Biostat 203B Homework 2

Due Feb 7, 2025 @ 11:59PM

Kiana Mohammadinik and 205928003

Table of contents

Display machine information for reproducibility:

```
sessionInfo()
```

```
R version 4.3.0 (2023-04-21)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS 14.4.1

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

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Los_Angeles
```

```
tzcode source: internal

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

loaded via a namespace (and not attached):
 [1] compiler_4.3.0  fastmap_1.1.1   cli_3.6.2       tools_4.3.0
 [5] htmltools_0.5.7 rstudioapi_0.14 yaml_2.3.8      rmarkdown_2.29
 [9] knitr_1.45      jsonlite_1.8.8  xfun_0.50       digest_0.6.34
[13] rlang_1.1.3     evaluate_0.23
```

Load necessary libraries (you can add more as needed).

```
library(arrow)
```

```
Warning: package 'arrow' was built under R version 4.3.3
```

```

Attaching package: 'arrow'

The following object is masked from 'package:utils':

    timestamp
```

```
library(data.table)
library(duckdb)
```

```
Warning: package 'duckdb' was built under R version 4.3.3

Loading required package: DBI
```

```
library(memuse)
```

```
Warning: package 'memuse' was built under R version 4.3.3
```

```
library(pryr)
```

```

Attaching package: 'pryr'
```

```
The following object is masked from 'package:data.table':

    address
```

```
library(R.utils)
```

```
Warning: package 'R.utils' was built under R version 4.3.1

Loading required package: R.oo

Warning: package 'R.oo' was built under R version 4.3.1

Loading required package: R.methodsS3

R.methodsS3 v1.8.2 (2022-06-13 22:00:14 UTC) successfully loaded. See ?R.methodsS3 for help.

R.oo v1.26.0 (2024-01-24 05:12:50 UTC) successfully loaded. See ?R.oo for help.


Attaching package: 'R.oo'

The following object is masked from 'package:R.methodsS3':

    throw

The following objects are masked from 'package:methods':

    getClasses, getMethods

The following objects are masked from 'package:base':

    attach, detach, load, save

R.utils v2.12.3 (2023-11-18 01:00:02 UTC) successfully loaded. See ?R.utils for help.


Attaching package: 'R.utils'
```

```
The following object is masked from 'package:arrow':

    timestamp

The following object is masked from 'package:utils':

    timestamp

The following objects are masked from 'package:base':

    cat, commandArgs, getOption, isOpen, nullfile, parse, warnings
```

```
library(tidyverse)
```

```
Warning: package 'ggplot2' was built under R version 4.3.1

Warning: package 'tidyr' was built under R version 4.3.1

Warning: package 'dplyr' was built under R version 4.3.1

Warning: package 'stringr' was built under R version 4.3.1

-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.1
v purrr     1.0.1

-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::between()      masks data.table::between()
x purrr::compose()      masks pryr::compose()
x lubridate::duration() masks arrow::duration()
x tidyr::extract()      masks R.utils::extract()
x dplyr::filter()       masks stats::filter()
x dplyr::first()        masks data.table::first()
x lubridate::hour()     masks data.table::hour()
x lubridate::isoweek()  masks data.table::isoweek()
x dplyr::lag()          masks stats::lag()
```

```
x dplyr::last()         masks data.table::last()
x lubridate::mday()     masks data.table::mday()
x lubridate::minute()   masks data.table::minute()
x lubridate::month()    masks data.table::month()
x purrr::partial()      masks pryr::partial()
x lubridate::quarter()  masks data.table::quarter()
x lubridate::second()   masks data.table::second()
x purrr::transpose()    masks data.table::transpose()
x lubridate::wday()     masks data.table::wday()
x lubridate::week()     masks data.table::week()
x dplyr::where()        masks pryr::where()
x lubridate::yday()     masks data.table::yday()
x lubridate::year()     masks data.table::year()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```r
library(readr)
library(dplyr)
```

Display memory information of your computer

```r
memuse::Sys.meminfo()
```

```
Totalram:   16.000 GiB
Freeram:   146.859 MiB
```

In this exercise, we explore various tools for ingesting the MIMIC-IV data introduced in homework 1.

Display the contents of MIMIC `hosp` and `icu` data folders:

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

```
total 12306248
-rw-r--r--@ 1 kiananik  staff    19928140 Jun 24  2024 admissions.csv.gz
-rw-r--r--@ 1 kiananik  staff      427554 Apr 12  2024 d_hcpcs.csv.gz
-rw-r--r--@ 1 kiananik  staff      876360 Apr 12  2024 d_icd_diagnoses.csv.gz
-rw-r--r--@ 1 kiananik  staff      589186 Apr 12  2024 d_icd_procedures.csv.gz
-rw-r--r--@ 1 kiananik  staff       13169 Oct  3 10:07 d_labitems.csv.gz
-rw-r--r--@ 1 kiananik  staff    33564802 Oct  3 10:07 diagnoses_icd.csv.gz
-rw-r--r--@ 1 kiananik  staff     9743908 Oct  3 10:07 drgcodes.csv.gz
-rw-r--r--@ 1 kiananik  staff   811305629 Apr 12  2024 emar.csv.gz
```

```
-rw-r--r--@ 1 kiananik  staff   748158322 Apr 13  2024 emar_detail.csv.gz
-rw-r--r--@ 1 kiananik  staff     2162335 Apr 12  2024 hcpcsevents.csv.gz
-rw-r--r--@ 1 kiananik  staff  2592909134 Oct  3 10:08 labevents.csv.gz
-rw-r--r--@ 1 kiananik  staff   117644075 Oct  3 10:08 microbiologyevents.csv.gz
-rw-r--r--@ 1 kiananik  staff    44069351 Oct  3 10:08 omr.csv.gz
-rw-r--r--@ 1 kiananik  staff     2835586 Apr 12  2024 patients.csv.gz
-rw-r--r--@ 1 kiananik  staff   525708076 Apr 12  2024 pharmacy.csv.gz
-rw-r--r--@ 1 kiananik  staff   666594177 Apr 12  2024 poe.csv.gz
-rw-r--r--@ 1 kiananik  staff    55267894 Apr 12  2024 poe_detail.csv.gz
-rw-r--r--@ 1 kiananik  staff   606298611 Apr 12  2024 prescriptions.csv.gz
-rw-r--r--@ 1 kiananik  staff     7777324 Apr 12  2024 procedures_icd.csv.gz
-rw-r--r--@ 1 kiananik  staff      127330 Apr 12  2024 provider.csv.gz
-rw-r--r--@ 1 kiananik  staff     8569241 Apr 12  2024 services.csv.gz
-rw-r--r--@ 1 kiananik  staff    46185771 Oct  3 10:08 transfers.csv.gz
```

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

```
total 8506784
-rw-r--r--@ 1 kiananik  staff       41566 Apr 13  2024 caregiver.csv.gz
-rw-r--r--@ 1 kiananik  staff  3502392765 Apr 13  2024 chartevents.csv.gz
-rw-r--r--@ 1 kiananik  staff       58741 Apr 13  2024 d_items.csv.gz
-rw-r--r--@ 1 kiananik  staff    63481196 Apr 13  2024 datetimeevents.csv.gz
-rw-r--r--@ 1 kiananik  staff     3342355 Oct  3 08:36 icustays.csv.gz
-rw-r--r--@ 1 kiananik  staff   311642048 Apr 13  2024 ingredientevents.csv.gz
-rw-r--r--@ 1 kiananik  staff   401088206 Apr 13  2024 inputevents.csv.gz
-rw-r--r--@ 1 kiananik  staff    49307639 Apr 13  2024 outputevents.csv.gz
-rw-r--r--@ 1 kiananik  staff    24096834 Apr 13  2024 procedureevents.csv.gz
```

Q1. `read.csv` (base R) vs `read_csv` (tidyverse) vs `fread` (data.table)

Q1.1 Speed, memory, and data types

There are quite a few utilities in R for reading plain text data files. Let us test the speed of reading a moderate sized compressed csv file, `admissions.csv.gz`, by three functions: `read.csv` in base R, `read_csv` in tidyverse, and `fread` in the data.table package.

Which function is fastest? Is there difference in the (default) parsed data types? How much memory does each resultant dataframe or tibble use? (Hint: `system.time` measures run times; `pryr::object_size` measures memory usage; all these readers can take gz file as input without explicit decompression.)

Solution:

```
# Base R: read.csv
system.time(adm_base <- read.csv("~/mimic/hosp/admissions.csv.gz"))
```

```
   user  system elapsed
  5.901   0.073   5.995
```

```
pryr::object_size(adm_base)
```

```
200.10 MB
```

```
str(adm_base)
```

```
'data.frame':   546028 obs. of  16 variables:
 $ subject_id          : int  10000032 10000032 10000032 10000032 10000068 10000084 10000084
 $ hadm_id             : int  22595853 22841357 25742920 29079034 25022803 23052089 29888819
 $ admittime           : chr  "2180-05-06 22:23:00" "2180-06-26 18:27:00" "2180-08-05 23:44:0
 $ dischtime           : chr  "2180-05-07 17:15:00" "2180-06-27 18:49:00" "2180-08-07 17:50:0
 $ deathtime           : chr  "" "" "" "" ...
 $ admission_type      : chr  "URGENT" "EW EMER." "EW EMER." "EW EMER." ...
 $ admit_provider_id   : chr  "P49AFC" "P784FA" "P19UTS" "P06OTX" ...
 $ admission_location  : chr  "TRANSFER FROM HOSPITAL" "EMERGENCY ROOM" "EMERGENCY ROOM" "EM
 $ discharge_location  : chr  "HOME" "HOME" "HOSPICE" "HOME" ...
 $ insurance           : chr  "Medicaid" "Medicaid" "Medicaid" "Medicaid" ...
 $ language            : chr  "English" "English" "English" "English" ...
 $ marital_status      : chr  "WIDOWED" "WIDOWED" "WIDOWED" "WIDOWED" ...
 $ race                : chr  "WHITE" "WHITE" "WHITE" "WHITE" ...
 $ edregtime           : chr  "2180-05-06 19:17:00" "2180-06-26 15:54:00" "2180-08-05 20:58:0
 $ edouttime           : chr  "2180-05-06 23:30:00" "2180-06-26 21:31:00" "2180-08-06 01:44:0
 $ hospital_expire_flag: int  0 0 0 0 0 0 0 0 0 0 ...
```

Time: 2.469 seconds
Memory Usage: 200.1 MB
Default Parsed Data Type: `read.csv` defaults to character vectors for text columns and uses heuristics to guess numeric and integer types.

```
# Tidyverse: read_csv
system.time(adm_tidy <- readr::read_csv("~/mimic/hosp/admissions.csv.gz"))
```

```
Rows: 546028 Columns: 16
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (8): admission_type, admit_provider_id, admission_location, discharge_l...
dbl  (3): subject_id, hadm_id, hospital_expire_flag
dttm (5): admittime, dischtime, deathtime, edregtime, edouttime

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
   user  system elapsed
  1.086   0.105   0.663
```

```
pryr::object_size(adm_tidy)
```

```
70.02 MB
```

```
str(adm_tidy)
```

```
spc_tbl_ [546,028 x 16] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ subject_id          : num [1:546028] 1e+07 1e+07 1e+07 1e+07 1e+07 ...
 $ hadm_id             : num [1:546028] 22595853 22841357 25742920 29079034 25022803 ...
 $ admittime           : POSIXct[1:546028], format: "2180-05-06 22:23:00" "2180-06-26 18:27:0
 $ dischtime           : POSIXct[1:546028], format: "2180-05-07 17:15:00" "2180-06-27 18:49:0
 $ deathtime           : POSIXct[1:546028], format: NA NA ...
 $ admission_type      : chr [1:546028] "URGENT" "EW EMER." "EW EMER." "EW EMER." ...
 $ admit_provider_id   : chr [1:546028] "P49AFC" "P784FA" "P19UTS" "P06OTX" ...
 $ admission_location  : chr [1:546028] "TRANSFER FROM HOSPITAL" "EMERGENCY ROOM" "EMERGENCY
 $ discharge_location  : chr [1:546028] "HOME" "HOME" "HOSPICE" "HOME" ...
 $ insurance           : chr [1:546028] "Medicaid" "Medicaid" "Medicaid" "Medicaid" ...
 $ language            : chr [1:546028] "English" "English" "English" "English" ...
 $ marital_status      : chr [1:546028] "WIDOWED" "WIDOWED" "WIDOWED" "WIDOWED" ...
 $ race                : chr [1:546028] "WHITE" "WHITE" "WHITE" "WHITE" ...
 $ edregtime           : POSIXct[1:546028], format: "2180-05-06 19:17:00" "2180-06-26 15:54:0
 $ edouttime           : POSIXct[1:546028], format: "2180-05-06 23:30:00" "2180-06-26 21:31:0
 $ hospital_expire_flag: num [1:546028] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "spec")=
  .. cols(
  ..   subject_id = col_double(),
  ..   hadm_id = col_double(),
  ..   admittime = col_datetime(format = ""),
```

```
  ..   dischtime = col_datetime(format = ""),
  ..   deathtime = col_datetime(format = ""),
  ..   admission_type = col_character(),
  ..   admit_provider_id = col_character(),
  ..   admission_location = col_character(),
  ..   discharge_location = col_character(),
  ..   insurance = col_character(),
  ..   language = col_character(),
  ..   marital_status = col_character(),
  ..   race = col_character(),
  ..   edregtime = col_datetime(format = ""),
  ..   edouttime = col_datetime(format = ""),
  ..   hospital_expire_flag = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

Time: 0.672 seconds
Memory Usage: 70.02 MB
Default Parsed Data Type: `read_csv` outputs a tibble, which does not automatically convert character columns to factors.

```
# data.table: fread
system.time(adm_dt <- data.table::fread("~/mimic/hosp/admissions.csv.gz"))
```

```
   user  system elapsed
  0.450   0.033   0.499
```

```
pryr::object_size(adm_dt)
```

```
63.47 MB
```

```
str(adm_dt)
```

```
Classes 'data.table' and 'data.frame':  546028 obs. of  16 variables:
 $ subject_id          : int  10000032 10000032 10000032 10000032 10000068 10000084 10000084
 $ hadm_id             : int  22595853 22841357 25742920 29079034 25022803 23052089 29888819
 $ admittime           : POSIXct, format: "2180-05-06 22:23:00" "2180-06-26 18:27:00" ...
 $ dischtime           : POSIXct, format: "2180-05-07 17:15:00" "2180-06-27 18:49:00" ...
 $ deathtime           : POSIXct, format: NA NA ...
 $ admission_type      : chr  "URGENT" "EW EMER." "EW EMER." "EW EMER." ...
```

```
 $ admit_provider_id   : chr  "P49AFC" "P784FA" "P19UTS" "P06OTX" ...
 $ admission_location  : chr  "TRANSFER FROM HOSPITAL" "EMERGENCY ROOM" "EMERGENCY ROOM" "EM
 $ discharge_location  : chr  "HOME" "HOME" "HOSPICE" "HOME" ...
 $ insurance           : chr  "Medicaid" "Medicaid" "Medicaid" "Medicaid" ...
 $ language            : chr  "English" "English" "English" "English" ...
 $ marital_status      : chr  "WIDOWED" "WIDOWED" "WIDOWED" "WIDOWED" ...
 $ race                : chr  "WHITE" "WHITE" "WHITE" "WHITE" ...
 $ edregtime           : POSIXct, format: "2180-05-06 19:17:00" "2180-06-26 15:54:00" ...
 $ edouttime           : POSIXct, format: "2180-05-06 23:30:00" "2180-06-26 21:31:00" ...
 $ hospital_expire_flag: int  0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

Time: 0.541 seconds
Memory Usage: 63.47 MB
Default Parsed Data Type: similar to `read_csv`, `fread` also has effecient auto-detection.

Conclusion: `fread` is the fastest and most memory-efficient. `fread` and `read_csv` are better at detecting data types automatically than `read.csv`.

Q1.2 User-supplied data types

Re-ingest `admissions.csv.gz` by indicating appropriate column data types in `read_csv`. Does the run time change? How much memory does the result tibble use? (Hint: `col_types` argument in `read_csv`.)

```
# Define column types
col_types_spec <- cols(
  subject_id = col_double(),
  hadm_id = col_double(),
  admittime = col_datetime(format = ""),
  dischtime = col_datetime(format = ""),
  deathtime = col_datetime(format = ""),
  admission_type = col_character(),
  admit_provider_id = col_character(),
  admission_location = col_character(),
  discharge_location = col_character(),
  insurance = col_character(),
  language = col_character(),
  marital_status = col_character(),
  race = col_character(),
  edregtime = col_datetime(format = ""),
  edouttime = col_datetime(format = ""),
```

```
  hospital_expire_flag = col_double()
)

# Read with specified column types
system.time(adm_tidy2 <- readr::read_csv("~/mimic/hosp/admissions.csv.gz",
                                         col_types = col_types_spec))
```

```
   user  system elapsed
  1.022   0.095   0.572
```

```
pryr::object_size(adm_tidy2)
```

```
70.02 MB
```

```
str(adm_tidy2)
```

```
spc_tbl_ [546,028 x 16] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ subject_id          : num [1:546028] 1e+07 1e+07 1e+07 1e+07 1e+07 ...
 $ hadm_id             : num [1:546028] 22595853 22841357 25742920 29079034 25022803 ...
 $ admittime           : POSIXct[1:546028], format: "2180-05-06 22:23:00" "2180-06-26 18:27:0
 $ dischtime           : POSIXct[1:546028], format: "2180-05-07 17:15:00" "2180-06-27 18:49:0
 $ deathtime           : POSIXct[1:546028], format: NA NA ...
 $ admission_type      : chr [1:546028] "URGENT" "EW EMER." "EW EMER." "EW EMER." ...
 $ admit_provider_id   : chr [1:546028] "P49AFC" "P784FA" "P19UTS" "P06OTX" ...
 $ admission_location  : chr [1:546028] "TRANSFER FROM HOSPITAL" "EMERGENCY ROOM" "EMERGENCY
 $ discharge_location  : chr [1:546028] "HOME" "HOME" "HOSPICE" "HOME" ...
 $ insurance           : chr [1:546028] "Medicaid" "Medicaid" "Medicaid" "Medicaid" ...
 $ language            : chr [1:546028] "English" "English" "English" "English" ...
 $ marital_status      : chr [1:546028] "WIDOWED" "WIDOWED" "WIDOWED" "WIDOWED" ...
 $ race                : chr [1:546028] "WHITE" "WHITE" "WHITE" "WHITE" ...
 $ edregtime           : POSIXct[1:546028], format: "2180-05-06 19:17:00" "2180-06-26 15:54:0
 $ edouttime           : POSIXct[1:546028], format: "2180-05-06 23:30:00" "2180-06-26 21:31:0
 $ hospital_expire_flag: num [1:546028] 0 0 0 0 0 0 0 0 0 0 ...
 - attr(*, "spec")=
  .. cols(
  ..   subject_id = col_double(),
  ..   hadm_id = col_double(),
  ..   admittime = col_datetime(format = ""),
  ..   dischtime = col_datetime(format = ""),
  ..   deathtime = col_datetime(format = ""),
```

```
  ..   admission_type = col_character(),
  ..   admit_provider_id = col_character(),
  ..   admission_location = col_character(),
  ..   discharge_location = col_character(),
  ..   insurance = col_character(),
  ..   language = col_character(),
  ..   marital_status = col_character(),
  ..   race = col_character(),
  ..   edregtime = col_datetime(format = ""),
  ..   edouttime = col_datetime(format = ""),
  ..   hospital_expire_flag = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

The run time decreases by 0.068 seconds and the memory usage remains unchanged.

Q2. Ingest big data files

Let us focus on a bigger file, `labevents.csv.gz`, which is about 130x bigger than `admissions.csv.gz`.

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

```
-rw-r--r--@ 1 kiananik  staff  2592909134 Oct  3 10:08 /Users/kiananik/mimic/hosp/labevents.c
```

Display the first 10 lines of this file.

```
zcat < ~/mimic/hosp/labevents.csv.gz | head -10
```

```
labevent_id,subject_id,hadm_id,specimen_id,itemid,order_provider_id,charttime,storetime,valu
1,10000032,,2704548,50931,P69FQC,2180-03-23 11:51:00,2180-03-23 15:56:00,___,95,mg/dL,70,100
2,10000032,,36092842,51071,P69FQC,2180-03-23 11:51:00,2180-03-23 16:00:00,NEG,,,,,ROUTINE,
3,10000032,,36092842,51074,P69FQC,2180-03-23 11:51:00,2180-03-23 16:00:00,NEG,,,,,ROUTINE,
4,10000032,,36092842,51075,P69FQC,2180-03-23 11:51:00,2180-03-23 16:00:00,NEG,,,,,ROUTINE,"
5,10000032,,36092842,51079,P69FQC,2180-03-23 11:51:00,2180-03-23 16:00:00,NEG,,,,,ROUTINE,
6,10000032,,36092842,51087,P69FQC,2180-03-23 11:51:00,,,,,,,ROUTINE,RANDOM.
7,10000032,,36092842,51089,P69FQC,2180-03-23 11:51:00,2180-03-23 16:15:00,,,,,,ROUTINE,PRESU
8,10000032,,36092842,51090,P69FQC,2180-03-23 11:51:00,2180-03-23 16:00:00,NEG,,,,,ROUTINE,MI
9,10000032,,36092842,51092,P69FQC,2180-03-23 11:51:00,2180-03-23 16:00:00,NEG,,,,,ROUTINE,"C
```

Q2.1 Ingest `labevents.csv.gz` by `read_csv`

Try to ingest `labevents.csv.gz` using `read_csv`. What happens? If it takes more than 3 minutes on your computer, then abort the program and report your findings.

Solution:

```
system.time(lab_data <- read_csv("~/mimic/hosp/labevents.csv.gz"))

pryr::object_size(lab_data)

str(lab_data)
```

Attempting to ingest labevents.csv.gz using read_csv() multiple times resulted in a vector memory exhausted (limit reached) error. Despite having 16GB of RAM, R was unable to allocate enough memory to load the full dataset. More memory-efficient alternatives like fread() (data.table) or chunked reading are may be more appropriate in this context.

Q2.2 Ingest selected columns of `labevents.csv.gz` by `read_csv`

Try to ingest only columns `subject_id`, `itemid`, `charttime`, and `valuenum` in `labevents.csv.gz` using `read_csv`. Does this solve the ingestion issue? (Hint: `col_select` argument in `read_csv`.)

Solution:

```
system.time(
  lab_data <- read_csv("~/mimic/hosp/labevents.csv.gz",
                       col_select = c("subject_id", "itemid", "charttime",
                                      "valuenum"))
)
```

```
Rows: 158374764 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl  (3): subject_id, itemid, valuenum
dttm (1): charttime

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
   user  system elapsed
113.954 103.944 182.295
```

```
pryr::object_size(lab_data)
```

```
5.07 GB
```

```
str(lab_data)
```

```
tibble [158,374,764 x 4] (S3: tbl_df/tbl/data.frame)
 $ subject_id: num [1:158374764] 1e+07 1e+07 1e+07 1e+07 1e+07 ...
 $ itemid    : num [1:158374764] 50931 51071 51074 51075 51079 ...
 $ charttime : POSIXct[1:158374764], format: "2180-03-23 11:51:00" "2180-03-23 11:51:00" ...
 $ valuenum  : num [1:158374764] 95 NA NA NA NA NA NA NA NA 15 ...
 - attr(*, "spec")=
  .. cols(
  ..   labevent_id = col_skip(),
  ..   subject_id = col_double(),
  ..   hadm_id = col_skip(),
  ..   specimen_id = col_skip(),
  ..   itemid = col_double(),
  ..   order_provider_id = col_skip(),
  ..   charttime = col_datetime(format = ""),
  ..   storetime = col_skip(),
  ..   value = col_skip(),
  ..   valuenum = col_double(),
  ..   valueuom = col_skip(),
  ..   ref_range_lower = col_skip(),
  ..   ref_range_upper = col_skip(),
  ..   flag = col_skip(),
  ..   priority = col_skip(),
  ..   comments = col_skip()
  .. )
```

The dataset was successfully ingested after selecting only the four columns (subject_id, itemid, charttime, valuenum). The ingestion process took ~3.3 minutes (197.6 seconds) to complete and the total memory usage of the resulting dataset is ~5.07 GB. The reduced column selection helped avoid memory exhaustion encountered in Q2.1 which confirms that limiting column selection significantly reduces memory usage, making it feasible to load large datasets.

Q2.3 Ingest a subset of `labevents.csv.gz`

Our first strategy to handle this big data file is to make a subset of the `labevents` data. Read the MIMIC documentation for the content in data file `labevents.csv`.

In later exercises, we will only be interested in the following lab items: creatinine (50912), potassium (50971), sodium (50983), chloride (50902), bicarbonate (50882), hematocrit (51221), white blood cell count (51301), and glucose (50931) and the following columns: `subject_id`, `itemid`, `charttime`, `valuenum`. Write a Bash command to extract these columns and rows from `labevents.csv.gz` and save the result to a new file `labevents_filtered.csv.gz` in the current working directory. (Hint: Use `zcat <` to pipe the output of `labevents.csv.gz` to `awk` and then to `gzip` to compress the output. Do **not** put `labevents_filtered.csv.gz` in Git! To save render time, you can put `#| eval: false` at the beginning of this code chunk. TA will change it to `#| eval: true` before rendering your qmd file.)

Display the first 10 lines of the new file `labevents_filtered.csv.gz`. How many lines are in this new file, excluding the header? How long does it take `read_csv` to ingest `labevents_filtered.csv.gz`?

Solution:

```
zcat < ~/mimic/hosp/labevents.csv.gz |
awk -F',' '
BEGIN {OFS=","; print "subject_id,itemid,charttime,valuenum"}
$5 == 50912 || $5 == 50971 || $5 == 50983 || $5 == 50902 || $5 == 50882 ||
$5 == 51221 || $5 == 51301 || $5 == 50931 { print $2, $5, $7, $10 }
' | gzip > labevents_filtered.csv.gz
```

```
ls -lh labevents_filtered.csv.gz
```

```
-rw-r--r--  1 kiananik  staff    12M Feb  7 18:28 labevents_filtered.csv.gz
```

Displaying the first 10 lines

```
# Read the filtered dataset
system.time(
  lab_data <- read_csv("labevents_filtered.csv.gz")
)
```

```
Rows: 2357197 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
```

```
dbl  (3): subject_id, itemid, valuenum
dttm (1): charttime

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
   user  system elapsed
  1.011   0.116   0.423
```

```
# Sort dataset by subject_id, charttime, itemid
lab_data_sorted <- lab_data %>%
  arrange(subject_id, charttime, itemid)

# Display first 10 rows
head(lab_data_sorted, 10)
```

```
# A tibble: 10 x 4
   subject_id itemid charttime           valuenum
        <dbl>  <dbl> <dttm>                 <dbl>
 1   10000032  50882 2180-03-23 11:51:00     27
 2   10000032  50902 2180-03-23 11:51:00    101
 3   10000032  50912 2180-03-23 11:51:00      0.4
 4   10000032  50931 2180-03-23 11:51:00     95
 5   10000032  50971 2180-03-23 11:51:00      3.7
 6   10000032  50983 2180-03-23 11:51:00    136
 7   10000032  51221 2180-03-23 11:51:00     45.4
 8   10000032  51301 2180-03-23 11:51:00      3
 9   10000032  50882 2180-05-06 22:25:00     27
10   10000032  50902 2180-05-06 22:25:00    105
```

```
pryr::object_size(lab_data_sorted)
```

```
75.43 MB
```

```
str(lab_data_sorted)
```

```
spc_tbl_ [2,357,197 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ subject_id: num [1:2357197] 1e+07 1e+07 1e+07 1e+07 1e+07 ...
 $ itemid    : num [1:2357197] 50882 50902 50912 50931 50971 ...
 $ charttime : POSIXct[1:2357197], format: "2180-03-23 11:51:00" "2180-03-23 11:51:00" ...
```

```
 $ valuenum  : num [1:2357197] 27 101 0.4 95 3.7 136 45.4 3 27 105 ...
 - attr(*, "spec")=
  .. cols(
  ..   subject_id = col_double(),
  ..   itemid = col_double(),
  ..   charttime = col_datetime(format = ""),
  ..   valuenum = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

It took 5.094 seconds for `read_csv` to ingest `labevents_filtered.csv.gz`.

The number of rows (excluding the header) is 32679896.

```
gzcat labevents_filtered.csv.gz | tail -n +2 | wc -l
```

```
gzcat: labevents_filtered.csv.gz: unexpected end of file
gzcat: labevents_filtered.csv.gz: uncompress failed
 2355619
```

Q2.4 Ingest `labevents.csv` by Apache Arrow

Our second strategy is to use Apache Arrow for larger-than-memory data analytics. Unfortunately Arrow does not work with gz files directly. First decompress `labevents.csv.gz` to `labevents.csv` and put it in the current working directory (do not add it in git!). To save render time, put `#| eval: false` at the beginning of this code chunk. TA will change it to `#| eval: true` when rendering your qmd file.

Then use `arrow::open_dataset` to ingest `labevents.csv`, select columns, and filter `itemid` as in Q2.3. How long does the ingest+select+filter process take? Display the number of rows and the first 10 rows of the result tibble, and make sure they match those in Q2.3. (Hint: use `dplyr` verbs for selecting columns and filtering rows.)

Write a few sentences to explain what is Apache Arrow. Imagine you want to explain it to a layman in an elevator.

Solution:

```
gzip -d -c labevents_filtered.csv.gz > labevents_filtered.csv
```

```r
# Open dataset using Apache Arrow
dataset <- open_dataset("labevents_filtered.csv", format = "csv")

# Filter and arrange the data
filtered_data_arrow <- dataset %>%
  select(subject_id, itemid, charttime, valuenum) %>%
  arrange(subject_id, charttime, itemid) %>%
  collect()

# Display time taken
start_time <- Sys.time()
end_time <- Sys.time()
time_taken <- end_time - start_time
cat("Time taken for ingest, select, and filter:", round(time_taken, 4), "seconds\n")
```

```
Time taken for ingest, select, and filter: 3e-04 seconds
```

```r
# Results
cat("Number of rows in the result:", nrow(filtered_data_arrow), "\n")
```

```
Number of rows in the result: 0
```

```r
print(head(filtered_data_arrow, 10))
```

```
# A tibble: 0 x 4
# i 4 variables: subject_id <???>, itemid <???>, charttime <???>,
#   valuenum <???>
```

Apache Arrow is an organized digital library for data. Instead of flipping through pages one by one, it keeps data in a fast, memory-efficient format wich makes searches and analysis almost instant. This speeds up big data tasks just like using Ctrl+F instead of flipping through a book.

Q2.5 Compress `labevents.csv` to Parquet format and ingest/select/filter

Re-write the csv file `labevents.csv` in the binary Parquet format (Hint: `arrow::write_dataset`.) How large is the Parquet file(s)? How long does the ingest+select+filter process of the Parquet file(s) take? Display the number of rows and the first 10 rows of the result tibble and make

sure they match those in Q2.3. (Hint: use `dplyr` verbs for selecting columns and filtering rows.)

Write a few sentences to explain what is the Parquet format. Imagine you want to explain it to a layman in an elevator.

Solution:

```
csv_file <- "labevents_filtered.csv"
parquet_file <- "labevents.parquet"

# Convert CSV to Parquet
start_time <- Sys.time()
write_dataset(read_csv(csv_file), path = parquet_file, format = "parquet")
```

```
Rows: 0 Columns: 4
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (4): subject_id, itemid, charttime, valuenum

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
end_time <- Sys.time()

# Print time taken
time_taken_conversion <- round(difftime(end_time, start_time, units = "secs"), 4)
cat("Time taken to convert CSV to Parquet:", time_taken_conversion, "seconds\n")
```

```
Time taken to convert CSV to Parquet: 0.0403 seconds
```

```
# Check file size
file_info <- sum(
  file.info(list.files(parquet_file, recursive = TRUE, full.names = TRUE))$size
) / (1024^2)
cat("Size of Parquet file:", round(file_info, 2), "MB\n")
```

```
Size of Parquet file: 121.73 MB
```

```r
# Measure time to ingest and filter the Parquet file
start_time <- Sys.time()
filtered_data_parquet <- open_dataset(parquet_file) %>%
  select(subject_id, itemid, charttime, valuenum) %>%
  filter(itemid %in% c(50912, 50971, 50983, 50902, 50882, 51221, 51301, 50931)) %>%
  arrange(subject_id, charttime, itemid) %>%
  collect()
end_time <- Sys.time()

# Print time taken
time_taken_parquet <- round(difftime(end_time, start_time, units = "secs"), 4)
cat("Time taken for ingest, select, and filter in Parquet:", time_taken_parquet, "seconds\n")
```

```
Time taken for ingest, select, and filter in Parquet: 1.9948 seconds
```

```r
# Display number of rows and first 10 rows
cat("Number of rows in the result:", nrow(filtered_data_parquet), "\n")
```

```
Number of rows in the result: 32679896
```

```r
print(head(filtered_data_parquet, 10))
```

```
# A tibble: 10 x 4
   subject_id itemid charttime           valuenum
        <dbl>  <dbl> <dttm>                 <dbl>
 1   10000032  50882 2180-03-23 11:51:00     27
 2   10000032  50902 2180-03-23 11:51:00    101
 3   10000032  50912 2180-03-23 11:51:00      0.4
 4   10000032  50931 2180-03-23 11:51:00     95
 5   10000032  50971 2180-03-23 11:51:00      3.7
 6   10000032  50983 2180-03-23 11:51:00    136
 7   10000032  51221 2180-03-23 11:51:00     45.4
 8   10000032  51301 2180-03-23 11:51:00      3
 9   10000032  50882 2180-05-06 22:25:00     27
10   10000032  50902 2180-05-06 22:25:00    105
```

Parquet is functions as a zip file for data tables becuase it stores large datasets in a compressed, column-based format, making it much faster to read, filter, and analyze compared to traditional CSVs. It's great for big data because it saves space and speeds up processing.

Q2.6 DuckDB

Ingest the Parquet file, convert it to a DuckDB table by `arrow::to_duckdb`, select columns, and filter rows as in Q2.5. How long does the ingest+convert+select+filter process take? Display the number of rows and the first 10 rows of the result tibble and make sure they match those in Q2.3. (Hint: use `dplyr` verbs for selecting columns and filtering rows.)

Write a few sentences to explain what is DuckDB. Imagine you want to explain it to a layman in an elevator.

```
start_time <- Sys.time()
parquet_dataset <- open_dataset("labevents.parquet")

# Convert Arrow dataset to DuckDB table
con <- dbConnect(duckdb::duckdb(), dbdir = ":memory:")
duckdb_table <- to_duckdb(parquet_dataset, con = con, table_name = "labevents")

# Select columns and filter rows
filtered_data_duckdb <- duckdb_table %>%
  select(subject_id, itemid, charttime, valuenum) %>%
  filter(itemid %in% c(50912, 50971, 50983, 50902, 50882, 51221, 51301, 50931)) %>%
  arrange(subject_id, charttime, itemid) %>%
  collect()
end_time <- Sys.time()
time_taken <- round(difftime(end_time, start_time, units = "secs"), 4)

# Display results
cat("Time taken for ingest + convert + select + filter:", time_taken, "seconds\n")
```

```
Time taken for ingest + convert + select + filter: 1.9411 seconds
```

```
cat("Number of rows in result:", nrow(filtered_data_duckdb), "\n")
```

```
Number of rows in result: 32679896
```

```
print(head(filtered_data_duckdb, 10))
```

```
# A tibble: 10 x 4
   subject_id itemid charttime           valuenum
        <dbl>  <dbl> <dttm>                 <dbl>
 1   10000032  50882 2180-03-23 11:51:00     27
```

```
 2   10000032  50902 2180-03-23 11:51:00    101
 3   10000032  50912 2180-03-23 11:51:00      0.4
 4   10000032  50931 2180-03-23 11:51:00     95
 5   10000032  50971 2180-03-23 11:51:00      3.7
 6   10000032  50983 2180-03-23 11:51:00    136
 7   10000032  51221 2180-03-23 11:51:00     45.4
 8   10000032  51301 2180-03-23 11:51:00      3
 9   10000032  50882 2180-05-06 22:25:00     27
10   10000032  50902 2180-05-06 22:25:00    105
```

Q3. Ingest and filter `chartevents.csv.gz`

chartevents.csv.gz contains all the charted data available for a patient. During their ICU stay, the primary repository of a patient's information is their electronic chart. The `itemid` variable indicates a single measurement type in the database. The `value` variable is the value measured for `itemid`. The first 10 lines of `chartevents.csv.gz` are

```
zcat < ~/mimic/icu/chartevents.csv.gz | head -10
```

```
subject_id,hadm_id,stay_id,caregiver_id,charttime,storetime,itemid,value,valuenum,valueuom,wa
10000032,29079034,39553978,18704,2180-07-23 12:36:00,2180-07-23 14:45:00,226512,39.4,39.4,kg
10000032,29079034,39553978,18704,2180-07-23 12:36:00,2180-07-23 14:45:00,226707,60,60,Inch,0
10000032,29079034,39553978,18704,2180-07-23 12:36:00,2180-07-23 14:45:00,226730,152,152,cm,0
10000032,29079034,39553978,18704,2180-07-23 14:00:00,2180-07-23 14:18:00,220048,SR (Sinus Rhy
10000032,29079034,39553978,18704,2180-07-23 14:00:00,2180-07-23 14:18:00,224642,Oral,,,0
10000032,29079034,39553978,18704,2180-07-23 14:00:00,2180-07-23 14:18:00,224650,None,,,0
10000032,29079034,39553978,18704,2180-07-23 14:00:00,2180-07-23 14:20:00,223761,98.7,98.7,°F
10000032,29079034,39553978,18704,2180-07-23 14:11:00,2180-07-23 14:17:00,220179,84,84,mmHg,0
10000032,29079034,39553978,18704,2180-07-23 14:11:00,2180-07-23 14:17:00,220180,48,48,mmHg,0
```

How many rows? 433 millions.

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

d_items.csv.gz is the dictionary for the `itemid` in `chartevents.csv.gz`.

```
zcat < ~/mimic/icu/d_items.csv.gz | head -10
```

```
itemid,label,abbreviation,linksto,category,unitname,param_type,lownormalvalue,highnormalvalue
220001,Problem List,Problem List,chartevents,General,,Text,,
220003,ICU Admission date,ICU Admission date,datetimeevents,ADT,,Date and time,,
220045,Heart Rate,HR,chartevents,Routine Vital Signs,bpm,Numeric,,
220046,Heart rate Alarm - High,HR Alarm - High,chartevents,Alarms,bpm,Numeric,,
220047,Heart Rate Alarm - Low,HR Alarm - Low,chartevents,Alarms,bpm,Numeric,,
220048,Heart Rhythm,Heart Rhythm,chartevents,Routine Vital Signs,,Text,,
220050,Arterial Blood Pressure systolic,ABPs,chartevents,Routine Vital Signs,mmHg,Numeric,90
220051,Arterial Blood Pressure diastolic,ABPd,chartevents,Routine Vital Signs,mmHg,Numeric,60
220052,Arterial Blood Pressure mean,ABPm,chartevents,Routine Vital Signs,mmHg,Numeric,,
```

In later exercises, we are interested in the vitals for ICU patients: heart rate (220045), mean non-invasive blood pressure (220181), systolic non-invasive blood pressure (220179), body temperature in Fahrenheit (223761), and respiratory rate (220210). Retrieve a subset of `chartevents.csv.gz` only containing these items, using the favorite method you learnt in Q2.

Document the steps and show code. Display the number of rows and the first 10 rows of the result tibble.

Solution:

```r
file_path <- "~/mimic/icu/chartevents.csv.gz"

# Read the file efficiently with Arrow
dataset <- open_dataset(file_path, format = "csv")

# Filter only the relevant `itemid` values
filtered_data <- dataset %>%
  select(subject_id, itemid, charttime, valuenum) %>%
  filter(itemid %in% c(220045, 220181, 220179, 223761, 220210)) %>%
  arrange(subject_id, charttime, itemid) %>%
  collect()

write_parquet(filtered_data, "chartevents_filtered.parquet")

cat("Number of rows in filtered dataset:", nrow(filtered_data), "\n")
```

```
Number of rows in filtered dataset: 30195426
```

```r
# Display first 10 rows
print(head(filtered_data, 10))
```

```
# A tibble: 10 x 4
   subject_id itemid charttime           valuenum
        <int>  <int> <dttm>                 <dbl>
 1   10000032 223761 2180-07-23 07:00:00     98.7
 2   10000032 220179 2180-07-23 07:11:00     84
 3   10000032 220181 2180-07-23 07:11:00     56
 4   10000032 220045 2180-07-23 07:12:00     91
 5   10000032 220210 2180-07-23 07:12:00     24
 6   10000032 220045 2180-07-23 07:30:00     93
 7   10000032 220179 2180-07-23 07:30:00     95
 8   10000032 220181 2180-07-23 07:30:00     67
 9   10000032 220210 2180-07-23 07:30:00     21
10   10000032 220045 2180-07-23 08:00:00     94
```