Summary Statistics*

Chapter 8

Anna Ziff

R Workflow for Economists

Contents

*Please contact anna.ziff@duke.edu if there are errors.

This chapter covers statistics useful to understand and describe data that are already pre-processed. At the end of this chapter is an interactive exercise to write a function to output a table of descriptive statistics.

Here are all the libraries you should install for this chapter.

```
library(dplyr)
library(ggplot2)
library(purrr)
library(readr)
library(stargazer)
library(stringr)
library(tidyr)
```

Built-in Functions

We will use the dataset `gapminder_large.csv` which contains measures of development, environment, and society from the countries in the world. GDP and the Gini Index are measured in 2015. The Corruption Perception Index (`cpi`) is measured between 2012 and 2017. A higher score means less corruption. Life expectancy (`lifeexp`) is measured between 2012 and 2018. It is measured in years and is the average number of years a newborn would live holding constant contemporaneous mortality patterns. C02 emissions (`co2`) is measured between 2015 and 2018. The units are metric tonnes of CO2 per person.

```
df <- read.csv("gapminder_large.csv")
```

First, we want to get a sense of the data. How many observations are there? How many variables? What are the names of the variables and what classes are they?

```
head(df) # Display the first 6 rows
```

```
##                country gdp_2015 gini_2015               region co2_2015 co2_2016
## 1          Afghanistan      574      36.8      Asia & Pacific    0.262    0.245
## 2              Albania     4520      29.0              Europe    1.600    1.570
## 3              Algeria     4780      27.6         Arab States    3.800    3.640
## 4              Andorra    42100      40.0              Europe    5.970    6.070
## 5               Angola     3750      42.6              Africa    1.220    1.180
## 6  Antigua and Barbuda    13300      40.0 South/Latin America    5.840    5.900
##   co2_2017 co2_2018 cpi_2012 cpi_2013 cpi_2014 cpi_2015 cpi_2016 cpi_2017
## 1    0.247    0.254        8        8       12       11       15       15
## 2    1.610    1.590       33       31       33       36       39       38
## 3    3.560    3.690       34       36       36       36       34       33
## 4    6.270    6.120       NA       NA       NA       NA       NA       NA
## 5    1.140    1.120       22       23       19       15       18       19
## 6    5.890    5.880       NA       NA       NA       NA       NA       NA
##   lifeexp_2012 lifeexp_2013 lifeexp_2014 lifeexp_2015 lifeexp_2016 lifeexp_2017
## 1         60.8         61.3         61.2         61.2         61.2         63.4
## 2         77.8         77.9         77.9         78.0         78.1         78.2
## 3         76.8         76.9         77.0         77.1         77.4         77.7
## 4         82.4         82.5         82.5         82.6         82.7         82.7
## 5         61.3         61.9         62.8         63.3         63.8         64.2
## 6         76.7         76.8         76.8         76.9         77.0         77.0
##   lifeexp_2018
## 1         63.7
## 2         78.3
## 3         77.9
## 4           NA
## 5         64.6
```

```
## 6         77.2
```

```
dim(df) # Confirm the number of rows and columns
```

```
## [1] 195  21
```

```
names(df) # List the variable names
```

```
##  [1] "country"      "gdp_2015"     "gini_2015"    "region"       "co2_2015"
##  [6] "co2_2016"     "co2_2017"     "co2_2018"     "cpi_2012"     "cpi_2013"
## [11] "cpi_2014"     "cpi_2015"     "cpi_2016"     "cpi_2017"     "lifeexp_2012"
## [16] "lifeexp_2013" "lifeexp_2014" "lifeexp_2015" "lifeexp_2016" "lifeexp_2017"
## [21] "lifeexp_2018"
```

```
sapply(df, typeof)
```

```
##      country     gdp_2015    gini_2015       region     co2_2015     co2_2016
##  "character"    "integer"     "double"  "character"     "double"     "double"
##     co2_2017     co2_2018     cpi_2012     cpi_2013     cpi_2014     cpi_2015
##     "double"     "double"    "integer"    "integer"    "integer"    "integer"
##     cpi_2016     cpi_2017 lifeexp_2012 lifeexp_2013 lifeexp_2014 lifeexp_2015
##    "integer"    "integer"     "double"     "double"     "double"     "double"
## lifeexp_2016 lifeexp_2017 lifeexp_2018
##     "double"     "double"     "double"
```

The function `mean()` calculates the arithmetic mean. Here is a simple demonstration of it with a vector of 50 draws from the N(0,1) distribution.

```
x <- rnorm(50, mean = 0, sd = 1)
mean(x)
```

```
## [1] -0.02938889
```

If there are any elements of the input that are `NA`, you must specify the argument `na.rm = TRUE`. Otherwise, the result will be `NA`.

```
mean(df$gdp_2015)
```

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

```
mean(df$gdp_2015, na.rm = TRUE)
```

```
## [1] 14298.43
```

If you will only be using one data frame and do not want to repeatedly call variables using the format above, you can attach the data and then refer just to the variable name.

```
attach(df)
mean(gdp_2015, na.rm = TRUE)
```

```
## [1] 14298.43
```

While this is convenient, it is not always clear to which data frame the variable belongs. Also, if any variables have the same names as functions, those functions will be masked. This chapter thus relies on `data$colname` format for clarity. Let us detach the dataset and continue.

```
detach(df)
```

To calculate the mean for more than one column, we can use apply-like functions. Here, we are calculating the mean of every column except `country` and `region`, which are string variables.

```r
sapply(df[, -c(1, 4)], mean, na.rm = TRUE)
```

```
##     gdp_2015    gini_2015     co2_2015     co2_2016     co2_2017     co2_2018 
## 14298.427807    38.932821     4.456147     4.423509     4.446485     4.455041 
##     cpi_2012     cpi_2013     cpi_2014     cpi_2015     cpi_2016     cpi_2017 
##    42.906977    42.306358    42.929825    42.339394    42.687861    42.790960 
## lifeexp_2012 lifeexp_2013 lifeexp_2014 lifeexp_2015 lifeexp_2016 lifeexp_2017 
##    71.309091    71.642781    71.867380    72.144385    72.448128    72.737433 
## lifeexp_2018 
##    72.969022
```

The median is calculated with a function that is very similar to the mean function.

```r
median(x)
```

```
## [1] 0.1714187
```

```r
median(df$gini_2015, na.rm = TRUE)
```

```
## [1] 39.1
```

The function `quantile()` allows you to calculate other percentiles. Without specifying the probabilities in the `probs` argument, the function automatically outputs the minimum and maximum values, and the 25th, 50th, and 75th percentiles.

```r
quantile(x)
```

```
##         0%        25%        50%        75%       100% 
## -2.5396155 -0.7656675  0.1714187  0.7085117  2.1604500
```

```r
quantile(df$lifeexp_2015, probs = c(0.10, 0.90), na.rm = TRUE)
```

```
##  10%  90% 
## 61.3 81.5
```

Here are some functions to calculate measures of dispersion. Note the importance of specifying `na.rm = TRUE`.

```r
min(df$co2_2015, na.rm = TRUE)
```

```
## [1] 0.0367
```

```r
max(df$co2_2015, na.rm = TRUE)
```

```
## [1] 41.3
```

```r
range(df$co2_2015, na.rm = TRUE)
```

```
## [1]  0.0367 41.3000
```

```r
IQR(df$co2_2015, na.rm = TRUE)
```

```
## [1] 5.18525
```

```r
var(df$co2_2015, na.rm = TRUE) # Unbiased estimator
```

```
## [1] 33.79678
```

```r
sd(df$co2_2015, na.rm = TRUE)
```

```
## [1] 5.813499
```

The function `summary()` is a fast way to calculate many summary statistics at once. There is no need to add the `na.rm = TRUE` argument, and the function actually counts the number of `NA` values, if there are any.

```
summary(x)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.53962 -0.76567  0.17142 -0.02939  0.70851  2.16045
```

```
summary(df$cpi_2015)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    8.00   28.00   37.00   42.34   54.00   91.00      30
```

The covariance and correlation coefficient are calculated using `cov()` and `corr()`. Specifying what to do with `NA` values is a little more complicated for these functions. The argument `use` determines the strategy more precisely. If `use = "pairwise.complete.obs"`, then the covariance/correlation is only calculated for observations with two non-missing values. .

```
cov(df$gdp_2015, df$co2_2015)
```

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

```
cov(df$gdp_2015, df$co2_2015, use = "pairwise.complete.obs")
```

```
## [1] 65913.78
```

```
cor(df$gdp_2015, df$co2_2015, use = "pairwise.complete.obs")
```

```
## [1] 0.6068677
```

Data frames can be input into these functions, producing pairwise correlations.

```
cov(df[, c(2, 3, 5)], use = "pairwise.complete.obs")
```

```
##                 gdp_2015      gini_2015     co2_2015
## gdp_2015  510161355.07 -42624.157967 65913.780200
## gini_2015    -42624.16     54.232732    -7.687662
## co2_2015      65913.78     -7.687662    33.796776
```

```
cor(df[, c(2, 3, 5)], use = "pairwise.complete.obs")
```

```
##             gdp_2015  gini_2015   co2_2015
## gdp_2015   1.0000000 -0.2532801  0.6068677
## gini_2015 -0.2532801  1.0000000 -0.1782023
## co2_2015   0.6068677 -0.1782023  1.0000000
```

The function `t.test()` performs a t-test. The arguments augment the details of the test, including the null and alternative hypotheses.

```
t.test(df$lifeexp_2012, mu = 72, alternative = "two.sided")
```

```
##
##  One Sample t-test
##
## data:  df$lifeexp_2012
## t = -1.1733, df = 186, p-value = 0.2422
## alternative hypothesis: true mean is not equal to 72
## 95 percent confidence interval:
##  70.14742 72.47076
## sample estimates:
## mean of x
```

```
## 71.30909
```

```
t.test(df$lifeexp_2012, df$lifeexp_2017, paired = TRUE, var.equal = FALSE, conf.level = 0.90)
```

```
##
##  Paired t-test
##
## data:  df$lifeexp_2012 and df$lifeexp_2017
## t = -12.386, df = 186, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 90 percent confidence interval:
##  -1.618977 -1.237707
## sample estimates:
## mean of the differences
##               -1.428342
```

The function `ks.test()` performs the Kolmogorov-Smirnov Test to compare two distributions

```
ks.test(df[df$region == "Africa", "co2_2015"],
        df[df$region == "Middle east", "co2_2015"],
        alternative = "two.sided")
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  df[df$region == "Africa", "co2_2015"] and df[df$region == "Middle east", "co2_2015"]
## D = 0.7803, p-value = 2.778e-06
## alternative hypothesis: two-sided
```

Practice Exercises 8.1

1. Save the below code to an object. What is the data structure of this object? How can you extract information from this object?

```
t.test(df$lifeexp_2012, mu = 72, alternative = "two.sided")
```

`tidyverse` Functions

The advantages of `dplyr` functions and pipes are especially clear for producing summary statistics. We read in the data as a tibble.

```
tib <- read_csv("gapminder_large.csv")
```

```
## Rows: 195 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): country, region
## dbl (19): gdp_2015, gini_2015, co2_2015, co2_2016, co2_2017, co2_2018, cpi_2...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The function `summarise()` allows for many types of summary statistics. The output is itself a tibble. Here are examples naming the column of the output. Note that we need to specify `na.rm = TRUE`.

```
tib %>%
  summarise("Mean GDP 2015" = mean(gdp_2015, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   `Mean GDP 2015`
##             <dbl>
## 1          14298.
```

```
tib %>%
  summarise(MeanGDP2015 = mean(gdp_2015, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   MeanGDP2015
##         <dbl>
## 1      14298.
```

It is fine to refrain from naming the column. R automatically assigns the name based on the statistic.

```
tib %>%
  summarise(mean(gdp_2015, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   `mean(gdp_2015, na.rm = TRUE)`
##                            <dbl>
## 1                         14298.
```

It is possible to calculate many statistics at once.

```
tib %>%
  summarise(Median = median(gdp_2015, na.rm = TRUE),
            Variance = var(gdp_2015, na.rm =TRUE),
            SD = sd(gdp_2015, na.rm = TRUE),
            Minimum = min(gdp_2015, na.rm = TRUE),
            Maximum = max(gdp_2015, na.rm = TRUE),
            N = n())
```

```
## # A tibble: 1 x 6
##   Median   Variance     SD Minimum Maximum     N
##    <dbl>      <dbl>  <dbl>   <dbl>   <dbl> <int>
## 1   5740 510161355. 22587.     228  190000   195
```

The function `across()` can be used inside `summarise()` and `mutate()`. In the first argument, specify the vector of column names or indices. In the second argument, specify the function(s) to apply. The format comes from the package `purrr` and allows you to specify the values of the other arguments of the function.

```
tib %>%
  summarise(across(c(2, 3, 5), ~ mean(.x, na.rm = TRUE)))
```

```
## # A tibble: 1 x 3
##   gdp_2015 gini_2015 co2_2015
##      <dbl>     <dbl>    <dbl>
## 1   14298.      38.9     4.46
```

```
tib %>%
  summarise(across(starts_with("co2"), ~ median(.x, na.rm = TRUE)),
            across(starts_with("lifeexp"), ~ median(.x, na.rm = TRUE)))
```

```
## # A tibble: 1 x 11
##   co2_2015 co2_2016 co2_2017 co2_2018 lifeexp_2012 lifeexp_2013 lifeexp_2014
##      <dbl>    <dbl>    <dbl>    <dbl>        <dbl>        <dbl>        <dbl>
## 1     2.48     2.48     2.50     2.53         73.2         73.1         73.1
## # ... with 4 more variables: lifeexp_2015 <dbl>, lifeexp_2016 <dbl>,
```

```
## #   lifeexp_2017 <dbl>, lifeexp_2018 <dbl>
```

```
tib %>%
  summarise(across(c(2, 3, 5), list(mean = ~ mean(.x, na.rm = TRUE),
                                    median = ~ median(.x, na.rm = TRUE))))
```

```
## # A tibble: 1 x 6
##   gdp_2015_mean gdp_2015_median gini_2015_mean gini_2015_median co2_2015_mean
##           <dbl>           <dbl>          <dbl>            <dbl>         <dbl>
## 1        14298.            5740           38.9             39.1          4.46
## # ... with 1 more variable: co2_2015_median <dbl>
```

Now that we are comfortable with `summarise()`, let's add layers using the pipe operator. Adding `group_by()` beforehand allows for this.

```
tib %>%
  group_by(region) %>%
  summarise(MeanGINI = mean(gini_2015, na.rm = TRUE),
            N = n(),
            N_NA = sum(is.na(gini_2015)))
```

```
## # A tibble: 7 x 4
##   region              MeanGINI     N  N_NA
##   <chr>                  <dbl> <int> <int>
## 1 Africa                  43.8    44     0
## 2 Arab States             38.3    10     0
## 3 Asia & Pacific          36.7    45     0
## 4 Europe                  32.8    49     0
## 5 Middle east             36.8    12     0
## 6 North America           36.5     2     0
## 7 South/Latin America     45.7    33     0
```

We can also filter to only focus on certain observations.

```
tib %>%
  filter(region %in% c("Africa", "Middle east")) %>%
  group_by(region) %>%
  summarise(Mean_Gini = mean(gini_2015),
            SD_Gini = sd(gini_2015))
```

```
## # A tibble: 2 x 3
##   region      Mean_Gini SD_Gini
##   <chr>           <dbl>   <dbl>
## 1 Africa           43.8    7.85
## 2 Middle east      36.8    4.09
```

The data itself can be transformed in the pipe operations. Here, we are creating a variable that is then summarized.

```
tib %>%
  mutate(gini_rescaled = gini_2015/100) %>%
  group_by(region) %>%
  summarise(InterQuartileRange = IQR(gini_rescaled))
```

```
## # A tibble: 7 x 2
##   region              InterQuartileRange
##   <chr>                            <dbl>
## 1 Africa                          0.0865
```

```
## 2 Arab States                      0.088
## 3 Asia & Pacific                   0.065
## 4 Europe                           0.0720
## 5 Middle east                      0.0677
## 6 North America                    0.0480
## 7 South/Latin America              0.067
```

Tables

Creating Tables with stargazer

The package `stargazer` provides a simple way to output summary statistics from data frames. The simplest way to use the `stargazer()` function is to input a data frame. By default, it will return the LaTeX code for a table with summary statistics for all numeric variables. The default statistics are the number of observations, the mean, the standard deviation, the minimum, the 25th percentile, the 75th percentile, and the maximum.

```
head(df)
```

```
##               country gdp_2015 gini_2015              region co2_2015 co2_2016
## 1         Afghanistan      574      36.8      Asia & Pacific    0.262    0.245
## 2             Albania     4520      29.0              Europe    1.600    1.570
## 3             Algeria     4780      27.6         Arab States    3.800    3.640
## 4             Andorra    42100      40.0              Europe    5.970    6.070
## 5              Angola     3750      42.6              Africa    1.220    1.180
## 6 Antigua and Barbuda    13300      40.0 South/Latin America    5.840    5.900
##   co2_2017 co2_2018 cpi_2012 cpi_2013 cpi_2014 cpi_2015 cpi_2016 cpi_2017
## 1    0.247    0.254        8        8       12       11       15       15
## 2    1.610    1.590       33       31       33       36       39       38
## 3    3.560    3.690       34       36       36       36       34       33
## 4    6.270    6.120       NA       NA       NA       NA       NA       NA
## 5    1.140    1.120       22       23       19       15       18       19
## 6    5.890    5.880       NA       NA       NA       NA       NA       NA
##   lifeexp_2012 lifeexp_2013 lifeexp_2014 lifeexp_2015 lifeexp_2016 lifeexp_2017
## 1         60.8         61.3         61.2         61.2         61.2         63.4
## 2         77.8         77.9         77.9         78.0         78.1         78.2
## 3         76.8         76.9         77.0         77.1         77.4         77.7
## 4         82.4         82.5         82.5         82.6         82.7         82.7
## 5         61.3         61.9         62.8         63.3         63.8         64.2
## 6         76.7         76.8         76.8         76.9         77.0         77.0
##   lifeexp_2018
## 1         63.7
## 2         78.3
## 3         77.9
## 4           NA
## 5         64.6
## 6         77.2
```

```
stargazer(df)
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:17:28
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
```

```
## \begin{tabular}{@{\extracolsep{5pt}}lccccccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St. Dev.} & \multic
## \hline \\[-1.8ex]
## gdp\_2015 & 187 & 14,298.430 & 22,586.750 & 228.000 & 1,720.000 & 15,050.000 & 190,000.000 \\
## gini\_2015 & 195 & 38.933 & 7.364 & 24.800 & 33.450 & 42.700 & 63.100 \\
## co2\_2015 & 192 & 4.456 & 5.813 & 0.037 & 0.665 & 5.850 & 41.300 \\
## co2\_2016 & 192 & 4.424 & 5.644 & 0.025 & 0.681 & 6.048 & 38.500 \\
## co2\_2017 & 192 & 4.446 & 5.652 & 0.024 & 0.670 & 5.935 & 39.800 \\
## co2\_2018 & 192 & 4.455 & 5.609 & 0.024 & 0.669 & 5.925 & 38.000 \\
## cpi\_2012 & 172 & 42.907 & 19.614 & 8.000 & 28.000 & 55.000 & 90.000 \\
## cpi\_2013 & 173 & 42.306 & 19.881 & 8.000 & 28.000 & 54.000 & 91.000 \\
## cpi\_2014 & 171 & 42.930 & 19.811 & 8.000 & 28.500 & 55.000 & 92.000 \\
## cpi\_2015 & 165 & 42.339 & 20.150 & 8.000 & 28.000 & 54.000 & 91.000 \\
## cpi\_2016 & 173 & 42.688 & 19.375 & 10.000 & 29.000 & 56.000 & 90.000 \\
## cpi\_2017 & 177 & 42.791 & 18.978 & 9.000 & 29.000 & 56.000 & 89.000 \\
## lifeexp\_2012 & 187 & 71.309 & 8.052 & 48.900 & 65.000 & 77.450 & 83.600 \\
## lifeexp\_2013 & 187 & 71.643 & 7.882 & 48.500 & 65.450 & 77.600 & 83.900 \\
## lifeexp\_2014 & 187 & 71.867 & 7.752 & 48.700 & 65.950 & 77.750 & 84.200 \\
## lifeexp\_2015 & 187 & 72.144 & 7.497 & 50.500 & 66.950 & 77.850 & 84.400 \\
## lifeexp\_2016 & 187 & 72.448 & 7.296 & 51.700 & 67.300 & 78.050 & 84.700 \\
## lifeexp\_2017 & 187 & 72.737 & 7.070 & 51.900 & 67.800 & 78.150 & 84.800 \\
## lifeexp\_2018 & 184 & 72.969 & 6.968 & 52.400 & 68.100 & 78.325 & 85.000 \\
## \hline \\[-1.8ex]
## \end{tabular}
## \end{table}
```

Inputting a selected set of variables will restrict the table.

```
stargazer(df[, 5:8])
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:17:28
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lccccccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St. Dev.} & \multic
## \hline \\[-1.8ex]
## co2\_2015 & 192 & 4.456 & 5.813 & 0.037 & 0.665 & 5.850 & 41.300 \\
## co2\_2016 & 192 & 4.424 & 5.644 & 0.025 & 0.681 & 6.048 & 38.500 \\
## co2\_2017 & 192 & 4.446 & 5.652 & 0.024 & 0.670 & 5.935 & 39.800 \\
## co2\_2018 & 192 & 4.455 & 5.609 & 0.024 & 0.669 & 5.925 & 38.000 \\
## \hline \\[-1.8ex]
## \end{tabular}
## \end{table}
```

There are many options to alter the output. Here are a few examples. See `?stargazer` and this document for more examples.

```
stargazer(df[, 5:8], title = "CO2 Emissions")
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:17:28
## \begin{table}[!htbp] \centering
##   \caption{CO2 Emissions}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lccccccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St. Dev.} & \multic
## \hline \\[-1.8ex]
## co2\_2015 & 192 & 4.456 & 5.813 & 0.037 & 0.665 & 5.850 & 41.300 \\
## co2\_2016 & 192 & 4.424 & 5.644 & 0.025 & 0.681 & 6.048 & 38.500 \\
## co2\_2017 & 192 & 4.446 & 5.652 & 0.024 & 0.670 & 5.935 & 39.800 \\
## co2\_2018 & 192 & 4.455 & 5.609 & 0.024 & 0.669 & 5.925 & 38.000 \\
## \hline \\[-1.8ex]
## \end{tabular}
## \end{table}
```

```
stargazer(df[, 5:8], float = FALSE)
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:17:28
## \begin{tabular}{@{\extracolsep{5pt}}lccccccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St. Dev.} & \multic
## \hline \\[-1.8ex]
## co2\_2015 & 192 & 4.456 & 5.813 & 0.037 & 0.665 & 5.850 & 41.300 \\
## co2\_2016 & 192 & 4.424 & 5.644 & 0.025 & 0.681 & 6.048 & 38.500 \\
## co2\_2017 & 192 & 4.446 & 5.652 & 0.024 & 0.670 & 5.935 & 39.800 \\
## co2\_2018 & 192 & 4.455 & 5.609 & 0.024 & 0.669 & 5.925 & 38.000 \\
## \hline \\[-1.8ex]
## \end{tabular}
```

```
stargazer(df[, 5:8], out = "table1.tex")
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:17:28
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lccccccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St. Dev.} & \multic
## \hline \\[-1.8ex]
## co2\_2015 & 192 & 4.456 & 5.813 & 0.037 & 0.665 & 5.850 & 41.300 \\
## co2\_2016 & 192 & 4.424 & 5.644 & 0.025 & 0.681 & 6.048 & 38.500 \\
## co2\_2017 & 192 & 4.446 & 5.652 & 0.024 & 0.670 & 5.935 & 39.800 \\
## co2\_2018 & 192 & 4.455 & 5.609 & 0.024 & 0.669 & 5.925 & 38.000 \\
## \hline \\[-1.8ex]
## \end{tabular}
```

```
## \end{table}
```

```r
stargazer(df[, 5:8], summary.stat = c("n", "mean", "sd"))
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:17:28
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St. Dev.} \\
## \hline \\[-1.8ex]
## co2\_2015 & 192 & 4.456 & 5.813 \\
## co2\_2016 & 192 & 4.424 & 5.644 \\
## co2\_2017 & 192 & 4.446 & 5.652 \\
## co2\_2018 & 192 & 4.455 & 5.609 \\
## \hline \\[-1.8ex]
## \end{tabular}
## \end{table}
```

```r
stargazer(df[, 5:8], flip = TRUE)
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Tue, Aug 02, 2022 - 15:17:28
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lcccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & co2\_2015 & co2\_2016 & co2\_2017 & co2\_2018 \\
## \hline \\[-1.8ex]
## N & 192 & 192 & 192 & 192 \\
## Mean & 4.456 & 4.424 & 4.446 & 4.455 \\
## St. Dev. & 5.813 & 5.644 & 5.652 & 5.609 \\
## Min & 0.037 & 0.025 & 0.024 & 0.024 \\
## Pctl(25) & 0.665 & 0.681 & 0.670 & 0.669 \\
## Pctl(75) & 5.850 & 6.048 & 5.935 & 5.925 \\
## Max & 41.300 & 38.500 & 39.800 & 38.000 \\
## \hline \\[-1.8ex]
## \end{tabular}
## \end{table}
```

Creating Tables from Scratch

These exercises will take you through creating a function that outputs a table of summary statistics for inclusion in a LaTeX document.

1. This function will take a tibble of summary statistics as an input. Create a tibble that lists, for each region, the mean, standard deviation, minimum, maximum, and number of non-missing observations for 2015 life expectancy. Save this tibble to an object so you can access it.

```
gapminder <- read_csv("gapminder_large.csv")
out <- gapminder %>%
  group_by(region) %>%
  summarise(mean(lifeexp_2015, na.rm = TRUE),
            sd(lifeexp_2015, na.rm = TRUE),
            min(lifeexp_2015, na.rm = TRUE),
            max(lifeexp_2015, na.rm = TRUE),
            sum(!is.na(lifeexp_2015)))
```

2. Define the name of your function. The first argument will be a tibble like the one you produced in question 1.

```
create_table <- function(tib) {

  # Open file connection
  # Define header lines
  # Define body lines
  # Define footer lines
  # Write header, body, and footer lines
  # Close file connection

}
```

3. We want to write an output to a LaTeX file. This will require a file connection. That is, you will open a file with a certain file name, write lines from the tibble of question 1, and close the file. The second argument will be the filename. A file connection is opened and closed with the following commands.

```
connection <- file(filename) # Open a file
close(connection)
````
Add these to your function and include an argument for the filename.
```

```
create_table <- function(tib, filename) {

  # Open file connection
  connection <- file(filename)

  # Define header lines
  # Define body lines
  # Define footer lines
  # Write header, body, and footer lines

  # Close file connection
  close(connection)

}
```

4. Tables in LaTeX require a header and footer to open and close the tabular environment. Start by defining the footer as this is simplest. Add the following object to your function in between opening and closing the file connection. Why do we use two backslashes instead of one?

```
create_table <- function(tib, filename) {
  # Open file connection
  connection <- file(filename)

  # Define header lines
```
```

```
  # Define body lines

  # Define footer lines
  foot <- c("\\bottomrule", "\\end{tabular}")

  # Write header, body, and footer lines

  # Close file connection
  close(connection)

}
```

5. To actually write a line in the file, we will use the function `writeLines()`. Add this to your function. Start by writing the footer to the file. At this point, test the function out to see how the `writeLines()` function works.

```
create_table <- function(tib, filename) {
  # Open file connection
  connection <- file(filename)

  # Define header lines
  # Define body lines

  # Define footer lines
  foot <- c("\\bottomrule", "\\end{tabular}")

  # Write header, body, and footer lines
  writeLines(foot, connection)

  # Close file connection
  close(connection)

}
create_table(out, "test.tex")
```

6. Now let's define the header. We need to begin the tabular environment, the title columns, and the alignment of the columns. The names of the columns will be the third argument. Let's start with all centrally aligned columns. Add the header to the `writeLines()` function. We have a tibble with 1 column for the region and 5 columns for summary statistics. What does your file look like now? Make adjustments if there are some oddities.

```
create_table <- function(tib, filename, colnames) {
  # Open file connection
  connection <- file(filename)

  # Define header lines
  head <- c(paste0("\\begin{tabular}{", str_dup("c", dim(tib)[2]), "}"),
            "\\toprule",
            paste(str_c(colnames, collapse = " & "), "\\\\"),
            "\\midrule")

  # Define body lines

  # Define footer lines
  foot <- c("\\bottomrule", "\\end{tabular}")
```

```
  # Write header, body, and footer lines
  writeLines(c(head, foot), connection)

  # Close file connection
  close(connection)

}
create_table(out, "test.tex", c("Region", "Mean", "SD", "Min.", "Max.", "N"))
```

7. Finally, loop through each row in the tibble to print each line.

```
create_table <- function(tib, filename, colnames) {
  # Open file connection
  connection <- file(filename)

  # Define header lines
  head <- c(paste0("\\begin{tabular}{", str_dup("c", dim(tib)[2]), "}"),
            "\\toprule",
            paste(str_c(colnames, collapse = " & "), "\\\\"),
            "\\midrule")

  # Define body lines
  for (i in 1:dim(tib)[1]) {

    if (i == 1) {
      body <- paste(str_c(tib[i, ], collapse = " & "), "\\\\")
    } else {
      body <- c(body, paste(str_c(tib[i, ], collapse = " & "), "\\\\"))
    }

  }

  # Define footer lines
  foot <- c("\\bottomrule", "\\end{tabular}")

  # Write header, body, and footer lines
  writeLines(c(head, foot), connection)

  # Close file connection
  close(connection)

}
create_table(out, "test.tex", c("Region", "Mean", "SD", "Min.", "Max.", "N"))
```

Further Reading

Reference the dplyr cheat sheet. Higher-order moments are available in the `moments` package.