W271-2 – Spring 2016 – **HW 6**

Juanjo Carin, Kevin Davis, Ashley Levato, Minghu Song

March 16, 2016

Contents

Exercises

```
## Warning: package 'knitr' was built under R version 3.1.3
```

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 deinfed by the function:

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

This function has a time component so the mean could be different in different time periods. This is different from a mean in classical linear models where the mean is constant.

The variance functions for a time series analys is defined by the function:

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

Again this function is time dependant which means it varies with time unlike the variance in a classical linear model.

. . .

b. **Define strict and weak statonarity**

Statonarity indicates the parameter is consistent is accross time.

Strict stationary is when the joint distributions $F(x_{t_1}, ..., x_{t_n})$ and $F(x_{t_1+m}, ..., x_{t_n+m})$ are the same impling that the distribution is unchanged for any time shift.

Weak stationarity (also called second-order stationary) is when its mean and variancec stationary and its autocovariance $Cov(x_t, x_{t+k})$ depends on teh time placement k and can be written as $\gamma^{(k)}$. Once a distribution assumption is imposed the series can be completely characterized by the mean and covariance.

. . .

Exercise 2

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

```
# QUESTION 2 --------------------------------------------------------------
x<-w<-rnorm(500)
for (t in 2:500) x[t] <- x[t-1] + w[t]
```

...

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.**

```
mean(x)
```

```
## [1] -13.07839
```

```
sd(x)
```

```
## [1] 9.252352
```

```
min(x)
```

```
## [1] -30.93562
```

```
max(x)
```

```
## [1] 6.59579
```

```
quantile(x, probs=c(.25, .5, .75))
```

```
##        25%        50%        75%
## -21.280157 -12.362306  -6.182762
```

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

```
plot(x, type= "l")
```

...

d. **Plot the autocorrelation graph.**

```
acf(x, main="ACF Zero-drift random walk")
```

...

e. **Plot the partial autocorrelation graph.**

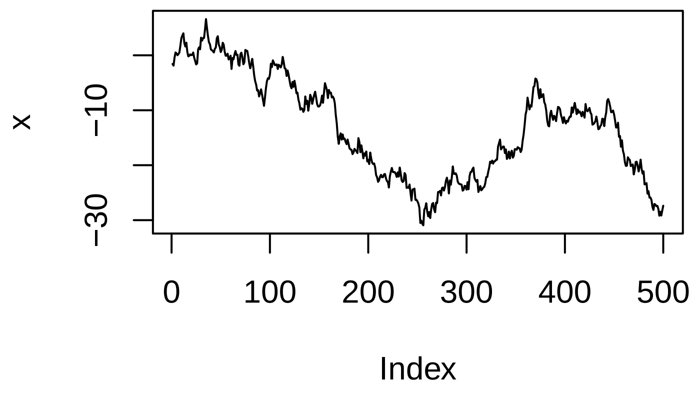


Figure 1:

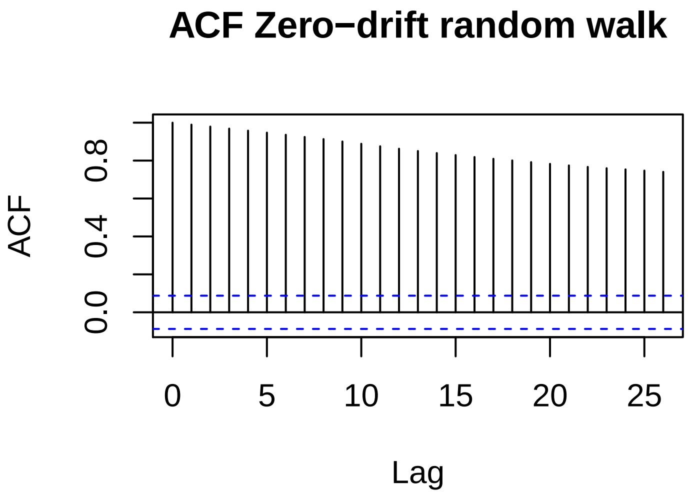
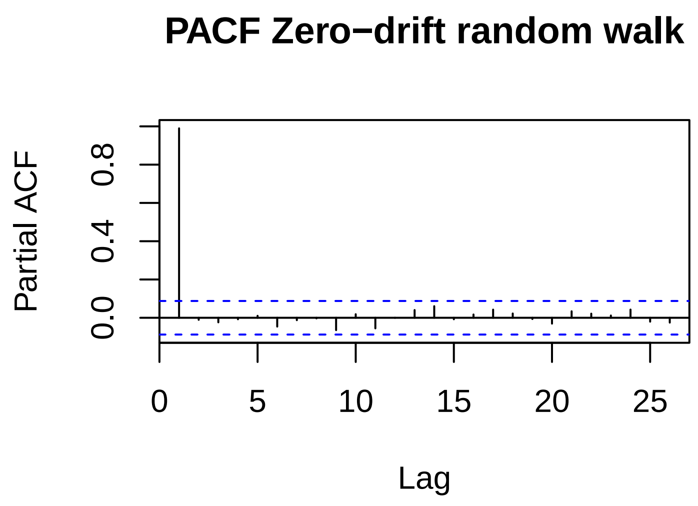


Figure 2:

```r
pacf(x, main ="PACF Zero-drift random walk")
```



. . .

Exercise 3

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

```
x1<-w1<-rnorm(500)
d<-.5
for (t in 2:500) x1[t] <- x1[t-1] + w1[t] + d
```

...

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.**

```
mean(x1)
```

```
## [1] 161.6304
```

```
sd(x1)
```

```
## [1] 87.48769
```

```
min(x1)
```

```
## [1] 0.1342579
```

```
max(x1)
```

```
## [1] 299.1543
```

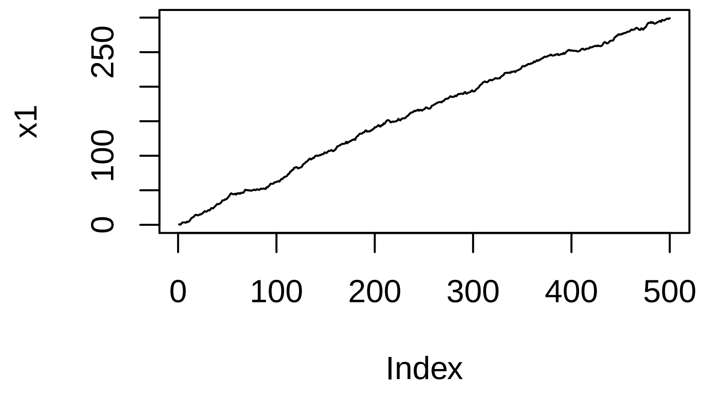
```
quantile(x1, probs=c(.25, .5, .75))
```

```
##       25%       50%       75%
##  84.18863 167.63741 243.74097
```

...

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

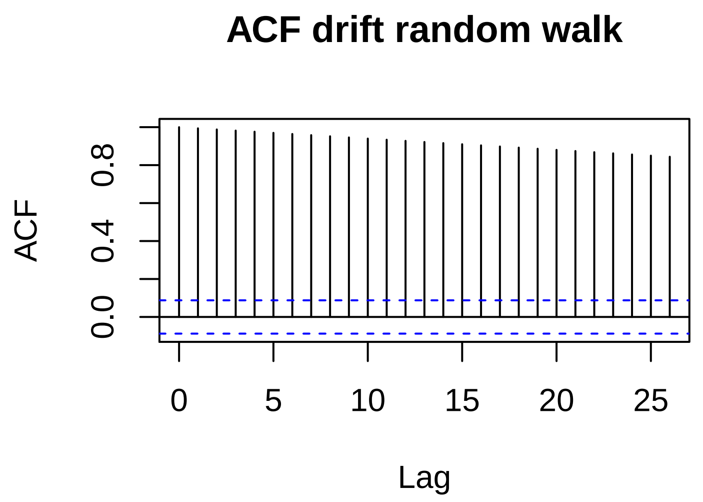
```
plot(x1, type= "l")
```



...

d. **Plot the autocorrelation graph.**

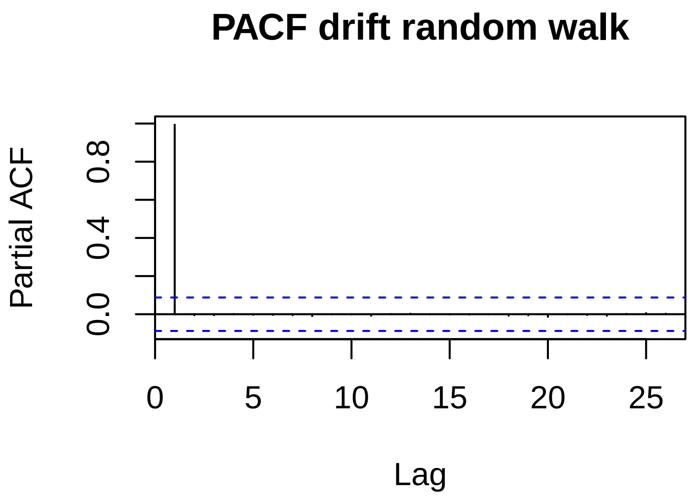
```
acf(x1, main="ACF drift random walk")
```



...

e. **Plot the partial autocorrelation graph.**

```
pacf(x1, main ="PACF drift random walk")
```


PACF drift random walk
Partial ACF
0.0
0.4
0.8
0
5
10
15
20
25
Lag

. . .

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.**

```
data<-read.csv("INJCJC.csv")
str(data)
```

```
## '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(data)
```

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

```
head(data)
```

```
##        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(data)
```

```
##           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.**

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

```
## [1] 1259
```

```
head(data_ts)
```

```
## [1] 355 369 375 345 368 367
```

```
tail(data_ts)
```

```
## [1] 329 334 345 328 343 330
```

...

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

```
data_ts.time<-time(data_ts)
```

...

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(data_ts.time, data_ts), 10)
```

```
##       data_ts.time data_ts
##  [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
```

```
head(cbind(data_ts.time, data_ts), 13)
```

```
##       data_ts.time data_ts
##  [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
```

```r
head(cbind(data_ts.time, data_ts), 6)
```

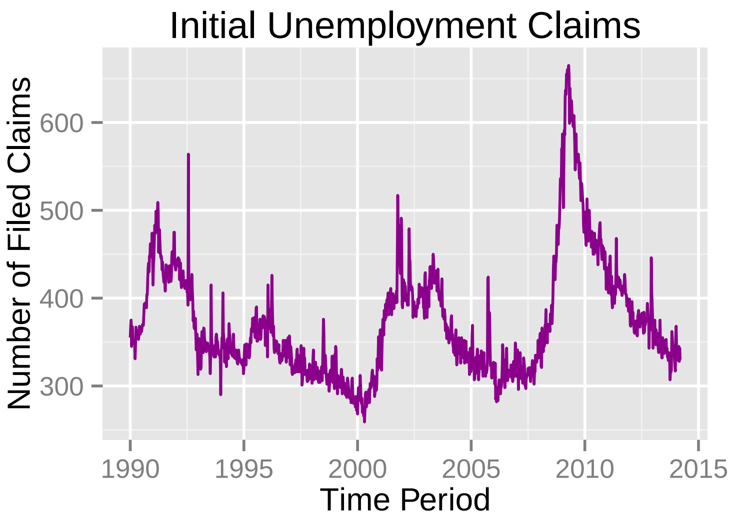
```
##      data_ts.time data_ts
## [1,]     1990.000     355
## [2,]     1990.019     369
## [3,]     1990.038     375
## [4,]     1990.058     345
## [5,]     1990.077     368
## [6,]     1990.096     367
```

...

e.

1. **Plot the time series plot of `INJCJC`. Remember that the graph must be well labelled.**

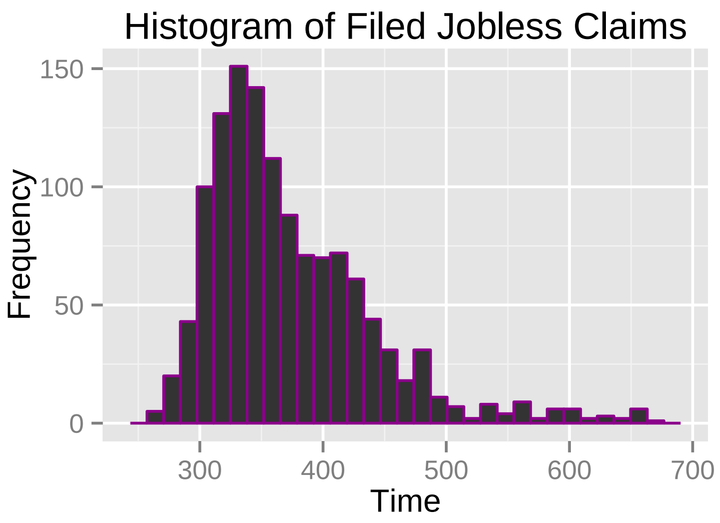
```r
autoplot(data_ts , xlab = "Time Period", ylab = "Number of Filed Claims",
         main="Initial Unemployment Claims", ts.colour = 'magenta4' )
```



...

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

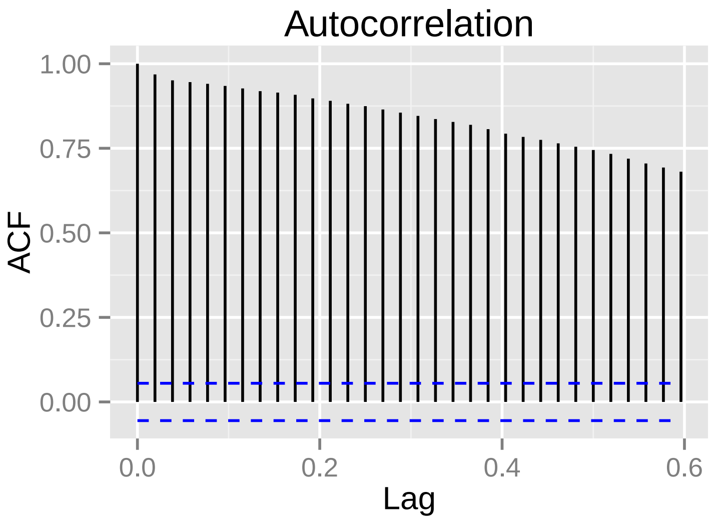
```r
qplot(data_ts , geom="histogram", main='Histogram of Filed Jobless Claims',
      ylab='Frequency', xlab='Time', colour = I('magenta4'))
```



. . .

3. **Plot the autocorrelation graph of `INJCJC` series.**

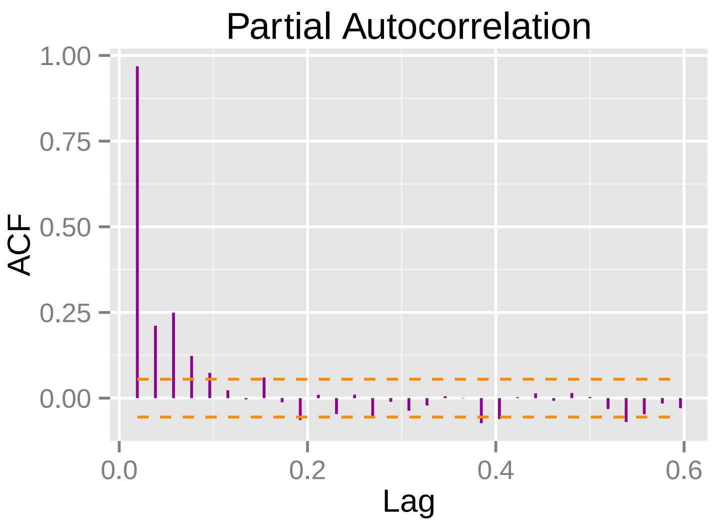
```
autoplot(acf(data_ts, plot = FALSE), xlab = "Lag", main="Autocorrelation",
        ts.colour = 'magenta4')
```



. . .

4. **Plot the partial autocorrelation graph of `INJCJC` series.**

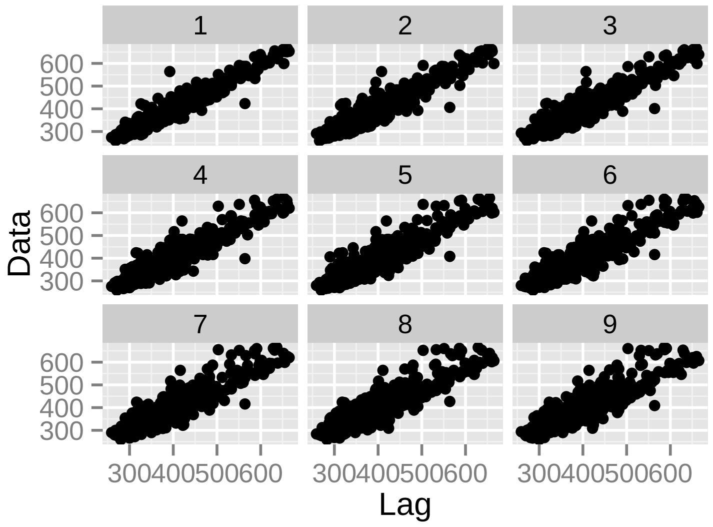
```
autoplot(pacf(data_ts, plot = FALSE), xlab = "Lag",main="Partial Autocorrelation",
        colour = "magenta4", conf.int.colour = "darkorange")
```



. . .

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

```
gglagplot(data_ts, lags = 9, nrow = 3, ncol = 3)
```

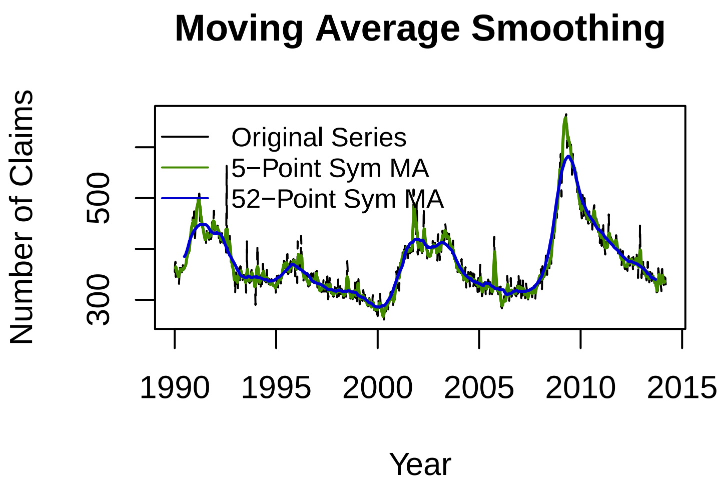


. . .

f.

1. **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.**

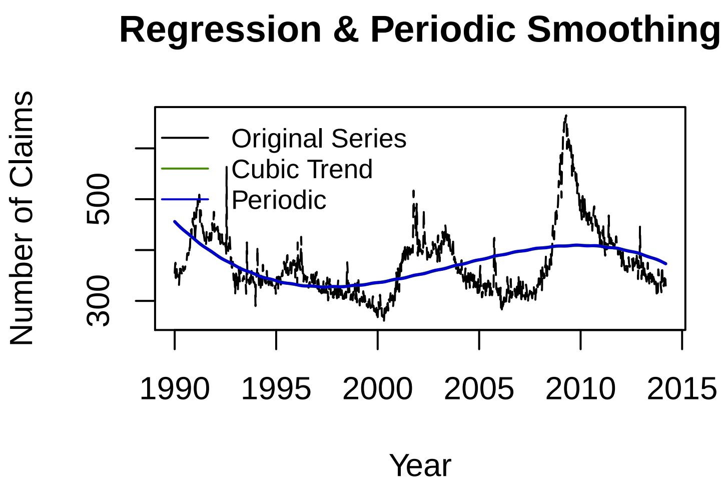
```
data_ts_rep5 = filter(data_ts, sides=2, rep(1,5)/5)
data_ts_rep52 = filter(data_ts, sides=2, rep(1,52)/52)
plot(data_ts, main="Moving Average Smoothing",
     pch=4, lty=5, lwd=1, xlab="Year",
     ylab="Number of Claims")
lines(data_ts_rep5, lty=1, lwd=1.5, col="chartreuse4")
lines(data_ts_rep52, lty=1, lwd=1.5, col="mediumblue")
# Add Legend
leg.txt <- c("Original Series", "5-Point Sym MA", "52-Point Sym MA")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","chartreuse4","mediumblue"),
       bty='n', cex=0.8, merge = TRUE, bg=336)
```



. . .

2. **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.**

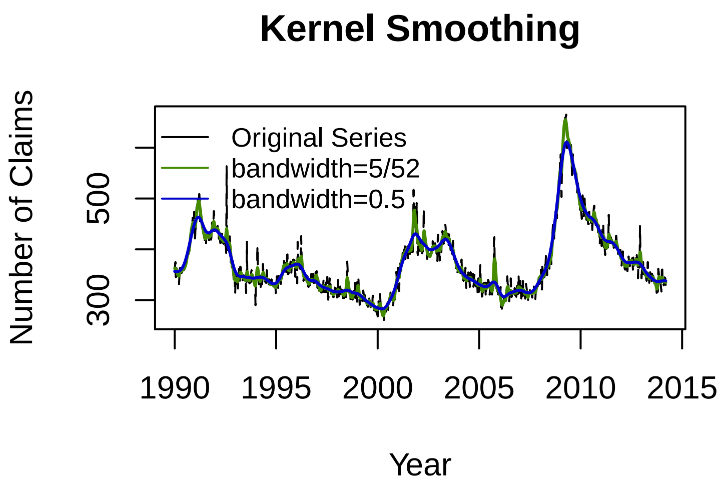
```
wk = time(data_ts) - mean(time(data_ts))
wk2 = wk^2
wk3 = wk^3
cs = cos(2*pi*wk)
sn = sin(2*pi*wk)
reg1 = lm(data_ts~wk + wk2 + wk3, na.action=NULL)
reg2 = lm(data_ts~wk + wk2 + wk3 + cs + sn, na.action=NULL)
plot(data_ts, main="Regression & Periodic Smoothing",
     pch=4, lty=5, lwd=1, xlab="Year",
     ylab="Number of Claims")
lines(fitted(reg1), lty=1, lwd=1.5, col="chartreuse4")
lines(fitted(reg2), lty=1, lwd=1.5, col="mediumblue")
# Add Legend
leg.txt <- c("Original Series", "Cubic Trend", "Periodic")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","chartreuse4","mediumblue"),
       bty='n', cex=0.8, merge = TRUE, bg=336)
```



. . .

3. **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(data_ts, main="Kernel Smoothing",
     pch=4, lty=5, lwd=1, xlab="Year",
     ylab="Number of Claims")
lines(ksmooth(time(data_ts), data_ts, "normal", bandwidth=5/52),lty=1, lwd=1.5, col="chartreuse4")
lines(ksmooth(time(data_ts), data_ts, "normal", bandwidth=0.5),lty=1, lwd=1.5, col="mediumblue")
# Add Legend
leg.txt <- c("Original Series", "bandwidth=5/52", "bandwidth=0.5")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","chartreuse4","mediumblue"),
       bty='n', cex=0.8, merge = TRUE, bg=336)
```

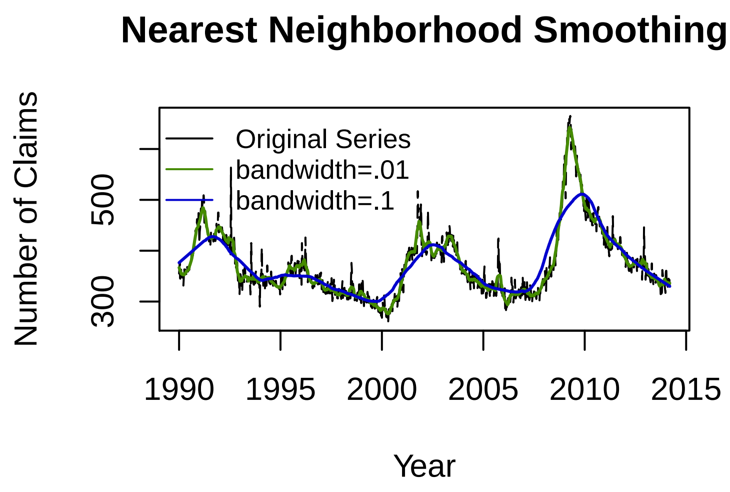


. . .

4. **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.

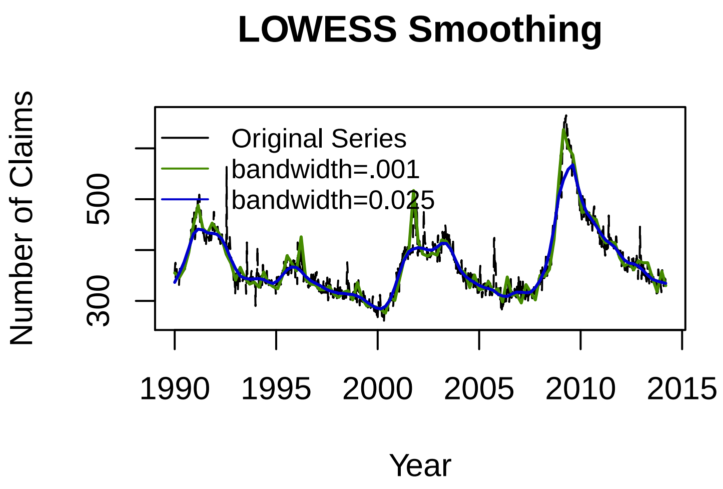
```
plot(data_ts, main="Nearest Neighborhood Smoothing",
     pch=4, lty=5, lwd=1, xlab="Year",
     ylab="Number of Claims")
lines(supsmu(time(data_ts), data_ts, span=.01),lty=1, lwd=1.5, col="chartreuse4")
lines(supsmu(time(data_ts), data_ts, span=.1),lty=1, lwd=1.5, col="mediumblue")
# Add Legend
leg.txt <- c("Original Series", "bandwidth=.01", "bandwidth=.1")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","chartreuse4","mediumblue"),
       bty='n', cex=0.8, merge = TRUE, bg=336)
```



. . .

5. **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.**

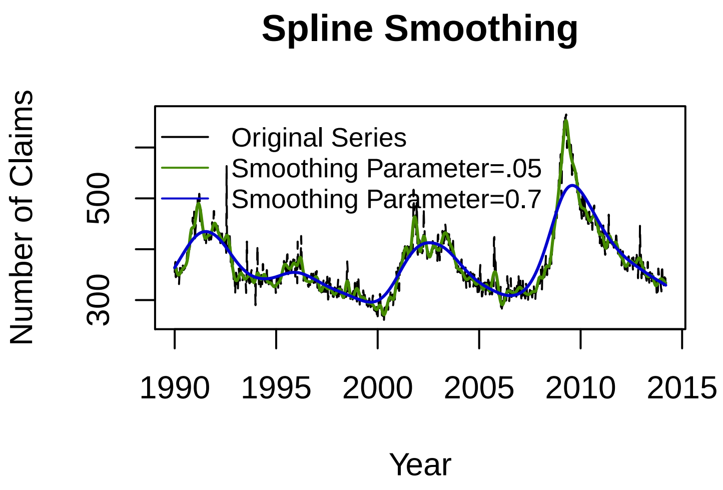
```
plot(data_ts, main="LOWESS Smoothing",
     pch=4, lty=5, lwd=1, xlab="Year",
     ylab="Number of Claims")
lines(lowess(data_ts, f=.001),lty=1, lwd=1.5, col="chartreuse4")
lines(lowess(data_ts, f=0.05),lty=1, lwd=1.5, col="mediumblue")
# Add Legend
leg.txt <- c("Original Series", "bandwidth=.001", "bandwidth=0.025")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","chartreuse4","mediumblue"),
       bty='n', cex=0.8, merge = TRUE, bg=336)
```



. . .

6. **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.**

```r
plot(data_ts, main="Spline Smoothing",
     pch=4, lty=5, lwd=1, xlab="Year",
     ylab="Number of Claims")
lines(smooth.spline(time(data_ts), data_ts, spar=0.05),lty=1, lwd=1.5, col="chartreuse4")
lines(smooth.spline(time(data_ts), data_ts, spar=0.75),lty=1, lwd=1.5, col="mediumblue")
# Add Legend
leg.txt <- c("Original Series", "Smoothing Parameter=.05", "Smoothing Parameter=0.7")
legend("topleft", legend=leg.txt, lty=c(1,1,1), col=c("black","chartreuse4","mediumblue"),
       bty='n', cex=0.8, merge = TRUE, bg=336)
```



. . .