Summarizing data with R

This chapter will introduce you to how to summarize data using R, as well as providing an introduction to a popular set of R tools known as the "Tidyverse."

Before doing anything else we need to load the libraries that we will use in this notebook.

```
library(tidyverse)
```

```
## -- Attaching packages ---- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1       v purrr   0.2.5
## v tibble  2.0.0       v dplyr   0.8.0.1
## v tidyr   0.8.2       v stringr 1.3.1
## v readr   1.3.1       v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
## Warning: package 'tibble' was built under R version 3.5.2
```

```
## Warning: package 'dplyr' was built under R version 3.5.2
```

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

```
library(cowplot)
```

```
## Warning: package 'cowplot' was built under R version 3.5.2
```

```
##
## Attaching package: 'cowplot'
```

```
## The following object is masked from 'package:ggplot2':
##
##     ggsave
```

```
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.5.2
```

We will use the NHANES dataset for several of our examples, so let's load the library that contains the data.

```
# load the NHANES data library
library(NHANES)
```

Introduction to the Tidyverse

In this chapter we will introduce a way of working with data in R that is often referred to as the "Tidyverse."

Pipes

You will also notice something we haven't discussed before: %>%. This is called a "pipe", which is commonly used within the tidyverse (which we will discuss more in the R lab on data wrangling); you can read more here. A pipe takes the output from one command and feeds it as input to the next command. In this case, simply writing the name of the data frame (myDataFrame) causes it to be input to the slice() command following the pipe. The benefit of pipes will become especially apparent when we want to start stringing together multiple functions into a single command.

To see pipes in action, let's clean up the NHANES dataset. Each individual in the NHANES dataset has a unique identifier stored in the variable `ID`. First let's look at the number of rows in the dataset:

```
nrow(NHANES)
```

```
## [1] 10000
```

Now let's see how many unique IDs there are. The `unique()` function returns a vector containing all of the unique values for a particular variable, and the `length()` function returns the length of the resulting vector.

```
length(unique(NHANES$ID))
```

```
## [1] 6779
```

This shows us that while there are 10,000 observations in the data frame, there are only 6779 unique IDs. This means that if we were to use the entire dataset, we would be reusing data from some individuals, which could give us incorrect results. For this reason, we wold like to discard any observations that are duplicated.

Let's create a new variable called `NHANES_unique` that will contain only the distinct observations, with no individuals appearing more than once. The `dplyr` library provides a function called `distinct()` that will do this for us. You may notice that we didn't explicitly load the `dplyr` library above; however, if you look at the messages that appeared when we loaded the `tidyverse` library, you will see that it loaded `dplyr` for us. To create the new data frame with unique observations, we will pipe the NHANES data frame into the `distinct()` function and then save the output to our new variable.

```
NHANES_unique <-
  NHANES %>%
  distinct(ID, .keep_all = TRUE)
```

If we number of rows in the new data frame, it should be the same as the number of unique IDs (6779):

```
nrow(NHANES_unique)
```

```
## [1] 6779
```

Computing a frequency distribution

We would like to compute a frequency distribution

```
PhysActive_table <- NHANES %>%
  dplyr::select(PhysActive) %>% # select the variable
  group_by(PhysActive) %>% # group by values of the variable
  summarize(AbsoluteFrequency = n()) # count the values
```

```
## Warning: Factor `PhysActive` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```