Lab 1

PSTAT 174/274

Basic R Operations

Simple operations and lists

1. Define a variable and look at it

```
x <- 12
x
```

```
## [1] 12
```

2. Mathematical operations on a variable

```
x + 2; x - 2; x/2; x*2; x^2
```

```
## [1] 14
```

```
## [1] 10
```

```
## [1] 6
```

```
## [1] 24
```

```
## [1] 144
```

3. Saving the result of mathematical operations in another variable

```
y <- x + 2
y
```

```
## [1] 14
```

Showing the result of an assignment as you do it (sometimes handy for debugging)

```
(y <- x + 2)
```

```
## [1] 14
```

4. Define a list with the `c()` function

```
x <- c(1, 2, 3, 4)
x
```

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

5. Mathematical operations on a list

   (a) Some examples

```
x + 2; x^2
```

```
## [1] 3 4 5 6
```

```
## [1]  1  4  9 16
```

R works on lists, not vectors; you must use special forms of mathematical operations to perform matrix operations correctly. Example, in which R not handling matrix operations correctly, is as follows.

```
y <- c(1:8)
y
```

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

```
x*y
```

```
## [1]  1  4  9 16  5 12 21 32
```

If we mistake lists for vectors (or matrices) then it looks like we just multiplied a 1×4 matrix with a 1×8 matrix. Take another look at x, y, and the result and see if you can figure out what we really did.

```
y <- c(1, 2, 3, 4, 5, 6, 7, 8)
y
```

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

```
x*y
```

```
## [1]  1  4  9 16  5 12 21 32
```

   (b) Matrix multiplication

```
x %*% t(x)
```

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

```
## [2,]    2    4    6    8
## [3,]    3    6    9   12
## [4,]    4    8   12   16
```

```
t(x) %*% x
```

```
##      [,1]
## [1,]   30
```

Note: the function `t(x)` takes the transpose of x.

(c) Matrix function

```
z1 <- matrix(y, 2)
z1
```

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

```
z2 <- matrix(y, 2, byrow=TRUE)
z2
```

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

6. Structure manipulation of lists

- Select a specific element

```
x[2]
```

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

- Select a range of elements

```
x[2:4]
```

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

- Drop a specific element

```
x[-2]
```

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

- Drop a range of elements

```
x[-c(2:3)]
```

```
## [1] 1 4
```

- Use list to select and drop elements

```
y <- c(1, 3)
x[y]; x[-y]
```

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

```
## [1] 2 4
```

- Combine two lists

```
x <- c(x,y)
x
```

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

- Reverse a list

```
rev(x)
```

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

7. Logical operations

- Test equality

```
1 == 5
```

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

```
1 == 1
```

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

```
1 != 5
```

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

```
1 != 1
```

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

8. Ranges

```r
1:3
```

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

```r
seq(from=0, to=1, by=0.12)
```

```
## [1] 0.00 0.12 0.24 0.36 0.48 0.60 0.72 0.84 0.96
```

```r
seq(from=0, to=1, length=7)
```

```
## [1] 0.0000000 0.1666667 0.3333333 0.5000000 0.6666667 0.8333333 1.0000000
```

Useful Functions

1. Mean, variance, standard deviation

```r
x <- c(1, 2, 3, 4)
mean(x)
```

```
## [1] 2.5
```

```r
var(x)  # NOTE: gives unbiased estimate, not MLE
```

```
## [1] 1.666667
```

```r
sd(x)
```

```
## [1] 1.290994
```

2. Summing a list

```r
sum(x)
```

```
## [1] 10
```

```r
prod(x)
```

```
## [1] 24
```

3. Length of a list

```r
length(x)
```

```
## [1] 4
```

4. Densities for various distributions

```r
dnorm(0, mean=0, sd=1)  # normal distribution
```

```
## [1] 0.3989423
```

```r
dexp(1, rate=1)  # exponential distribution
```

```
## [1] 0.3678794
```

```r
dbinom(5, size=10, prob=0.5)  # binomial distribution
```

```
## [1] 0.2460938
```

```r
dpois(10, lambda=10)  # Poisson distribution
```

```
## [1] 0.12511
```

Here the density function is being evaluated at the given x with the specified parameters. There's several others; look in the help file under "`distributions`" and scroll down a bit.

```r
help.search("distributions")
```

5. CDFs, inverse CDFs, and random generation Use normal distribution for illustration.

```r
pnorm(0, mean=0, sd=1)  # P(X <= 0)
```

```
## [1] 0.5
```

```r
qnorm(0.95, mean=0, sd=1)  # 95% upper tail for a normal RV
```

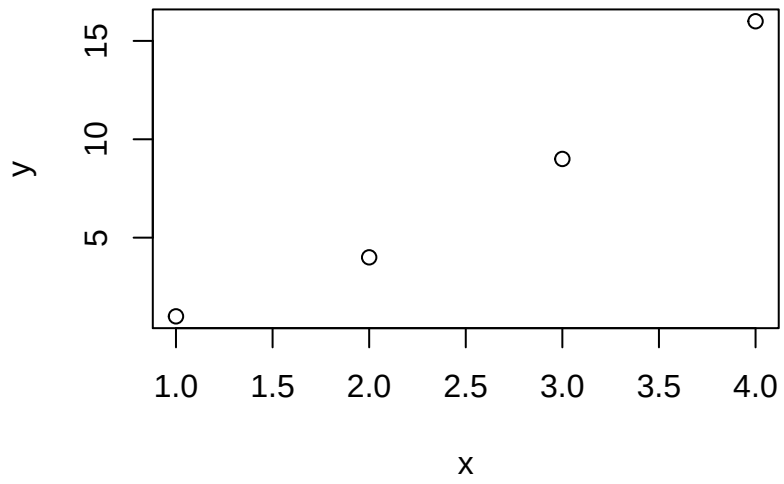
```
## [1] 1.644854
```

```r
rnorm(5, mean=0, sd=1)  # Generates 5 realizations of the standard normal RV
```

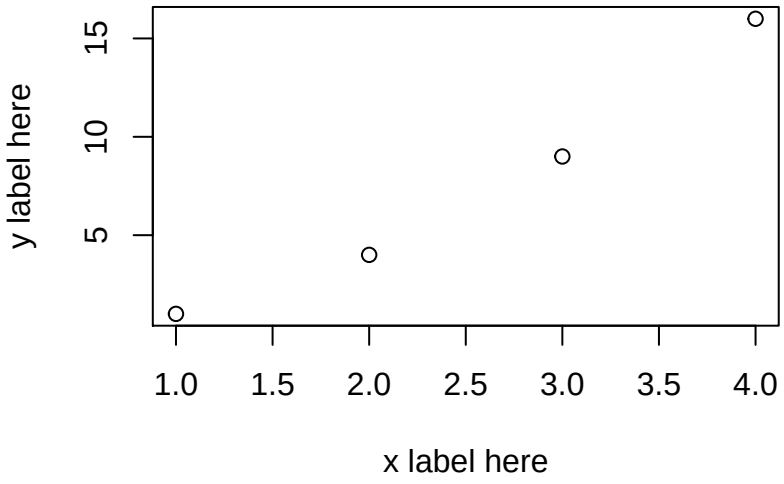
```
## [1]  1.9672784 -0.5183019  1.2472782 -0.7214694  1.4792329
```

6. Basic plotting

```r
x <- c(1, 2, 3, 4)
y <- x^2
plot(x, y)
```




```
plot(x,y, xlab="x label here", ylab="y label here")
```



IMPORTANT: Make sure to always use labels! Otherwise your plots are not helpful for other people (i.e. graders) looking at your plots!

Programming and Flow Control

1. `for` loops

```
for (x in 1:5) {
  print(x + 1)
}
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
```

2. `while` loops

```
x <- 1
while (x < 10) {
  print(x)
  x <- x + x
}
```

```
## [1] 1
## [1] 2
## [1] 4
## [1] 8
```

3. Defining functions

```
Fact <- function (n) {
  if (n == 1) {
return(1)
  } else {
return(n*Fact(n - 1))
  }
}
Fact(5)
```

```
## [1] 120
```