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    1217158   65.1    2109754  112.7   2109754  112.7
## Vcells 397162855 3030.2  830918504 6339.5 797843079 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 stores by sales: 44, 45 Considering time series of store 44, Item 1503823

```
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_1503823")
```

```
  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(2,0,2)(2,1,1)[7]
##
## Coefficients:
##          ar1     ar2      ma1      ma2    sar1    sar2     sma1
##       0.3031  0.5758  -0.0807  -0.4594  0.0364  0.0056  -0.7325
## s.e.  0.4367  0.4028   0.4347   0.3072  0.0536  0.0443   0.0429
##
## sigma^2 estimated as 33.24:  log likelihood=-3049.12
## AIC=6114.25   AICc=6114.4   BIC=6153.2
```

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

```
##          Point Forecast      Lo 80    Hi 80      Lo 95    Hi 95
## 139.4286       8.553684  1.1646239 15.94274 -2.7469083 19.85428
## 139.5714       5.866297 -1.7032683 13.43586 -5.7103539 17.44295
## 139.7143      13.390628  5.7001786 21.08108  1.6291004 25.15216
## 139.8571       6.916667 -0.8927458 14.72608 -5.0267993 18.86013
## 140.0000       7.404020 -0.4960974 15.30414 -4.6781671 19.48621
## 140.1429      18.893021 10.9105363 26.87551  6.6848642 31.10118
## 140.2857      14.822266  6.7730822 22.87145  2.5121015 27.13243
## 140.4286       8.914853  0.2462449 17.58346 -4.3426389 22.17234
## 140.5714       6.026591 -2.7526820 14.80586 -7.4001484 19.45333
## 140.7143      13.678302  4.8115573 22.54505  0.1177864 27.23882
## 140.8571       6.992968 -1.9515212 15.93746 -6.6864478 20.67238
## 141.0000       7.518037 -1.4904752 16.52655 -6.2592934 21.29537
## 141.1429      19.002492  9.9383615 28.06662  5.1401004 32.86488
## 141.2857      14.991184  5.8803567 24.10201  1.0573758 28.92499
## 141.4286       9.071390 -0.5071531 18.64993 -5.5777273 23.72051
## 141.5714       6.144710 -3.5159768 15.80540 -8.6300356 20.91946
```

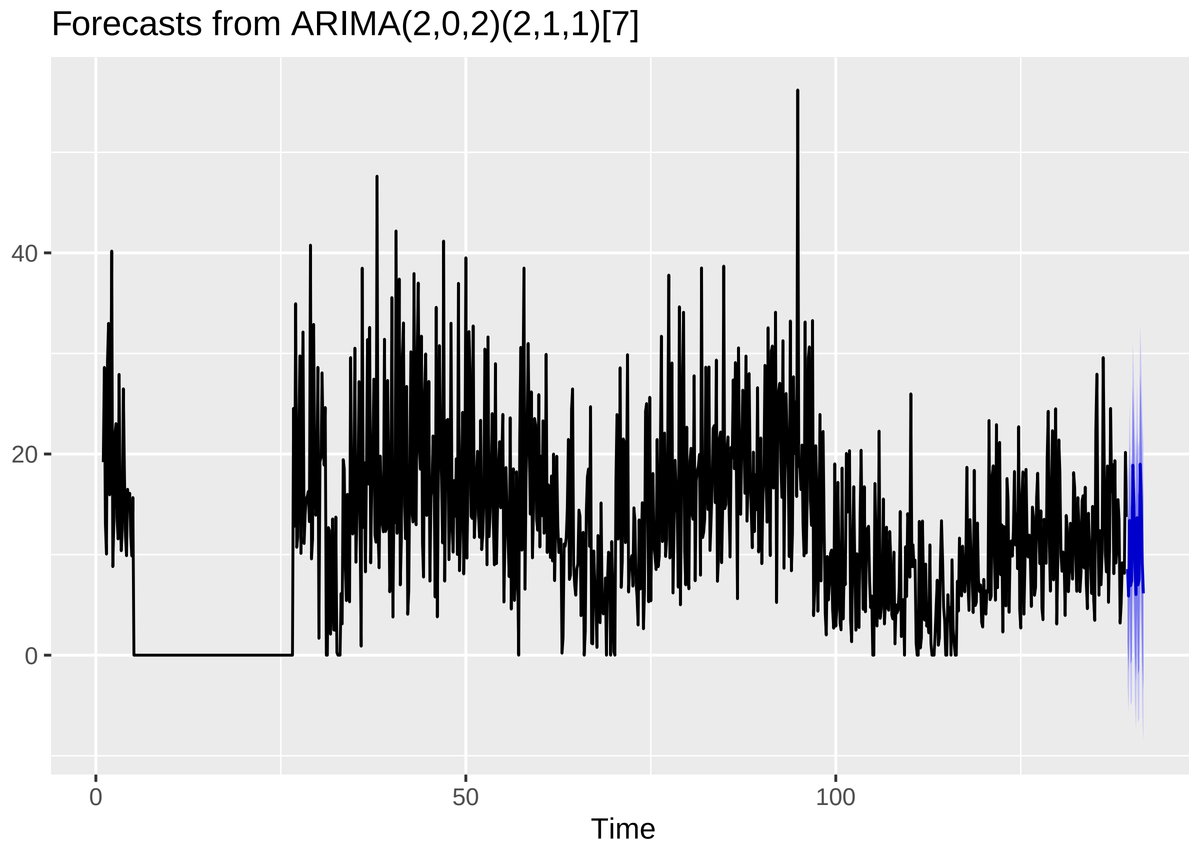
accuracy metrics

```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 -0.06418749 5.723912 4.014630      NaN      Inf 0.5485505
## Test set      0.79180558 3.405526 2.719444 4.590383 23.82331 0.3715791
##                     ACF1
## Training set -0.01706721
## Test set              NA
```

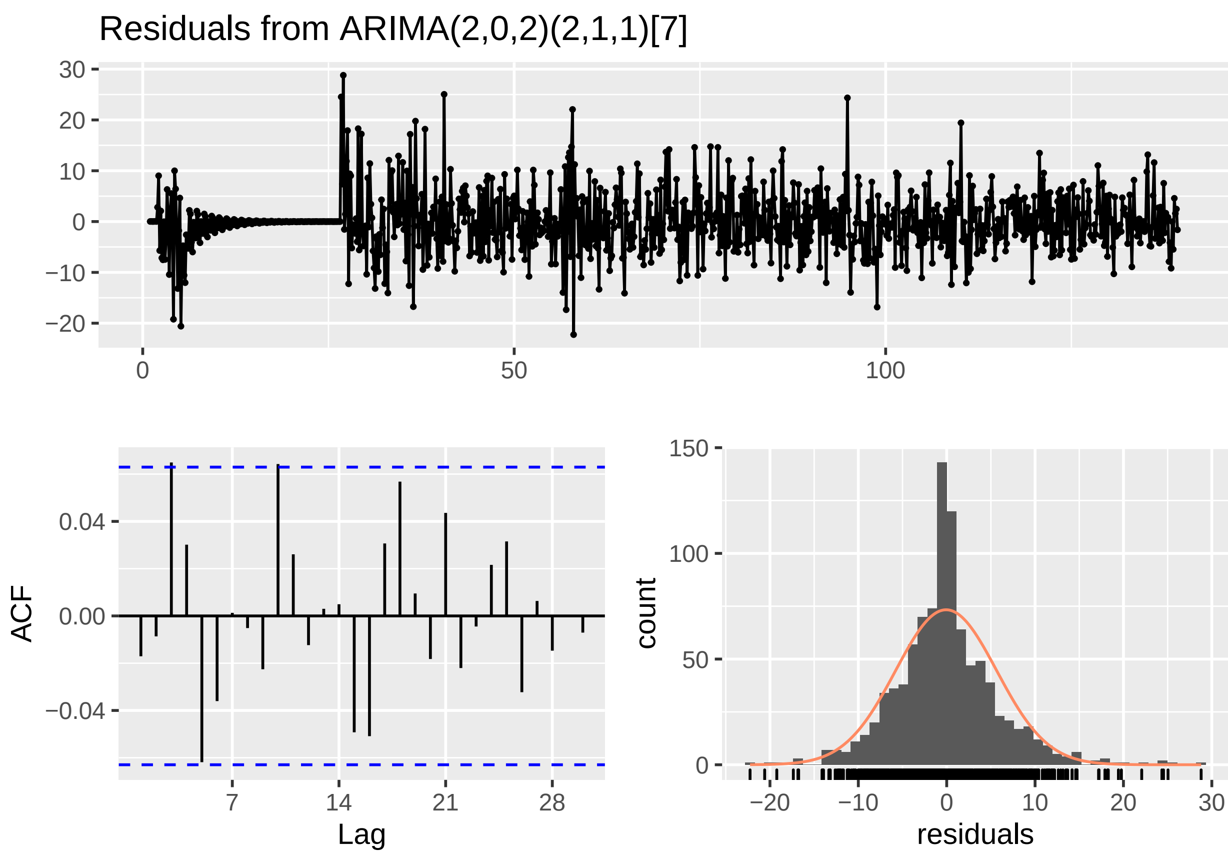
arima forecast

```
autoplot(forecast_arima)
```



check residuals

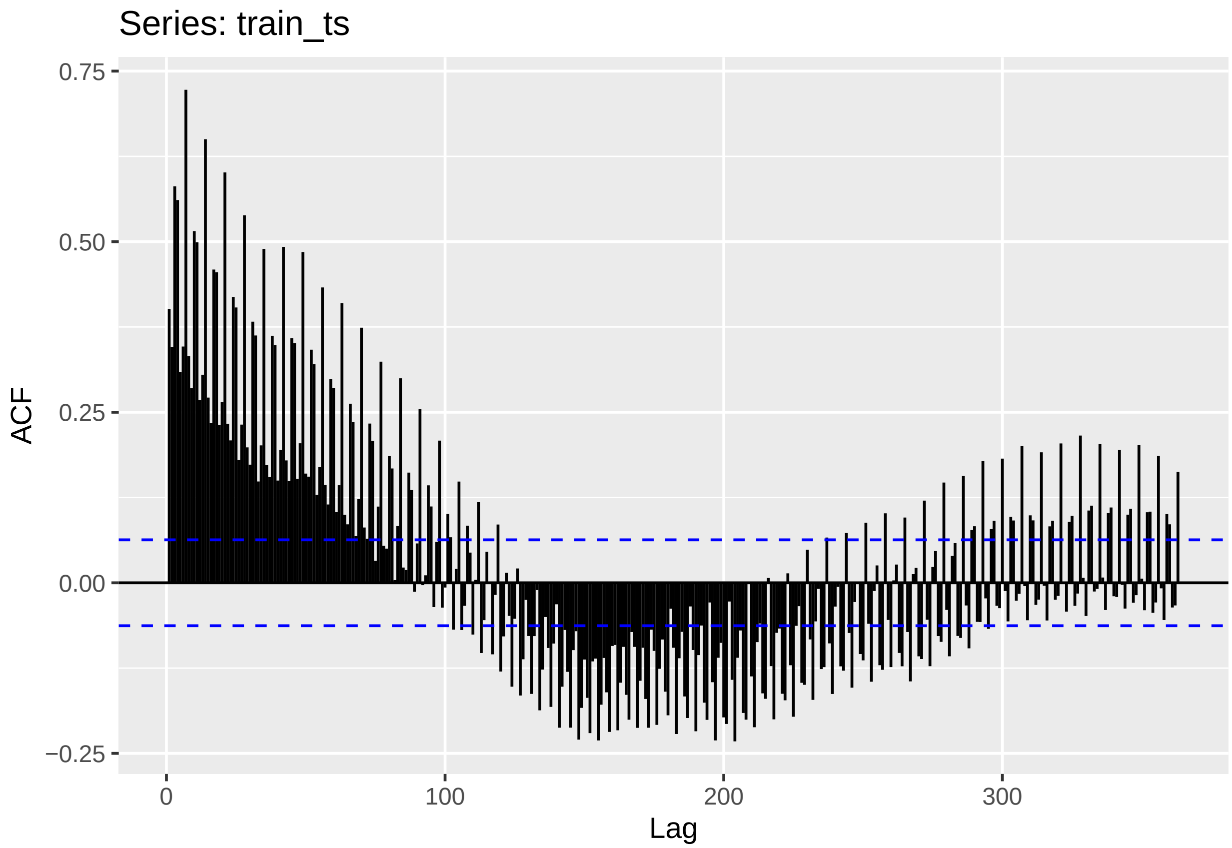
```
checkresiduals(fit_arima)
```



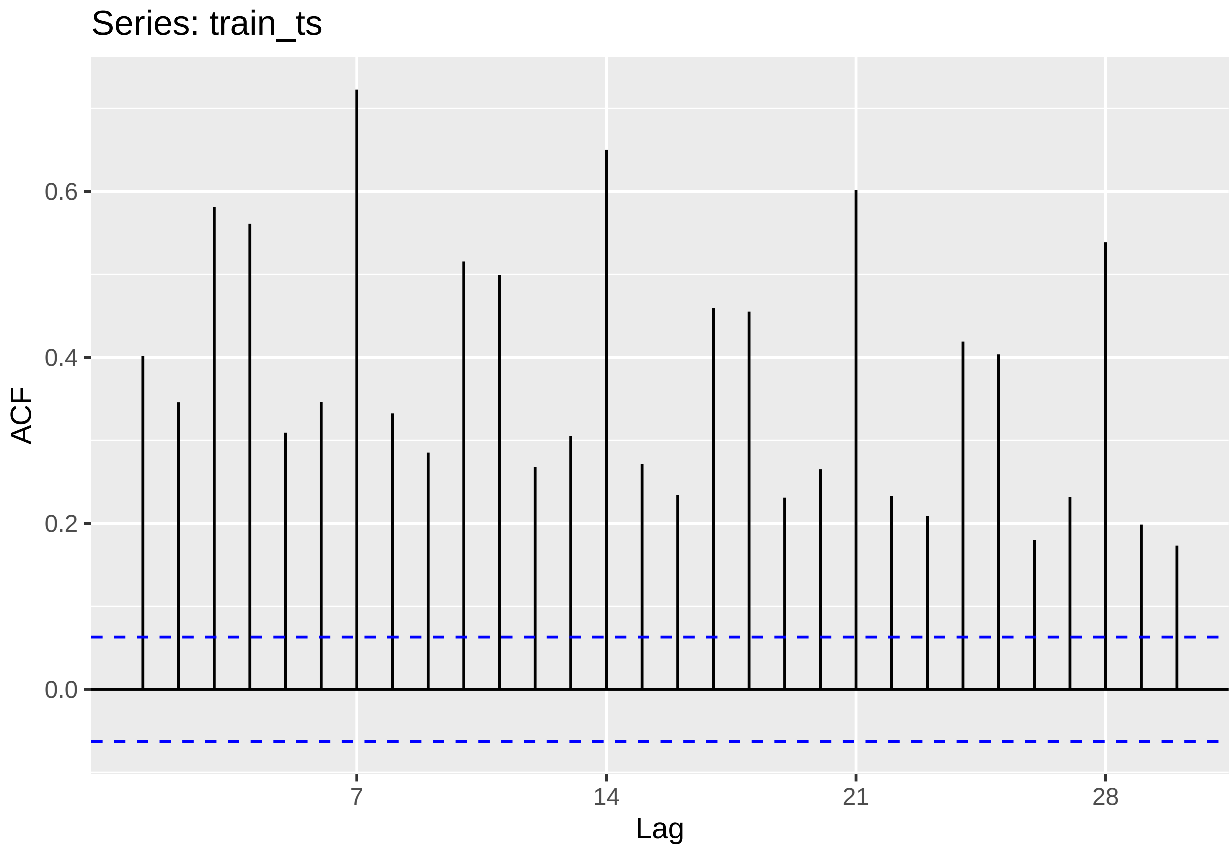

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,2)(2,1,1)[7]
## Q* = 15.78, df = 7, p-value = 0.0272
##
## Model df: 7.   Total lags used: 14
```

##ACF plots

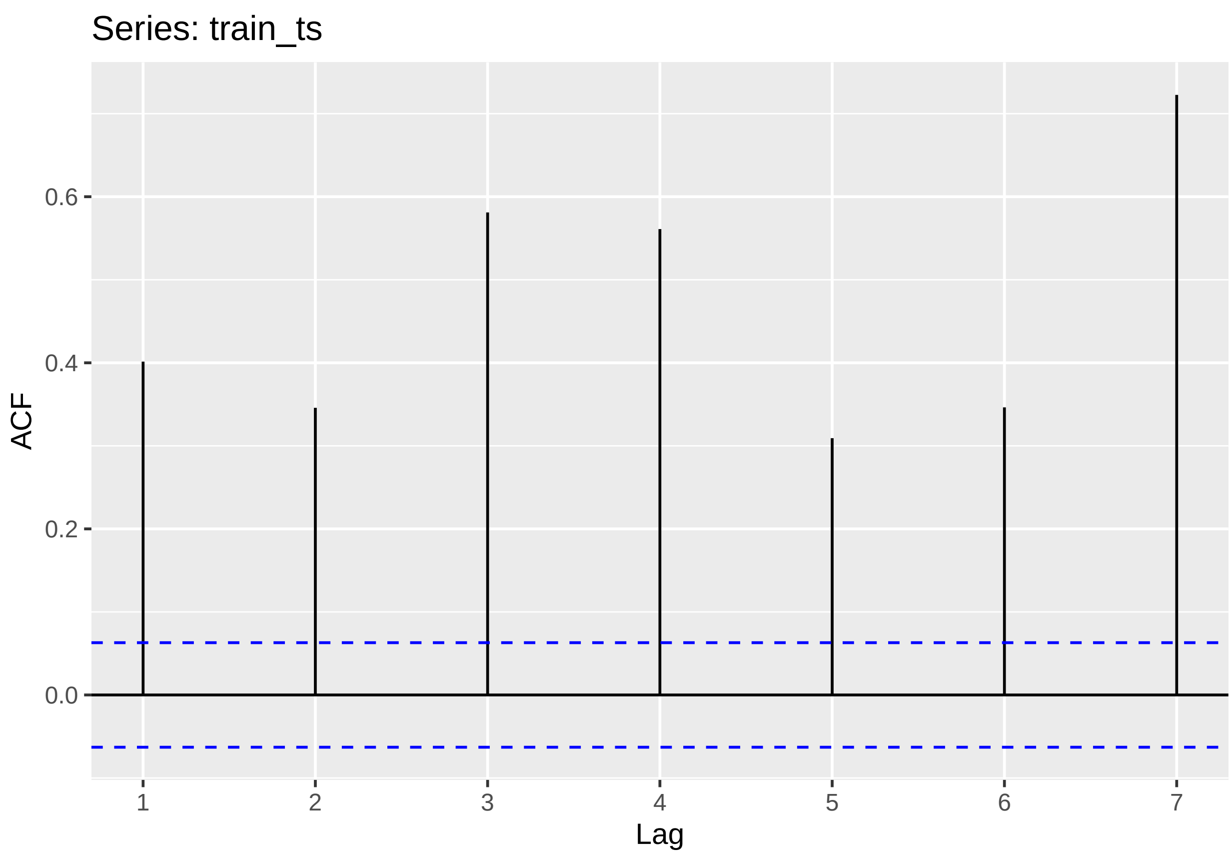
```
ggAcf(train_ts, lag = 363)
```




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




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



prophet

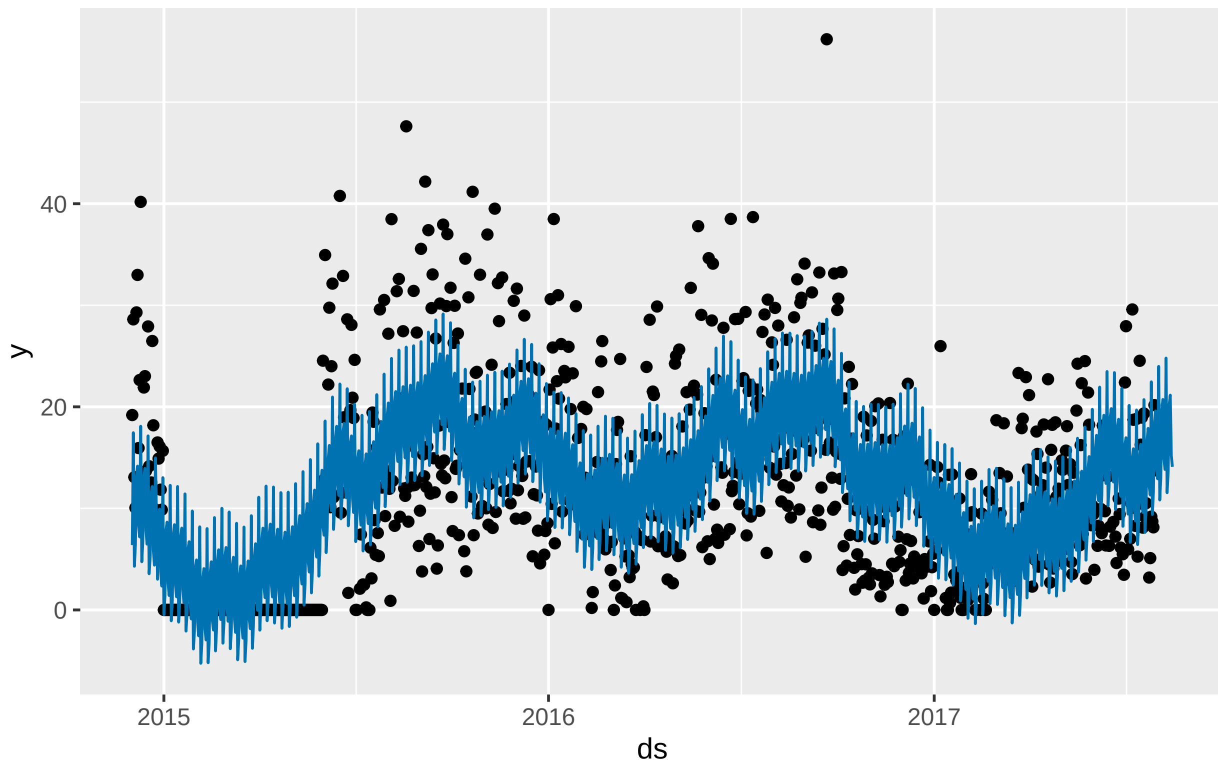
```
test_index <- tail(rownames(data_df), h) %>% as.numeric()
test_data <- data_df %>% slice(test_index) %>% `row.names<-`(test_index)
train_data <- data_df %>% slice(-test_index)


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)
```

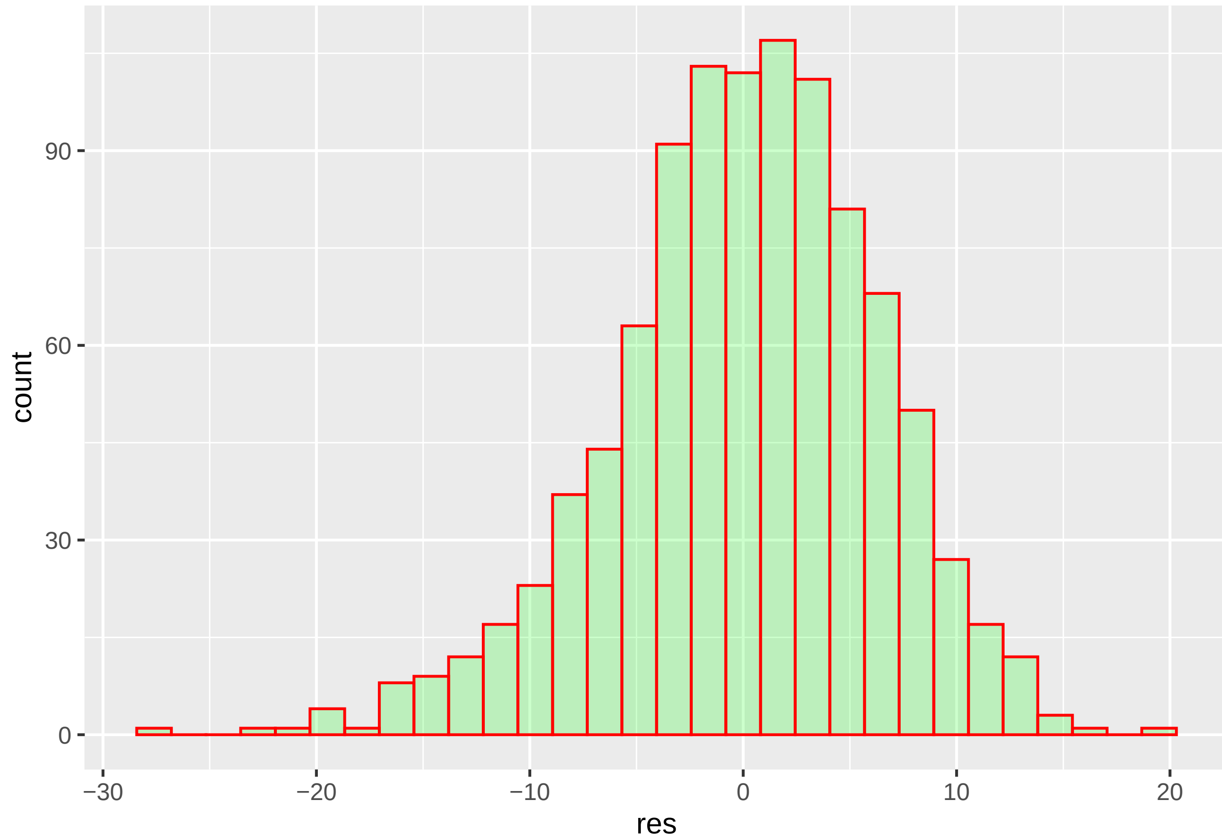


prophet residuals plot

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

```
##          res
## 1 -12.804108
## 2 -11.158240
## 3  -8.775529
## 4  -3.602418
## 5 -16.138064
## 6 -18.823040
```

```r
ggplot(residuals_f, aes(res)) +
geom_histogram( 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 -5.559209 8.726081 7.045657 -74.55215 83.42445
```

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"