More Docker, Git(hub), and the Unix Shell

MAS Departmental Disclaimers:

For students trying to take or audit from outside the MAS program.

Taking or auditing 400 courses is simply is not permitted because this is a self-supporting program. Sorry, unfortunately, you will NOT be able to take any of the 400 level Stats courses.

There are NO exceptions that can be made by the department. These classes were designed specifically for students who applied directly to the program.

The students of this program are also not allowed to audit or enroll in classes outside of the program as it was created for working professionals.

If you would like to apply for the program, you are welcome to do so: https://master.stat.ucla.edu/admissions/

Information is found here: https://master.stat.ucla.edu/ And here: https://master.stat.ucla.edu/faq/

A reminder…

class communications

Slack Channel
uclastat418-class

last time we saw an introduction to docker

today we'll make use of it while also hearing a bit more about what it is.

throughout the rest of course (in most weeks) we will continue to make use of docker environments to incrementally learn about its features

Docker: what is it?

a software to build and run containers with a container being a single instance of an image

an image is a unit of software that allows one to package up code and all its dependencies

Docker: why is it important?

different software development stacks require different environments; especially dependent on task

classical solutions entail virtual environments (anaconda, python virtualenv) or virtual machines (vagrant, vmware)

Docker: so why do data science with it?

it is platform agnostic; as long as a machine has docker we can use the same environment (perfect in a classroom setting where we all have different machines; perfect in a work setting where we might have different machines and certainly different versions)

can share version-controlled environments akin to GitHub for docker images

containers are ephemeral; if you mess up you can start over easily

build on top of other peoples work

ready-made workflow for deployment through Kubernetes or similiar

Docker: running Rstudio

run the following

```
docker run -e PASSWORD='class' --rm -p 8787:8787 rocker/rstudio
```

and then visit in any browser

`http://localhost:8787` or `http://<your-ip-address>:8787`

finally login with the following credentials

Username: `rstudio` Password: `<what you used above>`

Docker: running Rstudio

to exit

Copy (CTRL-C)

We will see another way to exit in a moment

Docker: running Rstudio with mounting local volume (connect to your computer)

cd to desired directory, then use for Mac

```
docker run -d --rm -e PASSWORD=class -p 8787:8787 -v /Users/nlangholz/Documents/:/home/rstudio/Documents rocker/rstudio
```

chdir to desired directory in Windows command line

```
docker run -d --rm -e PASSWORD=class -p 8787:8787 -v %cd%:/home/rstudio/Documents rocker/rstudio
```

cd to desired directory in Windows powershell

```
docker run -d --rm -e PASSWORD=class -p 8787:8787 -v ${PWD}:/home/rstudio/Documents rocker/rstudio
```

Again visit in your browser port 8787 either at local host or <your-ip-address> and login with your credentials

Docker: a few other commands

Now to exit we will need to take a look at what containers are running to find the container id or name

```
docker container ls
```

```
docker kill <container>
```

And finally, to see what docker images we have available

```
docker images
```

Docker: a few other commands

And finally, to see what docker images exist on our machine

```
docker images
```

If you'd like to remove any of these images

```
docker rmi <image id>
```

Git

Git: what is it?

a version control system to track the history of changes as people and teams collaborate on code projects together

Git manages the evolution of a set of files (or in our case a data analysis or model development) in a repository

this provides a transparent history of changes, who made them and how each contribute all while reducing redundancy and unnecessary work

Git

|< < Prev Random Next > >|

|< < Prev Random Next > >|

Permanent link to this comic: https://xkcd.com/1597/
Image URL (for hotlinking/embedding): https://imgs.xkcd.com/comics/git.png

Repository management services

Repository management services

GitHub

Code Collaboration & Version Control

Follow

Stacks
32.5K

I Use This

Fans	Jobs	Votes
28.7K	3.75K	10K

Bitbucket

Code Collaboration & Version Control

Follow

Stacks
9.82K

I Use This

Fans	Jobs	Votes
8.44K	438	2.77K

GitLab

Code Collaboration & Version Control

Follow

Stacks
8.35K

I Use This

Fans	Jobs	Votes
7.57K	357	2K

Repository management services

What companies use GitHub?

5837 companies on StackShare use GitHub

- Airbnb
- Netflix
- Medium
- Instacart
- reddit
- Lyft
- StackShare
- Shopify
- 9GAG
- Asana

What companies use Bitbucket?

2270 companies on StackShare use Bitbucket

- PayPal
- Salesforce
- Zillow
- Starbucks
- CircleCI
- Tesla Motors
- Bitbucket
- Pandora
- Sellsuki
- Movielala

What companies use GitLab?

1584 companies on StackShare use GitLab

- Alibaba.com
- Coderus
- Webedia
- Ticketmaster
- WebbyLab
- Edify
- Electronic Arts
- Freelancer.com
- Citrix
- WILD

We’re going to use GitHub

http://github.com

Make an account if you don’t have one

Let it be known…

Microsoft | Stories Our Company Our Products Blogs & Communities Press Tools

Microsoft to acquire GitHub for $7.5 billion

June 4, 2018 | Microsoft News Center

Acquisition will empower developers, accelerate GitHub's growth and advance Microsoft services with new audiences

https://www.theverge.com/2018/6/18/17474284/microsoft-github-acquisition-developer-reaction

It’s not all fun and games. Actually it’s quite hard. Initially just working on solo repositories becomes fine but once collaborating you’ll start to run into merge conflicts (yikes)

Keep going!!

GitHub: what tool to use?

Use what makes you feel comfortable and efficient.

> *"I certainly know Git very well, and honestly think I'm far faster and more efficient in a Git GUI than I could possibly be in the command line – and I'm certainly not slow in the CLI."*
>
> —Dan Clarke, blogger and co-organizer of .Net Oxford group

> *"I sometimes encounter people who feel it's "better" to use command line Git, but for very ill-defined reasons. These people may feel like they should work in the shell, even if it leads to Git-avoidance, frequent mistakes, or limiting themselves to a small set of ~3 Git commands. This is counterproductive."*
>
> —Jenny Bryan, Software Engineer at RStudio & Adjunct Professor at the University of British Columbia

Both quotes take from https://blog.axosoft.com/git-gui-vs-cli/

GitHub GUIs

There are a wide variety of choices for GUIs if you are interested.

GitHub: some (basic) guidelines

Personally, I start all my repositories remotely (on GitHub.com). I find it much easier to then pull to my local machine and to then make my first commit

This includes forking others' repos and then cloning to local

Create new branches to make edits (add new features) to the master code base

Commit early and often; typically don't push data (especially when large) and definitely not credentials

(these are all quite simple; you'll surely run into many issues and there are many more resources online that will help you, but don't be afraid to try some new things in this course to learn how it works)

GitHub: our class repository

Let’s create a repo for you to use and push hw to my master repo. cd to the directory where you want to

Fork the class repo on GitHub. Clone your https version of the repo.

Now let’s go to docker and do this in a (hopefully) more familiar environment…get an Rstudio container running with a local volume mounted. (if its not still running at the moment)

…will walk through as a class.

A bit more on reproducibility…

there is a bit of crisis taking place in regards to reproducibility as a whole in science fields. and this true of the data science field as well.

We've seen some of the tools to enable us to do this reproducible work, but what concepts should we be thinking of as we use them?

Repeatability

or test–retest reliability is the variation in measurements taken by a single person or instrument on the same item, under the same conditions, and in a short period of time. A less-than- perfect test–retest reliability causes test–retest variability.

Your systems must be repeatable and reliable, the same query on the same data should return the same results. Otherwise, none of this applies.

Low repeatability, Low accuracy

High repeatability, Low accuracy

High Repeatability, High accuracy

Reproducibility

an analysis is *reproducible* if there is a specific set of computational functions/ analyses (usually specified in terms of code) that exactly reproduce all of the numbers in a published paper from raw data. It is now recognized that a critical component of the scientific process is that data analyses can be reproduced.

Replicability

but just because a study is reproducible does not mean that it is *replicable*. Replicability is stronger than *reproducibility*. A study is only *replicable* if you perform the exact same experiment (at least) twice, collect data in the same way both times, perform the same data analysis, and arrive at the same conclusions.

Stodden's Taxonomy of Reproducibility

Computational

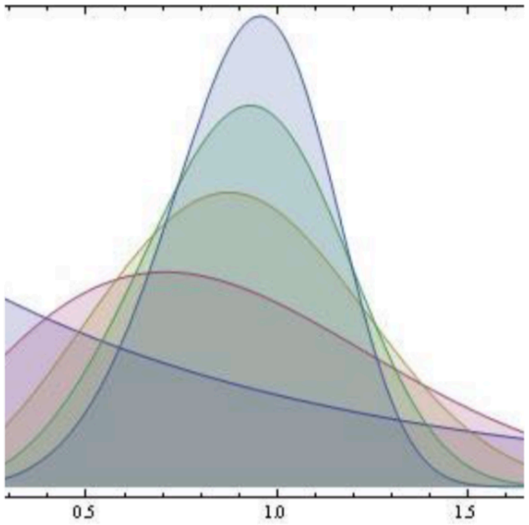
when detailed information is provided about code, software, hardware and implementation details.

Empirical

when detailed information is provided about code, software, hardware and implementation details.

Statistical

when detailed information is provided about code, software, hardware and implementation details.



Computational reproducibility should be…

Reviewable

The descriptions of the research methods can be independently assessed and the results judged credible

Replicable

tools are made available that would allow one to duplicate the results of the research, for example by running the authors' code to produce the plots shown in the publication.

Auditable

sufficient records (including data and software) have been archived so that the research can be defended later if necessary or differences between independent confirmations resolved. The archive might be private, as with traditional laboratory notebooks.

Confirmable

the main conclusions of the research, or the model in production can be attained independently without the use of software provided by the author.

Simple rules for reproducible computational research

track results - whenever a result may be of potential interest, keep track of how it was produced. as a minimum, you should at least record sufficient details on programs, parameters, and manual procedures to allow yourself, in a year or so, to approximately reproduce the results.

script everything - whenever possible, rely on the execution of programs instead of manual procedures to modify data. If manual operations cannot be avoided, you should as a minimum note down which data files were modified or moved, and for what purpose.

store program versions - In order to exactly reproduce a given result, it may be necessary to use programs in the exact versions used originally. As a minimum, you should note the exact names and versions of the main programs you use.

Simple rules for reproducible computational research

use version control - even the slightest change to a computer program can have large intended or unintended consequences. As a minimum, you should archive copies of your scripts from time to time, so that you keep a rough record of the various states the code has taken during development

store data & intermediate results - in principle, as long as the full process used to produce a given result is tracked, all intermediate data can also be regenerated. In practice, having easily accessible intermediate results may be of great value. As a minimum, archive any intermediate result files that are produced when running an analysis

set a random number seed- many analyses and predictions include some element of randomness, meaning the same program will typically give slightly different results every time it is executed. As a minimum, note which analysis steps involve randomness, so that a level of discrepancy can be anticipated when reproducing the results.

Simple rules for reproducible computational research

store data viz inputs- from the time a figure is first generated to it being part of an analysis. It is critical to store the data and process that generated it. As a minimum, one should note which data formed the basis of a given plot and how this data could be reconstructed.

allow levels of analysis - in order to validate and fully understand the main result, it is oLen useful to inspect the detailed values underlying the summaries. Make those fluid and explorable, as a minimum at least once generate, inspect, and validate the detailed values underlying the summaries.

generate text programmatically- there is nothing quite as embarrassing as having a disagreement between your analytical narrative in a writeup and having the tables and figures disagree. Connect them so that there is no change of disagreement.

.

Unix Shell

The shell: why is it important??

to introduce the shell means to have a discussion about the structure of the computer, operating systems, file systems and history

the shell offers programmatic access to a computer's underlying parts, providing the ability to "do" data analysis on directories, on processes, and on their networks

as there are many flavors of the shell, it is the first time we come to a decision about choosing 'tools', about evaluating which shell is best for you and the shifting terrain of software development

The shell: why is it important?

as a practical matter, shell tools are an indispensable part for my own data science practice, data ‘cleaning’, data processing, exploratory analysis, automation; by design they let us deal with data on a scale that can be difficult from with R or Python

it also provides a low level link to other data platforms, data sources, and remote environments

The shell: how we'll learn

everyone needs access to the shell. everyone with a Mac and Linux already have one. those with Windows machines need something else..

there are instructions on the site for Windows users to get a version of it, but..

as we all have docker installed we can use an image that provides us all with the same environment for learning and exploration. not all functionality works in the image but overall will be useful for us (also we've seem a version in our docker Rstudio environment)

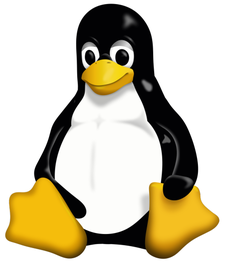
Operating Systems

most devices that contain a computer of some kind will have an OS, they tend to emerge when the appliance will have to deal with new applications, complex user-input and possibly changing requirements on its function

Operating Systems

An operating system is a piece of software that organizes and controls hardware and other software so your computer behaves in a flexible but predictable way

maybe obviously to us all, Window, MacOS and Linux are those most commonly used for home computers. iOS and Android are most common on mobile devices



A history

In 1964, Bell Labs partnered with MIT and GE to create Multics (Multiplexed Information and Computing Service)

“Such systems must run continuously and reliably 7 days a week, 24 hours a day in a way similar to telephone or power systems, and must be capable of meeting wide service demand from multiple man-machine interaction to the sequential processing of absentee-user jobs, from the use of the system with dedicated languages and subsystems to the programming of the stem itself”

A history

Bell Labs pulled out of the Multics project in 1969, a group of researchers at Bell Labs started work on Unica (uniplexed information and computing system) because initially it could only support one user; as the system matured it was renamed Unix, which isn't an acronym for anything

Ritchie simply says that Unix is a 'somewhat treacherous pun on Multics'

The Unix filesystem

Multics brought about the first notion of a hierarchical file system; files were arranged in a tree structure allowing users to have control of they on areas

Unix began (more or less) as a file system and then an interactive shell emerged to let you examine its contest and perform basic operations

The kernel and the shell

the Unix kernel is the part of the operating system that carries out basic functions like accessing files, handing communication, and others

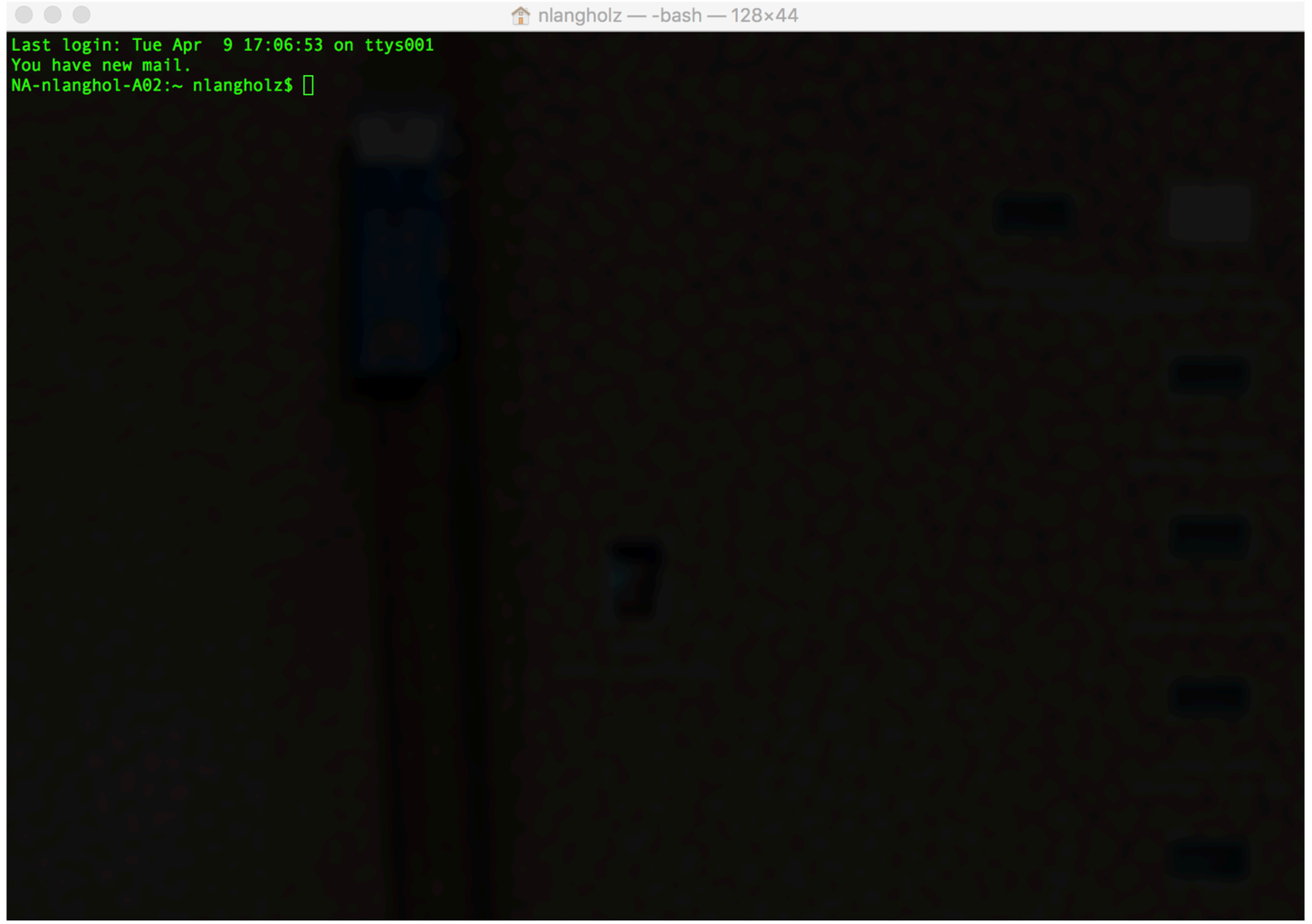
the Unix shell is a user interface to the kernel (keep in mind that Unix was designed for computer scientists and the interface is not optimized for novices)

Unix shells

A shell is a type of program called an interpreter, think of it as a text-based interface to the kernel

It operates in simple loop: it accepts a command, interprets it, executes the command and waits for another

The shell displays a prompt to tell you that it is ready to accept a command



Not much going on but makes us feel like we're in the matrix

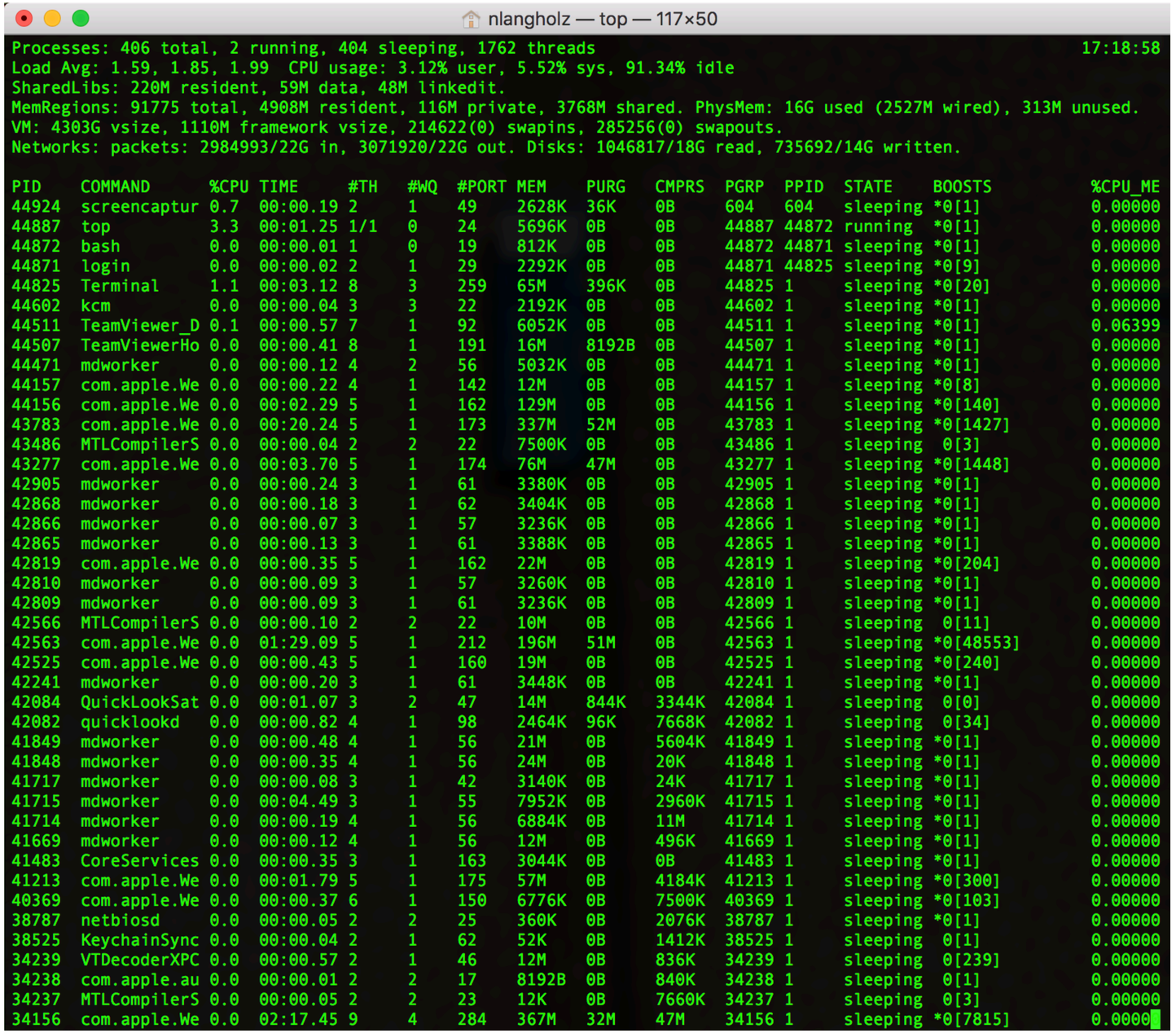
Unix shells

The shell is itself a program the the Unix operating system runs for you (a program is referred to as a process when its running)

The kernel manages many processes at once, many of which are the result of user commands (others provide services that keep the computer running)

Some commands are built into the shell, others have been added by users

Either way, the shell waits until the command is executed



Result of the command **`top`**; this is a printout of all the processes running on your computer

Operating Systems

Process Management

Schedules jobs(formally referred to as processes) to be executed by the computer

Memory and storage management

Allocate space required for each running process in main memory (RAM) or in some other temporary location if space is tight and supervise the storage of data onto disk

Operating Systems

Device Management

A program called a driver translates data (files from the filesystem) into signals that

Application Interface

An API (application programming interface) let programmers use functions of the computer and the operating system without having to know *how* something is done

User Interface

Finally, the operating system turns and looks at you; the UI is a program that defines how users interact with the computer; some are graphical (Windows is a GUI) and some are text-based (your Unix shell)

Unix Shell(s)

There are, in fact, many different kinds of Unix shells

The table on the right lists a few of the most popular

KSH(1) General Commands Manual KSH(1)

NAME
ksh, ksh93 - KornShell, a command and programming language

SYNOPSIS
ksh [±abcefhikmnoprstuvxBCDP] [-R file] [±o option] ... [-] [arg ...]
rksh [±abcefhikmnoprstuvxBCD] [-R file] [±o option] ... [-] [arg ...]

DESCRIPTION
Ksh is a command and programming language that executes commands read from a terminal or a file. Rksh is a restricted version of the command interpreter ksh; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. Rpfksh is a profile shell version of the command interpreter ksh; it is used to to execute commands with the attributes specified by the user's profiles (see pfexec(1)). See Invocation below for the meaning of arguments to the shell.

TCSH(1) General Commands Manual TCSH(1)

NAME
tcsh - C shell with file name completion and command line editing

SYNOPSIS
tcsh [-bcdefFimnqstvVxX] [-Dname[=value]] [arg ...]
tcsh -l

DESCRIPTION
tcsh is an enhanced but completely compatible version of the Berkeley UNIX C shell, csh(1). It is a command language interpreter usable both as an interactive login shell and a shell script command processor. It includes a command-line editor (see The command-line editor), programmable word completion (see Completion and listing), spelling correction (see Spelling correction), a history mechanism (see History substitution), job control (see Jobs) and a C-like syntax. The NEW FEATURES section describes major enhancements of tcsh over csh(1). Throughout this manual, features of tcsh not found in most csh(1) implementations (specifically, the 4.4BSD csh) are labeled with `(+)', and features which are present in csh(1) but not usually documented are labeled with `(u)'.

BASH(1) General Commands Manual BASH(1)

NAME
bash - GNU Bourne-Again SHell

SYNOPSIS
bash [options] [command_string | file]

COPYRIGHT
Bash is Copyright (C) 1989-2013 by the Free Software Foundation, Inc.

DESCRIPTION
Bash is an sh-compatible command language interpreter that executes commands read from the standard input or from a file. Bash also incorporates useful features from the Korn and C shells (ksh and csh).

Bash is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). Bash can be configured to be POSIX-conformant by default.

ZSH(1) General Commands Manual ZSH(1)

NAME
zsh - the Z shell

OVERVIEW
Because zsh contains many features, the zsh manual has been split into a number of sections:

zsh	Zsh overview (this section)
zshroadmap	Informal introduction to the manual
zshmisc	Anything not fitting into the other sections
zshexpn	Zsh command and parameter expansion
zshparam	Zsh parameters
zshoptions	Zsh options
zshbuiltins	Zsh built-in functions
zshzle	Zsh command line editing
zshcompwid	Zsh completion widgets
zshcompsys	Zsh completion system
zshcompctl	Zsh completion control
zshmodules	Zsh loadable modules
zshcalsys	Zsh built-in calendar functions

fish(1) fish fish(1)

NAME
fish - fish - the friendly interactive shell

fish - the friendly interactive shell
Synopsis
fish [-h] [-v] [-c command] [FILE [ARGUMENTS...]]

Description
fish is a command-line shell written mainly with interactive use in mind. The full manual is available in HTML by using the help command from inside fish.

The following options are available:

- -c or --command=COMMANDS evaluate the specified commands instead of reading from the commandline

- -d or --debug-level=DEBUG_LEVEL specify the verbosity level of fish. A higher number means higher verbosity. The default level is 1.

Why the choices?

A shell program was originally meant to take commands, interpret them and then execute some operation

Inevitably, one wants to collect a number of these operations into programs that execute compound tasks at the the same time you want to make interaction on the command line as easy as possible (a history mechanism, editing capabilities and so on)

The original Bourne shell is ideal for programming; the C-shell and its variants are good for interactive use; the Korn shell is a combination of both

Steve Bourne, created of sh, 2005 (source wikipedia)

And while were at it

unix itself comes in different flavors; the 1980s saw an incredible proliferation of Unix versions, somewhere around 100 (System V, AIX, Berkely BSD, SunOS, Linux, …)

vendors provided (diverging) version of Unix, optimized for their own computer architectures and supporting different features

despite the diversity it was still easier to “port” applications between versions of Unix than it was between different proprietary OS

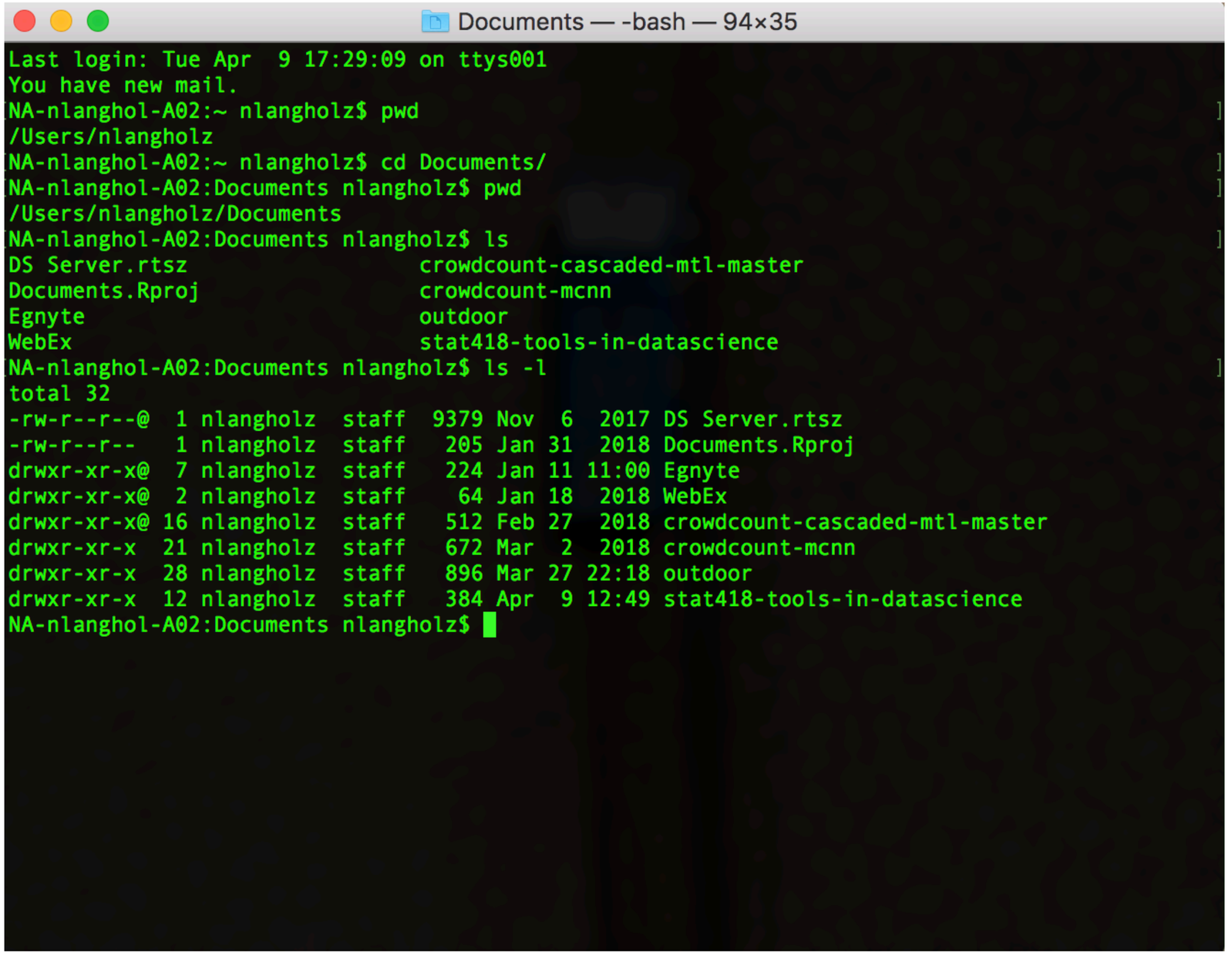
A few common commands

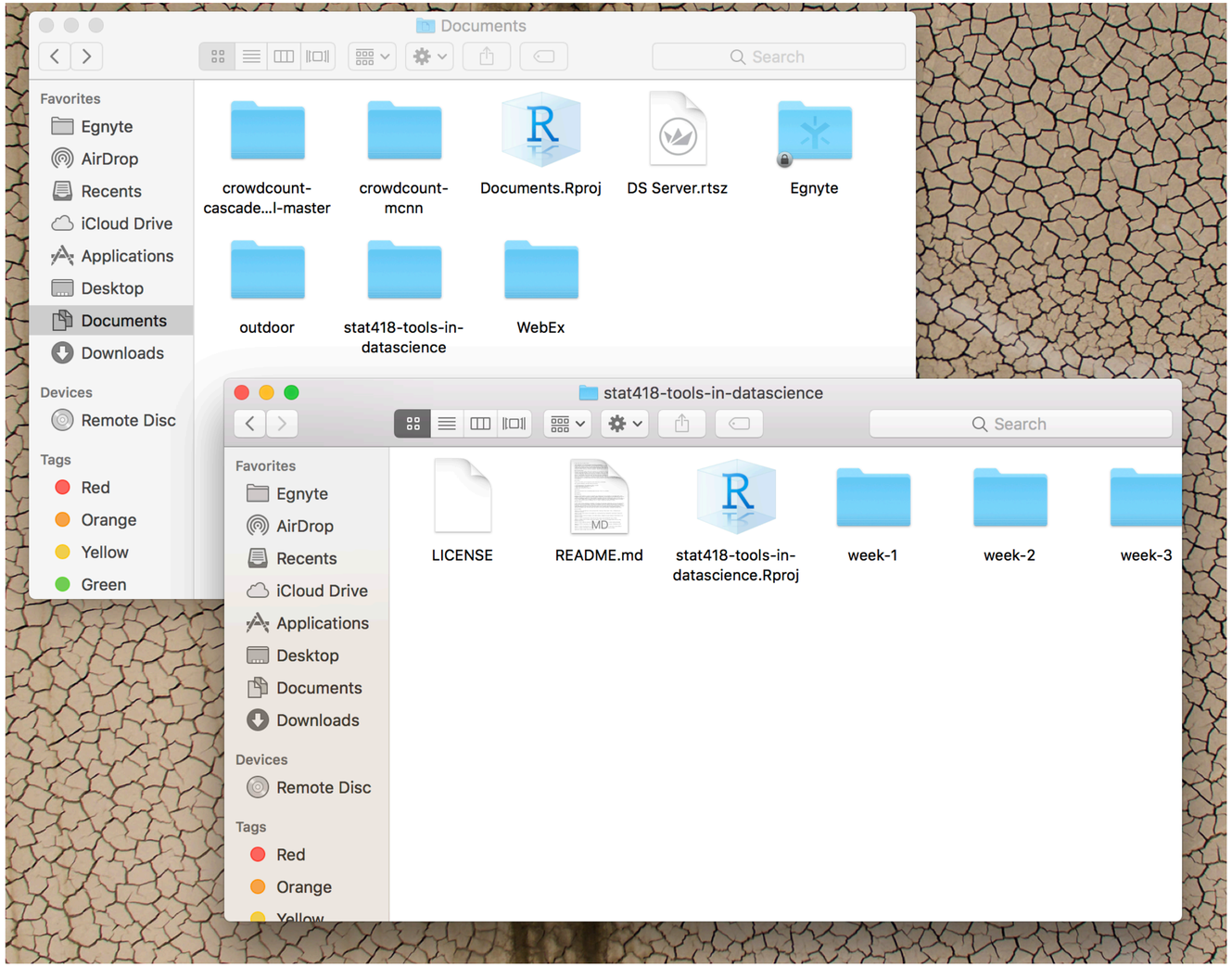
FIrst, commands to explore your file system, walk through directories and list flies

`pwd, ls, cd`

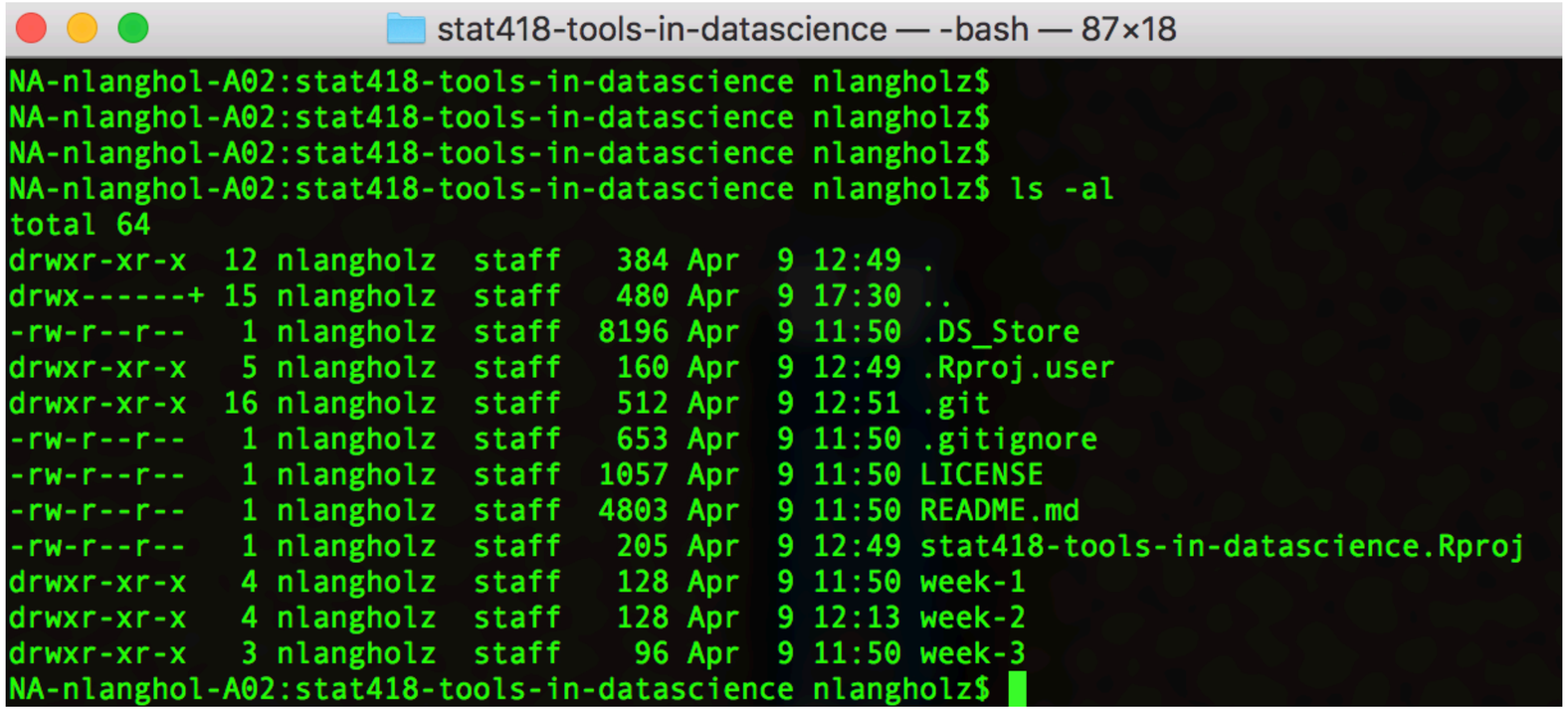
`mkdir, rmdir`

`cp, mv, rm`


Documents — -bash — 94×35
Last login: Tue Apr 9 17:29:09 on ttys001
You have new mail.
NA-nlanghol-A02:~ nlangholz$ pwd
/Users/nlangholz
NA-nlanghol-A02:~ nlangholz$ cd Documents/
NA-nlanghol-A02:Documents nlangholz$ pwd
/Users/nlangholz/Documents
NA-nlanghol-A02:Documents nlangholz$ ls
DS Server.rtsz
Documents.Rproj
Egnyte
WebEx
crowdcount-cascaded-mtl-master
crowdcount-mcnn
outdoor
stat418-tools-in-datascience
NA-nlanghol-A02:Documents nlangholz$ ls -l
total 32
-rw-r--r--@ 1 nlangholz staff 9379 Nov 6 2017 DS Server.rtsz
-rw-r--r--   1 nlangholz staff   205 Jan 31 2018 Documents.Rproj
drwxr-xr-x@ 7 nlangholz staff   224 Jan 11 11:00 Egnyte
drwxr-xr-x@ 2 nlangholz staff   64 Jan 18 2018 WebEx
drwxr-xr-x@ 16 nlangholz staff   512 Feb 27 2018 crowdcount-cascaded-mtl-master
drwxr-xr-x 21 nlangholz staff   672 Mar 2 2018 crowdcount-mcnn
drwxr-xr-x 28 nlangholz staff   896 Mar 27 22:18 outdoor
drwxr-xr-x 12 nlangholz staff   384 Apr 9 12:49 stat418-tools-in-datascience
NA-nlanghol-A02:Documents nlangholz$



Another view of the filesystem; here your Mac will display directories as folders and of course you navigate by clicking rather than typing commands


stat418-tools-in-datascience — -bash — 87×18
NA-nlanghol-A02:stat418-tools-in-datascience nlangholz$
NA-nlanghol-A02:stat418-tools-in-datascience nlangholz$
NA-nlanghol-A02:stat418-tools-in-datascience nlangholz$
NA-nlanghol-A02:stat418-tools-in-datascience nlangholz$ ls -al
total 64
drwxr-xr-x 12 nlangholz staff   384 Apr 9 12:49 .
drwx------+ 15 nlangholz staff   480 Apr 9 17:30 ..
-rw-r--r--   1 nlangholz staff 8196 Apr 9 11:50 .DS_Store
drwxr-xr-x   5 nlangholz staff   160 Apr 9 12:49 .Rproj.user
drwxr-xr-x 16 nlangholz staff   512 Apr 9 12:51 .git
-rw-r--r--   1 nlangholz staff   653 Apr 9 11:50 .gitignore
-rw-r--r--   1 nlangholz staff 1057 Apr 9 11:50 LICENSE
-rw-r--r--   1 nlangholz staff 4803 Apr 9 11:50 README.md
-rw-r--r--   1 nlangholz staff   205 Apr 9 12:49 stat418-tools-in-datascience.Rproj
drwxr-xr-x   4 nlangholz staff   128 Apr 9 11:50 week-1
drwxr-xr-x   4 nlangholz staff   128 Apr 9 12:13 week-2
drwxr-xr-x   3 nlangholz staff   96 Apr 9 11:50 week-3
NA-nlanghol-A02:stat418-tools-in-datascience nlangholz$

Kinds of Files

What you'll notice right away is that there are different types fo files having different permissions

Unix filesystem conventions places (charaed, commonly used) executable files in places like /usr/bin or /usr/local/bin

Different files are opened by different kinds of programs, in OSX, there is a beautiful command called open that decides which program to use

Although we saw a terminal environment in our Rstudio docker container, now let's get another docker image that has some nice data science specific code.

we will use the docker image from the book Data Science at the Command Line by Jeroen Janssens

https://www.datascienceatthecommandline.com/index.html

First, we need to pull the image using the following

```
docker pull datascienceworkshops/data-science-at-the-command-line
```

Data Science at the Command Line

To run the image

```
docker run --rm -it datascienceworkshops/data-science-at-the-command-line
```

To leave the environment simply type **`exit`**

again we didn't mount a local volume so we have no connection to our computer

Now to mount our local directories, cd to desired directory, then use

on a Mac

```
docker run --rm -it -v`pwd`:/data datascienceworkshops/data-science-at-the-command-line
```

in Windows command line

```
docker run --rm -it -v %cd%:/data datascienceworkshops/data-science-at-the-command-line
```

in Windows powershell

```
docker run --rm -it -v ${PWD}:/data datascienceworkshops/data-science-at-the-command-line
```