Lesson 2: Create your own data frame

Background for this activity

This activity is focused on creating and using data frames in `R`. A data frame is a collection of columns containing data, similar to a spreadsheet or SQL table. Data frames are one of the basic tools you will use to work with data in `R`. And you can create data frames from different data sources.

There are three common sources for data:

- A `package` with data that can be accessed by loading that `package`
- An external file like a spreadsheet or CSV that can be imported into `R`
- Data that has been generated from scratch using `R` code

Wherever data comes from, you will almost always want to store it in a data frame object to work with it. Now, you can start creating and exploring data frames with the code chunks in the RMD space. To interact with the code chunk, click the green arrow in the top-right corner of the chunk. The executed code will appear in the RMD space and your console.

Throughout this activity, you will also have the opportunity to practice writing your own code by making changes to the code chunks yourself. If you encounter an error or get stuck, you can always check the Lesson2_Dataframe_Solutions .rmd file in the Solutions folder under Week 3 for the complete, correct code.

Step 1: Load packages

Start by installing the required package; in this case, you will want to install `tidyverse`. If you have already installed and loaded `tidyverse` in this session, feel free to skip the code chunks in this step.

```
install.packages("tidyverse")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.3'
## (as 'lib' is unspecified)
```

Once a package is installed, you can load it by running the `library()` function with the package name inside the parentheses:

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

Step 2: Create data frame

Sometimes you will need to generate a data frame directly in `R`. There are a number of ways to do this; one of the most common is to create individual vectors of data and then combine them into a data frame using the `data.frame()` function.

Here's how this works. First, create a vector of names by inserting four names into this code block between the quotation marks and then run it:

```
names <- c("Marco", "Pedro", "María", "Mencho")
```

Then create a vector of ages by adding four ages separated by commas to the code chunk below. Make sure you are inputting numeric values for the ages or you might get an error.

```
age <- c(10L,20L ,40L ,50L )
```

With these two vectors, you can create a new data frame called `people`:

```
people <- data.frame(names, age)
```

Step 3: inspect the data frame

Now that you have this data frame, you can use some different functions to inspect it.

One common function you can use to preview the data is the `head()` function, which returns the columns and the first several rows of data. You can check out how the `head()` function works by running the chunk below:

```
head(people)
```

```
##    names age
## 1  Marco  10
## 2  Pedro  20
## 3  María  40
## 4 Mencho  50
```

In addition to `head()`, there are a number of other useful functions to summarize or preview your data. For example, the `str()` and `glimpse()` functions will both provide summaries of each column in your data arranged horizontally. You can check out these two functions in action by running the code chunks below:

```
str(people)
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ names: chr  "Marco" "Pedro" "María" "Mencho"
##  $ age  : int  10 20 40 50
```

```
glimpse(people)
```

```
## Rows: 4
## Columns: 2
## $ names <chr> "Marco", "Pedro", "María", "Mencho"
## $ age   <int> 10, 20, 40, 50
```

You can also use `colnames()` to get a list the column names in your data set. Run the code chunk below to check out this function:

```
colnames(people)
```

```
## [1] "names" "age"
```

Now that you have a data frame, you can work with it using all of the tools in `R`. For example, you could use `mutate()` if you wanted to create a new variable that would capture each person's age in twenty years. The code chunk below creates that new variable:

```
mutate(people, age_in_20 = age + 20)
```

```
##    names age age_in_20
## 1  Marco  10        30
## 2  Pedro  20        40
## 3  María  40        60
## 4 Mencho  50        70
```

Step 4: Try it yourself

To get more familiar with creating and using data frames, use the code chunks below to create your own custom data frame.

First, create a vector of any five different fruits. You can type directly into the code chunk below; just place your cursor in the box and click to type. Once you have input the fruits you want in your data frame, run the code chunk.

```
comidas=c("ceviche","pizza","shukos")
```

Now, create a new vector with a number representing your own personal rank for each fruit. Give a 1 to the fruit you like the most, and a 5 to the fruit you like the least. Remember, the scores need to be in the same order as the fruit above. So if your favorite fruit is last in the list above, the score `1` needs to be in the last position in the list below. Once you have input your rankings, run the code chunk.

```
precios=c(30,55,10)
```

Finally, combine the two vectors into a data frame. You can call it `fruit_ranks`. Edit the code chunk below and run it to create your data frame.

```
menu=data.frame(precios,comidas)
```

After you run this code chunk, it will create a data frame with your fruits and rankings.

Activity Wrap Up

In this activity, you learned how to create data frames, view them with summary functions like `head()` and `glimpse()`, and then made changes with the `mutate()` function. You can continue practicing these skills by modifying the code chunks in the rmd file, or use this code as a starting point in your own project console. As you explore data frames, consider how they are similar and different to the tables you have worked with in other data analysis tools like spreadsheets and SQL. Data frames are one of the most basic building blocks you will need to work with data in `R`. So understanding how to create and work with data frames is an important first step to analyzing data.

Make sure to mark this activity as complete in Coursera.