Lesson 2: Importing and working with data

Background for this activity / Introduction

By now, you have some experience manually entering data in R to create a data frame. As a data analyst, it will also be common for you to import data from external files into your R console and use it to create a data frame to analyze it. In this activity you will learn how to import data from outside of R using the `read_csv()` function. Then, once you have imported a data file, you will use R functions to manipulate and interact with that data.

You can start importing and exploring data 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_Import_Solutions .rmd file in the Solutions folder under Week 3 for the complete, correct code.

The scenario

In this scenario, you are a junior data analyst working for a hotel booking company. You have been asked to clean a .csv file that was created after querying a database to combine two different tables from different hotels. In order to learn more about this data, you are going to need to use functions to preview the data's structure, including its columns and rows. You will also need to use basic cleaning functions to prepare this data for analysis.

Step 1: Load packages

Start by installing your required package. 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, we 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 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Step 2: Import data

One of the most common file types data analysts import into R is comma separated values files, or .csv files. The tidyverse library package readr has a number of functions for "reading in" or importing data, including .csv files and other external sources.

In the chunk below, use the read_csv() function to import data from a .csv in the project folder called "hotel_bookings.csv" and save it as a data frame called bookings_df.

If this line causes an error, copy in the line setwd("/cloud/project/Course 7/Week 3") before it.

The results will display as column specifications:

```
bookings_df <- read_csv("hotel_bookings.csv")
```

```
## Rows: 119390 Columns: 32
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (13): hotel, arrival_date_month, meal, country, market_segment, distrib...
## dbl  (18): is_canceled, lead_time, arrival_date_year, arrival_date_week_numb...
## date  (1): reservation_status_date
##
## 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.
```

Now that you have the bookings_df, you can work with it using all of the R functions you have learned so far.

Step 3: Inspect & clean data

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

```
head(bookings_df)
```

```
## # A tibble: 6 x 32
##   hotel        is_canceled lead_time arrival_date_year arrival_date_month
##   <chr>              <dbl>     <dbl>             <dbl> <chr>
## 1 Resort Hotel           0       342              2015 July
## 2 Resort Hotel           0       737              2015 July
## 3 Resort Hotel           0         7              2015 July
## 4 Resort Hotel           0        13              2015 July
## 5 Resort Hotel           0        14              2015 July
## 6 Resort Hotel           0        14              2015 July
## # i 27 more variables: arrival_date_week_number <dbl>,
## #   arrival_date_day_of_month <dbl>, stays_in_weekend_nights <dbl>,
## #   stays_in_week_nights <dbl>, adults <dbl>, children <dbl>, babies <dbl>,
## #   meal <chr>, country <chr>, market_segment <chr>,
## #   distribution_channel <chr>, is_repeated_guest <dbl>,
## #   previous_cancellations <dbl>, previous_bookings_not_canceled <dbl>,
## #   reserved_room_type <chr>, assigned_room_type <chr>, ...
```

In addition to head() there are a number of other useful functions to summarize or preview your data frame. For example, the str() and function will provide summaries of each column in your data arranged horizontally. Check out the str() function by running the code chunk below:

```
str(bookings_df)
```

```
## spc_tbl_ [119,390 x 32] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
```

```
##  $ hotel                         : chr [1:119390] "Resort Hotel" "Resort Hotel" "Resort Hotel" "Reso
##  $ is_canceled                   : num [1:119390] 0 0 0 0 0 0 0 0 1 1 ...
##  $ lead_time                     : num [1:119390] 342 737 7 13 14 14 0 9 85 75 ...
##  $ arrival_date_year             : num [1:119390] 2015 2015 2015 2015 2015 ...
##  $ arrival_date_month            : chr [1:119390] "July" "July" "July" "July" ...
##  $ arrival_date_week_number      : num [1:119390] 27 27 27 27 27 27 27 27 27 27 ...
##  $ arrival_date_day_of_month     : num [1:119390] 1 1 1 1 1 1 1 1 1 1 ...
##  $ stays_in_weekend_nights       : num [1:119390] 0 0 0 0 0 0 0 0 0 0 ...
##  $ stays_in_week_nights          : num [1:119390] 0 0 1 1 2 2 2 2 3 3 ...
##  $ adults                        : num [1:119390] 2 2 1 1 2 2 2 2 2 2 ...
##  $ children                      : num [1:119390] 0 0 0 0 0 0 0 0 0 0 ...
##  $ babies                        : num [1:119390] 0 0 0 0 0 0 0 0 0 0 ...
##  $ meal                          : chr [1:119390] "BB" "BB" "BB" "BB" ...
##  $ country                       : chr [1:119390] "PRT" "PRT" "GBR" "GBR" ...
##  $ market_segment                : chr [1:119390] "Direct" "Direct" "Direct" "Corporate" ...
##  $ distribution_channel          : chr [1:119390] "Direct" "Direct" "Direct" "Corporate" ...
##  $ is_repeated_guest             : num [1:119390] 0 0 0 0 0 0 0 0 0 0 ...
##  $ previous_cancellations        : num [1:119390] 0 0 0 0 0 0 0 0 0 0 ...
##  $ previous_bookings_not_canceled: num [1:119390] 0 0 0 0 0 0 0 0 0 0 ...
##  $ reserved_room_type            : chr [1:119390] "C" "C" "A" "A" ...
##  $ assigned_room_type            : chr [1:119390] "C" "C" "C" "A" ...
##  $ booking_changes               : num [1:119390] 3 4 0 0 0 0 0 0 0 0 ...
##  $ deposit_type                  : chr [1:119390] "No Deposit" "No Deposit" "No Deposit" "No Deposit
##  $ agent                         : chr [1:119390] "NULL" "NULL" "NULL" "304" ...
##  $ company                       : chr [1:119390] "NULL" "NULL" "NULL" "NULL" ...
##  $ days_in_waiting_list          : num [1:119390] 0 0 0 0 0 0 0 0 0 0 ...
##  $ customer_type                 : chr [1:119390] "Transient" "Transient" "Transient" "Transient" ..
##  $ adr                           : num [1:119390] 0 0 75 75 98 ...
##  $ required_car_parking_spaces   : num [1:119390] 0 0 0 0 0 0 0 0 0 0 ...
##  $ total_of_special_requests     : num [1:119390] 0 0 0 0 1 1 0 1 1 0 ...
##  $ reservation_status            : chr [1:119390] "Check-Out" "Check-Out" "Check-Out" "Check-Out" ..
##  $ reservation_status_date       : Date[1:119390], format: "2015-07-01" "2015-07-01" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   hotel = col_character(),
##   ..   is_canceled = col_double(),
##   ..   lead_time = col_double(),
##   ..   arrival_date_year = col_double(),
##   ..   arrival_date_month = col_character(),
##   ..   arrival_date_week_number = col_double(),
##   ..   arrival_date_day_of_month = col_double(),
##   ..   stays_in_weekend_nights = col_double(),
##   ..   stays_in_week_nights = col_double(),
##   ..   adults = col_double(),
##   ..   children = col_double(),
##   ..   babies = col_double(),
##   ..   meal = col_character(),
##   ..   country = col_character(),
##   ..   market_segment = col_character(),
##   ..   distribution_channel = col_character(),
##   ..   is_repeated_guest = col_double(),
##   ..   previous_cancellations = col_double(),
##   ..   previous_bookings_not_canceled = col_double(),
##   ..   reserved_room_type = col_character(),
```

```
##   ..   assigned_room_type = col_character(),
##   ..   booking_changes = col_double(),
##   ..   deposit_type = col_character(),
##   ..   agent = col_character(),
##   ..   company = col_character(),
##   ..   days_in_waiting_list = col_double(),
##   ..   customer_type = col_character(),
##   ..   adr = col_double(),
##   ..   required_car_parking_spaces = col_double(),
##   ..   total_of_special_requests = col_double(),
##   ..   reservation_status = col_character(),
##   ..   reservation_status_date = col_date(format = "")
##   .. )
##  - attr(*, "problems")=<externalptr>
```

To find out what columns you have in your data frame, try running the the `colnames()` function in the code chunk below:

```
colnames(bookings_df)
```

```
##  [1] "hotel"                          "is_canceled"
##  [3] "lead_time"                      "arrival_date_year"
##  [5] "arrival_date_month"             "arrival_date_week_number"
##  [7] "arrival_date_day_of_month"      "stays_in_weekend_nights"
##  [9] "stays_in_week_nights"           "adults"
## [11] "children"                       "babies"
## [13] "meal"                           "country"
## [15] "market_segment"                 "distribution_channel"
## [17] "is_repeated_guest"              "previous_cancellations"
## [19] "previous_bookings_not_canceled" "reserved_room_type"
## [21] "assigned_room_type"             "booking_changes"
## [23] "deposit_type"                   "agent"
## [25] "company"                        "days_in_waiting_list"
## [27] "customer_type"                  "adr"
## [29] "required_car_parking_spaces"    "total_of_special_requests"
## [31] "reservation_status"             "reservation_status_date"
```

If you want to create another data frame using `bookings_df` that focuses on the average daily rate, which is referred to as `adr` in the data frame, and `adults`, you can use the following code chunk to do that:

```
new_df <- select(bookings_df, `adr`, adults)
```

To create new variables in your data frame, you can use the `mutate()` function. This will make changes to the data frame, but not to the original data set you imported. That source data will remain unchanged.

```
mutate(new_df, total = `adr` / adults)
```

```
## # A tibble: 119,390 x 3
##      adr adults total
##    <dbl>  <dbl> <dbl>
##  1    0       2   0
##  2    0       2   0
##  3   75       1  75
##  4   75       1  75
##  5   98       2  49
##  6   98       2  49
##  7  107       2  53.5
```

```
##  8  103       2  51.5
##  9   82       2  41
## 10  106.      2  52.8
## # i 119,380 more rows
```

Step 4: Import your own data

Now you can find your own .csv to import! Using the RStudio Cloud interface, import and save the file in the same folder as this R Markdown document. To do this, go to the Files tab in the lower-right console. Then, click the Upload button next to the + New Folder button. This will open a popup to let you browse your computer for a file. Select any .csv file, then click Open. Now, write code in the chunk below to read that data into `R`:

You can check the solutions document for this activity to check your work.

Activity Wrap Up

Now that you know how to import data using the `read_csv()` function, you will be able to work with data that has been stored externally right in your `R` console. You can continue to practice 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 become more familiar with the process of importing data, consider how importing data from a .csv file changed the way you accessed and interacted with the data. Did you do anything differently? Being able to import data from external sources will allow you to work with even more data, giving you even more options for analyzing data in `R`.

Make sure to mark this activity as complete in Coursera.