Lab 1

This is loosely based on notes by Dr. Leemis: here. If you're not familiar with `r` I highly recommend you go through this.

Notebook basics

Code notebooks allow interweaving of

1. rich-text input (html, markdown, etc.)
2. code input
3. code output

```
print('hello darkness my old friend')
```

```
## [1] "hello darkness my old friend"
```

We can write notebooks using a couple different IDEs, popular being

1. jupyter which is a bit more advanced and requires installing anaconda
2. rstudio

either can be used to write code in R

R as a calculator

We can use R as a calculator

```
1+1
```

```
## [1] 2
```

```
1+2*5
```

```
## [1] 11
```

there are also some built-in constants in R

```
pi
```

```
## [1] 3.141593
```

```
exp(1)
```

```
## [1] 2.718282
```

there are also special infinite values

```
1/0
```

```
## [1] Inf
```

```
-1/0
```

```
## [1] -Inf
```

```
0/0
```

```
## [1] NaN
```

to denote nothing we have

```
NULL
```

```
## NULL
```

to denote a missing value

```
NA
```

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

Variable assignment and simple objects

One can assign variables with either an `=` or `<-`

```
x = 1
x
```

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

```
x <- 1
x
```

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

Vectors

vectors are made with the `c` command

```
x = c(5, 3, 7)
x
```

```
## [1] 5 3 7
```

we get the elements with `[]`

```
x[1]
```

```
## [1] 5
```

negative numbers selects all but those elements

```
x[c(-1,-2)]
```

```
## [1] 7
```

we can make a vector out of any type

```
truths <- c(TRUE, FALSE, TRUE)
truths
```

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

if we subset by boolean then it select those elements with `TRUE`

```
x
```

```
## [1] 5 3 7
```

```
truths
```

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

```
x[truths]
```

```
## [1] 5 7
```

we can make consecutive integers using:

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

or the `seq` function

```
seq(1, 20, by=3)
```

```
## [1]  1  4  7 10 13 16 19
```

matrices

we can make a matrix using the `matrix` function

```
X <- matrix(1:25,nrow=5,byrow=TRUE)
X
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
## [4,]   16   17   18   19   20
## [5,]   21   22   23   24   25
```

or the `array` function

```
Y <- array(25:49, c(5,5))
Y
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   25   30   35   40   45
## [2,]   26   31   36   41   46
## [3,]   27   32   37   42   47
## [4,]   28   33   38   43   48
## [5,]   29   34   39   44   49
```

matrix multiplication is done with the operator `%*%`

```
X %*% Y
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  415  490  565  640  715
## [2,] 1090 1290 1490 1690 1890
## [3,] 1765 2090 2415 2740 3065
## [4,] 2440 2890 3340 3790 4240
## [5,] 3115 3690 4265 4840 5415
```

I can extract or assign individual elements with `[,]`

```
X[1,2]
```

```
## [1] 2
```

```
X[1,2] <- 3
X
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
## [4,]   16   17   18   19   20
## [5,]   21   22   23   24   25
```

or entire rows/columns as follows

```
X[,1]
```

```
## [1]  1  6 11 16 21
```

```
X[1,]
```

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

Flow Control

if statements are as follows

```
A = 5
if(A==0){
    print("A=0")
} else if(A == 1){
    print("A=1")
} else if(A==2) {
    print("A=2")
} else {
    print("A is not 0, 1 or 2")
}
```

```
## [1] "A is not 0, 1 or 2"
```

we can also make `for` loops

```
for(i in 1:10){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

and `while` loops

```
i=1
while(i <= 10){
  print(i)
  i = i+1
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

major types

numeric

```
c(1,2,3)
```

```
## [1] 1 2 3
```

```
class(c(1,2,3))
```

```
## [1] "numeric"
```

strings

```
c('hello','darkness','my','old','friend')
```

```
## [1] "hello"    "darkness" "my"       "old"      "friend"
```

```
class(c('hello','darkness','my','old','friend'))
```

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

boolean

```
c(TRUE,FALSE)
```

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

```
class(c(TRUE,FALSE))
```

```
## [1] "logical"
```

factors for discrete variables

```r
fctr = as.factor(c("hello", "darkness"))
```

```r
fctr
```

```
## [1] hello    darkness
## Levels: darkness hello
```

```r
class(fctr)
```

```
## [1] "factor"
```

```r
as.numeric(fctr)
```

```
## [1] 2 1
```

```r
fctr*2
```

```
## Warning in Ops.factor(fctr, 2): '*' not meaningful for factors
```

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

data-frames and matrices:

matrcies all have to be the same type:

```r
X = array(0, c(5,5))
X
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    0    0    0    0    0
## [3,]    0    0    0    0    0
## [4,]    0    0    0    0    0
## [5,]    0    0    0    0    0
```

```r
X = array(LETTERS[1:25], c(5, 5))
X
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] "A"  "F"  "K"  "P"  "U"
## [2,] "B"  "G"  "L"  "Q"  "V"
## [3,] "C"  "H"  "M"  "R"  "W"
## [4,] "D"  "I"  "N"  "S"  "X"
## [5,] "E"  "J"  "O"  "T"  "Y"
```

so if I have a character `X` and I assign the number `5` to the first column, what happens?

```
X[,1] = rep(5,5)
X
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] "5"  "F"  "K"  "P"  "U"
## [2,] "5"  "G"  "L"  "Q"  "V"
## [3,] "5"  "H"  "M"  "R"  "W"
## [4,] "5"  "I"  "N"  "S"  "X"
## [5,] "5"  "J"  "O"  "T"  "Y"
```

```
X[,1]
```

```
## [1] "5" "5" "5" "5" "5"
```

this is not the number `5`, its the character `'5'`

The proper way to have mixed types is to have a `data.frame`

```
df = data.frame(x=rep(5,5),y=LETTERS[1:5])
df
```

```
##   x y
## 1 5 A
## 2 5 B
## 3 5 C
## 4 5 D
## 5 5 E
```

I can access the elements like matrices

```
df[1,1]
```

```
## [1] 5
```

```
df[1,2]
```

```
## [1] "A"
```

or with names

```
df$x
```

```
## [1] 5 5 5 5 5
```

```
df$y
```

```
## [1] "A" "B" "C" "D" "E"
```

```
df[["x"]]
```

```
## [1] 5 5 5 5 5
```

```
df[["y"]]
```

```
## [1] "A" "B" "C" "D" "E"
```