Lab 4 Graphing Data (Part 1)

B. Sosnovski

01/04/2022

```
knitr::opts_chunk$set(echo = TRUE)
```

Graphing Data (Part 1)

When we measure things, we get lots of numbers. Sometimes so many that your head explodes just thinking about them. One of the most helpful things you can do to begin to make sense of these numbers is to look at them in graphical form.

General Goals

In this lab, you will learn the following:

1. To make graphs of the qualitative data

Data

We will continue to use the data for NYC film permits (Film_Permits.csv) used in the previous lab instructions. We will also try to answer the earlier questions by graphing the data.

```
library(data.table)
nyc_films <-fread("Film_Permits.csv")
colnames(nyc_films)
```

```
##  [1] "Event ID"          "EventType"         "StartDateTime"
##  [4] "EndDateTime"       "EnteredOn"         "EventAgency"
##  [7] "ParkingHeld"       "Borough"           "CommunityBoard(s)"
## [10] "PolicePrecinct(s)" "Category"          "SubCategoryName"
## [13] "Country"           "ZipCode(s)"
```

We will also use the data containing information from 2020-2022 police-reported motor vehicle collisions in NYC (Motor_Vehicle_Collisions_Crashes.csv). The data is available here: https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95 (https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95)

Make Plots to answer questions

Where are the most film permits being requested?

Most of the film permits were requested for which of the five boroughs of NYC?

We can find out by plotting the data using a bar graph. But to plot the data, we first summarized it. In the previous lab, we learned to make frequency tables with the number of film permits made in each borough and then make different bars to represent the counts (the frequency of each borough).

Here is the code used in the previous lab (copied and pasted!):

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

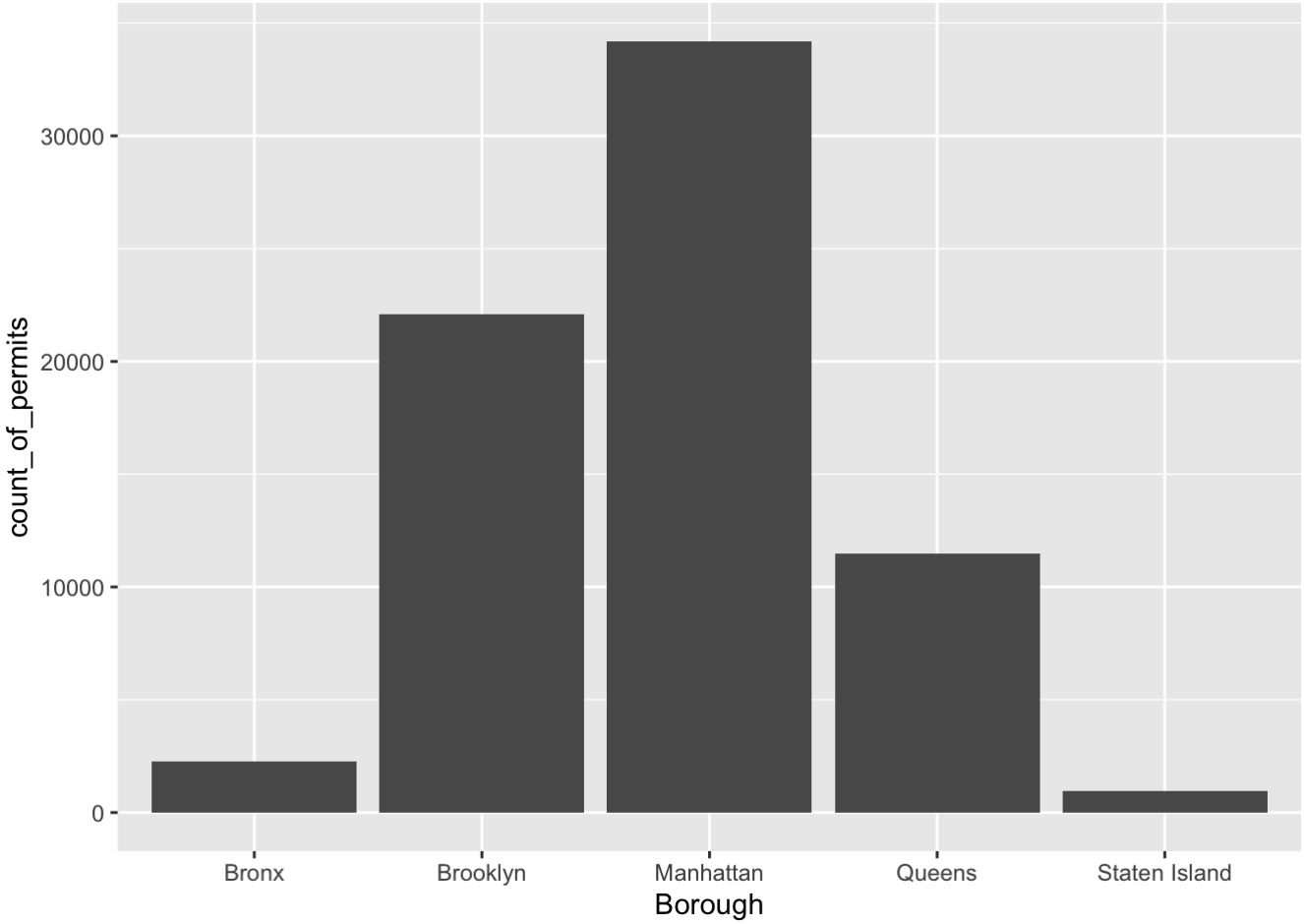
```
counts <- nyc_films %>%
          group_by(Borough) %>%
          summarize(count_of_permits = length(Borough))
counts
```

```
## # A tibble: 5 × 2
##   Borough       count_of_permits
##   <chr>                    <int>
## 1 Bronx                     2280
## 2 Brooklyn                 22087
## 3 Manhattan                34200
## 4 Queens                   11484
## 5 Staten Island              938
```

We can also produce plots using a fantastic package called `ggplot2`. It is very powerful once you get the hang of it, and when you do, you can make all sorts of interesting graphs. Here's the code to make the bar graph.

```
library(ggplot2)

ggplot(counts, aes(x = Borough, y = count_of_permits )) +
  geom_bar(stat="identity")
```



The bar graph shows that most film permits were requested for the borough of Manhattan.

Example

In your Lab's directory, there is a file called *Motor_Vehicle_Collisions_Crashes.csv*. Perform the following tasks:

1. Load the file into R, find out the columns' names and read the top lines of the file.
2. Graph the number of crashes per borough

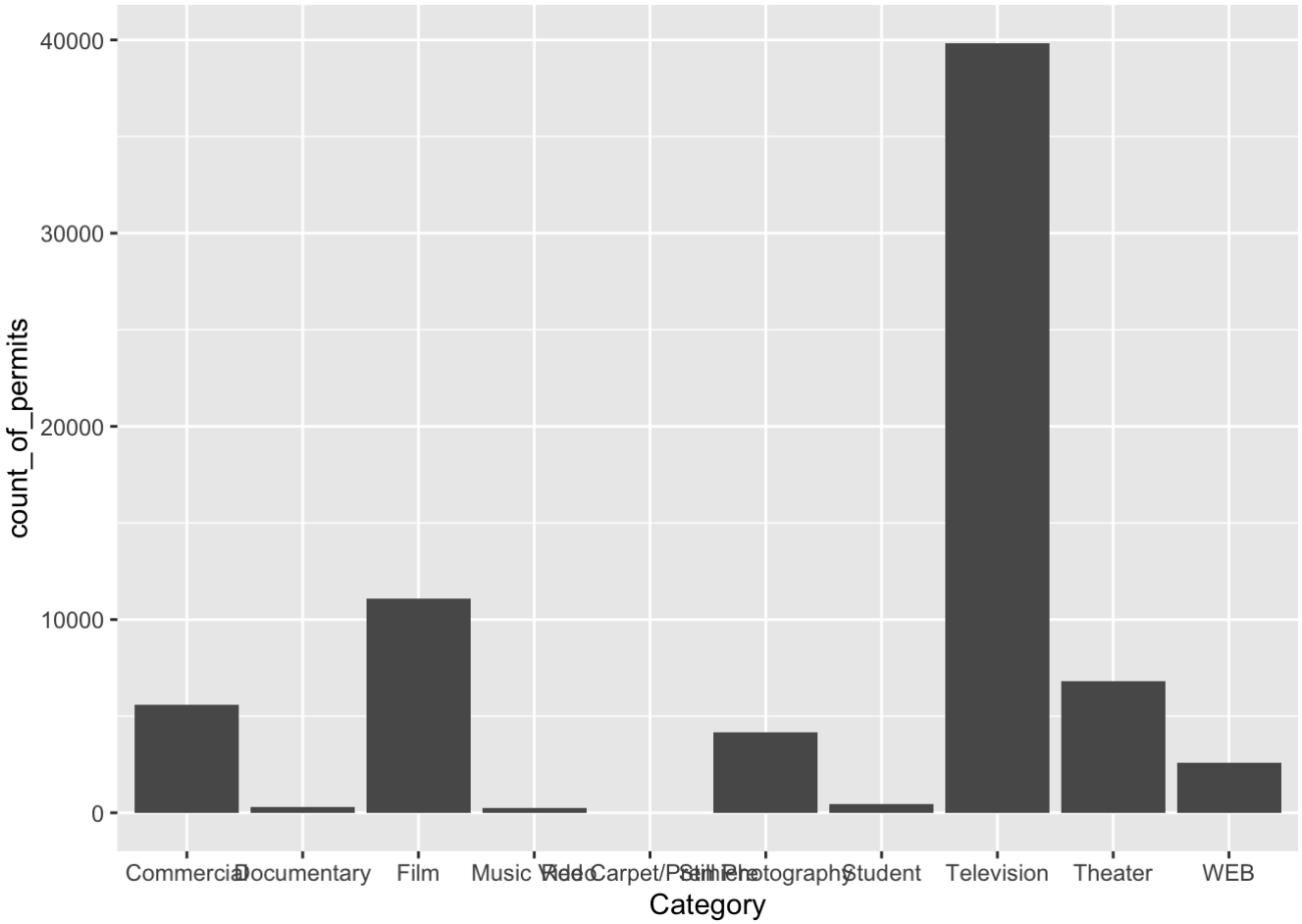
```
Enter your code below:
```

What kind of "films" were made, and what is the category?

Let's make the bar graph of the data regarding the variable `Category` .

```
counts <- nyc_films %>%
          group_by(Category) %>%
          summarize(count_of_permits = length(Category))

ggplot(counts, aes(x = Category, y = count_of_permits )) +
  geom_bar(stat="identity")
```



OK, this figure might look a bit weird because the labels on the bottom are running into each other. We'll fix that in a bit. First, let's notice the changes.

1. `Borough` was changed to `Category`. That was the main thing.
2. Note that none of the `library()` commands are used again, and I didn't re-run the very early code to get the data. R already has those things in its memory, so we don't need to do that again. If you ever clear the memory of R, you will need to reload those things.

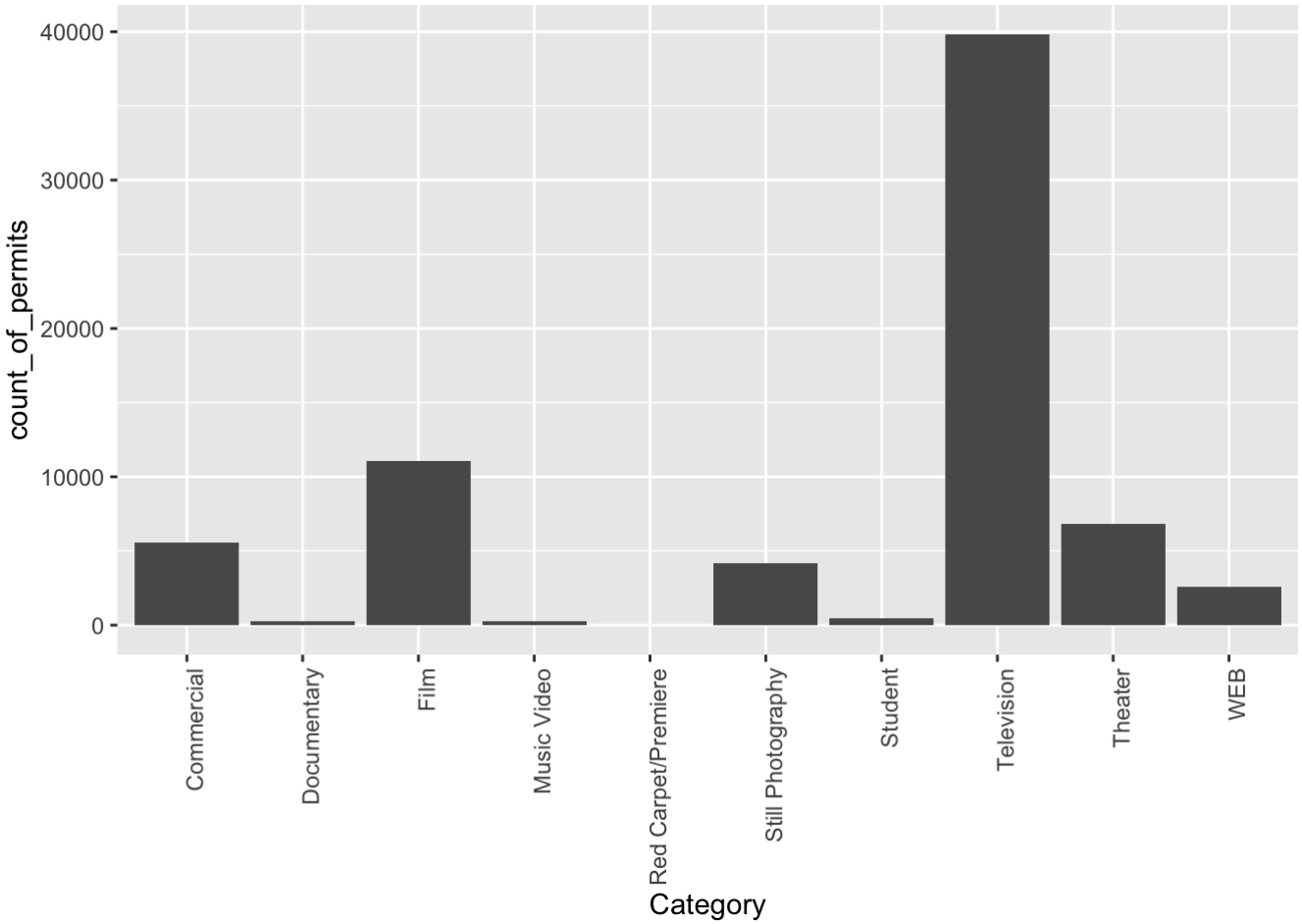
Fine, so how do we fix the graph? Good question. One way to find out is to search using Google.

Googling your questions is an excellent way to learn R. It's what everybody does these days…[goes to Google…].

The trick is to add the last line to the code. I just copy-pasted it from the solution found on stack overflow (https://stackoverflow.com/questions/1330989/rotating-and-spacing-axis-labels-in-ggplot2) (I hope you will become friends with stack overflow, there are many solutions there to all of your questions)

```
counts <- nyc_films %>%
          group_by(Category) %>%
          summarize(count_of_permits = length(Category))

ggplot(counts, aes(x = Category, y = count_of_permits )) +
  geom_bar(stat="identity")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



ggplot2 basics

Before we go further, let's check some basic properties of `ggplot2` to give you a sense of how it works. This will make more sense later with other labs, so come back here to remind yourself.

We'll do just a bit of basic, then move on to making more graphs by copying and pasting.

The ggplot function uses layers. What are these layers? Well, it draws things from the bottom up. It lays down one layer of graphics; then, you can keep adding on top, drawing more things. So the idea is like: Layer 1 + Layer 2 + Layer 3, and so on. If you want Layer 3 to be Layer 2, then you switch them in the code.

Here is a way of thinking about ggplot code:

```
ggplot(name_of_data, aes(x = name_of_x_variable, y = name_of_y_variable)) +
    geom_layer()+
    geom_layer()+
    geom_layer()
```

Let's focus on the + signs in the above description. What we are doing with the plus signs is adding layers to the plot. The layers get added in the order that they are written. If you look back to our previous code, you will see we added a `geom_bar` layer; then, we added another layer to change the rotation of the words on the x-axis. This is how it works.

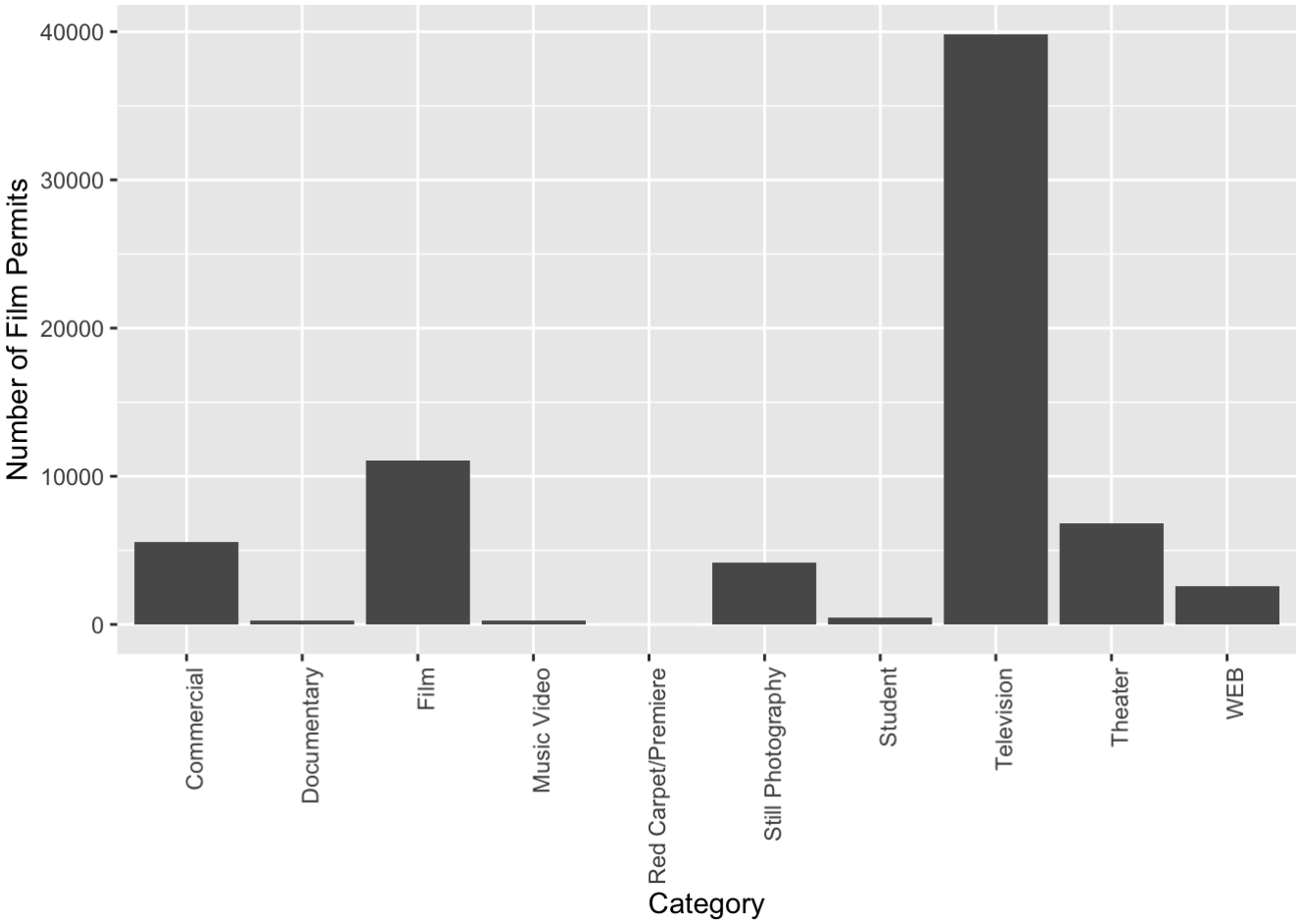
BUT WAIT? How are we supposed to know what to add? This is nuts! We'll give you lots of examples where you can copy and paste, and they will work. That's how you'll learn. If you want to read the help manual (https://ggplot2.tidyverse.org/reference/index.html), you can do that too. It's on the ggplot2 website. This will become useful after you already know what you are doing; before that, it will probably just seem confusing. However, it is pretty neat to look at and see all the different things you can do (http://www.ggplot2-exts.org/gallery/); it's very powerful.

Let's get the hang of adding things to the graph that let us change some stuff we might want to change. For example, how do you add a title? Or change the labels on the axes? Or add different colors, change the font size, or change the background? You can adjust these things by adding other lines to the existing code.

ylab() changes y label

The last graph had `count_of_permits` as the label on the y-axis. That doesn't look right. ggplot2 automatically took the title from the column and made it the name on the y-axis. We can change that by adding `ylab("what we want")`. We do this by adding a + to the last line, then adding `ylab()`.

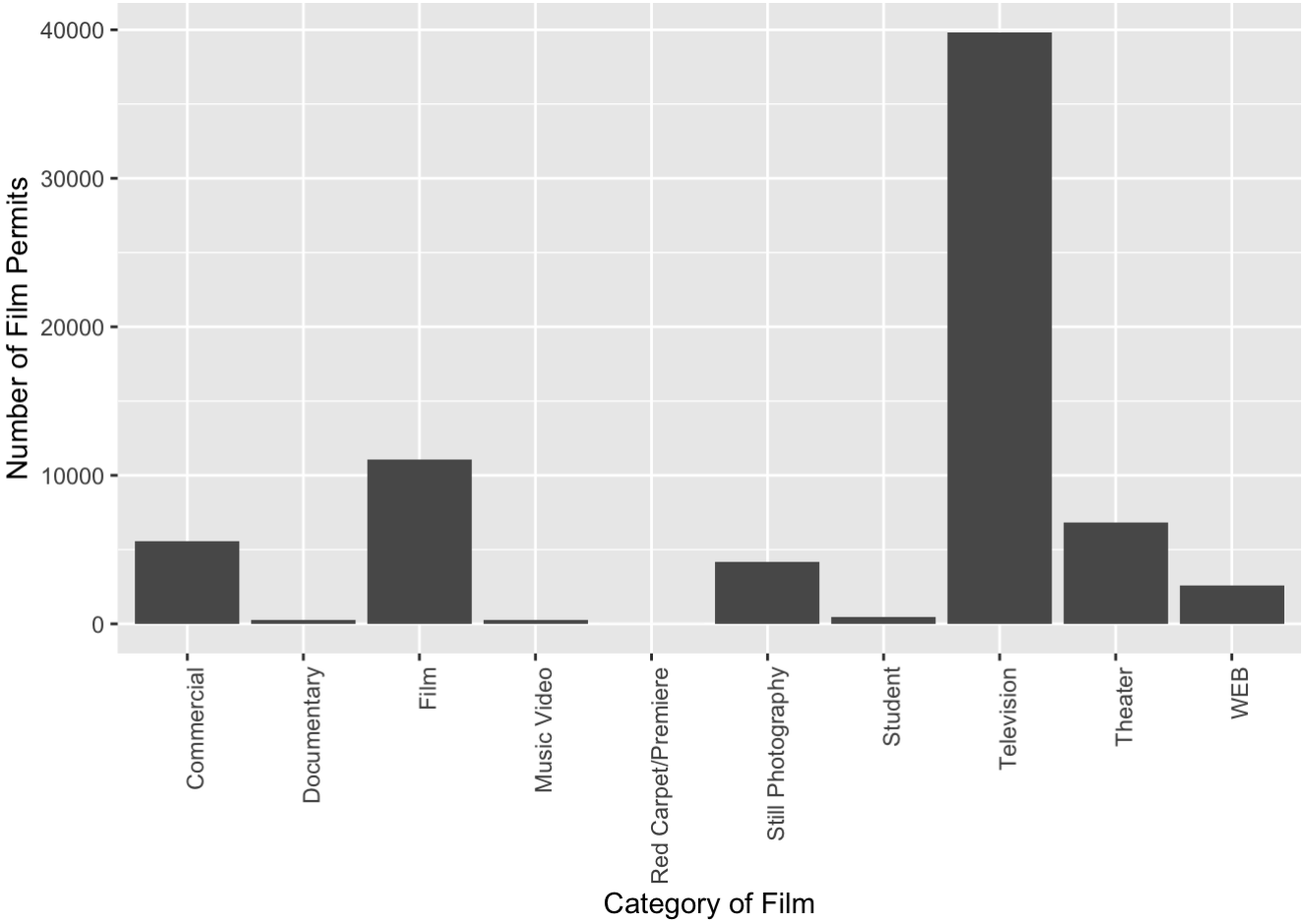
```
ggplot(counts, aes(x = Category, y = count_of_permits )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits")
```



xlab() changes x label

Let's slightly modify the x label too:

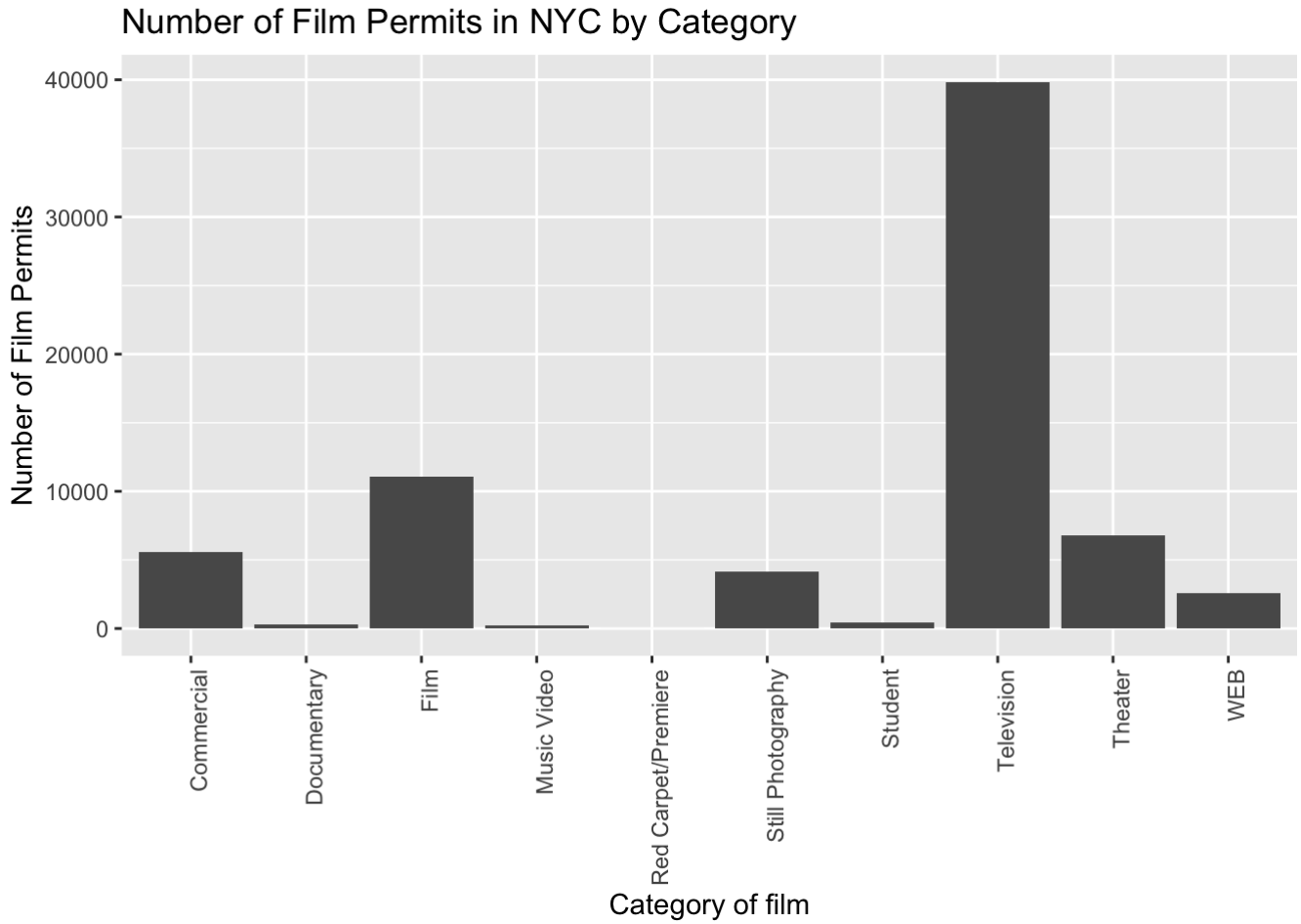
```
ggplot(counts, aes(x = Category, y = count_of_permits )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of Film")
```



ggtitle() adds title

Let's give our graph a title.

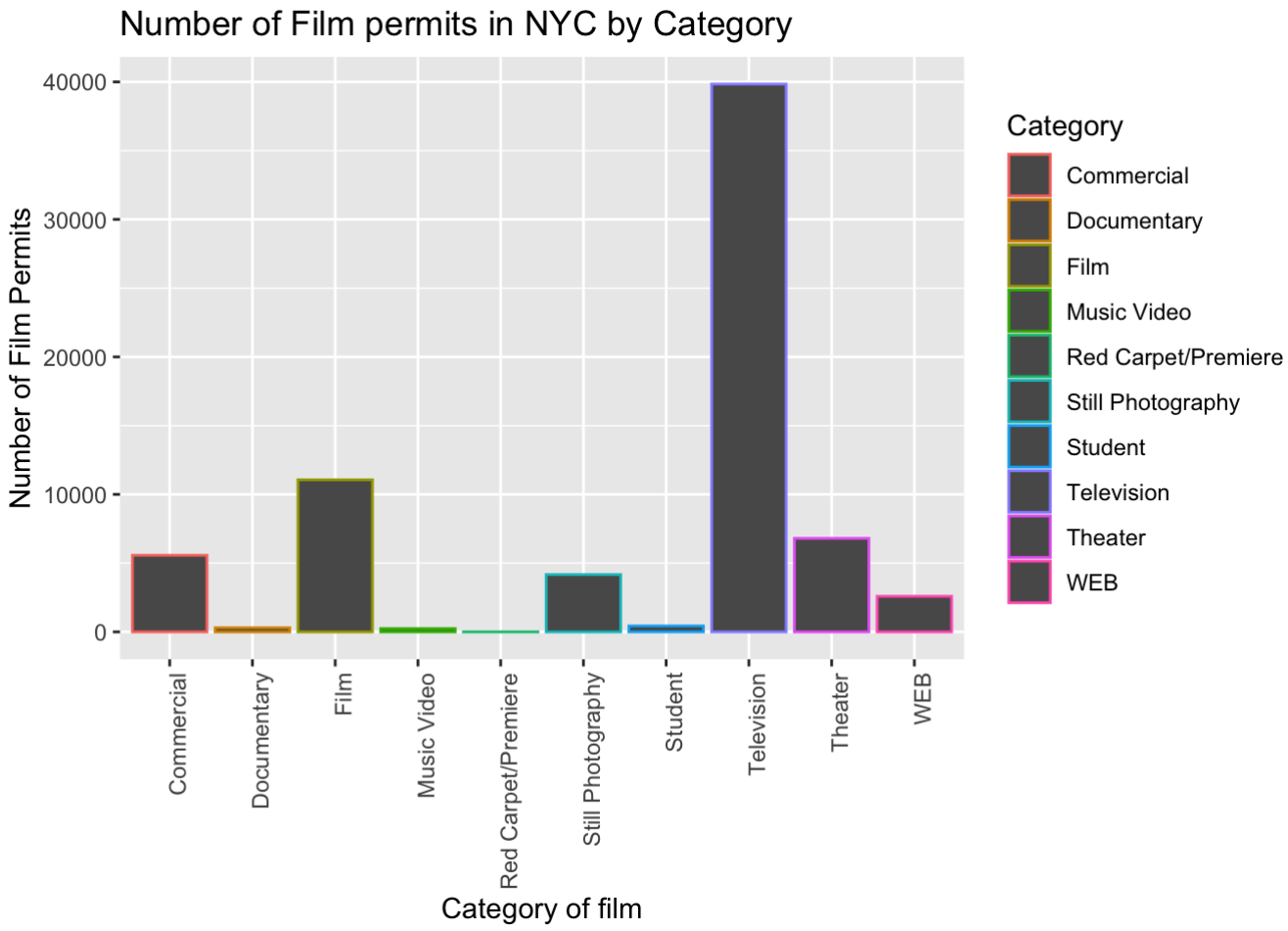
```
ggplot(counts, aes(x = Category, y = count_of_permits )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film Permits in NYC by Category")
```



Color adds color

Let's make the bars different colors. To do this, we add new code to the inside of the `aes()` part:

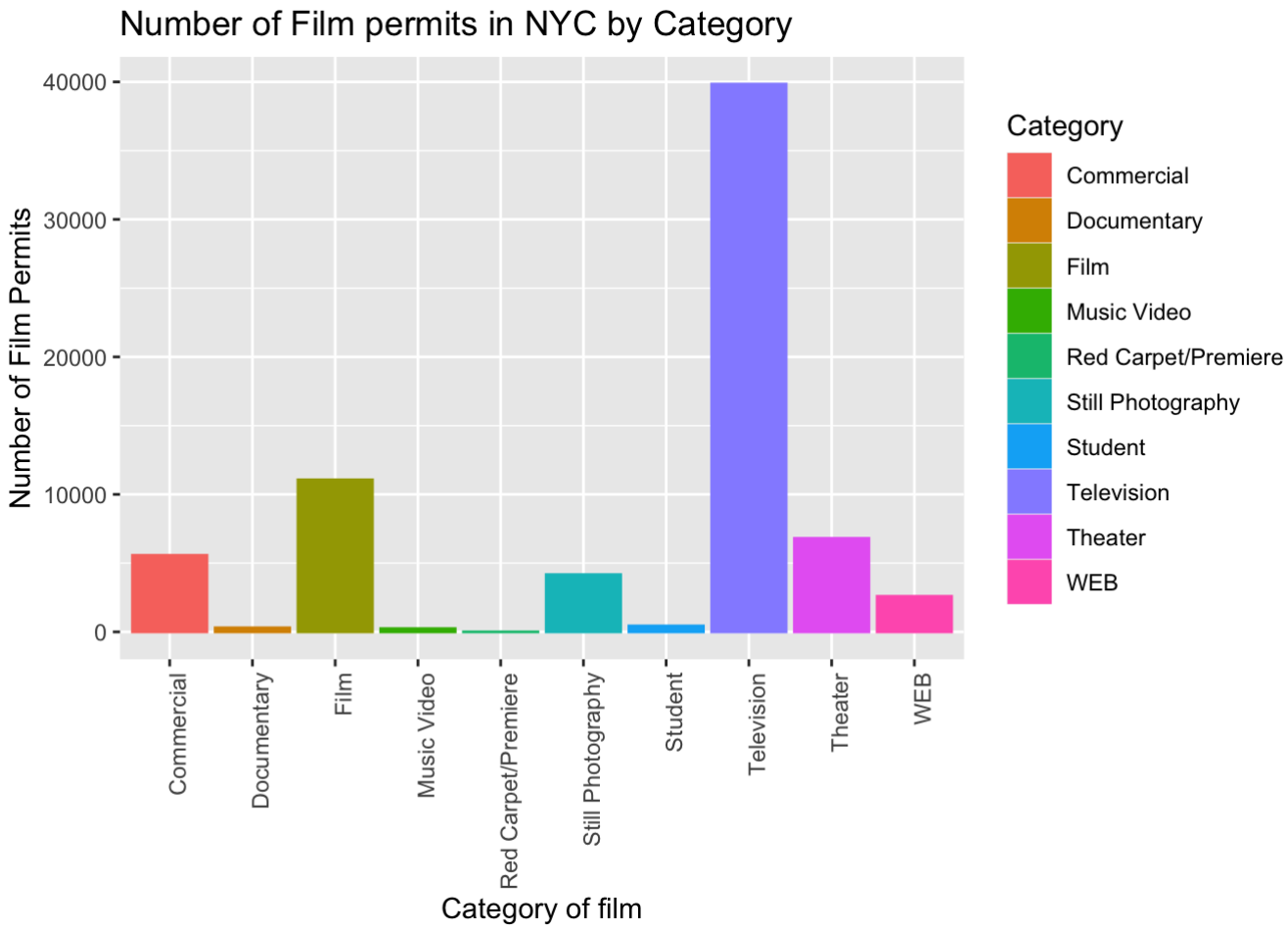
```
ggplot(counts, aes(x = Category, y = count_of_permits, color=Category )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category")
```



Fill fills in color

Let's make the bars different colors. To do this, we add new code to the inside of the `aes()` part… Notice that I've started using new lines to make the code more readable.

```
ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category")
```

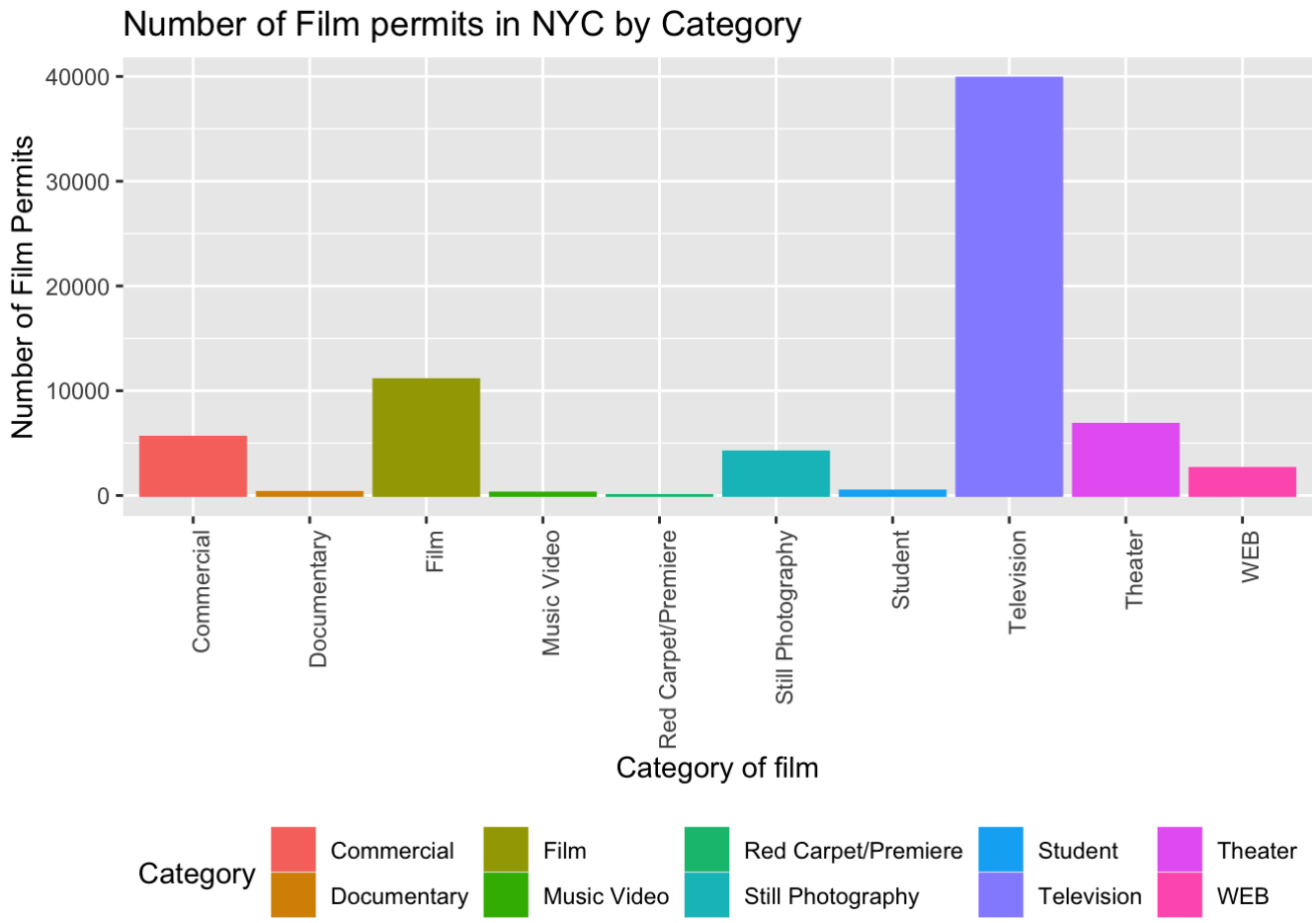


Modify or get rid of the legend

If we want to move the legend to appear below the graph, we can add the following:

```
theme(legend.position="bottom")
```

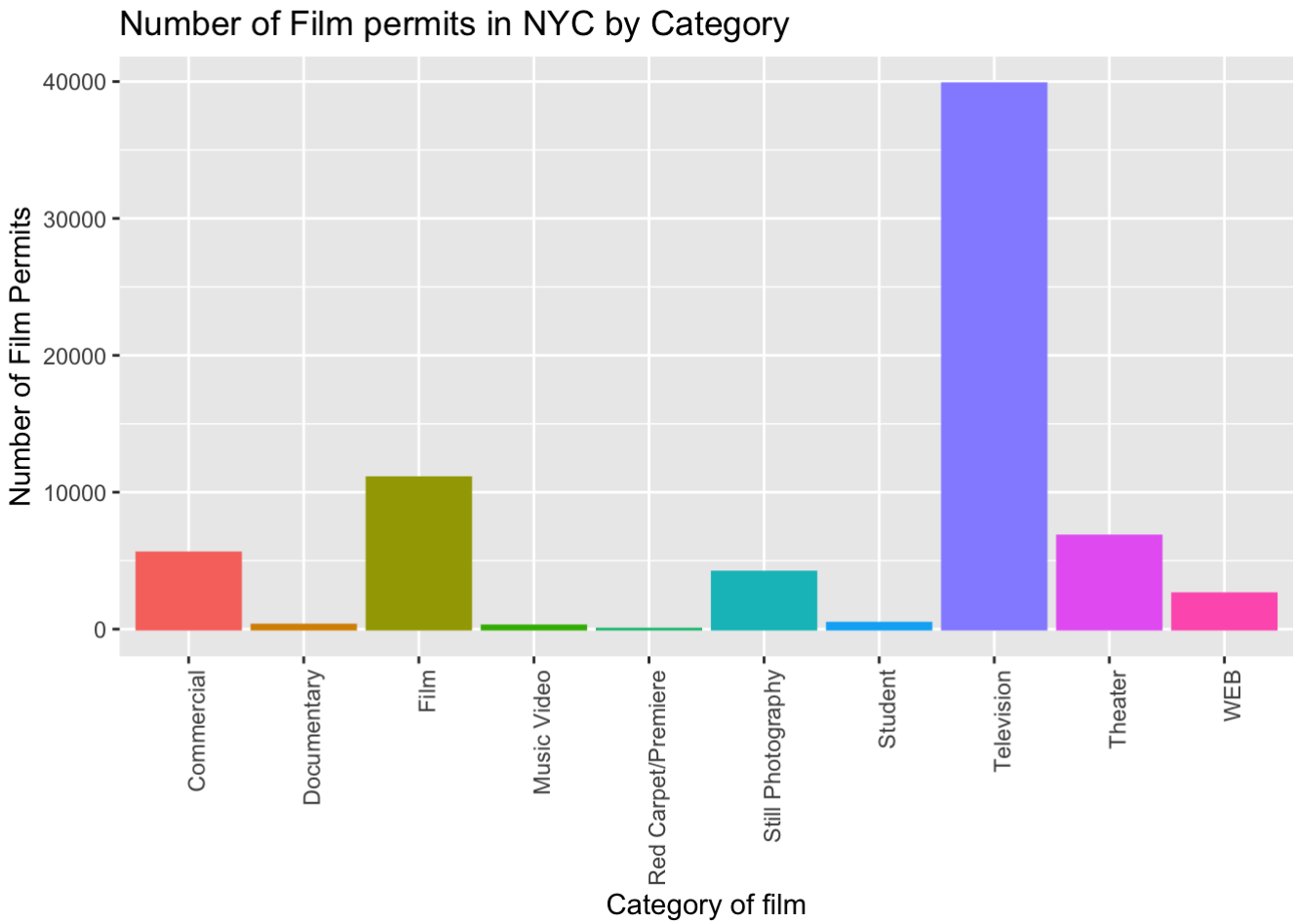
```
ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="bottom")
```



Sometimes you don't want a legend at all, to remove it, add the line:

```
theme(legend.position="none")
```

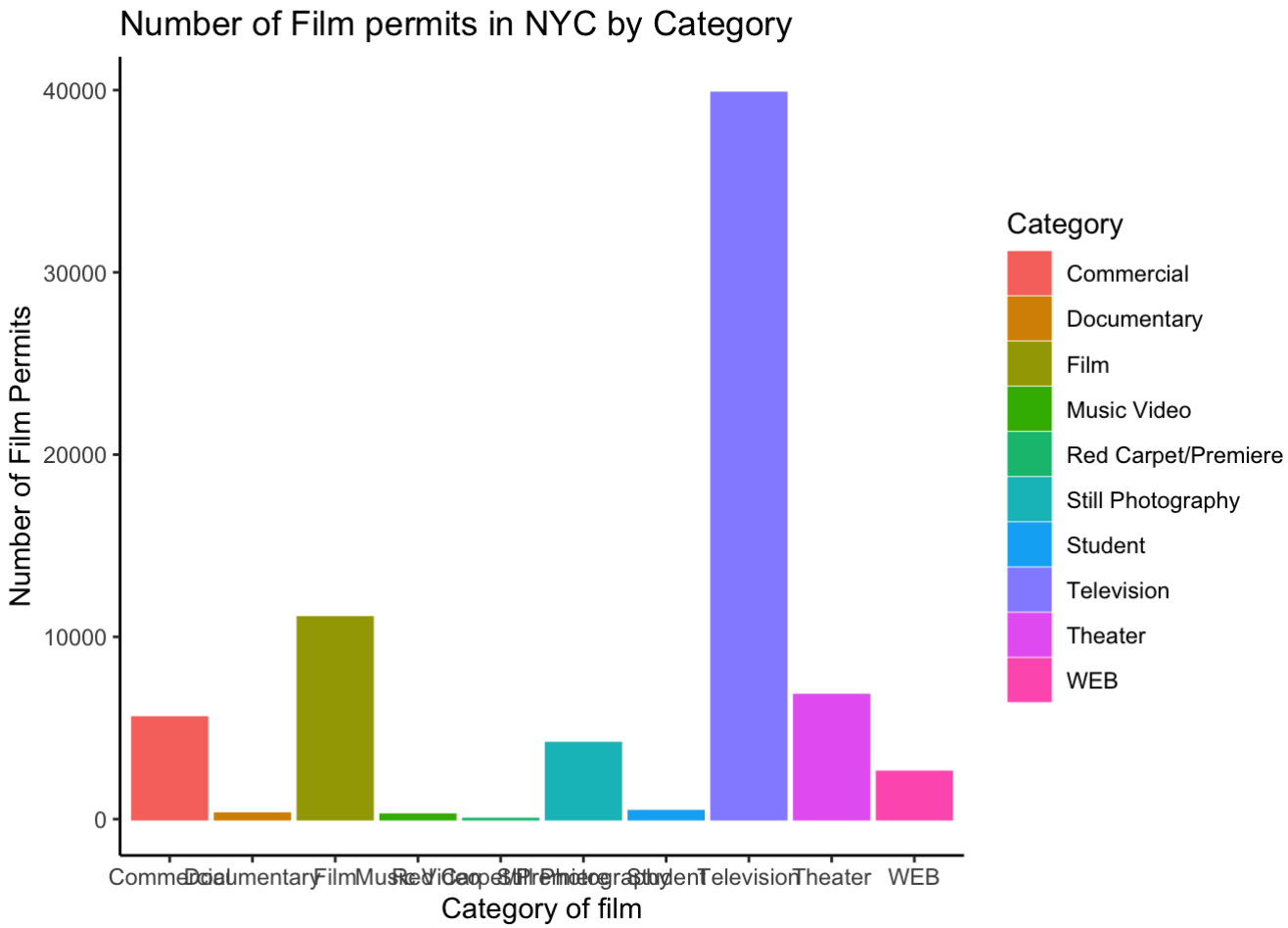
```
ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none")
```



theme_classic() makes white background

The rest is often just visual preference. For example, the graph above has this grey grid behind the bars. Use `theme_classic()` to remove the grid for a clean, classic no-nonsense look.

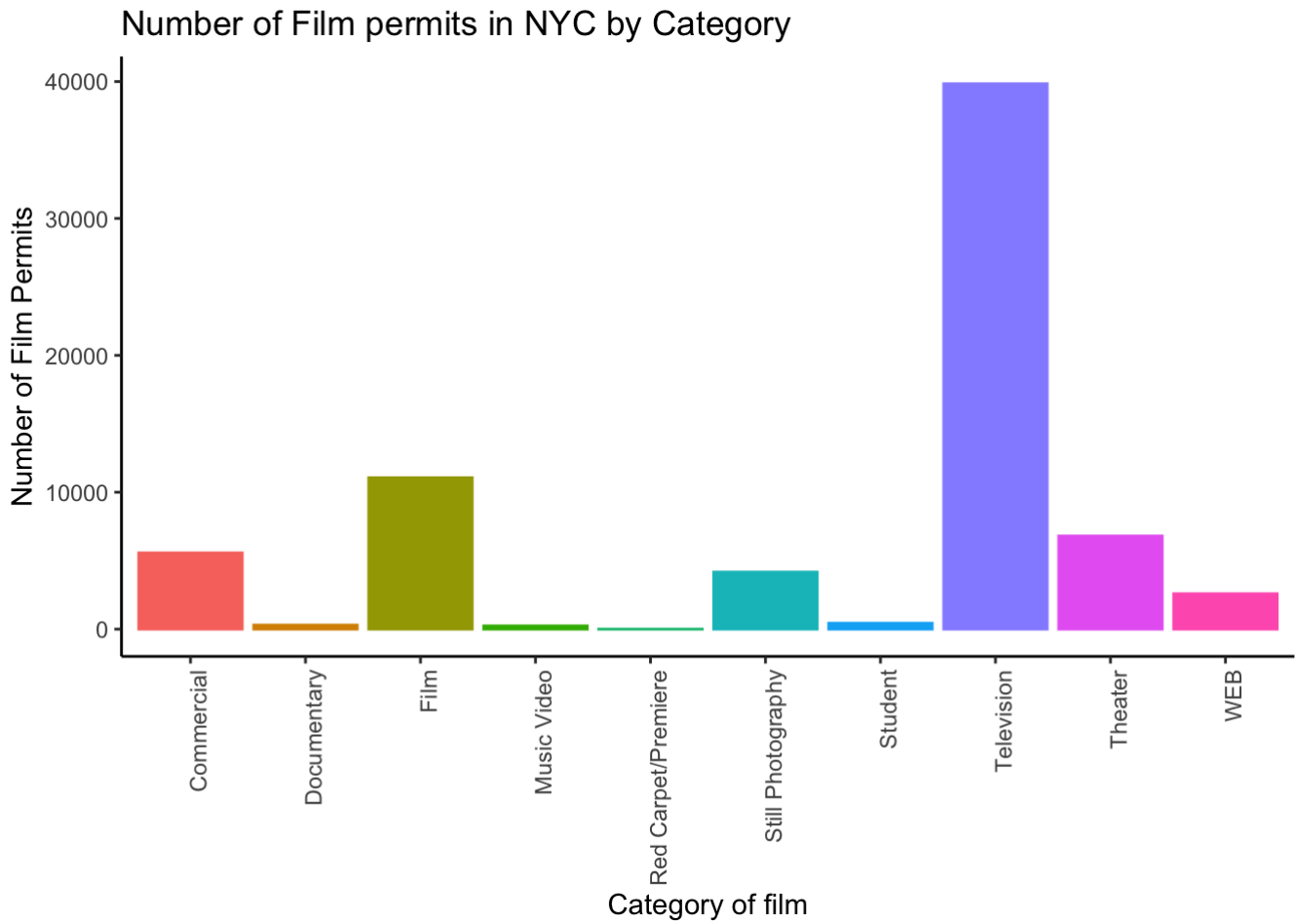
```
ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none") +
  theme_classic()
```



Sometimes, layer order matters

Interesting, `theme_classic()` is misbehaving a little bit. We have some of our layers out of order; let's re-order. I just moved `theme_classic()` to underneath the `geom_bar()` line. Now everything gets drawn properly.

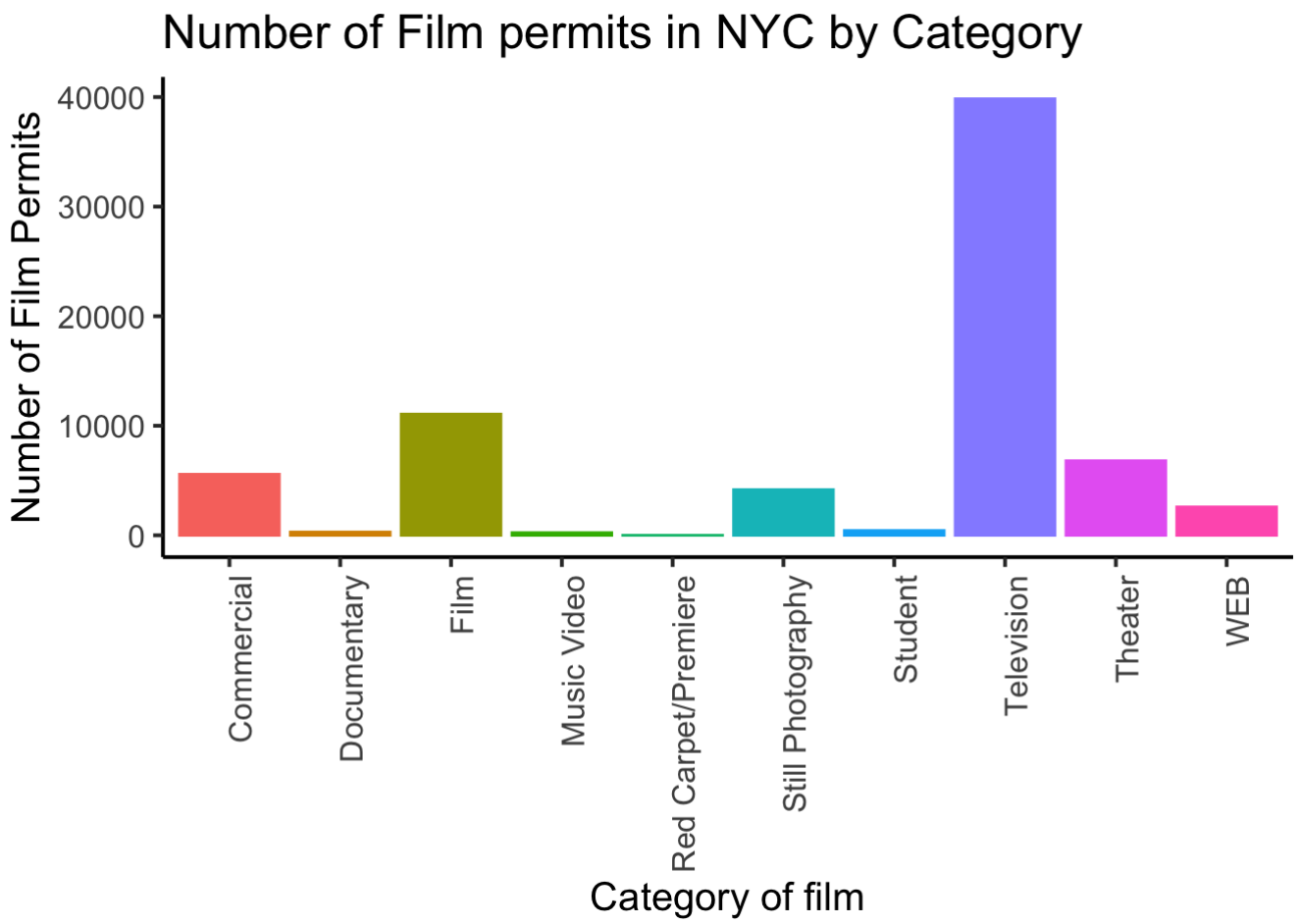
```
ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none")
```



Font-size

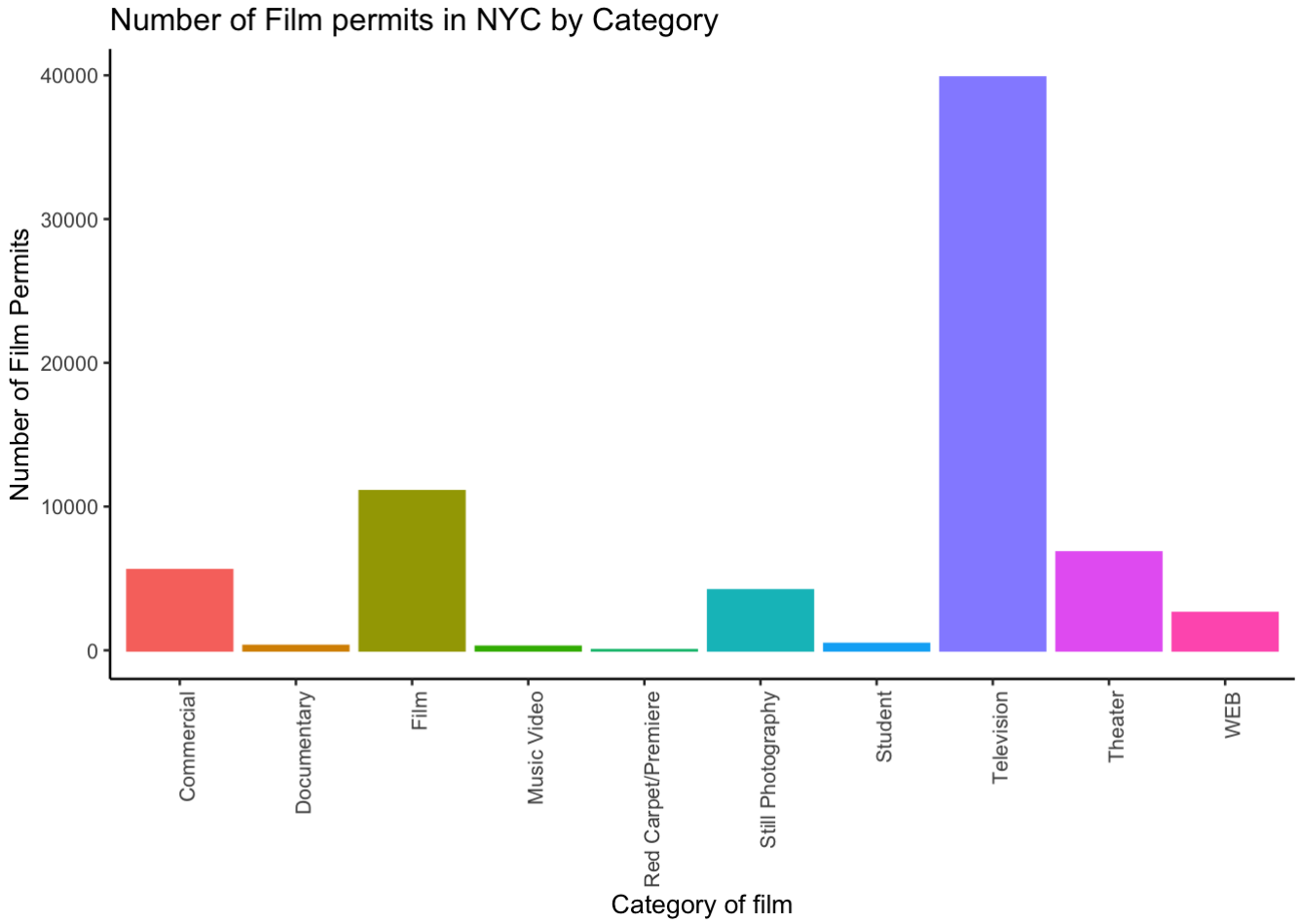
Changing font size is often something you want to do. ggplot2 can do this in different ways. I suggest using the `base_size` option inside `theme_classic()`. You set one number for the largest font size in the graph, and everything else gets scaled to fit that number. It's convenient. Look for the inside of `theme_classic()`.

```
ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 15) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none")
```



Or make things small… to see what happens.

```
ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 10) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category") +
  theme(legend.position="none")
```



Example

Plot bar graph for the data set *Motor_Vehicle_Collisions_Crashes.csv* again. But this time, your graph should include a white background, colored bars, properly labeled horizontal and vertical axes, a title, and no legend.

```
Enter your code below:
```

ggplot2 summary

That's enough of the ggplot2 basics for now. You will discover that many things are possible with ggplot2. It is amazing. We will get back to answering some questions about the data with graphs. But now that we have built the code to make the graphs, we need to copy-paste and make a few minor changes, and boom, we have our chart.

More questions about NYC films

What are the sub-categories of films?

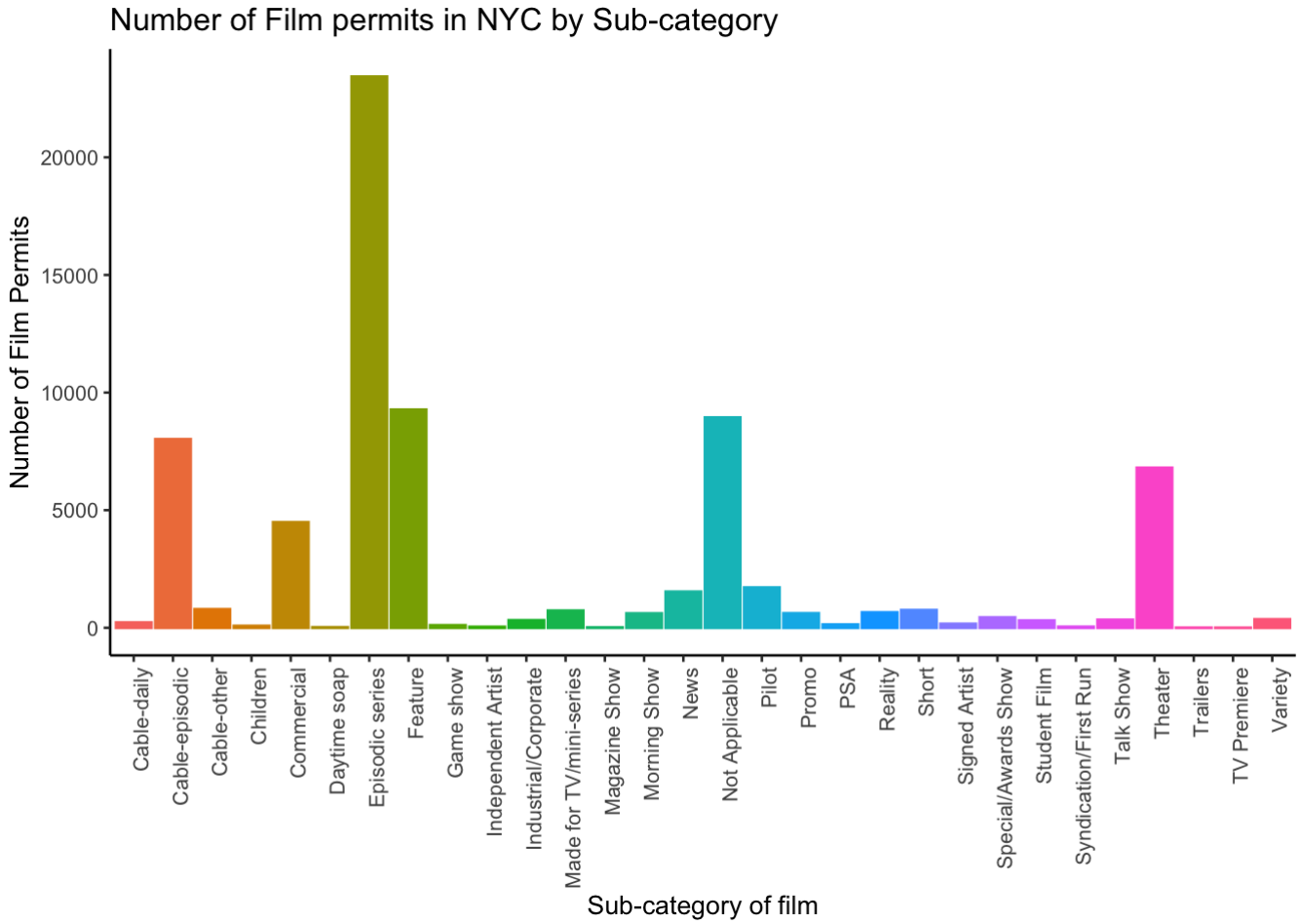
Let's graph the data for `SubCategoryName` in the **nyc_films**.

```
# get the counts (this is a comment. It's just here for you to read)

counts <- nyc_films %>%
          group_by(SubCategoryName) %>%
          summarize(count_of_permits = length(SubCategoryName))

# make the plot

ggplot(counts, aes(x = SubCategoryName, y = count_of_permits,
                   color=SubCategoryName,
                   fill= SubCategoryName )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 10) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Sub-category of film") +
  ggtitle("Number of Film permits in NYC by Sub-category") +
  theme(legend.position="none")
```



It shows that “episodic series” is the most common. Using a graph like this gave us our answer super fast.

Categories by different Boroughs

Let’s see one more thing about ggplot2. It’s called `facet_wrap()`.

We know that some films are made in different boroughs and the same films are made in different categories, but do different boroughs have different patterns for the kinds of categories of films they request permits for? Are there more TV shows in Brooklyn? How do we find out? Watch just like this:

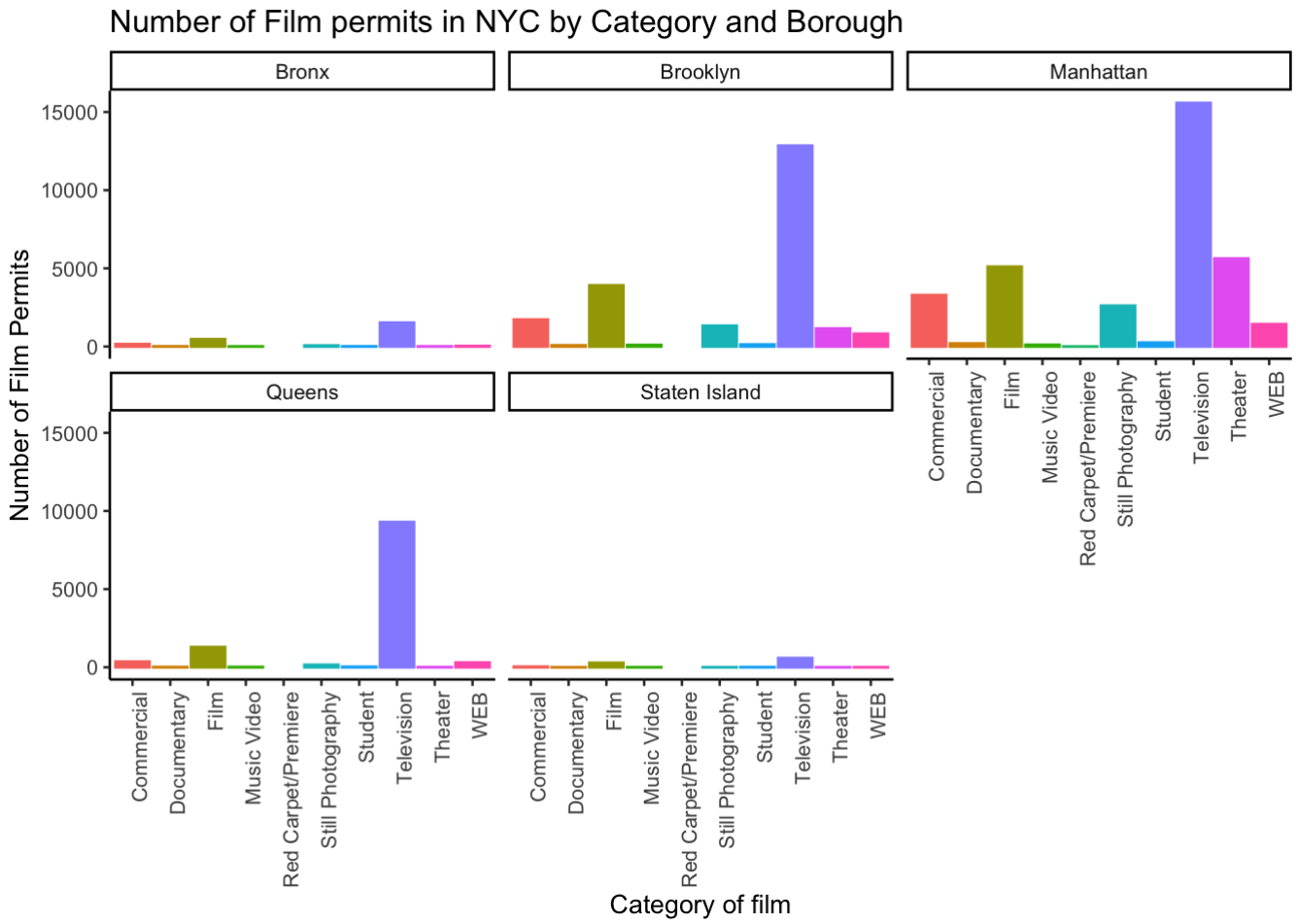
```
# get the counts (This is a comment. It's just here for you to read)

counts <- nyc_films %>%
          group_by(Borough,Category) %>%
          summarize(count_of_permits = length(Category))
```

```
## `summarise()` has grouped output by 'Borough'. You can override using the
## `.groups` argument.
```

```
# make the plot

ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 10) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in NYC by Category and Borough") +
  theme(legend.position="none") +
  facet_wrap(~Borough, ncol=3)
```



We did two critical things. First, we added `Borough` and `Category` into the `group_by()` function. This automatically gives separate counts for each film category for each Borough. Then we added `facet_wrap(~Borough, ncol=3)` to the end of the plot, and it automatically drew us 5 different bar graphs, one for each Borough! That was fast. Imagine doing that by hand.

Example

Plot the number of crashes by the variable *NUMBER OF PERSONS INJURED* and separated by boroughs

Hint: Remember that R doesn't accept spaces in the name of the variables.

```
Enter your code below:
```

Filtering data by a single Borough

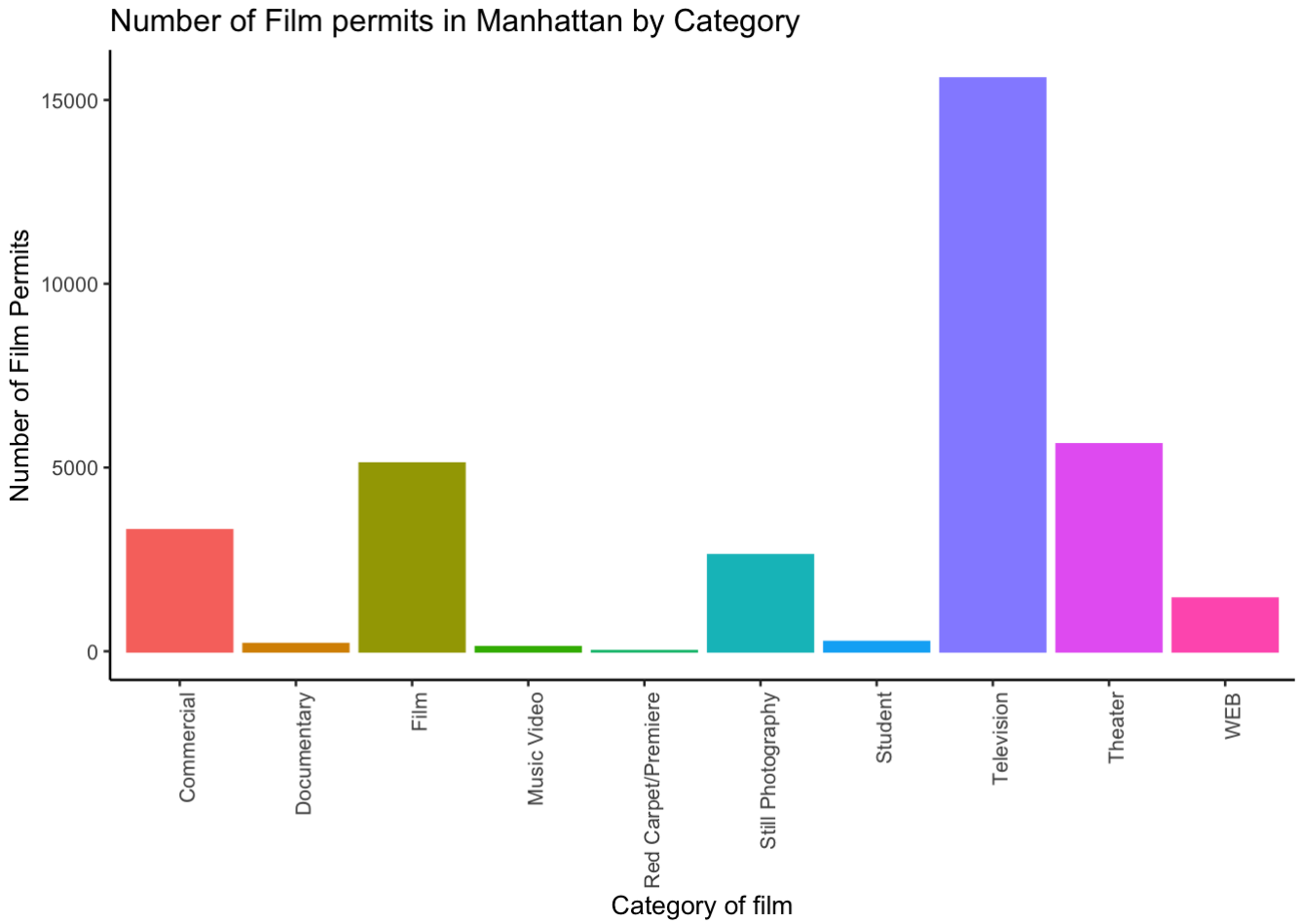
The library `dplyr` allows us to filter data by a single borough (see Lab 3). Let's use the above example about plotting categories, but we only plan the data for Manhattan this time.

```
# get the counts of categories for Manhattan (this is a comment. It's just here for you to read)

counts <- nyc_films %>%
          group_by(Category) %>%
          filter(Borough=='Manhattan')%>%
          summarize(count_of_permits = length(Category))

# make the plot

ggplot(counts, aes(x = Category, y = count_of_permits,
                   color=Category,
                   fill= Category )) +
  geom_bar(stat="identity") +
  theme_classic(base_size = 10) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ylab("Number of Film Permits") +
  xlab("Category of film") +
  ggtitle("Number of Film permits in Manhattan by Category") +
  theme(legend.position="none")
```



Note that this graph is the same as obtained for Manhattan as in the previous graph with facet wrap.

Example

Plot the *NUMBER OF PERSONS INJURED* for Manhattan with legend.

```
Enter your code below:
```

References

The material used in this document contains excerpts and modifications from:

- Matthew J. C. Crump, Anjali Krishnan, Stephen Volz, and Alla Chavarga (2018) “Answering questions with data: Lab Manual.” Last compiled on 2019-04-06. https://www.crumplab.com/statisticsLab/ (https://www.crumplab.com/statisticsLab/)