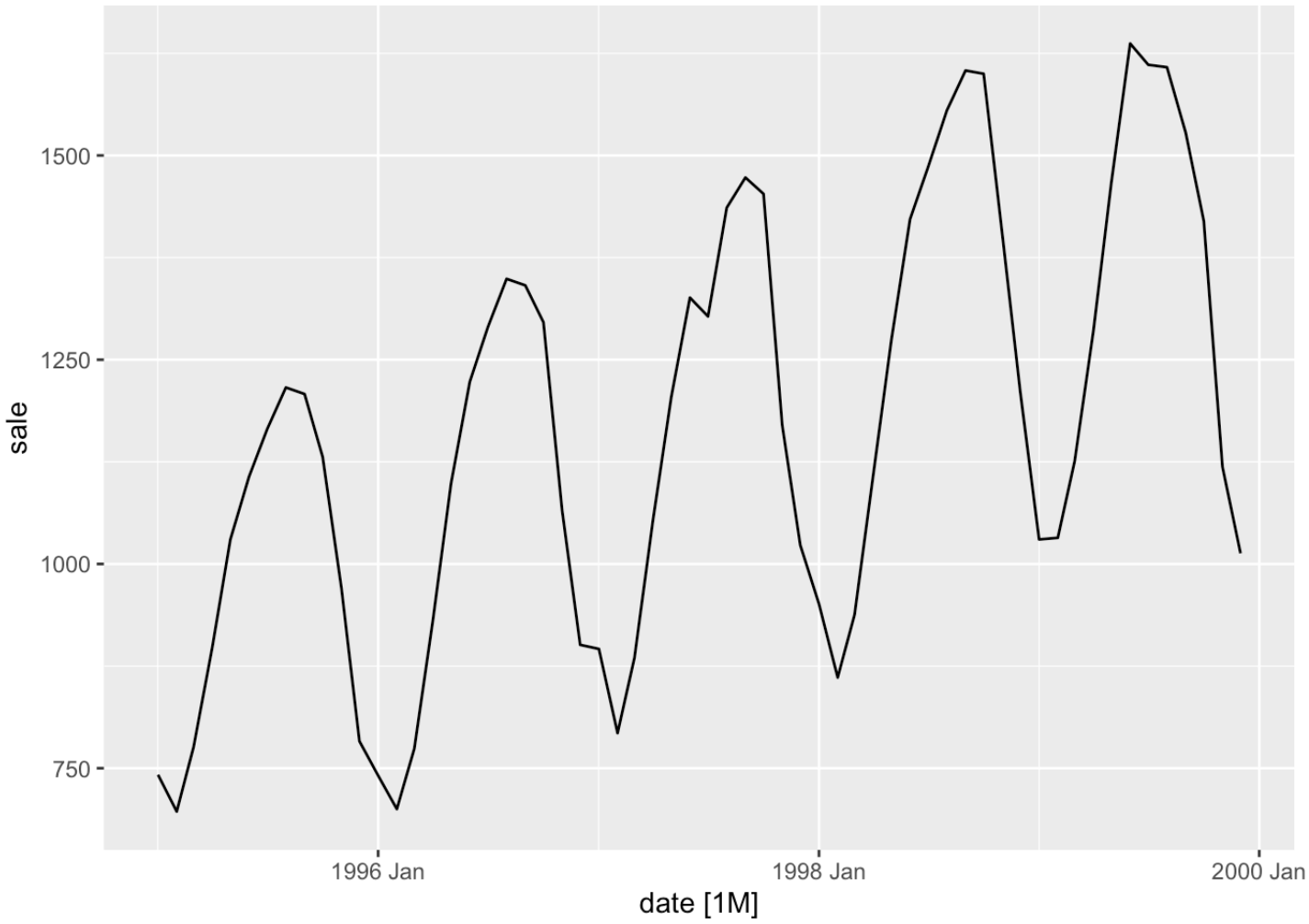
DATA Plastics:

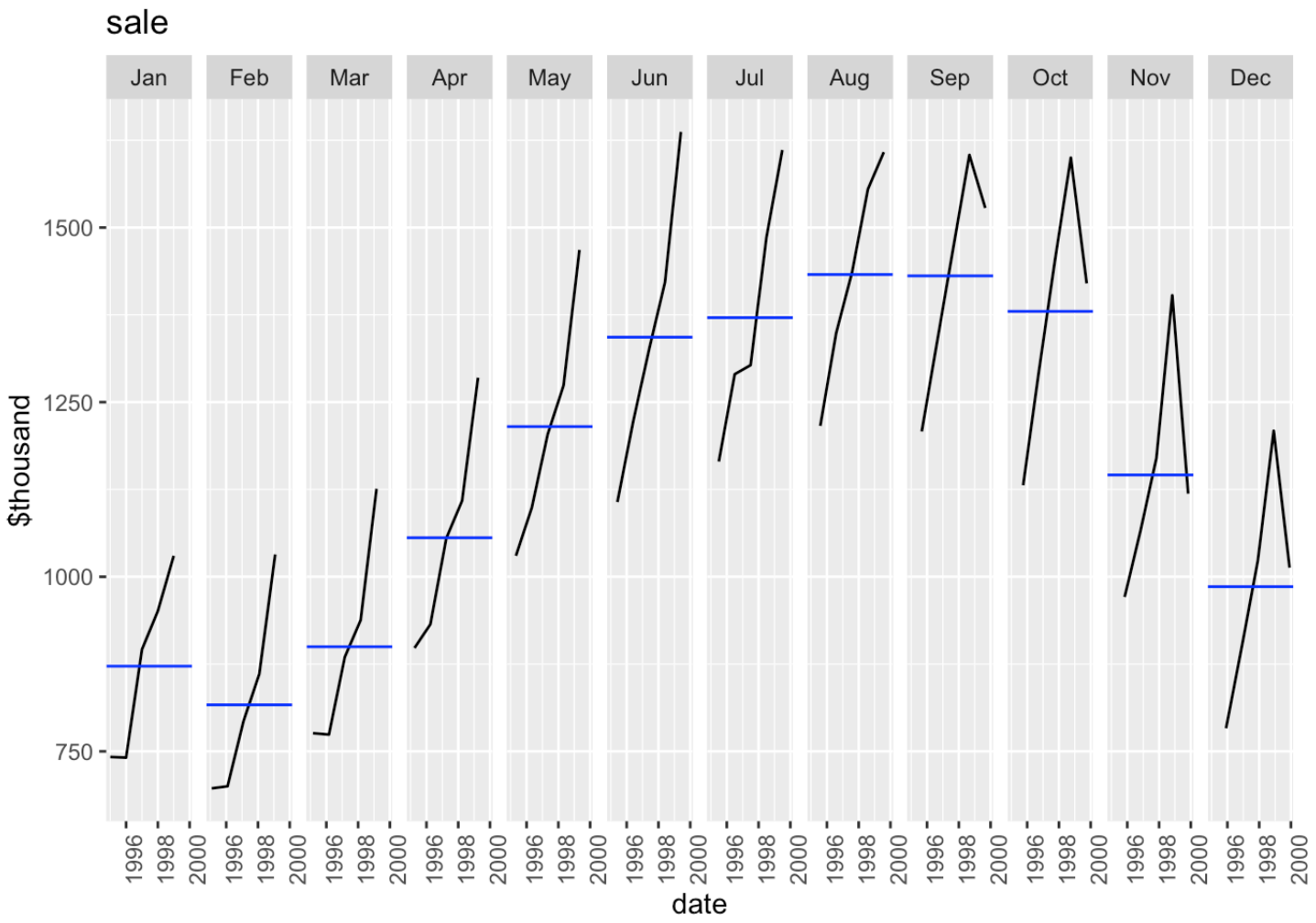
```
df <- readr::read_csv("plastics.csv",show_col_types = FALSE)
df = df%>%mutate(date = yearmonth(date))%>%
  tsibble(index = date)
df
```

```
## # A tsibble: 60 x 2 [1M]
##        date  sale
##       <mth> <dbl>
##  1 1995 Jan   742
##  2 1995 Feb   697
##  3 1995 Mar   776
##  4 1995 Apr   898
##  5 1995 May  1030
##  6 1995 Jun  1107
##  7 1995 Jul  1165
##  8 1995 Aug  1216
##  9 1995 Sep  1208
## 10 1995 Oct  1131
## # ℹ 50 more rows
```

```
df %>% autoplot(sale)
```




```
df %>% gg_subseries(sale) +labs(y = "$thousand", title = "sale")
```

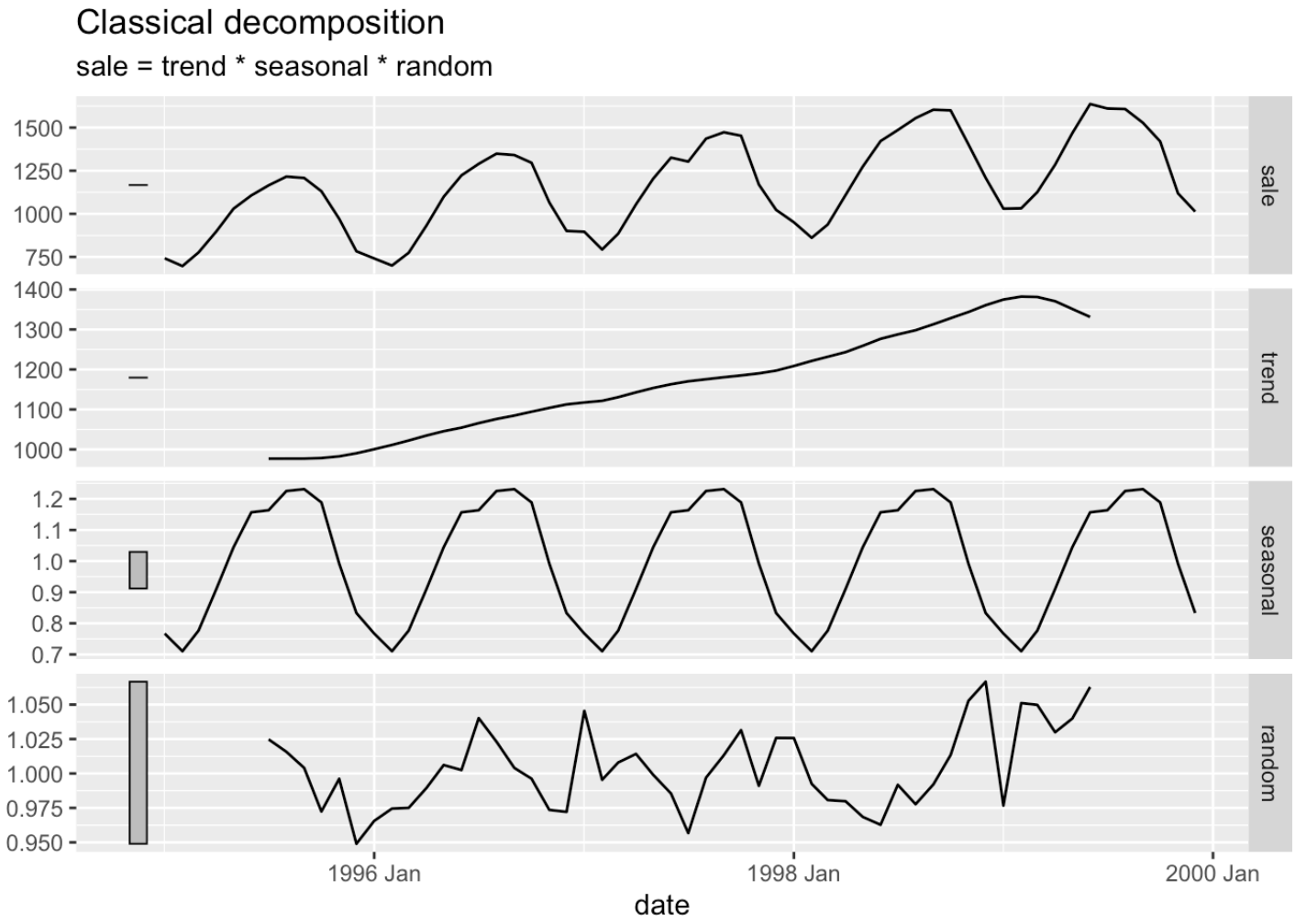


We can observe a steady increase in sales, which indicate the presence of trend in the data also we can observe not much difference in variance. With trend we can observe a seasonal behavior where every year around 6-8 month there is spike in sales and it has reduced by end of year. Therefore, time series plot shows seasonal fluctuations as well as a trend-cycle in Product A sales.

From the above sub- series plot, we can confirmly say there is a seasonality as every year in August we can see high sales of Product A.

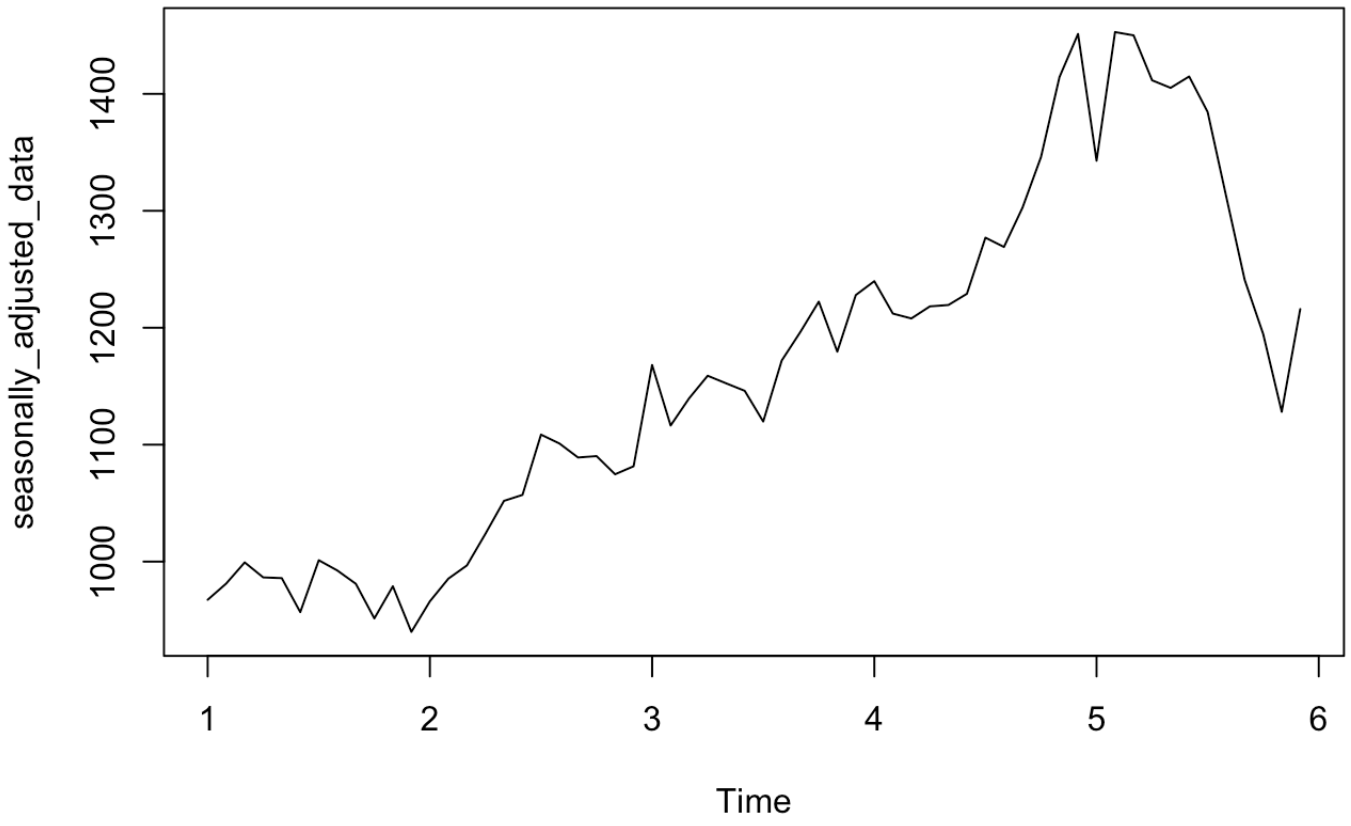
```
dcmp = df %>%model(classical_decomposition(sale, type='m'))
components(dcmp) %>% autoplot()
```

```
## Warning: Removed 6 rows containing missing values (`geom_line()`).
```

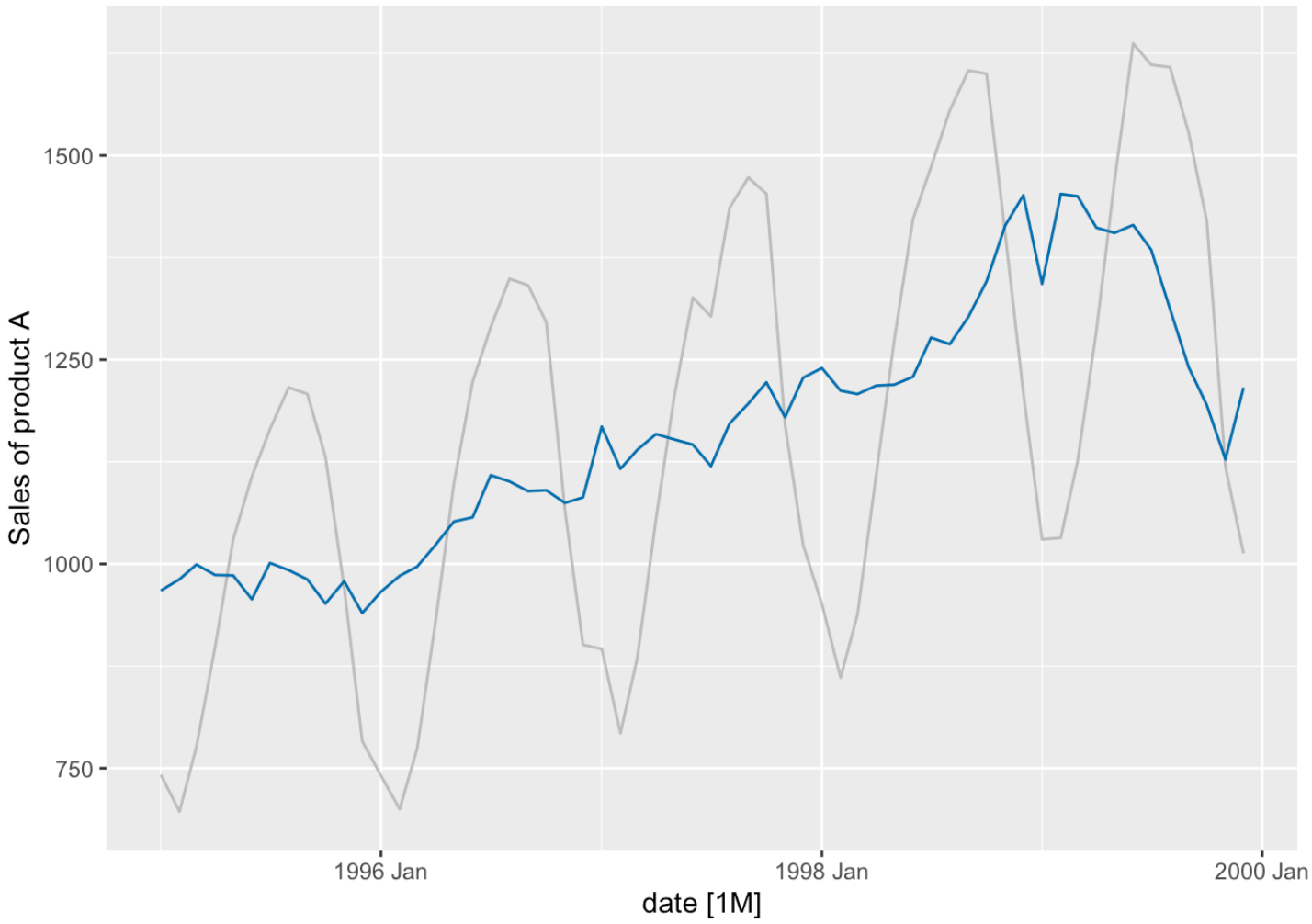



```
df1 <- ts(df$sale, frequency = 12)
decomp1 <- decompose(df1, type = 'multiplicative')
trend <- decomp1$trend
seasonal <- decomp1$seasonal
rand <- decomp1$random
```

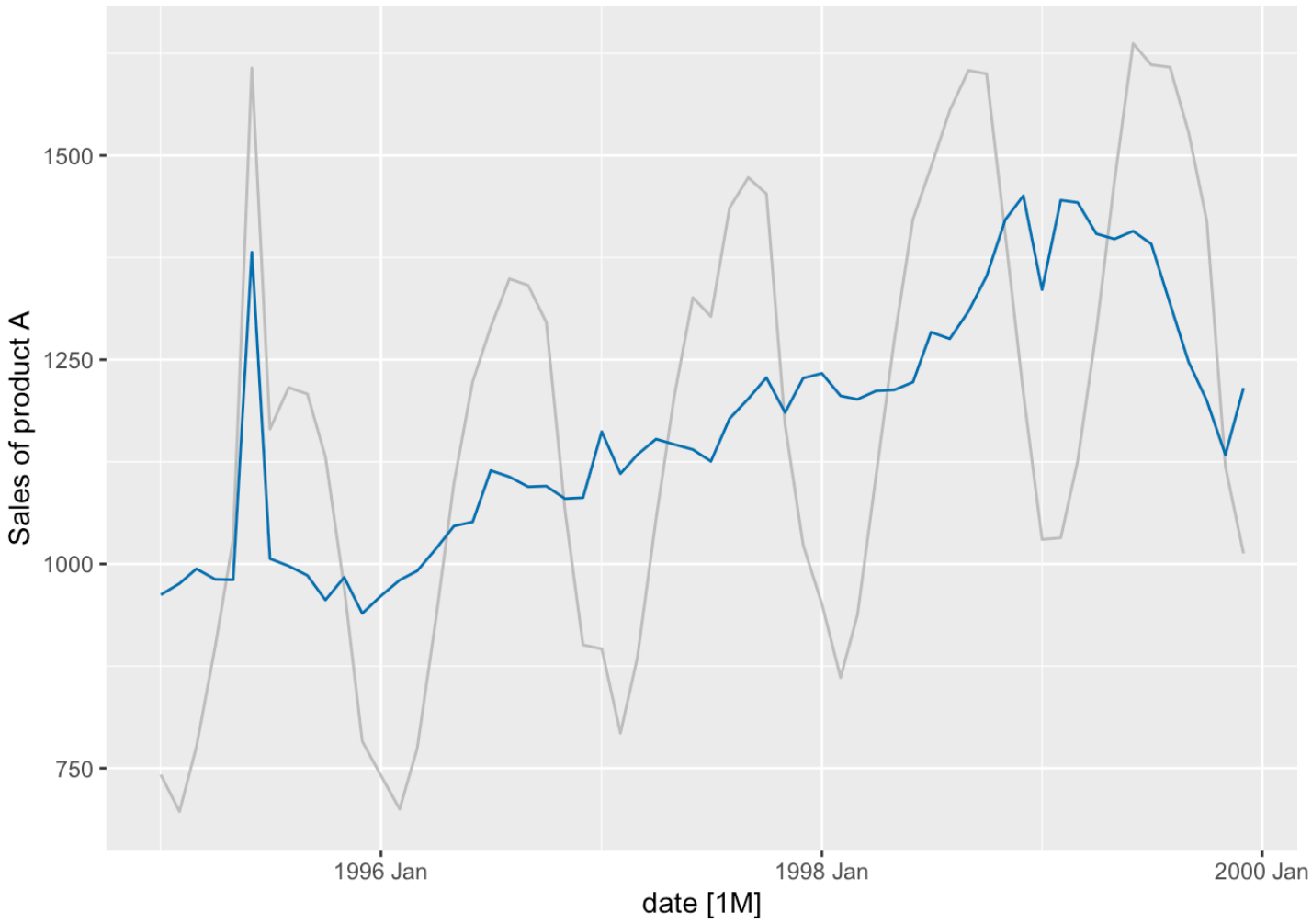
```
seasonally_adjusted_data = df$sale / seasonal
plot(seasonally_adjusted_data)
```




```
df %>% autoplot(sale, color='gray') + autolayer(components(dcmp), season_adjust, colo
r='#0072B2') +labs(y="Sales of product A")
```




```
df1 = df
df1$sale[6] <- df1$sale[6] + 500
dcmp <- df1 %>%model(classical_decomposition(sale, type='m'))
seasonally_adjusted_data <- df1$sale / seasonal
df1 %>% autoplot(sale, color='gray') + autolayer(components(dcmp), season_adjust, col
or='#0072B2') +labs(y="Sales of product A")
```

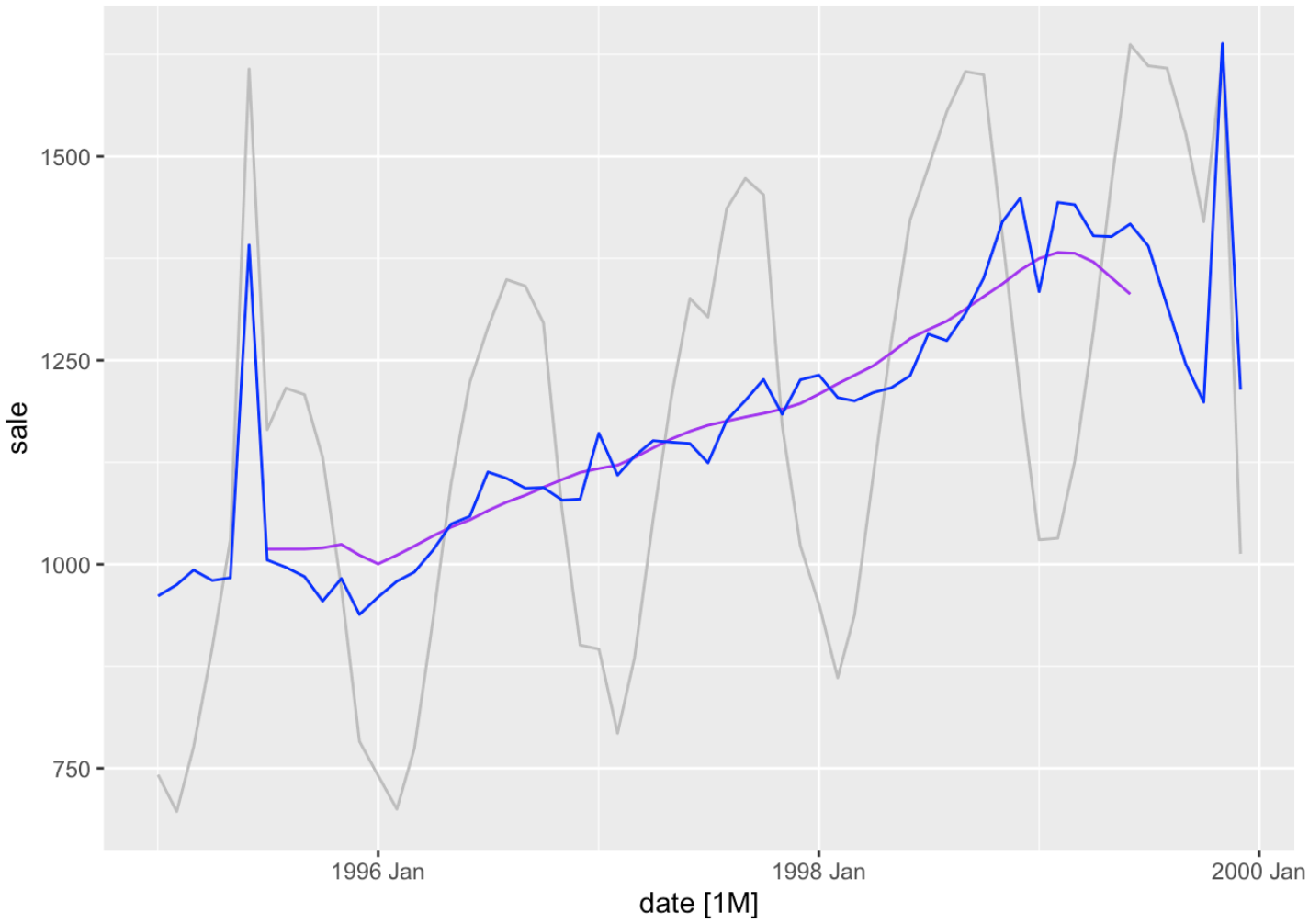


tip: use autoplot to plot original and add outlier plot with autolayer

An Effect of out-lier is causing cause skewed seasonally adjusted data, making it more difficult to determine the underlying trend and seasonal trends.Both trends and seasonal trend is been effected by the out-lier making hard to find the real pattern and lead to false prediction. But as the outlier is in initial year data, the prediction is not much effected. Still, outliers does effect in the time series.

```
df1$sale[59] <- df1$sale[59] + 500
dcmp1 <- df1 %>% model (classical_decomposition(sale, type='m'))
df1 %>%autoplot(sale, color = "gray") +autolayer(components(dcmp), trend, color = 'pu
rple')+
autolayer(components(dcmp1), season_adjust, color ='blue')
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```



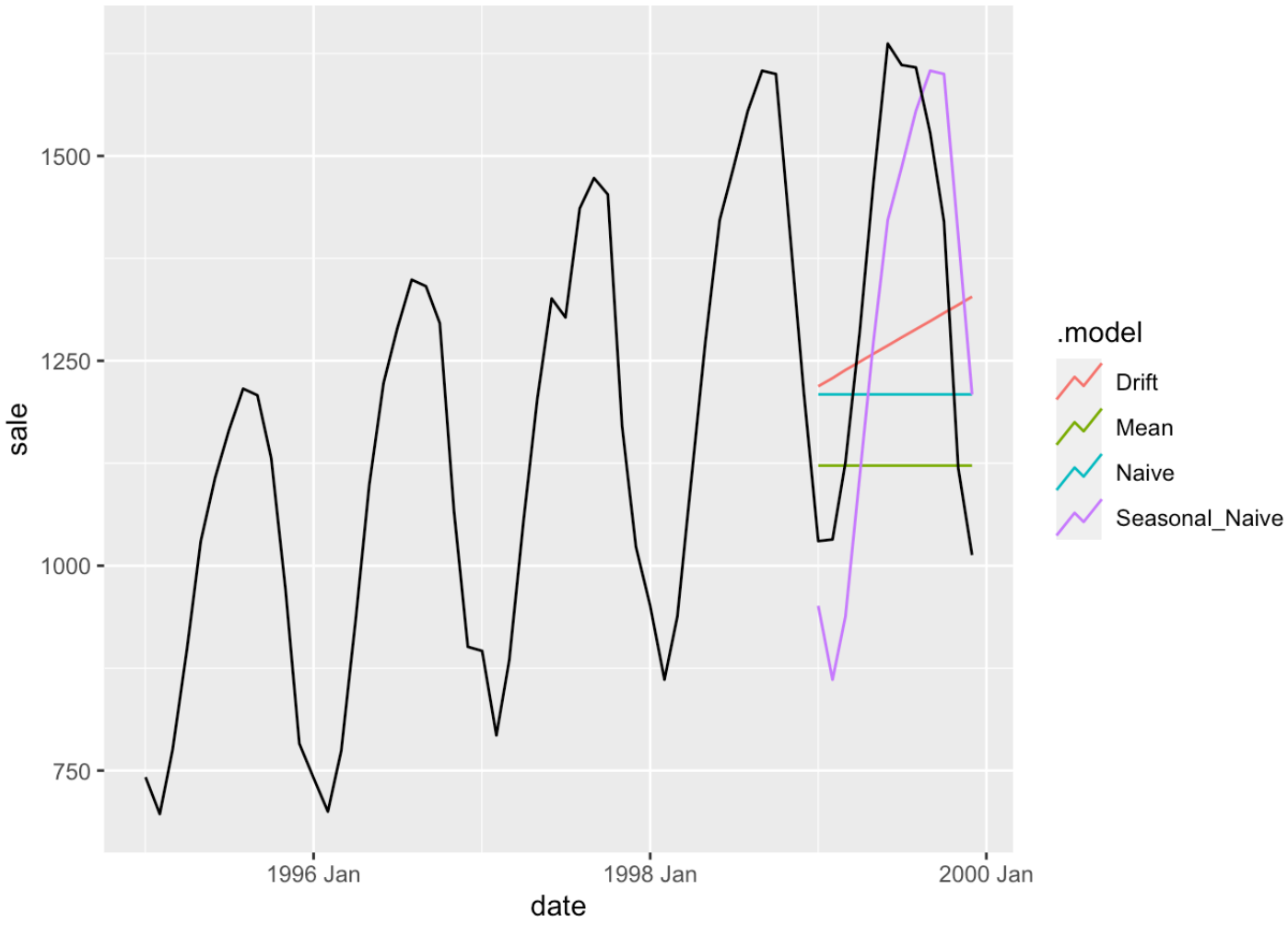
The position of an outlier in a time series is important because it influences the estimated trend and seasonality components. Outliers have a more obvious influence on recent data when they are at the end, potentially distorting recent trends. Outliers in the middle, on the other hand, have a lesser influence, mostly impacting the surrounding time range. The structure of the data and the time of the outlier have a role as well, and dealing with outliers at the end can be more difficult.For seasonality the if outlier is at the end of the time series, it may have less effect on the estimation of the seasonal component than if it were in the middle. Seasonal components are frequently calculated based on data trends, and outliers can alter these patterns.

```
train <- df %>% filter(year(date) < 1999)
test <- df %>% filter(year(date) == 1999)
```

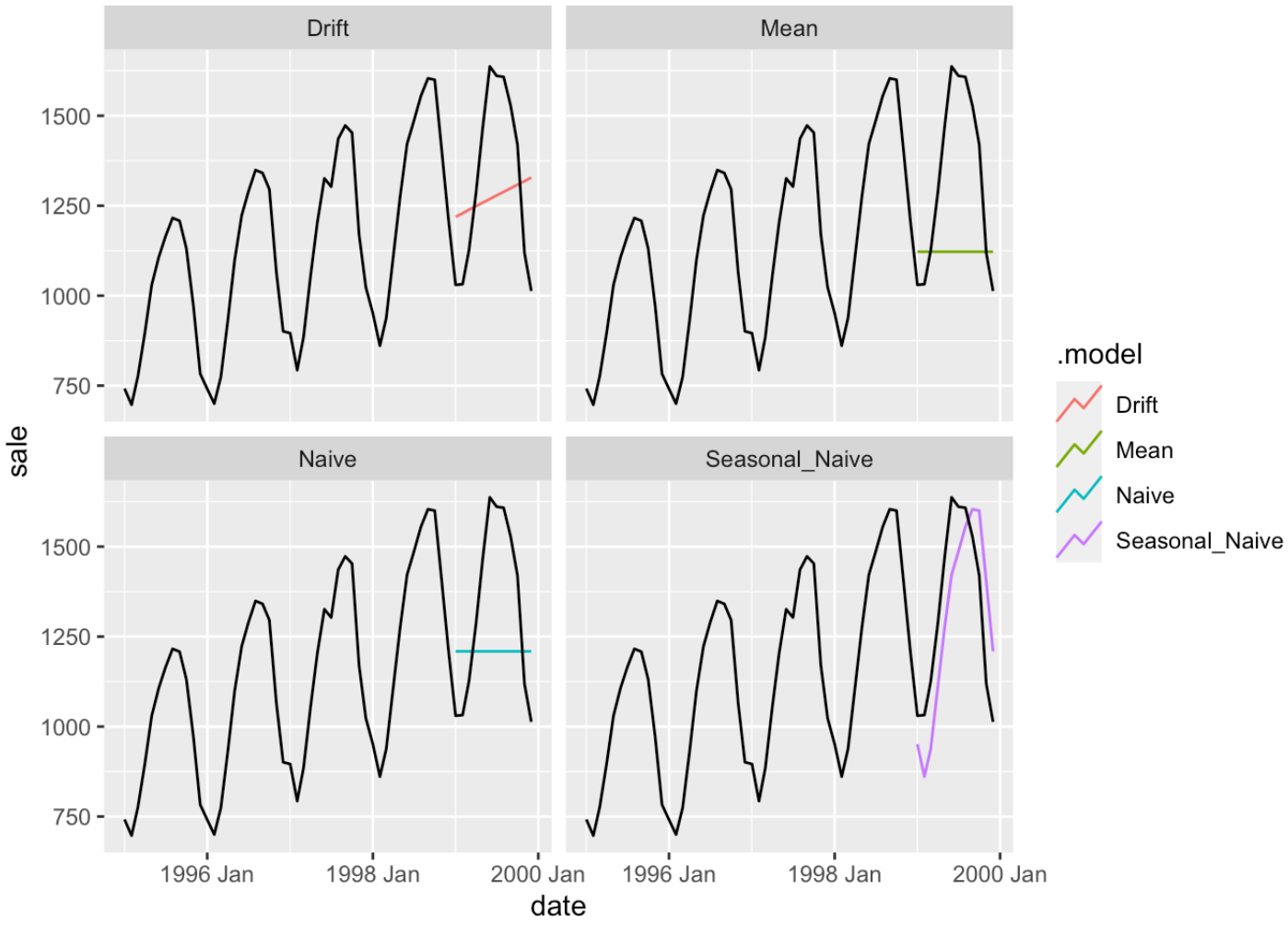
```
fit <- train %>%
  model(
    Mean = MEAN(sale),
    Naive = NAIVE(sale),
    Seasonal_Naive = SNAIVE(sale),
    Drift = RW(sale ~ drift())
  )
fc <- fit %>% forecast(h = 12)
accuracy(fc,df)
```

```
## # A tibble: 4 × 10
##   .model         .type    ME  RMSE   MAE    MPE  MAPE  MASE RMSSE  ACF1
##   <chr>          <chr> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Drift          Test   49.5  239.  218.  0.551  16.6  2.13  2.07 0.689
## 2 Mean           Test  201.   312.  250. 12.3    17.1  2.44  2.70 0.708
## 3 Naive          Test  114.   265.  235.  5.49   17.0  2.29  2.29 0.708
## 4 Seasonal_Naive Test   38.8  174.  161.  2.47   12.9  1.57  1.50 0.817
```

```
fc %>% autoplot(df,level = NULL)
```



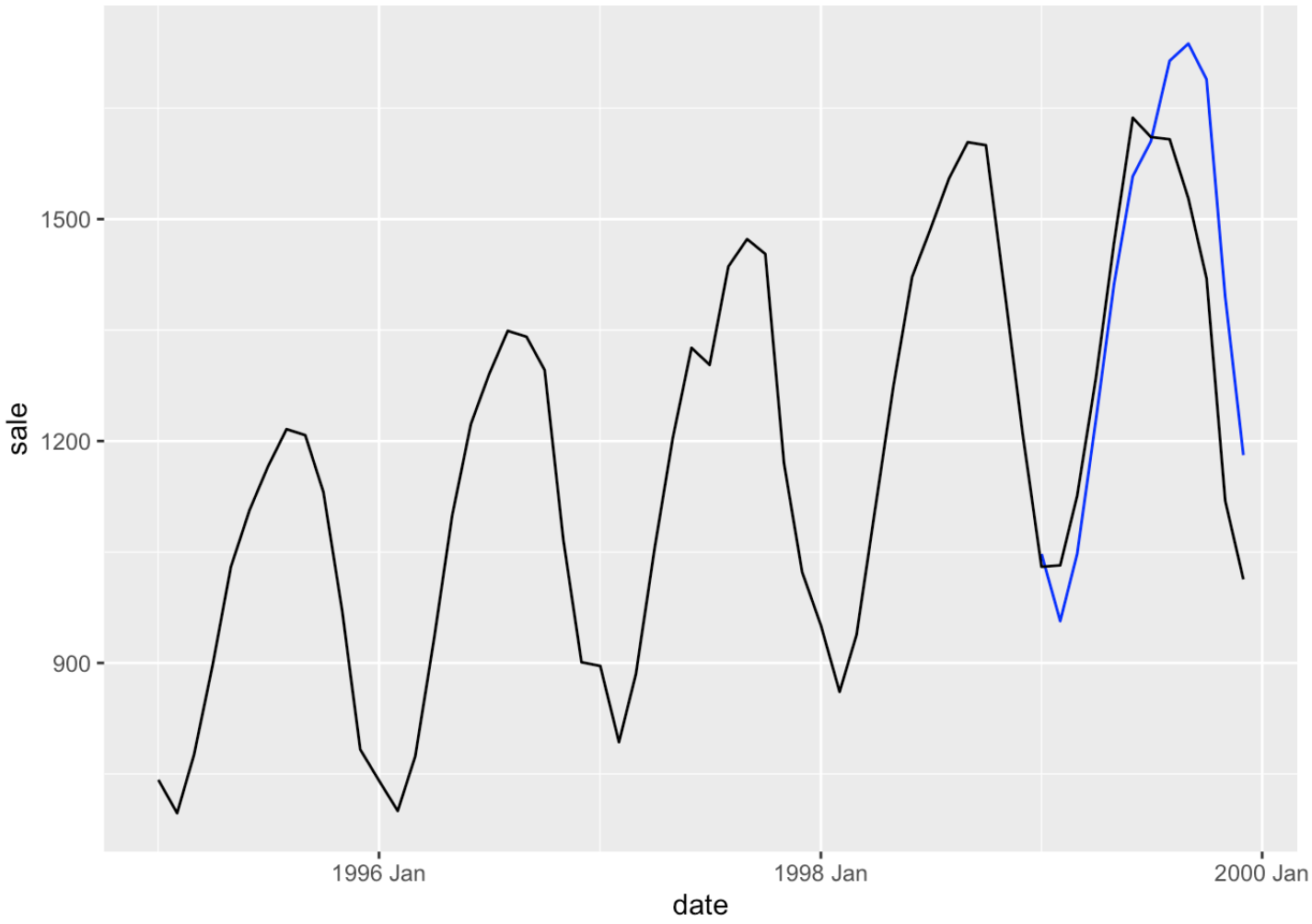

```
fc %>% autoplot(df,level = NULL) + facet_wrap(~.model)
```



The Seasonal Naive Method looks to be the most effective. It has the lowest MAE, RMSE, and MAPE, implying that it gives the most accurate data projections. The Drift approach also performs well, but in terms of MAE and MAPE, the Seasonal Naive method exceeds it. The MAE and MAPE scores for the Mean and Naive techniques are greater, suggesting less accurate projections. Even the predicted plot to actual confirms the choice of Seasonal Naive is performing better.

Among 4 models, The Seasonal Naive Method is the best choice for predicting for the above data-set.

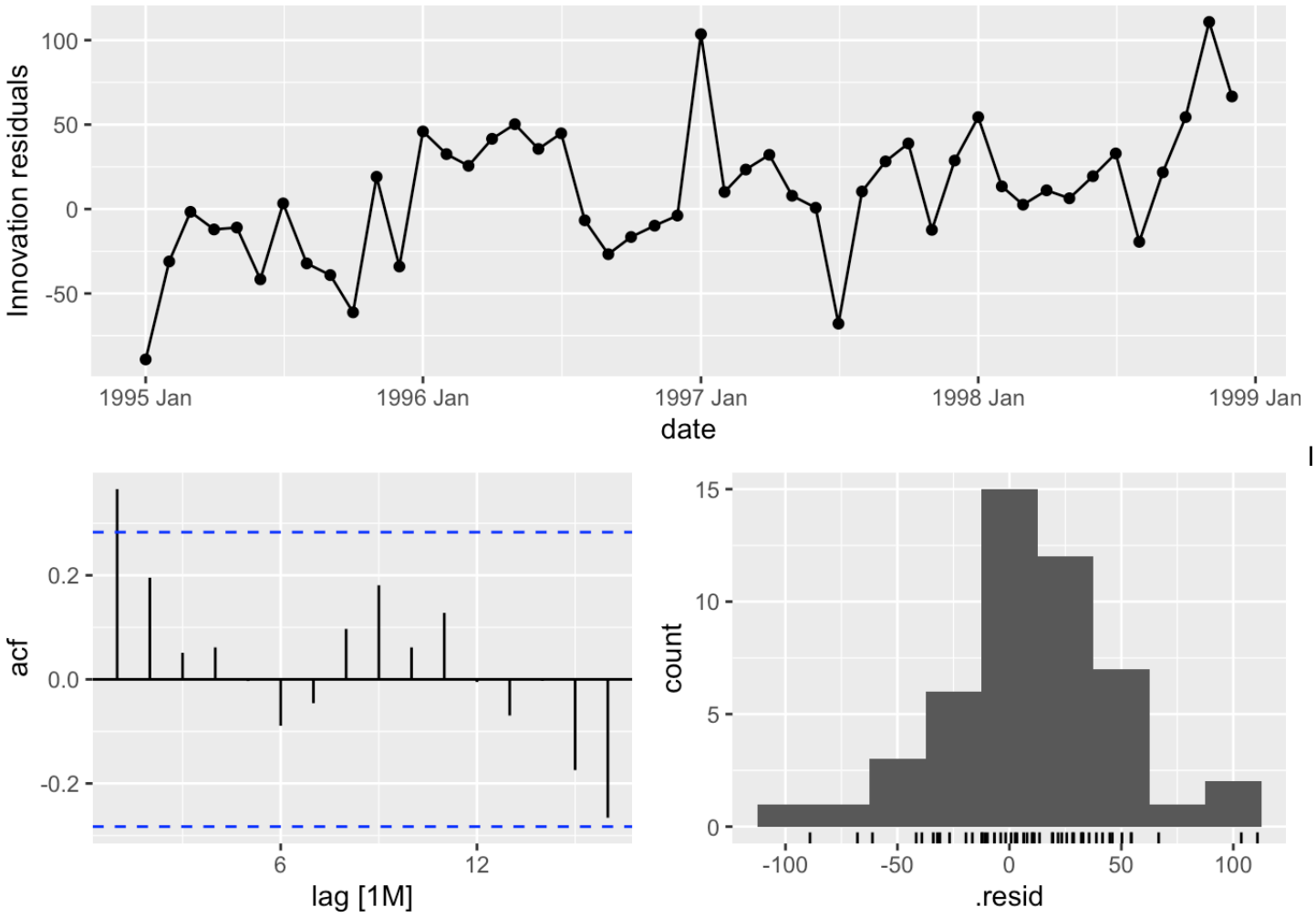
```
fit <- train %>%
  model(
    ANM = ETS(sale ~ error("A") + trend("N") + season("M"))
  )

fit <- fit %>% dplyr::select(ANM)
fc <- fit %>% forecast(h = "1 years")

fc %>% autoplot(df,level = NULL)
```




```
accuracy(fc,df)
```

```
## # A tibble: 1 × 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ANM    Test  -57.8  146.  116. -4.54  9.15  1.13  1.26 0.845
```

```
gg_tsresiduals(fit)
```



choose model ANM to be the best compare to MNN

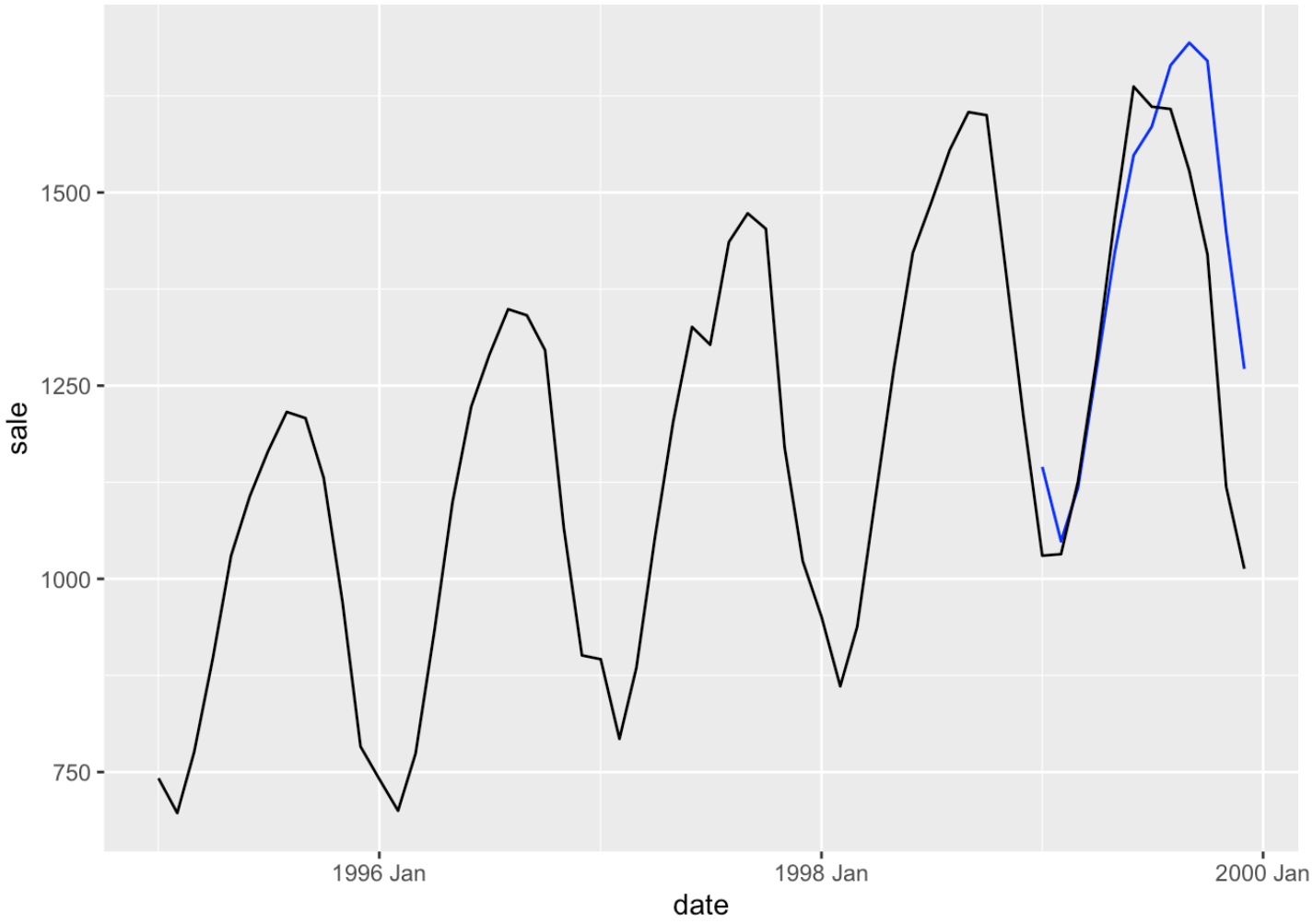
```
fit <- train %>%
  model(
    arima  = ARIMA(sale ~ pdq(1,0,0) + PDQ(0,1,1))
  )
accuracy(fit)
```

```
## # A tibble: 1 × 10
##   .model .type       ME  RMSE   MAE    MPE  MAPE  MASE RMSSE    ACF1
##   <chr>  <chr>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 arima  Training 0.703  31.1  21.7 -0.235  1.94 0.211 0.269 -0.0620
```

```
report(fit[1])
```

```
## Series: sale
## Model: ARIMA(1,0,0)(0,1,1)[12] w/ drift
##
## Coefficients:
##          ar1     sma1  constant
##       0.7769  -0.5223   23.1041
## s.e.  0.1237   0.3394    3.8041
##
## sigma^2 estimated as 1410:  log likelihood=-182.35
## AIC=372.7   AICc=373.99   BIC=379.03
```

```
fc <- fit %>% forecast(h = "1 year")
fc %>% autoplot(df,level = NULL)
```




```
accuracy(fc,df)
```

```
## # A tibble: 1 × 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 arima  Test  -84.4  156.  114. -7.33  9.30  1.12  1.35 0.794
```

The root mean square error (RMSE) is a common statistic used to evaluate forecasting models. Lower RMSE indicates better performance. Therefore, based on the RMSE criterion, the “Seasonal_Naive” model has the lowest RMSE among the listed models, with a value of 173.5283.

According to the other parameters we have best models as below: ME (Mean Error): The ME for “ets_auto” is -84.37989. The model with the smallest ME is “ets_auto.”

MAE (Mean Absolute Error): The MAE for “arima” is 146.1166 . The model with the smallest MAE is “arima.”

MPE (Mean Percentage Error): The MPE for “arima” is1 114.41. The model with the smallest MPE is “arima.”

MAPE (Mean Absolute Percentage Error): The MAPE for “ets” is -7.33266 The model with the smallest MAPE is “ets.”

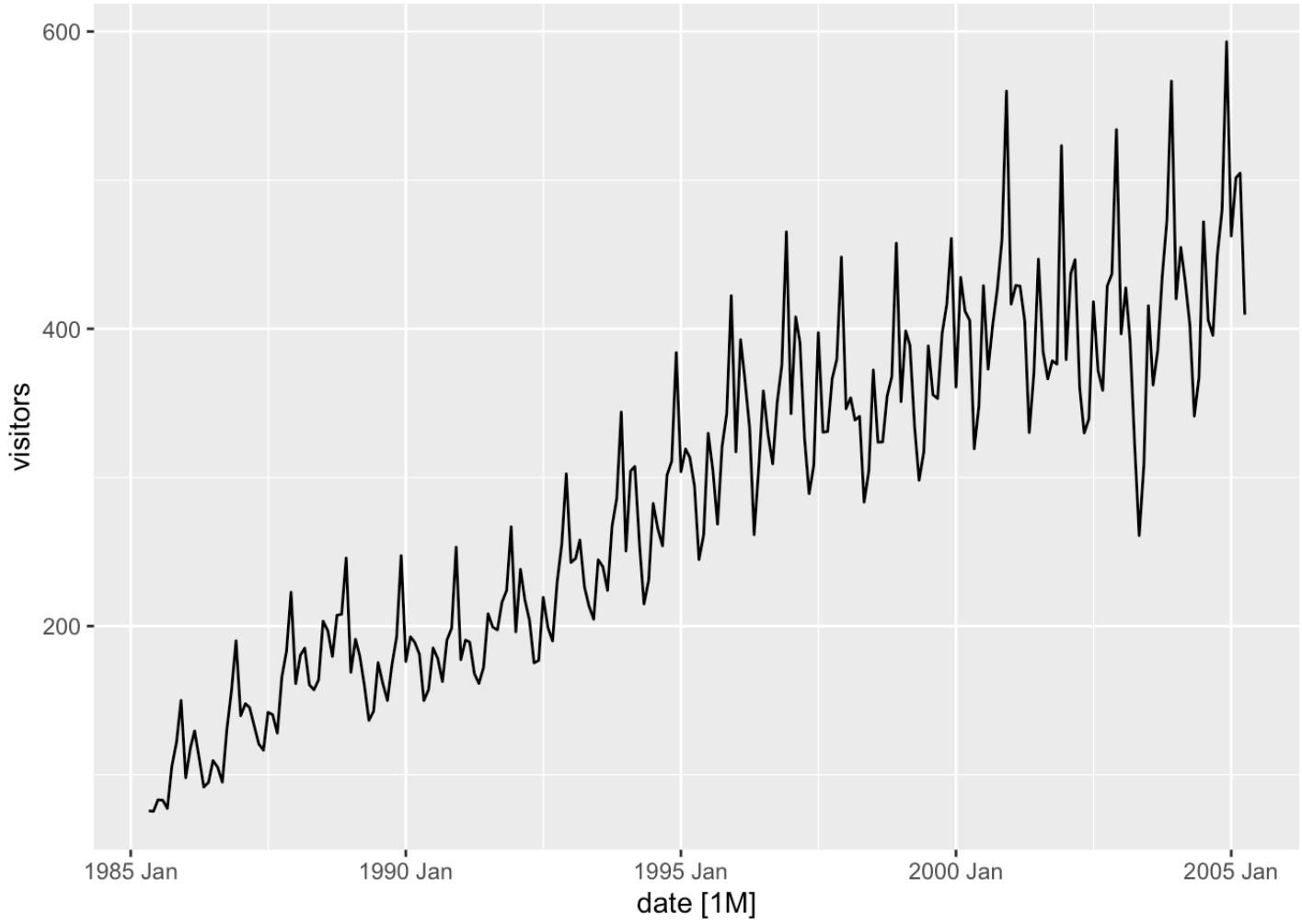
MASE (Mean Absolute Scaled Error): The MASE for “arima” is 9.29 The model with the smallest MASE is “arima.”

Given the many measures, no single model consistently outperforms the others in every criterion. I would say, arima model and ANN ets model is performing better for the above data in forecasting the sales of product A. If we have to chose one model “arima” appears to be a good option.

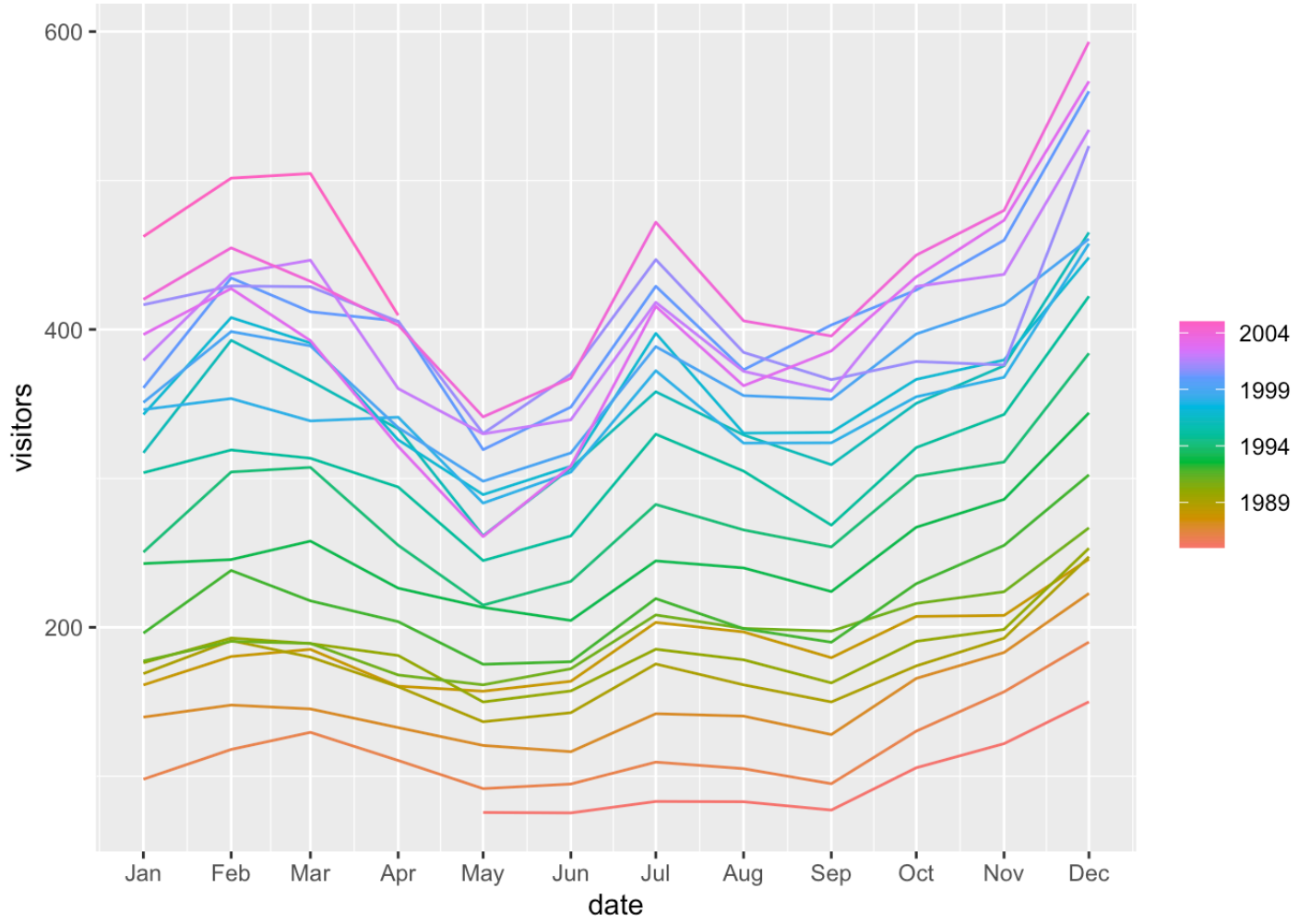
DATA visitors:

```
df <- readr::read_csv("visitors.csv",show_col_types = FALSE)
df = df%>%mutate(date = yearmonth(date))%>%
  tsibble(index = date)
```

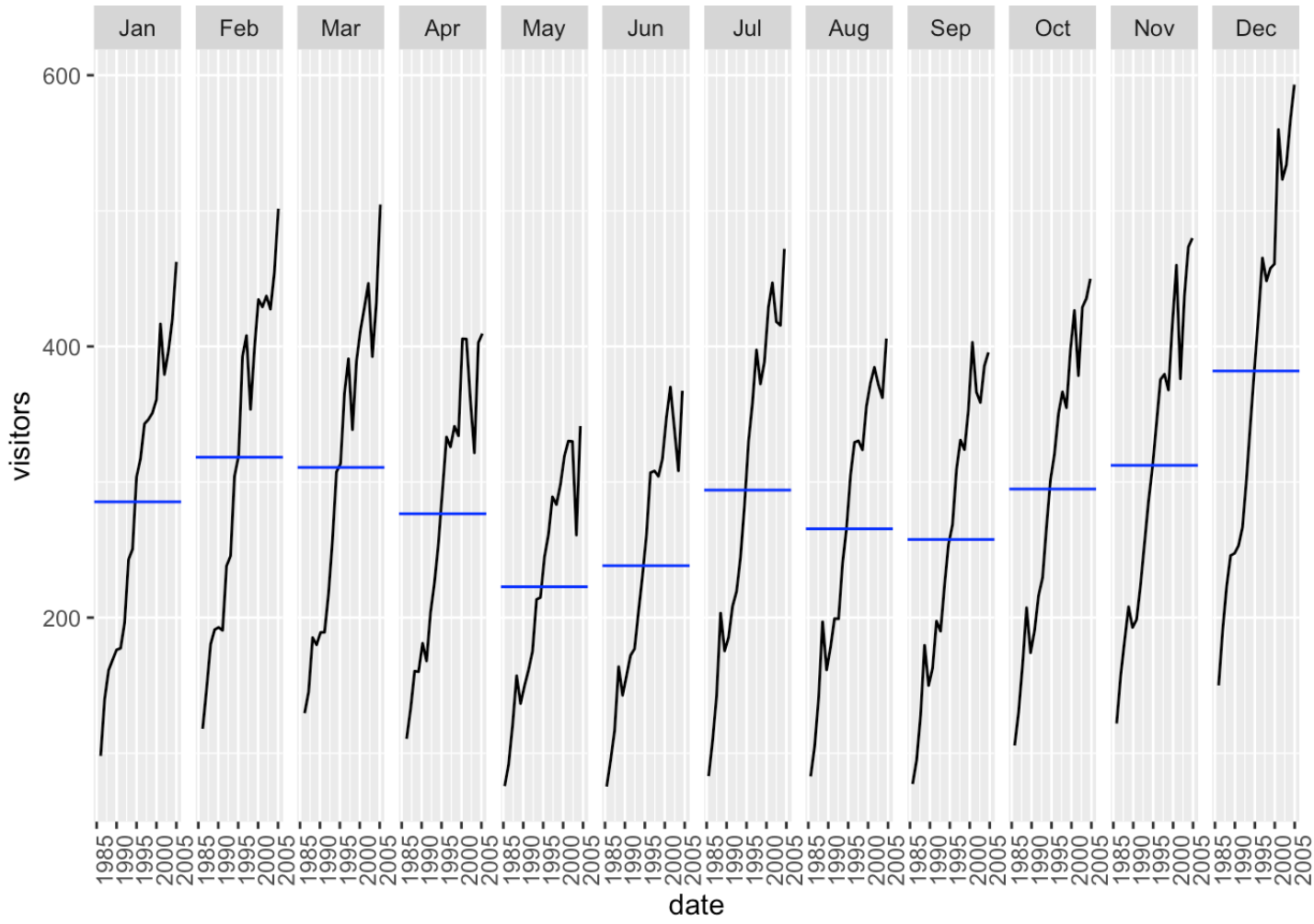
```
df %>% autoplot(visitors)
```



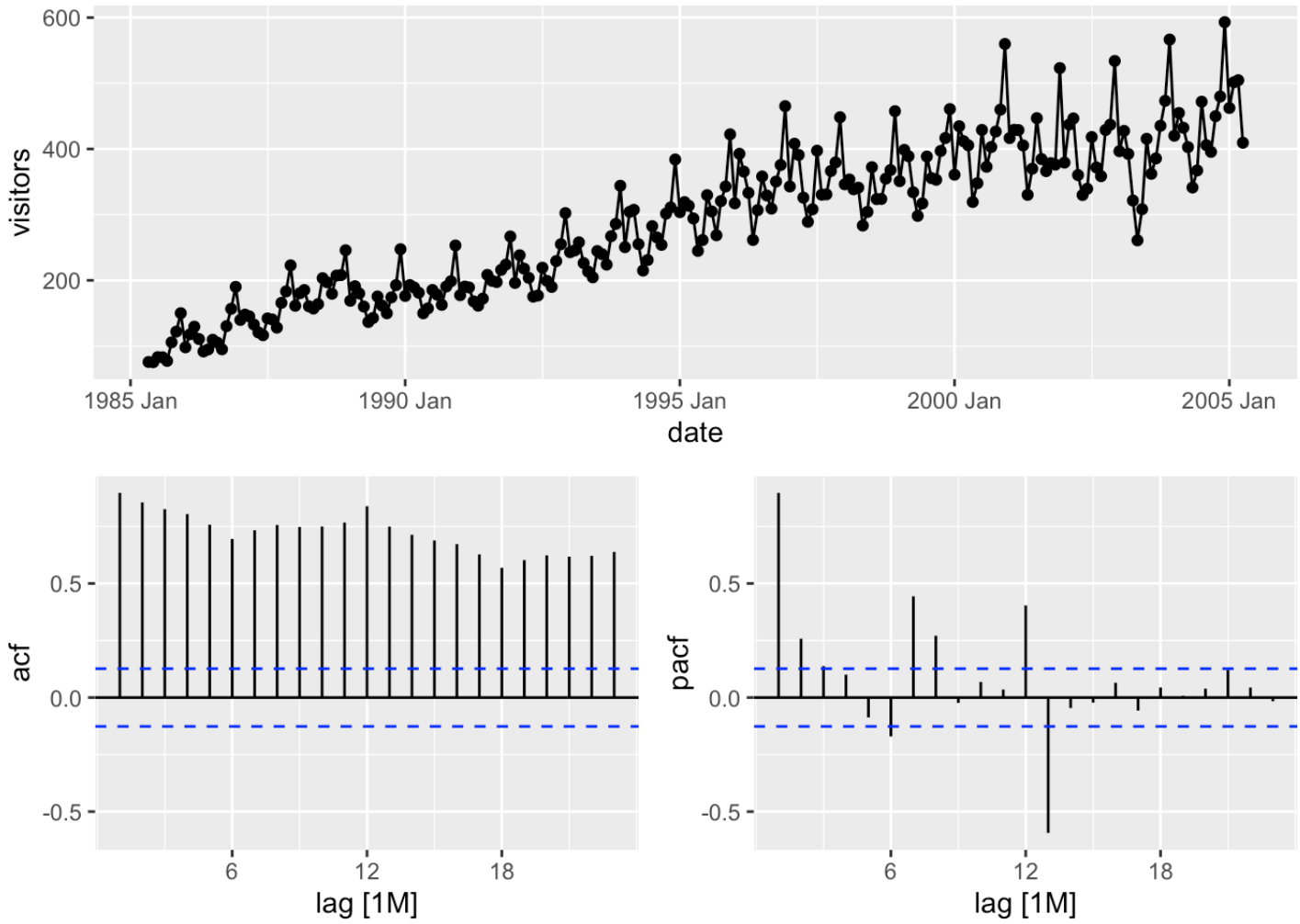

```
df %>% gg_season(visitors)
```




```
df %>%gg_subseries(visitors)
```




```
gg_tsdisplay(df, visitors, plot_type='partial')
```

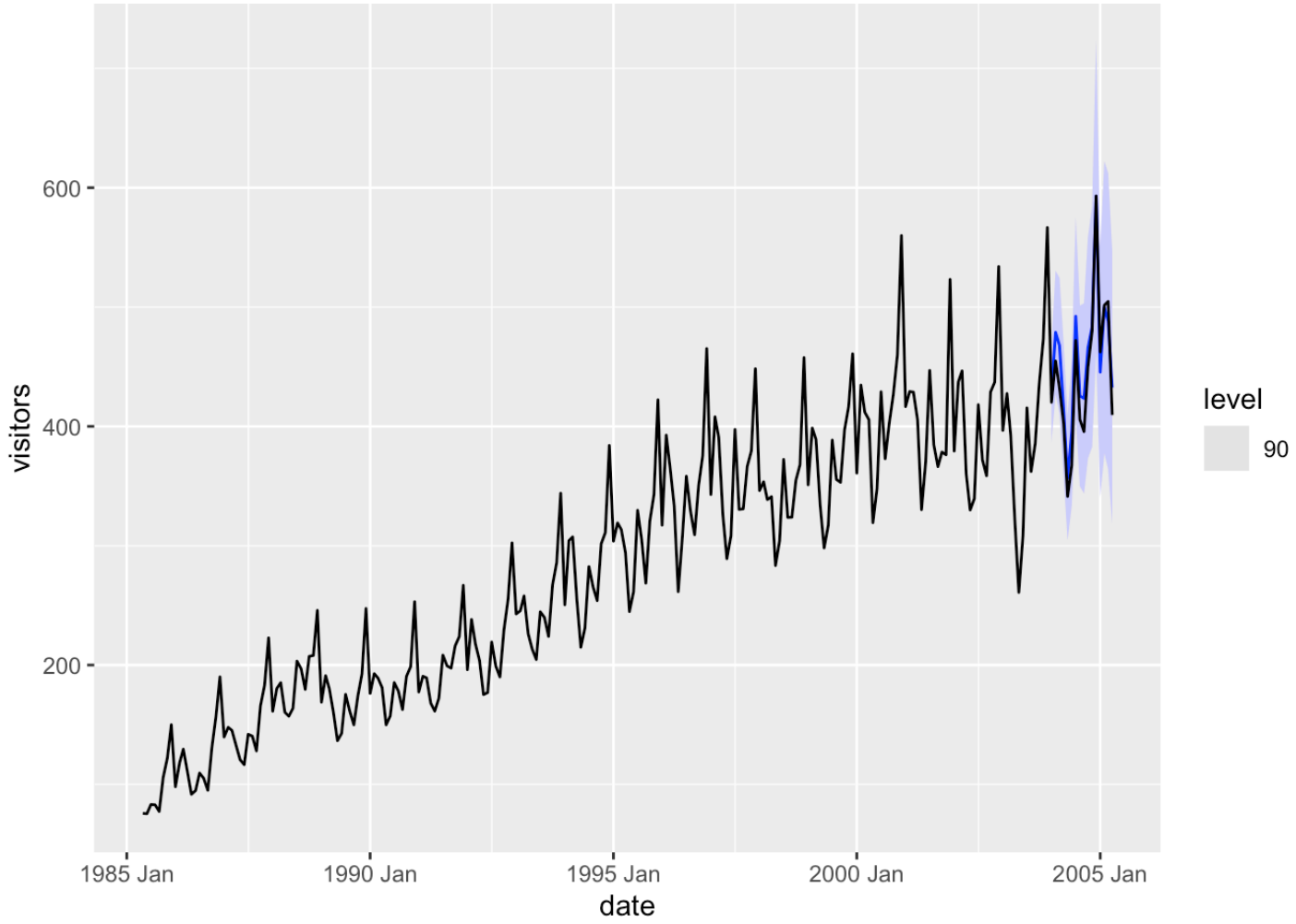


Over the whole time period, from May 1985 to April 2005, the pattern is shown as a progressive increase in the number of visits. This long-term increasing trend is constant, indicating a growing tendency in visitors. Seasonality behavior is also seen in the data as a recurring pattern of peaks and troughs at regular periods. which also confirmed by seasonality plot where, in December it has highest peak and even in july there is seasonal behavior found. Also, there might be an outlier present as 2003 we can see huge fall in visitors might be due to out-liers. *As same months exhibit relatively stable visitor numbers the variance in the data is stable.* Also, the lag in the data is 12 months as every year, the same seasonal pattern appears. * We can observe cyclic behavior as well as patterns or variations that reoccur over longer time periods than regular seasonality.

```
train <- df %>% filter(year(date) < 2004)
test <- df %>% filter(year(date) >= 2004 & year(date) <= 2005 )
```

Holt-Winters' multiplicative

```
fit <- train %>%model(multiplicative = ETS(visitors ~error("M") + trend("A") + seaso
n("M")))
fc <- fit %>% forecast(h= 16)
fc %>% autoplot(df, level = 90)
```



WHY

“multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series.” Seasonal variation rises over time.We can see vary in magnitude(visitors) over time. If we see in year 1985 to 2005 the seasonal nature has been increasing. Its clearly seen that variance is not constant and its varying over time.

HOW IT WILL HELP?

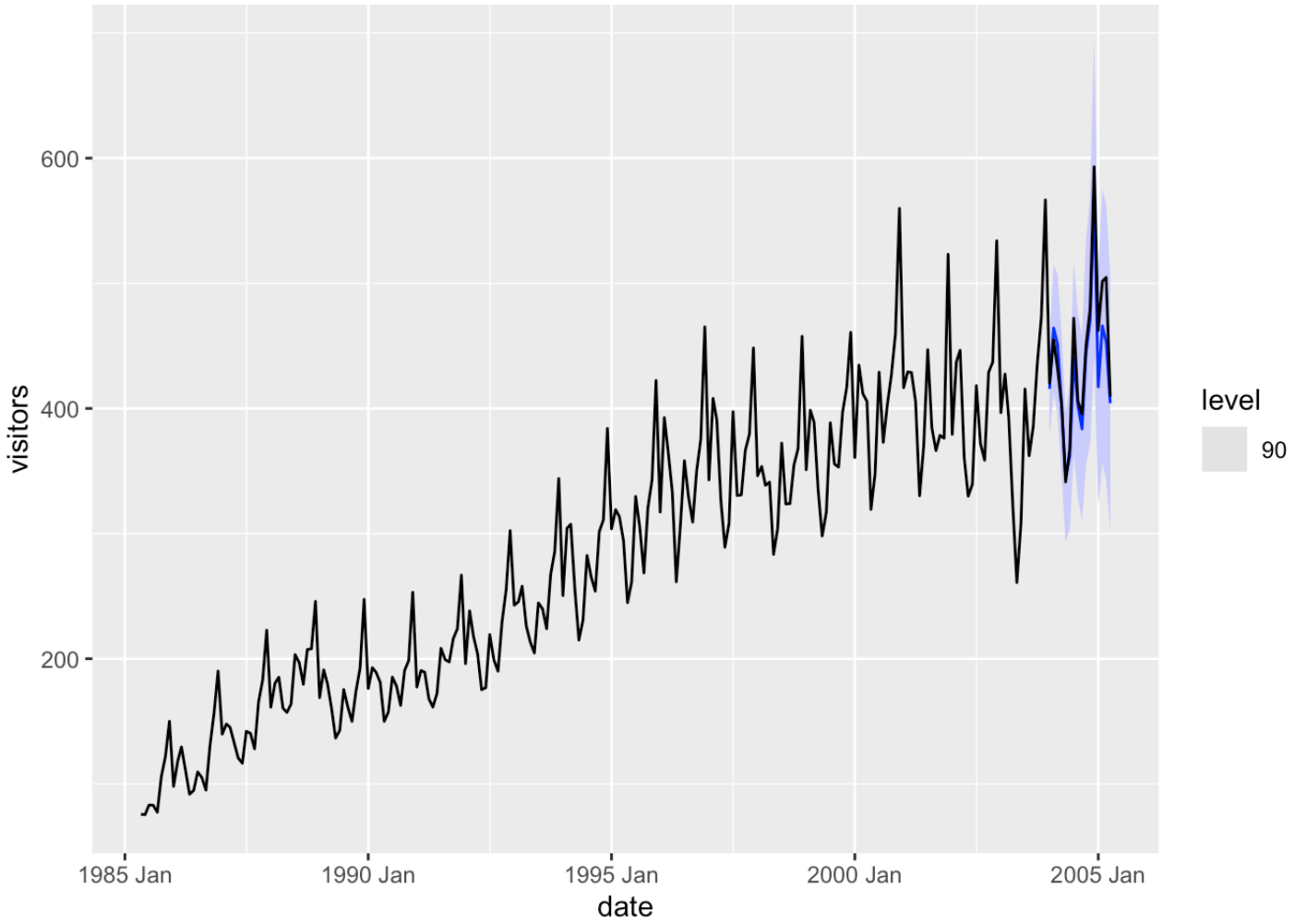
Using multiplicative seasonality ensures that the model captures the relationship correctly, making projections more relevant and accurate. Furthermore, it corresponds to the natural growth and patterns identified in the data, resulting in a more acceptable and interpretable model.

I. an ETS model;

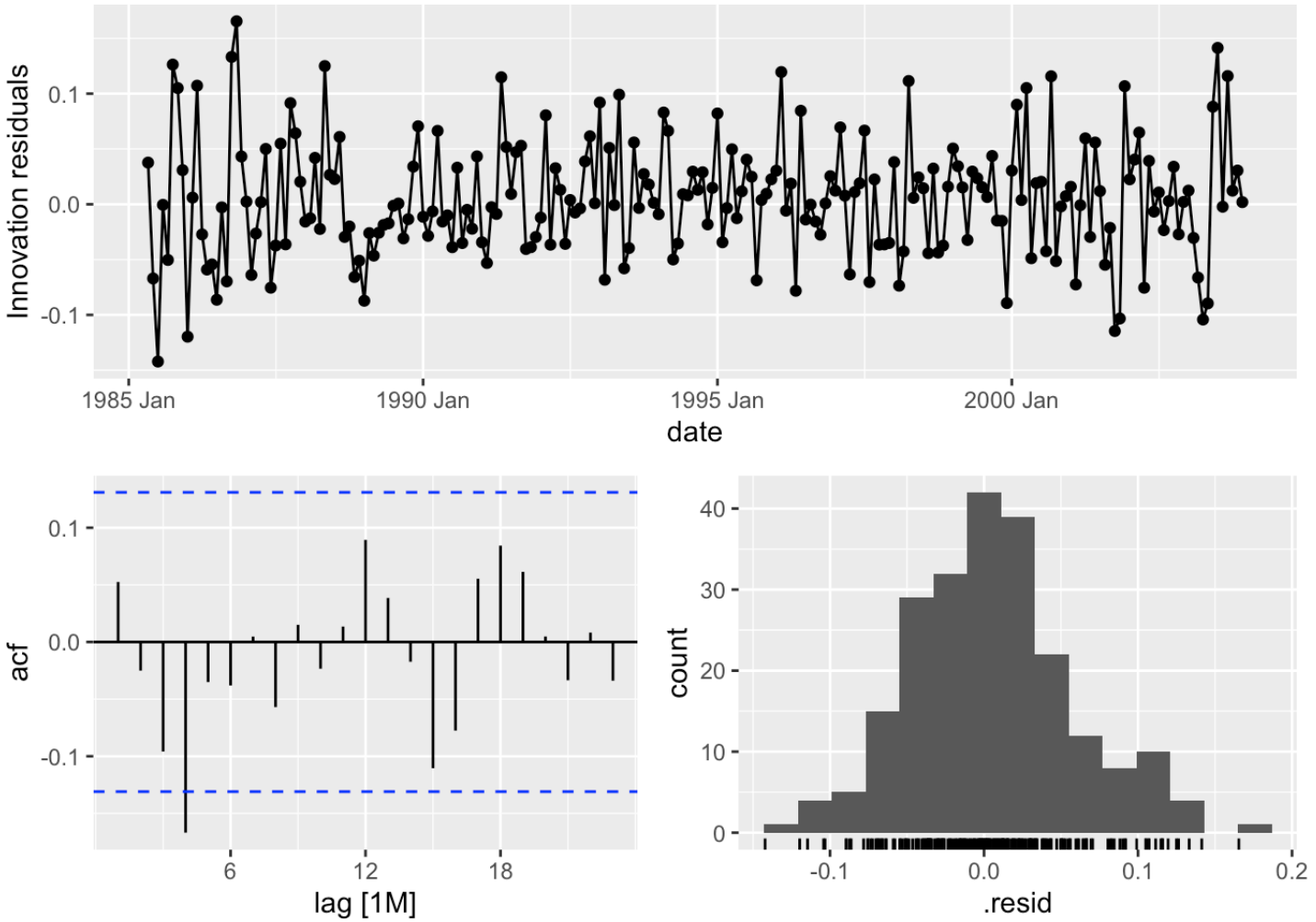
```
fit <- train %>%
  model(
    ets = ETS((visitors))
  )

fit <- fit %>% dplyr::select(ets)
fc <- fit %>% forecast(h = 16)
accuracy(fc,df)
```

```
## # A tibble: 1 × 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ets    Test   12.6  22.9  16.6  2.58  3.52 0.637 0.733 0.518
```

```
fc %>% autoplot(df,level = 90)
```



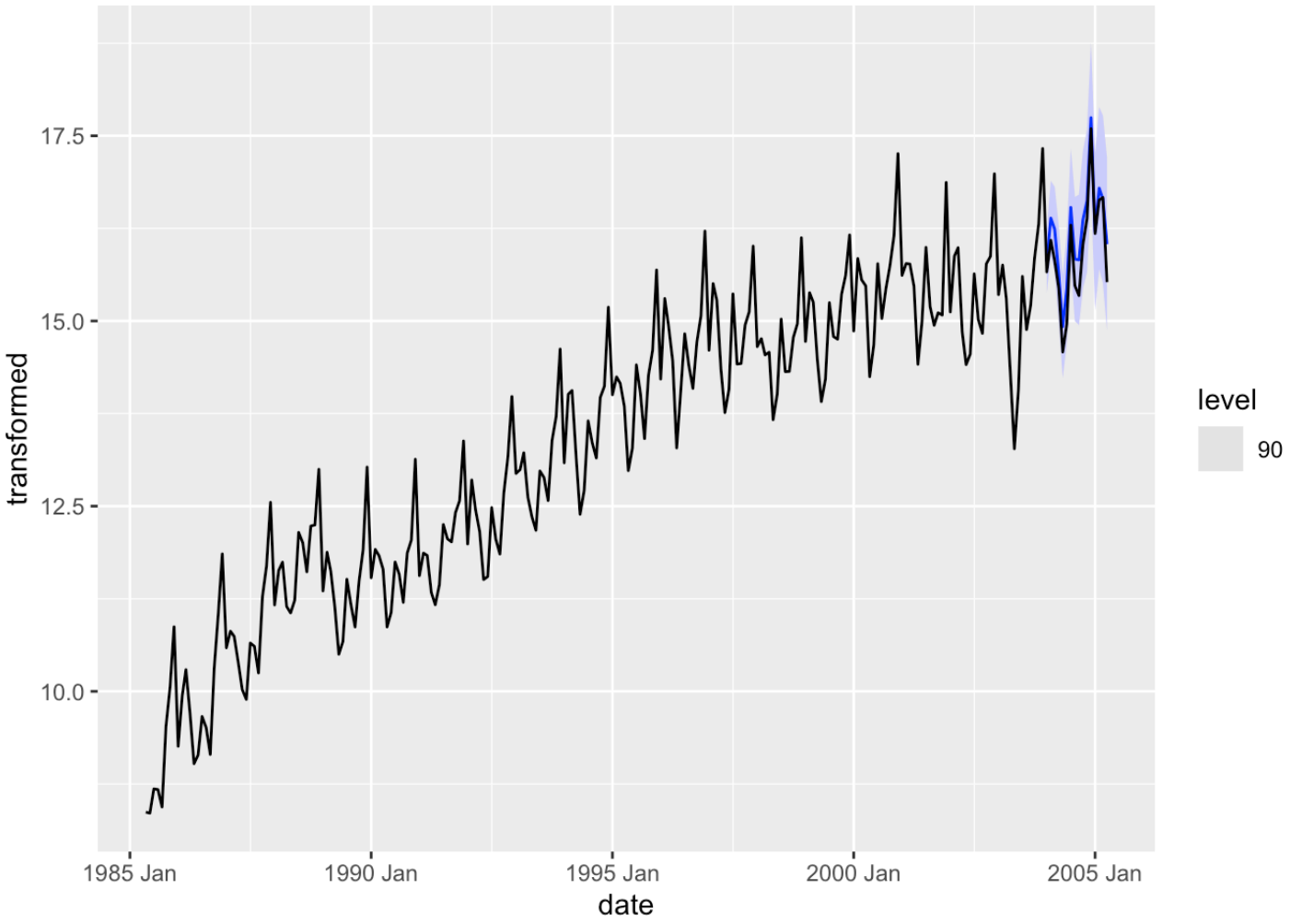
```
gg_tsresiduals(fit)
```



II.

an additive ETS model applied to a Box-Cox transformed series;

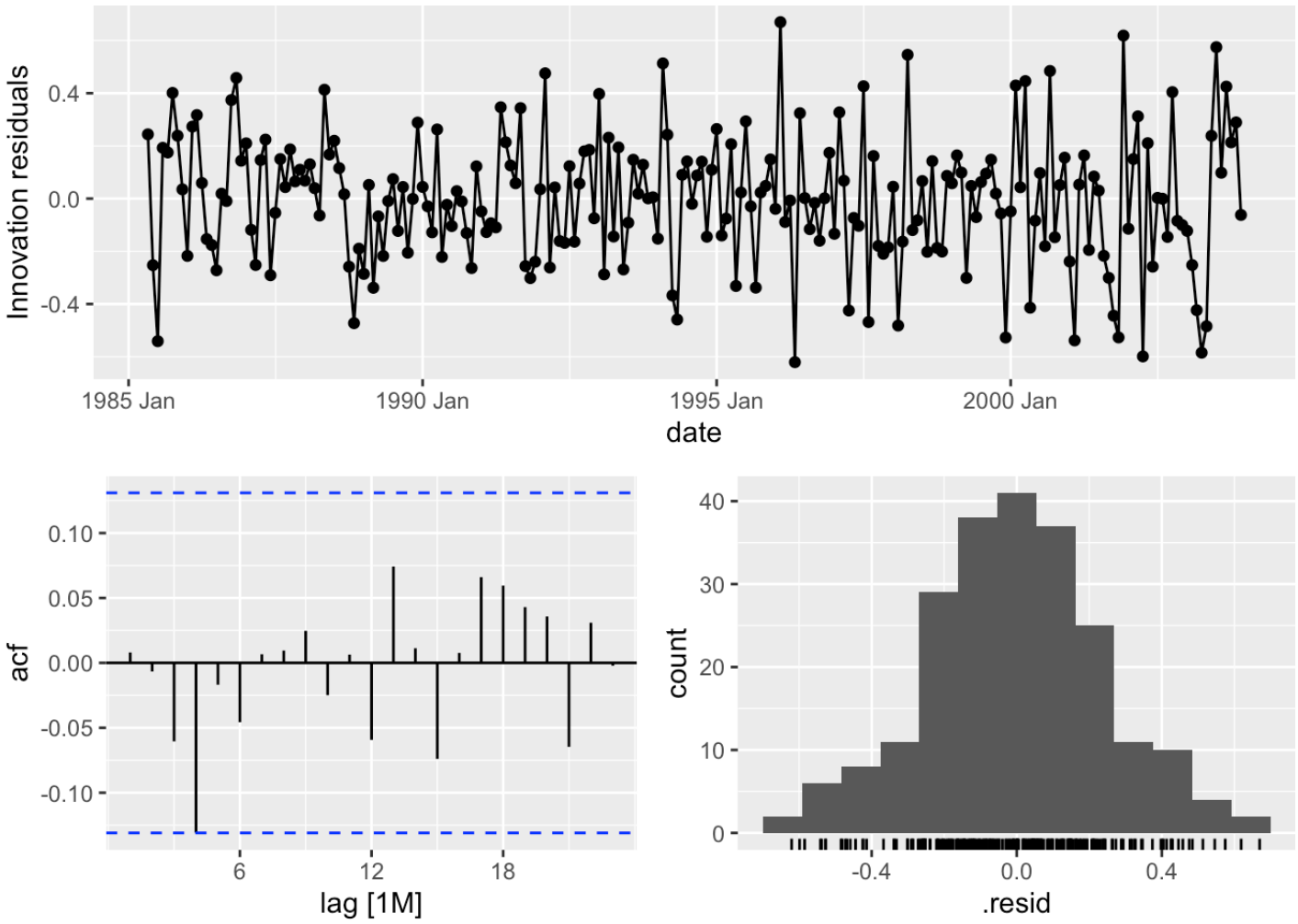
```
lam <- df %>% features(visitors, features = guerrero)
transformed <- BoxCox(df$visitors, lam$lambda_guerrero)
df$transformed = transformed
df1 <- df[, -2]

train <- df1 %>% filter(year(date) < 2004)
test <- df1 %>% filter(year(date) >= 2004 & year(date) <= 2005 )

fit <- train %>% model(additive = ETS(transformed ~ error("A") + trend("A") + seaso
n("A")))
fc <- fit %>% forecast(h = 16)
fc %>% autoplot(df1,level = 90)
```




```
accuracy(fc,df)
```

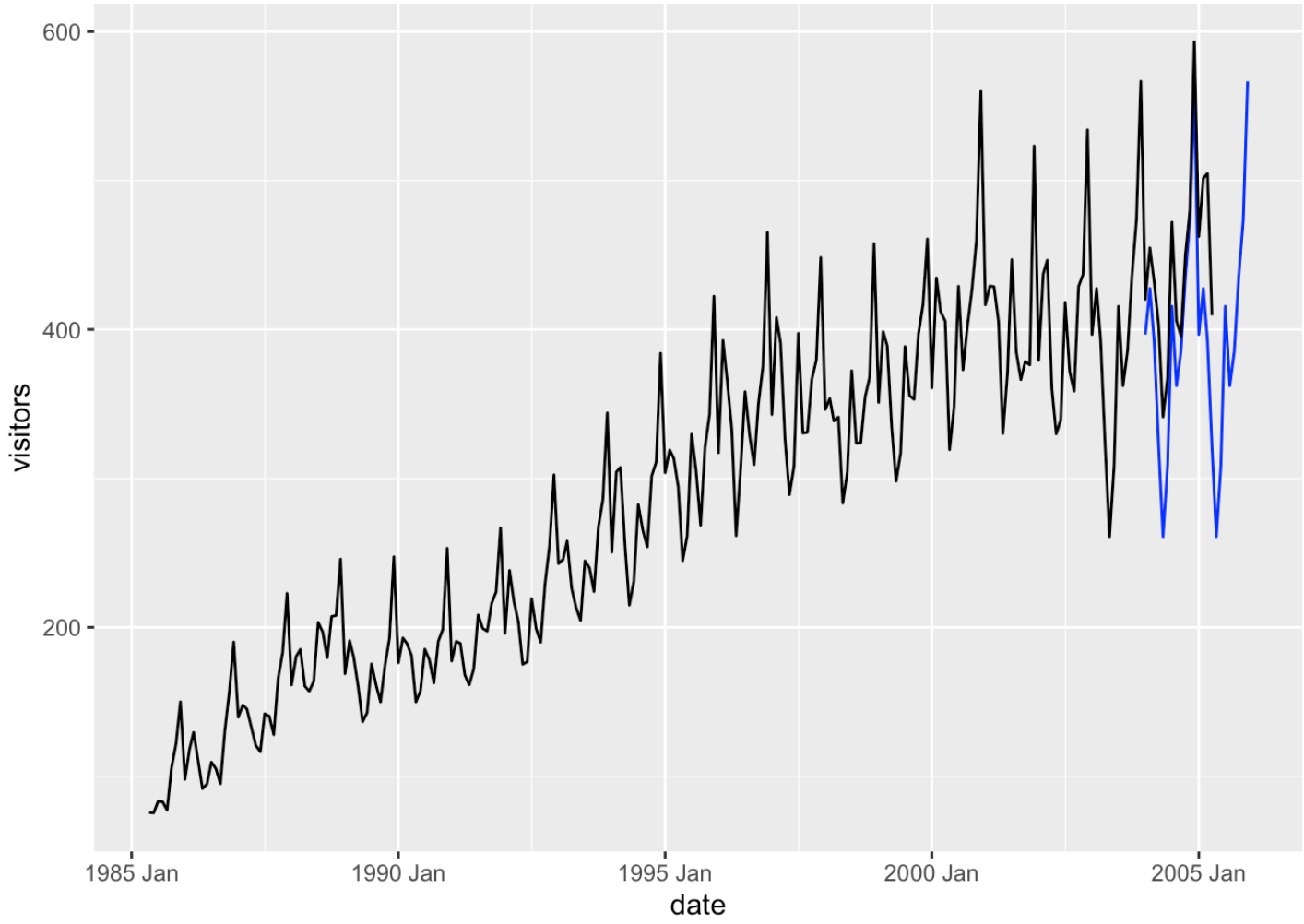
```
## # A tibble: 1 × 10
##   .model   .type     ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>    <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 additive Test  -0.273 0.314 0.276 -1.75  1.76 0.567 0.538 0.108
```

```
gg_tsresiduals(fit)
```



III. a seasonal naïve method;

```
train <- df %>% filter(year(date) < 2004)
test <- df %>% filter(year(date) >= 2004 & year(date) <= 2005 )

fit <- train %>%model(Seasonal_Naive = SNAIVE(visitors))

fc <- fit %>% forecast(h = 24)
fc %>% autoplot(df,level = NULL)
```




```
accuracy(fc,df)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be t
reated as missing.
## 8 observations are missing between 2005 May and 2005 Dec
```

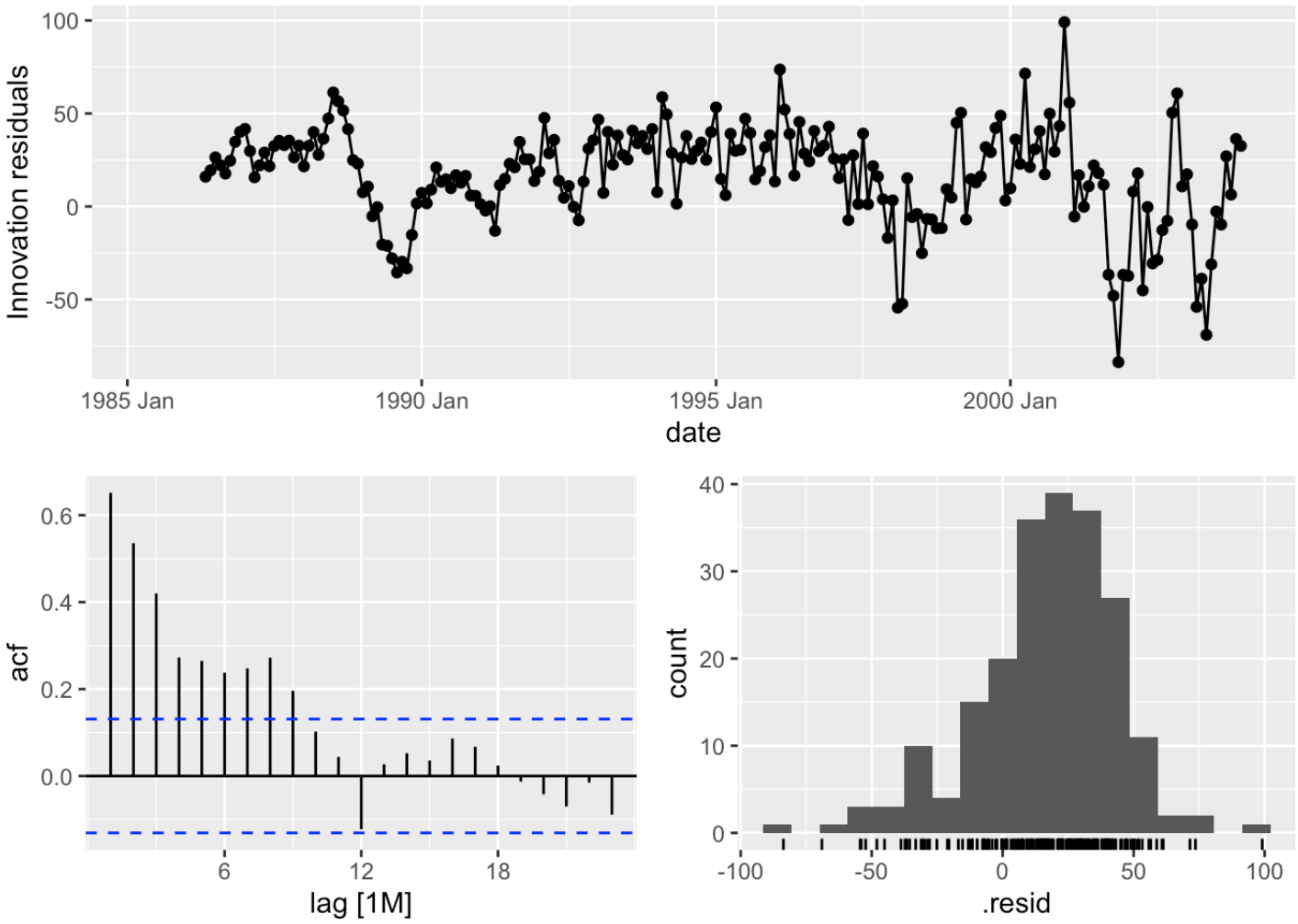
```
## # A tibble: 1 × 10
##   .model         .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>          <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Seasonal_Naive Test   50.6  59.0  50.6  11.7  11.7  1.94  1.89 0.665
```

```
gg_tsresiduals(fit)
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```

```
## Warning: Removed 12 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 12 rows containing non-finite values (`stat_bin()`).
```



ABOVE PLOTTED THE RESIDUAL PLOT ETS Additive Model:

ME: -0.2730922 RMSE: 0.3143414 MAE: 0.275704 MAPE: 1.761355 MASE: 0.56687

The Additive ETS Model has lower values for the above five parameters, indicating that it performs well in comparison to other models. However, when compared to the ETS model and the seasonal naive technique, the ETS model has lower values across all measures, indicating that it is somewhat more favorable based on the assessment criteria.

Also, if I can check AIC value of ETS model 2411.094 and for additive ets model of transformed data it is -61.94 Which clearly proves that model 2 - an additive ETS model applied to a Box-Cox transformed is performing well.

DATA electricty:

```
df <- readr::read_csv("usmelec.csv",show_col_types = FALSE)
df = df%>%mutate(index = yearmonth(index))%>%
  tsibble(index = index)
df
```

```
## # A tsibble: 486 x 2 [1M]
##       index value
##       <mth> <dbl>
##  1 1973 Jan  160.
##  2 1973 Feb  144.
##  3 1973 Mar  148.
##  4 1973 Apr  140.
##  5 1973 May  147.
##  6 1973 Jun  161.
##  7 1973 Jul  174.
##  8 1973 Aug  177.
##  9 1973 Sep  157.
## 10 1973 Oct  154.
## # ℹ 476 more rows
```

```
dcmp <- df %>%model(classical_decomposition(value, type='m'))
trend <- dcmp$trend
```

```
## Warning: Unknown or uninitialised column: `trend`.
```

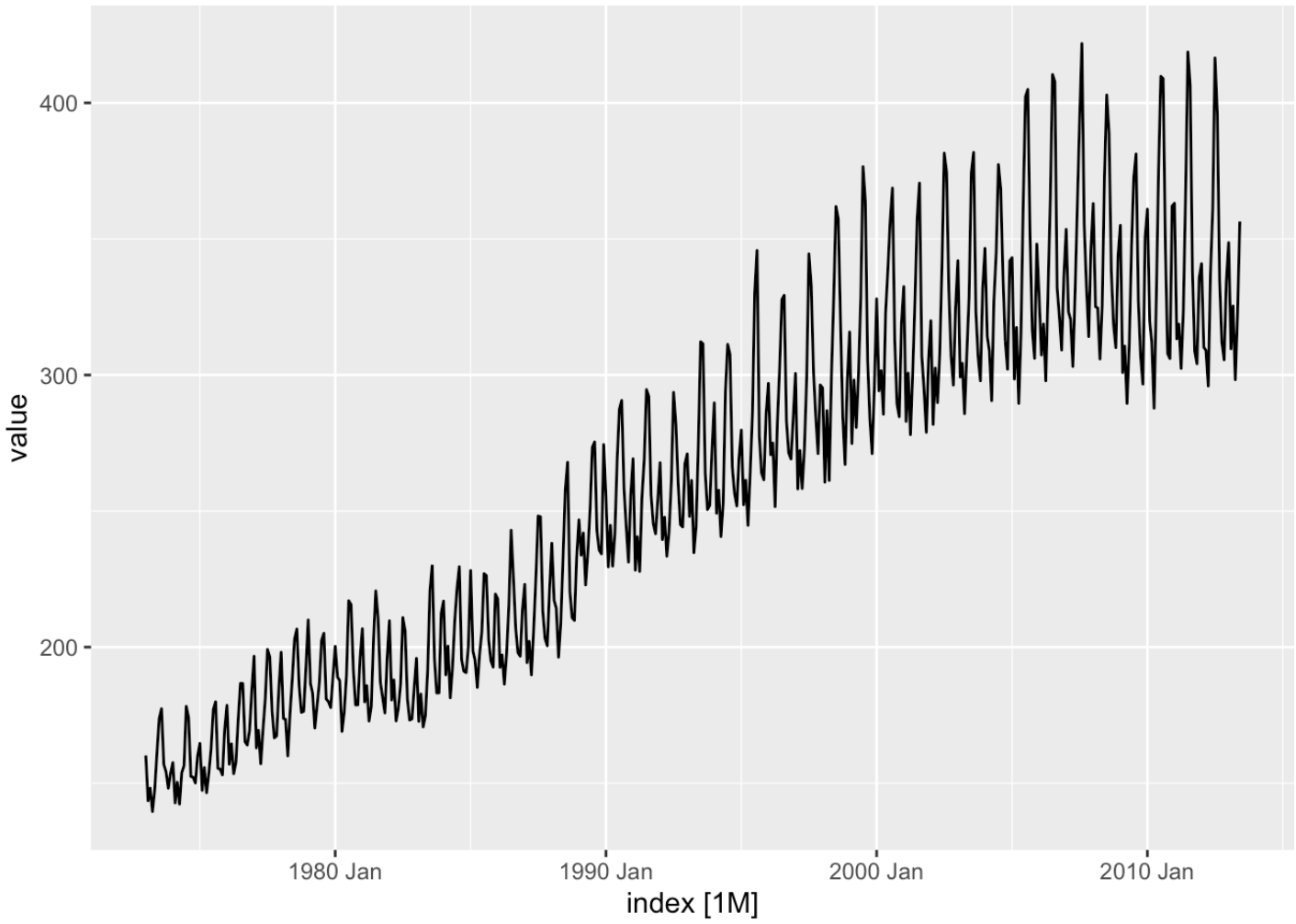
```
seasonal <- dcmp$seasonal
```

```
## Warning: Unknown or uninitialised column: `seasonal`.
```

```
rand <- dcmp$random
```

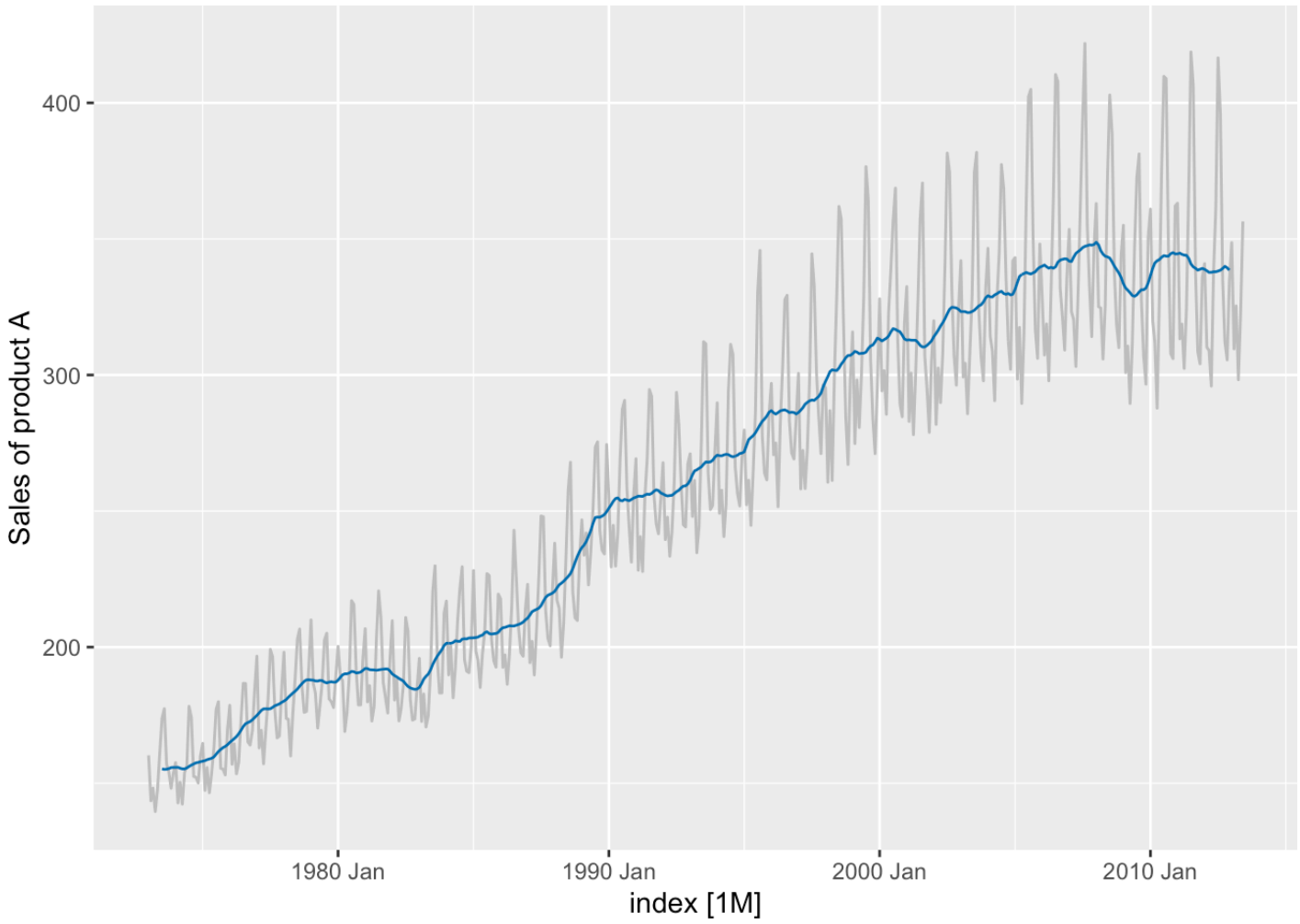
```
## Warning: Unknown or uninitialised column: `random`.
```

```
df %>% autoplot(value)
```




```
df %>% autoplot(value, color='gray') + autolayer(components(dcmp), trend, color='#007
2B2') +labs(y="Sales of product A")
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```

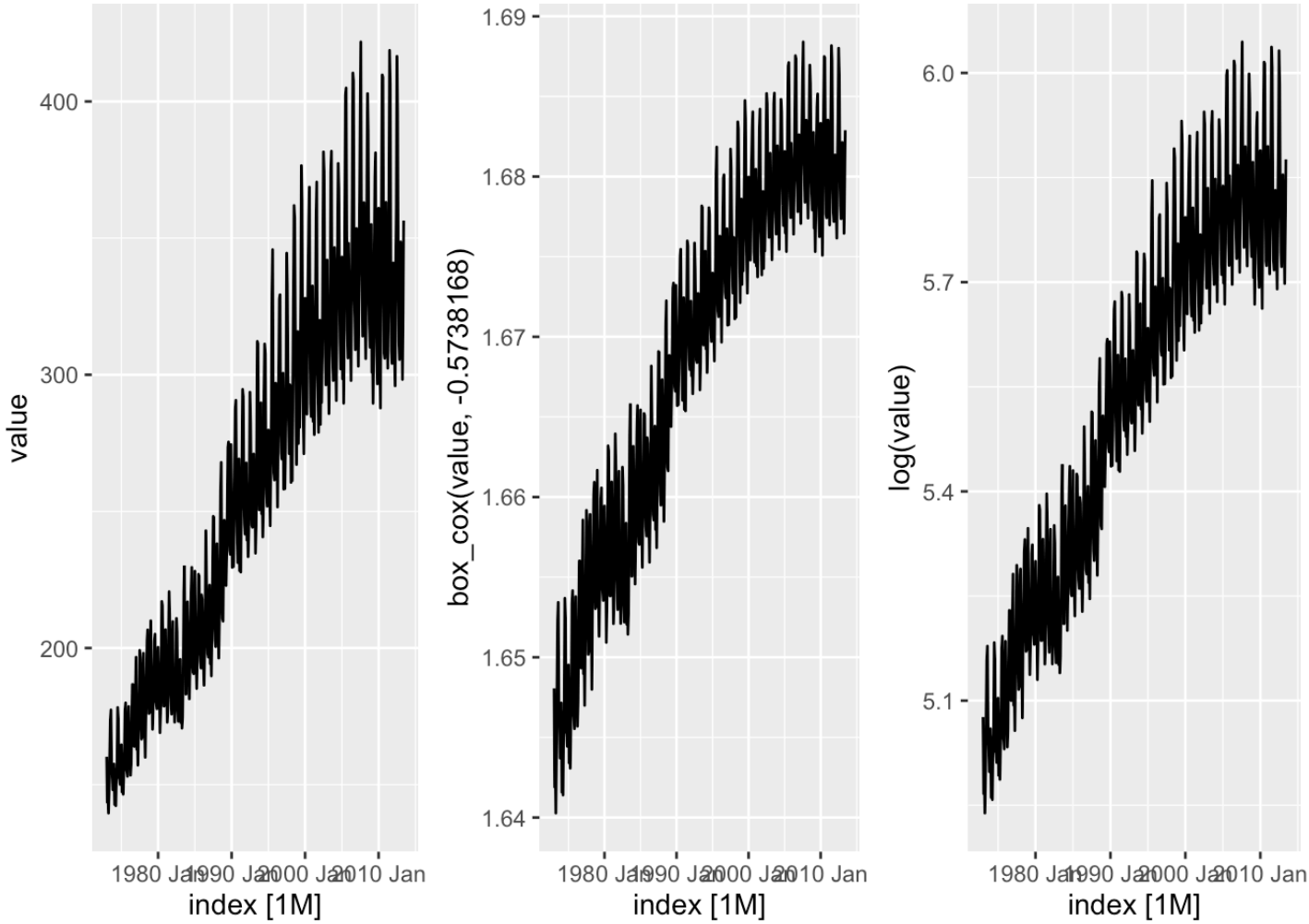


Analysing the 12-month moving average from a multiplicative decomposition entails comparing it to the original time series graphically to discover patterns, cycles, and outliers. Understanding underlying patterns is aided by assessing its efficacy in smoothing short-term variations and capturing longer-term trends. The research emphasises the importance of stationarity, seasonality representation, and forecasting utility. To assess accuracy, quantitative measurements such as MSE or RMSE might be utilised. The 12-month window size is subjective and may need testing.

```
df %>% features(value, features = guerrero)
```

```
## # A tibble: 1 × 1
##   lambda_guerrero
##             <dbl>
## 1          -0.574
```

```
plot_grid(
  autoplot(df,value),
  autoplot(df,box_cox(value,-0.5738168)),
  autoplot(df,log(value)),
  ncol=3)
```

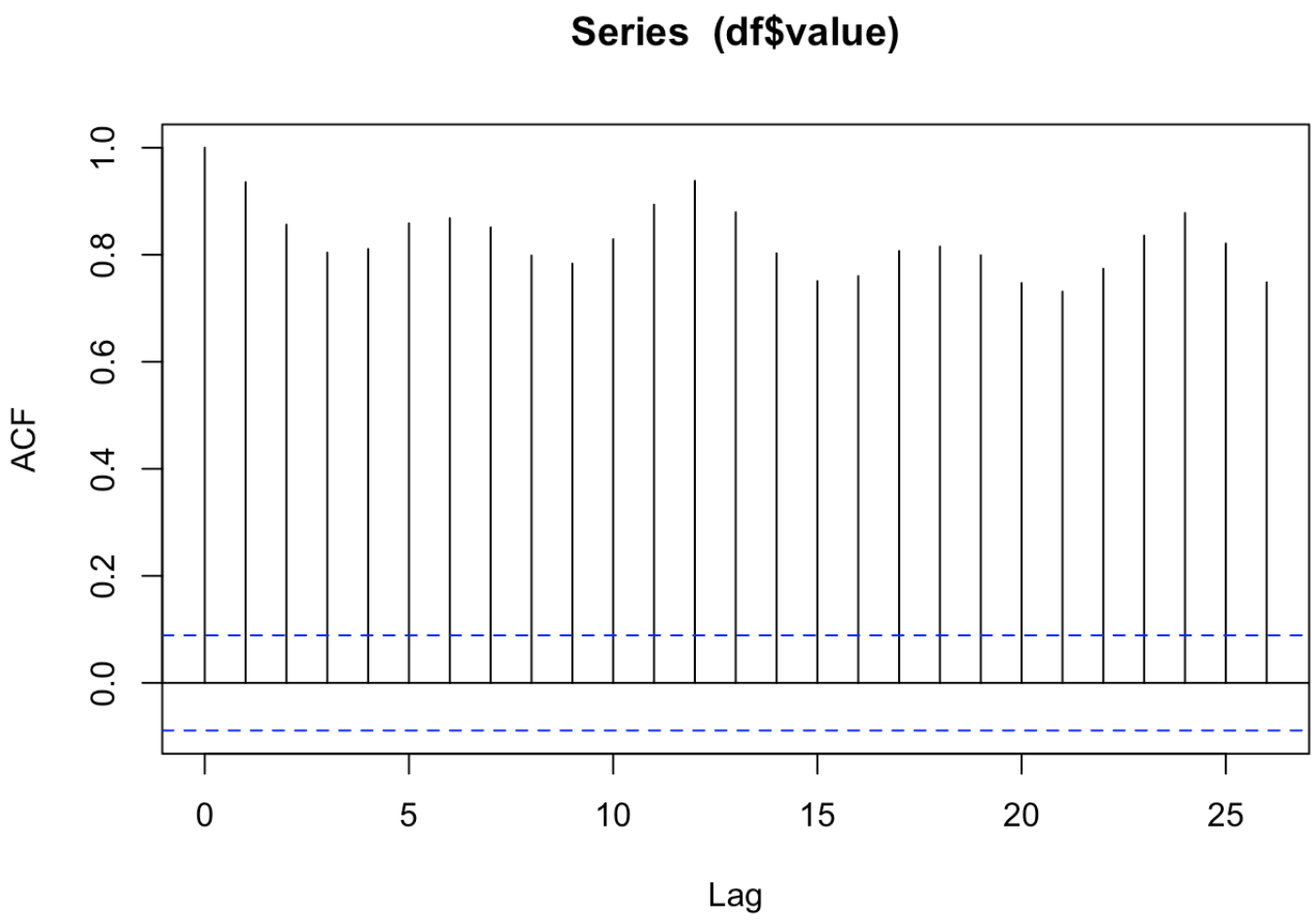


Given the seasonal nature of energy generation, the existence of peaks in the middle of the summer and middle of the winter shows seasonality that might benefit from transformation approaches. Logarithmic transformation, for example, can be useful in stabilising variance and dealing with right-skewed distributions found in energy consumption data. We need the transforming as the data as the mean, variance and autocorrelation structure is changing over time where the data is not stationary

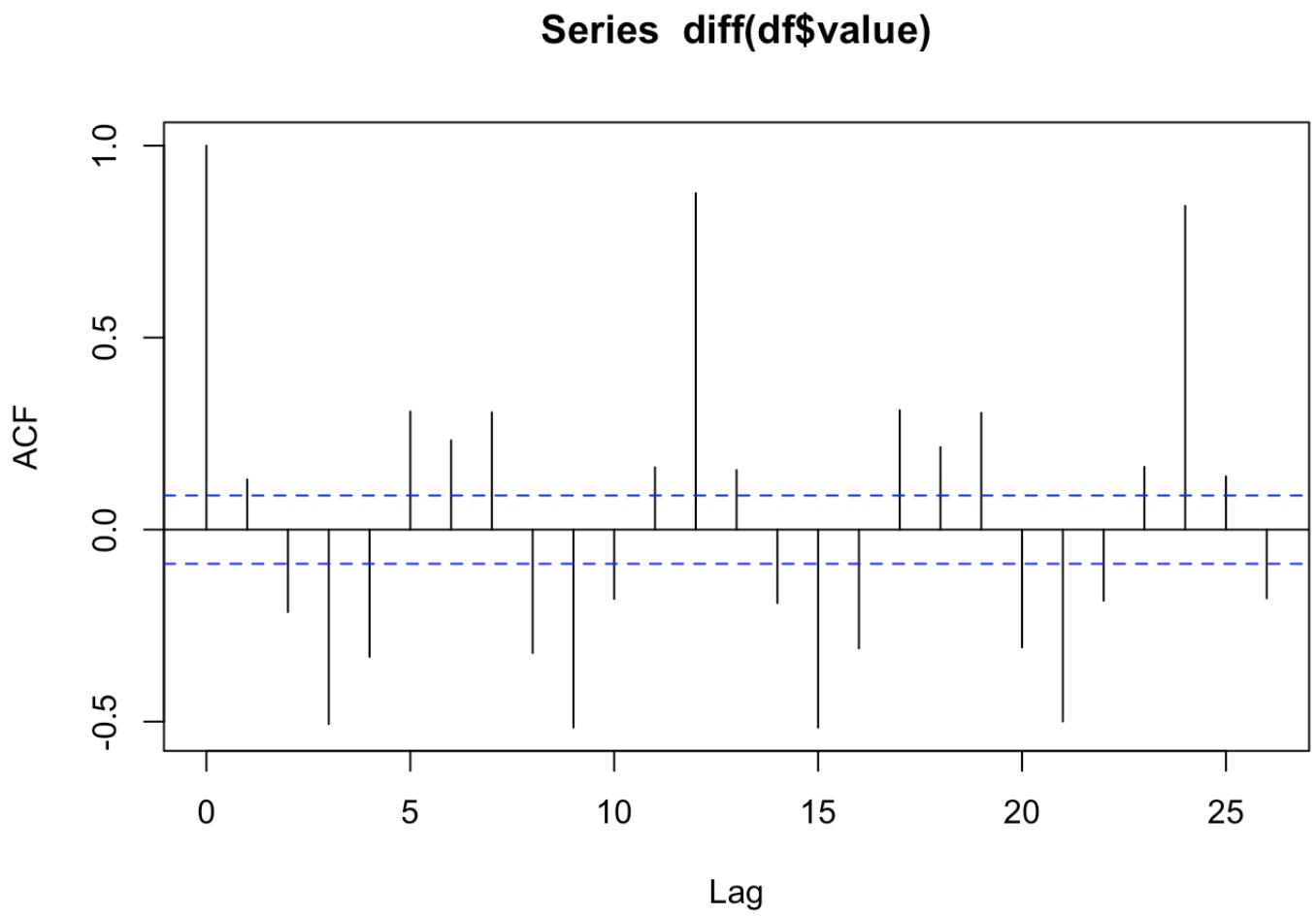
From plot, we can confirm box-cox and log bot the tranformation perform better at maintaing the variance and mean constant.

Data is not stationary. As total net generation of electricity is dependent on the time. We can see predictable patterns in the long-term. Time plots is also confirming that its changing over time. Also, the mean, variance and autocorrelation structure is changing over time.

```
acf((df$value))
```




```
acf(diff(df$value))
```




```
Box.test(df$value, lag=10, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  df$value
## X-squared = 3487.1, df = 10, p-value < 2.2e-16
```

The ACF plot may also be used to identify non-stationary time series. The ACF of a stationary time series falls to zero relatively rapidly, but the ACF of non-stationary data declines slowly. We don't see any decline here. As a result, it confirms that the data is non-stationary.

Test Statistic (X-squared): 3487.1 Degrees of Freedom:10 p-value:2.2e-16

All three results indicate that the auto-correlations are not zero.And there is a pattern in the time series data that cannot be explained. There is many autocorrelation lying just outside the 95% limits.

Finding the approriate differntiate value

```
ndiffs(df$value)
```

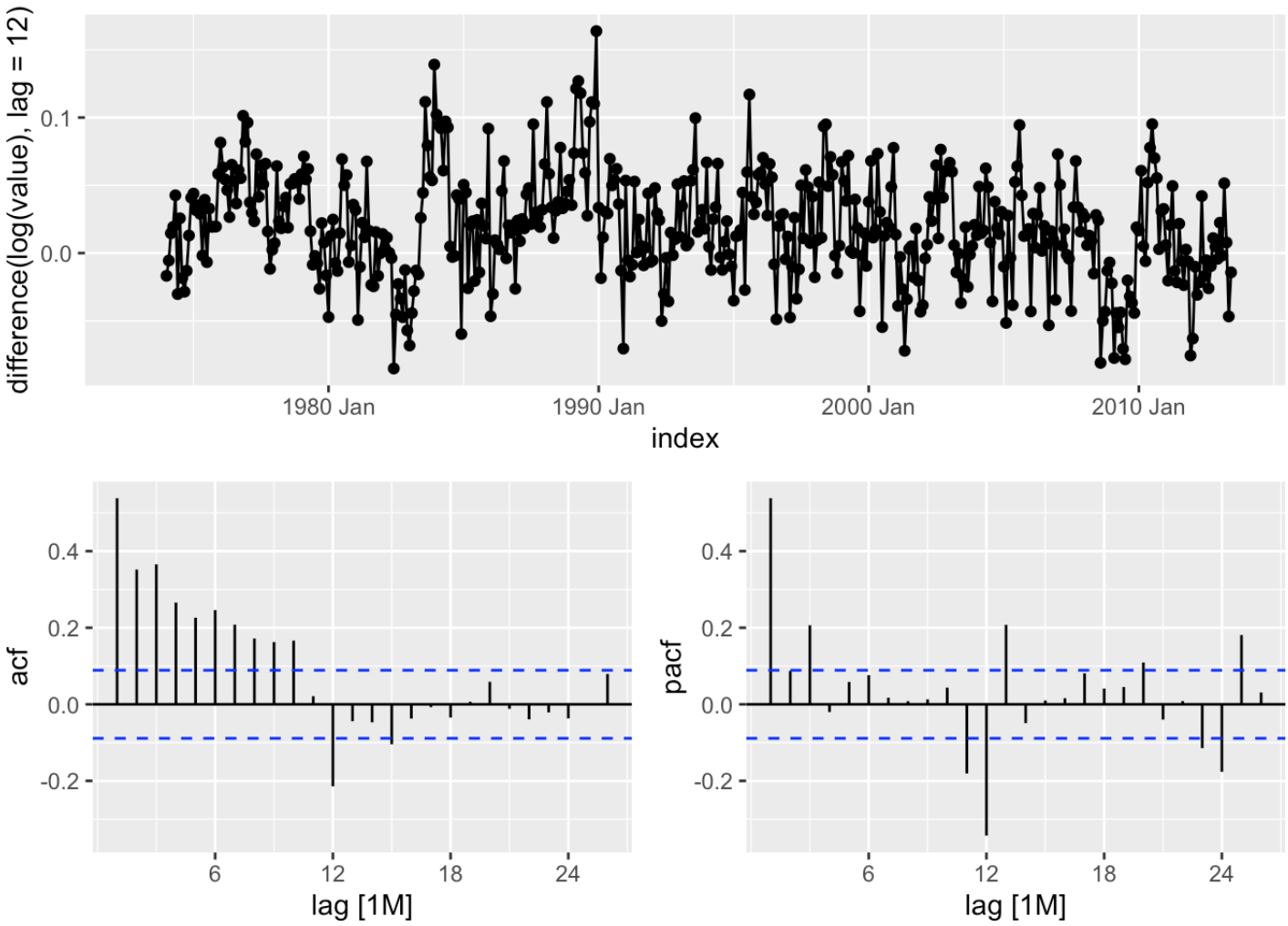
```
## [1] 1
```

According to ndiffs(), the first differentiation should be sufficient to remove stationary.

```
lam <- df %>% features(value, features = guerrero)
transformed <- BoxCox(df$value, lam$lambda_guerrero)
df$diff_transformed_boxcox = difference(transformed, 12)
df$diff_log = difference(log(df$value), 12)

gg_tsdisplay(df, difference(log(value), lag=12), plot_type='partial')
```

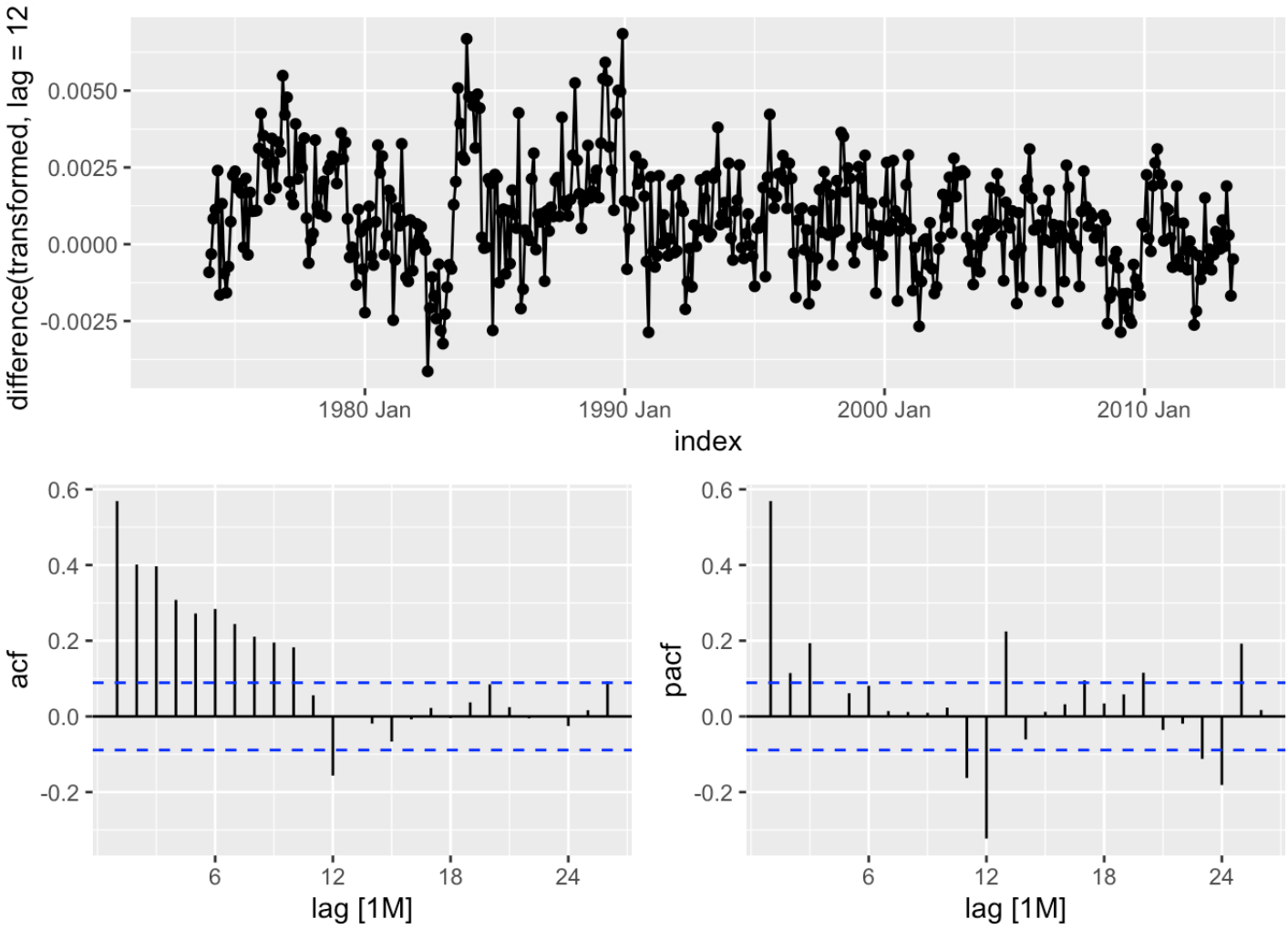
```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
```

```
## Warning: Removed 12 rows containing missing values (`geom_point()`).
```




```
gg_tsdisplay(df, difference(transformed, lag=12), plot_type='partial')
```

```
## Warning: Removed 12 rows containing missing values (`geom_line()`).
## Removed 12 rows containing missing values (`geom_point()`).
```



The ACF and PACF graphs of the differences data reveal the following patterns, indicating that the data may follow an ARIMA(p, d, 0) model.

```
arima1 = Arima(df$diff_transformed_boxcox, order = c(6, 1, 0), seasonal = c(0,
    1, 1))
arima2 = Arima(df$diff_transformed_boxcox, order = c(0, 1, 2), seasonal = c(3,
    1, 0))
arima3 = Arima(df$diff_transformed_boxcox, order = c(2, 1, 0), seasonal = c(1,
    1, 0))
arima1
```

```
## Series: df$diff_transformed_boxcox
## ARIMA(6,1,0)
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5      ar6
##       -0.4701  -0.4257  -0.2154  -0.2278  -0.1769  -0.0665
## s.e.   0.0459   0.0502   0.0529   0.0529   0.0503   0.0463
##
## sigma^2 = 1.993e-06:  log likelihood = 2435.84
## AIC=-4857.68   AICc=-4857.44   BIC=-4828.57
```

```
arima2
```

```
## Series: df$diff_transformed_boxcox
## ARIMA(0,1,2)
##
## Coefficients:
##           ma1      ma2
##       -0.4753  -0.2275
## s.e.   0.0466   0.0520
##
## sigma^2 = 1.986e-06:  log likelihood = 2434.67
## AIC=-4863.35   AICc=-4863.3   BIC=-4850.87
```

```
arima3
```

```
## Series: df$diff_transformed_boxcox
## ARIMA(2,1,0)
##
## Coefficients:
##           ar1      ar2
##       -0.4021  -0.3097
## s.e.   0.0437   0.0437
##
## sigma^2 = 2.082e-06:  log likelihood = 2423.59
## AIC=-4841.18   AICc=-4841.13   BIC=-4828.71
```

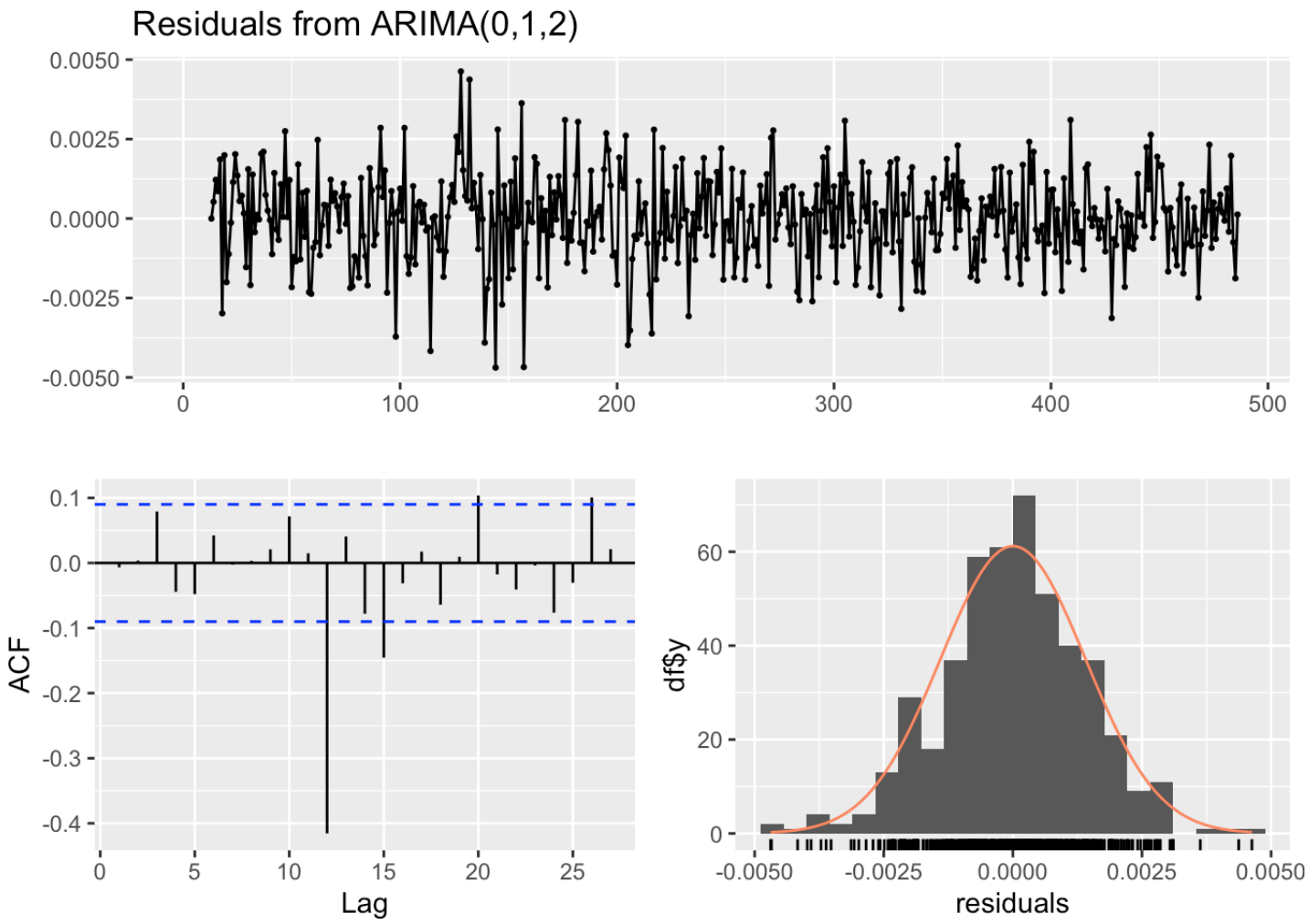
explanation

Model 1 - AIC=-4857.68 Model 2 - AIC=-4863.35 Model 3 - AIC=-4841.18

Model 2, has lower AIC VALUE of AIC=-4863.35 compare to other 2 models tested.

ARIMA(0,1,2)

```
residuals_df <- data.frame(index = df$index, residuals = residuals(arima2))
checkresiduals(arima2)
```

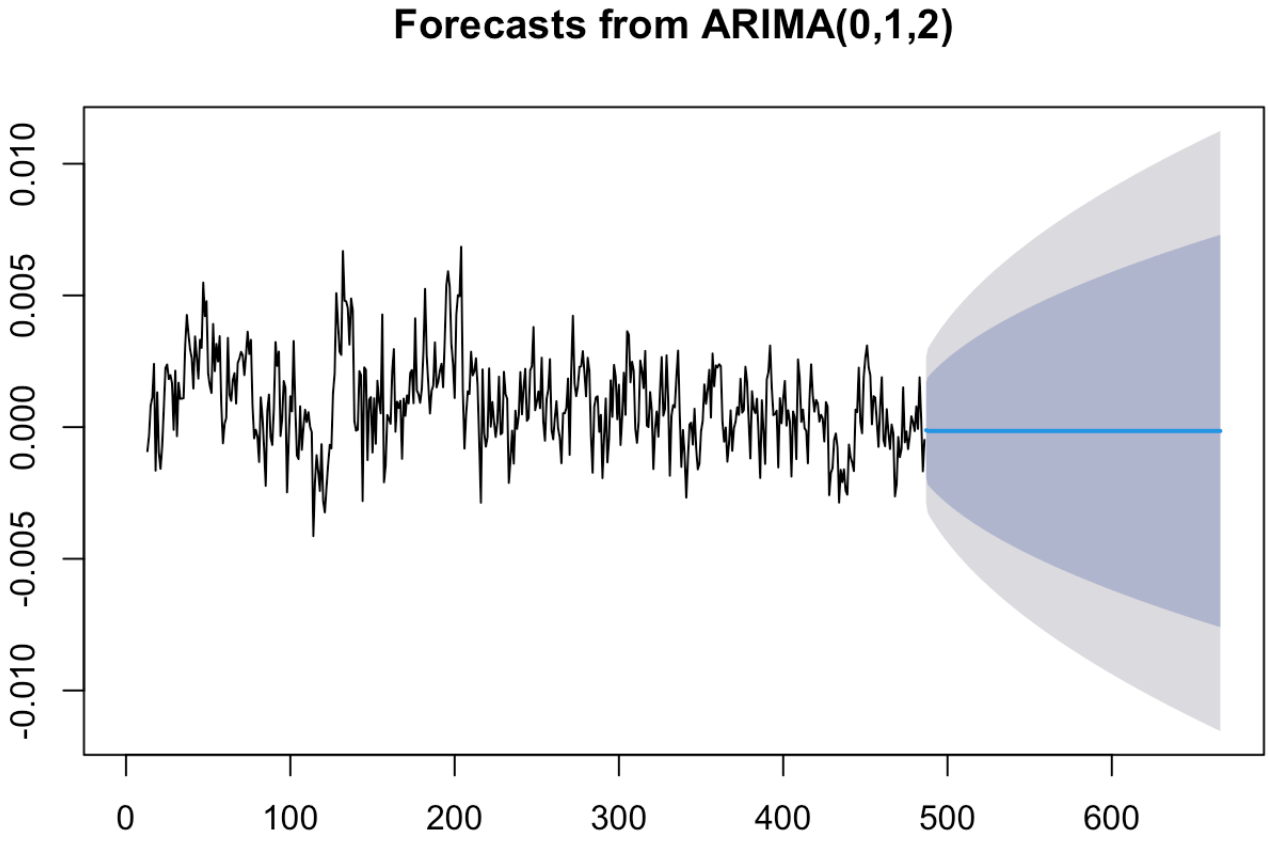



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)
## Q* = 8.6578, df = 8, p-value = 0.372
##
## Model df: 2.   Total lags used: 10
```

We are not rejecting the null hypothesis since the p-value (0.372) is larger than the significance value of 0.05. This implies that the residuals show no indication of considerable autocorrelation. In other words, the ARIMA(0,1,2)(3,1,0)12 model residuals appear to be white noise.

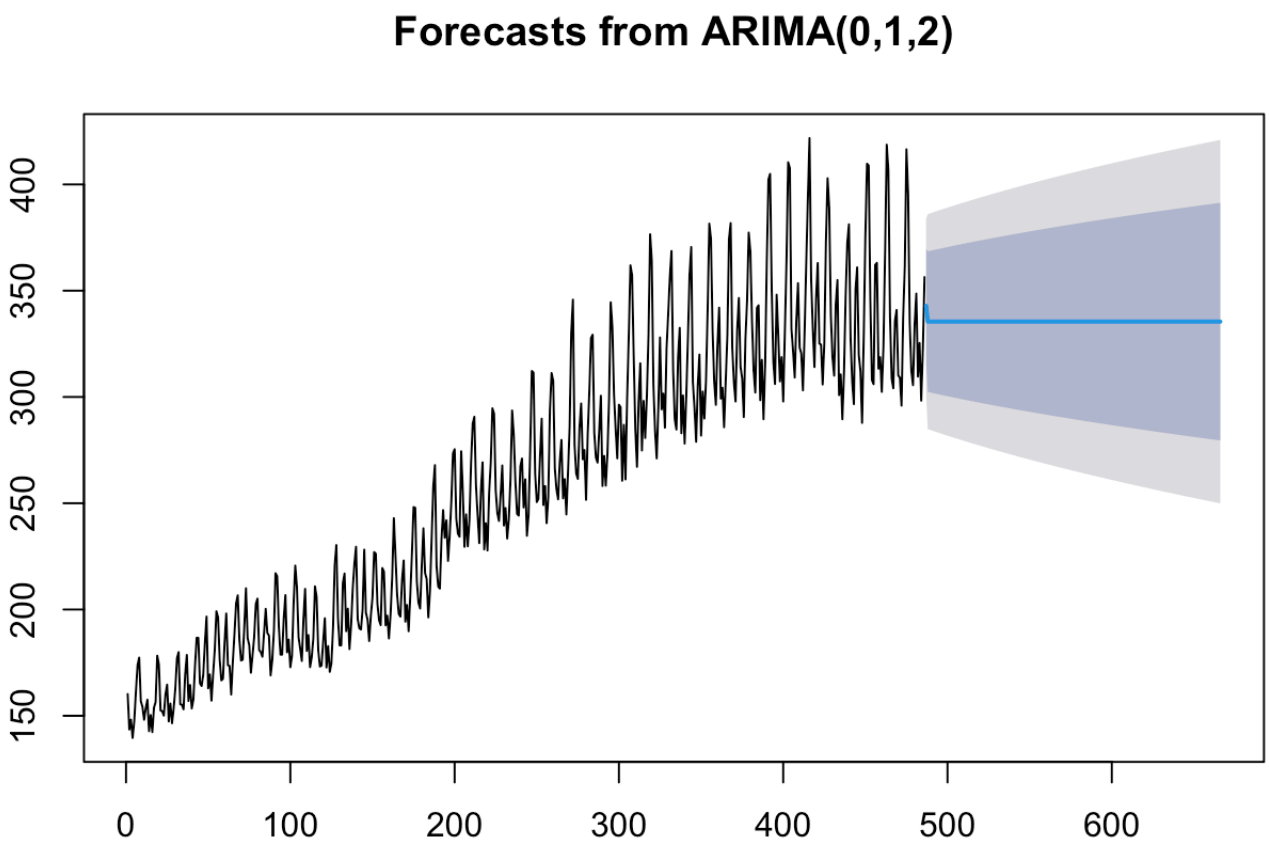
We do not have to try and fit better model as the residuals appers to be the white noise.

```
fc = forecast(arima2, h = 180)
plot( fc)
```




```
arima2 = Arima(df$value, order = c(0, 1, 2), seasonal = c(3,
    1, 0))
```

```
fc = forecast(arima2, h = 180)
plot(fc)
```




```
df1 = data.frame(fc=fc)
```

```
df2 <- readr::read_csv("ele.csv",show_col_types = FALSE)
df2 %>%
  mutate(Month=yearmonth(Month)) %>%
  tsibble(index=Month) ->
  df
df2 <- df2[, -2]
```

According to my comparison, these figures are still relevant after 6 years. The average absolute percentage error was just 2.17 percent and an RMSE of 9.71, which is only slightly lower than what we found in the best training model (RMSE = 7.23). Some model deterioration is to be expected, but these forecasts are obviously still relevant and might be much better with further retraining than what we observe 6 years out. This was accomplished by comparing actual EIA readings to expected levels between January 2017 and July 2019.
