Retail Sales Forecasting

Contents

Introduction

The training data includes dates, store and item information, whether that item was being promoted, as well as the unit sales. Additional files include supplementary information that may be useful in building your models.

File Descriptions and Data Field Information

`train.csv`

- Training data, which includes the target unit_sales by date, store_nbr, and item_nbr and a unique id to label rows.
- The target unit_sales can be integer (e.g., a bag of chips) or float (e.g., 1.5 kg of cheese). Negative values of unit_sales represent returns of that particular item.
- The onpromotion column tells whether that item_nbr was on promotion for a specified date and store_nbr.
- Approximately 16% of the onpromotion values in this file are NaN.
- NOTE: The training data does not include rows for items that had zero unit_sales for a store/date combination. There is no information as to whether or not the item was in stock for the store on the date, and teams will need to decide the best way to handle that situation. Also, there are a small number of items seen in the training data that aren't seen in the test data.

`stores.csv`

- Store metadata, including city, state, type, and cluster.
- cluster is a grouping of similar stores.

`items.csv`

- Item metadata, including family, class, and perishable.
- NOTE: Items marked as perishable have a score weight of 1.25; otherwise, the weight is 1.0.

`transactions.csv`

- The count of sales transactions for each date, store_nbr combination. Only included for the training data timeframe.

`oil.csv`

- Daily oil price. Includes values during both the train and test data timeframe. (Ecuador is an oil-dependent country and it's economical health is highly vulnerable to shocks in oil prices.)

`holidays_events.csv`

- Holidays and Events, with metadata

- NOTE: Pay special attention to the transferred column. A holiday that is transferred officially falls on that calendar day, but was moved to another date by the government. A transferred day is more like a normal day than a holiday. To find the day that it was actually celebrated, look for the corresponding row where type is Transfer. For example, the holiday Independencia de Guayaquilwas transferred from 2012-10-09 to 2012-10-12, which means it was celebrated on 2012-10-12. Days that are type Bridge are extra days that are added to a holiday (e.g., to extend the break across a long weekend). These are frequently made up by the type Work Day which is a day not normally scheduled for work (e.g., Saturday) that is meant to payback the Bridge.

- Additional holidays are days added a regular calendar holiday, for example, as typically happens around Christmas (making Christmas Eve a holiday).

- Additional Notes

- Wages in the public sector are paid every two weeks on the 15 th and on the last day of the month. Supermarket sales could be affected by this.

- A magnitude 7.8 earthquake struck Ecuador on April 16, 2016. People rallied in relief efforts donating water and other first need products which greatly affected supermarket sales for several weeks after the earthquake.

Time Series forecasting

libraries

```r
library('ggplot2')
library('dplyr')
library('readr')
library('data.table')
library('forecast')
library('prophet')
library('tibble')
library('tidyr')
library('stringr')
library('forcats')
library('lubridate')
```

Load data

training data is 4.7 GB in size with 126 million rows.

```r
set.seed(32)
## reading train data
train_data_f <- fread('../data/raw/train.csv', skip = 36458909,
               col.names = c('id', 'date','store_nbr', 'item_nbr', 'unit_sales', 'onpromotion'))
```

Training data

```r
summary(train_data_f)
```

```
##        id                date             store_nbr       item_nbr
##  Min.   : 36458908   Length:89038132    Min.   : 1.00   Min.   :  96995
##  1st Qu.: 58718441   Class :character   1st Qu.:12.00   1st Qu.: 584188
##  Median : 80977974   Mode  :character   Median :28.00   Median :1089044
##  Mean   : 80977974                      Mean   :27.65   Mean   :1059528
##  3rd Qu.:103237506                      3rd Qu.:43.00   3rd Qu.:1459225
##  Max.   :125497039                      Max.   :54.00   Max.   :2127114
##    unit_sales        onpromotion
##  Min.   :-15372.00   Mode :logical
##  1st Qu.:     2.00   FALSE:81596506
##  Median :     4.00   TRUE :7441626
##  Mean   :     8.39
##  3rd Qu.:     9.00
##  Max.   : 89440.00
```

```r
gc()
```

```
##             used   (Mb) gc trigger   (Mb)  max used   (Mb)
## Ncells   1217160   65.1    2109738  112.7   2109738  112.7
## Vcells 397162882 3030.2  830918542 6339.5 797843106 6087.1
```

```r
glimpse(train_data_f)
```

```
## Observations: 89,038,132
## Variables: 6
## $ id         <int> 36458908, 36458909, 36458910, 36458911, 36458912, ...
## $ date       <chr> "2014-12-02", "2014-12-02", "2014-12-02", "2014-12...
## $ store_nbr  <int> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10...
## $ item_nbr   <int> 464336, 464339, 464374, 464906, 464940, 467808, 46...
## $ unit_sales <dbl> 4, 5, 4, 1, 7, 34, 1, 7, 2, 4, 2, 1, 1, 3, 14, 5, ...
## $ onpromotion <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, F...
```

time series data for a store,item

Top 2 items by sales: 1503844, 1047679 Top 2 store_nbr by sales: 44, 45 Considering time series of store 44, Item 1503844

```
train_data_f[, ':='(
  date = ymd(date, tz = NULL),
  store_item = paste(store_nbr, item_nbr, sep="_")
)]

data_wide <- dcast(train_data_f, store_item ~ date, value.var = "unit_sales", fill = 0)
```

```
data_item <- data_wide %>%
    filter(store_item == "44_1503844")
```

```
  h <- 16
  frequency <- 7
  date_index_test <- tail(colnames(data_wide), 16)

  data_df <- melt(data_item, id.vars = c("store_item"))

  store_item_val <- data_df$store_item[1]
  data_df$store_item <- NULL
```

```
  colnames(data_df) <- c("ds", "y")

  # Date column handling
  data_ts <- data_df
  data_ts <- data_ts %>%
              select(-ds)


  test_index <- tail(rownames(data_ts), h) %>% as.numeric()
  test_data <- data_ts %>% slice(test_index) %>% `row.names<-`(test_index)
  train_data <- data_ts %>% slice(-test_index)

  # Converts train data to time.series object
  train_ts <- ts(train_data, frequency = frequency, start = 1)
```

Arima forcast

```r
fit_arima <- auto.arima(x = train_ts)
fit_arima
```

```
## Series:
## ARIMA(2,0,5)(0,1,2)[7]
##
## Coefficients:
##          ar1      ar2      ma1      ma2     ma3      ma4      ma5     sma1
##       1.1092  -0.1347  -0.7431  -0.1930  0.1851  -0.0024  -0.1172  -0.5995
## s.e.  0.2049   0.1959   0.2025   0.1275  0.0665   0.0475   0.0374   0.0362
##          sma2
##       -0.0817
## s.e.   0.0348
##
## sigma^2 estimated as 27279:  log likelihood=-6275.37
## AIC=12570.73   AICc=12570.96   BIC=12619.42
```

```r
forecast_arima <- forecast(fit_arima, h = h)
forecast_arima
```

```
##          Point Forecast    Lo 80     Hi 80     Lo 95     Hi 95
## 139.4286       561.1213 349.4570  772.7856 237.40866  884.8339
## 139.5714       513.1366 287.7310  738.5422 168.40844  857.8647
## 139.7143       859.8978 633.8814 1085.9142 514.23561 1205.5600
## 139.8571       459.6699 228.7863  690.5536 106.56386  812.7760
## 140.0000       560.7540 324.6086  796.8995 199.60070  921.9074
## 140.1429       845.6041 608.2587 1082.9495 482.61564 1208.5926
## 140.2857      1180.3778 942.2110 1418.5446 816.13310 1544.6224
## 140.4286       580.7255 321.0523  840.3986 183.58969  977.8613
## 140.5714       492.8263 228.5468  757.1059  88.64559  897.0071
## 140.7143       858.4678 593.0826 1123.8530 452.59611 1264.3395
## 140.8571       470.6846 202.8802  738.4890  61.11314  880.2561
## 141.0000       549.6415 279.3776  819.9054 136.30856  962.9745
## 141.1429       832.3683 560.8962 1103.8404 417.18756 1247.5491
## 141.2857      1188.1436 915.6581 1460.6292 771.41300 1604.8743
## 141.4286       582.0387 295.0526  869.0249 143.13131 1020.9462
## 141.5714       489.4470 198.7008  780.1932  44.78903  934.1050
```

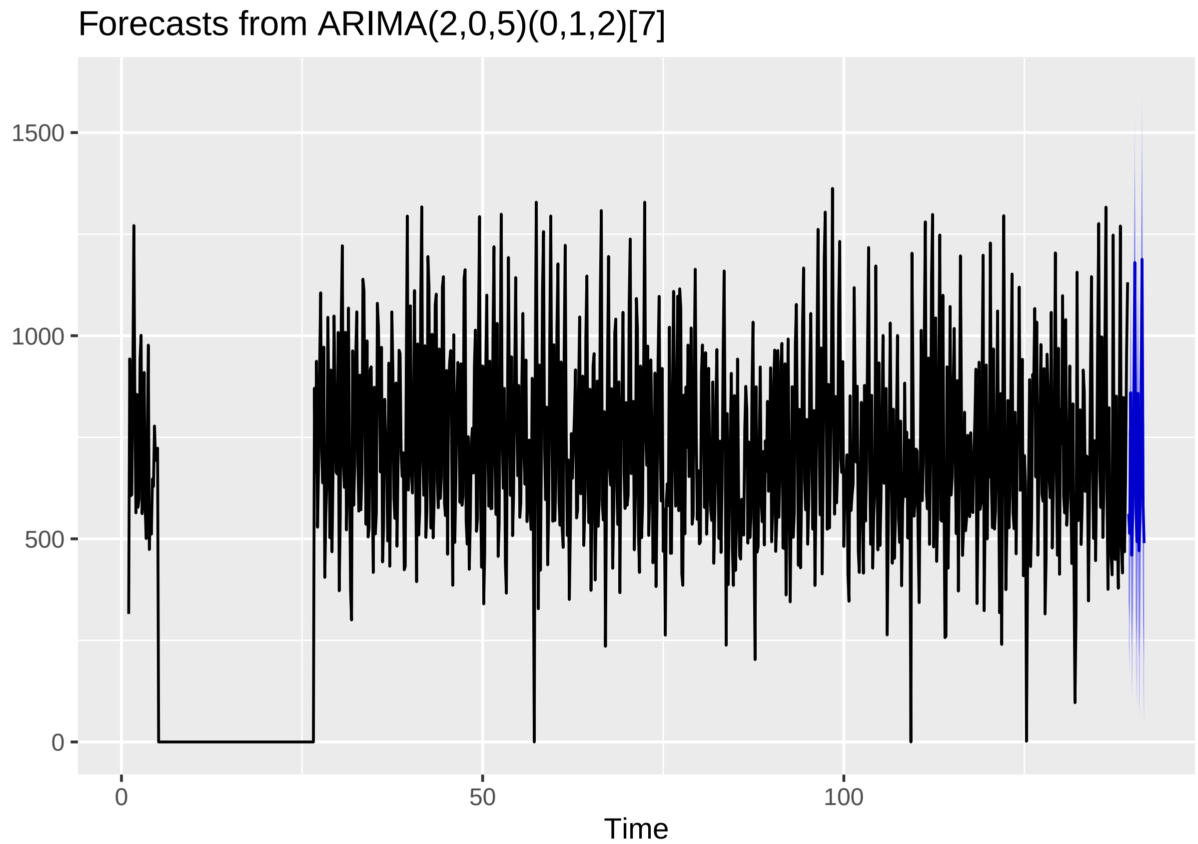
accuracy metrics

```r
t_data <- test_data$y %>% as.numeric()
accuracy_arima <- forecast::accuracy(forecast_arima, t_data)
accuracy_arima
```

```
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.2869613 163.7933 108.6452      NaN      Inf 0.4467585
## Test set     40.5820635 121.1368 102.8731 5.150944 15.00143 0.4230229
##                       ACF1
## Training set -0.0007281439
## Test set                NA
```

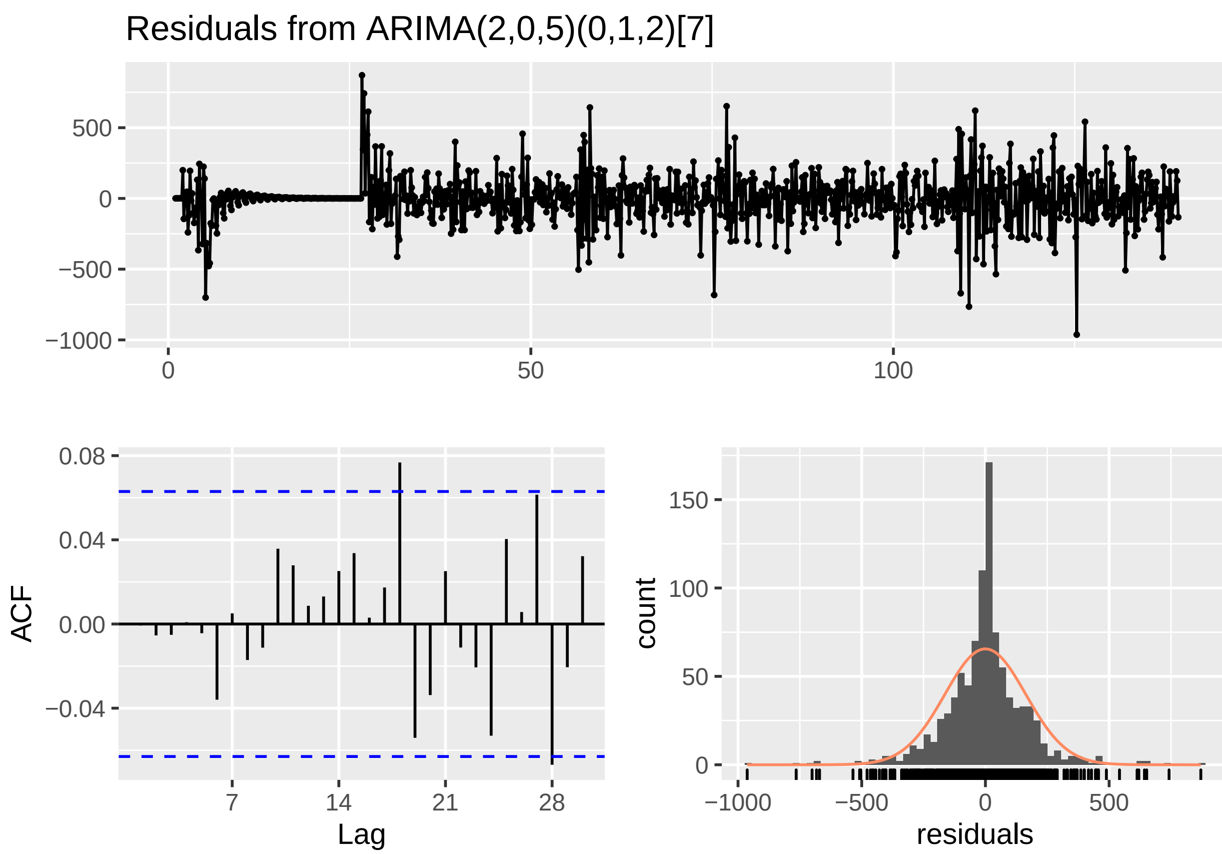
arima forecast

```r
autoplot(forecast_arima)
```



check residuals

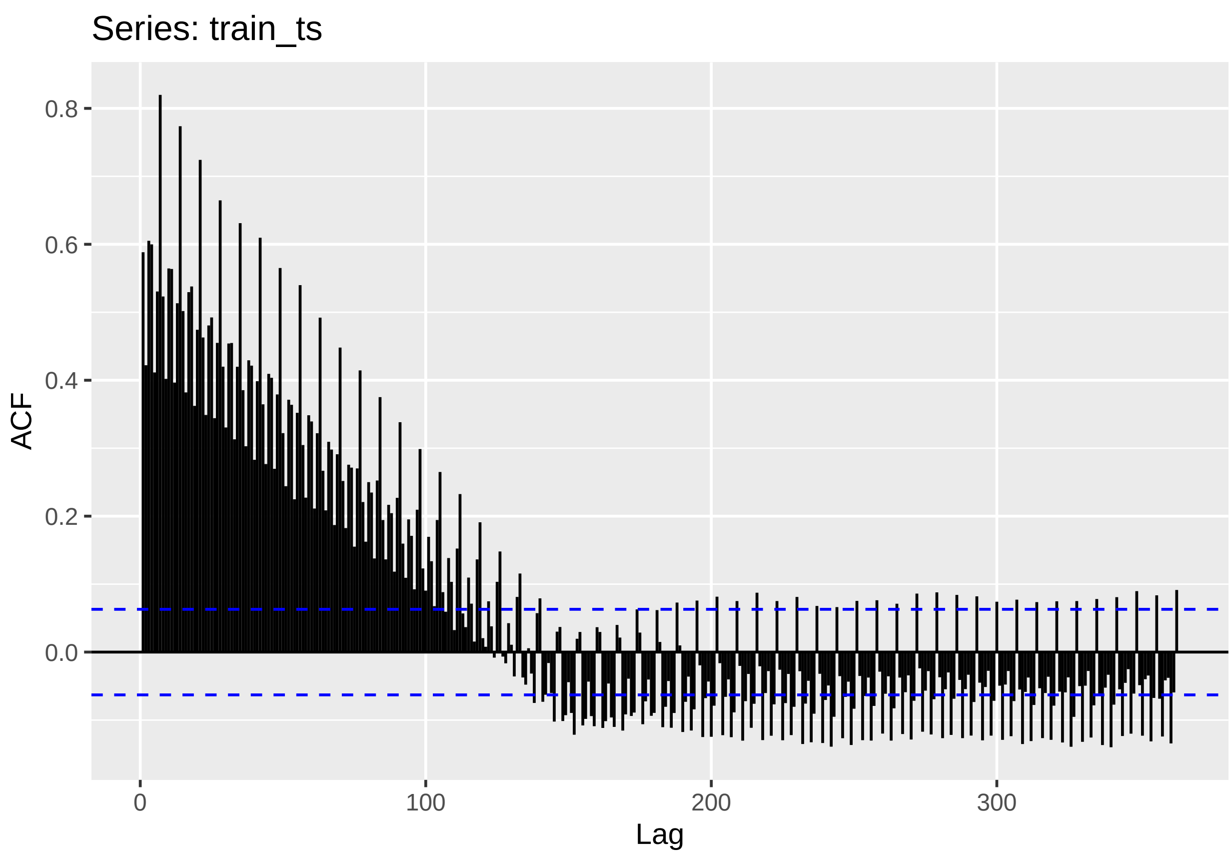
```r
checkresiduals(fit_arima)
```



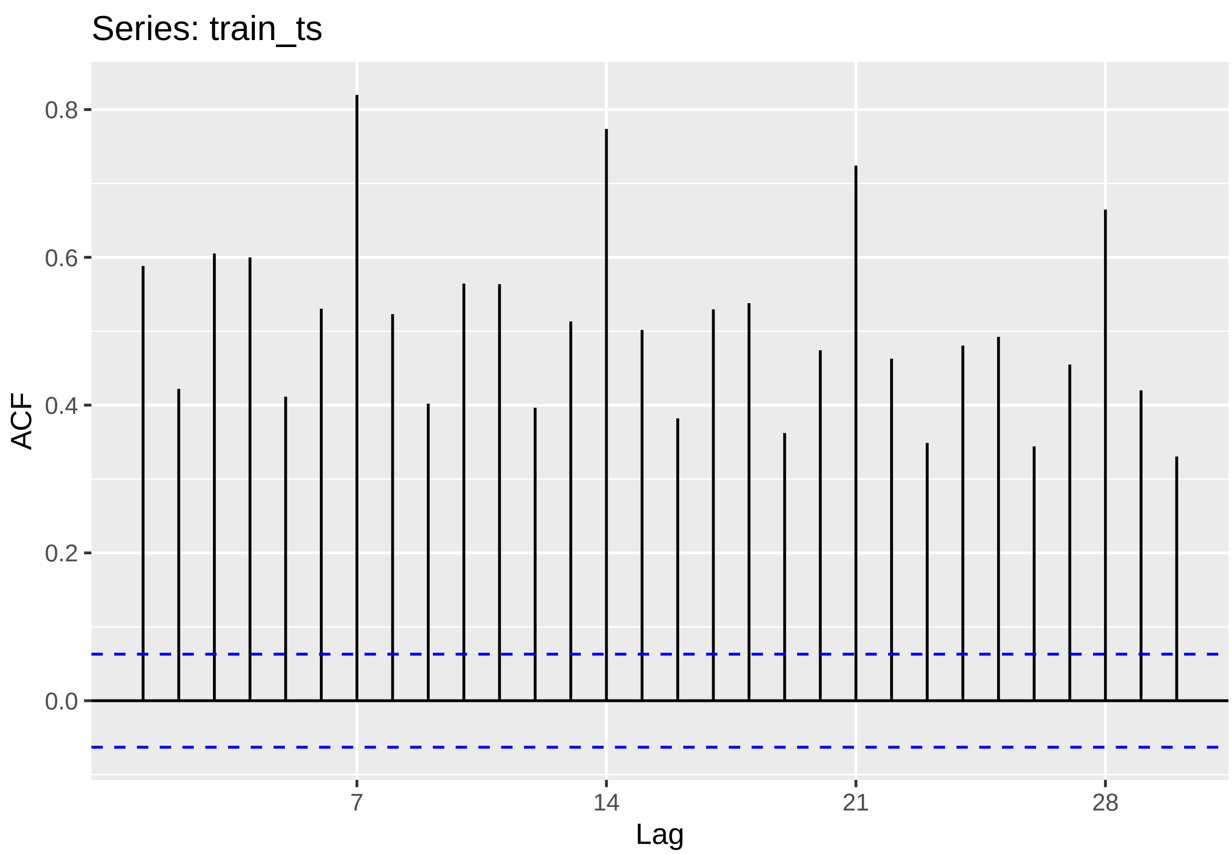

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,5)(0,1,2)[7]
## Q* = 4.658, df = 5, p-value = 0.459
##
## Model df: 9.   Total lags used: 14
```

##ACF plots

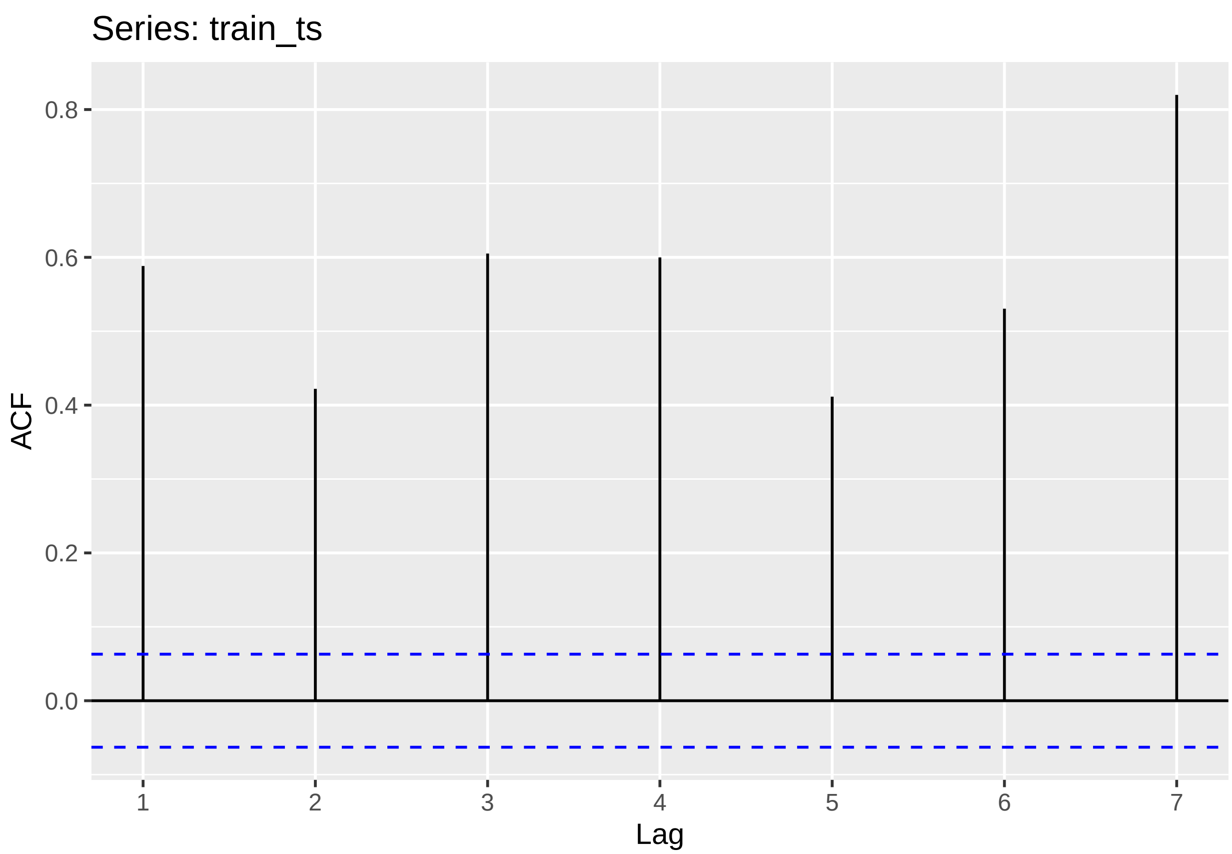
```
ggAcf(train_ts, lag = 363)
```




```r
ggAcf(train_ts, lag = 30)
```




```r
ggAcf(train_ts, lag = 7)
```



prophet

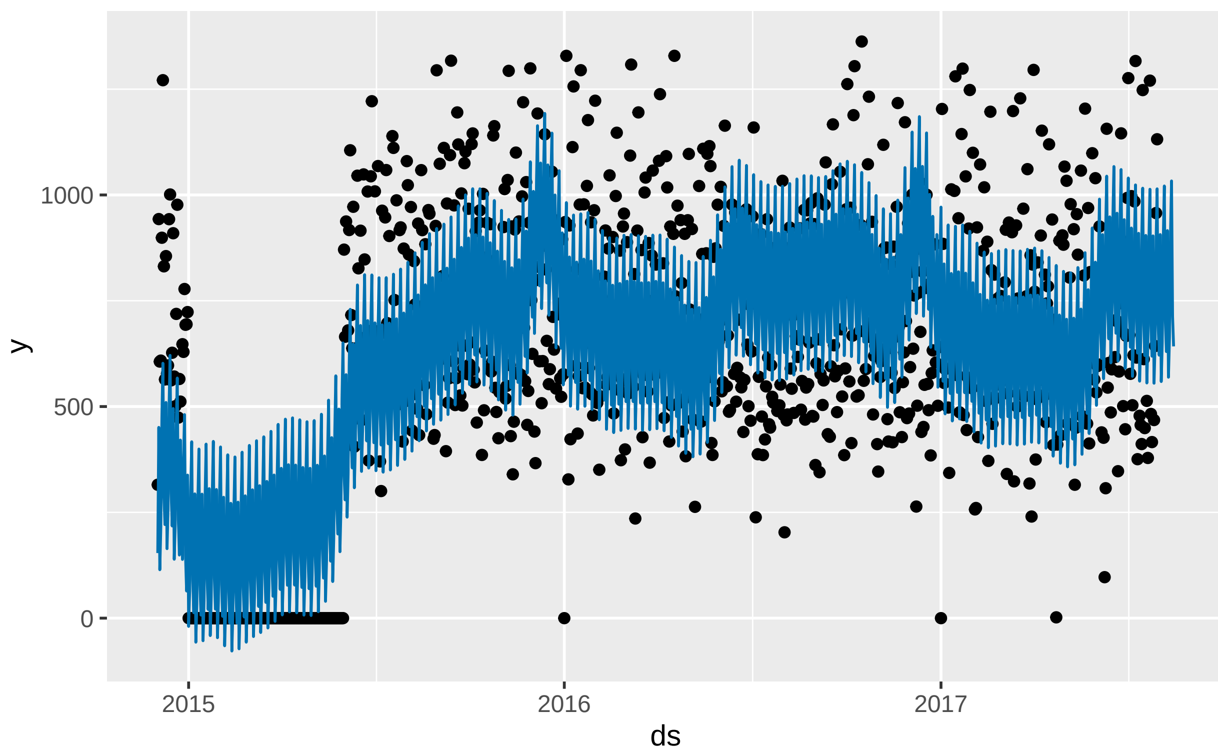
```
test_index <- tail(rownames(data_df), h) %>% as.numeric()
test_data <- data_df %>% slice(test_index) %>% `row.names<-`(test_index)
train_data <- data_df %>% slice(-test_index)


df = train_data
yearly.seasonality=TRUE
weekly.seasonality = TRUE
daily.seasonality=FALSE

fit_prophet <- prophet(df = train_data, yearly.seasonality=TRUE, weekly.seasonality = TRUE,
             daily.seasonality=FALSE)
future <- make_future_dataframe(fit_prophet, periods = 16)
forecast <- predict(fit_prophet, future)
forecast <- forecast[c('ds', 'yhat')]
forecast$store_item <- store_item_val
```

prophet forecast plot

```r
plot(fit_prophet, forecast)
```

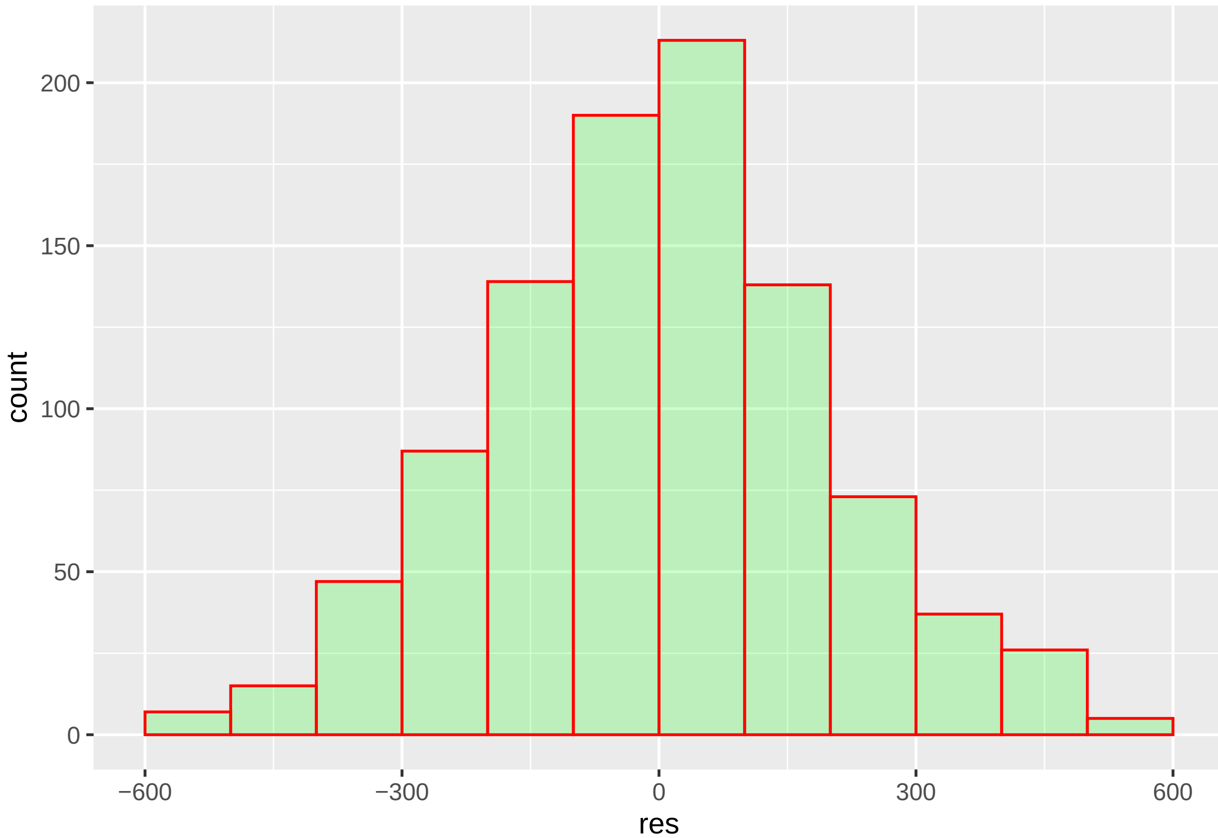


prophet residuals plot

```r
colnames(data_df) <- c("ds", "y")
residuals_f = forecast['yhat'] - data_df['y']
colnames(residuals_f) <- c("res")
head(residuals_f)
```

```
##         res
## 1 -161.3908
## 2 -492.2163
## 3 -491.9077
## 4 -391.6313
## 5 -426.6077
## 6 -668.7870
```

```r
ggplot(residuals_f, aes(res)) +
geom_histogram( breaks = seq(-600, 600, by=100),
                col="red",
                fill="green",
                alpha=.2)
```




```
forecast <- forecast[forecast$ds >= ymd("2017-07-30"),]

t_data <- test_data$y %>% as.numeric()
accuracy_prophet <- forecast::accuracy(forecast$yhat, t_data)
accuracy_prophet
```

```
##                 ME     RMSE      MAE       MPE     MAPE
## Test set -51.62214 276.8998 241.3559 -17.49498 38.87372
```

Accuracy measures (Multiple time series)

Accuracy measures for the 18310 time series considering Top 5 store numbers (44, 45, 47, 3, 49) by total sales as the baseline for processing multiple time series.

- `"mean_rmse",      "time_taken"`
- mean 7.45920546451115, "40.76 mins"
- ets 8.0311892817118, "48.55 mins"
- auto.arima 6.589306454993, "7.83 hours"
- prophet 7.49642904779891, "3.24 hours"