Time-Series Analysis of Apple Prices in St. Petersburg from 2013 to 2020

Theo Teske

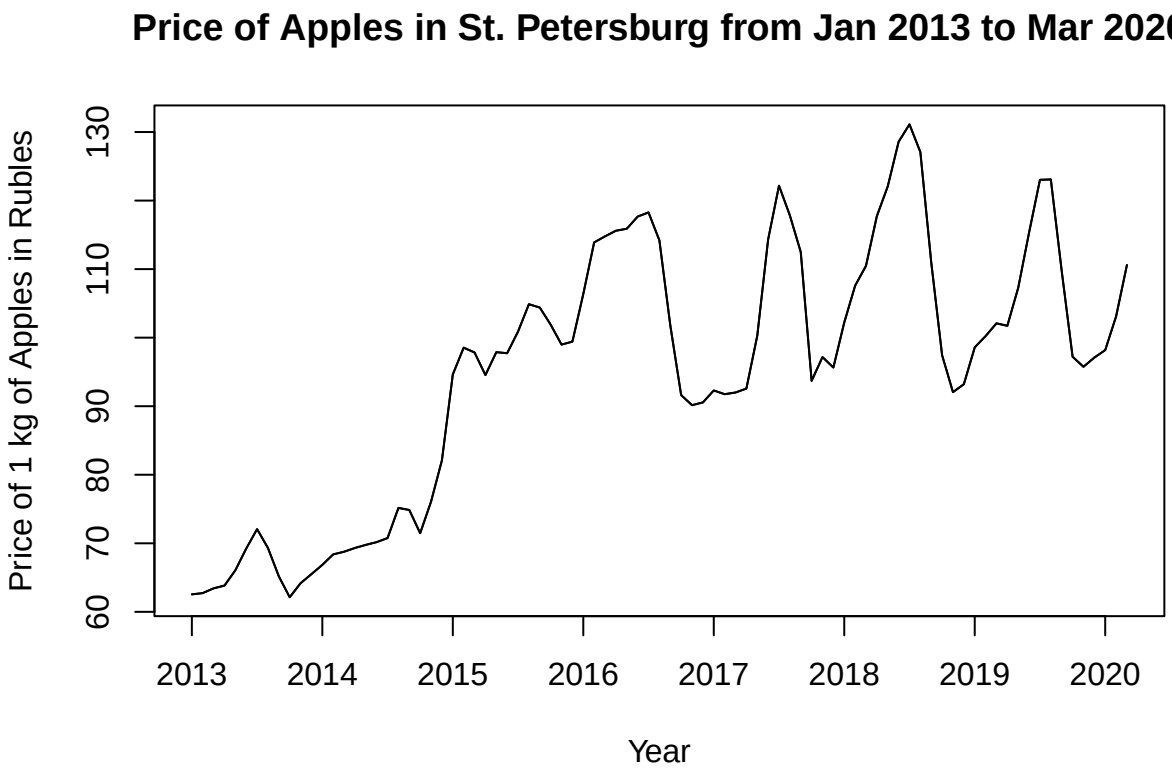
2023-01-26

Introduction

The dataset we use describes the price of apples in five different Russian cities over time, and is sourced from the Russian Statistical Service. We focus specifically on the data from St. Petersburg, spanning from January 2013 to March, 2020. Precisely, the units are the prices, in Rubles, of 1 kilogram of apples in St. Petersburg, Russia, measured monthly. We treat this price data as a time series. Apples are generally harvested in the fall, so the time series should exhibit clear seasonality, and due to inflation, the price of apples should increase over time, resulting in an upward trend.

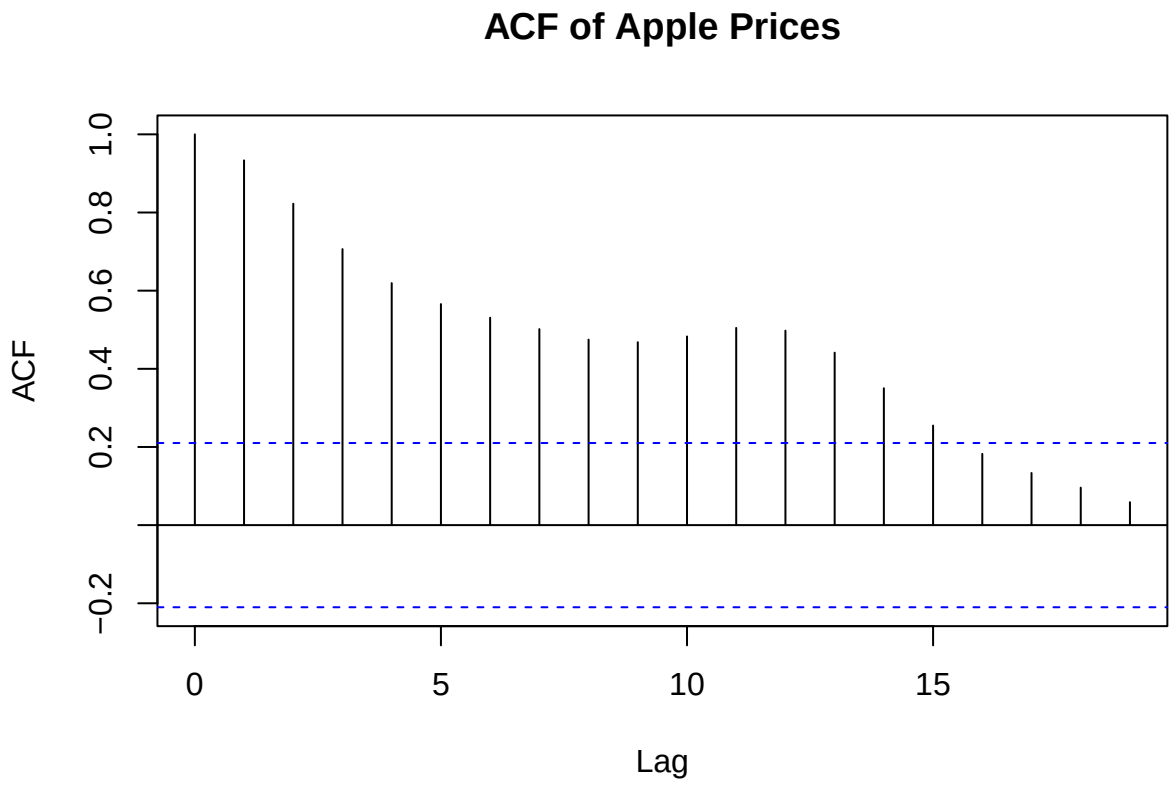
Analyzing the Data

```
# Read in data
dat <- read.table("C:/Users/Theo/Downloads/apples_ts.csv", header=TRUE,
   sep=",")

# Transpose data so each city is a column rather than years being columns
fr <- as.data.frame(dat)
final_df <- as.data.frame(t(fr))

# Choose Petersburg
vpr <- as.numeric(final_df$V3[-1])

# Create time series, then plot it
tspr<- ts(vpr,start=2013,freq=12)

plot(tspr, xlab = "Year", ylab="Price of 1 kg of Apples in Rubles", main="Price of Apples in St. Peters
lines(tspr)
```

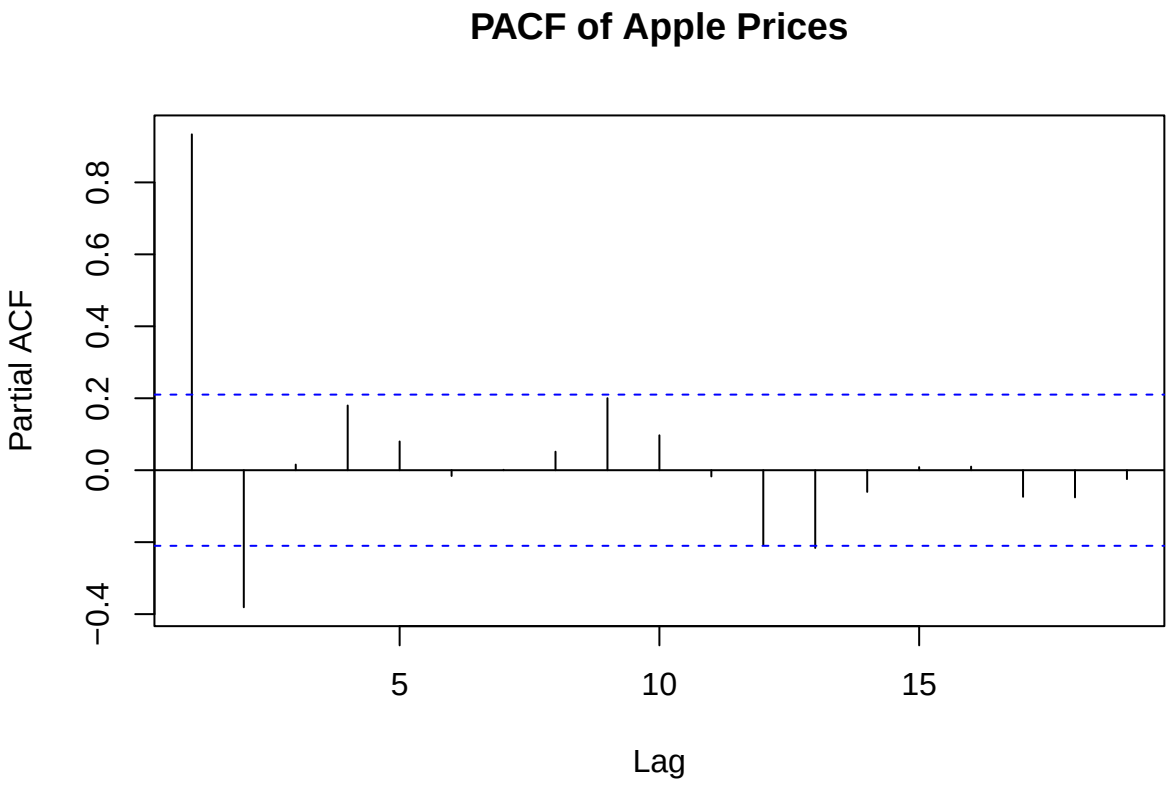


As expected, apple price seems to peak around midsummer each year, declining in the fall (likely due to the harvest increasing supply). This pattern becomes particularly pronounced in the last three full years of data. Therefore, we observe seasonality. What's more, we see a clear upward trend in apple price. This indicates that the mean is increasing over time, so the data cannot be covariance stationary.

```
# Take ACF and PACF of time series
acf(coredata(tspr),main="ACF of Apple Prices")
```




```
pacf(coredata(tspr),main="PACF of Apple Prices")
```

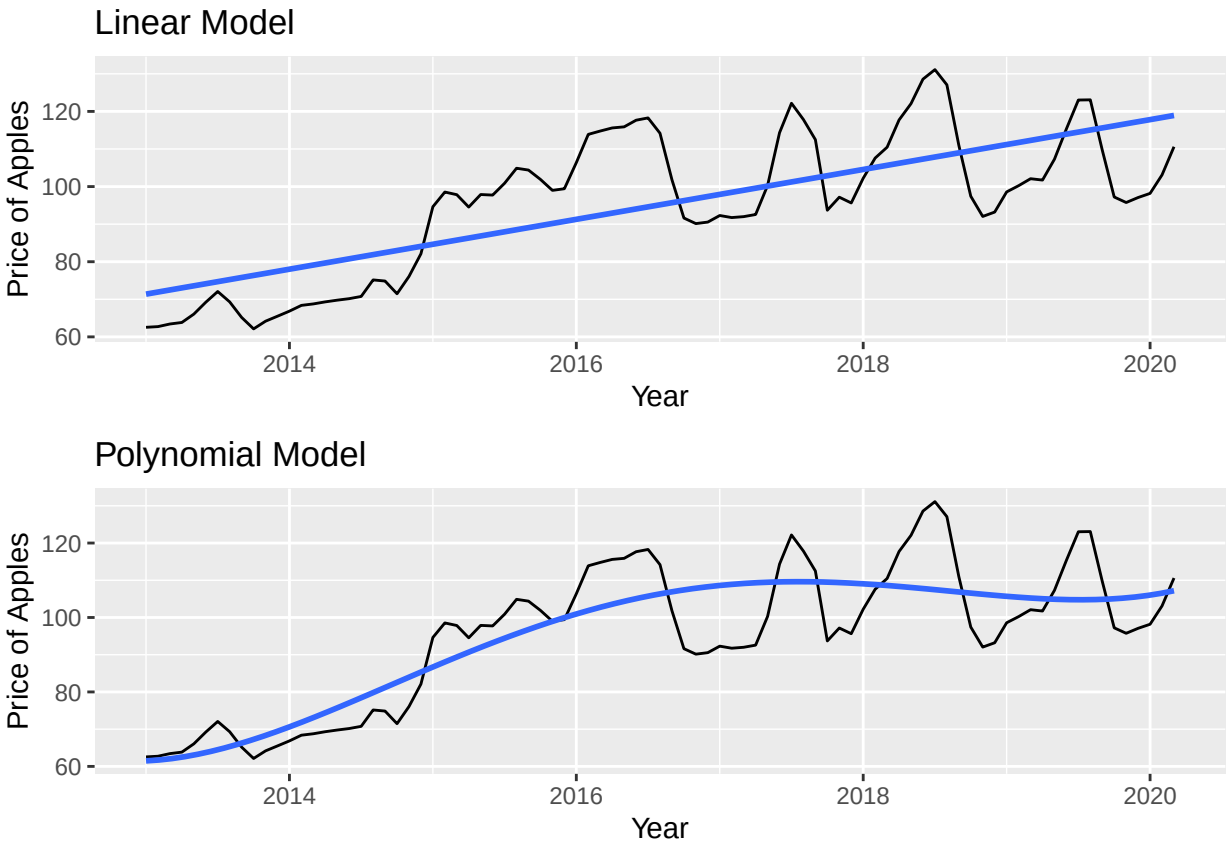


In the ACF, we observe several autocorrelations that are significantly non-zero. Therefore, the time series is non-random. However, the autocorrelations do decay to zero eventually. In the PACF, we see strong correlation at lags of 1 and 2 months, which indicates that there is a high degree of autocorrelation between adjacent and near-adjacent observations.

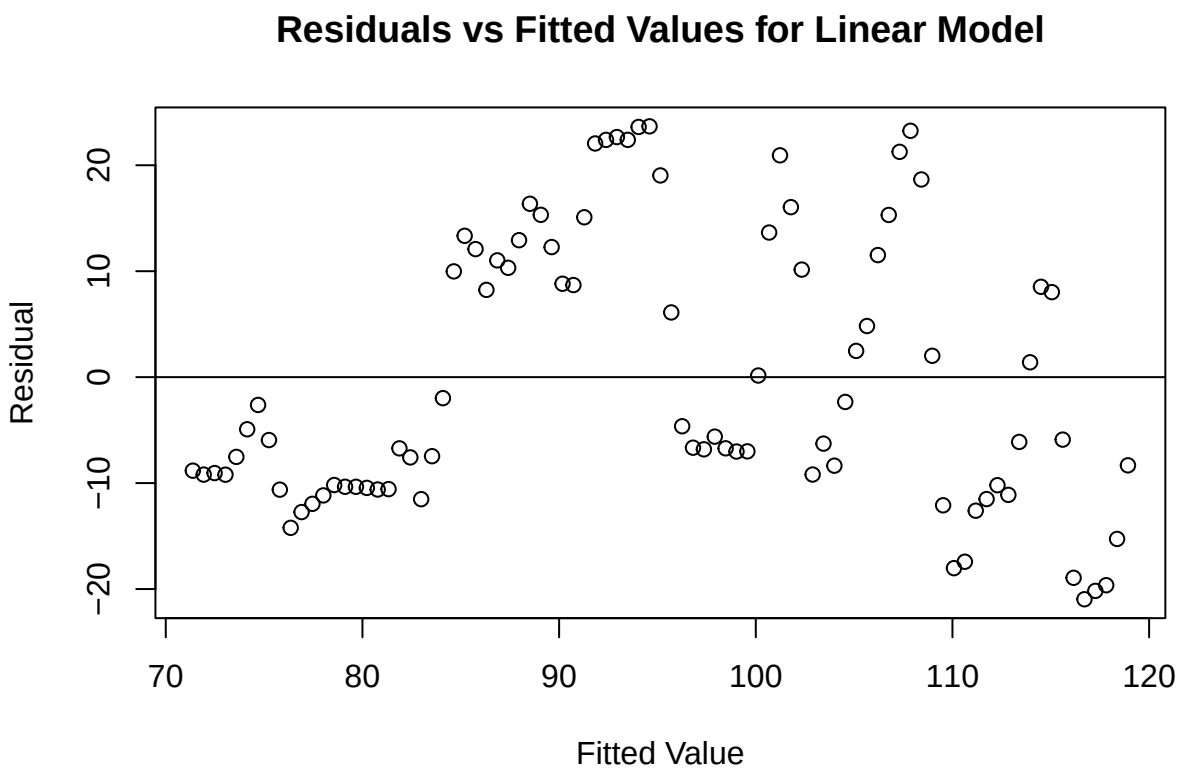
Fitting different models to the data

```
# Creating time data for both models
x1 <- seq(2013,24242/12,by=1/12)
df <- data.frame(tspr, x1)

# Fitting both a linear and a polynomial model to the data
reg1 <- lm (tspr ~ x1)
reg2 <- lm(tspr ~ poly(x1,4))

# Plotting each fit on top of the original time series, with both figures in the same window
require(gridExtra)
plot1<-ggplot(df, aes(x1, tspr)) +
      labs(title = "Linear Model",
        x = "Year",
        y = "Price of Apples") +
      geom_line() +
      geom_smooth(method = "lm", se = FALSE, formula = y ~ x)
```

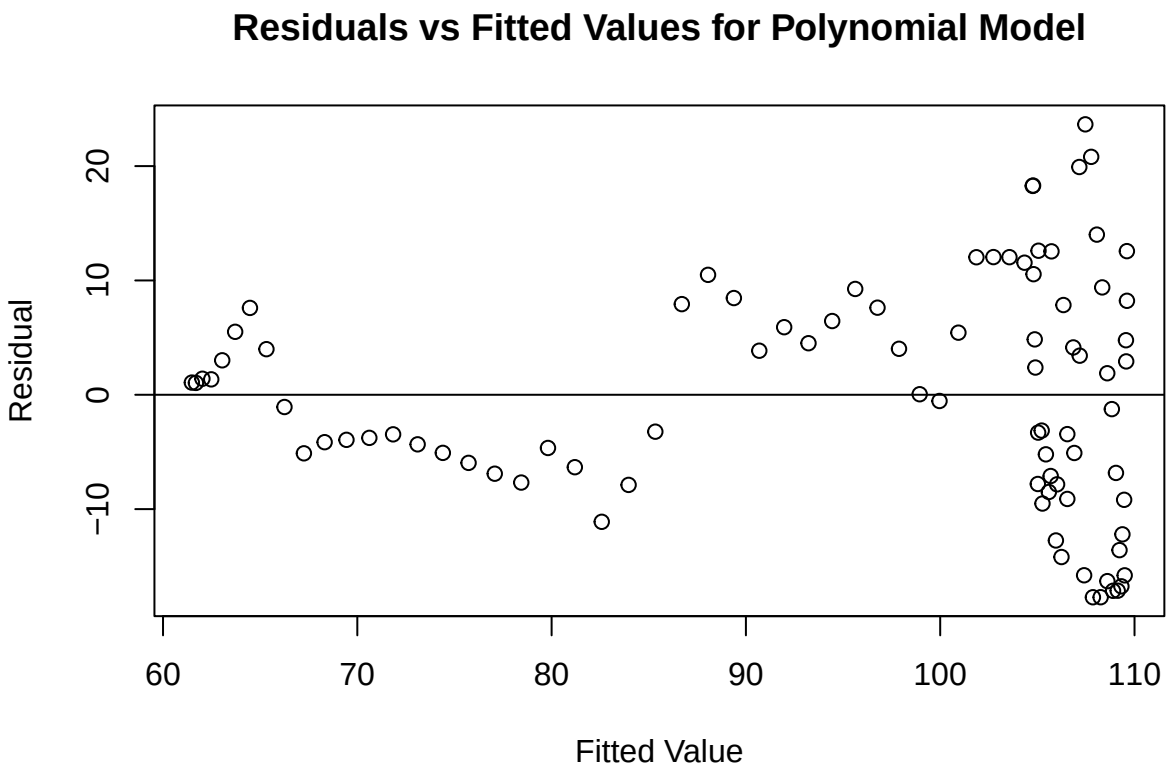
```
plot2<-ggplot(df, aes(x1, tspr)) +
      labs(title = "Polynomial Model",
        x = "Year",
        y = "Price of Apples") +
      geom_line() +
      geom_smooth(method = "lm", se = FALSE, formula = y ~ poly(x,4))

grid.arrange(plot1, plot2, nrow=2)
```




```
#residual vs fitted plot for linear model
res <- resid(reg1)
plot(fitted(reg1), res, main="Residuals vs Fitted Values for Linear Model", xlab="Fitted Value", ylab =

#add a horizontal line at 0
abline(0,0)
```

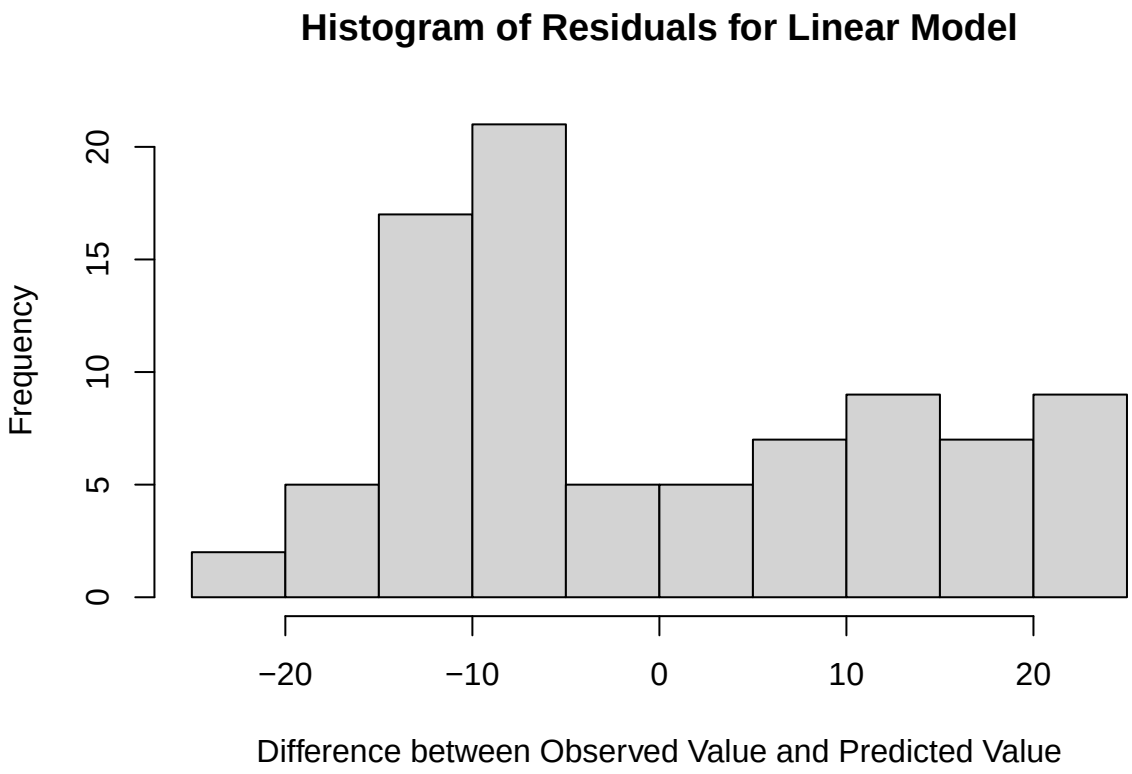



```
#residual vs fitted plot for polynomial model
res <- resid(reg2)
plot(fitted(reg2), res, main="Residuals vs Fitted Values for Polynomial Model", xlab="Fitted Value", yla

#add a horizontal line at 0
abline(0,0)
```

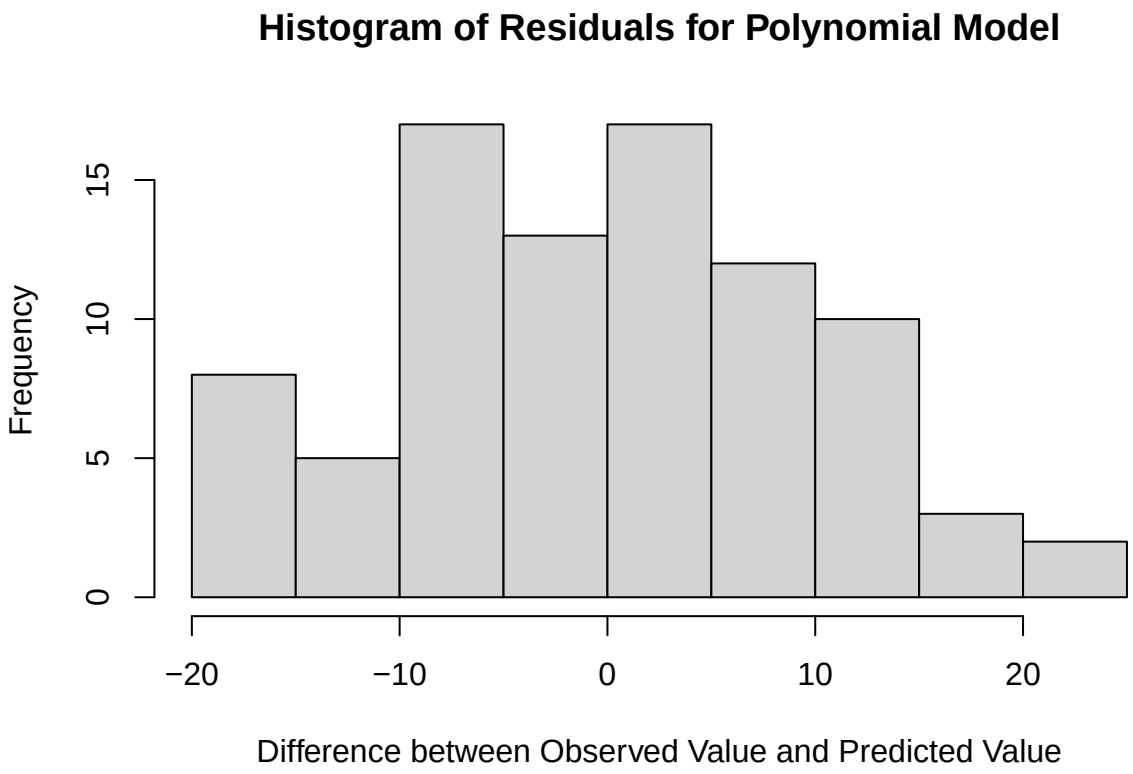


The residuals of the polynomial model have much higher variance at higher time values, which suggests that heteroskedasticity is present. The residuals of the linear model look like they might have a quadratic or otherwise polynomial trend, which suggests that a linear model might be poorly specified and a polynomial model might be preferred.

```
# Creating histograms of the residuals for each model
hist(resid(reg1), main = "Histogram of Residuals for Linear Model", xlab = "Difference between Observed
```




```
hist(resid(reg2), main = "Histogram of Residuals for Polynomial Model", xlab = "Difference between Obse
```



It appears that the residuals of the polynomial model are centered around zero and follow a roughly symmetric, bell-shaped distribution, which suggests that they are normally distributed, as are the model's error terms. In contrast, the residuals of the linear model are centered around -10 and are clearly right-skewed, which suggests that the error terms of the model are not normally distributed. Thus, we conclude that the linear model is likely not correctly specified.

```
summary(reg1)
```

```
##
## Call:
## lm(formula = tspr ~ x1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.962 -10.273  -5.897  11.273  23.673
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.328e+04  1.335e+03  -9.949 6.50e-16 ***
## x1           6.635e+00  6.621e-01  10.021 4.67e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.92 on 85 degrees of freedom
## Multiple R-squared:  0.5416, Adjusted R-squared:  0.5362
## F-statistic: 100.4 on 1 and 85 DF,  p-value: 4.668e-16
```

We note that the F-statistic of the linear model is 100.4, the t-value of its single variable is 10.021, which suggests that its estimated coefficient is more than 10 standard deviations away from zero. The adjusted R-squared is of the linear model is 0.5362, which means that only around 53.6% of the variation in the response variable, price, can be explained by the change in time.

```
summary(reg2)
```

```
##
## Call:
## lm(formula = tspr ~ poly(x1, 4))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.7088  -7.0043   0.0464   7.6067  23.6522
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    95.150      1.092  87.133  < 2e-16 ***
## poly(x1, 4)1  129.505     10.186  12.715  < 2e-16 ***
## poly(x1, 4)2  -72.428     10.186  -7.111 3.91e-10 ***
## poly(x1, 4)3   -3.494     10.186  -0.343   0.7324
## poly(x1, 4)4   20.775     10.186   2.040   0.0446 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.19 on 82 degrees of freedom
## Multiple R-squared:  0.7253, Adjusted R-squared:  0.7119
## F-statistic: 54.13 on 4 and 82 DF,  p-value: < 2.2e-16
```

Note that the t-statistic for the 3rd-degree term is -0.343, indicating that the term's coefficient is only 0.343 standard deviations from zero, and its associated p-value is 0.7324, which is large. Therefore, the F-statistic is influenced downward by this term, and is only 54.13, which is significantly less than the F-statistic of the linear model. However, realize that every other term has a t-value with absolute value greater than or equal to 2, and associated p-values less than or equal to 0.05, which suggests that their coefficients are all estimated to be significantly far from zero. So, the polynomial model has two more terms with significant explanatory power than the linear model; this explains why the adjusted R-squared value of this model is 0.7119, which is significantly greater than the adjusted R-squared value of the linear model.

Now we test each of our models using both the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC).

```
AIC(reg1, reg2)
```

```
##      df      AIC
## reg1  3 696.1492
## reg2  6 657.5962
```

```
BIC(reg1, reg2)
```

```
##      df      BIC
## reg1  3 703.5469
## reg2  6 672.3917
```

Both the AIC and BIC return a lower value for the polynomial model, so both tests agree in their preference for the polynomial model over the linear model. It's especially noteworthy that the BIC preferred the polynomial model as well, because the BIC penalizes the model's extra variables more heavily.

Forecasting Using the Polynomial Model

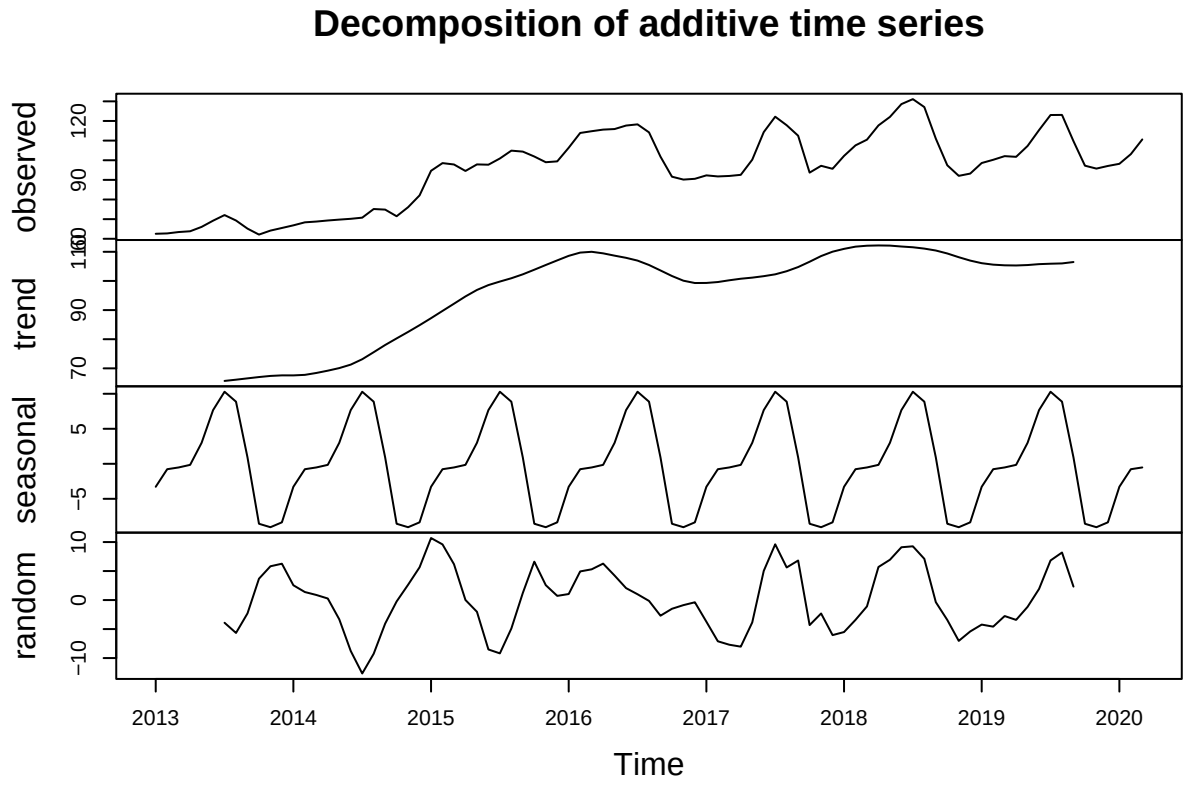
Now, we use the polynomial model to forecast 25 months ahead. The results can be seen below, with the lower and upper bounds of the respective 95% prediction interval for each prediction included.

```
# Create new time data to generate predictions for
steps_ahead <- data.frame(x1 = seq(2020.25, 2022.25, by=1/12))

# Create the forecast then print it
reg2_cast = predict(reg2, new=steps_ahead, interval = "prediction", level=0.95)
print(reg2_cast)
```

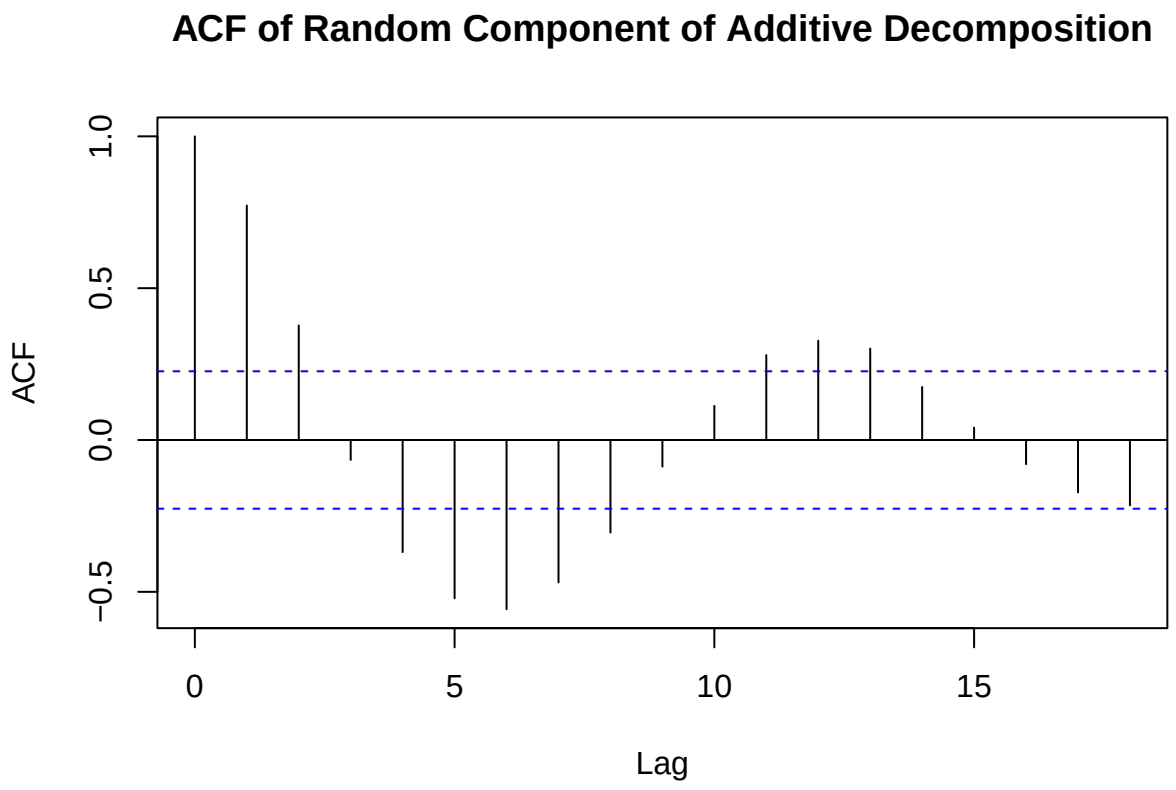
```
##         fit      lwr      upr
## 1  107.9388 84.56290 131.3147
## 2  108.8252 84.56062 133.0897
## 3  109.8482 84.49626 135.2001
## 4  111.0166 84.35874 137.6744
## 5  112.3393 84.13944 140.5391
## 6  113.8254 83.83243 143.8184
## 7  115.4844 83.43437 147.5344
## 8  117.3257 82.94424 151.7072
## 9  119.3591 82.36285 156.3553
## 10 121.5945 81.69243 161.4965
## 11 124.0420 80.93613 167.1479
## 12 126.7120 80.09763 173.3263
## 13 129.6149 79.18092 180.0489
## 14 132.7616 78.18996 187.3332
## 15 136.1628 77.12863 195.1970
## 16 139.8298 76.00060 203.6590
## 17 143.7738 74.80927 212.7383
## 18 148.0063 73.55777 222.4548
## 19 152.5390 72.24892 232.8291
## 20 157.3838 70.88531 243.8823
## 21 162.5529 69.46925 255.6365
## 22 168.0584 68.00280 268.1140
## 23 173.9129 66.48784 281.3379
## 24 180.1290 64.92605 295.3320
## 25 186.7197 63.31892 310.1204
```

Additive and Multiplicative Decomposition

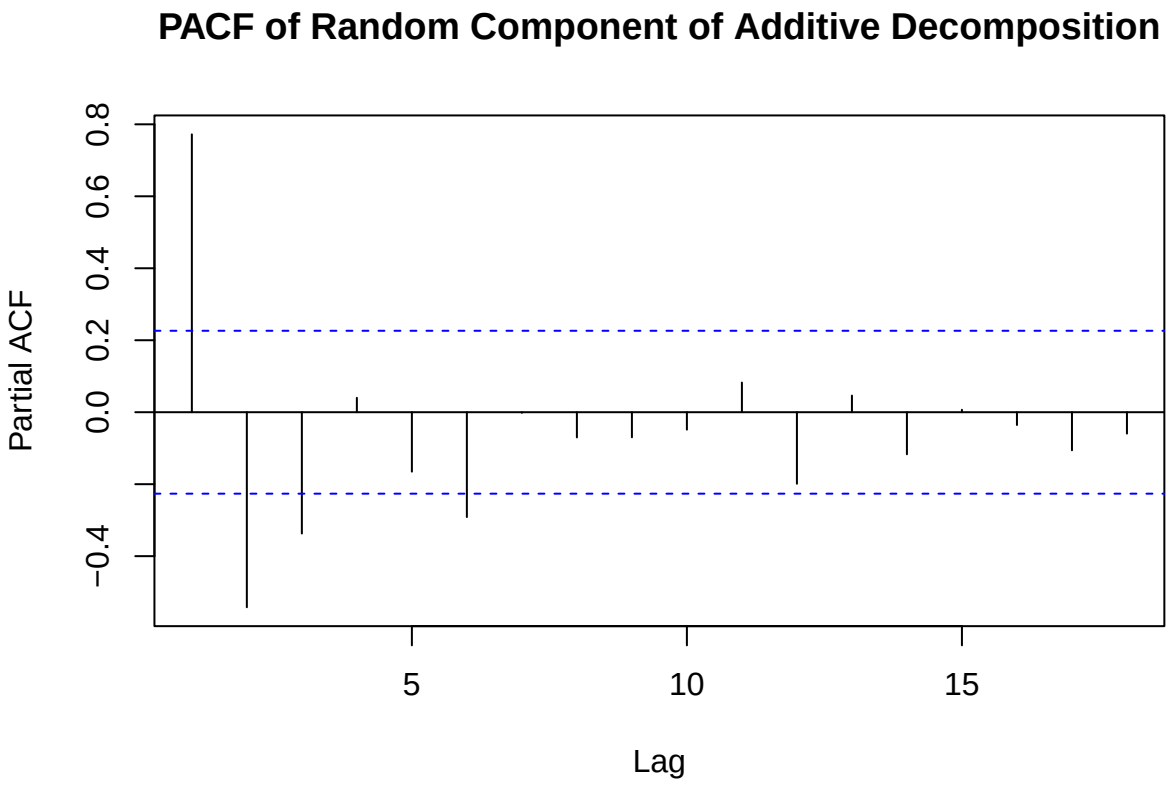
```
# Perform additive decomposition then plot it
dcmp_a <- decompose(tspr, "additive")
plot(dcmp_a)
```




```r
# Store the inidividual components
trend_a = dcmp_a$trend
seasonal_a = dcmp_a$seasonal
random_a = dcmp_a$random

# Remove trend and seasonality
detrend_seas_adj_a = tspr[7:81] - trend_a[7:81] - seasonal_a[7:81]

# Do ACF and PACF
acf(coredata(detrend_seas_adj_a), main= "ACF of Random Component of Additive Decomposition")
```



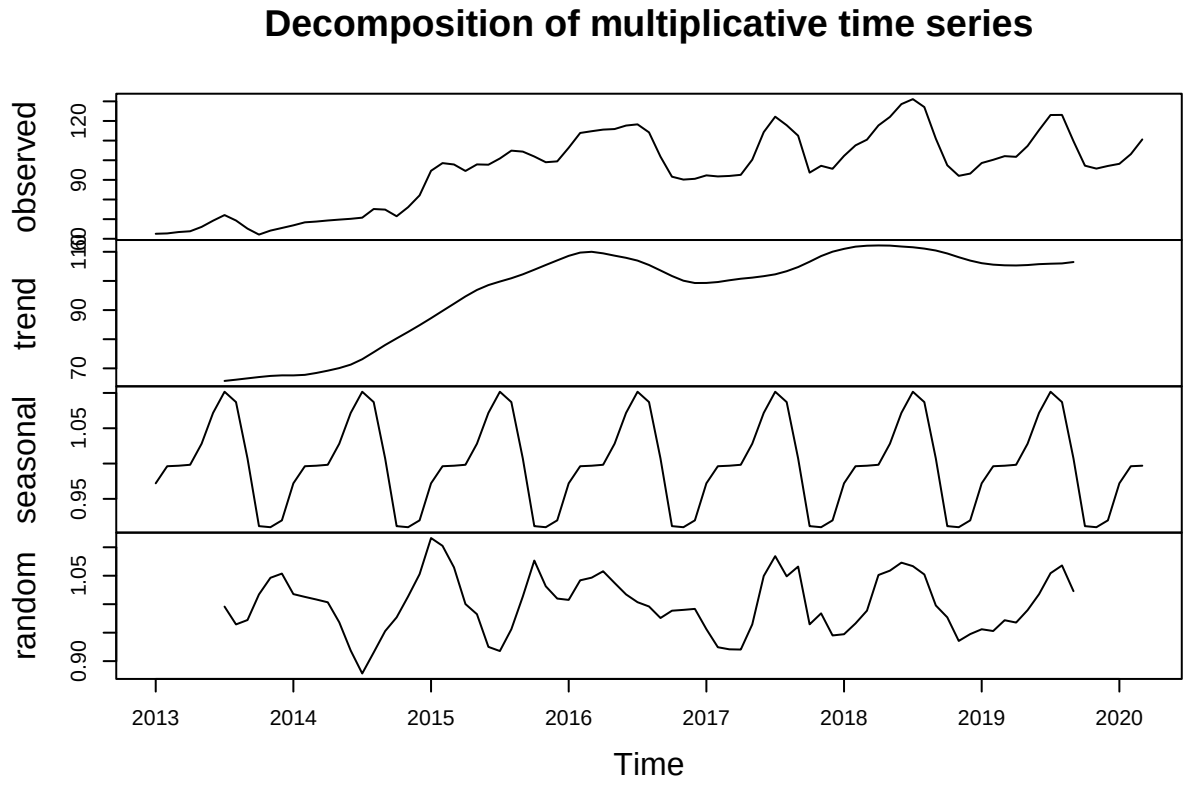

```
pacf(coredata(detrend_seas_adj_a), main="PACF of Random Component of Additive Decomposition")
```



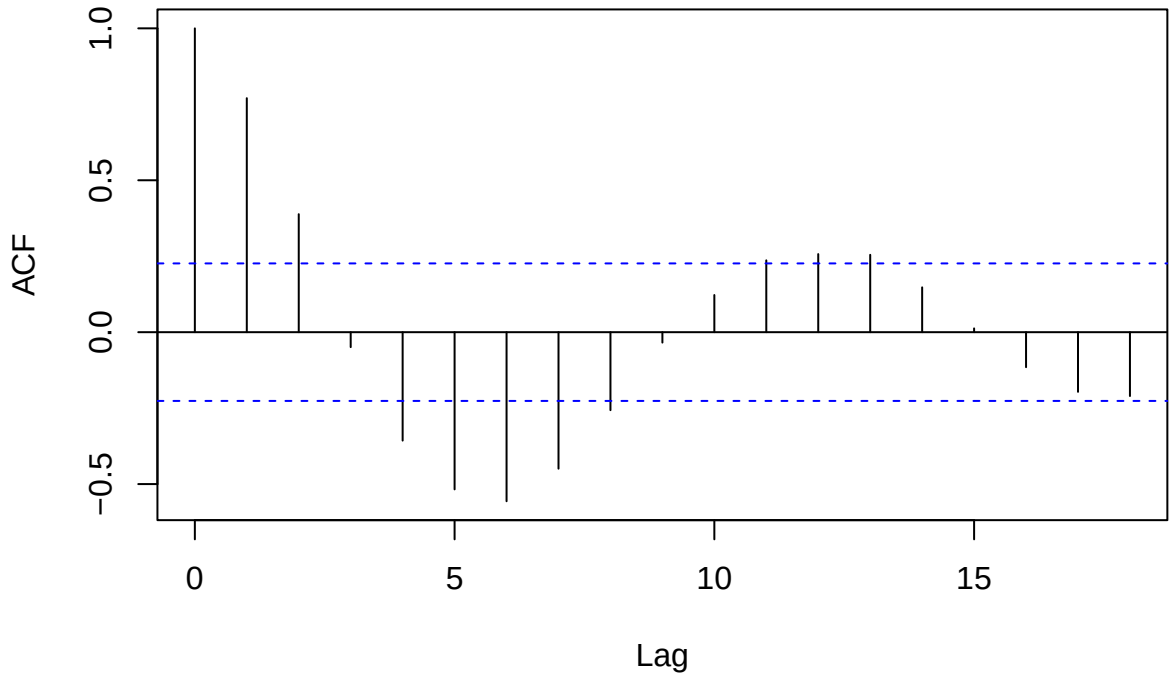
The ACF shows a significant amount of values above the significance threshold, so the time-series is not random. The autocorrelations appear to form a damped sine wave; they do eventually decay to zero. The PACF shows lags of 1, 2, 3, and 6 months are significant.

Next, we perform a multiplicative decomposition of our time series.

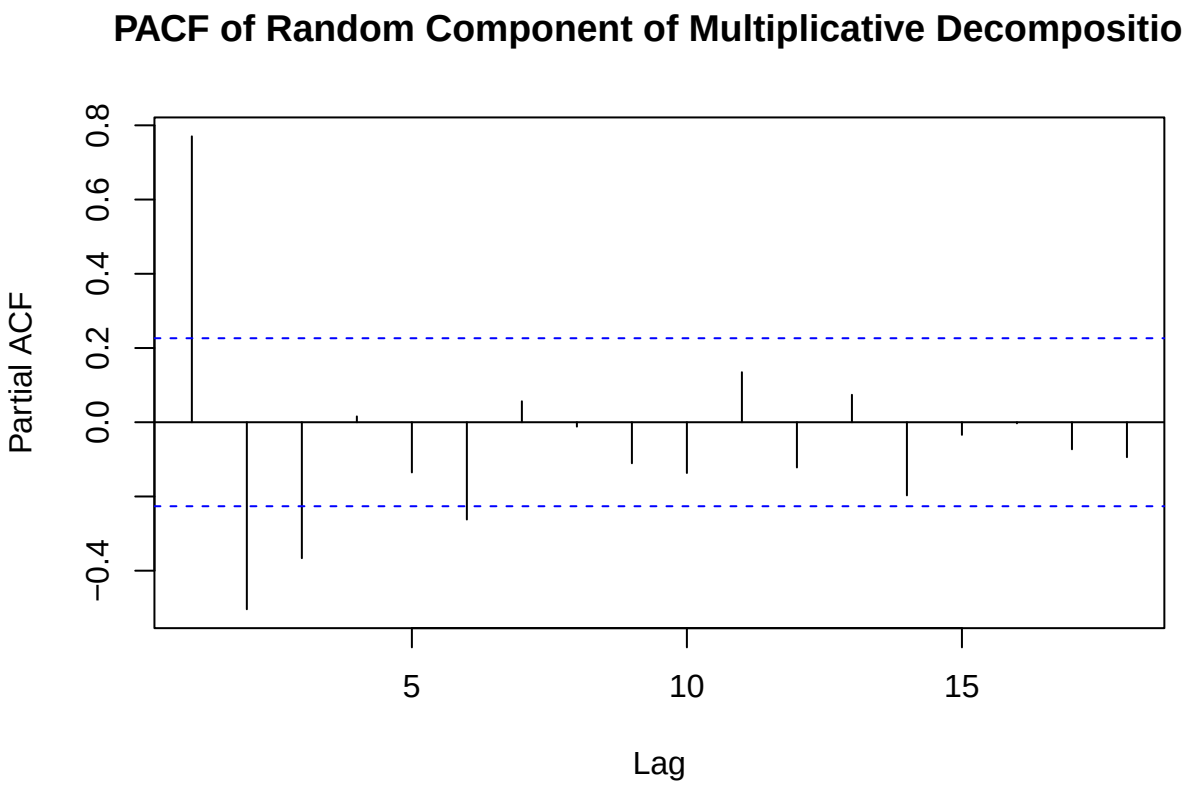
```
# Perform multiplicative decomposition then plot it
dcmp_m <- decompose(tspr, "multiplicative")
plot(dcmp_m)
```




```r
# Store the inidividual components
trend_m = dcmp_m$trend
seasonal_m = dcmp_m$seasonal
random_m = dcmp_m$random

# Remove trend and seasonality
detrend_seas_adj_m = (tspr[7:81]/trend_m[7:81])/seasonal_m[7:81]

# Do ACF and PACF
acf(coredata(detrend_seas_adj_m), main="ACF of Random Component of Multiplicative Decomposition")
```




```
pacf(coredata(detrend_seas_adj_m), main="PACF of Random Component of Multiplicative Decomposition")
```



The ACF and PACF of the random component of the multiplicative decomposition both look much the same as the ACF and PACF of the random component of the additive decomposition.

It appears from looking at the time-series plot on page 2 that a multiplicative decomposition would do well because the amplitude of the seasonality appears to increase over time. We can check whether this is supported by the respective Root Mean Square Error (RMSE) of each decomposition by noting that what is left after removing trend and seasonality are the residuals.

```
#RMSE for additive decomposition
rmse_a <- sqrt(sum(detrend_seas_adj_a^2)/length(detrend_seas_adj_a))
print(rmse_a)
```

```
## [1] 5.444131
```

```
#RMSE for multiplicative decomposition
rmse_m <- sqrt(sum(detrend_seas_adj_m^2)/length(detrend_seas_adj_m))
print(rmse_m)
```

```
## [1] 1.001393
```

By the above, we find that the RMSE of the additive decomposition is 5.444, while the RMSE of the multiplicative decomposition is 1.001. This means that, on average, the value predicted by the additive decomposition is off by more than 5, while the value predicted by the multiplicative decomposition is barely off by 1. So, we conclude that the multiplicative decomposition is better.

The residuals which are left over after removing trend and seasonality from each of our decompositions are very similarly distributed, as noted in our discussion of the ACF and PACF for each random component. Therefore, based on the decompositions, our models for the cycles would be very similar to each other.

Conclusions and Future Work

The fact that the multiplicative decomposition had a smaller RMSE than the additive decomposition suggests that seasonality has increasing variance over time. Therefore, a multiplicative model should perform better than an additive (i.e. linear) one. This is further supported by both the AIC and BIC preferring the polynomial model to the linear model. We conclude that the polynomial model is the better trend model. To further support this, we could perform a RESET Test to confirm that the linear model is poorly specified. The forecast generated by the polynomial model is reprinted here:

```
print(reg2_cast)
```

```
##         fit      lwr      upr
## 1  107.9388 84.56290 131.3147
## 2  108.8252 84.56062 133.0897
## 3  109.8482 84.49626 135.2001
## 4  111.0166 84.35874 137.6744
## 5  112.3393 84.13944 140.5391
## 6  113.8254 83.83243 143.8184
## 7  115.4844 83.43437 147.5344
## 8  117.3257 82.94424 151.7072
## 9  119.3591 82.36285 156.3553
## 10 121.5945 81.69243 161.4965
## 11 124.0420 80.93613 167.1479
## 12 126.7120 80.09763 173.3263
## 13 129.6149 79.18092 180.0489
## 14 132.7616 78.18996 187.3332
## 15 136.1628 77.12863 195.1970
## 16 139.8298 76.00060 203.6590
## 17 143.7738 74.80927 212.7383
## 18 148.0063 73.55777 222.4548
## 19 152.5390 72.24892 232.8291
## 20 157.3838 70.88531 243.8823
## 21 162.5529 69.46925 255.6365
## 22 168.0584 68.00280 268.1140
## 23 173.9129 66.48784 281.3379
## 24 180.1290 64.92605 295.3320
## 25 186.7197 63.31892 310.1204
```

We could also improve the analysis of the additive and multiplicative decompositions by using an STL decomposition rather than a classical one, as the classical decomposition can't provide trend data for the the first 6 or last 6 months. In an already small dataset, this could be a significant improvement. Of course, another potential improvement would be adding more years to the dataset, so there could be more data to analyze. Similarly, performing an analysis of apple prices in the other 4 cities and comparing them to one another could yield greater insight into apple price as well.

References

Russian Statistical Service. (2021). *Apples: Monthly Prices in Five Cities* [Data set]. https://www.kaggle.com/datasets/kapatsa/apple-prices-in-russian-regions