```
#  descriptiveStatistics.r
#
#  script file for computing descriptive statistics
#
#       author: Eric Zivot
#       created: September 18, 2008
#       revision history:
# July 11, 2012
#   updated examples for Summer 2012
# July 14, 2011
#   updated example for Summer 2011
# October 20, 2009
#   updated examples for Fall 2009 class
# October 13, 2008
#
#       Core R functions used:
#
#       acf                             compute sample autocovariances or autocorrelations
#       apply                           apply function to rows or columns of matrix
#       args                            determine agruments of a function
#       boxplot                 compute boxplot
#       cbind                           combine data objects vertically
#       class                           determine class of object
#       colIds                  get column names from object
#       cor                             compute sample correlation matrix
#       density                 compute smoothed histogram
# ecdf      compute empirical CDF
#       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
#       par                             set graphics parameters
#       plot                            generic plot function
#       points                  add points to a graph
#       qqline                  add line to qq-plot
#       qqnorm                  qq-plot against normal distribution
#       qt                              compute quantiles of student t distribution
#       rlnorm                  generate random data from log-normal distribution
#       rt                              generate random data from student t distribution
#       scale                           standardize a vector of data
#       pnorm                           compute normal CDF
#       seq                             generate sequence of numbers
#       sort                            sort data
#       start                           get start date of time series
#       stdev                           compute sample standard deviation
#       ts.plot                 time series plot
#       var                             compute sample variance or covariance matrix
#       ?                               invoke help system
#
# R packages used
# PerformanceAnalytics
#   skewness    compute sample skewness
#   kurtosis    compute excess kurtosis
# tseries                       Time series and computational finance
#               get.hist.quote  load data from Yahoo!
#       zoo
#               plot.zoo                plot zoo object
# sn
#   rsn       simulate from skew-normal distribution
#   dsn       compute pdf of skew-normal distribution

# set options
options(digits=4, width=70)


# load packages
library(PerformanceAnalytics)
library("tseries")
library("zoo")
library(sn)

# get monthly adjusted closing price data on MSFT and SP500 from Yahoo
# using the tseries function get.hist.quote. Set sample to Jan 1998 through
# May 2012. 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

MSFT.prices = get.hist.quote(instrument="msft", start="1998-01-01",
                             end="2012-05-31", quote="AdjClose",
                             provider="yahoo", origin="1970-01-01",
                             compression="m", retclass="zoo")
class(MSFT.prices)
colnames(MSFT.prices)
start(MSFT.prices)
end(MSFT.prices)

# change date index class to yearmon
# index(MSFT.prices) = as.yearmon(index(MSFT.prices))
# note: x-axis on graphs do not look good if this is done

SP500.prices = get.hist.quote(instrument="^gspc", start="1998-01-01",
                             end="2012-05-31", quote="AdjClose",
                             provider="yahoo", origin="1970-01-01",
                             compression="m", retclass="zoo")
# change date index class to yearmon
# index(SP500.prices) = as.yearmon(index(SP500.prices))

# add column names
colnames(MSFT.prices) = "MSFT"
colnames(SP500.prices) = "SP500"

# create zoo object with both prices
MSFTSP500.prices = merge(MSFT.prices, SP500.prices)

#
# 2. compute cc returns
#

MSFT = diff(log(MSFT.prices))
SP500 = diff(log(SP500.prices))
MSFTSP500 = merge(MSFT,SP500)
colnames(MSFTSP500)

class(MSFT)
class(SP500)
class(MSFTSP500)

# convert zoo data to matrix data for non time series analysis
# many R functions do not have methods implemented for zoo objects
# and results may be unpredictable
MSFT.mat = coredata(MSFT)
colnames(MSFT.mat) = "MSFT"
rownames(MSFT.mat) = as.character(index(MSFT))

SP500.mat = coredata(SP500)
colnames(SP500.mat) = "SP500"
rownames(SP500.mat) = as.character(index(SP500))

MSFTSP500.mat = coredata(MSFTSP500)
colnames(MSFTSP500.mat) = c("MSFT","SP500")
rownames(MSFTSP500.mat) = as.character(index(MSFTSP500))

#
# 3. time plots
#

# look at help file for plot method for zoo objects
?plot.zoo

# plot individual prices in separate graphs
plot(MSFT.prices,main="Monthly closing price of MSFT",
     ylab="Price", lwd=2, col="blue")
plot(SP500.prices,main="Monthly closing price of SP500",
     ylab="Price", lwd=2, col="blue")
# plot individual prices in two panel graph
plot(MSFTSP500.prices, main="Adjusted Closing Prices",
     lwd=2, col="blue")

# put returns on the same plot in separate panels
# panel function for plot.zoo to add horizontal line at zero in each panel
my.panel <- function(...) {
  lines(...)
  abline(h=0)
}

plot(MSFTSP500, main="Monthly cc returns on MSFT and SP500",
     panel=my.panel, lwd=2, col="blue")

# put returns on same plot in one panel and add a horizontal line
plot(MSFTSP500, plot.type="single", main="Monthly cc returns on MSFT and SP500",
     col = c("red", "blue"), lwd=2)
abline(h=0)
legend(x="topleft", legend=c("MSFT","SP500"), col=c("red","blue"))

# use PerformanceAnalytics function chart.TimeSeries for nicer time series graphs
chart.TimeSeries(MSFT)
chart.TimeSeries(MSFTSP500)
par(mfrow=c(2,1))
  chart.TimeSeries(MSFT)
  chart.TimeSeries(SP500)
par(mfrow=c(1,1))

#
# 5. create simulated iid Gaussian data with same mean and SD
# as MSFT
#

set.seed(123)
gwn = rnorm(length(MSFT),mean=mean(MSFT),sd=sd(MSFT))
par(mfrow=c(2,1))
```

```
  plot(MSFT,main="Monthly cc returns on MSFT", lwd=2, col="blue")
  abline(h=0)
  ts.plot(gwn,main="Gaussian data with same mean and sd as MSFT", lwd=2, col="blue")
  abline(h=0)
par(mfrow=c(1,1))

#
# 6. histograms
#

# use hist() command
# note: hist() does not have a method for objects of class zoo
# must use coredata to extract data prior to using hist

args(hist)
?hist
hist(MSFT.mat,main="Histogram of MSFT monthly cc returns",
     col="slateblue1")
# scale histogram so that total area = 1
hist(MSFT.mat,main="Histogram of MSFT monthly returns",
     probability=TRUE, col="slateblue1")

# histogram of simulated Gaussian data
hist(gwn,main="Histogram of simulated Gaussian data", col="slateblue1")

# histogram of S&P 500 data
hist(SP500.mat,main="Histogram of SP500 monthly cc returns",
     col="slateblue1")

# plot both histograms on same page
# note different scales
par(mfrow=c(2,1))
  hist(MSFT.mat,main="Histogram of MSFT monthly returns", col="slateblue1")
  hist(SP500.mat,main="Histogram of SP500 monthly returns", col="slateblue1")
par(mfrow=c(1,1))

# try different layout
par(mfrow=c(1,2))
  hist(MSFT.mat,main="MSFT")
  hist(SP500.mat,main="SP500")
par(mfrow=c(1,1))

# use same breakpoints for both histograms
MSFT.hist = hist(MSFT.mat,plot=F,breaks=15)
class(MSFT.hist)
names(MSFT.hist)

par(mfrow=c(2,1))
  hist(MSFT.mat,main="MSFT", col="slateblue1", xlab="returns")
  hist(SP500.mat,main="SP500", col="slateblue1", xlab="returns",
       breaks=MSFT.hist$breaks)
par(mfrow=c(1,1))

# Use PerformanceAnalytics function chart.Histogram
chart.Histogram(MSFT)

#
# 7. smoothed histograms
#

# use density() command
```

```
?density
args(density)
MSFT.density = density(MSFT.mat)
class(MSFT.density)
names(MSFT.density)
MSFT.density
plot(MSFT.density,type="l",xlab="monthly cc return", col="orange", lwd=2,
     ylab="density estimate",main="Smoothed histogram")
# put histogram and density plot on same graph
hist(MSFT.mat,main="Histogram and smoothed density", col="slateblue1",
     probability=T, ylim=c(0,5))
points(MSFT.density,type="l", col="orange", lwd=2)

SP500.density = density(SP500)
plot(SP500.density,type="l",xlab="monthly return",
     ylab="density estimate",main="Smoothed histogram")

# combine density plots on one graph
hist(SP500.mat,main="Histogram and smoothed density",
     probability=T, ylim=c(0,10), col="slateblue1")
points(SP500.density,type="l", lwd=2, col="orange")

#
# 8. Empirical quantiles and summary statistics
#


# use quantile() function
?quantile
args(quantile)
# empirical quantiles for MSFT
quantile(MSFT.mat)
quantile(MSFT.mat,probs=c(0.01,0.05))
# compare to normal quantiles
qnorm(p=c(0.01,0.05), mean=mean(MSFT.mat), sd=sd(MSFT.mat))
# empirical and normal quantiles for SP500
quantile(SP500.mat,probs=c(0.01,0.05))
qnorm(p=c(0.01,0.05), mean=mean(SP500.mat), sd=sd(SP500.mat))

# note: empirical quantiles are useful for computing historical simulation value-at-risk (VaR)

#
# historical VaR
#

q.01 = quantile(MSFT.mat, probs=0.01)
q.05 = quantile(MSFT.mat, probs=0.05)
q.01
q.05
VaR.01 = 100000*(exp(q.01) - 1)
VaR.05 = 100000*(exp(q.05) - 1)
VaR.01
VaR.05

# compute median
median(MSFT.mat)


# compute interquartile range IQR
quantile(MSFT.mat,probs=0.75) - quantile(MSFT.mat,probs=0.25)
```

```
# use mean, var, sd, skewness and kurtosis functions
mean(MSFT.mat)
var(MSFT.mat)
sd(MSFT.mat)
skewness(MSFT.mat)
kurtosis(MSFT.mat)
# kurtosis function actually computes excess kurtosis
kurtosis(MSFT) + 3
# note: summary is a generic function with several methods
summary(MSFT.mat)

# Use apply to compute statistics for columns
apply(MSFTSP500.mat, 2, mean)
apply(MSFTSP500.mat, 2, sd)
apply(MSFTSP500.mat, 2, skewness)
apply(MSFTSP500.mat, 2, kurtosis)

#
# 9. empirical distribution function
#

# compute and plot empirical distribution function for simulated gaussian data
n1 = length(gwn)
plot(sort(gwn),(1:n1)/n1,type="s",ylim=c(0,1), col="slateblue1", lwd=2,
     main="Empirical CDF of Gaussian data", ylab="#x(i) <= x")

# compare empirical cdf to standard normal cdf for simulated gaussian data
z1 = scale(gwn)                 # standardize to have mean zero and sd 1
n1 = length(gwn)
F.hat = 1:n1/n1                 # empirical cdf
x1 = sort(z1)                           # sort from smallest to largest
y1 = pnorm(x1)                  # compute standard normal cdf at x

# The following plot options are used
# type          determine type of plot: "l" is line plot; "s" is step plot
# lty           line type: 1 is solid line; 3 is dot-dashed line
# lwd           line thickness: higher values give thicker lines
# col           line color: 1 is black, 2 is blue etc.
# For help on plot options, see help(par)

plot(x1,y1,main="Empirical CDF vs. Normal CDF for Gaussian data",
     type="l",lwd=2,xlab="standardized gwn",ylab="CDF")
points(x1,F.hat, type="s", lty=1, lwd=3, col="orange")
legend(x="topleft",legend=c("Normal CDF","Empirical CDF"),
       lty=c(1,1), lwd=2, col=c("black","orange"))

# compare empirical cdf to standard normal cdf for MSFT returns
z1 = scale(MSFT.mat)                    # standardize to have mean zero and sd 1
n1 = length(MSFT.mat)
F.hat = 1:n1/n1                 # empirical cdf
x1 = sort(z1)                           # sort from smallest to largest
y1 = pnorm(x1)                  # compute standard normal cdf at x

plot(x1,y1,main="Empirical CDF vs. Normal CDF for MSFT returns",
     type="l",lwd=2,xlab="standardized MSFT returns",ylab="CDF")
points(x1,F.hat, type="s", lty=1, lwd=3, col="orange")
legend(x="topleft",legend=c("Normal CDF","Empirical CDF"),
       lty=c(1,1), lwd=c(2,3), col=c("black","orange"))

# compare empirical cdf to standard normal cdf for SP500 returns
z1 = scale(SP500.mat)                   # standardize to have mean zero and sd 1
```

```
n1 = length(SP500.mat)
F.hat = 1:n1/n1                  # empirical cdf
x1 = sort(z1)                            # sort from smallest to largest
y1 = pnorm(x1)                   # compute standard normal cdf at x

plot(x1,y1,main="Empirical CDF vs. Normal CDF for SP500 returns",
     type="l",lwd=2,xlab="standardized SP500 returns",ylab="CDF")
points(x1,F.hat, type="s", lty=1, lwd=3, col="orange")
legend(x="topleft",legend=c("Normal CDF","Empirical CDF"),
       lty=c(1,1), lwd=c(2,3), col=c("black","orange"))

#
# 10. QQ plots: compare empirical quantiles to those from normal distribution
#

par(mfrow=c(2,2))       # 4 panel layout: 2 rows and 2 columns
        qqnorm(gwn, main="Gaussian data")
        qqline(gwn)
        qqnorm(MSFT.mat, main="MSFT")
        qqline(MSFT.mat)
        qqnorm(SP500.mat, main="SP500")
        qqline(SP500.mat)
par(mfrow=c(1,1))

# Data for student t with 3 df: tails fatter than normal
set.seed(123)
tdata = rt(100,df=3)             # Student-t with 3 df
gdata = rnorm(100)                       # N(0,1) data
xx = seq(from=-5,to=5,length=100)

# data for log-normal: asymmetric distribution
lndata = rlnorm(100)
yy = seq(from=-3, to = 3, length=100)

# data for skew normal: asymmetric distribution
skew.norm.data = rsn(100, shape = 2)


par(mfrow=c(2,2))
        # 1st plot
        plot(xx,dnorm(xx),type="l", lwd=2,
             main="Normal and Student-t with 3 df", xlab = "z, t", ylab = "pdf")
        points(xx,dt(xx,df=3), type="l", col="orange", lwd=3)
        legend(x="topright", legend=c("Normal","Student-t"), lty=c(1,1),
col=c("black","orange"),
               lwd=c(2,3))
        # 2nd plot
        plot(yy,dnorm(yy,sd=1),type="l", lwd=2, ylim=c(0,0.7),
             main="Normal and Skew-Normal", xlab = "z, skew-z", ylab = "pdf")
        points(yy,dsn(yy, shape=2), type="l", lwd=3, col="orange")
        legend(x="topleft", legend=c("Normal","Skew-Normal"), lty=c(1,1),
col=c("black","orange"),
               lwd=c(2,3))
        # 3rd plot
        qqnorm(tdata)
        qqline(tdata)
        # 4th plot
        qqnorm(skew.norm.data)
        qqline(skew.norm.data)
par(mfrow=c(1,1))
```

```
#
# 11. outliers
#

# create GWN return data polluted by outlier
gwn.new = gwn
gwn.new[20] = -0.9
par(mfrow=c(2,1))
        ts.plot(gwn.new,main="GWN polluted polluted by outlier", lwd=2, col="blue")
        abline(h=0)
        hist(gwn.new, main="", col="slateblue1")
par(mfrow=c(1,1))
# compare summary statistic for MSFT returns and returns polluted by outlier
tmp = cbind(gwn, gwn.new)
apply(tmp, 2, mean)
apply(tmp, 2, sd)
apply(tmp, 2, skewness)
apply(tmp, 2, kurtosis)

# outlier robust measures
apply(tmp, 2, median)
apply(tmp, 2, IQR)

#
# 12. boxplots
#
# use boxplot() function
?boxplot
args(boxplot)
boxplot(MSFT.mat,outchar=T,main="Boxplot of monthly cc returns on Microsoft",
        ylab="monthly cc return", col="slateblue1")
boxplot(SP500,outchar=T,main="Boxplot of monthly cc returns on SP500",
        ylab="monthly cc return")
boxplot(gwn,MSFT.mat,SP500.mat,names=c("gwn","MSFT","SP500"),outchar=T, col="slateblue1",
        main="Comparison of return distributions", ylab="monthly cc return")


#
# 13. graphically summarize data (see Smith notes and Carmona book)
#

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

#
# 14. bivariate graphical summaries
#

# bivariate scatterplot
plot(MSFT.mat,SP500.mat,main="Monthly cc returns on MSFT and SP500", col="slateblue1")
abline(h=mean(SP500.mat))       # horizontal line at SP500 mean
abline(v=mean(MSFT.mat))        # vertical line at MSFT mean
```

```
# all pairwise scatterplots
pairs(cbind(gwn,MSFT.mat,SP500.mat), col="slateblue1")

# compute covariance and correlation matrix
var(cbind(gwn,MSFT.mat,SP500.mat))
cor(cbind(gwn,MSFT.mat,SP500.mat))

#
# 15. Time series descriptive statistics
#

# sample autocovariances and autocorrelations

?acf
args(acf)
# autocorrelations for simulated gaussian data and SP500

acf(SP500.mat, lwd=2)
acf(MSFT.mat, lwd=2)
```