Biostat 203B Homework 1

Due Jan 24, 2025 @ 11:59PM

AUTHOR
Julie Lee and 806409381

Display machine information for reproducibility:

```
sessionInfo()
```

```
R version 4.3.2 (2023-10-31)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.0

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;  LAPACK version 3.11.0

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] htmlwidgets_1.6.4 compiler_4.3.2    fastmap_1.1.1     cli_3.6.2
 [5] tools_4.3.2       htmltools_0.5.7   rstudioapi_0.15.0 yaml_2.3.8
 [9] rmarkdown_2.29    knitr_1.45        jsonlite_1.8.8    xfun_0.50
[13] digest_0.6.33     rlang_1.1.2       evaluate_0.23
```

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). **Completed.**

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. **Completed.**

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. **Completed.**

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. **Completed.**

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: Here is the Completion Report of my CITI Training.

Here is the Completion Certification of my CITI Training.

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).

(3.1) Solution: I downloaded the MIMIC IV v3.1 data and it is available under `~/mimic' folder as requested.

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

```
total 35943120
-rw-r--r--@  1 julielee  staff        15199 Oct 10 16:29 CHANGELOG.txt
-rw-r--r--@  1 julielee  staff         2518 Oct 10 17:30 LICENSE.txt
-rw-r--r--@  1 julielee  staff         2884 Oct 11 17:55 SHA256SUMS.txt
drwxr-xr-x@ 24 julielee  staff          768 Jan 23 17:49 hosp
drwxr-xr-x@ 11 julielee  staff          352 Jan 16 00:51 icu
-rw-r--r--   1 julielee  staff  18402851720 Jan 19 21:09 labevents.csv
```

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.

(3.2) Solution: Here is the content of 'hosp' folder:

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

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

Here is the content of 'icu' folder:

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

```
total 8506784
-rw-r--r--@ 1 julielee  staff       41566 Apr 12  2024 caregiver.csv.gz
-rw-r--r--@ 1 julielee  staff  3502392765 Apr 12  2024 chartevents.csv.gz
-rw-r--r--@ 1 julielee  staff       58741 Apr 12  2024 d_items.csv.gz
-rw-r--r--@ 1 julielee  staff    63481196 Apr 12  2024 datetimeevents.csv.gz
-rw-r--r--@ 1 julielee  staff     3342355 Oct  3 07:36 icustays.csv.gz
```

```
-rw-r--r--@ 1 julielee  staff   311642048 Apr 12  2024 ingredientevents.csv.gz
-rw-r--r--@ 1 julielee  staff   401088206 Apr 12  2024 inputevents.csv.gz
-rw-r--r--@ 1 julielee  staff    49307639 Apr 12  2024 outputevents.csv.gz
-rw-r--r--@ 1 julielee  staff    24096834 Apr 12  2024 procedureevents.csv.gz
```

These data were distributed as ".csv.gz" files instead of .csv files because the ".gz" extension insinuates that the CSV data has been compressed due to its large size. When compressing the CSV file, the underlying data format is still CSV but users are more easily able to open and the read the data in any CSV reader even after decompression. The data that we downloaded (Mimic) is an extremely large file and beig able to compress it using the Gzip algorithm reduces the file size and makes it more efficient to store.

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

(3.3) Solution: The Bash command `zcat` is equivalent to the cat for gzip-compressed files. It is used to output the contents of the .gz file without creating an uncompressed version of the file.

The Bash command `zless` is similar to the less command. It allows users to view the contents of the compressed files without having to decompress or extract them.

The Bash command `zmore` is used to display compressed or plan text files one screen at a time without first decompressing them.

The Bash command `zgrep` is used to search through a combination of uncompressed and compressed files without explicitly decompressing them first.

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

(3.4) Solution: The output of the following bash script is down below. It is a list of the compressed (.gz) files that start with "a", "l", and "pa" in ~/mimic/hosp/. More specifically, the compressed files listed correspond to admissions, labevents, and patients. Information regarding the compressed files correspond to the file permissions, file sizes, dates and times, and file paths.

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

```
-rw-r--r--@ 1 julielee  staff  19928140 Jun 24  2024
/Users/julielee/mimic/hosp/admissions.csv.gz
-rw-r--r--@ 1 julielee  staff  2592909134 Oct  3 09:08
/Users/julielee/mimic/hosp/labevents.csv.gz
-rw-r--r--@ 1 julielee  staff  2835586 Apr 12  2024
/Users/julielee/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 number of lines in each data file for admission.csv.gv, labevents.csv.gz, and patients.csv.gz are as follows:

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

```
/Users/julielee/mimic/hosp/admissions.csv.gz
  546029
/Users/julielee/mimic/hosp/labevents.csv.gz
 158374765
/Users/julielee/mimic/hosp/patients.csv.gz
  364628
```

In summary, there are 546029 lines in the file 'admissions.csv.gz", there are 158374765 lines in the file 'labevents.csv.gz', and there are 364628 lines in the 'patients.csv.gz' data file. In the following code, we iterate through all the .gz files in the ~/mimic/hosp directory that matches the "a, l, pa" prefixes. The "gzcat" command decompresses each file and outputs its content to standard output. Afterwards, the "wc -l" command prints the number of lines in each iterated file.

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.)

(3.5) Solution:

The first few lines of admissions.csv.gz are:

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

```
subject_id,hadm_id,admittime,dischtime,deathtime,admission_type,admit_provider_id,admissi
on_location,discharge_location,insurance,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
```

If each hadm_id identifies a hospitalization, the number of hospitalizations in this data file is the same as the number of rows in the file (546028). Therefore, we are able to conclude that each hospital admission is unique and independent as none of the entries repeat.

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

```
546028
```

The number of unique patients there are in this data file (admissions) is 223452 patients (identified by subject_id).

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

```
223452
```

Now, let's first peek the first few lines of the patient.csv.gz file:

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

```
subject_id,gender,anchor_age,anchor_year,anchor_year_group,dod
10000032,F,52,2180,2014 - 2016,2180-09-09
10000048,F,23,2126,2008 - 2010,
```

```
10000058,F,33,2168,2020 - 2022,
10000068,F,19,2160,2008 - 2010,
10000084,M,72,2160,2017 - 2019,2161-02-13
10000102,F,27,2136,2008 - 2010,
10000108,M,25,2163,2014 - 2016,
10000115,M,24,2154,2017 - 2019,
10000117,F,48,2174,2008 - 2010,
```

The number of patients listed in this patients.csv.gz file is 364627 patients. Recall that the number of unique patients that are in the admission.csv.gz file(identified by subject_id) is 223452 patients. These 2 numbers do not match as the number of patients listed in the patients.csv.gv file is greater than the number of unique patients in the admissions.csv.gz file.

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

```
364627
```

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

(3.6) Solution

Start with listing out all the variable names (header row):

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

```
subject_id,hadm_id,admittime,dischtime,deathtime,admission_type,admit_provider_id,admissi
on_location,discharge_location,insurance,language,marital_status,race,edregtime,edouttime
,hospital_expire_flag
```

The possible unique values and their corresponding counts taken by the variable 'admission_type' are:

```
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
```

The possible unique values and their corresponding counts taken by the variable ‘admission_location’ are:

```
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
```

The possible unique values and their corresponding counts taken by the variable ‘insurance’ are:

```
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
```

The possible values taken by the varaible ‘ethnicity’ are:

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

```
sort |
uniq -c |
sort -nr
```

```
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?

(3.7) Solution Let's first take a peek of the first few rows of the 'icusays.csv.gv'

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

```
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
(MICU),2180-07-23 14:00:00,2180-07-23 23:50:47,0.4102662037037037
10000690,25860671,37081114,Medical Intensive Care Unit (MICU),Medical Intensive Care Unit
(MICU),2150-11-02 19:37:00,2150-11-06 17:03:17,3.8932523148148146
10000980,26913865,39765666,Medical Intensive Care Unit (MICU),Medical Intensive Care Unit
```

```
(MICU),2189-06-27 08:42:00,2189-06-27 20:38:27,0.4975347222222222
10001217,24597018,37067082,Surgical Intensive Care Unit (SICU),Surgical Intensive Care
Unit (SICU),2157-11-20 19:18:02,2157-11-21 22:08:00,1.1180324074074075
10001217,27703517,34592300,Surgical Intensive Care Unit (SICU),Surgical Intensive Care
Unit (SICU),2157-12-19 15:42:24,2157-12-20 14:27:41,0.948113425925926
10001725,25563031,31205490,Medical/Surgical Intensive Care Unit
(MICU/SICU),Medical/Surgical Intensive Care Unit (MICU/SICU),2110-04-11 15:52:22,2110-04-
12 23:59:56,1.338587962962963
10001843,26133978,39698942,Medical/Surgical Intensive Care Unit
(MICU/SICU),Medical/Surgical Intensive Care Unit (MICU/SICU),2134-12-05 18:50:03,2134-12-
06 14:38:26,0.8252662037037037
10001884,26184834,37510196,Medical Intensive Care Unit (MICU),Medical Intensive Care Unit
(MICU),2131-01-11 04:20:05,2131-01-20 08:27:30,9.17181712962963
10002013,23581541,39060235,Cardiac Vascular Intensive Care Unit (CVICU),Cardiac Vascular
Intensive Care Unit (CVICU),2160-05-18 10:00:53,2160-05-19 17:33:33,1.314351851851852
```

The number of ICU stays (as identified by 'stay_id') is in the data file is 94458 ICU stays.

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

```
94458
```

The number of unique patients (identified by 'subject_id') in the data file is 65366 patients.

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

```
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.)

(3.8) Solution: The compressed gz file (`labevents.csv.gz`) has a size of 2.4G. The uncompressed file ('labevents.csv') has a size of 17G.

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

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

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

```
-rw-r--r--  1 julielee  staff    17G Jan 23 17:54
/Users/julielee/mimic/hosp/labevents.csv
```

The run times of `zcat < ~/mimic/labevents.csv.gz | wc -l` are displayed below:

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

```
 158374765

real    0m41.934s
user    0m46.350s
sys 0m11.309s
```

The run time of `wc -l labevents.csv` are displayed below:

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

```
 158374765 /Users/julielee/mimic/hosp/labevents.csv

real    0m39.056s
user    0m29.488s
sys 0m5.753s
```

From the outputs above, we observe that the compressed .gz file is significantly smaller in size compared to the uncompressed file. However, running the same command to count the number of lines takes longer on the compressed file than on the uncompressed file. This highlights the trade-offs between storage and speed: while compressed files are more efficient in saving disk space, they require decompression, which increases processing time and computational resource usage. For large datasets, operations like wc -l are slower on compressed files compared to uncompressed ones, due to the added overhead of decompression (computational effort and use of more system resources such as CPU and memory).

We erase the 'labevents.csv' file before proceeding on the next exercise.

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

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. **Done**

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

Explain what `wget -nc` does.

(4.1) Solution: The 'wget -nc' command is used to download a file from the internet, retrieving files using HTTP, HTTPS, and FTP protocols. The '-nc' flag is used to prevent the download of the file if it already exists in the current directory. If a file with the same name is already downloaded, then the wget command will skip downloading the file.

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.

Explanation of Code: By utilizing the grep -o, we are able to search for all the occurrences that the exact value of "$char" takes on in the pg42671.txt file. Every time an occurrence (in this case, a specific name) is detected, it outputs the specific name on a new line. The wc -l counts the number of lines or number of occurrences (names) there are in the entire pg42671.txt file.

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

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

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

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

(4.2) Solution: The first command writes the string "hello, world" to the file test1.txt. If the file test1.txt already exists, then the contents of the existing file will be overwritten by the contents of the new text. However, if text1.txt file does not exist, then it will be created and the text will be written. The second command appends the string "hello, world" to the file test2.txt. In other words, if the file test2.txt already existed, the new string will be added to the end of the existing content. However, if the file test2.txt does not exist, it will be created and the text will be written to it. The >> operator is in indicator that the new content appends to the existing content in the file.

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

and

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

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

(4.3) Solution:

I manually created the middle.sh file using the text editor nano and included middle.sh in the submission folder on Git.

```
#!/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 the following command to test the script: (Note that instead of 20 5, I showcased the results for 18 6 to better understand the content of these lines)

```
chmod a+x middle.sh
./middle.sh pg42671.txt 18 6
```

```
Author: Jane Austen

Editor: R. W. Chapman

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

Explain the output.

Solution: The output showcases that the author of the book “Price and Prejudice is Jane Austen, the Editor is R.W Chapman, and the release date was May 9, 2013 (eBook $42671). We extracted lines 13-18 from the original text.

Explain the meaning of `"$1"`, `"$2"`, and `"$3"` in this shell script.

Solution: In this shelll script, the `"$1"` refers to the first argument passed to the script, which is the pg42671.txt file. The `"$2"` refers to the second argument passed to the script, which is the ending line number (line number 18). The `"$3"` refers to the third argument passed to the script, which is the num_lines or the number of lines to extract before and including the 18th line. In this case, we extract 6 lines from line number 18.

Why do we need the first line of the shell script?

Solution: The first line of the script is a shebang, which specifies the interpreter used to execute the script. It tells the system which shell or program should run the script. In this case, the shebang #!/bin/sh directs the system to use the Bourne shell as the interpreter.

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" The "cal" command showcases the calendar for 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 _2_3 24 25
26 27 28 29 30 31
```

The "cal 2025" command showcases the calendar for the year 2025 (for all 12 months).

```
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 _2_3 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
```

The “cal 9 1752” command showcases the calendar for the month of September in the year of 1752. However, there is unusual behavior in this command. We are able to see that the calendar is missing 11 days (September 3 - September 13, 1752.) This is because the Gregorian calendar was adopted in 1752 and the Julian calendar was replaced.

```
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
```

The “date” command showcases the current date and time in PST.

```
date
```

```
Thu Jan 23 17:55:23 PST 2025
```

The “hostname” command showcases the hostname of the current system.

```
hostname
```

```
Julies-Laptop-2.local
```

The “arch” command showcases the hardware architecture of the current system.

```
arch
```

```
arm64
```

The “uname -a” command showcases the kernel name/version, system information, hostname, architecture, and OS of the current machine.

```
uname -a
```

```
Darwin Julies-Laptop-2.local 23.0.0 Darwin Kernel Version 23.0.0: Fri Sep 15 14:42:57 PDT
2023; root:xnu-10002.1.13~1/RELEASE_ARM64_T8112 arm64
```

The uptime command showcases how long the machine has been running, system load averages, and the number of users logged in.

```
uptime
```

```
17:55  up 4 days,  3:47, 3 users, load averages: 5.74 5.18 5.35
```

The who am i command showcases the current user logged in (username) and login time.

```
who am i
```

```
julielee                     Jan 23 17:55
```

The who command showcases the list of all currently logged-in users.

```
who
```

```
julielee         ttys000      Jan 23 17:11
julielee         console      Jan 19 23:57
julielee         ttys001      Jan 23 17:11
```

The w command showcases information regarding the users who are currently logged into the system and their activities. It includes system information including the current time, how long the system has been running, and the number of users logged in. It includes user information including usernmae, terminal decive, login time, idle time, and current directory used.

```
w
```

```
17:55  up 4 days,  3:47, 3 users, load averages: 5.74 5.18 5.35
USER     TTY      FROM              LOGIN@  IDLE WHAT
julielee s000     -                 17:11      44 -zsh
julielee console  -                 Sun23   3days -
julielee s001     -                 17:11      24 -zsh
```

The id command showcases the user ID, groupID, user name, and the groups for the current user.

```
id
```

```
uid=501(julielee) gid=20(staff)
groups=20(staff),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_appserver
adm),98(_lpadmin),701(com.apple.sharepoint.group.1),33(_appstore),100(_lpoperator),204(_d
```

```
eveloper),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.access_screenshari
ng),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
```

The last | head command showcases the most recent logins (first 10 logg-in users).

```
last | head
```

```
julielee   ttys001                      Thu Jan 23 17:11   still logged in
julielee   ttys000                      Thu Jan 23 17:11   still logged in
julielee   ttys000                      Thu Jan 23 13:23 - 13:23  (00:00)
julielee   ttys000                      Wed Jan 22 21:24 - 21:24  (00:00)
julielee   ttys000                      Wed Jan 22 21:08 - 21:08  (00:00)
julielee   ttys000                      Wed Jan 22 20:33 - 20:33  (00:00)
julielee   ttys000                      Wed Jan 22 20:31 - 20:31  (00:00)
julielee   ttys000                      Wed Jan 22 20:28 - 20:28  (00:00)
julielee   ttys000                      Wed Jan 22 18:08 - 18:08  (00:00)
julielee   ttys000                      Wed Jan 22 17:11 - 17:11  (00:00)
```

The echo {con, pre} {sent,fer}{s,ed} command showcases different combinations of words.

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

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

The time sleep 5 command showcases how long the operation takes to execute after the system pasues for 5 seconds.

```
time sleep 5
```

```
real    0m5.009s
user    0m0.000s
sys 0m0.002s
```

The history | tail command showcases the last 10 commands that were executed in the terminal.

```
history | tail

#My following output for this command on my terminal:
  #158  date
  #159  hostname
  #160  arch
  #161  uname
  #162  uptime
  #163  who am i
  #164  history | tail
  #165  wget
  #166  wget -V
  #167  cal  9 1752
```

Q6. Book

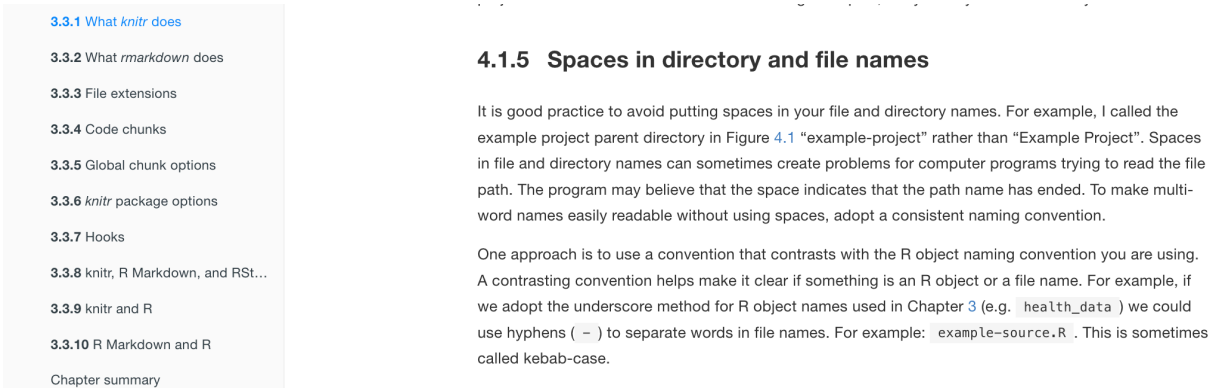
1. Git clone the repository https://github.com/christophergandrud/Rep-Res-Book for the book *Reproducible Research with R and RStudio* to your local machine. Do **not** put this repository within your homework repository `biostat-203b-2025-winter`.

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: Here is the screenshot of Section 4.1.5 of the book:



This is the End of Homework #1