Stats 207: HW1

Problem 1

First, read in the data from trends.google.com for IBM (2004-Now):

```
df = data.frame(read.csv("./IBM.csv", header = TRUE))
df[,'value'] = as.integer(df[,'value'])
df[,'date'] = parse_date_time(df[,'date'], "ym")
str(df)
```

```
## 'data.frame':    196 obs. of  2 variables:
##  $ date : POSIXct, format: "2004-01-01" "2004-02-01" ...
##  $ value: int  91 100 91 91 87 90 91 95 92 84 ...
```

Part a

Consider fitting a cubic polynomial by the least squares method. This is fitting a model of the form:

$$y_t = \beta_0 + x_t\beta_1 + x_t^2\beta_2 + x_t^3\beta_3 + \epsilon_t$$

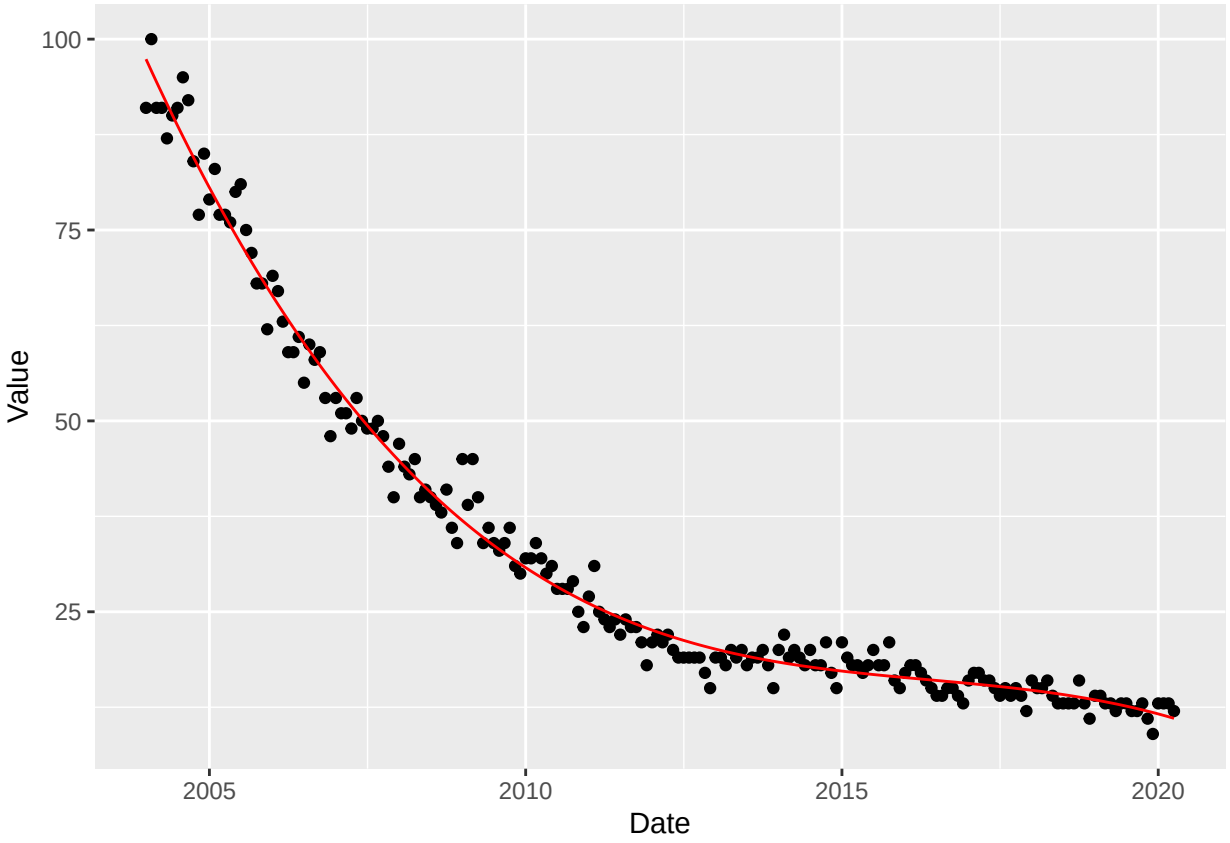
Where y is the value and x's represents the date predictor in a higher dimensional space (note that it is still being fit linearly with respect to β). Further, we assume an $\epsilon_t \sim WN(0, \sigma^2)$.

```
fit1 = lm(value~poly(date,3), data = df)
summary(fit1)
```

```
##
## Call:
## lm(formula = value ~ poly(date, 3), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4168 -1.2939 -0.1481  1.3000  9.1739
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      33.0102     0.1814  181.96   <2e-16 ***
## poly(date, 3)1 -285.9862     2.5398 -112.60   <2e-16 ***
## poly(date, 3)2  134.8104     2.5398   53.08   <2e-16 ***
## poly(date, 3)3  -43.4683     2.5398  -17.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.54 on 192 degrees of freedom
## Multiple R-squared:  0.988,  Adjusted R-squared:  0.9878
## F-statistic:  5263 on 3 and 192 DF,  p-value: < 2.2e-16
```

Now plot the original data plus the estimated trend (with the aid of LM's fitted.values object):

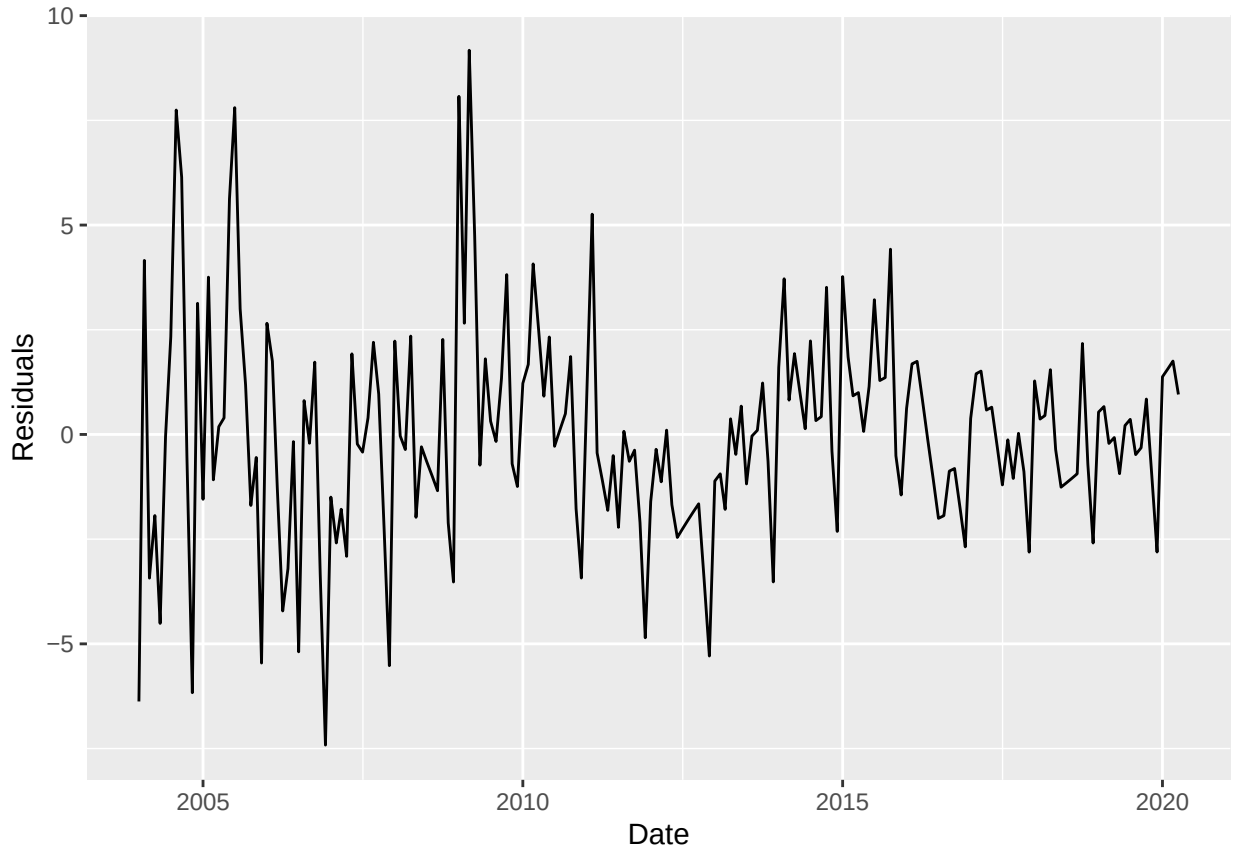
```
ggplot(df, aes(x = date, y = value)) + geom_point() +
  geom_line(aes(y =  fit1$fitted.values), col = "red") + ylab("Value") + xlab("Date")
```



The cubic polynomial appears to fit well to the values. Thus, we would expect this trend to capture most of the variation in the time series, thus leaving the residuals looking mostly white noise with equal variance (as assumed in our model).

Now show a time plot of the residuals (using LM's residuals object):

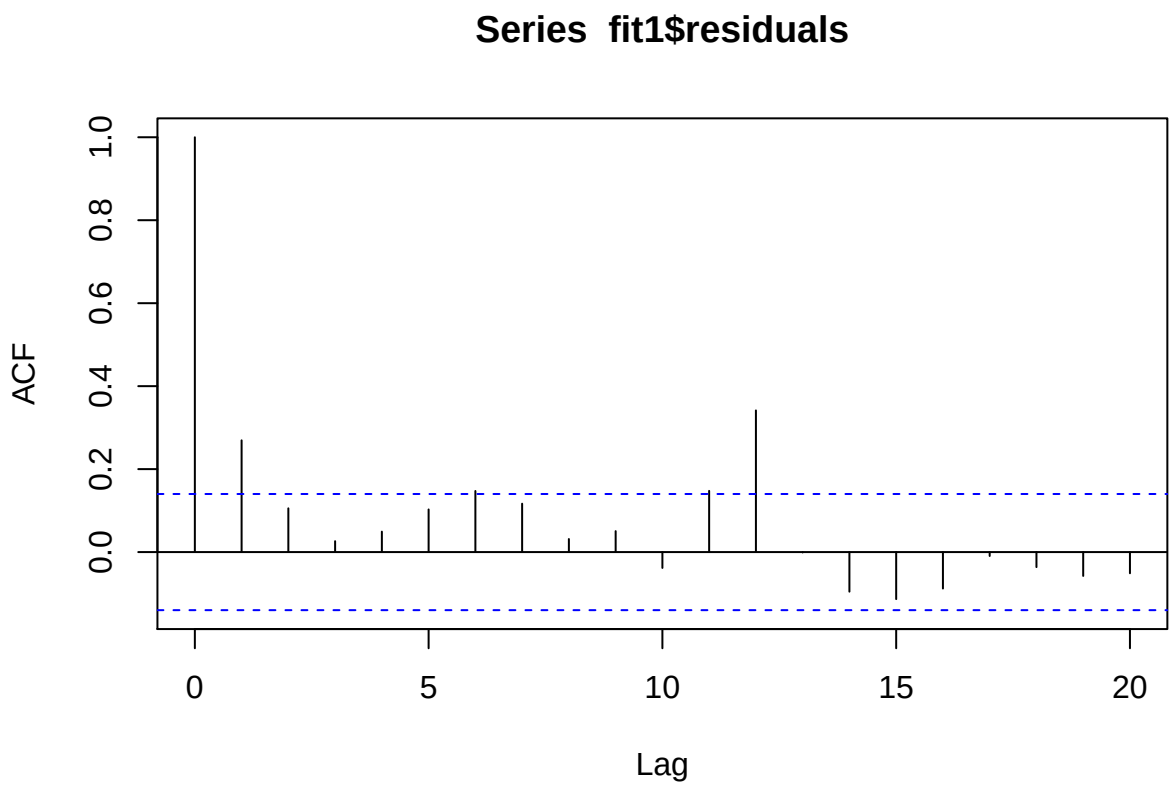
```
ggplot(df, aes(x = date, y = fit1$residuals)) + geom_line() + ylab("Residuals") + xlab("Date")
```



However, after looking at the resdiuals we can see that after a certain point (roughly 2012) the residuals deviate less from 0. Thus, this fit is showing signs of heteroscedasticity which violates our assumed distribution assumptions on ϵ_t.

Finally, show the acf of the residuals. We are thus plotting $\hat{\rho}(h)$:

```
acf(fit1$residuals, lag.max = 20)
```



From the ACF plot, we can detect departures from the large sample distribution of the autocorrelation function by assessing whether peaks in $\hat{\rho}(h)$ exceed $\pm 2/\sqrt{n}$. Thus, with 20 $h's$, we expect 0.05*20 = 1 departure from normality. However, we observe at least 4 peaks (excluding for $h = 0$) exceeding this threshold. This gives evidence against the assumption that the trend model reduced the observed data to white noise.

Part b

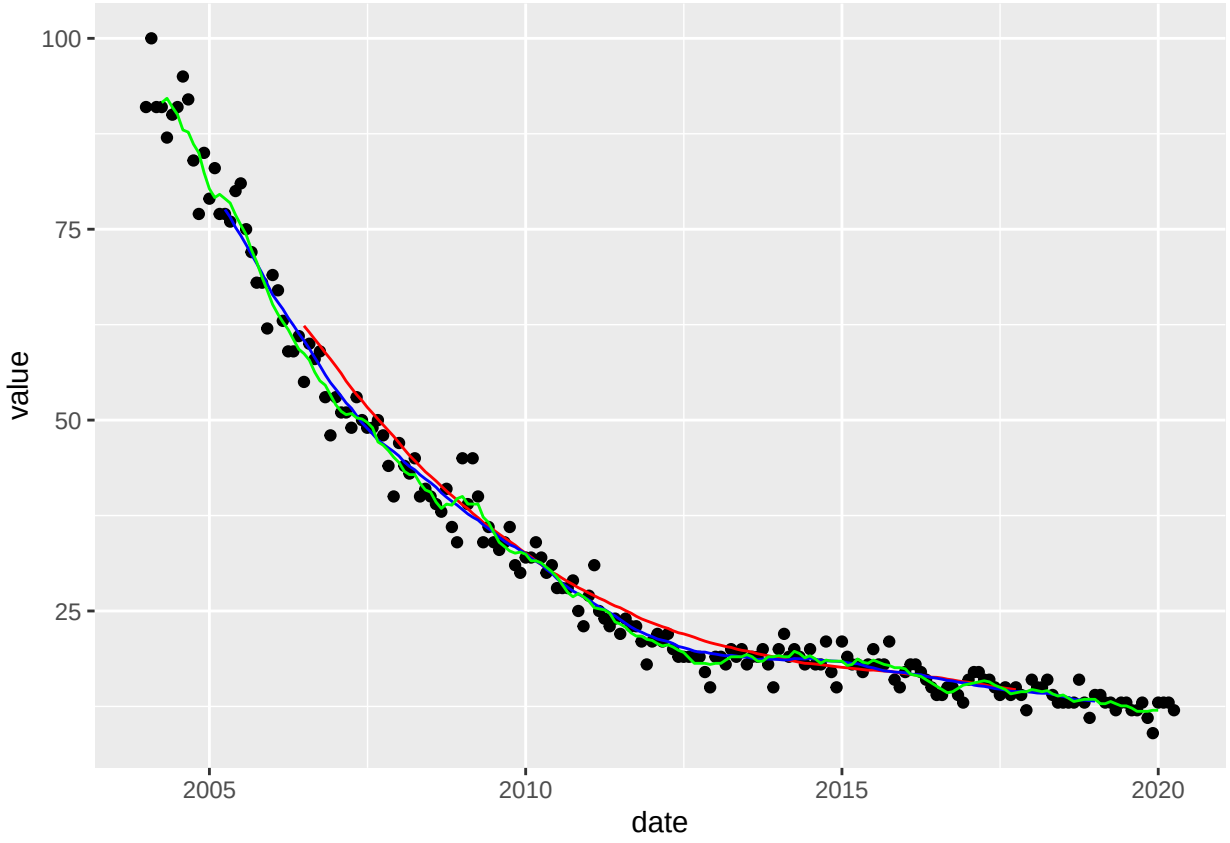
Now we consider smoothing the time series by use of the form: $y_t = m_t + \epsilon_t$, where ϵ_t is distributed as before and $\hat{m}_t$ has the form:

$$\hat{m}_t = \frac{1}{2q+1} \sum_{j=-q}^{q} X_{t+j}$$

Three values of q were considered; 3, 15, and 30.

```
q = 30
smooth1 <- stats::filter(df['value'], sides=2, filter=rep(1/(2*q+1),2*q+1))
q = 15
smooth2 <- stats::filter(df['value'], sides=2, filter=rep(1/(2*q+1),2*q+1))
q = 3
smooth3 <- stats::filter(df['value'], sides=2, filter=rep(1/(2*q+1),2*q+1))
```

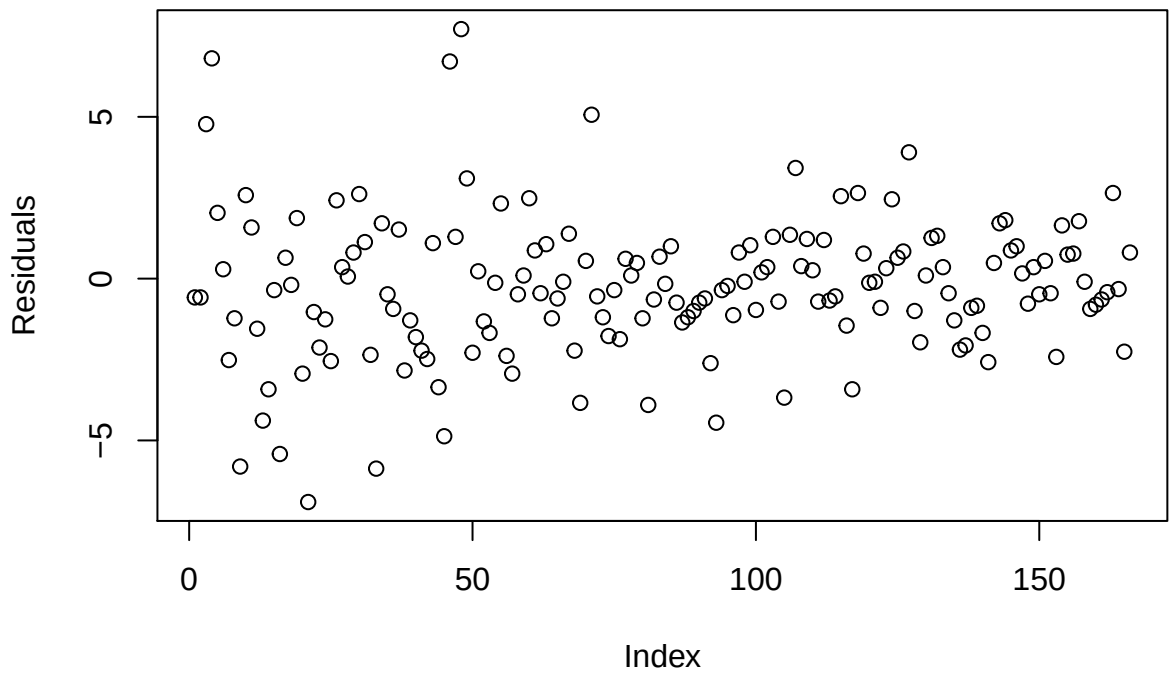
```
ggplot(df, aes(x = date, y = value)) + geom_point() +
  geom_line(aes(y =  smooth1), col = "red") +
  geom_line(aes(y = smooth2), col = "blue") +
  geom_line(aes(y = smooth3), col = "green")
```



From the resulting fits, it appears that $q = 3$ allows too much noise into the trend component and $q = 30$ suffers from excessive bias (and also doesn't include enough data). Therefore, $q = 15$ will be considered for the model.

Now to plot the residuals as a function of time:

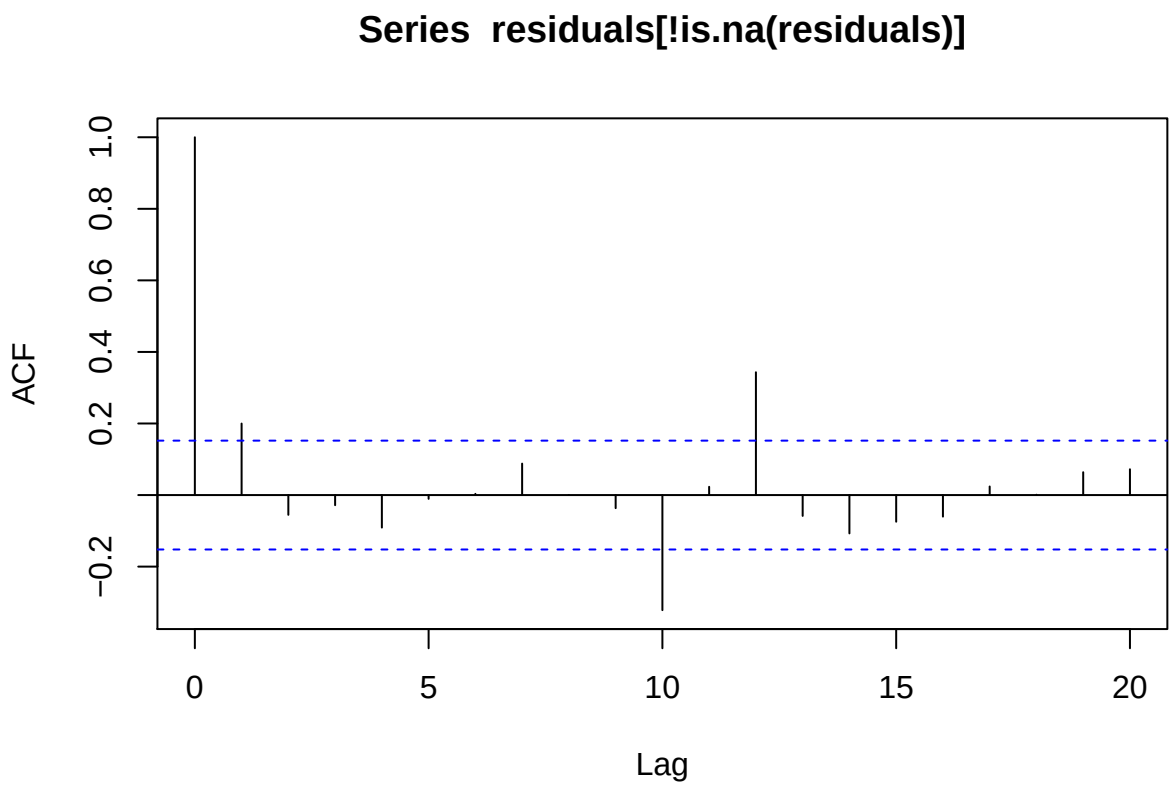
```
residuals = df['value'] - as.vector(smooth2)
plot(residuals[!is.na(residuals)], ylab = "Residuals")
```



Like the parametric model, it appears that the residual's variance decreases as the time progresses. This indicates that the model doesn't fully account for the autocorrelation in the data leaving some dependence structure in the residuals.

Finally the ACF plot:

```
acf(residuals[!is.na(residuals)], lag.max = 20)
```



From the ACF plot, we can detect departures from the large sample distribution of the autocorrelation function by assessing whether peaks in $\hat{\rho}(h)$ exceed $\pm 2/\sqrt{n}$. Thus, with 20 $h's$, we expect 0.05*20 = 1 departure from normality. However, we observe at least 3 peaks (excluding for $h = 0$) exceeding this threshold. This gives evidence against the assumption that the trend model reduced the observed data to white noise, although it is an improvement from the previous parameteric model.

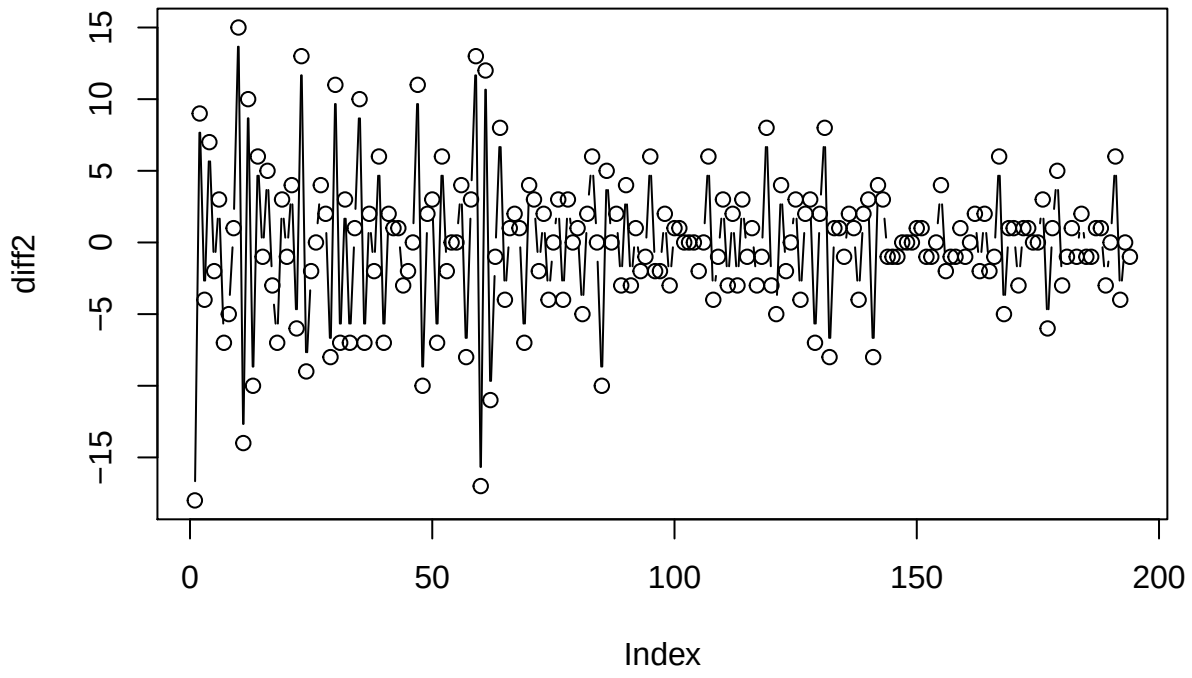
Part c

A third way to detrend the data is using differencing. Since the original IBM data appears to be nonlinear, we consider a difference of order 2:

$$\nabla^2 X_t = X_t - 2X_{t-1} + X_{t-2}$$

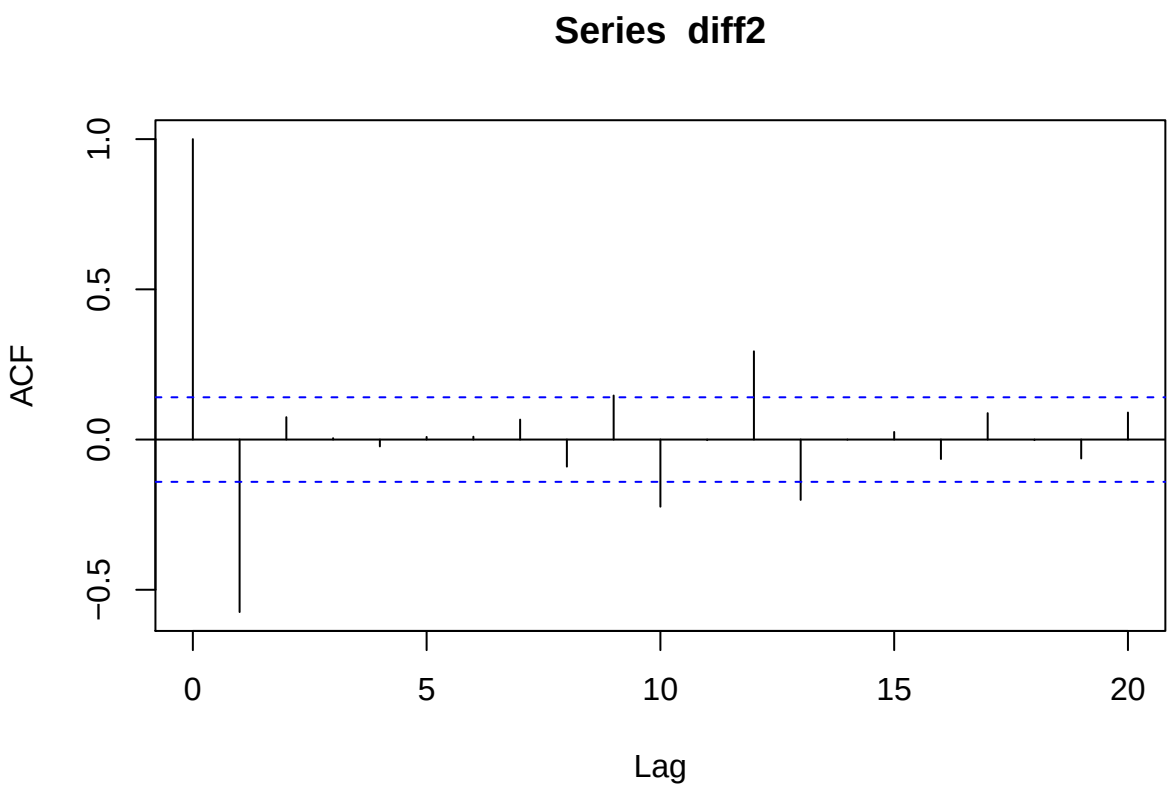
The differenced time series looks like:

```
diff2 = diff(df$value, differences = 2)
plot(diff2,type="b")
```



And plotting the ACF of the differenced series there still appears to be the same heteroscedasticity from the previous two methods.

```
acf(diff2, lag.max = 20)
```



Similar to the two previous methods, we only expect 1 departure from normality given this amount of lags. However, we observe that 5 lags exceed $\pm 2/\sqrt{n}$. Thus we have evidence against these residuals not representing white noise.

Problem 2

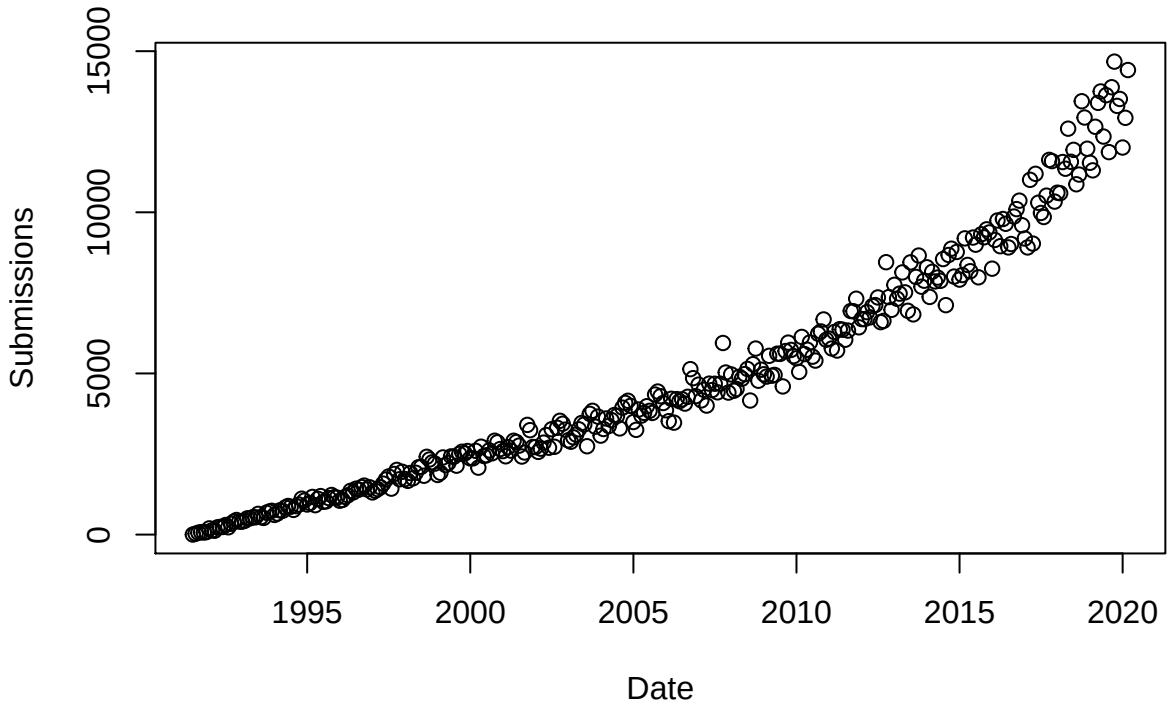
```
arx = read.csv('./arxiv.csv')
arx[,'month'] = parse_date_time(arx[,'month'], "ym")

# Remove partial April observation
arx <- arx %>%
  slice(1:n()-1)
str(df)
```

```
## 'data.frame':    196 obs. of  2 variables:
##  $ date : POSIXct, format: "2004-01-01" "2004-02-01" ...
##  $ value: int  91 100 91 91 87 90 91 95 92 84 ...
```

Part a

Plotting the submissions (X_t) as a function of time:

```
plot(arx$month, arx$submissions, xlab = "Date", ylab = "Submissions")
```

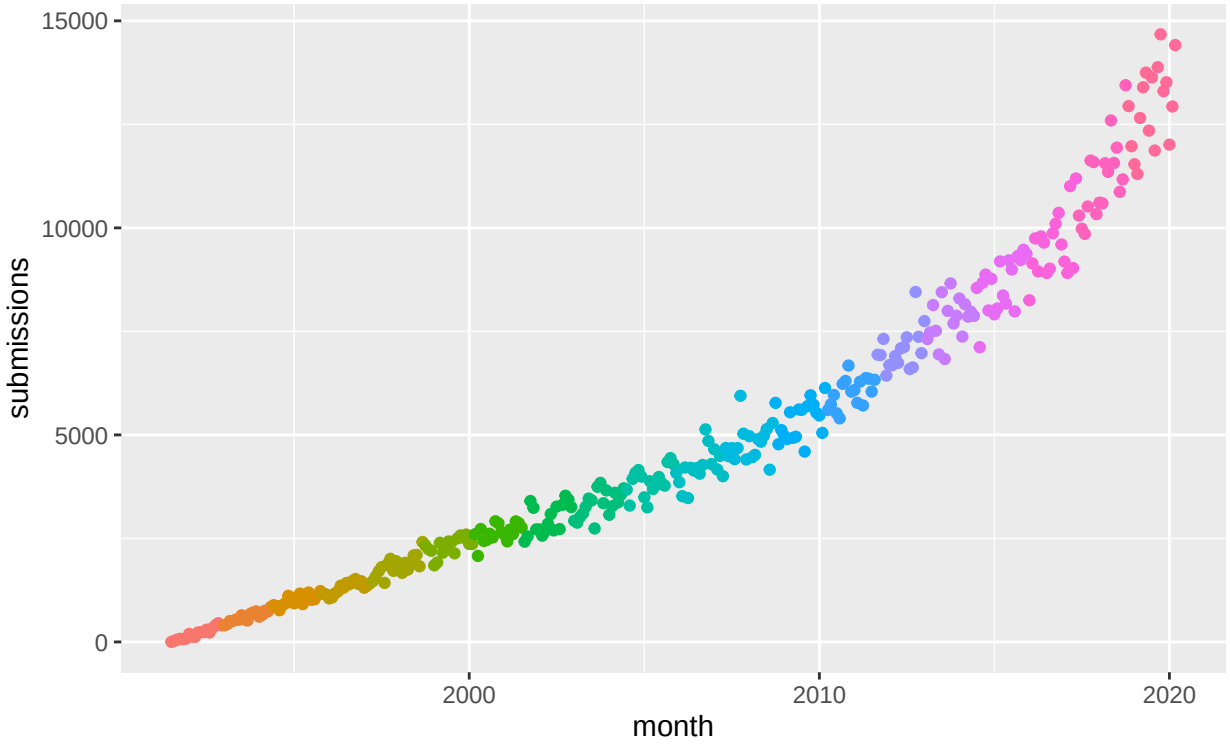


From this plot, I would conclude that the variance is not contant over time. Such data is known as heteroscedastic. In particular, it appears that the variance may vary as a function of the mean:

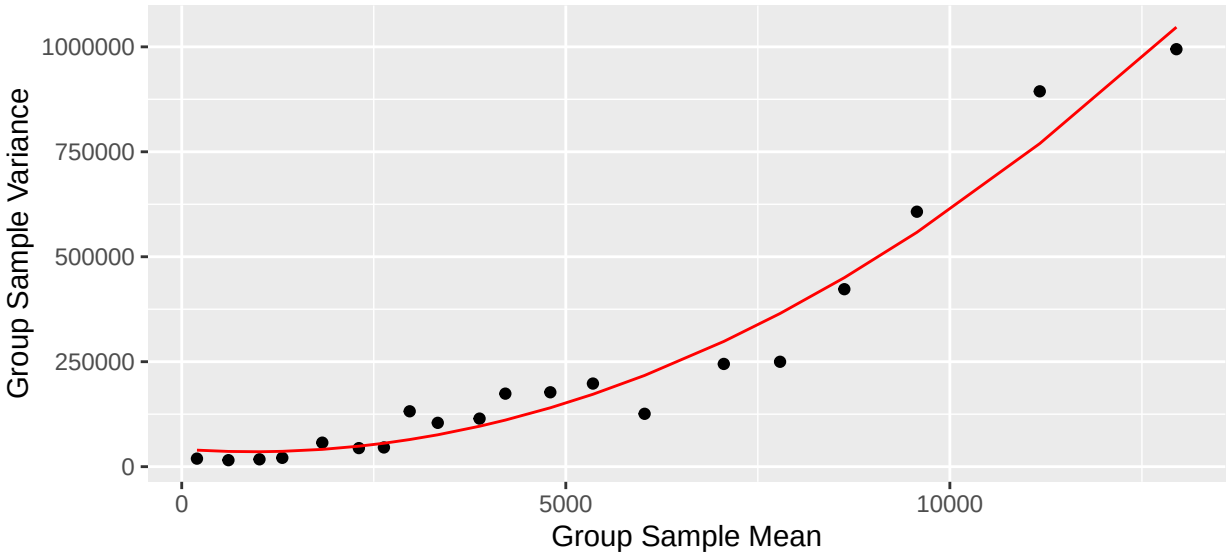
$$Var(X_t) = g(\mu_t)$$

Depending on whether the $g(\mu_t)$ is linear or quadratic, different transformations would be appropriate. A square root and logarithmic transformation would work in theory, respectively.

To develop a sense of this trend, first bin the response variable into equal groups:



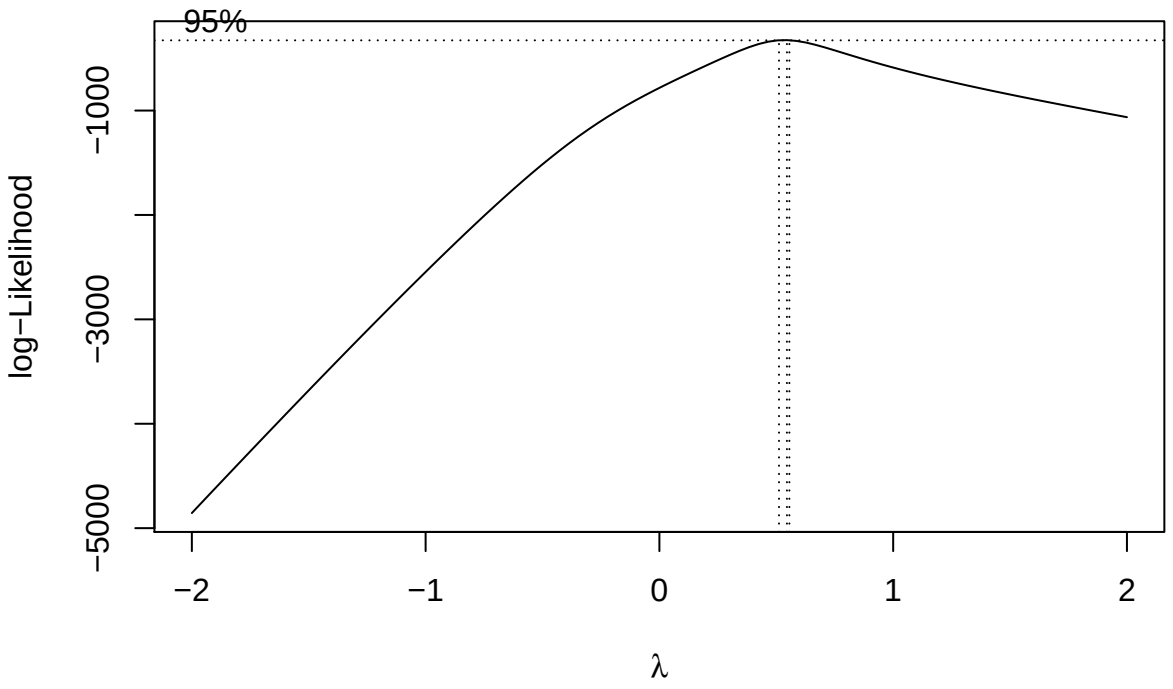
And then plot the mean of each group versus its sample variance:



As this is only meant to be an estimation of the true relationships, we can see that the $Var(X_t) \approx g(\mu_t)^2$. Thus we have some evidence for conducting a log transformation.

Perhaps a more illustrative plot is the boxcox transformation:

```
fit = lm(submissions~month,data = arx)
MASS::boxcox(fit, plotit = TRUE)
```



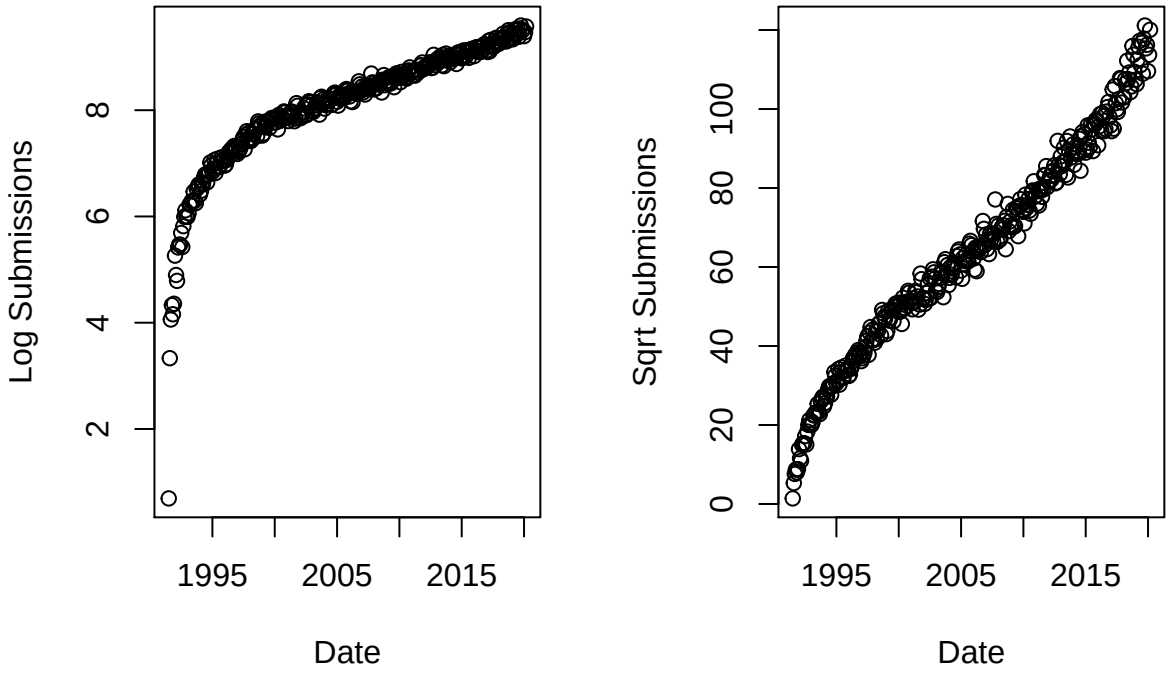
The boxcox MLE is maximized at $\lambda = 0.5$ which directly translates to a square root transformation. This somewhat contradicts the previous plot so we will try both transformations.

Part b

We now plot the log transformed and square root transformed data,

$$Z_1(t) = \log X_t, Z_2(t) = \sqrt{X_t}$$

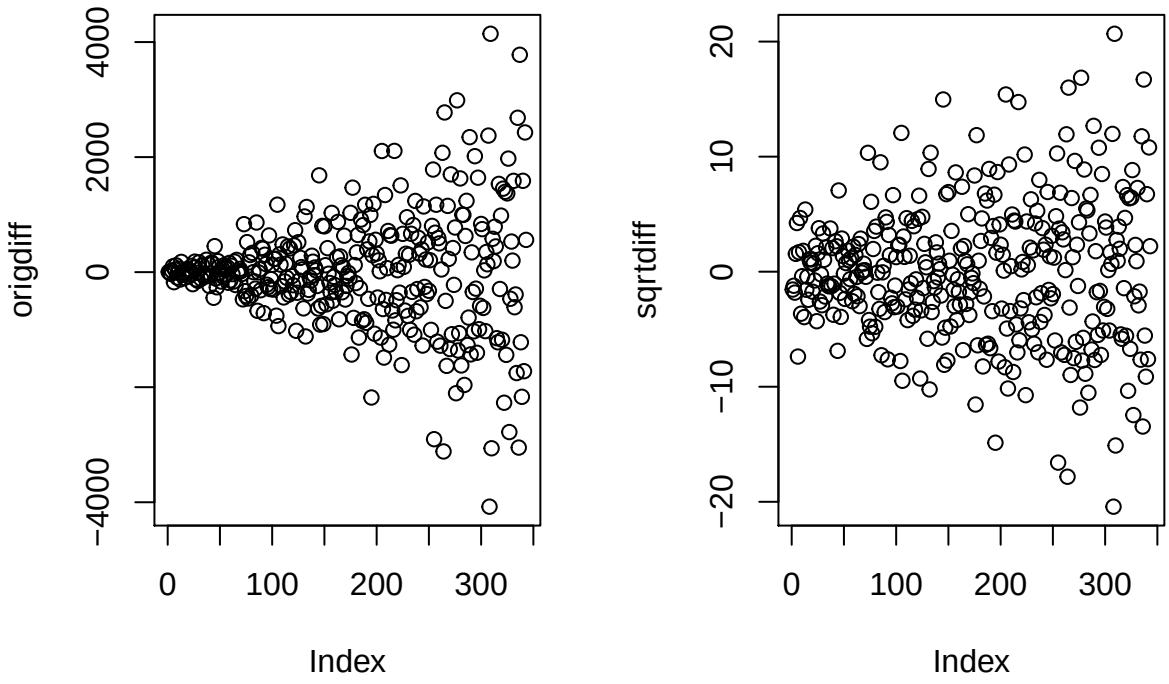
```
par(mfrow = c(1,2))
arx["logsub"] = log(arx$submissions)
plot(arx$month, arx$logsub, xlab = "Date", ylab = "Log Submissions")
arx["sqsub"] = sqrt(arx$submissions)
plot(arx$month, arx$sqsub, xlab = "Date", ylab = "Sqrt Submissions")
```



Although both transformations seem to stabilize the variance compared to the original data, on the margin the sqrt transfrom is preferred since the submissions is count style data. There is a theoretical justification for modeling count data as Poisson distributed which is linear in both mean and variance. Hence, given this data type, we will proceed with the square root transformation.

Now we compare the original differenced observations (∇X_t) with sqrt transformed differenced data (∇Z_t).

```
origdiff = diff(arx$submissions, differences = 2)
sqrtdiff = diff(arx$sqsub, differences = 2)
par(mfrow= c(1,2))
plot(origdiff)
plot(sqrtdiff)
```



Based off this comparison plot, it appears that the sqrt transform stabilized the variance more than just the original data. This stands to reason since when $Var(X_t) = C\mu_t$, for $f(x) = \sqrt{x}$, then $Var(Z_t) = Var(\sqrt{X_t}) \approx C$.

Part c

To forecast differenced data, first observe that ∇Z_t the differenced data, behaves like white noise after transformation. If we let

$$Y_t = \nabla Z_t$$

Then we can forecast Y_{t+1} using

$$\bar{Y} = (Y_2 + ... + Y_t)/(t-1)$$

Then relating it back to the X_t data:

$$Y_{t+1} = \bar{Y} = \nabla Z_{t+1} = Z_{t+1} - Z_t = \sqrt{X_{t+1}} - \sqrt{X_t}$$

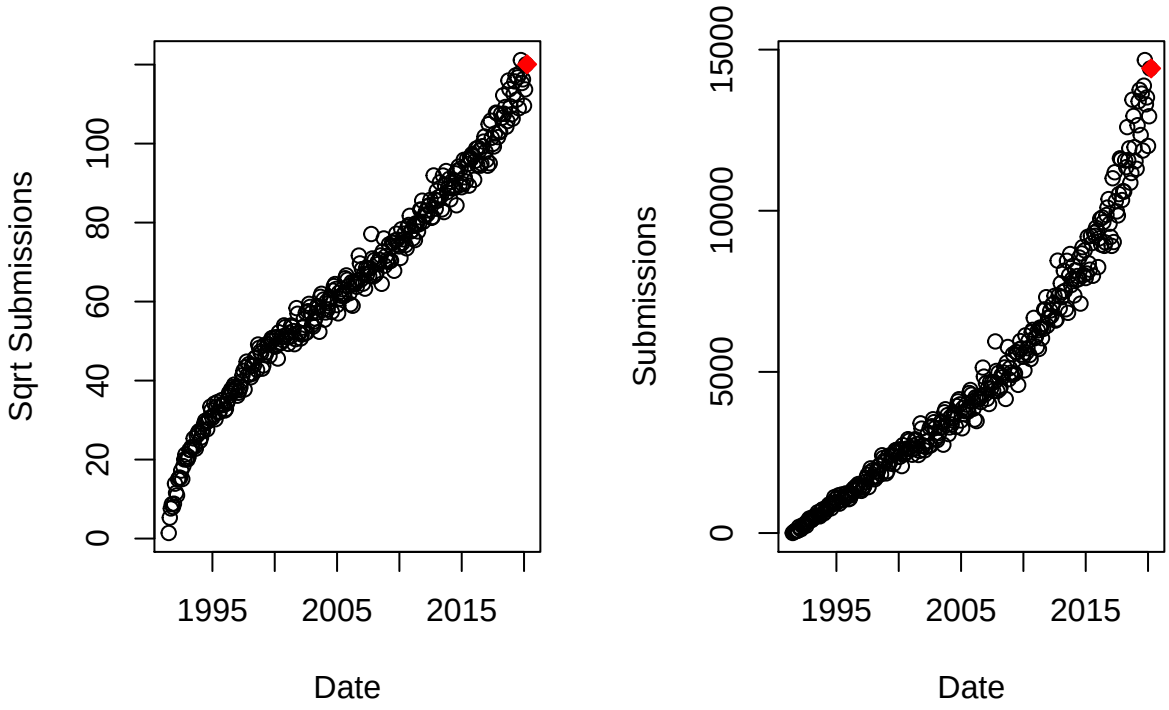
then,

$$\bar{Y} = \sqrt{X_{t+1}} - \sqrt{X_t}$$

Finally, to predict at X_{t+1} we must use the relation:

$$X_{t+1} = (\sqrt{X_t} + \bar{Y})^2$$

```
ybar = mean(sqrtdiff)
prediction = (ybar + arx$sqsub[length(arx$sqsub)])^2
par(mfrow= c(1,2))
plot(arx$month, arx$sqsub, xlab = "Date", ylab = "Sqrt Submissions")
points(x = parse_date_time('2020-04-01', "ymd"), y = ybar + arx$sqsub[length(arx$sqsub)], col = 'red', p
plot(arx$month, arx$submissions, xlab = "Date", ylab = "Submissions")
points(x = parse_date_time('2020-04-01', "ymd"), y = prediction, col = 'red', pch = 23, bg = 2)
```



```
prediction
```

```
## [1] 14416.73
```

The predicted value is shown as the red dot in both the sqrt-transformed and original space.