W271-2 – Spring 2016 – **HW 6**

Juanjo Carin, Kevin Davis, Ashley Levato, Minghu Song

March 16, 2016

Contents

Exercises **2**

Exercise 1 2

Exercise 2 3

Exercise 3 5

Exercise 4 7

Exercises

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:

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

Where the probability density function f_t is the marginal distribution of x_t derived from the complete joint probability distribution $F(c_1, c_2, \ldots, c_n) = \Pr(x_{t_1} \leq c_1, x_{t_2} \leq c_2, \ldots, x_{t_n} \leq c_n)$.

This function has a time component so the mean could be different in different time periods (if it is not, the time series is *stationary in the mean*). This is different from a mean in classical linear models where the mean is always constant.

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

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

Again this function is time dependent, which means it may vary with time unlike the variance in a classical linear model. If it is constant, the time series is *stationary in the variance* or *variance stationary*.

In both cases we are calculating expectations over the ensemble—the set of all time series that could be generated by the stochastic process.

b. **Define strict and weak stationarity**

Strict stationary occurs when the joint distributions $F(x_{t_1}, ..., x_{t_n})$ and $F(x_{t_{1+m}}, ..., x_{t_{n+m}})$ are the same for all $t_1, \ldots, t_n, m$, implying that the distribution is unchanged for any time shift m.

A time series is *weakly stationary* (also called *second-order stationary*) when it is mean and variance stationary and its autocovariance $Cov(x_t, x_{t+k})$ depends only on the time shift k (it is then written as γ_k. Once a distribution assumption is imposed, the time series can be completely characterized by its mean and covariance structure.

Exercise 2

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

```
set.seed(123)
N <- 500 # number of simulations / time periods
wn <- rnorm(n = N, mean = 0, sd = 1) # white noise (can use any mean and sd)
rw <- cumsum(wn)
```

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.**

```
# See the definition of the function in ## @knitr Libraries-Functions-Constants
desc_stat(rw, 'Random walk',
          'Descriptive statistics of the simulated random walk')
```

Table 1: Descriptive statistics of the simulated random walk

	Random walk
Mean	4.58
St. Dev	4.87
1st Quartile	0.97
Median	3.42
3rd Quartile	8.27
Min	-5.16
Max	19.35

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

See the last part of this Exercise in the following page.

d. **Plot the autocorrelation graph.**

See the last part of this Exercise in the following page.

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

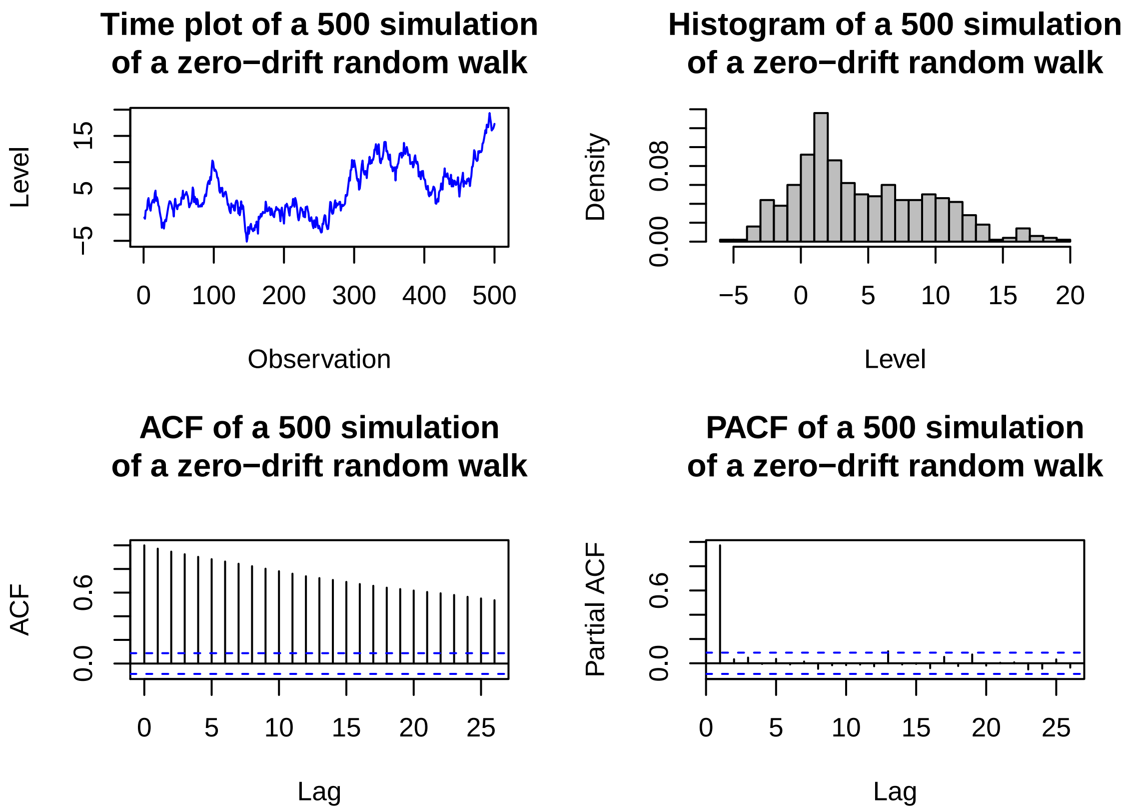


Figure 1: Time-series plot, histogram, correlogram, and partial autocorrelogram of the 500 simulation of a zero-drift random walk

Exercise 3

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

```
drift <- 0.5 # drift
# Use the same GWN that genereate the prev. zero-drift RW
# set.seed(123); wn <- rnorm(n = N, mean = 0, sd = 1)
rw_drift <- cumsum(wn + drift)
```

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.**

```
# See the definition of the function in ## @knitr Libraries-Functions-Constants
desc_stat(cbind(rw, rw_drift), c('Random walk', 'Random walk with 0.5 drift'),
          'Descriptive statistics of the two simulated random walks')
```

Table 2: Descriptive statistics of the two simulated random walks

	Random walk	Random walk with 0.5 drift
Mean	4.58	129.83
St. Dev	4.87	75.43
1st Quartile	0.97	64.58
Median	3.42	122.91
3rd Quartile	8.27	199.68
Min	-5.16	-0.06
Max	19.35	267.30

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

See the last part of this Exercise in the following page.

d. **Plot the autocorrelation graph.**

See the last part of this Exercise in the following page.

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

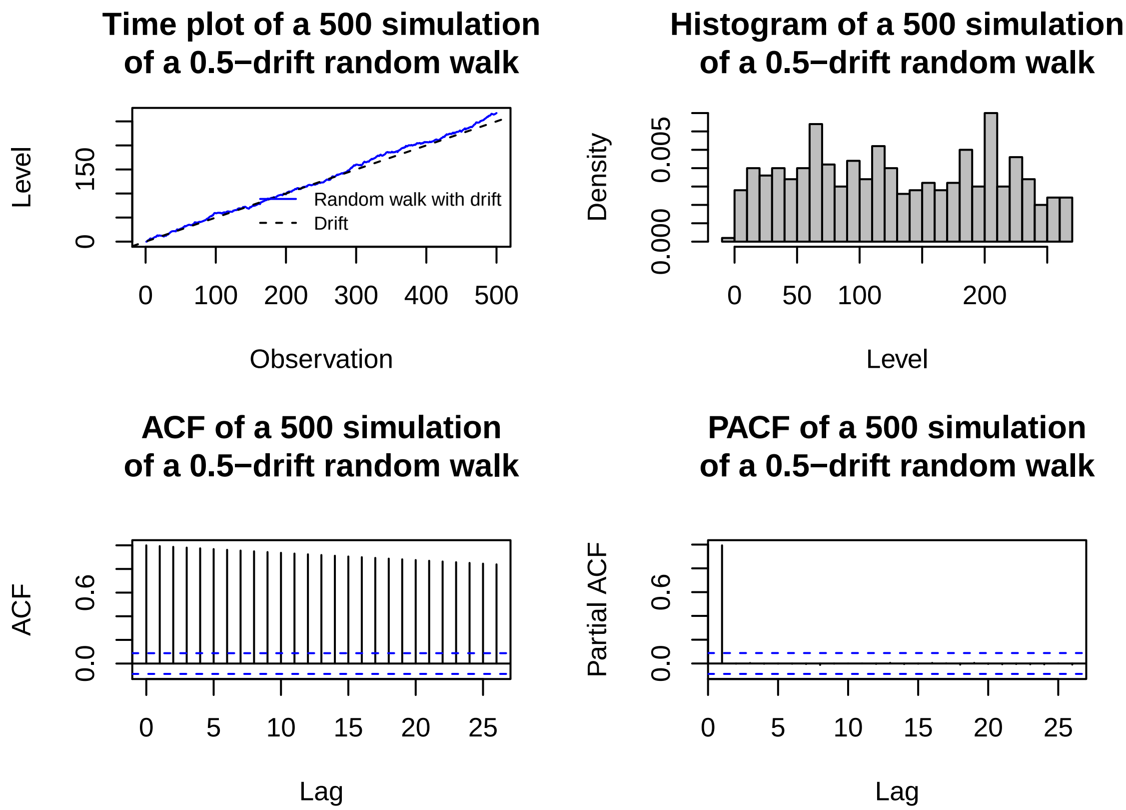


Figure 2: Time-series plot, histogram, correlogram, and partial autocorrelogram of the 500 simulation of a 0.5-drift random walk

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.**

```
INJCJC_df <- read.csv('INJCJC.csv', header = TRUE)
str(INJCJC_df)
```

```
## 'data.frame':    1300 obs. of  3 variables:
##  $ Date   : Factor w/ 1300 levels "10-Apr-09","10-Apr-92",..: 1102 101 400 741 907 1271 270 611 922
##  $ INJCJC : int  355 369 375 345 368 367 348 350 351 349 ...
##  $ INJCJC4: num  362 366 364 361 364 ...
```

```
dim(INJCJC_df); obs <- dim(INJCJC_df)[1]
```

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

```
head(INJCJC_df)
```

```
##        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_df)
```

```
##           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
```

Table 3: Descriptive statistics of the INJCJC variables

	INJCJC	INJCJC4
Mean	371.14	371.24
St. Dev	67.38	66.30
1st Quartile	324.00	324.69
Median	353.50	352.12
3rd Quartile	406.00	405.75
Min	259.00	266.25
Max	665.00	659.25

The 1300 observations (of two variables, `INJCJC` and `INJCJC4`) correspond to $1,300/52 = 25$ periods of 52 weeks, i.e., almost 25 years from January 5, 1990, until November 28, 2014.

All observations correspond to Fridays.

```
levels(as.factor(weekdays(as.Date(as.character(INJCJC_df$Date), '%d-%b-%y'))))
```

```
## [1] "Friday"
```

Since years are slightly longer (by 1 or 2 days) than 52 weeks, some years have 53 Fridays. Those years with 53 Fridays sum up, and as a result (and since the number of observations we have is a multiple of 52) the last year in the sample (2014) does not include all of the Fridays there were that year (there were 1,304 Fridays in that 25-year period from 1990 to 2014).

```
INJCJC_df %>%
  mutate(Date = as.Date(as.character(Date), '%d-%b-%y')) %>%
  mutate(Year = year(Date)) %>%
  group_by(Year) %>%
  summarise(obs = n(), start_date = min(Date), end_date = max(Date)) %>%
  print(n=Inf)
```

```
## Source: local data frame [25 x 4]
##
##     Year   obs start_date   end_date
##    (dbl) (int)     (date)     (date)
## 1   1990    52 1990-01-05 1990-12-28
## 2   1991    52 1991-01-04 1991-12-27
## 3   1992    52 1992-01-03 1992-12-25
## 4   1993    53 1993-01-01 1993-12-31
## 5   1994    52 1994-01-07 1994-12-30
## 6   1995    52 1995-01-06 1995-12-29
## 7   1996    52 1996-01-05 1996-12-27
## 8   1997    52 1997-01-03 1997-12-26
## 9   1998    52 1998-01-02 1998-12-25
## 10  1999    53 1999-01-01 1999-12-31
## 11  2000    52 2000-01-07 2000-12-29
## 12  2001    52 2001-01-05 2001-12-28
## 13  2002    52 2002-01-04 2002-12-27
## 14  2003    52 2003-01-03 2003-12-26
## 15  2004    53 2004-01-02 2004-12-31
## 16  2005    52 2005-01-07 2005-12-30
## 17  2006    52 2006-01-06 2006-12-29
## 18  2007    52 2007-01-05 2007-12-28
## 19  2008    52 2008-01-04 2008-12-26
## 20  2009    52 2009-01-02 2009-12-25
## 21  2010    53 2010-01-01 2010-12-31
## 22  2011    52 2011-01-07 2011-12-30
## 23  2012    52 2012-01-06 2012-12-28
## 24  2013    52 2013-01-04 2013-12-27
## 25  2014    48 2014-01-03 2014-11-28
```

```r
# Count weeks
sum(sapply(1990:2014, function(y)
  ifelse(((y %% 4 == 0) & (y %% 100 != 0)) | (y %% 400 == 0), 366, 365))) / 7
```

```
## [1] 1304.429
```

```r
# Count Fridays in that period
ceiling(as.numeric(as.Date('2014-12-31') + 1 - 5 + 4) / 7) -
  ceiling(as.numeric(as.Date('1990-01-01') - 5 + 4) / 7)
```

```
## [1] 1304
```

Another way to see it: because some years had more than 52 Fridays, each set of 52 observations gradually ends earlier (see the even rows below, that correspond to `week == 52`: December 28, December 27, December 25, December 24. . .). Hence, the next set of 52 observations start earlier, even in the previous year (see row 8), and the aggregate effect is that the 25th set of 52 observations—which should correspond to 2014—start as early as December 6, 2013, and ends in November 28, 2014.

```r
INJCJC_df %>%
  mutate(week_num = 1:obs,
         week = ifelse(week_num %% 52== 0, 52, week_num %% 52)) %>%
  filter(week %% 52 < 2)
```

```
##         Date INJCJC INJCJC4 week_num week
## 1   5-Jan-90    355  362.25        1    1
## 2  28-Dec-90    454  456.00       52   52
## 3   4-Jan-91    415  447.50       53    1
## 4  27-Dec-91    441  456.75      104   52
## 5   3-Jan-92    432  446.00      105    1
## 6  25-Dec-92    313  339.00      156   52
## 7   1-Jan-93    341  336.75      157    1
## 8  24-Dec-93    290  323.00      208   52
## 9  31-Dec-93    341  324.00      209    1
## 10 23-Dec-94    314  324.25      260   52
## 11 30-Dec-94    319  323.00      261    1
## 12 22-Dec-95    374  366.50      312   52
## 13 29-Dec-95    359  363.00      313    1
## 14 20-Dec-96    350  347.25      364   52
## 15 27-Dec-96    357  353.50      365    1
## 16 19-Dec-97    310  317.00      416   52
## 17 26-Dec-97    303  313.25      417    1
## 18 18-Dec-98    297  309.50      468   52
## 19 25-Dec-98    336  316.00      469    1
## 20 17-Dec-99    287  283.50      520   52
## 21 24-Dec-99    268  278.50      521    1
## 22 15-Dec-00    354  342.25      572   52
## 23 22-Dec-00    364  344.25      573    1
## 24 14-Dec-01    389  434.50      624   52
## 25 21-Dec-01    416  415.75      625    1
## 26 13-Dec-02    429  405.25      676   52
## 27 20-Dec-02    394  406.25      677    1
## 28 12-Dec-03    363  360.25      728   52
```

```
## 29 19-Dec-03    354  360.25      729    1
## 30 10-Dec-04    316  326.75      780   52
## 31 17-Dec-04    322  329.00      781    1
## 32  9-Dec-05    327  320.75      832   52
## 33 16-Dec-05    312  317.75      833    1
## 34  8-Dec-06    311  328.25      884   52
## 35 15-Dec-06    318  326.25      885    1
## 36  7-Dec-07    332  340.00      936   52
## 37 14-Dec-07    350  344.50      937    1
## 38  5-Dec-08    570  541.75      988   52
## 39 12-Dec-08    566  549.25      989    1
## 40  4-Dec-09    497  490.25     1040   52
## 41 11-Dec-09    498  488.00     1041    1
## 42  3-Dec-10    431  427.50     1092   52
## 43 10-Dec-10    428  426.00     1093    1
## 44  2-Dec-11    390  390.25     1144   52
## 45  9-Dec-11    369  386.25     1145    1
## 46 30-Nov-12    379  406.25     1196   52
## 47  7-Dec-12    343  380.50     1197    1
## 48 29-Nov-13    317  328.75     1248   52
## 49  6-Dec-13    358  332.75     1249    1
## 50 28-Nov-14    297  299.00     1300   52
```

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.**

The above parameters may be wrong: according to the `ts` documentation, `start` and `end` have to be "*Either a single number or a vector of two integers, which specify a natural time unit and a (1-based) number of samples into the time unit*," i.e., and not a date in `y-m-d` format. If we use the parameters in the instructions, R discards `28` and takes `11` observations from 2014, not the 52 that there really are.

```
INJCJC_wrong <- ts(INJCJC_df$INJCJC, frequency = 52, start = c(1990, 1, 1),
                   end = c(2014, 11, 28))
length(INJCJC_wrong) # 1300 - (52 -11)
```

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

```
tail(INJCJC_wrong, 10)
```

```
##  [1] 368 339 344 333 329 334 345 328 343 330
```

```
tail(INJCJC_df$INJCJC[1:length(INJCJC_wrong)], 10)
```

```
##  [1] 368 339 344 333 329 334 345 328 343 330
```

We can plot both the original vector and the time series object to confirm this.

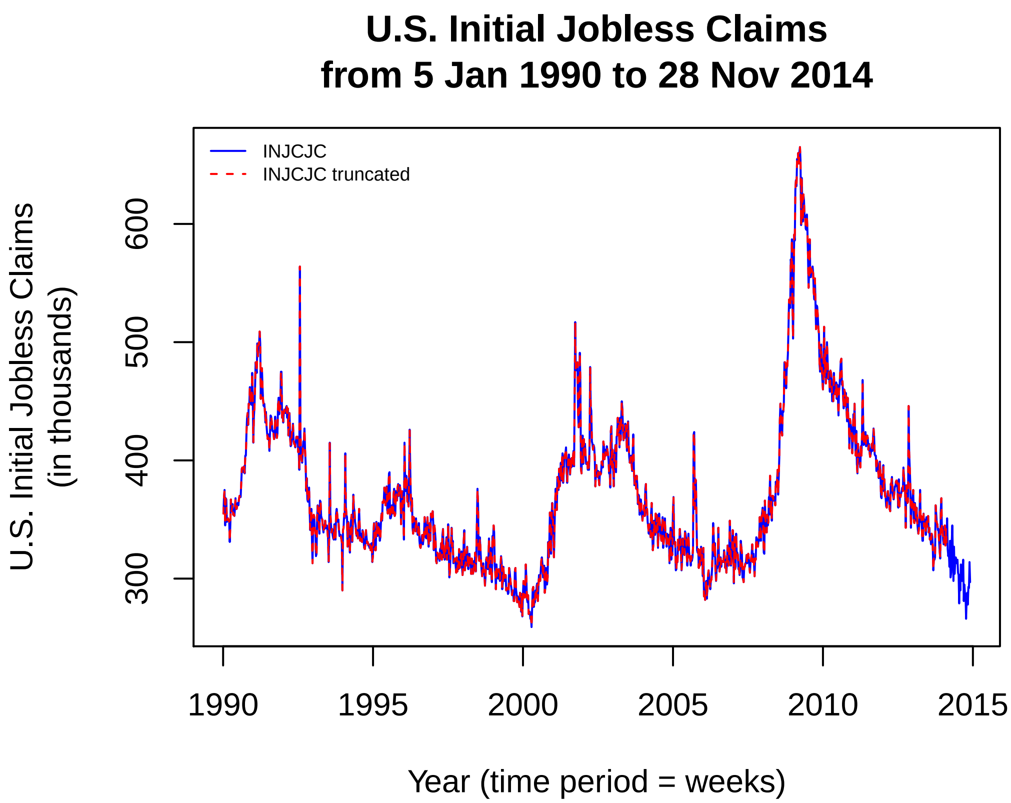


Figure 3: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014

Hence, we slightly change the R command to get the proper time series object:

```
INJCJC <- ts(INJCJC_df$INJCJC, frequency = 52, start = c(1990, 1),
             end = c(2014, 52))
length(INJCJC)
```

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

The way `ts` works, our time scale does not correspond to the original dates (every Friday between the start and end date, including both) because we've set a fixed frequency of 52 (instead of $365/7 = 52.143$ if the year is not leap, and $366/7 = 52.286$ if it is) and the 1st time period is set to the very beginning of the year. Compare the original dates in our dataset with the dates `ts` sets, for the 1st and 52nd observation (while the day and month of the 1st and 52nd observations of each of the 25 subsets will vary, as we have seen, the day and month—and even time—will be fixed in the `ts` object, regardless of the year); the date of the last observation in the "wrong/truncated version" of `INJCJC` is also shown.:

```
comp <- rbind(as.character(INJCJC_df$Date[c(1, 52, length(INJCJC_wrong))]),
              formatC(time(INJCJC)[c(1, 52, length(INJCJC_wrong))], 3,
                      format = 'f'),
              format(date_decimal(time(INJCJC)[c(1, 52, length(INJCJC_wrong))]),
                     "%d-%b-%Y %H:%M:%S"))
comp
```

```
##                     1                         52
## Original date       "5-Jan-90"                "28-Dec-90"
## time(ts)            "1990.000"                "1990.981"
## Corresponding date "01-Jan-1990 00:00:00" "24-Dec-1990 23:32:18"
##                     1259
## Original date       "14-Feb-14"
## time(ts)            "2014.192"
## Corresponding date "12-Mar-2014 04:36:55"
```

This does not happen with montly or quarterly data (and happens to a lesser extent with daily data).

One way to preserve the sampling dates is by using the `xts` library:

```
INJCJC_2 <- xts(INJCJC_df$INJCJC,
                order.by = as.Date(as.character(INJCJC_df$Date), '%d-%b-%y'))
head(INJCJC_2)
```

```
##            [,1]
## 1990-01-05  355
## 1990-01-12  369
## 1990-01-19  375
## 1990-01-26  345
## 1990-02-02  368
## 1990-02-09  367
```

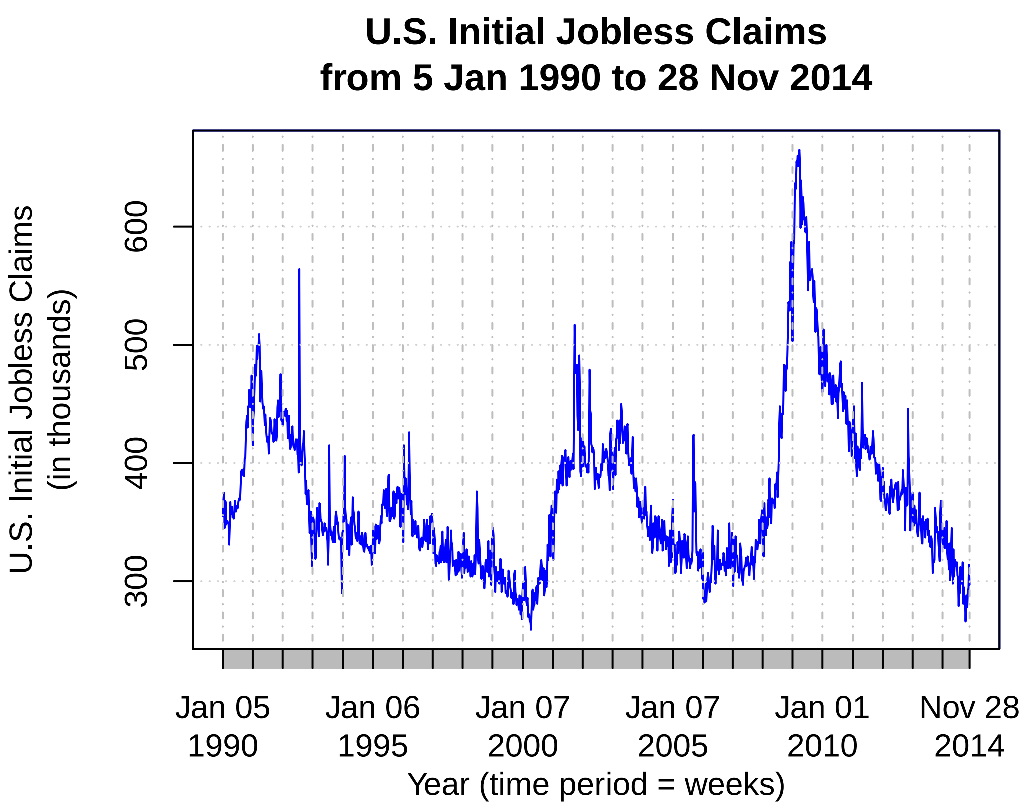


Figure 4: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014, using the `xts` package

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

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

```
## [1] 1990.000 1990.019 1990.038 1990.058 1990.077 1990.096
```

```
tail(INJCJC.time)
```

```
## [1] 2014.885 2014.904 2014.923 2014.942 2014.962 2014.981
```

```
tail(time(INJCJC_wrong))
```

```
## [1] 2014.096 2014.115 2014.135 2014.154 2014.173 2014.192
```

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), 10)
```

```
##       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
```

```
head(cbind(INJCJC.time, INJCJC)) # default: 6
```

```
##      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
```

```
head(cbind(INJCJC.time, INJCJC), -(length(INJCJC)-6)) # -1294: equivalent
```

```
##      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
```

```r
head(cbind(INJCJC.time, INJCJC), 13) # approximately 3 months (1 quarter)
```

```
##       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
```

e.

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

```
plot.ts(INJCJC, col = 'blue', xlab = "Year (time period = weeks)",
        ylab = "U.S. Initial Jobless Claims\n(in thousands)",
        main = "U.S. Initial Jobless Claims\nfrom 5 Jan 1990 to 28 Nov 2014")
```

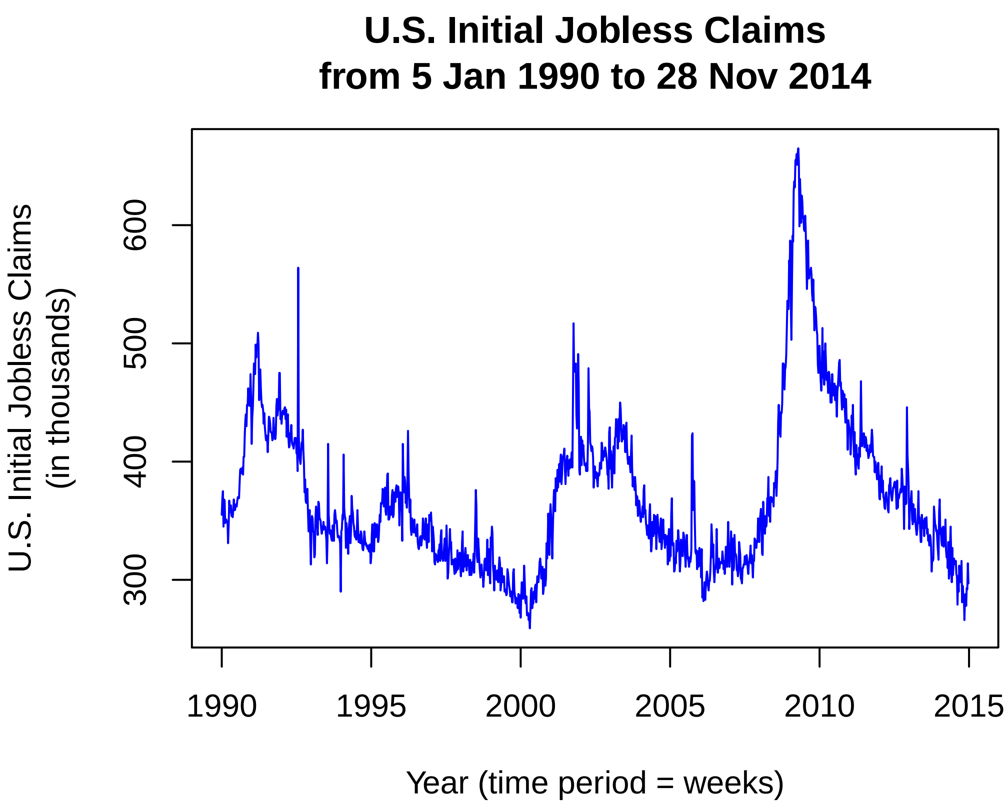
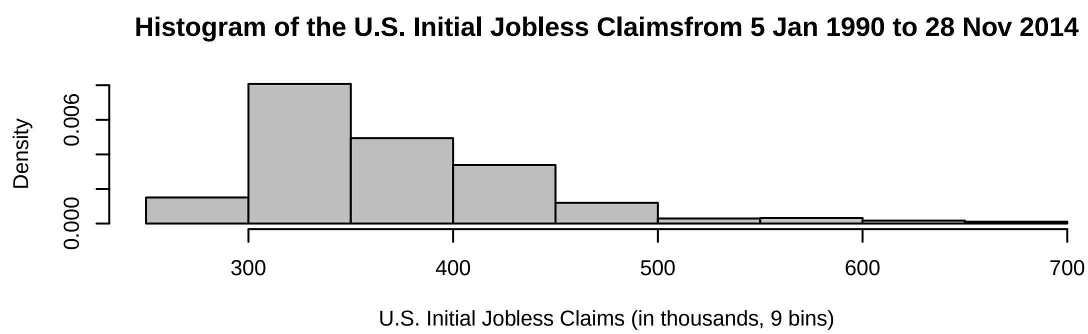


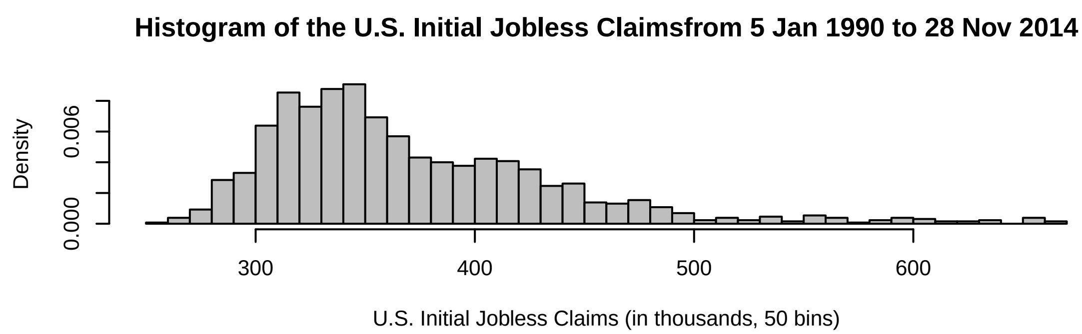
Figure 5: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014

Compare the previous Figure, plotted using `xts`, with the one above: the former shows the proper time scale (starting in January 5, 1990, and ending in November 28, 2014), while the latter covers the full 25-year period.

2. **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 = 50, col="gray", freq = FALSE,
     xlab = "U.S. Initial Jobless Claims (in thousands, 50 bins)",
     main = paste0("Histogram of the U.S. Initial Jobless Claims",
                   "from 5 Jan 1990 to 28 Nov 2014"))
```





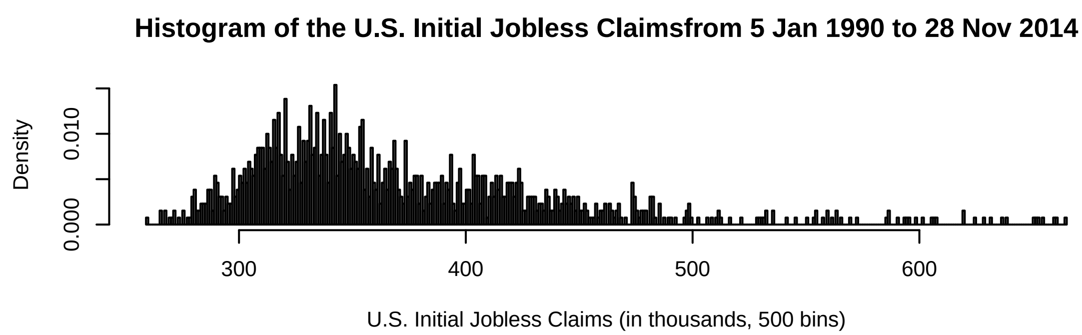


Figure 6: Histogram of the U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014, using different bins

The histograms tells us nothi ng about the dynamics of the series; e.g., it lets us know that there were a few time periods where the number of Initial Jobless Claims was greater than 600 (thousand), but not when that happened (in the 1st half of 2009).

A good number of bins is one that neither *oversimplifies* the (sample) distribution (i.e., it is not so low that some modes may be hidden) nor shows too much detail that is due to sampling and makes the underlying (real) distribution much more complex than it really is (i.e., it is so high that a low of modes, just due to sampling, are shown). Hence, the best representation in the previous Figure would be the one in the midle.

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

We plot the first 52 correlations (apart from ρ_0) just to check all possible seasonality components (incl. annual).

```
acf(INJCJC, lag.max = 52,
    main="ACF of U.S. Initial Jobless Claims\nfrom 5 Jan 1990 to 28 Nov 2014")
```

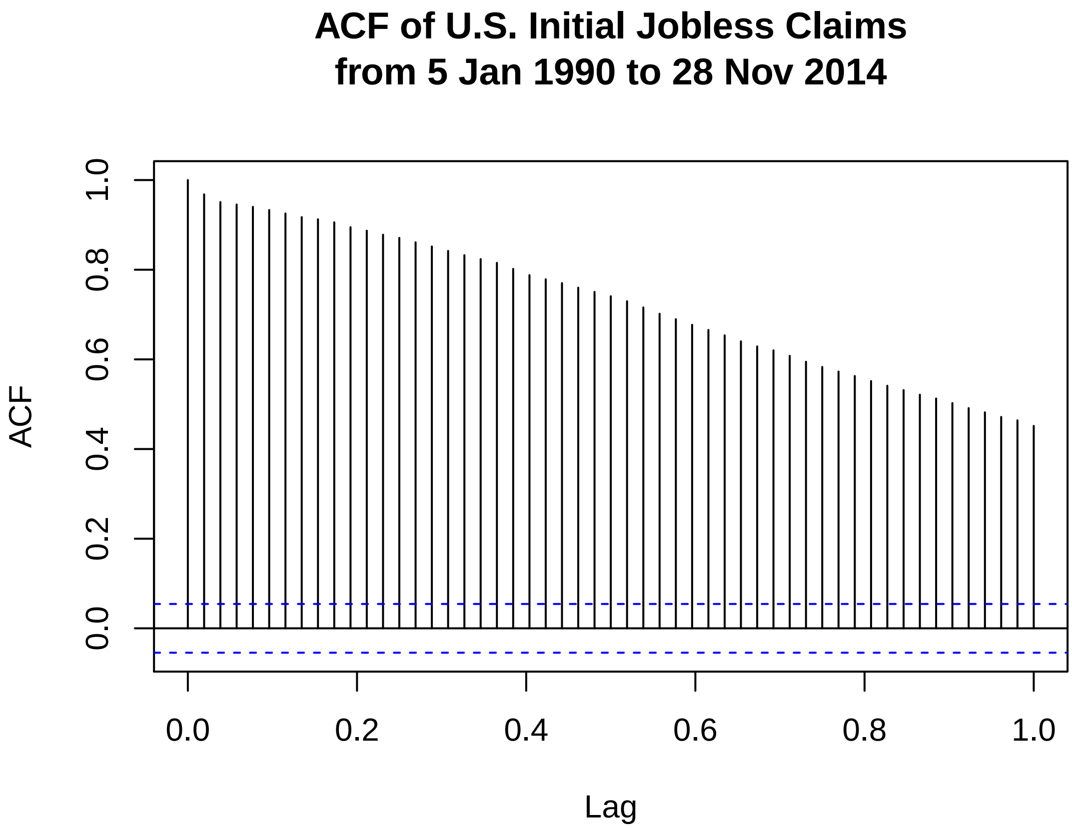


Figure 7: Partial autocorrelation graph of U.S. Initial Jobless Claims from 5 Jan 1990 to 28 Nov 2014

The autocorrleation decreases exponentially, but very slowly (this is typical of an AR(1) model with a coefficient very close to 1). Nonetheless, we must note that the time series has neither been seasonally nor detrended.

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

```
pacf(INJCJC, lag.max = 52,
     main="PACF of U.S. Initial Jobless Claims\nfrom 5 Jan 1990 to 28 Nov 2014")
```

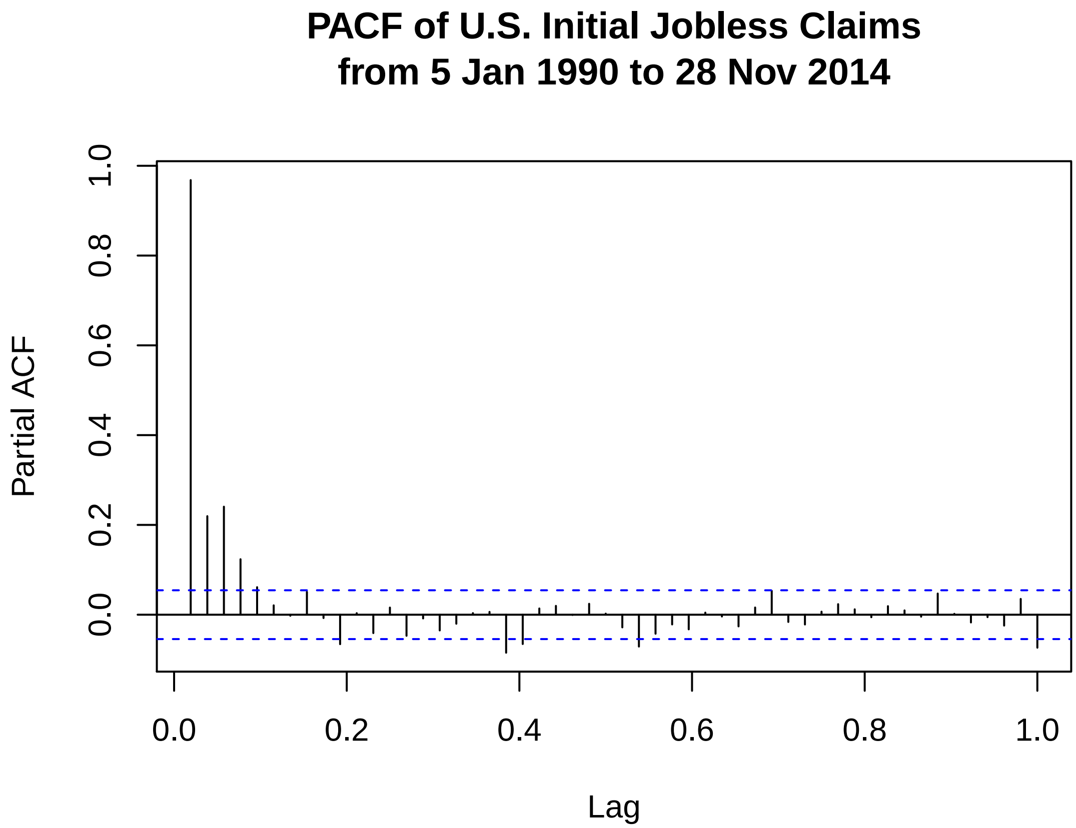


Figure 8: Partial autocorrelation graph of U.S. Initial Jobless Claims from 5 Jan 1990 to 28 Nov 2014

The partial autocorrelation plot makes it evident that this time series was not generated by an AR(1) model. After controlling for the effect of the process at lags 1 and 2, the partial correlation at lags 2 and 3, respectively, is still significant.

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

```
lag.plot(INJCJC, lags = 9, layout = c(3, 3), diag=TRUE, diag.col="red",
         main = paste("Autocorr. between the INJCJC time series and its",
                      "Own Lags"))
```

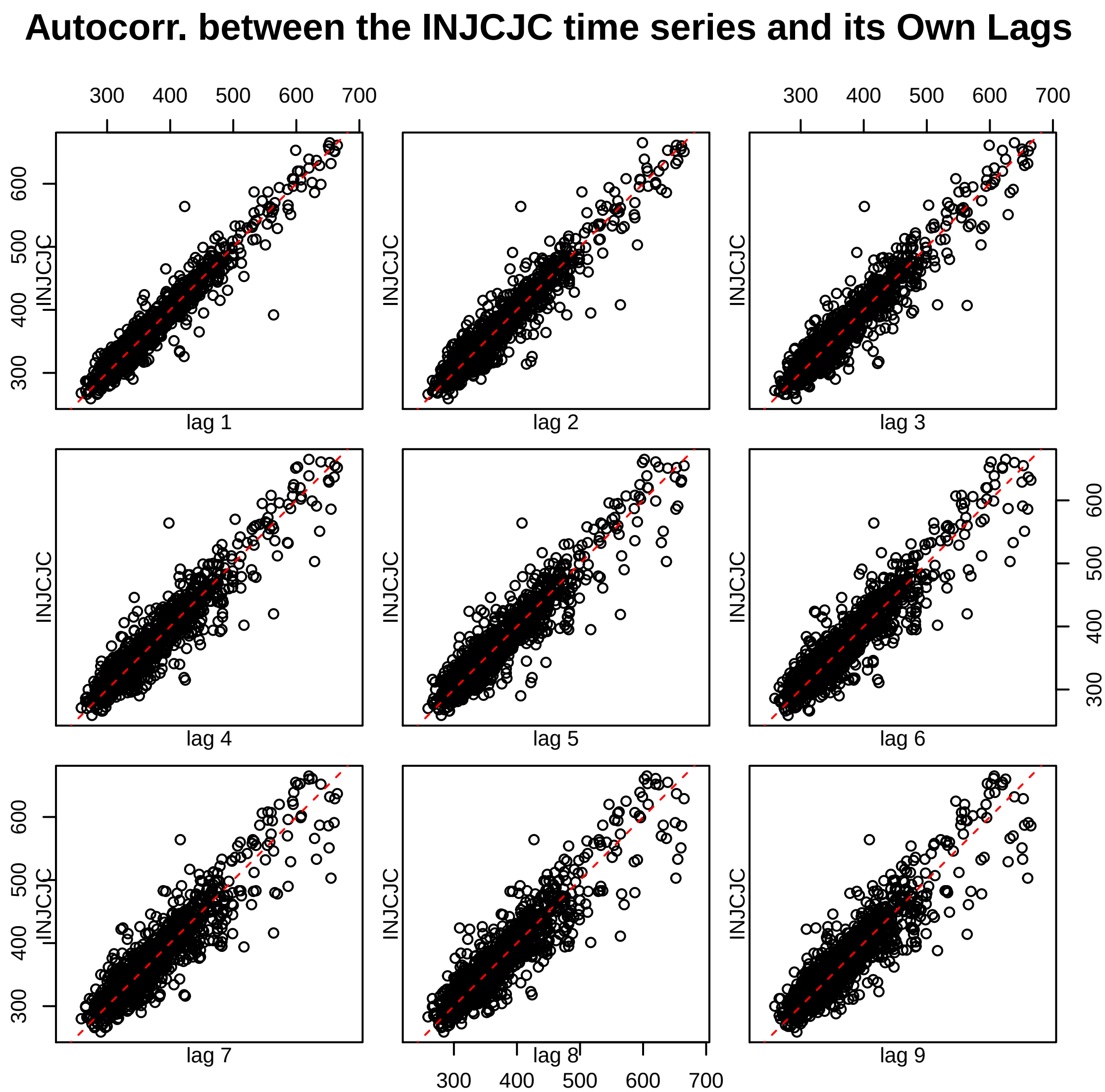


Figure 9: Scatterplot matrix of the correlation of the U.S. Initial Jobless Claims time series against its first 9 own lags

As the correlogram in Figue 7 suggested, the correlation between the time series and its lagged version is very high, and that correlation decreases very slowly with the lag.

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.**

```
INJCJC.1 = stats::filter(INJCJC, sides=2, rep(1, 5)/5)
INJCJC.2 = stats::filter(INJCJC, sides=2, rep(1, 53)/53)
```

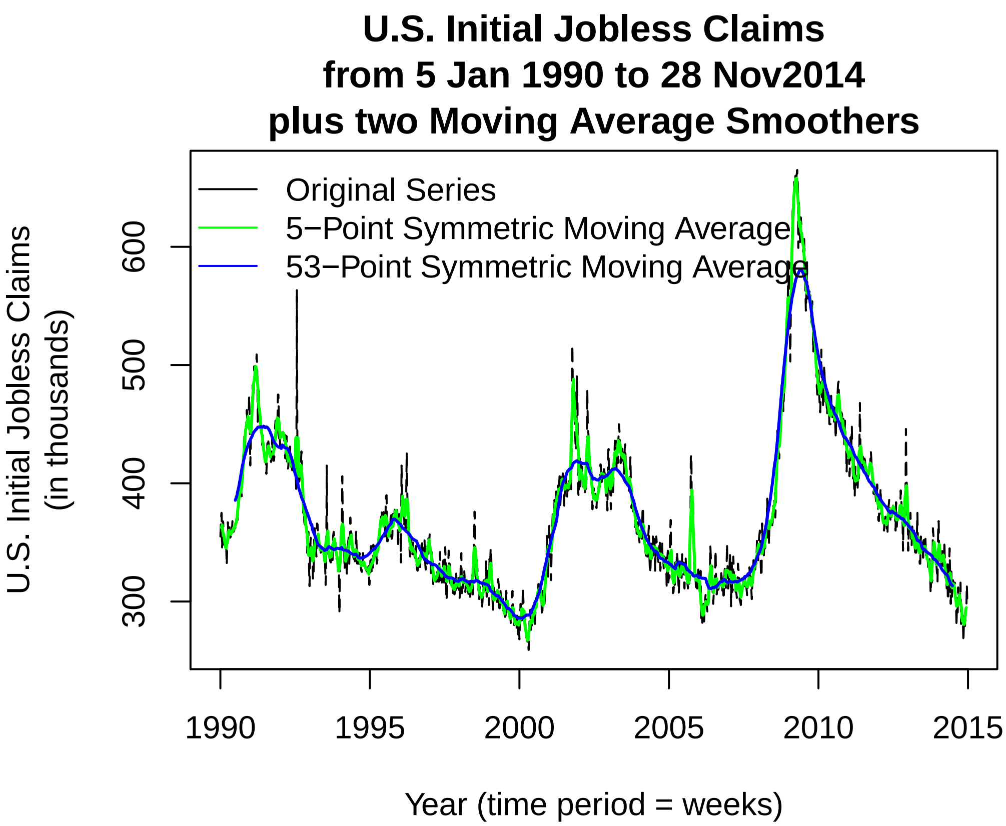


Figure 10: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014 (original series plus two Moving Average Smoothers)

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.**

```
cbind(INJCJC.time[c(1:2, (length(INJCJC)-1):length(INJCJC))],
      mean(INJCJC.time))
```

```
##          [,1]    [,2]
## [1,] 1990.000 2002.49
## [2,] 1990.019 2002.49
## [3,] 2014.962 2002.49
## [4,] 2014.981 2002.49
```

```
wk = INJCJC.time - mean(INJCJC.time)
wk2 = wk^2
wk3 = wk^3
cs = cos(2 * pi * wk)
sn = sin(2 * pi * wk)
reg1 = lm(INJCJC ~ wk + wk2 + wk3, na.action = NULL)
reg2 = lm(INJCJC ~ wk + wk2 + wk3 + cs + sn, na.action = NULL)
```

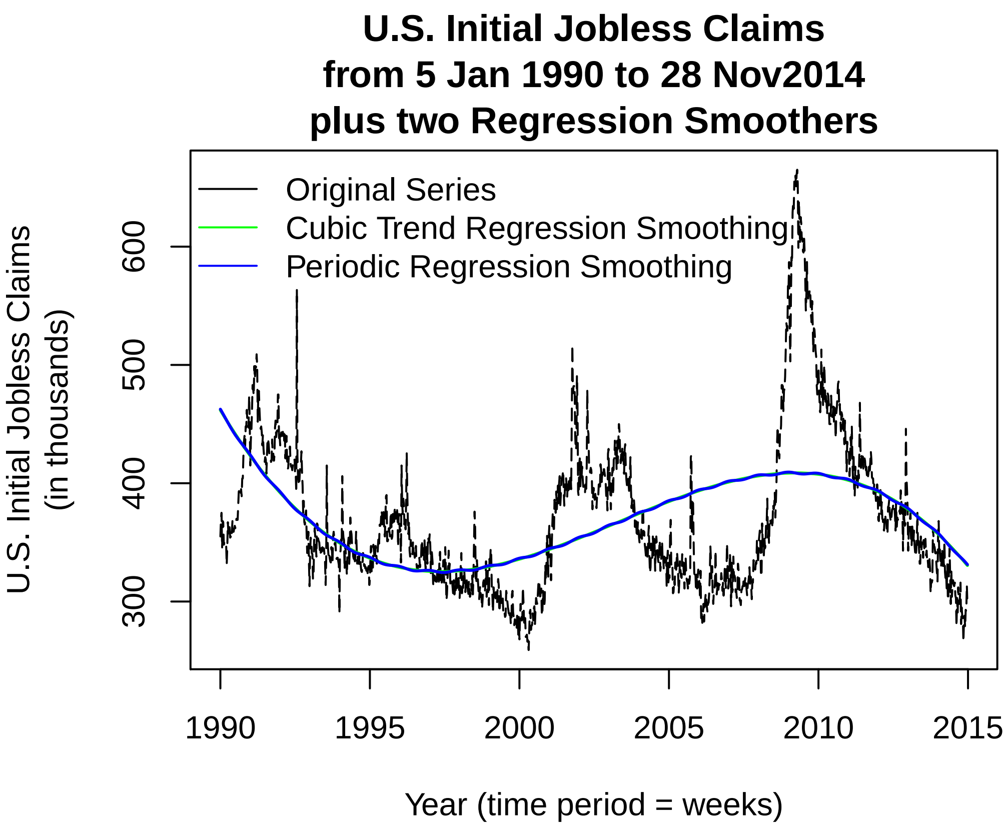


Figure 11: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014 (original series plus two Regression Smoothers)

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.**

```
INJCJC.1 <- ksmooth(INJCJC.time, INJCJC, "normal", bandwidth = 5/52)
INJCJC.2 <- ksmooth(INJCJC.time, INJCJC, "normal", bandwidth = 2)
```

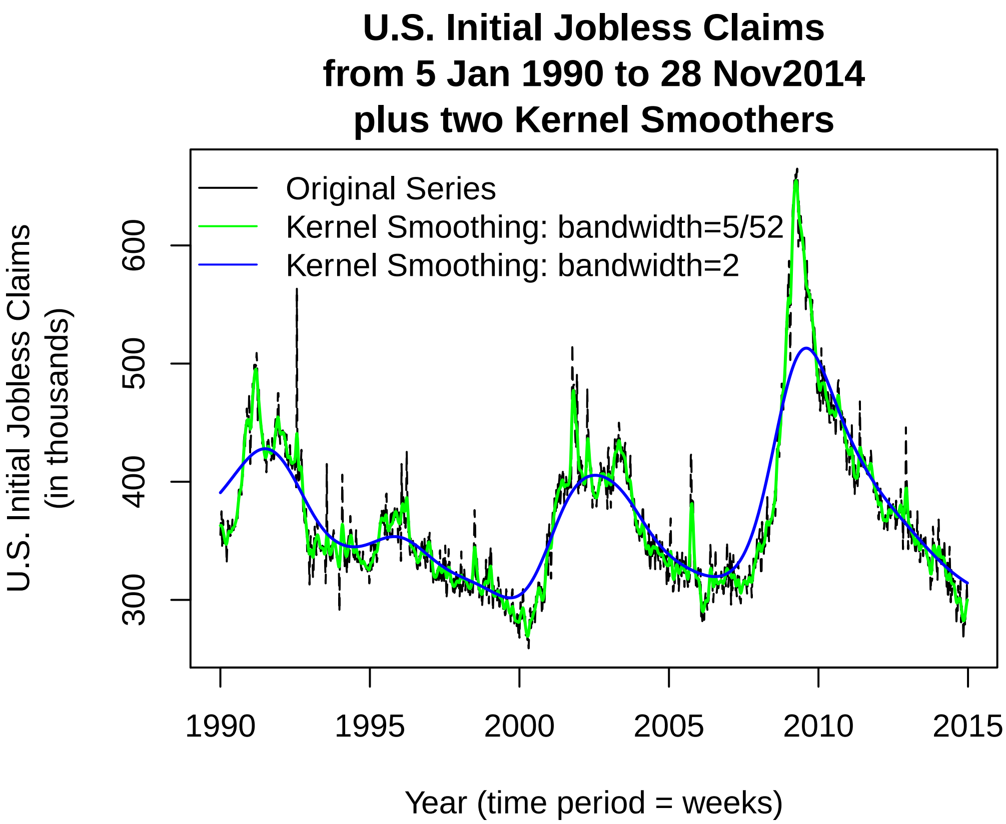


Figure 12: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014 (original series plus two Regression Smoothers)

4. **Generate two nearest neighborhood smoothers. Choose the smoothing parameters 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.**

```
INJCJC.1 <- supsmu(INJCJC.time, INJCJC, span = .01)
INJCJC.2 <- supsmu(INJCJC.time, INJCJC, span = .1)
```

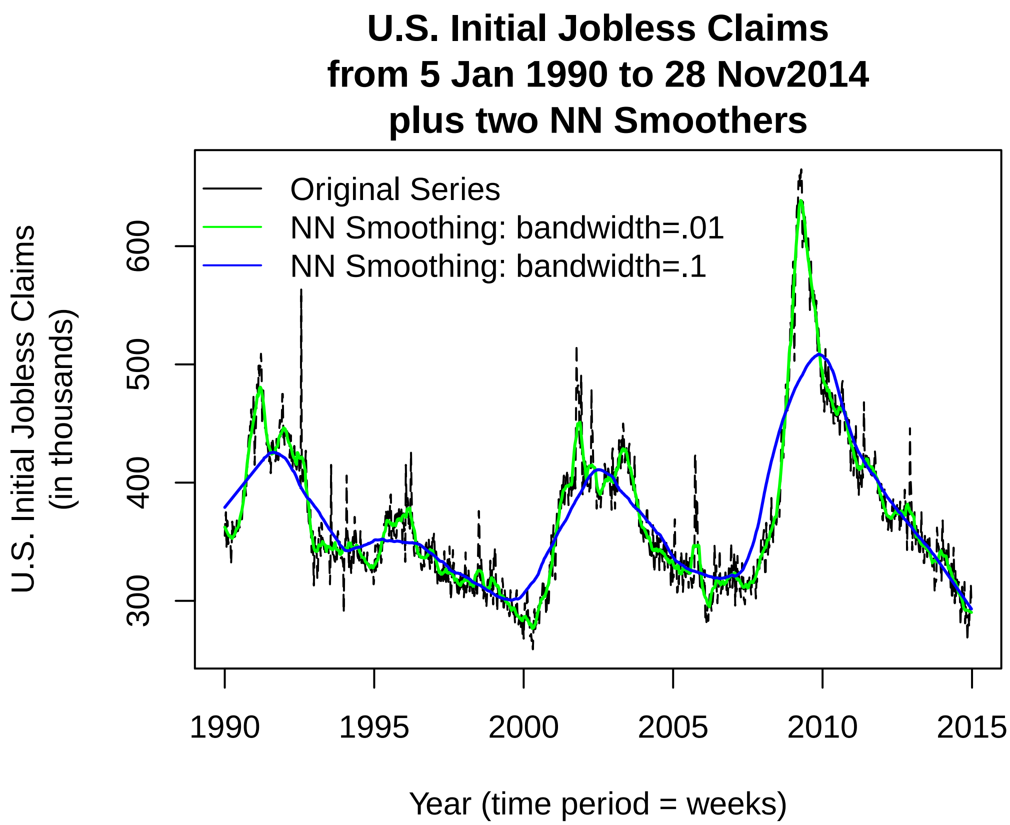


Figure 13: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014 (original series plus two Nearest Neighborhood Smoothers)

5. **Generate two LOWESS smoothers. Choose the smoothing parameters 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.**

```
INJCJC.1 <- lowess(INJCJC, f = .02)
INJCJC.2 <- lowess(INJCJC, f = .2)
```

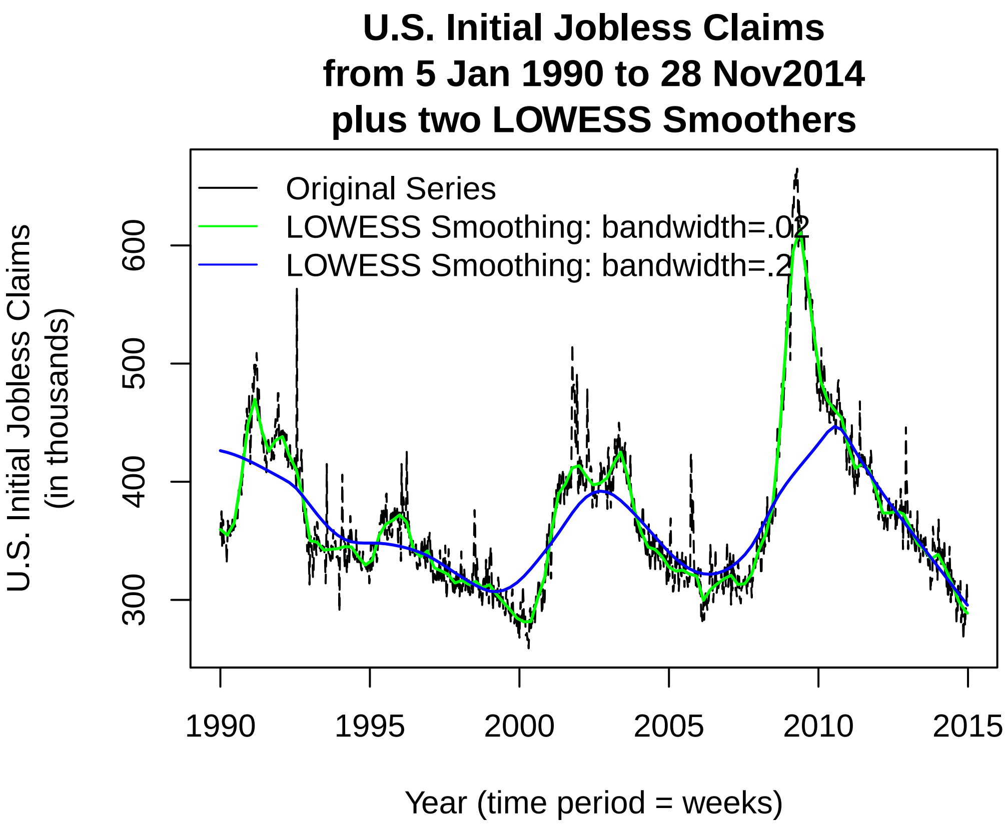


Figure 14: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014 (original series plus two LOWESS Smoothers)

6. **Generate two spline smoothers. Choose the smoothing parameters 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.**

```
INJCJC.1 <- smooth.spline(INJCJC.time, INJCJC, spar = 0.05)
INJCJC.2 <- smooth.spline(INJCJC.time, INJCJC, spar = 0.8)
```

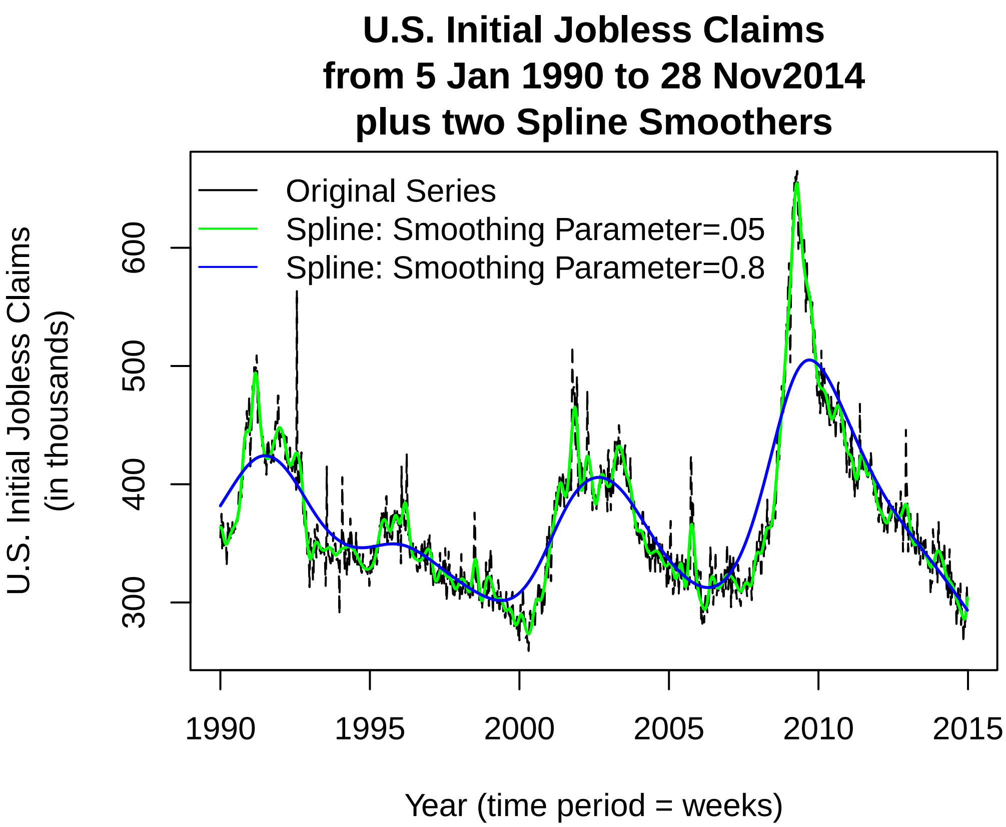


Figure 15: U.S. Initial Jobless Claims (in thousands) from 5 Jan 1990 to 28 Nov 2014 (original series plus two Spline Smoothers)