starting fMRIprep at:
echo working on sub-JAM014
date
echo ----------
```

The remainder of the script is what you want it to do! If you are wanting to load any programs, do that in the first few lines, and have at it!

All the example code that I will show for the remainder of the time (and more!) can be found on the github repository we sent out:

https://github.com/UMD-COMBINE/IntroToHPCs

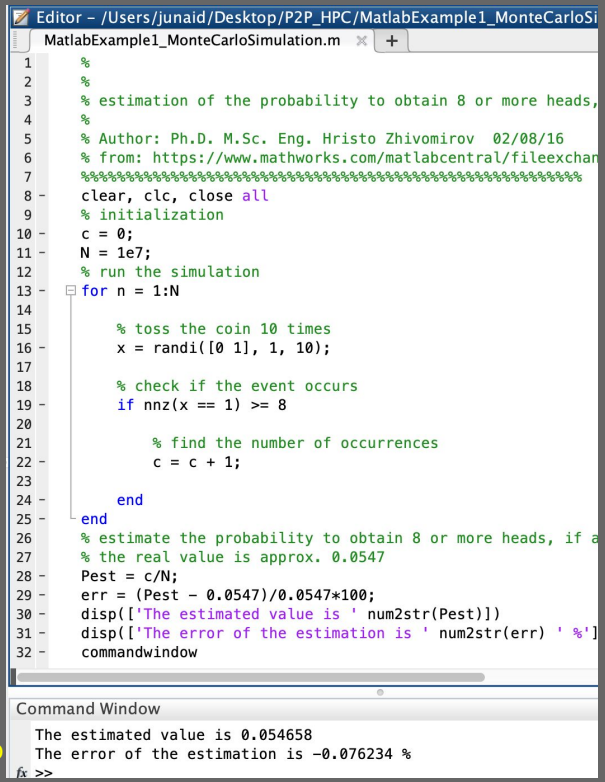


Matlab example: Monte Carlo simulation for estimation of the probability to obtain 8 or more heads, if a coin is tossed 10 times.

1) Write the standalone matlab script as you would run on your local workstation, and it outputs the solution in the command window.

2) Copy it up to the server

3) Write a wrapper script to submit it to a node using sbatch



This example can be found at:
https://github.com/UMD-COMBINE/IntroToHPCs/blob/main/MatlabExample1_MonteCarloSimulation.m

Code from:
https://www.mathworks.com/matlabcentral/fileexchange/55306-monte-carlo-estimation-examples-with-matlab

Matlab example: Monte Carlo simulation for estimation of the probability to obtain 8 or more heads, if a coin is tossed 10 times.

```
clear, clc, close all
% initialization
c = 0;
N = 1e7;
% run the simulation
for n = 1:N
% toss the coin 10 times
x = randi([0 1], 1, 10);
% check if the event occurs
if nnz(x == 1) >= 8
% find the number of occurrences
c = c + 1;
end
end
% estimate the probability to obtain 8 or more heads, if a coin is tossed 10 times
% the real value is approx. 0.0547
Pest = c/N;
err = (Pest - 0.0547)/0.0547*100;
disp(['The estimated value is ' num2str(Pest)])
disp(['The error of the estimation is ' num2str(err) ' %'])
commandwindow
```

SLURM Job Scripts: MATLAB Wrapper

Now, let's look at the special case of creating a job script with a MATLAB process.

For this, you want to create a bash script as before

```
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=MatlabExample1.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
# This is the simplest way of running a single piece of matlab code. You would submit this script,
# which will launch matlab and run the accompanying matlab code for an Ising model.
# This approach is not great for running things in parallel, but gives you an idea of how job submission
# works.
#
# to run this on a compute node:
# > sbatch /path/to/MatlabExample1_Ising_wrapper.sh
#
# load the matlab module
module load matlab
matlab -nodesktop -nodisplay -nosplash -r "clear; run /data/bswift-1/jmerch/P2P/MatlabExample1_MonteCarloSimulation.m ; exit"
```

SLURM Job Scripts: MATLAB Wrapper

```
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=MatlabExample1.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
# This is the simplest way of running a single piece of matlab code. You would submit this script,
# which will launch matlab and run the accompanying matlab code for an Ising model.
# This approach is not great for running things in parallel, but gives you an idea of how job submission
# works.
#
# to run this on a compute node:
# > sbatch /path/to/MatlabExample1_Ising_wrapper.sh
#
# load the matlab module
module load matlab
matlab -nodesktop -nodisplay -nosplash -r "clear; run /data/bswift-1/jmerch/P2P/MatlabExample1_MonteCarloSimulation.m ; exit"
```

First, make sure you have all SLURM options set

SLURM Job Scripts: MATLAB Wrapper

```
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=MatlabExample1.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
# This is the simplest way of running a single piec
# which will launch matlab and run the accompanying
# This approach is not great for running things in parallel, but gives you an idea of how job submission
# works.
#
# to run this on a compute node:
# > sbatch /path/to/MatlabExample1_Ising_wrapper.sh
#
# load the matlab module
module load matlab
matlab -nodesktop -nodisplay -nosplash -r "clear; run /data/bswift-1/jmerch/P2P/MatlabExample1_MonteCarloSimulation.m ; exit"
```

First, load matlab using module commands. More on modules and available software: https://www.glue.umd.edu/hpcc/help/software.html

SLURM Job Scripts: MATLAB Wrapper

```
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=MatlabExample1.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
# This is the simplest way of running a single piec
# which will launch matlab and run the accompanying
# This approach is not great for running things in
# works.
#
# to run this on a compute node:
# > sbatch /path/to/MatlabExample1_Ising_wrapper.sh
#
# load the matlab module
module load matlab
matlab -nodesktop -nodisplay -nosplash -r "clear; run /data/bswift-1/jmerch/P2P/MatlabExample1_MonteCarloSimulation.m ; exit"
```

Next, you want to start matlab using the nodisplay options because otherwise it will try to launch the matlab window and crash because UMD HPCs have no graphics abilities

SLURM Job Scripts: MATLAB Wrapper

```
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=M
#SBATCH --mail-use
#SBATCH --mail-typ
#
# This is the simp
# which will laun
# This approach is
# works.
#
# to run this on a
# > sbatch /path/to/MatlabExample1_Ising_wrapper.sh
#
# load the matlab module
module load matlab
matlab -nodesktop -nodisplay -nosplash -r "clear; run /data/bswift-1/jmerch/P2P/MatlabExample1_MonteCarloSimulation.m ; exit"
```

Finally, you want to feed into the matlab command everything you want to do within matlab using the -r option, followed by all the matlab operations within quotes. This can be tricky if you're used to using matlab interactively, so I recommend making sure you have a set of working matlab that you test outside of a job submission before starting this.

SLURM Job Scripts: MATLAB Wrapper

```
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=MatlabExample1.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
# This is the simplest way of running a single piece of matlab code. You would submit this script,
# which will launch matlab and run the accompanying matlab code for an MC simulation.
# This approach is not great for running things in parallel, but gives you an idea of how job submission
# works.
#
# to run this on a compute node:
# > sbatch /path/to/MatlabExample1_MonteCarloSimulation.m
#
# load the matlab module
module load matlab
matlab -nodesktop -nodisplay -nosplash -r "clear; run /data/bswift-1/jmerch/P2P/MatlabExample1_MonteCarloSimulation.m ; exit"
```

SLURM Job Scripts: MATLAB Wrapper

Now, to run this code, you can use the following command:

```
sbatch /path/to/MatlabExample1_wrapper.sh
```

specifically:

```
sbatch /data/bswift-1/jmerch/P2P/MatlabExample1_wrapper.sh
```

```
login-1:/data/bswift-1/jmerch/P2P: sbatch MatlabExample1_wrapper.sh
Submitted batch job 57516
```

SLURM Job Scripts: MATLAB Wrapper

Once it finishes, you can check the output file using the ‘more’ command, which captures everything that would have printed to screen:

```
more /data/bswift-1/jmerch/P2P/MatlabExample1.log
```

R example

R and Python scripts are much easier because they don't require a wrapper!

RExample1.R

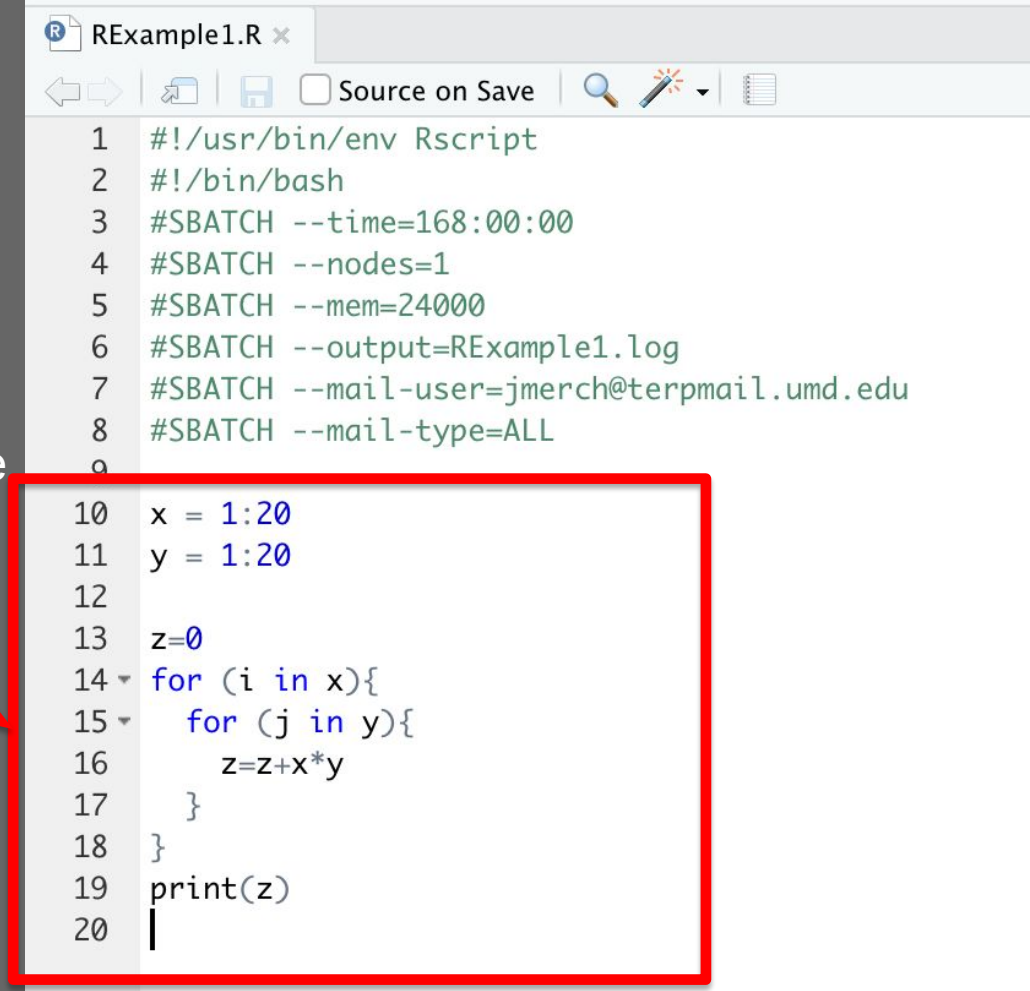
Source on Save

```
#!/usr/bin/env Rscript
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=RExample1.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL

x = 1:20
y = 1:20

z=0
for (i in x){
  for (j in y){
    z=z+x*y
  }
}
print(z)
```

R example

R and Python scripts are much easier because they don't require a wrapper!

For example, this is a set of standalone R commands



R example (will generali

R and Python scripts are much easier because they don't require a wrapper!

For example, this is a set of standalone R commands

As long as you add the shebang and SLURM options on top, this can run as-is!

RExample1.R

Source on Save

```
1  #!/usr/bin/env Rscript
2  #!/bin/bash
3  #SBATCH --time=168:00:00
4  #SBATCH --nodes=1
5  #SBATCH --mem=24000
6  #SBATCH --output=RExample1.log
7  #SBATCH --mail-user=jmerch@terpmail.umd.edu
8  #SBATCH --mail-type=ALL
9
10 x = 1:20
11 y = 1:20
12
13 z=0
14 for (i in x){
15   for (j in y){
16     z=z+x*y
17   }
18 }
19 print(z)
20
```

R example

For instance:

```
sbatch /data/bswift-1/jmerch/P2P/RExample1.R
```

```
login-1:P2P$ sbatch RExample1.R
Submitted batch job 57529
```

RExample1.R

Source on Save

```
#!/usr/bin/env Rscript
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=RExample1.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL

x = 1:20
y = 1:20

z=0
for (i in x){
  for (j in y){
    z=z+x*y
  }
}
print(z)
```

Then check output:

```
login-1:P2P$ more RExample1.log
 [1]     400    1600    3600    6400   10000   14400   19600   25600   32400   40000
[11]   48400   57600   67600   78400   90000  102400  115600  129600  144400  160000
login-1:P2P$
```

Python example

And the same holds true if you add the shebang and SLURM options to a Python script!

PythonExample.py

```
#!/usr/bin/env python
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --output=PythonExample.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL

# creating a simple data - set
sample = [1, 2, 3, 4, 5]
print('range of sample is: ',max(sample)-min(sample))
```

More Information and example code

https://github.com/UMD-COMBINE/IntroToHPCs

I have added additional example code that will let you really allow you to take advantage of a super computing. For example:

- A general MATLAB wrapper that will run any MATLAB script
- A wrapper for MATLAB functions that require input variables
- And an example of an Array job that will automatically run the code multiple times based on the specified input variables
  - This is how you parallelize efficiently
  - Can be used for things like parameter tuning etc

Questions?

I'm available for consulting on how to get you set up on an HPC and running code, but I might ask for some compensation (i.e., cash, food, beer etc) since I'm really busy **&** underpaid!

#AlwaysLookingForASideHustle

Email: jmerch@umd.edu

@junaidsmerchant

```
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --mem=24000
#SBATCH --array=5,10,50,100
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
# This wrapper has an array option in slurm, which means that it will run this job for each of the numbers defined
# in the array above!
#
# This is a great way to parallelize something!!
#
# We can use the same function from before, but don't give it an iteration amount
# > sbatch /data/bswift-1/jmerch/P2P/MatlabExample4_Arraywrapper.sh /data/bswift-1/jmerch/P2P/MatlabExample3_FunctionUsingInputs.m
#
# The dollar sign $ in bash indicates a variable, and the numbered variables are the order of inputs this wrapper was given
#
# load the matlab module
module load matlab
# get the full path and name of function using bash commands dirname and basename.
# These will be used to add folder to path in matlab, and give function call
Path=$(dirname $1)
Func=$(basename $1 .m)
#
# And this time, the iteration amounts will be defined by the array numbers defined in the slurm options
Num=${SLURM_ARRAY_TASK_ID}

matlab -nodesktop -nodisplay -nosplash -r "clear; addpath('$Path'); $Func $Num; exit"
```