Lesson 3: Changing your data

Background for this activity

In this activity, you'll review a scenario, and focus on manipulating and changing real data in R. You will learn more about functions you can use to manipulate your data, use statistical summaries to explore your data, and gain initial insights for your stakeholders.

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 Lesson3_Change_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. You have already performed some basic cleaning functions on this data; this activity will focus on using functions to conduct basic data manipulation.

Step 1: Load packages

Start by installing the required packages. If you have already installed and loaded `tidyverse`, `skimr`, and `janitor` in this session, feel free to skip the code chunks in this step. This may take a few minutes to run, and you may get a pop-up window asking if you want to proceed. Click yes to continue installing the packages.

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

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

```
install.packages("skimr")
```

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

```
install.packages("janitor")
```

```
## 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 --
## 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
```

```
library(skimr)
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

Step 2: Import data

In the chunk below, you will 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 `hotel_bookings`. Type "hotel_bookings.csv" between the quotation marks in the following code chunk.

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

```
hotel_bookings <- 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.
```

Step 3: Getting to know your data

Like you have been doing in other examples, you are going to use summary functions to get to know your data. This time, you are going to complete the code chunks below in order to use these different functions. You can use the `head()` function to preview the columns and the first several rows of data. Finish the code chunk below and run it:

```
head(hotel_bookings)
```

```
## # 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>, ...
```

Now you know this dataset contains information on hotel bookings. Each booking is a row in the dataset, and each column contains information such as what type of hotel was booked, when the booking took place, and how far in advance the booking took place (the 'lead_time' column).

In addition to `head()` you can also use the `str()` and `glimpse()` functions to get summaries of each column in your data arranged horizontally. You can try these two functions by completing and running the code chunks below:

```
str(hotel_bookings)
```

```
## 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>
```

You can see the different column names and some sample values to the right of the colon.

```
glimpse(hotel_bookings)
```

```
## Rows: 119,390
## Columns: 32
## $ hotel                         <chr> "Resort Hotel", "Resort Hotel", "Resort~
## $ is_canceled                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, ~
## $ lead_time                     <dbl> 342, 737, 7, 13, 14, 14, 0, 9, 85, 75, ~
## $ arrival_date_year             <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 201~
## $ arrival_date_month            <chr> "July", "July", "July", "July", "July",~
## $ arrival_date_week_number      <dbl> 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,~
## $ arrival_date_day_of_month     <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ stays_in_weekend_nights       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ stays_in_week_nights          <dbl> 0, 0, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 4, ~
## $ adults                        <dbl> 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ children                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ babies                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ meal                          <chr> "BB", "BB", "BB", "BB", "BB", "BB", "BB~
## $ country                       <chr> "PRT", "PRT", "GBR", "GBR", "GBR", "GBR~
## $ market_segment                <chr> "Direct", "Direct", "Direct", "Corporat~
## $ distribution_channel          <chr> "Direct", "Direct", "Direct", "Corporat~
## $ is_repeated_guest             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ previous_cancellations        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ previous_bookings_not_canceled <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ reserved_room_type            <chr> "C", "C", "A", "A", "A", "A", "C", "C",~
## $ assigned_room_type            <chr> "C", "C", "C", "A", "A", "A", "C", "C",~
## $ booking_changes               <dbl> 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ deposit_type                  <chr> "No Deposit", "No Deposit", "No Deposit~
```

```
## $ agent                          <chr> "NULL", "NULL", "NULL", "304", "240", "~
## $ company                        <chr> "NULL", "NULL", "NULL", "NULL", "NULL",~
## $ days_in_waiting_list           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ customer_type                  <chr> "Transient", "Transient", "Transient", ~
## $ adr                            <dbl> 0.00, 0.00, 75.00, 75.00, 98.00, 98.00,~
## $ required_car_parking_spaces    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
## $ total_of_special_requests      <dbl> 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 3, ~
## $ reservation_status             <chr> "Check-Out", "Check-Out", "Check-Out", ~
## $ reservation_status_date        <date> 2015-07-01, 2015-07-01, 2015-07-02, 20~
```

You can also use `colnames()` to get the names of the columns in your dataset. Run the code chunk below to get the column names:

```
colnames(hotel_bookings)
```

```
##  [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"
```

Manipulating your data

Let's say you want to arrange the data by most lead time to least lead time because you want to focus on bookings that were made far in advance. You decide you want to try using the `arrange()` function; input the correct column name after the comma and run this code chunk:

```
arrange(hotel_bookings, )
```

```
## # A tibble: 119,390 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
##  7 Resort Hotel           0         0              2015 July
##  8 Resort Hotel           0         9              2015 July
##  9 Resort Hotel           1        85              2015 July
## 10 Resort Hotel           1        75              2015 July
## # i 119,380 more rows
## # 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>, ...
```

But why are there so many zeroes? That's because `arrange()` automatically orders by ascending order, and you need to specifically tell it when to order by descending order, like the below code chunk below:

```
arrange(hotel_bookings, desc(lead_time))
```

```
## # A tibble: 119,390 x 32
##    hotel        is_canceled lead_time arrival_date_year arrival_date_month
##    <chr>              <dbl>     <dbl>             <dbl> <chr>
##  1 Resort Hotel           0       737              2015 July
##  2 Resort Hotel           0       709              2016 February
##  3 City Hotel             1       629              2017 March
##  4 City Hotel             1       629              2017 March
##  5 City Hotel             1       629              2017 March
##  6 City Hotel             1       629              2017 March
##  7 City Hotel             1       629              2017 March
##  8 City Hotel             1       629              2017 March
##  9 City Hotel             1       629              2017 March
## 10 City Hotel             1       629              2017 March
## # i 119,380 more rows
## # 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>, ...
```

Now it is in the order you needed. You can click on the different pages of results to see additional rows of data, too.

Notice that when you just run `arrange()` without saving your data to a new data frame, it does not alter the existing data frame. Check it out by running `head()` again to find out if the highest lead times are first:

```
head(hotel_bookings)
```

```
## # 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>, ...
```

This will be true of all the functions you will be using in this activity. If you wanted to create a new data

frame that had those changes saved, you would use the assignment operator, <- , as written in the code chunk below to store the arranged data in a data frame named 'hotel_bookings_v2':

```
hotel_bookings_v2 <-
  arrange(hotel_bookings, desc(lead_time))
```

Run `head()`to check it out:

```
head(hotel_bookings_v2)
```

```
## # 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       737              2015 July
## 2 Resort Hotel           0       709              2016 February
## 3 City Hotel             1       629              2017 March
## 4 City Hotel             1       629              2017 March
## 5 City Hotel             1       629              2017 March
## 6 City Hotel             1       629              2017 March
## # 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>, ...
```

You can also find out the maximum and minimum lead times without sorting the whole dataset using the `arrange()` function. Try it out using the max() and min() functions below:

```
max(hotel_bookings$lead_time)
```

```
## [1] 737
```

```
min(hotel_bookings$lead_time)
```

```
## [1] 0
```

Remember, in this case, you need to specify which dataset and which column using the $ symbol between their names. Try running the below to see what happens if you forget one of those pieces:

```
min(hotel_bookings$lead_time)
```

```
## [1] 0
```

This is a common error that R users encounter. To correct this code chunk, you will need to add the data frame and the dollar sign in the appropriate places.

Now, let's say you just want to know what the average lead time for booking is because your boss asks you how early you should run promotions for hotel rooms. You can use the `mean()` function to answer that question since the average of a set of number is also the mean of the set of numbers:

```
mean(hotel_bookings$lead_time)
```

```
## [1] 104.0114
```

You should get the same answer even if you use the v2 dataset that included the `arrange()` function. This is because the `arrange()` function doesn't change the values in the dataset; it just re-arranges them.

```
mean(hotel_bookings_v2$lead_time)
```

```
## [1] 104.0114
```

Practice Quiz

What is the average lead time? A: 14.0221 B: 45.0283 C: 100.0011 D: 104.0114

Remember to select the correct answer to this question on the Coursera platform after you complete this activity.

You were able to report to your boss what the average lead time before booking is, but now they want to know what the average lead time before booking is for just city hotels. They want to focus the promotion they're running by targeting major cities.

You know that your first step will be creating a new dataset that only contains data about city hotels. You can do that using the `filter()` function, and name your new data frame 'hotel_bookings_city':

```
hotel_bookings_city <-
  filter(hotel_bookings, hotel_bookings$hotel=="City Hotel")
```

Check out your new dataset:

```
head(hotel_bookings_city)
```

```
## # A tibble: 6 x 32
##   hotel      is_canceled lead_time arrival_date_year arrival_date_month
##   <chr>            <dbl>     <dbl>             <dbl> <chr>
## 1 City Hotel           0         6              2015 July
## 2 City Hotel           1        88              2015 July
## 3 City Hotel           1        65              2015 July
## 4 City Hotel           1        92              2015 July
## 5 City Hotel           1       100              2015 July
## 6 City Hotel           1        79              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>, ...
```

You quickly check what the average lead time for this set of hotels is, just like you did for all of hotels before:

```
mean(hotel_bookings_city$lead_time)
```

```
## [1] 109.7357
```

Now, your boss wants to know a lot more information about city hotels, including the maximum and minimum lead time. They are also interested in how they are different from resort hotels. You don't want to run each line of code over and over again, so you decide to use the `group_by()`and`summarize()` functions. You can also use the pipe operator to make your code easier to follow. You will store the new dataset in a data frame named 'hotel_summary':

```
hotel_summary <-
  hotel_bookings %>%
  group_by(hotel) %>%
  summarise(average_lead_time=mean(lead_time),
            min_lead_time=min(lead_time),
            max_lead_time=max(lead_time))
```

Check out your new dataset using head() again:

```r
head(hotel_summary)
```

```
## # A tibble: 2 x 4
##   hotel        average_lead_time min_lead_time max_lead_time
##   <chr>                    <dbl>         <dbl>         <dbl>
## 1 City Hotel               110.              0           629
## 2 Resort Hotel              92.7             0           737
```

Activity Wrap Up

Being able to manipulate data is a key skill for working in `R`. After this activity, you should be more familiar with functions that allow you to change your data, such as `arrange()`, `group_by()`, and `filter()`. You also have some experience using statistical summaries to make insights into your data. 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 practice, consider how performing tasks is similar and different in `R` compared to other tools you have learned throughout this program.