```
# probReview.r
#
# R examples for probability concepts chapter
#
# author: Eric Zivot
# created: Sept 27, 2008
# revision history:
# June 26, 2012
#   Modified log-normal example
# July 5, 2011
#   Update and added graphs
# June 28, 2011
#   Updated code for summer 2011
# October 13, 2009
#   Added examples for cdf of discrete distribution
# October 11, 2009
#   Added examples for bivariate distributions
# October 10, 2009
#   Added code for multivariate examples
# October 3, 2009
#   Added code for univariate examples
#
# Core R functions used:
#

# R packages/functions used
#
# mvtnorm
#   dmvnorm             density of multivariate normal
#   pmvnorm             CDF of multivariate normal
#   qmvnorm             quantiles of multivariate normal
#   rmvnorm             simulate random numbers from multivariate normal

# scatterplot3D
options(digits = 4)

##############################################################################
#
# univariate distributions
#
##############################################################################

#
# discrete distribution for msft
#

r.msft = c(-0.3, 0, 0.1, 0.2, 0.5)
prob.vals = c(0.05, 0.20, 0.50, 0.20, 0.05)
barplot(prob.vals, names.arg = as.character(r.msft), xlab="return")
title("Annual Return on Microsoft")
# alternatively create spike plot
plot(r.msft, prob.vals, type="h", lwd=4, xlab="return", ylab="probability")
points(r.msft, prob.vals, pch=16, cex=2, col="blue")
axis(1, at=r.msft)

# plot cdf of discrete distribution
cdf = c(0, 0.05, 0.25, 0.75, 0.95, 1)
x.vals = c(-0.4, -0.3, 0, 0.1, 0.2, 0.5)
plot(x.vals, cdf, lwd=4, type="S")
axis(1, at=x.vals)
```

```
#
# Uniform Distribution on [0,1]
#

# plot pdf
x.vals = seq(0,1,length=100)
plot(x.vals, dunif(x.vals), type="l",
     xlab="u", ylab="pdf", lwd=2, ylim=c(0,1))
segments(x0=0,y0=0,x1=0,y1=1, lwd=2)
segments(x0=1,y0=0,x1=1,y1=1, lwd=2)

# plot cdf
plot(x.vals, punif(x.vals), type="l")

# compute quantiles
a.vals = c(0.01,0.05, 0.10, 0.50)
q.vals = qunif(a.vals)
q.vals

# simulate 100 uniform random numbers

u = runif(100)
u[1:5]

#
# Standard normal distribution
#

# plot density
x.vals = seq(-4, 4, length=150)
plot(x.vals, dnorm(x.vals), type="l", lwd=2, col="blue", xlab="x", ylab="pdf")
# plot cdf
plot(x.vals, pnorm(x.vals), type="l", lwd=2, col="blue", xlab="x", ylab="CDF")

# plot density with shaded area showing Pr(-2 <= X <= 1)
lb = -2
ub = 1
x.vals = seq(-4, 4, length=150)
d.vals = dnorm(x.vals)
# plot normal density
plot(x.vals, d.vals, type="l", xlab="x", ylab="pdf")
i = x.vals >= lb & x.vals <= ub
# add shaded region between -2 and 1
polygon(c(lb, x.vals[i], ub), c(0, d.vals[i], 0), col="grey")

# compute quantiles
q.vals = qnorm(a.vals)
names(q.vals) = as.character(a.vals)
q.vals

# area under normal curve
# Pr(X >= 2) = Pr(X <= -2)
pnorm(-2)
# Pr(-1 <= X <= 2)
pnorm(2) - pnorm(-1)
# Pr(-1 <= X <= 1)
pnorm(1) - pnorm(-1)
# Pr(-2 <= X <= 2)
pnorm(2) - pnorm(-2)
# Pr(-3 <= X <= 3)
```

```
pnorm(3) - pnorm(-3)

#
# General normal distribution
#

# Example: R ~ N(0.01, 0.10)
mu.r = 0.01
sd.r = 0.1
x.vals = seq(-4, 4, length=150)*sd.r + mu.r
plot(x.vals, dnorm(x.vals, mean=mu.r, sd=sd.r), type="l", lwd=2,
     col="blue", xlab="x", ylab="pdf")
pnorm(-0.5, mean=0.01, sd=0.1)
pnorm(0, mean=0.01, sd=0.1)
1 - pnorm(0.5, mean=0.01, sd=0.1)
1 - pnorm(1, mean=0.01, sd=0.1)

a.vals = c(0.01, 0.05, 0.95, 0.99)
qnorm(a.vals, mean=0.01, sd=0.10)


# Example: risk-return tradeoff
mu.r = 0.02
sd.r = 0.10
x.vals = seq(-3, 3, length=150)*sd.r + mu.r
plot(x.vals, dnorm(x.vals, mean=mu.r, sd=sd.r), type="l", lwd=2,
     ylim=c(0, max(dnorm(x.vals, mean=0.01, sd=0.05))),
     col="black", xlab="x", ylab="pdf")
points(x.vals, dnorm(x.vals, mean=0.01, sd=0.05), type="l", lwd=2,
       col="blue", lty="dotted")
segments(0.02, 0, 0.02, dnorm(0.02, mean=0.02, sd=0.1), lwd=2)
segments(0.01, 0, 0.01, dnorm(0.01, mean=0.01, sd=0.05), lwd=2,
         col="blue", lty="dotted")
legend(x="topleft", legend=c("Amazon", "Boeing"), lwd=2,
       col=c("black", "blue"), lty=c("solid","dotted"))



# Example: why normal distribution is not appropriate for simple returns
# set mean = 0.05 and sd = 0.5
mu.x = 0.05
sd.x = 0.5
x.vals = seq(from=(mu.x - 3*sd.x), to=(mu.x + 3*sd.x), length = 100)

# plot density
plot(x.vals, dnorm(x.vals, mean=mu.x, sd=sd.x), type="l",
     main="N(0.05,(.50)^2)", xlab="R", ylab="pdf")

# area less than -1 for R(t)
pnorm(-1, mean=mu.x, sd=sd.x)

#
# log-normal distribution
#

# example: r(t) = ln(1 + R(t)) ~ N(0.05, (0.5)^2))
#          1 + R(t) = exp(r(t)) ~ logNormal(0.05, (0.5)^2)
#          R(t) = e(r(t)) - 1 ~ logNormal(0.5, (0.5)^2) - 1
#

# plot normal and log normal density
```

```
# plot density

plot(x.vals, dnorm(x.vals, mean=mu.x, sd=sd.x), type="l", lwd=2,
     xlim = c(min(x.vals), max(exp(x.vals))),
     ylim = c(0, 0.9),
     xlab="R", ylab="pdf")
points(exp(x.vals)-1, dlnorm(exp(x.vals), mean=mu.x, sd=sd.x), type="l",
       lwd = 2, col="blue", lty="dashed",
     xlab="1 + R", ylab="density", main="Log-Normal(0.05, (0.5)^2)")
legend(x="topright", legend=c("Normal", "Log-Normal"), lwd=2,
       col=c("black", "blue"), lty = c("solid", "dashed"))

# prob that 1 + R < 0
plnorm(0, 0.05, 0.5)

# Prob(r(t) < -2) = Prob(1 + R(t) < exp(-2))

pnorm(-2, 0.05, 0.50)
plnorm(exp(-2), 0.05, 0.50)

#
# create pdfs with positive, zero and negative skewness
#


#
# student's t distribution
#

# plot Student's t densities with differen degrees of freedom
x.vals = seq(-5, 5, length=100)
ub = max( c(dt(0, df=c(1,5,10,60)), dnorm(0)) )

plot(x.vals, dt(x.vals, df=1), type="l", lwd=2, ylim=c(0, ub), ylab="pdf", xlab="x")
points(x.vals, dt(x.vals, df=5), type="l", lwd=2, lty=2, col=2)
points(x.vals, dt(x.vals, df=10), type="l", lwd=2, lty=3, col=3)
points(x.vals, dt(x.vals, df=60), type="l", lwd=2, lty=4, col=4)
legend(x="topleft", legend=paste("v=", c(1,5,10,60), sep=""), lty=1:4, col=1:4,
       lwd=2)


# example: compute quantiles from student t
v = c(1, 2, 5, 10, 60, 100, Inf)
qt(0.01, df=v)
qnorm(0.01)
pt(-3, df=v)
pnorm(-3)

#
# create graphs with positive, zero and negative excess kurtosis
#


#
# Value-at-Risk calculations
#

# R(t) ~ N(0.05, (0.10)^2)
# W = 10000
w0 = 10000
```

```
# plot return and wealth distributions
mu.R = 0.05
sd.R = 0.1
R.vals = seq(from=(mu.R - 3*sd.R), to=(mu.R + 3*sd.R), length = 100)
mu.w1 = 10500
sd.w1 = 1000
w1.vals = seq(from=(mu.w1 - 3*sd.w1), to=(mu.w1 + 3*sd.w1), length = 100)

par(mfrow=c(2,1))
# plot return density
plot(R.vals, dnorm(R.vals, mean=mu.R, sd=sd.R), type="l",
     main="R(t) ~ N(0.05,(.10)^2)", xlab="R", ylab="pdf",
     lwd=2, col="blue")
# plot wealth density
plot(w1.vals, dnorm(w1.vals, mean=mu.w1, sd=sd.w1), type="l",
     main="W1 ~ N(10,500,(1,000)^2)", xlab="W1", ylab="pdf",
     lwd=2, col="blue")
par(mfrow=c(1,1))



# Pr(W1 < 9000)
pnorm(9000, mu.w1, sd.w1)
qnorm(pnorm(9000, mu.w1, sd.w1), mu.w1, sd.w1)

# compute 5% quantile of return and wealth distributions
q.R.05 = qnorm(0.05, mu.R, sd.R)
q.R.05
q.w1.05 = qnorm(0.05, mu.w1, sd.w1)
q.w1.05

# compute 5% VaR using return quantile
w0*q.R.05

# compute 5% VaR using wealth quantile
q.w1.05 - w0

# plot return and loss distributions with VaR
loss.vals = w0*R.vals
mu.loss = w0*mu.R
sd.loss = w0*sd.R
par(mfrow=c(2,1))
# plot return density
plot(R.vals, dnorm(R.vals, mean=mu.R, sd=sd.R), type="l",
     main="R(t) ~ N(0.05,(.10)^2)", xlab="R", ylab="pdf",
     lwd=2, col="blue")
abline(v=q.R.05, lwd=2, col="red")
# plot wealth density
plot(loss.vals, dnorm(loss.vals, mean=mu.loss, sd=sd.loss), type="l",
     main="R*W0 ~ N(500,(1,000)^2)", xlab="W0*R", ylab="pdf",
     lwd=2, col="blue")
abline(v=q.R.05*w0, lwd=2, col="red")
par(mfrow=c(1,1))


# VaR example
mu.R = 0.05
sd.R = 0.10
w0 = 10000
q.01.R = mu.R + sd.R*qnorm(0.01)
q.05.R = mu.R + sd.R*qnorm(0.05)
```

```
VaR.01 = abs(q.01.R*w0)
VaR.05 = abs(q.05.R*w0)
VaR.01
VaR.05

mu.r = 0.05
sd.r = 0.10
q.01.R = exp(mu.r + sd.r*qnorm(0.01)) - 1
q.05.R = exp(mu.r + sd.r*qnorm(0.05)) - 1
VaR.01 = abs(q.01.R*w0)
VaR.05 = abs(q.05.R*w0)
VaR.01
VaR.05

##############################################################################
#
# bivariate distributions
#
##############################################################################

# discrete distributions
x.probs = matrix(c(1/8,3/8,3/8,1/8,
                   2/8,4/8,2/8,0,
                   0,2/8,4/8,2/8), 4, 3)
y.probs = matrix(c(1/2,1/2,
                   1, 0,
                   2/3,1/3,
                   1/3,2/3,
                   0,1), 2, 5)
x.vals = 0:3
y.vals = 0:1

mu.x = sum(x.probs[,1] *x.vals)
mu.x.y0 = sum(x.probs[,2] *x.vals)
mu.x.y1 = sum(x.probs[,3] *x.vals)

var.x = sum(x.probs[,1] *x.vals^2) - mu.x^2
var.x.y0 = sum(x.probs[,2] *x.vals^2) - mu.x.y0^2
var.x.y1 = sum(x.probs[,3] *x.vals^2) - mu.x.y1^2

mu.y = sum(y.probs[,1] *y.vals)
mu.y.x0 = sum(y.probs[,2] *y.vals)
mu.y.x1 = sum(y.probs[,3] *y.vals)
mu.y.x2 = sum(y.probs[,4] *y.vals)
mu.y.x3 = sum(y.probs[,5] *y.vals)

var.y = sum(y.probs[,1] *y.vals^2) - mu.y^2
var.y.x0 = sum(y.probs[,2] *y.vals^2) - mu.y.x0^2
var.y.x1 = sum(y.probs[,3] *y.vals^2) - mu.y.x1^2
var.y.x2 = sum(y.probs[,4] *y.vals^2) - mu.y.x2^2
var.y.x3 = sum(y.probs[,5] *y.vals^2) - mu.y.x3^2

# plot regression function of y on x

plot(x.vals, c(mu.y.x0, mu.y.x1, mu.y.x2, mu.y.x3), type="b", lwd=2,
     xlab = "x", ylab="E[Y|X=x]")

# load mvtnorm package with functions for multivariate normal distribution
library(mvtnorm)

# plot bivariate standard normal distribution
```

```
# specify grid of x-y values
x = seq(-3, 3, length=50)
y = seq(-3, 3, length=50)
sigma = matrix(c(1, 0, 0, 1), 2, 2)
# function to evaluate bivariate normal pdf on grid
bv.norm <- function(x, y, sigma) {
  z = cbind(x,y)
  return(dmvnorm(z, sigma=sigma))
}
# use outer function to evaluate pdf on 2D grid of x-y values
fxy = outer(x, y, bv.norm, sigma)
persp(x, y, fxy, theta=60, phi=30, expand=0.5, ticktype="detailed",
      zlab="f(x,y)", col="blue")

# plot simulated data from bivariate standard normal
set.seed(123)
norm.dat = rmvnorm(200, sigma=sigma)
plot(norm.dat, pch=16, col="blue", xlab="x", ylab="y")
abline(h=0,v=0)

# compute area under bivariate standard normal distribution
# Finc P( -1 < X < 1 and -1 < Y < 1)

pmvnorm(lower=c(-1, -1), upper=c(1, 1))

# create 6 scatterplots to illustrate covariance and correlation
par(mfrow=c(3,2))
set.seed(123)
sigma = matrix(c(1, 0, 0, 1), 2, 2,)
x.mat = rmvnorm(20, sigma=sigma)
x = x.mat[, 1]
y = x.mat[, 2]
plot(x, y, pch = 16, main="(a)")
# abline(h = 0, v = 0, lwd = 2)
# panel (b): cor(x,y) = 1
sigma = matrix(c(1, 1, 1, 1), 2, 2,)
x.mat = rmvnorm(20, sigma=sigma)
x = x.mat[, 1]
y = x.mat[, 2]
plot(x, y, pch = 16, main="(b)")
#abline(h = 0, v = 0, lwd = 2)
# panel (c): cor(x,y) = -1
sigma = matrix(c(1, -1, -1, 1), 2, 2,)
x.mat = rmvnorm(20, sigma=sigma)
x = x.mat[, 1]
y = x.mat[, 2]
plot(x, y, pch = 16, main="(c)")
#abline(h = 0, v = 0, lwd = 2)
# cor(x,y) > 0
sigma = matrix(c(1, 0.75, 0.75, 1), 2, 2,)
x.mat = rmvnorm(20, sigma=sigma)
x = x.mat[, 1]
y = x.mat[, 2]
plot(x, y, pch = 16, main="(d)")
# cor(x,y) > 0
sigma = matrix(c(1, -0.75, -0.75, 1), 2, 2,)
x.mat = rmvnorm(20, sigma=sigma)
x = x.mat[, 1]
y = x.mat[, 2]
plot(x, y, pch = 16, main="(e)")
# nonlinear
```

```
sigma = matrix(c(0.5, 0, 0, 0.1), 2, 2,)
x.mat = rmvnorm(20, sigma=sigma)
x = x.mat[, 1]
y = x^2 + x.mat[, 2]
plot(x, y, pch = 16, main="(f)")
par(mfrow=c(1,1))


# Create plot to illustrate covariance calculation
sigma = matrix(c(1, 0.7, 0.7, 1), 2, 2,)
x.mat = rmvnorm(20, sigma=sigma)
x = x.mat[, 1]
y = x.mat[, 2]
# define 4 quandrants
q1 = (x < 0 & y > 0)
q2 = (x > 0 & y > 0)
q3 = (x < 0 & y < 0)
q4 = (x > 0 & y < 0)
# vector of plotting symbols
pch.vals = rep(16, 20)    # filled circles
pch.vals[q2] = 15         # filled squares
pch.vals[q3] = 17         # filled triangles
pch.vals[q4] = 18         # filled diamonds
col.vals = rep("black", 20)
col.vals[q2] = "blue"
col.vals[q3] = "red"
col.vals[q4] = "green"
plot(x, y, cex = 1.5, pch=pch.vals, col=col.vals,
     main = "Cov(x, y) > 0",
     xlab = expression(x - mu[x]), ylab = expression(y - mu[y]))
abline(h=0, v=0, lwd=2)
# 1st quadrant
text(-2, 1.6, labels="QI")
text(-1.8, 1.45, labels=expression(x - mu[x] < 0))
text(-1.8, 1.3, labels=expression(y - mu[y] > 0))
# 2nd quadrant
text(0.1, 1.6, labels="QII")
text(0.3, 1.45, labels=expression(x - mu[x] > 0))
text(0.3, 1.3, labels=expression(y - mu[y] > 0))
# 3rd quadrant
text(-2, -0.2, labels="QII")
text(-1.8, -0.35, labels=expression(x - mu[x] < 0))
text(-1.8, -0.5, labels=expression(y - mu[y] < 0))
# 4th quadrant
text(0.13, -0.2, labels="QIV")
text(0.3, -0.35, labels=expression(x - mu[x] > 0))
text(0.3, -0.5, labels=expression(y - mu[y] < 0))


# Example: plot bivariate normal

# specify grid of x-y values
x = seq(-3, 3, length=50)
y = seq(-3, 3, length=50)
sigma = matrix(c(1, 0.5, 0.5, 1), 2, 2)
mu = c(1,1)
# function to evaluate bivariate normal pdf on grid
bv.norm <- function(x, y, mu, sigma) {
  z = cbind(x,y)
  return(dmvnorm(z, mean=mu, sigma=sigma))
}
```

```
# use outer function to evaluate pdf on 2D grid of x-y values
fxy = outer(x, y, bv.norm, mu, sigma)
persp(x, y, fxy, theta=60, phi=30, expand=0.5, ticktype="detailed",
      zlab="f(x,y)", col="blue")

set.seed(123)
norm.dat = rmvnorm(200, mean=mu, sigma=sigma)
plot(norm.dat, pch=16, col="blue", xlab="x", ylab="y")
abline(h=0,v=0)
```