Showcasing my R Proficiency

Azizah

2024-07-08

Installing the 'Tidyverse' package

The Tidyverse package is a compilation of R packages specifically crafted for efficient data manipulation and analysis. It is considered a key part of programming for most R users.

I will install the tidyverse package first as it will be helpful for all of my analyses.

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

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

The code above doesn't show an output because 'tidyverse' has already been pre-installed since I first started using R.

Loading the 'Tidyverse' package

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── 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 become errors
```

No output, also because it's been pre-loaded.

Creating data frames

Data frames are one of the basic tools used to work with data in R. One of the most common ways to create them is using individual vectors of data and then combining them into a data frame using the `data.frame()` function.

First, I will create a vector of names of Arsenal players.

```
names <- c("Saka", "Jesus", "Rice", "Saliba")
```

Then, I will create a vector of their respective ages

```
age <- c(22, 27, 25, 23)
```

With these two vectors, I will create a new data frame called `arsenal_players`:

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

There are different ways of previewing data in R, such as using the 'head()', 'str()', and 'glimpse()' functions. I will use the head function in the following code:

```
head(arsenal_players)
```

```
##    names age
## 1   Saka  22
## 2  Jesus  27
```

```
## 3   Rice  25
## 4 Saliba  23
```

As I have now created a data frame, I can work with it using all of the tools in R. I will use the `mutate()` function to create a new variable that would capture each player's age in twenty years:

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

```
##    names age age_in_20
## 1   Saka  22        42
## 2  Jesus  27        47
## 3   Rice  25        45
## 4 Saliba  23        43
```

To showcase my skills, I will perform some analysis based on hypothetical situations involving a junior data analyst.

Case 1 - Cleaning Data

As a junior data analyst working for a hotel booking company, I 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, I am going to need to use functions to preview the data's structure, including its columns and rows. I will also need to use basic cleaning functions to prepare this data for analysis.

I will use the `read_csv()` function to import data from the .csv file in a project folder called "hotel_bookings.csv" and save it as a data frame called `bookings_df`.

```
setwd("/cloud/project/Course 7/Week 3")
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.
```

To find out what columns I have in my data frame, I will run the `colnames()` function:

```
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"
```

To preview the columns and first several rows of data, I will use the 'head()' function:

```
head(bookings_df)
```

```
## # A tibble: 6 × 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 that I have imported the data I want to work with, one of the important things to do while cleaning and organising data is focusing on a part of the dataset that is relevant to the goal of a project. I will create a data frame using `bookings_df` that focuses on the average daily rate, which is referred to as `adr` in the data frame, and `adults`. This also shows my prowess working with data frames as data frames can be created from vectors (Arsenal Players) or from an external file that has been imported into R (hotel bookings file)

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

To view,

```
head(new_df)
```

```
## # A tibble: 6 × 2
##     adr adults
##   <dbl>  <dbl>
## 1     0      2
## 2     0      2
## 3    75      1
## 4    75      1
## 5    98      2
## 6    98      2
```

I will also create new variables in my data frame using the `mutate()` function. This will make changes to the data frame, but not to the original data set I imported.

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

```
## # A tibble: 119,390 × 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
```

Some important packages for cleaning data include the 'skimr' and 'janitor' packages. I will install them so they can help me in my next steps:

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

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

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

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

I will now load them for use:

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

```
##
## Attaching package: 'janitor'
```

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

I am now primarily interested in the variables: 'hotel', 'is_canceled', and 'lead_time'. Therefore, I will create a new data frame with just those columns, calling it `trimmed_df`:

```
trimmed_df <- bookings_df %>%
  select(hotel, is_canceled, lead_time)
```

Some of the column names aren't very intuitive, therefore, I will rename them to make them easier to understand. I will create the same exact data frame as above, but rename the variable 'hotel' as 'hotel_type' to be crystal clear on what the data is about:

```
trimmed_df %>%
  select(hotel, is_canceled, lead_time) %>%
  rename(hotel_type = hotel)

## # A tibble: 119,390 × 3
##    hotel_type   is_canceled lead_time
##    <chr>              <dbl>     <dbl>
##  1 Resort Hotel           0       342
##  2 Resort Hotel           0       737
##  3 Resort Hotel           0         7
##  4 Resort Hotel           0        13
##  5 Resort Hotel           0        14
##  6 Resort Hotel           0        14
##  7 Resort Hotel           0         0
##  8 Resort Hotel           0         9
##  9 Resort Hotel           1        85
## 10 Resort Hotel           1        75
## # ℹ 119,380 more rows
```

I can also split or combine data in different columns. In this example, I will combine the arrival month and year into one column using the unite() function:

```
example_df <- bookings_df %>%
  select(arrival_date_year, arrival_date_month) %>%
  unite(arrival_month_year, c("arrival_date_month", "arrival_date_year"), sep
= " ")
```

I will also use the`mutate()` function to make changes to the columns. I will create a new column that sums up all the adults, children, and babies on a reservation for the total number of people:

```
example_df <- bookings_df %>%
  mutate(guests = adults + children + babies)

head(example_df)
```

```
## # A tibble: 6 × 33
##   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 28 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, I will calculate some summary statistics! I will calculate the total number of canceled bookings and the average lead time for booking - you'll want to start your code after the %>% symbol. Make a column called 'number_canceled' to represent the total number of canceled bookings. Then, make a column called 'average_lead_time' to represent the average lead time. Use the `summarize()` function to do this in the code chunk below:

```
example_df <- bookings_df %>%
  summarize(number_canceled = sum(is_canceled), average_lead_time =
mean(lead_time))
head(example_df)
```

```
## # A tibble: 1 × 2
##   number_canceled average_lead_time
##             <dbl>             <dbl>
## 1           44224              104.
```

Case 2 - Manipulating and Changing Data in R

The next steps I will take will involve manipulating and changing data. I will use functions to manipulate the data, use statistical summaries to explore the data, and gain initial insights for my hypothetical stakeholders.

In this scenario, I am a junior data analyst working for a hypothetical hotel booking company. I have been asked to clean a .csv file that was created after querying a database to combine two different tables from different hotels. I have already performed some basic cleaning functions on this data; the next steps will focus on using functions to conduct basic data manipulation.

I will arrange the data by most lead time to least lead time because I want to focus on bookings that were made far in advance. I will do this using the `arrange()` function:

```
setwd("/cloud/project/Course 7/Week 3")
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.
```

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

```
## # A tibble: 119,390 × 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>, …
```

I will now find the maximum and minimum lead times without sorting the whole dataset using the `arrange()` function:

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

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

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

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

Now, I want to find what the average lead time for booking is because my boss has asked me how early we should run promotions for hotel rooms:

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

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

I have reported to my 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 we're running by targeting major cities. My first step will be creating a new dataset that only contains data about city hotels. I can do that using the `filter()` function, and name my new data frame 'hotel_bookings_city':

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

To preview:

```
head(hotel_bookings_city)
```

```
## # A tibble: 6 × 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>, …
```

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

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

Now, my boss wants to know a lot more information about city hotels, including the maximum and minimum lead time. They are also interested in how these are different from resort hotels. I don't want to run each line of code over and over again, so i've decided to use the `group_by()`and`summarize()` functions. I will also use the pipe operator to make my code easier to follow. I 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))
```

To preview:

```
head(hotel_summary)
```

```
## # A tibble: 2 × 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
```

Case 3 - Data Visualization

As a junior data analyst working for a hypothetical hotel booking company, I have cleaned and manipulated the data, and gotten some initial insights that I would now like to share. Now, I am going to create some simple data visualizations with the `ggplot2` package. I will use basic `ggplot2` syntax and troubleshoot some common errors I might encounter:

Now, I need to install and load the 'ggplot2' package.

```
install.packages('ggplot2')
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

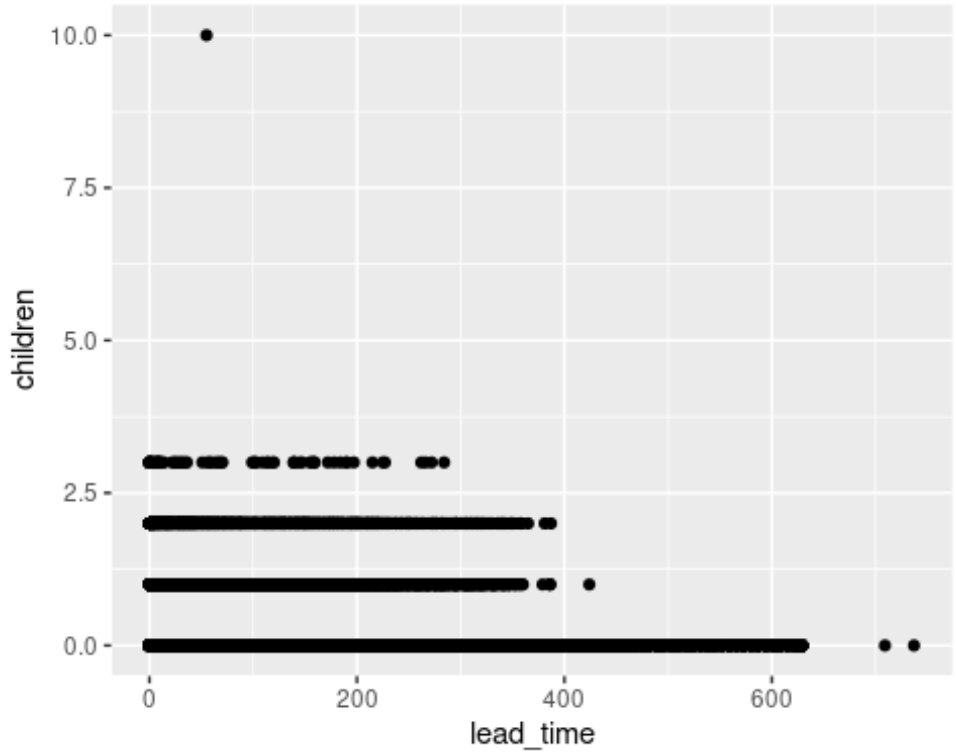
```
library(ggplot2)
```

A stakeholder tells me, "I want to target people who book early, and I have a hypothesis that people with children have to book in advance."

I have decide to create a visualization to see how true that statement is or isn't. I will use `ggplot2` to do this:

```
ggplot(data = hotel_bookings) +
  geom_point(mapping = aes(x = lead_time, y = children))
```

```
## Warning: Removed 4 rows containing missing values or values outside the
scale range
## (`geom_point()`).
```



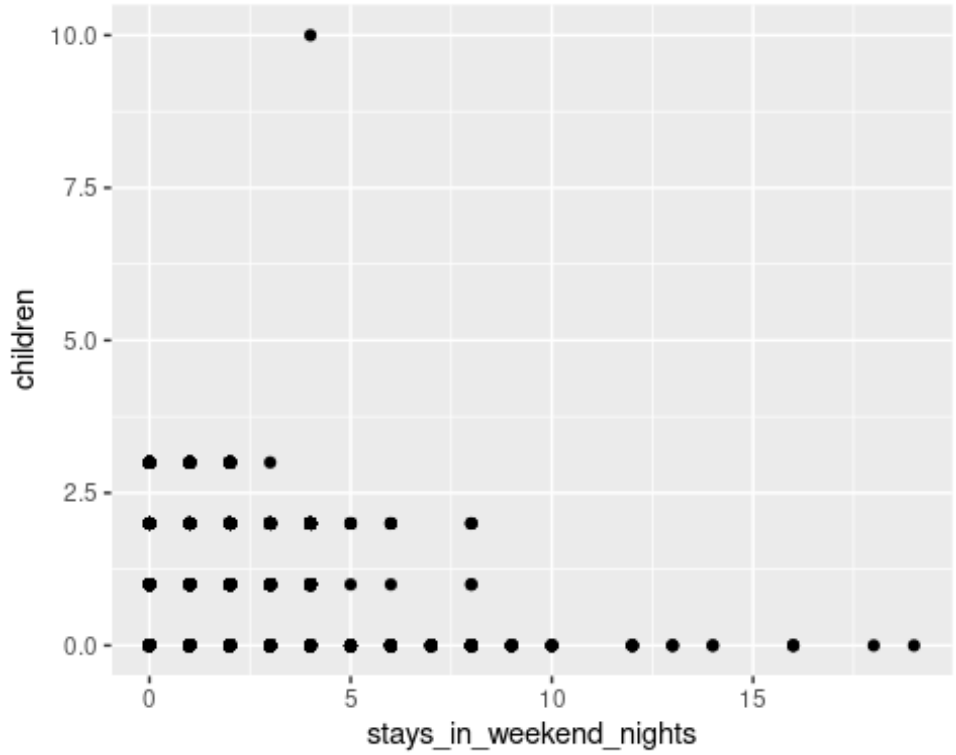
On the x-axis, the plot shows how far in advance a booking is made, with the bookings furthest to the right happening the most in advance. On the y-axis it shows how many children there are in a party.

The plot reveals that my stakeholder's hypothesis is incorrect. I will report back to my stakeholder to inform them that many of the advanced bookings are being made by people with 0 children.

My stakeholder now says that she wants to increase weekend bookings, an important source of revenue for the hotel. Shw wants to know what group of guests book the most weekend nights in order to target that group in a new marketing campaign. She suggests that guests without children book the most weekend nights. I need to find out if this is true:

```
ggplot(data = hotel_bookings) +
 geom_point(mapping = aes(x = stays_in_weekend_nights, y = children))
```

```
## Warning: Removed 4 rows containing missing values or values outside the
scale range
## (`geom_point()`).
```

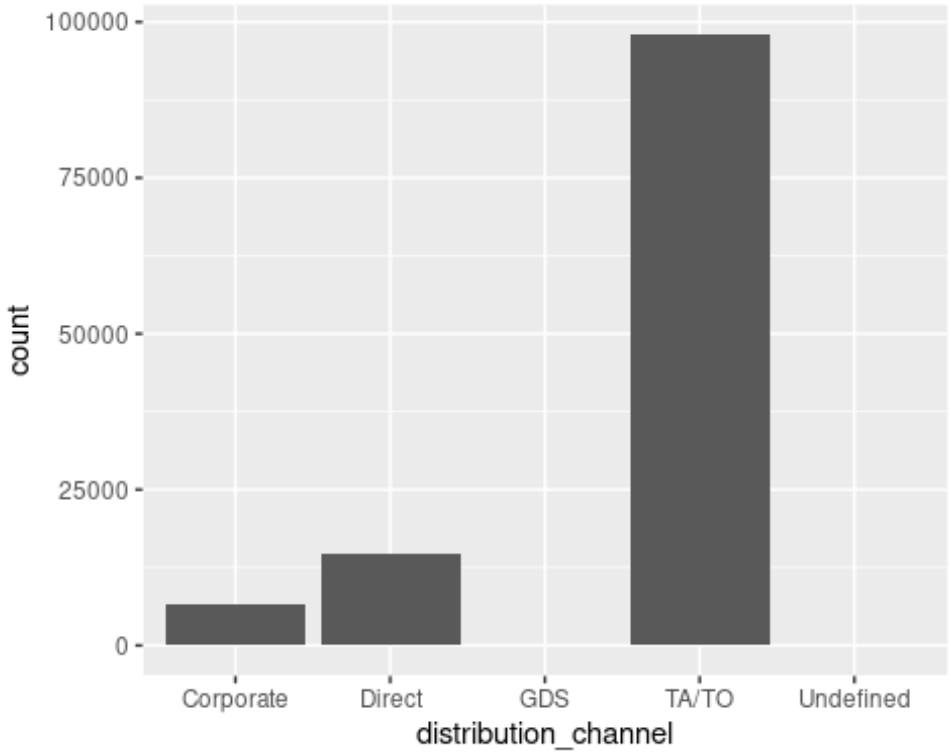


My stakeholder is correct. It is true that guests without children book the most weekend nights.

Now, I am interested in creating visualizations that highlight different aspects of the data to present to my stakeholder:

My stakeholder is interested in developing promotions based on different booking distributions, but first they need to know how many of the transactions are occurring for each different distribution type. I will create a bar chart for this:

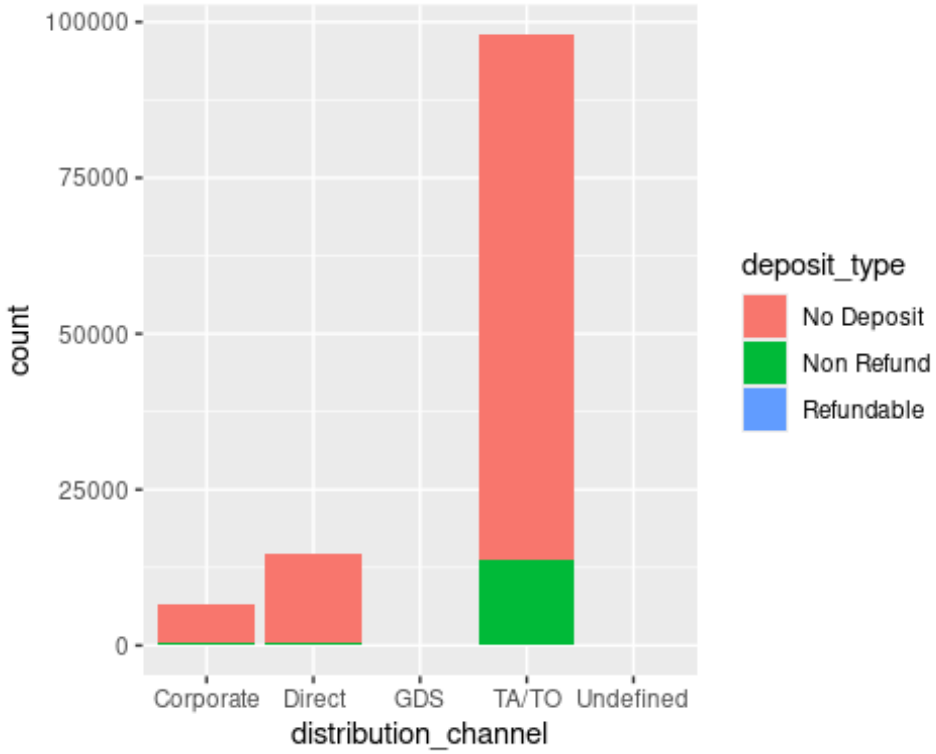
```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = distribution_channel))
```



The TA/TO distribution type has the most number of bookings.

My stakeholder has more questions. Now they want to know if the number of bookings for each distribution type is different depending on whether or not there was a deposit or what market segment they represent. I will edit the previous code to show this for deposit and market segment:

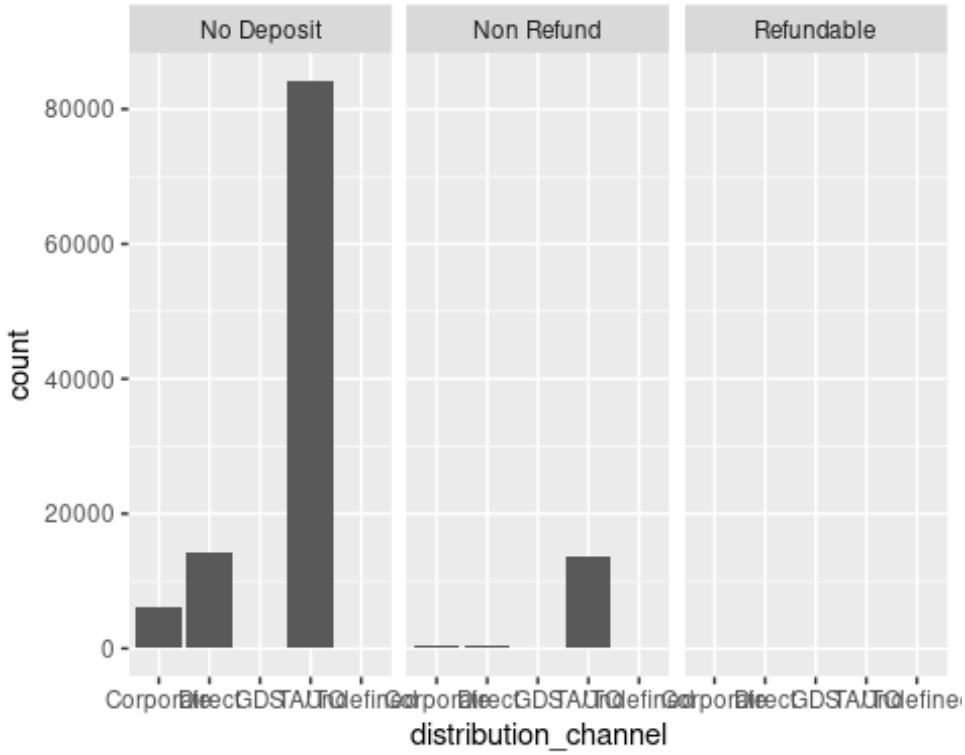
```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = distribution_channel, fill=deposit_type))
```




```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = distribution_channel, fill=market_segment))
```

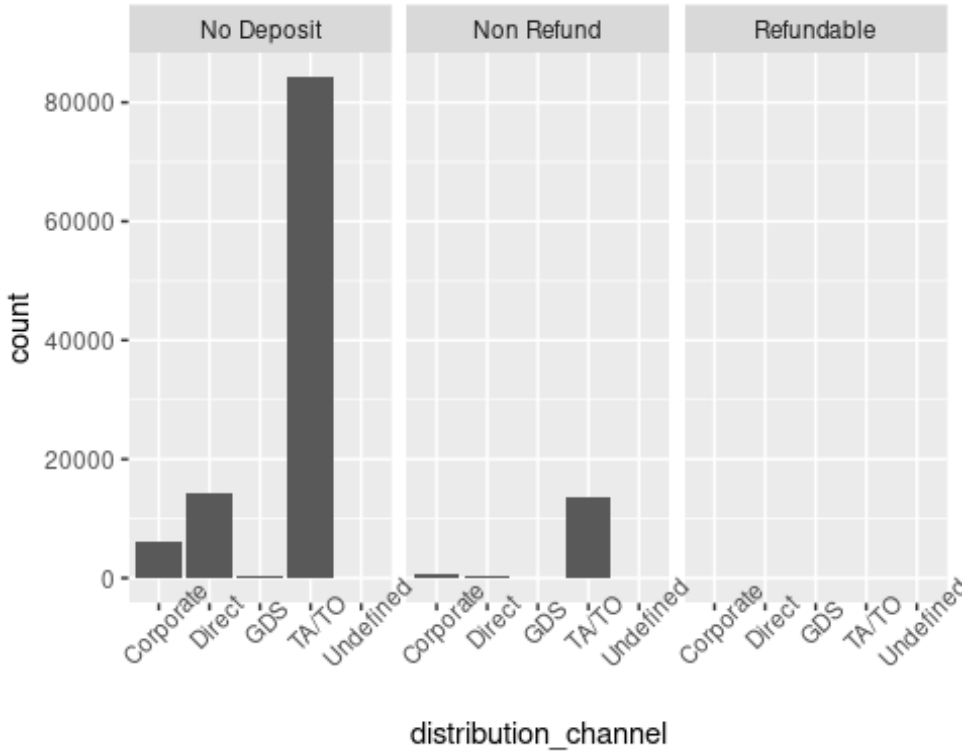
After reviewing the new charts, my stakeholder asks me to create separate charts for each deposit type and market segment to help them understand the differences more clearly.

```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = distribution_channel)) +
  facet_wrap(~deposit_type)
```



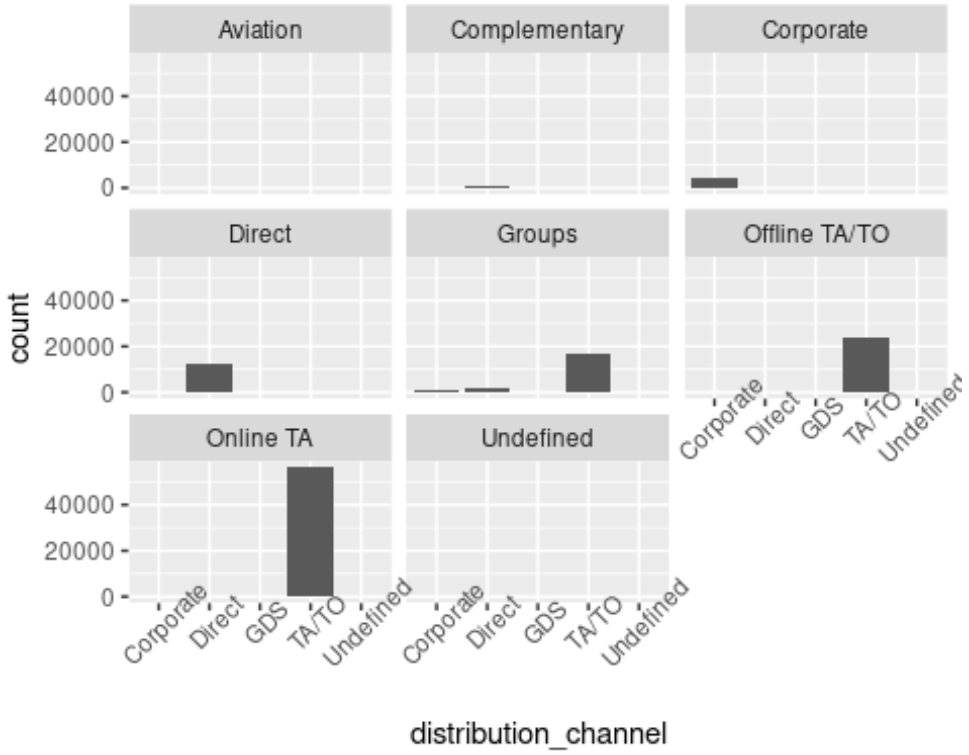
My new code creates three bar charts for 'no_deposit', non_refund', and 'refundable' deposit types. However, it's hard to read the x-axis labels here, so I'll add one piece of code at the end that rotates the text to 45 degrees to make it easier to read.

```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = distribution_channel)) +
  facet_wrap(~deposit_type) +
  theme(axis.text.x = element_text(angle = 45))
```



Now, i'd create a chart for different market segments:

```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = distribution_channel)) +
  facet_wrap(~market_segment) +
  theme(axis.text.x = element_text(angle = 45))
```



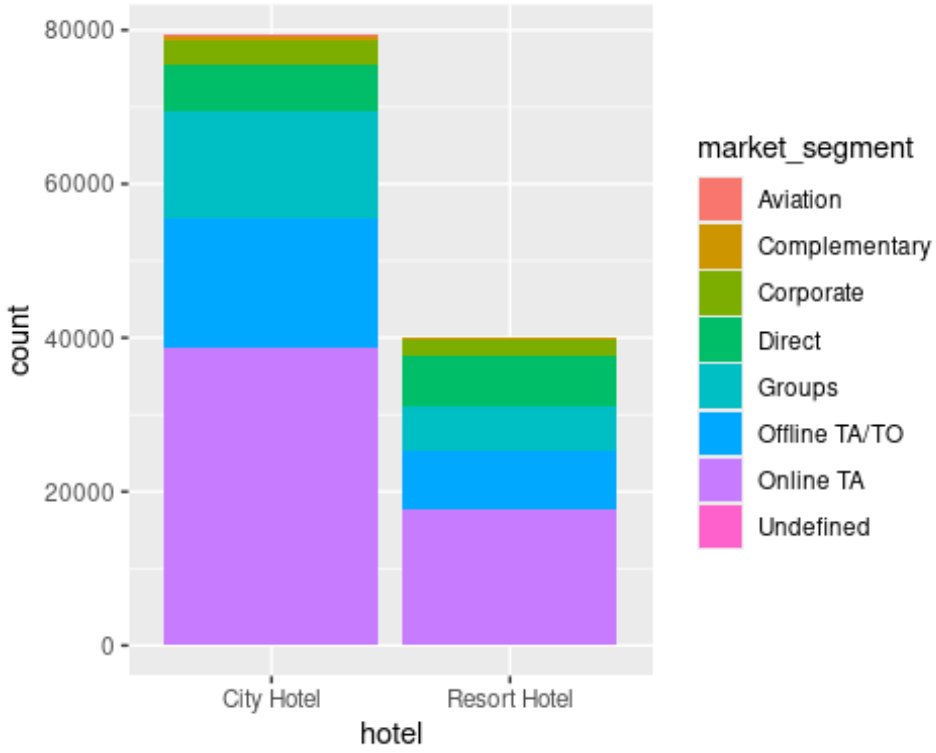
Case 4 - Applying Filters

As a junior data analyst for a hotel booking company, I have been asked to clean hotel booking data, create visualizations with `ggplot2` to gain insight into the data, and present different facets of the data through visualization. Now, I am going to build on the work I performed previously to apply filters to the data visualizations in `ggplot2`:

Now, my stakeholder wants to run a family-friendly promotion targeting key market segments. She wants to know which market segments generate the largest number of bookings, and where these bookings are made (city hotels or resort hotels).

First, I will create a bar chart showing each hotel type and market segment:

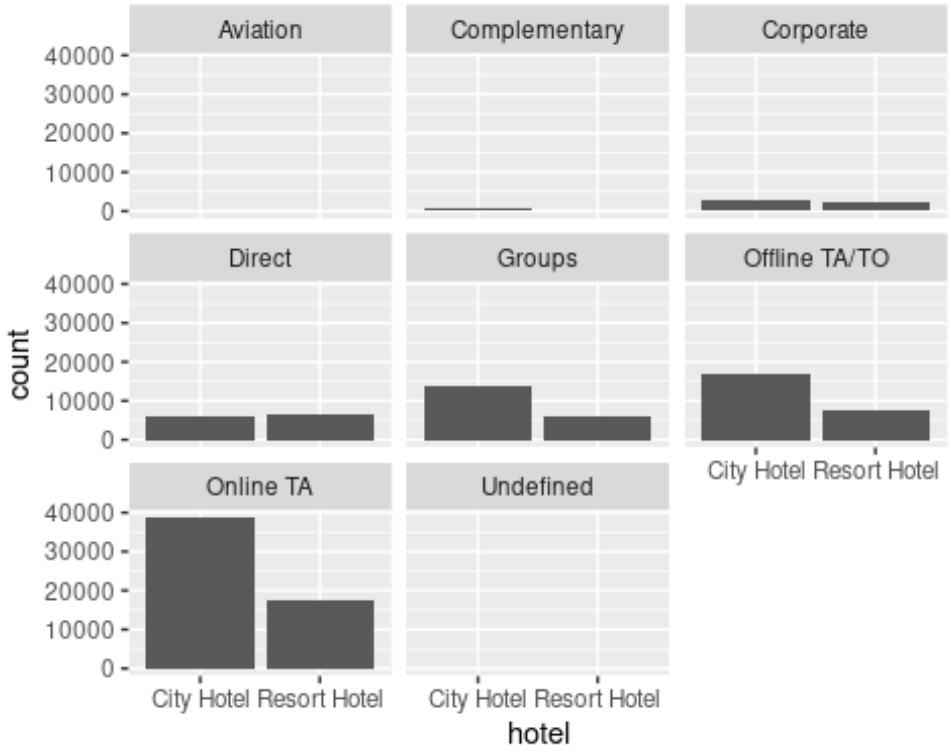
```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = hotel, fill = market_segment))
```



It's difficult to compare the size of the market segments at the top of the bars. I want my stakeholder to be able to clearly compare each segment.

Therefore, I will use the facet_wrap() function to create a separate plot for each market segment.

```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = hotel)) +
  facet_wrap(~market_segment)
```



After considering all the data, my stakeholder decides to send the promotion to families that make online bookings for city hotels. The online segment is the fastest growing segment, and families tend to spend more at city hotels than other types of guests.

She asks if I can create a plot that shows the relationship between lead time and guests traveling with children for online bookings at city hotels. This will give her a better idea of the specific timing for the promotion.

I've thought about it, and I have all the tools I need to fulfill the request. I'll break it down into the following two steps:

- filtering your data
- plotting your filtered data

```
onlineta_city_hotels <- filter(hotel_bookings,
                               (hotel=="City Hotel" &
                                 hotel_bookings$market_segment=="Online TA"))
```

I will use the`View()` function to check out my new data frame:

```
head(onlineta_city_hotels)
```

```
## # A tibble: 6 × 32
##   hotel      is_canceled lead_time arrival_date_year arrival_date_month
##   <chr>            <dbl>     <dbl>             <dbl> <chr>
## 1 City Hotel           1        88              2015 July
## 2 City Hotel           1        65              2015 July
## 3 City Hotel           1        92              2015 July
```

```
## 4 City Hotel           1       100              2015 July
## 5 City Hotel           1        79              2015 July
## 6 City Hotel           1        63              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>, …
```

I can also use the pipe operator (%>%) to do this in steps!

```
onlineta_city_hotels_v2 <- hotel_bookings %>%
  filter(hotel=="City Hotel") %>%
  filter(market_segment=="Online TA")
```

To preview:

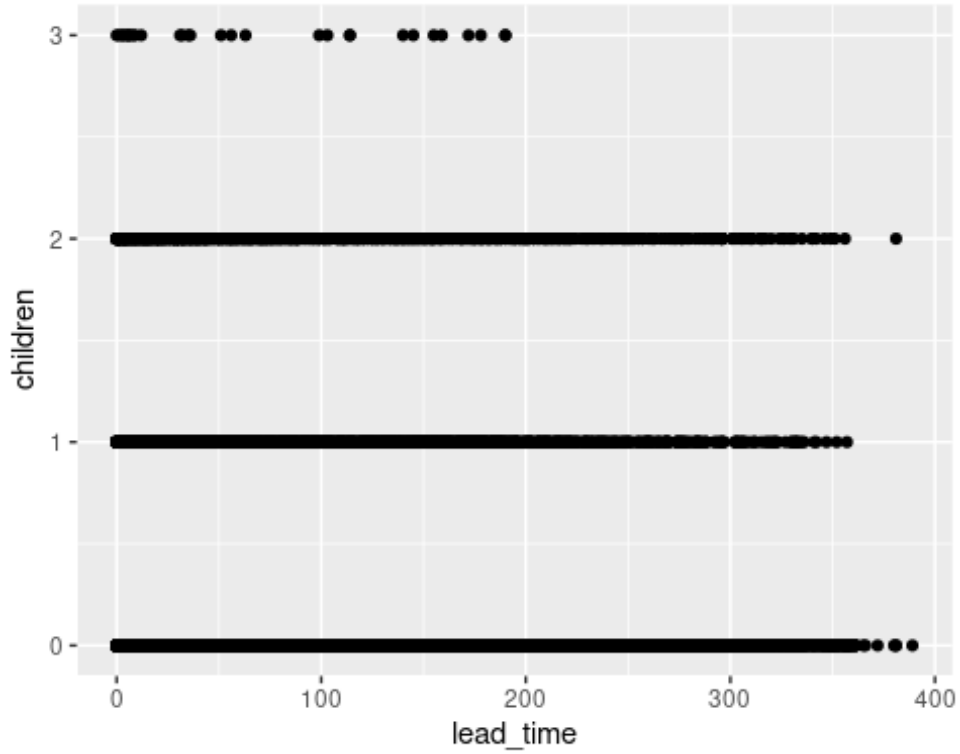
```
head(onlineta_city_hotels_v2)
```

```
## # A tibble: 6 × 32
##   hotel      is_canceled lead_time arrival_date_year arrival_date_month
##   <chr>            <dbl>     <dbl>             <dbl> <chr>
## 1 City Hotel           1        88              2015 July
## 2 City Hotel           1        65              2015 July
## 3 City Hotel           1        92              2015 July
## 4 City Hotel           1       100              2015 July
## 5 City Hotel           1        79              2015 July
## 6 City Hotel           1        63              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>, …
```

To plot the data my stakeholder has requested:

```
ggplot(data = onlineta_city_hotels) +
  geom_point(mapping = aes(x = lead_time, y = children))
```

```
## Warning: Removed 1 row containing missing values or values outside the
scale range
## (`geom_point()`).
```



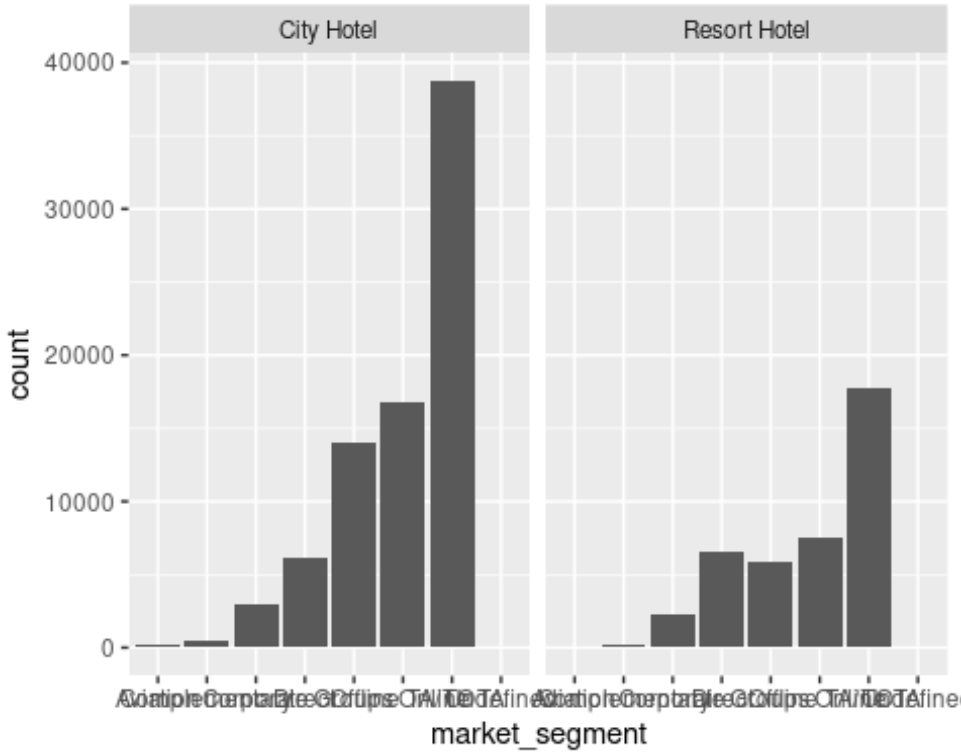
The plot reveals that bookings with children tend to have a shorter lead time, and bookings with 3 children have a significantly shorter lead time (<200 days). So, promotions targeting families can be made closer to the valid booking dates.

Case 5 - Chart Annotations

For the following steps, I'm going to be adding annotations to data visualizations with ggplot2. I will also save the images so I can add them directly to formal presentations for my stakeholders.

My stakeholder tells me that they would like me to create a visualization that compares market segments between city hotels and resort hotels. This will help inform how the company targets promotions in the future. They ask me to create a cleaned and labeled version and save it as a .png file so it can be included in a presentation.

```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = market_segment)) +
  facet_wrap(~hotel)
```

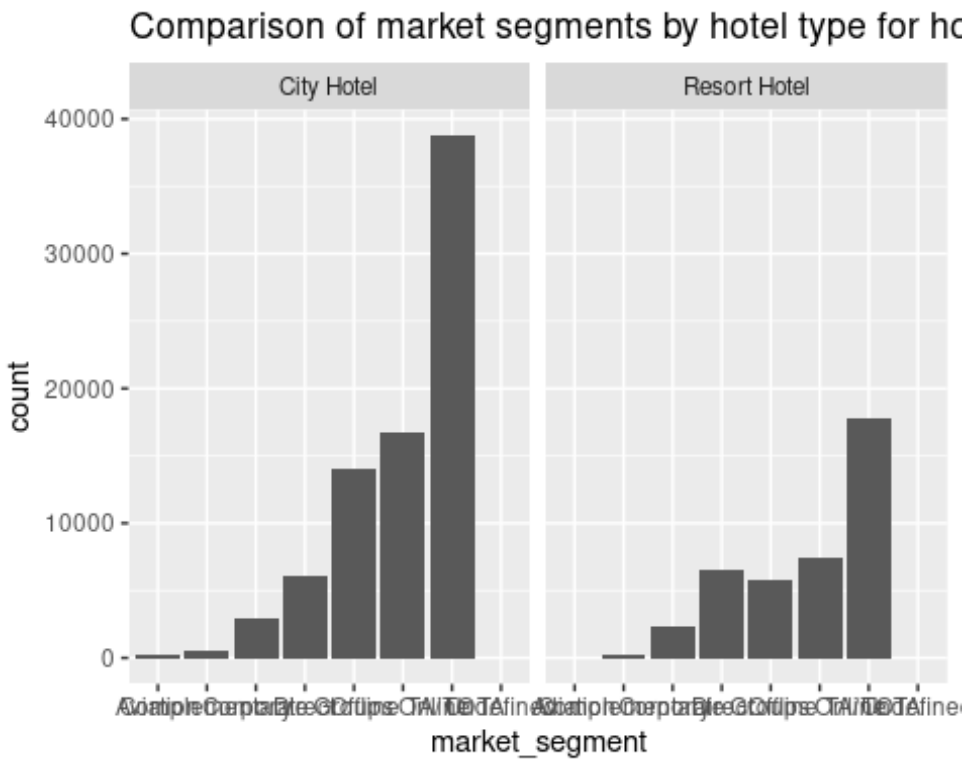


This creates two bar graphs: one for 'city_hotel' data and one for 'resort_hotel' data. The x axis is 'market_segment' and the y axis is 'count' for both charts.

In this visualization it is unclear where the data is from, what the main takeaway is, or even what the data is showing. To explain all of that, I will leverage annotations in `ggplot2`.

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, I will add `labs()` at the end of my `ggplot()` command and then input a title there:

```
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")
```



I also want to add another detail about what time period this data covers. To do this, I need to find out when the data is from. I will use the 'min()' and 'max()' functions on the year column in the data:

```
min(hotel_bookings$arrival_date_year)

## [1] 2015

max(hotel_bookings$arrival_date_year)

## [1] 2017
```

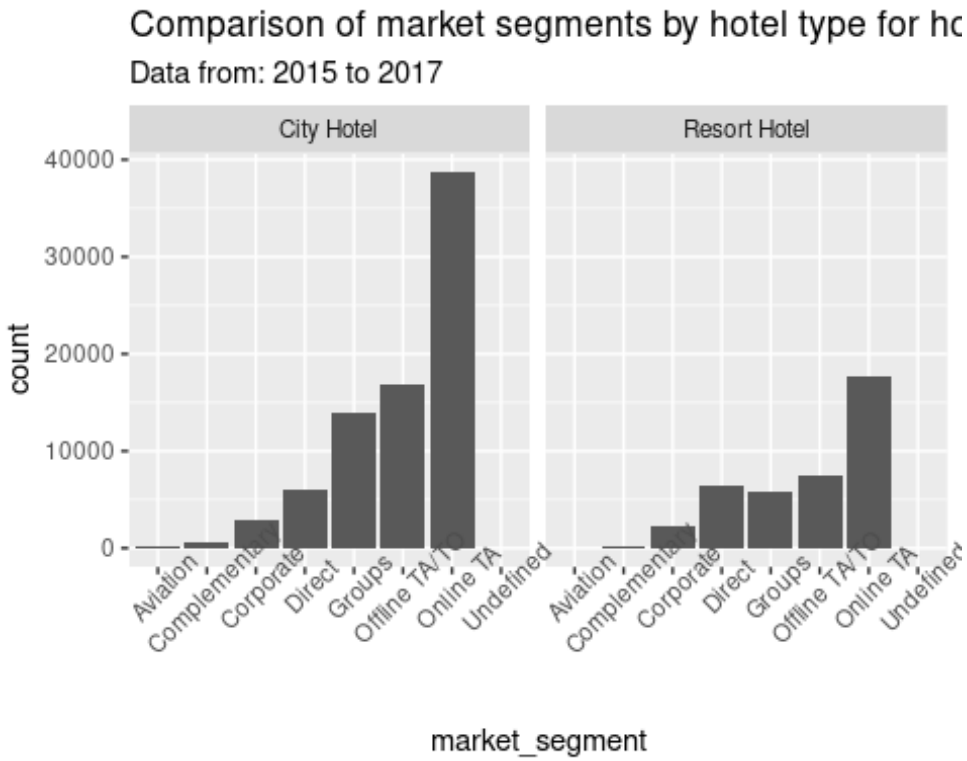
I need to save them as variables in order to easily use them in my labeling:

```
mindate <- min(hotel_bookings$arrival_date_year)
maxdate <- max(hotel_bookings$arrival_date_year)
```

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

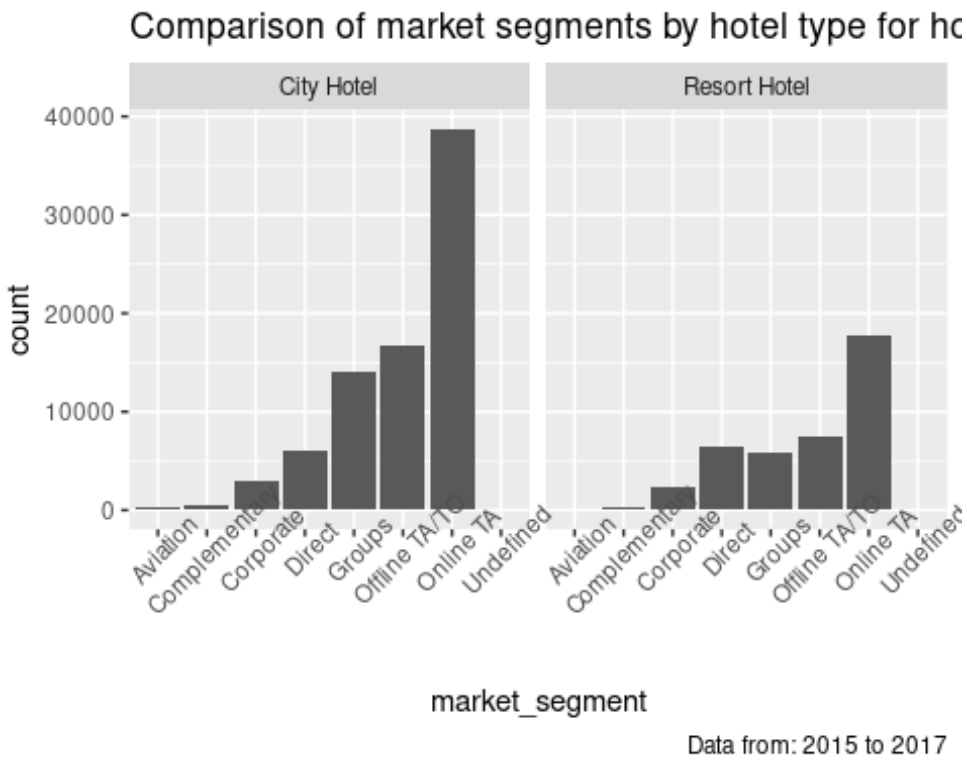
```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = market_segment)) +
  facet_wrap(~hotel) +
  theme(axis.text.x = element_text(angle = 45)) +
  labs(title="Comparison of market segments by hotel type for hotel
```

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



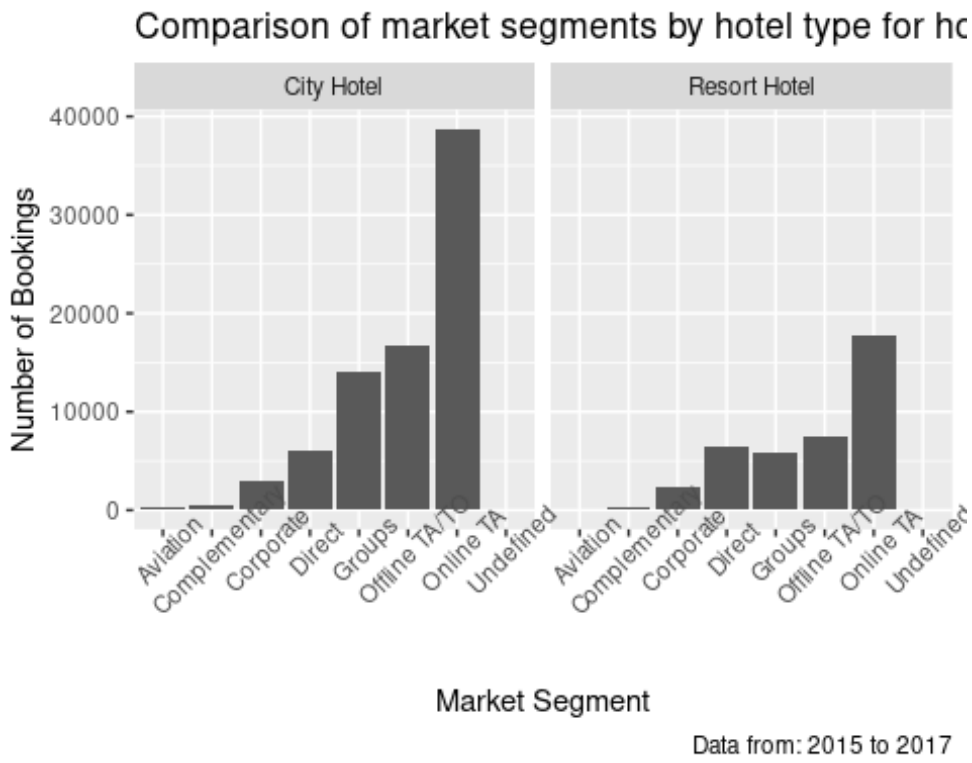
This chart is displaying the technical details a little too prominently. I don't want that to be the second thing people notice during the presentation. Therefore, I will switch the `subtitle` to a `caption` which will appear in the bottom right corner instead:

```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = market_segment)) +
  facet_wrap(~hotel) +
  theme(axis.text.x = element_text(angle = 45)) +
  labs(title="Comparison of market segments by hotel type for hotel
bookings",
       caption=paste0("Data from: ", mindate, " to ", maxdate))
```



Now I want to clean up the x and y axis labels to make sure they are really clear. To do that, I will add to the `labs()` function using `x=` and `y=`:

```
ggplot(data = hotel_bookings) +
  geom_bar(mapping = aes(x = market_segment)) +
  facet_wrap(~hotel) +
  theme(axis.text.x = element_text(angle = 45)) +
  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")
```



Saving the charts

Now, I need to save the above chart so I can easily share with stakeholders. As 'ggsave()' by default saves the last plot that was generated, I will use it:

```
ggsave('hotel_booking_chart.png', width=7, height=7)
```

To make my chart bigger and more rectangular to fit the slide show presentation, I would specify the height and width of my .png in the `ggsave()` command.

```
ggsave('hotel_booking_chart.png',
       width=16,
       height=8)
```