WALMART SALES PREDICTION

FINAL REPORT – GROUP 3

Jia Hao Chuah, Beatriz Lopez Garcia, Andy Huang, Bhargavi Kashyap

Abstract

Businesses need accuracy in the forecasts not just for revenue generation, but also inventory management and resource allocation. This is even more important when businesses are customer facing and product oriented. Various factors such as holiday, day of the week, seasons impact the customer behavior. Optimal forecasting enables businesses to minimize inventory, decide markdowns, and create targeted marketing and pricing efforts (sales, vouchers etc.). The purpose of this report is to analyze and utilize different forecasting models with the objective of finding optimal ways with which a major retail company, in this case Walmart, can best and most accurately predict future sales. Sales data from four holidays that highly impact Walmart's weekly sales, namely Super Bowl, Easter, Thanksgiving and Christmas, were included in the analysis.

The accuracy of the models was evaluated based on the RMSE and MAE values as it were the only common parameters for all the different techniques used. The ones that performed better were the two ARIMA models (order = 0,0,2 & 2,0,2) and the Multiple Regression model. However, the best one was the ARIMA (2,0,2) using differentiated data by 52 (weekly data) as it obtained the lowest RMSE and MAE values. Though the selected model performed well and predicted sales for Walmart's a small portion of the data, deeper analysis on this dataset will uncover additional insights. Adding geographic location, more departments at a store or various stores and department will provide richer insights and accurate predictions.

Introduction

With the advancements in martech today, marketers now have the ability to create a more accurate forecast and plan for revenue. Marketing forecasting is a core component of market analysis, where it projects future number, characteristics and trends in one's target market, allowing marketers to showcase the downstream impact of their efforts. This allows many different upsides for companies, such as sales opportunities, new customer base, and higher revenue. Marketers are required to know historical conversion rates for different income channels and the types to see how leads flow through the funnel. Most marketing forecasting in general still tends to be unreliable because it was regularly based on averages across multiple channels and metrics. It requires too many assumptions, and since every channel pertains to different variation across leads, the assumptions get exponentially inaccurate the farther the forecast.

Therefore, many companies have pivoted to utilizing machine learning in time-series to gain the ability to produce better and more sophisticated forecasting results. Simple forecasting methods that produce good results have low variance and predictable trends. But for data that has complex, underlying structures such as seasonality and multiple causal factors, marketers need advanced forecasting methods to minimize the magnitude and variance of forecast errors. Machine learning forecasting learns from historical data, could possibly help marketers predict sudden changes in demand levels, price-cutting maneuvers of the competition, and large swings of the economy in the future.

The purpose for the report is to present an overview of time series forecasting models by discussing ways that Walmart can predict future sales accurately. Various time-series models are performed, and the accuracy for each time series model was validated and compared with one another at the end in order to select the best one. The variety of model goes from simpler to more complex models such as: mean, naïve and seasonal naïve methods, simple exponential smoothing and holt method, AR, MA, ARMA, ARIMA and SARIMA, and also time series regression models.

Literature Review

According to Tsoumakas, G. (2019) it is important for the companies to forecast future sales in order to minimize stocked and expired products inside stores and at the same time avoid missing sales. Also, it says that short-term predictions help in production planning and stock management, while long-term predictions can help in business development decision making. Additionally, Tsoumakas, G. (2019) states that food sales prediction is primarily a time-series forecasting problem but exhibits a number of interesting research challenges. The first challenge would be all the external features that also have predictive values (e.g. holidays, weather). Another challenge would be all the different predictions for each product in the store.

Another analysis involving sales forecasting was conducted by David A. Aaker, James M. Carman, and Robert Jacobson. The paper uses time series analysis to model advertising-sales relationships of six cereal brands under the assumption that there is feedback involved. More than 16 years of monthly data was used to explore the relationship of advertising and sales, while time series

analysis was chosen to be an appropriate approach because of the nature and the richness of the data. Another objective is that the researchers wish to demonstrate a two-way causation by utilizing time series analysis. The ARMA filters for both the time series of advertising and that of sales were examined, and the seasonality of each cereal brand was revealed. In addition, the analysis identified that there is either a moving average or an autoregressive term for each cereal brand with a one-month lag, which could possibly represent that past advertising budget decisions are affecting advertising budget response. As for the sales data, the seasonality was found as a 12-month seasonal term, while there is 3-month seasonality exist in some of the cereal brands. Furthermore, residual diagnostic was also properly performed, and the residuals gave no indication of significant autocorrelation left.

Another time series analysis on causality was conducted by Naci Büyükdağ, Ahmet Kaya, and Olgun Kitapci in 2019. Many companies wish to invest more resources in marketing to generate traffic, thus increase the profitability. The authors performed time series analysis with the purpose of revealing the way marketing expenditures influencing business performance, particularly in the insurance sector. The results revealed that marketing expenditure does have significant positive effect on business performance. The authors have also stated that the improvement in the business performance would also enhance the increase in marketing expenditure, since these two factors are bi-directional. This relationship allows the companies to attract investors more easily, and time series analysis is an ideal approach to achieve this objective since time series is composed of data collected at reg

Regression analysis and correlation analysis are static techniques that often being used in forecasting sales data. However, Kapoor, S. G., Madhok, P., and Wu, S. M. suggest that time series analysis could be an optimal approach when it comes to achieving accurate sales forecast. Time series analysis has been used in marketing research since it is an optimal solution to reveal insights hidden within serially correlated data. In this analysis, the sales of a particular line of the consumer product was analyzed, and the use of an alternative modeling approach to time series analysis in a real business situation was demonstrated. ARMA models, once again, appeared in the analysis and represent the stochastic component of the analysis. Same with what Tsoumakas, G. (2019) has suggested, the authors stated that one competitive advantage of time series analysis is that it is capable of capturing and explaining both short-term and long-term effects, thus is widely applicable in marketing research.

Data Source and Preprocessing

The dataset was obtained from Kaggle and contains anonymized information for Walmart stores. There are four different files available, however, for the purpose of this project only 3 were used:

- The **stores** file, contains information about the 45 stores, indicating the type and size of store
- The **train** file covers information regarding the store number, the department number, date), weekly sales and whether the week is a special holiday week or not. The date range for this document is: 02/05/2010 to 10/26/2012

- **Features** is the last file which contains additional data related to the store, department, and regional activity for the given dates

The Walmart dataset contains information on various features for 45 stores, up to 99 department each. To ensure replicable and optimal model is created - data for only store 1 – department 1 is used.

At the onset, distribution, trend and pattern in the data was analyzed. Some pre-processing of the dataset is required in order to create a new dataset that suits the direction of the analysis direction. After merging new variables by date and creating a new dataset, the new dataset contains 143 observations and 13 variables.

It is important to point out that from now on the dependent variable is going to be "Weekly sales" and that the variable "Markdowns" are going to be dropped as it is anonymized data related to promotional markdowns that Walmart is running and this information is only available after Nov 2011, and is not available for all stores all the time.

The description for the variables after data preprocessing are mentioned below:

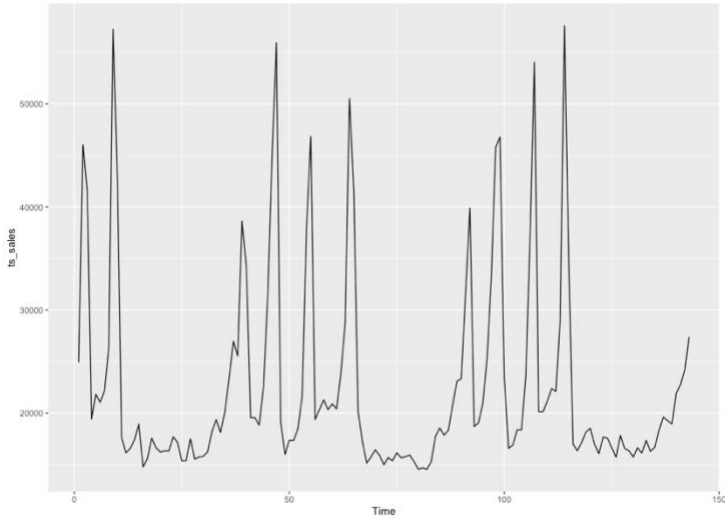
Variable name	Type of variable	Description
Store	Numeric	Labeled 1-45*[1]
Type	Categorical	Categorized to 4 levels (A/B/C/D) *
Size	Numeric	Size of the store
Department	Numeric	Represent the department number *
Date	Factor variable	Monday of each week
Weekly Sales	Numeric	Sales for the given department in the given store
IsHoliday	Categorical	Special holiday week – TRUE or FALSE**
Temperature	Numeric	Average temperature in the region ***
Fuel_Price	Numeric	Cost of fuel in the region ***
Markdown 1-5	Numeric	Promotional markdowns the Walmart
CPI	Numeric	Consumer Price Index
Unemployment	Numeric	Unemployment rate

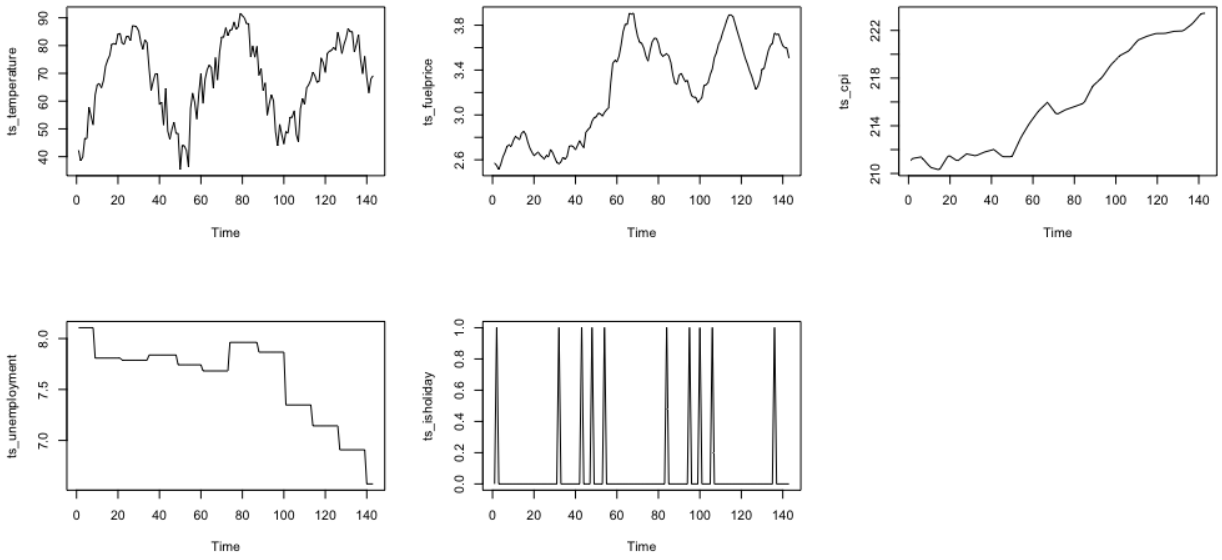
[1] *There is no information regarding which Walmart store is each one, neither the different types of stores or the department number
** There are four main holidays each year: Super Bowl, Easter, Thanksgiving and Christmas.
*** There is no information regarding the region of the department store.

Methodology

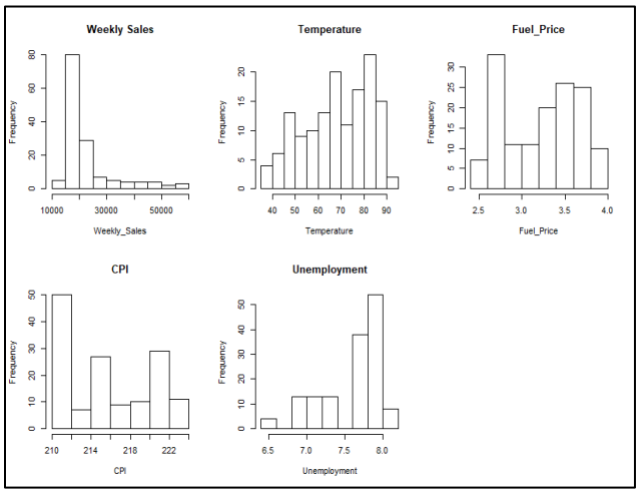
EXPLORATORY DATA ANALYSIS (EDA)

EDA was performed on the dataset prior to model building to further understand the dataset. The time series plot for weekly sales for Store 1 Department 1 is shown below:





The time series plot shows four spikes of weekly sales increase corresponding to the holiday events mentioned (Superbowl, Easter, Thanksgiving and Christmas). Even though the yearly patterns were repetitive, the data is not strictly seasonal. The time series plots of other features are also plotted. Temperature shows correlation with weekly sales, where periods with high weekly sales corresponds to lower temperature.



The histogram for weekly sales and the other attributes were also plotted. The histogram shows that the weekly sales are generally around $10,000 to $30,000. Weekly sales on the higher range indicates sales for the holiday periods. The average temperature in this region falls on the range of 80 °F to 85 °F most frequently, followed by 65 °F to 70 °F. As for the cost of fuel in the region, typically the fuel usually cost approximately $2.75, but also frequently increase to roughly $3.5 to $3.75. Consumer price index typically falls on 210 to 212. Lastly, the unemployment rate occurs most frequently at approximately 8%.

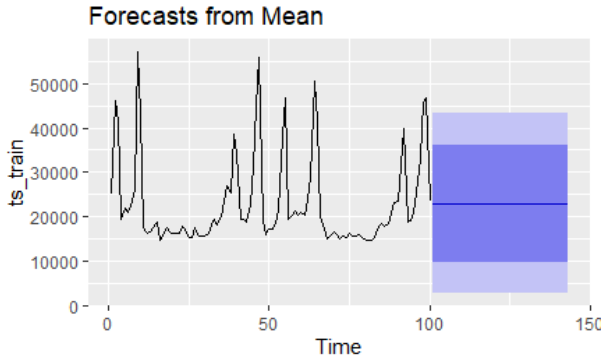
MODEL BUILDING AND RESULTS

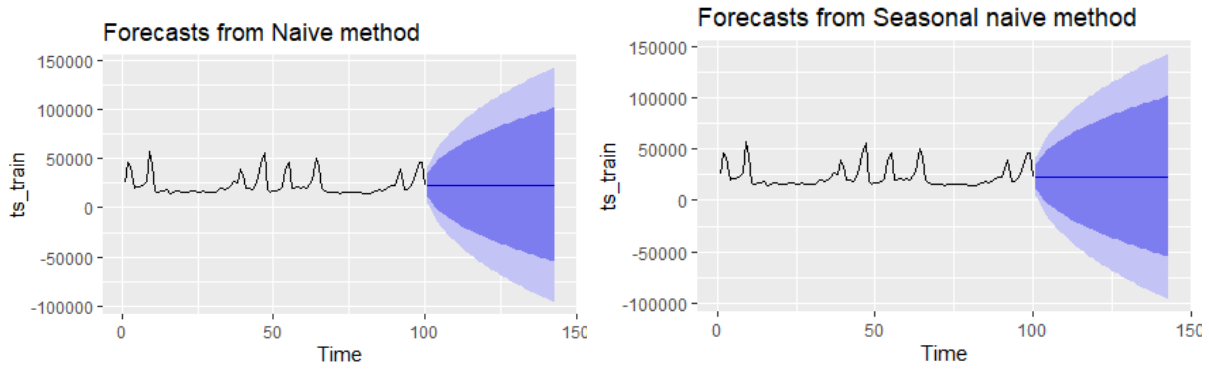
DATA TRAINING/TEST SPLIT

The dataset was split into training and test set prior to model building. The training set includes weekly sales data from 2010 to 2011, whereas the test set includes sales in 2012. The training set data was be used to build the model, whereas the test data was used to compare accuracy of the forecast data to evaluate our models.

MEAN, NAÏVE & SEASONAL NAÏVE FORECASTING

Simple forecasting methods were used to forecast results from the training data. Mean forecasting takes the mean of the historical data to perform the forecasting, Naïve forecasting takes the last data point predict future trends, and Seasonal Naïve method incorporates seasonal settings. The figures below show the result of the simple forecasting methods:

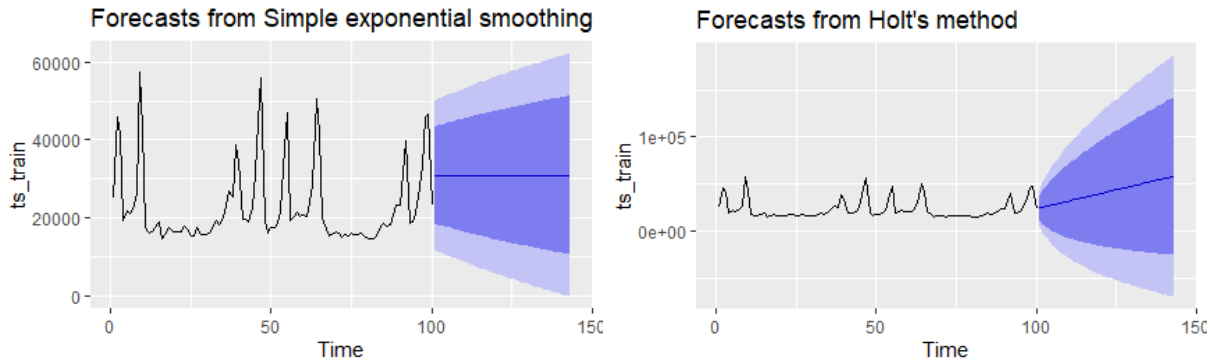




The results show simple but undesirable forecasting results. There is little difference between results between Naive and Seasonal Naive forecasting since the data is not seasonal.

SIMPLE EXPONENTIAL SMOOTHING & HOLT'S METHOD FORECASTING

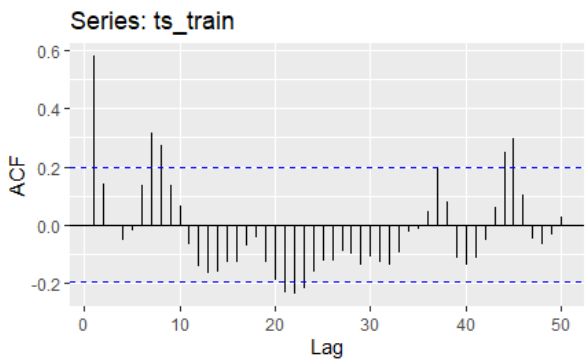
SES and Holt's model were built to forecast the model. The setting for SES alpha value was set to 0.2, whereas the initial setting for Holt's method was set to optimal.

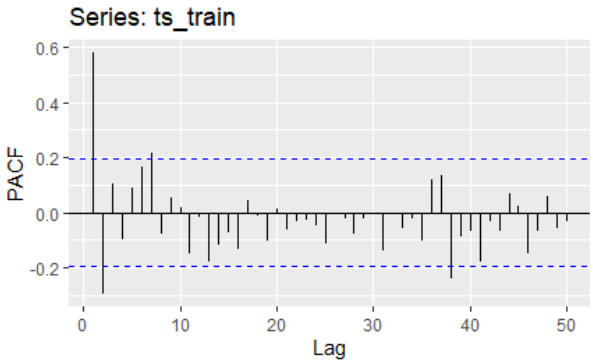


The SES and Holt's models yield better results compared to the simple forecasting methods. The SES forecast are relatively stable, whereas Holt's method forecast identifies an upward trend. The AIC/BIC values were also obtained from both models, which were 2299.13/2304.34 for SES, and 2299.24/2312.27 for Holt's. The results show that SES forecast performs slightly better than Holt's method forecast.

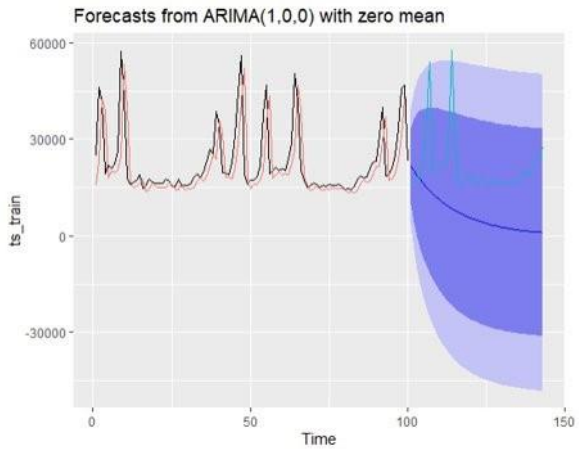
ARIMA FORECASTING

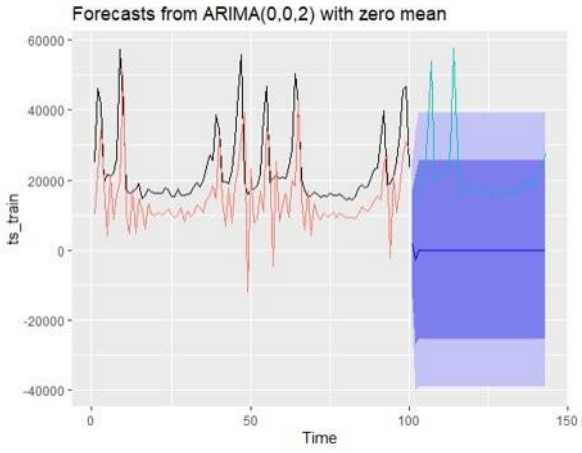
To build AR, MA, and ARMA models, the ACF and PACF plots were investigated to identify suitable orders for the plots. Based on the plots below, the order p = 1 and q = 2 were initially identified, but various combinations of orders were tested on to build the models with the best and lowest AIC/BIC values.

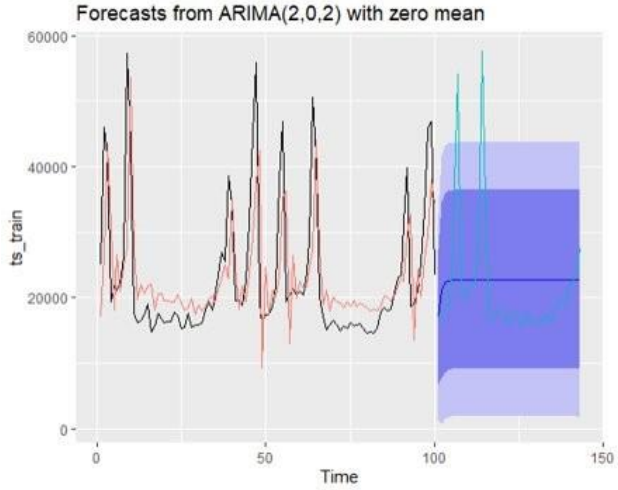




The AR, MA, ARMA were built on the training set with settings AR: ARIMA (1,0,0); MA: ARIMA (0,0,2) and ARMA: ARIMA (2,0,2). The models were used to create forecast predictions, where the accuracy metrics were used to evaluate the predictions against the test set. The AR and MA model provided unsatisfactory results as their forecast drops to 0. Based on the forecast and AIC/BIC values, ARMA performed best out of the three models.





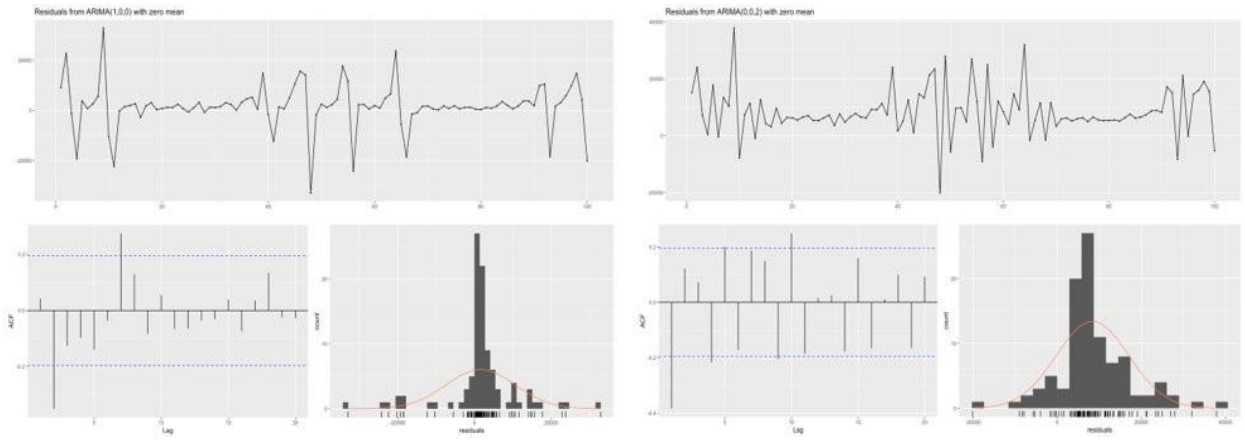


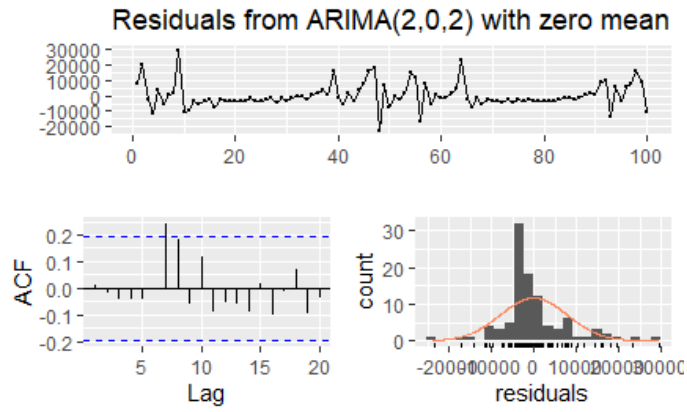
The coefficient test and residual diagnostics to validate the quality of the model was performed. While the coefficient test shows that all variables from all three models are significant, the residual plots shows that there are still serial correlations present in the models, and the three models did not pass the Ljung's Box test.

This means that the models can be further improved. Since the models built so far did not model the holiday seasons well, differencing needs to be applied on the dataset to generate stationary time series data to create a better model.

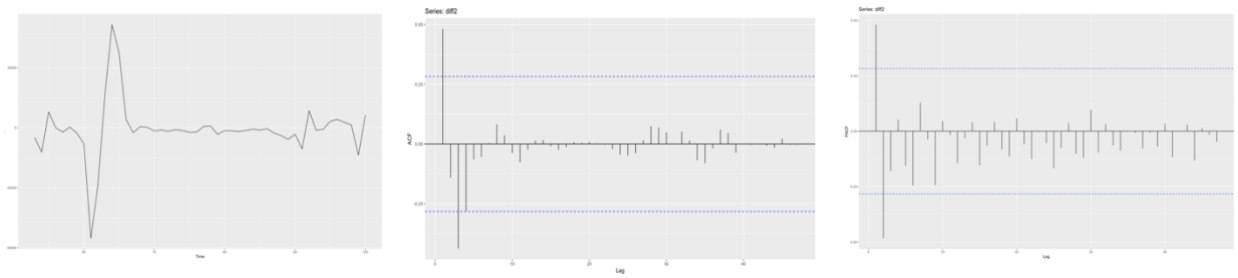
ARIMA - USING DIFFERENCED DATA

To further validate that ARMA: ARIMA (2,0,2) is the best model up to this point, both coefficient test and the residual diagnostics were performed for all three of AR, MA, and ARMA models. The ACF plot of the residuals of the ARMA model present the best outcome amongst all three models.

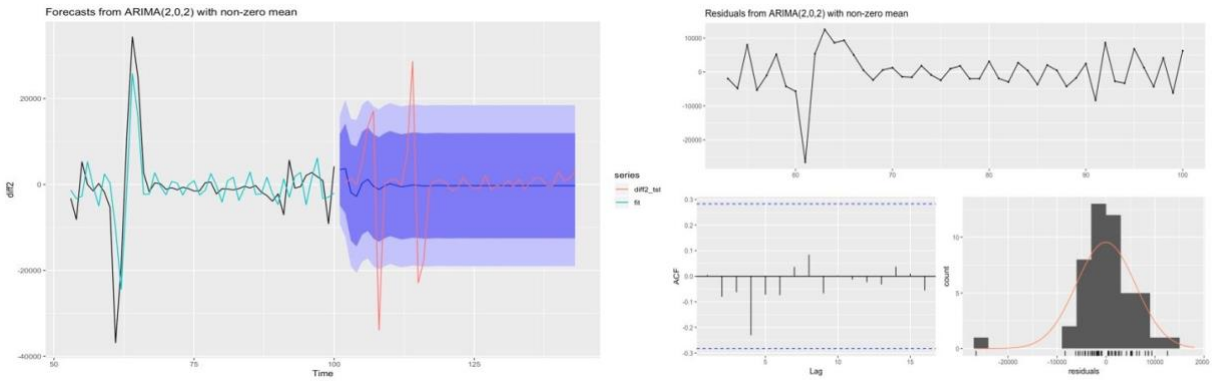




However, even though the ACF plot of the ARMA model performed relatively better than AR and MA, there is still evidence of serial correlation in the ARMA model. Specifically, even though the coefficients of the ARMA model passed the significance test, the residuals plot as well as the Ljung's Box test indicate that there is still collinearity exist in the ARMA model, which means that the model could still be improved. As a result, log transformation, differencing, and seasonal differencing for our data were each tested to find out the best method.



Log transformation does not benefit our analysis much, as the data after transformation is still significantly unstable. As for differencing, after exploring regular differencing, differencing by 12, and differencing by 52; the result after differencing by 52 present the best outcome. The graphs presented above as well as the ACF and PACF plots are the results after differencing by 52. It is not a surprise that differencing by 52 performed better, since our data is weekly sales data. The ACF and PACF plot of differencing by 52 also present the best result, as shown above.



Going forward, an ARMA model has been fitted on the data after differencing by 52. The AIC and BIC values, the coefficient test, the residual plots, ant the Ljung's box test have been examined as the evaluation of the model performance. The result is that the ARMA model has significantly been improved after differencing. The AIC and BIC values have been reduced to 989.09 and 1000.31, respectively. Moreover, the coefficient test of significance, the residuals plot, and the Ljung's box test of normality shown a unified result that this ARMA model is the best model up until now. As a result, this optimal ARMA model has been used to perform forecasting, as shown above.

Furthermore, the possibility of fitting a SARIMA model was also considered in order to test out the seasonality of our data. However, the result is that the data does not fit with SARIMA model. The ACF plot of the original time series data and the ACF plot of the data after differencing do not display patterns to incorporate SARIMA. As a conclusion at this stage of the analysis, our best model is ARIMA (2,0,2) based on the training data that differenced by 52.

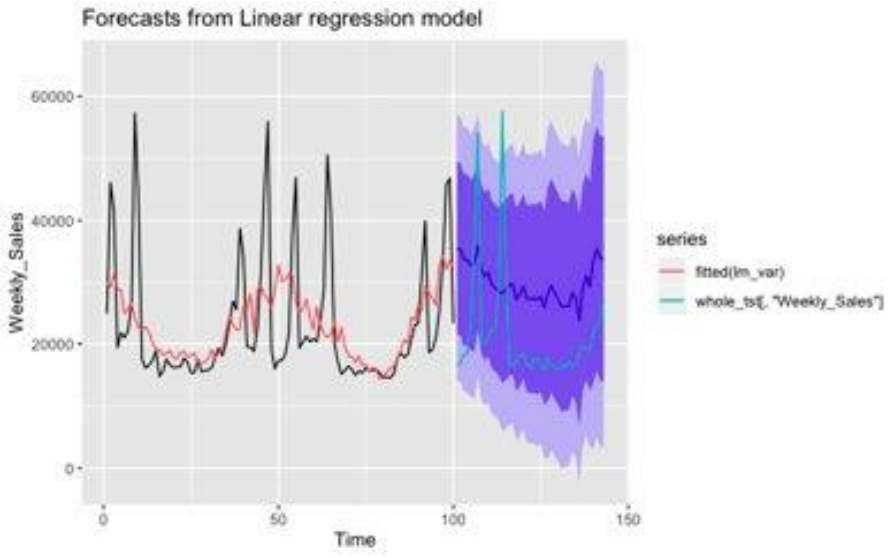
TIME SERIES REGRESSION

A time series analysis regression was performed on the data. The data used was **not differentiated** and the different models used on the data were: linear trend, multiplicative trend, quadratic trend and multiple regression.

Below there is a table with the RMSE and MAE of the different models:

Models	**RMSE**	**MAE**
Multiple Regression	8750.152	5809.469
Quadratic Regression	10012.2	7379.959
Linear Regression	10149.16	7514.46
Multiplicative Regression	10278.86	6882.345

The best linear model is the multiple regression one, as it has the lowest RMSE and MAE. Also, it is possible to see in the plot how it tries to forecast in the best possible way the holidays periods.



However, it is important to point out that there was only one variable significant, *temperature*. Provided below is the model equation:

Weekly sales = -68036.417 – 323.107*temperature

Conclusion

Through the process of weekly sales prediction, various methods from simple to complex timeseries were utilized. At the onset, exploratory data analysis was conducted to understand the distribution of variables, both dependent and independent. Additionally, timeseries plot were created to understand the trend and/or pattern in the data and any potential correlation among them. It was observed that temperature and holiday had similar peaks and troughs and fuel price data showcased upward trend. Weekly Sales data showcased pattern, especially during holiday season such as Easter, super bowl, however it lacked seasonality. Understanding the initial pattern and for model validation, the dataset was split into training and test set to build an optimal model for prediction of the weekly sales for Walmart.

Simple prediction model such as mean, naïve and seasonal naïve was utilized. Since the data does not have seasonal characteristics, there is no difference in performance of naïve and seasonal naïve. However, overall, none of the simple model were able to predict weekly sales effectively. RMSE and MAE values were considered when analyzing the performance of the model. Simple Exponential Smoothening (SES) and Holt's Method was tried with alpha set was 0.2 and initial at 'Optimal'. Both these methods were not able to capture the trend in data and predict the weekly sales. Slightly complex methods were evaluated i.e. AR, MA or ARMA. ACF and PACF plots were analyzed for the order of p and q; p = 1 and q = 2. AR (1), MA(2), ARMA (2,0,2) were applied on the original data. Both AR and MA did not perform well as the forecast value dropped to zero. ARMA (2,0,2) on original data performed better than previous models but could not forecast optimally. Residuals diagnostic and coefficient test of these models was performed. ACF model of ARMA model performed the best amongst all three, however Ljung Box Test showed serial correlation.

Since the data was weekly, differencing (52) was performed to address the correlation and ARMA (2,0,2) model was created. ACF and PACF plots were evaluated. This model performed well when tested for residual, coefficient and Ljung Box Test. It also had the lowest AIC and BIC value.

To ensure all the forecasting models were compared on uniform parameters, accuracy of each model was checked and considered RMSE and MAE parameters were considered as the baselines.

Overall, ARIMA (2,0,2) model performed the best

Models	**RMSE**	**MAE**
ARMA (2,0,2) (diff)	5963.25	4185.526
ARMA (2,0,2)	7826.237	5465.461
Multiple Regression	8750.152	5809.469
AR (1,0,0)	9164.032	5568.944
Naive	9329.313	5119.153
Snaive	9329.313	5119.153
Holt's Method	9354.685	5127.211
SES	9634.053	7124.46

Models	RMSE	MAE
Quadratic Regression	10012.2	7379.959
Linear Regression	10149.16	7514.46
Mean	10149.26	7510.949
Multiplicative Regression	10278.86	6882.345
MA (0,0,2)	12019.59	9710.882

Future Work

Although the analysis was performed as thorough as it could be, there are still some future research directions that could potentially providing a more cohesive and meaningful results. For instance, the optimal model, ARIMA (2,0,2) was built on data that already been differenced by 52. Coercing the data back to its usual form (reverse transform) will provide actual forecast values, thus highly likely to provide more meaningful result. In addition, more sophisticated techniques such as Neural Network could also be used in time series and sales forecasting, that has the potential to predict sales information with more precision.

Moreover, as an extension of analysis, using multiple store data or multiple department within a store data would possibly provide better and more holistic results. Additionally, understanding the geographic location could also be beneficial in terms of providing a more realistic and meaningful analysis. Lastly, raw data on markdowns and their relation to sales will also be another dimension on which sales, thus profitability can be evaluated.

References

Kaggle.com. 2020. *Walmart Recruiting - Store Sales Forecasting | Kaggle*. [online] Available at: <https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/data> .

Tsoumakas, G. A survey of machine learning techniques for food sales prediction. *Artif Intell Rev* 52, 441–447 (2019). https://doi.org/10.1007/s10462-018-9637-z

Ruff, Kathy . Northeast Pennsylvania Business Journal ; Dallas Vol. 32, Iss. 11, (Nov 2017): 22.

Aaker, D.A., Carmen, J.M., & Jacobson, R. (1982). Modeling Advertising Sales Relationships Involving Feedback: A Time Series Analysis of Six Cereal brands. *Journal of marketing Research (JMR), 19*(1), 116-125. https://doi-org.ezproxy.depaul.edu/10.2307/3151536

Büyükdag, N., Kaya, A., & Kitapci, O. (2019). The Effect of Marketing Expenditure on Business Perfromance: Time Series Analysis on Causality. *Journal of Applied Economics & Business Research, 9*(4), 197-211.

Kapoor, S.G., Madhok, P., & Wu, S. M. (1981). Modeling and Forecasting Sales Data by Time Series Analysis. *Journal of marketing Research (JMR),* 18(1), 94-100. https://doi-org.ezproxy.depaul.edu/10.2307/3151318

Group 3 - Walmart Sales Analysis & Forecast

Imported libraries

Reading Data and Preprocessing

Read stores data: Store No, store type, store size. Can evaluate size of stores.

```
stores = read.csv('stores.csv')
head(stores)
```

	Store <int>	Type <fctr>	Size <int>
1	1	A	151315
2	2	A	202307
3	3	B	37392
4	4	A	205863
5	5	B	34875
6	6	A	202505

6 rows

```
#stores[which.max(stores$Size),]
#stores[which.min(stores$Size),]
```

The biggest store is number 13, while the smallest store is number 5.

Read Stores Features: Date, Temperature, Fuel Price, MarkDowns, CPI, Unemployment, IsHoliday

```
features = read.csv('features.csv')
head(features)
```

	St… <int>	Date <fctr>	Temperature <dbl>	Fuel_Price <dbl>	MarkDo… <dbl>	MarkDo… <dbl>	MarkDo… <dbl>	MarkDo… <dbl>	Mark <dbl>
1	1	2010-02-05	42.31	2.572	*NA*	*NA*	*NA*	*NA*	
2	1	2010-02-12	38.51	2.548	*NA*	*NA*	*NA*	*NA*	
3	1	2010-02-19	39.93	2.514	*NA*	*NA*	*NA*	*NA*	
4	1	2010-02-26	46.63	2.561	*NA*	*NA*	*NA*	*NA*	
5	1	2010-03-05	46.50	2.625	*NA*	*NA*	*NA*	*NA*	

	St…	Date	Temperature	Fuel_Price	MarkDo…	MarkDo…	MarkDo…	MarkDo…	Mark
	<int>	<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<
6	1	2010-03-12	57.79	2.667	*NA*	*NA*	*NA*	*NA*	

6 rows | 1-10 of 13 columns

Read train data : store, dept, date, Weekly Sales, IsHoliday

```
train = read.csv('train.csv')
head(train, 10)
```

	Store	Dept	Date	Weekly_Sales	IsHoliday
	<int>	<int>	<fctr>	<dbl>	<lgl>
1	1	1	2010-02-05	24924.50	FALSE
2	1	1	2010-02-12	46039.49	TRUE
3	1	1	2010-02-19	41595.55	FALSE
4	1	1	2010-02-26	19403.54	FALSE
5	1	1	2010-03-05	21827.90	FALSE
6	1	1	2010-03-12	21043.39	FALSE
7	1	1	2010-03-19	22136.64	FALSE
8	1	1	2010-03-26	26229.21	FALSE
9	1	1	2010-04-02	57258.43	FALSE
10	1	1	2010-04-09	42960.91	FALSE

1-10 of 10 rows

The goal is to create a good model for the time series, therefore we would use Store 1 Dept 1's time series data. Below are the preprocessing steps to create the data subset we would be analyzing.

```
train_new = train[train$Store == 1,]
train_new = train_new[train_new$Dept == 1,]

feature_new = features[features$Store == 1,]
feature_new = feature_new[1:143,]

# Merge new variables by date, dropping duplicated columns
clean = merge(feature_new, train_new, by = "Date", all = TRUE)
drops <- c("Store.y","Dept","IsHoliday.y")
clean = clean[ , !(names(clean) %in% drops)]
View(clean)

# Clean up dataset, reformate date type
data <- rename(clean, c("Store.x"="Store", "IsHoliday.x"="IsHoliday"))
head(data,20)
```

	Date <fctr>	St... <int>	Temperature <dbl>	Fuel_Price <dbl>	MarkDo... <dbl>	MarkDo... <dbl>	MarkDo... <dbl>	MarkDo... <dbl>	Mar
1	2010-02-05	1	42.31	2.572	*NA*	*NA*	*NA*	*NA*	
2	2010-02-12	1	38.51	2.548	*NA*	*NA*	*NA*	*NA*	
3	2010-02-19	1	39.93	2.514	*NA*	*NA*	*NA*	*NA*	
4	2010-02-26	1	46.63	2.561	*NA*	*NA*	*NA*	*NA*	
5	2010-03-05	1	46.50	2.625	*NA*	*NA*	*NA*	*NA*	
6	2010-03-12	1	57.79	2.667	*NA*	*NA*	*NA*	*NA*	
7	2010-03-19	1	54.58	2.720	*NA*	*NA*	*NA*	*NA*	
8	2010-03-26	1	51.45	2.732	*NA*	*NA*	*NA*	*NA*	
9	2010-04-02	1	62.27	2.719	*NA*	*NA*	*NA*	*NA*	
10	2010-04-09	1	65.86	2.770	*NA*	*NA*	*NA*	*NA*	

1-10 of 20 rows | 1-10 of 14 columns Previous **1** 2 Next

```
data$Date = as.Date(as.character(data$Date), format = '%Y-%m-%d')
str(data)
```

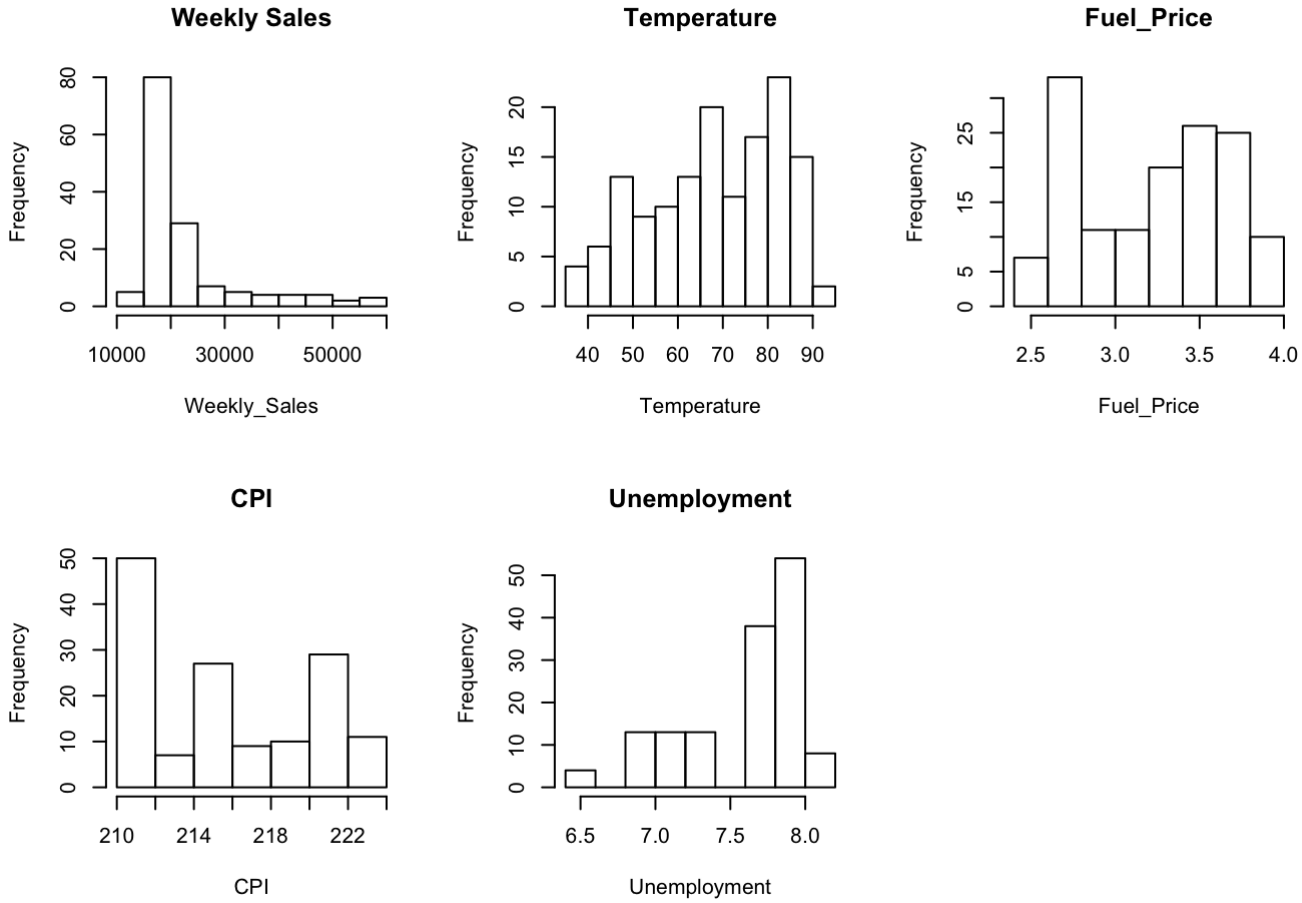
```
## 'data.frame':    143 obs. of  13 variables:
##  $ Date        : Date, format: "2010-02-05" "2010-02-12" ...
##  $ Store       : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Temperature : num  42.3 38.5 39.9 46.6 46.5 ...
##  $ Fuel_Price  : num  2.57 2.55 2.51 2.56 2.62 ...
##  $ MarkDown1   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MarkDown2   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MarkDown3   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MarkDown4   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ MarkDown5   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ CPI         : num  211 211 211 211 211 ...
##  $ Unemployment: num  8.11 8.11 8.11 8.11 8.11 ...
##  $ IsHoliday   : logi  FALSE TRUE FALSE FALSE FALSE FALSE ...
##  $ Weekly_Sales: num  24924 46039 41596 19404 21828 ...
```

Our subset data will have 143 observation and 13 features.

EDA Analysis on the data

Histogram for our features to identify distribution

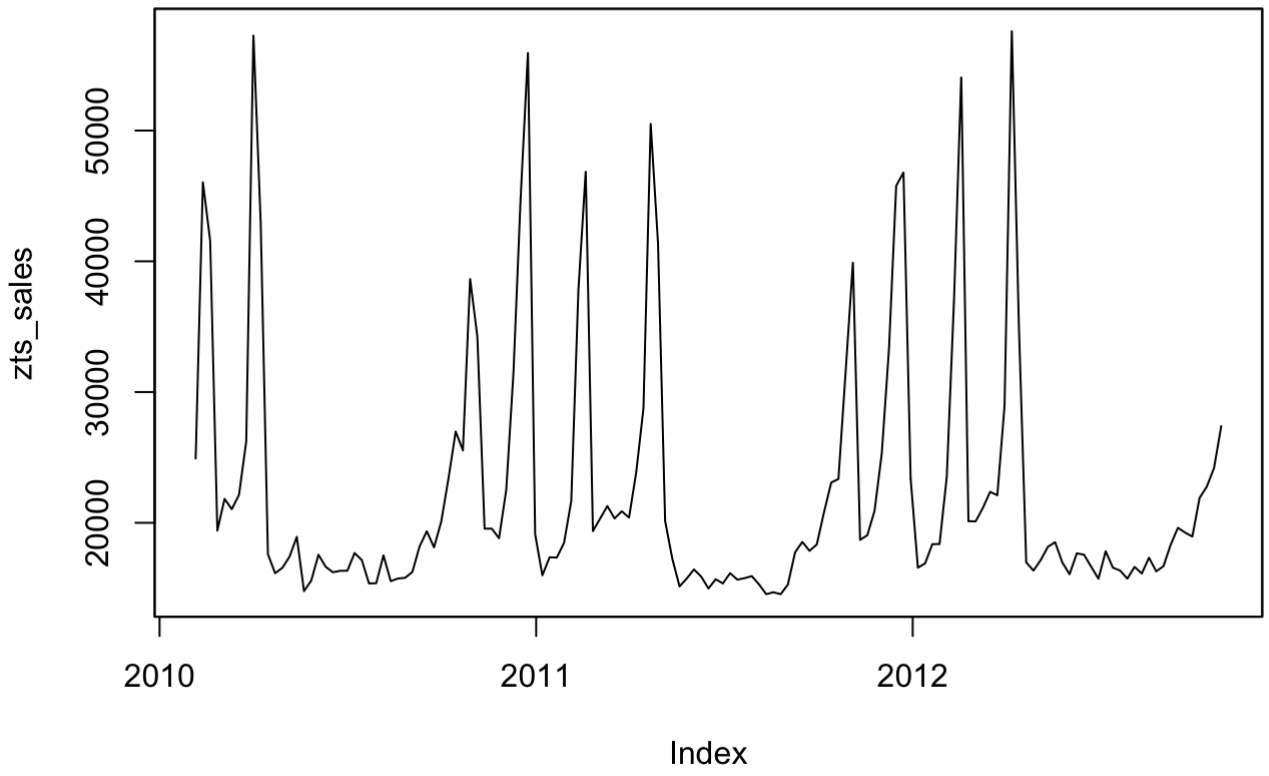
```
#distribution of weekly sales
attach(data)
par(mfrow=c(2,3))
hist(Weekly_Sales, main="Weekly Sales")
hist(Temperature, main="Temperature")
hist(Fuel_Price, main="Fuel_Price")
hist(CPI, main="CPI")
hist(Unemployment, main="Unemployment")
```



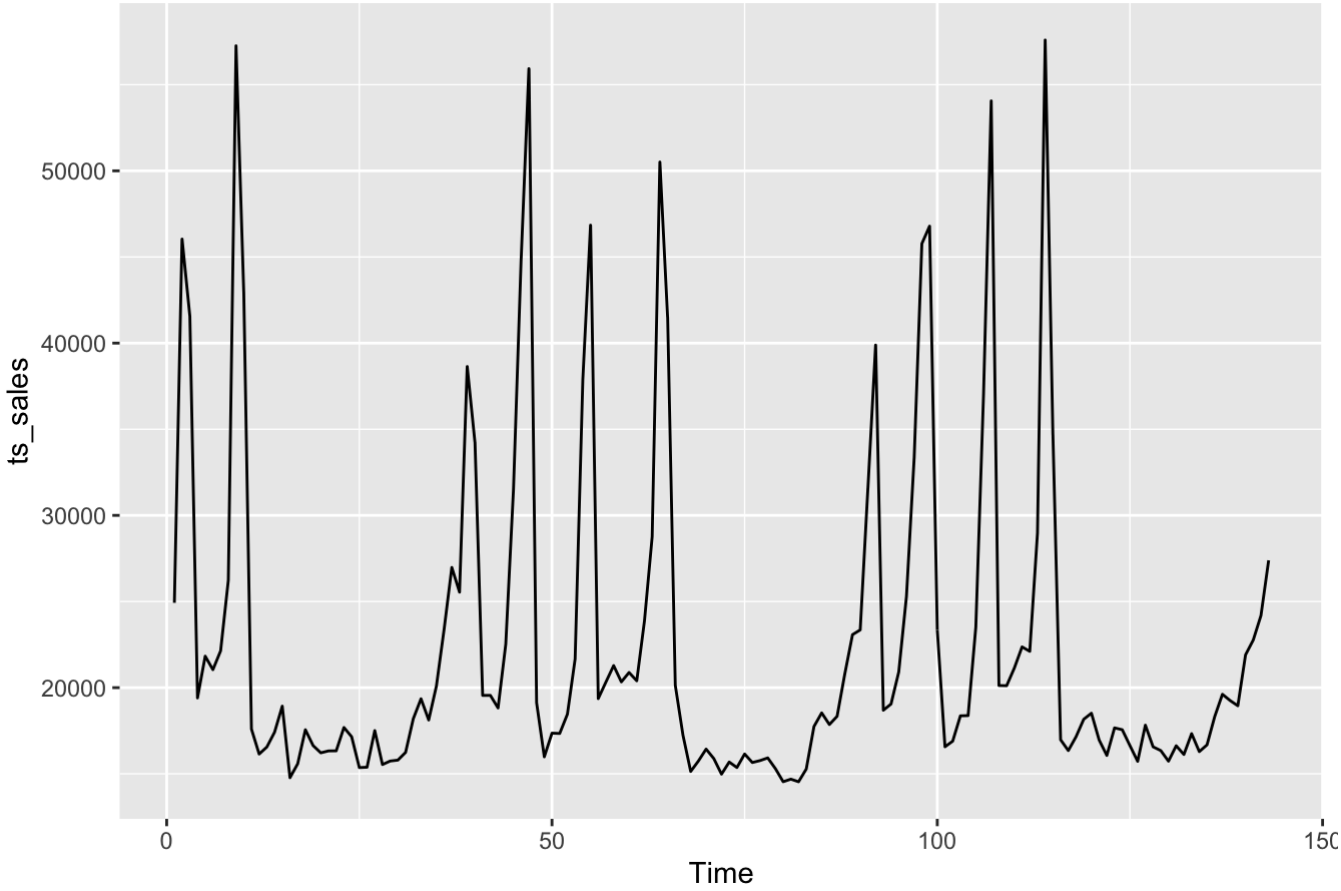
While we are mostly concerned with weekly sales, we can see that the weekly sales are mostly distriibuted around $10000 to $30000. The weekly sales on the high end are due to holiday seasons. The rest of the features are either evenly distributed or right skewed (Unemployment)

Time series plot: Weekly Sales

```
#Zoo Data
zts_sales = zoo(data$Weekly_Sales, data$Date)
plot(zts_sales)
```




```
#TS Data
ts_sales = ts(data[,'Weekly_Sales'])
autoplot(ts_sales)
```

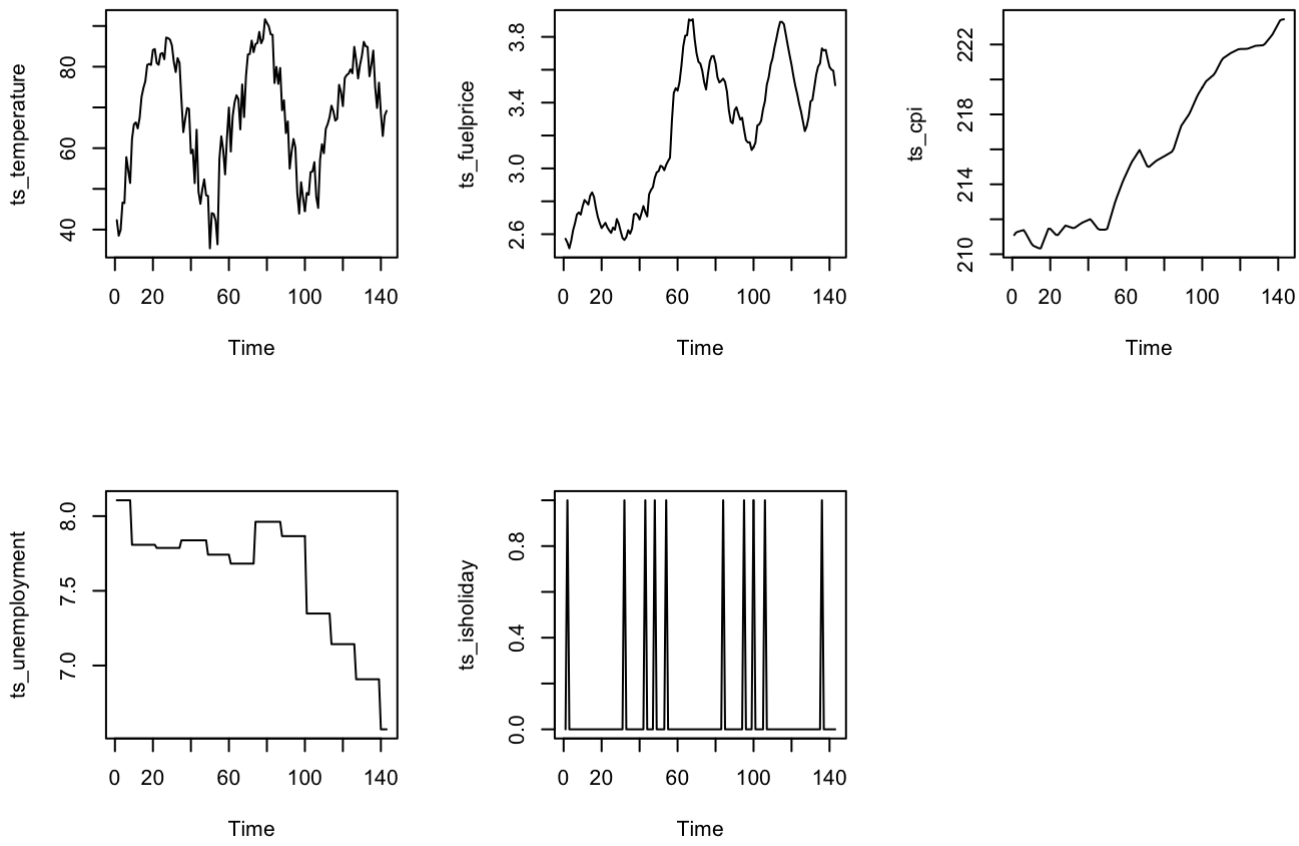



```
#frequency(ts_sales)
```

For our time series plot, we can see 4 spikes of weekly sales per year due to the 4 big holidays. Checking frequency, we realize the weekly sales are not strictly seasonal.

Time Series Plots: Other features

```
ts_temperature = ts(data[,'Temperature'])
ts_fuelprice = ts(data[,'Fuel_Price'])
ts_cpi = ts(data[,'CPI'])
ts_unemployment = ts(data[,'Unemployment'])
ts_isholiday = ts(data[,'IsHoliday'])
par(mfrow=c(2,3))
plot(ts_temperature)
plot(ts_fuelprice)
plot(ts_cpi)
plot(ts_unemployment)
plot(ts_isholiday)
```



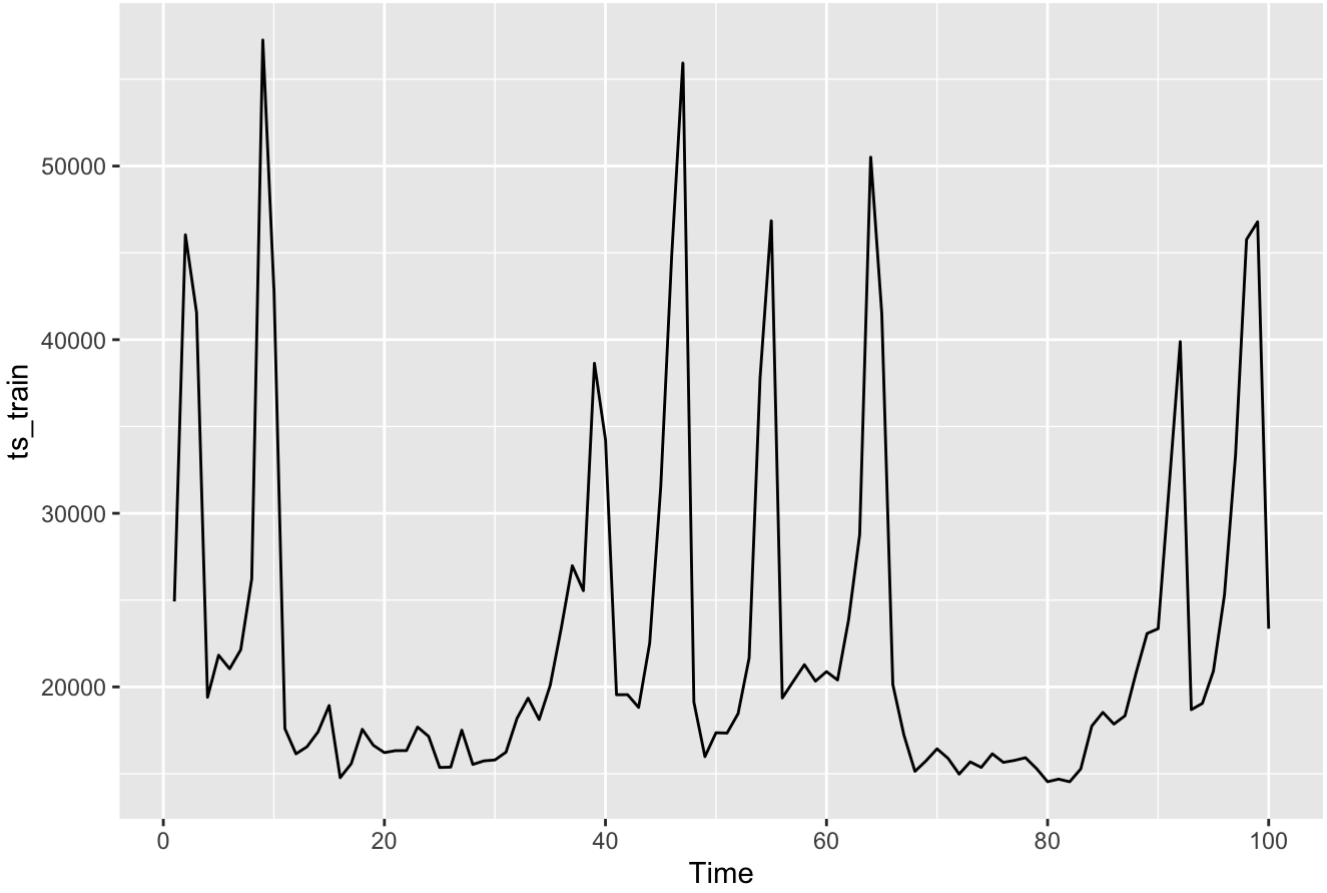
We plot our features against time to see if they seemingly correlate to the pattern of weekly sales price. We can identify that temperature dips during the holiday seasons, and the isHoliday plot corresponds to the 4 spikes in our weekly sales. On the other hand, fuelprice and api has been increasing over the years, whereas unemployment has been decreasing.

Time Series Plots: Markdowns

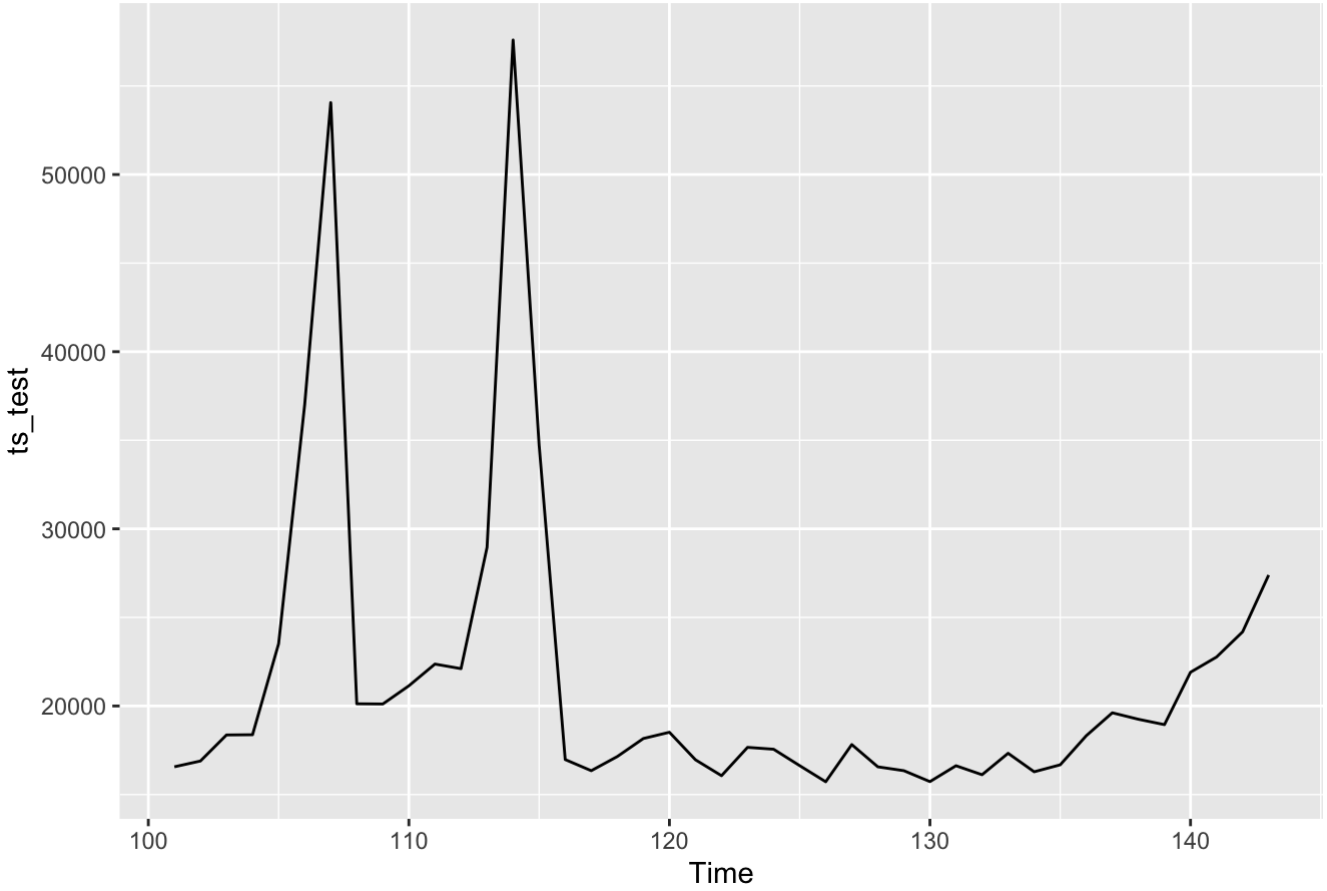
MarkDown1-5 - we are not going to analyze it, as it is anonymized data related to promotional markdowns that Walmart is running. It is only available after Nov 2011, and is not available for all stores all the time.

Train/Test Split (Split 1st portion, then forecast and compare with test (Only on the best model for now))

```
ts_train = window(x = ts_sales, end = c(100))
ts_test = window(x = ts_sales, start = c(101))
autoplot(ts_train)
```




```
autoplot(ts_test)
```

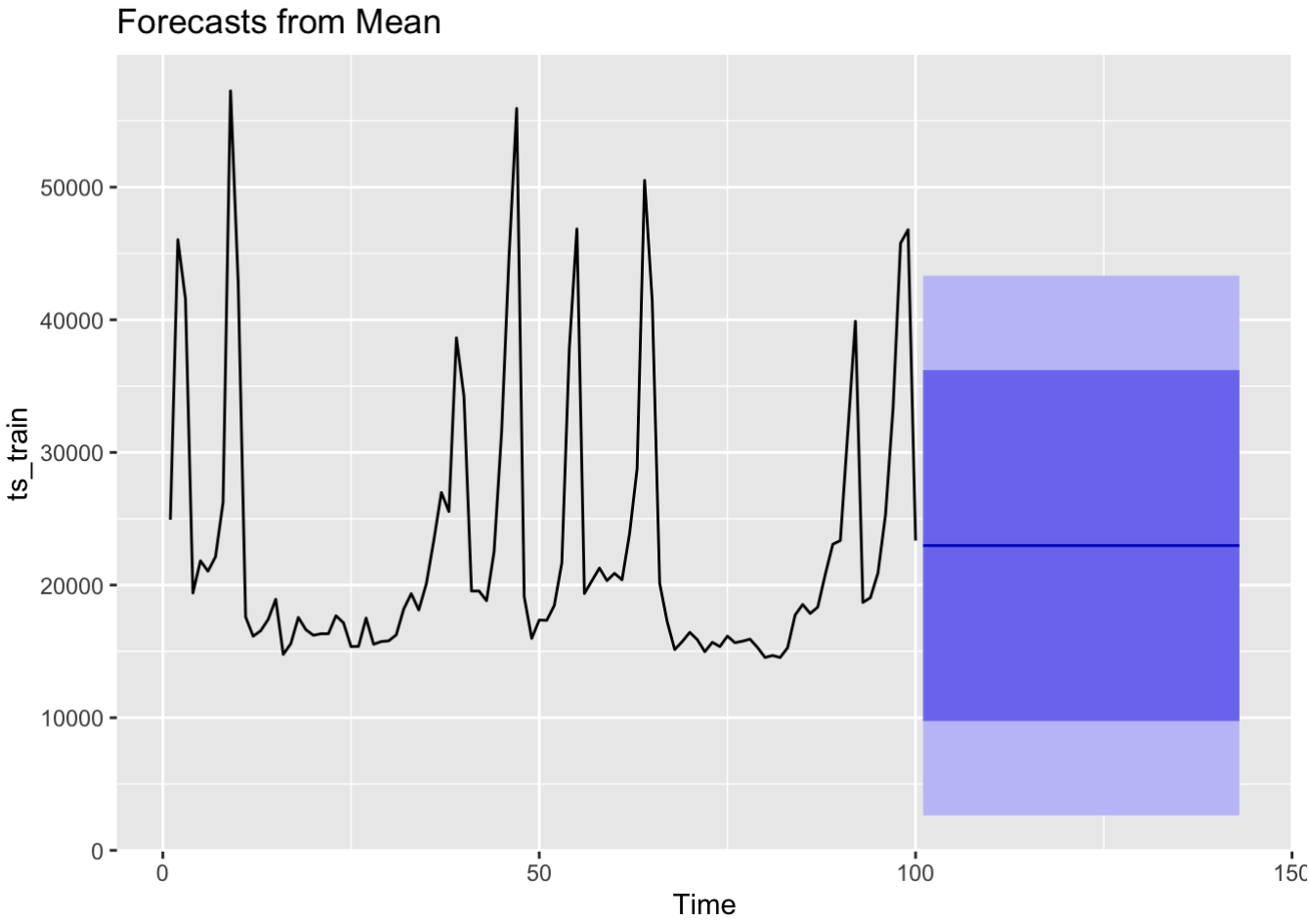


Building Models

Simple Forecast Models

We decided to perform a couple of simple forecasting model to predict sales for the next 12 months. We first forecast our data using Mean, Naive and Seasonal Naive forecasting methods.

```
autoplot(meanf(ts_train, h = 43))
```



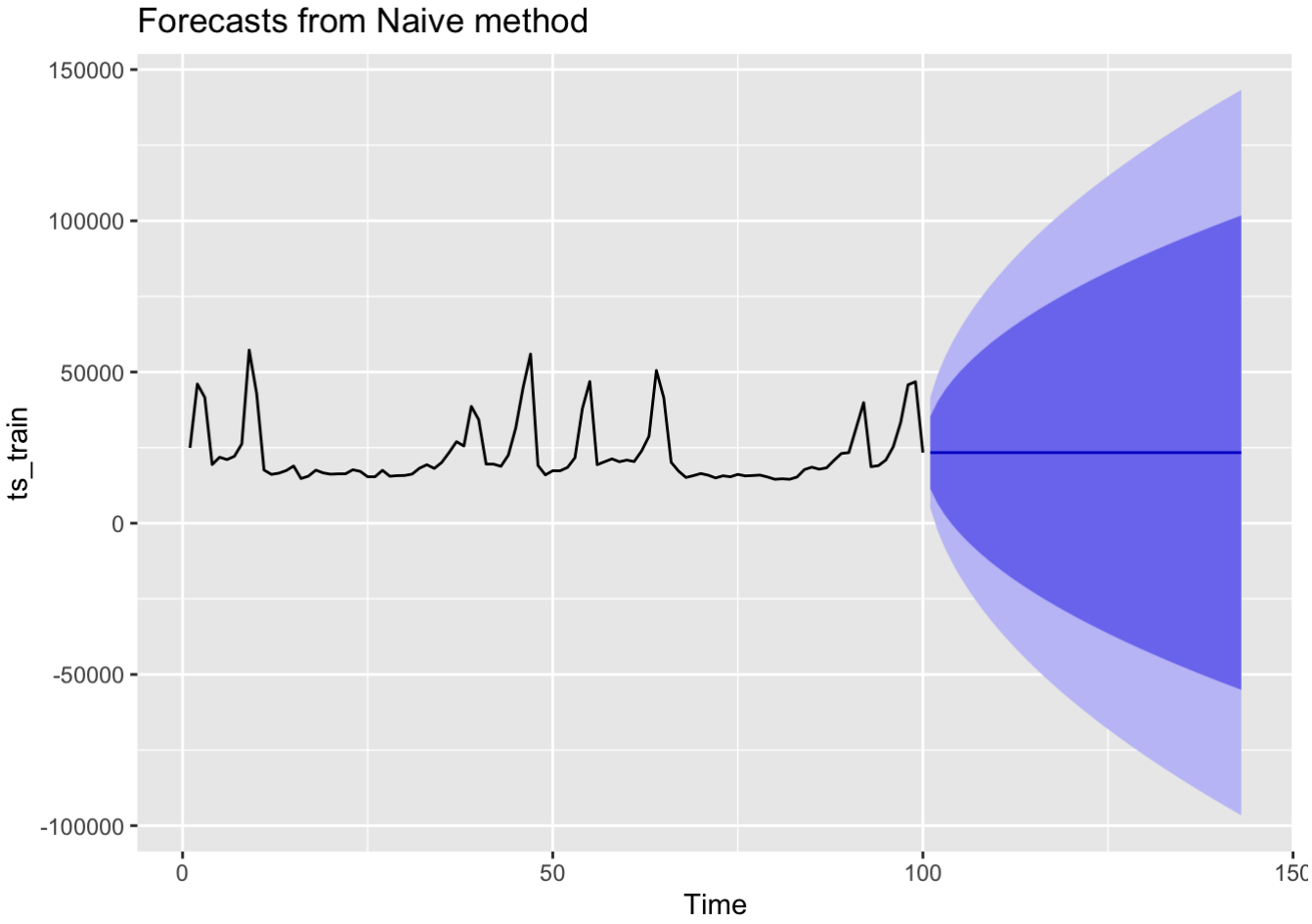

```
Fmean = meanf(ts_train, h=43)
Fmean
```

```
##     Point Forecast   Lo 80    Hi 80    Lo 95    Hi 95
## 101          22979 9753.21 36204.78 2638.265 43319.73
## 102          22979 9753.21 36204.78 2638.265 43319.73
## 103          22979 9753.21 36204.78 2638.265 43319.73
## 104          22979 9753.21 36204.78 2638.265 43319.73
## 105          22979 9753.21 36204.78 2638.265 43319.73
## 106          22979 9753.21 36204.78 2638.265 43319.73
## 107          22979 9753.21 36204.78 2638.265 43319.73
## 108          22979 9753.21 36204.78 2638.265 43319.73
## 109          22979 9753.21 36204.78 2638.265 43319.73
## 110          22979 9753.21 36204.78 2638.265 43319.73
## 111          22979 9753.21 36204.78 2638.265 43319.73
## 112          22979 9753.21 36204.78 2638.265 43319.73
## 113          22979 9753.21 36204.78 2638.265 43319.73
## 114          22979 9753.21 36204.78 2638.265 43319.73
## 115          22979 9753.21 36204.78 2638.265 43319.73
## 116          22979 9753.21 36204.78 2638.265 43319.73
## 117          22979 9753.21 36204.78 2638.265 43319.73
## 118          22979 9753.21 36204.78 2638.265 43319.73
## 119          22979 9753.21 36204.78 2638.265 43319.73
## 120          22979 9753.21 36204.78 2638.265 43319.73
## 121          22979 9753.21 36204.78 2638.265 43319.73
## 122          22979 9753.21 36204.78 2638.265 43319.73
## 123          22979 9753.21 36204.78 2638.265 43319.73
## 124          22979 9753.21 36204.78 2638.265 43319.73
## 125          22979 9753.21 36204.78 2638.265 43319.73
## 126          22979 9753.21 36204.78 2638.265 43319.73
## 127          22979 9753.21 36204.78 2638.265 43319.73
## 128          22979 9753.21 36204.78 2638.265 43319.73
## 129          22979 9753.21 36204.78 2638.265 43319.73
## 130          22979 9753.21 36204.78 2638.265 43319.73
## 131          22979 9753.21 36204.78 2638.265 43319.73
## 132          22979 9753.21 36204.78 2638.265 43319.73
## 133          22979 9753.21 36204.78 2638.265 43319.73
## 134          22979 9753.21 36204.78 2638.265 43319.73
## 135          22979 9753.21 36204.78 2638.265 43319.73
## 136          22979 9753.21 36204.78 2638.265 43319.73
## 137          22979 9753.21 36204.78 2638.265 43319.73
## 138          22979 9753.21 36204.78 2638.265 43319.73
## 139          22979 9753.21 36204.78 2638.265 43319.73
## 140          22979 9753.21 36204.78 2638.265 43319.73
## 141          22979 9753.21 36204.78 2638.265 43319.73
## 142          22979 9753.21 36204.78 2638.265 43319.73
## 143          22979 9753.21 36204.78 2638.265 43319.73
```

```
accuracy(Fmean)
```

```
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 1.401253e-12 10149.26 7510.949 -13.71342 31.38612 1.467225
##                    ACF1
## Training set 0.5815596
```

```
autoplot(naive(ts_train, h = 43))
```



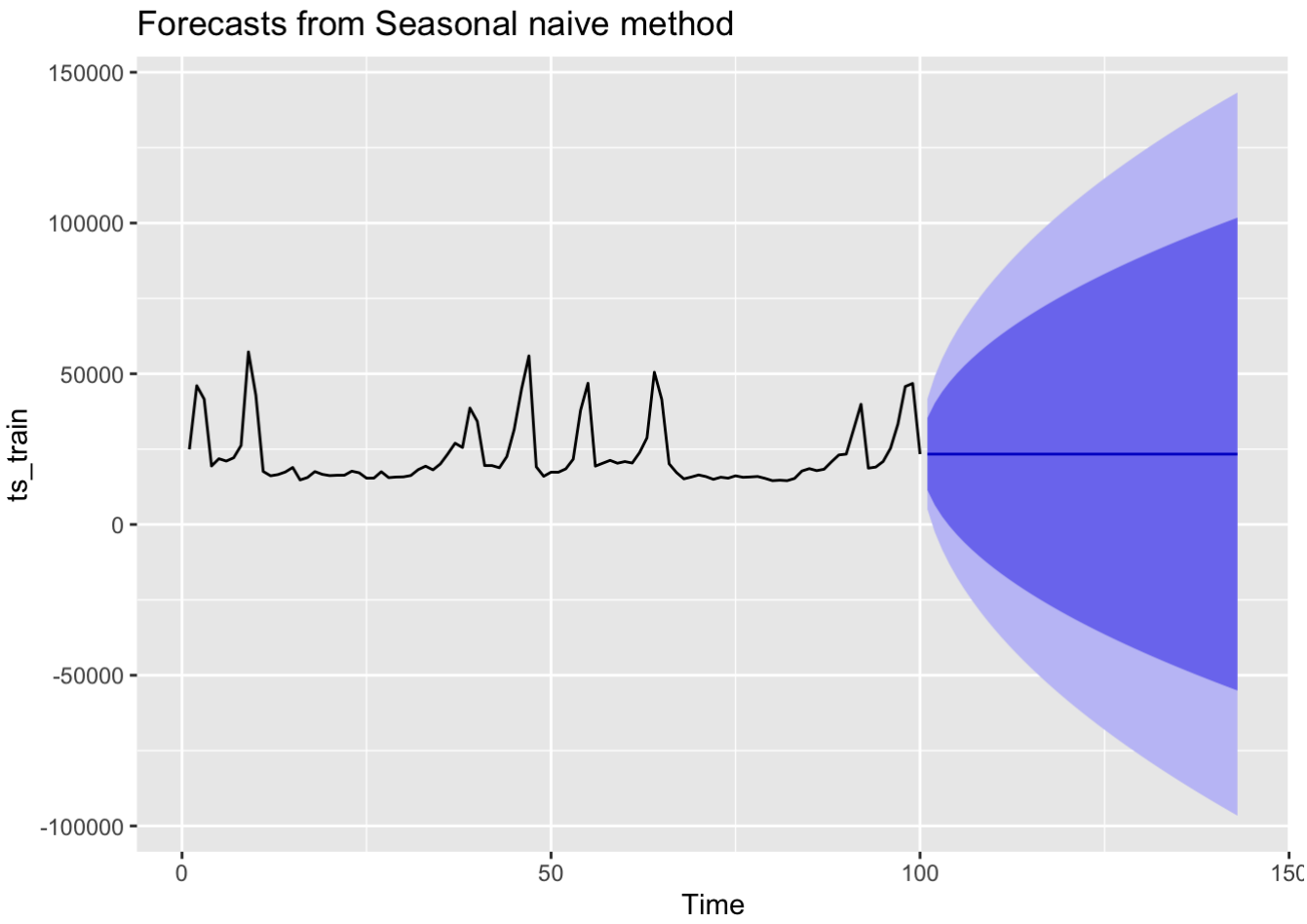

```
Fnaive = naive(ts_train, h=43)
Fnaive
```

```
##     Point Forecast       Lo 80     Hi 80      Lo 95     Hi 95
## 101       23350.88  11394.8838  35306.88   5065.762  41636.00
## 102       23350.88   6442.5480  40259.21  -2508.182  49209.94
## 103       23350.88   2642.4871  44059.27  -8319.874  55021.63
## 104       23350.88   -561.1125  47262.87 -13219.357  59921.12
## 105       23350.88  -3383.5403  50085.30 -17535.888  64237.65
## 106       23350.88  -5935.2101  52636.97 -21438.330  68140.09
## 107       23350.88  -8281.7127  54983.47 -25026.996  71728.76
## 108       23350.88 -10465.7840  57167.54 -28367.245  75069.00
## 109       23350.88 -12517.1087  59218.87 -31504.475  78206.23
## 110       23350.88 -14457.2998  61159.06 -34471.741  81173.50
## 111       23350.88 -16302.6735  63004.43 -37293.997  83995.76
## 112       23350.88 -18065.9058  64767.67 -39990.628  86692.39
## 113       23350.88 -19757.0774  66458.84 -42577.052  89278.81
## 114       23350.88 -21384.3616  68086.12 -45065.768  91767.53
## 115       23350.88 -22954.4943  69656.25 -47467.079  94168.84
## 116       23350.88 -24473.1049  71174.86 -49789.593  96491.35
## 117       23350.88 -25944.9553  72646.72 -52040.594  98742.35
## 118       23350.88 -27374.1161  74075.88 -54226.307 100928.07
## 119       23350.88 -28764.0993  75465.86 -56352.103 103053.86
## 120       23350.88 -30117.9606  76819.72 -58422.655 105124.42
## 121       23350.88 -31438.3777  78140.14 -60442.059 107143.82
## 122       23350.88 -32727.7131  79429.47 -62413.927 109115.69
## 123       23350.88 -33988.0636  80689.82 -64341.467 111043.23
## 124       23350.88 -35221.3003  81923.06 -66227.540 112929.30
## 125       23350.88 -36429.1011  83130.86 -68074.712 114776.47
## 126       23350.88 -37612.9781  84314.74 -69885.295 116587.06
## 127       23350.88 -38774.2988  85476.06 -71661.382 118363.14
## 128       23350.88 -39914.3054  86616.07 -73404.872 120106.63
## 129       23350.88 -41034.1301  87735.89 -75117.496 121819.26
## 130       23350.88 -42134.8083  88836.57 -76800.838 123502.60
## 131       23350.88 -43217.2897  89919.05 -78456.350 125158.11
## 132       23350.88 -44282.4481  90984.21 -80085.369 126787.13
## 133       23350.88 -45331.0893  92032.85 -81689.128 128390.89
## 134       23350.88 -46363.9589  93065.72 -83268.765 129970.53
## 135       23350.88 -47381.7476  94083.51 -84825.339 131527.10
## 136       23350.88 -48385.0974  95086.86 -86359.830 133061.59
## 137       23350.88 -49374.6059  96076.37 -87873.153 134574.91
## 138       23350.88 -50350.8306  97052.59 -89366.159 136067.92
## 139       23350.88 -51314.2925  98016.05 -90839.647 137541.41
## 140       23350.88 -52265.4796  98967.24 -92294.362 138996.12
## 141       23350.88 -53204.8492  99906.61 -93731.004 140432.76
## 142       23350.88 -54132.8313 100834.59 -95150.230 141851.99
## 143       23350.88 -55049.8303 101751.59 -96552.659 143254.42
```

```
accuracy((Fnaive))
```

```
##                      ME     RMSE      MAE       MPE     MAPE MASE       ACF1
## Training set -15.89515 9329.313 5119.153 -5.054183 20.10871    1 0.01965934
```

```
autoplot(snaive(ts_train, h = 43))
```




```
Fsnaive = snaive(ts_train, h=43)
Fsnaive
```

```
##     Point Forecast       Lo 80     Hi 80      Lo 95     Hi 95
## 101       23350.88  11394.8838  35306.88   5065.762  41636.00
## 102       23350.88   6442.5480  40259.21  -2508.182  49209.94
## 103       23350.88   2642.4871  44059.27  -8319.874  55021.63
## 104       23350.88   -561.1125  47262.87 -13219.357  59921.12
## 105       23350.88  -3383.5403  50085.30 -17535.888  64237.65
## 106       23350.88  -5935.2101  52636.97 -21438.330  68140.09
## 107       23350.88  -8281.7127  54983.47 -25026.996  71728.76
## 108       23350.88 -10465.7840  57167.54 -28367.245  75069.00
## 109       23350.88 -12517.1087  59218.87 -31504.475  78206.23
## 110       23350.88 -14457.2998  61159.06 -34471.741  81173.50
## 111       23350.88 -16302.6735  63004.43 -37293.997  83995.76
## 112       23350.88 -18065.9058  64767.67 -39990.628  86692.39
## 113       23350.88 -19757.0774  66458.84 -42577.052  89278.81
## 114       23350.88 -21384.3616  68086.12 -45065.768  91767.53
## 115       23350.88 -22954.4943  69656.25 -47467.079  94168.84
## 116       23350.88 -24473.1049  71174.86 -49789.593  96491.35
## 117       23350.88 -25944.9553  72646.72 -52040.594  98742.35
## 118       23350.88 -27374.1161  74075.88 -54226.307 100928.07
## 119       23350.88 -28764.0993  75465.86 -56352.103 103053.86
## 120       23350.88 -30117.9606  76819.72 -58422.655 105124.42
## 121       23350.88 -31438.3777  78140.14 -60442.059 107143.82
## 122       23350.88 -32727.7131  79429.47 -62413.927 109115.69
## 123       23350.88 -33988.0636  80689.82 -64341.467 111043.23
## 124       23350.88 -35221.3003  81923.06 -66227.540 112929.30
## 125       23350.88 -36429.1011  83130.86 -68074.712 114776.47
## 126       23350.88 -37612.9781  84314.74 -69885.295 116587.06
## 127       23350.88 -38774.2988  85476.06 -71661.382 118363.14
## 128       23350.88 -39914.3054  86616.07 -73404.872 120106.63
## 129       23350.88 -41034.1301  87735.89 -75117.496 121819.26
## 130       23350.88 -42134.8083  88836.57 -76800.838 123502.60
## 131       23350.88 -43217.2897  89919.05 -78456.350 125158.11
## 132       23350.88 -44282.4481  90984.21 -80085.369 126787.13
## 133       23350.88 -45331.0893  92032.85 -81689.128 128390.89
## 134       23350.88 -46363.9589  93065.72 -83268.765 129970.53
## 135       23350.88 -47381.7476  94083.51 -84825.339 131527.10
## 136       23350.88 -48385.0974  95086.86 -86359.830 133061.59
## 137       23350.88 -49374.6059  96076.37 -87873.153 134574.91
## 138       23350.88 -50350.8306  97052.59 -89366.159 136067.92
## 139       23350.88 -51314.2925  98016.05 -90839.647 137541.41
## 140       23350.88 -52265.4796  98967.24 -92294.362 138996.12
## 141       23350.88 -53204.8492  99906.61 -93731.004 140432.76
## 142       23350.88 -54132.8313 100834.59 -95150.230 141851.99
## 143       23350.88 -55049.8303 101751.59 -96552.659 143254.42
```

```
accuracy(Fsnaive)
```

```
##                      ME     RMSE      MAE       MPE     MAPE MASE       ACF1
## Training set -15.89515 9329.313 5119.153 -5.054183 20.10871    1 0.01965934
```

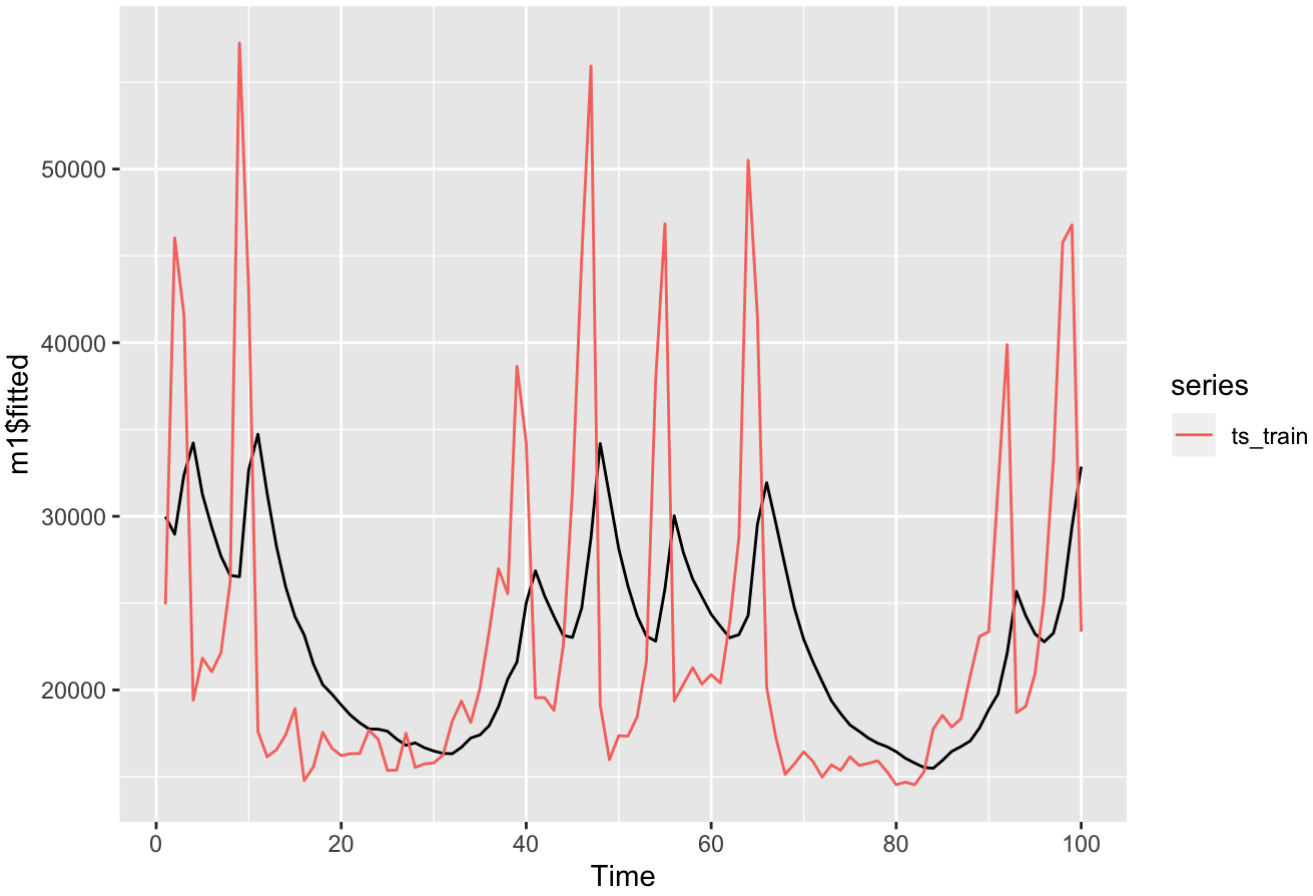
It shows that with the 3 methods, it does not really help out with forecasting since it gives us a general average and does not model the holiday sales well. We also proceeded to model our data with Simple Exponential smoothing and Holt's methods. We wanted to use Holt-Winter's but couldnt since the frequency of our data is not greater than 1.

Simple Exponential Smoothing & Holt method

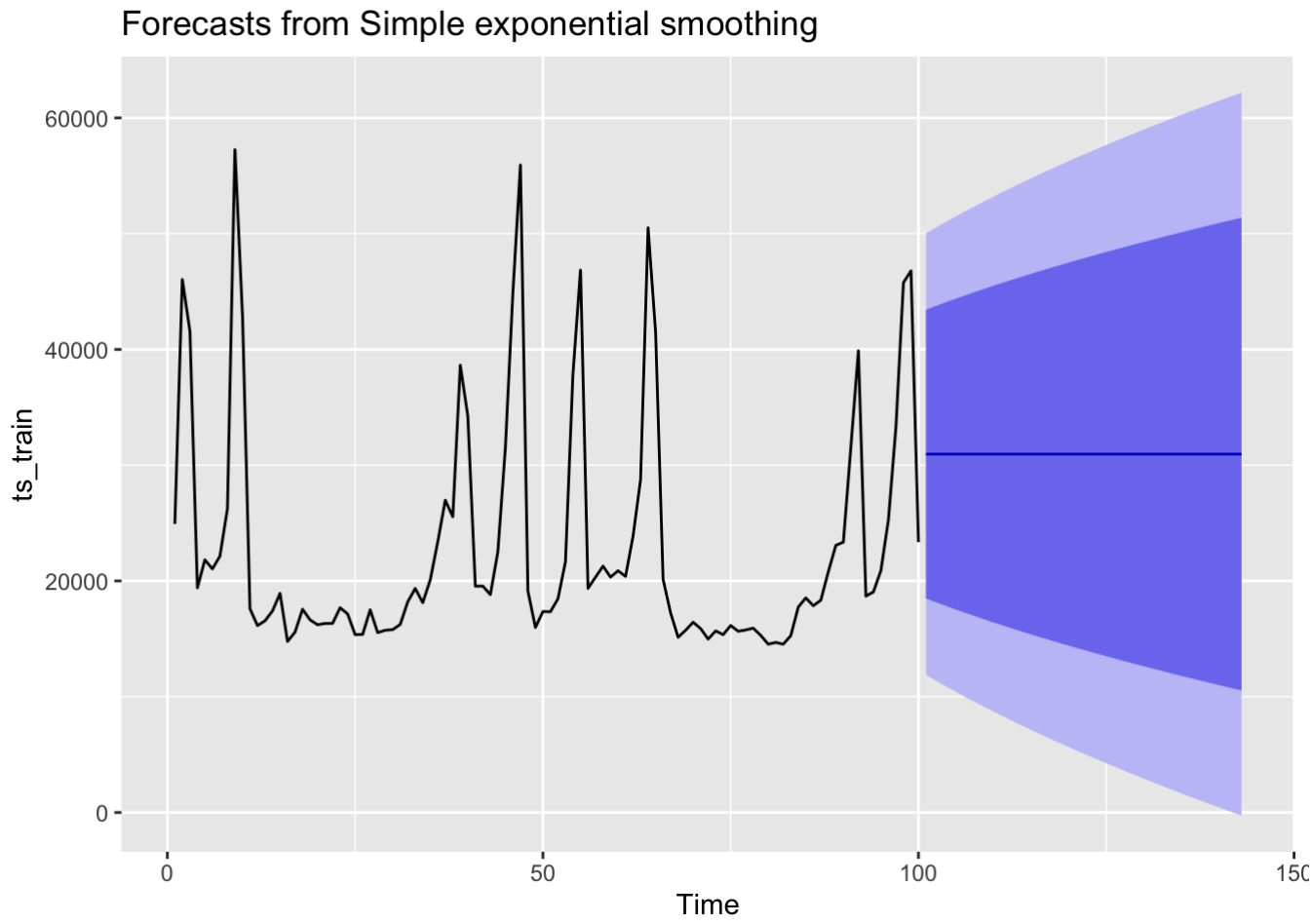
```
m1 = ses(ts_train, h = 43,alpha = 0.2) # Set middle value for smoothing parameter
m1$model
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = ts_train, h = 43, alpha = 0.2)
##
##   Smoothing parameters:
##     alpha = 0.2
##
##   Initial states:
##     l = 29979.4772
##
##   sigma:  9731.863
##
##      AIC     AICc      BIC
## 2299.129 2299.253 2304.339
```

```
autoplot(m1$fitted) + autolayer(ts_train)
```


m1$fitted
50000
40000
30000
20000
0
20
40
60
80
100
Time
series
ts_train

```
autoplot(m1)
```



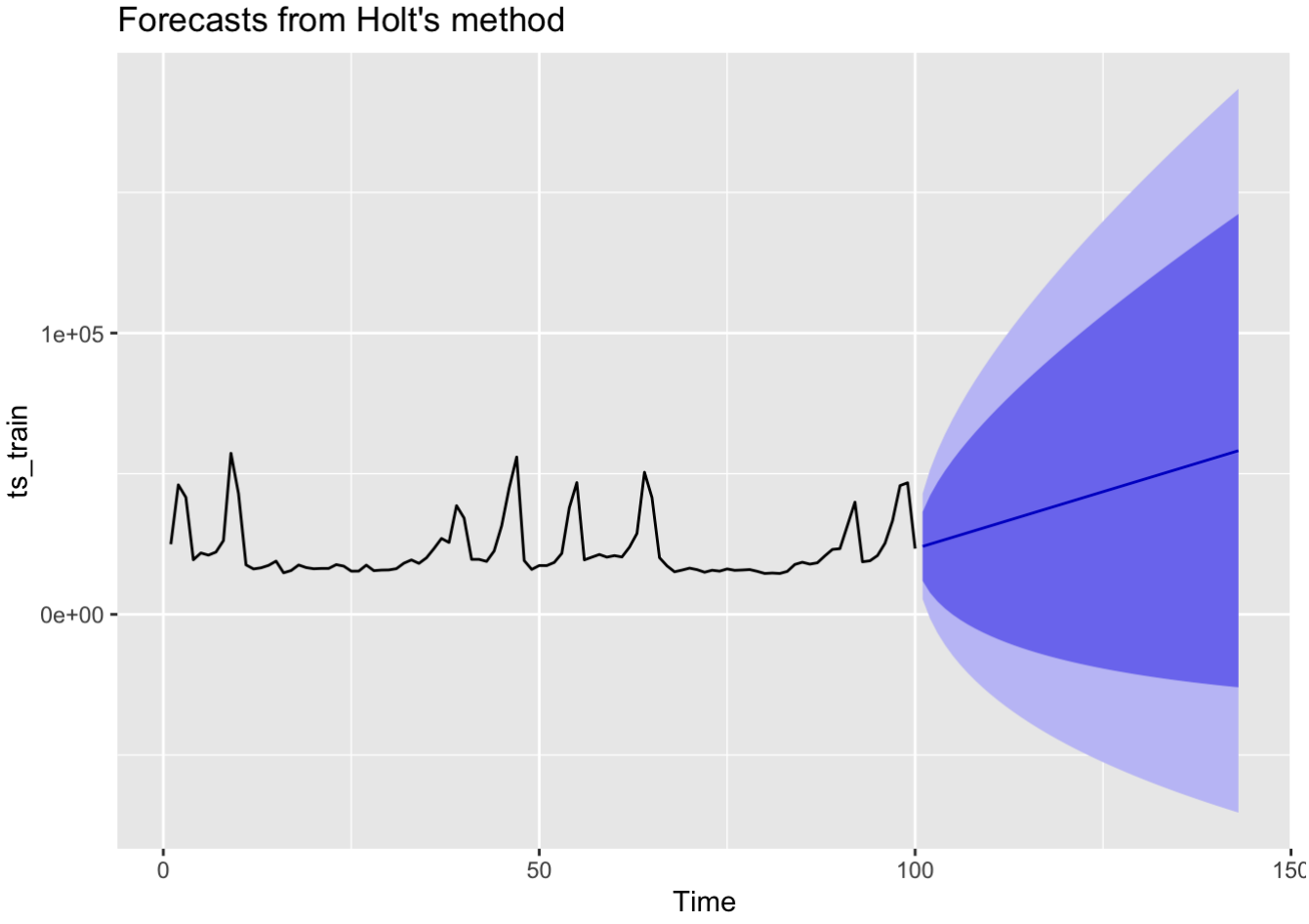

```
accuracy(m1)
```

```
##                     ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
## Training set 48.95848 9634.053 7124.46 -10.41808 29.26813 1.391726 0.4390652
```

```
m2 = holt(ts_train, h=43, initial = 'optimal')
m2$model
```

```
## Holt's method
##
## Call:
##  holt(y = ts_train, h = 43, initial = "optimal")
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.0023
##
##   Initial states:
##     l = 29293.2986
##     b = 1031.7525
##
##   sigma:  9547.585
##
##      AIC     AICc      BIC
## 2299.244 2299.882 2312.269
```

```
autoplot(m2)
```




```
accuracy(m2)
```

```
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -969.9662 9354.685 5127.211 -9.665174 20.67744 1.001574
##                       ACF1
## Training set 0.009562296
```

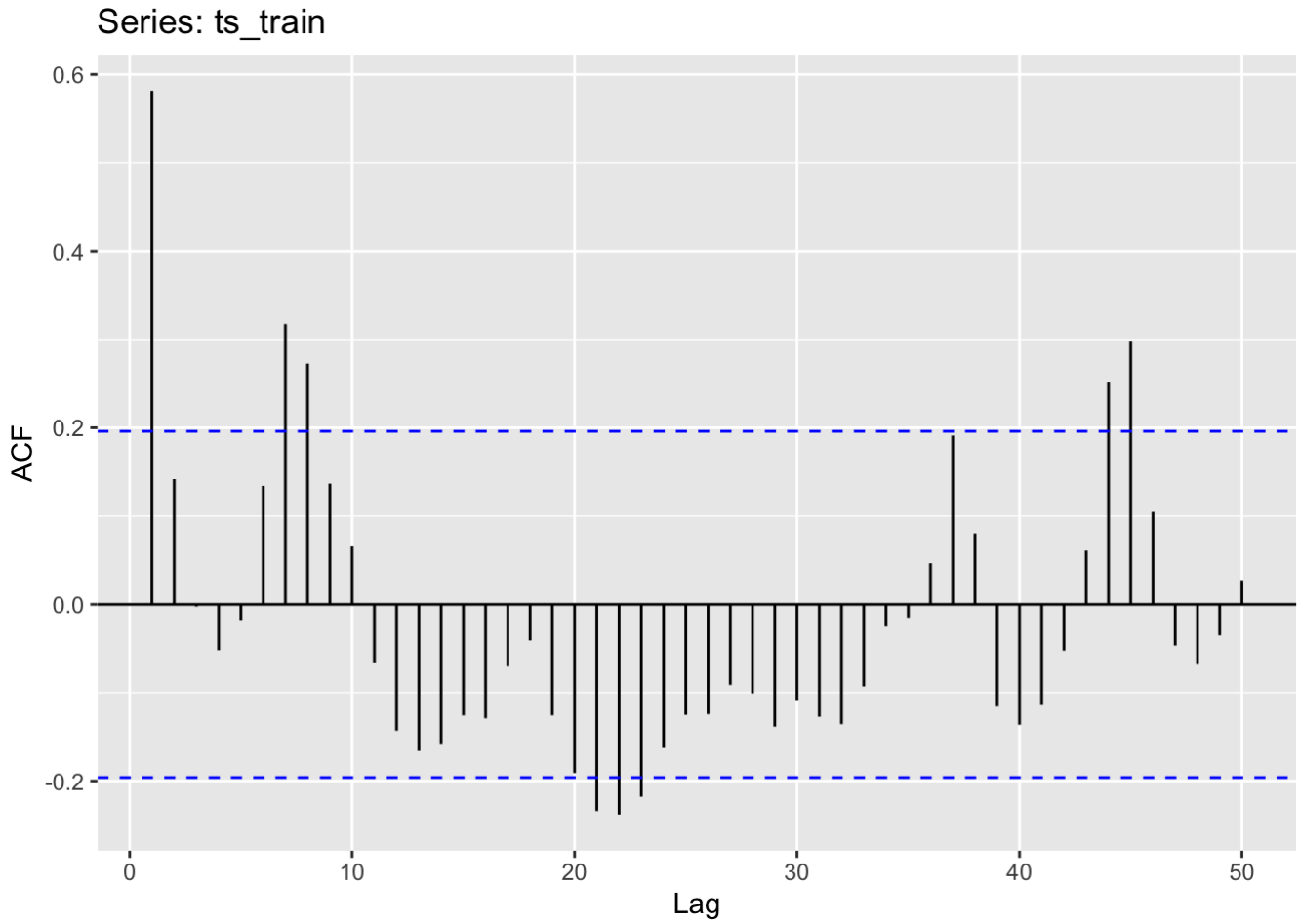
Using the SES and Holt's method, we can record the AIC and BIC values from the generated models, which are 2299.129/2304.339 (SES) and 2299.244 and 2312.269 (Holt's Method). Judging by the numbers, the SES model performed better, but we see that Holt's model attempts to model the trend in it's forecast. Therefore, we believe that we need a more complex model to analyze our data

We next attempt to use ARIMA/SARIMA models to fit our data. We believe that the models generated would be better, validated by the AIC/BIC values. We would also perform residual analysis on our data to to find the best fitted model of different techniques.

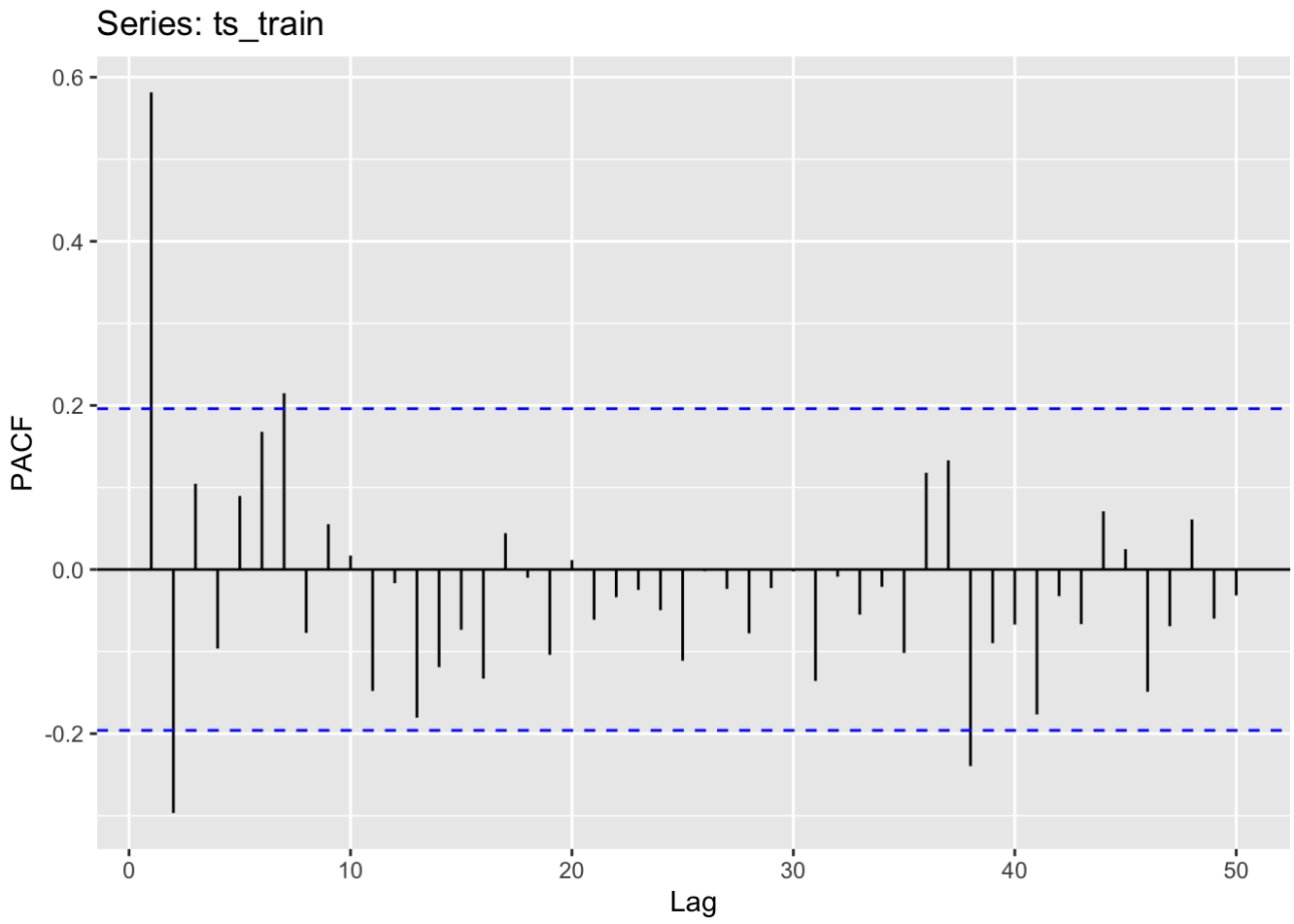
Analyzing with AR, MA, ARIMA, SARIMA

Check Stationarity of Our Data and identifying order

```
ggAcf(ts_train, lag.max = 50)
```




```
ggPacf(ts_train, lag.max = 50)
```




```
Box.test(ts_train, type = 'L')
```

```
##
##  Box-Ljung test
##
## data:  ts_train
## X-squared = 34.846, df = 1, p-value = 3.568e-09
```

```
eacf(ts_train)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o o o o o x x o o o  o  o  o
## 1 x x o o o o x o o o o  o  o  o
## 2 x x x o o o o o o o o  o  o  o
## 3 x o o o o o o o o o o  o  o  o
## 4 x o o o o o o o o o o  o  o  o
## 5 x o o o o o o o o o o  o  o  o
## 6 x o o x x o o o o o o  o  o  o
## 7 x o o x o x o o o o o  o  o  o
```

```
auto.arima(ts_train)
```

```
## Series: ts_train
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ma1       mean
##       0.2562  0.5554  22854.351
## s.e.  0.1341  0.1155   1611.326
##
## sigma^2 estimated as 62038792:  log likelihood=-1037.89
## AIC=2083.77   AICc=2084.19   BIC=2094.19
```

We identify serial correlation of our data through the Acf and Pacf plot. The Ljung-Box test shows that the time series is not normal. Through the plots, we can identify the orders that we would use for the AR and MA models. We identify p to be 2 and q to be 1. The eacf on the other hand shows that the simplest order of p and q should be 0 and 1 respectively, whereas the auto.arima function shows the best order for the best AIC/BIC value is 1,1, although it includes a mean. We decided to run models against a couple of parameters to see which one gives us a better result.

AR Modeling and MA Modeling

```
AR_model = Arima(ts_train, order = c(1,0,0), include.constant = F) # 1 better than 2
AR_model
```

```
## Series: ts_train
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.9305
## s.e.  0.0345
##
## sigma^2 estimated as 84827768:  log likelihood=-1055.2
## AIC=2114.4   AICc=2114.53   BIC=2119.62
```

```
FcastAR = forecast(AR_model)
FcastAR
```

```
##     Point Forecast       Lo 80    Hi 80      Lo 95    Hi 95
## 101       21728.35   9925.0079 33531.70   3676.694 39780.01
## 102       20218.57   4095.6168 36341.52  -4439.357 44876.49
## 103       18813.69   -275.5387 37902.91 -10380.764 48008.14
## 104       17506.43  -3820.4748 38833.33 -15110.254 50123.10
## 105       16290.00  -6799.8985 39379.89 -19022.951 51602.95
## 106       15158.09  -9356.1028 39672.29 -22333.135 52649.32
## 107       14104.84 -11578.8761 39788.55 -25175.014 53384.69
## 108       13124.77 -13530.1651 39779.70 -27640.435 53889.97
## 109       12212.80 -15255.3483 39680.95 -29796.108 54221.71
## 110       11364.20 -16789.1098 39517.51 -31692.571 54420.97
```

```
accuracy(FcastAR)
```

```
##                     ME     RMSE      MAE      MPE     MAPE     MASE       ACF1
## Training set 1656.012 9164.032 5568.944 2.589265 21.87665 1.087864 0.04220328
```

```
coeftest(AR_model)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ar1 0.930515   0.034479  26.988 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
MA_model = Arima(ts_train, order = c(0,0,2), include.constant = F) # 2 better than 1
MA_model
```

```
## Series: ts_train
## ARIMA(0,0,2) with zero mean
##
## Coefficients:
##          ma1     ma2
##       1.2007  0.5202
## s.e.  0.1034  0.0649
##
## sigma^2 estimated as 147418909:  log likelihood=-1082.13
## AIC=2170.25   AICc=2170.5   BIC=2178.07
```

```
FcastMA = forecast(MA_model)
FcastMA
```

```
##     Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 101       1753.071 -13807.04 17313.18 -22044.07 25550.21
## 102      -2762.146 -27075.59 21551.30 -39946.35 34422.05
## 103          0.000 -25625.50 25625.50 -39190.82 39190.82
## 104          0.000 -25625.50 25625.50 -39190.82 39190.82
## 105          0.000 -25625.50 25625.50 -39190.82 39190.82
## 106          0.000 -25625.50 25625.50 -39190.82 39190.82
## 107          0.000 -25625.50 25625.50 -39190.82 39190.82
## 108          0.000 -25625.50 25625.50 -39190.82 39190.82
## 109          0.000 -25625.50 25625.50 -39190.82 39190.82
## 110          0.000 -25625.50 25625.50 -39190.82 39190.82
```

```
accuracy(FcastMA)
```

```
##                     ME     RMSE      MAE      MPE     MAPE    MASE       ACF1
## Training set 8408.406 12019.59 9710.882 36.10883 42.34913 1.89697 -0.3819234
```

```
coeftest(MA_model)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1  1.20065    0.10336 11.6161 < 2.2e-16 ***
## ma2  0.52022    0.06487  8.0193 1.063e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ARMA_model = Arima(ts_train, order = c(2,0,2), include.constant = F)# 2,2 is better
ARMA_model
```

```
## Series: ts_train
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1      ma2
##       1.2818  -0.2818  -0.4506  -0.5383
## s.e.  0.1356   0.1356   0.1196   0.1180
##
## sigma^2 estimated as 63802064:  log likelihood=-1041.06
## AIC=2092.11   AICc=2092.75   BIC=2105.14
```

```
FcastARMA = forecast(ARMA_model)
FcastARMA
```

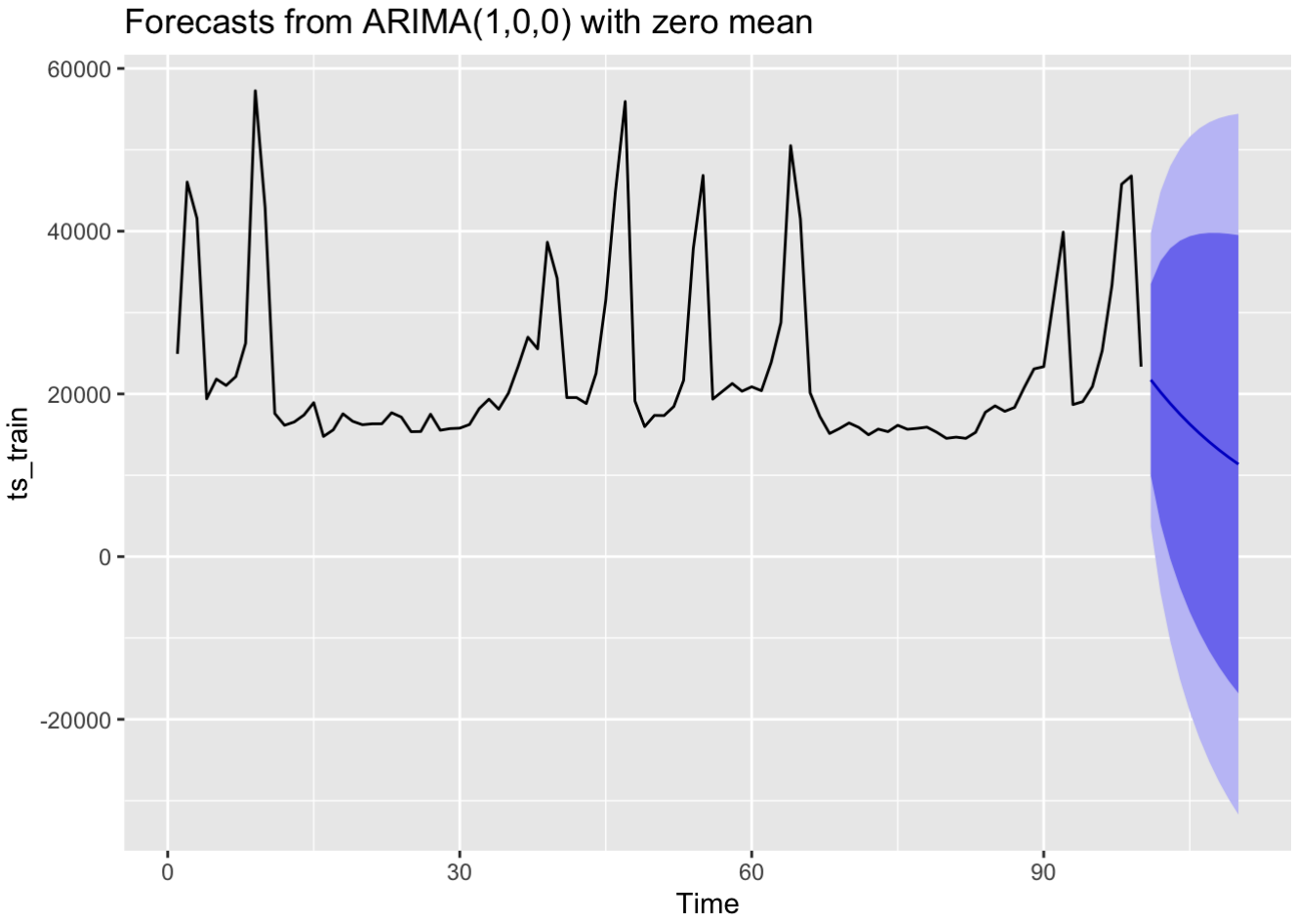
```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 101       16985.35 6727.042 27243.65 1296.623 32674.07
## 102       21113.91 7747.016 34480.79  671.012 41556.80
## 103       22277.18 8661.631 35892.72 1453.997 43100.35
## 104       22604.77 8960.011 36249.52 1736.913 43472.62
## 105       22696.85 9046.451 36347.24 1820.367 43573.33
## 106       22722.56 9070.172 36374.94 1843.037 43602.08
## 107       22729.56 9075.981 36383.14 1848.214 43610.91
## 108       22731.29 9076.724 36385.86 1848.432 43614.15
## 109       22731.54 9076.037 36387.04 1847.251 43615.83
## 110       22731.37 9074.947 36387.79 1845.676 43617.06
```

```
accuracy(FcastARMA)
```

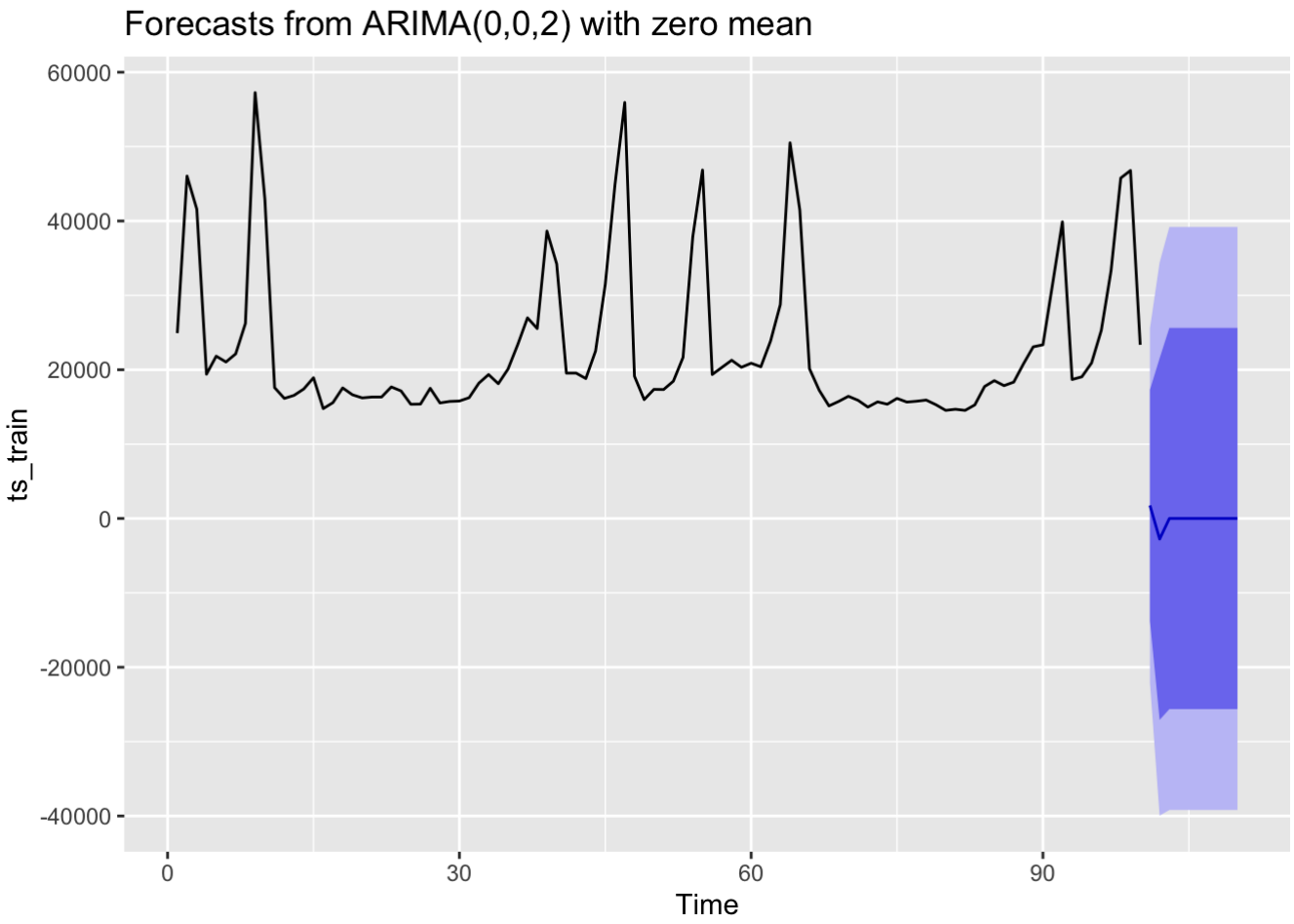
```
##                     ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
## Training set -57.3541 7826.237 5465.461 -7.691301 22.73799 1.067649 0.01171095
```

We can see that after running a couple of parameters, the ARMA model performs best with AIC value of 2092.11 and BIC value of 2105.14. These models also perform better than the SES and Holts methods. The forecast below also proves that ARMA fits our dataset best at this point.

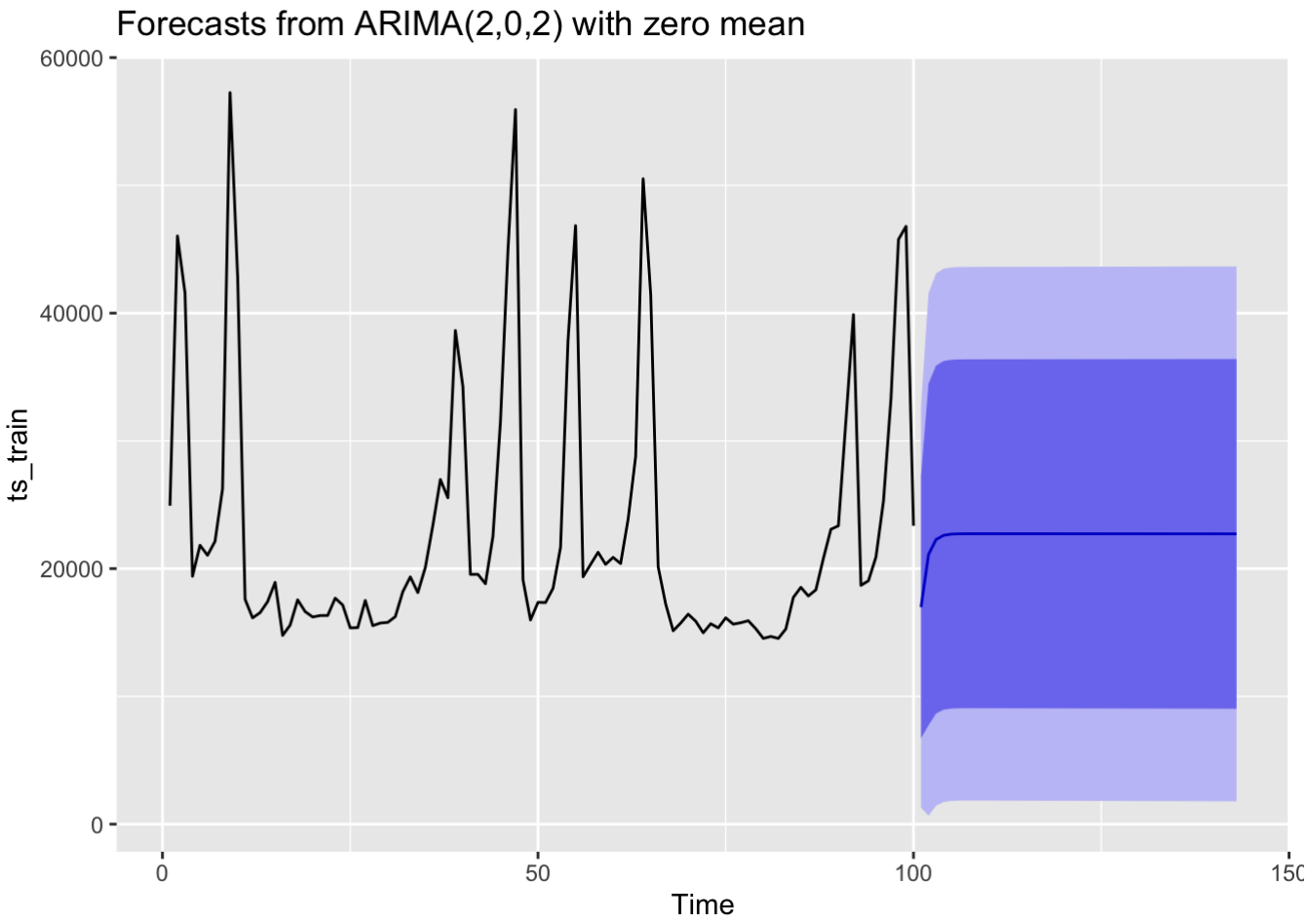
```
forecast(AR_model) %>% autoplot()
```




```
forecast(MA_model) %>% autoplot()
```



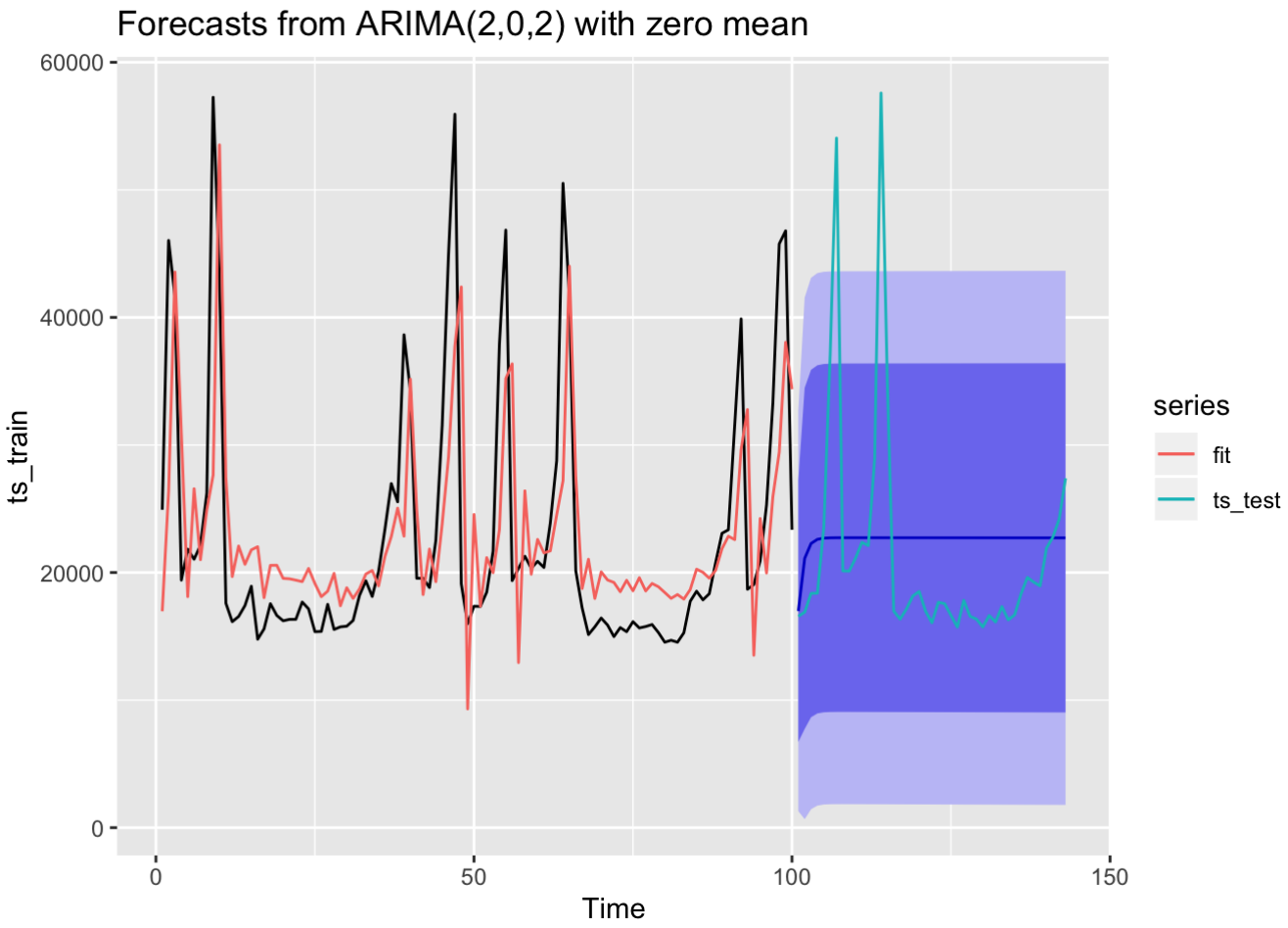

```
forecast(ARMA_model, h = 43) %>% autoplot()
```



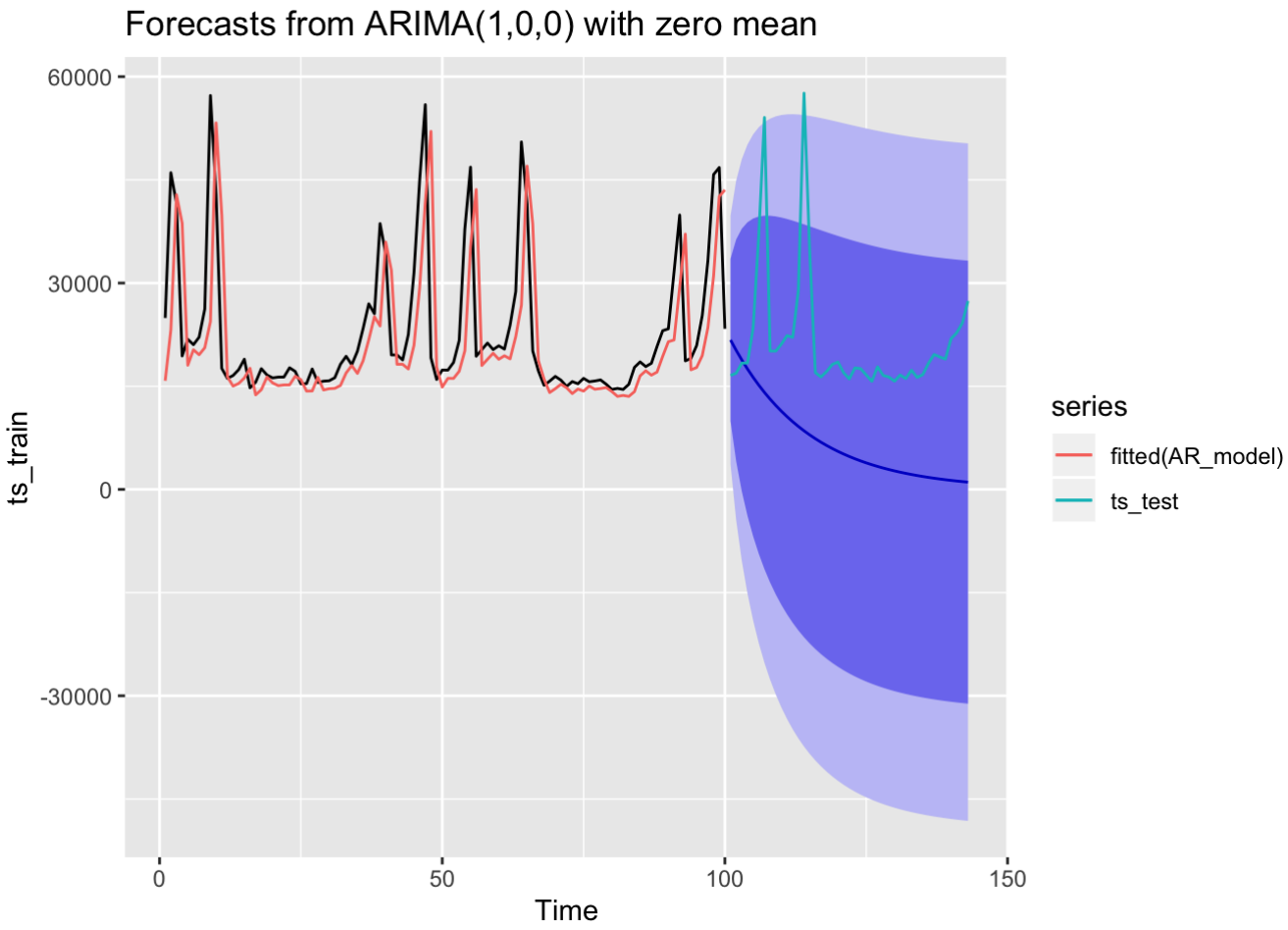

```
fcast = forecast(ARMA_model, h = 43)
fit = fitted(ARMA_model)
accuracy(fcast)
```

```
##                     ME     RMSE      MAE       MPE     MAPE     MASE        ACF1
## Training set -57.3541 7826.237 5465.461 -7.691301 22.73799 1.067649 0.01171095
```

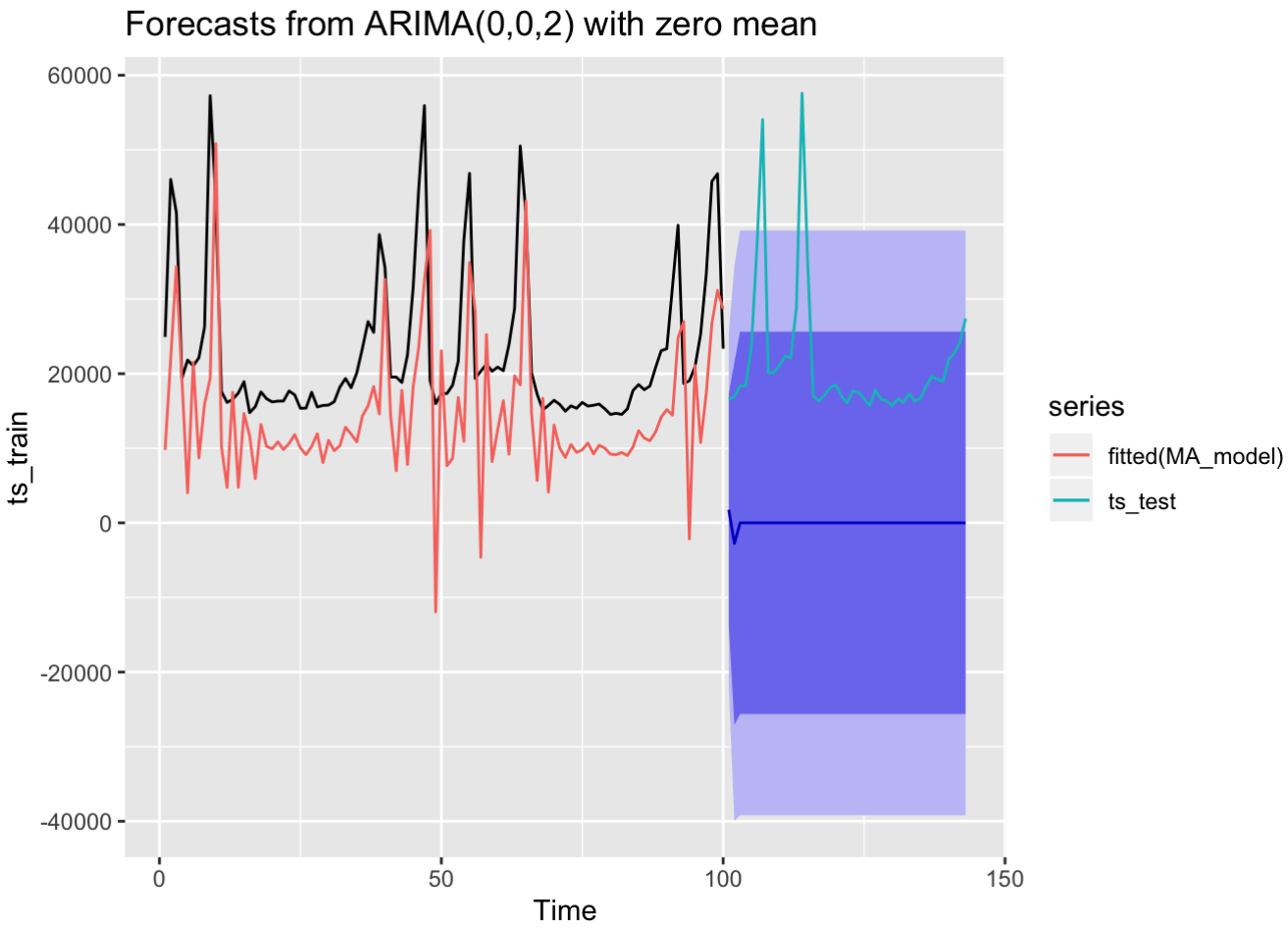
```
autoplot(fcast) + autolayer(fit) + autolayer(ts_test)
```



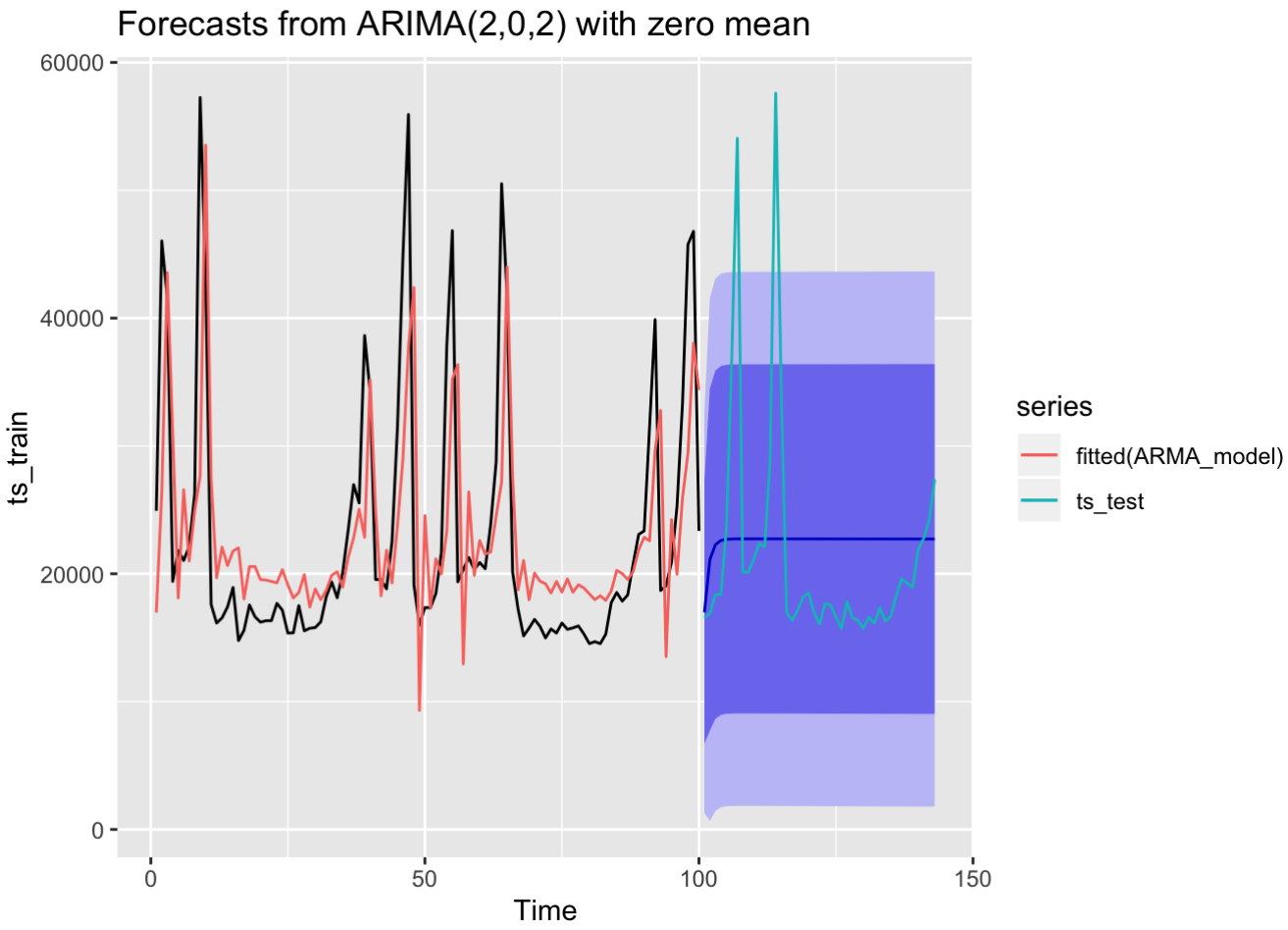

```
autoplot(forecast(AR_model, h = 43)) + autolayer(fitted(AR_model)) + autolayer(ts_test)
```




```
autoplot(forecast(MA_model, h = 43)) + autolayer(fitted(MA_model)) + autolayer(ts_test)
```




```
autoplot(forecast(ARMA_model, h = 43)) + autolayer(fitted(ARMA_model)) + autolayer(ts_te
st)
```



Next, we took a look at the coefficient tests and residual diagnostics for the 3 models.

```
coeftest(AR_model)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ar1 0.930515   0.034479  26.988 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

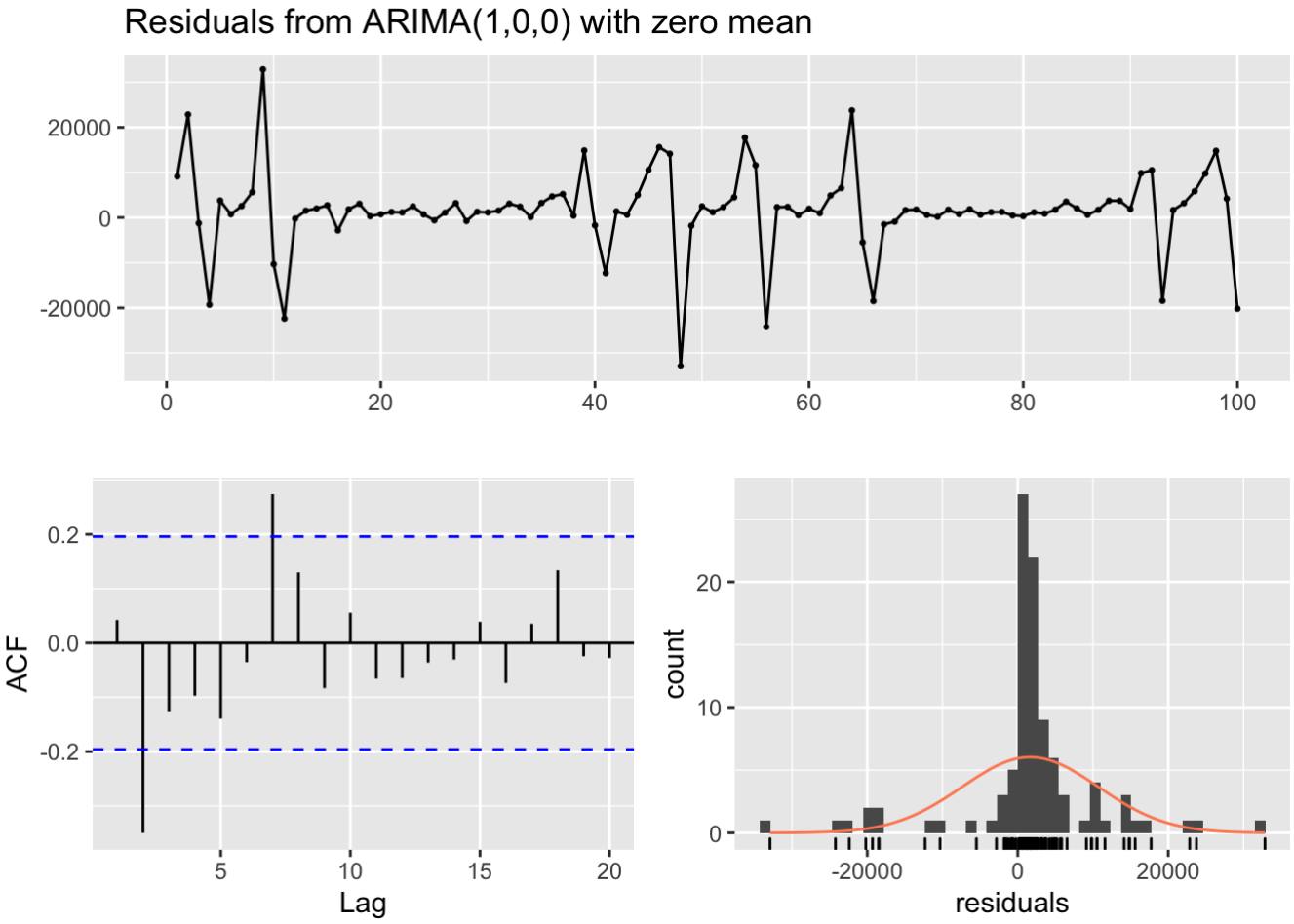
```
coeftest(MA_model)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1  1.20065    0.10336 11.6161 < 2.2e-16 ***
## ma2  0.52022    0.06487  8.0193 1.063e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(ARMA_model)
```

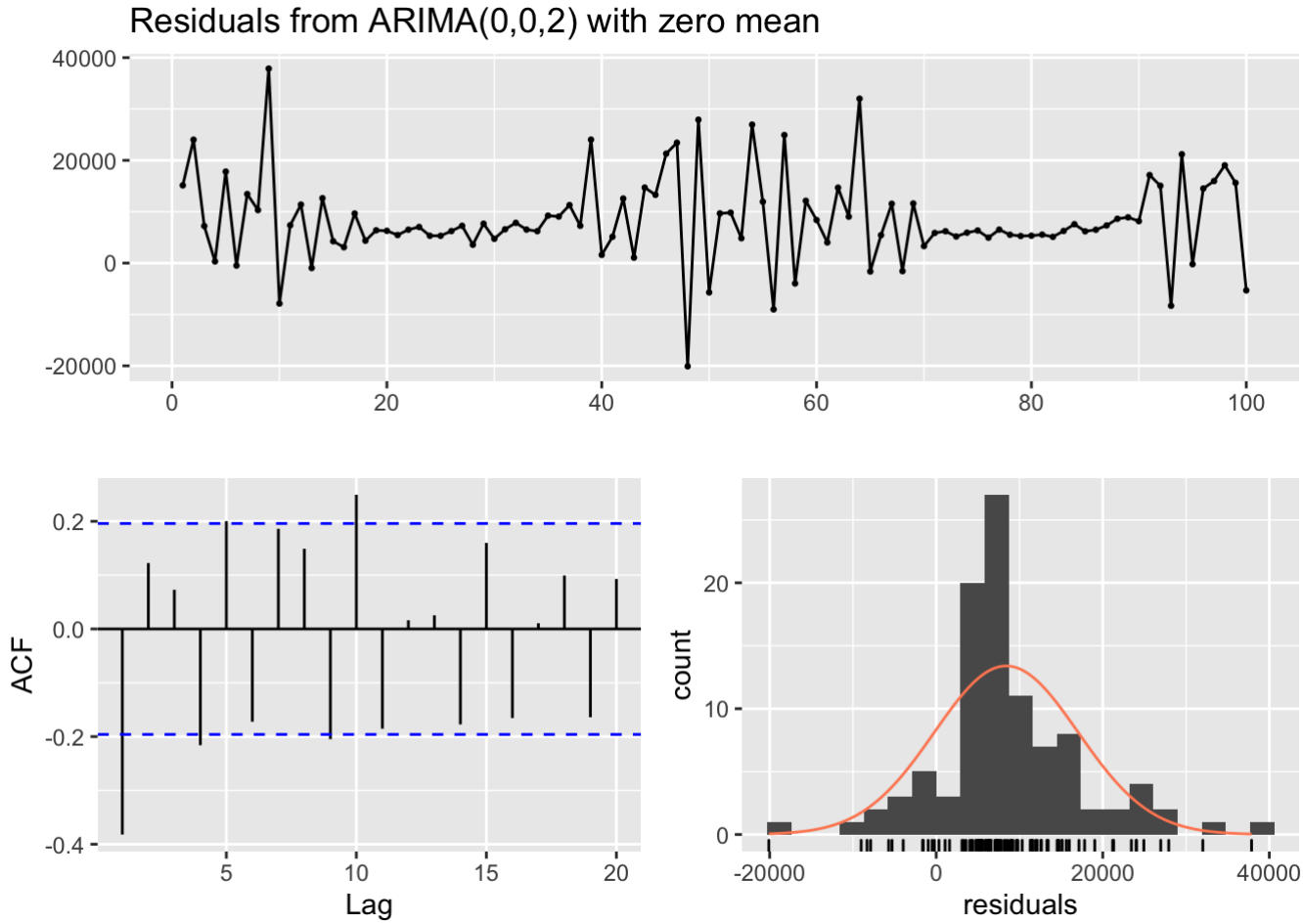
```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ar1  1.28181    0.13560  9.4527 < 2.2e-16 ***
## ar2 -0.28182    0.13560 -2.0783 0.0376862 *
## ma1 -0.45057    0.11956 -3.7686 0.0001642 ***
## ma2 -0.53833    0.11804 -4.5604 5.106e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
checkresiduals(AR_model)
```



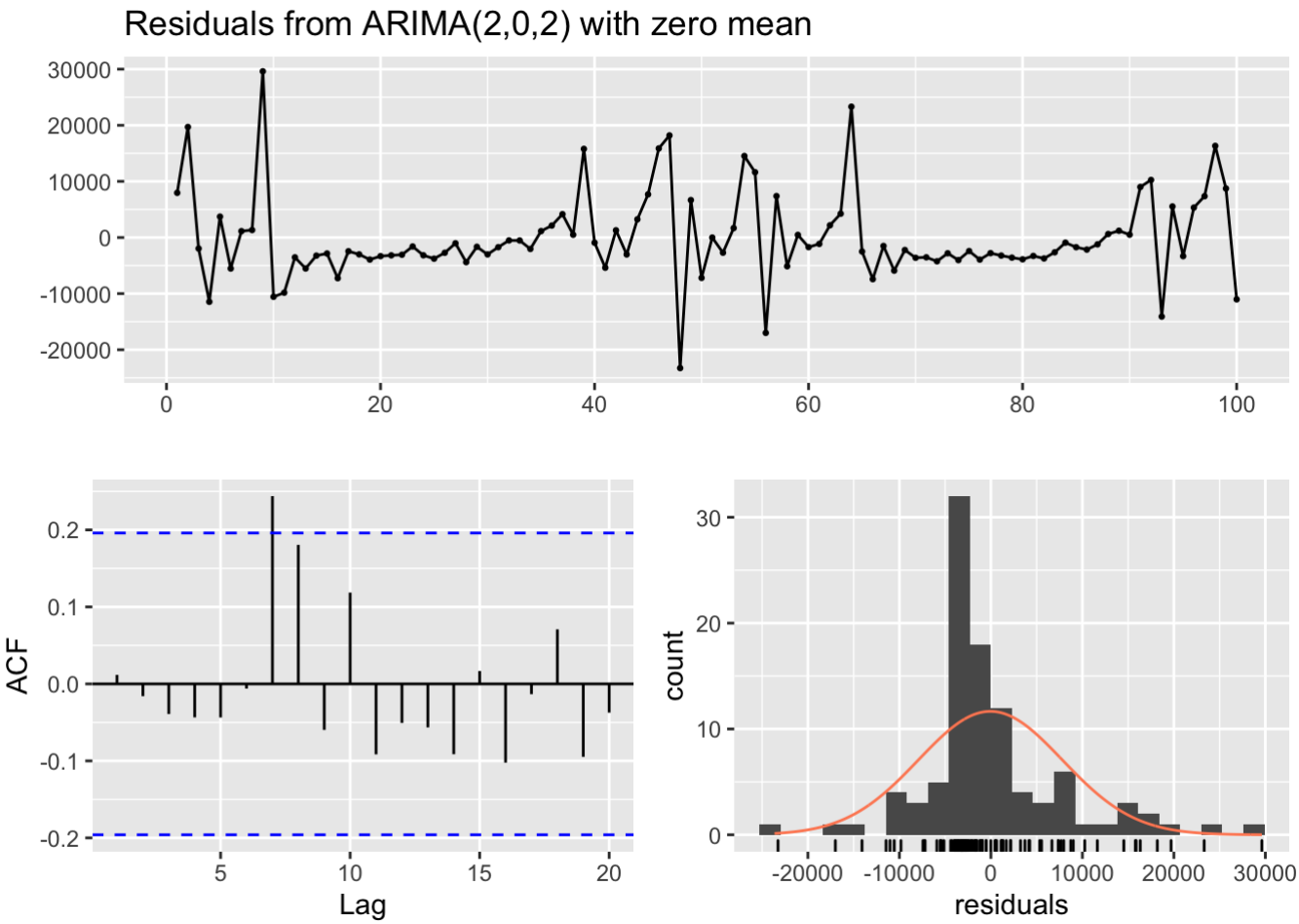

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0) with zero mean
## Q* = 28.998, df = 9, p-value = 0.0006485
##
## Model df: 1.   Total lags used: 10
```

```
checkresiduals(MA_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,2) with zero mean
## Q* = 47.639, df = 8, p-value = 1.158e-07
##
## Model df: 2.   Total lags used: 10
```

```
checkresiduals(ARMA_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,2) with zero mean
## Q* = 12.747, df = 6, p-value = 0.04723
##
## Model df: 4.   Total lags used: 10
```

```
Box.test(AR_model$residuals, lag = 50, type = 'L')
```

```
##
##  Box-Ljung test
##
## data:  AR_model$residuals
## X-squared = 97.561, df = 50, p-value = 6.601e-05
```

```
Box.test(MA_model$residuals, lag = 50, type = 'L')
```

```
##
##  Box-Ljung test
##
## data:  MA_model$residuals
## X-squared = 145.7, df = 50, p-value = 2.731e-11
```

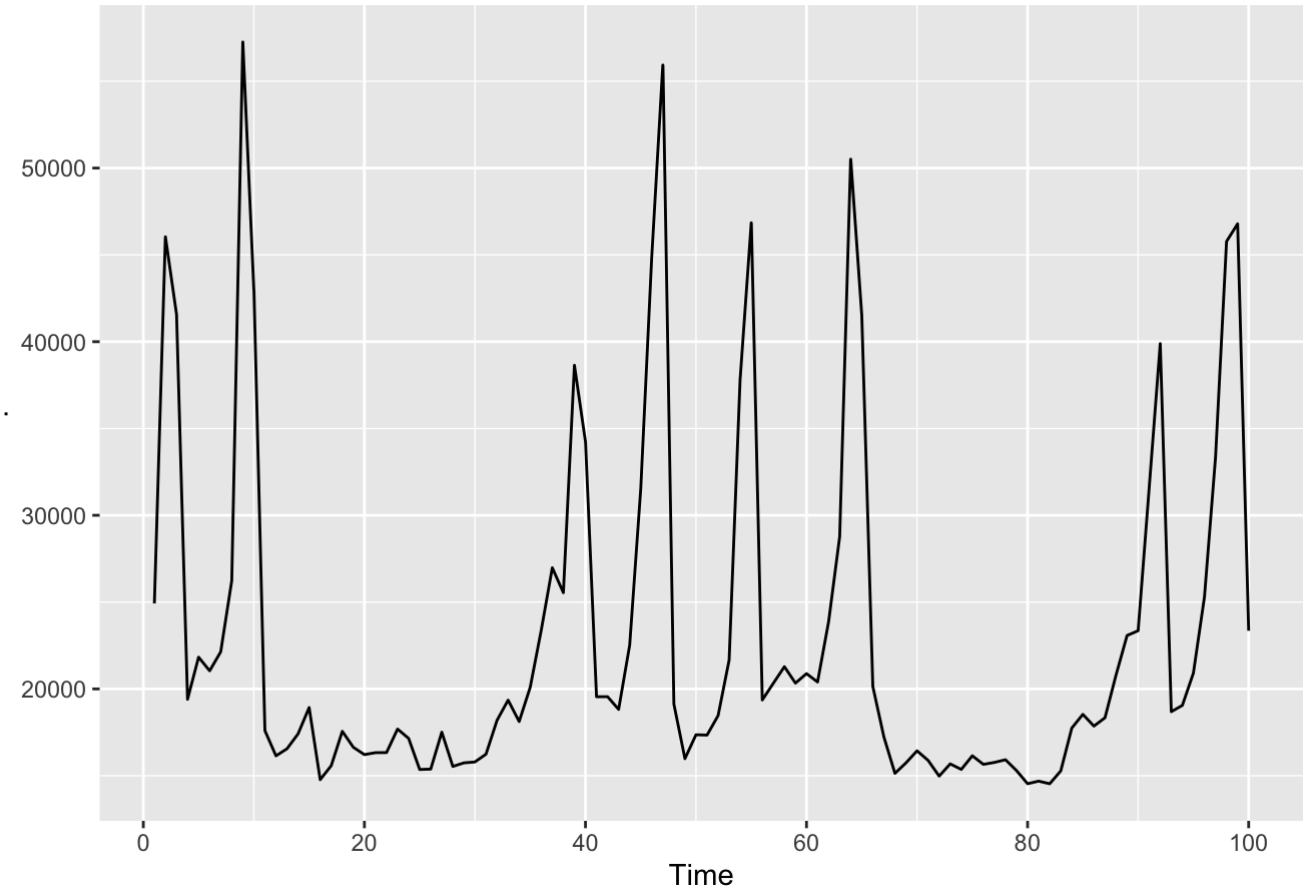
```
Box.test(ARMA_model$residuals, lag = 50, type = 'L')
```

```
##
##  Box-Ljung test
##
## data:  ARMA_model$residuals
## X-squared = 65.493, df = 50, p-value = 0.06966
```

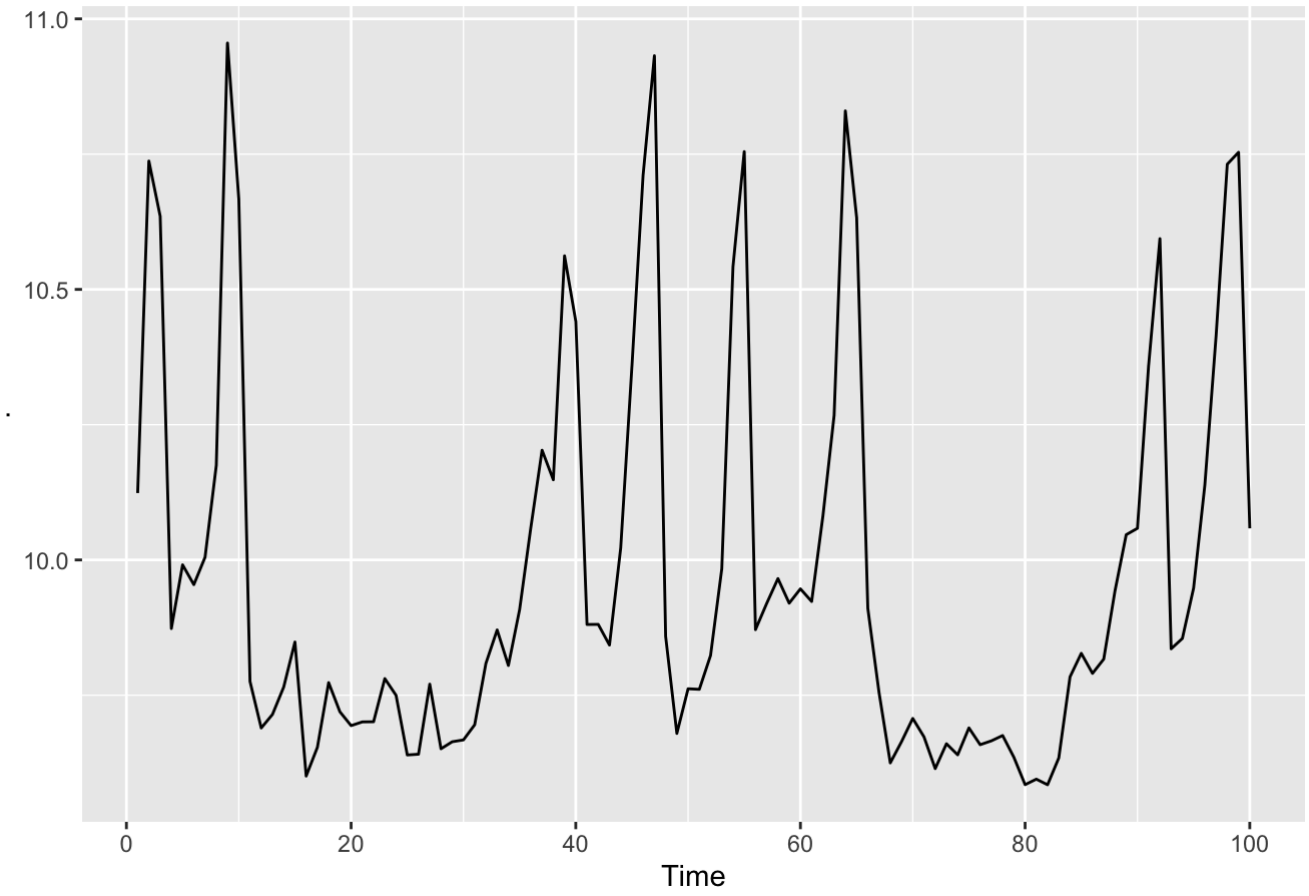
We can see that the coefficients passed the significance test, however the residual plots and Ljung-Box test shows that there are still collinearity for the fitted models. Therefore, our models can still be improved. Knowing that holiday data is involved, we explore transformation, differencing and seasonal differencing of our data.

Log transform and Differencing

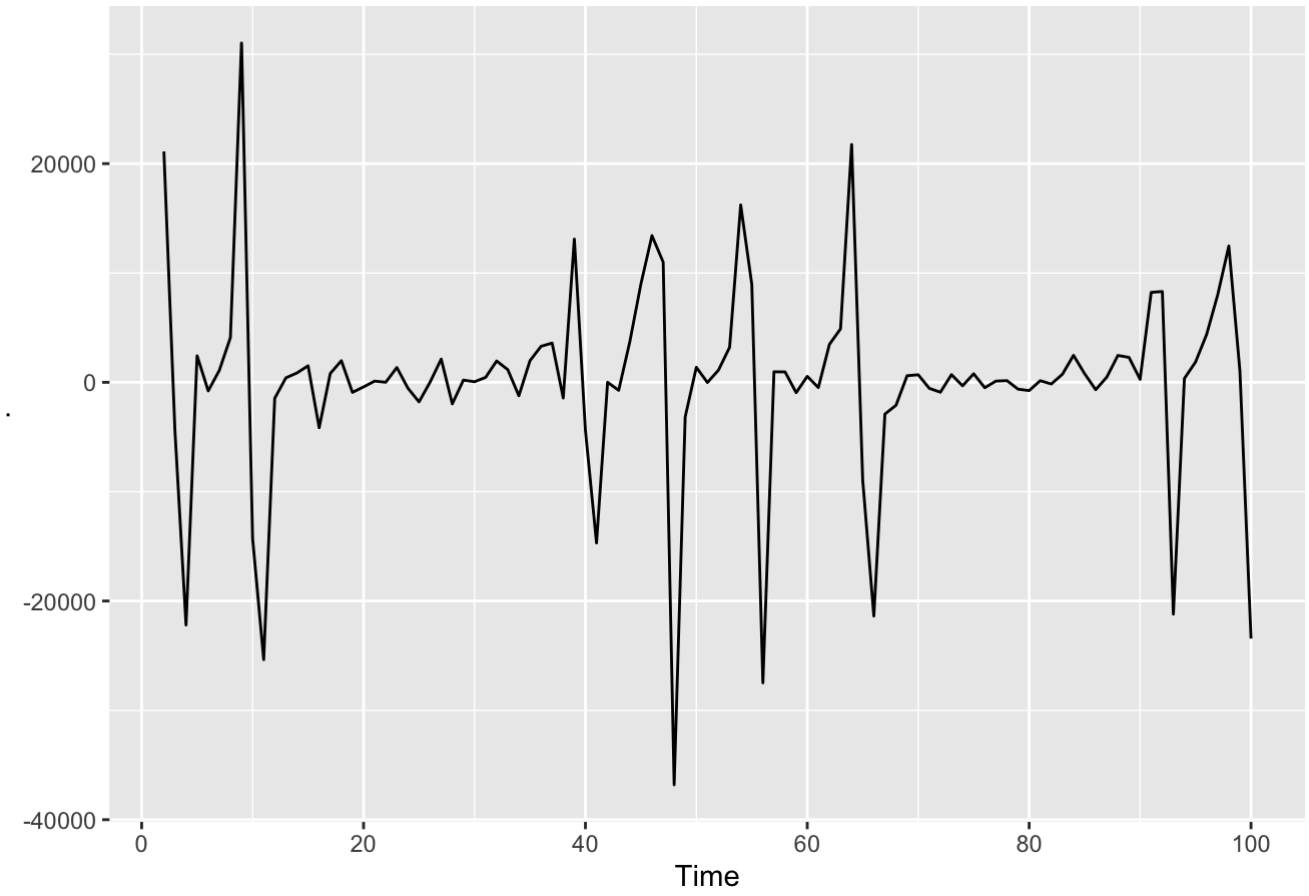
```
ts_train %>% autoplot()
```



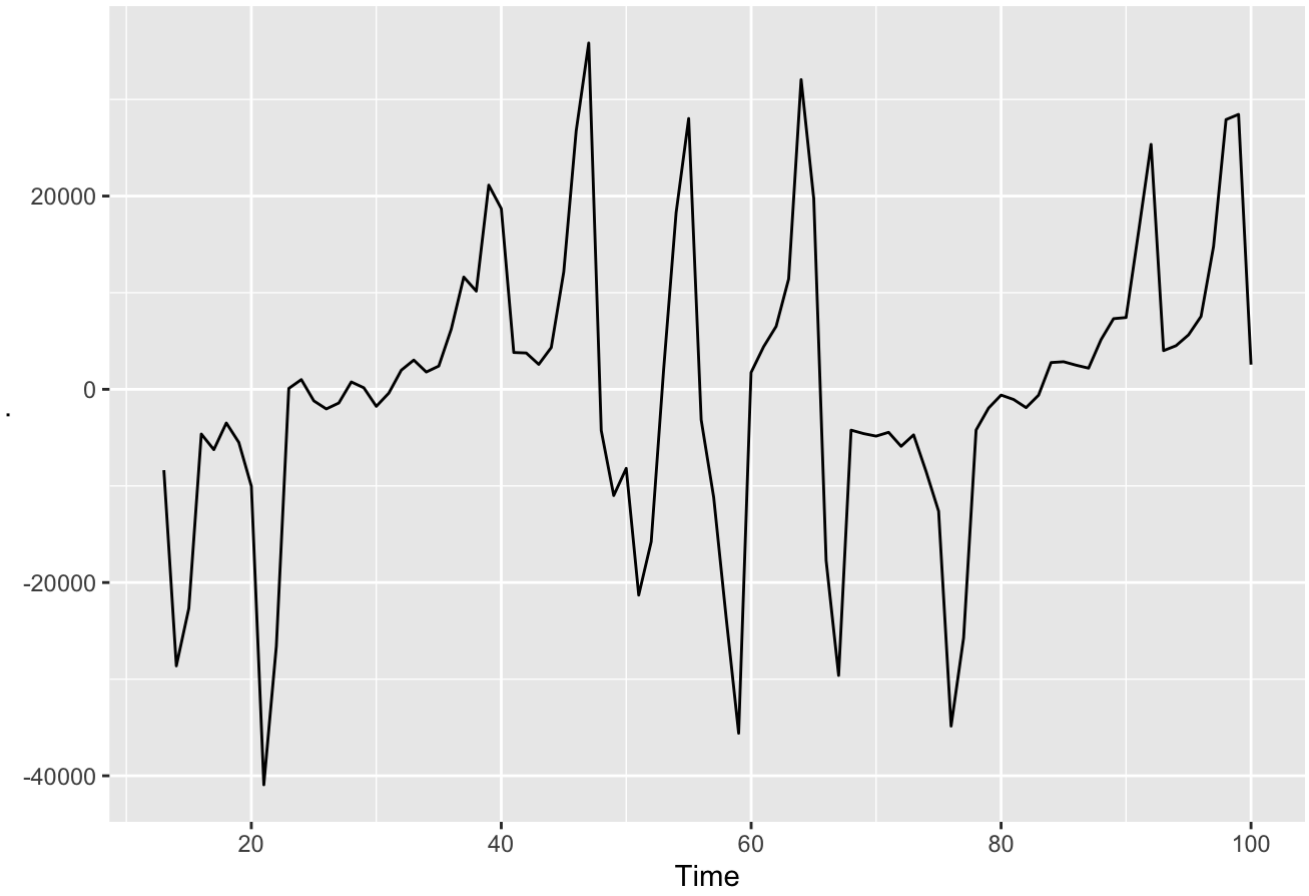

```
ts_train %>% log() %>% autoplot()
```



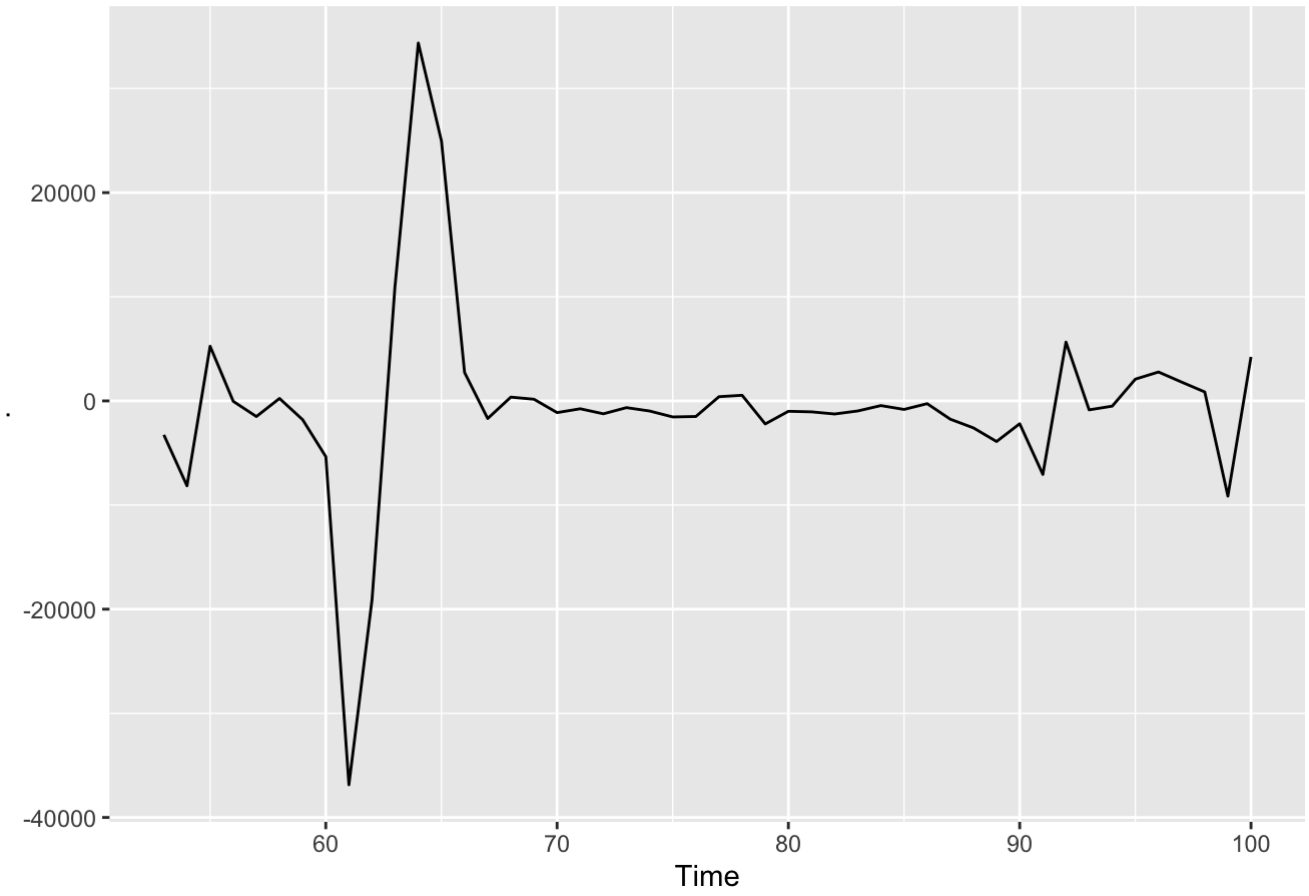

```
ts_train %>% diff() %>% autoplot()
```



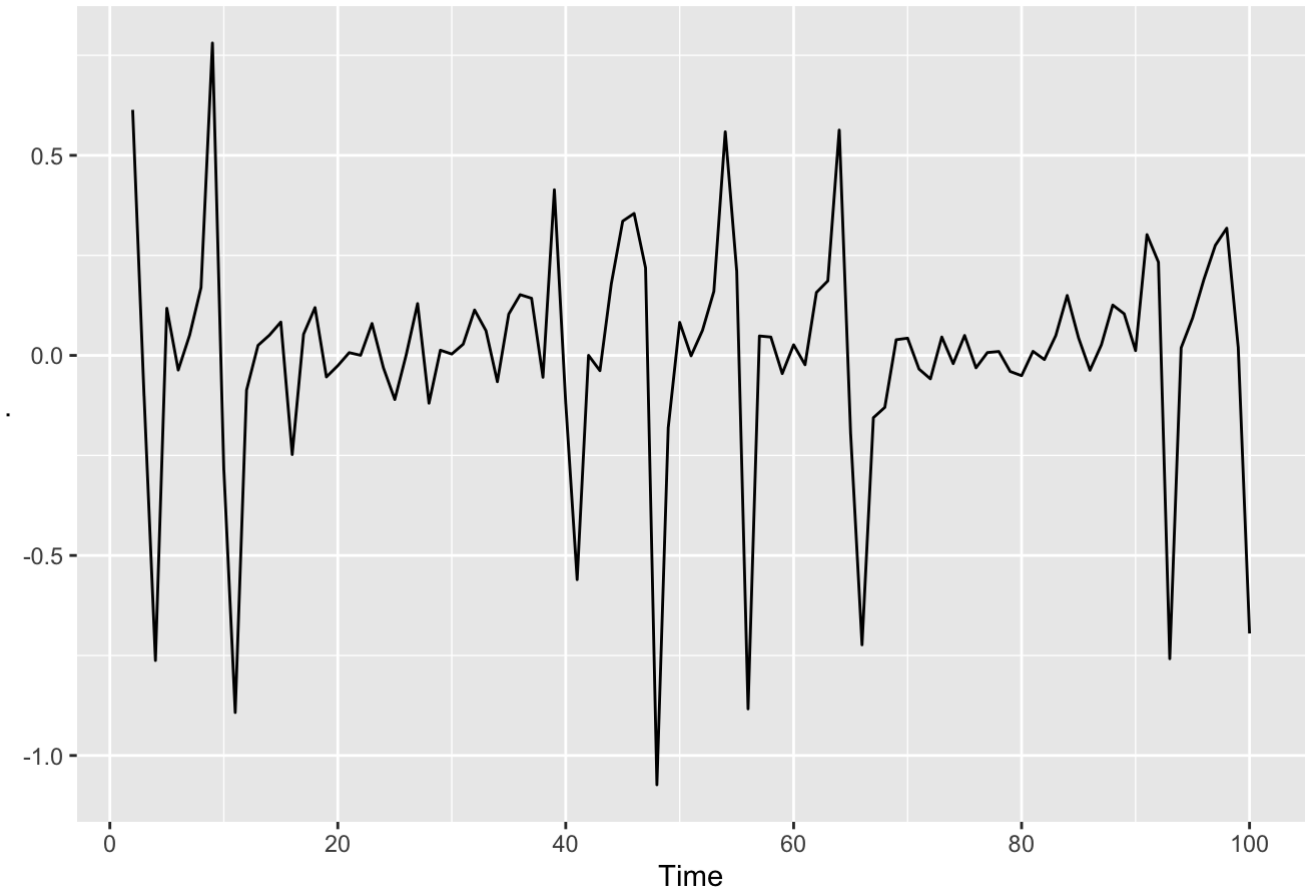

```
ts_train %>% diff(12) %>% autoplot()
```



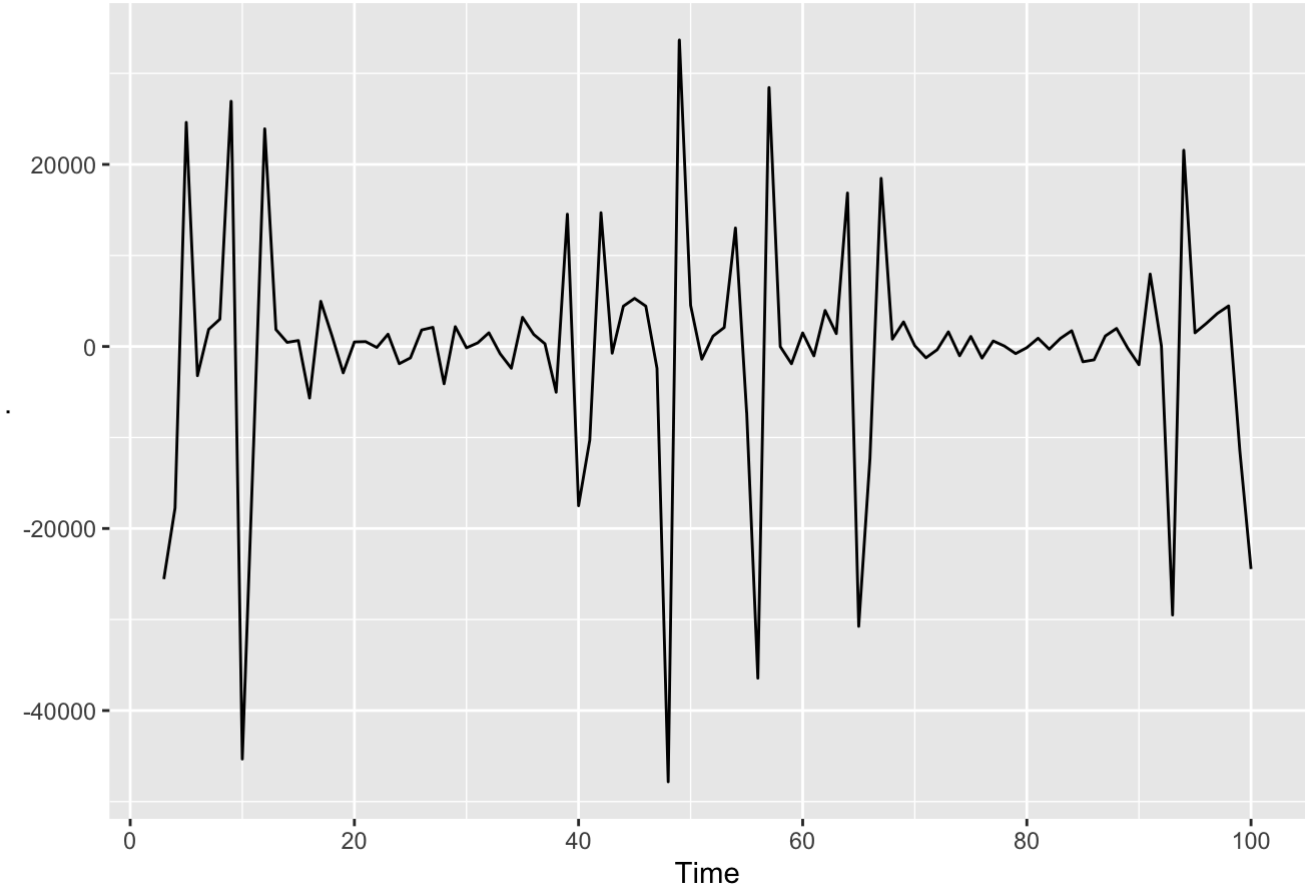

```
ts_train %>% diff(52) %>% autoplot()
```




```
ts_train %>% log() %>% diff() %>% autoplot()
```




```
ts_train %>% diff() %>% diff() %>% autoplot()
```

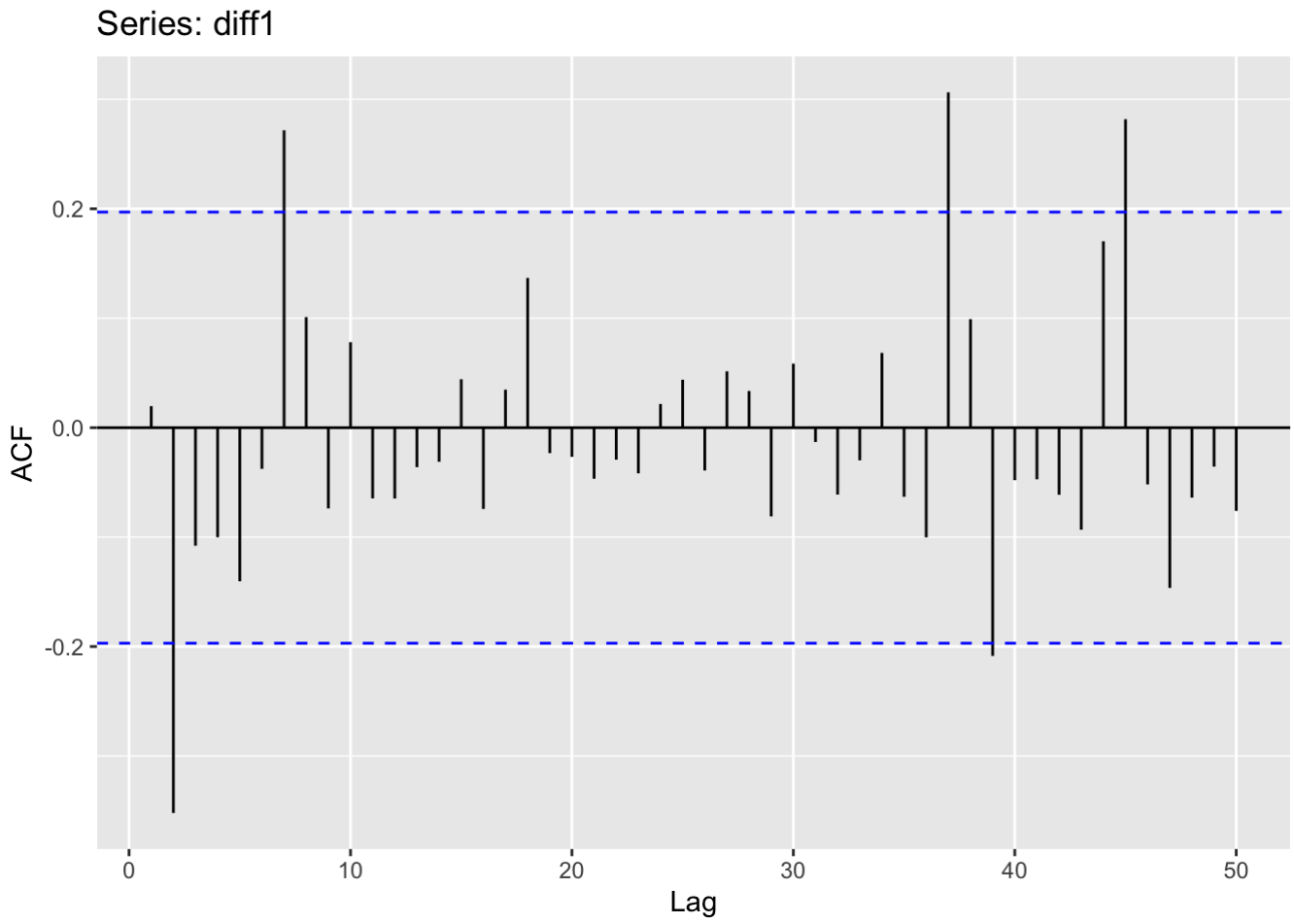



```
#ndiffs(ts_sales)

diff1 = ts_train %>% diff()
diff2 = ts_train %>% diff(52)
diff3 = ts_train %>% diff() %>% diff()
```

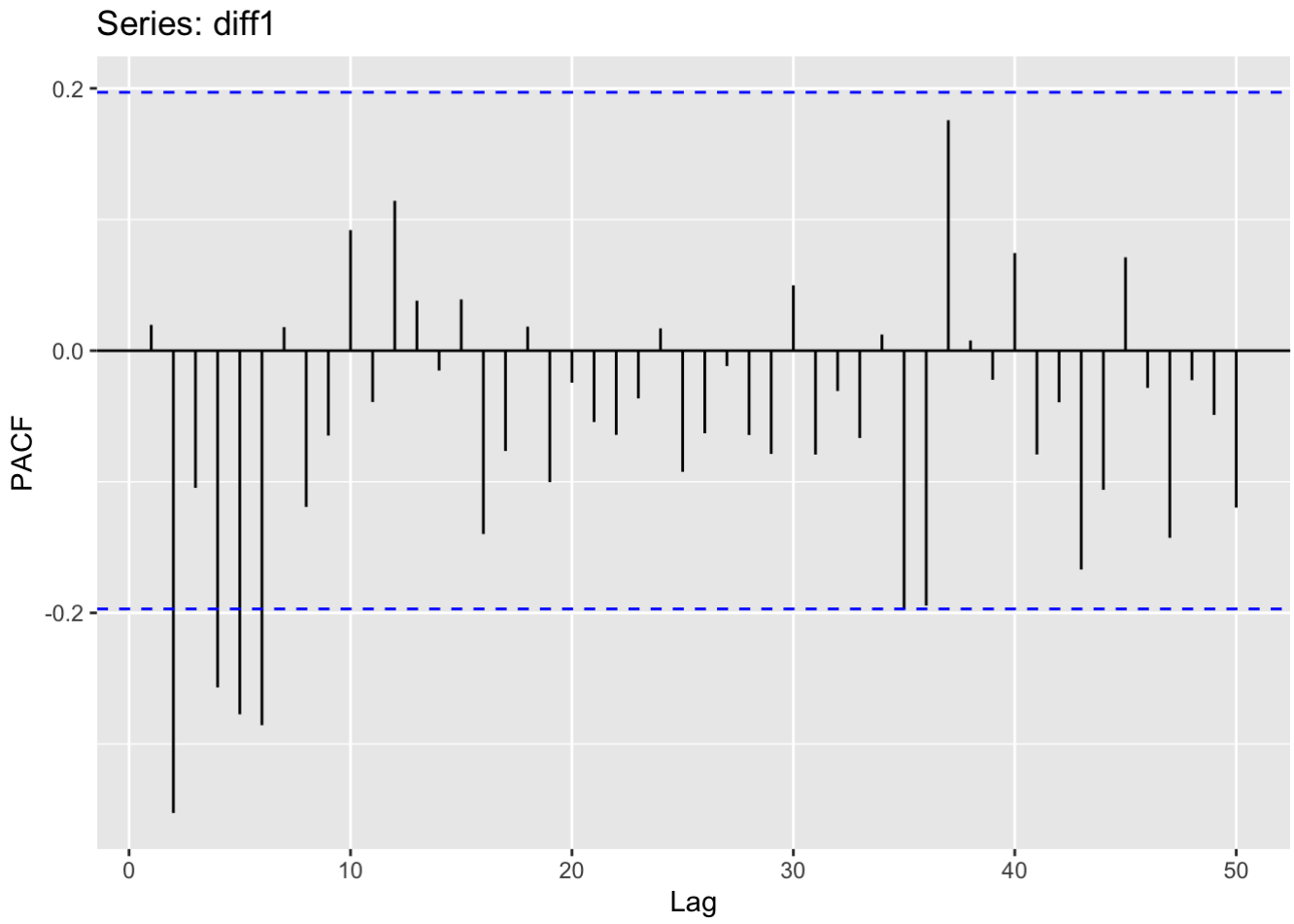
We can see that log transform doesnt really help much with our data. For differencing we tried our regular differencing, differencing by 12 and 52 to incorporate the “seasonal factor”. We can see that differencing it once shows the best plot for our model. Using the ndiffs function does not show us on what order we need to difference it with.

We decide to take the top 3 picks of our transformed data to look at the Acf and Pacf plot.

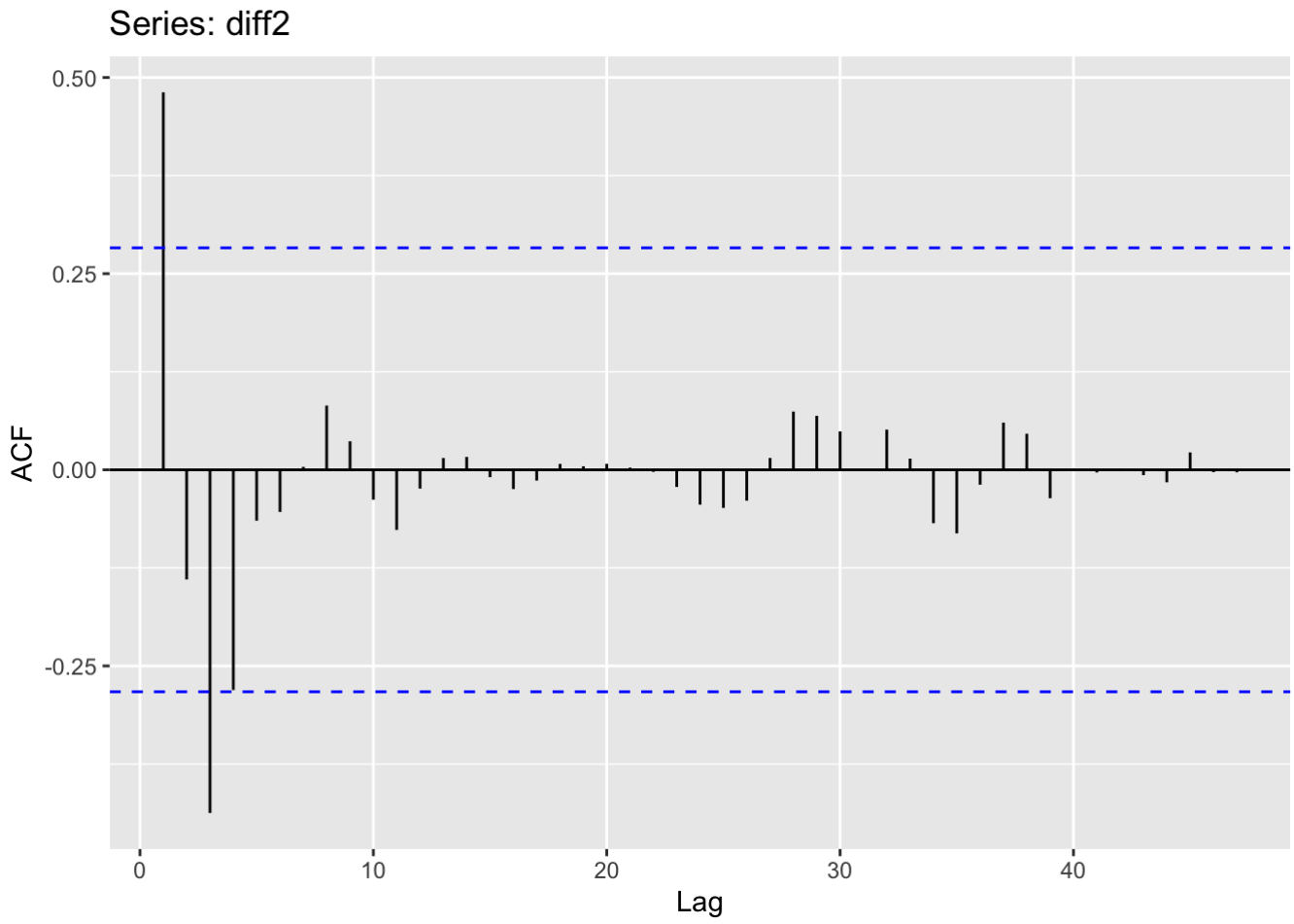
```
ggAcf(diff1, lag.max =50)
```



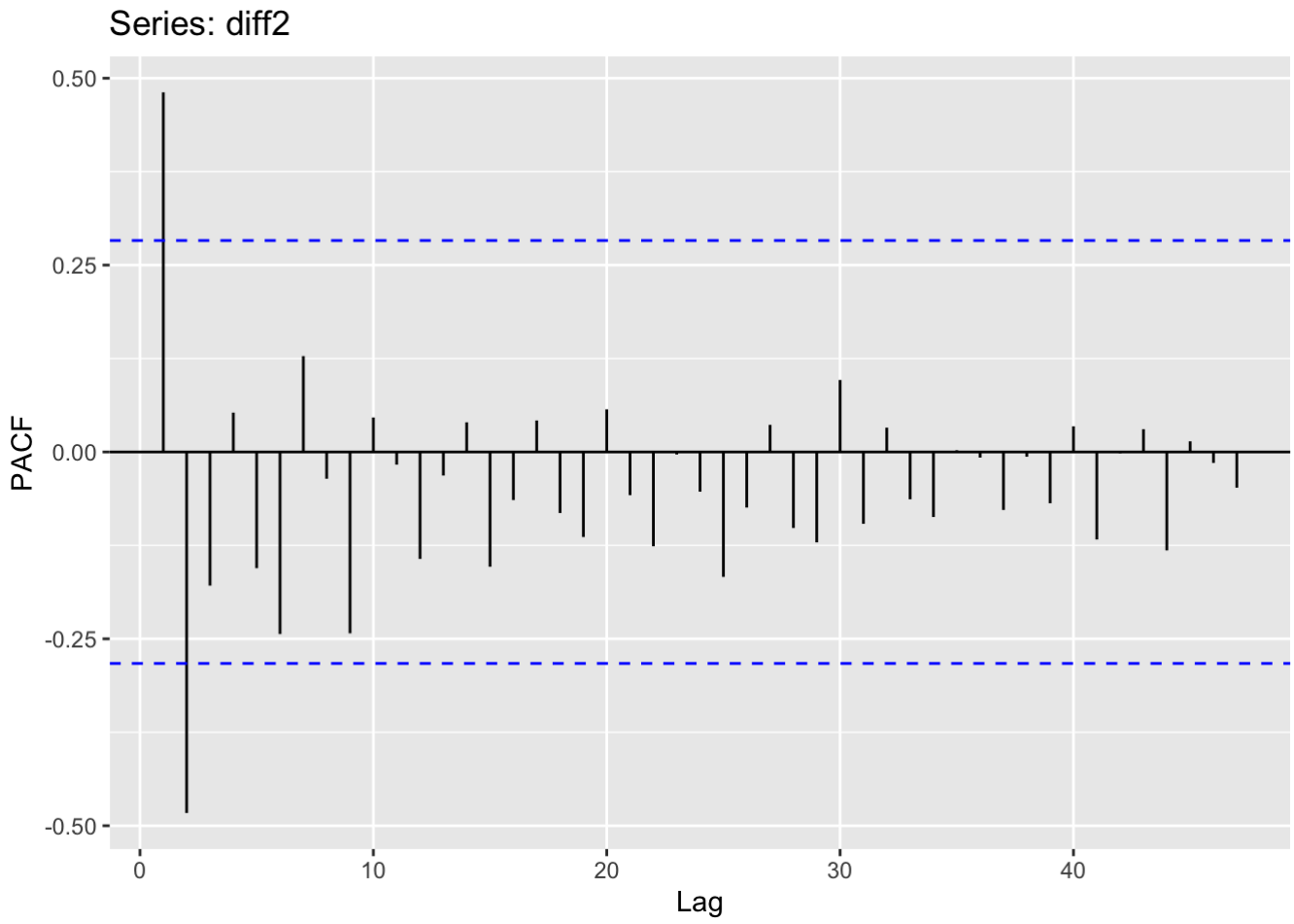

```
ggPacf(diff1, lag.max =50)
```



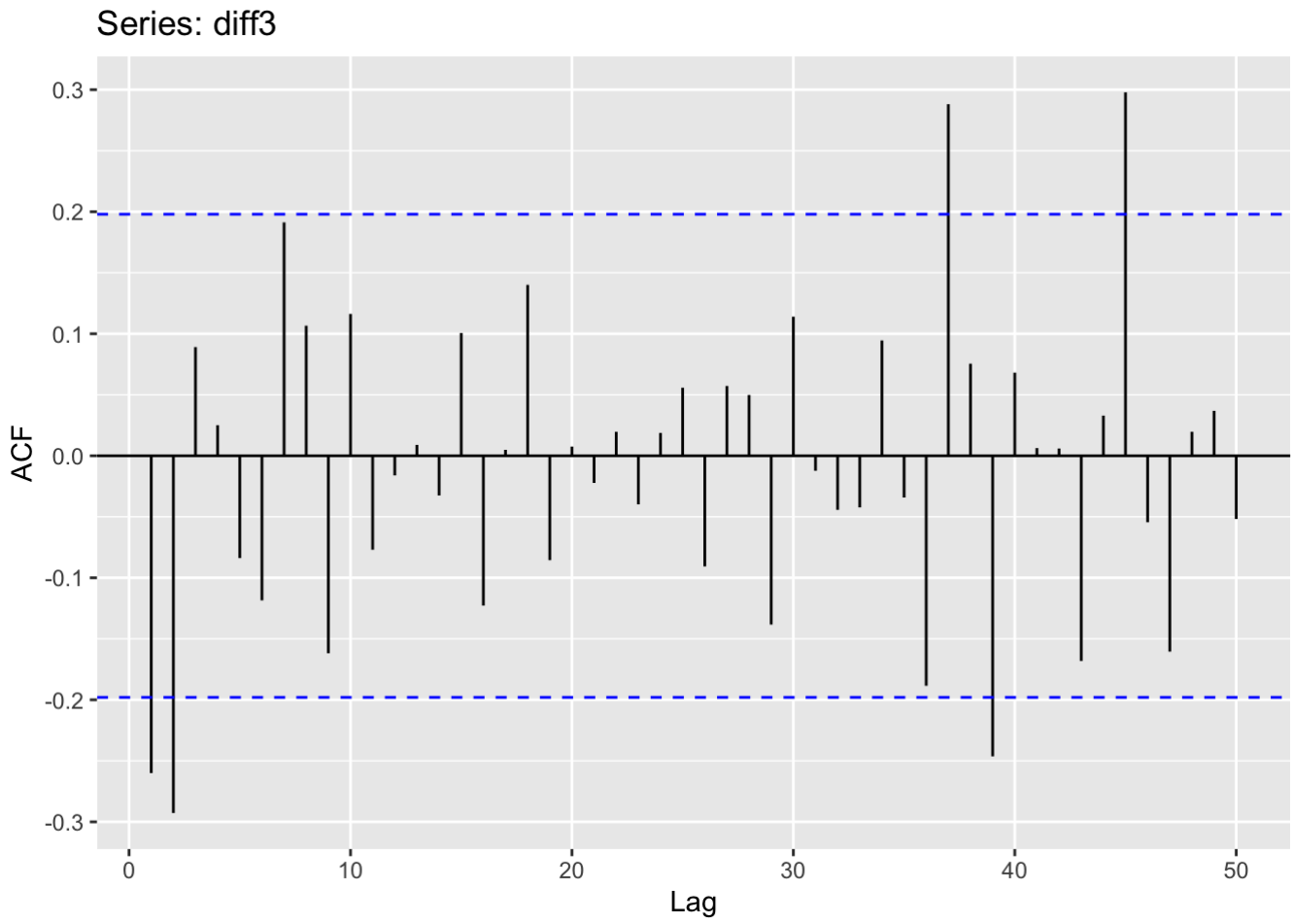

```
ggAcf(diff2, lag.max =50)
```



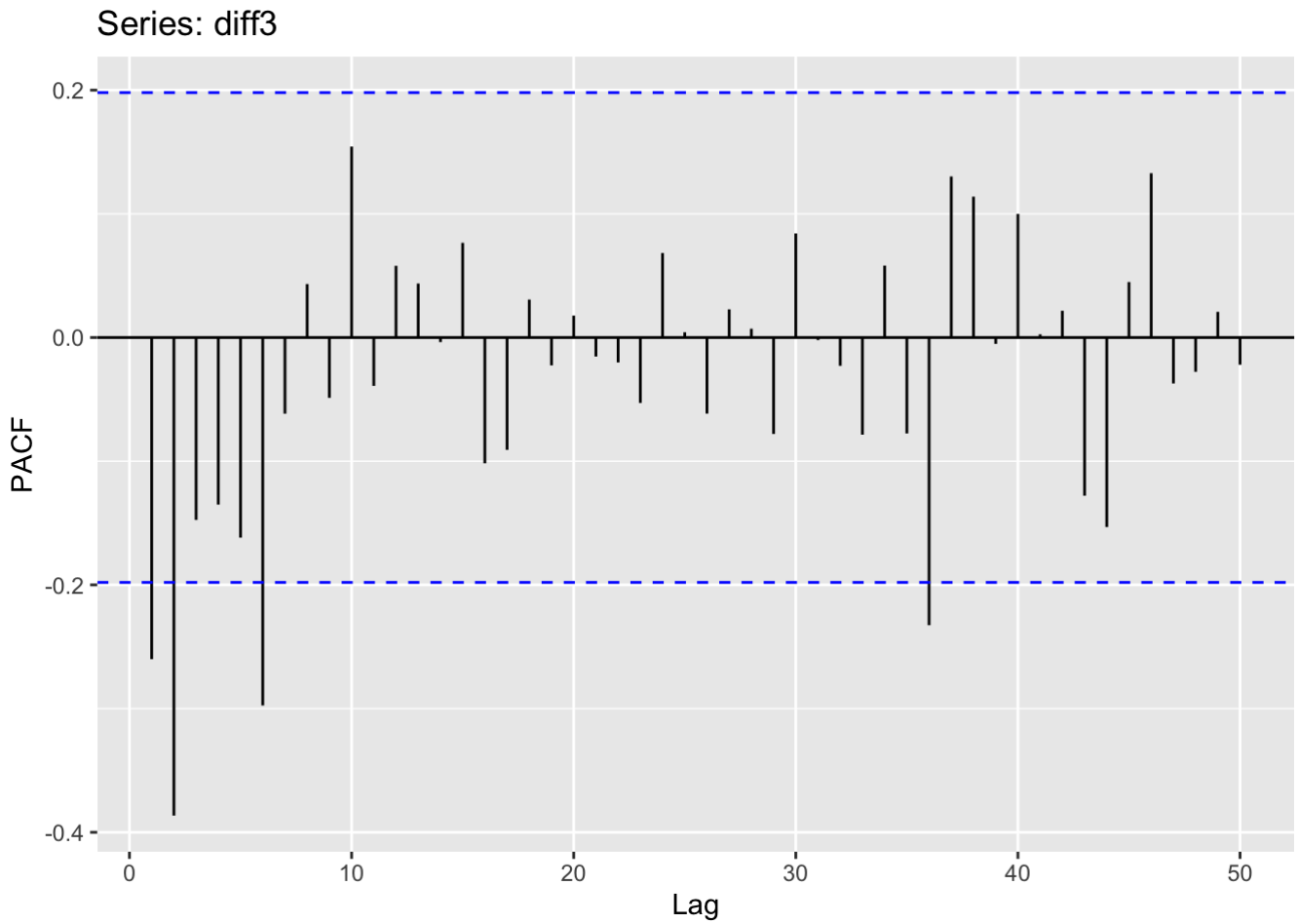

```
ggPacf(diff2, lag.max =50)
```




```
ggAcf(diff3, lag.max =50)
```




```
ggPacf(diff3, lag.max =50)
```



Comparing the ACF and PACF charts, we can see that diff2 (differencing by 52) performs the best. This make sense since our data is weekly sales. We can go forward to use diff2 and perform the ARMA model, and check the AIC/BIC values, coefficient test, residual plots and the box test.

```
eacf(diff2)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o x o o o o o o o o  o  o  o
## 1 x o x o o o o o o o o  o  o  o
## 2 o o o o o o o o o o o  o  o  o
## 3 x o o o o o o o o o o  o  o  o
## 4 x o o o o o o o o o o  o  o  o
## 5 x x x o o o o o o o o  o  o  o
## 6 x o x o o o o o o o o  o  o  o
## 7 x o o o o o o o o o o  o  o  o
```

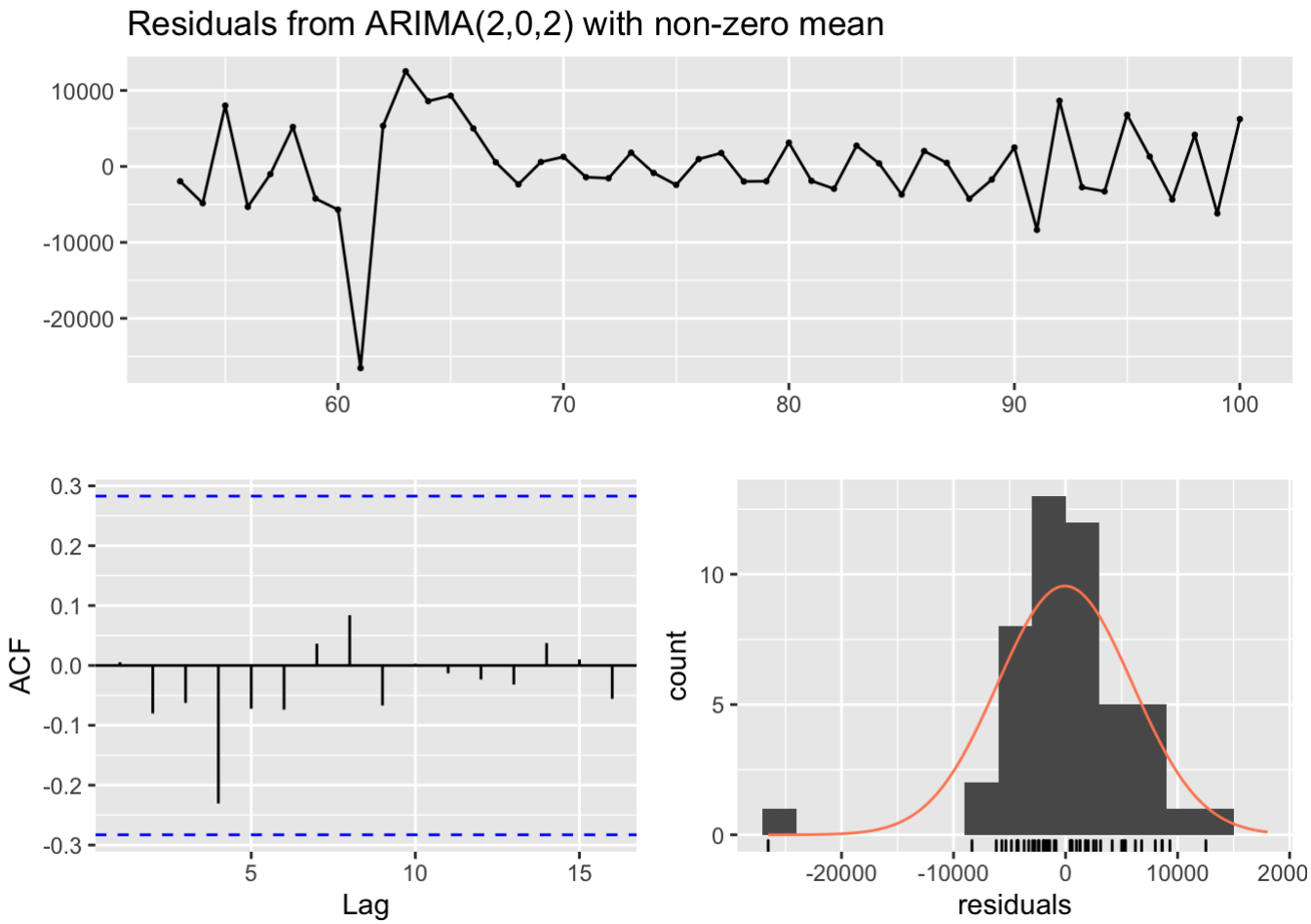
```
Diff52_ARMA = Arima(diff2, order = c(2,0,2)) # 2,2 is the best
Diff52_ARMA
```

```
## Series: diff2
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ma1     ma2       mean
##       0.1304  -0.5709  0.6647  0.9999  -286.1717
## s.e.  0.1312   0.1347  0.0898  0.1451  1585.7939
##
## sigma^2 estimated as 39695270:  log likelihood=-488.54
## AIC=989.09   AICc=991.13   BIC=1000.31
```

```
coeftest(Diff52_ARMA)
```

```
##
## z test of coefficients:
##
##              Estimate  Std. Error z value  Pr(>|z|)
## ar1          0.130351    0.131226  0.9933    0.3206
## ar2         -0.570931    0.134744 -4.2371 2.264e-05 ***
## ma1          0.664662    0.089842  7.3982 1.381e-13 ***
## ma2          0.999942    0.145111  6.8909 5.545e-12 ***
## intercept -286.171667 1585.793905 -0.1805    0.8568
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
checkresiduals(Diff52_ARMA)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,2) with non-zero mean
## Q* = 4.8221, df = 5, p-value = 0.438
##
## Model df: 5.   Total lags used: 10
```

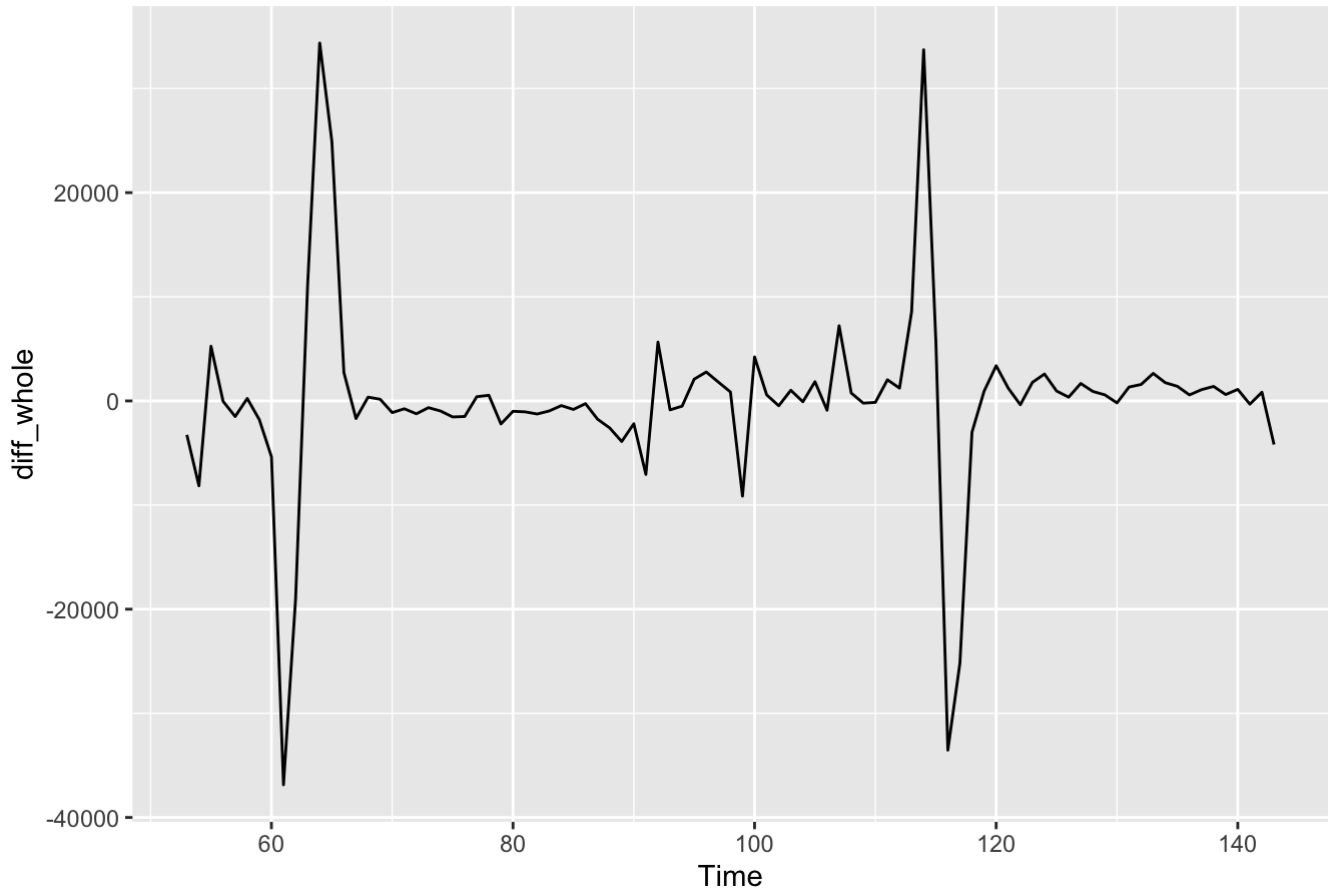
```
Box.test(Diff52_ARMA$residuals, lag = 20, type = 'L')
```

```
##
##  Box-Ljung test
##
## data:  Diff52_ARMA$residuals
## X-squared = 5.6572, df = 20, p-value = 0.9993
```

The results show by differencing our dataset by 52 (representing 52 weeks a year) improves our model significantly, with AIC valueof 989.09 and BIC value of 1000.31. The model passed the coefficient test of significance, as well as residual diagnostics and Ljung Box test for normality. Therefore, this is a good model.

With that, we performed a forecast using the fitted model

```
diff_whole = ts_sales %>% diff(52)
autoplot(diff_whole)
```



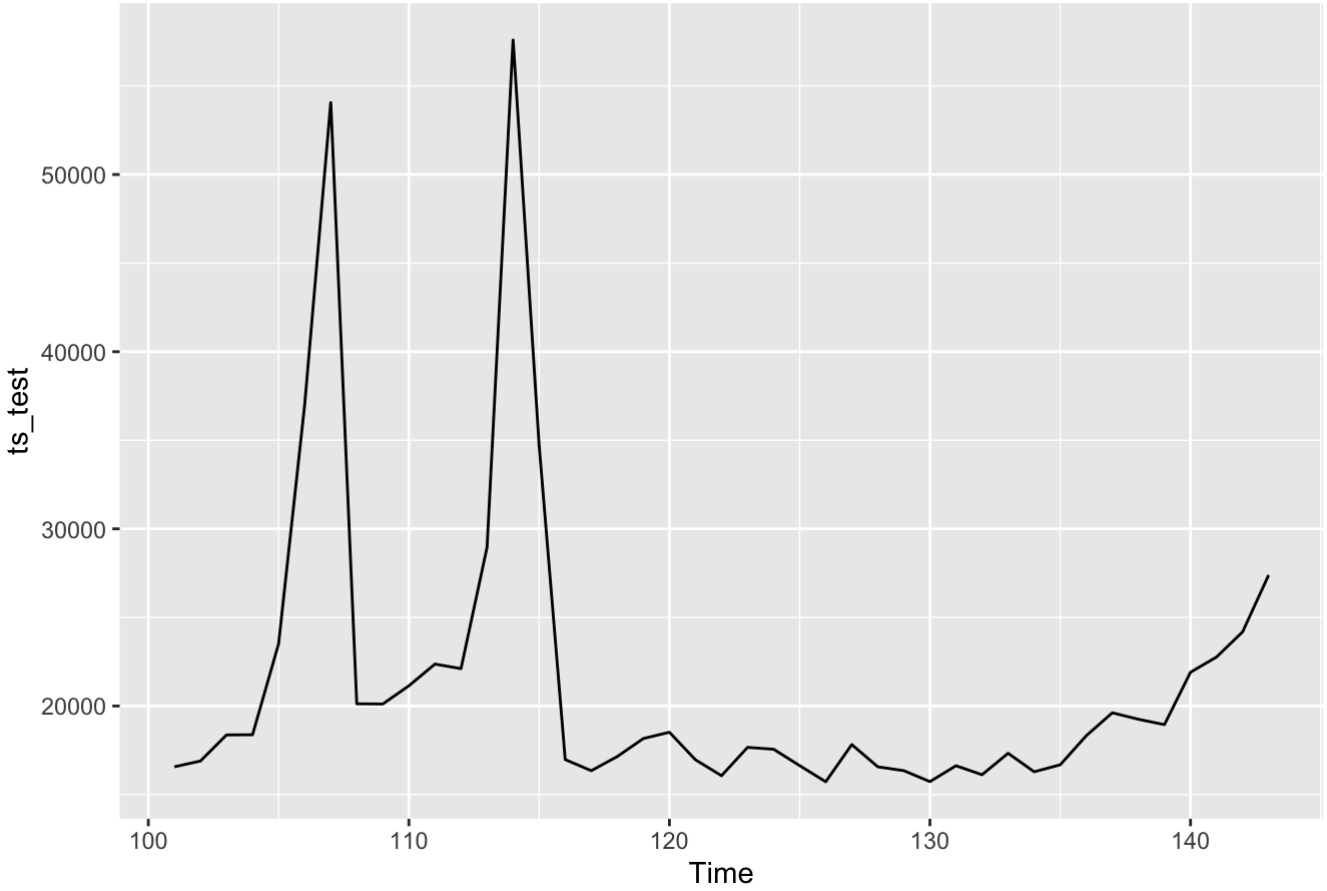

```
### The dataset is shorter cuz of the differencing. THats why it didnt work previously


diff2_tst = ts_test %>% diff()
class(ts_test)
```

```
## [1] "ts"
```

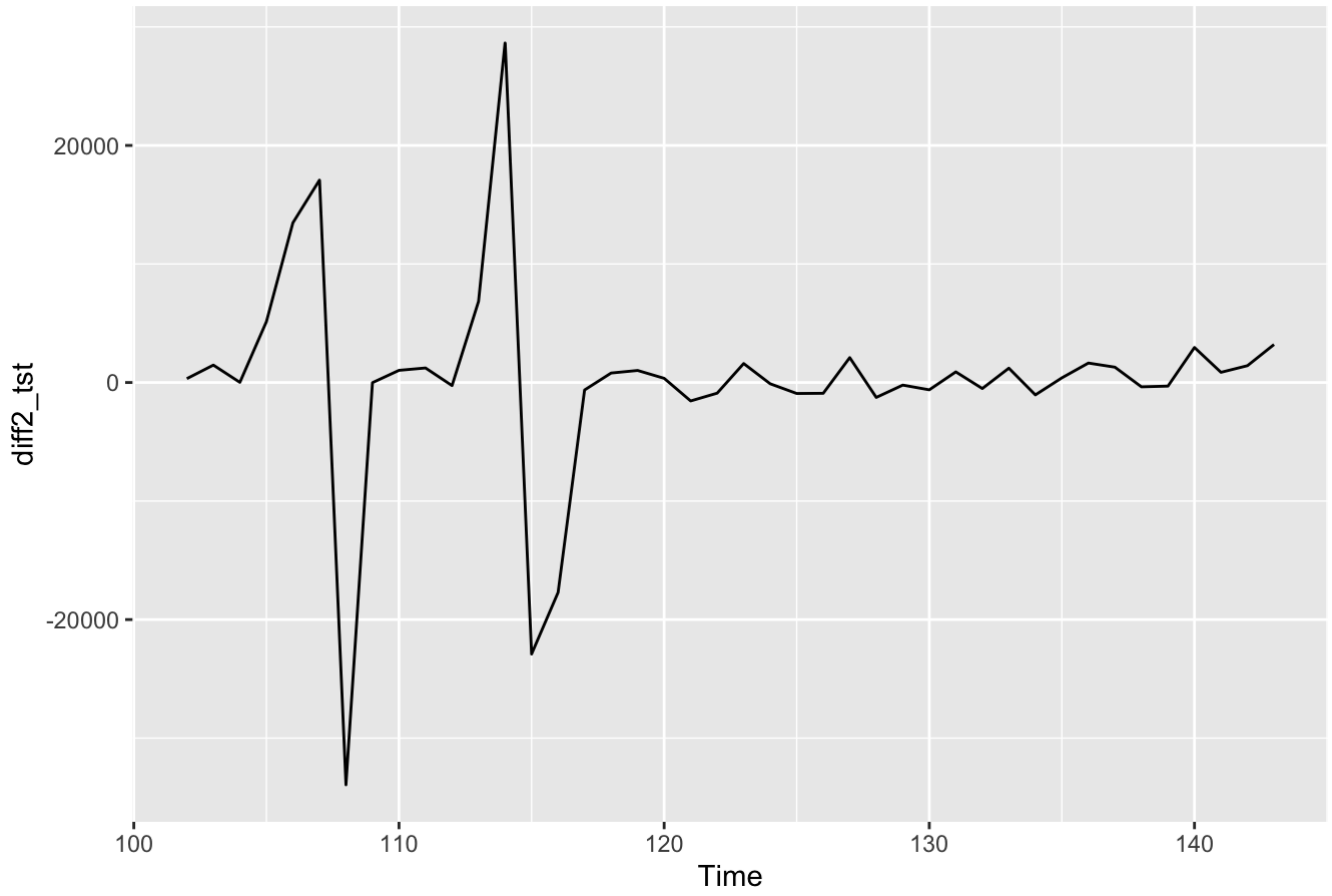
```
class(diff2_tst)
```

```
## [1] "ts"
```

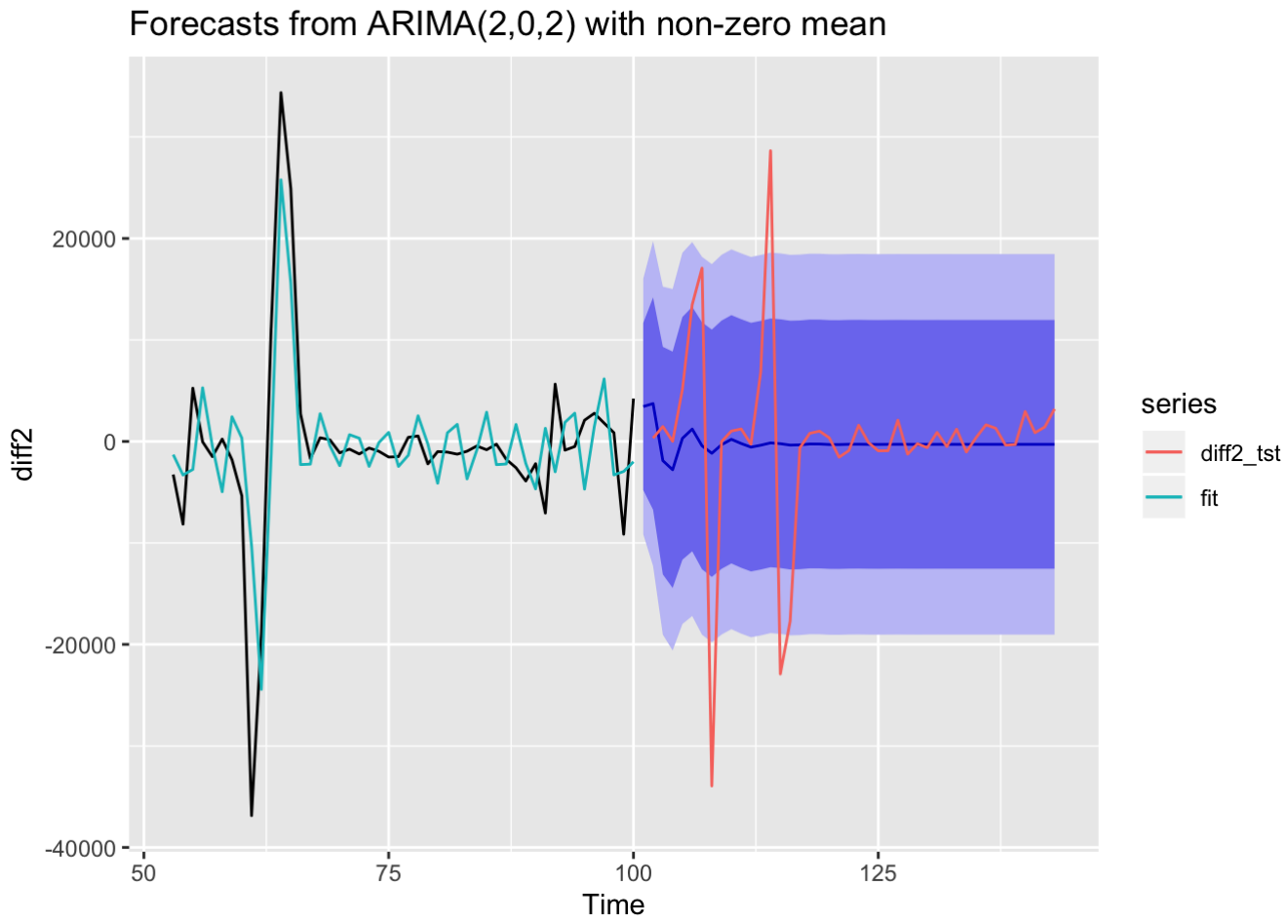
```
autoplot(ts_test)
```




```
autoplot(diff2_tst)
```




```
fcast_diff52 = forecast(Diff52_ARMA, h = 43)
fit = fitted(Diff52_ARMA)

autoplot(fcast_diff52) + autolayer(fit) + autolayer(diff2_tst)
```

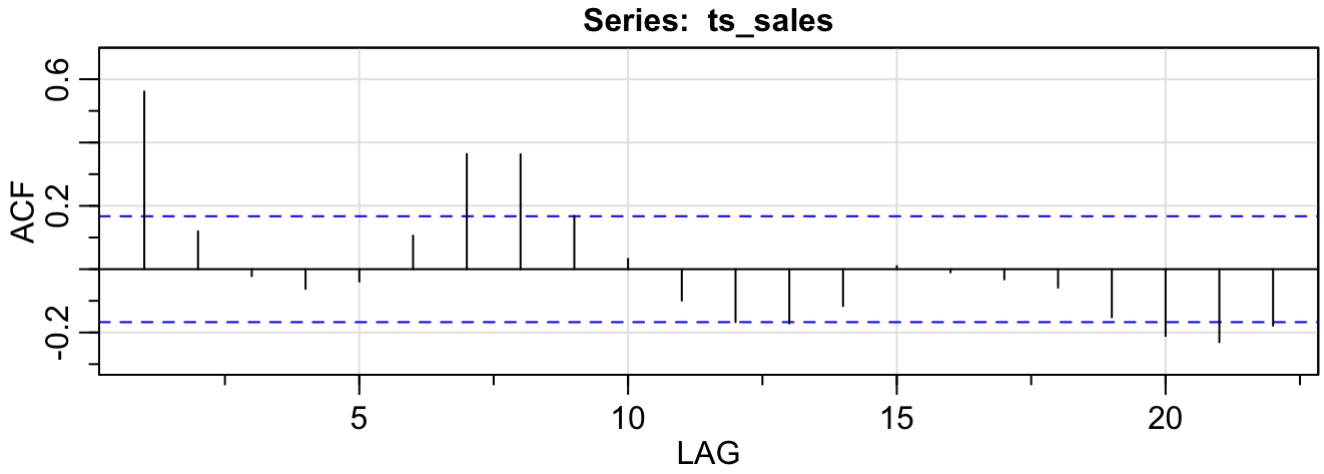


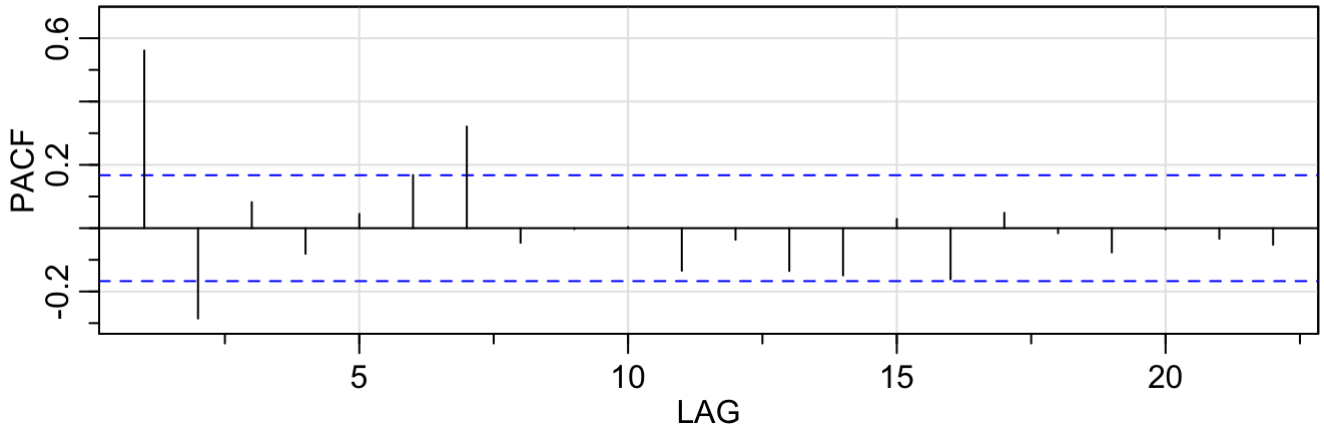

```
accuracy(f = fcast_diff52, x = diff2_tst)
```

```
##                     ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -50.0668 5963.250 4185.526 378.8517 519.2404 0.8080619
## Test set     512.6685 8914.428 4417.776 499.1221 695.3357 0.8529002
##                      ACF1 Theil's U
## Training set  0.005470685        NA
## Test set     -0.102722560  0.936152
```

ARIMA AND SARIMA

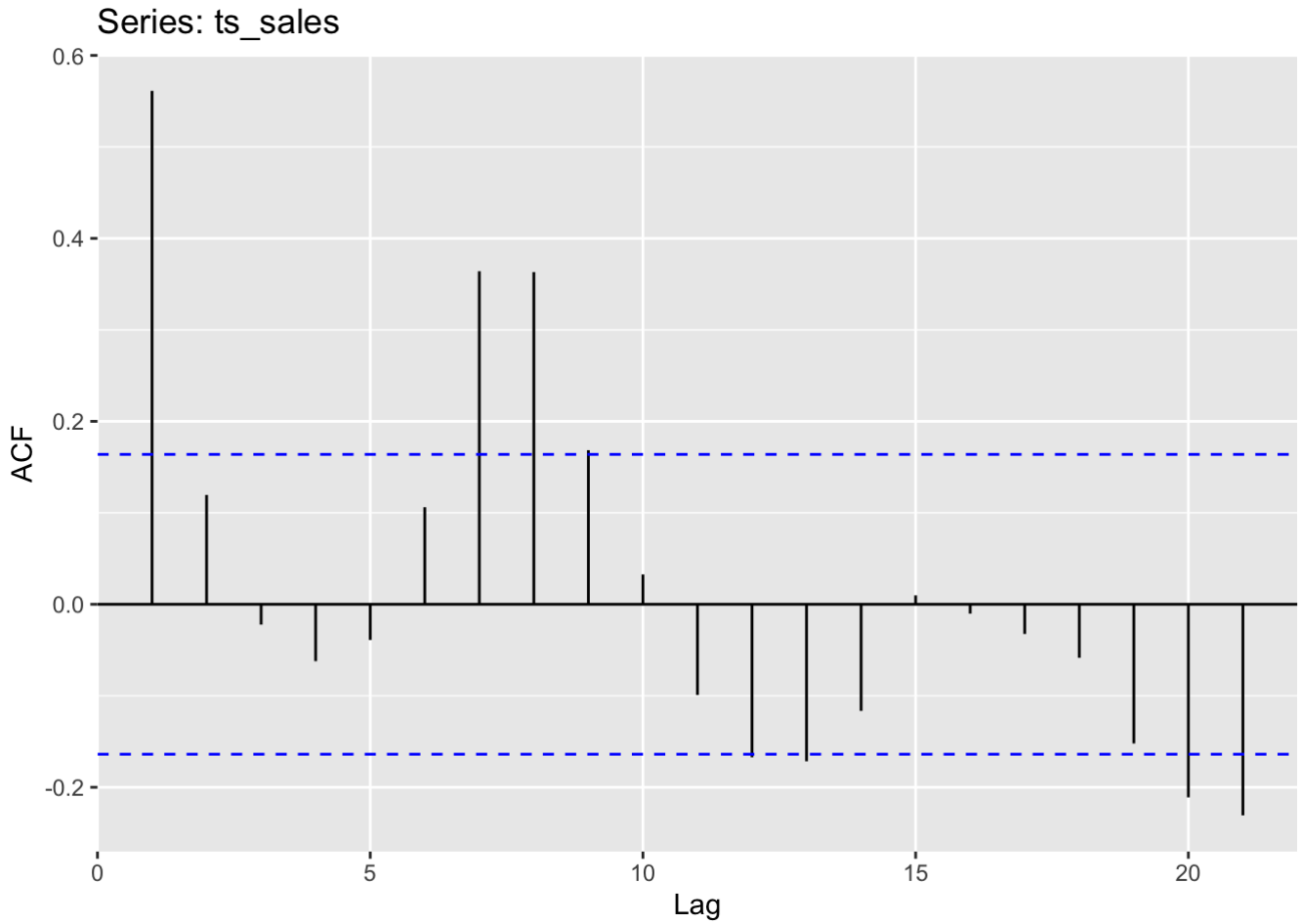
```
acf2(ts_sales)
```



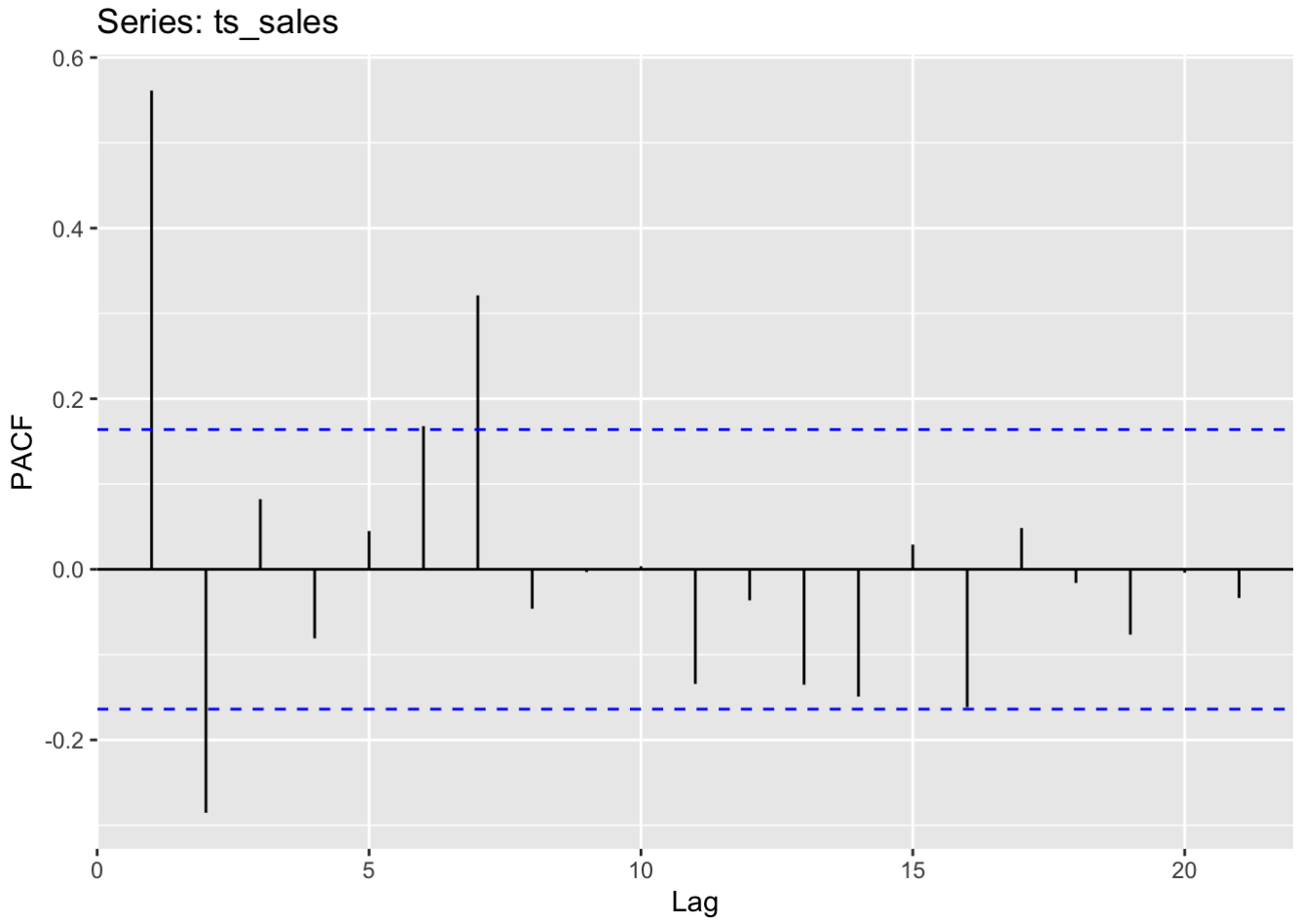



```
##          ACF  PACF
##  [1,]  0.56  0.56
##  [2,]  0.12 -0.29
##  [3,] -0.02  0.08
##  [4,] -0.06 -0.08
##  [5,] -0.04  0.04
##  [6,]  0.11  0.17
##  [7,]  0.36  0.32
##  [8,]  0.36 -0.05
##  [9,]  0.17  0.00
## [10,]  0.03  0.00
## [11,] -0.10 -0.13
## [12,] -0.17 -0.04
## [13,] -0.17 -0.14
## [14,] -0.12 -0.15
## [15,]  0.01  0.03
## [16,] -0.01 -0.16
## [17,] -0.03  0.05
## [18,] -0.06 -0.02
## [19,] -0.15 -0.08
## [20,] -0.21  0.00
## [21,] -0.23 -0.03
## [22,] -0.18 -0.05
```

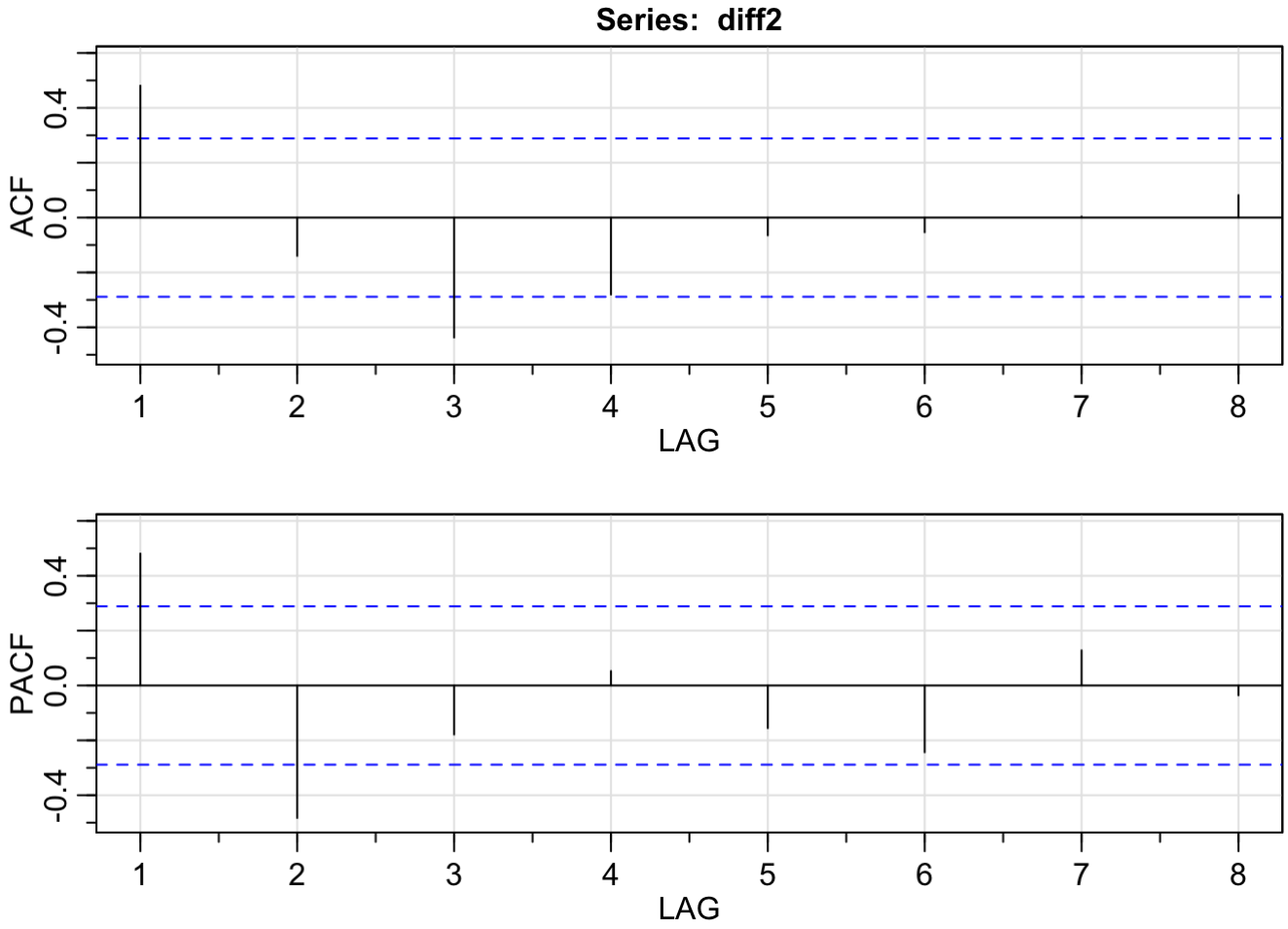
```
ggAcf(ts_sales)
```




```
ggPacf(ts_sales)
```




```
acf2(diff2)
```




```
##        ACF  PACF
## [1,]  0.48  0.48
## [2,] -0.14 -0.48
## [3,] -0.44 -0.18
## [4,] -0.28  0.05
## [5,] -0.06 -0.16
## [6,] -0.05 -0.24
## [7,]  0.00  0.13
## [8,]  0.08 -0.04
```

```
auto.arima(ts_sales, max.d = 5, max.D = 5, seasonal = T, lambda = 0, stepwise = F)
```

```
## Series: ts_sales
## ARIMA(1,0,1) with non-zero mean
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1     ma1    mean
##       0.3912  0.5424  9.9552
## s.e.  0.1006  0.0941  0.0504
##
## sigma^2 estimated as 0.05856:  log likelihood=1.05
## AIC=5.9   AICc=6.19   BIC=17.75
```

```
auto.arima(diff2, max.d = 5, max.D = 5, seasonal = T, lambda = 0, stepwise = F)
```

```
## Warning in log(x): NaNs produced

## Warning in log(x): NaNs produced
```

```
## Series: diff2
## ARIMA(2,0,0) with non-zero mean
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1      ar2    mean
##       1.3011  -0.7351  7.5201
## s.e.  0.1419   0.1324  0.3374
##
## sigma^2 estimated as 0.5018:  log likelihood=-22.16
## AIC=52.33   AICc=53.3   BIC=59.64
```

```
#sarima(ts_sales,2,2,2,2,2,2,2, S = 12)
#sarima(diff2,2,2,2,2,2,2,2, S = 12)
```

Using the acf2 function on both ts_sales and diff2 data, it does not display patterns for us to incorporate sARIMA. This makes sense since our best fitted model have been derived by differencing our sales data weekly. Based on the SARIMA model produced, our dataset does not fit the sarima models.

In conclusion, our best fitted model is Arima(diff2, order = c(2,0,2))

Time Series Regression

Linear trend

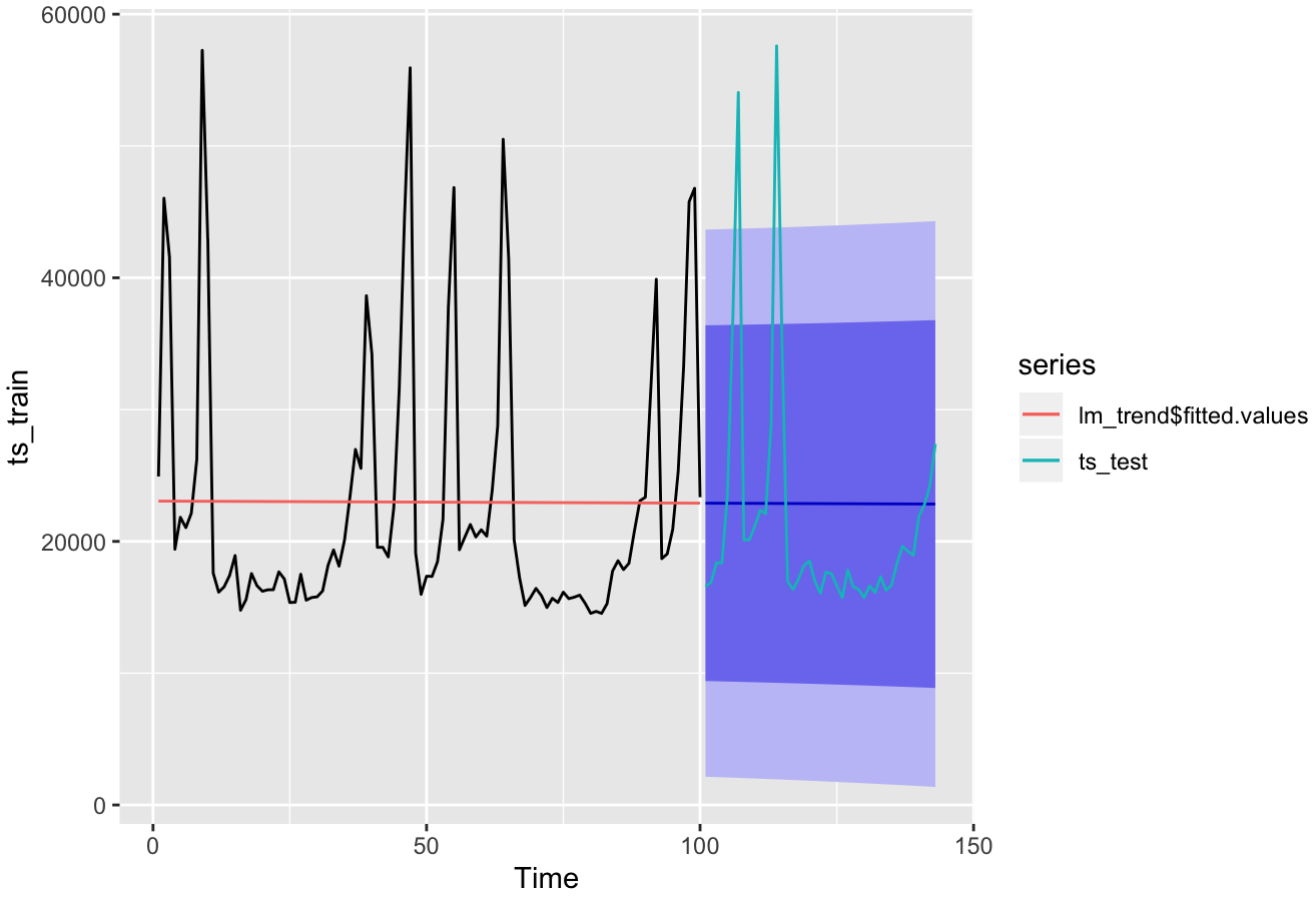
```
lm_trend = tslm(ts_train ~ trend)
lm_trend
```

```
##
## Call:
## tslm(formula = ts_train ~ trend)
##
## Coefficients:
## (Intercept)        trend
##   23058.503       -1.574
```

```
lm_trend_fcast = forecast(lm_trend, h = 43)
coeftest(lm_trend)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23058.5031  2065.9145 11.1614   <2e-16 ***
## trend          -1.5744    35.5164 -0.0443   0.9647
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
autoplot(ts_train) + autolayer(lm_trend$fitted.values) + autolayer(lm_trend_fcast) + aut
olayer(ts_test)
```




```
accuracy(lm_trend)
```

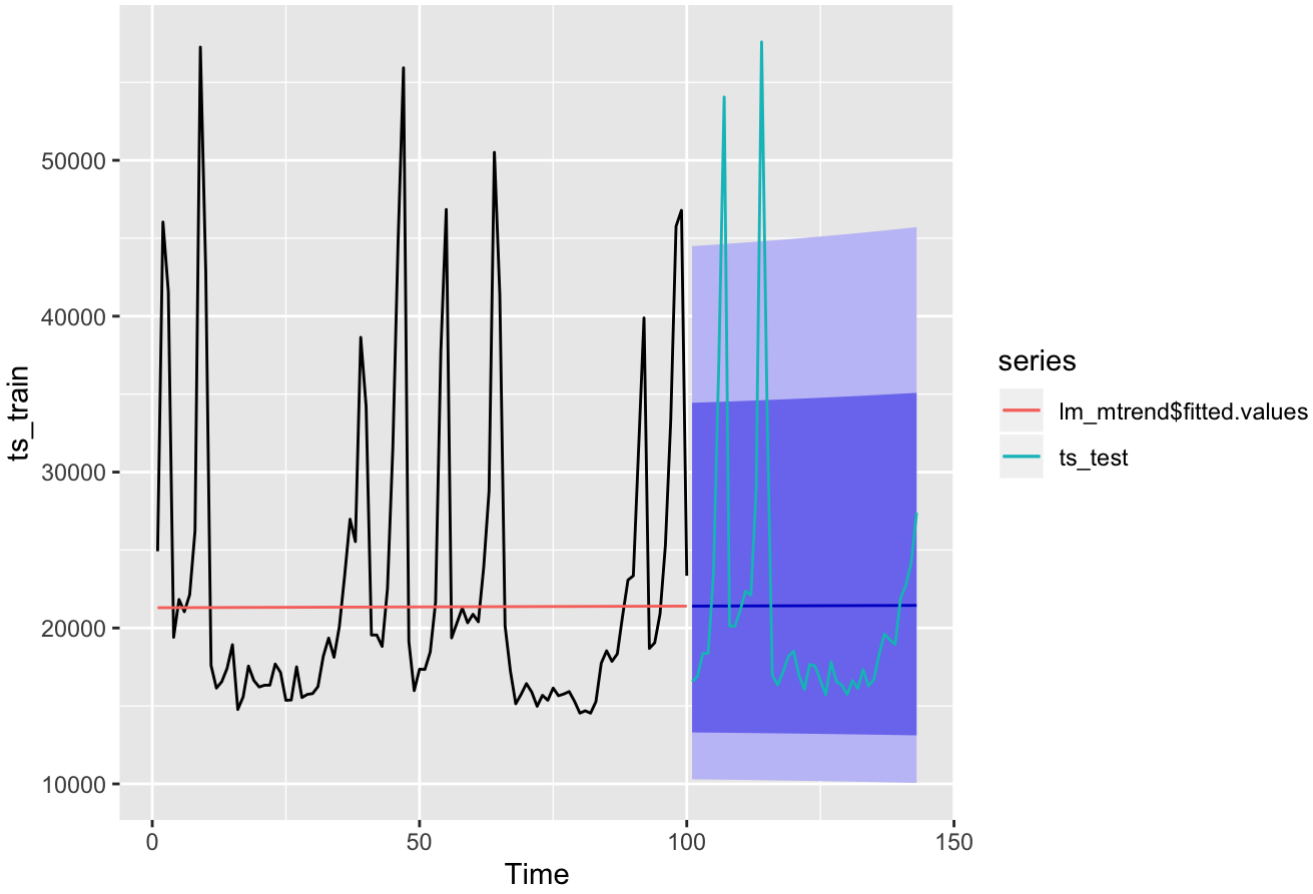
```
##                        ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -9.035966e-14 10149.16 7514.46 -13.71381 31.40154 1.467911
##                    ACF1
## Training set 0.5815627
```

Multiplicative trend

```
lm_mtrend = tslm(ts_train ~ trend, lambda=0)
lm_mtrend
```

```
##
## Call:
## tslm(formula = ts_train ~ trend, lambda = 0)
##
## Coefficients:
## (Intercept)        trend
##   9.966e+00    4.855e-05
```

```
lm_mtrend_fcast = forecast(lm_mtrend, h = 43)
autoplot(ts_train) + autolayer(lm_mtrend$fitted.values) + autolayer(lm_mtrend_fcast) + a
utolayer(ts_test)
```




```
coeftest(lm_mtrend)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 9.96649257 0.07283741 136.8321   <2e-16 ***
## trend       0.00004855 0.00125219   0.0388   0.9692
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
accuracy(lm_mtrend)
```

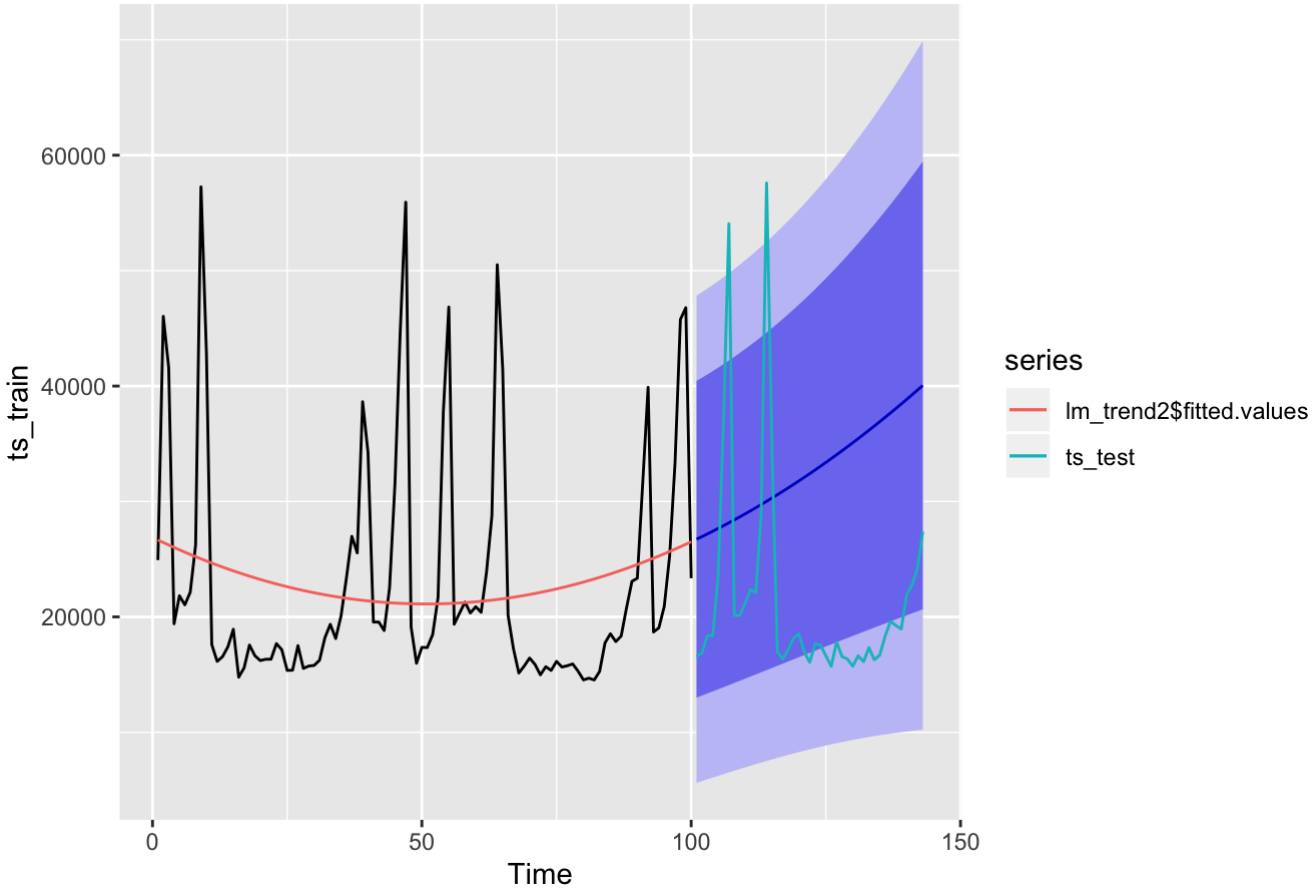
```
##                      ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
## Training set 1626.043 10278.86 6882.345 -5.66655 26.43838 1.34443 0.5815657
```

Quadratic trend

```
lm_trend2 = tslm(ts_train ~ trend+I(trend^2))
lm_trend2
```

```
##
## Call:
## tslm(formula = ts_train ~ trend + I(trend^2))
##
## Coefficients:
## (Intercept)        trend   I(trend^2)
##    26887.32      -226.80         2.23
```

```
lm_trend_fcast2 = forecast(lm_trend2, h = 43)
autoplot(ts_train) + autolayer(lm_trend2$fitted.values) + autolayer(lm_trend_fcast2) + a
utolayer(ts_test)
```



```
coeftest(lm_trend2)
```

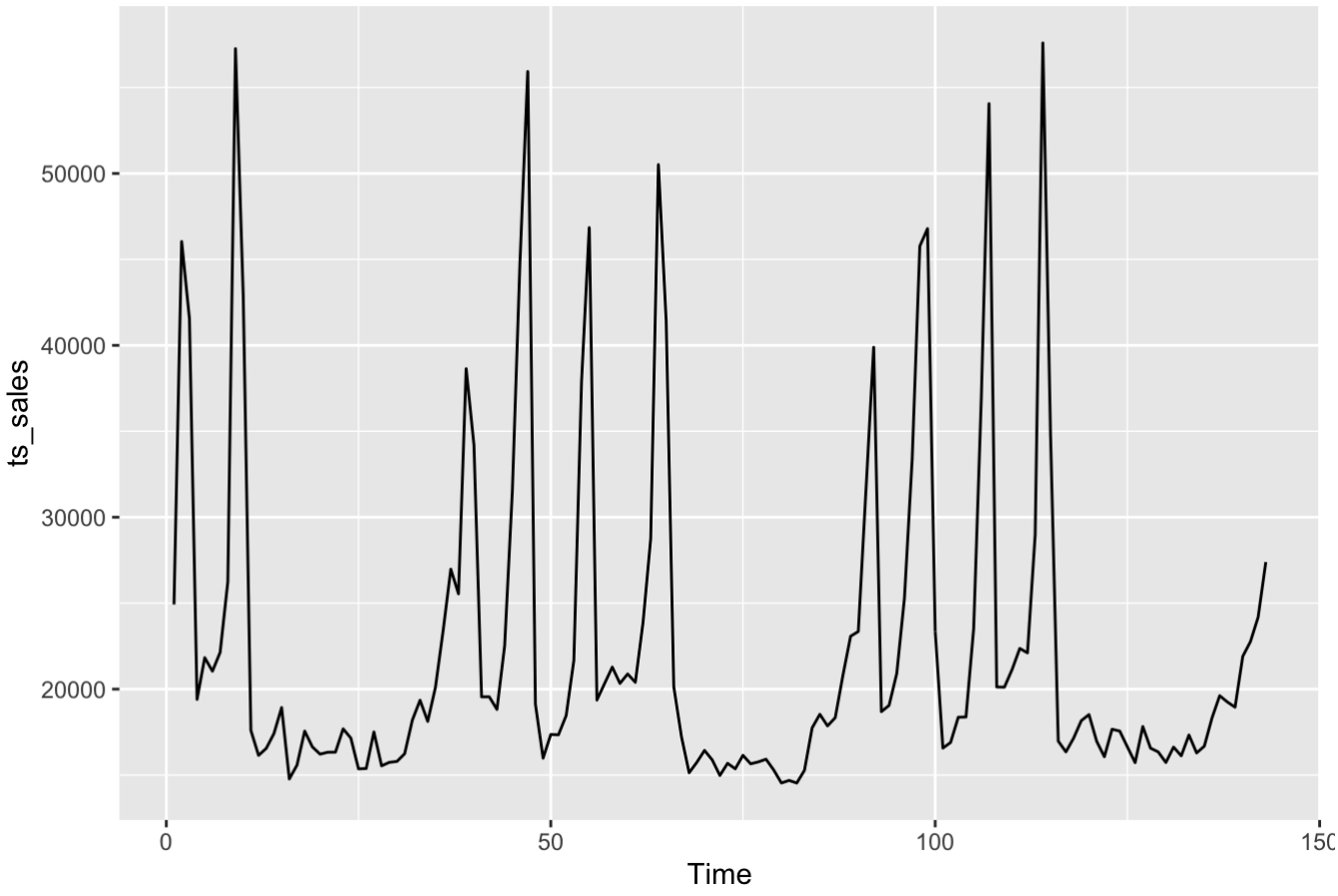
```
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 26887.3168  3111.7833  8.6405 1.148e-13 ***
## trend        -226.7987   142.2170 -1.5947    0.1140
## I(trend^2)      2.2299     1.3642  1.6346    0.1054
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
accuracy(lm_trend2)
```

```
##                        ME    RMSE      MAE       MPE     MAPE     MASE
## Training set 4.547474e-13 10012.2 7379.959 -13.28582 30.73493 1.441637
##                    ACF1
## Training set 0.5695448
```

fit linear trend + season –> Non-seasonal data cannot be modelled using a seasonal

```
autoplot(ts_sales)
```



```
#lm_season = tslm(ts_sales ~ season)

#lm_strend = tslm(ts_sales ~ season + trend)
#fcast_strend=forecast(lm_strend, h=12)
#fit_strend = fitted(fcast_strend)
#fit_strend
#autoplot(fcast_strend) + autolayer(fit_strend) + autolayer(ts_test)
#accuracy(f=fcast_strend, x=ts_test)
```

Multiple Variables

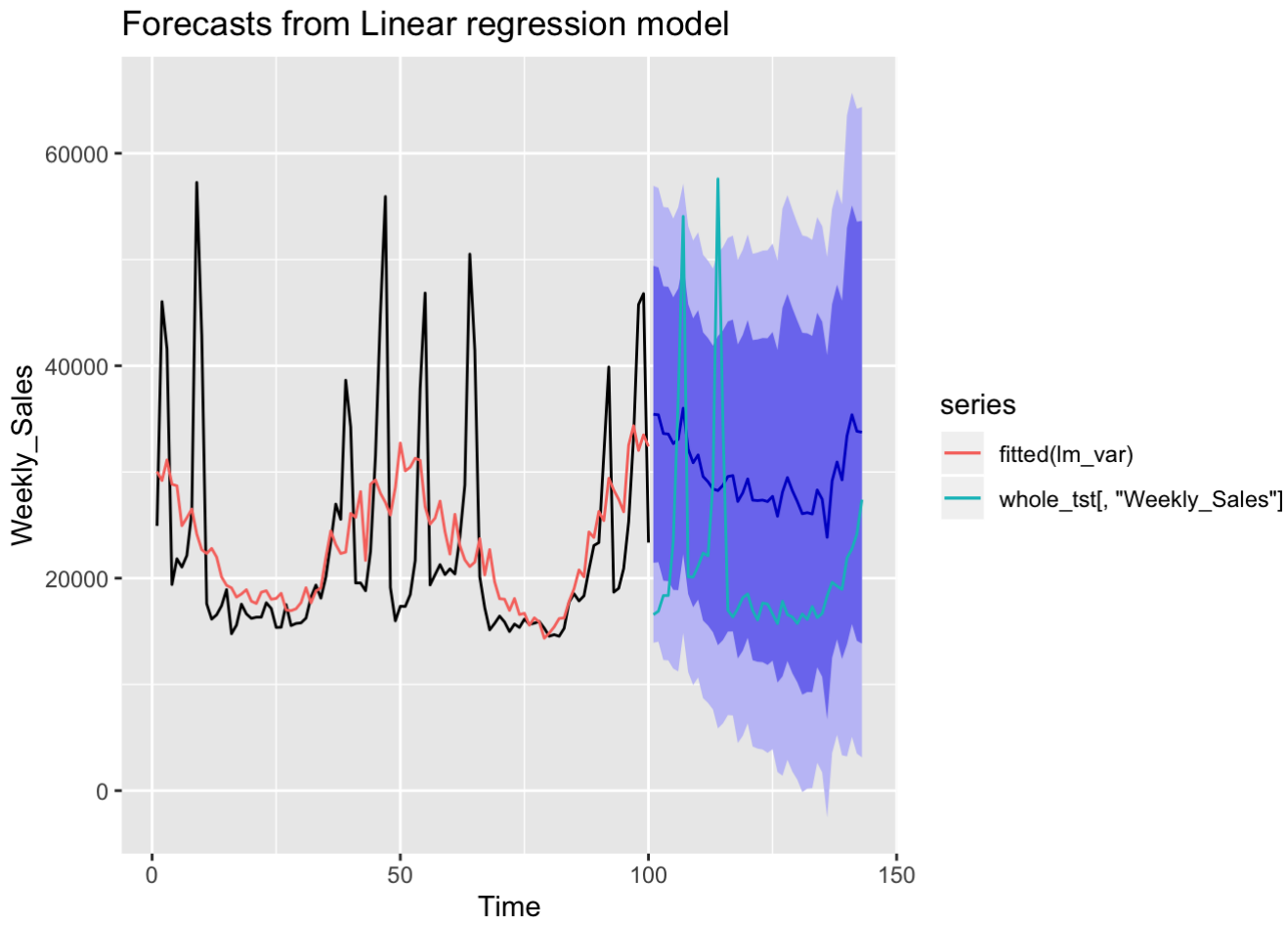
```
data_tsform = ts(data)
whole_tr = window(data_tsform, end=c(100))
whole_tst = window(data_tsform, start=c(101))

lm_var = tslm(Weekly_Sales ~ Temperature+Fuel_Price+CPI+Unemployment+IsHoliday, data = w
hole_tr)
lm_var
```

```
##
## Call:
## tslm(formula = Weekly_Sales ~ Temperature + Fuel_Price + CPI +
##     Unemployment + IsHoliday, data = whole_tr)
##
## Coefficients:
##  (Intercept)   Temperature    Fuel_Price           CPI  Unemployment
##     -68036.4        -323.1       -3231.2         729.0       -4172.1
##    IsHoliday
##      -2260.6
```

```
summary(lm_var)
```

```
##
## Call:
## tslm(formula = Weekly_Sales ~ Temperature + Fuel_Price + CPI +
##     Unemployment + IsHoliday, data = whole_tr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15370  -4418  -1649    187  33098
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -68036.42  115569.31  -0.589    0.557
## Temperature    -323.11      63.52  -5.087 1.86e-06 ***
## Fuel_Price    -3231.19    3644.65  -0.887    0.378
## CPI             728.95     557.04   1.309    0.194
## Unemployment  -4172.12    8547.45  -0.488    0.627
## IsHoliday     -2260.58    3450.87  -0.655    0.514
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9025 on 94 degrees of freedom
## Multiple R-squared:  0.2567, Adjusted R-squared:  0.2172
## F-statistic: 6.493 on 5 and 94 DF,  p-value: 3.155e-05
```

```
lm_varfcast = forecast(lm_var, newdata = as.data.frame(whole_tst))
autoplot(lm_varfcast) + autolayer(fitted(lm_var)) + autolayer(whole_tst[,'Weekly_Sales'
])
```




```
coeftest(lm_var)
```

```
##
## t test of coefficients:
##
##                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  -68036.417 115569.307 -0.5887    0.5575
## Temperature    -323.107     63.517 -5.0869 1.856e-06 ***
## Fuel_Price    -3231.191   3644.648 -0.8866    0.3776
## CPI             728.955    557.044  1.3086    0.1939
## Unemployment  -4172.116   8547.453 -0.4881    0.6266
## IsHoliday     -2260.575   3450.867 -0.6551    0.5140
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
accuracy(lm_var)
```

```
##                         ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -5.463185e-14 8750.152 5809.469 -8.965671 22.37255 1.13485
##                    ACF1
## Training set 0.4108873
```