6. DESCRIPTIVE STATISTICS

Now that we know how to import and mutate data to create a mature and functional dataframe, we usually want to continue by exploring our data through descriptive statistics and graphs. Again, we can choose between base R commands or the more versatile and powerful collection of Tidyverse commands. Let's jump right in and start with importing data using the import pipe chain developed in lesson 5.

```
df <-
    read_csv(file = "data/Friends_Cholesterol.csv",
             col_types = "cnfnfnnnnnn") %>%
    rename(sex = gender) %>%

    # recode the sex and group variables

    mutate(sex = fct_recode(sex, 'male' = '0', 'female' = '1'),
           group = fct_recode(group, 'control' = '0', 'statin' = '1'),

           # add 6 more variables within mutate

           tc_i = hdl_i + ldl_i,
           tc_f = hdl_f + ldl_f,
           bmi_i = (weight_i/(height)^2) * 703,
           bmi_f = (weight_f/(height)^2) * 703,
           weight_min = weight_i * 18.5 / bmi_i,
           weight_max = weight_i * 24.9 / bmi_i,

           # add categorical weight change recommendation variable within mutate

           weight_chg_rec = ifelse(test = weight_i >= weight_max,
                                   yes = "decrease weight",
                                   no = ifelse(test = weight_i <= weight_min,
                                               yes = "increase weight",
                                               no = "no change rec")),

           # add quantitative weight change recommendation variable within mutate

           weight_chg_val = ifelse(test = weight_i >= weight_max,
                                   yes = weight_max - weight_i,
                                   no = ifelse(test = weight_i <= weight_min,
                                               yes = weight_min - weight_i,
                                               no = 0))
           )
```

Base R Descriptive Statistics

Base R uses intuitively named descriptive statistics commands that work with data frames and vectors. Often the command is as simple as one word with a data object passed into the command.

```
# Base R commands with datafames
mean(df$height)
```

```
## [1] 64.65
```

```
min(df$height)
```

```
## [1] 54
```

```
max(df$height)
```

```
## [1] 74
```

```
range(df$height)
```

```
## [1] 54 74
```

```
sd(df$age)
```

```
## [1] 9.906453
```

```
# Base R commands with vectors
mean(0:100)
```

```
## [1] 50
```

```
min(0:100)
```

```
## [1] 0
```

```
max(0:100)
```

```
## [1] 100
```

```
range(0:100)
```

```
## [1]   0 100
```

```
sd(0:100)
```

```
## [1] 29.30017
```

The summary command

Like many commands in R, the summary command uses different methods for different types of objects. For example, summary will produce different output for dataframes, categorical variables and quantitative variables as shown below. To see a list of all the types of summary methods in base r, execute methods(summary) in the r console. Furthermore, when we load packages such as tidyverse, those packages often add additional methods to commands like summary. Presently, base r has 32 different methods available with the summary command. Loading tidyverse adds another 15 for a total of 47 methods available with one command. An incredible feature about R is that the user doesn't have to choose form this list of methods. In the vast majority of scenarios, we just pass an object into summary, R recognizes the type of object and applies the appropriate method.

```
summary(df$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   21.00   32.50   40.50   39.62   47.25   55.00
```

```
summary(df$sex)
```

```
##   male female
##     20     20
```

```
summary(df)
```

```
##      name                age            sex         height          group
##  Length:40          Min.   :21.00   male  :20   Min.   :54.00   control:20
##  Class :character   1st Qu.:32.50   female:20   1st Qu.:61.75   statin :20
##  Mode  :character   Median :40.50               Median :65.00
##                     Mean   :39.62               Mean   :64.65
##                     3rd Qu.:47.25               3rd Qu.:67.00
##                     Max.   :55.00               Max.   :74.00
##     weight_i         hdl_i           ldl_i           weight_f
##  Min.   :172.0   Min.   :44.00   Min.   : 67.00   Min.   :156.0
##  1st Qu.:184.8   1st Qu.:50.00   1st Qu.: 89.25   1st Qu.:170.8
##  Median :192.0   Median :53.50   Median : 98.00   Median :179.0
##  Mean   :192.0   Mean   :52.77   Mean   : 98.72   Mean   :179.5
##  3rd Qu.:199.2   3rd Qu.:55.25   3rd Qu.:109.00   3rd Qu.:189.0
##  Max.   :214.0   Max.   :62.00   Max.   :137.00   Max.   :203.0
##      hdl_f           ldl_f             tc_i            tc_f
##  Min.   :46.00   Min.   : 47.00   Min.   :118.0   Min.   :104.0
##  1st Qu.:51.75   1st Qu.: 81.50   1st Qu.:140.8   1st Qu.:136.2
##  Median :55.00   Median : 89.00   Median :149.0   Median :144.0
##  Mean   :55.25   Mean   : 90.08   Mean   :151.5   Mean   :145.3
##  3rd Qu.:59.00   3rd Qu.:102.25   3rd Qu.:163.8   3rd Qu.:158.0
##  Max.   :64.00   Max.   :140.00   Max.   :189.0   Max.   :190.0
##      bmi_i           bmi_f         weight_min      weight_max
##  Min.   :22.21   Min.   :20.03   Min.   : 76.74   Min.   :103.3
##  1st Qu.:28.82   1st Qu.:26.18   1st Qu.:100.35   1st Qu.:135.1
##  Median :32.45   Median :30.33   Median :111.18   Median :149.6
##  Mean   :32.91   Mean   :30.80   Mean   :110.51   Mean   :148.7
##  3rd Qu.:36.71   3rd Qu.:34.61   3rd Qu.:118.13   3rd Qu.:159.0
##  Max.   :49.18   Max.   :47.98   Max.   :144.11   Max.   :194.0
##  weight_chg_rec     weight_chg_val
##  Length:40          Min.   :-100.72
##  Class :character   1st Qu.: -64.60
##  Mode  :character   Median : -45.35
##                     Mean   : -43.97
##                     3rd Qu.: -25.00
##                     Max.   :   0.00
```

Frequency tables

For categorical data (factor variables) we often wish to know the frequency of observations for different groups and combinations of groups. Base r provides the table command that provides this information as demonstrated below.

```
table(df$sex, df$group)
```

```
##
##          control statin
##   male         9     11
##   female      11      9
```

Grouped statistics with tapply

Often we wish to know descriptive statistics of quantitative variables at all levels of some other factor variable, such as sex, male and female. The base R command tapply is designed for this exact purpose and will return a descriptive statistic (or other function) of one variable for each level of a factor variable.

```
tapply(X = df$height,
       INDEX = df$sex,
       FUN = mean)
```

```
##   male female
##  67.45  61.85
```

```
# Note that if we use the correct order (X variable, index variable, function), we can omit
# the argument names.
tapply(df$height, df$sex, mean)
```

```
##   male female
##  67.45  61.85
```

Other looping functions

Base R includes a collection of functions that are designed to apply a function over the elements of a list, dataframe or vector. The most common are apply, sapply and lapply. These can be useful commands, but because the output is generally not a dataframe, I tend to favor tidyverse approaches that work more predictably with dataframes.

```
# return the mean of every numeric variable in our dataframe as a vector.
# Here I use select_if, a variation on the select command that allows us to
# select variables based on variable types.
df %>% select_if(is.numeric) %>% apply(MARGIN = 2, FUN = mean)
```

```
##            age         height       weight_i          hdl_i          ldl_i
##       39.62500       64.65000      192.02500       52.77500       98.72500
##       weight_f          hdl_f          ldl_f           tc_i           tc_f
##      179.47500       55.25000       90.07500      151.50000      145.32500
##          bmi_i          bmi_f     weight_min     weight_max weight_chg_val
##       32.91006       30.80461      110.50789      148.73765      -43.96706
```

```
# return the mean of every numeric variable in our dataframe as a list.
# note the vertical orientation of list output.
df %>% select_if(is.numeric) %>% lapply(mean)
```

```
## $age
## [1] 39.625
##
## $height
## [1] 64.65
##
## $weight_i
## [1] 192.025
##
## $hdl_i
## [1] 52.775
##
## $ldl_i
```

```
## [1] 98.725
##
## $weight_f
## [1] 179.475
##
## $hdl_f
## [1] 55.25
##
## $ldl_f
## [1] 90.075
##
## $tc_i
## [1] 151.5
##
## $tc_f
## [1] 145.325
##
## $bmi_i
## [1] 32.91006
##
## $bmi_f
## [1] 30.80461
##
## $weight_min
## [1] 110.5079
##
## $weight_max
## [1] 148.7377
##
## $weight_chg_val
## [1] -43.96706
```

```
# return the mean of every numeric variable in our dataframe using the most
# simple data structure available, a vector in this case.
df %>% select_if(is.numeric) %>% sapply(mean)
```

```
##            age         height       weight_i          hdl_i          ldl_i
##       39.62500       64.65000      192.02500       52.77500       98.72500
##       weight_f          hdl_f          ldl_f           tc_i           tc_f
##      179.47500       55.25000       90.07500      151.50000      145.32500
##          bmi_i          bmi_f     weight_min     weight_max weight_chg_val
##       32.91006       30.80461      110.50789      148.73765      -43.96706
```

Tidyverse Descriptive Statistics

Summary tables: group_by & summarize

With the tidyverse commands group_by and summarize we can create a new dataframe with summary statistics, and optionally grouped the output by one or more factor variables. Because these summary dataframes are for presentation only, we can also break some of the naming rules for aesthetic purposes by using back ticks as shown below. Lastly, we can use commands from the knitr and kableExtra packages that format our dataframe with publication-ready aesthetics.

sex	group	Count	Age Mean	Age Median	Age min	Height Mean	Height SD
male	control	9	39.77778	40	26	66.66667	4.330127
male	statin	11	38.81818	41	21	68.09091	3.448320
female	control	11	38.09091	40	23	62.36364	3.139195
female	statin	9	42.33333	47	23	61.22222	3.345810

```
df %>%
  group_by(sex, group) %>%
  summarize(Count = n(),
            `Age Mean` = mean(age),
            `Age Median` = median(age),
            `Age min` = min(age),
            `Height Mean` = mean(height),
            `Height SD` = sd(height)) %>%

# The next two commands below are from the knitr and kable_Extra package and work
# together to render a more aesthetic table. The table will appear inline with
# your R Markdown file or under the Viewer tab of R Studio (bottom right)
# depending on your settings.

  kable() %>%
    kable_classic(full_width = F)
```

Frequency tables with janitor

With multiple categorical variables, we often wish to know conditional percentages in addition to observation counts. Unfortunately, the base R table command is ill-equiped for this purpose. The Janitor package provides the tabyl command (the y is not a typo) for creating frequency tables with conditional percentages, and has the added benefit of being tidyverse aligned. However, because the package is not formally part of the tidyverse suite, the package must be loaded with a separate library command, or added to the p_load command in the setup code chunk.

Below we load the janitor package and create a frequency table for the base R data set called starwars. Note that because the tabyl command is tidyverse aligned, we can perform filtering steps prior to generating the table, but in a single short pipe chain.

```
library(janitor)

starwars %>%
  filter(species=='Human') %>%
  tabyl(gender, eye_color)
```

```
##     gender blue blue-gray brown dark hazel yellow
##   feminine    3         0     5    0     1      0
##  masculine    9         1    12    1     1      2
```

After the initial table setup using tabyl, we can "adorn" the table with additional features linked together in a pipe chain. Lastly, because the output is still a dataframe, we can pipe into the kable() and kable_classic commands to create an aesthetically pleasing table ready for presentation or publication.

```
starwars %>%
    filter(species=='Human') %>%
    tabyl(gender, eye_color) %>%
```

Gender/Eye Color	blue	blue-gray	brown	dark	hazel	yellow	Total
feminine	33% (3)	0% (0)	56% (5)	0% (0)	11% (1)	0% (0)	100% (9)
masculine	35% (9)	4% (1)	46% (12)	4% (1)	4% (1)	8% (2)	100% (26)
Total	34% (12)	3% (1)	49% (17)	3% (1)	6% (2)	6% (2)	100% (35)

```
# now we add count totals, row conditional percents and specify rounding

adorn_totals(c("row", "col")) %>%
adorn_percentages("row") %>%
adorn_pct_formatting(rounding = "half up", digits = 0) %>%

# we can even add sample sizes in parentheses

adorn_ns() %>%

# Here we add a title to the upper left corner of the table
adorn_title("combined",
            row_name = "Gender",
            col_name = "Eye Color") %>%

# lastly we add aesthetics that render the table publication-worthy

kable %>% kable_classic(full_width = FALSE)
```