Biostat 203B Homework 2

Due Feb 7, 2025 @ 11:59PM

Ningke Zhang 705834790

Display machine information for reproducibility:

```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31)
Platform: aarch64-apple-darwin20
Running under: macOS Sequoia 15.3

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-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_4.4.2    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_1.0.1
```

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

```
library(arrow)
```

```
Attaching package: 'arrow'

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

    timestamp
```

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

```
Loading required package: DBI
```

```
library(memuse)
library(pryr)
```

```
Attaching package: 'pryr'

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

    address
```

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

```
Loading required package: R.oo

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 v1.27.0 (2024-11-01 18:00:02 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, use, warnings
```

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.2
```

```
-- 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
```

Display memory information of your computer

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

```
Totalram:  16.000 GiB
Freeram:    1.742 GiB
```

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:

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

```
total 12306256
-rw-rw-r--@ 1 ningkezhang  staff    19928140 Jun 24  2024 admissions.csv.gz
```

```
-rw-rw-r--@ 1 ningkezhang  staff      427554 Apr 12  2024 d_hcpcs.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff      876360 Apr 12  2024 d_icd_diagnoses.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff      589186 Apr 12  2024 d_icd_procedures.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff       13169 Oct  3 06:07 d_labitems.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff    33564802 Oct  3 06:07 diagnoses_icd.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff     9743908 Oct  3 06:07 drgcodes.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff   811305629 Apr 12  2024 emar.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff   748158322 Apr 12  2024 emar_detail.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff     2162335 Apr 12  2024 hcpcsevents.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff        2907 Dec 28 18:04 index.html
-rw-rw-r--@ 1 ningkezhang  staff  2592909134 Oct  3 06:08 labevents.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff   117644075 Oct  3 06:08 microbiologyevents.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff    44069351 Oct  3 06:08 omr.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff     2835586 Apr 12  2024 patients.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff   525708076 Apr 12  2024 pharmacy.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff   666594177 Apr 12  2024 poe.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff    55267894 Apr 12  2024 poe_detail.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff   606298611 Apr 12  2024 prescriptions.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff     7777324 Apr 12  2024 procedures_icd.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff      127330 Apr 12  2024 provider.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff     8569241 Apr 12  2024 services.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff    46185771 Oct  3 06:08 transfers.csv.gz
```

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

```
total 8506792
-rw-rw-r--@ 1 ningkezhang  staff       41566 Apr 12  2024 caregiver.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff  3502392765 Apr 12  2024 chartevents.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff       58741 Apr 12  2024 d_items.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff    63481196 Apr 12  2024 datetimeevents.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff     3342355 Oct  3 04:36 icustays.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff        1336 Dec 28 18:04 index.html
-rw-rw-r--@ 1 ningkezhang  staff   311642048 Apr 12  2024 ingredientevents.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff   401088206 Apr 12  2024 inputevents.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff    49307639 Apr 12  2024 outputevents.csv.gz
-rw-rw-r--@ 1 ningkezhang  staff    24096834 Apr 12  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.)

Solusion: `fread` is the fastest. The default parsed data types are different. `read.csv` uses `factor` for Character Columns, `num` or `int` for Numeric Columns, and `Chr` for Logical Columns, and automatically converted from character for factors. `read_csv` uses `chr` for character columns, `dbl` or `int` for numeric columns, parsed as date or datetime for date columns,`lgl` for logical columns. `fread` uses `chr` for character columns, `num` or `int` for numeric columns, `lgl` for logical columns. The memory usage of `fread` is the smallest.

```
system.time(read.csv("~/mimic/hosp/admissions.csv.gz"))
```

```
   user  system elapsed
  4.905   0.045   4.950
```

```
system.time(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
  0.838   0.077   0.473
```

```
system.time(fread("~/mimic/hosp/admissions.csv.gz"))
```

```
   user  system elapsed
  0.492   0.046   0.241
```

```
str(read.csv("~/mimic/hosp/admissions.csv.gz"))
```

```
'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 ...
```

```
glimpse(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.

Rows: 546,028
Columns: 16
```

```
$ subject_id          <dbl> 10000032, 10000032, 10000032, 10000032, 10000068,~
$ hadm_id             <dbl> 22595853, 22841357, 25742920, 29079034, 25022803,~
$ admittime           <dttm> 2180-05-06 22:23:00, 2180-06-26 18:27:00, 2180-0~
$ dischtime           <dttm> 2180-05-07 17:15:00, 2180-06-27 18:49:00, 2180-0~
$ deathtime           <dttm> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
$ admission_type      <chr> "URGENT", "EW EMER.", "EW EMER.", "EW EMER.", "EU~
$ admit_provider_id   <chr> "P49AFC", "P784FA", "P19UTS", "P06OTX", "P39NWO",~
$ admission_location  <chr> "TRANSFER FROM HOSPITAL", "EMERGENCY ROOM", "EMER~
$ discharge_location  <chr> "HOME", "HOME", "HOSPICE", "HOME", NA, "HOME HEAL~
$ insurance           <chr> "Medicaid", "Medicaid", "Medicaid", "Medicaid", N~
$ language            <chr> "English", "English", "English", "English", "Engl~
$ marital_status      <chr> "WIDOWED", "WIDOWED", "WIDOWED", "WIDOWED", "SING~
$ race                <chr> "WHITE", "WHITE", "WHITE", "WHITE", "WHITE", "WHI~
$ edregtime           <dttm> 2180-05-06 19:17:00, 2180-06-26 15:54:00, 2180-0~
$ edouttime           <dttm> 2180-05-06 23:30:00, 2180-06-26 21:31:00, 2180-0~
$ hospital_expire_flag <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
```

```
str(fread("~/mimic/hosp/admissions.csv.gz"))
```

```
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>
```

```
pryr::object_size(read.csv("~/mimic/hosp/admissions.csv.gz"))
```

```
200.10 MB
```

```
pryr::object_size(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.
```

```
70.02 MB
```

```
pryr::object_size(fread("~/mimic/hosp/admissions.csv.gz"))
```

```
63.47 MB
```

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`.)

Solution: The run time is faster than the default data types. The memory usage is stay same.

```
system.time(read_csv("~/mimic/hosp/admissions.csv.gz",
                     col_types = cols()))
```

```
   user  system elapsed 
  0.794   0.055   0.406 
```

```
pryr::object_size(read_csv("~/mimic/hosp/admissions.csv.gz",
                           col_types = cols()))
```

```
70.02 MB
```

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-rw-r--@ 1 ningkezhang  staff  2592909134 Oct  3 06:08 /Users/ningkezhang/mimic/hosp/labev
```

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,value
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,"(
```

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

Figure 1: readr

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: It takes more than 5 minutes, and timing stopped, only 79.594 MiB of free RAM is available. Also, when I try `fread`, it shows: vector memory limit of 16.0 Gb reached, so processing Big Data is very demanding on RAM.

```
system.time(read_csv("~/mimic/hosp/labevents.csv.gz"))
pryr::object_size(read_csv("~/mimic/hosp/labevents.csv.gz"))

system.time(fread("~/mimic/hosp/labevents.csv.gz"))
```

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

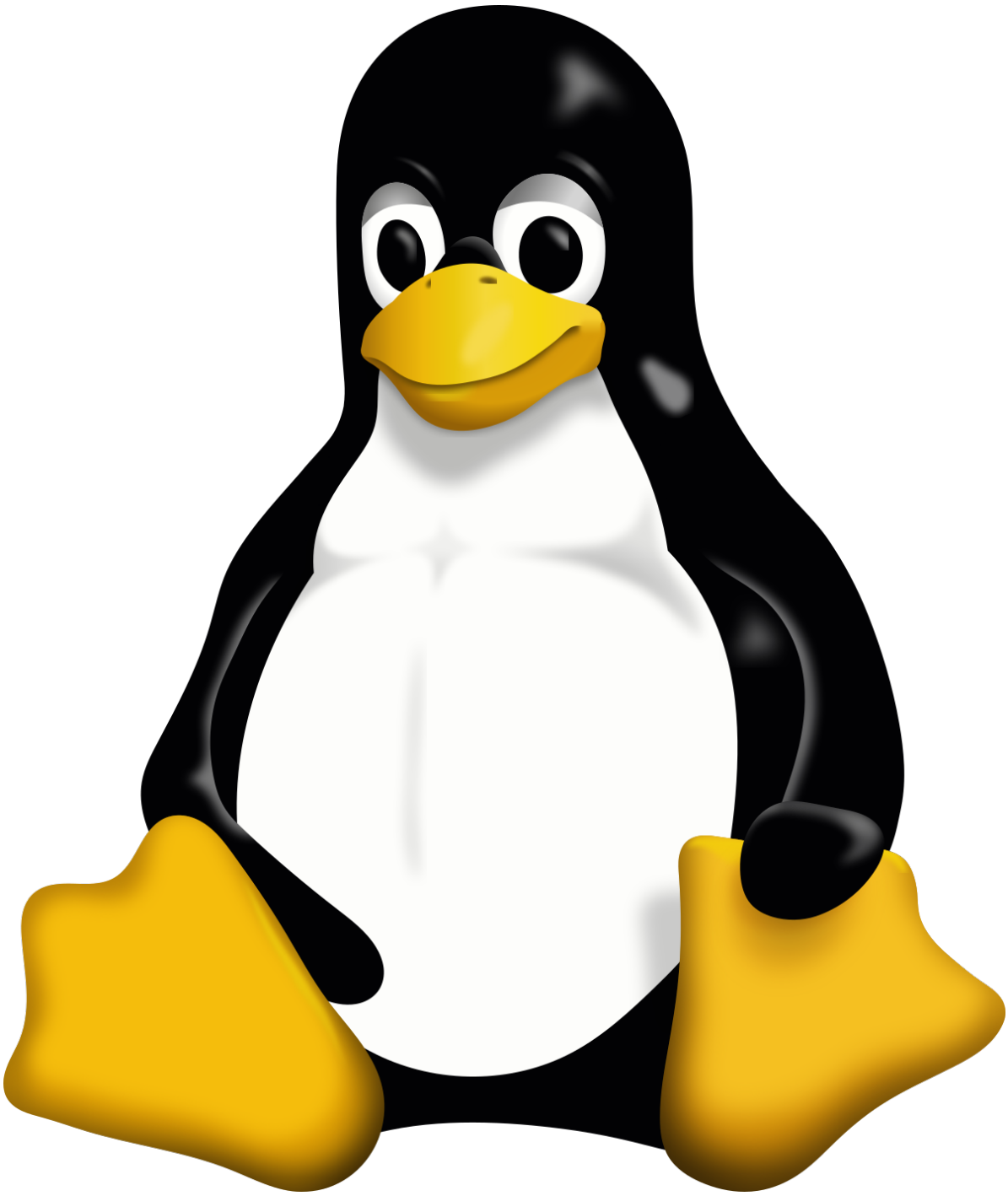
Try to ingest only columns `subject_id`, `itemid`, `charttime`, and `valuenum` in `labevents.csv.gz` using `read_csv`. Does this solve the ingestion issue? (Hint: `col_select` argument in `read_csv`.)

Solution: For `read_csv` still can not solve the ingestion issue, but `fread` can take 36.124s to process, which menas fewer columns come with less memory usage and less processing to get faster ingestion. `read_csv` maybe cannot ingest gz files directly.

```
system.time(read_csv("~/mimic/hosp/labevents.csv.gz",
            col_select = c("subject_id", "itemid", "charttime", "valuenum")))
pryr::object_size(read_csv("~/mimic/hosp/labevents.csv.gz",
            col_select = c("subject_id", "itemid", "charttime", "valuenum")))

system.time(fread("~/mimic/hosp/labevents.csv.gz",
            select = c("subject_id", "itemid", "charttime", "valuenum")))
```

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: There are 32679896 lines, excluding the header. It takes 12.492s, with 1.05 GB RAM.

```
zcat < ~/mimic/hosp/labevents.csv.gz |
awk -F',' 'NR==1 {print "subject_id,itemid,charttime,valuenum"; next}
           ($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
```

```
zcat < labevents_filtered.csv.gz | head -10

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

```
subject_id,itemid,charttime,valuenum
10000032,50931,2180-03-23 11:51:00,95
10000032,50882,2180-03-23 11:51:00,27
10000032,50902,2180-03-23 11:51:00,101
10000032,50912,2180-03-23 11:51:00,0.4
10000032,50971,2180-03-23 11:51:00,3.7
10000032,50983,2180-03-23 11:51:00,136
10000032,51221,2180-03-23 11:51:00,45.4
10000032,51301,2180-03-23 11:51:00,3
10000032,51221,2180-05-06 22:25:00,42.6
 32679896
```

```
system.time(
  lab <- read_csv("labevents_filtered.csv.gz")
)
```

```
Rows: 32679896 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
 12.805   1.272   5.156
```

```
pryr::object_size(lab)
```

```
1.05 GB
```

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

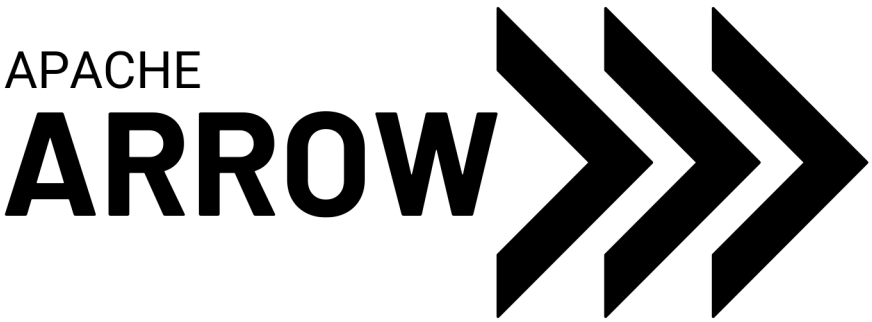


Figure 2: arrow

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: The process takes 37.127s. Apache Arrow keeps data in a special format that we can read instantly, without wasting time converting files.

```
gunzip -c ~/mimic/hosp/labevents.csv.gz > labevents.csv
```

```
system.time({
  lab_csv <- open_dataset("labevents.csv", format = "csv")

  arrow <- lab_csv |>
    select(subject_id, itemid, charttime, valuenum) |>
    filter(itemid %in%
             c(50912, 50971, 50983, 50902, 50882, 51221, 51301, 50931)) |>
    collect()
})
```

```
   user  system elapsed
 37.771   4.019  35.867
```

```
nrow(arrow)
```

```
[1] 32679896
```

```
print(head(arrow, 10))
```

```
# A tibble: 10 x 4
   subject_id itemid charttime           valuenum
        <int>  <int> <dttm>                 <dbl>
 1   10000032  50931 2180-03-23 04:51:00     95
```

```
 2   10000032  50882 2180-03-23 04:51:00     27
 3   10000032  50902 2180-03-23 04:51:00    101
 4   10000032  50912 2180-03-23 04:51:00      0.4
 5   10000032  50971 2180-03-23 04:51:00      3.7
 6   10000032  50983 2180-03-23 04:51:00    136
 7   10000032  51221 2180-03-23 04:51:00     45.4
 8   10000032  51301 2180-03-23 04:51:00      3
 9   10000032  51221 2180-05-06 15:25:00     42.6
10   10000032  51301 2180-05-06 15:25:00      5
```

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

Figure 3: parquet

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: The Parquet is 2.5G, process takes 6.801s. Parquet is like a ZIP file for big data. It compresses and organizes information, only reads what's needed, so it takes up less space and loads much faster than read_csv.

```
parquet <- "labevents_parquet"

write_dataset(open_dataset("labevents.csv", format = "csv"),
              path = parquet, format = "parquet")

system(sprintf("du -sh %s", parquet))
```

```
system.time({
  parquet_ds <- open_dataset(parquet, format = "parquet")

  parquet_df <- parquet_ds |>
    select(subject_id, itemid, charttime, valuenum) |>
    filter(itemid %in%
             c(50912, 50971, 50983, 50902, 50882, 51221, 51301, 50931)) |>
    collect()
})
```

```
   user  system elapsed
  6.556   1.990   2.967
```

```
nrow(parquet_df)
```

```
[1] 32679896
```

```
print(head(parquet_df, 10))
```

```
# A tibble: 10 x 4
   subject_id itemid charttime           valuenum
        <int>  <int> <dttm>                 <dbl>
 1   10000032  50931 2180-03-23 04:51:00     95
 2   10000032  50882 2180-03-23 04:51:00     27
 3   10000032  50902 2180-03-23 04:51:00    101
 4   10000032  50912 2180-03-23 04:51:00      0.4
 5   10000032  50971 2180-03-23 04:51:00      3.7
 6   10000032  50983 2180-03-23 04:51:00    136
 7   10000032  51221 2180-03-23 04:51:00     45.4
 8   10000032  51301 2180-03-23 04:51:00      3
 9   10000032  51221 2180-05-06 15:25:00     42.6
10   10000032  51301 2180-05-06 15:25:00      5
```

Q2.6 DuckDB

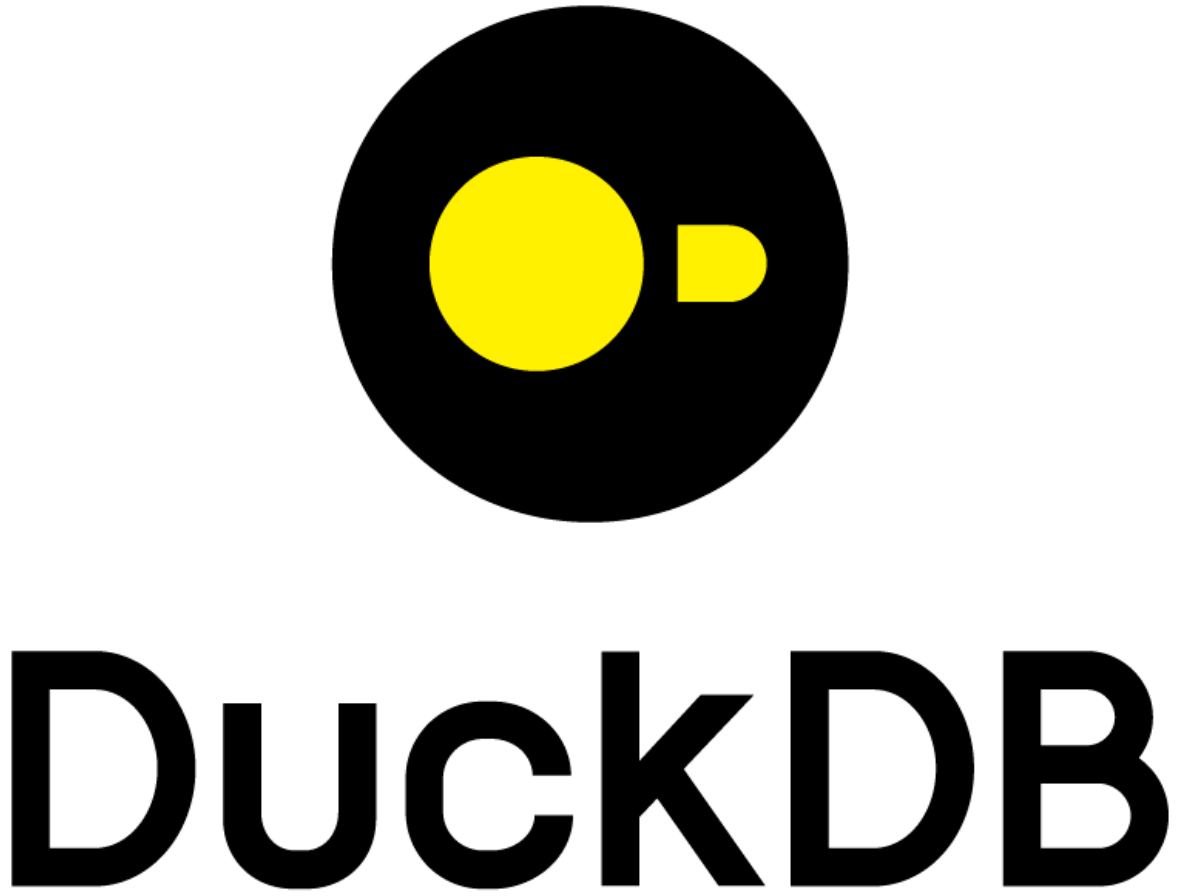


Figure 4: duckdb

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.

Solution: It takes 7.782s to process. DuckDB is a high-performance, lightweight database that runs locally. If opening an excel table with millions of rows of data will get stuck, use DuckDB to filter, calculate and merge data will be on the fly.

```r
system.time({
  parquet_db <- open_dataset("labevents_parquet", format = "parquet")
  con <- dbConnect(duckdb())
  db <- to_duckdb(parquet_db, con = con)
  duckdb_df <- db |>
    select(subject_id, itemid, charttime, valuenum) |>
    filter(itemid %in%
             c(50912, 50971, 50983, 50902, 50882, 51221, 51301, 50931)) |>
    collect()
})
```

```
   user  system elapsed
  7.340   2.195   3.022
```

```r
nrow(duckdb_df)
```

```
[1] 32679896
```

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

```
# A tibble: 10 x 4
   subject_id itemid charttime           valuenum
        <dbl>  <dbl> <dttm>                 <dbl>
 1   10003417  51301 2111-01-11 11:42:00      7
 2   10003417  50882 2111-01-24 18:03:00     25
 3   10003417  50902 2111-01-24 18:03:00    103
 4   10003417  50912 2111-01-24 18:03:00      2.5
 5   10003417  50971 2111-01-24 18:03:00      5
 6   10003417  50983 2111-01-24 18:03:00    138
 7   10003417  50882 2111-02-18 13:34:00     23
 8   10003417  50902 2111-02-18 13:34:00    101
 9   10003417  50912 2111-02-18 13:34:00      2.8
10   10003417  50931 2111-02-18 13:34:00     90
```

```r
dbDisconnect(con)
```

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: The file `chartevents_filtered.csv.gz` which is after compressed contains 30195426 rows, the first 10 rows are displaied.

```
# Open dataset as arrow table
chartevents <- open_dataset("~/mimic/icu/chartevents.csv.gz", format = "csv")

# Filter and select relevant columns
chartevents_filtered <- chartevents |>
  select(subject_id, itemid, charttime, value) |>
  filter(itemid %in% c(220045, 220181, 220179, 223761, 220210)) |>
  collect()
```

```
# Save as a compressed CSV
write_csv_arrow(chartevents_filtered, "chartevents_filtered.csv.gz")
```

```
#Display first 10 rows
zcat < chartevents_filtered.csv.gz | head -10
#Display the number of rows
zcat < chartevents_filtered.csv.gz | tail -n +2 | wc -l
```

```
"subject_id","itemid","charttime","value"
10000032,223761,2180-07-23 06:00:00.000000-0800,"98.7"
10000032,220179,2180-07-23 06:11:00.000000-0800,"84"
10000032,220181,2180-07-23 06:11:00.000000-0800,"56"
10000032,220045,2180-07-23 06:12:00.000000-0800,"91"
10000032,220210,2180-07-23 06:12:00.000000-0800,"24"
10000032,220045,2180-07-23 06:30:00.000000-0800,"93"
10000032,220179,2180-07-23 06:30:00.000000-0800,"95"
10000032,220181,2180-07-23 06:30:00.000000-0800,"67"
10000032,220210,2180-07-23 06:30:00.000000-0800,"21"
 30195426
```