Linear Algebra in R with Applications to Linear Regression

Patrick Walls

June 23, 2016

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] 1.202539 1.485816 4.097890
```

```r
rnorm(4)
```

```
## [1] -0.5266198  1.6161428 -1.1687849 -0.2176985
```

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

```
##  [1] 3 6 5 4 2 2 6 0 6 1 3 1 2 1 2 6 2 3 0 0
```

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
```

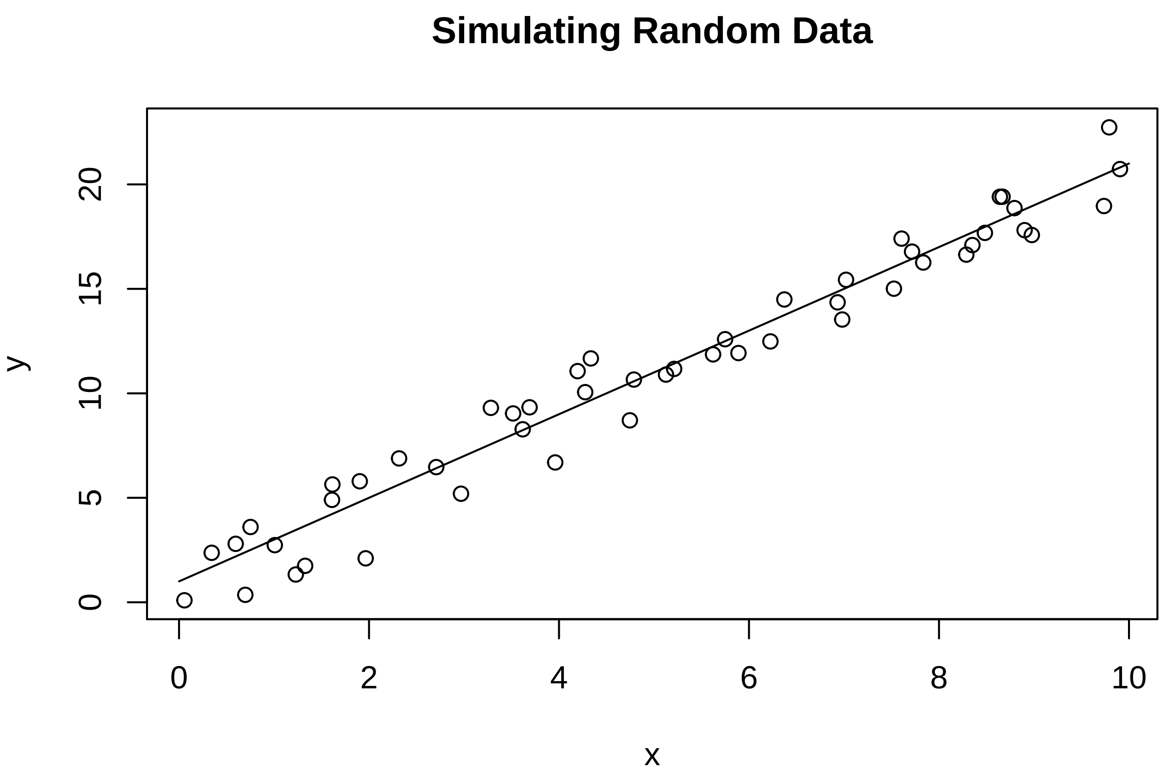
```r
1:5 * 1:5
```

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

Example: 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

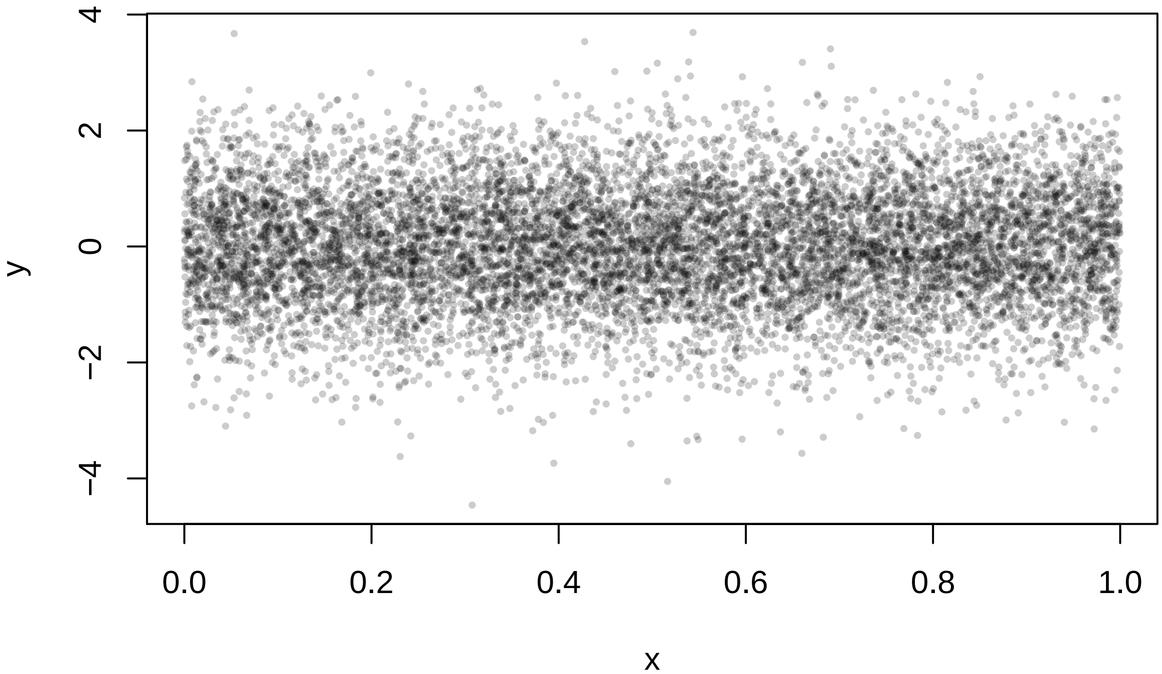
Create the following vectors without using the function `c()`:

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

Exercise

1. Plot the function $y = e^{-x^2}$ for $x \in [-5, 5]$.
2. Plot 10,000 random points (x, y) where x is sampled from the uniform distribution and y is sampled from the standard normal distribution.

```
x <- runif(10000)
y <- rnorm(10000)
plot(x,y,col=rgb(0,0,0,0.2),pch=16,cex=0.5)
```



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,]   -8    8    3
## [2,]   -6   -4   -3
## [3,]    7   -7   -8
```

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

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

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

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

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

Exercise

1. Create the following matrices:

- $\begin{bmatrix} 2 & 3 \\ -4 & 1 \end{bmatrix}$
- $\begin{bmatrix} 1 & 1 & 0 \\ -2 & 5 & 2 \end{bmatrix}$

2. Create a 5 by 5 matrix with alternating entries 1, -1, 1, -1, etc. (starting in position (1,1) and proceeding by row).

3. Create a 10 by 10 matrix with entries randomly sampled from $\{0, 1\}$.

4. Create the 4 by 4 identity matrix.

3. Matrix Operations: Multiplication, Inverse and Transpose

We must use the `%*%` operator to perform matrix multiplication. For example:

$$\begin{bmatrix} 2 & 3 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 2 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} -2 & 10 \\ 1 & 0 \end{bmatrix}$$

```r
A <- matrix(c(2,3,-1,1),nrow=2,byrow=TRUE)
A
```

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

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

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

```r
A %*% B
```

```
##      [,1] [,2]
## [1,]   -2   10
## [2,]    1    0
```

We use the function `t()` to compute the transpose of a matrix.

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

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

```r
t(A)
```

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

For example, we can take the dot product of a vector v with itself using the operator `%*%` along with the transpose function *t(v)*:

```r
v <- matrix( c(1,2) , nrow=1 )
w <- t(v)
v %*% w # 1^2 + 2^2 = 5
```

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

We use the function `solve()` to compute the inverse of a matrix:

```r
A <- matrix( sample(-10:10,16,replace=TRUE) , nrow=4 , byrow=TRUE )
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0    2    1   -3
## [2,]   -4    0   -3  -10
## [3,]    9   -4   -7   -9
## [4,]   -2   -4    9   -9
```

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

```
##              [,1]        [,2]        [,3]         [,4]
## [1,]  0.13646055 -0.10234542  0.06609808  0.002132196
## [2,]  0.32899787 -0.02174840 -0.02345416 -0.062046908
## [3,]  0.09381663 -0.07036247 -0.01705757  0.063965885
## [4,] -0.08272921 -0.03795309 -0.02132196 -0.020042644
```

```r
A %*% B
```

```
##               [,1]          [,2]          [,3]          [,4]
## [1,]  1.000000e+00 -6.938894e-17 -1.387779e-17 -6.245005e-17
## [2,] -1.110223e-16  1.000000e+00  0.000000e+00  5.551115e-17
## [3,] -2.220446e-16  5.551115e-17  1.000000e+00 -2.775558e-17
## [4,]  0.000000e+00  5.551115e-17 -5.551115e-17  1.000000e+00
```

Exercise

1. Create a 3 by 3 matrix A with integer entries randomly sampled from $[-99, 99]$ and compute $A^T A$. Do you notice something special about the result?
2. Find the inverse of $\begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}$ and confirm $A^{-1}A = I$.

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

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

Exercise

Write a function `linreg(x,y)` which takes vectors x and y of the same length and return the coefficients of the linear regression. Also, the function should plot the points and the linear model, and return the R2 value.

```r
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)
```

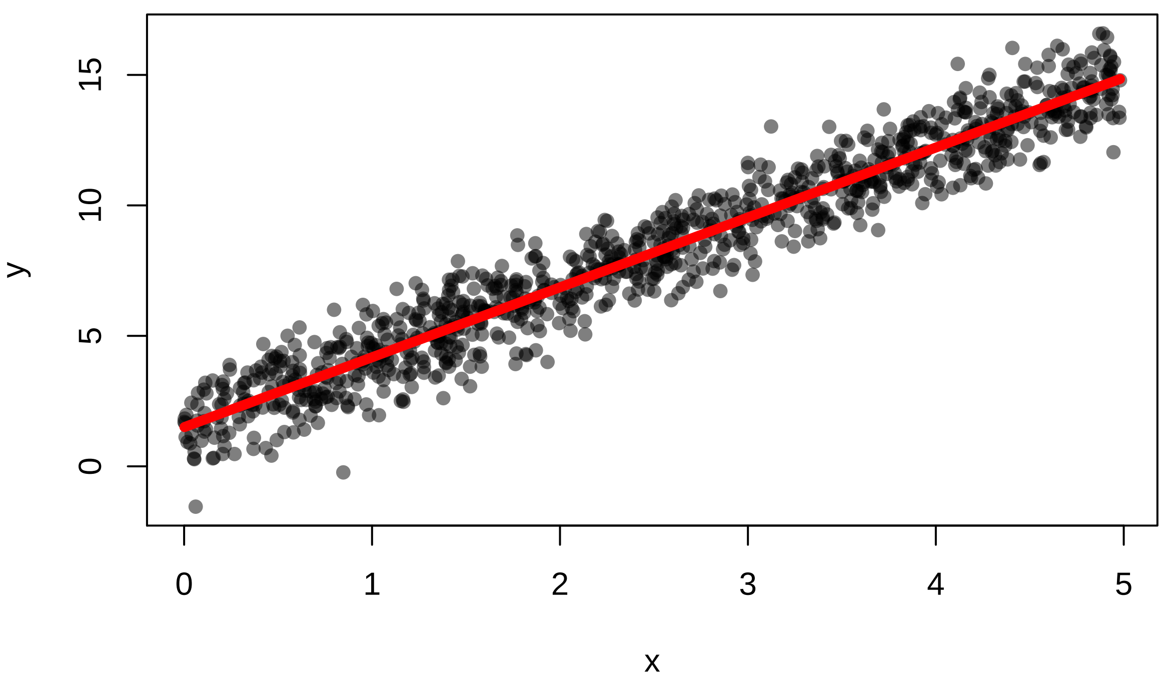
```r
  A <- solve( t(M) %*% M ) %*% t(M) %*% Y

  plot(x,y,col=rgb(0,0,0,0.5),pch=16)
  X <- seq(min(x),max(x),length=3)
  lines(X, A[1,1] + A[2,1]*X,col='red',lwd=5)

  ymean <- sum(y) / length(y)
  total_err <- sum( (y - ymean)^2 )
  residues <- sum( (M %*% A - y)^2 )
  R2 <- 1 - residues / total_err

  print( paste('Linear model: y = ',
               round(A[1,1],2), ' + ',
               round(A[2,1],2),
               'x, and R2 = ',
               round(R2,2),'.',sep=''))
  return(A)
}
```

```r
x <- runif(1000,0,5)
err <- rnorm(1000)
y <- 2.7*x + 1.4 + err
coefs <- linreg(x,y)
```




```
## [1] "Linear model: y = 1.5 + 2.68x, and R2 = 0.93."
```