Weavenow’s Case Study Report

Duc Anh Bui (Tyler)

2022-09-26

Task

We would like to how to predict demand in retail environments using some standard datasets. You are requested to do the following steps:

Data processing and modeling factors

Common demand prediction methods (including feature selection and regularization)

Evaluation and visualization

Loading libraries

```
library(tidyverse)
library(dplyr)
library(tseries)
library(forecast)
library(randomForest)
library(prophet)
library(reshape2)
```

Data processing and cleaning

```
data <- read.csv("data_raw.csv")
df <- as.data.frame(data)
week <- as.Date(sapply(strsplit(as.character(df$week.sku.weekly_sales.feat_main_page.color.price.vendor.functiona
lity), "\\,"), '[',1), "%Y-%m-%d")
sku <- as.numeric(sapply(strsplit(as.character(df$week.sku.weekly_sales.feat_main_page.color.price.vendor.functio
nality), "\\,"), '[',2))
weekly_sales <- as.numeric(sapply(strsplit(as.character(df$week.sku.weekly_sales.feat_main_page.color.price.vendo
r.functionality), "\\,"), '[',3))
feat_main_page <- sapply(strsplit(as.character(df$week.sku.weekly_sales.feat_main_page.color.price.vendor.functio
nality), "\\,"), '[',4)
color <- sapply(strsplit(as.character(df$week.sku.weekly_sales.feat_main_page.color.price.vendor.functionality),
"\\,"), '[',5)
price <- as.numeric(sapply(strsplit(as.character(df$week.sku.weekly_sales.feat_main_page.color.price.vendor.funct
ionality), "\\,"), '[',6))
vendor <- as.numeric(sapply(strsplit(as.character(df$week.sku.weekly_sales.feat_main_page.color.price.vendor.func
tionality), "\\,"), '[',7))
functionality <- sapply(strsplit(as.character(df$week.sku.weekly_sales.feat_main_page.color.price.vendor.function
ality), "\\,"), '[',8)
df <- data.frame(week, sku, weekly_sales, feat_main_page, color, price, vendor, functionality)

write_csv(df,"processed_data.csv")
```

Helper function to monitor the model’s performence

```
options(scipen=999)
validation_stats_script <- function(y_actual, y_hat) {
  r2 <- cor(y_actual,y_hat)^2
  mse <- mean((y_actual - y_hat)^2)
  mape <- mean(abs((y_actual-y_hat)/y_actual))*100
  mae <- mean(abs(y_actual - y_hat))
  rmse <- sqrt(mse)
  res <- list("r2" = signif(r2,2),
              "mse" = signif(mse,2),
              "mape" = signif(mape,2),
              "mae" = signif(mae,2),
              "rmse" = signif(rmse,2))
  return(res)
}
```

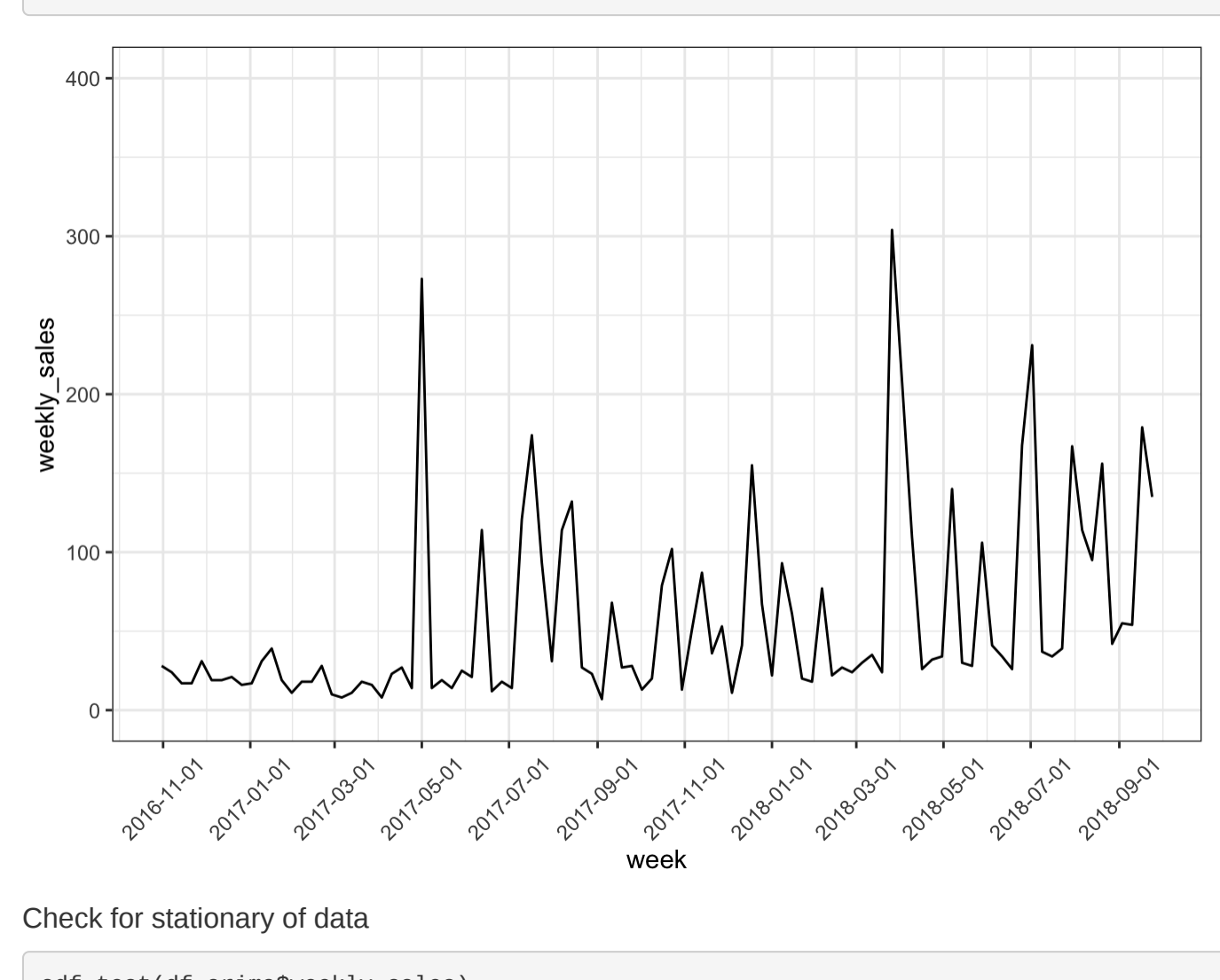
Part 1: Weekly Sales of SKU = 11

ARIMA Model

Although it is widely known that Arima does not perform well with weekly data, it still is one of the basic/ traditional forecasting model for time series forecasting.

More data processing and plotting

```
df_arima <- filter(df, sku == 11)
keep <- c("week", "weekly_sales")
df_arima <- df_arima[keep]
ggplot(df_arima, aes(x = week, y = weekly_sales)) +
  geom_line() + ylim(0, 400) +
  scale_x_date(date_labels = "%Y-%m-%d", date_breaks = "2 month") +
  theme_bw() + theme(legend.title = element_blank(),
                     axis.text.x  = element_text(angle=45, vjust=0.5))
```



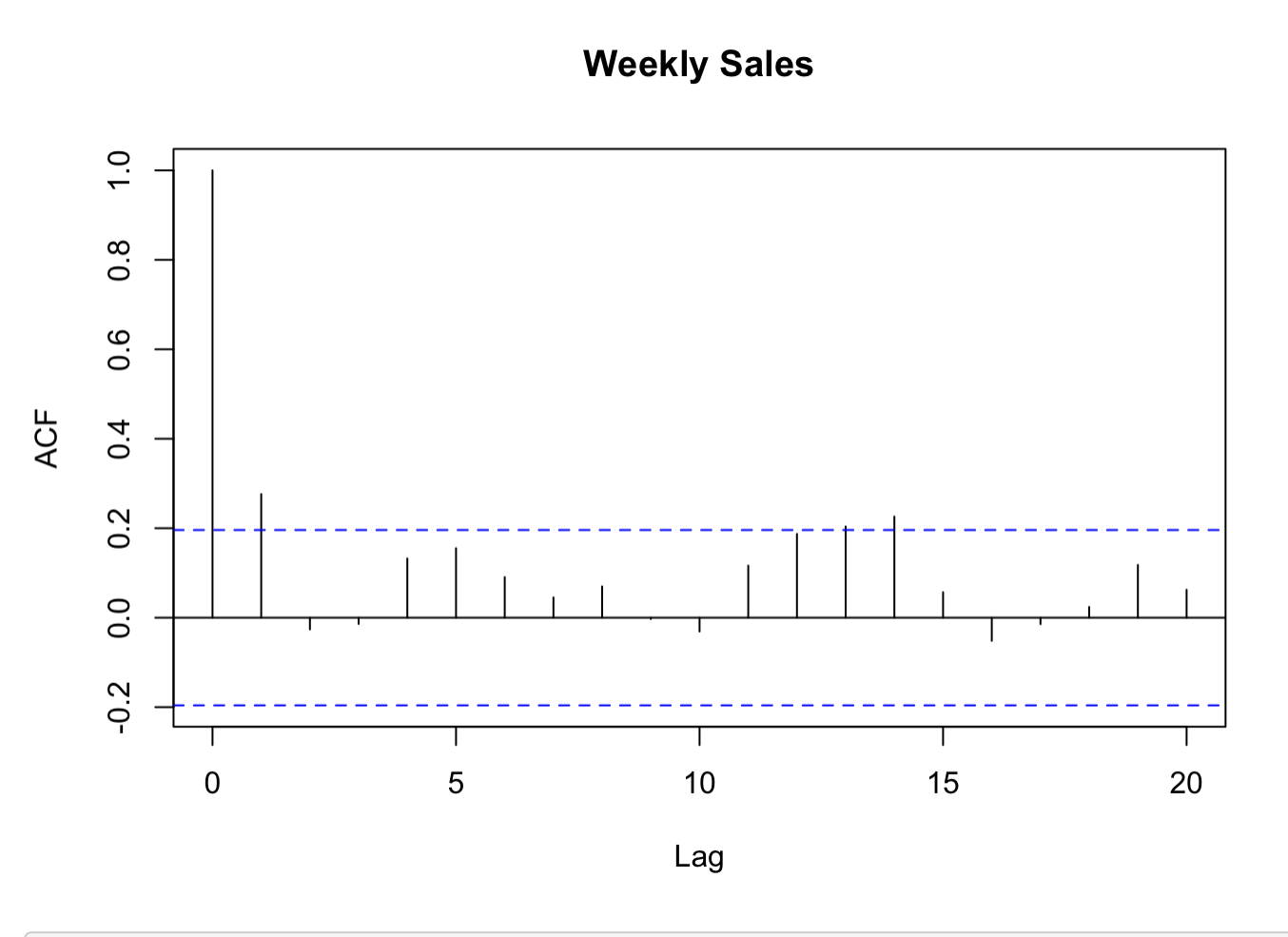
Check for stationary of data

```
adf.test(df_arima$weekly_sales)
```

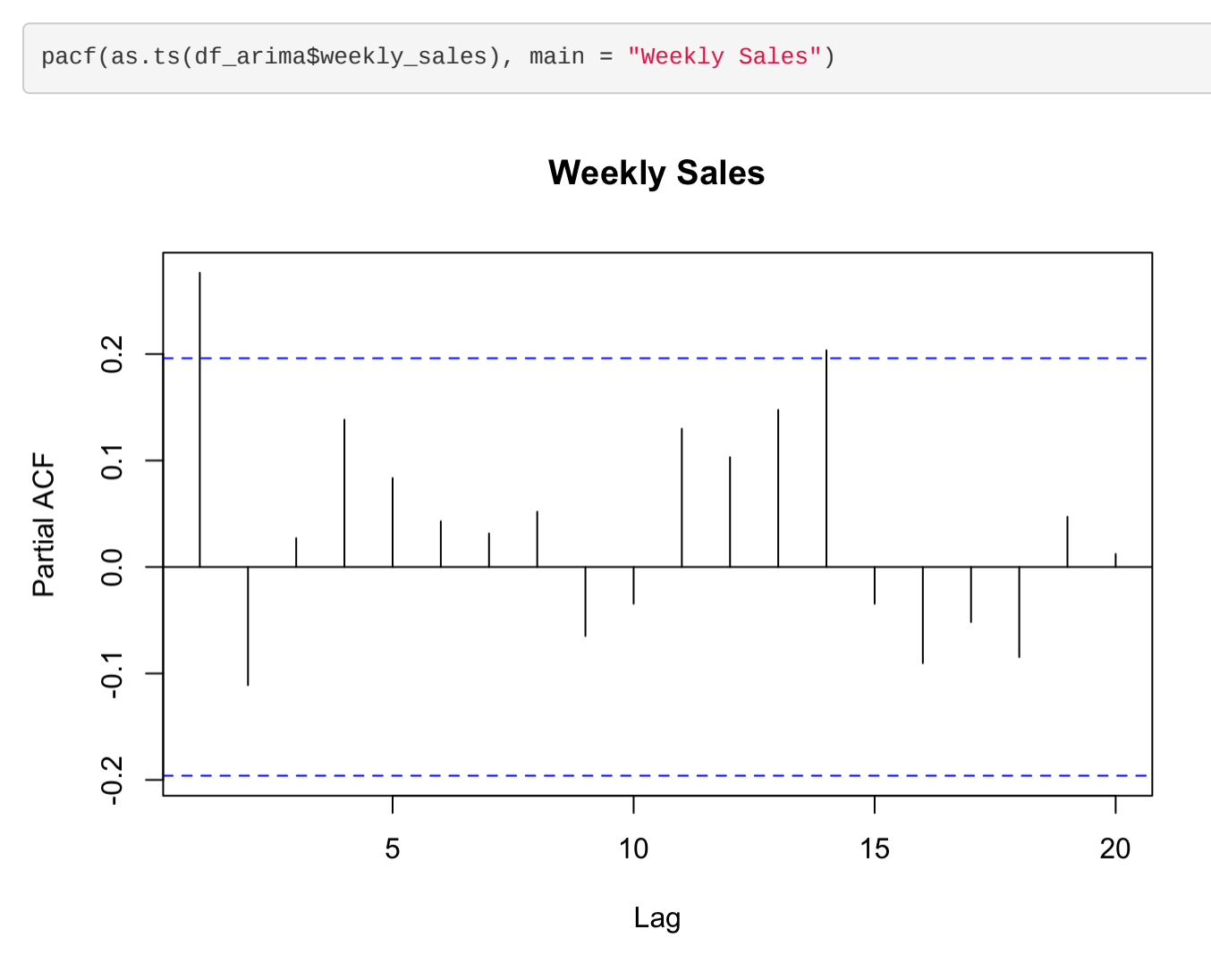
```
## Warning in adf.test(df_arima$weekly_sales): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  df_arima$weekly_sales
## Dickey-Fuller = -4.7482, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
acf(as.ts(df_arima$weekly_sales), main = "Weekly Sales")
```




```
pacf(as.ts(df_arima$weekly_sales), main = "Weekly Sales")
```



Based on the p-val of the ADF test, we reject the null hypothesis (data is non-stationary) when we choose the alpha level of 0.05. We are 95% confidence about this result that the data is stationary.
However,the ACF and PACF above say otherwise, there is a correlation at lag 1.
There are 3 parameters for the Arima model as Arima(p,d,q)
With the result of PACF, it can be seen the p,q are 1
With the result of the ACF, it can be seen the d is 1
So, lets test around the parameters values around (1,3)

Fitting the Arima model

Since R has an amazing function to fit the best Arima parameters called auto.arima(), let’s use it instead

```
arima_mod <- auto.arima(as.ts(df_arima$weekly_sales))
```

The best model is Arima(2,1,1)

Splitting training and testing datasets

In order to test for the validity of the model, we first have to split the dataset in to training and testing datasets.

```
train_index <- 0.7 # set the how much data to put into train data set
n_total <- nrow(df_arima) # number of rows in the dataframe
df_arima_train <- df_arima[1:(train_index*n_total),]  # subsetting train dataset
df_arima_test <- df_arima[(train_index*n_total+1):n_total,] # subsetting test dataset
arima_predict <- numeric(n_total-nrow(df_arima_train)) #predetermine the dataset for the predictions from arima m
odel
```

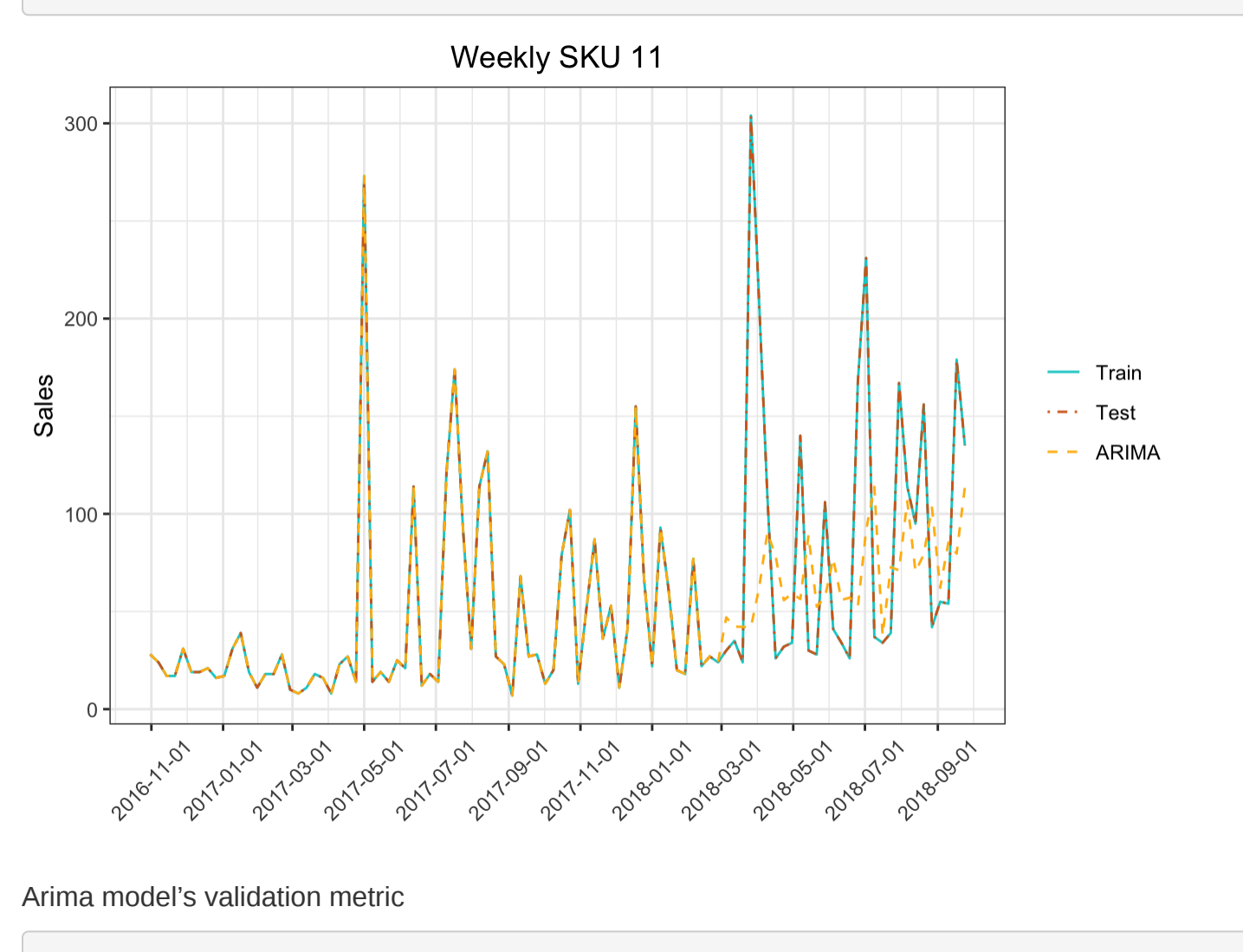
Fitting the ARIMA model recursively

```
for (i in 1:(n_total-(train_index*n_total))) {
  df_arima_train1 <- df_arima[1:(train_index*n_total-1+i),]
  arima_model <- auto.arima(as.ts(df_arima_train1$weekly_sales))
  pred <- forecast(arima_model,1)
  arima_predict[i]<- pred$mean
}
```

Visualize the result

```
df_arima_pred <- tibble("Train" = c(df_arima_train$weekly_sales, df_arima_test$weekly_sales),
                        "Test" = c(df_arima_train$weekly_sales, df_arima_test$weekly_sales),
                        "ARIMA" = c(df_arima_train$weekly_sales, arima_predict),
                        time = df_arima$week) # create a tibble
final1 <- melt(data = df_arima_pred, id.vars = "time") # melt the data into molten data frame

final1 %>%
  ggplot(aes(x = time, y = value, col = variable, linetype = variable)) +
  geom_line() +
  xlab("") + ylab("Sales") +
  scale_color_manual(values=c("cyan3", "chocolate", "darkgoldenrod1")) +
  scale_linetype_manual(values=c(1, 4, 2)) +
  scale_x_date(date_labels = "%Y-%m-%d", date_breaks = "2 month") +
  theme_bw() + theme(legend.title = element_blank(),
                     axis.text.x  = element_text(angle=45, vjust=0.5)) +
  ggtitle("Weekly SKU 11") +
  theme(plot.title = element_text(hjust = 0.5))
```



Arima model’s validation metric

```
metric <- validation_stats_script(df_arima_test$weekly_sales, arima_predict)
metric.name <- c("r2", "mse", "mape", "mae", "rmse")
metric.val <- c(metric$r2, metric$mse, metric$mape, metric$mae, metric$rmse)
metric.final <- data.frame(metric.name, metric.val)
metric.final
```

metric.name <chr>	metric.val <dbl>
r2	0.004
mse	6100.000
mape	73.000
mae	56.000
rmse	78.000

5 rows

As expected, the Arima model does not perform well with weekly data that the r2 only shows 0.004 mean the model only explains 0.4% of the data. With MAPE of 73, the forecast is off by 73% from true value. Similar with MAPE, MAE shows 56 meaning the forecast is off by 56 unit from true value.

Part 2: Weekly Total Sales

Random Forest Model

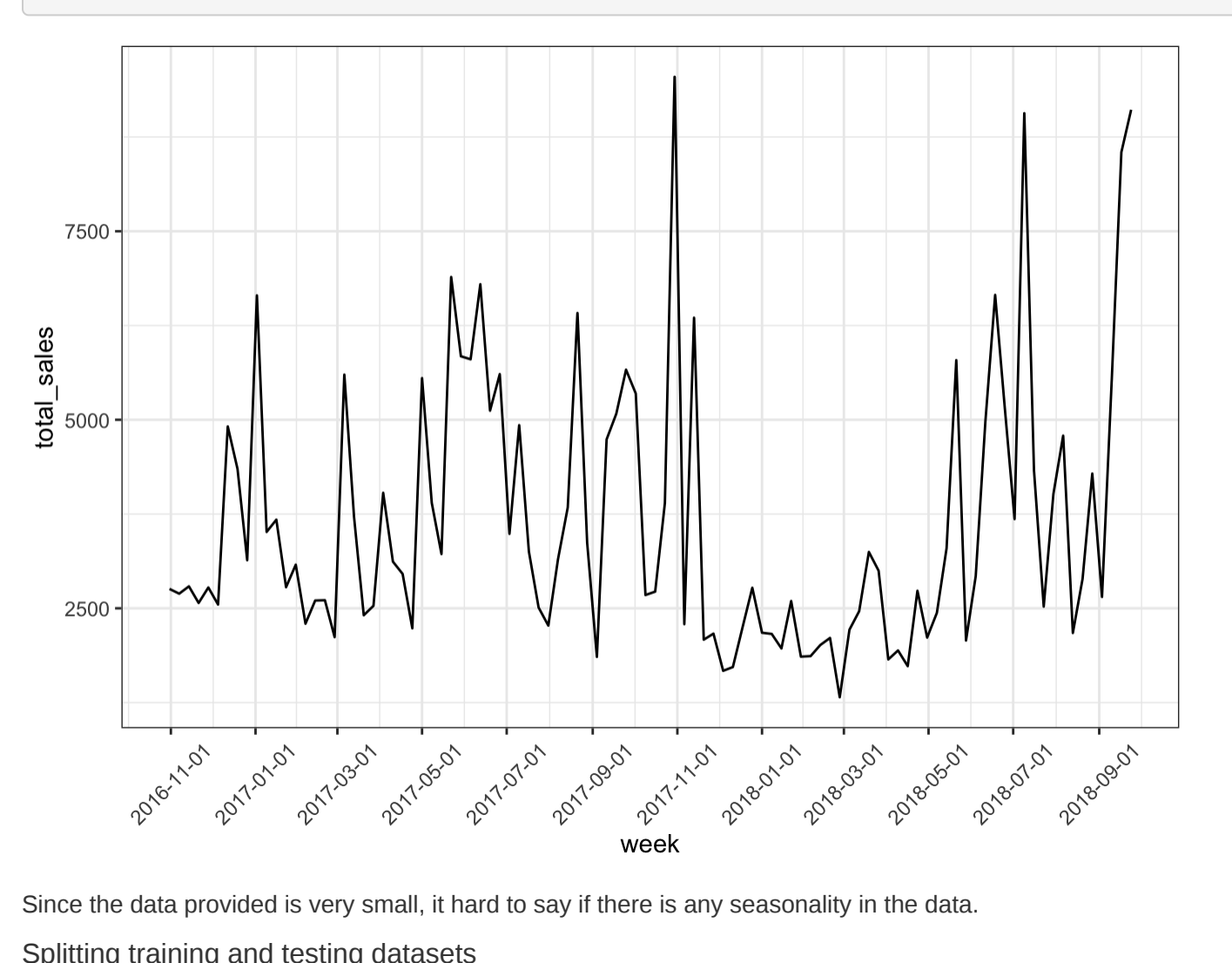
Random Forest model is an ensemble learning method (use multiple learning algorithm to obtain better predictive performance) of classification, regression and other tasks that operates by constructing a multitude of decision trees at training time.

Data processing to find the Weekly Total Sales

```
df_total_sales <- df %>%
  group_by(week) %>%
  summarise(total_sales = sum(weekly_sales))
```

Initial visualization of the Weekly Total Sales by Week

```
df_total_sales %>%
  ggplot(aes(x= week, y = total_sales)) + geom_line() +
  scale_x_date(date_labels = "%Y-%m-%d", date_breaks = "2 month") +
  theme_bw() + theme(legend.title = element_blank(),
                     axis.text.x  = element_text(angle=45, vjust=0.5))
```



Since the data provided is very small, it hard to say if there is any seasonality in the data.

Splitting training and testing datasets

```
train_index <- 0.7
n_total <- nrow(df_total_sales)
train <- df_total_sales[1:(train_index*n_total),]
test <- df_total_sales[(train_index*n_total+1):n_total,]
```

Fitting Random Forest model

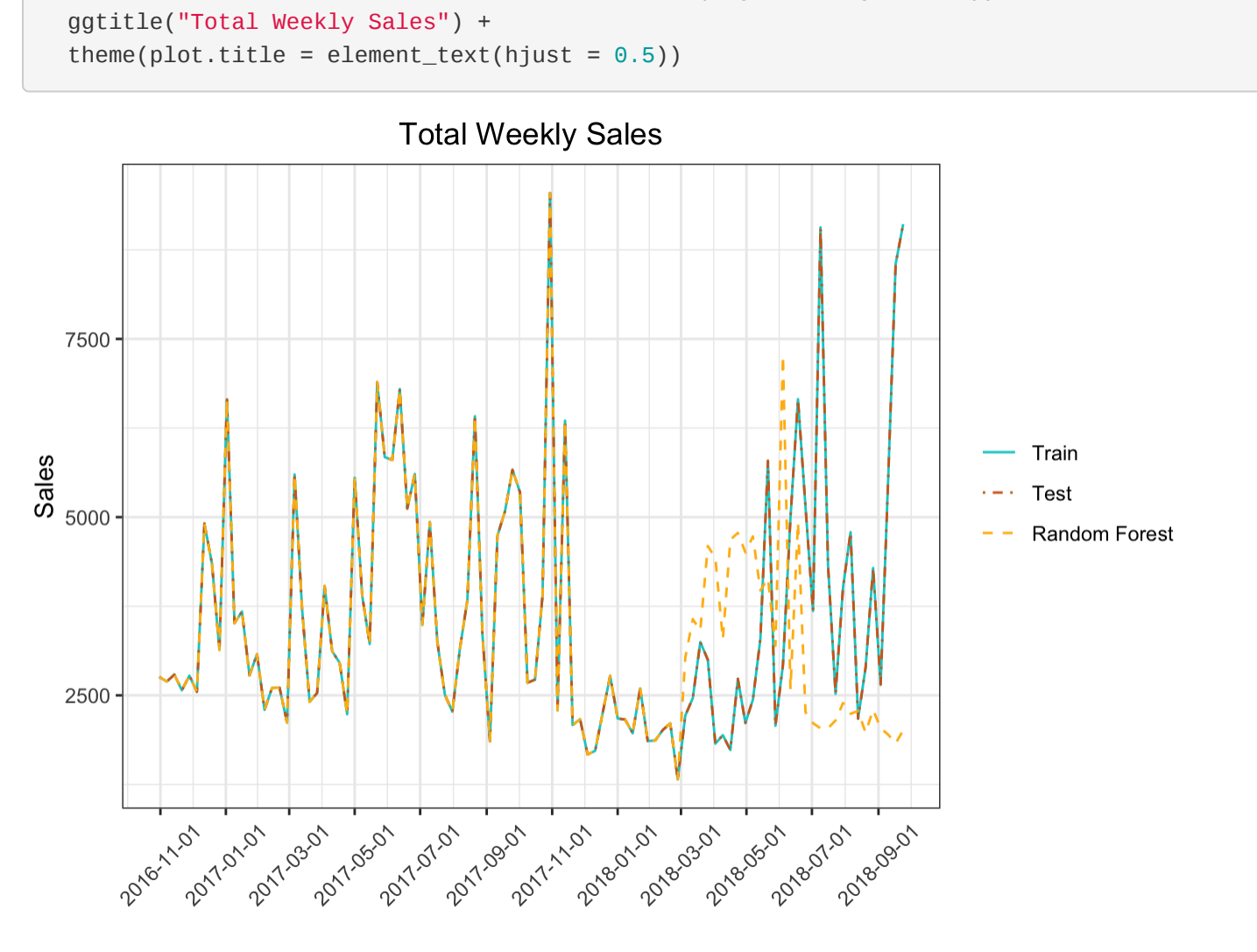
```
rd <- randomForest(data = train, total_sales~week) # fit the model
rd_predict <- tail(rd$predicted, nrow(test)) # forecast the data
```

Visualize the forecasting reports with Random Forest model

```
df_rd_pred <- tibble("Train" = c(train$total_sales, test$total_sales),
                     "Test" = c(train$total_sales, test$total_sales),
                     "Random Forest" = c(train$total_sales,rd_predict),
                     time = df_total_sales$week) # create a tibble

final2 <- melt(data = df_rd_pred, id.vars = "time") # melt the tibble into molten data frame

final2 %>%
  ggplot(aes(x = time, y = value, col = variable, linetype = variable)) +
  geom_line() +
  xlab("") + ylab("Sales") +
  scale_color_manual(values=c("cyan3", "chocolate", "darkgoldenrod1")) +
  scale_linetype_manual(values=c(1, 4, 2)) +
  scale_x_date(date_labels = "%Y-%m-%d", date_breaks = "2 month") +
  theme_bw() + theme(legend.title = element_blank(),
                     axis.text.x  = element_text(angle=45, vjust=0.5)) +
  ggtitle("Total Weekly Sales") +
  theme(plot.title = element_text(hjust = 0.5))
```



Random Forest model’s validation metric

```
metric2 <- validation_stats_script(test$total_sales, rd_predict)
metric.name2 <- c("r2", "mse", "mape", "mae", "rmse")
metric.val2 <- c(metric2$r2, metric2$mse, metric2$mape, metric2$mae, metric2$rmse)
metric.final2 <- data.frame(metric.name2, metric.val2)
metric.final2
```

metric.name2 <chr>	metric.val2 <dbl>
r2	0.14
mse	8600000.00
mape	60.00
mae	2300.00
rmse	2900.00

5 rows

r2 only shows 0.15 mean the model only explains 15% of the data. With MAPE of 60, the forecast is off by 60% from true value. Similar with MAPE, MAE shows 2300 meaning the forecast is off by 2300 unit from true value.

TBATS

Another traditional model to look at is the TBATS model. TBATS stands for:

T: Trigonometric seasonality
B: Box-Cox transformation
A: ARIMA errors
T: Trend
S: Seasonal components

Although TBATS performs the best with seasonal data, and is hard to say if there’s any seasonality in the data, TBATS is still a great model to learn and use to forecast time series data.

Fitting the TBATS model

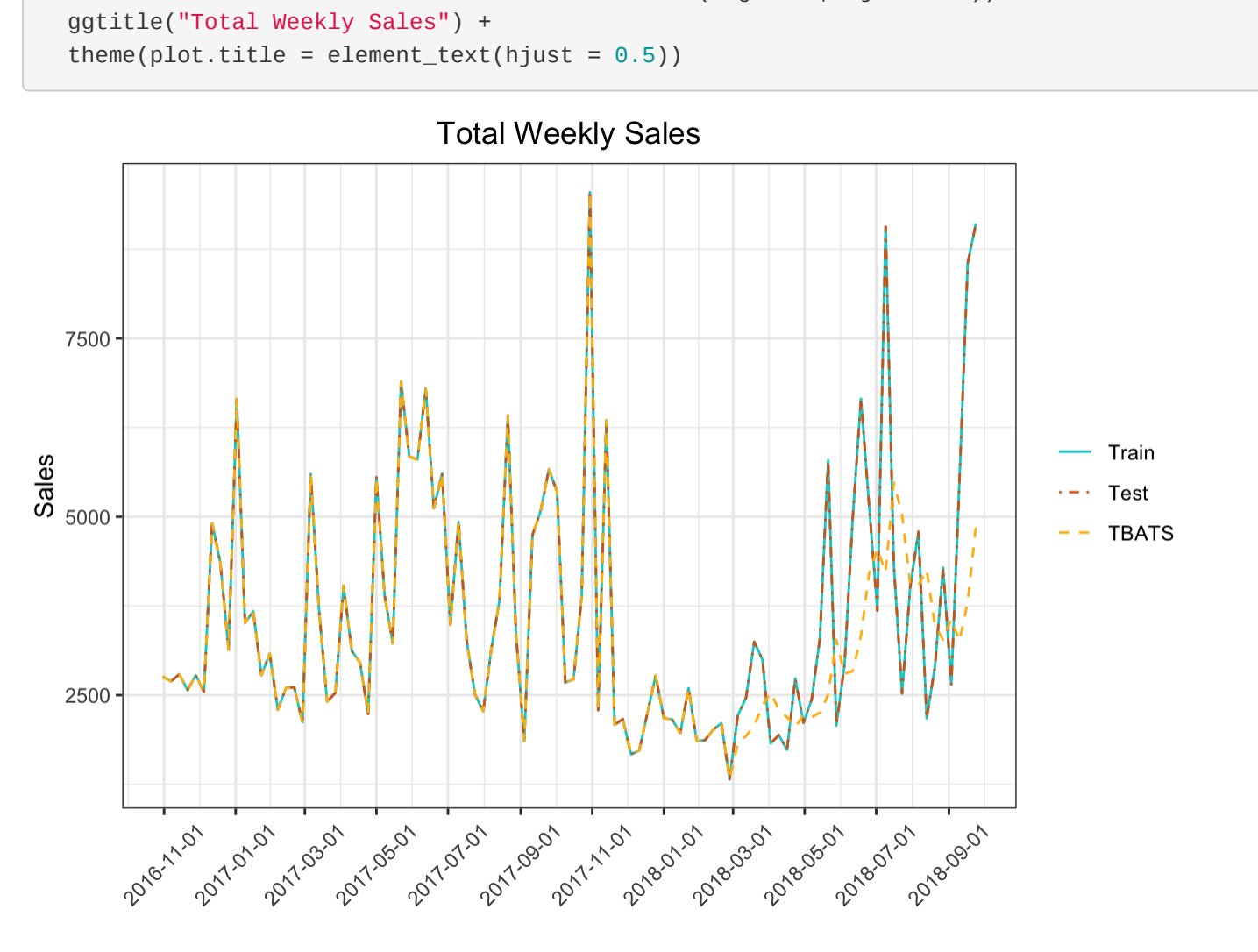
```
tbats_predict <- numeric(n_total-nrow(train))
for (i in 1:(n_total-(train_index*n_total))) {
  df_tbats_train1 <- df_total_sales[1:(train_index*n_total-1+i),]
  tbats_model <- tbats(as.ts(df_tbats_train1$total_sales))
  pred <- forecast(tbats_model, 1)
  tbats_predict[i] <- pred$mean
}
```

Visualize the forecasting reports with TBATS model

```
df_tbats_pred <- tibble("Train" = c(train$total_sales, test$total_sales),
                        "Test" = c(train$total_sales, test$total_sales),
                        "TBATS" = c(train$total_sales, tbats_predict),
                        time = df_total_sales$week)

final3 <- melt(data = df_tbats_pred, id.vars = "time")

final3 %>%
  ggplot(aes(x = time, y = value, col = variable, linetype = variable)) +
  geom_line() +
  xlab("") + ylab("Sales") +
  scale_color_manual(values=c("cyan3", "chocolate", "darkgoldenrod1")) +
  scale_linetype_manual(values=c(1, 4, 2)) +
  scale_x_date(date_labels = "%Y-%m-%d", date_breaks = "2 month") +
  theme_bw() + theme(legend.title = element_blank(),
                     axis.text.x  = element_text(angle=45, vjust=0.5)) +
  ggtitle("Total Weekly Sales") +
  theme(plot.title = element_text(hjust = 0.5))
```



TBATS model’s validation metric

```
metric3 <- validation_stats_script(test$total_sales, tbats_predict)
metric.name3 <- c("r2", "mse", "mape", "mae", "rmse")
metric.val3 <- c(metric3$r2, metric3$mse, metric3$mape, metric3$mae, metric3$rmse)
metric.final3 <- data.frame(metric.name3, metric.val3)

metric.final3
```

metric.name3 <chr>	metric.val3 <dbl>
r2	0.22
mse	3900000.00
mape	34.00
mae	1400.00
rmse	2000.00

5 rows

The r2 only shows 0.22 mean the model only explains 22% of the data. With MAPE of 34, the forecast is off by 34% from true value. Similar with MAPE, MAE shows 1400 meaning the forecast is off by 1400 unit from true value.

Prophet

Lastly, the most modern to look at is the Prophet, designed by the Facebook team to help both experienced and non-experienced statistician to forecast the time series data.

Finding the lambda value for Prophet Model

```
train1 <- train
lam <- BoxCox.lambda(train1$total_sales, method="loglik") #lambda
```

Transform the data with the lambda found and change the column names to fit the model’s requirment

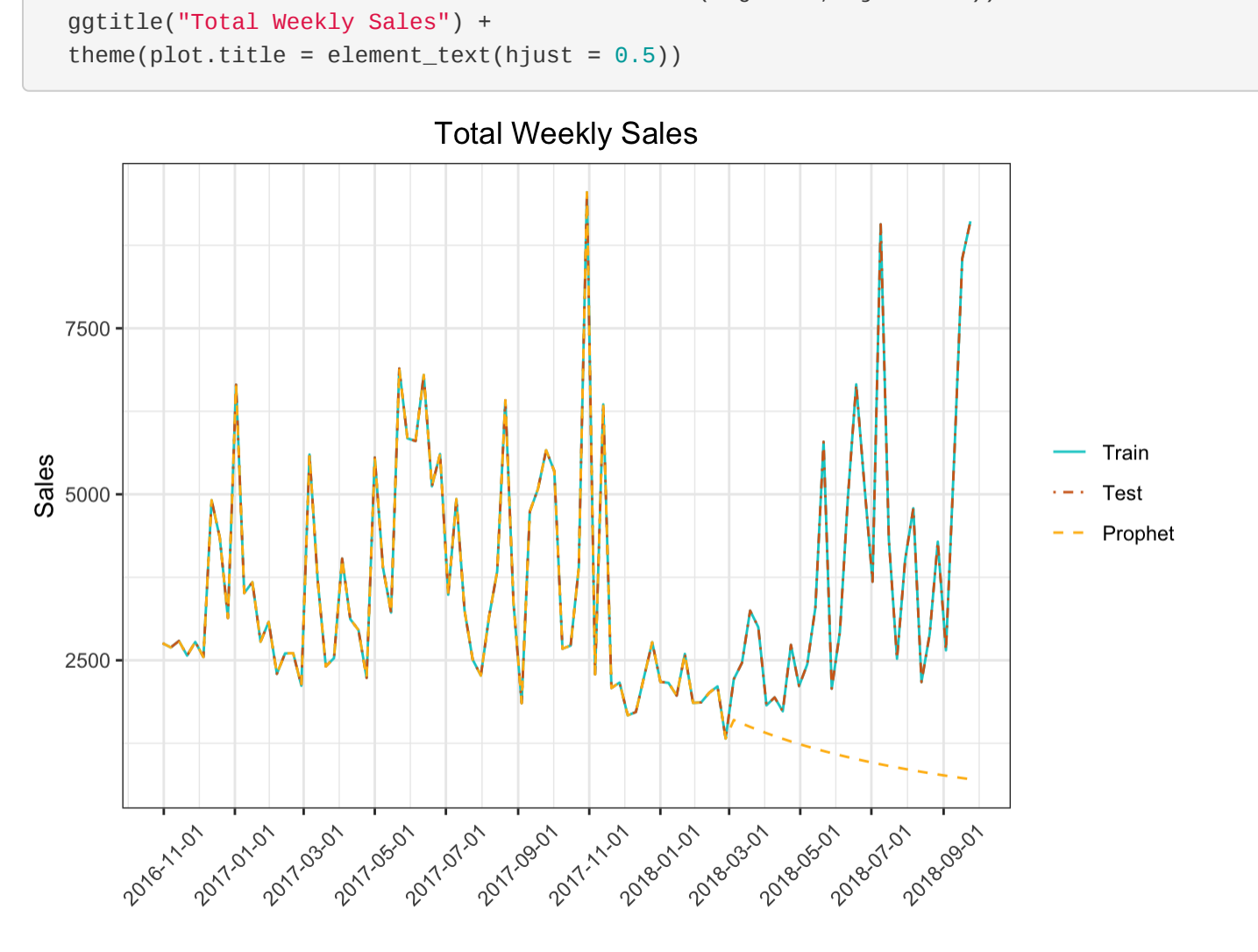
```
train1$y <- BoxCox(train1$total_sales, lam) #transform the data
keep <- c("week", "y")
train1 <- train1[keep]
colnames(train1) <- c("ds", "y") # change the column name because the model required to
```

Fitting the prophet model and visualizing the performence

```
prophet_model <- prophet(train1) # fit the model
future <- make_future_dataframe(prophet_model, periods = 30, freq = "week") # make future datas for forecasting
forecast <- predict(prophet_model, future) # forecast data
prophet_predict <- tail(InvBoxCox(forecast$yhat, lam),30) # inverse transform the prediction from the prophet mod
el
```

Visualizing the model against train and test sets

```
df_prophet_pred <- tibble("Train" = c(train$total_sales, test$total_sales),
                          "Test" = c(train$total_sales, test$total_sales),
                          "Prophet" = c(train$total_sales, tail(prophet_predict,30)),
                          time = df_total_sales$week)
final4 <- melt(data = df_prophet_pred, id.vars = "time")
final4 %>%
  ggplot(aes(x = time, y = value, colour = variable, linetype = variable)) +
  geom_line() +
  xlab("") + ylab("Sales") +
  scale_color_manual(values=c("cyan3", "chocolate", "darkgoldenrod1")) +
  scale_linetype_manual(values=c(1, 4, 2)) +
  scale_x_date(date_labels = "%Y-%m-%d", date_breaks = "2 month") +
  theme_bw() + theme(legend.title = element_blank(),
                     axis.text.x  = element_text(angle=45, vjust=0.5)) +
  ggtitle("Total Weekly Sales") +
  theme(plot.title = element_text(hjust = 0.5))
```



Prophet model’s validation metric

```
metric4 <- validation_stats_script(test$total_sales, prophet_predict)
metric.name4 <- c("r2", "mse", "mape", "mae", "rmse")
metric.val4 <- c(metric4$r2, metric4$mse, metric4$mape, metric4$mae, metric4$rmse)
metric.final4 <- data.frame(metric.name4, metric.val4)

metric.final4
```

metric.name4 <chr>	metric.val4 <dbl>
r2	0.29
mse	13000000.00
mape	64.00
mae	2900.00
rmse	3600.00

5 rows

The r2 only shows 0.29 mean the model only explains 29% of the data. With MAPE of 64, the forecast is off by 64% from true value. Similar with MAPE, MAE shows 2900 meaning the forecast is off by 2900 unit from true value. Here the model’s mse and rmse show extremely large values of 1.3x10^6 and 3600 respectively. I suspect this is due to the model not being able to detect weekly seasonality causing the yhat (y_predicted) to be off. This is shown through the plot above.

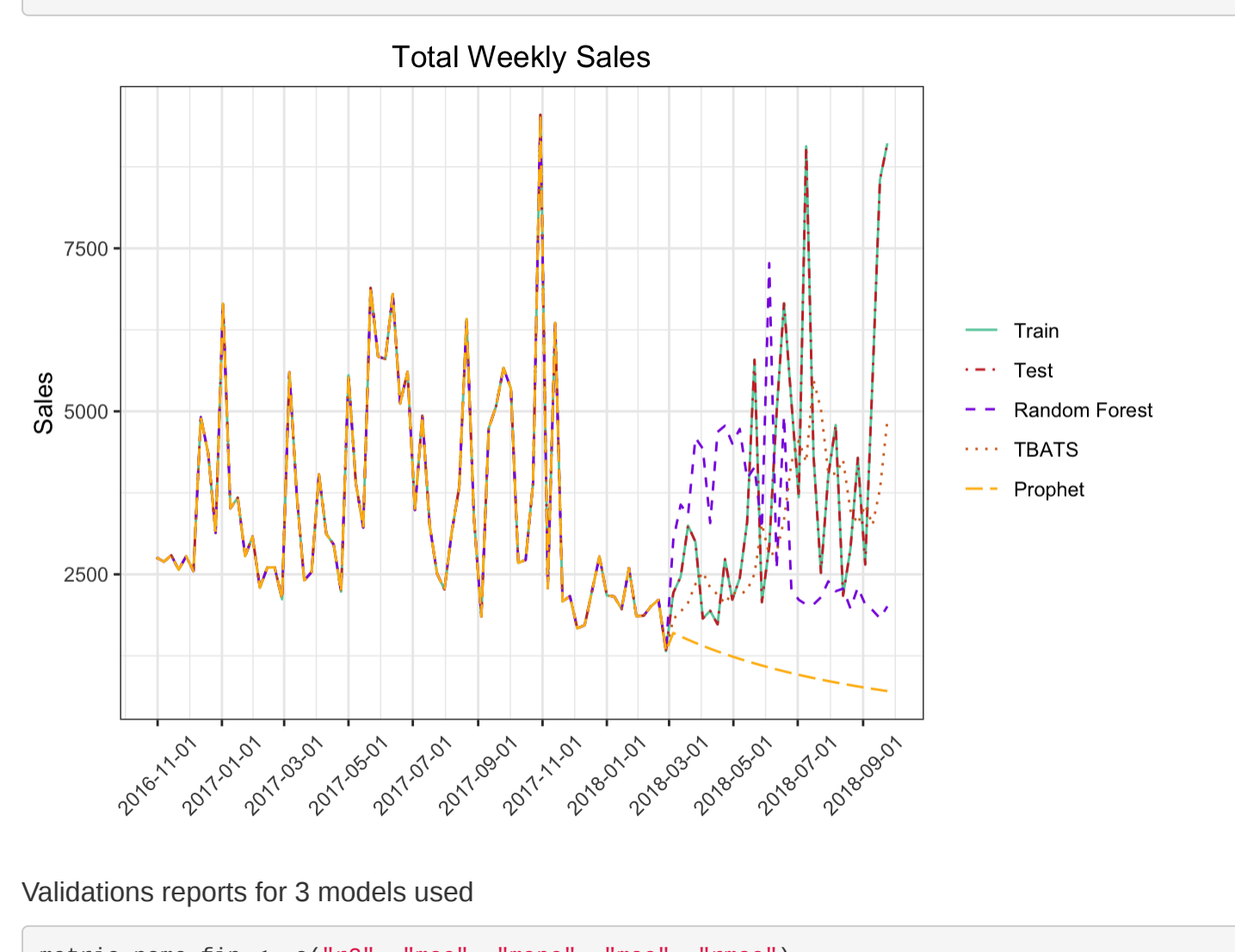
Summary

Visualizing all 3 models together

```
df_final_pred <- tibble("Train" = c(train$total_sales, test$total_sales),
                        "Test" = c(train$total_sales, test$total_sales),
                        "Random Forest" = c(train$total_sales, rd_predict),
                        "TBATS" = c(train$total_sales, tbats_predict),
                        "Prophet" = c(train$total_sales, prophet_predict),
                        time = df_total_sales$week)
df_final_plot <- melt(data= df_final_pred, id.vars = "time")
```

```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

```
df_final_plot %>%
ggplot(aes(x= time, y = value, colour = variable, linetype= variable)) +
  geom_line() +
  xlab("") + ylab("Sales") +
  scale_color_manual(values=c("aquamarine3","brown3","blueviolet", "chocolate", "darkgoldenrod1")) +
  scale_linetype_manual(values=c(1,4, 2,3,5)) +
  scale_x_date(date_labels = "%Y-%m-%d", date_breaks = "2 month") +
  theme_bw() + theme(legend.title = element_blank(),
                     axis.text.x  = element_text(angle=45, vjust=0.5)) +
  ggtitle("Total Weekly Sales") +
  theme(plot.title = element_text(hjust = 0.5))
```



Validations reports for 3 models used

```
metric.name_fin <- c("r2", "mse", "mape", "mae", "rmse")
rd <- c(metric2$r2, metric2$mse, metric2$mape, metric2$mae, metric2$rmse)
tbats <- c(metric3$r2, metric3$mse, metric3$mape, metric3$mae, metric3$rmse)
prophet <- c(metric4$r2, metric4$mse, metric4$mape, metric4$mae, metric4$rmse)
metric.final_fin <- data.frame(metric.name_fin,rd, tbats, prophet)
colnames(metric.final_fin) <- c("Model Metric","Random Forest", "TBATS", "Prophet")
metric.final_fin
```

Model Metric <chr>	Random Forest <dbl>	TBATS <dbl>	Prophet <dbl>
r2	0.14	0.22	0.29
mse	8600000.00	3900000.00	13000000.00
mape	60.00	34.00	64.00
mae	2300.00	1400.00	2900.00
rmse	2900.00	2000.00	3600.00

5 rows

Comparing these 3 models that I used, TBATS is the best model to forecast the Weekly Total Sales with the metric values the lowest across mse, mape, mae, and rmse. However, with r2 is so low of 0.22, I suspect there’s an underfit of the model. More work is needed to confirm the underfitness of model.