Time Series Analysis

Zander Bonnet

The goal of the analysis is to accurately predict future values of the data. This data is collection of data related to climate change. It contains a date column and a temp column. In this analysis I will clean and manipulate the data so that it can be efficiently used in an ARIMA model, and then build the model. With this model I will predict the next 2 years I will choose this threshold as it will give an insight into the near future while maintaining some form of accuracy. If we were to forecast longer than this we might encounter some dramatic accuracy loss.

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tsutils)
library(ggplot2)
library(lmtest)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(tseries)
library(Metrics)
```

```
##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
d <- read.csv('/Users/zanderbonnet/Desktop/GCU/DSC-570/Topic_2/data/new_train.csv')
d$date <- as.Date(d$date, format = '%d-%m-%Y')
which(is.na(d))
```

```
## integer(0)
```

```r
sc <- scale(d$temp)
which(sc >3 | sc < -3)
```

```
## integer(0)
```

The data contains no missing values and has no extreme outliers so we can proceed with the analysis.

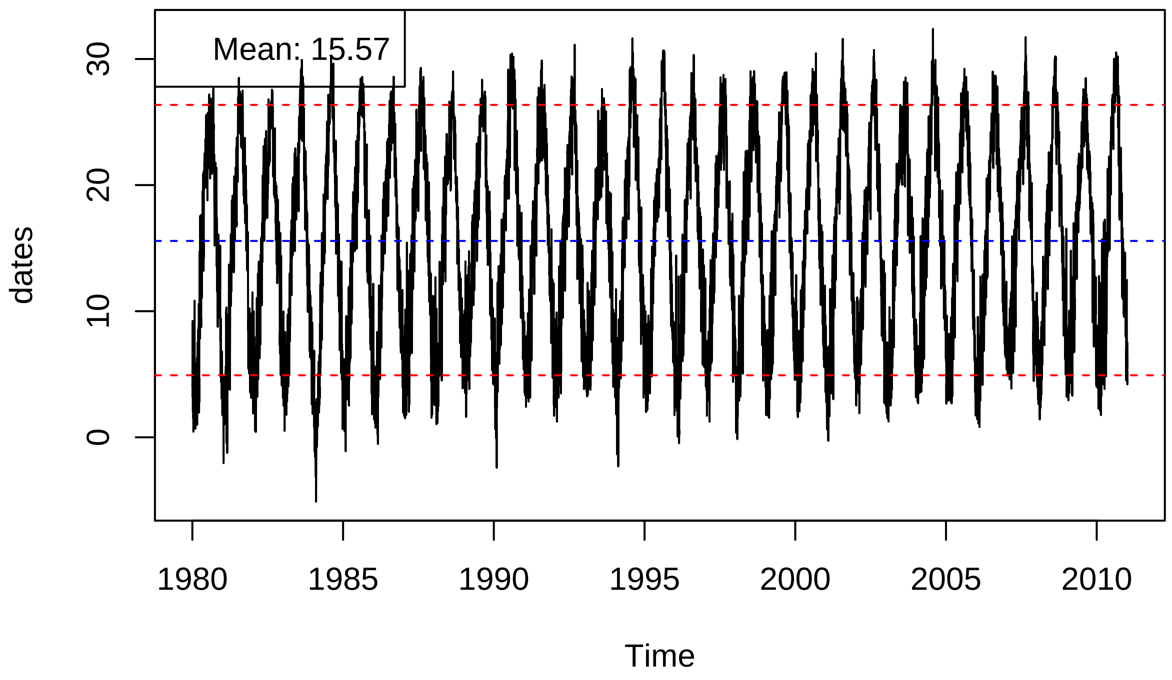
```r
dates <- ts(d$temp, start = 1980, frequency = 365)
head(dates)
```

```
## Time Series:
## Start = c(1980, 1)
## End = c(1980, 6)
## Frequency = 365
## [1] 4.16 4.06 7.12 9.23 3.20 7.01
```

```r
train <- dates[1:floor(length(dates)*.9)]
test <- dates[ceiling(length(dates)*.9):length(dates)]
```

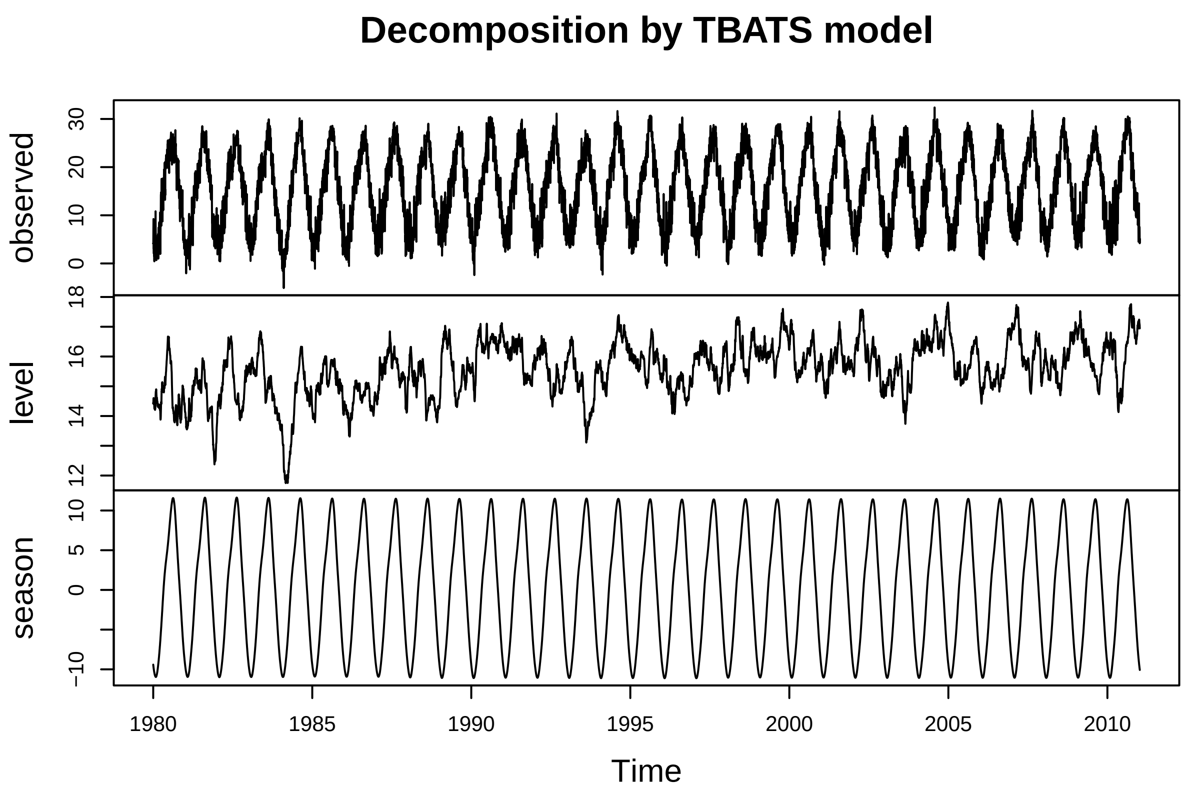
The data was successfully transformed into a time series object.

```r
plot(dates)
abline(h = mean(dates), col = 'blue', lty = 2)
abline(h = quantile(dates,.9), col = 'red', lty = 2)
abline(h = quantile(dates,.1), col = 'red', lty = 2)
legend(x = 'topleft', legend = c(
  paste('Mean:', round(mean(dates),2))
  ))
```



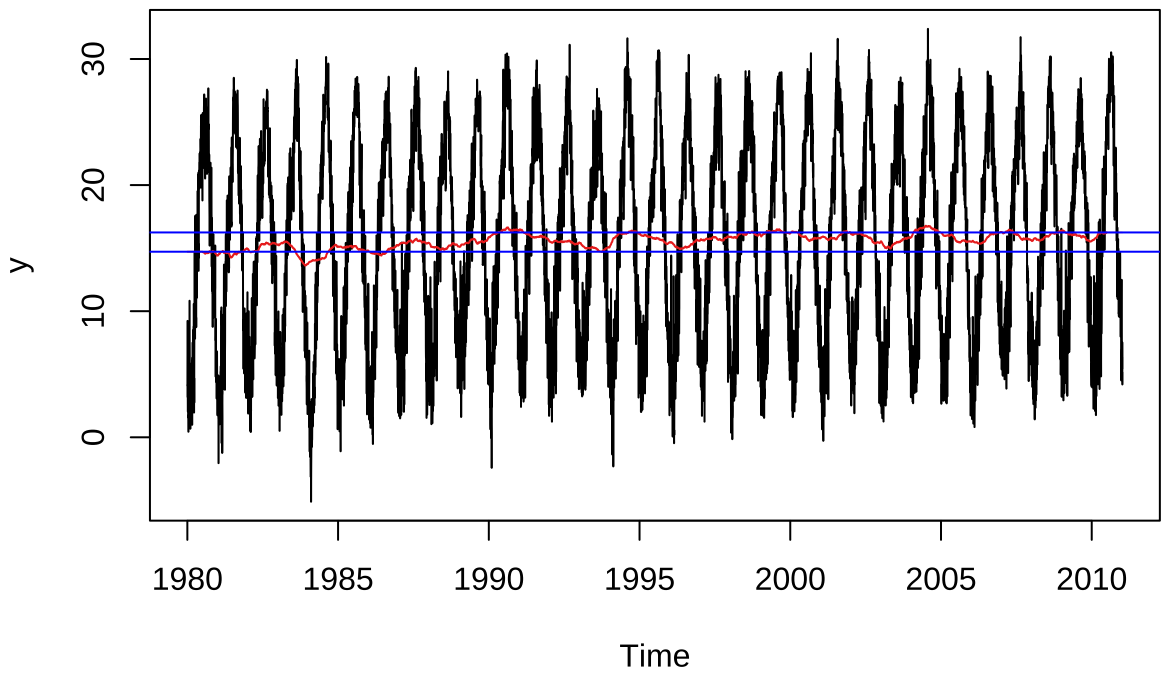
The data shows that it is potentially highly seasonal, and appears to be stationary with a mean of 15.7.

```
tb <- tbats(dates)
plot(tb)
```



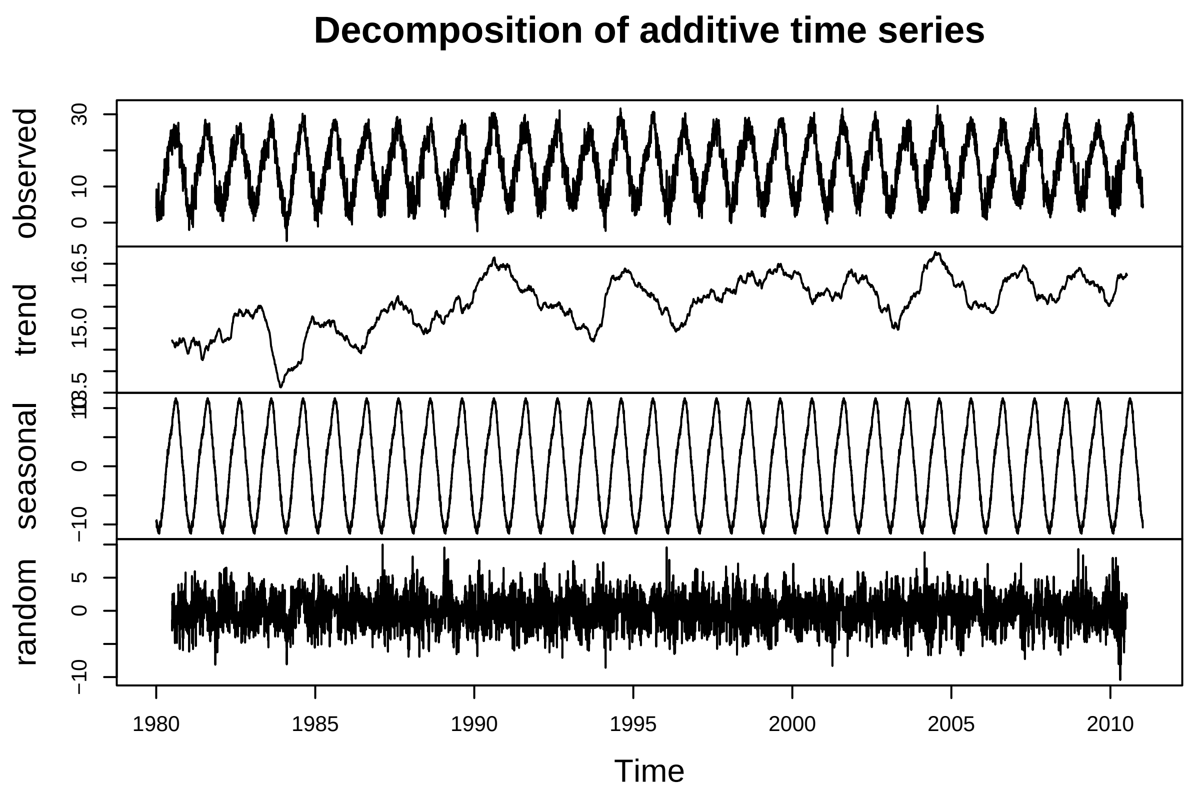
Using decomposition by TBATS we can see that there is strong seasonality with a period of 1 year.

```r
av <- cmav(dates, outplot = T)
abline(h = tail(av,1), col = 'blue', lty = 1)
abline(h = head(av,1), col = 'blue', lty = 1)
```



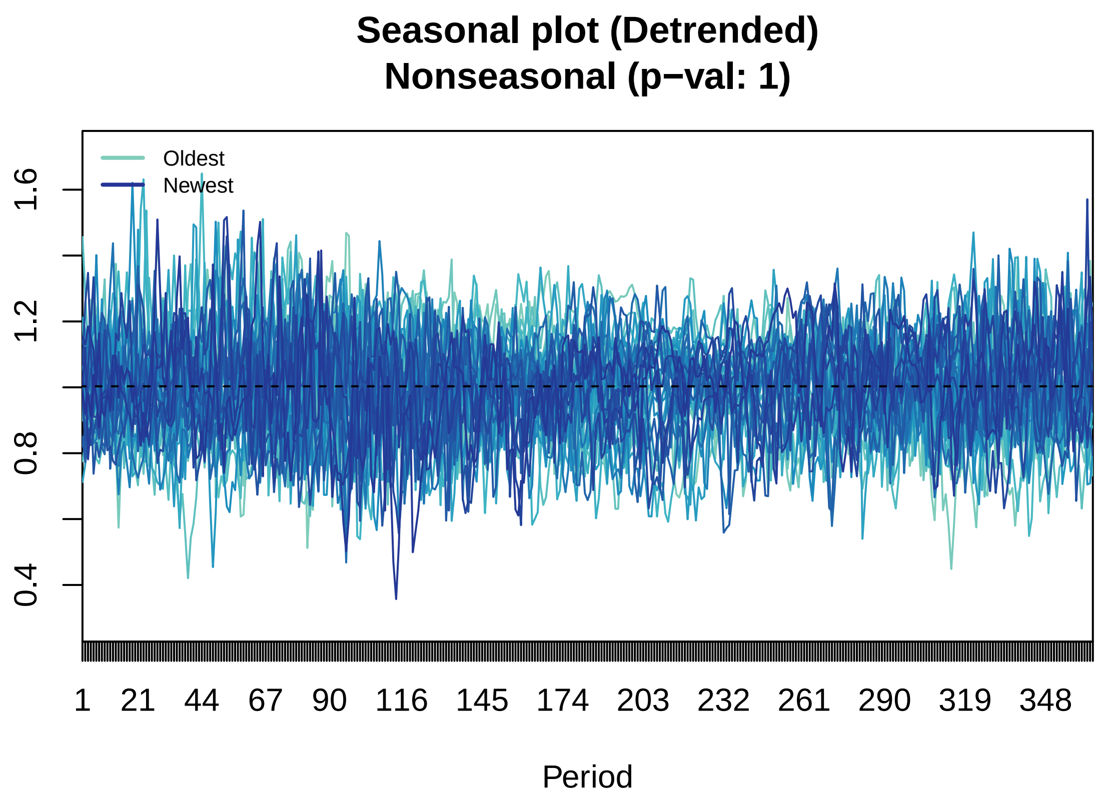
The central moving average shows that data has a slight upward trend.

```
dec <- decompose(dates)
plot(dec)
```



By decomposing the data we can see that there is a positive trend over time. Just like we saw with the central moving average. We can also see that there is a strong seasonal pattern within the data.

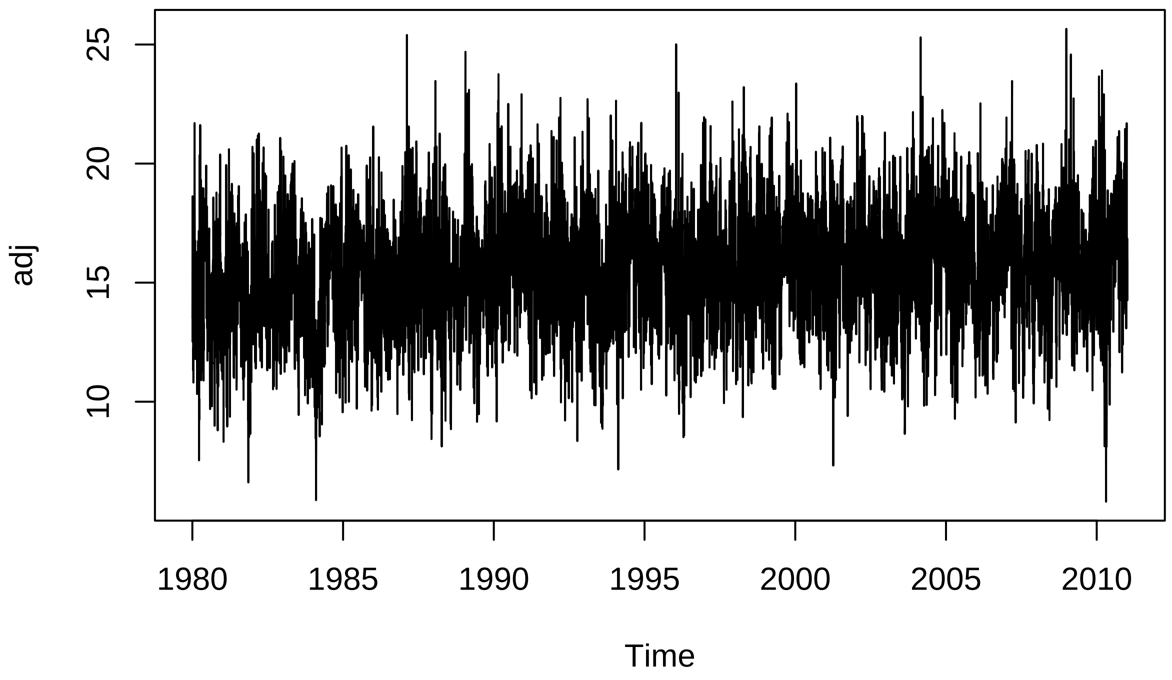
```
adj <- dates - dec$seasonal
seasplot(adj)
```



```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: FALSE  (pval: 1)
```

By decomposing the data we were able to remove the seasonality and preserve the trend within the data. This is seen in the results of the seasplot function where the is no statistical evidence for seasonality and very string evidence to suggest a trend within the data.

```r
plot(adj)
```

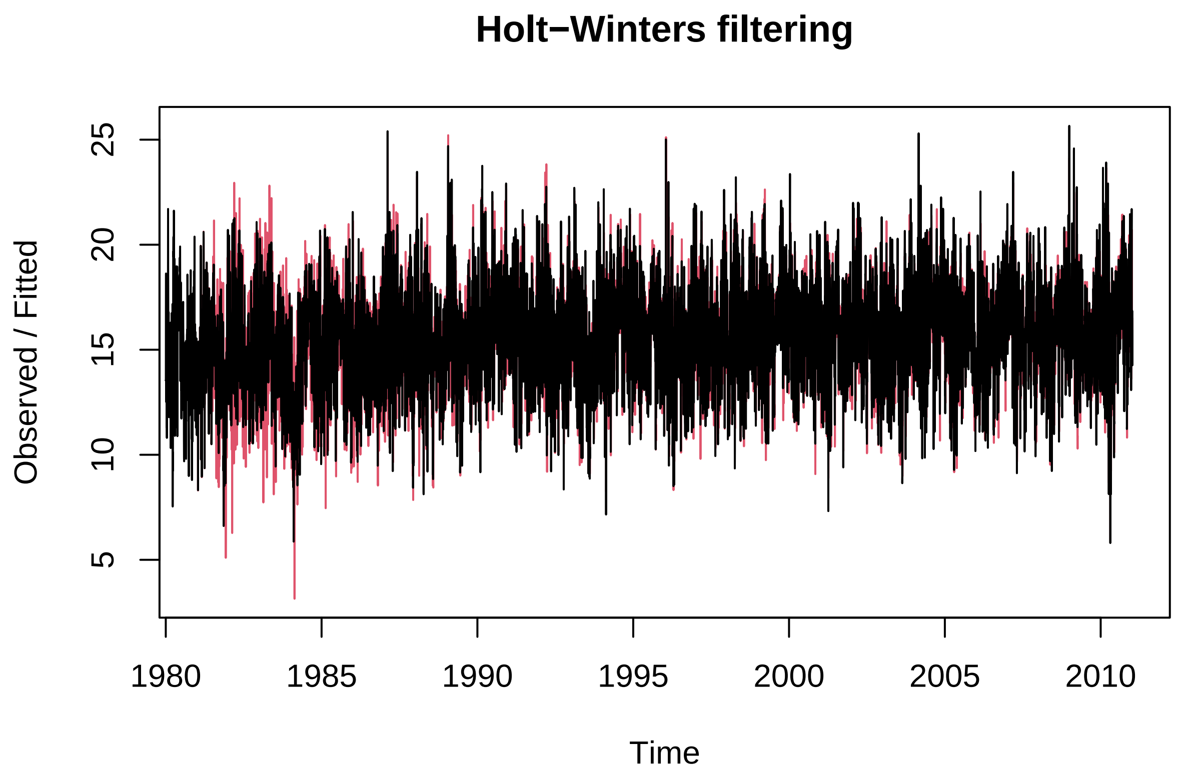


The new plot shows no seasonality while maintaining the positive trend from the original data.

```
adj_hw_mod <- HoltWinters(adj)
summary(adj_hw_mod)
```

```
##              Length Class  Mode
## fitted       43832  mts    numeric
## x            11323  ts     numeric
## alpha            1  -none- numeric
## beta             1  -none- numeric
## gamma            1  -none- numeric
## coefficients   367  -none- numeric
## seasonal         1  -none- character
## SSE              1  -none- numeric
## call             2  -none- call
```

```
plot(adj_hw_mod)
```



```r
print(adj_hw_mod$alpha)
```

```
##     alpha
## 0.6972688
```

```r
rmse(adj, adj_hw_mod$fitted)
```

```
## [1] 11.31554
```

When applying the Holt-Winters algorithm to the adjusted data we can see that it performs fairly well but it does have an RMSE of 11.31 on the training data. The alpha parameter is .697. This parameter is the weight that the model places on the level. This is to say how much weight do the values immediately before this value have. The higher this value the more weight they have.

```r
am_mod <- arima(adj)
coeftest(am_mod)
```

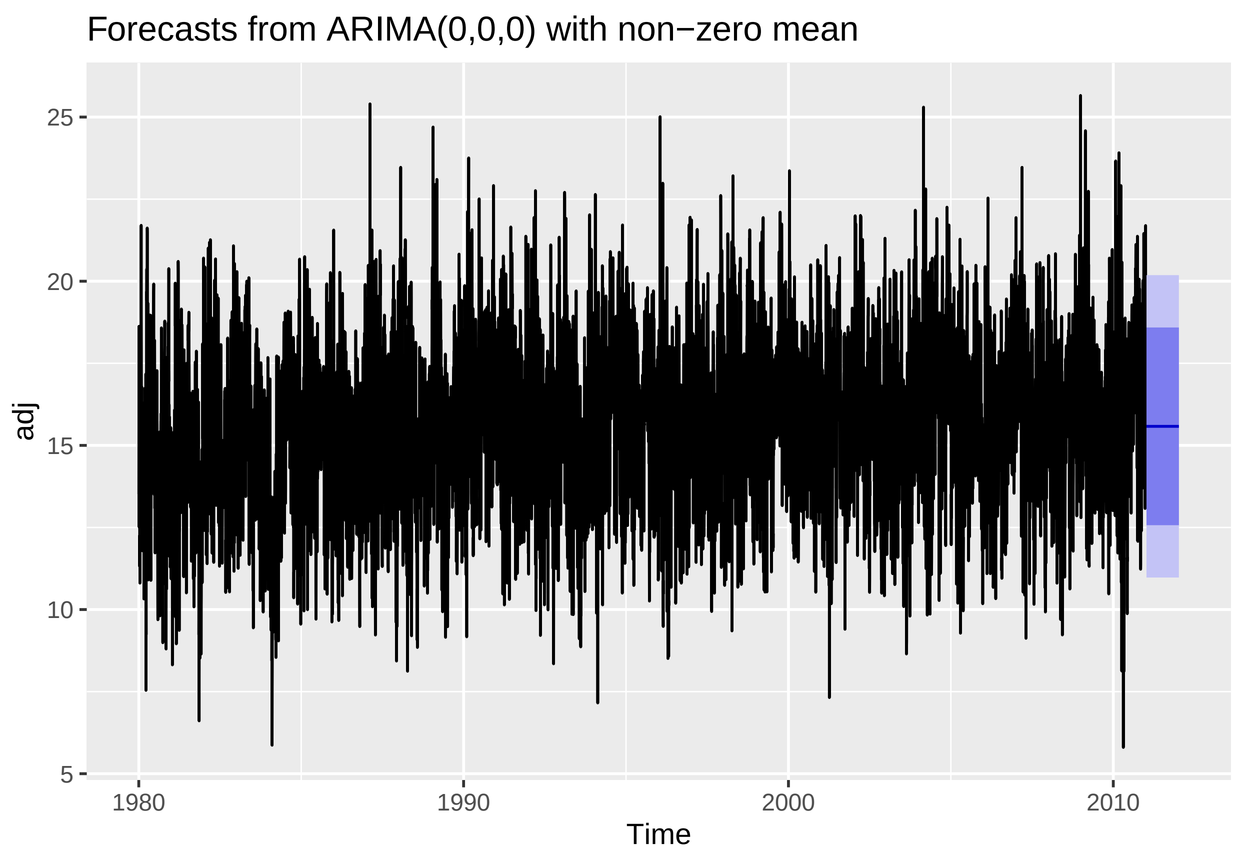
```
##
## z test of coefficients:
##
##            Estimate Std. Error z value  Pr(>|z|)
## intercept 15.580061   0.022077  705.71 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Creating the arima model creates a significant model as the intercept value is statistically significant according to the coeftest function.

```
fit <- Arima(adj)
summary(fit)
```

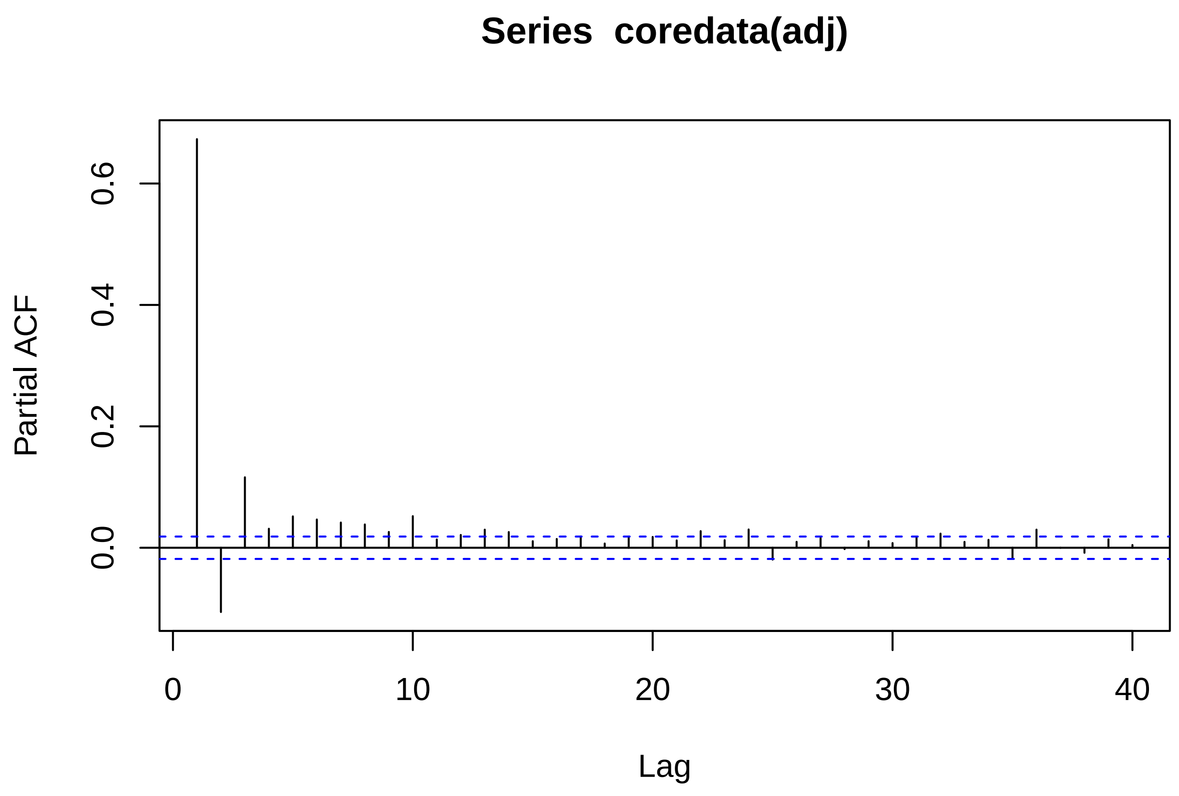
```
## Series: adj
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##          mean
##       15.5801
## s.e.   0.0221
##
## sigma^2 = 5.519:  log likelihood = -25737.5
## AIC=51479   AICc=51479   BIC=51493.67
##
## Training set error measures:
##                         ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -1.838117e-14 2.349235 1.864179 -2.486799 12.65357 0.718346
##                   ACF1
## Training set 0.6731363
```

```
autoplot(forecast(fit, 365))
```



The base Arima model is able to acurtaly predict the training data with an rmse of 2.3.

```r
pacf(coredata(adj))
```

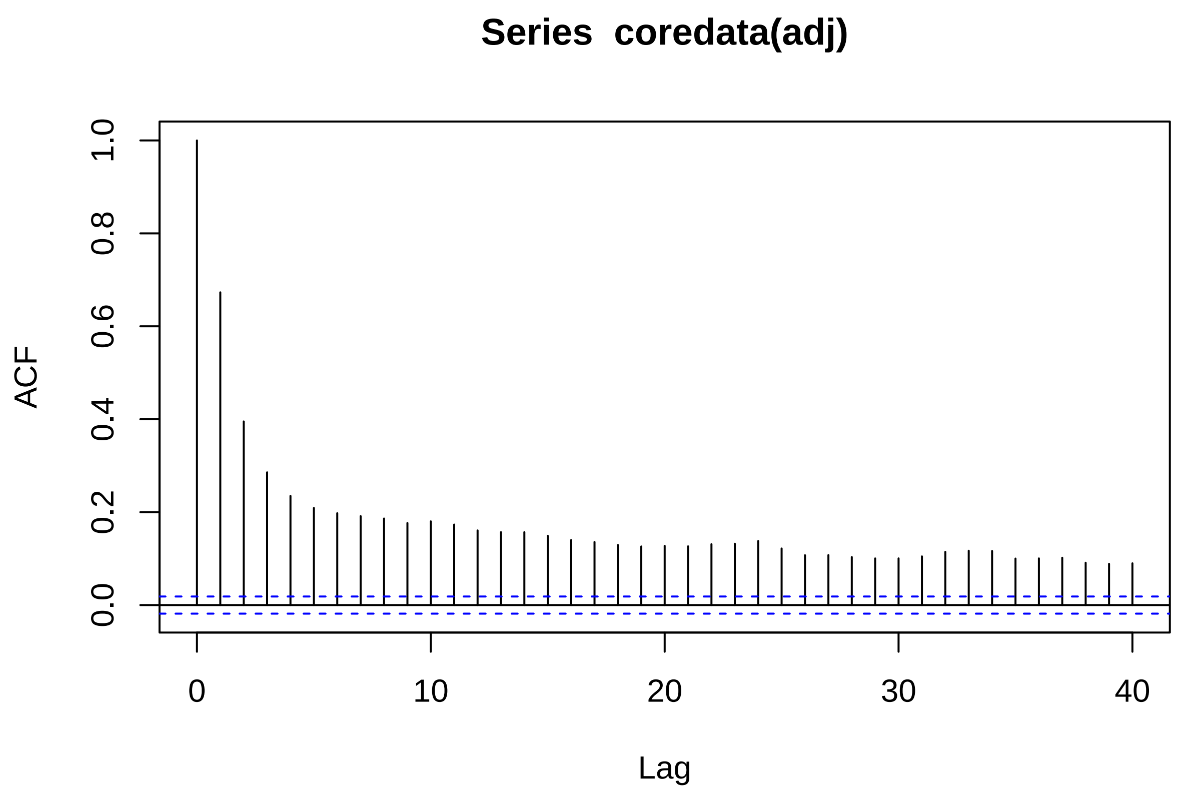


```r
adf.test(adj)
```

```
## Warning in adf.test(adj): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  adj
## Dickey-Fuller = -15.45, Lag order = 22, p-value = 0.01
## alternative hypothesis: stationary
```

```r
acf(coredata(adj))
```



```
am <- Arima(adj, order = c(4,1,4))
summary(am)
```

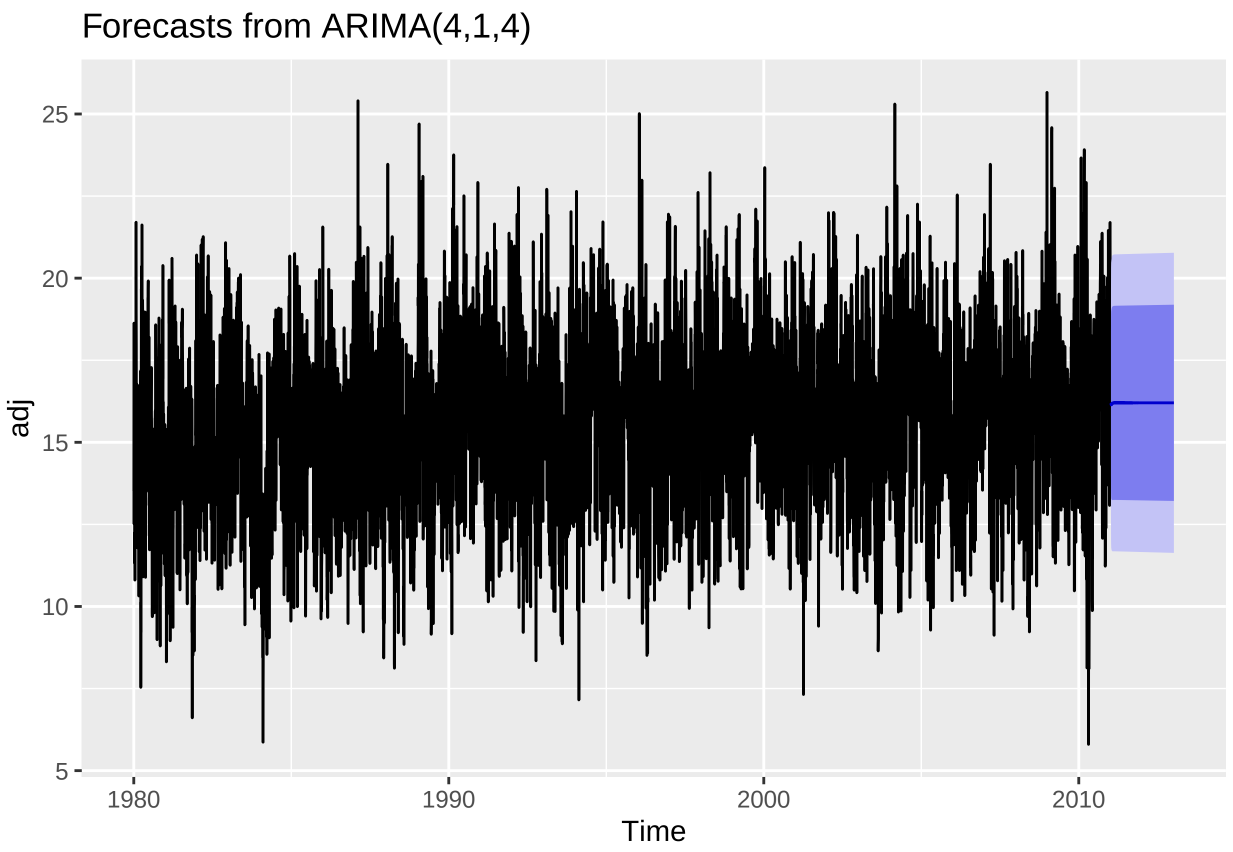
```
## Series: adj
## ARIMA(4,1,4)
##
## Coefficients:
##          ar1     ar2      ar3     ar4      ma1      ma2     ma3     ma4
##       0.3348  0.8661  -0.3810  0.0588  -0.5979  -1.1634  0.5827  0.1797
## s.e.  0.0487  0.0309   0.0501  0.0329   0.0486   0.0466  0.0502  0.0488
##
## sigma^2 = 2.901:  log likelihood = -22092.83
## AIC=44203.67   AICc=44203.68   BIC=44269.68
##
## Training set error measures:
##                      ME    RMSE      MAE       MPE     MAPE     MASE
## Training set 0.01733184 1.70264 1.312661 -1.188375 8.806544 0.505823
##                     ACF1
## Training set 0.001241014
```

```
coeftest(am)
```

```
##
## z test of coefficients:
##
```

```
##       Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.334800   0.048669   6.8791 6.024e-12 ***
## ar2  0.866122   0.030911  28.0194 < 2.2e-16 ***
## ar3 -0.380966   0.050095  -7.6049 2.851e-14 ***
## ar4  0.058835   0.032928   1.7868 0.0739753 .
## ma1 -0.597944   0.048631 -12.2954 < 2.2e-16 ***
## ma2 -1.163430   0.046573 -24.9807 < 2.2e-16 ***
## ma3  0.582651   0.050168  11.6141 < 2.2e-16 ***
## ma4  0.179651   0.048810   3.6807 0.0002326 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
autoplot(forecast(am,365*2))
```



We can increase the strength of the Arima model by tuning the parameters. By tuning the parameters I was able to get the RMSE down to about 1.7.

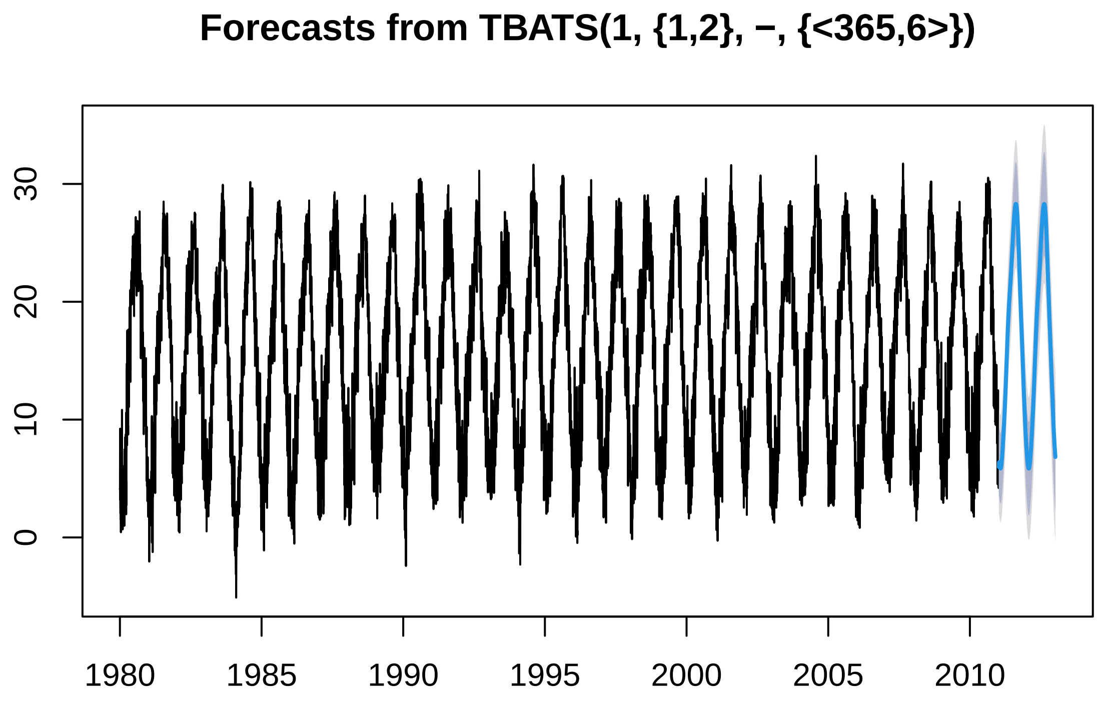
```r
summary(tb)
```

```
##                    Length Class  Mode
## lambda                  0 -none- NULL
## alpha                   1 -none- numeric
## beta                    0 -none- NULL
## damping.parameter       0 -none- NULL
## gamma.one.values        1 -none- numeric
## gamma.two.values        1 -none- numeric
## ar.coefficients        1 -none- numeric
## ma.coefficients        2 -none- numeric
## likelihood             1 -none- numeric
## optim.return.code      1 -none- numeric
## variance               1 -none- numeric
## AIC                    1 -none- numeric
## parameters             2 -none- list
## seed.states           16 -none- numeric
## fitted.values      11323 ts     numeric
## errors             11323 ts     numeric
## x                 181168 -none- numeric
## seasonal.periods       1 -none- numeric
## k.vector               1 -none- numeric
## y                  11323 ts     numeric
## p                      1 -none- numeric
## q                      1 -none- numeric
## call                   2 -none- call
## series                 1 -none- character
## method                 1 -none- character
```

```r
rmse(dates,tb$fitted.values)
```

```
## [1] 1.729167
```

```r
plot(forecast(tb, 365*2))
```



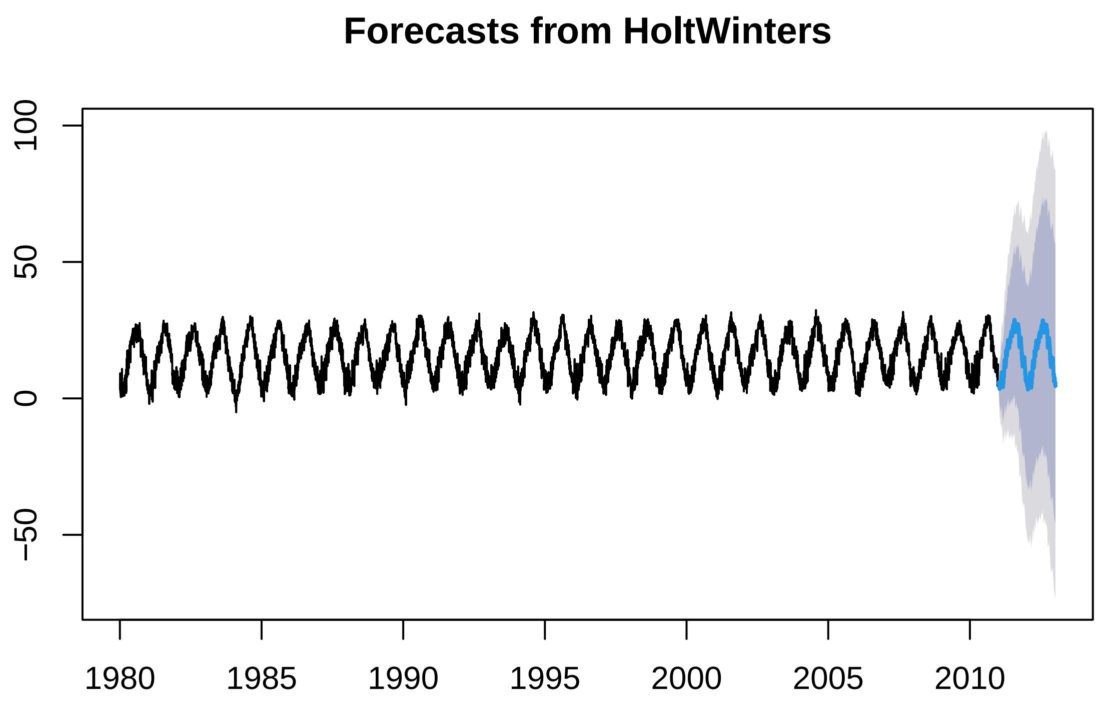
```r
hw_mod <- HoltWinters(dates)
summary(hw_mod)
```

```
##              Length Class  Mode
## fitted       43832  mts    numeric
## x            11323  ts     numeric
## alpha            1  -none- numeric
## beta             1  -none- numeric
## gamma            1  -none- numeric
## coefficients   367  -none- numeric
## seasonal         1  -none- character
## SSE              1  -none- numeric
## call             2  -none- call
```

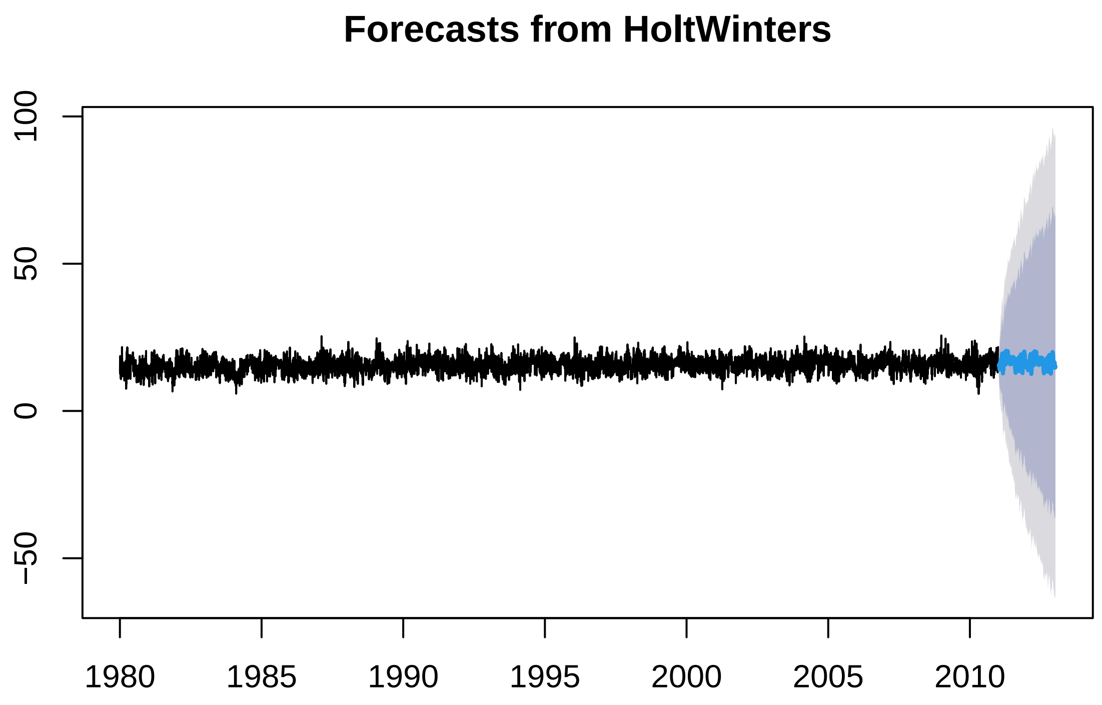
```r
rmse(dates,hw_mod$fitted)
```

```
## [1] 12.48037
```

```r
plot(forecast(hw_mod, 365*2))
```



```r
plot(forecast(adj_hw_mod, 365*2))
```



I was able to build and optimize an ARIMA model that is able to predict 2 years into the future. This model is very good at predicting the training data and has a tight confidence window for the future predictions.The model predicts that the average temperature in two years would be 16.2. That is increase of 1.4 degree increase in the average yearly temperature since 1980.

Looking at some of the other models we can see that TBATS also performs very well in this task, while Holt-Winters does lag behind slightly in terms of RMSE. The benefit of these models though is they do offer a more robust forecasting then in ARIMA. In many cases ARIMA gives a general trend that it predicts will be followed. While Holt-Winters and TBATS offer a more detailed prediction as can be seen above where I look at the TBATS and Holt-Winters models with both the original and decomposed data.

Exponential smoothing is also helpful in situations where we need to focus more on the data nearest to the observation. This is where the alpha parameter comes in. In cases where the data changes quickly and dramatically we must be able to put more weight on data points that are closer to the observation then distant points. This allows us to accurately forecast through seasonal data.

The goal of these models is to make predictions on future values within the data, and use that to make decisions. In this case we find that there will be increase in the average temperature as we look into the future. This will lead to many issues pertaining to regulation, legislation, and standards. It is our job as data scientists to look at the data objectively and lay out the data in an unbiased and ethical way. By doing this we will allow other professionals to make educated decisions. There are many aspects of the world that are dependent on things like climate change, so we must be careful on how we present our data. Large oil companies face lots of backlash when it comes to climate change, but a simple analysis could not soley place the blame on them. We presenting our findings we must be able define the scope of our findings, so they cannot be misinterpreted our used in a malicious way. In the Christian world view we are to be ethical and fair at all times, so doing these thing will allow us to accomplish that.

Reference

Time Series Dataset(Climate change prediction). Kaggle[Dataset]. https://www.kaggle.com/datasets/pranjalt/aurora-roundhacks/data?select=new_train.csv