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/

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)

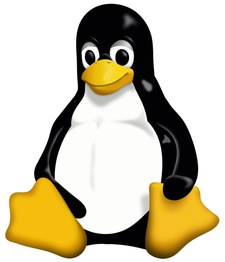
Operating Systems

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

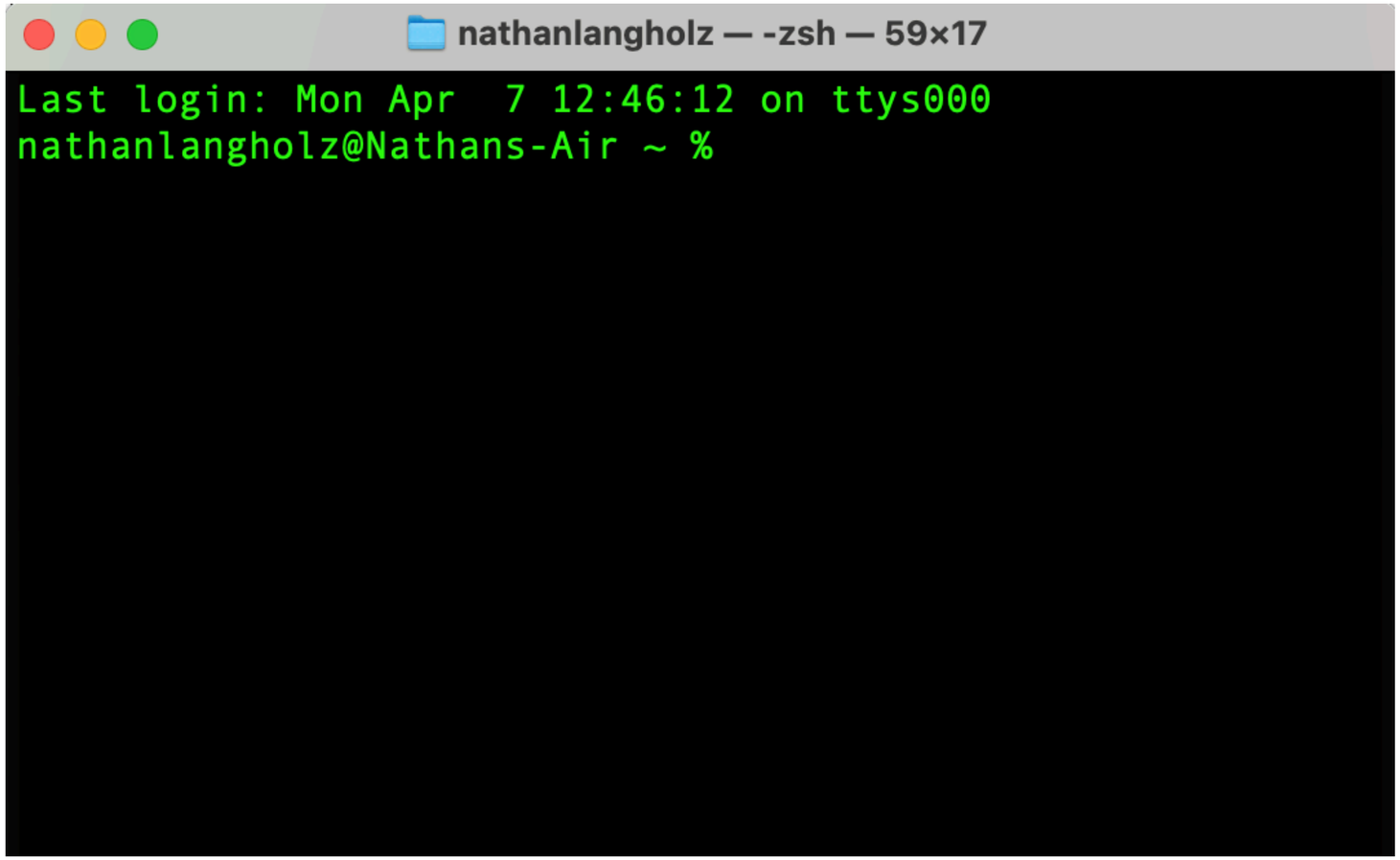
the Unix shell is a user interface to the kernel (keep in mind that Unix was designed for computer scientists and the interface is not optimized for novices)

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 lots of potential!

The % is the prompt

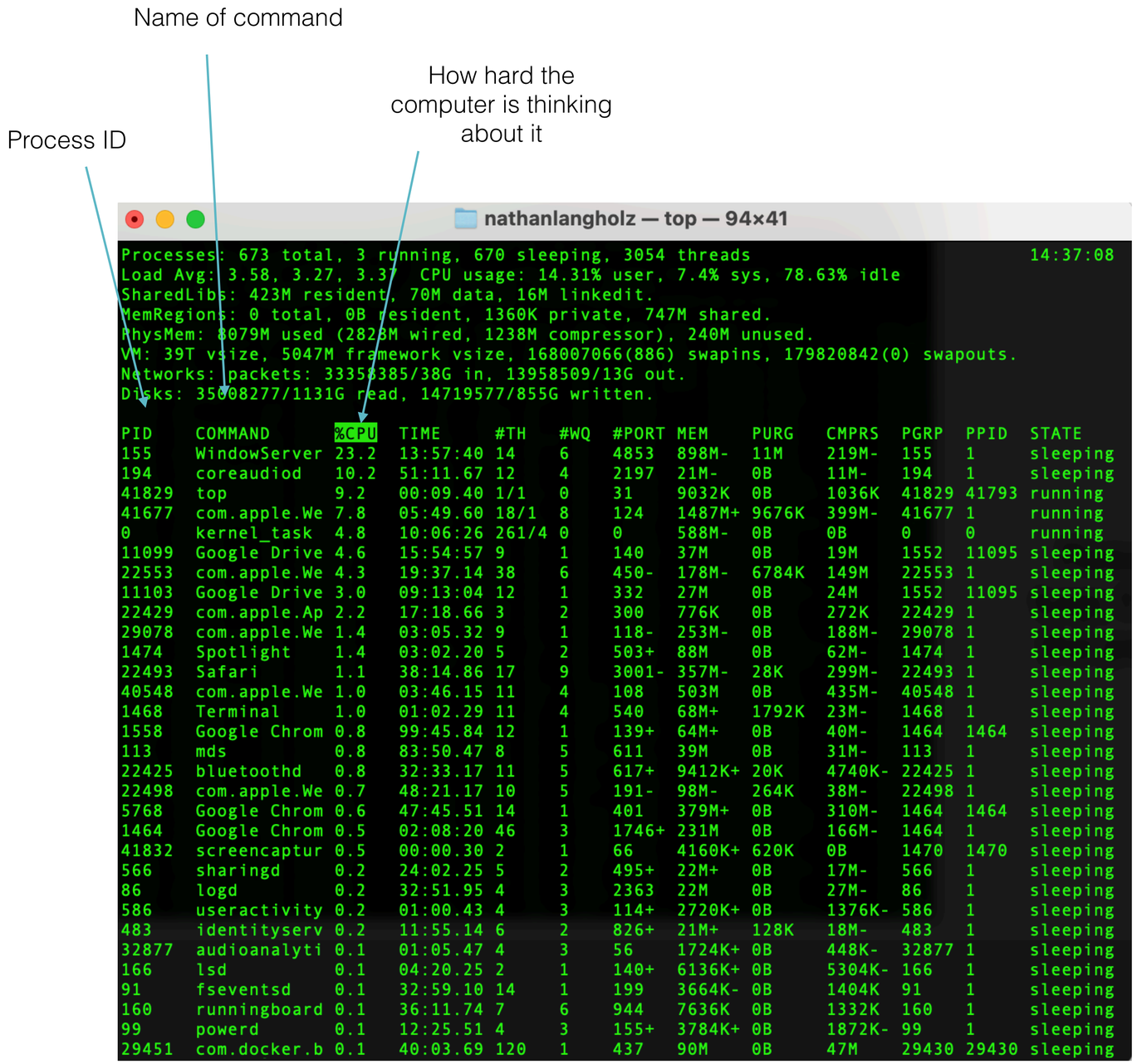
Unix shells

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 Programming 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

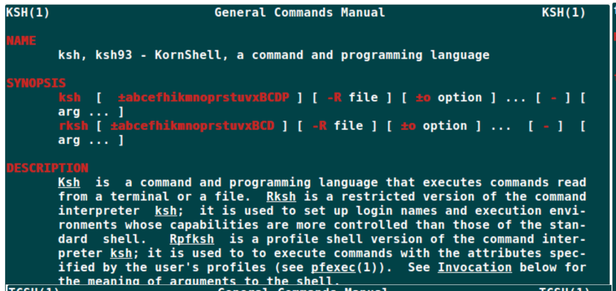
User Interface

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)

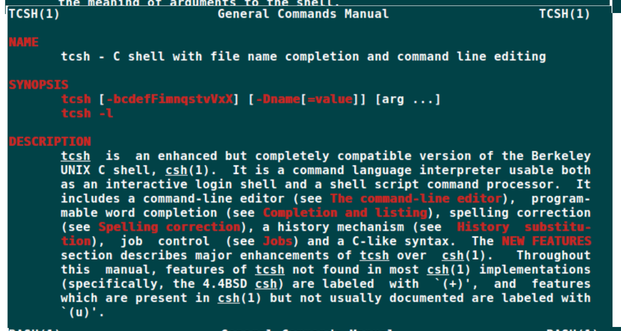
Unix Shell(s)

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

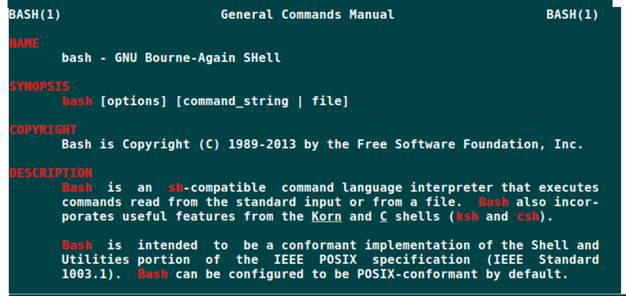
The table on the right lists a few of the most popular



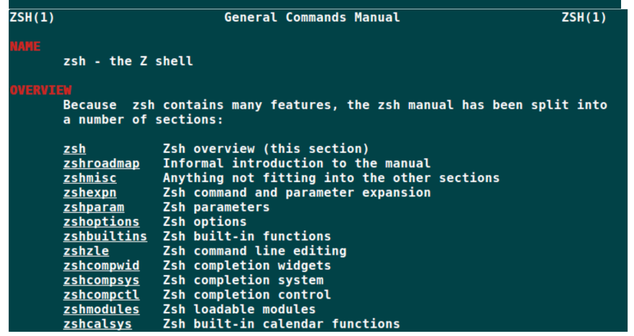
```
KSH(1)                    General Commands Manual                    KSH(1)

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 envi-
       ronments whose capabilities are more controlled than those of the stan-
       dard  shell.   Rpfksh  is a profile shell version of the command inter-
       preter ksh; it is used to to execute commands with the attributes spec-
       ified by the user's profiles (see pfexec(1)).  See Invocation below for
       the meaning of arguments to the shell.
```



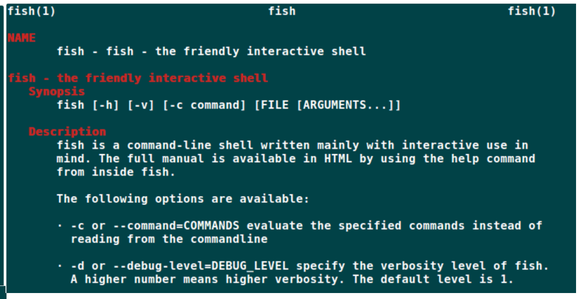
```
TCSH(1)                   General Commands Manual                    TCSH(1)

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),  program-
       mable word completion (see Completion and listing), spelling correction
       (see Spelling correction), a history mechanism (see  History  substitu-
       tion),  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 incor-
       porates 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

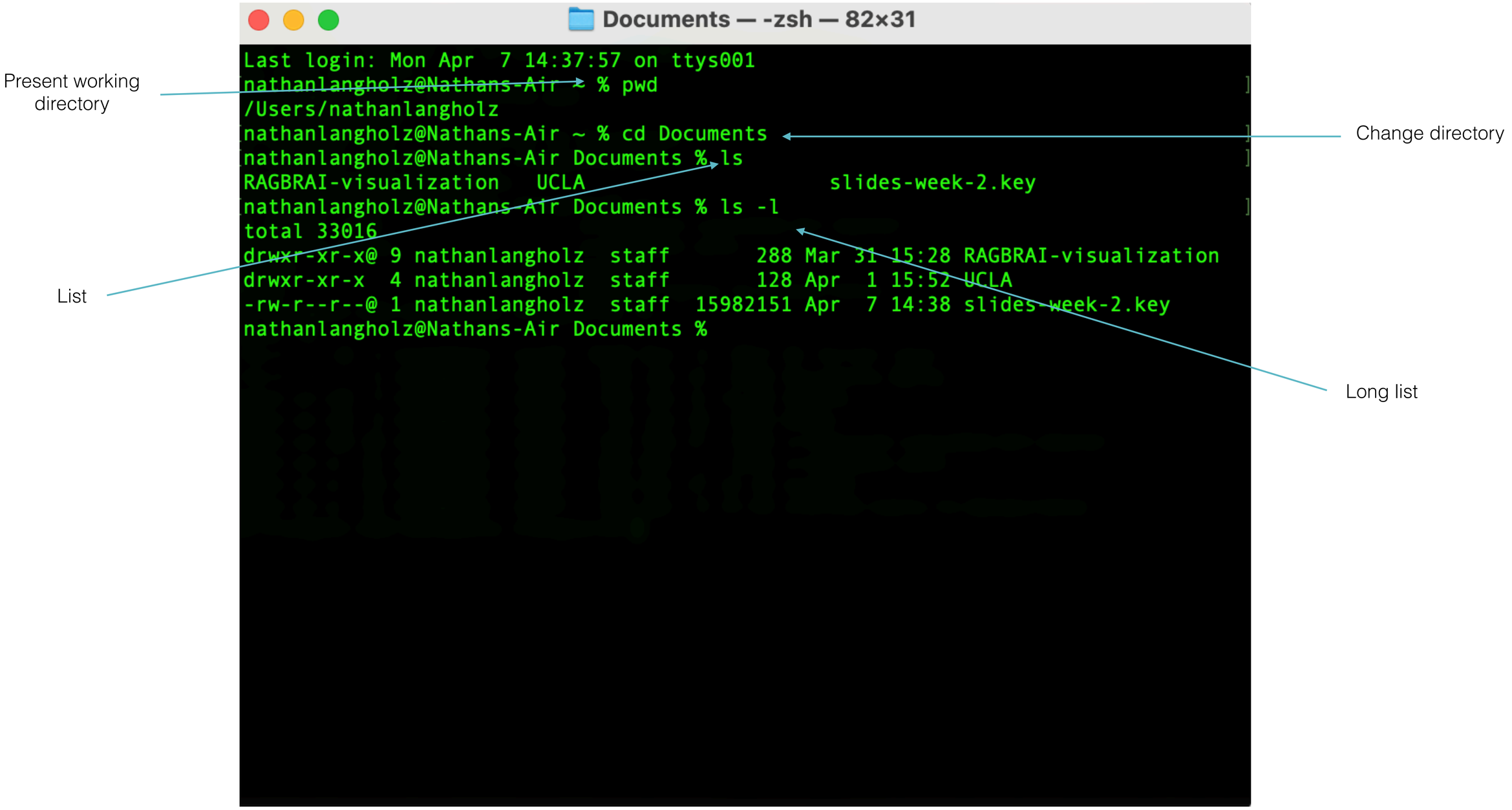
A few common commands

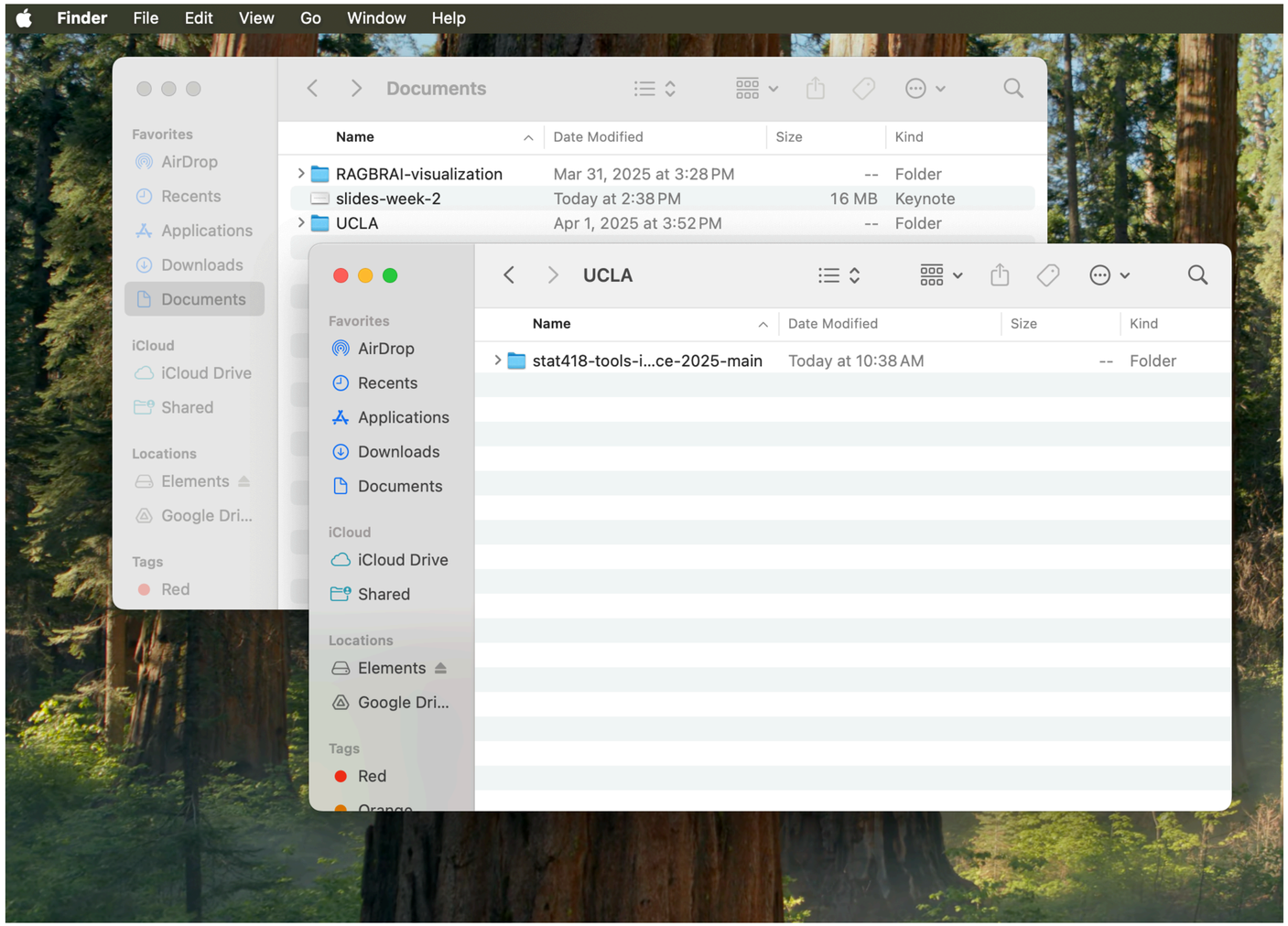
FIrst, commands to explore your file system, walk through directories and list flies

`pwd, ls, cd`

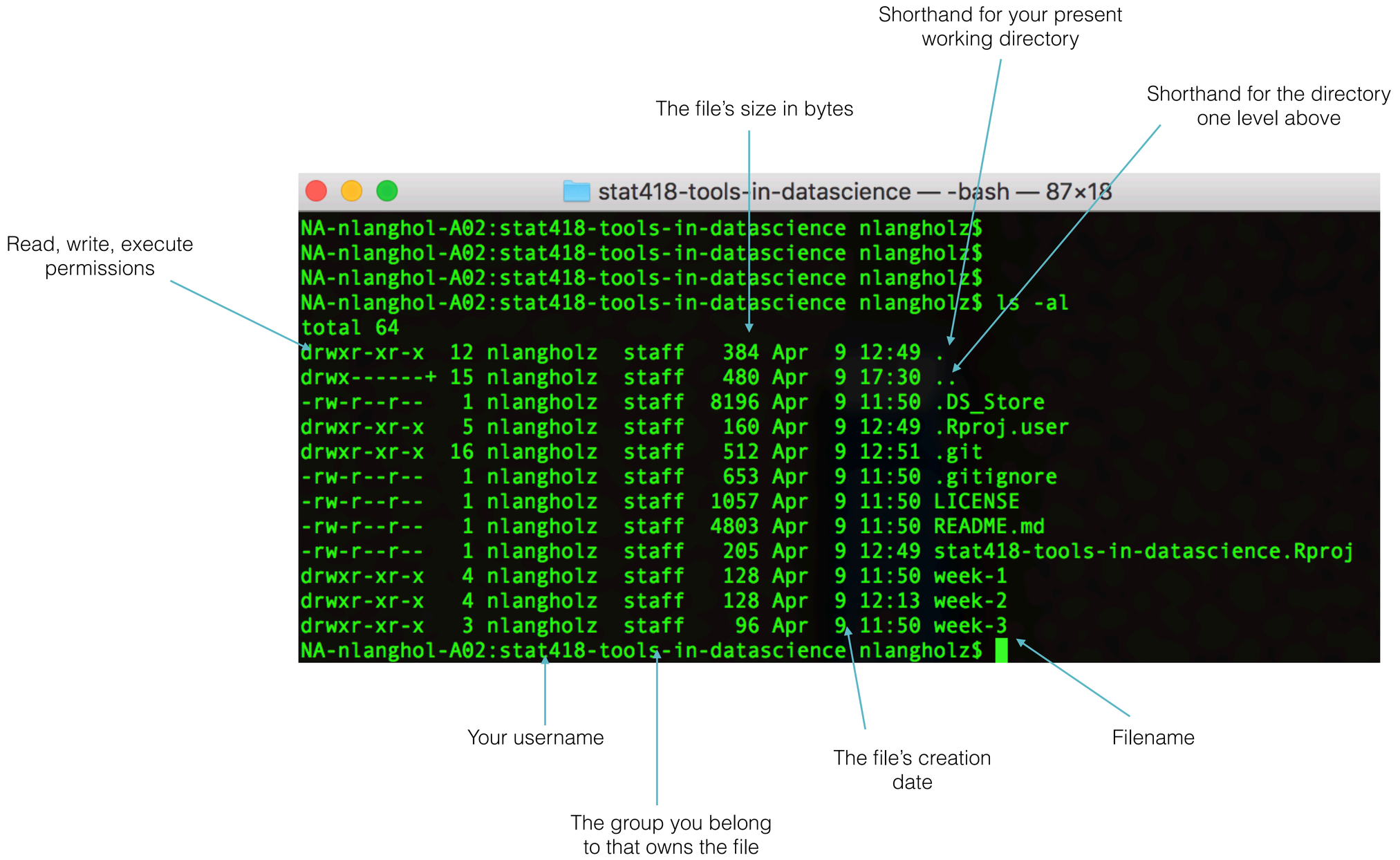
`mkdir, rmdir`

`cp, mv, rm`


Present working directory
List
Change directory
Long list
Documents — -zsh — 82×31
Last login: Mon Apr 7 14:37:57 on ttys001
nathanlangholz@Nathans-Air ~ % pwd
/Users/nathanlangholz
nathanlangholz@Nathans-Air ~ % cd Documents
nathanlangholz@Nathans-Air Documents % ls
RAGBRAI-visualization UCLA slides-week-2.key
nathanlangholz@Nathans-Air Documents % ls -l
total 33016
drwxr-xr-x@ 9 nathanlangholz staff 288 Mar 31 15:28 RAGBRAI-visualization
drwxr-xr-x 4 nathanlangholz staff 128 Apr 1 15:52 UCLA
-rw-r--r--@ 1 nathanlangholz staff 15982151 Apr 7 14:38 slides-week-2.key
nathanlangholz@Nathans-Air Documents %



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


Shorthand for your present working directory
The file's size in bytes
Shorthand for the directory one level above
Read, write, execute permissions
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$
Your username
The group you belong to that owns the file
The file's creation date
Filename

Kinds of Files

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

Unix filesystem conventions places (shared, 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 command called open that decides which program to use

Permission bits

Unix can support many users on a single system and each user can belong to one or more groups

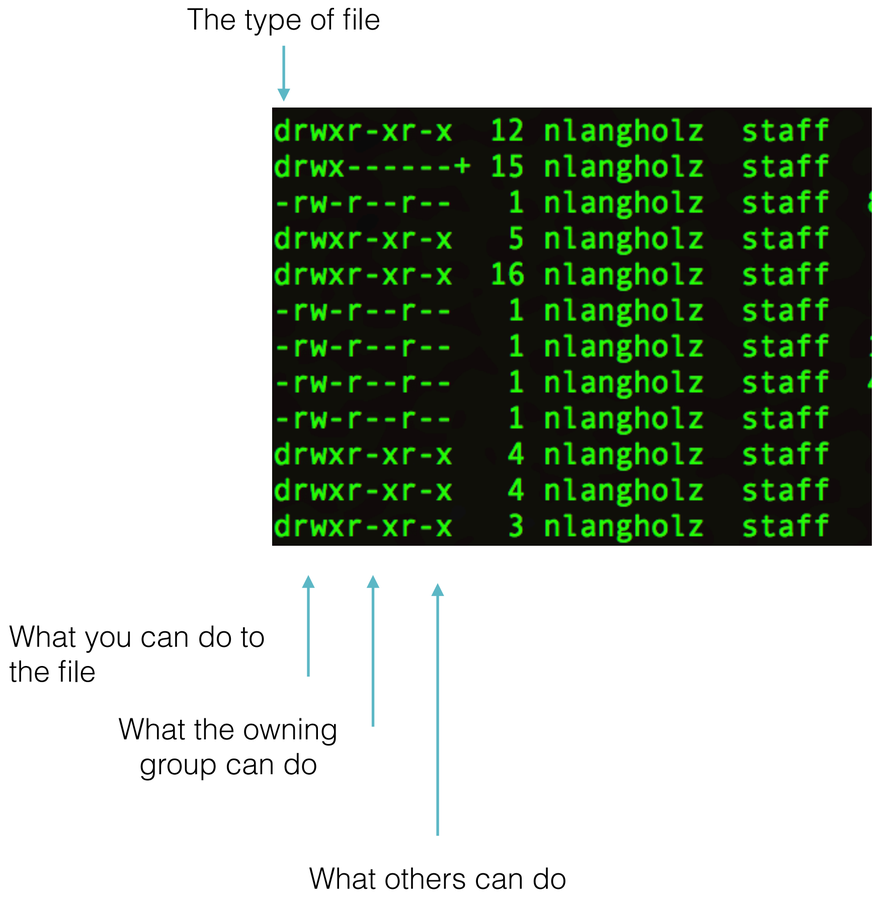
every file in a Unix filesystem is owned by some user and one of that user's groups; each file also has a set of permissions specifying which users can

`r: read` or **`w: write (modify)`** or **`x: execute`**

the file; these are specified with three 'bits' and we need three sets of bits to define what the user can do, what their group (that owns the file) can do and what others can do

the command **`chmod`** changes the permissions on a file but we will leave that for you to discover on your own

Permission bits



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 datasciencetoolbox/dsatcl2e
```

Data Science at the Command Line

To run the image

```
docker run --rm -it datasciencetoolbox/dsatcl2e
```

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 datasciencetoolbox/dsatcl2e
```

in Windows command line

```
C:\> docker run --rm -it -v "%cd%":/data datasciencetoolbox/dsatcl2e
```

in Windows powershell

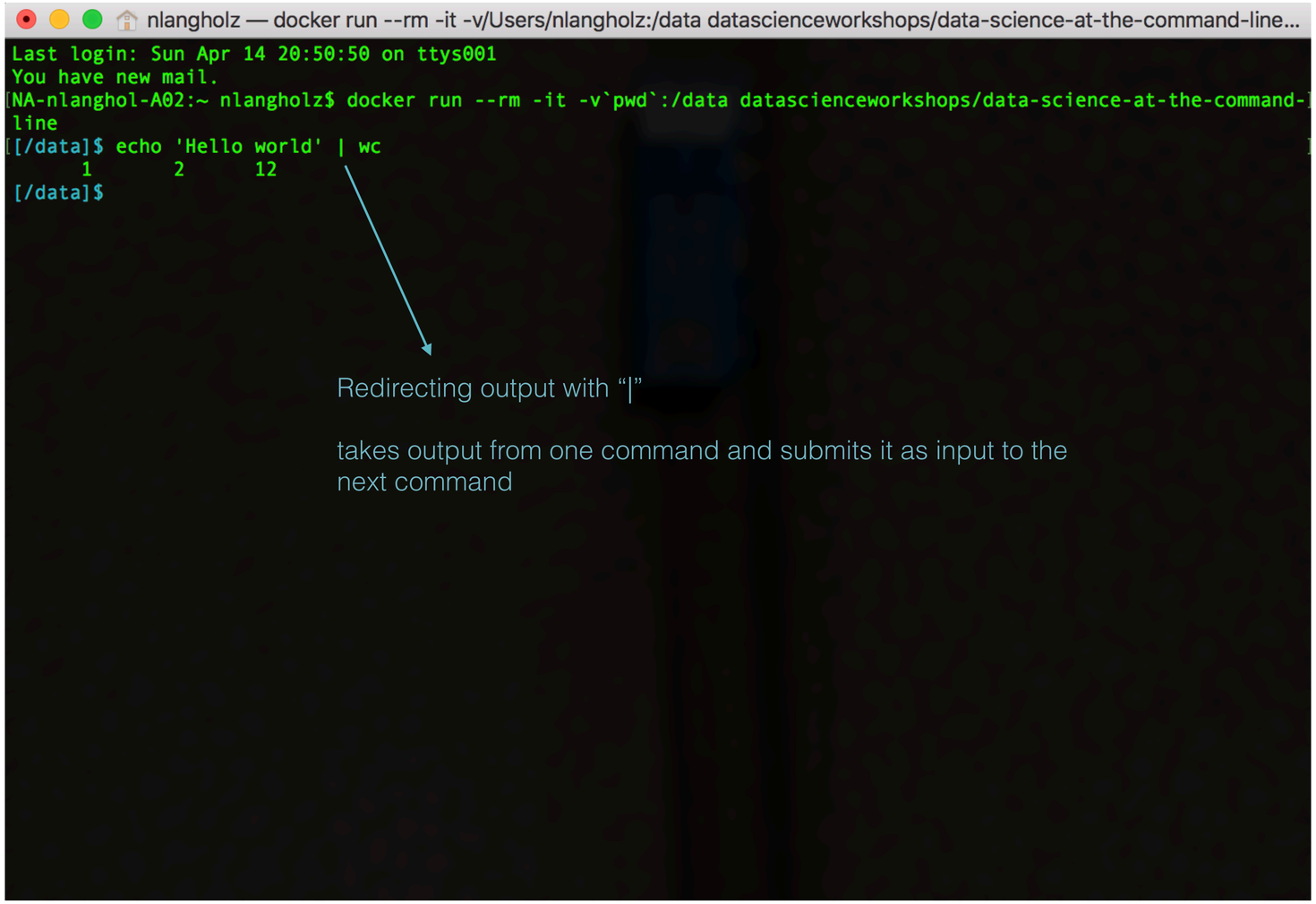
```
PS C:\> docker run --rm -it -v ${PWD}:/data datasciencetoolbox/
dsatcl2e
```

To make sure everything is running correctly, type

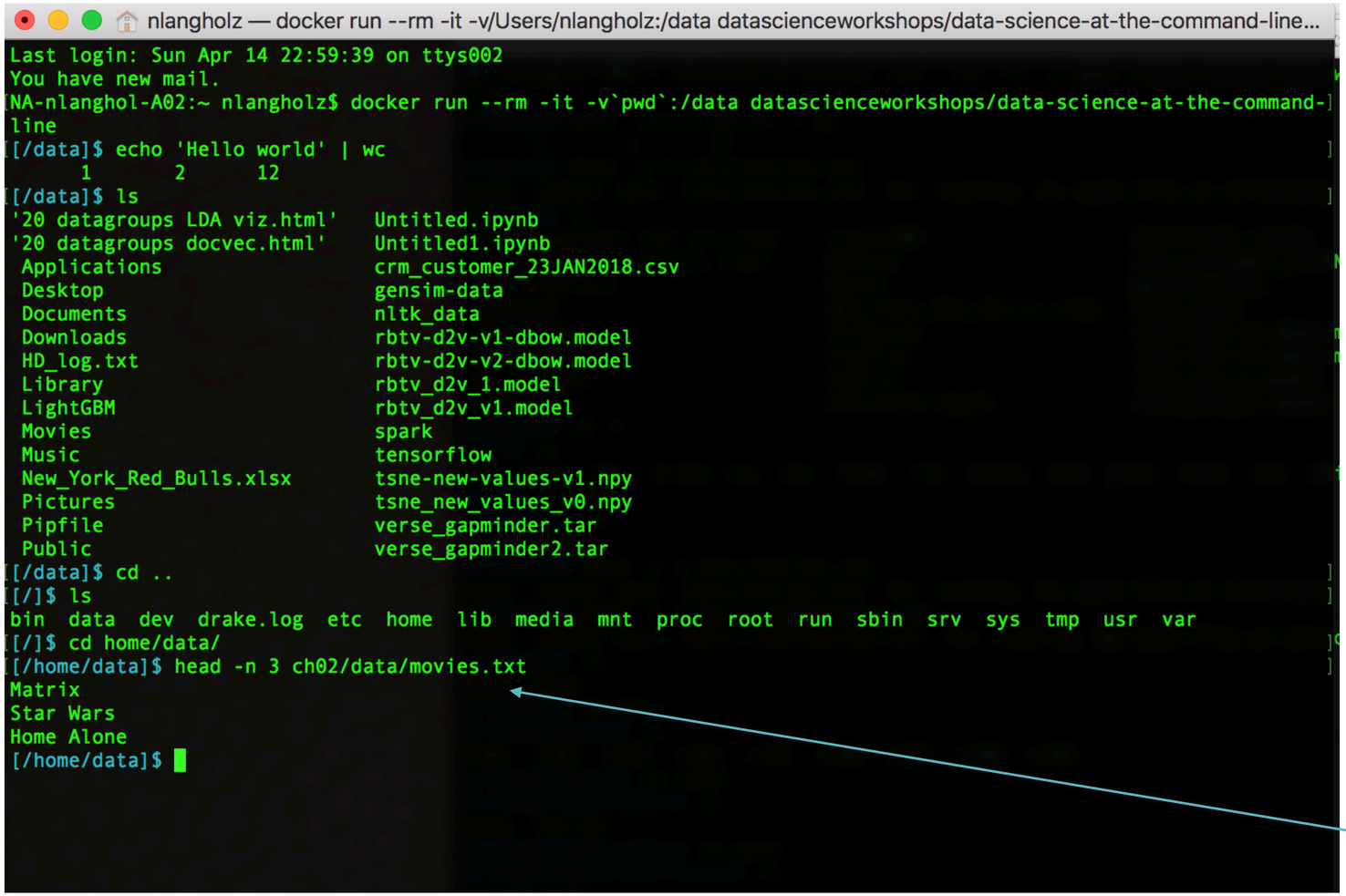
```
cowsay “ready to go!”
```

This environment we now have is a bash shell running on a Linux operating system.

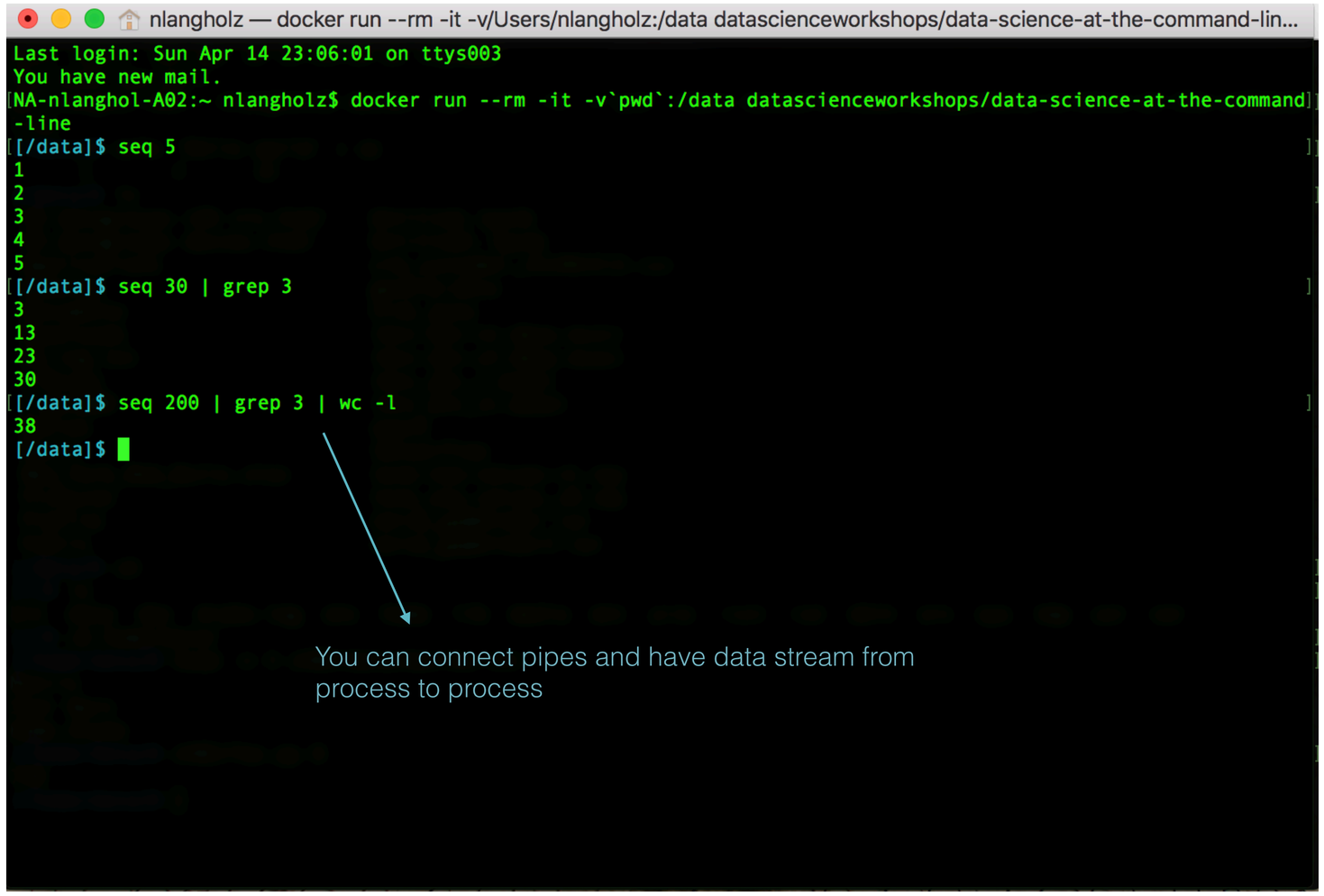
Note: this only works in this environment not in all terminal windows


nlangholz — docker run --rm -it -v/Users/nlangholz:/data datascienceworkshops/data-science-at-the-command-line...
Last login: Sun Apr 14 20:50:50 on ttys001
You have new mail.
NA-nlanghol-A02:~ nlangholz$ docker run --rm -it -v`pwd`:/data datascienceworkshops/data-science-at-the-command-line
[/data]$ echo 'Hello world' | wc
1 2 12
[/data]$
Redirecting output with "|"
takes output from one command and submits it as input to the next command

Listing pwd, then changing to directory one level up to access docker container book contents



Listing the first 3 movies from the file movies.txt


nlangholz — docker run --rm -it -v/Users/nlangholz:/data datascienceworkshops/data-science-at-the-command-lin...
Last login: Sun Apr 14 23:06:01 on ttys003
You have new mail.
NA-nlanghol-A02:~ nlangholz$ docker run --rm -it -v`pwd`:/data datascienceworkshops/data-science-at-the-command-line
[/data]$ seq 5
1
2
3
4
5
[/data]$ seq 30 | grep 3
3
13
23
30
[/data]$ seq 200 | grep 3 | wc -l
38
[/data]$
You can connect pipes and have data stream from process to process

“Grep was invented for me. I was making a program to read text aloud through a voice synthesizer. As I invented phonetic rules I would check Webster's dictionary for words on which they might fail. For example, how do you cope with the digraph `ui', which is pronounced many different ways: `fruit', `guile', `guilty', `anguish', `intuit', `beguine'? I would break the dictionary up into pieces that fit in ed's limited buffer and use a global command to select a list. I would whittle this list down by repeated scannings with ed to see how each proposed rule worked.”

The process was tedious, and terribly wasteful, since the dictionary had to be split (one couldn't afford to leave a split copy on line). Then ed copied each part into /tmp, scanned it twice to accomplish the g command, and finally threw it away, which takes time too.”

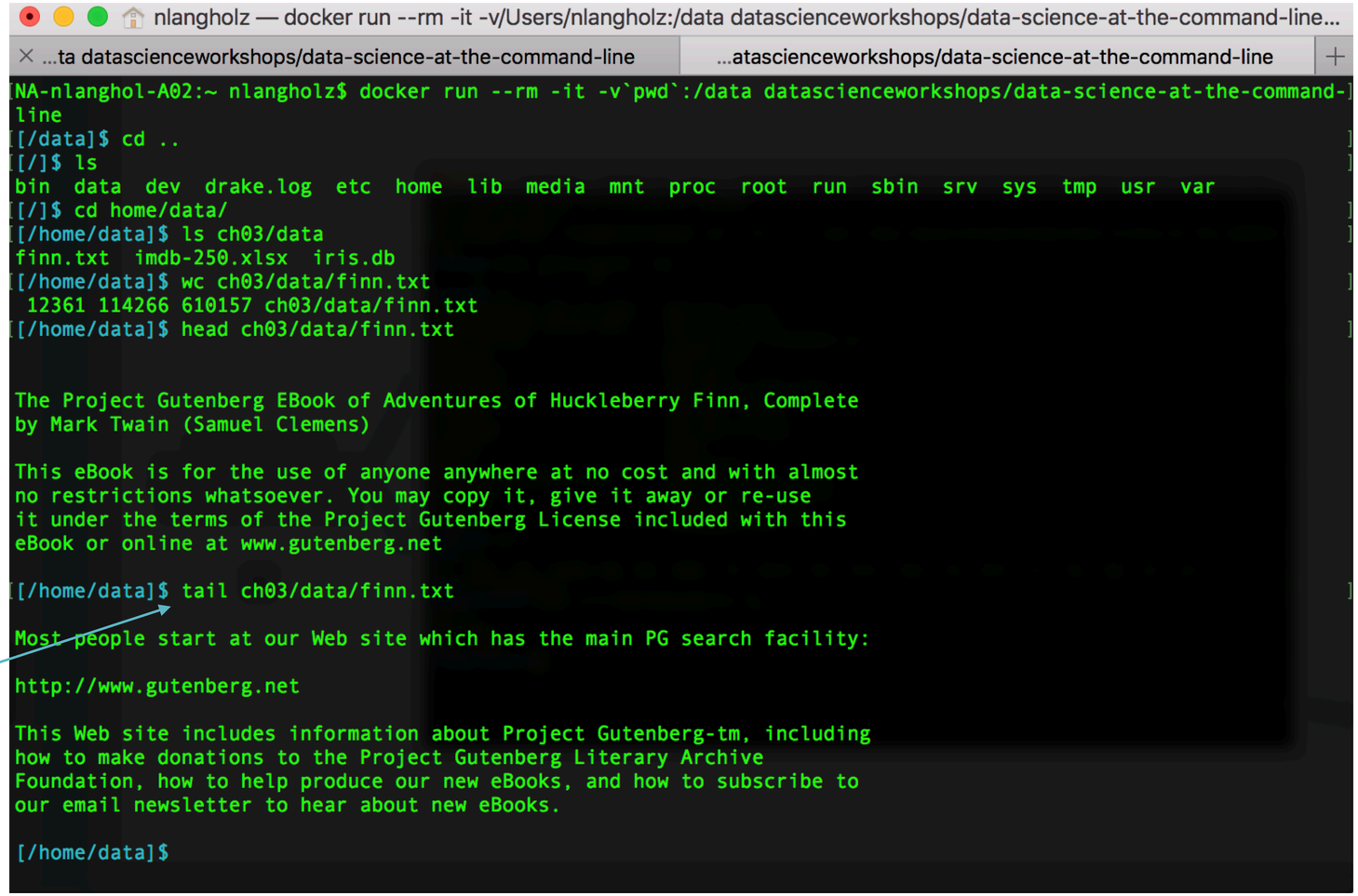
- Doug McIlroy, Adjunct Professor of Computer Science Dartmouth college

Chapter 9, On the Early History and Impact of Unix Tools to Build the Tools for a New Millennium
http://www.columbia.edu/~rh120/ch001j.c11

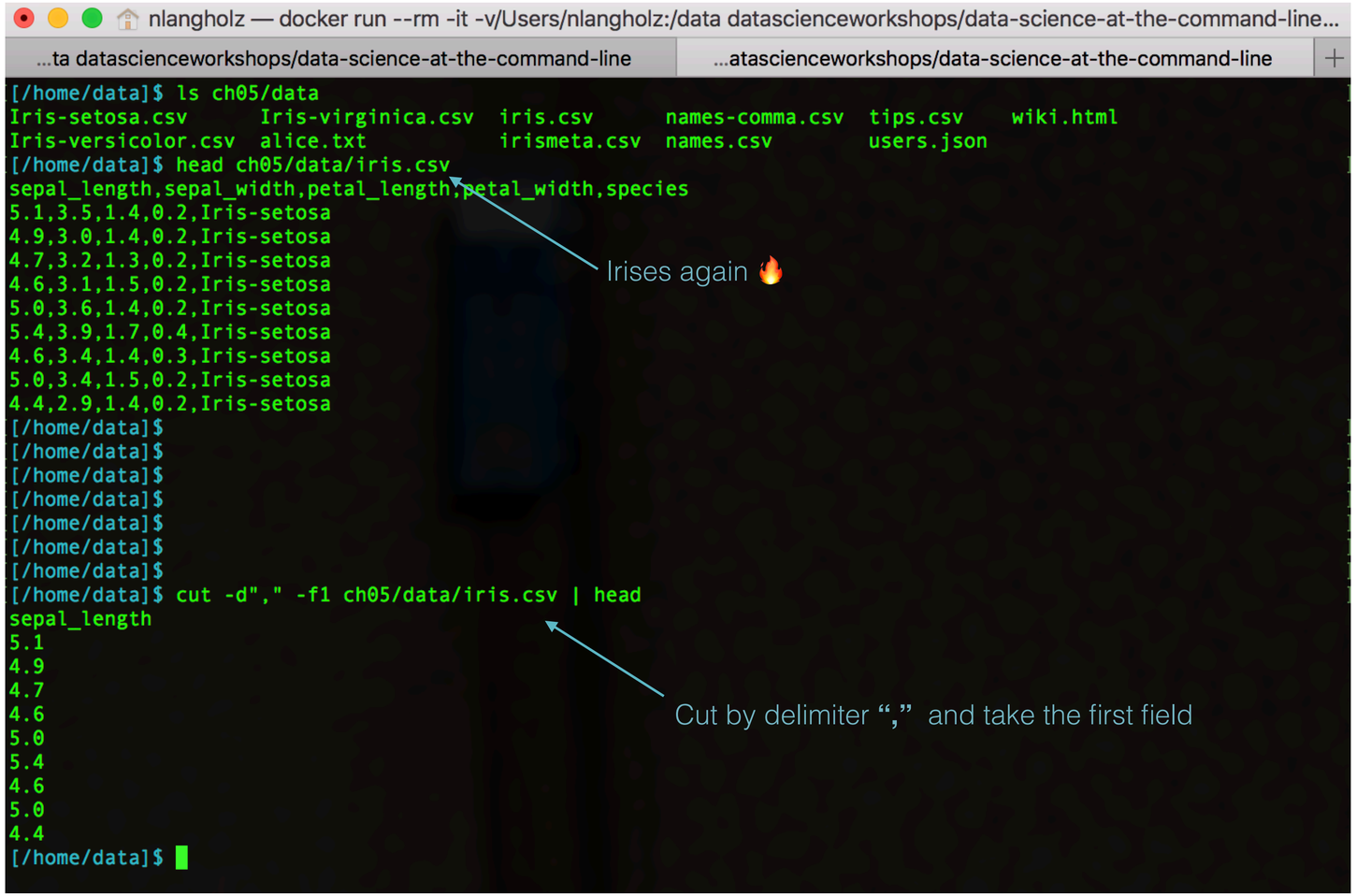
“One afternoon I asked Ken Thompson if he could lift the regular expression recognizer out of the editor and make a one-pass program to do it. He said yes. The next morning I found a note in my mail announcing a program named grep. It worked like a charm. When asked what that funny name meant, Ken said it was obvious. It stood for the editor command that it simulated, g/re/p (global regular expression print).”

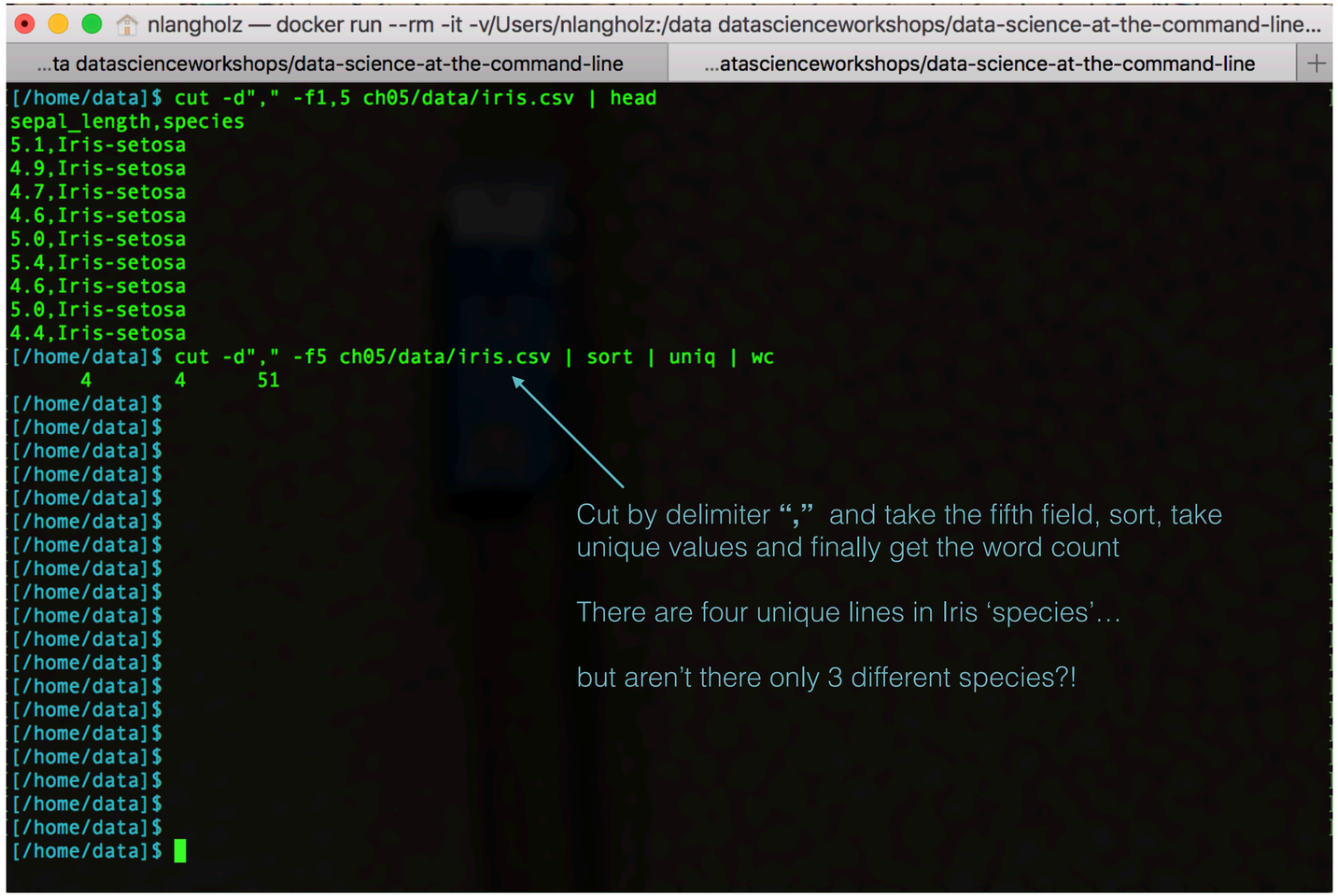
- Doug McIlroy, Adjunct Professor of Computer Science Dartmouth college

Chapter 9, On the Early History and Impact of Unix Tools to Build the Tools for a New Millennium
http://www.columbia.edu/~rh120/ch001j.c11

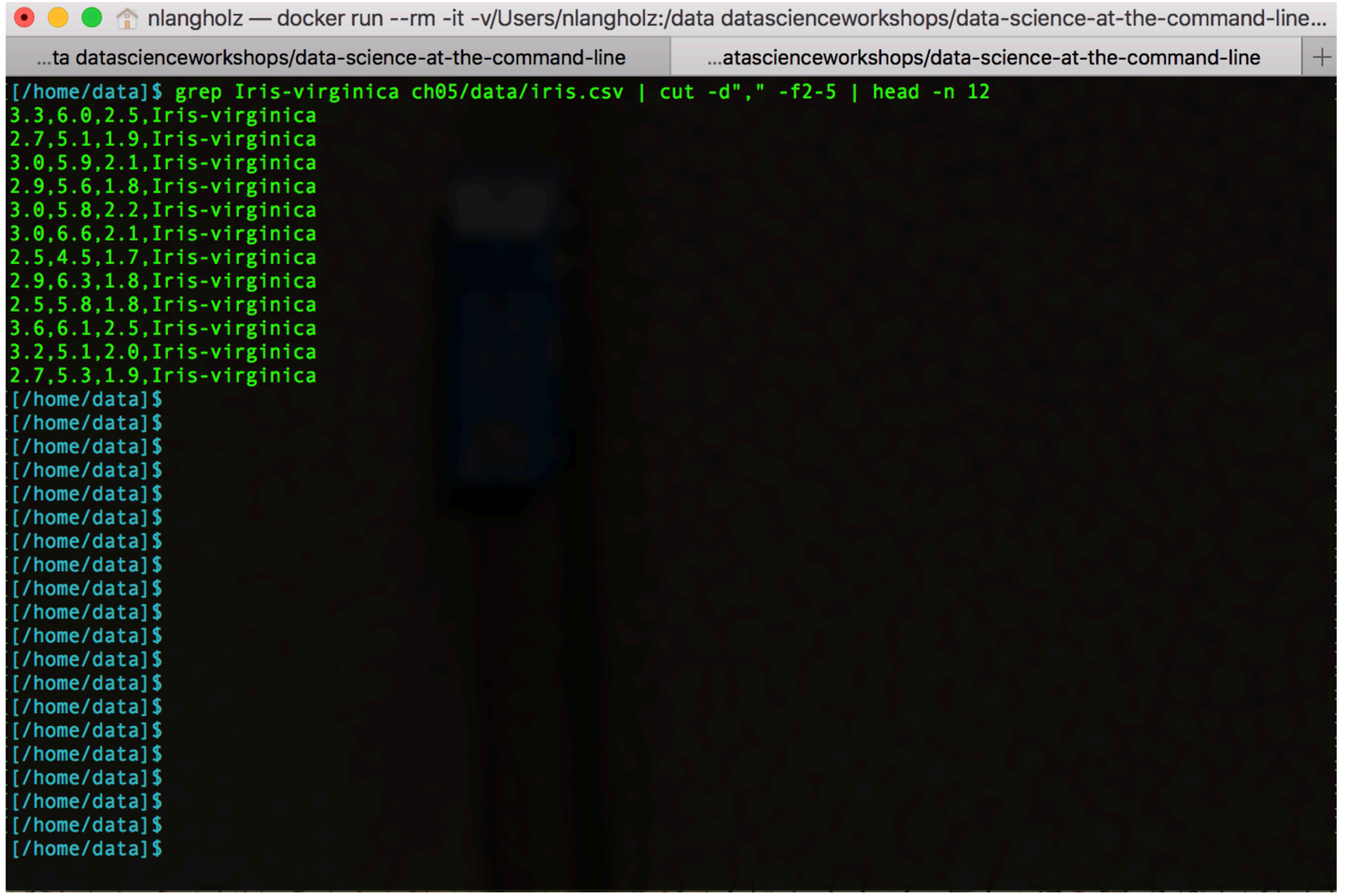


Similar to head we have
'tail' end of the document


nlangholz — docker run --rm -it -v/Users/nlangholz:/data datascienceworkshops/data-science-at-the-command-line...
...ta datascienceworkshops/data-science-at-the-command-line
...atascienceworkshops/data-science-at-the-command-line
[/home/data]$ ls ch05/data
Iris-setosa.csv Iris-virginica.csv iris.csv names-comma.csv tips.csv wiki.html
Iris-versicolor.csv alice.txt irismeta.csv names.csv users.json
[/home/data]$ head ch05/data/iris.csv
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
[/home/data]$
[/home/data]$
[/home/data]$
[/home/data]$
[/home/data]$
[/home/data]$
[/home/data]$
[/home/data]$ cut -d"," -f1 ch05/data/iris.csv | head
sepal_length
5.1
4.9
4.7
4.6
5.0
5.4
4.6
5.0
4.4
[/home/data]$
Irises again 🔥
Cut by delimiter “,” and take the first field


nlangholz — docker run --rm -it -v/Users/nlangholz:/data datascienceworkshops/data-science-at-the-command-line...
...ta datascienceworkshops/data-science-at-the-command-line
...atascienceworkshops/data-science-at-the-command-line
[/home/data]$ cut -d"," -f1,5 ch05/data/iris.csv | head
sepal_length,species
5.1,Iris-setosa
4.9,Iris-setosa
4.7,Iris-setosa
4.6,Iris-setosa
5.0,Iris-setosa
5.4,Iris-setosa
4.6,Iris-setosa
5.0,Iris-setosa
4.4,Iris-setosa
[/home/data]$ cut -d"," -f5 ch05/data/iris.csv | sort | uniq | wc
4 4 51
[/home/data]$
Cut by delimiter “,” and take the fifth field, sort, take unique values and finally get the word count
There are four unique lines in Iris ‘species’…
but aren’t there only 3 different species?!


nlangholz — docker run --rm -it -v/Users/nlangholz:/data datascienceworkshops/data-science-at-the-command-line...
...ta datascienceworkshops/data-science-at-the-command-line
...atascienceworkshops/data-science-at-the-command-line
[/home/data]$ cut -d"," -f5 ch05/data/iris.csv | sort | uniq | wc
4 4 51
[/home/data]$
[/home/data]$ cut -d"," -f5 ch05/data/iris.csv | sort | uniq -c | sort -rn
50 Iris-virginica
50 Iris-versicolor
50 Iris-setosa
1 species
[/home/data]$
Well, yes, since there is no notion of header (or column names) here
in reality there are 50 of each species and then the 1 row of 'species' at the top



You can connect pipes and have data process to process

Sending output to a file with “>”

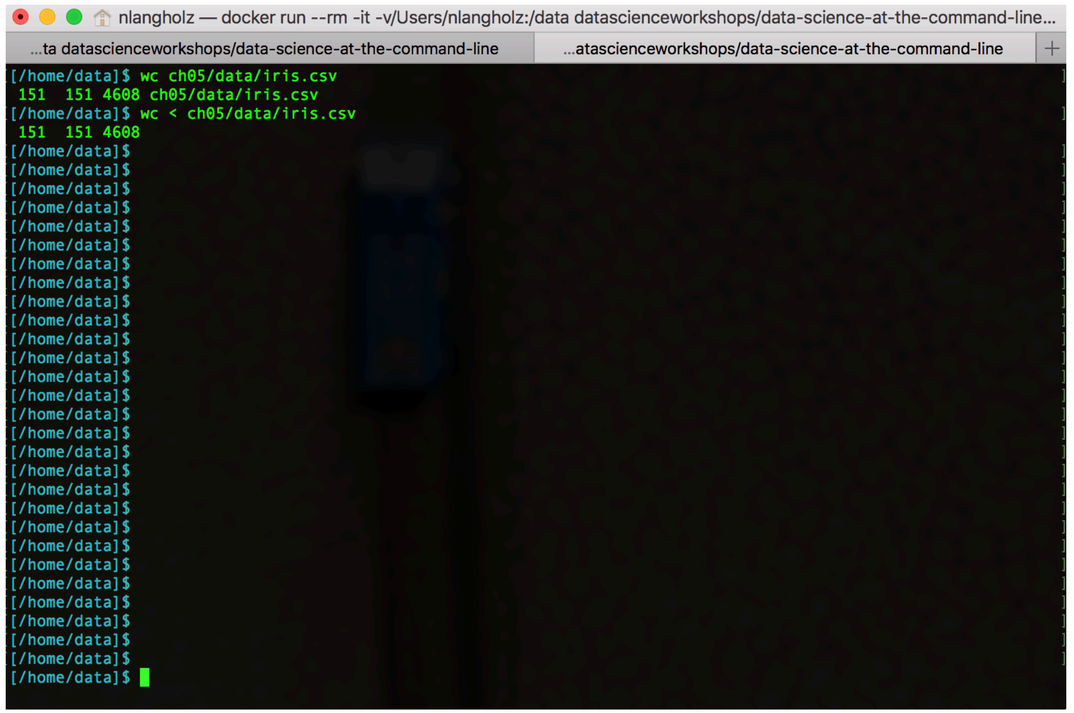
With this form of redirection, we take a stream of processed data and store it in a file

Example

```
cut -d”,” -f5 ch05/data/iris.csv > species
```

Taking input from a file with “<”

With this form of redirection, we create an input stream from a file



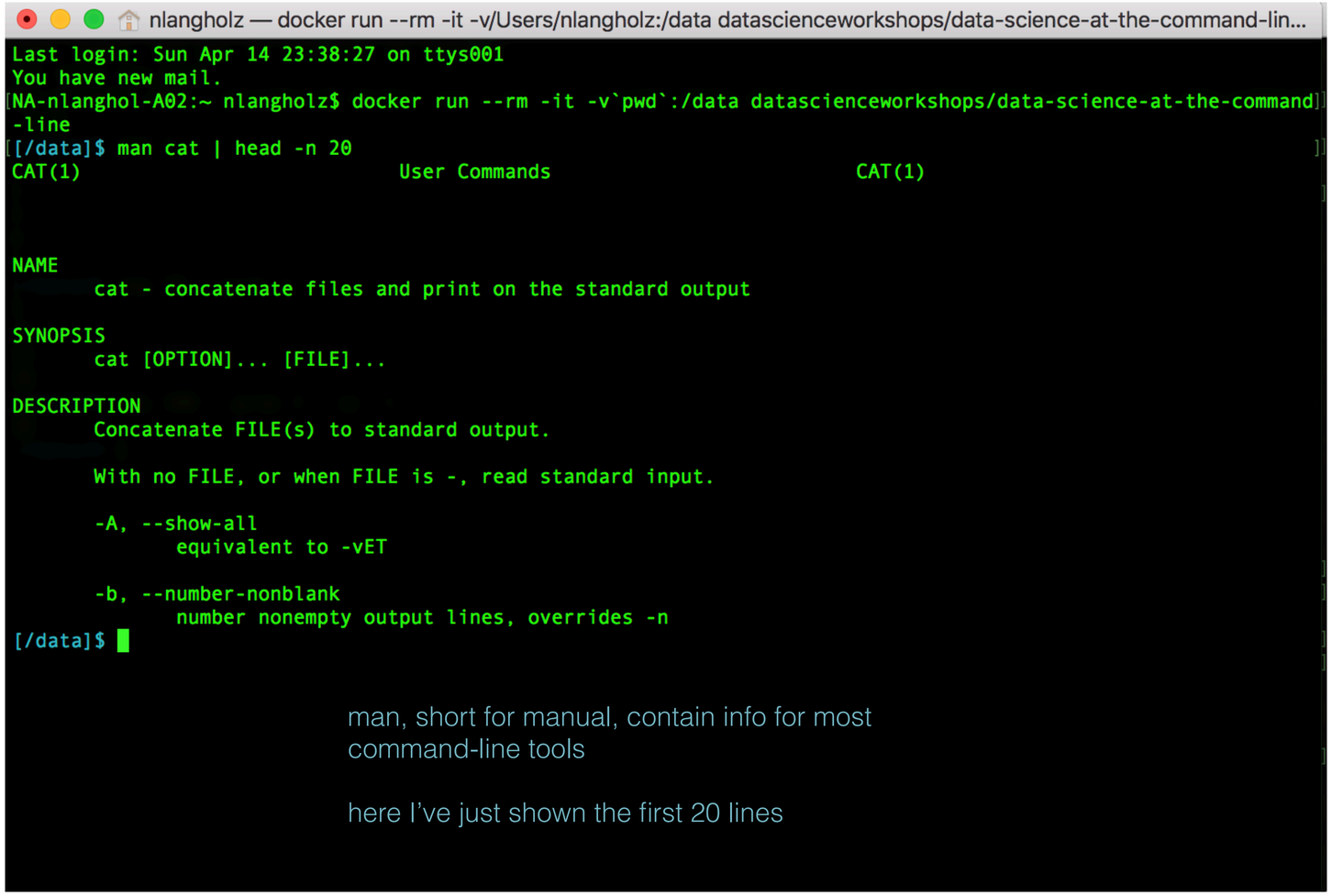
Read the man and help pages!

If the command uniq is unfamiliar you can look up its usage

```
man uniq
```

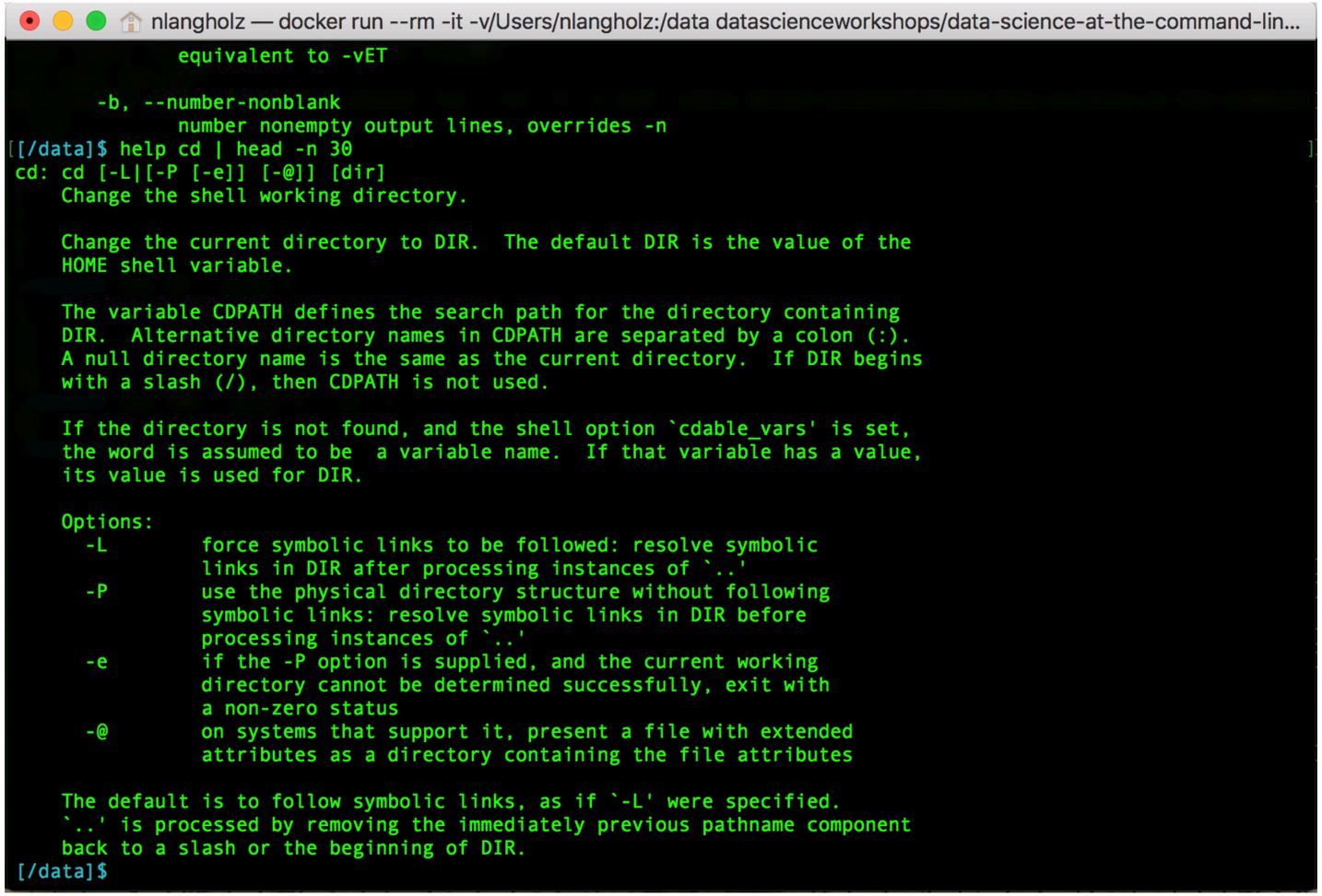
or

```
uniq --help
```



man, short for manual, contain info for most command-line tools

here I've just shown the first 20 lines

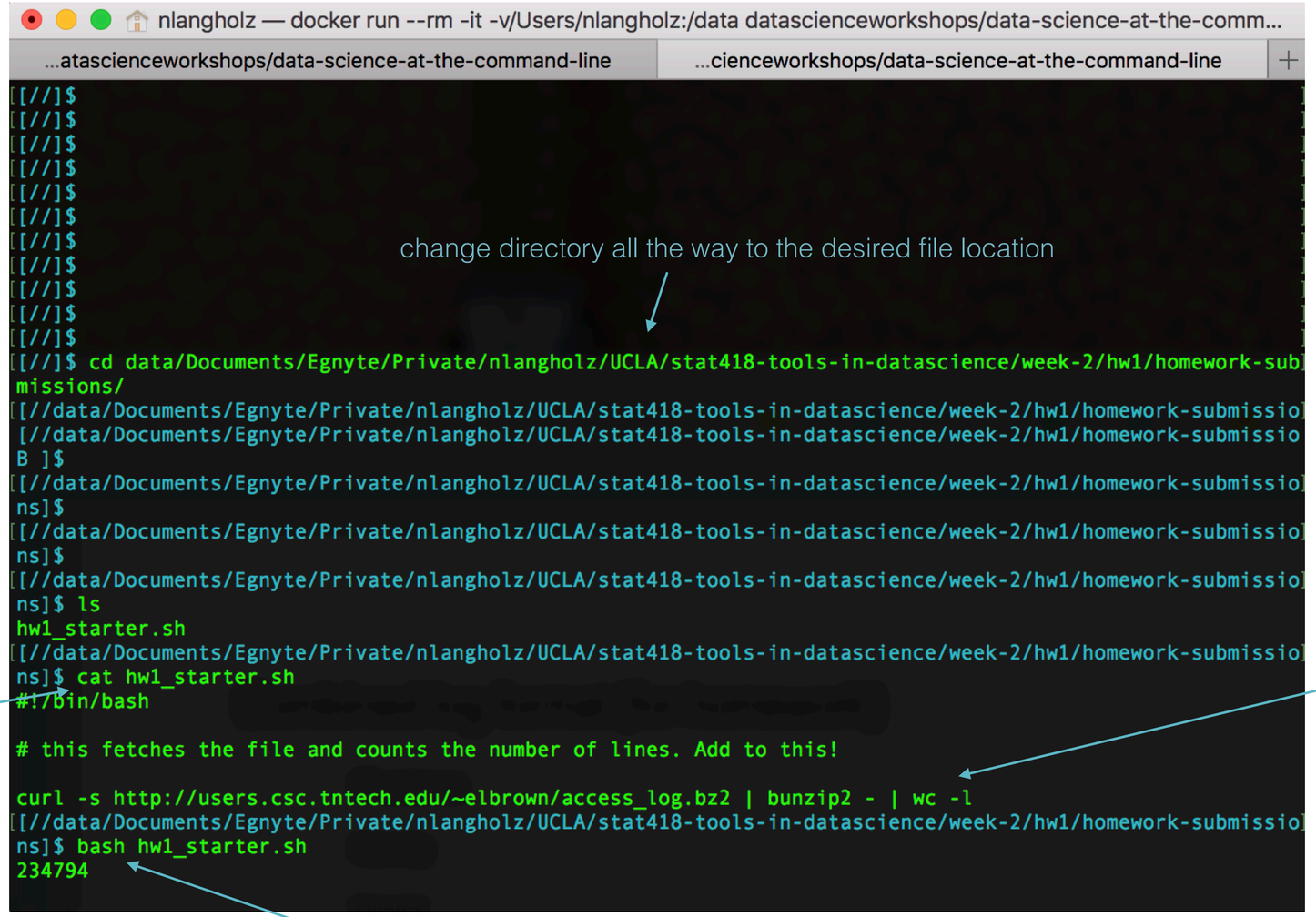


Not every command-line tool has a man page. For some shell builtins there is only a help page

again I've shown only the top 30 lines

There is a lot packed into this slide

cat returns the entirety of the file in the terminal



curl goes to the specified url to download the zip file and bunzip2 will unzip

what does bash do?

Running a shell script

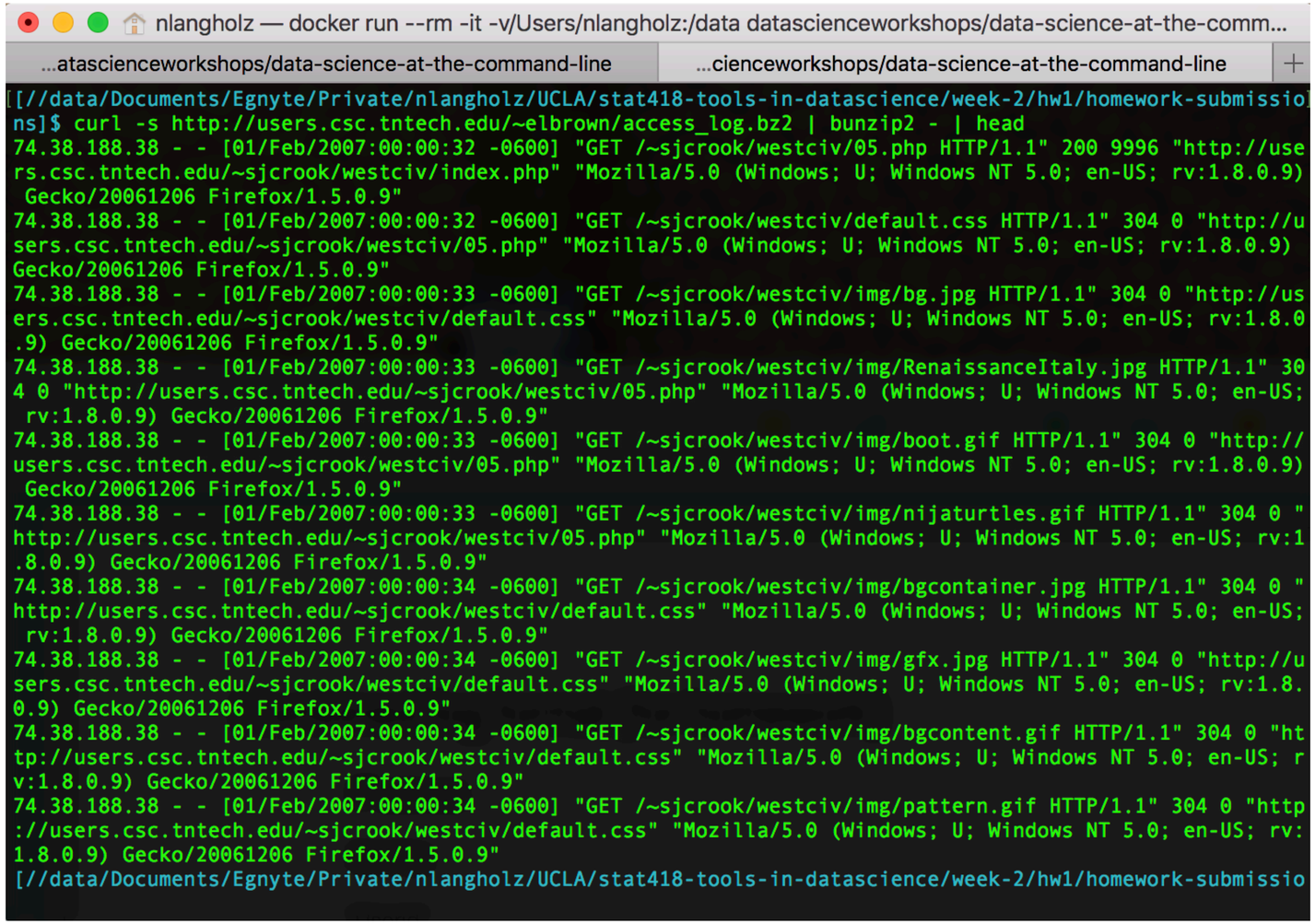
There are two ways to run a shell script; you can either execute it within a new shell or make the file executable. Executing in a new shell using:

```
$ bash <filename.sh>
```

This should explain the .sh suffix we used for the homework starter filename; this naming convention will help you (and others) recognize this as a shell script

Making a file executable means making it become just like any command Unix knows. This requires changing file permissions which we saw briefly earlier but we won't cover in this lecture

What the web log actually looks like…



Combined log format (for homework)

IP address

Identity

Userid

Date

Request

Status

Bytes

Referrer

Agent

Accessing a URL

When accessing a URL (uniform resource locator) through your browser the data that is being downloaded can be interpreted; html as a website, MP3 as played music, pdf automatically downloaded

When cURL is used to access a URL the data is downloaded and is printed to standard output. As we have seen we can simply specify as an arguement

```
$ curl -s http://www.gutenberg.org/files/76/76-0.txt | head -n 10
```

This downloads the book Adventures of Huckleberry Finn by Mark Twain from Project Gutenberg that we saw before. The `-s` command stands for *silent* so a progress meter is disabled so we can pipe to another command

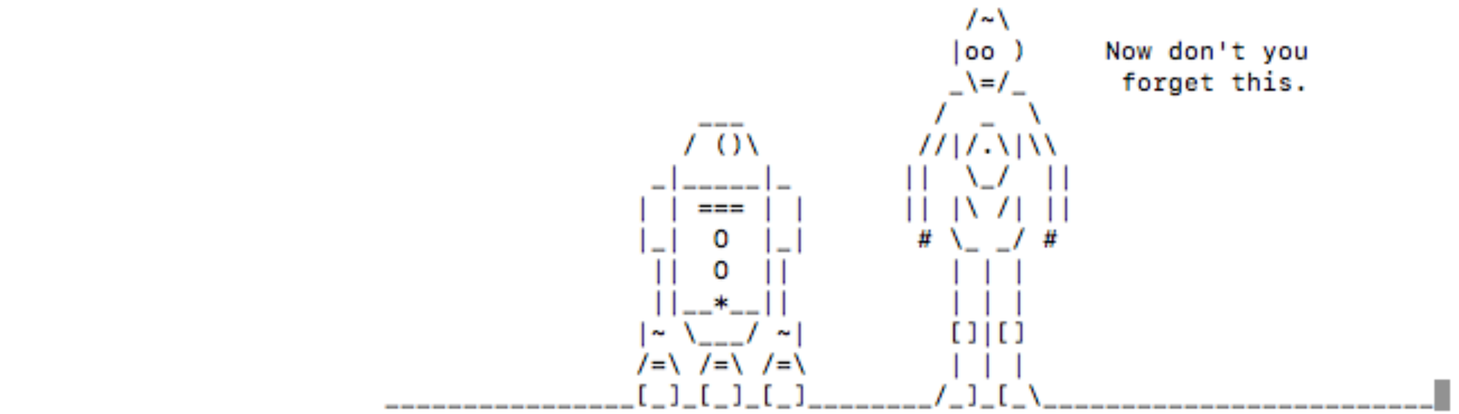
Saving a URL

Or course we can continue to combine our commands to save a file

```
$ curl -s http://www.gutenberg.org/files/76/76-0.txt > data/finn.txt
```

Don’t need to specify **`-s`** in this case

Star Wars: Fun, but… useless?



try

```
$ nc towel.blinkenlights.nl 23
```

Web Scraping (…still in the shell)

Scraping the New York Times

The New York Times is one of the most well-known, reputable, widely circulated news outlets worldwide. The have the complete archive of all their articles available dating back to 1851. Obviously this is a huge wealth of information that we can access through their developers API.

Why does NYT (or anyone) provide an API? In their words:

3. Why are you offering APIs?

Like many organizations, we hope to encourage innovation through collaboration. When you build applications, create mashups and otherwise reveal the potential of our data, we learn more about what our readers want and gain insight into how news and information can be reimagined. We're hoping you'll show us what's next for The Times.

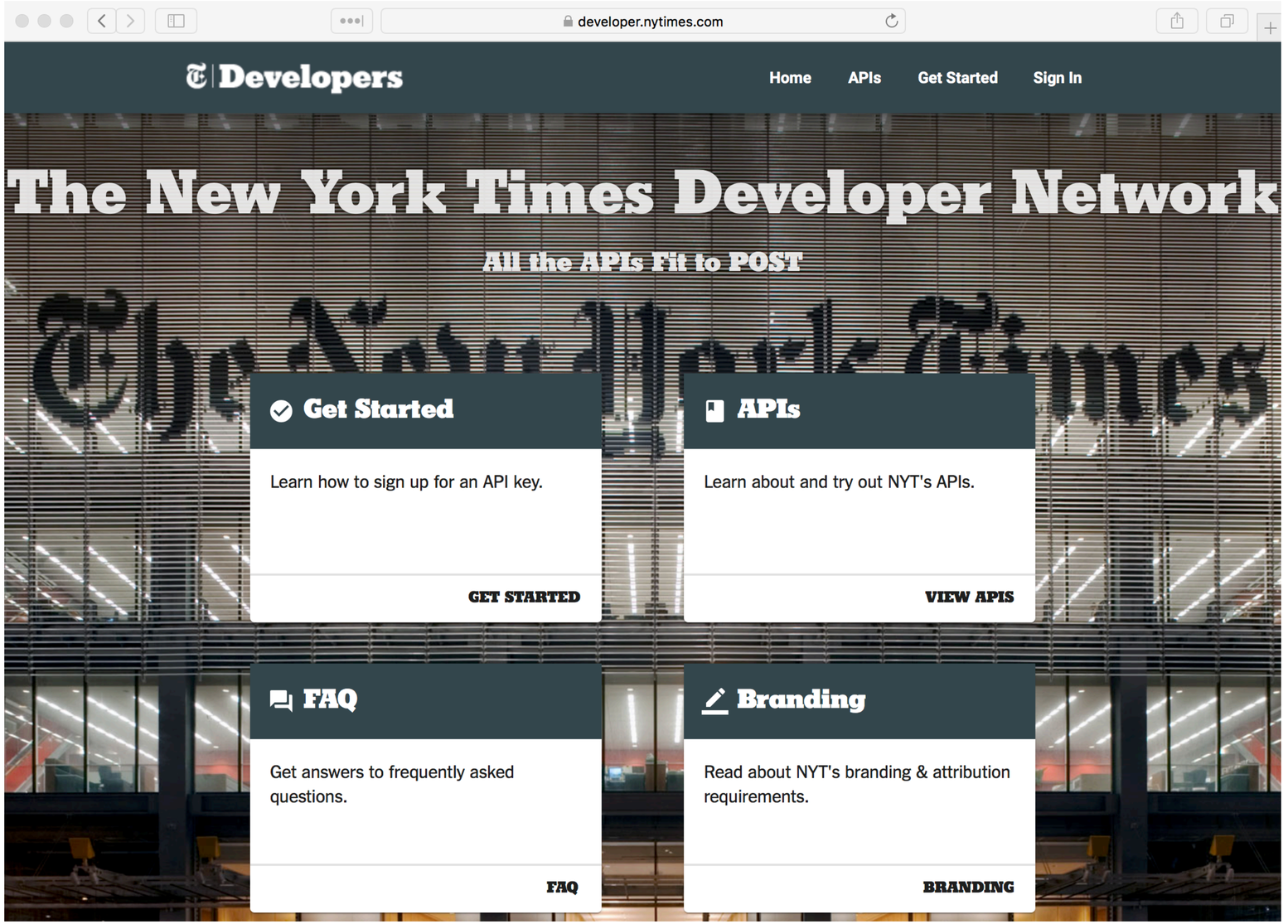
But we also have a simpler, more compelling reason: journalism. To inform the public or tell a story, we use articles, photos, videos, interactive graphics, slideshows and more. Data has always been the primary force behind those features, and now it can become a feature in its own right. Our APIs help us fulfill the newspaper's journalistic mission by putting more information in the hands of the public — and they also expand that mission by giving users the ability to find and tell their own stories.

With the sad news yesterday of the Notre-Dame fire, today's NYT front page is full of articles about the cathedral.

I was curious as to how many articles are typically written about Notre-Dame given that it sees about ~12 millions visitors a year (according to the Paris Convention and Visitors Bureau) how much of an increase we saw as a result

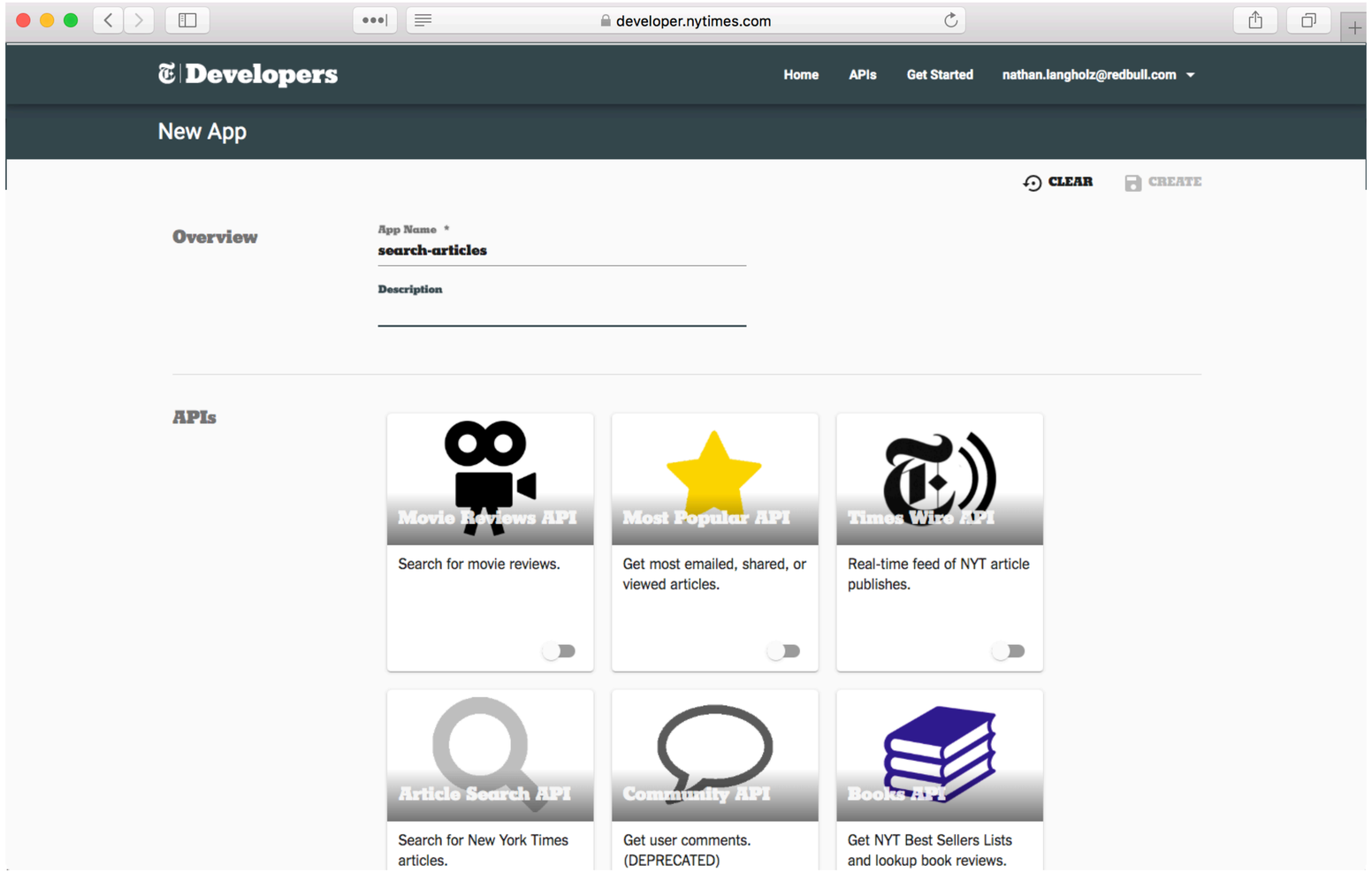
This is the New York Times Developer Network

If you want to follow along you will need to make an account to get your own API key



You will need to create an app; name doesn't matter

We will be using the Article Search API, but it is a similar process to use any of the others



FAQ

1. What are the Times APIs?

Our APIs (**Application Programming Interfaces**) allow you to programmatically access New York Times data for use in your own applications. Our goal is to facilitate a wide range of uses, from custom link lists to complex visualizations. Why just read the news when you can hack it?

NYT currently has ten public APIs: Archive, Article Search, Books, Community, Geographic, Most Popular, Semantic, Times Newswire, TimesTags, and Top Stories.

2. Who is the intended audience for the Times APIs?

We've designed our APIs for the web developer community, but all non-commercial users are welcome. See our **Terms of Use** for more information.

3. Why are you offering APIs?

Like many organizations, we hope to encourage innovation through collaboration. When you build applications, create mashups and otherwise reveal the potential of our data, we learn more about what our readers want and gain insight into how news and information can be reimagined. We're hoping you'll show us what's next for The Times.

But we also have a simpler, more compelling reason: journalism. To inform the public or tell a story, we use articles, photos, videos, interactive graphics, slideshows and more. Data has always been the primary force behind those features, and now it can become a feature in its own right. Our APIs help us fulfill the newspaper's journalistic mission by putting more information in the hands of the public — and they also expand that mission by giving users the ability to find and tell their own stories.

4. What kinds of data can I access with Times APIs?

Please see our APIs page for the current list of available APIs.

5. How do I use the Times APIs?

Our APIs use a RESTful style and a resource-oriented architecture. Calls are made via HTTPS requests. Your request URIs should be patterned after the examples in the API documentation, and you should always include your API key in a query string. See the documentation for each API for more details on request parameters and URI structure.

6. Why can't I access all NYTimes.com content?

For each set of data we open up to the developer community, we have to consider a host of issues, ranging from load balancing to copyright law. If you have ideas for our APIs or are hoping to get access to a specific kind of data, contact us at code@nytimes.com.

7. Why do I have to register for a key to use your APIs?

Key registration allows us to monitor usage levels and ensure that developers are complying with our **Terms of Use**. We respect your privacy and will not share your registration information with third parties.

8. Do I have to agree to your Terms of Use just to test an API?

Our **Terms of Use** apply to all uses of our APIs. Please agree to the terms before testing or using an API.

9. Can I use your APIs commercially?

Our APIs are for non-commercial use only. For details, see our **Terms of Use**. If you have a commerical use case, please visit https://nytlicensing.com/ or email code@nytimes.com.

10. What do you mean by "commercial purposes"?

Here are a few examples of what we consider commercial purposes:

- Selling New York Times content or data in any application.
- Charging a subscription fee for any New York Times content or data.
- Selling any application built with one of our APIs.
- Uses our content inside an application that is paid or has a paid tier.

If you aren't sure whether your plans constitute "commercial purposes," please contact us at code@nytimes.com. For commercial use please visit https://nytlicensing.com/.

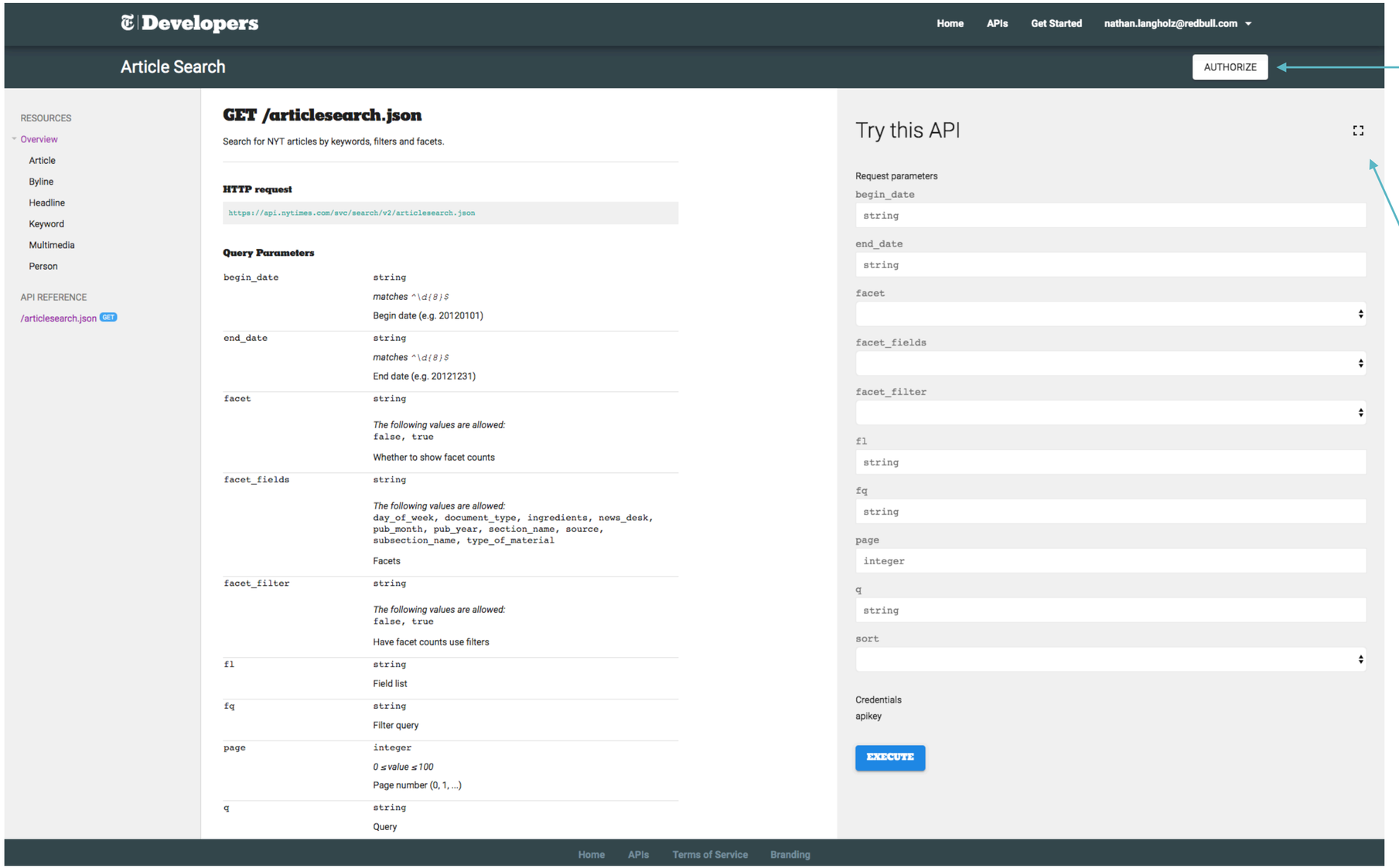
11. Is there an API call limit?

Yes, there are two rate limits per API: 4,000 requests per day and 10 requests per minute. You should sleep 6 seconds between calls to avoid hitting the per minute rate limit. If you need a higher rate limit, please contact us at code@nytimes.com.

12. What response formats do you support?

Data is returned as JSON. Specific APIs may also return other formats. See the documentation for each API for more details.

This is important to note



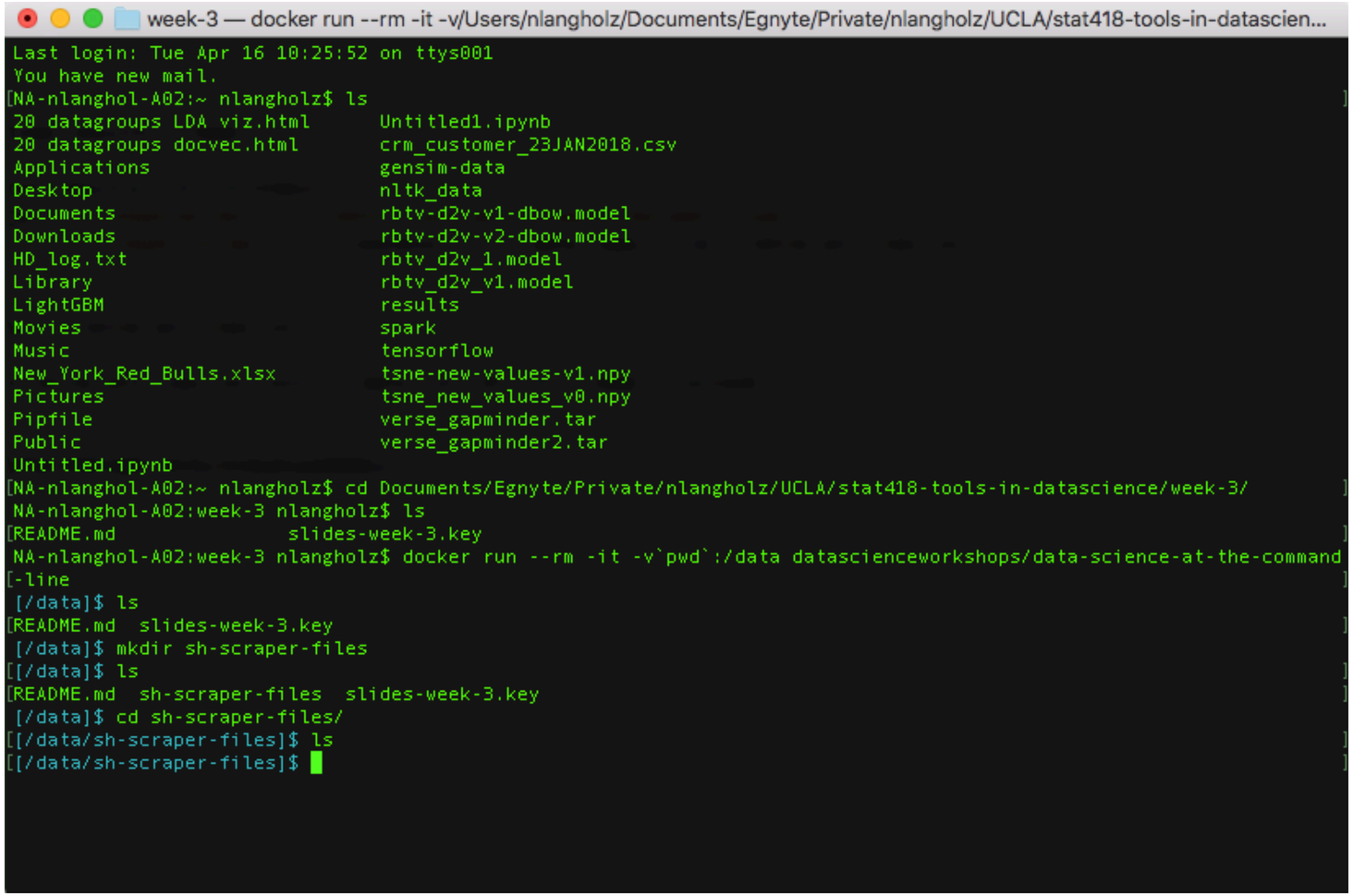
You will need to authorize your credentials

You can also test query and get the resulting url to test in a browser

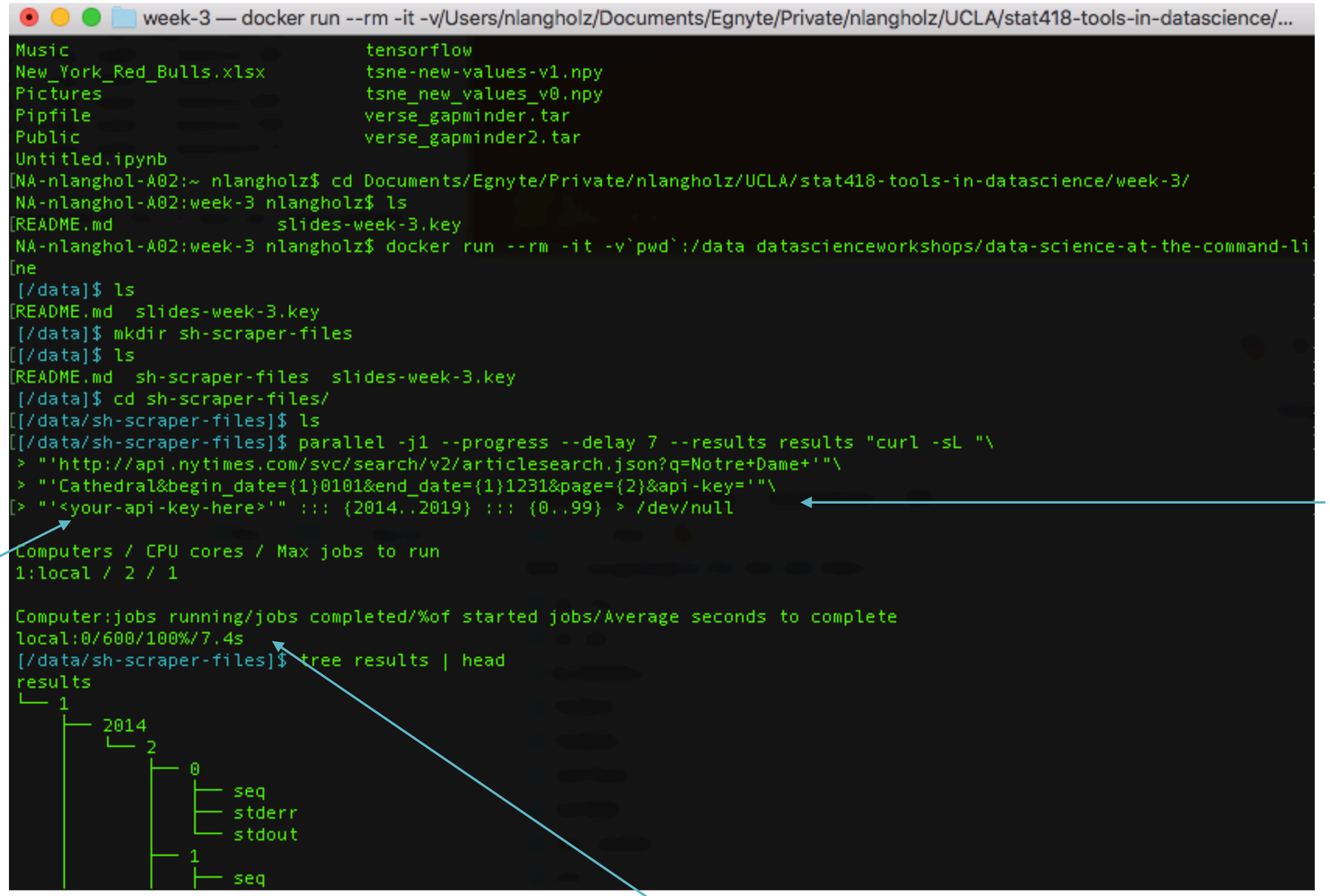
{"status":"OK","copyright":"Copyright (c) 2019 The New York Times Company. All Rights Reserved.","response":{"docs":[{"web_url":"https://www.nytimes.com/2019/04/15/opinion/notre-dame-paris-fire.html","snippet":"A hundred years from now, people will still be talking about the fire of 2019.","lead_paragraph":"A hundred years from now, people will still be talking about the fire of 2019.","blog":{},"source":"The New York Times","multimedia":
[{"rank":0,"subtype":"xlarge","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-articleLarge.jpg","height":450,"width":600,"legacy":
{"xlarge":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-articleLarge.jpg","xlargewidth":600,"xlargeheight":450},"subType":"xlarge","crop_name":"articleLarge"},
{"rank":0,"subtype":"wide","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-thumbWide.jpg","height":126,"width":190,"legacy":{"wide":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
thumbWide.jpg","widewidth":190,"wideheight":126},"subType":"wide","crop_name":"thumbWide"},{"rank":0,"subtype":"thumbnail","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
thumbStandard.jpg","height":75,"width":75,"legacy":{"thumbnail":"images/2019/04/15/opinion/15Druckerman/15Druckerman-thumbStandard.jpg","thumbnailwidth":75,"thumbnailheight":75},"subType":"thumbnail","crop_name":"thumbStandard"},
{"rank":0,"subtype":"articleInline","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-articleInline.jpg","height":142,"width":190,"legacy":
{},"subType":"articleInline","crop_name":"articleInline"},{"rank":0,"subtype":"blog225","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-
blog225.jpg","height":169,"width":225,"legacy":{},"subType":"blog225","crop_name":"blog225"},{"rank":0,"subtype":"blog427","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-
a7c2-743068a88ecf-blog427.jpg","height":320,"width":427,"legacy":{},"subType":"blog427","crop_name":"blog427"},
{"rank":0,"subtype":"blog480","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-blog480.jpg","height":360,"width":480,"legacy":
{},"subType":"blog480","crop_name":"blog480"},{"rank":0,"subtype":"blog533","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-
blog533.jpg","height":400,"width":533,"legacy":{},"subType":"blog533","crop_name":"blog533"},{"rank":0,"subtype":"blogSmallInline","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-
fd2e-4786-a7c2-743068a88ecf-blogSmallInline.jpg","height":113,"width":151,"legacy":{},"subType":"blogSmallInline","crop_name":"blogSmallInline"},
{"rank":0,"subtype":"blogSmallThumb","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-blogSmallThumb.jpg","height":50,"width":50,"legacy":
{},"subType":"blogSmallThumb","crop_name":"blogSmallThumb"},{"rank":0,"subtype":"facebookJumbo","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
facebookJumbo.jpg","height":549,"width":1050,"legacy":{},"subType":"facebookJumbo","crop_name":"facebookJumbo"},{"rank":0,"subtype":"filmstrip","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
filmstrip-v2.jpg","height":190,"width":190,"legacy":{},"subType":"filmstrip","crop_name":"filmstrip"},{"rank":0,"subtype":"hpLarge","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
hpLarge.jpg","height":287,"width":511,"legacy":{},"subType":"hpLarge","crop_name":"hpLarge"},{"rank":0,"subtype":"hpSmall","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-
a7c2-743068a88ecf-hpSmall.jpg","height":122,"width":163,"legacy":{},"subType":"hpSmall","crop_name":"hpSmall"},
{"rank":0,"subtype":"jumbo","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-jumbo.jpg","height":768,"width":1024,"legacy":
{},"subType":"jumbo","crop_name":"jumbo"},{"rank":0,"subtype":"largeHorizontal375","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-largeHorizontal375.jpg","height":250,"width":375,"legacy":
{},"subType":"largeHorizontal375","crop_name":"largeHorizontal375"},{"rank":0,"subtype":"largeHorizontalJumbo","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
largeHorizontalJumbo.jpg","height":682,"width":1024,"legacy":{},"subType":"largeHorizontalJumbo","crop_name":"largeHorizontalJumbo"},
{"rank":0,"subtype":"largeWidescreen573","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-largeWidescreen573.jpg","height":322,"width":573,"legacy":
{},"subType":"largeWidescreen573","crop_name":"largeWidescreen573"},{"rank":0,"subtype":"largeWidescreen1050","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
largeWidescreen1050.jpg","height":590,"width":1050,"legacy":{},"subType":"largeWidescreen1050","crop_name":"largeWidescreen1050"},
{"rank":0,"subtype":"master1050","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-master1050.jpg","height":787,"width":1050,"legacy":
{},"subType":"master1050","crop_name":"master1050"},{"rank":0,"subtype":"master180","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-
master180.jpg","height":135,"width":180,"legacy":{},"subType":"master180","crop_name":"master180"},{"rank":0,"subtype":"master315","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-
fd2e-4786-a7c2-743068a88ecf-master315.jpg","height":236,"width":315,"legacy":{},"subType":"master315","crop_name":"master315"},
{"rank":0,"subtype":"master495","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-master495.jpg","height":371,"width":495,"legacy":
{},"subType":"master495","crop_name":"master495"},{"rank":0,"subtype":"master768","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-
master768.jpg","height":576,"width":768,"legacy":{},"subType":"master768","crop_name":"master768"},{"rank":0,"subtype":"master675","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-
fd2e-4786-a7c2-743068a88ecf-master675.jpg","height":506,"width":675,"legacy":{},"subType":"master675","crop_name":"master675"},
{"rank":0,"subtype":"mediumFlexible177","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-mediumFlexible177.jpg","height":133,"width":177,"legacy":
{},"subType":"mediumFlexible177","crop_name":"mediumFlexible177"},{"rank":0,"subtype":"mediumSquare149","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-mediumSquare149-
v2.jpg","height":149,"width":149,"legacy":{},"subType":"mediumSquare149","crop_name":"mediumSquare149"},{"rank":0,"subtype":"mediumThreeByTwo210","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
mediumThreeByTwo210.jpg","height":140,"width":210,"legacy":{},"subType":"mediumThreeByTwo210","crop_name":"mediumThreeByTwo210"},
{"rank":0,"subtype":"mediumThreeByTwo225","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-mediumThreeByTwo225.jpg","height":150,"width":225,"legacy":
{},"subType":"mediumThreeByTwo225","crop_name":"mediumThreeByTwo225"},{"rank":0,"subtype":"mediumThreeByTwo440","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
mediumThreeByTwo440.jpg","height":293,"width":440,"legacy":{},"subType":"mediumThreeByTwo440","crop_name":"mediumThreeByTwo440"},
{"rank":0,"subtype":"mediumThreeByTwo378","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-mediumThreeByTwo378.jpg","height":252,"width":378,"legacy":
{},"subType":"mediumThreeByTwo378","crop_name":"mediumThreeByTwo378"},{"rank":0,"subtype":"mediumThreeByTwo252","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
mediumThreeByTwo252.jpg","height":168,"width":252,"legacy":{},"subType":"mediumThreeByTwo252","crop_name":"mediumThreeByTwo252"},
{"rank":0,"subtype":"miniMoth","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-miniMoth.jpg","height":70,"width":151,"legacy":{},"subType":"miniMoth","crop_name":"miniMoth"},
{"rank":0,"subtype":"moth","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-moth-v2.jpg","height":151,"width":151,"legacy":{},"subType":"moth","crop_name":"moth"},
{"rank":0,"subtype":"popup","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-popup.jpg","height":487,"width":650,"legacy":
{},"subType":"popup","crop_name":"popup"},{"rank":0,"subtype":"sfSpan","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-sfSpan.jpg","height":263,"width":395,"legacy":
{},"subType":"sfSpan","crop_name":"sfSpan"},{"rank":0,"subtype":"slide","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-
slide.jpg","height":450,"width":600,"legacy":{},"subType":"slide","crop_name":"slide"},{"rank":0,"subtype":"smallSquare168","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
smallSquare168.jpg","height":168,"width":168,"legacy":{},"subType":"smallSquare168","crop_name":"smallSquare168"},
{"rank":0,"subtype":"smallSquare252","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-smallSquare252.jpg","height":252,"width":252,"legacy":
{},"subType":"smallSquare252","crop_name":"smallSquare252"},{"rank":0,"subtype":"square320","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-square320-v2.jpg","height":320,"width":320,"legacy":
{},"subType":"square320","crop_name":"square320"},{"rank":0,"subtype":"square640","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-square640-v2.jpg","height":640,"width":640,"legacy":
{},"subType":"square640","crop_name":"square640"},{"rank":0,"subtype":"superJumbo","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-
superJumbo.jpg","height":1535,"width":2048,"legacy":{},"subType":"superJumbo","crop_name":"superJumbo"},{"rank":0,"subtype":"thumbLarge","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
thumbLarge.jpg","height":150,"width":150,"legacy":{},"subType":"thumbLarge","crop_name":"thumbLarge"},
{"rank":0,"subtype":"tmagArticle","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-tmagArticle.jpg","height":444,"width":592,"legacy":
{},"subType":"tmagArticle","crop_name":"tmagArticle"},{"rank":0,"subtype":"tmagSF","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/merlin_153570822_6863270c-fd2e-4786-a7c2-743068a88ecf-
tmagSF.jpg","height":271,"width":362,"legacy":{},"subType":"tmagSF","crop_name":"tmagSF"},{"rank":0,"subtype":"verticalTwoByThree735","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
verticalTwoByThree735.jpg","height":1102,"width":735,"legacy":{},"subType":"verticalTwoByThree735","crop_name":"verticalTwoByThree735"},
{"rank":0,"subtype":"videoFifteenBySeven1305","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-videoFifteenBySeven1305.jpg","height":609,"width":1305,"legacy":
{},"subType":"videoFifteenBySeven1305","crop_name":"videoFifteenBySeven1305"},{"rank":0,"subtype":"videoHpMedium","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
videoHpMedium.jpg","height":211,"width":375,"legacy":{},"subType":"videoHpMedium","crop_name":"videoHpMedium"},{"rank":0,"subtype":"videoLarge","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
videoLarge.jpg","height":507,"width":768,"legacy":{},"subType":"videoLarge","crop_name":"videoLarge"},
{"rank":0,"subtype":"videoSixteenByNine1050","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-videoSixteenByNine1050.jpg","height":591,"width":1050,"legacy":
{},"subType":"videoSixteenByNine1050","crop_name":"videoSixteenByNine1050"},{"rank":0,"subtype":"videoSixteenByNine150","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
videoSixteenByNine150.jpg","height":84,"width":150,"legacy":{},"subType":"videoSixteenByNine150","crop_name":"videoSixteenByNine150"},
{"rank":0,"subtype":"videoSixteenByNine225","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-videoSixteenByNine225.jpg","height":126,"width":225,"legacy":
{},"subType":"videoSixteenByNine225","crop_name":"videoSixteenByNine225"},{"rank":0,"subtype":"videoSixteenByNine310","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
videoSixteenByNine310.jpg","height":174,"width":310,"legacy":{},"subType":"videoSixteenByNine310","crop_name":"videoSixteenByNine310"},
{"rank":0,"subtype":"videoSixteenByNine390","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-videoSixteenByNine390.jpg","height":219,"width":390,"legacy":
{},"subType":"videoSixteenByNine390","crop_name":"videoSixteenByNine390"},{"rank":0,"subtype":"videoSixteenByNine480","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
videoSixteenByNine480.jpg","height":270,"width":480,"legacy":{},"subType":"videoSixteenByNine480","crop_name":"videoSixteenByNine480"},
{"rank":0,"subtype":"videoSixteenByNine495","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-videoSixteenByNine495.jpg","height":278,"width":495,"legacy":
{},"subType":"videoSixteenByNine495","crop_name":"videoSixteenByNine495"},{"rank":0,"subtype":"videoSixteenByNine540","caption":null,"credit":null,"type":"image","url":"images/2019/04/15/opinion/15Druckerman/15Druckerman-
videoSixteenByNine540.jpg","height":304,"width":540,"legacy":{},"subType":"videoSixteenByNine540","crop_name":"videoSixteenByNine540"},

Lets see how this works in the shell

First, I've gone to my class directory to make a scraper file directory



Now I have a shell command that uses curl to call the API for 'Notre Dame Cathedral'



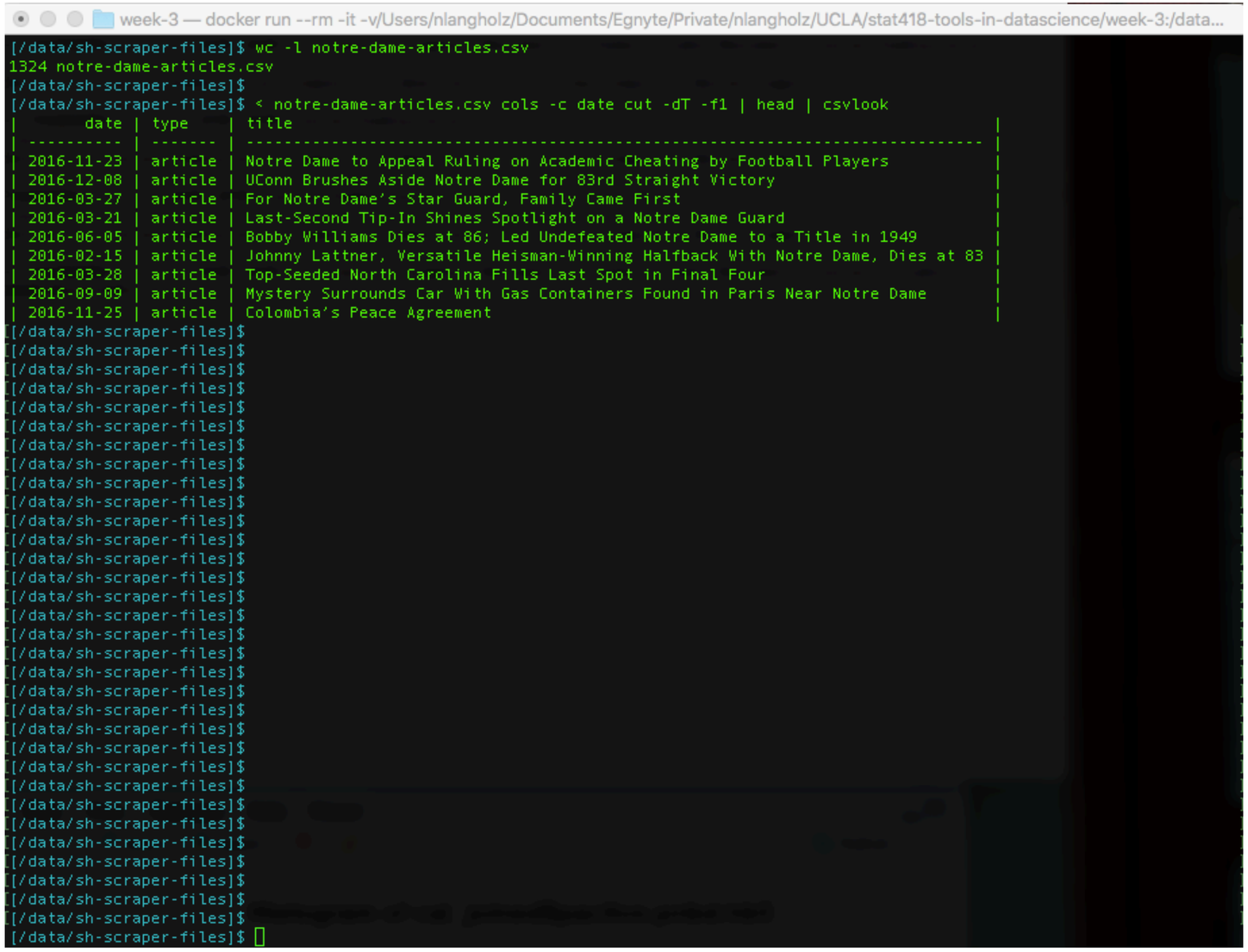
There is a lot going on in this command. I have delayed each request by 7 seconds, and am searching for articles from 2014 to 2019 with 99 pages of results

Oops

How long is this going to take?

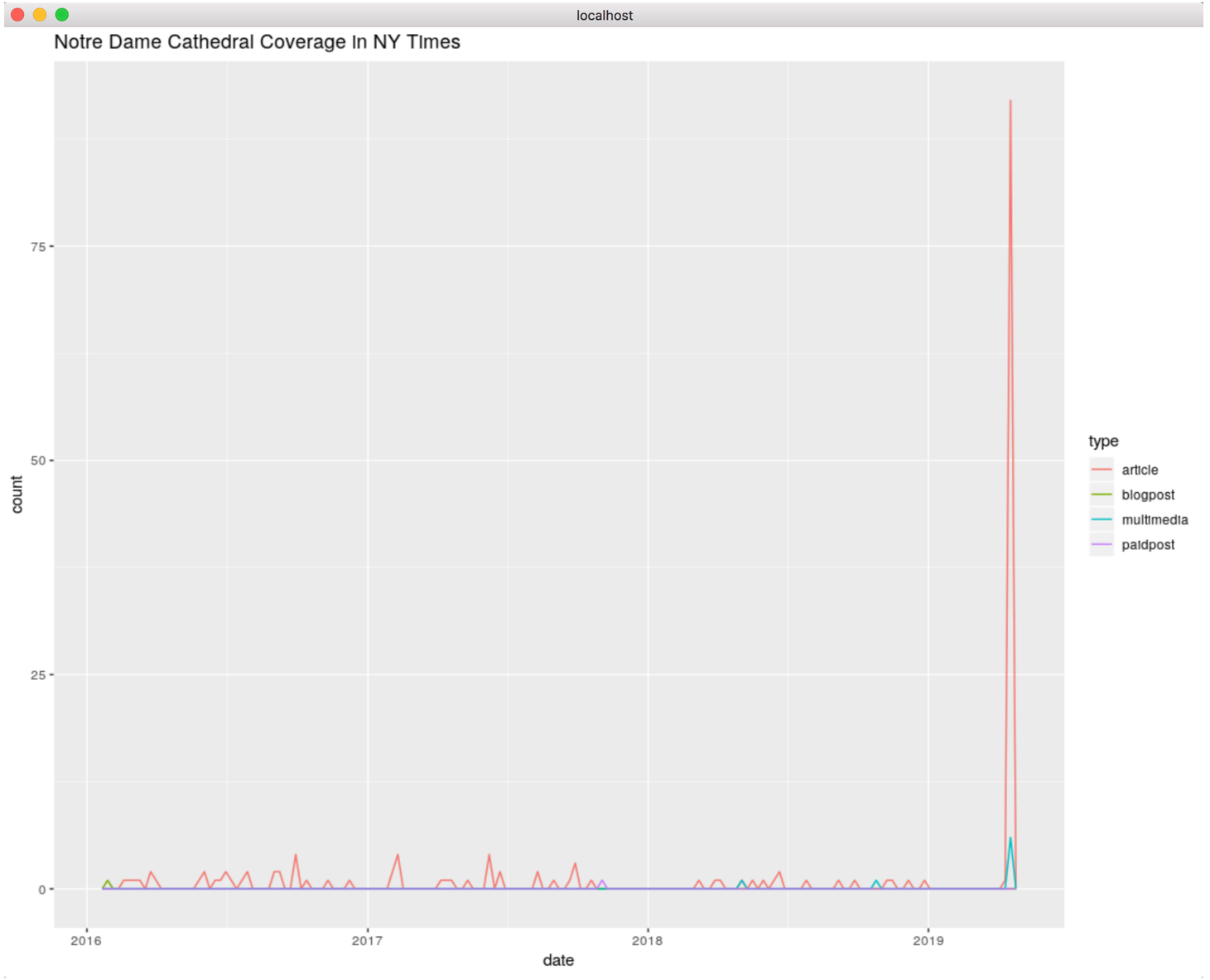
The results

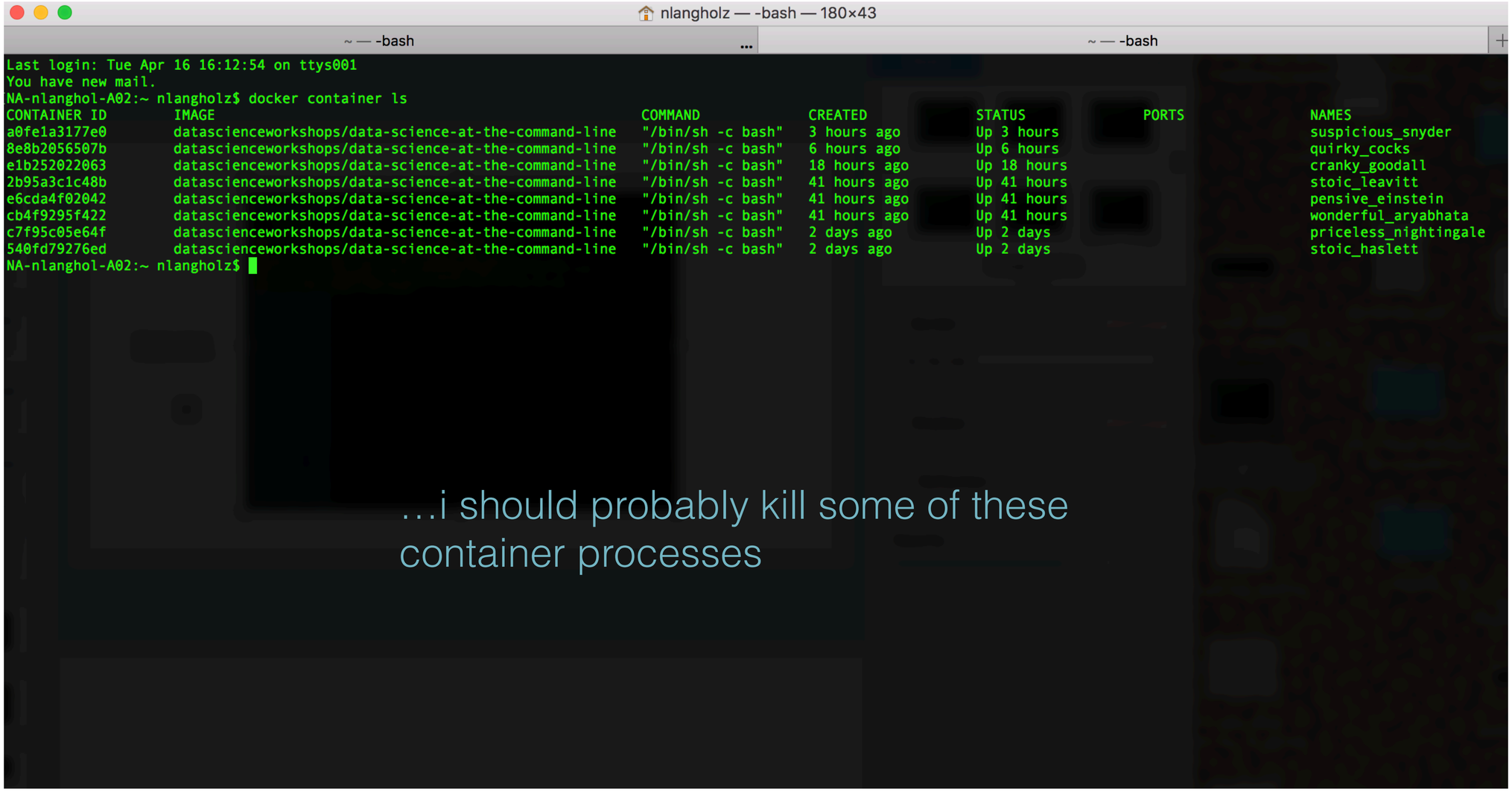
A .sh script exists in the GitHub repo to recreate this web scrape



The results

A .sh script exists in the GitHub repo to recreate this web scrape




nlangholz — -bash — 180×43
~ — -bash
~ — -bash
Last login: Tue Apr 16 16:12:54 on ttys001
You have new mail.
NA-nlanghol-A02:~ nlangholz$ docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
a0fe1a3177e0 datascienceworkshops/data-science-at-the-command-line "/bin/sh -c bash" 3 hours ago Up 3 hours suspicious_snyder
8e8b2056507b datascienceworkshops/data-science-at-the-command-line "/bin/sh -c bash" 6 hours ago Up 6 hours quirky_cocks
e1b252022063 datascienceworkshops/data-science-at-the-command-line "/bin/sh -c bash" 18 hours ago Up 18 hours cranky_goodall
2b95a3c1c48b datascienceworkshops/data-science-at-the-command-line "/bin/sh -c bash" 41 hours ago Up 41 hours stoic_leavitt
e6cda4f02042 datascienceworkshops/data-science-at-the-command-line "/bin/sh -c bash" 41 hours ago Up 41 hours pensive_einstein
cb4f9295f422 datascienceworkshops/data-science-at-the-command-line "/bin/sh -c bash" 41 hours ago Up 41 hours wonderful_aryabhata
c7f95c05e64f datascienceworkshops/data-science-at-the-command-line "/bin/sh -c bash" 2 days ago Up 2 days priceless_nightingale
540fd79276ed datascienceworkshops/data-science-at-the-command-line "/bin/sh -c bash" 2 days ago Up 2 days stoic_haslett
NA-nlanghol-A02:~ nlangholz$
…i should probably kill some of these container processes