Homework 1: Cohort Building [80 points]

LinyuLi

In this assignment you will gain experience extracting and transforming clinical data into datasets for downstream 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 a cohort of patients admitted to the Intensive Care Unit of the Beth Israel Deaconess Medical Center in Boston, Massachusetts. You will analyze this cohort to identify patients that undergo septic shock during their admission. We will be following along with the cohort building process presented in "A targeted real-time early warning score (TREWScore) for septic shock" by Henry et al. published in Science Translational Medicine in 2015. We will also be referring to a cited paper by Angus et al. All of the data you need for this assignment is available on Box

Please edit this document directly using Jupyter or Rmarkdown and answer each of the questions in-line. Jupyter is a useful tool for reproducible research and it is worth taking the short amount of time necessary to learn it. Turn in a single .pdf document (alongwith the completed Rmarkdown or Jupyter notebook) 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

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

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

```
library(lubridate)
```

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

## The following object is masked from 'package:base':
##
##     date
```

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

```
##
## 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(dplyr)
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
Sys.setenv(TZ='UTC')
library(tinytex)
```

1. Building a Cohort Based on Inclusion Criteria and Defining Endpoints

Loading Data

1.1 (5 pts)

The first part of any patient-level study is to identify a cohort of patients who are relevant to the study, and at what point during their records they became eligible. Typically, this is done with a set of "inclusion critera", which, if met, qualify the patient for inclusion in the cohort.

In our study, we will consider the inclusion criteria used in the TREWScore paper.

Read the first paragraph of the *Materials and Methods - Study Design* in the TREWScore paper. What criteria did the authors use to determine which patients should enter the study?

Ans: The authors identified 16,234 distinct patients age 15 years or greater at ICU admission with at least one assessment each of GCS, BUN, hematocrit, and heart rate recorded in the EHR.

1.2 (5 pts)

Once you have found the inclusion criteria, take a look at the MIMIC documentation and report which table(s) in MIMIC you would query in order to identify patients that meet the inclusion criteria. If you're stuck, try looking through the mimic-code repository provided by the MIMIC maintainers to get some ideas as to how to query the database.

Ans: patients,labevents, d_items, chartevents

1.3 (5 pts)

It can be tricky to develop the SQL queries necessary to extract the cohort of interest. Fortunately, the course staff ran the necessary query on the MIMIC III database to extract the identifiers of patients that meet the inclusion criteria discussed above.

Read the vitals and labs data for our cohort stored in *vitals_cohort_sirs.csv* and *labs_cohort* into R dataframes.

Since these CSV files are moderately sized, we suggest using a function from the readr or data.table packages to load the data.

Once you have loaded the files into R dataframes, call `head` and `str` on each dataframe to get a feel for what is contained in each column.

```
"./data/labs_cohort.csv" %>%
  read_csv()-> labs_cohort
```

```
## Parsed with column specification:
## cols(
##   subject_id = col_double(),
##   hadm_id = col_double(),
##   icustay_id = col_double(),
##   charttime = col_datetime(format = ""),
##   lab_id = col_character(),
##   valuenum = col_double()
## )
```

```
"./data/vitals_cohort_sirs.csv" %>%
  read_csv()-> vitals_cohort_sirs
```

```
## Parsed with 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()
## )
```

```
  head(labs_cohort)
```

```
## # A tibble: 6 x 6
##   subject_id hadm_id icustay_id charttime           lab_id      valuenum
```

```
##        <dbl>   <dbl>      <dbl> <dttm>              <chr>          <dbl>
## 1          3  145834     211552 2101-10-20 16:40:00 ANION GAP       17
## 2          3  145834     211552 2101-10-20 16:40:00 BANDS            2
## 3          3  145834     211552 2101-10-20 16:40:00 BICARBONATE     25
## 4          3  145834     211552 2101-10-20 16:40:00 BUN             53
## 5          3  145834     211552 2101-10-20 16:40:00 CHLORIDE        99
## 6          3  145834     211552 2101-10-20 16:40:00 CREATININE       3.2
```

```
  str(labs_cohort)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 9947103 obs. of  6 variables:
##  $ subject_id: num  3 3 3 3 3 3 3 3 3 3 ...
##  $ hadm_id   : num  145834 145834 145834 145834 145834 ...
##  $ icustay_id: num  211552 211552 211552 211552 211552 ...
##  $ charttime : POSIXct, format: "2101-10-20 16:40:00" "2101-10-20 16:40:00" ...
##  $ lab_id    : chr  "ANION GAP" "BANDS" "BICARBONATE" "BUN" ...
##  $ valuenum  : num  17 2 25 53 99 3.2 91 30.2 10 1.3 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   subject_id = col_double(),
##   ..   hadm_id = col_double(),
##   ..   icustay_id = col_double(),
##   ..   charttime = col_datetime(format = ""),
##   ..   lab_id = col_character(),
##   ..   valuenum = col_double()
##   .. )
```

```
  head(vitals_cohort_sirs)
```

```
## # 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 19:15:00      217 SysBP
## 2          3  145834     211552 2101-10-20 19:30:00      151 HeartRate
## 3          3  145834     211552 2101-10-20 19:30:00      102 SysBP
## 4          3  145834     211552 2101-10-20 19:45:00      135 HeartRate
## 5          3  145834     211552 2101-10-20 19:45:00       94 SysBP
## 6          3  145834     211552 2101-10-20 20:00:00      143 HeartRate
```

```
  str(vitals_cohort_sirs)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 20633223 obs. of  6 variables:
##  $ subject_id: num  3 3 3 3 3 3 3 3 3 3 ...
##  $ hadm_id   : num  145834 145834 145834 145834 145834 ...
##  $ icustay_id: num  211552 211552 211552 211552 211552 ...
##  $ charttime : POSIXct, format: "2101-10-20 19:15:00" "2101-10-20 19:30:00" ...
##  $ valuenum  : num  217 151 102 135 94 143 24 78 80 165 ...
##  $ vital_id  : chr  "SysBP" "HeartRate" "SysBP" "HeartRate" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   subject_id = col_double(),
##   ..   hadm_id = col_double(),
```

```
##   ..   icustay_id = col_double(),
##   ..   charttime = col_datetime(format = ""),
##   ..   valuenum = col_double(),
##   ..   vital_id = col_character()
##   .. )
```

```
  "./data/diagnoses.csv" %>%
read_csv()-> diagnoses
```

```
## Parsed with column specification:
## cols(
##   row_id = col_double(),
##   subject_id = col_double(),
##   hadm_id = col_double(),
##   seq_num = col_double(),
##   icd9_code = col_character(),
##   mimic_id = col_double()
## )
```

1.4 (10 pts)

The Systemic Inflammatory Response Syndrome (SIRS) criteria has been an integral tool for the clinical definition of sepsis for the past several decades. In the TREWScore paper, the authors considered a patient to have sepsis if at least two of the four SIRS criteria were simultaneously met during an admission where a suspicion of infection was also present.

The four SIRS criteria are as follows: 1. Temperature $> 38°C$ or $< 36°C$ 2. Heart Rate > 90 3. Respiratory Rate > 20 or $PaCO_2 < 32mmHg$ 4. WBC $> 12{,}000/mm^3$, $< 4000/mm^3$, or $> 10\%$ bands

You may read more about SIRS (and some recent associated controversies surrounding its use) at https://www.ncbi.nlm.nih.gov/pubmed/1303622 and http://www.nejm.org/doi/full/10.1056/NEJMoa1415236#t=article.

The next step in our process will be to assess whether patients satisfy each of the SIRS criteria at each time step that vitals or lab data is available. To this end, we would like to have a dataframe where each row corresponds to a unique combination of *subject_id*, *hadm_id*, *icustay_id*, and *charttime*, and with one column for each unique type of lab or vital that was measured at that time. This may seem complicated at first, so let's walk through it step-by-step to build some intuition.

First, notice that some patients have multiple measurements of a given vital or lab that were taken at the same time. There may also have been multiple types of raw measurement that were mapped to the same measurement label.

You can count the number of measurements of a given lab or vital taken at a given time for every patient using the `group_by` and `summarise` commands from `dplyr`:

There are a few ways to summarize the data for a patient at a given time point into a single value. One way is to pick a random measurement from the set taken at a given time, another is to summarize the measurements by some operation e.g. taking the mean.

Implement a solution to summarize the values by calculating the mean value for each patient at a given time point. One approach could use the `group_by`, `summarise` and `ungroup` commands from `dplyr`.

After you have done this, pivot on *lab_id* and *vital_id* again and use `str` to inspect the resulting dataframes. In other words, create a new column for each unique measurement type in your labs and vitals data frames where the rows are given by unique combinations of *subject_id*, *hadm_id*, *icustay_id*, *charttime*, *vital_id*. If a complete set of measurements is not available in a particular row of the resulting dataframe, fill in the value with an `NA`. A solution could use `spread` from `dplyr` or `dcast` from `data.table`.

```
labs_cohort <- labs_cohort %>%
    group_by(subject_id, hadm_id, icustay_id, charttime, lab_id) %>%
    summarise(valuenum=mean(valuenum))%>%
    ungroup()

pivot_labs_cohort <-labs_cohort %>%
    spread(lab_id,valuenum)

  head(pivot_labs_cohort)
```

```
## # A tibble: 6 x 24
##   subject_id hadm_id icustay_id charttime           ALBUMIN `ANION GAP`
##        <dbl>   <dbl>      <dbl> <dttm>                <dbl>       <dbl>
## 1          3  145834     211552 2101-10-20 16:40:00    NA            17
## 2          3  145834     211552 2101-10-20 16:49:00    NA            NA
## 3          3  145834     211552 2101-10-20 19:12:00    NA            NA
## 4          3  145834     211552 2101-10-20 19:14:00    NA            NA
## 5          3  145834     211552 2101-10-20 19:26:00    NA            23
## 6          3  145834     211552 2101-10-20 19:59:00     1.8          22
## # ... with 18 more variables: BANDS <dbl>, BICARBONATE <dbl>,
## #   BILIRUBIN <dbl>, BUN <dbl>, CHLORIDE <dbl>, CREATININE <dbl>,
## #   GLUCOSE <dbl>, HEMATOCRIT <dbl>, HEMOGLOBIN <dbl>, INR <dbl>,
## #   LACTATE <dbl>, PaCO2 <dbl>, PLATELET <dbl>, POTASSIUM <dbl>, PT <dbl>,
## #   PTT <dbl>, SODIUM <dbl>, WBC <dbl>
```

```
vitals_cohort_sirs <- vitals_cohort_sirs %>%
    group_by(subject_id, hadm_id, icustay_id, charttime, vital_id) %>%
    summarise(valuenum=mean(valuenum))%>%
    ungroup()

pivot_vitals_cohort_sirs <-vitals_cohort_sirs %>%
    spread(vital_id,valuenum)

  head(pivot_vitals_cohort_sirs)
```

```
## # A tibble: 6 x 8
##   subject_id hadm_id icustay_id charttime           HeartRate RespRate
##        <dbl>   <dbl>      <dbl> <dttm>                  <dbl>    <dbl>
## 1          3  145834     211552 2101-10-20 19:15:00        NA       NA
## 2          3  145834     211552 2101-10-20 19:30:00       151       NA
## 3          3  145834     211552 2101-10-20 19:45:00       135       NA
## 4          3  145834     211552 2101-10-20 20:00:00       143       24
## 5          3  145834     211552 2101-10-20 20:15:00       165        5
## 6          3  145834     211552 2101-10-20 20:30:00       168       21
## # ... with 2 more variables: SysBP <dbl>, TempC <dbl>
```

1.5 (5 pts)

Since the measurement times for the vital signs may be different from those of the labs, the next step is to merge the vitals and labs dataframes together to get the full timeline for each patient.

Using a command such as `full_join` or `merge`, merge the pivoted labs and vitals dataframes you generated previously, using the common columns in the two dataframes. There should be one row for each unique combination of *subject_id*, *hadm_id*, *icustay_id*, *charttime*, *vital_id* observed in either dataframe.

```r
m0 <- merge(pivot_labs_cohort,pivot_vitals_cohort_sirs,all=TRUE)
head(m0)
```

```
##   subject_id hadm_id icustay_id           charttime ALBUMIN ANION GAP
## 1          3  145834     211552 2101-10-20 16:40:00      NA        17
## 2          3  145834     211552 2101-10-20 16:49:00      NA        NA
## 3          3  145834     211552 2101-10-20 19:12:00      NA        NA
## 4          3  145834     211552 2101-10-20 19:14:00      NA        NA
## 5          3  145834     211552 2101-10-20 19:15:00      NA        NA
## 6          3  145834     211552 2101-10-20 19:26:00      NA        23
##   BANDS BICARBONATE BILIRUBIN BUN CHLORIDE CREATININE GLUCOSE HEMATOCRIT
## 1     2          25        NA  53       99        3.2      91       30.2
## 2    NA          NA        NA  NA       NA         NA      NA       31.0
## 3    NA          NA        NA  NA       NA         NA      NA         NA
## 4    NA          NA        NA  NA      114         NA     140         NA
## 5    NA          NA        NA  NA       NA         NA      NA         NA
## 6    NA          13        NA  41      111        2.4     162       24.9
##   HEMOGLOBIN INR LACTATE PaCO2 PLATELET POTASSIUM   PT  PTT SODIUM  WBC
## 1       10.0 1.3      NA    NA      282       5.4 13.5 30.7    136 12.7
## 2       10.3  NA      NA    NA       NA        NA   NA   NA     NA   NA
## 3         NA  NA     4.3    40       NA       4.4   NA   NA    138   NA
## 4         NA  NA     8.8    28       NA       4.0   NA   NA    153   NA
## 5         NA  NA      NA    NA       NA        NA   NA   NA     NA   NA
## 6        7.8 1.7      NA    NA      190       4.0 15.7 58.3    143 11.3
##   HeartRate RespRate SysBP TempC
## 1        NA       NA    NA    NA
## 2        NA       NA    NA    NA
## 3        NA       NA    NA    NA
## 4        NA       NA    NA    NA
## 5        NA       NA   217    NA
## 6        NA       NA    NA    NA
```

1.6 (5 pts)

You will notice that the resulting dataframe contains a lot of "missing" values recorded as `NA`. There are many potential approaches for handling missing values that we could take to address this issue. In this case, we are going to use a last-value-carried-forward approach within an ICU stay to fill in missing values.

In a sentence or two, discuss any potential benefits and drawbacks of this approach. After that, implement this strategy using commands of your choice.

```r
m0 <- m0 %>%
 group_by(subject_id, hadm_id, icustay_id) %>%
 fill(ALBUMIN:TempC) %>%
 ungroup()
```

Benefits:This method can fill in items with few missing values very well, and items with only a few missing values can be included in the final discussion, so we can get a more comprehensive discussion.

Drawbacks:Some items have a lot of missing values, these will also be counted later in the discussion, which is not conducive to the accuracy of our calculation later

1.7 (10 pts)

Now we have a record of the most recent value for each lab or vital within an ICU stay for each patient in our development set. From this data, create a new dataframe called *SIRS* that has a record for each row in your timeline dataframe developed previously and a column indicating whether each of the SIRS criteria were satisfied at each chart time, and a final column indicating whether at least 2 of the SIRS criteria were satisfied. Assume that if a value is unknown that the patient does not meet that SIRS criterion.

1. Temperature $> 38°C$ or $< 36°C$
2. Heart Rate > 90
3. Respiratory Rate > 20 or $PaCO_2 < 32$mmHg
4. WBC $> 12{,}000/mm^3$, $< 4000/mm^3$, or $> 10\%$ bands

```
SIRS=m0%>%
  mutate(temperature=(ifelse(m0$TempC>38|m0$TempC<36,1,0)&is.na(m0$TempC)==FALSE))%>%
  mutate(heart_rate=(ifelse(m0$HeartRate>90,1,0)&is.na(m0$HeartRate)==FALSE))%>%
  mutate(res_rate=(ifelse(m0$RespRate>20|m0$PaCO2<32,1,0)&is.na(m0$RespRate)==FALSE)&(is.na(m0$PaCO2)==
  mutate(wbc=(ifelse(m0$WBC>12|m0$WBC<4|m0$BANDS>10,1,0)&is.na(m0$TempC)==0&(is.na(m0$BANDS)==FALSE)))%
  mutate(sirs=ifelse(temperature+heart_rate+res_rate+wbc>=2,"is_SIRS","non_SIRS")) %>%
  select(subject_id,hadm_id,icustay_id,charttime,temperature,heart_rate,res_rate,wbc,sirs)


head(SIRS)
```

```
## # A tibble: 6 x 9
##   subject_id hadm_id icustay_id charttime           temperature heart_rate
##        <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 FALSE       FALSE
## 5          3  145834     211552 2101-10-20 19:15:00 FALSE       FALSE
## 6          3  145834     211552 2101-10-20 19:26:00 FALSE       FALSE
## # ... with 3 more variables: res_rate <lgl>, wbc <lgl>, sirs <chr>
```

1.8 (10 pts)

At this point, we have computed the SIRS criteria for every patient in our cohort. Now it's time to determine which patients had suspicion of infection. In the TREWScore paper, the authors use a set of ICD9 codes to identify infection-related diagnoses.

The course staff has extracted the entirety of the relevant table where ICD9 codes are stored in MIMIC and provided it for you in *diagnoses.csv*.

Additionally, for your convenience, we include the set of reference information from the paper that will be useful in determining which admissions indicate infection. Using this reference information, filter the provided table such that it includes only admissions from the cohort that have at least one string that *starts with* one of the provided ICD9 codes that indicate infection.

We suggest using functions from the **stringr** package.

```
# Provided
infection3digit <- c('001','002','003','004','005','008',
                     '009','010','011','012','013','014','015','016','017','018',
                     '020','021','022','023','024','025','026','027','030','031',
```

```
                     '032','033','034','035','036','037','038','039','040','041',
                     '090','091','092','093','094','095','096','097','098','100',
                     '101','102','103','104','110','111','112','114','115','116',
                     '117','118','320','322','324','325','420','421','451','461',
                     '462','463','464','465','481','482','485','486','494','510',
                     '513','540','541','542','566','567','590','597','601','614',
                     '615','616','681','682','683','686','730'
                    )
infection4digit <- c('5695','5720','5721','5750','5990','7110',
                     '7907','9966','9985','9993'
                    )
infection5digit <- c('49121','56201','56203','56211','56213', '56983')
infection_codes <- c(infection3digit, infection4digit, infection5digit)
```

```
infection3digit <- c('001','002','003','004','005','008',
                     '009','010','011','012','013','014','015','016','017','018',
                     '020','021','022','023','024','025','026','027','030','031',
                     '032','033','034','035','036','037','038','039','040','041',
                     '090','091','092','093','094','095','096','097','098','100',
                     '101','102','103','104','110','111','112','114','115','116',
                     '117','118','320','322','324','325','420','421','451','461',
                     '462','463','464','465','481','482','485','486','494','510',
                     '513','540','541','542','566','567','590','597','601','614',
                     '615','616','681','682','683','686','730'
                     )
infection4digit <- c('5695','5720','5721','5750','5990','7110',
                     '7907','9966','9985','9993'
                     )
infection5digit <- c('49121','56201','56203','56211','56213','56983')

x=diagnoses%>%
  group_by(subject_id)%>%
  mutate(x1=str_sub(icd9_code,1,3) %in% infection3digit)%>%
  mutate(x2=str_sub(icd9_code,1,4) %in% infection4digit)%>%
  mutate(x3=str_sub(icd9_code,1,5) %in% infection5digit)%>%
  mutate(x = if_else(x1 == TRUE|x2 == TRUE|x3 == TRUE, TRUE, FALSE, missing = FALSE)) %>%
select(subject_id,hadm_id,x) %>%
  filter(x == TRUE) %>%
  distinct(subject_id, .keep_all = TRUE)
head(x)
```

```
## # A tibble: 6 x 3
## # Groups:   subject_id [6]
##   subject_id hadm_id x
##        <dbl>   <dbl> <lgl>
## 1          3  145834 TRUE
## 2          4  185777 TRUE
## 3         19  109235 TRUE
## 4         21  109451 TRUE
## 5         31  128652 TRUE
## 6         33  176176 TRUE
```

1.9 (5 pts)

In the paper, the authors also consider a patient to have infection during an admission if there is at least one mention of 'sepsis' or 'septic' in a clinical note for the admission. The course staff has done the work of extracting the clinical notes for the cohort.

Load the notes data from *notes.csv* into a dataframe. Once you have done so, apply the string matching techniques you developed above to identify admissions that mention the terms 'sepsis' or 'septic'.

```
"./data/notes.csv" %>%
read_csv()-> notes
```

```
## Parsed with column specification:
## cols(
##   row_id = col_double(),
##   subject_id = col_double(),
##   hadm_id = col_double(),
##   chartdate = col_datetime(format = ""),
##   charttime = col_datetime(format = ""),
##   storetime = col_datetime(format = ""),
##   category = col_character(),
##   description = col_character(),
##   cgid = col_double(),
##   iserror = col_logical(),
##   text = col_character(),
##   mimic_id = col_double()
## )
```

```
y=notes%>%
  mutate(y1=str_detect(text, 'sepsis'))%>%
  mutate(y2=str_detect(text, 'septic'))%>%
  mutate(y = if_else(y1==TRUE|y2==TRUE, TRUE,FALSE, missing = FALSE)) %>%
  select(subject_id,hadm_id,y) %>%
  filter(y == TRUE) %>%
  distinct(subject_id, .keep_all = TRUE)



  # str_detect(string = notes$text, pattern = "septic")
  # str_detect(string = notes$text, pattern = "sepsis")
```

1.10 (5 pts)

At this stage, we now have all the information we need to determine the times that patients meet the criteria for sepsis. Join the results from the search for patients with infection codes and sepsis notes with your SIRS data frame and label the chart times that meet the TREWScore paper's definition of sepsis.

```
"./data/notes.csv" %>%
read_csv()-> notes
```

```
## Parsed with column specification:
## cols(
##   row_id = col_double(),
##   subject_id = col_double(),
##   hadm_id = col_double(),
```

```
##   chartdate = col_datetime(format = ""),
##   charttime = col_datetime(format = ""),
##   storetime = col_datetime(format = ""),
##   category = col_character(),
##   description = col_character(),
##   cgid = col_double(),
##   iserror = col_logical(),
##   text = col_character(),
##   mimic_id = col_double()
## )
```

```
#xy <- merge(x,y,all=TRUE)
# str(xy)


xy_SIRS <- SIRS %>% merge(x,all=TRUE) %>% merge(y,all=TRUE)
# str(xy_SIRS)

sepsis=xy_SIRS%>%
  mutate(sepsis= sirs=="is_SIRS" & (x == TRUE | y == TRUE))

head(sepsis)
```

```
##   subject_id hadm_id icustay_id           charttime temperature heart_rate
## 1          3  145834     211552 2101-10-23 15:00:00       FALSE      FALSE
## 2          3  145834     211552 2101-10-24 18:00:00       FALSE      FALSE
## 3          3  145834     211552 2101-10-24 04:00:00       FALSE      FALSE
## 4          3  145834     211552 2101-10-22 20:00:00       FALSE      FALSE
## 5          3  145834     211552 2101-10-23 02:00:00       FALSE      FALSE
## 6          3  145834     211552 2101-10-24 09:00:00       FALSE      FALSE
##   res_rate   wbc     sirs    x    y sepsis
## 1     TRUE  TRUE  is_SIRS TRUE TRUE   TRUE
## 2    FALSE FALSE non_SIRS TRUE TRUE  FALSE
## 3     TRUE FALSE non_SIRS TRUE TRUE  FALSE
## 4     TRUE  TRUE  is_SIRS TRUE TRUE   TRUE
## 5     TRUE  TRUE  is_SIRS TRUE TRUE   TRUE
## 6    FALSE FALSE non_SIRS TRUE TRUE  FALSE
```

1.11 (10 pts)

In the TREWScore paper, the authors also identify patients with **severe sepsis** and **septic shock**. Severe sepsis is defined as sepsis with **organ dysfunction**. Septic shock is defined as **severe sepsis**, **hypotension**, and **adequate fluid resuscitation** occurring at the same time. In order to determine which patients met the criteria for *severe sepsis* and *septic shock* according to the TREWScore paper, we will first need to define the concepts of **organ dysfunction**, **adequate fluid resuscitation**, and **hypotension**.

Unfortunately, the criteria the authors use to define organ dysfunction is rather cumbersome. Instead of implementing that criteria explicitly, we adopt a simpler approach. In the Angus 2001 paper (linked at the top of the assignment), the authors did just that by defining a set of ICD9 codes as a proxy for sepsis-related organ dysfunction. As before, we provide the list of relevant parent ICD9 codes. Using those codes *prefixes*, identify the admissions where an ICD9 codes that starts with one of those prefixes was assigned to determine those admissions that meet the crtieria for organ dysfunction.

Once you have identified the admissions where patients suffered from organ failure, derive labels for **severe sepsis** for the set of chart times that you have previously labeled for sepsis.

```
code_prefixes <- c('458','293','570','584', '7855','3483','3481', '2874','2875','2869','2866','5734')
```

```
code_prefixes <- c('458','293','570','584', '7855','3483','3481', '2874','2875','2869','2866','5734')
code_prefixes <- paste("^",code_prefixes,sep="",collapse="|")

organ_dysfunction=diagnoses %>%
  mutate(suffered=str_detect(icd9_code, code_prefixes)) %>%
  select(subject_id,hadm_id,suffered) %>%
  filter(suffered == TRUE) %>%
  distinct(subject_id, .keep_all = TRUE)
head(organ_dysfunction)
```

```
## # A tibble: 6 x 3
##   subject_id hadm_id suffered
##        <dbl>   <dbl> <lgl>
## 1          3  145834 TRUE
## 2          9  150750 TRUE
## 3         17  161087 TRUE
## 4         21  109451 TRUE
## 5         35  166707 TRUE
## 6         38  185910 TRUE
```

```
#severe_sepsis=sepsis
#head(severe_sepsis)

organ_severe <- merge(sepsis,organ_dysfunction,all=TRUE)
head(organ_severe)
```

```
##   subject_id hadm_id icustay_id           charttime temperature heart_rate
## 1          3  145834     211552 2101-10-20 20:00:00       FALSE       TRUE
## 2          3  145834     211552 2101-10-22 04:31:00       FALSE      FALSE
## 3          3  145834     211552 2101-10-21 10:15:00       FALSE       TRUE
## 4          3  145834     211552 2101-10-25 23:00:00       FALSE       TRUE
## 5          3  145834     211552 2101-10-24 11:12:00       FALSE      FALSE
## 6          3  145834     211552 2101-10-22 13:02:00       FALSE      FALSE
##   res_rate   wbc     sirs    x    y sepsis suffered
## 1     TRUE FALSE  is_SIRS TRUE TRUE   TRUE     TRUE
## 2     TRUE  TRUE  is_SIRS TRUE TRUE   TRUE     TRUE
## 3    FALSE  TRUE  is_SIRS TRUE TRUE   TRUE     TRUE
## 4     TRUE FALSE  is_SIRS TRUE TRUE   TRUE     TRUE
## 5    FALSE FALSE non_SIRS TRUE TRUE  FALSE     TRUE
## 6     TRUE  TRUE  is_SIRS TRUE TRUE   TRUE     TRUE
```

```
organ_severe = organ_severe %>%
 mutate(severe = hadm_id %in% organ_dysfunction$hadm_id & sepsis=="TRUE")

head(organ_severe)
```

```
##   subject_id hadm_id icustay_id           charttime temperature heart_rate
## 1          3  145834     211552 2101-10-20 20:00:00       FALSE       TRUE
## 2          3  145834     211552 2101-10-22 04:31:00       FALSE      FALSE
```

```
## 3          3  145834     211552 2101-10-21 10:15:00       FALSE       TRUE
## 4          3  145834     211552 2101-10-25 23:00:00       FALSE       TRUE
## 5          3  145834     211552 2101-10-24 11:12:00       FALSE      FALSE
## 6          3  145834     211552 2101-10-22 13:02:00       FALSE      FALSE
##   res_rate   wbc     sirs    x    y sepsis suffered severe
## 1     TRUE FALSE  is_SIRS TRUE TRUE   TRUE     TRUE   TRUE
## 2     TRUE  TRUE  is_SIRS TRUE TRUE   TRUE     TRUE   TRUE
## 3    FALSE  TRUE  is_SIRS TRUE TRUE   TRUE     TRUE   TRUE
## 4     TRUE FALSE  is_SIRS TRUE TRUE   TRUE     TRUE   TRUE
## 5    FALSE FALSE non_SIRS TRUE TRUE  FALSE     TRUE  FALSE
## 6     TRUE  TRUE  is_SIRS TRUE TRUE   TRUE     TRUE   TRUE
```

1.12 Tying it all together (5 pts)

The course staff have created a dataset with a variable indicating whether a patient had adequate fluid resuscitation or was hypotensive at each timepoint in their record. These data are stored in **fluids_all.csv**, **hypotension_labels.csv**. Use this data in combination with your dataframe that indicates whether a patient has severe sepsis at a given time to determine whether a patient has adequate fluid resuscitation and is hypotensive at each timepoint in their record. You may handle "missing" data in any way you deem reasonable.

Use this data in combination with your dataframe that indicates whether a patient has severe sepsis at a given time to determine whether a patient is in septic shock at each timepoint in their record. Create a timeline that merges the labels you derived for each sepsis grade such that for each unique observation time for each patient, you have a binary label for each of *sepsis*, *severe sepsis*, and *septic shock*.

In the next assignment, we will use this cohort to derive labels as targets of a predictive model.

Congratulations! You've extracted a patient cohort from MIMIC and derived multiple sepsis-related endpoints. You're done!

```
"./data/fluids_all.csv" %>%
read_csv()-> fluids
```

```
## Parsed with column specification:
## cols(
##   subject_id = col_double(),
##   hadm_id = col_double(),
##   icustay_id = col_double(),
##   charttime = col_datetime(format = ""),
##   amount_24h = col_double(),
##   current_amount = col_double(),
##   relative_amount = col_double(),
##   adequate_fluid = col_logical()
## )
```

```
"./data/hypotension_labels.csv" %>%
read_csv()-> hypotension
```

```
## Parsed with column specification:
## cols(
##   subject_id = col_double(),
##   hadm_id = col_double(),
##   icustay_id = col_double(),
```

```
##   charttime = col_datetime(format = ""),
##   hypotension = col_logical()
## )
```

```
head(fluids)
```

```
## # A tibble: 6 x 8
##   subject_id hadm_id icustay_id charttime           amount_24h
##        <dbl>   <dbl>      <dbl> <dttm>                  <dbl>
## 1          3  145834     211552 2101-10-21 00:00:00     10000
## 2          3  145834     211552 2101-10-21 02:00:00     10375
## 3          3  145834     211552 2101-10-21 03:00:00     10625
## 4          3  145834     211552 2101-10-21 04:00:00     10979
## 5          3  145834     211552 2101-10-21 05:00:00     11364.
## 6          3  145834     211552 2101-10-21 07:00:00     11504.
## # ... with 3 more variables: current_amount <dbl>, relative_amount <dbl>,
## #   adequate_fluid <lgl>
```

```
head(hypotension)
```

```
## # A tibble: 6 x 5
##   subject_id hadm_id icustay_id charttime           hypotension
##        <dbl>   <dbl>      <dbl> <dttm>              <lgl>
## 1          3  145834     211552 2101-10-20 16:40:00 FALSE
## 2          3  145834     211552 2101-10-20 16:49:00 FALSE
## 3          3  145834     211552 2101-10-20 19:12:00 FALSE
## 4          3  145834     211552 2101-10-20 19:14:00 FALSE
## 5          3  145834     211552 2101-10-20 19:15:00 FALSE
## 6          3  145834     211552 2101-10-20 19:26:00 FALSE
```

```
m1<- merge(organ_severe, fluids,all=TRUE)
m2<- merge(m1,hypotension,all=TRUE)

 septic_shock <- m2 %>%
 group_by(subject_id, hadm_id, icustay_id) %>%
  fill(severe,adequate_fluid,hypotension) %>%
  ungroup()

head(septic_shock)
```

```
## # A tibble: 6 x 19
##   subject_id hadm_id icustay_id charttime           temperature heart_rate
##        <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 FALSE       FALSE
## 5          3  145834     211552 2101-10-20 19:15:00 FALSE       FALSE
## 6          3  145834     211552 2101-10-20 19:26:00 FALSE       FALSE
## # ... with 13 more variables: res_rate <lgl>, wbc <lgl>, sirs <chr>,
## #   x <lgl>, y <lgl>, sepsis <lgl>, suffered <lgl>, severe <lgl>,
## #   amount_24h <dbl>, current_amount <dbl>, relative_amount <dbl>,
## #   adequate_fluid <lgl>, hypotension <lgl>
```

```r
septic_shock <- septic_shock %>%
  mutate(septic_shock=severe & adequate_fluid & hypotension)
head(septic_shock)
```

```
## # A tibble: 6 x 20
##   subject_id hadm_id icustay_id charttime           temperature heart_rate
##        <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 FALSE       FALSE
## 5          3  145834     211552 2101-10-20 19:15:00 FALSE       FALSE
## 6          3  145834     211552 2101-10-20 19:26:00 FALSE       FALSE
## # ... with 14 more variables: res_rate <lgl>, wbc <lgl>, sirs <chr>,
## #   x <lgl>, y <lgl>, sepsis <lgl>, suffered <lgl>, severe <lgl>,
## #   amount_24h <dbl>, current_amount <dbl>, relative_amount <dbl>,
## #   adequate_fluid <lgl>, hypotension <lgl>, septic_shock <lgl>
```