Final Project

Jessica Bao

3/15/2021

```r
# Load libraries and prepare data
library("dplyr")
library("anytime")
library("lubridate")
library("readxl")
library("tidyverse")
library("TSA")
library("fpp")
library("Metrics")
library("corrplot")
library("tidymodels")

train <- read.csv("rossmann-store-sales/train.csv")
store_8 <- train %>% filter(Store==8) %>% arrange(Date)
#store_8

# Convert Date column to type "Date", get the first day of our data
store_8$Date = as.Date(store_8$Date, format = "%Y-%m-%d")

# Convert "03-01" to day of the year
dayOfYear = as.numeric(format(store_8[1,3], "%j"))

#keep the date information for future use
store_8$Date <- anydate(store_8$Date)
store_8$year <- year(store_8$Date)
store_8$month <- month(store_8$Date)
store_8$week <- week(store_8$Date)
store_8$day <- day(store_8$Date)

# double check 365 dates each year without missing
store_8 %>% count(year,sort = TRUE)

#take a look at the data: no missing data
#summary(store_8)

#drop store number
store_8 <- subset(store_8, select=-1)

# adjust the col sequence
store_8 <- subset(store_8,select= c(2:ncol(store_8),1))

#factor state holiday
```

```r
store_8$StateHoliday<-factor(store_8$StateHoliday)
store_8$SchoolHoliday<-factor(store_8$SchoolHoliday)
store_8$DayOfWeek<-factor(store_8$DayOfWeek)
store_8$Open<-factor(store_8$Open)
store_8$Promo<-factor(store_8$Promo)

#convert to time series data
store_8 <- ts(store_8, frequency = 7)
#autoplot(store_8)

#split train test data
train <- window(store_8, start = c(1, 1), end = c(132, 4))
test <-window(store_8, start = c(132, 5))

#autoplot(train)
#autoplot(train[,2])
```

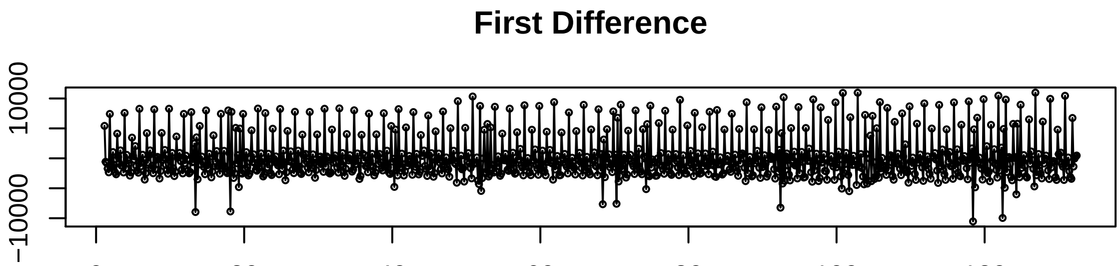
```r
# Create a correlation matrix
train_corr <- cor(train)
round(train_corr, 2)[, 2] # Look at the correlations to Sales only
```

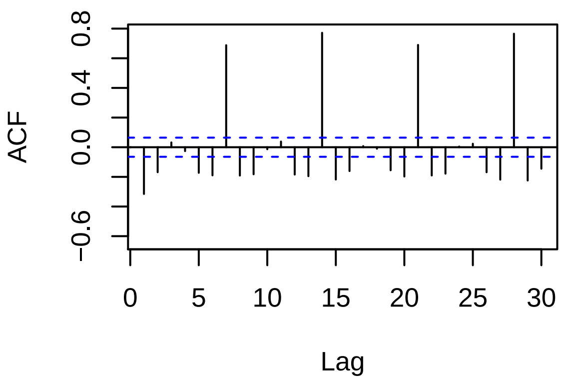
```
##          Date         Sales     Customers          Open         Promo
##          0.15          1.00          0.97          0.77          0.69
##  StateHoliday SchoolHoliday          year         month          week
##         -0.25          0.11          0.13          0.04          0.04
##           day     DayOfWeek
##         -0.03         -0.71
```

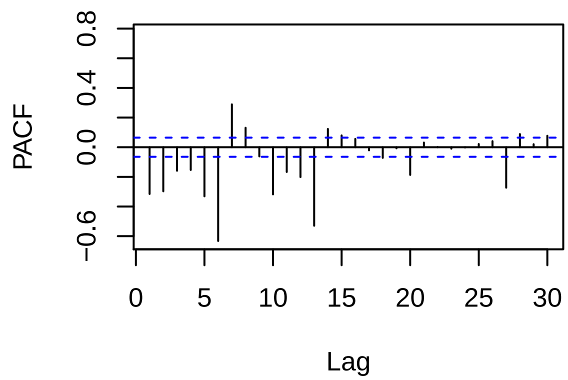
```r
# Sales is weakly correlated with date, StateHoliday, SchoolHoliday, year, Month, Week, and Day.
# Keep: Open, Promo, and DayofWeek.
```

```r
# Create the dependent and independent variables.
sales <- train[, "Sales"]
predictors <- train[, c("Open", "Promo", "DayOfWeek")]
```

```r
# Differencing
sales_diff1 <- diff(sales, differences = 1)
tsdisplay(sales_diff1, main = "First Difference")
```








```
# ADF test
kpss.test(sales) #p = .01
```

```
## Warning in kpss.test(sales): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  sales
## KPSS Level = 2.8265, Truncation lag parameter = 6, p-value = 0.01
```

```
kpss.test(sales_diff1) #p = .1
```

```
## Warning in kpss.test(sales_diff1): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  sales_diff1
## KPSS Level = 0.013299, Truncation lag parameter = 6, p-value = 0.1
```

The null hypothesis of the KPSS test is that the data is stationary. Because the p-value of the KPSS test of the first difference is less than alpha of .05, we reject the null, meaning the data is not stationary.

The p-value of the KPSS test of the second difference is greater than alpha of .05, so we fail to reject the null, meaning the data is stationary.

```
open <- diff(predictors[, "Open"], differences = 1)
promo <- diff(predictors[, "Promo"], differences = 1)
dayofweek <- diff(predictors[, "DayOfWeek"], differences = 1)

kpss.test(open) #p = .1
```

```
## Warning in kpss.test(open): p-value greater than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  open
## KPSS Level = 0.04784, Truncation lag parameter = 6, p-value = 0.1
```

```
kpss.test(promo) #p = .1
```

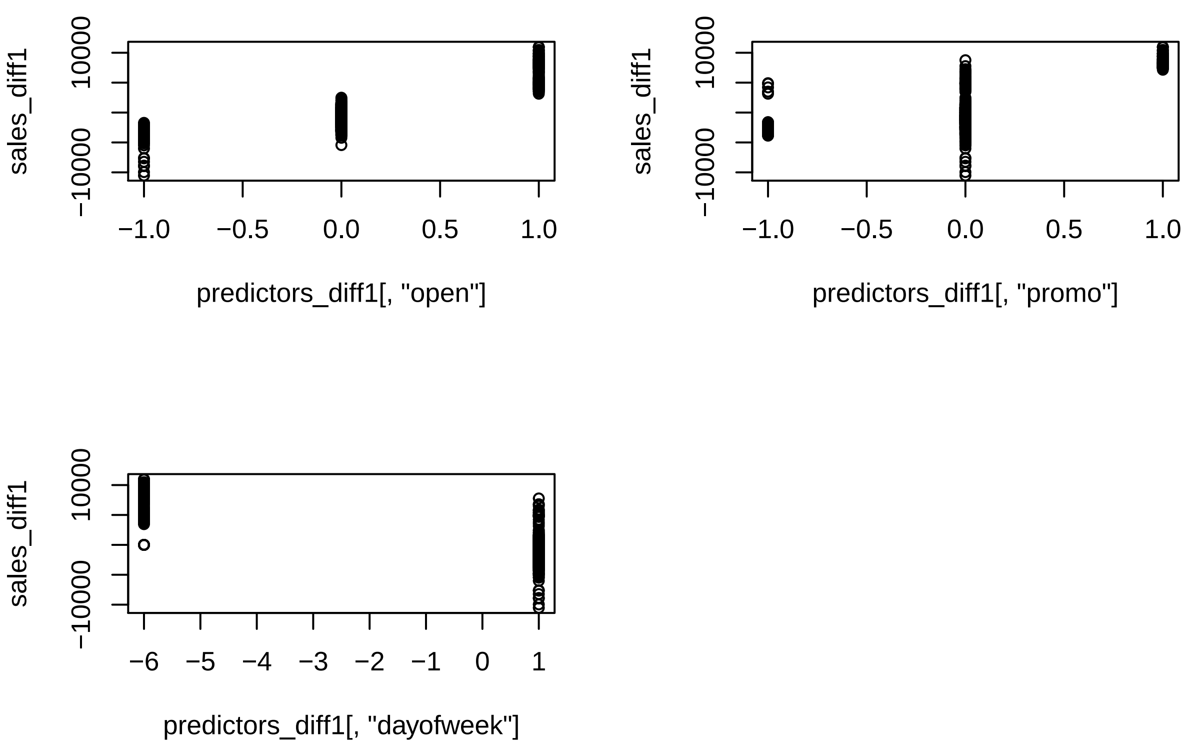
```
## Warning in kpss.test(promo): p-value greater than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  promo
## KPSS Level = 0.0045136, Truncation lag parameter = 6, p-value = 0.1
```

```
kpss.test(dayofweek) #p = .08411
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  dayofweek
## KPSS Level = 0.38387, Truncation lag parameter = 6, p-value = 0.08411
```

```
predictors_diff1 = cbind(open, promo, dayofweek)
```

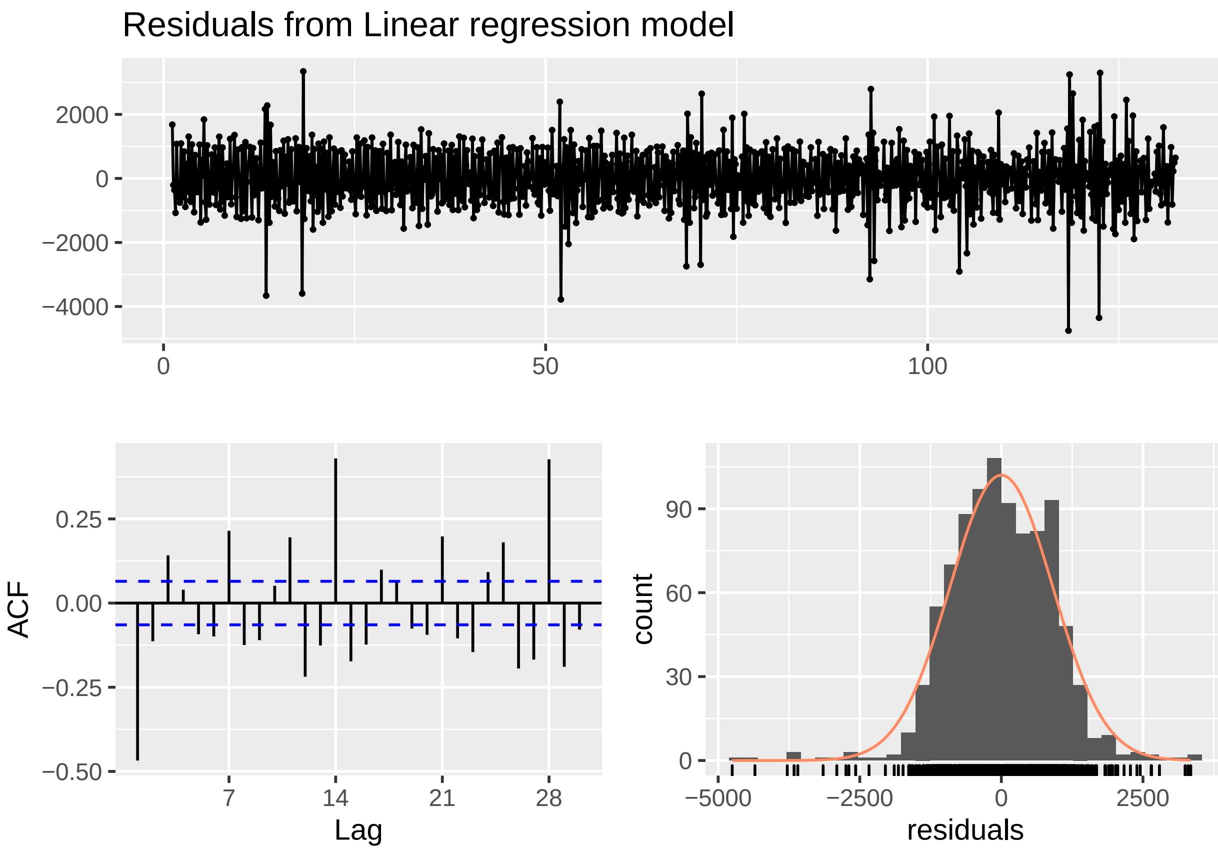
```
par(mfrow = c(2, 2))
plot(sales_diff1 ~ predictors_diff1[, "open"])
plot(sales_diff1 ~ predictors_diff1[, "promo"])
plot(sales_diff1 ~ predictors_diff1[, "dayofweek"])
```



```r
# Fit a TSLM model.
tslm_fit <- tslm(sales_diff1 ~ predictors_diff1, lambda = "auto")

# Look at the summary of the model.
summary(tslm_fit)
```

```
##
## Call:
## tslm(formula = sales_diff1 ~ predictors_diff1, lambda = "auto")
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4752.6  -597.0   -21.0   651.7  3343.7
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -29.50      30.06  -0.981    0.327
## predictors_diff1open        2608.70      73.83  35.332   <2e-16 ***
## predictors_diff1promo       1978.67      93.40  21.185   <2e-16 ***
## predictors_diff1dayofweek  -234.91      19.44 -12.082   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 911.9 on 916 degrees of freedom
## Multiple R-squared:  0.8688, Adjusted R-squared:  0.8684
## F-statistic:  2022 on 3 and 916 DF,  p-value: < 2.2e-16
```

```r
# Look at the residuals.
checkresiduals(tslm_fit)
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 14
##
## data:  Residuals from Linear regression model
## LM test = 552.53, df = 14, p-value < 2.2e-16
```

```
# Look at AIC
#glance(tslm_fit)
AIC(tslm_fit, k = 3) # default k is 2 but because we want AICc, adding extra penalty parameter
```

```
## [1] 15162.39
```

```
# TSLM Box Ljung
Box.test(tslm_fit$residuals, type = c("Ljung-Box"))
```

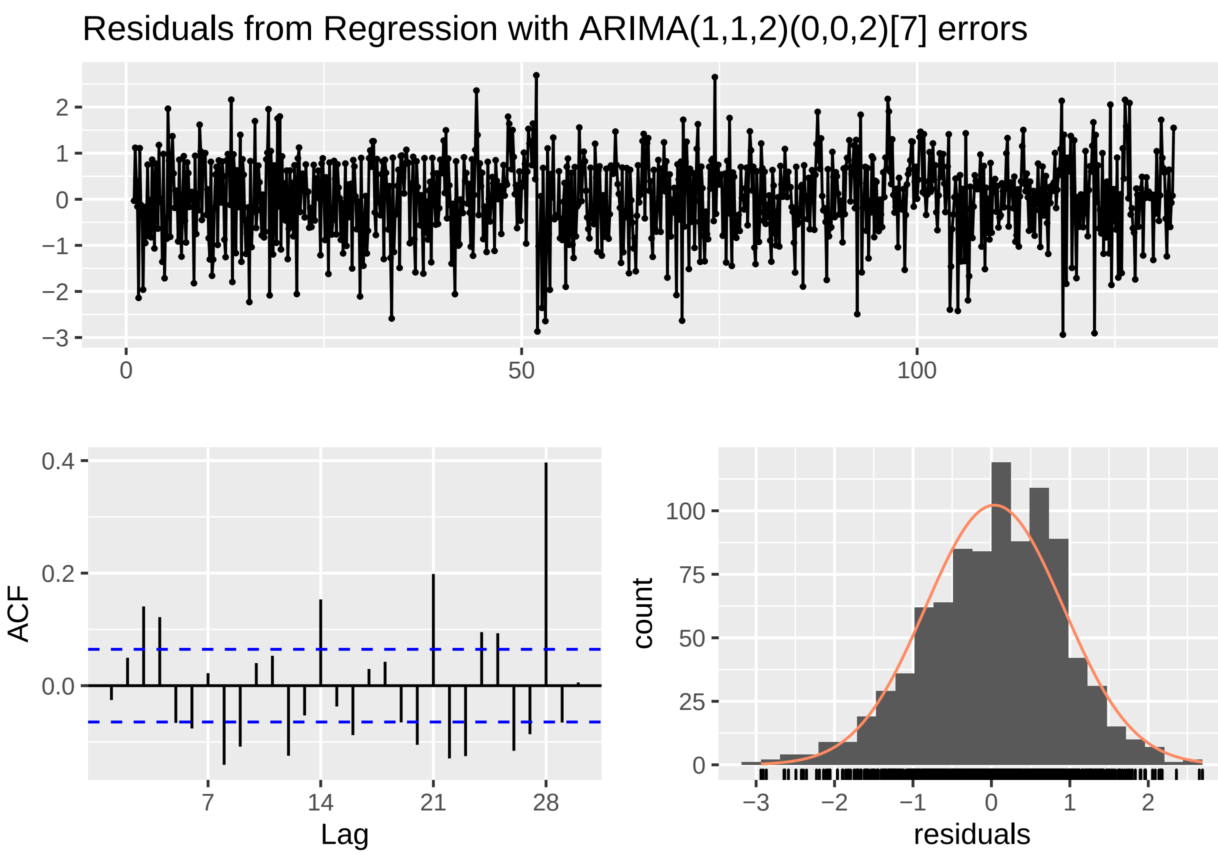
```
##
##  Box-Ljung test
##
## data:  tslm_fit$residuals
## X-squared = 201.76, df = 1, p-value < 2.2e-16
```

```
# Fit regression with ARIMA errors
linear_arima_fit <- auto.arima(sales, xreg = predictors, lambda = "auto", seasonal = TRUE, allowdrift =

# Look at the summary of the model.
summary(linear_arima_fit)
```

```
## Series: sales
## Regression with ARIMA(1,1,2)(0,0,2)[7] errors
```

```
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##           ar1      ma1      ma2    sma1    sma2     Open   Promo  DayOfWeek
##       -0.9029  -0.0455  -0.9161  0.3345  0.4026  26.9716  2.3035    -0.3542
## s.e.   0.0512   0.0421   0.0408  0.0393  0.0293   0.1489  0.0647     0.0297
##
## sigma^2 estimated as 0.7824:  log likelihood=-1191.09
## AIC=2400.19   AICc=2400.38   BIC=2443.6
##
## Training set error measures:
##                    ME     RMSE     MAE MPE MAPE      MASE      ACF1
## Training set 58.94127 740.8819 526.195 NaN  Inf 0.2935999 0.1634032
```

```
# Look at the residuals.
checkresiduals(linear_arima_fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,1,2)(0,0,2)[7] errors
## Q* = 117.55, df = 6, p-value < 2.2e-16
##
## Model df: 8.   Total lags used: 14
```

```
# Check EACF of sales
eacf(sales)
```

```
## AR/MA
```

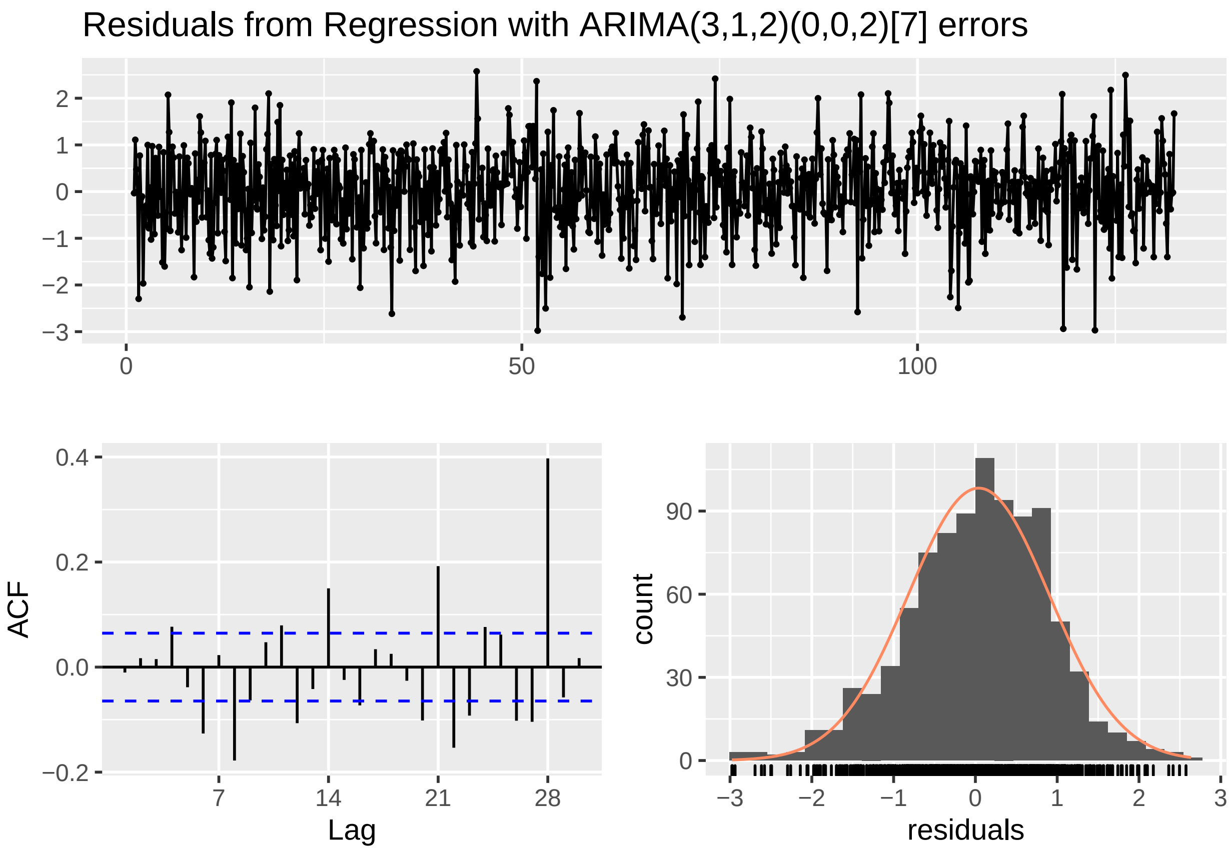
```
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x o x o x x x  x  x  x
## 1 x x o o x o x o x o o  x  o  x
## 2 x x x o o x x x x x o  o  o  x
## 3 x x o o o o x x o o o  o  o  x
## 4 x x o o o x x x o o o  o  o  x
## 5 x x o x x x o x x x o  x  o  x
## 6 x x x x o x o x o o x  x  o  x
## 7 x x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

Try the following:

```
eacf1 <- Arima(sales, xreg = predictors, order = c(3, 1, 2), seasonal=list(order=c(0,0,2), period = 7),
summary(eacf1)
```

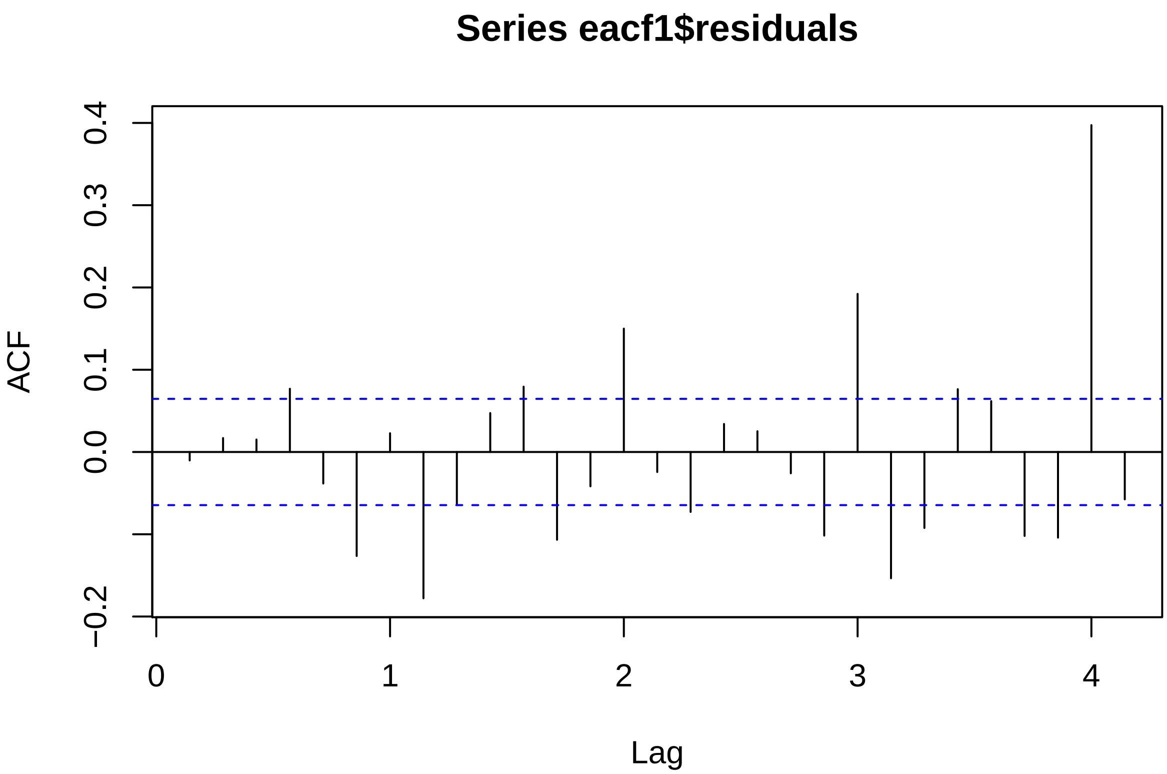
```
## Series: sales
## Regression with ARIMA(3,1,2)(0,0,2)[7] errors
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##          ar1     ar2     ar3      ma1     ma2    sma1    sma2     Open   Promo
##       0.2607  0.0412  0.1844  -1.2677  0.2771  0.3095  0.3982  26.9737  2.2855
## s.e.  0.0949  0.0352  0.0345   0.0939  0.0924  0.0396  0.0288   0.1341  0.0642
##       DayOfWeek
##         -0.3769
## s.e.     0.0255
##
## sigma^2 estimated as 0.7582:  log likelihood=-1175.57
## AIC=2373.14   AICc=2373.43   BIC=2426.21
##
## Training set error measures:
##                    ME     RMSE      MAE MPE MAPE      MASE      ACF1
## Training set 60.93221 719.4383 511.3967 NaN  Inf 0.2853429 0.1843593
```

```
checkresiduals(eacf1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,1,2)(0,0,2)[7] errors
## Q* = 97.285, df = 4, p-value < 2.2e-16
##
## Model df: 10.   Total lags used: 14
```

```
acf(eacf1$residuals)
```




```
# Split the test data into dependent and independent variables.
sales_test <- test[, "Sales"]
predictors_test <- test[, c("Open", "Promo", "DayOfWeek")]

# For TSLM because we're taking the differences, add the last train data
last_train <- train[921, 4]
last_predictors <- tail(predictors,1)

# Append to the test data
sales_test_for_tslm <- ts(c(last_train, sales_test), start = c(132, 4), frequency = frequency(sales_tes
predictors_test_for_tslm <- ts(rbind(last_predictors, predictors_test), start = c(132, 4), frequency = 

# Take differences
sales_test_diff1 <- diff(sales_test_for_tslm, differences = 1)
open_test <- diff(predictors_test_for_tslm[, "Open"], differences = 1)
promo_test <- diff(predictors_test_for_tslm[, "Promo"], differences = 1)
dayofweek_test <- diff(predictors_test_for_tslm[, "DayOfWeek"], differences = 1)

predictors_test_diff1 = cbind(open_test, promo_test, dayofweek_test)
colnames(predictors_test_diff1) <- c("open", "promo", "dayofweek")
```

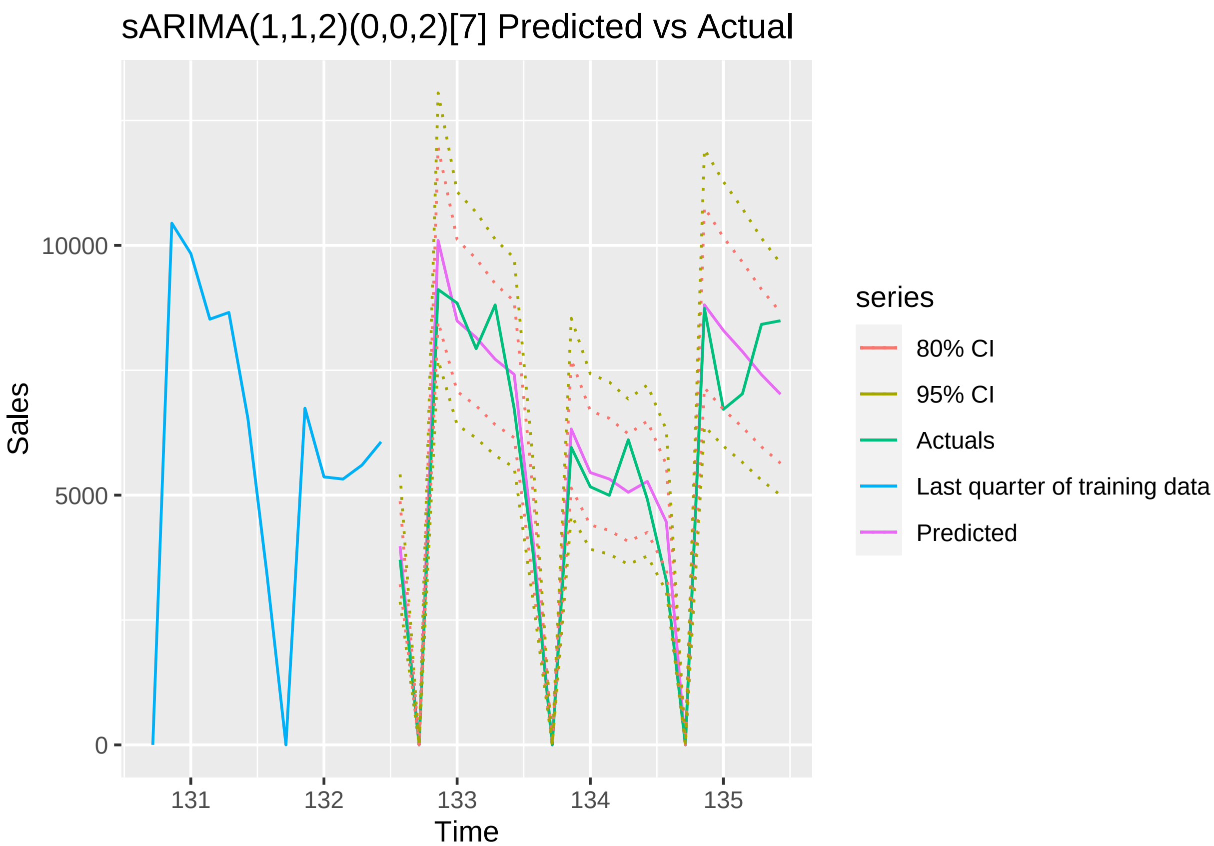
```
# Forecast the next three weeks
pred_tslm <- forecast(tslm_fit, h = 21, predictors_test_diff1, level = c(80, 95))
```

```
## Warning in forecast.lm(tslm_fit, h = 21, predictors_test_diff1, level = c(80, :
## newdata column names not specified, defaulting to first variable required.
```

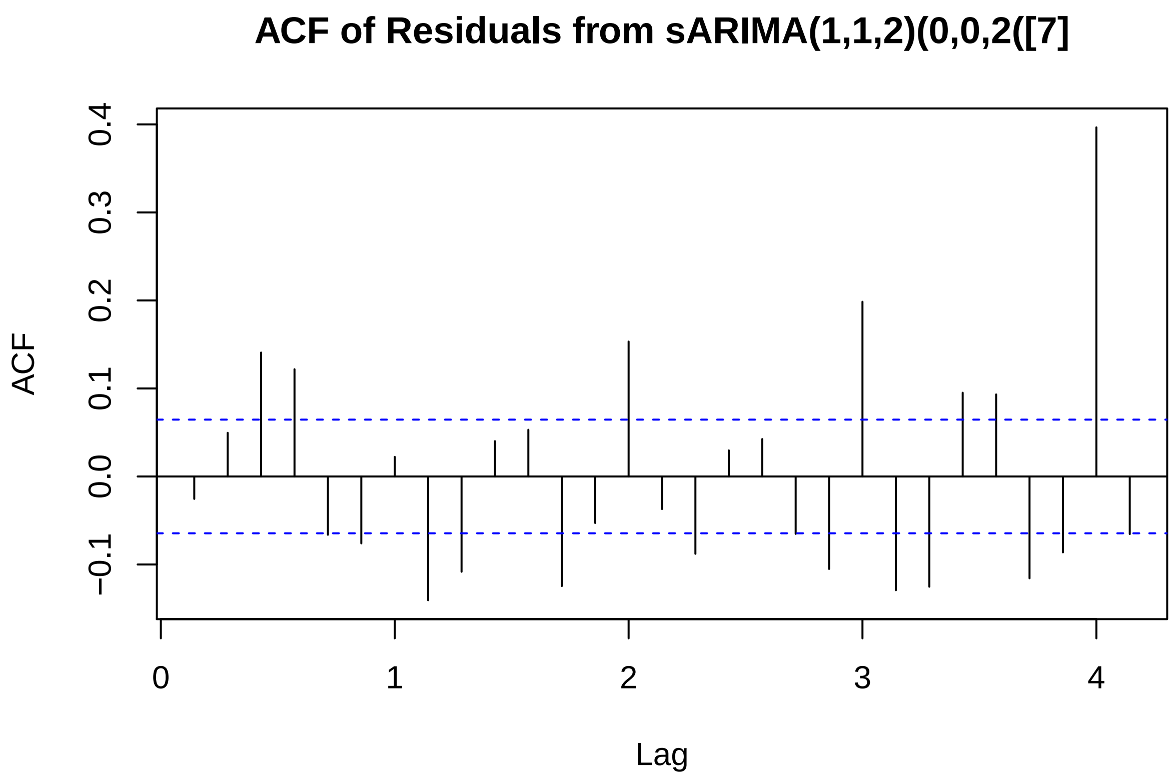
```
pred_lm <- forecast(linear_arima_fit, h = 21, xreg = predictors_test, level= c(80, 95))

# Because the TSLM model used differenced data, its forecasts are also the differences between observat
```

```r
# Take the last data point of the training data
pred_tslm_concat <- c(last_train, pred_tslm$mean)

# Add the TSLM predictions to the last observation and remove the first observation
pred_tslm_act <- cumsum(pred_tslm_concat)
pred_tslm_act <- head(pred_tslm_act, -1)
```

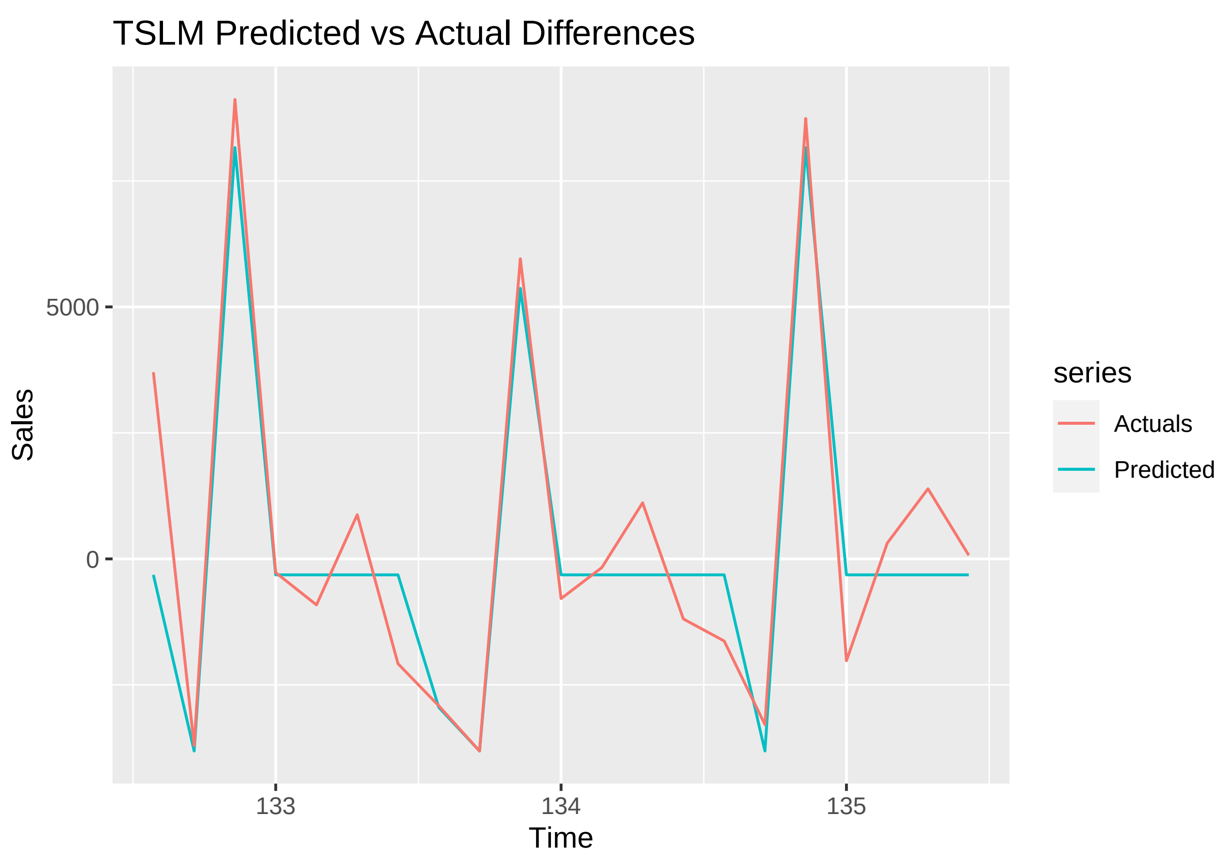
```r
#ARIMA with linear error actuals + forecast
autoplot(pred_lm$mean, series = "Predicted", ylab = "Sales") + autolayer(sales_test, series = "Actuals")
```



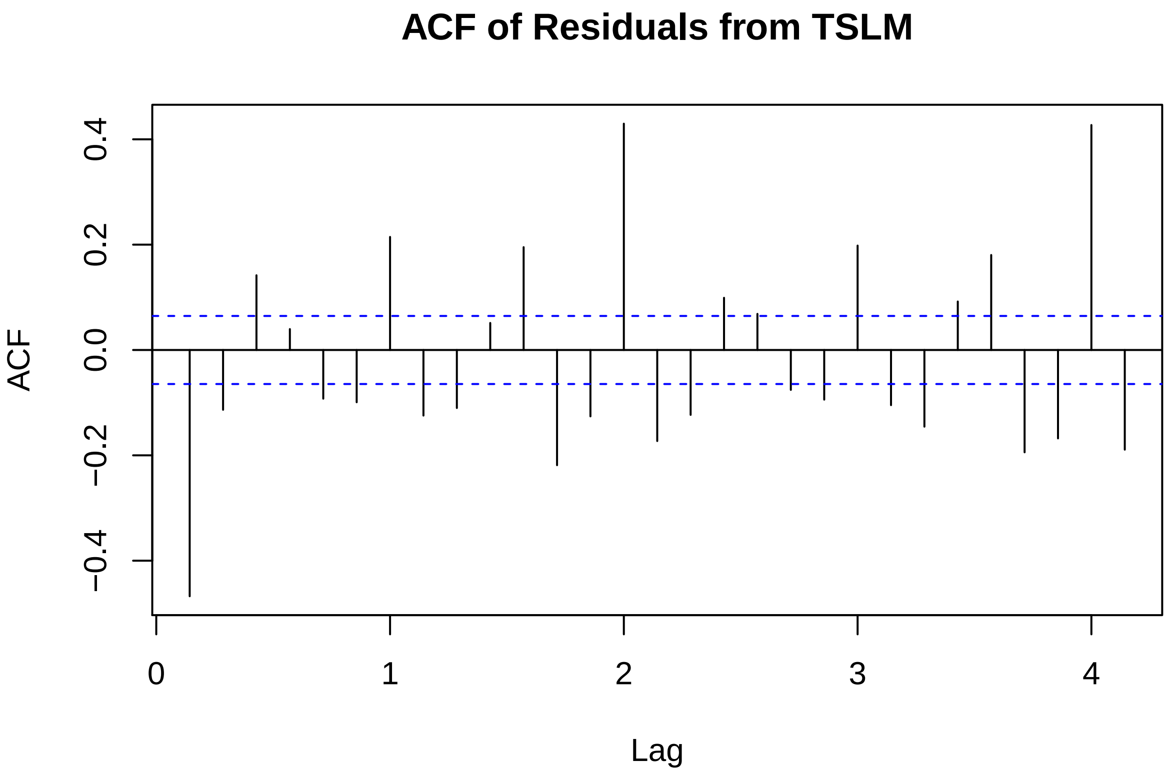

```r
#ARIMA with linear error ACF
acf(linear_arima_fit$residuals, main = "ACF of Residuals from sARIMA(1,1,2)(0,0,2([7]")
```




```r
#TSLM of differences
autoplot(pred_tslm$mean, series = "Predicted") + autolayer(sales_test_diff1, series = "Actuals") + xlab
```



```r
#TSLM ACF
acf(tslm_fit$residuals, main = "ACF of Residuals from TSLM")
```




```r
# Convert test data to a dataframe
sales_test_df <- data.frame(Y=as.matrix(sales_test))

pred_lm_df <- data.frame(Y=as.matrix(pred_lm$mean))

lm_df <- cbind(sales_test_df, pred_lm_df)
colnames(lm_df) <- c("Y", "pred_lm_act")

tslm_df <- as.data.frame(cbind(sales_test_df, pred_tslm_act))
colnames(tslm_df) <- c("Y", "pred_tslm_act")

# RMSE
rmse_tslm <- rmse(tslm_df, "Y", "pred_tslm_act")
rmse_lm <- rmse(lm_df, "Y", "pred_lm_act")

cat("RMSE for TSLM model is ", rmse_tslm$.estimate)
```

```
## RMSE for TSLM model is  5748.325
```

```r
cat("RMSE for linear model with ARIMA errors is ", rmse_lm$.estimate)
```

```
## RMSE for linear model with ARIMA errors is  764.3892
```

```r
# Test RMSEs for the linear models with ARIMA error selected by EACF
pred_eacf1 <- forecast(eacf1, h = 21, xreg = predictors_test)
pred_eacf1_df <- data.frame(Y=as.matrix(pred_eacf1$mean))
eacf1_df <- cbind(sales_test_df, pred_eacf1_df)
colnames(eacf1_df) <- c("Y", "pred_eacf1_act")
```

```r
rmse_eacf1 <- rmse(eacf1_df, "Y", "pred_eacf1_act")

cat("RMSE for ARIMA with sARIMA(3,1,2)(0,0,7) model is ", rmse_eacf1$.estimate)
```

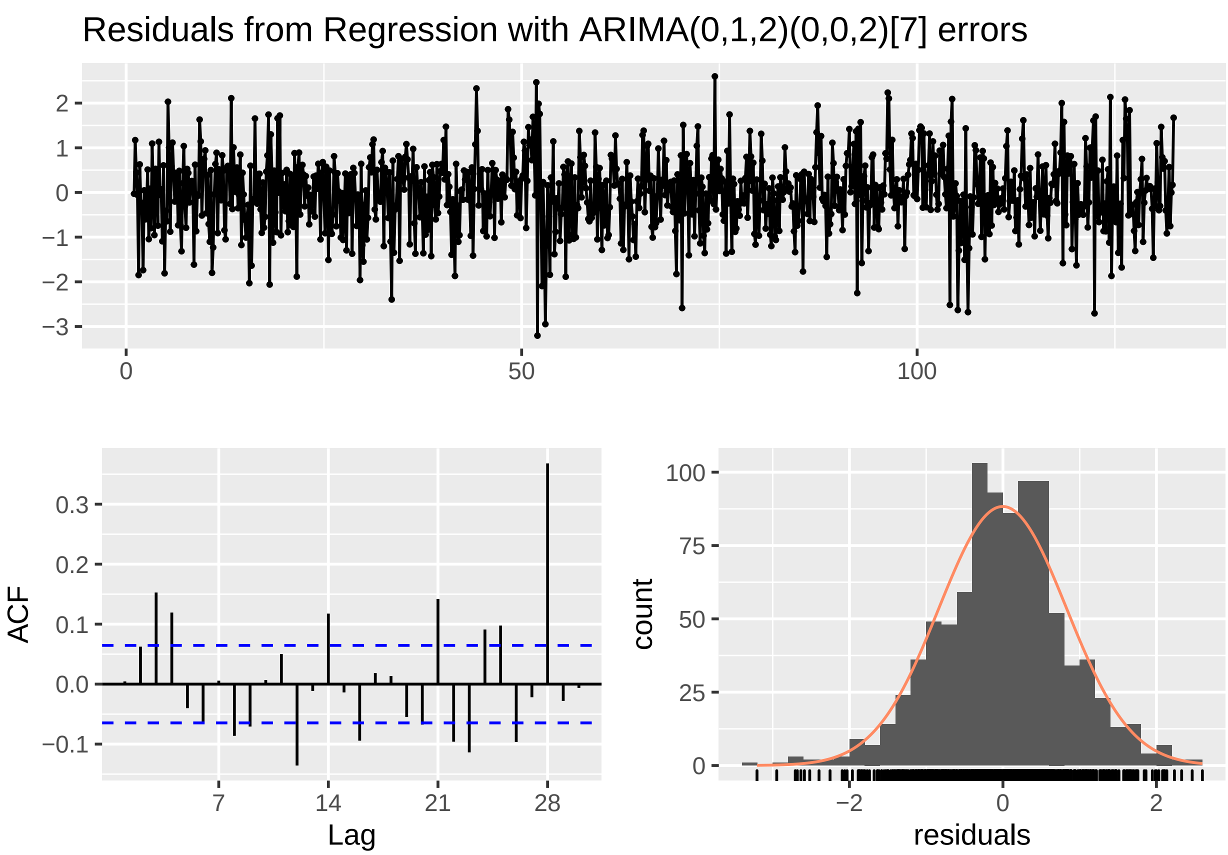
```
## RMSE for ARIMA with sARIMA(3,1,2)(0,0,7) model is  790.1098
```

ADDING IN ALL PREDICTORS

```r
# Retrain the model using all predictors
predictors_all <- train[, c("Open", "Promo", "StateHoliday", "SchoolHoliday", "DayOfWeek")] #excluded:
linear_arima_fit_all <- auto.arima(sales, xreg = predictors_all, lambda = "auto", seasonal = TRUE, allo
summary(linear_arima_fit_all)
```

```
## Series: sales
## Regression with ARIMA(0,1,2)(0,0,2)[7] errors
## Box Cox transformation: lambda= 0.2045416
##
## Coefficients:
##           ma1      ma2    sma1    sma2   drift     Open   Promo  StateHoliday
##       -0.8801  -0.1016  0.3129  0.3478  0.0016  25.7623  2.3166       -1.3564
## s.e.   0.0356   0.0362  0.0377  0.0285  0.0009   0.1608  0.0654        0.1331
##       SchoolHoliday  DayOfWeek
##              0.1365    -0.4723
## s.e.         0.1010     0.0297
##
## sigma^2 estimated as 0.7008:  log likelihood=-1139.14
## AIC=2300.29   AICc=2300.58   BIC=2353.35
##
## Training set error measures:
##                    ME     RMSE      MAE MPE MAPE      MASE      ACF1
## Training set 38.88672 718.8552 509.9592 NaN  Inf 0.2845409 0.1571306
```

```r
checkresiduals(linear_arima_fit_all) #Ljung Box p-value is 2.2e-16
```

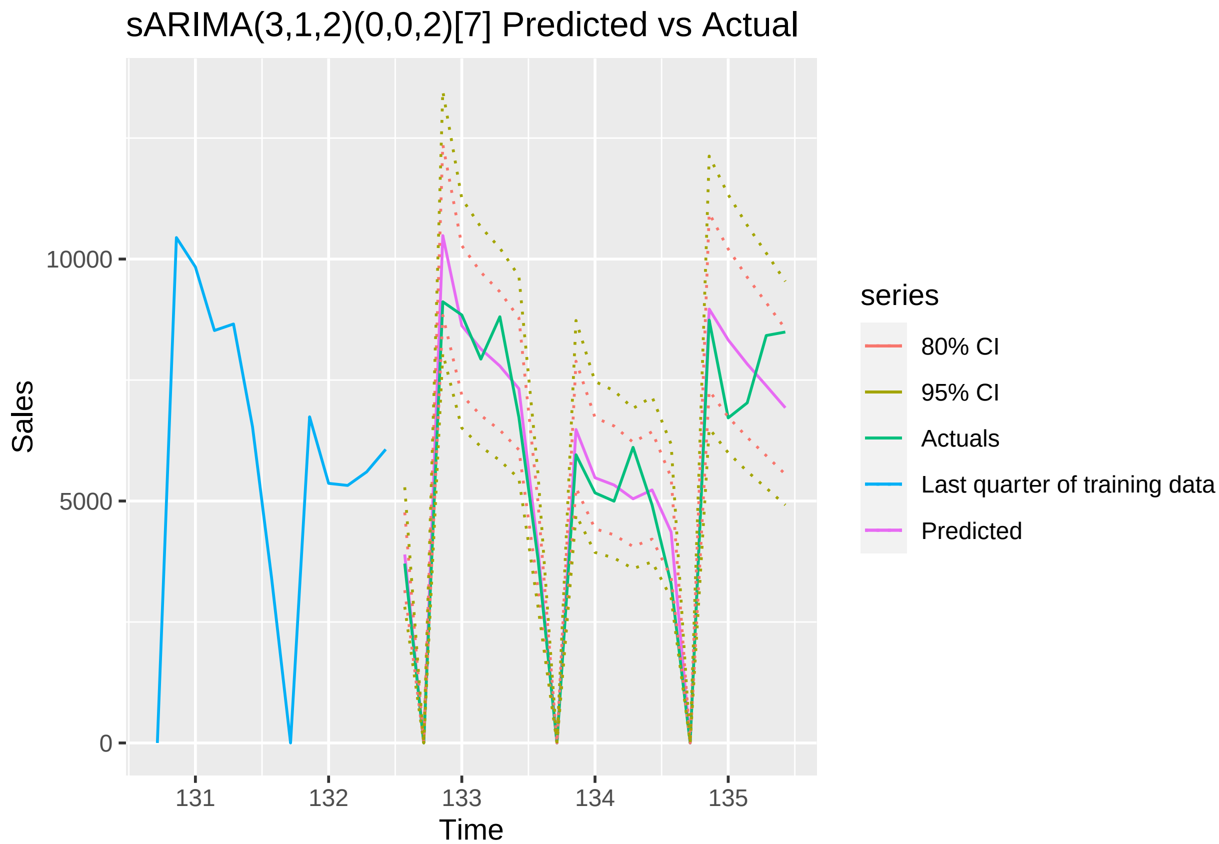



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,1,2)(0,0,2)[7] errors
## Q* = 88.181, df = 4, p-value < 2.2e-16
##
## Model df: 10.   Total lags used: 14
```

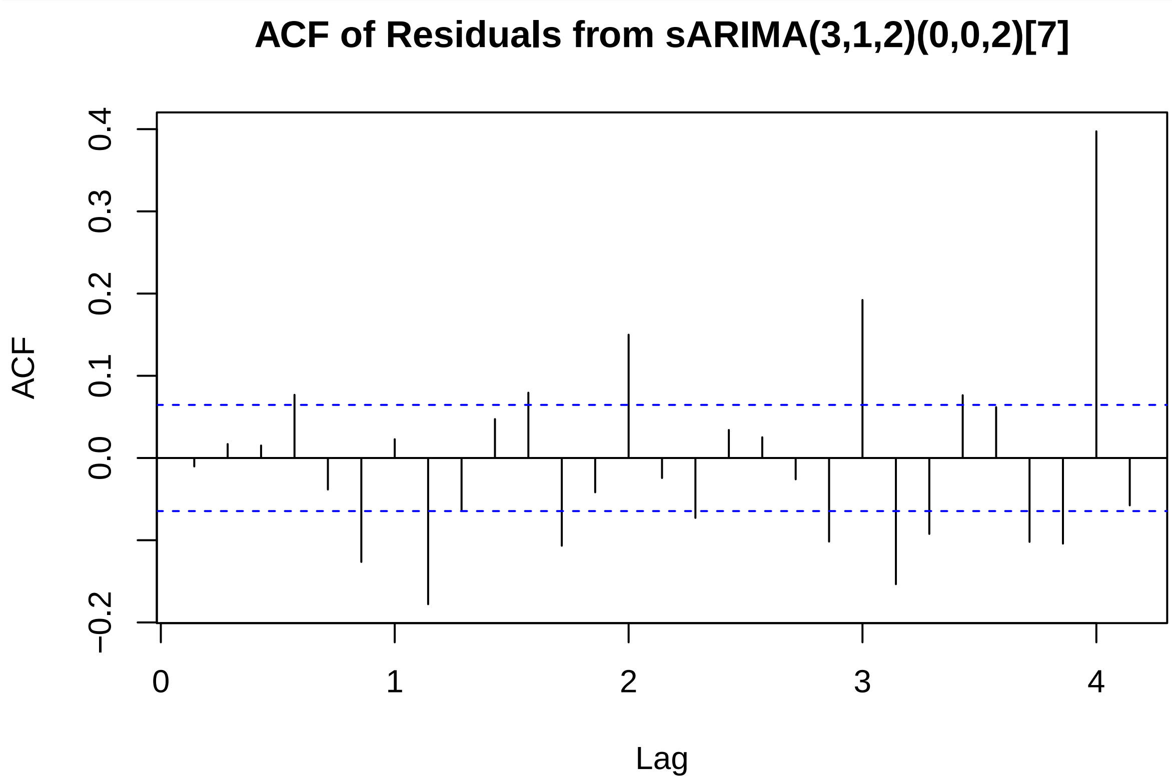
```
# Reforecast using updated model
predictors_test_all <- test[, c("Open", "Promo", "StateHoliday", "SchoolHoliday", "DayOfWeek")]
pred_lm_all <- forecast(linear_arima_fit_all, h = 21, xreg = predictors_test_all, level= c(80, 95))

# Reformat the predicted data
pred_lm_df_all <- data.frame(Y=as.matrix(pred_lm_all$mean))
lm_df_all <- cbind(sales_test_df, pred_lm_df_all)
colnames(lm_df_all) <- c("Y", "pred_lm_act_all")

# RMSE
rmse_lm_all <- rmse(lm_df_all, "Y", "pred_lm_act_all")
cat("RMSE for ARIMA with linear errors model is ", rmse_lm_all$.estimate)
```

```
## RMSE for ARIMA with linear errors model is  830.5224
```

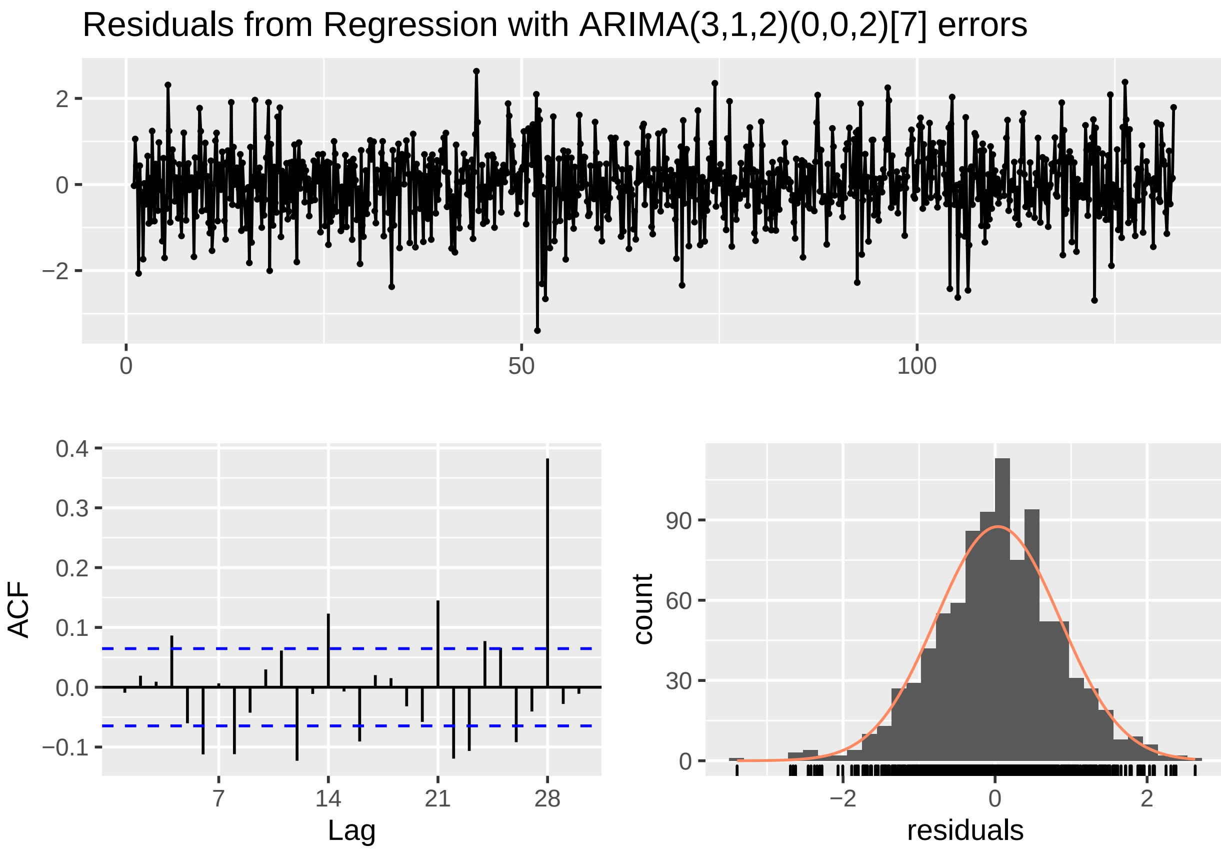
```
#ARIMA with linear error actuals + forecast
pred_eacf1 <- forecast(eacf1, h = 21, xreg = predictors_test, level= c(80, 95))

autoplot(pred_eacf1$mean, series = "Predicted", ylab = "Sales") + autolayer(sales_test, series = "Actua
```




```r
#EACF ARIMA with linear error ACF
acf(eacf1$residuals, main = "ACF of Residuals from sARIMA(3,1,2)(0,0,2)[7]")
```



```r
# Retrain using all predictors
eacf1_all <- Arima(sales, xreg = predictors_all, order = c(3, 1, 2), seasonal=list(order=c(0,0,2), peri
checkresiduals(eacf1_all)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(3,1,2)(0,0,2)[7] errors
## Q* = 68.693, df = 3, p-value = 8.105e-15
##
## Model df: 12.   Total lags used: 15
```

```r
# Predict
pred_eacf1_all <- forecast(eacf1_all, h = 21, xreg = predictors_test_all, level= c(80, 95))

# Reformat the predicted data
pred_eacf1_df_all <- data.frame(Y=as.matrix(pred_eacf1_all$mean))
eacf_df_all <- cbind(sales_test_df, pred_eacf1_df_all)
colnames(eacf_df_all) <- c("Y", "pred_lm_act_all")

# RMSE
rmse_eacf1_all <- rmse(eacf_df_all, "Y", "pred_lm_act_all")
cat("RMSE for ARIMA with linear errors model is ", rmse_eacf1_all$.estimate)
```

```
## RMSE for ARIMA with linear errors model is  847.2358
```