Lab 1

IMC 490: Machine Learning for IMC

In this lab, we'll be going over the following topics:

- navigating your filesystem from R
- basic data types
- vectors
- dataframes
- reading data into R

Remember to use `?command` or `help(command)` in the R console to access documentation at any time if you have questions.

Navigation

```
setwd()
getwd()
list.files()
```

Let's start by getting around your filesystem. You'll use `getwd()` (**get** **w**orking **d**irectory) to figure out where you are, `setwd()` (**set** **w**orking **d**irectory) to move around, and `list.files()` to look at the stuff that's in the folder.

```r
setwd("~/Documents/Machine-Learning-IMC490/Lab1/")
```

```r
getwd()
```

```
## [1] "/home/eric/Documents/Machine-Learning-IMC490/Lab1"
```

```r
list.files()
```

```
## [1] "Auto.csv" "HW1"      "Lab1.pdf" "Lab1.Rmd"
```

Common gotcha: Be sure to enter the filepath in `setwd()` as a string (in quotes " "). A common mistake is to forget the quotes. If you do, R will think that */your/filepath/* is a variable holding some value… And complain when it finds that it isn't.

Data Types and Vectors

`c()`

From the R documentation: “R has six basic (‘atomic’) vector types: logical, integer, real, complex, string (or character) and raw.”

We often use integer, real, string, and logical types. In normal use we almost never see complex and raw.

Let’s make some vectors. `c()` (**c**ombine) is the generic function for creating a vector.

```r
# 2 ways of making a vector with numbers 1 through 5
c(1, 2, 3, 4, 5)
```

```
## [1] 1 2 3 4 5
```

```r
c(1:5)
```

```
## [1] 1 2 3 4 5
```

```r
# create a character (string) vector and check its type
a_char_vec = c("a", "b", "c")
typeof(a_char_vec)
```

```
## [1] "character"
```

```r
# create a boolean vector
some_numbers = c(1:5)
a_bool_vec = some_numbers >= 3
a_bool_vec
```

```
## [1] FALSE FALSE  TRUE  TRUE  TRUE
```

The operation *>= 3* was vectorized and applied to each element of some_numbers. Vectorized operations are at the core of R. A vectorized operation is an operation that is applied to each element of a vector. This includes arithmetic and comparison operations.

```r
# create a vector 1:5 and add 3 to each element
x = c(1:5)
x + 3
```

```
## [1] 4 5 6 7 8
```

```r
# add two vectors to each other
x = c(1:5)
y = c(10, 20, 30, 40, 50)
x + y
```

```
## [1] 11 22 33 44 55
```

Common gotcha: If you perform an operation with two vectors of different length, the shorter vector will be extended to complete the operation.

```r
x = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
y = c(0, 100)
x + y
```

```
##  [1]   1 102   3 104   5 106   7 108   9 110
```

Some important vector functions:

Casting vectors from one type to another:
`as.integer()`
`as.numeric()`
`as.character()`

Checking characteristics of a vector:
`typeof()`
`length()`
`unique()`

Calculating statistics of a numeric vector:
`mean()`
`sd()`
`min()`
`max()`

Getting frequency counts of elements in a vector:
`table()`

Dataframes

`data.frame()`

A dataframe is simply a collection of vectors, where each vector is a column in the dataframe. Here we're making a dataframe with column x which is an index, and y, which are random samples from a standard normal distribution.

```r
dat = data.frame(x = c(1:5), y = rnorm(5))
dat
```

```
##   x          y
## 1 1  0.7229946
## 2 2 -0.4475875
## 3 3  1.1220001
## 4 4  0.1368707
## 5 5  0.5419758
```

To extract vectors from a dataframe, use the dollar sign operator.

```r
dat$y
```

```
## [1]  0.7229946 -0.4475875  1.1220001  0.1368707  0.5419758
```

To create new columns, use the dollar sign operator with a new variable name.

```r
dat$z = c(11:15)
dat
```

```
##   x          y  z
## 1 1  0.7229946 11
## 2 2 -0.4475875 12
## 3 3  1.1220001 13
## 4 4  0.1368707 14
## 5 5  0.5419758 15
```

Some important dataframe functions:
`names()`
`str()`
`summary()`
`nrow()`
`ncol()`
`na.omit()`

To index dataframes, use square brackets after the dataframe name. To index a dataframe using any of the following methods, you need to provide 2 dimensions - for rows and columns. You can pass in integer, character, or logical vectors - or leave the dimension blank to get the all rows or all columns.

Indexing using integer vectors:

```
dat[1:3, 1:2]  # grab the first 3 rows from the first 2 columns
```

```
##   x          y
## 1 1  0.7229946
## 2 2 -0.4475875
## 3 3  1.1220001
```

```
dat[1:3, ]  # grab the first 3 rows from all columns
```

```
##   x          y  z
## 1 1  0.7229946 11
## 2 2 -0.4475875 12
## 3 3  1.1220001 13
```

```
dat[ , 3]  # grab the 3rd column with all the rows
```

```
## [1] 11 12 13 14 15
```

Indexing by name using a character vector:

```
dat[1:3, c('x', 'y')]
```

```
##   x          y
## 1 1  0.7229946
## 2 2 -0.4475875
## 3 3  1.1220001
```

Indexing using a logical vector:

```
dat[dat$x >= 3, ]
```

```
##   x         y  z
## 3 3 1.1220001 13
## 4 4 0.1368707 14
## 5 5 0.5419758 15
```

```
dat[dat$x >= 3 & dat$z <= 14, ]
```

```
##   x         y  z
## 3 3 1.1220001 13
## 4 4 0.1368707 14
```

Reading and exploring data

```
read.csv()
head()
str()
summary()
```

Link to sample dataset: http://www-bcf.usc.edu/~gareth/ISL/Auto.csv

To read data into R, use the `read.csv()` function with a filepath or a URL.

```r
auto = read.csv("Auto.csv")
auto = read.csv("http://www-bcf.usc.edu/~gareth/ISL/Auto.csv")
head(auto)
```

```
##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

Exercises

Try exploring the automobile dataset with `head()`, `str()` (**str**ucture), and `summary()` before you get started. Open up a new R script and save your code.

1. What is the average car weight? Heaviest? Lightest?
2. How many types of engines are there (using # of cylinders) and which is the most common type?
3. How many unique car models are there?
4. How many cars weigh over 3000 pounds?
5. Create a new column `kpg` - kilometers per gallon, for our metric friends. (1 mile = 1.6 km)
6. What is the average mpg of cars made after '75? Before '75? Does it look like newer cars have more efficient engines?
7. What is the model of the car with the fastest acceleration?
8. Extract a dataframe of only the names and years of cars that get fewer than 12 mpg.
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.
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)