Lab 1

IMC 490: Machine Learning for IMC

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

- operations and data structures
- dataframes
- accessing data
- indexing data
- data types
- gotchas
- regression

Website for Introduction to Statistical Learning: http://www-bcf.usc.edu/~gareth/ISL/

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()
```

The above three commands will allow you to navigate 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] "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 1:10 and add 5 to each element
x = c(1:10)
x + 5
```

```
##  [1]  6  7  8  9 10 11 12 13 14 15
```

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 more 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()`

Dataframes

`data.frame()`

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

```
dat = data.frame(x = c(1:10), y = rnorm(10))
dat
```

```
##     x           y
## 1   1 -0.54881116
## 2   2  0.73988377
## 3   3 -0.11700799
## 4   4  0.08545879
## 5   5 -1.44733763
## 6   6  0.33142182
## 7   7 -0.54676942
## 8   8  1.09998854
## 9   9  0.75496501
## 10 10 -0.31216168
```

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

```
dat$y
```

```
##  [1] -0.54881116  0.73988377 -0.11700799  0.08545879 -1.44733763
##  [6]  0.33142182 -0.54676942  1.09998854  0.75496501 -0.31216168
```

Reading in data

`read.csv()` `data()`

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