Biostat 203B Homework 1

Due Jan 24, 2025 @ 11:59PM

Wenqiang Ge ID:106371961

Table of contents

Display machine information for reproducibility:

```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.1 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.12.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0

locale:
 [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
 [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
 [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
[10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C

time zone: America/Los_Angeles
tzcode source: system (glibc)

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_1.1.5       evaluate_1.0.3
```

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!

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 Report Link: https://www.citiprogram.org/verify/?kcc436586-af5d-4128-bb91-ef37f2326698-67297842

Completion Certificate Link:https://www.citiprogram.org/verify/?w7ddecc3c-d853-4bc4-95ec-c08cd3268665-67297842

sessionInfo()

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 28
-rwxrwxrwx 1 gewenqiang gewenqiang 15199 Jan 16 19:00 CHANGELOG.txt
-rwxrwxrwx 1 gewenqiang gewenqiang  2518 Jan 16 19:00 LICENSE.txt
-rwxrwxrwx 1 gewenqiang gewenqiang  2884 Jan 16 19:00 SHA256SUMS.txt
drwxrwxrwx 1 gewenqiang gewenqiang  4096 Jan 23 18:37 hosp
drwxrwxrwx 1 gewenqiang gewenqiang  4096 Jan 16 19:02 icu
-rwxrwxrwx 1 gewenqiang gewenqiang   789 Jan 16 19:00 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.

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:

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

```
total 6153128
-rwxrwxrwx 1 gewenqiang gewenqiang   19928140 Jan 16 19:00 admissions.csv.gz
```

```
-rwxrwxrwx 1 gewenqiang gewenqiang     427554 Jan 16 19:00 d_hcpcs.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang     876360 Jan 16 19:00 d_icd_diagnoses.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang     589186 Jan 16 19:00 d_icd_procedures.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang      13169 Jan 16 19:00 d_labitems.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang   33564802 Jan 16 19:00 diagnoses_icd.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang    9743908 Jan 16 19:00 drgcodes.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang  811305629 Jan 16 19:00 emar.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang  748158322 Jan 16 19:00 emar_detail.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang    2162335 Jan 16 19:00 hcpcsevents.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang       2907 Jan 16 19:00 index.html
-rwxrwxrwx 1 gewenqiang gewenqiang 2592909134 Jan 16 19:01 labevents.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang  117644075 Jan 16 19:01 microbiologyevents.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang   44069351 Jan 16 19:01 omr.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang    2835586 Jan 16 19:01 patients.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang  525708076 Jan 16 19:01 pharmacy.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang  666594177 Jan 16 19:01 poe.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang   55267894 Jan 16 19:01 poe_detail.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang  606298611 Jan 16 19:01 prescriptions.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang    7777324 Jan 16 19:01 procedures_icd.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang     127330 Jan 16 19:01 provider.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang    8569241 Jan 16 19:01 services.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang   46185771 Jan 16 19:01 transfers.csv.gz
```

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

```
total 4253396
-rwxrwxrwx 1 gewenqiang gewenqiang      41566 Jan 16 19:01 caregiver.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang 3502392765 Jan 16 19:02 chartevents.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang      58741 Jan 16 19:02 d_items.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang   63481196 Jan 16 19:02 datetimeevents.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang    3342355 Jan 16 19:02 icustays.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang       1336 Jan 16 19:02 index.html
-rwxrwxrwx 1 gewenqiang gewenqiang  311642048 Jan 16 19:02 ingredientevents.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang  401088206 Jan 16 19:02 inputevents.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang   49307639 Jan 16 19:02 outputevents.csv.gz
-rwxrwxrwx 1 gewenqiang gewenqiang   24096834 Jan 16 19:02 procedureevents.csv.gz
```

Compressed files save disk space as `.gz` files are smaller than raw `.csv` files. This will help to reduce network bandwidth usage and time required to download or transfer files. In addition, compression ensures the original data remains unaltered.

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

zcat: Outputs the contents of a compressed .gz file to the standard output without decompressing it to disk.

Solution:

zless: Allows you to view a compressed .gz file interactively, similar to less for uncompressed files.

zmore: Similar to zless, but works like more, showing the file in a paginated manner.

zgrep: Searches for a pattern inside a .gz compressed file, equivalent to grep for uncompressed files.

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
```

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/*.gz
do
  echo "File: $datafile"
  zcat "$datafile" | wc -l
done
```

```
File: /home/gewenqiang/mimic/hosp/admissions.csv.gz
546029
File: /home/gewenqiang/mimic/hosp/d_hcpcs.csv.gz
89209
File: /home/gewenqiang/mimic/hosp/d_icd_diagnoses.csv.gz
112108
File: /home/gewenqiang/mimic/hosp/d_icd_procedures.csv.gz
86424
File: /home/gewenqiang/mimic/hosp/d_labitems.csv.gz
1651
File: /home/gewenqiang/mimic/hosp/diagnoses_icd.csv.gz
6364489
File: /home/gewenqiang/mimic/hosp/drgcodes.csv.gz
761857
File: /home/gewenqiang/mimic/hosp/emar.csv.gz
42808594
File: /home/gewenqiang/mimic/hosp/emar_detail.csv.gz
87371065
File: /home/gewenqiang/mimic/hosp/hcpcsevents.csv.gz
```

```
186075
File: /home/gewenqiang/mimic/hosp/labevents.csv.gz
158374765
File: /home/gewenqiang/mimic/hosp/microbiologyevents.csv.gz
3988225
File: /home/gewenqiang/mimic/hosp/omr.csv.gz
7753028
File: /home/gewenqiang/mimic/hosp/patients.csv.gz
364628
File: /home/gewenqiang/mimic/hosp/pharmacy.csv.gz
17847568
File: /home/gewenqiang/mimic/hosp/poe.csv.gz
52212110
File: /home/gewenqiang/mimic/hosp/poe_detail.csv.gz
8504983
File: /home/gewenqiang/mimic/hosp/prescriptions.csv.gz
20292612
File: /home/gewenqiang/mimic/hosp/procedures_icd.csv.gz
859656
File: /home/gewenqiang/mimic/hosp/provider.csv.gz
42245
File: /home/gewenqiang/mimic/hosp/services.csv.gz
593072
File: /home/gewenqiang/mimic/hosp/transfers.csv.gz
2413582
```

```
for datafile in ~/mimic/icu/*.gz
do
  echo "File: $datafile"
  zcat "$datafile" | wc -l
done
```

```
File: /home/gewenqiang/mimic/icu/caregiver.csv.gz
17985
File: /home/gewenqiang/mimic/icu/chartevents.csv.gz
432997492
File: /home/gewenqiang/mimic/icu/d_items.csv.gz
4096
File: /home/gewenqiang/mimic/icu/datetimeevents.csv.gz
9979762
File: /home/gewenqiang/mimic/icu/icustays.csv.gz
94459
File: /home/gewenqiang/mimic/icu/ingredientevents.csv.gz
14253481
File: /home/gewenqiang/mimic/icu/inputevents.csv.gz
```

```
10953714
File: /home/gewenqiang/mimic/icu/outputevents.csv.gz
5359396
File: /home/gewenqiang/mimic/icu/procedureevents.csv.gz
808707
```

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 da'ta 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:

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

```
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,HO
10000032,29079034,2180-07-23 12:35:00,2180-07-25 17:55:00,,EW EMER.,P06OTX,EMERGENCY ROOM,HO
10000068,25022803,2160-03-03 23:16:00,2160-03-04 06:26:00,,EU OBSERVATION,P39NWO,EMERGENCY R
10000084,23052089,2160-11-21 01:56:00,2160-11-25 14:52:00,,EW EMER.,P42H7G,WALK-IN/SELF REFE
10000084,29888819,2160-12-28 05:11:00,2160-12-28 16:07:00,,EU OBSERVATION,P35NE4,PHYSICIAN R
10000108,27250926,2163-09-27 23:17:00,2163-09-28 09:04:00,,EU OBSERVATION,P40JML,EMERGENCY R
10000117,22927623,2181-11-15 02:05:00,2181-11-15 14:52:00,,EU OBSERVATION,P47EY8,EMERGENCY R
```

```
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 $2}' | sort | uniq | wc -l

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

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

```
546028
546028
223452
364627
```

There are 546028 lines in the adimission.csv.gz.

546028 hospitalizations; 223452 unique patients.

There are 364627 unique patients listed in the patients.csv.gz file. They are not matched.

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:

```
zcat ~/mimic/hosp/admissions.csv.gz | head -n 1 | tr ',' '\n' | nl
```

```
 1  subject_id
 2  hadm_id
 3  admittime
 4  dischtime
 5  deathtime
 6  admission_type
 7  admit_provider_id
 8  admission_location
 9  discharge_location
10  insurance
11  language
12  marital_status
13  race
14  edregtime
15  edouttime
16  hospital_expire_flag
```

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

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

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

zcat ~/mimic/hosp/admissions.csv.gz | tail -n+2 | \
  awk -F ',' '{print $13}' | 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
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
244576 Medicare
173399 Private
104229 Medicaid
 14006 Other
  9355
   463 No charge
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:

```
zcat ~/mimic/icu/icustays.csv.gz | head -n 1 | tr ',' '\n' |nl
```

```
     1  subject_id
     2  hadm_id
     3  stay_id
     4  first_careunit
     5  last_careunit
     6  intime
     7  outtime
     8  los
```

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

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

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

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

```
94458
65366
94458
94458
```

There are 94458 ICU stays and 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.)

Solution:

Comparison of file size:

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

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

```
-rwxrwxrwx 1 gewenqiang gewenqiang 2.5G Jan 16 19:01 /home/gewenqiang/mimic/hosp/labevents.c
-rwxrwxrwx 1 gewenqiang gewenqiang 18G Jan 16 19:01 /home/gewenqiang/mimic/hosp/labevents.cs
```

Comparison of run times:

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

```
158374765

real    1m50.148s
user    1m15.779s
sys 0m25.595s
```

```
time wc -l ~/mimic/hosp/labevents.csv
```

```
rm ~/mimic/hosp/labevents.csv
```

Trade off: Storage trade-off Compressing files can save a lot of disk space and reduce network transmission time when moving data. Uncompressed files require more disk space, which may not be feasible for very large datasets. Due to decompression overhead, reading compressed files (zcat) may be slightly slower. However, for linear operations such as counting lines (wc-l), this difference can usually be ignored. The operation of uncompressed files can be faster because there is no decompression step. For very large files, the speed improvement may be obvious, but the storage overhead may outweigh the benefits.

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
```

system("quarto –version") 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 $char:
  grep -ow "$char" pg42671.txt | wc -l
done
```

Solution:

wget is used to download files from the web.

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

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

and

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

Solution:

The first one creates a new file test1.txt (if it doesn't already exist) and writes 'hello, world' to it. If test1.txt already exists, this command overwrites the file, replacing its contents with 'hello, world'.

If test2.txt doesn't exist, the second command creates the file first and writes 'hello, world' to it. If test2.txt already exists, this command adds 'hello, world' to the end of the existing file without deleting its previous contents.

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
./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:

head -n 20 pg42671.txt: Extracts the first 20 lines of the file pg42671.txt.

tail -n 5: From the 20 lines produced by head, it extracts the last 5 lines.

The command runs the shell script middle.sh on the file pg42671.txt, extracting 5 lines from the end of the first 20 lines of the file.

In shell scripting, $1, $2, $3, etc., refer to the positional parameters.

$1: The first argument, in this case, the file name (pg42671.txt). $2: The second argument, representing the total number of lines to extract from the beginning of the file. $3: The third argument, representing the number of lines to extract from the result of head.

The shebang ensures that the correct shell is used to interpret the script. If we don't use it, the script might default to another shell (e.g., bash, zsh), which could lead to unexpected behavior if the script uses syntax specific to sh.

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:

```
cal
cal 2025
cal 9 1752
```

```
    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 24 25
26 27 28 29 30 31
```

```
                                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 24 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
```

```
   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
```

```
date
hostname
arch
uname -a
uptime
```

```
Thu Jan 23 20:43:26 PST 2025
gewww
x86_64
Linux gewww 5.15.167.4-microsoft-standard-WSL2 #1 SMP Tue Nov 5 00:21:55 UTC 2024 x86_64 x86
 20:43:26 up 10:48,  1 user,  load average: 1.01, 1.00, 0.69
```

```
who am i
who
w
id
last | head
echo {con,pre}{sent,fer}{s,ed}
time sleep 5
history | tail
```

```
gewenqiang pts/1        2025-01-23 14:06
 20:43:26 up 10:48,  1 user,  load average: 1.01, 1.00, 0.69
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
gewenqia pts/1    -                14:06    6:37m  0.03s  0.02s -bash
uid=1000(gewenqiang) gid=1000(gewenqiang) groups=1000(gewenqiang),4(adm),20(dialout),24(cdro
reboot   system boot  5.15.167.4-micro Thu Jan 23 14:06   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 23:10   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 21:56   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 21:15   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 20:31   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 20:26   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 18:39   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 18:39   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 18:38   still running
reboot   system boot  5.15.167.4-micro Wed Jan 22 18:37   still running
consents consented confers confered presents presented prefers prefered

real    0m5.003s
user    0m0.001s
sys 0m0.000s
```

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.

Solution:

I cloned the reporsitory and compiled the book. Here is the screenshot of Section 4.1.5 of the book.

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.

4.2 Organizing Your Research Project

Figure 4.1 gives an example of how the files in a simple reproducible research project could be organized. The project's parent directory is called *example-project*. Inside this directory are the primary knittable documents (*paper.Rmd slideshow.Rmd*, and *website.Rmd*). In addition, there is an *analysis* sub-directory with the R files to run the statistical analyses followed by a further *data* child directory.

The nested file structure allows you to use relative file paths. The knittable documents can call *analysis-1.R* with the relative path *analysis/analysis-1.R*.

In addition to the main files and sub-directories in *example-project*, you will notice files called *README.md* and *example-project.Rproj*. We'll discuss the *example-project.Rproj* file in the next section. The *README.md* file is a human readable overview of all the files in the project. It should briefly describe the project including things like its title, author(s), topic, any copyright information, and so on. It should also indicate how the folders in the project are organized and give instructions for how to reproduce the project. The README file should be in the main project folder—in our example this is called *example-project*—so that it is easy to find. If you are storing your project as a GitHub repository (see Chapter 5) and the file is called *README*, its contents will automatically be displayed on the repository's main page. If the *README* file is written using Markdown (e.g. *README.md*), it will also be properly formatted. Figure 5.2 shows an example of this.

It is good practice to dynamically include the system information for the R ses-