Intro to R: Lab 3-walkthrough

Simon Caton

Prep

```
library(dplyr)
```

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

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

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

```
titanicData <- read.csv("titanic.csv", header=T, na.strings=c(""), stringsAsFactors = T)
```

Spend some time to understand what the different parameters passed to read.csv do

header=T means read the first line of the csv as the names of each attribute (column) na.strings=c(“”) means all values in the vector defined by c should be replaced by NAs stringsAsFactors = T means encode all columns identified as string attributes as factors

Encode the following attributes as factors

Survived

```
titanicData$Survived <- as.factor(titanicData$Survived)
```

Pclass

```
titanicData$Pclass <- as.factor(titanicData$Pclass)
```

Get these three details (class, fare, and name) of the two passengers.

```
embarkedNAs <- titanicData %>%
  filter(PassengerId == 62 | PassengerId == 830)
```

```r
print(embarkedNAs[, c(3,10,4)])
```

```
##   Pclass Fare                                        Name
## 1      1   80                        Icard, Miss. Amelie
## 2      1   80 Stone, Mrs. George Nelson (Martha Evelyn)
```

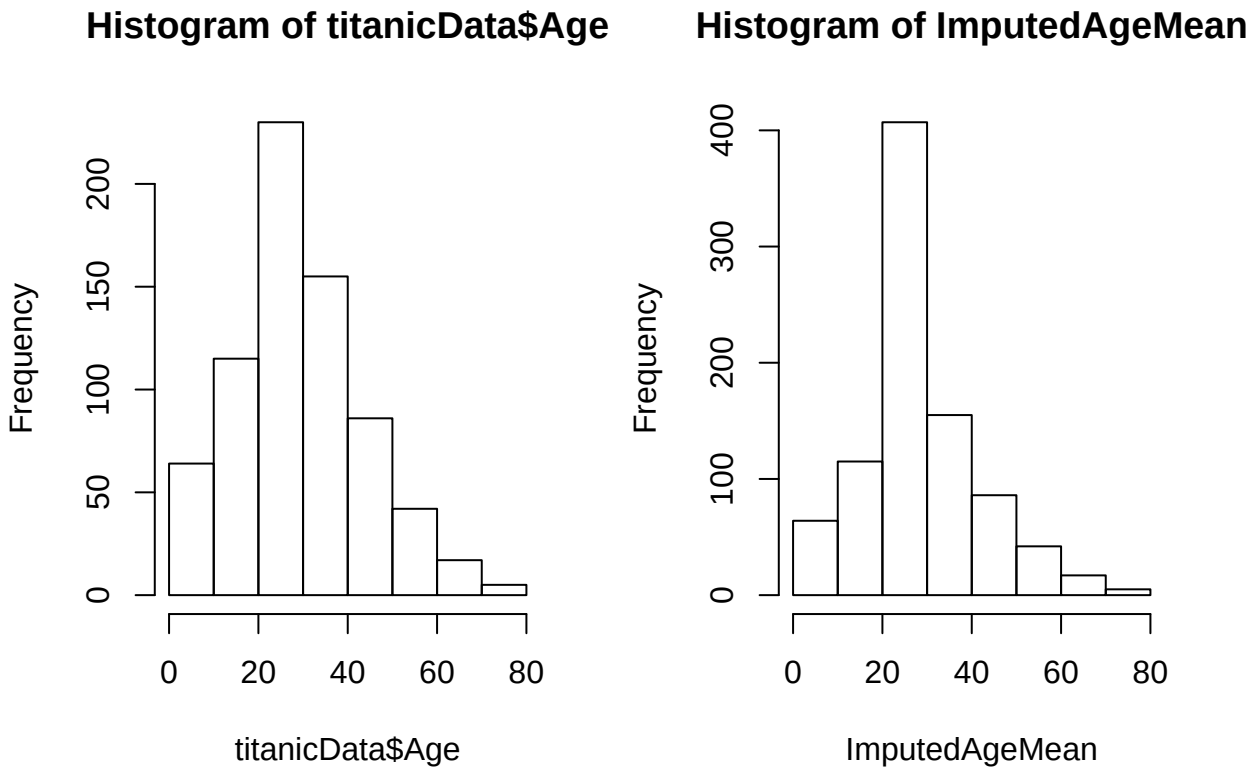
Impute the missing values of ImputedAgeMean to be the mean age

```r
ImputedAgeMean <- titanicData$Age
ImputedAgeMean[is.na(ImputedAgeMean)] <- mean(ImputedAgeMean, na.rm = TRUE)
```

If you didn't remove the NAs, you will have got a mean of: 29.6991176.

Use histograms to inspect the distribution of titanicData$Age compared to ImputedAgeMean.

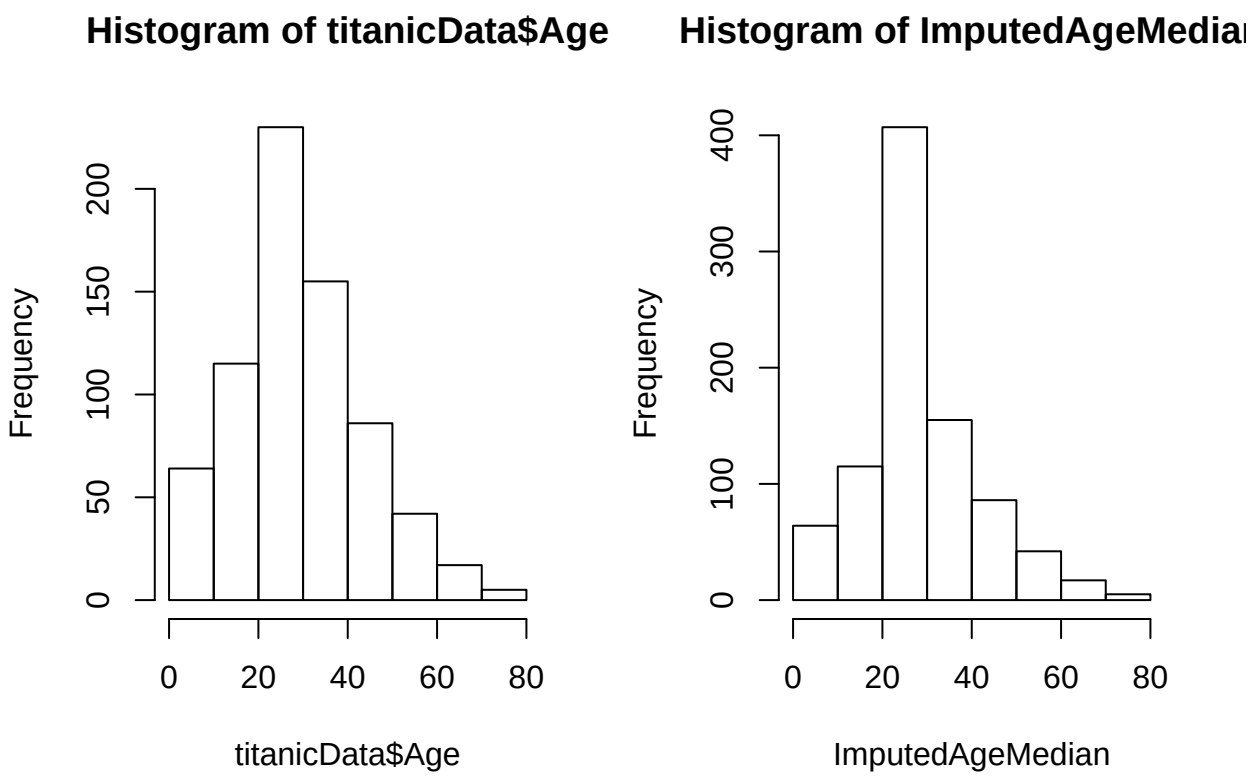
```r
par(mfrow=c(1,2))
hist(titanicData$Age)
hist(ImputedAgeMean)
```



Chosing the mean has not retained the same distribution, in fact it has made the peak (values around the mean) more frequent: compare the y-axes!!

Replace mean with median and see what difference it makes.

```
ImputedAgeMedian <- titanicData$Age
ImputedAgeMedian[is.na(ImputedAgeMedian)] <- median(ImputedAgeMedian, na.rm = TRUE)
par(mfrow=c(1,2))
hist(titanicData$Age)
hist(ImputedAgeMedian)
```



Looks largely the same, now: let's check some descriptive statistics of the data:

```
df <- data.frame(titanicData$Age, ImputedAgeMean, ImputedAgeMedian)
summary(df)
```

```
##  titanicData.Age ImputedAgeMean  ImputedAgeMedian
##  Min.   : 0.42   Min.   : 0.42   Min.   : 0.42
##  1st Qu.:20.12   1st Qu.:22.00   1st Qu.:22.00
##  Median :28.00   Median :29.70   Median :28.00
##  Mean   :29.70   Mean   :29.70   Mean   :29.36
##  3rd Qu.:38.00   3rd Qu.:35.00   3rd Qu.:35.00
##  Max.   :80.00   Max.   :80.00   Max.   :80.00
##  NA's   :177
```

So, what we see is that in the case of imputing via the mean, the mean of the data is preserved, likewise for the median, the median of the data is preserved. If instead we inspect the standard deviation:

```
sapply(df, function(x) sd(x, na.rm=T))
```

```
##  titanicData.Age   ImputedAgeMean ImputedAgeMedian
##         14.52650         13.00202         13.01970
```

We see that both methods of imputation have *tightened* up the standard deviation of Age (or more formally

reduced the variance). This is essentially a somewhat lazy way of imputation, but you will see it a lot online as a method "*of choice*", however, invariably model-based methods such as using linear regression or packages like *mice* are the better choice. Let's check, first we need to build the LM and mice models.

```
idx_na <- is.na(titanicData$Age)
age_train <- titanicData[-idx_na, ]
age_test <- titanicData[idx_na, ]

ageModel <-lm(Age~Pclass + Survived + SibSp, data = age_train)
age_test$Age <- predict(ageModel, newdata = age_test)

ImputedAgeLM <- titanicData

ImputedAgeLM[ImputedAgeLM$PassengerId %in% age_test$PassengerId, "Age"] <- age_test$Age

library(mice)
# Perform mice imputation, excluding some variables that probably won't help:
mice_mod <- mice(titanicData[, !names(titanicData) %in%
     c('PassengerId','Name','Ticket','Cabin','Survived')], method='rf')
```

```
##
##  iter imp variable
##   1   1  Age  Embarked
##   1   2  Age  Embarked
##   1   3  Age  Embarked
##   1   4  Age  Embarked
##   1   5  Age  Embarked
##   2   1  Age  Embarked
##   2   2  Age  Embarked
##   2   3  Age  Embarked
##   2   4  Age  Embarked
##   2   5  Age  Embarked
##   3   1  Age  Embarked
##   3   2  Age  Embarked
##   3   3  Age  Embarked
##   3   4  Age  Embarked
##   3   5  Age  Embarked
##   4   1  Age  Embarked
##   4   2  Age  Embarked
##   4   3  Age  Embarked
##   4   4  Age  Embarked
##   4   5  Age  Embarked
##   5   1  Age  Embarked
##   5   2  Age  Embarked
##   5   3  Age  Embarked
##   5   4  Age  Embarked
##   5   5  Age  Embarked
```

```
# Complete the missing values
mice_output <- complete(mice_mod)

df <- data.frame(titanicData$Age, ImputedAgeMean, ImputedAgeMedian,
                 ImputedAgeLM$Age, mice_output$Age)
```

Now, rebuild our comparision data frame, and look

```
summary(df)
```

```
##  titanicData.Age ImputedAgeMean  ImputedAgeMedian ImputedAgeLM.Age
##  Min.   : 0.42   Min.   : 0.42   Min.   : 0.42    Min.   :-6.226
##  1st Qu.:20.12   1st Qu.:22.00   1st Qu.:22.00    1st Qu.:21.000
##  Median :28.00   Median :29.70   Median :28.00    Median :29.441
##  Mean   :29.70   Mean   :29.70   Mean   :29.36    Mean   :29.327
##  3rd Qu.:38.00   3rd Qu.:35.00   3rd Qu.:35.00    3rd Qu.:36.000
##  Max.   :80.00   Max.   :80.00   Max.   :80.00    Max.   :80.000
##  NA's   :177
##  mice_output.Age
##  Min.   : 0.42
##  1st Qu.:20.00
##  Median :28.00
##  Mean   :29.43
##  3rd Qu.:38.00
##  Max.   :80.00
##
```

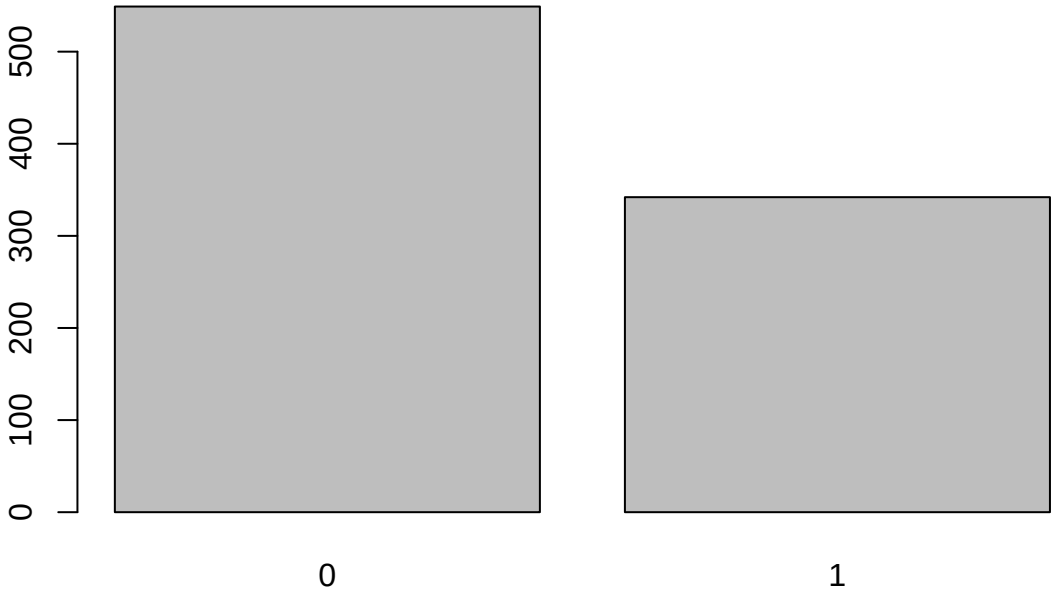
```
sapply(df, function(x) sd(x, na.rm=T))
```

```
##  titanicData.Age   ImputedAgeMean ImputedAgeMedian ImputedAgeLM.Age
##         14.52650         13.00202         13.01970         13.73605
##  mice_output.Age
##         14.36608
```

So, we see that the LM approach has produced a few negative ages – we clearly don't want that! The mice model, however, is pretty good, the mean, and standard deviation are more or less retained.

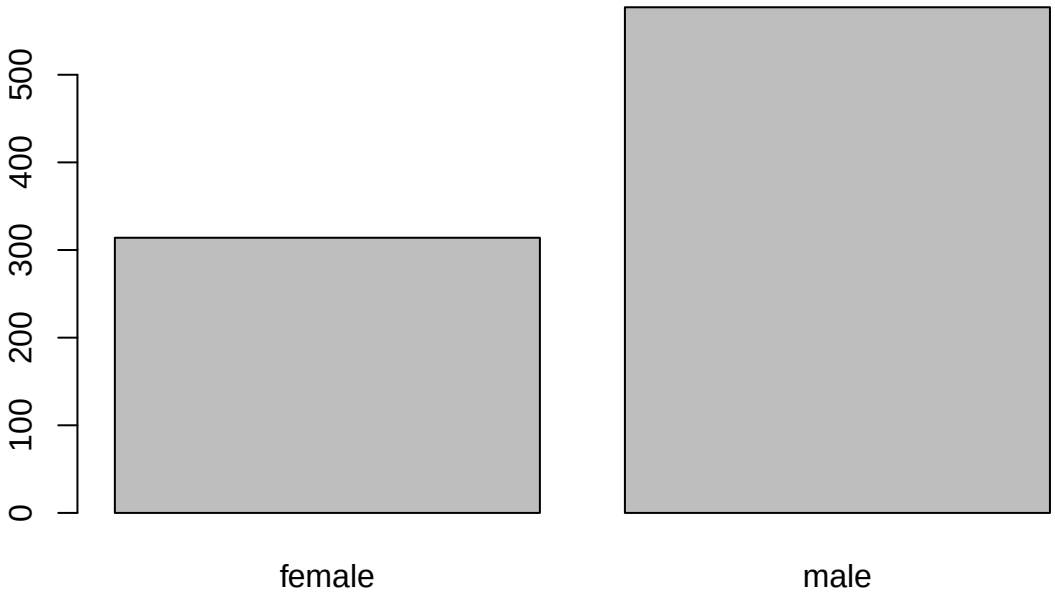
Make a barplot of the survived variable, to gauge the survival rate.

```
barplot(table(titanicData$Survived))
```



Make a barplot of the Sex variable, to see the distribution of gender.

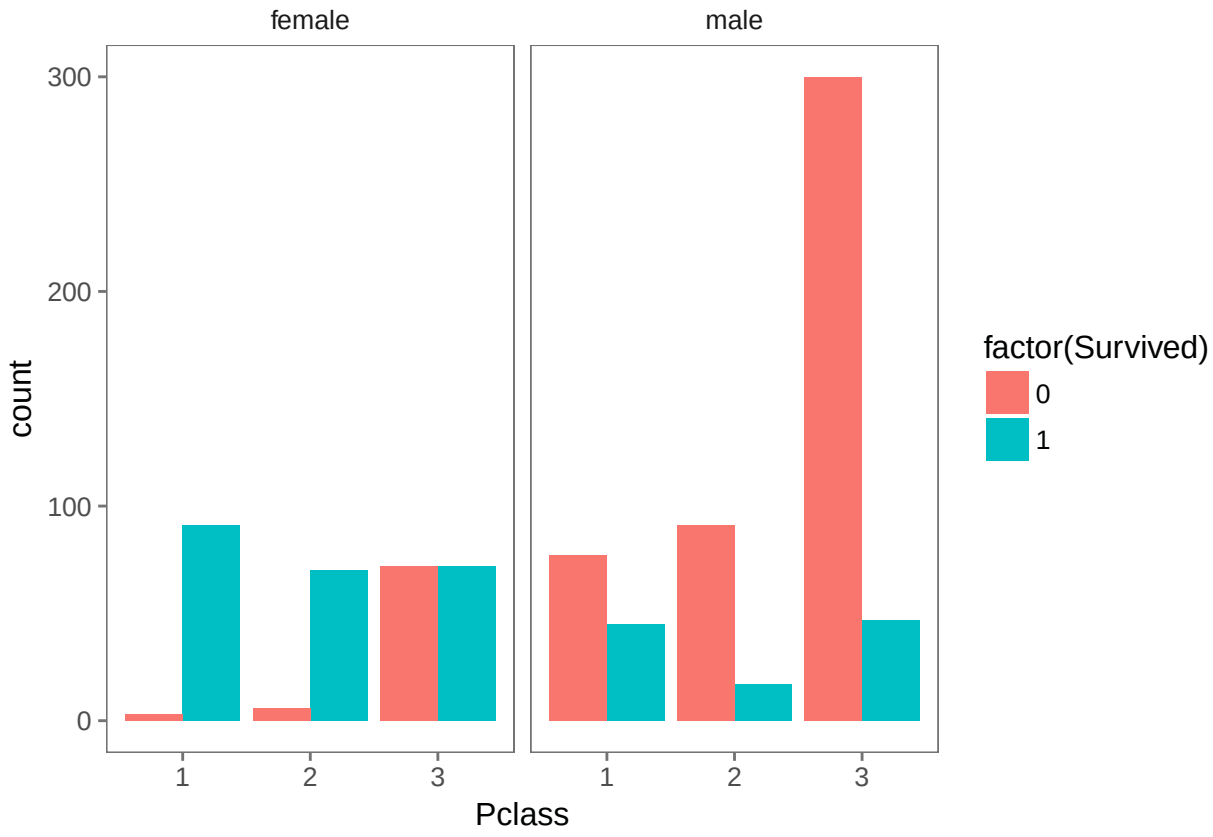
```
barplot(table(titanicData$Sex))
```



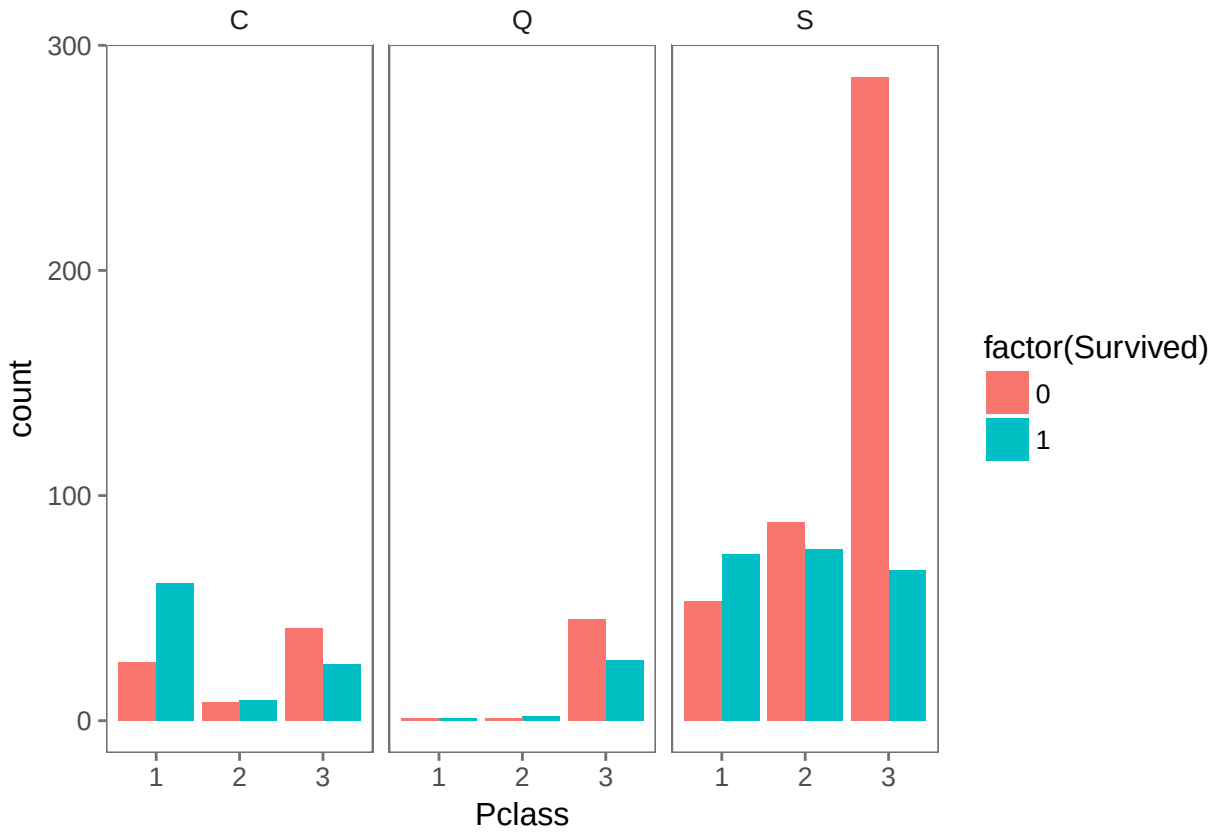
Try out different combinations of Survived with other explanatory variables

```
library(ggplot2)
library(ggthemes)

ggplot(titanicData, aes(Pclass, fill = factor(Survived))) +
  geom_bar(stat='count', position='dodge') +
  facet_grid(.~Sex) +
  theme_few()
```

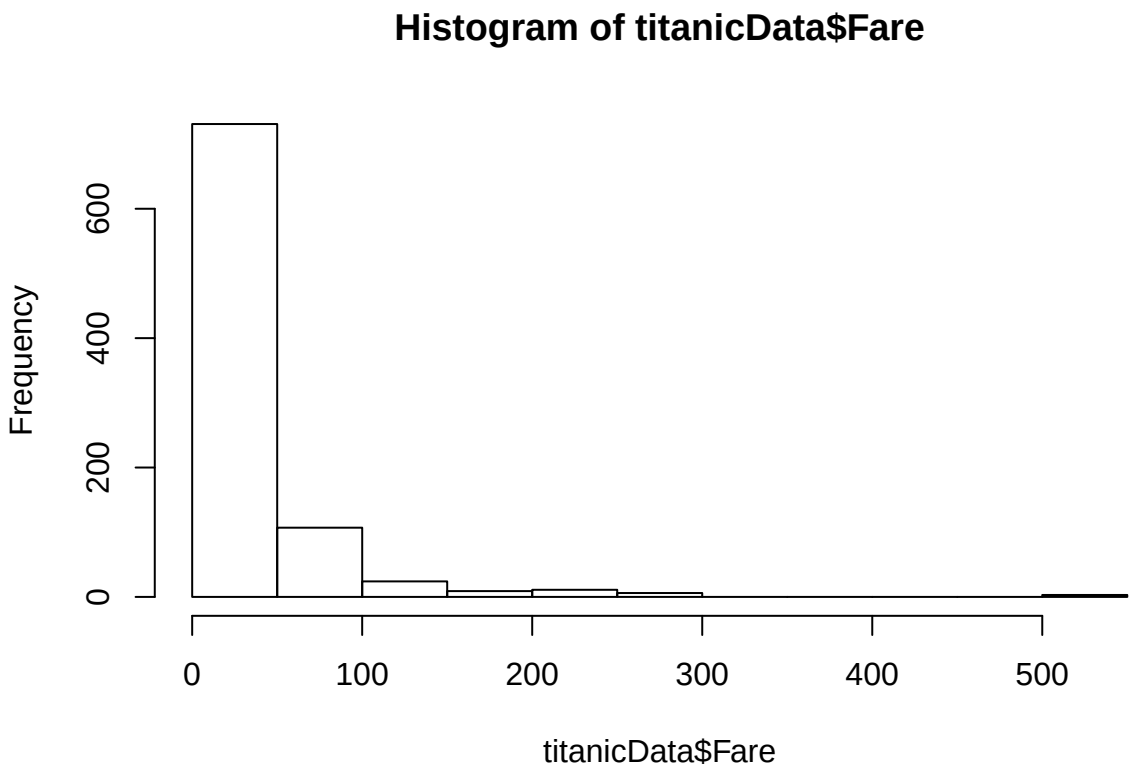



```
ggplot(titanicData, aes(Pclass, fill = factor(Survived))) +
  geom_bar(stat='count', position='dodge') +
  facet_grid(.~Embarked) +
  theme_few()
```



On the chance that you tried to put Fare on the x-axis, you probably found that it looks ridiculous. Its range (0 to 512.3292) means you have too many possible x values. So, what can we do in such a case? Well, we can create a new variable that considers groups of values. For this we'll use a function called *cut*.

```
#first inspect Fare
hist(titanicData$Fare)
```

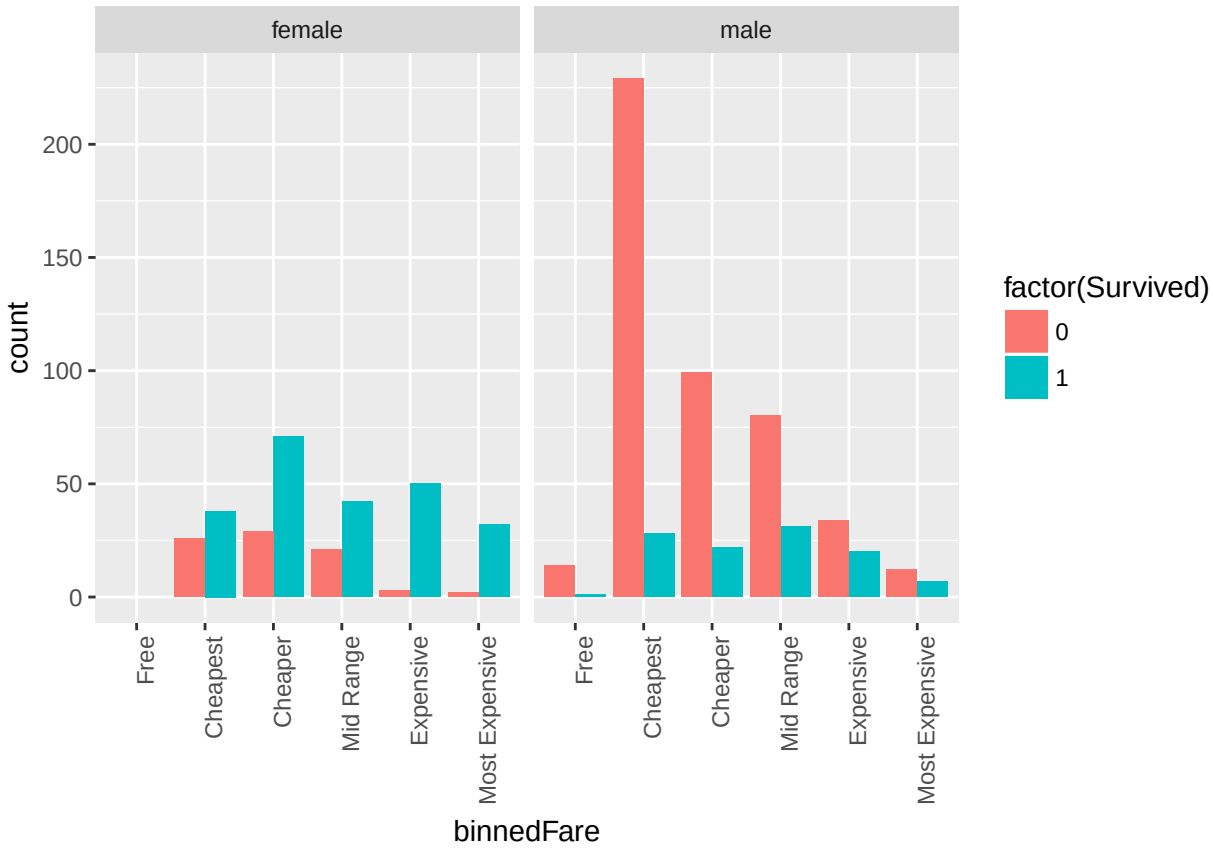


Lot of values between 0 and 50. A few between 50 and 100. Not so many above that. Note we also have some people that apparently paid nothing for their ticket. Let's make group as follows:

- 0: Free
- 1 – 10: Cheapest
- 11 – 25: Cheaper
- 26 – 50: Mid Range
- 51 – 100: Expensive
- 101 and greater: Most Expensive

```
titanicData$binnedFare <- cut(titanicData$Fare, breaks=c(-1, 0, 10, 25, 50, 100, 600),
  labels = c("Free", "Cheapest", "Cheaper", "Mid Range", "Expensive", "Most Expensive"))

ggplot(titanicData, aes(binnedFare, fill = factor(Survived))) +
  geom_bar(stat='count', position='dodge') +
  facet_grid(.~Sex) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

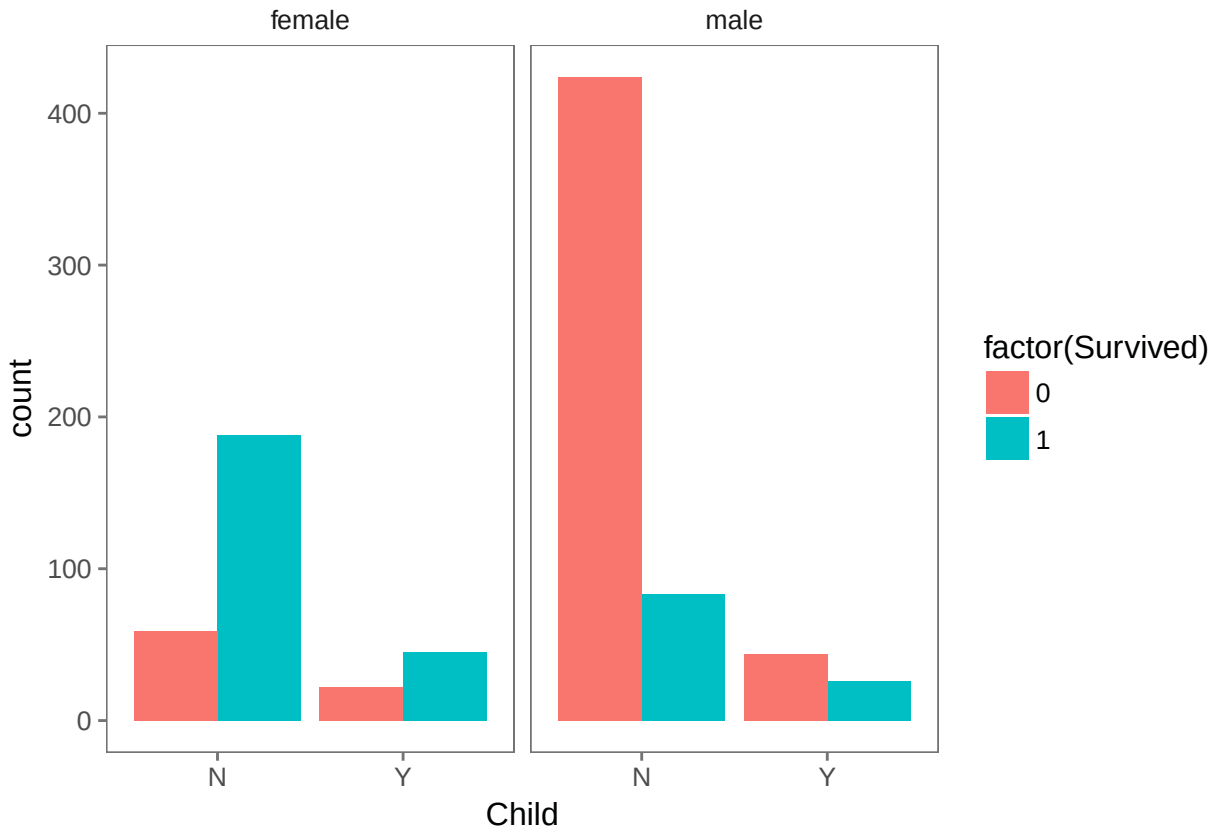


Create a new factor called Child

```
titanicData$Child[titanicData$Age < 18] <- "Y"
titanicData$Child[titanicData$Age >= 18] <- "N"
titanicData$Child <- as.factor(titanicData$Child)
```

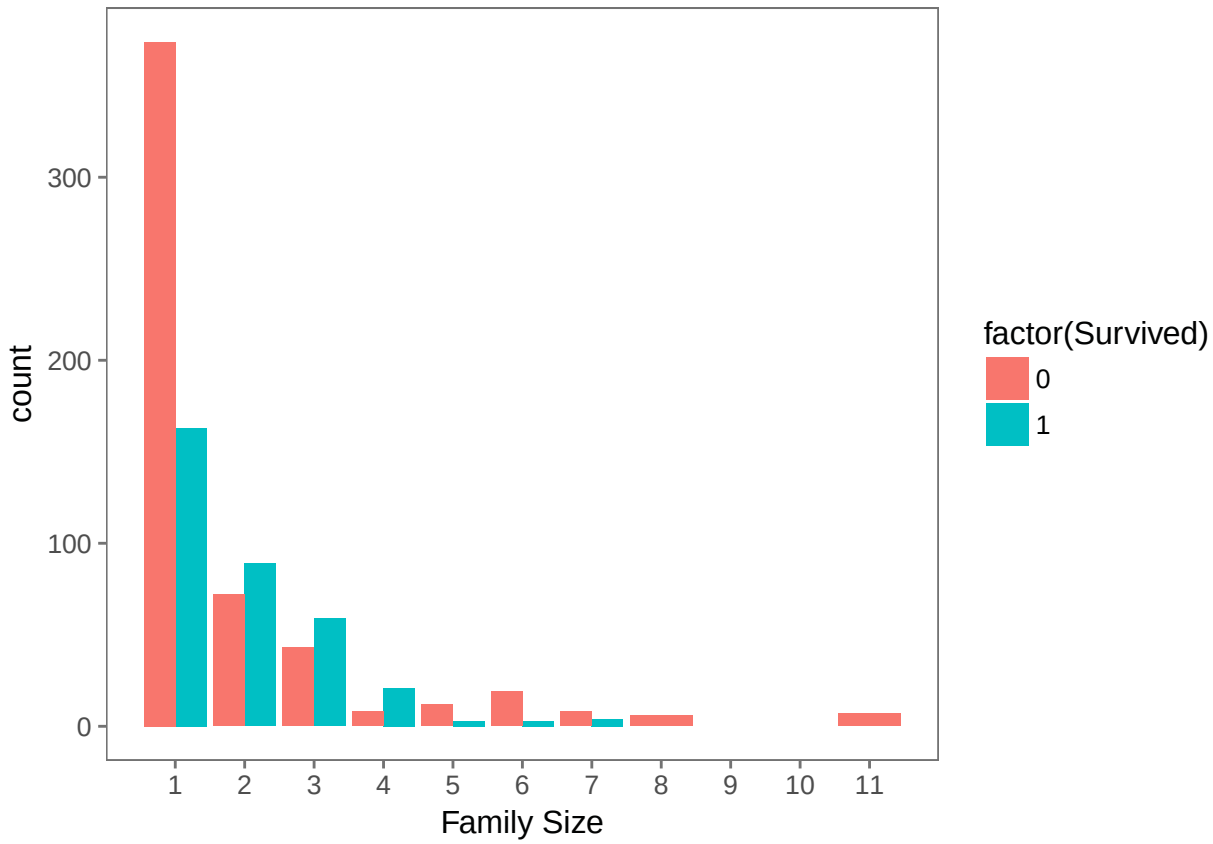
Create the plot below with the new Child column.

```
ggplot(titanicData, aes(Child, fill = factor(Survived))) +
  geom_bar(stat='count', position='dodge') +
  facet_grid(.~Sex) +
  theme_few()
```



Make the plot below, to help us understand if Fsize has any relationship with survival

```
ggplot(titanicData, aes(x = Fsize, fill = factor(Survived))) +
  geom_bar(stat='count', position='dodge') +
  scale_x_continuous(breaks=c(1:11)) +
  labs(x = 'Family Size') +
  theme_few()
```



Make a variable called FsizeD

```
titanicData$FsizeD[titanicData$Fsize == 1] <- 'singleton'
titanicData$FsizeD[titanicData$Fsize < 5 & titanicData$Fsize > 1] <- 'small'
titanicData$FsizeD[titanicData$Fsize > 4] <- 'large'
titanicData$FsizeD <- as.factor(titanicData$FsizeD)

table(titanicData$FsizeD)
```

```
##
##     large singleton     small
##        62       537       292
```

Incidentally, if the order in the factor is important, we can also instruct R to capture that:

```
titanicData$FsizeO[titanicData$Fsize == 1] <- 'singleton'
titanicData$FsizeO[titanicData$Fsize < 5 & titanicData$Fsize > 1] <- 'small'
titanicData$FsizeO[titanicData$Fsize > 4] <- 'large'

titanicData$FsizeO <- factor(titanicData$FsizeO, ordered = TRUE,
  levels = c("singleton", "small", "large"))

table(titanicData$FsizeO)
```

```
##
## singleton     small     large
```

```
##         537         292          62
```