Biostat 203B Homework 1

Due Jan 24, 2025 @ 11:59PM

AUTHOR
Tanya Wang, 605587605

Display machine information for reproducibility:

```
sessionInfo()
```

```
R version 4.2.0 (2022-04-22)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur/Monterey 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.2/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

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

loaded via a namespace (and not attached):
 [1] htmlwidgets_1.6.2 compiler_4.2.0    fastmap_1.1.1     cli_3.6.1
 [5] tools_4.2.0       htmltools_0.5.6   rstudioapi_0.15.0 yaml_2.3.7
 [9] rmarkdown_2.25    knitr_1.44        jsonlite_1.8.7    xfun_0.50
[13] digest_0.6.33     rlang_1.1.1       evaluate_0.21
```

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.

Question 1 Done

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:

Completion Certificate: https://www.citiprogram.org/verify/?wf72ab98b-d60a-444e-9a3b-063bd8d7f7e9-67209420

Completion Report: https://www.citiprogram.org/verify/?wf72ab98b-d60a-444e-9a3b-063bd8d7f7e9-67209420

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 48
-rw-r--r--@  1 tanyawang  staff  15199 Oct 10 16:29 CHANGELOG.txt
-rw-r--r--@  1 tanyawang  staff   2518 Oct 10 17:30 LICENSE.txt
-rw-r--r--@  1 tanyawang  staff   2884 Oct 11 17:55 SHA256SUMS.txt
drwxr-xr-x@ 24 tanyawang  staff    768 Jan 17 15:05 hosp
drwxr-xr-x@ 11 tanyawang  staff    352 Jan 17 14:53 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.

Solution: I downloaded the MIMIC IV v3.1 data and it is available under "~/mimic" folder as requested.

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: Here is the content of `hosp` folder

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

```
total 12306248
-rw-r--r--@ 1 tanyawang  staff    19928140 Jun 24  2024 admissions.csv.gz
-rw-r--r--@ 1 tanyawang  staff      427554 Apr 12  2024 d_hcpcs.csv.gz
-rw-r--r--@ 1 tanyawang  staff      876360 Apr 12  2024 d_icd_diagnoses.csv.gz
-rw-r--r--@ 1 tanyawang  staff      589186 Apr 12  2024 d_icd_procedures.csv.gz
-rw-r--r--@ 1 tanyawang  staff       13169 Oct  3 09:07 d_labitems.csv.gz
-rw-r--r--@ 1 tanyawang  staff    33564802 Oct  3 09:07 diagnoses_icd.csv.gz
-rw-r--r--@ 1 tanyawang  staff     9743908 Oct  3 09:07 drgcodes.csv.gz
-rw-r--r--@ 1 tanyawang  staff   811305629 Apr 12  2024 emar.csv.gz
-rw-r--r--@ 1 tanyawang  staff   748158322 Apr 12  2024 emar_detail.csv.gz
-rw-r--r--@ 1 tanyawang  staff     2162335 Apr 12  2024 hcpcsevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff  2592909134 Oct  3 09:08 labevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff   117644075 Oct  3 09:08 microbiologyevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff    44069351 Oct  3 09:08 omr.csv.gz
-rw-r--r--@ 1 tanyawang  staff     2835586 Apr 12  2024 patients.csv.gz
-rw-r--r--@ 1 tanyawang  staff   525708076 Apr 12  2024 pharmacy.csv.gz
-rw-r--r--@ 1 tanyawang  staff   666594177 Apr 12  2024 poe.csv.gz
-rw-r--r--@ 1 tanyawang  staff    55267894 Apr 12  2024 poe_detail.csv.gz
-rw-r--r--@ 1 tanyawang  staff   606298611 Apr 12  2024 prescriptions.csv.gz
-rw-r--r--@ 1 tanyawang  staff     7777324 Apr 12  2024 procedures_icd.csv.gz
-rw-r--r--@ 1 tanyawang  staff      127330 Apr 12  2024 provider.csv.gz
-rw-r--r--@ 1 tanyawang  staff     8569241 Apr 12  2024 services.csv.gz
-rw-r--r--@ 1 tanyawang  staff    46185771 Oct  3 09:08 transfers.csv.gz
```

and the content of `icu` folder

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

```
total 8506784
-rw-r--r--@ 1 tanyawang  staff       41566 Apr 12  2024 caregiver.csv.gz
-rw-r--r--@ 1 tanyawang  staff  3502392765 Apr 12  2024 chartevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff       58741 Apr 12  2024 d_items.csv.gz
-rw-r--r--@ 1 tanyawang  staff    63481196 Apr 12  2024 datetimeevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff     3342355 Oct  3 07:36 icustays.csv.gz
-rw-r--r--@ 1 tanyawang  staff   311642048 Apr 12  2024 ingredientevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff   401088206 Apr 12  2024 inputevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff    49307639 Apr 12  2024 outputevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff    24096834 Apr 12  2024 procedureevents.csv.gz
```

These data were distributed as `gz` file because they are compressed using gzip, which will reduce the file size and make the download and storage process more efficient. Besides, gzip format includes a checksum feature that ensures the integrity of the data. It also allows people to decompress only the necessary files, saving storage space.

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

Solution:

`zcat`: This command is to display the contents of compressed files directly to the standard output.

`zless`: This command is used for viewing text files one screen at a time in a terminal. zless allows people to view gzip-compressed text files in a paginated form.

`zmore`: This command is for compressed files. It allows people to view the contents of gzip-compressed files page by page.

`zgrep`: This command is to search for patterns within gzip-compressed files without decompressing them, outputting any lines which match the specified patterns.

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
```

```
-rw-r--r--@ 1 tanyawang  staff  19928140 Jun 24  2024 /Users/tanyawang/mimic/hosp/admissions.csv.gz
-rw-r--r--@ 1 tanyawang  staff  2592909134 Oct  3 09:08 /Users/tanyawang/mimic/hosp/labevents.csv.gz
-rw-r--r--@ 1 tanyawang  staff  2835586 Apr 12  2024 /Users/tanyawang/mimic/hosp/patients.csv.gz
```

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

Solution: The bash script looks for any files starting with 'a', 'l', or 'pa' and ending with .gz.

The number of lines in each data file is:

```
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:

Here is the first few lines of `admissions.csv.gz`

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

```
subject_id,hadm_id,admittime,dischtime,deathtime,admission_type,admit_provider_id,admission_location,discharge_location,insura
nce,language,marital_status,race,edregtime,edouttime,hospital_expire_flag
10000032,22595853,2180-05-06 22:23:00,2180-05-07 17:15:00,,URGENT,P49AFC,TRANSFER FROM
HOSPITAL,HOME,Medicaid,English,WIDOWED,WHITE,2180-05-06 19:17:00,2180-05-06 23:30:00,0
10000032,22841357,2180-06-26 18:27:00,2180-06-27 18:49:00,,EW EMER.,P784FA,EMERGENCY
ROOM,HOME,Medicaid,English,WIDOWED,WHITE,2180-06-26 15:54:00,2180-06-26 21:31:00,0
10000032,25742920,2180-08-05 23:44:00,2180-08-07 17:50:00,,EW EMER.,P19UTS,EMERGENCY
ROOM,HOSPICE,Medicaid,English,WIDOWED,WHITE,2180-08-05 20:58:00,2180-08-06 01:44:00,0
10000032,29079034,2180-07-23 12:35:00,2180-07-25 17:55:00,,EW EMER.,P06OTX,EMERGENCY
ROOM,HOME,Medicaid,English,WIDOWED,WHITE,2180-07-23 05:54:00,2180-07-23 14:00:00,0
10000068,25022803,2160-03-03 23:16:00,2160-03-04 06:26:00,,EU OBSERVATION,P39NWO,EMERGENCY ROOM,,,English,SINGLE,WHITE,2160-
03-03 21:55:00,2160-03-04 06:26:00,0
10000084,23052089,2160-11-21 01:56:00,2160-11-25 14:52:00,,EW EMER.,P42H7G,WALK-IN/SELF REFERRAL,HOME HEALTH
CARE,Medicare,English,MARRIED,WHITE,2160-11-20 20:36:00,2160-11-21 03:20:00,0
10000084,29888819,2160-12-28 05:11:00,2160-12-28 16:07:00,,EU OBSERVATION,P35NE4,PHYSICIAN
REFERRAL,,Medicare,English,MARRIED,WHITE,2160-12-27 18:32:00,2160-12-28 16:07:00,0
10000108,27250926,2163-09-27 23:17:00,2163-09-28 09:04:00,,EU OBSERVATION,P40JML,EMERGENCY ROOM,,,English,SINGLE,WHITE,2163-
09-27 16:18:00,2163-09-28 09:04:00,0
10000117,22927623,2181-11-15 02:05:00,2181-11-15 14:52:00,,EU OBSERVATION,P47EY8,EMERGENCY
ROOM,,Medicaid,English,DIVORCED,WHITE,2181-11-14 21:51:00,2181-11-15 09:57:00,0
```

The number of rows in this data file, excluding the header line, is:

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

```
  546028
```

The number of hospitolizations in this file is:

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

```
uniq |
wc -l
```

```
546028
```

The same as number of rows in the file.

Peek the first few lines of `patients.csv.gz`

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

```
subject_id,hadm_id,admittime,dischtime,deathtime,admission_type,admit_provider_id,admission_location,discharge_location,insura
nce,language,marital_status,race,edregtime,edouttime,hospital_expire_flag
10000032,22595853,2180-05-06 22:23:00,2180-05-07 17:15:00,,URGENT,P49AFC,TRANSFER FROM
HOSPITAL,HOME,Medicaid,English,WIDOWED,WHITE,2180-05-06 19:17:00,2180-05-06 23:30:00,0
10000032,22841357,2180-06-26 18:27:00,2180-06-27 18:49:00,,EW EMER.,P784FA,EMERGENCY
ROOM,HOME,Medicaid,English,WIDOWED,WHITE,2180-06-26 15:54:00,2180-06-26 21:31:00,0
10000032,25742920,2180-08-05 23:44:00,2180-08-07 17:50:00,,EW EMER.,P19UTS,EMERGENCY
ROOM,HOSPICE,Medicaid,English,WIDOWED,WHITE,2180-08-05 20:58:00,2180-08-06 01:44:00,0
10000032,29079034,2180-07-23 12:35:00,2180-07-25 17:55:00,,EW EMER.,P06OTX,EMERGENCY
ROOM,HOME,Medicaid,English,WIDOWED,WHITE,2180-07-23 05:54:00,2180-07-23 14:00:00,0
10000068,25022803,2160-03-03 23:16:00,2160-03-04 06:26:00,,EU OBSERVATION,P39NWO,EMERGENCY ROOM,,,English,SINGLE,WHITE,2160-
03-03 21:55:00,2160-03-04 06:26:00,0
10000084,23052089,2160-11-21 01:56:00,2160-11-25 14:52:00,,EW EMER.,P42H7G,WALK-IN/SELF REFERRAL,HOME HEALTH
CARE,Medicare,English,MARRIED,WHITE,2160-11-20 20:36:00,2160-11-21 03:20:00,0
10000084,29888819,2160-12-28 05:11:00,2160-12-28 16:07:00,,EU OBSERVATION,P35NE4,PHYSICIAN
REFERRAL,,Medicare,English,MARRIED,WHITE,2160-12-27 18:32:00,2160-12-28 16:07:00,0
10000108,27250926,2163-09-27 23:17:00,2163-09-28 09:04:00,,EU OBSERVATION,P40JML,EMERGENCY ROOM,,,English,SINGLE,WHITE,2163-
09-27 16:18:00,2163-09-28 09:04:00,0
10000117,22927623,2181-11-15 02:05:00,2181-11-15 14:52:00,,EU OBSERVATION,P47EY8,EMERGENCY
ROOM,,Medicaid,English,DIVORCED,WHITE,2181-11-14 21:51:00,2181-11-15 09:57:00,0
```

The number of unique patients in this data file is

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

```
223452
```

which is less than the number of patients listed in the `patients.csv.gz` file.

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

```
364628
```

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:

Here is the first line of `admissions.csv.gz`

```
zcat < ~/mimic/hosp/admissions.csv.gz | head -1
```

```
subject_id,hadm_id,admittime,dischtime,deathtime,admission_type,admit_provider_id,admission_location,discharge_location,insura
nce,language,marital_status,race,edregtime,edouttime,hospital_expire_flag
```

Counts for `admission_type` (Column 6):

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

```
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
```

Counts for `admission_location` (Column 8):

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

```
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
```

Counts for `insurance` (Column 10):

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

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

Counts for `ethnicity` (Column 12):

```
zcat < ~/mimic/hosp/admissions.csv.gz |
tail -n +2 |
awk -F, '{print $12}' |
sort |
uniq -c |
sort -nr
```

```
229134 MARRIED
206232 SINGLE
56687 WIDOWED
40356 DIVORCED
13619
```

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?

Sulotion:

Here is the first line of `icusays.csv.gz`

```
zcat < ~/mimic/icu/icustays.csv.gz | head -1
```

```
subject_id,hadm_id,stay_id,first_careunit,last_careunit,intime,outtime,los
```

The number of ICU stays in this file is:

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

```
  94458
```

The number of unique patients in this data file is

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

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:

The compressed file size is:

```
ls -lh ~/mimic/hosp/labevents.csv.gz
```

```
-rw-r--r--@ 1 tanyawang  staff   2.4G Oct  3 09:08 /Users/tanyawang/mimic/hosp/labevents.csv.gz
```

The uncompressed file size is:

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

```
-rw-r--r--@ 1 tanyawang  staff   8.4G Jan 24 02:47 ./labevents.csv
```

The runtime comparison:

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

```
 158374765

real    0m21.986s
user    0m37.857s
sys 0m3.522s
```

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

```
 77671070 labevents.csv

real    1m25.571s
user    0m17.423s
sys 0m4.182s
```

Compressed files significantly reduce storage needs, which can save costs and minimize data transfer bandwidth. However, they require additional CPU resources for decompression, which will slow down data processing. Uncompressed files, while larger, provide faster access and processing speeds. Therefore, if storage is limited and CPU resources are abundant, compression will be preferable. Instead, if speed and quick data access are required, and sufficient storage is available, uncompressed data file will be better.

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
```

```
File 'pg42671.txt' already there; not retrieving.
```

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.

```
wget -nc http://www.gutenberg.org/cache/epub/42671/pg42671.txt
for char in Elizabeth Jane Lydia Darcy
do
  echo $char:
  grep -o $char pg42671.txt | wc -l
done
```

```
File ‘pg42671.txt’ already there; not retrieving.

Elizabeth:
     634
Jane:
     293
Lydia:
     170
Darcy:
     417
```

Solution:

The `wget -nc` command is used to download files from the internet. The -nc option tells wget not to download a file if it already exists in the specified location.

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

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

and

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

Solution:

`echo 'hello, world' > test1.txt` writes the string ‘hello, world’ to the file test1.txt, replacing any existing content within the file. If test1.txt does not exist, it will create the file.

`echo 'hello, world' >> test2.txt` appends the string ‘hello, world’ to the end of test2.txt. If test2.txt does not exist, it will also create the file.

The difference between them is that `>>` will append to rather than replacing the file’s content.

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:

The output shows the information of an eBook. Its release date is May 9, 2013, and its identifier is eBook #42671. It also shows that the eBook is written in English.

“$1”, “$2”, and “$3” are positional parameters used in shell scripts:

“$1” is the first argument passed to the script, which is filename her.

“$2” represents the second argument, which is end_line here. It determines how many lines from the start of the file will be considered for the future process.

“$3” is the third argument, which is num_lines, and it indicates the number of lines to be displayed from the end of the range specified by “$2”.

The first line of the shell script, #!/bin/sh, is called a shebang. It tells the operating system which interpreter to use to run the script. Without this line, the script would require the user to manually specify which interpreter to use, which is not convenient.

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` displays the calendar of the current month:

```
cal
```

```
    January 2025
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 _2_4 25
26 27 28 29 30 31
```

`cal 2025` displays the calendar of year 2025:

```
cal 2025
```

```
                            2025
      January               February              March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4                     1                     1
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   2  3  4  5  6  7  8
12 13 14 15 16 17 18   9 10 11 12 13 14 15   9 10 11 12 13 14 15
19 20 21 22 23 _2_4 25  16 17 18 19 20 21 22  16 17 18 19 20 21 22
26 27 28 29 30 31     23 24 25 26 27 28     23 24 25 26 27 28 29
                                            30 31

       April                 May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5              1  2  3   1  2  3  4  5  6  7
 6  7  8  9 10 11 12   4  5  6  7  8  9 10   8  9 10 11 12 13 14
13 14 15 16 17 18 19  11 12 13 14 15 16 17  15 16 17 18 19 20 21
20 21 22 23 24 25 26  18 19 20 21 22 23 24  22 23 24 25 26 27 28
27 28 29 30           25 26 27 28 29 30 31  29 30


        July                August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                  1  2      1  2  3  4  5  6
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   7  8  9 10 11 12 13
13 14 15 16 17 18 19  10 11 12 13 14 15 16  14 15 16 17 18 19 20
20 21 22 23 24 25 26  17 18 19 20 21 22 23  21 22 23 24 25 26 27
27 28 29 30 31        24 25 26 27 28 29 30  28 29 30
                      31

      October              November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4                     1      1  2  3  4  5  6
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   7  8  9 10 11 12 13
12 13 14 15 16 17 18   9 10 11 12 13 14 15  14 15 16 17 18 19 20
19 20 21 22 23 24 25  16 17 18 19 20 21 22  21 22 23 24 25 26 27
26 27 28 29 30 31     23 24 25 26 27 28 29  28 29 30 31
                      30
```

`cal 9 1752` displays the calendar of September 1752:

```
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
```

One special thing is that this month misses several days (9.3-9.13). According to the materials, this is because of the adoption of the Gregorian calendar by Britain and its colonies, which required an adjustment of 11 days to correct for accumulated discrepancies over centuries.

`date` displays the current date and time:

```
date
```

```
Fri Jan 24 03:16:30 PST 2025
```

`hostname` displays the name of computer I am using:

```
hostname
```

```
Tanyas-MacBook-Pro-5.local
```

`arch` outputs the architecture of the processor in the machine:

```
arch
```

```
i386
```

`uname -a` displays the kernel name, hostname, kernel release, kernel version, machine, processor, hardware platform, and operating system of the system:

```
uname -a
```

```
Darwin Tanyas-MacBook-Pro-5.local 23.2.0 Darwin Kernel Version 23.2.0: Wed Nov 15 21:54:10 PST 2023; root:xnu-
10002.61.3~2/RELEASE_X86_64 x86_64
```

`uptime` shows how long the system has been running since its last startup, along with the current number of users and load averages for the past 1, 5, and 15 minutes:

```
uptime
```

```
 3:16  up 5 days, 12:31, 2 users, load averages: 2.58 2.76 3.29
```

`who am i` shows who I am logged in with and login time:

```
who am i
```

```
tanyawang                   Jan 24 03:16
```

`who` shows the login name, terminal line, and login time of the user currently logged into the computer:

```
who
```

```
tanyawang        console      Jan 18 14:45
tanyawang        ttys011      Jan 24 02:04
```

`w` shows who is currently using the system, their activities, and the system's current load:

```
w
```

```
 3:16  up 5 days, 12:31, 2 users, load averages: 2.58 2.76 3.29
USER     TTY      FROM              LOGIN@  IDLE WHAT
tanyawang console  -                Sat14   5days -
tanyawang s011     -                 2:04    1:11 -bash
```

`id` displays the user and group information for the current user, showing user ID, group ID, and the groups the user is part of:

```
id
```

```
uid=501(tanyawang) gid=20(staff)
groups=20(staff),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),701(com.apple.share
point.group.1),33(_appstore),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.acc
ess_screensharing),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
```

`last | head` shows the last several logins in the system, how long each session lasted, and from where the login occurred:

```
last | head
```

```
tanyawang  ttys011                      Fri Jan 24 02:04   still logged in
tanyawang  ttys003                      Wed Jan 22 22:40 - 22:40  (00:00)
tanyawang  ttys003                      Tue Jan 21 23:21 - 23:21  (00:00)
tanyawang  ttys001                      Tue Jan 21 16:04 - 16:04  (00:00)
tanyawang  ttys001                      Tue Jan 21 16:02 - 16:02  (00:00)
tanyawang  ttys001                      Tue Jan 21 16:02 - 16:02  (00:00)
```

```
tanyawang  ttys002                  Tue Jan 21 15:58 - 15:58  (00:00)
tanyawang  ttys001                  Tue Jan 21 15:09 - 15:09  (00:00)
tanyawang  ttys001                  Tue Jan 21 15:07 - 15:07  (00:00)
tanyawang  ttys003                  Mon Jan 20 15:06 - 15:06  (00:00)
```

`echo {con, pre}{sent, fer}{s, ed}` uses brace expansion to generate combinations of the specified elements:

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

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

`time sleep 5` measures the time taken to execute sleep 5, which pauses the command line for 5 seconds:

```
time sleep 5
```

```
real    0m5.007s
user    0m0.001s
sys 0m0.002s
```

`history | tail` displays the last few commands I have executed in the terminal

```
history | tail
```

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`.

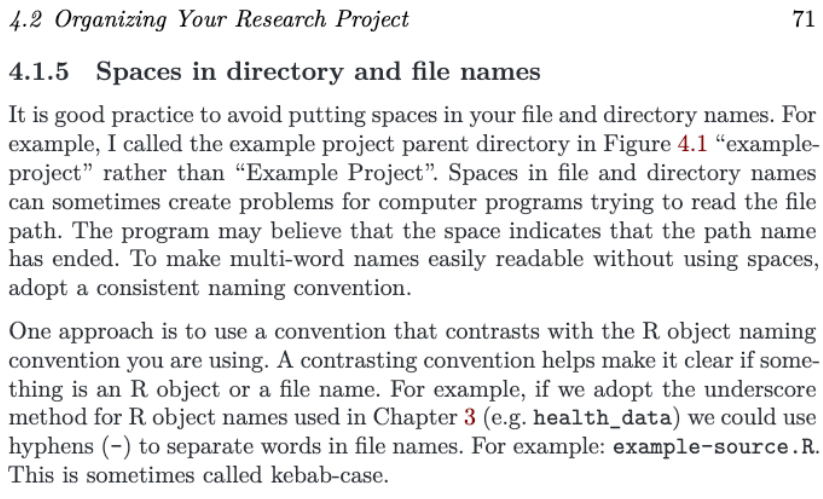
Solution: Done.

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:



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.

Section 4.1.5