Lab 1 Solutions

IMC 490: Machine Learning for IMC

March 30, 2017

Here are solutions to the 10 exercises at the end of Lab 1. Remember there are several ways to do each problem, so in some cases I'll provide more than 1 answer to demonstrate this. You'll be able to do all of these problems using the commands presented in the lab.

Remember to use `?command` *or* `help(command)` *to read up on how to use any command you're not familiar with. Googling for StackOverflow answers is also a key way to find answers.*

Note: In lab we spoke about R scripts. Remember that the console doesn't save your output - you always want to save your commands in an R script (File -> New File -> R Script). The script will run each command sequentially, so you typically start by loading data, then clean the data, and then train models and run statistics. I'll provide an example of what these exercises in script form will look like on Canvas.

Preliminary work

```
# begin by reading the Auto data into a dataframe
auto = read.csv("http://www-bcf.usc.edu/~gareth/ISL/Auto.csv")

# let's explore the dataframe a bit
names(auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"
## [5] "weight"       "acceleration" "year"         "origin"
## [9] "name"
```

```
str(auto)
```

```
## 'data.frame':    397 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : Factor w/ 94 levels "?","100","102",..: 17 35 29 29 24 42 47 46 48 40 ...
##  $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
```

1. What is the average car weight? Heaviest? Lightest?

```
mean(auto$weight)  # average
```

```
## [1] 2970.262
```

```
max(auto$weight)  # heaviest
```

```
## [1] 5140
```

```
min(auto$weight)  # lightest
```

```
## [1] 1613
```

2. How many types of engines are there (using # of cylinders) and which is the most common type?

There are 5 types of engines and 4 cylinder is the most common.

```
table(auto$cylinders)
```

```
##
##   3   4   5   6   8
##   4 203   3  84 103
```

3. How many unique car models are there?

304 models.

```
# There are too many models to use table() and count the models
# manually this time. Instead, find the length of the vector with all unique car models.
length(unique(auto$name))
```

```
## [1] 304
```

4. How many cars weigh over 3000 pounds?

167 cars. Note the three different ways to obtain this answer. All are about equal in efficiency.

```
# Method 1: Count the number of TRUEs in the logical vector
table(auto$weight > 3000)
```

```
##
## FALSE  TRUE
##   230   167
```

```
# Method 2: Index the dataframe using the logical vector and count the rows
nrow(auto[auto$weight > 3000, ])
```

```
## [1] 167
```

```
# Method 3: Index the weight column (a vector) and find the length
length(auto$weight[auto$weight > 3000])
```

```
## [1] 167
```

5. Create a new column for kpg - kilometers per gallon, for our metric friends. (1 mile = 1.6 km)

```
auto$kpg = auto$mpg * 1.6
```

6. What is the average mpg of cars made after '75? Before '75? Does it look like newer cars have more efficient engines?

26.9 after '75, 19.4 before '75. Looks like newer cars do have more efficient engines.

Note the two ways to do this problem. Method 1 is more efficient if you are just trying to obtain the number, but Method 2 is better if you plan on continuing the analysis beyond just mpg, since you can just use the new dataframes you made and don't have to reindex every time.

```
# Method 1: directly indexing the mpg column using the logical vector
mean(auto$mpg[auto$year > 75])
```

```
## [1] 26.97116
```

```
mean(auto$mpg[auto$year <= 75])
```

```
## [1] 19.43407
```

```
# Method 2: splitting the dataframe first
auto_before_75 = auto[auto$year > 75, ]
auto_after_75 = auto[auto$year <= 75, ]
mean(auto_before_75$mpg)
```

```
## [1] 26.97116
```

```
mean(auto_after_75$mpg)
```

```
## [1] 19.43407
```

7. What is the model of the car with the fastest acceleration?

It's a Peugeot 504. Surprisingly only a mid range car (~ $30K used). Looks like there aren't any Ferraris in this dataset.

```
auto$name[auto$acceleration == max(auto$acceleration)]
```

```
## [1] peugeot 504
## 304 Levels: amc ambassador brougham ... vw rabbit custom
```

8. Extract a dataframe of only the names and years of cars that get fewer than 12 mpg.

```
auto[auto$mpg < 12, c("name", "year")]
```

```
##                 name year
## 26         ford f250   70
## 27         chevy c20   70
## 28        dodge d200   70
## 29          hi 1200d   70
## 68   mercury marquis   72
## 104 chevrolet impala   73
```

```
## 125 oldsmobile omega   73
```

9. I need a car recommendation. I'm looking for a vehicle with a 4 cylinder engine, made after 1980, with an acceleration rating of over 17.5. Grab me the list of cars that meet these criteria and recommend the one that gets the most miles per gallon.

VW Pickup.

```
car_recs = auto[auto$cylinders == 4 & auto$year > 80 & auto$acceleration > 17.5, ]

car_recs[car_recs$mpg == max(car_recs$mpg), ]
```

```
##     mpg cylinders displacement horsepower weight acceleration year origin
## 394  44         4           97         52   2130         24.6   82      2
##          name  kpg
## 394 vw pickup 70.4
```

10. What is the car with the highest horsepower? (You'll run into 2 gotchas with this problem. You need to convert the horsepower column from a factor to an integer)

Pontiac Grand Prix.

This problem demonstrates a MAJOR gotcha when it comes to R dataframes. Note that the type of the horsepower column is a **Factor**, which we have not yet covered in lab. A "factor" in R is simply a categorical variable.

```
str(auto$horsepower)
```

```
##  Factor w/ 94 levels "?","100","102",..: 17 35 29 29 24 42 47 46 48 40 ...
```

If we try to directly convert the column (just the first 10 observations so it's not too long) to an integer, this is what we get.

```
as.integer(auto$horsepower[1:10])
```

```
##  [1] 17 35 29 29 24 42 47 46 48 40
```

```
auto$horsepower[1:10]
```

```
##  [1] 130 165 150 150 140 198 220 215 225 190
## 94 Levels: ? 100 102 103 105 107 108 110 112 113 115 116 120 122 ... 98
```

Oh crap! What happened? The numbers are completely off!

It turns out that R encodes factors as integers representing which LEVEL of an the factor the observation is. Take the first entry. The actual horsepower is 130, but the integer conversion says 17. If we look at the 17th level in the factor...

```
levels(auto$horsepower)[17]
```

```
## [1] "130"
```

We'll see that it is 130, the correct value. To convert the column correctly, first convert the column to a character (getting rid of the Factor encodings), then to an integer.

```
as.integer(as.character(auto$horsepower[1:10]))
```

```
##  [1] 130 165 150 150 140 198 220 215 225 190
```

Awesome. Now to finish the problem. Below we see that the reason the column was read in as a Factor instead an Integer in the first place was because there are "?" entries representing missing values. **Note the warning message signifying that there the "?" cannot be converted to numbers, and are therefore converted to NA. Also note that we need to remove the NAs using `na.omit()` first, otherwise we can't calculate a max horsepower!**

```
auto$horsepower_corrected = as.integer(as.character(auto$horsepower))
```

```
## Warning: NAs introduced by coercion
```

```
auto = na.omit(auto)  # remove the NAs
auto[auto$horsepower_corrected == max(auto$horsepower_corrected), ]
```

```
##     mpg cylinders displacement horsepower weight acceleration year origin
## 117  16         8          400        230   4278          9.5   73      1
##                   name  kpg horsepower_corrected
## 117 pontiac grand prix 25.6                 230
```