Assignment0

Todo: Calculator

```
((2018-2014)/(2014-1994))*100
```

```
## [1] 20
```

Todo: Functions

```
m <-(2018 - 2014)
x <- (2014 - 1994)
a <- ((m/x)*100)
a
```

```
## [1] 20
```

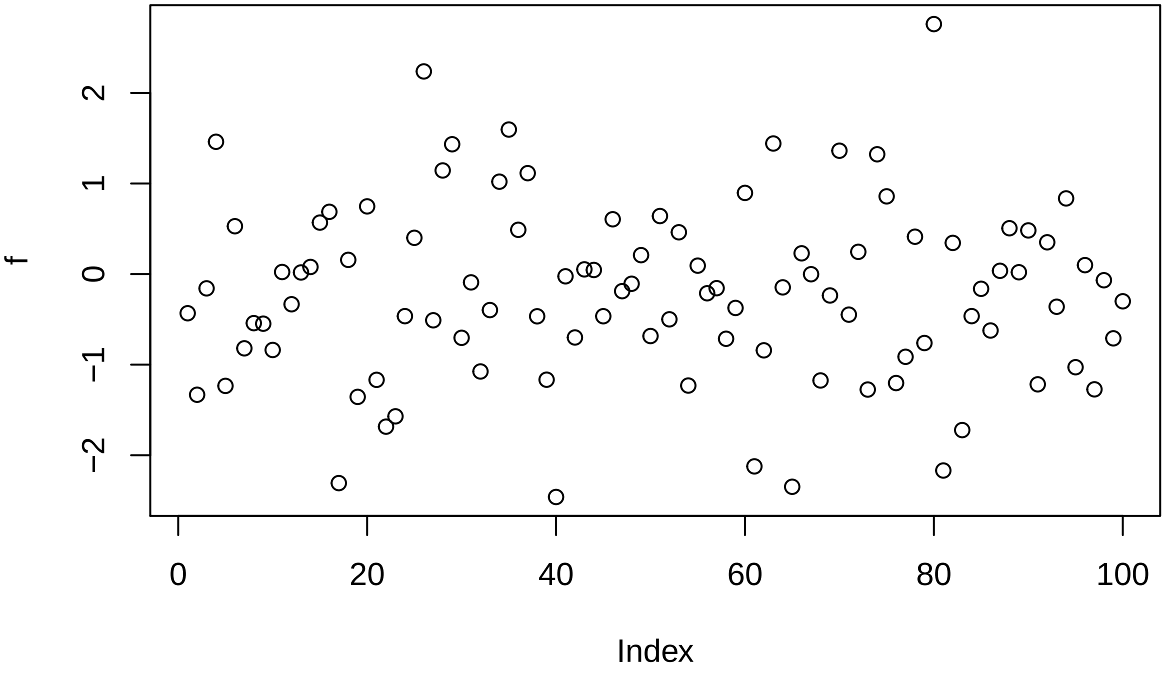
3) Todo: Workplace

```
b= c(4,5,8,11)
sum(b)
```

```
## [1] 28
```

4)Todo: Plots

```
f = rnorm(100)
plot(f)
```



5)Todo: Help

```
help (sqrt)
```

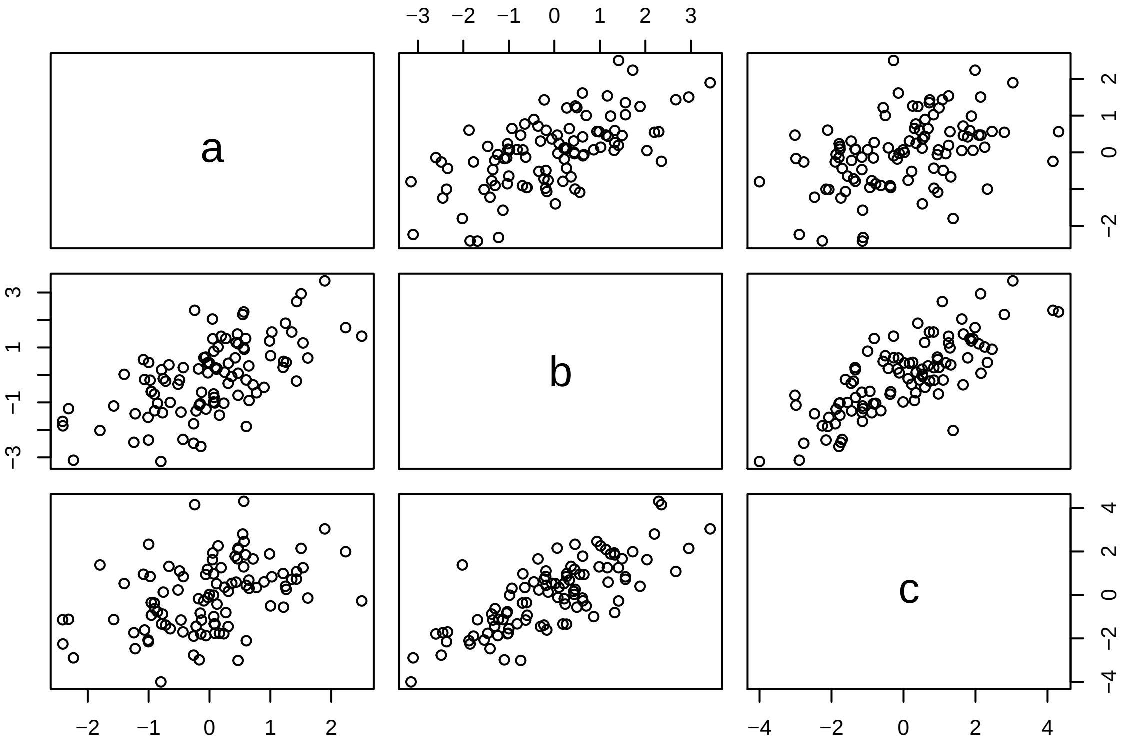
6)Todo: Data Structures

```
P= seq(from= 31, to= 60)
Q= (matrix(data= P, ncol=5, nrow=6))
Q
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   31   37   43   49   55
## [2,]   32   38   44   50   56
## [3,]   33   39   45   51   57
## [4,]   34   40   46   52   58
## [5,]   35   41   47   53   59
## [6,]   36   42   48   54   60
```

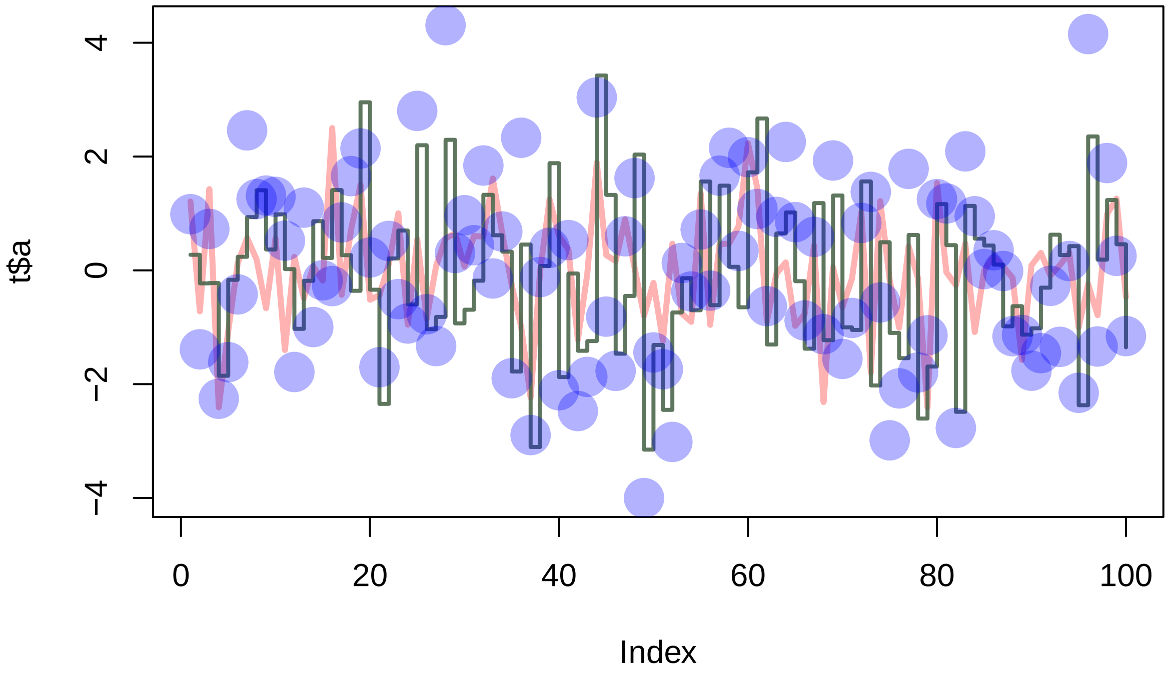
7)Todo: Lists

```
x1= rnorm(100)
x2= rnorm(100)
x3= rnorm(100)
t= data.frame(a = x1, b=(x1+x2), c= (x1+x2+x3))
plot(t)
```



8)Todo: Graphics

```
plot(t$a, type="l", ylim=range(t),
     lwd=3, col=rgb(1,0,0,0.3))
lines(t$b, type="s", lwd=2,
      col=rgb(0.3,0.4,0.3,0.9))
points(t$c, pch=20, cex=4, col=rgb(0,0,1,0.3))
```



Explanation: rgb = red, green and blue. This is used for colours in the group.

9) Todo: Not available data

```
sqrt (rnorm(100))
```

```
## Warning in sqrt(rnorm(100)): NaNs produced
```

```
##   [1] 0.9105822 0.7019308 1.0778409 1.4475873       NaN       NaN 0.6119400
##   [8]       NaN 0.6841878 1.1137968 0.8457343       NaN 1.0663255 1.0456338
##  [15] 1.0452535 0.7416618 1.2510715 0.5344272 0.8567262       NaN       NaN
##  [22]       NaN 0.8215576       NaN       NaN 1.1886330 0.7257968 1.3275678
##  [29]       NaN 0.6395209 1.3857950 1.2549671 1.0701537       NaN       NaN
##  [36] 1.6256415 1.3366528       NaN 0.7776503 0.7869669       NaN 0.3651057
##  [43]       NaN 0.6815222       NaN 0.8442499       NaN 1.0537553 0.8760712
##  [50]       NaN 0.7433433       NaN 1.4643929       NaN       NaN 0.6302708
##  [57]       NaN       NaN       NaN       NaN       NaN       NaN 1.1413418
##  [64] 0.8262495       NaN 0.3145811       NaN 1.1371502 0.5656233       NaN
##  [71]       NaN 0.2699262 0.9171340       NaN       NaN       NaN       NaN
##  [78] 0.7502967 0.8056887       NaN 0.4943219       NaN       NaN       NaN
##  [85]       NaN       NaN       NaN 0.8660330 0.1850874       NaN       NaN
##  [92] 0.7822924       NaN 0.5442033       NaN       NaN       NaN 0.4669886
##  [99] 0.6658300       NaN
```

You get NaNs, not a number.

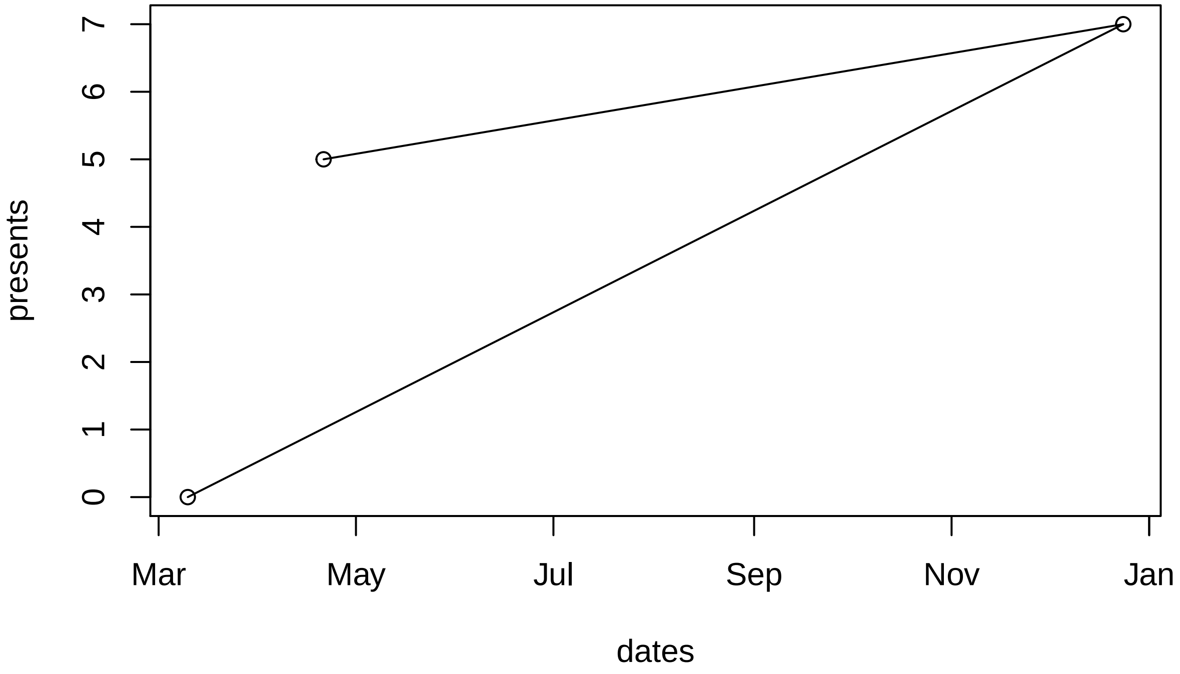
10) Todo: Reading from a file

```
myfile<- read.table(file="tst1.txt" , header =TRUE)
myvar <- myfile$g * 5
write.table (myvar, file="tst2.txt")
```

11)Todo: Dates

```
dates= strptime(c("20190310", "20191224","20190421"),format="%Y%m%d")
presents = c(0,7,5)
plot(dates,presents)
```

```r
lines(dates,presents)
```



12)Todo: For-loop

```r
myvec = seq(from=1, to=100)
s = c()
for(i in 1:length(myvec))
{
  if(myvec[i] < 5) {
    s[i] <- (myvec[i] *10)
  }else if (myvec[i] > 90){
    s[i] <- (myvec[i] *10)
  }else{
    s[i] = (myvec[i]*0.1)
  }
}
s
```

```
##   [1]   10.0   20.0   30.0   40.0    0.5    0.6    0.7    0.8    0.9    1.0
##  [11]    1.1    1.2    1.3    1.4    1.5    1.6    1.7    1.8    1.9    2.0
##  [21]    2.1    2.2    2.3    2.4    2.5    2.6    2.7    2.8    2.9    3.0
##  [31]    3.1    3.2    3.3    3.4    3.5    3.6    3.7    3.8    3.9    4.0
##  [41]    4.1    4.2    4.3    4.4    4.5    4.6    4.7    4.8    4.9    5.0
##  [51]    5.1    5.2    5.3    5.4    5.5    5.6    5.7    5.8    5.9    6.0
##  [61]    6.1    6.2    6.3    6.4    6.5    6.6    6.7    6.8    6.9    7.0
##  [71]    7.1    7.2    7.3    7.4    7.5    7.6    7.7    7.8    7.9    8.0
##  [81]    8.1    8.2    8.3    8.4    8.5    8.6    8.7    8.8    8.9    9.0
##  [91]  910.0  920.0  930.0  940.0  950.0  960.0  970.0  980.0  990.0 1000.0
```

14)Todo: Writing your own functions

```r
myfunc = function(argv1)
{
  S=c()
  for(i in 1:length(argv1))
  {
    if(argv1[i] < 5) {
      S[i] <- (argv1[i] *10)
    }else if (argv1[i] > 90){
      S[i] <- (argv1[i] *10)
    }else{
      S[i] = (argv1[i]*0.1)
    }
  }
  S
}
myfunc(argv1 = c(2:50))
```

```
##  [1] 20.0 30.0 40.0  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4  1.5
## [15]  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9
## [29]  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3
## [43]  4.4  4.5  4.6  4.7  4.8  4.9  5.0
```