Homework 2: Working with Clincal Data

BIOMEDIN 215 (Data Driven Medicine), Fall 2016

Due: Thursday, October 27, 2016

In this assignment you will gain experience extracting and trasforming clinical data from the state it is stored in into datasets for later statistical analysis. You will practice using common time-saving tools in the `R` programming language that are ideally suited to these tasks.

You will use the MIMIC III database as a sandbox to create a dataset describing patients in the intensive care unit whose respiratory function rapidly deteriorated while under mechanical ventilation. P/F ratio referes to a ratio of Pao2 (arterial oxygen partial pressure) and Fio2 (fractional inspired oxygen), which is how much oxygen is reaching the patient's blood relative to how much is being artificially supplied by the mechanical ventillator. All of the data you need for this assignment is here, or can be found in the files section of Canvas.

Please edit this document directly using either Jupyter Notebook or R markdown in R Studio and answer each of these questions in-line. Jupyter and R markdown are useful tools for reproducible research that you will use over and over again in your later work. They are worth taking the short amount of time necessary to learn them. Turn in a single `.pdf` document showing all of your code and output for the entire assignment, with each question clearly demarcated. Submit your completed assignment through Canvas.

0. Getting Ready

a) The first thing we need to do is load all of the packages we will use for this assignment. Please load the packages `dplyr`, `tidyr`, `lubridate`, `stringr`, and `ggplot2`. Also, please run the command `Sys.setenv(TZ='UTC')`. Without it, your date-time data will misbehave.

```
library(dplyr)
library(tidyr)
library(lubridate)
library(stringr)
library(ggplot2)
Sys.setenv(TZ='UTC')
```

1. Building a Cohort Based on Inclusion Criteria

Loading Data

1.1 (5 pts)

The first part of any patient-level study is identify the patients who are relevant to the study, and at what point during their records they became elligible. Typically, this is done with a set of "inclusion critera", which, if met, qualify the patient for inclusion. In our study, we will consider the following inclusion criteria based on http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4649796: - Patients who were on mechanical ventilation - and have a P/F ratio > 250 for first 12 hours of ventilation - After at least the initial 12 hour period, P/F ratio drops below 300 and remains below 300 for > 3 hours

The process of identifying patients who meet certain clinical criteria is sometimes refered to as *electronic phenotyping*.

It seems that to do this, we will need to know who was on *mechanical ventilaion* and what their *P/F ratios* were over time. We need to find if and where those are recorded in the database by seeing if there is an internal code or ID assigned to them. Using the MIMIC docomumentation, determine what table(s) in MIMIC you would query to find out if and where mechanical ventillation or P/F ratios would be recorded.

MY ANWSER: I would query table CHARTEVENTS to filter out patients' ICU stays in which mechanical ventilations were performed. The query can be made based on ITEMID value which is an identifier for a single measurement type in the database. The presense of various measurements (for example, inspiratory pressure, minute volume, PCV value) is an indication that the patient was on mechanical ventilation. P/F ratio is the ratiom of Pao2 value and Fio2 value. Pao2 and Fio2 value can also be found in CHARTEVENT by querying on ITEMID. ITEMID of 490 and 779 indicates a measurement of Pao2. ITEMID of 189, 190, 2981, 7570 and 3420 indicates a measurement of FiO2.

1.2 *(5 pts)*

If you search that table using the measurement label and find counts of the measurements (from other tables) of the relevant results, we see that the duration of mechanical ventilation is not specified and P/F measurements are not directly recorded. However, there may be a way to create this data from what we do have. If we're lucky, other researchers have already done this work for us, most likely using SQL.

Search the web for *SQL* that uses *MIMIC* data to compute *mechanical ventilation duration* and for *SQL* to compute the *Pao2 Fao2 ratio*. Report the URLs of the sources you find.

MY ANWSER: https://github.com/MIT-LCP/mimic-code/blob/master/etc/ventilation-durations.sql
https://github.com/MIT-LCP/mimic-cookbook/blob/master/postgres/sofa_score_inserts.sql

1.3 *(5 pts)*

Looks like we're in luck this time, but it's often the case that nobody has been gracious enough to do this and you need to find clinical experts and database managers to work with in order to extract quantities of interest. The course team has done the work of running the SQL we found against the full MIMIC database and exporting the results to two csv files, called `pf.csv` and `vent.csv`. We'll use those tables to find our patients.

Load those two tables into R dataframes and use `head()` and to examine each of them. The tables are moderately sized, so don't worry if it takes a minute or two.

```
pf <- read.csv("../hw2/data/pf.csv", as.is = TRUE)
head(pf)
```

```
##   icustay_id subject_id         p_charttime p_value         f_charttime
## 1     200003      27513 2199-08-03 02:55:00      88 2199-08-03 02:45:00
## 2     200003      27513 2199-08-03 04:42:00      94 2199-08-03 03:00:00
## 3     200003      27513 2199-08-03 11:49:00     108 2199-08-03 11:00:00
## 4     200003      27513 2199-08-03 18:52:00      74 2199-08-03 12:00:00
## 5     200003      27513 2199-08-03 19:40:00     163 2199-08-03 19:00:00
## 6     200003      27513 2199-08-03 21:13:00     169 2199-08-03 20:00:00
##   f_value  pfratio
## 1     1.0  88.0000
## 2     1.0  94.0000
## 3     0.5 216.0000
## 4     0.5 148.0000
## 5     1.0 163.0000
## 6     0.7 241.4286
```

```
vent <- read.csv("../hw2/data/vent.csv", as.is =TRUE)
head(vent)
```

```
##   icustay_id           starttime             endtime
```

```
## 1     200003 2199-08-03 19:00:00 2199-08-07 14:00:00
## 2     200006 2159-09-03 14:00:00 2159-09-04 04:50:00
## 3     200009 2189-11-30 18:10:00 2189-12-01 10:00:00
## 4     200011 2188-08-06 01:00:00 2188-08-06 04:00:00
## 5     200014 2105-02-16 23:15:00 2105-02-17 18:00:00
## 6     200017 2138-03-17 22:00:00 2138-03-21 16:00:00
```

Using `dplyr` commands, make a dataframe of ICU stays where the patient had a stable start: i.e. they did not have any P/F ratios under 250 for the first 12 hours of their ventilation. Use `p_charttime` as the time of the P/F ratio measurement (given what you know about Pao2 and Fio2 measurements, think about why that makes sense). Exclude ICU stays without P/F ratio measurements in the first 12 hours of ventilation, and ignore P/F measurements that appear to be taken while the patient was not ventilated. The result should be a dataframe with a single column. Use `head()` to examine it. Perhaps something has gone wrong...

Identify the problem. Perhaps use `str()` to see what is going on in the intermediate results. Which variables are not represented correctly?

MY ANWSER: Yes, something went wrong. The p_charttime, starttime and endtime stored in the tables are in character type so I couldn't compare them directly or add 12 hours by using plus sign directly. I am not showing the buggy code here because the presence of not-working code would prevent R markdown file from creating PDF.

1.4 (5 pts)

Solve the problem using `as.POSIXct` and one or two other functions. Try your code again from above to find the ICU stays where the patient had a stable start. The result should be a dataframe with a single column.

```
# use vent table to extract the start_time of ventilation and calculate the 12 hour
# point after the start of ventilation for each icustay
first_12_hour <-
  vent %>%
  mutate(start_time = as.POSIXct(starttime), start_plus_12 = as.POSIXct(starttime) +
           12*60*60) %>%
  select(icustay_id, start_time, start_plus_12)

# a table containing the icustay_ids for icustay that has a stable first 12 hours
stable_12 <-
  pf %>%
  select(icustay_id, p_charttime, pfratio) %>%
  mutate(p_charttime=as.POSIXct(p_charttime)) %>%
  inner_join(first_12_hour, by = 'icustay_id') %>%
  filter(p_charttime>=start_time, p_charttime<start_plus_12) %>%
  group_by(icustay_id) %>%
  summarise(minimum_pf = min(pfratio)) %>%
  filter(minimum_pf > 250) %>%
  select(icustay_id)

nrow(stable_12)
```

```
## [1] 5260
```

How many ICU stays are there in your final list?
MY ANWSER: 5260.

Cohort Building

1.5 (10 pts)

Now we're going to do a bit of tricky table manipulation to find those stable-start ICU stays (what we did above) for which the patients additionally had a longer-than 3-hour period during which their P/F ratios remained under 300.

If you have experience with programming and algorithms, your first intuition might be to scan each ICU stay sequentially in a loop and march forward in time through all of the P/F ratios in each ICU stay to find 3+ hour-long periods with P/F ratios all under 300. That would be a reasonable approach in `C++`, but not in `R` or in a database setting. Generally, loops should be avoided at all cost in R. Thankfully, there's a fast and idiomatic way to do this that doesn't require any more tools than those we've already been using.

This approach can seem complicated if you're not used to thinking about data in terms of tables, so we'll go through it incrementally. First we will find all the PF measurements that are under 300 and happened 12 hours or more after the beginning of ventillation, but before the end of ventillation. Then we will build 3-hour long windows, starting with the measurements identified, and using a join with the original measurements we will see if any measurements in the window go above 300. If they do not, then that patient's stay in the ICU satisfies the inclusion criteria.

Continue using only the set of ICU stays that you identified above. Find the times of all of the PF measurements of people in this cohort that are under 300 and happened 12 hours or more after the beginning of ventilation, but before the end of ventilation. Use `dplyr` commands to create this dataframe. It should have the columns `icustay_id` and `p_charttime`.

```
# a dataframe contains the above seleted stable start icustays' 12 hour time point,
# and end of ventilation time
time12hr_andabove <-
  stable_12 %>%
  inner_join(vent, by='icustay_id') %>%  # icustay_id, starttime, endtime
  mutate(hour12_timepoint=as.POSIXct(starttime)+12*60*60, ventend=as.POSIXct(endtime)) %>%
  select(icustay_id, hour12_timepoint, ventend)
under300 <-
  pf %>%
  select(icustay_id, p_charttime, pfratio) %>%
  mutate(p_charttime=as.POSIXct(p_charttime)) %>%
  inner_join(time12hr_andabove, by='icustay_id') %>%
  filter(pfratio<300, p_charttime>=hour12_timepoint, p_charttime<ventend) %>%
  select(icustay_id, p_charttime)

nrow(under300)
```

```
## [1] 12073
```

How many rows does your table have?
MY ANWSER: 12073.

1.6 (20 pts)

Using a self-join, we will build the shortest possible time windows that begin and end with two PF values under 300 and which are longer than 3 hours. Using a join with the original measurements we will see if any measurements in the windows go above 300, and remove those windows that do.

Implement this using dplyr commands. The result should be a dataframe with three columns: `icustay_id`, `window_begin`, and `window_end`. The last two columns should be datetimes indicating the periods (windows) during that ICU stay during which there were no PF measurements made above 300.

```r
# shortest possible time windows that begin and end with two PF under 300
# and which are longer than 3 hours
non_overlap_bt3hr_window <-
  under300 %>%
  inner_join(under300, by='icustay_id') %>%
  filter(p_charttime.y-p_charttime.x>3*60*60) %>%
  # if start at the same time, keep the one that end time is smallest.
  group_by(icustay_id, p_charttime.x) %>%
  filter(min_rank(p_charttime.y)<=1) %>%
  # if end at the same time, keep the one that start time is largest
  group_by(icustay_id, p_charttime.y) %>%
  filter(min_rank(desc(p_charttime.x))<=1) %>%
  # give the columns proper names
  rename(window_begin = p_charttime.x, window_end = p_charttime.y)

# Using a join with the original measurements we will see if any measurements in the
# windows go above 300, and remove those windows that do.
# First, based on stable start icustays, filter out pfratio that are measured after
# 12 hour point and before vent end
pf_after12hour <-
  pf %>%
  transmute(icustay_id, p_charttime=as.POSIXct(p_charttime), pfratio) %>%
  inner_join(time12hr_andabove, by='icustay_id') %>%
  filter(p_charttime>=hour12_timepoint, p_charttime<ventend) %>%
  select(icustay_id, p_charttime, pfratio)
# Then, check if any pf ratio in a window is larger than 300
windows_no_above_300 <-
  non_overlap_bt3hr_window %>%
  inner_join(pf_after12hour, by='icustay_id') %>%
  # keep the rows where p_charttime is in window
  filter(p_charttime>=window_begin, p_charttime<=window_end) %>%
  # for each icustay's each non-overlap window, find the maximum pfratio,
  # if the maximum is no larger than 300, keep
  group_by(icustay_id, window_begin, window_end) %>%
  summarise(max_pf=max(pfratio)) %>%
  filter(max_pf<300) %>%
  select(icustay_id, window_begin, window_end)

nrow(windows_no_above_300)
```

```
## [1] 6526
```

How many rows does the result have?
MY ANWSER: 6526.

1.7 (20 pts)

Let's visualize what we've done so far. This should help you identify any errors that you might of made. It's often difficult to keep track of all the data manipulation, even for expert researchers! That's why it's good practice to plot things every once in a while as a sanity check.

Use `ggplot` and `facet_wrap` to show the entire PF ratio trajectories of the first 9 ICU stays in your windows dataframe. Plot the datetime on the x-axis and the PF value on the y-axis. Use color to

indicate PF measurements that are within or at the borders of your windows. Include a horizontal line on each plot at PF=300, and a green vertical line on each plot indicating the start of ventilation time plus 12 hours (the last example here http://docs.ggplot2.org/0.9.3.1/geom_vline.html and this question http://stackoverflow.com/questions/5388832/how-to-get-a-vertical-geom-vline-to-an-x-axis-of-class-date will come in handy). Your code may take a minute or two to run. *(20 pts)*

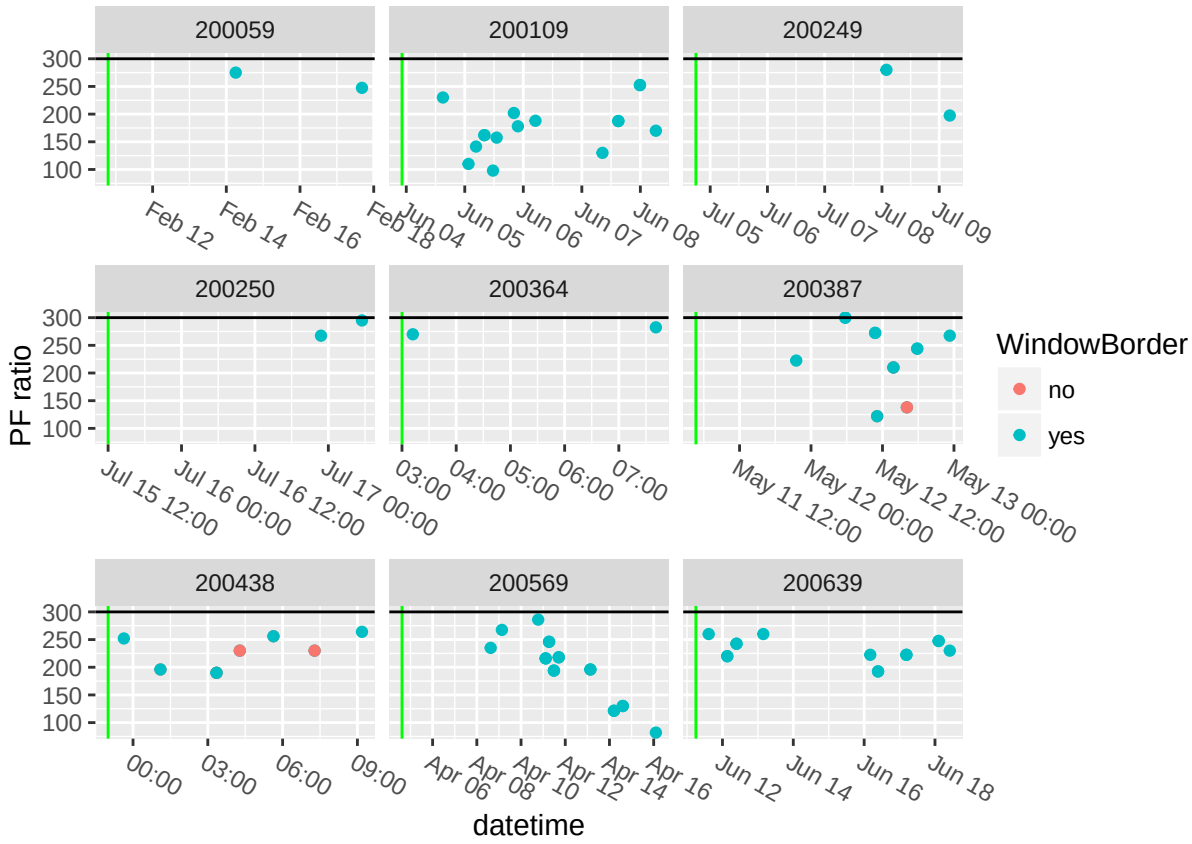
```
df_to_plot <-
  windows_no_above_300 %>%
  ungroup() %>%
  # since the table created above does not have pfratio, join and filter again
  inner_join(pf_after12hour, by='icustay_id') %>%
  filter(p_charttime>=window_begin, p_charttime<=window_end) %>%
  # create a varibale to indiate if pfratio is in window or at the boarder
  mutate(WindowBorder=ifelse(p_charttime==window_begin|p_charttime==window_end,
                             'yes','no')) %>%
  # to include the 12 hour time point in data frame
  inner_join(time12hr_andabove, by='icustay_id') %>%
  select(icustay_id, p_charttime, pfratio, WindowBorder, hour12_timepoint, ventend)

# filter(package %in% pkgname)
# first 9 icustay ids
icu_ids_9 <- c(200059, 200109, 200249, 200250, 200364, 200387, 200438, 200569, 200639)
df_to_plot_9 <-
  df_to_plot %>%
  filter(icustay_id %in% icu_ids_9)

# convert character variables to factor variables
df_to_plot_9$icustay_id <- as.factor(df_to_plot_9$icustay_id)
df_to_plot_9$WindowBorder <- as.factor(df_to_plot_9$WindowBorder)

# the purpose of this dataframe is to make sure all 9 plots' x-axis start from their
# vent start. It will be used for a geom_blank function which will not plot anything.
# Without this function, the plots will start from their first window starts instead
# of vent start.
dummy <-
  df_to_plot_9 %>%
  select(icustay_id, hour12_timepoint) %>%
  mutate(p_charttime=hour12_timepoint, pfratio=300) %>%
  group_by(icustay_id)

g <- ggplot(df_to_plot_9, aes(x=p_charttime, y=pfratio))
g+geom_point(aes(color=WindowBorder))+
  geom_vline(aes(xintercept=as.numeric(df_to_plot_9$hour12_timepoint)),df_to_plot_9,
             color='green') +
  geom_hline(yintercept = 300) +
  facet_wrap(~icustay_id, scales = "free_x", shrink=FALSE) +
  geom_blank(data = dummy) +
  xlab("datetime") + ylab("PF ratio") +
  scale_x_datetime() +
  theme(axis.text.x=element_text(angle = -30, hjust = 0))
```



Constructing your final cohort data frame

1.8 (10 pts)

We're at the last step. Now we know what patients seemed normal to start with, but later experienced a sustained drop in their PF ratios. If we're going to predict a treatment or outcome or see what treatments worked well for this kind of people, we need to establish the point in time at which the clinician would have concluded that the patient was experiencing the condition, starting the decision process of how to respond. If we don't do this, then we might accidentally use data from after the decision was made to try to predict the decision! In our case, let's say that time the end of the first window for each patient. We'll call that time the "index time" for each ICU stay.

Use `dplyr`, the results from above, and the file `icustays.csv` to create a dataframe containing three columns: `subject_id`, `icustay_id` and `index_time`. Some patients (subjects) may have had multiple qualifying ICU stays, so only use the first of them.

```
# read in icustays table
icustays <- read.csv("../hw2/data/icustays.csv", as.is = TRUE)
# extract icustay_id and subject_id
icu_subject <-
  icustays %>%
  select(icustay_id, subject_id)
# find index time (the end of the first window)
icu_sub_index <-
  windows_no_above_300 %>%
  group_by(icustay_id) %>%
```

```
  # create a new variable index_time, it equals to the first window_end, so min()
  mutate(index_time = min(window_end)) %>%
  select(icustay_id, index_time) %>%
  # remove duplicates
  distinct(.keep_all=TRUE) %>%
  inner_join(icu_subject, by='icustay_id') %>%
  # for patients have more than one icu, only keep the first
  group_by(subject_id) %>%
  arrange(icustay_id) %>%
  top_n(1, desc(icustay_id))

nrow(icu_sub_index)
```

```
## [1] 1010
```

How many ICU stays are there in the final cohort?
MY ANWSER: 1010.

2. Building a Patient-Feature Matrix for this Cohort

Now that we know what patients are relevant to our question, we can gather up their data. In case you're still having trouble with part I, you can simply load the `cohort.csv` file at this point and use that in your analysis instead of the result you produced.

Diagnoses

2.1 (3 pts)

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

```
cohort <- read.csv("../hw2/data/cohort.csv", as.is = TRUE)
diagnoses_icd <- read.csv("../hw2/data/diagnoses_icd.csv", as.is = TRUE)
```

What table in MIMIC would you use to find the times of each diagnoses? Use the online documentation to find out. *(3 pts)*
MY ANWSER: ADMISSION table. Specifically, the discharge time in admission table.

2.2 (4 pts)

This table is contained in `mystery.csv`. Load it into R and use it in conjunction with the diagnoses and cohort tables to filter the diagnoses for each patient that occured before the index date. Use the `dischtime` column as the time of diagnosis (think about why this makes sense, given that diagnoses are recored for billing). The final result should have the columns `subject_id`, `diagnosis_time`, `diagnosis`, and `index_time`.

```
mystery <- read.csv("../hw2/data/mystery.csv", as.is = TRUE)
# from mystery tabble, subject, admission, diagnosis time
sub_adm_dischtime <-
  mystery %>%
  select(subject_id, hadm_id, dischtime)
# from diagnoses_icd table, subject, admission, icd9 code, icustay id
```

```
sub_adm_diag_icu <-
  diagnoses_icd %>%
  select(subject_id, hadm_id, icd9_code, icustay_id)


sub_diagtime_diag_indext <-
  # combine all three tables
  cohort %>%  # subject, index time, icustay id
  inner_join(sub_adm_diag_icu, by = c("subject_id", "icustay_id")) %>%
  inner_join(sub_adm_dischtime, by = c('subject_id', 'hadm_id')) %>%
  # by now, table has subject_id, icustay_id, hadm_id, icd9_code, dischtime
  # keep diagnoses occur before the index time
  filter(dischtime<index_time) %>%
  rename(diagnosis_time = dischtime, diagnosis = icd9_code) %>%
  select(subject_id, diagnosis_time, diagnosis, index_time)

nrow(sub_diagtime_diag_indext)
```

```
## [1] 2684
```

How many rows are in the result?
MY ANWSER: 2684.

2.3 (4 pts)

What are the top 10 most common diagnosis codes (by number of patients who have had them) in the cohort from question 2.2? Google the top 3 codes and think about whether or not you think they make sense for this cohort. This is another kind of sanity check

```
icd9_group <-
  sub_diagtime_diag_indext %>%
  group_by(diagnosis) %>%
  summarise(n=n()) %>%
  arrange(desc(n))

icd9_top_10 <-
  icd9_group %>%
  head(10)
icd9_top_10
```

```
## # A tibble: 10 × 2
##    diagnosis     n
##        <chr> <int>
## 1       4280    88
## 2       4019    85
## 3      42731    78
## 4      41401    56
## 5      25000    49
## 6       5849    46
## 7      51881    39
## 8       5990    37
## 9        486    30
## 10      5119    30
```

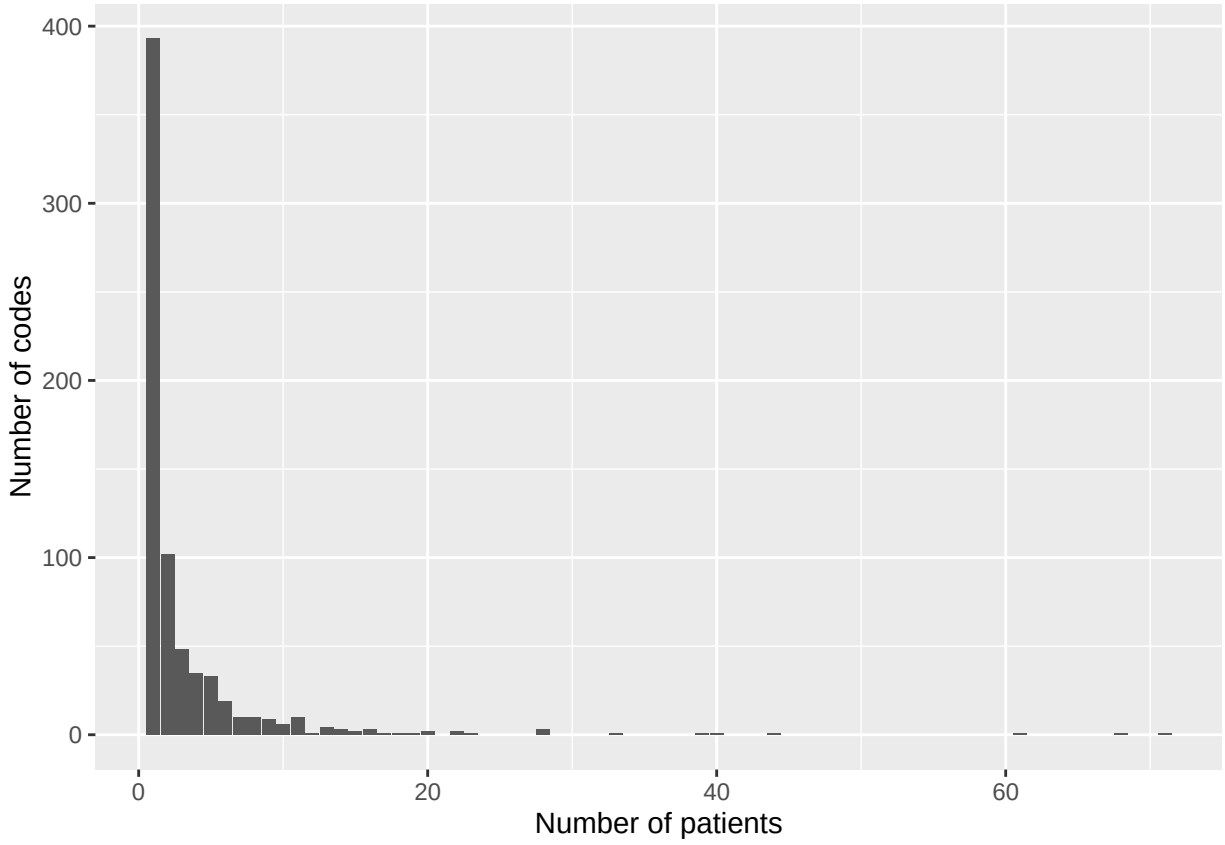
MY ANWSER: The top 3 most common icd9 diagnosis codes are: 4280 (Congestive heart failure), 4019 (Hypertension NOS) and 42731 (Atrial fibrillation). This makes sense because patients with these problems are mostly under mechanical ventilation.

2.4 (4 pts)

Make a plot of the number of codes that are present in N number of patients. The x-axis should be the number of patients, and the y-axis should be the number of codes that are present in that number of patients. I.e. plot the number of codes that appear in 1 patient, then the number of codes that appear in 2 patients, etc. In one sentence, dscribe the meaning of the plot in your own words. *(4 pts)*

```
df_to_plot_2.4 <-
  sub_diagtime_diag_indext %>%
  select(subject_id, diagnosis) %>%
  # one code might occur in a same patient multiple times, deduplicate for this
  distinct() %>%
  group_by(diagnosis) %>%
  summarise(n=n()) %>%
  arrange(desc(n))

ggplot(data=df_to_plot_2.4) +
  geom_bar(mapping=aes(x=n)) +
  labs(x='Number of patients', y='Number of codes')
```



MY ANWSER:
There are a huge number of codes that only occur in one patient and there are a few codes that occur in many patients. This indicates the diagnosis codes used in the dataset might be too specific. And this can cause a sparse feature space, greatly increase the dimension. Therefore, action to deal with this problem is required.

2.5 (2 pts)

As you observed from the plot you created above, rare diagnoses can result in a sparse feature space. One way to manage this is to group rare features into broader categories that are more common in the data. *Information content (IC)* is a measure of specificity based on the frequency of occurrence of features that can be used in combination with a *concept hierarchy* to identify broader categories of features.

The IC of a term that occurs in a set of documents is calculated as

$log_2\left(\frac{frequency(\text{term})}{count(\text{document})}\right)$

We have adapted this equation to calculate the IC of ICD9 codes and their parent concepts from the SNOMED CT concept hierarchy, based on their occurrence in Stanford EHRs:

$log_2\left(\frac{frequency(\text{ICD9 concept and descendants})}{count(\text{patients})}\right)$

In this next step, you will use ICs we have calculated in combination with SNOMED CT's concept hierarchy to aggregate ICD9 codes into their parent categories within a specific IC range.

First, load the following three files in R: `icd9_cui.csv`, `child_parent_cui.csv` and `cui_ic.csv` and examine them with `head()`. `icd9_cui.csv` contains ICD9 codes and their corresponding UMLS concept unique identifier (CUI). `child_parent_cui.csv` contains the transitive closure for each ICD9 CUI - that is, the set of all parent concepts for each ICD9 CUI (including itself). `cui_ic.csv` contains the IC values for ICD9 SNOMED CT CUIs.

```
icd9_cui <- read.csv("../hw2/data/icd9_cui.csv", as.is = TRUE)
child_parent_cui <- read.csv("../hw2/data/child_parent_cui.csv", as.is = TRUE)
cui_ic <- read.csv("../hw2/data/cui_ic.csv", as.is = TRUE)
```

Join `icd9_cui.csv` and `child_parent_cui.csv` to get all the parent concepts for each ICD9 code. How many parent concepts does ICD9 code 401.9 have?

```
icd9_parent <-
  icd9_cui %>%
  inner_join(child_parent_cui, by=c('cui'= 'child')) %>%
  distinct()
icd9_401.9 <- filter(icd9_parent, icd9==401.9)
nrow(icd9_401.9)
```

```
## [1] 10
```

MY ANWSER:
10 parent concepts including itself.

2.6 (2 pts)

What is the range (min and max) of ICs observed in the cui_ic file? What are the 10 most general CUIs?

```
cui_ic %>% summarise(min_ic = min(ic))  # minimum ic
```

```
##      min_ic
## 1 0.4280127
```

```r
cui_ic %>% summarise(max_ic = max(ic))  # maximum ic
```

```
##     max_ic
## 1 20.41637
```

```r
most_general_cui_10 <-
  cui_ic %>%
  arrange(ic) %>%
  head(10)
most_general_cui_10
```

```
##         cui        ic
## 1  C2720507 0.4280127
## 2  C0037088 0.4627634
## 3  C0012634 0.5238956
## 4  C1290906 0.6454774
## 5  C1290853 0.6694133
## 6  C1285159 0.7581572
## 7  C0578533 0.9020149
## 8  C0579140 1.7102572
## 9  C1290837 1.7189939
## 10 C1320355 2.1124498
```

MY ANWSER:
The range of ICs is [0.4280127, 20.41637]. The 10 most general CUIs are shown as the output above.

2.7 (3 pts)

For each ICD9 code, find its parent CUIs with an IC between 4 and 8, and keep only the most specific parent CUI. The result should be a table with the rows `icd9`, `cui`, `parent_cui` (the most specific parent cui with an IC between 4 and 8) and `ic` (which should be between 4 and 8).

```r
icd9_cui_4to8icparent <-
  icd9_parent %>%
  inner_join(cui_ic, by=c('parent'='cui')) %>%
  filter(ic>=4, ic<=8) %>%
  # one icd9 code can have multiple parent cui for it
  group_by(icd9) %>%
  # keep the most specific one, highest IC
  filter(ic==max(ic)) %>%
  ungroup() %>%
  transmute(icd9, cui, parent_cui=parent, ic)
```

2.8 (3 pts)

Use the resulting data frame to replace diagnoses in the dx_cohort with their parent CUI that is in the desired IC range. If there is no parent CUI or no parent CUI in the right IC range, leave the diagnosis as it is. How many diagnosis features do you have after aggregation by IC? If something seems off, check how the data is coded in MIMIC relative to how it is coded in the snomed files...

Code for converting MIMIC icd9 code to SNOMED icd9 code

```
############ Special Cases: this is to deal with the code spans over a range
# 327 -> C1561892
# 5401 -> C0577030
# numbers starts with 800 - 829 -> C3872870
# 042 -> C0042769

# this is a helper function to extract the last n character from a string
substrRight <- function(x, n){
  substr(x, nchar(x)-n+1, nchar(x))
}
### Convert the icd9 code in table (sub_diagtime_diag_indext) to the format of
# icd9 code in table (icd9_cui_4to8icparent)
replace_special <- sub_diagtime_diag_indext
for (i in 1:nrow(replace_special)){
  if (replace_special[i, 'diagnosis'] == "042"){
    replace_special[i, 'diagnosis'] = "C0042769"
  }else if (replace_special[i, 'diagnosis'] == "5401"){
    replace_special[i, 'diagnosis'] = "C0577030"
  }else if (replace_special[i, 'diagnosis']=="327"){
    replace_special[i, 'diagnosis'] = "C1561892"
  }else if (substr(replace_special[i, 'diagnosis'], 1, 1) == '8'){
    first_three = as.numeric(substr(replace_special[i, 'diagnosis'], 1, 3))
    if (first_three >= 800 & first_three <= 829){
      replace_special[i, 'diagnosis'] = "C3872870"
    }
  }else if (substr(replace_special[i, 'diagnosis'], 1, 1) == 'E'){
    if (nchar(replace_special[i, 'diagnosis']) == 4){
      replace_special[i, 'diagnosis'] = replace_special[i, 'diagnosis']
    }else if (substrRight(replace_special[i, 'diagnosis'], 1)=="0"){
      replace_special[i, 'diagnosis'] = substr(replace_special[i, 'diagnosis'],
                                               1, 4)
    }else{ # add a point between third and fourth digits
      replace_special[i, 'diagnosis'] = paste(substr(replace_special[i, 'diagnosis'],
                  1, 4), ".", substr(replace_special[i, 'diagnosis'], 5, 5), sep = "")
    }
  }else{
    if (nchar(replace_special[i, 'diagnosis']) > 3){
      n = nchar(replace_special[i, 'diagnosis'])
      if (n == 5 & substr(replace_special[i, 'diagnosis'], 4, n) == '00'){
        replace_special[i, 'diagnosis'] = substr(replace_special[i, 'diagnosis'],
                                                 1, 3)
      }else if(n == 5 & substr(replace_special[i, 'diagnosis'], 5, n) == '0'){
        replace_special[i, 'diagnosis'] = paste(substr(replace_special[i, 'diagnosis'],
                    1, 3), ".", substr(replace_special[i, 'diagnosis'], 4, 4), sep = "")
      }else if (n == 4 & substr(replace_special[i, 'diagnosis'], 4, n) == '0'){
        replace_special[i, 'diagnosis'] = substr(replace_special[i, 'diagnosis'],
                                                 1, 3)
      }else{
        replace_special[i, 'diagnosis'] = paste(substr(replace_special[i, 'diagnosis'],
                    1, 3), ".", substr(replace_special[i, 'diagnosis'], 4, n), sep = "")
      }
      if (substr(replace_special[i, 'diagnosis'], 1, 2) == '00'){
        n = n = nchar(replace_special[i, 'diagnosis'])
```

```
        replace_special[i, 'diagnosis'] = substr(replace_special[i, 'diagnosis'], 3, n)
      }else if(substr(replace_special[i, 'diagnosis'], 1, 1) == '0'){
        n = nchar(replace_special[i, 'diagnosis'])
        replace_special[i, 'diagnosis'] = substr(replace_special[i, 'diagnosis'], 2, n)
      }
    }
  }
}
```

replace_special is the table sub_diagtime_diag_indext with diagnosis code coverted to the same version as in table icd9_cui_4to8icparent.

```
# left join two tables, keep the icd9 code in replace_special if not occurs in
# icd9_cui_4to8icparent
icd9_to_cui <-
  replace_special %>%
  left_join(icd9_cui_4to8icparent, by=c('diagnosis'='icd9'))

# replace diagnosis with parent_cui if parent_cui is present
for (i in 1:nrow(icd9_to_cui)){
  if (!is.na(icd9_to_cui[i, 'parent_cui'])){
    icd9_to_cui[i,'diagnosis'] <- icd9_to_cui[i,'parent_cui']
  }
}
diag_table_with_general_cui <-
  icd9_to_cui %>%
  select(subject_id, diagnosis_time, diagnosis, index_time)

length(unique(diag_table_with_general_cui$diagnosis))
```

```
## [1] 587
```

MY ANWSER:
I have 587 diagnosis features after aggregation by IC.

2.9 (10 pts)

Now we have our list of diagnoses features and the times they occured for each patient. All that is left to do is to create the patient-feature matrix. Using `tidyr` and `dplyr`, make a patient-feature matrix where each row is a single patient and each column is a diagnosis code, time binned by whether or not it occured in the 6 months most recent to the index time. There should be two columns for every diagnosis- one that indicates how many times the patient had the code in her record in the past six months, and one that indicates how many times that patient had the code in her record before that time. The way to implement this is using a `mutate` to create a time bin indicator and then grouping on that before summarizing and spreading. Use `unite` before spreading to create a unique name for each feature based on its diagnosis code and time bin.

Note that the ICU stay is the first time many patients have been seen at this hospital, so most patients may have few or no prior recorded diagnoses.

```
# assume there are 180 days in 6 months
# create a dataframe with a variable relative_time indicating if the diagnosis time is
# within 6 months (in6m) before index time or even older (before6m)
with_time_indicator <-
```

```
  diag_table_with_general_cui %>%
  mutate(difference=as.numeric(difftime(as.POSIXct(index_time), as.POSIXct(diagnosis_time),
                                        units='days'))) %>%
  mutate(relative_time = ifelse(difference > 180, "before6m", "in6m")) %>%
  select(subject_id, diagnosis, relative_time)

diagnosis_matrix <-
  with_time_indicator %>%
  unite(diagnosis_relativetime, diagnosis, relative_time) %>%
  group_by(subject_id, diagnosis_relativetime) %>%
  mutate(n=n()) %>%
  ungroup() %>%
  distinct() %>%
  spread(key=diagnosis_relativetime, value=n)
# replace NA in the matrix with 0
diagnosis_matrix[is.na(diagnosis_matrix)] <- 0

dim(diagnosis_matrix)
```

```
## [1] 176 807
```

What are the dimensions of your resultant dataframe?

MY ANWSER:
The dimensions of my resultant dataframe is 176 rows and 807 columns.

Notes

2.10 (7 pts)

Now let's add features from notes. To do so, we'll have to process some text.

The `noteevents` table in MIMIC is too large and unweildy to load into R, so we've extracted the rows from that table that you will need by loading the cohort table into the database and using SQL. Write the SQL that you would use to get every row of the notes table that is for a patient in the cohort and that was written before the index date.
MY ANWSER:

```
SELECT subject_id, index_time
FROM cohort
INNER JOIN NOTEEVENTS
ON cohort.subject_id= NOTEEVENTS.SUBJECT_ID
WHERE NOTEEVENTS.CHARTDATE < cohort.index_time;
```

2.11 (3 pts)

The result is in the file `notes.csv`. Load it into R and examine it with `head()`.

```
notes <- read.csv("../hw2/data/notes.csv", as.is = TRUE)
```

UMLS terminologies provide concept hierarchies, as well as sets of terms for individual concepts. For example, there are more than 50 terms in UMLS terminologies for the concept 'myocardial infarction'!

In this step, you will use the SNOMED CT hierarchy and UMLS term sets to construct a dictionary of terms for respiratory system disorders, and then search for those terms in MIMIC III notes.

First, load `snomed_ct_isaclosure.csv` and `snomed_ct_str_cui.csv` in R, and examine them with `head()`. `snomed_ct_isaclosure.csv` contains the child-parent CUI relationships for all of SNOMED CT. `snomed_ct_str_cui.csv` contains the terms (each with a unique term identifier, tid) for each SNOMED CT CUI. These are large files, so loading them may take a minute.

```
snomed_ct_isaclosure <- read.csv("../hw2/data/snomed_ct_isaclosure.csv", as.is = TRUE)
snomed_ct_str_cui<- read.csv("../hw2/data/snomed_ct_str_cui.csv", as.is = TRUE)
```

Join `snomedct_isa_closure` with `snomedct_cui_string` to find all terms for each CUI (including the terms associated with its children).

```
cui_str <-
  snomed_ct_isaclosure %>%
  select(descendant, ancestor) %>%
  inner_join(snomed_ct_str_cui, by=c('descendant'='CUI')) %>%
  select(ancestor, str) %>%
  rename(cui=ancestor) %>%
  distinct()
```

2.12 (4 pts)

Find the CUI for respiratory system disorders in SNOMED CT, and construct a dictionary of all terms (a set of terms) corresponding to respiratory disorders that have 20 characters or fewer. How many terms are in the dictionary?

The CUI for respiratory system disorders in SNOMED CT is 'C0035204'.

```
RSD <-
  cui_str %>%
  filter(cui=='C0035204') %>%
  filter(nchar(str)<=20)
dict <- RSD$str
length(dict)
```

```
## [1] 1193
```

MY ANWSER:
There are 1193 terms in the dictionary.

2.13 (7 pts)

With the dictionary in hand, use `str_detect` from `stringr` and `sapply` from base R to add columns to your notes dataframe indicating whether or not text in that note matches one of the first fifty terms in your dictionary (limited for computational purposes). Your answer should have the columns `note_id` (the same as `row_id` in `notes.csv`), `subject_id`, `chartdate`, and as many more columns as there are terms in the dictionary (50).

```
first_50 <- dict[1:50]
notes_cols <-
  notes %>%
```

```
  select(row_id, subject_id, chartdate, text)
add_column <- function(term){
  str_detect(notes_cols$text, term)
}
notes_50_terms <-
  cbind(notes_cols, sapply(first_50,add_column)) %>%
  select(-text) %>%
  rename(note_id=row_id)
# Note: in notes_50_terms, values are TRUE or FALSE to indicate the presence or
# absence of the term in notes
dim(notes_50_terms)
```

```
## [1] 31944    53
```

What are the dimensions of the resulting dataframe?
MY ANWSER:
There are 31944 rows and 53 columns in the resulting dataframe.

2.14 (7 pts)

Now use `snomed_ct_concept_string` to convert terms back to their concepts and construct a dataframe of `subject_id`, `chartdate` and `concept`.

```
sub_chartdate_concept <-
  notes_50_terms %>%
  gather(`inhalation injury`:`fryns syndrome`, key=term, value=TorF) %>%
  filter(TorF==TRUE) %>%
  inner_join(cui_str, by=c('term'='str')) %>%
  select(subject_id, chartdate, cui) %>%
  rename(concept=cui)
```

2.15 (7 pts)

As with the diagnoses, we must transform this data into a patient-feature matrix. Use `dplyr` and `tidyr` to transform this table of concept mentions into a patient-feature matrix where each row is a patient and each column is the presence or absence of a concept. Do not do any time-binning. Each concept should have only one column. Instead of counts, use a binary indicator to indicate that the concept was present in the patient's notes.

```
note_matrix <-
  sub_chartdate_concept %>%
  select(subject_id, concept) %>%
  distinct() %>% # so the count will be 1 or 0, binary
  group_by(subject_id, concept) %>%
  mutate(n=n()) %>%
  spread(key=concept, value=n)
note_matrix[is.na(note_matrix)] <- 0
dim(note_matrix)
```

```
## [1] 80 61
```

What are the dimensions of the resulting table?
MY ANWSER:
There are 80 rows and 61 columns in the resulting dataframe.

Vitals

2.16 (4 pts)

Finally, let's try to engineer some features from vital sign measurements.

We will focus on heart rate measurements. In what table would you find heart rate measurements in MIMIC? Use the online documentation.
MY ANWSER:
In CHARTEVENTS table, I can find heart rate measurements by querying ITEMID equals to 212.

2.17 (5 pts)

As with the notes, we've done the work of extracting the patients' heart rates for you and filtering them for those that are before the index time. Load the table `heart_rates.csv` to proceed.

```
heart_rates <- read.csv("../hw2/data/heart_rates.csv", as.is = TRUE)
```

Let's do a sanity check. Make sure none of the measurements in this file were actually recorded after the index time or each patient. How many incorrect rows are there? If there are any, filter them out.

```
# inner_join heart_rates table with cohort table
# check if all records are good (meaning chartime < index time).
incorrect_rows <-
  cohort %>%
  inner_join(heart_rates, by=c('icustay_id', 'subject_id')) %>%
  filter(charttime > index_time)
nrow(incorrect_rows)
```

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

There is one incorrect row. Filter it out:

```
corrected_heart_rates <-
  cohort %>%
  inner_join(heart_rates, by=c('icustay_id', 'subject_id')) %>%
  filter(charttime < index_time)
```

2.18 (10 pts)

One feature of interest might be the latest value of the heart rate before the index time. Use `dplyr` to make a dataframe with three columns: `subject_id`, `latest_heart_rate`, and `charttime`.

```
latest_heart_rate_table <-
  corrected_heart_rates %>%
  group_by(subject_id) %>%
  filter(min_rank(desc(charttime))<=1) %>%
  select(subject_id, valuenum, charttime) %>%
  rename(latest_heart_rate=valuenum)
```

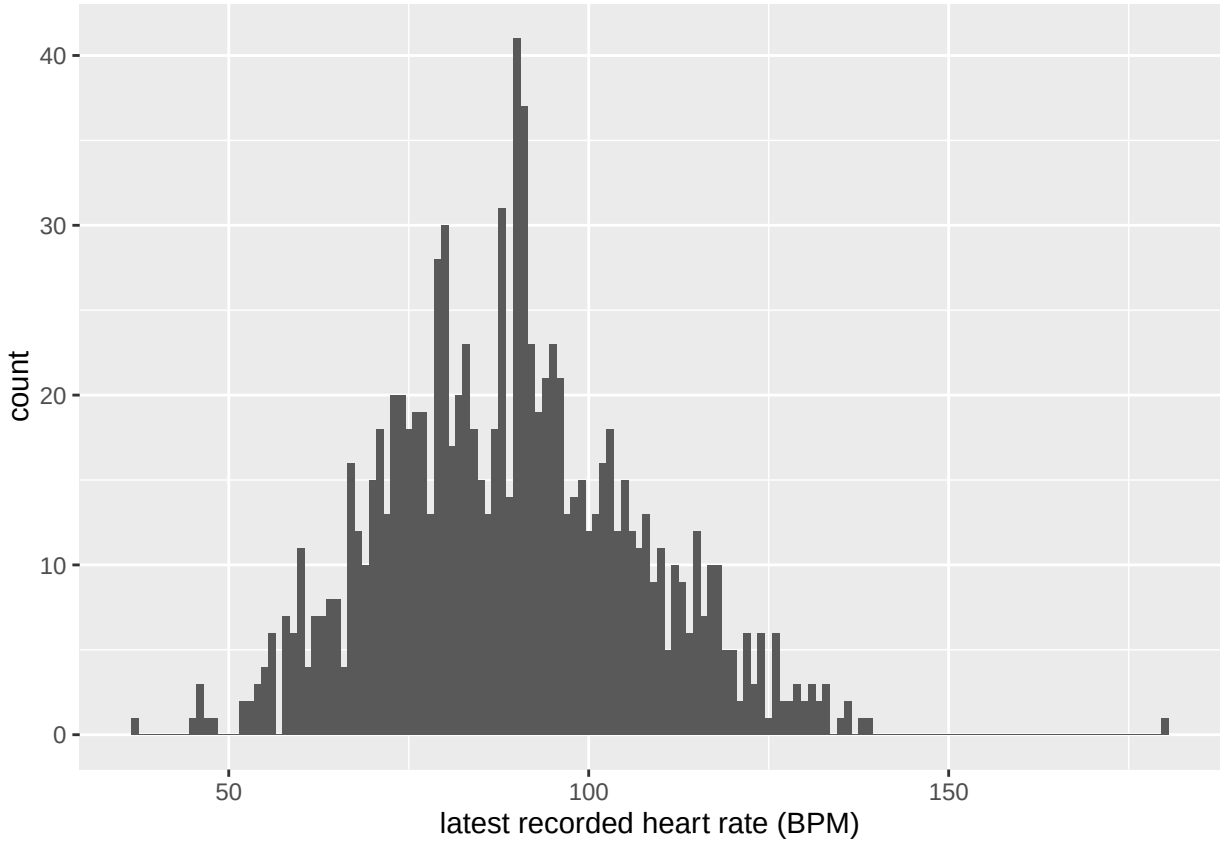
What is the average value of the latest recorded heart rate in this cohort? Make a histogram or density plot to convince yourself that the data are consistent with what you might expect heart rate measurements to be in this cohort. *(10 pts)*

```
mean(latest_heart_rate_table$latest_heart_rate)
```

```
## [1] 89.28172
```

The average value of the latest recorded heart rate in this cohort is 89.28172.

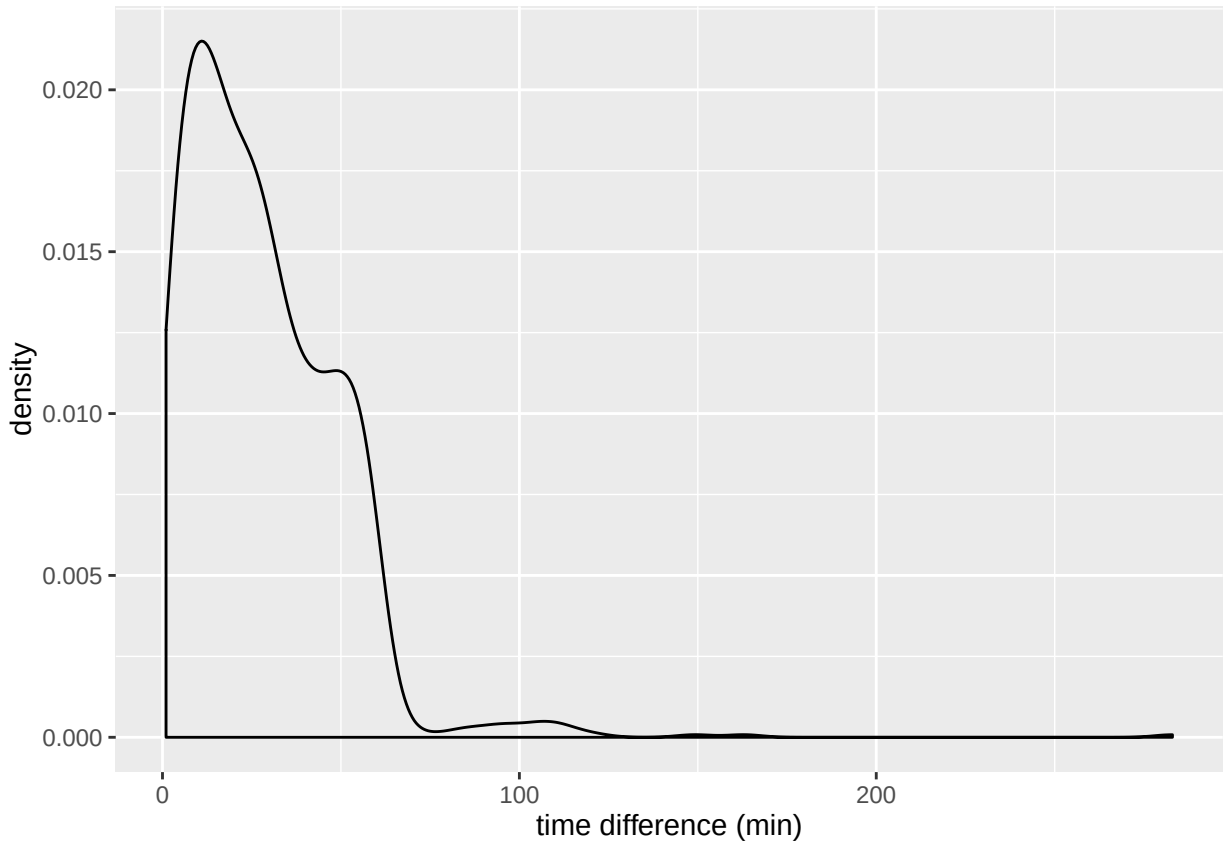
```
qplot(latest_heart_rate_table$latest_heart_rate,
      geom="histogram",
      binwidth = 1,
      xlab='latest recorded heart rate (BPM)'
      )
```



2.19 (5 pts)

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 index time.

```
time_diff <-
  corrected_heart_rates %>%
  group_by(subject_id) %>%
  filter(min_rank(desc(charttime))<=1) %>%
  mutate(difference=as.numeric(difftime(as.POSIXct(index_time), as.POSIXct(charttime),
                                        units='mins')))
ggplot(data=time_diff, aes(difference)) +
  geom_density() +
  labs(x='time difference (min)')
```



2.20 (7 pts)

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 index 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`. Beware of measurements that were noted but have no recorded value.

```
weighted_HR <-
  corrected_heart_rates[complete.cases(corrected_heart_rates),] %>%
  mutate(diff=as.numeric(difftime(as.POSIXct(index_time), as.POSIXct(charttime),
                                  units='hours'))) %>%
  mutate(w=exp(-diff-1)) %>%
  mutate(xw=valuenum*w) %>%
  select(subject_id, xw, w) %>%
  group_by(subject_id) %>%
  summarise_each(funs(sum)) %>%
  mutate(time_wt_avg=xw/w) %>%
  select(subject_id, time_wt_avg)

mean(weighted_HR$time_wt_avg)
```

```
## [1] 88.86548
```

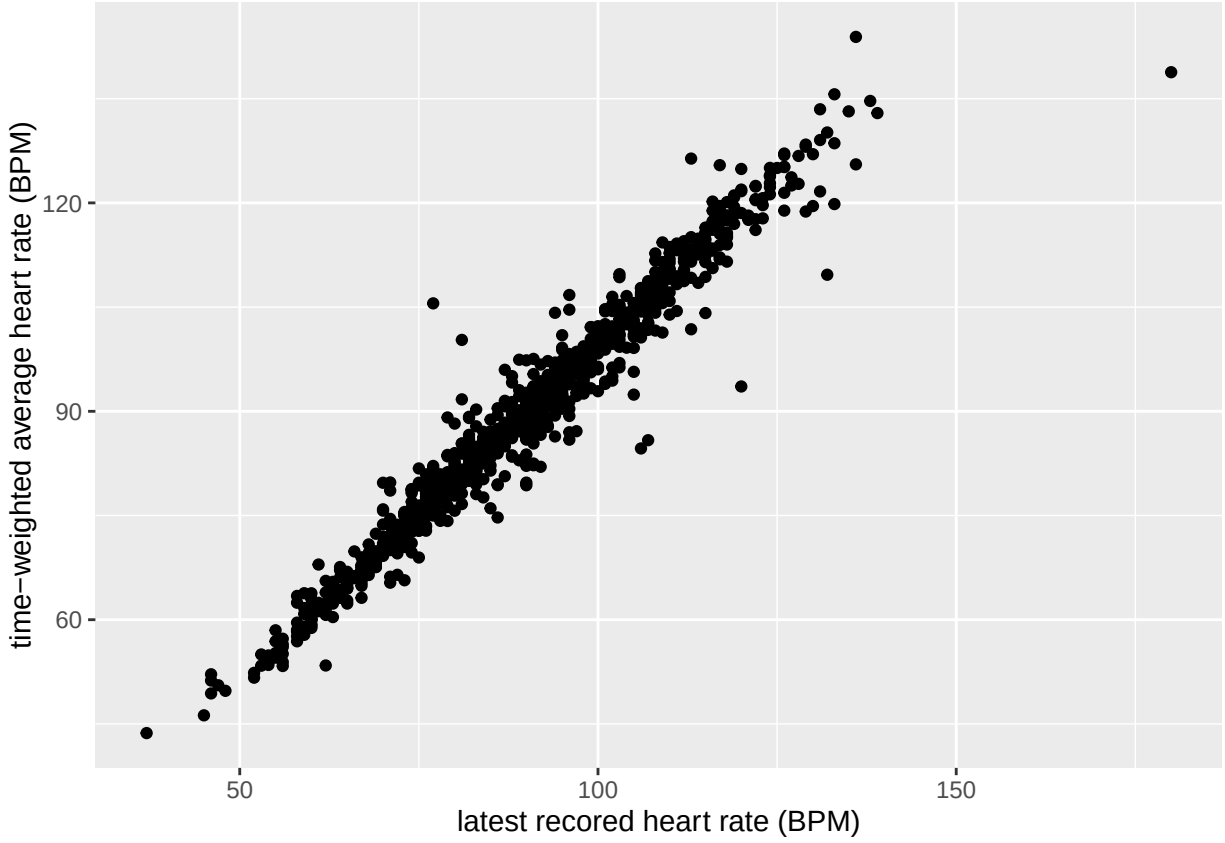
What is the average time-weighted average heart rate across all patients?
MY ANWSER:
The average time-weighted average heart rate across all patients is 88.86548.

2.21 (4 pts)

Again let's do a sanity check to see if what we've done makes sense. We should 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.

```
latest_avg <-
  latest_heart_rate_table %>%
  select(subject_id, latest_heart_rate) %>%
  inner_join(weighted_HR, by='subject_id')
ggplot(data=latest_avg) +
  geom_point(mapping=aes(x=latest_heart_rate, y=time_wt_avg)) +
  labs(x='latest recored heart rate (BPM)', y='time-weighted average heart rate (BPM)')
```



Joining the Features

2.22 (15 pts)

Our final patient-feature matrix will simply be the amalgamation of the different feature matrices we've created. Use an outer join to combine the columns of the feature matrices from diagnoses, notes, and heart

rate measurements. Not all patients have diagnoses or note features, so fill in any NA values with 0 to indicate that there were no diagnoses or notes counted. Use `names` to look at all the features and make sure everything seems ok.

```
patient_feature_matrix <-
  diagnosis_matrix %>%
  full_join(note_matrix) %>%
  full_join(weighted_HR)
```

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

```
patient_feature_matrix[is.na(patient_feature_matrix)] <- 0
names(patient_feature_matrix)
```

```
##   [1] "subject_id"        "117.9_in6m"        "12.15_before6m"
##   [4] "150.5_in6m"        "155.2_in6m"        "162.3_in6m"
##   [7] "162.4_before6m"    "162.8_in6m"        "162.9_in6m"
##  [10] "18.8_in6m"         "189_in6m"          "189.1_in6m"
##  [13] "198.1_before6m"    "198.5_in6m"        "198.82_before6m"
##  [16] "198.89_before6m"   "198.89_in6m"       "200_in6m"
##  [19] "202.8_before6m"    "202.8_in6m"        "205.01_in6m"
##  [22] "212.1_in6m"        "230.2_in6m"        "238.4_before6m"
##  [25] "238.7_in6m"        "251.8_in6m"        "253.5_in6m"
##  [28] "253.6_in6m"        "255.4_before6m"    "255.4_in6m"
##  [31] "263.9_before6m"    "263.9_in6m"        "266.2_before6m"
##  [34] "269.8_in6m"        "273.8_in6m"        "278_before6m"
##  [37] "278.01_before6m"   "278.01_in6m"       "278.1_in6m"
##  [40] "282.49_before6m"   "282.5_in6m"        "282.6_before6m"
##  [43] "283.19_in6m"       "284.1_in6m"        "284.8_before6m"
##  [46] "284.8_in6m"        "285.1_before6m"    "285.1_in6m"
##  [49] "285.21_before6m"   "285.21_in6m"       "285.22_in6m"
##  [52] "285.29_before6m"   "285.29_in6m"       "285.9_before6m"
##  [55] "285.9_in6m"        "287.4_before6m"    "287.4_in6m"
##  [58] "287.5_before6m"    "287.5_in6m"        "289_before6m"
##  [61] "289.81_in6m"       "291_in6m"          "292.12_in6m"
##  [64] "293_before6m"      "293_in6m"          "294_in6m"
##  [67] "294.8_before6m"    "294.8_in6m"        "295.8_in6m"
##  [70] "300_before6m"      "300_in6m"          "300.4_before6m"
##  [73] "300.4_in6m"        "305_before6m"      "305_in6m"
##  [76] "305.01_before6m"   "305.1_before6m"    "305.1_in6m"
##  [79] "305.5_in6m"        "305.6_in6m"        "309.28_in6m"
##  [82] "311_before6m"      "311_in6m"          "318.1_in6m"
##  [85] "323.9_before6m"    "331_before6m"      "331.4_in6m"
##  [88] "336.3_in6m"        "341.2_before6m"    "343.9_in6m"
##  [91] "344.1_before6m"    "344.1_in6m"        "344.9_before6m"
##  [94] "345.9_in6m"        "348.1_in6m"        "348.39_in6m"
##  [97] "348.4_in6m"        "348.5_in6m"        "348.8_in6m"
## [100] "349.82_before6m"   "349.82_in6m"       "350.1_in6m"
## [103] "355.9_in6m"        "357.2_before6m"    "357.2_in6m"
## [106] "357.7_before6m"    "362.01_before6m"   "362.5_in6m"
## [109] "369.3_in6m"        "369.6_before6m"    "38.11_before6m"
## [112] "38.11_in6m"        "38.3_before6m"     "38.42_before6m"
```

```
## [115] "38.43_before6m"    "38.49_before6m"    "38.8_in6m"
## [118] "38.9_before6m"     "38.9_in6m"         "394.1_in6m"
## [121] "396.2_before6m"    "396.3_before6m"    "396.3_in6m"
## [124] "398.91_before6m"   "398.91_in6m"       "401.9_before6m"
## [127] "401.9_in6m"        "41_in6m"           "41.04_in6m"
## [130] "41.11_before6m"    "41.11_in6m"        "41.3_before6m"
## [133] "41.4_before6m"     "41.4_in6m"         "41.6_before6m"
## [136] "41.7_before6m"     "41.7_in6m"         "41.85_before6m"
## [139] "41.85_in6m"        "41.86_before6m"    "41.86_in6m"
## [142] "411.1_before6m"    "411.1_in6m"        "414_before6m"
## [145] "414_in6m"          "414.01_before6m"   "414.01_in6m"
## [148] "414.02_in6m"       "414.8_before6m"    "414.8_in6m"
## [151] "423_in6m"          "423.1_in6m"        "423.3_in6m"
## [154] "423.9_before6m"    "423.9_in6m"        "424_before6m"
## [157] "424_in6m"          "424.1_before6m"    "424.1_in6m"
## [160] "429.79_in6m"       "429.9_before6m"    "432.1_in6m"
## [163] "433.01_in6m"       "433.1_before6m"    "437.2_before6m"
## [166] "437.5_in6m"        "441.01_in6m"       "441.2_before6m"
## [169] "441.4_before6m"    "441.4_in6m"        "442.2_before6m"
## [172] "442.3_before6m"    "442.84_before6m"   "442.84_in6m"
## [175] "443.81_in6m"       "443.9_before6m"    "443.9_in6m"
## [178] "444.1_in6m"        "444.22_before6m"   "447.6_before6m"
## [181] "447.8_in6m"        "451.82_before6m"   "453.4_before6m"
## [184] "453.4_in6m"        "453.8_before6m"    "453.8_in6m"
## [187] "456.2_before6m"    "456.2_in6m"        "456.21_before6m"
## [190] "456.8_before6m"    "457.1_in6m"        "459_in6m"
## [193] "459.81_in6m"       "466_in6m"          "466.19_before6m"
## [196] "478.29_in6m"       "482.1_before6m"    "482.1_in6m"
## [199] "482.2_before6m"    "482.2_in6m"        "482.41_before6m"
## [202] "482.41_in6m"       "482.83_in6m"       "486_before6m"
## [205] "486_in6m"          "491.21_before6m"   "491.21_in6m"
## [208] "496_before6m"      "496_in6m"          "507_before6m"
## [211] "507_in6m"          "510.9_in6m"        "512.1_before6m"
## [214] "512.1_in6m"        "516.8_in6m"        "517.8_before6m"
## [217] "518_before6m"      "518_in6m"          "518.4_before6m"
## [220] "518.4_in6m"        "518.5_before6m"    "518.5_in6m"
## [223] "518.81_before6m"   "518.81_in6m"       "518.82_before6m"
## [226] "518.82_in6m"       "518.83_before6m"   "518.83_in6m"
## [229] "518.84_before6m"   "518.84_in6m"       "518.89_before6m"
## [232] "518.89_in6m"       "519.02_before6m"   "519.1_before6m"
## [235] "519.1_in6m"        "522.4_before6m"    "527.2_before6m"
## [238] "53.9_before6m"     "531.9_before6m"    "532.9_in6m"
## [241] "537.84_before6m"   "537.89_before6m"   "552.21_before6m"
## [244] "552.9_in6m"        "553.1_in6m"        "553.21_in6m"
## [247] "553.3_before6m"    "555.9_before6m"    "555.9_in6m"
## [250] "557.1_in6m"        "560.1_before6m"    "560.1_in6m"
## [253] "560.2_before6m"    "560.81_before6m"   "564_before6m"
## [256] "564_in6m"          "564.1_before6m"    "567.2_before6m"
## [259] "567.2_in6m"        "567.21_before6m"   "567.21_in6m"
## [262] "567.22_in6m"       "567.8_before6m"    "568_in6m"
## [265] "568.81_in6m"       "569.61_before6m"   "569.69_before6m"
## [268] "569.83_in6m"       "569.84_in6m"       "569.85_in6m"
## [271] "573.3_in6m"        "573.4_in6m"        "575_before6m"
## [274] "576.1_before6m"    "576.1_in6m"        "576.2_before6m"
```

```
## [277] "576.8_before6m"    "576.8_in6m"        "581.81_in6m"
## [280] "583_in6m"          "583.81_before6m"   "583.81_in6m"
## [283] "583.89_in6m"       "585_in6m"          "585.3_before6m"
## [286] "585.6_before6m"    "585.6_in6m"        "585.9_before6m"
## [289] "585.9_in6m"        "593.2_before6m"    "593.2_in6m"
## [292] "593.9_before6m"    "593.9_in6m"        "596_in6m"
## [295] "596.54_before6m"   "596.54_in6m"       "596.8_in6m"
## [298] "598.8_in6m"        "599_before6m"      "599_in6m"
## [301] "599.7_before6m"    "599.7_in6m"        "608.89_before6m"
## [304] "614.6_in6m"        "680.2_in6m"        "682.2_before6m"
## [307] "682.2_in6m"        "682.3_before6m"    "682.3_in6m"
## [310] "682.6_before6m"    "682.6_in6m"        "682.7_before6m"
## [313] "696.1_in6m"        "70.3_in6m"         "70.32_in6m"
## [316] "70.41_in6m"        "70.54_before6m"    "70.54_in6m"
## [319] "70.7_before6m"     "70.7_in6m"         "703.8_in6m"
## [322] "709.01_before6m"   "710_before6m"      "710.2_in6m"
## [325] "710.4_before6m"    "710.9_in6m"        "711.06_in6m"
## [328] "711.09_in6m"       "712.33_before6m"   "714_before6m"
## [331] "715.36_before6m"   "715.9_in6m"        "715.96_before6m"
## [334] "719.03_in6m"       "723_before6m"      "724_before6m"
## [337] "724.02_in6m"       "724.2_before6m"    "730.05_before6m"
## [340] "730.08_before6m"   "730.16_in6m"       "730.27_before6m"
## [343] "731.8_before6m"    "733_before6m"      "733_in6m"
## [346] "733.13_before6m"   "751.69_in6m"       "759.89_in6m"
## [349] "780.2_before6m"    "780.39_before6m"   "780.39_in6m"
## [352] "780.52_in6m"       "780.57_before6m"   "780.57_in6m"
## [355] "780.59_before6m"   "780.6_before6m"    "780.99_in6m"
## [358] "782_before6m"      "783.7_in6m"        "785_before6m"
## [361] "785_in6m"          "785.4_in6m"        "785.51_before6m"
## [364] "785.51_in6m"       "785.52_before6m"   "785.52_in6m"
## [367] "786.3_before6m"    "787.6_before6m"    "787.91_in6m"
## [370] "788.2_before6m"    "788.2_in6m"        "788.21_before6m"
## [373] "788.5_before6m"    "789.06_before6m"   "789.07_before6m"
## [376] "789.5_before6m"    "789.5_in6m"        "79.99_in6m"
## [379] "790.01_in6m"       "790.5_before6m"    "790.5_in6m"
## [382] "790.6_before6m"    "790.7_before6m"    "790.7_in6m"
## [385] "790.92_before6m"   "790.92_in6m"       "790.93_before6m"
## [388] "792.1_before6m"    "792.1_in6m"        "796.3_in6m"
## [391] "799.02_before6m"   "799.02_in6m"       "8.45_before6m"
## [394] "8.45_in6m"         "83100_in6m"        "85220_in6m"
## [397] "85221_in6m"        "85222_in6m"        "85405_in6m"
## [400] "86121_in6m"        "86330_before6m"    "86350_before6m"
## [403] "86504_before6m"    "8748_before6m"     "88.81_in6m"
## [406] "8911_before6m"     "917.2_before6m"    "933.1_in6m"
## [409] "934.1_in6m"        "952_in6m"          "962.3_before6m"
## [412] "965.4_before6m"    "969.4_before6m"    "992_in6m"
## [415] "995.91_in6m"       "995.92_before6m"   "995.92_in6m"
## [418] "996.59_in6m"       "996.61_in6m"       "996.62_before6m"
## [421] "996.62_in6m"       "996.64_before6m"   "996.64_in6m"
## [424] "996.72_before6m"   "996.72_in6m"       "996.73_in6m"
## [427] "996.74_before6m"   "996.76_in6m"       "996.81_before6m"
## [430] "996.81_in6m"       "996.82_before6m"   "996.82_in6m"
## [433] "996.85_in6m"       "997.02_in6m"       "997.1_before6m"
## [436] "997.1_in6m"        "997.3_before6m"    "997.3_in6m"
```

```
## [439] "997.4_before6m"    "997.4_in6m"        "997.5_before6m"
## [442] "997.5_in6m"        "998_in6m"          "998.11_before6m"
## [445] "998.11_in6m"       "998.12_before6m"   "998.12_in6m"
## [448] "998.13_in6m"       "998.2_before6m"    "998.2_in6m"
## [451] "998.3_before6m"    "998.3_in6m"        "998.32_in6m"
## [454] "998.59_before6m"   "998.59_in6m"       "998.6_in6m"
## [457] "998.81_in6m"       "998.89_before6m"   "998.89_in6m"
## [460] "999.9_in6m"        "C0001973_before6m" "C0001973_in6m"
## [463] "C0002962_before6m" "C0002962_in6m"     "C0003811_before6m"
## [466] "C0003811_in6m"     "C0003850_in6m"     "C0004096_before6m"
## [469] "C0004096_in6m"     "C0004153_before6m" "C0004153_in6m"
## [472] "C0004936_in6m"     "C0005779_before6m" "C0005779_in6m"
## [475] "C0005956_before6m" "C0005956_in6m"     "C0006840_before6m"
## [478] "C0006840_in6m"     "C0007102_before6m" "C0008350_before6m"
## [481] "C0008350_in6m"     "C0009759_in6m"     "C0010266_before6m"
## [484] "C0011849_before6m" "C0011849_in6m"     "C0013221_before6m"
## [487] "C0014852_before6m" "C0014852_in6m"     "C0017181_before6m"
## [490] "C0017181_in6m"     "C0017417_before6m" "C0017601_before6m"
## [493] "C0017601_in6m"     "C0018099_before6m" "C0018099_in6m"
## [496] "C0018801_before6m" "C0018801_in6m"     "C0019112_in6m"
## [499] "C0020295_before6m" "C0020649_before6m" "C0020649_in6m"
## [502] "C0021831_in6m"     "C0021841_in6m"     "C0022660_before6m"
## [505] "C0022660_in6m"     "C0023510_in6m"     "C0028790_before6m"
## [508] "C0028790_in6m"     "C0030286_before6m" "C0030286_in6m"
## [511] "C0030846_in6m"     "C0031117_in6m"     "C0032231_before6m"
## [514] "C0032231_in6m"     "C0033578_before6m" "C0034067_before6m"
## [517] "C0034067_in6m"     "C0034069_before6m" "C0034069_in6m"
## [520] "C0034139_before6m" "C0034139_in6m"     "C0035078_in6m"
## [523] "C0037299_before6m" "C0037299_in6m"     "C0037932_in6m"
## [526] "C0042769_before6m" "C0042769_in6m"     "C0149516_in6m"
## [529] "C0149670_before6m" "C0149670_in6m"     "C0149725_before6m"
## [532] "C0149939_before6m" "C0151699_before6m" "C0151699_in6m"
## [535] "C0151971_before6m" "C0153606_before6m" "C0153675_before6m"
## [538] "C0153675_in6m"     "C0154251_before6m" "C0154251_in6m"
## [541] "C0154260_before6m" "C0154260_in6m"     "C0154728_in6m"
## [544] "C0155626_before6m" "C0155626_in6m"     "C0155668_before6m"
## [547] "C0155668_in6m"     "C0155671_before6m" "C0155671_in6m"
## [550] "C0155732_before6m" "C0155732_in6m"     "C0155795_in6m"
## [553] "C0156084_before6m" "C0156084_in6m"     "C0156189_before6m"
## [556] "C0156189_in6m"     "C0156191_before6m" "C0156191_in6m"
## [559] "C0156367_before6m" "C0156383_before6m" "C0157718_in6m"
## [562] "C0157738_before6m" "C0157738_in6m"     "C0158252_before6m"
## [565] "C0158252_in6m"     "C0158352_before6m" "C0158698_in6m"
## [568] "C0160918_in6m"     "C0162316_before6m" "C0162316_in6m"
## [571] "C0162429_in6m"     "C0175999_before6m" "C0175999_in6m"
## [574] "C0235527_before6m" "C0235527_in6m"     "C0235652_in6m"
## [577] "C0235653_in6m"     "C0238074_before6m" "C0238074_in6m"
## [580] "C0261615_before6m" "C0261615_in6m"     "C0261713_before6m"
## [583] "C0261794_before6m" "C0261794_in6m"     "C0264716_before6m"
## [586] "C0264716_in6m"     "C0264717_before6m" "C0264717_in6m"
## [589] "C0264886_before6m" "C0264886_in6m"     "C0264995_in6m"
## [592] "C0266805_in6m"     "C0267166_before6m" "C0267166_in6m"
## [595] "C0267994_before6m" "C0267994_in6m"     "C0268104_before6m"
## [598] "C0268104_in6m"     "C0268195_before6m" "C0268195_in6m"
```

```
## [601] "C0270292_before6m" "C0271903_before6m" "C0271903_in6m"
## [604] "C0271976_before6m" "C0271976_in6m"     "C0282609_before6m"
## [607] "C0282609_in6m"     "C0311364_before6m" "C0311364_in6m"
## [610] "C0346617_in6m"     "C0346627_in6m"     "C0376358_before6m"
## [613] "C0403604_before6m" "C0414083_in6m"     "C0425703_in6m"
## [616] "C0426330_in6m"     "C0426684_in6m"     "C0428712_in6m"
## [619] "C0432742_in6m"     "C0433004_in6m"     "C0455787_before6m"
## [622] "C0455787_in6m"     "C0476247_before6m" "C0476247_in6m"
## [625] "C0497538_before6m" "C0497538_in6m"     "C0549469_before6m"
## [628] "C0549469_in6m"     "C0554472_before6m" "C0561015_in6m"
## [631] "C0576482_in6m"     "C0577030_in6m"     "C0577255_before6m"
## [634] "C0577829_in6m"     "C0586559_before6m" "C0686582_before6m"
## [637] "C0686582_in6m"     "C0686619_before6m" "C0686619_in6m"
## [640] "C0700502_before6m" "C0700502_in6m"     "C0729531_in6m"
## [643] "C0748882_in6m"     "C0876973_before6m" "C0878544_before6m"
## [646] "C0878544_in6m"     "C1257958_before6m" "C1257958_in6m"
## [649] "C1272757_in6m"     "C1285162_before6m" "C1285390_before6m"
## [652] "C1285390_in6m"     "C1285425_before6m" "C1290383_before6m"
## [655] "C1290383_in6m"     "C1290889_before6m" "C1290889_in6m"
## [658] "C1291734_before6m" "C1298757_before6m" "C1442902_in6m"
## [661] "C1456434_before6m" "C1456434_in6m"     "C1510475_before6m"
## [664] "C1510475_in6m"     "C1532237_in6m"     "C1535926_before6m"
## [667] "C1561892_in6m"     "C1565662_before6m" "C1565662_in6m"
## [670] "C2198003_in6m"     "C2586211_before6m" "C2586211_in6m"
## [673] "C2937421_before6m" "C2937421_in6m"     "C3532905_in6m"
## [676] "C3536895_before6m" "C3536895_in6m"     "C3662261_before6m"
## [679] "C3662261_in6m"     "C3695318_before6m" "C3695318_in6m"
## [682] "C3714509_in6m"     "C3872870_before6m" "C3872870_in6m"
## [685] "E816.1_in6m"       "E849_before6m"     "E849.5_in6m"
## [688] "E849.7_in6m"       "E849.8_in6m"       "E850.2_before6m"
## [691] "E850.4_before6m"   "E853.2_before6m"   "E858_before6m"
## [694] "E878_in6m"         "E878.2_in6m"       "E878.6_before6m"
## [697] "E878.8_before6m"   "E878.8_in6m"       "E879_in6m"
## [700] "E879.8_in6m"       "E880.9_in6m"       "E884.2_before6m"
## [703] "E884.9_before6m"   "E888_in6m"         "E888.9_before6m"
## [706] "E888.9_in6m"       "E927_in6m"         "E928.8_in6m"
## [709] "E928.9_in6m"       "E930.8_before6m"   "E930.8_in6m"
## [712] "E931_in6m"         "E932_before6m"     "E932_in6m"
## [715] "E932.3_before6m"   "E934.2_before6m"   "E934.2_in6m"
## [718] "E939.3_in6m"       "E939.8_in6m"       "E942_in6m"
## [721] "E947.8_in6m"       "V02.59_in6m"       "V02.62_before6m"
## [724] "V04.81_before6m"   "V08_before6m"      "V08_in6m"
## [727] "V09_before6m"      "V09_in6m"          "V10.05_before6m"
## [730] "V10.05_in6m"       "V10.11_before6m"   "V10.11_in6m"
## [733] "V10.3_before6m"    "V10.3_in6m"        "V10.41_in6m"
## [736] "V10.46_before6m"   "V10.46_in6m"       "V10.47_in6m"
## [739] "V10.51_in6m"       "V10.79_before6m"   "V10.82_in6m"
## [742] "V10.83_in6m"       "V10.87_in6m"       "V12.01_before6m"
## [745] "V12.01_in6m"       "V12.51_before6m"   "V12.51_in6m"
## [748] "V12.59_before6m"   "V12.59_in6m"       "V13.01_in6m"
## [751] "V15.3_in6m"        "V15.81_before6m"   "V15.81_in6m"
## [754] "V15.82_before6m"   "V15.82_in6m"       "V16_before6m"
## [757] "V16.2_before6m"    "V16.41_in6m"       "V17.3_before6m"
## [760] "V17.3_in6m"        "V17.4_before6m"    "V17.5_before6m"
```

```
## [763] "V18_before6m"      "V18_in6m"          "V42_before6m"
## [766] "V42.2_before6m"    "V42.2_in6m"        "V42.5_in6m"
## [769] "V43.3_before6m"    "V43.3_in6m"        "V43.4_in6m"
## [772] "V43.64_before6m"   "V43.65_in6m"       "V44_in6m"
## [775] "V44.1_before6m"    "V44.1_in6m"        "V44.4_before6m"
## [778] "V45.01_before6m"   "V45.01_in6m"       "V45.02_in6m"
## [781] "V45.1_before6m"    "V45.1_in6m"        "V45.3_before6m"
## [784] "V45.61_before6m"   "V45.61_in6m"       "V45.77_in6m"
## [787] "V45.81_before6m"   "V45.81_in6m"       "V45.82_before6m"
## [790] "V45.82_in6m"       "V46.11_in6m"       "V46.2_before6m"
## [793] "V49.72_before6m"   "V49.75_before6m"   "V49.75_in6m"
## [796] "V49.76_in6m"       "V55.3_in6m"        "V58.61_before6m"
## [799] "V58.61_in6m"       "V58.65_before6m"   "V58.65_in6m"
## [802] "V58.67_before6m"   "V58.67_in6m"       "V58.69_in6m"
## [805] "V58.83_before6m"   "V59.6_before6m"    "V64.4_before6m"
## [808] "C0004096"          "C0004144"          "C0008679"
## [811] "C0009450"          "C0009566"          "C0009782"
## [814] "C0012634"          "C0024115"          "C0032226"
## [817] "C0032285"          "C0032290"          "C0035204"
## [820] "C0035243"          "C0037088"          "C0039978"
## [823] "C0149725"          "C0161836"          "C0205721"
## [826] "C0264220"          "C0264545"          "C0274432"
## [829] "C0277556"          "C0425442"          "C0521530"
## [832] "C0575479"          "C0577910"          "C0577914"
## [835] "C0577916"          "C0577939"          "C0578491"
## [838] "C0578533"          "C0579140"          "C0694550"
## [841] "C0729704"          "C0876973"          "C0949083"
## [844] "C1285159"          "C1285218"          "C1285331"
## [847] "C1285332"          "C1285333"          "C1285340"
## [850] "C1290325"          "C1290837"          "C1290853"
## [853] "C1290884"          "C1290890"          "C1290906"
## [856] "C1298756"          "C1320355"          "C1536731"
## [859] "C1761609"          "C2062952"          "C2720436"
## [862] "C2720507"          "C3266628"          "C3661979"
## [865] "C3661989"          "C3714636"          "C3873618"
## [868] "time_wt_avg"
```

How many total features are there? What is the correlation between the number of unspecified hypertension diagnoses in the past six months and the latest measured heart rate? *(15 pts)*

```
ncol(patient_feature_matrix)
```

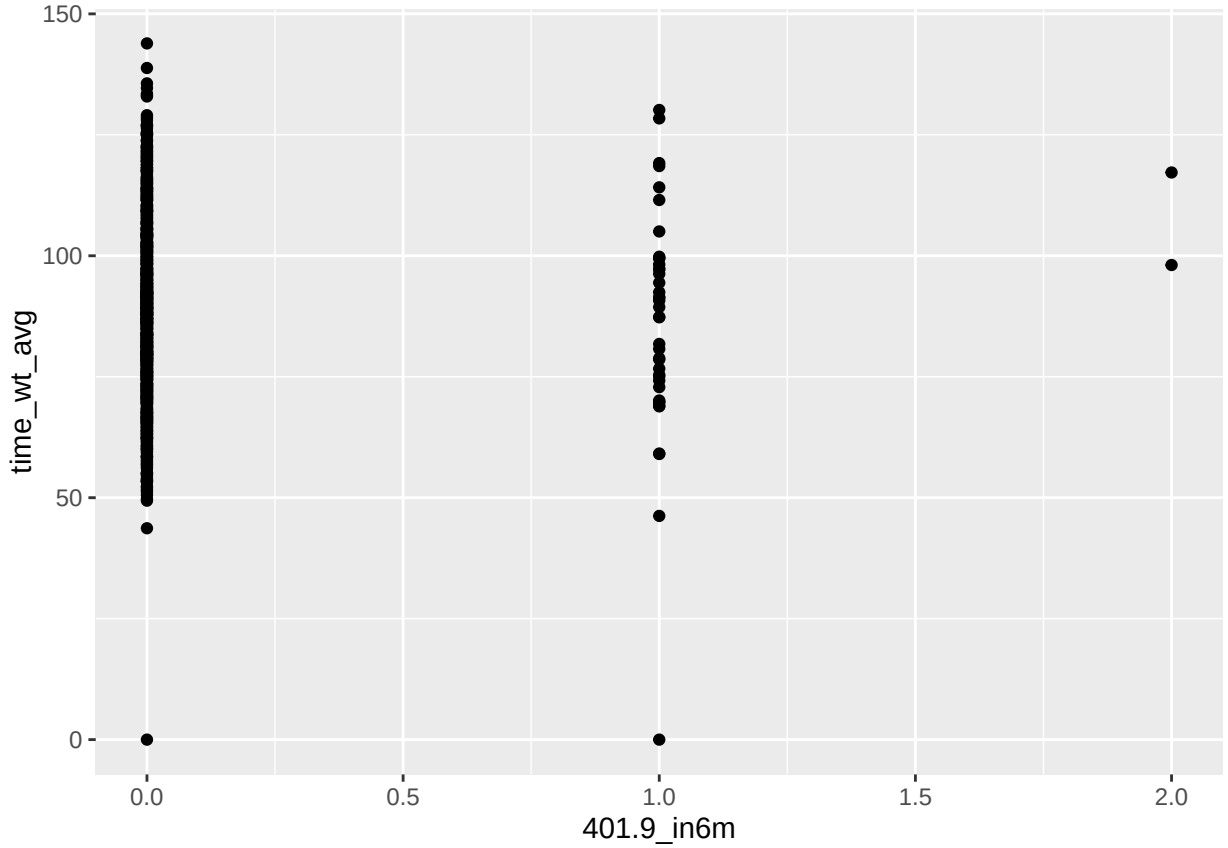
```
## [1] 868
```

There are 867 features for each patient (not including subject_id).

After searching the internet, I decided icd9 code equals to 401.9 is an indication of unspecified hypertension diagnosis. Therefore, from patient_feature_matrix, extract the columns '401.9_in6m' and time_wt_avg for analysis.

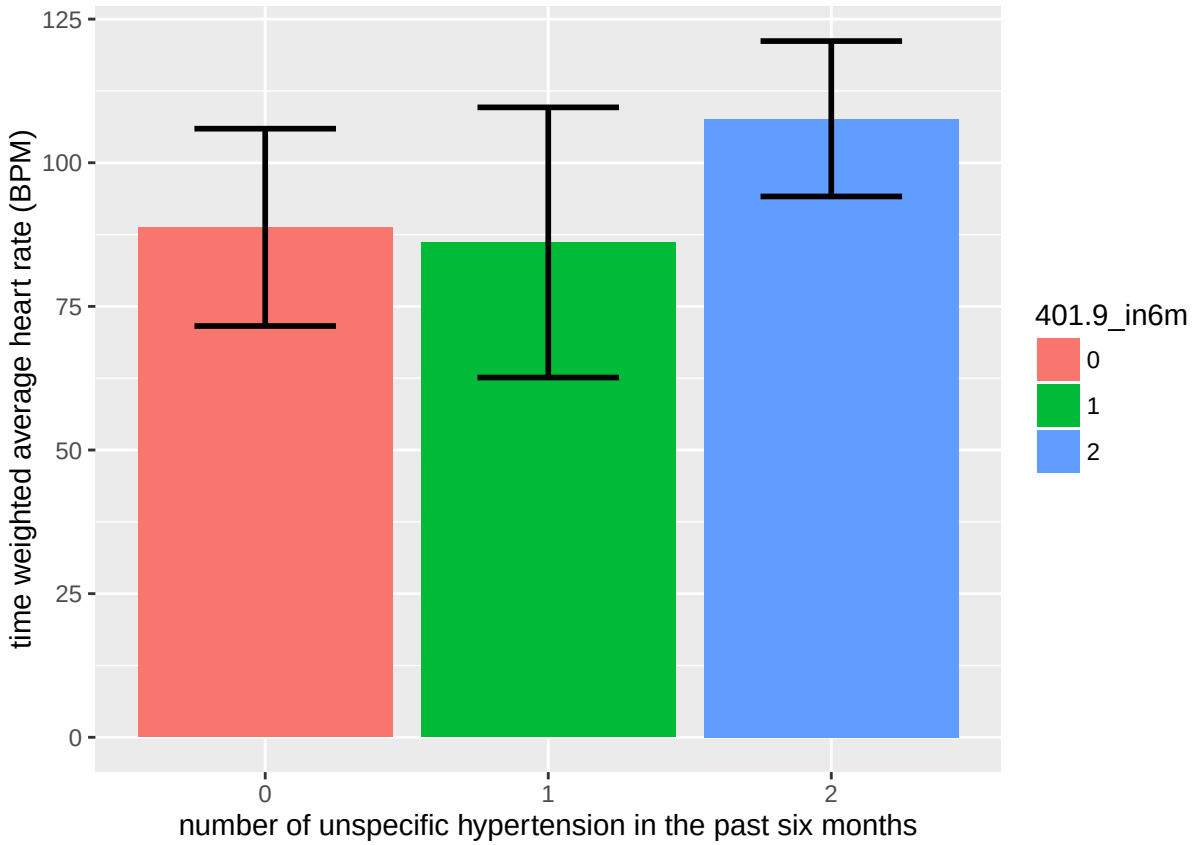
```
UHD <-
  patient_feature_matrix %>%
  select(subject_id, `401.9_in6m`,time_wt_avg)
```

```
ggplot(data=UHD) +
  geom_point(mapping=aes(x=`401.9_in6m`, y=time_wt_avg))
```



An inspection of scatter plot shows that there are three possible numbers for unspecified hypertension diagnosis, 0, 1 and 2. There are a lot more patients have 0 dianosis and only 2 patients have 2 diagnoses. Next, plot the average and standard deviation of heart rate for each group.

```
UHD$`401.9_in6m` <- as.factor(UHD$`401.9_in6m`)
UHD_summary <-
  UHD %>%
  group_by(`401.9_in6m`) %>%
  summarise(avg=mean(time_wt_avg), sd=sd(time_wt_avg))
UHD_summary$`401.9_in6m` <- as.factor(UHD_summary$`401.9_in6m`)
ggplot(data=UHD_summary) +
  geom_bar(mapping=aes(x=`401.9_in6m`, y=avg, fill=`401.9_in6m`),stat='identity') +
  geom_errorbar(aes(x=`401.9_in6m`,y=avg,ymax=avg+sd,ymin=avg-sd), size=1, width=0.5) +
  labs(x='number of unspecific hypertension in the past six months',
       y='time weighted average heart rate (BPM)')
```



This bar graph indicates that patients having 2 unspecified hypertension diagnoses have higher average heart rate than those having only 1 diagnosis or no diagnosis. However, we should keep in mind that there are only two data point for '2 diagnoses' group. Therefore, the significance of this correlation needs to be validated by collecting more data and doing a proper statistic analysis.

Done!

That's it! We've gone through the major steps of transforming different kinds of data stored in a longitudinal database into a patient-feature matrix that we can use for association tests and prediction tasks. Along the way we hope you have gained practice in how to effectively use the `dplyr` and `tidyr` packages to manipulate data and the `ggplot2` package to make visual diagnostics. You are well on your way to being able to perform a clinical informatics study.