7. MISSING VALUES

NA, , NaN & non-standard missing values.

Recoding non-standard missing values as NA

R represents missing numeric with "NA" (or "" for character variables). R also uses NaN for numeric values that are undefined, such as zero divided by zero. Unfortunately, the use of NA to code for missing values is not universal. Other conventions for coding missing values include using white space, a dash, the value 99, the value 999, N/A, n/a, and many others. For R to understand these as missing values, we merely have to include an argument to read_csv that lists all the ways missing values are coded in the raw data.

Let's imagine that different parts of a cholesterol data set were produced by different technologies and different people. Perhaps the name, age and gender of study participants were collected from a paper-based bubble-in form and occasionally people skip questions. Perhaps the initial weight was recorded manually by a technician that used one convention for missing values, while the final weight was recorded by a different technician who used another. Lastly, perhaps the cholesterol values were generated by a medical device that uses a fourth convention for missing values. Now let's imagine that none of them use "NA" to code for missing values. What a nightmare!

One option is to open the file in Excel and manually fix all the missing values to read NA. This is a tedious and error prone process though, and possibly impractical for extremely large data sets. Secondly, this practice changes the raw data! Even with the best intentions, directly altering the raw data file is dangerous and ill-advised for reasons already discussed.

Fortunately, a single argument to the read_csv command may be able to solve all of these problems. First though, let's import a dataframe with the problems mentioned above and use the summary command to explore the data.

```
read_csv('data/Friends_Cholesterol_Missing.csv') %>% summary
```

```
##      name                age            gender          height     
##  Length:40          Min.   :21.00   Min.   :  0.0   Min.   :54.00  
##  Class :character   1st Qu.:32.50   1st Qu.:  0.0   1st Qu.:61.75  
##  Mode  :character   Median :40.50   Median :  1.0   Median :65.00  
##                     Mean   :39.62   Mean   : 75.4   Mean   :64.65  
##                     3rd Qu.:47.25   3rd Qu.:  1.0   3rd Qu.:67.00  
##                     Max.   :55.00   Max.   :999.0   Max.   :74.00  
##      group       weight_i             hdl_i              ldl_i          
##  Min.   :0.0   Length:40          Length:40          Length:40          
##  1st Qu.:0.0   Class :character   Class :character   Class :character   
##  Median :0.5   Mode  :character   Mode  :character   Mode  :character   
##  Mean   :0.5                                                            
##  3rd Qu.:1.0                                                            
##  Max.   :1.0                                                            
##    weight_f            hdl_f              ldl_f          
##  Length:40          Length:40          Length:40          
##  Class :character   Class :character   Class :character   
##  Mode  :character   Mode  :character   Mode  :character   
## 
```

```
##
##
```

From the output, we can see clear evidence of problems. The maximum value of gender is 999, and the weight and cholesterol variables are all interpreted as character variables. To explore the problem further, we can view the first 10 rows of our data using the head command.

```
read_csv('data/Friends_Cholesterol_Missing.csv') %>% head(10)
```

```
## # A tibble: 10 x 11
##    name       age gender height group weight_i hdl_i ldl_i weight_f hdl_f ldl_f
##    <chr>    <dbl>  <dbl>  <dbl> <dbl> <chr>    <chr> <chr> <chr>    <chr> <chr>
##  1 William     26      0     69     0 172      49    81    156      48    86
##  2 Richard     37      0     67     0 184      54    95    165      53    93
##  3 Joseph      45      0     66     0 191      49    106   180      46    112
##  4 Daniel      35    999     72     0 181      50    98    missing  51    100
##  5 Jennifer    44      1     64     0 203      44    107   189      47    111
##  6 Barbara     23      1     61     0 193      ###   103   187      59    111
##  7 Susan       31      1     61     0 N/A      62    107   177      60    112
##  8 Jessica     44      1     66     0 192      54    91    181      60    101
##  9 Kimberly    35      1     62     0 200      54    98    184      ###   96
## 10 Emily       41      1     61     0 203      48    96    199      49    94
```

The output above is quite revealing. We can see that missing values are coded as 999, N/A, ### and “missing”. Now we merely need to run the import command again with an “na =” argument that identifies all the missing value codes with a string vector.

```
read_csv('data/Friends_Cholesterol_Missing.csv', na = c(999, "N/A", "###", "missing")) %>% summary
```

```
##      name                age            gender           height     
##  Length:40          Min.   :21.00   Min.   :0.0000   Min.   :54.00  
##  Class :character   1st Qu.:32.50   1st Qu.:0.0000   1st Qu.:61.75  
##  Mode  :character   Median :40.50   Median :1.0000   Median :65.00  
##                     Mean   :39.62   Mean   :0.5135   Mean   :64.65  
##                     3rd Qu.:47.25   3rd Qu.:1.0000   3rd Qu.:67.00  
##                     Max.   :55.00   Max.   :1.0000   Max.   :74.00  
##                                     NA's   :3                       
##      group        weight_i         hdl_i            ldl_i           weight_f    
##  Min.   :0.0   Min.   :172.0   Min.   :44.00   Min.   : 67.00   Min.   :156.0  
##  1st Qu.:0.0   1st Qu.:184.0   1st Qu.:50.00   1st Qu.: 90.25   1st Qu.:171.0  
##  Median :0.5   Median :191.0   Median :53.00   Median : 99.00   Median :180.0  
##  Mean   :0.5   Mean   :191.3   Mean   :52.53   Mean   : 99.24   Mean   :180.1  
##  3rd Qu.:1.0   3rd Qu.:199.0   3rd Qu.:54.75   3rd Qu.:109.00   3rd Qu.:189.0  
##  Max.   :1.0   Max.   :214.0   Max.   :62.00   Max.   :137.00   Max.   :203.0  
##                NA's   :3       NA's   :2       NA's   :2        NA's   :3      
##      hdl_f           ldl_f       
##  Min.   :46.00   Min.   : 47.00  
##  1st Qu.:51.00   1st Qu.: 80.50  
##  Median :55.00   Median : 89.00  
##  Mean   :54.89   Mean   : 89.39  
##  3rd Qu.:58.00   3rd Qu.:101.75  
##  Max.   :64.00   Max.   :140.00  
##  NA's   :3       NA's   :2       
```

Viola! All missing values are now coded as “NA”, R’s internal structure for missing values.

Dealing with NA's

Missing values can cause commands to fail, produce anomalies in plots and cause other problems. If missing values can be replaced with real values, without invalidating the study, this is obviously the best solution. When we are really stuck with missing values though, we have four common strategies in R.

1. Ignore missing values from inside commands using the na.rm = TRUE argument.
2. Remove rows with missing values for a single variable using filter(df, is.na()).
3. Remove rows with any missing value using filter(df, complete.cases(df)).
4. Impute the missing values (replace NA with a "best guess").

Before exploring these approaches, let's clear our environment, import new data and identify the presence or absence of missing values using the summary command.

```
rm(list = ls())
df <-
    read_csv("data/cardiac.csv") %>%
    mutate(sex = factor(sex),
           sex = fct_recode(sex, 'female' = '0', 'male' = '1'))

summary(df)
```

```
##    subject                age            sex          height         group    
##  Length:1000        Min.   :30.00   female:493   Min.   :56.0   Min.   :0.0  
##  Class :character   1st Qu.:35.00   male  :507   1st Qu.:64.0   1st Qu.:0.0  
##  Mode  :character   Median :40.00                Median :67.0   Median :0.5  
##                     Mean   :40.02                Mean   :67.1   Mean   :0.5  
##                     3rd Qu.:45.00                3rd Qu.:70.0   3rd Qu.:1.0  
##                     Max.   :50.00                Max.   :80.0   Max.   :1.0  
##                                                                              
##    weight_t0    systolic_t0     diastolic_t0     weight_t2      systolic_t2   
##  Min.   :138   Min.   :123.0   Min.   : 86.0   Min.   :130.0   Min.   :112.0  
##  1st Qu.:170   1st Qu.:137.0   1st Qu.:104.0   1st Qu.:166.0   1st Qu.:131.0  
##  Median :184   Median :140.0   Median :109.0   Median :180.0   Median :137.0  
##  Mean   :186   Mean   :140.5   Mean   :108.8   Mean   :182.1   Mean   :136.5  
##  3rd Qu.:202   3rd Qu.:144.0   3rd Qu.:113.0   3rd Qu.:197.0   3rd Qu.:142.0  
##  Max.   :264   Max.   :158.0   Max.   :132.0   Max.   :263.0   Max.   :158.0  
##                                                NA's   :17      NA's   :17     
##   diastolic_t2     weight_t4       systolic_t4     diastolic_t4  
##  Min.   : 77.0   Min.   : 129.0   Min.   :104.0   Min.   : 71.0  
##  1st Qu.:100.0   1st Qu.: 163.0   1st Qu.:125.0   1st Qu.: 96.0  
##  Median :106.0   Median : 177.0   Median :135.0   Median :103.5  
##  Mean   :105.7   Mean   : 181.3   Mean   :133.3   Mean   :103.1  
##  3rd Qu.:111.0   3rd Qu.: 195.0   3rd Qu.:141.0   3rd Qu.:110.0  
##  Max.   :132.0   Max.   :1810.0   Max.   :158.0   Max.   :130.0  
##  NA's   :17      NA's   :30       NA's   :30      NA's   :30
```

By inspecting the output, we can see that three variables, weight_t2, systolic_t2, and diastolic_t2 17 each have 17 missing values. Three other variables, weight_t4, systolic_t4 and diastolic_t4 variables, have missing 30 missing values each. Let's see what happens when we try to generate descriptive statistics with variables that have missing values.

```
mean(df$systolic_t2)
```

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

```
tapply(df$weight_t2, df$sex, max)
```

```
## female   male
##     NA     NA
```

Assuming we are executing code sequentially within the script (I.E. top to bottom), we should see "NA" returned for both of the above commands. The commands didn't fail to execute or throw an error, they merely returned (mostly) useless information. Essentially, R is telling us that missing values exist and we must either fix the problem or explicitly tell R to ignore missing values.

na.rm = TRUE

We can usually tell R to ignore missing values by including the optional argument, na.rm = TRUE. An advantage to this approach is that we do not change the dataframe, such as by eliminating an entire row with 1 or more missing values. Let's run the same commands as above, but with the na.rm = TRUE argument.

```
mean(df$systolic_t2, na.rm = TRUE)
```

```
## [1] 136.4619
```

```
tapply(df$weight_t2, df$sex, max, na.rm = TRUE)
```

```
## female   male
##    207    263
```

Problem solved!

na.omit & complete.cases

We sometimes wish to evaluate just the cases that have no missing values, the complete cases, as well as characterize the cases that have one or more missing values. Perhaps we wish to know if the missing values are randomly spread through the data, or if they are concentrated in a particular group. We can accomplish this by splitting our dataframe into two versions, one with complete cases and one with incomplete cases.

The na.omit(df) command needs only one argument, a dataframe, and it outputs a dataframe with no missing values. The complete.cases command takes the same input, a dataframe, but outputs a logical vector. Rows that have no missing values will return TRUE, while those with 1 or more NA values will return FALSE. The complete.cases command is most often used in combination with commands such as filter that take logical arguments, or with commands that perform some arithmetic function based on the quantitative attribute of all logical variables, TRUE = 0 and FALSE = 1.

summarizing the complete cases Below we first count the complete and incomplete cases in df. Then we create two new dataframes with observations that have no missing values using na.omit and complete.cases. Next we count the complete rows using nrow and create summary statistics with the summary command. Lastly, we verify that the dataframes created with na.omit and complete.cases contain the same information.

```
# Count the complete and incomplete cases
complete.cases(df) %>% summary
```

```
##    Mode   FALSE    TRUE
## logical      30     970
```

```
# Equivalent approaches for creating dataframes with complete cases
df.c <- filter(df, complete.cases(df))
df.o <- na.omit(df)
```

```r
# Explore the complete cases
nrow(df.c)
```

```
## [1] 970
```

```r
summary(df.c)
```

```
##    subject               age            sex          height     
##  Length:970         Min.   :30.00   female:477   Min.   :56.00  
##  Class :character   1st Qu.:35.00   male  :493   1st Qu.:64.00  
##  Mode  :character   Median :40.00                Median :67.00  
##                     Mean   :40.05                Mean   :67.09  
##                     3rd Qu.:45.00                3rd Qu.:70.00  
##                     Max.   :50.00                Max.   :80.00  
##      group          weight_t0      systolic_t0     diastolic_t0     weight_t2  
##  Min.   :0.0000   Min.   :138.0   Min.   :123.0   Min.   : 86.0   Min.   :130  
##  1st Qu.:0.0000   1st Qu.:170.0   1st Qu.:137.0   1st Qu.:104.0   1st Qu.:166  
##  Median :1.0000   Median :184.0   Median :140.0   Median :109.0   Median :180  
##  Mean   :0.5082   Mean   :186.1   Mean   :140.5   Mean   :108.8   Mean   :182  
##  3rd Qu.:1.0000   3rd Qu.:202.0   3rd Qu.:144.0   3rd Qu.:113.0   3rd Qu.:197  
##  Max.   :1.0000   Max.   :264.0   Max.   :155.0   Max.   :132.0   Max.   :263  
##   systolic_t2     diastolic_t2     weight_t4       systolic_t4   
##  Min.   :112.0   Min.   : 77.0   Min.   : 129.0   Min.   :104.0  
##  1st Qu.:131.0   1st Qu.:100.0   1st Qu.: 163.0   1st Qu.:125.0  
##  Median :137.0   Median :106.0   Median : 177.0   Median :135.0  
##  Mean   :136.4   Mean   :105.7   Mean   : 181.3   Mean   :133.3  
##  3rd Qu.:142.0   3rd Qu.:111.0   3rd Qu.: 195.0   3rd Qu.:141.0  
##  Max.   :158.0   Max.   :132.0   Max.   :1810.0   Max.   :158.0  
##   diastolic_t4  
##  Min.   : 71.0  
##  1st Qu.: 96.0  
##  Median :103.5  
##  Mean   :103.1  
##  3rd Qu.:110.0  
##  Max.   :130.0
```

```r
# verify that both approaches produce the same information
all.equal(summary(df.c), summary(df.o))
```

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

Warning, the dataframes with exclusively complete cases should be used with caution! If a certain row has a missing value for just one variable, the commands above will remove the entire row, discarding useful information in the columns with observed values. Using these dataframes for subsequent analyses can then produce erroneous results that don't involve all the original data. Consequently, the dataframes above are often utilized just to exploring the data and look for patterns related to missing values.

summarizing the incomplete cases To produce a dataframe of incomplete observations, we need only add the negation operator, !, in front of complete.cases. Negation will switch all the TRUE and FALSE values. Our filter command will now retain observations WITH one or more missing values. Below we create a new dataframe with only the observations that have missing values. Then we count the rows with nrow and create summary statistics with the summary command.

```r
df.m <- filter(df, !complete.cases(df))
nrow(df.m)
```

```
## [1] 30
```

```
summary(df.m)
```

```
##    subject                age            sex          height         group       
##  Length:30          Min.   :31.00   female:16   Min.   :58.0   Min.   :0.0000  
##  Class :character   1st Qu.:33.00   male  :14   1st Qu.:64.0   1st Qu.:0.0000  
##  Mode  :character   Median :39.50               Median :67.0   Median :0.0000  
##                     Mean   :39.03               Mean   :67.3   Mean   :0.2333  
##                     3rd Qu.:43.75               3rd Qu.:70.5   3rd Qu.:0.0000  
##                     Max.   :50.00               Max.   :73.0   Max.   :1.0000  
##                                                                                
##    weight_t0      systolic_t0     diastolic_t0     weight_t2    
##  Min.   :148.0   Min.   :132.0   Min.   : 98.0   Min.   :157.0  
##  1st Qu.:168.0   1st Qu.:137.2   1st Qu.:106.0   1st Qu.:178.0  
##  Median :188.0   Median :140.5   Median :111.0   Median :186.0  
##  Mean   :185.4   Mean   :141.3   Mean   :111.1   Mean   :186.2  
##  3rd Qu.:201.8   3rd Qu.:143.0   3rd Qu.:114.0   3rd Qu.:200.0  
##  Max.   :220.0   Max.   :158.0   Max.   :132.0   Max.   :209.0  
##                                                  NA's   :17     
##   systolic_t2     diastolic_t2     weight_t4    systolic_t4   diastolic_t4 
##  Min.   :130.0   Min.   : 92.0   Min.   : NA   Min.   : NA   Min.   : NA  
##  1st Qu.:134.0   1st Qu.:109.0   1st Qu.: NA   1st Qu.: NA   1st Qu.: NA  
##  Median :141.0   Median :112.0   Median : NA   Median : NA   Median : NA  
##  Mean   :140.7   Mean   :112.8   Mean   :NaN   Mean   :NaN   Mean   :NaN  
##  3rd Qu.:146.0   3rd Qu.:117.0   3rd Qu.: NA   3rd Qu.: NA   3rd Qu.: NA  
##  Max.   :154.0   Max.   :132.0   Max.   : NA   Max.   : NA   Max.   : NA  
##  NA's   :17      NA's   :17      NA's   :30    NA's   :30    NA's   :30   
```

is.na

As an alternative to filtering rows with any missing values, we can filter based on a single variable using is.na. Like complete.cases, is.na is a logical operator that returns TRUE or FALSE. Unlike complete.cases, is.na takes a single variable as an argument. We should still be careful when using is.na though, as filtering a dataframe with an is.na argument can still throw away useful information. For example, the filtering command below, throws away ALL information in rows that have a missing weight_t2 value.

```
df.c.weight_t2 <- df %>% filter(!is.na(weight_t2))

nrow(df)
```

```
## [1] 1000
```

```
nrow(df.c)
```

```
## [1] 970
```

```
nrow(df.c.weight_t2)
```

```
## [1] 983
```

One good use of is.na is for including missing value counts in summary tables created with summarize as shown below. Logical output can be quantified numerically because the output actually has numeric value, TRUE = 0 and FALSE = 1. The approach below using summarize below provides useful information without changing our dataframe or cluttering our environment with additional objects.

total observations	complete observations	time 2 average weight	time 2 missing values
1000	970	182	17

```
summarize(df,
          'total observations' = n(), # n() is a tidyverse command for counting rows
          'complete observations' = sum(complete.cases(df)),
          'time 2 average weight' = mean(weight_t2, na.rm = TRUE),
          'time 2 missing values' = sum(is.na(weight_t2))) %>%
    round %>%
    kable %>% kable_classic(full_width = F)
```

Imputation

Advanced statistical approaches sometimes benefit from replacing missing values with a best guess. If our best guesses are better than random guess, we often achieve a predictive model that is more accurate than one that ignores missing values. While these statistical models are well beyond the scope of this course, the process of imputation can be straight forward. For example, in the chunk below we use the replace command inside of mutate to replace all NAs for the weight_t2 variable with the median value of weight_2.

```
df %>%
    mutate(weight_t2 = replace(weight_t2,
                               is.na(weight_t2),
                               median(weight_t2, na.rm = TRUE))) %>%
    summary()
```

```
##    subject               age            sex          height         group    
##  Length:1000        Min.   :30.00   female:493   Min.   :56.0   Min.   :0.0  
##  Class :character   1st Qu.:35.00   male  :507   1st Qu.:64.0   1st Qu.:0.0  
##  Mode  :character   Median :40.00                Median :67.0   Median :0.5  
##                     Mean   :40.02                Mean   :67.1   Mean   :0.5  
##                     3rd Qu.:45.00                3rd Qu.:70.0   3rd Qu.:1.0  
##                     Max.   :50.00                Max.   :80.0   Max.   :1.0  
##                                                                              
##    weight_t0     systolic_t0     diastolic_t0     weight_t2      systolic_t2   
##  Min.   :138   Min.   :123.0   Min.   : 86.0   Min.   :130.0   Min.   :112.0  
##  1st Qu.:170   1st Qu.:137.0   1st Qu.:104.0   1st Qu.:166.0   1st Qu.:131.0  
##  Median :184   Median :140.0   Median :109.0   Median :180.0   Median :137.0  
##  Mean   :186   Mean   :140.5   Mean   :108.8   Mean   :182.1   Mean   :136.5  
##  3rd Qu.:202   3rd Qu.:144.0   3rd Qu.:113.0   3rd Qu.:197.0   3rd Qu.:142.0  
##  Max.   :264   Max.   :158.0   Max.   :132.0   Max.   :263.0   Max.   :158.0  
##                                                                NA's   :17     
##   diastolic_t2     weight_t4       systolic_t4     diastolic_t4  
##  Min.   : 77.0   Min.   : 129.0   Min.   :104.0   Min.   : 71.0  
##  1st Qu.:100.0   1st Qu.: 163.0   1st Qu.:125.0   1st Qu.: 96.0  
##  Median :106.0   Median : 177.0   Median :135.0   Median :103.5  
##  Mean   :105.7   Mean   : 181.3   Mean   :133.3   Mean   :103.1  
##  3rd Qu.:111.0   3rd Qu.: 195.0   3rd Qu.:141.0   3rd Qu.:110.0  
##  Max.   :132.0   Max.   :1810.0   Max.   :158.0   Max.   :130.0  
##  NA's   :17      NA's   :30       NA's   :30      NA's   :30     
```

Note that we could have also used the arithmetic average (the mean) of weight_t2 to impute values. Even better would be to use values that are based on correlations within the data. For example, if two subjects, A and B, are the same sex, age, height and weight and have similar initial and final hdl cholesterol and similar initial ldl cholesterol levels, it is reasonable to guess that they their final ldl cholesterol levels will be similar. If this value is missing for subject B, one strategy is to use subject A's ldl cholesterol as a best guess for subject B. Methods exist that take advantage of these correlations to impute missing values, but they are well outside the scope of this course.