Telling stories with charts - visualizations with ggplot2

Adi Sarid

What for and why now?

When we think about the data science workflow, it makes a lot of sense to start our lesson with tidying data or data transformations, however, I chose to start with visualizations. Why is that?

I'm adopting the approach of the "R for Data Science book", which talks about visualizations in one of its early chapters (even before data transformations), because charts give a lot of motivation to our next chapters and also helps us develop a way of thinking.

As you will see, the following exercises will also contain some elements of tidying and transforming data (because sometimes its a necessity towards working on a chart). In a future lesson we will delve deeper and extend our knowledge into tidying and data transformations.

Remember: When you do the exercises, if you get stuck, put on a pink sticky note. If I haven't noticed it, call me. If you finished, put up the green one.

Exercise 1: the *google play* data set

In this exercise we will work with a file downloaded from the competition website kaggle. You can see the details here. You are going to load the file directly from our course's repository, using the following command:

```r
suppressMessages(
   library(tidyverse)
   )
google_play_raw <- read_csv(file = "https://raw.githubusercontent.com/adisarid/Riskified_training/maste
```

```
## Parsed with column specification:
## cols(
##   App = col_character(),
##   Category = col_character(),
##   Rating = col_double(),
##   Reviews = col_double(),
##   Size = col_character(),
##   Installs = col_character(),
##   Type = col_character(),
##   Price = col_character(),
##   `Content Rating` = col_character(),
##   Genres = col_character(),
##   `Last Updated` = col_character(),
##   `Current Ver` = col_character(),
##   `Android Ver` = col_character()
## )

## Warning: 2 parsing failures.
##   row     col               expected     actual
## 10473 Reviews no trailing characters M          'https://raw.githubusercontent.com/adisarid/Riskifie
## 10473 NA      13 columns             12 columns 'https://raw.githubusercontent.com/adisarid/Riskifie
```

Question 1: Did you notice I used the `supressMessages(...)` function. What does it do? Why did I use it?

It was used because `tidyverse` overrides some commands and then throws out a nasty message. When using it in Rmarkdown it is a way (one of a few) to hide these messages.

Question 2: Did you notice any "parsing failures" when reading the file? Try to figure out what do they mean, and think about how would you solve the problem. The error message contains a row location so you can look at the data in the area of the error to try and figure it out, e.g.:

```
location <- ???
google_play_raw[(location-3):(location+3),]
```

Once you figure out what's wrong, filter this line out of the file (but note that there are other ways to handle this).

```
google_play <- google_play_raw[-location,]
```

Since this file is the result of web scraping, some errors are expected, but in fact it is a relatively clean file. This parsing error is caused by the fact that the `read_csv` reads the first 10,000 lines and uses them to decide on data types of each column. The error is in row 10473, and is caused by the fact that there is a value missing from the line. Then all other values get pushed to the wrong position.

Question 3: Is there any other way you could have filtered this messy line? (think about `filter()`, the `is.na()` and `!` operator)

```
google_play <- google_play_raw %>%
   filter(!is.na(`Android Ver`))
```

Look at the dataset using `glimpse(???)`, notice that most columns were read as characters and a few as double.

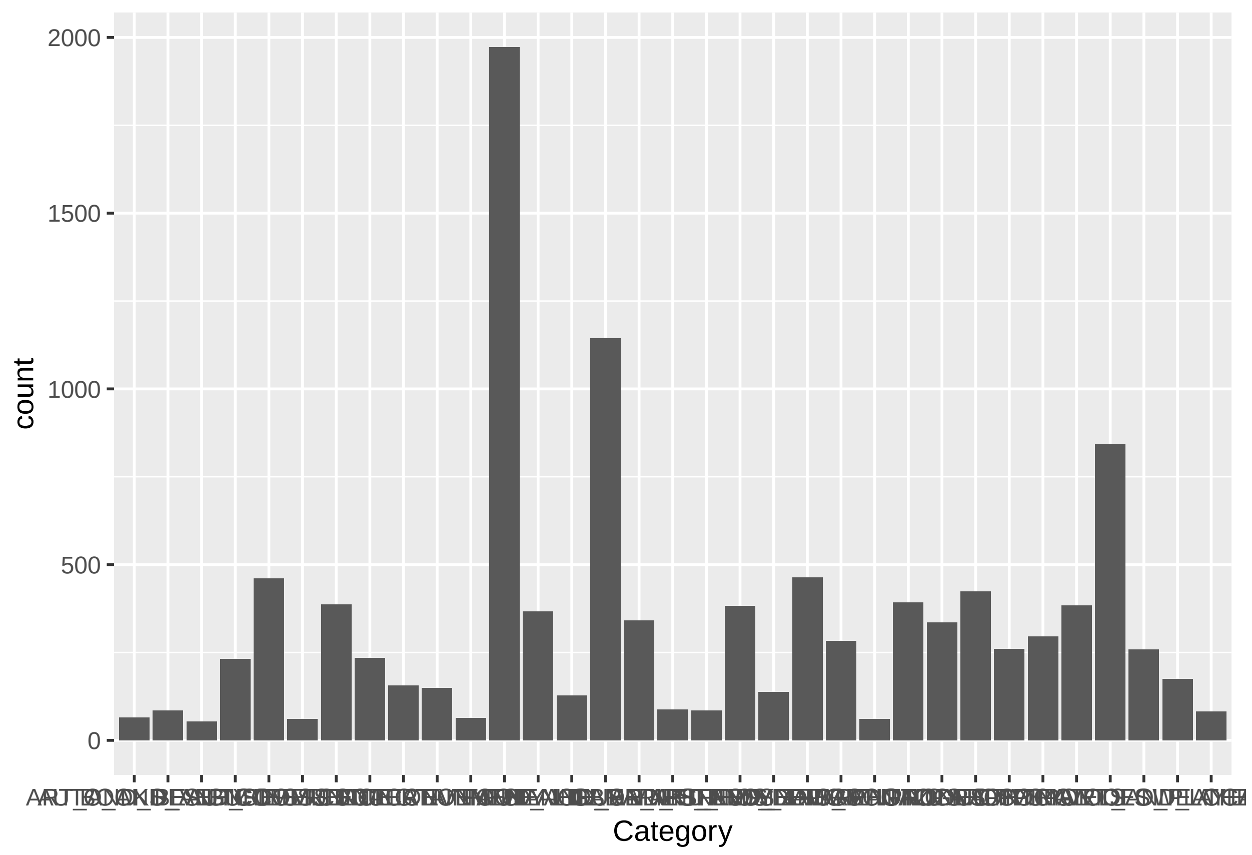
Question 4: What variables were read as character but you think that you would be better off if they were tranformed to a different type?

To factor: Category, Installs, Type, Content Rating, Genres, Android Ver. To date: Last Updated. To numeric: using string manipulation: Size (the app's size, perhaps also Installs).

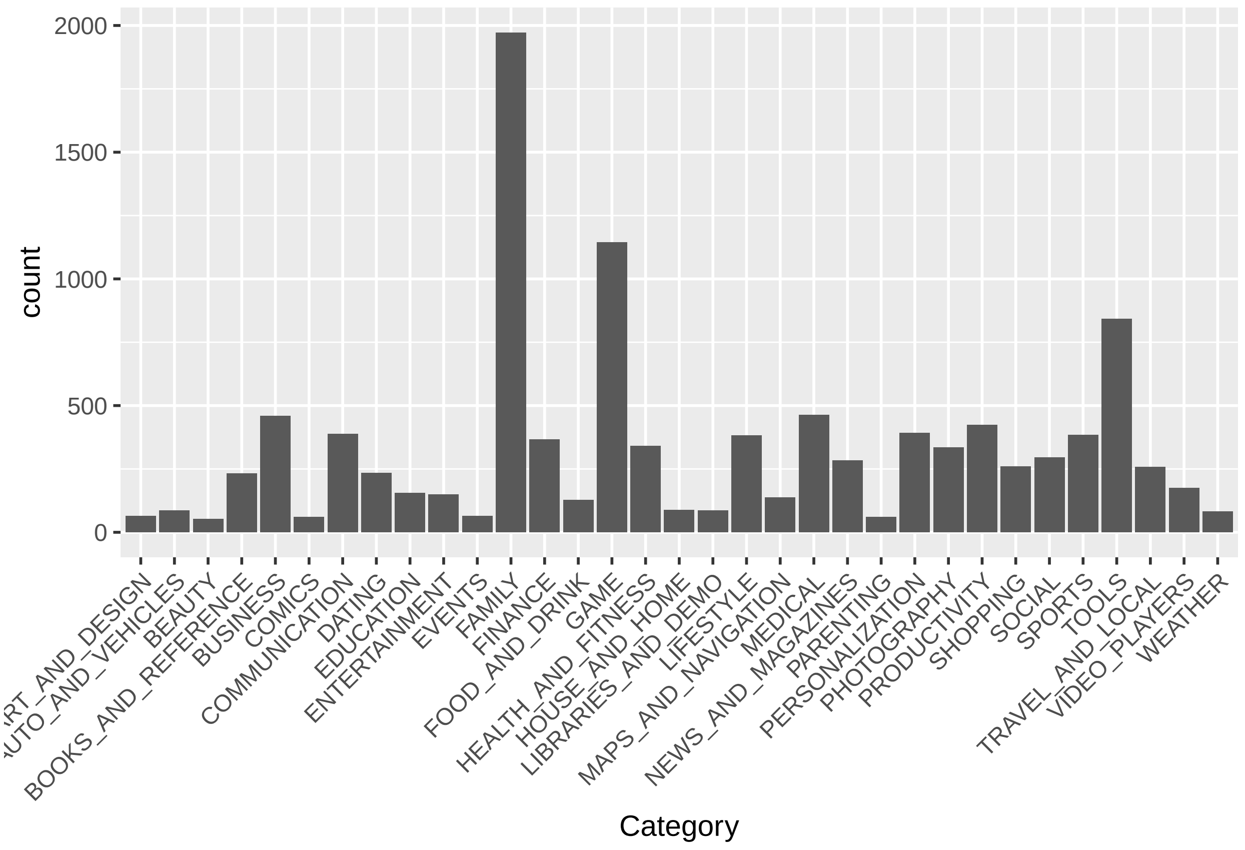
The `Category` variable represents the app's category.

Question 5: Complete the code below and answer the questions that follow. Note that the last part of the code (`theme(...)`) is meant to rotate the x axis labels, for better readability. First try without it and then add it.

```
ggplot(data = google_play, mapping = aes(x = Category)) +
   geom_bar()
```



```r
ggplot(data = google_play, mapping = aes(x = Category)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



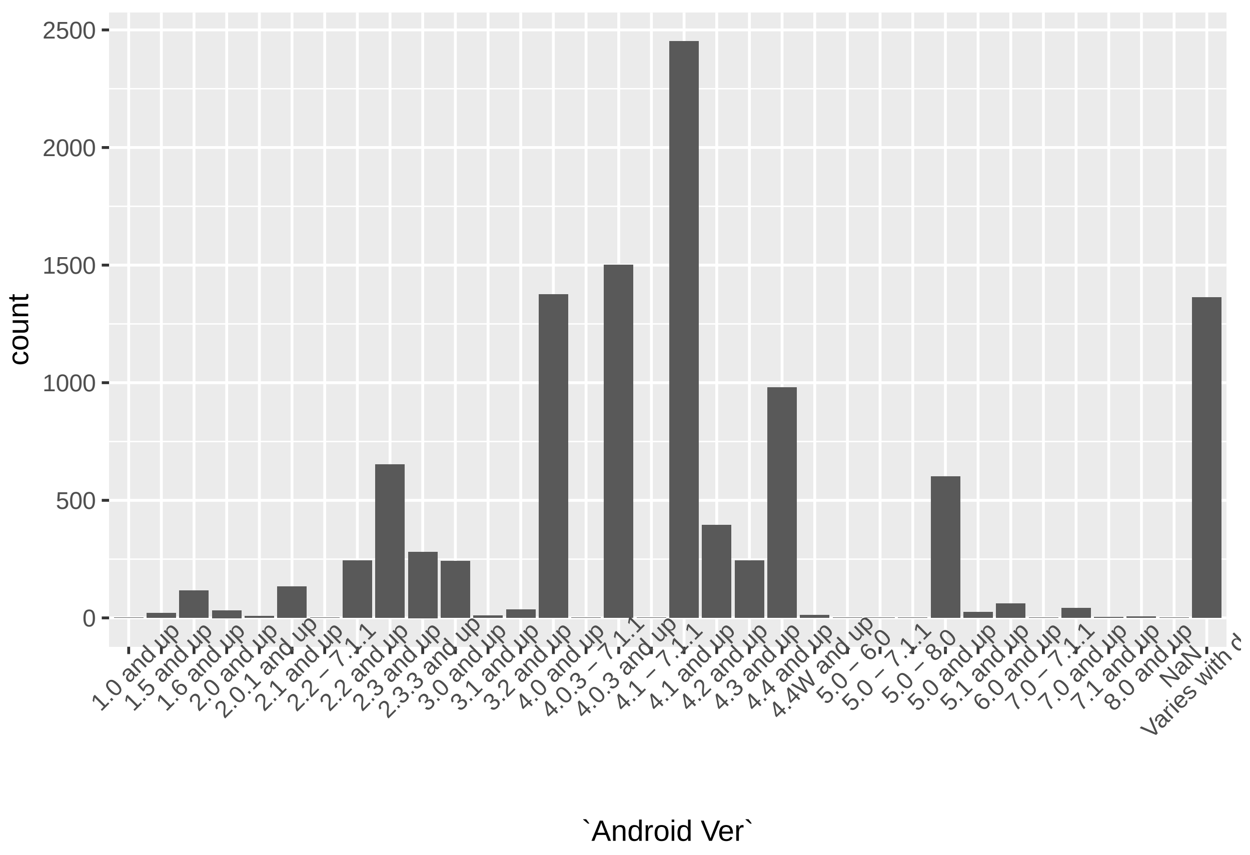
1. What is the order by which the columns are sorted? **Lexicographic (abc. . .)**

Build a chart that shows which android versions are supported by each app.

Question 6:

1. What was the latest android version when the file was generated? **Probably 8.0+**
2. Would you say that some categories of `Android Ver` should be grouped together? **Depends. for example 4.0.03-7.1.1 and 4.1-7.7.7 might be a proper grouping.**.

```r
ggplot(google_play, aes(x = `Android Ver`)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 45))
```



In the following segment we will examine the relationship between the rating of an app and the number of users who rated it. Build a chart that will help you examine this relationship.

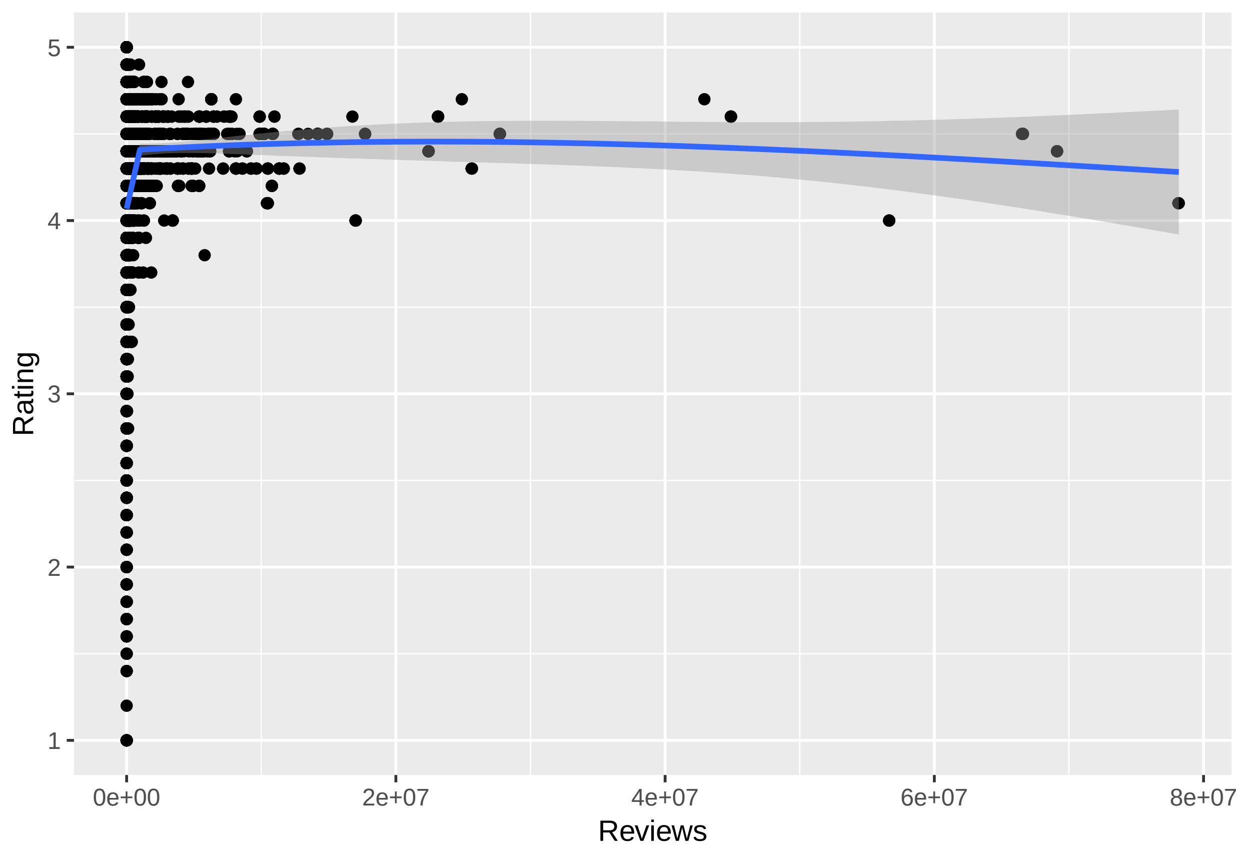
Hint: use a `geom_point()` with x as number of reviews and y as the average rank.

```r
ggplot(google_play, aes(x = Reviews, y = Rating)) +
   geom_point() +
   stat_smooth(method = "auto")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 1474 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1474 rows containing missing values (geom_point).
```

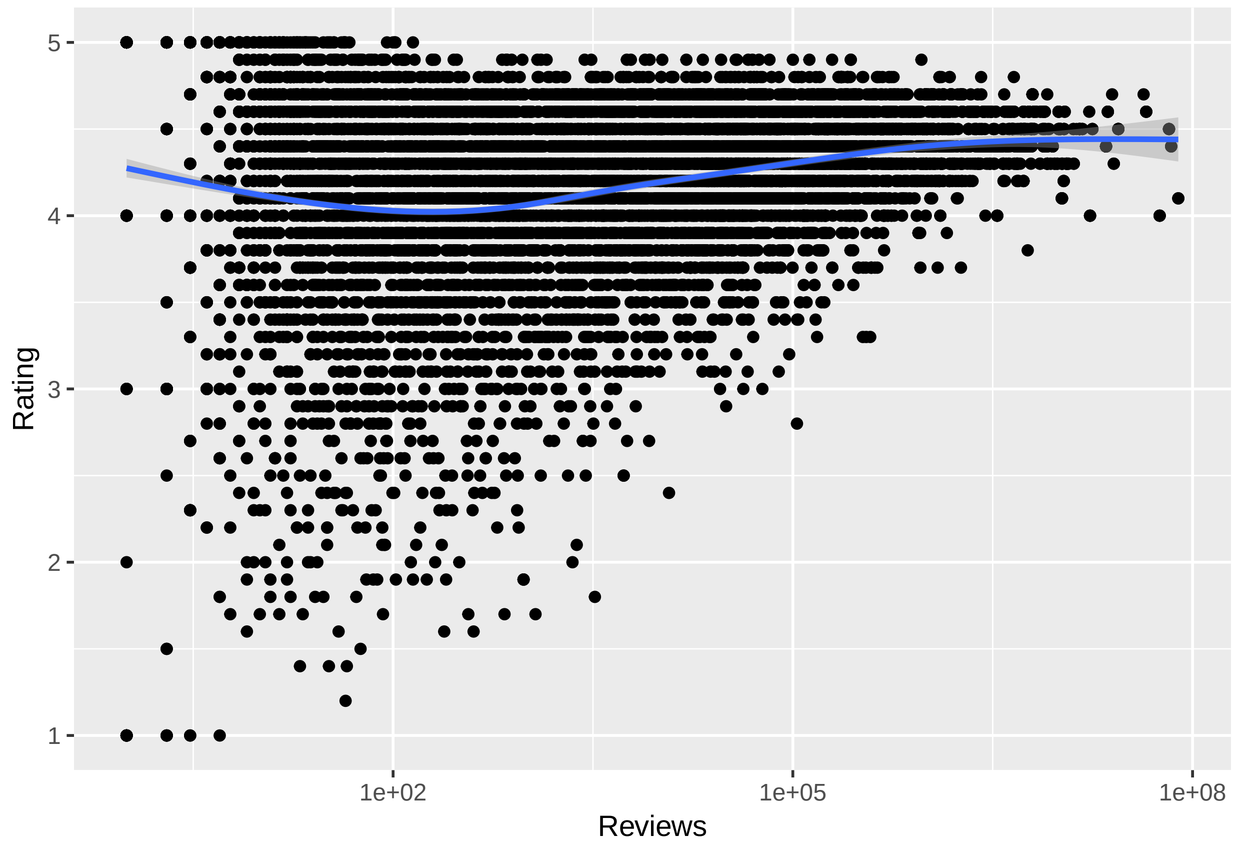


Question 7:

1. In the documentation of `stat_smooth` there are a number of smoothing methods. What did you choose and why? **Sometimes its a good guess to go with the default, in this case "auto". If that doesn't make sense - keep exploring. Loess is too sensitive and yields really unreasonable results. "lm" can also work, but the real deal is coming right up...**
2. Do you see anything wrong with the chart? **Very few observations on the higher end of Reviews, but these are "significant" apps. Currently the graph doesn't make any sense.**
3. Bonus: is there a transformation which you can use on the x-axis, for the chart to make more sense? What is it? **Use log(reviews) or use ggplot2's scale function for a log.**

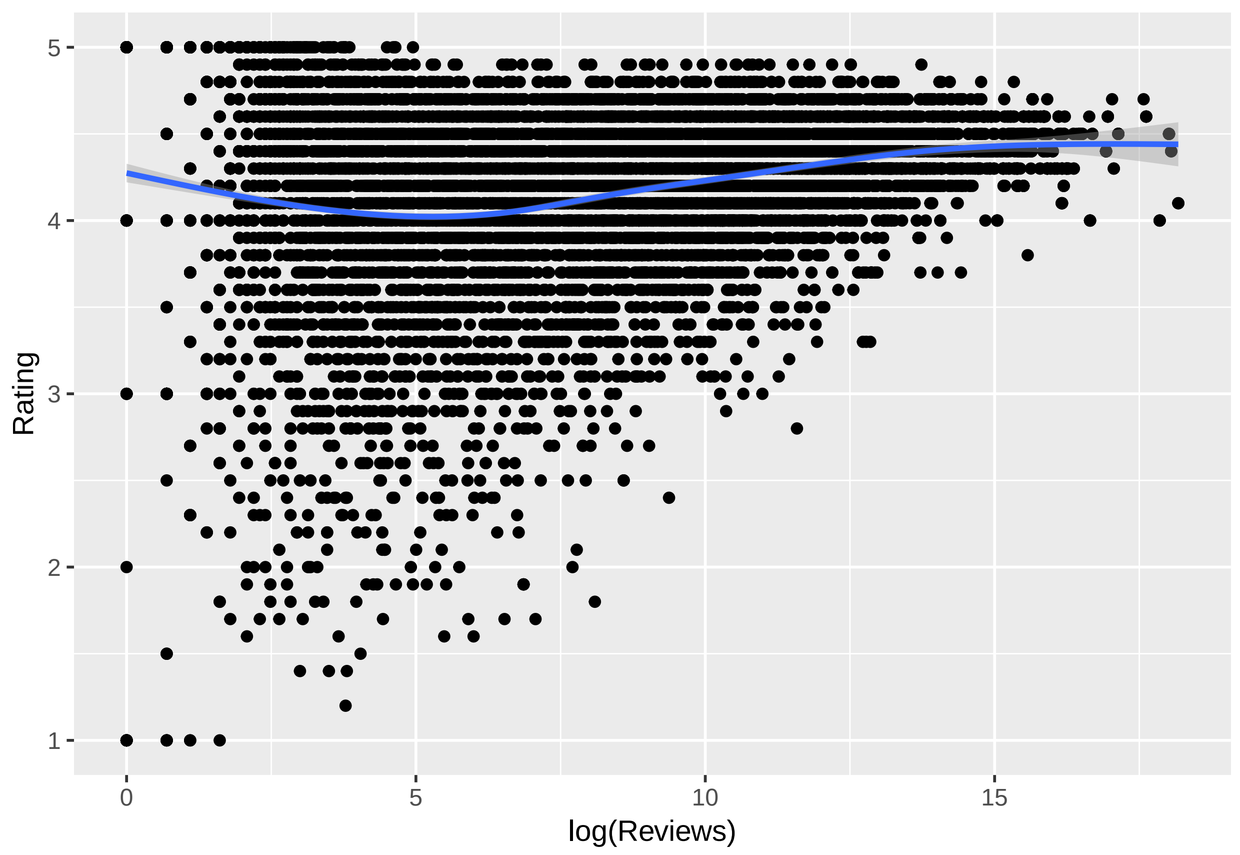
```r
# like this:
ggplot(google_play, aes(x = Reviews, y = Rating)) +
   geom_point() +
   stat_smooth(method = "auto") +
   scale_x_log10()
```

```
## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Transformation introduced infinite values in continuous x-axis

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 1474 rows containing non-finite values (stat_smooth).
## Warning: Removed 1474 rows containing missing values (geom_point).
```



```
# this will also work. spot the differences:
ggplot(google_play, aes(x = log(Reviews), y = Rating)) +
  geom_point() +
  stat_smooth(method = "auto")
```

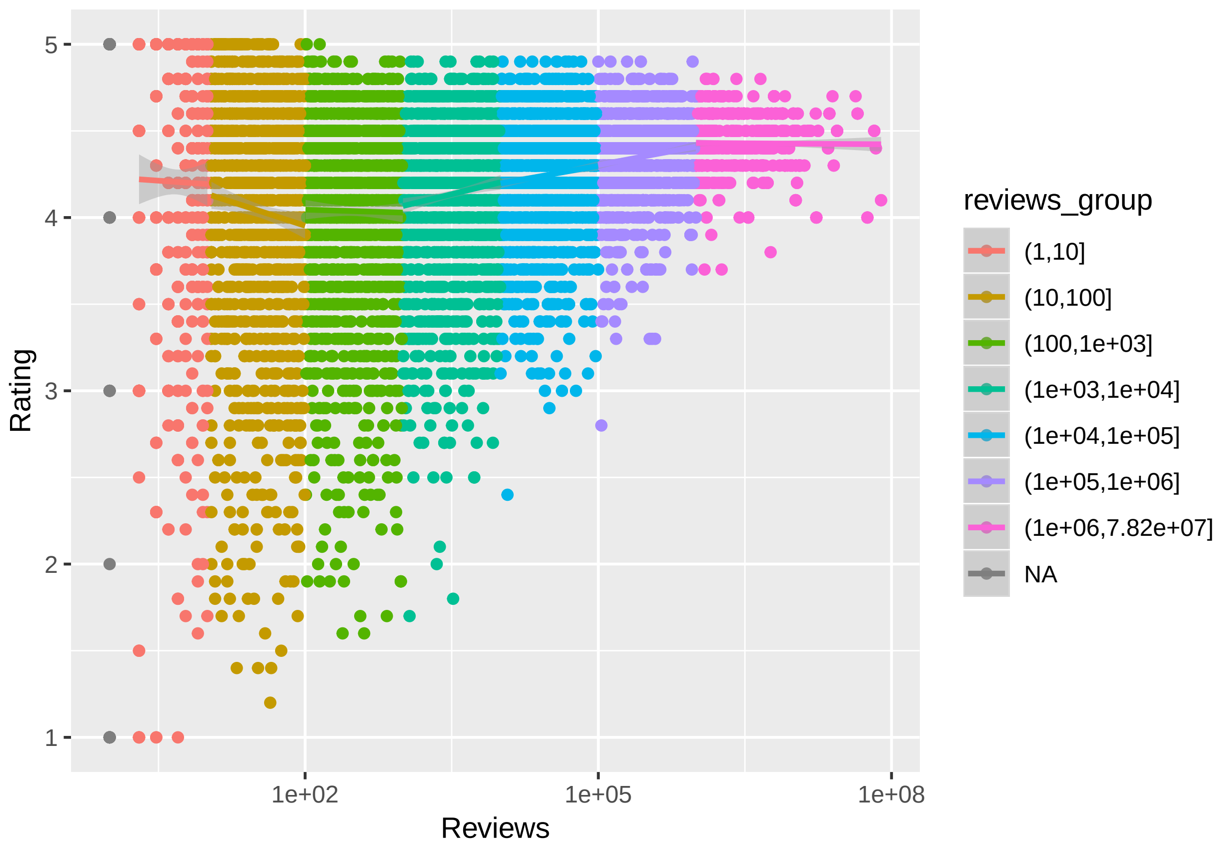
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 1474 rows containing non-finite values (stat_smooth).

## Warning: Removed 1474 rows containing missing values (geom_point).
```



Question 8: Use the function cut to split the number of reviews to five different groups. Generate a chart of rank versus reviews in which each new reviews-group is colored in different color.

```
# note the use of tidyverse here (%>%, mutate, etc.)
google_play_groups <- google_play %>%
   mutate(reviews_group = cut(Reviews, breaks = c(10^(0:6), max(Reviews) + 1)))

rating_reviews_grouped_chart <-
  ggplot(google_play_groups, aes(x = Reviews, y = Rating, color = reviews_group)) +
  geom_point() +
  stat_smooth(method = "lm") +
  scale_x_log10()

rating_reviews_grouped_chart
```

```
## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 1474 rows containing non-finite values (stat_smooth).

## Warning: Removed 1474 rows containing missing values (geom_point).
```



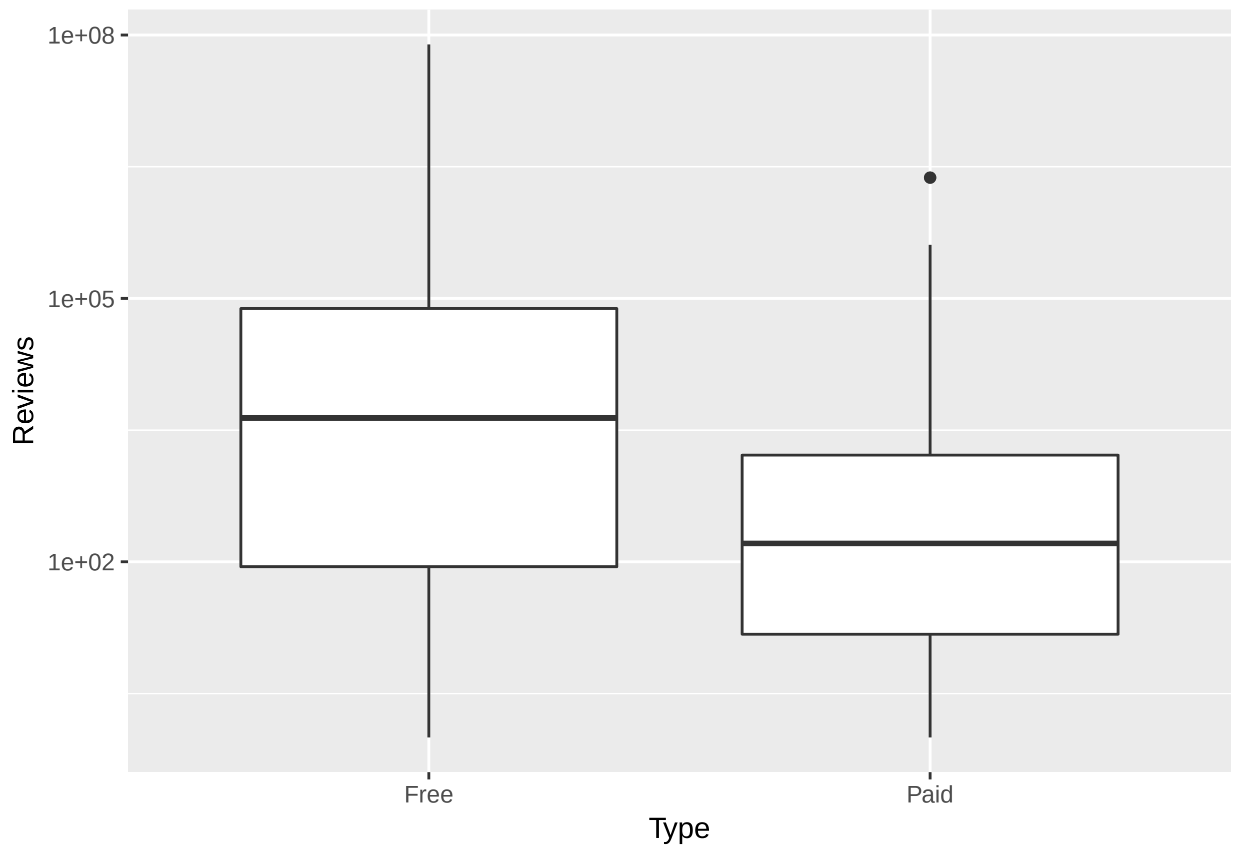
1. Do you identify any new relationships? **If we examine linear regression it looks like the relationship between rating and log(reviews) flips between the groups.**
2. Do specific ranges have stronger relationships than other ranges? **The coefficient of the lm model is smaller for Reviews < 10 and for reviews > 10^6.**

Question 9: Use a boxplot to compare the distribution of the number of reviews between free and paid apps

1. Do you see a redundent category on the x-axis (NaN)? try to get rid of it using `filter()` and `is.nan()`. Did it work? how would you get rid of it?
2. Did you notice the error message about infinite values when you plotted the data? what does the error message mean?
3. What does this comparison between paid and free tells you?
4. Use a `log()` transformation on the number of reviews, and re-plot. What do you see now?
5. Describe the difference between using `log(Reviews)` versus `scale_y_log()`.
6. Compare the distribution of average rank of free apps versus paid apps. What does this comparison tells you?

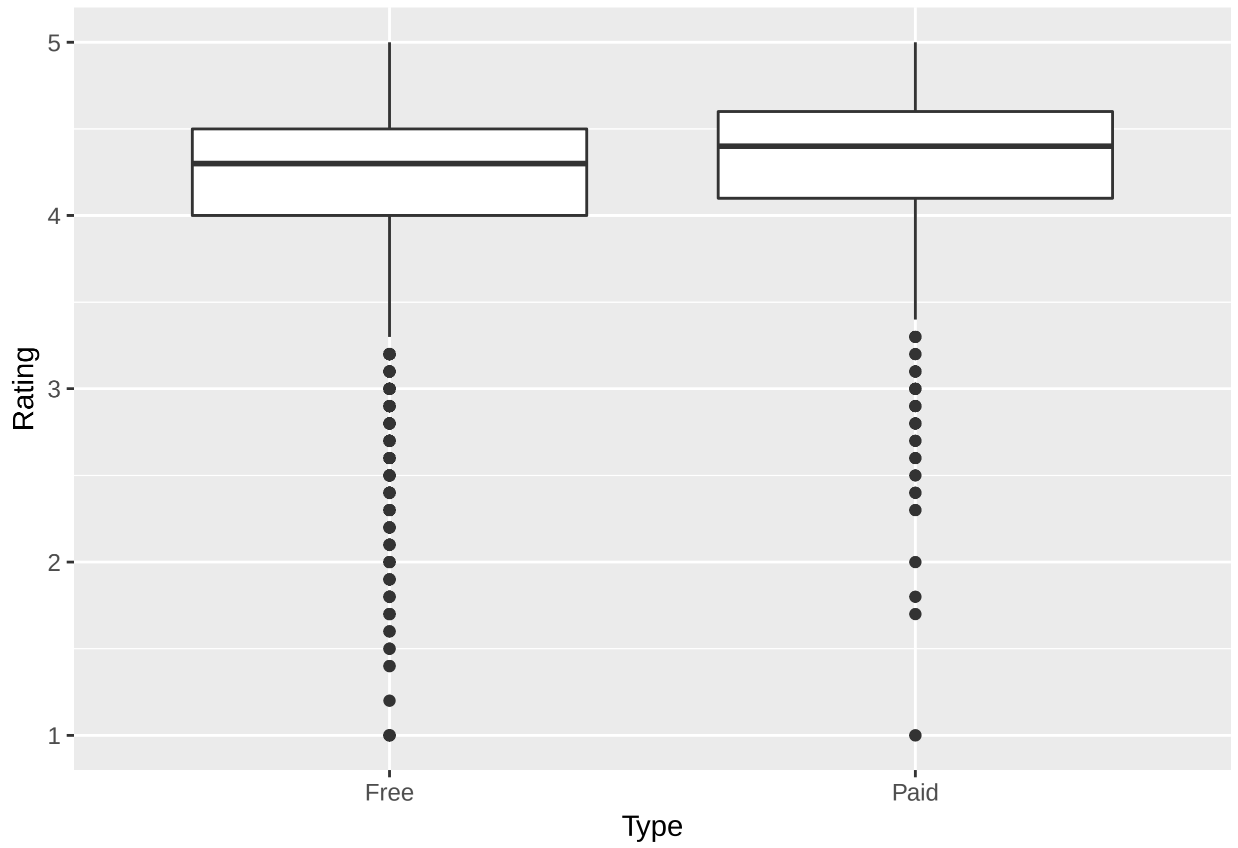
```r
review_price_chart <- ggplot(google_play %>% filter(Type %in% c("Free", "Paid")), aes(x = Type, y = Revi
  geom_boxplot() +
  scale_y_log10()
review_price_chart
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
## Warning: Removed 595 rows containing non-finite values (stat_boxplot).
```



```
rating_price_chart <- ggplot(google_play %>% filter(Type %in% c("Free", "Paid")), aes(x = Type, y = Rat
  geom_boxplot()
rating_price_chart
```

```
## Warning: Removed 1473 rows containing non-finite values (stat_boxplot).
```



Question 10: An alternative way to examine distributions is with a histogram. In this question we will examine the distribution of app size.

1. What is the problem with the app size variable?
2. Use `mutate`, `str_replace` (from package `stringr`) and the function `as.numeric` to convert it.
3. Use the functions `geom_histogram` and `geom_freqpoly`. Both has a parameter called `bins`. It's default value is 30. Try to examine different values. What does it affect? What happens when you choose the same value for both? what happens when you choose different values?
4. Where is the "main mass" of the distribution?

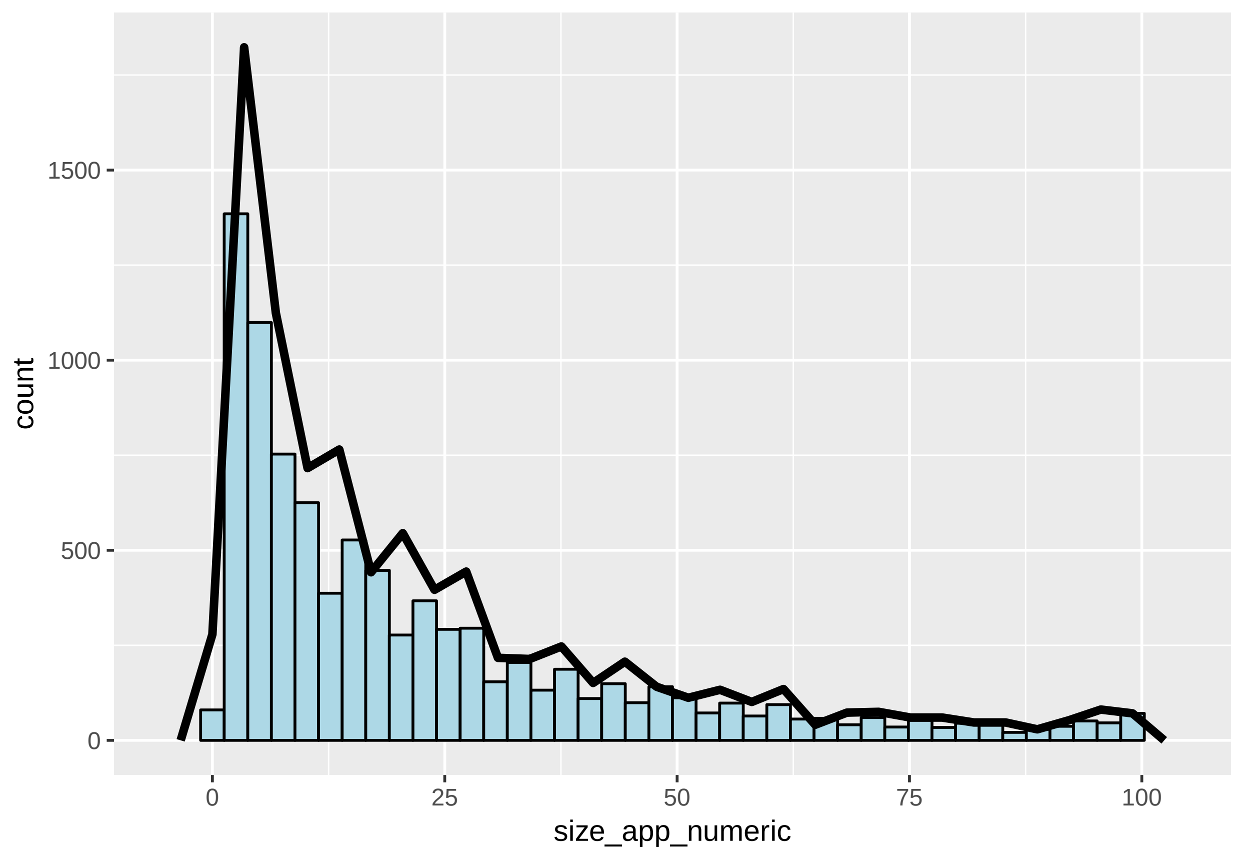
```
google_play <- google_play %>%
   mutate(size_app_numeric = as.numeric(str_replace(google_play$Size, "M", "")))
```

```
## Warning in evalq(as.numeric(str_replace(google_play$Size, "M", "")),
## <environment>): NAs introduced by coercion
```

```
size_app_chart <- ggplot(google_play, aes(size_app_numeric)) +
  geom_histogram(fill = "lightblue", color = "black", bins = 40) +
  geom_freqpoly(size = 1.5, bins = 30)

size_app_chart
```

```
## Warning: Removed 2011 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 2011 rows containing non-finite values (stat_bin).
```

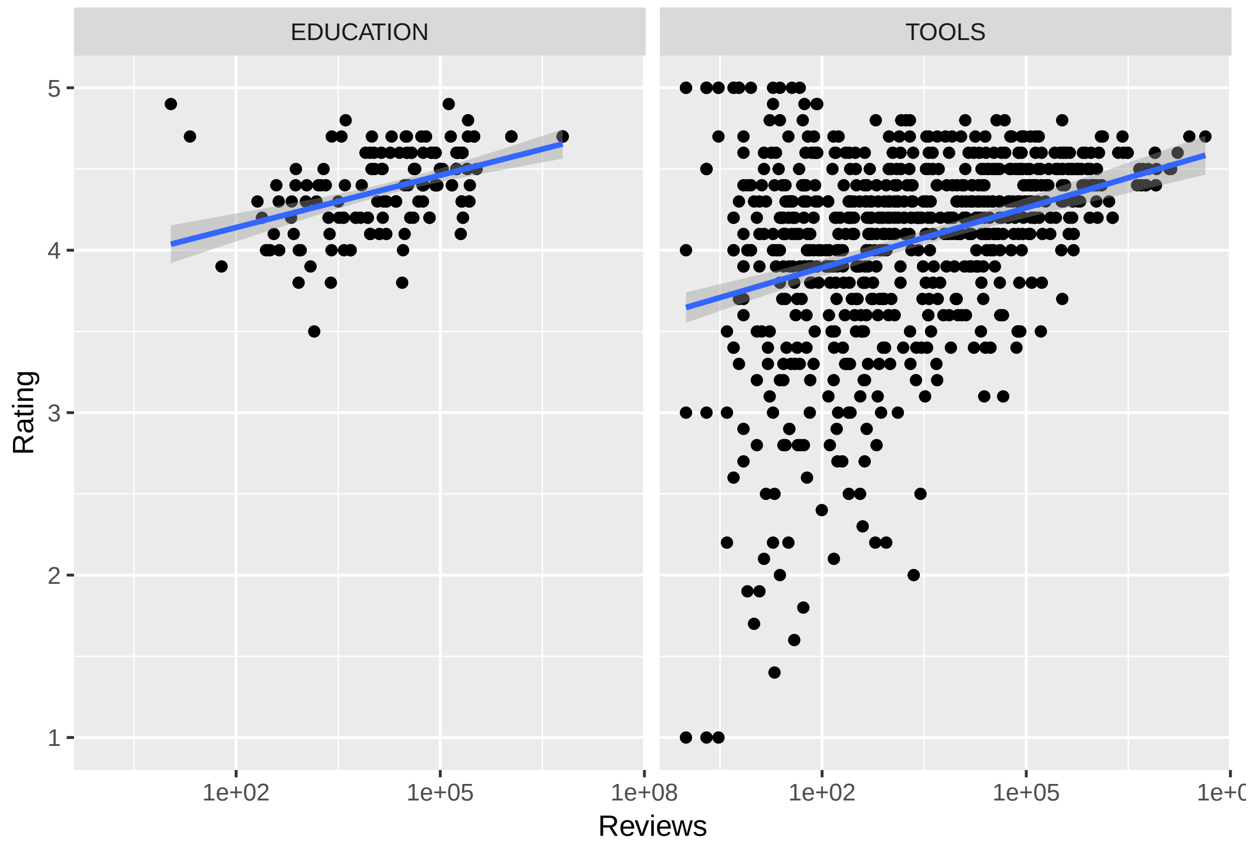


Question 11: One capability of `ggplot2` is splitting the charts with facets. To the chart you used of average rating versus log(reviews), add a facet (`facet_wrap`) by the variable Category, i.e., add `+ facet_wrap(~ ???)`.

1. What happened to the trend you identified earlier?
2. Compare the EDUCATION category versus TOOLS. What differences do you see and what is their meaning?

```r
rating_reviews_cont_rate <-
  ggplot(google_play_groups %>%
           filter(Category %in% c("EDUCATION", "TOOLS")),
         aes(x = Reviews, y = Rating)) +
  geom_point() +
  stat_smooth(method = "lm") +
  scale_x_log10() +
  facet_wrap( ~ Category)

rating_reviews_cont_rate
```

```
## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 110 rows containing non-finite values (stat_smooth).

## Warning: Removed 110 rows containing missing values (geom_point).
```



Checkpoint Alpha: If you reached here you have completed the first exercise. Hurray!.

A very useful trick - automated parameterized reports (with RMarkdown)

This trick is not directly related to plotting, but it seems appropriete to show it now, after we have done a few ggplots and now want to automate our process.

Suppose that there are difference moderators that are in charge of each "google play category". You want to provide each moderator with its own data but not others'. This means generating 33 RMarkdown reports (the number of categories), or does it?

Open file `02-Automated-report-sample.Rmd`. For now, skip the first bit that is in the —. It is called yaml (these are instructions to the RMarkdown compiler).Review the code, mainly it is taken from the exercise you have just solved.

Knit the RMarkdown into an html.

Now, do the same but instead of clicking on knit, open the knit menu (small arrow next to the knit icon), and click *knit with parameters*. Examin what options you have and look back at the *yaml* to see how they correspond to the *yaml* settings. If this didn't work, you might need to install `shiny` using: `install.packages("shiny")`.

To automate the knitting process you can use the following command (try to run it for category "BUSINESS" with app size > 5):

```r
rmarkdown::render(input = "../02-Automated-report-sample.Rmd",
                  params = list(app_category = "BUSINESS", app_size = 5),
                  output_file = "tmp_output.html")
```

```
##
##
## processing file: 02-Automated-report-sample.Rmd

##
  |
  |                                                                 |   0%
  |
  |...........                                                      |  17%
##   ordinary text without R code
##
##
  |
  |......................                                           |  33%
## label: setup
##
  |
  |................................                                 |  50%
##    inline R code fragments
##
##
  |
  |...........................................                      |  67%
## label: read file
##
  |
  |......................................................           |  83%
##   ordinary text without R code
##
##
  |
  |.................................................................| 100%
## label: plot something

## output file: 02-Automated-report-sample.knit.md

## "C:/PROGRA~1/Pandoc/pandoc" +RTS -K512m -RTS 02-Automated-report-sample.utf8.md --to html4 --from ma

##
## Output created: tmp_output.html
```

When we will learn the `purrr` package, you will see how you can create a script that will render all the categories there are with just a few lines of code.

We now go back to flow of plotting and discuss how ggplot2 protects you against shooting yourself in the leg.

The double y-axis paradigm

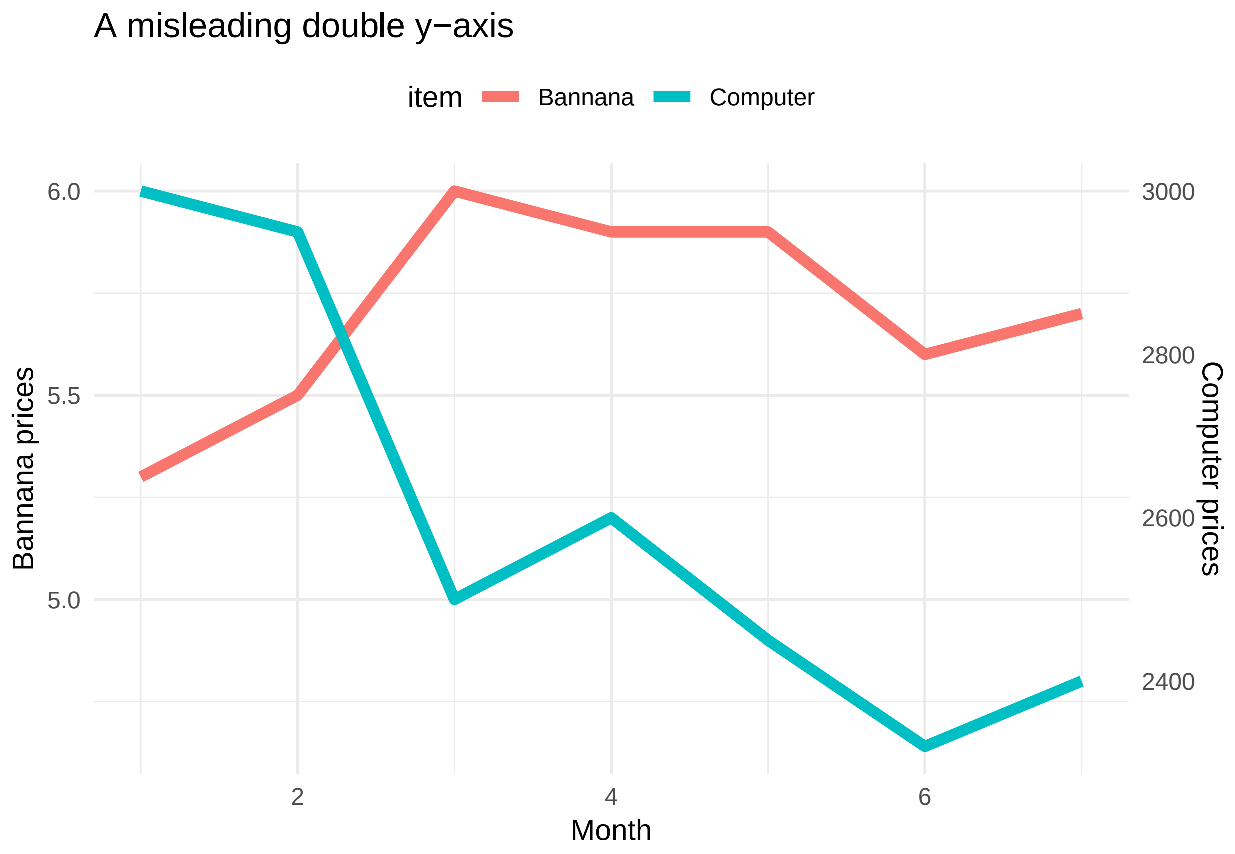
Double y-axis can be useful sometimes but can also be misleading. `ggplot2` makes it "hard" to generate a double y-axis, with a sound philosophy behind. See Hadley's response:

It's not possible in `ggplot2` because I believe plots with separate y scales (not y-scales that are transofrmations of each other) are fundamentally flawed…

Let's try to analyze this.

Question 1: In the following example, what is the relationship between Bannana prices and Computer prices? what is the meaning of their intersection around the second month? **We are manipulating the transformation to create whatever we want, but the intersection has no real meaning.**

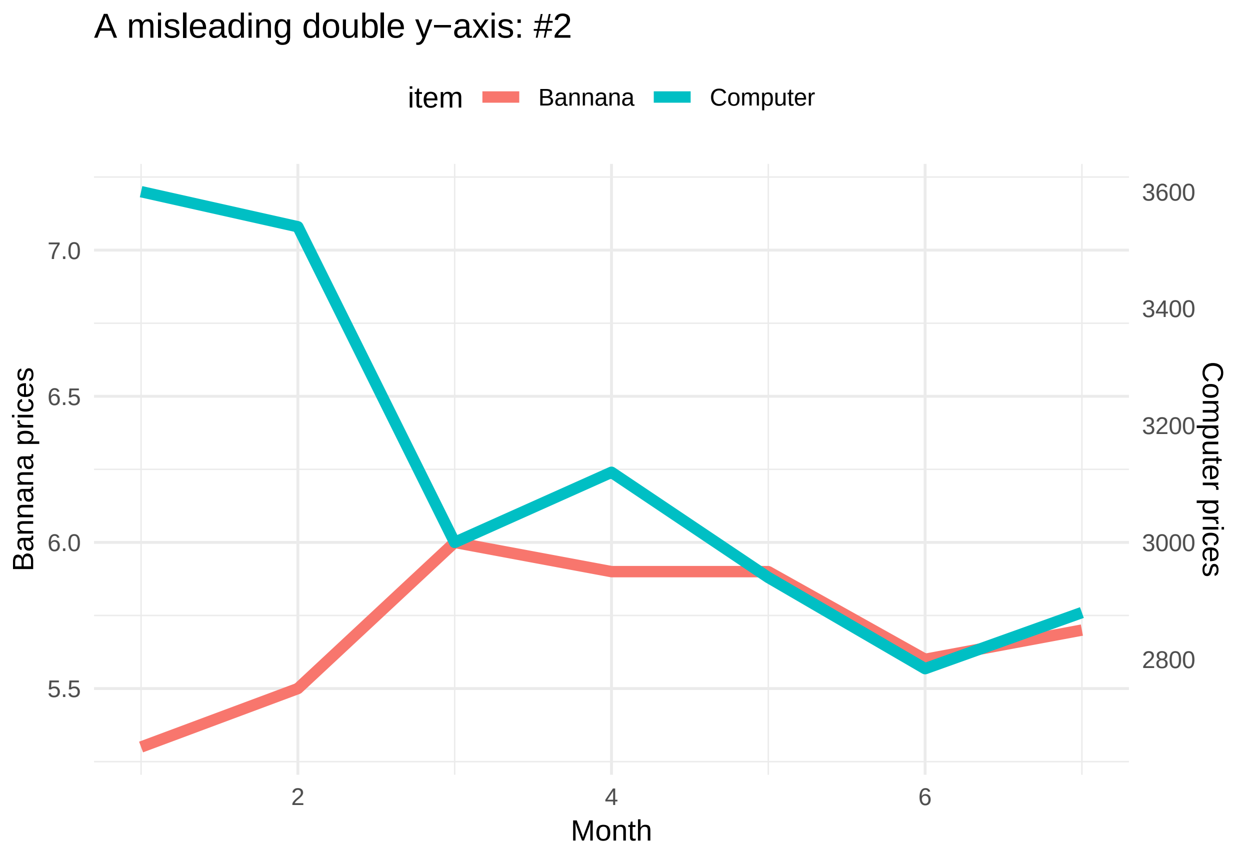
```r
sales <- tribble(
  ~Month, ~Bannana, ~Computer,
  1, 5.3, 3000,
  2, 5.5, 2950,
  3, 6, 2500,
  4, 5.9, 2600,
  5, 5.9, 2450,
  6, 5.6, 2320,
  7, 5.7, 2400
) %>%
  gather(item, price, -Month) %>%
  mutate(price = ifelse(item == "Computer", price/3000*6, price))

ggplot(sales, aes(x = Month, y = price, color = item)) +
  geom_line(size = 2) +
  scale_y_continuous("Bannana prices", sec.axis = sec_axis(~ .*3000/6, name = "Computer prices")) +
  theme_minimal() +
  theme(legend.position = "top") +
  ggtitle("A misleading double y-axis")
```



Question 2: Now examine the following chart. Non of the numbers or values changed, just the transfor-

mation that positions the computer's axis. The two lines seem to coinside more, but what does that mean? **Absolutely nothing**

```r
sales <- tribble(
  ~Month, ~Bannana, ~Computer,
  1, 5.3, 3000,
  2, 5.5, 2950,
  3, 6, 2500,
  4, 5.9, 2600,
  5, 5.9, 2450,
  6, 5.6, 2320,
  7, 5.7, 2400
) %>%
  gather(item, price, -Month) %>%
  mutate(price = ifelse(item == "Computer", price/2500*6, price))

ggplot(sales, aes(x = Month, y = price, color = item)) +
  geom_line(size = 2) +
  scale_y_continuous("Bannana prices", sec.axis = sec_axis(~ .*3000/6, name = "Computer prices")) +
  theme_minimal() +
  theme(legend.position = "top") +
  ggtitle("A misleading double y-axis: #2")
```



Question 3: There is one case in which a double y-axis is OK (and may even contribute).

1. When would using a dual y-axis be beneficial? **When we are showing a transformation of the**

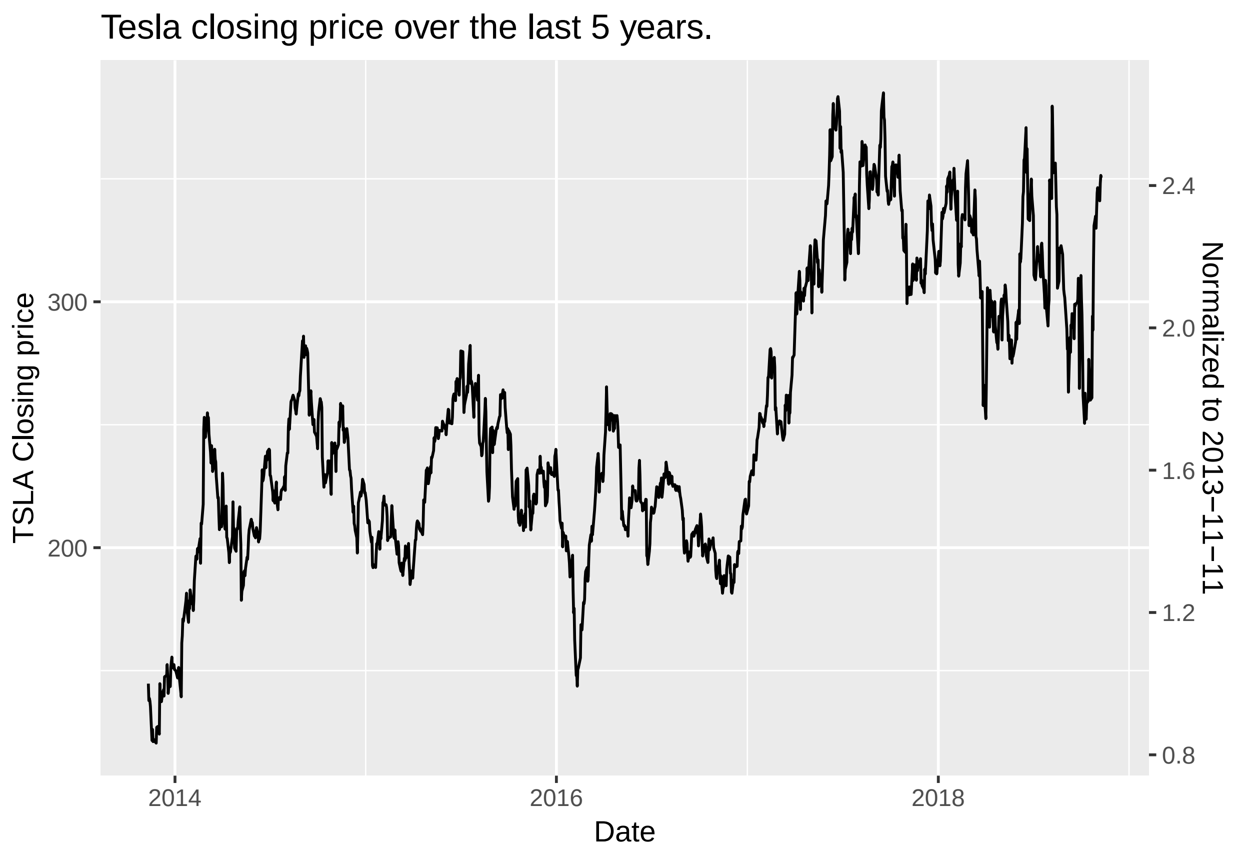
original axis, unrelated to the data itself. e.g., currency, Metric/US, Celsius/fahrenheit, etc.

2. Download (or make up) relevant data and illustrate it via a `ggplot2` chart with the `sec.axis` function/argument.

```
tsla <- read_csv("https://raw.githubusercontent.com/adisarid/Riskified_training/master/datasets/TSLA.cs
```

```
## Parsed with column specification:
## cols(
##   Date = col_date(format = ""),
##   Open = col_double(),
##   High = col_double(),
##   Low = col_double(),
##   Close = col_double(),
##   `Adj Close` = col_double(),
##   Volume = col_double()
## )
```

```
ggplot(tsla, aes(Date, Close)) +
   geom_line() +
   scale_y_continuous("TSLA Closing price", sec.axis = sec_axis(trans = ~./144.699997, "Normalized to 20
   ggtitle("Tesla closing price over the last 5 years.")
```

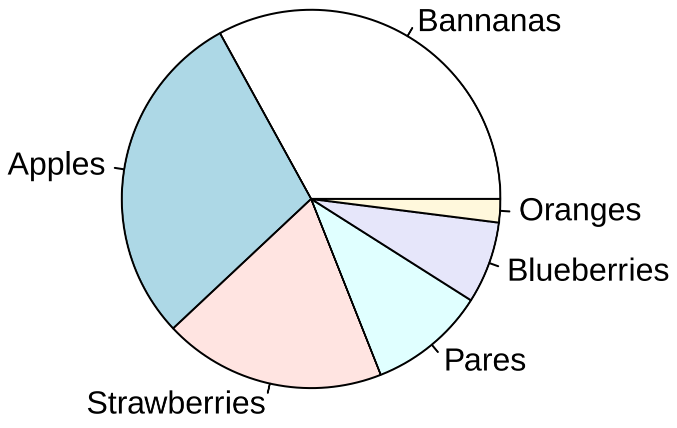


Pie charts

The problem with pie charts is that they require much more effort. They render one-dimentional data on a two dimentional plane, so in fact pie charts **do not exist in `ggplot2`!** (there are some blog posts talking about how to make a pie chart with `coord_polar`, and they are really long).

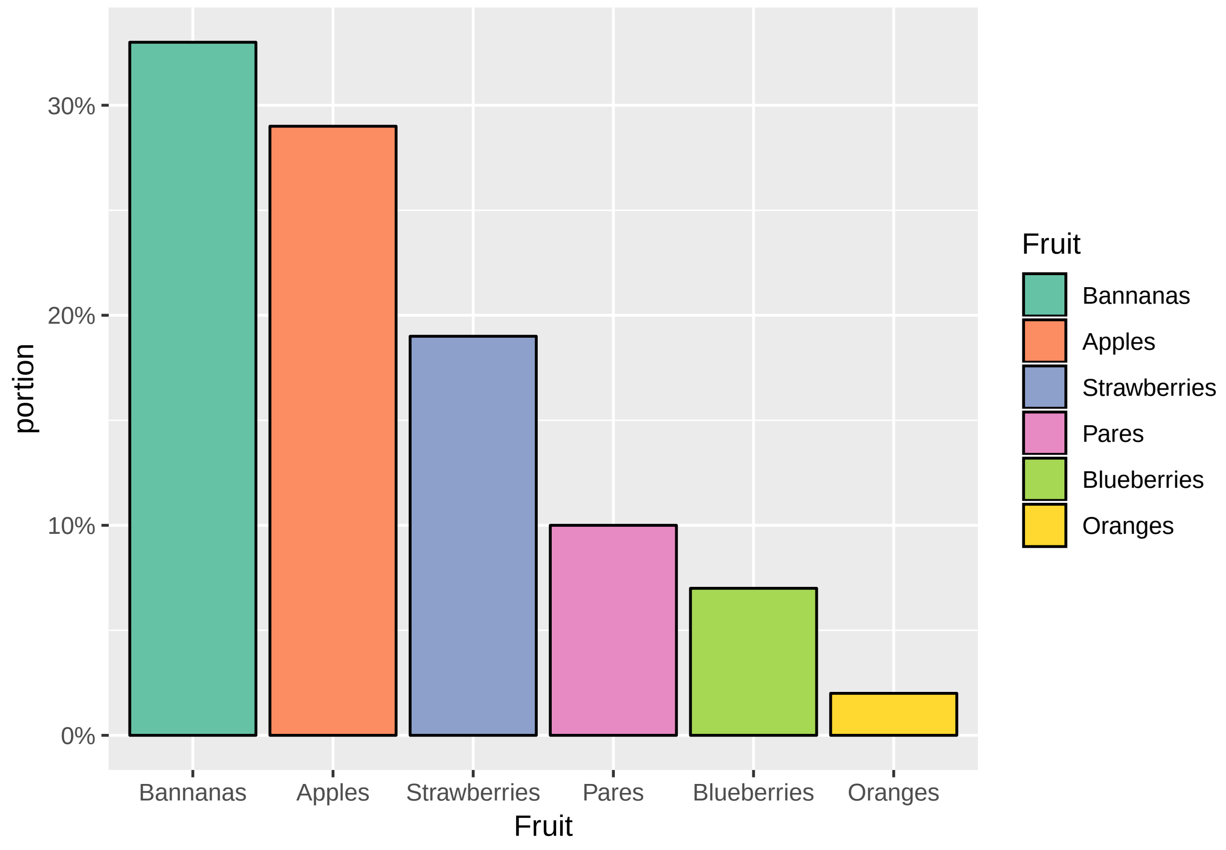
Base-r though has pie charts. DO NOT TRY THIS AT HOME:

```r
fruits <- tribble(
   ~fruit, ~portion,
   "Bannanas", 0.33,
   "Apples", 0.29,
   "Strawberries", 0.19,
   "Pares", 0.1,
   "Blueberries", 0.07,
   "Oranges", 0.02
)

# DO NOT TRY THIS AT HOME
pie(fruits$portion, labels = fruits$fruit)
```



Question 1: How would you plot the `fruits` data in `ggplot2`'s ecosystem? (think bars) and do that.

```r
ggplot(fruits, aes(x = fct_inorder(fruit), y = portion, fill = fct_inorder(fruit))) +
   geom_col(color = "black") +
   scale_fill_brewer(palette = "Set2") +
   scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
   xlab("Fruit") +
   guides(fill = guide_legend("Fruit"))
```



```r
# the "fill" is just for fun. It is redundent.
```

Exercise 3: The Telco customer churn

This exercise is based on a file downloaded from the IBM website (it's probably synthetic data, but it is still interesting to examine). The original file is available here, but you can also load it from our github repo.

```r
telco_churn <- read_csv("https://raw.githubusercontent.com/adisarid/Riskified_training/master/datasets/W
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   SeniorCitizen = col_double(),
##   tenure = col_double(),
##   MonthlyCharges = col_double(),
##   TotalCharges = col_double()
## )

## See spec(...) for full column specifications.
```

```r
glimpse(telco_churn)
```

```
## Observations: 7,043
## Variables: 21
## $ customerID       <chr> "7590-VHVEG", "5575-GNVDE", "3668-QPYBK", "77...
## $ gender           <chr> "Female", "Male", "Male", "Male", "Female", "...
## $ SeniorCitizen    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ Partner          <chr> "Yes", "No", "No", "No", "No", "No", "No", "N...
## $ Dependents       <chr> "No", "No", "No", "No", "No", "No", "Yes", "N...
## $ tenure           <dbl> 1, 34, 2, 45, 2, 8, 22, 10, 28, 62, 13, 16, 5...
## $ PhoneService     <chr> "No", "Yes", "Yes", "No", "Yes", "Yes", "Yes"...
## $ MultipleLines    <chr> "No phone service", "No", "No", "No phone ser...
## $ InternetService  <chr> "DSL", "DSL", "DSL", "DSL", "Fiber optic", "F...
## $ OnlineSecurity   <chr> "No", "Yes", "Yes", "Yes", "No", "No", "No", ...
## $ OnlineBackup     <chr> "Yes", "No", "Yes", "No", "No", "No", "Yes", ...
## $ DeviceProtection <chr> "No", "Yes", "No", "Yes", "No", "Yes", "No", ...
## $ TechSupport      <chr> "No", "No", "No", "Yes", "No", "No", "No", "N...
## $ StreamingTV      <chr> "No", "No", "No", "No", "No", "Yes", "Yes", "...
## $ StreamingMovies  <chr> "No", "No", "No", "No", "No", "Yes", "No", "N...
## $ Contract         <chr> "Month-to-month", "One year", "Month-to-month...
## $ PaperlessBilling <chr> "Yes", "No", "Yes", "No", "Yes", "Yes", "Yes"...
## $ PaymentMethod    <chr> "Electronic check", "Mailed check", "Mailed c...
## $ MonthlyCharges   <dbl> 29.85, 56.95, 53.85, 42.30, 70.70, 99.65, 89....
## $ TotalCharges     <dbl> 29.85, 1889.50, 108.15, 1840.75, 151.65, 820....
## $ Churn            <chr> "No", "No", "Yes", "No", "Yes", "Yes", "No", ...
```

Question 1: Using a scatter plot (`geom_point`), examine the distribution of TotalCharges as a function of tenure.

1. What can you deduce from the way the points are scattered?
2. Use facets to split the chart by contract, and add smoothing (`stat_smooth`). What does this tells you?

Question 2: Examine the distribution of monthly charges versus churn.

1. Is there a relationship between the monthly payments and the churn?
2. Now, add a chart which examines the monthly charges versus churtn but also separates the the internet service type. What do you deduce now that you see the new chart?

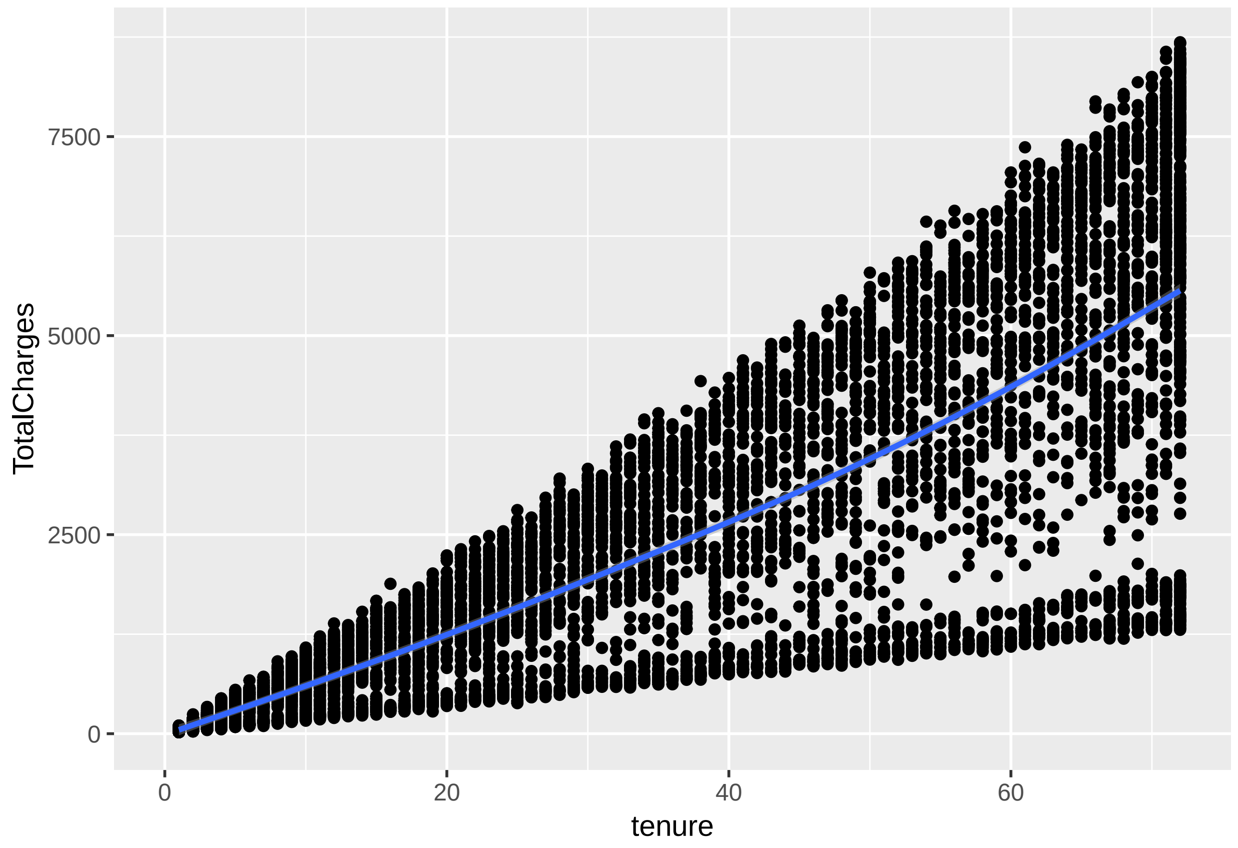
Question 3: Build a chart that shows the number of churning and remaining clients versus the type of services they received. Are you able to explain the phenomena in the last part better?

```r
ggplot(telco_churn, aes(x=tenure, y=TotalCharges)) + geom_point() + stat_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

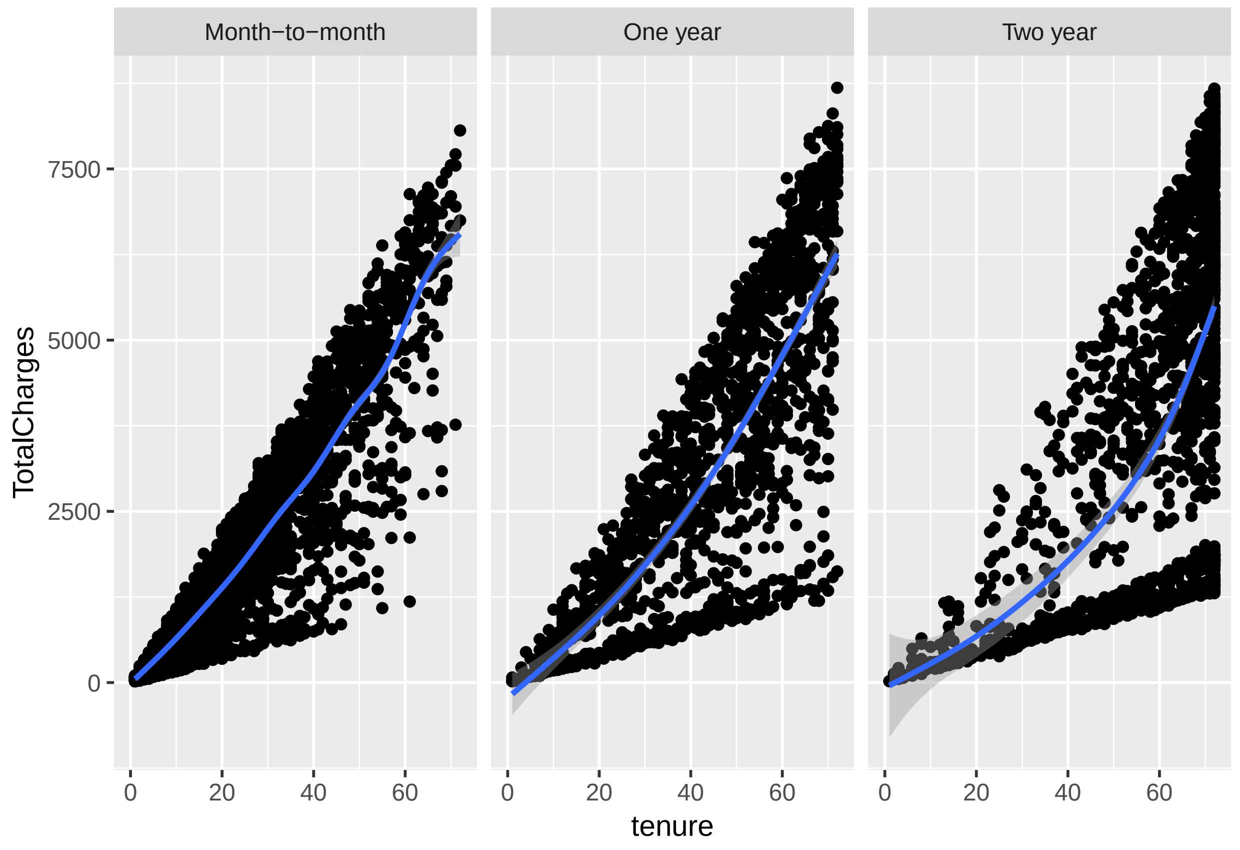
```
## Warning: Removed 11 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 11 rows containing missing values (geom_point).
```

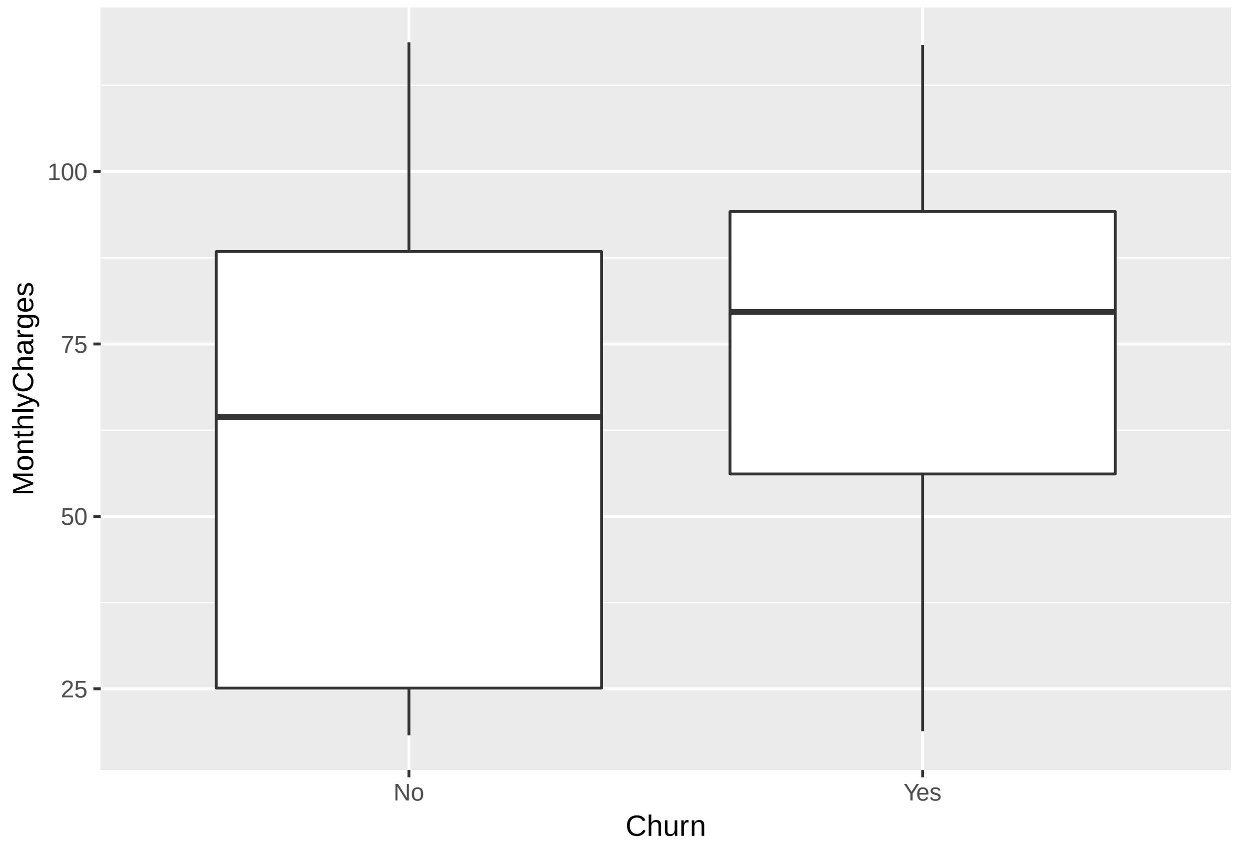


```
ggplot(telco_churn, aes(x=tenure, y=TotalCharges)) + geom_point() + facet_wrap(~ Contract) + stat_smoot
```

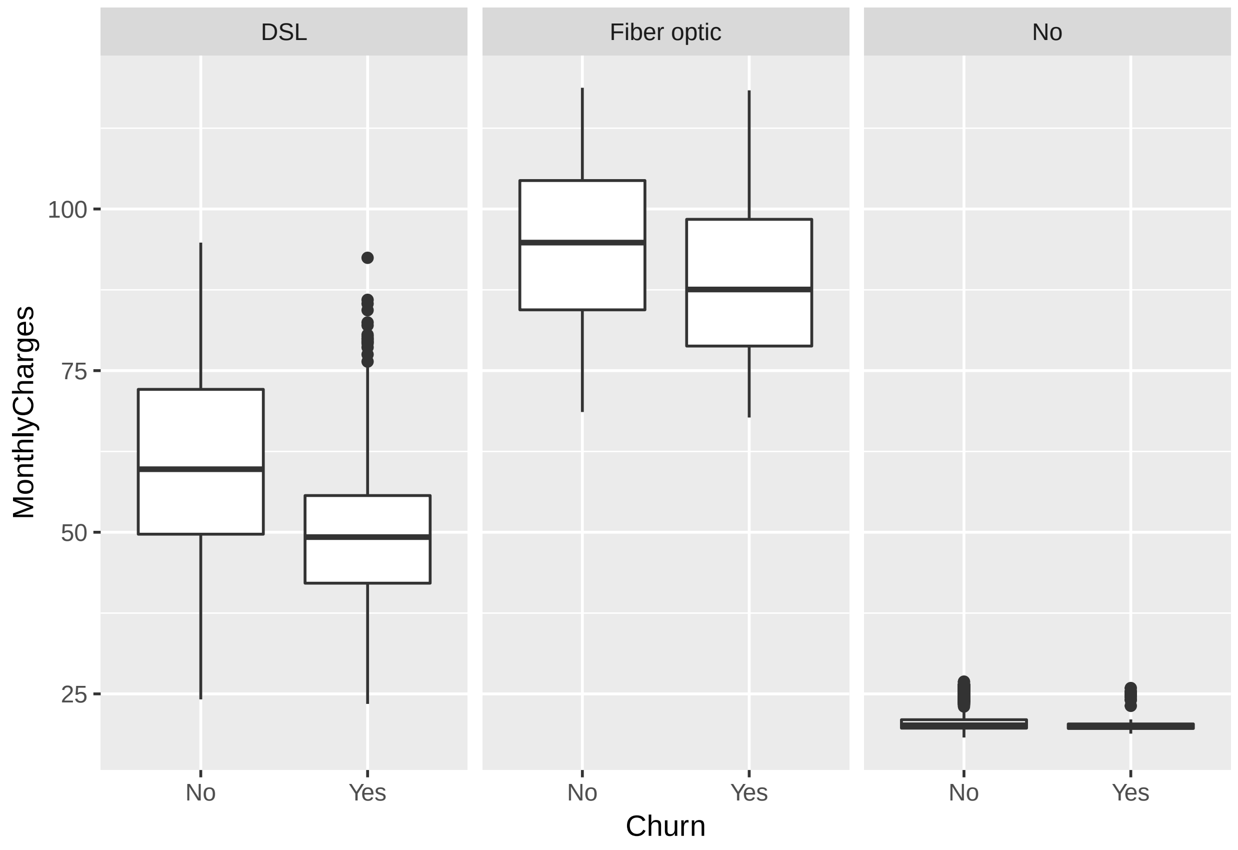
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

## Warning: Removed 11 rows containing non-finite values (stat_smooth).

## Warning: Removed 11 rows containing missing values (geom_point).
```



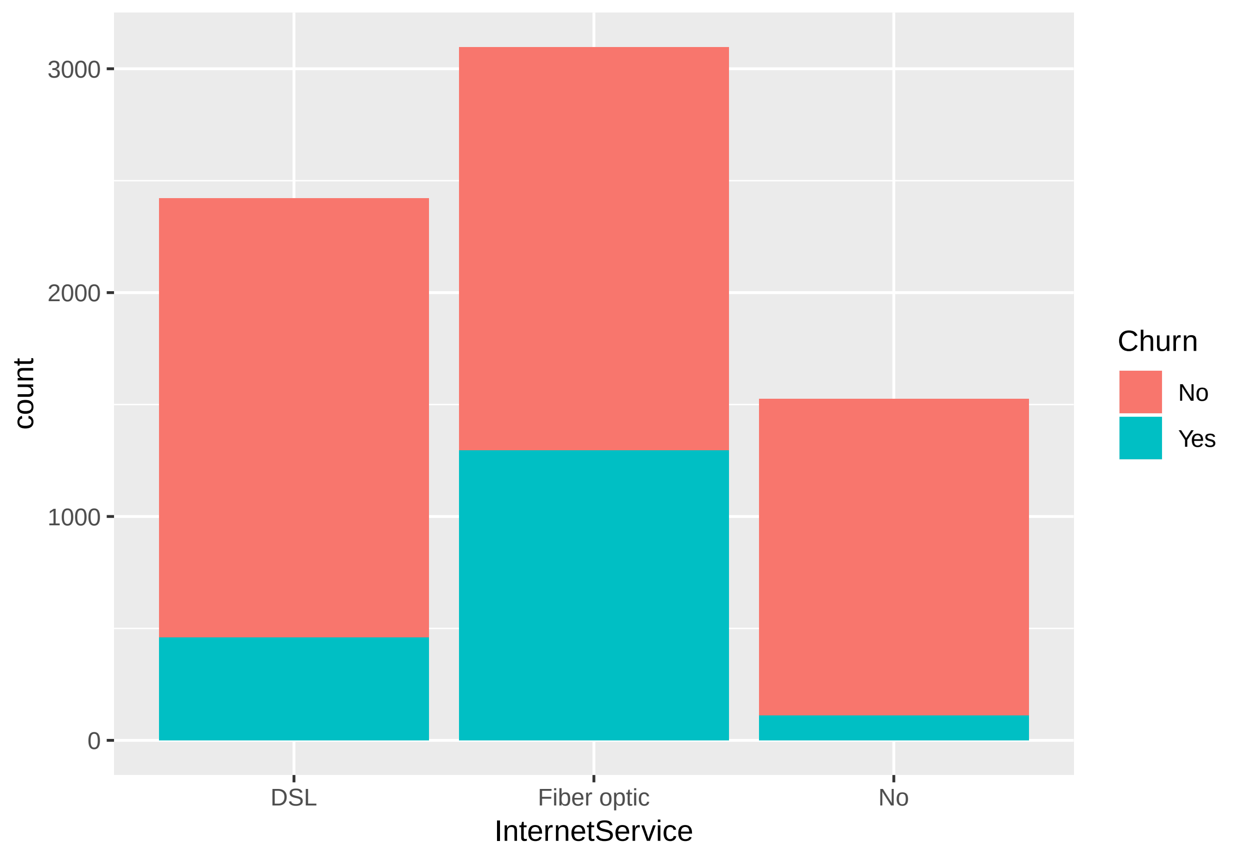
```r
ggplot(telco_churn, aes(x=Churn,y=MonthlyCharges)) + geom_boxplot()
```



```r
ggplot(telco_churn, aes(x=Churn,y=MonthlyCharges)) + geom_boxplot() + facet_wrap(~InternetService)
```



```r
ggplot(telco_churn, aes(x = InternetService, fill=Churn)) + geom_bar(position = "stack")
```



```
# The churn is much higher in fiber optics, and the monthly payments of fiber optics are higher.
# Hence, when we compare the monthly charges vs. churn it looks like you churn when you pay more,
# but in fact, when we examine the interaction of the internet service type,
# we see that actually churning customers paid less. The bias was created since we mixed the service ty

telco_churn %>%
  group_by(InternetService, Churn) %>%
  summarize(pymnt = mean(MonthlyCharges))
```

```
## # A tibble: 6 x 3
## # Groups:   InternetService [?]
##   InternetService Churn pymnt
##   <chr>           <chr> <dbl>
## 1 DSL             No     60.2
## 2 DSL             Yes    49.1
## 3 Fiber optic     No     93.9
## 4 Fiber optic     Yes    88.1
## 5 No              No     21.1
## 6 No              Yes    20.4
```