```
# bootStrap.r                   script file for computing descriptive statistics
# author: Eric Zivot
# created: October 10, 2003
# update history:
# July 17, 2012
#   updated for summer 2012
# November 2, 2009
#
# Remarks:
# 1.  The R function boot() from the package boot may be used for bootping arbitrary
#     functions applied to data.
#
# Example data sets
# cerExample.csv                price data on sbux, msft and sp500 from
#                               June 1992 - October 2000
#                       can be downloaded from
#
# http://faculty.washington.edu/ezivot/econ424/424notes.htm
#

# R function used
#
# abline()                      add line to plot
# cbind()                       combine columns
# class()                       return class of object
# colnames()            extract column names from data object
# cor()                 compute correlation matrix
# diff()                        compute differences
# exp()                 compute exponential
# function()            create R function
# lag()                 compute lags
# length()                      compute length of data object
# library()                     load package
# mean()                        compute sample mean
# names()                       extract names from list
# par()                 set graphics parameters
# qnorm()                       compute normal quantile
# read.csv()            read comma separated value files
# sd()                  compute sample standard deviation
# sqrt()                        compute square root
#
# Package boot functions used
#
# boot()                        implement boot for given statistic
# print.boot()          print boot object
# plot.boot()           plot bootstrap distribution
# boot.ci()                     compute bootstrap confidence intervals
#
# Package zoo functions used
#
# zoo()                 create zoo object
#

# load boot and zoo packages
options(digits=4, width=70)
library(boot)
library(zoo)

# read prices from csv file on class webpage
cerExample.df =
```

```
read.csv(file="http://faculty.washington.edu//ezivot//econ424//cerExample.csv")
# create zooreg object - regularly spaced zoo object
cerExample.z = zooreg(data=as.matrix(cerExample.df), start=c(1992,6),
                      end=c(2000,10), frequency=12)
index(cerExample.z) = as.yearmon(index(cerExample.z))
cerExample.z = as.zoo(cerExample.z)

colnames(cerExample.z)
start(cerExample.z)
end(cerExample.z)

returns.z = diff(log(cerExample.z))
colnames(returns.z)
start(returns.z)
end(returns.z)

#
# plot prices and returns for Microsoft
#
par(mfrow=c(2,1))
        plot(cerExample.z[,"msft"], col="blue", lwd=2, ylab="price",
        main="Monthly Prices on MSFT")
        plot(log(cerExample.z[,"msft"]),col="blue", lwd=2, ylab="log price")
par(mfrow=c(1,1))

plot(returns.z[,"msft"],ylab="cc return",
     main="Monthly cc returns on Microsoft",
     col="blue", lwd=2)
abline(h=0)

#
# put data in matrix
#
returns.mat = coredata(returns.z)
MSFT = returns.mat[,"msft"]
SBUX = returns.mat[,"sbux"]
SP500 = returns.mat[,"sp500"]

#
# compute estimates of mu, sigma for MSFT and rho for MSFT and SP500
#
muhat.MSFT = mean(MSFT)
sigmahat.MSFT = sd(MSFT)
rhohat.MSFT.SP500 = cor(returns.mat[,c("msft","sp500")])[1,2]
muhat.MSFT
sigmahat.MSFT
rhohat.MSFT.SP500

#
# brute force bootstrap: MSFT mean
#
muhat.MSFT = mean(MSFT)
sigmahat.MSFT = sd(MSFT)
se.muhat.MSFT = sigmahat.MSFT/sqrt(length(MSFT))
rbind(muhat.MSFT, se.muhat.MSFT)

# sample() function
# random permutations of the index vector 1:5
sample(5)
sample(5)
# random sample of size 5 from MSFT return with replacement
```

```
sample(MSFT, 5, replace=TRUE)

B = 999
muhat.boot = rep(0, B)
nobs = length(MSFT)
for (i in 1:B) {
  boot.data = sample(MSFT, nobs, replace=TRUE)
  muhat.boot[i] = mean(boot.data)
}

# bootstrap bias
mean(muhat.boot) - muhat.MSFT
# bootstrap SE
sd(muhat.boot)
# analytic SE
sigmahat.MSFT/sqrt(length(MSFT))

par(mfrow=c(1,2))
hist(muhat.boot, col="slateblue1")
abline(v=muhat.MSFT, col="white", lwd=2)
qqnorm(muhat.boot)
qqline(muhat.boot)
par(mfrow=c(1,1))






#
# boot mean estimates using R boot package function boot()
#
?boot
# note: boot requires user-supplied functions that take
# two arguments: data and an index. The index is created
# by the boot function and represents random resampling with
# replacement

# function for bootstrapping sample mean
mean.boot = function(x, idx) {
# arguments:
# x             data to be resampled
# idx           vector of scrambled indices created by boot() function
# value:
# ans           mean value computed using resampled data
     ans = mean(x[idx])
     ans
}

MSFT.mean.boot = boot(MSFT, statistic = mean.boot, R=999)
class(MSFT.mean.boot)
names(MSFT.mean.boot)

# print, plot and qqnorm methods
MSFT.mean.boot
# compare boot SE with analytic SE
se.muhat.MSFT = sigmahat.MSFT/sqrt(length(MSFT))
se.muhat.MSFT

# plot bootstrap distribution and qq-plot against normal
plot(MSFT.mean.boot)
```

```
# compute bootstrap confidence intervals from normal approximation
# basic bootstrap method and percentile intervals
boot.ci(MSFT.mean.boot, conf = 0.95, type = c("norm","perc"))

# compare boot confidence intervals with analytic confidence interval
MSFT.lower = muhat.MSFT - 2*se.muhat.MSFT
MSFT.upper = muhat.MSFT + 2*se.muhat.MSFT
cbind(MSFT.lower,MSFT.upper)

#
# boostrap SD estimate
#
# function for bootstrapping sample standard deviation
sd.boot = function(x, idx) {
# arguments:
# x             data to be resampled
# idx           vector of scrambled indices created by boot() function
# value:
# ans           sd value computed using resampled data
     ans = sd(x[idx])
     ans
}

MSFT.sd.boot = boot(MSFT, statistic = sd.boot, R=999)
MSFT.sd.boot

# compare boot SE with analytic SE
se.sigmahat.MSFT = sigmahat.MSFT/sqrt(2*length(MSFT))
se.sigmahat.MSFT

# plot bootstrap distribution
plot(MSFT.sd.boot)

# compute confidence intervals
boot.ci(MSFT.sd.boot, conf=0.95, type=c("norm", "basic", "perc"))

# compare boot confidence intervals with analytic confidence interval
MSFT.lower = sigmahat.MSFT - 2*se.sigmahat.MSFT
MSFT.upper = sigmahat.MSFT + 2*se.sigmahat.MSFT
cbind(MSFT.lower,MSFT.upper)

# bootstrap correlation

# function to compute correlation between 1st 2 variables in matrix
rho.boot = function(x.mat, idx) {
# x.mat n x 2 data matrix to be resampled
# idx           vector of scrambled indices created by boot() function
# value:
# ans           correlation value computed using resampled data

        ans = cor(x.mat[idx,])[1,2]
        ans
}
MSFT.SP500.cor.boot = boot(returns.mat[,c("msft","sp500")],
                           statistic=rho.boot, R = 999)
MSFT.SP500.cor.boot
# compare boot SE with analytic SE based on CLT
se.rhohat.MSFT.SP500 = (1 - rhohat.MSFT.SP500^2)/sqrt(length(MSFT))
se.rhohat.MSFT.SP500
```

```
# plot bootstrap distribution
plot(MSFT.SP500.cor.boot)

# bootstrap confidence intervals
boot.ci(MSFT.SP500.cor.boot, conf=0.95, type=c("norm", "basic", "perc"))

# boot estimate of normal distribution quantile

norm.quantile.boot = function(x, idx, p=0.05) {
# x.mat data to be resampled
# idx           vector of scrambled indices created by boot() function
# p             probability value for quantile
# value:
# ans           normal quantile value computed using resampled data

        q = mean(x[idx]) + sd(x[idx])*qnorm(p)
        q
}


MSFT.q05.boot = boot(MSFT, statistic=norm.quantile.boot, R=999)
MSFT.q05.boot
plot(MSFT.q05.boot)
boot.ci(MSFT.q05.boot, conf=0.95, type=c("norm", "basic", "perc"))

# 5% Value-at-Risk
ValueAtRisk.boot = function(x, idx, p=0.05, w=100000) {
# x.mat data to be resampled
# idx           vector of scrambled indices created by boot() function
# p             probability value for VaR calculation
# w             value of initial investment
# value:
# ans           Value-at-Risk computed using resampled data

        q = mean(x[idx]) + sd(x[idx])*qnorm(p)
        VaR = (exp(q) - 1)*w
        VaR
}

MSFT.VaR.boot = boot(MSFT, statistic = ValueAtRisk.boot, R=999)
MSFT.VaR.boot
boot.ci(MSFT.VaR.boot, conf=0.95, type=c("norm", "perc"))

plot(MSFT.VaR.boot)
```