W271 - Applied Regression and Time Series Analysis - HW6

Subhashini R., Lei Yang, Ron Cordell

March 17, 2016

Exercise 1:

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

b). Define strict and weak statonarity

Exercise 2:

a). Generate a zero-drift random walk model using 500 simulation

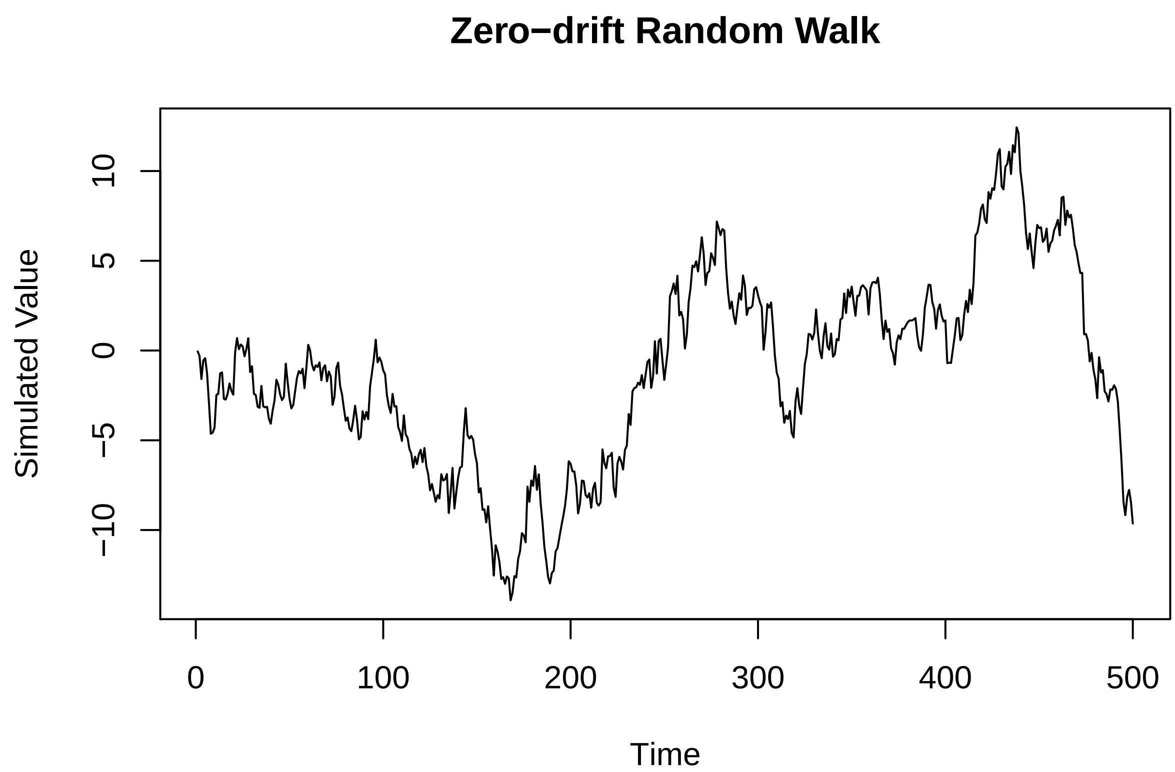
```r
# white noise with sample size 500
w <- rnorm(500,0,1)
# random walk
rw <- cumsum(w)
```

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

Statistic	Value
mean	-1.1638
standard deviation	5.3953
25% Q	-4.7680
50% Q	-0.9953
75% Q	2.3824
minimum	-13.9137
maximum	12.4350

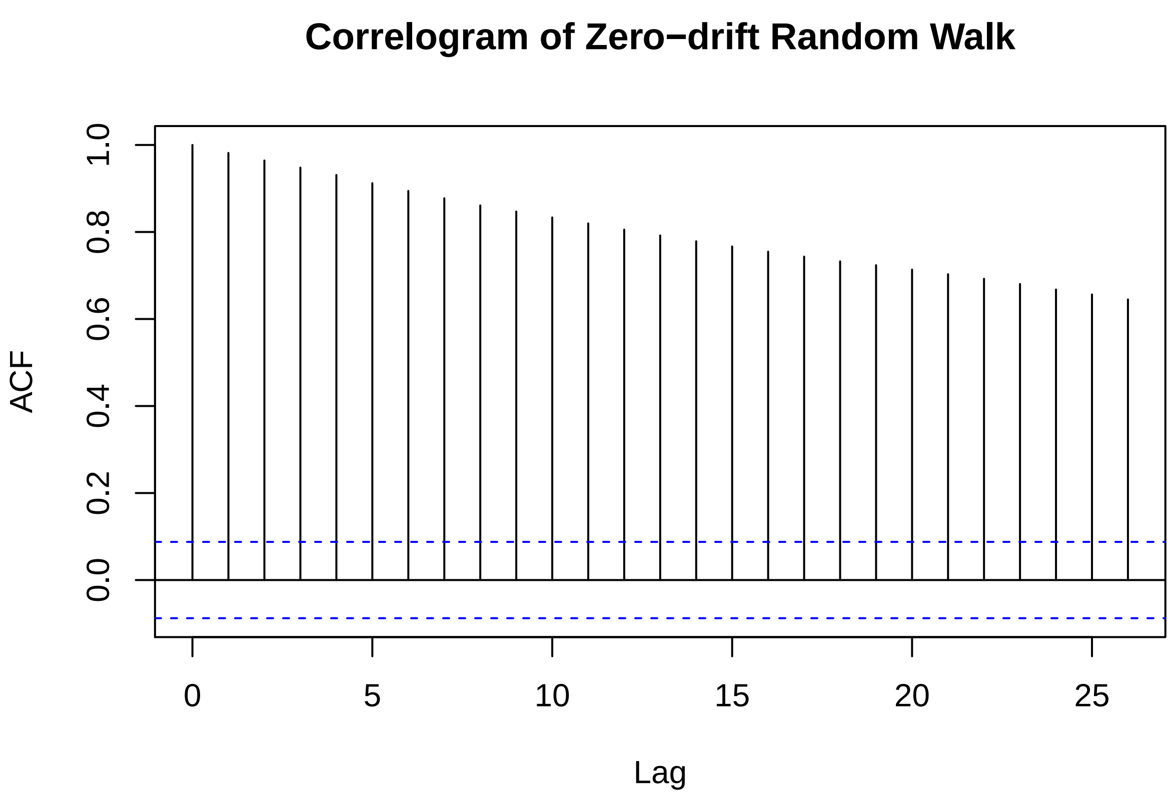
c). Plot the time-series plot of the simulated realizations

```r
plot.ts(rw, ylab='Simulated Value', main='Zero-drift Random Walk')
```



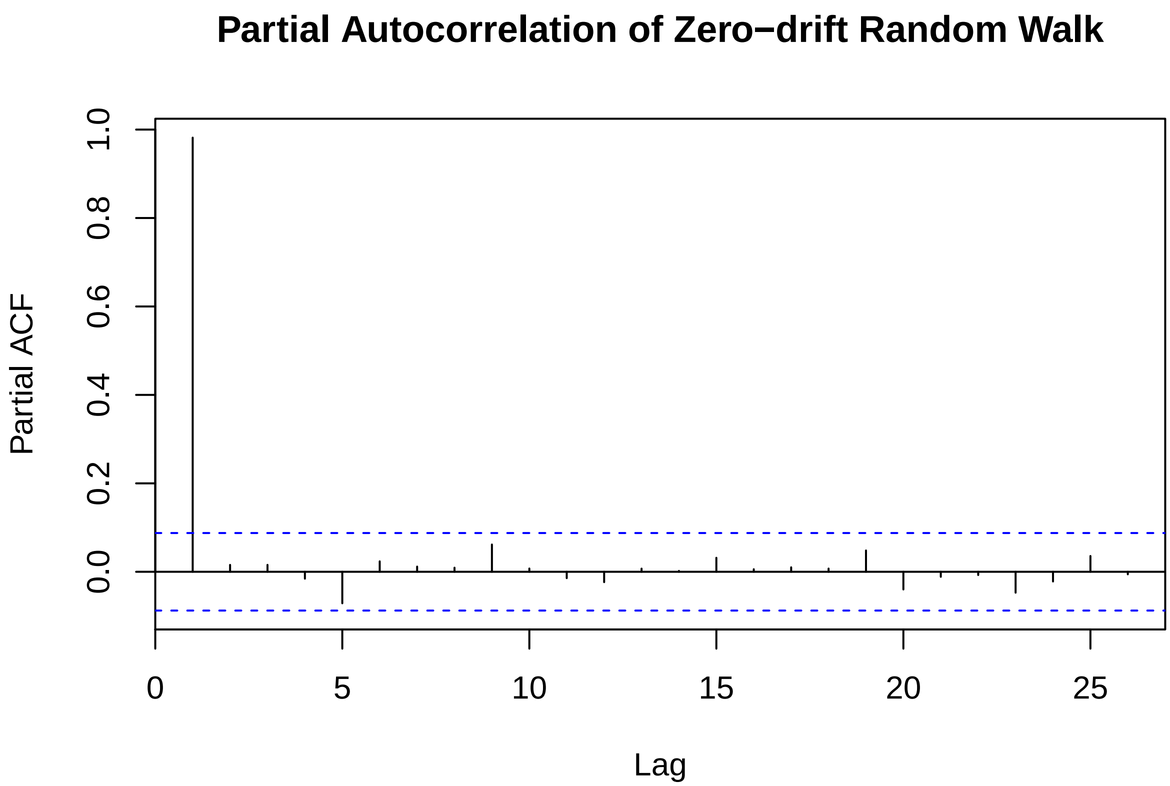
d). Plot the autocorrelation graph

```r
acf(rw, main="Correlogram of Zero-drift Random Walk")
```



e). Plot the partial autocorrelation graph

```r
pacf(rw, main="Partial Autocorrelation of Zero-drift Random Walk")
```



Exercise 3:

a). Generate arandom walk with drift model using 500 simulation, with the drift = 0.5

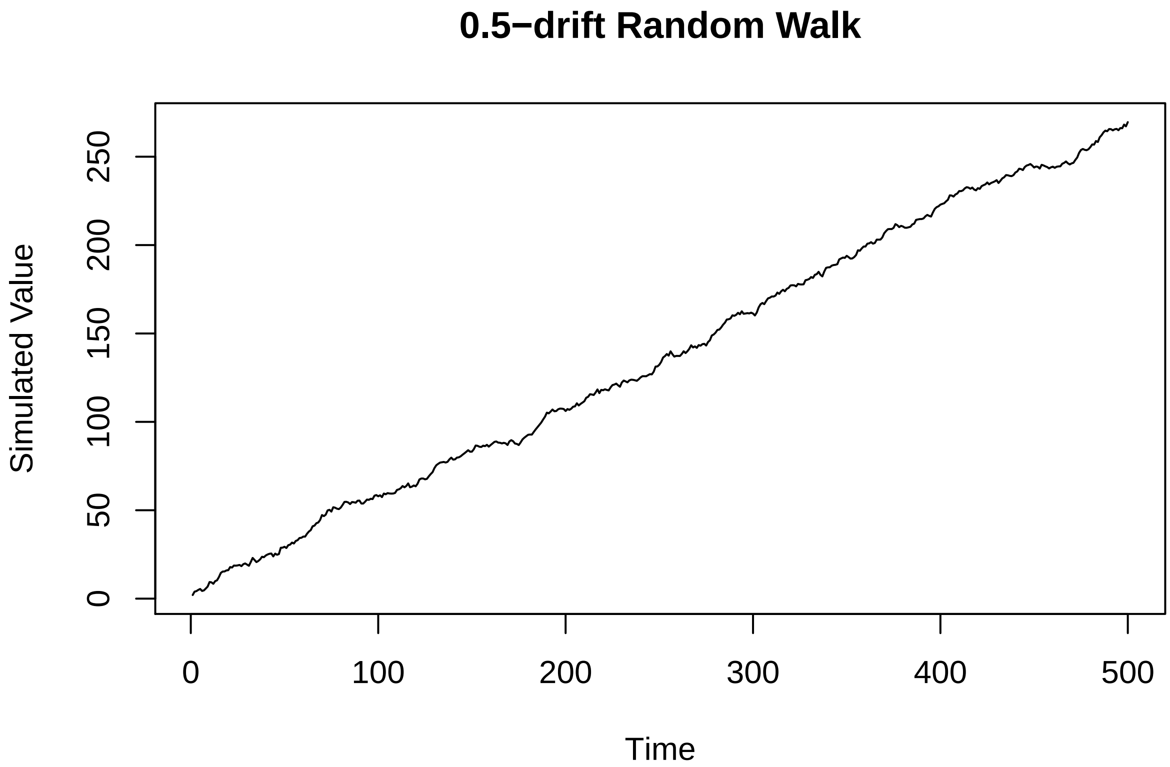
```r
w <- rnorm(500,0,1)+0.5
rw <- cumsum(w)
```

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

Statistic	Value
mean	137.0758
standard deviation	77.5828
25% Q	67.9529
50% Q	133.1274
75% Q	209.7542
minimum	2.0347
maximum	269.5588

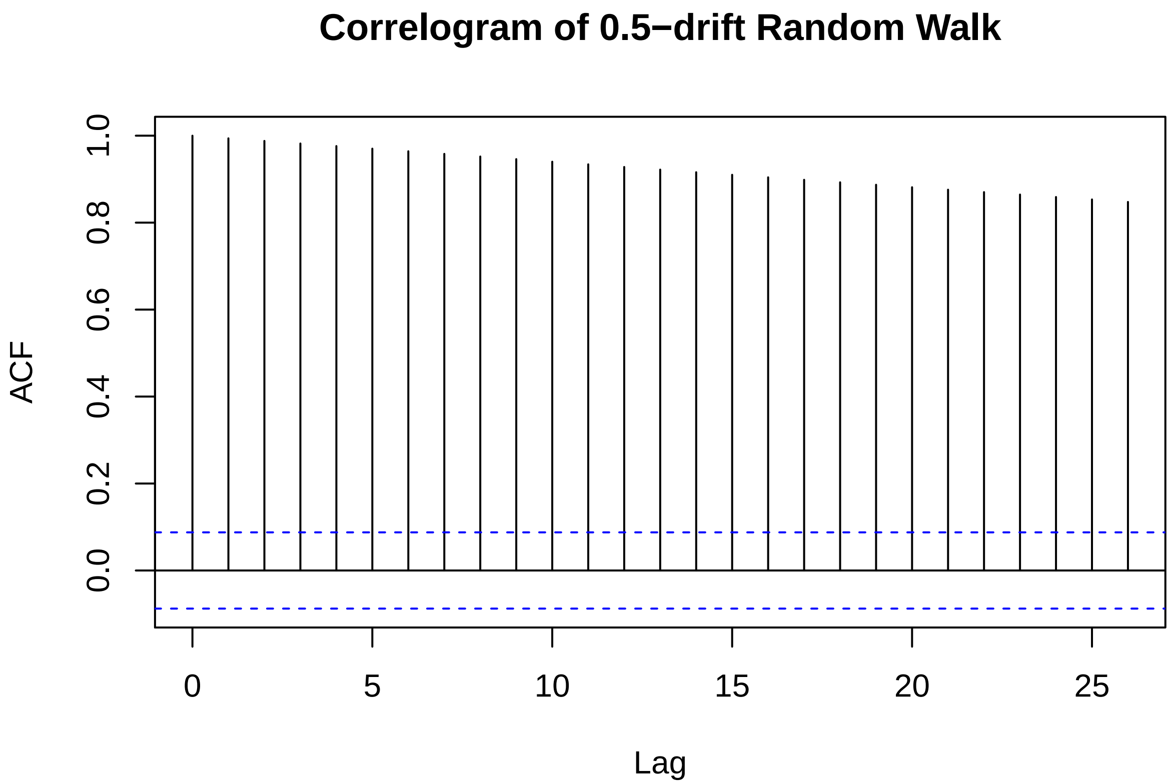
c). Plot the time-series plot of the simulated realizations

```r
plot.ts(rw, ylab='Simulated Value', main='0.5-drift Random Walk')
```



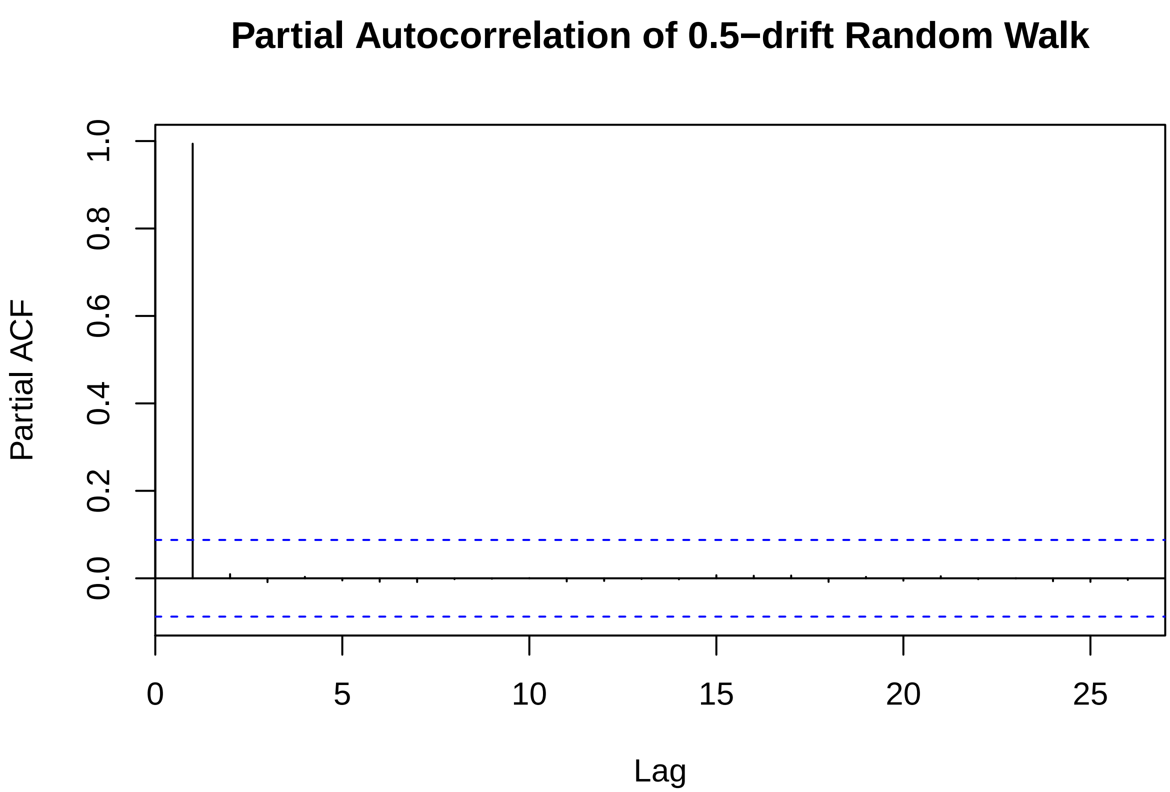
d). Plot the autocorrelation graph

```r
acf(rw, main="Correlogram of 0.5-drift Random Walk")
```



e). Plot the partial autocorrelation graph

```
pacf(rw, main="Partial Autocorrelation of 0.5-drift Random Walk")
```



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

```
setwd("~/Desktop/W271Data")
INJCJC <- read.csv('INJCJC.csv')
str(INJCJC)
```

```
## '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 ...
```

```
dim(INJCJC)
```

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

```
head(INJCJC)
```

```
##        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
```

```
tail(INJCJC)
```

```
##           Date INJCJC INJCJC4
## 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(data=INJCJC$INJCJC, frequency=52, start=c(1990,1,1), end=c(2014,11,28))
injcjc4 <- ts(data=INJCJC$INJCJC4, frequency=52, start=c(1990,1,1), end=c(2014,11,28))
#plot.ts(injcjc4, main='', ylab='INJCJC4')
```

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

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

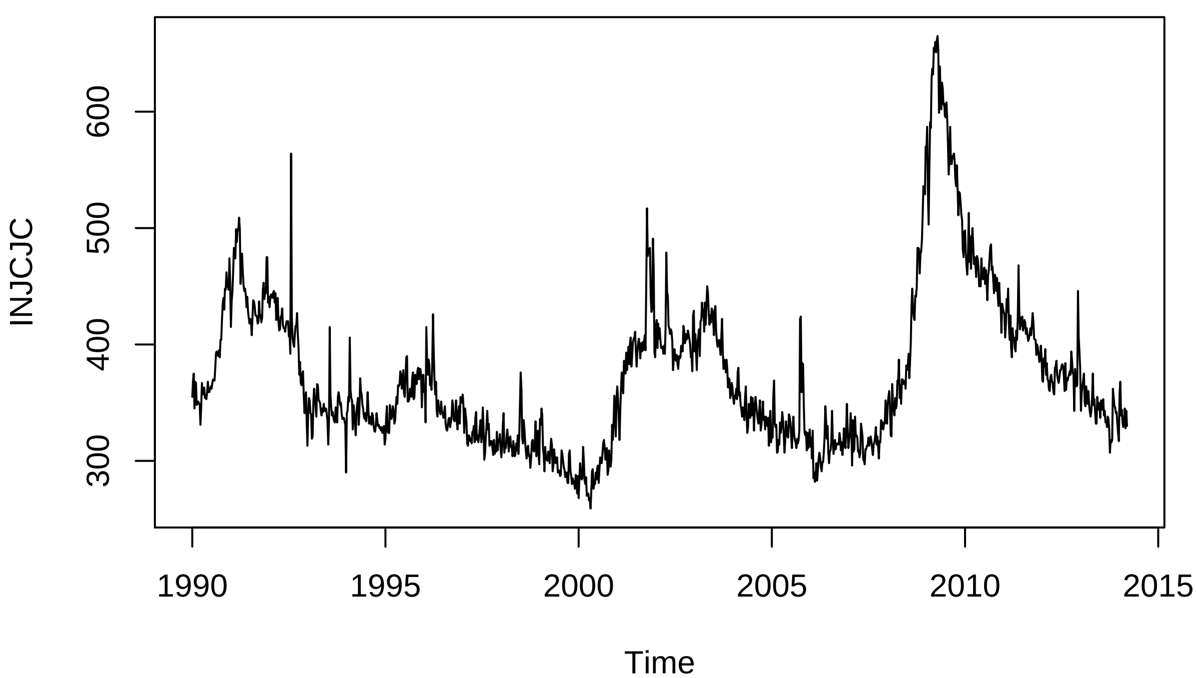
d). Using the following command to examine the first 10 rows of the data. Change the parameter to examine different number of rows of data

```r
head(cbind(INJCJC.time, injcjc),18)
```

```
##       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
```

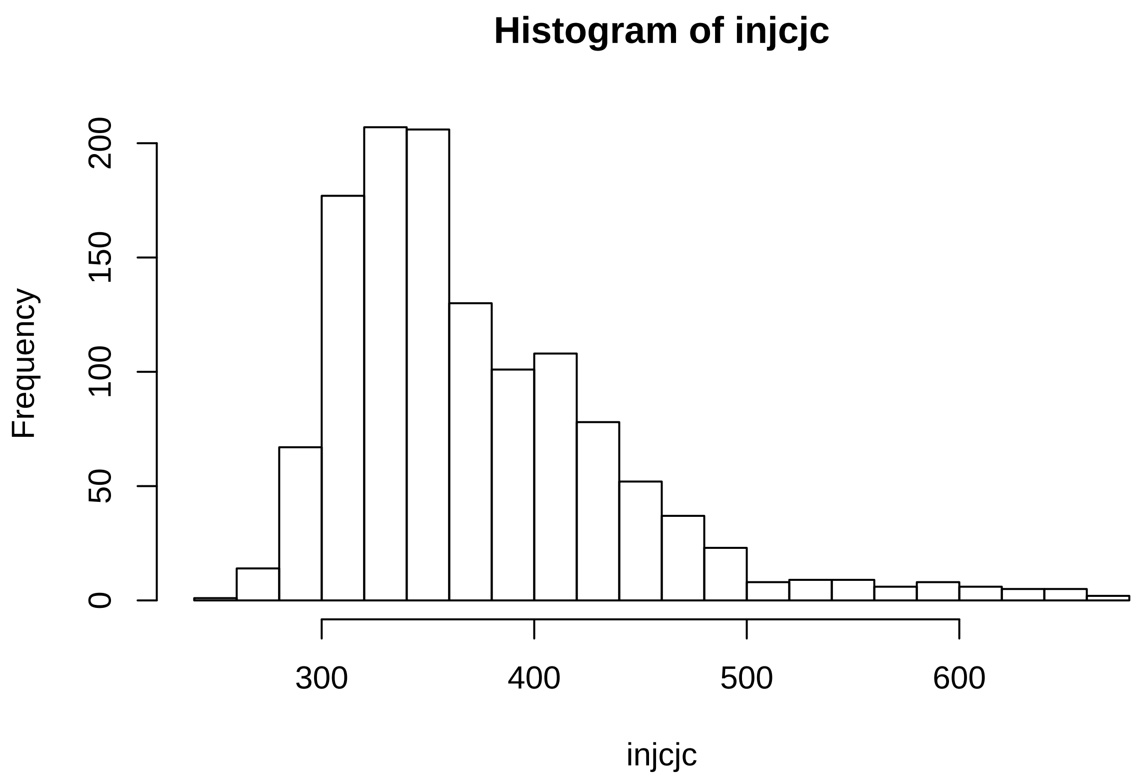
e1). Plot the time series plot of INJCJC. Remember that the graph must be well labelled.

```r
plot.ts(injcjc, main='', ylab='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?

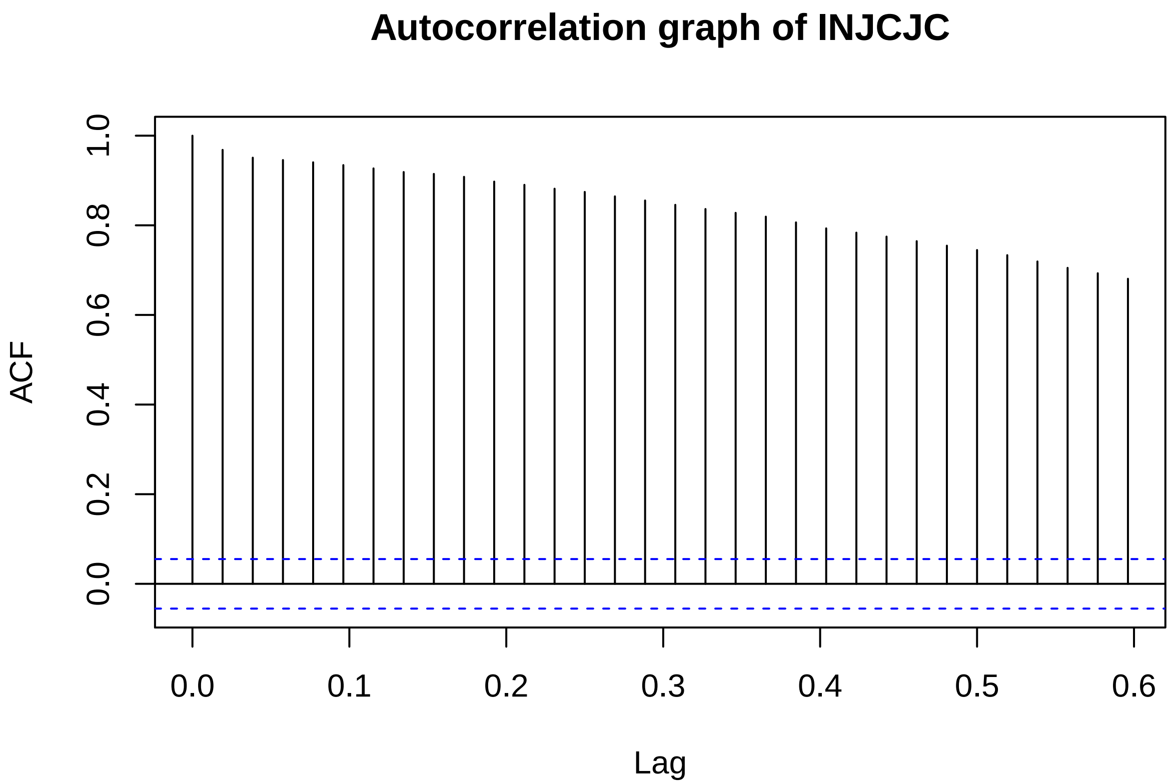
```r
hist(injcjc, breaks=20)
```



the histogram contains no temperal information in the series, it is impossible to know the dynamics over time from histogram

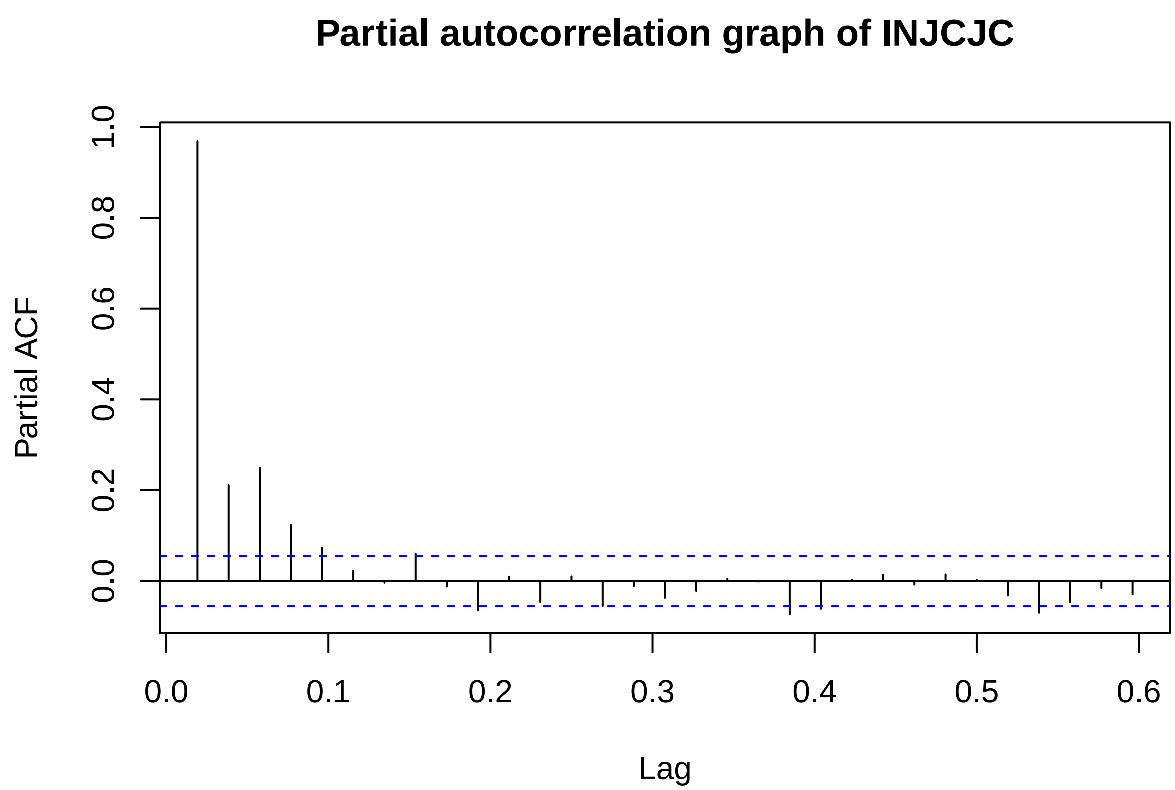
e3). Plot the autocorrelation graph of INJCJC series

```r
acf(injcjc, main='Autocorrelation graph of INJCJC')
```



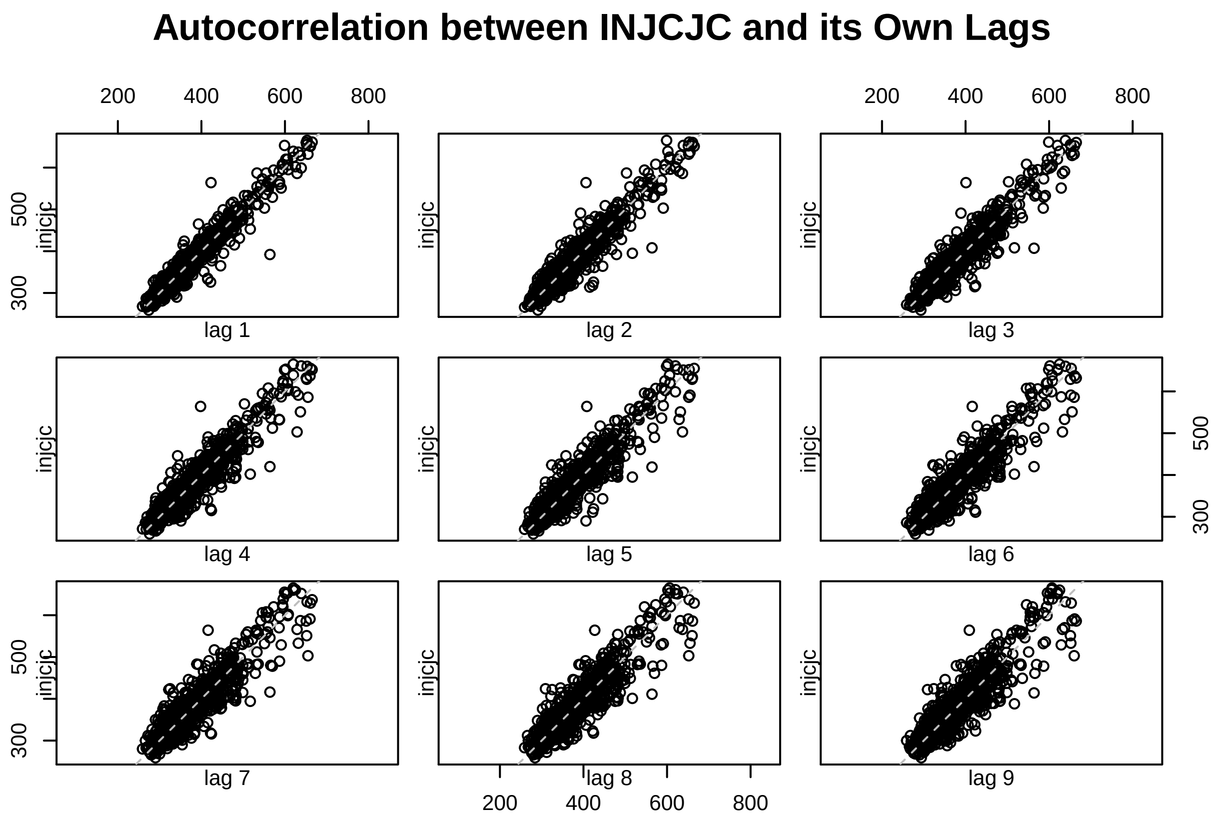
e4). Plot the partial autocorrelation graph of INJCJC series

```
pacf(injcjc, main='Partial autocorrelation graph of INJCJC')
```



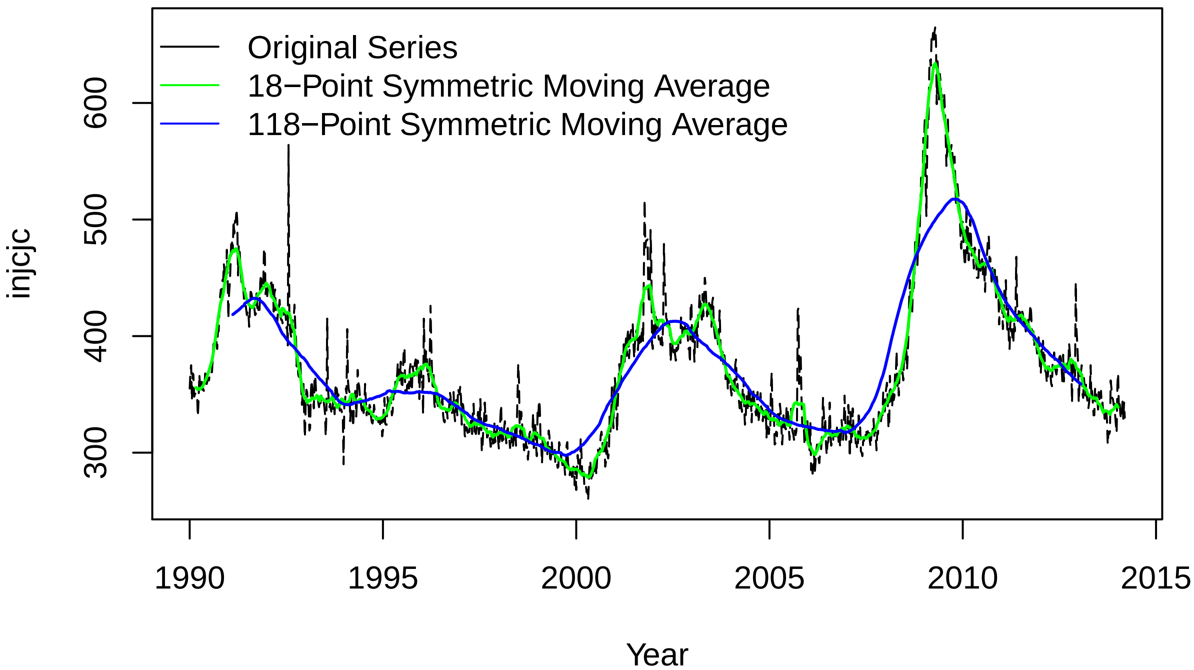
e5). Plot a 3x3 Scatterplot Matrix of correlation against lag values

```r
lag.plot(injcjc, lags=9, layout=c(3,3),
         diag=TRUE, disg.col="red",
         main="Autocorrelation between INJCJC and its Own Lags")
```



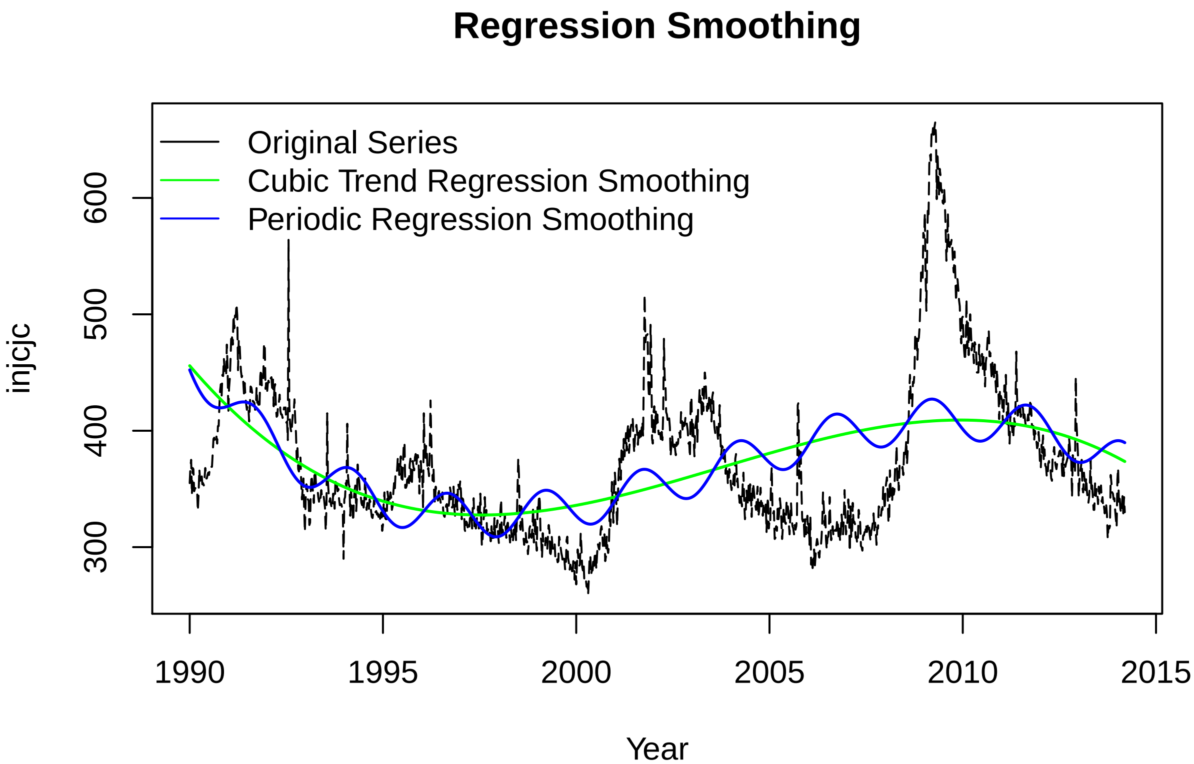
f1). Generate two symmetric Moving Average Smoothers. Choose the number of moving average terms such that one of the smoothers is very smoother and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

```r
ma3 = filter(injcjc, sides=2, rep(1,18)/18)
ma18 = filter(injcjc, sides=2, rep(1,118)/118)
plot(injcjc, main="", pch=4, lty=5, lwd=1, xlab="Year")
lines(ma3, lty=1, lwd=1.5, col="green")
lines(ma18, lty=1, lwd=1.5, col="blue")
leg.txt <- c("Original Series", "18-Point Symmetric Moving Average", "118-Point Symmetric Moving Averag
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","green","blue"),
       bty='n', cex=1, merge = TRUE, bg=336)
```



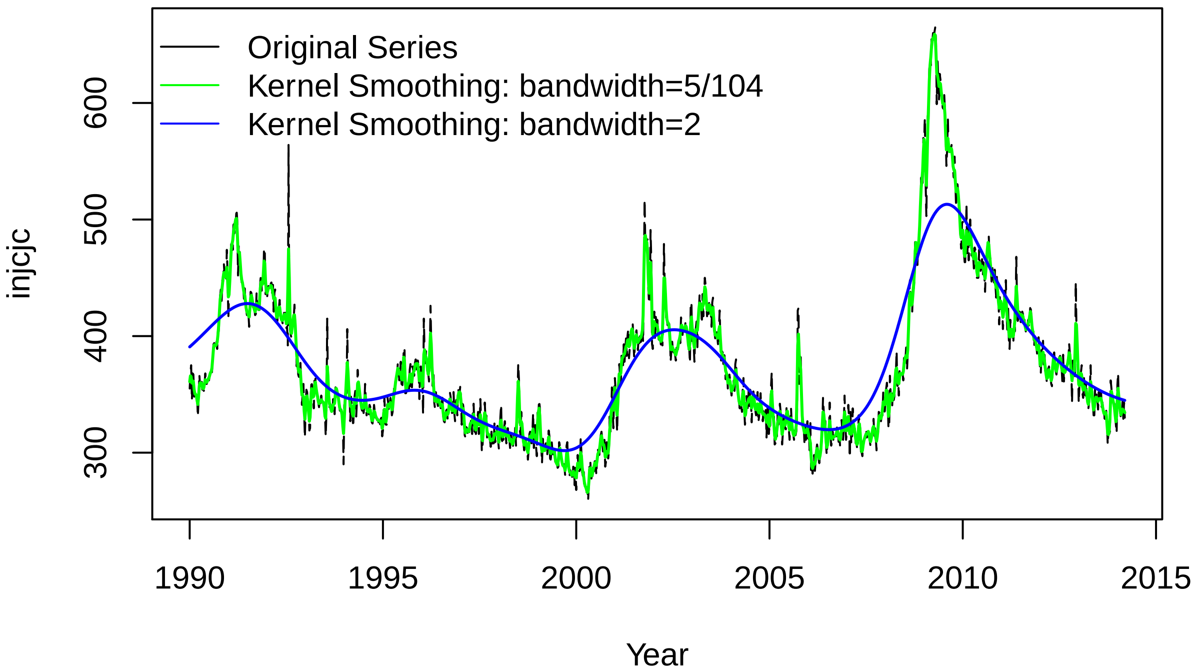
f2). Generate two regression smoothers, one being a cubic trend regression and the other being a periodic regression. Plot the smoothers and the original series in one graph.

```
wk = time(injcjc) - mean(time(injcjc))
wk2 = wk^2
wk3 = wk^3
cs = cos(0.8*pi*wk)
sn = sin(0.8*pi*wk)
reg1 = lm(injcjc~wk + wk2 + wk3, na.action=NULL)
reg2 = lm(injcjc~wk + wk2 + wk3 + cs + sn, na.action=NULL)
plot(injcjc, main="Regression Smoothing",
     pch=4, lty=5, lwd=1, xlab="Year")
lines(fitted(reg1), lty=1, lwd=1.5, col="green")
lines(fitted(reg2), lty=1, lwd=1.5, col="blue")
# Add Legend
leg.txt <- c("Original Series", "Cubic Trend Regression Smoothing", "Periodic Regression Smoothing")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","green","blue"),
       bty='n', cex=1, merge = TRUE, bg=336)
```



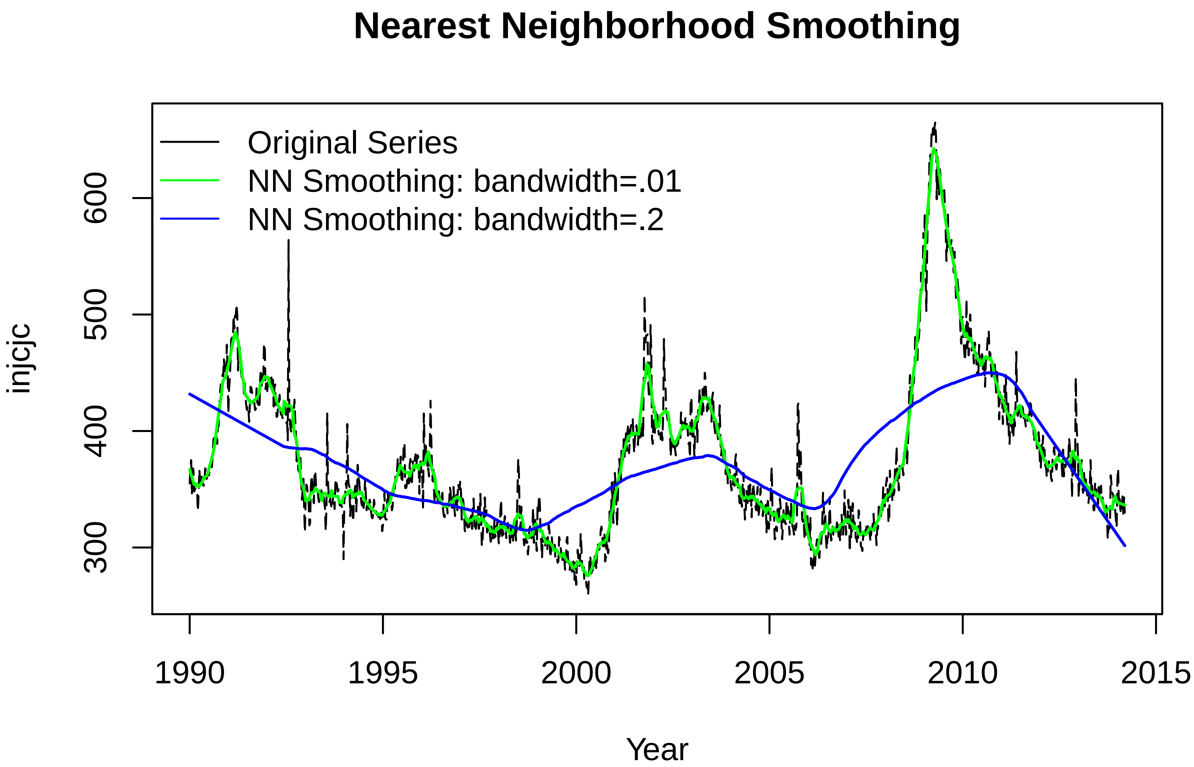
f3). Generate kernel smoothers. Choose the smoothing parametrs such that one of the smoothers is very smoother and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

```
plot(injcjc, pch=4, lty=5, lwd=1, xlab="Year")
lines(ksmooth(time(injcjc), injcjc, "normal", bandwidth=5/104),lty=1, lwd=1.5, col="green")
lines(ksmooth(time(injcjc), injcjc, "normal", bandwidth=2),lty=1, lwd=1.5, col="blue")
# Add Legend
leg.txt <- c("Original Series", "Kernel Smoothing: bandwidth=5/104", "Kernel Smoothing: bandwidth=2")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","green","blue"),
       bty='n', cex=1, merge = TRUE, bg=336)
```



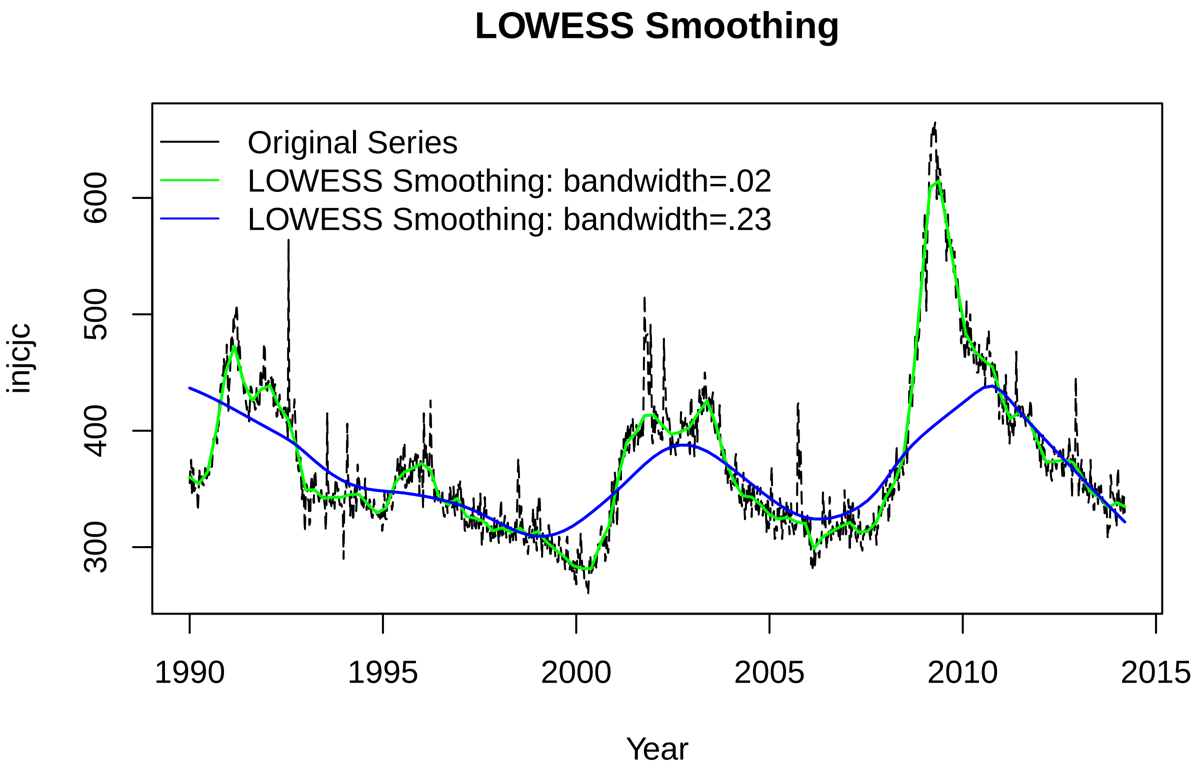
f4). Generate two nearest neighborhood smoothers. Choose the smoothing parametrs such that one of the smoothers is very smoother and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

```
plot(injcjc, main="Nearest Neighborhood Smoothing",
     pch=4, lty=5, lwd=1, xlab="Year")
lines(supsmu(time(injcjc), injcjc, span=.01),lty=1, lwd=1.5, col="green")
lines(supsmu(time(injcjc), injcjc, span=.2),lty=1, lwd=1.5, col="blue")
# Add Legend
leg.txt <- c("Original Series", "NN Smoothing: bandwidth=.01", "NN Smoothing: bandwidth=.2")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","green","blue"),
       bty='n', cex=1, merge = TRUE, bg=336)
```



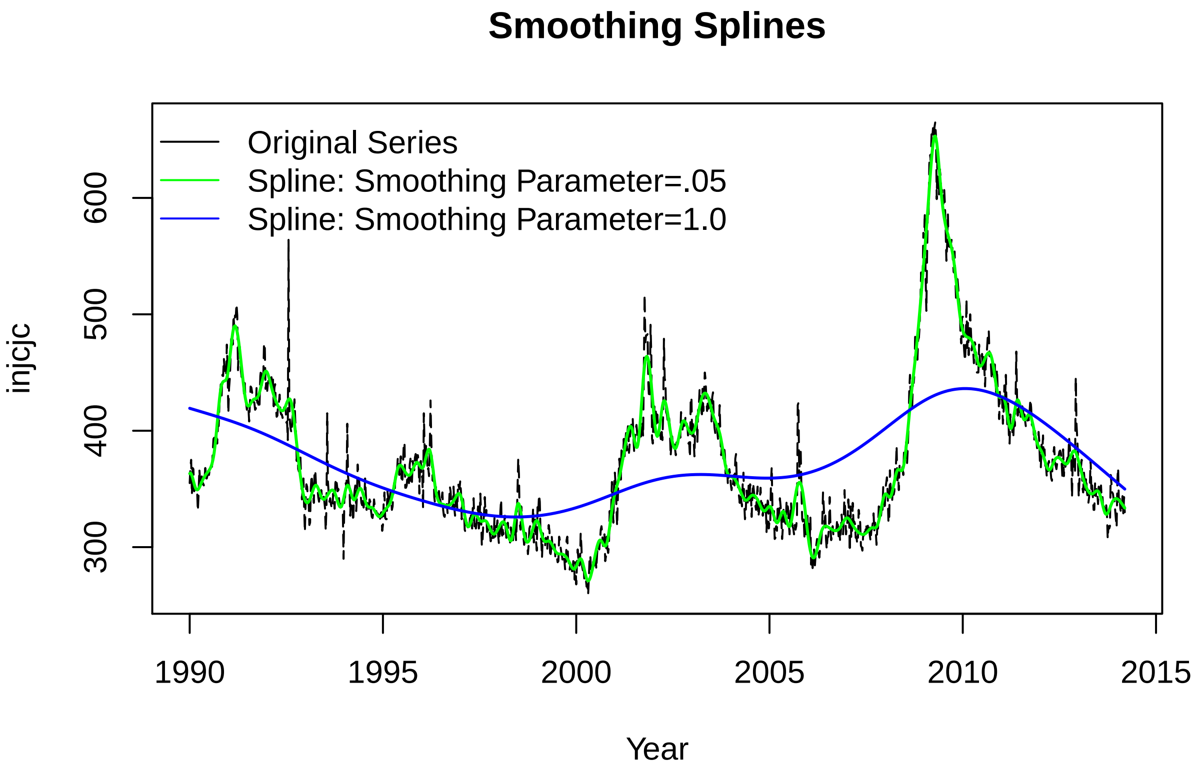
f5). Generate two LOWESS smoothers. Choose the smoothing parametrs such that one of the smoothers is very smoother and the other one can trace through the dynamics of the series. Plot the smoothers and the original series in one graph.

```
plot(injcjc, main="LOWESS Smoothing", pch=4, lty=5, lwd=1, xlab="Year")
lines(lowess(injcjc, f=.02),lty=1, lwd=1.5, col="green")
lines(lowess(injcjc, f=.23),lty=1, lwd=1.5, col="blue")
# Add Legend
leg.txt <- c("Original Series", "LOWESS Smoothing: bandwidth=.02", "LOWESS Smoothing: bandwidth=.23")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","green","blue"),
       bty='n', cex=1, merge = TRUE, bg=336)
```



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, main="Smoothing Splines", pch=4, lty=5, lwd=1, xlab="Year")
lines(smooth.spline(time(injcjc), injcjc, spar=0.05),lty=1, lwd=1.5, col="green")
lines(smooth.spline(time(injcjc), injcjc, spar=1.0),lty=1, lwd=1.5, col="blue")
# Add Legend
leg.txt <- c("Original Series", "Spline: Smoothing Parameter=.05", "Spline: Smoothing Parameter=1.0")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","green","blue"),
       bty='n', cex=1, merge = TRUE, bg=336)
```



Smoothing Splines
Original Series
Spline: Smoothing Parameter=.05
Spline: Smoothing Parameter=1.0
injcjc
300
400
500
600
1990
1995
2000
2005
2010
2015
Year