Time Series Forecasting Model

Elizabeth Goodwin

3/01/2022

Contents

1 Preparation **1**

1.1 Loading in Data 1

1.2 Splitting Dataset 1

2 Exploration **2**

2.1 Unit Root Testing: 2

2.2 Differencing 2

2.3 Seasonality 3

3 Selecting the model **5**

3.1 Comparing the models 5

4 Results **7**

1 Preparation

1.1 Loading in Data

The first step was loading in the data. The R code is shown below. It imports the .xls file, filters it, and puts it into a tsibble format (tidyverse R package for dealing with time series data).

1.2 Splitting Dataset

Within this project, I used several different test/training split dates to test the model on. The dates reflected in the code below is not representative of all of all of the splits.

```
train36 <- cleaned %>% filter(Date <= yearmonth('201007'))
test36 <- cleaned %>% filter(Date > yearmonth('201007'))
train24 <- cleaned %>% filter(Date <= yearmonth('201107'))
test24 <- cleaned %>% filter(Date > yearmonth('201107'))
train12 <- cleaned %>% filter(Date <= yearmonth('201207'))
test12 <- cleaned %>% filter(Date > yearmonth('201207'))
```

2 Exploration

In Figure 1, you can see the graph, ACF, and PACF figures for the undifferenced data. As you can see, the data is not stationary by default, and the ACF/PACF graphs are all over the place.

```
cleaned %>% gg_tsdisplay(y, plot_type='partial')
```

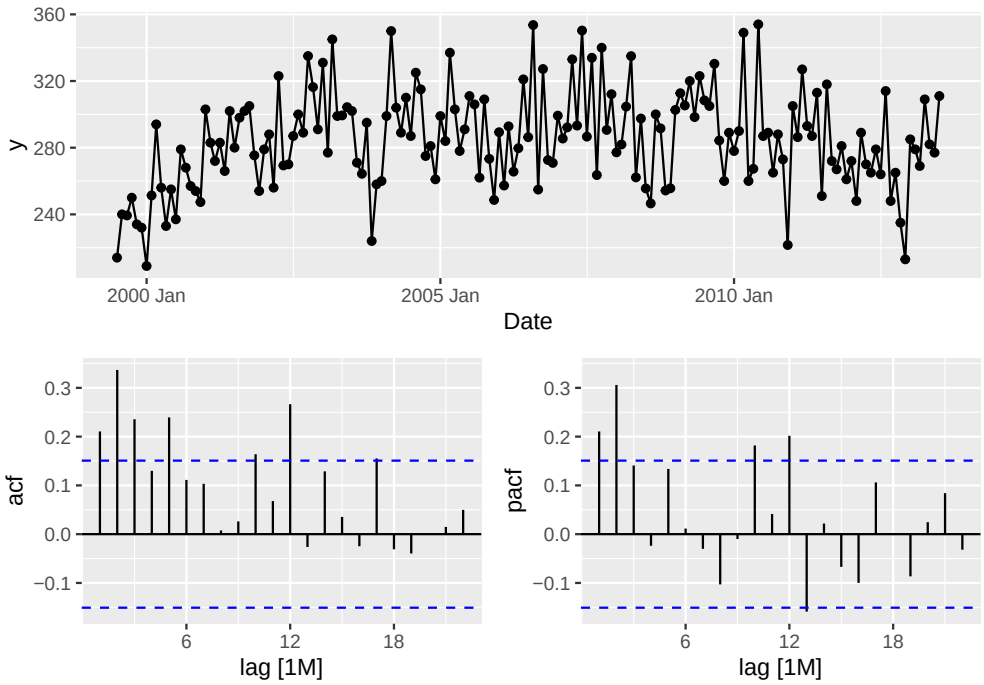

Figure 1: Undifferenced data

2.1 Unit Root Testing:

In this stage I tested the cleaned data using a unit root test. This test (KPSS) is used to tell if differencing is required. The p-value is significant, so differencing is required.

```
cleaned %>% features(y, unitroot_kpss)
```

kpss_stat	kpss_pvalue
0.4768355	0.0468839

2.2 Differencing

I then created took the first differencing. As you can see, in Figure 2, the trend looks far more stationary. In addition, the primary ACF and PACF lags have become insignificant aside from the first. There are still some left over, but largely not. Based on this, an AR(1) and MA(1) model seem likely to be a good choice.

```
diffed <- cleaned %>% mutate(
  y = difference(y)
)
```

```
diffed %>% gg_tsdisplay(y, plot_type='partial')
```

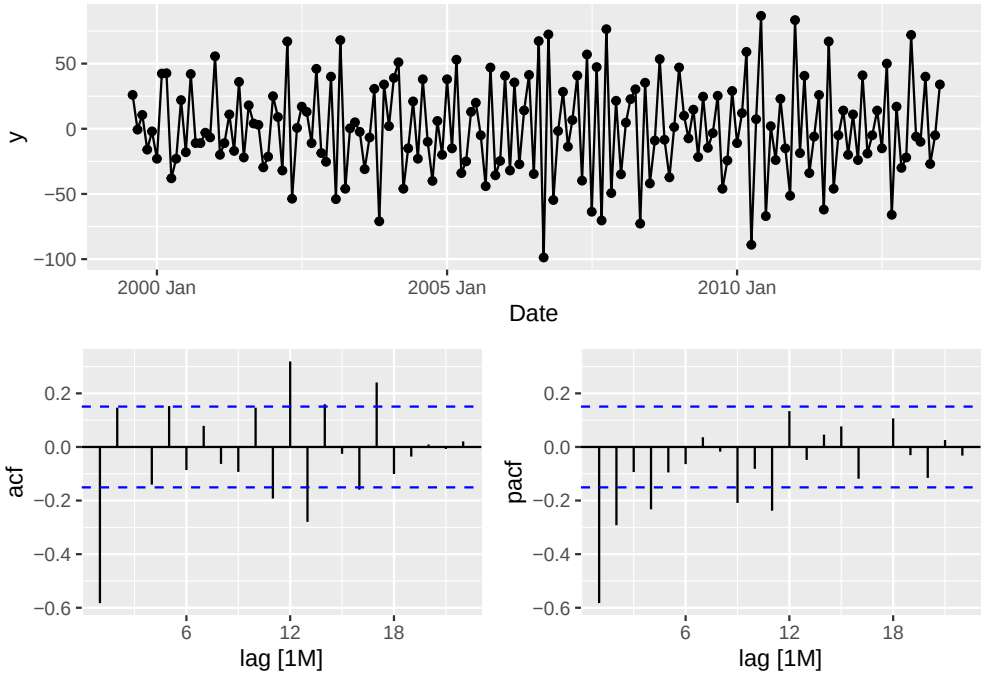

Figure 2: Differenced data

2.3 Seasonality

In Figures 3 and 4, the various years observed are overlayed and color coded to help figure out seasoned trends.

```
cleaned %>% gg_season(y)
```

```
cleaned %>% gg_subseries(y)
```

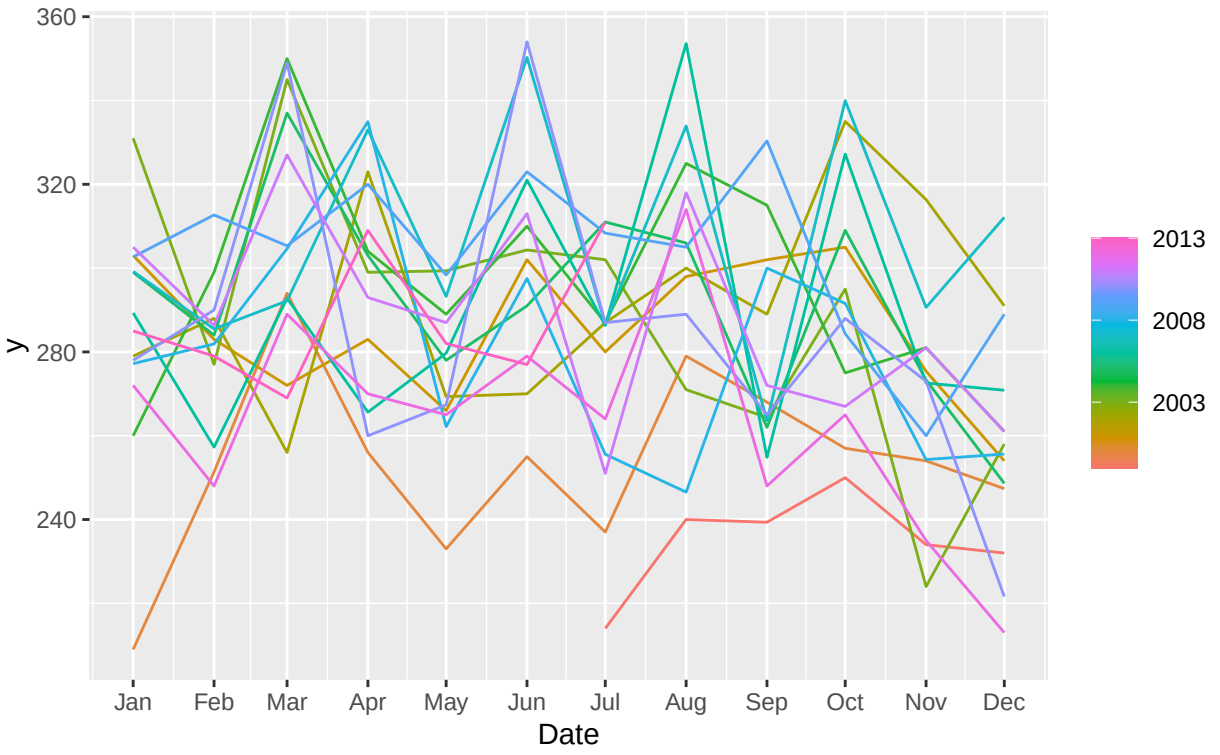

Figure 3: Seasonality

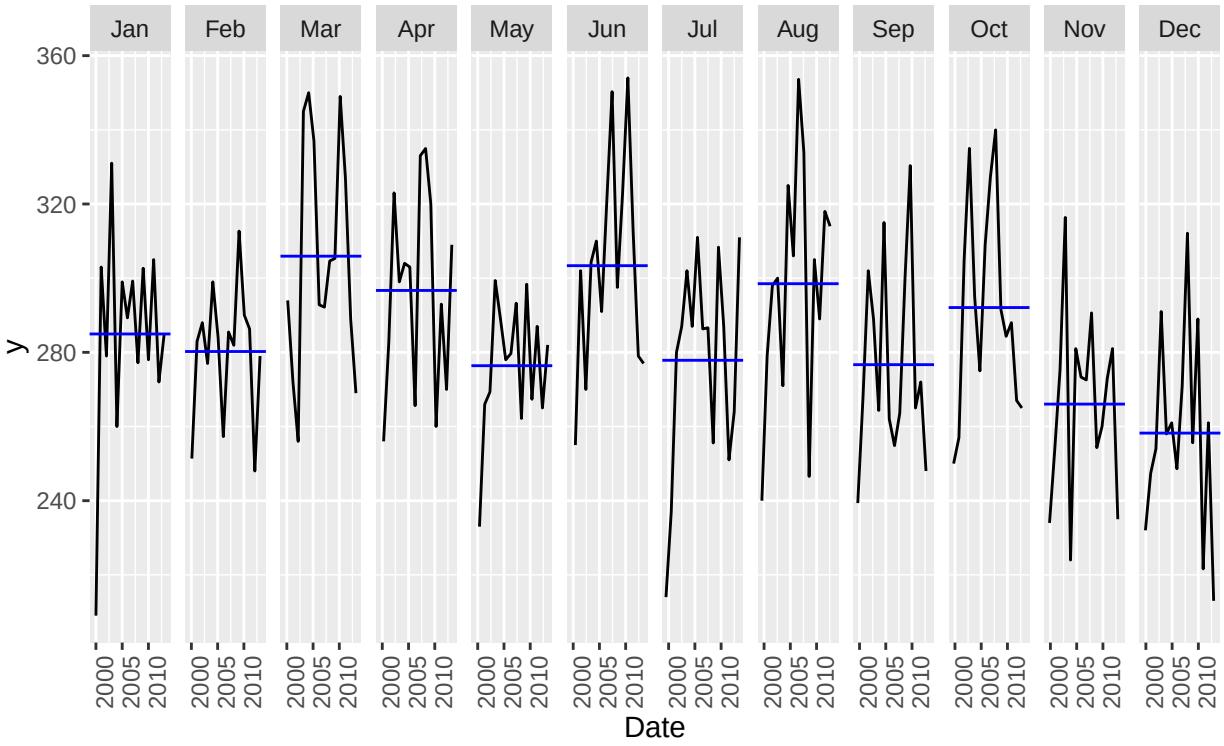

Figure 4: Seasonality 2

3 Selecting the model

There are a few different ways to go about doing this. `fable`, the modeling package I used, has a built in method to automatically select the model it sees as the best. It uses the Hyndman-Khandakar algorithm for model selection. This algorithm works by selecting the every possible combination of p and q to select the combination that minimizes the AICc.

To begin this, I setup the model on a testing/training split with a 36 month test set. I had the algorithm search with one level of differencing, including one layer for seasonal adjustments, and let it pick the rest itself. The model it selected was ARIMA(1,1,1)(2,1,2). I set this as auto36. After that, I ran it again but instead on 24 months of test set. I got the same model as a result. This made me more confident in this model. As a final test, I ran it with just 12 months dedicated to the testing set. This selected ARIMA(1,1,0)(2,1,2). I also let it pick everything for me at 24mo, including the differencing. it selected ARIMA(0,1,1)(1,0,0)

```
# Creating the test/training sets for 36 months.

fit36 <- train36 %>%
  model(
    auto = ARIMA(y ~ pdq(0,0,1) + PDQ(1,0,0)),
    auto36 = ARIMA(y ~ pdq(1,1,1) + PDQ(2,1,2)),
    auto12 = ARIMA(y ~ pdq(1,1,0) + PDQ(2,1,2)),
    )

fit24 <- train24 %>%
  model(
    auto = ARIMA(y ~ pdq(0,0,1) + PDQ(1,0,0)),
    auto36 = ARIMA(y ~ pdq(1,1,1) + PDQ(2,1,2)),
    auto12 = ARIMA(y ~ pdq(1,1,0) + PDQ(2,1,2))
    )

fit12 <- train12 %>%
  model(
    auto = ARIMA(y ~ pdq(0,0,1) + PDQ(1,0,0)),
    auto36 = ARIMA(y ~ pdq(1,1,1) + PDQ(2,1,2)),
    auto12 = ARIMA(y ~ pdq(1,1,0) + PDQ(2,1,2))
    )
fitall <- cleaned %>%
  model(
    auto = ARIMA(y ~ pdq(0,0,1) + PDQ(1,0,0)),
    auto36 = ARIMA(y ~ pdq(1,1,1) + PDQ(2,1,2)),
    auto12 = ARIMA(y ~ pdq(1,1,0) + PDQ(2,1,2))
    )
```

3.1 Comparing the models

Overall, the 12 month model seems to be the worst of them all. While its MAPE in 12 month is the lowest, it is very high in 36 months and middle of the pack in 24 months. The automaticly chosen model is the most interesting. It's comparative lack of seasonal differencing means it looks quite flat in the long term. While MAPE wise it does pretty well, it doesn't actually match up with the model itself all that much. So overall considering that I chose to go with the 36mo model.

```
# 36 Month
forecast <- fit36 %>% forecast(h=36)
accuracy(forecast, test36)
```

.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
auto	Test	-9.867491	26.40861	20.14029	-4.433068	7.744826	NaN	NaN	-0.0667031
auto12	Test	-37.105588	43.25558	37.13176	-14.099575	14.108157	NaN	NaN	0.0574268
auto36	Test	-23.455704	31.44346	25.01778	-9.092043	9.597745	NaN	NaN	-0.0452609

```
# 24 Month
forecast <- fit24 %>% forecast(h=24)
accuracy(forecast, test24)
```

.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
auto	Test	-10.127981	25.63515	20.28655	-4.479767	7.720835	NaN	NaN	0.1181119
auto12	Test	12.539177	25.27868	20.52985	4.162131	7.361535	NaN	NaN	-0.1973717
auto36	Test	-7.301252	22.00771	19.32504	-3.161749	7.274366	NaN	NaN	-0.1408967

```
# 12 Month
forecast <- fit12 %>% forecast(h=12)
accuracy(forecast, test12)
```

.model	.type	ME	RMSE	MAE	MPE	MAPE	MASE	RMSSE	ACF1
auto	Test	-6.323668	29.26909	23.17077	-3.5385833	9.050937	NaN	NaN	0.1619994
auto12	Test	1.919820	23.60154	21.82632	-0.1563025	8.100266	NaN	NaN	-0.0364685
auto36	Test	3.907402	25.40634	23.26511	0.5267491	8.496932	NaN	NaN	-0.1243428

4 Results

I wasn't sure what exactly you needed, so I included a forecast for the next 24 months. Shown below. The "mean" values are the forecasted values.

```
fitall %>% select(auto36) %>% forecast(h=24)
```

.model	Date	y	.mean
auto36	2013 Aug	N(295, 613)	295.3424
auto36	2013 Sep	N(279, 621)	278.6221
auto36	2013 Oct	N(292, 664)	292.2349
auto36	2013 Nov	N(267, 700)	266.5490
auto36	2013 Dec	N(261, 737)	260.8714
auto36	2014 Jan	N(283, 773)	282.8120
auto36	2014 Feb	N(279, 810)	279.3972
auto36	2014 Mar	N(306, 846)	306.0401
auto36	2014 Apr	N(296, 883)	295.6113
auto36	2014 May	N(275, 920)	275.4459
auto36	2014 Jun	N(303, 956)	303.1209
auto36	2014 Jul	N(282, 993)	281.6202
auto36	2014 Aug	N(302, 1058)	302.0092
auto36	2014 Sep	N(280, 1097)	280.0108
auto36	2014 Oct	N(296, 1140)	295.8225
auto36	2014 Nov	N(270, 1183)	269.6804
auto36	2014 Dec	N(259, 1225)	259.4270
auto36	2015 Jan	N(292, 1268)	292.0603
auto36	2015 Feb	N(286, 1310)	286.2117
auto36	2015 Mar	N(310, 1353)	309.6204
auto36	2015 Apr	N(303, 1395)	303.4486
auto36	2015 May	N(283, 1438)	282.7267
auto36	2015 Jun	N(308, 1480)	308.0592
auto36	2015 Jul	N(288, 1524)	287.6655

sss

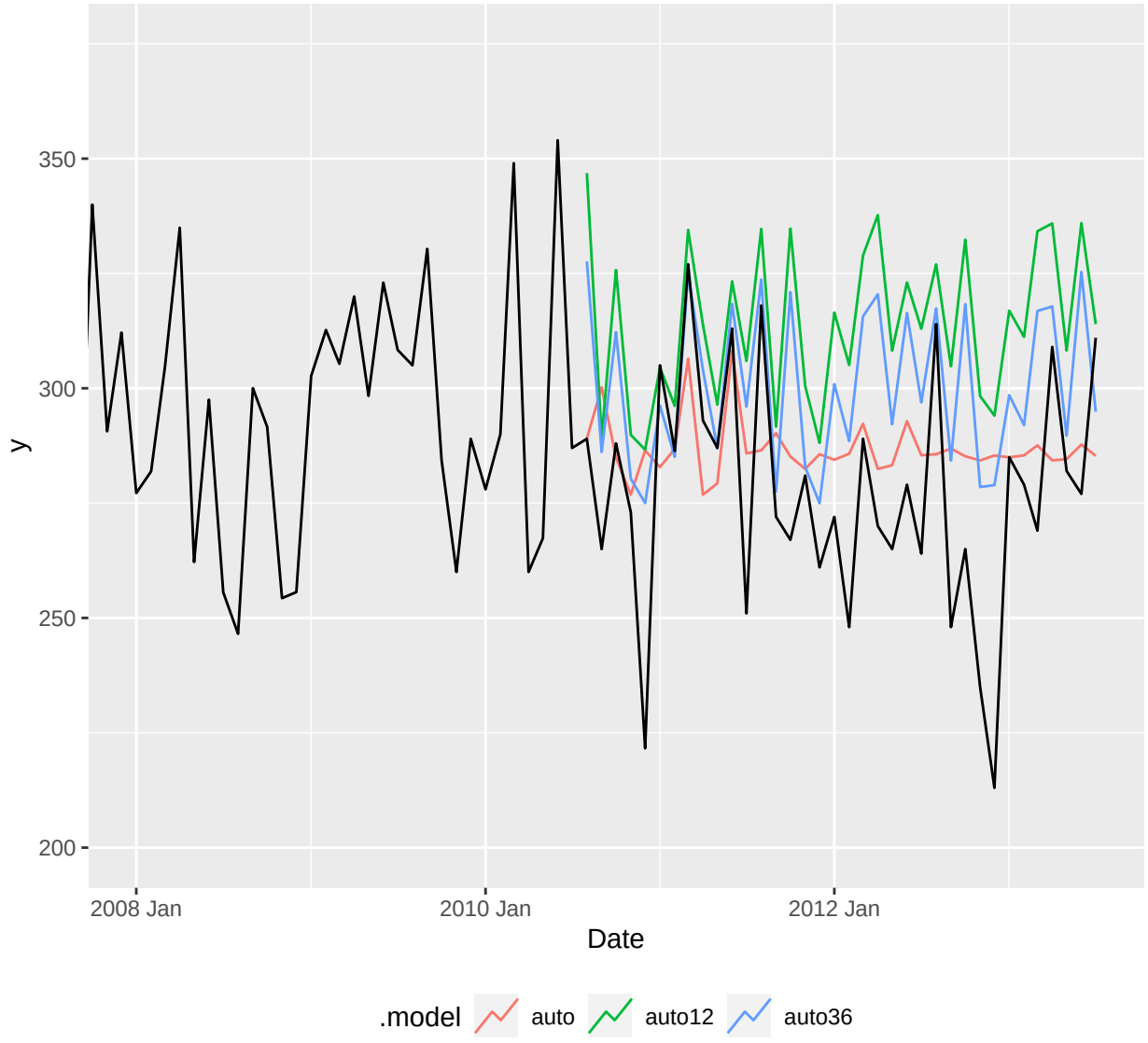

Figure 5: 36 Month Split Comparison

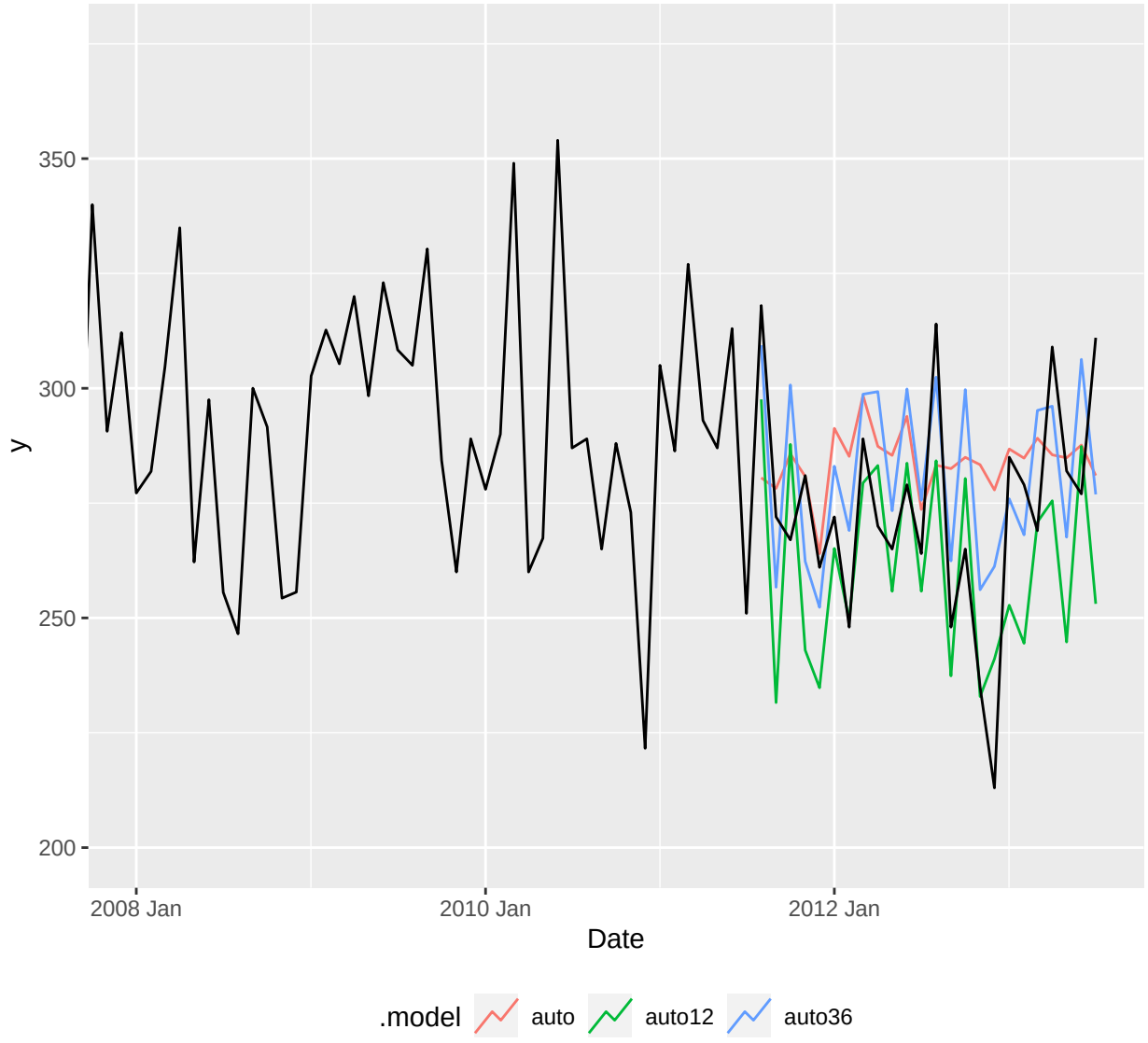

Figure 6: 24 Month Split Comparison

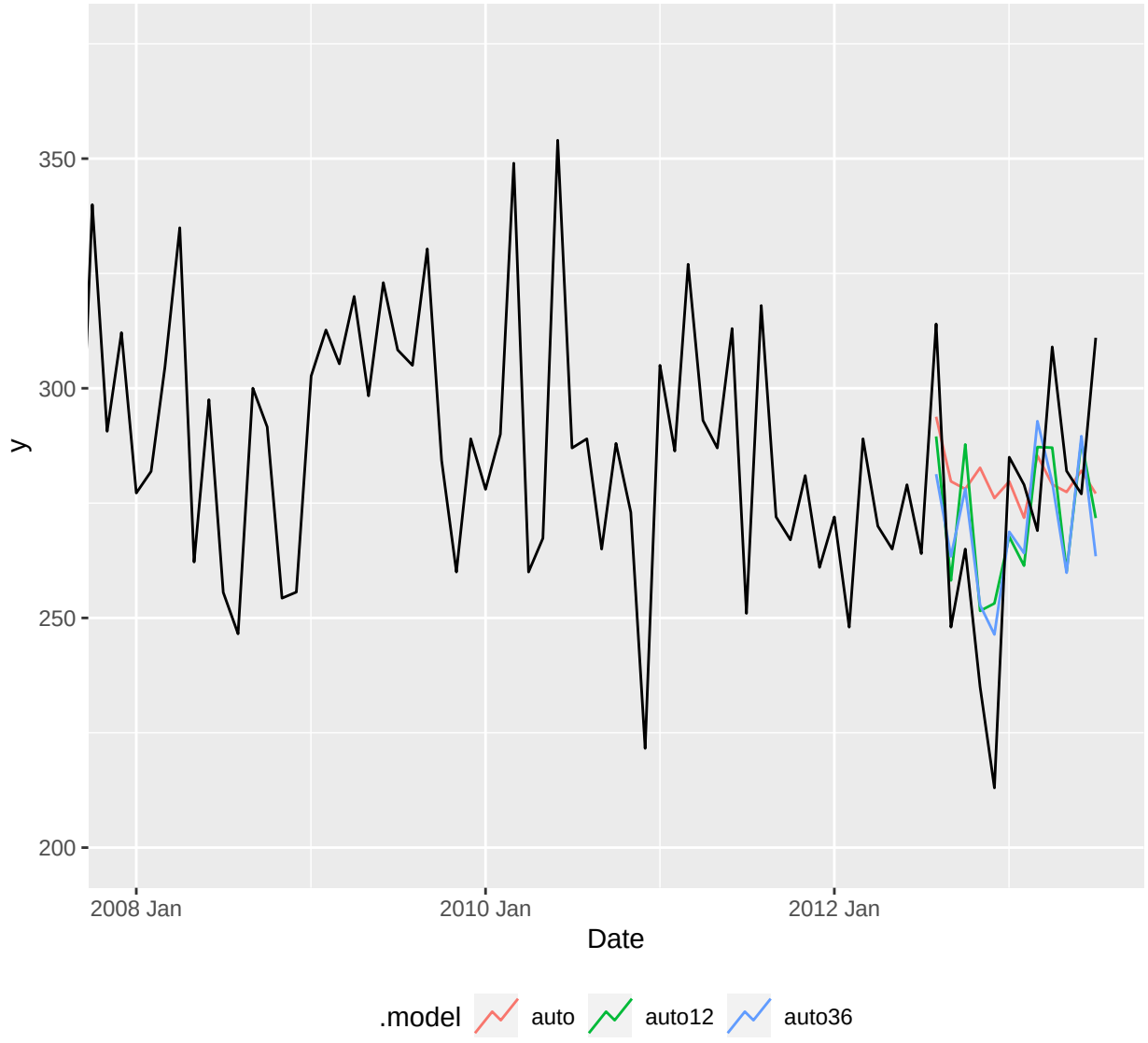

Figure 7: 12 Month Split Comparison

Figure 8: Future Projection