4290 Project

Wenhang Bao,wb2304

April 21, 2016

1.Data

```
ADBE=read.csv("~/Downloads/4290/Adobe Systems Incorporated (ADBE).csv",header=T)[,5]
AMG=read.csv("~/Downloads/4290/Affiliated Managers Group Inc. (AMG).csv",header=T)[,5]
GOOG=read.csv("~/Downloads/4290/Alphabet Inc. (GOOG).csv",header=T)[,5]
AMZN=read.csv("~/Downloads/4290/Amazon.com, Inc. (AMZN).csv",header=T)[,5]
BRKA=read.csv("~/Downloads/4290/Berkshire Hathaway Inc. (BRK-A).csv",header=T)[,5]
EA=read.csv("~/Downloads/4290/Electronic Arts Inc. (EA).csv",header=T)[,5]
FSLR=read.csv("~/Downloads/4290/First Solar, Inc. (FSLR).csv",header=T)[,5]
SINA=read.csv("~/Downloads/4290/SINA Corporation (SINA).csv",header=T)[,5]
URBN=read.csv("~/Downloads/4290/Urban Outfitters Inc. (URBN).csv",header=T)[,5]
YHOO=read.csv("~/Downloads/4290/Yahoo! Inc. (YHOO).csv",header=T)[,5]
SP=read.csv("~/Downloads/4290/table.csv",header=T)[,5][1:114]

dat=cbind(ADBE[1:114],AMG[1:114],GOOG[1:114],AMZN[1:114],BRKA[1:114],EA[1:114],FSLR[1:114],SINA[1:114],
colnames(dat)=c("ADBE","AMG","GOOG","AMZN","BRKA","EA","FSLR","SINA","URBN","YHOO","SP")
temp=matrix(rep(0,114*11),ncol=11,nrow=114)
for (i in 1:114){
    temp[i,]=dat[115-i,]
}

data=ts(temp,frequency=12,start=c(2006,11)) # Dataset
colnames(data)=c("ADBE","AMG","GOOG","AMZN","BRKA","EA","FSLR","SINA","URBN","YHOO","SP")
```

2.Descriptive Statistics

```
library(ggplot2)
Mean=apply(data,2,mean);Mean
```

```
##         ADBE          AMG         GOOG         AMZN         BRKA
##     45.61123    124.94596    599.01265    223.12860 142631.79825
##           EA         FSLR         SINA         URBN         YHOO
##     33.19377     96.02202     51.20684     31.02193     24.35921
##           SP
##   1472.74429
```

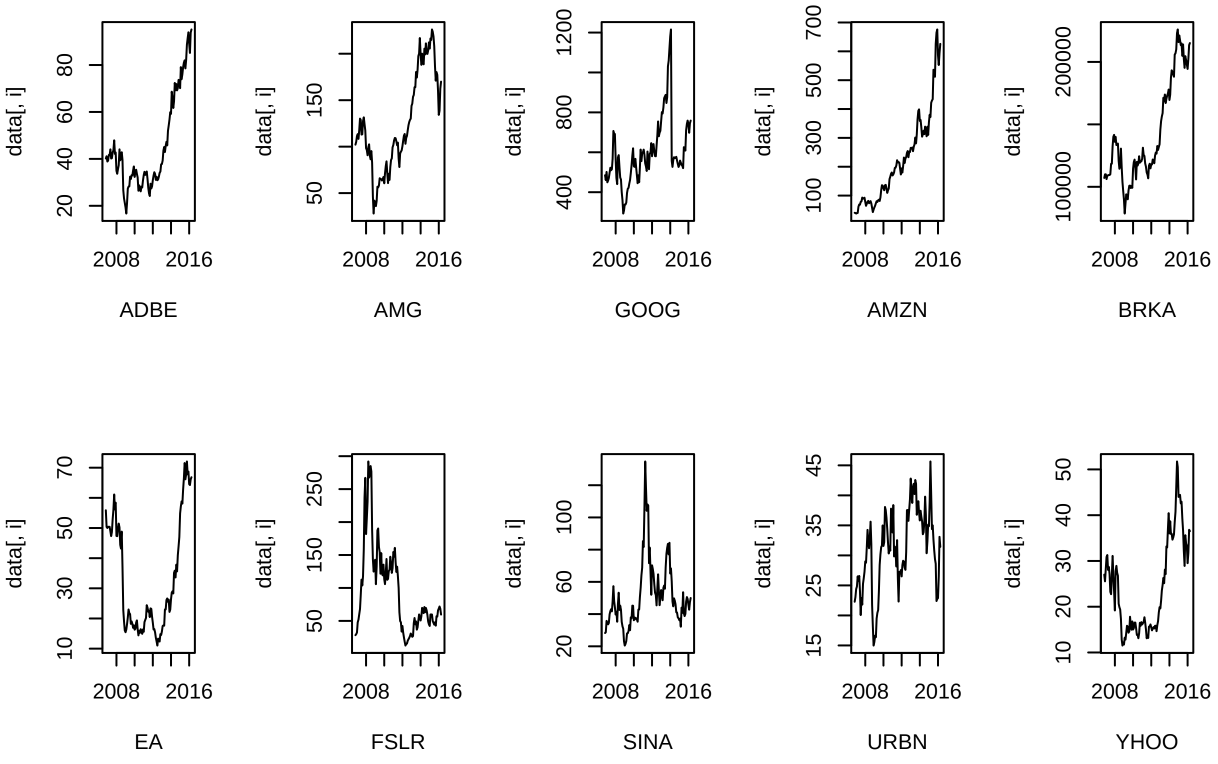
```
Var=cov(data);Var
```

```
##              ADBE          AMG         GOOG         AMZN          BRKA
## ADBE    399.06186     864.1005   1243.12442    2669.6240     735242.63
## AMG     864.10052    2747.7032   4946.76515    6080.3477    1917798.52
## GOOG   1243.12442    4946.7652  28070.83163   14234.5654    2986395.75
## AMZN   2669.62403    6080.3477  14234.56535   24510.9121    5325366.11
## BRKA 735242.63294 1917798.5220 2986395.74598 5325366.1067 1615465068.87
## EA      270.57190     461.8192      2.32371    1206.3001     409855.06
## FSLR   -430.96883   -1738.1141   -4185.67117   -5269.5494   -1024847.99
## SINA    -32.98610     176.4529    1512.50131     614.9052      24765.62
## URBN     35.40903     190.8951     596.98557     351.9614     129460.18
## YHOO    174.82613     474.6476     599.02501     961.6031     355192.47
## SP     6790.33109   18366.0065   30406.67590   46690.5413   13892330.87
##                EA          FSLR        SINA         URBN         YHOO
## ADBE    270.57190 -4.309688e+02   -32.98610     35.40903    174.82613
## AMG     461.81917 -1.738114e+03   176.45288    190.89507    474.64757
## GOOG      2.32371 -4.185671e+03  1512.50131    596.98557    599.02501
## AMZN   1206.30012 -5.269549e+03   614.90524    351.96141    961.60308
## BRKA 409855.06246 -1.024848e+06 24765.61916 129460.17740 355192.46971
## EA      339.31005  7.082138e+01  -100.19081    -16.06686    126.84306
## FSLR     70.82138  4.462464e+03  -131.34368    -96.60562   -204.34513
## SINA   -100.19081 -1.313437e+02   429.85736     39.67397    -22.27717
## URBN    -16.06686 -9.660562e+01    39.67397     44.02365     22.53098
## YHOO    126.84306 -2.043451e+02   -22.27717     22.53098    103.96512
## SP     4366.57019 -1.118021e+04   739.92399   1044.56058   3369.22794
##                SP
## ADBE     6790.331
## AMG     18366.007
## GOOG    30406.676
## AMZN    46690.541
## BRKA 13892330.871
## EA       4366.570
## FSLR   -11180.213
## SINA      739.924
## URBN     1044.561
## YHOO     3369.228
## SP     134550.803
```

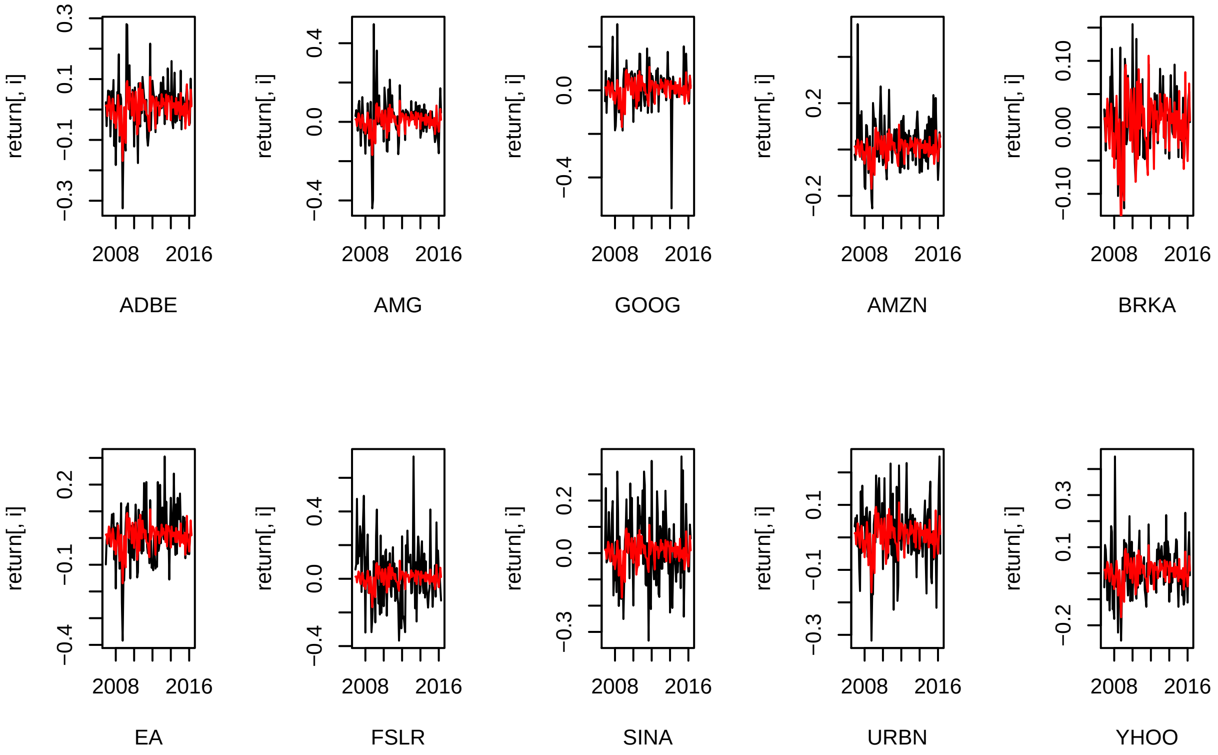
```
SD=sqrt(diag(Var));SD
```

```
##         ADBE          AMG         GOOG         AMZN         BRKA
##    19.976533    52.418538   167.543522   156.559612 40192.848479
##           EA         FSLR         SINA         URBN         YHOO
##    18.420371    66.801680    20.733002     6.635032    10.196329
##           SP
##   366.811673
```

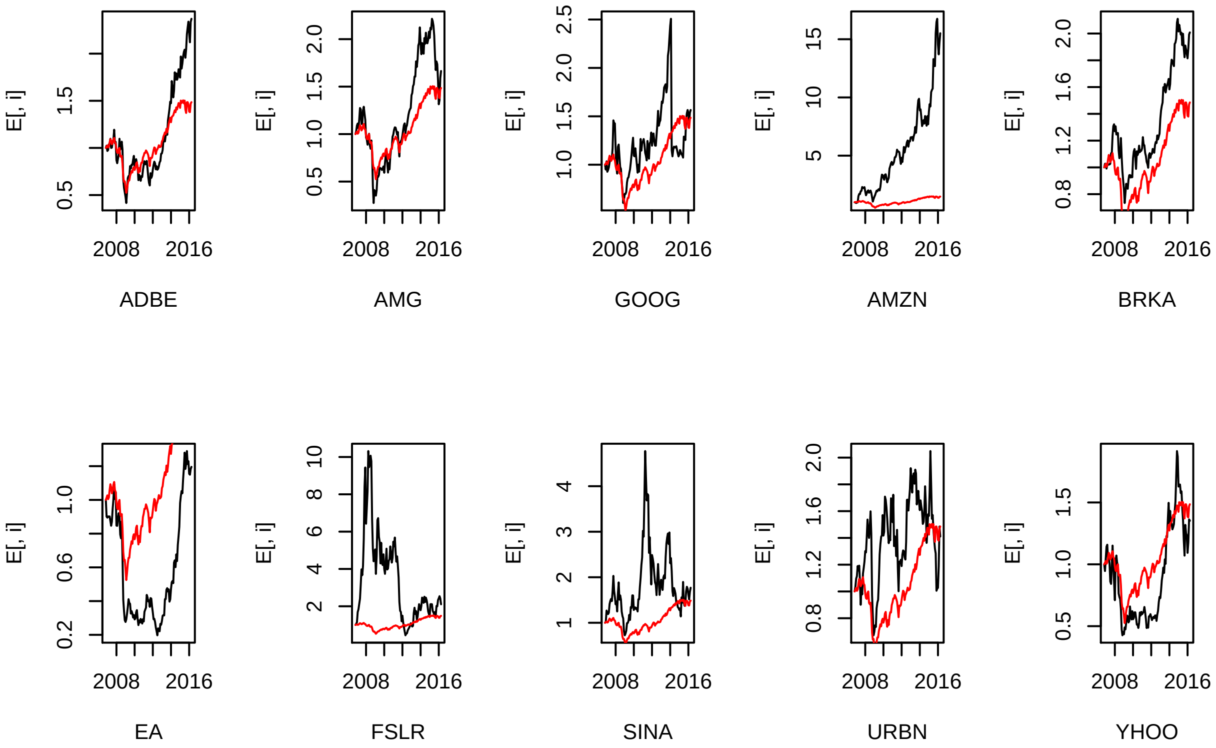
```
library(timeDate)


par(mfrow=c(2,5))
#Price
for (i in 1:10){
    plot(data[,i],xlab=colnames(data)[i])
}
```



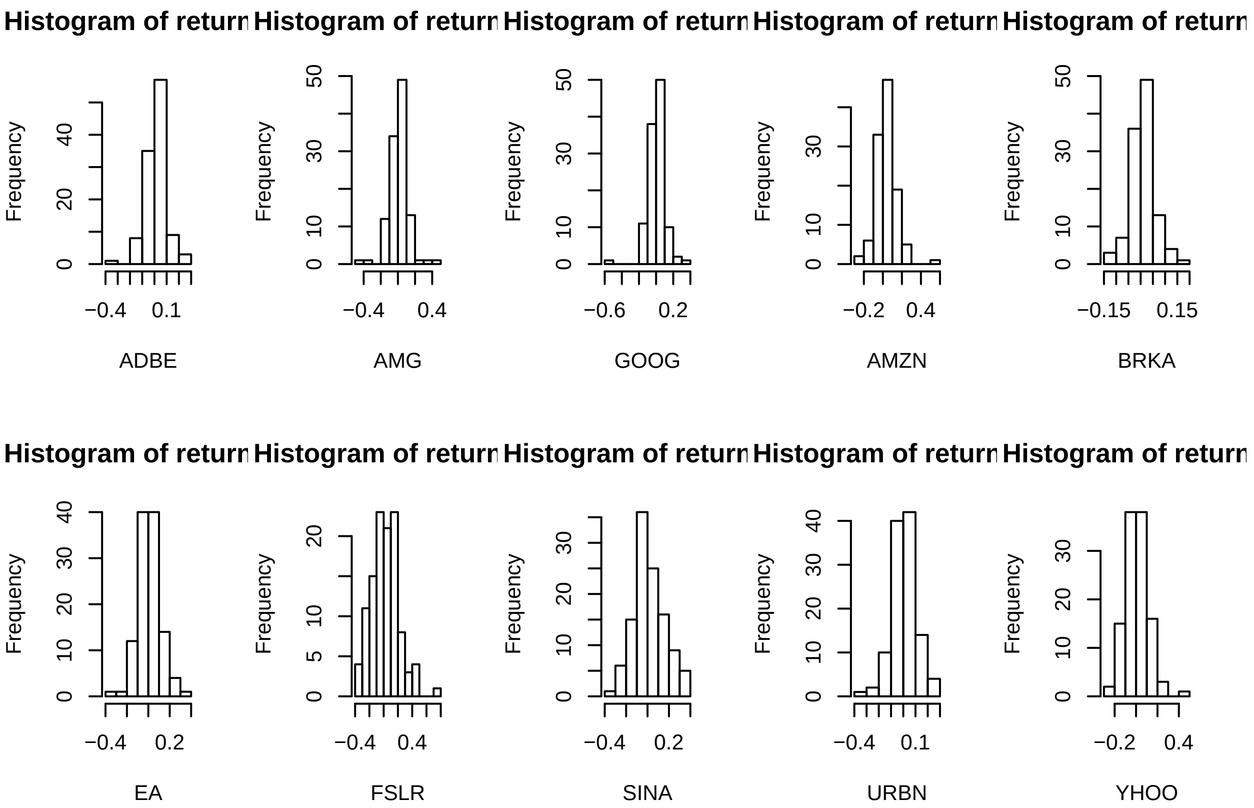
```r
#Return
lag.data=lag(data,1)
options(digit=3)
return=(data[2:114,]-lag.data[1:113,])/lag.data[1:113,]
return=ts(return,frequency=12,start=c(2006,12))
par(mfrow=c(2,5))

for (i in 1:10){
    plot(return[,i],type="l",xlab=colnames(data)[i])
  lines(return[,11],col="red")
}
```



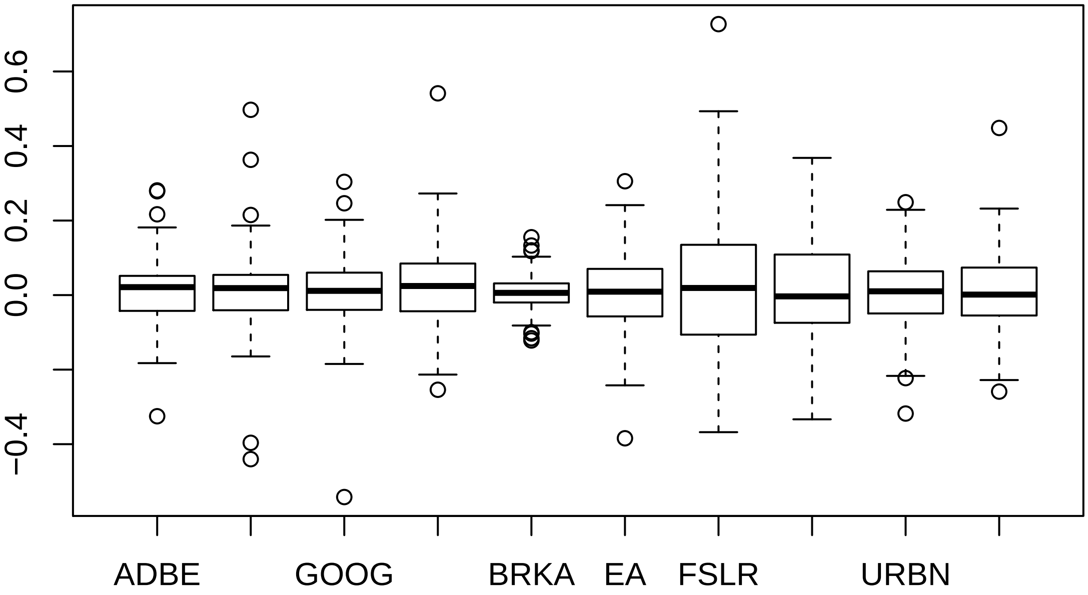
```
#Equity Curve
E=matrix(rep(0,114*11),ncol=11,nrow=114)
E[1,]=c(rep(1,11))
i=2
for(i in 2:114){
    E[i,]=E[i-1,]+E[i-1,]*return[i-1,]
}
E=ts(E,frequency=12,start=c(2006,11))
par(mfrow=c(2,5))
for(i in 1:10){
    plot(E[,i],type="l",xlab=colnames(data)[i])
  lines(E[,11],col="red")
}
```



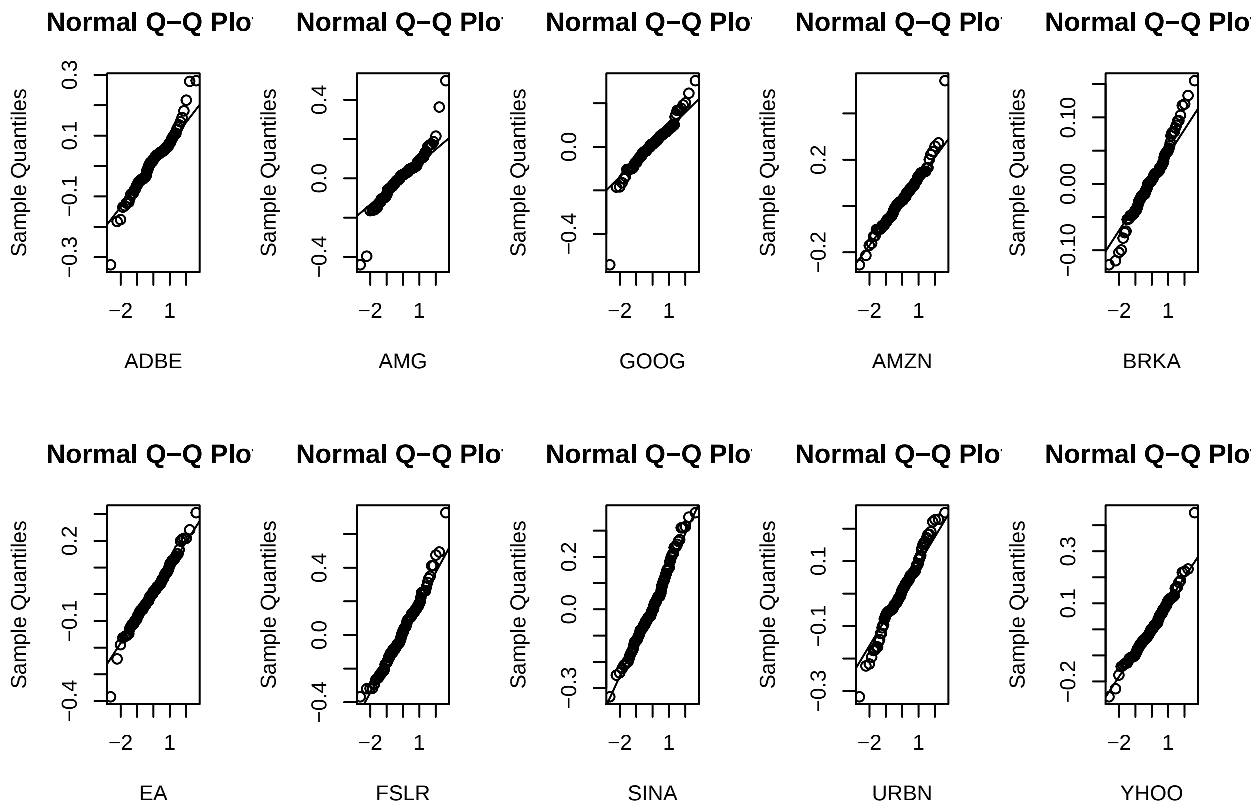
```r
#Histogram
par(mfrow=c(2,5))
for (i in 1:10) {
    hist(return[,i],xlab=colnames(data)[i])
}
```



```
#boxpolt
par(mfrow=c(1,1))
boxplot(return[,1:10])
```



```
#qq-plot
par(mfrow=c(2,5))
for (i in 1:10) {
    qqnorm(y=return[,i],xlab=colnames(data)[i])
    qqline(return[,i])
}
```



```
#Sharpe ratio
risk.free=0.03/12
mean.return=apply(return,2,mean)
var.return=apply(return,2,var)
sd.return=sqrt(var.return)
cov.return=cov(return)
Sharpe=(mean.return[1:10]*12-risk.free*12)/(sd.return[1:10]*sqrt(12));Sharpe
```

```
##      ADBE       AMG      GOOG      AMZN      BRKA        EA      FSLR
## 0.3570450 0.2643414 0.2474320 0.8861977 0.3398916 0.1527004 0.3896592
##      SINA      URBN      YHOO
## 0.3015697 0.1966767 0.1763576
```

```
which.max(Sharpe)
```

```
## AMZN
##    4
```

```
#Beta
beta=matrix(nrow=10)
for(i in 1:10){
  beta[i]=summary(lm(I(return[,i]-risk.free)~I(return[,11]-risk.free)))$coefficients[2,1]
}
beta
```

```
##             [,1]
##  [1,] 1.4653268
##  [2,] 1.9250886
##  [3,] 1.0681635
##  [4,] 1.0905255
##  [5,] 0.5564552
##  [6,] 1.1304666
##  [7,] 1.5287909
##  [8,] 1.4085790
##  [9,] 0.9307040
## [10,] 1.0897323
```

```
which.max(beta)
```

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

```
#Annual numbers
mean.return.annual=12*mean.return;mean.return.annual
```

```
##       ADBE        AMG       GOOG       AMZN       BRKA         EA 
## 0.13743677 0.13311813 0.11546700 0.35901027 0.08912898 0.08486105 
##       FSLR       SINA       URBN       YHOO         SP 
## 0.28942796 0.17854674 0.09931056 0.09278241 0.05450718
```

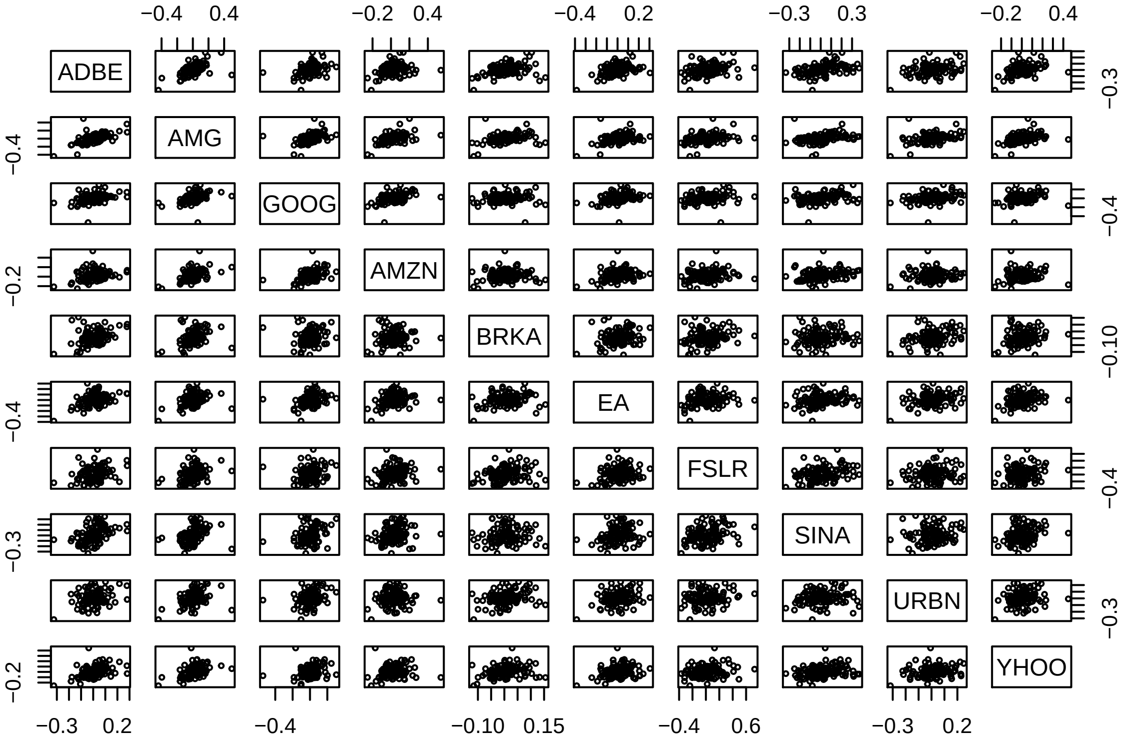
```
sd.return.annual=sqrt(12)*sd.return;sd.return.annual
```

```
##      ADBE       AMG      GOOG      AMZN      BRKA        EA      FSLR 
## 0.3009054 0.3900945 0.3454161 0.3712606 0.1739642 0.3592724 0.6657816 
##      SINA      URBN      YHOO        SP 
## 0.4925785 0.3524086 0.3559948 0.1566488
```

```
(mean.return.annual[1:10]-risk.free*12)/sd.return.annual[1:10]
```

```
##      ADBE       AMG      GOOG      AMZN      BRKA        EA      FSLR 
## 0.3570450 0.2643414 0.2474320 0.8861977 0.3398916 0.1527004 0.3896592 
##      SINA      URBN      YHOO 
## 0.3015697 0.1966767 0.1763576
```

```
par(mfrow=c(1,1))
stocks=return[,1:10]
pairs(data.frame(stocks),cex=0.5)
```



```
#Sample Covariance Matrix
cor(return[,1:10])
```

```
##           ADBE       AMG       GOOG       AMZN       BRKA        EA
## ADBE 1.0000000 0.5796468 0.37822585 0.32180540 0.35668248 0.4544962
## AMG  0.5796468 1.0000000 0.41632624 0.49033170 0.33750089 0.3767973
## GOOG 0.3782259 0.4163262 1.00000000 0.43634739 0.09885975 0.2992661
## AMZN 0.3218054 0.4903317 0.43634739 1.00000000 0.07541115 0.3021106
## BRKA 0.3566825 0.3375009 0.09885975 0.07541115 1.00000000 0.2300454
## EA   0.4544962 0.3767973 0.29926608 0.30211061 0.23004538 1.0000000
## FSLR 0.2890877 0.3184114 0.18257925 0.25920779 0.19926839 0.2421080
## SINA 0.4077213 0.3113059 0.30296479 0.26731875 0.07735569 0.2094385
## URBN 0.3138782 0.3611083 0.29292693 0.11792968 0.30167710 0.3041197
## YHOO 0.4090196 0.4424258 0.25201862 0.13033119 0.24904753 0.3337269
##           FSLR       SINA      URBN      YHOO
## ADBE 0.2890877 0.40772131 0.3138782 0.4090196
## AMG  0.3184114 0.31130589 0.3611083 0.4424258
## GOOG 0.1825793 0.30296479 0.2929269 0.2520186
## AMZN 0.2592078 0.26731875 0.1179297 0.1303312
## BRKA 0.1992684 0.07735569 0.3016771 0.2490475
## EA   0.2421080 0.20943854 0.3041197 0.3337269
## FSLR 1.0000000 0.32366645 0.1628847 0.2167611
## SINA 0.3236664 1.00000000 0.1624596 0.2516614
## URBN 0.1628847 0.16245961 1.0000000 0.2267494
## YHOO 0.2167611 0.25166139 0.2267494 1.0000000
```

```
#Stationary Test
library("tseries")
tt=list()
```

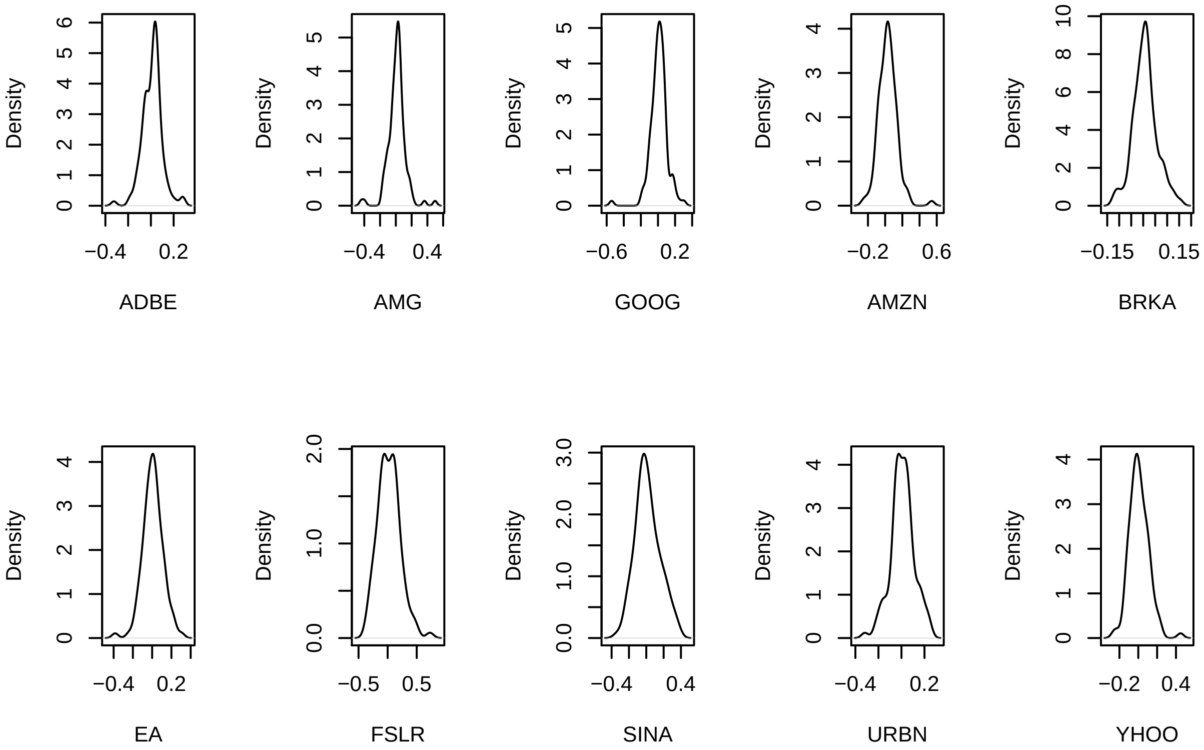
```
p_value=c()
for (i in 1:10){
  tt[[i]]=adf.test(return[,i])
  p_value[i]=tt[[i]]$p.value
}
```

```
## Warning in adf.test(return[, i]): p-value smaller than printed p-value

## Warning in adf.test(return[, i]): p-value smaller than printed p-value

## Warning in adf.test(return[, i]): p-value smaller than printed p-value

## Warning in adf.test(return[, i]): p-value smaller than printed p-value

## Warning in adf.test(return[, i]): p-value smaller than printed p-value

## Warning in adf.test(return[, i]): p-value smaller than printed p-value

## Warning in adf.test(return[, i]): p-value smaller than printed p-value

## Warning in adf.test(return[, i]): p-value smaller than printed p-value

## Warning in adf.test(return[, i]): p-value smaller than printed p-value
```

```
t(p_value)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]       [,7] [,8] [,9] [,10]
## [1,] 0.01 0.01 0.01 0.01 0.01 0.01 0.03990603 0.01 0.01  0.01
```

```
### Distribution of the stocks
par(mfrow=c(2,5))
for (i in 1:10){
  plot(density(return[,i]),main="",xlab=colnames(dat)[i],lty=1)
}
```



```
###Fit normal Distribution
i=8
for (i in 1:10){
  print(ks.test(return[,i],"pnorm",mean=mean(return[,i]
  ),sd=sd(return[,i]))$p.value
  )
}
```

```
## [1] 0.2951703
## [1] 0.07894488
## [1] 0.2991341
## [1] 0.7559177
## [1] 0.437729
## [1] 0.9420083
## [1] 0.8215911
## [1] 0.531717
## [1] 0.4257704
## [1] 0.5895173
```

3.Portfolio Theory

```
library(Ecfun)
library(quadprog)

s.mean=apply(stocks,2,mean)
s.cov=cov(stocks)
s.sd=sqrt(diag(s.cov))
Amat=cbind(rep(1,10),s.mean)
muP=seq(0,0.04,length=1000)
```

```
sdP=muP
weights=matrix(0,nrow=1000,ncol=10)
for(i in 1:length(muP)){
    bvec=c(1,muP[i])
    result=solve.QP(Dmat=2*s.cov,dvec=rep(0,10),Amat=Amat,bvec=bvec,meq=2)
    sdP[i]=sqrt(result$value)
    weights[i,]=result$solution
}
par(mfrow=c(1,1))
plot(sdP,muP,type="l",xlim=c(0,0.15),ylim=c(0,0.05))

#MVP
ind=(sdP==min(sdP))
weights[ind,]
```

```
##  [1]  0.02009877 -0.12476149  0.08953182  0.13295719  0.70839131
##  [6]  0.02865948 -0.01916003  0.03726972  0.04808988  0.07892336
```

```
points(sdP[ind],muP[ind],cex=2,pch="+")
VaR.mvp=-100000*(muP[ind]+sdP[ind]*qnorm(0.05))
ES.mvp=100000*(-muP[ind]+sdP[ind]*(dnorm(qnorm(0.05))/0.05))

#Annually
muP[ind]*12
```

```
## [1] 0.123003
```

```
sdP[ind]*sqrt(12)
```

```
## [1] 0.1498811
```

```
#VaR
VaR=vector()
for (i in 1:10) {
  VaR[i] = - 100000*(s.mean[i]+s.sd[i]*qnorm(0.05))
}
portfolio_var = -100000*(muP[ind]+sdP[ind]*qnorm(0.05))

#Tangency Portfolio
points(0,risk.free,cex=4,pch="*")
sharpe=(muP-risk.free)/sdP
ind2=(sharpe==max(sharpe))
points(sdP[ind2],muP[ind2],cex=2,pch="+")
weights[ind2,]
```

```
##  [1]  0.14891235 -0.32435305 -0.12667396  0.72565795  0.44215923
##  [6] -0.16038675  0.06020728  0.02694427  0.10721882  0.10031387
```

```
muP[ind2]#expected return
```

```
## [1] 0.02426426
```

```
sdP[ind2]#expected standard deviation
```

```
## [1] 0.07252162
```

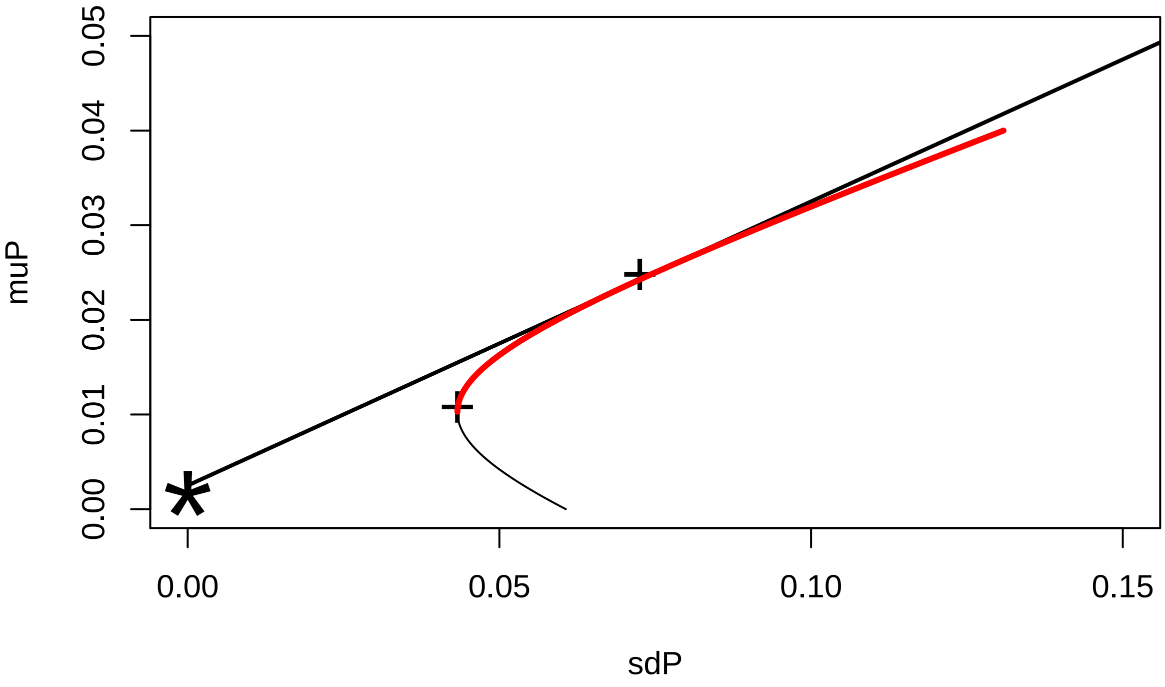
```
sdP[ind2]^2#expected variance
```

```
## [1] 0.005259385
```

```
lines(c(0,2),risk.free+c(0,2)*(muP[ind2]-risk.free)/sdP[ind2],lwd=2,lty=1)
sdP[ind2]
```

```
## [1] 0.07252162
```

```
VaR.t=-100000*(muP[ind2]+sdP[ind2]*qnorm(0.05))
ES.t=100000*(-muP[ind2]+sdP[ind2]*(dnorm(qnorm(0.05))/0.05))
#Frontier
ind3=(muP>muP[ind])
lines(sdP[ind3],muP[ind3],type="l",lwd=3,col="red")
```



```
#Short sale not allowed
muP.cons = seq(min(s.mean) + 0.0001, max(s.mean) - 0.0001,
length = 1000)
sdP.cons=muP.cons
Amat.cons=cbind(rep(1,10),s.mean,diag(1,nrow=10))
weights.cons=matrix(0,nrow=1000,ncol=10)
for(i in 1:length(muP)){
    bvec=c(1,muP.cons[i],rep(0,10))
    result=solve.QP(Dmat=2*s.cov,dvec=rep(0,10),Amat=Amat.cons,bvec=bvec,meq=2)
    sdP.cons[i]=sqrt(result$value)
    weights.cons[i,]=result$solution
}
par(mfrow=c(1,1))
plot(sdP.cons,muP.cons,type="l",xlim=c(0,0.15),ylim=c(0,0.05))

#MVP
ind.cons=(sdP.cons==min(sdP.cons))
weights.cons[ind.cons,]
```

```
##  [1] -1.403486e-18  2.295314e-17  8.367751e-02  8.866607e-02  7.006009e-01
##  [6]  2.325326e-02  2.549194e-21  2.832799e-02  2.877367e-02  4.670057e-02
```

```
points(sdP.cons[ind.cons],muP.cons[ind.cons],cex=2,pch="+")
VaR.norm=-100000*(s.mean+s.sd*qnorm(0.05))
ES.nss.mvp=100000*(-muP.cons[ind.cons]+sdP.cons[ind.cons]*(dnorm(qnorm(0.05))/0.05))
#Annually
muP.cons[ind.cons]*12
```

```
## [1] 0.1181596
```

```
sdP.cons[ind.cons]*sqrt(12)
```

```
## [1] 0.1546092
```

```
#VaR
VaR=vector()
for (i in 1:10) {
  VaR[i] = - 100000*(s.mean[i]+s.sd[i]*qnorm(0.05))
}
portfolio_var = -100000*(muP.cons[ind.cons]+sdP[ind.cons]*qnorm(0.05))

#Tangency Portfolio
points(0,risk.free,cex=4,pch="*")
sharpe.cons=(muP.cons-risk.free)/sdP.cons
ind.cons2=(sharpe.cons==max(sharpe.cons))
points(sdP.cons[ind.cons2],muP.cons[ind.cons2],cex=2,pch="+")
weights.cons[ind.cons2,]
```

```
##  [1] 7.208024e-19 8.733691e-18 6.436264e-18 5.724065e-01 3.712133e-01
##  [6] 7.995447e-17 4.486368e-02 1.150905e-02 7.444153e-06 0.000000e+00
```

```
muP.cons[ind.cons2]#expected return
```

```
## [1] 0.02113551
```

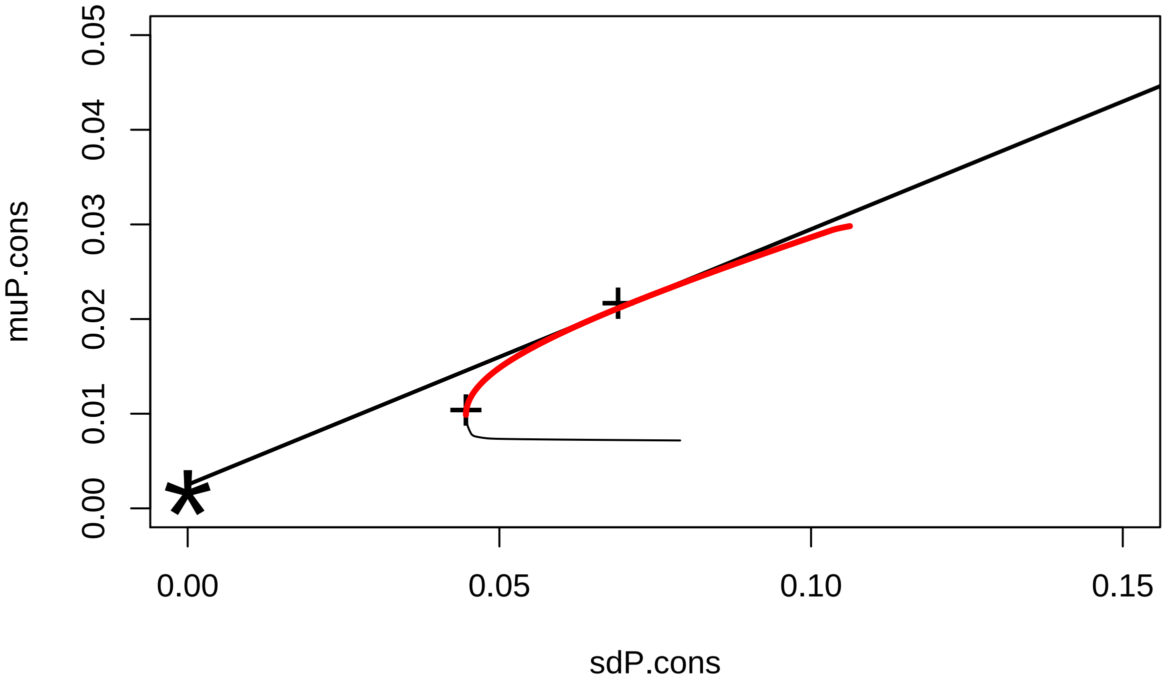
```
sdP.cons[ind.cons2]#expected standard deviation
```

```
## [1] 0.06903854
```

```
sdP.cons[ind.cons2]^2#expected variance
```

```
## [1] 0.00476632
```

```
lines(c(0,2),risk.free+c(0,2)*(muP.cons[ind.cons2]-risk.free)/sdP.cons[ind.cons2],lwd=2,lty=1)
VaR.t.cons=-100000*(muP.cons[ind.cons2]+sdP.cons[ind.cons2]*qnorm(0.05))
ES.t.cons=100000*(-muP.cons[ind.cons2]+sdP.cons[ind.cons2]*(dnorm(qnorm(0.05))/0.05))

#Frontier
ind.cons3=(muP.cons>muP.cons[ind.cons])
lines(sdP.cons[ind.cons3],muP.cons[ind.cons3],type="l",lwd=3,col="red")
```



4.Asset Allocation

```
#Only risky assets
muP[which.min(abs(muP-0.005))]
```

```
## [1] 0.005005005
```

```
sdP[which.min(abs(muP-0.005))]
```

```
## [1] 0.04841895
```

```
100000*weights[which.min(abs(muP-0.005)),]
```

```
##  [1] -2811.431 -5005.722 17045.455 -8888.223 80803.818  9941.678 -4886.609
##  [8]  4113.439  2595.876  7091.720
```

```
#VaR and Shortfall
VaR.1=-100000*(muP[which.min(abs(muP-0.005))]+sdP[which.min(abs(muP-0.005))]*qnorm(0.05))
shortfall=100000*(-muP[which.min(abs(muP-0.005))]+sdP[which.min(abs(muP-0.005))]*(dnorm(qnorm(p = 0.05)


# Incorporate with T-bills
# ind2=tangency portfolio
muP[ind2]
```

```
## [1] 0.02426426
```

```
sdP[ind2]
```

```
## [1] 0.07252162
```

```
sharpe[ind2]
```

```
## [1] 0.3001073
```

```
expected.risk=(0.005-risk.free)/sharpe[ind2]
w=expected.risk/sdP[ind2]  #weights on risky assets
w*weights[ind2] #weights on each stocks
```

```
##  [1]  0.017105144 -0.037257525 -0.014550682  0.083354293  0.050789591
##  [6] -0.018423176  0.006915841  0.003095013  0.012315925  0.011522773
```

```
100000*w*weights[ind2]
```

```
##  [1]  1710.5144 -3725.7525 -1455.0682  8335.4293  5078.9591 -1842.3176
##  [7]   691.5841   309.5013  1231.5925  1152.2773
```

```
VaR.2=-100000*(0.005+sdP[ind2]*w*qnorm(0.05))
shortfall.1=100000*(-muP[ind2]+sdP[ind2]*(dnorm(qnorm(p = 0.05))/0.05))


#No short sale
#Only risky assets
muP.cons[which.min(abs(muP.cons-0.005))]
```

```
## [1] 0.007171754
```

```
sdP.cons[which.min(abs(muP.cons-0.005))]
```

```
## [1] 0.0790048
```

```
100000*weights[which.min(abs(muP.cons-0.005)),]
```

```
##  [1]  -7411.9161   2122.5476  24767.0897 -30056.1070  90312.1062
##  [6]  16693.3289  -7721.1554   4482.2044    484.1287   6327.7730
```

```
#VaR and Shortfall
VaR.cons=-100000*(muP.cons[which.min(abs(muP.cons-0.005))]+sdP.cons[which.min(abs(muP.cons-0.005))]*qno
shortfall.cons=100000*(-muP.cons[which.min(abs(muP.cons-0.005))]+sdP.cons[which.min(abs(muP.cons-0.005))


# Incorporate with T-bills
# ind2=tangency portfolio
muP.cons[ind.cons2]
```

```
## [1] 0.02113551
```

```
sdP.cons[ind.cons2]
```

```
## [1] 0.06903854
```

```
sharpe[ind.cons2]
```

```
## [1] 0.3000823
```

```
expected.risk2=(0.005-risk.free)/sharpe.cons[ind.cons2]
w.cons=expected.risk2/sdP[ind.cons2]  #weights on risky assets
w.cons*weights.cons[ind.cons2] #weights on each stocks
```

```
##  [1] 9.038278e-20 1.095134e-18 8.070553e-19 7.177514e-02 4.654714e-02
##  [6] 1.002564e-17 5.625542e-03 1.443141e-03 9.334364e-07 0.000000e+00
```

```
100000*w.cons*weights.cons[ind.cons2]
```

```
##  [1] 9.038278e-15 1.095134e-13 8.070553e-14 7.177514e+03 4.654714e+03
##  [6] 1.002564e-12 5.625542e+02 1.443141e+02 9.334364e-02 0.000000e+00
```

```
VaR.cons2=-100000*(0.005+sdP.cons[ind.cons2]*w.cons*qnorm(0.05))
shortfall.cons2=100000*(-muP.cons[ind.cons2]+sdP.cons[ind.cons2]*(dnorm(qnorm(p = 0.05))/0.05))
```

5.PCA

```
cor(stocks)
```

```
##           ADBE       AMG       GOOG       AMZN       BRKA        EA
## ADBE 1.0000000 0.5796468 0.37822585 0.32180540 0.35668248 0.4544962
## AMG  0.5796468 1.0000000 0.41632624 0.49033170 0.33750089 0.3767973
## GOOG 0.3782259 0.4163262 1.00000000 0.43634739 0.09885975 0.2992661
## AMZN 0.3218054 0.4903317 0.43634739 1.00000000 0.07541115 0.3021106
## BRKA 0.3566825 0.3375009 0.09885975 0.07541115 1.00000000 0.2300454
## EA   0.4544962 0.3767973 0.29926608 0.30211061 0.23004538 1.0000000
## FSLR 0.2890877 0.3184114 0.18257925 0.25920779 0.19926839 0.2421080
## SINA 0.4077213 0.3113059 0.30296479 0.26731875 0.07735569 0.2094385
## URBN 0.3138782 0.3611083 0.29292693 0.11792968 0.30167710 0.3041197
## YHOO 0.4090196 0.4424258 0.25201862 0.13033119 0.24904753 0.3337269
##           FSLR       SINA      URBN      YHOO
## ADBE 0.2890877 0.40772131 0.3138782 0.4090196
## AMG  0.3184114 0.31130589 0.3611083 0.4424258
## GOOG 0.1825793 0.30296479 0.2929269 0.2520186
## AMZN 0.2592078 0.26731875 0.1179297 0.1303312
## BRKA 0.1992684 0.07735569 0.3016771 0.2490475
## EA   0.2421080 0.20943854 0.3041197 0.3337269
## FSLR 1.0000000 0.32366645 0.1628847 0.2167611
## SINA 0.3236664 1.00000000 0.1624596 0.2516614
## URBN 0.1628847 0.16245961 1.0000000 0.2267494
## YHOO 0.2167611 0.25166139 0.2267494 1.0000000
```

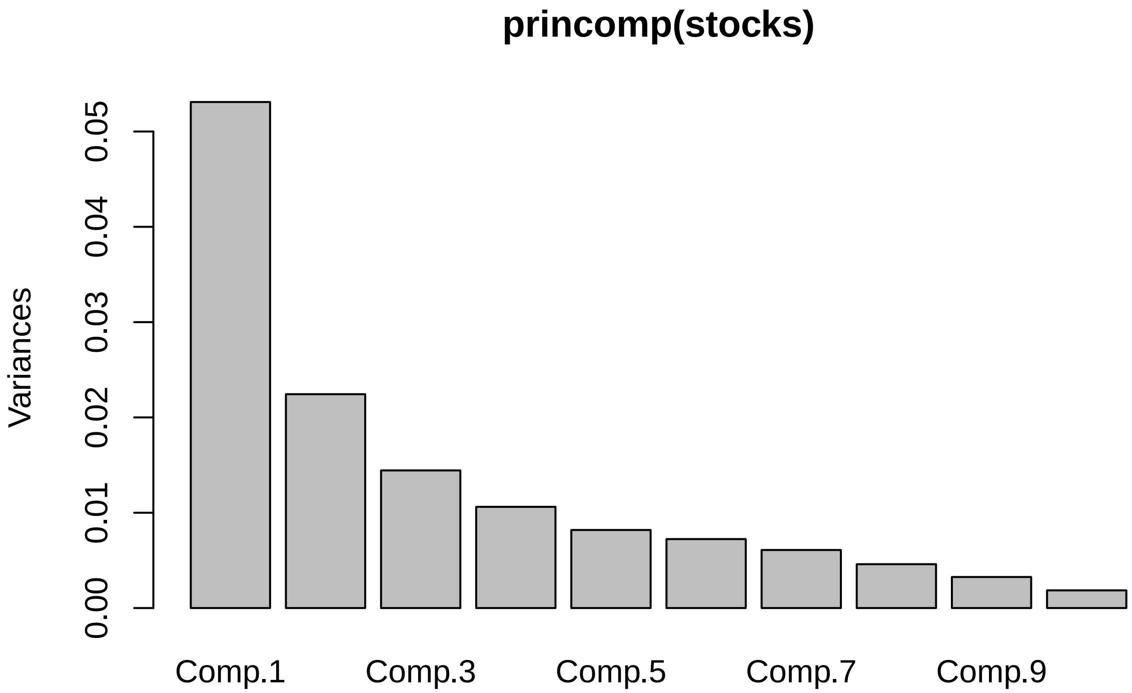
```r
princomp(stocks)
```

```
## Call:
## princomp(x = stocks)
##
## Standard deviations:
##     Comp.1     Comp.2     Comp.3     Comp.4     Comp.5     Comp.6
## 0.23042236 0.14979447 0.12016630 0.10305379 0.09050284 0.08506584
##     Comp.7     Comp.8     Comp.9    Comp.10
## 0.07804557 0.06783246 0.05706891 0.04308412
##
##  10  variables and  113 observations.
```

```r
summary(princomp(stocks))
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     0.2304224 0.1497945 0.1201663 0.10305379 0.09050284
## Proportion of Variance 0.4027639 0.1702130 0.1095385 0.08056181 0.06213348
## Cumulative Proportion  0.4027639 0.5729769 0.6825154 0.76307719 0.82521068
##                            Comp.6     Comp.7     Comp.8     Comp.9
## Standard deviation     0.08506584 0.07804557 0.06783246 0.05706891
## Proportion of Variance 0.05489233 0.04620593 0.03490410 0.02470589
## Cumulative Proportion  0.88010301 0.92630894 0.96121304 0.98591893
##                           Comp.10
## Standard deviation     0.04308412
## Proportion of Variance 0.01408107
## Cumulative Proportion  1.00000000
```

```r
par(mfrow=c(1,1))
plot(princomp(stocks))
```



```r
eig = eigen(cor(stocks))
eig$values
```

```
##  [1] 3.7067725 1.1693780 0.9268825 0.8033509 0.7641764 0.6981122 0.6332749
##  [8] 0.5278858 0.4508496 0.3193172
```

```r
eig$vectors
```

```
##             [,1]         [,2]        [,3]        [,4]        [,5]
##  [1,] -0.4023930 -0.081358190 -0.06063040  0.15445991  0.07616948
##  [2,] -0.4132667  0.002846957  0.10380356 -0.04752721  0.21012983
##  [3,] -0.3192622  0.322043249  0.40376197  0.06607643 -0.22784436
##  [4,] -0.2966481  0.497366415  0.25208466 -0.32425851  0.32895275
##  [5,] -0.2378042 -0.583761623 -0.07730185 -0.39094259  0.17378636
##  [6,] -0.3269007 -0.095063259  0.15167171  0.11466184  0.21019565
##  [7,] -0.2625713  0.113355453 -0.62078521 -0.46951275 -0.03673203
##  [8,] -0.2812085  0.314921193 -0.45637119  0.23774565 -0.45939718
##  [9,] -0.2706814 -0.354376707  0.32891808 -0.17057069 -0.67264564
## [10,] -0.3043959 -0.234233243 -0.16781668  0.62636300  0.22514355
##              [,6]         [,7]        [,8]        [,9]       [,10]
##  [1,]  0.11880284 -0.385854972  0.08500865  0.65597963 -0.44451220
##  [2,]  0.25818828  0.160353214  0.41722552  0.19923829  0.68254422
##  [3,]  0.13667553  0.191100272 -0.69846664  0.14316736  0.10726083
##  [4,]  0.08115028  0.005894982  0.24772030 -0.38364816 -0.41539740
##  [5,]  0.37160787 -0.237327069 -0.35260043 -0.30791682  0.01008810
##  [6,] -0.77272275 -0.352071515 -0.11186181 -0.15847857  0.21232466
##  [7,] -0.31295166  0.394077241 -0.13769778  0.19427984 -0.03316879
##  [8,]  0.18741544 -0.399259174  0.05928429 -0.35560823  0.14628297
##  [9,] -0.15429586  0.193829038  0.33113859 -0.08551409 -0.18185918
## [10,]  0.04354047  0.506256632 -0.02252115 -0.26621282 -0.22743623
```

6.Risk Management

```
#VaR and shortfall with normality
VaR.norm=-100000*(s.mean+s.sd*qnorm(0.05))
ES.norm=100000*(-s.mean+s.sd*(dnorm(qnorm(0.05))/0.05))
VaR.norm
```

```
##      ADBE       AMG      GOOG      AMZN      BRKA        EA      FSLR
## 13142.537 17413.474 15439.110 14636.752  7517.573 16352.099 29201.299
##      SINA      URBN      YHOO
## 21901.136 15905.771 16130.457
```

```
ES.norm
```

```
##      ADBE       AMG      GOOG      AMZN      BRKA        EA      FSLR
## 16772.226 22119.010 19605.711 19115.103  9616.026 20685.843 37232.327
##      SINA      URBN      YHOO
## 27842.892 20156.719 20424.664
```

```
#VaR and shortfall with nonparametric
VaR.nonp=vector()
ES.nonp=vector()
t=vector()
for(i in 1:10)
{
  VaR.nonp[i]=-100000*quantile(return[,i],0.05)
  ES.nonp[i]=-100000*sum(return[,i][return[,i]<quantile(return[,i],0.05)])/sum(as.numeric(return[,i]<qua
}
VaR.nonp
```

```
##  [1] 12087.248 15004.833 11654.156 11259.087  7208.361 15299.584 26190.192
##  [8] 20423.951 16822.695 13041.526
```

```
ES.nonp
```

```
##  [1] 17855.330 24588.559 22604.437 17658.369  9922.349 21541.780 31366.086
##  [8] 24568.077 21686.287 17928.176
```

```
#bootstraping
#bootstrap with normality
num.boot=1000
n=length(return[,1])
resample.VaR.norm=matrix(nrow=num.boot,ncol=10)
SD.norm=vector()
ci.norm.upper=vector()
ci.norm.lower=vector()
for(i in 1:10)
{
  resample.VaR.norm[,i]=rep(0,num.boot)
  for(j in 1:num.boot)
  {
    r=sample(return[,i],n,replace=TRUE)
    resample.VaR.norm[j,i]=-100000*(mean(r)+sd(r)*qnorm(0.05))
  }
  SD.norm[i]=sqrt(1/(num.boot-1)*sum((resample.VaR.norm[j,i]- mean(resample.VaR.norm[,i]))^2))
  ci.norm.upper[i]=quantile(resample.VaR.norm[,i],0.975)
  ci.norm.lower[i]=quantile(resample.VaR.norm[,i],0.025)
}
SD.norm
```

```
##  [1] 81.499811 46.521835 37.040269  2.940446  6.132863 83.843789 86.246651
##  [8] 24.434170 26.034027 26.815028
```

```
ci.norm.upper
```

```
##  [1] 16158.950 22768.346 21284.261 18338.789  8771.949 19919.210 33916.838
##  [8] 25026.858 19059.037 19188.797
```

```
ci.norm.lower
```

```
##  [1] 10121.113 12429.539 10574.237 10817.547  6025.408 12900.254 23547.362
##  [8] 18153.489 12728.119 12978.939
```

```
#bootstrap with nonparametric
num.boot=1000
n=length(return[,1])
resample.VaR.nonp=matrix(nrow=num.boot,ncol=10)
SD.nonp=vector()
ci.nonp.upper=vector()
ci.nonp.lower=vector()
for(i in 1:10)
{
  resample.VaR.nonp[,i]=rep(0,num.boot)
  for(j in 1:num.boot)
  {
    r=sample(return[,i],n,replace=TRUE)
    resample.VaR.nonp[j,i]=-100000*quantile(r,0.05)
  }
  SD.nonp[i]=sqrt(1/(num.boot-1)*sum((resample.VaR.nonp[j,i]-mean(resample.VaR.nonp[,i]))^2))
  ci.nonp.upper[i]=quantile(resample.VaR.nonp[,i],0.975)
  ci.nonp.lower[i]=quantile(resample.VaR.nonp[,i],0.025)
}
SD.nonp
```

```
##  [1]   2.494484 148.432694 128.958617  33.826329  31.973382  99.267804
##  [7] 142.319307  27.117265   1.900274  20.046375
```

```
ci.nonp.upper
```

```
##  [1] 17612.22 16457.39 18392.44 17027.02 10323.06 18900.87 31739.13
##  [8] 24192.64 21678.32 15597.24
```

```
ci.nonp.lower
```

```
##  [1]  8394.261 10374.177  9902.726  8986.365  4715.571 11278.540 21724.475
##  [8] 15769.707 11232.336 10727.670
```

7.Copula

```
library(MASS)
library(copula)
library(fGarch)
```

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

```
## Warning: package 'timeSeries' was built under R version 3.2.3
```

```
## Loading required package: fBasics
```

```
##
```

```
## Rmetrics Package fBasics
```

```
## Analysing Markets and calculating Basic Statistics
```

```
## Copyright (C) 2005-2014 Rmetrics Association Zurich
```

```
## Educational Software for Financial Engineering and Computational Science
```

```
## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.
```

```
## https://www.rmetrics.org --- Mail to: info@rmetrics.org
```

```
library(QRM)
```

```
## Warning: package 'QRM' was built under R version 3.2.4
```

```
## Loading required package: gsl
```

```
## Warning: package 'gsl' was built under R version 3.2.3
```

```
##
## Attaching package: 'gsl'
```

```
## The following objects are masked from 'package:copula':
##
##     psi, sinc
```

```
## Loading required package: Matrix
## Loading required package: mvtnorm

## Loading required package: numDeriv

##
## Attaching package: 'QRM'

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

```
n=length(return[,i])
fit.par=matrix(nrow=11,ncol=3)

for (i in 1:11){
start = c(mean(return[,i]), sd(return[,i]), 5)
loglik_t = function(beta) sum( - dt((return[,i] - beta[1]) / beta[2],
beta[3], log = TRUE) + log(beta[2]) )
fit_t = optim(start, loglik_t, hessian = T,
method = "L-BFGS-B", lower = c(-1, 0.001, 1))
fit.par[i,]=fit_t$par
}


est.norm=matrix(ncol=2,nrow=11)
for(i in 1:11){
    est.norm[i,]=as.numeric(fitdistr(return[,i],"normal")$estimate)
    est.norm[i,2]=est.norm[i,2]^2
}

n=length(return[,1])
data1=matrix(nrow=nrow(stocks),ncol=ncol(stocks))
data2=matrix(nrow=nrow(stocks),ncol=ncol(stocks))
data3=matrix(nrow=nrow(stocks),ncol=ncol(stocks))
for(i in 1:10){
  data1[,i]=pstd(return[,i],fit.par[i,1],fit.par[i,2],fit.par[i,3])
  data2[,i]=rank(return[,i])/(1+n)
  data3[,i]=pnorm(return[,i],est.norm[i,1],est.norm[i,2])
}

omega_t=cor(data1)[lower.tri(cor(data1))]
omega_norm=cor(data3)[lower.tri(cor(data3))]

fit.tcopula(data1,method="Kendall")
```

```
## $P
##            [,1]      [,2]      [,3]       [,4]       [,5]      [,6]
##  [1,] 1.0000000 0.5462480 0.4370122 0.31477165 0.35453347 0.3836006
##  [2,] 0.5462480 1.0000000 0.5341323 0.45743740 0.43567206 0.3688814
##  [3,] 0.4370122 0.5341323 1.0000000 0.47369378 0.20749081 0.3349610
##  [4,] 0.3147717 0.4574374 0.4736938 1.00000000 0.06994357 0.3086392
##  [5,] 0.3545335 0.4356721 0.2074908 0.06994357 1.00000000 0.2288084
##  [6,] 0.3836006 0.3688814 0.3349610 0.30863924 0.22880841 1.0000000
##  [7,] 0.2739720 0.3512819 0.2696721 0.32089095 0.23460378 0.2418363
##  [8,] 0.4697540 0.4688773 0.3803889 0.36472476 0.10554880 0.2504975
##  [9,] 0.2244565 0.3255890 0.3100556 0.11936136 0.28303143 0.2672809
## [10,] 0.4095768 0.4710683 0.3410351 0.19387287 0.31900950 0.3679584
##             [,7]      [,8]      [,9]     [,10]
##  [1,] 0.2739720 0.4697540 0.2244565 0.4095768
##  [2,] 0.3512819 0.4688773 0.3255890 0.4710683
##  [3,] 0.2696721 0.3803889 0.3100556 0.3410351
##  [4,] 0.3208909 0.3647248 0.1193614 0.1938729
##  [5,] 0.2346038 0.1055488 0.2830314 0.3190095
##  [6,] 0.2418363 0.2504975 0.2672809 0.3679584
##  [7,] 1.0000000 0.3199504 0.1390529 0.2065194
##  [8,] 0.3199504 1.0000000 0.1753320 0.2528999
##  [9,] 0.1390529 0.1753320 1.0000000 0.2639305
## [10,] 0.2065194 0.2528999 0.2639305 1.0000000
##
## $nu
## [1] 2.635911
##
## $converged
## [1] TRUE
##
## $ll.max
## [1] -594.3292
##
## $fit
## $fit$par
## [1] 2.635911
##
## $fit$objective
## [1] 594.3292
##
## $fit$convergence
## [1] 0
##
## $fit$iterations
## [1] 10
##
## $fit$evaluations
## function gradient
##       12       13
##
## $fit$message
## [1] "relative convergence (4)"
```

```
cop_norm=normalCopula(param=omega_norm,dim=10,dispstr='un')
fit_norm=fitCopula(data=data1,copula=cop_norm,method='ml')
fit_norm2=fitCopula(data=data2,copula=cop_norm,method="ml")

cop_t=tCopula(omega_t,dim=10,dispstr="un")
fit_t=fitCopula(data=data1,copula=cop_t,method="ml")
fit_t2=fitCopula(data=data2,copula=cop_t,method="ml")

cop_gum=archmCopula(family="gumbel",dim=10,param=5)
```

```
fit_gum=fitCopula(data=data1,copula=cop_gum,method="ml")
fit_gum2=fitCopula(data=data2,copula=cop_gum,method="ml")

cop_clay=archmCopula(family="clayton",dim=10,param=5)
fit_clay=fitCopula(data=data1,copula=cop_clay,method="ml")
fit_clay2=fitCopula(data=data2,copula=cop_clay,method="ml")

c(AIC(fit_norm,fit_t,fit_gum,fit_clay))
```

```
## $df
## [1] 45 46  1  1
##
## $AIC
## [1] -233.7391 -257.0022 -145.2595 -265.9154
```

```
c(AIC(fit_norm2,fit_t2,fit_gum2,fit_clay2))
```

```
## $df
## [1] 45 46  1  1
##
## $AIC
## [1] -205.8718 -222.9012 -185.3823 -246.6889
```

8.MCMC

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.2.0
```

```
## Loaded modules: basemod,bugs
```

```
library(Ecdat)
```

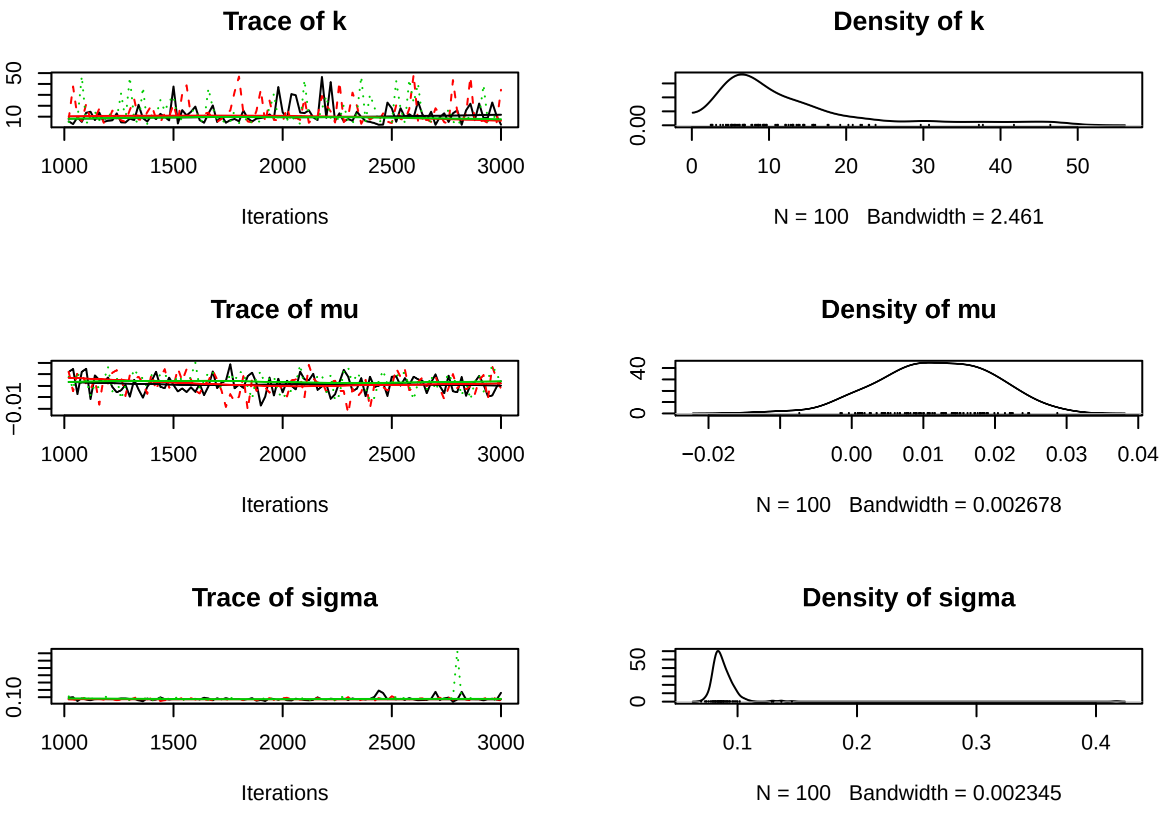
```
##
## Attaching package: 'Ecdat'
```

```
## The following object is masked from 'package:MASS':
##
##     SP500
```

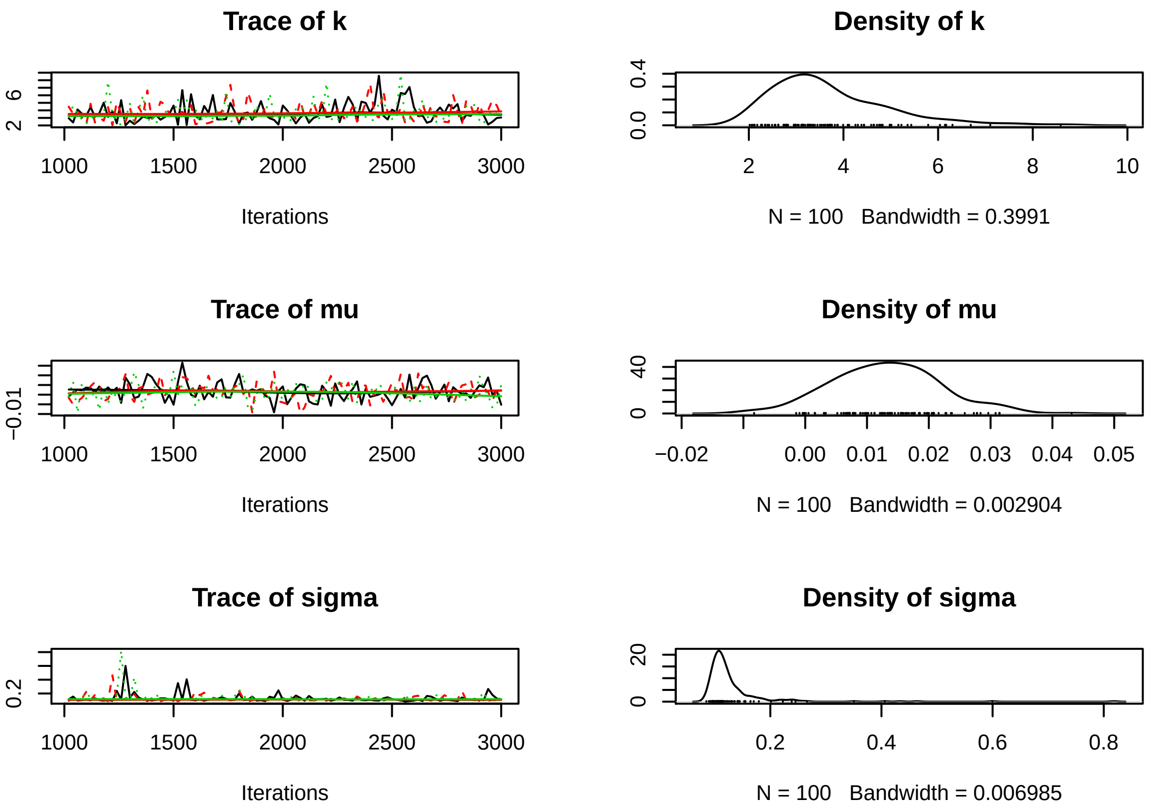
```
## The following object is masked from 'package:datasets':
##
##     Orange
```

```
N=length(return[,1])
i=2
mcmc.est=matrix(0,nrow=11,ncol=3)
for (i in 1:11){
  print(c("Posterior Distribution of",colnames(data)[i]))
  r=return[,i]
  data.mcmc=list(r=r,N=N)
inits=function(){list(mu=rnorm(1,mean=mean(r),sd=2*sd(r)),tau=runif(1,0.2/var(r),2/var(r)),k=runif(1,2.5

t1=proc.time()
univt.mcmc=jags.model("~/Downloads/bugs/univt.bug",data=data.mcmc,inits=inits,n.chains=3,n.adapt=1000,qu
nthin = 20
univt.coda = coda.samples(univt.mcmc, c("mu","k","sigma"), 100*nthin, thin = nthin)
mcmc.est[i,]=summary(univt.coda,digits=2)$statistics[,1]

t2 = proc.time()
(t2-t1)/60

par(mfrow = c(3, 2))
plot(univt.coda, auto.layout = F)
dic.samples(univt.mcmc, 100*nthin, thin = nthin, type = "pD")
}
```

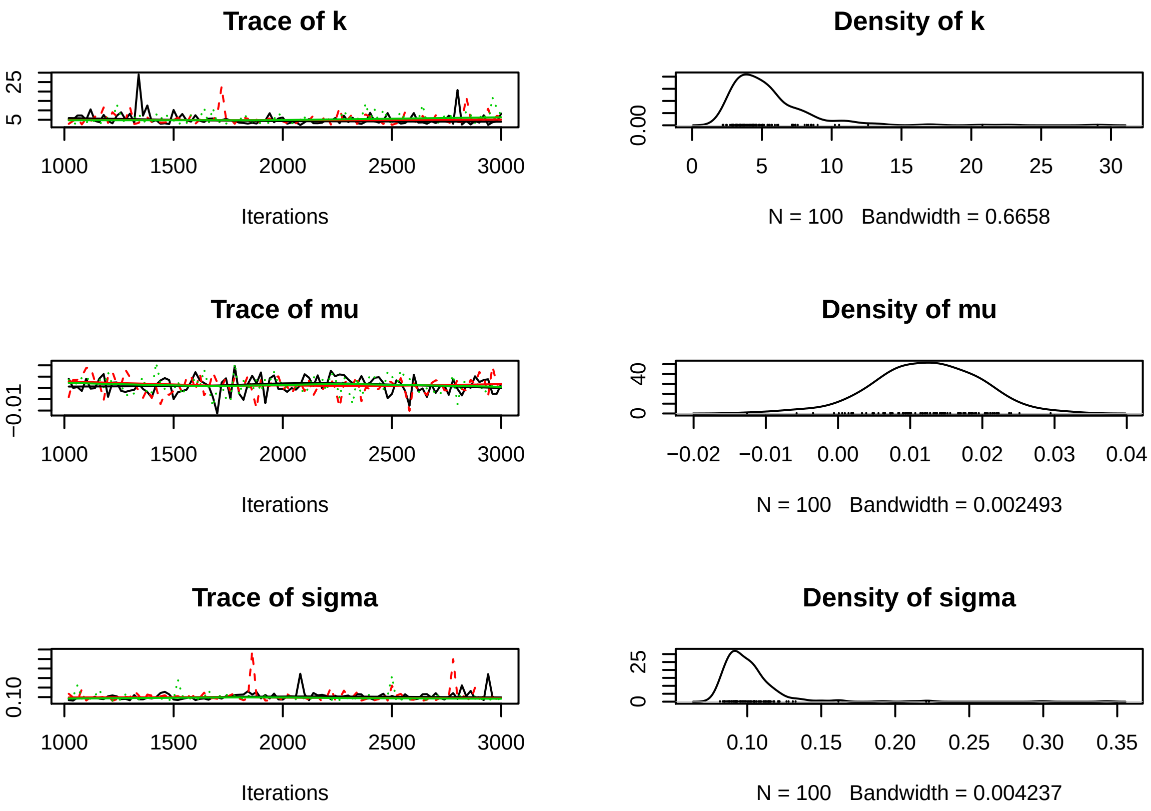
```
## [1] "Posterior Distribution of" "ADBE"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



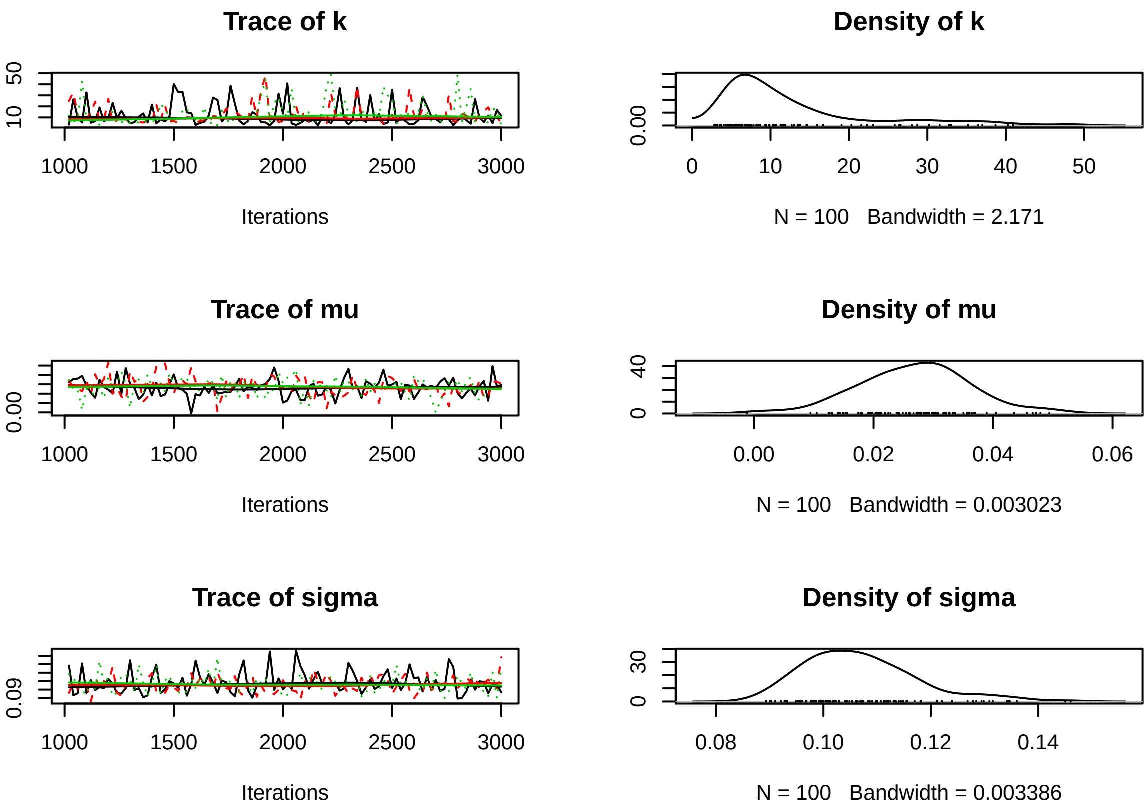
```
## [1] "Posterior Distribution of" "AMG"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



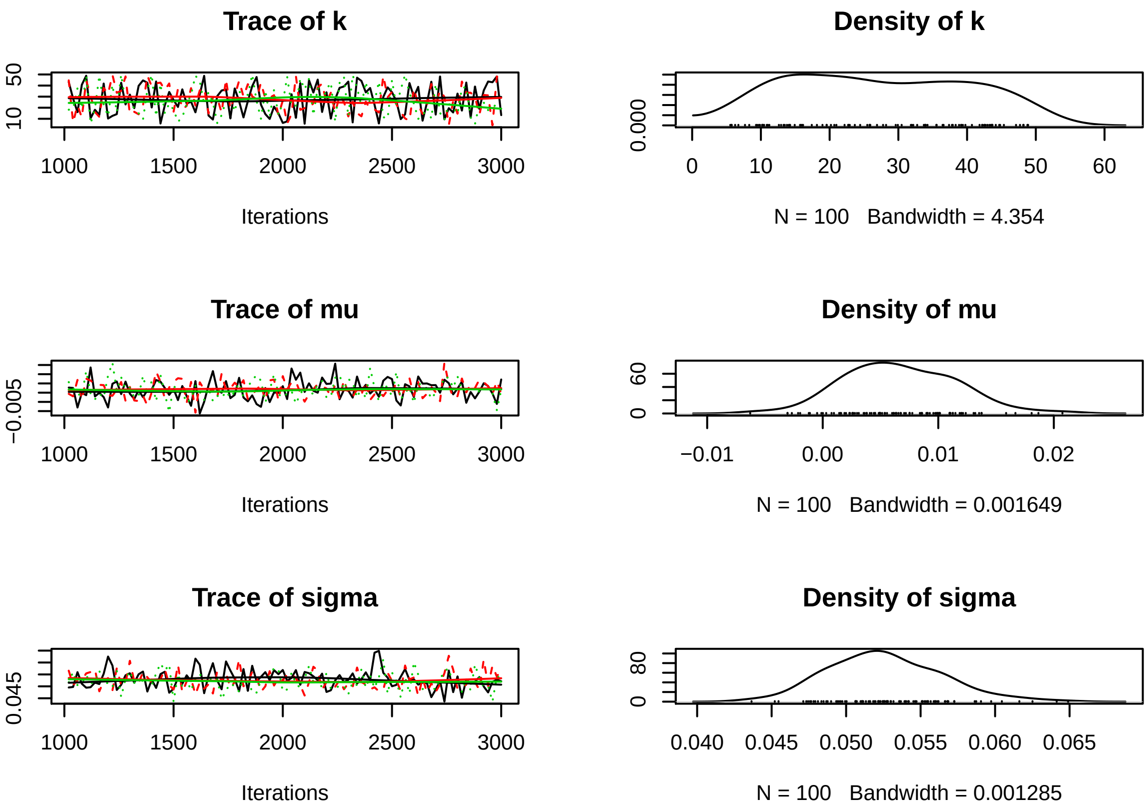
```
## [1] "Posterior Distribution of" "GOOG"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



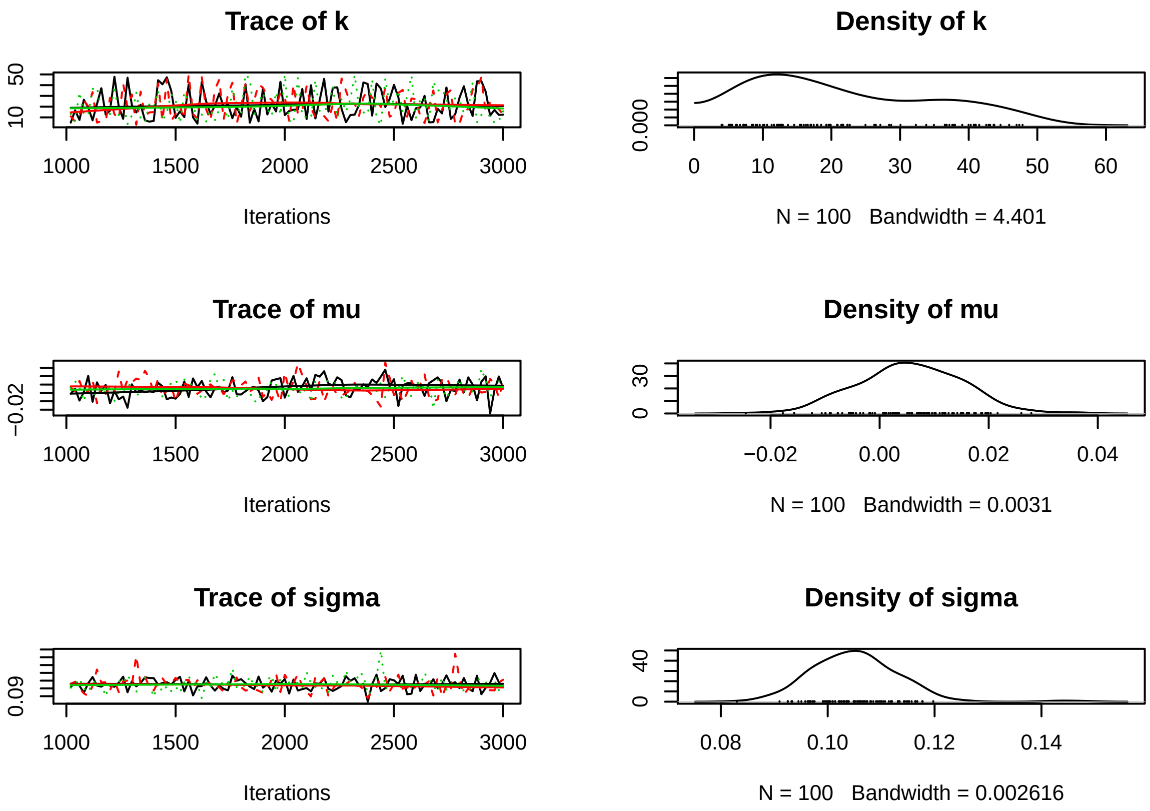
```
## [1] "Posterior Distribution of" "AMZN"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



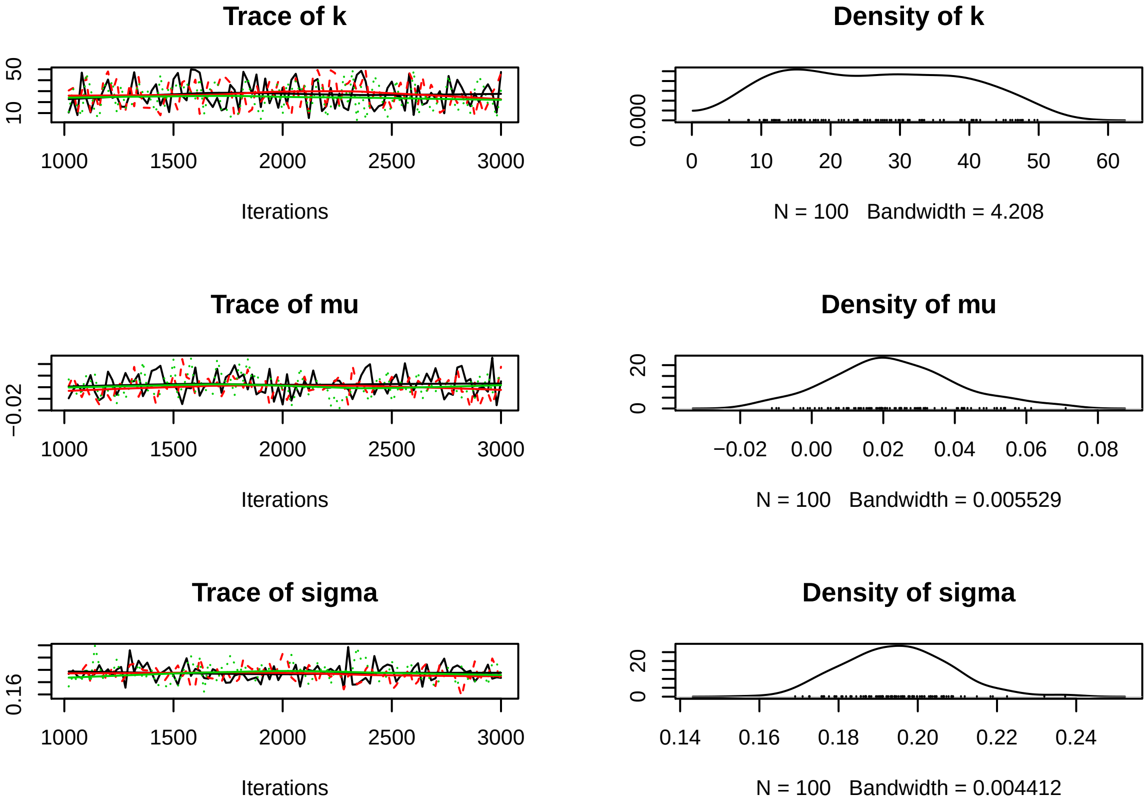
```
## [1] "Posterior Distribution of" "BRKA"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



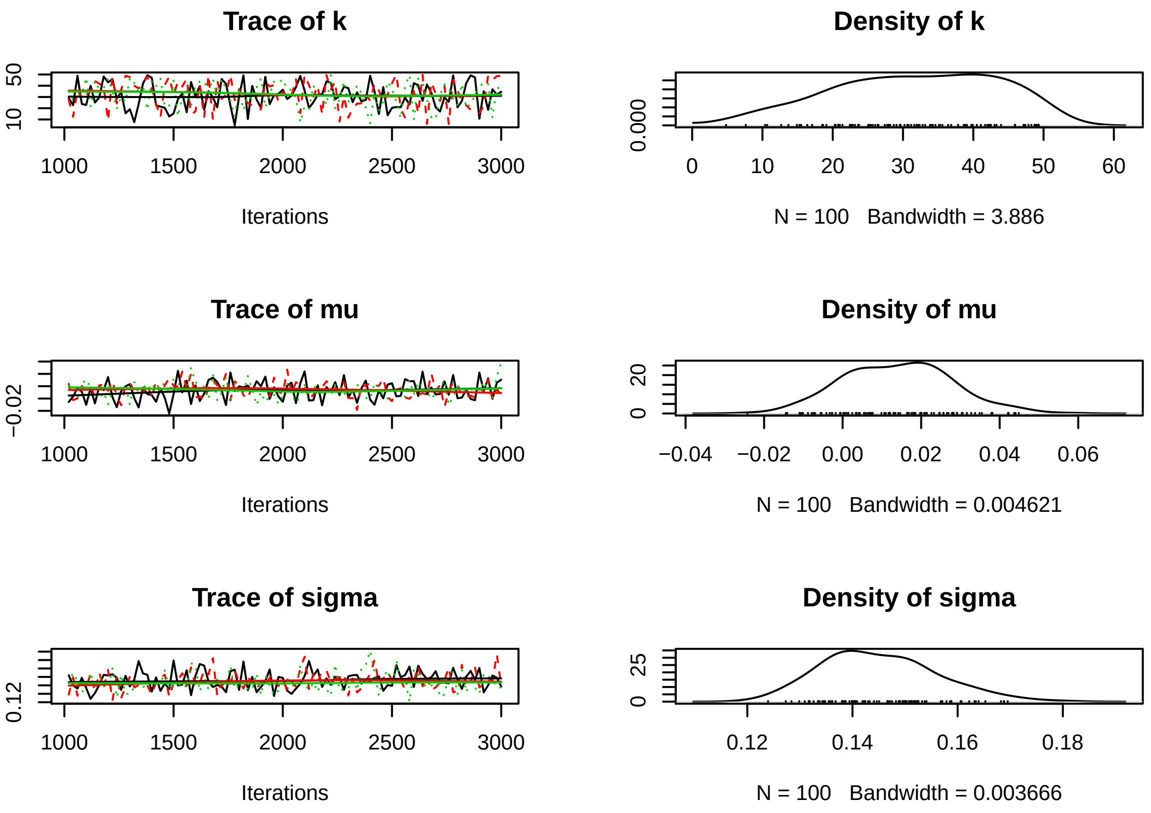
```
## [1] "Posterior Distribution of" "EA"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



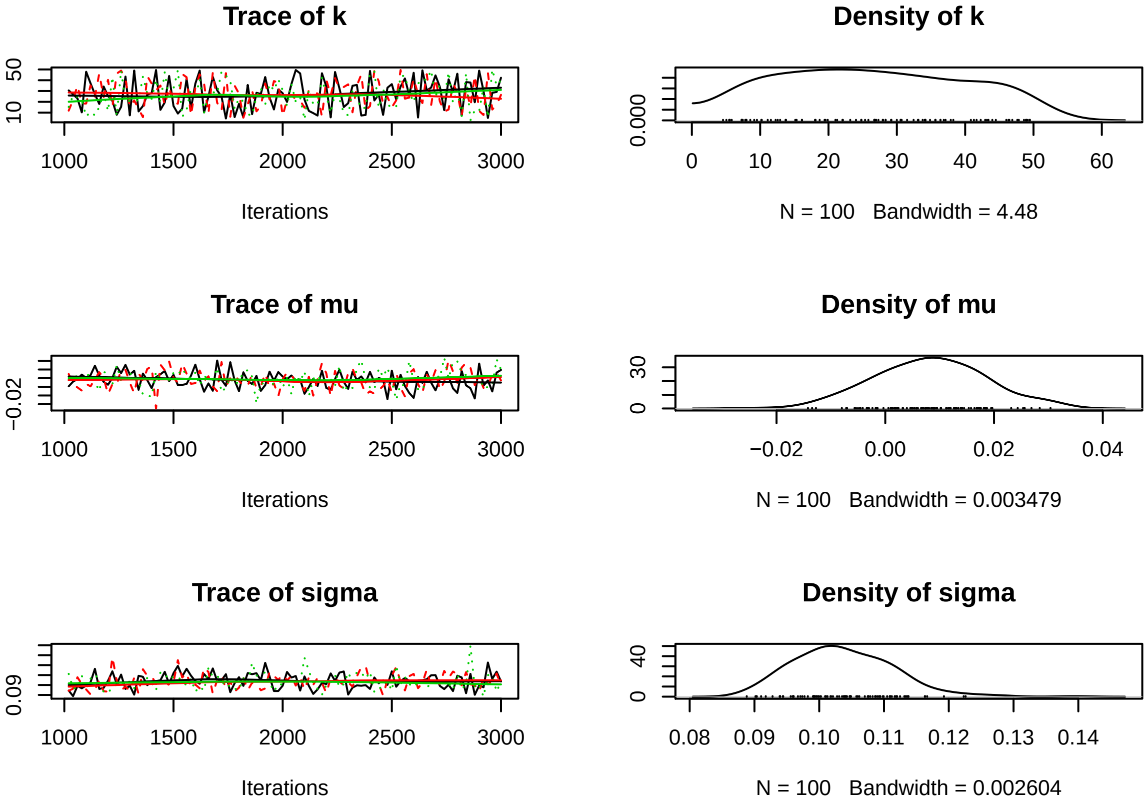
```
## [1] "Posterior Distribution of" "FSLR"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



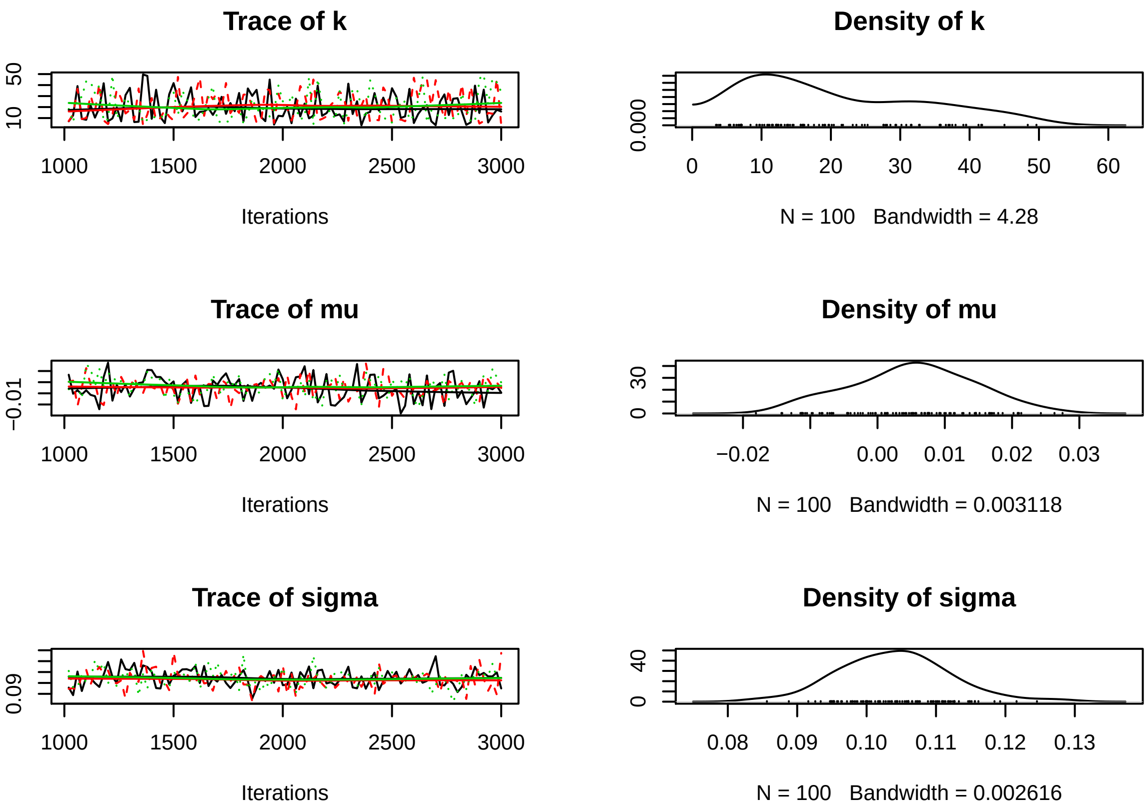
```
## [1] "Posterior Distribution of" "SINA"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



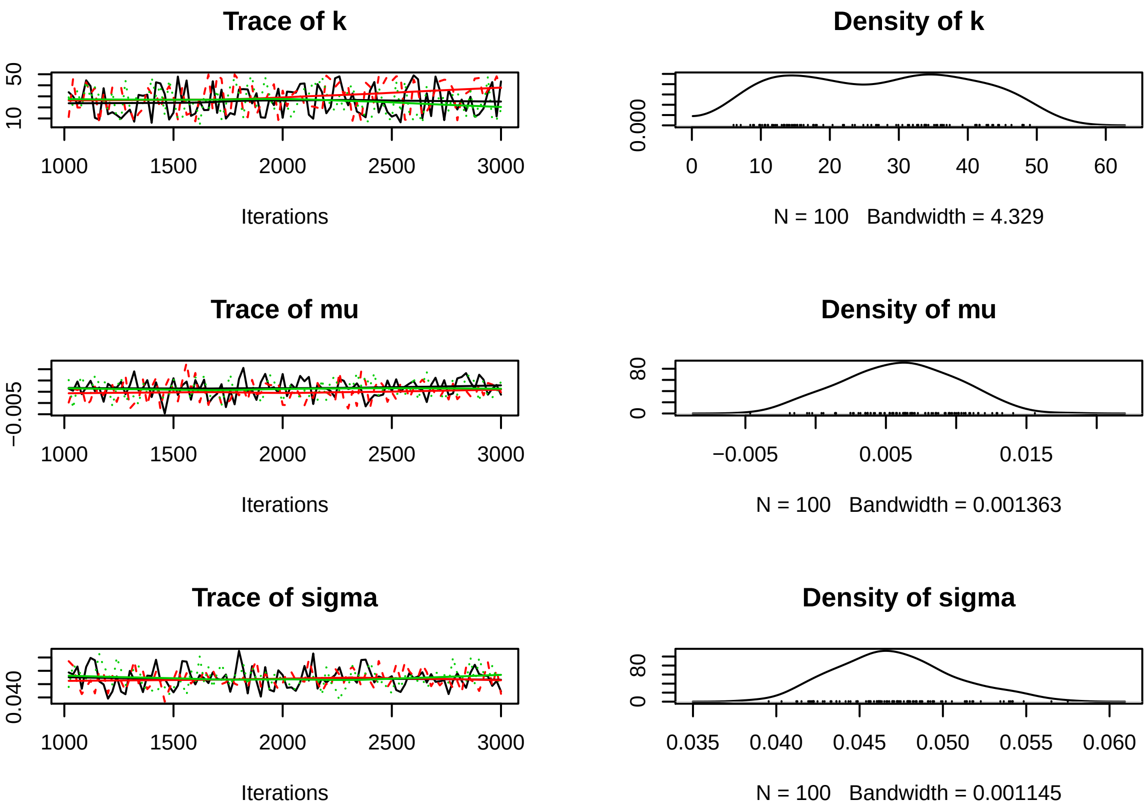
```
## [1] "Posterior Distribution of" "URBN"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



```
## [1] "Posterior Distribution of" "YHOO"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



```
## [1] "Posterior Distribution of" "SP"
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 113
##    Unobserved stochastic nodes: 3
##    Total graph size: 129
##
## Initializing model
```



```
colnames(mcmc.est)=c("k","mu","sigma")
rownames(mcmc.est)=c("ADBE","AMG","GOOG","AMZN","BRKA","EA","FSLR","SINA","URBN","YHOO","SP")

var.mcmc=-100000*(mcmc.est[,2]+mcmc.est[,3]*qt(0.05,mcmc.est[,1]))
es.mcmc=100000*(-mcmc.est[,2]+mcmc.est[,3]*(dt(qt(0.05,mcmc.est[,1]),mcmc.est[,1])/0.05))
var.mcmc
```

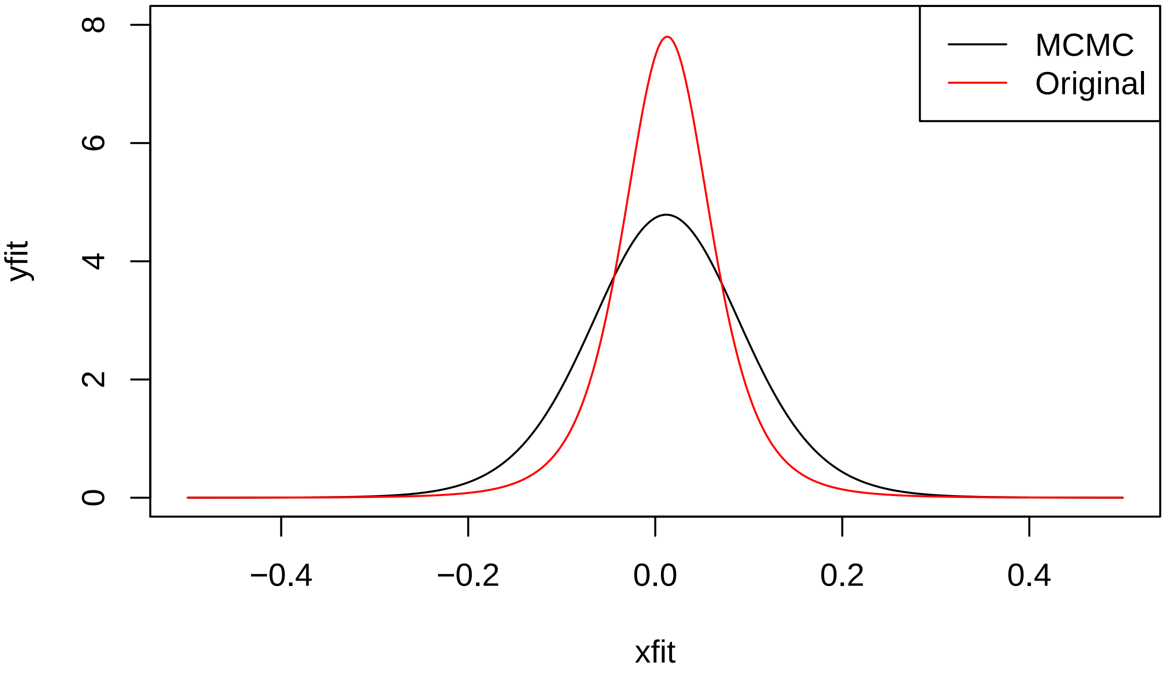
```
##      ADBE       AMG      GOOG      AMZN      BRKA        EA      FSLR
## 14558.743 27131.128 19042.430 16293.832  8286.440 17514.885 30907.492
##      SINA      URBN      YHOO        SP
## 23189.806 16863.080 17403.200  7433.549
```

```
es.mcmc
```

```
##      ADBE       AMG      GOOG      AMZN      BRKA        EA      FSLR
## 14140.447 12563.875 12450.269 15648.260  9295.633 18961.729 34514.967
##      SINA      URBN      YHOO        SP
## 26435.908 18799.221 18658.927  8338.956
```

```
s1=rstd(5000,mcmc.est[1,2],mcmc.est[1,3],mcmc.est[1,1])
s2=rstd(5000,fit.par[1,1],fit.par[1,2],fit.par[1,3])
par(mfrow=c(1,1))
xfit=seq(-0.5,0.5,0.001)
yfit=dstd(xfit,mcmc.est[1,2],mcmc.est[1,3],mcmc.est[1,1])
yfit2=dstd(xfit,fit.par[1,1],fit.par[1,2],fit.par[1,3])
help.search("plot distribution")
plot(yfit~xfit,type="l",ylim=c(0,8),xlim=c(-0.5,0.5))
```

```
lines(yfit2~xfit,type="l",col="red")
legend("topright",legend=c("MCMC","Original"),lty=c(1,1),col=c("black","red"),cex = 1, pt.cex = 1, text
```


MCMC
Original
yfit
xfit