# DaiglePredictionofSocialSecurity.R

*daiglechris*

*Wed Dec 5 11:19:08 2018*

---

Chris Daigle Prediction of Social Security Awards

```
# 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.
```

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

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

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

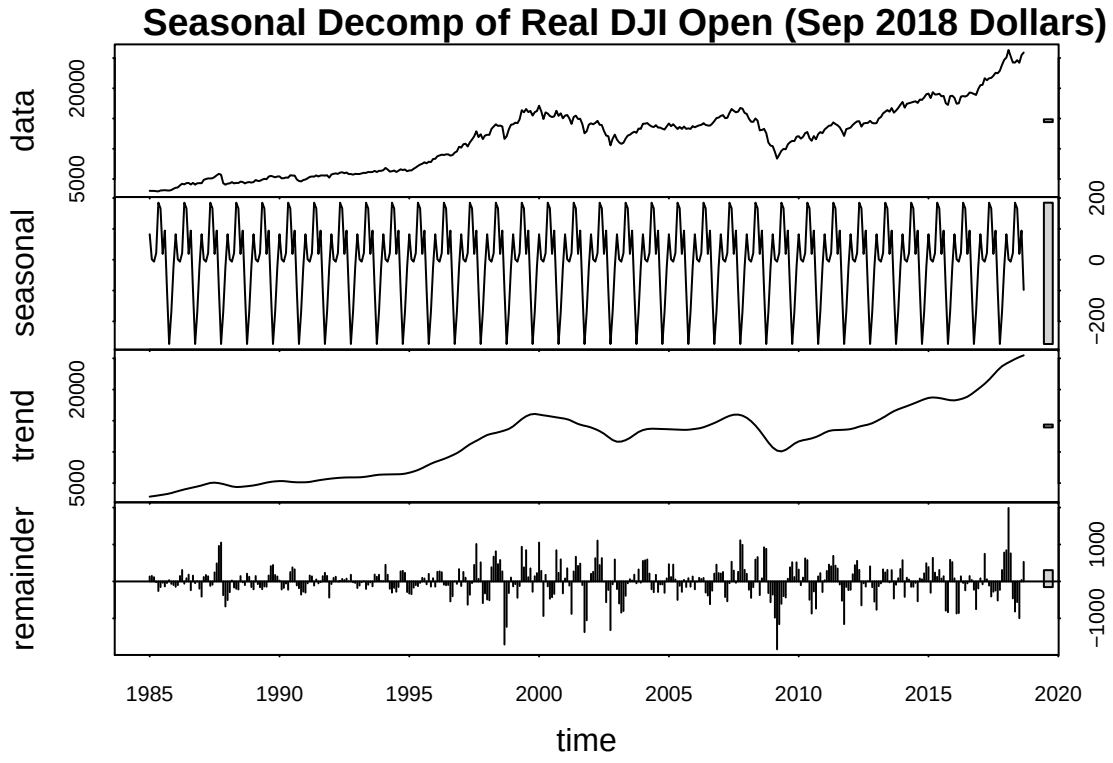
```
library(leaps)
setwd('~/Git/MachineLearningAndBigDataWithR/Data')
dataName <- 'assembled.csv'
df <- read.csv(dataName, stringsAsFactors = FALSE)
# Summarize and clean data ####
# head(df)
df <- df[-1]
# head(df)
# str(df)

# 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
```

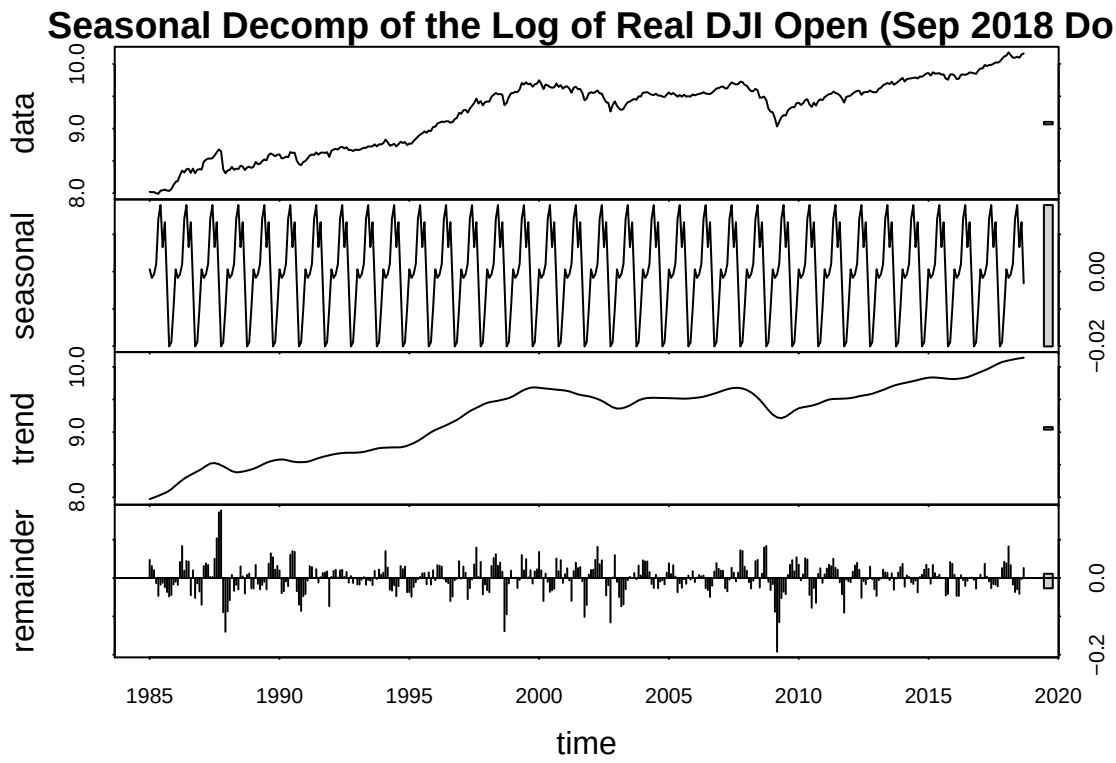
```
# 8 SPvolume                  : num
# 9 DJIopen                   : num
# 10 DJIhigh                   : num
# 11 DJIlow                    : num
# 12 DJIclose                  : num
# 13 DJIadjClose               : num
# 14 SPopen                    : num
# 15 SPhigh                    : num
# 16 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 #
```

```
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 <-
```

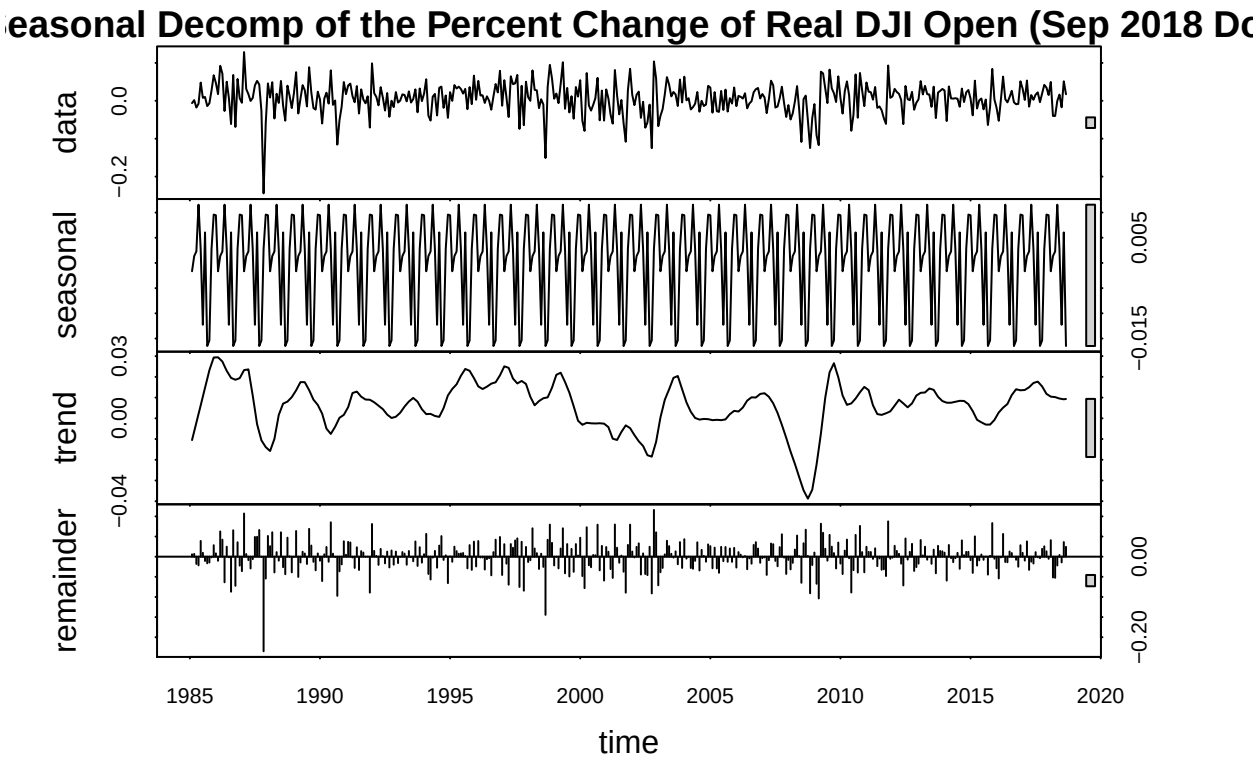
```
  ts(
    df$realDJIopen,
    start = c(1985, 1),
    end = c(2018, 9),
    frequency = 12
  )
percRealDJIOpen <-
  ts(
    df1$percChangeRealDJIopen,
    start = c(1985, 2),
    end = c(2018, 9),
    frequency = 12
  )
realSPOpen <-
  ts(
    df$realSPopen,
    start = c(1985, 1),
    end = c(2018,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


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



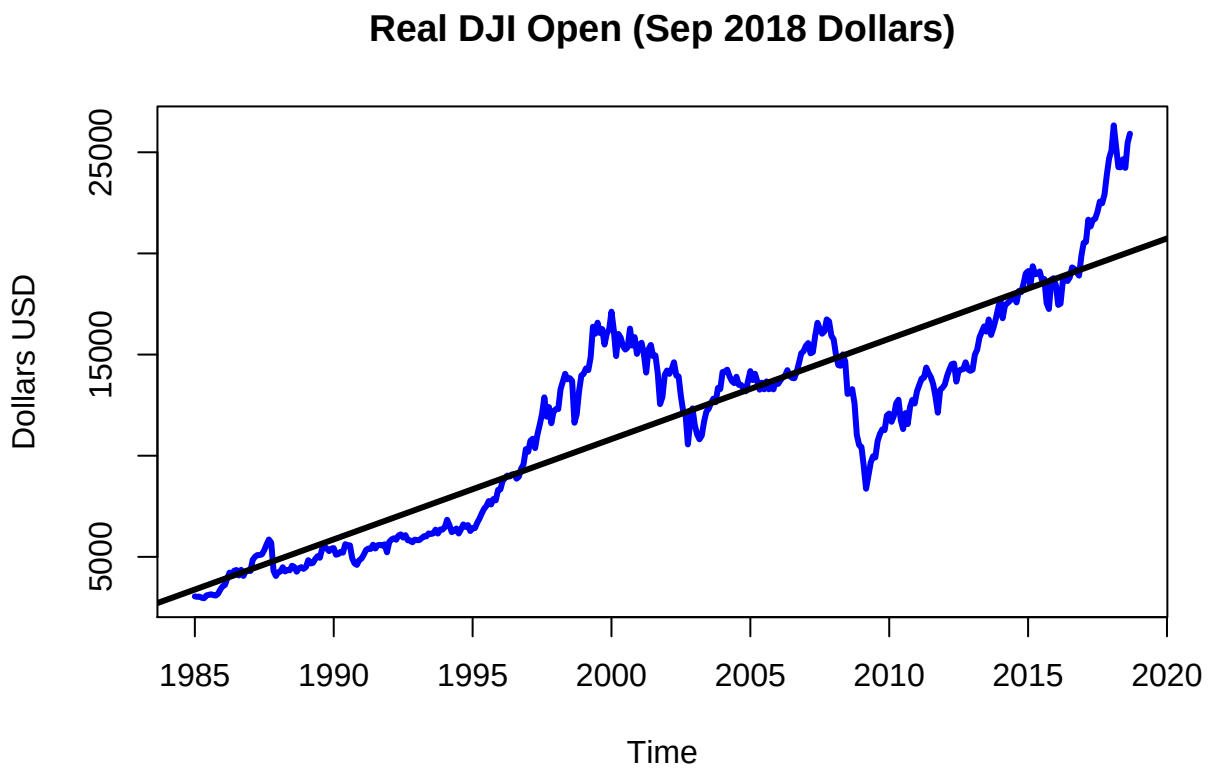

```
plot(stl(log(realDJIOpen), s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of the Log of Real DJI Open (Sep 2018 Dollars)')
```



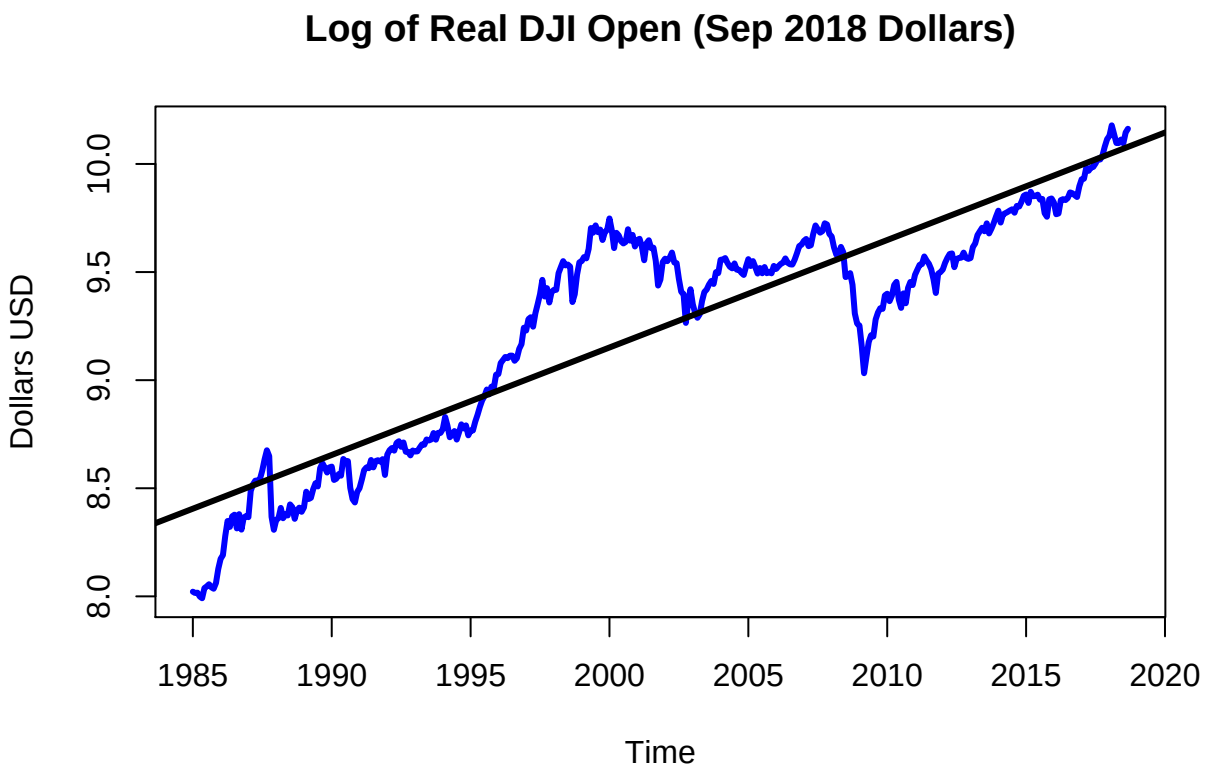

```
plot(stl(percRealDJIOpen, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of the Percent Change of Real DJI Open (Sep 2018 Dollars)')
```



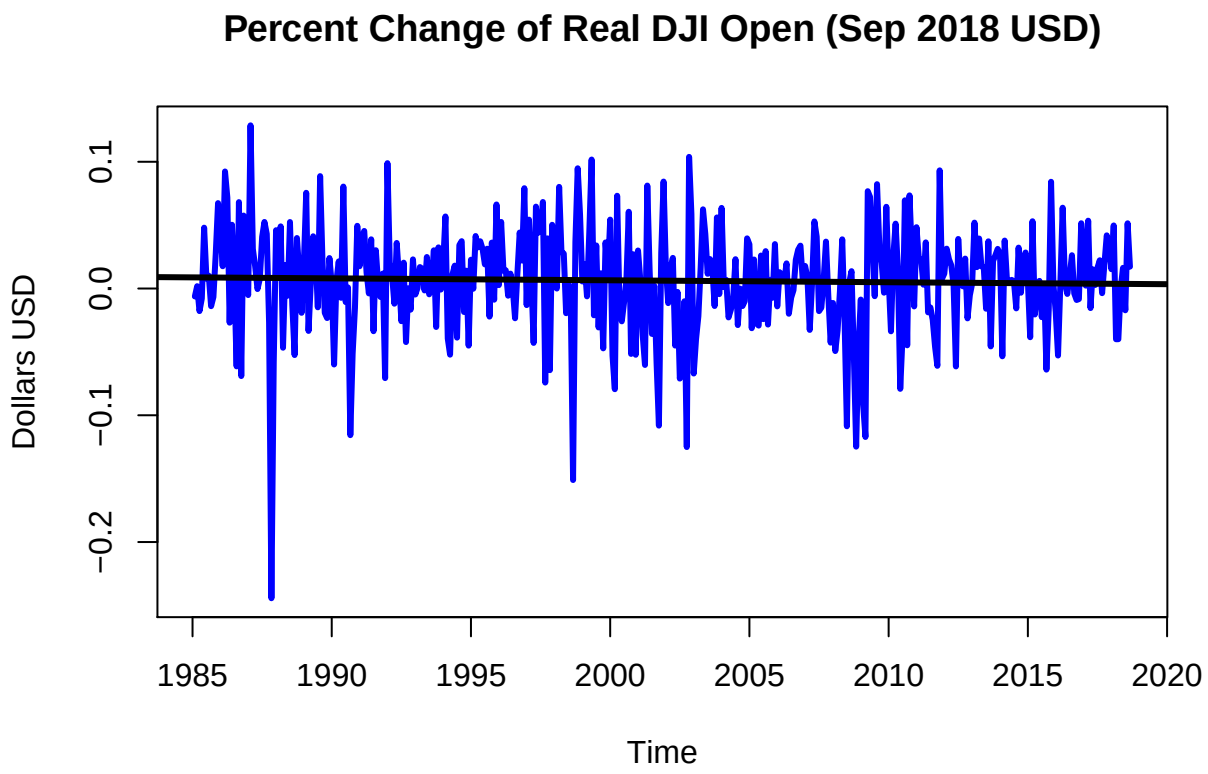

```
plot(realDJIOpen,
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(realDJIOpen ~ time(realDJIOpen)), lwd = 3)
title(main = 'Real DJI Open (Sep 2018 Dollars)')
```



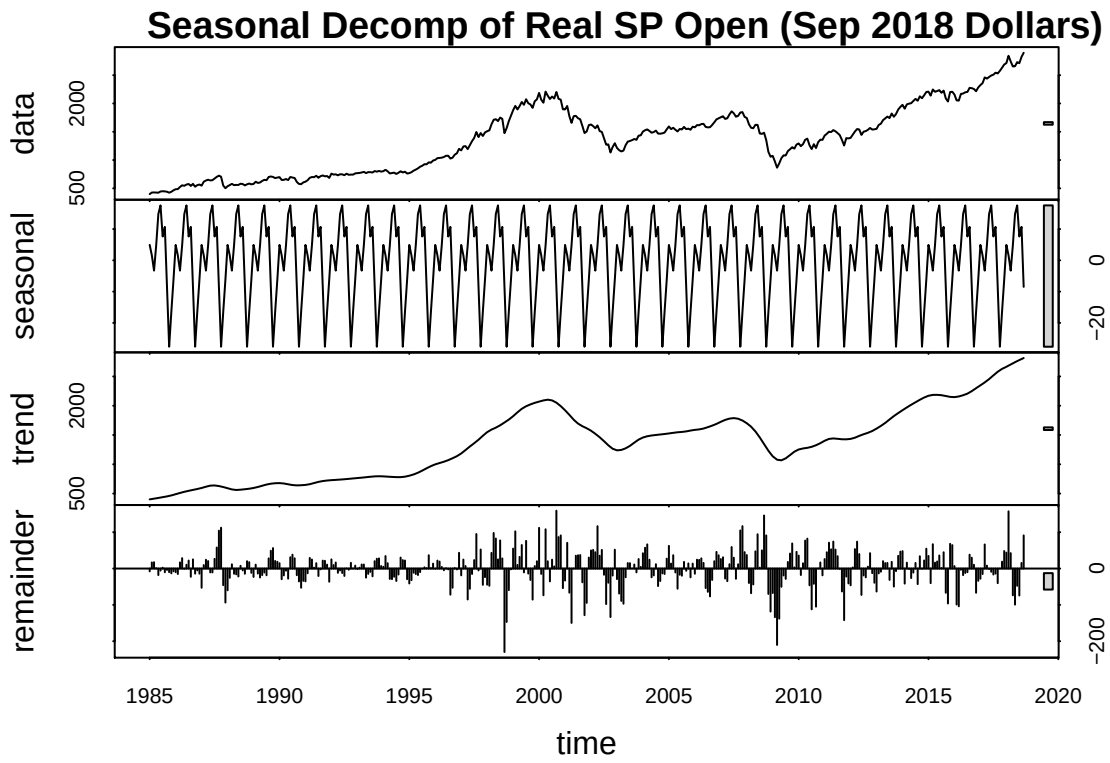

```
plot(log(realDJIOpen),
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(log(realDJIOpen) ~ time(log(realDJIOpen))), lwd = 3)
title(main = 'Log of Real DJI Open (Sep 2018 Dollars)')
```



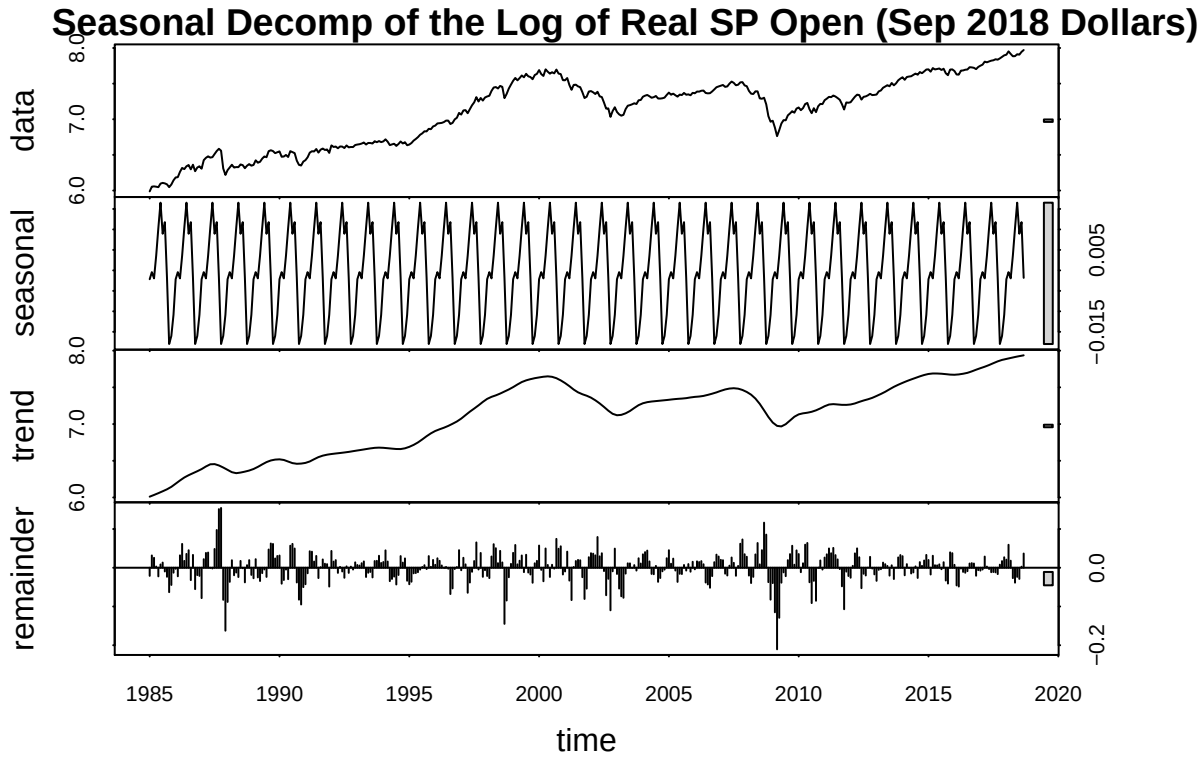

```
plot(percRealDJIOpen,
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(percRealDJIOpen ~ time(percRealDJIOpen)), lwd = 3)
title(main = 'Percent Change of Real DJI Open (Sep 2018 USD)')
```



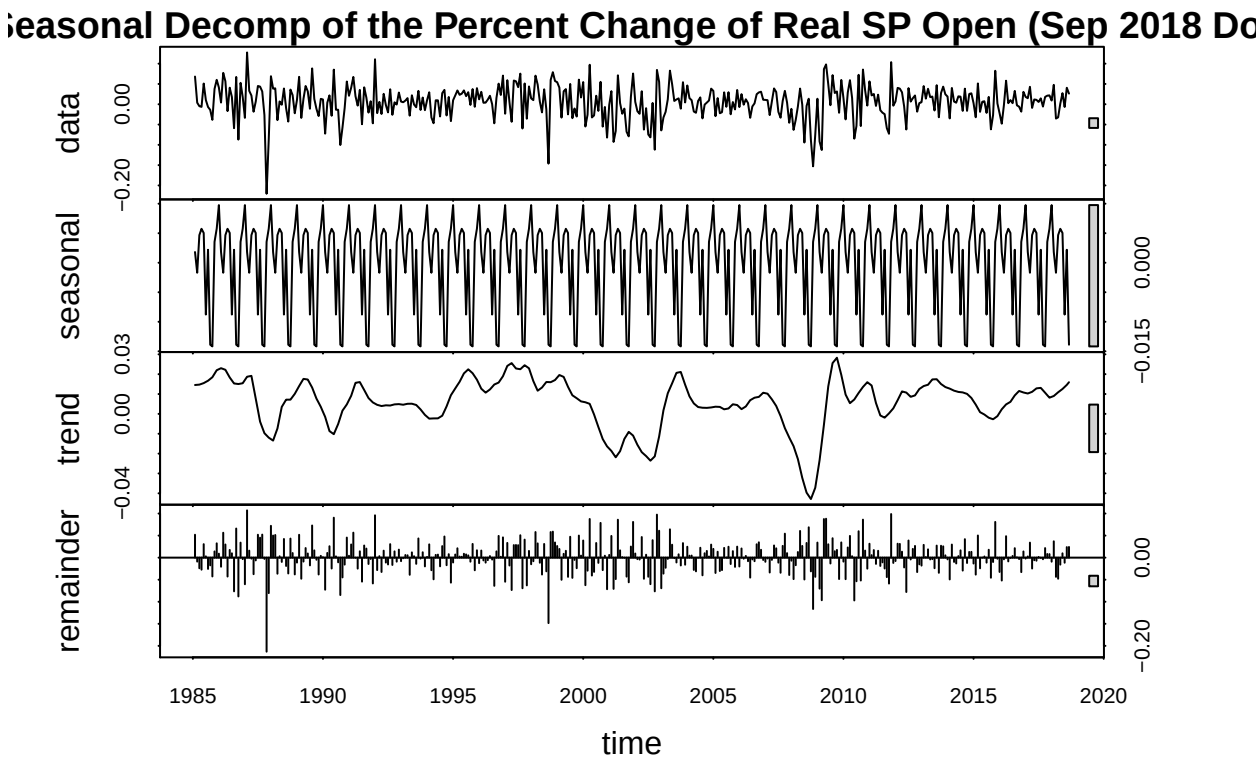

```
plot(stl(realSPOpen, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of Real SP Open (Sep 2018 Dollars)')
```




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

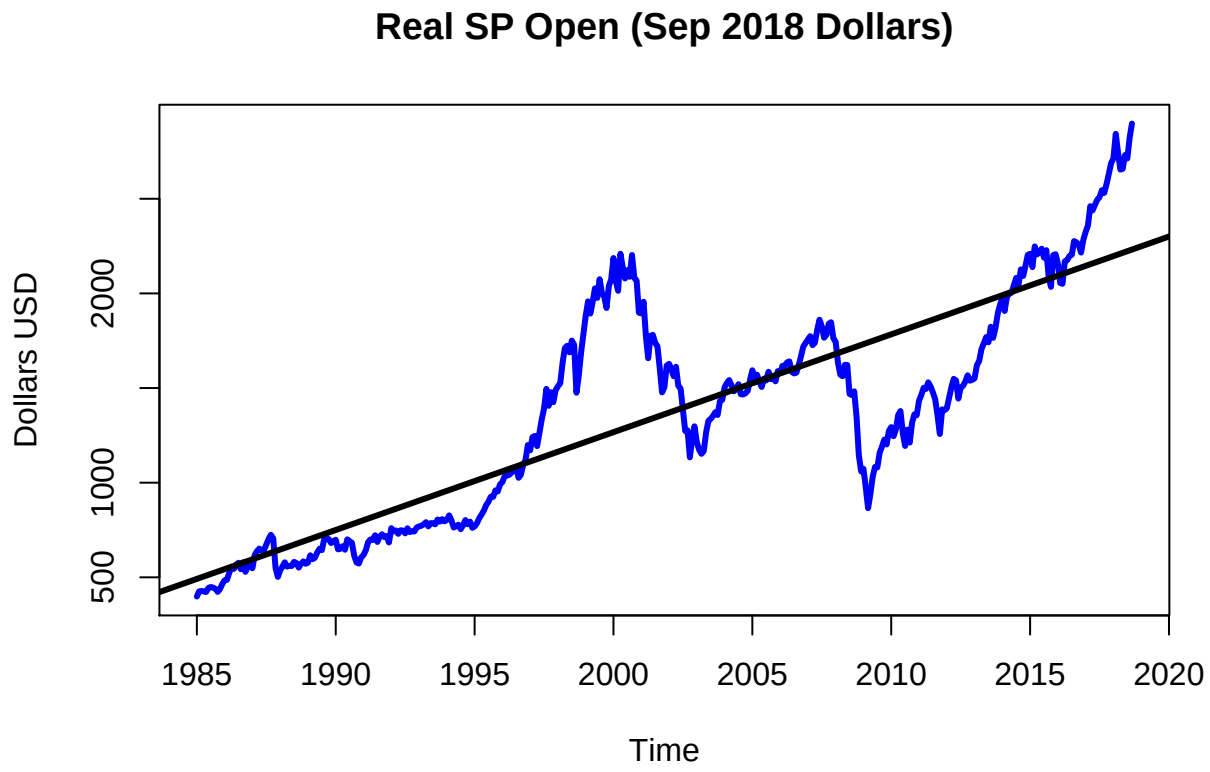



```
plot(stl(percRealSPOpen, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of the Percent Change of Real SP Open (Sep 2018 Dollars)')
```

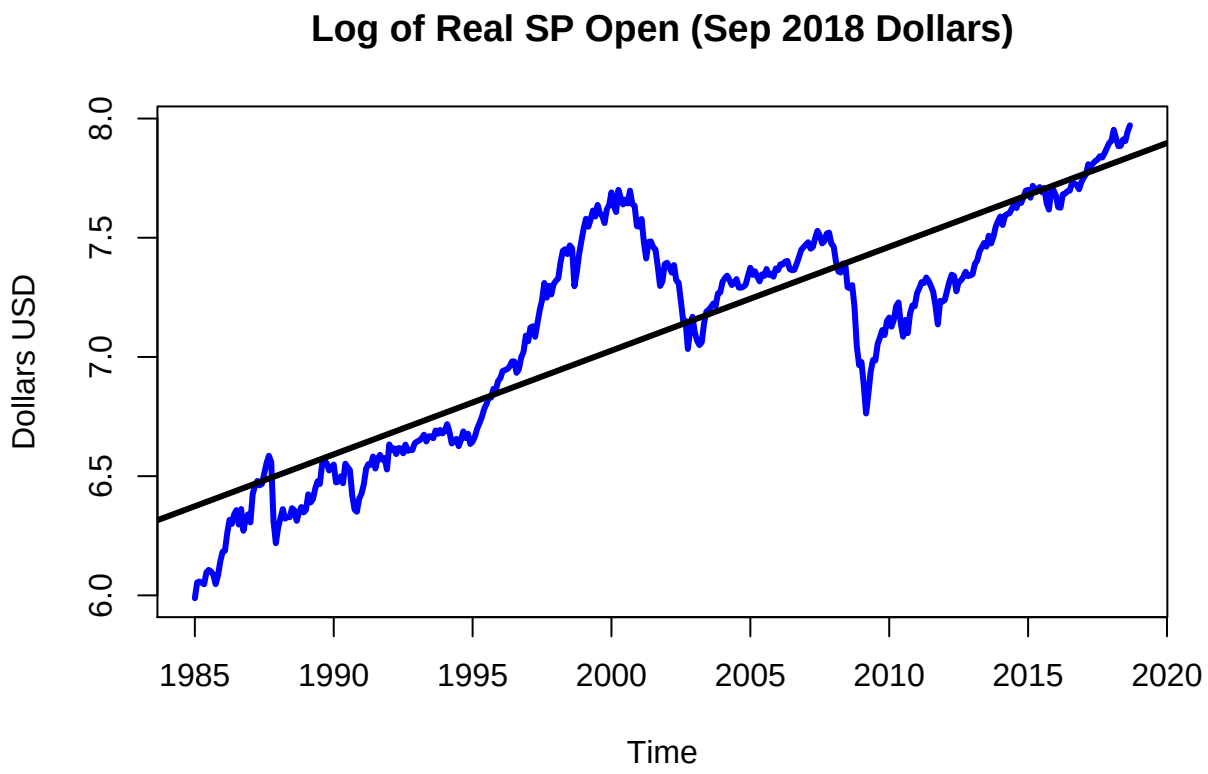



```
plot(realSPOpen,
     col = 'blue',
     lwd = 3,
```

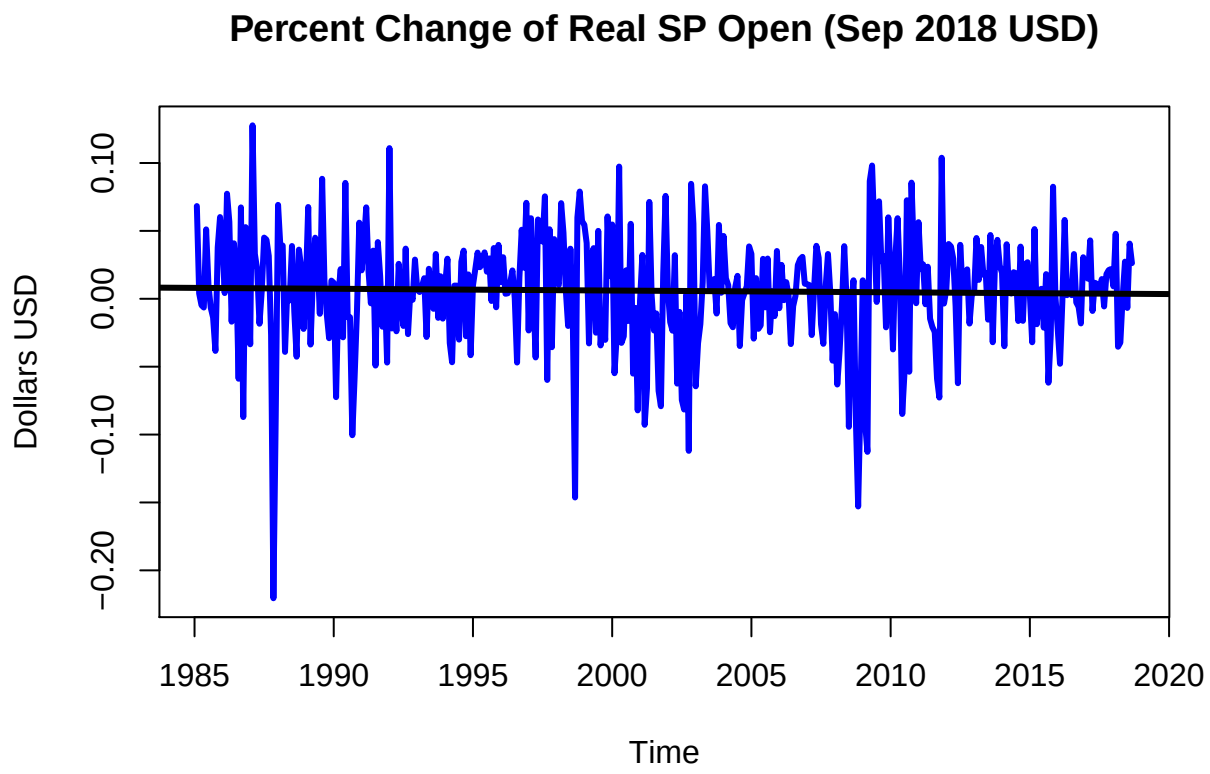
```
     ylab = 'Dollars USD')
abline(reg = lm(realSPOpen ~ time(realSPOpen)), lwd = 3)
title(main = 'Real SP Open (Sep 2018 Dollars)')
```



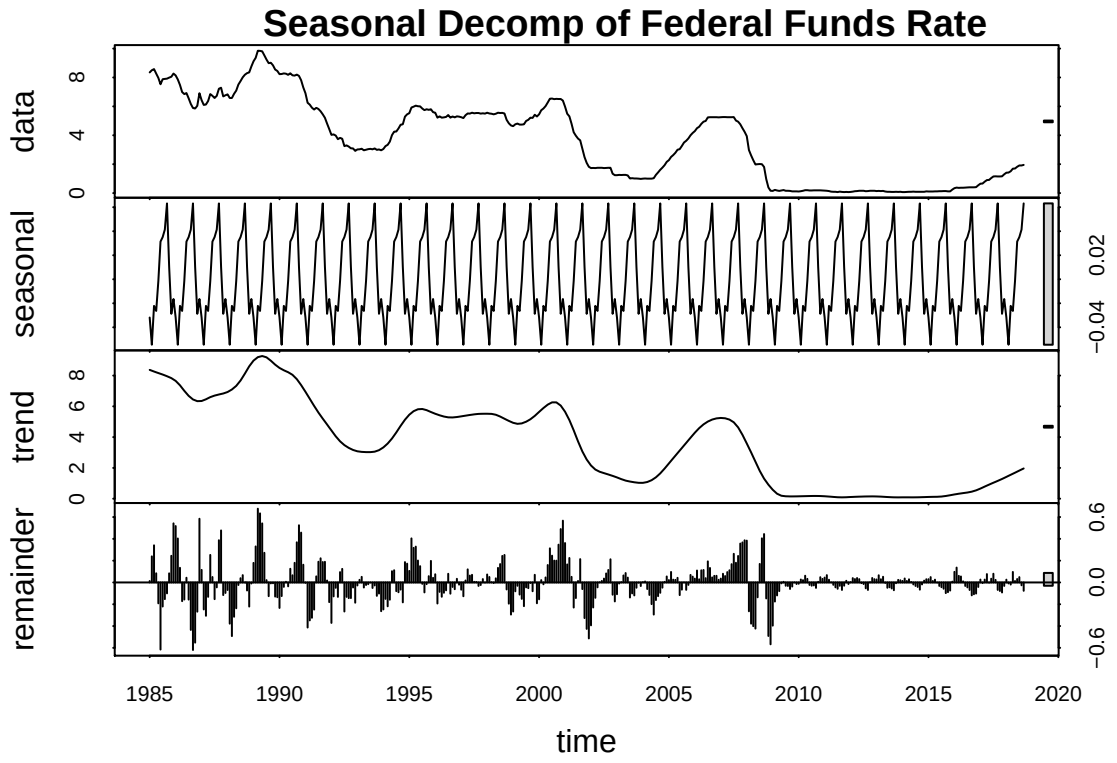

```
plot(log(realSPOpen),
     col = 'blue',
     lwd = 3,
     ylab = 'Dollars USD')
abline(reg = lm(log(realSPOpen) ~ time(log(realSPOpen))), lwd = 3)
title(main = 'Log of Real SP Open (Sep 2018 Dollars)')
```




```
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)')
```

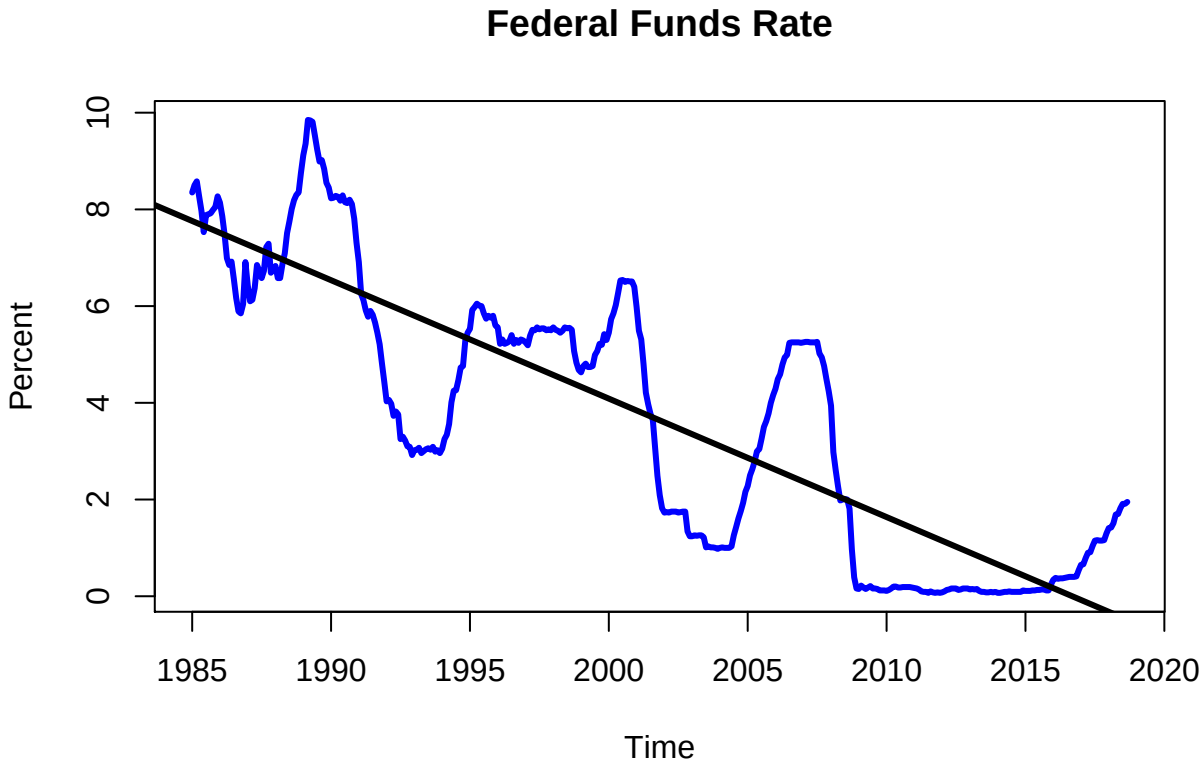



```
plot(stl(fedFund, s.window = "period"), lwd = 1)
title(main = 'Seasonal Decomp of Federal Funds Rate')
```

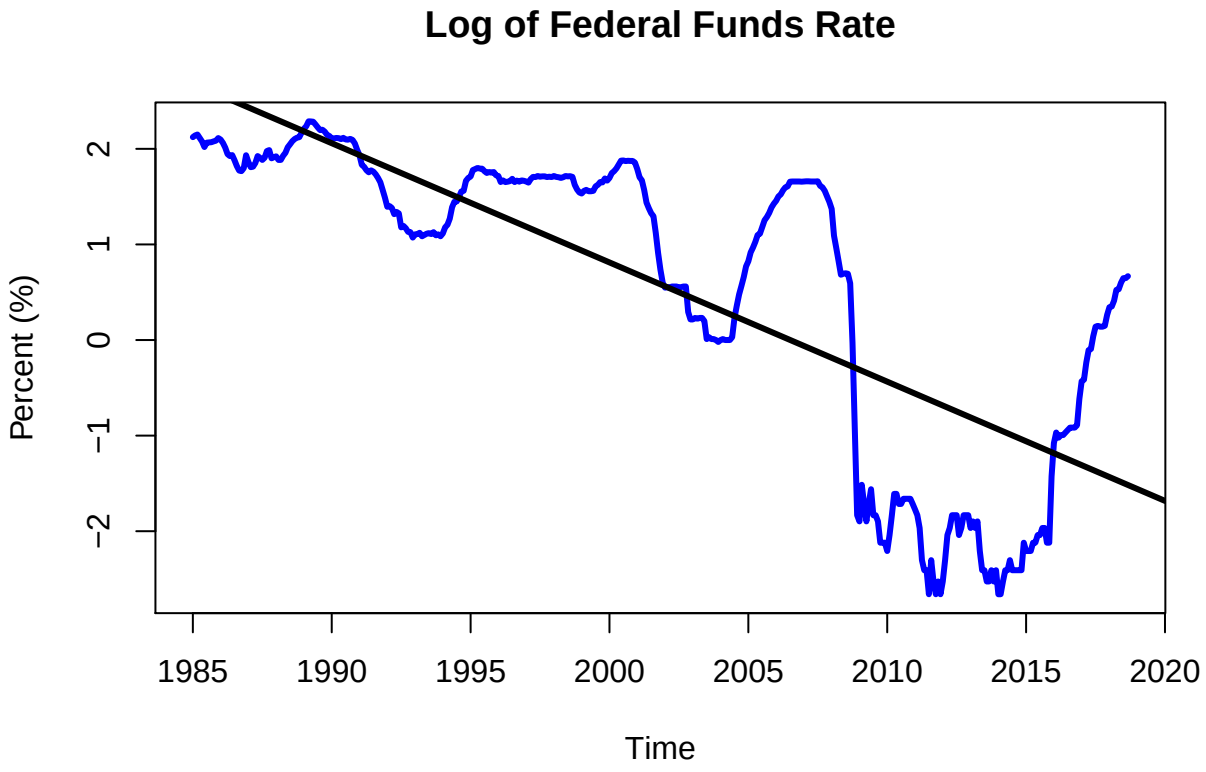



```
plot(fedFund,
     col = 'blue',
     lwd = 3,
```

```
     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')
```




```
# Hypothesis Tests ####
adf.test(realDJIOpen, alternative = 'stationary')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  realDJIOpen
## Dickey-Fuller = -1.4548, Lag order = 7, p-value = 0.8076
## alternative hypothesis: stationary
```

```
# Augmented Dickey-Fuller Test
# Fail to reject the null (0.8076 = p > 0.05), so it's likely non-stationary
adf.test(log(realDJIOpen), alternative = 'stationary')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  log(realDJIOpen)
## Dickey-Fuller = -2.3078, Lag order = 7, p-value = 0.4473
## alternative hypothesis: stationary
```

```
# Fail to reject the null (0.4473 = p > 0.05), so it's likely non-stationary still with the transform
adf.test(percRealDJIOpen, alternative = 'stationary')
```

```
## Warning in adf.test(percRealDJIOpen, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  percRealDJIOpen
## Dickey-Fuller = -6.7471, Lag order = 7, p-value = 0.01
```

```
## alternative hypothesis: stationary
```

```
# Reject the null (0.01 = p < 0.05), so it's likely stationary still with the transform
```

Will use percRealDJIOpen as a predictor

```
adf.test(realSPOpen, alternative = 'stationary')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  realSPOpen
## Dickey-Fuller = -1.5131, Lag order = 7, p-value = 0.783
## alternative hypothesis: stationary
```

```
# Augmented Dickey-Fuller Test
# Fail to reject the null (0.783 = p > 0.05), so it's likely non-stationary
adf.test(log(realSPOpen), alternative = 'stationary')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  log(realSPOpen)
## Dickey-Fuller = -2.1276, Lag order = 7, p-value = 0.5234
## alternative hypothesis: stationary
```

```
# Fail to reject the null (0.5234 = p > 0.05), so it's likely non-stationary still with the transform
adf.test(percRealSPOpen, alternative = 'stationary')
```

```
## Warning in adf.test(percRealSPOpen, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  percRealSPOpen
## Dickey-Fuller = -6.583, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```
# Reject the null (0.01 = p < 0.05), so it's likely stationary still with the transform
```

Will use percRealSPOpen as a predictor

```
adf.test(fedFund, alternative = 'stationary')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  fedFund
## Dickey-Fuller = -3.4096, Lag order = 7, p-value = 0.05242
## alternative hypothesis: stationary
```

```
# Fail to reject the null (0.05242 = p > 0.05), so it's on the cusp of being likely non-stationary
```

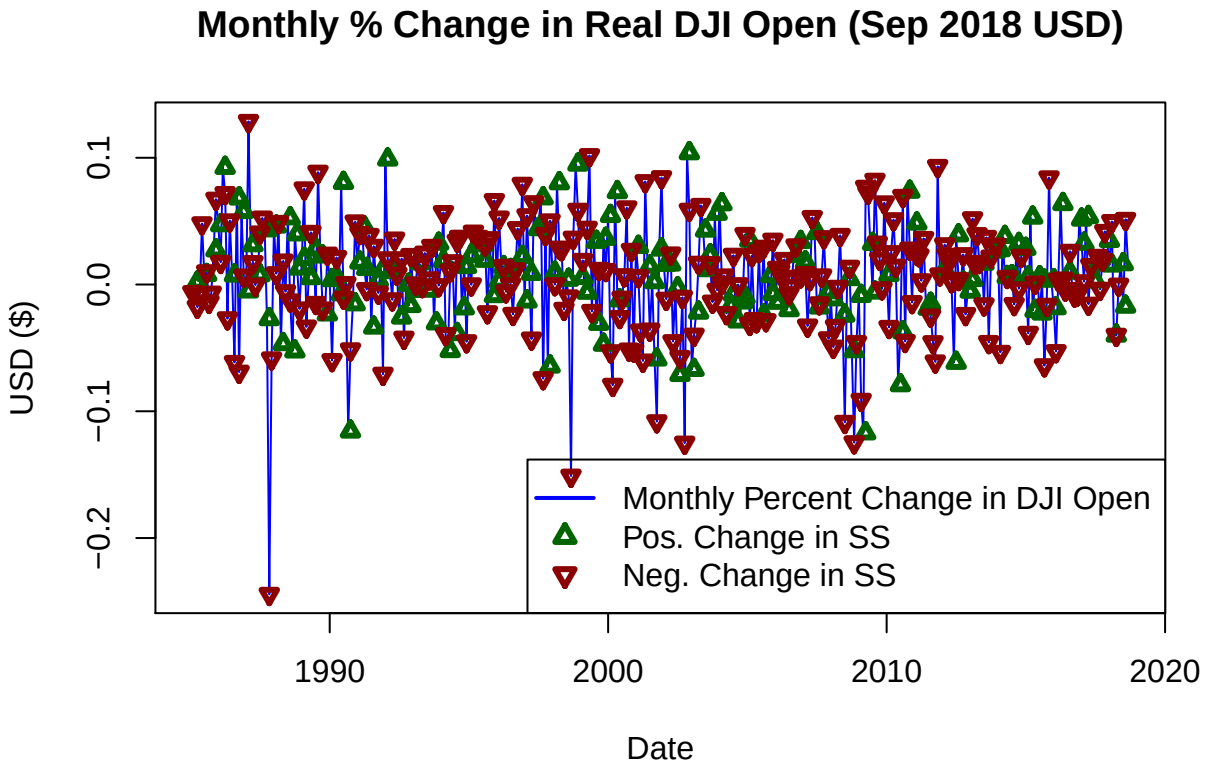
Will try model with fedFund as a predictor

```
adf.test(log(fedFund), alternative = 'stationary')
```

```
##
##  Augmented Dickey-Fuller Test
```

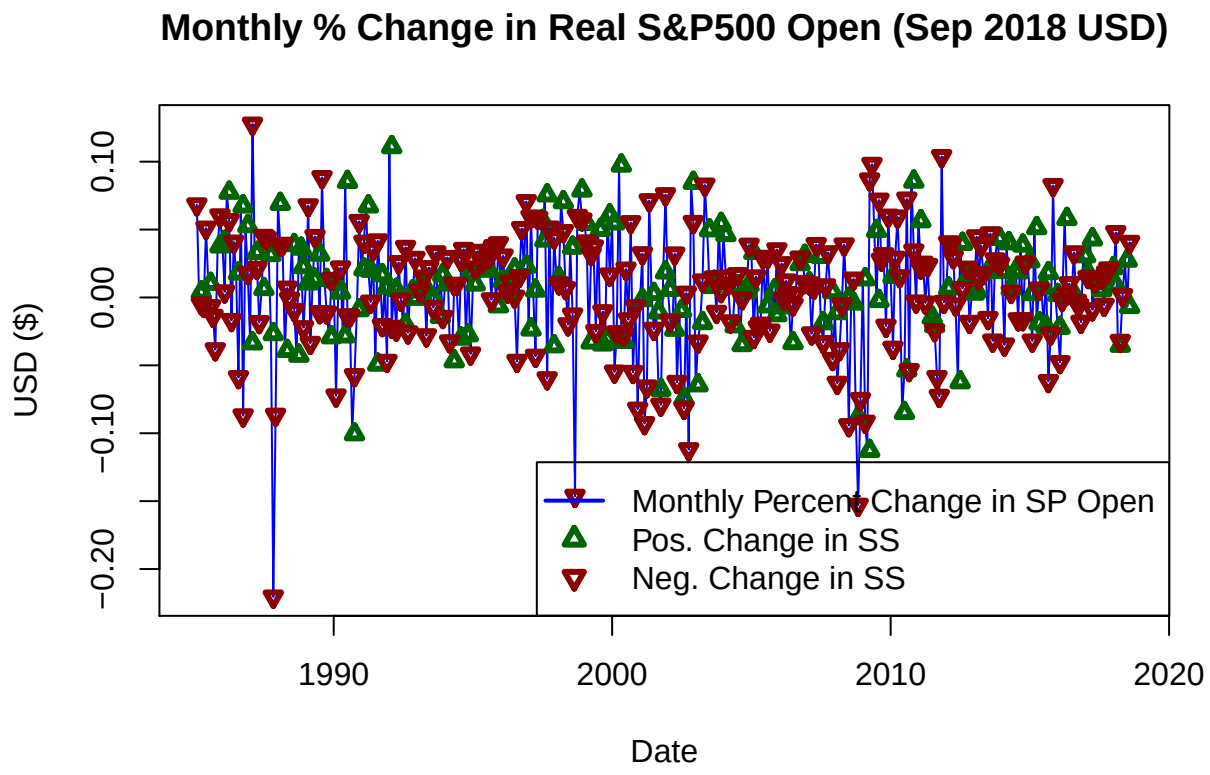
```
##
## data:  log(fedFund)
## Dickey-Fuller = -1.7739, Lag order = 7, p-value = 0.6728
## alternative hypothesis: stationary
```

```
# Fail to reject the null (0.6728 = p > 0.05), so it's likely non-stationary with the transform

# Remove nominal values aside indicators of positive change
df2 <- df1[, c(1, 8:10, 11:18, 32:44, 58:96, 110:122)]
# Visualizations ####
plot(
  x = df2$date,
  y = df2$percChangeRealDJIopen,
  col = 'blue',
  lwd = 1,
  type = 'l',
  ylab = 'USD ($)',
  xlab = 'Date'
)
points(
  x = df2$date[df$postotalSSRetired == 1],
  y = df2$percChangeRealDJIopen[df2$postotalSSRetired == 1],
  pch = 24,
  col = 'darkgreen',
  cex = 0.8,
  lwd = 3
)
points(
  x = df2$date[df2$postotalSSRetired == 0],
  y = df2$percChangeRealDJIopen[df2$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)')
```



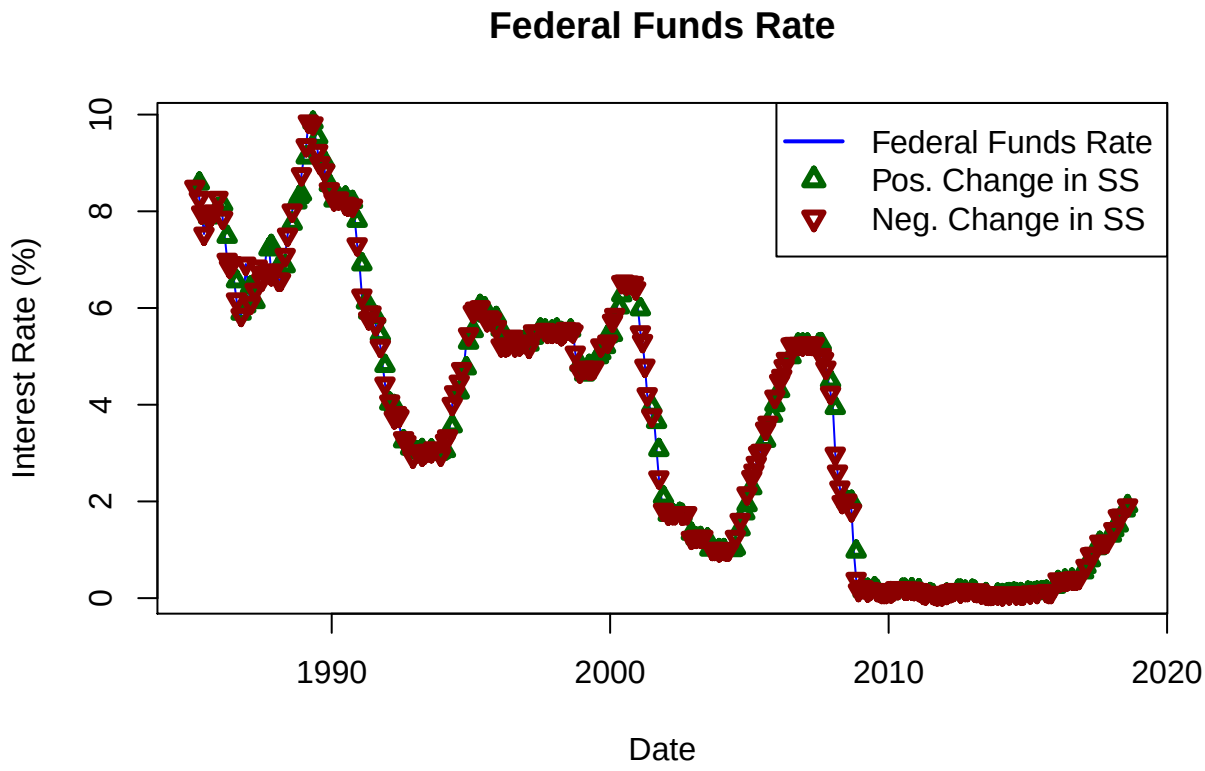

```
plot(
  x = df2$date,
  y = df2$percChangeRealSPopen,
  col = 'blue',
  lwd = 1,
  type = 'l',
  ylab = 'USD ($)',
  xlab = 'Date'
)
points(
  x = df2$date[df$postotalSSRetired == 1],
  y = df2$percChangeRealSPopen[df2$postotalSSRetired == 1],
  pch = 24,
  col = 'darkgreen',
  cex = 0.8,
  lwd = 3
)
points(
  x = df2$date[df2$postotalSSRetired == 0],
  y = df2$percChangeRealSPopen[df2$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 = df2$date,
  y = df2$fedFundRate,
  col = 'blue',
  lwd = 1,
  type = 'l',
  ylab = 'Interest Rate (%)',
  xlab = 'Date'
)
points(
  x = df2$date[df$postotalSSRetired == 1],
  y = df2$fedFundRate[df2$postotalSSRetired == 1],
  pch = 24,
  col = 'darkgreen',
  cex = 0.8,
  lwd = 3
)
points(
  x = df2$date[df2$postotalSSRetired == 0],
  y = df2$fedFundRate[df2$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')
```




```
# Selection ####
regFitSelect <- regsubsets(
  postotalSSRetired~realDJIopen + realDJIhigh + realDJIlow + realDJIclose + realSPopen + realSPhigh + r
                           data=df2,
                           really.big=T,
                           intercept = F)
regSummary <- summary(regFitSelect)
names(regSummary)
```

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

```
regSummary$rsq                          # rsq for the best model with given number of predictors.
```

```
## [1] 0.3447995 0.3505265 0.3562218 0.3609035 0.3613143 0.3633166 0.3639426
```

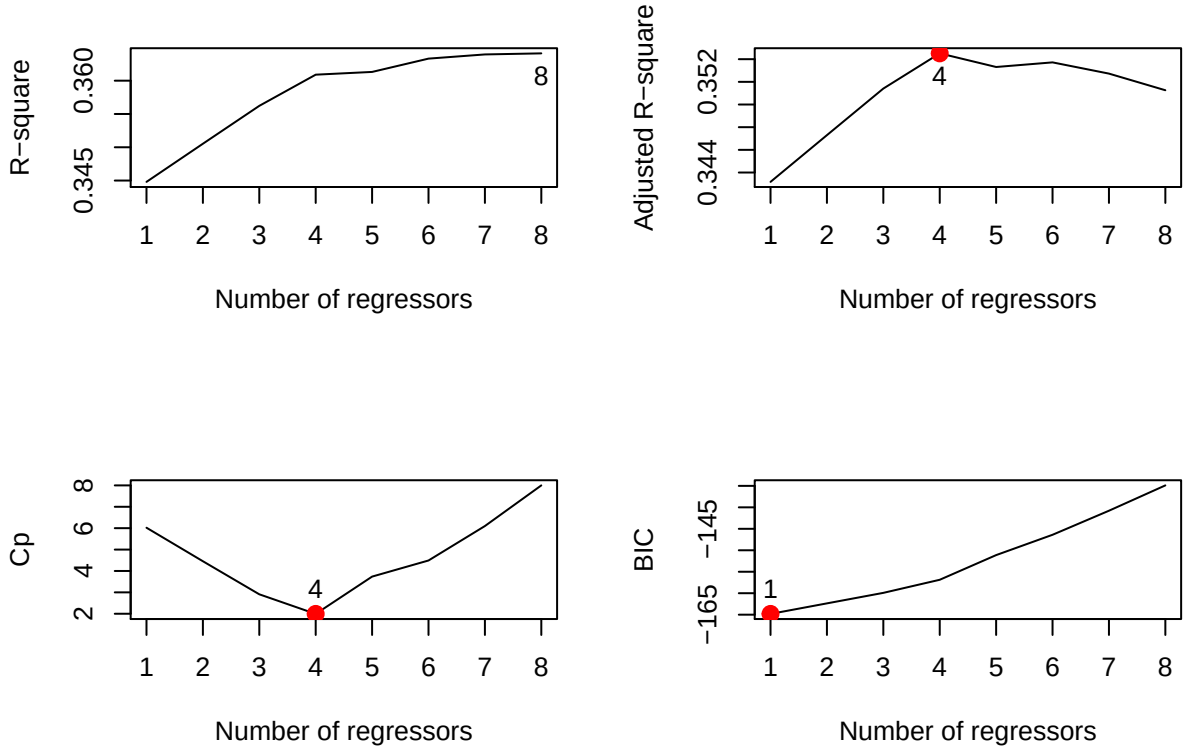
```
## [8] 0.3641037
```

```
regSummary$adjr2
```

```
## [1] 0.3431737 0.3472953 0.3514056 0.3545126 0.3533107 0.3537183 0.3527275
## [8] 0.3512573
```

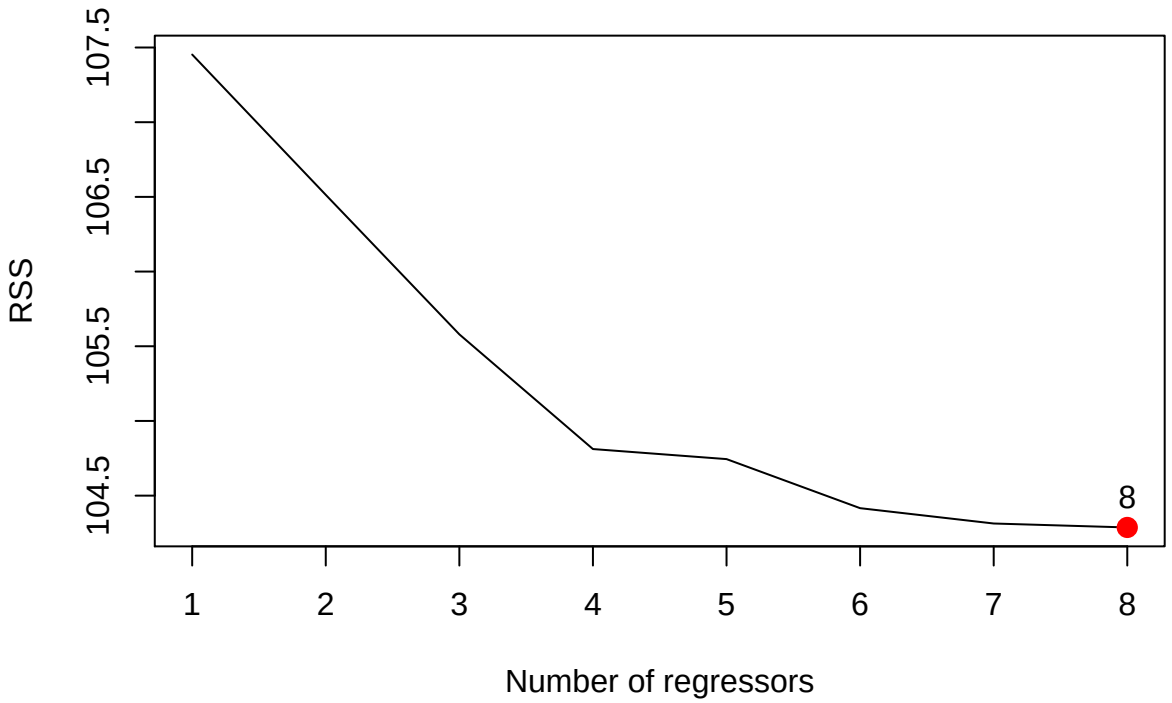
```
par(mfrow=c(2,2))
aRSQ <- which.max(regSummary$rsq)
aARSQ <- which.max(regSummary$adjr2)
aCP <- which.min(regSummary$cp)
aBIC <- which.min(regSummary$bic)
aRSS <- which.min(regSummary$rss)

par(mfrow = c(2, 2))

plot(
  regSummary$rsq,
  xlab = "Number of regressors",
  ylab = "R-square",
  type = "l"
)
text(aRSQ,
     regSummary$rsq[aRSQ],
     labels = aRSQ,
     pos = 1)

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

plot(regSummary$cp,
     xlab = "Number of regressors",
     ylab = "Cp",
     type = "l")
points(
  aCP,
  regSummary$cp[aCP],
  col = "red",
  cex = 2,
  pch = 20
```

```
)
text(aCP,
     regSummary$cp[aCP],
     labels = aCP,
     pos = 3)

plot(
  regSummary$bic,
  xlab = "Number of regressors",
  ylab = "BIC",
  type = "l"
)
points(
  aBIC,
  regSummary$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",
  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(regFitFull, scale = "r2")
# plot(regFitFull, scale = "adjr2")
# plot(regFitFull, scale = "Cp")
# plot(regFitFull, scale = "bic")

# Model ####
# Setting train/test split
set.seed(1)
trainSample <- sample(1:nrow(df2), round(nrow(df2)/2), replace = F)
trainData <- df2[trainSample,]
testData <- df2[-trainSample,]

trainX <- trainData[,c(1, 5:77)]
trainY <- trainData[,c(1:4)]
testX <- testData[,c(1, 5:77)]
trainY <- testData[,c(1:4)]
```

```r
# Basic logistic
glmFit <- glm(postotalSSRetired ~ realDJIopen + realDJIhigh + realDJIlow + realDJIclose + realSPopen, fa
summary(glmFit)
```

```
##
## Call:
## glm(formula = postotalSSRetired ~ realDJIopen + realDJIhigh +
##     realDJIlow + realDJIclose + realSPopen, family = binomial,
##     data = df2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4705  -1.0154  -0.9387   1.3238   1.5393
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.4376422  0.2628795  -1.665   0.0960 .
## realDJIopen  -0.0002092  0.0004390  -0.477   0.6337
## realDJIhigh   0.0006851  0.0005705   1.201   0.2298
## realDJIlow    0.0004898  0.0003681   1.330   0.1834
## realDJIclose -0.0009853  0.0004751  -2.074   0.0381 *
## realSPopen    0.0001988  0.0011179   0.178   0.8588
## ---
## 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: 540.50  on 398  degrees of freedom
## AIC: 552.5
##
## Number of Fisher Scoring iterations: 4
```