R Kaggle Guide (Titanic)

UWaterloo Data Science Club

August 18, 2017

This guide is based on R 3.3.2. We recommend downloading R here along with R Studio, a set of integrated tools that will make your life a lot easier. This guide assumes that you have some sort of programming experience.

This guide is written in something called R markdown, which allows us to describe our process while showing and executing code (kind of like a notebook). This notebook process is type of what Kaggle calls a **kernel**. When working in R Studio, pressing `ctrl + enter` will run the current line of code.

This guide will look at the Titanic dataset, we will see if we can predict what types of people would have survived on the Titanic.

So first we will import some useful libraries. R is old and there are confusing things about the language that came up over time, the tidyverse stack is a set of libraries that make these functions more consistent and powerful.

```
library("tidyverse")
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## filter(): dplyr, stats
## lag():    dplyr, stats
```

Note the conflict errors indicate that two different libraries have the same function name. We don't need to worry about this for now. Now we can import our data into a dataframe.

```
## Parsed with column specification:
## cols(
##   PassengerId = col_integer(),
##   Survived = col_integer(),
##   Pclass = col_integer(),
##   Name = col_character(),
##   Sex = col_character(),
##   Age = col_double(),
##   SibSp = col_integer(),
##   Parch = col_integer(),
##   Ticket = col_character(),
##   Fare = col_double(),
##   Cabin = col_character(),
##   Embarked = col_character()
## )
```

The code output tells us about how each column was imported such as what data type is stored. To understand more about what options you have, you can type `?read_csv` in the console, or `?<function_name>` for any

function. If you don't know what exactly the function name is, you can do `??<query>` which will return you all manual pages relevant to the query.

Normally we won't worry to much about datatypes, but notice how certain columns like `Survived` and `Pclass` were imported as integers? The problem is that we use integers to differentiate the data value but there isn't any ineherent order to the numbers. Instead we can convert integers, characters, etc.to categories, which is called a **factor** in R.

The `$` let's us select specific variables in a dataframe.

```
titanic_data$Survived <- as.factor(titanic_data$Survived)
titanic_data$Pclass <- as.factor(titanic_data$Pclass)
titanic_data$Sex <- as.factor(titanic_data$Sex)
titanic_data$Embarked <- as.factor(titanic_data$Embarked)
```

We can observe the first `n` entries of our dataframe by using the `head()` function, likewise we to observe the last `n` entires we can use `tail()`. If there are too many variables, the output will omit them to save space.

```
head(titanic_data, 5)
```

```
## # A tibble: 5 x 12
##   PassengerId Survived Pclass
##         <int>   <fctr> <fctr>
## 1           1        0      3
## 2           2        1      1
## 3           3        1      3
## 4           4        1      1
## 5           5        0      3
## # ... with 9 more variables: Name <chr>, Sex <fctr>, Age <dbl>,
## #   SibSp <int>, Parch <int>, Ticket <chr>, Fare <dbl>, Cabin <chr>,
## #   Embarked <fctr>
```

After a quick look, let's get a summary of our data.

```
summary(titanic_data)
```

```
##   PassengerId    Survived Pclass      Name              Sex
##  Min.   :  1.0   0:549    1:216   Length:891         female:314
##  1st Qu.:223.5   1:342    2:184   Class :character   male  :577
##  Median :446.0            3:491   Mode  :character
##  Mean   :446.0
##  3rd Qu.:668.5
##  Max.   :891.0
##
##       Age            SibSp           Parch           Ticket
##  Min.   : 0.42   Min.   :0.000   Min.   :0.0000   Length:891
##  1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000   Class :character
##  Median :28.00   Median :0.000   Median :0.0000   Mode  :character
##  Mean   :29.70   Mean   :0.523   Mean   :0.3816
##  3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000
##  Max.   :80.00   Max.   :8.000   Max.   :6.0000
##  NA's   :177
##       Fare           Cabin           Embarked
##  Min.   :  0.00   Length:891         C   :168
##  1st Qu.:  7.91   Class :character   Q   : 77
##  Median : 14.45   Mode  :character   S   :644
##  Mean   : 32.20                      NA's:  2
##  3rd Qu.: 31.00
```

```
##  Max.   :512.33
##
```

The `NA's` in some columns indicate the number of missing values. One could either remove the rows with missing values, or try to fill in the data based on surrounding data. Since our dataset is fairly small, the latter is preferred. This is called **imputation**.

Let's get a closer look at who these people with missing embarked locations are. We can use the `filter()` function to select rows that satisfy a certain criteria. Note that we do not have to use $ to indicate that `Embarked` is from `titanic_data` because it's inferred when we put what data we're looking at as the first parameter in `filter`.

NOTE: `NA == NA` will return `NA`. While this may be confusing think of it this way.

```
alice.age <- NA  # We don't know Alice's age
bob.age  <- NA  # We don't know Bob's age
alice.age == bob.age  # Are Alice and Bob the same age? We don't know!
```

```
## [1] NA
```

That's why we use `is.na` to test for missing values instead.

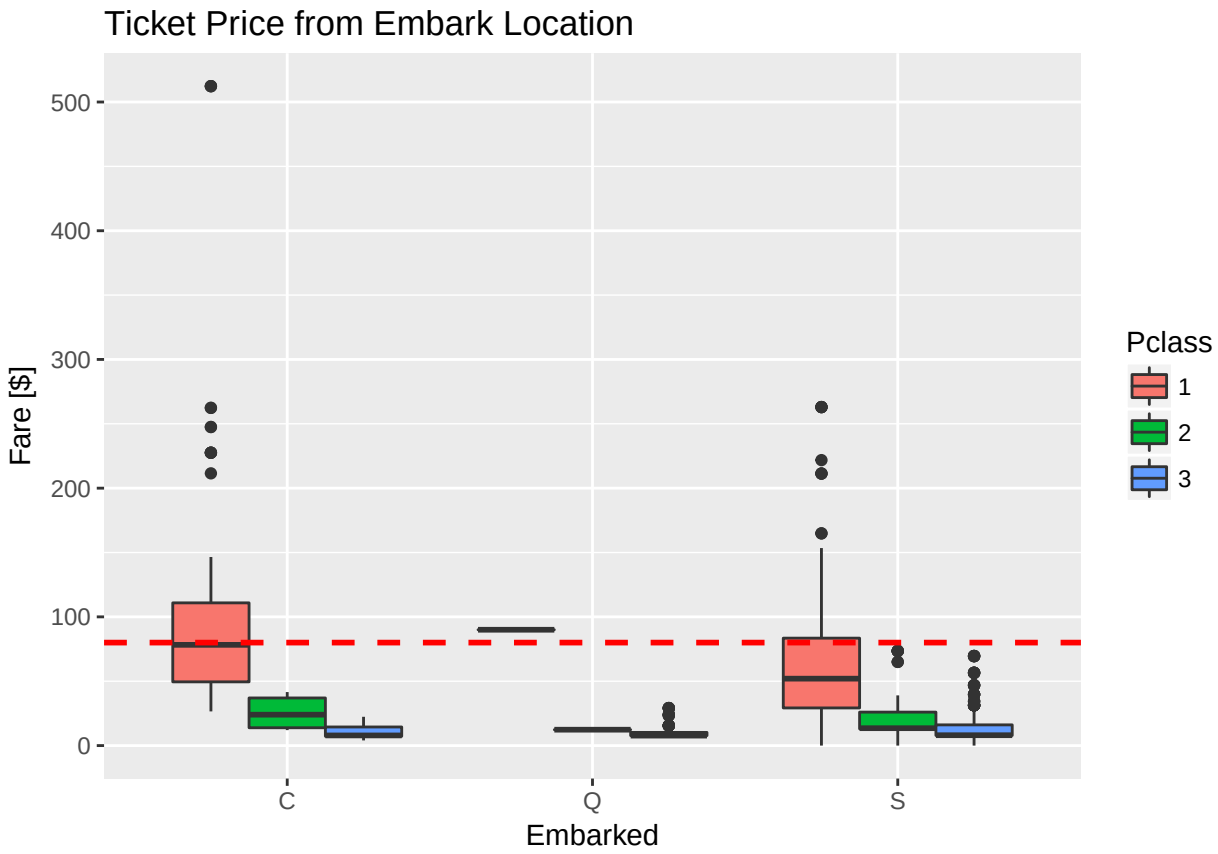
NOTE: The code below might start to look a little convoluted. We'll soon look at some syntactic sugar to make everything easier to read.

```
filter(titanic_data, is.na(Embarked))[c('Name', 'Fare', 'Ticket', 'Cabin')]
```

```
## # A tibble: 2 x 4
##                                         Name  Fare Ticket Cabin
##                                        <chr> <dbl>  <chr> <chr>
## 1                       Icard, Miss. Amelie    80 113572   B28
## 2 Stone, Mrs. George Nelson (Martha Evelyn)    80 113572   B28
```

It seems the passenger's had the same ticket, hence the identical fare. Let's visualize how much a passenger paid and their class based off the location and they embarked. We add $80 as the dashed red line to make a comparison.

```
ggplot(filter(titanic_data, !is.na(Embarked)),
       aes(x = Embarked, y = Fare, fill = Pclass)) +
  geom_boxplot() +
  scale_y_continuous() +
  labs(title = "Ticket Price from Embark Location",
       y = "Fare [$]") +
  geom_hline(aes(yintercept = 80),
             colour = "red", linetype = "dashed", lwd = 1)
```



The red line is aligned with the median of the fare paid in location C. Thus we will fill in the missing embarked locations with C.

```
titanic_data$Embarked[is.na(titanic_data$Embarked)] <- 'C'
```

INCOMPLETE SECTION

Another method of imputation is through prediction. It would be naive to use simple methods such as mean because we have other data that hint towards the age of a passenger. We can make a model to estimate the age from the other information we have.

```
model <- lm(Age ~ Survived + Pclass * Fare, titanic_data)
summary(model)
```

```
##
## Call:
## lm(formula = Age ~ Survived + Pclass * Fare, data = titanic_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.537  -8.920  -0.874   7.707  45.506
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   45.49802    1.50899  30.151  < 2e-16 ***
## Survived1     -6.68593    1.05170  -6.357 3.68e-10 ***
## Pclass2       -8.23046    2.32860  -3.535 0.000435 ***
```

```
## Pclass3      -14.50490    1.82011  -7.969 6.40e-15 ***
## Fare          -0.03273    0.01170  -2.798 0.005283 **
## Pclass2:Fare  -0.16205    0.07495  -2.162 0.030950 *
## Pclass3:Fare  -0.28864    0.06876  -4.198 3.04e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.8 on 707 degrees of freedom
##   (177 observations deleted due to missingness)
## Multiple R-squared:  0.2296, Adjusted R-squared:  0.2231
## F-statistic: 35.12 on 6 and 707 DF,  p-value: < 2.2e-16
```