5.1: 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." This comprises a set of packages that provide various tools for working with data, as well as a few special ways of using those functions

5.1.1 Making a data frame using tibble()

The tidyverse provides its over version of a data frame, which known as a *tibble*. A tibble is a data frame but with some smart tweaks that make it easier to work with, expecially when using functions from the tidyverse. See here for more information on the function `tibble()` : https://r4ds.had.co.nz/tibbles.html

```
# first create the individual variables
n <- c("russ", "lucy", "jaclyn", "tyler")
x <- c(1, 2, 3, 4)
y <- c(4, 5, 6, 7)
z <- c(7, 8, 9, 10)

# create the data frame
myDataFrame <-
  tibble(
    n, #list each of your columns in the order you want them
    x,
    y,
    z
  )

myDataFrame
```

```
## # A tibble: 4 x 4
##   n          x     y     z
##   <chr>  <dbl> <dbl> <dbl>
## 1 russ       1     4     7
## 2 lucy       2     5     8
## 3 jaclyn     3     6     9
## 4 tyler      4     7    10
```

Take a quick look at the properties of the data frame using `glimpse()` :

```
glimpse(myDataFrame)
```

```
## Observations: 4
## Variables: 4
## $ n <chr> "russ", "lucy", "jaclyn", "tyler"
## $ x <dbl> 1, 2, 3, 4
## $ y <dbl> 4, 5, 6, 7
## $ z <dbl> 7, 8, 9, 10
```

5.1.2 Selecting an element

There are various ways to access the contents within a data frame.

5.1.2.1 Selecting a row or column by name

```
myDataFrame$x
```

```
## [1] 1 2 3 4
```

The first index refers to the row, the second to the column.

```
myDataFrame[1, 2]
```

```
## # A tibble: 1 x 1
##       x
##   <dbl>
## 1     1
```

```
myDataFrame[2, 3]
```

```
## # A tibble: 1 x 1
##       y
##   <dbl>
## 1     5
```

5.1.2.2 Selecting a row or column by index

```
myDataFrame[1, ]
```

```
## # A tibble: 1 x 4
##   n         x     y     z
##   <chr> <dbl> <dbl> <dbl>
## 1 russ      1     4     7
```

```
myDataFrame[, 1]
```

```
## # A tibble: 4 x 1
##   n
##   <chr>
## 1 russ
## 2 lucy
## 3 jaclyn
## 4 tyler
```

5.1.2.3 Select a set of rows

```
myDataFrame %>%
  slice(1:2)
```

```
## # A tibble: 2 x 4
##   n         x     y     z
##   <chr> <dbl> <dbl> <dbl>
## 1 russ      1     4     7
## 2 lucy      2     5     8
```

`slice()` is a function that selects out rows based on their row number.

You will also notice something we haven't discussed before: `%>%` . This is called a "pipe", which is commonly used within the tidyverse; 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 data processing operations into a single command.

In this example, no new variable is created - the output is printed to the screen, just like it would be if you typed the name of the variable. If you wanted to save it to a new variable, you would use the `<-` assignment operator, like this:

```
myDataFrameSlice <- myDataFrame %>%
  slice(1:2)

myDataFrameSlice
```

```
## # A tibble: 2 x 4
##   n         x     y     z
##   <chr> <dbl> <dbl> <dbl>
## 1 russ      1     4     7
## 2 lucy      2     5     8
```

5.1.2.4 Select a set of rows based on specific value(s)

```
myDataFrame %>%
  filter(n == "russ")
```

```
## # A tibble: 1 x 4
##   n         x     y     z
##   <chr> <dbl> <dbl> <dbl>
## 1 russ      1     4     7
```

`filter()` is a function that retains only those rows that meet your stated criteria. We can also filter for multiple criteria at once — in this example, the `|` symbol indicates "or":

```
myDataFrame %>%
  filter(n == "russ" | n == "lucy")
```

```
## # A tibble: 2 x 4
##   n         x     y     z
##   <chr> <dbl> <dbl> <dbl>
## 1 russ      1     4     7
## 2 lucy      2     5     8
```

5.1.2.5 Select a set of columns

```
myDataFrame %>%
  select(x:y)
```

```
## # A tibble: 4 x 2
##       x     y
##   <dbl> <dbl>
## 1     1     4
## 2     2     5
## 3     3     6
## 4     4     7
```

`select()` is a function that selects out only those columns you specify using their names

You can also specify a vector of columns to select.

```
myDataFrame %>%
  select(c(x,z))
```

```
## # A tibble: 4 x 2
##       x     z
##   <dbl> <dbl>
## 1     1     7
## 2     2     8
## 3     3     9
## 4     4    10
```

5.1.3 Adding a row or column

add a named row

```
tiffanyDataFrame <-
  tibble(
    n = "tiffany",
    x = 13,
    y = 14,
    z = 15
  )

myDataFrame %>%
  bind_rows(tiffanyDataFrame)
```

```
## # A tibble: 5 x 4
##   n           x     y     z
##   <chr>   <dbl> <dbl> <dbl>
## 1 russ        1     4     7
## 2 lucy        2     5     8
## 3 jaclyn      3     6     9
## 4 tyler       4     7    10
## 5 tiffany    13    14    15
```

`bind_rows()` is a function that combines the rows from another dataframe to the current dataframe