InClass-Day3-Haswah.R

renahaswah

2019-06-11

```
library(fBasics)
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

```
library(moments)
```

```
##
## Attaching package: 'moments'
```

```
## The following objects are masked from 'package:timeDate':
##
##     kurtosis, skewness
```

```
data<-read.table("~/Desktop/MATH523/data/m-ibmsp6709.txt",header=T)
head(data)
```

```
##       date       ibm        sp
## 1 19670131  0.075370  0.078178
## 2 19670228  0.079099  0.001963
## 3 19670331  0.048837  0.039410
## 4 19670428  0.100887  0.042239
## 5 19670531 -0.035234 -0.052441
## 6 19670630  0.067024  0.017512
```

```
length(data)
```

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

```
class(data)
```

```
## [1] "data.frame"
```

```
mode(data)
```

```
## [1] "list"
```

```
dim(data)
```

```
## [1] 516   3
```

```
tail(data)
```

```
##         date      ibm        sp
## 511 20090731 0.129381  0.074142
## 512 20090831 0.005681  0.033560
## 513 20090930 0.013215  0.035723
## 514 20091030 0.008361 -0.019762
## 515 20091130 0.052152  0.057364
## 516 20091231 0.036011  0.017771
```

```r
#Store ibm data in list to pull that info
dibm<-data$ibm
class(dibm)
```

```
## [1] "numeric"
```

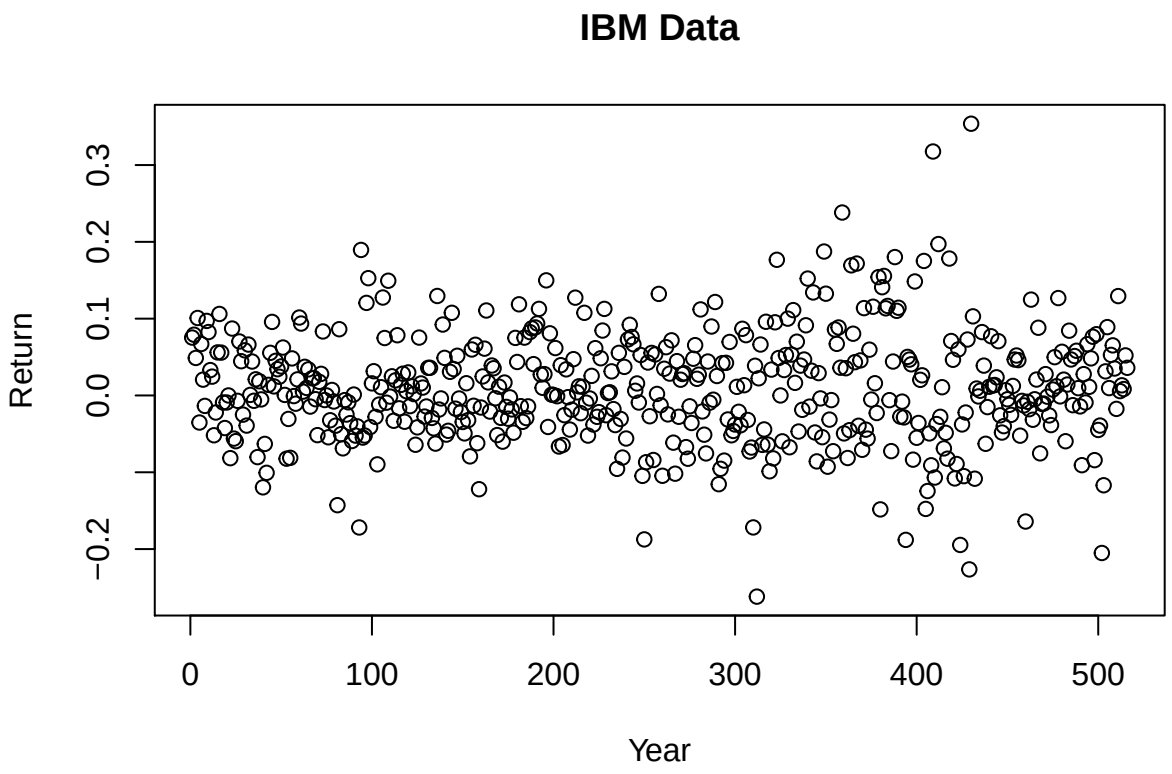
```r
length(dibm)
```

```
## [1] 516
```

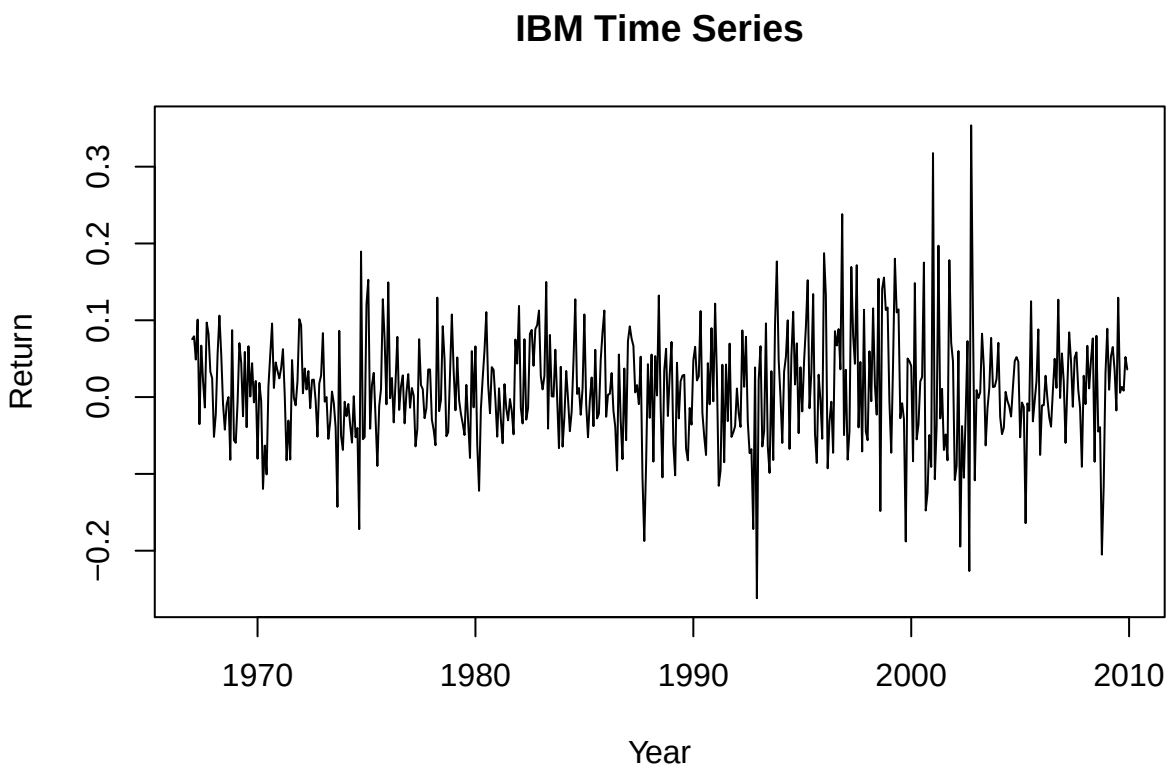
```r
dim(dibm)
```

```
## NULL
```

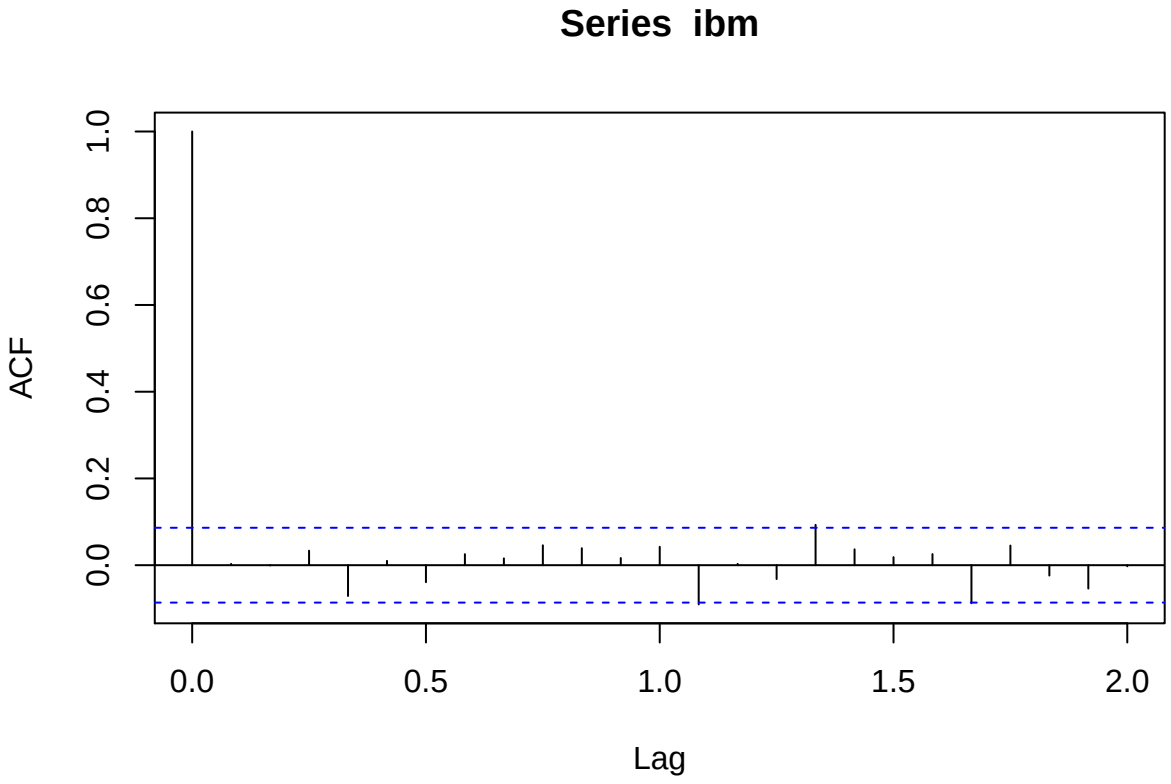
```r
plot(dibm,xlab="Year",ylab="Return",main="IBM Data")
```



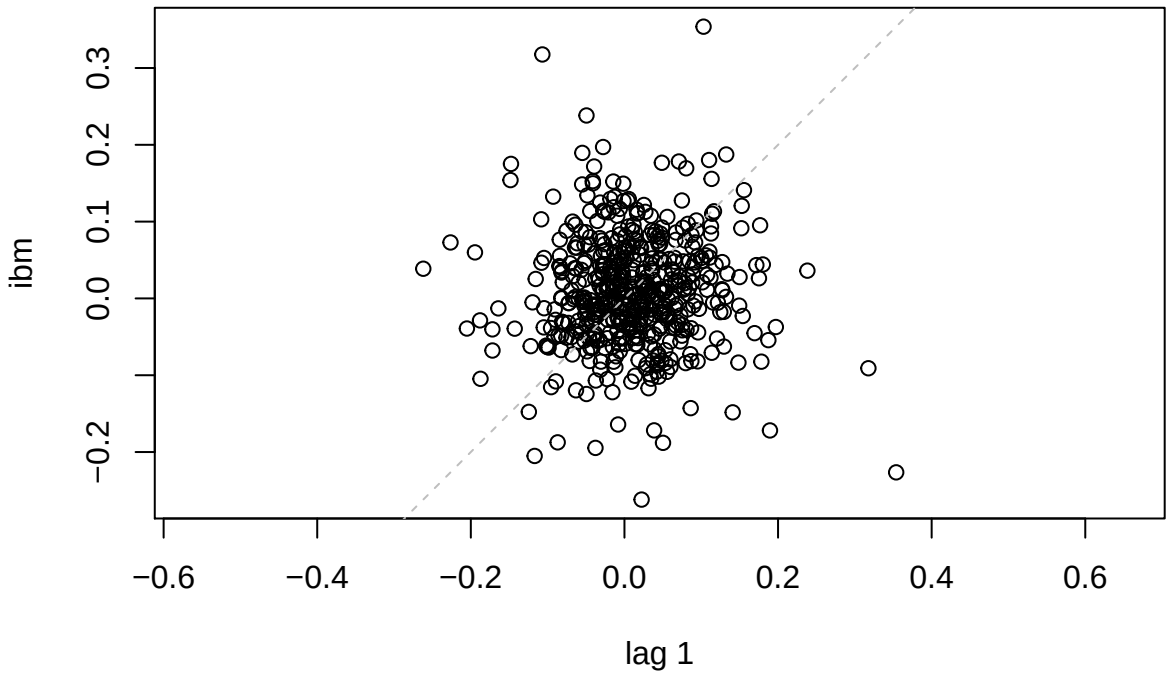

```r
ibm<-ts(dibm,frequency=12, start=c(1967,1))
plot(ibm,xlab="Year",ylab="Return",main="IBM Time Series")
```



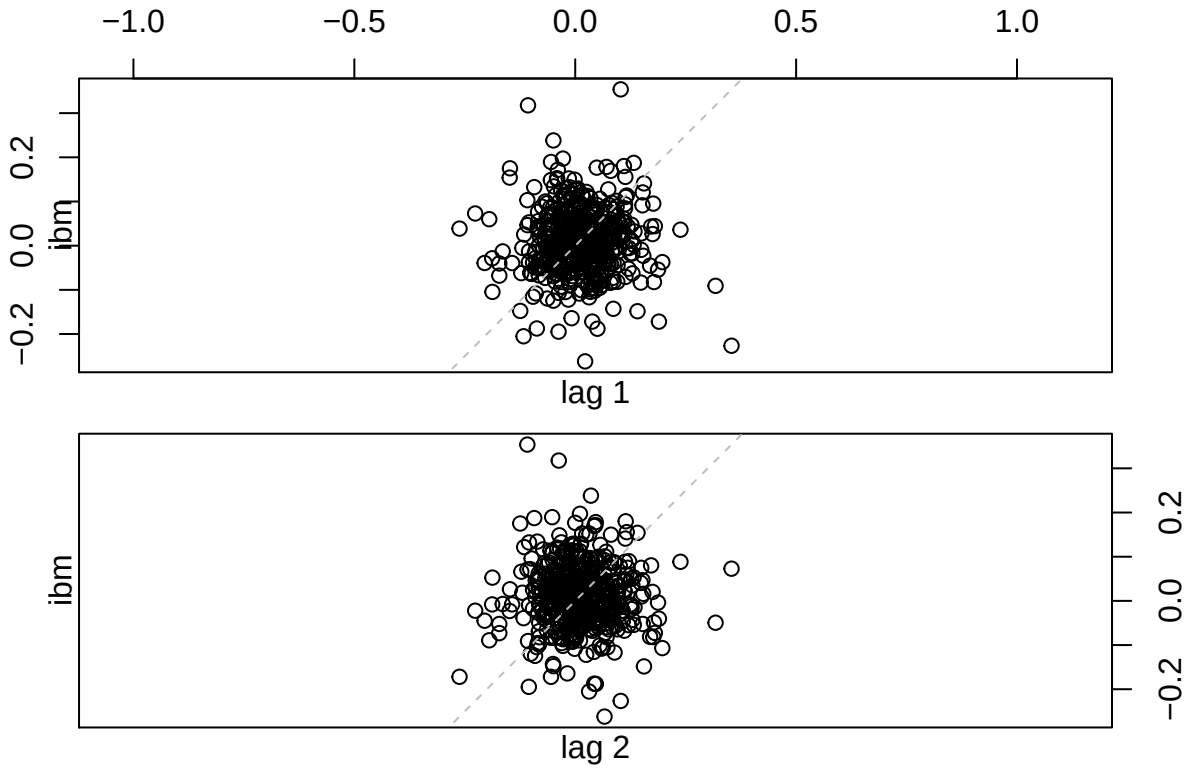

```
plot(acf(ibm,lag=24))
```



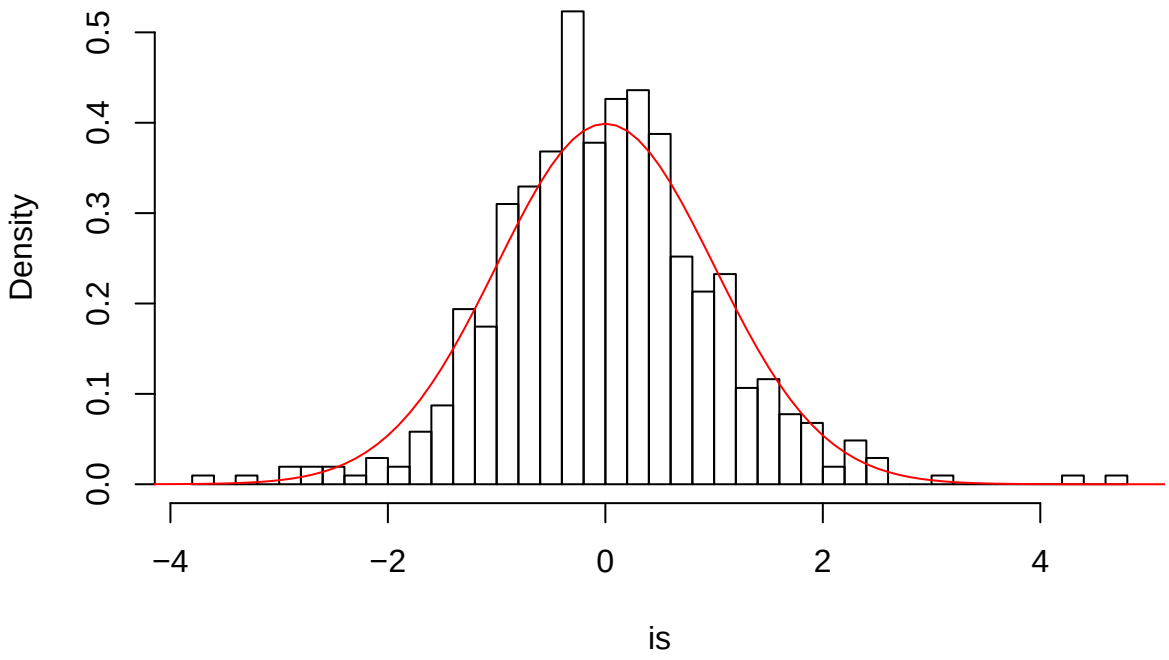

```
lag.plot(ibm, lags = 1)
```




```
lag.plot(ibm, lags = 2)
```




```
is<-(ibm-mean(ibm))/sd(ibm)
hist(is,breaks=34,freq=FALSE)
par(new=T)
s <- seq(-40,40,.1)
lines(s,dnorm(s,0,1),type = "l",col="red")
```




```
skewness(is)
```

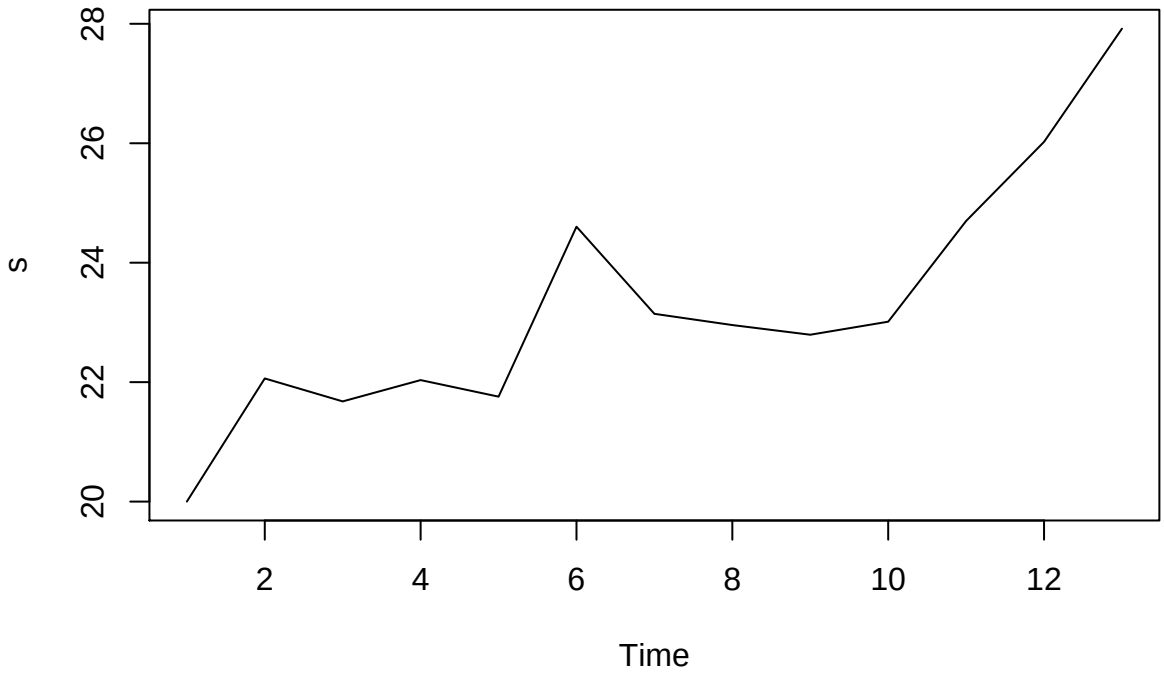
```
## [1] 0.2383948
```

```
kurtosis(is)
```

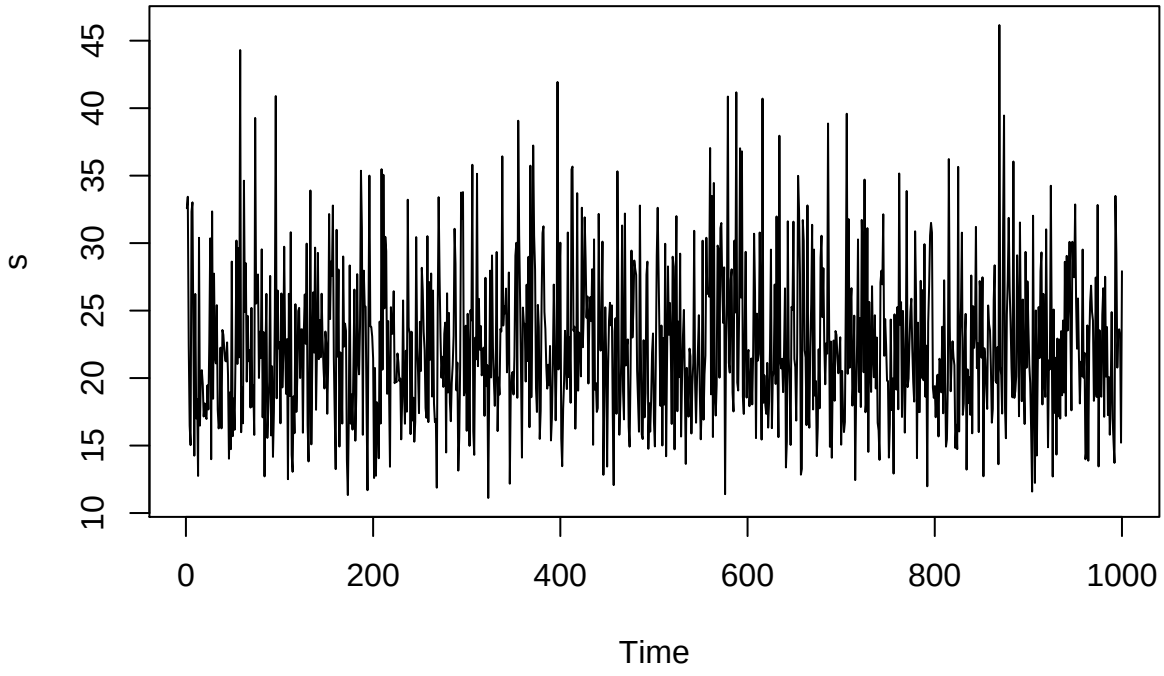
```
## [1] 4.842736
```

```
#We will run 1,000 trials of random walk paths.
niter <- 1000
xx <- rep(0,niter)

# simulation loop
s=c()
s[1]=20
t <- c()
for (j in 1:niter)
{
  for(i in 1:12)
  {
    s[i+1]=s[i]+mean(ibm)*s[i]+sd(ibm)*s[i]*rnorm(1)
  }
  t[j] <- s[length(s)]
}
plot(s, xlab="Time", ylab="s", type="l")
```




```
# The plot is the TS of the final trial. We can tell that
# it is mean reverting.
plot(t, xlab="Time", ylab="s", type="l")
```




```
var(t)
```

```
## [1] 33.08333
```

```
# The mean is close to its starting value.
mean(t)
```

```
## [1] 22.45662
```

```r
stoptime <- 3
m=1000
n=m*stoptime
t=seq((1/m),(1/m))
ss=sample(c(-sqrt(1/m),sqrt(1/m)),n,replace=T)
y=cumsum(ss)

niter <- 1000
xx <- rep(0,niter)
for (i in 1:niter)
{
  ss=sample(c(-sqrt(1/m),sqrt(1/m)),n,replace=T)
  y=cumsum(ss)
  xx[i] <- y[stoptime]
}
plot(y,type="l")

library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following object is masked from 'package:timeSeries':
##
##     time<-
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

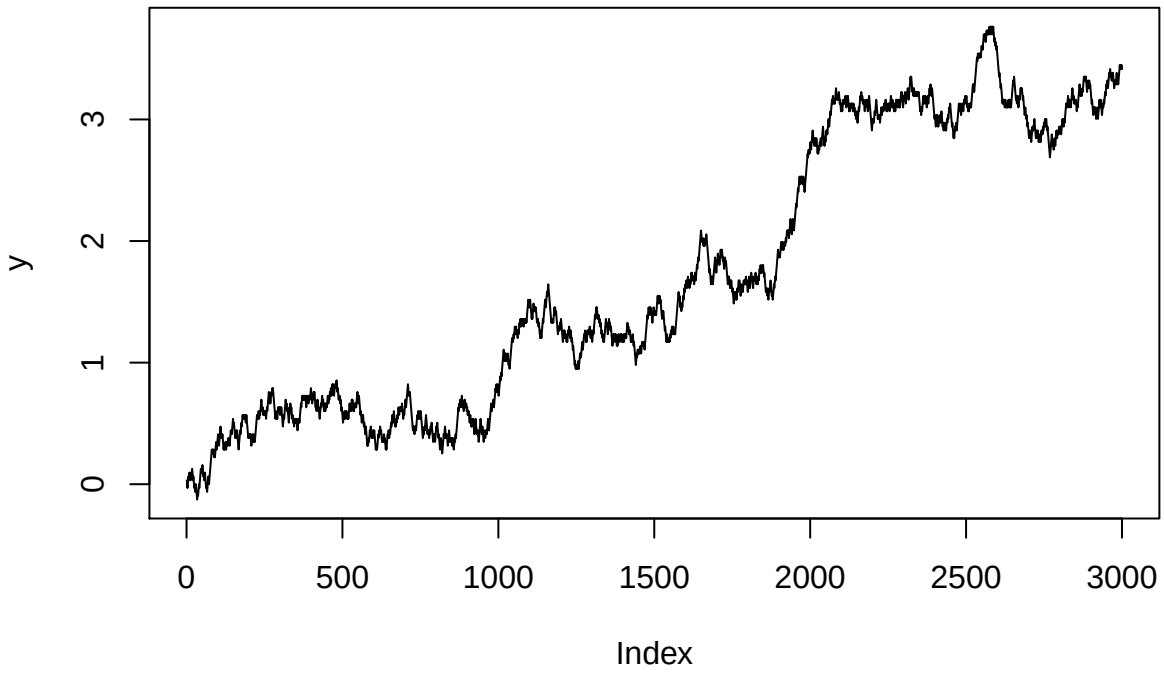
```
## Loading required package: TTR
```

```
##
## Attaching package: 'TTR'
```

```
## The following object is masked from 'package:fBasics':
##
##     volatility
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```




```
sp500=new.env()
getSymbols("^GSPC",env=sp500,src="yahoo",
           from=as.Date("2014-07-24"), to =as.Date("2016-07-24"))
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "^GSPC"
```

```
GSPC=sp500$GSPC
head(GSPC)
```

```
##            GSPC.Open GSPC.High GSPC.Low GSPC.Close GSPC.Volume
## 2014-07-24   1988.07   1991.39  1985.79    1987.98  3203530000
## 2014-07-25   1984.60   1984.60  1974.37    1978.34  2638960000
## 2014-07-28   1978.25   1981.52  1967.31    1978.91  2803320000
## 2014-07-29   1980.03   1984.85  1969.95    1969.95  3183300000
## 2014-07-30   1973.21   1978.90  1962.42    1970.07  3448250000
## 2014-07-31   1965.14   1965.14  1930.67    1930.67  4193000000
##            GSPC.Adjusted
## 2014-07-24       1987.98
## 2014-07-25       1978.34
## 2014-07-28       1978.91
## 2014-07-29       1969.95
## 2014-07-30       1970.07
## 2014-07-31       1930.67
```

```
mode(GSPC)
```

```
## [1] "numeric"
```

```
length(GSPC)
```

```
## [1] 3024
```

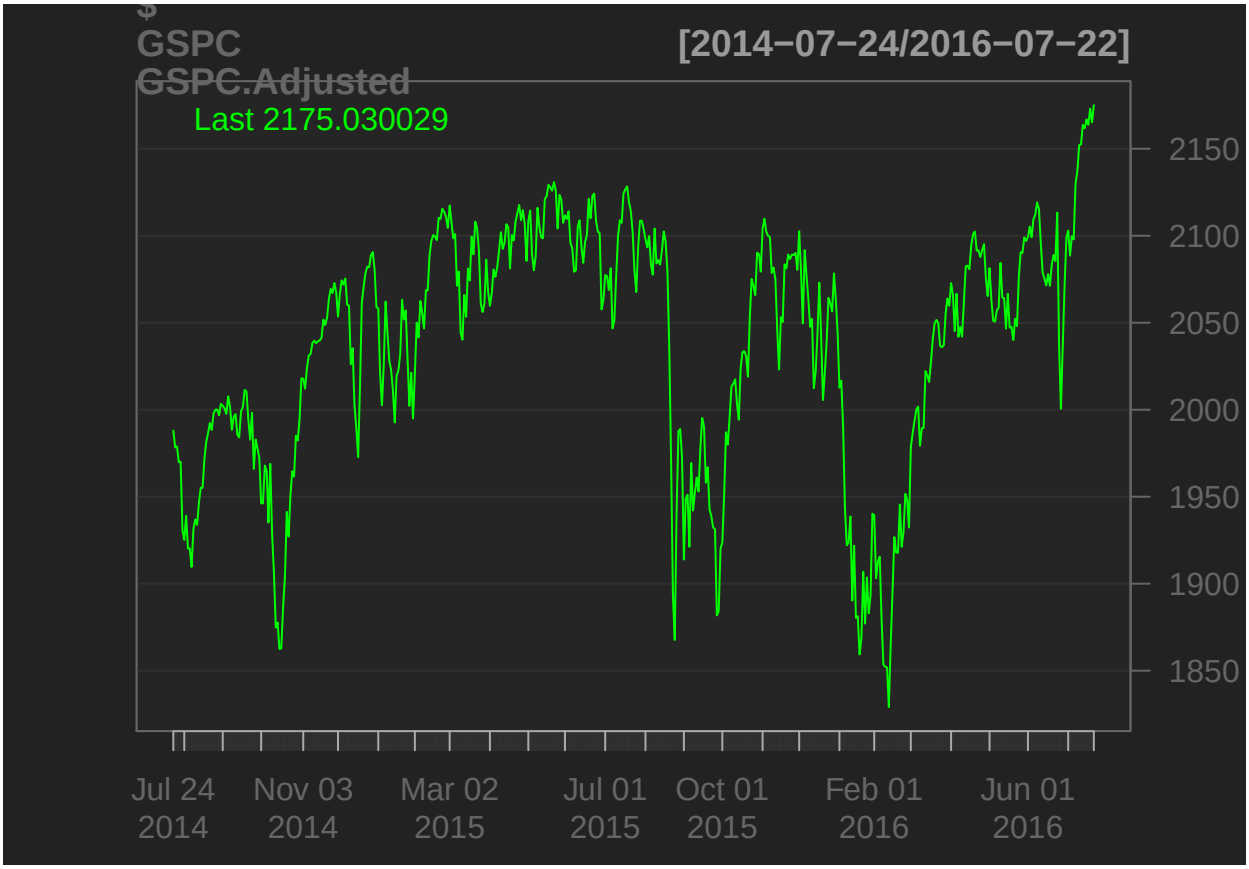
```
dim(GSPC)
```

```
## [1] 504   6
```

```
class(GSPC)
```

```
## [1] "xts" "zoo"
```

```
GA<-GSPC$GSPC.Adjusted
chartSeries(GSPC$GSPC.Adjusted)
```




```
#Kurtosis and IBM Stuff
require("Ecdat")
```

```
## Loading required package: Ecdat
```

```
## Loading required package: Ecfun
```

```
##
## Attaching package: 'Ecfun'
```

```
## The following object is masked from 'package:base':
##
##     sign
```

```
##
```

```
## Attaching package: 'Ecdat'
```

```
## The following object is masked from 'package:datasets':
##
##     Orange
```

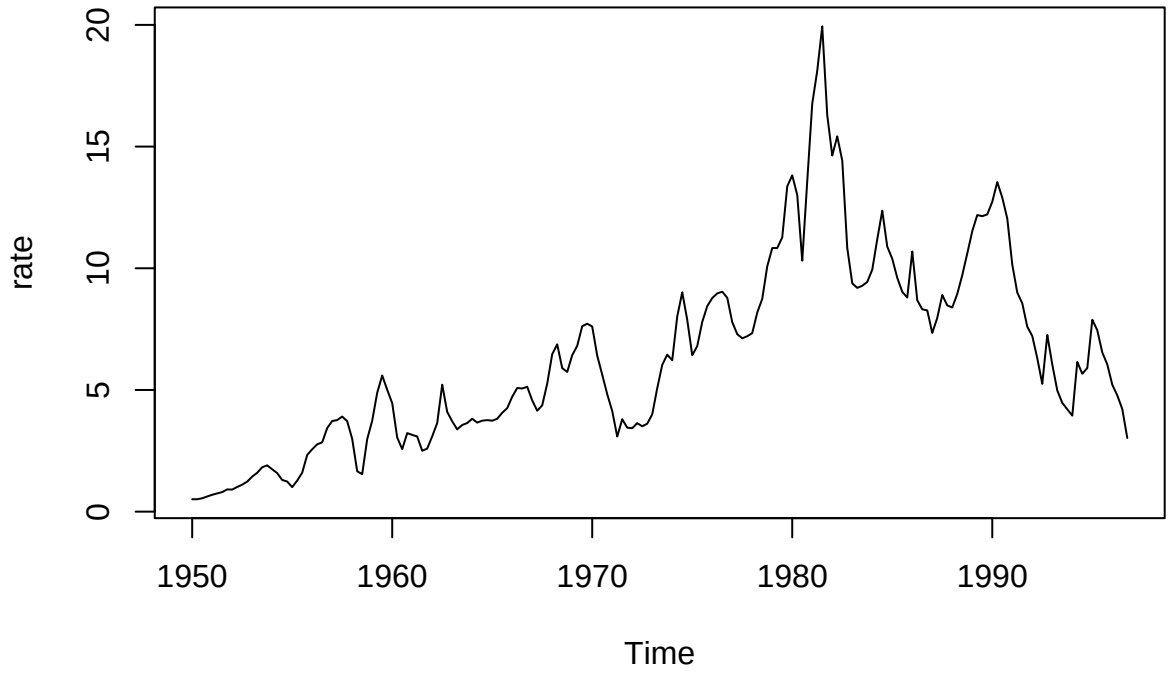
```
data(Tbrate,package="Ecdat")
head(Tbrate)
```

```
## [1] 0.510 0.510 0.550 0.623 0.693 0.750
```

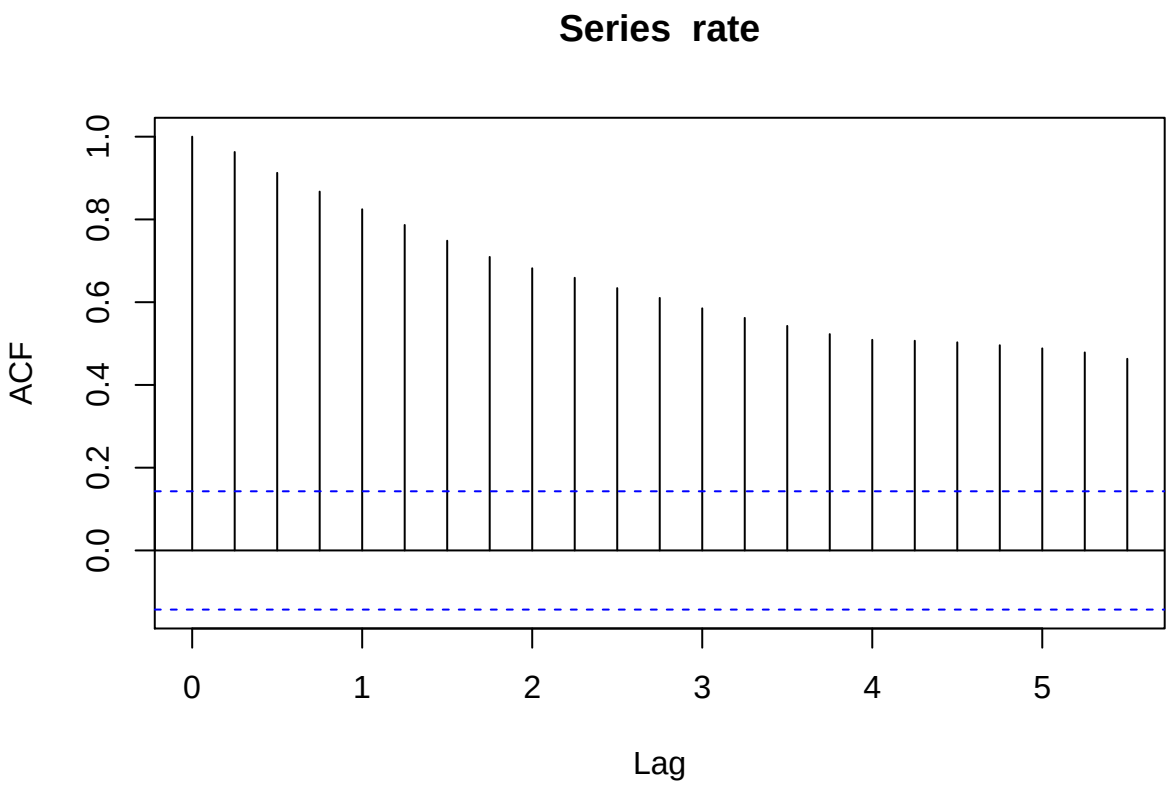
```
rate=Tbrate[,1]
class(Tbrate)
```

```
## [1] "mts" "ts"
```

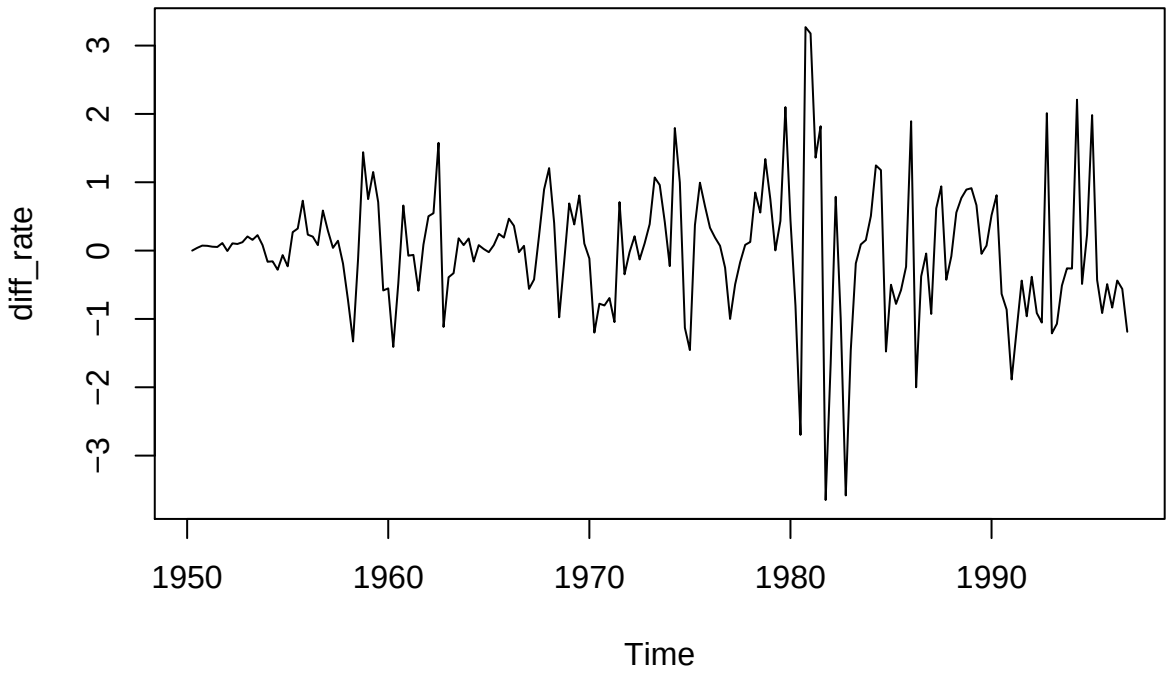
```
#Take a look at rate over time and its lag
plot(rate)
```



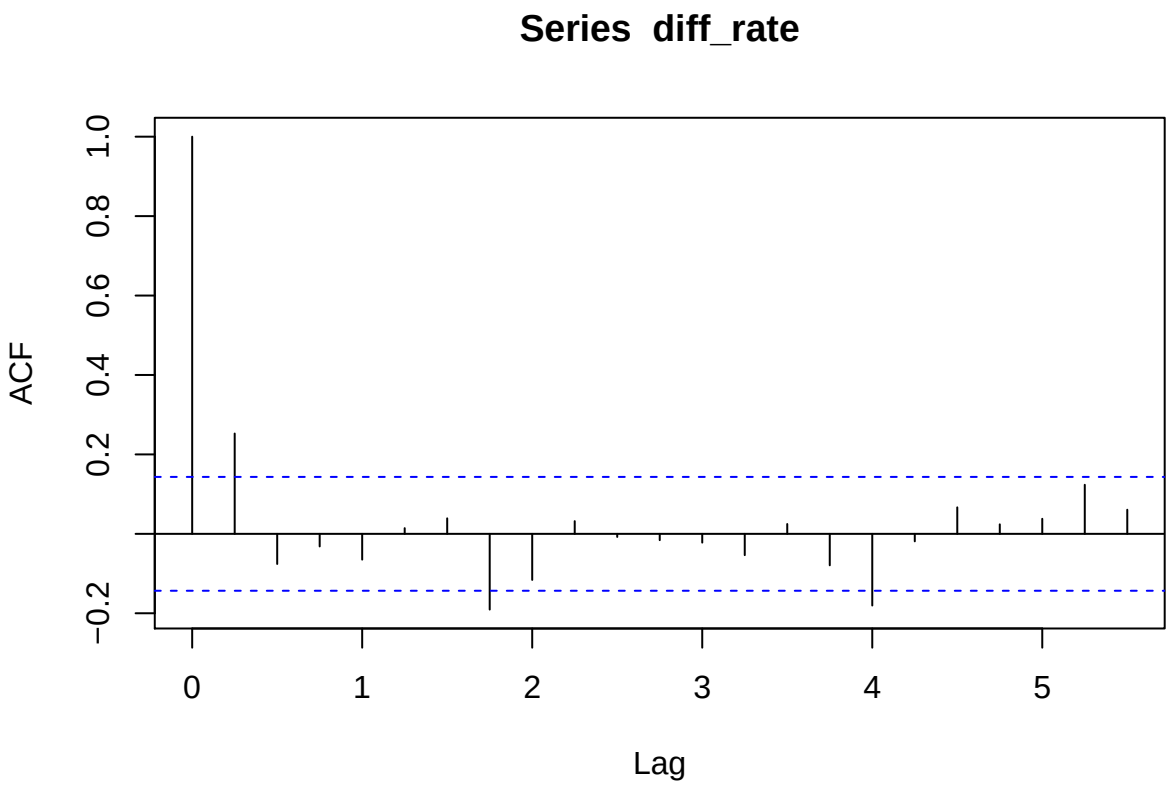

```
acf(rate)
```



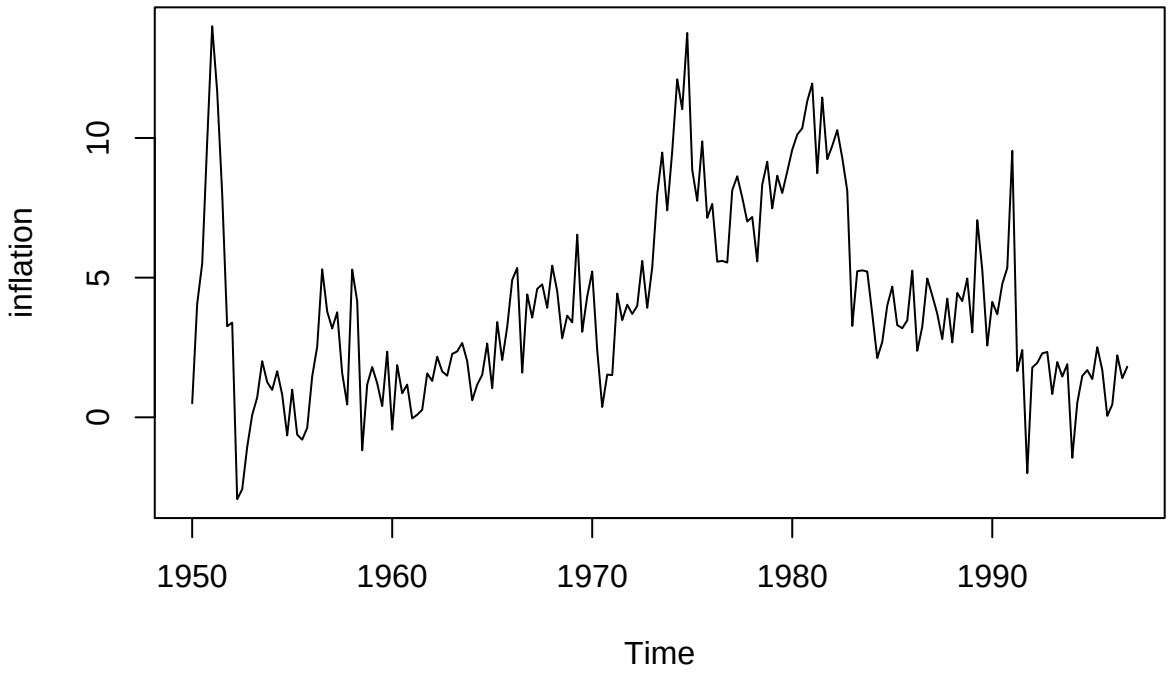

```
#It is not stationary
#look at difference of rates
diff_rate=diff(rate)
plot(diff_rate)
```



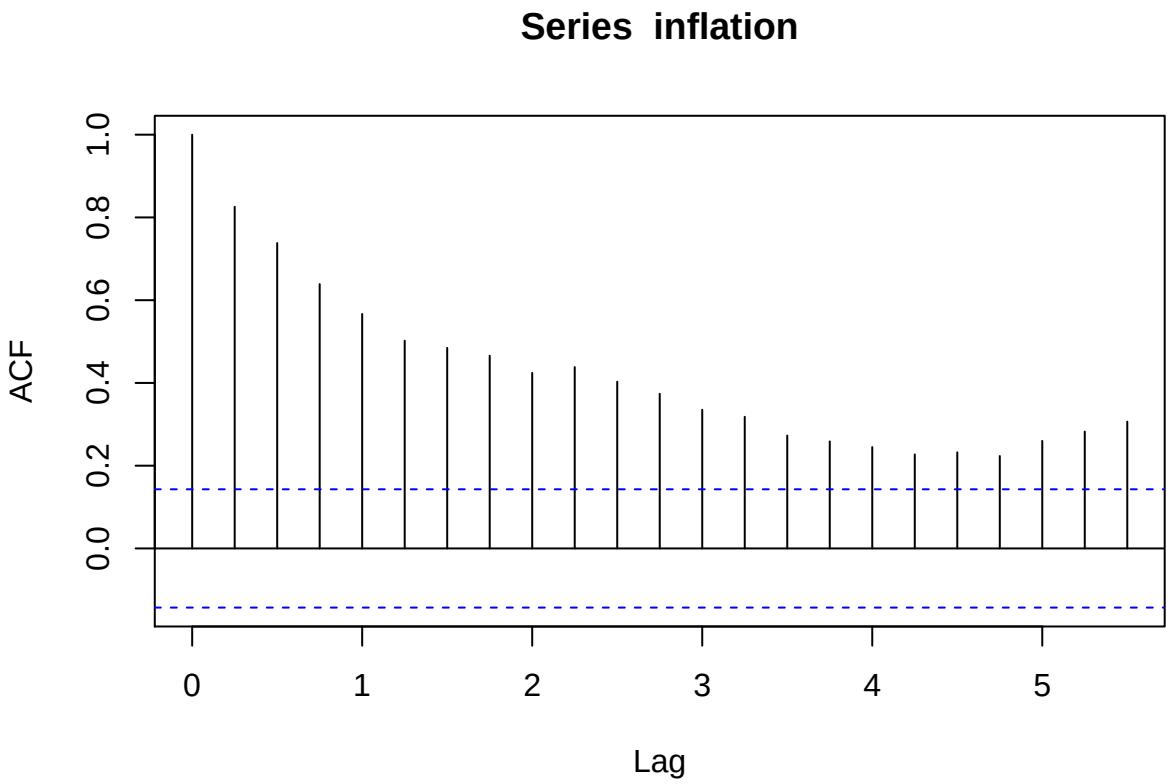

```
acf(diff_rate)
```



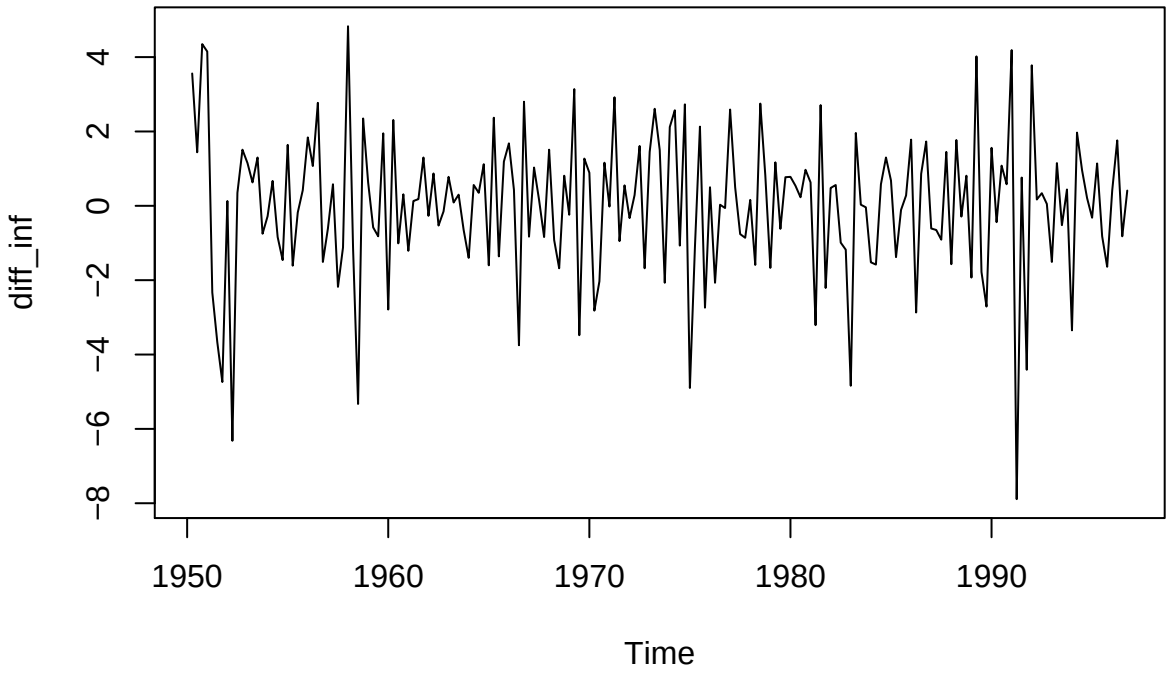

```
#Check out inflation
inflation=Tbrate[,3]
plot(inflation)
```



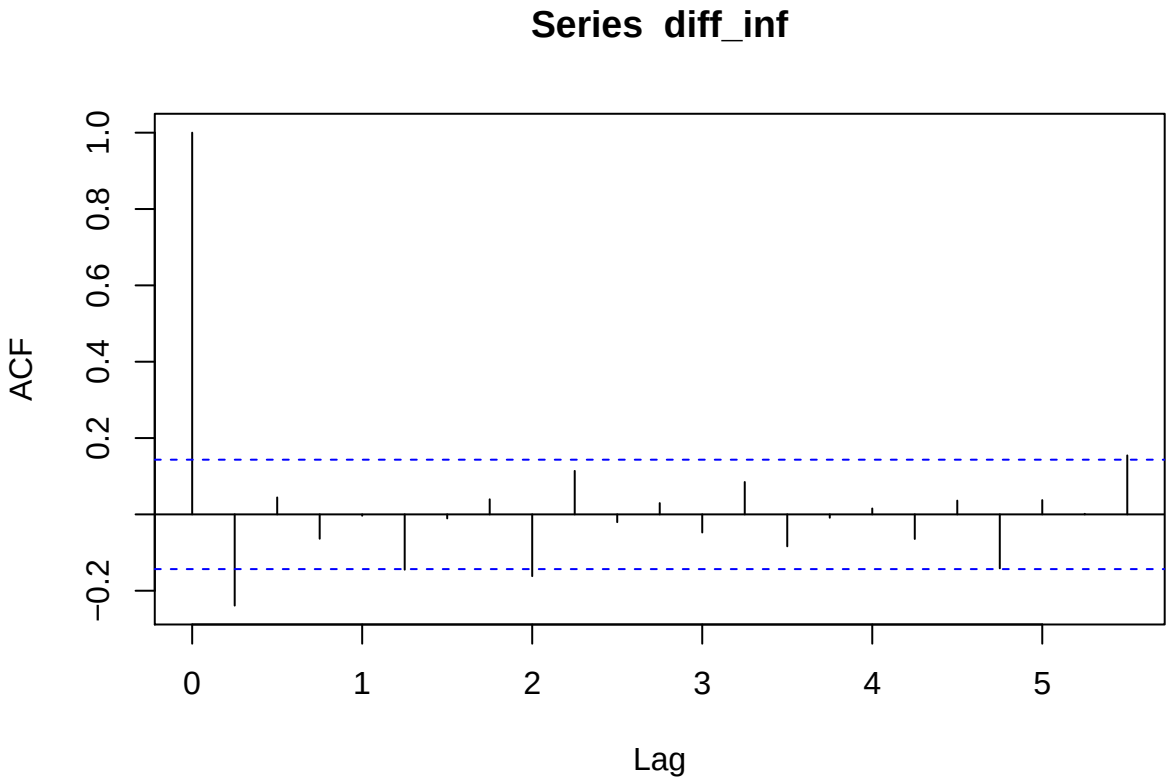

```
acf(inflation)
```



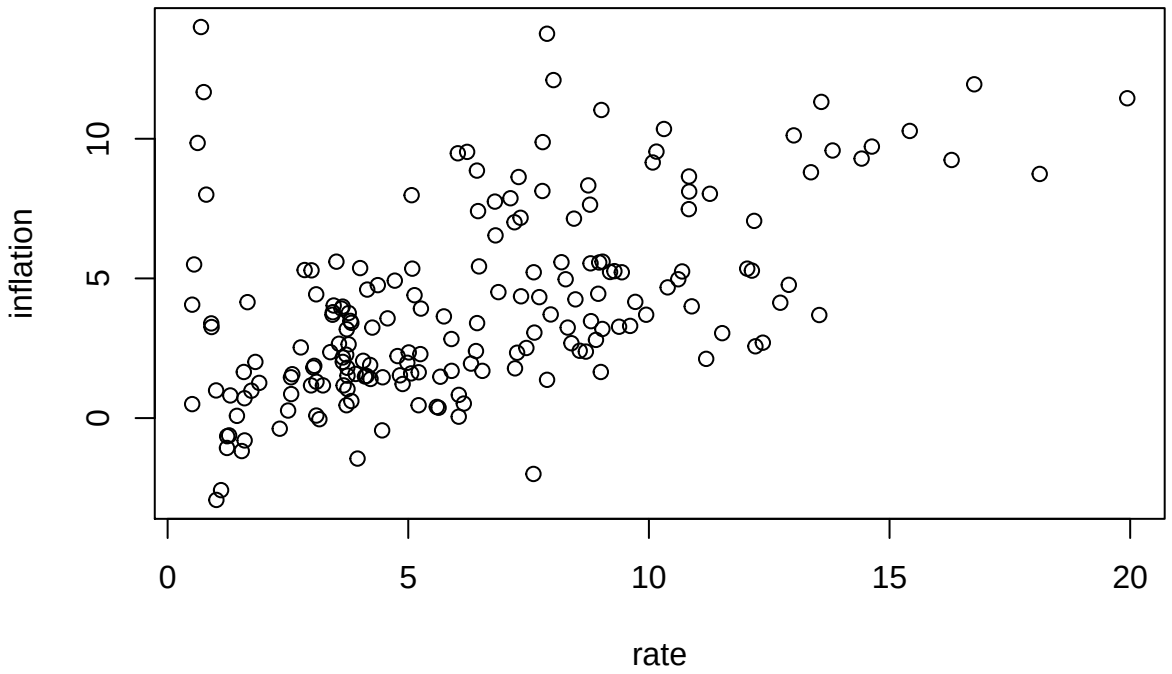

```
#Try differences
diff_inf=diff(inflation)
plot(diff_inf)
```




```
acf(diff_inf)
```




```
plot(rate,inflation)
```




```
#Returns are


#Checking out Skewness
c<-rnorm(1000)
summary(c)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -3.247078 -0.641794 -0.000337 -0.017786  0.651410  2.780435
```

```
basicStats(c)
```

```
##                        c
## nobs        1000.000000
## NAs            0.000000
## Minimum       -3.247078
## Maximum        2.780435
## 1. Quartile   -0.641794
## 3. Quartile    0.651410
## Mean          -0.017786
## Median        -0.000337
## Sum          -17.785594
## SE Mean        0.031358
## LCL Mean      -0.079321
## UCL Mean       0.043750
## Variance       0.983342
## Stdev          0.991636
## Skewness      -0.147074
## Kurtosis      -0.000630
```

```
moment(c,central=T)
```

```
## [1] -2.20813e-17
```

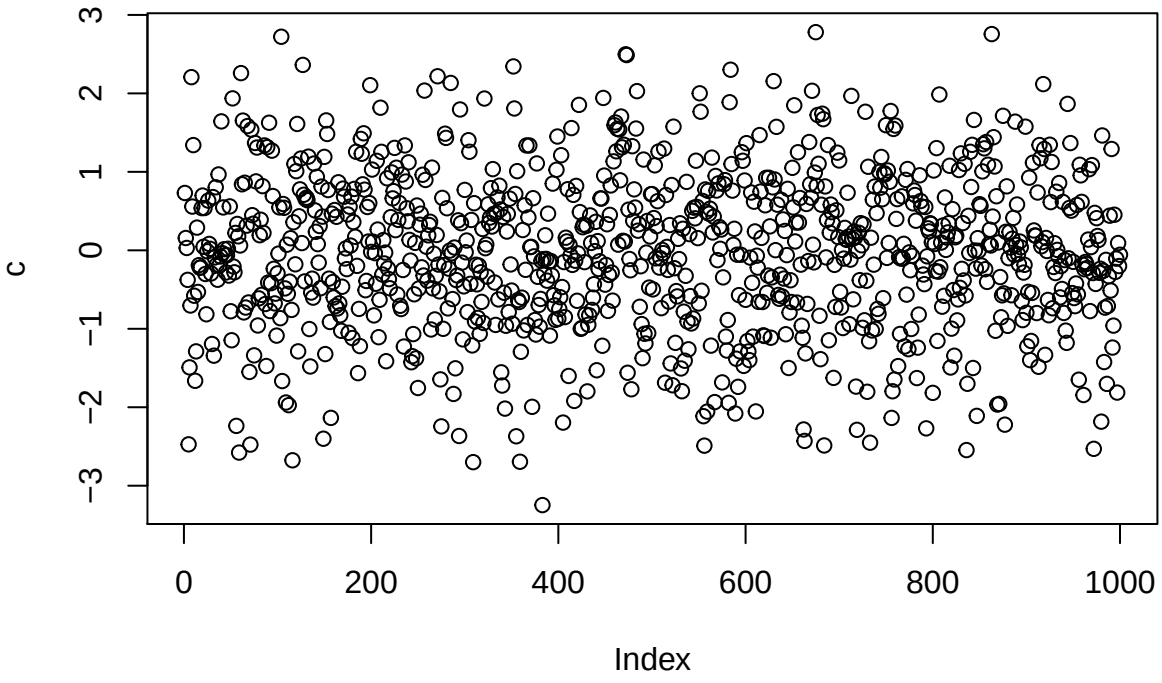
```
skewness(c)
```

```
## [1] -0.1472948
```

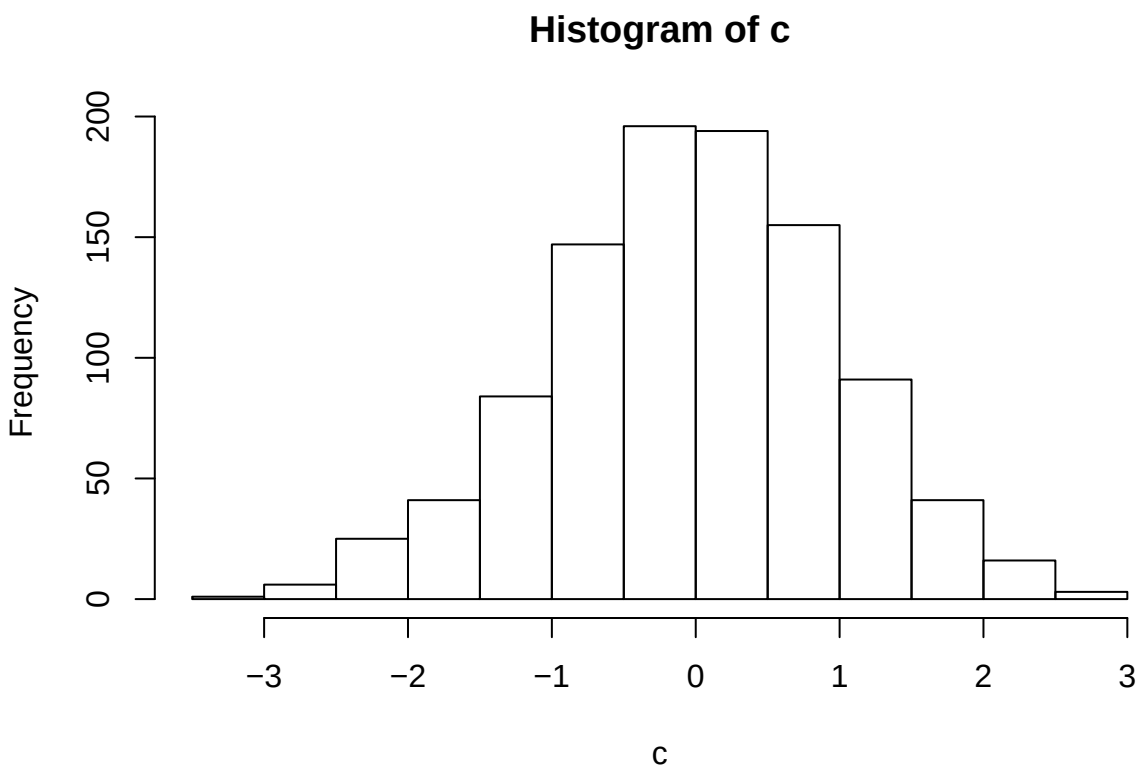
```
kurtosis(c)
```

```
## [1] 3.005378
```

```
plot(c)
```



```r
hist(c)
```




```r
d<-rexp(10000, rate = 1)
summary(d)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000008 0.287919 0.695523 0.994213 1.382277 9.297071
```

```r
basicStats(d)
```

```
##                        d
## nobs        10000.000000
## NAs             0.000000
## Minimum         0.000008
## Maximum         9.297071
## 1. Quartile     0.287919
## 3. Quartile     1.382277
## Mean            0.994213
## Median          0.695523
## Sum          9942.126016
## SE Mean         0.009918
## LCL Mean        0.974771
## UCL Mean        1.013654
## Variance        0.983666
## Stdev           0.991799
## Skewness        2.002462
## Kurtosis        5.983486
```

```r
moment(d,central=T)
```

```
## [1] 3.081979e-17
```

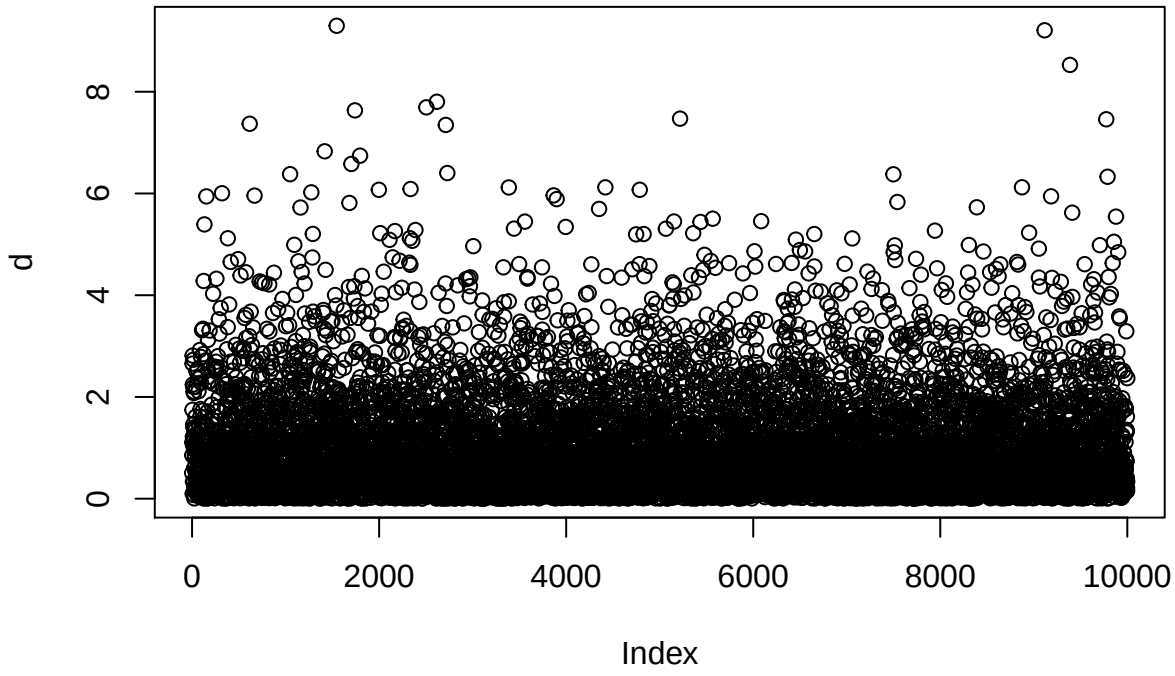
```r
skewness(d)
```

```
## [1] 2.002762
```

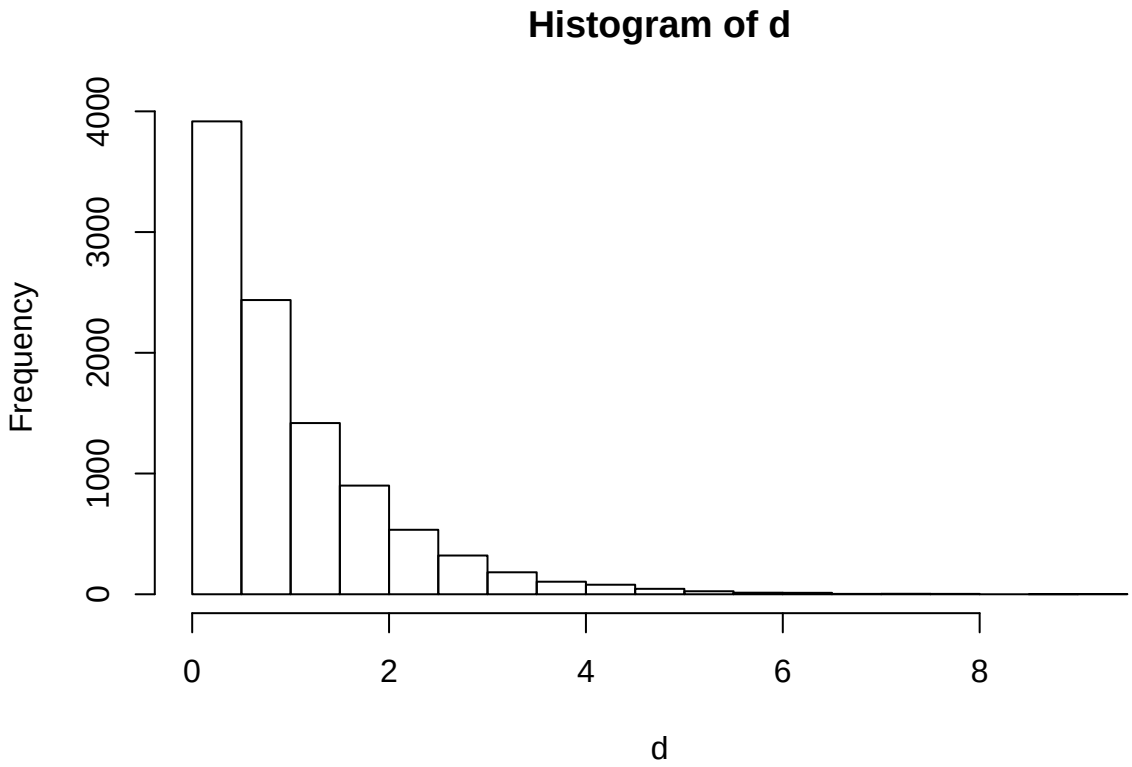
```r
kurtosis(d)
```

```
## [1] 8.985283
```

```r
plot(d)
```




```r
hist(d)
```




```
mean
```

```
## function (x, ...)
## UseMethod("mean")
## <bytecode: 0x7fb0fcf0bb68>
## <environment: namespace:base>
```