Biostat 203B Homework 1

Due Jan 24, 2024 @ 11:59PM

Yanzi Sun 106183069

Table of contents

Display machine information for reproducibility:

```r
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS Sonoma 14.7.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Los_Angeles
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base
```

```
loaded via a namespace (and not attached):
 [1] compiler_4.4.2    fastmap_1.2.0     cli_3.6.3         tools_4.4.2
 [5] htmltools_0.5.8.1 rstudioapi_0.17.1 yaml_2.3.10       rmarkdown_2.29
 [9] knitr_1.49        jsonlite_1.8.9    xfun_0.50         digest_0.6.37
[13] rlang_1.1.4       evaluate_1.0.1
```

Q1. Git/GitHub

No handwritten homework reports are accepted for this course. We work with Git and GitHub. Efficient and abundant use of Git, e.g., frequent and well-documented commits, is an important criterion for grading your homework.

1. Apply for the Student Developer Pack at GitHub using your UCLA email. You'll get GitHub Pro account for free (unlimited public and private repositories).
2. Create a **private** repository `biostat-203b-2025-winter` and add `Hua-Zhou` and TA team (`Tomoki-Okuno` for Lec 1; `parsajamshidian` and `BowenZhang2001` for Lec 82) as your collaborators with write permission.
3. Top directories of the repository should be `hw1`, `hw2`, … Maintain two branches `main` and `develop`. The `develop` branch will be your main playground, the place where you develop solution (code) to homework problems and write up report. The `main` branch will be your presentation area. Submit your homework files (Quarto file `qmd`, `html` file converted by Quarto, all code and extra data sets to reproduce results) in the `main` branch.
4. After each homework due date, course reader and instructor will check out your `main` branch for grading. Tag each of your homework submissions with tag names `hw1`, `hw2`, … Tagging time will be used as your submission time. That means if you tag your `hw1` submission after deadline, penalty points will be deducted for late submission.
5. After this course, you can make this repository public and use it to demonstrate your skill sets on job market.

Solution: Done on Jan11.

Q2. Data ethics training

This exercise (and later in this course) uses the MIMIC-IV data v3.1, a freely accessible critical care database developed by the MIT Lab for Computational Physiology. Follow the instructions at https://mimic.mit.edu/docs/gettingstarted/ to (1) complete the CITI `Data or Specimens Only Research` course and (2) obtain the PhysioNet credential for using the MIMIC-IV data. Display the verification links to your completion report and completion certificate here. **You must complete Q2 before working on the remaining questions.**

(Hint: The CITI training takes a few hours and the PhysioNet credentialing takes a couple days; do not leave it to the last minute.)

Solution: Here is the link of the completion report:https://www.citiprogram.org/verify/?k1283ea47-0ca3-42b7-9d2b-e721bfb691cb-67209968, completion certificate:https://www.citiprogram.org/verify/?w3df54344-f850-42f3-b0df-9a632ad2eecf-67209968

Q3. Linux Shell Commands

1. Make the MIMIC-IV v3.1 data available at location `~/mimic`. The output of the `ls -l ~/mimic` command should be similar to the below (from my laptop).

```
# content of mimic folder
ls -l ~/mimic/
```

```
total 56
-rw-r--r--   1 yanzisun  staff  15199 Oct 10 13:29 CHANGELOG.txt
-rw-r--r--   1 yanzisun  staff   2518 Oct 10 14:30 LICENSE.txt
-rw-r--r--   1 yanzisun  staff   2884 Oct 11 14:55 SHA256SUMS.txt
drwxr-xr-x@ 24 yanzisun  staff    768 Jan 23 23:05 hosp
drwxr-xr-x@ 11 yanzisun  staff    352 Jan 21 09:25 icu
-rw-r--r--   1 yanzisun  staff    789 Jan 14 11:41 index.html
```

Refer to the documentation https://physionet.org/content/mimiciv/3.1/ for details of data files. Do **not** put these data files into Git; they are big. Do **not** copy them into your directory. Do **not** decompress the gz data files. These create unnecessary big files and are not big-data-friendly practices. Read from the data folder `~/mimic` directly in following exercises.

Use Bash commands to answer following questions.

Solution: I downloaded MIMIC IV v3.1 to my computer and made it available at ‘∼/mimic/’.

2. Display the contents in the folders `hosp` and `icu` using Bash command `ls -l`. Why are these data files distributed as `.csv.gz` files instead of `.csv` (comma separated values) files? Read the page https://mimic.mit.edu/docs/iv/ to understand what's in each folder.

Solution: Below is the answer to question 2. The files are gzip-compressed csv files so they has .gz at the end of file name. Compressing files can save storage space.

```
ls -l ~/mimic/hosp
ls -l ~/mimic/icu
```

```
total 12306248
-rw-r--r--@ 1 yanzisun  staff    19928140 Jun 24  2024 admissions.csv.gz
-rw-r--r--@ 1 yanzisun  staff      427554 Apr 12  2024 d_hcpcs.csv.gz
-rw-r--r--@ 1 yanzisun  staff      876360 Apr 12  2024 d_icd_diagnoses.csv.gz
-rw-r--r--@ 1 yanzisun  staff      589186 Apr 12  2024 d_icd_procedures.csv.gz
-rw-r--r--@ 1 yanzisun  staff       13169 Oct  3 09:07 d_labitems.csv.gz
-rw-r--r--@ 1 yanzisun  staff    33564802 Oct  3 09:07 diagnoses_icd.csv.gz
-rw-r--r--@ 1 yanzisun  staff     9743908 Oct  3 09:07 drgcodes.csv.gz
-rw-r--r--@ 1 yanzisun  staff   811305629 Apr 12  2024 emar.csv.gz
-rw-r--r--@ 1 yanzisun  staff   748158322 Apr 12  2024 emar_detail.csv.gz
-rw-r--r--@ 1 yanzisun  staff     2162335 Apr 12  2024 hcpcsevents.csv.gz
-rw-r--r--@ 1 yanzisun  staff  2592909134 Oct  3 09:08 labevents.csv.gz
-rw-r--r--@ 1 yanzisun  staff   117644075 Oct  3 09:08 microbiologyevents.csv.gz
-rw-r--r--@ 1 yanzisun  staff    44069351 Oct  3 09:08 omr.csv.gz
-rw-r--r--@ 1 yanzisun  staff     2835586 Apr 12  2024 patients.csv.gz
-rw-r--r--@ 1 yanzisun  staff   525708076 Apr 12  2024 pharmacy.csv.gz
-rw-r--r--@ 1 yanzisun  staff   666594177 Apr 12  2024 poe.csv.gz
-rw-r--r--@ 1 yanzisun  staff    55267894 Apr 12  2024 poe_detail.csv.gz
-rw-r--r--@ 1 yanzisun  staff   606298611 Apr 12  2024 prescriptions.csv.gz
-rw-r--r--@ 1 yanzisun  staff     7777324 Apr 12  2024 procedures_icd.csv.gz
-rw-r--r--@ 1 yanzisun  staff      127330 Apr 12  2024 provider.csv.gz
-rw-r--r--@ 1 yanzisun  staff     8569241 Apr 12  2024 services.csv.gz
-rw-r--r--@ 1 yanzisun  staff    46185771 Oct  3 09:08 transfers.csv.gz
total 8506784
-rw-r--r--@ 1 yanzisun  staff       41566 Apr 12  2024 caregiver.csv.gz
-rw-r--r--@ 1 yanzisun  staff  3502392765 Apr 12  2024 chartevents.csv.gz
-rw-r--r--@ 1 yanzisun  staff       58741 Apr 12  2024 d_items.csv.gz
-rw-r--r--@ 1 yanzisun  staff    63481196 Apr 12  2024 datetimeevents.csv.gz
-rw-r--r--@ 1 yanzisun  staff     3342355 Oct  3 07:36 icustays.csv.gz
-rw-r--r--@ 1 yanzisun  staff   311642048 Apr 12  2024 ingredientevents.csv.gz
-rw-r--r--@ 1 yanzisun  staff   401088206 Apr 12  2024 inputevents.csv.gz
-rw-r--r--@ 1 yanzisun  staff    49307639 Apr 12  2024 outputevents.csv.gz
-rw-r--r--@ 1 yanzisun  staff    24096834 Apr 12  2024 procedureevents.csv.gz
```

3. Briefly describe what Bash commands `zcat`, `zless`, `zmore`, and `zgrep` do.

Solution: zcat allows to view a compressed (zipped) file directly, basically a cat command for zipped files.zless view compressed file contents in a paginated way, can also do search option; zmore view compressed file contents in a new tab; zgrep search string/pattern in the compressed file.

4. (Looping in Bash) What's the output of the following bash script?

```
for datafile in ~/mimic/hosp/{a,l,pa}*.gz
do
  ls -l $datafile
done
```

Solution: The results are a long list(including file permission and sizes) of all files starting with “a,l, or pa” inside the hosp folder.

Display the number of lines in each data file using a similar loop. (Hint: combine linux commands `zcat <` and `wc -l`.)

Solution:

```
for datafile in ~/mimic/hosp/{a,l,pa}*.gz
do
  zcat < $datafile | wc -l
done
```

```
 546029
158374765
 364628
```

5. Display the first few lines of `admissions.csv.gz`. How many rows are in this data file, excluding the header line? Each `hadm_id` identifies a hospitalization. How many hospitalizations are in this data file? How many unique patients (identified by `subject_id`) are in this data file? Do they match the number of patients listed in the `patients.csv.gz` file? (Hint: combine Linux commands `zcat <`, `head`/`tail`, `awk`, `sort`, `uniq`, `wc`, and so on.)

Solution: There are 546028 rows in this file excluding the headerline. There are 546028 hospitalizations in this data file. There are 223452 unique patients, which does not matchthe number of patients listed in the patients.csv(364627)

```
zcat < ~/mimic/hosp/admissions.csv.gz | head -5
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | wc -l
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F',' '{print $2}' | wc -l
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F',' '{print $1}' | uniq| wc -l
zcat < ~/mimic/hosp/patients.csv.gz | tail -n +2 | uniq | wc -l
```

```
subject_id,hadm_id,admittime,dischtime,deathtime,admission_type,admit_provider_id,admission_
10000032,22595853,2180-05-06 22:23:00,2180-05-07 17:15:00,,URGENT,P49AFC,TRANSFER FROM HOSPI
10000032,22841357,2180-06-26 18:27:00,2180-06-27 18:49:00,,EW EMER.,P784FA,EMERGENCY ROOM,HO
```

```
10000032,25742920,2180-08-05 23:44:00,2180-08-07 17:50:00,,EW EMER.,P19UTS,EMERGENCY ROOM,HOS
10000032,29079034,2180-07-23 12:35:00,2180-07-25 17:55:00,,EW EMER.,P06OTX,EMERGENCY ROOM,HOM
  546028
  546028
  223452
  364627
```

6. What are the possible values taken by each of the variable `admission_type`, `admission_location`, `insurance`, and `ethnicity`? Also report the count for each unique value of these variables in decreasing order. (Hint: combine Linux commands `zcat`, `head`/`tail`, `awk`, `uniq -c`, `wc`, `sort`, and so on; skip the header line.)

Solution: Column 6,8,10,13 corresponds to `admission_type`, `admission_location`, `insurance`, and `ethnicity`.

```
for col in 6 8 10 13;
  do
  echo "Count and sorted unique values of variable $col:"
  zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F, -v c=$col '{print $c}' | sort
  echo
done
```

```
Count and sorted unique values of variable 6:
177459 EW EMER.
119456 EU OBSERVATION
84437 OBSERVATION ADMIT
54929 URGENT
42898 SURGICAL SAME DAY ADMISSION
24551 DIRECT OBSERVATION
21973 DIRECT EMER.
13130 ELECTIVE
7195 AMBULATORY OBSERVATION

Count and sorted unique values of variable 8:
244179 EMERGENCY ROOM
163228 PHYSICIAN REFERRAL
56227 TRANSFER FROM HOSPITAL
42365 WALK-IN/SELF REFERRAL
12965 CLINIC REFERRAL
8518 PROCEDURE SITE
6317 TRANSFER FROM SKILLED NURSING FACILITY
5837 INTERNAL TRANSFER TO OR FROM PSYCH
5734 PACU
```

```
 402 INFORMATION NOT AVAILABLE
 255 AMBULATORY SURGERY TRANSFER
   1

Count and sorted unique values of variable 10:
244576 Medicare
173399 Private
104229 Medicaid
14006 Other
9355
 463 No charge

Count and sorted unique values of variable 13:
336538 WHITE
75482 BLACK/AFRICAN AMERICAN
19788 OTHER
13972 WHITE - OTHER EUROPEAN
13870 UNKNOWN
10903 HISPANIC/LATINO - PUERTO RICAN
8287 HISPANIC OR LATINO
7809 ASIAN
7644 ASIAN - CHINESE
6597 WHITE - RUSSIAN
6205 BLACK/CAPE VERDEAN
6070 HISPANIC/LATINO - DOMINICAN
3875 BLACK/CARIBBEAN ISLAND
3495 BLACK/AFRICAN
3478 UNABLE TO OBTAIN
2162 PATIENT DECLINED TO ANSWER
2082 PORTUGUESE
1973 ASIAN - SOUTH EAST ASIAN
1886 WHITE - EASTERN EUROPEAN
1858 HISPANIC/LATINO - GUATEMALAN
1661 ASIAN - ASIAN INDIAN
1526 WHITE - BRAZILIAN
1320 HISPANIC/LATINO - SALVADORAN
1247 AMERICAN INDIAN/ALASKA NATIVE
 920 HISPANIC/LATINO - COLUMBIAN
 883 HISPANIC/LATINO - MEXICAN
 774 SOUTH AMERICAN
 725 HISPANIC/LATINO - HONDURAN
 664 ASIAN - KOREAN
 641 HISPANIC/LATINO - CUBAN
```

```
603 HISPANIC/LATINO - CENTRAL AMERICAN
596 MULTIPLE RACE/ETHNICITY
494 NATIVE HAWAIIAN OR OTHER PACIFIC ISLANDER
```

7. The `icusays.csv.gz` file contains all the ICU stays during the study period. How many ICU stays, identified by `stay_id`, are in this data file? How many unique patients, identified by `subject_id`, are in this data file?

Solution: There are 94459 stays and 65367 unique patients in this data file.

```
zcat < ~/mimic/icu/icustays.csv.gz | awk -F ',' '{print $3}' | wc -l
zcat < ~/mimic/icu/icustays.csv.gz | awk -F ',' '{print $1}' | uniq | wc -l
```

```
   94459
   65367
```

8. *To compress, or not to compress. That's the question.* Let's focus on the big data file `labevents.csv.gz`. Compare compressed gz file size to the uncompressed file size. Compare the run times of `zcat < ~/mimic/labevents.csv.gz | wc -l` versus `wc -l labevents.csv`. Discuss the trade off between storage and speed for big data files. (Hint: `gzip -dk < FILENAME.gz > ./FILENAME`. Remember to delete the large `labevents.csv` file after the exercise.)

Solution: From the results I get, I see the runtimes are basically the same. however, the unzipped .csv file takes much larger storage than its zipped .csv.gz form. I will prefer to use zcat for compressed files.

```
time zcat < ~/mimic/hosp/labevents.csv.gz | wc -l
gzip -k -d ~/mimic/hosp/labevents.csv.gz
time wc -l ~/mimic/hosp/labevents.csv
rm ~/mimic/hosp/labevents.csv
```

```
 158374765

real    0m18.724s
user    0m29.245s
sys 0m1.795s
 158374765 /Users/yanzisun/mimic/hosp/labevents.csv

real    0m19.497s
user    0m17.492s
sys 0m1.741s
```

Q4. Who's popular in Price and Prejudice

1. You and your friend just have finished reading *Pride and Prejudice* by Jane Austen. Among the four main characters in the book, Elizabeth, Jane, Lydia, and Darcy, your friend thinks that Darcy was the most mentioned. You, however, are certain it was Elizabeth. Obtain the full text of the novel from http://www.gutenberg.org/cache/epub/42671/pg42671.txt and save to your local folder.

```
wget -nc http://www.gutenberg.org/cache/epub/42671/pg42671.txt
```

Explain what `wget -nc` does. Do **not** put this text file `pg42671.txt` in Git. Complete the following loop to tabulate the number of times each of the four characters is mentioned using Linux commands.

Solution: ‘wget -nc’ downloads file from websites, -nc is the option that specifies not to create duplicate if the file was downloaded already. The counts for each names are calculated below.

```
for char in Elizabeth Jane Lydia Darcy
do
  echo $char:
  grep -c "$char" pg42671.txt
done
```

```
Elizabeth:
633
Jane:
289
Lydia:
166
Darcy:
414
```

2. What's the difference between the following two commands?

```
echo 'hello, world' > test1.txt
```

and

```
echo 'hello, world' >> test2.txt
```

Solution: > redirects the standard output to a file.Specifically, > save text to output file (overwrite if file exist); » append to the end of the output file and saves it.

3. Using your favorite text editor (e.g., `vi`), type the following and save the file as `middle.sh`:

```
#!/bin/sh
# Select lines from the middle of a file.
# Usage: bash middle.sh filename end_line num_lines
head -n "$2" "$1" | tail -n "$3"
```

Using `chmod` to make the file executable by the owner, and run

```
./middle.sh pg42671.txt 20 5
```

```


Release date: May 9, 2013 [eBook #42671]

Language: English

```

Explain the output. Explain the meaning of `"$1"`, `"$2"`, and `"$3"` in this shell script. Why do we need the first line of the shell script?

Solution: I used this chmod command line to edit user access:chmod u+x middle.sh The output is the result of running shell script written in [middle.sh] on [pg42671.txt]. The "$1" represents the filename, "$2" represents the end_line, and "$3" represents the number of lines. Basically the head command is translated to: head -n endline filename | tail -n num lines. When the shell script is executable and applied to the txt file, it extracts line 15 to 20 as the end line is 20 and the number of lines are 5.

Q5. More fun with Linux

Try following commands in Bash and interpret the results: `cal`, `cal 2025`, `cal 9 1752` (anything unusual?), `date`, `hostname`, `arch`, `uname -a`, `uptime`, `who am i`, `who`, `w`, `id`, `last | head`, `echo {con,pre}{sent,fer}{s,ed}`, `time sleep 5`, `history | tail`.

Solution: I played with all commands and # them so they do not run multiple times when I test the other commands. cal gives the calender of the year or month indicated. date gives the current date and time based off your system time zone. hostname gives the name of my PC. arch gives my PC's architecture type. uname -a gives detail information of my PC, including kernel, hostname, operating system, time, root, and architecture. uptime gives current system time, how long the PC has been running since last restart, average load time, and number of users. who am i tells the user is me and gives system time. who gives a list of users and time using the PC. w provides a detailed use history of past users. id gives information about the user identity. last provides a list of logins into the PC, head option limits the output number to 10. echo {con,pre}{sent,fer}{s,ed} prints out all possible combinations of the characters in curly brackets. 2x2x2=8 the sleep command pauses execution for 5 seconds and the time command measures how long it actually takes. history | tail displays the last 10 commands from my shell history.

```
#cal 2025
#cal 1 2001
#date
#hostname
arch
#uname -a
#uptime
#who am i
who
w
#id
#last |head
#echo {con,pre}{sent,fer}{s,ed}
#time sleep 5
#history | tail
```

```
arm64yanzisun         ttys000      Jan 23 15:34
yanzisun         ttys001      Jan 23 15:34
yanzisun         ttys002      Jan 23 15:34
yanzisun         ttys003      Jan 23 17:31
yanzisun         console      Jan 23 15:34
23:09  up 20 days,  1:26, 5 users, load averages: 2.03 2.08 2.07
USER       TTY      FROM    LOGIN@  IDLE WHAT
yanzisun   s000     -      15:34    7:35 -bash
yanzisun   s001     -      15:34    7:35 -bash
yanzisun   s002     -      15:34    6:01 -bash
yanzisun   s003     -      17:31       7 -bash
yanzisun   console  -      15:34    7:35 -
```

Q6. Book

1. Git clone the repository https://github.com/christophergandrud/Rep-Res-Book for the book *Reproducible Research with R and RStudio* to your local machine. Do **not** put this repository within your homework repository `biostat-203b-2025-winter`.

Solution: Done. I # them because rendering it on my local creates duplicates everytime running this cell. If you want to run it, please remove #.

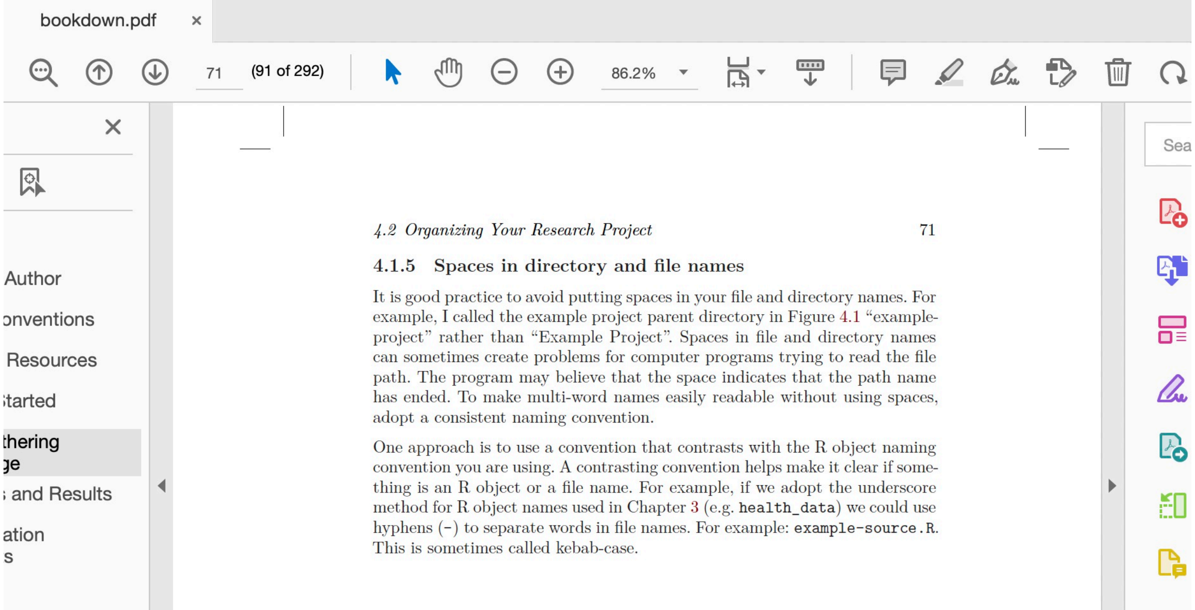
```
#git init
#git clone https://github.com/christophergandrud/Rep-Res-Book
```

2. Open the project by clicking `rep-res-3rd-edition.Rproj` and compile the book by clicking `Build Book` in the `Build` panel of RStudio. (Hint: I was able to build `git_book` and `epub_book` directly. For `pdf_book`, I needed to add a line `\usepackage{hyperref}` to the file `Rep-Res-Book/rep-res-3rd-edition/latex/preabmle.tex`.)

The point of this exercise is (1) to obtain the book for free and (2) to see an example how a complicated project such as a book can be organized in a reproducible way. Use `sudo apt install PKGNAME` to install required Ubuntu packages and `tlmgr install PKGNAME` to install missing TexLive packages.

For grading purpose, include a screenshot of Section 4.1.5 of the book here.

Solution: I cloned the repository and compiled the book. here is the screenshot of 4.1.5 of the book.



4.2 Organizing Your Research Project 71

4.1.5 Spaces in directory and file names

It is good practice to avoid putting spaces in your file and directory names. For example, I called the example project parent directory in Figure 4.1 "example-project" rather than "Example Project". Spaces in file and directory names can sometimes create problems for computer programs trying to read the file path. The program may believe that the space indicates that the path name has ended. To make multi-word names easily readable without using spaces, adopt a consistent naming convention.

One approach is to use a convention that contrasts with the R object naming convention you are using. A contrasting convention helps make it clear if something is an R object or a file name. For example, if we adopt the underscore method for R object names used in Chapter 3 (e.g. `health_data`) we could use hyphens (`-`) to separate words in file names. For example: `example-source.R`. This is sometimes called kebab-case.