Biostat 203B Homework 1

Due Jan 24, 2024 @ 11:59PM

Khoa Vu 705600710

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_1.1.4       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.

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: CITI Conflicts of Interest Course - Completetion Report/Completetion Certificiate
CITI Data or Specimens Only Research - Completetion Report/Completetion Certificiate

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 kvu1702 kvu1702 15199 Jan 14 21:13 CHANGELOG.txt
-rwxrwxrwx 1 kvu1702 kvu1702  2518 Jan 14 21:13 LICENSE.txt
-rwxrwxrwx 1 kvu1702 kvu1702  2884 Jan 14 21:13 SHA256SUMS.txt
drwxrwxrwx 1 kvu1702 kvu1702  4096 Jan 19 16:56 hosp
drwxrwxrwx 1 kvu1702 kvu1702  4096 Jan 14 21:15 icu
-rwxrwxrwx 1 kvu1702 kvu1702   789 Jan 14 21:13 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 kvu1702 kvu1702   19928140 Jan 14 21:13 admissions.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702     427554 Jan 14 21:13 d_hcpcs.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702     876360 Jan 14 21:13 d_icd_diagnoses.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702     589186 Jan 14 21:13 d_icd_procedures.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702      13169 Jan 14 21:13 d_labitems.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702   33564802 Jan 14 21:13 diagnoses_icd.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702    9743908 Jan 14 21:13 drgcodes.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702  811305629 Jan 14 21:13 emar.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702  748158322 Jan 14 21:13 emar_detail.csv.gz
```

```
-rwxrwxrwx 1 kvu1702 kvu1702    2162335 Jan 14 21:13 hcpcsevents.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702       2907 Jan 14 21:13 index.html
-rwxrwxrwx 1 kvu1702 kvu1702 2592909134 Jan 14 21:13 labevents.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702  117644075 Jan 14 21:14 microbiologyevents.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702   44069351 Jan 14 21:14 omr.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702    2835586 Jan 14 21:14 patients.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702  525708076 Jan 14 21:14 pharmacy.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702  666594177 Jan 14 21:14 poe.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702   55267894 Jan 14 21:14 poe_detail.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702  606298611 Jan 14 21:14 prescriptions.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702    7777324 Jan 14 21:14 procedures_icd.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702     127330 Jan 14 21:14 provider.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702    8569241 Jan 14 21:14 services.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702   46185771 Jan 14 21:14 transfers.csv.gz
```

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

```
total 4253396
-rwxrwxrwx 1 kvu1702 kvu1702      41566 Jan 14 21:14 caregiver.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702 3502392765 Jan 14 21:14 chartevents.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702      58741 Jan 14 21:15 d_items.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702   63481196 Jan 14 21:15 datetimeevents.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702    3342355 Jan 14 21:15 icustays.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702       1336 Jan 14 21:15 index.html
-rwxrwxrwx 1 kvu1702 kvu1702  311642048 Jan 14 21:15 ingredientevents.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702  401088206 Jan 14 21:15 inputevents.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702   49307639 Jan 14 21:15 outputevents.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702   24096834 Jan 14 21:15 procedureevents.csv.gz
```

These files are distributed as `.csv.gz` files because gzip compression ensures that the large data files in MIMIC-IV v3.1 are transferred efficiently between users.

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

Solution: The `z` in front of Bash commands such as `cat`, `less`, `more`, and `grep` means that the file is uncompressed before the Bash command is executed. `zcat`: Uncompresses the file before printing its contents.

`zless`: Uncompresses the file and browses a text file by screen through actions such as upwards and downward scrolling.

`zmore`: Uncompresses the file and also acts as a pager. `more` has less functionality than `less`, only allowing downward scrolling. `more` reads the whole file, making it slower in reading files than `less`. Exit `more` by pressing the q key and scroll/page using the spacebar.

`zgrep`: Uncompresses a file and prints lines matching an expression.

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
```

```
-rwxrwxrwx 1 kvu1702 kvu1702 19928140 Jan 14 21:13 /home/kvu1702/mimic/hosp/admissions.csv.gz
-rwxrwxrwx 1 kvu1702 kvu1702 2592909134 Jan 14 21:13 /home/kvu1702/mimic/hosp/labevents.csv.g
-rwxrwxrwx 1 kvu1702 kvu1702 2835586 Jan 14 21:14 /home/kvu1702/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 output of the above Bash script prints all files in /mimic/hosp/ that starts with a, l, or pa.

```
for datafile in ~/mimic/hosp/{a,l,pa}*.gz
do
  # For each file that starts with a, l, or pa, count the number of lines with
  # wc and the -l option for lines.
  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:

```
# Printing the first 10 rows in admissions.csv.gz
echo 'The first ten lines of admissions.csv.gz are:'
zcat < ~/mimic/hosp/admissions.csv.gz | head -10
echo ''

# Counting the number of lines minus the header using wc -l.
# The n- option for tail displays the lines beginning from the n-th line.

echo 'The total count of lines in admissions.csv.gz, minus the header, are:'
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | wc -l
echo ''

# Counting the number of unique values in the second column, hadm_id, minus the
# header, using the Bash command awk.
# awk scans files line by line. The -F option in awk determines the delimiter
# for the file, and {print $2} prints the value after the second delimiter,
# representing the second column.
# For a .csv file, this delimiter is a ','
# sort is a bash command that sorts values. We sort because the uniq bash
# command only removes and identifies duplicates directly adjacent.
# The bash command uniq with option -c counts the occurrences.
# We sort again using -n, an option that sorts numerically, and -r, an option
# for reverse order (descending instead of ascending).
# Using wc -l to count the number of rows counted by uniq. Repeated values are
# compressed to one line, a count followed by the counted instance.

echo 'The count of unique hadm_id in admissions.csv.gz, minus the header, are:'
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F ',' '{print $2}' |
sort | uniq -c | wc -l
echo ''

# Counting the number of unique patients/subject_id, the first column in
# admissions.csv.gz.
echo 'The count of unique patients by subject_id in admissions.csv.gz, are:'
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F ',' '{print $1}' |
sort | uniq -c | wc -l
echo ''

#subject_id is the first column in patients.csv.gz.
echo 'The total count of unique patients by subject_id in patients.csv.gz, are:'
zcat < ~/mimic/hosp/patients.csv.gz | tail -n +2 | awk -F ',' '{print $1}' |
sort | uniq -c | wc -l
```

```
echo ''
```

```
The first ten lines of admissions.csv.gz are:
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
10000068,25022803,2160-03-03 23:16:00,2160-03-04 06:26:00,,EU OBSERVATION,P39NWO,EMERGENCY RO
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 RE
10000108,27250926,2163-09-27 23:17:00,2163-09-28 09:04:00,,EU OBSERVATION,P40JML,EMERGENCY RO
10000117,22927623,2181-11-15 02:05:00,2181-11-15 14:52:00,,EU OBSERVATION,P47EY8,EMERGENCY RO

The total count of lines in admissions.csv.gz, minus the header, are:
546028

The count of unique hadm_id in admissions.csv.gz, minus the header, are:
546028

The count of unique patients by subject_id in admissions.csv.gz, are:
223452

The total count of unique patients by subject_id in patients.csv.gz, are:
364627
```

We find that there are 546028 rows in `admissions.csv.gz` minus the header file. Of those 546028, 223452 unique patients are hospitalized. However, there are 364627 patients in `patients.csv.gz`., meaning that some patients did not get hospitalized.

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:

```
#admission_type is the sixth column in admissions.csv.gz.
echo 'The possible values for admission types and their frequencies are:'
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F ',' '{print $6}' |
sort | uniq -c | sort -nr
echo ''
```

```
#admission_location is the eighth column in admissions.csv.gz.
echo 'The possible values for admission location and their frequencies are:'
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F ',' '{print $8}' |
sort | uniq -c | sort -nr
echo ''

#insurance is the tenth column in admissions.csv.gz.
echo 'The possible values for insurance and their frequencies are:'
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F ',' '{print $10}' |
sort | uniq -c | sort -nr
echo ''

#ethnicity/race is the tenth column in admissions.csv.gz.
echo 'The possible values for ethnicity/race and their frequencies are:'
zcat < ~/mimic/hosp/admissions.csv.gz | tail -n +2 | awk -F ',' '{print $13}' |
sort | uniq -c | sort -nr
echo ''
```

```
The possible values for admission types and their frequencies are:
 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

The possible values for admission location and their frequencies are:
 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

The possible values for insurance and their frequencies are:
 244576 Medicare
 173399 Private
 104229 Medicaid
  14006 Other
   9355
    463 No charge

The possible values for ethnicity/race and their frequencies are:
 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 `icustays.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:

```
# Printing the first 5 rows in icustays.csv.gz.
echo 'The first 5 lines of icustays.csv.gz are:'
zcat < ~/mimic/icu/icustays.csv.gz | head -5
echo ''

#stay_id is the third column of icustays.csv.gz.
echo 'The total count of ICU stays by stay_id in icustays.csv.gz are:'
zcat < ~/mimic/icu/icustays.csv.gz | tail -n +2 | awk -F ',' '{print $3}' |
sort | uniq -c | sort -nr | wc -l
echo ''

#subject_id is the third column of icustays.csv.gz
echo 'The total count of unique patients by subject_id in icustays.csv.gz, are:'
zcat < ~/mimic/icu/icustays.csv.gz | tail -n +2 | awk -F ',' '{print $1}' |
sort | uniq -c | sort -nr | wc -l
echo ''
```

```
The first 5 lines of icustays.csv.gz are:
subject_id,hadm_id,stay_id,first_careunit,last_careunit,intime,outtime,los
10000032,29079034,39553978,Medical Intensive Care Unit (MICU),Medical Intensive Care Unit (MI
10000690,25860671,37081114,Medical Intensive Care Unit (MICU),Medical Intensive Care Unit (MI
10000980,26913865,39765666,Medical Intensive Care Unit (MICU),Medical Intensive Care Unit (MI
10001217,24597018,37067082,Surgical Intensive Care Unit (SICU),Surgical Intensive Care Unit

The total count of ICU stays by stay_id in icustays.csv.gz are:
94458

The total count of unique patients by subject_id in icustays.csv.gz, are:
65366
```

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:

```
#Running wc on the compressed file labevents.csv.gz.
zcat < ~/mimic/hosp/labevents.csv.gz | wc -l

#Unzipping the compressed file labevents.csv.gz.
gzip -dk ~/mimic/hosp/labevents.csv.gz

#Running wc on the uncompressed file labevents.csv
wc -l ~/mimic/hosp/labevents.csv

#Deleting the file afterwards
rm ~/mimic/hosp/labevents.csv
```

```
158374765
158374765 /home/kvu1702/mimic/hosp/labevents.csv
```

The tradeoff between using compressed versus uncompressed files is that they take up much less storage space than their compressed counterparts. However, it is faster to access the raw, large, uncompressed files versus their uncompressed counterparts because you must first uncompress the compressed file before accessing it. The uncompression algorithm can be lengthy depending on your computer specifications, file size, and software.

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: wget is a Bash command that can download files from websites. The -nc option prevents the file from getting clobbered. This means it is not overwritten if it already exists in the directory and is renamed with an incrementing counter based on how many copies have been installed. In *Pride and Prejudice*, Elizabeth is mentioned 634 times, Jane is mentioned 293 times, Lydia is mentioned 170 times, and Darcy is mentioned 417 times, making you correct.

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

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

and

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

Solution: `>` Overwrites the content in the file following the symbol with the text preceding the symbol. `>>` Appends the text preceding the symbol to the end of the content in the file following the symbol.

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 +x middle.sh
./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: This output is the last five of the first 20 lines in the full text of *Pride and Prejudice* by Jane Austen from Project Gutenberg. `"$1"`, `"$2"`, and `"$3"` are input variables into the shell script in the order typed in the command line. `"$1"` is the file path. `"$2"` is the number of lines to take the head of. `"$3"` is the number of lines to take the tail end.

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
```

```
    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
```

cal is a Bash command that displays the present month's calendar dates.

```
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 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
```

By adding the year, cal displays all calendar dates for the year.

```
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
```

By adding the month and the year to cal, cal displays the calendar date for the specified month and year. However, for this specific month and year, we are missing the dates of 3-13.

```
date
```

```
Tue Jan 21 15:34:21 PST 2025
```

date is a Bash command that displays information about the current date.

```
hostname
```

```
DESKTOP-7TAH72S
```

hostname is a Bash command that displays current host/computer information.

```
arch
```

```
x86_64
```

arch is a Bash command that displays information about the architecture of the current host/computer.

```
uname -a
```

```
Linux DESKTOP-7TAH72S 5.15.167.4-microsoft-standard-WSL2 #1 SMP Tue Nov 5 00:21:55 UTC 2024 x
```

uname is a Bash command that displays information about the current operating system. The -a option adds more information, including system architecture, hostname, etc.

```
uptime
```

```
 15:34:21 up  2:15,  1 user,  load average: 1.08, 1.02, 0.73
```

uptime is a Bash command that displays the system uptime, number of users, and load times for the past 1, 5, and 15 minutes.

```
whoami
```

```
kvu1702
```

whoami is a Bash command that displays the user's username.

```
who
```

```
kvu1702  pts/1        2025-01-21 04:19
```

who is a Bash command that displays the number and usernames of currently logged-in users and the time of the last system boot.

```
w
```

```
 15:34:21 up  2:15,  1 user,  load average: 1.08, 1.02, 0.73
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
kvu1702  pts/1    -                04:19   11:15m  0.01s  0.01s -bash
```

w is a Bash command displaying information about current users, including computer usage, running programs, usernames, load times, etc. It displays information from whoami and who as well.

```
id
```

```
uid=1000(kvu1702) gid=1000(kvu1702) groups=1000(kvu1702),4(adm),20(dialout),24(cdrom),25(flo
```

id is a Bash command that displays the user and group names and IDs.

```
last | head
```

```
reboot   system boot  5.15.167.4-micro Tue Jan 21 04:19   still running
reboot   system boot  5.15.167.4-micro Tue Jan 21 02:42   still running
reboot   system boot  5.15.167.4-micro Sun Jan 19 15:38   still running
reboot   system boot  5.15.167.4-micro Thu Jan 16 15:37   still running
reboot   system boot  5.15.167.4-micro Thu Jan 16 15:04   still running
reboot   system boot  5.15.167.4-micro Wed Jan 15 23:18   still running
reboot   system boot  5.15.167.4-micro Wed Jan 15 12:16   still running
reboot   system boot  5.15.167.4-micro Tue Jan  7 16:46   still running
reboot   system boot  5.15.153.1-micro Tue Oct  1 13:45   still running
reboot   system boot  5.15.153.1-micro Tue Oct  1 13:45   still running
```

last is a Bash command that displays each user's previous login information. By default, head prints the first ten lines or the first ten logins.

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

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

This command prints all words matching a combination of con or pre, sent or fer, and s or ed.

```
time sleep 5
```

```
real    0m5.002s
user    0m0.000s
sys 0m0.001s
```

This command puts the console to sleep for 5 seconds.

```
history | tail
```

history is a bash command that displays the command history in the console. By default, tail prints the last ten lines or ten most recent commands.

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:

Section 4.1.5 of Reproducible Research with R and RStudio