R Tutorial #2 – Econ 103

Contents

This tutorial has two parts. In the first part, you'll compute summary statistics and graphs using R. In the second part, you'll use what you've learned to explore a real-world dataset: the passenger manifest from the Titanic.

Summary Stats and Graphics

Load and View the Data

In this section we'll work with car data that is preloaded in R. First, make sure you have the "datasets" package loaded and then create a data.table with the data.

```
library(datasets)
carData <- mtcars
```

It's always a good idea to take a look at your data after loading it. The functions **head** and **tail**, introduced in the previous tutorial, make this easy (this dataset is small enough that you can see the whole thing too):

```
head(carData)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
tail(carData)
```

```
##                 mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
## Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
## Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

Making Histograms

Making beautiful charts is a valuable skill. Unfortunately, the default R graphics are really ugly. Thankfully, there's a package for that! You'll need to install the `plotly` package to generate the prettier charts. Another huge plus is that it makes **interactive charts**!

Generating a chart is a simple matter of using the command `plot_ly(data, xData, yData, chartType, mode)` (there's tons of other options, but we'll only focus on the basics). A histogram is just a frequency chart of one variable, so there's no need to specify the `y` variable when you're making a histogram.

```
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
plot_ly(data = carData, x = mpg, type = "histogram")
```

The chart isn't too well labeled. Having a title and proper axis labels would be nice. Let's add the proper labels.

```
plot_ly(data = carData, x = mpg, type = "histogram") %>%
  layout(title = "MPG Histogram", xaxis = list(title = "MPG"),
         yaxis = list(title = "Frequency"))
```

By default, R produces histograms in terms of *frequencies*. That is, it counts the *numbers* of observations in each bin. To change this to *relative frequencies*, that is proportions, we add the argument `histnorm` and set it to `percent`. Note, I also changed the y-axis label.

```
plot_ly(data = carData, x = mpg, type = "histogram", histnorm = "percent") %>%
  layout(title = "MPG Histogram", xaxis = list(title = "MPG"),
         yaxis = list(title = "Percent"))
```

Scatter Plots

Making scatter plots is very similar to how we made histograms, with the appropriate change to the plot `type` as well as specifying the `y` data. Let's make a scatter plot of horsepower and MPG. What happens if you forget to specify `mode = "markers"`?

```
plot_ly(data = carData, x = mpg, y = hp, type = "scatter", mode = "markers") %>%
  layout(title = "MPG and Horsepower", xaxis = list(title = "MPG"),
         yaxis = list(title = "Horsepower"))
```

Let's say your were interested in how different engine sizes were related to this scatter. You could color the dots by the number of cylinders.

```
plot_ly(data = carData, x = mpg, y = hp, type = "scatter",
        mode = "markers", color = cyl) %>%
  layout(title = "MPG and Horsepower", xaxis = list(title = "MPG"),
         yaxis = list(title = "Horsepower"))
```

```
## Warning in doColorRamp(colorMatrix, x, alpha, ifelse(is.na(na.color), "", :
## '.Random.seed' is not an integer vector but of type 'NULL', so ignored
```

If you're familiar with cars, the color grouping should not surprise you.

Box Plots

A boxplot is an alternative way to visualize the distribution of a dataset. Once again, a few tweaks to the chart type should give us what we need.

```
plot_ly(data = carData, y = mpg, type = "box", boxpoints = FALSE) %>%
  layout(title = "MPG Box Plot")
```

Let's say you're interested in the MPG spread by engine size. How can you easily create the box plots for those different groups? Just tell R that you want to group the data by `cyl` and you're good to go!

```
plot_ly(data = carData, y = mpg, type = "box", boxpoints = FALSE,
        group = cyl) %>%
  layout(title = "MPG Box Plot")
```

Summary Statistics

The `summary` command takes a *whole dataframe* as its argument, unlike the other commands for summary statistics. It gives us the mean of each column along with the five-number summary. It also indicates if there are any missing observations (NA's) in the columns:

```
summary(carData)
```

```
##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

mean

Calculates the sample mean of a numeric vector

```r
mean(carData$mpg, na.rm = TRUE)
```

```
## [1] 20.09062
```

median

Calculates the sample median of a numeric vector

```r
median(carData$mpg, na.rm = TRUE)
```

```
## [1] 19.2
```

var

Calculates the sample variance of a numeric vector

```r
var(carData$mpg, na.rm = TRUE)
```

```
## [1] 36.3241
```

sd

Calculates the sample standard deviation of a numeric vector

```r
sd(carData$mpg, na.rm = TRUE)
```

```
## [1] 6.026948
```

This is identical to the (positive) square root of the sample variance:

```r
sqrt(var(carData$mpg, na.rm = TRUE))
```

```
## [1] 6.026948
```

IQR

Calculates the interquartile range of a numeric vector (the 75th percentile minus the 25th percentile)

```r
IQR(carData$mpg, na.rm = TRUE)
```

```
## [1] 7.375
```

`max, min`

These do exactly what they say

```r
max(carData$mpg, na.rm = TRUE)
```

```
## [1] 33.9
```

```r
min(carData$mpg, na.rm = TRUE)
```

```
## [1] 10.4
```

and can be used to calculate the range

```r
max(carData$mpg, na.rm = TRUE) - min(carData$mpg, na.rm = TRUE)
```

```
## [1] 23.5
```

To get both the maximum and minimum at once, you can use the function range;

```r
range(carData$mpg, na.rm = TRUE)
```

```
## [1] 10.4 33.9
```

Note that this does *not* compute the *summary statistic* called the range.

`quantile`

This function calculates sample quantiles, aka percentiles, of a numeric vector. If you simply pass it a numeric vector with no other arguments, it will give you the five-number summary:

```r
quantile(carData$mpg, na.rm = TRUE)
```

```
##     0%    25%    50%    75%   100% 
## 10.400 15.425 19.200 22.800 33.900
```

You can ask for specific quantile by using the argument probs

```r
quantile(carData$mpg, na.rm = TRUE, probs = 0.3)
```

```
##   30% 
## 15.98
```

Note that quantiles are specified as *probabilities* so the preceding command gives the 30th percentile. You can also ask for multiple quantiles at once:

```
quantile(carData$mpg, na.rm = TRUE, probs = c(0.1, 0.3, 0.7, 0.9))
```

```
##   10%   30%   70%   90% 
## 14.34 15.98 21.47 30.09
```

cor

This command calculates sample correlation coefficients. If you pass it two vectors of the same length it returns the correlation between them. Unlike the univariate summary statistics, in which we used `na.rm = TRUE` to ignore missing observations, for `cor` we use the argument `use = "complete.obs"`. The reason for this difference is that `cor` can handle a matrix or dataframe as its input. In this case there are actually many different ways of handling missing observations. For more details, see the help file. Setting `use = "complete.obs"` removes any rows with missing observations before proceeding.

```
cor(carData$mpg, carData$hp, use = "complete.obs")
```

```
## [1] -0.7761684
```

We see that there is a large negative correlation between `mpg` and `hp`

You can also put a whole data table into `cor` and it will return the whole matrix of all correlations between each variable (i.e. *all pairwise correlations*).

```
cor(carData, use = "complete.obs")
```

```
##             mpg        cyl       disp         hp        drat         wt
## mpg   1.0000000 -0.8521620 -0.8475514 -0.7761684  0.68117191 -0.8676594
## cyl  -0.8521620  1.0000000  0.9020329  0.8324475 -0.69993811  0.7824958
## disp -0.8475514  0.9020329  1.0000000  0.7909486 -0.71021393  0.8879799
## hp   -0.7761684  0.8324475  0.7909486  1.0000000 -0.44875912  0.6587479
## drat  0.6811719 -0.6999381 -0.7102139 -0.4487591  1.00000000 -0.7124406
## wt   -0.8676594  0.7824958  0.8879799  0.6587479 -0.71244065  1.0000000
## qsec  0.4186840 -0.5912421 -0.4336979 -0.7082234  0.09120476 -0.1747159
## vs    0.6640389 -0.8108118 -0.7104159 -0.7230967  0.44027846 -0.5549157
## am    0.5998324 -0.5226070 -0.5912270 -0.2432043  0.71271113 -0.6924953
## gear  0.4802848 -0.4926866 -0.5555692 -0.1257043  0.69961013 -0.5832870
## carb -0.5509251  0.5269883  0.3949769  0.7498125 -0.09078980  0.4276059
##             qsec         vs          am       gear        carb
## mpg   0.41868403  0.6640389  0.59983243  0.4802848 -0.55092507
## cyl  -0.59124207 -0.8108118 -0.52260705 -0.4926866  0.52698829
## disp -0.43369788 -0.7104159 -0.59122704 -0.5555692  0.39497686
## hp   -0.70822339 -0.7230967 -0.24320426 -0.1257043  0.74981247
## drat  0.09120476  0.4402785  0.71271113  0.6996101 -0.09078980
## wt   -0.17471588 -0.5549157 -0.69249526 -0.5832870  0.42760594
## qsec  1.00000000  0.7445354 -0.22986086 -0.2126822 -0.65624923
## vs    0.74453544  1.0000000  0.16834512  0.2060233 -0.56960714
## am   -0.22986086  0.1683451  1.00000000  0.7940588  0.05753435
## gear -0.21268223  0.2060233  0.79405876  1.0000000  0.27407284
## carb -0.65624923 -0.5696071  0.05753435  0.2740728  1.00000000
```

We can see that `mpg` is also negatively correlated with `cyl` (engine cylinders), `wt` (car weight), but it is positively correlated with `qsec` (the time to cover 0.25 miles). The preceding matrix has ones on its main diagonal since the correlation between any variable and itself is one. (A good exercise for extra practice would be to prove this assertion using the formula for correlation from class. It's not very difficult.)

`cov`

This command works just like `cor` but returns covariances rather than correlations:

```r
cov(carData$mpg, carData$hp, use = "complete.obs")
```

```
## [1] -320.7321
```

```r
cov(carData, use = "complete.obs")
```

```
##              mpg         cyl        disp          hp         drat
## mpg    36.324103  -9.1723790  -633.09721 -320.732056   2.19506351
## cyl    -9.172379   3.1895161   199.66028  101.931452  -0.66836694
## disp -633.097208 199.6602823 15360.79983 6721.158669 -47.06401915
## hp   -320.732056 101.9314516  6721.15867 4700.866935 -16.45110887
## drat    2.195064  -0.6683669   -47.06402  -16.451109   0.28588135
## wt     -5.116685   1.3673710   107.68420   44.192661  -0.37272073
## qsec    4.509149  -1.8868548   -96.05168  -86.770081   0.08714073
## vs      2.017137  -0.7298387   -44.37762  -24.987903   0.11864919
## am      1.803931  -0.4657258   -36.56401   -8.320565   0.19015121
## gear    2.135685  -0.6491935   -50.80262   -6.358871   0.27598790
## carb   -5.363105   1.5201613    79.06875   83.036290  -0.07840726
##               wt         qsec           vs           am        gear
## mpg   -5.1166847   4.50914919   2.01713710   1.80393145   2.1356855
## cyl    1.3673710  -1.88685484  -0.72983871  -0.46572581  -0.6491935
## disp 107.6842040 -96.05168145 -44.37762097 -36.56401210 -50.8026210
## hp    44.1926613 -86.77008065 -24.98790323  -8.32056452  -6.3588710
## drat  -0.3727207   0.08714073   0.11864919   0.19015121   0.2759879
## wt     0.9573790  -0.30548161  -0.27366129  -0.33810484  -0.4210806
## qsec  -0.3054816   3.19316613   0.67056452  -0.20495968  -0.2804032
## vs    -0.2736613   0.67056452   0.25403226   0.04233871   0.0766129
## am    -0.3381048  -0.20495968   0.04233871   0.24899194   0.2923387
## gear  -0.4210806  -0.28040323   0.07661290   0.29233871   0.5443548
## carb   0.6757903  -1.89411290  -0.46370968   0.04637097   0.3266129
##             carb
## mpg  -5.36310484
## cyl   1.52016129
## disp 79.06875000
## hp   83.03629032
## drat -0.07840726
## wt    0.67579032
## qsec -1.89411290
## vs   -0.46370968
## am    0.04637097
## gear  0.32661290
## carb  2.60887097
```

Exploring the Titanic

Now it's your turn: using what you've learned above, you'll analyze some data about everyone that was aboard the Titanic! You already know enough R to answer some interesting questions, so let's dive in!

A Helpful Hint: Taking the mean of a vector of ones and zeros is the *same thing* as calculating the *proportion of ones.*

1. Read the documentation file for the titanic dataset so that you know what each column in the data represents: https://github.com/emallickhossain/Econ103Public/raw/master/Rtutorials/titanic.txt
2. Download the data, store it in a data table called titanic, and display the first few rows.
3. We'll only be using the following columns of titanic for this example: pclass, survived, sex, age, and fare. Select only these columns and store then in the data table titanic, overwriting the original dataset. Display the first few rows to make sure your command worked.
4. Using the command summary to get an overview of the dataset and answer the following questions:

- Are there any missing values in this dataset?
- What was the average age of passengers on the Titanic?
- What was the average ticket price for passengers on the ship?
- Do the ticket prices show evidence of skewness? If so, are they positively or negatively skewed?
- Were there more men or women aboard the Titanic?
- What proportion of the passengers survived the disaster?
- To which social class did most of the passengers belong?

5. Being sure to allow for missing values if necessary, calculate (or plot) and interpret the following:

- The standard deviation of fares
- The 90th Percentile of fares
- Histogram of fares

6. Create a boxplot of `fare` by `pclass` and interpret your results.
7. Create a data table called survivors, containing data for only those passengers who survived the disaster and answer the following questions:

- What was the average age of the survivors?
- Among the survivors, were there more men or women?
- Do you notice anything different about social class for survivors compared to all passengers?

8. Determine whether men or women paid more, on average, for passage on the titanic. Hint: use something that looks like the following `dataset[, operation, by = group]`
9. How did the fraction of survivors vary by `sex pclass`? Can you think of a possible explanation for the pattern you see in the data? Hint: use something that looks like the following `dataset[, operation, by = list(group1, group2)]`