Time-Series Analysis

Ryan Schraeder

2022-04-17

Introduction

The dataset provided consists of housing market information for the United States in 1963, where I'll be diving into time series data to analyze and potentially forecast prices for homes after the time period of this data based upon observed patterns.

Importing the Data

```
df <- read_excel("/Users/rschraeder/Downloads/pricereg_cust.xls", sheet="Reg Price Qtr", col_names=c("pe
head(df)
```

```
## # A tibble: 6 x 6
##   period unitedstates northeast midwest south  west
##   <chr>         <dbl>     <dbl>   <dbl> <dbl> <dbl>
## 1 1963Q1        17800     20800   17500 16800 18000
## 2 1963Q2        18000     20600   17700 15800 18900
## 3 1963Q3        17900     19600   17800 15900 19000
## 4 1963Q4        18500     20600   19100 15800 19500
## 5 1964Q1        18500     20300   18700 16500 19600
## 6 1964Q2        18900     19800   19800 16800 20100
```

Null Value Counts

```
sum(is.na(df))
```

```
## [1] 0
```

Summary Statistics & Time Series Transformation

```
summary(df)
```

```
##     period           unitedstates      northeast         midwest
##  Length:227         Min.   : 17800   Min.   : 19600   Min.   : 17500
##  Class :character   1st Qu.: 47550   1st Qu.: 50550   1st Qu.: 49600
##  Mode  :character   Median :120000   Median :159900   Median :112900
```

```
##                       Mean   :134057   Mean   :183903   Mean   :126028
##                       3rd Qu.:219250   3rd Qu.:301000   3rd Qu.:194300
##                       Max.   :337900   Max.   :566500   Max.   :300300
##      south              west       
##  Min.   : 15800   Min.   : 18000  
##  1st Qu.: 42850   1st Qu.: 52450  
##  Median :100900   Median :136000  
##  Mean   :118816   Mean   :159137  
##  3rd Qu.:189000   3rd Qu.:259350  
##  Max.   :298500   Max.   :423400  
```

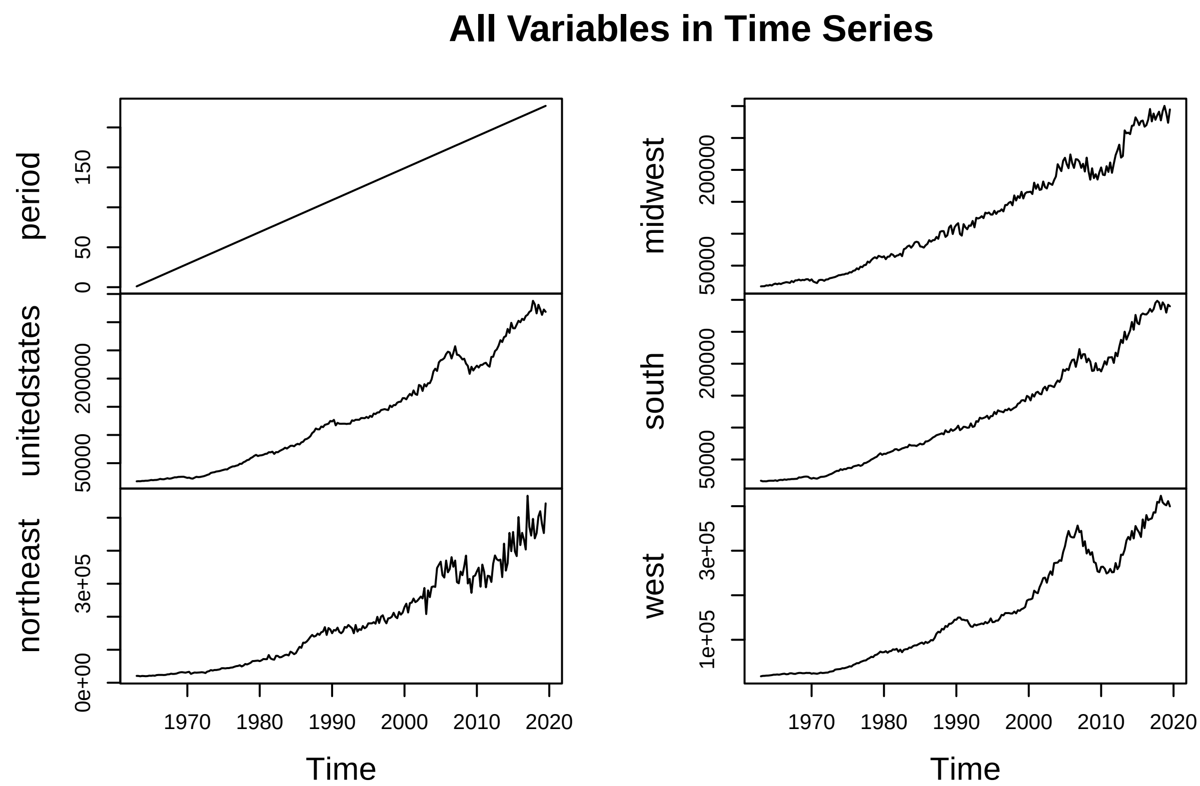
The summary statistics suggest that the means are much larger than the median, indicating high variance and potential for strong trends.

```r
# Convert columns to datetime/time series by the period column.
df_ts <- ts(df, start=c(1963,1), frequency=4)
```

After converting the data to time series, we can observe plots for trends, seasonality, and any notable patterns.

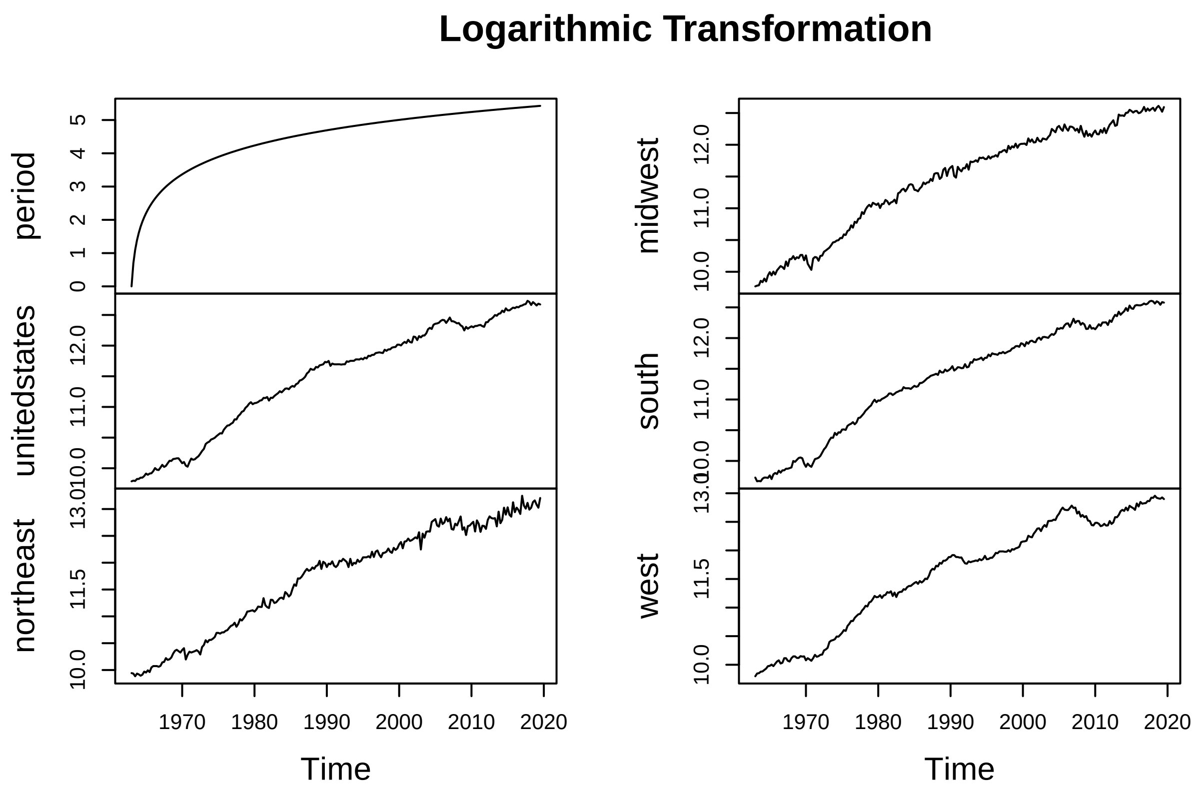
Linear Plot

```r
plot(
  df_ts,
  main="All Variables in Time Series"
)
```



Log Transform

```r
log_df<-log(df_ts)
plot(log_df, main="Logarithmic Transformation")
```



Based upon the linear plots, a similar trend can be observed throughout the data from the midwest, south, and west regions of the United States. A drop occurs after 2010 and climbs after 2015. In the log transformation, a smooth upward trend can be observed similarly across all variables.

Stationarity

To best understand the data, a stationary set will need to be utilized. Testing the data for stationarity will accomplish value for forecasting.

```
# Manual ADF Tests by Column. Tests for Stationarity by P-Value for a Test-Statistic.
adf.test(log_df[,2])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag  ADF p.value
## [1,]   0 6.46    0.99
## [2,]   1 7.04    0.99
## [3,]   2 6.06    0.99
## [4,]   3 4.50    0.99
## [5,]   4 3.44    0.99
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -2.16  0.2642
## [2,]   1 -2.55  0.1142
```

```
## [3,]   2 -2.60  0.0969
## [4,]   3 -2.22  0.2402
## [5,]   4 -2.05  0.3064
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -0.807   0.960
## [2,]   1 -0.525   0.980
## [3,]   2 -0.608   0.976
## [4,]   3 -0.819   0.959
## [5,]   4 -1.102   0.920
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(log_df[,3])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag  ADF p.value
## [1,]   0 2.32    0.99
## [2,]   1 3.87    0.99
## [3,]   2 5.12    0.99
## [4,]   3 4.85    0.99
## [5,]   4 5.64    0.99
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -1.04   0.681
## [2,]   1 -1.16   0.641
## [3,]   2 -1.49   0.524
## [4,]   3 -1.53   0.510
## [5,]   4 -1.78   0.415
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -3.64  0.0304
## [2,]   1 -2.02  0.5651
## [3,]   2 -1.39  0.8333
## [4,]   3 -1.37  0.8402
## [5,]   4 -1.05  0.9291
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(log_df[,4])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag  ADF p.value
## [1,]   0 3.27    0.99
## [2,]   1 4.71    0.99
## [3,]   2 5.20    0.99
## [4,]   3 5.87    0.99
```

```
## [5,]   4 4.96    0.99
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -1.62  0.4780
## [2,]   1 -2.07  0.3011
## [3,]   2 -2.36  0.1862
## [4,]   3 -2.63  0.0928
## [5,]   4 -2.53  0.1195
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -2.44   0.392
## [2,]   1 -1.60   0.744
## [3,]   2 -1.38   0.837
## [4,]   3 -1.03   0.931
## [5,]   4 -1.14   0.915
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
adf.test(log_df[,5])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag  ADF p.value
## [1,]   0 5.28    0.99
## [2,]   1 6.79    0.99
## [3,]   2 6.69    0.99
## [4,]   3 5.19    0.99
## [5,]   4 3.93    0.99
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -1.59   0.489
## [2,]   1 -2.26   0.225
## [3,]   2 -2.57   0.103
## [4,]   3 -2.50   0.133
## [5,]   4 -2.16   0.265
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -1.196   0.905
## [2,]   1 -0.817   0.959
## [3,]   2 -0.726   0.967
## [4,]   3 -1.019   0.934
## [5,]   4 -1.204   0.904
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```r
adf.test(log_df[,6])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
```

```
##      lag  ADF p.value
## [1,]   0 5.16    0.99
## [2,]   1 6.03    0.99
## [3,]   2 4.74    0.99
## [4,]   3 4.03    0.99
## [5,]   4 3.10    0.99
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -1.75   0.426
## [2,]   1 -1.94   0.353
## [3,]   2 -1.75   0.424
## [4,]   3 -1.65   0.467
## [5,]   4 -1.52   0.515
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -1.172   0.909
## [2,]   1 -0.771   0.963
## [3,]   2 -0.977   0.940
## [4,]   3 -1.106   0.920
## [5,]   4 -1.446   0.808
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

Among all tests, no P-values consistently prove stationarity. So, the data needs to be transformed with a differencing method for smoothing, or in other words, removing any trends for consistent data. I'm going to use difference transform, since the upward trend is very aggressive for this data.

Smoothing

```r
smoothed_df<-diff(log_df, lag=3, differences=1) ## Getting differences by quarter (lag = 3 or 3 months)
adf.test(smoothed_df[,2])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -4.98    0.01
## [2,]   1 -3.63    0.01
## [3,]   2 -4.03    0.01
## [4,]   3 -2.63    0.01
## [5,]   4 -2.72    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -6.79    0.01
## [2,]   1 -5.05    0.01
## [3,]   2 -5.80    0.01
## [4,]   3 -3.78    0.01
## [5,]   4 -3.99    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
```

```
## [1,]   0 -7.12    0.01
## [2,]   1 -5.38    0.01
## [3,]   2 -6.18    0.01
## [4,]   3 -4.14    0.01
## [5,]   4 -4.36    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(smoothed_df[,3])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -10.57    0.01
## [2,]   1  -7.33    0.01
## [3,]   2  -8.11    0.01
## [4,]   3  -5.81    0.01
## [5,]   4  -3.97    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -12.30    0.01
## [2,]   1  -8.95    0.01
## [3,]   2 -10.74    0.01
## [4,]   3  -8.26    0.01
## [5,]   4  -5.89    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -12.41    0.01
## [2,]   1  -9.08    0.01
## [3,]   2 -10.97    0.01
## [4,]   3  -8.54    0.01
## [5,]   4  -6.14    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(smoothed_df[,4])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -9.41    0.01
## [2,]   1 -6.43    0.01
## [3,]   2 -6.76    0.01
## [4,]   3 -3.84    0.01
## [5,]   4 -3.34    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -11.35    0.01
## [2,]   1  -8.09    0.01
```

```
## [3,]   2  -9.07    0.01
## [4,]   3  -5.27    0.01
## [5,]   4  -4.58    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -11.68    0.01
## [2,]   1  -8.41    0.01
## [3,]   2  -9.53    0.01
## [4,]   3  -5.62    0.01
## [5,]   4  -4.88    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
adf.test(smoothed_df[,5])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -5.93    0.01
## [2,]   1 -4.03    0.01
## [3,]   2 -4.44    0.01
## [4,]   3 -3.10    0.01
## [5,]   4 -3.16    0.01
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -8.20    0.01
## [2,]   1 -5.67    0.01
## [3,]   2 -6.46    0.01
## [4,]   3 -4.60    0.01
## [5,]   4 -4.87    0.01
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -8.60    0.01
## [2,]   1 -5.99    0.01
## [3,]   2 -6.83    0.01
## [4,]   3 -4.95    0.01
## [5,]   4 -5.29    0.01
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

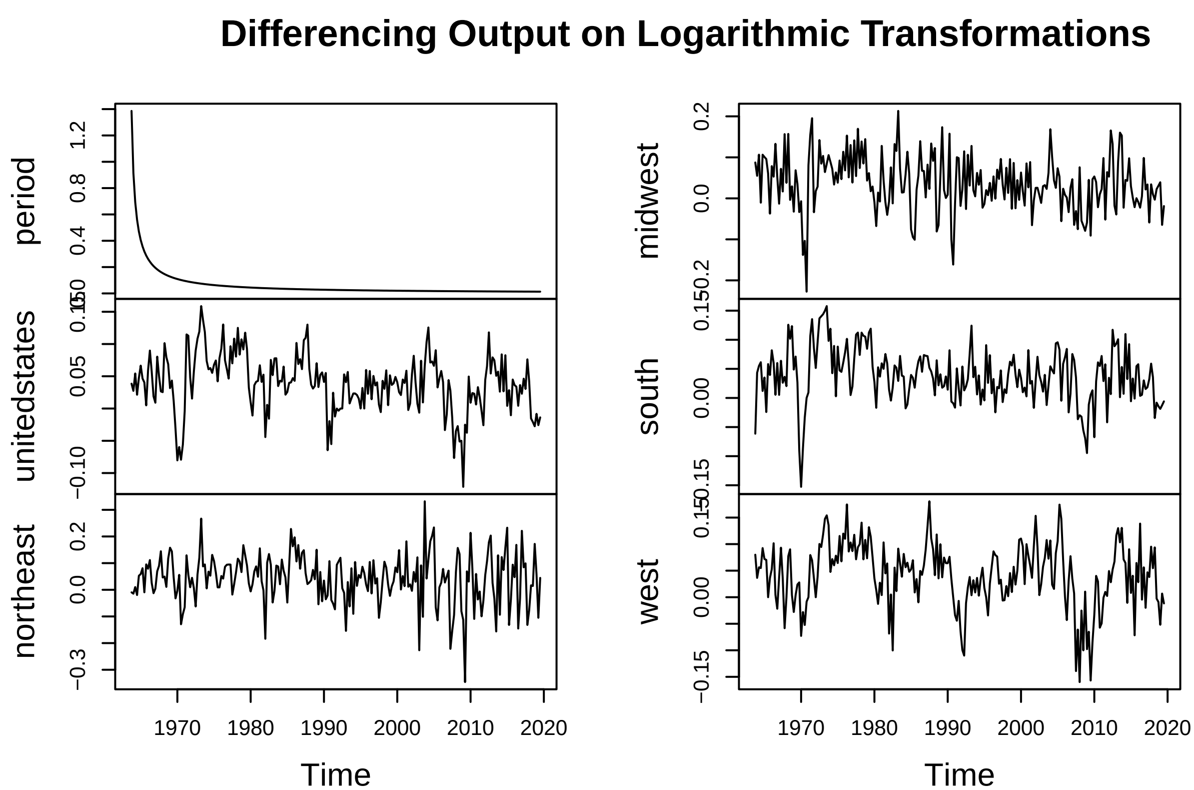
```
adf.test(smoothed_df[,6])
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag   ADF p.value
## [1,]   0 -5.98    0.01
## [2,]   1 -3.94    0.01
## [3,]   2 -4.85    0.01
## [4,]   3 -2.78    0.01
```

```
## [5,]   4 -2.72    0.01
## Type 2: with drift no trend
##       lag   ADF p.value
## [1,]   0 -7.40    0.01
## [2,]   1 -4.95    0.01
## [3,]   2 -6.38    0.01
## [4,]   3 -3.55    0.01
## [5,]   4 -3.48    0.01
## Type 3: with drift and trend
##       lag   ADF p.value
## [1,]   0 -7.54  0.0100
## [2,]   1 -5.07  0.0100
## [3,]   2 -6.54  0.0100
## [4,]   3 -3.67  0.0276
## [5,]   4 -3.58  0.0353
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

The P-Values are revealing a much more sound set of variables with stationarity. The data can be forecasted with confidence, and a further glance of the patterns can be observed in another plot matrix:

```
plot.ts(smoothed_df, main="Differencing Output on Logarithmic Transformations")
```

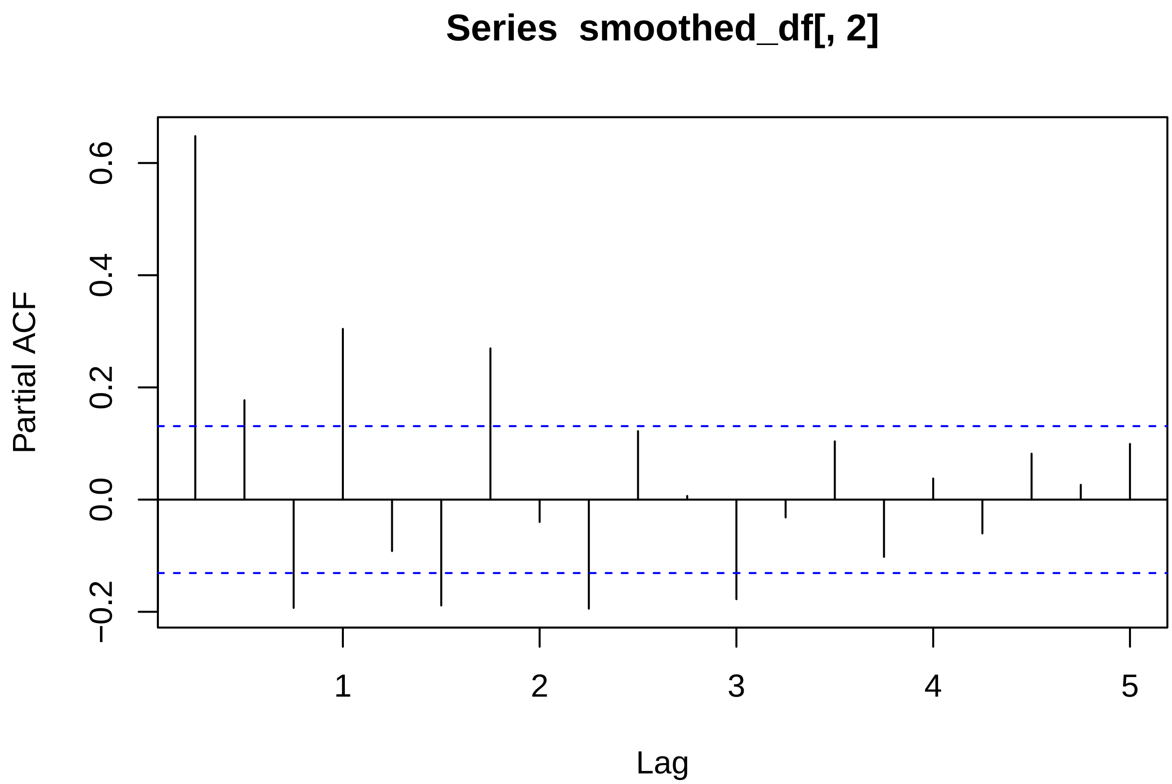


Forming the Time-Series Forecast

The behavior of the data is consistent, and less influenced by time. This indication of stationary data is perfect for a forecasting model, in which the correct values for an ARIMA (Autoregressive Integrated Moving

Average) model may be selected and proper simulation of this moving average can be used to predict a future trend. In this case, I want to see where the housing market may be in each region with accordance to past differences.

```
pacf(smoothed_df[,2], lag.max=20) # plot
```




```
pacf(smoothed_df[,2], lag.max=20, plot=FALSE) ## values only
```

```
##
## Partial autocorrelations of series 'smoothed_df[, 2]', by lag
##
##   0.25   0.50   0.75   1.00   1.25   1.50   1.75   2.00   2.25   2.50   2.75
##  0.648  0.177 -0.193  0.304 -0.092 -0.189  0.270 -0.040 -0.194  0.122  0.007
##   3.00   3.25   3.50   3.75   4.00   4.25   4.50   4.75   5.00
## -0.178 -0.032  0.104 -0.102  0.038 -0.060  0.082  0.026  0.099
```

In this correlogram, the key focus is to pay attention to the dotted blue lines, which represent significance boundaries. If within those boundaries, averages can be considered statistically significant. Thus, the significant values may be selected as the orders for the ARIMA. Since the values are zero after roughly lag 2.5, a fair order could be an ARMA(2.5,0) selection. To make this easier, I'll use the `auto.arima()` function from the forecast library.

```
fit<-auto.arima(smoothed_df[,2])
arima<-arima(smoothed_df[,2], order=c(3,0,1))
forecast_test<-forecast(arima, h=4, level=c(99.5))
```

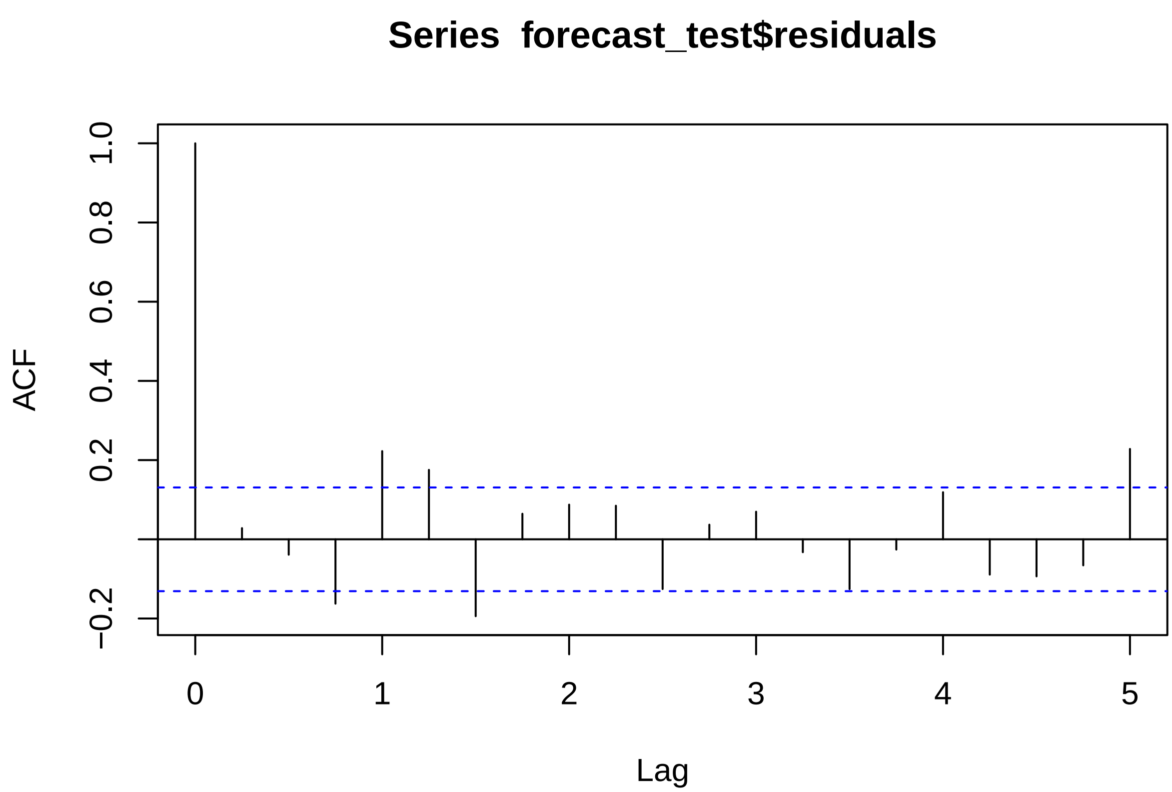
The selection for the model being ARIMA(0,1,3) falls within bounds of the witnessed plot, where the difference of lag 2.5 and 3 showed the nearest to zero. The most accurate assumption here is that the model will forecast within bounds of each, and what this specifically means is the difference in housing price over 1-3 months can dictate the most accurate price in a a future estimate. Being the ARIMA uses a moving average, the prediction can be widely trusted with the reduction of "noise" and consistent pattern.

To ensure the significance bounds are themselves trustworthy, I will use the "Ljung-Box" test to plot residuals.

```
Box.test(smoothed_df[,2], lag=20, type="Ljung-Box")
```

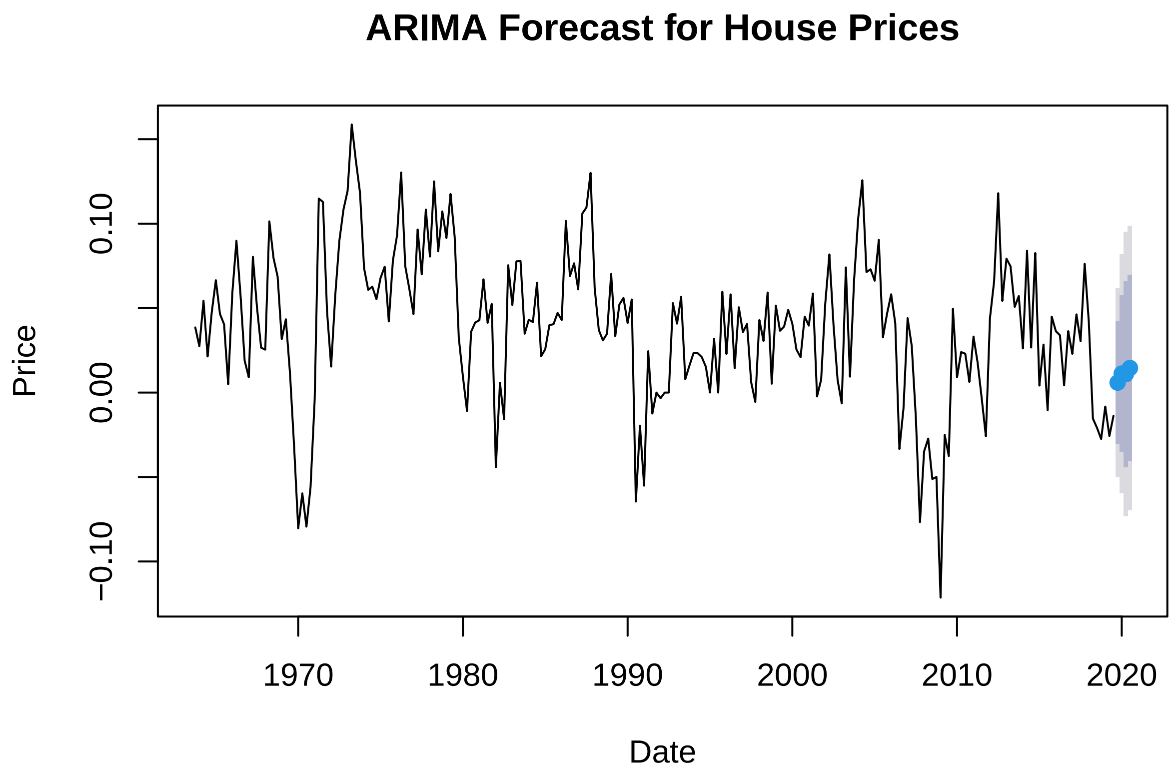
```
##
##  Box-Ljung test
##
## data:  smoothed_df[, 2]
## X-squared = 246.33, df = 20, p-value < 2.2e-16
```

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



The residuals are much cleaner, showing most values are within bounds and consistent. This indicates a reliable forecasting model that can be used to generate predictions and be related back to the original data.

```
plot(forecast(fit, 4), xlab="Date", ylab="Price", main="ARIMA Forecast for House Prices")
```



Clarifying Assumptions

Given the log transform process and differencing method, the data was much easier to work with as stationary for an ARIMA model being the best case. Selecting the United States overall as a target variable allowed for much more information that can surmise a trustworthy forecast. The increase in price after 2020 appears accurate as we know, and certainly the housing crash of 2008 is evident. The dates are behind in this data and the predictive interval ARIMA calculates is not robust enough to extend so far into the future (even to present day). If I were to continue forward, I'd check back in 4 months and retrieve updated data, then potentially try the keener stochastic models Holt-Winters' exponential smoothing provides. I have noticed ARIMA is far complex but very accurate if given the correct transformations, however. This has been very entertaining and I'm excited for more!

References

- https://otexts.com/fpp2/stationarity.html
- https://www.statology.org/dickey-fuller-test-in-r/
- https://machinelearningmastery.com/remove-trends-seasonality-difference-transform-python/
- https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html