Tables Lab

For this lab I want only the Questions and your answers turned in to Gradescope in .pdf format. I recommend creating a new file with an appropriate name, and writing down (or copy-pasting) only the questions and answers in it.

Introduction

Titanic Data

We will be using the `Titanic` dataset built into R, providing information on the fate of the passengers on the fatal maiden voyage of the ocean liner Titanic summarized according to economic status (class), sex, age, and survival. See `?Titanic` for more details.

As the `Titanic` dataset in R is, by default, stored as a 4 dimensional array, we will start by transforming it into a data.frame.

```r
library(ggplot2)
library(dplyr)

theme_set(theme_bw())


## Data for lab
data(Titanic)
titanic <- as.data.frame(Titanic)
titanic <- titanic[rep(1:nrow(titanic), times = titanic$Freq), ]
titanic$Freq <- NULL

## head() shows us the first 6 rows of a data.frame
head(titanic)
```

```
##     Class  Sex   Age Survived
## 3     3rd Male Child       No
## 3.1   3rd Male Child       No
## 3.2   3rd Male Child       No
## 3.3   3rd Male Child       No
## 3.4   3rd Male Child       No
## 3.5   3rd Male Child       No
```

Our goal in this lab is to familiarize ourselves with the use of tables in R. To this end, there are two primary functions we will be concerning ourselves with:

- The `table()` function which returns a count of categorical data
- The `proportion()` function which takes a table as an argument and returns a (optionally conditional) table of proportions

In addition to these, we will introduce four helper functions to assist us:

- The "pipe" operator, `%>%` (Ctrl+Shift+M), which takes the results of the left hand side of the pipe and passes them to the right hand side. This is helpful for "chaining together" a sequence of operations. We will use this extensively later in the semester
- The `sort()` function will sort either a table or numeric vector in increasing or decreasing order
- The `addmargins()` function will sum the margins of a table
- We can use `facet_grid` to facet two nested categorical variables in `ggplot2`

You can learn more about these functions, and see some additional examples, by using the help documentation (i.e., `?table()`).

"Pipe" Operator

The pipe operator, `%>%` (Ctrl+Shift+M) is a special type of operator (similar to `+`, for example) that is included in the R package `dplyr` which we can load into our R session with `library(dplyr)`. While not essential, the pipe operator facilitates interactive programming by "chaining together" sequences of operations; it works by taking the output on the left hand side and putting it as the first argument on the right hand side.

For example, the standard deviation of a vector is the square root of its variance. Consider two ways that we could do this, with and without the pipe operator

```r
## Create random vector x with 50 entries
# (rnorm = random normal)
x <- rnorm(50)

## Without pipe
sqrt(var(x))
```

```
## [1] 0.9191973
```

```r
## With pipe
var(x) %>% sqrt()
```

```
## [1] 0.9191973
```

Using Tables

The first part of this lab will be oriented around the construction and manipulation of one- and two-way tables in R.

One-way tables

The first function we will introduce is the `table()` function; in its most basic form, `table()` takes as its argument a single vector, returning a named numeric vector detailing the quantities of each category. There are two ways to express this in R, though I tend to prefer the first. This is both because the first method prints out the name of the variable corresponding to the table and it omits the ungainly use of `$` for selecting variables (this issue is more evident when considering two-way tables, as we will see):

```
## How many people lived or died on the Titanic
with(titanic, table(Survived))
```

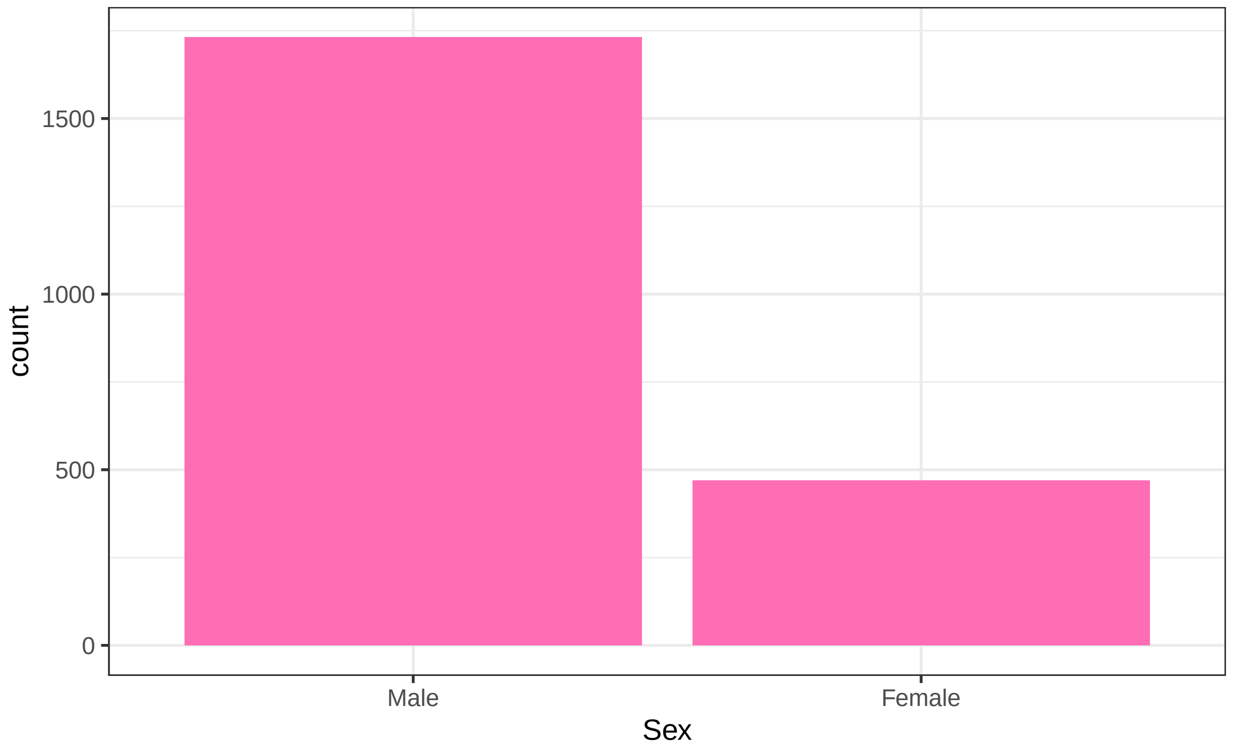
```
## Survived
##   No  Yes
## 1490  711
```

```
## How many males and females were on the Titanic
table(titanic$Sex)
```

```
##
##   Male Female
##   1731    470
```

Each gives us an example of a *frequency table*. Note that this corresponds directly with a univariate bar chart that we saw previously

```
ggplot(titanic, aes(Sex)) + geom_bar(fill = 'hotpink1')
```



Additionally, we see in both cases that all of our observations are included in the count, as both of the table totals sum up to 2201. We can verify this with the helper function `addmargins` which will include a "Sum" column adding up the table

```r
## Assign table to tab variable
tab <- with(titanic, table(Survived))

## Adds the Sum value
addmargins(tab)
```

```
## Survived
##   No  Yes  Sum
## 1490  711 2201
```

Notice how in this case we first assigned the result of `table()` to a variable, `tab` (though you could name it whatever you wanted) and then passed that variable to `addmargins()`. This is an excellent example of a process that is facilitated with the pipe operator, `%>%` (Ctrl+Shift+M):

```r
## This is equivalent to what we saw above
with(titanic, table(Survived)) %>% addmargins()
```

```
## Survived
##   No  Yes  Sum
## 1490  711 2201
```

We may also be interested in identifying the *proportion* of individuals who survived or died on the Titanic. Similar to the `addmargins()` function, we can pass the results of `table()` to the function `proportions()`

```r
with(titanic, table(Survived)) %>% proportions()
```

```
## Survived
##       No      Yes
## 0.676965 0.323035
```

Here we see that 68% of the passengers aboard the Titanic died while 32% survived. This is the same information that we would have found had we computed the values by hand:

$$\% \text{ Dead} = \frac{\#\text{Dead}}{\text{Total Passengers}} = \frac{1490}{2201} = 0.677$$

Finally, let's introduce the `sort()` function, which takes either a one-way table or a vector and sorts the values from smallest to largest (or in alphabetical order if they are character strings)

```r
## Unsorted table
with(titanic, table(Class))
```

```
## Class
##  1st  2nd  3rd Crew
##  325  285  706  885
```

```r
## Table sorted smallest to largest
with(titanic, table(Class)) %>% sort()
```

```
## Class
##  2nd  1st  3rd Crew
##  285  325  706  885
```

```r
## Table sorted largest to smallest
with(titanic, table(Class)) %>% sort(decreasing = TRUE)
```

```
## Class
## Crew  3rd  1st  2nd
##  885  706  325  285
```

There are a few things to note about the `sort()` function:

1. Like `addmargins()` and `proportions()`, we can pass an argument to `sort()` using the pipe operator `%>%`
2. By default, `sort()` reorders observations from smallest to largest. We can change this behavior by adding `decreasing = TRUE` as an argument to `sort()` as we did above
3. When we have a *named vector*, i.e., a set of values with names, such as the case with the table above, `sort()` will sort the vector according to its values, not its names.

Let's practice using some of these functions on the Titanic dataset.

Question 1 **Part A** Create a *frequency table* using the `titanic` data set to find how many children and adults were on board the Titanic.

Part B Determine what percentage of the passengers on-board the Titanic were adults.

Part C Determine what percentage of the passengers on-board the Titanic were members of the crew.

Two-way Tables

The two-way table, as the name suggests, is a table of two categorical variables. In R, this can be done by passing each of the two vectors in as arguments, with the first vector becoming the rows and the second becoming the column:

```r
## Sex as row
with(titanic, table(Sex, Survived))
```

```
##         Survived
## Sex        No  Yes
##   Male   1364  367
##   Female  126  344
```

```r
## Sex as column
with(titanic, table(Survived, Sex))
```

```
##          Sex
## Survived Male Female
##      No  1364    126
##      Yes  367    344
```

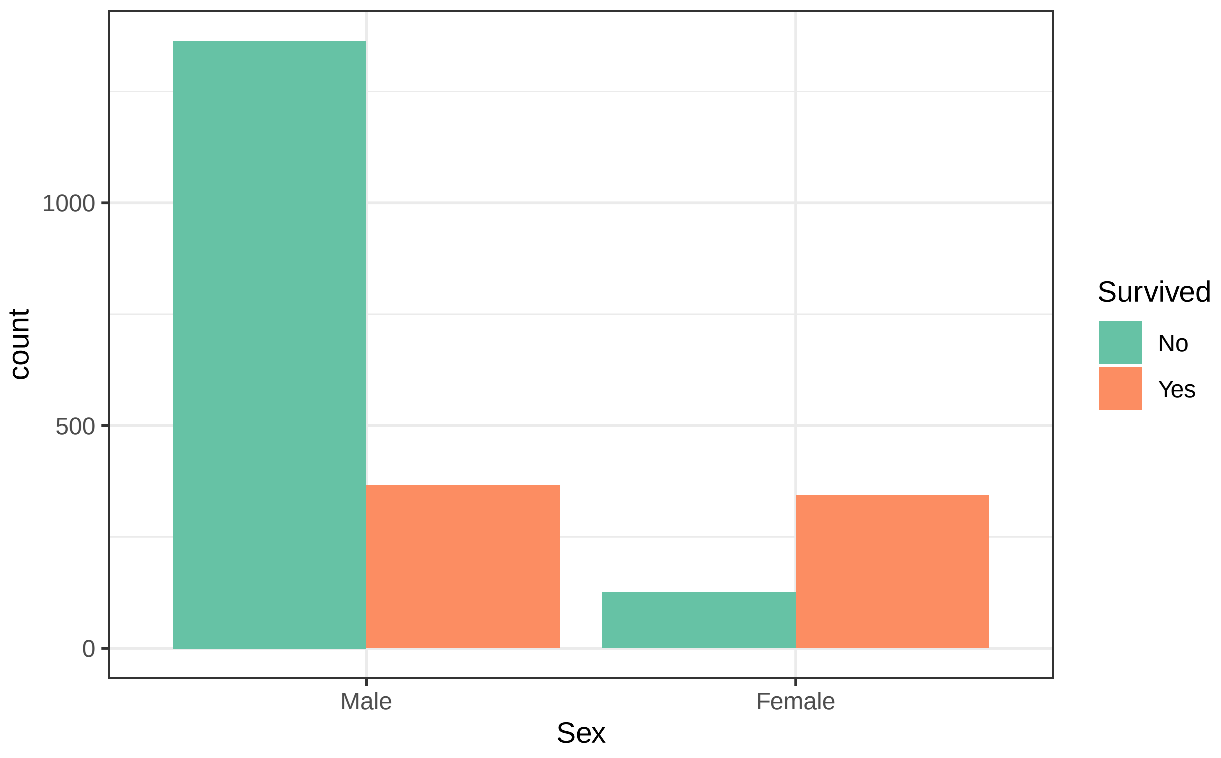
You can, as before, use only the `table()` function with the `$` operator to extract vectors, though in addition to losing the variable name in the table, you also have to duplicate much of your typing

```r
table(titanic$Sex, titanic$Survived)
```

```
##
##            No  Yes
##   Male   1364  367
##   Female  126  344
```

The two-way table gives us frequencies of the cross-sections of groups: for example, in the one-way table we saw that there were 470 women on-board the titanic. By also including the `Survived` variable, I can see that of the 470 women, 126 died while 344 lived. Additionally, of the 711 individuals who survived, the two-way table shows me that 367 were men and 344 were women. Note that this table corresponds to both of the following plots (note that it is customary for the row variable to serve as the x-axis)

```r
## See ?scale_fill_brewer for more palette colors
ggplot(titanic, aes(x = Sex, fill = Survived)) +
  geom_bar(position = "dodge") +
  scale_fill_brewer(palette = "Set2")
```



We can recover the information from the one-way tables by using `addmargins()` to give us row and column totals

```r
## Add row and column margins to  plots
with(titanic, table(Sex, Survived)) %>% addmargins()
```

```
##         Survived
## Sex        No  Yes  Sum
##   Male   1364  367 1731
##   Female  126  344  470
##   Sum    1490  711 2201
```

We should see that the Sum row and columns are exactly the one-way tables we created in the last section for their corresponding variables. In both cases, they sum to 2201 in the bottom right corner.

Just as before, we can pass a two-way table to the `proportions()` function to return a table of proportions:

```r
with(titanic, table(Sex, Survived)) %>% proportions()
```

```
##         Survived
## Sex              No        Yes
##   Male   0.61971831 0.16674239
##   Female 0.05724671 0.15629259
```

By default, this returns *absolute proportions*, calculated by dividing each of the entries in the two-way table by its total of 2201. This table of proportions tells us, for example, that *of all of the passengers* who were on the Titanic, 62% of them were males who died.

We can specify *conditional proportions* by passing in an additional argument called `margins` to the `proportions()` function. In R, 1 refers to rows and 2 refers to columns: so, in order to compute a table of proportions conditional on the row variable (meaning that proportions are taken *within* the row), we will pass the argument `margin = 1` to the `proportion()` function:

```r
# Compute row proportions
with(titanic, table(Sex, Survived)) %>% proportions(margin = 1)
```

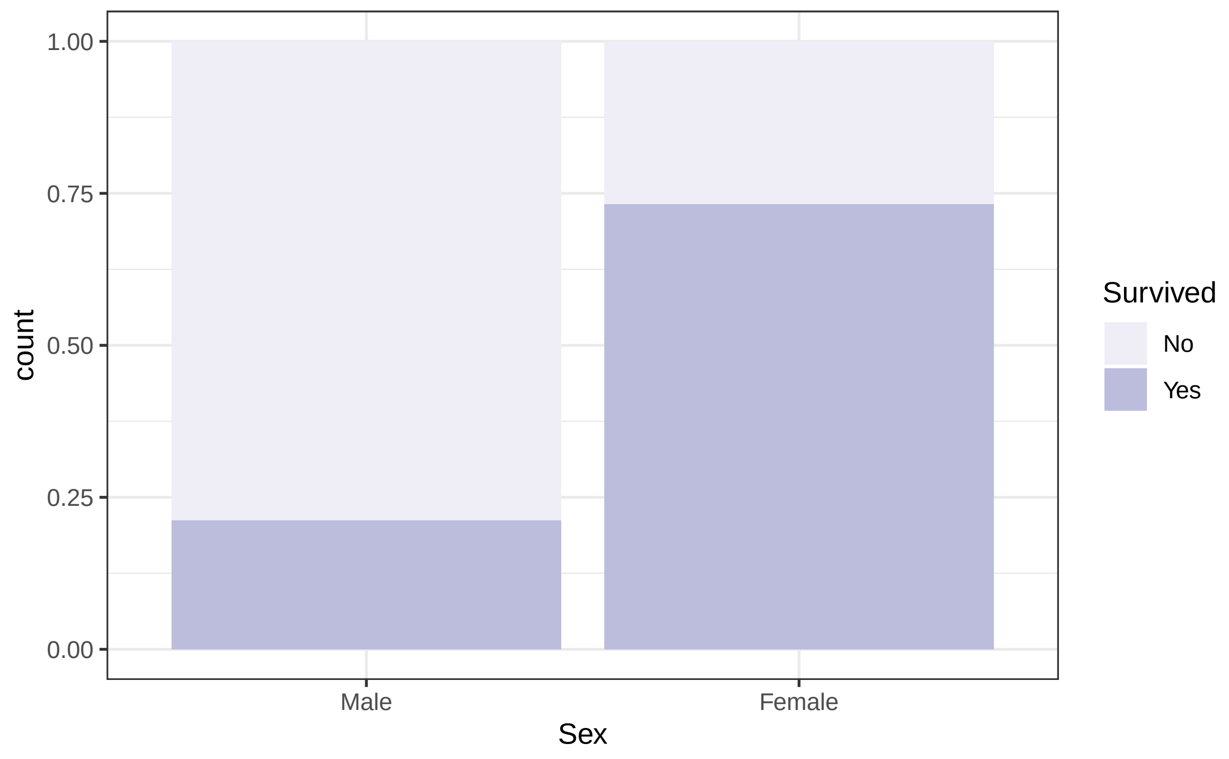
```
##         Survived
## Sex             No       Yes
##   Male   0.7879838 0.2120162
##   Female 0.2680851 0.7319149
```

We can be sure that we are conditioning on the row margins because the sum of each row is equal to 1.

From this, we see that *given that an individual was male*, 21% survived while 79% did not. Similarly, *given that an individual was female*, we see that 27% were cast to a watery grave while 73% went to normal graves at some other point in their lives. In this example, we are *conditioning* on sex, our explanatory variable, with survival status serving as our response.

The variable we condition on should also serve as the x-axis in any conditional plots we make

```r
ggplot(titanic, aes(Sex, fill = Survived)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Purples")
```



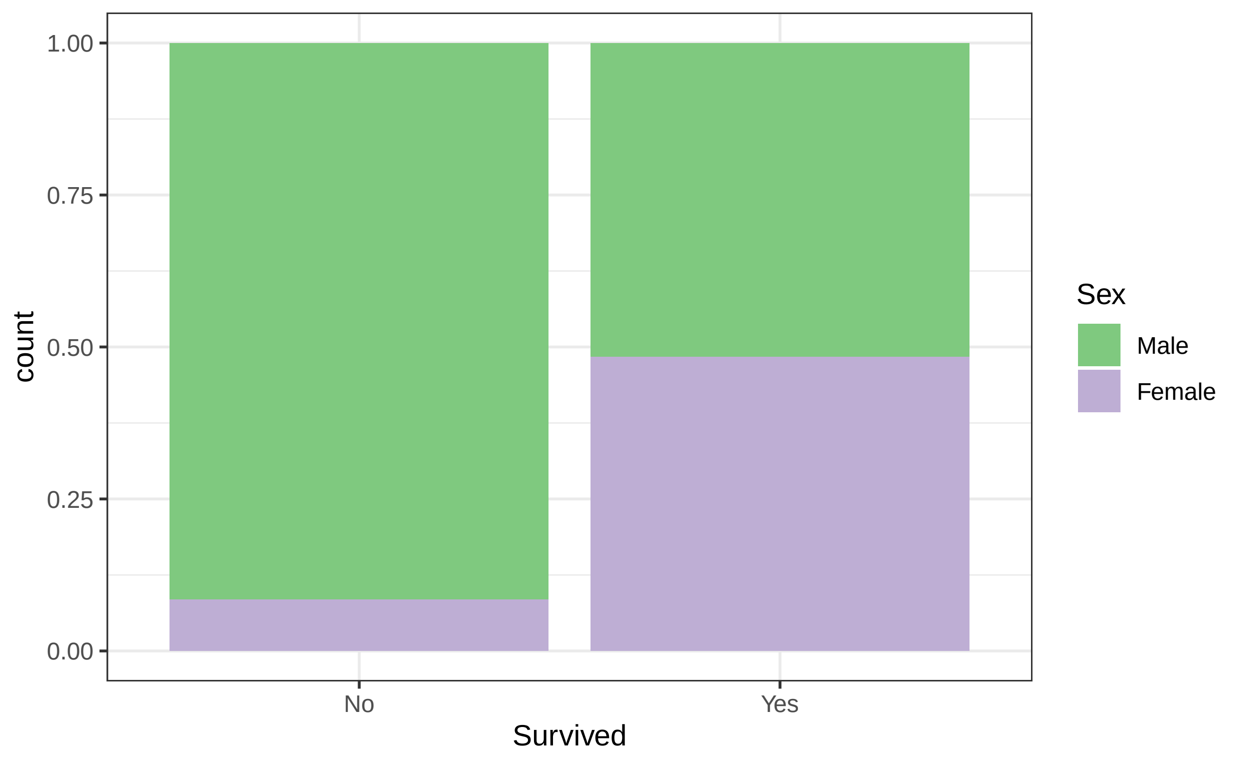
Finding column proportions works the same way, passing `margin = 2` into our function instead

```r
# Compute column proportions
with(titanic, table(Sex, Survived)) %>% proportions(margin = 2)
```

```
##         Survived
## Sex              No        Yes
##   Male   0.91543624 0.51617440
##   Female 0.08456376 0.48382560
```

Doing so replaces which variable we include as the x-axis on our plot

```r
ggplot(titanic, aes(Survived, fill = Sex)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Accent")
```



We will wrap up this section by showing a slightly more detailed use of `addmargins()` for two-way tables. Just like `proportions()`, `addmargins()` also takes an argument telling it which margin to take the sum, though it is unfortunately backwards from what it should be based on the `proportions()` function:

```
## Row and column margins
with(titanic, table(Sex, Age)) %>% addmargins()
```

```
##         Age
## Sex       Child Adult  Sum
##   Male       64  1667 1731
##   Female     45   425  470
##   Sum       109  2092 2201
```

```
## Adds a row for sums (across the columns)
with(titanic, table(Sex, Age)) %>% addmargins(margin = 1)
```

```
##         Age
## Sex       Child Adult
##   Male       64  1667
##   Female     45   425
##   Sum       109  2092
```

```
## Adds a column for sums (across the rows)
with(titanic, table(Sex, Age)) %>% addmargins(margin = 2)
```

```
##         Age
## Sex      Child Adult  Sum
##   Male      64  1667 1731
##   Female    45   425  470
```

Let's conclude this section with a little bit of practice.

Question 2 **Part A** How many children were included in second class?

Part B What percentage of the crew survived? How about children?

Part C What proportion of individuals who survived were members of the crew? Construct the plot associated with the table you create.

Three-way tables

A natural extension of the two-way table is the three-way table (and four-way, and so on). These differ from the two-way and one-way tables in that switching the order of the variables is no longer as simple as changing out the rows and columns. We won't be asked to do much with three-way tables, but it is worth considering what information can be gained from them. Consider for example the two table below:

```r
## Table 1
with(titanic, table(Class, Sex, Survived))
```

```
## , , Survived = No
## 
##        Sex
## Class  Male Female
##   1st   118      4
##   2nd   154     13
##   3rd   422    106
##   Crew  670      3
## 
## , , Survived = Yes
## 
##        Sex
## Class  Male Female
##   1st    62    141
##   2nd    25     93
##   3rd    88     90
##   Crew  192     20
```

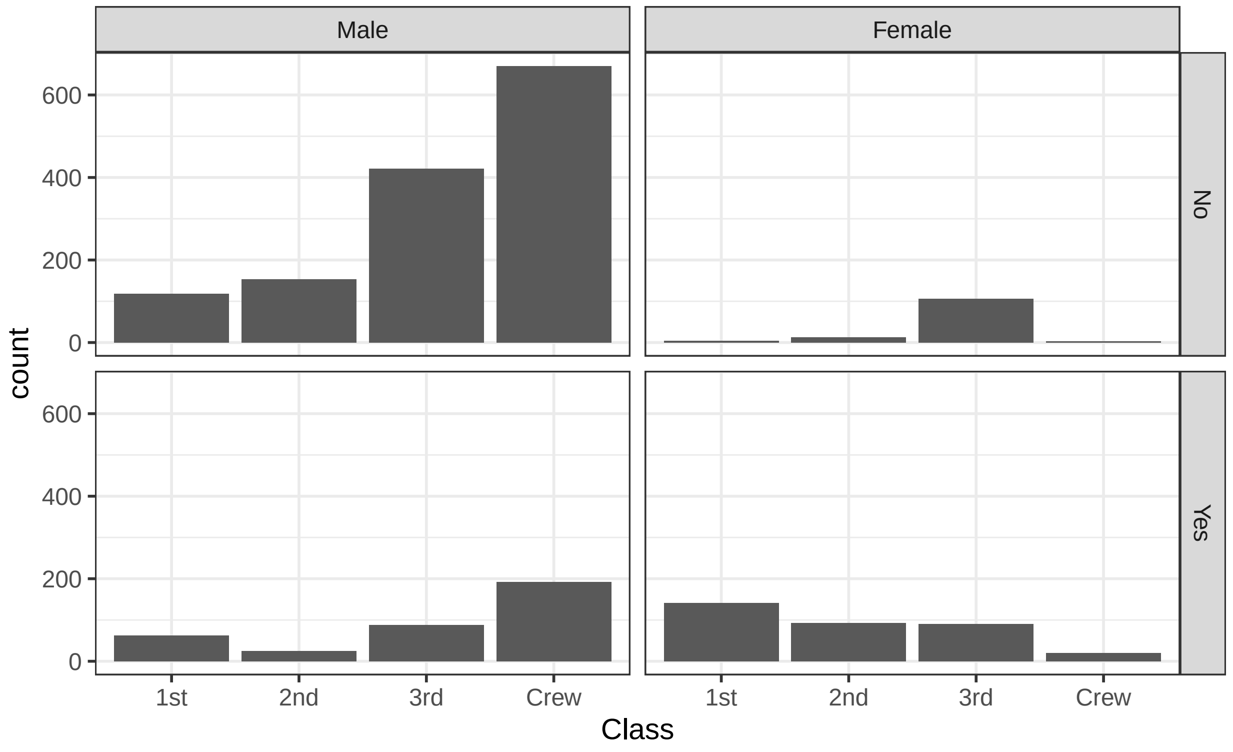
```r
## Table 2
with(titanic, table(Sex, Class, Survived))
```

```
## , , Survived = No
## 
##          Class
## Sex       1st 2nd 3rd Crew
##   Male    118 154 422  670
##   Female    4  13 106    3
```

```
##
## , , Survived = Yes
##
##         Class
## Sex       1st 2nd 3rd Crew
##   Male     62  25  88  192
##   Female 141  93  90   20
```

Complimentary to the three-way table, we introduce here the `ggplot2` function `facet_grid()`; it works just as `facet_wrap()`, except it takes *two* categorical variables and creates a grid of facets

```
ggplot(titanic, aes(Class)) +
  geom_bar() +
  facet_grid(Survived ~ Sex)
```



Question 3 Use the `facet_grid` chart or Table 1 or Table 2 above to answer:

Part A: Amongst female passengers, which class had the most who did not survive? How many female passengers in this class did not survive?

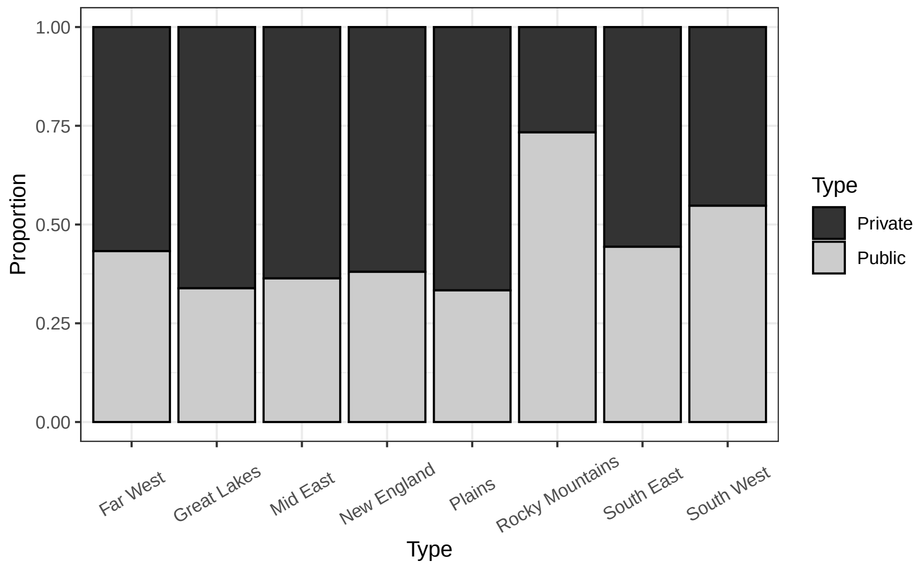
Part B: Amongst male passengers, which class had the fewest people survive? How many male passengers in this class survived?

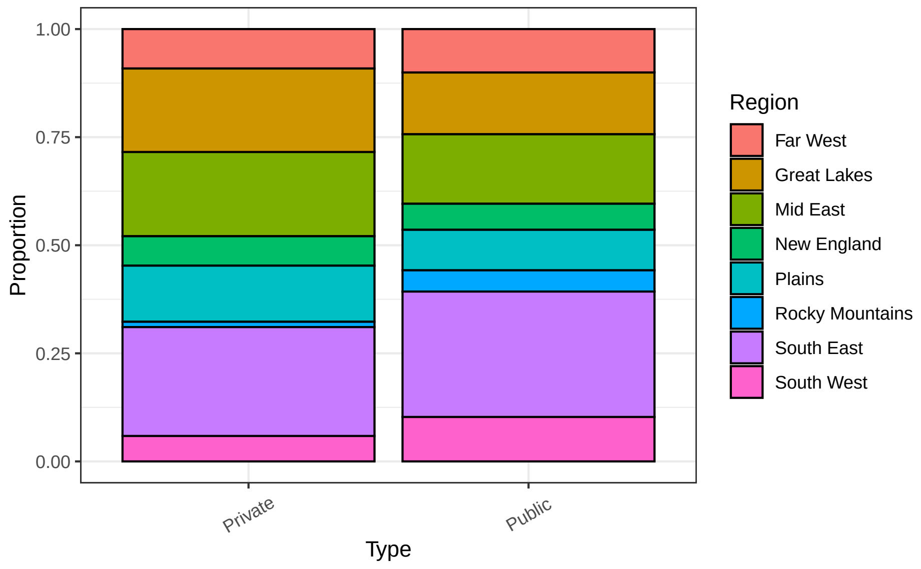
More Practice (required!)

Below is the code to make a smaller version of the colleges dataset by only including a few variables.

```r
college <- read.csv("https://collinn.github.io/data/college2019.csv") # read in dataset

college <- college[,c(1, 5, 6)] # keep only columns 1, 5, and 6
```





Question 4 Part A: Write the code to produce tables displaying information for the two plots of college data (Region and Type)

Part B: Using the appropriate graph and table, do any regions have public schools as a majority and if so what percent of schools in that region are public?

Part C: Using the appropriate graph and table, amongst public schools, which region has the largest percentage (also give me that percentage).