DaiglePredictionofSocialSecurity.R

daiglechris

Wed Dec 5 20:26:26 2018

Chris Daigle Prediction of Social Security Awards

```r
# Prepare workspace ####
rm(list = ls())
library(tseries)
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: TTR

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

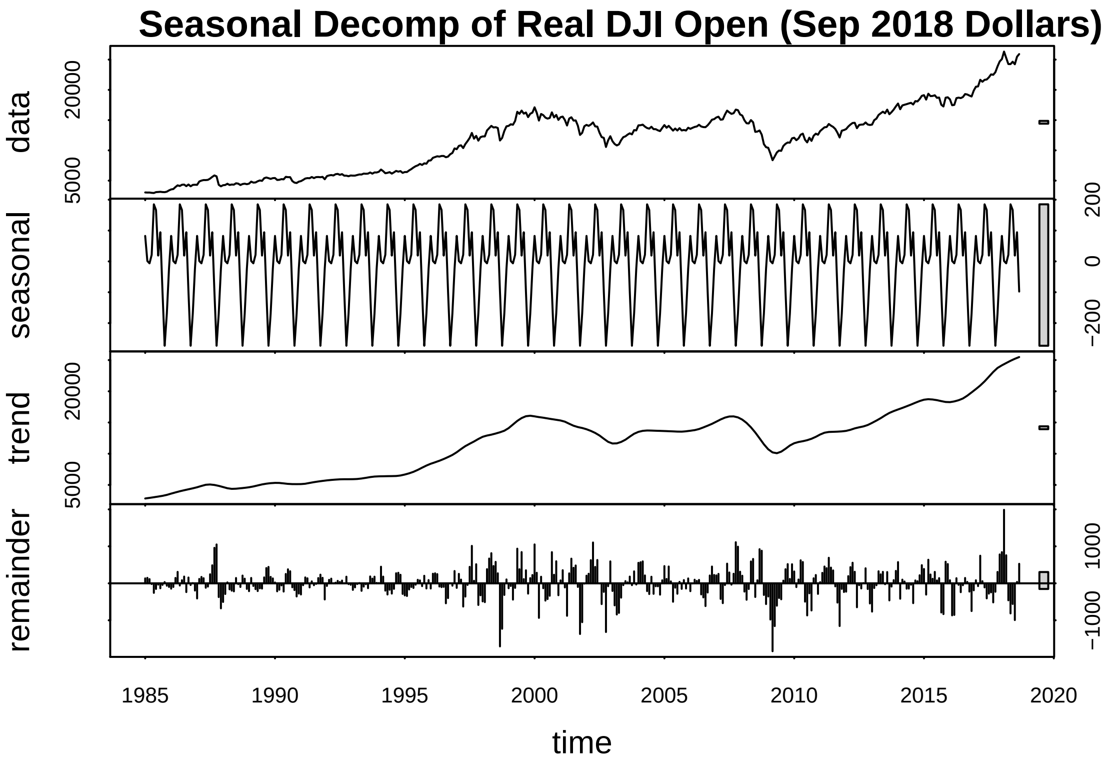
```r
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:xts':
##
##     first, last
```

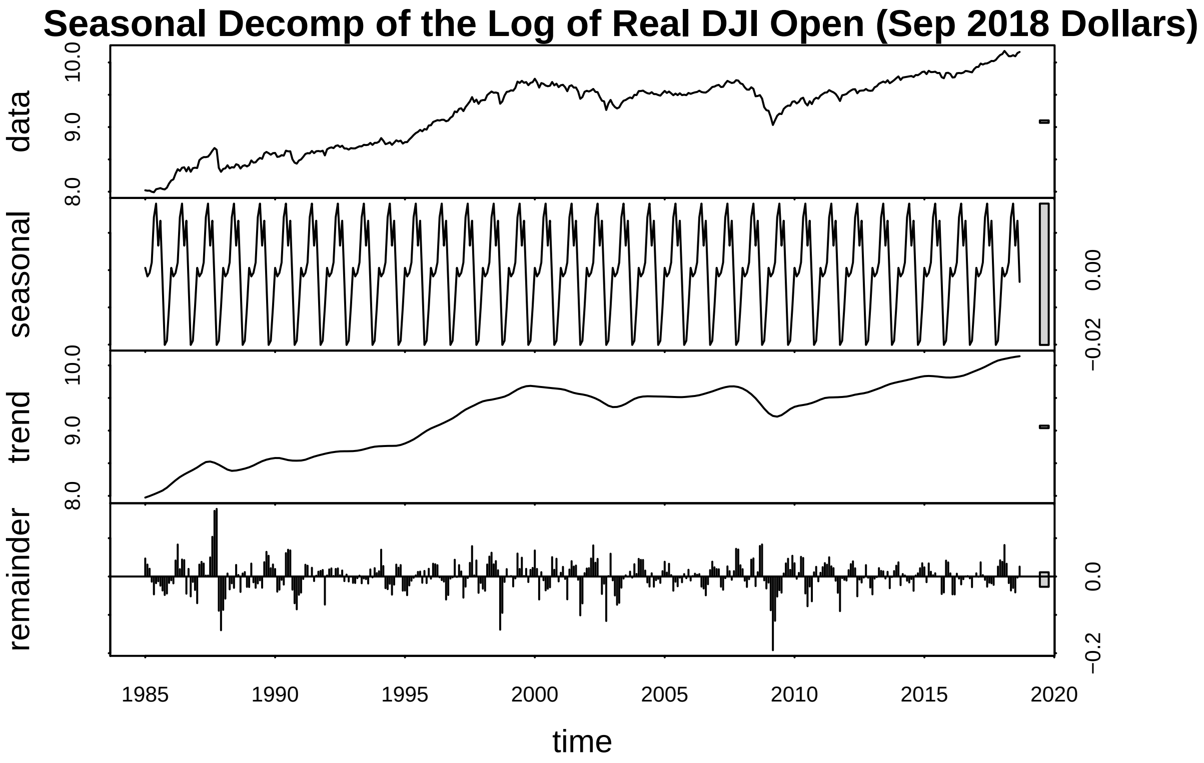
```r
library(leaps)
setwd('~/Git/MachineLearningAndBigDataWithR/Data')
dataName <- 'assembled.csv'
df <- read.csv(dataName, stringsAsFactors = FALSE)
# Summarize and clean data ####
# head(df)
df <- df[-1]
# head(df)
# str(df)

# Variable Manipulation ####
# Set dates
df$date <- as.Date(df$date, "%Y-%m-%d")
# Functions to clean data #
spaceless <- function(x) {
  x <- gsub(" ", ".", x)
  x
}

commaless <- function(x) {
  x <- gsub(",", "", x)
  x
}

dollarless <- function(x) {
  x <- gsub("\\$", "", x)
  x
}

# Loops to apply functions #
for (i in 15:20) {
  df[, i] <- commaless(df[, i])
}
for (i in 15:20) {
  df[, i] <- dollarless(df[, i])
}

# Loop to transform variable types #
for (i in 15:20) {
  df[, i] <- as.numeric(df[, i])
}
# Names with Index ####
# 1 date                     : Date
# 2 DJIopen                  : num
# 3 DJIhigh                  : num
# 4 DJIlow                   : num
# 5 DJIclose                 : num
# 6 DJIadjClose              : num
# 7 DJIvolume                : num
# 8 SPopen                   : num
# 9 SPhigh                   : num
# 10 SPlow                    : num
# 11 SPclose                  : num
# 12 SPadjClose               : num
# 13 SPvolume                 : num
# 14 fedFundRate              : num
# 15 totalSSRetired           : num
# 16 averageSSRetiredPay      : num
# 17 totalMaleSSRetired       : num
# 18 averageMaleSSRetiredPay  : num
# 19 totalFemaleSSRetired     : num
# 20 averageFemaleSSRetiredPay: num
# 21 cpi                      : num
#
# Order Change #
df <- df[, c(1, 15, 17, 19, 21, 14, 7, 13, 2:6, 8:12, 16, 18, 20)]
# 1 date                     : Date
# 2 totalSSRetired           : num
# 3 totalMaleSSRetired       : num
# 4 totalFemaleSSRetired     : num
# 5 cpi                      : num
# 6 fedFundRate              : num
# 7 DJIvolume                : num
# 8 SPvolume                 : num
# 9 DJIopen                  : num
# 10 DJIhigh                  : num
# 11 DJIlow                   : num
# 12 DJIclose                 : num
# 13 DJIadjClose              : num
# 14 SPopen                   : num
# 15 SPhigh                   : num
# 16 SPlow                    : num
# 17 SPclose                  : num
# 18 SPadjClose               : num
# 19 averageSSRetiredPay      : num
# 20 averageMaleSSRetiredPay  : num
# 21 averageFemaleSSRetiredPay: num
#
# Variable Creation ####
# CPI Inflator
latestDate <- tail(df$date, n = 1)

baseCpi <- df$cpi[df$date == latestDate]
df$inflator <- baseCpi / df$cpi

df <- df[, c(1:6, 22, 7:21)]

realNames <-
  paste('real',
        colnames(df[, 10:22]),
        sep = "")

df[, realNames] <- df$inflator * df[10:22]

# Differences #
diffNames <-
  paste('diff',
        c(colnames(df[10:22]),
          paste('Real',
                colnames(df[10:22]),
                sep = "")),
        sep = "")
df[, diffNames] <- rep(NA, nrow(df))
for (i in 36:61) {
  df[, i][2:nrow(df)] <- diff(df[, i - 26], lag = 1)
}
diffTargetNames <-
  paste('diff',
        c(colnames(df[2:4])),
        sep = "")
df[, diffTargetNames] <- rep(NA, nrow(df))
for (i in 62:64) {
  df[, i][2:nrow(df)] <- diff(df[, i - 60], lag = 1)
}

# Positive Indicator #
```

```r
posNames <-
  paste('pos',
        c(colnames(df[10:22]),
          paste('Real',
                colnames(df[10:22]),
                sep = "")),
        sep = "")
df[, posNames] <- rep(0, nrow(df))
for (i in 65:90) {
  df[, i][df[, i - 20] > 0] <- 1
}

posTargetNames <-
  paste('pos',
        c(colnames(df[2:4])),
        sep = "")
df[, posTargetNames] <- rep(0, nrow(df))
for (i in 91:93) {
  df[, i][df[, i - 29] > 0] <- 1
}

# Percent Changes #
percChangeNames <-
  paste('percChange',
        c(colnames(df[10:22]),
          paste('Real', colnames(df[10:22]), sep = "")),
        sep = "")
df[, percChangeNames] <- rep(NA, nrow(df))

for (i in 94:119) {
  df[, i] <- Delt(df[, i - 84])
}
for (i in 94:119) {
  df[, i] <- as.numeric(df[, i])
}
percChangeTargetNames <-
  paste('percChange',
        c(colnames(df[2:4])),
        sep = "")
df[, percChangeTargetNames] <- rep(NA, nrow(df))
for (i in 120:122) {
  df[, i] <- Delt(df[, i - 118])
}
for (i in 120:122) {
  df[, i] <- as.numeric(df[, i])
}

# Place all target variables - totalRetired* - together
df <- df[, c(1:4, 62:64, 91:93, 120:122, 5:61, 65:90, 94:119)]
df1 <- df[complete.cases(df), ]

# Timeseries Evaluation ####
realDJIOpen <-
```

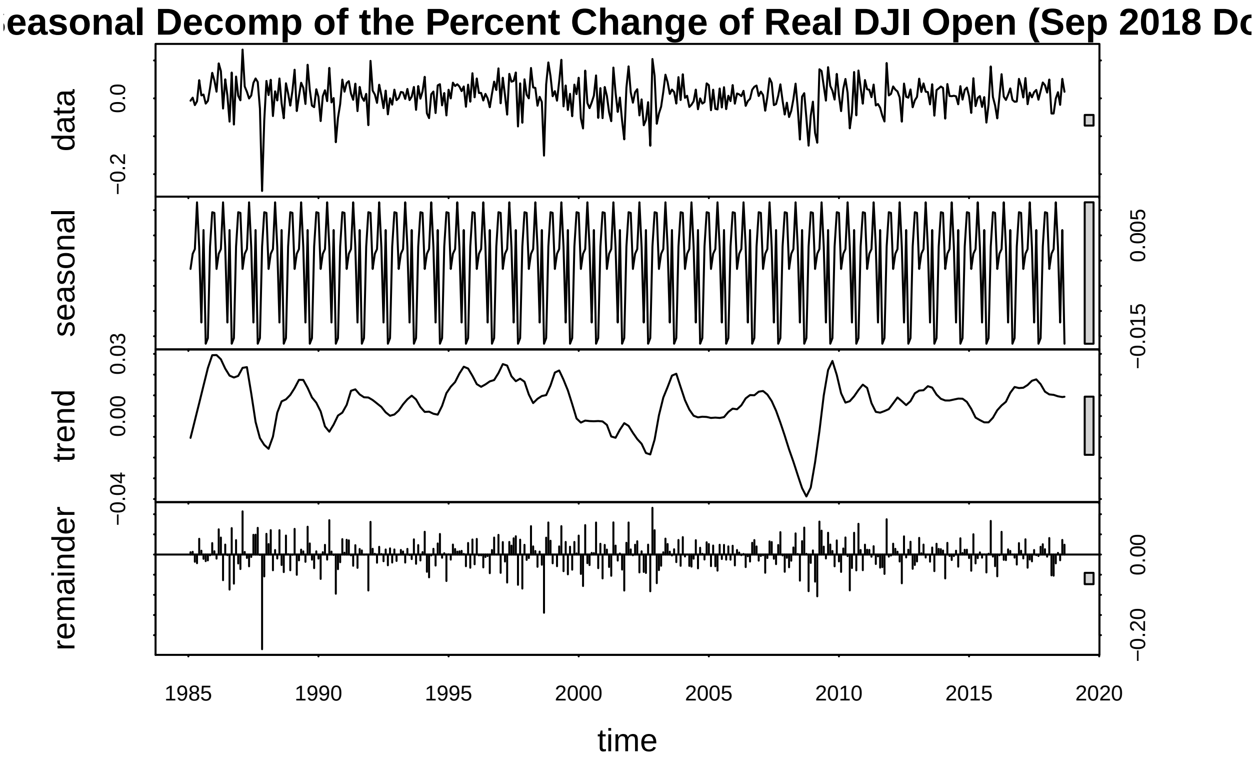
```
  ts(
    df$realDJIopen,
    start = c(1985, 1),
    end = c(2018, 9),
    frequency = 12
  )
percRealDJIOpen <-
  ts(
    df1$percChangeRealDJIopen,
    start = c(1985, 2),
    end = c(2018, 9),
    frequency = 12
  )
realSPOpen <-
  ts(
    df$realSPopen,
    start = c(1985, 1),
    end = c(2018, 9),
    frequency = 12
  )
percRealSPOpen <-
  ts(
    df1$percChangeRealSPopen,
    start = c(1985, 2),
    end = c(2018, 9),
    frequency = 12
  )
fedFund <-
  ts(
    df$fedFundRate,
    start = c(1985, 1),
    end = c(2018, 9),
    frequency = 12
  )

totalRetired <-
  ts(
    df$totalSSRetired,
    start = c(1985, 1),
    end = c(2018, 9),
    frequency = 12
  )


plot(stl(realDJIOpen, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of Real DJI Open (Sep 2018 Dollars)')
```



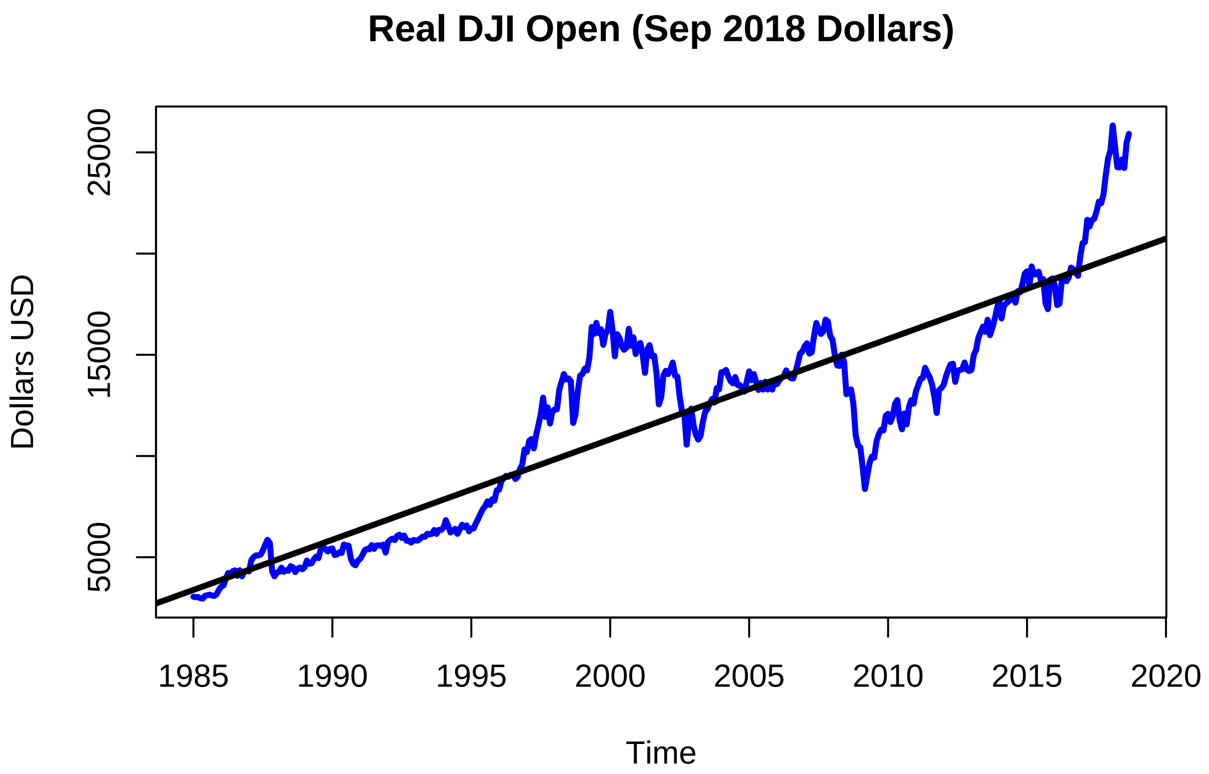
```r
plot(stl(log(realDJIOpen), s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of the Log of Real DJI Open (Sep 2018 Dollars)')
```



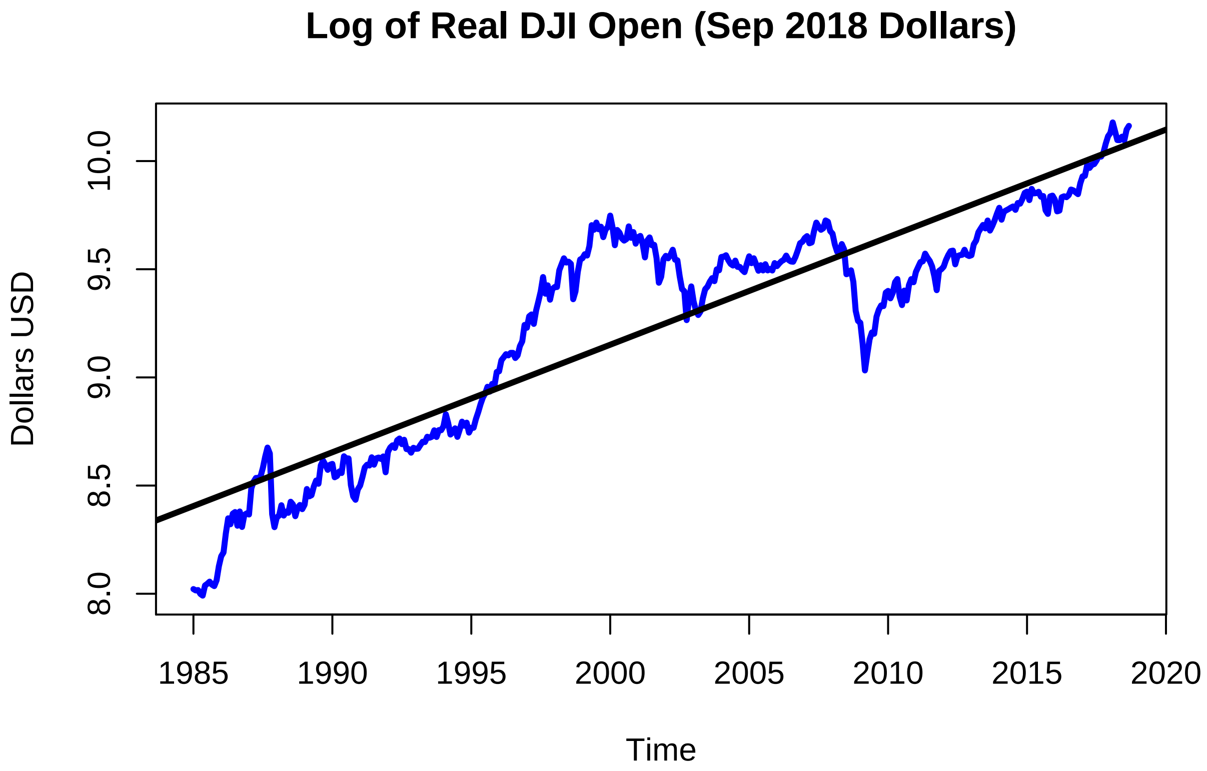
```r
plot(stl(percRealDJIOpen, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of the Percent Change of Real DJI Open (Sep 2018 Dollars)')
```



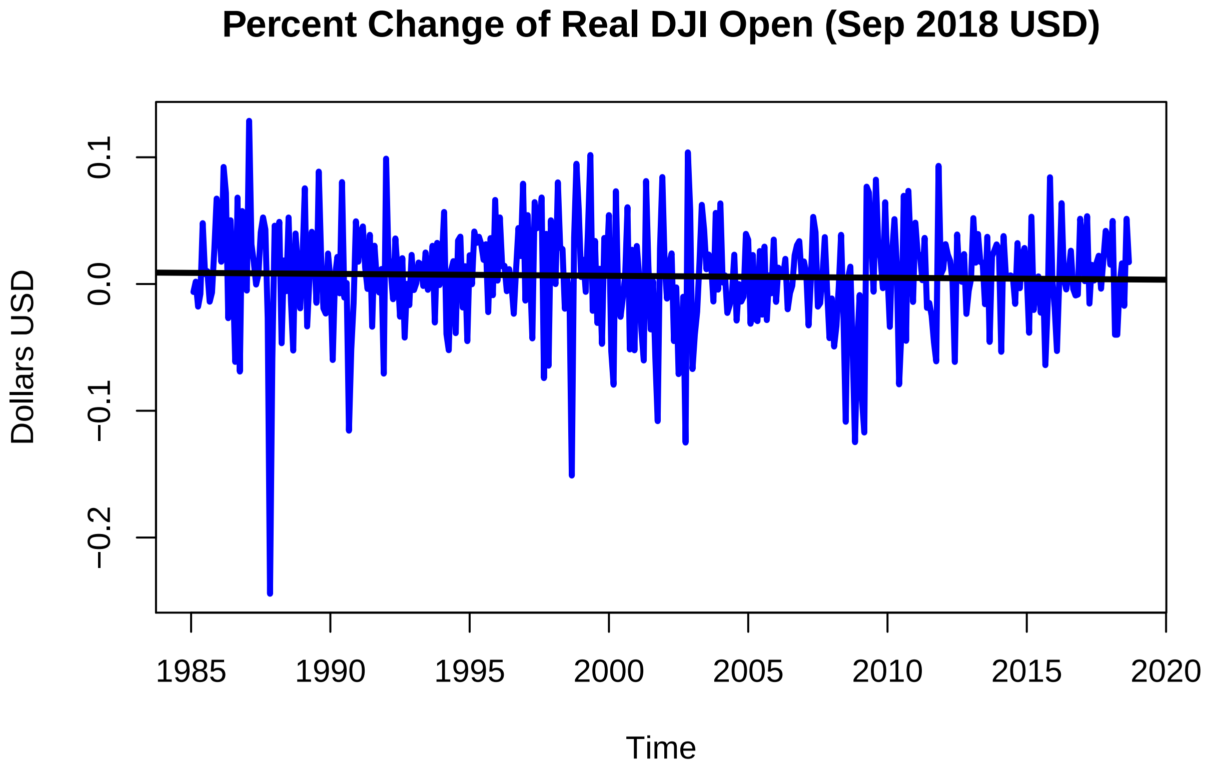
```r
plot(realDJIOpen,
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(realDJIOpen ~ time(realDJIOpen)), lwd = 3)
title(main = 'Real DJI Open (Sep 2018 Dollars)')
```



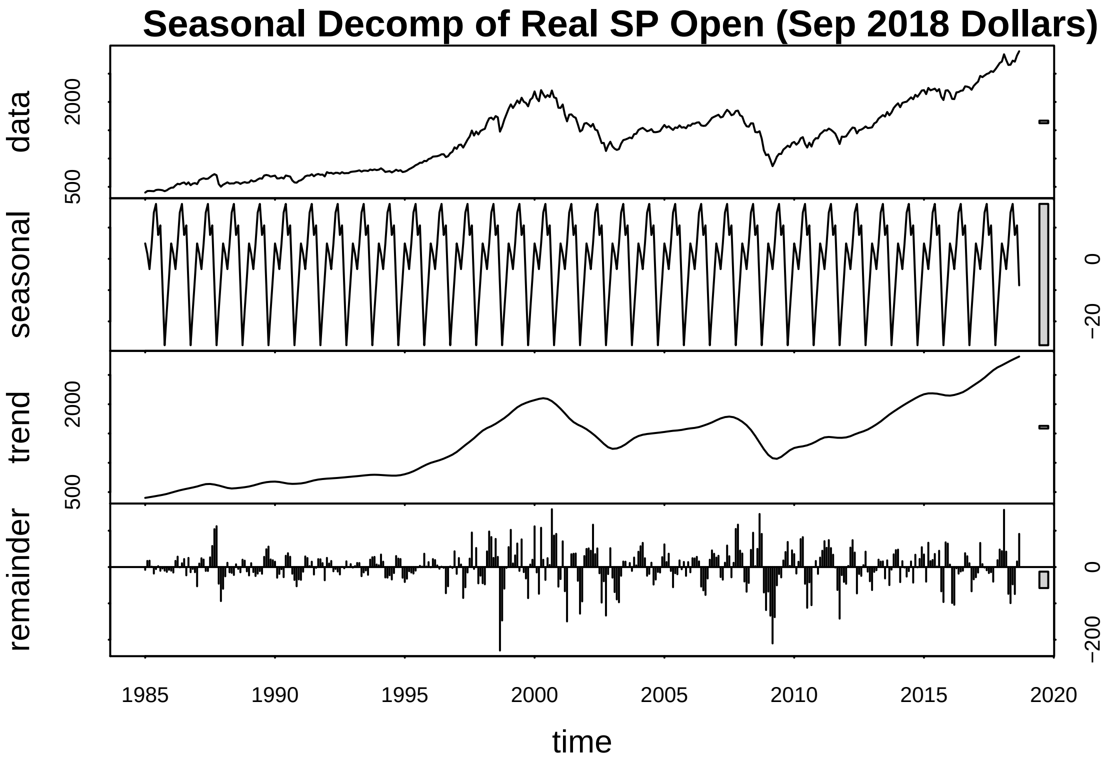
```r
plot(log(realDJIOpen),
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(log(realDJIOpen) ~ time(log(realDJIOpen))), lwd = 3)
title(main = 'Log of Real DJI Open (Sep 2018 Dollars)')
```



```
plot(percRealDJIOpen,
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(percRealDJIOpen ~ time(percRealDJIOpen)), lwd = 3)
title(main = 'Percent Change of Real DJI Open (Sep 2018 USD)')
```

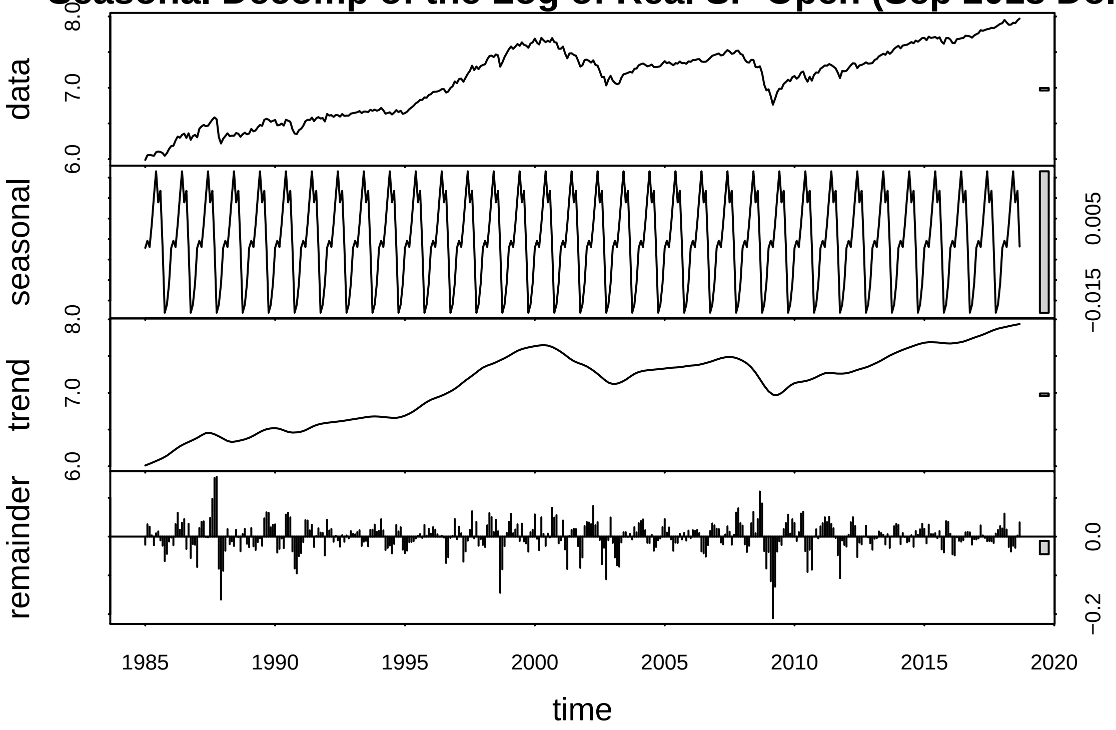


```
plot(stl(realSPOpen, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of Real SP Open (Sep 2018 Dollars)')
```



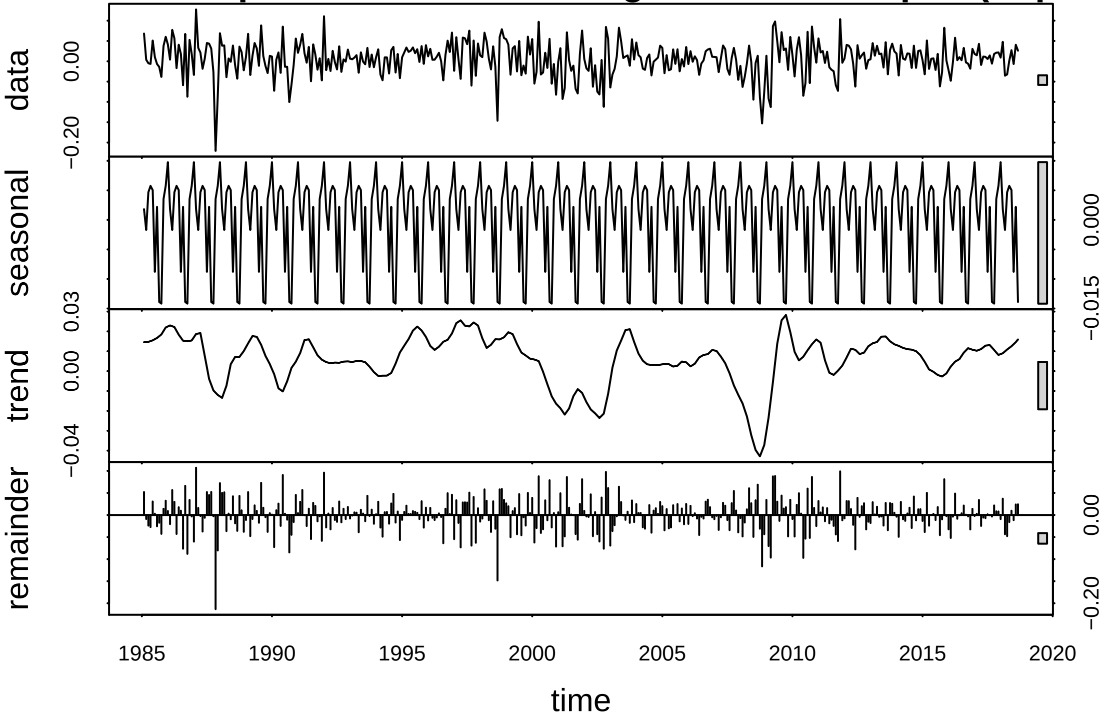
```
plot(stl(log(realSPOpen), s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of the Log of Real SP Open (Sep 2018 Dollars)')
```

Seasonal Decomp of the Log of Real SP Open (Sep 2018 Dollars)

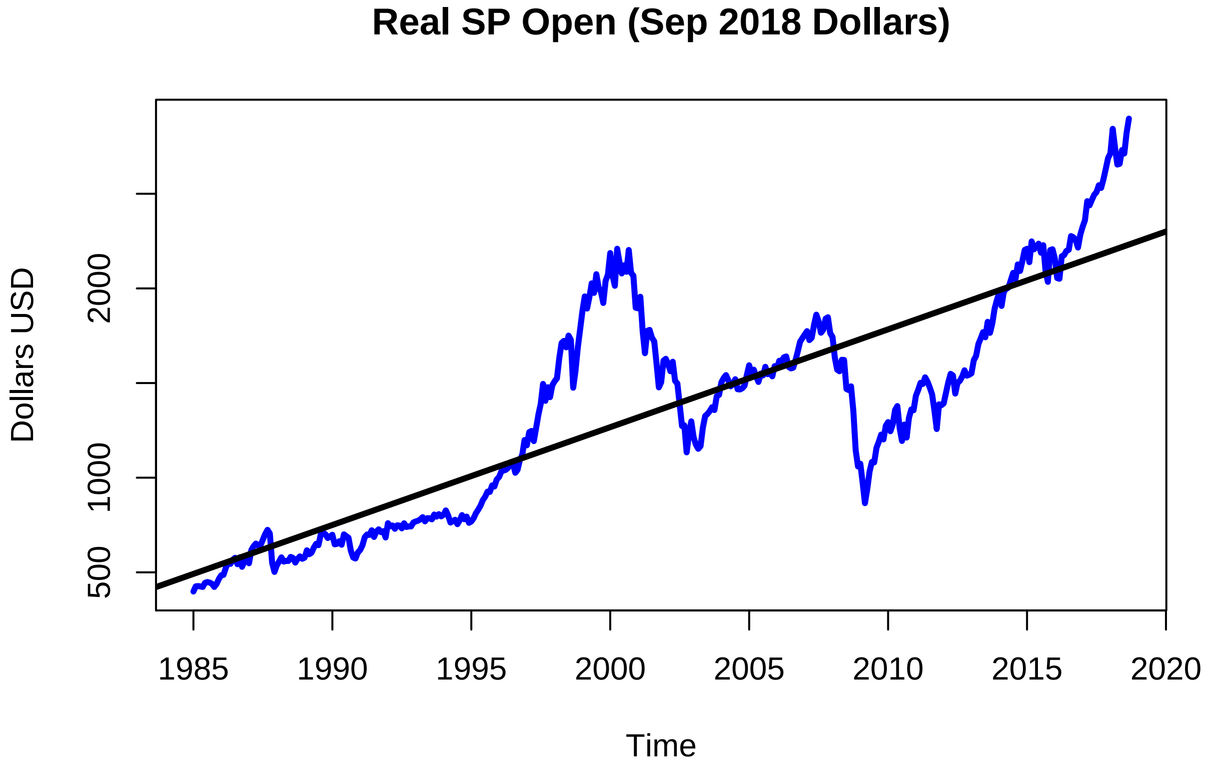


```
plot(stl(percRealSPOpen, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of the Percent Change of Real SP Open (Sep 2018 Dollars)')
```

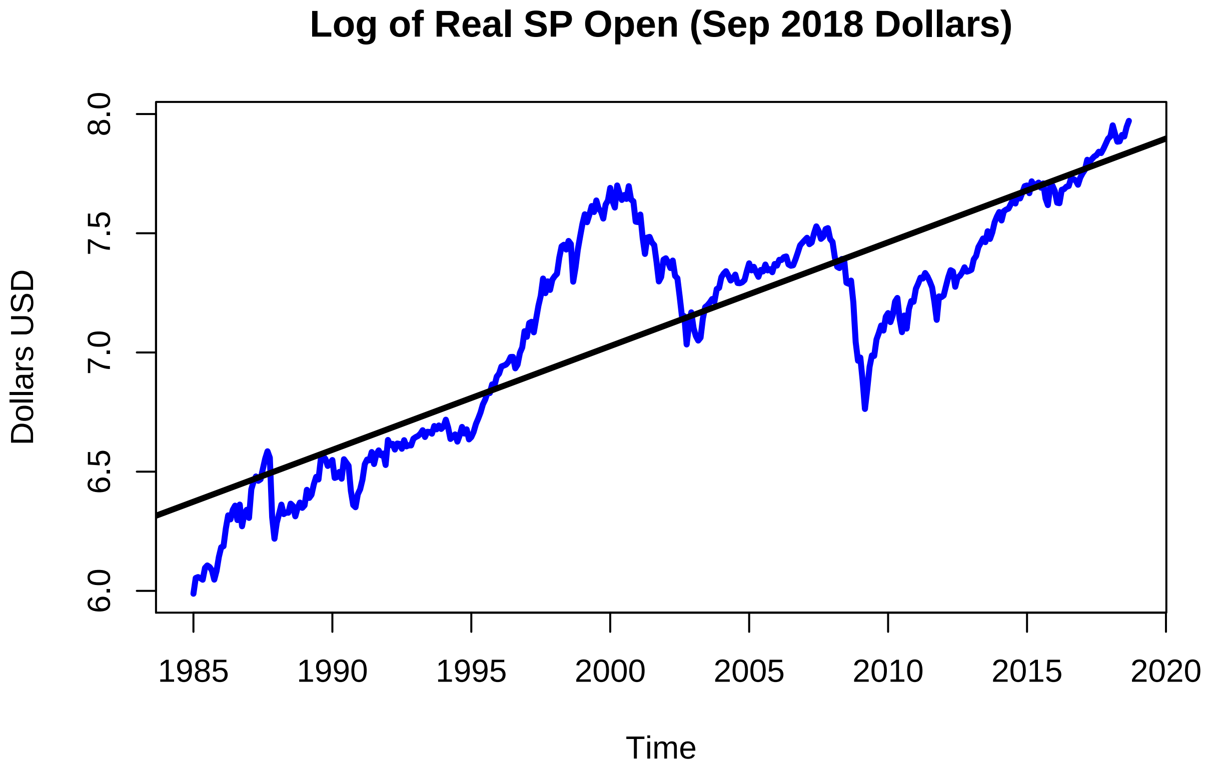
;easonal Decomp of the Percent Change of Real SP Open (Sep 2018 Dc



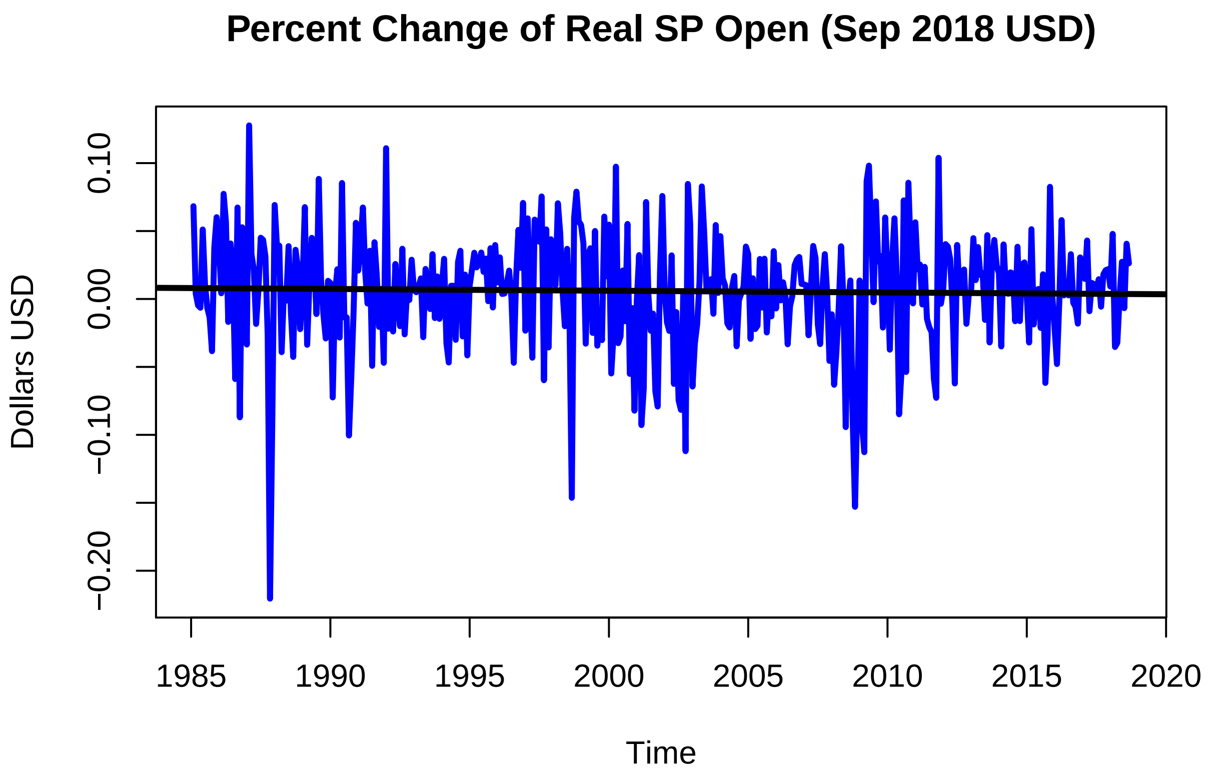
```
plot(realSPOpen,
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(realSPOpen ~ time(realSPOpen)), lwd = 3)
title(main = 'Real SP Open (Sep 2018 Dollars)')
```



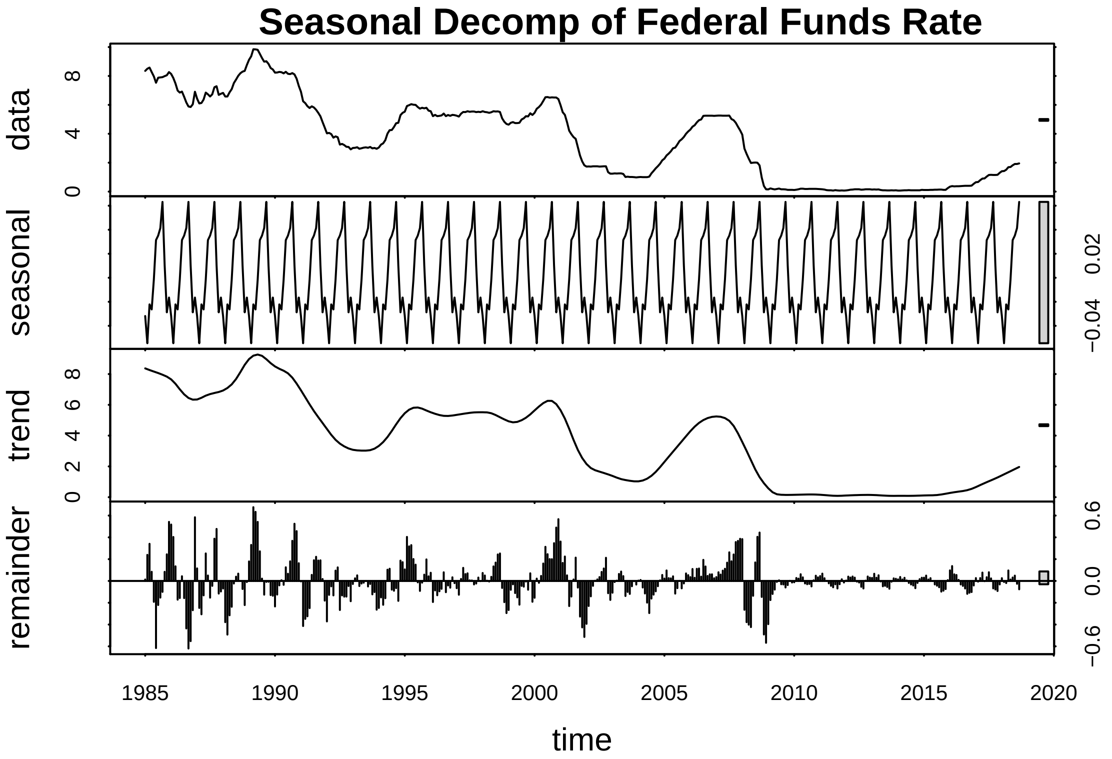
```
plot(log(realSPOpen),
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(log(realSPOpen) ~ time(log(realSPOpen))), lwd = 3)
title(main = 'Log of Real SP Open (Sep 2018 Dollars)')
```



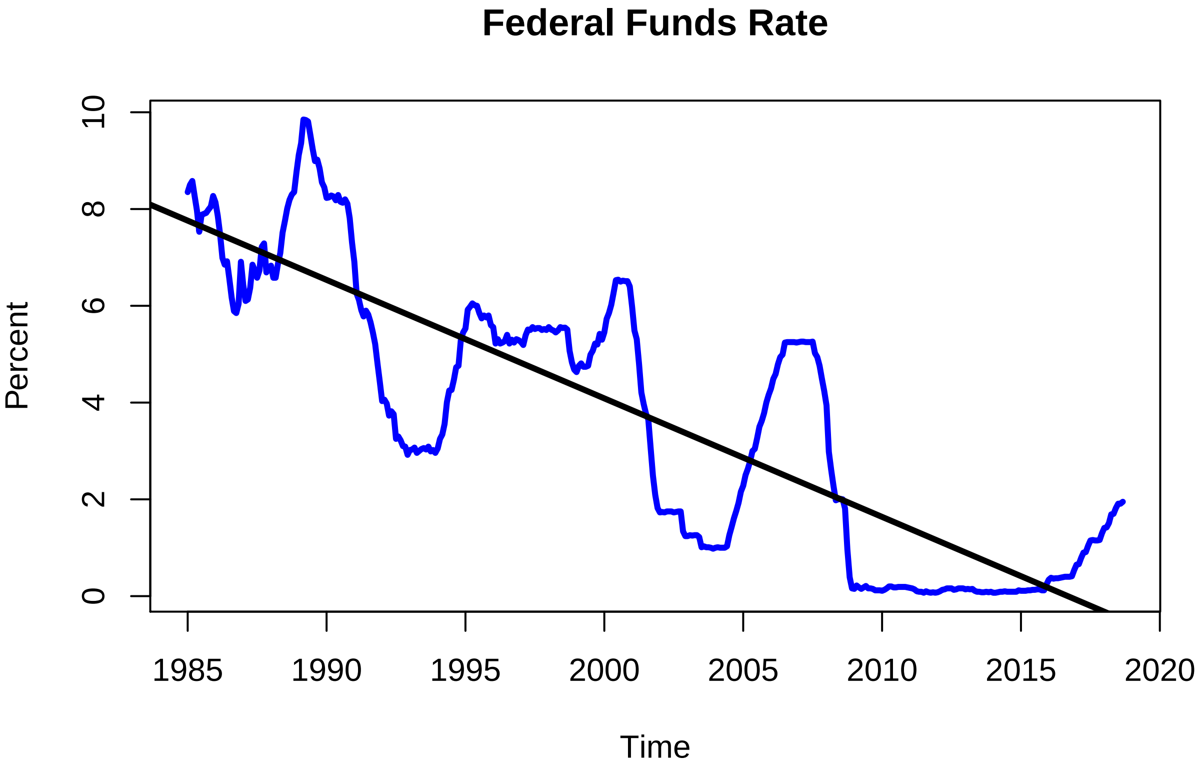
```
plot(percRealSPOpen,
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(percRealSPOpen ~ time(percRealSPOpen)), lwd = 3)
title(main = 'Percent Change of Real SP Open (Sep 2018 USD)')
```



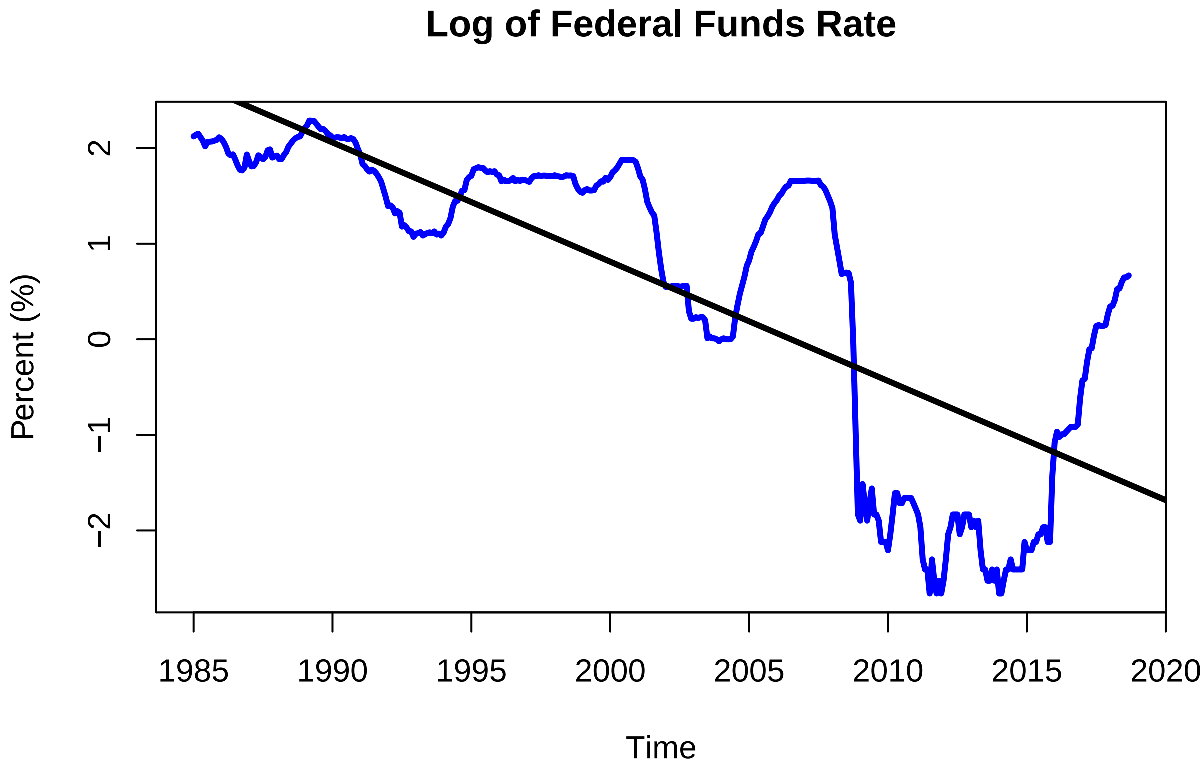
```
plot(stl(fedFund, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of Federal Funds Rate')
```



```
plot(fedFund,
     col = 'blue',
     lwd = 3,
     ylab = 'Percent')
abline(reg = lm(fedFund ~ time(fedFund)), lwd = 3)
title(main = 'Federal Funds Rate')
```

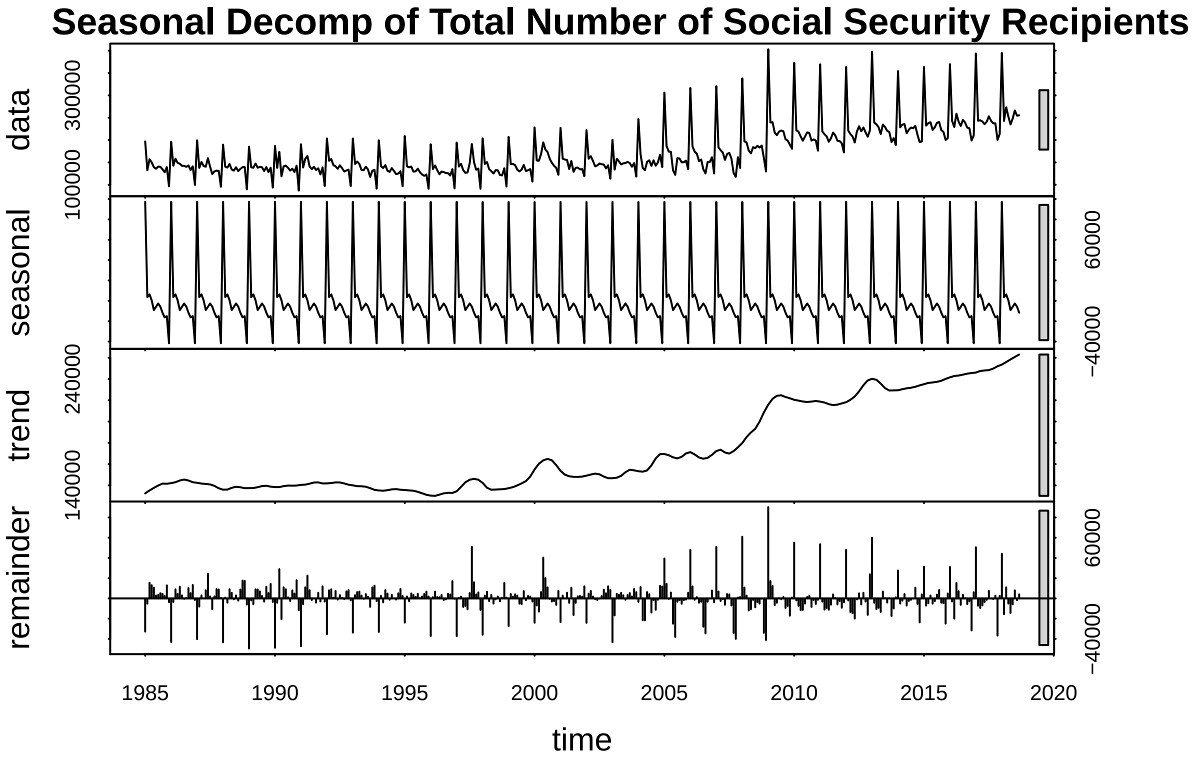


```
plot(log(fedFund),
     col = 'blue',
     lwd = 3,
     ylab = 'Percent (%)')
abline(reg = lm(log(fedFund) ~ time(log(fedFund))), lwd = 3)
title(main = 'Log of Federal Funds Rate')
```



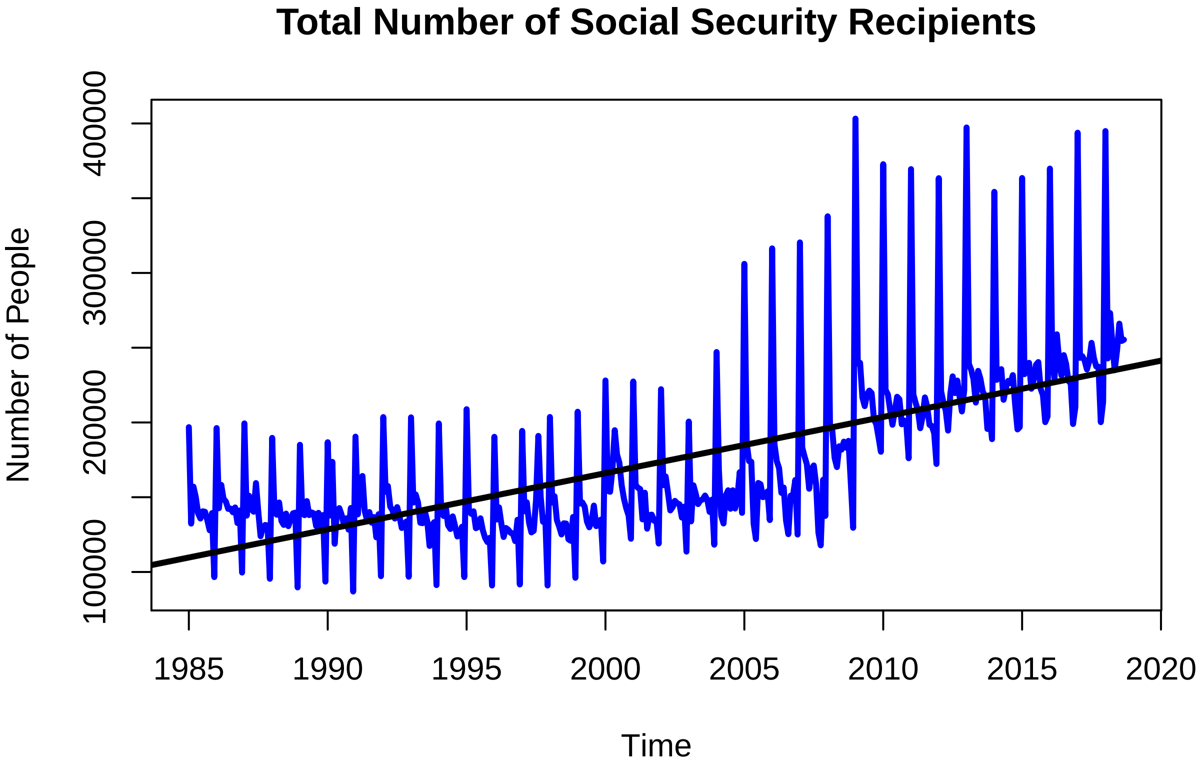
Log of Federal Funds Rate

```r
plot(stl(totalRetired, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of Total Number of Social Security Recipients')
```

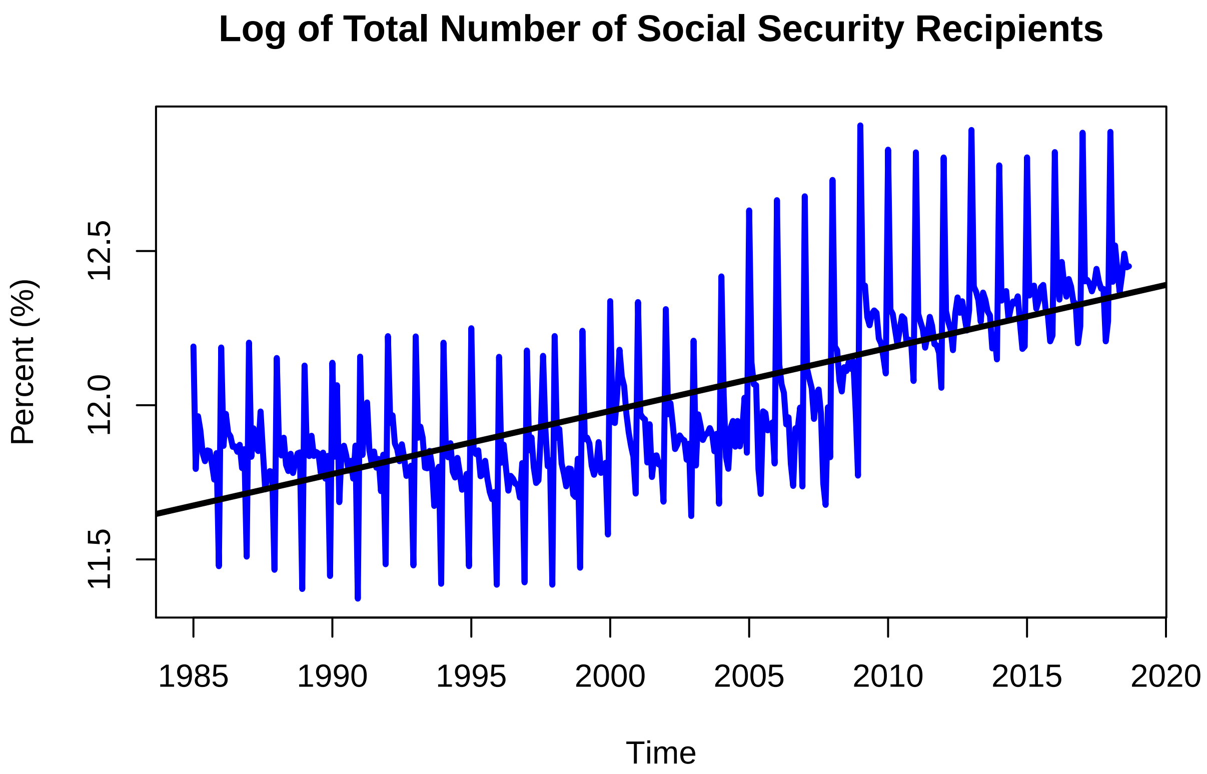


Seasonal Decomp of Total Number of Social Security Recipients

```r
plot(totalRetired,
     col = 'blue',
     lwd = 3,
     ylab = 'Number of People')
abline(reg = lm(totalRetired ~ time(totalRetired)), lwd = 3)
title(main = 'Total Number of Social Security Recipients')
```



```
plot(log(totalRetired),
     col = 'blue',
     lwd = 3,
     ylab = 'Percent (%)')
abline(reg = lm(log(totalRetired) ~ time(log(totalRetired))), lwd = 3)
title(main = 'Log of Total Number of Social Security Recipients')
```

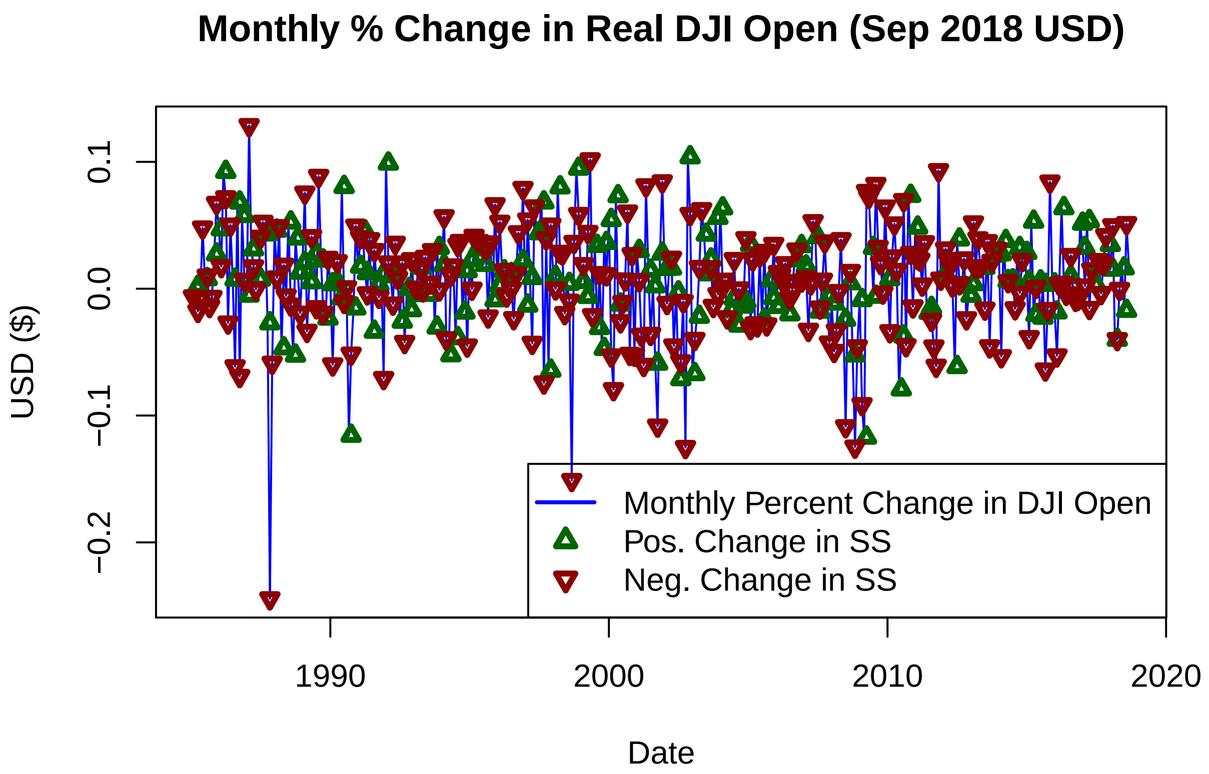


```r
# Remove nominal values aside indicators of positive change
df2 <- df1[, c(1, 8:10, 11:18, 32:44, 58:96, 110:122)]
# remove components of the total SS Retirees (male + female = total) and percent increases and decrease
df3 <- df2[, c(1:2, 8:9, 13:77)]

# Hypothesis Tests ####
# Stationarity Loop Testing
statVars <- matrix(data = NA, nrow = 68, ncol = 2)
df3TS <- ts(
  df3,
  start = c(1985, 12),
  end = c(2018, 9),
  frequency = 12
)
for (i in c(1:68)) {
  statVars[i,1] <- i+1
  statVars[i,2] <- adf.test(df3TS[,i+1], alternative = 'stationary')[[4]]
}
```

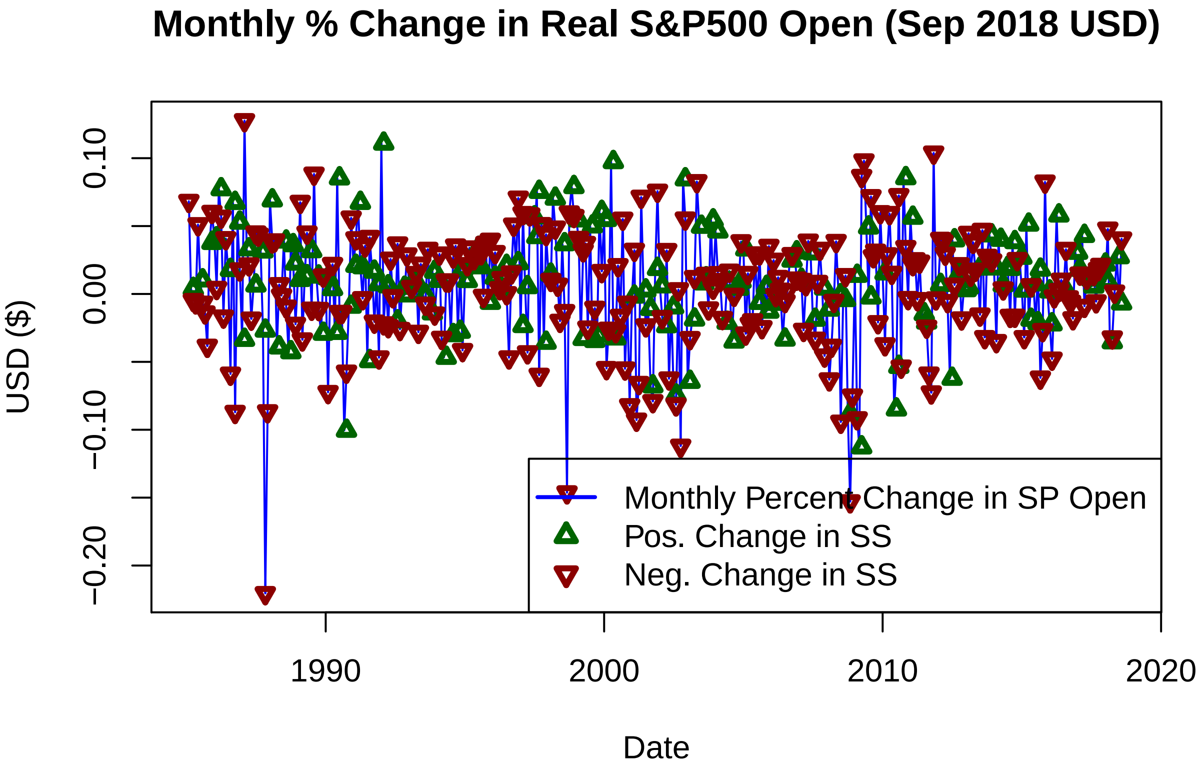
```
## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value
## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value
## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value
## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value

## Warning in adf.test(df3TS[, i + 1], alternative = "stationary"): p-value
## smaller than printed p-value
```

```r
# Reject the null when p < 0.05. So, the variables associated with this are
# likely stationary and useful for prediction of time series
dfStatSelect<- statVars[,1][statVars[,2] < 0.05]
dfStationary<- df3[,c(1,dfStatSelect)]
```

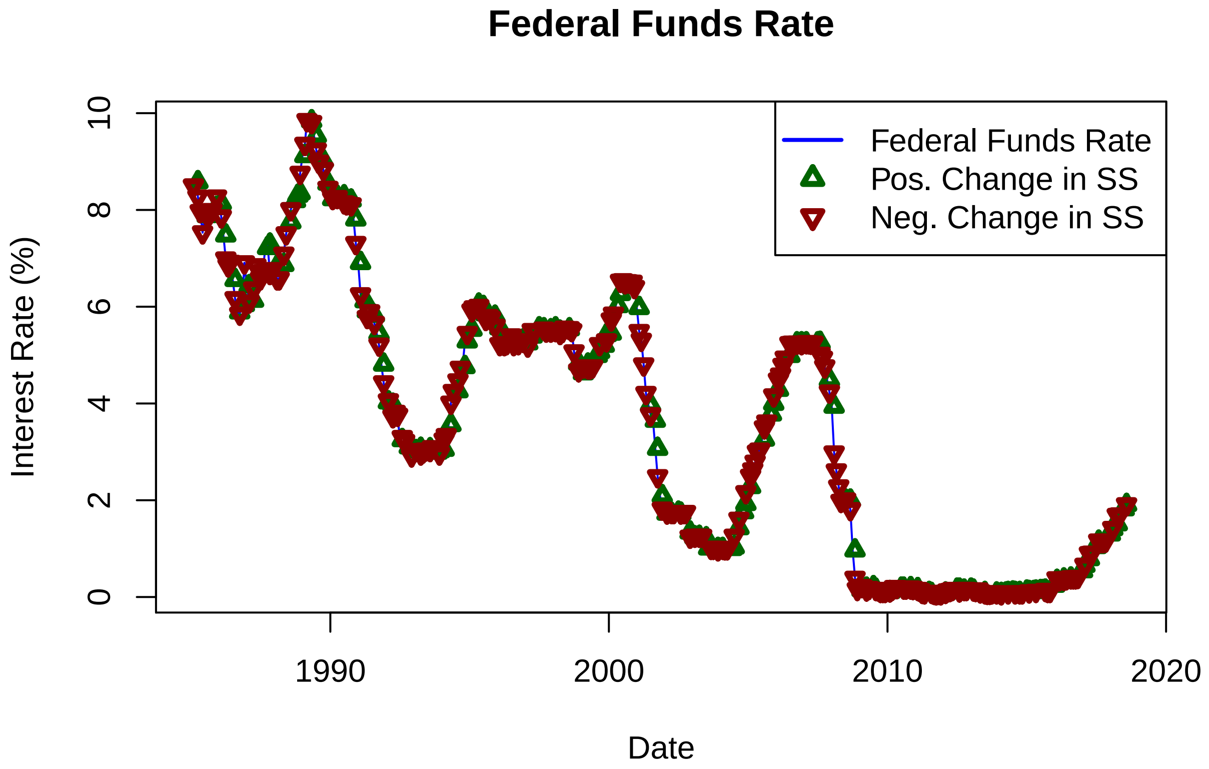
```r
# Visualizations ####
plot(
  x = dfStationary$date,
  y = dfStationary$percChangeRealDJIopen,
  col = 'blue',
  lwd = 1,
  type = 'l',
  ylab = 'USD ($)',
  xlab = 'Date'
)
points(
  x = dfStationary$date[df$postotalSSRetired == 1],
  y = dfStationary$percChangeRealDJIopen[dfStationary$postotalSSRetired == 1],
  pch = 24,
  col = 'darkgreen',
  cex = 0.8,
  lwd = 3
)
points(
  x = dfStationary$date[dfStationary$postotalSSRetired == 0],
  y = dfStationary$percChangeRealDJIopen[dfStationary$postotalSSRetired == 0],
  pch = 25,
  col = 'darkred',
  cex = 0.8,
  lwd = 3
)
legend(
  'bottomright',
  legend = c(
    'Monthly Percent Change in DJI Open',
    c('Pos. Change in SS', 'Neg. Change in SS')
  ),
  lty = c(1, c(NA, NA)),
  pch = c(NA, c(24, 25)),
  col = c('blue', c('darkgreen', 'darkred')),
  bg = c(NA, c('darkgreen', 'darkred')),
  lwd = c(2, c(3, 3))
)
title(main = 'Monthly % Change in Real DJI Open (Sep 2018 USD)')
```



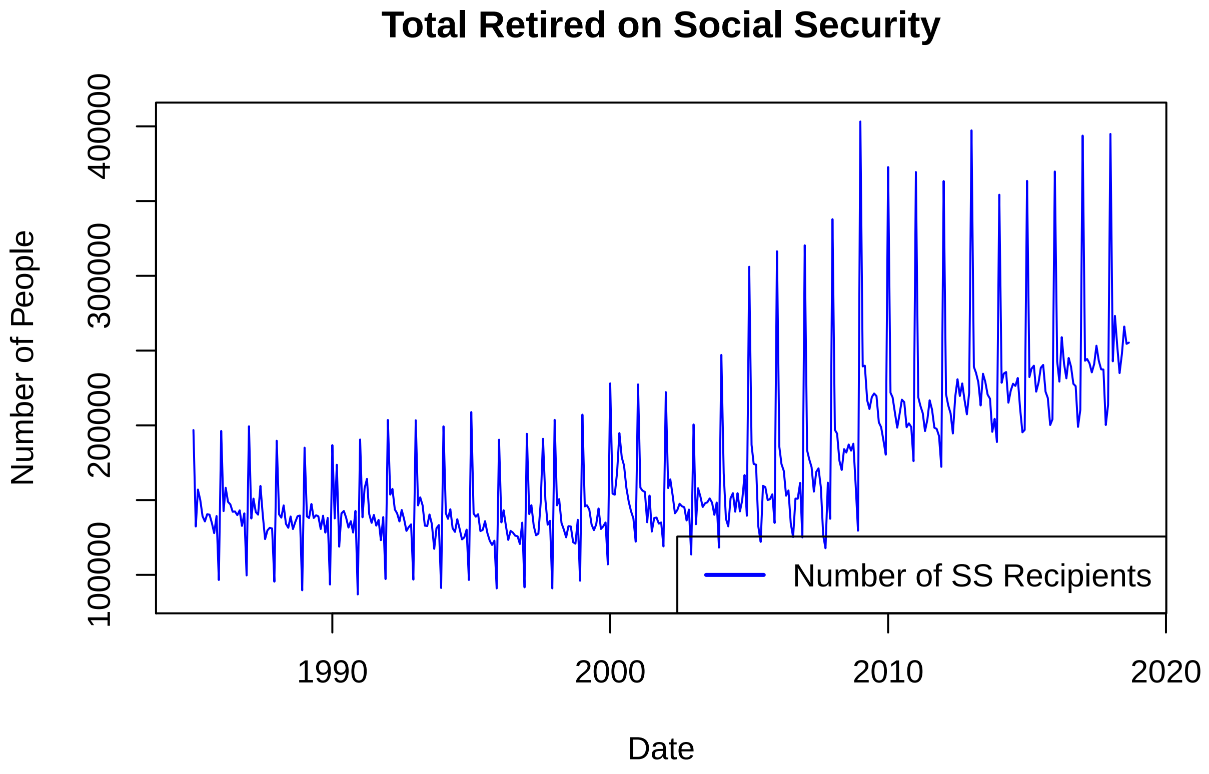
```
plot(
  x = dfStationary$date,
  y = dfStationary$percChangeRealSPopen,
  col = 'blue',
  lwd = 1,
  type = 'l',
  ylab = 'USD ($)',
  xlab = 'Date'
)
points(
  x = dfStationary$date[df$postotalSSRetired == 1],
  y = dfStationary$percChangeRealSPopen[dfStationary$postotalSSRetired == 1],
  pch = 24,
  col = 'darkgreen',
  cex = 0.8,
  lwd = 3
)
points(
  x = dfStationary$date[dfStationary$postotalSSRetired == 0],
  y = dfStationary$percChangeRealSPopen[dfStationary$postotalSSRetired == 0],
  pch = 25,
  col = 'darkred',
  cex = 0.8,
  lwd = 3
)
legend(
  'bottomright',
  legend = c(
    'Monthly Percent Change in SP Open',
    c('Pos. Change in SS', 'Neg. Change in SS')
  ),
  lty = c(1, c(NA, NA)),
  pch = c(NA, c(24, 25)),
  col = c('blue', c('darkgreen', 'darkred')),
  bg = c(NA, c('darkgreen', 'darkred')),
  lwd = c(2, c(3, 3))
)
title(main = 'Monthly % Change in Real S&P500 Open (Sep 2018 USD)')
```



```
plot(
  x = dfStationary$date,
  y = dfStationary$fedFundRate,
  col = 'blue',
  lwd = 1,
  type = 'l',
  ylab = 'Interest Rate (%)',
  xlab = 'Date'
)
points(
  x = dfStationary$date[df$postotalSSRetired == 1],
  y = dfStationary$fedFundRate[dfStationary$postotalSSRetired == 1],
  pch = 24,
  col = 'darkgreen',
  cex = 0.8,
  lwd = 3
)
points(
  x = dfStationary$date[dfStationary$postotalSSRetired == 0],
  y = dfStationary$fedFundRate[dfStationary$postotalSSRetired == 0],
  pch = 25,
  col = 'darkred',
  cex = 0.8,
  lwd = 3
)
legend(
  'topright',
  legend = c('Federal Funds Rate',
             c('Pos. Change in SS', 'Neg. Change in SS')),
  lty = c(1, c(NA, NA)),
  pch = c(NA, c(24, 25)),
  col = c('blue', c('darkgreen', 'darkred')),
  bg = c(NA, c('darkgreen', 'darkred')),
  lwd = c(2, c(3, 3))
)
title(main = 'Federal Funds Rate')
```



```
plot(
  x = df$date,
  y = df$totalSSRetired,
  col = 'blue',
  lwd = 1,
  type = 'l',
  ylab = 'Number of People',
  xlab = 'Date'
)
legend(
  'bottomright',
  legend = c('Number of SS Recipients'),
  lty = c(1),
  col = c('blue'),
  lwd = c(2)
)
title(main = 'Total Retired on Social Security')
```



```
# Selection ####
# Set a few dataframes for different variables
dfDiff <- dfStationary[,c(2:5,7:19)]
dfPosChange <- dfStationary[,c(2:5, 20:45)]
dfPerc <- dfStationary[,c(2:5, 46:58)]
# Run the selections
# Differences ####
regFitSelect <- regsubsets(
  postotalSSRetired~.,
  data=dfDiff,
  nvmax=17)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 2 linear dependencies found
```

```
## Reordering variables and trying again:
```

```
regSummary <- summary(regFitSelect)
names(regSummary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
regSummary$rsq
```

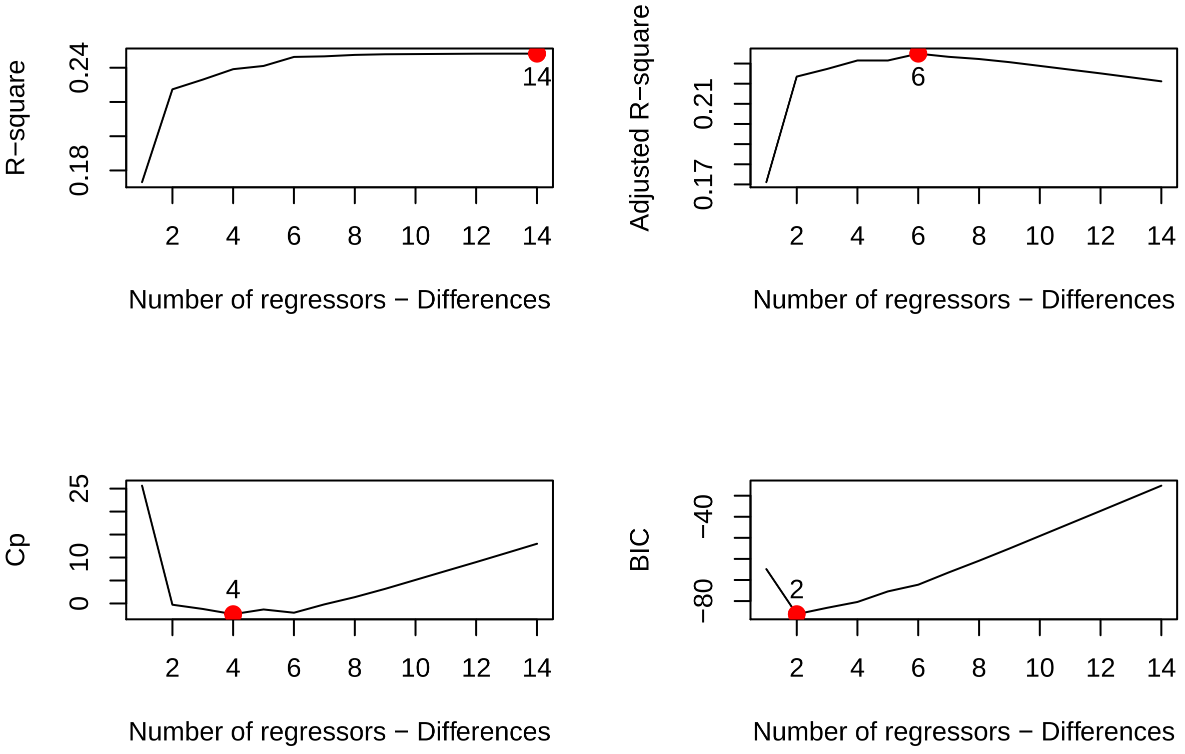
```
##  [1] 0.1731698 0.2273926 0.2330672 0.2391592 0.2410517 0.2463118 0.2466627
##  [8] 0.2474956 0.2478743 0.2479814 0.2480829 0.2481942 0.2482266 0.2482272
```

```r
regSummary$adjr2
```

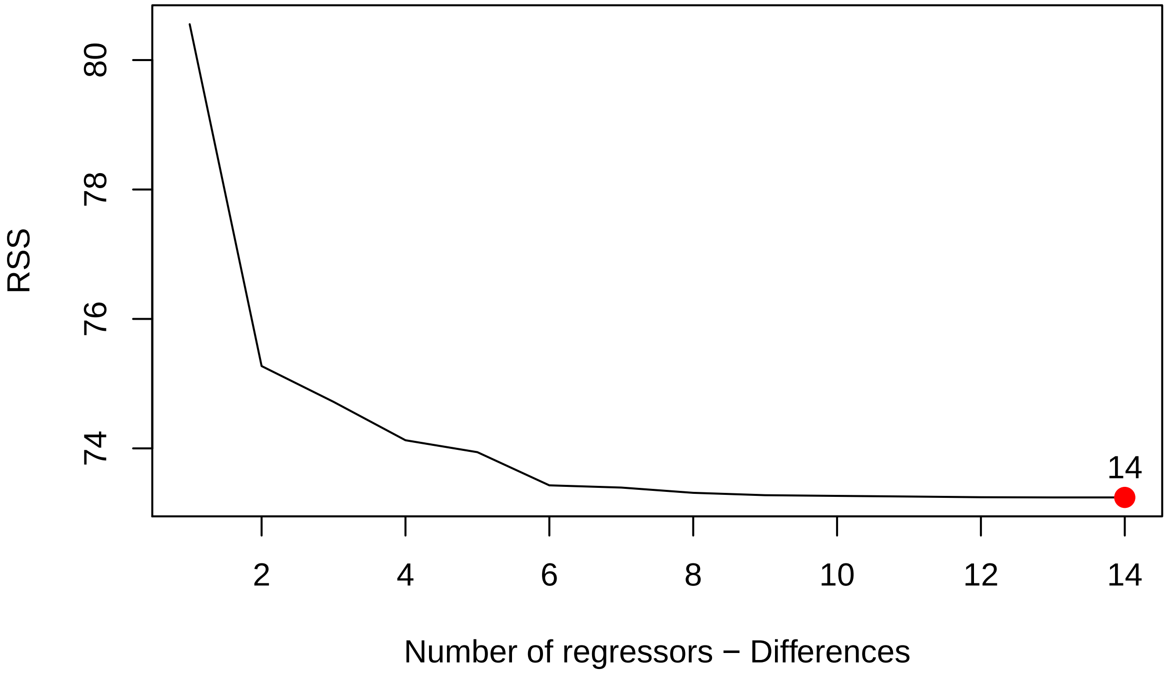
```
##  [1] 0.1711130 0.2235392 0.2273153 0.2315317 0.2315172 0.2349210 0.2333461
##  [8] 0.2322550 0.2306938 0.2288461 0.2269832 0.2251209 0.2231675 0.2211711
```

```r
par(mfrow=c(2,2))
aRSQ <- which.max(regSummary$rsq)
aARSQ <- which.max(regSummary$adjr2)
aCP <- which.min(regSummary$cp)
aBIC <- which.min(regSummary$bic)
aRSS <- which.min(regSummary$rss)

par(mfrow = c(2, 2))

plot(
  regSummary$rsq,
  xlab = "Number of regressors - Differences",
  ylab = "R-square",
  type = "l"
)
points(
  aRSQ,
  regSummary$rsq[aRSQ],
  col = "red",
  cex = 2,
  pch = 20
)
text(aRSQ,
     regSummary$rsq[aRSQ],
     labels = aRSQ,
     pos = 1)

plot(
  regSummary$adjr2,
  xlab = "Number of regressors - Differences",
  ylab = "Adjusted R-square",
  type = "l"
)
points(
  aARSQ,
  regSummary$adjr2[aARSQ],
  col = "red",
  cex = 2,
  pch = 20
)
text(aARSQ,
     regSummary$adjr2[aARSQ],
     labels = aARSQ,
     pos = 1)

plot(regSummary$cp,
     xlab = "Number of regressors - Differences",
     ylab = "Cp",
     type = "l")
```

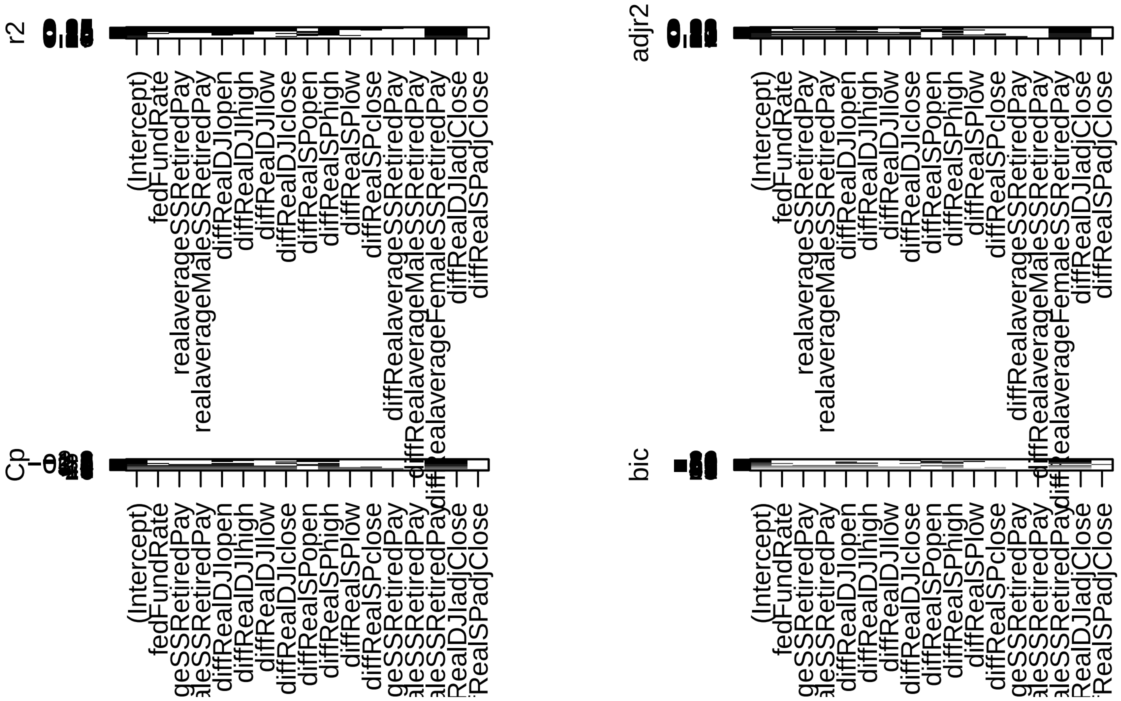
```
points(
  aCP,
  regSummary$cp[aCP],
  col = "red",
  cex = 2,
  pch = 20
)
text(aCP,
     regSummary$cp[aCP],
     labels = aCP,
     pos = 3)

plot(
  regSummary$bic,
  xlab = "Number of regressors - Differences",
  ylab = "BIC",
  type = "l"
)
points(
  aBIC,
  regSummary$bic[aBIC],
  col = "red",
  cex = 2,
  pch = 20
)
text(aBIC,
     regSummary$bic[aBIC],
     labels = aBIC,
     pos = 3)
```



```r
par(mfrow = c(1, 1))
plot(
  regSummary$rss,
  xlab = "Number of regressors - Differences",
  ylab = "RSS",
  type = "l"
)
points(
  aRSS,
  regSummary$rss[aRSS],
  col = "red",
  cex = 2,
  pch = 20
)
text(aRSS,
     regSummary$rss[aRSS],
     labels = aRSS,
     pos = 3)
```



```r
par(mfrow = c(2, 2))
plot(regFitSelect, scale = "r2")
plot(regFitSelect, scale = "adjr2")
plot(regFitSelect, scale = "Cp")
plot(regFitSelect, scale = "bic")
```



```r
# Percentages ####
regFitSelect <- regsubsets(
  postotalSSRetired~.,
  data=dfPerc,
  nvmax=17)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 2 linear dependencies found

## Reordering variables and trying again:
```

```r
regSummary <- summary(regFitSelect)
names(regSummary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```r
regSummary$rsq
```

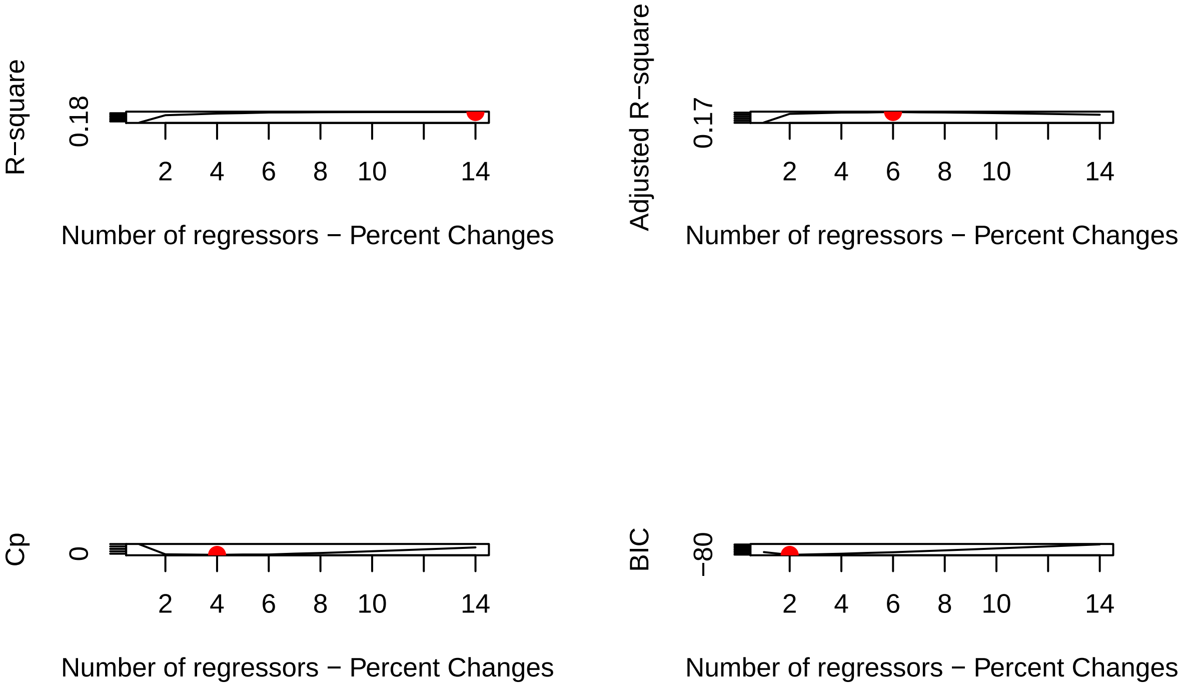
```
##  [1] 0.1734833 0.2178484 0.2228415 0.2279842 0.2305041 0.2344034 0.2348847
##  [8] 0.2361503 0.2366112 0.2368900 0.2370863 0.2371085 0.2371529 0.2371530
```

```r
regSummary$adjr2
```

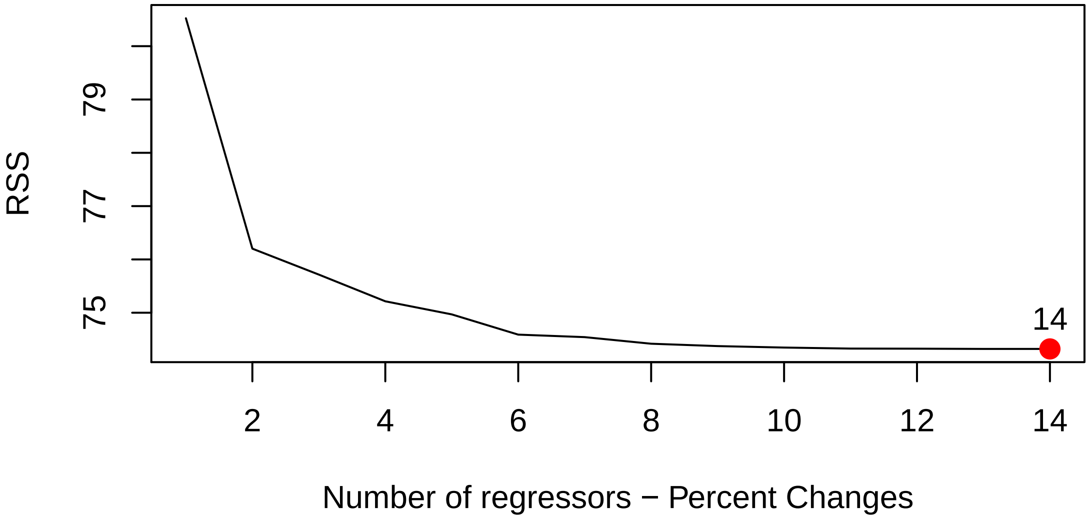
```
##  [1] 0.1714273 0.2139474 0.2170129 0.2202447 0.2208371 0.2228327 0.2213599
##  [8] 0.2206799 0.2191734 0.2174725 0.2156780 0.2136949 0.2117246 0.2096984
```

```r
par(mfrow=c(2,2))
aRSQ <- which.max(regSummary$rsq)
aARSQ <- which.max(regSummary$adjr2)
aCP <- which.min(regSummary$cp)
aBIC <- which.min(regSummary$bic)
aRSS <- which.min(regSummary$rss)

par(mfrow = c(2, 2))
```

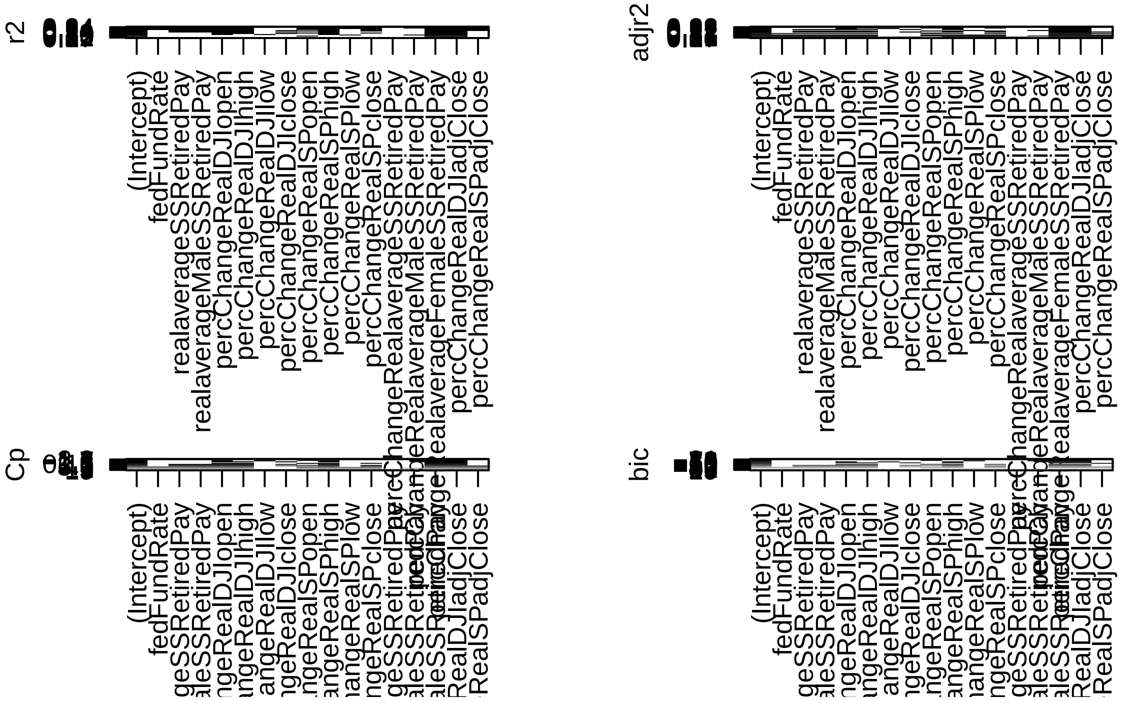
```r
plot(
  regSummary$rsq,
  xlab = "Number of regressors - Percent Changes",
  ylab = "R-square",
  type = "l"
)
points(
  aRSQ,
  regSummary$rsq[aRSQ],
  col = "red",
  cex = 2,
  pch = 20
)
text(aRSQ,
     regSummary$rsq[aRSQ],
     labels = aRSQ,
     pos = 1)

plot(
  regSummary$adjr2,
  xlab = "Number of regressors - Percent Changes",
  ylab = "Adjusted R-square",
  type = "l"
)
points(
  aARSQ,
  regSummary$adjr2[aARSQ],
  col = "red",
  cex = 2,
  pch = 20
)
text(aARSQ,
     regSummary$adjr2[aARSQ],
     labels = aARSQ,
     pos = 1)

plot(regSummary$cp,
     xlab = "Number of regressors - Percent Changes",
     ylab = "Cp",
     type = "l")
points(
  aCP,
  regSummary$cp[aCP],
  col = "red",
  cex = 2,
  pch = 20
)
text(aCP,
     regSummary$cp[aCP],
     labels = aCP,
     pos = 3)

plot(
  regSummary$bic,
  xlab = "Number of regressors - Percent Changes",
  ylab = "BIC",
  type = "l"
)
points(
  aBIC,
  regSummary$bic[aBIC],
  col = "red",
  cex = 2,
  pch = 20
)
text(aBIC,
     regSummary$bic[aBIC],
     labels = aBIC,
     pos = 3)
```



```
par(mfrow = c(1, 1))
plot(
  regSummary$rss,
  xlab = "Number of regressors - Percent Changes",
  ylab = "RSS",
  type = "l"
)
points(
  aRSS,
  regSummary$rss[aRSS],
  col = "red",
  cex = 2,
  pch = 20
)
text(aRSS,
     regSummary$rss[aRSS],
     labels = aRSS,
     pos = 3)
```



```
par(mfrow = c(2, 2))
plot(regFitSelect, scale = "r2")
plot(regFitSelect, scale = "adjr2")
plot(regFitSelect, scale = "Cp")
plot(regFitSelect, scale = "bic")
```



```
# Model ####
# Setting train/test split
set.seed(1)
trainSample <- sample(1:nrow(dfStationary), round(nrow(dfStationary)/2), replace = F)
trainData <- dfStationary[trainSample,]
```

```r
testData <- dfStationary[-trainSample,]

trainX <- trainData[,c(1, 3:58)]
trainY <- trainData[,c(1:2)]
testX <- testData[,c(1, 3:58)]
trainY <- testData[,c(1:2)]

# Logistic ####
glmFit <- glm(postotalSSRetired ~ diffRealDJIopen + diffRealDJIhigh + diffRealSPopen + diffRealSPhigh +
summary(glmFit)
```

```
##
## Call:
## glm(formula = postotalSSRetired ~ diffRealDJIopen + diffRealDJIhigh +
##     diffRealSPopen + diffRealSPhigh + diffRealaverageFemaleSSRetiredPay +
##     diffRealDJIclose, family = binomial, data = dfStationary)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4489  -0.9187  -0.5662   1.0736   2.0489
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -0.4797452  0.1170904  -4.097 4.18e-05
## diffRealDJIopen                  -0.0015955  0.0007932  -2.011   0.0443
## diffRealDJIhigh                   0.0009670  0.0008389   1.153   0.2490
## diffRealSPopen                    0.0115620  0.0067957   1.701   0.0889
## diffRealSPhigh                    0.0009465  0.0069646   0.136   0.8919
## diffRealaverageFemaleSSRetiredPay  0.0346821  0.0048299   7.181 6.93e-13
## diffRealDJIclose                 -0.0008797  0.0003696  -2.380   0.0173
##
## (Intercept)                      ***
## diffRealDJIopen                  *
## diffRealDJIhigh
## diffRealSPopen                   .
## diffRealSPhigh
## diffRealaverageFemaleSSRetiredPay ***
## diffRealDJIclose                 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 545.68  on 403  degrees of freedom
## Residual deviance: 451.92  on 397  degrees of freedom
## AIC: 465.92
##
## Number of Fisher Scoring iterations: 5
```

```r
glmProbs <- predict(glmFit, type = 'response')
glmProbs[1:10]
```

```
##         2         3         4         5         6         7         8
## 0.1273136 0.4627141 0.3036248 0.3521064 0.4062105 0.5305135 0.2119409
##         9        10        11
## 0.3454653 0.2727542 0.3951145
```

```
glmPred <- rep(0, dim(dfStationary)[2])
glmPred[glmProbs > 0.5] <- 1
table(glmPred)
```

```
## glmPred
##   0   1
##  46 110
```

```
table(glmPred, dfStationary[,2])
```

```
##
## glmPred  0  1
##       0 29 17
##       1 36 74
```

```
mean(glmPred == dfStationary[,2])
```

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

```
# Testing Prediction
train <- subset(dfStationary, dfStationary$date < as.Date('2010-04-08'))
test3rdQuart <- subset(dfStationary, dfStationary$date >= as.Date('2010-04-08'))

glmFit <- glm(postotalSSRetired ~ diffRealDJIopen + diffRealDJIhigh + diffRealSPopen + diffRealSPhigh +
glmProbs <- predict(glmFit, test3rdQuart, type = 'response') # setting prediction for the testing set F

glmPred <- rep(0, 101)
glmPred[glmProbs > 0.5] = 1

table(glmPred, test3rdQuart$postotalSSRetired)
```

```
##
## glmPred  0  1
##       0 52 19
##       1  7 23
```

```
mean(glmPred == test3rdQuart$postotalSSRetired)
```

```
## [1] 0.7425743
```

Predicition accuracy is approximately 74% with a train/test split at the 3rd quartile mark of the dates. Seems good

```
# Basic ARIMA
```