```
# cerModelExamples.r            scripts for examples in the CER model chapter
#
# author: Eric Zivot
# created: January 30, 2002
# update History:

# July 26, 2011
#   updated examples for summer 2011.
#
# data for examples
# cerExample            dataframe with month prices on Microsoft, Starbucks and SP500
#                                       from June 1992 through October 2000. Price data taken
from Yahoo
#
# R function used
#       abline                          draw line on graph
#       acf                                     compute sample autocovariances or
autocorrelations
#       args                            determine agruments of a function
#       boxplot                 compute boxplot
#       cbind                           combine data objects vertically
#       class                           determine class of object
#       colnames                                get column names from object
#       cor                                     compute sample correlation matrix
#       density                 compute smoothed histogram
#       dnorm                           compute density from normal distribution
#       end                                     get end date of time series
#       help                            invoke help system
#       hist                            compute histogram
#       legend                          add legend to graph
#       length                          compute column length of matrix
#       library                 load R package
#       mean                            compute sample mean
#       names                           show names of object
#       pairs                           all pairwise scatterplots
#       par                                     set graphics parameters
#       plot                            generic plot function
#       pnorm                           compute normal CDF
#       points                          add points to a graph
#       qqline                          add line to qq-plot
#       qqnorm                          qq-plot against normal distribution
#       quantile                        compute empirical quantiles
#       range                           compute range = max - min
#       rmvnorm                 generate multivariate normal random numbers
#       rnorm                           generate normal random numbers
#       seq                                     generate sequence of numbers
#       set.seed                        set random number seed
#       sort                            sort data
#       start                           get start date of time series
#       sd                              compute sample standard deviation
#       ts.plot                         time series plot
#       var                                     compute sample variance or covariance matrix
#       ?                                       invoke help system
#
# R Packages Used
#       zoo
#       PerformanceAnalytics
# mvtnorm
#
options(digits=4, width=70)
```

```
# load R packages
library("PerformanceAnalytics")
library("mvtnorm")

#library("zoo")
#library("TSA")


#
# Example data sets
# cerExample            data.frame contain price data on sbux, msft and sp500 from
# June 1992 - October 2000. Source data is in the csv file cerExample.csv
# that can be downloaded from
#
# http://faculty.washington.edu/ezivot/econ424/424notes.htm
#
# 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)

# create cc returns from prices
returns.z = diff(log(cerExample.z))
colnames(returns.z)
start(returns.z)
end(returns.z)
head(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)


# graphically summarize empirical distribution

#
# put data in matrix and compute mean and standard deviation
#
returns.mat = coredata(returns.z)
mu.msft = mean(returns.mat[,"msft"])
sigma.msft = sd(returns.mat[,"msft"])
mu.msft
sigma.msft
```

```
par(mfrow=c(1,2))
plot(returns.z[,"msft"],ylab="returns",
     main="",
     col="blue", lwd=2)
abline(h=0)
hist(returns.mat[, "msft"], main="", xlab="returns", col="slateblue1")
par(mfrow=c(1,1))


par(mfrow=c(2,2))
        hist(returns.mat[,"msft"], xlab="return",ylab="frequency",
        main="Monthly cc returns on MSFT", col="slateblue1")
        boxplot(returns.mat[,"msft"], col="slateblue1")
        plot(density(returns.mat[,"msft"]),type="l",xlab="return",ylab="density",
           lwd=2, col="slateblue1", main="smoothed densiy")
        qqnorm(returns.mat[,"msft"], col="blue")
        qqline(returns.mat[,"msft"])
par(mfrow=c(1,1))



#
# Illustrate simulation of random numbers using inverse CDF technique
#

xx = seq(from=-4, to=4, length=250)
plot(xx,pnorm(xx), type="l", ylab="u", xlab="x", lwd=3)


#
# simulate CER model calibrated to monthly returns on MSFT
#
mu = 0.03
sd.e = 0.10
nobs = 100
set.seed(111)
sim.e = rnorm(nobs, mean=0, sd=sd.e)
sim.ret = mu + sim.e

# plot simulated returns
par(mfrow=c(1,2))
ts.plot(sim.ret, main="",
       xlab="months",ylab="return", lwd=2, col="blue")
abline(h=mu)
hist(sim.ret, main="", xlab="returns", col="slateblue1")
par(mfrow=c(1,1))

# graphically summarize empirical distribution of simulated data
par(mfrow=c(2,2))
        hist(sim.ret, xlab="return",ylab="frequency",
        main="Simulated returns from CER model", col="slateblue1")
        boxplot(sim.ret, col="slateblue1")
        plot(density(sim.ret),type="l",xlab="return",ylab="density",
           col="slateblue1", lwd=2, main="smoothed densiy")
        qqnorm(sim.ret, col="slateblue1")
        qqline(sim.ret)
par(mfrow=c(1,1))

# compare msft data to simulated gaussian data
boxplot(returns.z[,"msft"], sim.ret, col="slateblue1", names=c("MSFT","GWN"))
```

```
# compute quantiles and compare to normal quantiles
pvals = c(0.01,0.05,0.25,0.5,0.75,0.95,0.99)
quantile(probs=pvals,sim.ret)
qnorm(p=pvals,mean=mu,sd=sd)

# compute descriptive statistics
mean(sim.ret)
sd(sim.ret)
skewness(sim.ret)
kurtosis(sim.ret)

# plot autocorrelations

tmp = acf(sim.ret)

#
# simulate random walk model with initial log Price = 1
#
mu = 0.03
sd.e = 0.10
nobs = 100
set.seed(111)
sim.e = rnorm(nobs, mean=0, sd=sd.e)
sim.p = 1 + mu*seq(nobs) + cumsum(sim.e)
sim.P = exp(sim.p)
par(mfrow=c(2,1))
        ts.plot(sim.p, col="blue",lwd=2,
          ylim=c(-2, 4), ylab="log price")
        lines( (1+mu*seq(nobs)), lty="dotted", col="black", lwd=2)
        lines(cumsum(sim.e), col="orange", lty="dashed", lwd=2)
        abline(h=0)
        legend(x="topleft",legend=c("p(t)","E[p(t)]","p(t)-E[p(t)]"),
               lty=c("solid","dotted","dashed"), col=c("blue","black","orange"),
         lwd=2, cex=c(0.75,0.75,0.75))
        ts.plot(sim.P, lwd=2, col="blue", ylab="price")
par(mfrow=c(1,1))

#
# plot returns on msft, sbux and sp500
#

msft.ret = returns.z[,"msft"]
sbux.ret = returns.z[,"sbux"]
sp500.ret = returns.z[,"sp500"]

# panel function to put horizontal lines at zero in each panel
my.panel <- function(...) {
  lines(...)
  abline(h=0)
}
plot(returns.z, col="blue", lwd=2, main="", panel=my.panel)
plot(returns.z, plot.type="single", lwd=2, col=1:3)
legend(x="bottomleft", legend=colnames(returns.z), col=1:3, lwd=2)
abline(h=0)

pairs(returns.mat, col="blue")

# multivariate simulation
mu = c(0.03,0.03,0.01)
sig2.msft = 0.018
```

```
sig2.sbux = 0.011
sig2.sp500 = 0.001
sig.msft.sbux = 0.004
sig.msft.sp500 = 0.002
sig.sbux.sp500 = 0.002
Sigma = matrix(c(sig2.sbux, sig.msft.sbux, sig.sbux.sp500,
                 sig.msft.sbux, sig2.msft, sig.msft.sp500,
                 sig.sbux.sp500, sig.msft.sp500, sig2.sp500),
                 nrow=3, ncol=3, byrow=TRUE)

nobs = 100
set.seed(123)
returns.sim = rmvnorm(nobs, mean=mu, sigma=Sigma)
colnames(returns.sim) = c("sbux", "msft", "sp500")
plot(as.zoo(returns.sim), lwd=2, col="blue", main="",
     ylab=c("sbux", "msft", "sp500"), panel=my.panel)
plot(as.zoo(returns.sim), lwd=2, main="",
     ylab=c("sbux", "msft", "sp500"), plot.type="single",
     col=c("black","red","green"))
abline(h=0)
legend(x="topleft", legend=colnames(returns.sim), col=1:3, lwd=2)
pairs(returns.sim, col="blue")


#
# plot pdf's for unbiased estimator with high variance, and for
# biased estimator with low variance
#
x.min = -3
x.max = 3
npts = 100
x.vals = seq(from=x.min,to=x.max,length=npts)
pdf1 = dnorm(x.vals,mean=0,sd=1)
pdf2 = dnorm(x.vals,mean=0.1,sd=0.25)
ylimits = range(pdf1,pdf2)
plot(x.vals,pdf1,ylim=ylimits,type="l",lty=1,ylab="pdf",
     xlab="estimate value", lwd=2)
segments(x0=0, y0=0, x1=0, y1=dnorm(0), lwd=2)
lines(x.vals, pdf2, lty=2, lwd=2, col="red")
legend(x="topleft",legend=c("theta.hat 1","theta.hat 2"), lty=1:2,
       lwd=c(2,2), col=c(1,2))
segments(x0=0.1, y0=0, x1=0.1, y1=dnorm(0.1, mean=0.1, sd=0.25), lwd=2, col="red")




#
# estimate parameters from CER model
#
options(digits=4)

muhat.vals = apply(returns.mat,2,mean)
muhat.vals
sigma2hat.vals = apply(returns.mat,2,var)
sigma2hat.vals
sigmahat.vals = apply(returns.mat,2,sd)
sigmahat.vals
cov.mat = var(returns.z)
cov.mat
cor.mat = cor(returns.z)
cor.mat
```

```
covhat.vals = cov.mat[lower.tri(cov.mat)]
rhohat.vals = cor.mat[lower.tri(cor.mat)]
names(covhat.vals) <- names(rhohat.vals) <-
c("sbux,msft","sbux,sp500","msft,sp500")
covhat.vals
rhohat.vals


# compute estimated standard error for mean
nobs = nrow(returns.mat)
nobs
se.muhat = sigmahat.vals/sqrt(nobs)
se.muhat
# compute t-ratios
muhat.vals/se.muhat

# evaluate sampling distribution of muhat
x.vals = seq(3, -3, length=100)
pdf1 = dnorm(x.vals)
pdf2 = dnorm(x.vals, sd=1/sqrt(10))
pdf3 = dnorm(x.vals, sd=1/sqrt(50))
ub = max(pdf1, pdf2, pdf3)
plot(x.vals, pdf1, ylim=c(0, ub), type="l", lwd=2)
lines(x.vals, pdf2, lwd=2, col=2, lty=2)
lines(x.vals, pdf3, lwd=2, col=3, lty=3)
legend(x="topleft", legend=c("pdf: T=1", "pdf: T=10", "pdf: T=50"),
       lty=1:3, col=1:3, lwd=2)


# compute exact 95% confidence intervals
t.975 = qt(0.975, df=99)
mu.lower = muhat.vals - t.975*se.muhat
mu.upper = muhat.vals + t.975*se.muhat
mu.width = mu.upper - mu.lower
cbind(mu.lower,mu.upper,mu.width)

# compute approx 95% confidence intervals
mu.lower = muhat.vals - 2*se.muhat
mu.upper = muhat.vals + 2*se.muhat
mu.width = mu.upper - mu.lower
cbind(mu.lower,mu.upper,mu.width)

# compute estimated standard errors for variance and sd
se.sigma2hat = sigma2hat.vals/sqrt(nobs/2)
se.sigmahat = sigmahat.vals/sqrt(2*nobs)
se.sigma2hat
se.sigmahat
# compute approx 95% confidence intervals
sigma2.lower = sigma2hat.vals - 2*se.sigma2hat
sigma2.upper = sigma2hat.vals + 2*se.sigma2hat
sigma2.width = sigma2.upper - sigma2.lower
cbind(sigma2.lower,sigma2.upper,sigma2.width)

sigma.lower = sigmahat.vals - 2*se.sigmahat
sigma.upper = sigmahat.vals + 2*se.sigmahat
sigma.width = sigma.upper - sigma.lower
cbind(sigma.lower,sigma.upper,sigma.width)

# compute estimated standard errors for correlation
se.rhohat = (1-rhohat.vals^2)/sqrt(nobs)
se.rhohat
```

```
# compute approx 95% confidence intervals
rho.lower = rhohat.vals - 2*se.rhohat
rho.upper = rhohat.vals + 2*se.rhohat
rho.width = rho.upper - rho.lower
cbind(rho.lower,rho.upper,rho.width)

#
# Monte Carlo evaluation of unbiasedness
#

# generate 10 simulated samples from CER model and plot
# r = 0.05 + e, e ~ iid N(0, 0.10^2)
mu = 0.05
sd = 0.10
n.obs = 100
n.sim = 10
set.seed(111)
sim.ret = matrix(0, n.obs, n.sim)
for (sim in 1:n.sim) {
        sim.ret[,sim] = rnorm(n.obs,mean=mu,sd=sd)
}
ts.plot(sim.ret,ylab="returns", col=1:10, lty=1:10)
abline(h=0)

plot(as.zoo(sim.ret), col="blue", lwd=2, main="", panel=my.panel)

# generate 1000 samples from CER and evaluate sampling properties of muhat
mu = 0.05
sd = 0.10
n.obs = 100
n.sim = 1000
set.seed(111)
sim.means = rep(0,n.sim)        # initialize vectors
mu.lower = rep(0,n.sim)
mu.upper = rep(0,n.sim)
qt.975 = qt(0.975, nobs-1)
for (sim in 1:n.sim) {
        sim.ret = rnorm(n.obs,mean=mu,sd=sd)
        sim.means[sim] = mean(sim.ret)
        se.muhat = sd(sim.ret)/sqrt(n.obs)
        mu.lower[sim] = sim.means[sim]-qt.975*se.muhat
        mu.upper[sim] = sim.means[sim]+qt.975*se.muhat
}
mean(sim.means)
sd(sim.means)
in.interval = mu >= mu.lower & mu <= mu.upper
sum(in.interval)/n.sim

hist(sim.means, col="slateblue1", ylim=c(0,40), main="", xlab="muhat", probability=T)
abline(v=mean(sim.means), col="white", lwd=4, lty=2)
# overlay normal curve
x.vals = seq(0.02, 0.08, length=100)
lines(x.vals, dnorm(x.vals, mean=mu, sd=sd/sqrt(100)), col="orange", lwd=2)

# compute expected value of estimates and bias
mean(sim.means)
mean(sim.means) - mu
sd(sim.means)
se.muhat["msft"]
```

```
mu = 0.05
sd = 0.10
n.obs = 100
n.sim = 1000
set.seed(111)
sim.means = rep(0,n.sim)        # initialize vectors
sim.vars = rep(0,n.sim)
sim.sds = rep(0,n.sim)
for (sim in 1:n.sim) {
        sim.ret = rnorm(n.obs,mean=mu,sd=sd)
        sim.means[sim] = mean(sim.ret)
        sim.vars[sim] = var(sim.ret)
        sim.sds[sim] = sqrt(sim.vars[sim])
}

# compute expected value of estimates and bias
mean(sim.means)
mean(sim.means) - mu
mean(sim.vars)
mean(sim.vars) - sd^2
mean(sim.sds)
mean(sim.sds) - sd

par(mfrow=c(2,2))
        hist(sim.means, col="slateblue1", xlab="mu hat values", main="mu hat")
        abline(v=mean(sim.means), col="white", lwd=4, lty=2)
        hist(sim.vars, col="slateblue1", xlab="sigma2 hat values", main="sigma2 hat")
        abline(v=mean(sim.vars), col="white", lwd=4, lty=2)
        hist(sim.sds, col="slateblue1", xlab="sigma hat values", main="sigma hat")
        abline(v=mean(sim.sds), col="white", lwd=4, lty=2)
par(mfrow=c(1,1))

#
# compute Monte Carlo standard errors and compare with
# asymptotic formulas based on true values
#

sd(sim.means)
sd/sqrt(nobs)
sd(sim.vars)
sd^2/sqrt(nobs/2)
sd(sim.sds)
sd/sqrt(2*nobs)

#
# plot pdf for different sample sizes
#
x.min = -3
x.max = 3
npts = 100
x.vals = seq(from=x.min,to=x.max,length=npts)
pdf1 = dnorm(x.vals,mean=0,sd=1)
pdf2 = dnorm(x.vals,mean=0,sd=1/sqrt(10))
pdf3 = dnorm(x.vals,mean=0,sd=1/sqrt(50))

ylimits = range(pdf1,pdf2,pdf3)
plot(x.vals,pdf1,ylim=ylimits,type="l",lty=1,ylab="pdf",
lwd=2,xlab="estimate value")
lines(x.vals,pdf2,type="l", lty=1, col=2, lwd=2)
lines(x.vals,pdf3,type="l", lty=1, col=3, lwd=2)
legend(1.5,2.5,legend=c("pdf T=1","pdf T=10","pdf T=50"),
```

```
lty=rep(1,3),col=1:3, lwd=rep(2,3))

#
# LLN
#

nobs = 1000
# Draw iid N(0.03,0.11) values
set.seed(123)
z = rnorm(1000, mean=0.03, sd=0.11)
mu.hat.n = cumsum(z)/1:nobs
ts.plot(mu.hat.n, main="Consistency of Sample Mean from CER Model",
       ylab="Sample Mean", xlab="Sample Size, T")
abline(h=0.03)


#
# Illustration of CLT for muhat, sigma2hat and sigmahat
#

# generate 1000 samples from CER and compute sample statistics
mu = 0.03
sd = 0.11
n.obs = 25
n.sim = 1000
set.seed(111)
sim.means = rep(0,n.sim)        # initialize vectors
sim.vars = rep(0,n.sim)
sim.sds = rep(0,n.sim)
for (sim in 1:n.sim) {
        sim.ret = rnorm(n.obs,mean=mu,sd=sd)
        sim.means[sim] = mean(sim.ret)
        sim.vars[sim] = var(sim.ret)
        sim.sds[sim] = sqrt(sim.vars[sim])
}

# plot pdfs
par(mfrow=c(2,2))
        hist(sim.means,xlab="mu hat", col="slateblue1")
        abline(v=mu, lwd=2, col="white")
        hist(sim.vars,xlab="sigma2 hat", col="slateblue1")
        abline(v=sd^2, lwd=2, col="white")
        hist(sim.sds,xlab="sigma hat", col="slateblue1")
        abline(v=sd, lwd=2, col="white")
par(mfrow=c(1,1))


#
# use Monte Carlo to evaluate confidence interval coverage
#

# generate 1000 samples from CER and compute sample statistics
n.sim = 1000
set.seed(111)
mu.lower = rep(0,n.sim) # initialize vectors
mu.upper = rep(0,n.sim)
for (sim in 1:n.sim) {
        sim.ret = rnorm(n.obs,mean=mu,sd=sd)
        mu.hat = mean(sim.ret)
        se.muhat = sd(sim.ret)/sqrt(n.obs)
        mu.lower[sim] = mu.hat - 2*se.muhat
```

```
        mu.upper[sim] = mu.hat + 2*se.muhat
}
in.interval = (mu >= mu.lower) & (mu <= mu.upper)
sum(in.interval)/n.sim


#
# plot actual data
#

plot(returns.z, lwd=c(2,2,2), col=c(1,2,5),
                    lty=c(1,1,1))
plot(returns.z, plot.type = "single", col=c(1,2,5), lwd=2)
abline(h=0)
legend(x="bottomleft", legend=c("SBUX","MSFT","SP500"),
       lwd=c(2,2,2), col=c(1,2,5), lty=c(1,1,1))

#
# simulate data for three asset returns
#

nobs = 100
set.seed(123)
sim.e = rmvnorm(nobs,mean=rep(0,3),cov=cov.mat)
sim.ret = muhat.vals + sim.e
colIds(sim.ret) = paste(colIds(returns.mat),".sim",sep="")

# plot data and scatterplots
ts.plot(sim.ret,main="Simulated return data",
lty=rep(1,3),lwd=rep(2,3),col=c(1,2,5))
legend(0,-0.2,legend=colIds(sim.ret),
lty=rep(1,3),lwd=rep(2,3),col=c(1,2,5))
abline(h=0)
pairs(sim.ret)

# compare with actual data
ts.plot(returns.mat,main="Actual return data",
lty=rep(1,3),lwd=rep(2,3),col=c(1,2,5))
legend(0,-0.2,legend=colIds(sim.ret),
lty=rep(1,3),lwd=rep(2,3),col=c(1,2,5))
abline(h=0)
pairs(returns.mat)


# generate 1000 samples from CER and compute correlations
n.obs = 100
n.sim = 1000
set.seed(111)
sim.corrs = matrix(0,n.sim,3)   # initialize vectors
colIds(sim.corrs) = c("sbux,msft","sbux,sp500","msft,sp500")

for (sim in 1:n.sim) {
        sim.ret = rmvnorm(n.obs,mean=muhat.vals,sigma=cov.mat)
        cor.mat = cor(sim.ret)
        sim.corrs[sim,] = cor.mat[lower.tri(cor.mat)]
}

par(mfrow=c(2,2))
        hist(sim.corrs[,1], xlab="rhohat(sbux,msft)", col="slateblue1",
       main="sbux,msft")
        abline(v=rhohat.vals[1], lwd=4, col="white")
```

```
        hist(sim.corrs[,2], xlab="rhohat(sbux,sp500)", col="slateblue1",
       main="sbux,sp500")
        abline(v=rhohat.vals[2], lwd=4, col="white")
        hist(sim.corrs[,3], xlab="rhohat(msft,sp500)", col="slateblue1",
       main="msft,sp500")
        abline(v=rhohat.vals[3], lwd=4, col="white")
par(mfrow=c(1,1))

# MC means and standard deviations
colMeans(sim.corrs)
colStdevs(sim.corrs)
se.rhohat

#
# Compute rolling mean and SD values: using rolling functions from zoo
#
class(returns.z)
colIds(returns.z)
start(returns.z)
end(returns.z)

MSFT.z = returns.z[,"msft"]
class(MSFT.z)
plot(MSFT.z,reference.grid=F)


# compute rolling means of width 24 months
roll.mean.24 = aggregateSeries(MSFT.z,moving=23,adj=1,FUN=mean)
roll.mean.24
plot(roll.mean.24,reference.grid=F,
     main="Rolling 24 month mean estimates")
abline(h=muhat.vals["msft"])

plot(roll.mean.24,MSFT.z,reference.grid=F,
     main="MSFT returns and rolling 24 month means")
abline(h=muhat.vals["msft"])

# compute rolling sds of width 24 months
roll.sd.24 = aggregateSeries(MSFT.z,moving=23,adj=1,FUN=sd)
roll.sd.24
plot(roll.sd.24,reference.grid=F,
     main="Rolling 24 month SD estimates")
abline(h=sigmahat.vals["msft"])

plot(roll.sd.24,MSFT.z,reference.grid=F,
     main="MSFT returns and rolling 24 month sds")
abline(h=muhat.vals["msft"])

#
# Estimate Quantiles and VaR from CER model
#

# use R function qnorm to compute quantiles from standard normal distribution
args(qnorm)
qhat.05 = muhat.vals + sigmahat.vals*qnorm(0.05)
qhat.05
qnorm(0.05,muhat.vals,sigmahat.vals)

W0 = 100000
VaRhat.05 = (exp(qhat.05)-1)*W0
VaRhat.05
```

```
# VaR example from slides
muhat = 0.02
sigmahat = 0.10
qhat.05 = muhat + sigmahat*qnorm(0.05)
W0 = 10000
VaRhat.05 = (exp(qhat.05)-1)*W0
VaRhat.05
```