```
# timeSeriesConcepts.r
#
# R commands to produce graphs for powerpoint document
# timeSeriesConceptsPowerPoint.ppt and reproduce examples in
# the chapter Time Series Concepts.
#
# author: Eric Zivot
# created: October 15, 2004
# revision history:
# July 7, 2011
#   Added new examples
# October 18, 2010
#   Updated examples
# October 17, 2009
#   Updated for examples in the Time Series Concepts chapter
# October 5, 2008
#
# base R functions used
#
# abline                    draw horizontal line
# arima.sim               simulate from ARIMA model
# ARMAacf                   true ACF for ARMA model
# cumsum                    compute cumulative sums
# for                         start for loop
# layout                    partition screen into parts
# par                         set graphics defaults
# plot                  create default xy-plot
# rep                         repeat a sequence of values
# rnorm                 generate iid normal random variables
# set.seed                set random number seed
# ts.plot                   create time series plot
#
# R package functions used
#

options(digits=4, width=70)


# simulate Gaussian White Noise process
set.seed(123)
y = rnorm(250)
ts.plot(y,main="Gaussian White Noise Process",xlab="time",ylab="y(t)",
        col="blue", lwd=2)
abline(h=0)

# equivalent plot using plot()
plot(y, main="Gaussian White Noise Process", type="l", xlab="time",ylab="y(t)",
        col="blue", lwd=2)
abline(h=0)

# simulate Gaussian White Noise process for cc returns
# added July 7, 2011
set.seed(123)
y = rnorm(60, mean=0.01, sd=0.05)
ts.plot(y,main="GWN Process for Monthly Continuously Compounded Returns",
        xlab="time",ylab="r(t)", col="blue", lwd=2, type="h")
abline(h=c(0,-0.05,0.05), lwd=2, lty=c("solid","dotted","dotted"),
       col=c("black", "red", "red"))
```

```
# equivalent plot using plot()
plot(y, main="Gaussian White Noise Process", type="l", xlab="time",ylab="y(t)",
        col="blue", lwd=2)
abline(h=0)

#
# simultate deterministically trending process
#
set.seed(123)
e = rnorm(250)
y.dt = 0.1*seq(1,250) + e
ts.plot(y.dt, lwd=2, col="blue", main="Deterministic Trend + Noise")
abline(a=0, b=0.1)

#
# simulate random walk random walk
#

set.seed(321)
e = rnorm(250)
y.rw = cumsum(e)
ts.plot(y.rw, lwd=2, col="blue", main="Random Walk")
abline(h=0)


# simulate MA(1) process with theta 0.9 and e(t) ~ N(0,1)
ma1.model = list(ma=0.9)
mu = 1
set.seed(123)
ma1.sim = mu + arima.sim(model=ma1.model,n=250)

# simulate MA(1) process with theta 0.9 and e(t) ~ N(0,(0.1)^2)
set.seed(123)
ma1.sim2 = mu + arima.sim(model=ma1.model, n=250, innov=rnorm(n=250, mean=0, sd=0.1))

# ACF for MA(1) model
ma1.acf = ARMAacf(ar=0, ma=0.9, lag.max=10)
ma1.acf

par(mfrow=c(2,1))
ts.plot(ma1.sim,main="MA(1) Process: mu=1, theta=0.9",
xlab="time",ylab="y(t)", col="blue", lwd=2)
abline(h=c(0,1))
plot(0:10, ma1.acf,type="h", col="blue", lwd=2,
main="ACF for MA(1): theta=0.9",xlab="lag",ylab="rho(j)")
abline(h=0)
par(mfrow=c(1,1)

# simulate MA(1) process with theta < 0
ma1.model = list(ma=-0.75)
mu = 1
set.seed(123)
ma1.sim = mu + arima.sim(model=ma1.model,n=250)
ts.plot(ma1.sim,main="MA(1) Process: mu=1, theta=-0.75",
xlab="time",ylab="y(t)", col="blue", lwd=2)
abline(h=0)

# ACF for MA(1) model
ma1.acf = ARMAacf(ar=0, ma=-0.75, lag.max=10)
plot(0:10, ma1.acf,type="h", col="blue", lwd=2,
main="ACF for MA(1): theta=-0.75",xlab="lag",ylab="rho(j)")
```

```
abline(h=0)


# simulate AR(1) process: phi = 0.9
ar1.model = list(ar=0.9)
mu = 1
set.seed(123)
ar1.sim = mu + arima.sim(model=ar1.model,n=250)
ar1.acf = ARMAacf(ar=0.9, ma=0, lag.max=10)

par(mfrow=c(2,1))
ts.plot(ar1.sim,main="AR(1) Process: mu=1, phi=0.9",
xlab="time",ylab="y(t)", col="blue", lwd=2)
abline(h=0)
abline(h=1)
# ACF for AR(1) model
plot(0:10, ar1.acf,type="h", col="blue", lwd=2,
main="ACF for AR(1): phi=0.9",xlab="lag",ylab="rho(j)")
abline(h=0)
par(mfrow=c(1,1))

# simulate AR(1) process: phi = -0.75
ar1.model = list(ar=-0.75)
mu = 1
set.seed(123)
ar1.sim = mu + arima.sim(model=ar1.model,n=250)

# ACF for AR(1) model
ar1.acf = ARMAacf(ar=-0.75, ma=0, lag.max=10)

par(mfrow=c(2,1))
ts.plot(ar1.sim,main="AR(1) Process: mu=1, phi=-0.75",col="blue", lwd=2,
xlab="time",ylab="y(t)")
abline(h=0)
plot(0:10, ar1.acf,type="h", col="blue", lwd=2,
main="ACF for AR(1): phi=-0.75",xlab="lag",ylab="rho(j)")
abline(h=0)

par(mfrow=c(1,1))

# simulate AR(1) process: phi = 0.99
ar1.model = list(ar=0.99)
mu = 1
set.seed(123)
ar1.sim = mu + arima.sim(model=ar1.model,n=250)

# ACF for AR(1) model
ar1.acf = ARMAacf(ar=0.99, ma=0, lag.max=10)

par(mfrow=c(2,1))
ts.plot(ar1.sim,main="AR(1) Process: mu=1, phi=0.99",
xlab="time",ylab="y(t)")
abline(h=0)
plot(0:10, ar1.acf,type="h", col="blue", lwd=2,
main="ACF for AR(1): phi=0.99",xlab="lag",ylab="rho(j)")
abline(h=0)
par(mfrow=c(1,1))

# simulate AR(1) process: phi = 1
set.seed(123)
ar1.sim = cumsum(rnorm(250))
```

```
# simulate AR(1) process: phi > 1
set.seed(123)
phi = 1.01
e = rnorm(250)
y = rep(0,250)
for (i in 2:250) {
        y[i] = phi*y[i-1] + e[i]
}

par(mfrow=c(2,1))
ts.plot(ar1.sim,main="AR(1) Process: phi=1",
xlab="time",ylab="y(t)",lwd=2, col="blue")
abline(h=0)
ts.plot(y,main="AR(1) Process: phi=1.01",
xlab="time",ylab="y(t)", lwd=2, col="blue")
abline(h=0)
par(mfrow=c(1,1))

# do same plot but use layout() function
layout(matrix(c(1,2,1,2), 2, 2))
ts.plot(ar1.sim,main="AR(1) Process: phi=1",
xlab="time",ylab="y(t)")
abline(h=0)
ts.plot(y,main="AR(1) Process: phi=1.01",
xlab="time",ylab="y(t)")
abline(h=0)
```