Biostat 203B Homework 1

Due Jan 24, 2024 @ 11:59PM

Palash Raval and 406551574

Display machine information for reproducibility:

```
sessionInfo()
```

```
R version 4.3.1 (2023-06-16)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;  l

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_4.3.1    fastmap_1.2.0     cli_3.6.3         tools_4.3.1
 [5] htmltools_0.5.8.1 rstudioapi_0.17.1 yaml_2.3.10       rmarkdown_2.28
 [9] knitr_1.48        jsonlite_1.8.9    xfun_0.48         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: I completed all these steps and have a GitHub repository with `main` and `develop` branches.

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.)

Here is the Completion Certificate

Here is the Completion Report

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 0
-rw-r--r--@  1 palashraval  staff  15199 Oct 10 16:29 CHANGELOG.txt
-rw-r--r--@  1 palashraval  staff   2518 Oct 10 17:30 LICENSE.txt
-rw-r--r--@  1 palashraval  staff   2884 Oct 11 17:55 SHA256SUMS.txt
drwxr-xr-x@ 24 palashraval  staff    768 Jan 24 12:44 hosp
drwxr-xr-x@ 11 palashraval  staff    352 Jan 24 12:44 icu
```

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.

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.

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

```
total 5064280
-rw-r--r--@ 1 palashraval  staff    19928140 Jun 24  2024 admissions.csv.gz
-rw-r--r--@ 1 palashraval  staff      427554 Apr 12  2024 d_hcpcs.csv.gz
-rw-r--r--@ 1 palashraval  staff      876360 Apr 12  2024 d_icd_diagnoses.csv.gz
-rw-r--r--@ 1 palashraval  staff      589186 Apr 12  2024 d_icd_procedures.csv.gz
-rw-r--r--@ 1 palashraval  staff       13169 Oct  3 09:07 d_labitems.csv.gz
-rw-r--r--@ 1 palashraval  staff    33564802 Oct  3 09:07 diagnoses_icd.csv.gz
-rw-r--r--@ 1 palashraval  staff     9743908 Oct  3 09:07 drgcodes.csv.gz
-rw-r--r--@ 1 palashraval  staff   811305629 Apr 12  2024 emar.csv.gz
-rw-r--r--@ 1 palashraval  staff   748158322 Apr 12  2024 emar_detail.csv.gz
-rw-r--r--@ 1 palashraval  staff     2162335 Apr 12  2024 hcpcsevents.csv.gz
-rw-r--r--@ 1 palashraval  staff  2592909134 Oct  3 09:08 labevents.csv.gz
-rw-r--r--@ 1 palashraval  staff   117644075 Oct  3 09:08 microbiologyevents.csv.gz
```

```
-rw-r--r--@ 1 palashraval  staff    44069351 Oct  3 09:08 omr.csv.gz
-rw-r--r--@ 1 palashraval  staff     2835586 Apr 12  2024 patients.csv.gz
-rw-r--r--@ 1 palashraval  staff   525708076 Apr 12  2024 pharmacy.csv.gz
-rw-r--r--@ 1 palashraval  staff   666594177 Apr 12  2024 poe.csv.gz
-rw-r--r--@ 1 palashraval  staff    55267894 Apr 12  2024 poe_detail.csv.gz
-rw-r--r--@ 1 palashraval  staff   606298611 Apr 12  2024 prescriptions.csv.gz
-rw-r--r--@ 1 palashraval  staff     7777324 Apr 12  2024 procedures_icd.csv.gz
-rw-r--r--@ 1 palashraval  staff      127330 Apr 12  2024 provider.csv.gz
-rw-r--r--@ 1 palashraval  staff     8569241 Apr 12  2024 services.csv.gz
-rw-r--r--@ 1 palashraval  staff    46185771 Oct  3 09:08 transfers.csv.gz
```

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

```
total 0
-rw-r--r--@ 1 palashraval  staff       41566 Apr 12  2024 caregiver.csv.gz
-rw-r--r--@ 1 palashraval  staff  3502392765 Apr 12  2024 chartevents.csv.gz
-rw-r--r--@ 1 palashraval  staff       58741 Apr 12  2024 d_items.csv.gz
-rw-r--r--@ 1 palashraval  staff    63481196 Apr 12  2024 datetimeevents.csv.gz
-rw-r--r--@ 1 palashraval  staff     3342355 Oct  3 07:36 icustays.csv.gz
-rw-r--r--@ 1 palashraval  staff   311642048 Apr 12  2024 ingredientevents.csv.gz
-rw-r--r--@ 1 palashraval  staff   401088206 Apr 12  2024 inputevents.csv.gz
-rw-r--r--@ 1 palashraval  staff    49307639 Apr 12  2024 outputevents.csv.gz
-rw-r--r--@ 1 palashraval  staff    24096834 Apr 12  2024 procedureevents.csv.gz
```

Solution: Both icu and hosp contain large amounts of data. Hosp contains detailed information about the patient at a hospital-level, while icu contains detailed information about ICU admissions. Due to the large volume of data, the data files must be compressed in order to reduce the overall size of the data files, which is why they are .csv.gz and not just .csv

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

Solution: 'zcat' allows for the entire viewing of compressed files(.gz), but will temporarily decompress the file first. 'zless' allows for the viewing of compressed files, but is done so in a way that scrolling is utilized to scan through the entire text in the data file. 'zmore' allows for the viewing of compressed files one bunch/grouping/chunk at a time. 'zgrep' helps with searching through a compressed file for a specific text within the file in question.

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 output of the script gives information about permissions, user name, group name, file size, etc. for these 3 files: 'admissions.csv.gz', 'labevents.csv.gz', and 'patients.csv.gz'.

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

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

Solution: 'admissions.csv.gz' has 546029 lines, 'labevents.csv.gz' has 158374765 lines, and 'patients.csv.gz' has 364628 lines.

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.)

```
zcat < ~/mimic/hosp/admissions.csv.gz | head -5
```

```
subject_id,hadm_id,admittime,dischtime,deathtime,admission_type,admit_provider_id,admission_l
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,HO
```

Solution: Above is the first 5 lines of the 'admissions.csv.gz' file.

```
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | wc -l
```

```
  546028
```

Solution: There are 546028 rows in the 'admissions.csv.gz' file without the header line.

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

```
  546028
```

Solution: There are 546028 hospitalizations in this data file.

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

```
  223452
```

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

```
  364627
```

Solution: There are 223452 unique patients in the 'admissions.csv.gz' data file. No, it does not match the number of patients in the 'patients.csv.gz' data file since it had 364627 unique values.

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.)

```
zcat < ~/mimic/hosp/admissions.csv.gz | awk -F, '{print $6}' | tail -n +2 | sort | uniq -c |
```

```
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
```

```
zcat < ~/mimic/hosp/admissions.csv.gz | awk -F, '{print $8}' | tail -n +2 | sort | uniq -c |
```

```
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
```

```
zcat < ~/mimic/hosp/admissions.csv.gz | awk -F, '{print $10}' | tail -n +2 | sort | uniq -c
```

```
244576 Medicare
173399 Private
104229 Medicaid
14006 Other
9355
 463 No charge
```

```
zcat < ~/mimic/hosp/admissions.csv.gz | awk -F, '{print $13}' | tail -n +2 | sort | uniq -c
```

```
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
```

Solution: Above are all the possible values and the number of appearances of each for 4 columns in the 'admissions.csv.gz' data file(`admission_type`, `admission_location`, `insurance`, and `race`) in decreasing order.

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?

```
zcat < ~/mimic/icu/icustays.csv.gz | awk -F, '{print $3}' | tail -n +2 | wc -l
```

```
94458
```

```
zcat < ~/mimic/icu/icustays.csv.gz | awk -F, '{print $1}' | tail -n +2 | uniq | wc -l
```

```
65366
```

Solution: There are 94458 icu stays and there are 65366 unique patients in this data file.

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.)

```
gzip -dk < ~/mimic/hosp/labevents.csv.gz > ./labevents.csv
```

```
wc -c < ~/mimic/hosp/labevents.csv.gz
```

```
 2592909134
```

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

```
 158374765

real    0m37.367s
user    0m40.555s
sys 0m9.449s
```

```
time wc -l ./labevents.csv
```

```
 158374765 ./labevents.csv

real    0m36.440s
user    0m22.355s
sys 0m5.428s
```

```
wc -c < ./labevents.csv
```

```
 18402851720
```

Solution: The compressed labevents file has 2592909134 bytes(2.41 GB) and took around 39.731 seconds to run, while the uncompressed labevents file has 9624879104 bytes(8.86 GB) and took about 16.904 seconds to run. Typically, smaller files will run faster than larger files. However, csv.gz first need to be uncompressed when using `zcat` before the `ls -l` command is run, which takes a while. The .csv file has already been uncompressed, so while it will take up more storage, it will also run faster since it only needs to run the command.

```
rm ./labevents.csv
```

The uncompressed labevents.csv file has been removed

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 a file from the web. '-nc' makes it so it will not download the file if a file with the same name already exists on my device.

```
for char in Elizabeth Jane Lydia Darcy
do
  echo $char:
  grep -w -o $char pg42671.txt | wc -l
done
```

Solution: Elizabeth is mentioned the most in this novel compared to the other names. Her name was mentioned 634 times, while Jane was mentioned 293 times, Lydia was mentioned 170 times, and Darcy was mentioned 416 times.

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

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

and

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

Solution: The first command will direct the output from one command to a file. It will either overwrite the contents of the file if it already exists or create the file and write 'hello,world' into it. The second command appends the output to a file. It will either add 'hello, world' to the file at the end if it already exists, or create a new file and write it there.

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

```
chmod u+x ~/middle.sh
```

```
ls -l ~/middle.sh
```

```
-rwxr--r--  1 palashraval  staff  137 Jan 22 09:32 /Users/palashraval/middle.sh
```

Solution: The `middle.sh` file is now executable by the owner.

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

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: `"$1"`, `"$2"`, and `"$3"` are the numerical order of the arguments. For the code '~/middle.sh pg42671.txt 20 5', `"$1` will be the first argument, which is 'pg42671.txt'. `"$2"` will be the second argument, which is "20". `"$3"` is the third argument, which is "5". The inputted arguments will be substituted in their appropriate positions in the 'middle.sh' code, so it will end up running: "head -n 20 pg42671.txt | tail -n 5". This line of code gets the first 20 lines in the 'pg42671.txt' file and then will return the final 5 lines of the first 20 lines in the file. You need the first line because it tells the respective system to use /bin/sh interpreter to run the 'middle.sh' script.

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`.

```
#cal
```

'cal' gives the calendar dates for the current month, which is January 2025.

```
#cal 2025
```

'cal 2025' gives the calendar dates of all of 2025.

```
cal 9 1752
```

```
   September 1752
Su Mo Tu We Th Fr Sa
       1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

'cal 9 1752' gives the calendar dates of September 1752. The dates 3-11 are missing in this month.

```
date
```

```
Fri Jan 24 16:05:45 PST 2025
```

'date' gives the date, time in respective timezone, and year at the time of running the code.

```
hostname
```

```
Palashs-MacBook-Air.local
```

'hostname' gives the device name.

```
arch
```

```
arm64
```

‘arch’ tells about my system’s hardware architecture.

```
uname -a
```

```
Darwin Palashs-MacBook-Air.local 23.2.0 Darwin Kernel Version 23.2.0: Wed Nov 15 21:59:33 PS
```

‘uname -a’ gives all information about my system.

```
uptime
```

```
16:05  up 203 days,  7:17, 1 user, load averages: 7.06 7.43 7.11
```

‘uptime’ shows the amount of time the system has been running. It also provides information about the current time, how many users are logged in, and the system’s load averages.

```
who am i
```

```
palashraval                  Jan 24 16:05
```

‘who am i’ shows the username of the user who executed the command and when they did.

```
who
```

```
palashraval      console      Jul  5 09:48
```

‘who’ shows more information about all the users. It gives the username, the device, and the appropriate log-in time.

```
w
```

```
16:05  up 203 days,  7:17, 1 user, load averages: 7.06 7.43 7.11
USER     TTY      FROM              LOGIN@  IDLE WHAT
palashraval console  -                05Jul24 203days -
```

'w' gives more information about all the users. It gives: USER(username), TTY(terminal), FROM, LONGIN@, IDLE, and WHAT.

```
id
```

```
uid=501(palashraval) gid=20(staff) groups=20(staff),12(everyone),61(localaccounts),79(_appse
```

'id' gives user ID, group ID, and groups that the user is a part of.

```
last | head
```

```
palashraval ttys001                          Fri Jan 24 14:32 - 14:32  (00:00)
palashraval ttys002                          Wed Jan 22 10:48 - 10:48  (00:00)
palashraval ttys001                          Thu Jan 16 18:48 - 18:48  (00:00)
palashraval ttys001                          Thu Jan 16 18:32 - 18:32  (00:00)
palashraval ttys000                          Sun Jan 12 11:37 - 11:37  (00:00)
palashraval ttys000                          Sat Jan 11 15:38 - 15:38  (00:00)
palashraval ttys000                          Sat Jan 11 15:30 - 15:30  (00:00)
palashraval ttys000                          Fri Nov  8 09:49 - 09:49  (00:00)
palashraval ttys002                          Wed Jul 17 20:33 - 20:33  (00:00)
palashraval ttys002                          Wed Jul 17 18:35 - 18:35  (00:00)
```

'last | head' shows the 10 most recent logins in the system and gives the username, terminal, date of login, and login duration.

```
echo {con,pre}{sent,fer}{s,ed}
```

```
consents consented confers confered presents presented prefers prefered
```

'echo {con,pre}{sent,fer}{s,ed}' prints all the possible combinations of words that can be formed using the 3 sets of 2 words.

```
time sleep 5
```

```
real    0m5.009s
user    0m0.000s
sys 0m0.001s
```

'time sleep 5' pauses the system for 5 seconds and provides information about how long it took to complete this command, how long it was in user mode, and how much it was in kernel mode.

```
history | tail
```

When I ran this code in my terminal, it gave me the past 10 commands I ran. Running it in this bash code box did not provide me with any results.

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`.
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.

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.

.