Linear Algebra in R with Applications to Linear Regression

1. Vectors
2. Matrices
3. Matrix operations: multiplication, inverse and transpose
4. Linear regression

1. Vectors

We can create vectors in several ways using the following functions:

- `c(value,value, ...  ,value)` concatenates its arguments to construct a vector
- `seq(from,to,by=1,length)` creates a vector with sequential values
- `from:to` (colon operator) is a shorthand method for `seq(from,to,by=1)`
- `rep(x,times)` takes a value/vector and repeats it according to the parameter *times*

For example:

```
c(1,0,-1,2)
```

```
## [1]  1  0 -1  2
```

```
seq(0,10,by=2)
```

```
## [1]  0  2  4  6  8 10
```

```
0:10
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

```
rep(0,times=5)
```

```
## [1] 0 0 0 0 0
```

We can also create vectors with random entries using the functions:

- `runif(n,min=0,max=1)` generates a vector of length n sampled uniformly from the interval $[min,max]$
- `rnorm(n,mean=0,sd=1)` generates a vectors of length n sampled from the normal distribution
- `sample(x,n,replace=FALSE)` generates a vector of length n sampled from the vector x (with or without replacement)

For example:

```r
runif(3,0,5)
```

```
## [1] 0.5351424 1.5575462 1.6208274
```

```r
rnorm(4)
```

```
## [1] -0.2806734 -0.8090762  0.7818351 -0.5097186
```

```r
sample(0:6,20,replace=TRUE)
```

```
##  [1] 0 1 6 1 6 4 4 1 3 3 1 4 6 6 6 0 0 4 0 5
```

Note that arithmetic operations are performed elementwise on vectors:

For example:

```r
1:10 / rep(2,10)
```

```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

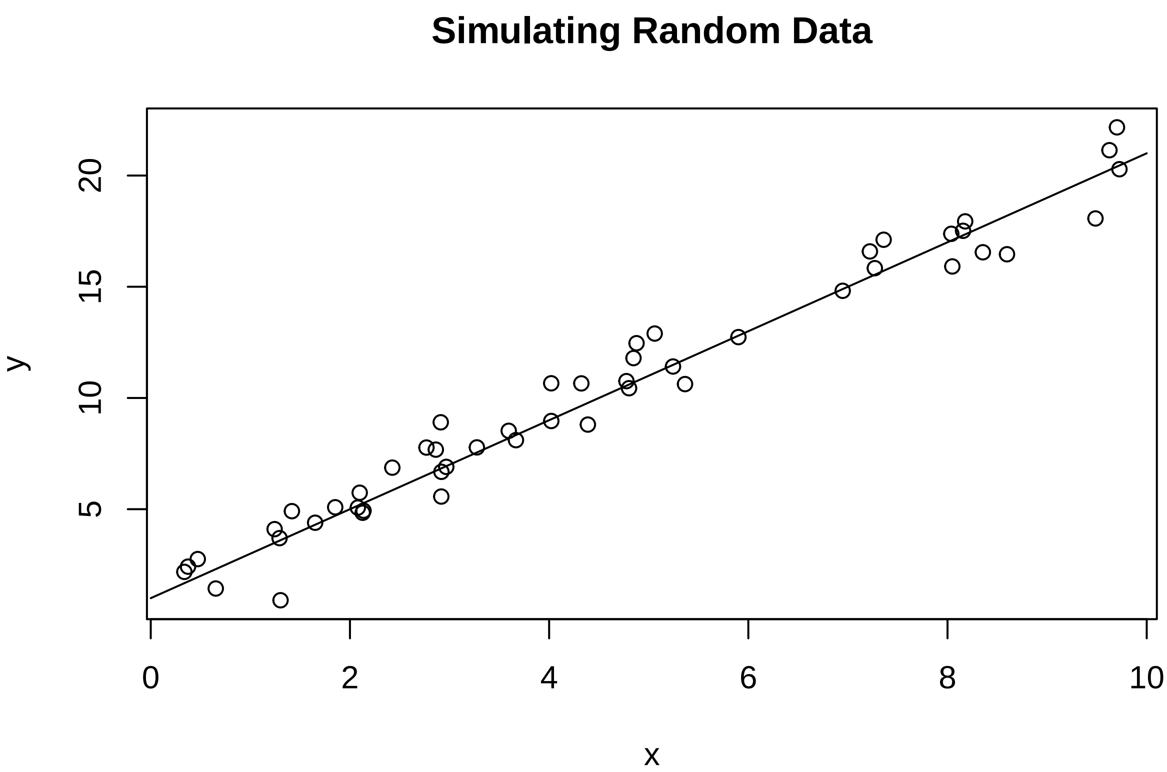
```r
1:5 * 1:5
```

```
## [1]  1  4  9 16 25
```

Exercise: Simulating Random Data

- Create a vector x of length 50 sampled uniformly from the interval [0,10]
- Create a vector err of length 50 sampled from the standard normal distribution
- Create a vector y by the formula $y = 2\ x + 1 + err$
- Plot x versus y in a scatter plot
- Create vectors $X = [0,1,2,\ldots,10]$ and $Y = 2\ X + 1$ and plot the line X versus Y in the same plot

```r
n <- 50
x <- runif(n,0,10)
err <- rnorm(n)
y <- 2*x + 1 + err
plot(x,y)
X <- 0:10
Y <- 2*X + 1
lines(X,Y,'l')
title('Simulating Random Data')
```



Exercise

Create the following vectors without using the function `c()`:

- [1,2,3,1,2,3,1,2,3,1,2,3]
- [5,4,3,2,1]
- [-100,-75,-50,-25,0,25,50,75,100]

2. Matrices

We use the function `matrix(data,nrow,ncol,byrow=FALSE)` to create a matrix from the entries in *data*. The shape of the matrix is determined by *nrow* and *ncol*. The input *data* can be a vector or a value and the `matrix()` function will fill in the values of the matrix. For example:

```
matrix(1:4,2,byrow=TRUE)
```

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

```
matrix(1,3,2)
```

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

```
matrix(sample(-9:9,9,replace=TRUE),nrow=3)
```

```
##      [,1] [,2] [,3]
## [1,]    3   -3    1
## [2,]    0   -2    5
## [3,]    7    2   -9
```

Arithmetic operations are **performed elementwise**. For example:

```
A <- matrix(1:4,2,2,byrow=TRUE)
A * A
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    9   16
```

```
B <- matrix(2,2,2,byrow=TRUE)
A / B
```

```
##      [,1] [,2]
## [1,]  0.5    1
## [2,]  1.5    2
```

Exercises

1. Create the matrix A = [2,3; -4,1].

```
A <- matrix( c(2,-4,3,1) , nrow=2)
A
```

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

2. Create the matrix B = [1,1,0; -2,5,2].

```
B <- matrix( c(1,1,0,-2,5,2) , nrow=2 , byrow=TRUE)
B
```

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

3. Matrix Operations: Multiplication, Inverse and Transpose

We must use the `%*%` operator to perform matrix multiplication. For example, we can take the dot product of a vector `v` with itself using the operator `%*%` along with the transpose function `t(v)`:

```
v <- matrix( c(1,2) , nrow=1 )
w <- t(v)
v %*% w
```

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

```
A <- matrix( sample(-1:1,16,replace=TRUE) , nrow=4 , byrow=TRUE )
A
```

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

```
t(A)
```

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

```
det(A)
```

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

```
B <- solve(A)
B
```

```
##            [,1] [,2]       [,3]       [,4]
## [1,]  0.6666667    0  0.3333333 -0.3333333
## [2,] -1.0000000   -1  0.0000000  0.0000000
## [3,]  0.6666667    1  0.3333333  0.6666667
## [4,]  0.3333333    0 -0.3333333  0.3333333
```

```
A %*% B
```

```
##               [,1] [,2]         [,3] [,4]
## [1,]  1.000000e+00    0 0.000000e+00    0
## [2,] -5.551115e-17    1 0.000000e+00    0
## [3,]  5.551115e-17    0 1.000000e+00    0
## [4,] -5.551115e-17    0 5.551115e-17    1
```

4. Linear Regression

Consider a set of n data points: $(x_1, y_1), \ldots, (x_n, y_n)$. The linear model which best fits the data (by minimizing the sum of squared errors) is:

$$y = a_1 + a_2 x$$

where the coefficients a_1 and a_2 are determined by

$$A = (M^T M)^{-1} M^T Y$$

where

$$A = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \ , \ M = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \ , \ Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

```
linreg <- function(x,y) {
  M <- matrix(nrow=length(x),ncol=2)
  M[,1] <- 1 # Set all entries in first column to 1
  M[,2] <- x # Set the second column to the x values
  Y <- matrix(y,ncol=1)
  A <- solve( t(M) %*% M ) %*% t(M) %*% Y
  return(A)
}
```

```
linreg(0:5,0:5 + rnorm(6))
```

```
##            [,1]
## [1,] -0.2732474
## [2,]  1.3123676
```