CSSS 512: Lab 4

Modeling Nonstationary Time Series

2018-4-27

Agenda

0. Review of nonstationary processes
1. Studying the time series
2. Differencing the time series
3. Estimating and selecting ARIMA models
4. Counterfactual forecasting
5. Cointegration analysis
6. Homework 2, Question 2

A Word on Unit Root Tests

Intuition: if time series is stationary, then regressing $y_t - y_{t-1}$ on y_{t-1} should produce a negative coefficient. Why?

In a stationary series, knowing the past value of the series helps to predict the next period's change. Positive shifts should be followed by negative shifts (mean reversion).

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

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

$$\Delta y_t = \gamma y_{t-1} + \epsilon_t, \text{ where } \gamma = (\rho - 1)$$

Augmented Dickey-Fuller test: null hypothesis of unit root.

Same with Phillips-Perron test, but differs in how the AR(p) time series is modeled: lags, serial correlation, heteroskedasticity.

A Word on Unit Root Tests

The ADF test regression can be expressed as follows:

$$\Delta y_t = \beta \boldsymbol{D_t} + \pi y_{t-1} + \sum_{j=1}^{p} \psi_j \Delta y_{t-j} + \epsilon_t$$

where $\boldsymbol{D}_t$ is a vector of deterministic trends. The p lagged difference terms, Δy_{t-j}, are used to approximate the ARMA structure of the errors, and the value of p is set so that the error is serially uncorrelated. The error term is assumed to be homoskedastic.

Under the null hypothesis, π is set to 0 and is used to construct the ADF test statistic.

A Word on Unit Root Tests

The PP test regression can be expressed as follows:

$$\Delta y_t = \beta \boldsymbol{D_t} + \pi y_{t-1} + u_t$$

The PP test ignores any serial correlation in the test regression. Instead, it corrects for serial correlation and heteroskedasticity in the errors u_t by directly modifying the test statistic, $t_{\pi=0}$

As we can see, the tests vary in how the time series is modeled. But in both tests, the series does not need to be de-trended. Other tests can include a structural break in the underling model: Zivot-Andrews test.

A Word on Counterfactual Forecasting Using `simcf`

There are several functions you can use to forecast using simulation:

These are `ldvsimev()`, `ldvsimpv()`, `ldvsimfd()`, `ldvsimrr()`, and `ldvsimpr()`.

We will be mainly using the first two. `ldvsimev()` forecasts expected values. `ldvsimpv()` forecasts predicted values.

Main difference in code is that `ldvsimpv()` needs the `sigma` argument while `ldvsimev()` does not.

`ldvsimev()` also does not take `rho` or `lagEPS` arguments since errors are not used to compute the expected values.

We will be looking at examples.

Review of Nonstationary Processes

Recall that stationary time series have three properties:

1. Mean stationarity
   - mean does not depend on t, constant over time
   - variance also does not depend on t, constant over time
2. Covariance stationarity
   - covariance of y_t and y_{t-1} do not depend on t
   - does not depend on the actual time the covariance is computed
3. Ergodicity
   - sample moments coverge in probability to the population moments
   - sample mean and variance tend to be the same as entire series

Nonstationary procesess lack these properties. Note that we are abstracting from trends and other covariates.

Review of Nonstationary Processes

Why do nonstationary processes matter?

1. ACF and PACF not defined since covariances depend on t

2. Spurious regression: we may detect strong correlations between nonstationary processes although they are really independent

3. Long run forecasts are very difficult since they do not tend toward any mean

Review of Nonstationary Processes

Solutions?

1. Analyze nonstationary process using ARIMA (differencing)
    - effective at capturing short run changes
    - outcome is transformed to a difference
    - long-run predictions not feasible

2. Analyze nonstationary process using cointegration
    - effective at capturing long run relationships between nonstationary processes
    - outcome is left as a level
    - short-run and long-run predictions feasible

    - appropriate for analyzing multiple time series

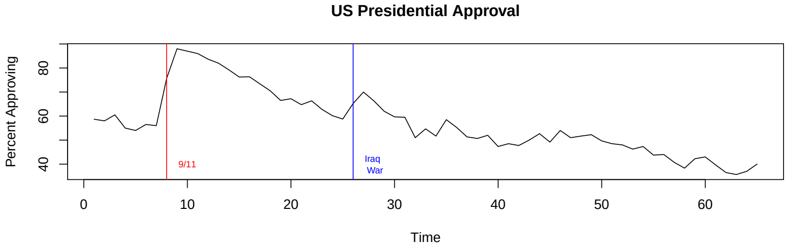
We will review both in this lab session.

Studying the Time Series

```r
rm(list=ls())

#Load libraries
library(tseries)            # For unit root tests
library(forecast)           # For decompose()
library(lmtest)             # For Breusch-Godfrey LM test of serial correlation
library(urca)               # For estimating cointegration models
library(simcf)              # For counterfactual simulation via ldvsimev()
library(MASS)               # For mvrnorm()
library(RColorBrewer)       # For nice colors
library(Zelig)              # For approval data
library(quantmod)           # For creating lags
source("TSplotHelper.R")    # Helper function for counterfactual time series plots

############################################################################
#Load data
#US Presidential approval data (Bush, Monthly, 2/2001--6/2006)
#Includes average oil price data ($/barrel?)
#
#Variable names:  month year approve disapprove unsure
#                 sept.oct.2001 iraq.war avg.price

data(approval)
attach(approval)
```

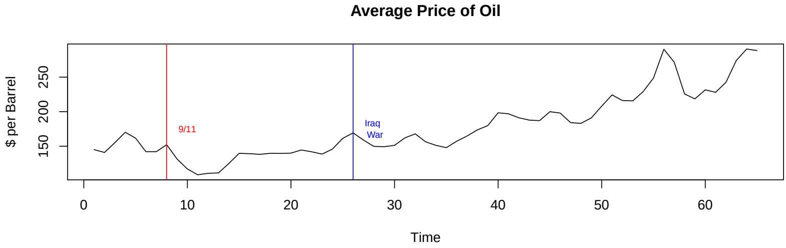
Studying the Time Series

```r
# Look at the time series
par(mfrow=c(2,1))
plot(approve,type="l",ylab="Percent Approving",xlab="Time",
     main = "US Presidential Approval")
lines(x=c(8,8),y=c(-1000,1000),col="red")
lines(x=c(26,26),y=c(-1000,1000),col="blue")
text("9/11",x = 10, y = 40, col="red",cex=0.7)
text("Iraq \n War",x = 28, y = 40, col="blue",cex=0.7)
```



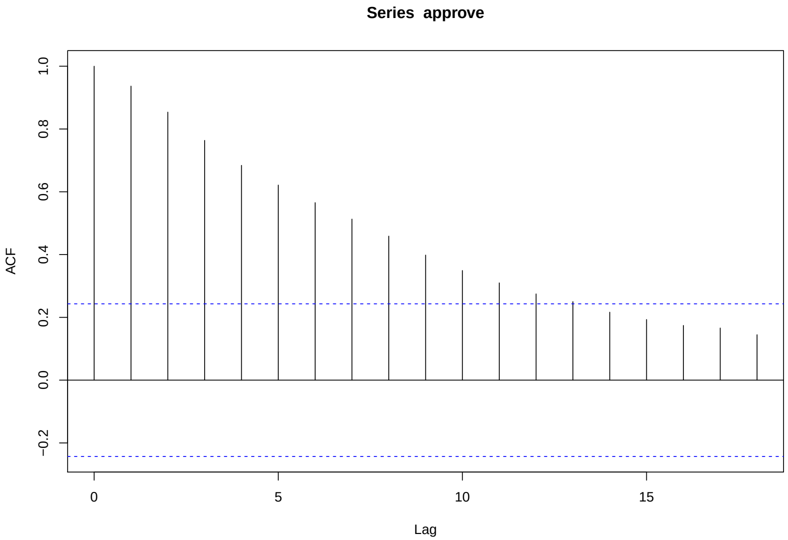
Studying the Time Series

```r
par(mfrow=c(2,1))
plot(avg.price,type="l",ylab="$ per Barrel",xlab="Time",
     main = "Average Price of Oil")
lines(x=c(8,8),y=c(-1000,1000),col="red")
lines(x=c(26,26),y=c(-1000,1000),col="blue")
text("9/11",x = 10, y = 175, col="red",cex=0.7)
text("Iraq \n War",x = 28, y = 175, col="blue",cex=0.7)
```



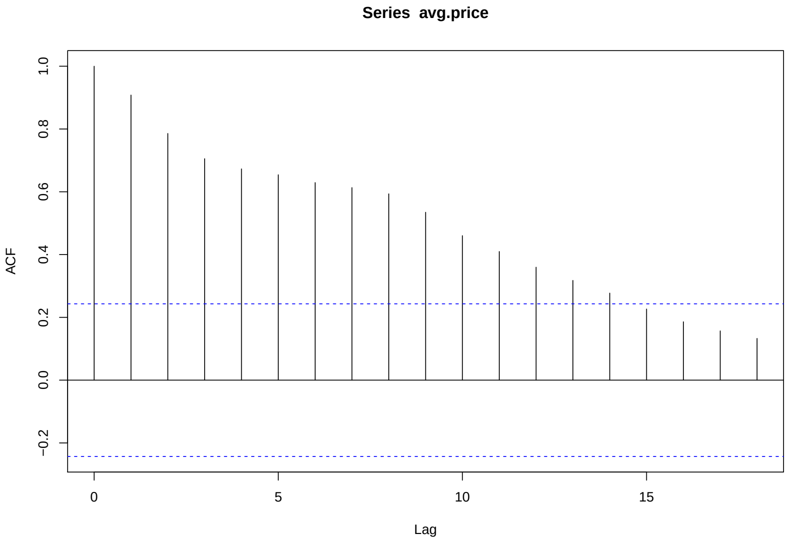
Studying the Time Series

```
acf(approve)
```



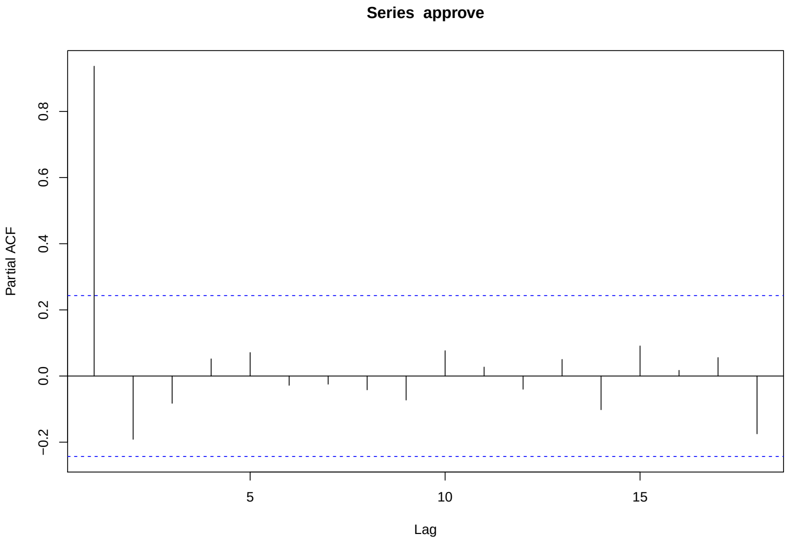
Studying the Time Series

```r
acf(avg.price)
```



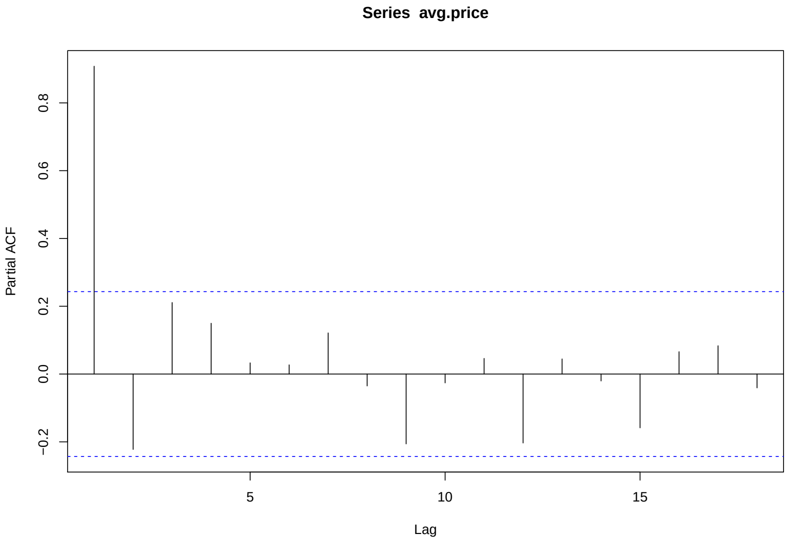
Studying the Time Series

```r
#Look at the PACF
pacf(approve)
```



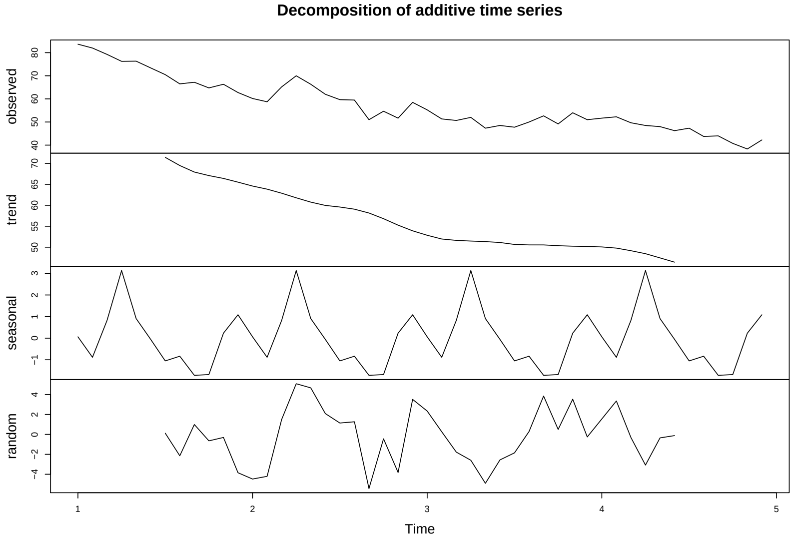
Studying the Time Series

```r
pacf(avg.price)
```



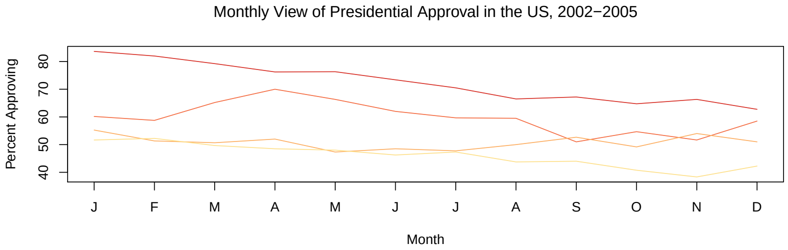
Studying the Time Series

```r
#Look at the decomposed time series
par(mfrow=c(2,1))
plot(decompose(ts(approve[12:59],freq=12)))
```



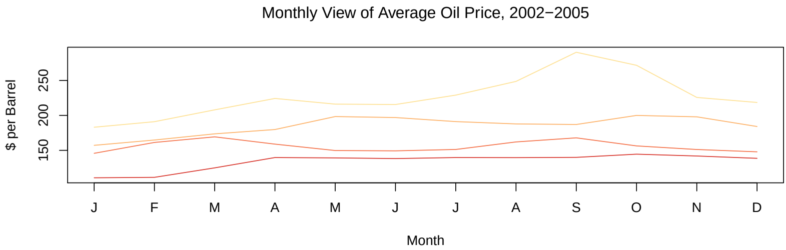
Studying the Time Series

```
#Check for seasonality in approval
col <- brewer.pal(8, "RdYlGn")

#Gather the data (sort the number of deaths by month and year in a matrix)
appmat <- matrix(approve[12:59],nrow=12,ncol=length(approve[12:59])/12, byrow=FALSE)

#Repeat them as many times as needed
col <-  as.vector(t(matrix(col, nrow=length(col), ncol=ceiling(ncol(appmat)/length(col)))))

#Plot each year over the months
par(mfrow=c(2,1))
matplot(appmat, type="l", col=col, lty=1, xaxt="n", ylab="Percent Approving", xlab="Month",
        main=expression(paste("Monthly View of Presidential Approval in the US, 2002-2005")))
axis(1, at=1:12, labels=c("J", "F", "M", "A", "M", "J", "J", "A", "S", "O", "N", "D"))
abline(a=0,b=0,lty="dashed")
```



Studying the Time Series

```r
#Check for seasonality in avg price
col <- brewer.pal(8, "RdYlBu")
pricemat <- matrix(avg.price[12:59],nrow=12,ncol=length(avg.price[12:59])/12, byrow=FALSE)
col <-  as.vector(t(matrix(col, nrow=length(col), ncol=ceiling(ncol(pricemat)/length(col)))))
par(mfrow=c(2,1))
matplot(pricemat, type="l", col=col, lty=1, xaxt="n", ylab="$ per Barrel", xlab="Month",
        main=expression(paste("Monthly View of Average Oil Price, 2002-2005")))
axis(1, at=1:12, labels=c("J", "F", "M", "A", "M", "J", "J", "A", "S", "O", "N", "D"))
abline(a=0,b=0,lty="dashed")
```



Studying the Time Series

```r
#Check for a unit root in approval
PP.test(approve)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  approve
## Dickey-Fuller = -2.8387, Truncation lag parameter = 3, p-value =
## 0.235
```

```r
adf.test(approve)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  approve
## Dickey-Fuller = -3.9565, Lag order = 3, p-value = 0.01721
## alternative hypothesis: stationary
```

Studying the Time Series

```r
#Check for a unit root in average price
PP.test(avg.price)
```

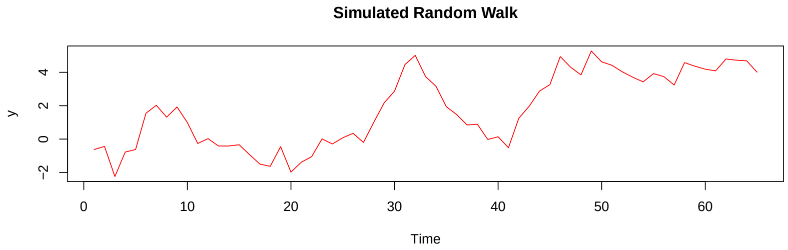
```
##
##  Phillips-Perron Unit Root Test
##
## data:  avg.price
## Dickey-Fuller = -2.3318, Truncation lag parameter = 3, p-value =
## 0.4405
```

```r
adf.test(avg.price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  avg.price
## Dickey-Fuller = -3.0115, Lag order = 3, p-value = 0.1649
## alternative hypothesis: stationary
```

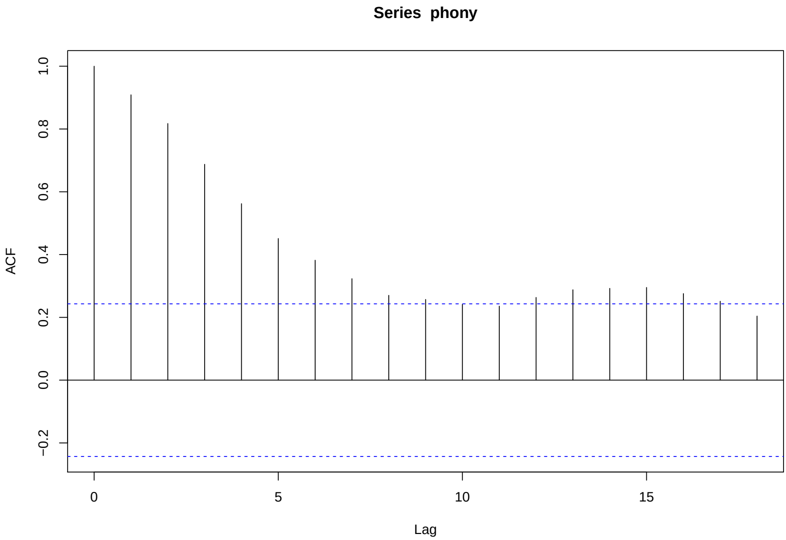
Studying the Time Series

```r
#Investigate some other (potentially) non-stationary time series data
#Simulated random walk
set.seed(1)
phony <- rnorm(length(approve))
for (i in 2:length(phony)){
    phony[i] <- phony[i-1] + rnorm(1)
}

#Plot the data
par(mfrow=c(2,1))
plot(phony, type="l", col="red", ylab="y",xlab="Time", main = "Simulated Random Walk")
```



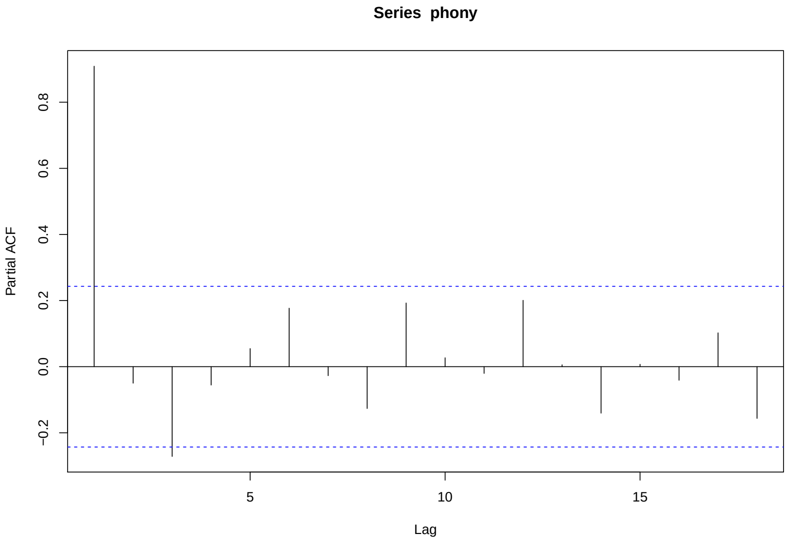
Studying the Time Series

```r
#Check the ACF
acf(phony)
```



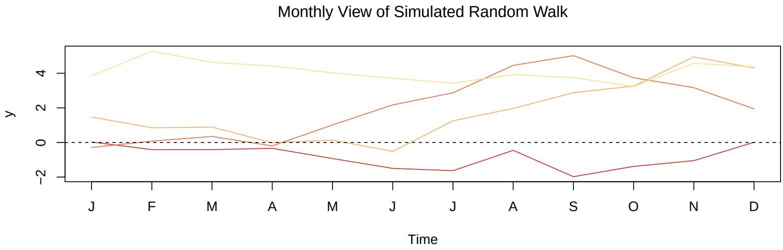
Studying the Time Series

```r
#Check the PACF
pacf(phony)
```



Studying the Time Series

```r
#Check for seasonality
col <- brewer.pal(8, "RdYlGn")
phonymat <- matrix(phony[12:59],nrow=12,ncol=length(phony[12:59])/12, byrow=FALSE)
par(mfrow=c(2,1))
matplot(phonymat, type="l", col=col, lty=1, xaxt="n", ylab="y", xlab="Time",
        main=expression(paste("Monthly View of Simulated Random Walk")))
axis(1, at=1:12, labels=c("J", "F", "M", "A", "M", "J", "J", "A", "S", "O", "N", "D"))
abline(a=0,b=0,lty="dashed")
```



Studying the Time Series

```r
PP.test(phony)
```

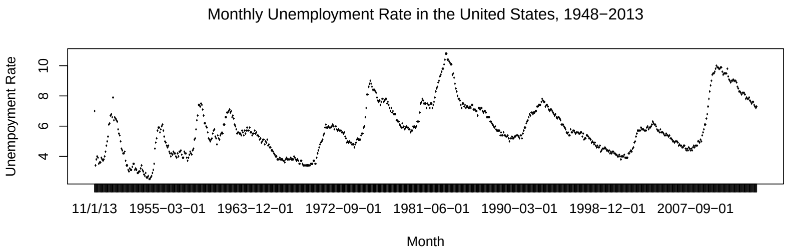
```
##
##  Phillips-Perron Unit Root Test
##
## data:  phony
## Dickey-Fuller = -2.8321, Truncation lag parameter = 3, p-value =
## 0.2377
```

```r
adf.test(phony)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  phony
## Dickey-Fuller = -3.5359, Lag order = 3, p-value = 0.04576
## alternative hypothesis: stationary
```

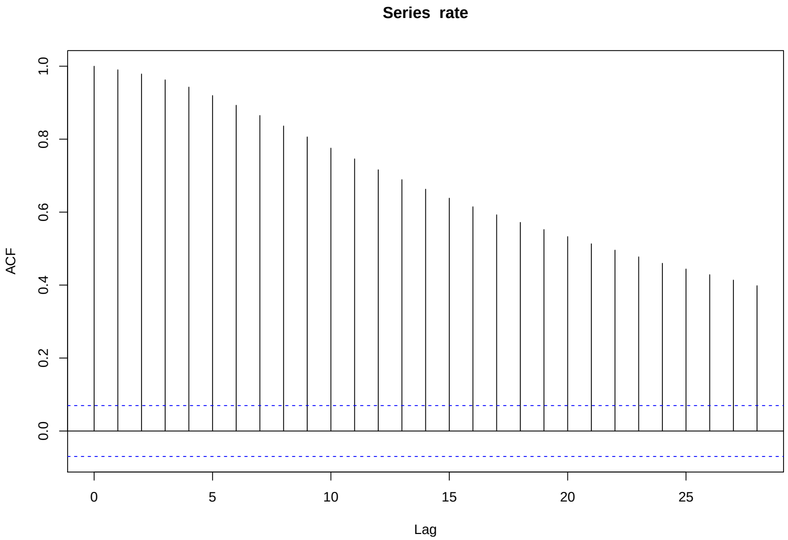
Studying the Time Series

```
#US unemployment rate, Watson (2014)
unemployment <- read.csv("unemployment.csv", header=TRUE)
date <- unemployment$date
rate <- unemployment$unemployment_rate

#Plot the data
par(mfrow=c(2,1))
plot(date, rate, xlab="Month", ylab="Unempoyment Rate",
     main=expression(paste("Monthly Unemployment Rate in the United States, 1948-2013")))
```



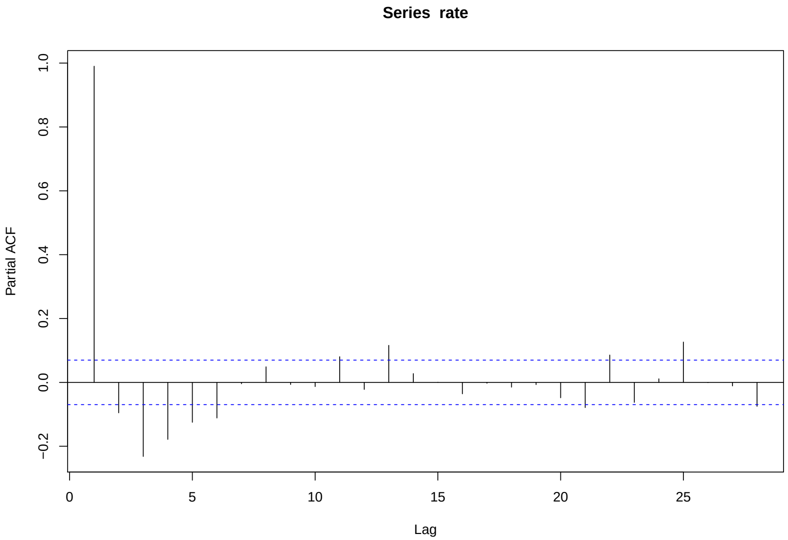
Studying the Time Series

```r
#Check the ACF
acf(rate)
```



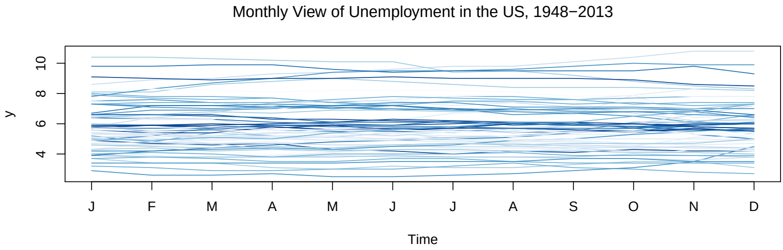
Studying the Time Series

```r
#Check the PACF
pacf(rate)
```



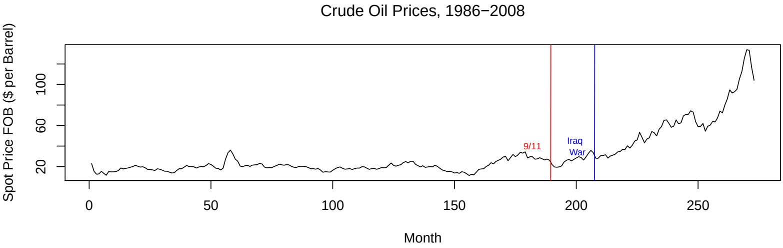
Studying the Time Series

```
#Check for seasonality
col <- brewer.pal(8, "Blues")
unempmat <- matrix(rate[1:780],nrow=12,ncol=length(rate[1:780])/12, byrow=FALSE)
par(mfrow=c(2,1))
matplot(unempmat, type="l", col=col, lty=1, xaxt="n", ylab="y", xlab="Time",
        main=expression(paste("Monthly View of Unemployment in the US, 1948-2013")))
axis(1, at=1:12, labels=c("J", "F", "M", "A", "M", "J", "J", "A", "S", "O", "N", "D"))
abline(a=0,b=0,lty="dashed")
```



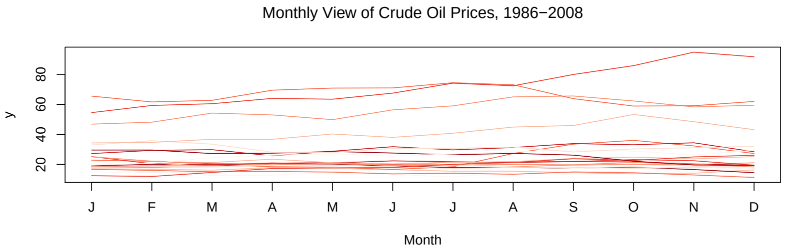
Studying the Time Series

```
#Crude oil prices, Hamilton (2008)
crude <- read.csv("crude_oil.csv", header=TRUE)
price <- crude$spot_price_fob

#Plot the data
par(mfrow=c(2,1))
plot(price, type="l", xlab="Month", ylab="Spot Price FOB ($ per Barrel)",
     main=expression(paste("Crude Oil Prices, 1986-2008")))
lines(x=c(189,190),y=c(-1000,1000),col="red")
lines(x=c(207,208),y=c(-1000,1000),col="blue")
text("9/11",x = 182, y = 40, col="red",cex=0.7)
text("Iraq \n War",x = 200, y = 40, col="blue",cex=0.7)
```



Studying the Time Series

```
#Check for seasonality
col <- brewer.pal(8, "Reds")
oilmat <- matrix(price[1:264],nrow=12,ncol=length(rate[1:264])/12, byrow=FALSE)
#col <-  as.vector(t(matrix(col, nrow=length(col), ncol=ceiling(ncol(price)/length(col)))))
par(mfrow=c(2,1))
matplot(oilmat, type="l", col=col, lty=1, xaxt="n", ylab="y", xlab="Month",
        main=expression(paste("Monthly View of Crude Oil Prices, 1986-2008")))
axis(1, at=1:12, labels=c("J", "F", "M", "A", "M", "J", "J", "A", "S", "O", "N", "D"))
abline(a=0,b=0,lty="dashed")
```



Studying the Time Series

```r
#Perform a unit root test
PP.test(price)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  price
## Dickey-Fuller = -0.77132, Truncation lag parameter = 5, p-value =
## 0.9634
```

```r
adf.test(price)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  price
## Dickey-Fuller = 0.83045, Lag order = 6, p-value = 0.99
## alternative hypothesis: stationary
```

Differencing the Time Series

Recall that we can use differencing to address the issue of nonstationarity.

We can difference a random walk as follows:

$$y_t = 1y_{t-1} + \boldsymbol{x_t}\beta + e_t$$

$$y_t - y_{t-1} = y_{t-1} - y_{t-1} + \boldsymbol{x_t}\beta + e_t$$

$$\Delta y_t = \boldsymbol{x_t}\beta + e_t$$

The result is AR(0) and stationary.

But note that we are now explaining the short run difference not the level.

Differencing the Time Series

```r
#Differencing time series data
#Back to the Presidential Approval dataset
#Consider the first difference of each variable
approveLag <- c(NA, approve[1:(length(approve)-1)])
approveLag2 <- as.vector(Lag(approve,k=1))

approveDiff <- approve - approveLag
approveLagDiff <- cbind(approve, approveLag, approveDiff)
head(approveLagDiff)
```

```
##      approve approveLag approveDiff
## [1,]   58.67         NA          NA
## [2,]   58.00      58.67       -0.67
## [3,]   60.50      58.00        2.50
## [4,]   55.00      60.50       -5.50
## [5,]   54.00      55.00       -1.00
## [6,]   56.50      54.00        2.50
```

```r
avg.priceLag <- c(NA, avg.price[1:(length(avg.price)-1)])
avg.priceDiff <- avg.price - avg.priceLag

avg.priceLagDiff <- cbind(avg.price, avg.priceLag, avg.priceDiff)
head(avg.priceLagDiff)
```

```
##      avg.price avg.priceLag avg.priceDiff
## [1,]   144.975           NA            NA
## [2,]   140.925      144.975        -4.050
## [3,]   155.160      140.925        14.235
## [4,]   170.175      155.160        15.015
## [5,]   161.625      170.175        -8.550
## [6,]   142.060      161.625       -19.565
```

Differencing the Time Series

```r
#Consider the second difference of each variable
approve2Lag <- c(NA, NA, approve[2:length(approve)-2])
approve2Lag2 <- as.vector(Lag(approve,k=2))

approve2Diff <- approve - approve2Lag
approveLagDiff <- cbind(approve, approveLag, approve2Lag, approveDiff, approve2Diff)
head(approveLagDiff)
```

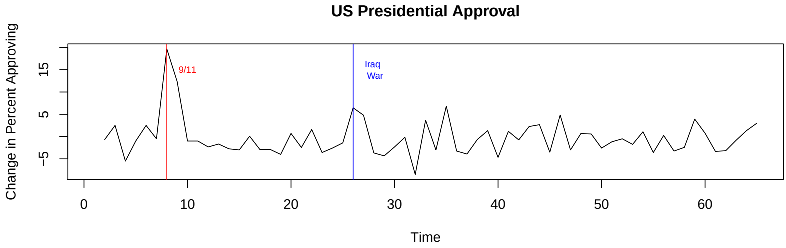
```
##      approve approveLag approve2Lag approveDiff approve2Diff
## [1,]   58.67         NA          NA          NA           NA
## [2,]   58.00      58.67          NA       -0.67           NA
## [3,]   60.50      58.00       58.67        2.50         1.83
## [4,]   55.00      60.50       58.00       -5.50        -3.00
## [5,]   54.00      55.00       60.50       -1.00        -6.50
## [6,]   56.50      54.00       55.00        2.50         1.50
```

```r
avg.price2Lag <- c(NA, NA, avg.price[2:length(avg.price)-2])
avg.price2Lag2 <- as.vector(Lag(avg.price,k=2))

avg.price2Diff <- avg.price - avg.price2Lag
avg.priceLagDiff <- cbind(avg.price, avg.priceLag, avg.price2Lag, avg.priceDiff, avg.price2Diff)
head(avg.priceLagDiff)
```

```
##      avg.price avg.priceLag avg.price2Lag avg.priceDiff avg.price2Diff
## [1,]   144.975           NA            NA            NA             NA
## [2,]   140.925      144.975            NA        -4.050             NA
## [3,]   155.160      140.925       144.975        14.235         10.185
## [4,]   170.175      155.160       140.925        15.015         29.250
## [5,]   161.625      170.175       155.160        -8.550          6.465
## [6,]   142.060      161.625       170.175       -19.565        -28.115
```

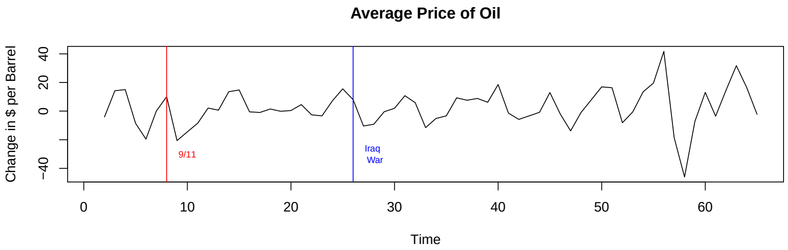
Differencing the Time Series

```r
#Look at the DIFFERENCED time series
par(mfrow=c(2,1))
plot(approveDiff,type="l",ylab="Change in Percent Approving",xlab="Time",
     main = "US Presidential Approval")
lines(x=c(8,8),y=c(-1000,1000),col="red")
lines(x=c(26,26),y=c(-1000,1000),col="blue")
text("9/11",x = 10, y = 15, col="red",cex=0.7)
text("Iraq \n War",x = 28, y = 15, col="blue",cex=0.7)
```



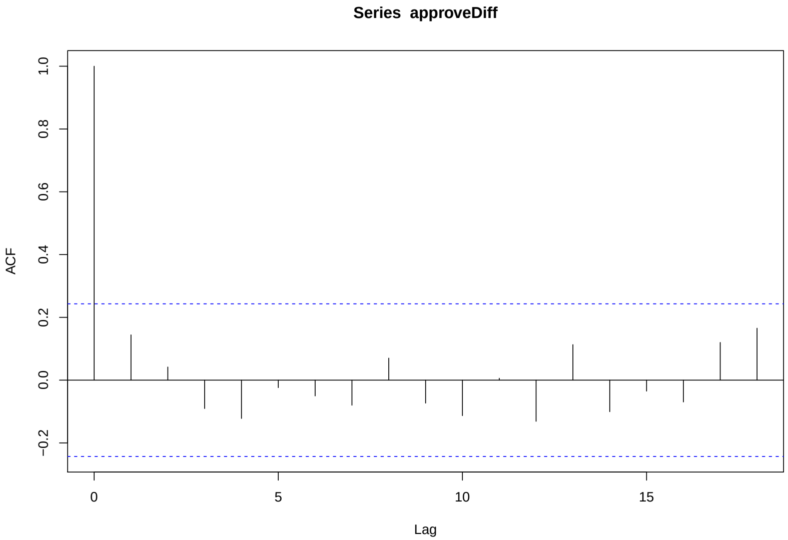
Differencing the Time Series

```
par(mfrow=c(2,1))
plot(avg.priceDiff,type="l",ylab="Change in $ per Barrel",xlab="Time",
     main = "Average Price of Oil")
lines(x=c(8,8),y=c(-1000,1000),col="red")
lines(x=c(26,26),y=c(-1000,1000),col="blue")
text("9/11",x = 10, y = -30, col="red",cex=0.7)
text("Iraq \n War",x = 28, y = -30, col="blue",cex=0.7)
```



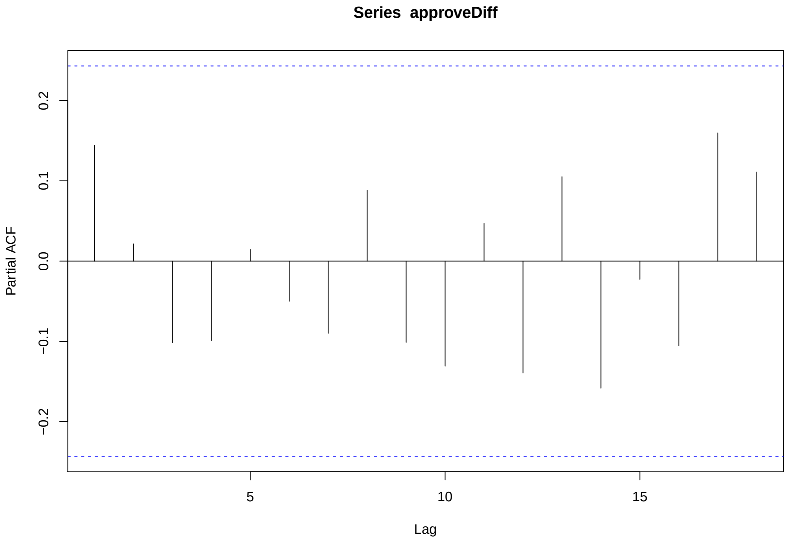
Differencing the Time Series

```r
#Look at the new ACF and PACF for approval
acf(approveDiff, na.action=na.pass)
```



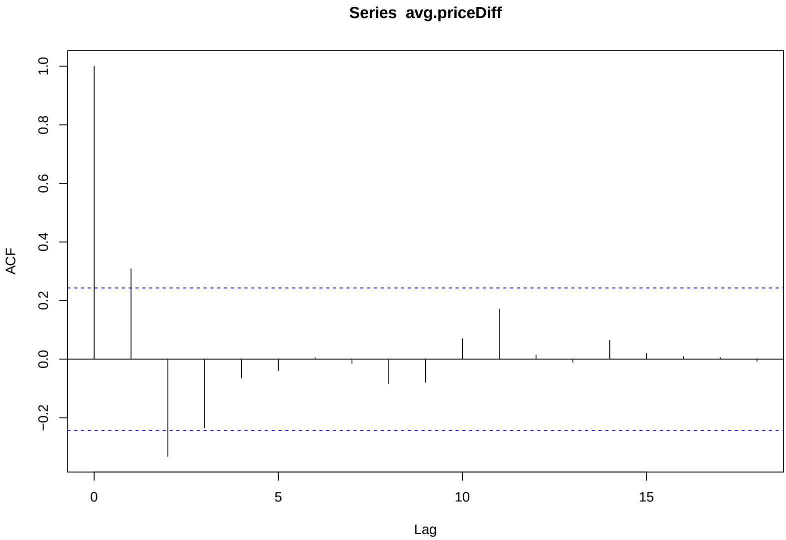
Differencing the Time Series

```r
pacf(approveDiff, na.action=na.pass)
```



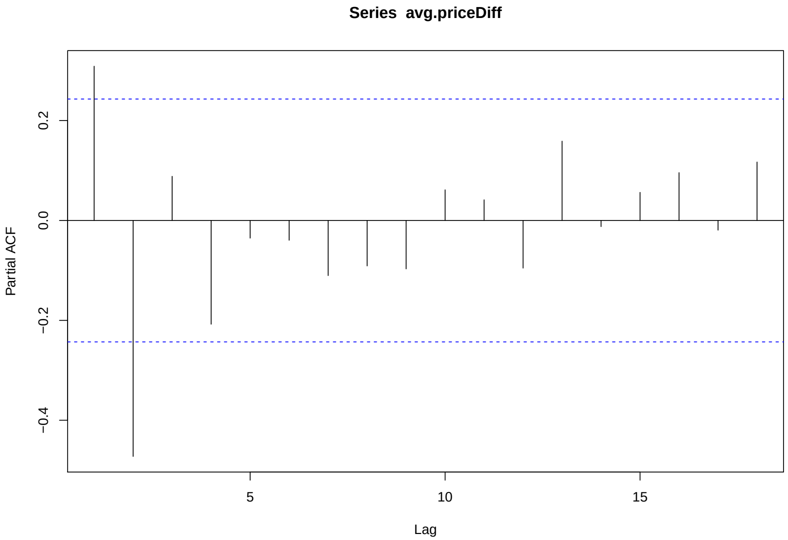
Differencing the Time Series

```
#Look at the new ACF and PACF for oil prices
acf(avg.priceDiff, na.action=na.pass)
```



Differencing the Time Series

```
pacf(avg.priceDiff, na.action=na.pass)
```



Differencing the Time Series

```r
# Check for a unit root in differenced time series
PP.test(as.vector(na.omit(approveDiff)))
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  as.vector(na.omit(approveDiff))
## Dickey-Fuller = -6.703, Truncation lag parameter = 3, p-value =
## 0.01
```

```r
adf.test(na.omit(approveDiff))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(approveDiff)
## Dickey-Fuller = -4.3461, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

Differencing the Time Series

```r
PP.test(as.vector(na.omit(avg.priceDiff)))
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  as.vector(na.omit(avg.priceDiff))
## Dickey-Fuller = -5.4685, Truncation lag parameter = 3, p-value =
## 0.01
```

```r
adf.test(na.omit(avg.priceDiff))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(avg.priceDiff)
## Dickey-Fuller = -5.3361, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

Differencing the Time Series

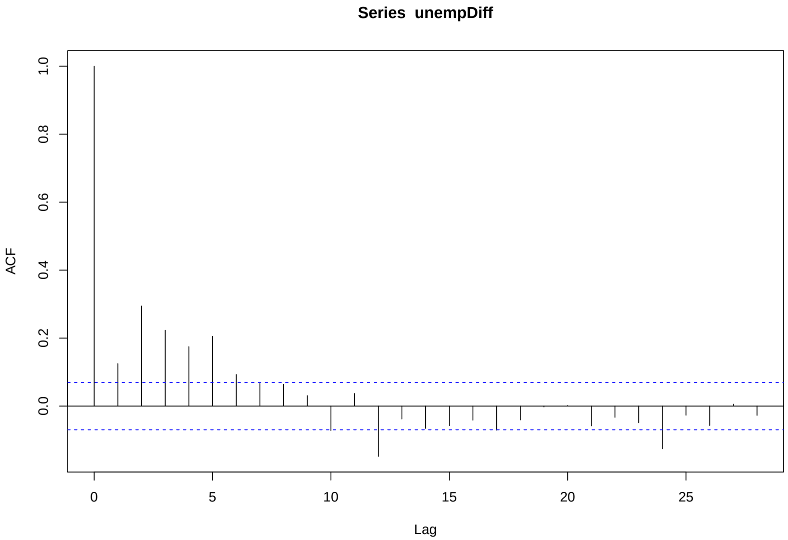
```r
unempLag <- c(NA, rate[1:(length(rate)-1)])
unempLag2 <- as.vector(Lag(rate,k=1))
unempDiff <- rate - unempLag
unempLagDiff <- cbind(rate, unempLag, unempDiff)
head(unempLagDiff)
```

```
##      rate unempLag unempDiff
## [1,]  3.4       NA        NA
## [2,]  3.8      3.4       0.4
## [3,]  4.0      3.8       0.2
## [4,]  3.9      4.0      -0.1
## [5,]  3.5      3.9      -0.4
## [6,]  3.6      3.5       0.1
```

```r
par(mfrow=c(2,1))
plot(unempDiff,type="l",ylab="Change in Unemployment Rate",xlab="Time",
     main = "Unemployment in the US, 1948-2013")
```

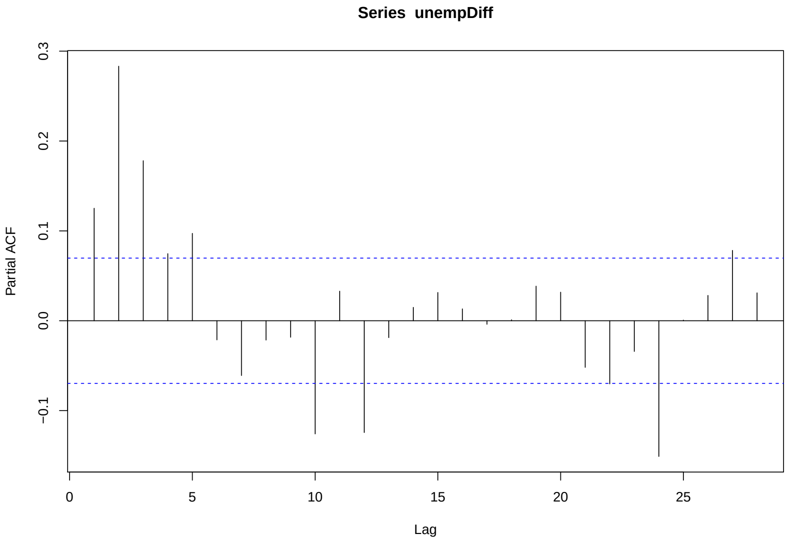
Differencing the Time Series

```
#Look at the new ACF and PACF for unemployment
acf(unempDiff, na.action=na.pass)
```



Differencing the Time Series

```
pacf(unempDiff, na.action=na.pass)
```



Differencing the Time Series

```r
PP.test(as.vector(na.omit(unempDiff)))
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  as.vector(na.omit(unempDiff))
## Dickey-Fuller = -26.554, Truncation lag parameter = 6, p-value =
## 0.01
```

```r
adf.test(na.omit(unempDiff))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(unempDiff)
## Dickey-Fuller = -8.2076, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

Estimation and Model Selection

```
#Estimation and model selection

## Model 1a:  ARIMA(0,1,0) model of approve
##

## Estimate an ARIMA(0,1,0) using arima
xcovariates <- cbind(sept.oct.2001, iraq.war, avg.price)
arima.res1a <- arima(approve, order = c(0,1,0),
                     xreg = xcovariates, include.mean = TRUE
                     )
print(arima.res1a)
```

```
##
## Call:
## arima(x = approve, order = c(0, 1, 0), xreg = xcovariates, include.mean = TRUE)
##
## Coefficients:
##       sept.oct.2001  iraq.war  avg.price
##             11.2072    5.6899    -0.0710
## s.e.         2.5192    2.4891     0.0337
##
## sigma^2 estimated as 12.35:  log likelihood = -171.25,  aic = 350.5
```

Estimation and Model Selection

```r
# Extract estimation results from arima.res1a
pe.1a <- arima.res1a$coef                    # parameter estimates (betas)
se.1a <- sqrt(diag(arima.res1a$var.coef))    # standard errors
ll.1a <- arima.res1a$loglik                  # log likelihood at its maximum
sigma2hat.1a <- arima.res1a$sigma2           # standard error of the regression
aic.1a <- arima.res1a$aic                    # Akaike Information Criterion
resid.1a <- arima.res1a$resid                # residuals
```

Estimation and Model Selection

```r
## Model 1b:  ARIMA(1,1,0) model of approve
##

## Estimate an ARIMA(1,1,0) using arima
xcovariates <- cbind(sept.oct.2001, iraq.war, avg.price)
arima.res1b <- arima(approve, order = c(1,1,0),
                     xreg = xcovariates, include.mean = TRUE
                     )
print(arima.res1b)
```

```
##
## Call:
## arima(x = approve, order = c(1, 1, 0), xreg = xcovariates, include.mean = TRUE)
##
## Coefficients:
##          ar1  sept.oct.2001  iraq.war  avg.price
##       0.0701        11.1213    5.3167    -0.0669
## s.e.  0.1320         2.5128    2.5723     0.0352
##
## sigma^2 estimated as 12.3:  log likelihood = -171.11,  aic = 352.22
```

Estimation and Model Selection

```r
# Extract estimation results from arima.res1a
pe.1b <- arima.res1b$coef                     # parameter estimates (betas)
se.1b <- sqrt(diag(arima.res1b$var.coef))     # standard errors
ll.1b <- arima.res1b$loglik                   # log likelihood at its maximum
sigma2hat.1b <- arima.res1b$sigma2            # standard error of the regression
aic.1b <- arima.res1b$aic                     # Akaike Information Criterion
resid.1b <- arima.res1b$resid                 # residuals
```

Estimation and Model Selection

```r
## Model 1c:  ARIMA(2,1,2) model of approve
##

## Estimate an ARIMA(2,1,2) using arima
xcovariates <- cbind(sept.oct.2001, iraq.war, avg.price)
arima.res1c <- arima(approve, order = c(2,1,2),
                     xreg = xcovariates, include.mean = TRUE
                     )
print(arima.res1c)
```

```
##
## Call:
## arima(x = approve, order = c(2, 1, 2), xreg = xcovariates, include.mean = TRUE)
##
## Coefficients:
##           ar1      ar2     ma1    ma2  sept.oct.2001  iraq.war  avg.price
##       -0.7338  -0.6364  0.8715  1.000         9.7795    5.7884    -0.0511
## s.e.   0.1382   0.1162  0.0672  0.069         1.9202    2.7251     0.0307
##
## sigma^2 estimated as 10.26:  log likelihood = -167.42,  aic = 350.84
```

Estimation and Model Selection

```
# Extract estimation results from arima.res1a
pe.1c <- arima.res1c$coef                       # parameter estimates (betas)
se.1c <- sqrt(diag(arima.res1c$var.coef))       # standard errors
ll.1c <- arima.res1c$loglik                     # log likelihood at its maximum
sigma2hat.1c <- arima.res1c$sigma2              # standard error of the regression
aic.1c <- arima.res1c$aic                       # Akaike Information Criterion
resid.1c <- arima.res1c$resid                   # residuals
```

Estimation and Model Selection

```r
## Model 1d:  ARIMA(0,1,0) model of approve including a spurrious regressor
##

## Estimate an ARIMA(0,1,0) using arima
xcovariates <- cbind(sept.oct.2001, iraq.war, avg.price, phony)
arima.res1d <- arima(approve, order = c(0,1,0),
                     xreg = xcovariates, include.mean = TRUE
                     )
print(arima.res1d)
```

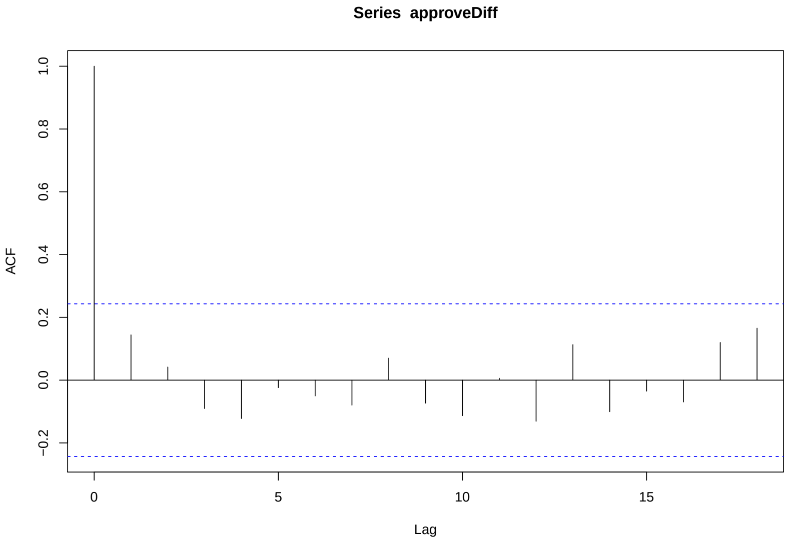
```
##
## Call:
## arima(x = approve, order = c(0, 1, 0), xreg = xcovariates, include.mean = TRUE)
##
## Coefficients:
##       sept.oct.2001  iraq.war  avg.price    phony
##             11.3747    5.2898    -0.0760  -0.9433
## s.e.         2.4553    2.4338     0.0329   0.5065
##
## sigma^2 estimated as 11.72:  log likelihood = -169.56,  aic = 349.12
```

Estimation and Model Selection

```r
# Extract estimation results from arima.res1a
pe.1d <- arima.res1d$coef                    # parameter estimates (betas)
se.1d <- sqrt(diag(arima.res1d$var.coef))    # standard errors
ll.1d <- arima.res1d$loglik                  # log likelihood at its maximum
sigma2hat.1d <- arima.res1d$sigma2           # standard error of the regression
aic.1d <- arima.res1d$aic                    # Akaike Information Criterion
resid.1d <- arima.res1d$resid                # residuals

#Based on ACF, PACF, and AIC, let's select Model 1a to be Model 1
```

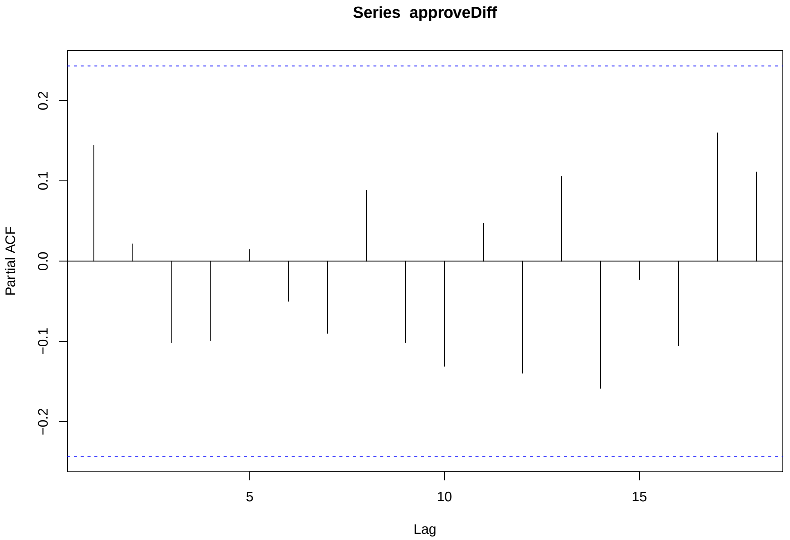
Estimation and Model Selection

```
acf(approveDiff, na.action=na.pass)
```



Estimation and Model Selection

```
pacf(approveDiff, na.action=na.pass)
```



Estimation and Model Selection

```
arima.res1a$aic
```

```
## [1] 350.5013
```

```
arima.res1b$aic
```

```
## [1] 352.2207
```

```
arima.res1c$aic
```

```
## [1] 350.8394
```

```
arima.res1d$aic
```

```
## [1] 349.1238
```

Estimation and Model Selection

```
## What would happen if we used linear regression on a single lag of approval?
lm.res1e <- lm(approve ~ approveLag + sept.oct.2001 + iraq.war + avg.price)
print(summary(lm.res1e))
```

```
##
## Call:
## lm(formula = approve ~ approveLag + sept.oct.2001 + iraq.war +
##     avg.price)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4501 -2.0149 -0.0019  2.0957  5.9562
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   14.06619    5.20581   2.702  0.00899 **
## approveLag     0.83429    0.04988  16.725  < 2e-16 ***
## sept.oct.2001 17.28452    1.97765   8.740 3.11e-12 ***
## iraq.war       4.16158    1.62395   2.563  0.01296 *
## avg.price     -0.03131    0.01427  -2.195  0.03213 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.721 on 59 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9608, Adjusted R-squared:  0.9582
## F-statistic:   362 on 4 and 59 DF,  p-value: < 2.2e-16
```

Estimation and Model Selection

```r
# linear regression with a spurious regressor?
lm.res1f <- lm(approve ~ approveLag + sept.oct.2001 + iraq.war + avg.price + phony)
print(summary(lm.res1f))
```

```
##
## Call:
## lm(formula = approve ~ approveLag + sept.oct.2001 + iraq.war +
##     avg.price + phony)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7439 -1.9322 -0.0066  1.9619  6.0529
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   14.26313    5.21775   2.734  0.00829 **
## approveLag     0.82427    0.05115  16.115  < 2e-16 ***
## sept.oct.2001 17.48380    1.99251   8.775 3.13e-12 ***
## iraq.war       4.03040    1.63262   2.469  0.01653 *
## avg.price     -0.02717    0.01499  -1.813  0.07504 .
## phony         -0.19967    0.21899  -0.912  0.36567
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.724 on 58 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9614, Adjusted R-squared:  0.9581
## F-statistic: 288.9 on 5 and 58 DF,  p-value: < 2.2e-16
```

Estimation and Model Selection

```r
# Check LS result for serial correlation in the first or second order (null of no serial correlation)
bgtest(lm.res1e,1)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lm.res1e
## LM test = 4.9524, df = 1, p-value = 0.02605
```

```r
bgtest(lm.res1f,2)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 2
##
## data:  lm.res1f
## LM test = 4.4133, df = 2, p-value = 0.1101
```

Forecasting

```r
#### Because Zelig's arima simulator isn't currently available,
#### Let's use ldvsimev().

# First we need the simulated parameters
sims <- 10000
pe <- arima.res1a$coef
vc <- arima.res1a$var.coef
simparams <- mvrnorm(sims, pe, vc)
simbetas <- simparams[, (1+sum(arima.res1a$arma[1:2])):ncol(simparams)]
#simphi <- simparams[, 1:arima.res1$arma[1]]  # if AR(1) or greater

# Choose counterfactual periods
periodsToSim <- seq(from=26, to=65, by=1) #March 2003 to June 2006
nPeriodsToSim <- length(periodsToSim)

## Create hypothetical covariates over these periods
# Start with factual values
# For ARIMA, need to enter these covariates in same order as model
model <- approve ~ sept.oct.2001 + iraq.war + avg.price
selectdata <- extractdata(model, approval, na.rm = TRUE)
perioddata <- selectdata[periodsToSim,]
```

Forecasting

```r
# Treat factual data as the baseline counterfactual
xhyp0 <- subset(perioddata, select = -approve )

# Suppose no Iraq War occurred
xhyp <- subset(perioddata, select = -approve )
xhyp$iraq.war <- rep(0,nPeriodsToSim)

# Leave out phi because this specification has no AR component
# phi <- simphi

# Construction of prior lags depends on order of ARIMA
# Only need initialY for I(1)
initialY <- selectdata$approve[periodsToSim[1] - 1]
# Original level of the response (for differenced models)
# if ARIMA(1,1,0), instead: selectdata$approve[periodsToSim[1] - 2]
# Only need lagY for AR(1) or higher
lagY <- selectdata$approve[periodsToSim[1] - 1]
# The prior levels of y (scalar or vector)
# if ARIMA(1,1,0), append: - selectdata$approve[periodsToSim[1] - 2]
```

Forecasting

```r
# Simulate expected values of Y out to periods
# given hypothetical values of X, and an initial level of Y
# No Iraq scenario
noIraq.ev1 <- ldvsimev(xhyp,              # The matrix of hypothetical x's
                       simbetas,           # The matrix of simulated betas
                       ci=0.95,            # Desired confidence interval
                       constant=NA,        # Column containing the constant;
                                           #  no constant because model is differenced
                       #phi=phi,           # estimated AR parameters; length must match lagY
                       #lagY=lagY,         # lags of y, most recent last
                       transform="diff",   # "log" to undo log transformation,
                                           # "diff" to under first differencing
                                           # "difflog" to do both
                       initialY=initialY   # for differenced models, lag of level of y
                       )
```

Forecasting

```
# Iraq scenario (factual)
Iraq.ev1 <- ldvsimev(xhyp0,               # The matrix of hypothetical x's
                     simbetas,            # The matrix of simulated betas
                     ci=0.95,             # Desired confidence interval
                     constant=NA,         # Column containing the constant;
                                          #  no constant because model is differenced
                     #phi=phi,            # estimated AR parameters; length must match lagY
                     #lagY=lagY,          # lags of y, most recent last
                     transform="diff",    # "log" to undo log transformation,
                                          # "diff" to under first differencing
                                          # "difflog" to do both
                     initialY=initialY    # for differenced models, lag of level of y
                     )
```

Forecasting

```r
at.xaxis <- seq(from = -1, to = length(approve[1:25]) + nPeriodsToSim, by = 12)
lab.xaxis <- seq(from = 2001, by = 1, length.out = length(at.xaxis))

pdf("noIraqARIMA.pdf",width=6,height=3.25)
ctrfactTS(observed = approve[1:25],
          predicted = noIraq.ev1$pe,
          lower = noIraq.ev1$lower,
          upper = noIraq.ev1$upper,
          #se = NULL,
          predicted0 = Iraq.ev1$pe,
          lower0 = Iraq.ev1$lower,
          upper0 = Iraq.ev1$upper,
          #se0 = NULL,
          factual = approve[26:65],
          at.xaxis = at.xaxis,
          lab.xaxis = lab.xaxis,
          #ylim = c(0, 100),
          main = "Ctrfactual Effect of No Iraq War from ARIMA(0,1,0)",
          xlab = "Year",
          ylab = "Presidential Approval (%)",
          col = "red",
          col0 = "blue")
dev.off()
```

```
## pdf
##   2
```

Forecasting

```r
# What if we used an ARMA model in this case (ignored possible
# unit root)?  Would we get better long run estimates?

## Model 2a:  ARIMA(1,0,0) model of approve
##

## Estimate an ARIMA(1,0,0) using arima
xcovariates <- cbind(sept.oct.2001, iraq.war, avg.price)
arima.res2a <- arima(approve, order = c(1,0,0),
                     xreg = xcovariates, include.mean = TRUE
                     )
print(arima.res2a)
```

```
##
## Call:
## arima(x = approve, order = c(1, 0, 0), xreg = xcovariates, include.mean = TRUE)
##
## Coefficients:
##         ar1  intercept  sept.oct.2001  iraq.war  avg.price
##       0.918    71.4311        11.5152    5.8123    -0.0892
## s.e.  0.049     8.2425         2.5738    2.5299     0.0357
##
## sigma^2 estimated as 11.82:  log likelihood = -173.43,  aic = 358.86
```

Forecasting

```
# Extract estimation results from arima.res1a
pe.2a <- arima.res2a$coef                          # parameter estimates (betas)
se.2a <- sqrt(diag(arima.res2a$var.coef))          # standard errors
ll.2a <- arima.res2a$loglik                        # log likelihood at its maximum
sigma2hat.2a <- arima.res2a$sigma2                 # standard error of the regression
aic.2a <- arima.res2a$aic                          # Akaike Information Criterion
resid.2a <- arima.res2a$resid                      # residuals

# First we need the simulated parameters
sims <- 10000
pe <- arima.res2a$coef
vc <- arima.res2a$var.coef
simparams <- mvrnorm(sims, pe, vc)
simbetas <- simparams[, (1+sum(arima.res2a$arma[1:2])):ncol(simparams)]
simphi <- simparams[, 1:arima.res2a$arma[1]]  # if AR(1) or greater
```

Forecasting

```r
# Extract estimation results from arima.res1a
pe.2a <- arima.res2a$coef                         # parameter estimates (betas)
se.2a <- sqrt(diag(arima.res2a$var.coef))    # standard errors
ll.2a <- arima.res2a$loglik                       # log likelihood at its maximum
sigma2hat.2a <- arima.res2a$sigma2              # standard error of the regression
aic.2a <- arima.res2a$aic                         # Akaike Information Criterion
resid.2a <- arima.res2a$resid                     # residuals

# First we need the simulated parameters
sims <- 10000
pe <- arima.res2a$coef
vc <- arima.res2a$var.coef
simparams <- mvrnorm(sims, pe, vc)
simbetas <- simparams[, (1+sum(arima.res2a$arma[1:2])):ncol(simparams)]
simphi <- simparams[, 1:arima.res2a$arma[1]]  # if AR(1) or greater
```

Forecasting

```
# Extract estimation results from arima.res1a
pe.2a <- arima.res2a$coef                      # parameter estimates (betas)
se.2a <- sqrt(diag(arima.res2a$var.coef))     # standard errors
ll.2a <- arima.res2a$loglik                    # log likelihood at its maximum
sigma2hat.2a <- arima.res2a$sigma2             # standard error of the regression
aic.2a <- arima.res2a$aic                      # Akaike Information Criterion
resid.2a <- arima.res2a$resid                  # residuals

# First we need the simulated parameters
sims <- 10000
pe <- arima.res2a$coef
vc <- arima.res2a$var.coef
simparams <- mvrnorm(sims, pe, vc)
simbetas <- simparams[, (1+sum(arima.res2a$arma[1:2])):ncol(simparams)]
simphi <- simparams[, 1:arima.res2a$arma[1]]  # if AR(1) or greater
```

Forecasting

```r
# Choose counterfactual periods
periodsToSim <- seq(from=26, to=65, by=1)
nPeriodsToSim <- length(periodsToSim)

## Create hypothetical covariates over these periods
# Start with factual values
# For ARIMA, need to enter these covariates in same order as model
model <- approve ~ sept.oct.2001 + iraq.war + avg.price
selectdata <- extractdata(model, approval, na.rm = TRUE)
perioddata <- selectdata[periodsToSim,]

# Treat factual data as the baseline counterfactual
xhyp0 <- subset(perioddata, select = -approve )

# Suppose no Iraq War occurred
xhyp <- subset(perioddata, select = -approve )
xhyp$iraq.war <- rep(0,nPeriodsToSim)

# Construction of prior lags depends on order of ARIMA
# Only need initialY for I(1)
# Only need lagY for AR(1) or higher
lagY <- selectdata$approve[periodsToSim[1] - 1]
```

Forecasting

```r
# Simulate expected values of Y out to periods
# given hypothetical values of X, and an initial level of Y
# No Iraq scenario
noIraq.ev1 <- ldvsimev(xhyp,               # The matrix of hypothetical x's
                       simbetas,           # The matrix of simulated betas
                       ci=0.95,            # Desired confidence interval
                       constant=1,         # Column containing the constant;
                       phi=mean(simphi),   # estimated AR parameters; length must match lagY
                       lagY=lagY           # lags of y, most recent last
                       )

# Iraq scenario (factual)
Iraq.ev1 <- ldvsimev(xhyp0,              # The matrix of hypothetical x's
                     simbetas,           # The matrix of simulated betas
                     ci=0.95,            # Desired confidence interval
                     constant=1,         # Column containing the constant;
                     phi=mean(simphi),   # estimated AR parameters; length must match lagY
                     lagY=lagY           # lags of y, most recent last
                     )
```

Forecasting

```r
at.xaxis <- seq(from = -1, to = length(approve[1:25]) + nPeriodsToSim, by = 12)
lab.xaxis <- seq(from = 2001, by = 1, length.out = length(at.xaxis))

pdf("noIraqARMA.pdf",width=6,height=3.25)
ctrfactTS(observed = approve[1:25],
          predicted = noIraq.ev1$pe,
          lower = noIraq.ev1$lower,
          upper = noIraq.ev1$upper,
          #se = NULL,
          predicted0 = Iraq.ev1$pe,
          lower0 = Iraq.ev1$lower,
          upper0 = Iraq.ev1$upper,
          #se0 = NULL,
          factual = approve[26:65],
          at.xaxis = at.xaxis,
          lab.xaxis = lab.xaxis,
          #ylim = c(0, 100),
          main = "Ctrfactual Effect of No Iraq War (Red) from AR(1)",
          xlab = "Year",
          ylab = "Presidential Approval (%)",
          col = "red",
          col0 = "blue")
dev.off()
```

```
## pdf
##   2
```

Forecasting

```r
#What if we used the linear regression model with a lagged DV?

lm.res3 <- lm(approve ~ approveLag + sept.oct.2001 + iraq.war + avg.price)
print(summary(lm.res3))
```

```
##
## Call:
## lm(formula = approve ~ approveLag + sept.oct.2001 + iraq.war +
##     avg.price)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4501 -2.0149 -0.0019  2.0957  5.9562
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   14.06619    5.20581   2.702  0.00899 **
## approveLag     0.83429    0.04988  16.725  < 2e-16 ***
## sept.oct.2001 17.28452    1.97765   8.740 3.11e-12 ***
## iraq.war       4.16158    1.62395   2.563  0.01296 *
## avg.price     -0.03131    0.01427  -2.195  0.03213 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.721 on 59 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.9608, Adjusted R-squared:  0.9582
## F-statistic:   362 on 4 and 59 DF,  p-value: < 2.2e-16
```

Forecasting

```r
#Extract estimation results from arima.res1a
pe.3 <- coef(lm.res3)                # parameter estimates
vc.3 <- vcov(lm.res3)                # variance-covariance (of params)
se.3 <- sqrt(diag(vc.3))             # standard errors

#First we need the simulated parameters
sims <- 10000
simparams <- mvrnorm(sims, pe.3, vc.3)
# Pluck out the lag parameter
simphi <- simparams[,2]
# Get the rest of the betas
simbetas <- simparams[,c(1,3:ncol(simparams))]

#Choose counterfactual periods
periodsToSim <- seq(from=26, to=65, by=1)
nPeriodsToSim <- length(periodsToSim)
```

Forecasting

```r
#Create hypothetical covariates over these periods
# Start with factual values (note I leave out the lag;
# ldvsimev() will construct it)
model <- approve ~ sept.oct.2001 + iraq.war + avg.price
selectdata <- extractdata(model, approval, na.rm = TRUE)
perioddata <- selectdata[periodsToSim,]

#Treat factual data as the baseline counterfactual
xhyp0 <- subset(perioddata, select = -approve )

#Suppose no Iraq War occurred
xhyp <- subset(perioddata, select = -approve )
xhyp$iraq.war <- rep(0,nPeriodsToSim)

#Construction of prior lags depends on order of ARIMA
#Only need initialY for I(1)
#Only need lagY for AR(1) or higher
lagY <- selectdata$approve[periodsToSim[1] - 1]
```

Forecasting

```r
#Simulate expected values of Y out to periods
#given hypothetical values of X, and an initial level of Y
#No Iraq scenario
noIraq.ev1 <- ldvsimev(xhyp,               # The matrix of hypothetical x's
                       simbetas,           # The matrix of simulated betas
                       ci=0.95,            # Desired confidence interval
                       constant=1,         # Column containing the constant;
                       phi=mean(simphi),   # estimated AR parameters; length must match lagY
                       lagY=lagY           # lags of y, most recent last
                       )

#Iraq scenario (factual)
Iraq.ev1 <- ldvsimev(xhyp0,              # The matrix of hypothetical x's
                     simbetas,           # The matrix of simulated betas
                     ci=0.95,            # Desired confidence interval
                     constant=1,         # Column containing the constant;
                     phi=mean(simphi),   # estimated AR parameters; length must match lagY
                     lagY=lagY           # lags of y, most recent last
                     )
```

Forecasting

```r
at.xaxis <- seq(from = -1, to = length(approve[1:25]) + nPeriodsToSim, by = 12)
lab.xaxis <- seq(from = 2001, by = 1, length.out = length(at.xaxis))

pdf("noIraqLS.pdf",width=6,height=3.25)
ctrfactTS(observed = approve[1:25],
          predicted = noIraq.ev1$pe,
          lower = noIraq.ev1$lower,
          upper = noIraq.ev1$upper,
          #se = NULL,
          predicted0 = Iraq.ev1$pe,
          lower0 = Iraq.ev1$lower,
          upper0 = Iraq.ev1$upper,
          #se0 = NULL,
          factual = approve[26:65],
          at.xaxis = at.xaxis,
          lab.xaxis = lab.xaxis,
          #ylim = c(0, 100),
          main = "Ctrfactual Effect of No Iraq War (Red) from LS with Lagged DV",
          xlab = "Year",
          ylab = "Presidential Approval (%)",
          col = "red",
          col0 = "blue")
dev.off()
```

```
## pdf
##   2
```

Cointegration Analysis

Thus far, we have been examining a single time series with covariates. This assumes that there is no feedback between variables.

Yet, we may be interested in the relationship between two potentially nonstationary time series that influence each other.

The original idea behind cointegration is that two time series may be in equilibrium in the long run but in the short run the two series deviate from that equilibrium.

Cointegrated time series are two nonstationary time series that are causally connected and do not tend toward any particular level but tend toward each other.

Cointegration means that a specific combination of two nonstationary series may be stationary. We say that these two series are cointegrated and the vector that defines the stationary linear combination is called the cointegrating vector.

Cointegration Analysis

```
set.seed(123456)

# Generate cointegrated data
e1 <- rnorm(100)
e2 <- rnorm(100)
x <- cumsum(e1)
y <- 0.6*x + e2

#Run step 1 of the Engle-Granger two step
coint.reg <- lm(y ~ x -1)
#Estimate the cointegration vector by least squares with no constant
coint.err <- residuals(coint.reg)
#This gives us the cotingeration vector

#Check for stationarity of the cointegration vector
punitroot(adf.test(coint.err)$statistic, trend="nc")
```

```
## Dickey-Fuller
##  6.551997e-05
```

```
#Make the lag of the cointegration error term
coint.err.lag <- coint.err[1:(length(coint.err)-2)]

#Make the difference of y and x
dy <- diff(y)
dx <- diff(x)

#And their lags
dy.lag <- dy[1:(length(dy)-1)]
dx.lag <- dx[1:(length(dx)-1)]

#Delete the first dy, because we are missing lags for this obs
dy <- dy[2:length(dy)]
```

Cointegration Analysis

```r
 #Estimate an Error Correction Model with LS
ecm1 <- lm(dy ~ coint.err.lag + dy.lag + dx.lag)
summary(ecm1)
```

```
##
## Call:
## lm(formula = dy ~ coint.err.lag + dy.lag + dx.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9553 -0.5375  0.1538  0.7042  2.3240
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.02267    0.10381   0.218    0.828
## coint.err.lag -0.96617    0.15864  -6.090 2.45e-08 ***
## dy.lag        -1.05776    0.10848  -9.751 6.21e-16 ***
## dx.lag         0.81035    0.11223   7.221 1.33e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.026 on 94 degrees of freedom
## Multiple R-squared:  0.5456, Adjusted R-squared:  0.5311
## F-statistic: 37.62 on 3 and 94 DF,  p-value: 4.624e-16
```

Cointegration Analysis

```r
#Alternatively, we can use the Johansen estimator
#Create a matrix of the cointegrated variables
cointvars <- cbind(y,x)
# Perform cointegration tests
coint.test1 <- ca.jo(cointvars,
                    ecdet = "const",
                    type="eigen",
                    K=2,
                    spec="longrun")
```

Cointegration Analysis

```r
summary(coint.test1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegrat
##
## Eigenvalues (lambda):
## [1] 3.105216e-01 2.077094e-02 3.335727e-18
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  2.06  7.52  9.24 12.97
## r = 0  | 36.44 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                 y.l2      x.l2   constant
## y.l2      1.00000000   1.00000   1.000000
## x.l2     -0.58297186  10.12695  -1.215134
## constant -0.02960597 -50.23990 -38.501184
##
## Weights W:
## (This is the loading matrix)
##
##             y.l2         x.l2     constant
## y.d -0.967714950 -0.001015446 1.784095e-17
## x.d  0.002461222 -0.002817005 1.142521e-18
```

Cointegration Analysis

```r
ecm.res1 <- cajorls(coint.test1,
                    r = 1,           # Cointegration rank
                    reg.number = 1)  # which variable(s) to put on LHS
#(column indexes of cointvars)
summary(ecm.res1)
```

```
##      Length Class  Mode
## rlm  12     lm     list
## beta  3     -none- numeric
```

Cointegration Analysis

```r
#For the presidential approval example, use an ECM equivalent to the
#ARIMA(1,0,1) model that we chose earlier

cointvars <- cbind(approve,avg.price)
ecm.test1 <- ca.jo(cointvars,
                   ecdet = "const",
                   type="eigen",
                   K=2,
                   spec="longrun",
                   dumvar=cbind(sept.oct.2001,iraq.war)
                   )
```

Cointegration Analysis

```
summary(ecm.test1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegrati
##
## Eigenvalues (lambda):
## [1] 2.391542e-01 1.367744e-01 1.387779e-16
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 1 |  9.27  7.52  9.24 12.97
## r = 0  | 17.22 13.75 15.67 20.20
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                approve.l2 avg.price.l2 constant
## approve.l2     1.0000000    1.0000000  1.00000
## avg.price.l2   0.1535049    0.3382829 -1.05616
## constant     -76.0019182 -120.7778161 90.24200
##
## Weights W:
## (This is the loading matrix)
##
##               approve.l2 avg.price.l2     constant
## approve.d   -0.12619985   0.02231967 5.407866e-17
## avg.price.d -0.02220281  -0.58771490 3.727850e-16
```

Cointegration Analysis

```
ecm.res1 <- cajorls(ecm.test1,
                    r = 1,
                    reg.number = 1)
summary(ecm.res1$rlm)
```

```
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1403 -1.6753 -0.2261  1.6430  5.9537
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## ect1           -0.12620    0.03006  -4.198 9.37e-05 ***
## sept.oct.2001 19.55846    2.11737   9.237 5.40e-13 ***
## iraq.war        5.01870    1.62432   3.090  0.00307 **
## approve.dl1    -0.31757    0.09448  -3.361  0.00138 **
## avg.price.dl1 -0.05055    0.02593  -1.949  0.05613 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.668 on 58 degrees of freedom
## Multiple R-squared:  0.6301, Adjusted R-squared:  0.5983
## F-statistic: 19.76 on 5 and 58 DF,  p-value: 1.915e-11
```

Cointegration Analysis

```r
ecm.res1 <- cajorls(ecm.test1,
                    r = 1,
                    reg.number = 1)
summary(ecm.res1$rlm)
```

```
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1403 -1.6753 -0.2261  1.6430  5.9537
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## ect1           -0.12620    0.03006  -4.198 9.37e-05 ***
## sept.oct.2001 19.55846    2.11737   9.237 5.40e-13 ***
## iraq.war        5.01870    1.62432   3.090  0.00307 **
## approve.dl1    -0.31757    0.09448  -3.361  0.00138 **
## avg.price.dl1 -0.05055    0.02593  -1.949  0.05613 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.668 on 58 degrees of freedom
## Multiple R-squared:  0.6301, Adjusted R-squared:  0.5983
## F-statistic: 19.76 on 5 and 58 DF,  p-value: 1.915e-11
```

Cointegration Analysis

```r
#Cointegration analysis with a spurious regressor
cointvars <- cbind(approve,avg.price,phony)

ecm.test1 <- ca.jo(cointvars,
                   ecdet = "const",
                   type="eigen",
                   K=2,
                   spec="longrun",
                   dumvar=cbind(sept.oct.2001,iraq.war)
                   )
```

Cointegration Analysis

```r
summary(ecm.test1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegrat
##
## Eigenvalues (lambda):
## [1] 2.667954e-01 1.759317e-01 1.204386e-01 4.440892e-16
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  8.08  7.52  9.24 12.97
## r <= 1 | 12.19 13.75 15.67 20.20
## r = 0  | 19.55 19.77 22.00 26.81
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                 approve.l2 avg.price.l2    phony.l2    constant
## approve.l2      1.00000000     1.000000   1.0000000    1.000000
## avg.price.l2    0.09790496     0.525264   0.1065106    1.817901
## phony.l2        1.49035973    -4.923281   5.0616688   22.257963
## constant      -70.59385440  -143.177005 -92.9643009 -360.516198
##
## Weights W:
## (This is the loading matrix)
##
##                 approve.l2 avg.price.l2    phony.l2      constant
## approve.d    -0.1309280611 -0.007465878  0.03966997 -4.085651e-17
## avg.price.d   0.0007716341 -0.434815272 -0.14680124 -2.260861e-17
```

Cointegration Analysis

```
ecm.res1 <- cajorls(ecm.test1,
                    r = 1,
                    reg.number = 1)
summary(ecm.res1$rlm)
```

```
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.119 -1.797 -0.305  1.244  6.401
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## ect1          -0.13093    0.03352  -3.905 0.000252 ***
## sept.oct.2001 19.38002    2.16345   8.958 1.81e-12 ***
## iraq.war       4.54226    1.63994   2.770 0.007558 **
## approve.dl1   -0.29503    0.09809  -3.008 0.003912 **
## avg.price.dl1 -0.04917    0.02668  -1.843 0.070505 .
## phony.dl1      0.10540    0.41502   0.254 0.800440
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.719 on 57 degrees of freedom
## Multiple R-squared:  0.6226, Adjusted R-squared:  0.5829
## F-statistic: 15.67 on 6 and 57 DF,  p-value: 1.576e-10
```