Homework6

Megan Jasek, Rohan Thakur, Charles Kekeh

Tuesday, March 15, 2016

Exercise 1

part a

Mean Function A time series defined as an observation of a stochastic process resulting in a set of variables $x_1, x_2, \cdots, x_n$ is defined by a joint distribution function $F(c_1, c_2, \cdots, c_n) = P(x_{i1} \leq c_1, x_{i2} \leq c_2, \cdots, x_{in} \leq c_n)$
Assuming knowledge of such a joint probability distribution, we would derive the marginal probability distributions $f_t(x_t)$
And from such marginal probability distributions, we define the mean function:

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

This mean function is different from the mean function of observations of a single random variable, as seen with with the classical linear model.
For time series, the observation of x_t is dependent on previous observations of $x_{t-1}, x_{t-2}, \ldots$. That dependency is captured in the joint probability distribution which is unavailable to us, as the time series represents the single instance of the realization of the stochastic process that we are able to observe.

Variance Function For time series defined as described in the mean functionm discussion above, the variance function, a function of time t, is defined as:

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

Where $f_t(x_t)$ is the marginal probability distribution of x_t in the stochastic process.
This variance function is also different from the variance of the observations of a single random variable studied with classical linear models, because of the dependency of x_t over $x_{t-1}, x_{t-2}, \ldots$ as expressed in the joint probability distribution.

part b

The assumption of strict stationarity is very strong strong assumption of stationarity.
For a given time series, we say that it is **strictly stationary** is its distribution is unchanged for any time shift. i.e. given a joint distribution $F(x_{t1}, x_{t2}, \cdots, x_{tn})$ as introduced earlier, a time series x_t is strictly stationary if $F(x_{t1}, x_{t2}, \cdots, x_{tn}) = F(x_{t1+m}, x_{t2+m}, \cdots, x_{tn+m}), \forall t_1, \cdots, t_n$ and m

The assumption of **weak stationarity** (or second order stationarity) is a weaker assumption of stationarity. A time series x_t is weak stationary if its mean and variance are stationary and its auto-covariance $Cov(x_t, x_{t+k})$ depends only on the lag k, and is not a function of time t.
The auto-covariance of a time series that is only dependent of lag k is defined as:

$$\gamma_k = E[(x_t - \mu)(x_{t+k} - \mu)]$$

where μ is the stationary mean of the time series.

Exercise 2

Part a

```
rw.wod <- white.noise <- rnorm(500)
for (t in 2:length(rw.wod)) {
    rw.wod[t] <- rw.wod[t - 1] + white.noise[t]


    # From Megan: Random walk uses cumsum.  Above is an AR(1) model
    white.noise <- rnorm(500)
    rw.wod = cumsum(white.noise)

}
```

Part b

Mean of time series - The mean of the time series is: 9.838132
- The standard deviation of the time series is: 4.723985
- The 25th, 50th and 75th quantiles of the time series are: 6.187022 9.525720 13.694139

- The mean of the time series is: 3.750754
- The standard deviation of the time series is: 4.907826
- The 25th, 50th and 75th quantiles of the time series are: -0.1391342 3.8627068 8.1896239
- The minimum of the time series is: -8.7110
- The maximum of the time series is: 13.7200

```
mean(rw.wod)
```

```
## [1] 12.1471
```

```
sd(rw.wod)
```

```
## [1] 7.146717
```

```
quantile(rw.wod)
```

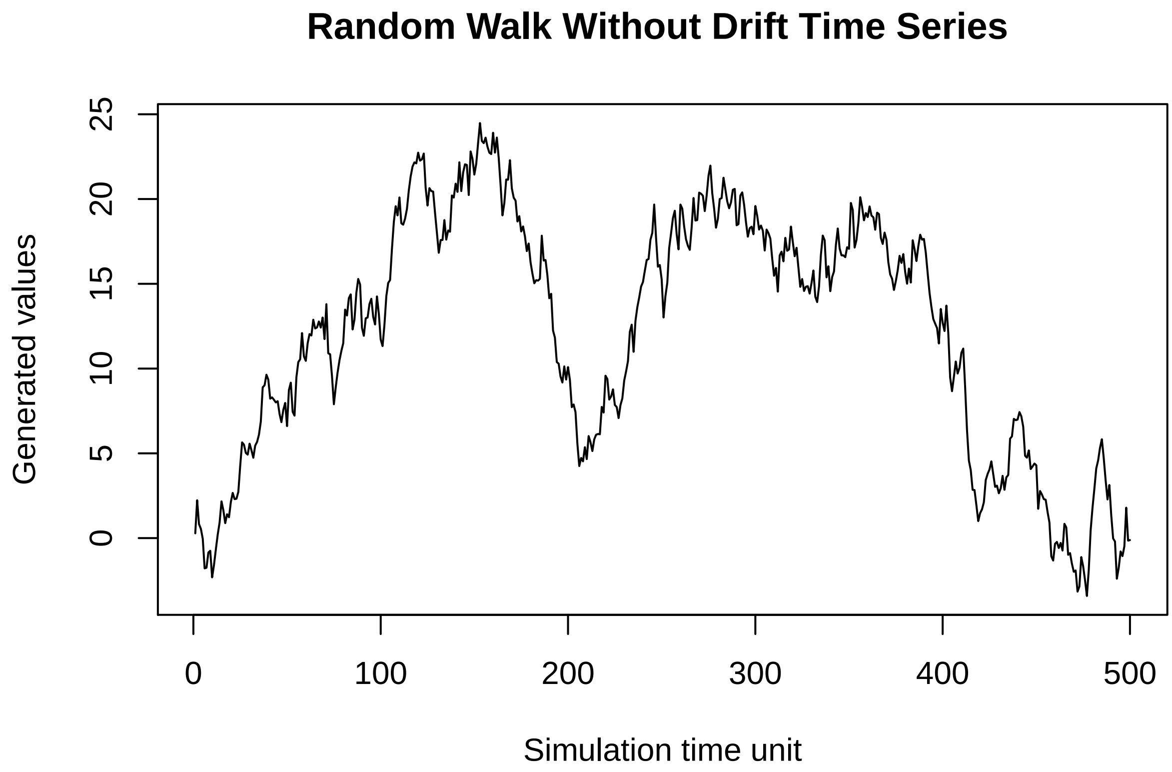
```
##        0%       25%       50%       75%      100%
## -3.410292  6.079161 14.019764 18.033968 24.481549
```

```
describe(rw.wod)
```

```
##    vars   n  mean   sd median trimmed  mad   min   max range  skew kurtosis
## 1     1 500 12.15 7.15  14.02   12.54 7.65 -3.41 24.48 27.89 -0.42    -1.02
##      se
## 1 0.32
```

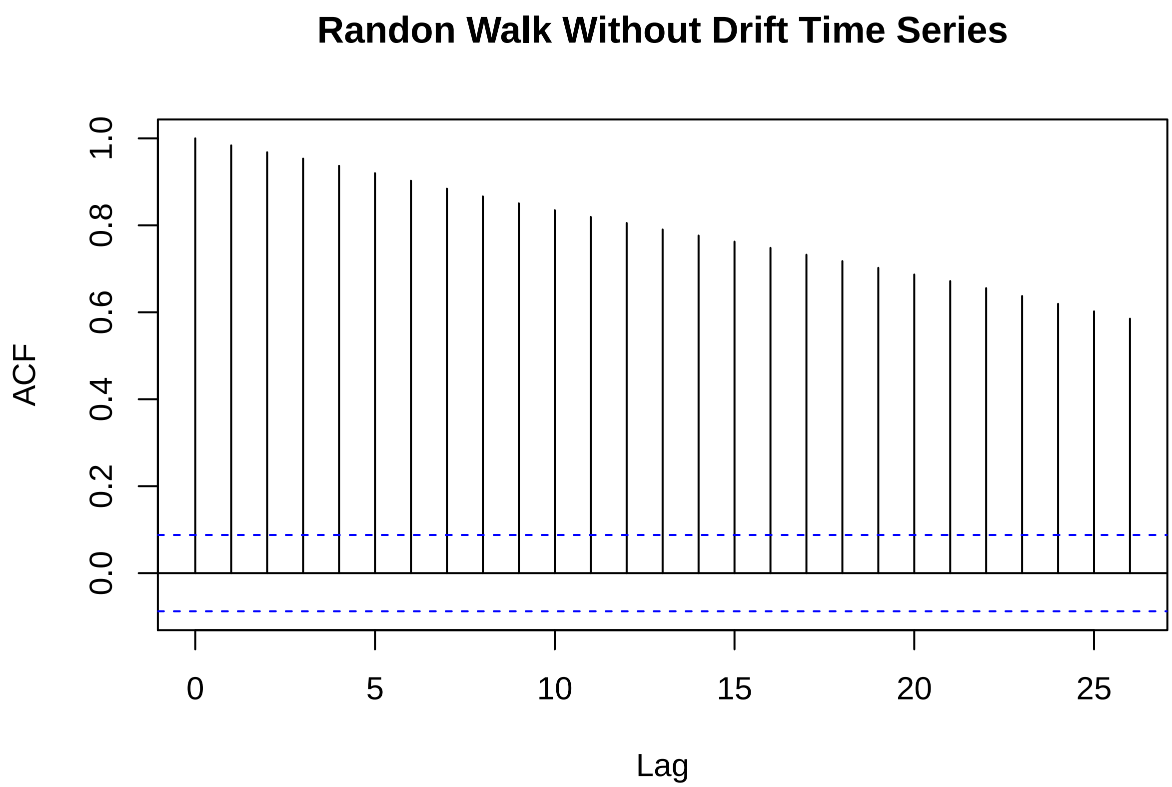
Part c

```
plot.ts(rw.wod, xlab = "Simulation time unit", ylab = "Generated values",
    main = "Random Walk Without Drift Time Series ")
```



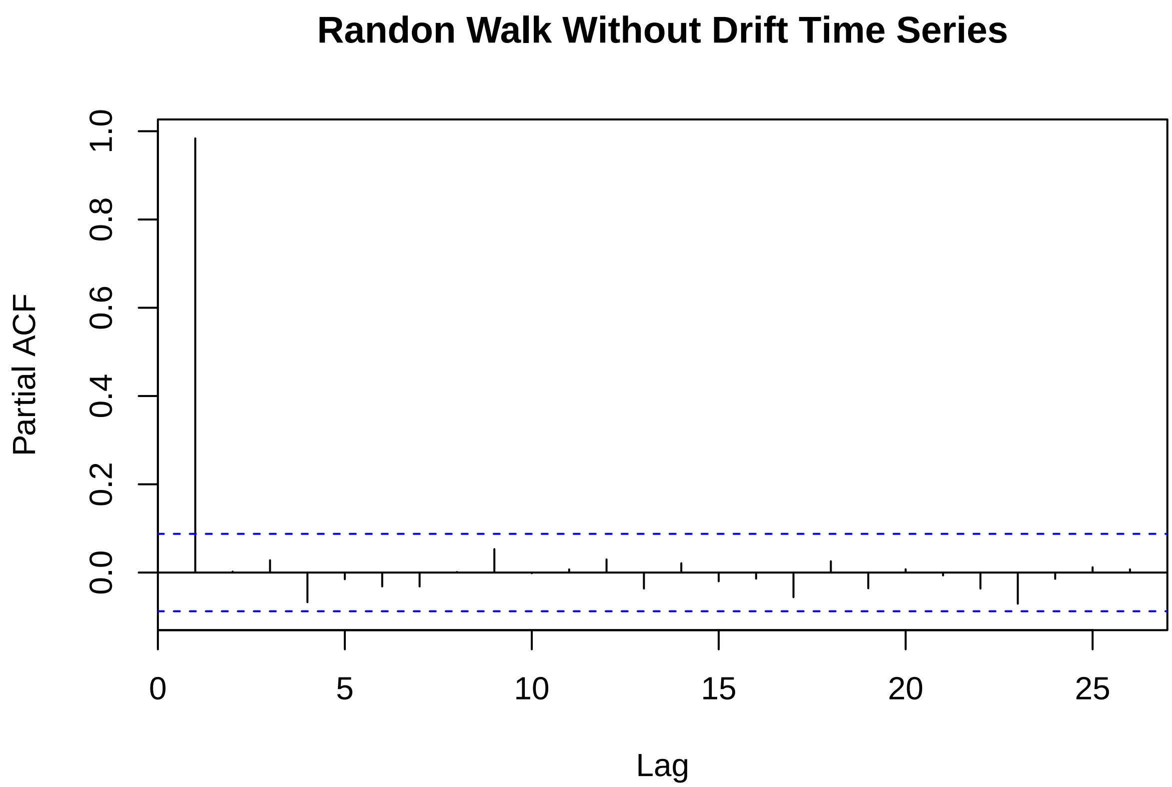
Part d

```
acf(ts(rw.wod), main = "Randon Walk Without Drift Time Series")
```



Part e

```
pacf(ts(rw.wod), main = "Randon Walk Without Drift Time Series")
```



Exercise 3

Part a

```
rw.wid <- white.noise
for (t in 2:length(rw.wid)) {
    rw.wid[t] <- rw.wid[t - 1] + 0.5 + white.noise[t]
}

# Megan's random walk with drift
w1 = white.noise + 0.5
rw.wid = cumsum(w1)
```

Mean of time series - The mean of the time series is: 134.5881
- The standard deviation of the time series is: 74.88504
- The 25th, 50th and 75th quantiles of the time series are: 76.3171217 130.3604551 199.4784637

- The mean of the time series is: 132.693
- The standard deviation of the time series is: 75.42878
- The 25th, 50th and 75th quantiles of the time series are: 68.187578 125.052338 204.076424
- The minimum of the time series is: -1.102

- The maximum of the time series is: 259.700

```r
mean(rw.wid)
```

```
## [1] 137.3971
```

```r
sd(rw.wid)
```

```
## [1] 71.1683
```

```r
quantile(rw.wid)
```

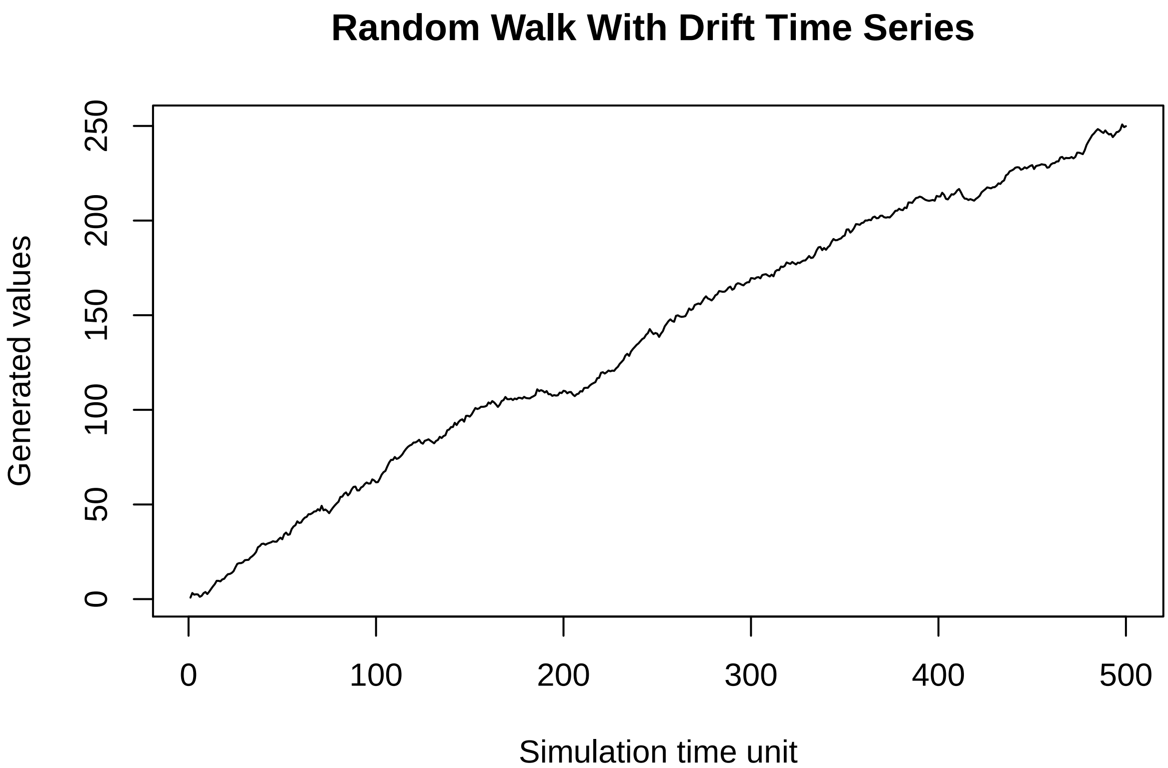
```
##          0%         25%         50%         75%        100%
##   0.7819444  83.2606888 140.9049757 202.9413390 250.7949307
```

```r
describe(rw.wid)
```

```
##   vars   n  mean    sd median trimmed  mad  min    max  range skew
## 1    1 500 137.4 71.17  140.9  139.89 89.8 0.78 250.79 250.01 -0.2
##   kurtosis   se
## 1    -1.15 3.18
```

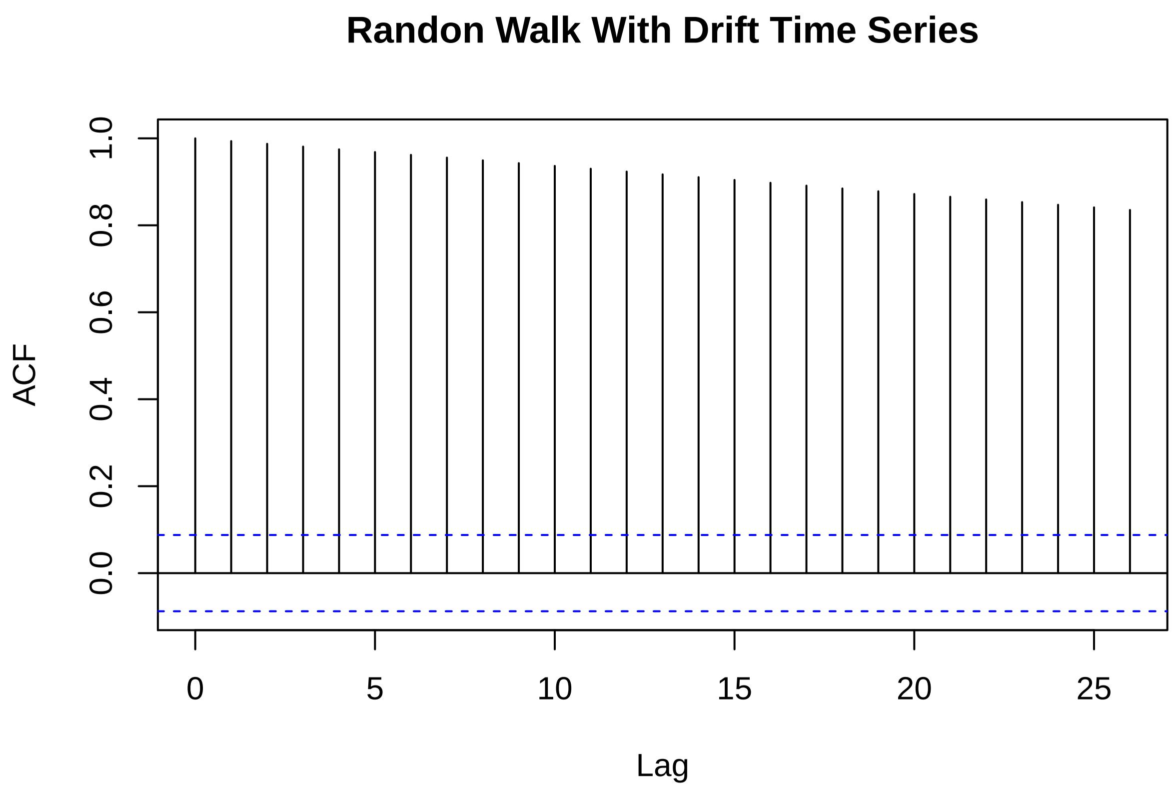
Part c

```r
plot.ts(rw.wid, xlab = "Simulation time unit", ylab = "Generated values",
    main = "Random Walk With Drift Time Series ")
```



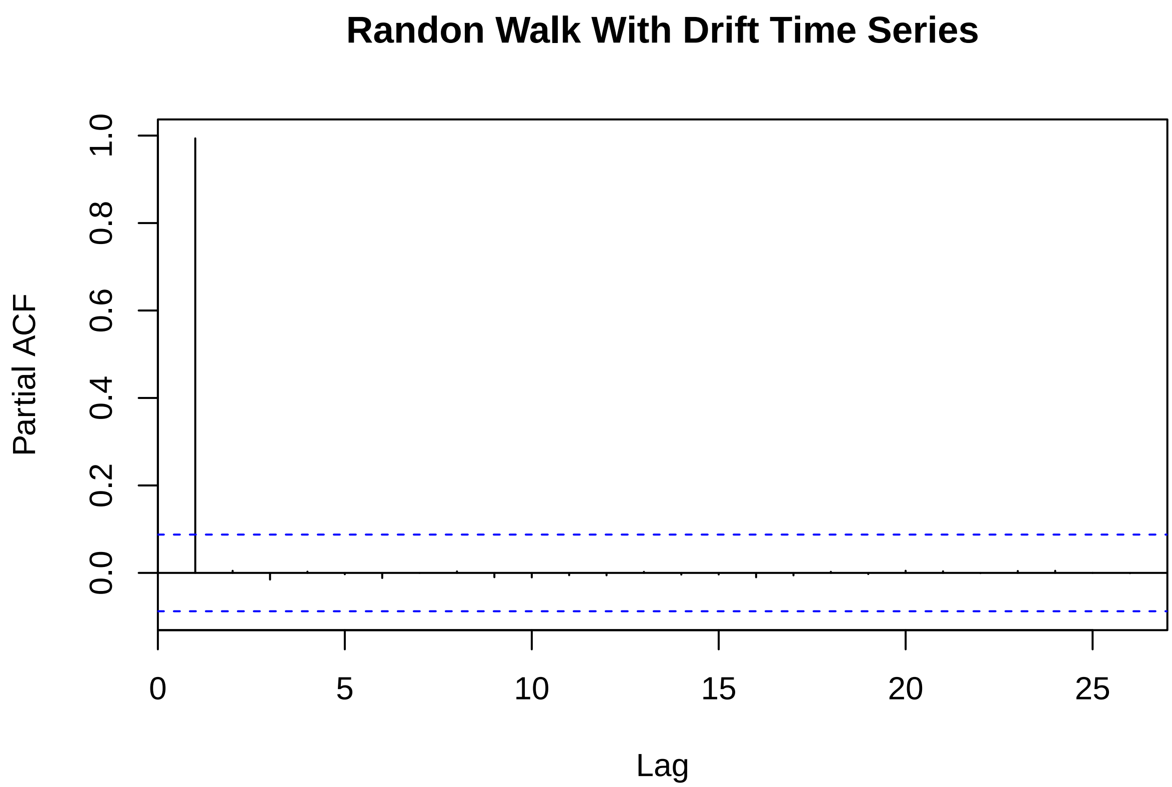
Part d

```r
acf(ts(rw.wid), main = "Randon Walk With Drift Time Series")
```



Part e

```r
pacf(ts(rw.wid), main = "Randon Walk With Drift Time Series")
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Exercise 4

Part a

```
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
```

Part b

```
data.ts <- ts(data$INJCJC, frequency = 52, start = c(1990, 1), end = c(2014,
    52))
summary(data.ts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   259.0   324.0   353.5   371.1   406.0   665.0
```

```
quantile(data.ts)
```

```
##    0%   25%   50%   75%  100%
## 259.0 324.0 353.5 406.0 665.0
```

Part c

```
INJCJC.time <- time(data.ts)
```

Part d

```
head(cbind(INJCJC.time, data.ts), 5)
```

```
##      INJCJC.time data.ts
## [1,]    1990.000     355
## [2,]    1990.019     369
## [3,]    1990.038     375
## [4,]    1990.058     345
## [5,]    1990.077     368
```

```r
head(cbind(INJCJC.time, data.ts), 10)
```

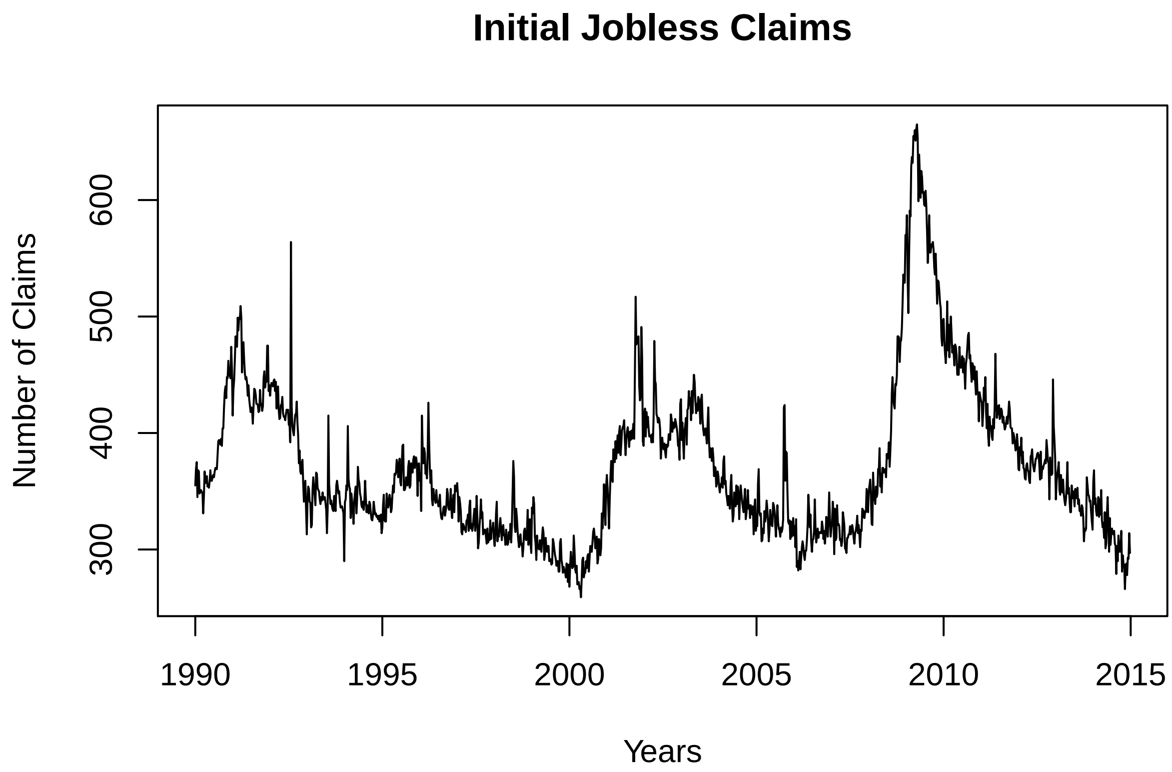
```
##       INJCJC.time data.ts
##  [1,]    1990.000     355
##  [2,]    1990.019     369
##  [3,]    1990.038     375
##  [4,]    1990.058     345
##  [5,]    1990.077     368
##  [6,]    1990.096     367
##  [7,]    1990.115     348
##  [8,]    1990.135     350
##  [9,]    1990.154     351
## [10,]    1990.173     349
```

```r
head(cbind(INJCJC.time, data.ts), 12)
```

```
##       INJCJC.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
```

Part e1

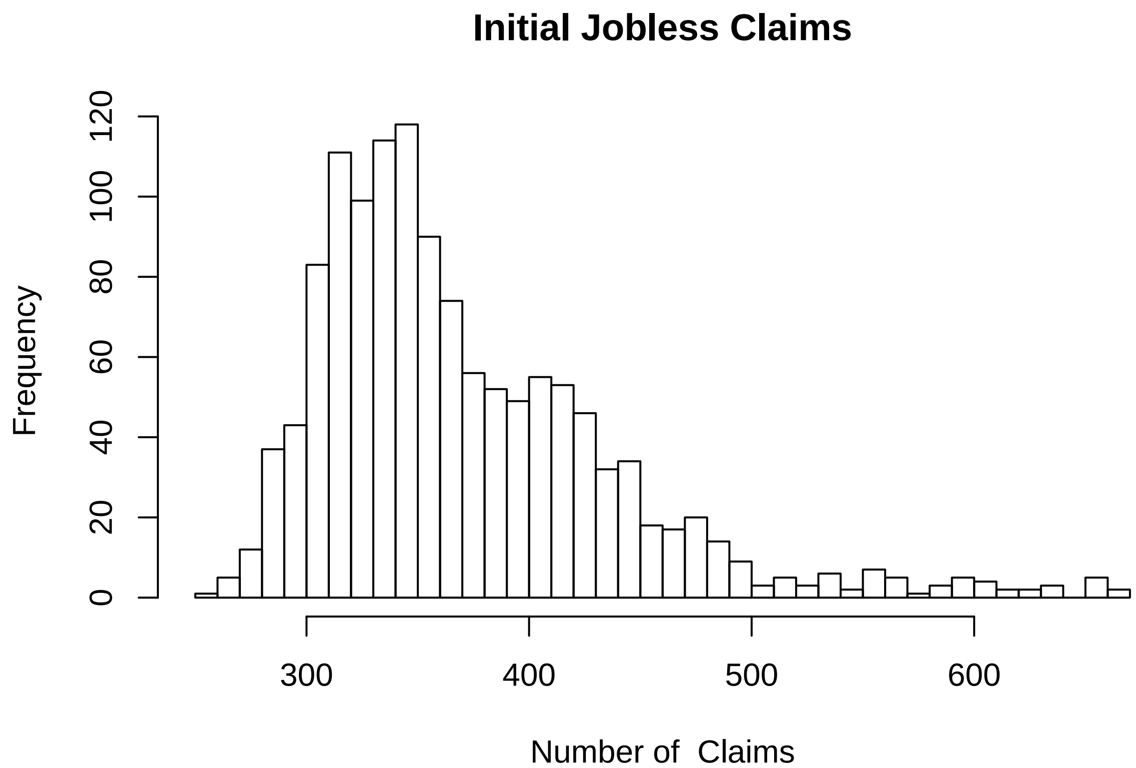
```r
plot.ts(data.ts, xlab = "Years", ylab = "Number of Claims", main = "Initial Jobless Claims")
```



Part e2

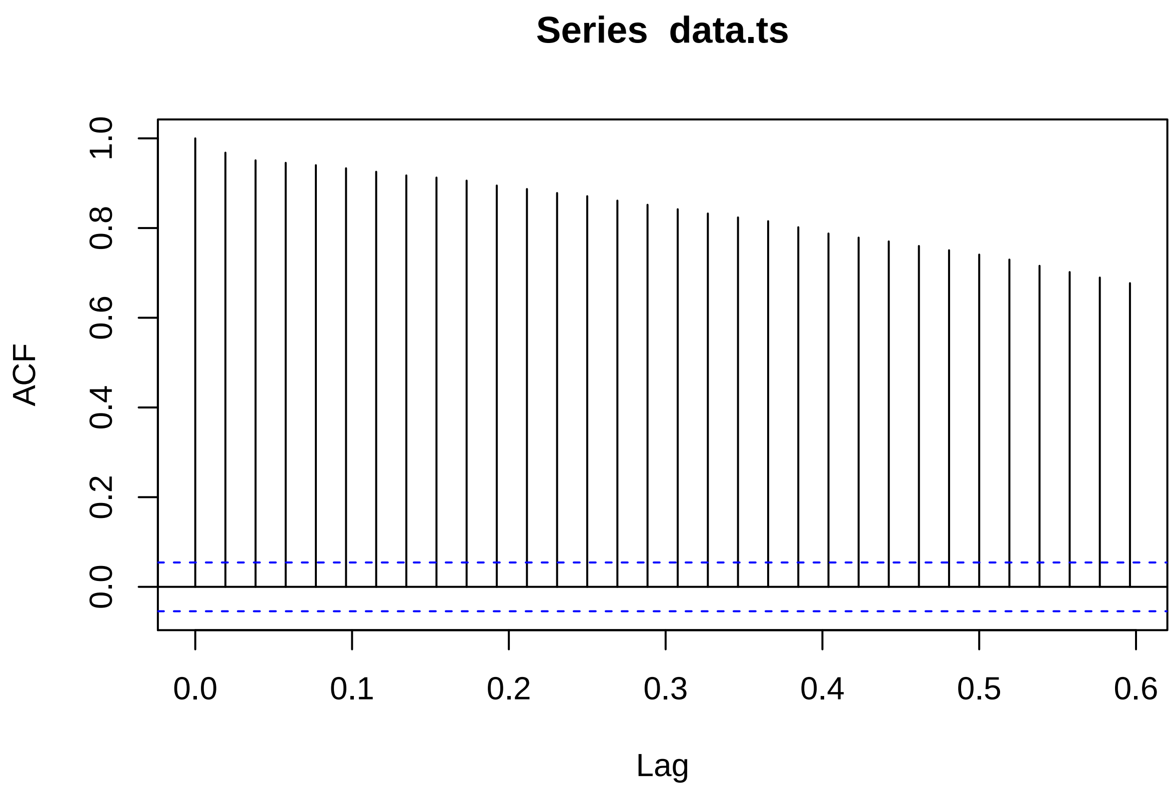
What the histogram doesn't show is how the values in the distribution occur over time and the dependencies between the values. It does show the distribution of the values. The number of bins is selected based on the representation that provides a more visually complete rendering of the distribution of the values of the time series. The range the values is considered and then an appropriate granularity is chosen based on how many different values occur within the range.

```
hist(data.ts, xlab = "Number of  Claims", main = "Initial Jobless Claims",
    breaks = 30)
```



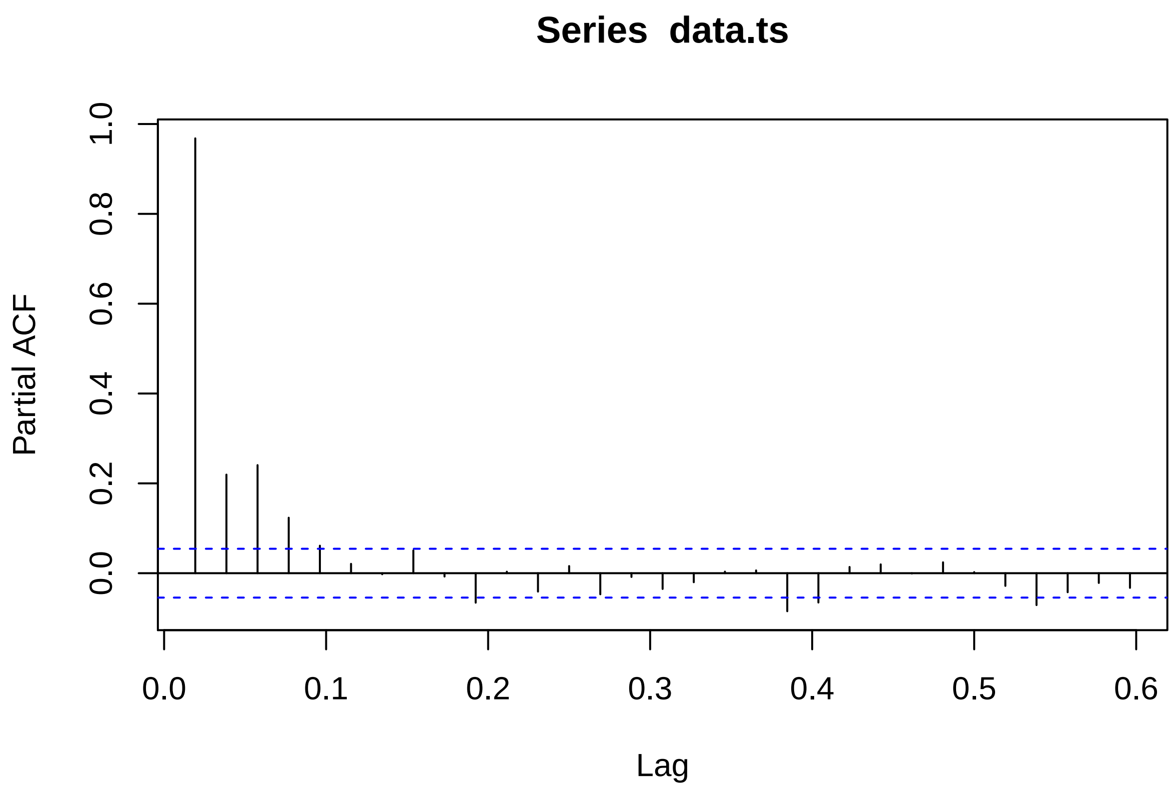
Part e3

```
acf(data.ts)
```



Part e4

```
pacf(data.ts)
```



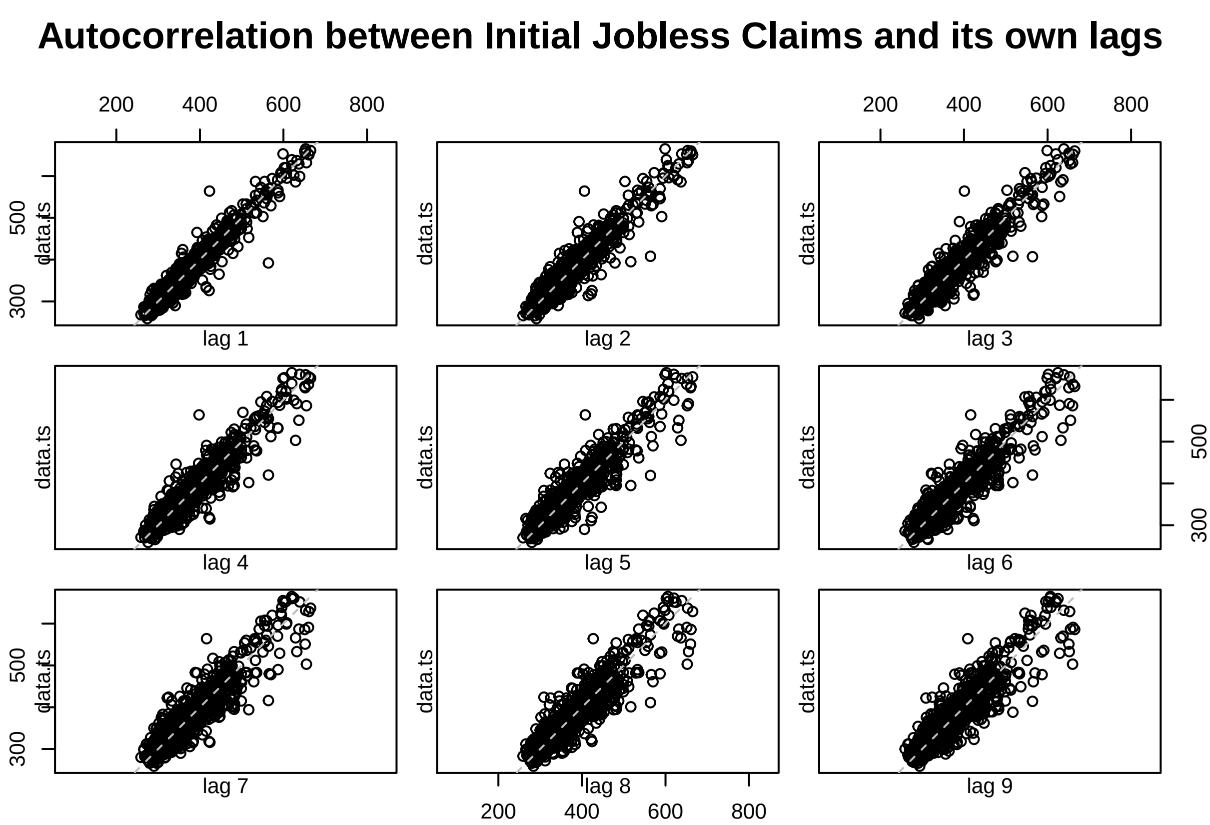
Part e5

```r
lag.plot(data.ts, lags = 9, layout = c(3, 3), diag = TRUE, disg.col = "red",
    main = "Autocorrelation between Initial Jobless Claims and its own lags")
```

```
## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter

## Warning in title(...): "disg.col" is not a graphical parameter

## Warning in box(...): "disg.col" is not a graphical parameter

## Warning in axis(side, ..., xpd = NA): "disg.col" is not a graphical
## parameter

## Warning in axis(side, ..., xpd = NA): "disg.col" is not a graphical
## parameter

## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter
```

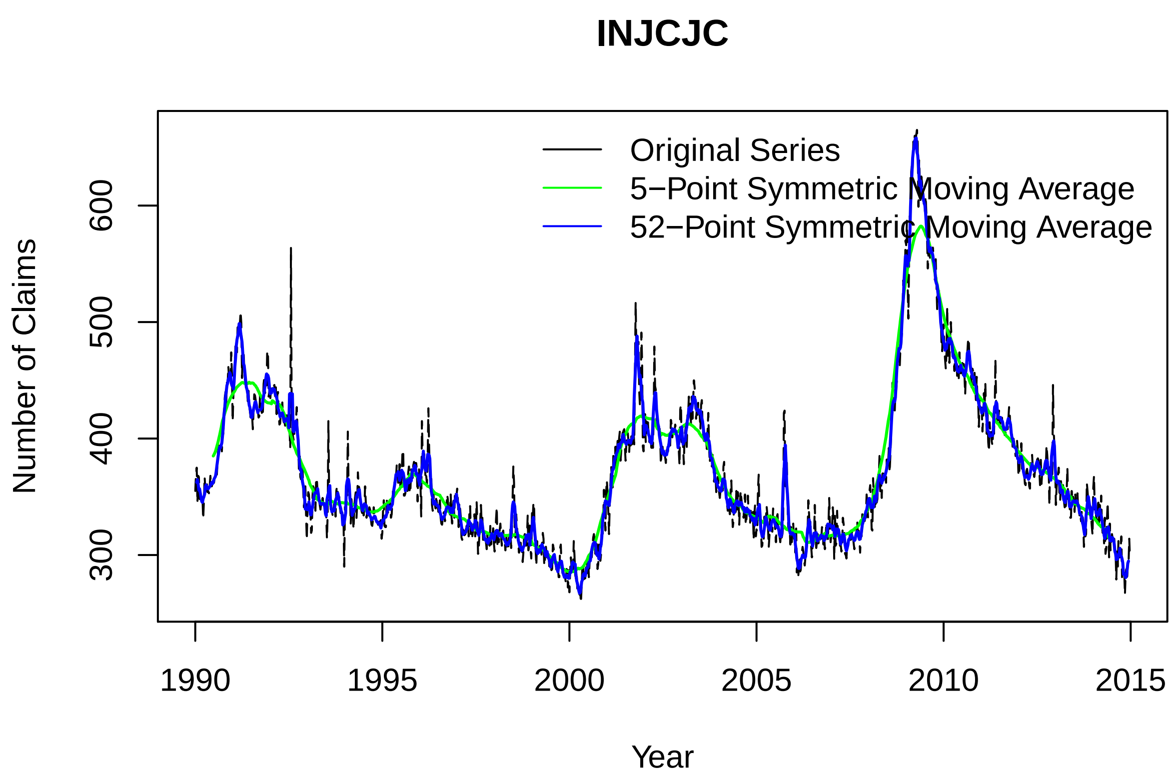
```
## Warning in title(...): "disg.col" is not a graphical parameter

## Warning in box(...): "disg.col" is not a graphical parameter

## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter

## Warning in title(...): "disg.col" is not a graphical parameter

## Warning in box(...): "disg.col" is not a graphical parameter

## Warning in axis(side, ..., xpd = NA): "disg.col" is not a graphical
## parameter

## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter

## Warning in title(...): "disg.col" is not a graphical parameter

## Warning in box(...): "disg.col" is not a graphical parameter

## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter

## Warning in title(...): "disg.col" is not a graphical parameter

## Warning in box(...): "disg.col" is not a graphical parameter

## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter

## Warning in title(...): "disg.col" is not a graphical parameter

## Warning in box(...): "disg.col" is not a graphical parameter

## Warning in axis(side, ..., xpd = NA): "disg.col" is not a graphical
## parameter

## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter

## Warning in title(...): "disg.col" is not a graphical parameter
```

```
## Warning in box(...): "disg.col" is not a graphical parameter

## Warning in axis(side, ..., xpd = NA): "disg.col" is not a graphical
## parameter

## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter

## Warning in title(...): "disg.col" is not a graphical parameter

## Warning in box(...): "disg.col" is not a graphical parameter

## Warning in axis(side, ..., xpd = NA): "disg.col" is not a graphical
## parameter

## Warning in plot.window(...): "disg.col" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "disg.col" is not a graphical parameter

## Warning in title(...): "disg.col" is not a graphical parameter

## Warning in box(...): "disg.col" is not a graphical parameter
```



Part f1

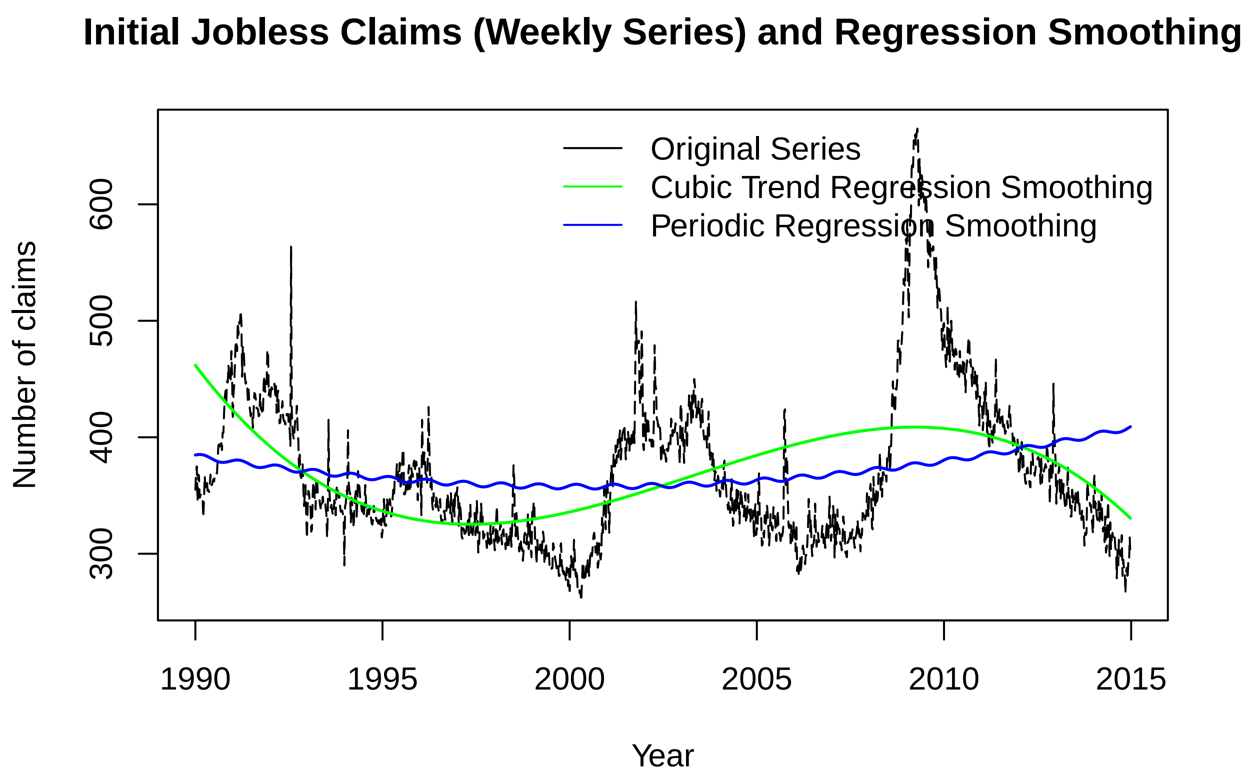
```
# Create a 52 weeks and a 5 weeks moving average smoother
mo52 = filter(data.ts, sides = 2, rep(1, 52)/52)
mo5 = filter(data.ts, sides = 2, rep(1, 5)/5)

plot(data.ts, main = "INJCJC", pch = 4, lty = 5, lwd = 1, xlab = "Year",
    ylab = "Number of Claims")
lines(mo52, lty = 1, lwd = 1.5, col = "green")
lines(mo5, lty = 1, lwd = 1.5, col = "blue")
# Add Legend
leg.txt <- c("Original Series", "5-Point Symmetric Moving Average",
    "52-Point Symmetric Moving Average")
legend("topright", legend = leg.txt, lty = c(1, 1, 1), col = c("black",
    "green", "blue"), bty = "n", cex = 1, merge = TRUE, bg = 336)
```



Part f2

```
wk = time(data.ts) - mean(time(data.ts))
wk2 = wk^2
wk3 = wk^3
cs = cos(2 * pi * wk)
```

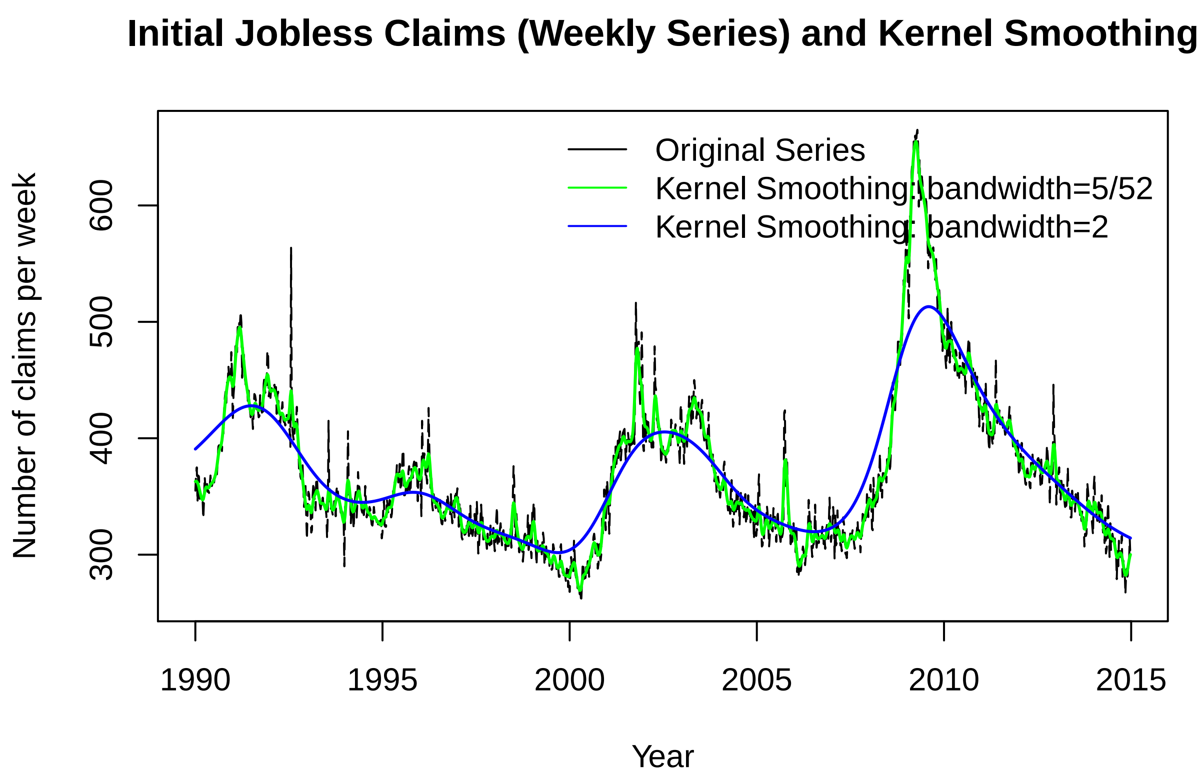
```r
sn = sin(2 * pi * wk)
reg1 = lm(data.ts ~ wk + wk2 + wk3, na.action = NULL)
reg2 = lm(data.ts ~ wk + wk2 + cs + sn, na.action = NULL)
plot(data.ts, main = "Initial Jobless Claims (Weekly Series) and Regression Smoothing",
    pch = 4, lty = 5, lwd = 1, xlab = "Year", ylab = "Number of claims")
lines(fitted(reg1), 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("topright", legend = leg.txt, lty = c(1, 1, 1), col = c("black",
    "green", "blue"), bty = "n", cex = 1, merge = TRUE, bg = 336)
```



Part f3

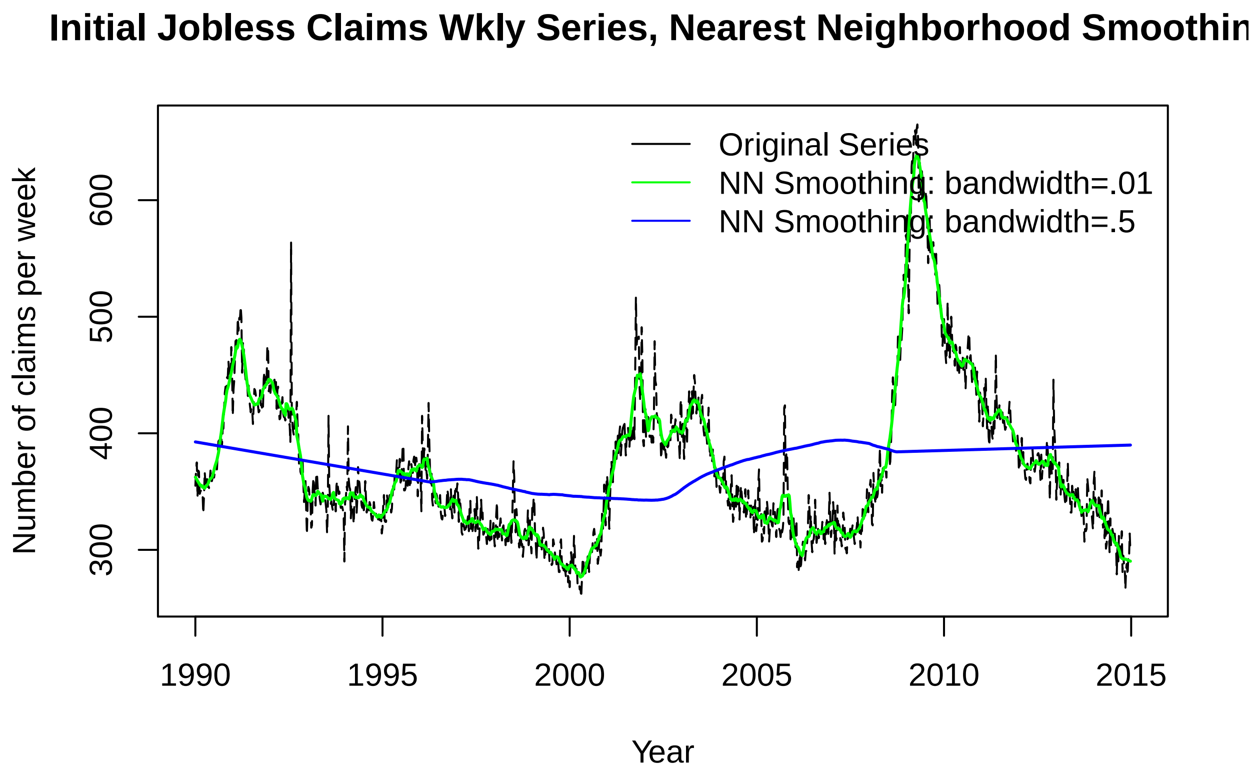
```r
plot(data.ts, main = "Initial Jobless Claims (Weekly Series) and Kernel Smoothing",
    pch = 4, lty = 5, lwd = 1, xlab = "Year", ylab = "Number of claims per week")
lines(ksmooth(time(data.ts), data.ts, "normal", bandwidth = 5/52),
    lty = 1, lwd = 1.5, col = "green")
lines(ksmooth(time(data.ts), data.ts, "normal", bandwidth = 2),
    lty = 1, lwd = 1.5, col = "blue")
# Add Legend
leg.txt <- c("Original Series", "Kernel Smoothing: bandwidth=5/52",
```

```
    "Kernel Smoothing: bandwidth=2")
legend("topright", legend = leg.txt, lty = c(1, 1, 1), col = c("black",
    "green", "blue"), bty = "n", cex = 1, merge = TRUE, bg = 336)
```



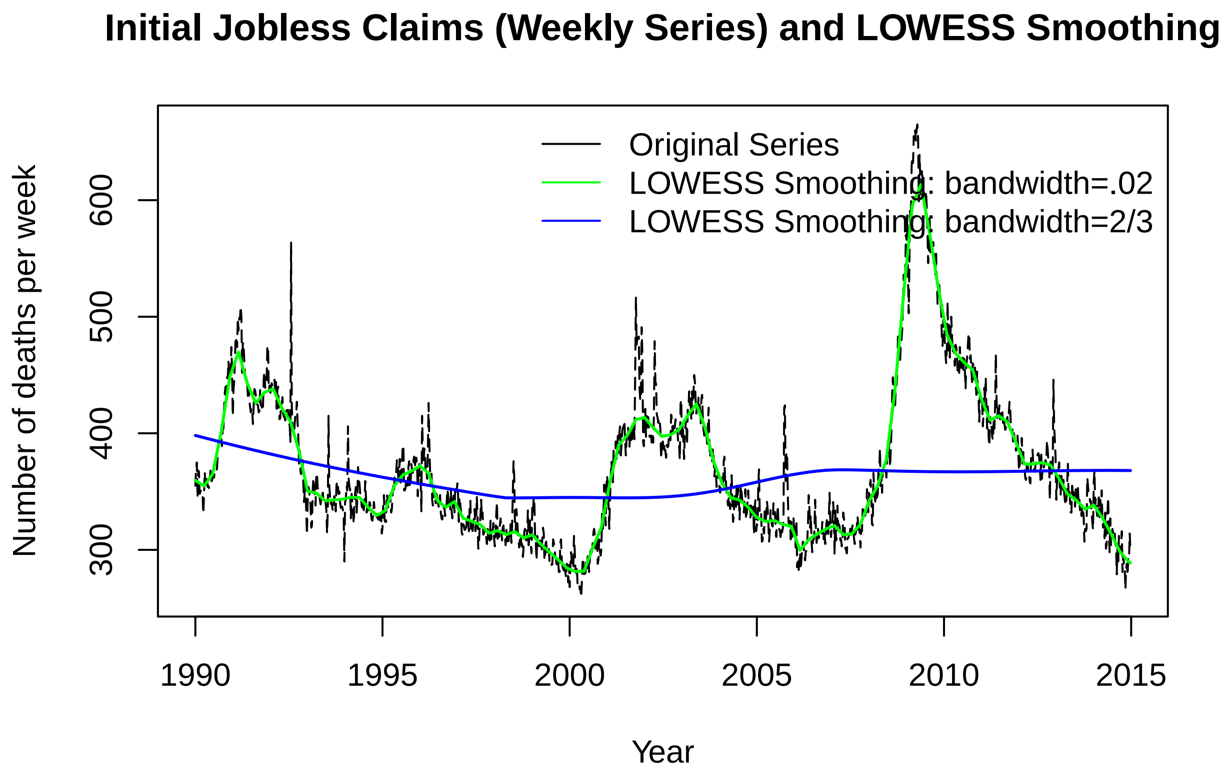
Part f4

```
plot(data.ts, main = "Initial Jobless Claims Wkly Series, Nearest Neighborhood Smoothing",
    pch = 4, lty = 5, lwd = 1, xlab = "Year", ylab = "Number of claims per week")
lines(supsmu(time(data.ts), data.ts, span = 0.01), lty = 1, lwd = 1.5,
    col = "green")
lines(supsmu(time(data.ts), data.ts, span = 0.5), lty = 1, lwd = 1.5,
    col = "blue")
# Add Legend
leg.txt <- c("Original Series", "NN Smoothing: bandwidth=.01",
    "NN Smoothing: bandwidth=.5")
legend("topright", legend = leg.txt, lty = c(1, 1, 1), col = c("black",
    "green", "blue"), bty = "n", cex = 1, merge = TRUE, bg = 336)
```



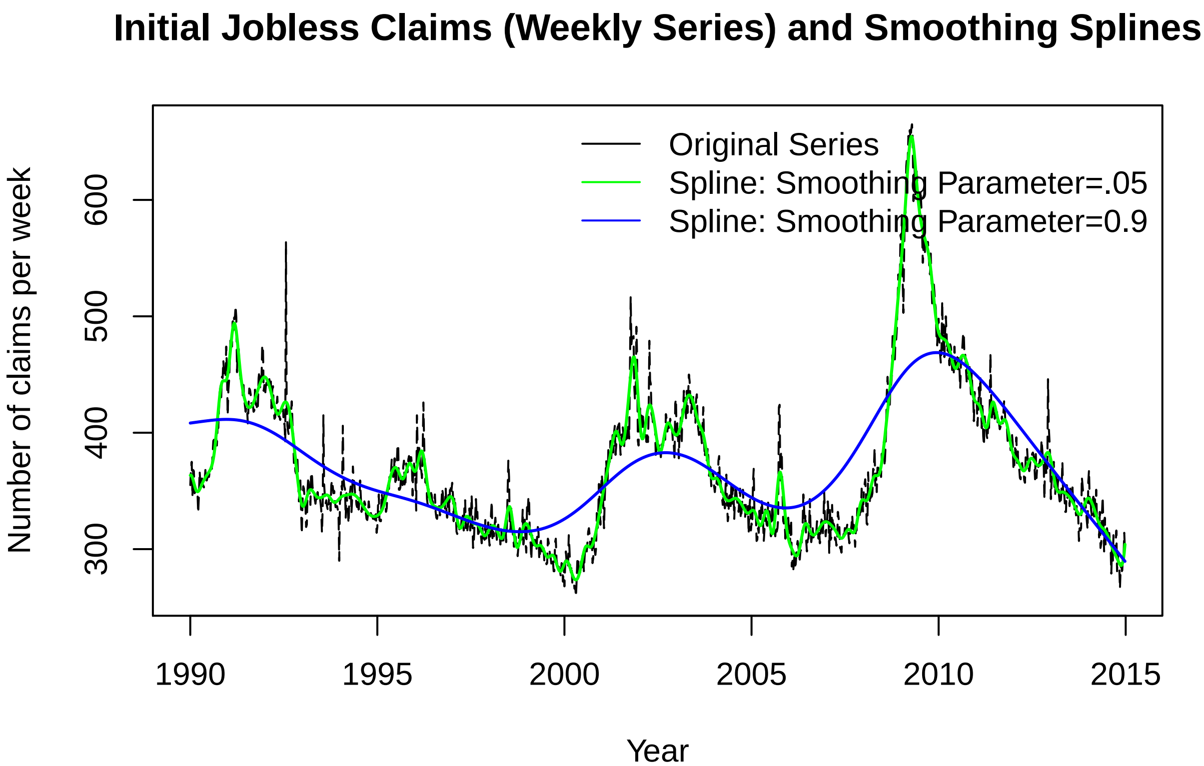
Part f5

```
plot(data.ts, main = "Initial Jobless Claims (Weekly Series) and LOWESS Smoothing",
    pch = 4, lty = 5, lwd = 1, xlab = "Year", ylab = "Number of deaths per week")
lines(lowess(data.ts, f = 0.02), lty = 1, lwd = 1.5, col = "green")
lines(lowess(data.ts, f = 2/3), lty = 1, lwd = 1.5, col = "blue")
# Add Legend
leg.txt <- c("Original Series", "LOWESS Smoothing: bandwidth=.02",
    "LOWESS Smoothing: bandwidth=2/3")
legend("topright", legend = leg.txt, lty = c(1, 1, 1), col = c("black",
    "green", "blue"), bty = "n", cex = 1, merge = TRUE, bg = 336)
```



Part f6

```
plot(data.ts, main = "Initial Jobless Claims (Weekly Series) and Smoothing Splines",
    pch = 4, lty = 5, lwd = 1, xlab = "Year", ylab = "Number of claims per week")
lines(smooth.spline(time(data.ts), data.ts, spar = 0.05), lty = 1,
    lwd = 1.5, col = "green")
lines(smooth.spline(time(data.ts), data.ts, spar = 0.9), lty = 1,
    lwd = 1.5, col = "blue")
# Add Legend
leg.txt <- c("Original Series", "Spline: Smoothing Parameter=.05",
    "Spline: Smoothing Parameter=0.9")
legend("topright", legend = leg.txt, lty = c(1, 1, 1), col = c("black",
    "green", "blue"), bty = "n", cex = 1, merge = TRUE, bg = 336)
```



Initial Jobless Claims (Weekly Series) and Smoothing Splines
Original Series
Spline: Smoothing Parameter=.05
Spline: Smoothing Parameter=0.9
Number of claims per week
300
400
500
600
1990
1995
2000
2005
2010
2015
Year