CSSS 512: Lab 2

Temporal Concepts: Trends, Stochastic Processes, and Seasonality

2018-4-13

Agenda

Box-Jenkins Method

Steps:

1. Study generic forms and properties
2. Study these realizations for an indication of which possibly applies to your data
3. Assess your guess–diagnose and iterate
4. Perform a meta-analysis at the end to determine which specification is best

The Box-Jenkins method assumes that time series are composed by multiple temporal processes. It then performs diagnostics to compare the observed series with generic forms to decide what processes occur in the data (i.e. the DGP)

We will cover the first three steps in this lab.

Deterministic Trends

$$y_t = \beta_0 + t\beta_1 + e_t$$
$$e_t \sim \mathcal{N}(0, \sigma^2)$$

- Each period entails another β_1 increase in $\mathbb{E}(y_t)$
- Time has a purely systematic relationship with y
- Once the time series is detrended, it is simply *white noise*

$$y_t - t\hat{\beta}_1 = \hat{\beta}_0 + \hat{e}_t$$
$$\hat{e}_t \sim \mathcal{N}(0, \hat{\sigma}^2)$$

- White noise is normally distributed with mean zero and constant variance

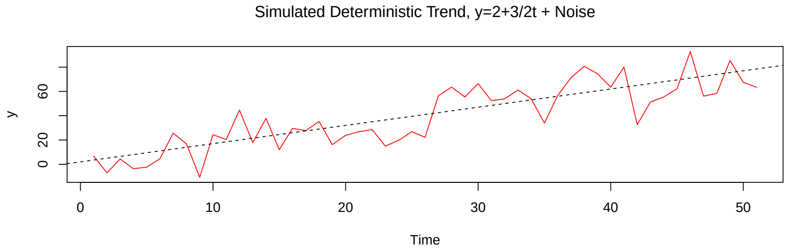
Deterministic Trends

Simulate a deterministic trend with noise, de-trend the data, and plot the time series.

```r
# Set the slope of the trend
b1 <- 3/2

#Set the intercept
b0 <- 2

#Set the number of periods
n <- 50
t <- seq(0,n)
y <- rep(0,n)

#Simulate the data
for (i in 1:length(t)){
    y[i] <- b1*t[i] + b0 + rnorm(1,0,15)
    #The rnorm gives us the noise with mean 0, variance 15
}
```

Deterministic Trends

```
#Plot the data
par(mfrow=c(2,1))
plot(y,type="l", col="red",ylab="y",xlab="Time",
     main=expression(paste
    ("Simulated Deterministic Trend, y=2+3/2t + Noise")))
abline(a=2,b=3/2,lty="dashed")
```

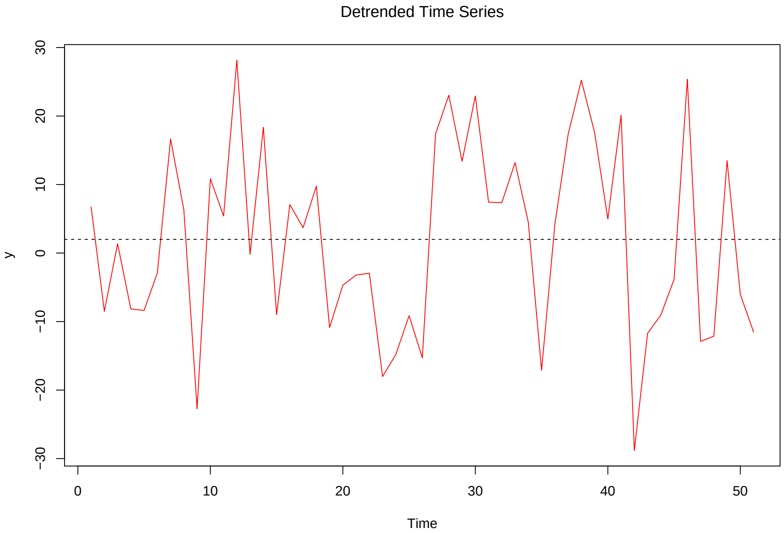


Deterministic Trends

```r
#Now de-trend the time series
y.minus.tbeta <- rep(0,n)
for (i in 1:length(t)){
    y.minus.tbeta[i] <- y[i] - b1*t[i]
}
```

Deterministic Trends

```r
#Plot and take a minute to inspect the residuals
plot(y.minus.tbeta,type="l", col="red",ylab="y",xlab="Time",
     main=expression(paste("Detrended Time Series"))); abline(a=2,b=0,lty="dashed")
```



Deterministic Trends

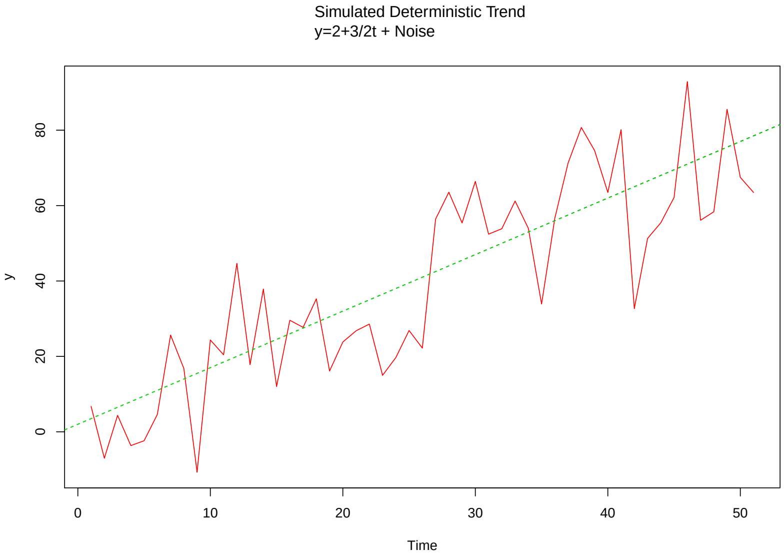
```
#Find the least squares estimate of the slope
slope1 <- lm(y~t)
slope1
```

```
##
## Call:
## lm(formula = y ~ t)
##
## Coefficients:
## (Intercept)            t
##       1.996        1.499
```

```
#How does it compare to the true beta?
#Plot the data with the true beta and the estimated beta
```

Deterministic Trends

```
plot(y,type="l", col="red",ylab="y",xlab="Time",main=expression(paste("Simulated Deterministic Trend
y=2+3/2t + Noise"))); abline(a=2,b=3/2,lty="dashed")
abline(a=slope1$coefficients[1],b=slope1$coefficients[2],lty="dashed",col="green")
```



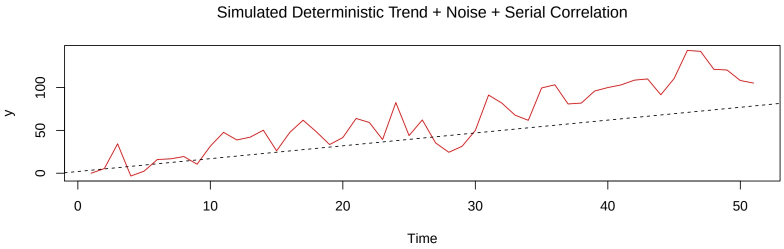
Deterministic Trends and Serial Correlation

Simulate new data with a deterministic trend and serial correlation

```r
#Set the slope
b1 <- 3/2

#Set the intercept
b0 <- 2

#Set phi
phi <- 0.33

#Set the number of periods
n <- 50
t <- seq(0,n)
y <- rep(0,n)

for (i in 2:length(t)){
    y[i] <- y[i-1]*phi + b1*t[i] + b0 + rnorm(1,0,15)
}
```

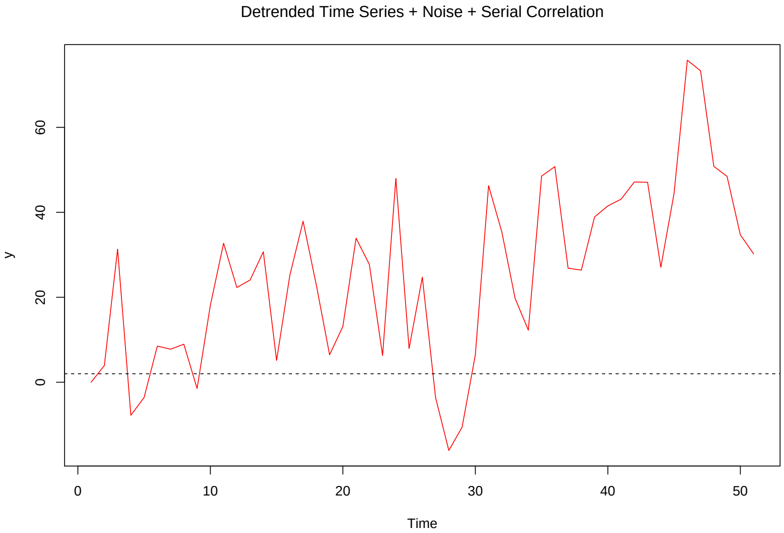
Deterministic Trends and Serial Correlation

```
#Plot the data and also the de-trended time series
par(mfrow=c(2,1))
plot(y,type="l", col="red",ylab="y",xlab="Time",
     main=expression(paste
("Simulated Deterministic Trend + Noise + Serial Correlation")))
abline(a=2,b=3/2,lty="dashed")

y.minus.tbeta2 <- rep(0,n)
for (i in 1:length(t)){
    y.minus.tbeta2[i] <- y[i] - b1*t[i]
}
```



Deterministic Trends and Serial Correlation

```r
#Plot the data and take a minute to inspect the residuals again
plot(y.minus.tbeta2,type="l", col="red",ylab="y",xlab="Time",main=expression(
  paste("Detrended Time Series + Noise + Serial Correlation")));abline(a=2,b=0,lty="dashed")
```

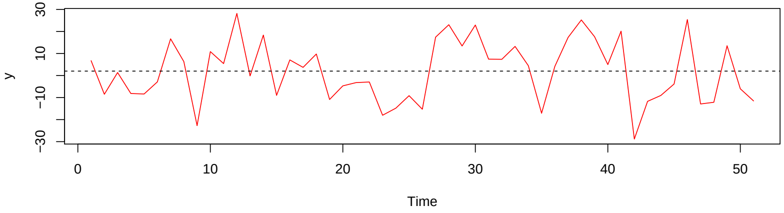


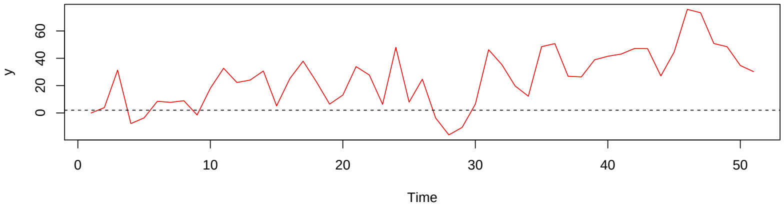
Deterministic Trends and Serial Correlation

1. Compare the two sets of plots and discuss the differences between a deterministic trend and stochastic process.
2. What are some issues that can arise when analyzing de-trended time series data using regression?

Deterministic Trends and Serial Correlation

```r
par(mfrow=c(2,1))
plot(y.minus.tbeta,type="l", col="red",ylab="y",xlab="Time");abline(a=2,b=0,lty="dashed")
plot(y.minus.tbeta2,type="l", col="red",ylab="y",xlab="Time");abline(a=2,b=0,lty="dashed")
```





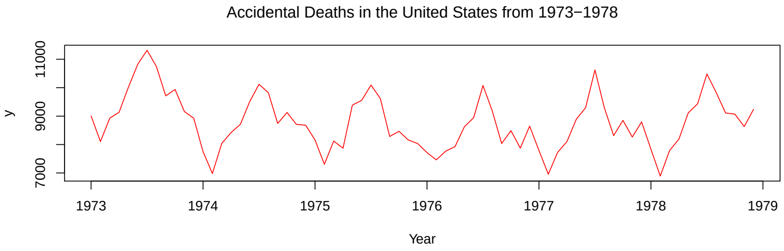
Seasonality

- any cyclical fluctuation in a time series that recurs or repeats itself at the same phase of the cycle
- y_t is an additive or multiplicative function of y_{t-c} for some fixed cycle c (e.g. $c = 12$ for months)
- additive seasonality: corresponding months in different years share a level component
- multiplicative seasonality: corresponding months in different years related by a factor change

Seasonality

Accidental Deaths in the United States from 1973-1978, (from P. J. Brockwell and R. A. Davis (1991))

```
accidents <- read.csv("USAccDeaths.csv",header=TRUE)
attach(accidents)
par(mfrow=c(2,1))
plot(time, USAccDeaths, type="l",
col="red",ylab="y",xlab="Year", main = expression(
paste("Accidental Deaths in the United States from 1973-1978")))
```

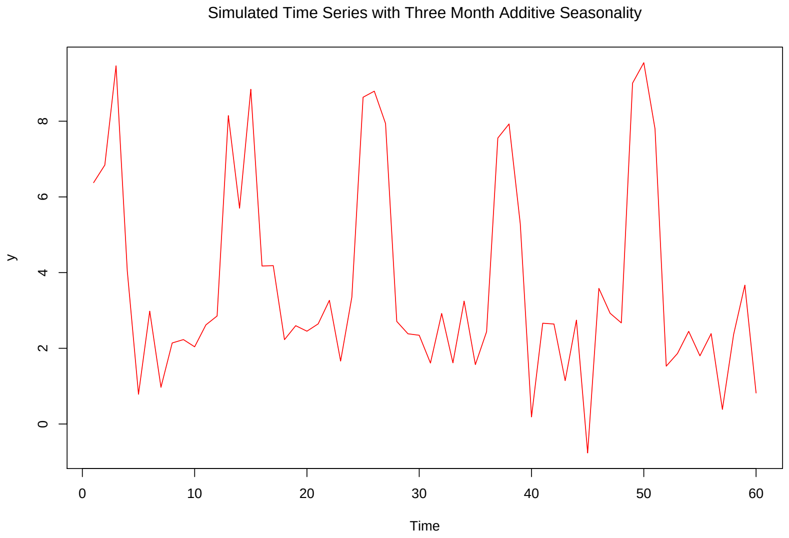


Seasonality

```r
#Simulate a time series with seasonal variation
#Assume the data is de-trended
b1 <- 0
#Set the intercept
b0 <- 2
#Set the number of periods
n <- 60          #Assume a one month period for 5 years
t <- seq(0,n)
y <- rep(0,n)
#Simulate the data
for (i in 1:n){
    y[i] <- b1*t[i] + b0 + rnorm(1,0,1)
}
#Introduce additive seasonality during the first three months of each year
a <- seq(1,60, by=12)
b <- seq(2,60, by=12)
c <- seq(3,60, by=12)
q <- sort(c(a,b,c))
for (i in q){
    y[i] <- y[i]+6 #Seasonality can be additive or multiplicative
}
```

Seasonality

```r
#Plot the data
plot(y,type="l",col="red",ylab="y",xlab="Time",
main = expression(paste("Simulated Time Series with Three Month Additive Seasonality")))
```



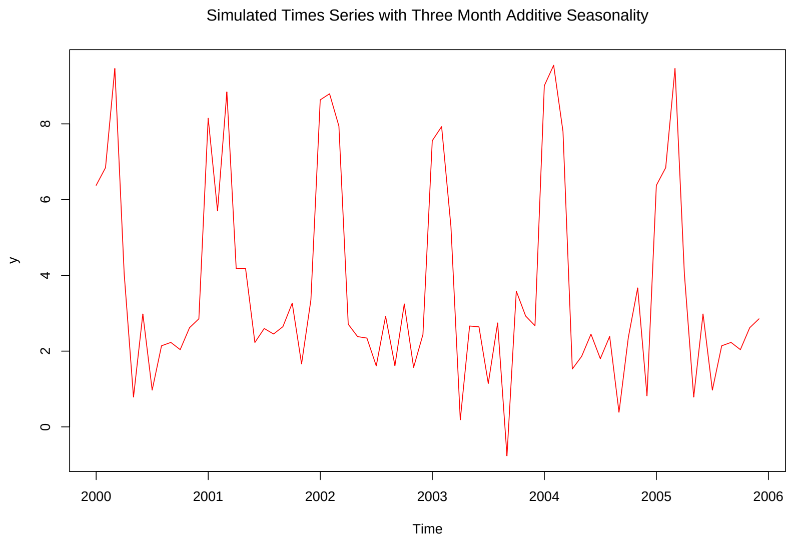
Seasonality

```r
#R has a special class of objects that corresponds to time series data
#The ts function allows for you to create a time series object, use help(ts) for reference
ts.1 <- ts(y, start=c(2000,1), end=c(2005,12), frequency=12)
#We are creating a time series of length 60 months that starts from Jan 2000 until Dec 2005
help(ts)
ts.1
```

```
##            Jan       Feb       Mar       Apr       May       Jun
## 2000 6.3761137 6.8423562 9.4620279 4.0463741 0.7879327 2.9807032
## 2001 8.1477820 5.7029378 8.8433553 4.1747984 4.1842872 2.2290593
## 2002 8.6333985 8.7931906 7.9392746 2.7103501 2.3831615 2.3446237
## 2003 7.5552225 7.9270708 5.2861032 0.1875826 2.6629234 2.6416772
## 2004 9.0059949 9.5456131 7.7974050 1.5283635 1.8627545 2.4483092
## 2005 6.3761137 6.8423562 9.4620279 4.0463741 0.7879327 2.9807032
##            Jul       Aug       Sep       Oct       Nov       Dec
## 2000 0.9715414 2.1407616 2.2300959 2.0397135 2.6181026 2.8537946
## 2001 2.5976124 2.4518413 2.6453466 3.2668540 1.6625305 3.3515326
## 2002 1.6106276 2.9195508 1.6165241 3.2468827 1.5694207 2.4388786
## 2003 1.1467976 2.7441896 -0.7646506 3.5833033 2.9264727 2.6717457
## 2004 1.8012018 2.3875689 0.3869596 2.3638356 3.6685765 0.8190498
## 2005 0.9715414 2.1407616 2.2300959 2.0397135 2.6181026 2.8537946
```

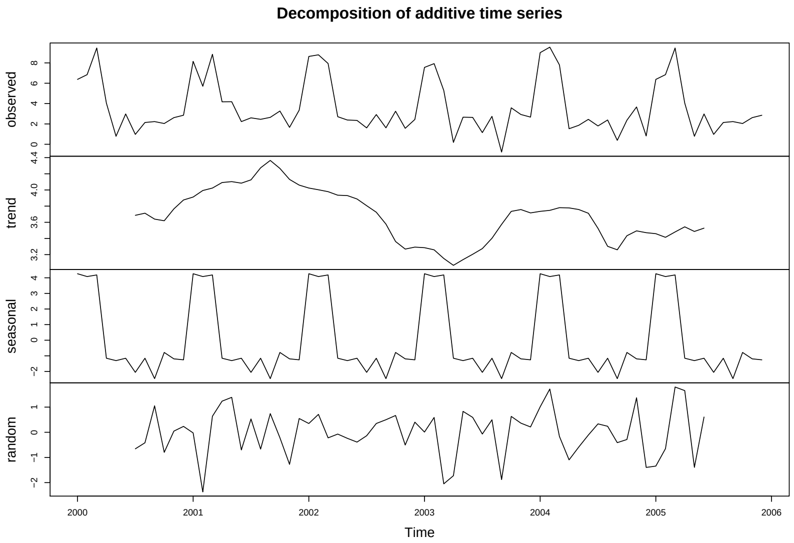
Seasonality

```
plot(ts.1,type="l",col="red",ylab="y",xlab="Time",
main = expression(paste("Simulated Times Series with Three Month Additive Seasonality")))
```



Seasonality

```r
#Now remove the seasonal variation with the decompose function, use help(decompose for reference)
rm.seas.1 <- decompose(ts.1,type="additive")
plot(rm.seas.1)
```



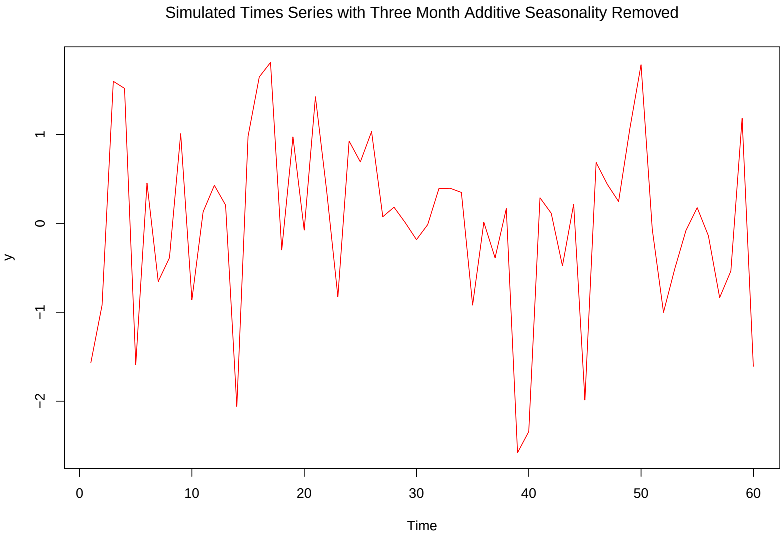
Seasonality

```r
#Alternatively, find the mean for each month,
#then subtract the corresponding monthly mean from each observation
month.avg <- rep(NA, 12)
m <- seq(0,48,by=12)

for (i in 1:12){
    month.avg[i] <- mean(y[m+i]) #Find the monthly average
}

month.avg <- rep(month.avg,5)
rm.seas.2 <- y-month.avg
rm.seas.1 <- ts.1-as.vector(rm.seas.1$seasonal)
cor(rm.seas.2, rm.seas.1[1:60])
```

```
## [1] 1
```

Seasonality

```
plot(rm.seas.2,type="l",col="red",ylab="y",xlab="Time",
main = expression(paste("Simulated Times Series with Three Month Additive Seasonality Removed")))
```



Autoregressive Processes

$$y_t = y_{t-1}\phi_1 + \epsilon_t$$

- Past realizations, y_{t-k}, influence current levels of y
- In the AR(1) case, each new realization of y_t incorporates the last period's realization, y_{t-1}

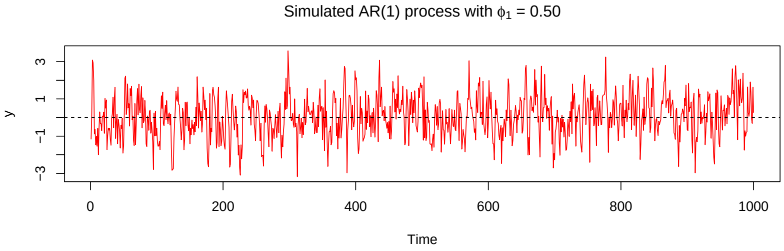
$$y_t = \sum_{j=0}^{\infty} \epsilon_{t-j}\phi^j$$

- If y_t is AR(1), then y_t includes the effects of every random shock back to the beginning of time
- It can also be thought of as the sum of exponentially weighted random shocks
- When $|\phi_1| < 1$, then with each passing observation, an increasing amount of the shock "leaks" out, but never completely disappears

Autoregressive Processes

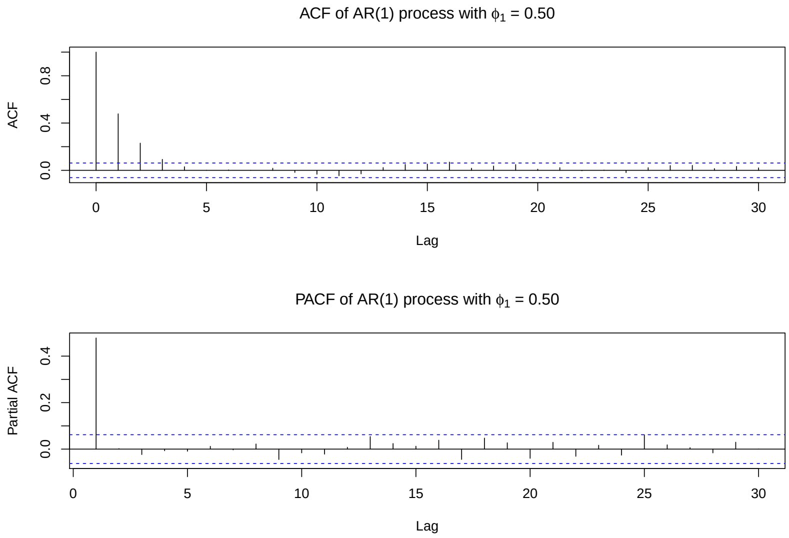
Simulate an AR(1) process with phi of 0.5. Plot the data and examine the ACF and PACF

```
#Sample from an AR(1), phi_1 = 0.5, using arima.sim()
y <- arima.sim(list(order = c(1,0,0), ar = 0.50, ma = NULL), n=1000)
#Plot the series against time
par(mfrow=c(2,1))
plot(y,type="l",col="red",ylab="y",xlab="Time",
main = expression(paste("Simulated AR(1) process with ",
phi[1]," = 0.50"))); abline(a=0,b=0,lty="dashed")
```



Autoregressive Processes

```r
#Plot the ACF and PACF
par(mfrow=c(2,1)); acf(y, main = expression(paste("ACF of AR(1) process with ",phi[1]," = 0.50")))
pacf(y, main = expression(paste("PACF of AR(1) process with ",phi[1]," = 0.50")))
```



Autoregressive Processes

Make some general observations about the AR(1) plot.

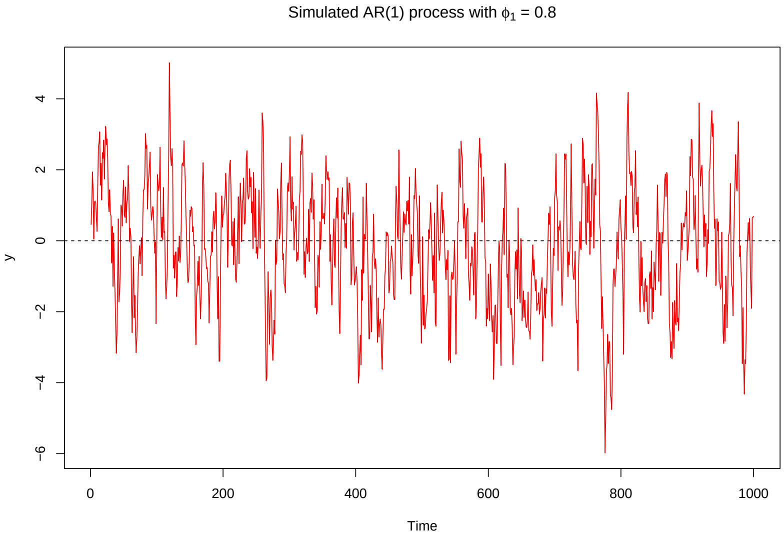
What do we learn from the ACF and PACF?

Autoregressive Processes

```r
#Simulate several AR(1) processes with -1 < phi < 1.
#Plot the data and examine the ACF and PACF

#Sample from AR(1) with phi of 0.8
ar1.1 <- arima.sim(list(order = c(1,0,0), ar=0.8, ma=NULL),n=1000)

#Sample from AR(1) with phi of 0.15
ar1.2 <- arima.sim(list(order = c(1,0,0), ar=0.15, ma=NULL),n=1000)

#Sample from AR(1) with phi of 0.99
ar1.3 <- arima.sim(list(order = c(1,0,0), ar=0.99, ma=NULL),n=1000)
```

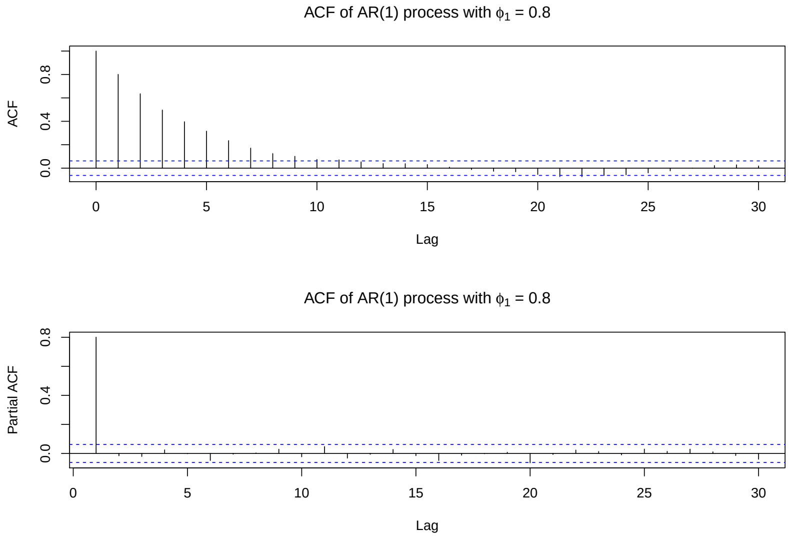
Autoregressive Processes

```
plot(ar1.1,type="l",col="red",ylab="y",xlab="Time", main = expression(
  paste("Simulated AR(1) process with ",phi[1]," = 0.8"))); abline(a=0,b=0,lty="dashed")
```



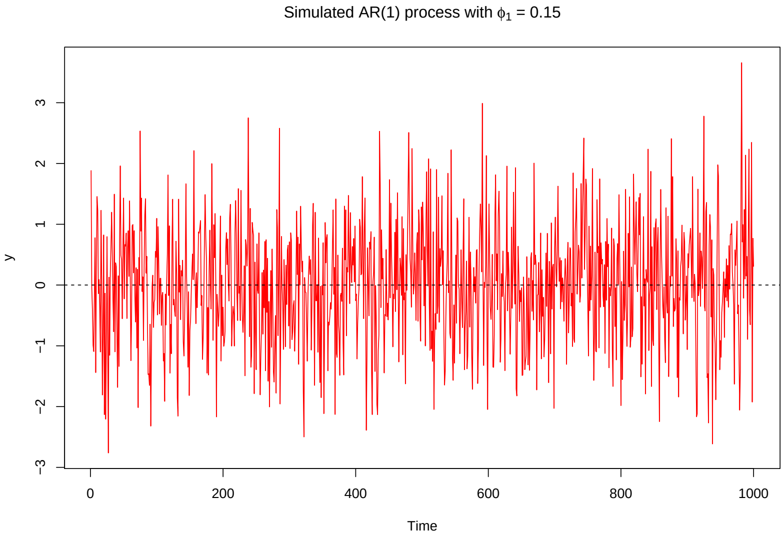
Autoregressive Processes

```
par(mfrow=c(2,1))
acf(ar1.1, main = expression(paste("ACF of AR(1) process with ",phi[1]," = 0.8")))
pacf(ar1.1, main = expression(paste("ACF of AR(1) process with ",phi[1]," = 0.8")))
```



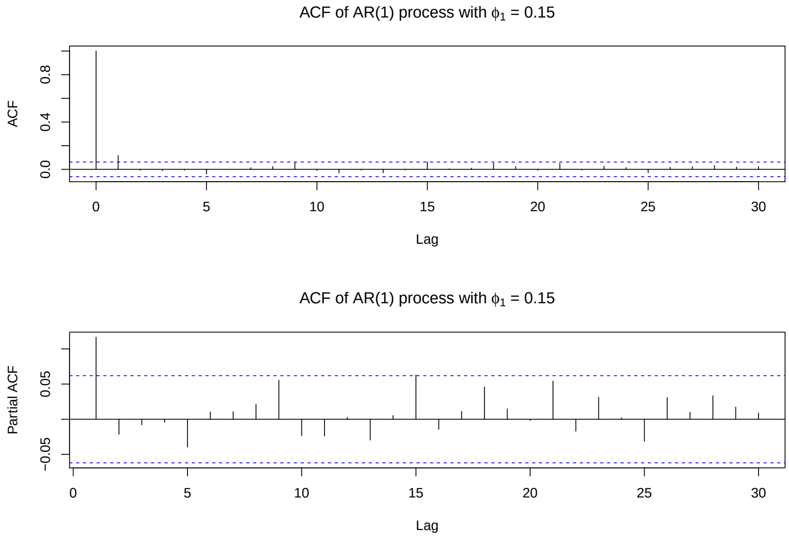
Autoregressive Processes

```r
plot(ar1.2,type="l",col="red",ylab="y",xlab="Time", main = expression(
  paste("Simulated AR(1) process with ",phi[1]," = 0.15"))); abline(a=0,b=0,lty="dashed")
```



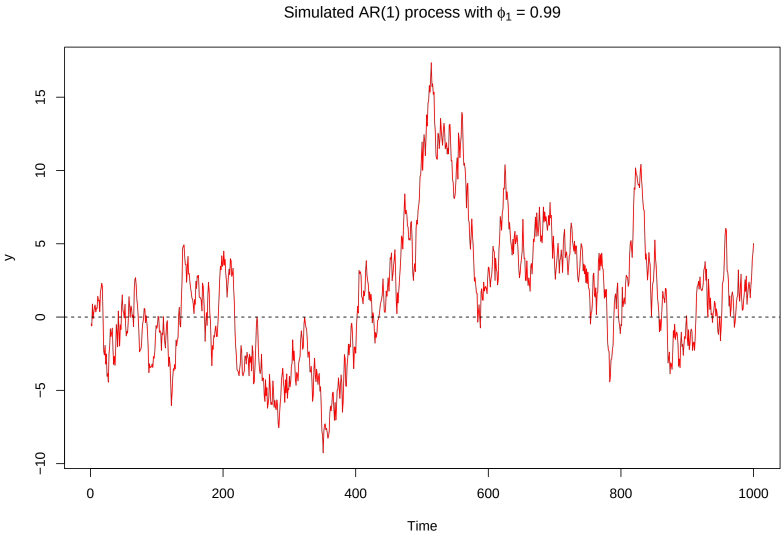
Autoregressive Processes

```r
par(mfrow=c(2,1))
acf(ar1.2, main = expression(paste("ACF of AR(1) process with ",phi[1]," = 0.15")))
pacf(ar1.2, main = expression(paste("ACF of AR(1) process with ",phi[1]," = 0.15")))
```



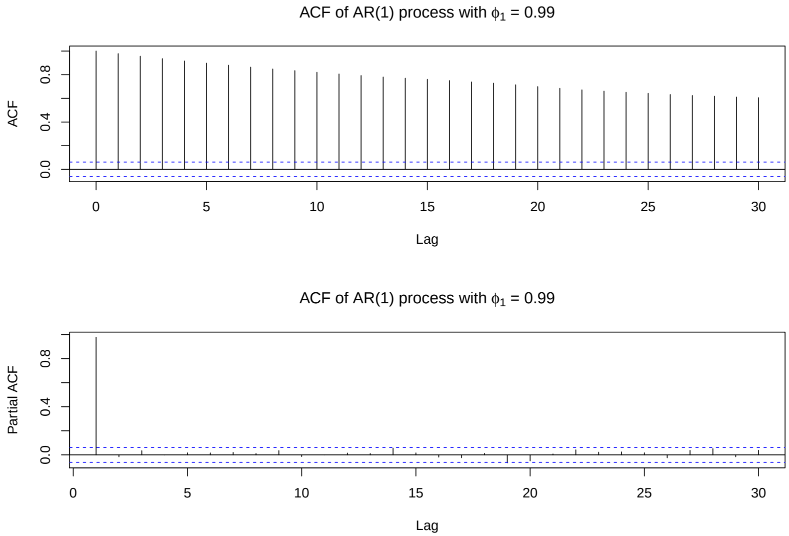
Autoregressive Processes

```
plot(ar1.3,type="l",col="red",ylab="y",xlab="Time", main = expression(
  paste("Simulated AR(1) process with ",phi[1]," = 0.99"))); abline(a=0,b=0,lty="dashed")
```



Autoregressive Processes

```
par(mfrow=c(2,1))
acf(ar1.3, main = expression(paste("ACF of AR(1) process with ",phi[1]," = 0.99")))
pacf(ar1.3, main = expression(paste("ACF of AR(1) process with ",phi[1]," = 0.99")))
```



Unit Root Tests

$$y_t = \sum_{j=0}^{\infty} \epsilon_{t-j}\phi^j$$

- If y_t is AR(1), then y_t includes the effects of every random shock back to the beginning of time
- When $|\phi_1| = 1$, then we have a random walk or unit root, and the impact of the random shocks accumulate over time rather than dissipate
- The mean of the time series is time dependent (non-stationary)

Unit Root Tests

```r
#Check for a unit root on one of the AR(1) processes

#Perform a Phillips-Perron test or Augmented Dickey-Fuller test
library(tseries)
PP.test(ar1.1)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  ar1.1
## Dickey-Fuller = -10.706, Truncation lag parameter = 7, p-value =
## 0.01
```

```r
adf.test(ar1.1)
```

```
## Warning in adf.test(ar1.1): p-value smaller than printed p-value
```

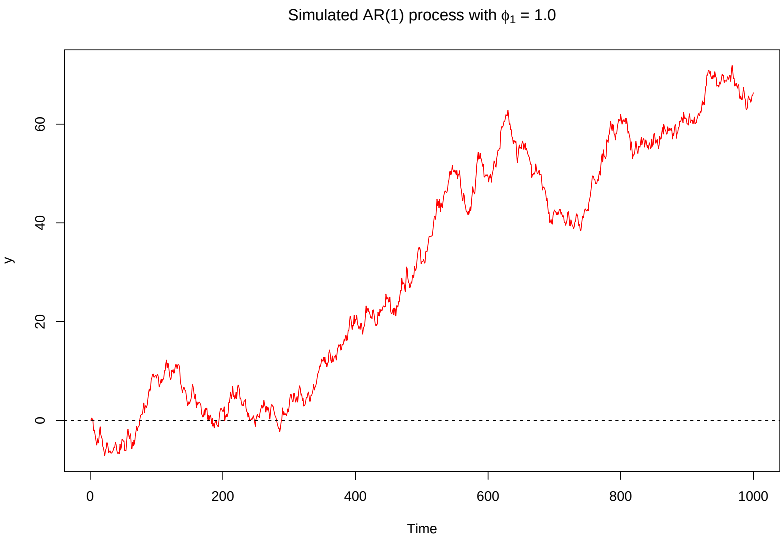
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ar1.1
## Dickey-Fuller = -7.8468, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

Autoregressive Processes

```r
# Simulate an AR(1) process with phi = 1.
# Plot the data and examine the ACF and PACF

#Set number of observations
n <- 1000

#Set phi
phi <- 1

#Set y
ar1.4 <- rep(0,n)

#Simulate AR(1) process with unit root
for (i in 2:n){
    ar1.4[i] <- ar1.4[i-1] + rnorm(1)
}
```

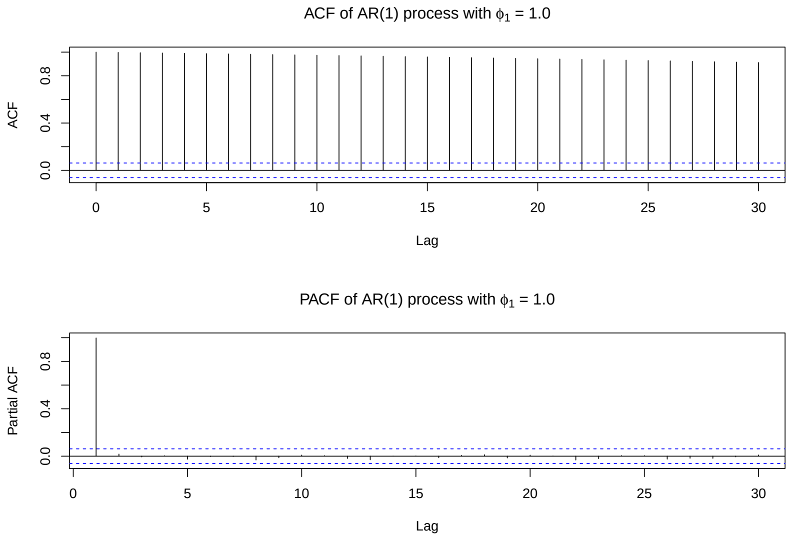
Autoregressive Processes

```r
#Plot the time series
plot(ar1.4,type="l",col="red",ylab="y",xlab="Time", main = expression(
paste("Simulated AR(1) process with ",phi[1]," = 1.0"))); abline(a=0,b=0,lty="dashed")
```



Autoregressive Processes

```
#Plot the ACF and PACF
par(mfrow=c(2,1)); acf(ar1.4, main = expression(paste("ACF of AR(1) process with ",phi[1]," = 1.0")))
pacf(ar1.4, main = expression(paste("PACF of AR(1) process with ",phi[1]," = 1.0")))
```



Autoregressive Processes

Make some general observations about the AR(1) plot.

What do we learn from the ACF and PACF?

Unit Root Tests

```r
#Perform a unit root test on the data

#Perform a Phillips-Perron test or Augmented Dickey-Fuller test
PP.test(ar1.4)
```

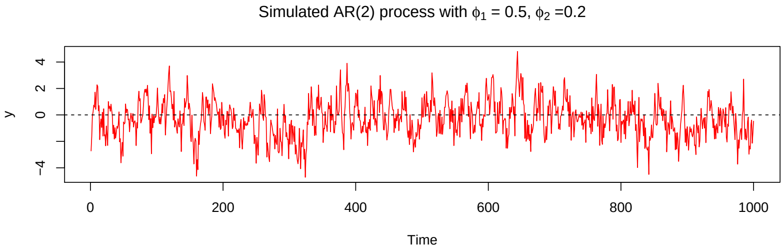
```
##
##  Phillips-Perron Unit Root Test
##
## data:  ar1.4
## Dickey-Fuller = -2.1789, Truncation lag parameter = 7, p-value =
## 0.5026
```

```r
adf.test(ar1.4)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ar1.4
## Dickey-Fuller = -2.0754, Lag order = 9, p-value = 0.5464
## alternative hypothesis: stationary
```

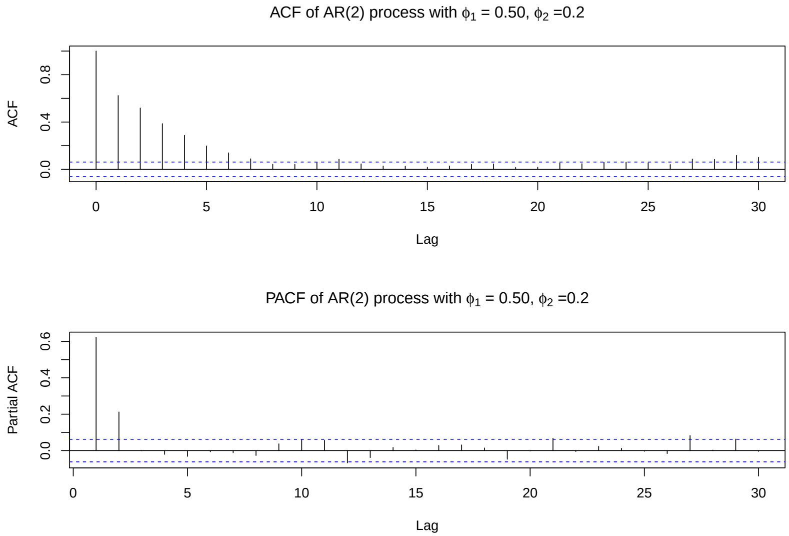
Autoregressive Processes

```
#Simulate an AR(2) process with phi_1 = 0.5 and phi_2 = 0.2.

#Plot the data and inspect the ACF and PACF
ar2.1 <- arima.sim(list(order = c(2,0,0), ar = c(0.50,0.2), ma = NULL), n=1000)

#Plot the series against time
par(mfrow=c(2,1))

plot(ar2.1,type="l",col="red",ylab="y",xlab="Time", main = expression(paste
("Simulated AR(2) process with ",phi[1]," = 0.5, ", phi[2]," =0.2")));abline(a=0,b=0,lty="dashed")
```



Autoregressive Processes

```
par(mfrow=c(2,1)) #Plot the ACF and PACF
acf(ar2.1, main = expression(paste("ACF of AR(2) process with ",phi[1]," = 0.50, ", phi[2]," =0.2")))
pacf(ar2.1, main = expression(paste("PACF of AR(2) process with ",phi[1]," = 0.50, ", phi[2]," =0.2")))
```



Unit Root Tests

```r
#Is the time series stationary?

#Confirm results with a unit root test
PP.test(ar2.1)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  ar2.1
## Dickey-Fuller = -15.945, Truncation lag parameter = 7, p-value =
## 0.01
```

```r
adf.test(ar2.1)
```

```
## Warning in adf.test(ar2.1): p-value smaller than printed p-value
```

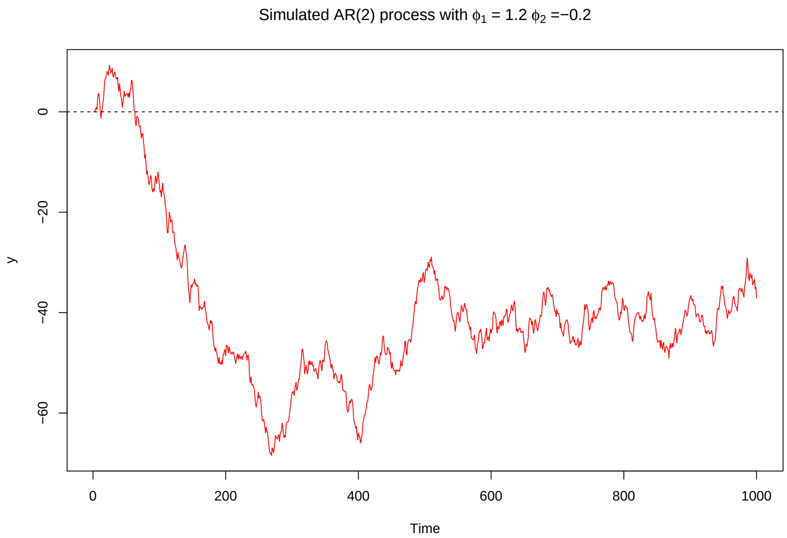
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ar2.1
## Dickey-Fuller = -7.6086, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

Autoregressive Processes

```r
#Sample from an AR(2), phi_1 = 1.2, phi_2 = -0.2 and plot the ACF and PACF

#Set number of observations
n <- 1000

#Set phi
phi_1 <- 1.2
phi_2 <- -0.2

#Set y vector
ar2.2 <- rep(0,n)

#Simulate AR(2) process with unit root
for (i in 3:n){
    ar2.2[i] <- ar2.2[i-1]*phi_1 + ar2.2[i-2]*phi_2 + rnorm(1)
}
```

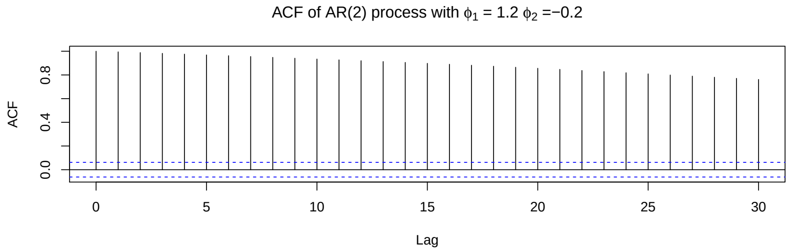
Autoregressive Processes

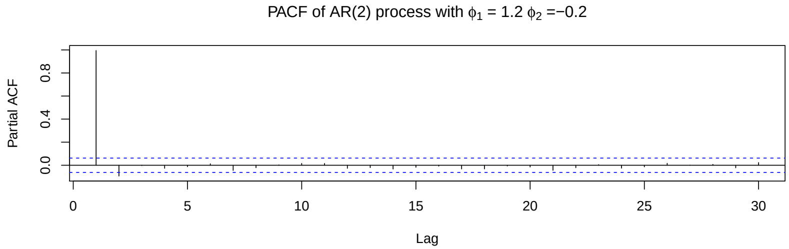
```
#Plot the time series
plot(ar2.2,type="l",col="red",ylab="y",xlab="Time", main = expression(paste
("Simulated AR(2) process with ",phi[1]," = 1.2 ", phi[2]," =-0.2")));abline(a=0,b=0,lty="dashed")
```



Autoregressive Processes

```r
par(mfrow=c(2,1)) #Again, what can we (and can we not) infer from the ACF and PACF?
acf(ar2.2, main = expression(paste("ACF of AR(2) process with ",phi[1]," = 1.2 ", phi[2]," =-0.2")))
pacf(ar2.2, main = expression(paste("PACF of AR(2) process with ",phi[1]," = 1.2 ", phi[2]," =-0.2")))
```





Autoregressive Processes

```r
#Try to check whether process is stationary with a unit root test

#Perform a Phillips-Perron or Augmented Dickey-Fuller test
adf.test(ar2.2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ar2.2
## Dickey-Fuller = -2.2725, Lag order = 9, p-value = 0.463
## alternative hypothesis: stationary
```

```r
PP.test(ar2.2)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  ar2.2
## Dickey-Fuller = -2.0299, Truncation lag parameter = 7, p-value =
## 0.5657
```

Moving Average Processes

$$y_t = \epsilon_{t-1}\rho_1 + \epsilon_t$$

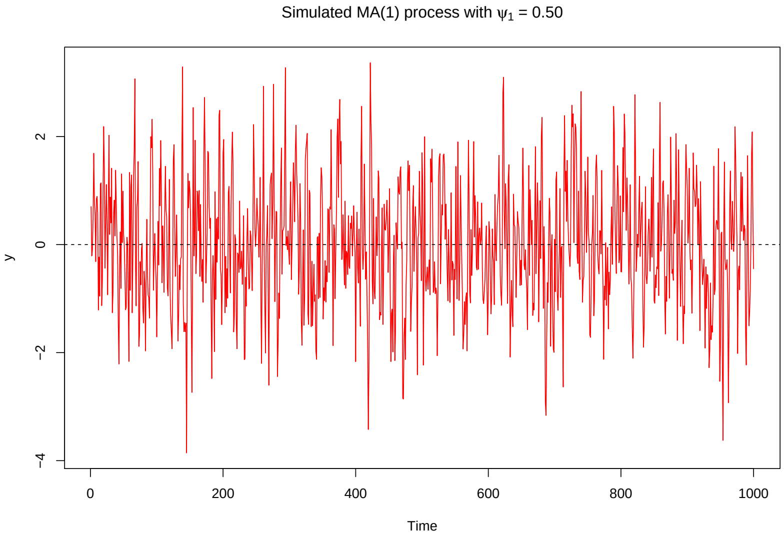
- Past random shocks, ϵ_{t-k}, influence current levels of y
- If y_t is MA(1), then the stochastic component is a weighted average of the current and previous error
- In an MA(q) process, the effects of past shocks die out after q periods
- MA(q) processes are always stationary for finite q

Moving Average Processes

```r
#Simulate several MA(q) processes.
#Plot the data and examine the ACFs and PACFs

#Sample from an MA(1), psi_1 = 0.5
ma1.1 <- arima.sim(list(order = c(0,0,1), ar = NULL, ma = 0.5), n=1000)

#Sample from MA(2) with psi_1 = 0.3 and psi_2=0.7
ma2.1 <- arima.sim(list(order=c(0,0,2), ar=NULL, ma=c(0.3,0.7)),n=1000)

#MA(5) with psi_1 = 0.3 and psi_2=0.7 and psi_3=0.5 and psi_4=0.7 and psi_5=1.2
ma5.1 <- arima.sim(list(order=c(0,0,5), ar=NULL, ma=c(0.3,0.7,0.5,0.7,1.2)),n=1000)
```

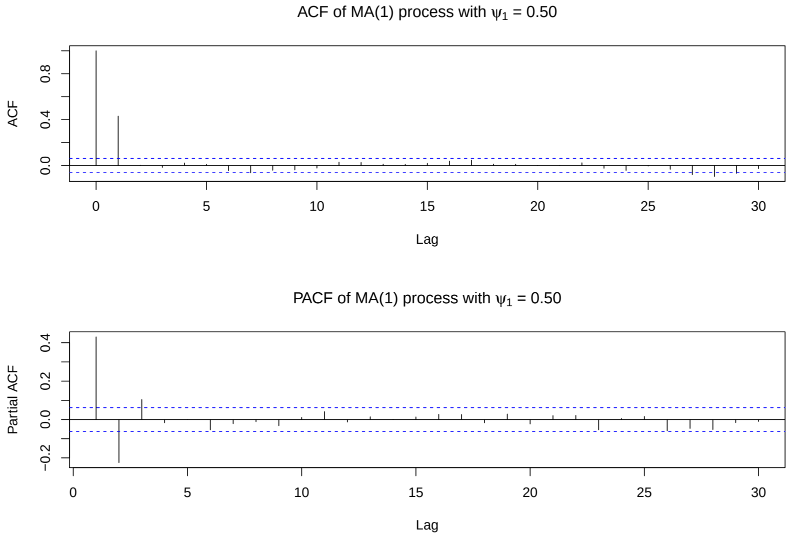
Moving Average Processes

```
plot(ma1.1,type="l",col="red",ylab="y",xlab="Time",main = expression(paste
("Simulated MA(1) process with ",psi[1]," = 0.50")));abline(a=0,b=0,lty="dashed")
```



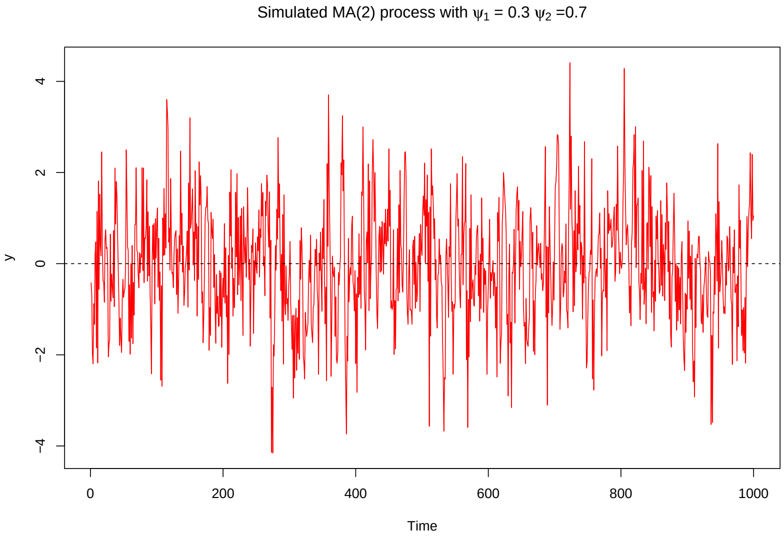
Moving Average Processes

```
#Plot the ACF and PACF
par(mfrow=c(2,1))
acf(ma1.1, main = expression(paste("ACF of MA(1) process with ",psi[1]," = 0.50")))
pacf(ma1.1, main = expression(paste("PACF of MA(1) process with ",psi[1]," = 0.50")))
```



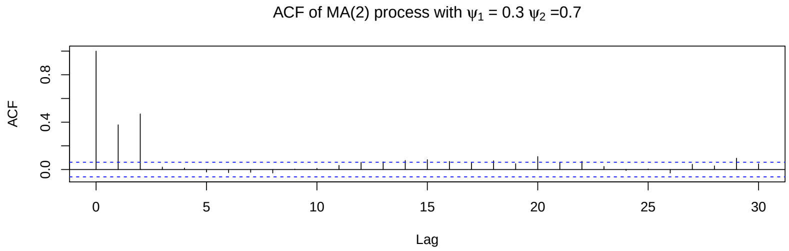
Moving Average Processes

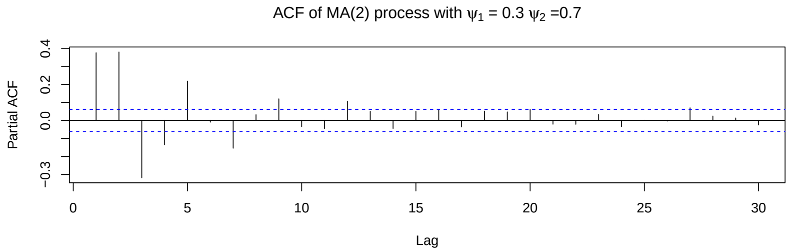
```
plot(ma2.1,type="l",col="red",ylab="y",xlab="Time", main = expression(paste
("Simulated MA(2) process with ",psi[1]," = 0.3 ",psi[2]," =0.7")));abline(a=0,b=0,lty="dashed")
```



Moving Average Processes

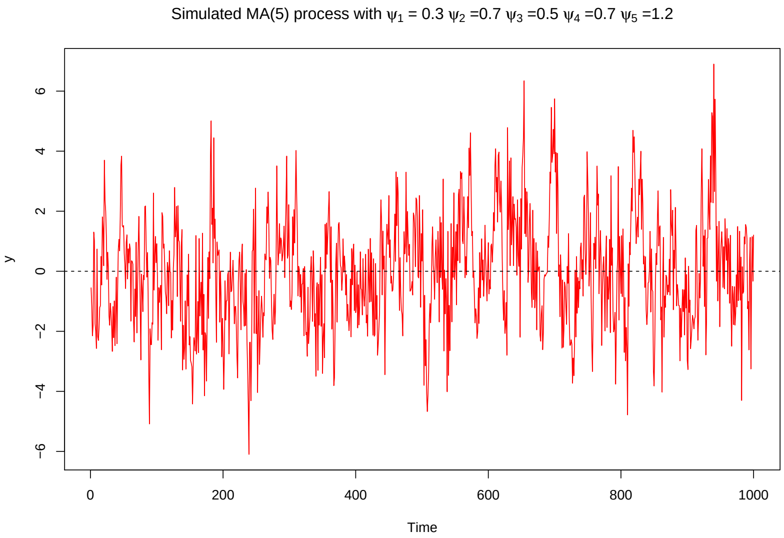
```
par(mfrow=c(2,1)) #Plot the ACF and PACF
acf(ma2.1, main = expression(paste("ACF of MA(2) process with ",psi[1]," = 0.3 ",psi[2]," =0.7")))
pacf(ma2.1, main = expression(paste("ACF of MA(2) process with ",psi[1]," = 0.3 ",psi[2]," =0.7")))
```





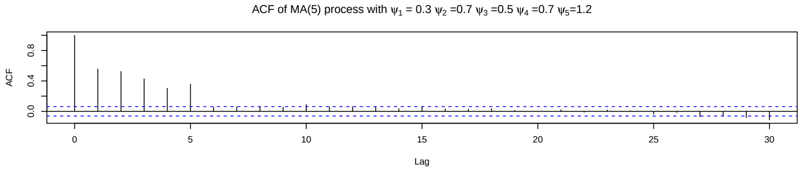
Moving Average Processes

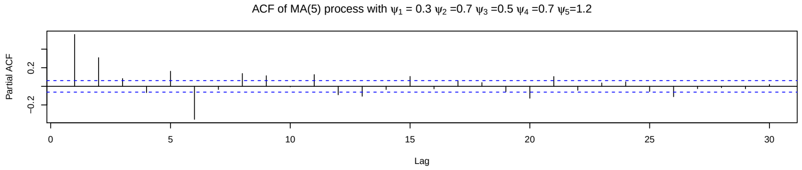
```r
plot(ma5.1,type="l",col="red",ylab="y",xlab="Time", main = expression(paste
("Simulated MA(5) process with ",psi[1]," = 0.3 ",psi[2]," =0.7 ",
psi[3]," =0.5 ", psi[4]," =0.7 ", psi[5], " =1.2")));abline(a=0,b=0,lty="dashed")
```



Moving Average Processes

```
par(mfrow=c(3,1)) #Plot the ACF and PACF
acf(ma5.1, main = expression(paste("ACF of MA(5) process with ",
psi[1]," = 0.3 ",psi[2]," =0.7 ",psi[3]," =0.5 ", psi[4]," =0.7 ", psi[5], "=1.2")))
pacf(ma5.1, main = expression(paste("ACF of MA(5) process with ",
psi[1]," = 0.3 ",psi[2]," =0.7 ",psi[3]," =0.5 ", psi[4]," =0.7 ", psi[5], "=1.2")))
```





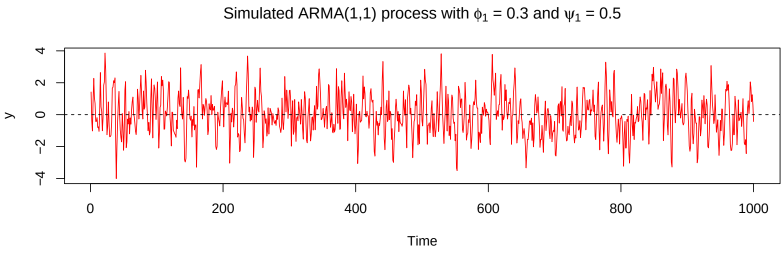
Moving Average Processes

What do we learn about the effect of past shocks in an MA(q) process from the ACFs and PACFs?

How can we identify an AR versus an MA process from the ACF and PACF.

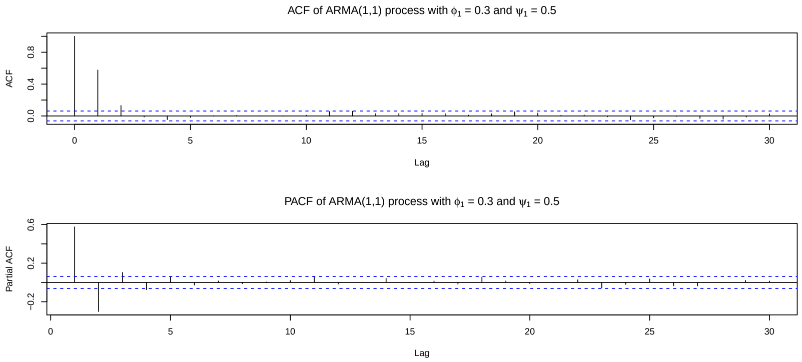
ARMA Processes

```
# Simulate an ARMA(1,1) process
arma1.1 <- arima.sim(list(order=c(1,0,1), ar=0.3, ma=0.5), n=1000)

par(mfrow=c(2,1)) # Plot the data
plot(arma1.1,type="l",col="red",ylab="y",xlab="Time", main = expression(
  paste("Simulated ARMA(1,1) process with ",phi[1]," = 0.3", " and ", psi[1]," = 0.5")))
abline(a=0,b=0,lty="dashed")
```



ARMA Processes

```
par(mfrow=c(3,1)) # Plot the ACF and PACF
acf(arma1.1, main = expression(paste("ACF of ARMA(1,1) process with ",
phi[1]," = 0.3", " and ", psi[1]," = 0.5")))
pacf(arma1.1, main = expression(paste("PACF of ARMA(1,1) process with ",
phi[1]," = 0.3", " and ", psi[1]," = 0.5")))
```


ACF of ARMA(1,1) process with ϕ_1 = 0.3 and ψ_1 = 0.5
ACF
Lag
PACF of ARMA(1,1) process with ϕ_1 = 0.3 and ψ_1 = 0.5
Partial ACF
Lag