```
# hypothesisTestingCER.r                        R script for hypothesis testing examples in
CER
#                            model
#
# author: Eric Zivot
# created: November 10, 2003
# update history
# July 28, 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
#
# apply()                       apply function to rows or columns of matrix
# colnames()            show column names
# dchisq()                      compute chi-square density
# dim()                 show matrix dimensions
# dt()                  compute Student's t density
# legend()                      add legend to plot
# log()                 compute natural log
# pchisq()                      compute chi-square CDF
# points()                      add points to plot
# pt()                  compute Student's t CDF
# qchisq()                      compute chi-square quantiles
# qt()                  compute Student's t quantiles
# read.csv                      Read csv file
# t.test                        Perform t-test
#
# R package zoo functions used
#
# diff()                        compute difference
# end()                 show ending date
# rollapply()           apply function over rolling window
# start()                       show starting date
# zooreg()                      create zooreg object
#
# Example data sets
# cerExample            data.frame contain price data on sbux, msft and sp500 from
# June 1992 - October 2000. Source data is in the Excel file cerExample.xls
# that can be downloaded from
#
# http://faculty.washington.edu/ezivot/econ424/424notes.htm
#

# load R packages
library("zoo")
library("TSA")
options(digits=4, width=70)

#
# 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)

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

# plot data
# panel function to put horizontal lines at zero in each panel
my.panel <- function(...) {
  lines(...)
  abline(h=0)
}
plot(returns.z, lwd=2, col="blue", panel=my.panel)

#
# chi-square distribution
#
?dchisq

# plot chi-square distribution for various degrees
# of freedom

xx = seq(from=0,to=20,length=200)
plot(xx,dchisq(xx,df=2),main="Chi-Square Distribution",
     type="l", xlab="x", ylab="pdf",lwd=2,col="black")
points(xx,dchisq(xx,df=5),type="l",lwd=2,col="blue")
points(xx,dchisq(xx,df=10),type="l",lwd=2,col="orange")
legend(10,0.4,legend=c("df=2","df=5","df=10"),lty=c(1,1,1),
       lwd=c(2,2,2), col=c("black","blue","orange"))

# plot Student's t distribution for various degrees
# of freedom

xx = seq(from=-5,to=5,length=200)
plot(xx,dt(xx,df=10),main="Student's t Distribution",
     type="l", xlab="t", ylab="pdf",lwd=2,col="black")
points(xx,dt(xx,df=5),type="l",lwd=2,col="blue")
points(xx,dt(xx,df=1),type="l",lwd=2,col="orange")
points(xx,dnorm(xx),type="l",lwd=2,col="green")
legend(x="topleft",legend=c("N(0,1)","df=10","df=5","df=2"),
       lty=c(1,1,1,1), lwd=c(2,2,2,2),
       col=c("green","black","blue","orange"))

# compare quantiles of different t distributions
qnorm(c(0.01,0.05))
qt(c(0.01,0.05), df=10)
qt(c(0.01,0.05), df=5)
```

```
qt(c(0.01,0.05), df=2)

#
# Basic significance tests for CER Model
#

# construct test by brute force
nobs = dim(returns.z)[1]
muhat.vals = apply(returns.z, 2, mean)
muhat.vals
sigmahat.vals = apply(returns.z, 2, sd)
se.muhat = sigmahat.vals/sqrt(nobs)
se.muhat
t.stats = muhat.vals/se.muhat
abs(t.stats)

# compute 2-sided 5% critical values
cv.2sided = qt(0.975, df=nobs-1)
cv.2sided
abs(t.stats) > cv.2sided

# compute 2-sided p-values
2*(1-pt(abs(t.stats),df=nobs-1))

#
?t.test
# Test H0: mu = 0 for msft
t.test.msft = t.test(returns.z[,"msft"],
                     alternative="two.sided",
                     mu=0, conf.level=0.95)
class(t.test.msft)
t.test.msft

# Test H0: mu = 0 for sbux and sp500
t.test(returns.z[,"sbux"], alternative="two.sided",
       mu=0, conf.level=0.95)
t.test(returns.z[,"sp500"], alternative="two.sided",
       mu=0, conf.level=0.95)

#
# test for specific value
#

# Test H0: mu = 0.03 for msft
t.test(returns.z[,"msft"],
       alternative="two.sided",
       mu = 0.03, conf.level=0.95)

#
# test for sign
#

# Test H0: mu > 0 for msft
t.test(returns.z[,"msft"],
       alternative="greater",
       mu = 0, conf.level=0.95)

#
# Paired t-test for equality of means
#
```

```
# Test H0: mu_msft = mu_sbux vs. H1: mu_msft /= mu_sbux
t.test(x=returns.z[,"msft"],
       y=returns.z[,"sbux"],
       paired=T)


#
# test for normality of returns
#

par(mfrow=c(2,2))
        hist(returns.z[,"sbux"],main="Starbucks monthly cc returns",
        probability=T, ylab="cc return", col="slateblue1")
        boxplot(returns.z[,"sbux"],outchar=T, ylab="cc return",
              col="slateblue1")
        plot(density(returns.z[,"sbux"]),type="l",xlab="cc return",
             ylab="density estimate", main="Smoothed density",
           lwd=2, col="slateblue1")
        qqnorm(returns.z[,"sbux"], col="slateblue1")
        qqline(coredata(returns.z[,"sbux"]))
par(mfrow=c(1,1))

sbux.skew = skewness(returns.z[,"sbux"])
sbux.ekurt= kurtosis(returns.z[,"sbux"])                       # note: this is excess
kurtosis

JB = nobs*(sbux.skew^2 + 0.25*sbux.ekurt^2)/6
JB


# compute p-value from chi-square 2 distn

p.value = 1 - pchisq(JB,df=2)
p.value

# use jarque.bera.test() function from tseries package
library(tseries)
jarque.bera.test(returns.z[,"sbux"])

#
# test for no autocorrelation in returns
#

sbux.acf = acf(returns.z[,"sbux"])

#
# diagnostics for constant parameters
#

#
# compute two sample t-test for equality of means
#

# Split sample into two equally sized pieces
# Test H0: E[r_sample1] = E[r_sample2]

t.test(x=returns.z[1:50,"msft"],
       y=returns.z[51:100,"msft"],
       paired=F)
```

```
# compute rolling means for sbux
# using function rollapply
# note: must use zoo objects
#

?rollapply
args(rollapply)

# compute rolling means over 24 month windows
roll.muhat = rollapply(returns.z[,"sbux"], width=24,
                       FUN=mean, align="right")
class(roll.muhat)
roll.muhat[1:5]

# plot rolling estimates with data
plot(merge(roll.muhat,returns.z[,"sbux"]), plot.type="single",
     main="24 month rolling means for SBUX",ylab="returns",
     lwd=c(2,2), col=c("blue","orange"))
abline(h=0)
legend(x="bottomleft",legend=c("Rolling mean","Monthly returns"),
       lwd=c(2,2), col=c("blue","orange"))

# compute rolling standard deviations over 24 month windows
roll.sigmahat = rollapply(returns.z[,"sbux"],width=24,
                          FUN=sd, align="right")
roll.sigmahat[1:5]

# plot rolling estimates with data
plot(merge(roll.sigmahat,returns.z[,"sbux"]), plot.type="single",
     main="24 month rolling SDs for SBUX", ylab="Returns",
     lwd=c(2,2), col=c("blue","orange"))
abline(h=0)
legend(x="bottomleft",legend=c("Rolling SD","Monthly returns"),
       lwd=c(2,2), col=c("blue","orange"))

# repeat analysis for sp500
# compute rolling means over 24 month windows
roll.muhat = rollapply(returns.z[,"sp500"], width=24,
                       FUN=mean, align="right")
class(roll.muhat)
roll.muhat[1:5]

# plot rolling estimates with data
plot(merge(roll.muhat,returns.z[,"sp500"]), plot.type="single",
     main="24 month rolling means for SP500",ylab="returns",
     lwd=c(2,2), col=c("blue","orange"))
abline(h=0)
legend(x="bottomleft",legend=c("Rolling mean","Monthly returns"),
       lwd=c(2,2), col=c("blue","orange"))

# compute rolling standard deviations over 24 month windows
roll.sigmahat = rollapply(returns.z[,"sp500"],width=24,
                          FUN=sd, align="right")
roll.sigmahat[1:5]

# plot rolling estimates with data
plot(merge(roll.sigmahat,returns.z[,"sp500"]), plot.type="single",
     main="24 month rolling SDs for SP500", ylab="Returns",
     lwd=c(2,2), col=c("blue","orange"))
abline(h=0)
legend(x="bottomleft",legend=c("Rolling SD","Monthly returns"),
```

```
       lwd=c(2,2), col=c("blue","orange"))

#
# compute rolling correlations
#
rhohat = function(x) {
        cor(x)[1,2]
}

# compute rolling estimates b/w sp500 and sbux
roll.rhohat = rollapply(returns.z[,c("sp500","sbux")],
                       width=24,FUN=rhohat, by.column=FALSE,
                       align="right")
class(roll.rhohat)
roll.rhohat[1:5]

# plot rolling estimates

plot(roll.rhohat, main="Rolling Correlation b/w SP500 and SBUX",
     lwd=2, col="blue", ylab="rho.hat")
abline(h=0)

# compute rolling correlations b/w sbux and msft
roll.rhohat = roll.rhohat = rollapply(returns.z[,c("sbux","msft")],
                       width=24,FUN=rhohat, by.column=FALSE,
                       align="right")
class(roll.rhohat)
roll.rhohat[1:5]

plot(roll.rhohat, main="Rolling Correlation b/w SBUX and MSFT",
     lwd=2, col="blue", ylab="rho.hat")
abline(h=0)
```