

# 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/](https://master.stat.ucla.edu/admissions/)

Information is found here: <https://master.stat.ucla.edu/> And here: [https://master.stat.ucla.edu/  
faq/](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

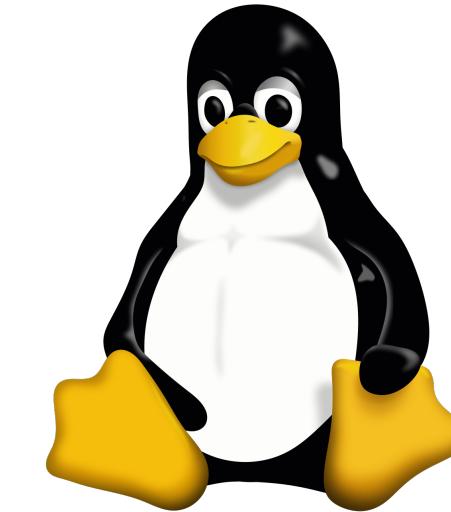
most devices that contain a computer of some kind will have an OS, they tend to emerge when the appliance will have to deal with new applications, complex user-input and possibly changing requirements on its function



# 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 them on areas

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

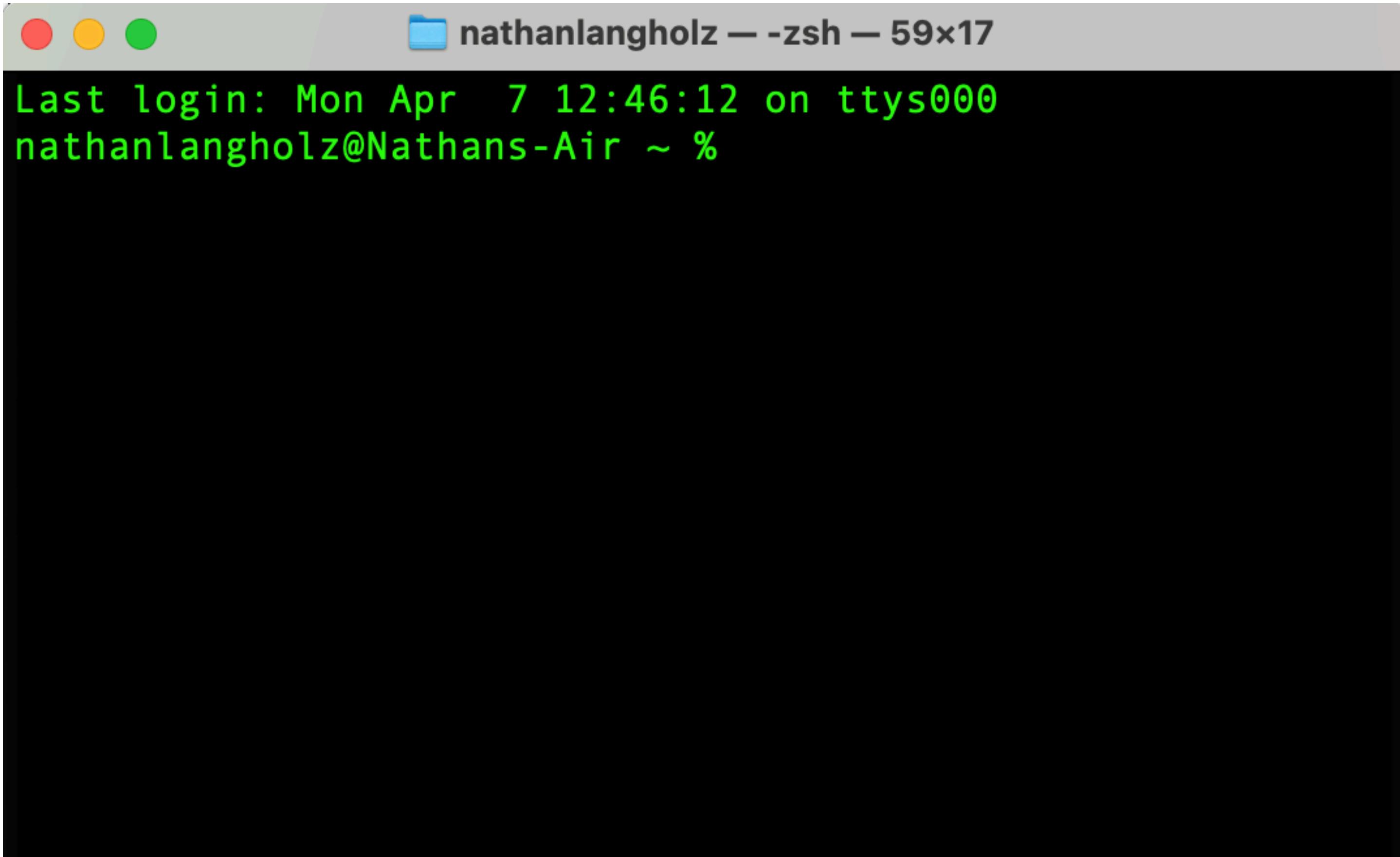
the Unix shell is a user interface to the kernel (keep in mind that Unix was designed for computer scientists and the interface is not optimized for novices)

## 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



```
Last login: Mon Apr 7 12:46:12 on ttys000
nathanlangholz@Nathans-Air ~ %
```

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

Name of command

How hard the computer is thinking about it

Process ID

```
nathanlangholz — top — 94x41
Processes: 673 total, 3 running, 670 sleeping, 3054 threads
Load Avg: 3.58, 3.27, 3.37 CPU usage: 14.31% user, 7.4% sys, 78.63% idle
SharedLibs: 423M resident, 70M data, 16M linkedit.
MemRegions: 0 total, 0B resident, 1360K private, 747M shared.
PhysMem: 8079M used (2823M wired, 1238M compressor), 240M unused.
VM: 39T vsize, 5047M framework vsize, 168007066(886) swapins, 179820842(0) swapouts.
Networks: packets: 33358385/38G in, 13958509/13G out.
Disks: 35008277/1131G read, 14719577/855G written.

PID COMMAND %CPU TIME #TH #WQ #PORT MEM PURG CMPRS PGRP PPID STATE
155 WindowServer 23.2 13:57:40 14 6 4853 898M- 11M 219M- 155 1 sleeping
194 coreaudiod 10.2 51:11.67 12 4 2197 21M- 0B 11M- 194 1 sleeping
41829 top 9.2 00:09.40 1/1 0 31 9032K 0B 1036K 41829 41793 running
41677 com.apple.We 7.8 05:49.60 18/1 8 124 1487M+ 9676K 399M- 41677 1 running
0 kernel_task 4.8 10:06:26 261/4 0 0 588M- 0B 0B 0 0 running
11099 Google Drive 4.6 15:54:57 9 1 140 37M 0B 19M 1552 11095 sleeping
22553 com.apple.We 4.3 19:37.14 38 6 450- 178M- 6784K 149M 22553 1 sleeping
11103 Google Drive 3.0 09:13:04 12 1 332 27M 0B 24M 1552 11095 sleeping
22429 com.apple.Ap 2.2 17:18.66 3 2 300 776K 0B 272K 22429 1 sleeping
29078 com.apple.We 1.4 03:05.32 9 1 118- 253M- 0B 188M- 29078 1 sleeping
1474 Spotlight 1.4 03:02.20 5 2 503+ 88M 0B 62M- 1474 1 sleeping
22493 Safari 1.1 38:14.86 17 9 3001- 357M- 28K 299M- 22493 1 sleeping
40548 com.apple.We 1.0 03:46.15 11 4 108 503M 0B 435M- 40548 1 sleeping
1468 Terminal 1.0 01:02.29 11 4 540 68M+ 1792K 23M- 1468 1 sleeping
1558 Google Chrom 0.8 99:45.84 12 1 139+ 64M+ 0B 40M- 1464 1464 sleeping
113 mds 0.8 83:50.47 8 5 611 39M 0B 31M- 113 1 sleeping
22425 bluetoothd 0.8 32:33.17 11 5 617+ 9412K+ 20K 4740K- 22425 1 sleeping
22498 com.apple.We 0.7 48:21.17 10 5 191- 98M- 264K 38M- 22498 1 sleeping
5768 Google Chrom 0.6 47:45.51 14 1 401 379M+ 0B 310M- 1464 1464 sleeping
1464 Google Chrom 0.5 02:08:20 46 3 1746+ 231M 0B 166M- 1464 1 sleeping
41832 screencaptur 0.5 00:00.30 2 1 66 4160K+ 620K 0B 1470 1470 sleeping
566 sharingd 0.2 24:02.25 5 2 495+ 22M+ 0B 17M- 566 1 sleeping
86 logd 0.2 32:51.95 4 3 2363 22M 0B 27M- 86 1 sleeping
586 useractivity 0.2 01:00.43 4 3 114+ 2720K+ 0B 1376K- 586 1 sleeping
483 identityserv 0.2 11:55.14 6 2 826+ 21M+ 128K 18M- 483 1 sleeping
32877 audioanalyti 0.1 01:05.47 4 3 56 1724K+ 0B 448K- 32877 1 sleeping
166 lsd 0.1 04:20.25 2 1 140+ 6136K+ 0B 5304K- 166 1 sleeping
91 fsevents 0.1 32:59.10 14 1 199 3664K- 0B 1404K 91 1 sleeping
160 runningboard 0.1 36:11.74 7 6 944 7636K 0B 1332K 160 1 sleeping
99 powerd 0.1 12:25.51 4 3 155+ 3784K+ 0B 1872K- 99 1 sleeping
29451 com.docker.b 0.1 40:03.69 120 1 437 90M 0B 47M 29430 29430 sleeping
```

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)
<b>NAME</b> <code>ksh, ksh93</code> - KornShell, a command and programming language		
<b>SYNOPSIS</b> <code>ksh [ <del>a</del>bcefhikmnoprstuvwxyzBCDP ] [ -R file ] [ <del>o</del> option ] ... [ - ] [ arg ... ] rksh [ <del>a</del>bcefhikmnoprstuvwxyzBCD ] [ -R file ] [ <del>o</del> option ] ... [ - ] [ arg ... ]</code>		
<b>DESCRIPTION</b> <code>Ksh</code> is a command and programming language that executes commands read from a terminal or a file. <code>Rksh</code> is a restricted version of the command interpreter <code>ksh</code> ; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. <code>Rpksh</code> is a profile shell version of the command interpreter <code>ksh</code> ; it is used to execute commands with the attributes specified by the user's profiles (see <code>pexec(1)</code> ). See <code>Invocation</code> below for the meaning of arguments to the shell.		
TCSH(1)	General Commands Manual	TCSH(1)
<b>NAME</b> <code>tcsh</code> - C shell with file name completion and command line editing		
<b>SYNOPSIS</b> <code>tcsh [ -bcdefFimnqstvVxX ] [ -Dname[=value] ] [arg ...] tcsh -l</code>		
<b>DESCRIPTION</b> <code>tcsh</code> is an enhanced but completely compatible version of the Berkeley UNIX C shell, <code>csh(1)</code> . 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 <code>The command-line editor</code> ), programmable word completion (see <code>Completion and listing</code> ), spelling correction (see <code>Spelling correction</code> ), a history mechanism (see <code>History substitution</code> ), job control (see <code>Jobs</code> ) and a C-like syntax. The <code>NEW FEATURES</code> section describes major enhancements of <code>tcsh</code> over <code>csh(1)</code> . Throughout this manual, features of <code>tcsh</code> not found in most <code>csh(1)</code> implementations (specifically, the 4.4BSD <code>csh</code> ) are labeled with '(+)', and features which are present in <code>csh(1)</code> but not usually documented are labeled with '(u)'.		
BASH(1)	General Commands Manual	BASH(1)
<b>NAME</b> <code>bash</code> - GNU Bourne-Again SHell		
<b>SYNOPSIS</b> <code>bash [options] [command_string   file]</code>		
<b>COPYRIGHT</b> <code>Bash</code> is Copyright (C) 1989-2013 by the Free Software Foundation, Inc.		
<b>DESCRIPTION</b> <code>Bash</code> is an <code>sh</code> -compatible command language interpreter that executes commands read from the standard input or from a file. <code>Bash</code> also incorporates useful features from the <code>Korn</code> and <code>C</code> shells ( <code>ksh</code> and <code>csh</code> ).  <code>Bash</code> is intended to be a conformant implementation of the Shell and Utilities portion of the IEEE POSIX specification (IEEE Standard 1003.1). <code>Bash</code> can be configured to be POSIX-conformant by default.		
ZSH(1)	General Commands Manual	ZSH(1)
<b>NAME</b> <code>zsh</code> - the Z shell		
<b>OVERVIEW</b> Because <code>zsh</code> contains many features, the <code>zsh</code> manual has been split into a number of sections:		
<code>zsh</code> Zsh overview (this section) <code>zshroadmap</code> Informal introduction to the manual <code>zshmisc</code> Anything not fitting into the other sections <code>zshepn</code> Zsh command and parameter expansion <code>zshparam</code> Zsh parameters <code>zshoptions</code> Zsh options <code>zshbuiltins</code> Zsh built-in functions <code>zshzle</code> Zsh command line editing <code>zshcomwid</code> Zsh completion widgets <code>zshcompsys</code> Zsh completion system <code>zshcomctl</code> Zsh completion control <code>zshmodules</code> Zsh loadable modules <code>zshcalsys</code> Zsh built-in calendar functions		

## 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 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, creator of sh, 2005 (source wikipedia)

And while we're 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 files

`pwd`, `ls`, `cd`

`mkdir`, `rmdir`

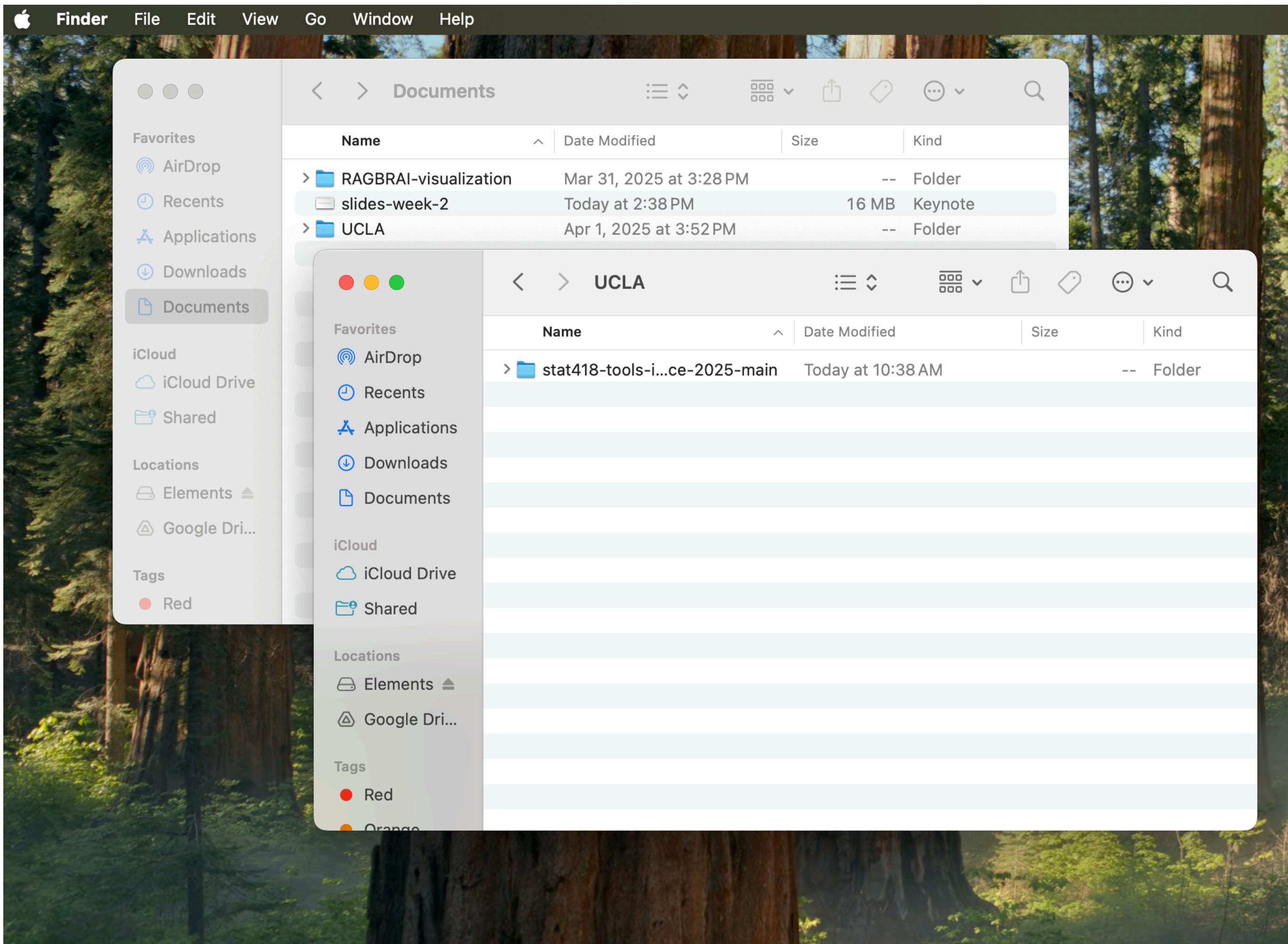
`cp`, `mv`, `rm`

The screenshot shows a terminal window titled "Documents --zsh-- 82x31". The terminal output is as follows:

```
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
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 %
```

Annotations with arrows point to specific parts of the terminal output:

- A line labeled "Present working directory" points to the line "pwd" in the history.
- A line labeled "Change directory" points to the line "cd Documents".
- A line labeled "List" points to the line "ls".
- A line labeled "Long list" points to the line "ls -l".



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

```
[nathanlangholz@MacBookAir stat418-tools-in-datascience-2025 %]
[nathanlangholz@MacBookAir stat418-tools-in-datascience-2025 %]
[nathanlangholz@MacBookAir stat418-tools-in-datascience-2025 %] ls -al
[total 192
drwxr-xr-x 13 nathanlangholz staff 416 Apr 8 19:46 .
drwxr-xr-x  4 nathanlangholz staff 128 Apr 8 15:05 ..
drwxr-xr-x  5 nathanlangholz staff 160 Apr 8 19:40 .Rproj.user
drwxr-xr-x 16 nathanlangholz staff 512 Apr 8 19:50 .git
-rw-r--r--  1 nathanlangholz staff   40 Apr 8 15:05 .gitignore
-rw-r--r--  1 nathanlangholz staff 1070 Apr 8 15:05 LICENSE
-rw-r--r--  1 nathanlangholz staff 4483 Apr 8 15:05 README.md
drwxr-xr-x  3 nathanlangholz staff   96 Apr 8 15:05 final-project
-rw-r--r--  1 nathanlangholz staff 75128 Apr 8 19:46 iris-plot-nlucla2.pdf
-rw-r--r--  1 nathanlangholz staff  253 Apr 8 19:40 stat418-tools-in-datasci
ence-2025.Rproj
drwxr-xr-x  4 nathanlangholz staff 128 Apr 8 15:05 week-1
drwxr-xr-x  5 nathanlangholz staff 160 Apr 8 15:24 week-2
drwxr-xr-x  4 nathanlangholz staff 128 Apr 8 15:05 week-3
```

Read, write, execute permissions

The file's size in bytes

Shorthand for your present working directory

Shorthand for the directory one level above

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 of 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

The type of file

```
drwxr-xr-x 12 nlangholz staff  
drwx-----+ 15 nlangholz staff  
-rw-r--r-- 1 nlangholz staff  
drwxr-xr-x  5 nlangholz staff  
drwxr-xr-x 16 nlangholz staff  
-rw-r--r-- 1 nlangholz staff  
-rw-r--r-- 1 nlangholz staff  
-rw-r--r-- 1 nlangholz staff  
-rw-r--r-- 1 nlangholz staff  
drwxr-xr-x  4 nlangholz staff  
drwxr-xr-x  4 nlangholz staff  
drwxr-xr-x  3 nlangholz staff
```

What you can do to  
the file

What the owning  
group can do

What others can do

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 jeroenjanssens/data-science-at-the-command-line
```

## Data Science at the Command Line

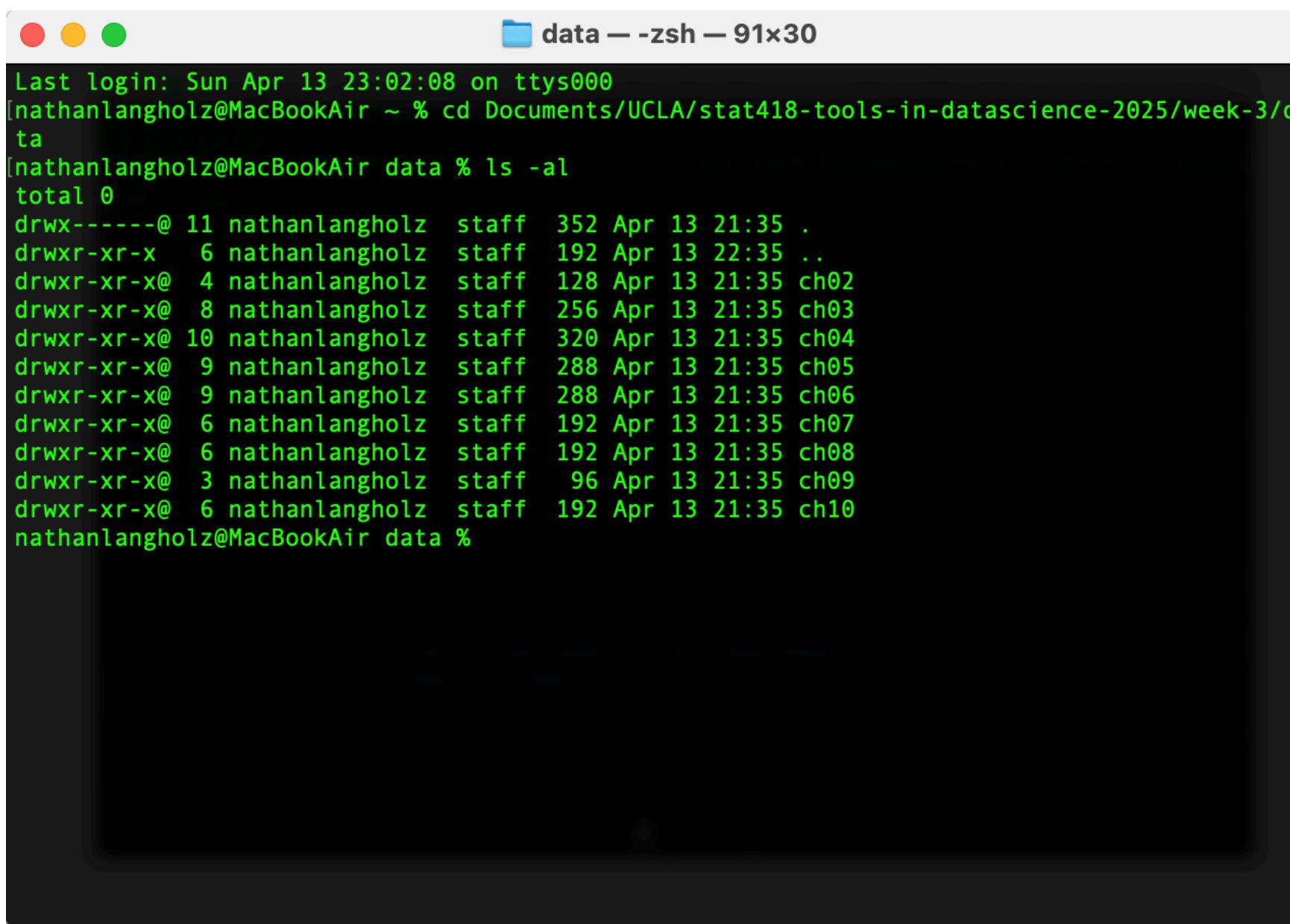
To run the image

```
docker run --rm -it jeroenjanssens/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

Before mounting to local directory go to the directory where you have downloaded the data from the book site.



```
Last login: Sun Apr 13 23:02:08 on ttys000
[nathanlangholz@MacBookAir ~ % cd Documents/UCLA/stat418-tools-in-datasience-2025/week-3/da
ta
[nathanlangholz@MacBookAir data % ls -al
total 0
drwx-----@ 11 nathanlangholz  staff  352 Apr 13 21:35 .
drwxr-xr-x  6 nathanlangholz  staff  192 Apr 13 22:35 ..
drwxr-xr-x@  4 nathanlangholz  staff  128 Apr 13 21:35 ch02
drwxr-xr-x@  8 nathanlangholz  staff  256 Apr 13 21:35 ch03
drwxr-xr-x@ 10 nathanlangholz  staff  320 Apr 13 21:35 ch04
drwxr-xr-x@  9 nathanlangholz  staff  288 Apr 13 21:35 ch05
drwxr-xr-x@  9 nathanlangholz  staff  288 Apr 13 21:35 ch06
drwxr-xr-x@  6 nathanlangholz  staff  192 Apr 13 21:35 ch07
drwxr-xr-x@  6 nathanlangholz  staff  192 Apr 13 21:35 ch08
drwxr-xr-x@  3 nathanlangholz  staff   96 Apr 13 21:35 ch09
drwxr-xr-x@  6 nathanlangholz  staff  192 Apr 13 21:35 ch10
nathanlangholz@MacBookAir data %
```

Download and unzip from here  
<https://www.datascienceatthecommandline.com/2e/data.zip>

Now to mount our local directories use

on a Mac

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

Or

```
docker run --rm -it -v "$(pwd)":/data jeroenjanssens/data-science-at-the-command-line
```

in Windows command line

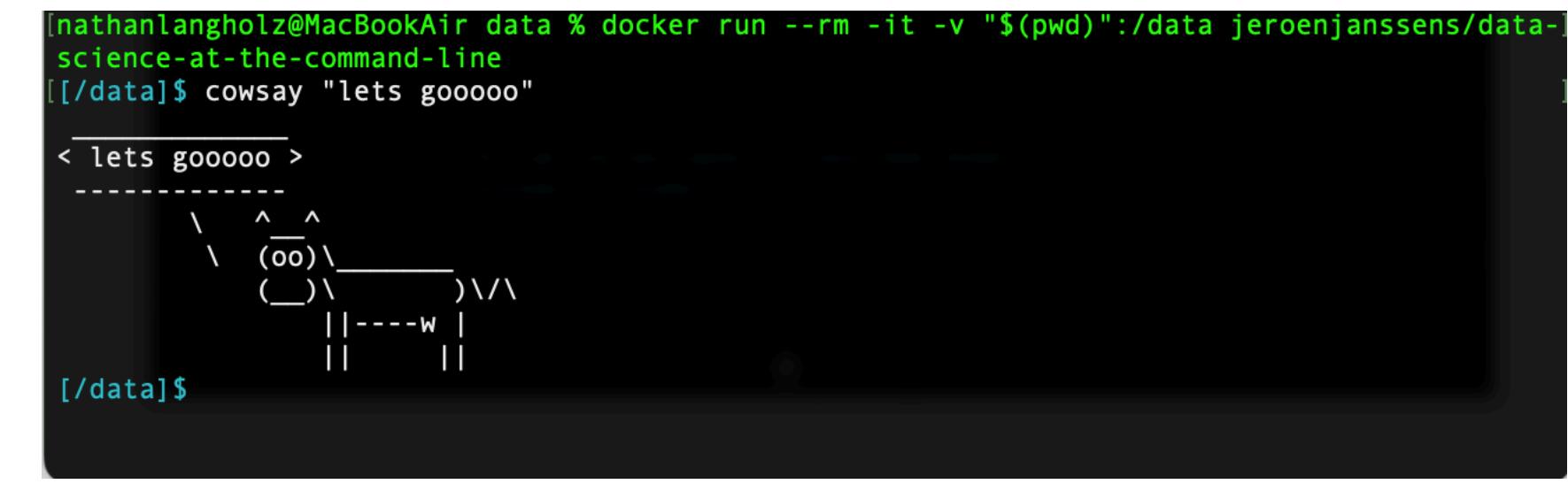
```
C:\> docker run --rm -it -v "%cd%":/data jeroenjanssens/data-science-at-the-command-line
```

in Windows powershell

```
PS C:\> docker run --rm -it -v ${PWD}:data jeroenjanssens/data-science-at-the-command-line
```

To make sure everything is running correctly, type

```
cowsay "ready to go!"
```



```
[nathanlangholz@MacBookAir data % docker run --rm -it -v "$(pwd)":/data jeroenjanssens/data-science-at-the-command-line
[[/data]$ cowsay "lets gooooo"
< lets gooooo >
-----+
 \  ^__^
  (oo)\_____
   (__)\       )\/\
    ||----w |
     ||     ||
[[/data]$
```

A screenshot of a terminal window showing a cow ASCII art and text output. The terminal prompt is '[nathanlangholz@MacBookAir data % docker run --rm -it -v "\$(pwd)":/data jeroenjanssens/data-science-at-the-command-line'. The user then runs the command 'cowsay "lets gooooo"'. The terminal displays the cow ASCII art followed by the text 'lets gooooo'.

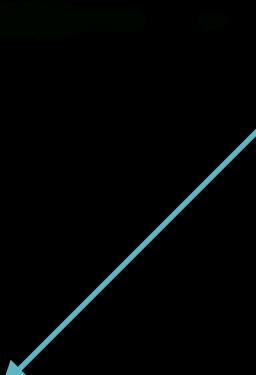
This environment we now have is a bash shell running on a Linux operating system.

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

```
● ● ● data — docker run --rm -it -v ~/Documents/UCLA/stat418-tools-in-datas...  
drwxr-xr-x@ 4 nathanlangholz staff 128 Apr 13 21:35 ch02  
drwxr-xr-x@ 8 nathanlangholz staff 256 Apr 13 21:35 ch03  
drwxr-xr-x@ 10 nathanlangholz staff 320 Apr 13 21:35 ch04  
drwxr-xr-x@ 9 nathanlangholz staff 288 Apr 13 21:35 ch05  
drwxr-xr-x@ 9 nathanlangholz staff 288 Apr 13 21:35 ch06  
drwxr-xr-x@ 6 nathanlangholz staff 192 Apr 13 21:35 ch07  
drwxr-xr-x@ 6 nathanlangholz staff 192 Apr 13 21:35 ch08  
drwxr-xr-x@ 3 nathanlangholz staff 96 Apr 13 21:35 ch09  
drwxr-xr-x@ 6 nathanlangholz staff 192 Apr 13 21:35 ch10  
nathanlangholz@MacBookAir data % docker run --rm -it -v "$(pwd)":/data jeroenjanssens/data-science-at-the-command-line  
[/data]$ cowsay "lets gooooo"  
  
< lets gooooo >  
-----  
 \ ^ ^ /  
  \ (oo)\_____  
   (____)\ )\/\ /  
      ||----w |  
      ||     ||  
-----  
[~/data]$  
[~/data]$  
[~/data]$  
[~/data]$  
[~/data]$  
[~/data]$  
[~/data]$  
[~/data]$  
[~/data]$ echo 'Hello world' | wc  
      1      2     12  
[~/data]$
```

Redirecting output with “|”

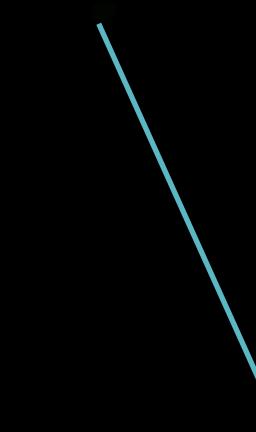
takes output from one command and submits it as input to the next command



```
● ● ● data — docker run --rm -it -v ~/Documents/UCLA/stat418-tools-in-datasience-2025...
drwxr-xr-x@ 9 nathanlangholz staff 288 Apr 13 21:35 ch06
drwxr-xr-x@ 6 nathanlangholz staff 192 Apr 13 21:35 ch07
drwxr-xr-x@ 6 nathanlangholz staff 192 Apr 13 21:35 ch08
drwxr-xr-x@ 3 nathanlangholz staff 96 Apr 13 21:35 ch09
drwxr-xr-x@ 6 nathanlangholz staff 192 Apr 13 21:35 ch10
[nathanlangholz@MacBookAir data % docker run --rm -it -v "$(pwd)":/data jeroenjanssens/data-science-at-the-command-line
[/data]$ cowsay "lets gooooo"
< lets gooooo >
-----
      \  ^ ^
       \  (oo)\_____
          (__)\       )\/\
              ||----w |
              ||     |
[/data]$
[/data]$ echo 'Hello world' | wc
      1      2      12
[/data]$ head -n 3 ch02/movies.txt ←
Matrix
Star Wars
Home Alone
[/data]$
```

Listing the first 3 movies  
from the file movies.txt

```
● ● ● data — docker run --rm -it -v ~/Documents/UCLA/stat418-tools-in-datascience-2025...
[[/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$
```



```
[[/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>

```
● ○ ● Documents — docker run --rm -it -v ~/Documents:/data jeroenjanssens/data-science-at-the-command-line — 132x41
nathanlangholz@MacBookAir Documents % docker run --rm -it -v "$(pwd)":/data jeroenjanssens/data-science-at-the-command-line
[/data]$ ls -al
total 1884
drwx----- 7 root root 224 Apr 14 04:37 .
drwxr-xr-x 1 root root 4096 Apr 14 22:55 ..
-rw-r--r-- 1 root root 10244 Apr 14 21:22 .DS_Store
-rw-r--r-- 1 root root 0 Jul 27 2023 .localized
-rw-r--r-- 1 root root 1909170 Apr 12 23:49 R-project-github-sync.pdf
drwxr-xr-x 9 root root 288 Mar 31 22:28 RAGBRAI-visualization
drwxr-xr-x 4 root root 128 Apr 14 05:35 UCLA
[/data]$ cd UCLA/stat418-tools-in-datasience-2025/week-3
[/data/UCLA/stat418-tools-in-datasience-2025/week-3]$ ls
R README.md data sh-scraper-file slides-week-3.pdf
[/data/UCLA/stat418-tools-in-datasience-2025/week-3]$ wc data/ch03/finn.txt
12346 114126 622513 data/ch03/finn.txt
[/data/UCLA/stat418-tools-in-datasience-2025/week-3]$ head data/ch03/finn.txt
The Project Gutenberg eBook of Adventures of Huckleberry Finn

This ebook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever. You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this ebook or online
at www.gutenberg.org. If you are not located in the United States,
you will have to check the laws of the country where you are located
before using this eBook.

[/data/UCLA/stat418-tools-in-datasience-2025/week-3]$ tail data/ch03/finn.txt
Most people start at our website which has the main PG search
facility: www.gutenberg.org.

This website includes information about Project Gutenberg™,
including how to make donations to the Project Gutenberg Literary
Archive Foundation, how to help produce our new eBooks, and how to
subscribe to our email newsletter to hear about new eBooks.

[/data/UCLA/stat418-tools-in-datasience-2025/week-3]$
[/data/UCLA/stat418-tools-in-datasience-2025/week-3]$
[/data/UCLA/stat418-tools-in-datasience-2025/week-3]$
[/data/UCLA/stat418-tools-in-datasience-2025/week-3]$
```

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

Finn.txt doesn't exist in the data folder, this will be added in slide 49, but can try on any of the other text files in the directory

```
data — docker run --rm -it -v ~/Documents/UCLA/stat418-tools-in-datascience-2025/w...
[/data]$ head ch05/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
[/data]$
[[/data]$
[[/data]$ cut -d"," -f1 ch05/iris.csv | head
[sepal_length
5.1
4.9
4.7
4.6
5.0
5.4
4.6
5.0
4.4
[[/data]$
```

Irises again 🔥

Cut by delimiter “,” and take the first field

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?!





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/iris.csv > species.txt
```

# 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`

```
data — docker run --rm -it -v ~/Documents/UCLA/stat418-tools-in-datascience-2025/w...
[[/data]$ man cat | head -n 20
CAT(1)                               User Commands                               CAT(1)

NAME
    cat - concatenate files and print on the standard output

SYNOPSIS
    cat [OPTION]... [FILE]...

DESCRIPTION
    Concatenate FILE(s) to standard output.

    With no FILE, or when FILE is -, read standard input.

    -A, --show-all
        equivalent to -vET

    -b, --number-nonblank
        number nonempty output lines, overrides -n

[[/data]$
```

```
● ○ ● ⚡ data — docker run --rm -it -v ~/Documents/UCLA/stat418-tools-in-datasience-2025/w...
[[/data]$ help cd | head -n 30
cd: cd [-L|[-P [-e]] [-@]] [dir]
    Change the shell working directory.

    Change the current directory to DIR.  The default DIR is the value of the
    HOME shell variable.

    The variable CDPATH defines the search path for the directory containing
    DIR.  Alternative directory names in CDPATH are separated by a colon (:).
    A null directory name is the same as the current directory.  If DIR begins
    with a slash (/), then CDPATH is not used.

    If the directory is not found, and the shell option `cable_vars' is set,
    the word is assumed to be a variable name.  If that variable has a value,
    its value is used for DIR.

Options:
-L      force symbolic links to be followed: resolve symbolic links in
DIR after processing instances of `...'
-P      use the physical directory structure without following symbolic
links: resolve symbolic links in DIR before processing instances
of `...'
-e      if the -P option is supplied, and the current working directory
cannot be determined successfully, exit with a non-zero status
-@      on systems that support it, present a file with extended attributes
as a directory containing the file attributes

The default is to follow symbolic links, as if '-L' were specified.
`...' is processed by removing the immediately previous pathname component
back to a slash or the beginning of DIR.

[/data]$
```

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

## 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 argument

```
$ curl -s https://www.gutenberg.org/cache/epub/76/pg76.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

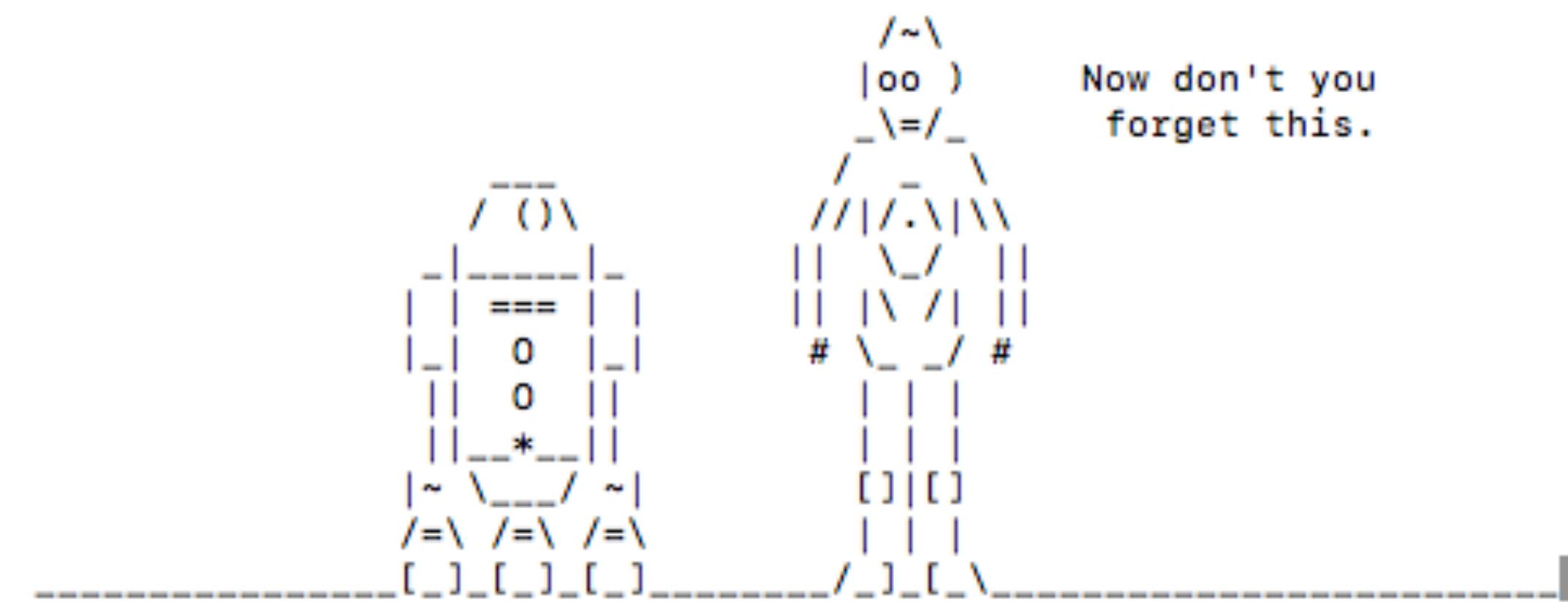
## Saving a URL

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

```
$ curl -s https://www.gutenberg.org/cache/epub/76/pg76.txt > data/ch03/  
finn.txt
```

Don't need to specify **-s** in this case

# Star Wars: Fun, but... useless?



try

```
$ nc towel.blinkenlights.nl 23 [Unfortunately this doesn't work anymore]
```

It does exist at <https://telehack.com> and run starwars, but not the same as doing it directly from cl

Data Acquisition (...still in the shell)

# Accessing the New York Times

The New York Times is one of the most well-known, reputable, widely circulated news outlets worldwide. They 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.

Monday, April 14, 2025 [Today's Paper](#)

# The New York Times

Nasdaq -0.05% +

U.S. [World](#) [Business](#) [Arts](#) [Lifestyle](#) [Opinion](#) | [Audio](#) [Games](#) [Cooking](#) [Wirecutter](#) [The Athletic](#)

---

**Inside Trump's Pressure Campaign on Universities**

The opaque process, part of a strategy by President Trump and conservatives to realign the liberal tilt of elite universities, has upended higher education.

9 MIN READ



Sophie Park for The New York Times

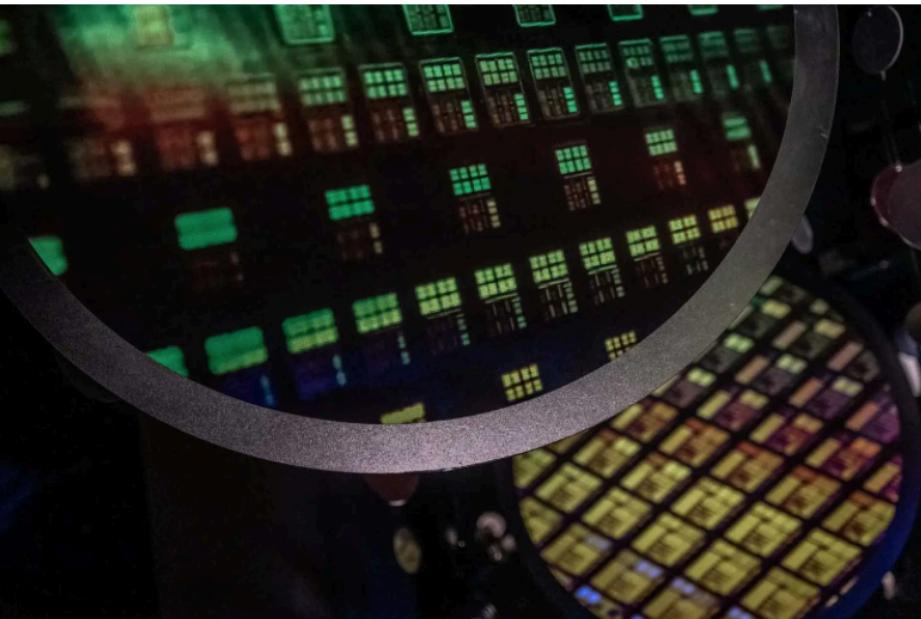
---

[Tariffs](#) [What to Know](#) [Fact Check](#) [Timeline](#) [Inside Trump's Reversal](#)

**Trump Signals Semiconductor Tariffs Are Coming 'Very Soon'**

President Trump has suggested that he will move forward with a national security investigation this week that is likely to result in tariffs on chips.

4 MIN READ



Lam Yik Fei for The New York Times

---

**Europe Fears U.S. Tariffs Will Set Off a Flood of Cheap Goods From China**

President Trump's tariffs on China could lead to a hazardous scenario for Europe: artificially cheap products that could undermine local industries.

5 MIN READ

---

**China Halts Critical Exports as Trade War Intensifies**

6 MIN READ

**When Elected Leaders Pursue Risky Policies, What Can Stop Them?**

6 MIN READ

**Stocks Jump After More Tariff Whiplash**

3 MIN READ

---

**Opinion**

**James Carville: How to Turn Trump's Economic Chaos Against Him**

5 MIN READ

---

**MICHELLE COTTLE**

**Matt Gaetz Hit the Skids. These Days, It's Not Disqualifying.**

5 MIN READ



Justin T. Gellerson for The New York Times

## She's Young, Trump-Friendly, and Has a White House Press Pass

Natalie Winters, a protégée of Stephen Bannon, belongs to a newly high-profile contingent of West Wing reporters.

6 MIN READ



**What a Wirecutter Editor Learned After Joining the Cult of Quince**

FROM WIRECUTTER



**'S.N.L.': Laughing All the Way to Financial Chaos**

4 MIN READ

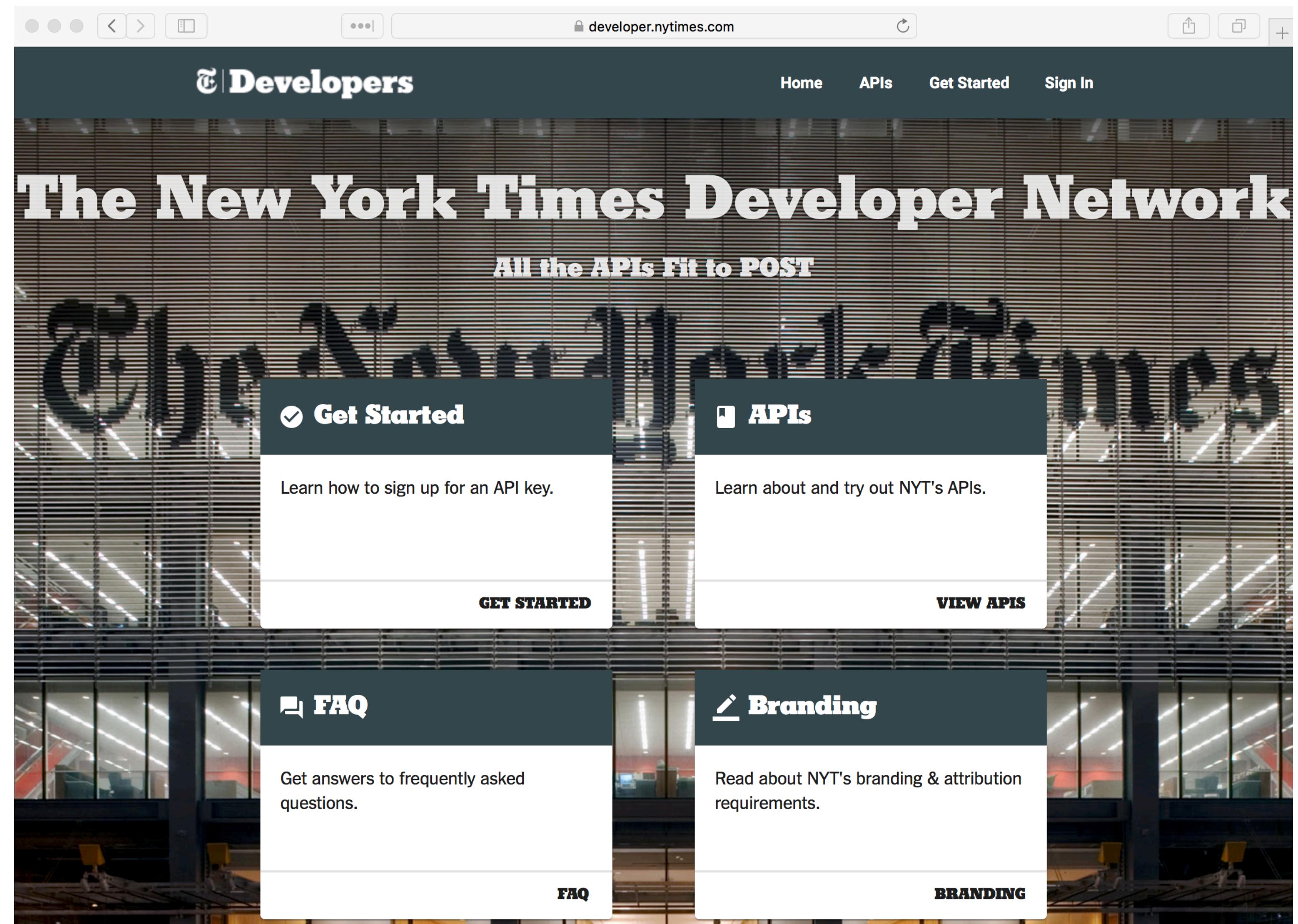


With the ongoing news of tariffs since President Trump came into office, the NYT front page is full of articles about them on a daily basis..

I was curious as to how many articles had been posted in the previous five years prior to his election and inauguration.

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

The screenshot shows a web browser window for developer.nytimes.com. The title bar says "developer.nytimes.com". The main header has "T|Developers" on the left and navigation links for "Home", "APIs", "Get Started", and an account dropdown on the right. Below the header, it says "New App". On the left, there's a sidebar with "Overview" and "APIs" sections. In the "Overview" section, there are fields for "App Name \*" (containing "search-articles") and "Description". A "CLEAR" button is next to the name field, and a "CREATE" button is on the right. In the "APIs" section, there are six cards arranged in two rows of three. The first row includes "Movie Reviews API" (camera icon), "Most Popular API" (yellow star icon), and "Times Wire API" (NYT logo icon). The second row includes "Article Search API" (magnifying glass icon), "Community API" (speech bubble icon), and "Books API" (stack of books icon). Each card has a brief description and a toggle switch below it.

App Name \*

search-articles

Description

CLEAR CREATE

Overview

APIs

Movie Reviews API

Search for movie reviews.

Most Popular API

Get most emailed, shared, or viewed articles.

Times Wire API

Real-time feed of NYT article publishes.

Article Search API

Search for New York Times articles.

Community API

Get user comments. (DEPRECATED)

Books API

Get NYT Best Sellers Lists and lookup book reviews.

## 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, 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 reimaged. 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](mailto: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 commercial use case, please contact us at [nytg-sales@nytimes.com](mailto:nytg-sales@nytimes.com) or visit <https://nytlicensing.com/>. For Text & Data Mining options, please visit <https://nytlicensing.com/data-solutions/>.

### 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 or selling advertising for any service or application that includes New York Times content or data.
- Selling any application built with one of our APIs.
- Using our content inside an application that is paid or has a paid tier currently or in the future.

If you aren't sure whether your plans constitute "commercial purposes," please contact us at [nytg-sales@nytimes.com](mailto:nytg-sales@nytimes.com). For commercial use, please visit <https://nytlicensing.com/> and for Text & Data Mining, please visit <https://nytlicensing.com/data-solutions/>.

### 11. Is there an API call limit?

Yes, there are two rate limits per API: 500 requests per day and 5 requests per minute. You should sleep 12 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](mailto: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.

### 13. What happened to the Community API?

The Times is no longer providing reader comments for non-commercial use cases via the Developer Portal. We apologize for any inconvenience that this might cause. We expect to have details regarding a commercial offering in 2023 and for further information, please contact [nytg-sales@nytimes.com](mailto:nytg-sales@nytimes.com).

### 14. Can I make suggestions for future development?

Yes, please! You can email us at [code@nytimes.com](mailto:code@nytimes.com).

This is important  
to note

This is what it  
was in 2019



If you didn't see this in your previous version of the developer portal, please contact us at [code@nytimes.com](mailto:code@nytimes.com) for feedback.

### 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](mailto:code@nytimes.com).

The screenshot shows a web browser window for developer.nytimes.com. The URL bar contains 'developer.nytimes.com'. The page title is '{T} Developers' and the main navigation includes 'Home', 'APIs', 'Get Started', and a user account dropdown. The main content area is titled 'Article Search' and features a sidebar with 'ARTICLE SEARCH' (Overview), 'PATHS' (selected, showing '/articlesearch.json GET'), and 'COMPONENTS' (Schemas, Article, Byline, Headline, Image, Keyword, Multimedia). The main content area displays the 'GET /articlesearch.json' endpoint, which is described as 'Search for NYT articles by keywords and filters.' It includes an 'HTTP request' section with the URL <https://api.nytimes.com/svc/search/v2/articlesearch.json> and a 'Query Parameters' section with fields for begin\_date, end\_date, fq, page, q, and sort. The 'begin\_date' field is set to 'string matches ^\d{8}\$' and has a placeholder 'Begin date (e.g. 20120101)'. The 'end\_date' field is set to 'string matches ^\d{8}\$' and has a placeholder 'End date (e.g. 20121231)'. The 'fq' field is set to 'string' and has a placeholder 'Filter query'. The 'page' field is set to 'integer' and has a placeholder '0 ≤ value ≤ 100'. The 'q' field is set to 'string'. The 'sort' field is a dropdown menu. On the right side of the page is a 'Try this API' panel with 'Request parameters' for begin\_date, end\_date, fq, page, q, and sort, and a 'Credentials' section for 'apikey'. A blue button labeled 'EXECUTE' is at the bottom of the panel. A teal arrow points from the 'AUTHORIZE' button in the top right to the 'apikey' credential field in the 'Try this API' panel.

ARTICLE SEARCH

PATHS

/articlesearch.json **GET**

COMPONENTS

Schemas

Article

Byline

Headline

Image

Keyword

Multimedia

**GET /articlesearch.json**

Search for NYT articles by keywords and filters.

**HTTP request**

<https://api.nytimes.com/svc/search/v2/articlesearch.json>

**Query Parameters**

begin\_date string matches ^\d{8}\$  
Begin date (e.g. 20120101)

end\_date string matches ^\d{8}\$  
End date (e.g. 20121231)

fq string  
Filter query

page integer  
0 ≤ value ≤ 100  
Page number (0, 1, ...)

Try this API

Request parameters

begin\_date string

end\_date string

fq string

page integer

q string

sort

Credentials

apikey

**EXECUTE**

You will need to authorize your credentials

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

## 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 either:

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

```
$ ./<filename.sh>
```

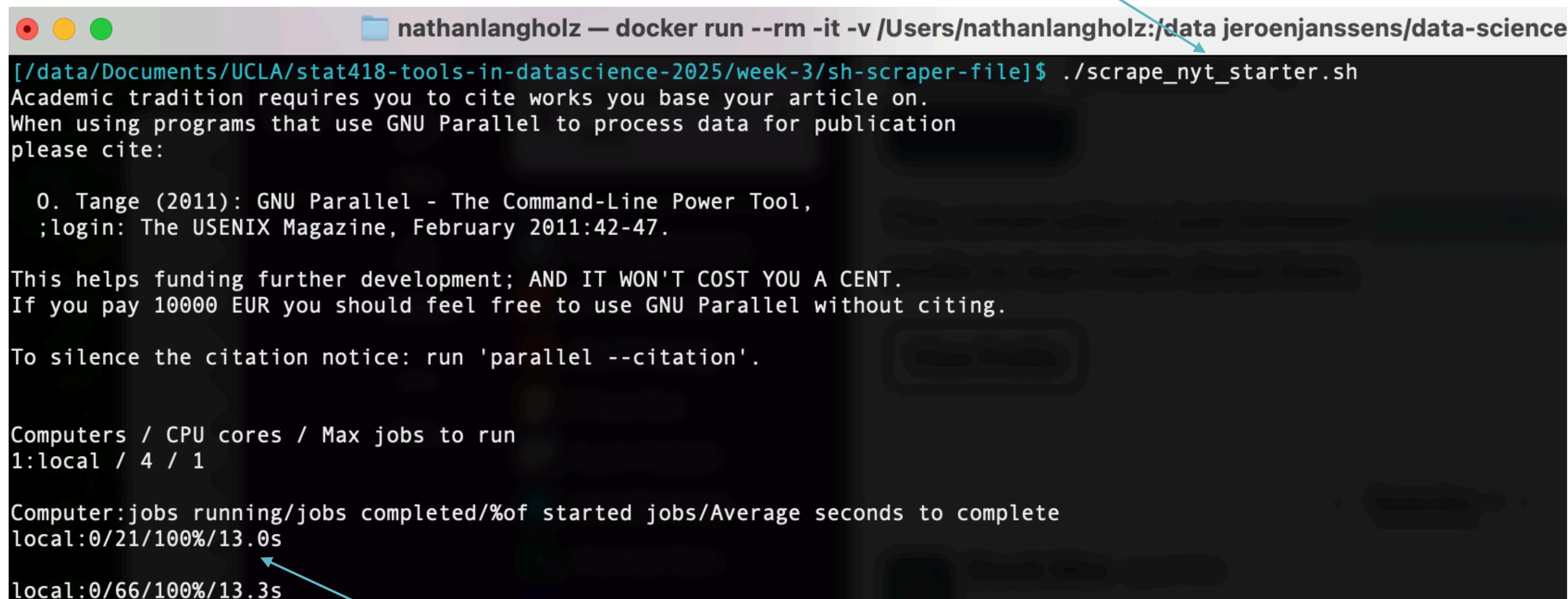
This should explain the .sh suffix; this naming convention will help you (and others) recognize this as a shell script

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

Lets see how this works  
in the shell

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

Executing .sh file



```
nathanlangholz — docker run --rm -it -v /Users/nathanlangholz:/data jeroenjanssens/data-science
[/data/Documents/UCLA/stat418-tools-in-datasience-2025/week-3/sh-scraper-file]$ ./scrape_nyt_starter.sh
Academic tradition requires you to cite works you base your article on.
When using programs that use GNU Parallel to process data for publication
please cite:

O. Tange (2011): GNU Parallel - The Command-Line Power Tool,
;login: The USENIX Magazine, February 2011:42-47.

This helps funding further development; AND IT WON'T COST YOU A CENT.
If you pay 10000 EUR you should feel free to use GNU Parallel without citing.

To silence the citation notice: run 'parallel --citation'.

Computers / CPU cores / Max jobs to run
1:local / 4 / 1

Computer:jobs running/jobs completed/%of started jobs/Average seconds to complete
local:0/21/100%/13.0s
local:0/66/100%/13.3s
```

Average amount of time  
to complete request

```
#!/bin/bash

#API_KEY = $1
#echo "API Key: $API_KEY"

# this pulls data from the NY Times developer api through the 'Data Science at the Command Line' book linux environment
#script adapted from 'Data Science at the Command Line' book

#https://api.nytimes.com/svc/search/v2/articlesearch.json?q=tariffs&begin_date=20200101&end_date=20251231&page=1&api-key='$API_KEY'

#takes some time as there is a 13 second delay
parallel -j1 --progress --delay 13 --results results "curl -sL \"\
"http://api.nytimes.com/svc/search/v2/articlesearch.json?q=tariffs&\"\
"begin_date={1}0101&end_date={1}1231&page={2}&sort=newest&api-key='\"\
""{your-api-key}" :: {2021..2025} :: {0..20} > /dev/null
cat results/*/*/*stdout | head -n 5
tree results | head
cat results/*/*/*stdout | jq -c '.response.docs[] | {date: .pub_date, type: .document_type, \
'title: .headline.main }' > articles
jq -r '[.date, .type, .title] | @csv' articles > tariff-articles.csv
rm articles
rm -r results
wc -l tariff-articles.csv
#< tariff-articles.csv cols -c date cut -dT -f1 | head | csvlook
```

Would need to pass put your api-key or pass it in the execution command

There is a lot going on in this command. I have delayed each request by 13 seconds, and am searching for articles from 2021 to 2025 with 20 pages of results

Format results from json to csv from these two lines. Also where columns are selected

Snippet of data returned from API request

```
nk":9}, {"name": "Location", "value": "Sunderland (England)", "rank": 10}], "multimedia": {"caption": "Ashwani Gupta, Nissan's chief operating officer, in Sunderland, England, on Thursday.", "credit": "Phil Noble/Reuters", "default": {"url": "https://static01.nyt.com/images/2021/07/01/business/01economy-briefing-nissan/merlin_190183539_efee7418-a627-4e98-995d-8010a75a9c54-articleLarge.jpg", "height": 399, "width": 600}, "thumbnail": {"url": "https://static01.nyt.com/images/2021/07/01/business/01economy-briefing-nissan/thumbStandard.jpg", "height": 75, "width": 75}}, "news_desk": "Business", "print_page": 3, "print_section": "B", "pub_date": "2021-07-01T09:35:29Z", "section_name": "Business", "snippet": "The Japanese carmaker will also make a new electric vehicle in England as part of a £1 billion investment, partly supported by the government.", "source": "The New York Times", "subsection_name": "", "type_of_material": "News", "uri": "nyt://article/c22f8bf4-2a49-5b5a-a60a-2760d1b892f4", "web_url": "https://www.nytimes.com/2021/07/01/business/nissan-battery-factory-britain.html", "word_count": 613}, {"abstract": "Forced to pay more for wood, home builders want the Biden administration to settle a long-running dispute over Canadian imports. It won't be easy.", "byline": {"original": "By Thomas Kaplan"}, "document_type": "article", "headline": {"main": "High Lumber Prices Add Urgency to a Decades-Old Trade Fight", "kicker": "", "print_headline": "Lumber Prices Raise Stakes In an Old Feud With Canada"}, "id": "nyt://article/c763b808-86fa-5bb8-9a83-435e66b841a0", "keywords": [{"name": "Subject", "value": "Wood and Wood Products", "rank": 1}, {"name": "Subject", "value": "International Trade and World Market", "rank": 2}, {"name": "Subject", "value": "Customs (Tariff)", "rank": 3}, {"name": "Subject", "value": "Prices (Fares, Fees and Rates)", "rank": 4}, {"name": "Organization", "value": "National Assn of Home Builders", "rank": 5}, {"name": "Subject", "value": "Real Estate and Housing (Residential)", "rank": 6}, {"name": "Subject", "value": "United States International Relations", "rank": 7}, {"name": "Subject", "value": "United States Politics and Government", "rank": 8}], "multimedia": {"caption": "A house being built in Vestavia Hills, Ala. A trade dispute between the United States and Canada is adding a layer of diplomatic intrigue to the scramble for in-demand building materials.", "credit": "Wes Frazer for The New York Times", "default": {"url": "https://static01.nyt.com/images/2021/06/25/us/00dc-lumberfight01/merlin_189846987_a6129ce0-4bde-4d72-93f2-9a2c6073bb9d-articleLarge.jpg", "height": 400, "width": 600}, "thumbnail": {"url": "https://static01.nyt.com/images/2021/06/25/us/00dc-lumberfight01-thumbStandard.jpg", "height": 75, "width": 75}}, "news_desk": "Business", "print_page": 1, "print_section": "B", "pub_date": "2021-06-28T19:50:05Z", "section_name": "Business", "snippet": "Forced to pay more for wood, home builders want the Biden administration to settle a long-running dispute over Canadian imports. It won't be easy.", "source": "The New York Times", "subsection_name": "Economy", "type_of_material": "News", "uri": "nyt://article/c763b808-86fa-5bb8-9a83-435e66b841a0", "web_url": "https://www.nytimes.com/2021/06/28/business/economy/lumber-prices-canadian-trade.html", "word_count": 1518}, {"abstract": "It lifts all boats, but not by equal amounts.", "byline": {"original": "By Thomas B. Edsall"}, "document_type": "article", "headline": {"main": "Is Education No Longer the 'Great Equalizer?'", "kicker": "Guest Essay", "print_headline": ""}, "id": "nyt://article/bd5fb540-4ac3-5281-9bee-44cb33639e7a", "keywords": [{"name": "Subject", "value": "United States Economy", "rank": 1}, {"name": "Subject", "value": "Education", "rank": 2}, {"name": "Subject", "value": "Income Inequality", "rank": 3}, {"name": "Subject", "value": "Parenting", "rank": 4}, {"name": "Location", "value": "United States", "rank": 5}], "multimedia": {"caption": "", "credit": "Audra Melton for The New York Times", "default": {"url": "https://static01.nyt.com/images/2021/06/23/opinion/23edsall-new/merlin_174339084_85264678-082c-4a8d-95e7-de6b906d4ef5-articleLarge.jpg", "height": 400, "width": 600}, "thumbnail": {"url": "https://static01.nyt.com/images/2021/06/23/opinion/23edsall-new/23edsall-new-thumbStandard.jpg", "height": 75, "width": 75}}, "news_desk": "OpEd", "print_page": "", "print_section": "", "pub_date": "2021-06-23T09:00:14Z", "section_name": "Opinion", "snippet": "It lifts all boats, but not by equal amounts.", "source": "The New York Times", "subsection_name": "", "type_of_material": "Op-Ed", "uri": "nyt://article/bd5fb540-4ac3-5281-9bee-44cb33639e7a", "web_url": "https://www.nytimes.com/2021/06/23/opinion/education-poverty-intervention.html", "word_count": 2664}, {"metadata": {"hits": 10000, "offset": 120, "time": 138}}]}  
results  
└ 1  
  └ 2021  
    └ 2  
      └ 0  
        └ seq  
        └ stderr  
        └ stdout  
      └ 1  
        └ seq  
  └ 1050 tariff-articles.csv
```

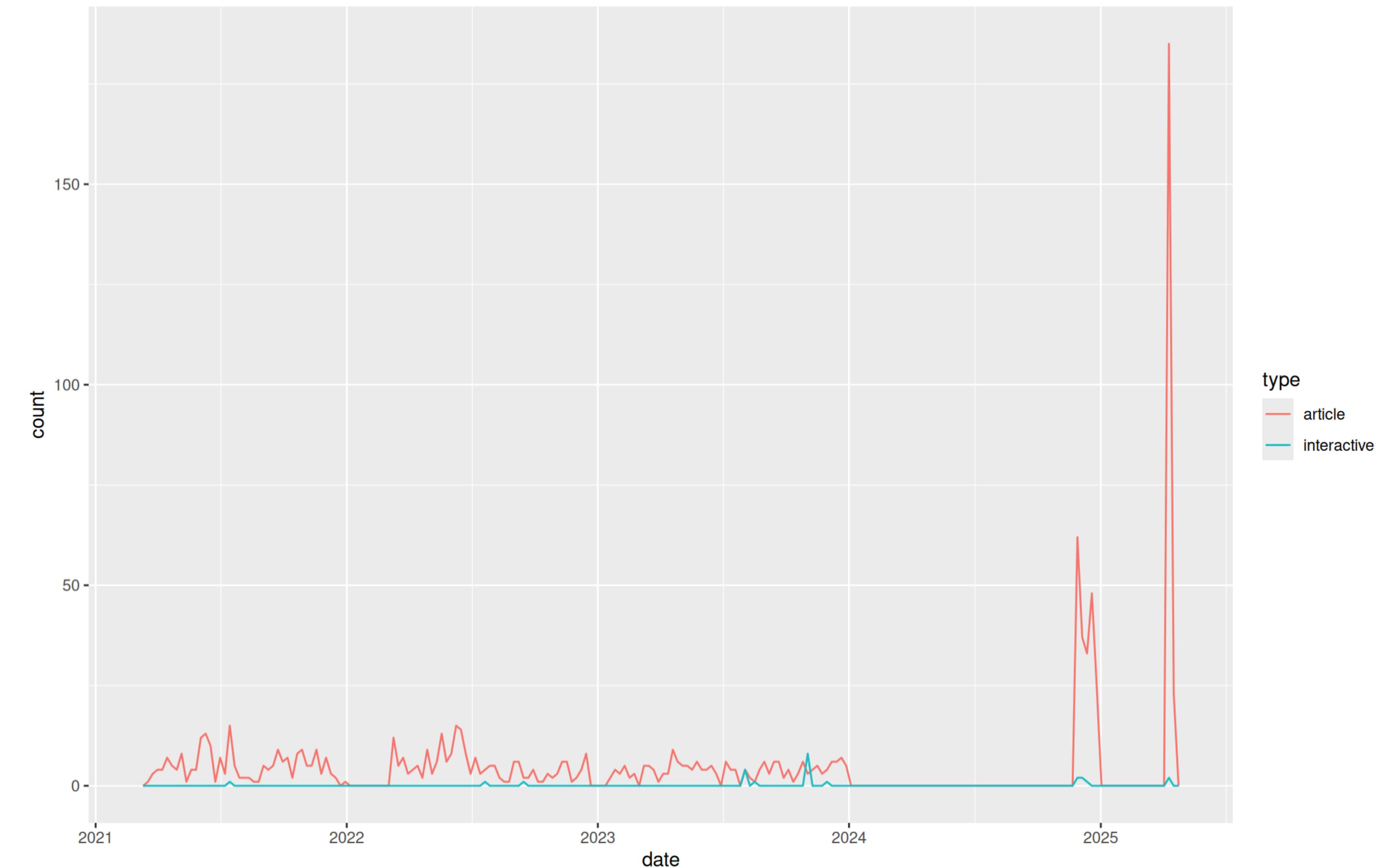
Number of articles

Results tree

## Tariffs Coverage in NY Times

The results

A .sh script exists in the GitHub repo to recreate this data acquisition



## R and the Shell

There is interactivity between R and the shell in way of:

- executing shell commands in R (`system2`)

- executing shell script commands in Rstudio (`ctrl + enter` sends to terminal)

- executing shell commands in RMarkdown

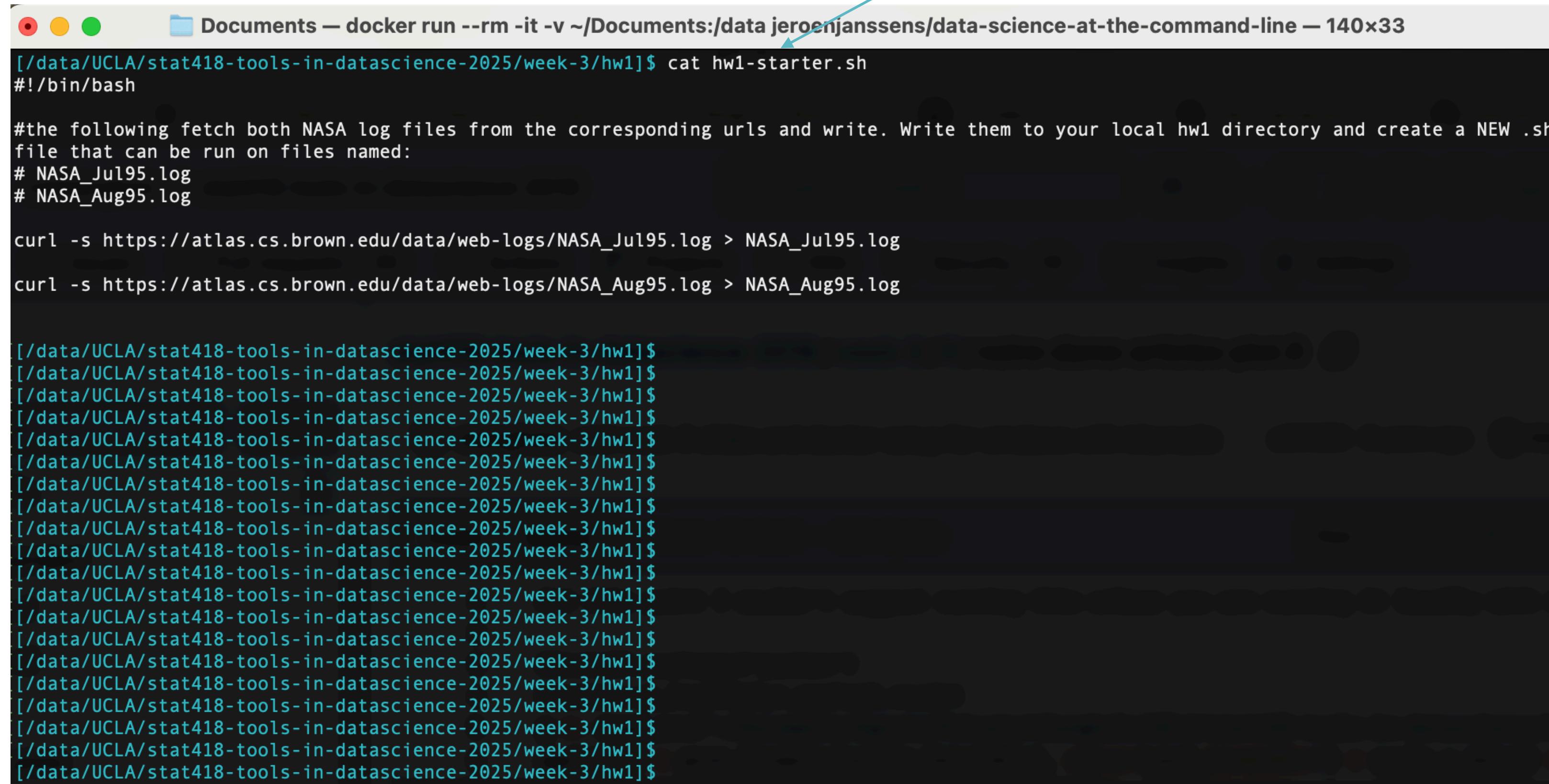
- execute R commands and scripts in the shell (assuming you have R on your machine)

```
$ R -e "head(mtcars); tail(mtcars)"
```

```
$ Rscript -e "head(mtcars)"
```

```
$ Rscript <Rscript.R>
```

cat returns the entirety of  
the file in the terminal



The screenshot shows a terminal window titled "Documents — docker run --rm -it -v ~/Documents:/data jeroenjanssens/data-science-at-the-command-line — 140x33". The terminal displays a bash script named "hw1-starter.sh". The script starts with "#!/bin/bash" and contains comments explaining its purpose: "the following fetch both NASA log files from the corresponding urls and write. Write them to your local hw1 directory and create a NEW .sh file that can be run on files named:". It then lists two "curl" commands to download "NASA\_Jul95.log" and "NASA\_Aug95.log" from the specified URLs and save them to the local "hw1" directory.

```
#!/bin/bash

#the following fetch both NASA log files from the corresponding urls and write. Write them to your local hw1 directory and create a NEW .sh
#file that can be run on files named:
# NASA_Jul95.log
# NASA_Aug95.log

curl -s https://atlas.cs.brown.edu/data/web-logs/NASA_Jul95.log > NASA_Jul95.log

curl -s https://atlas.cs.brown.edu/data/web-logs/NASA_Aug95.log > NASA_Aug95.log
```

curl goes to the  
specified url to  
download log file and  
then write it

What the July web log  
actually looks like...

```
Documents — docker run --rm -it -v ~/Documents:/data jeroenjanssens/data-science-at-the-command-line — 140x33
[/data/UCLA/stat418-tools-in-datasience-2025/week-3/hw1]$ ./data/UCLA/stat418-tools-in-datasience-2025/week-3/hw1$ cat NASA_Jul95.log | head -n 30
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/" HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/" HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/countdown.html HTTP/1.0" 200 3985
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET /shuttle/countdown/" HTTP/1.0" 200 3985
129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0" 200 7074
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /shuttle/countdown/count.gif HTTP/1.0" 200 40310
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 200 786
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
d104.aa.net - - [01/Jul/1995:00:00:15 -0400] "GET /shuttle/countdown/count.gif HTTP/1.0" 200 40310
d104.aa.net - - [01/Jul/1995:00:00:15 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 200 786
d104.aa.net - - [01/Jul/1995:00:00:15 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
129.94.144.152 - - [01/Jul/1995:00:00:17 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:17 -0400] "GET /images/launch-logo.gif HTTP/1.0" 200 1713
ppptky391.asahi-net.or.jp - - [01/Jul/1995:00:00:18 -0400] "GET /facts/about_ksc.html HTTP/1.0" 200 3977
net-1-141.eden.com - - [01/Jul/1995:00:00:19 -0400] "GET /shuttle/missions/sts-71/images/KSC-95EC-0916.jpg HTTP/1.0" 200 34029
ppptky391.asahi-net.or.jp - - [01/Jul/1995:00:00:19 -0400] "GET /images/launchpalms-small.gif HTTP/1.0" 200 11473
205.189.154.54 - - [01/Jul/1995:00:00:24 -0400] "GET /shuttle/countdown/" HTTP/1.0" 200 3985
waters-gw.starway.net.au - - [01/Jul/1995:00:00:25 -0400] "GET /shuttle/missions/51-l/mission-51-l.html HTTP/1.0" 200 6723
ppp-mia-30.shadow.net - - [01/Jul/1995:00:00:27 -0400] "GET / HTTP/1.0" 200 7074
205.189.154.54 - - [01/Jul/1995:00:00:29 -0400] "GET /shuttle/countdown/count.gif HTTP/1.0" 200 40310
alyssa.prodigy.com - - [01/Jul/1995:00:00:33 -0400] "GET /shuttle/missions/sts-71/sts-71-patch-small.gif HTTP/1.0" 200 12054
ppp-mia-30.shadow.net - - [01/Jul/1995:00:00:35 -0400] "GET /images/ksclogo-medium.gif HTTP/1.0" 200 5866
dial22.lloyd.com - - [01/Jul/1995:00:00:37 -0400] "GET /shuttle/missions/sts-71/images/KSC-95EC-0613.jpg HTTP/1.0" 200 61716
smyth-pc.moorecap.com - - [01/Jul/1995:00:00:38 -0400] "GET /history/apollo/apollo-13/images/70HC314.GIF HTTP/1.0" 200 101267
205.189.154.54 - - [01/Jul/1995:00:00:40 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 200 786
[/data/UCLA/stat418-tools-in-datasience-2025/week-3/hw1]$
```

Log format (for homework) - The logs are ASCII with one line per request with the following comments

host - making the request. A hostname when possible otherwise, the IP address

timestamp - in the format “DAY MON DD HH:MM:SS YYYY”

request - given in quotes

HTTP - reply code

bytes - in the reply