Missing Data

H. David Shea

21 Jul 2021

Contents

Source MIMIC-III demo version data **1**

Extract base data 1

Exercise: Selection of variables 2

Exercise: Handling missing data in dataframes 3

Exercise: Plotting missing data 6

Exercise: Missing data imputation 11

(Note: Updated and modified from the hst953-edx github version.)

Source MIMIC-III demo version data

In the original version (on the hst953-edx github site), they used the MIMIC-III demo version directly loaded. Now, I have in *mimic_base_dir*/database/mimic3.db the SQLite version of the full MIMIC-III v1.4 database loaded. I'll use that in the processing below - with some pre-coded inclusion criteria to extract just the demo data. The following code chunk attaches the database and loads auxiliary functions (available in *mimic_base_dir*/mimic_concepts) for extracting database data (`db_functions.R`) and for doing some MIMIC data interpretation and pre-processing (`mimic3_meta_data.R`) - including the processing to get just the demo data.

```
base_dir <- here::here("")
mimic_base_dir <- fs::path(base_dir,"../MIMIC-research")
db_file <- fs::path(mimic_base_dir, "database/mimic3.db")
if(dbCanConnect(RSQLite::SQLite(), db_file)) {
    mimic3 <- dbConnect(RSQLite::SQLite(), db_file)
} else {
    stop(str_c("Database file: ", db_file, " not found.", sep=""))
}
source(fs::path(mimic_base_dir, "mimic_concepts/db_functions.R"))
source(fs::path(mimic_base_dir, "mimic_concepts/mimic3_meta_data.R"))
```

Extract base data

These are the tibbles used in the following exercises:

```r
adm <- db_get_admissions(mimic3, where = demo_subject_ids)

icu <- db_get_icustays(mimic3, where = demo_subject_ids)

icu_adm <- adm %>%
  left_join(icu, by = c("SUBJECT_ID", "HADM_ID"))

vitals <- db_get_chartevents(mimic3, where = demo_subject_ids)
```

Exercise: Selection of variables

Aim: Select variables to analyze missing values.

Vital Signs

D_ITEMS is sourced from two distinct ICU databases. The main consequence is that there are duplicate ITEMID for each concept. For example, heart rate is captured both as an ITEMID of 211 (CareVue) and as an ITEMID of 220045 (Metavision). As a result, it is necessary to search for multiple ITEMID to capture a single concept across the entire database.

For more information read: https://mimic.mit.edu/docs/iii/tables/d_items/

In GitHub you may find code regarding vital signs and respective items IDs:

https://github.com/MIT-LCP/mimic-code/blob/main/mimic-iii/concepts/firstday/vitals_first_day.sql

We will use data from chartevents, namely heart rate (hr) and pulse oximetry (SpO2) measurements. Attention to the units. Since much of the information is manually typed in the system, human error can always be present.

Heart rate data:

```sql
SELECT ITEMID, LABEL, ABBREVIATION, DBSOURCE, LINKSTO, CATEGORY, UNITNAME, PARAM_TYPE, CONCEPTID
FROM   D_ITEMS
WHERE  ITEMID in (211, 220045);
```

Table 1: 2 records

ITEMID	LABEL	ABBREVIATION	DBSOURCE	LINKSTO	CATEGORY	UNITNAME	PARAM_TYPE	CONCEPTID
211	Heart Rate		carevue	chartevents	NA			NA
220045	Heart Rate	HR	metavision	chartevents	Routine Vital Signs	bpm	Numeric	NA

Pulse oximetry data:

```sql
SELECT ITEMID, LABEL, ABBREVIATION, DBSOURCE, LINKSTO, CATEGORY, UNITNAME, PARAM_TYPE, CONCEPTID
FROM   D_ITEMS
WHERE  ITEMID in (646, 220277);
```

Table 2: 2 records

ITEMID	LABEL	ABBREVIATION	DBSOURCE	LINKSTO	CATEGORY	UNITNAME	PARAM_TYPE	CONCEPTID
646	SpO2		carevue	chartevents	NA			NA
220277	O2 saturation pulseoxymetry	SpO2	metavision	chartevents	Respiratory	%	Numeric	NA

Review the amount of missing values in both vitals.

```
hr <- vitals %>%
  filter(((ITEMID == 211) | (ITEMID == 220045)) & (toupper(VALUEUOM) == "BPM"))

summary(hr$VALUENUM)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    0.00   75.00   87.00   87.77  100.00  189.00       5
length(hr$VALUENUM)
#> [1] 15490
na_hr <- hr %>% filter(is.na(VALUENUM)) %>% count() %>% pull(n)

sp <- vitals %>%
  filter(((ITEMID == 646) | (ITEMID == 220277)) & (VALUEUOM == "%"))

summary(sp$VALUENUM)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    0.00   96.00   98.00   97.01  100.00  100.00      15
length(sp$VALUENUM)
#> [1] 15315
na_sp <- sp %>% filter(is.na(VALUENUM)) %>% count() %>% pull(n)
```

We observe there are 5 missing values ("NA") for heart rate and 15 "NA" for SpO2. Heart rate has a slightly higher frequency of measurement, (however this is a demo version of the database).

Exercise: Handling missing data in dataframes

Two of the simplest methods to handle missing data are presented below. Recoding a missing value consists of assigning a value to an already existing value (eg. outlier) which we want to recode as missing. Excluding missing values can be performed by excluding objects (patients, rows in the dataframe) or variables (columns in the dataframe) with significant amount of missing data (see this chapter contents for more detailed theory).

Outliers

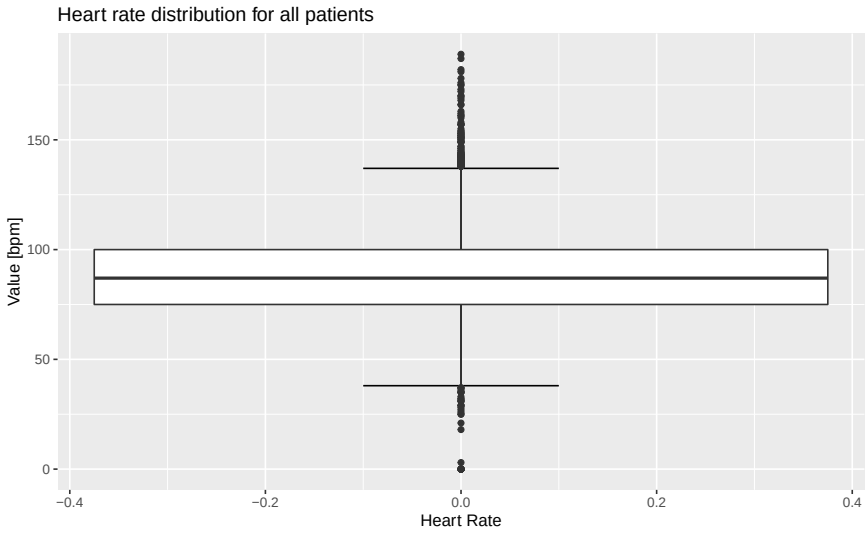
Before handling missing data, we should analyze the presence of outliers (refer to the respective chapter in the course). If we apply an imputation method before processing outliers, our imputation will be based on incorrect data and therefore not valid.

Looking at the distribution of heart rate for all patients, we can assess potential outliers through a box plot.

Observe the distribution of heart rate for all the patients in the dataset.

```
hr %>%
  ggplot(aes(y = VALUENUM)) +
  stat_boxplot(geom = "errorbar", width = 0.2, na.rm = TRUE) +
```

```
  geom_boxplot(na.rm = TRUE) +
  labs(title = "Heart rate distribution for all patients",
       x = "Heart Rate",
       y = 'Value [bpm]')
```




```
uw <- quantile(hr$VALUENUM, 3/4, na.rm = TRUE) + (1.5 * IQR(hr$VALUENUM, na.rm = TRUE))
lw <- quantile(hr$VALUENUM, 1/4, na.rm = TRUE) - (1.5 * IQR(hr$VALUENUM, na.rm = TRUE))

above <- sum(hr$VALUENUM > uw, na.rm = TRUE)
below <- sum(hr$VALUENUM < lw, na.rm = TRUE)
```

We observe there are a few outliers. 100 observations lie above the upper whisker (137.5) and 39 observations lie below the lower whisker (37.5).

Missing data recoding

Aim: Recode a value as missing.

All of the heart rate values that we see in these data are within a reasonable range. If there were extreme outlier values (greater than, say, 300, or less than zero), we would want to correct these. We can recode all values outside the physiological ranges [0, 300] bpm as "NA" in the dataset for all patients.

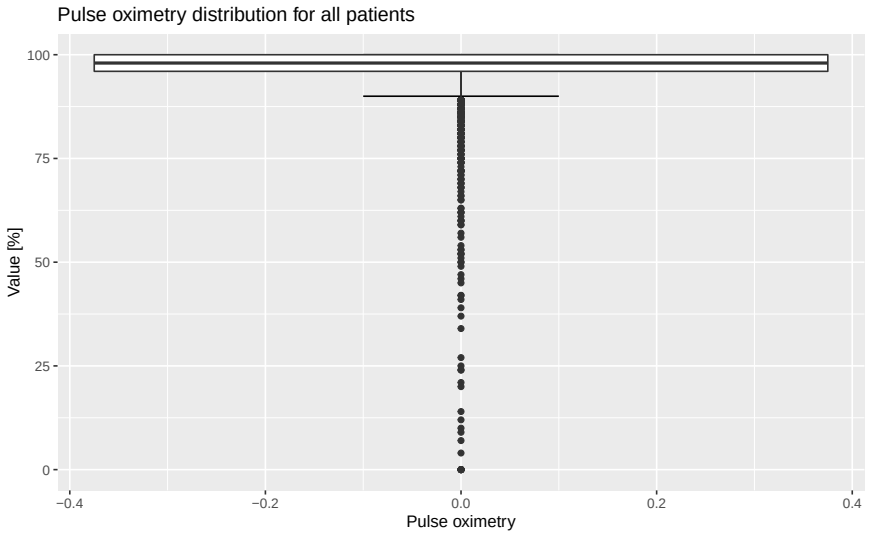
```
hr <- hr %>%
  mutate(
    VALUENUM = if_else((VALUENUM < 0) | (VALUENUM > 300), NA_real_, VALUENUM)
  )

summary(hr$VALUENUM)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    0.00   75.00   87.00   87.77  100.00  189.00       5
```

And now we have a complete dataset with missing data preprocessed for heart rate measurements.

Now repeat the process for pulse oximetry (SpO2).

Observe the distribution of SpO2 for all the patients in the dataset.

```r
sp %>%
  ggplot(aes(y = VALUENUM)) +
  stat_boxplot(geom = "errorbar", width = 0.2, na.rm = TRUE) +
  geom_boxplot(na.rm = TRUE) +
  labs(title = "Pulse oximetry distribution for all patients",
       x = "Pulse oximetry",
       y = 'Value [%]')
```




```r
uw <- quantile(sp$VALUENUM, 3/4, na.rm = TRUE) + (1.5 * IQR(sp$VALUENUM, na.rm = TRUE))
lw <- quantile(sp$VALUENUM, 1/4, na.rm = TRUE) - (1.5 * IQR(sp$VALUENUM, na.rm = TRUE))

above <- sum(sp$VALUENUM > uw, na.rm = TRUE)
below <- sum(sp$VALUENUM < lw, na.rm = TRUE)
```

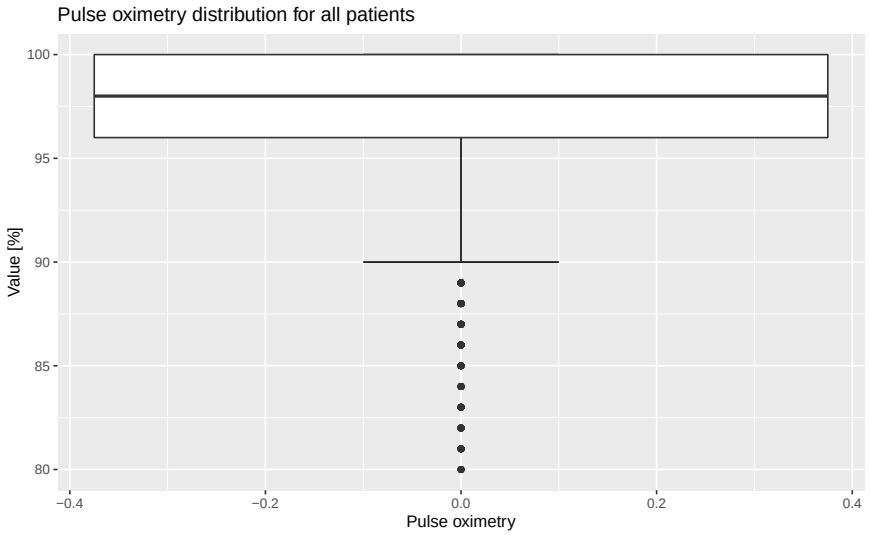
We observe there are a few outliers. 0 observations lie above the upper whisker (106) and 403 observations lie below the lower whisker (90).

As with the heart rate values, All of the pulse oximetry values that we see in these data are within the physiological range. If there were extreme outlier values (the physiological range is [0, 100] %), we would want to correct these. We can recode all values outside the physiological ranges as "NA" in the dataset for all patients.

```r
sp <- sp %>%
  mutate(
    VALUENUM = if_else((VALUENUM < 0) | (VALUENUM > 100), NA_real_, VALUENUM)
  )

summary(sp$VALUENUM)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    0.00   96.00   98.00   97.01  100.00  100.00      15
```

While within the physiological range, we observe that there are several values bellow 80%. Some of these values may correspond to other variables, such as respiratory or heart rate that were mistyped in the system. For our analysis we will recoded them as "NA".

```r
sp <- sp %>%
  mutate(
    VALUENUM = if_else(VALUENUM < 80, NA_real_, VALUENUM)
  )

summary(sp$VALUENUM)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>   80.00   96.00   98.00   97.33  100.00  100.00     131

sp %>%
  ggplot(aes(y = VALUENUM)) +
  stat_boxplot(geom = "errorbar", width = 0.2, na.rm = TRUE) +
  geom_boxplot(na.rm = TRUE) +
  labs(title = "Pulse oximetry distribution for all patients",
       x = "Pulse oximetry",
       y = 'Value [%]')
```




```r
uw <- quantile(sp$VALUENUM, 3/4, na.rm = TRUE) + (1.5 * IQR(sp$VALUENUM, na.rm = TRUE))
lw <- quantile(sp$VALUENUM, 1/4, na.rm = TRUE) - (1.5 * IQR(sp$VALUENUM, na.rm = TRUE))

above <- sum(sp$VALUENUM > uw, na.rm = TRUE)
below <- sum(sp$VALUENUM < lw, na.rm = TRUE)
```

After recoding, 0 observations lie above the upper whisker (106) and 287 observations lie below the lower whisker (90).

Exercise: Plotting missing data

Aim: Analyze and visualize the missing data

Complete cases for heart rate

The function `complete.cases` returns a logical vector indicating which cases *are* complete.

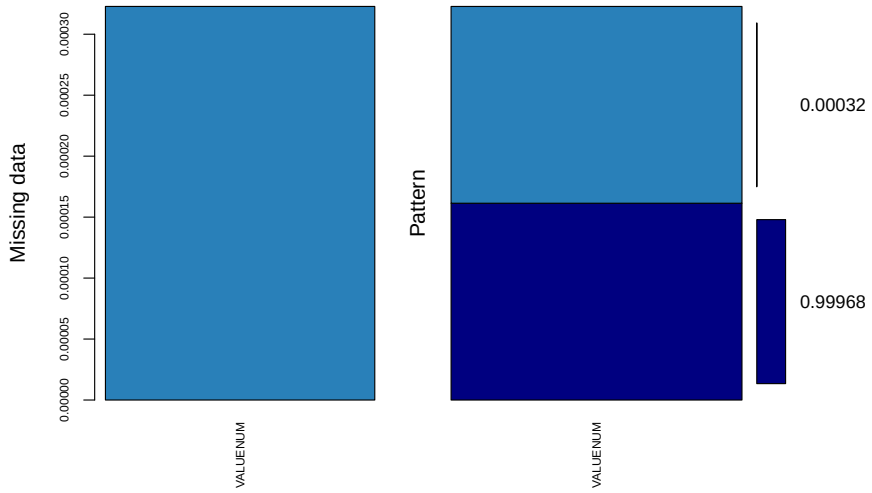
```
hr %>% filter(!complete.cases(SUBJECT_ID))
#> # A tibble: 0 x 14
#> # ... with 14 variables: SUBJECT_ID <int>, HADM_ID <int>, ICUSTAY_ID <int>,
#> #   ITEMID <int>, CHARTTIME <dttm>, STORETIME <dttm>, CGID <int>, VALUE <chr>,
#> #   VALUENUM <dbl>, VALUEUOM <chr>, WARNING <int>, ERROR <int>,
#> #   RESULTSTATUS <chr>, STOPPED <chr>
na_sid <- hr %>% filter(!complete.cases(SUBJECT_ID)) %>% count()
na_sid_pct <- round(100 * na_sid / (hr %>% count()), 2)

hr %>% filter(!complete.cases(VALUEUOM))
#> # A tibble: 0 x 14
#> # ... with 14 variables: SUBJECT_ID <int>, HADM_ID <int>, ICUSTAY_ID <int>,
#> #   ITEMID <int>, CHARTTIME <dttm>, STORETIME <dttm>, CGID <int>, VALUE <chr>,
#> #   VALUENUM <dbl>, VALUEUOM <chr>, WARNING <int>, ERROR <int>,
#> #   RESULTSTATUS <chr>, STOPPED <chr>
na_uom <- hr %>% filter(!complete.cases(VALUEUOM)) %>% count()
na_uom_pct <- round(100 * na_uom / (hr %>% count()), 2)

hr %>% filter(!complete.cases(VALUENUM))
#> # A tibble: 5 x 14
#>   SUBJECT_ID HADM_ID ICUSTAY_ID ITEMID CHARTTIME           STORETIME
#>        <int>   <int>      <int>  <int> <dttm>              <dttm>
#> 1      10069  146672     290490    211 2188-02-12 18:00:00 2188-02-12 18:36:00
#> 2      10089  190301     246080    211 2132-08-06 10:00:00 2132-08-06 10:23:00
#> 3      10126  160445     249805    211 2171-07-14 12:00:00 2171-07-14 12:28:00
#> 4      10126  160445     249805    211 2171-07-29 22:00:00 2171-07-29 22:33:00
#> 5      10127  182839     271544    211 2198-07-04 15:45:00 2198-07-04 17:07:00
#> # ... with 8 more variables: CGID <int>, VALUE <chr>, VALUENUM <dbl>,
#> #   VALUEUOM <chr>, WARNING <int>, ERROR <int>, RESULTSTATUS <chr>,
#> #   STOPPED <chr>
na_val <- hr %>% filter(!complete.cases(VALUENUM)) %>% count()
na_val_pct <- round(100 * na_val / (hr %>% count()), 2)
```

We have 0 observations with missing subjects IDs (0%), 0 observations with missing heart rate units (0%), and we have 5 observations with missing heart rate values (0.03%).

Plotting missing values for heart rate

The `aggr` function from the `VIM` package allows one to calculate and/or plot the amount of missing/imputed values in each variable and in certain combinations of variables.

```
na_hr <- hr %>%
  select(VALUENUM) %>%
  aggr(col = c('navyblue','#2980b9'), numbers = TRUE, sortVars = TRUE,
       cex.axis = 0.7, gap = 3, ylab = c("Missing data","Pattern"))
```




```
#>
#>  Variables sorted by number of missings:
#>  Variable        Count
#>  VALUENUM 0.0003227889
```

We observe that for the case of heart rate there are 99.97% of the values in the data set or 0.03% missing.

Omit cases for heart rate

The function `na.omit` returns the input object with listwise deletion of missing values.

```
hr_complete <- na.omit(hr)
```

And now we have a complete dataset with missing data preprocessed for heart rate measurements.

Now repeat the steps for pulse oximetry.

Complete cases for pulse oximetry

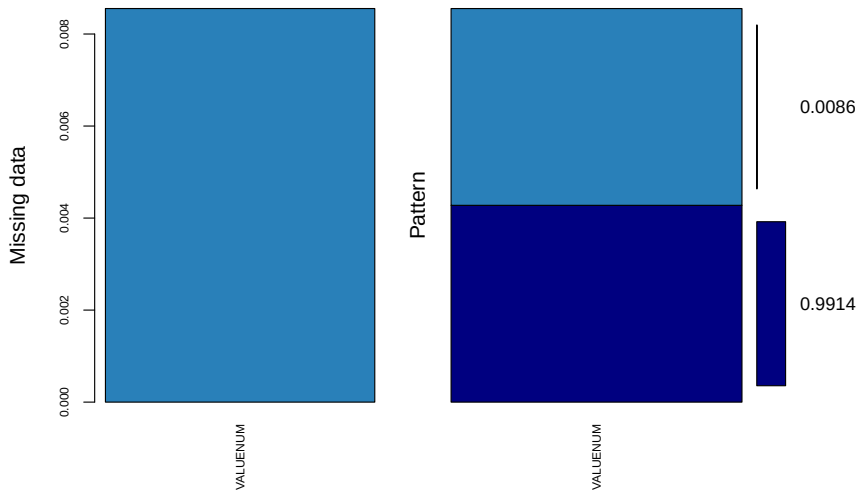
```
sp %>% filter(!complete.cases(SUBJECT_ID))
#> # A tibble: 0 x 14
#> # ... with 14 variables: SUBJECT_ID <int>, HADM_ID <int>, ICUSTAY_ID <int>,
#> #   ITEMID <int>, CHARTTIME <dttm>, STORETIME <dttm>, CGID <int>, VALUE <chr>,
#> #   VALUENUM <dbl>, VALUEUOM <chr>, WARNING <int>, ERROR <int>,
#> #   RESULTSTATUS <chr>, STOPPED <chr>
na_sid <- sp %>% filter(!complete.cases(SUBJECT_ID)) %>% count()
na_sid_pct <- round(100 * na_sid / (sp %>% count()), 2)

sp %>% filter(!complete.cases(VALUEUOM))
#> # A tibble: 0 x 14
#> # ... with 14 variables: SUBJECT_ID <int>, HADM_ID <int>, ICUSTAY_ID <int>,
#> #   ITEMID <int>, CHARTTIME <dttm>, STORETIME <dttm>, CGID <int>, VALUE <chr>,
#> #   VALUENUM <dbl>, VALUEUOM <chr>, WARNING <int>, ERROR <int>,
```

```
#> #   RESULTSTATUS <chr>, STOPPED <chr>
na_uom <- sp %>% filter(!complete.cases(VALUEUOM)) %>% count()
na_uom_pct <- round(100 * na_uom / (sp %>% count()), 2)

sp %>% filter(!complete.cases(VALUENUM))
#> # A tibble: 131 x 14
#>    SUBJECT_ID HADM_ID ICUSTAY_ID ITEMID CHARTTIME           STORETIME
#>         <int>   <int>      <int>  <int> <dttm>              <dttm>
#>  1      10013  165520     264446    646 2125-10-06 21:30:00 2125-10-06 21:17:00
#>  2      10013  165520     264446    646 2125-10-06 22:00:00 2125-10-06 23:32:00
#>  3      10013  165520     264446    646 2125-10-07 11:15:00 2125-10-07 11:19:00
#>  4      10013  165520     264446    646 2125-10-07 12:10:00 2125-10-07 12:14:00
#>  5      10019  177759     228977    646 2163-05-15 06:30:00 2163-05-15 08:28:00
#>  6      10040  157839     272047    646 2147-02-23 16:45:00 2147-02-23 16:59:00
#>  7      10040  157839     272047    646 2147-02-23 11:50:00 2147-02-23 11:51:00
#>  8      10040  157839     272047    646 2147-02-23 23:30:00 2147-02-23 23:19:00
#>  9      10045  126949     203766    646 2129-11-30 22:35:00 2129-11-30 22:43:00
#> 10      10045  126949     203766    646 2129-11-30 22:40:00 2129-11-30 22:43:00
#> # ... with 121 more rows, and 8 more variables: CGID <int>, VALUE <chr>,
#> #   VALUENUM <dbl>, VALUEUOM <chr>, WARNING <int>, ERROR <int>,
#> #   RESULTSTATUS <chr>, STOPPED <chr>
na_val <- sp %>% filter(!complete.cases(VALUENUM)) %>% count()
na_val_pct <- round(100 * na_val / (sp %>% count()), 2)
```

We have 0 observations with missing subjects IDs (0%), 0 observations with missing pulse oximetry units (0%), and we have 131 observations with missing pulse oximetry values (0.86%).

Plotting missing values for pulse oximetry

```
na_sp <- sp %>%
  select(VALUENUM) %>%
  aggr(col = c('navyblue','#2980b9'), numbers = TRUE, sortVars = TRUE,
       cex.axis = 0.7, gap = 3, ylab = c("Missing data","Pattern"))
```



```
#>
#>  Variables sorted by number of missings:
#>  Variable       Count
#>  VALUENUM 0.008553706
```

We observe that for the case of pulse oximetry there are 99.14% of the values in the data set or 0.86% missing.

Omit cases for pulse oximetry

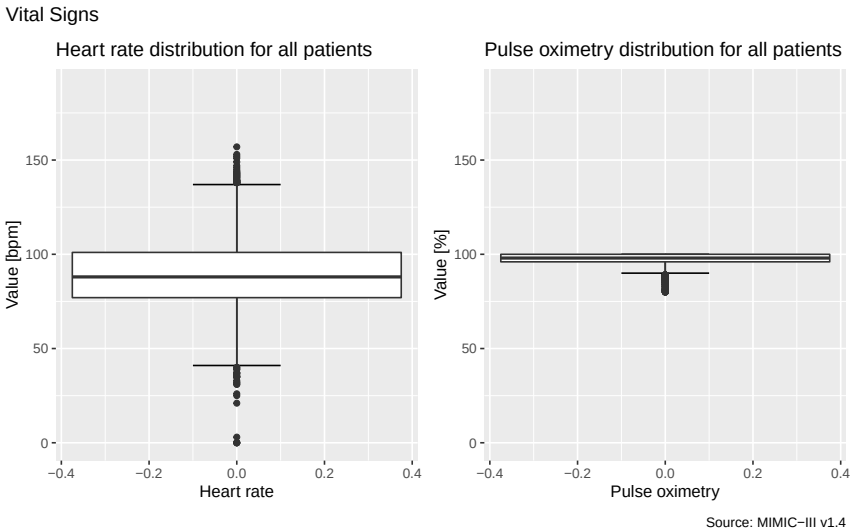
```
sp_complete <- na.omit(sp)
```

And now we have a complete dataset with missing data preprocessed for pulse oximetry measurements.

Attention however, because missing data might not be missing at random (MNAR - see chapter contents for details) and in that case it might be important to identify and handle in a different way the missing values in the dataset. Eg. this can be done through missing data imputation.

We can now see how the distribution looks like for complete and processed data.

```
top <- max(sp$VALUENUM, hr$VALUENUM, na.rm = TRUE)
bot <- min(sp$VALUENUM, hr$VALUENUM, na.rm = TRUE)

hr_g <- hr_complete %>%
  ggplot(aes(y = VALUENUM)) +
  stat_boxplot(geom = "errorbar", width = 0.2, na.rm = TRUE) +
  geom_boxplot(na.rm = TRUE) +
  scale_y_continuous(limits = c(bot, top)) +
  labs(title = "Heart rate distribution for all patients",
       x = "Heart rate",
       y = 'Value [bpm]')

sp_g <- sp_complete %>%
  ggplot(aes(y = VALUENUM)) +
  stat_boxplot(geom = "errorbar", width = 0.2, na.rm = TRUE) +
  geom_boxplot(na.rm = TRUE) +
  scale_y_continuous(limits = c(bot, top)) +
  labs(title = "Pulse oximetry distribution for all patients",
       x = "Pulse oximetry",
       y = 'Value [%]')

hr_g + sp_g + plot_layout(ncol = 2, guides = "collect") +
  plot_annotation(
    title = "Vital Signs",
    caption = "Source: MIMIC-III v1.4"
  )
```



Exercise: Missing data imputation

Aim: To impute missing data using several methods.

There are several approaches for missing data imputation. Here we show how to impute missing data with packages available in R Studio.

Before missing data imputation we must first remove outliers, so our imputation is performed based on values within the physiological ranges.

First, select heart rate and pulse oximetry values in the same dataframe.

```
hr_sp <- vitals %>%
  filter(
    (((ITEMID == 211) | (ITEMID == 220045)) & (toupper(VALUEUOM) == "BPM")) |
    (((ITEMID == 646) | (ITEMID == 220277)) & (VALUEUOM == "%"))
  )
```

And recode as "NA" where values are outside the physiological ranges. We also include a new variable `MEASURE` which quickly identifies whether the observation is for heart rate or pulse oximetry. (This will be used to pivot the data frame wider - see below.)

```
hr_sp <- hr_sp %>%
  mutate(
    MEASURE = ifelse(((ITEMID == 211) | (ITEMID == 220045)), "HR", "SpO2"),
    VALUENUM = ifelse((MEASURE == "HR") &
                        ((VALUENUM > 300) | (VALUENUM < 0)), NA_real_, VALUENUM),
    VALUENUM = ifelse((MEASURE == "SpO2") &
                        ((VALUENUM > 100) | (VALUENUM < 80)), NA_real_, VALUENUM)
  ) %>%
  select(SUBJECT_ID, HADM_ID, ICUSTAY_ID, CHARTTIME, VALUENUM, MEASURE) %>%
  pivot_wider(names_from = MEASURE, values_from = VALUENUM) %>%
  arrange(SUBJECT_ID, HADM_ID, ICUSTAY_ID, CHARTTIME)
```

Imputation using the population median values

We can directly impute the median of the population for each variable.

```
med_hr_sp <- hr_sp %>%
  mutate(
    HR = ifelse(is.na(HR), median(hr_sp$HR, na.rm = TRUE), HR),
    SpO2 = ifelse(is.na(SpO2), median(hr_sp$SpO2, na.rm = TRUE), SpO2)
  )
```

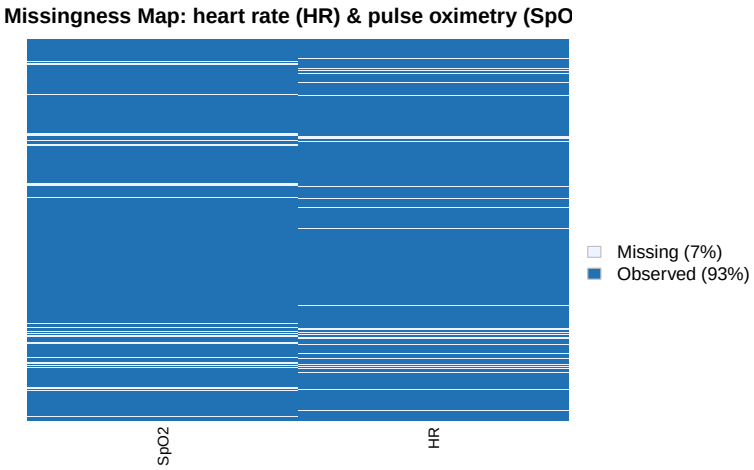
Imputation with the Amelia package

Let's visualize and then impute the missing values with functions from the `Amelia` package - named after Amelia Earhart.

We selected pulse oximetry and heart rate as examples, however this can be performed for all the variables.

Select 1 patient ICU stay as example for visualization.

```
hr_sp %>%
  filter(ICUSTAY_ID == 210989) %>%
  select(HR, SpO2) %>%
  missmap(main = "Missingness Map: heart rate (HR) & pulse oximetry (SpO2)",
          y.labels = NULL,
          y.at = NULL)
```



Now, impute missing data using the `Amelia` package. For the cases (correspondent to rows) where there is no value for heart rate or pulse oximetry, the value is not imputed. With a dataset containing higher amount of variables, we will not have so many of these cases. However, in the cases where we do not have any value we can proceed with eg. case deletion.

The `amelia` function runs the bootstrap EM algorithm on incomplete data and creates imputed datasets.

```
amelia_fit <- hr_sp %>%
  filter(!(is.na(HR) & is.na(SpO2))) %>%
  select(HR, SpO2) %>%
```

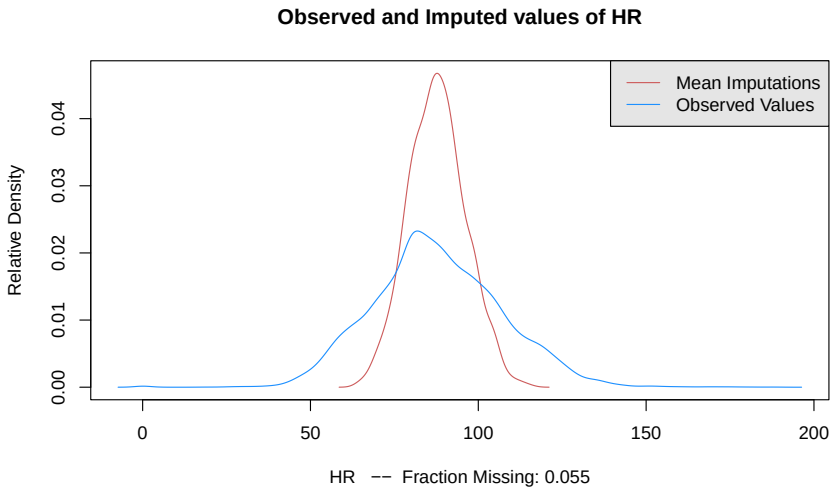
```
  as.data.frame() %>% # data needs to be in a pure data frame or matrix form
  amelia(parallel = "multicore")
#> -- Imputation 1 --
#>
#>   1  2  3  4
#>
#> -- Imputation 2 --
#>
#>   1  2  3  4
#>
#> -- Imputation 3 --
#>
#>   1  2  3  4
#>
#> -- Imputation 4 --
#>
#>   1  2  3  4
#>
#> -- Imputation 5 --
#>
#>   1  2  3  4
```

Analyze the imputed values

We will analyze the imputed values created to understand if the imputation method was adequate. A common practice consists in comparing the distribution of the imputed values and of the observed values. Use the `compare.density` function from the `Amelia` package for this.

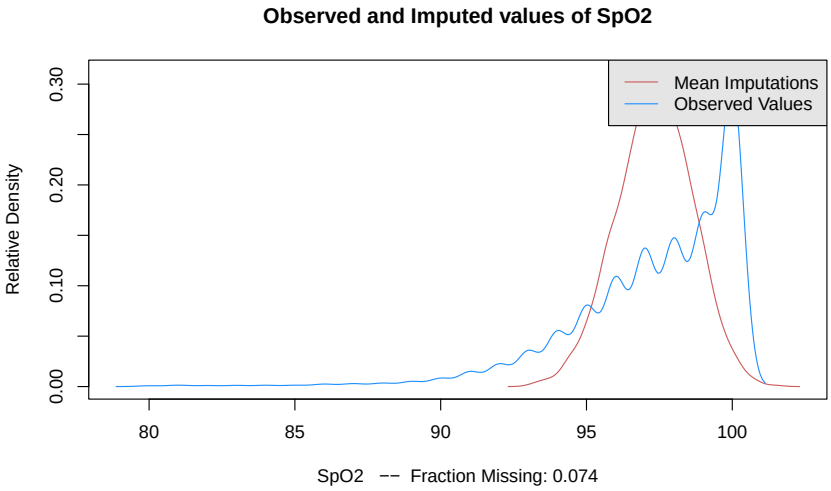
For heart rate:

```
compare.density(amelia_fit, var = "HR")
```



For pulse oximetry:

```
compare.density(amelia_fit, var = "SpO2")
```



We observe that other methods should be applied in order to have a more fitted distribution of the data.