W271 HW6

Subhashini R., Lei Yang, Ron Cordell

March 16, 2016

Exercise 1:

a. Discuss the mean and variance functions and how the similarities and differences from those we studied in classical linear model

The Mean Function for a time series is given by:

$$u_x(t) = E(x_t) = \int_{-\infty}^{\infty} x_t f_t(x_t) dx_t$$

The mean function is dependent on time t and the expectation is over the ensemble of time series x_t derived from the underlying process.

A process is stationary in the mean if the mean function is a constant in time.

The Variance Function for a time series is given by:

$$\sigma_x^2(t) = E(x_t - \mu_x(t))^2 = \int_{-\infty}^{\infty} (x_t - \mu_x(t))^2 f_t(x_t) dx_t$$

Once again the expectation is for the ensemble of all time series.

A series is stationary in variance if ther variance function is constant in time.

b. Define strict and weak stationarity

A time series is strictly stationary if the joint distributions $F(x_{t_1}, ..., x_{t_n})$ and $F(x_{t_1+m}, ..., x_{t_n+m})$ are the same $\forall t_1, ..., t_n$ and m. In other words, the distribution is the same regardless of the time period.

A time series is weakly stationary if it is mean and variance stationary and the autocovariance $Cov(x_t, x_{t+k})$ depends on the time placement k.

Exercise 2:

a. Generate a zero-drift random walk model using 500 simulation
b. Provide the descriptive statistics of the simulated realizations. The descriptive statistics should include the mean, standard deviation, 25th, 50th, and 75th quantiles, minimum, and maximum
c. Plot the time-series plot of the simulated realizations
d. Plot the autocorrelation graph
e. Plot the partial autocorrelation graph

```
# set seed for repeatability
set.seed(1)

# 500 random normal "draws"
x1 <- w <- rnorm(500)
```

```r
# create the lagged variable
for (i in 2:500) x1[i] <- x1[i-1] + w[i]

# let's see what we've got
str(x1)
```

```
##  num [1:500] -0.626 -0.443 -1.278 0.317 0.646 ...
```

```r
str(w)
```

```
##  num [1:500] -0.626 0.184 -0.836 1.595 0.33 ...
```

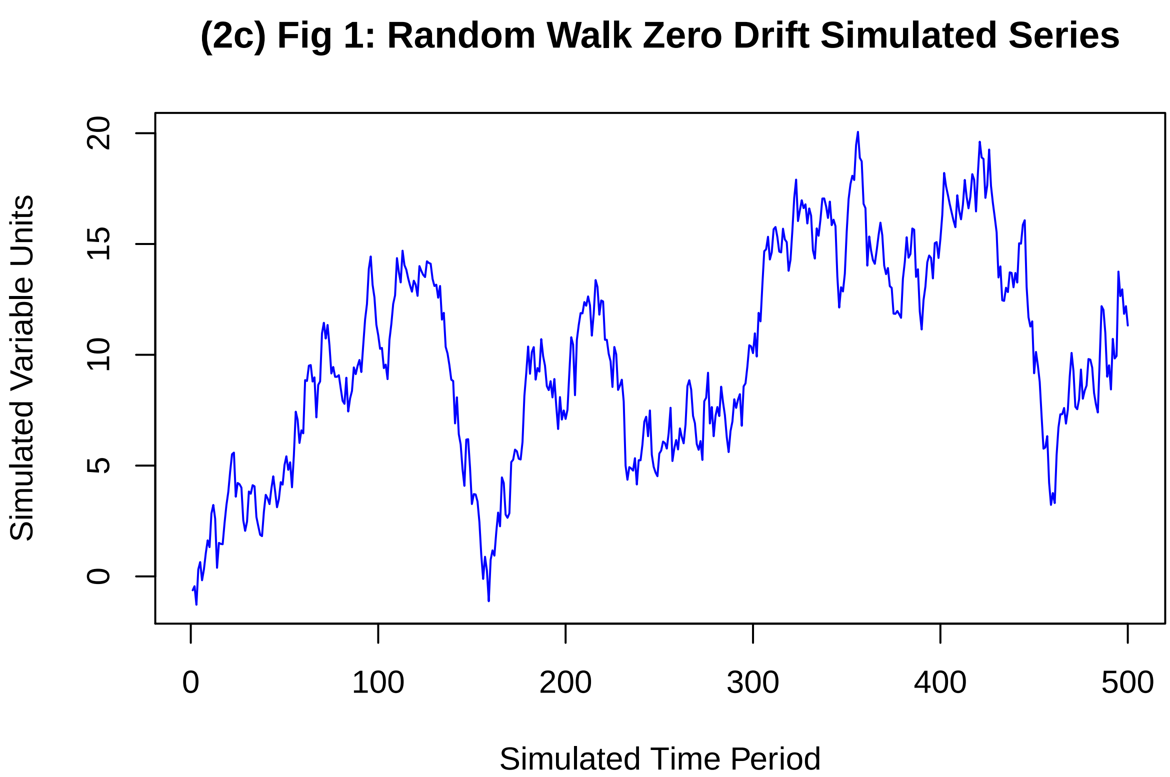
```r
# summary stats
summary(x1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.278   6.187   9.526   9.838  13.690  20.060
```

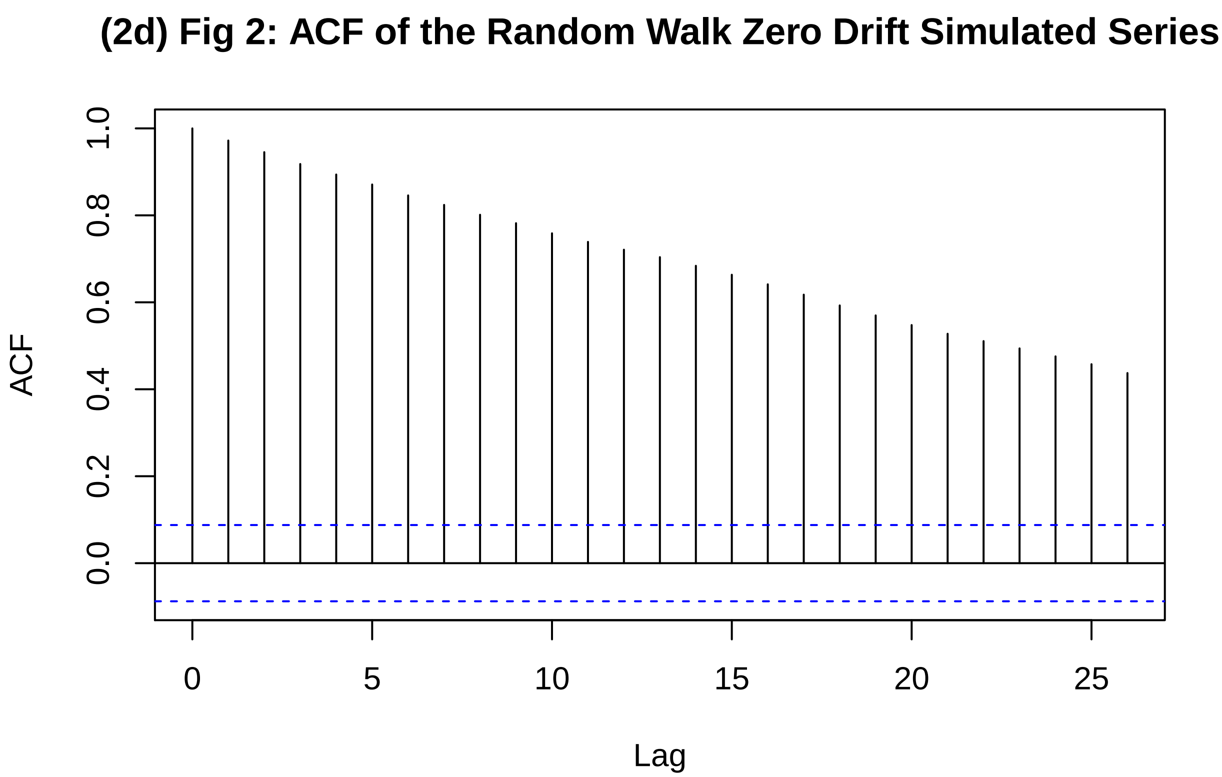
```r
sd(x1)
```

```
## [1] 4.723985
```

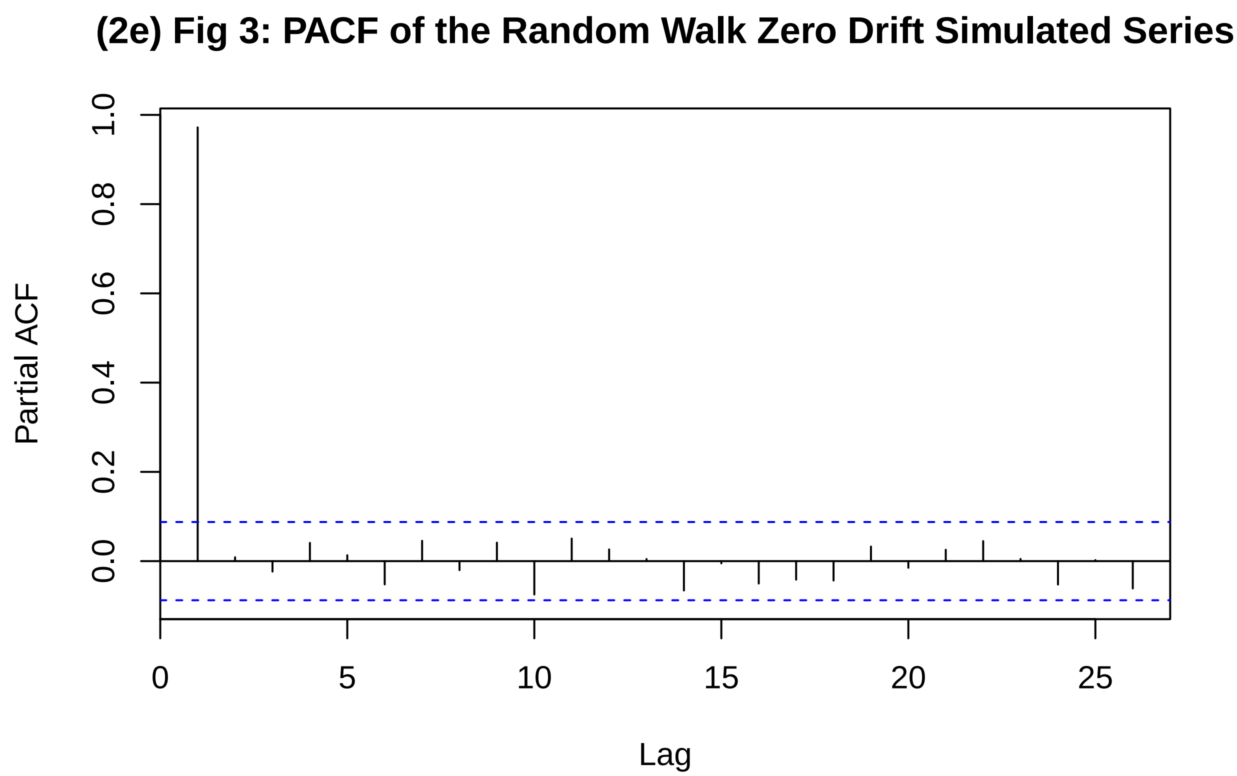
```r
# put our graphs in the same page/display canvas
plot.ts(x1, col='blue',
        xlab='Simulated Time Period',
        ylab='Simulated Variable Units')
title('(2c) Fig 1: Random Walk Zero Drift Simulated Series')
```



```r
acf(x1, main="",
    xlab='Lag')
title('(2d) Fig 2: ACF of the Random Walk Zero Drift Simulated Series')
```




```r
pacf(x1, main="",
     xlab='Lag')
title('(2e) Fig 3: PACF of the Random Walk Zero Drift Simulated Series')
```



Exercise 3:

a. Generate a random walk with drift model using 500 simulation, with the drift = 0.5
b. Provide the descriptive statistics of the simulated realizations. The descriptive statistics should include the mean, standard deviation, 25th, 50th, and 75th quantiles, minimum, and maximum
c. Plot the time-series plot of the simulated realizations
d. Plot the autocorrelation graph
e. Plot the partial autocorrelation graph

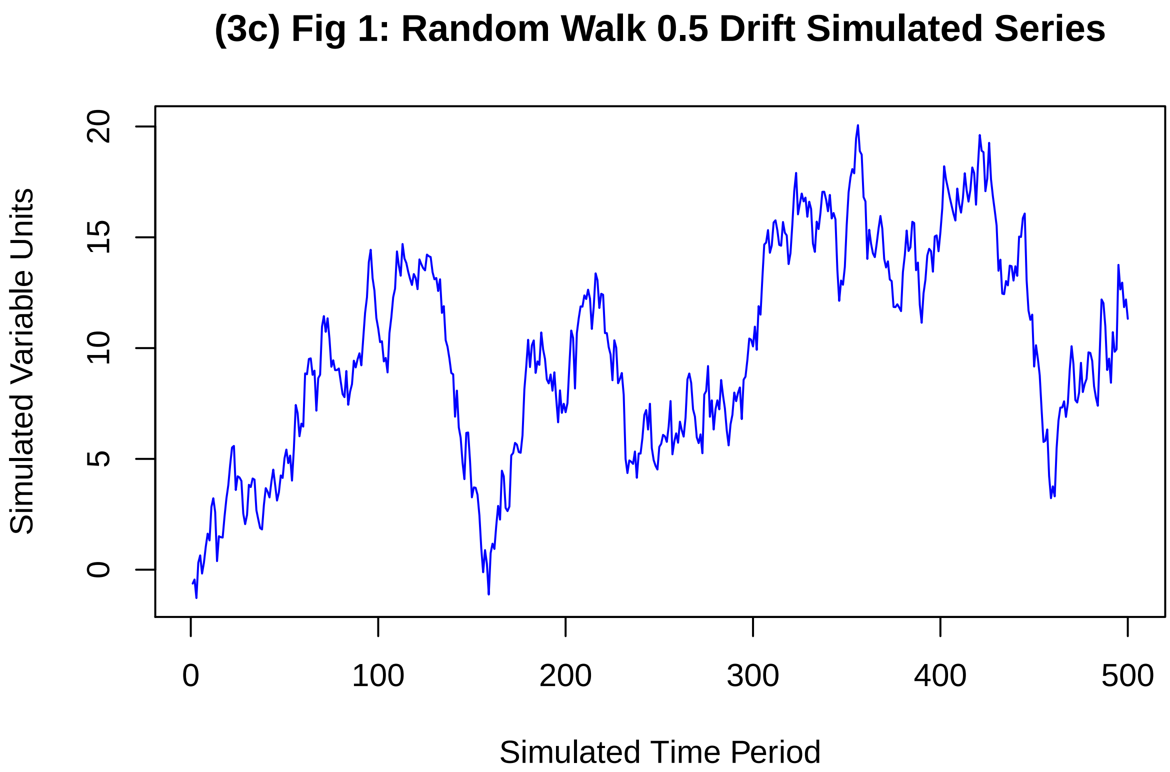
```
# we'll use our same series w
x2 <- w

# set the drift to 0.5
drift <- 0.5

# calculated the lagged variable with the drift
for (i in 2:500) x2[i] <- drift + x2[i-1] + w[i]

# summary stats
summary(x2)
```

```
##     Min.   1st Qu.   Median     Mean   3rd Qu.     Max.
##  -0.6265   76.3200 130.4000 134.6000 199.5000 261.2000
```

```r
sd(x2)
```

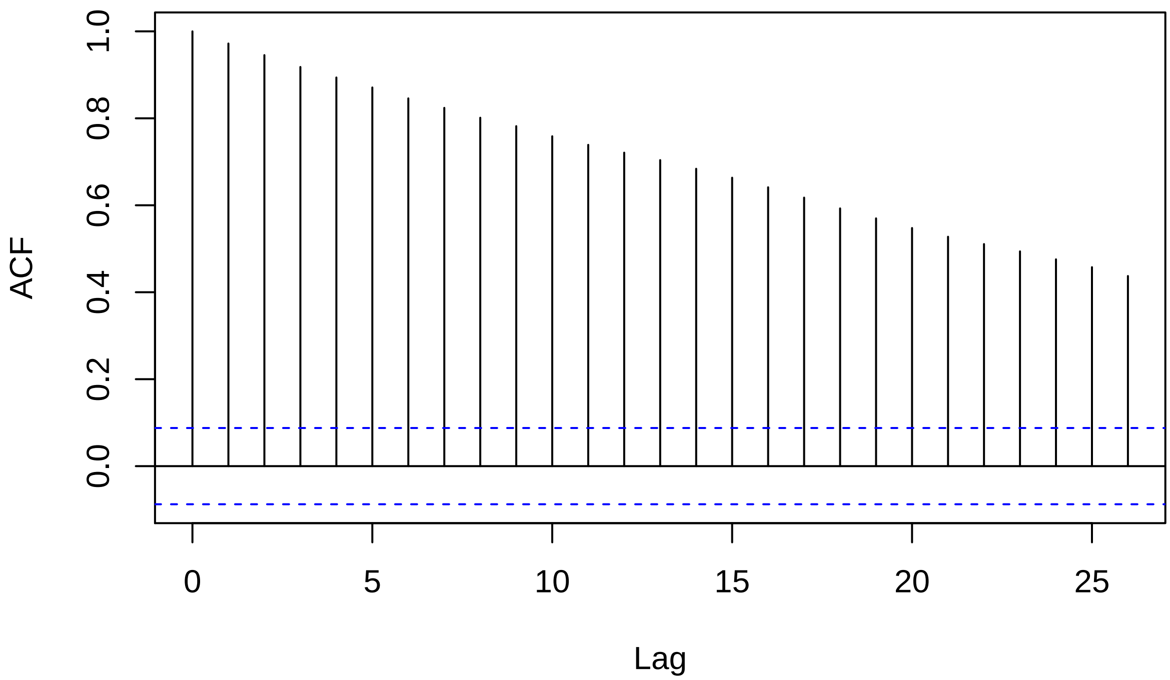
```
## [1] 74.88504
```

```r
# put our graphs in the same page/display canvas
plot.ts(x1, col='blue',
        xlab='Simulated Time Period',
        ylab='Simulated Variable Units')
title('(3c) Fig 1: Random Walk 0.5 Drift Simulated Series')
```



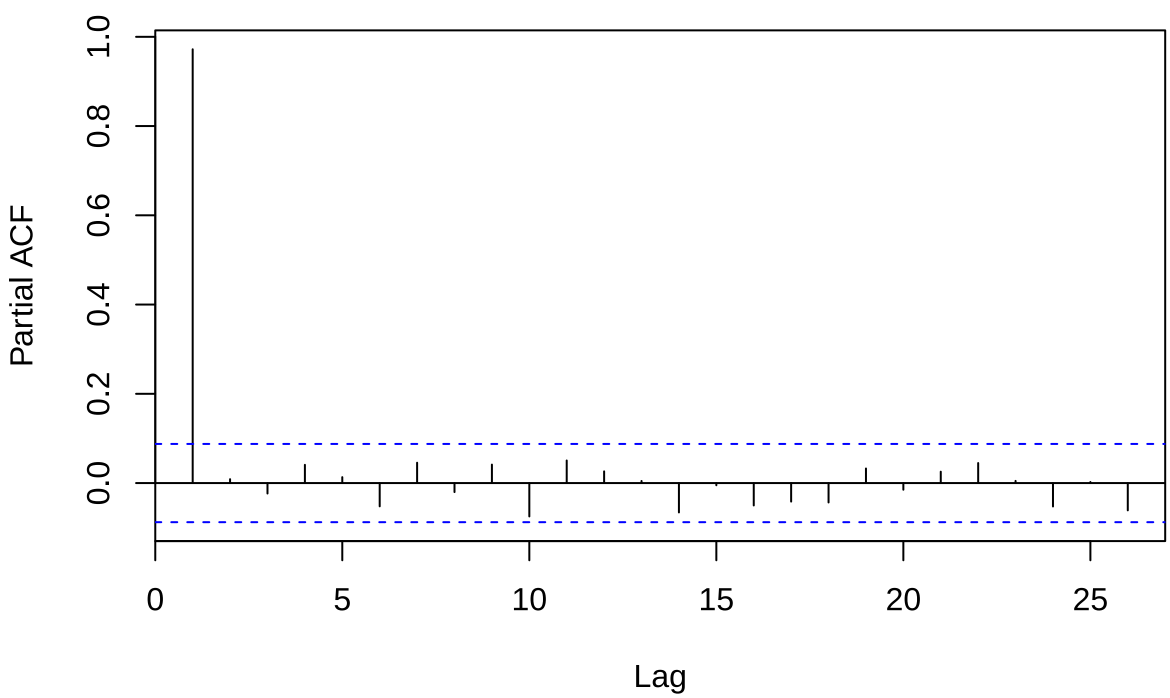

```r
acf(x1, main="",
    xlab='Lag')
title('(3d) Fig 2: ACF of the Random Walk 0.5 Drift Simulated Series')
```

(3d) Fig 2: ACF of the Random Walk 0.5 Drift Simulated Series




```r
pacf(x1, main="",
     xlab='Lag')
title('(3e) Fig 3: PACF of the Random Walk 0.5 Drift Simulated Series')
```

(3e) Fig 3: PACF of the Random Walk 0.5 Drift Simulated Series



Exercise 4:

Use the series from INJCJC.csv

a. Load the data and examine the basic structure of the data using str(), dim(), head(), and tail() functions

```
# load the INJCJC.csv data into a dataframe
df <- read.csv('INJCJC.csv')

# examine the structure
str(df)
```

```
## 'data.frame':    1300 obs. of  3 variables:
##  $ Date   : Factor w/ 1300 levels "1-Apr-05","1-Apr-11",..: 1102 143 442 784 483 1271 312 654 498 128
##  $ INJCJC : int  355 369 375 345 368 367 348 350 351 349 ...
##  $ INJCJC4: num  362 366 364 361 364 ...
```

```
# examine the dimensions of the data set
dim(df)
```

```
## [1] 1300    3
```

```
# examine the first 20 rows
head(df,20)
```

```
##         Date INJCJC INJCJC4
## 1   5-Jan-90    355  362.25
## 2  12-Jan-90    369  365.75
## 3  19-Jan-90    375  364.25
## 4  26-Jan-90    345  361.00
## 5   2-Feb-90    368  364.25
## 6   9-Feb-90    367  363.75
## 7  16-Feb-90    348  357.00
## 8  23-Feb-90    350  358.25
## 9   2-Mar-90    351  354.00
## 10  9-Mar-90    349  349.50
## 11 16-Mar-90    349  349.75
## 12 23-Mar-90    331  345.00
## 13 30-Mar-90    346  343.75
## 14  6-Apr-90    367  348.25
## 15 13-Apr-90    357  350.25
## 16 20-Apr-90    360  357.50
## 17 27-Apr-90    363  361.75
## 18  4-May-90    354  358.50
## 19 11-May-90    355  358.00
## 20 18-May-90    353  356.25
```

```
#examine the last 20 rows
tail(df,20)
```

```
##           Date INJCJC INJCJC4
## 1281 18-Jul-14    279  300.75
## 1282 25-Jul-14    303  297.50
## 1283  1-Aug-14    290  293.75
## 1284  8-Aug-14    312  296.00
## 1285 15-Aug-14    299  301.00
## 1286 22-Aug-14    298  299.75
## 1287 29-Aug-14    304  303.25
## 1288  5-Sep-14    316  304.25
## 1289 12-Sep-14    281  299.75
## 1290 19-Sep-14    295  299.00
## 1291 26-Sep-14    288  295.00
## 1292  3-Oct-14    287  287.75
## 1293 10-Oct-14    266  284.00
## 1294 17-Oct-14    284  281.25
## 1295 24-Oct-14    288  281.25
## 1296 31-Oct-14    278  279.00
## 1297  7-Nov-14    293  285.75
## 1298 14-Nov-14    292  294.25
## 1299 21-Nov-14    314  294.25
## 1300 28-Nov-14    297  299.00
```

b. Convert the variables INJCJC into a time series object frequency=52, start=c(1990,1,1), end=c(2014,11,28). Examine the converted data series

```
INJCJC <- ts(df$INJCJC, frequency=52, start=c(1990,1,1), end=c(2014,11,28))
summary(INJCJC)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   259.0   327.0   355.0   373.3   408.0   665.0
```

```
str(INJCJC)
```

```
##  Time-Series [1:1259] from 1990 to 2014: 355 369 375 345 368 367 348 350 351 349 ...
```

c. Define a variable using the command INJCJC.time<-time(INJCJC)

```
INJCJC.time <- time(INJCJC)
str(INJCJC)
```

```
##  Time-Series [1:1259] from 1990 to 2014: 355 369 375 345 368 367 348 350 351 349 ...
```

d. Using the following command to examine the first 10 rows of the data. Change the parameter to examine different number of rows of data

head(cbind(INJCJC.time, INJCJC),10)

```
head(cbind(INJCJC.time, INJCJC), 20)
```

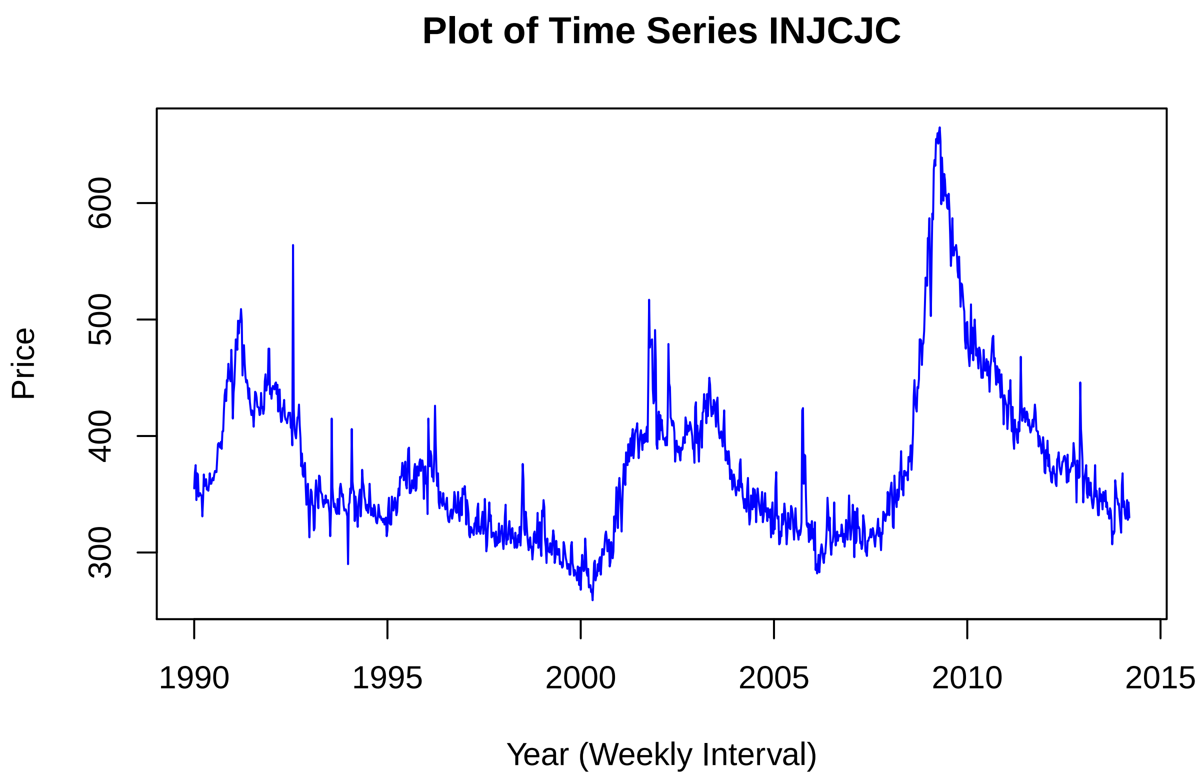
```
##       INJCJC.time INJCJC
##  [1,]    1990.000    355
##  [2,]    1990.019    369
##  [3,]    1990.038    375
##  [4,]    1990.058    345
##  [5,]    1990.077    368
##  [6,]    1990.096    367
##  [7,]    1990.115    348
##  [8,]    1990.135    350
##  [9,]    1990.154    351
## [10,]    1990.173    349
## [11,]    1990.192    349
## [12,]    1990.212    331
## [13,]    1990.231    346
## [14,]    1990.250    367
## [15,]    1990.269    357
## [16,]    1990.288    360
## [17,]    1990.308    363
## [18,]    1990.327    354
## [19,]    1990.346    355
## [20,]    1990.365    353
```

```r
tail(cbind(INJCJC.time, INJCJC), 20)
```

```
##         INJCJC.time INJCJC
## [1240,]    2013.827    362
## [1241,]    2013.846    355
## [1242,]    2013.865    347
## [1243,]    2013.885    346
## [1244,]    2013.904    341
## [1245,]    2013.923    342
## [1246,]    2013.942    332
## [1247,]    2013.962    324
## [1248,]    2013.981    317
## [1249,]    2014.000    358
## [1250,]    2014.019    368
## [1251,]    2014.038    339
## [1252,]    2014.058    344
## [1253,]    2014.077    333
## [1254,]    2014.096    329
## [1255,]    2014.115    334
## [1256,]    2014.135    345
## [1257,]    2014.154    328
## [1258,]    2014.173    343
## [1259,]    2014.192    330
```

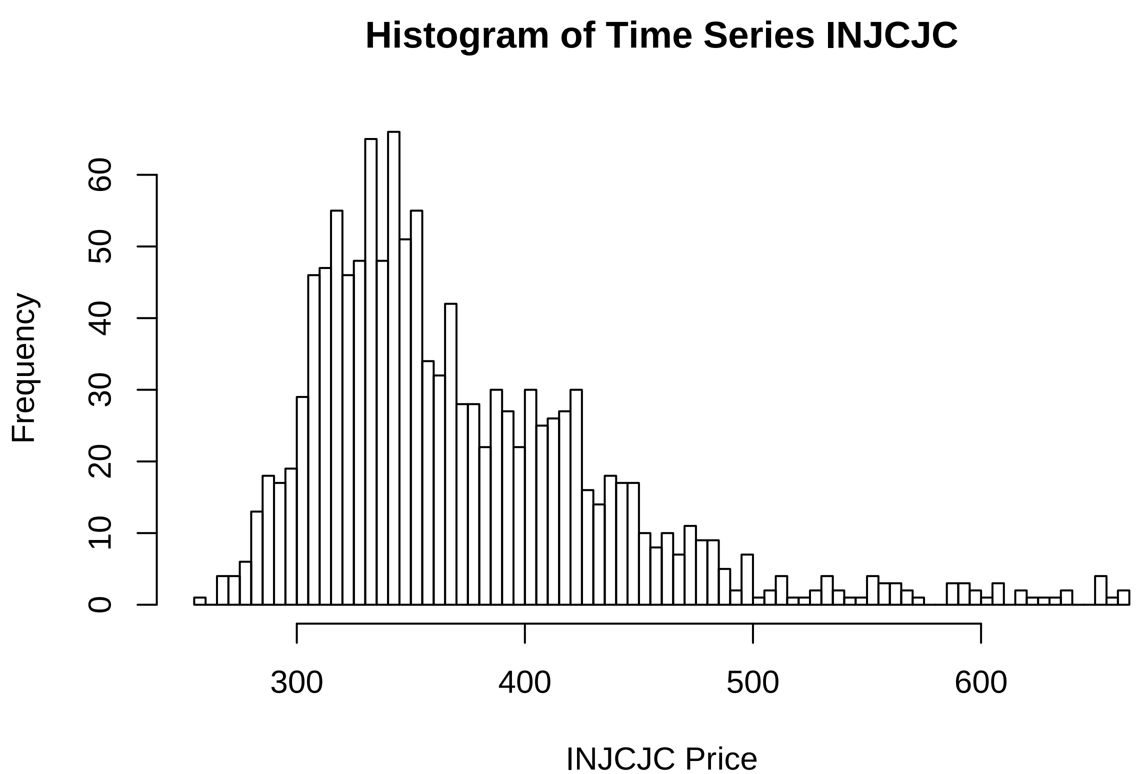
e1. Plot the time series plot of INJCJC. Remember that the graph must be well labelled.

```r
plot.ts(INJCJC, col='blue',
        xlab='Year (Weekly Interval)',
        ylab='Price',
        main='Plot of Time Series INJCJC')
```



e2. Plot the histogram of INJCJC. What is shown and not shown in a histogram? How do you decide the number of bins used?

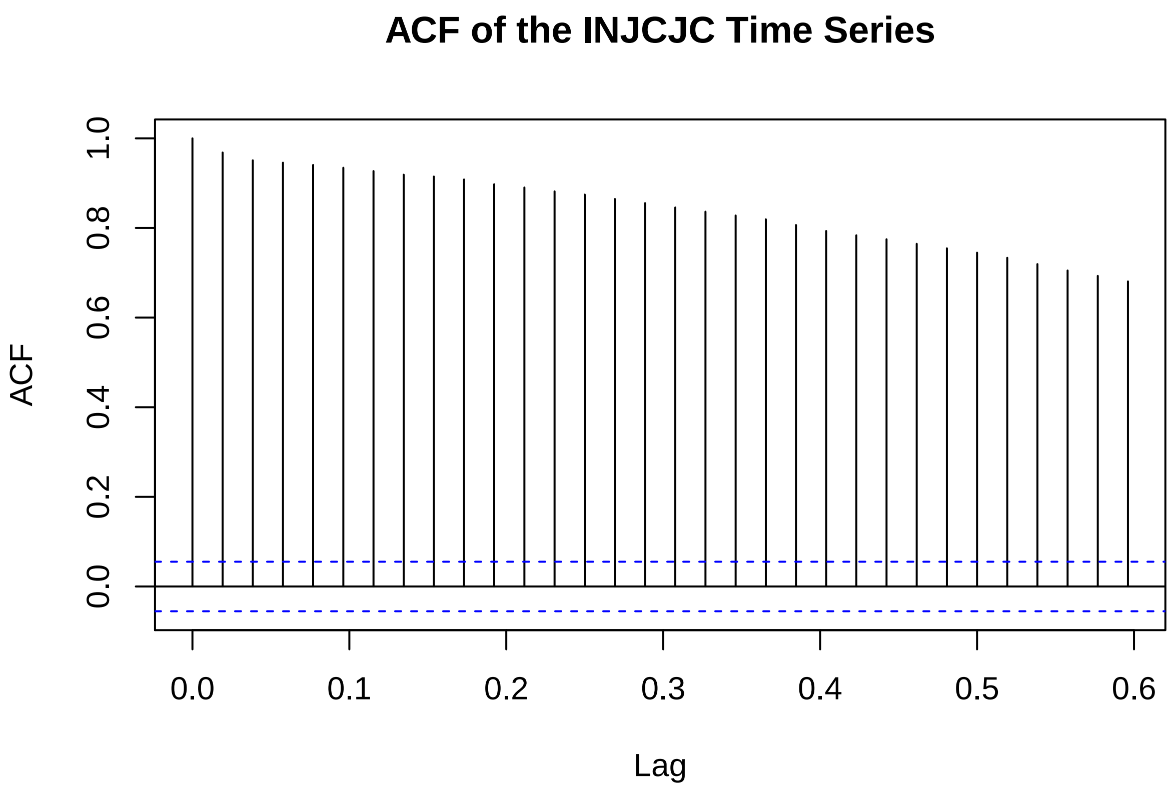
```
hist(INJCJC, breaks=80, main='Histogram of Time Series INJCJC',
     xlab='INJCJC Price')
```



The histogram shows the relative frequency of values in the series, but it does not show the time dependency of those values, and therefore their relationship to each other as a function of time. I decide the number of bins to use to show a reasonable amount of detail in the variation of the histogram while giving a good indication of the distribution of values.

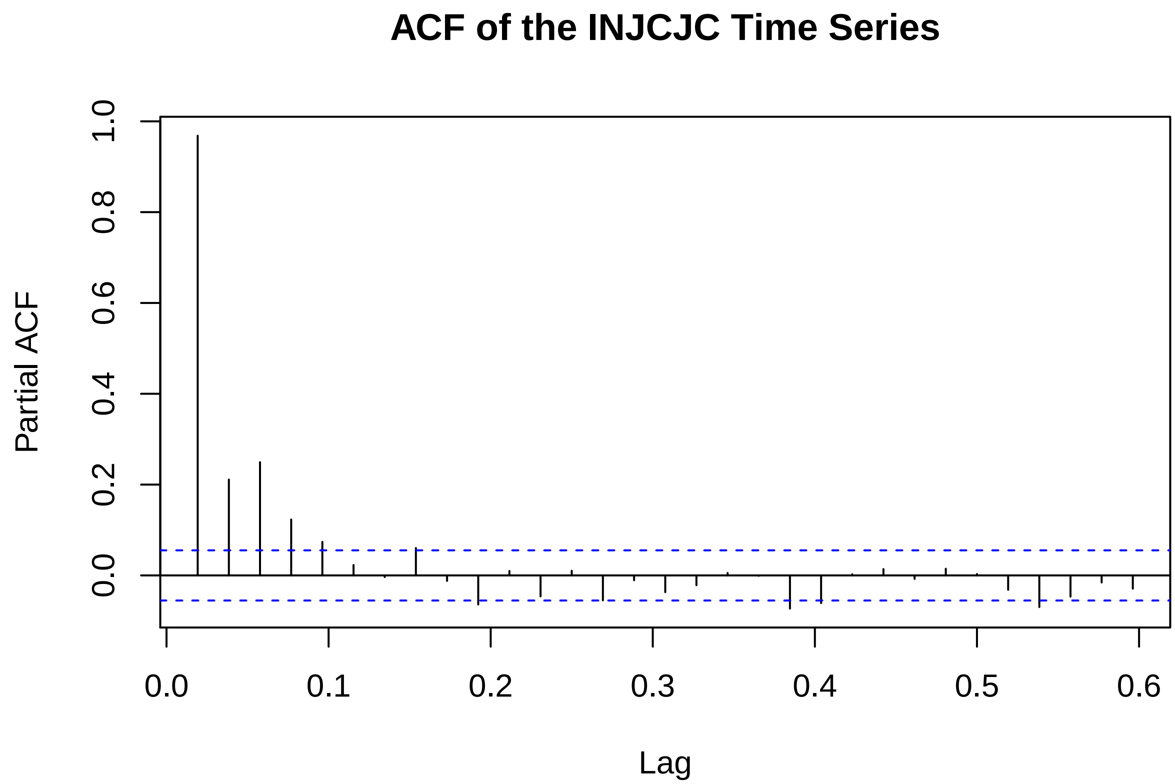
e3. Plot the autocorrelation graph of INJCJC series

```
acf(INJCJC, main='ACF of the INJCJC Time Series')
```



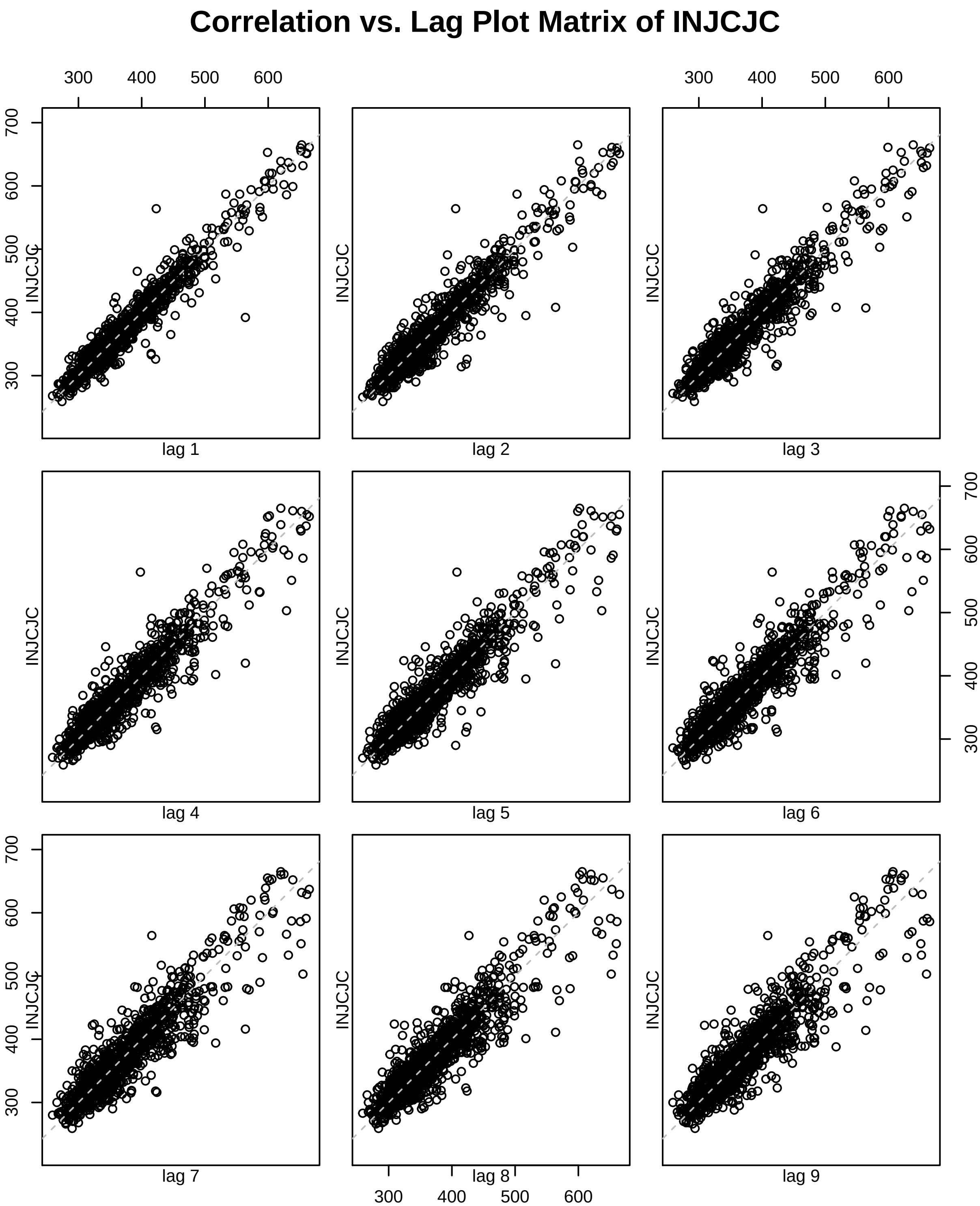
e4. Plot the partial autocorrelation graph of INJCJC series

```
pacf(INJCJC, main='ACF of the INJCJC Time Series')
```


ACF of the INJCJC Time Series
Partial ACF
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Lag

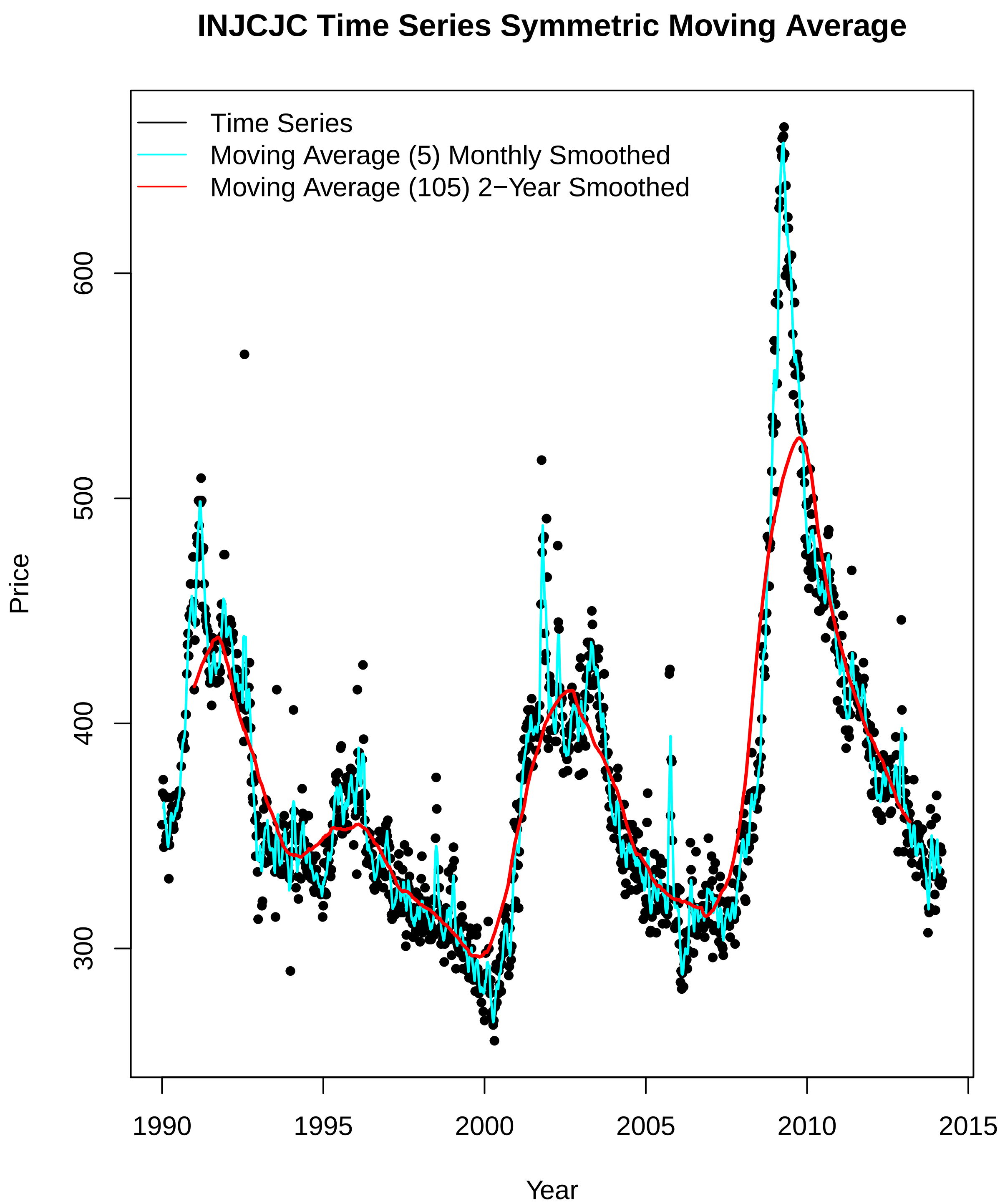
e5. Plot a 3x3 Scatterplot Matrix of correlation against lag values

```
lag.plot(INJCJC, 9, main='Correlation vs. Lag Plot Matrix of INJCJC')
```



f1. Generate two symmetric Moving Average Smoothers. Choose the number of moving average terms such that one of the smoothers is very smooth and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

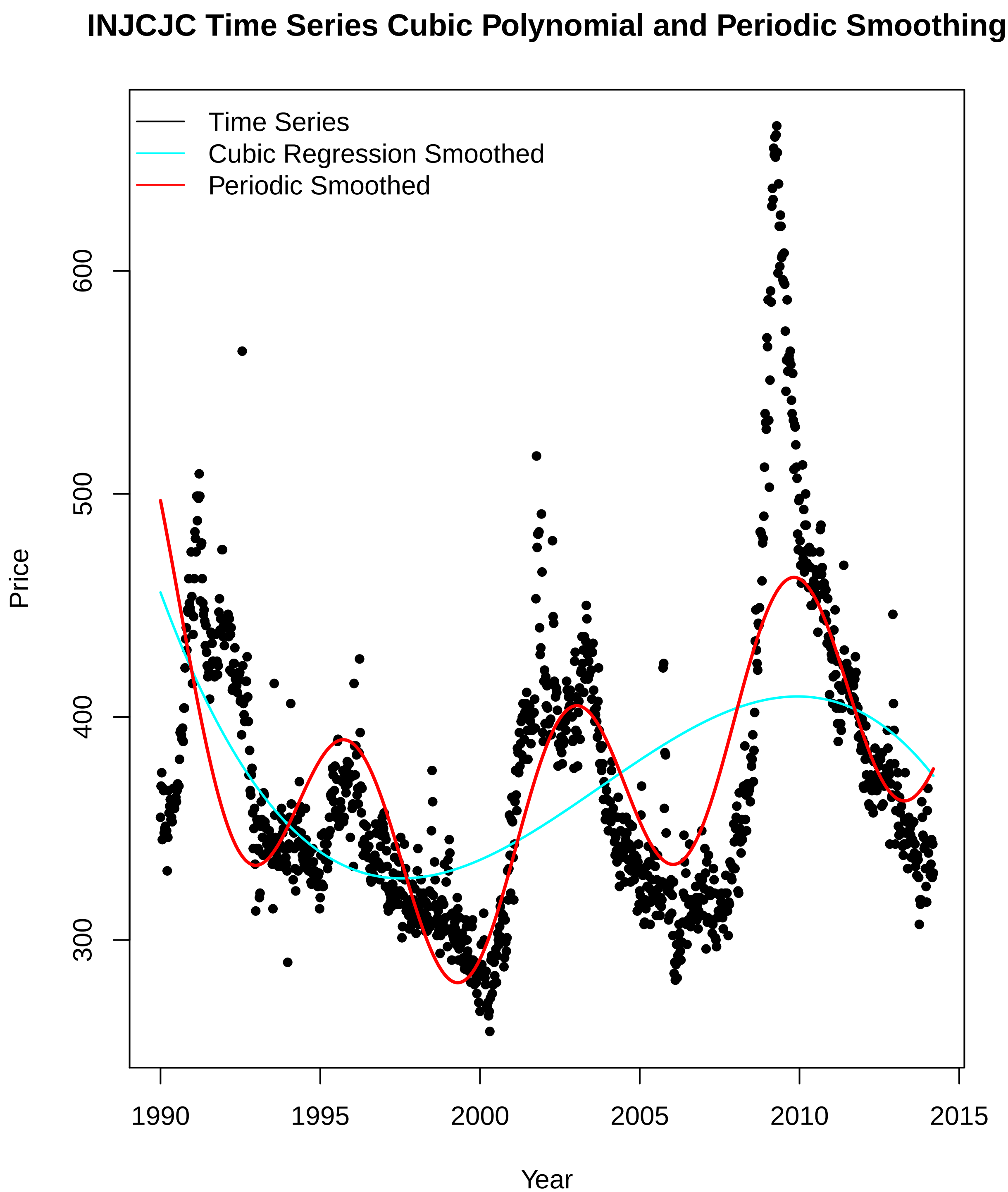
```
# The time series is weekly, we we can try a moving window
# average of 5 that would smooth across months
INJCJC.ma5 <- filter(INJCJC, sides=2, rep(1,5)/5)

# This symmetric MA filter would smooth across 2 years -
# the 1 year MA smoother is much less smooth
INJCJC.ma105 <- filter(INJCJC, sides=2, rep(1,105)/105)
plot(INJCJC, type='p', pch=20, col='black', xlab='Year',
     main='INJCJC Time Series Symmetric Moving Average', ylab='Price')
lines(INJCJC.ma5, col='cyan', lty=1, lwd=1.5)
lines(INJCJC.ma105, col='red', lty=1, lwd=2.0)
legend('topleft',legend=c('Time Series',
                          'Moving Average (5) Monthly Smoothed',
                          'Moving Average (105) 2-Year Smoothed'),
       lty=c(1,1,1), col=c('black','cyan','red'), bty='n', cex=1,
       merge=TRUE, bg=336)
```



INJCJC Time Series Symmetric Moving Average
Time Series
Moving Average (5) Monthly Smoothed
Moving Average (105) 2−Year Smoothed
Price
300
400
500
600
Year
1990
1995
2000
2005
2010
2015

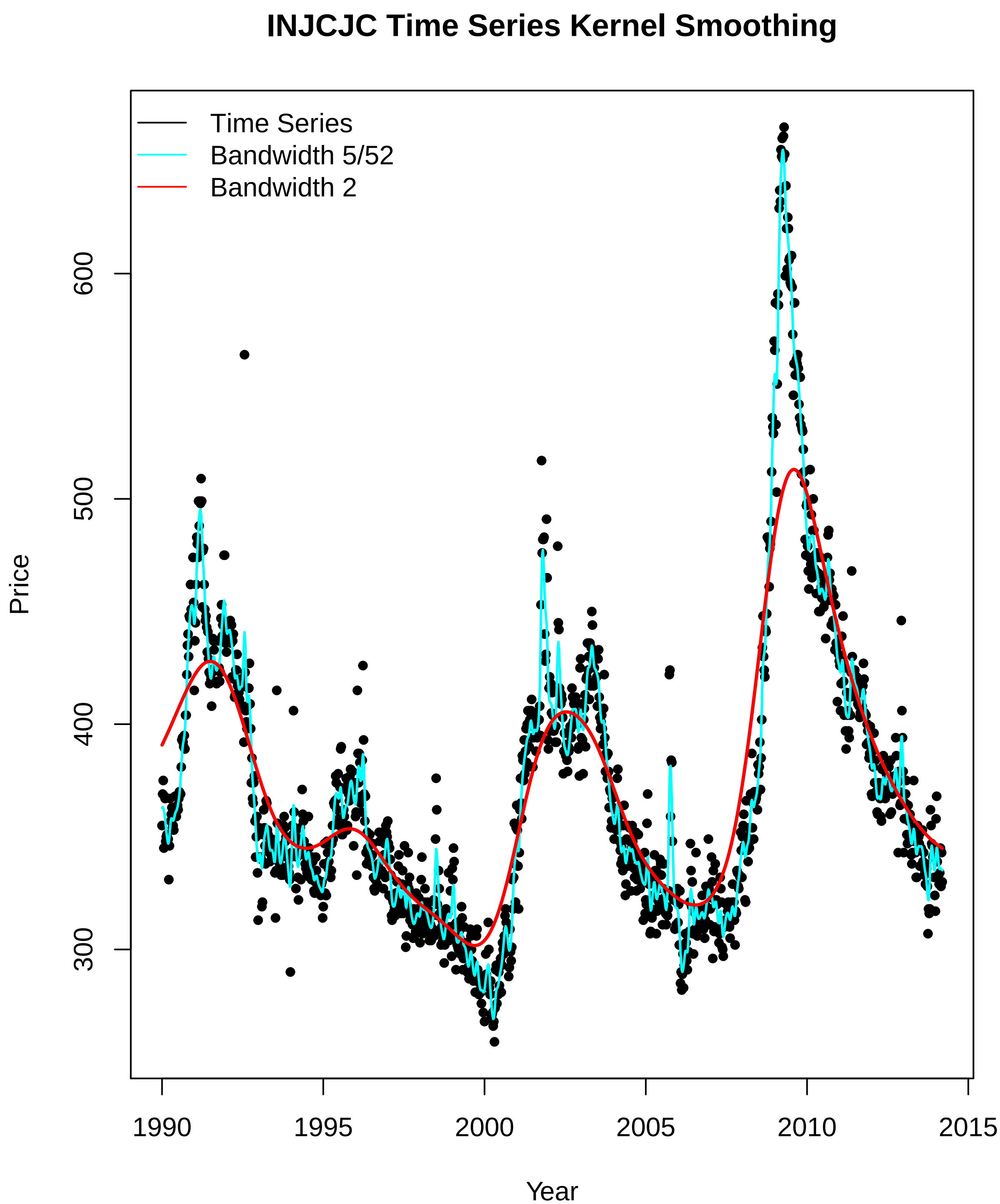
f2. Generate two regression smoothers, one being a cubic trend regression and the other being a periodic regression. Plot the smoothers and the original series in one graph.

```
wk <- INJCJC.time - mean(INJCJC.time)
wk2 <- wk^2
wk3 <- wk^3
cs <- cos(.29*pi*wk)
sn <- sin(.29*pi*wk)
INJCJC.cubic <- lm(INJCJC ~ wk + wk2 + wk3, na.action=NULL)
INJCJC.periodic <- lm(INJCJC ~ wk + wk2 + wk3 + cs + sn, na.action=NULL)
plot(INJCJC, type='p', pch=20, col='black', xlab='Year',
     main='INJCJC Time Series Cubic Polynomial and Periodic Smoothing', ylab='Price')
lines(fitted(INJCJC.cubic), lty=1, lwd=1.5, col='cyan')
lines(fitted(INJCJC.periodic), lty=1, lwd=2.0, col='red')
legend('topleft',legend=c('Time Series',
                          'Cubic Regression Smoothed',
                          'Periodic Smoothed'),
       lty=c(1,1,1), col=c('black','cyan','red'), bty='n', cex=1,
       merge=TRUE, bg=336)
```


INJCJC Time Series Cubic Polynomial and Periodic Smoothing
Time Series
Cubic Regression Smoothed
Periodic Smoothed
Price
300
400
500
600
1990
1995
2000
2005
2010
2015
Year

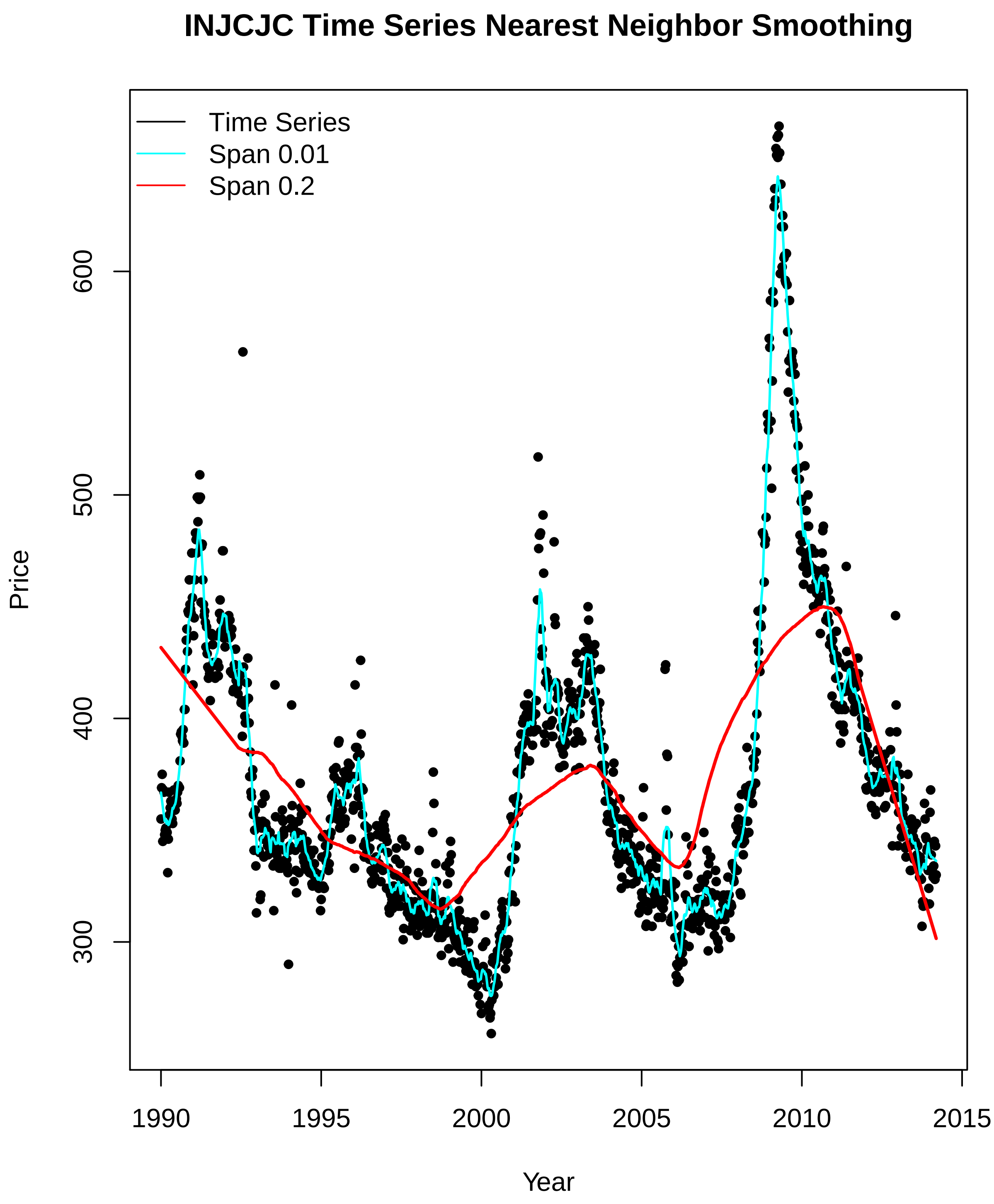
f3. Generate kernel smoothers. Choose the smoothing parametrs such that one of the smoothers is very smoother and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

```r
plot(INJCJC, type='p', pch=20, col='black', xlab='Year',
     main='INJCJC Time Series Kernel Smoothing', ylab='Price')
lines(ksmooth(time(INJCJC), INJCJC, "normal", bandwidth=5/52), lty=1, lwd=1.5, col='cyan')
lines(ksmooth(time(INJCJC), INJCJC, "normal", bandwidth=2), lty=1, lwd=2.0, col='red')
legend('topleft',legend=c('Time Series',
                          'Bandwidth 5/52',
                          'Bandwidth 2'),
       lty=c(1,1,1), col=c('black','cyan','red'), bty='n', cex=1,
       merge=TRUE, bg=336)
```


INJCJC Time Series Kernel Smoothing
Time Series
Bandwidth 5/52
Bandwidth 2
Price
300
400
500
600
Year
1990
1995
2000
2005
2010
2015

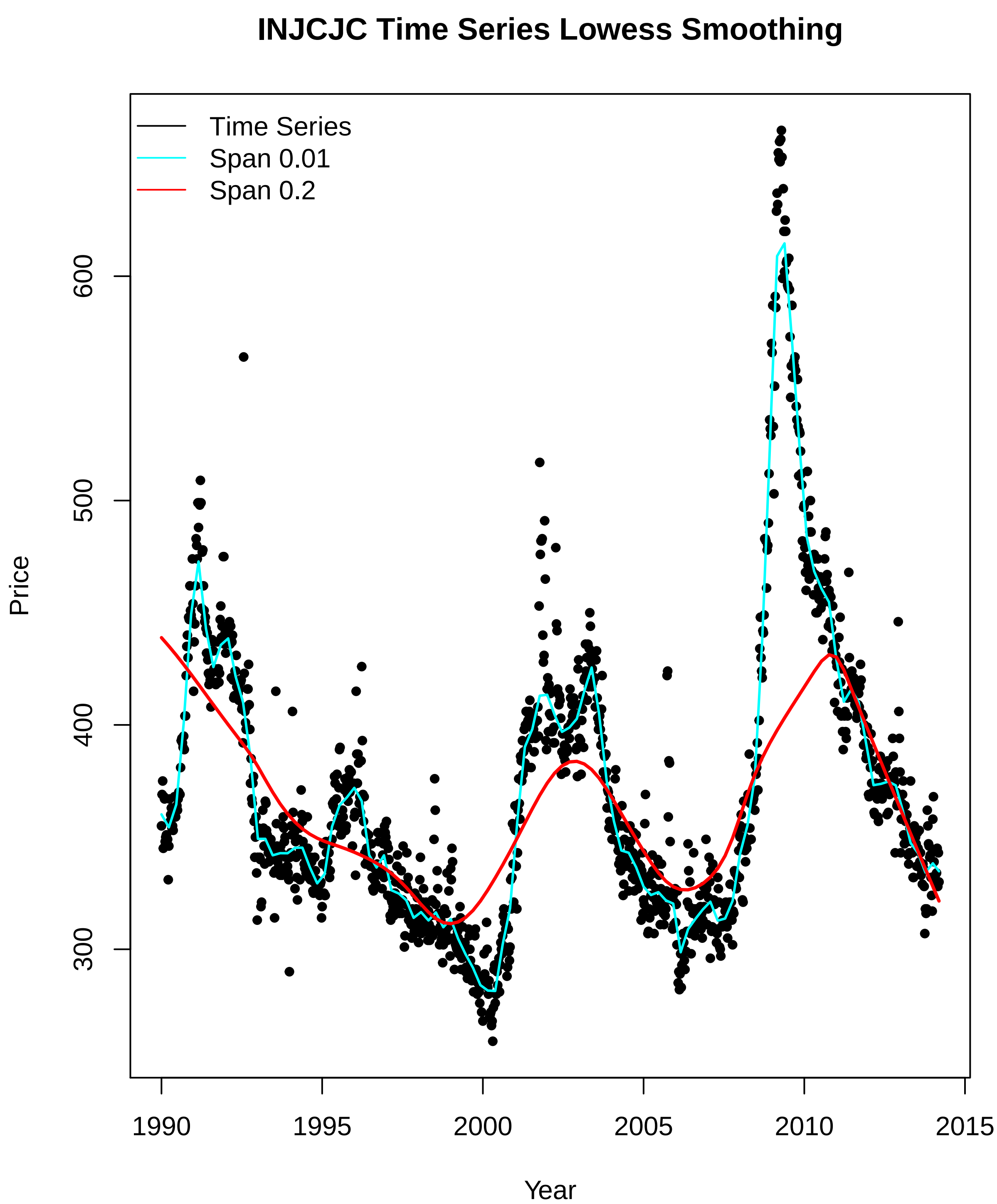
f4. Generate two nearest neighborhood smoothers. Choose the smoothing parametrs such that one of the smoothers is very smoother and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

```r
plot(INJCJC, type='p', pch=20, col='black', xlab='Year',
     main='INJCJC Time Series Nearest Neighbor Smoothing', ylab='Price')
lines(supsmu(time(INJCJC), INJCJC, span=0.01), lty=1, lwd=1.5, col='cyan')
lines(supsmu(time(INJCJC), INJCJC, span=0.2), lty=1, lwd=2.0, col='red')
legend('topleft',legend=c('Time Series',
                          'Span 0.01',
                          'Span 0.2'),
       lty=c(1,1,1), col=c('black','cyan','red'), bty='n', cex=1,
       merge=TRUE, bg=336)
```


INJCJC Time Series Nearest Neighbor Smoothing
Time Series
Span 0.01
Span 0.2
Price
Year
300
400
500
600
1990
1995
2000
2005
2010
2015

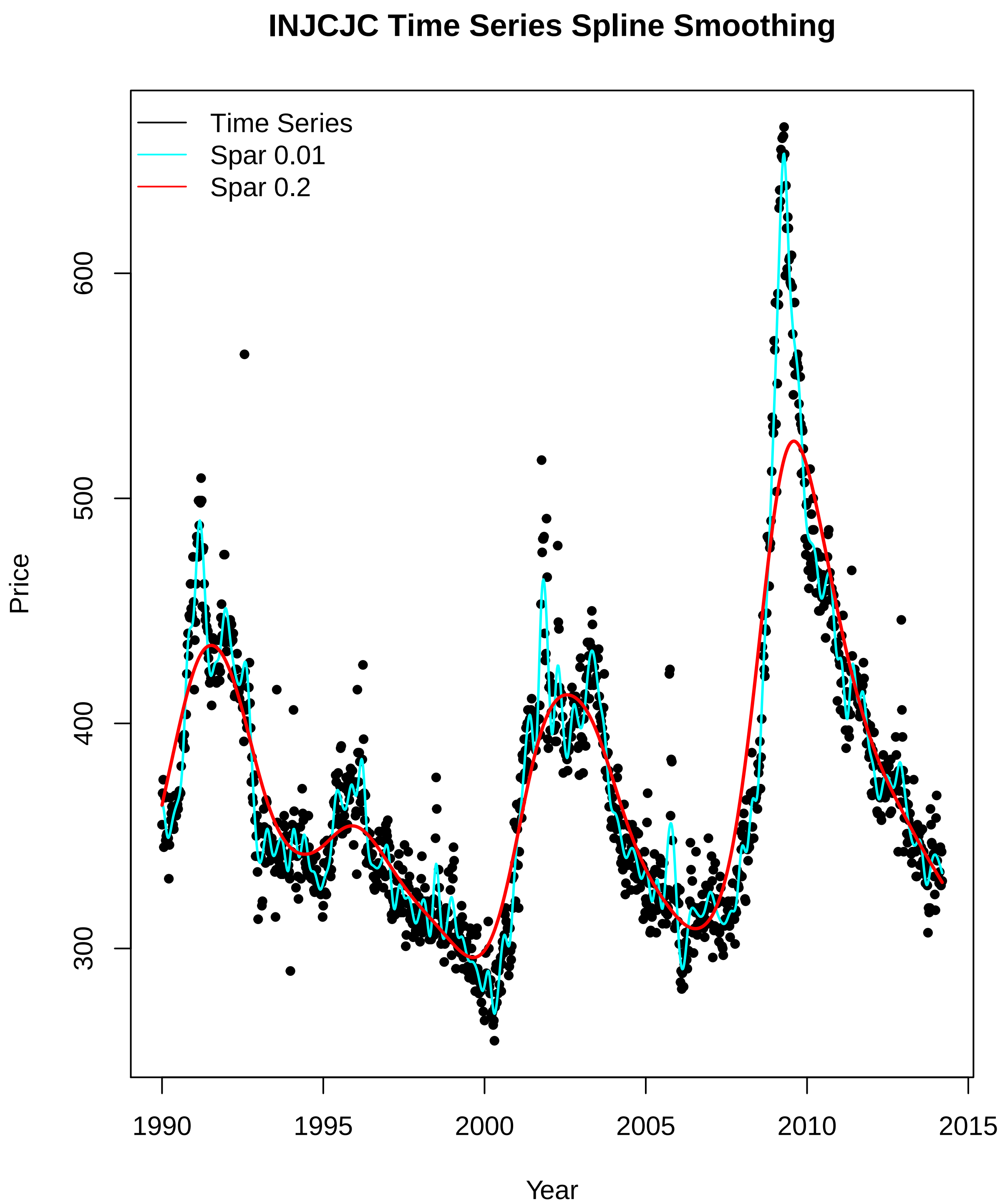
f5. Generate two LOWESS smoothers. Choose the smoothing parametrs such that one of the smoothers is very smoother and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

```r
plot(INJCJC, type='p', pch=20, col='black', xlab='Year',
     main='INJCJC Time Series Lowess Smoothing', ylab='Price')
lines(lowess(time(INJCJC), INJCJC, f=0.02), lty=1, lwd=1.5, col='cyan')
lines(lowess(time(INJCJC), INJCJC, f=0.25), lty=1, lwd=2.0, col='red')
legend('topleft',legend=c('Time Series',
                          'Span 0.01',
                          'Span 0.2'),
       lty=c(1,1,1), col=c('black','cyan','red'), bty='n', cex=1,
       merge=TRUE, bg=336)
```


INJCJC Time Series Lowess Smoothing
Time Series
Span 0.01
Span 0.2
Price
600
500
400
300
1990
1995
2000
2005
2010
2015
Year

f6. Generate two spline smoothers. Choose the smoothing parametrs such that one of the smoothers is very smoother and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

```
plot(INJCJC, type='p', pch=20, col='black', xlab='Year',
     main='INJCJC Time Series Spline Smoothing', ylab='Price')
lines(smooth.spline(time(INJCJC), INJCJC, spar=0.05), lty=1, lwd=1.5, col='cyan')
lines(smooth.spline(time(INJCJC), INJCJC, spar=0.75), lty=1, lwd=2.0, col='red')
legend('topleft',legend=c('Time Series',
                          'Spar 0.01',
                          'Spar 0.2'),
       lty=c(1,1,1), col=c('black','cyan','red'), bty='n', cex=1,
       merge=TRUE, bg=336)
```


INJCJC Time Series Spline Smoothing
Time Series
Spar 0.01
Spar 0.2
Price
300
400
500
600
Year
1990
1995
2000
2005
2010
2015