Coding Sample: Feature Engineering with Clinical Data

Author: Yue Li

Date: 01/24/2021

Introduction

"Sepsis is a major cause of death. Early aggressive treatment of this disease improves patient mortality. Henry et al. used readily available data from patient monitors and medical records to develop TREWScore, a targeted real-time early warning score that predicts in advance which patients are at risk for septic shock. With a median lead time of over 24 hours, this scoring algorithm may allow clinicians enough time to intervene before the patients suffer the most damaging effects of sepsis."

(Reference: "A targeted real-time early warning score (TREWScore) for septic shock" by Henry et al.)

Data resource: MIMIC III database

In this project, it shows how clinical data into sets of features for downstream statistical analysis.

I will extract features from EHRs, diagnosis codes, and more that can be used to predict the future development of septic shock.

0. Loading the packages

The first thing we need to do is load all of the packages we will use for this project.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 3.6.2

## Warning: package 'tibble' was built under R version 3.6.2

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'readr' was built under R version 3.6.2

## Warning: package 'purrr' was built under R version 3.6.2

## Warning: package 'dplyr' was built under R version 3.6.2

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
tidyverse_packages()
```

```
##  [1] "broom"      "cli"        "crayon"     "dbplyr"     "dplyr"
##  [6] "forcats"    "ggplot2"    "haven"      "hms"        "httr"
## [11] "jsonlite"   "lubridate"  "magrittr"   "modelr"     "pillar"
## [16] "purrr"      "readr"      "readxl"     "reprex"     "rlang"
## [21] "rstudioapi" "rvest"      "stringr"    "tibble"     "tidyr"
## [26] "xml2"       "tidyverse"
```

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.6.2
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

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

```
## Warning: package 'data.table' was built under R version 3.6.2
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:lubridate':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(Matrix)
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```r
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.2
```

```
## Loaded glmnet 4.0-2
```

```r
Sys.setenv(TZ='UTC')#change the default time zone
```

1. Defining labels for prediction

1.1

To predict, at 12 hours into an admission, whether septic shock will occur during the remainder of the admission, with at least 3 hours of lead time. This project is to engineer a set of features that may used as the inputs to a model that makes this prediction.

We will use the following definitions:

- We will only assign labels to admissions of at least **12 hours** in duration.
- An admission is assigned a **negative** label if septic shock does not occur at any time during the admission.
- An admission is assigned a **positive** label if septic shock occurs **15 hours after** admission or later.
- Admissions where the earliest time of septic shock occurs prior to fifteen hours after admission are removed from the study.
- For admissions that have valid labels, we assign an index time at twelve hours into the admission. For prediction, we only use information that occurs before the index time.
- In the case that a patient has multiple admissions for which a valid index time and label may be assigned, we only use the latest one.

To begin, given the above definitions, load `cohort_labels.csv` and `ADMISSIONS.csv` derive the binary classification labels for septic shock and the corresponding index times for each patient in the dataframe. The result should be a dataframe with one row per patient and additional columns for `index_time` and `label`.

```r
#Loading data
"cohort_labels.csv" %>% read_csv() -> cohort_labels
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   subject_id = col_double(),
##   hadm_id = col_double(),
##   icustay_id = col_double(),
##   charttime = col_datetime(format = ""),
##   sepsis = col_logical(),
##   severe_sepsis = col_logical(),
##   septic_shock = col_logical()
## )
```

```
"ADMISSIONS.csv" %>% read_csv() -> ADMISSIONS
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   ROW_ID = col_double(),
##   SUBJECT_ID = col_double(),
##   HADM_ID = col_double(),
##   ADMITTIME = col_datetime(format = ""),
##   DISCHTIME = col_datetime(format = ""),
##   DEATHTIME = col_datetime(format = ""),
##   ADMISSION_TYPE = col_character(),
##   ADMISSION_LOCATION = col_character(),
##   DISCHARGE_LOCATION = col_character(),
##   INSURANCE = col_character(),
##   LANGUAGE = col_character(),
##   RELIGION = col_character(),
##   MARITAL_STATUS = col_character(),
##   ETHNICITY = col_character(),
##   EDREGTIME = col_datetime(format = ""),
##   EDOUTTIME = col_datetime(format = ""),
##   DIAGNOSIS = col_character(),
##   HOSPITAL_EXPIRE_FLAG = col_double(),
##   HAS_CHARTEVENTS_DATA = col_double()
## )
```

```
#Data info:
dim(cohort_labels)
```

```
## [1] 8365524       7
```

```
head(cohort_labels,5)
```

```
## # A tibble: 5 x 7
##   subject_id hadm_id icustay_id charttime           sepsis severe_sepsis
##        <dbl>   <dbl>      <dbl> <dttm>              <lgl>  <lgl>
## 1          3  145834     211552 2101-10-20 16:40:00 FALSE  FALSE
## 2          3  145834     211552 2101-10-20 16:49:00 FALSE  FALSE
## 3          3  145834     211552 2101-10-20 19:12:00 FALSE  FALSE
## 4          3  145834     211552 2101-10-20 19:14:00 TRUE   TRUE
## 5          3  145834     211552 2101-10-20 19:15:00 TRUE   TRUE
## # ... with 1 more variable: septic_shock <lgl>
```

```
#str(cohort_labels)
dim(ADMISSIONS)
```

```
## [1] 58976    19
```

```
head(ADMISSIONS,5)
```

```
## # A tibble: 5 x 19
##   ROW_ID SUBJECT_ID HADM_ID ADMITTIME           DISCHTIME
##    <dbl>      <dbl>   <dbl> <dttm>              <dttm>
## 1     21         22  165315 2196-04-09 12:26:00 2196-04-10 15:54:00
## 2     22         23  152223 2153-09-03 07:15:00 2153-09-08 19:10:00
## 3     23         23  124321 2157-10-18 19:34:00 2157-10-25 14:00:00
## 4     24         24  161859 2139-06-06 16:14:00 2139-06-09 12:48:00
## 5     25         25  129635 2160-11-02 02:06:00 2160-11-05 14:55:00
## # ... with 14 more variables: DEATHTIME <dttm>, ADMISSION_TYPE <chr>,
## #   ADMISSION_LOCATION <chr>, DISCHARGE_LOCATION <chr>, INSURANCE <chr>,
## #   LANGUAGE <chr>, RELIGION <chr>, MARITAL_STATUS <chr>, ETHNICITY <chr>,
## #   EDREGTIME <dttm>, EDOUTTIME <dttm>, DIAGNOSIS <chr>,
## #   HOSPITAL_EXPIRE_FLAG <dbl>, HAS_CHARTEVENTS_DATA <dbl>
```

```
spetic_shock=cohort_labels %>% filter(septic_shock == TRUE)
dim(spetic_shock)
```

```
## [1] 59760     7
```

```
#Adding a new variable in Dataset ADMISSIONS:
#Length of stay (los)
ADMISSIONS=ADMISSIONS%>% mutate(los=difftime(DISCHTIME,ADMITTIME,units = "hours"))
cat("The type of los in ADMISSIONS is", mode(ADMISSIONS$los))
```

```
## The type of los in ADMISSIONS is numeric
```

```
head(ADMISSIONS,5)
```

```
## # A tibble: 5 x 20
##   ROW_ID SUBJECT_ID HADM_ID ADMITTIME           DISCHTIME
##    <dbl>      <dbl>   <dbl> <dttm>              <dttm>
## 1     21         22  165315 2196-04-09 12:26:00 2196-04-10 15:54:00
## 2     22         23  152223 2153-09-03 07:15:00 2153-09-08 19:10:00
## 3     23         23  124321 2157-10-18 19:34:00 2157-10-25 14:00:00
## 4     24         24  161859 2139-06-06 16:14:00 2139-06-09 12:48:00
## 5     25         25  129635 2160-11-02 02:06:00 2160-11-05 14:55:00
## # ... with 15 more variables: DEATHTIME <dttm>, ADMISSION_TYPE <chr>,
## #   ADMISSION_LOCATION <chr>, DISCHARGE_LOCATION <chr>, INSURANCE <chr>,
## #   LANGUAGE <chr>, RELIGION <chr>, MARITAL_STATUS <chr>, ETHNICITY <chr>,
## #   EDREGTIME <dttm>, EDOUTTIME <dttm>, DIAGNOSIS <chr>,
## #   HOSPITAL_EXPIRE_FLAG <dbl>, HAS_CHARTEVENTS_DATA <dbl>, los <drtn>
```

Based on the MIMIC database website, CHARTTIME represents the time recorded on the information system. It can be considered as end time or start time.

```
septic_shock_times=spetic_shock %>%
group_by(subject_id, hadm_id) %>%
summarise(min_septic_shock=min(charttime,na.rm=TRUE),max_septic_shock=max(charttime, na.rm=TRUE)) %>%
ungroup()
```

```
## `summarise()` has grouped output by 'subject_id'. You can override using the `.groups` argument.
```

```r
head(septic_shock_times,5)
```

```
## # A tibble: 5 x 4
##   subject_id hadm_id min_septic_shock    max_septic_shock
##        <dbl>   <dbl> <dttm>              <dttm>
## 1          3  145834 2101-10-20 20:04:00 2101-10-20 20:15:00
## 2         21  111970 2135-01-31 15:30:00 2135-01-31 15:35:00
## 3         38  185910 2166-08-10 13:00:00 2166-08-10 23:30:00
## 4         61  189535 2119-01-21 14:00:00 2119-02-02 07:26:00
## 5         68  108329 2174-01-09 09:00:00 2174-01-09 10:00:00
```

```r
#combine datasets ADMISSIONS and septic_shock_times
admission_septic_shock= ADMISSIONS %>%
  select(SUBJECT_ID:DISCHTIME, los)%>%
  left_join(septic_shock_times, by = c("SUBJECT_ID" = "subject_id", "HADM_ID" = "hadm_id"))
head(admission_septic_shock)
```

```
## # A tibble: 6 x 7
##   SUBJECT_ID HADM_ID ADMITTIME           DISCHTIME           los
##        <dbl>   <dbl> <dttm>              <dttm>              <drt>
## 1         22  165315 2196-04-09 12:26:00 2196-04-10 15:54:00  27.~
## 2         23  152223 2153-09-03 07:15:00 2153-09-08 19:10:00 131.~
## 3         23  124321 2157-10-18 19:34:00 2157-10-25 14:00:00 162.~
## 4         24  161859 2139-06-06 16:14:00 2139-06-09 12:48:00  68.~
## 5         25  129635 2160-11-02 02:06:00 2160-11-05 14:55:00  84.~
## 6         26  197661 2126-05-06 15:16:00 2126-05-13 15:00:00 167.~
## # ... with 2 more variables: min_septic_shock <dttm>, max_septic_shock <dttm>
```

```r
final_labels=admission_septic_shock %>%
  mutate(label = case_when(
      los < 12 ~ -1,
      is.na(min_septic_shock) ~ 0,
      difftime(min_septic_shock, ADMITTIME, units="hours") < 15 ~ -1,
      TRUE ~ 1)) %>%
  filter(label != -1) %>%
  mutate(index_time = ADMITTIME+hours(12)) %>%
  group_by(SUBJECT_ID) %>%
  filter(index_time == max(index_time)) %>%
  select(SUBJECT_ID, HADM_ID, index_time, label)

dim(final_labels)
```

```
## [1] 45254     4
```

```r
final_labels %>% pull(label) %>% table()
```

```
## .
##     0     1
## 42703  2551
```

Based on the result above, there are 2251 patients receive positive label and 42703 reveive negative label.

2. Building a Patient-Feature Matrix for the Septic Shock Cohort

Now that we know have derived labels and index times for each patient in our cohort, we can start to engineer some features from the data that occur prior to the index times and will be useful for predicting onset of septic shock.

Diagnoses

2.1

Let's first deal with diagnoses. Load `diagnoses_icd.csv`. We would like to find the diagnoses that occurred before the index time for each patient, but it looks like there is no time recorded in the diagnosis table.

Based on the website:[MIMIC III database]:(https://mit-lcp.github.io/mimic-schema-spy/index.html), "DISCHTIME" is the times of each diagnoses in Table:ADMISSIONS.

```
"diagnoses_icd.csv" %>%
read_csv() ->diagnoses
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   ROW_ID = col_double(),
##   SUBJECT_ID = col_double(),
##   HADM_ID = col_double(),
##   SEQ_NUM = col_double(),
##   ICD9_CODE = col_character()
## )
```

```
#data info:
head(diagnoses,5)
```

```
## # A tibble: 5 x 5
##   ROW_ID SUBJECT_ID HADM_ID SEQ_NUM ICD9_CODE
##    <dbl>      <dbl>   <dbl>   <dbl> <chr>
## 1   1297        109  172335       1 40301
## 2   1298        109  172335       2 486
## 3   1299        109  172335       3 58281
## 4   1300        109  172335       4 5855
## 5   1301        109  172335       5 4254
```

2.2

To filter the diagnoses for each patient that were recorded before the index time. The final result should have the columns `subject_id`, `hadm_id`, `diagnosis_time`, `icd9_code`, and `index_time`.

```
#1. Create a diagnosis table with diagnosis time
diagnosis_time=ADMISSIONS %>%
  inner_join(diagnoses, by = c("SUBJECT_ID","HADM_ID")) %>%
  mutate(diagnosis_time = DISCHTIME) %>%
  select(SUBJECT_ID, HADM_ID, diagnosis_time, ICD9_CODE)
```

```r
diagnosis_time %>% head()
```

```
## # A tibble: 6 x 4
##   SUBJECT_ID HADM_ID diagnosis_time      ICD9_CODE
##        <dbl>   <dbl> <dttm>              <chr>
## 1         22  165315 2196-04-10 15:54:00 9678
## 2         22  165315 2196-04-10 15:54:00 9693
## 3         22  165315 2196-04-10 15:54:00 E9502
## 4         22  165315 2196-04-10 15:54:00 E9503
## 5         22  165315 2196-04-10 15:54:00 3488
## 6         22  165315 2196-04-10 15:54:00 29620
```

```r
final_labels %>% head()
```

```
## # A tibble: 6 x 4
## # Groups:   SUBJECT_ID [6]
##   SUBJECT_ID HADM_ID index_time          label
##        <dbl>   <dbl> <dttm>              <dbl>
## 1         22  165315 2196-04-10 00:26:00     0
## 2         23  124321 2157-10-19 07:34:00     0
## 3         24  161859 2139-06-07 04:14:00     0
## 4         25  129635 2160-11-02 14:06:00     0
## 5         26  197661 2126-05-07 03:16:00     0
## 6         27  134931 2191-12-01 10:16:00     0
```

```r
#2. left join with final_labels
diagnoses_before_index=final_labels %>%
  left_join(diagnosis_time, by = c("SUBJECT_ID")) %>%
  filter(diagnosis_time < index_time)
head(diagnoses_before_index)
```

```
## # A tibble: 6 x 7
## # Groups:   SUBJECT_ID [1]
##   SUBJECT_ID HADM_ID.x index_time          label HADM_ID.y diagnosis_time
##        <dbl>     <dbl> <dttm>              <dbl>     <dbl> <dttm>
## 1         23    124321 2157-10-19 07:34:00     0    152223 2153-09-08 19:10:00
## 2         23    124321 2157-10-19 07:34:00     0    152223 2153-09-08 19:10:00
## 3         23    124321 2157-10-19 07:34:00     0    152223 2153-09-08 19:10:00
## 4         23    124321 2157-10-19 07:34:00     0    152223 2153-09-08 19:10:00
## 5         23    124321 2157-10-19 07:34:00     0    152223 2153-09-08 19:10:00
## 6         23    124321 2157-10-19 07:34:00     0    152223 2153-09-08 19:10:00
## # ... with 1 more variable: ICD9_CODE <chr>
```

```r
diagnoses_before_index %>%
  pull(SUBJECT_ID) %>%
  unique() %>% length()
```

```
## [1] 7242
```

2.3

What are the top 10 most common diagnosis codes (by number of unique patients who had the code in their history).

```
Rank=diagnoses_before_index %>%
  select(SUBJECT_ID,ICD9_CODE) %>%
  unique()%>%
  group_by(ICD9_CODE)%>%
  summarise(count=n())%>%
  arrange(-count)
head(Rank,10)
```

```
## # A tibble: 10 x 2
##    ICD9_CODE count
##    <chr>     <int>
##  1 4019       3217
##  2 4280       2412
##  3 41401      2008
##  4 42731      1995
##  5 5849       1794
##  6 25000      1577
##  7 51881      1371
##  8 5990       1335
##  9 2724       1288
## 10 2859       1070
```

```
tail(Rank,10)
```

```
## # A tibble: 10 x 2
##    ICD9_CODE count
##    <chr>     <int>
##  1 V741          1
##  2 V7651         1
##  3 V8489         1
##  4 V8521         1
##  5 V8522         1
##  6 V8523         1
##  7 V8534         1
##  8 V8535         1
##  9 V860          1
## 10 V8821         1
```

The top 3 ICD9_code are “4019”, “4280”,“41401”. ICD9_4091: Unspecified essential hypertension. ICD9_4280: Congestive heart failure, unspecified.

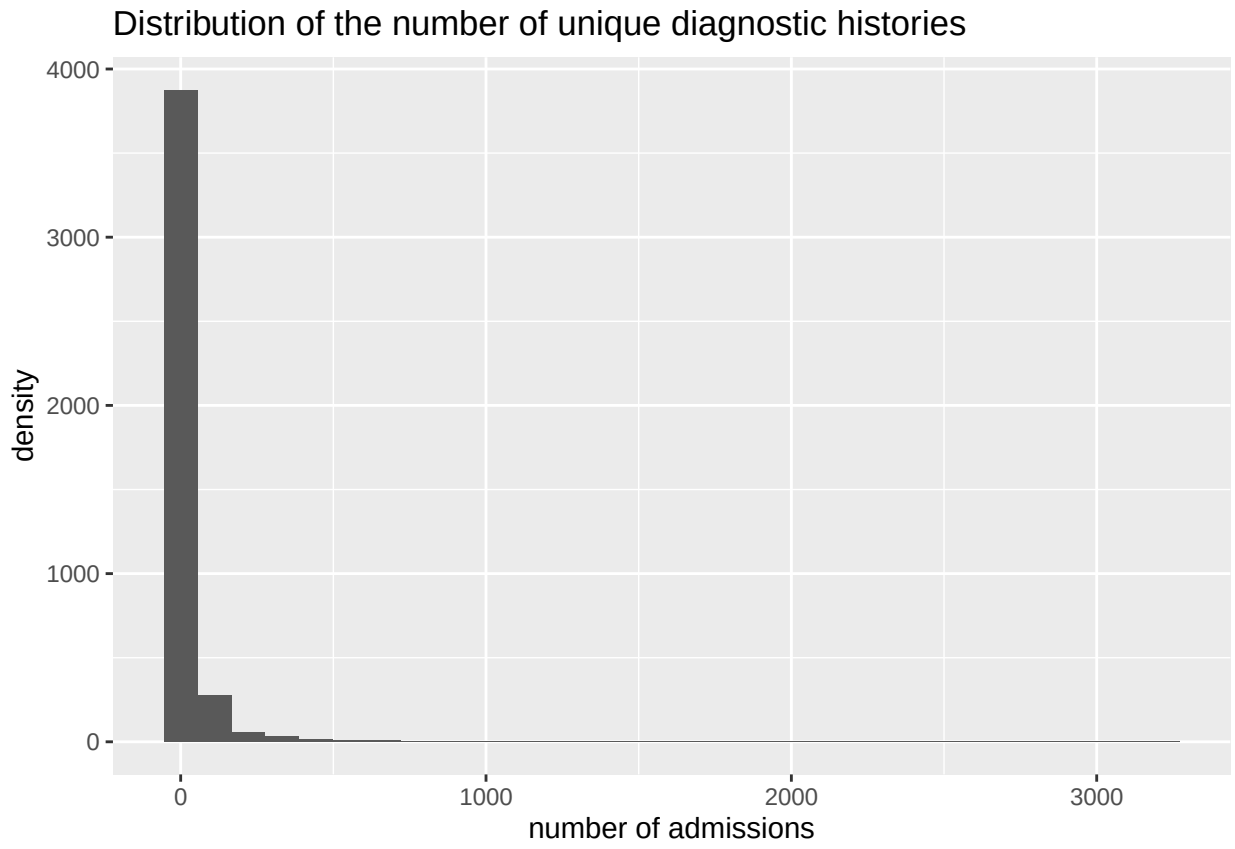
2.4

Generate a histogram of the count data which generated in 2.3. The x-axis represent the number of admissions that a code belongs to the history of and the y axis represent the number of codes that were observed in the same number of admissions.

In this picture, most diagnoses have less than 100 admissions.

```
Rank%>%
  ggplot(aes(x = count))+
  geom_histogram()+
  labs(x = "number of admissions",
       y = "density",
       title = "Distribution of the number of unique diagnostic histories")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



2.5

As you observed from the plot above, there are many rare diagnoses, resulting in a sparse feature space. One way to manage this is to identify rare (and similarly, very common) features using *Information content (IC)*. IC is a measure of specificity based on the frequency of occurrence of features.

The IC of a feature that occurs in a set of records is calculated as

$-log_2\left(\frac{count(\text{feature})}{count(\text{record})}\right)$

Use this equation to calculate the IC of ICD9 codes based on their occurrence in the diagnosis records for the sepsis cohort.

```
dim(Rank)
```

```
## [1] 4291    2
```

```
head(Rank)
```

```
## # A tibble: 6 x 2
##   ICD9_CODE count
##   <chr>     <int>
## 1 4019       3217
## 2 4280       2412
## 3 41401      2008
## 4 42731      1995
## 5 5849       1794
## 6 25000      1577
```

```
total_records=sum(Rank$count)
Rank=Rank%>% mutate(IC=-log2(count/total_records))
head(Rank)
```

```
## # A tibble: 6 x 3
##   ICD9_CODE count    IC
##   <chr>     <int> <dbl>
## 1 4019       3217  5.24
## 2 4280       2412  5.65
## 3 41401      2008  5.92
## 4 42731      1995  5.92
## 5 5849       1794  6.08
## 6 25000      1577  6.26
```

2.6

The IC is larger, the more specific of the ICD9 code. The information of ICs observed:

```
summary(Rank)
```

```
##   ICD9_CODE             count               IC
##  Length:4291        Min.   :   1.00   Min.   : 5.235
##  Class :character   1st Qu.:   1.00   1st Qu.:13.079
##  Mode  :character   Median :   3.00   Median :15.302
##                     Mean   :  28.24   Mean   :14.570
##                     3rd Qu.:  14.00   3rd Qu.:16.887
##                     Max.   :3217.00   Max.   :16.887
```

```
Rank %>%
  arrange(-IC) %>%
  pull(ICD9_CODE)%>%
  head(5)
```

```
## [1] "0030"  "0049"  "0051"  "00581" "00841"
```

The range of IC value is (5.235,16.887), 0030 is the most specific ICD9 codes.

2.7

Filter the set of ICD9 codes for the diagnoses associated with the set of admissions to those with an IC between 6 and 10.

```
A27=Rank%>%
  filter(IC >= 6 & IC <= 10)%>%
  pull(ICD9_CODE)
head(A27)
```

```
## [1] "5849"  "25000" "51881" "5990"  "2724"  "2859"
```

2.8

Now we have our diagnoses features and the times they occured for each patient. All that is left to do is to create a patient-feature matrix that summarizes and organizes the diagnoses features. In this matrix, each row is an patient and each column in a diagnosis code, time binned by whether or not it occured in the preceeding 6 months prior to the index time. In other words, we are going to generate two features for each diagnosis code where one feature represents the count of the number of times the code was observed in the six months prior to the index time and the other features represents the number of times that code was observed in the medical history older than six months.

We aim to generate a long three column matrix with the columns `subject_id`, `feature_name`, and `feature_value`.

```
icd9_patt <- str_c(A27, collapse = "|")
icd9_patt <- str_c("^(", icd9_patt,")$")
icd9_patt
```

```
## [1] "^(5849|25000|51881|5990|2724|2859|2720|486|53081|2851|2762|2449|496|5859|V5861|99592|40390|0389
```

```
diagnoses_before_index %>%
  mutate(icd9_feature =
    case_when(
      is.na(ICD9_CODE) ~ "None",
      !str_detect(ICD9_CODE,icd9_patt) ~ "None",
      difftime(index_time, diagnosis_time, units="days") <= 180 ~ paste(ICD9_CODE, "before_6", sep="_")
      TRUE ~ paste(ICD9_CODE, "after_6", sep="_"))) %>%
  group_by(SUBJECT_ID) %>%
  count(icd9_feature) %>%
  filter(icd9_feature != "None") %>%
  rename(subject_id = SUBJECT_ID, feature_name=icd9_feature, feature_value=n) %>%
  ungroup() -> cohort_diag_feature
```

```
head(cohort_diag_feature)
```

```
## # A tibble: 6 x 3
##   subject_id feature_name   feature_value
##        <dbl> <chr>                  <int>
## 1         17 2724_before_6              1
## 2         17 45829_before_6             1
## 3         21 25000_before_6             1
## 4         21 2720_before_6              1
## 5         21 2749_before_6              1
## 6         21 28521_before_6             1
```

```
dim(cohort_diag_feature)
```

```
## [1] 68678     3
```

Vitals

2.9

Now let's engineer some features from vital sign measurements that may also relevant to predicting septic shock.

Here we will work with the patient's heart rates. Load the file `vitals_cohort_sirs.csv`.

```
"vitals_cohort_sirs.csv" %>%
  read_csv()%>%
  filter(vital_id=="HeartRate") -> heart_rate
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   subject_id = col_double(),
##   hadm_id = col_double(),
##   icustay_id = col_double(),
##   charttime = col_datetime(format = ""),
##   valuenum = col_double(),
##   vital_id = col_character()
## )
```

```
heart_rate %>% dim()
```

```
## [1] 6199978       6
```

```
heart_rate %>% head()
```

```
## # A tibble: 6 x 6
##   subject_id hadm_id icustay_id charttime           valuenum vital_id
##        <dbl>   <dbl>      <dbl> <dttm>                 <dbl> <chr>
## 1          3  145834     211552 2101-10-20 19:30:00      151 HeartRate
## 2          3  145834     211552 2101-10-20 19:45:00      135 HeartRate
## 3          3  145834     211552 2101-10-20 20:00:00      143 HeartRate
## 4          3  145834     211552 2101-10-20 20:15:00      165 HeartRate
## 5          3  145834     211552 2101-10-20 20:30:00      168 HeartRate
## 6          3  145834     211552 2101-10-20 20:45:00      147 HeartRate
```

```
cohort_hrate=
  final_labels %>%
  inner_join(heart_rate, by = c("SUBJECT_ID" = "subject_id")) %>%
  filter(charttime < index_time)
head(cohort_hrate)
```

```
## # A tibble: 6 x 9
## # Groups:   SUBJECT_ID [1]
##   SUBJECT_ID HADM_ID index_time          label hadm_id icustay_id
##        <dbl>   <dbl> <dttm>              <dbl>   <dbl>      <dbl>
## 1         22  165315 2196-04-10 00:26:00     0  165315     204798
## 2         22  165315 2196-04-10 00:26:00     0  165315     204798
## 3         22  165315 2196-04-10 00:26:00     0  165315     204798
## 4         22  165315 2196-04-10 00:26:00     0  165315     204798
## 5         22  165315 2196-04-10 00:26:00     0  165315     204798
## 6         22  165315 2196-04-10 00:26:00     0  165315     204798
## # ... with 3 more variables: charttime <dttm>, valuenum <dbl>, vital_id <chr>
```

```
cohort_hrate %>%
  pull(HADM_ID) %>%
  unique() %>%
  length()
```

```
## [1] 28747
```

After this filtering step, 28747 admissions are left in the dataframe.

2.10

One feature of interest might be the latest value of the heart rate before the cutoff time. Make a dataframe with four columns: `subject_id`, `hadm_id`, `latest_heart_rate`, and `charttime`.

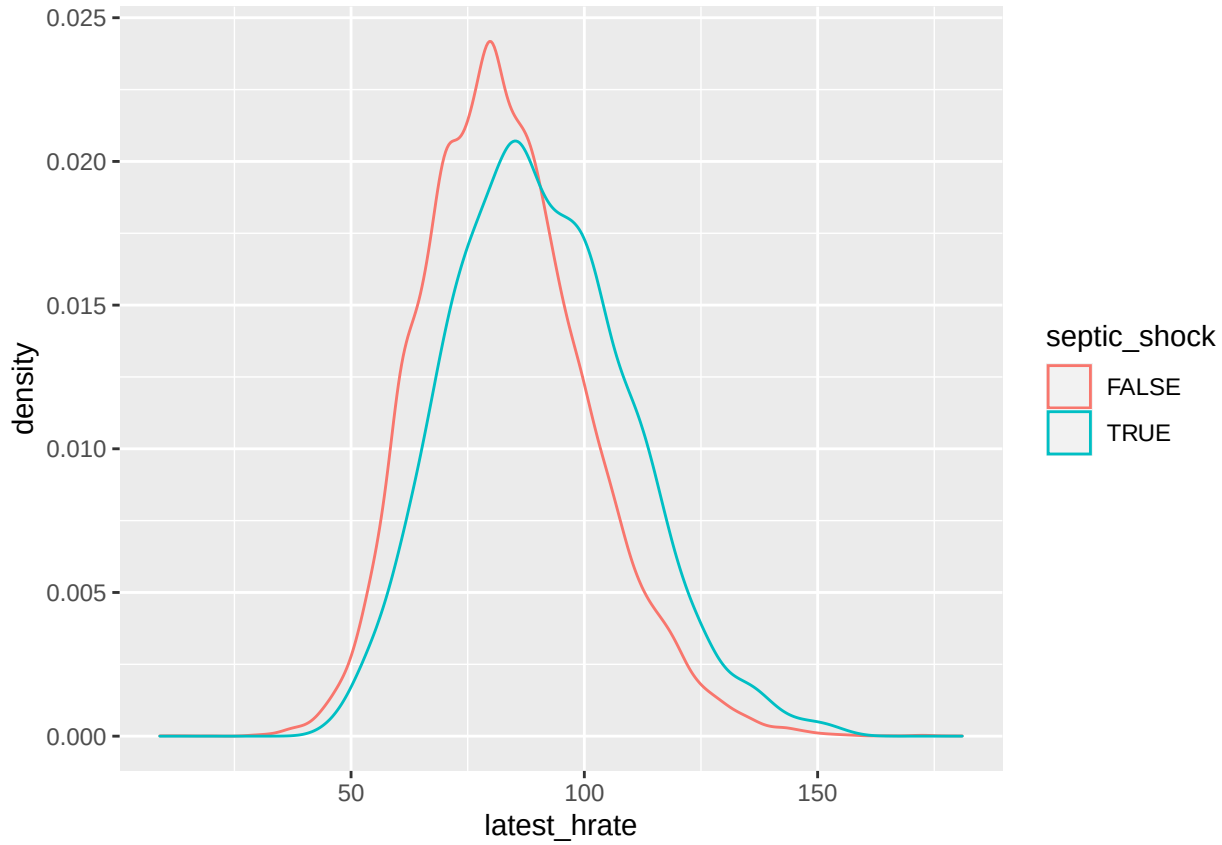
Make a density plot of the latest heart rate colored by whether a patient develops septic shock during the admission.

```
cohort_hrate_latest=cohort_hrate %>%
  group_by(SUBJECT_ID, HADM_ID, label) %>%
  filter(charttime == max(charttime)) %>%
  mutate(latest_hrate = valuenum) %>%
  ungroup()
```

```
cohort_hrate_latest %>% pull(latest_hrate) %>% mean()
```

```
## [1] 83.55496
```

```
cohort_hrate_latest %>%
  mutate(septic_shock = ifelse(label==1, TRUE, FALSE)) %>%
  ggplot(aes(x = latest_hrate, color=septic_shock))+
    geom_density()
```

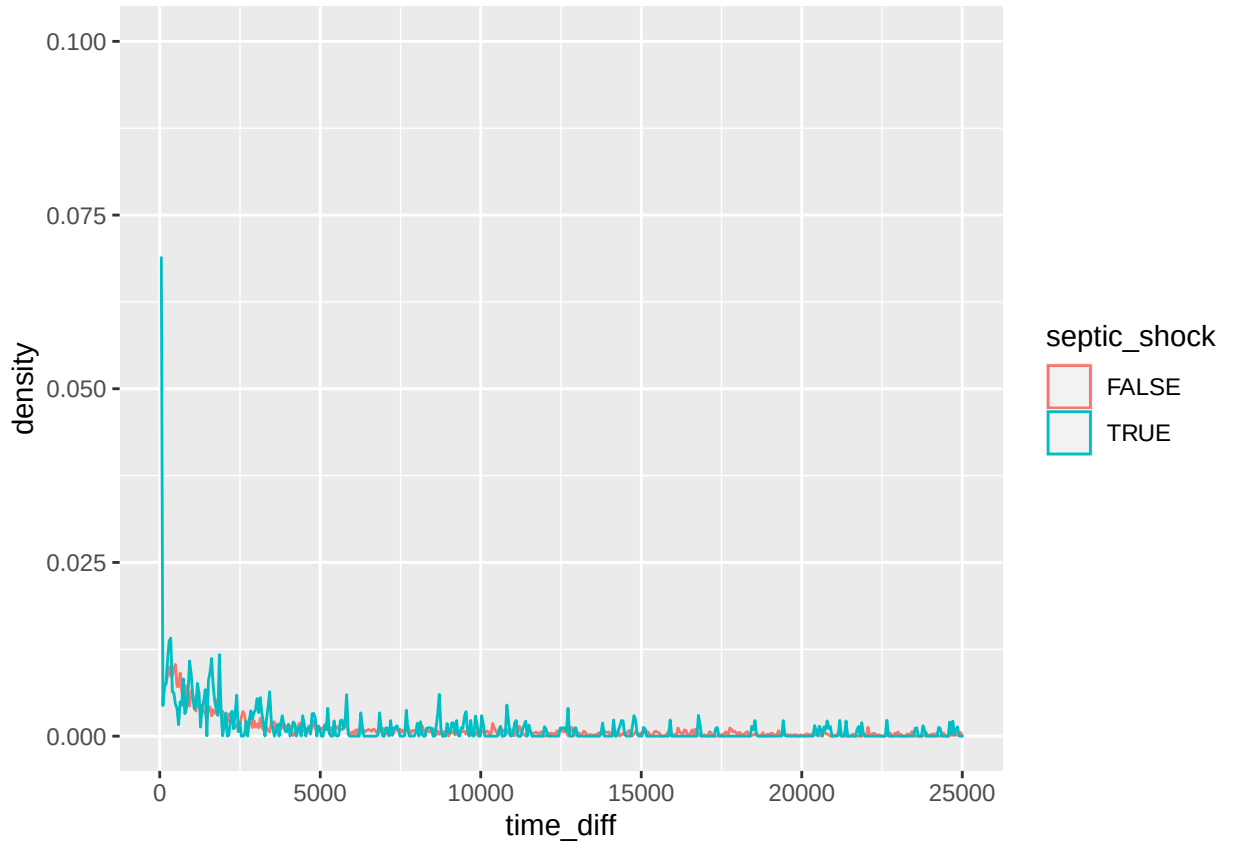


2.11

The latest recorded heart rate might not be a useful feature to use if the latest recording is not near the index time. Make a density plot of the time difference between the latest heart rate recording and the cutoff time colored by whether a patient develops septic shock during the admission.

```
cohort_hrate_latest %>%
  mutate(septic_shock = ifelse(label==1, TRUE, FALSE)) %>%
  mutate(time_diff = as.numeric(difftime(index_time,charttime, units="hours"))) %>%
  #filter(SUBJECT_ID==9)
  ggplot(aes(x = time_diff, color=septic_shock))+
    geom_density() +
    ylim(0,0.1) +
    xlim(0,25000)
```

```
## Warning: Removed 288 rows containing non-finite values (stat_density).
```




```
cohort_hrate_latest %>% head()
```

```
## # A tibble: 6 x 10
##   SUBJECT_ID HADM_ID index_time          label hadm_id icustay_id
##        <dbl>   <dbl> <dttm>              <dbl>   <dbl>      <dbl>
## 1         22  165315 2196-04-10 00:26:00     0  165315     204798
## 2         23  124321 2157-10-19 07:34:00     0  152223     227807
## 3         24  161859 2139-06-07 04:14:00     0  161859     262236
## 4         25  129635 2160-11-02 14:06:00     0  129635     203487
## 5         28  162569 2177-09-01 19:15:00     0  162569     225559
## 6         30  104557 2172-10-15 02:17:00     0  104557     225176
## # ... with 4 more variables: charttime <dttm>, valuenum <dbl>, vital_id <chr>,
## #   latest_hrate <dbl>
```

2.12

Some patients might have many heart rate recordings, and only using the last one might not be the best idea- it's possible the latest measurement is an outlier. Let's try to leverage all the heart rate measurements we have by creating a time-weighted average heart rate. Use the formula $w = e^{(-|\Delta t|-1)}$ to calculate the weights of each measurement, where Δt is the time difference between the measurement time and the cutoff time in hours. Calculate the weighted average with the formula $\bar{x}_w = \sum(x_i w_i)/\sum(w_i)$. The result should be a dataframe with two columns: `subject_id` and `time_wt_avg`.

```
cohort_hrate %>%
  mutate(time_diff = as.numeric(difftime(index_time,charttime, units="hours"))) %>%
```

```
  mutate(hrate_weight = exp(-abs(time_diff)-1)) %>%
  select(SUBJECT_ID,valuenum,hrate_weight) %>%
  group_by(SUBJECT_ID) %>%
  mutate(tot_wt = sum(hrate_weight)) %>%
  mutate(wt = ifelse(tot_wt > 0,hrate_weight, 1)) %>% ###handling divide by zero issue
  mutate(wt_hrate = sum(wt*valuenum)/sum(wt)) %>%
  select(SUBJECT_ID, wt_hrate) %>%
  unique() %>%
  ungroup() -> cohort_hrate_wt
cohort_hrate_wt %>% head()
```

```
## # A tibble: 6 x 2
##   SUBJECT_ID wt_hrate
##        <dbl>    <dbl>
## 1         22    107.
## 2         23     90.1
## 3         24     70.9
## 4         25     64.7
## 5         28     86.6
## 6         30     74.0
```

2.13

Let's do a sanity check to see if what we've done makes sense. We expect that the time-weighted average heart rate and the latest recorded heart rate should be similar.

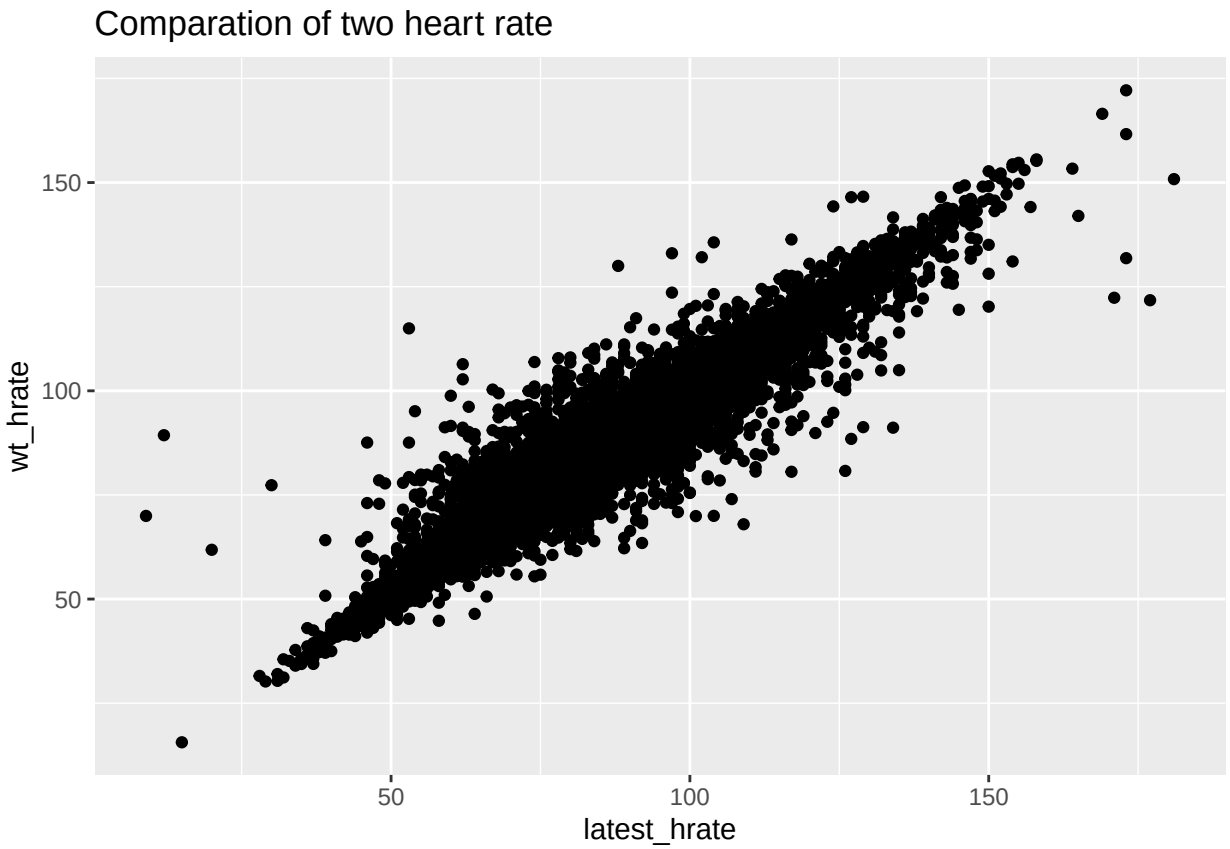
Make a scatterplot of the latest recorded heart rate (x-axis) and the time-weighted average heart rate (y-axis) of each patient.

```
cohort_hrate_final= cohort_hrate_latest %>%
  select(SUBJECT_ID, latest_hrate) %>%
  inner_join(cohort_hrate_wt, by = c("SUBJECT_ID"))

head(cohort_hrate_final)
```

```
## # A tibble: 6 x 3
##   SUBJECT_ID latest_hrate wt_hrate
##        <dbl>        <dbl>    <dbl>
## 1         22          105    107.
## 2         23           69     90.1
## 3         24           72     70.9
## 4         25           66     64.7
## 5         28           90     86.6
## 6         30           77     74.0
```

```
cohort_hrate_final %>%
  ggplot(aes(x=latest_hrate, y=wt_hrate)) +
    geom_point()+
  geom_point(size = 1, alpha = 0.5, color = "black")+
  labs(title = "Comparation of two heart rate")
```



2.14

Now we would like to condense our vital information into a three column dataframe with columns `subject_id`, `feature_name`, `feature_value`. Combine the latest heart rate and the time weighted heart rate dataframes to produce a dataframe that conforms to the specified format.

```
cohort_hrate_final %>%
  gather(latest_hrate, wt_hrate, key="feature_name", value="feature_value") %>%
  rename(subject_id = SUBJECT_ID) -> cohort_hrate_feature
```

```
cohort_hrate_feature %>%
  arrange(subject_id) %>%
  head()
```

```
## # A tibble: 6 x 3
##   subject_id feature_name feature_value
##        <dbl> <chr>                <dbl>
## 1          4 latest_hrate          86
## 2          4 wt_hrate              86.8
## 3          9 latest_hrate          92
## 4          9 wt_hrate              92.0
## 5         11 latest_hrate          78
## 6         11 wt_hrate              81.8
```

Stitching together Disease and Vitals Features

2.15

Our patient-feature matrix will simply be the amalgamation of the different feature matrices we've created. Use a full join to combine the feature matrices you derived from the diagnoses and heart rates measurements.

```
cohort_diag_feature %>% dim()
```

```
## [1] 68678     3
```

```
cohort_hrate_feature %>% dim()
```

```
## [1] 57496     3
```

```
cohort_diag_feature %>% head()
```

```
## # A tibble: 6 x 3
##   subject_id feature_name   feature_value
##        <dbl> <chr>                  <int>
## 1         17 2724_before_6              1
## 2         17 45829_before_6             1
## 3         21 25000_before_6             1
## 4         21 2720_before_6              1
## 5         21 2749_before_6              1
## 6         21 28521_before_6             1
```

```
cohort_hrate_feature %>% head()
```

```
## # A tibble: 6 x 3
##   subject_id feature_name feature_value
##        <dbl> <chr>                <dbl>
## 1         22 latest_hrate           105
## 2         23 latest_hrate            69
## 3         24 latest_hrate            72
## 4         25 latest_hrate            66
## 5         28 latest_hrate            90
## 6         30 latest_hrate            77
```

```
cohort_diag_feature %>%
  bind_rows(cohort_hrate_feature) %>%
  arrange(subject_id) -> all_features
```

```
all_features %>% pull(feature_name) %>% unique() %>% length()
```

```
## [1] 397
```

3. Classification

3.1 Logistic regression classifier

```r
features <- all_features
label_df <- final_labels %>% rename(subject_id = SUBJECT_ID)
```

```r
# Baseline implementation - provided
library(Matrix)
library(glmnet)

## Example of how to create sparse matrix
subject_map <- features %>% select(subject_id) %>% distinct() %>% mutate(subject_idx = 1:n())
feature_map <- features %>% select(feature_name) %>% distinct() %>% mutate(feature_idx = 1:n())
features <- features %>% left_join(subject_map) %>% left_join(feature_map)
```

```
## Joining, by = "subject_id"
```

```
## Joining, by = "feature_name"
```

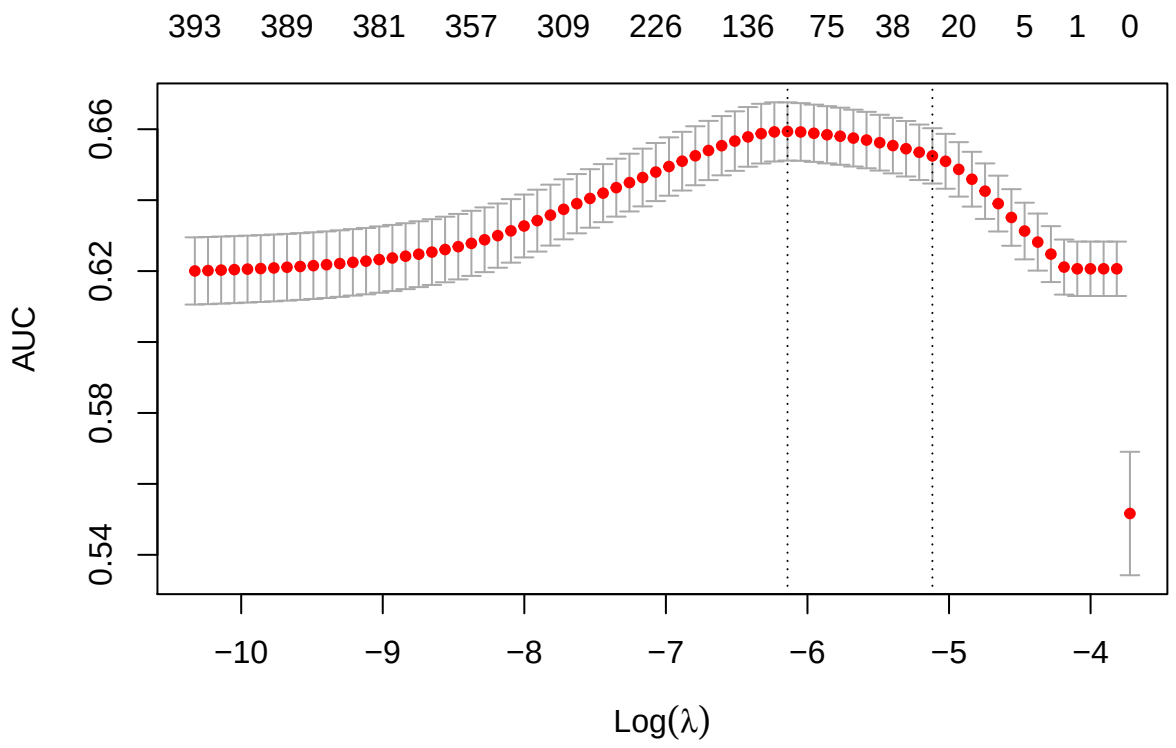
```r
subject_map <- subject_map %>% left_join(label_df[, c('subject_id', 'label')]) %>% mutate(label = as.fa
```

```
## Joining, by = "subject_id"
```

```r
## Create the sparse matrix
sparse_features <- sparseMatrix(i = features$subject_idx, j = features$feature_idx, x = features$featur
label_vector <- subject_map$label

## Fit the model with cross validation
model_cv <- cv.glmnet(sparse_features, label_vector, nfolds = 10, family = "binomial", type.measure = "

## Plot the result
plot(model_cv)
```




```
model_cv$lambda.min
```

```
## [1] 0.002153076
```

Done!