GRACEful use of High Performance Computing

Alicia Kacharia
Aaron Wolf

October 22, 2020

Project Files

A Github repository has been created with example scripts for this manual.

It is available at github.com/aarondwolf/egc_hpc_manual.

You are welcome to download/fork the repository and use it as a code base to build off for your own work. Throughout this manual, we will be working from this file for all batch files and scripts.

Data

In the data folder I have saved auto.dta and auto.csv.

These are simply saved versions of Stata's auto dataset. They are used throughout the manual.

For completeness, the code to create them is:

```
sysuse auto, clear
save data/auto.dta, replace
export delimited using data/auto.csv, replace
```

Outline

Introduction

What is GRACE?

- Its a computer. A big one. With a lot of CPUs.
- Open to Yale staff and students (with permission, of course).
- GRACE is a shared resource: You need to wait your turn if you want to use a lot of computational power at once.

How do I use it?

- You need to log in (securely), and politely ask for resources (i.e. nodes and CPUs) to accomplish a task. You accomplish this with SLURM.
- Beyond that, its just a computer. You tell it to execute certain files, and it does so, or tells you it can't.

Wait, what's SLURM???

- We'll get to that. Essentially, SLURM is to HPC clusters what .do files are to Stata. It's how you tell a HPC cluster what you want it to do.
- Crucially, SLURM software negotiates the "queue" of people who want to use CPU time. It sorts people onto compute nodes in the most socially optimal way, given people's time and resource needs.
- You use SLURM to say "I need 32 cores for 4 hours to run these .do/.r/.py/.m files", and the software will find the first available appointment for you with Dr. Computer.

Getting Started

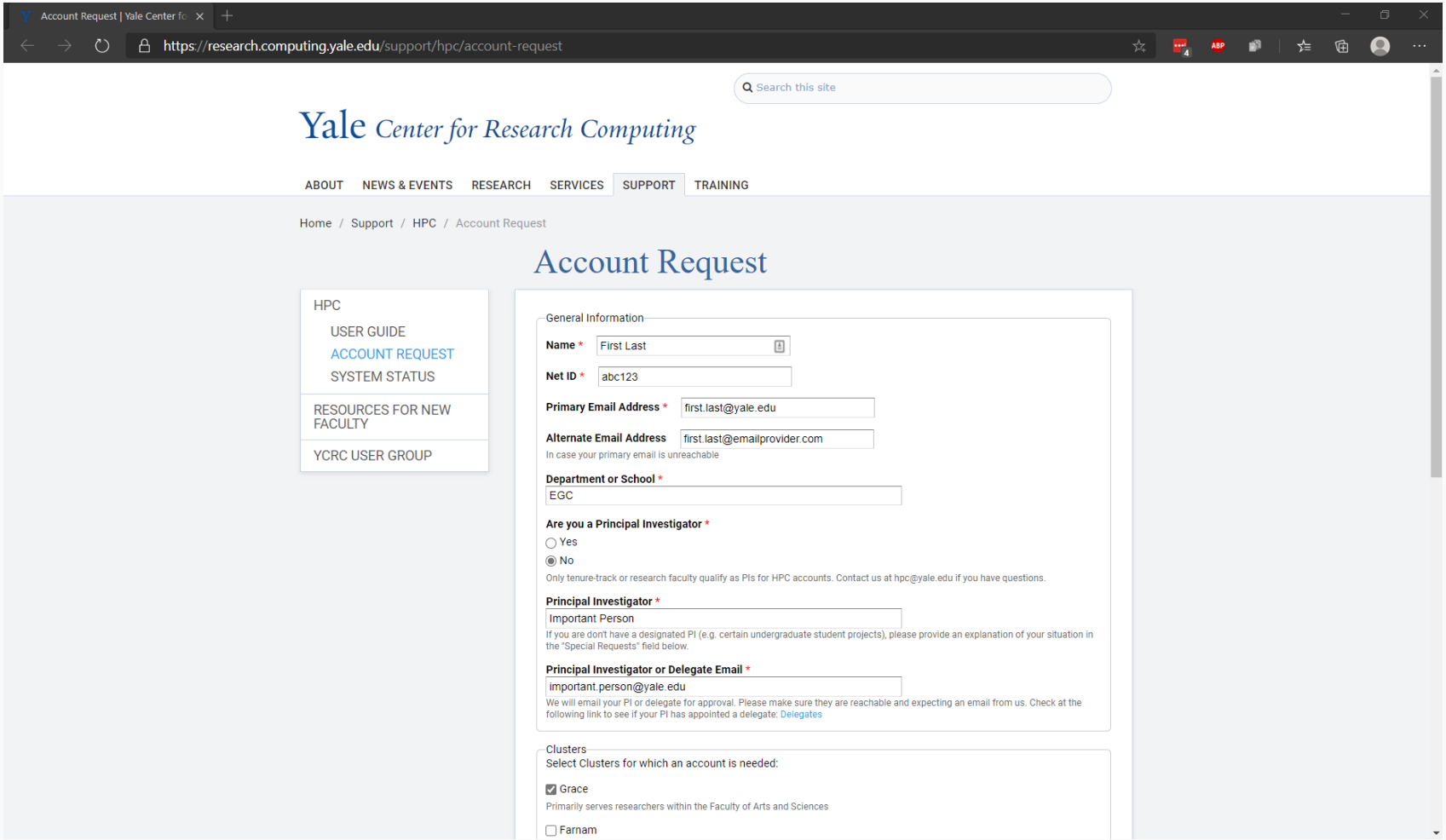
Yale's Getting Started Resources

- Yale has created a really fantastic set of materials, available at https://docs.ycrc.yale.edu/clusters-at-yale/
- Much of our *Getting Started* section will simply be providing screen shots of this process.
- The purpose of this guide is to provide more of a step-by-step guide to quickly getting started based on the kinds of tasks we are likely to do (e.g. run a Stata .do file).
- Please take a look at Yale's resources: They are great, they are just very comprehensive.

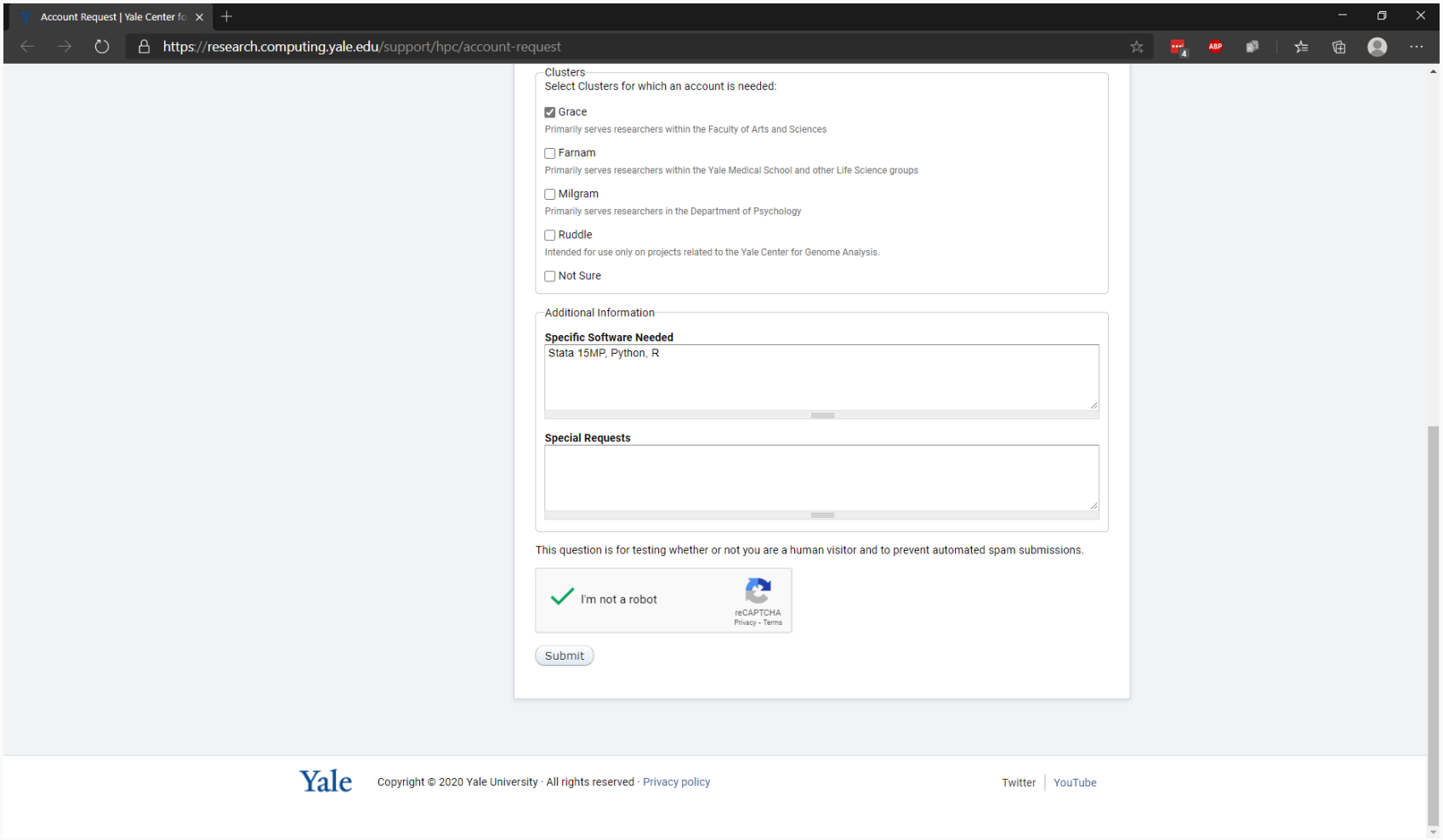
Step 1: Request an Account

- Open https://research.computing.yale.edu/support/hpc/account-request
- Fill in all details.
- Be sure to check "No" to "Are you a Principal Investigator?"
- Check "Grace"
- You can list any software you need, but it will likely already be installed (if its Stata, Python, Matlab, or R).

Step 1: Request an Account



Step 1: Request an Account



Step 1: Request an Account

- Make sure you check in with your PI to find out when they received and responded to the email: You may or may not receive an indication from Yale that you've been approved.

Step 2: Download Software (Windows)

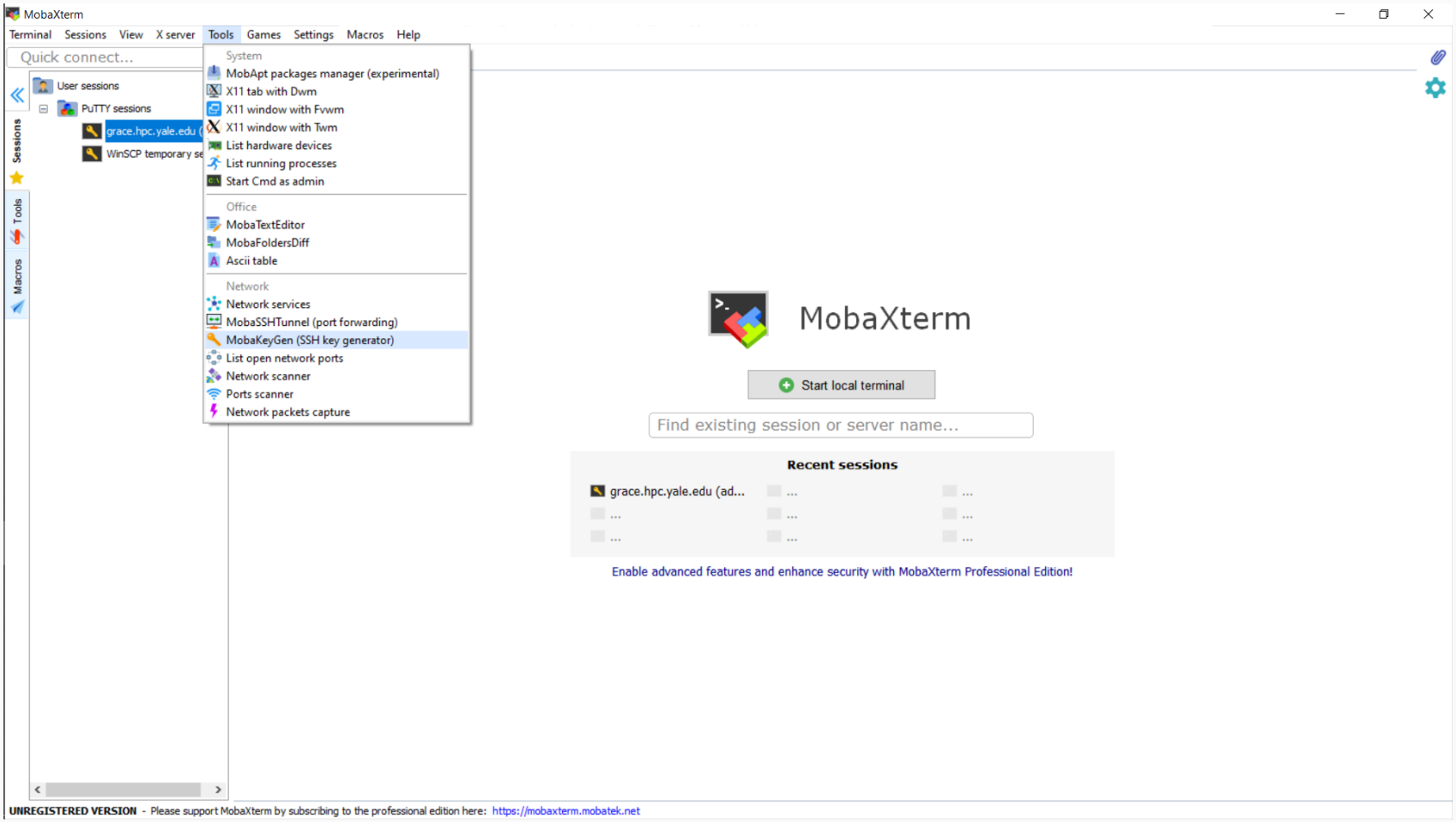
- While you are waiting for your PI to approve you, take the time to download and install all the software you will need. For Windows, these include:
  1. Cisco AnyConnect (VPN when off campus):
     https://cowles.yale.edu/faq/how-do-i-connect-yale-resouces-yale-vpn-while-campus
     We will not provide instructions for using Cisco, but the above link provides step by step instructions. **You must be on campus or connected to the VPN to be able to use Grace.**
  2. MobaXTerm (fancier shell for SLURM commands):
     https://mobaxterm.mobatek.net/
  3. WinSCP (synchronizing your folders):
     https://winscp.net/eng/download.php

Step 3: Create an SSH Key

- Your SSH key is a public-private key pair that will allow you to verify who you are when you SSH onto the cluster.
- This needs to be generated and uploaded to Grace.

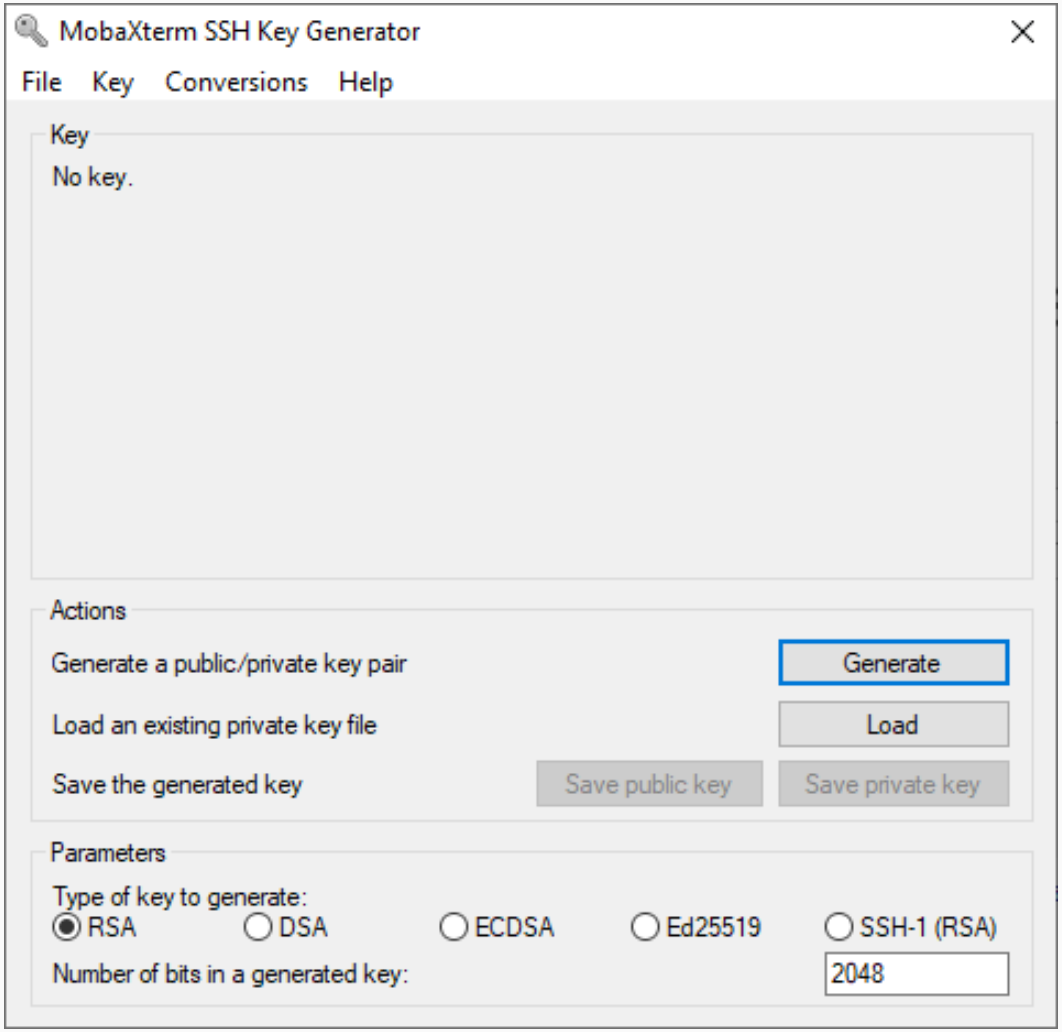
Step 3: Create an SSH Key (Using MobaXterm)

Open MobaXterm and select MobaKey Gen from the Tools menu.



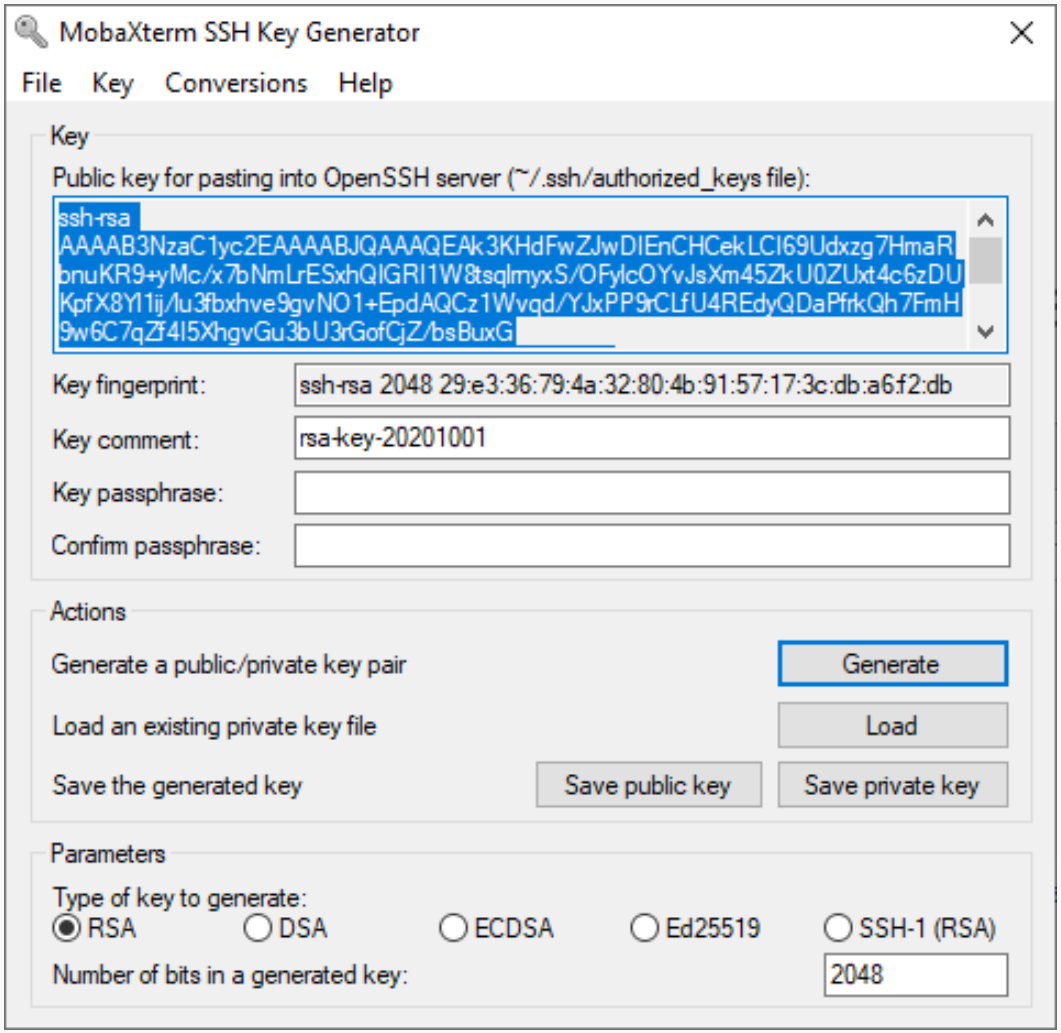
Step 3: Create an SSH Key (Using MobaXterm)

Select "Generate"



Step 3: Create an SSH Key (Using MobaXterm)

Leave all options as default. Save both the private and public keys.

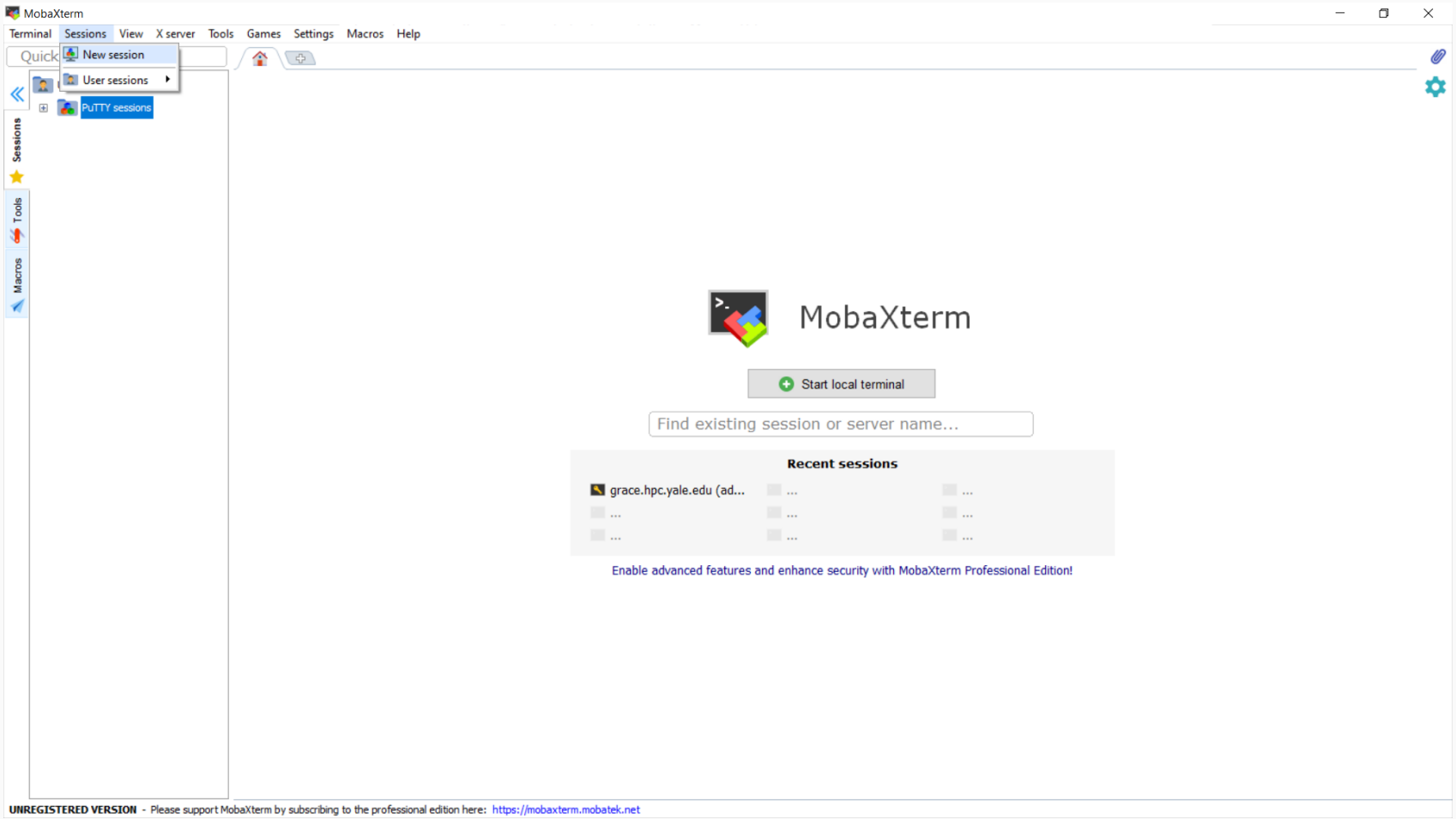


Step 4: Upload your SSH key

- Once you have been approved, navigate to http://gold.hpc.yale.internal/cgi-bin/sshkeys.py
- Choose your saved public key file (ends in .pub, not .ppk)
- Alternatively, you can copy and paste the text of the file (it does not matter what the file is called, just the random text within it).
- Wait a few minutes for the key to be sent to the clusters.

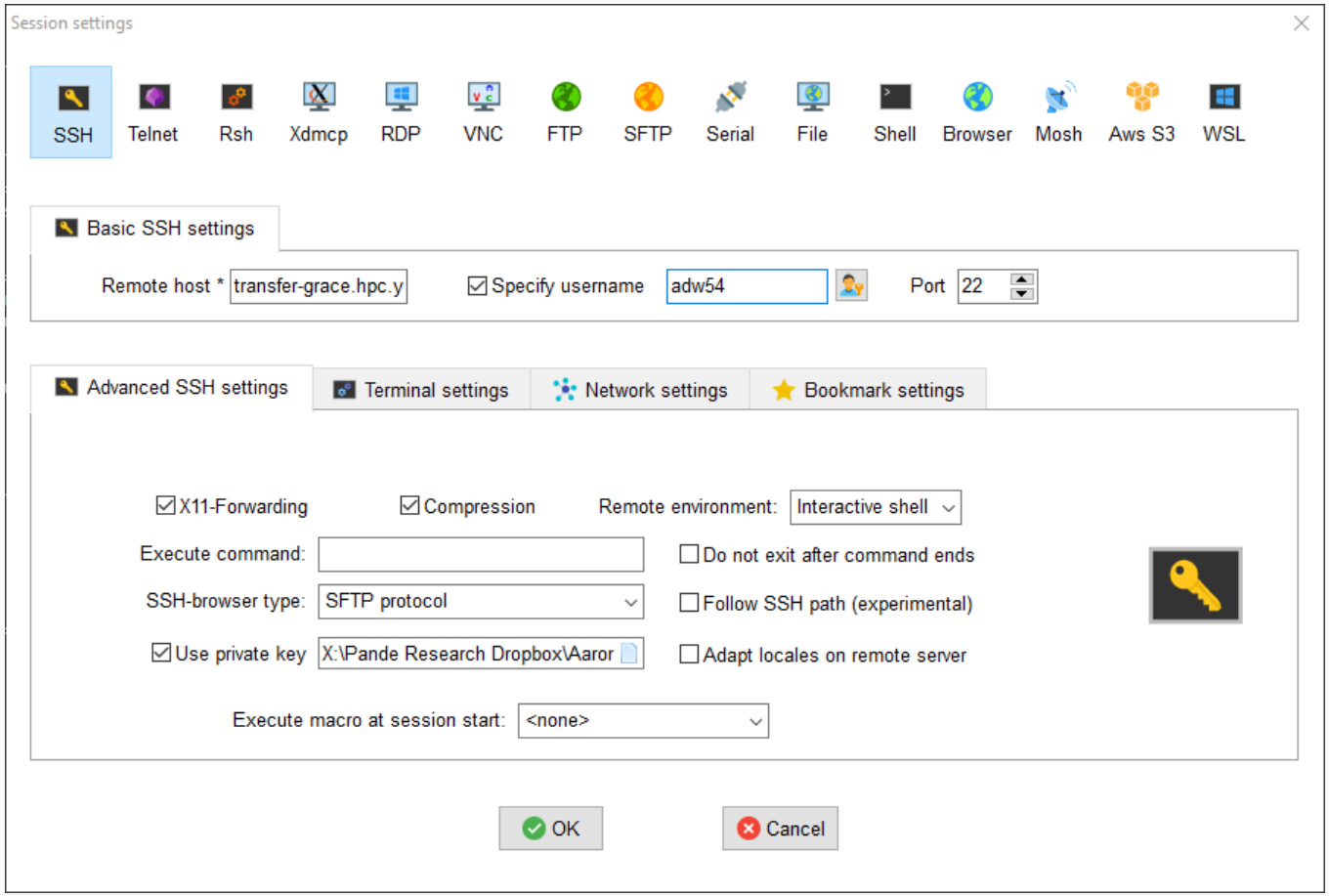
Step 5: Log into the cluster via MobaXterm

Open MobaXterm and select New Session from the Sessions menu.



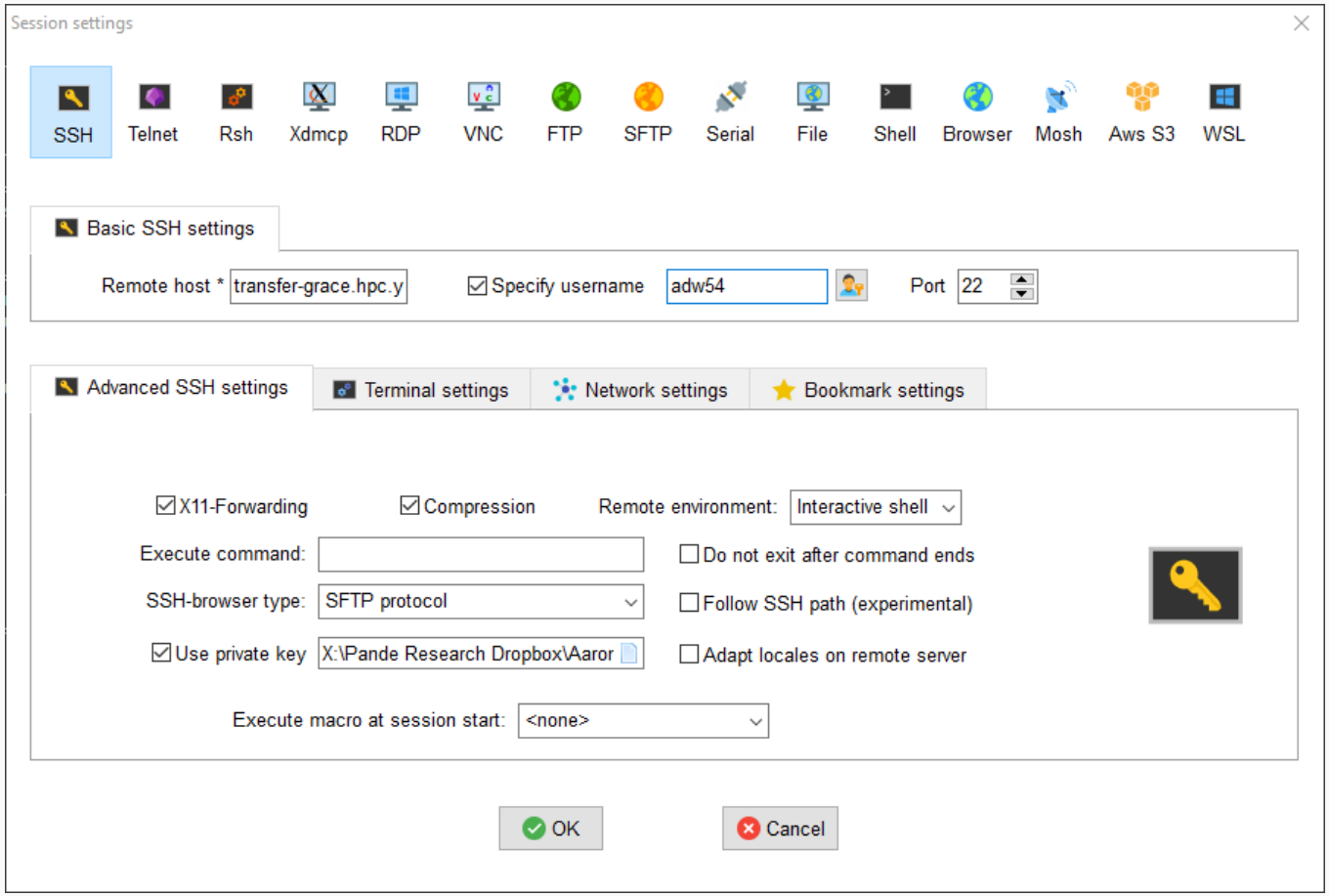
Step 5: Log into the cluster via MobaXterm

Fill in the provided fields



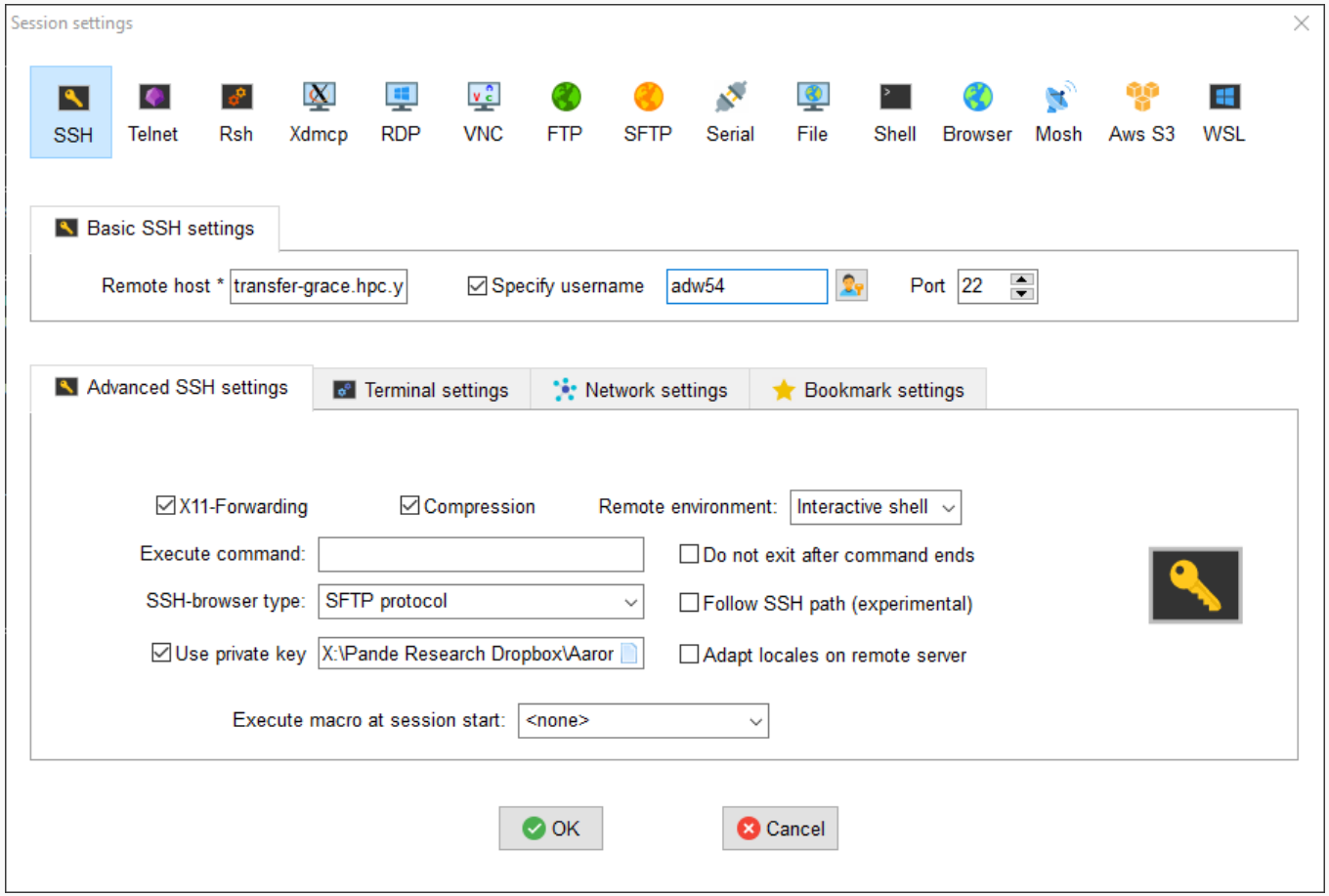
Step 5: Log into the cluster via MobaXterm

Under *Remote Host*, input "transfer-grace.hpc.yale.edu".



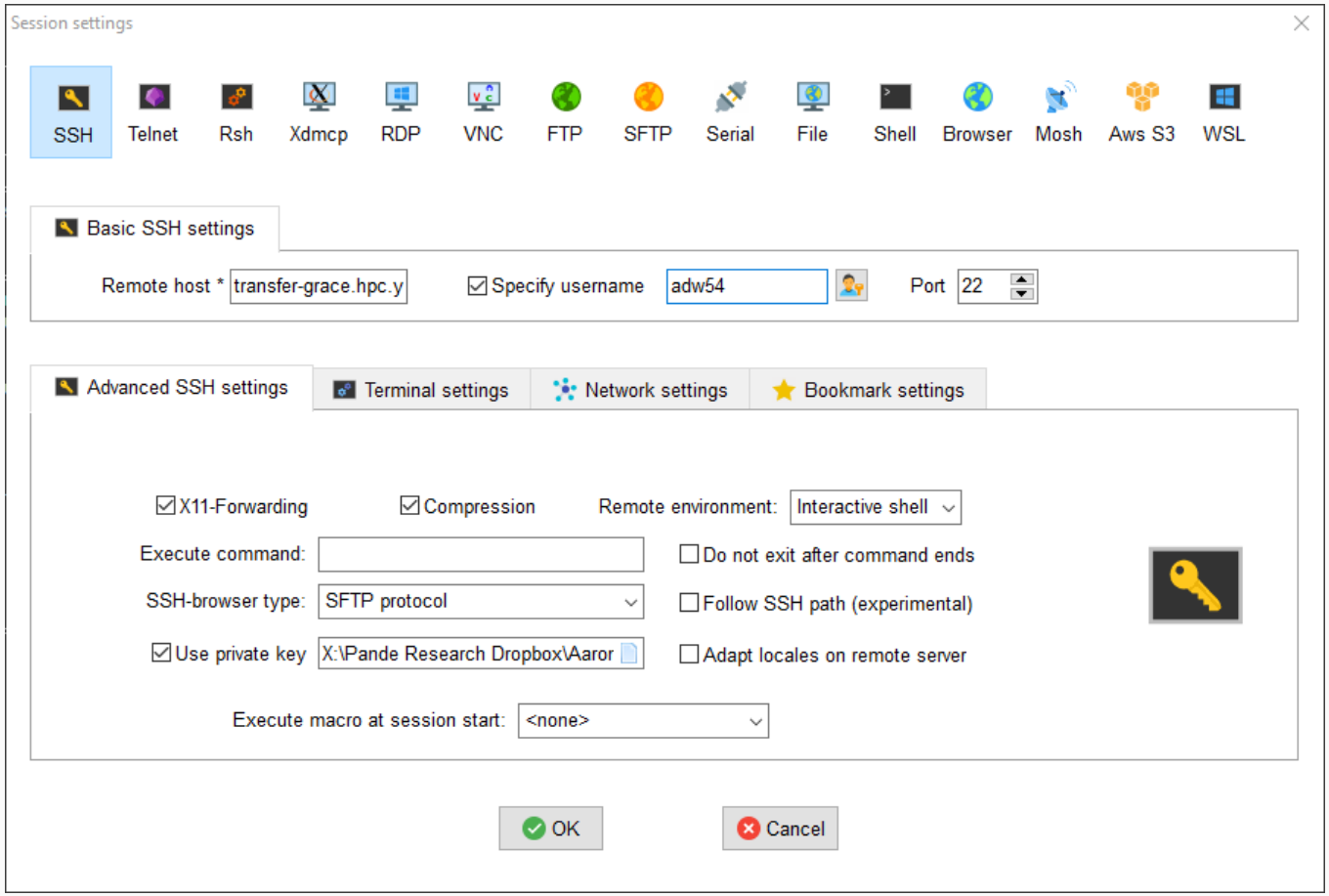
Step 5: Log into the cluster via MobaXterm

Under *Specify Username*, add yours.



Step 5: Log into the cluster via MobaXterm

Under *Use private key*, find the private key file you saved (ends in .ppk).

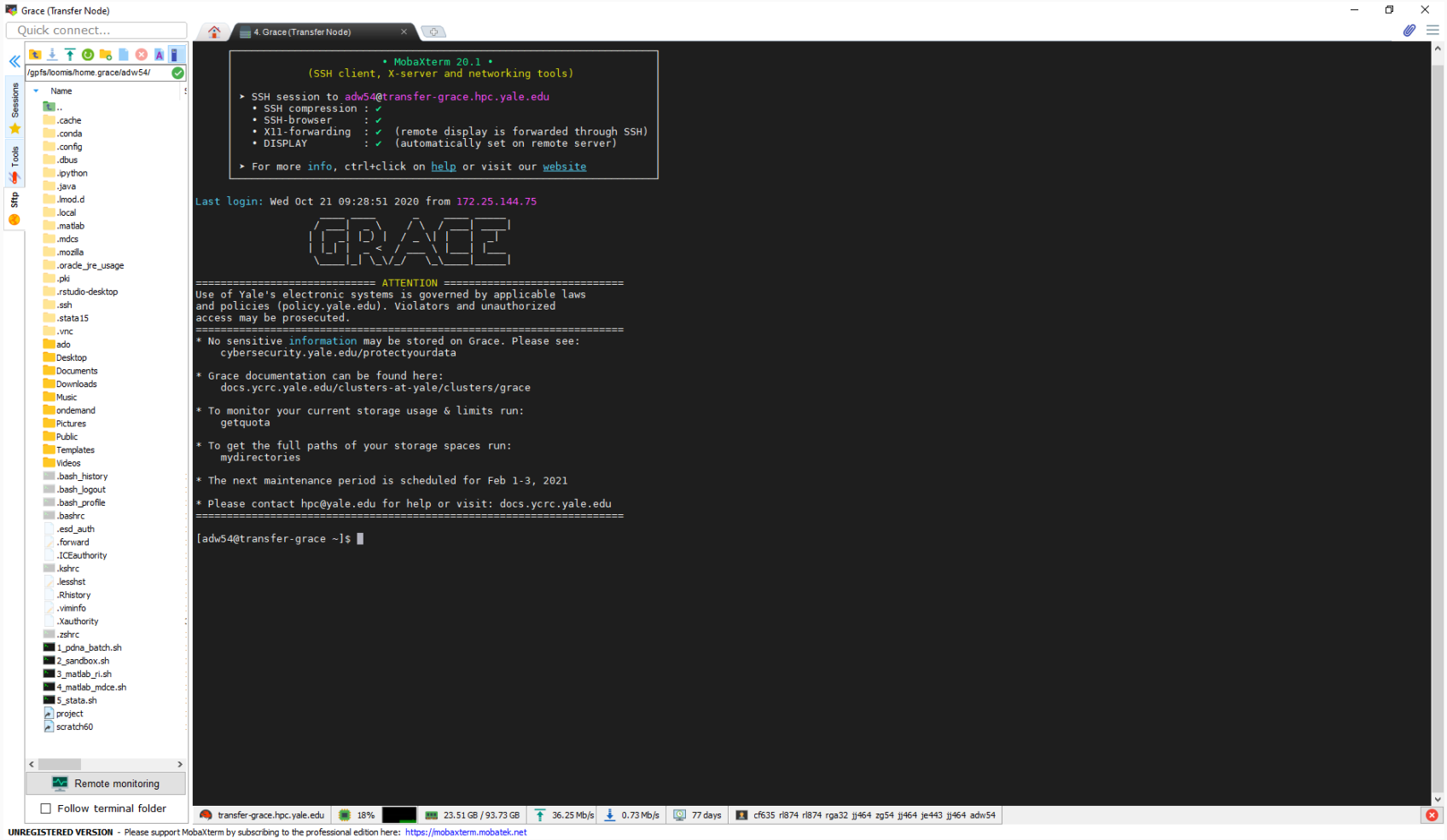


Step 5: Log into the cluster via MobaXterm

- If MobaXterm starts crashing: Select "None" under *SSH-browser type*.
- For now, keep the default option (*SFTP Protocol*): The SSH browser allows you to quickly upload small files via MobaXterm, which is useful.
- However, some users have reported that it begins crashing after a successfuly login for no particular reason. If it does this, this step may help.

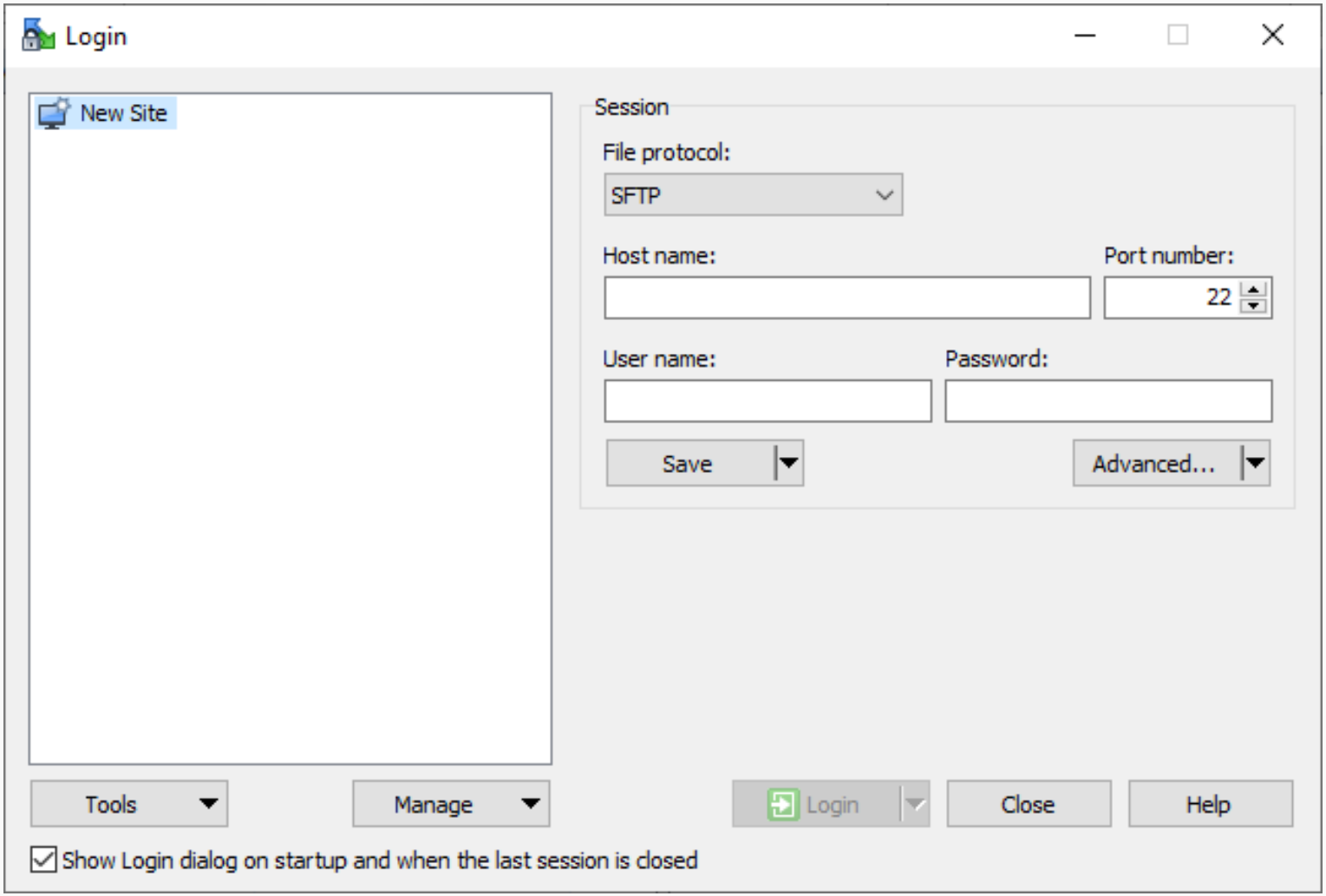
Step 5: Log into the cluster via MobaXterm

You should see a blank terminal ready for commands!



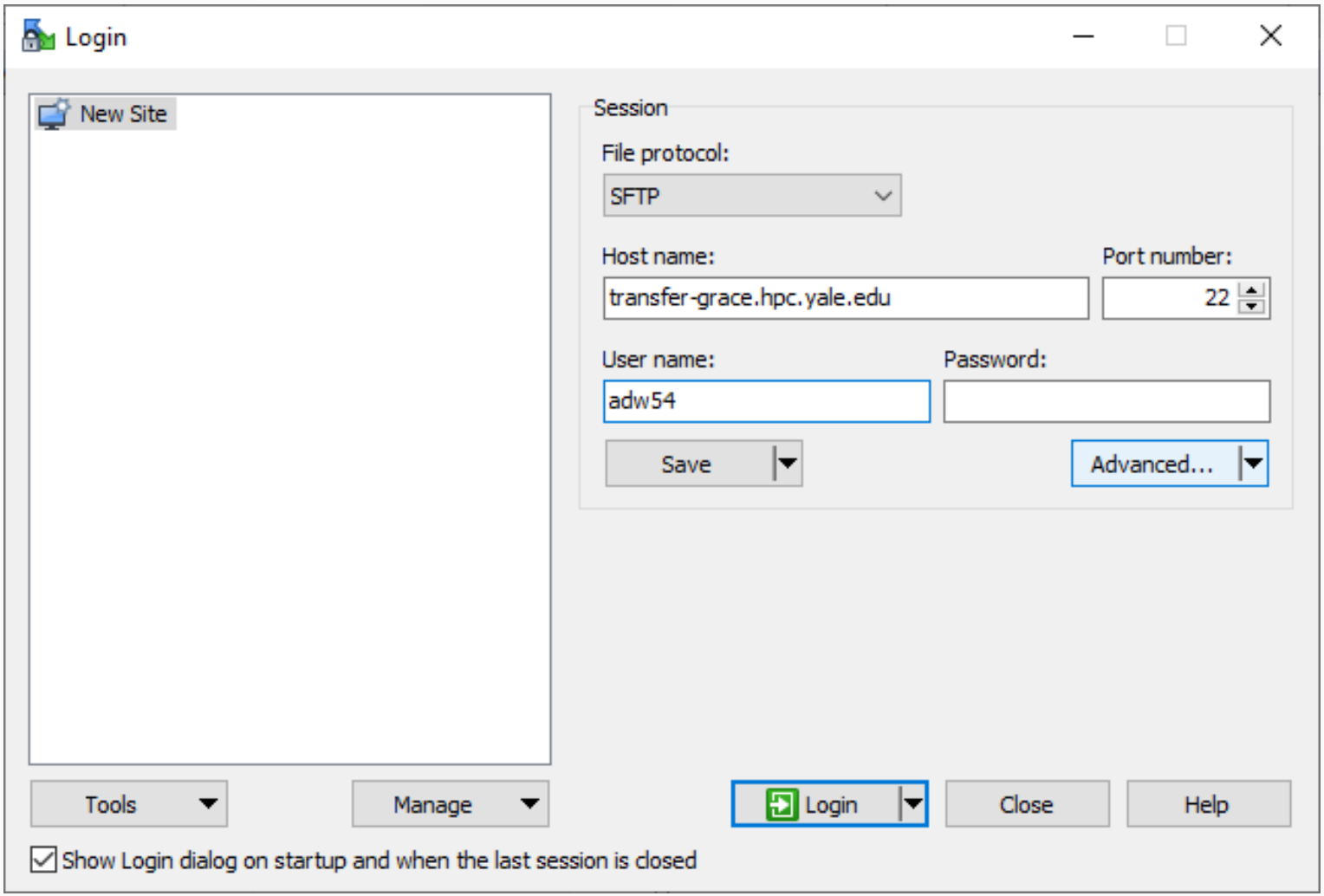
Step 6: Log into WinSCP and sync a folder

Open WinSCP and click on "New Site"



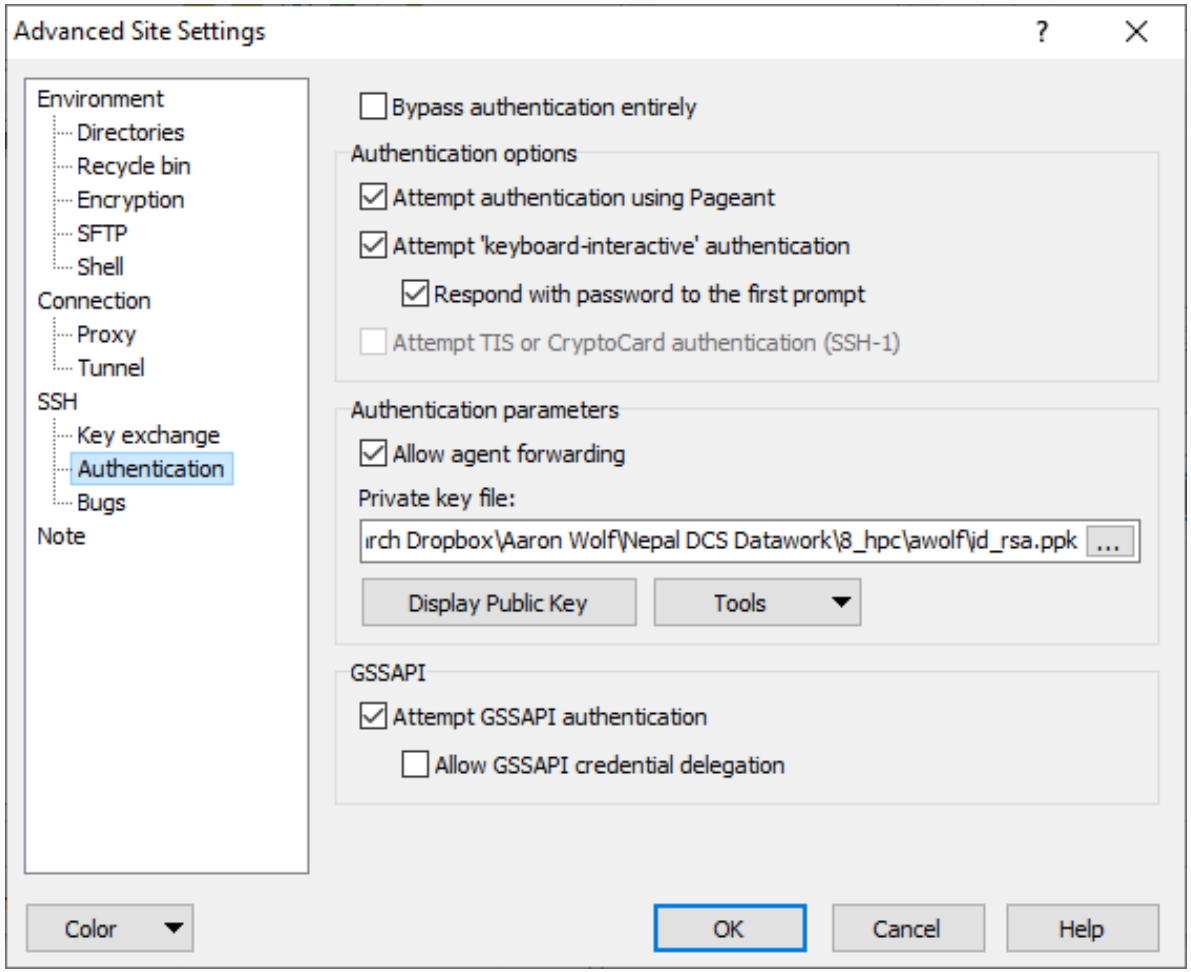
Step 6: Log into WinSCP and sync a folder

Input the *Host name* (transfer-grace.hpc.yale.edu) and use your NetID as the username. Then click on *Advanced*.



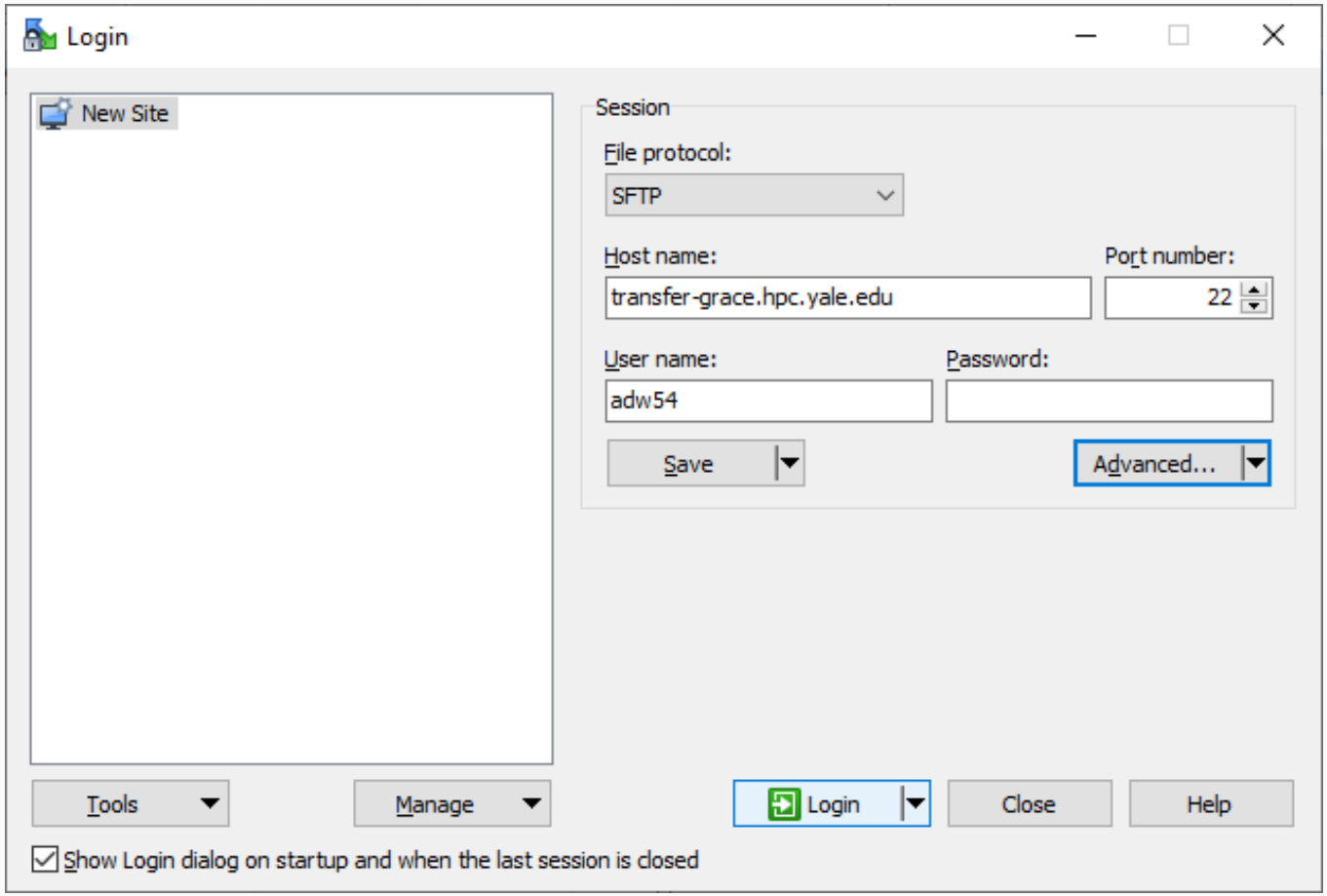
Step 6: Log into WinSCP and sync a folder

Under SSH > Authentication > *Private key file*, choose your private key (ends in .ppk). Click "OK".



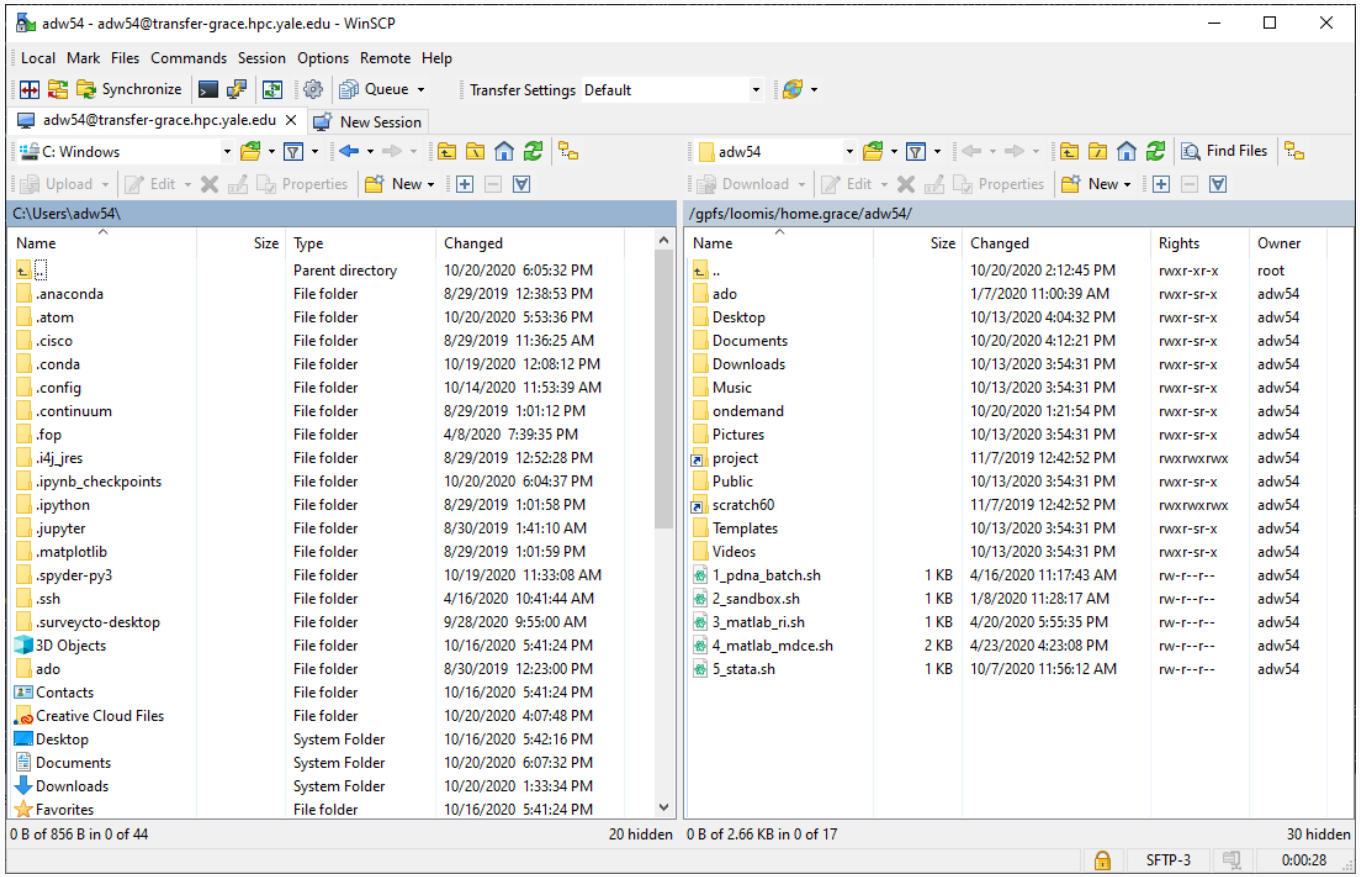
Step 6: Log into WinSCP and sync a folder

Click *Login*.



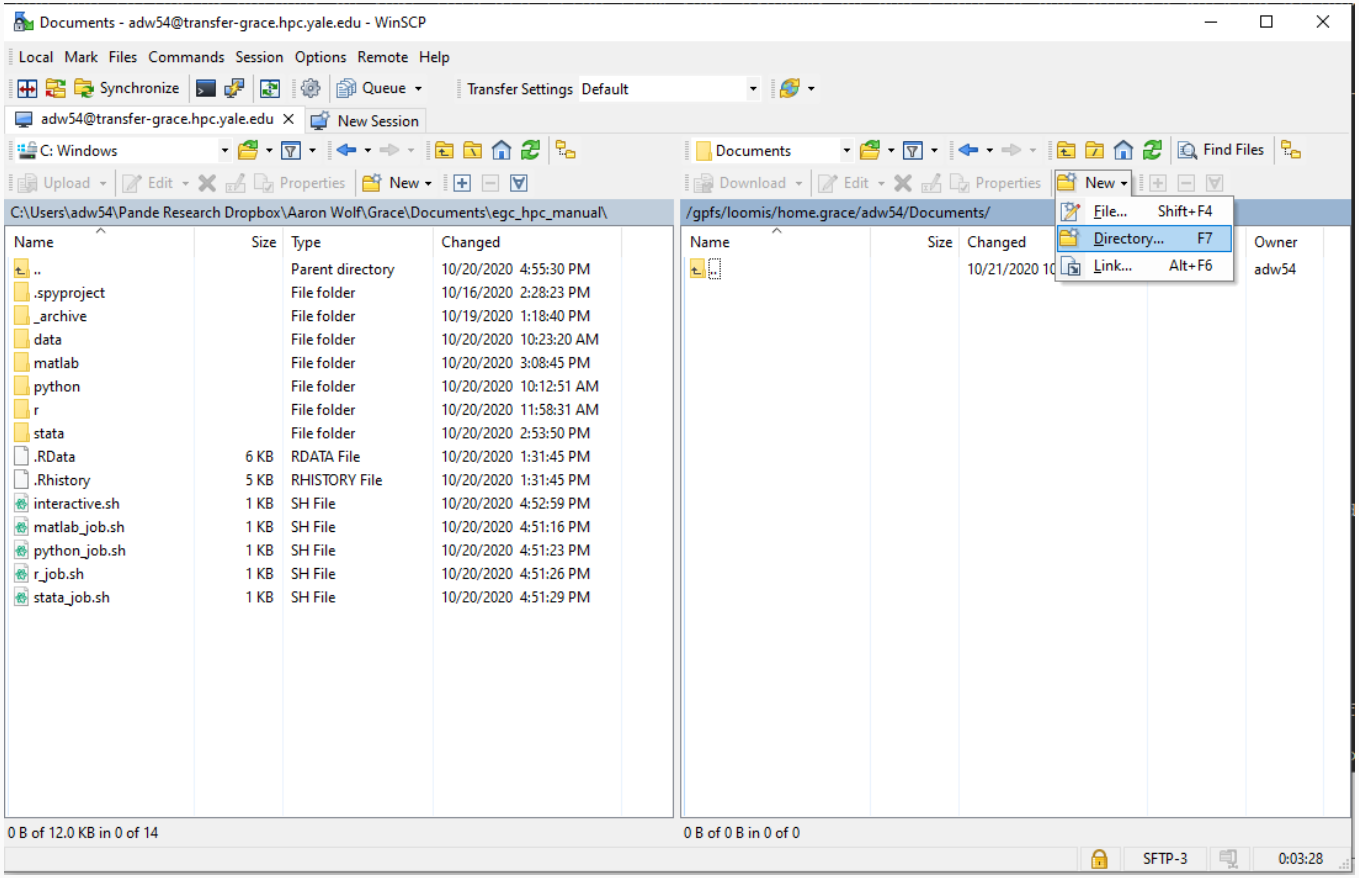
Step 6: Log into WinSCP and sync a folder

You should now see the following screen. On your left are files from your system. On the right is the default working directory on Grace. You can store larger files in the *Project* folder, but your home folder already has 125GB of space available. Double click *Documents*.



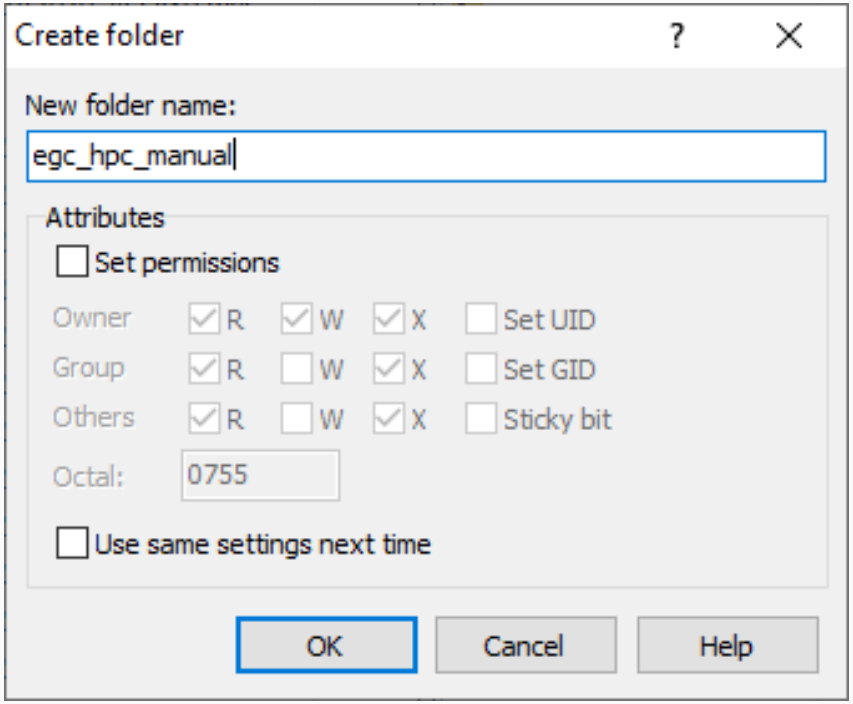
Step 6: Log into WinSCP and sync a folder

Use *New* to create a new folder to synchronize.



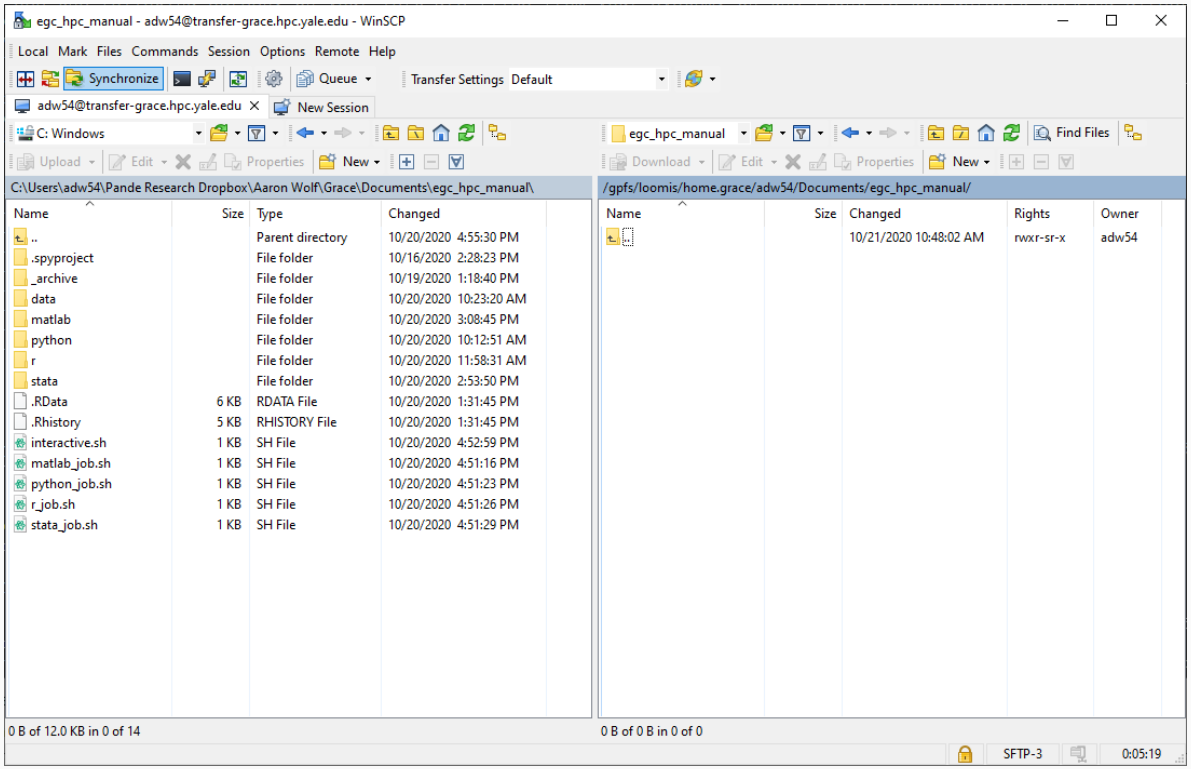
Step 6: Log into WinSCP and sync a folder

Name it *egc_hpc_manual*, or any other name you wish.



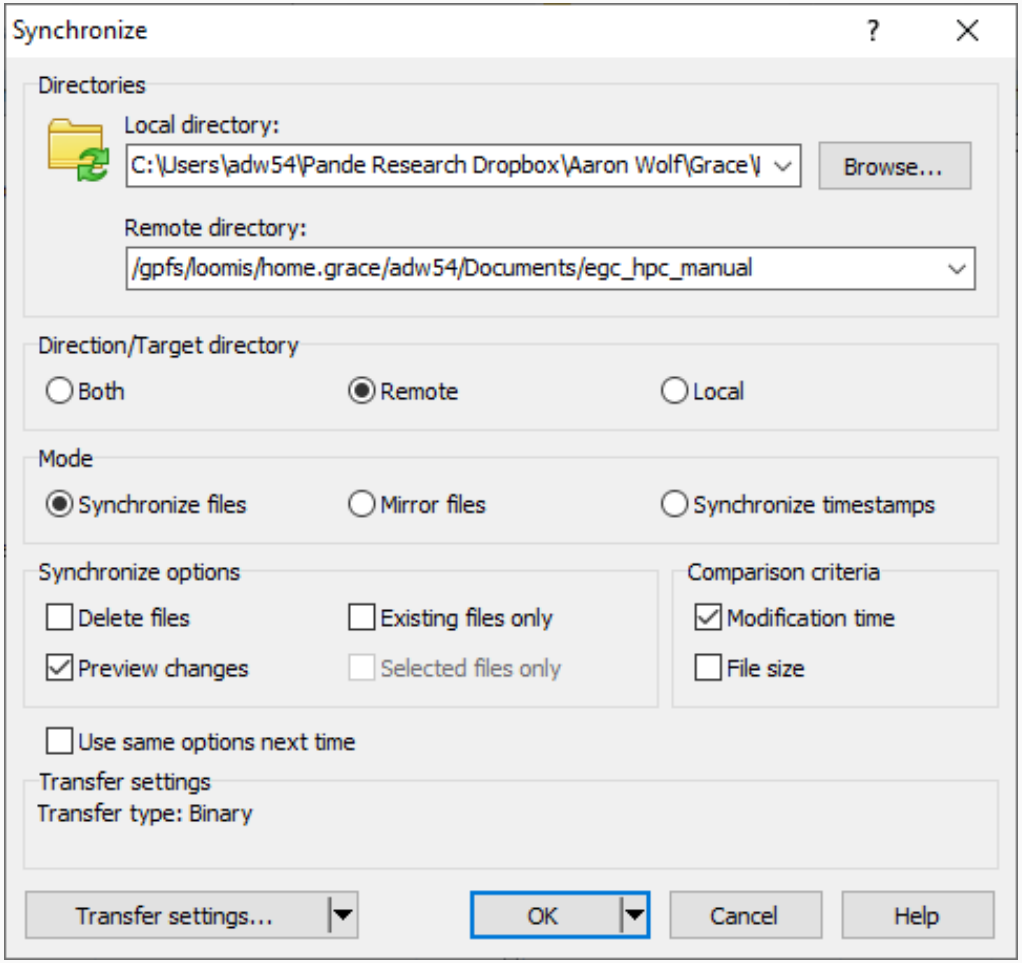
Step 6: Log into WinSCP and sync a folder

Ensure you have navigated to the appropriate folder you wish to sync on the left hand side as well. I have navigated to *egc_hpc_manual*. Click *Synchronize*.



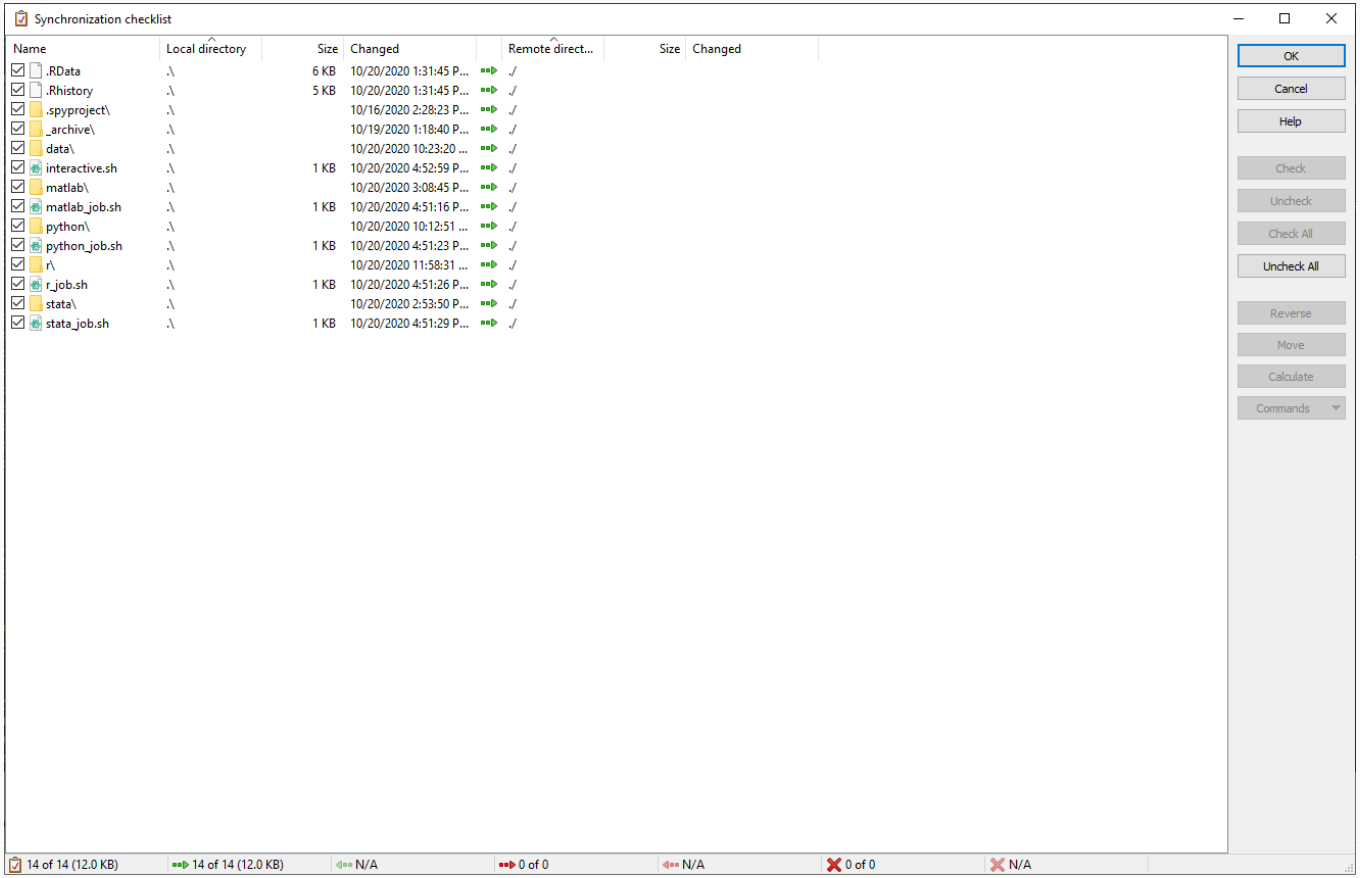
Step 6: Log into WinSCP and sync a folder

Select *Remote* under *Direction/Target directory* to push files from your PC to the server. You can specify local if you want files from the server to be copied to your hard drive, or both to synchronize.



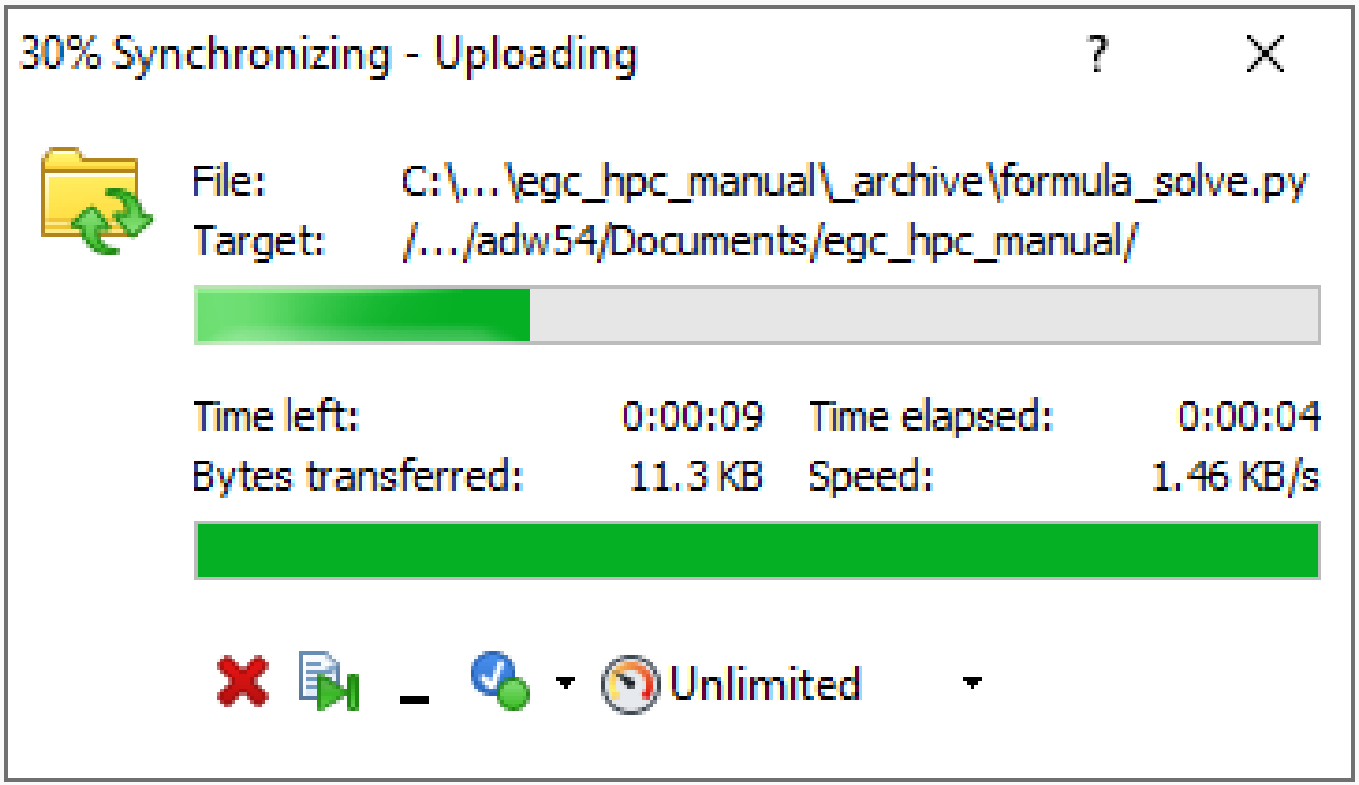
Step 6: Log into WinSCP and sync a folder

Choose the files to sync and press *OK*.



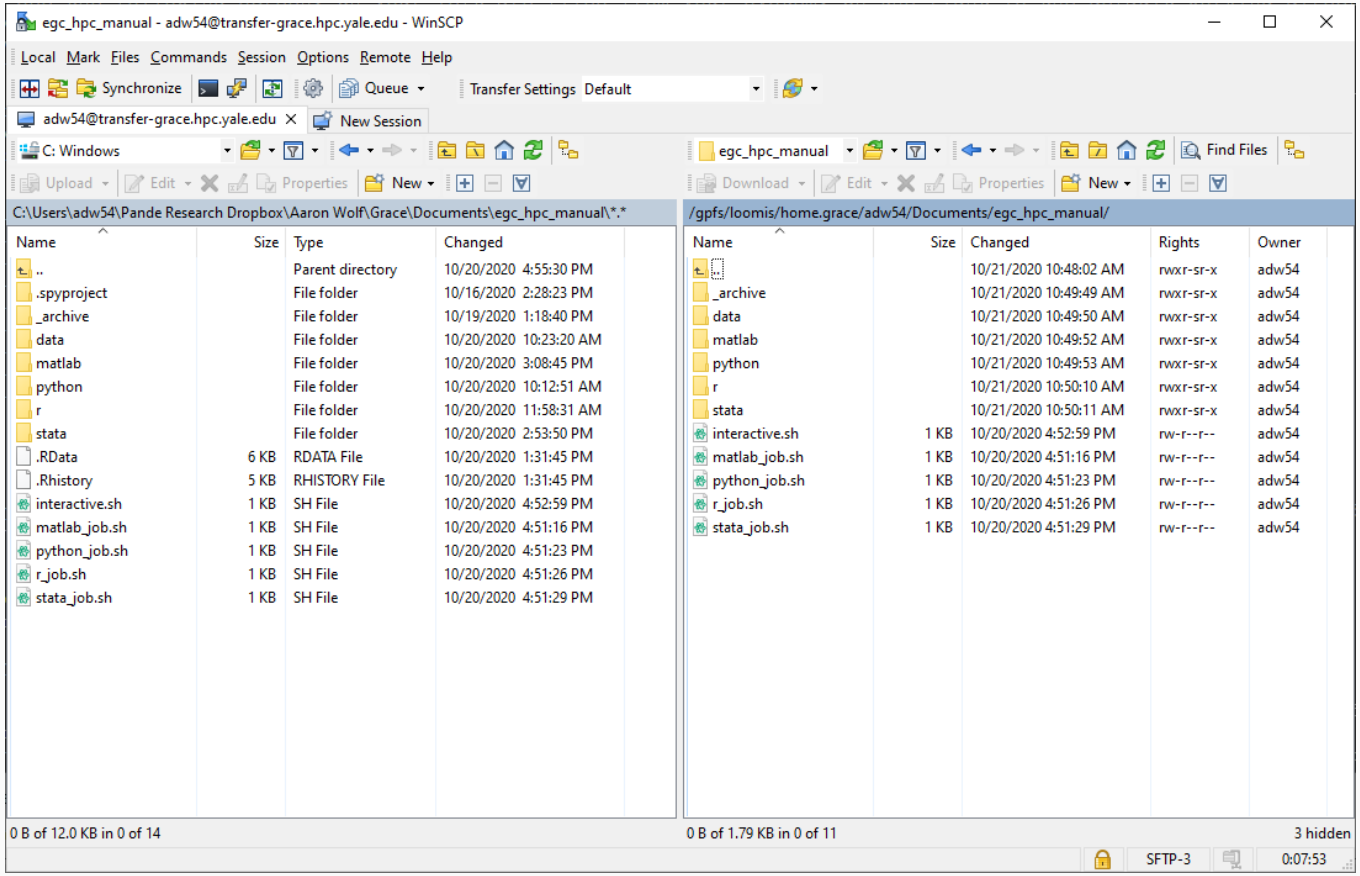
Step 6: Log into WinSCP and sync a folder

You will see an overall progress bar, as well as bars for each item.



Step 6: Log into WinSCP and sync a folder

When synchronization is complete, you should see similar structures on both sides.



Done!

Once you've successfully logged in to everything, go ahead and sync your project folder (or just the sections of it that are required to run your HPC job), and we can move on to running interactive and batch jobs.

Interactive Mode

What is interactive mode?

- By default, when you log into the cluster (via MobaXterm or other), you are in a "login node".
- This is basically a single node and CPU for you to use to interact with the cluster.
- From here you can request more resources and submit "batch scripts": Code that requests more resources and executes specific files.
- First, we will run in "interactive mode".
- This means requesting more resources than the default login node, but we will still type in commands one by one rather than submitting a batch file.

Starting an Interactive Session

Log into the cluster via MobaXterm (see Step 5)

Starting an Interactive Session

Input the following command:

```
srun --pty -c 4 -p interactive -t 4:00:00 bash
```

Starting an Interactive Session

Notice that the node has changed: At the start of the script, it used to say `[adw54@grace2 ]`. Now it says `[adw54@c38n01 ]` (the username and specific nodes will be different for you).

Starting an Interactive Session

SLURM gave you an interactive node (c38n01) with 4 cpus for 4 hours:

```
srun --pty -c 4 -p interactive -t 4:00:00 bash
```

Using interactive sessions to experiment

- Interactive mode is great for setting up your code and debugging.
- It is not great for running cumbersome files: Once you get above 16 CPUs, it'll likely take some time before you are actually granted the job (4 CPUs is nothing).
- When you need 32, 64, 128 etc. CPUs, you'll want to set up a batch script (we'll get there).

Running “Jobs”

What is a 'Job'?

- SLURM considers each request you make to the server as a 'job'.
- A job is simply a set of resources for a defined period of time.
- For example, you might request 32 cpu cores for 4 hours. You would do this by submitting a job request.
- You use a 'batch script' to request a job. Here is a basic example:

Example batch file (example.sh)

Listing 1: General Batch Script Example

```
#!/bin/bash
#SBATCH --job-name=example_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=16
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=1:30:00
#SBATCH --mail-type=ALL
#SBATCH --mail-user=aaron.wolf@yale.edu
```

Example batch file (example.sh)

- This script will request a new job, named 'example_job', which has:
  - 16 cores (CPUs) on 1 node;
  - 5GB of memory
  - 1.5 hours runtime; and
  - Emails the provided address when the job begins and completes.
- Note that we haven't actually requested that SLURM execute any commands or files here. We will be granted the resources, then SLURM will determine that our job has ended (if we actually sent this).
- We will discuss how to actually *do* something (i.e. run Stata, Python, R, etc.) in a batch file in sections for each piece of software.

Stata

Running Stata in interactive mode

Log into the cluster via MobaXterm (see Step 5) and initiate an interactive session:

```
srun --pty -c 4 -p interactive -t 4:00:00 bash
```

Running Stata in interactive mode

Load the Stata module (this must be done every session to tell Grace to fetch the relevant program files from its memory):

`module load Stata/15`

Running Stata in interactive mode

Initiate Stata MP (command line version):

`stata-mp`

Running Stata in interactive mode

You now have Stata running as a command-line program. Try typing a few common commands:

```
pwd
di "`c(username)'"
```

Running Stata in interactive mode

When you are done with Stata, type in `exit, clear`. Stata will close, and you will be back to the bash script and can resume typing commands directly to your interactive node.

Can I use the Stata GUI?

- Yes! Grace is a computer, so it is really just running Stata the same way you would.
- The difference is that you need to tell Grace that you want it to forward the GUI to you.
- Essentially, you will be "streaming" Stata in a weird interactive box.
- For this to work, we need to first tell Grace that we want to enable GUIs to display on our screen. This is called "X11 forwarding", in SLURM.

Running Stata in interactive mode with a GUI

Start an interactive session. This time, specify the `X11` option:

```
srun --pty --x11 -c 4 -p interactive -t 4:00:00 bash
```

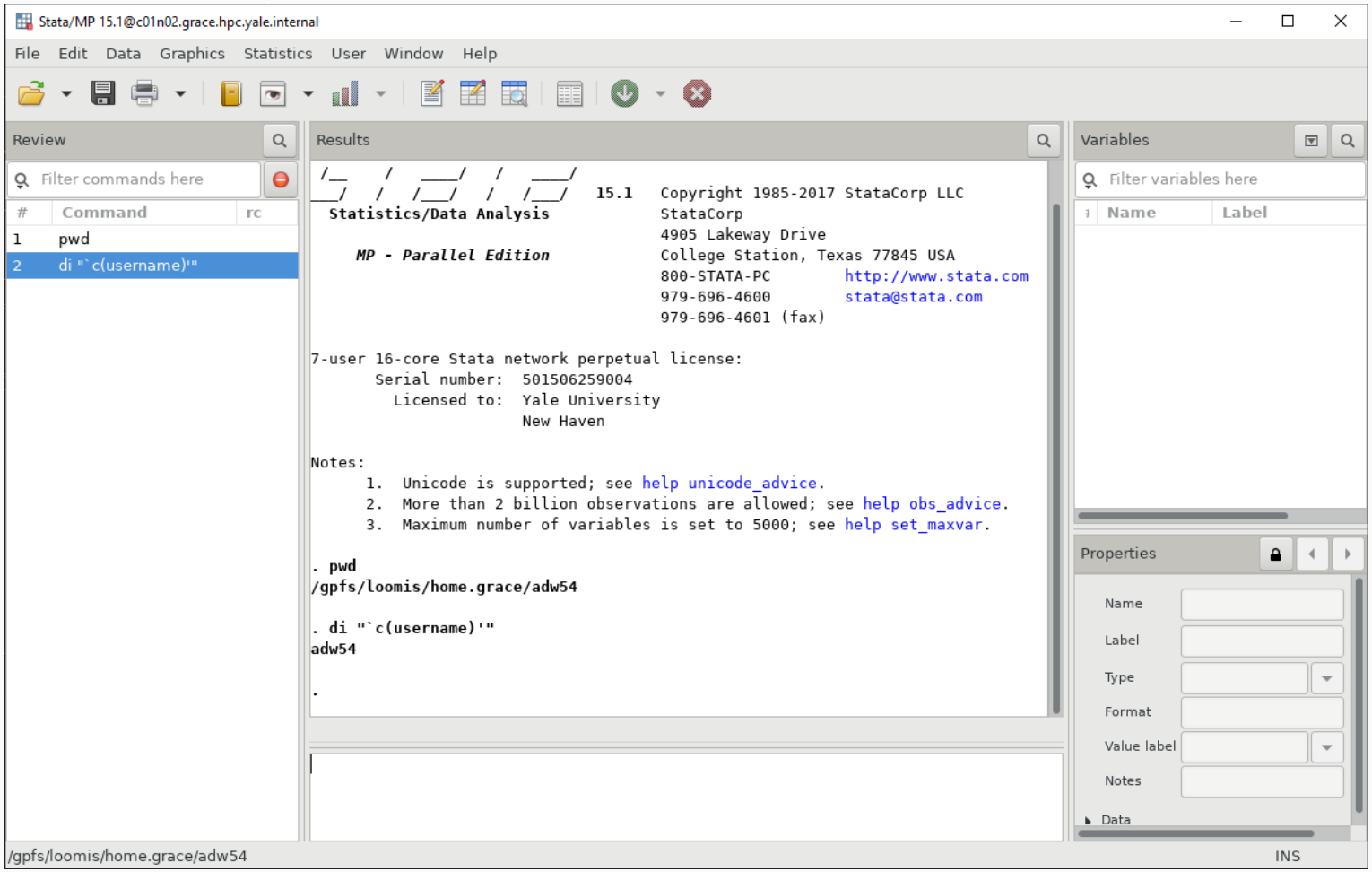
Running Stata in interactive mode with a GUI

Load the Stata module and initiate Stata. This time, input `xstata-mp`:

```
module load Stata/15
xstata-mp
```

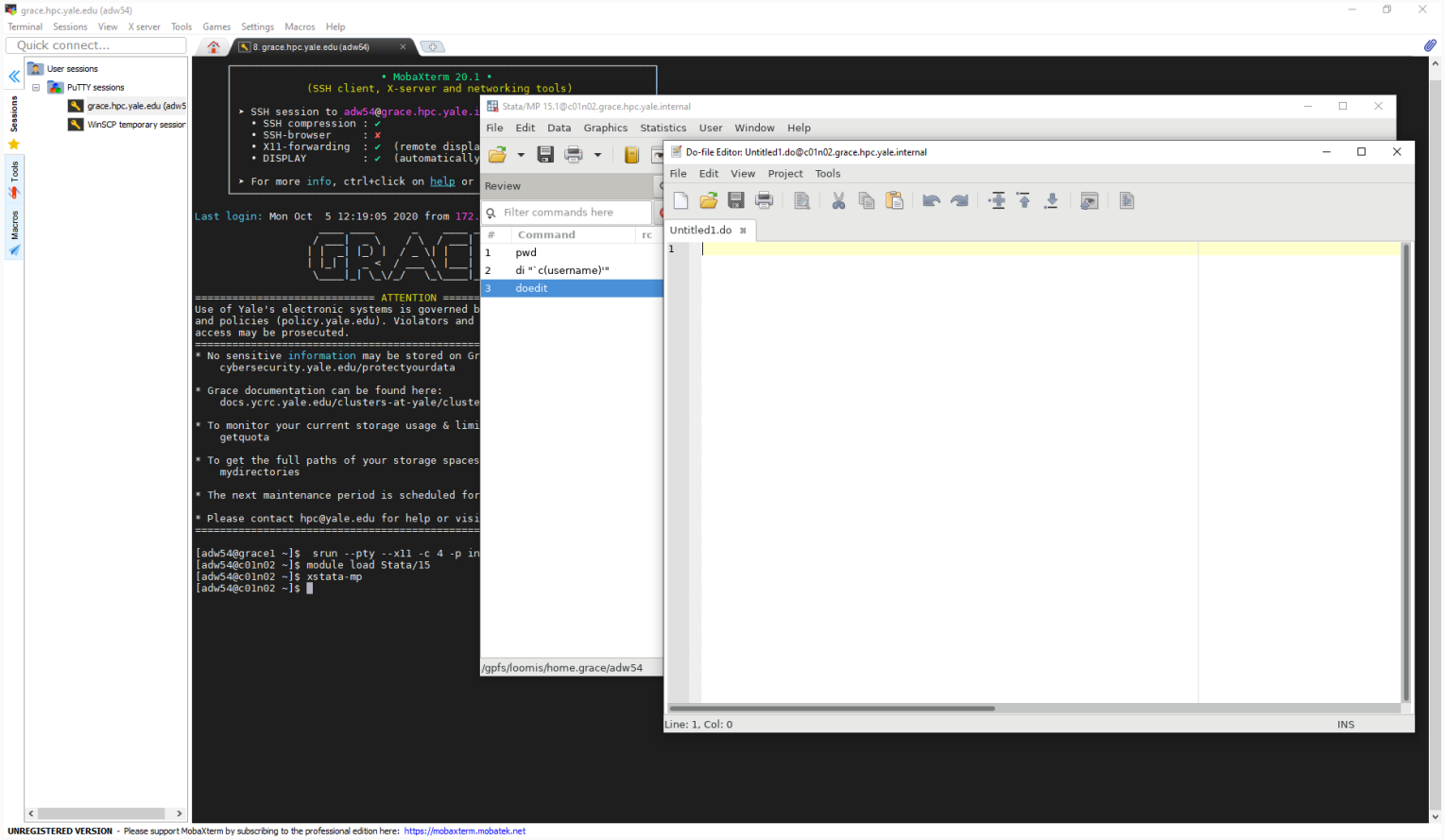
Running Stata in interactive mode with a GUI

The GUI will pop up in a different window. You can now input commands directly into the command window as you would on a desktop.



Running Stata in interactive mode with a GUI

You can even use the .do file editor to edit, save, and run commands, as usual.



Running Stata in interactive mode with a GUI

When you are done, simply exit Stata by typing `exit` in the command window or clicking the 'X' button as you would on a desktop.

Running Stata in Batch mode

- We can run Stata directly from our batch scripts, allowing us to use Stata with more resources than a typical interactive session.
- The codes for running Stata in batch mode are the same as if we were using the command line.
- Crucially, we want to submit a direct command to Stata, rather than just loading it.
- In this case, we created a master .do file which runs multiple project .do files.
    - This is a good practice generally, but for SLURM is also the most efficient way to run all .do files for a project.
    - The alternative is to write a similar `stata-mp do fileX.do` for each .do file.

Running Stata in Batch mode

Listing 2: stata_job.sh

```
#!/bin/bash
#SBATCH --job-name=stata_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# Load Stata module and run master .do file
module load Stata/15
stata-mp do  stata/0_master.do
```

Running Stata in Batch mode

Here, I have added a few new lines:

1. `#SBATCH --output=%x_%j.out` ensures that the SLURM output file (essentially a log) will have the format *job_name_job_id.out*.
2. `cd /home/adw54/Documents/egc_hpc_manual` is a usual *change directory* command.

Stata Example Project

Stata Example Project

Now, let's create a new Stata project with a batch script, master .do files, and do files with the core analysis commands.

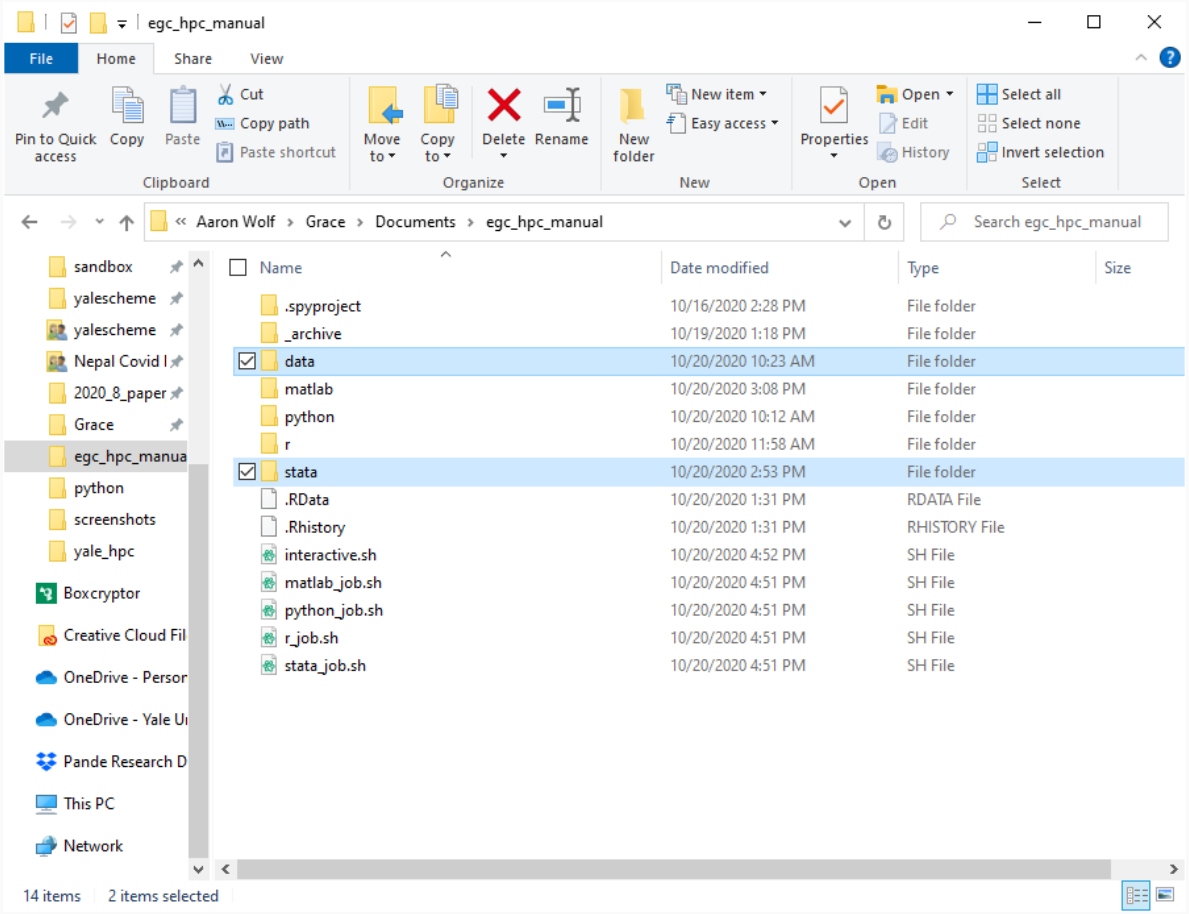
We will set it up so that you can run the do file on your own computer as well as Grace (once the files are synchronized).

Then we will create a batch script that will run the master .do file, which will itself run all the project .do files.

Step 1: Create project folder

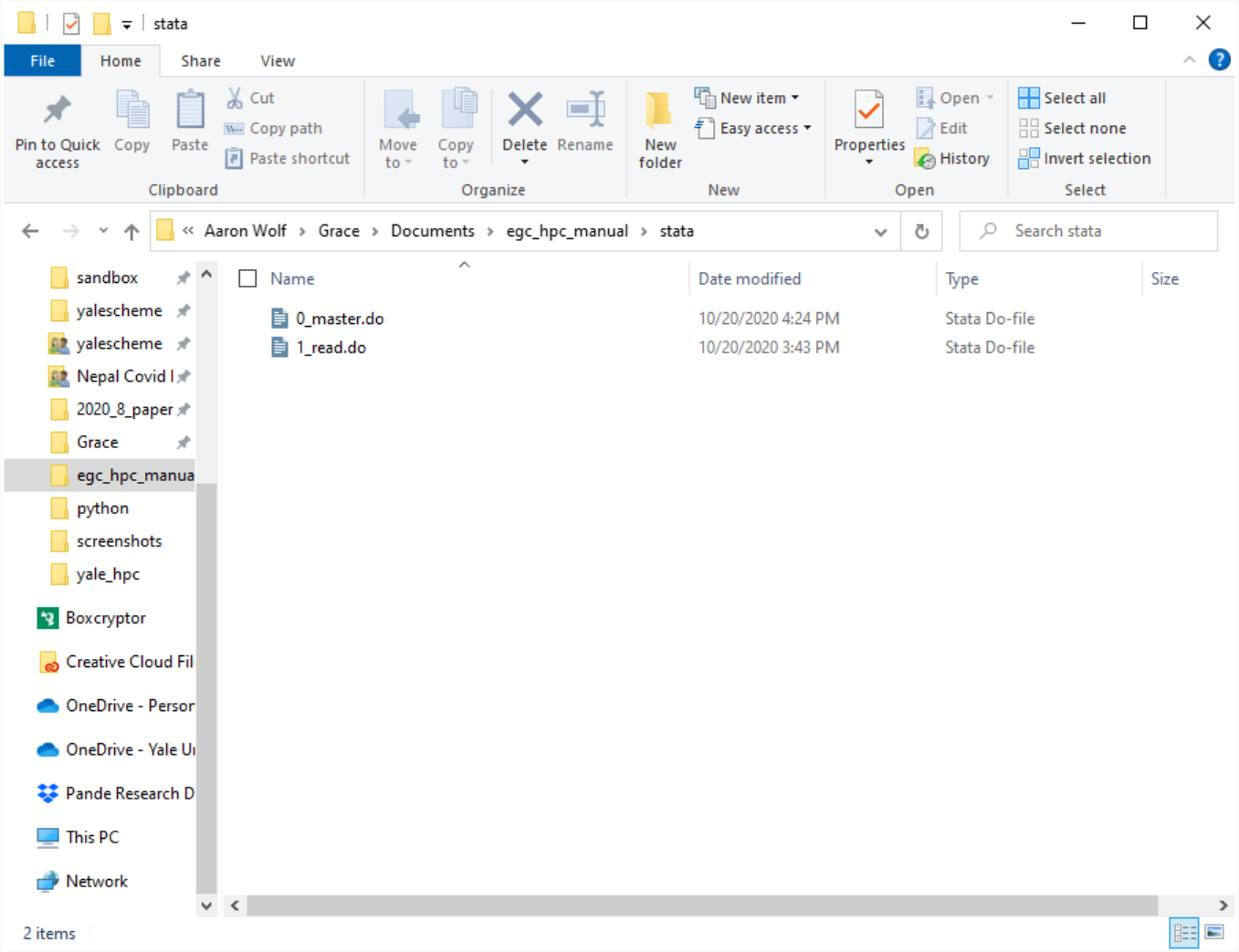
Create a new folder on your PC anywhere you like. This will be your main (root) folder.

Add folders for data and do files.



Step 2: Create.do file

Create a master do file, as well as a sub-files (we'll call this one *1_read.do*, as we will only use it to read in the auto dataset). The following slides will provide the content of these files.



Step 2: Create.do file - 0_master.do

Listing 3: 0_master.do

```
* ------------------------------------------------------------------------------
/*
*       Filename: 0_master.do


*       Purpose:   This is the master .do file for <PROJECT NAME>.


*       Created by: adw54
*       Created on: 7 Oct 2020
*       Last updated on: 7 Oct 2020
*/
* ------------------------------------------------------------------------------
```

Step 2: Create.do file - 0_master.do

Listing 3: 0_master.do

```
*******************************************************************************
*
*   SECTION 1: Directories
*
*******************************************************************************

//  Set-Up Working Directories
    if "`c(username)'" == "adw54"  & "c(os)" == "Windows" {
        global root "C:/Users/adw54/Pande Research Dropbox/Aaron
              Wolf/Grace/Documents/egc_hpc_manual"
    }
    else if "`c(username)'" == "adw54" & "`c(os)'" == "Unix" {
        global root "/home/adw54/Documents/egc_hpc_manual"
    }

  cd "$root"    // Set directory

  di "Setup Complete!"

*******************************************************************************
*
*   SECTION 2: Run Sub-Files
*
*******************************************************************************

    do stata/1_read.do
```

Step 3: Create sub files

Now we can make our subfiles. You can put anything you want here as a test.

For now, we will just load the auto dataset (saved in the data folder).

Step 3: Create sub-file 1_read.do

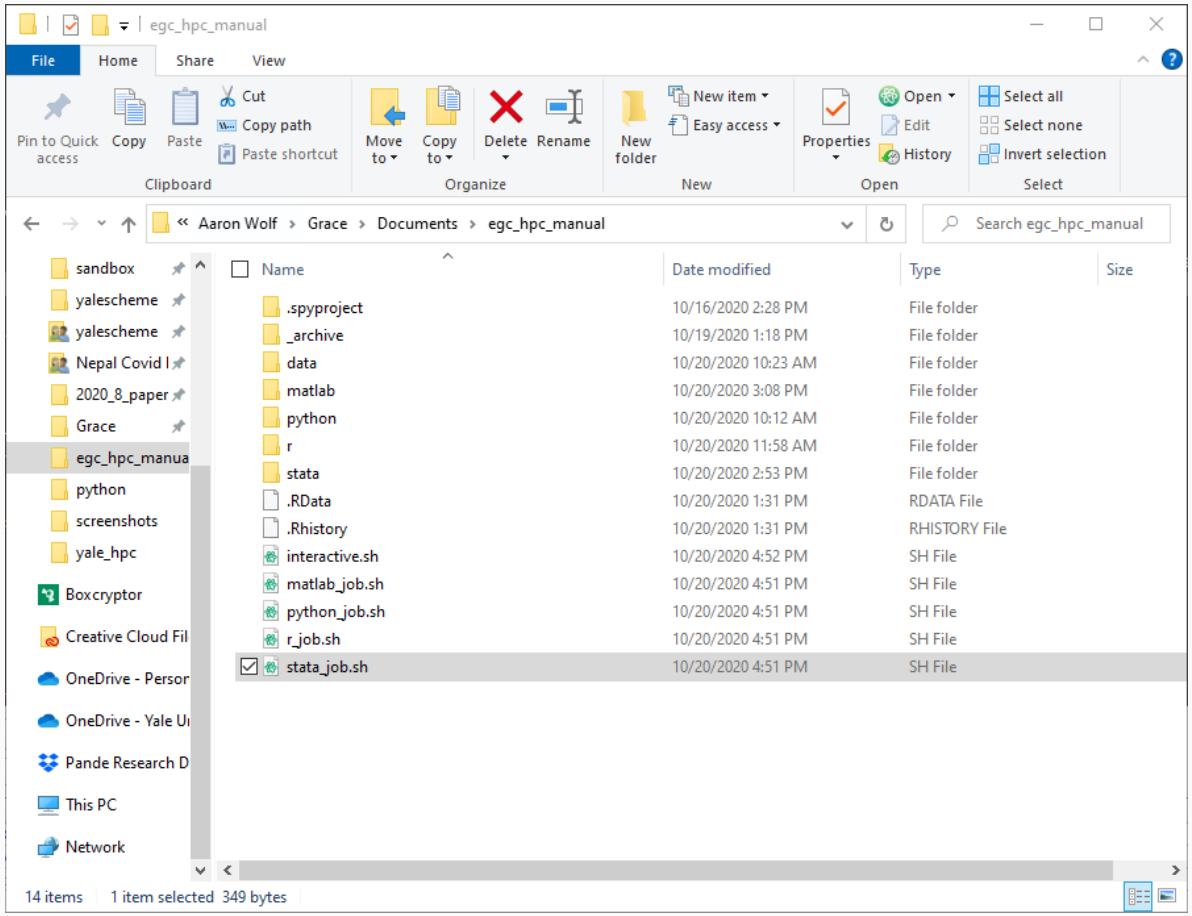
Listing 4: 1_read.do

```
* ------------------------------------------------------------------------------
*
*       Filename: 1_read.do


*       Purpose:    This .do file reads auto.dta


*       Created by: adw54
*       Created on: 7 Oct 2020
*       Last updated on: 7 Oct 2020
*------------------------------------------------------------------------------

//  Read data
    use data/auto.dta, clear
    di "Data read!"
```

Step 4: Create batch script

Next we will write a batch script which will run our master do file.

I will place this close in the project root directory, but you can also put it in your default working directory on Grace, as long as you remember where it is when you use `sbatch` to call it.



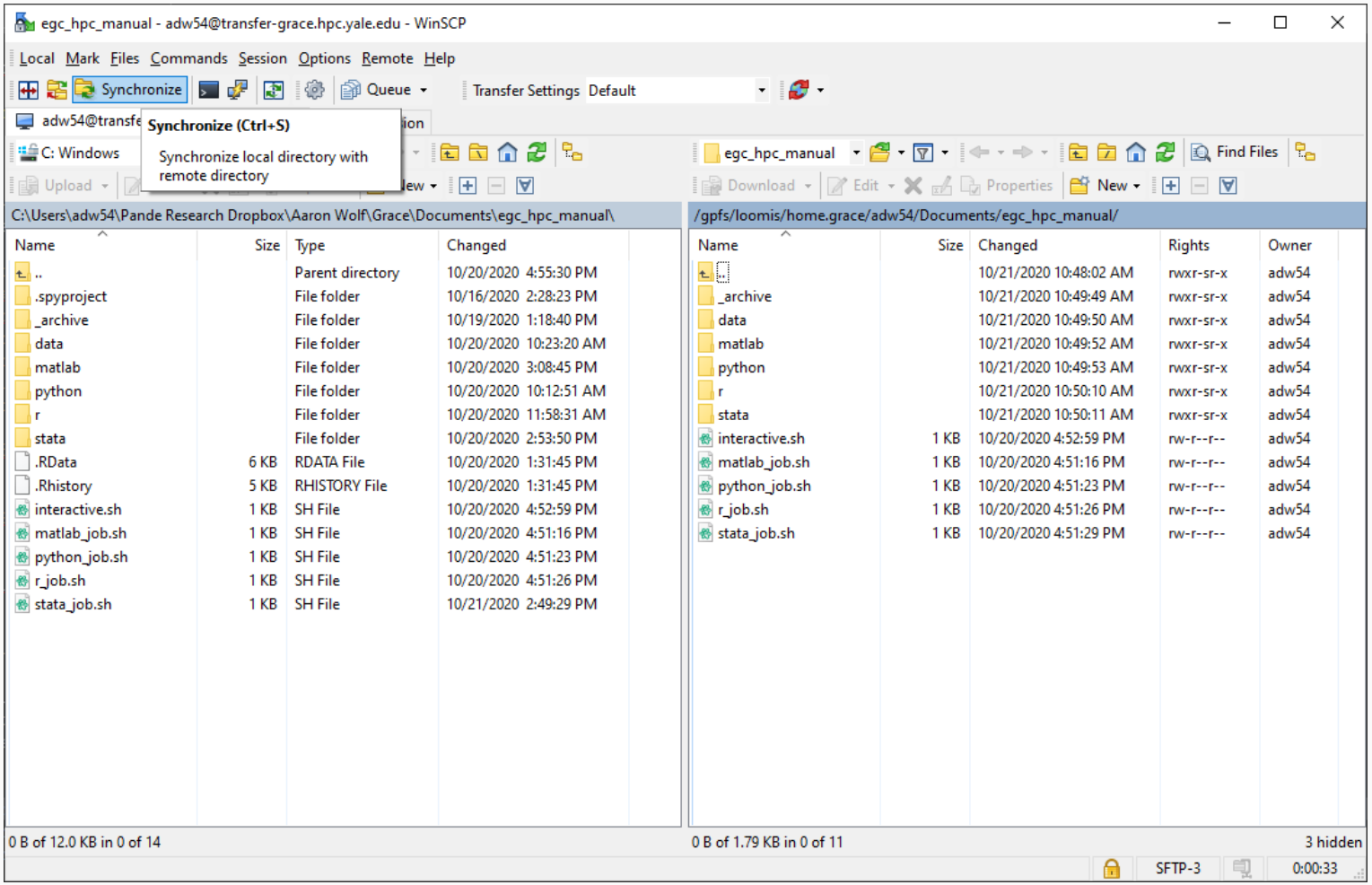
Step 4: Create batch script - stata_job.sh

Listing 5: stata_job.sh

```
#!/bin/bash
#SBATCH --job-name=stata_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# Load Stata module and run master .do file
module load Stata/15
stata-mp do  stata/0_master.do
```

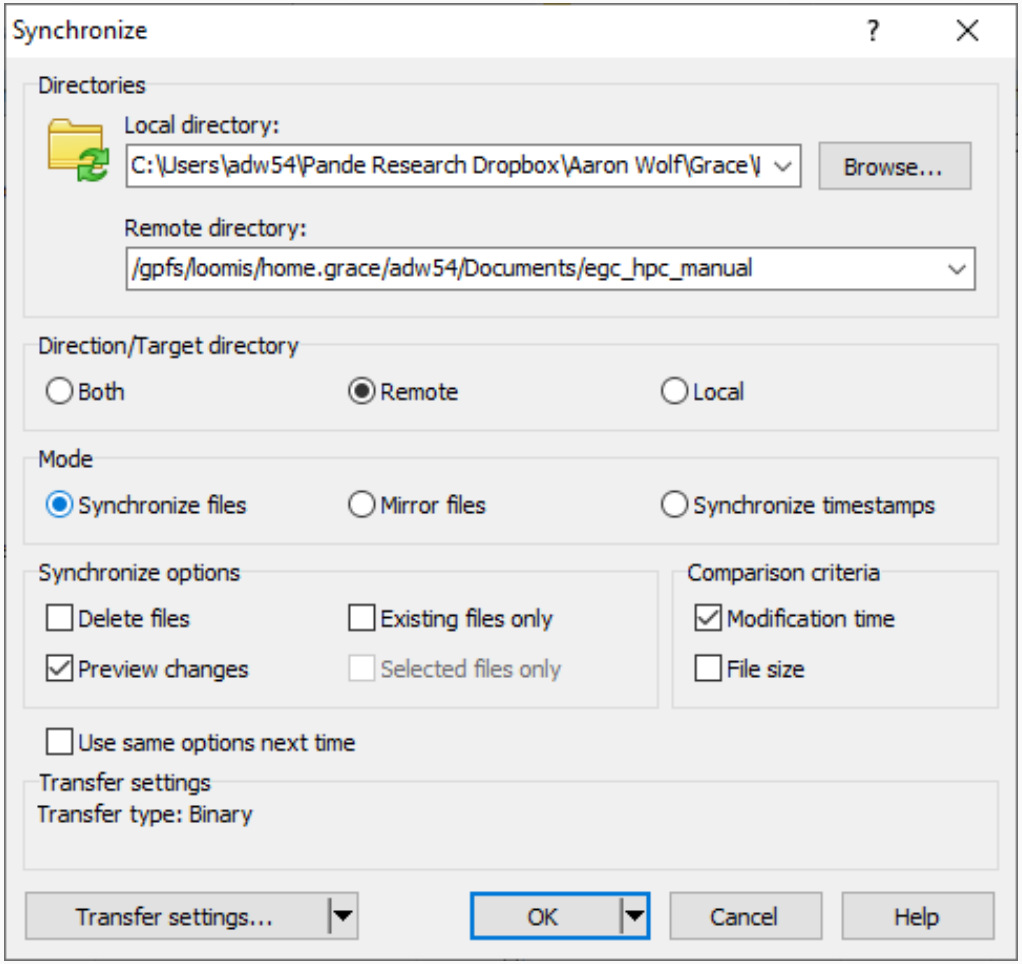
Step 6: Synchronize your root folder to your Grace project folder

Navigate to wherever you want your root directory to exist on Grace in WinSCP and select "Synchronize".



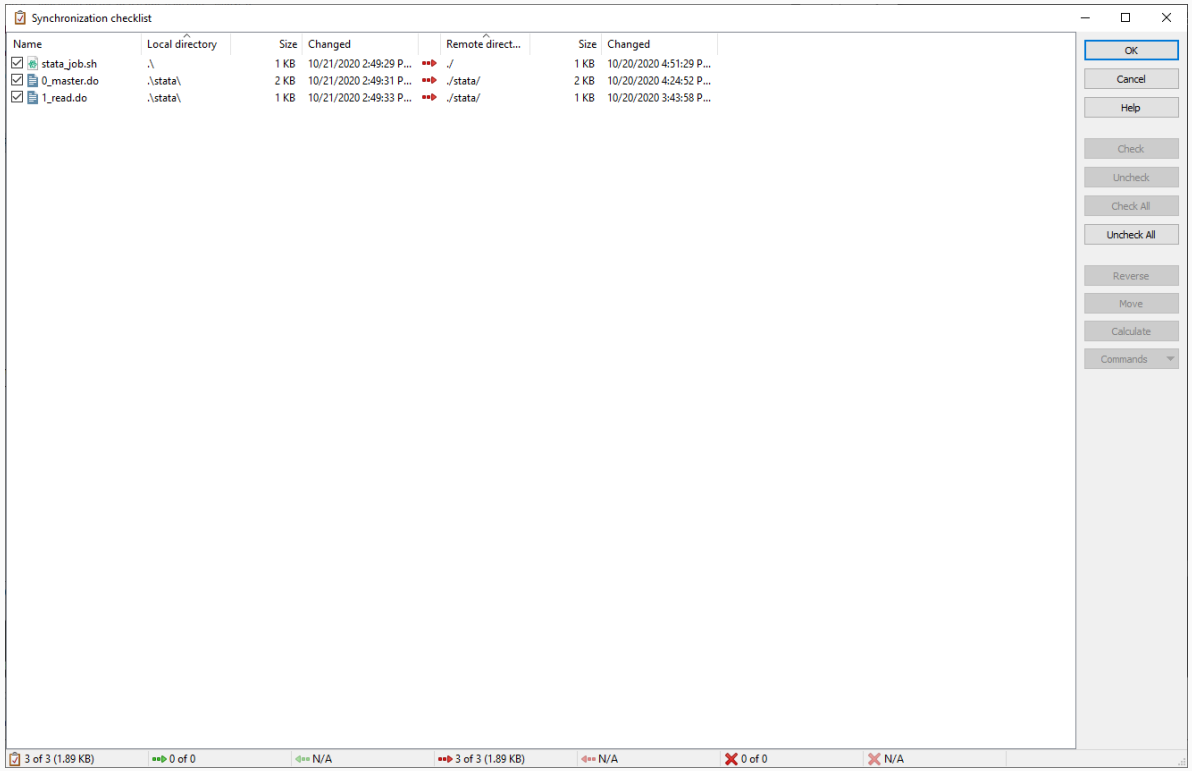
Step 6: Synchronize your root folder to your Grace project folder

Choose "Remote" and click "OK".



Step 6: Synchronize your root folder to your Grace project folder

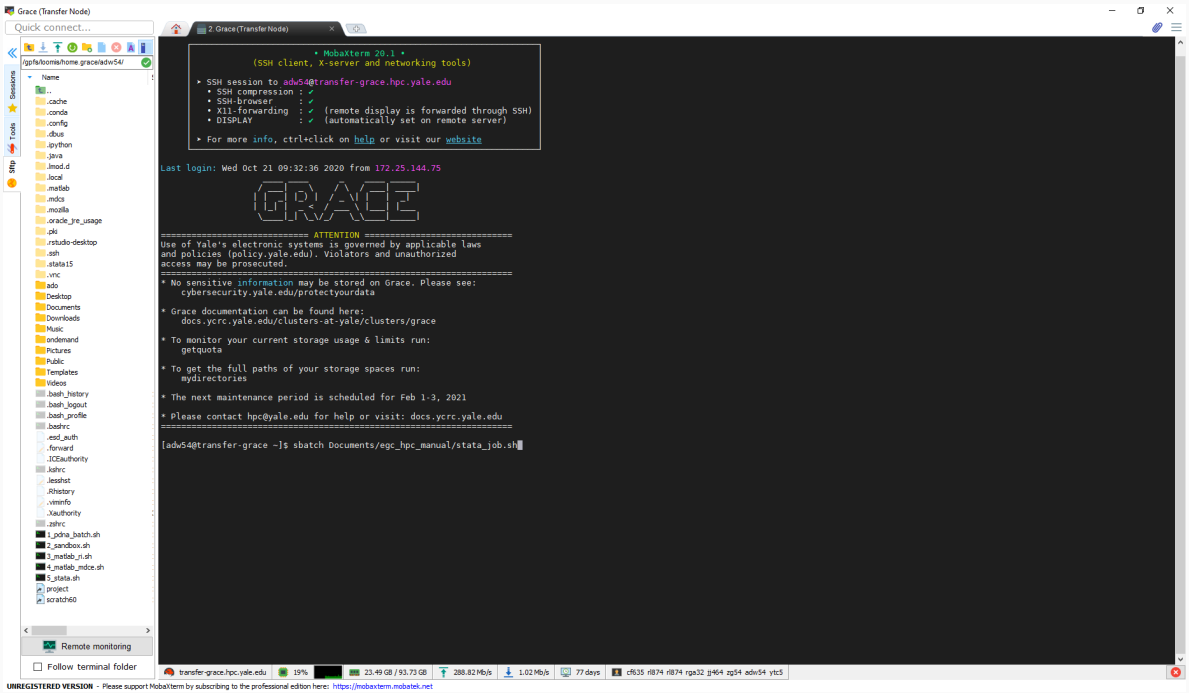
Select all files and click "OK".



Step 7: Run batch file

Open MobaXterm and input the following code (substituting the path for your .sh file):

`sbatch Documents/egc_hpc_manual/stata_job.sh`



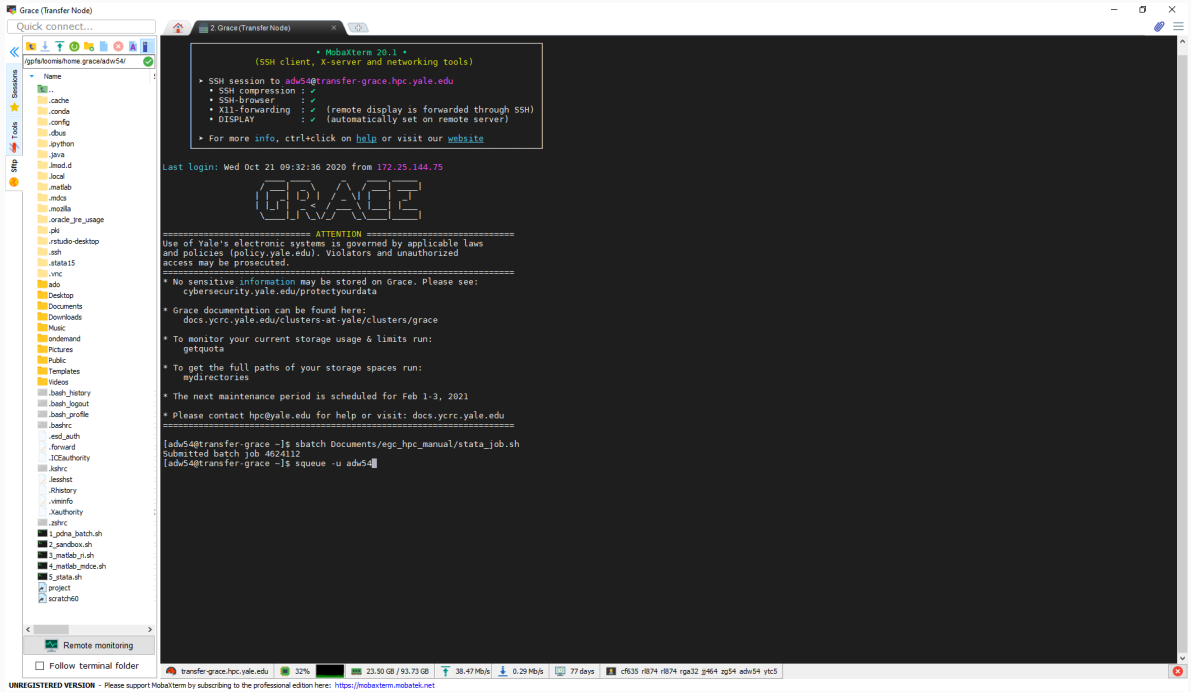
Step 7: Run batch file

You should see a message indicating the job was submitted. You can now input the following command (insert your username) to monitor the status of your job requests:

`squeue -u abc123`

Step 7: Run batch file

When Grace finishes your file, you will receive an email, and an output file in your home directory with the results (i.e. output screen). This can be used to diagnose errors and ensure everything ran.



That's it!

- When starting out, there are likely to be unforeseen errors/issues.
- We recommend starting with a similar example project and submitting batch scripts that ask for minimal resources to ensure you get all the paths right.
- Once the script runs smoothly without errors, switch the .do files called to the real ones and adjust the resource request accordingly.

Python

HPC: but with Python (!)

- Python code can be run directly from the command line using the `python myscrip.py` syntax.
- However, similar to Stata, where we had to "load" Stata MP (`module load Stata/15`), we need to tell Grace to find and use a Python installation.
- For this, we use an Anaconda Python environment.

What's Anaconda?

- If you've never used Anaconda think of it this way: Anaconda is to Python (and R!) what the App Store is to Apps. It is a single place to manage your installations of Python and R.
- Anaconda allows you to set up and control multiple Python "Environments": Separate installations of Python with different characteristics (e.g. you could have one environment on your computer with Python 2.7, and another with Python 3.6). In each environment, you can install all the packages you usually would with PIP.
- Anaconda is free for individual use: Check out https://www.anaconda.com/.
- **Before you can use Python on Grace, you need to first create a Python environment.** You can do this on a login node or an interactive node.

Create a new Python Environment called “py3_base”

Open MobaXTerm, login, and start a new interactive session:

```
srun --pty -c 4 -p interactive -t 4:00:00 bash
```

Create a new Python Environment called “py3_base”

Load the `miniconda` module:

`module load miniconda`

Create a new Python Environment called “py3_base”

Create a new Python environment with a few basic data science packages installed by default:

```
conda create -n py3_base python=3 numpy scipy matplotlib pandas ipython jupyter jupyterlab
```

Create a new Python Environment called “py3_base”

Now, whenever you want to use Python, whether in an interactive session or batch file, simply load up miniconda, and activate the environment:

```
module load miniconda
conda activate py3_base
```

Create a new Python Environment called “py3_base”

You can even start an ipython session for interactive Python!

```
ipython
```

Create a new Python Environment called “py3_base”

When you’re done, deactivate the current environment:

`conda deactivate`

Python Example Project

Python Example Project

Now, let's create a new Python project with a batch script, master .py files, and .py files with the core analysis commands.

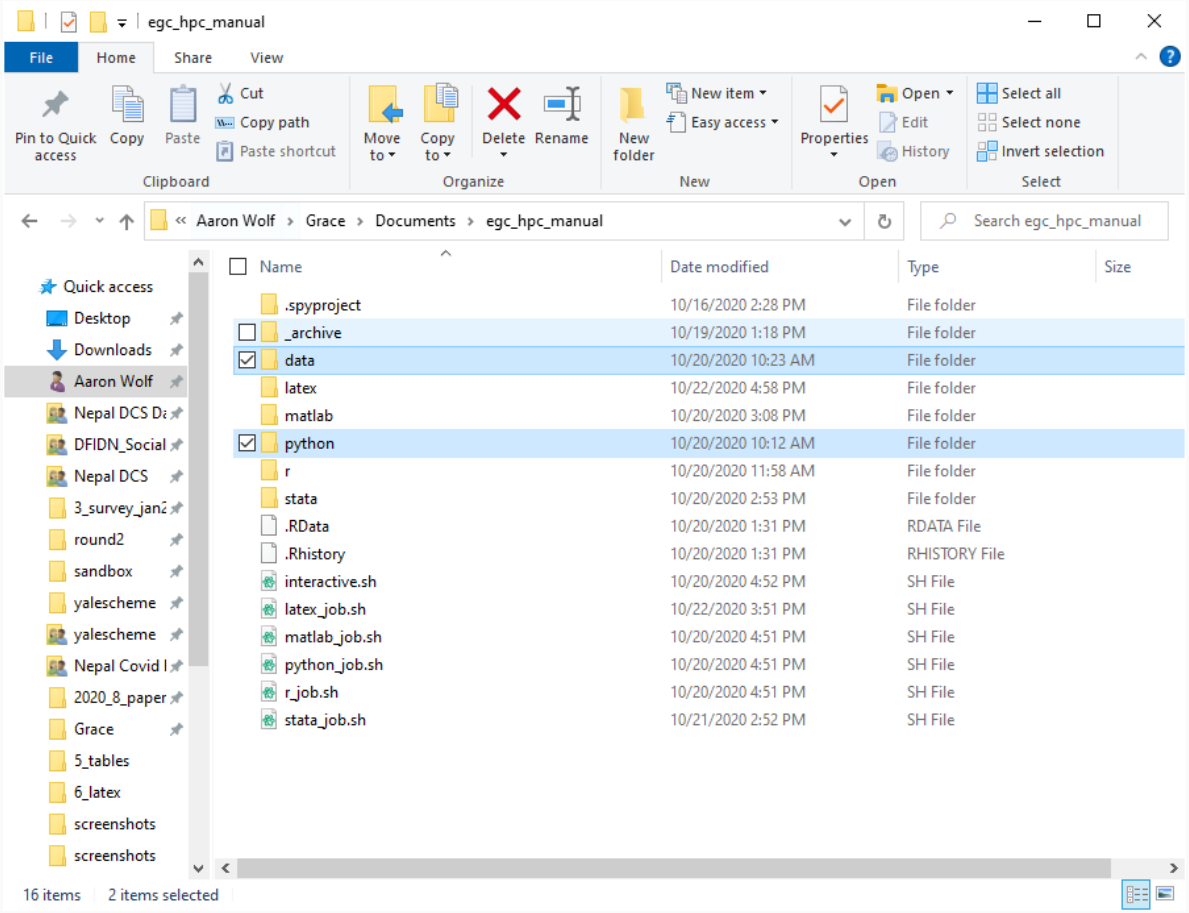
We will set it up so that you can run the .py file on your own computer as well as Grace (once the files are synchronized).

Then we will create a batch script that will run the master .py file, which will itself run all the project .py files.

Step 1: Create project folder

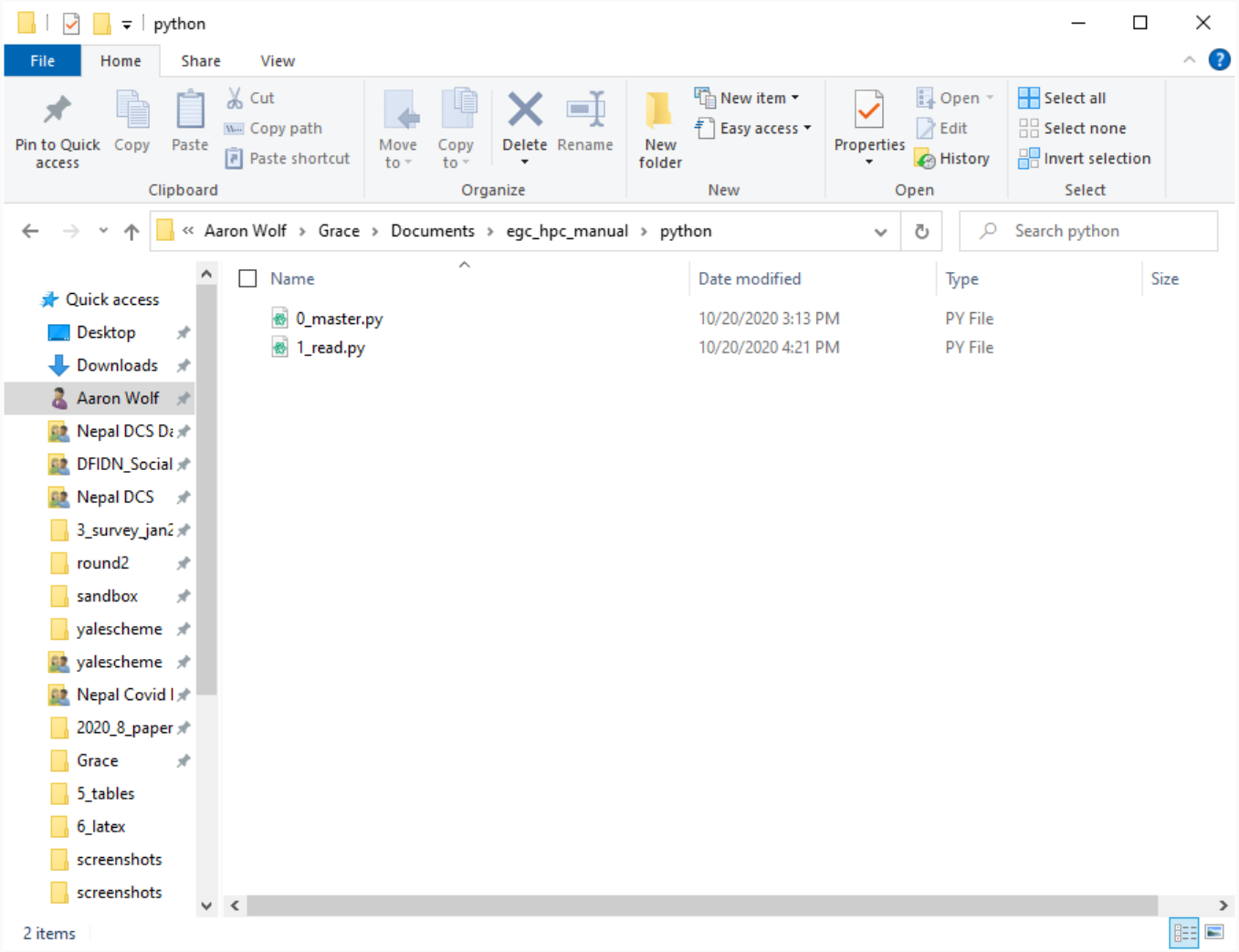
Create a new folder on your PC anywhere you like. This will be your main (root) folder.

Add folders for data and python files.



Step 2: Create .py files

Create a master .py file, as well as a sub-files (we'll call this one *1_read.py*, as we will only use it to read in the auto dataset). The following slides will provide the content of these files.



Step 2: Create .py file - 0_master.py

Listing 6: 0_master.py

```
# -*- coding: utf-8 -*-
"""
Filename: 0_master.py


Purpose:    This is the master .py file for <PROJECT NAME>.


Created by: adw54
Created on: 7 Oct 2020
Last updated on: 7 Oct 2020
"""
```

Step 2: Create .py file - 0_master.py

Listing 6: 0_master.py

```
##############################################################################
#
# SECTION 1: Directories
#
##############################################################################

## Set-Up Working Directories
if getpass.getuser() == 'adw54' and platform.system() == "Windows":
    root = os.path.abspath("C:/Users/adw54/Pande Research Dropbox/Aaron
        Wolf/Grace/Documents/egc_hpc_manual")
elif getpass.getuser() == 'adw54' and platform.system() == "Linux":
    root = os.path.abspath("/home/adw54/Documents/egc_hpc_manual")
else:
    print('No User Detected')

os.chdir(root)

print("Setup Complete!")

##############################################################################
#
# SECTION 2: Run Sub-Files
#
##############################################################################

exec(open("python/1_read.py").read())
```

Step 3: Create sub files

Now we can make our subfiles. You can put anything you want here as a test.

For now, we will just load the auto dataset (saved in the data folder).

Step 3: Create sub-file 1_read.py

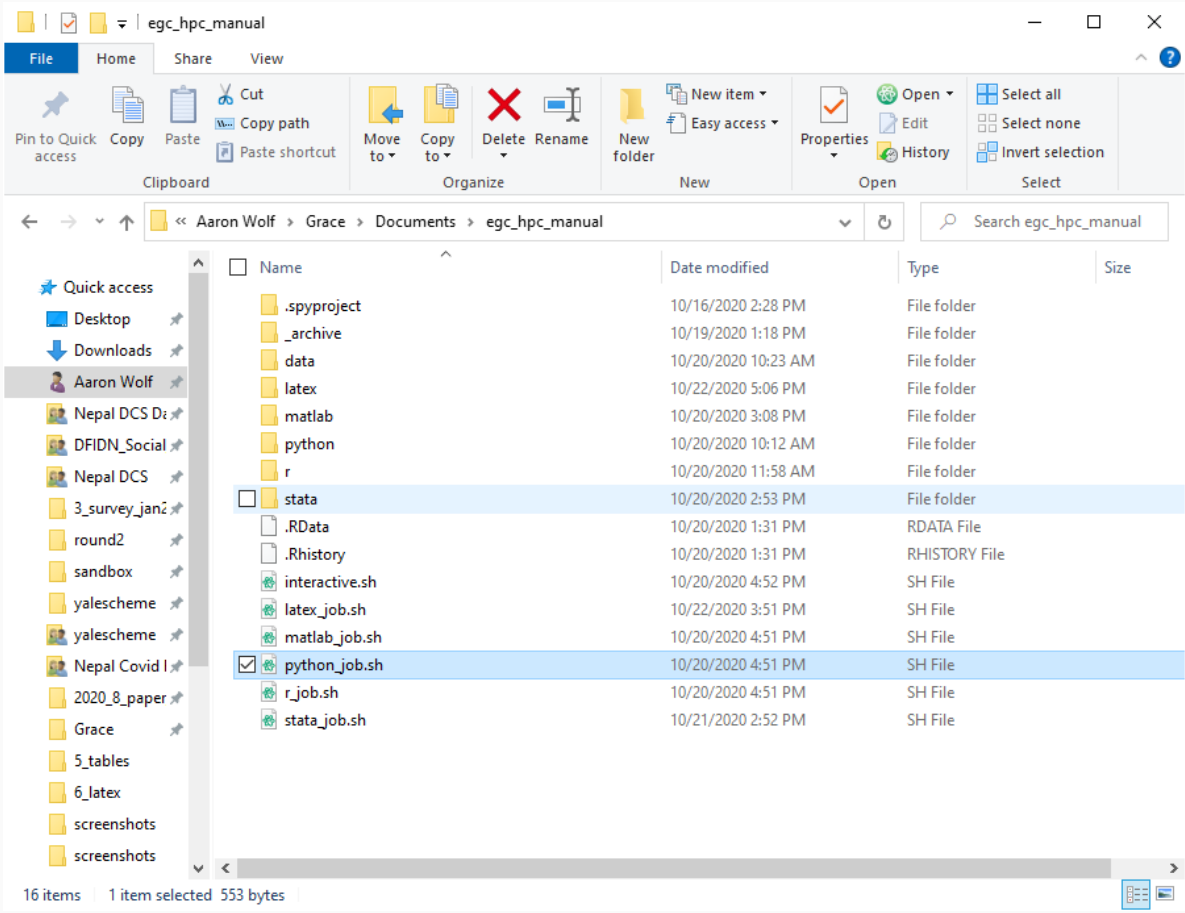
Listing 7: 1_read.do

```
# -*- coding: utf-8 -*-
"""
Filename: 1_read.py


Purpose:    This .py file reads auto.dta


Created by: adw54
Created on: 7 Oct 2020
Last updated on: 7 Oct 2020
"""

import os
import pandas as pd
os.chdir(root)

df = pd.read_stata('data/auto.dta')
print("Data Loaded!")
```

Step 4: Create batch script

Next we will write a batch script which will run our master .py file.

I will place this close in the project root directory, but you can also put it in your default working directory on Grace, as long as you remember where it is when you use `sbatch` to call it.



Step 4: Create batch script - python_job.sh

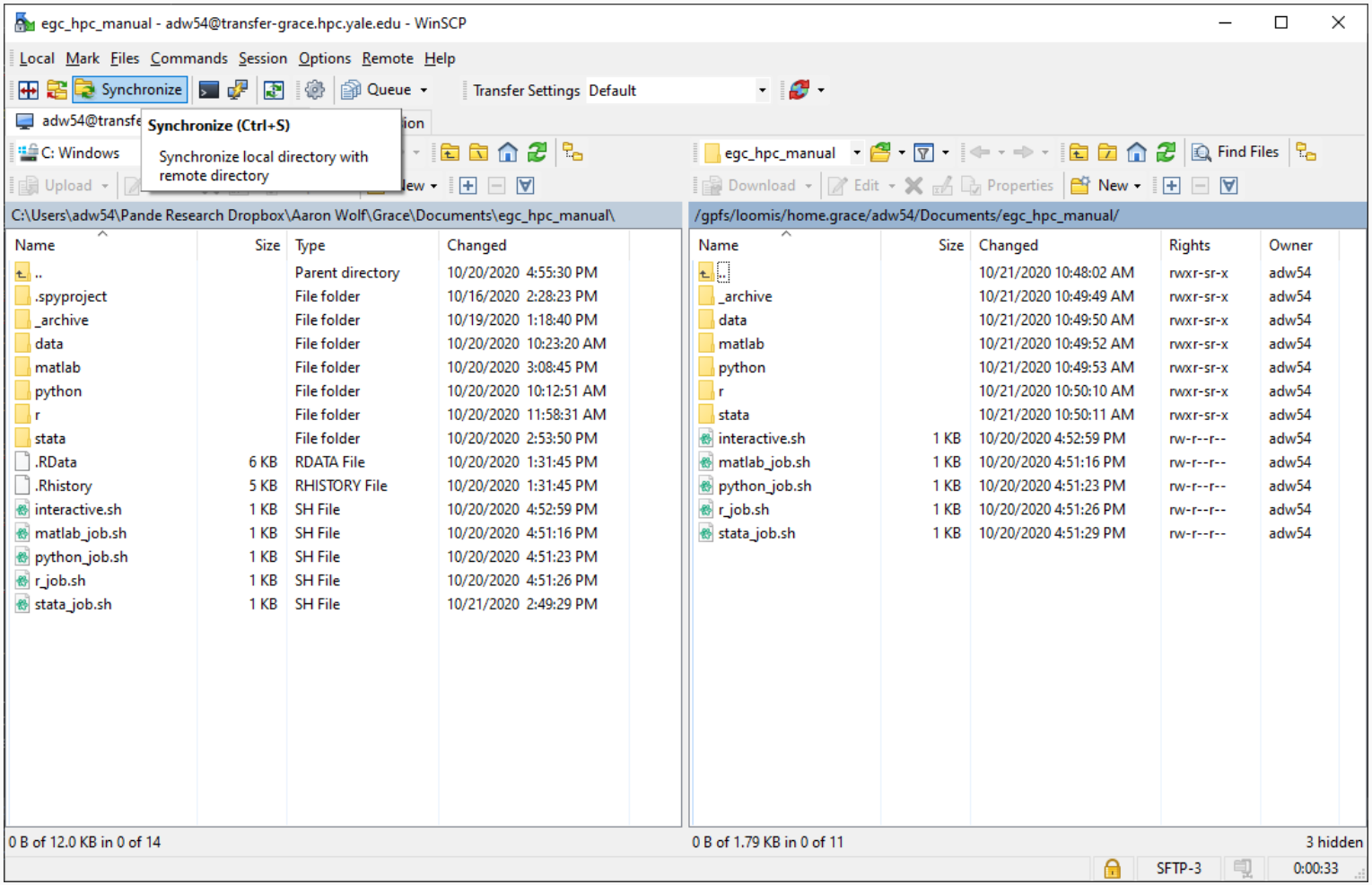
Note: I have included the line to create the new python environment. If you have not already created it, make sure to include this (or create it in a separate interactive session).

Listing 8: python_job.sh

```
#!/bin/bash
#SBATCH --job-name=python_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# load the miniconda module
module load miniconda

# create the environment with the required libraries
#conda create -n py3_base python=3 numpy scipy matplotlib pandas ipython jupyter jupyterlab

# activate the environment and run master py script
conda activate py3_base
python python/0_master.py
```

Step 6: Synchronize your root folder to your Grace project folder

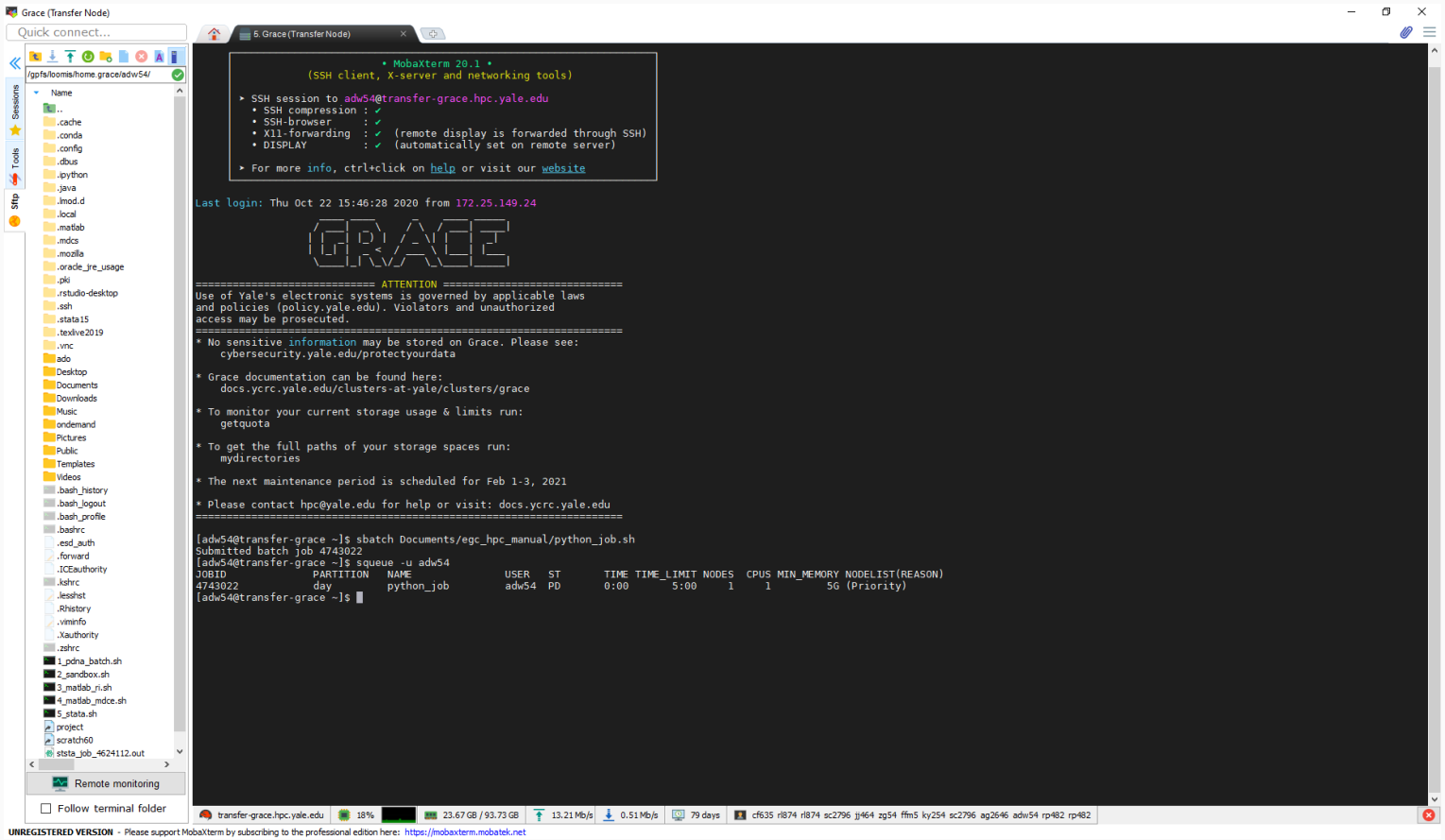
Navigate to wherever you want your root directory to exist on Grace in WinSCP and select "Synchronize". Choose your files and click "OK".



Step 7: Run batch file

Open MobaXterm and input the following code (substituting the path for your .sh file):

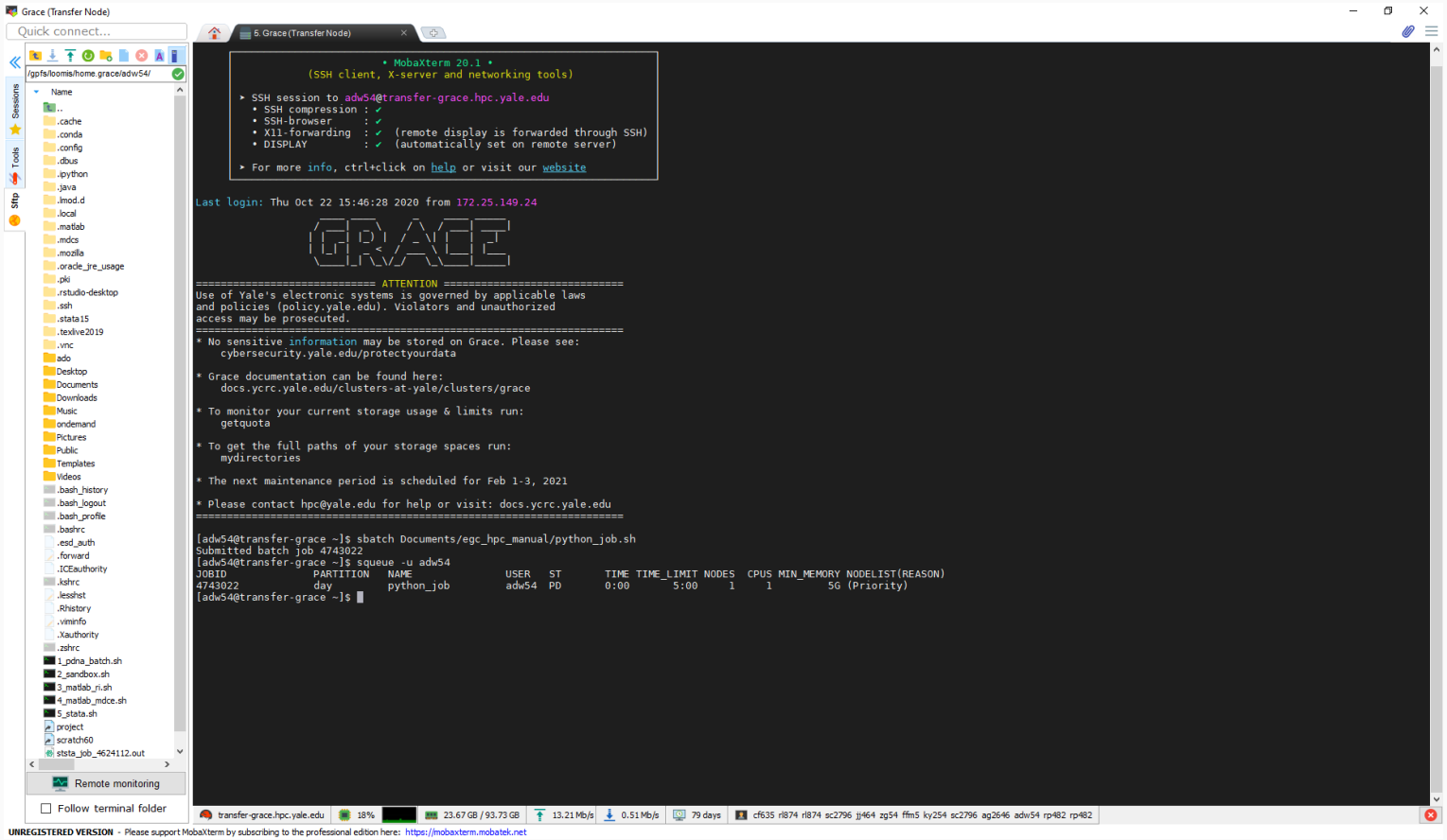
`sbatch Documents/egc_hpc_manual/python_job.sh`



Step 7: Run batch file

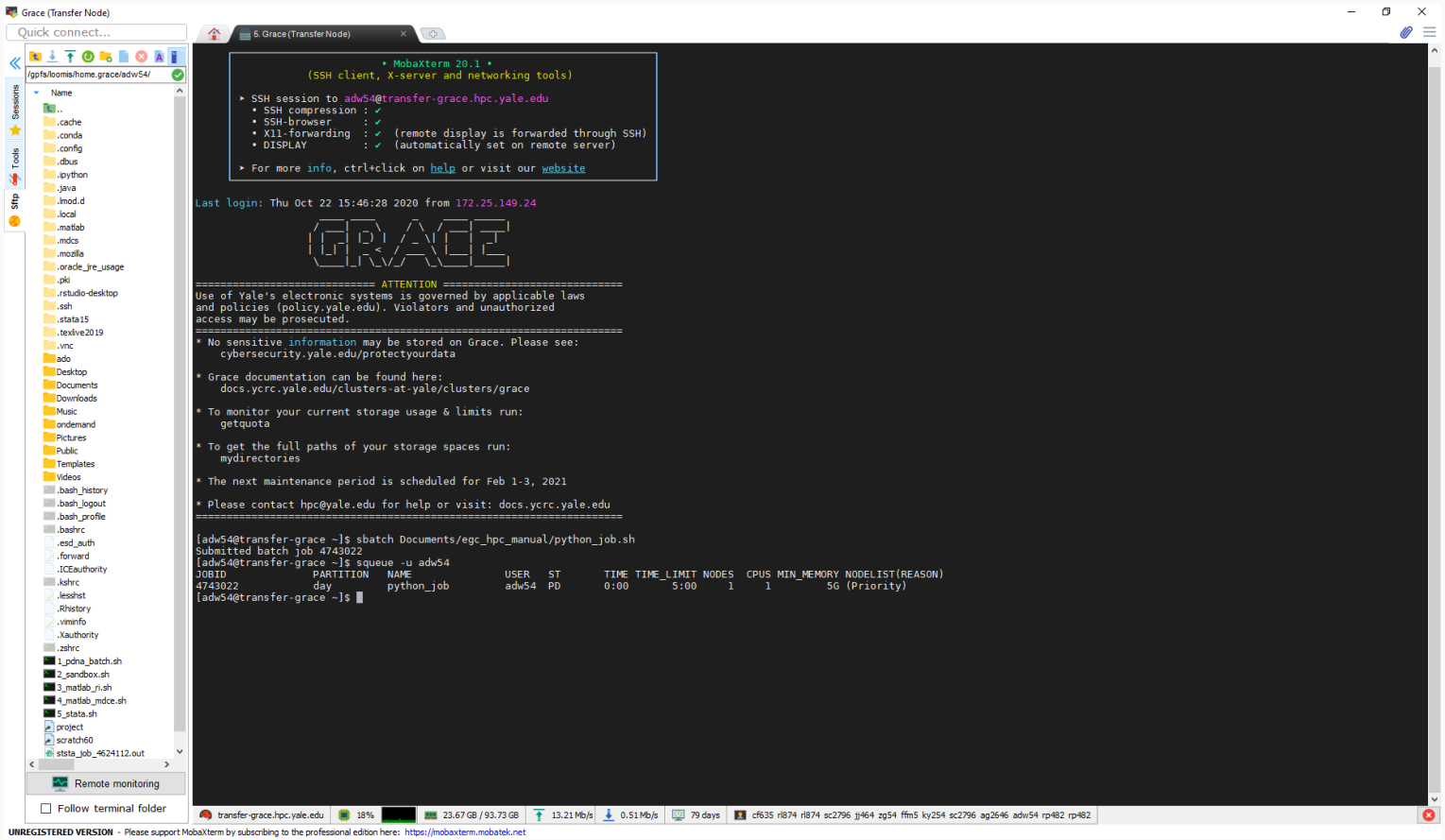
You should see a message indicating the job was submitted. You can now input the following command (insert your username) to monitor the status of your job requests:

`squeue -u abc123`



Step 7: Run batch file

When Grace finishes your file, you will receive an email, and an output file in your home directory with the results (i.e. output screen). This can be used to diagnose errors and ensure everything ran.



That's it!

- When starting out, there are likely to be unforeseen errors/issues.
- We recommend starting with a similar example project and submitting batch scripts that ask for minimal resources to ensure you get all the paths right.
- Once the script runs smoothly without errors, switch the .py files called to the real ones and adjust the resource request accordingly.

R

HPC: but with R (!)

- R code can be run directly from the command line using the `R --slave -f myscript.R` syntax.
- However, similar to Stata, where we had to "load" Stata MP (`module load Stata/15`), we need to tell Grace to find and use an R installation.
- For this, we use an Anaconda R environment.

What's Anaconda?

- If you've never used Anaconda think of it this way: Anaconda is to Python (and R!) what the App Store is to Apps. It is a single place to manage your installations of Python and R.
- Anaconda allows you to set up and control multiple R "Environments": Separate installations of R with different characteristics (e.g. you could have one environment on your computer with r-essentials, and another with specific machine learning libraries installed). In each environment, you can install all the libraries/packages you usually would with `install.packages`.
- Anaconda is free for individual use: Check out https://www.anaconda.com/.
- **Before you can use R on Grace, you need to first create an R environment.** You can do this on a login node or an interactive node.

Create a new R Environment called “r3_base”

Open MobaXTerm, login, and start a new interactive session:

```
srun --pty -c 4 -p interactive -t 4:00:00 bash
```

Create a new R Environment called “r3_base”

Load the `miniconda` module:

`module load miniconda`

Create a new R Environment called “r3_base”

Create a new R environment with a few basic data science packages installed by default:

```
conda create --name r3_base r-base r-essentials r-doMC r-Rmpi
```

Create a new R Environment called “r3_base”

Now, whenever you want to use R, whether in an interactive session or batch file, simply load up miniconda, and activate the environment:

```
module load miniconda
conda activate r3_base
```

Create a new R Environment called “r3_base”

You can even start an R session for interactive R!

R

Create a new R Environment called “r3_base”

When you’re done, deactivate the current environment:

`conda deactivate`

R Example Project

R Example Project

Now, let's create a new R project with a batch script, master .R files, and .R files with the core analysis commands.

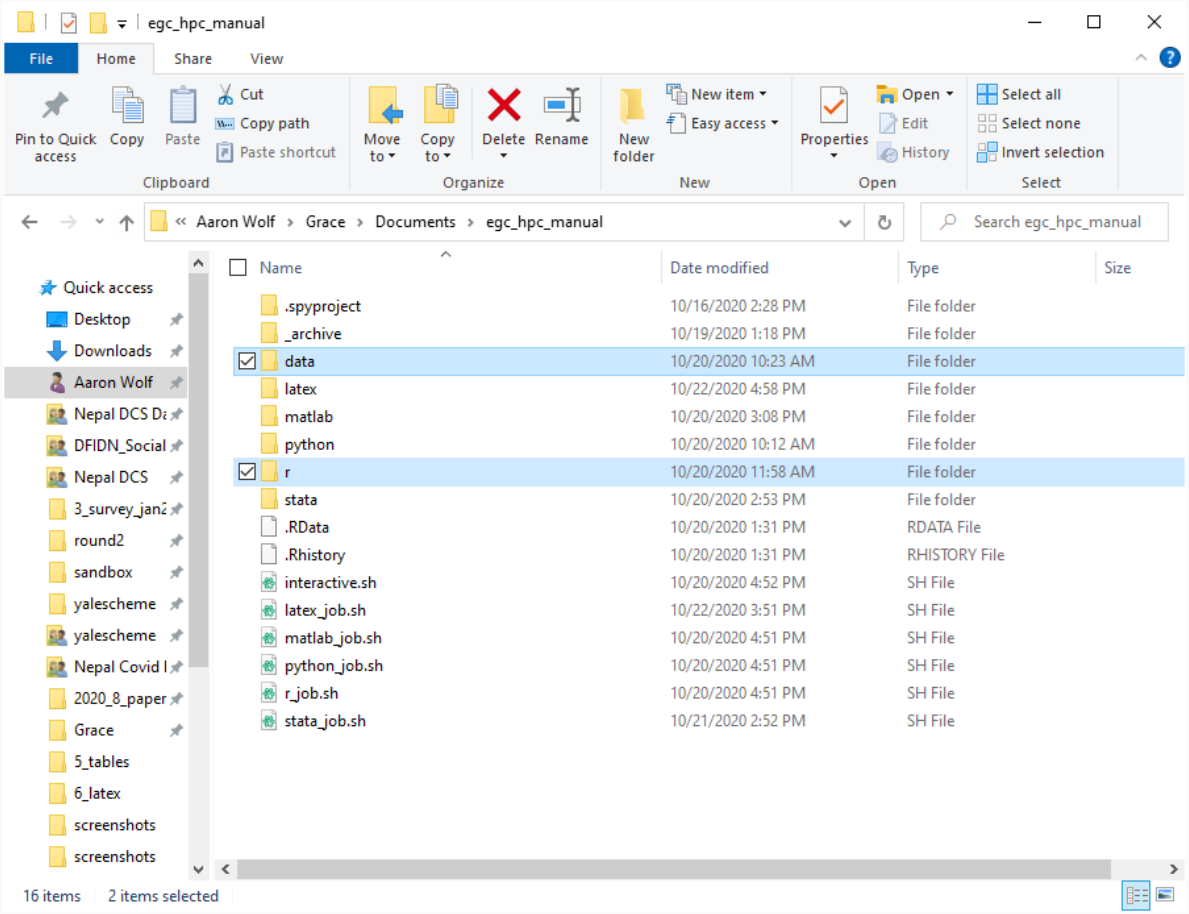
We will set it up so that you can run the .R file on your own computer as well as Grace (once the files are synchronized).

Then we will create a batch script that will run the master .R file, which will itself run all the project .R files.

Step 1: Create project folder

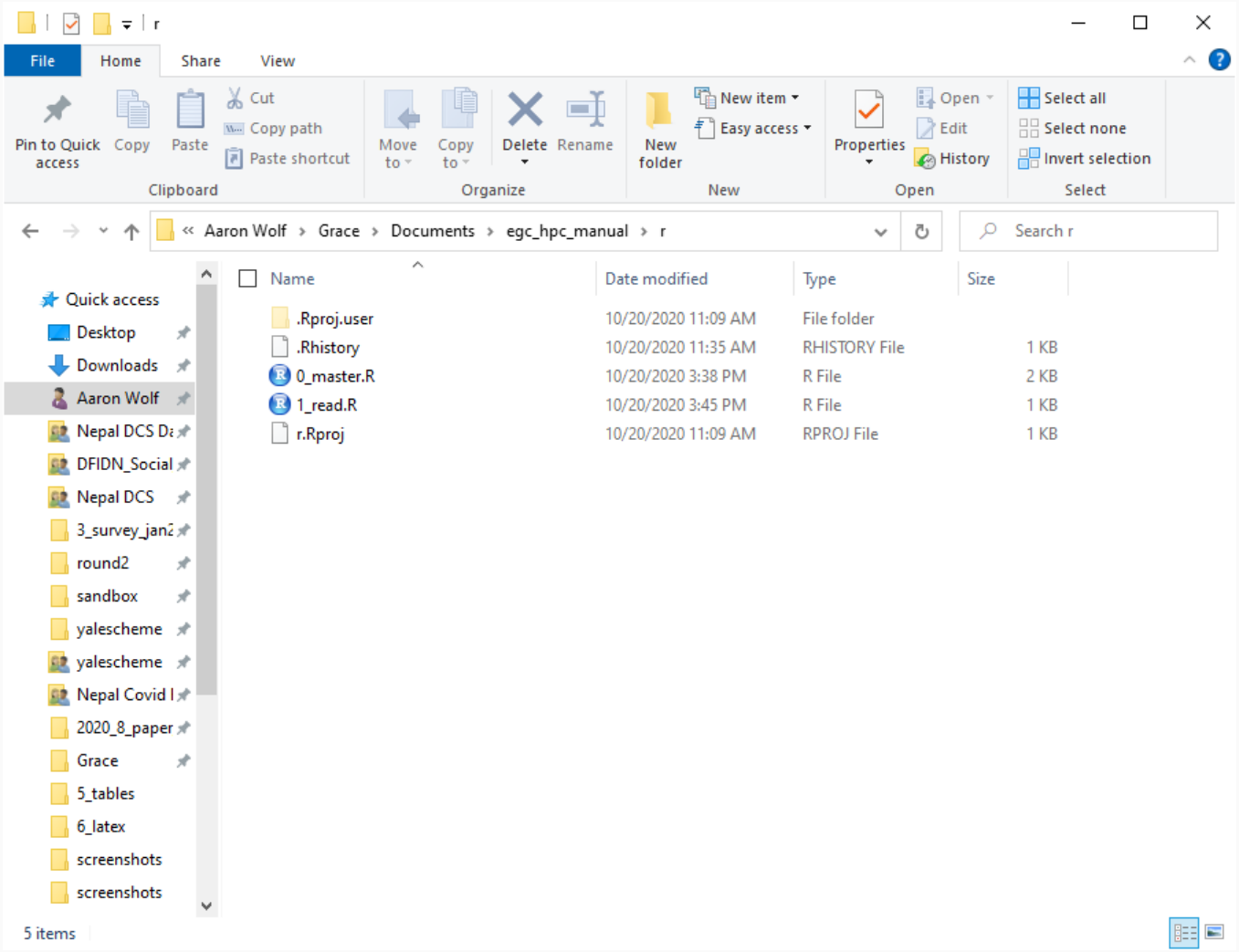
Create a new folder on your PC anywhere you like. This will be your main (root) folder.

Add folders for data and R files.



Step 2: Create .R files

Create a master .R file, as well as a sub-files (we'll call this one *1_read.R*, as we will only use it to read in the auto dataset). The following slides will provide the content of these files.



Step 2: Create .R file - 0_master.R

Listing 9: 0_master.R

```
##-----------------------------
##
## Script name: 0_master.r
##
##
## Purpose of script: This is the master .r file for <PROJECT NAME>.
##
##
## Created by: adw54
## Created on: 7 Oct 2020
## Last updated on: 7 Oct 2020
##-----------------------------
```

Step 2: Create .R file - 0_master.R

Listing 9: 0_master.R

```
#############################################################################################
#
# SECTION 1: Directories
#
#############################################################################################

## Set-Up Working Directories
if (Sys.getenv("USERNAME") == "adw54" & Sys.getenv("HOME") == "C:/Users/adw54/Documents") {
  root <- "C:/Users/adw54/Pande Research Dropbox/Aaron Wolf/Grace/Documents/egc_hpc_manual"
} else if (Sys.getenv("USER") == "adw54" & Sys.getenv("HOME") == "/home/adw54") {
  root <- "/home/adw54/Documents/egc_hpc_manual"
} else {
  print('No User Detected')
}
setwd(root)

print("Setup Done!")

#############################################################################################
#
# SECTION 2: Run Sub-Files
#
#############################################################################################

source("r/1_read.R")
```

Step 3: Create sub files

Now we can make our subfiles. You can put anything you want here as a test.

For now, we will just load the auto dataset (saved in the data folder).

Step 3: Create sub-file 1_read.R

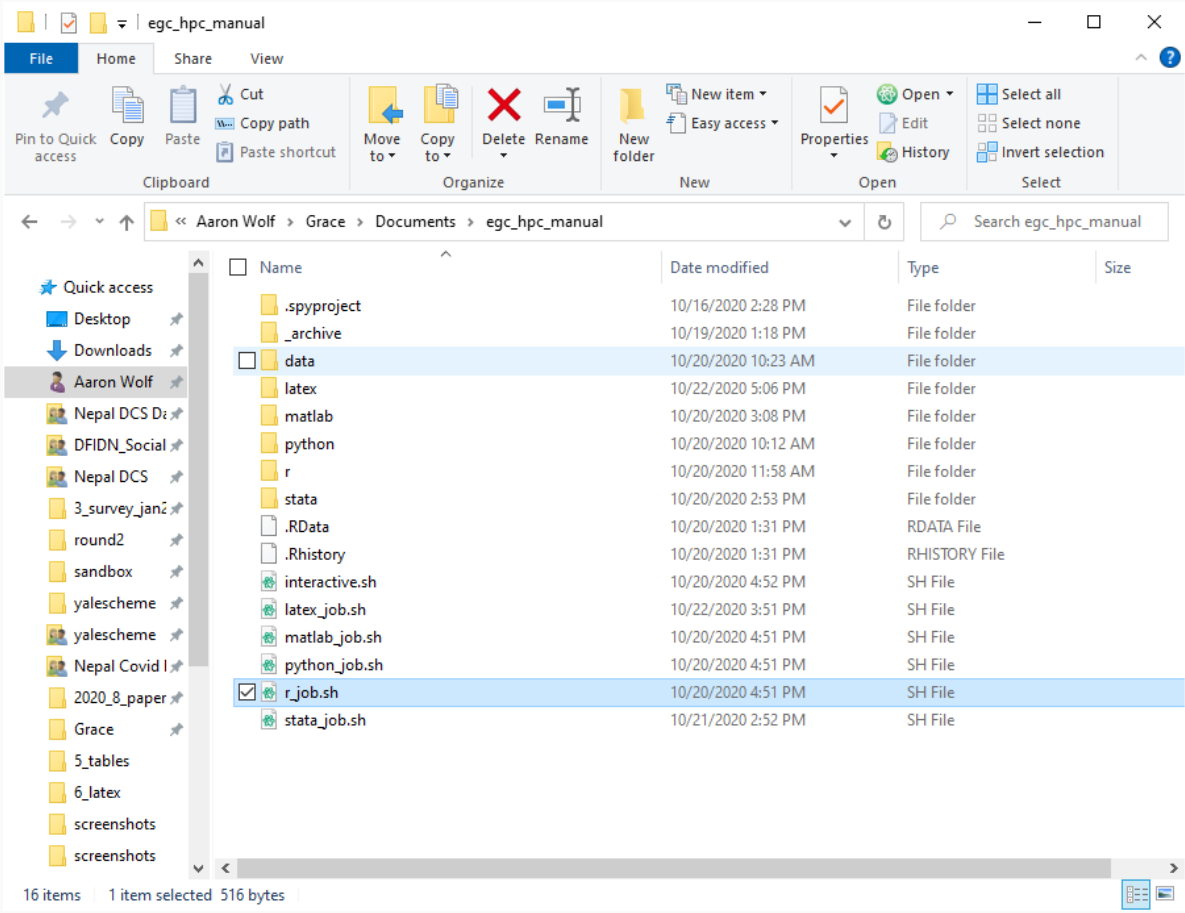
Listing 10: 1_read.do

```
###############################################################################
#
# Filename: read.m
#
#
# Purpose: This script reads auto.csv
#
# Created by: adw54
# Created on: 7 Oct 2020
# Last updated on: 7 Oct 2020
#
###############################################################################

library( haven)

## Read data
library( haven)
df <- read_dta("data/auto.dta")
print("Data Loaded!")
```

Step 4: Create batch script

Next we will write a batch script which will run our master .R file.

I will place this close in the project root directory, but you can also put it in your default working directory on Grace, as long as you remember where it is when you use `sbatch` to call it.



Step 4: Create batch script - r_job.sh

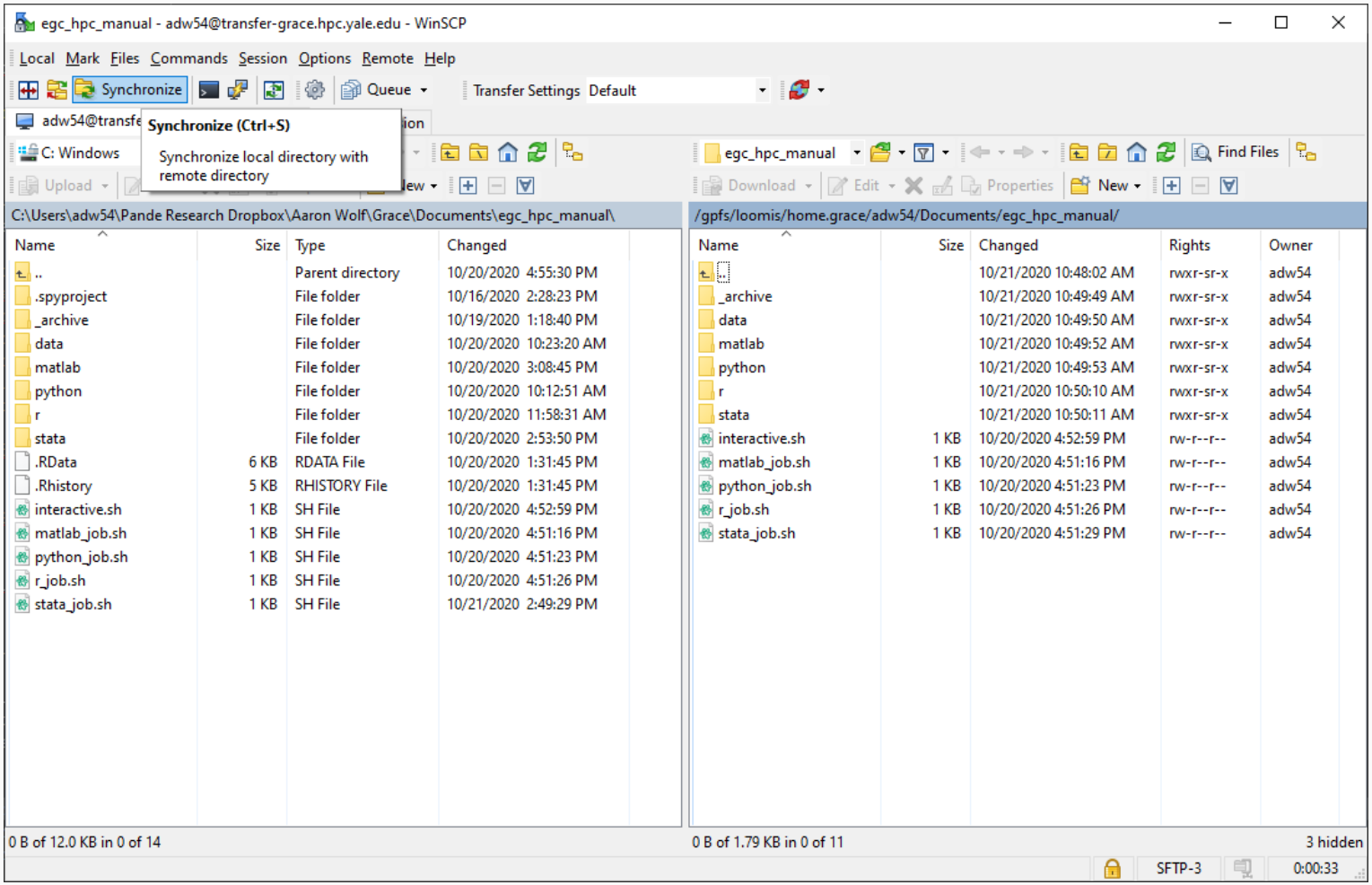
Note: I have included the line to create the new R environment. If you have not already created it, make sure to include this (or create it in a separate interactive session).

Listing 11: r_job.sh

```
#!/bin/bash
#SBATCH --job-name=r_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# load the miniconda module
module load miniconda

# create the environment with the required packages
#conda create --name r3_base r-base r-essentials r-doMC r-Rmpi

# activate the environment and run master R script
conda activate r3_base
R --slave -f r/0_master.R
```

Step 6: Synchronize your root folder to your Grace project folder

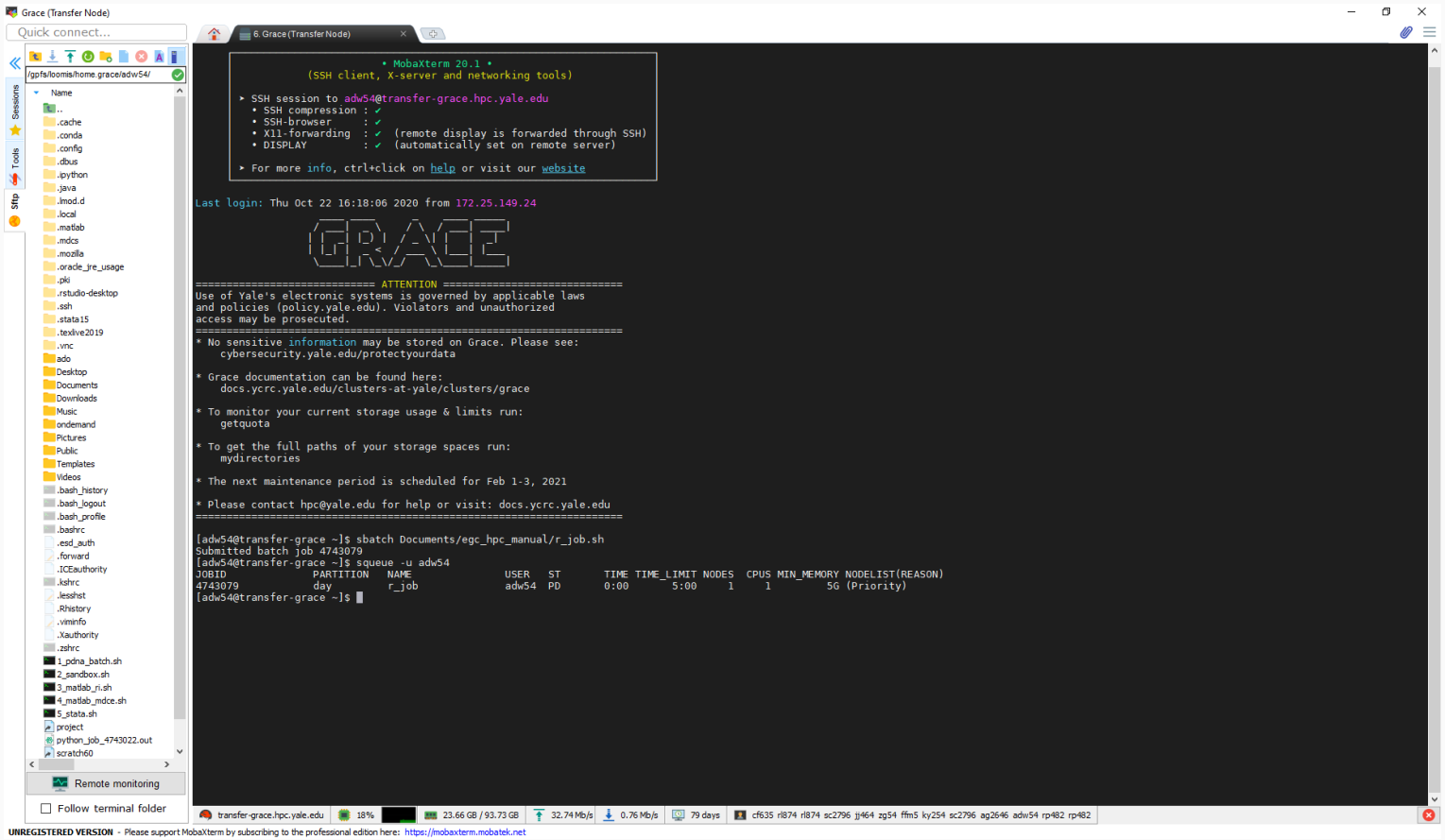
Navigate to wherever you want your root directory to exist on Grace in WinSCP and select "Synchronize". Choose your files and click "OK".



Step 7: Run batch file

Open MobaXterm and input the following code (substituting the path for your .sh file):

```
sbatch Documents/egc_hpc_manual/r_job.sh
```



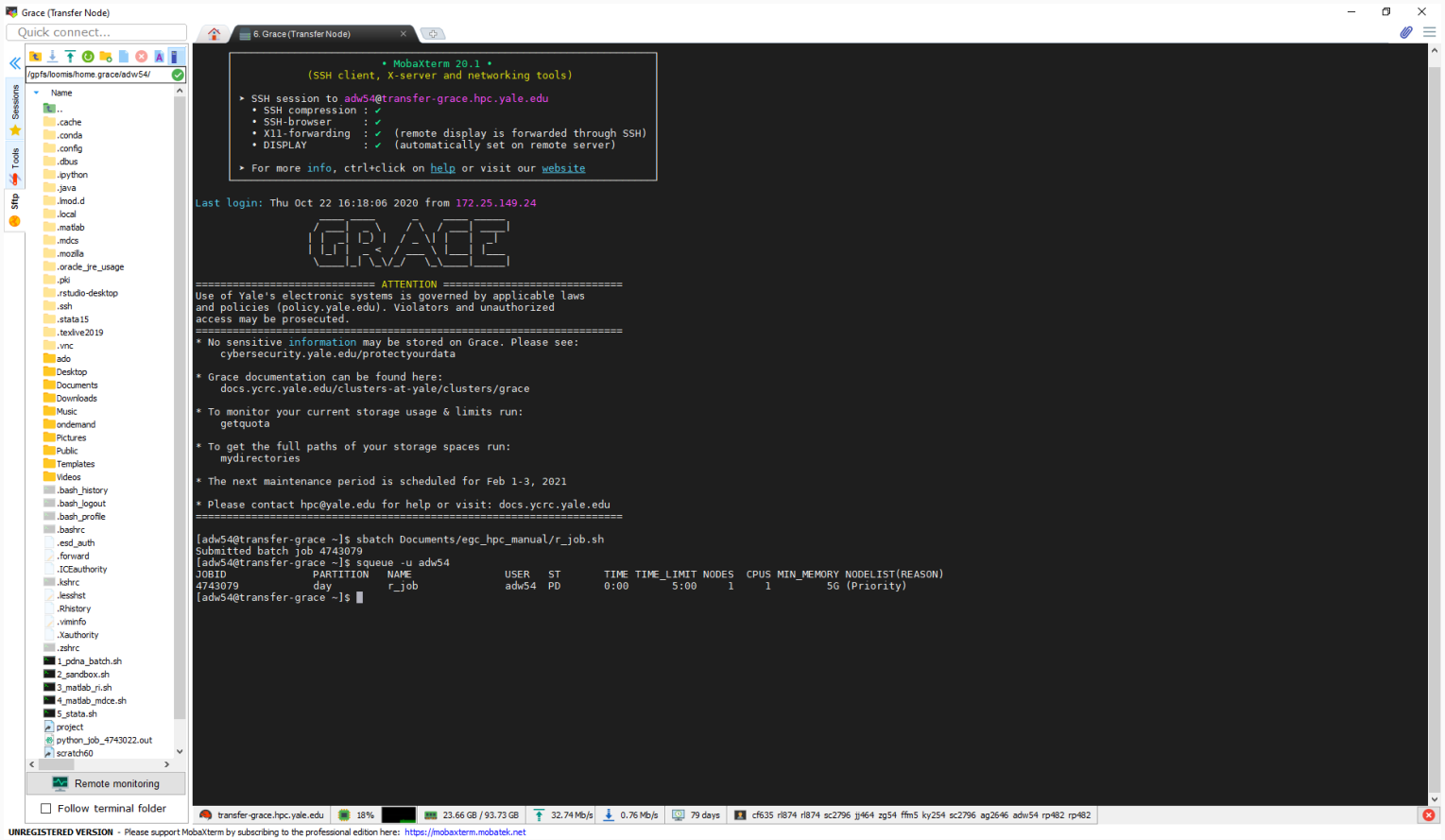
Step 7: Run batch file

You should see a message indicating the job was submitted. You can now input the following command (insert your username) to monitor the status of your job requests:

`squeue -u abc123`

Step 7: Run batch file

When Grace finishes your file, you will receive an email, and an output file in your home directory with the results (i.e. output screen). This can be used to diagnose errors and ensure everything ran.



That's it!

- When starting out, there are likely to be unforeseen errors/issues.
- We recommend starting with a similar example project and submitting batch scripts that ask for minimal resources to ensure you get all the paths right.
- Once the script runs smoothly without errors, switch the .R files called to the real ones and adjust the resource request accordingly.

Matlab

HPC: but with Matlab (!)

- Matlab code can be run directly from the command line using the `matlab` command (for interactive matlab) or `matlab -nodisplay < myscript.m` (to run a specific file).
- However, similar to Stata, where we had to "load" Stata MP (`module load Stata/15`), we need to tell Grace to find and use a matlab installation.
- For this, we simply type in `module load MATLAB/2020a`.
- There are actually multiple flavors of Matlab available. You'll likely want to stick with the latest, but if you need to use Matlab parallel, you can use `module load MATLAB/2019a-parallel`.
- Check out all available versions of Matlab using `module spider Matlab`
- Let's try running Matlab from the command line in Grace...

Open MobaXTerm, login, and start a new interactive session:

```
srun --pty -c 4 -p interactive -t 4:00:00 bash
```

Load the `miniconda` module:

`module load MATLAB/2019a-parallel`

Run `matlab` and fiddle around!

```
matlab
pwd
```

Matlab Example Project

Matlab Example Project

Now, let's create a new R project with a batch script, master .R files, and .R files with the core analysis commands.

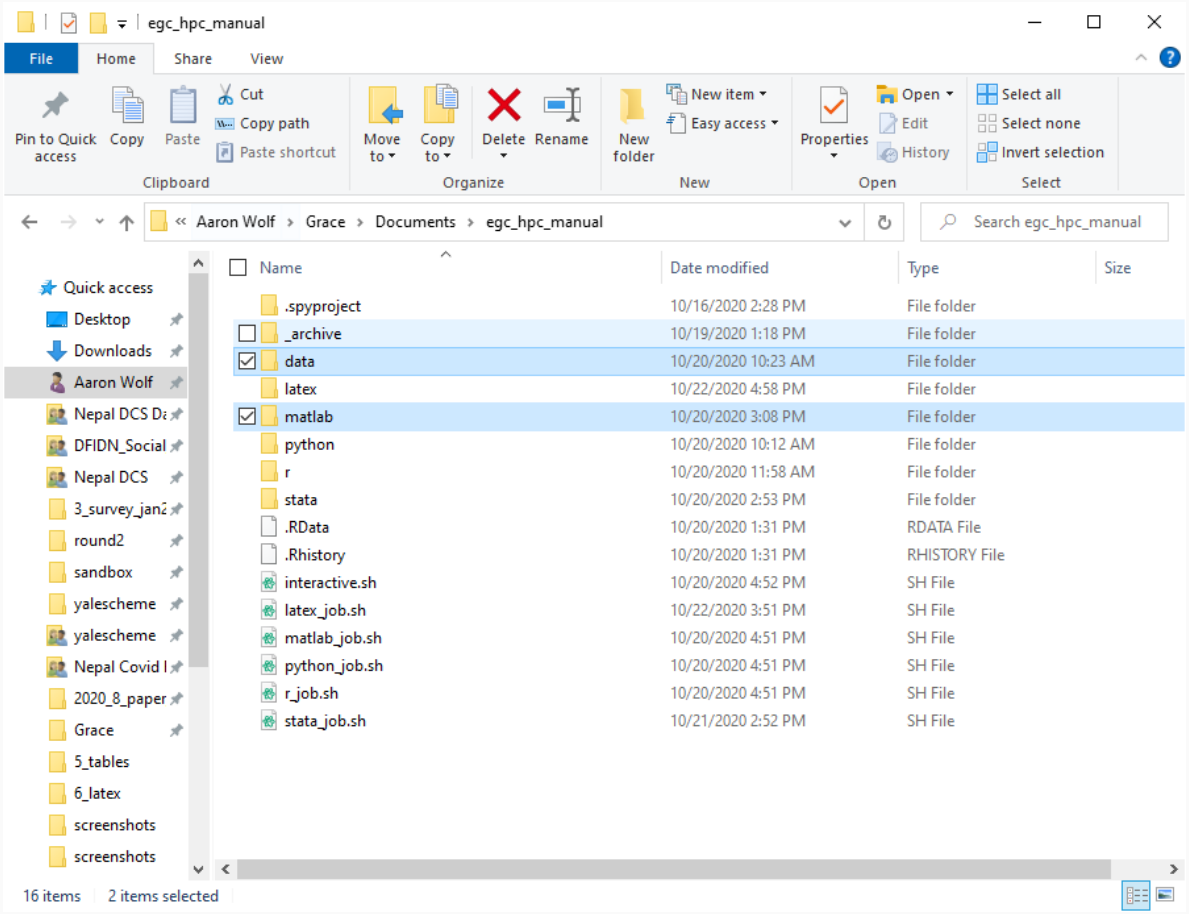
We will set it up so that you can run the .R file on your own computer as well as Grace (once the files are synchronized).

Then we will create a batch script that will run the master .R file, which will itself run all the project .R files.

Step 1: Create project folder

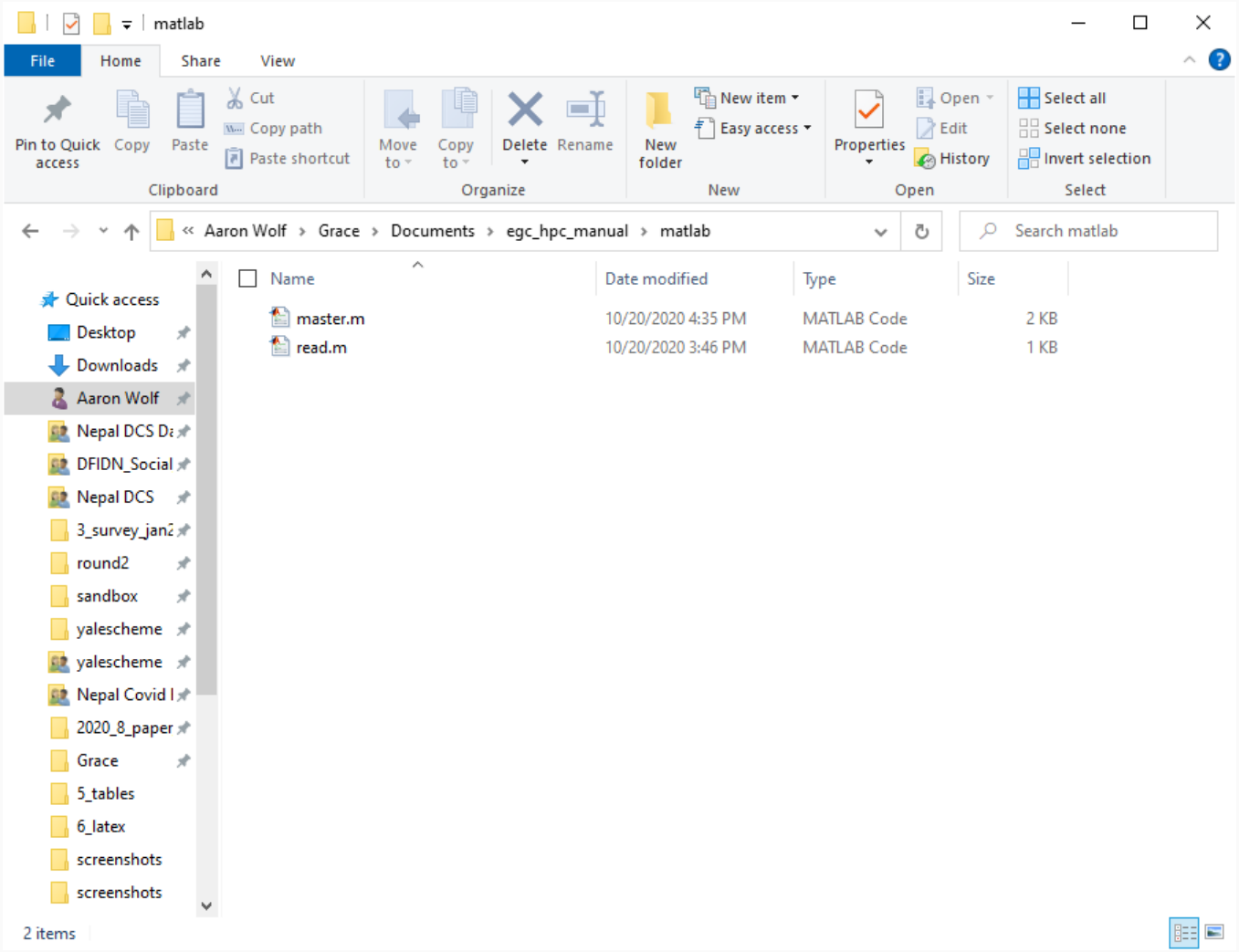
Create a new folder on your PC anywhere you like. This will be your main (root) folder.

Add folders for data and .m files.



Step 2: Create .m files

Create a master .m file, as well as a sub-files (we'll call this one *read.m*, as we will only use it to read in the auto dataset). The following slides will provide the content of these files.



Step 2: Create .m file - master.m

Listing 12: master.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Filename: 0_master.m
%
%
% Purpose: This is the master .m file for <PROJECT NAME>.
%
%
% Created by: adw54
% Created on: 7 Oct 2020
% Last updated on: 7 Oct 2020
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Step 2: Create .m file - master.m

Listing 12: 0_master.R

```
%------------------------------------------------------------------------
%
%% Section 1: Directories
%
%------------------------------------------------------------------------

% Set-Up Working Directories
if getenv('username') == "adw54" & ispc
    root = "C:/Users/adw54/Pande Research Dropbox/Aaron Wolf/Grace/Documents/egc_hpc_manual"
else if getenv('USER') == "adw54" & isunix
        root = "/home/adw54/Documents/egc_hpc_manual"
    end
end

cd(root)

disp('Setup Done!')

%------------------------------------------------------------------------
%
%% Section 2: Run sub-files
%
%------------------------------------------------------------------------

run 'matlab/read.m'
```

Step 3: Create sub files

Now we can make our subfiles. You can put anything you want here as a test.

For now, we will just load the auto dataset (saved in the data folder).

Step 3: Create sub-file read.m

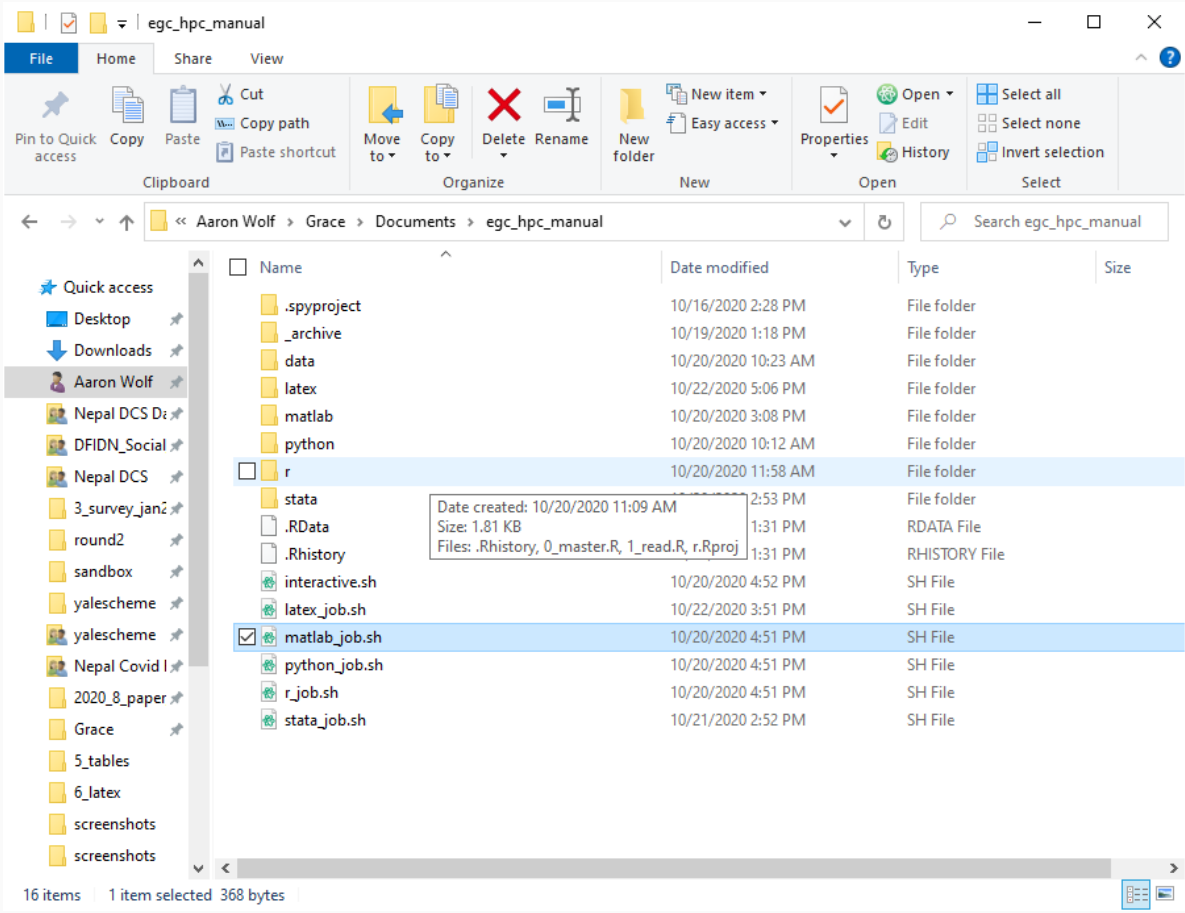
Listing 13: read.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Filename: read.m
%
%
% Purpose: This script reads auto.csv
%
% Created by: adw54
% Created on: 7 Oct 2020
% Last updated on: 7 Oct 2020
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Ensure CD is root
cd(root)

%% Read data
df = readtable('data/auto.csv');
disp('Data Loaded!')
```

Step 4: Create batch script

Next we will write a batch script which will run our master .m file.

I will place this close in the project root directory, but you can also put it in your default working directory on Grace, as long as you remember where it is when you use `sbatch` to call it.



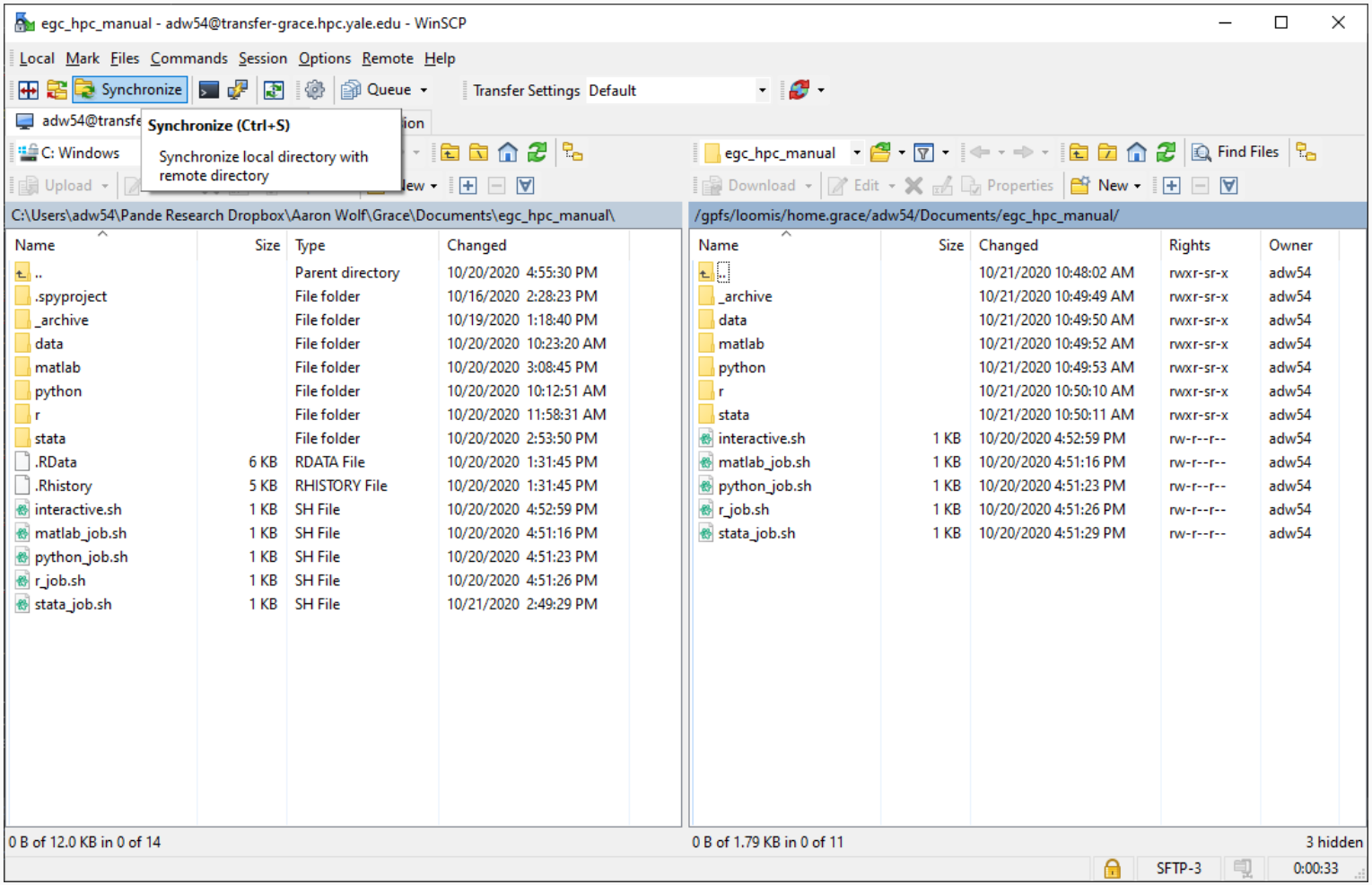
Step 4: Create batch script - matlab_job.sh

Listing 14: matlab_job.sh

```
#!/bin/bash
#SBATCH --job-name=matlab_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# Load matlab module and run master .m file
module load MATLAB/2019a-parallel
matlab -nodisplay < matlab/master.m
```

Step 6: Synchronize your root folder to your Grace project folder

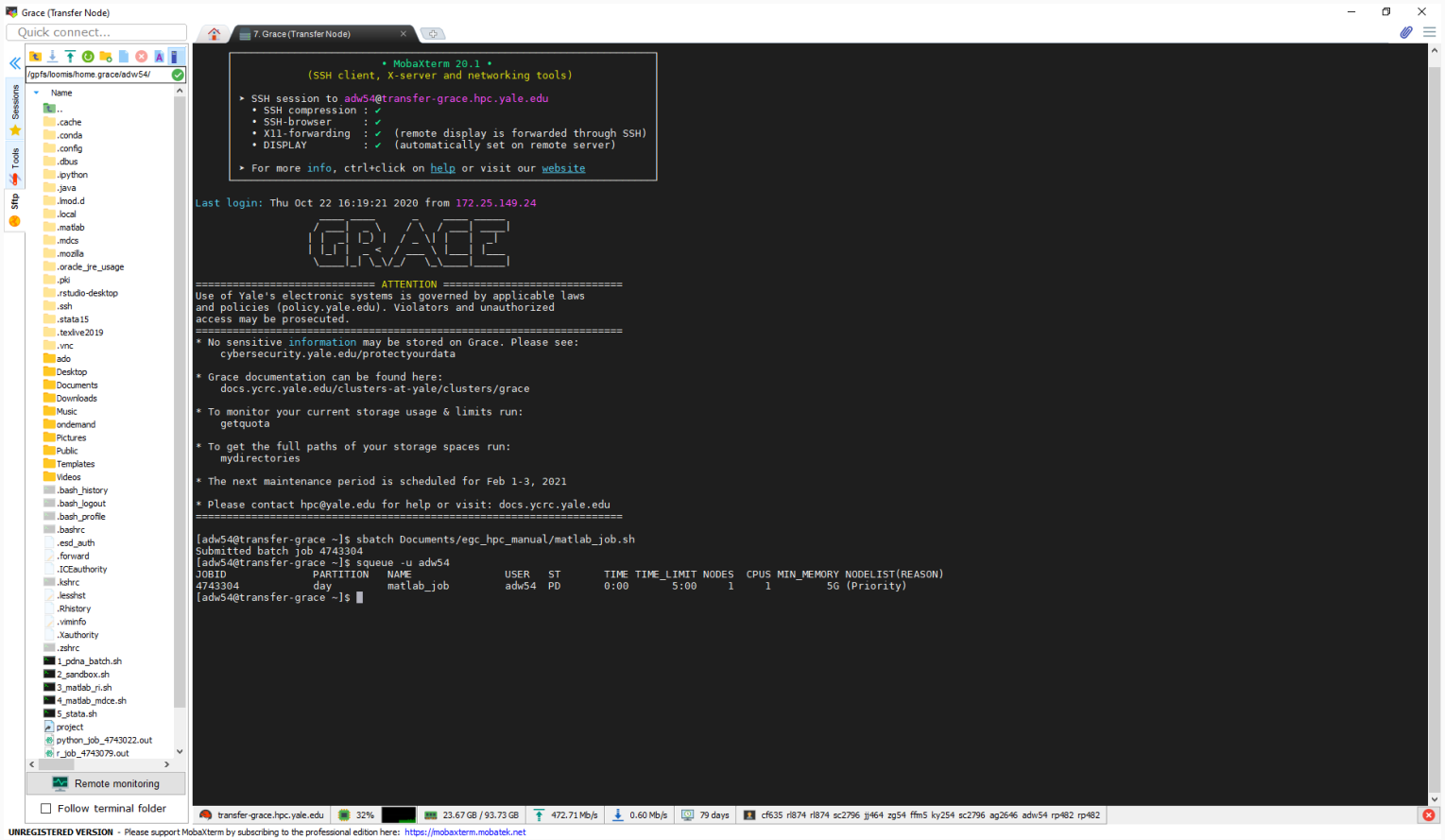
Navigate to wherever you want your root directory to exist on Grace in WinSCP and select "Synchronize". Choose your files and click "OK".



Step 7: Run batch file

Open MobaXterm and input the following code (substituting the path for your .sh file):

`sbatch Documents/egc_hpc_manual/matlab_job.sh`



Step 7: Run batch file

You should see a message indicating the job was submitted. You can now input the following command (insert your username) to monitor the status of your job requests:

`squeue -u abc123`

Step 7: Run batch file

When Grace finishes your file, you will receive an email, and an output file in your home directory with the results (i.e. output screen). This can be used to diagnose errors and ensure everything ran.

That's it!

- When starting out, there are likely to be unforeseen errors/issues.
- We recommend starting with a similar example project and submitting batch scripts that ask for minimal resources to ensure you get all the paths right.
- Once the script runs smoothly without errors, switch the .m files called to the real ones and adjust the resource request accordingly.

Yup. LaTeX

- You can use LaTeX on Grace to compile your documents!
- Grace has TeXLive available, and you can use the usual LaTeX commands.
- Since LaTeX documents are finicky, lets start by simply compiling the base sample2e.tex document.

Load LaTeX and compile sample2e document

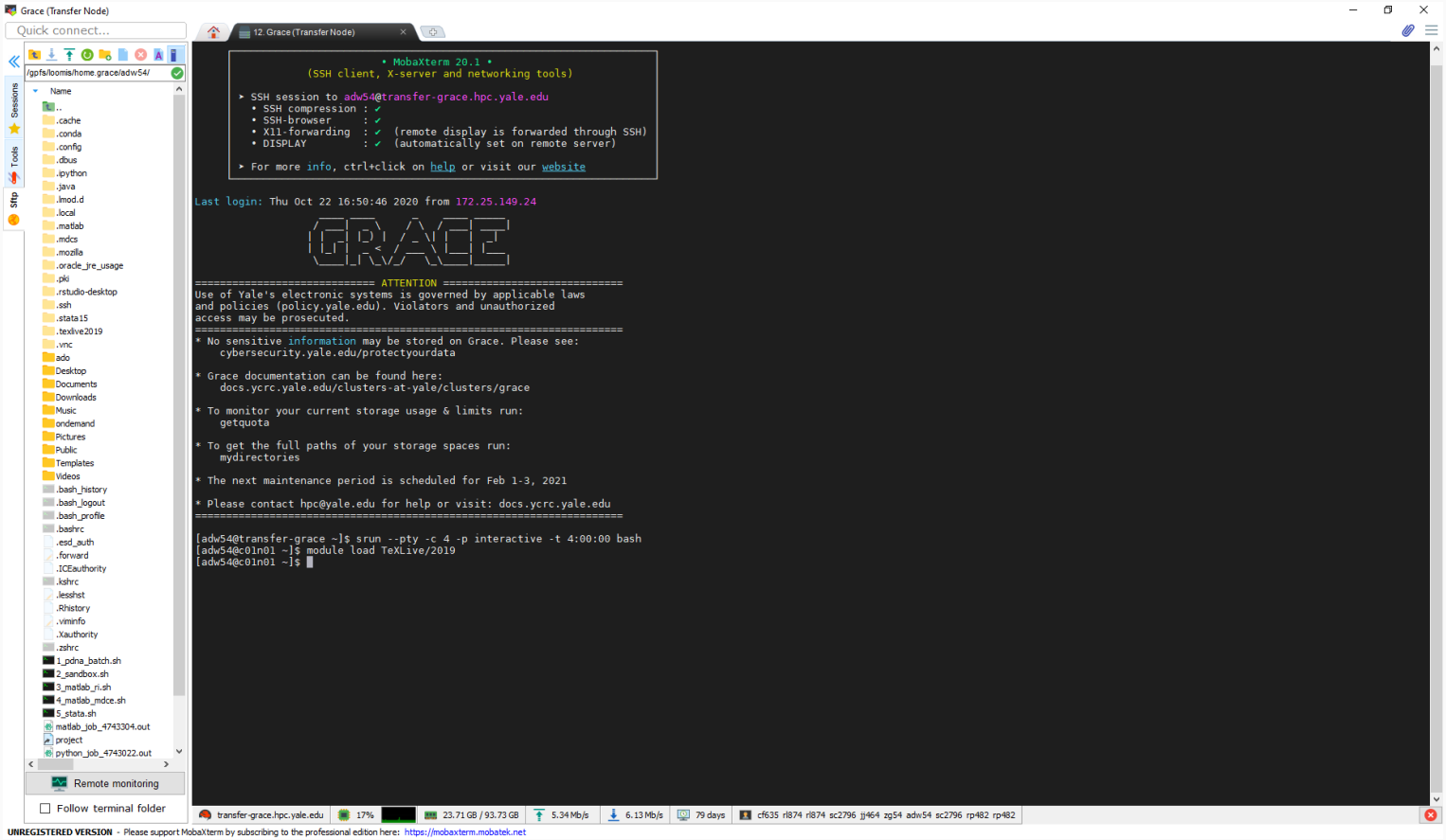
Open MobaXTerm, login, and start a new interactive session:

```
srun --pty -c 4 -p interactive -t 4:00:00 bash
```

Load LaTeX and compile sample2e document

Load the `TeXLive` module:

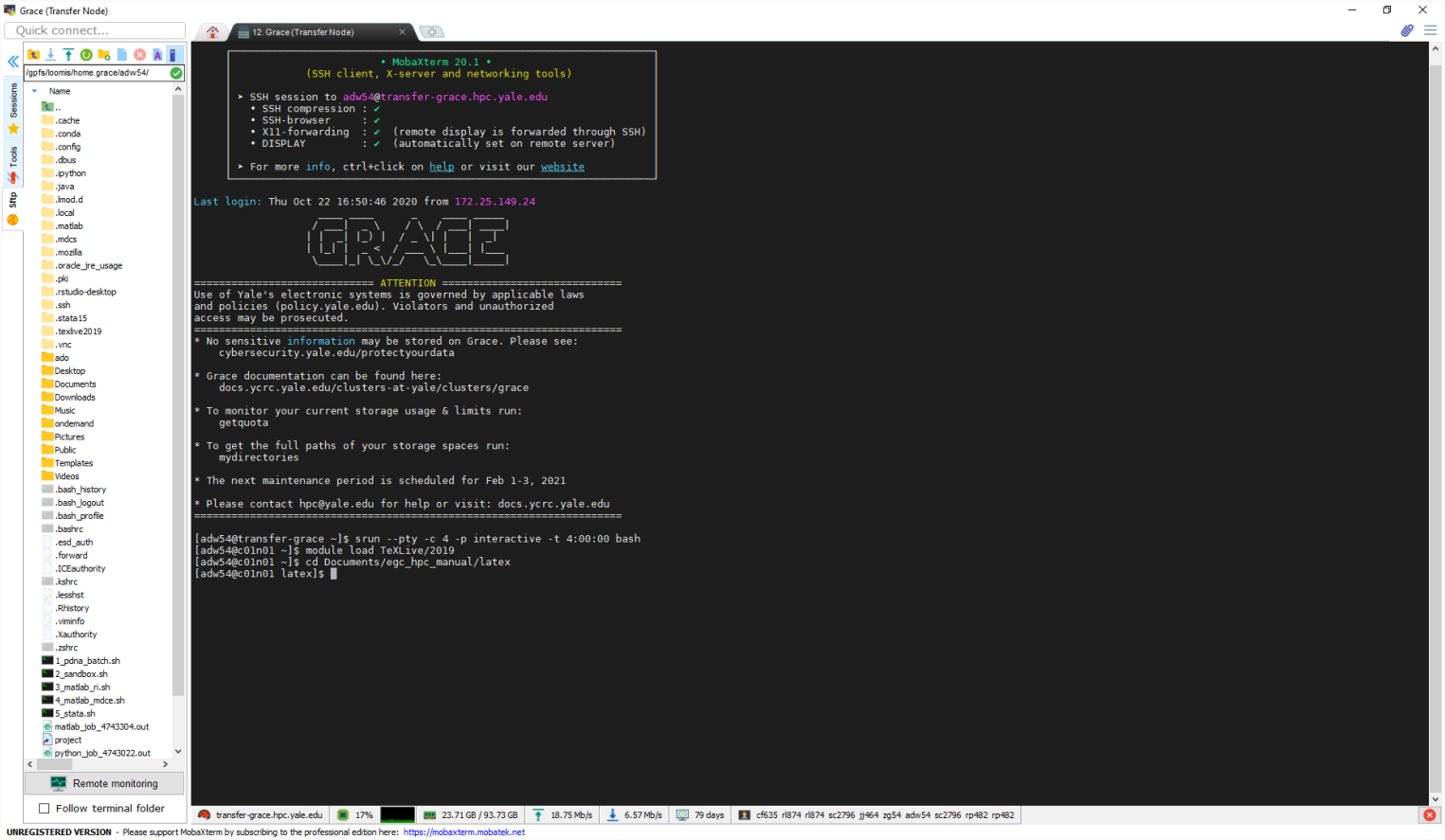
`module load TeXLive/2019`



Load LaTeX and compile sample2e document

CD to the directory where your LaTeX document is located, or where you want the sample2e document saved.

`cd Documents/egc_hpc_manual/latex`



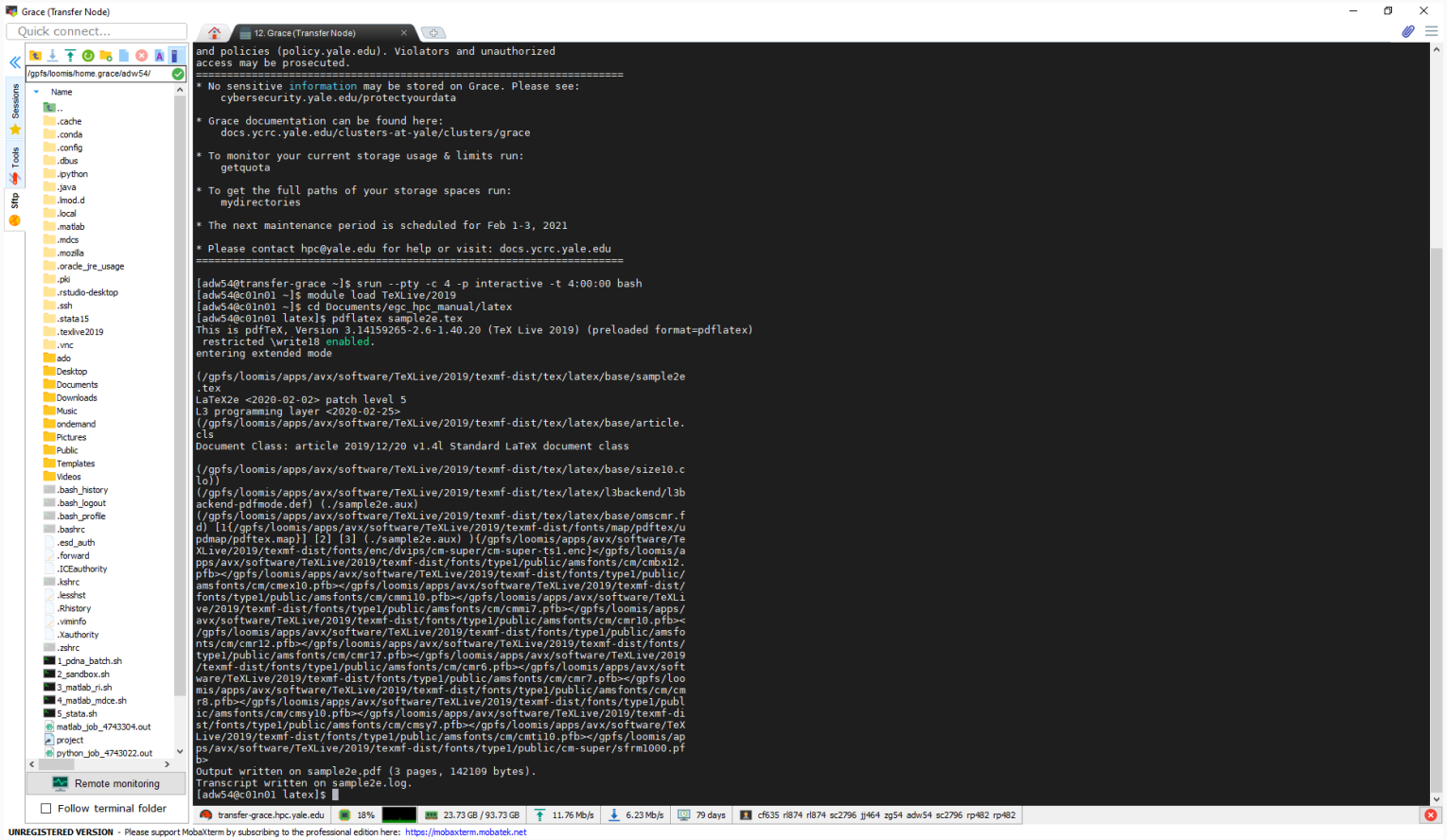
Load LaTeX and compile sample2e document

Compile your document

`pdflatex sample2e.tex`

Load LaTeX and compile sample2e document

When the document is finished compiling you will see it in the directory you specified.



Create a LaTeX batch script - latex_job.sh

Listing 15: latex_job.sh

```
#!/bin/bash
#SBATCH --job-name=latex_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual/latex

# Load Stata module and run master .do file
module load TeXLive/2019
pdflatex sample2e.tex
```

Zoo (Undergrads/CS Students)

Zoo vs HPC?

- For undergrads, there are two options in terms of setting up! (a) HPC Grace and (b) the zoo
- If you already have a zoo account, which you should if you've taken at least one CS class at Yale, using the Zoo will require less setup time
- Essentially, the following few slides will walk through how to set up the zoo virtual environment and use tmux!

Zoo: Setup Commands

After logging into your zoo account, create/go to your dedicated ECG folder and set up a virtual environment:

```
python -m venv venv
source venv/bin/activate
pip install selenium bs4 lxml webdrivermanager
    pandas
```

This virtual environment is necessary so you can install the dependencies. Webdrivermanager allows you to use drivers (chromedriver, geckodriver) without needing to wget them!

Zoo: Tmux Commands

Tmux can be used so you can run multiple jobs within the Zoo.
The commands below detail how to use tmux; do so within the virtual environment. Once in tmux, run job normally then ctrl b, then d to exit.
This is the Yale guide:
https://docs.ycrc.yale.edu/clusters-at-yale/guides/tmux/ but the two commands below are effectively what you'll need!

```
tmux new -s selDemo
tmux attach -t selDemo
```

Using Selenium within a Zoo Virtual Env

Pro tip: you need to specify some initial options for your driver when within a virtual environment. Not all of these are absolutely necessary, but these are the one's I'd recommend:

Using Selenium within a Zoo Virtual Env

```
chromeOptions = webdriver.ChromeOptions()
chromeOptions.add\textunderscore argument("start-maximized")
chromeOptions.add\textunderscore
    argument("enable-automation")
chromeOptions.add\textunderscore argument("--headless")
chromeOptions.add\textunderscore argument("--no-sandbox")
chromeOptions.add\textunderscore
    argument("--disable-infobars")
chromeOptions.add\textunderscore
    argument("--disable-dev-shm-usage")
chromeOptions.add\textunderscore
    argument("--disable-browser-side-navigation")
chromeOptions.add\textunderscore argument("--disable-gpu")
driver = webdriver.Chrome(executable\textunderscore
    path=ChromeDriverManager().install(),
    options=chromeOptions)
driver.get("insert url")
```

Quick Start Guides

Quick Start Guides

Each of the following slides provides a quick set of commands for those who are familiar with Grace, but need a refresher on the exact commands to run.

Interactive Sessions

- Request an interactive session:

```
srun --pty -c 4 -p interactive -t 4:00:00 bash
```

- Request an interactive session with X11 (GUI) forwarding:

```
srun --pty -x11 -c 4 -p interactive -t 4:00:00 bash
```

Batch Scripts

- Run a batch script from the terminal:

  `sbatch your_batch_script.sh`

- Monitor job status:

  `squeue -u abc123`

Batch Scripts

Listing 16: stata_job.sh

```
#!/bin/bash
#SBATCH --job-name=stata_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# Load Stata module and run master .do file
module load Stata/15
stata-mp do  stata/0_master.do
```

Stata: Interactive

`srun --pty -x11 -c 4 -p interactive -t 4:00:00 bash`, then
`module load Stata/15`, followed by
`stata-mp` (command-line) or
`xtata-mp` (GUI)

Stata: Batch

Listing 17: stata_job.sh

```
#!/bin/bash
#SBATCH --job-name=stata_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# Load Stata module and run master .do file
module load Stata/15
stata-mp do  stata/0_master.do
```

Python: Interactive

`srun --pty -x11 -c 4 -p interactive -t 4:00:00 bash`, then
`module load miniconda`, followed by
`conda create -n py3_base` (if not already done), then `conda activate py3_base`, then
`ipython`

Python: Batch

Listing 18: python_job.sh

```
#!/bin/bash
#SBATCH --job-name=python_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# load the miniconda module
module load miniconda

# create the environment with the required libraries
#conda create -n py3_base python=3 numpy scipy matplotlib pandas ipython jupyter jupyterlab

# activate the environment and run master py script
conda activate py3_base
python python/0_master.py
```

R: Interactive

`srun --pty -x11 -c 4 -p interactive -t 4:00:00 bash`, then
`module load miniconda`, followed by
`conda create -n r3_base` (if not already done), then `conda activate r3_base`, then
`R`

R: Batch

Listing 19: r_job.sh

```
#!/bin/bash
#SBATCH --job-name=r_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# load the miniconda module
module load miniconda

# create the environment with the required packages
#conda create --name r3_base r-base r-essentials r-doMC r-Rmpi

# activate the environment and run master R script
conda activate r3_base
R --slave -f r/0_master.R
```

Matlab: Interactive

`srun --pty -x11 -c 4 -p interactive -t 4:00:00 bash`, then

`module load MATLAB/2019a-parallel`, then

`matlab`

Matlab: Batch

Listing 20: matlab_job.sh

```
#!/bin/bash
#SBATCH --job-name=matlab_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual

# Load matlab module and run master .m file
module load MATLAB/2019a-parallel
matlab -nodisplay < matlab/master.m
```

LaTeX: Interactive

`srun --pty -x11 -c 4 -p interactive -t 4:00:00 bash`, then

`module load module load TeXLive/2019`, then

`pdflatex sample2e.tex`

LaTeX: Batch

Listing 21: latex_job.sh

```
#!/bin/bash
#SBATCH --job-name=latex_job
#SBATCH --ntasks=1 --nodes=1 --cpus-per-task=1
#SBATCH --mem-per-cpu=5G
#SBATCH --partition day
#SBATCH --time=0:05:00
#SBATCH --mail-type=ALL
#SBATCH --output=%x_%j.out

cd /home/adw54/Documents/egc_hpc_manual/latex

# Load Stata module and run master .do file
module load TeXLive/2019
pdflatex sample2e.tex
```