Welcome to my Workshop!

This markdown file should contain lots of helpful information. Feel free to save it and refer to it in the future for any work that you may have to do! ~ ericwan

Loading... the Data

Here we load libraries and install packages. This can be accomplished using `install.packages()` and `library()`.

```
# install.packages("tidyverse")
# Run the above command if you have not already installed the tidyverse package!
# If this is your first time using RStudio/R then you most likely have not yet installed this package.

library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# This library contains the library(ggplot2). Tidyverse is super useful.
```

Preliminary EDA

We begin by loading the data! We use the method `read.csv()` from tidyverse. Note the parameters we are passing into it.

```
data = read.csv('train.csv', header = TRUE, na.strings = c("","NA"))
head(data)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                 Name    Sex Age SibSp Parch
## 1                            Braund, Mr. Owen Harris   male  22     1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
## 3                             Heikkinen, Miss. Laina female  26     0     0
## 4       Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
## 5                           Allen, Mr. William Henry   male  35     0     0
```

```
## 6                                    Moran, Mr. James   male  NA     0     0
##              Ticket    Fare Cabin Embarked
## 1         A/5 21171  7.2500  <NA>        S
## 2          PC 17599 71.2833   C85        C
## 3 STON/O2. 3101282  7.9250  <NA>        S
## 4           113803 53.1000  C123        S
## 5           373450  8.0500  <NA>        S
## 6           330877  8.4583  <NA>        Q
```

Now that we have gotten a quick peek at our data, let's look at some other aspects. Note that there are 891 rows and 12 columns.

```
dim(data)
```

```
## [1] 891  12
```

The `names()` function helps us grab the column names, in order.

```
names(data)
```

```
##  [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"
##  [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
## [11] "Cabin"       "Embarked"
```

What do all of these names mean? Looking at our source for the dataset, we find the following descriptions for some of the less obvious names:

(https://www.kaggle.com/c/titanic/data)

sibsp : # of siblings / spouses aboard the Titanic (mistresses and fiancés were ignored)

parch : # of parents / children aboard the Titanic (parch = 0 for children travelling with only a nanny)

embarked : Port of Embarkation, where C = Cherbourg, Q = Queenstown, S = Southampton

Sometimes, we may work with datasets that are incomplete and are missing values. Luckily, there are several ways we can figure out if this is the case. Note from the output that we are missing information in the 'Age', 'Cabin', and 'Embarked' columns, and that we can even find out how many values are missing. We can also verify that other information is not incorrect.

Making sure there is no duplicated passenger ID.

```
sum(duplicated(data$PassengerId)) == 0
```

```
## [1] TRUE
```

```
sum(data$PassengerId == 1:891) == 891
```

```
## [1] TRUE
```

Checking if there any missing values.

```r
sum(is.na(data$PassengerId))
```

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

```r
sum(is.na(data$Survived))
```

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

```r
sum(is.na(data$Pclass))
```

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

```r
sum(is.na(data$Age))
```

```
## [1] 177
```

```r
any(is.na(data$Age))
```

```
## [1] TRUE
```

```r
colnames(data)[apply(data, 2, anyNA)]
```

```
## [1] "Age"      "Cabin"    "Embarked"
```

A good idea to begin with is to always to explore the distributions of our features. Below I have graphed the distributions of several variables.

```r
# install.packages("gridExtra")
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

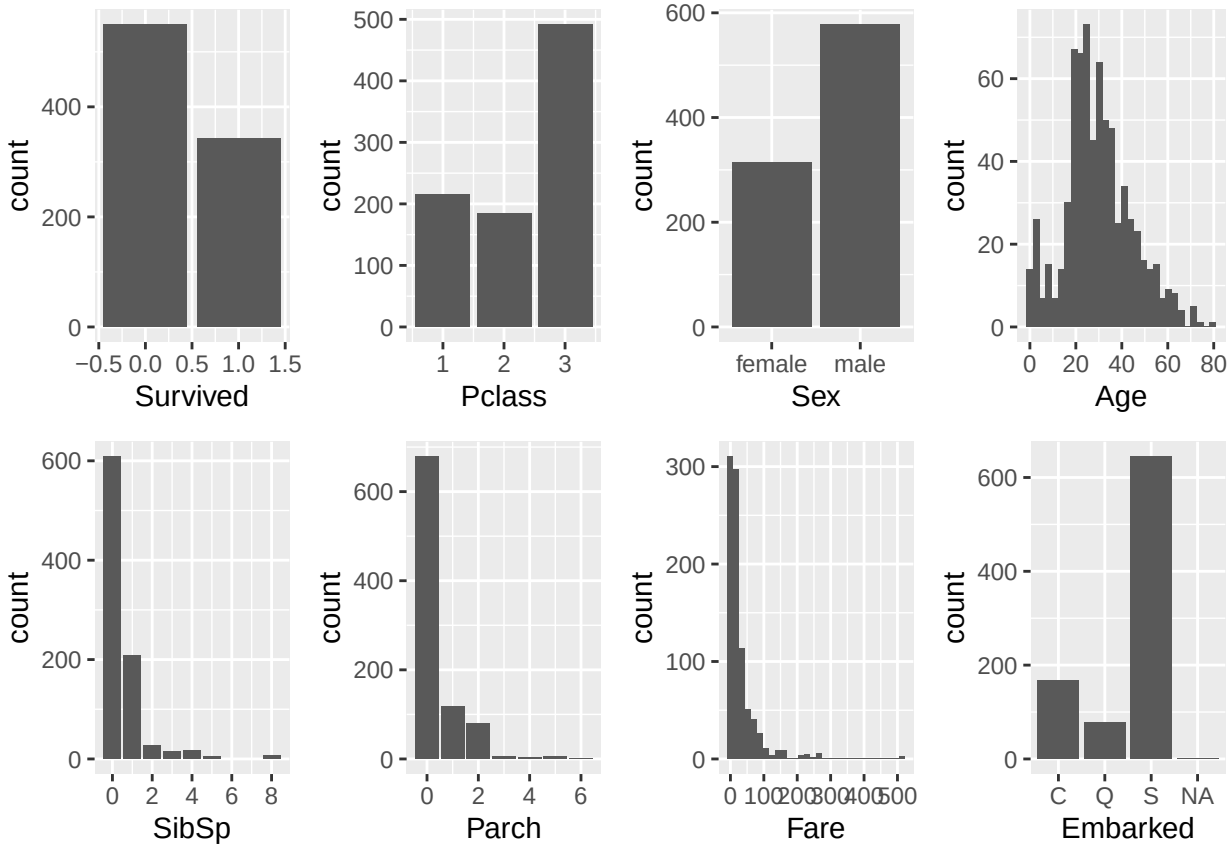
```r
plotSurvive <- ggplot(data) +
  geom_bar(aes(x = Survived))

plotClass <- ggplot(data) +
  geom_bar(aes(x = Pclass))

plotSex <- ggplot(data) +
  geom_bar(aes(x = Sex))

plotAge <- ggplot(data) +
  geom_histogram(aes(x = Age))
```

```
plotSibSp <- ggplot(data) +
  geom_bar(aes(x = SibSp))

plotParch <- ggplot(data) +
  geom_bar(aes(x = Parch))

plotFare <- ggplot(data) +
  geom_histogram(aes(x = Fare))

plotEmbark <- ggplot(data) +
  geom_bar(aes(x = Embarked))

grid.arrange(plotSurvive, plotClass, plotSex, plotAge, plotSibSp, plotParch, plotFare, plotEmbark, ncol
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 177 rows containing non-finite values (stat_bin).
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



What conclusions can we draw from the code below, including the information we already know and have seen?

```
length(unique(data$Name))
```

```
## [1] 891
```

```
length(unique(data$Ticket))
```

```
## [1] 681
```

```
length(unique(data$Cabin))
```

```
## [1] 148
```

Some Feature Engineering

A tiny bit of feature engineering first. Feature engineering can be defined as the process of extracting features from raw data using data mining techniques to provide new insights for machine learning.

That's a lot of fancy words.

With regards to our data, note that `Pclass` and `Survived` are stored as numbers, and we're gonna change that so they are instead categorical values. We're going to quickly create "factor" variables from them. This will help with our analysis.

```
library(dplyr)
titanic <- mutate(data,
                  passengerClass = fct_recode(as.factor(Pclass),
                                              "1st" = "1", "2nd" = "2", "3rd" = "3"),
                  Survival = fct_recode(as.factor(Survived),
                                        "died" = "0", "lived" = "1"))
```

Where did the extra two columns come from? If you guessed that they were created after our call to `mutate()`, you are correct! Also we grab our new column names and we see that the changes we made have held.

```
dim(titanic)
```

```
## [1] 891  14
```

```
names(titanic)
```

```
##  [1] "PassengerId"    "Survived"       "Pclass"         "Name"
##  [5] "Sex"            "Age"            "SibSp"          "Parch"
##  [9] "Ticket"         "Fare"           "Cabin"          "Embarked"
## [13] "passengerClass" "Survival"
```

Just a quick look at our new changes.

```
head(titanic)
```

```
##   PassengerId Survived Pclass
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## 6           6        0      3
##                                                  Name    Sex Age SibSp Parch
## 1                             Braund, Mr. Owen Harris   male  22     1     0
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
## 3                              Heikkinen, Miss. Laina female  26     0     0
## 4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
## 5                            Allen, Mr. William Henry   male  35     0     0
## 6                                    Moran, Mr. James   male  NA     0     0
##             Ticket    Fare Cabin Embarked passengerClass Survival
## 1        A/5 21171  7.2500  <NA>        S            3rd     died
## 2         PC 17599 71.2833   C85        C            1st    lived
## 3 STON/O2. 3101282  7.9250  <NA>        S            3rd    lived
## 4           113803 53.1000  C123        S            1st    lived
## 5           373450  8.0500  <NA>        S            3rd     died
## 6           330877  8.4583  <NA>        Q            3rd     died
```

Let's now try something called boolean indexing to help us filter our data into two groups that we can use to gain more insights. Don't you like how convenient and logical R is?

```r
train_titanic_survived <- titanic[titanic$Survival == "lived", ]
train_titanic_perished <- titanic[titanic$Survival == "died", ]

head(train_titanic_survived)
```

```
##    PassengerId Survived Pclass
## 2            2        1      1
## 3            3        1      3
## 4            4        1      1
## 9            9        1      3
## 10          10        1      2
## 11          11        1      3
##                                                   Name    Sex Age SibSp Parch
## 2  Cumings, Mrs. John Bradley (Florence Briggs Thayer) female  38     1     0
## 3                               Heikkinen, Miss. Laina female  26     0     0
## 4         Futrelle, Mrs. Jacques Heath (Lily May Peel) female  35     1     0
## 9      Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg) female  27     0     2
## 10                  Nasser, Mrs. Nicholas (Adele Achem) female  14     1     0
## 11                      Sandstrom, Miss. Marguerite Rut female   4     1     1
##              Ticket    Fare Cabin Embarked passengerClass Survival
## 2          PC 17599 71.2833   C85        C            1st    lived
## 3  STON/O2. 3101282  7.9250  <NA>        S            3rd    lived
## 4            113803 53.1000  C123        S            1st    lived
## 9            347742 11.1333  <NA>        S            3rd    lived
## 10           237736 30.0708  <NA>        C            2nd    lived
## 11          PP 9549 16.7000    G6        S            3rd    lived
```

```
head(train_titanic_perished)
```

```
##    PassengerId Survived Pclass                           Name  Sex Age SibSp
## 1            1        0      3        Braund, Mr. Owen Harris male  22     1
## 5            5        0      3       Allen, Mr. William Henry male  35     0
## 6            6        0      3               Moran, Mr. James male  NA     0
## 7            7        0      1        McCarthy, Mr. Timothy J male  54     0
## 8            8        0      3 Palsson, Master. Gosta Leonard male   2     3
## 13          13        0      3 Saundercock, Mr. William Henry male  20     0
##    Parch    Ticket    Fare Cabin Embarked passengerClass Survival
## 1      0 A/5 21171  7.2500  <NA>        S            3rd     died
## 5      0    373450  8.0500  <NA>        S            3rd     died
## 6      0    330877  8.4583  <NA>        Q            3rd     died
## 7      0     17463 51.8625   E46        S            1st     died
## 8      1    349909 21.0750  <NA>        S            3rd     died
## 13     0 A/5. 2151  8.0500  <NA>        S            3rd     died
```

Introduction to ggplot

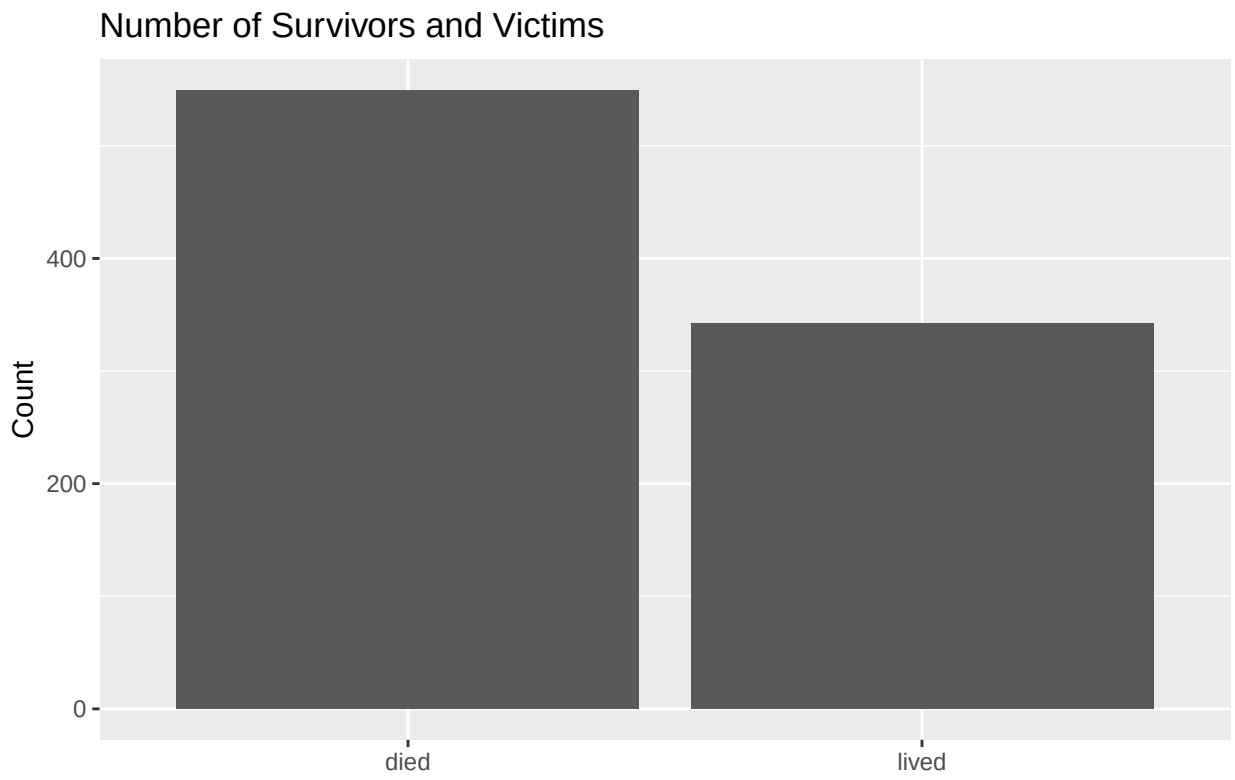
Using this sorted training data, let us now utilize plots to better understand the distribution of the data! We want to use the ggplot2 library because it is very flexible and has many useful methods. Also, it makes pretty graphs. Here we load the library, using `library()`. Note that ggplot2 is already included when we ran `load(tidyverse)` in the beginning of the file. That being said, we may not always want to import everything.

```
library(ggplot2)
```

Let's begin our more in-depth analysis of the Titanic dataset by examining how many people either survived or perished. We know that within our training dataset we have a total of 891 rows. We can reasonably infer that we are going to be training on a total of 891 people.

Take note of the specific calls I made within the code. In particular, what is the relevance of `aes()`, `geom_bar()` and `labs()`?

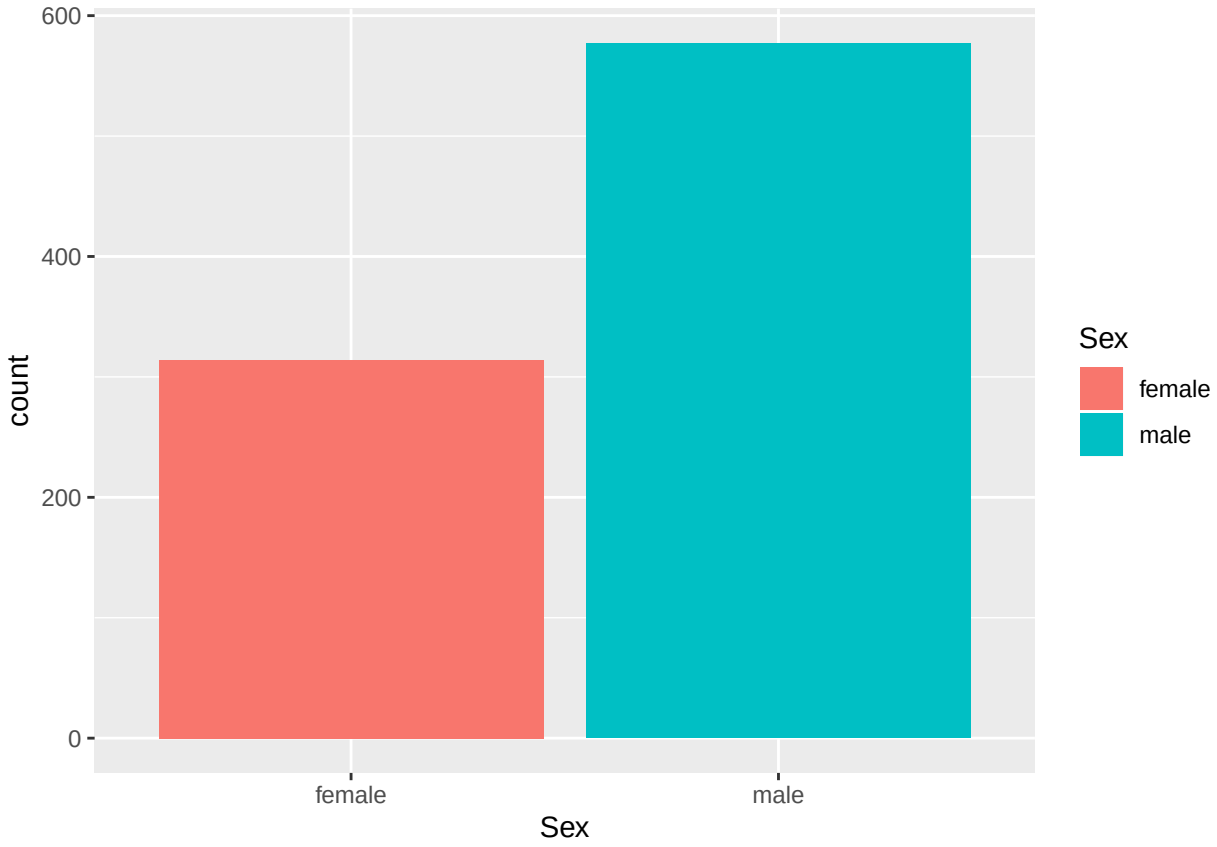
```
ggplot(titanic, aes(x = titanic$Survival)) +
  geom_bar() +
  labs(title = "Number of Survivors and Victims",
       x = "",
       y = "Count",
       caption = "This is a caption!")
```



Looking at Sex

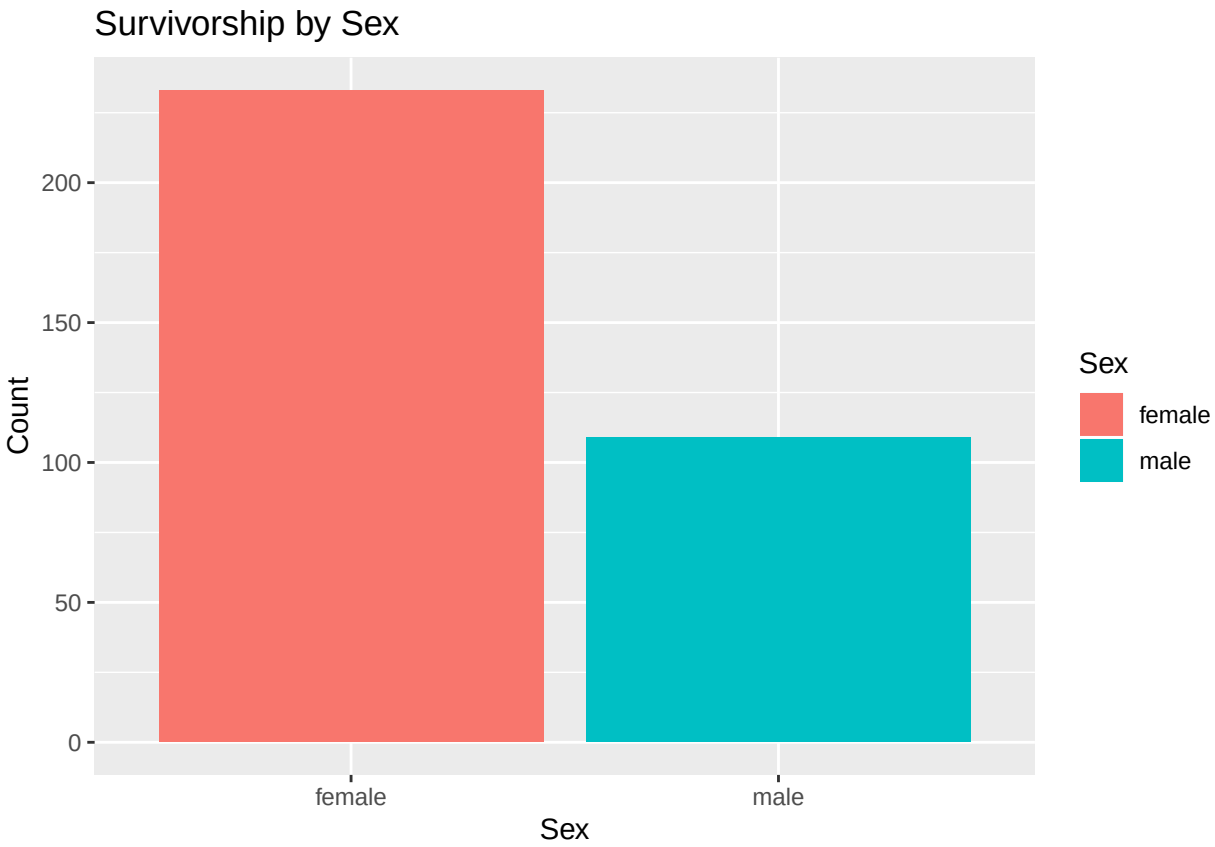
An iconic phrase from the Titanic disaster is "women and children first!". We will first investigate if this is true by examining the data. From our first plot below, we see that were more males than females onboard.

```
ggplot(titanic, aes(x = Sex, fill = Sex)) +
  geom_bar(aes(x = Sex, fill = Sex))
```



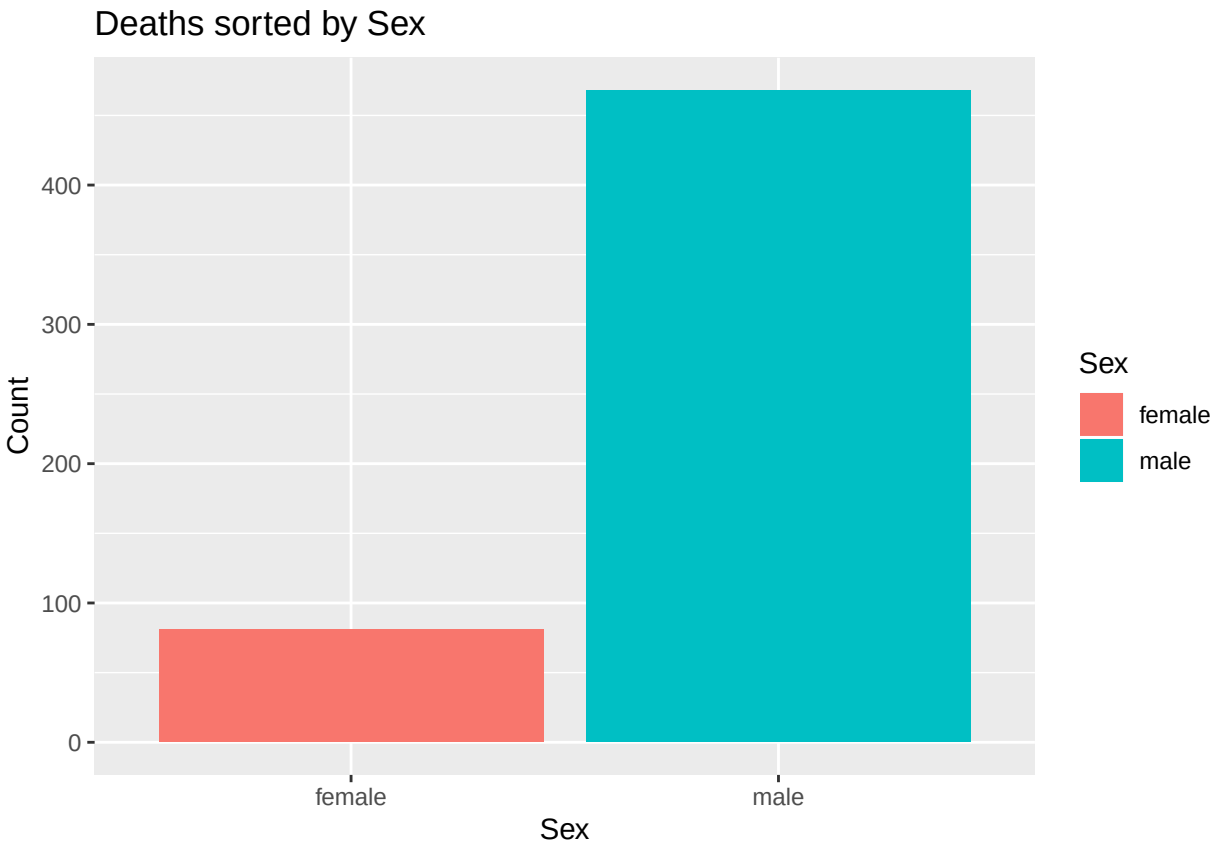
More females survived the Titanic disaster than males.

```
ggplot(train_titanic_survived) +
  geom_bar(aes(x = train_titanic_survived$Sex, fill = train_titanic_survived$Sex)) +
  labs(x = "Sex",
       y = "Count",
       title = "Survivorship by Sex",
       fill = "Sex")
```



More males died than females in the Titanic disaster.

```
ggplot(train_titanic_perished) +
  geom_bar(aes(x = train_titanic_perished$Sex, fill = train_titanic_perished$Sex)) +
  labs(x = "Sex",
       y = "Count",
       title = "Deaths sorted by Sex",
       fill = "Sex")
```



We thus see that it would probably be a great idea to use Sex in a model to help us predict if a passenger is going to or will not survive the titanic disaster. There are also other ways in which we can reach this conclusion, as shown below. We can use a contingency table.

In the code below the symbol `%>%` can be read as "pipe". The pipe operator inserts the object before the pipe into the function after the pipe. Here we're piping the output of `count()` into `spread()`. Piping objects is super useful. Also, make sure to run `library(dplyr)` and install the dplyr package.

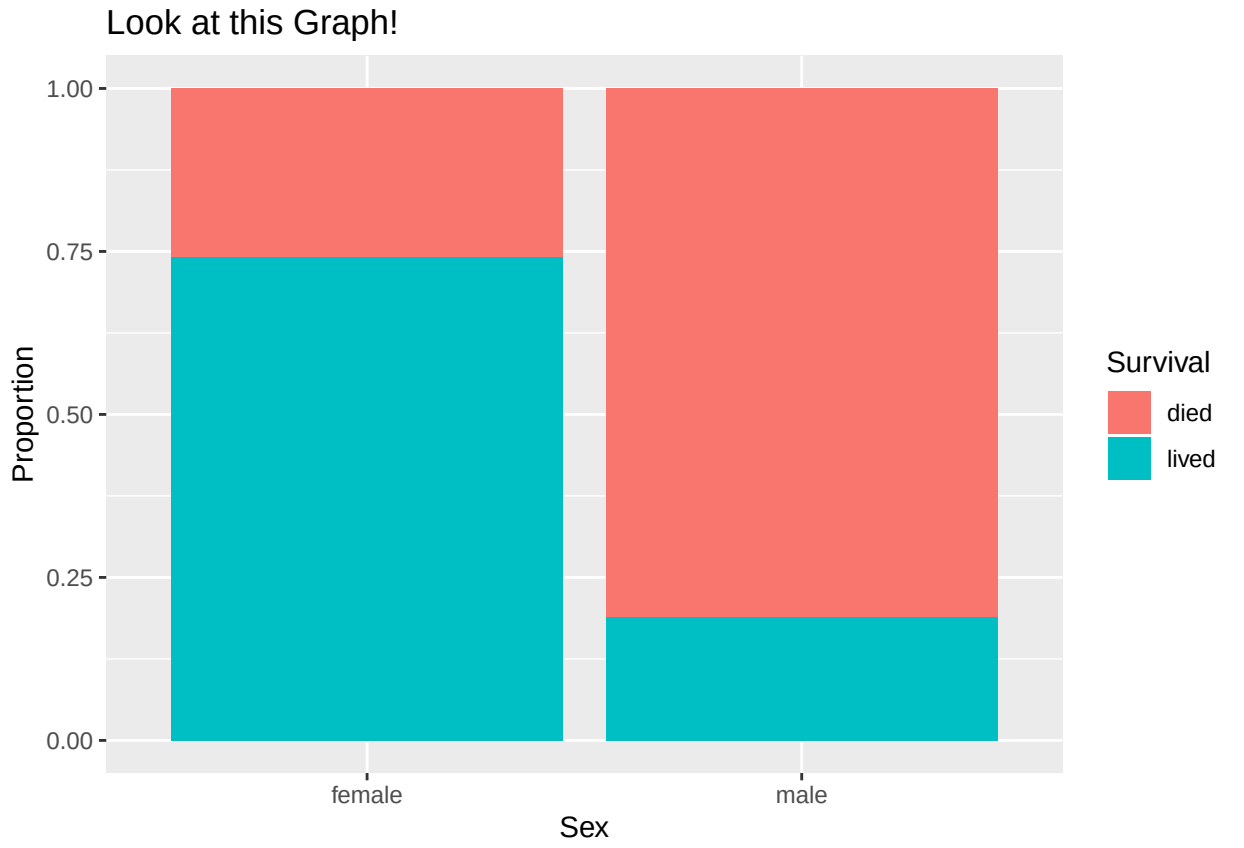
```
# install.packages("dplyr")
# library(dplyr)

count(titanic, Sex, Survival) %>%
  spread(Survival, n)
```

```
## # A tibble: 2 x 3
##   Sex     died lived
##   <fct>  <int> <int>
## 1 female    81   233
## 2 male     468   109
```

Fun thing, we can also build bar charts using proportions rather than counts. Which can be really helpful! Note the use of `position = "fill"`.

```
ggplot(titanic) +
  geom_bar(aes(Sex, fill = Survival), position = "fill") +
  labs(title = "Look at this Graph!",
```

```
       x = "Sex",
       y = "Proportion")
```



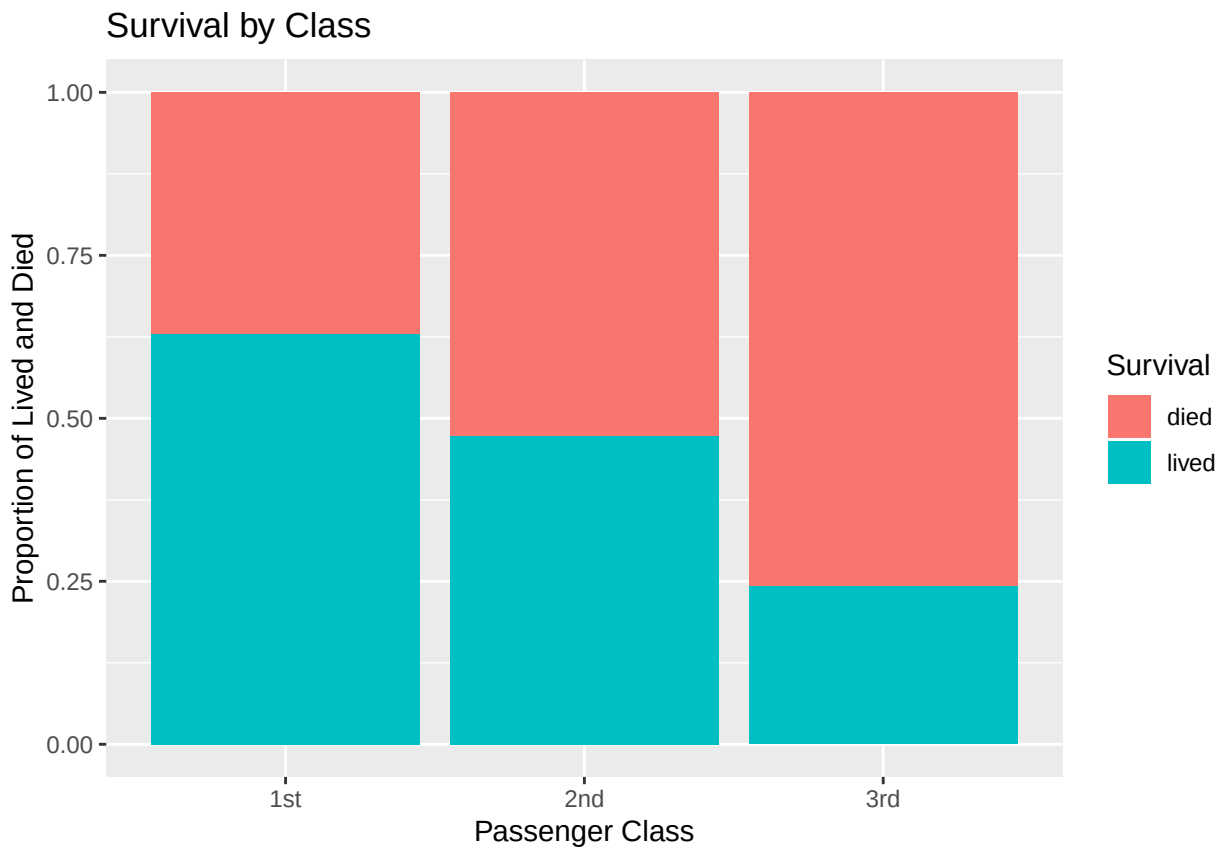
Okay, but what about class?

The class on a passenger's ticket is a useful proxy for their societal status. That makes it an interesting variable to consider, along with Sex.

Here is a bar plot that takes into account all three of these variables, by representing passenger class as a "facet" variable – we create different subplots for each class grouping. Also note `position = "fill"` also allows us to create and compare using proportions.

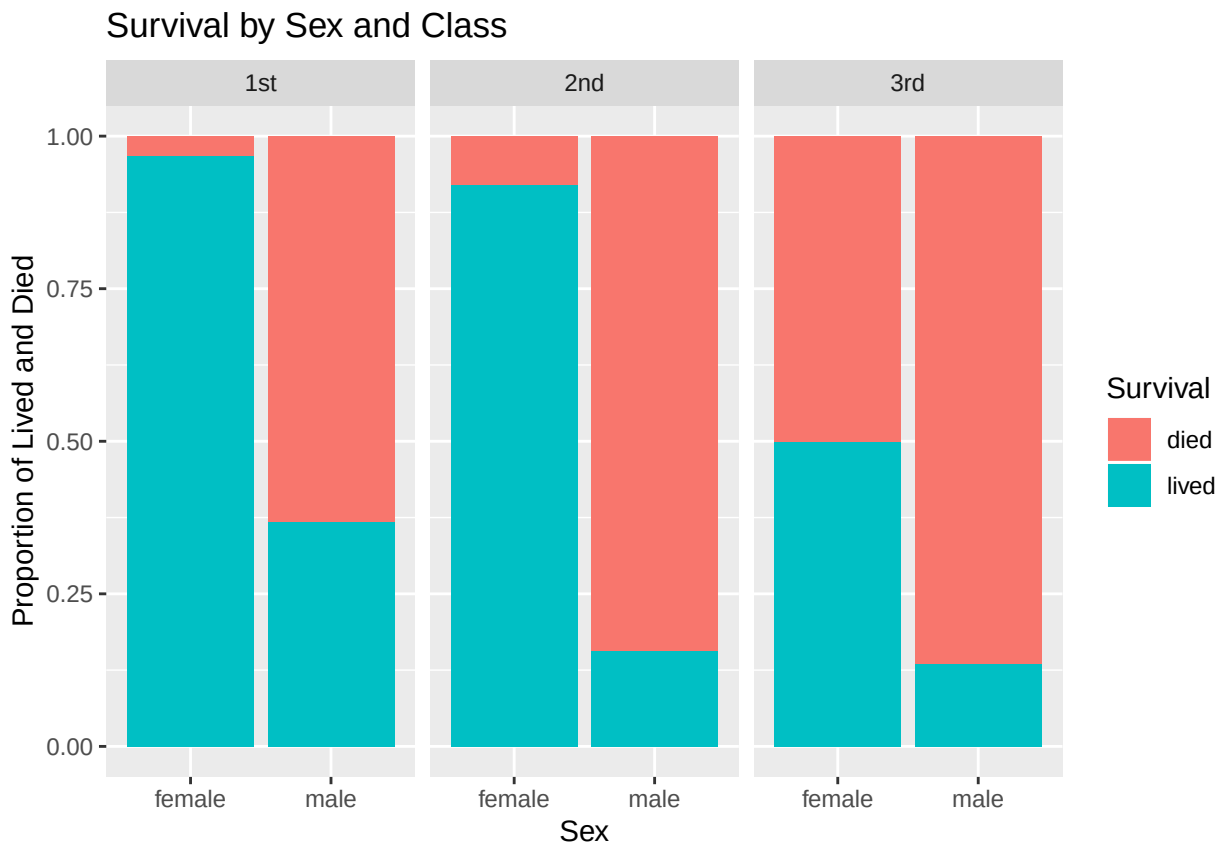
As we can observe from the plot, first class ticket holders survied the disaster at higher rates than their fellow passengers onboard the HMS Titanic.

```
ggplot(data = titanic) +
   geom_bar(aes(x = passengerClass, fill = Survival), position = "fill") +
   labs(x = "Passenger Class",
        y = "Proportion of Lived and Died",
        title = "Survival by Class")
```



What about how class intersects with sex in the Titanic disaster? It appears that regardless of class, females survived at higher rates than men, and that the first class ticket holders survied the disaster at higher rates than their fellow passengers onboard the HMS Titanic.

```
ggplot(titanic) +
   geom_bar(aes(Sex, fill = Survival), position = "fill") +
   facet_wrap(~passengerClass) +
   labs(y = "Proportion of Lived and Died",
        title = "Survival by Sex and Class")
```



Did Boomers die at a higher rate during the disaster?

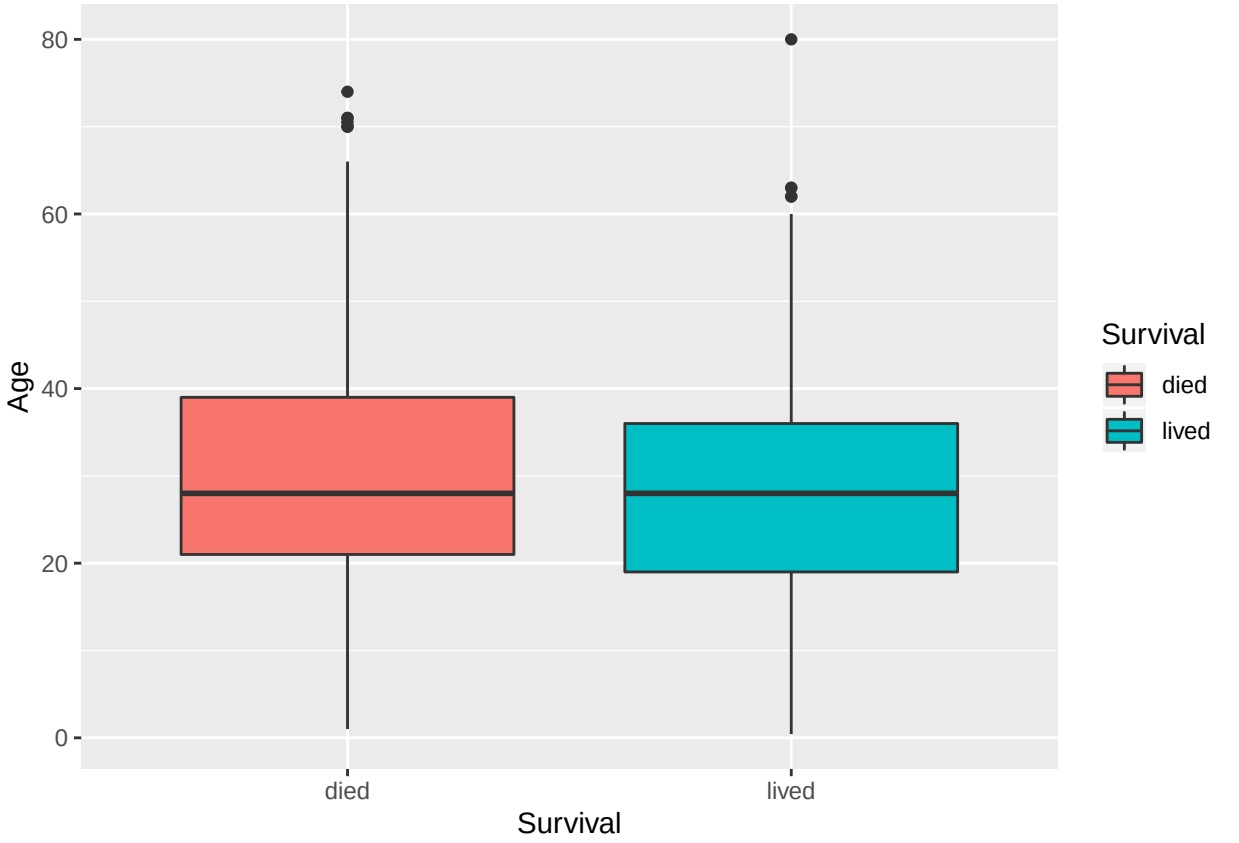
It might also be interesting to evaluate how age is related to the survival rates. Were older people sacrificing themselves to let the young live? Did babies just get tossed overboard? Let's find out!

Let's first use a histogram to visualize the distribution of a continuous variable. We filter out the 'NA' in the data using pipes and `filter()`.

```
titanic %>%
  filter(!is.na(Age)) %>%
  ggplot(data = ., aes(x = Age)) +
  geom_histogram(bins = 25) +
  labs(x = "Age",
       y = "Count",
       title = "Distribution of Age")
```

We can use boxplots to evaluate the spread of the `Age` variable. Judging from our plot, it does not appear that `Age` makes a significant difference between those who live and die.

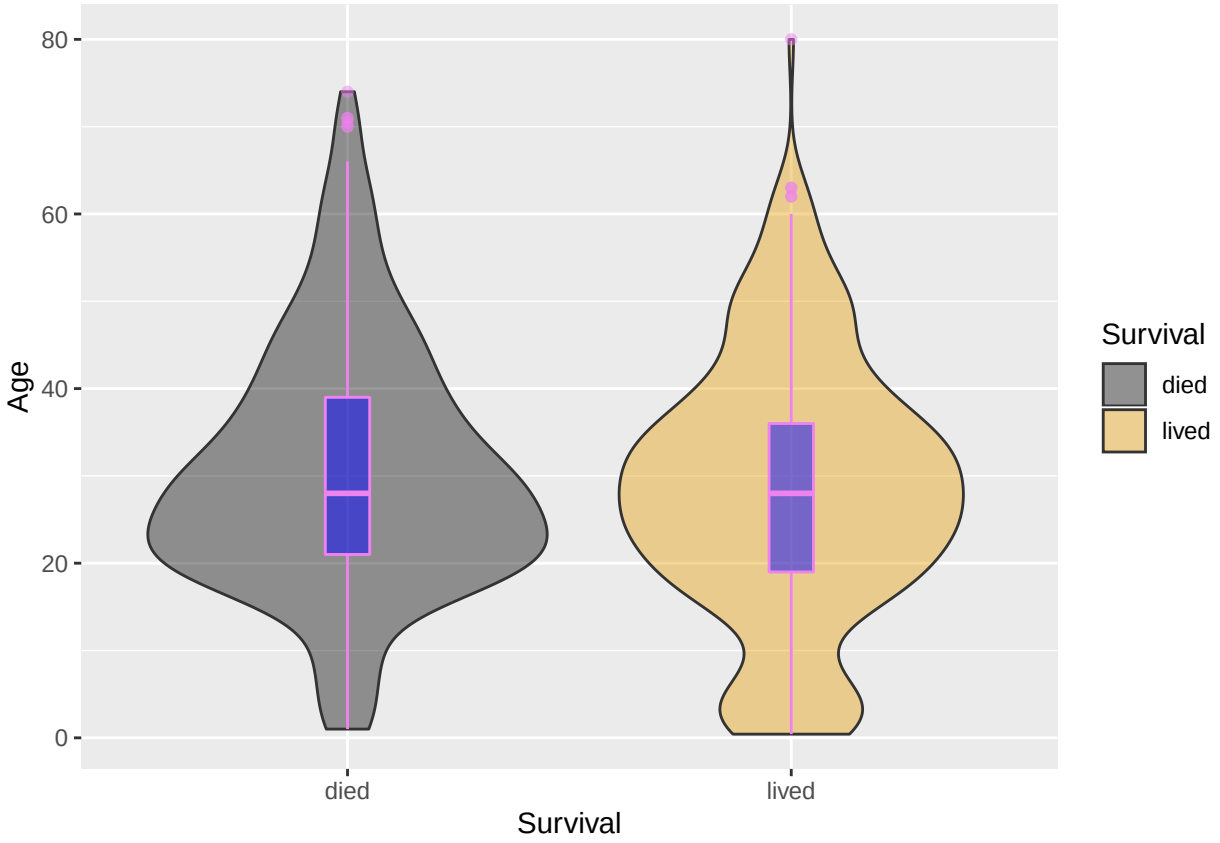
```
titanic %>%
  filter(!is.na(Age)) %>%
  ggplot() +
  geom_boxplot(aes(x = Survival, y = Age, fill = Survival))
```



We can also represent the information from the boxplot in a more visually appealing manner. Namely, we can use a violin plot. We can overlay a boxplot on top to get a "best of both worlds" type of approach. We can see a slight bulge at the bottom for the plot representing those who lived, but, this is relatively insignificant in comparison to other variables.

Again, we can set custom colors, transpaarency, and more with ggplot. More on the color palette of the graph later.

```
cbPalette = c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

titanic %>%
  filter(!is.na(Age)) %>%
  ggplot(aes(x = Survival, y = Age, fill = Survival)) +
  geom_violin(alpha = 0.4) +
  geom_boxplot(color = "violet", fill = "blue", alpha = 0.5, width = 0.1) +
  scale_fill_manual(values = cbPalette)
```



Some Conclusions

From our basic analysis of the data, it appears that the most important factors are probably Sex and Passenger Class. We did not look into the passenger names, fare and cabin because we assumed that they were linked with the passenger sex and class. That being said, some ideas for the future could include using feature engineering to aggregate, say, the passenger cabins into distinct groups and we could then perform analysis on that new feature. We could perhaps do the same with passenger names.

Some side notes:

Probably want to use a different color palette than the default provided by ggplot. Different palettes can be found online, here's a link: https://stackoverflow.com/questions/57153428/r-plot-color-combinations-that-are-colorblind-accessible

This is a color blind palette!

```
cbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
```

When Comparing...

numerical v.s. numerical

- scatterplots

categorical v.s. categorical

- contingency tables
- bar plots, mosaic plots

categorical v.s. numerical

- side-by-side boxplots