```
# lab5.r                        script file for lab5 calculations
#
# author: Eric Zivot
# created: October 20, 2003
# revised: July 12, 2012
#
# comments:
# Data for the lab are
# monthly continuously compounded returns on Vanguard long term bond index fund
#(VBLTX), Fidelity Magellan stock mutual fund (FMAGX), and Starbucks stock (SBUX)
#
# This lab requires the following packages
# PerformanceAnalytics  return and risk analytics
# zoo                                Zeilie's ordered observations
# tseries               various time series functions
# make sure you install these packages before you load them.

options(digits=4, width=70)

library(PerformanceAnalytics)
library(zoo)
library(tseries)

# get monthly adjusted closing price data on VBLTX, FMAGX and SBUX from Yahoo
# using the tseries function get.hist.quote(). Set sample to Jan 1998 through
# Dec 2009. Note: if you are not careful with the start and end dates
# or if you set the retclass to "ts" then results might look weird

# look at help on get.hist.quote
?get.hist.quote

# get the adjusted closing prices from Yahoo!
VBLTX.prices = get.hist.quote(instrument="vbltx", start="1998-01-01",
                              end="2009-12-31", quote="AdjClose",
                               provider="yahoo", origin="1970-01-01",
                               compression="m", retclass="zoo")
# change class of time index to yearmon which is appropriate for monthly data
# index() and as.yearmon() are functions in the zoo package
#
index(VBLTX.prices) = as.yearmon(index(VBLTX.prices))

class(VBLTX.prices)
colnames(VBLTX.prices)
start(VBLTX.prices)
end(VBLTX.prices)

FMAGX.prices = get.hist.quote(instrument="fmagx", start="1998-01-01",
                              end="2009-12-31", quote="AdjClose",
                               provider="yahoo", origin="1970-01-01",
                               compression="m", retclass="zoo")
index(FMAGX.prices) = as.yearmon(index(FMAGX.prices))

SBUX.prices = get.hist.quote(instrument="sbux", start="1998-01-01",
                              end="2009-12-31", quote="AdjClose",
                               provider="yahoo", origin="1970-01-01",
                               compression="m", retclass="zoo")
index(SBUX.prices) = as.yearmon(index(SBUX.prices))

# create merged price data
lab5Prices.z = merge(VBLTX.prices, FMAGX.prices, SBUX.prices)
```

```
# rename columns
colnames(lab5Prices.z) = c("VBLTX", "FMAGX", "SBUX")

# calculate cc returns as difference in log prices
lab5Returns.z = diff(log(lab5Prices.z))

#
# See the document "Working with Time Series in R" on the
# class webpage for more details on zoo objects
#

# look at the return data
start(lab5Returns.z)
end(lab5Returns.z)
colnames(lab5Returns.z)
head(lab5Returns.z)


###############################################################################
# Part I
###############################################################################

#
# 3. Create time plots of data
#

# 3 panel plot (each y axis has different scale)
# note: here, the generic plot() function invokes the plot method for objects
# of class zoo. See the help on plot.zoo
#
plot(lab5Returns.z,col="blue", lwd=2, main="Monthly cc returns on 3 assets")


# all on the same graph
plot(lab5Returns.z, plot.type="single", col=c("black","blue","red"), lwd=2,
     main="Monthly cc returns on 3 assets",
     ylab="Return")
legend(x="bottom", legend=colnames(lab5Returns.z), col=c("black","blue","red"), lwd=2)
abline(h=0)

# plot returns using the PerformanceAnalytics function chart.TimeSeries()
# this create a slightly nicer looking plot that plot.zoo()
?chart.TimeSeries
chart.TimeSeries(lab5Returns.z, legend.loc="bottom", main="")

# the previous charts are a bit hard to read. the PerformanceAnalytics function
# chart.Bar makes it easier to compare the returns of different assets on the
# same plot
?chart.Bar
chart.Bar(lab5Returns.z, legend.loc="bottom", main="")


# cumulative return plot - must use simple returns and not cc returns for this
# use PerformanceAnalytics function chart.CumReturns()
?chart.CumReturns
chart.CumReturns(diff(lab5Prices.z)/lag(lab5Prices.z, k=-1),
                 legend.loc="topleft", wealth.index=TRUE,
                 main="Future Value of $1 invested")
#
# 4. Create matrix of return data. some core R functions don't work
#    correctly with zoo objects
```

```
#

ret.mat = coredata(lab5Returns.z)
class(ret.mat)
colnames(ret.mat)
head(ret.mat)

#
# 5. Create graphical summaries of each data series
#

# online help on hist, boxplot, density, qqnorm
?hist
?boxplot
?density
?qqnorm

# here are the 4 panel plots
par(mfrow=c(2,2))
        hist(ret.mat[,"VBLTX"],main="VBLTX monthly returns",
             xlab="VBLTX", probability=T, col="slateblue1")
        boxplot(ret.mat[,"VBLTX"],outchar=T, main="Boxplot", col="slateblue1")
        plot(density(ret.mat[,"VBLTX"]),type="l", main="Smoothed density",
           xlab="monthly return", ylab="density estimate", col="slateblue1")
        qqnorm(ret.mat[,"VBLTX"], col="slateblue1")
        qqline(ret.mat[,"VBLTX"])
par(mfrow=c(1,1))

par(mfrow=c(2,2))
        hist(ret.mat[,"FMAGX"],main="FMAGX monthly returns",
             xlab="FMAGX", probability=T, col="slateblue1")
        boxplot(ret.mat[,"FMAGX"],outchar=T, main="Boxplot", col="slateblue1")
        plot(density(ret.mat[,"FMAGX"]),type="l", main="Smoothed density",
             xlab="monthly return", ylab="density estimate", col="slateblue1")
        qqnorm(ret.mat[,"FMAGX"], col="slateblue1")
        qqline(ret.mat[,"FMAGX"])
par(mfrow=c(1,1))

par(mfrow=c(2,2))
        hist(ret.mat[,"SBUX"],main="SBUX monthly returns",
             xlab="SBUX", probability=T, col="slateblue1")
        boxplot(ret.mat[,"SBUX"],outchar=T, main="Boxplot", col="slateblue1")
        plot(density(ret.mat[,"SBUX"]),type="l", main="Smoothed density",
             xlab="monthly return", ylab="density estimate", col="slateblue1")
        qqnorm(ret.mat[,"SBUX"], col="slateblue1")
        qqline(ret.mat[,"SBUX"])
par(mfrow=c(1,1))


# show boxplot of three series on one plot
boxplot(ret.mat[,"VBLTX"], ret.mat[,"FMAGX"], ret.mat[,"SBUX"],
        names=colnames(ret.mat), col="slateblue1")

# do the same thing using the PerformanceAnalytics function chart.Boxplot
chart.Boxplot(lab5Returns.z)

#
# 6. Compute univariate descriptive statistics
#

summary(ret.mat)
```

```
# compute descriptive statistics by column using the base R function apply()
# note: skewness and kurtosis are in the package PerformanceAnalytics
# note: kurtosis returns excess kurtosis

?apply
args(apply)
apply(ret.mat, 2, mean)
apply(ret.mat, 2, var)
apply(ret.mat, 2, sd)
apply(ret.mat, 2, skewness)
apply(ret.mat, 2, kurtosis)

# A nice PerformanceAnalytics function that computes all of the relevant
# descriptive statistics is table.Stats
?table.Stats
table.Stats(lab5Returns.z)

#
# 7. Annualize monthly estimates
#

# annualized cc mean
12*apply(ret.mat, 2, mean)

# annualized simple mean
exp(12*apply(ret.mat, 2, mean)) - 1

# annualized sd values
sqrt(12)*apply(ret.mat, 2, sd)

#
# 8. Compute bivariate descriptive statistics
#

# online help on pairs
?pairs
pairs(ret.mat, col="slateblue1", pch=16)

# online help on var and cor
?var
?cor

# compute 3 x 3 covariance and correlation matrices
var(ret.mat)
cor(ret.mat)

#
# 9. Compute time series diagnostics
#

# autocorrelations

# online help on acf
?acf

par(mfrow=c(3,1))
        acf.msft = acf(ret.mat[,"VBLTX"], main="VBLTX")
        acf.sbux = acf(ret.mat[,"FMAGX"], main="FMAGX")
        acf.sp500 = acf(ret.mat[,"SBUX"], main="SBUX")
par(mfrow=c(1,1))
```