Housing Market Forecasting Guide

This guide will help users to make forecasts using a variety of different methods, including Exponential Smoothing (ETS), Autoregressive Integrated Moving Averages (ARIMA), OLS Regression, Dynamic Regression (Regression with ARIMA elements), Prophet models, Neural-ARIMA models, and Bootstrapping and Bagging.

Organization

This forecasting file is organized into three folders:

- Data
- Data Processing
- Research

The "Data" folder contains the raw, unmodified data files to be used as inputs in the scripts found in the folder "Data Processing". "Research" contains some of the academic literature consulted to make our models and forecasts.

Within the Data Processing folder are the inputs, scripts, outputs, and visualizations of our analyses organized as such:

- Input
- Output
- Scripts

As mentioned prior, "Input" contains the raw data to be modified and cleaned for our analysis. The cleaned datasets are saved to "Output", as well as the visualizations of our forecasts.

Scripts contain folders and scripts that are numbered which indicate the run order.

- 00 - Libraries
- 01 - Imports and Merges
- 02 - Data Preparation
- 03 - Analysis

Scripts within "00 - Libraries", "01 - Imports and Merges", and "02 - Data Preparation" must be run in its numerical order, whereas scripts in "03 - Analysis" can be ran in any order.

REMEMBER! "00 - Libraries" must be ran first!

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lubridate)
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(fpp3)
```

```
## -- Attaching packages ---------------------------------------------- fpp3 0.5 --
## v tsibble     1.1.4     v fable       0.3.4
## v tsibbledata 0.4.1     v fabletools  0.4.2
## v feasts      0.3.2
## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x car::recode()        masks dplyr::recode()
## x tsibble::setdiff()   masks base::setdiff()
## x car::some()          masks purrr::some()
## x tsibble::union()     masks base::union()
```

```
library(np)
```

```
## Nonparametric Kernel Methods for Mixed Datatypes (version 0.60-17)
## [vignette("np_faq",package="np") provides answers to frequently asked questions]
## [vignette("np",package="np") an overview]
## [vignette("entropy_np",package="np") an overview of entropy-based methods]
```

```
library(fable.prophet)
```

```
## Loading required package: Rcpp
```

A word on the data

Our analysis and models are built primarily on data split into three subsets: Annual, Monthly, and Quarterly series. There are a wide array of explanatory variables explored in our analyses, but each variable contains a different time index. As such, all explanatory variables are averaged and aggregated into the Annual dataset, but this can severely hamper the utility of our forecasts. Due to this, most of our analysis is built on Monthly and Quarterly data.

```
#Annual
Master = read.csv("C:/Users/crite/OneDrive/Documents/Ivory Innovations/Home Price Forecasting/Data Proc

Master_tsibble = Master %>%
  mutate(DATE = year(DATE)) %>%
  as_tsibble(index = DATE)

Master_tsibble = Master_tsibble %>% select(-X.1,-X.y,-X)

#Monthly
Monthly = read.csv("C:/Users/crite/OneDrive/Documents/Ivory Innovations/Home Price Forecasting/Data Pro
Monthly = Monthly %>%
  mutate(DATE = yearmonth(DATE)) %>%
  as_tsibble(index = DATE) %>%
  select(-X)

#Monthly set with no NAs (for models and graphs that require complete datasets)

Monthly_NoNA = na.omit(Monthly)

#Quarterly
Quarterly = read.csv("C:/Users/crite/OneDrive/Documents/Ivory Innovations/Home Price Forecasting/Data Pr
Quarterly = Quarterly %>%
  mutate(DATE = yearquarter(DATE)) %>%
  as_tsibble(index = DATE) %>%
  select(-X)

#Quarterly set with no NAs (for models and graphs that require complete datasets)
Quarterly_NoNA = na.omit(Quarterly)
#Quarterly set with no NAs removing S&P500 series (as it's the most limiting range of our data)
Quarterly_NoNA_NoSP = Quarterly %>% select(-SP500)
Quarterly_NoNA_NoSP = na.omit(Quarterly_NoNA_NoSP)
```

Models

Exponential Smoothing (ETS)

Exponential Smoothing (ETS) is a simple but effective forecasting method by which moving averages are created, but more recent values hold more weight.

The model can be easily created using the following function found in the script "02 - Exponential Smoothing Model".

```
ETS_forecaster = function(Dataset,Variable, Horizon, Title, Subtitle, xlab, ylab) {
  variable_sym = ensym(Variable)

  Set_Name = Dataset %>%
    select(DATE, !!variable_sym) %>%
    filter(!is.na(!!variable_sym))

  Model = Set_Name %>%
    model(ETS(!!variable_sym))

  Forecast = Model %>%
    forecast(h = Horizon)

  Model_Specs = Model %>%
    report()

  Plot = Forecast %>%
    autoplot(Set_Name) +
    geom_line(aes(y = .fitted), col = "#D55E00", data = augment(Model)) +
    ggtitle(Title, Subtitle) +
    xlab(xlab) + ylab(ylab)

  list(Model = Forecast, Plot = Plot)
}
```

The function's arguments are:

- ***Dataset*** **-** Selects the dataset to be used for the ETS analysis.
- ***Variable*** **-** Selects the variable to be forecasted by the ETS model.
- ***Horizon*** **-** Determines the number of steps into the future the data is to be forecasted. For example, to forecast monthly Housing Starts two years into the future, the horizon would be 24. Forecasts give more confident values if the horizon is smaller.
- ***Title*** and ***Subtitle*** **-** Gives a title and subtitle to the plot of the ETS model.
- ***xlab*** and ***ylab*** - Gives headers for the x and y axes to the plot of the ETS model.

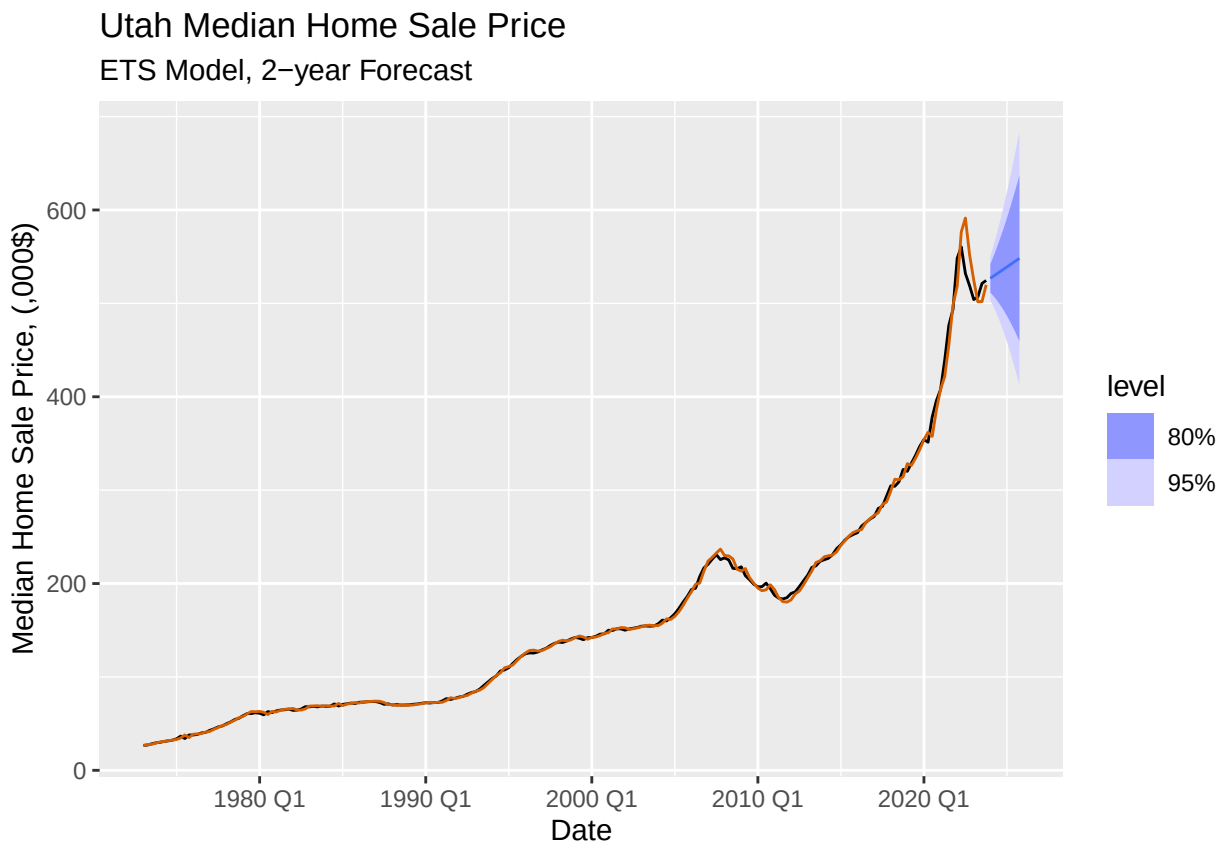
The function will automatically create the most well-fitting model as well as accompanying graphs. Fitted values from the model will be represented with the orange-colored series.

For the forecasting of Utah Median Home Sale Prices:

```
Median_Sales_Price_Model_ETS = ETS_forecaster(Quarterly,
                                              "Median_Sale_Price_UT",
                                              8,
                                              "Utah Median Home Sale Price",
                                              "ETS Model, 2-year Forecast",
                                              "Date",
                                              "Median Home Sale Price, (,000$)")
```

```
## Series: Median_Sale_Price_UT
## Model: ETS(M,A,N)
##   Smoothing parameters:
##     alpha = 0.838013
##     beta  = 0.3006527
##
##   Initial states:
##     l[0]      b[0]
##  25.7794 0.8223577
##
##   sigma^2:  5e-04
##
##      AIC     AICc      BIC
## 1511.820 1512.123 1528.411
```

```
Median_Sales_Price_Model_ETS$Plot
```

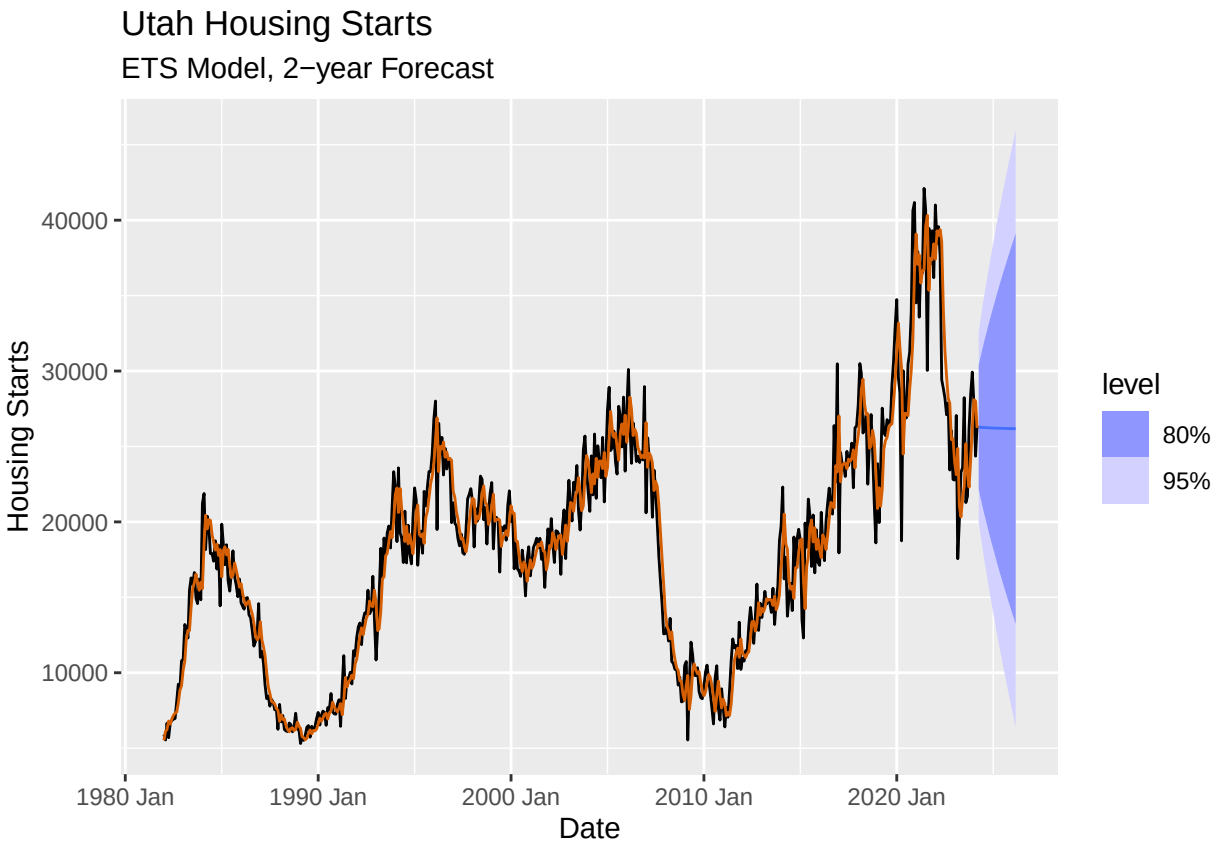


For the forecasting of Utah Housing Starts:

```
Housing_Starts_Model_ETS = ETS_forecaster(Monthly,
                                          "Housing_Starts_UT",
                                          24,
                                          "Utah Housing Starts",
                                          "ETS Model, 2-year Forecast",
                                          "Date",
                                          "Housing Starts")
```

```
## Series: Housing_Starts_UT
## Model: ETS(M,Ad,N)
##   Smoothing parameters:
##     alpha = 0.4887716
##     beta  = 0.01569634
##     phi   = 0.9369255
##
##   Initial states:
##      l[0]     b[0]
##  5073.656 472.6965
##
##   sigma^2:  0.0148
##
##      AIC     AICc      BIC
## 10872.46 10872.63 10897.83
```

```
Housing_Starts_Model_ETS$Plot
```



Autoregressive Integrated Moving Averages (ARIMA)

Autoregressive Integrated Moving Averages (ARIMA) is another univariate forecasting method that allow for data with auto-correlative properties (i.e. previous values have influence on current values) to be modeled more accurately. Despite the flexibility of ARIMA models, they may not necessarily provide better fits than ETS models.

The model can be easily created using the following function found in the script "03 - ARIMA Model".

```
ARIMA_forecaster = function(Dataset,Variable, Horizon, Title, Subtitle, xlab, ylab) {
  variable_sym = ensym(Variable)

  Set_Name = Dataset %>%
    select(DATE, !!variable_sym) %>%
    filter(!is.na(!!variable_sym))

  Model = Set_Name %>%
    model(ARIMA(!!variable_sym))

  Forecast = Model %>%
    forecast(h = Horizon)

  Model_Specs = Model %>%
    report()

  Plot = Forecast %>%
    autoplot(Set_Name) +
    geom_line(aes(y = .fitted), col = "#D55E00", data = augment(Model)) +
    ggtitle(Title, Subtitle) +
    xlab(xlab) + ylab(ylab)

  list(Model = Forecast, Plot = Plot)
}
```

The function's arguments are:

- ***Dataset*** **-** Selects the dataset to be used for the ARIMA analysis.
- ***Variable*** **-** Selects the variable to be forecasted by the ARIMA model.
- ***Horizon*** **-** Determines the number of steps in the future the data is to be forecasted. For example, to forecast monthly Housing Starts two years into the future, the horizon would be 24. Forecasts give more confident values if the horizon is smaller.
- ***Title*** and ***Subtitle*** **-** Gives a title and subtitle to the plot of the ARIMA model.
- ***xlab*** and ***ylab*** - Gives headers for the x and y axes to the plot of the ARIMA model.

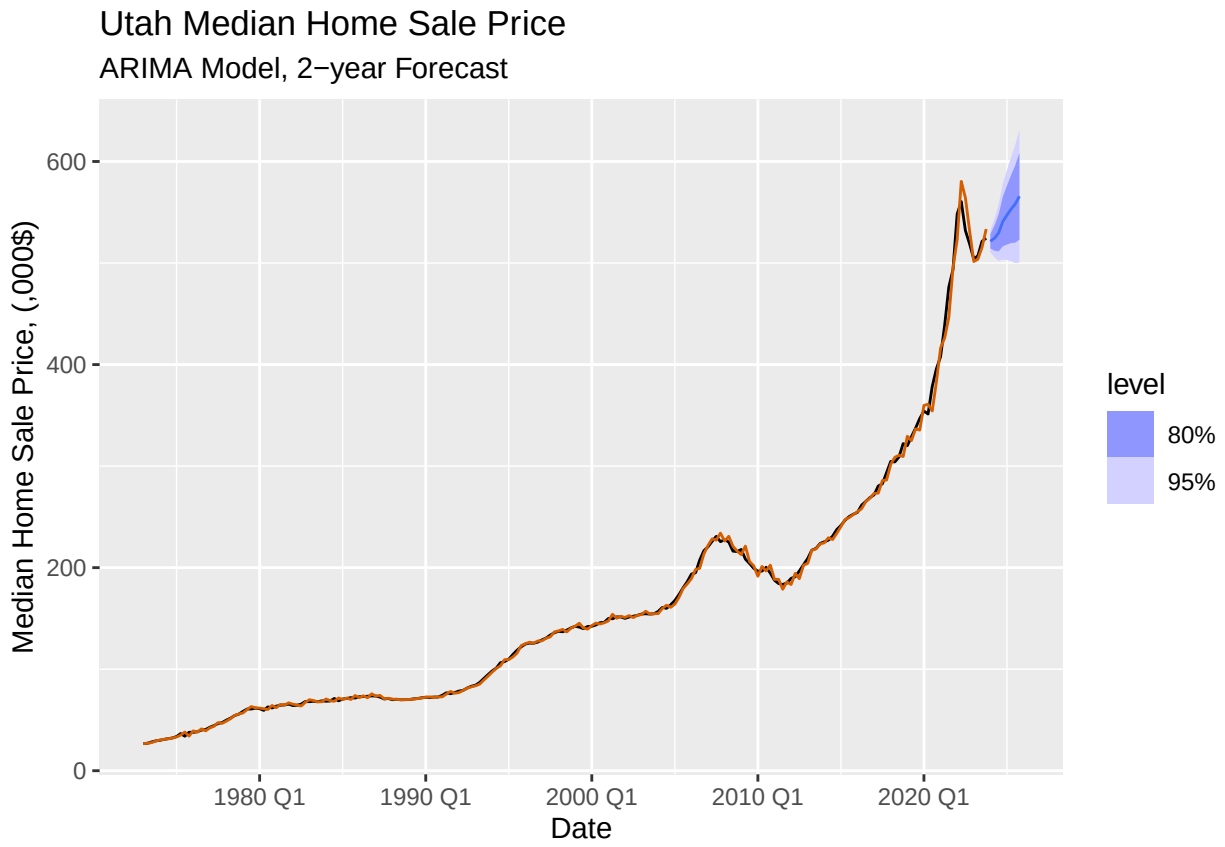
The function will automatically create the most well-fitting model as well as accompanying graphs. Fitted values from the model will be represented with the orange-colored series.

For the forecasting of Utah Median Home Sale Prices:

```
Median_Sale_Price_Model_ARIMA = ARIMA_forecaster(Quarterly,
                                                 "Median_Sale_Price_UT",
                                                 8,
                                                 "Utah Median Home Sale Price",
                                                 "ARIMA Model, 2-year Forecast",
                                                 "Date",
                                                 "Median Home Sale Price, (,000$)")
```

```
## Series: Median_Sale_Price_UT
## Model: ARIMA(0,2,1)(0,0,2)[4]
##
## Coefficients:
##           ma1     sma1     sma2
##       -0.5568  -0.5790  -0.2118
## s.e.   0.0641   0.0686   0.0655
##
## sigma^2 estimated as 30.23:  log likelihood=-631.38
## AIC=1270.75   AICc=1270.96   BIC=1283.99
```

```
Median_Sale_Price_Model_ARIMA$Plot
```



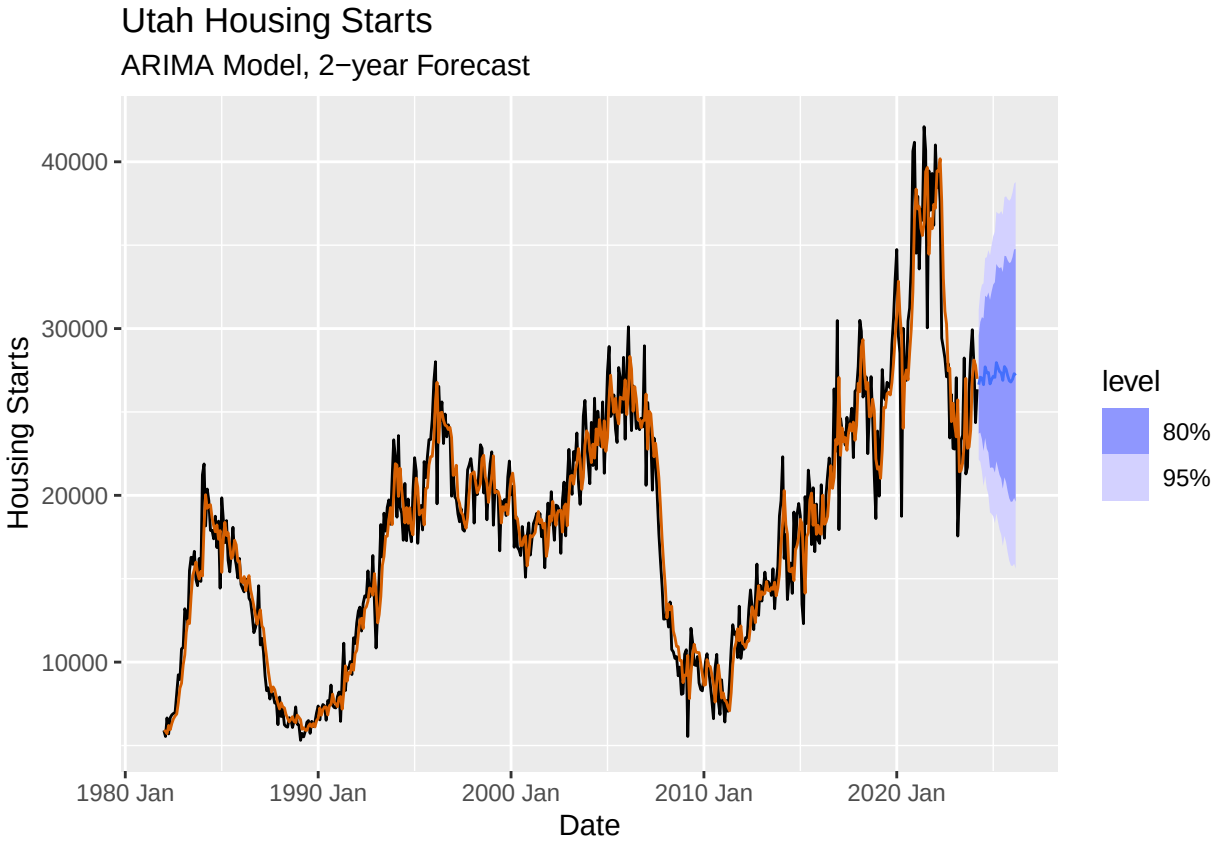
For the forecasting of Utah Housing Starts:

```
Housing_Starts_Model_ARIMA = ARIMA_forecaster(Monthly,
                                              "Housing_Starts_UT",
                                              24,
                                              "Utah Housing Starts",
                                              "ARIMA Model, 2-year Forecast",
                                              "Date",
                                              "Housing Starts")
```

```
## Series: Housing_Starts_UT
## Model: ARIMA(0,1,1)(0,0,2)[12]
```

```
##
## Coefficients:
##           ma1     sma1     sma2
##       -0.4727  -0.1082  -0.1009
## s.e.   0.0380   0.0448   0.0419
##
## sigma^2 estimated as 5243281:  log likelihood=-4631.35
## AIC=9270.69   AICc=9270.77   BIC=9287.6
```

```
Housing_Starts_Model_ARIMA$Plot
```



OLS Regression

Ordinary Least Squares Regression (OLS) is method to detect the relationship between explanatory and response variables. However, the OLS model comes with a large amount of caveats and assumptions about the data. Due to the limitations of our data, the OLS model is not recommended as a forecasting method, but it may present interesting insights and correlative relationships within the data.

Analysis of the data can be found within the script "06 - Least Squares and Dynamic Regression Models".

Fitted values from the model will be represented with the orange-colored series.

For the forecasting of the variable Utah Median Home Sale Price, the following model was selected to be the best fit. Forecasting is based on a created hypothetical future.

```
Quarterly_Regression = Forecast_Set = new_data(Quarterly_NoNA,8) %>%
  mutate(Consumer_Conf = mean(Quarterly_NoNA$Consumer_Conf),
         Treasury_Yield = mean(Quarterly_NoNA$Treasury_Yield),
         Total_Assets = mean(Quarterly_NoNA$Total_Assets),
         Household_net_worth = mean(Quarterly_NoNA$Household_net_worth),
         PPI = mean(Quarterly_NoNA$PPI),
         Revolving_Home_Loans = mean(Quarterly_NoNA$Revolving_Home_Loans),
         CER_UT_Avg = mean(Quarterly_NoNA$CER_UT_Avg),
         Affordability_Index_UT_Avg = mean(Quarterly_NoNA$Affordability_Index_UT_Avg),
         HMI = mean(Quarterly_NoNA$HMI),
         Net_Migration_UT = mean(Quarterly_NoNA$Net_Migration_UT),
         Rental_Vacancy_UT = mean(Quarterly_NoNA$Rental_Vacancy_UT),
         Foreclosed_UT = mean(Quarterly_NoNA$Foreclosed_UT),
         Median_Sale_Price_UT = mean(Quarterly_NoNA$Median_Sale_Price_UT))
```

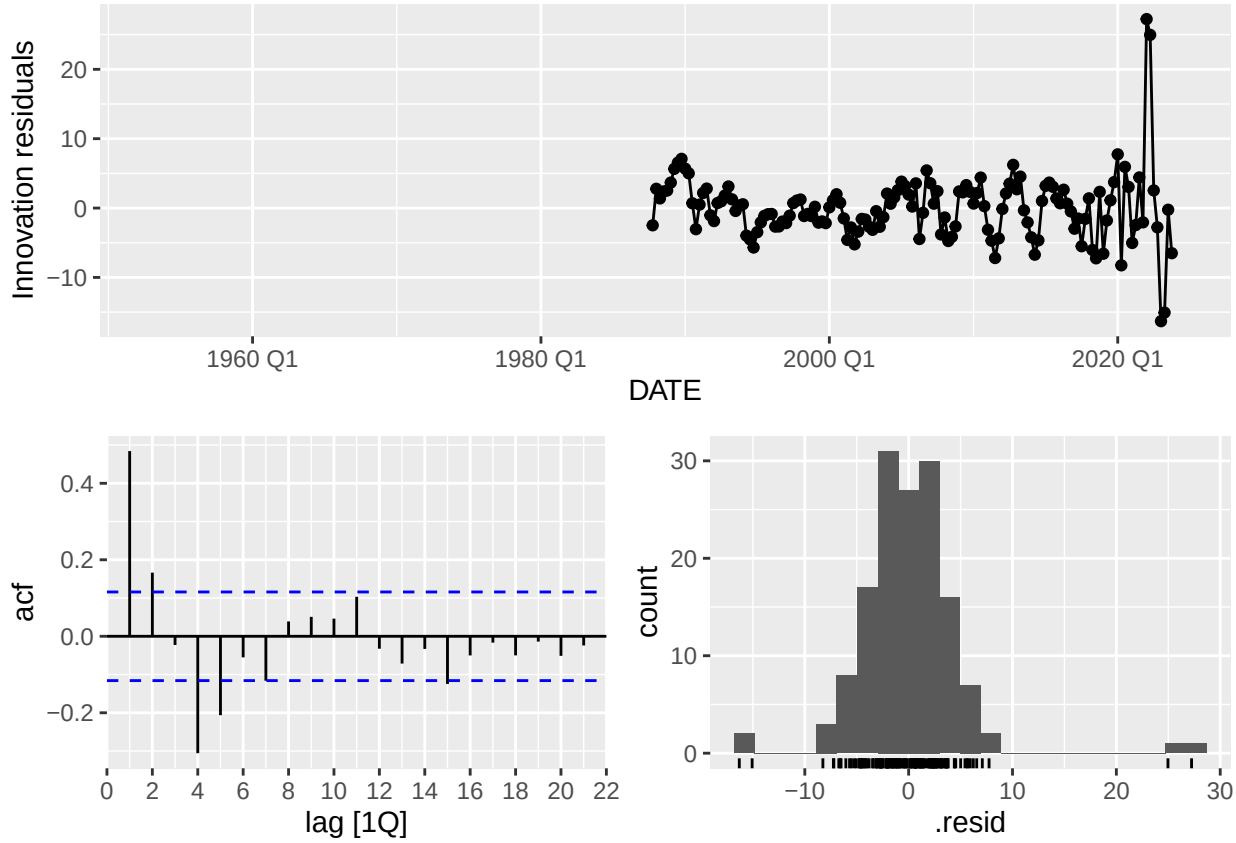
```
Best_Fit_Median_Sale_Price = Quarterly %>% model("Best" =
                                                  TSLM(Median_Sale_Price_UT ~
                                                         Consumer_Conf +
                                                         Treasury_Yield +
                                                         Total_Assets +
                                                         Household_net_worth +
                                                         PPI +
                                                         Revolving_Home_Loans +
                                                         CER_UT_Avg +
                                                         Affordability_Index_UT_Avg
                                                       + HMI))

Best_Fit_Median_Sale_Price %>% report()
```

```
## Series: Median_Sale_Price_UT
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.3076  -2.6524   0.1845   2.3785  27.2417
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  1.155e+02  2.244e+01   5.149 9.06e-07 ***
## Consumer_Conf               -3.452e-01  5.780e-02  -5.973 1.95e-08 ***
## Treasury_Yield               3.094e-03  1.183e-03   2.615 0.009926 **
## Total_Assets                 3.150e-03  7.437e-04   4.236 4.18e-05 ***
## Household_net_worth          5.227e-07  1.437e-07   3.638 0.000391 ***
## PPI                          1.390e-01  5.866e-02   2.370 0.019212 *
## Revolving_Home_Loans        -6.676e-02  5.216e-03 -12.799  < 2e-16 ***
## CER_UT_Avg                  -7.254e+00  1.301e+00  -5.578 1.28e-07 ***
## Affordability_Index_UT_Avg -3.805e-01  8.429e-02  -4.514 1.37e-05 ***
## HMI                          4.134e-01  3.376e-02  12.247  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.026 on 135 degrees of freedom
## Multiple R-squared: 0.9983,  Adjusted R-squared: 0.9982
```

```
## F-statistic:  8730 on 9 and 135 DF, p-value: < 2.22e-16
```

```
Best_Fit_Median_Sale_Price %>% gg_tsresiduals()
```

```
## Warning: Removed 141 rows containing missing values or values outside the scale range
## (`geom_line()`).

## Warning: Removed 141 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 141 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

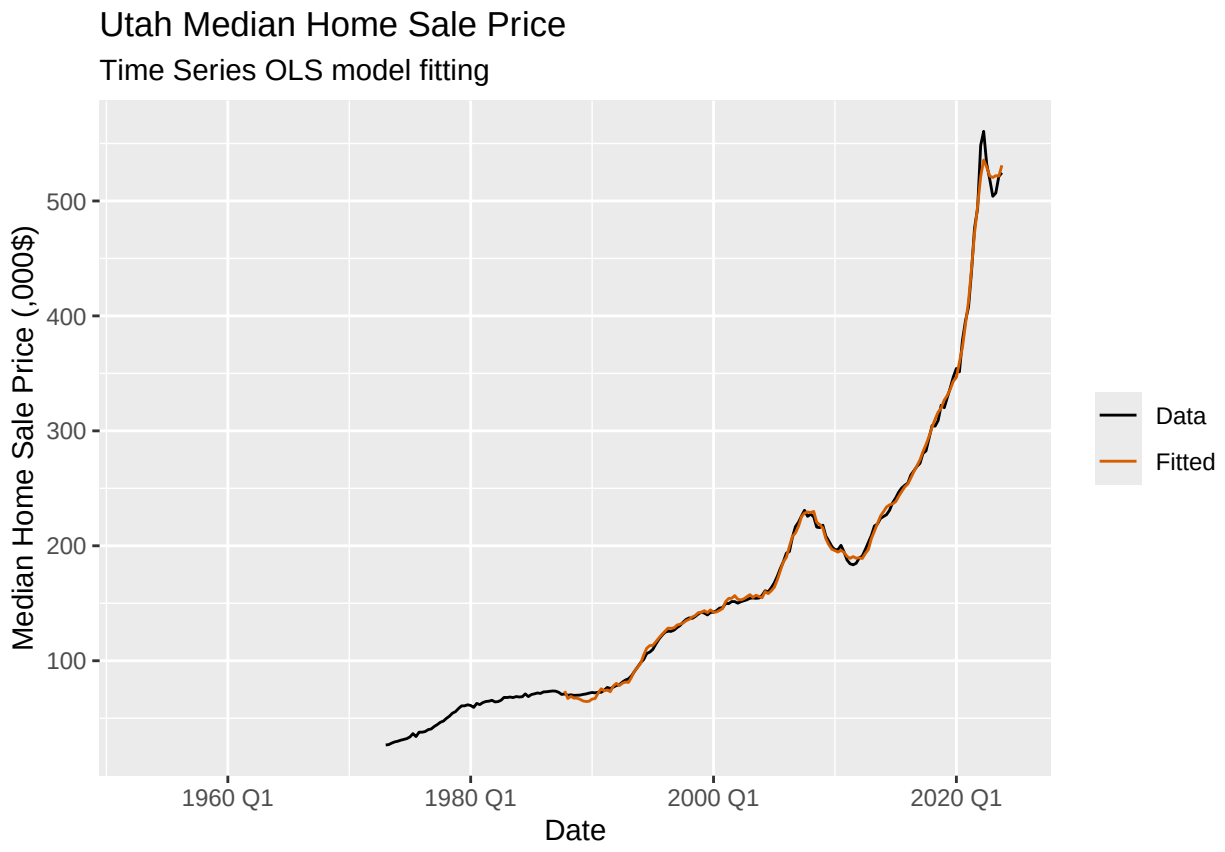


Despite the unusually high adjusted R-squared statistic, looking at the residuals of this model indicate that there is still some considerable autocorrelation issues, indicating that the fit of the model is not optimal. The plot reveals a rather awkward forecast. As such, it is not recommended that the OLS model is implemented to forecast Utah Median Home Sale Prices.

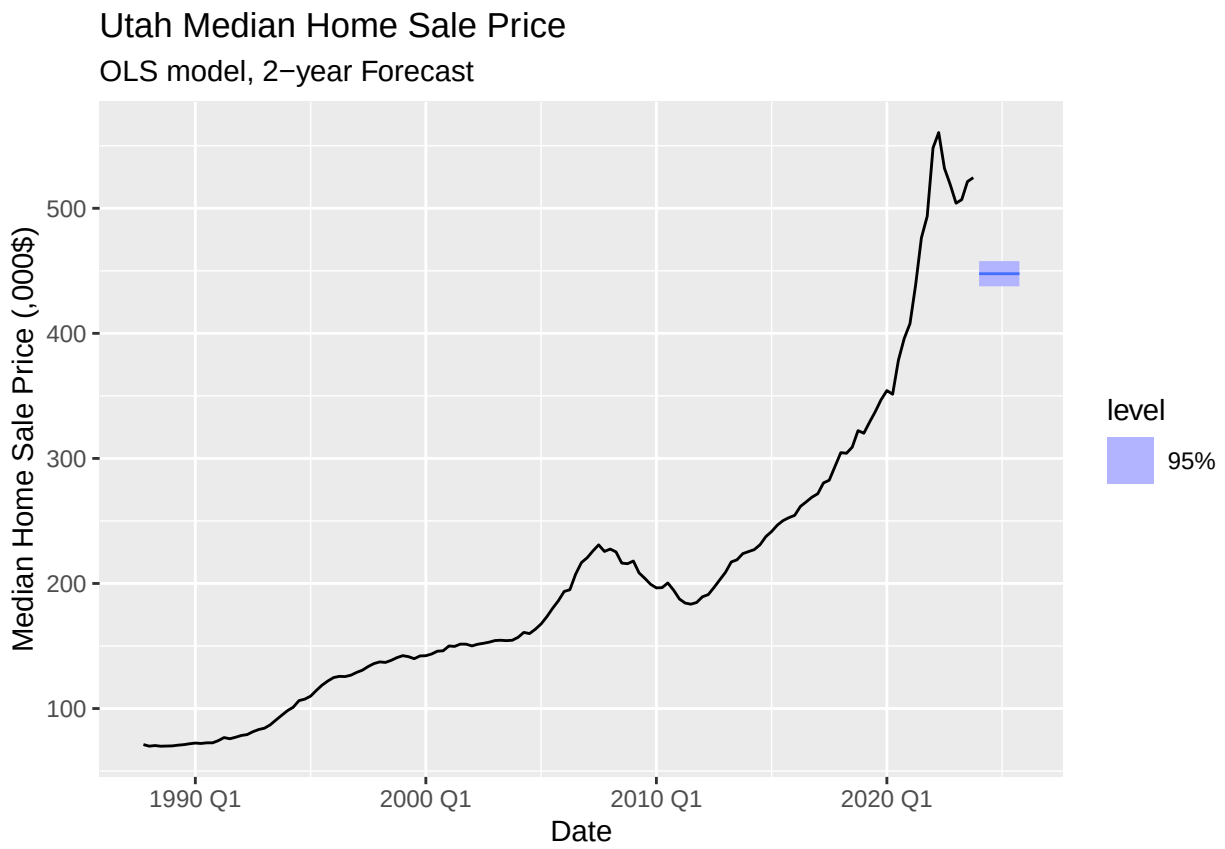
```
augment(Best_Fit_Median_Sale_Price) %>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = Median_Sale_Price_UT,colour = "Data")) +
  geom_line(aes(y = .fitted,colour = "Fitted")) +
  ggtitle("Utah Median Home Sale Price", "Time Series OLS model fitting") +
  xlab("Date") + ylab("Median Home Sale Price (,000$)") +
  scale_colour_manual(values=c(Data="black",Fitted="#D55E00")) +
  guides(colour = guide_legend(title = NULL)) #Practically overlapping
```

```
## Warning: Removed 82 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 141 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```
Best_Fit_Median_Sale_Price %>%
  forecast(new_data = Quarterly_Regression) %>%
  autoplot(Quarterly_NoNA_NoSP,level = 95) +
  labs(title = "Utah Median Home Sale Price",subtitle = "OLS model, 2-year Forecast",
                                          x = "Date",
       y = "Median Home Sale Price (,000$)")
```



For the forecasting of the variable Utah Housing Starts, the following model was selected to be the best fit. Forecasting is based on a created hypothetical future.

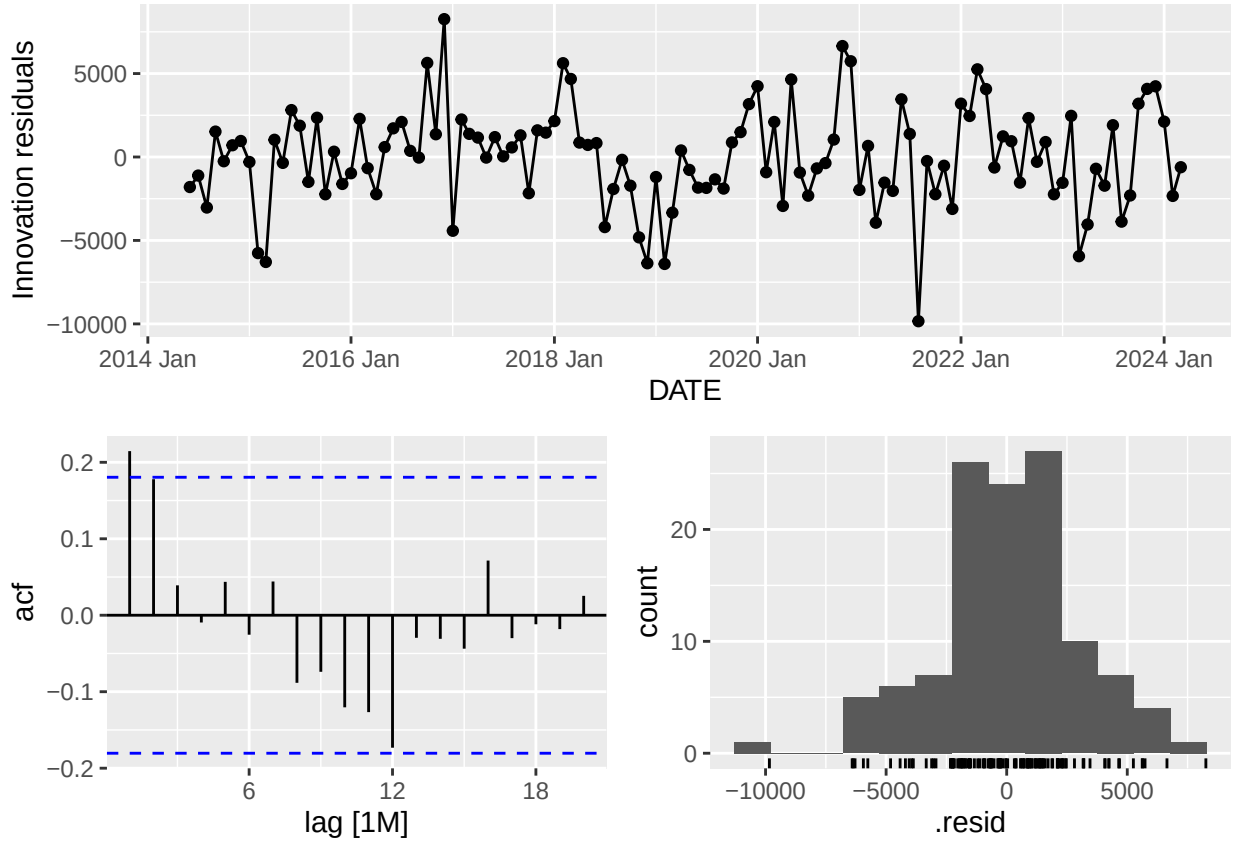
```
Monthly_Regression = new_data(Monthly_NoNA,8) %>%
  mutate(Consumer_Conf_Monthly = mean(Monthly_NoNA$Consumer_Conf_Monthly),
         Unemployment_Rate_Monthly = mean(Monthly_NoNA$Unemployment_Rate_Monthly),
         Housing_Permits_UT = mean(Monthly_NoNA$Housing_Permits_UT),
         PPI = mean(Monthly_NoNA$PPI),
         SP500_Monthly = mean(Monthly_NoNA$SP500_Monthly),
         Revolving_Home_Loan_Monthly = mean(Monthly_NoNA$Revolving_Home_Loan_Monthly),
         CPI = mean(Monthly_NoNA$CPI),
         Total_Assets_Monthly = mean(Monthly_NoNA$Total_Assets_Monthly))
```

```
Best_Fit_Housing_Starts = Monthly_NoNA %>%
  model("Best" = TSLM(Housing_Starts_UT ~ CPI +
                        Revolving_Home_Loan_Monthly +
                        SP500_Monthly +
                        Unemployment_Rate_Monthly))

Best_Fit_Housing_Starts %>% report()
```

```
## Series: Housing_Starts_UT
## Model: TSLM
##
## Residuals:
```

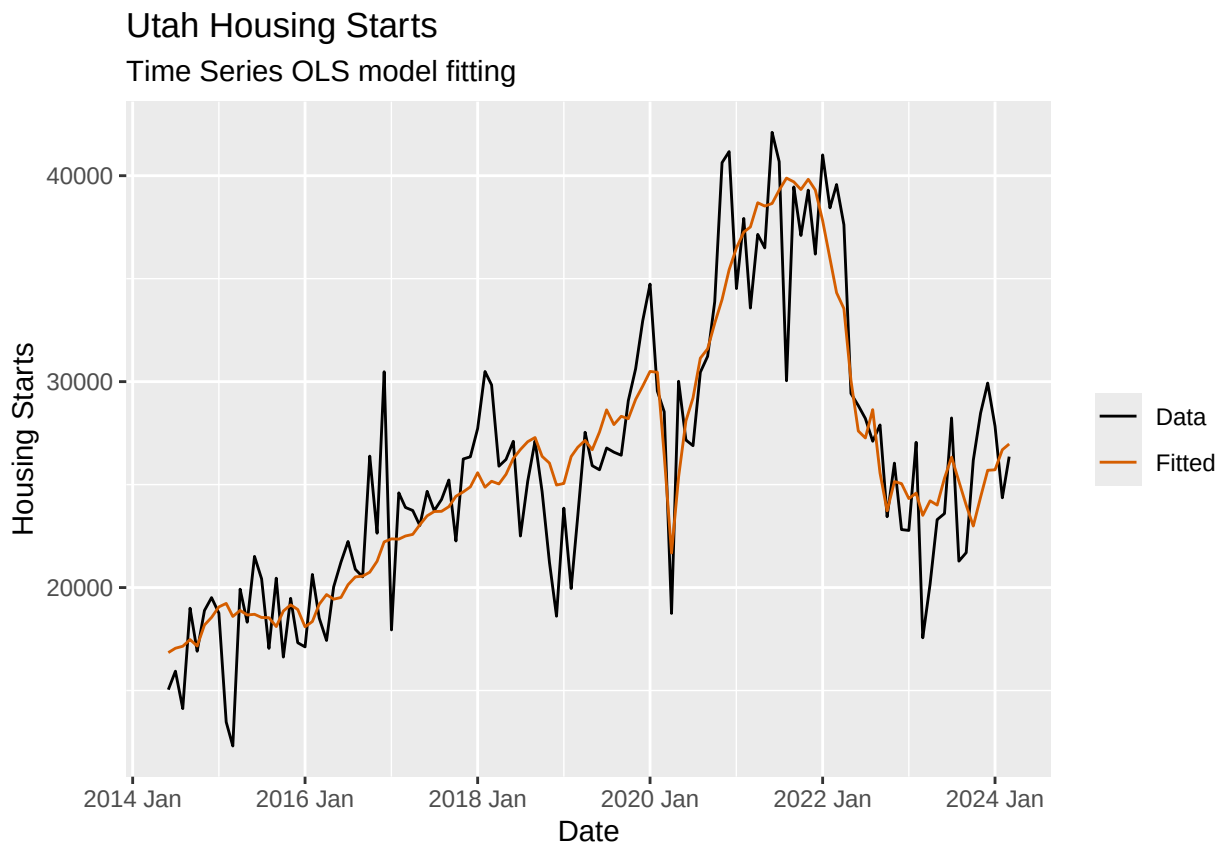
```
##      Min       1Q   Median       3Q      Max
## -9839.81 -1826.22   -36.04  1686.60  8260.25
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 162768.186  11103.675  14.659  < 2e-16 ***
## CPI                           -420.822     29.145 -14.439  < 2e-16 ***
## Revolving_Home_Loan_Monthly   -119.169     12.972  -9.187 2.33e-15 ***
## SP500_Monthly                    7.131      1.124   6.347 4.70e-09 ***
## Unemployment_Rate_Monthly     -836.034    302.040  -2.768  0.00659 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3001 on 113 degrees of freedom
## Multiple R-squared: 0.8139,  Adjusted R-squared: 0.8073
## F-statistic: 123.5 on 4 and 113 DF, p-value: < 2.22e-16
```

```
Best_Fit_Housing_Starts %>% gg_tsresiduals()
```

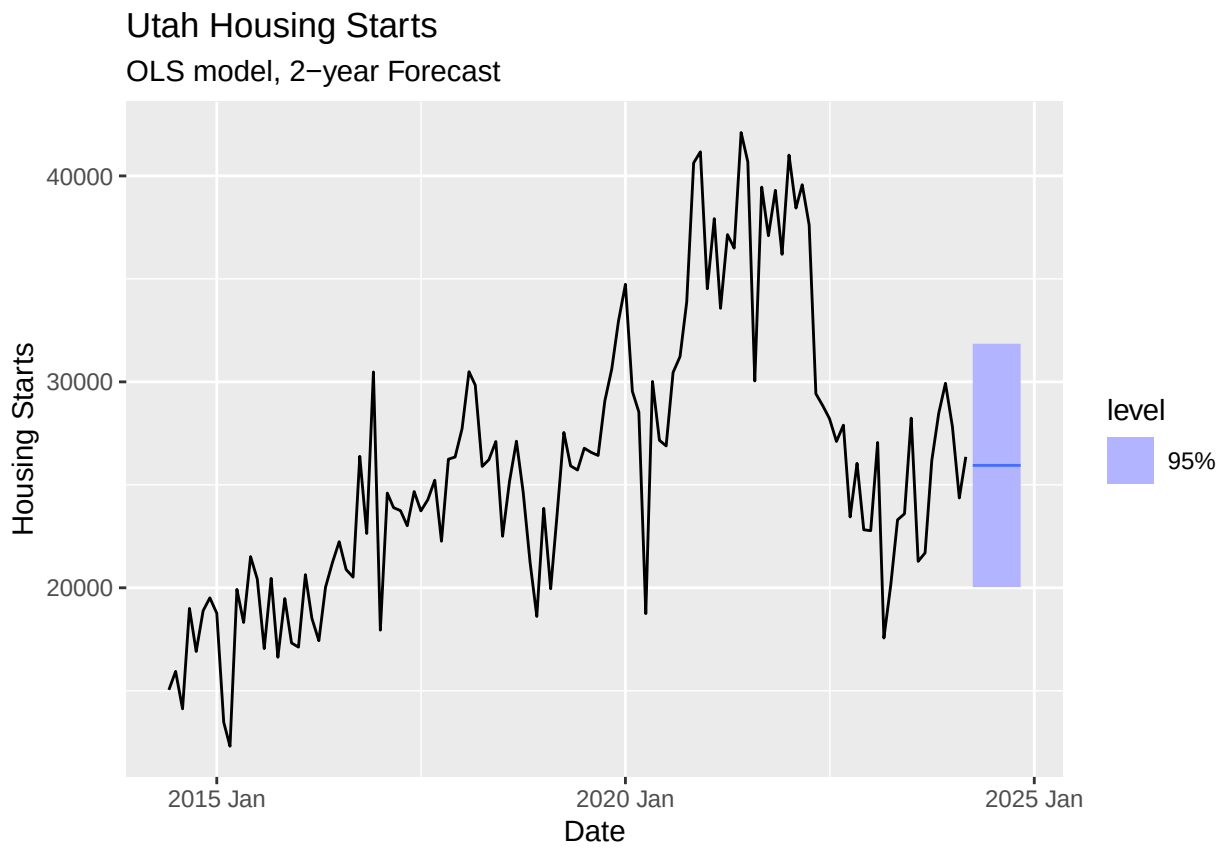


Despite the unusually high adjusted R-squared statistic, looking at the residuals of this model indicate that there is still some autocorrelation issues (though less considerable than the Utah Median Home Sale Price OLS model), indicating that the fit of the model is not optimal. The plot reveals a rather awkward forecast. As such, it is not recommended that the OLS model is implemented to forecast Utah Housing Starts.

```
augment(Best_Fit_Housing_Starts) %>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = Housing_Starts_UT,colour = "Data")) +
  geom_line(aes(y = .fitted,colour = "Fitted")) +
  ggtitle("Utah Housing Starts", "Time Series OLS model fitting") +
  xlab("Date") + ylab("Housing Starts") +
  scale_colour_manual(values=c(Data="black",Fitted="#D55E00")) +
  guides(colour = guide_legend(title = NULL))
```




```
Best_Fit_Housing_Starts %>%
  forecast(new_data = Monthly_Regression) %>%
  autoplot(Monthly_NoNA,level = 95) +
  labs(title = "Utah Housing Starts",subtitle = "OLS model, 2-year Forecast",
       x = "Date",y = "Housing Starts")
```



Dynamic ARIMA regression model

The OLS regression model can be implemented with ARIMA elements. In some instances, they can prove to be better fits for forecasting.

Analysis of the data can be found within the script "06 - Least Squares and Dynamic Regression Models".

Fitted values from the model will be represented with the orange-colored series.

For the forecasting of Utah Median Home Sale Prices, the following is the best fit:

```
ARIMA_Median_Sale_Price = Quarterly %>% model(ARIMA = ARIMA(Median_Sale_Price_UT ~
                                                            Consumer_Conf +
                                                            Treasury_Yield +
                                                            Total_Assets +
                                                            Household_net_worth +
                                                            PPI +
                                                            Revolving_Home_Loans +
                                                            CER_UT_Avg +
                                                          Affordability_Index_UT_Avg
                                                            + HMI))

ARIMA_Median_Sale_Price %>% report()
```

```
## Series: Median_Sale_Price_UT
## Model: LM w/ ARIMA(1,1,0)(1,0,0)[4] errors
```

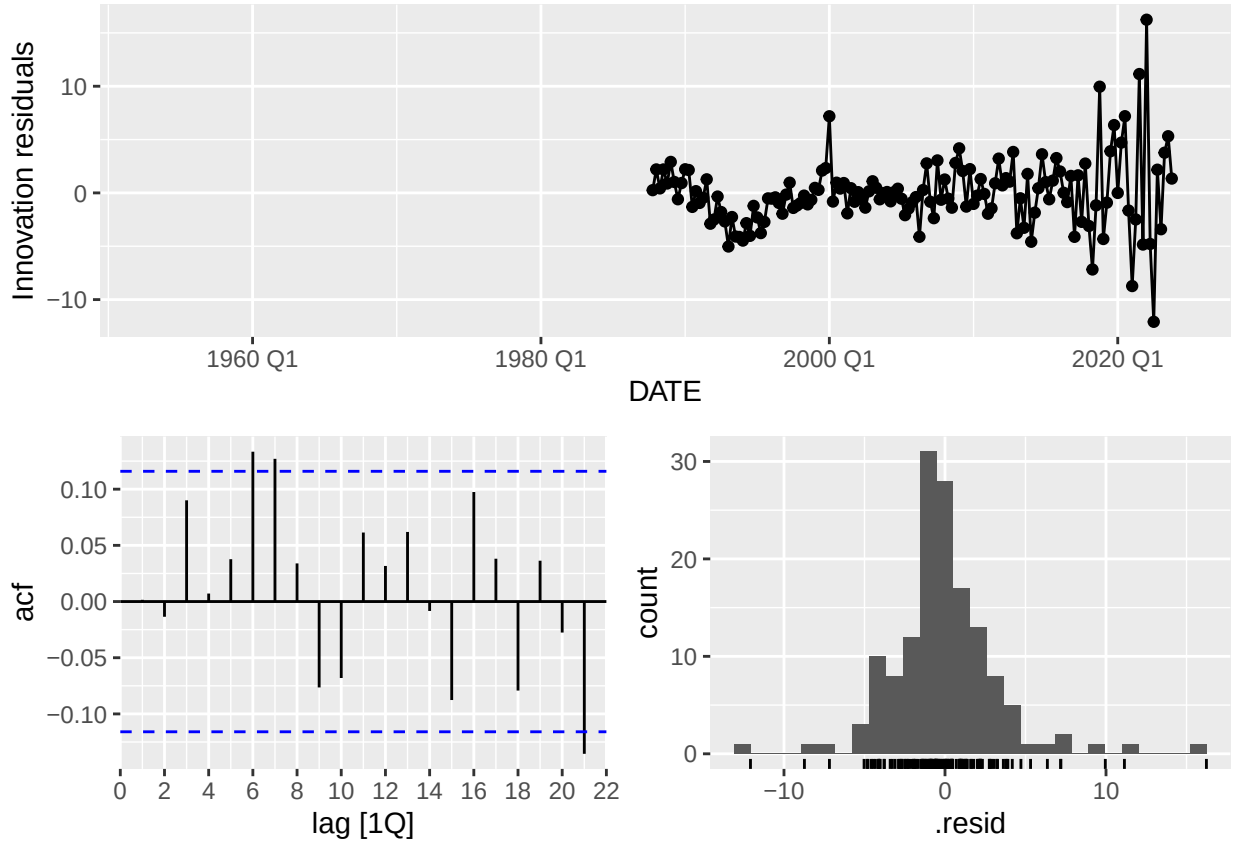
```
##
## Coefficients:
##          ar1     sar1  Consumer_Conf  Treasury_Yield  Total_Assets
##       0.1232  -0.4814        -0.0561         -0.0015       -0.0006
## s.e.  0.0931   0.0861         0.0545          0.0010        0.0013
##       Household_net_worth     PPI  Revolving_Home_Loans  CER_UT_Avg
##                         0  0.5694               -0.0486    -14.0911
## s.e.                    0  0.0798                0.0244      1.2373
##       Affordability_Index_UT_Avg     HMI
##                          -1.0262  0.3520
## s.e.                      0.0576  0.0381
##
## sigma^2 estimated as 5.722:  log likelihood=-376.76
## AIC=777.53   AICc=778.67   BIC=821.36
```

```
ARIMA_Median_Sale_Price %>% gg_tsresiduals()
```

```
## Warning: Removed 141 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 141 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

```
## Warning: Removed 141 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

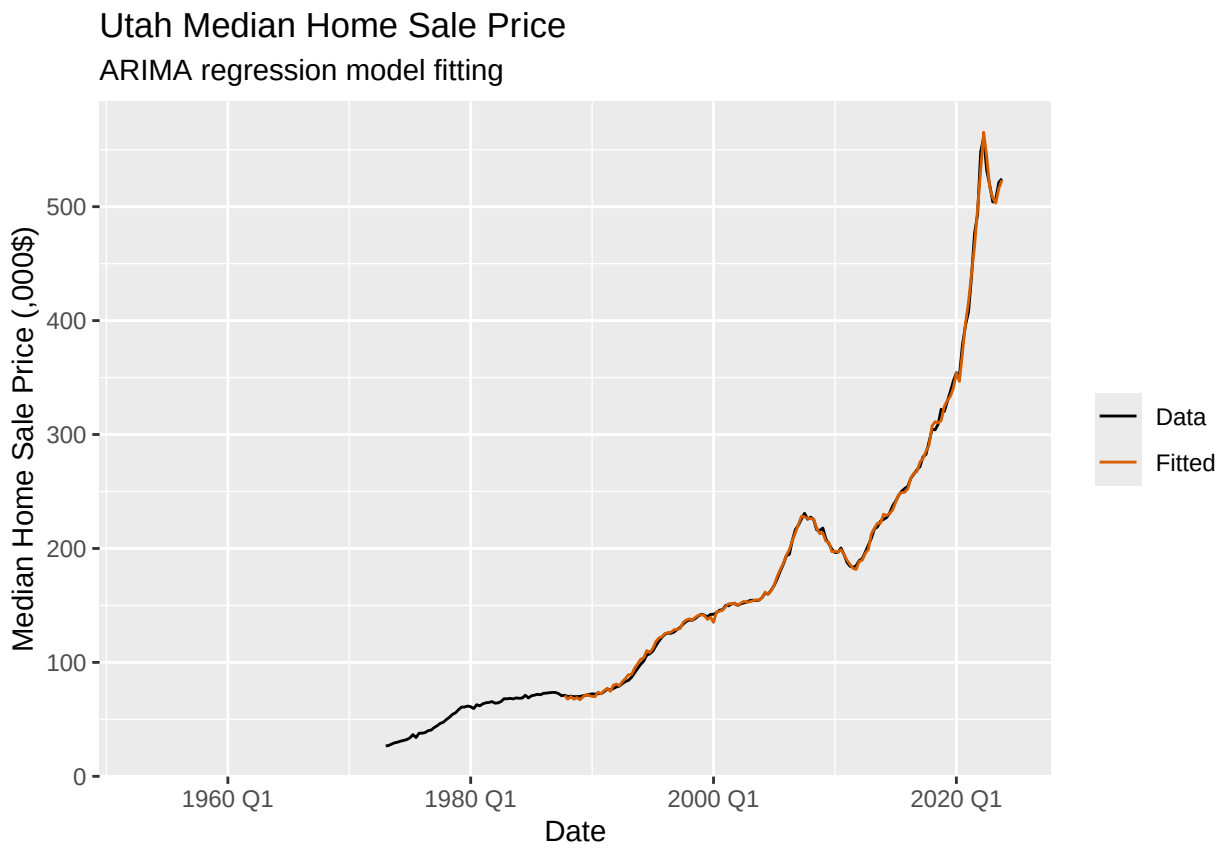


The data shows indications of some autocorrelation.

```
augment(ARIMA_Median_Sale_Price) %>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = Median_Sale_Price_UT,colour = "Data")) +
  geom_line(aes(y = .fitted,colour = "Fitted")) +
  ggtitle("Utah Median Home Sale Price", "ARIMA regression model fitting") +
  xlab("Date") + ylab("Median Home Sale Price (,000$)") +
  scale_colour_manual(values=c(Data="black",Fitted="#D55E00")) +
  guides(colour = guide_legend(title = NULL))
```

```
## Warning: Removed 82 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 141 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



Due to limitations, a forecasting graph for Housing Starts is unable to be provided.

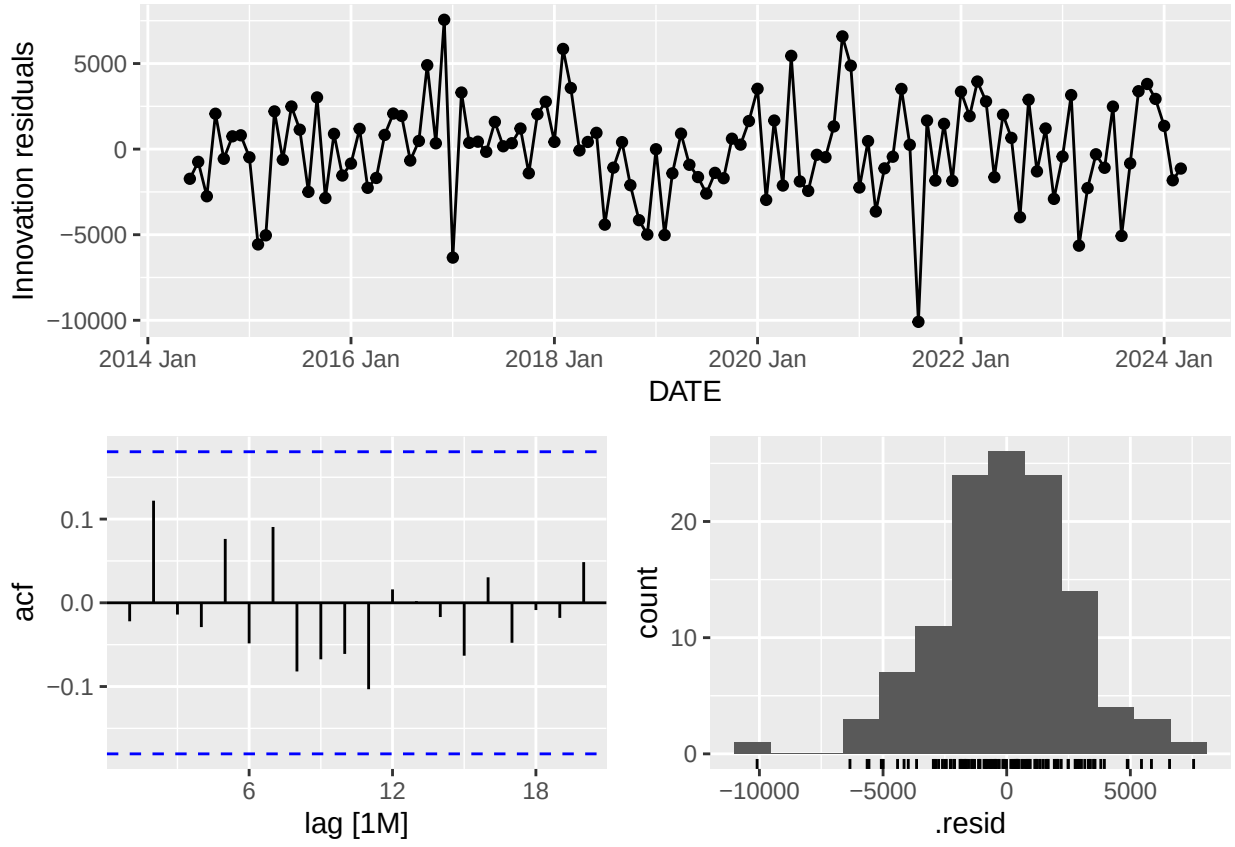
For Utah Housing Starts, the ARIMA model can prove to be a rather decent predictive model. For instance, there is no autocorrelation occurring and the plot seems realistic in terms of forecasted values.

```
ARIMA_Housing_Starts = Monthly_NoNA %>%
  model("ARIMA" = ARIMA(Housing_Starts_UT ~ CPI +
                          Revolving_Home_Loan_Monthly +
```

```
                          SP500_Monthly +
                          Unemployment_Rate_Monthly))
Results_ARIMA = ARIMA_Housing_Starts %>% report()
```

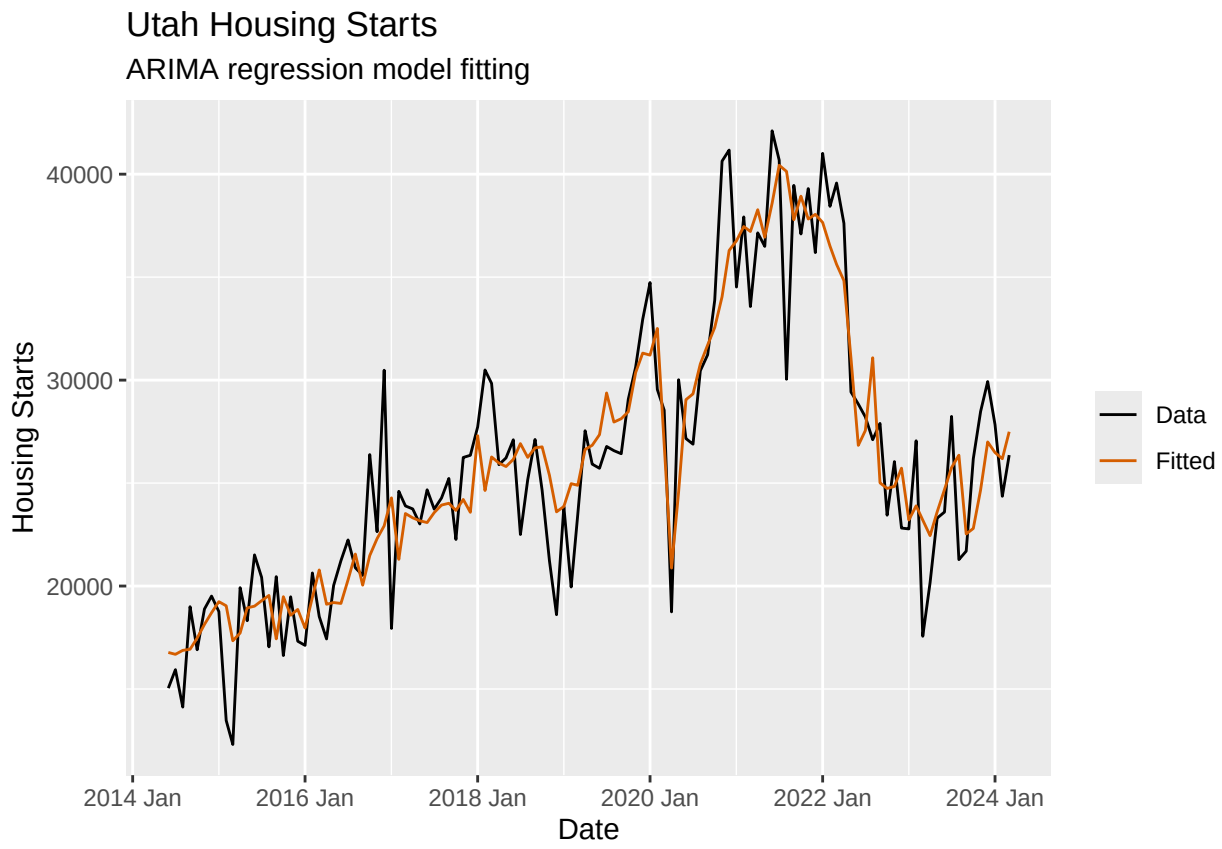
```
## Series: Housing_Starts_UT
## Model: LM w/ ARIMA(1,0,0)(0,0,1)[12] errors
##
## Coefficients:
##          ar1     sma1        CPI  Revolving_Home_Loan_Monthly  SP500_Monthly
##       0.1981  -0.2214  -419.5126                     -121.2332         6.8161
## s.e.  0.0925   0.1056    32.8303                       13.8476         1.3176
##       Unemployment_Rate_Monthly  intercept
##                       -1001.9349  164635.77
## s.e.                    326.9912   11650.61
##
## sigma^2 estimated as 8390692:  log likelihood=-1104.76
## AIC=2225.53   AICc=2226.85   BIC=2247.69
```

```
ARIMA_Housing_Starts %>% gg_tsresiduals()
```

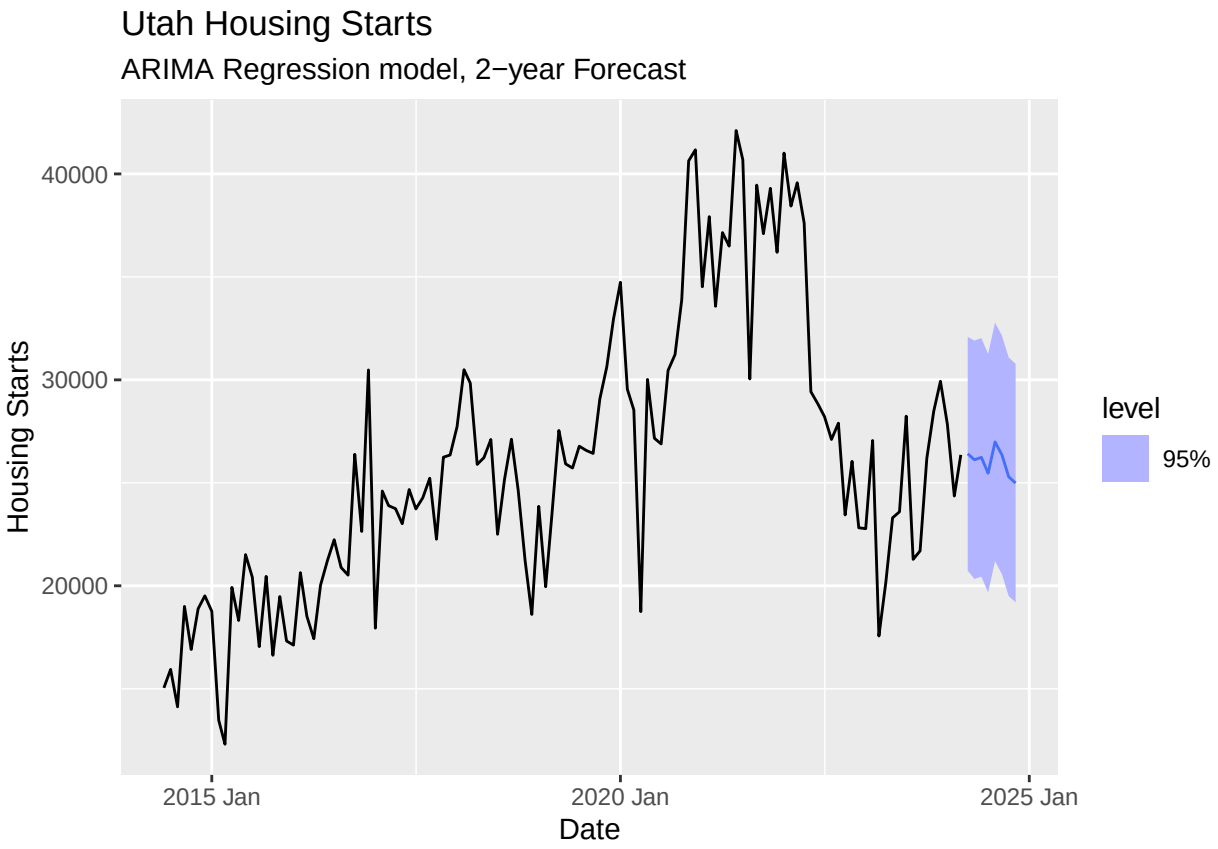



```
augment(ARIMA_Housing_Starts) %>%
  ggplot(aes(x = DATE)) +
  geom_line(aes(y = Housing_Starts_UT,colour = "Data")) +
  geom_line(aes(y = .fitted,colour = "Fitted")) +
```

```
  ggtitle("Utah Housing Starts", "ARIMA regression model fitting") +
  xlab("Date") + ylab("Housing Starts") +
  scale_colour_manual(values=c(Data="black",Fitted="#D55E00")) +
  guides(colour = guide_legend(title = NULL))
```




```
ARIMA_Housing_Starts %>%
  forecast(new_data = Monthly_Regression) %>%
  autoplot(Monthly_NoNA,level = 95) +
  labs(title = "Utah Housing Starts",
       subtitle = "ARIMA Regression model, 2-year Forecast",
       x = "Date",
       y = "Housing Starts")
```



Prophet Model

The prophet model is a univariate forecasting model created by Facebook. The prophet model is typically utilized for daily data with strong historical seasonality. As such, the prophet model may not be as effective as the ARIMA and ETS models for forecasting since our data is not as granular.

The model can be easily created using the following function found in the script "04 - Prophet Model".

```
Prophet_Forecaster = function(Dataset,Variable, Horizon, Title, Subtitle, xlab, ylab) {
  variable_sym = ensym(Variable)

  Set_Name = Dataset %>%
    select(DATE, !!variable_sym) %>%
    filter(!is.na(!!variable_sym))

  Model = Set_Name %>%
    model(prophet(!!variable_sym))

  Forecast = Model %>%
    forecast(h = Horizon)

  Plot = Forecast %>%
    autoplot(Set_Name) +
    geom_line(aes(y = .fitted), col = "#D55E00", data = augment(Model)) +
    ggtitle(Title, Subtitle) +
```

```
    xlab(xlab) + ylab(ylab)

  list(Model = Forecast,Plot = Plot)
}
```

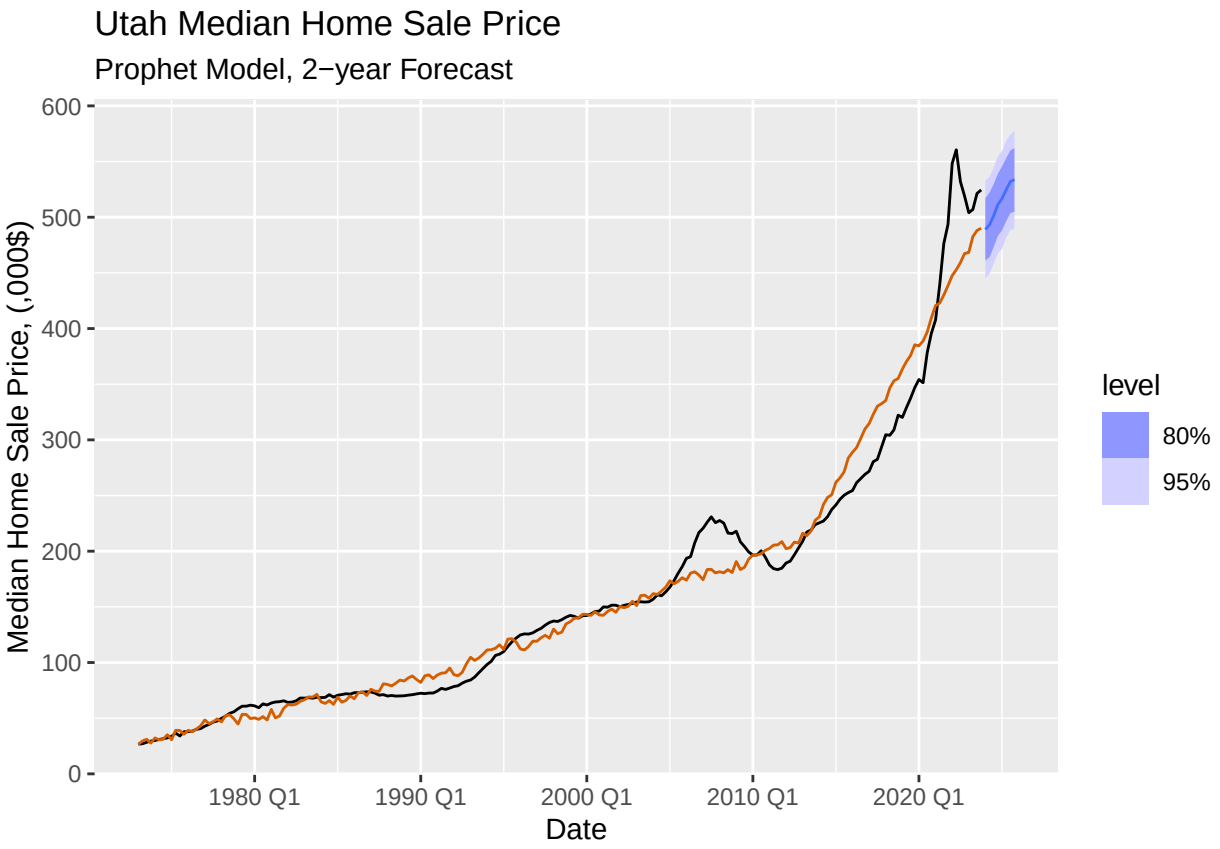
The function's arguments are:

- ***Dataset*** **-** Selects the dataset to be used for the prophet analysis.
- ***Variable*** **-** Selects the variable to be forecasted by the prophet model.
- ***Horizon*** **-** Determines the number of steps in the future the data is to be forecasted. For example, to forecast monthly Housing Starts two years into the future, the horizon would be 24. Forecasts give more confident values if the horizon is smaller.
- ***Title*** and ***Subtitle*** **-** Gives a title and subtitle to the plot of the prophet model.
- ***xlab*** and ***ylab*** - Gives headers for the x and y axes to the plot of the prophet model.

The function will automatically create the most well-fitting model as well as accompanying graphs. Fitted values from the model will be represented with the orange-colored series.

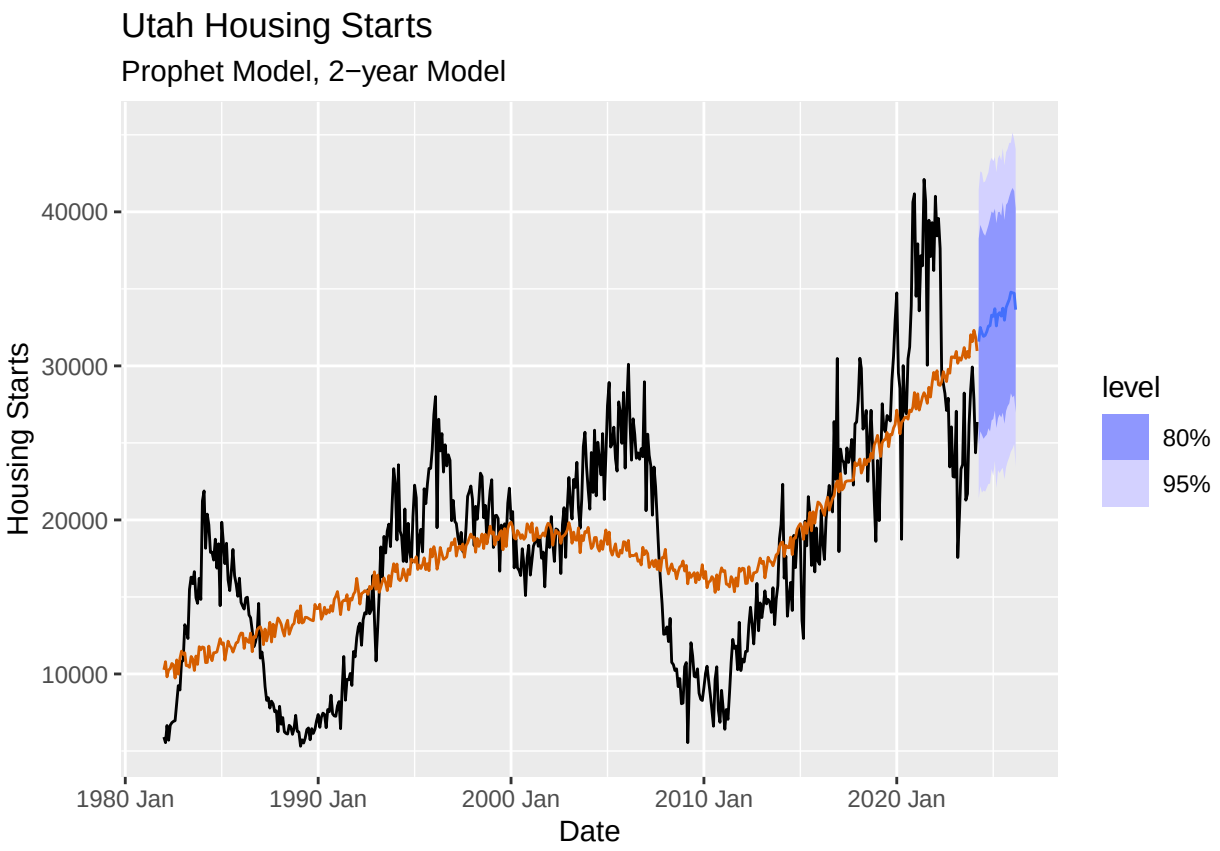
For the forecasting of Utah Median Home Sale Prices:

```
Median_Sale_Price_Model_Prophet = Prophet_Forecaster(Quarterly,
                                                     "Median_Sale_Price_UT",
                                                     8,
                                                     "Utah Median Home Sale Price",
                                                     "Prophet Model, 2-year Forecast",
                                                     "Date",
                                                     "Median Home Sale Price, (,000$)")

Median_Sale_Price_Model_Prophet$Plot
```



For the forecasting of Utah Housing Starts:

```
Housing_Starts_Model_Prophet = Prophet_Forecaster(Monthly,
                                                  "Housing_Starts_UT",
                                                  24,
                                                  "Utah Housing Starts",
                                                  "Prophet Model, 2-year Model",
                                                  "Date",
                                                  "Housing Starts")

Housing_Starts_Model_Prophet$Plot
```



Neural-ARIMA model

Elements of the ARIMA model can be implemented with Neural Networks. The Neural ARIMA model can be used for both univariate and regression purposes. However, Neural networks can have an overfitting effect, indicating that while the model can fit well with current and previous data, its *predictive ability* may not be as high as a more simple model (such as ETS or ARIMA).

The model can be easily created using the following function found in the script "05 - Neural ARIMA Model".

```
Neural_Forecaster = function(Dataset,Variable, Horizon, Title, Subtitle, xlab, ylab) {
  variable_sym = ensym(Variable)

  Set_Name = Dataset %>%
    select(DATE, !!variable_sym) %>%
    filter(!is.na(!!variable_sym))

  Model = Set_Name %>%
    model(NNETAR(!!variable_sym))

  Model_Specs = Model %>% report()

  Forecast = Model %>%
    forecast(h = Horizon)

  Plot = Forecast %>%
```

```
    autoplot(Set_Name) +
    geom_line(aes(y = .fitted), col = "#D55E00", data = augment(Model)) +
    ggtitle(Title, Subtitle) +
    xlab(xlab) + ylab(ylab)

  list(Model = Forecast,Plot = Plot,Specs = Model_Specs)
}
```

The function's arguments are:

- ***Dataset*** **-** Selects the dataset to be used for the Neural analysis.
- ***Variable*** **-** Selects the variable to be forecasted by the Neural model.
- ***Horizon*** **-** Determines the number of steps in the future the data is to be forecasted. For example, to forecast monthly Housing Starts two years into the future, the horizon would be 24. Forecasts give more confident values if the horizon is smaller.
- ***Title*** and ***Subtitle*** **-** Gives a title and subtitle to the plot of the Neural model.
- ***xlab*** and ***ylab*** - Gives headers for the x and y axes to the plot of the Neural model.

The function will automatically create the most well-fitting model as well as accompanying graphs. Fitted values from the model will be represented with the orange-colored series.

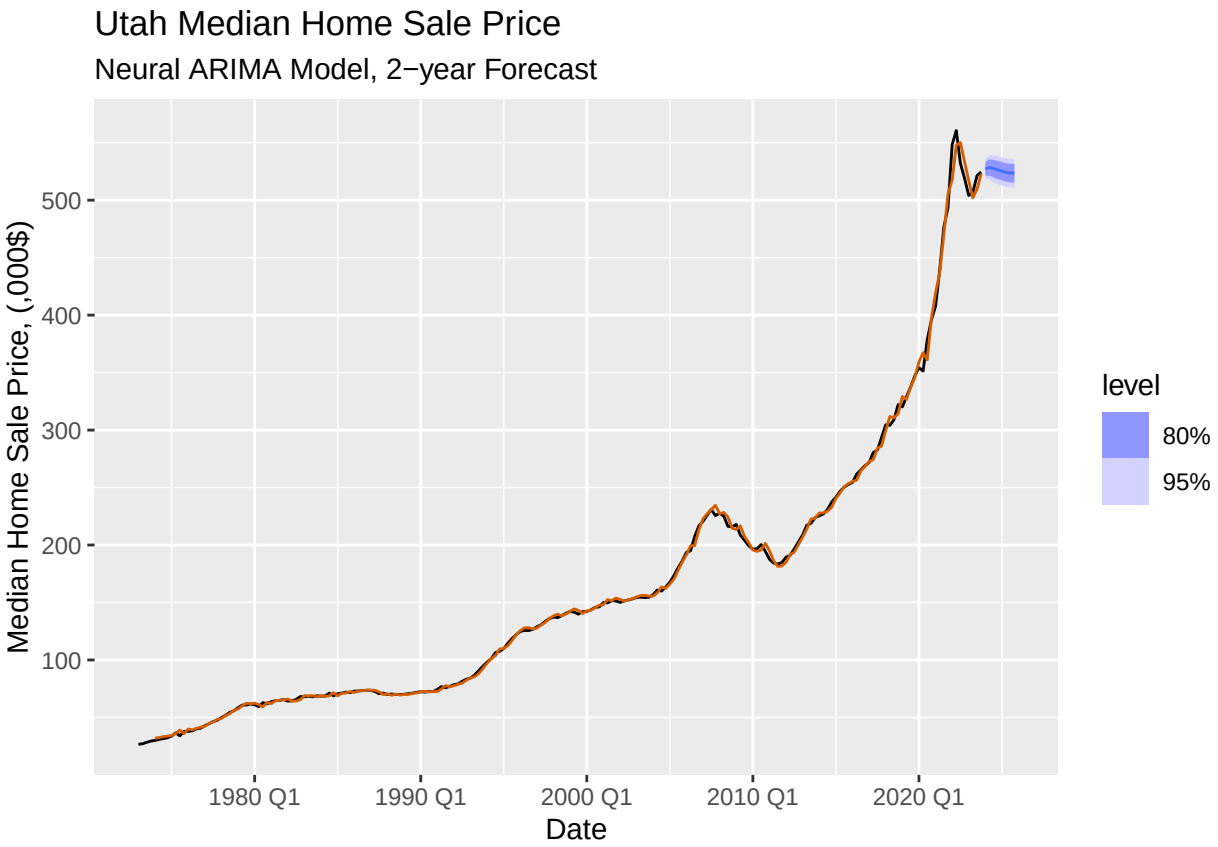
For the forecasting of Utah Median Home Sale Prices:

```
Median_Sale_Price_Model_Neural = Neural_Forecaster(Quarterly,
                                                   "Median_Sale_Price_UT",
                                                   8,
                                                   "Utah Median Home Sale Price",
                                                   "Neural ARIMA Model, 2-year Forecast",
                                                   "Date",
                                                   "Median Home Sale Price, (,000$)")
```

```
## Series: Median_Sale_Price_UT
## Model: NNAR(1,1,2)[4]
##
## Average of 20 networks, each of which is
## a 2-2-1 network with 9 weights
## options were - linear output units
##
## sigma^2 estimated as 20.68
```

```
Median_Sale_Price_Model_Neural$Plot
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



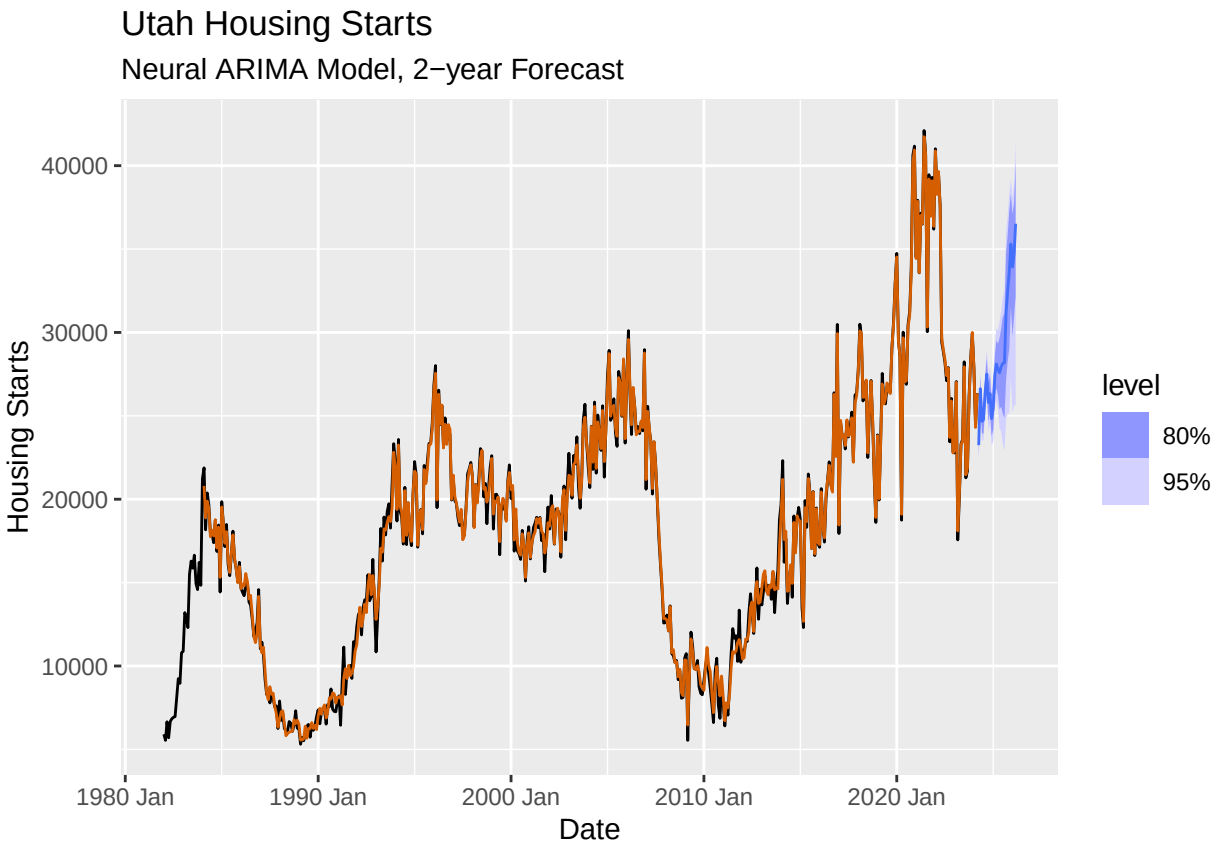
For the forecasting of Utah Housing Starts:

```
Housing_Starts__Model_Neural = Neural_Forecaster(Monthly,
                                                 "Housing_Starts_UT",
                                                 24,
                                                 "Utah Housing Starts",
                                                 "Neural ARIMA Model, 2-year Forecast",
                                                 "Date",
                                                 "Housing Starts")
```

```
## Series: Housing_Starts_UT
## Model: NNAR(25,1,13)[12]
##
## Average of 20 networks, each of which is
## a 25-13-1 network with 352 weights
## options were - linear output units
##
## sigma^2 estimated as 260120
```

```
Housing_Starts__Model_Neural$Plot
```

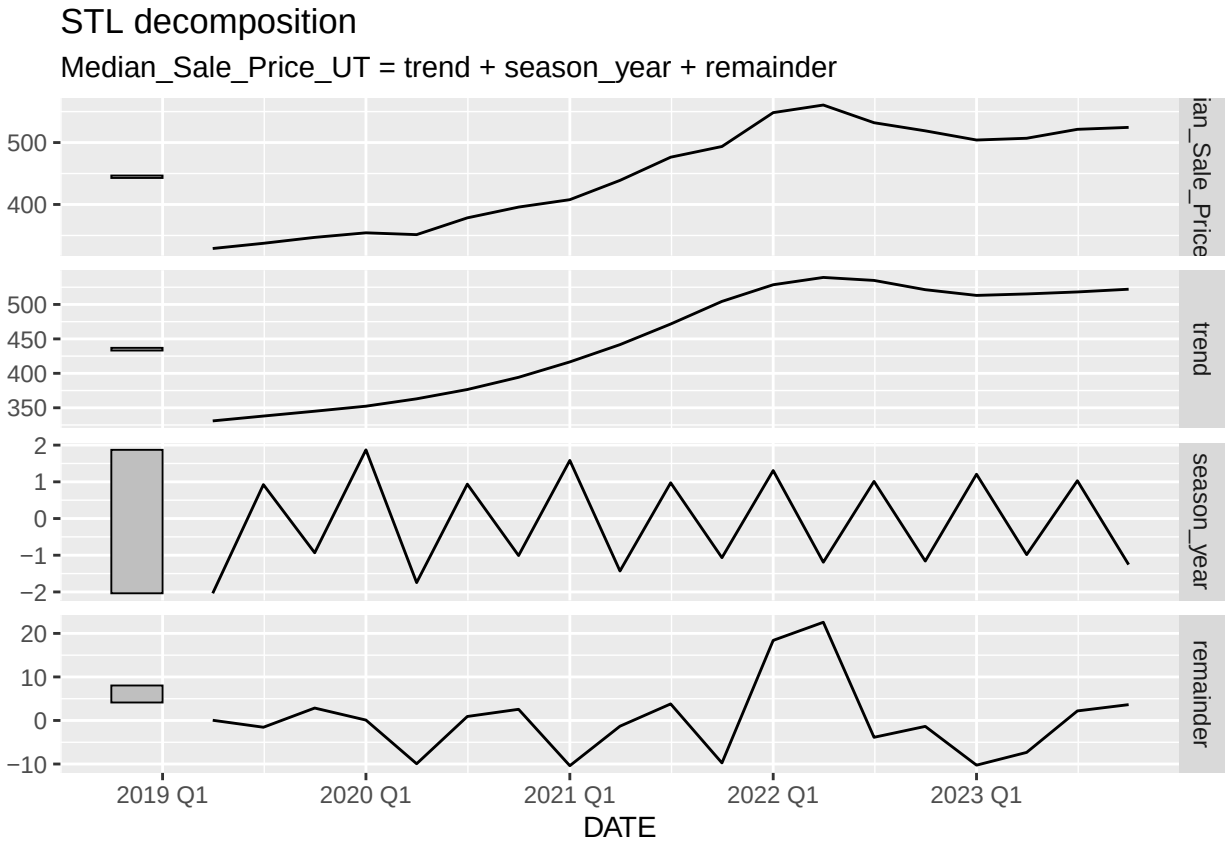
```
## Warning: Removed 25 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



Bootstrapping and Bagging

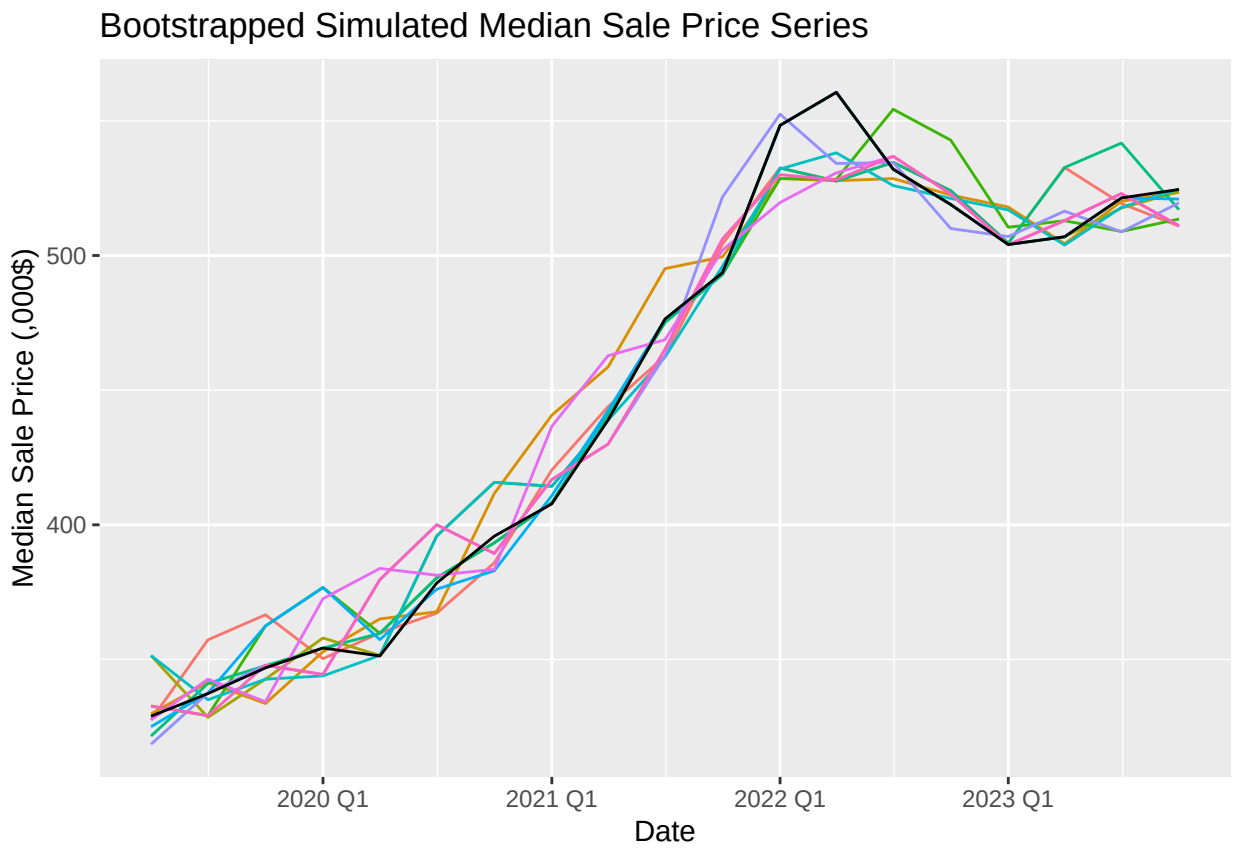
Bootstrapping and Bagging are methods to simulate and create similar series to the original. In this case, simulated series of Utah Median Home Sale Prices and Housing Starts. With many different series simulated, more models are able to be created. These models can be averaged out or "bagged", which can allow for more accurate forecasting as there are more scenarios implemented in the model analysis. Simulation begins with breaking down the series by its seasonal and trend components.

```
Median_Sale = Quarterly_NoNA %>% select(DATE,Median_Sale_Price_UT)

Median_Sale_Price_STL = Median_Sale %>%
  model(stl = STL(Median_Sale_Price_UT))

Median_Sale_Price_STL %>% components() %>% autoplot()
```

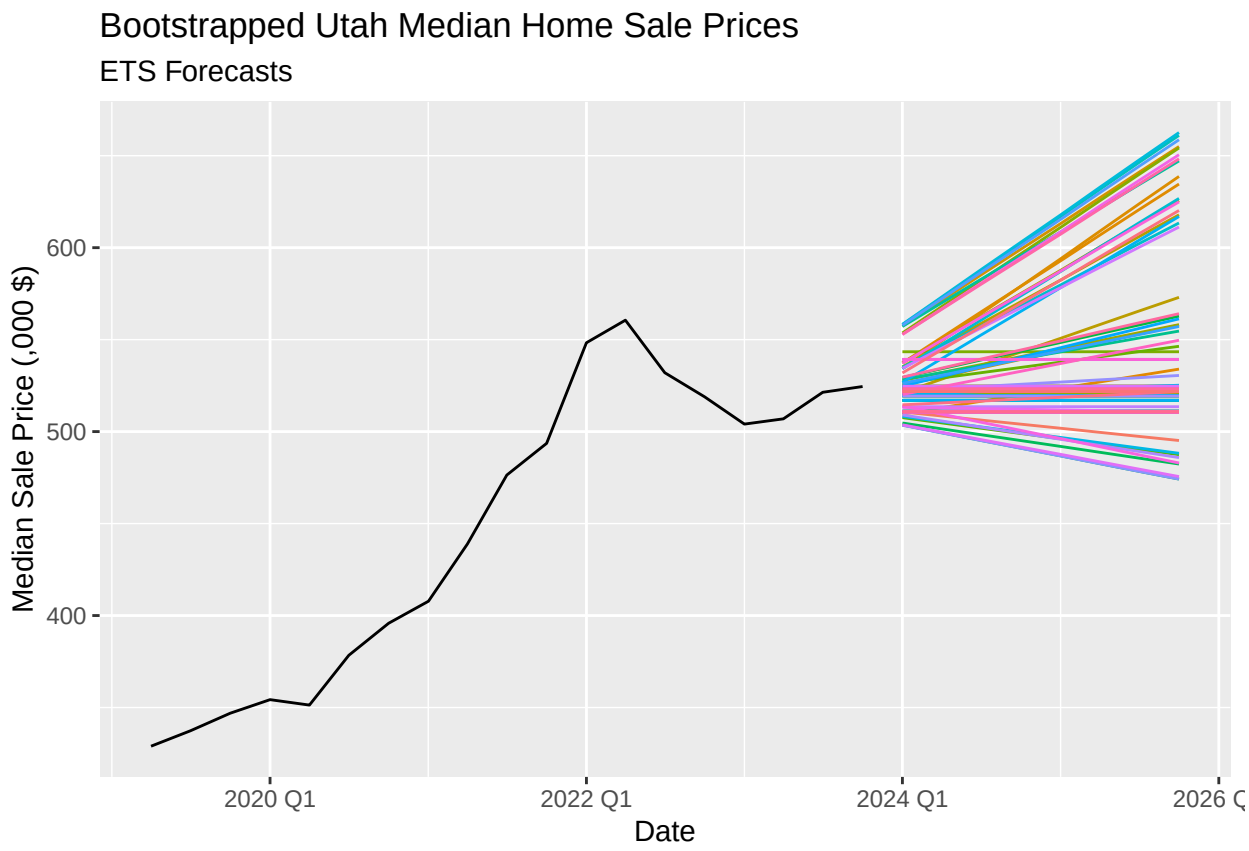


For the forecasting and bagging of Utah Median Home Sale Prices using ETS models:

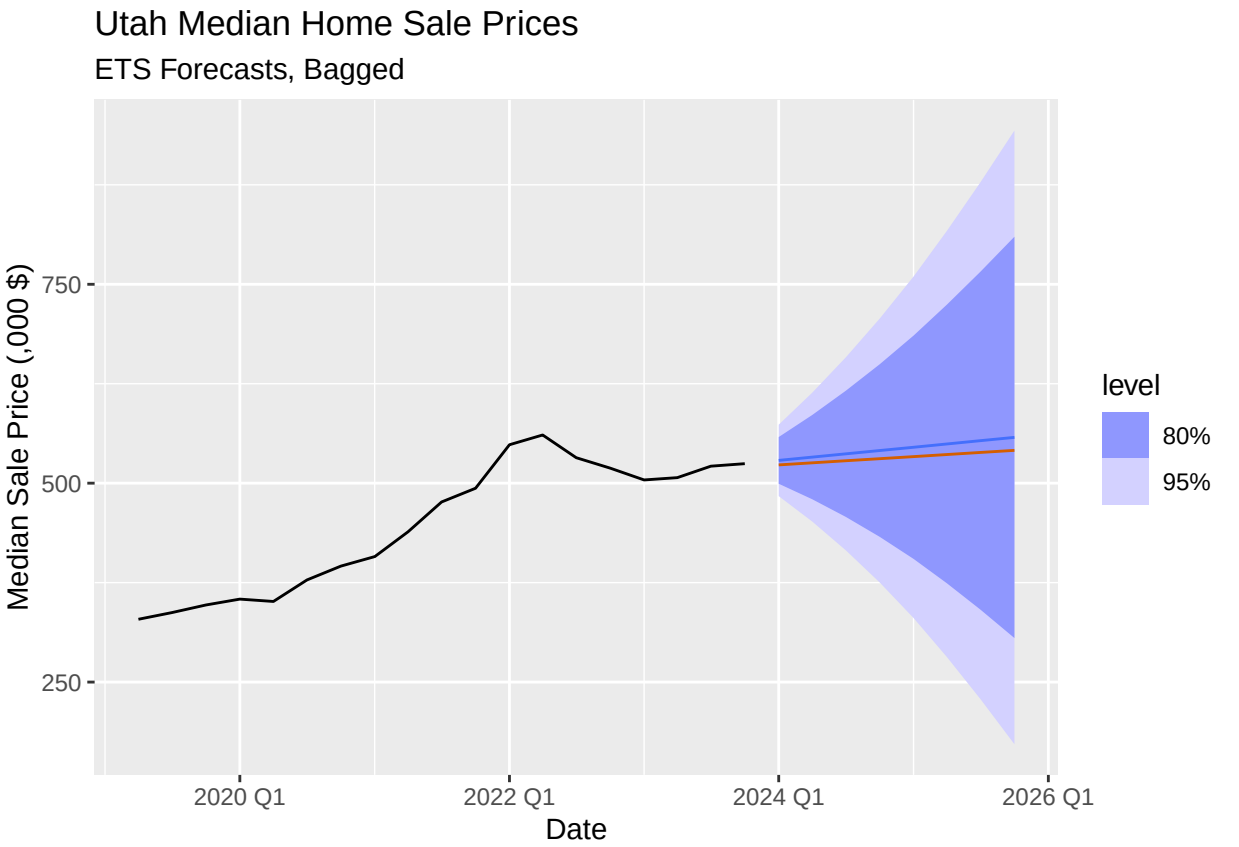
```
Median_Sale_Price_STL %>%
  generate(new_data = Median_Sale, times = 10, bootstrap_block_size = 8) %>%
  autoplot(.sim) +
  autolayer(Median_Sale,Median_Sale_Price_UT) +
  guides(colour = "none") +
  labs(title = "Bootstrapped Simulated Median Sale Price Series",
       x = "Date",
       y = "Median Sale Price (,000$)")
```




```
Generated_Set_2 = Median_Sale_Price_STL %>%
  generate(new_data = Median_Sale, times = 100,
           bootstrap_block_size = 4.24) %>%
  select(-.model,-Median_Sale_Price_UT)

#ETS forecasts
ETS_Forecast = Generated_Set_2 %>%
  model(ets = ETS(.sim)) %>%
  forecast(h = 8)

ETS_Set = ETS_Forecast %>% update_tsibble(key = .rep)

#ETS Plotted
ETS_Set %>% autoplot(.mean) +
  autolayer(Median_Sale,Median_Sale_Price_UT) +
  guides(colour = "none") +
  labs(title = "Bootstrapped Utah Median Home Sale Prices",
       subtitle = "ETS Forecasts",
       x = "Date",
       y = "Median Sale Price (,000 $)")
```




```
#Bagged (Averaged) Set
Bagged = ETS_Set %>% summarise(bagged_mean = mean(.mean))

Median_Sale %>% model(ETS = ETS(Median_Sale_Price_UT)) %>%
  forecast(h = 8) %>%
  autoplot(Median_Sale) +
  autolayer(Bagged,bagged_mean,col = "#D55E00") +
  labs(title = "Utah Median Home Sale Prices",
       subtitle = "ETS Forecasts, Bagged",
       x = "Date",
       y = "Median Sale Price (,000 $)")
```



Bootstrapping and Bagging for Utah Median Home Sale Prices and Housing Starts can be found in the scripts "07a - Bootstrapping Model Forecasts, Median Price Quarterly" and "07b - Bootstrapping Model Forecasts, Housing Starts". Plots of these bootstrapped and bagged models can be found at the end of this document.