---

title: "Lesson 4: Annotations"

output: html\_document

---

## Annotating and saving solutions

This document contains the solutions for the annotating and saving activity. You can use these solutions to check your work and ensure that your code is correct or troubleshoot your code if it is returning errors. If you haven't completed the activity yet, we suggest you go back and finish it before reading the solutions.

If you experience errors, remember that you can search the internet and the RStudio community for help:

https://community.rstudio.com/#

## Step 1: Import your data

The data in this example is originally from the article Hotel Booking Demand Datasets (https://www.sciencedirect.com/science/article/pii/S2352340918315191), written by Nuno Antonio, Ana Almeida, and Luis Nunes for Data in Brief, Volume 22, February 2019.

The data was downloaded and cleaned by Thomas Mock and Antoine Bichat for #TidyTuesday during the week of February 11th, 2020 (https://github.com/rfordatascience/tidytuesday/blob/master/data/2020/2020-02-11/readme.md).

You can learn more about the dataset here:

https://www.kaggle.com/jessemostipak/hotel-booking-demand

Run the code below to read in the file 'hotel\_bookings.csv' into a data frame:

```{r}

hotel\_bookings <- read.csv("hotel\_bookings.csv")

```

## Step 2: Refresh Your Memory

By now, you are pretty familiar with this data set. But you can refresh your memory with the `head()` and `colnames()` functions. Run two code chunks below to get at a sample of the data and also preview all the column names:

```{r look at data}

head(hotel\_bookings)

```

```{r look at column names}

colnames(hotel\_bookings)

```

## Step 3: Install and load the 'ggplot2' package (optional)

If you haven't already installed and loaded the `ggplot2` package, you will need to do that before you can use the `ggplot()` function. You only have to do this once though, not every time you call `ggplot()`.

You can also skip this step if you haven't closed your RStudio account since doing the last activity. If you aren't sure, you can run the code chunk and hit 'cancel' if the warning message pops up telling you that have already downloaded the `ggplot2` package.

Run the code chunk below to install and load `ggplot2`. This may take a few minutes!

```{r loading and installing ggplot2, echo=FALSE, message=FALSE}

install.packages('ggplot2')

library(ggplot2)

```

## Step 4: Annotating your chart

As a refresher, here is the chart you created before:

```{r faceting a plot}

ggplot(data = hotel\_bookings) +

geom\_bar(mapping = aes(x = market\_segment)) +

facet\_wrap(~hotel)

```

The first step will be adding a title; that is often the first thing people will pay attention to when they encounter a data visualization for the first time. To add a title, you will add `labs()` at the end of your `ggplot()` command and then input a title there:

```{r faceting a plot with a title}

ggplot(data = hotel\_bookings) +

geom\_bar(mapping = aes(x = market\_segment)) +

facet\_wrap(~hotel) +

labs(title="Comparison of market segments by hotel type for hotel bookings")

```

You also want to add another detail about what time period this data covers. To do this, you need to find out when the data is from.

You realize you can use the `min()` function on the year column in the data:

```{r earliest year}

min(hotel\_bookings$arrival\_date\_year)

```

And the `max()` function:

```{r latest year}

max(hotel\_bookings$arrival\_date\_year)

```

But you will need to save them as variables in order to easily use them in your labeling; the following code chunk creates two of those variables:

```{r latest date}

mindate <- min(hotel\_bookings$arrival\_date\_year)

maxdate <- max(hotel\_bookings$arrival\_date\_year)

```

Now, you will add in a subtitle using `subtitle=` in the `labs()` function. Then, you can use the `paste0()` function to use your newly-created variables in your labels. This is really handy, because if the data gets updated and there is more recent data added, you don't have to change the code below because the variables are dynamic:

```{r city bar chart with timeframe}

ggplot(data = hotel\_bookings) +

geom\_bar(mapping = aes(x = market\_segment)) +

facet\_wrap(~hotel) +

labs(title="Comparison of market segments by hotel type for hotel bookings",

subtitle=paste0("Data from: ", mindate, " to ", maxdate))

```

You decide to switch the `subtitle` to a `caption` which will appear in the bottom right corner instead.

```{r city bar chart with timeframe as caption}

ggplot(data = hotel\_bookings) +

geom\_bar(mapping = aes(x = market\_segment)) +

facet\_wrap(~hotel) +

labs(title="Comparison of market segments by hotel type for hotel bookings",

caption=paste0("Data from: ", mindate, " to ", maxdate))

```

Now you want to clean up the x and y axis labels to make sure they are really clear. To do that, you can add to the `labs()` function and use `x=` and `y=`. Feel free to change the text of the label and play around with it:

```{r city bar chart with x and y axis}

ggplot(data = hotel\_bookings) +

geom\_bar(mapping = aes(x = market\_segment)) +

facet\_wrap(~hotel) +

labs(title="Comparison of market segments by hotel type for hotel bookings",

caption=paste0("Data from: ", mindate, " to ", maxdate),

x="Market Segment",

y="Number of Bookings")

```

## Step 5: Saving your chart

The `ggsave()` function was used to save the last plot that was generated, so if you have run something after running the code chunk above, then run that code chunk again.

Then run the following code chunk to save that plot as a .png file named `city\_payment\_chart`, which makes it clear to your stakeholders what the .png file contains. Now you should be able to find this file in your 'Files' tab in the bottom right of your screen. Check it out!

```{r save your plot}

ggsave('hotel\_booking\_chart.png')

```

## Practice quiz

What are the default dimensions that `ggsave()` saved your image as?

A: 5x5

B: 10x10

C: 7x7

D: 25x25

Answer: C. The default dimensions of this ggsave() image are 7x7. You can see these dimensions listed after you run the code chunk.

If you wanted to make your chart bigger and more rectangular to fit the slide show presentation, you could specify the height and width of your .png in the `ggsave()` command. Edit the code chunk below to create a 16x8 .png image:

```{r save your plot}

ggsave('hotel\_booking\_chart.png',

width=16,

height=8)

```