Exploratory Data Analysis with Tidyverse

Estimated Time Needed: **40 min**

Welcome!

In this section, we will explore several methods to see if certain characteristics or features can be used to predict arrival delay minutes. The main question we will be trying to answer throughout this lab is: what are the main characteristics which have the most impact on how long a flight arrives late?

Table of Contents

Load Libraries and Data

First, load the tidyverse library. This lab will use `ggplot` which is already loaded when `tidyverse` is loaded.

[ ]:
```
# Load tidyverse
library(tidyverse)
```

The original Airline dataset is hosted on IBM Data Asset eXchange. This sample dataset can be found here.

Now using the subset dataset link, you can load it and store as a dataframe `sub_airline`:

[ ]:
```
# url where the data is located
url <- "https://dax-cdn.cdn.appdomain.cloud/dax-airline/1.0.1/lax_to_jfk.tar.gz"

# download the file
download.file(url, destfile = "lax_to_jfk.tar.gz")

# untar the file so we can get the csv only
# if you run this on your local machine, then can remove tar = "internal"
untar("lax_to_jfk.tar.gz", tar = "internal")

# read_csv only
sub_airline <- read_csv("lax_to_jfk/lax_to_jfk.csv",
                        col_types = cols('DivDistance' = col_number(),
                                         'DivArrDelay' = col_number()))
```

1. Analyzing Individual Feature Patterns using Visualization

Boxplots are a great way to visualize numeric (or quantitative) data, since you can visualize the various distributions of the data. They are similar to histograms but can show more informatioin such as:

- The median of the data, which represents the middle datapoint
- The Upper Quartile, which is the 75th percentile
- The Lower Quartile, which is the 25th percentile
- The Interquartile Range, which is the data between the Upper and Lower Quartile

Boxplots also display outliers as individual dots that occur outside the upper and lower extremes. With boxplots, you can easily spot outliers and also see the distribution and skewness of the data.

You will use the `ggplot` library to create plots, which is already loaded when the `tidyverse` library is loaded.

To create boxplots, the main function to use is `geom_boxplot()`. In the below code we also use additional functions primarly to customize the aesthetics of the plot. You can check out the Data Visualization Course in R if you want to know more about customizing plots with `ggplot`.

- `geom_boxplot()`: The boxplot compactly displays the distribution of a continuous variable. It visualises five summary statistics (the median, two hinges and two whiskers), and all "outlying" points individually.
- `geom_jitter()`: The jitter geom is a convenient shortcut for geom_point(position = "jitter"). It adds a small amount of random variation to the location of each point, and is a useful way of handling overplotting caused by discreteness in smaller datasets.
- `ggtitle()`: Modifies plot titles (main title, axis labels and legend titles).
- `guides()`: Guides for each scale can be set scale-by-scale with the guide argument.
- `theme_minimal()`: A minimalistic theme with no background annotations.
- `coord_cartesian()`: The Cartesian coordinate system is the most familiar, and common, type of coordinate system. Setting limits on the coordinate system will zoom the plot (like you're looking at it with a magnifying glass), and will not change the underlying data like setting limits on a scale will.

Below shows the distribution of arrival delays for each reporting airline. So here, ArrDelay is the numerica data.

[ ]:
```
# Boxplot
ggplot(data = sub_airline, mapping = aes(x = Reporting_Airline, y = ArrDelay)) +
```

```
  geom_boxplot(fill = "bisque",color = "black", alpha = 0.3) +
  geom_jitter(aes(color = 'blue'), alpha=0.2) +
  labs(x = "Airline") +
  ggtitle("Arrival Delays by Airline") +
  guides(color = FALSE) +
  theme_minimal() +
  coord_cartesian(ylim = quantile(sub_airline$ArrDelay, c(0, 0.99)))
```

Often times we see continuous variables in our data. These data points are numbers contained in some range.

For example, in our dataset, departure delays and arrival delays are continuous numeric variables. What if we want to understand the relationship between "DepDelay" and "ArrDelay"? Could departure delay possibly predict arrival delay?

One good way to visualize two continuous variables is to use a scatter plot. Each observation in a scatter plot is represented as a point.

Using `ggplot`, you can input the two continuous variables to compare and add `gemo_point()` to show the points in a scatter plot.

[ ]:
```
# Load Alaska data, deleting rows that have missing departure delay or arrival delay data
alaska_flights <- sub_airline %>%
  filter(Reporting_Airline == "AS") %>%
  filter(!is.na(DepDelay) & !is.na(ArrDelay)) %>%
  filter(DepDelay < 40)

ggplot(data = alaska_flights, mapping = aes(x = DepDelay, y = ArrDelay)) +
   geom_point() +
   ggtitle("Alaska Flight Depature Delays vs Arrival Delays")
```

How to choose the right visualization method?

When visualizing individual variables, it is important to first understand what type of variable you are dealing with. This will help us find the right visualization method for that variable. Below is a way to show each variable and their type in a dataframe.

[ ]:
```
# list the data types for each column
str(sub_airline)
```

Now, we turn our focus to "ArrDelayMinutes" as we want to create models to predict this variable in lab 4.

Question #1:

What is the data type of the column "ArrDelayMinutes"?

[ ]:
```
# Write your code below and press Shift+Enter to execute
```

▼Click here for the solution.

```
numerical (num)
```

Next, let's take a look at other variables, like "DepDelayMinutes" that could potentially help predict "ArrDelayMinutes".

Question #2:

Find the correlation between the following columns: DepDelayMinutes and ArrDelayMinutes.

Hint: if you would like to select those columns, use the dollar sign ($)

[ ]:
```
# Write your code below and press Shift+Enter to execute
```

▼Click here for the solution.

```
cor(sub_airline@@@@@ArrDelayMinutes)
```

Continuous numerical variables:

Continuous numerical variables are variables that may contain any value within some range. In R, continuous numerical variables can have the type "integer" or "numeric" (these are real numbers and sometimes are called "float" in other programming languages). A great way to visualize these variables is by using scatterplots with fitted lines.

With `ggplot`, we can visualize this by using `geom_point()` to plot the data points and `geom_smooth()` to plot a fitted linear regression line (by default the model uses `formula = y ~ x`).

Let's see several examples of different linear relationships:

Positive linear relationship

Let's find the scatterplot of "DepDelayMinutes" and "ArrDelayMinutes" of all airlines.

[ ]:
```
# DepDelayMinutes as potential predictor variable of ArrDelayMinutes
ggplot(data = sub_airline, mapping = aes(x = DepDelayMinutes, y = ArrDelayMinutes)) +
    geom_point() +
    geom_smooth(method = "lm", na.rm = TRUE)
```

From the plot, as the depature delay ("DepDelayMinutes") increases, the arrival delay ("ArrDelayMinutes") increases. This indicates a positive direct correlation between these two variables. "DepDelayMinutes" may be a decent predictor of "ArrDelayMinutes" since the regression line is increasing and generally matches the data points.

Next, we can examine the correlation between "DepDelayMinutes" and "ArrDelayMinutes" and see it's approximately 0.92

[ ]:
```
cor(sub_airline$DepDelayMinutes, sub_airline$ArrDelayMinutes)
```

Weak Linear Relationship

Let's now look at if "WeatherDelay" is a good predictor variable of "ArrDelayMinutes".

[ ]:
```
ggplot(data = sub_airline, mapping = aes(x = WeatherDelay, y = ArrDelayMinutes)) +
    geom_point() +
    geom_smooth(method = "lm", na.rm = TRUE)
```

Weather delay does not seem like a good predictor of arrival delay minutes since the regression line is close to horizontal. Also, for small values of "WeatherDelay", the data points are very

scattered and far from the fitted line, showing lots of variability. Therefore it is not a reliable variable.

In the `cor()` function, you can add `use = "complete.obs"` in order to only use complete observations, that is, exclude NAs. You can examine the correlation between "WeatherDelay" and "ArrDelayMinutes" and see that it's a very weak relationship since the value is close to 0.

[ ]:
```
cor(sub_airline$WeatherDelay, sub_airline$ArrDelayMinutes, use = "complete.obs")
```

Question 3 a):

Find the correlation between x="CarrierDelay", y="ArrDelayMinutes".

Hint: if you would like to select those columns, use the dollar sign ($)

[ ]:
```
# Write your code below and press Shift+Enter to execute

```

▼Click here for the solution.
```
# The correlation is 0.728.
# code
cor(sub_airline@@@@@ArrDelayMinutes,
    use = "complete.obs")
```

Question 3 b):

Given the correlation results between x="CarrierDelay", y="ArrDelayMinutes", do you expect a linear relationship?

Verify your results using the function of ggplot.

[ ]:
```
# Write your code below and press Shift+Enter to execute

```

▼Click here for the solution.
```
# There is a decent correlation between the variable "CarrierDelay" and "ArrDelayMinutes" since the correlation is somewhat close to 1. We can further demonstrate this
using "geom_point" and "geom_smooth".

# Code
ggplot(data = sub_airline, mapping = aes(x = CarrierDelay, y = ArrDelayMinutes)) +
    geom_point() +
    geom_smooth(method = "lm", na.rm = TRUE)
```

2. Descriptive Statistical Analysis

When you begin to analyze data, it's important to first explore your data before you spend time building complicated models. One easy way to do so is to calculate some descriptive statistics for your data.

Descriptive statistical analysis helps to describe basic features of a dataset and generates a short summary about the sample and measures of the data.

Let's take a look at a couple of different useful methods. One way in which we can do this is by using the `summarize()` function in `tidyverse:dplyr()`. We introduced this method in previous modules. Method `group_by()` is often used together with `summarize()`, which summarizes each group into a single-row summary of that group. `group_by()` takes as arguments the column names that contain the categorical variables for which you want to calculate the summary statistics.

This will show:

- the count of that variable
- the mean
- the standard deviation (std)
- the minimum value
- the median (50th percentile or quartile 2)
- the IQR (Interquartile Range: quartile 3 minus quartile 1)
- the maximum value

We can apply the method "summarize" and "group_by" as follows:

[ ]:
```
summary_airline_delays <- sub_airline %>%
  group_by(Reporting_Airline) %>%
  summarize(count = n(),
            mean = mean(ArrDelayMinutes, na.rm = TRUE),
            std_dev = sd(ArrDelayMinutes, na.rm = TRUE),
            min = min(ArrDelayMinutes, na.rm = TRUE),
            median = median(ArrDelayMinutes, na.rm=TRUE),
            iqr = IQR(ArrDelayMinutes, na.rm = TRUE),
            max = max(ArrDelayMinutes, na.rm = TRUE))

summary_airline_delays
```

To identify the data type of each column in a dataframe, we can use `sapply()` with `typeof`, which finds the type of something. So here, you *apply* `typeof` to every column in sub_airline.

[ ]:
```
sapply(sub_airline, typeof)
```

Value Counts

One way you can summarize the categorical data is by using the function `count()`. As an example, you can get the count of each reporting airline in this dataset. We see that we have 1096 flights from AA - American Airlines, 45 flights from AS - Alaska Airlines, 258 from B6 - jetBlue airlines, etc.

[ ]:
```
sub_airline %>%
  count(Reporting_Airline)
```

3. Basics of Grouping

We often ask questions like: Is there any relationship between the reporting airline and the flight delays? If so, which day of the week do flights have relatively longer delay times? For example, people take flights most frequently on Monday and Friday for business trips. Would that impact how long the flight is delayed? It would be nice if we could group the data by the different reporting airline, and compare the results of these different day of week against each other.

In tidyverse this can be done using the "group_by" method. The group by method is used on categorical variables, it groups the data into subsets according to the different categories of that variable. You can group by a single variable or you can group by multiple variables by passing in multiple variable names.

Let's say we are interested in finding the average delay minutes of flights and observe how they differ between different "Reporting_Airline" and "DayOfWeeks" variables.

To do this, you can use `group_by()` to group by "Reporting_Airline" and "DayOfWeeks", then use `summarize()` to calculate the mean delay minutes. Setting `.groups = 'keep'` keeps the grouping structure, if you instead were to set it to `"drop"` then the output would be ungrouped and essentially be a regualr tibble (dataframe).

[ ]:
```
avg_delays <- sub_airline %>%
  group_by(Reporting_Airline, DayOfWeek) %>%
  summarize(mean_delays = mean(ArrDelayMinutes), .groups = 'keep')

head(avg_delays)
```

The function `arrange()` will reorder the rows in an *ascending* order. However, using `arrange()` with `desc()` sorts data in *descending* order.

So now, we sort the mean_delays column by descending order using `arrange(desc())` and print the output table to see the top airlines and day of the week pairs that have the highest average arrival delays.

[ ]:
```
# sort the dataframe in R using multiple variables with Dplyr
sorted <- avg_delays %>%
    arrange(desc(mean_delays))

head(sorted)
```

To make it easier to understand, we can transform this table to a **heatmap**.

A heatmap has one variable displayed along the x-axis and the other variable displayed along the y-axis. A heat map (or heatmap) is a data visualization technique that shows magnitude of a phenomenon as color in two dimensions.

The variation in color may be by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varied over space. It is a great way to plot the target variable over multiple variables and through this get visual clues of the relationship between these variables and the target.

With `ggplot`, you can use `geom_tile()` to create heatmaps and use `scale_fill_gradient()` to change the coloring of the heatmap. So to interpret the heatmap below, the closer a tile is to the color red, the higher the mean arrival delay for that particualr day of the week and airline pair.

[ ]:
```
# The color is still hard to see and identify,  let's change the color
avg_delays %>%
  ggplot(aes(x = Reporting_Airline,
             y = DayOfWeek,
             fill = mean_delays)) +
  # set the tile's borders to be white with size 0.2
  geom_tile(color = "white", size = 0.2) +
  # define gradient color scales
  scale_fill_gradient(low = "yellow", high = "red")
```

BONUS Heatmap

For something more sophisticated, you can always add more blocks in your code.

[ ]:
```
# This visualization will use lubridate package
library(lubridate)
# Let's take a simple average across Reporting_Airline and DayOfWeek
avg_delays <- sub_airline %>%
  group_by(Reporting_Airline, DayOfWeek) %>%
  summarize(mean_delays = mean(ArrDelayMinutes), .groups = 'keep') %>%
  # create a new variable "bins" from mean_delays
  # make the first range -0.1 to 0.1 to include zero values
  mutate(bins = cut(mean_delays,breaks = c(-0.1,0.1,10,20,30,50, max(mean_delays)),
                    labels = c("0","0-10","10-20","20-30","30-50",">50"))) %>%
  mutate(bins = factor(as.character(bins),levels = rev(levels(bins))))


ggplot(avg_delays, aes(x = Reporting_Airline,
                       y = lubridate::wday(DayOfWeek, label = TRUE),
                       fill = bins)) +
   geom_tile(colour = "white", size = 0.2) +
   geom_text(aes(label = round(mean_delays, 3))) +
   guides(fill = guide_legend(title = "Delays Time Scale"))+
   labs(x = "Reporting Airline",y = "Day of Week",title = "Average Arrival Delays")+
   # Define color palette for the scale
   scale_fill_manual(values = c("#d53e4f", "#f46d43", "#fdae61", "#fee08b", "#e6f598", "#abdda4"))
```

Question 4:

Use the "groupby" and "summarize" function to find the average "ArrDelayMinutes" of each flight based on "Reporting_Airline" ?

[ ]:
```
# Write your code below and press Shift+Enter to execute
```

▼ Click here for the solution.

```
# grouping results
sub_airline %>%
  group_by(Reporting_Airline) %>%
  summarize(mean_delays = mean(ArrDelayMinutes))
```

Visualization is very important in data science and R visualization packages, namely `ggplot`, provide great freedom. We will go more in-depth in a separate R Data Visualization course.

4. Correlation and Causation

The main question we are trying to answer in this module is: "What causes flight delays?". To get a better measure of the important characteristics, we look at the correlation of these variables with the arrival delay, AKA, "ArrDelayMinutes", in other words: how is the arrival delay minutes dependent on this variable?

First, let's begin with some important definitions:

- **Correlation**: a measure of the extent of interdependence between variables.
- **Causation**: the relationship between cause and effect between two variables.

It is important to know the difference between these two and that correlation does not imply causation. Determining correlation is much simpler than determining causation as causation may require independent experimentation.

Pearson Correlation

The Pearson Correlation measures the linear dependence between two variables X and Y.

The resulting coefficient is a value between -1 and 1 inclusive, where:

- 1: Total positive linear correlation.
- 0: No linear correlation, the two variables most likely do not affect each other.
- -1: Total negative linear correlation.

Pearson Correlation is the default method of the function `cor()`, but other methods can be specified by setting `method`. Like before we can calculate the Pearson Correlation of the "integer" or "numeric" variables.

[ ]:
```
sub_airline %>%
  select(DepDelayMinutes, ArrDelayMinutes) %>%
  cor(method = "pearson")
```

Sometimes, we would like to know the significant of the correlation estimate.

P-value:

What is this P-value? The P-value is the probability value that the correlation between these two variables is statistically significant. Normally, we choose a significance level of 0.05, which means that we are 95% confident that the correlation between the variables is significant.

By convention, when the

- p-value is < 0.001: we say there is strong evidence that the correlation is significant.
- the p-value is < 0.05: there is moderate evidence that the correlation is significant.
- the p-value is < 0.1: there is weak evidence that the correlation is significant.
- the p-value is > 0.1: there is no evidence that the correlation is significant.

In R, to conduct a significance test and get the p-values, you can use `cor.test()`.

[ ]:
```
sub_airline %>%
  cor.test(~DepDelayMinutes + ArrDelayMinutes, data = .)
```

Conclusion:

See that the P-value is very small, much smaller than .001. And so we can conclude that we are certain about the strong positive correlation.

Correlations between multiple variables

To calculate correlations on multiple variables, you can `select` all the variable then use the `cor()` function but set `use = "pairwise.complete.obs"` so that the correlation of every pair of variables is calculated. This computes a matrix of Pearson's r correlation coefficients for all possible pairs of columns.

[ ]:
```
correlation <- sub_airline %>%
  select(ArrDelayMinutes, DepDelayMinutes,
         CarrierDelay, WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay) %>%
  cor(use = "pairwise.complete.obs", method = "pearson")

correlation
```

Taking all variables into account, we can now create a heatmap that visualizes the correlation between each of the variables with one another.

Let's use the `corrplot()` function to plot an elegant graph of a correlation matrix, you may need to install the library `corrplot` first before loading it. The color scheme indicates the Pearson correlation coefficient, indicating the strength of the correlation between two variables. We can see a diagonal line with a dark blue color, indicating that all the values on this diagonal are highly correlated. This makes sense because when you look closer, the values on the diagonal are the correlation of all variables with themselves, which will always be 1.

This correlation heatmap gives us a good overview of how the different variables are related to one another and, most importantly, how these variables are related to arrival delays.

[ ]:
```
# Download the corrplot library first if you have not already.
# install.packages("corrplot")
```

[ ]:
```
library(corrplot)

numerics_airline <- sub_airline %>%
  select(ArrDelayMinutes, DepDelayMinutes, CarrierDelay,
         WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay)

airlines_cor <- cor(numerics_airline, method = "pearson", use='pairwise.complete.obs')

col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))

corrplot(airlines_cor, method = "color", col = col(200),
         type = "upper", order = "hclust",
         addCoef.col = "black", # Add coefficient of correlation
         tl.col = "black", tl.srt = 45, #Text label color and rotation
         )
```

From the above correlation plot, you can see that of the features we used, "CarrierDelay", "DepDelayMinutes", and "LateAircraftDelay" have the highest correlations with "ArrDelayMinutes". The correlation between "CarrierDelay" and "ArrDelayMinutes" is 0.73, "DepDelayMinutes" and "ArrDelayMinutes" is 0.92, and so on.

5. ANOVA (Analysis of Variance)

Let's say that we want to analyze a categorical variable and see the correlation among different categories. For example, consider the airline dataset, the question we may ask is, how do different categories of the reporting airline feature (as a categorical variable) impact flight delays?

The bar graph below shows the average flight delays of different airlines. The code first groups the data by airline, then finds the average arrival delay for each airline, then it plots this as a bar chart.

In `geom_bar()`, you should set `stat = "identity"` since you are passing in the y values ("Average_Delays"). If you left out this parameter then by default `stat = "count"`, which would instead count the frequency of each airline.

[ ]:
```
summary_airline_delays <- sub_airline %>%
    group_by(Reporting_Airline) %>%
    summarize(Average_Delays = mean(ArrDelayMinutes, na.rm = TRUE))

summary_airline_delays %>%
    ggplot(aes(x = Reporting_Airline, y = Average_Delays)) +
```

```
    geom_bar(stat = "identity") +
    ggtitle("Average Arrival Delays by Airline")
```

To analyze categorical variables such as the "Reporting_Airline" variable, we can use a method such as the ANOVA method.

ANOVA: Analysis of Variance

The Analysis of Variance (ANOVA) is a statistical method used to test whether there are significant differences between the means of two or more groups. ANOVA returns two parameters:

- **F-test score**: ANOVA assumes the means of all groups are the same, calculates how much the actual means deviate from the assumption, and reports it as the F-test score. A larger score means there is a larger difference between the means.
- **P-value**: The p-value tells you how statistically significant the calculated score value is.

If our `ArrDelay` variable is strongly correlated with the variable we are analyzing, expect ANOVA to return a sizeable F-test score and a small p-value.

American Airline (AA) and Alaska Airline (AS)

The ANOVA test can be performed in base R's `stats` package using the `aov()` function. You can pass in the arrival delay data of the two airline groups that we want to compare and it calculates the ANOVA results.

In this first example, you can compare American Airline and Alaska Airline. The results confirm what we guessed at first. The flight delay between "AA" and "AS" are not significantly different, as the F score (0.13) is less than 1 and p-value is larger than 0.05.

[ ]:
```
aa_as_subset <- sub_airline %>%
  select(ArrDelay, Reporting_Airline) %>%
  filter(Reporting_Airline == 'AA' | Reporting_Airline == 'AS')

ad_aov <- aov(ArrDelay ~ Reporting_Airline, data = aa_as_subset)
summary(ad_aov)
```

American Airline (AA) and Pan Am Airline (PA (1))

As another example, you can compare American Airline and Pan Am Airline. From the below output, the arrival delay between "AA" and "PA (1)" are significantly different, since the F-score is very large (F = 17.95) and the p-value is 0.0000245 which is smaller than 0.05. All in all, we can say that there is a strong correlation between a categorical variable and other variables, if the ANOVA test gives us a large F-test value and a small p-value.

[ ]:
```
aa_pa_subset <- sub_airline %>%
  select(ArrDelay, Reporting_Airline) %>%
  filter(Reporting_Airline == 'AA' | Reporting_Airline == 'PA (1)')

ad_aov <- aov(ArrDelay ~ Reporting_Airline, data = aa_pa_subset)
summary(ad_aov)
```

Conclusion: Important Variables

We now have a better idea of what our data looks like and which variables are important to take into account when predicting the arrival delay (ArrDelay). We have narrowed it down to the following variables:

Continuous numerical variables:

- DepDelayMinutes
- CarrierDelay
- LateAircraftDelay

Categorical variables:

- ReportingAirline

As we now move into building machine learning models to automate our analysis, feeding the model with variables that meaningfully affect our target variable will improve our model's prediction performance.

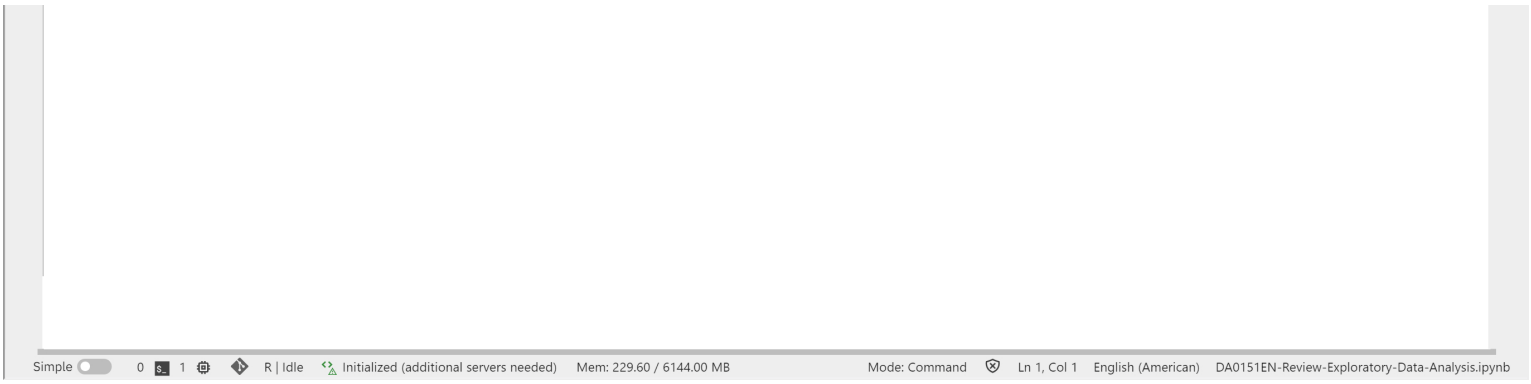
Thank you for completing this notebook

About the Authors:

This notebook was written by Yiwen Li and Gabriela de Queiroz.

Yiwen Li is a developer advocate and data scientist at IBM. She has been creating online content such as code patterns, tutorials, and blogs demonstrating the potential of products and services offered by IBM (like Watson Studio, Machine learning, Model Asset eXchange, Data Asset eXchange, etc.). She holds dual degree, including BS in Probability and Statistics and BA in Economics from the University of California, San Diego.

Gabriela de Queiroz is a Sr. Engineering & Data Science Manager at IBM where she manages and leads a team of developers working on Data & AI Open Source projects. She works to democratize AI by building tools and launching new open source projects. She is the founder of AI Inclusive, a global organization that is helping increase the representation and participation of gender minorities in Artificial Intelligence. She is also the founder of R-Ladies, a worldwide organization for promoting diversity in the R community with more than 190 chapters in 50+ countries. She has worked in several startups and where she built teams, developed statistical models, and employed a variety of techniques to derive insights and drive data-centric decisions


Simple
0
1
R | Idle
Initialized (additional servers needed)
Mem: 229.60 / 6144.00 MB
Mode: Command
Ln 1, Col 1
English (American)
DA0151EN-Review-Exploratory-Data-Analysis.ipynb