ISLR Lab 2

Suarez Rodes, Randy

Basic Commands

Vector creatinon c() "concatenate" function. Use <- or = to declare variables or functions.

```
x <- c(1,3,2,5)
x
```

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

```
x = c(1,6,2)
x
```

```
## [1] 1 6 2
```

```
y = c(1,4,3)
```

Adding vectors pointwise.

```
length(x)
```

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

```
length(y)
```

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

```
x+y
```

```
## [1]  2 10  5
```

ls() and rm() (List of objects and remove)

```
ls()
```

```
## [1] "x" "y"
```

```
rm(x,y)
ls()
```

```
## character(0)
```

Removing all at once

```
rm(list=ls())
```

The matrix function ?matrix to learn more about it (? it is used to get documentation help) For matrix specify data, number of rows and cols.

```
x=matrix (data=c(1,2,3,4) , nrow=2, ncol =2)
x
```

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

We can specify the order of the entries using byrow, by default byrow=FALSE, hence the entries are created by column as the default setting.

```
x=matrix (data=c(1,2,3,4) , nrow=2, ncol =2,byrow = TRUE)
x
```

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

Functions work in a vectorized fashion (entry wise)

```
sqrt(x) #square root
```

```
##          [,1]     [,2]
## [1,] 1.000000 1.414214
## [2,] 1.732051 2.000000
```

```
x^2 #power
```

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

Random Data generation Ex:rnorm() Normal random variates. Random generation changes each call (use set.seed() to fix a random generator seed) cor() correlation

```
x=rnorm (50)
y=x+rnorm (50, mean=50, sd=.1)
cor(x,y)
```

```
## [1] 0.9922775
```

```
set.seed (1303)
rnorm (50)
```

```
##  [1] -1.1439763145  1.3421293656  2.1853904757  0.5363925179  0.0631929665
##  [6]  0.5022344825 -0.0004167247  0.5658198405 -0.5725226890 -1.1102250073
## [11] -0.0486871234 -0.6956562176  0.8289174803  0.2066528551 -0.2356745091
## [16] -0.5563104914 -0.3647543571  0.8623550343 -0.6307715354  0.3136021252
## [21] -0.9314953177  0.8238676185  0.5233707021  0.7069214120  0.4202043256
## [26] -0.2690521547 -1.5103172999 -0.6902124766 -0.1434719524 -1.0135274099
## [31]  1.5732737361  0.0127465055  0.8726470499  0.4220661905 -0.0188157917
## [36]  2.6157489689 -0.6931401748 -0.2663217810 -0.7206364412  1.3677342065
## [41]  0.2640073322  0.6321868074 -1.3306509858  0.0268888182  1.0406363208
## [46]  1.3120237985 -0.0300020767 -0.2500257125  0.0234144857  1.6598706557
```

mean() and var() of a vector, apply sqrt() to the output of var() to obtain standard deviation or simply use sd()

```
set.seed(3)
y=rnorm (100)
mean(y)
```

```
## [1] 0.01103557
```

```
var(y)
```

```
## [1] 0.7328675
```

```
sqrt(var(y))
```

```
## [1] 0.8560768
```

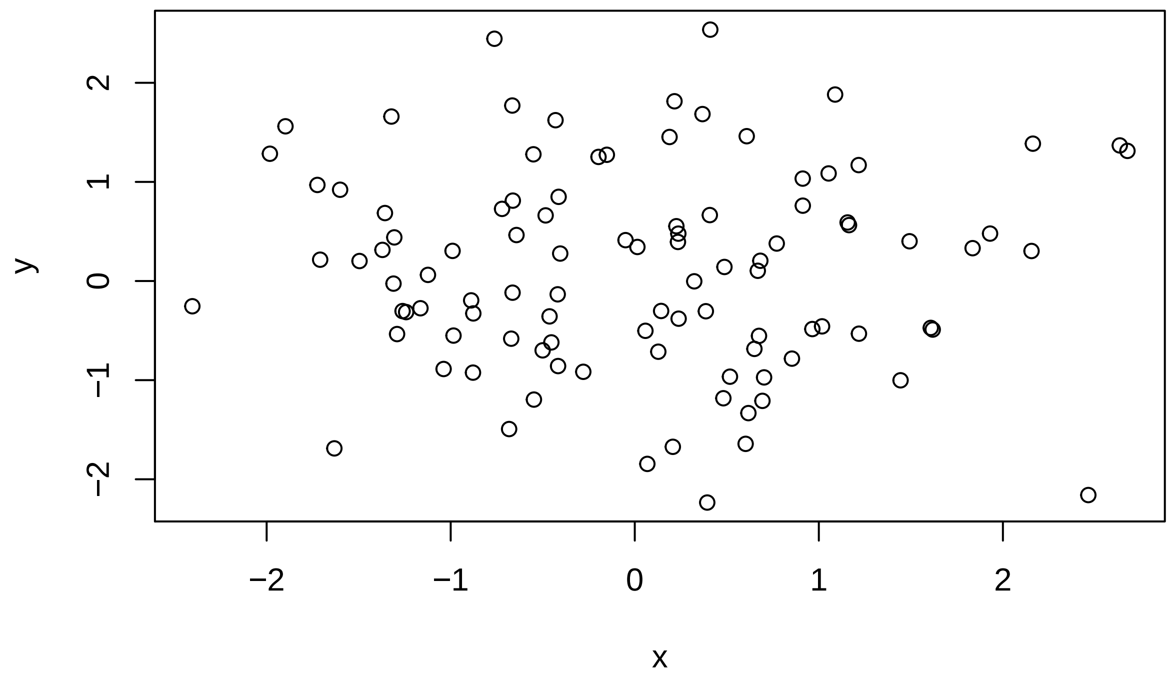
```
sd(y)
```

```
## [1] 0.8560768
```

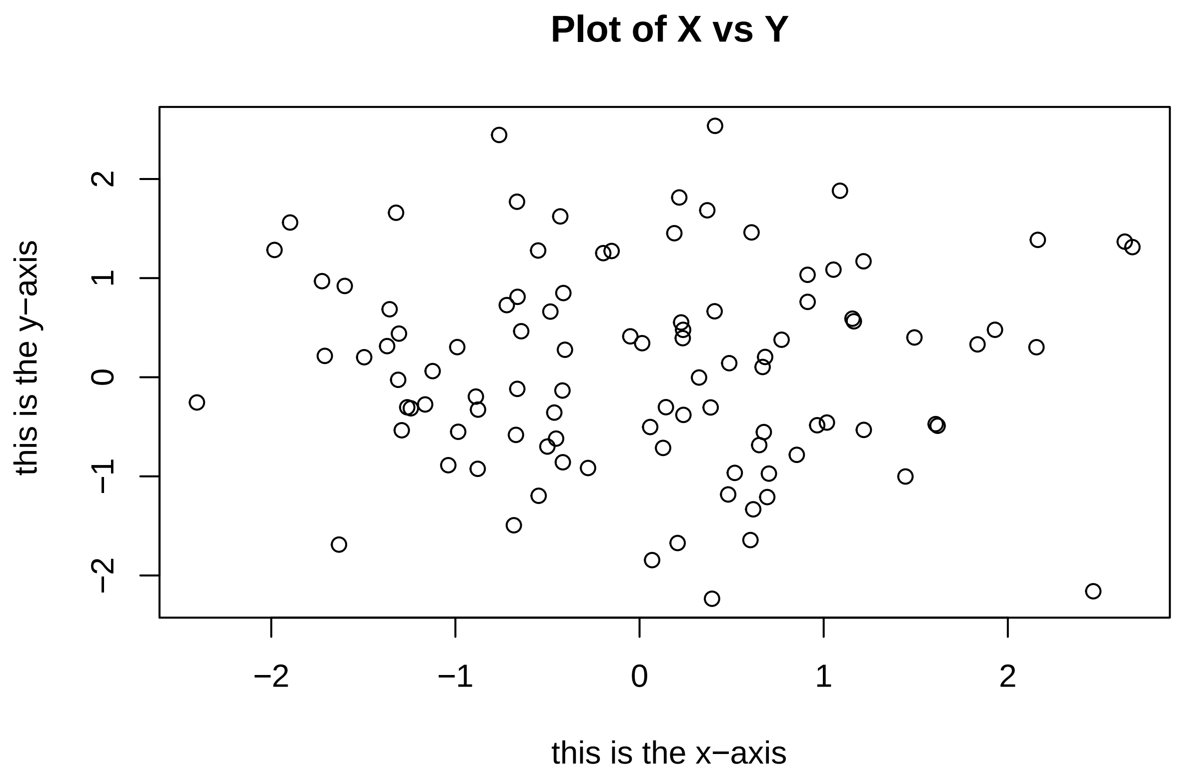
Graphics

plot() function. Example plot(x,y) produces a scatterplot of the numbers in x versus the numbers in y.

```
x=rnorm (100)
y=rnorm (100)
plot(x,y)
```



```
plot(x,y,xlab=" this is the x-axis",ylab=" this is the y-axis", main=" Plot of X vs Y")
```



To create a pdf, we use the pdf() function, and to create a jpeg, pdf() we use the jpeg() function

pdf (" Figure .pdf “) plot(x,y,col =” green ") dev.off () null device

The function dev.off() indicates to R that we are done creating the plot.

The function seq() can be used to create a sequence of numbers. For instance, seq(a,b) makes a vector of integers between a and b. There are many other options: for instance, seq(0,1,length=10) makes a sequence of 10 numbers that are equally spaced between 0 and 1. Typing 3:11 is a shorthand for seq(3,11) for integer arguments.

```
x=seq (1 ,10)
x
```

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

```
x=1:10
x
```

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

```
x=seq(-pi ,pi ,length =50)
```

Contour Plot

Function contour()