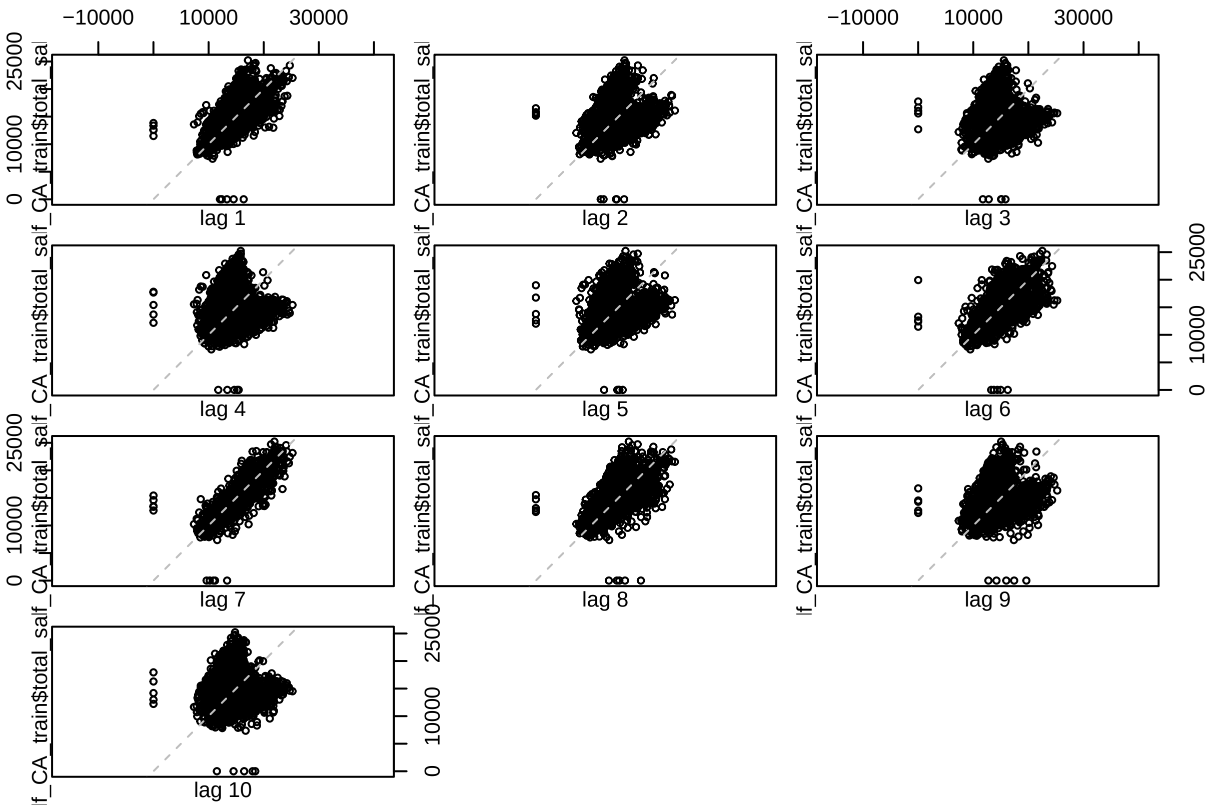
TimeSeries

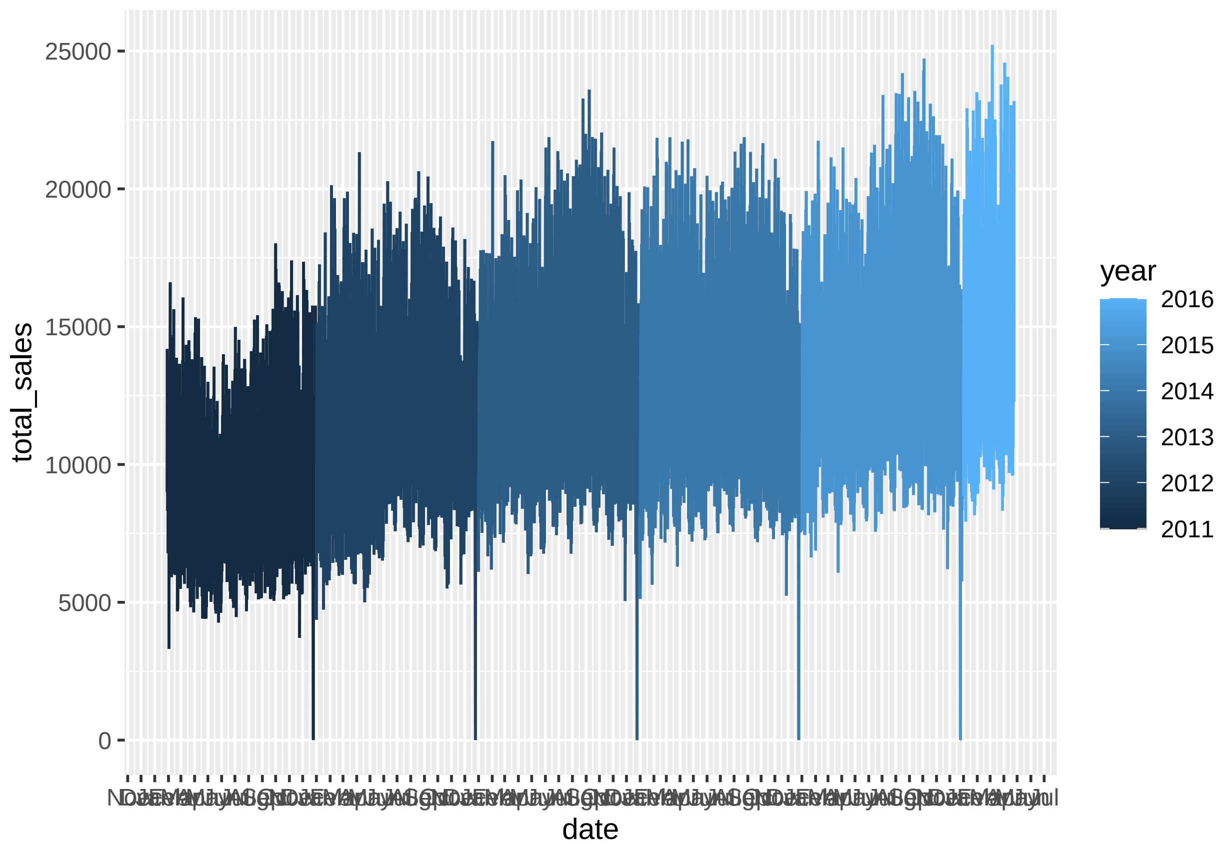
Graham Dynis

2024-11-06

```r
#Time Series AutoCorrelation Plot

# plot(df_CA_train[1:365,]$total_sales)
# plot(df_CA_train[1:365,]$mean_sell_price)
lag.plot(df_CA_train$total_sales, 10)
```




```r
p <- df_summary %>%
  mutate(
    date = as.Date(date,"%Y-%m-%d")
  ) %>%
  ggplot(aes(date, total_sales, color = year)) +
    scale_x_date(date_breaks = "1 month", date_labels = "%b")
p + geom_line()
```




```r
df_CA_train = dummy_cols(df_CA, select_columns = c("weekday"))

df_CA_train = subset(df_CA_train, select = -c(weekday))

rownames(df_CA_train) = df_CA_train$date
df_CA_train$date = NULL



train_df = df_CA_train

total_sales_ts <- ts(train_df$total_sales, frequency = 30)  # Adjust frequency based on your data (e.g.

# Define exogenous variables (all other columns except total_sales)
exogenous_vars <- train_df[, setdiff(names(train_df), c("total_sales", "weekday_Sunday"))]

library(forecast)

model <- auto.arima(total_sales_ts, xreg = as.matrix(exogenous_vars), seasonal = TRUE)

summary(model)
```

```
## Series: total_sales_ts
## Regression with ARIMA(3,1,4) errors
##
```

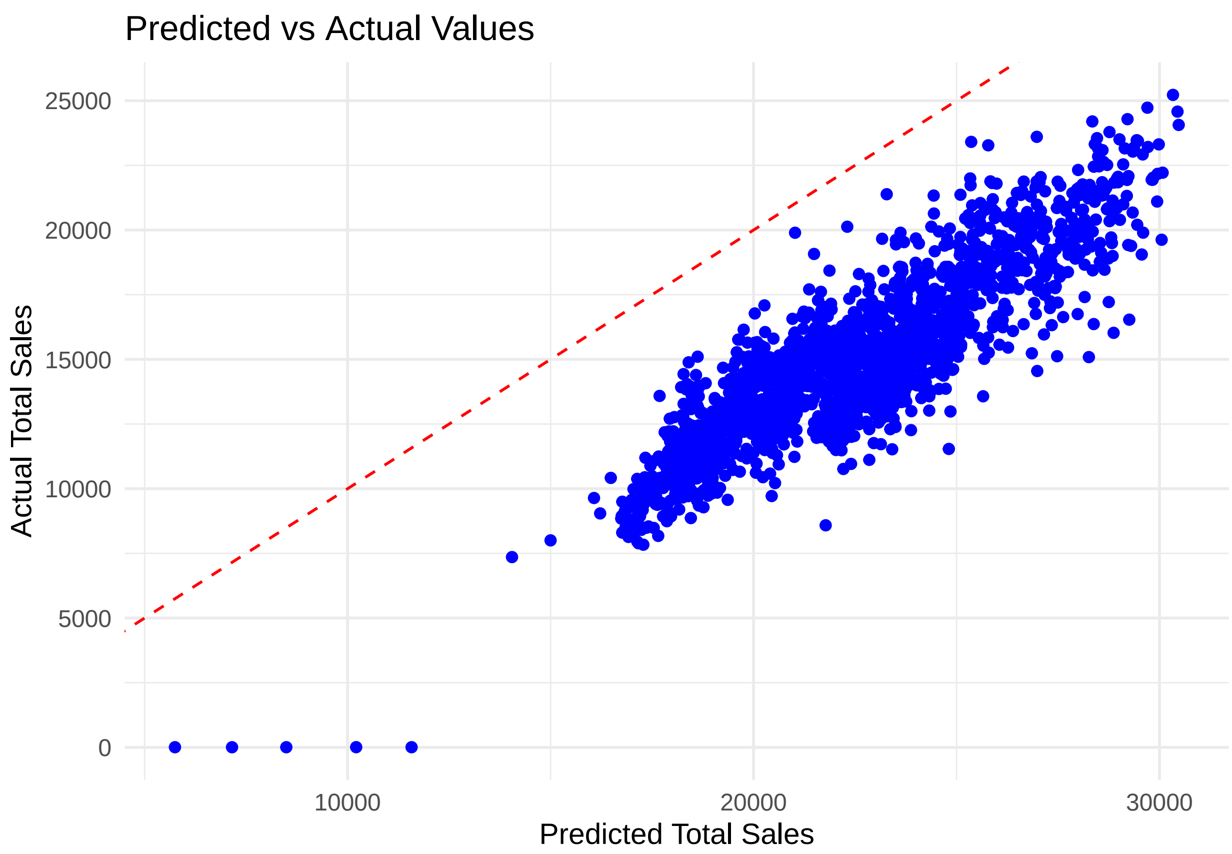
```
## Coefficients:
##          ar1      ar2      ar3      ma1     ma2      ma3      ma4   drift
##       1.2050  -0.9345  -0.0179  -1.7904  1.3648  -0.2641  -0.2225  4.1574
## s.e.  0.1106   0.1315   0.1000   0.1079  0.2034   0.1774   0.0714  2.7025
##         snap_CA  mean_sell_price  isChristmas    isEaster  isFathersDay
##       1157.3582        -3901.757  -13358.0057  -2429.0217    -1837.4382
## s.e.    58.6731         3688.789     398.6897    400.5022      446.8356
##       isHalloween  isMothersDay   isNewYear  isSuperBowl  isThanksgiving
##        -1483.1774     -2915.342  -4245.1484   -2049.4046      -3761.9283
## s.e.     398.6184       399.565    400.0467     367.2531        402.2365
##       isValentinesDay   isEvent  weekday_Friday  weekday_Monday
##            -1797.8463  444.2379      -3931.9090      -4151.9508
## s.e.         362.6347   88.3466        159.0697        103.7292
##       weekday_Saturday  weekday_Thursday  weekday_Tuesday  weekday_Wednesday
##              -486.0676        -5613.1016       -5383.1371         -5734.8379
## s.e.          103.4643          189.0862         158.9495           188.9024
##
## sigma^2 = 957754:  log likelihood = -15867.25
## AIC=31788.5   AICc=31789.3   BIC=31938.51
##
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.7284513 971.7181 700.6106 6.736395 65.28655 0.2135417
##                       ACF1
## Training set -0.0003675056
```

```
#### RMSE Calculation ####
test_exogenous_vars <- train_df[, setdiff(names(train_df), c("total_sales", "weekday_Sunday"))]

forecast_values <- forecast(model, xreg = as.matrix(test_exogenous_vars), h = nrow(train_df))

predicted_values <- as.numeric(forecast_values$mean)
actual_values <- train_df$total_sales

rmse <- sqrt(mean((actual_values - predicted_values)^2))

print(paste("Root Mean Squared Error (RMSE) for training dataset:", round(rmse, 4)))
```

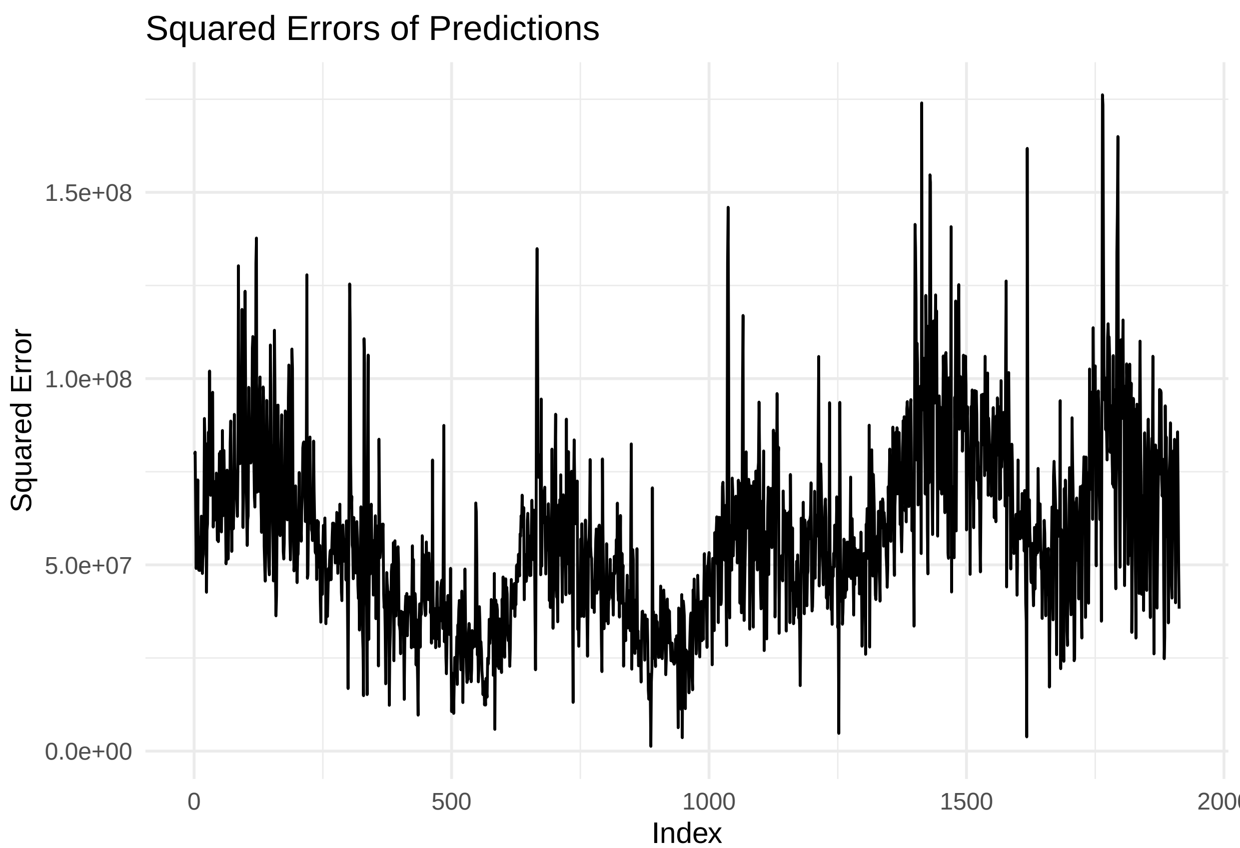
```
## [1] "Root Mean Squared Error (RMSE) for training dataset: 7626.3479"
```

```
#RMSE is 7626.3479
```

```
#### Plotting actual vs predicted

plot_data <- data.frame(Actual = actual_values, Predicted = predicted_values)

# Plot predicted vs actual values
ggplot(plot_data, aes(x = Predicted, y = Actual)) +
  geom_point(color = "blue") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +  # 1:1 line for reference
  labs(title = "Predicted vs Actual Values",
       x = "Predicted Total Sales",
       y = "Actual Total Sales") +
  theme_minimal()
```




```r
squared_errors <- (actual_values - predicted_values)^2

# Create a data frame for plotting
error_data <- data.frame(Index = 1:length(squared_errors), Squared_Error = squared_errors)

# Plot squared errors
ggplot(error_data, aes(x = Index, y = Squared_Error)) +
  geom_line() +  # Use geom_bar for a bar plot
  labs(title = "Squared Errors of Predictions",
       x = "Index",
       y = "Squared Error") +
  theme_minimal()
```




```
df_validation = read_csv("validation_set.csv")
```

```
## Rows: 84 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): state_id, weekday, event_name_1
## dbl  (7): month, year, snap_CA, snap_TX, snap_WI, total_sales, avg_sell_price
## lgl  (1): event_name_2
## date (1): date
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
df_ca_val = df_validation %>% filter(state_id == 'CA')
df_ca_val = subset(df_ca_val, select = -c(state_id, snap_WI, snap_TX))

df_ca_val$month = month.name[df_ca_val$month]

df_ca_val = df_ca_val %>%
            mutate(
              isChristmas = ifelse(event_name_1 == "Christmas", 1, 0),
    isEaster = ifelse(event_name_1 == "Easter", 1, 0),
    isFathersDay = ifelse(event_name_1 == "Father's day", 1, 0),
    isHalloween = ifelse(event_name_1 == "Halloween", 1, 0),
    isMothersDay = ifelse(event_name_1 == "Mother's day", 1, 0),
```

```r
    isNewYear = ifelse(event_name_1 == "NewYear", 1, 0),
    isSuperBowl = ifelse(event_name_1 == "SuperBowl", 1, 0),
    isThanksgiving = ifelse(event_name_1 == "Thanksgiving", 1, 0),
    isValentinesDay = ifelse(event_name_1 == "ValentinesDay", 1, 0),
    isEvent = ifelse(event_name_1 != "None" &
                       !event_name_1 %in% c("Christmas", "Easter", "Father's day",
                                            "Halloween", "Mother's day", "NewYear",
                                            "SuperBowl", "Thanksgiving", "ValentinesDay"), 1, 0)
           )
df_ca_val= subset(df_ca_val, select = -c(month, year, event_name_1, event_name_2))

df_ca_val_1 = dummy_cols(df_ca_val, select_columns = c("weekday"))

df_ca_val_1 = subset(df_ca_val_1, select = -c(weekday))

rownames(df_ca_val_1) = df_ca_val_1$date
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```r
df_ca_val_1 <- df_ca_val_1 %>%
  rename(mean_sell_price = avg_sell_price)
df_ca_val_1$date = NULL
```

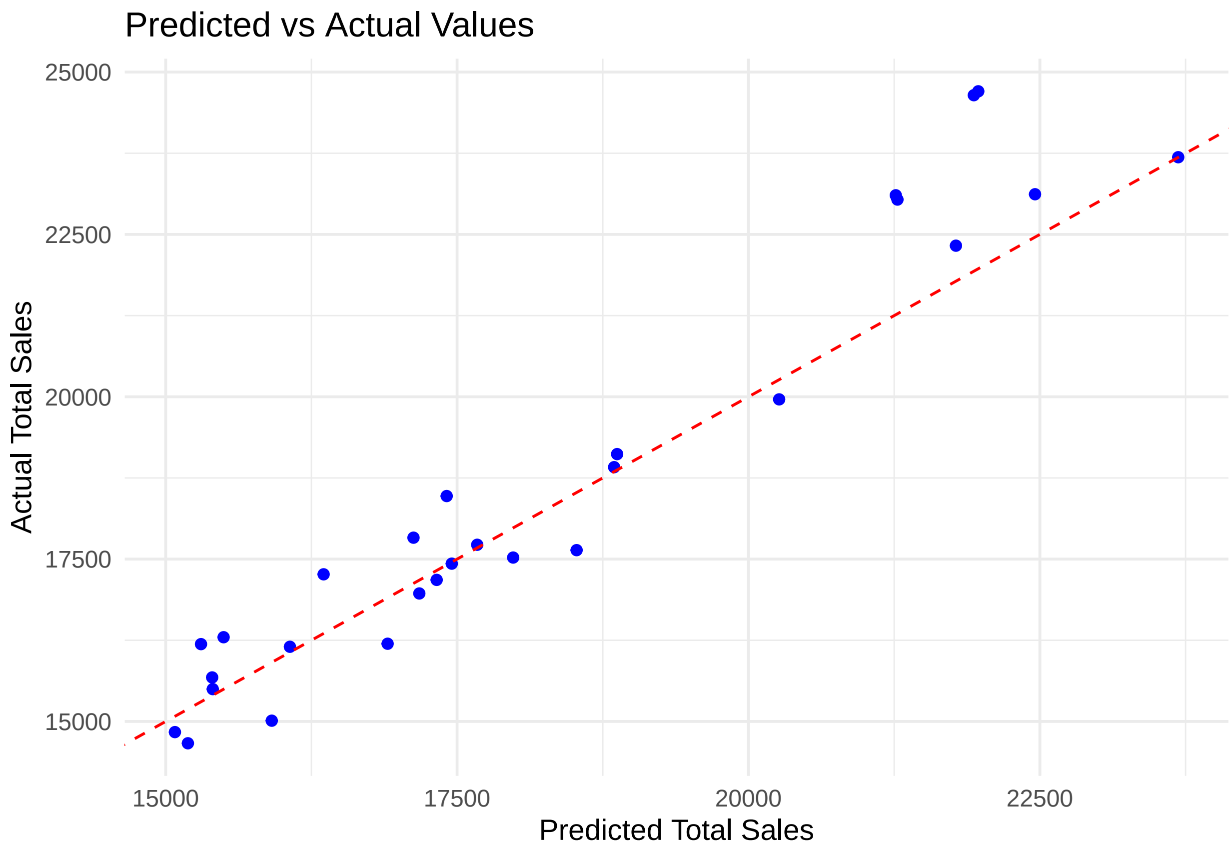
```r
#Replace NA's with 0

df_ca_val_1[is.na(df_ca_val_1)] <- 0

test_exogenous_vars <- df_ca_val_1[, setdiff(names(df_ca_val_1), c("total_sales", "weekday_Sunday"))]


forecast_values <- forecast(model, xreg = as.matrix(test_exogenous_vars), h = nrow(df_ca_val_1))

predicted_values <- as.numeric(forecast_values$mean)
actual_values <- df_ca_val_1$total_sales
```

```r
#### RMSE for test validation set
rmse <- sqrt(mean((actual_values - predicted_values)^2))

print(paste("Root Mean Squared Error (RMSE):", round(rmse, 4))) #1015.1854
```

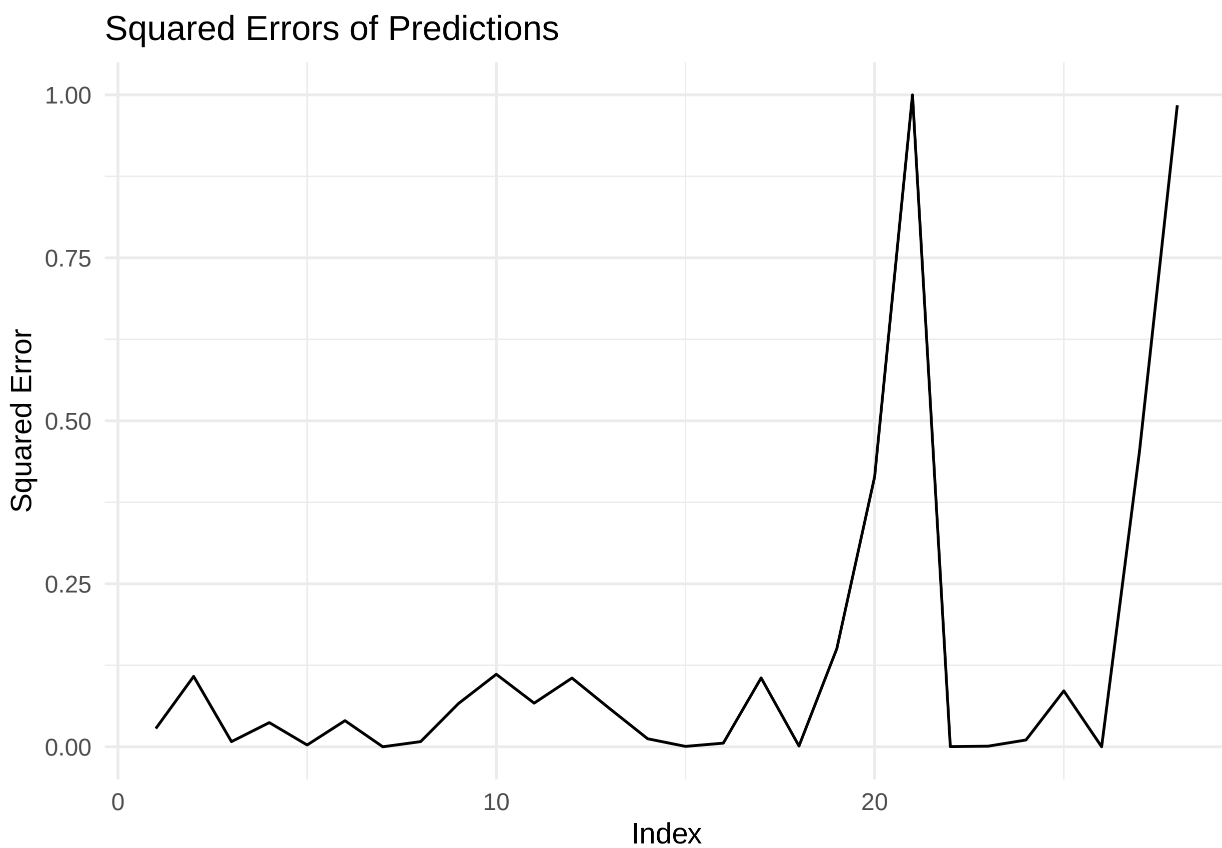
```
## [1] "Root Mean Squared Error (RMSE): 1015.1854"
```

```r
#### Plotting actual vs predicted

plot_data <- data.frame(Actual = actual_values, Predicted = predicted_values)

# Plot predicted vs actual values
ggplot(plot_data, aes(x = Predicted, y = Actual)) +
  geom_point(color = "blue") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +  # 1:1 line for reference
  labs(title = "Predicted vs Actual Values",
       x = "Predicted Total Sales",
```

```
       y = "Actual Total Sales") +
  theme_minimal()
```




```
squared_errors <- (actual_values - predicted_values)^2

# Create a data frame for plotting
error_data <- data.frame(Index = 1:length(squared_errors), Squared_Error = squared_errors)

error_data <- error_data %>%
  mutate(
    normalized_squared_errors = (Squared_Error - min(Squared_Error)) / (max(Squared_Error) - min(Square
)


# Plot squared errors
ggplot(error_data, aes(x = Index, y = normalized_squared_errors)) +
  geom_line() +  # Use geom_bar for a bar plot
  labs(title = "Squared Errors of Predictions",
       x = "Index",
       y = "Squared Error") +
  theme_minimal()
```




```r
### Clean and Arrange dataset for AR predictions
df_ca_val_2 <- df_validation %>%
  filter(state_id == 'CA') %>%
  arrange(date) %>%  # Make sure data is sorted by date
  mutate(
    total_sales_lag1 = lag(total_sales, 1),
    total_sales_lag2 = lag(total_sales, 2),
    total_sales_lag3 = lag(total_sales, 3),
    total_sales_lag4 = lag(total_sales, 4),
    total_sales_lag5 = lag(total_sales, 5),
    total_sales_lag6 = lag(total_sales, 6),
    total_sales_lag7 = lag(total_sales, 7)
  )

df_ca_val_2 = subset(df_ca_val_2, select = -c(state_id, snap_WI, snap_TX))

df_ca_val_2$month = month.name[df_ca_val_2$month]

df_ca_val_2 = df_ca_val_2 %>%
            mutate(
              isChristmas = ifelse(event_name_1 == "Christmas", 1, 0),
    isEaster = ifelse(event_name_1 == "Easter", 1, 0),
    isFathersDay = ifelse(event_name_1 == "Father's day", 1, 0),
    isHalloween = ifelse(event_name_1 == "Halloween", 1, 0),
    isMothersDay = ifelse(event_name_1 == "Mother's day", 1, 0),
    isNewYear = ifelse(event_name_1 == "NewYear", 1, 0),
```

```r
    isSuperBowl = ifelse(event_name_1 == "SuperBowl", 1, 0),
    isThanksgiving = ifelse(event_name_1 == "Thanksgiving", 1, 0),
    isValentinesDay = ifelse(event_name_1 == "ValentinesDay", 1, 0),
    isEvent = ifelse(event_name_1 != "None" &
                   !event_name_1 %in% c("Christmas", "Easter", "Father's day",
                                         "Halloween", "Mother's day", "NewYear",
                                         "SuperBowl", "Thanksgiving", "ValentinesDay"), 1, 0)
           )
df_ca_val_2= subset(df_ca_val_2, select = -c(year, event_name_1, event_name_2))

df_ca_val_2 = dummy_cols(df_ca_val_2, select_columns = c("weekday"))
df_ca_val_2 = dummy_cols(df_ca_val_2, select_columns = c("month"))

month_order <- c("month_April", "month_August", "month_December", "month_February",
                 "month_January", "month_July", "month_June", "month_March",
                 "month_May", "month_November", "month_October", "month_September")

for (month in month_order) {
  if (!month %in% colnames(df_ca_val_2)) {
    df_ca_val_2[[month]] <- 0
  }
}

month_columns <- df_ca_val_2[, month_order, drop = FALSE]
other_columns <- df_ca_val_2[, setdiff(names(df_ca_val_2), month_order), drop = FALSE]

# Combine other columns with the month columns in the correct order
df_ca_val_2 <- cbind(other_columns, month_columns)

df_ca_val_2 = subset(df_ca_val_2, select = -c(weekday, month))

rownames(df_ca_val_2) = df_ca_val_2$date

lag_columns <- paste("total_sales_lag", 1:7, sep = "")

# Remove rows where any of the lag columns have NA
df_ca_val_2 <- df_ca_val_2[!apply(df_ca_val_2[, lag_columns], 1, function(x) any(is.na(x))), ]

df_ca_val_2[is.na(df_ca_val_2)] <- 0

df_ca_val_2 <- df_ca_val_2 %>%
  rename(mean_sell_price = avg_sell_price)
df_ca_val_2$date = NULL
```

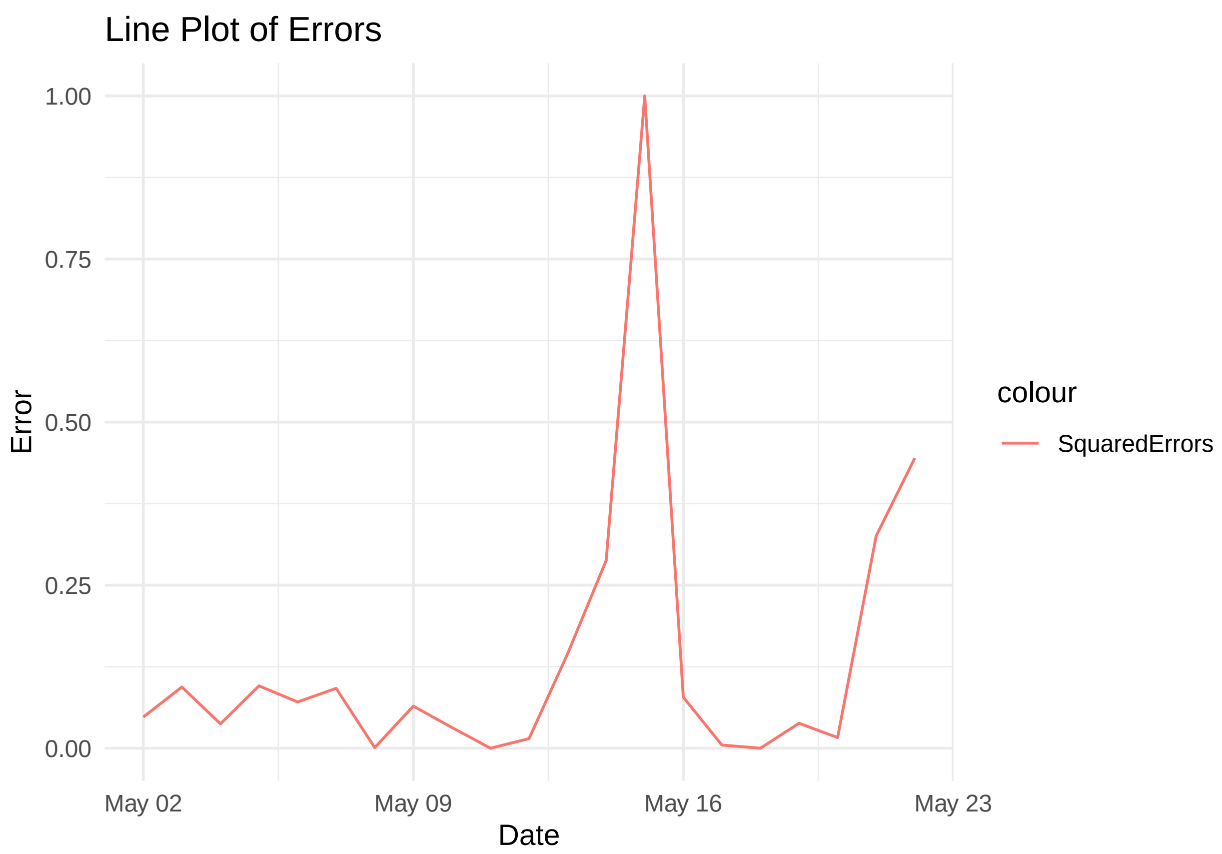
```r
#### RMSE for simple AR model

actual_y_ar = df_ca_val_2$total_sales
test_values_ar <- subset(df_ca_val_2, select = -c(total_sales))



predictions_ar = as.numeric(predict(lm_model_CA, test_values_ar))
```

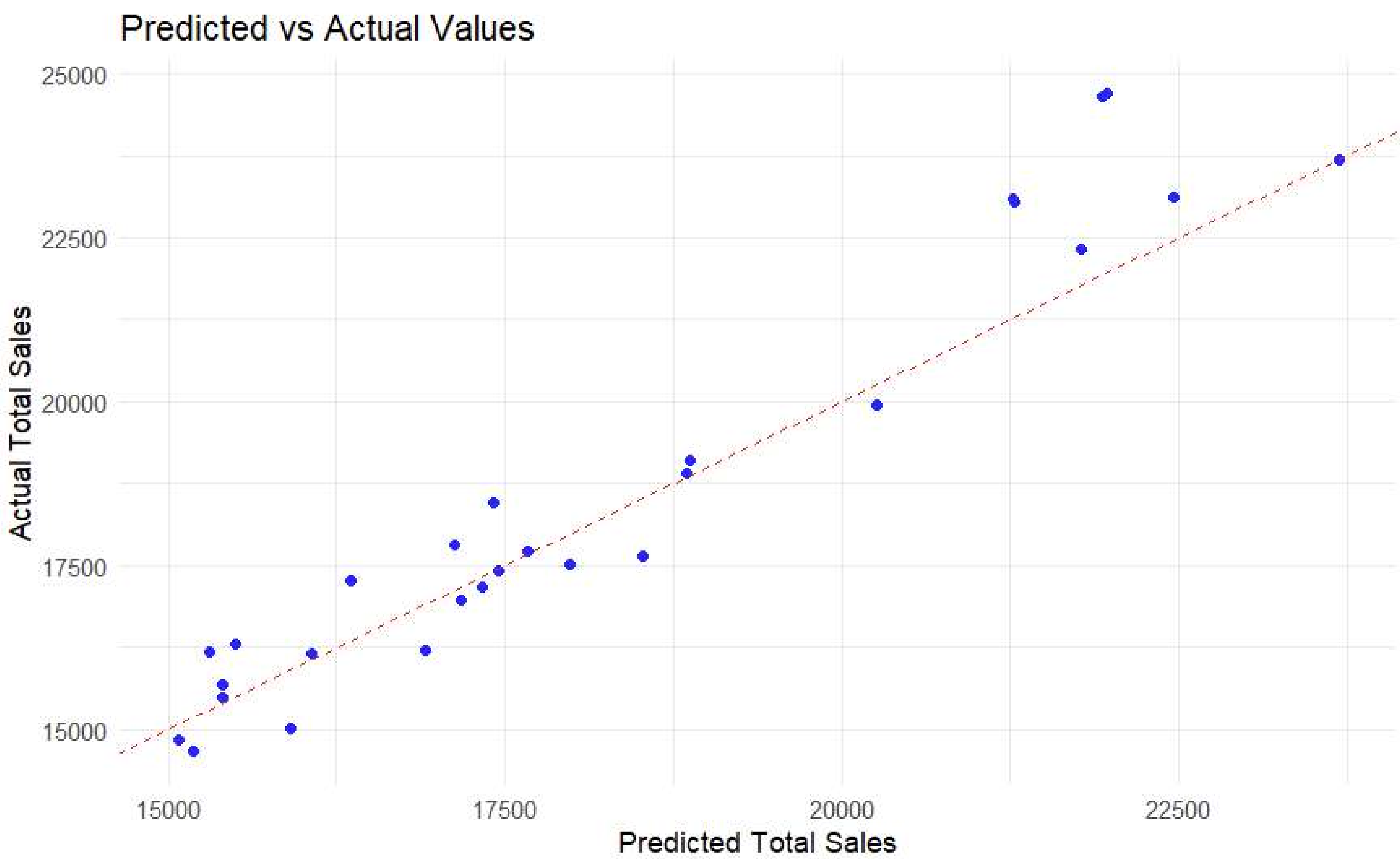
```
## Warning in predict.lm(lm_model_CA, test_values_ar): prediction from a
## rank-deficient fit may be misleading
```

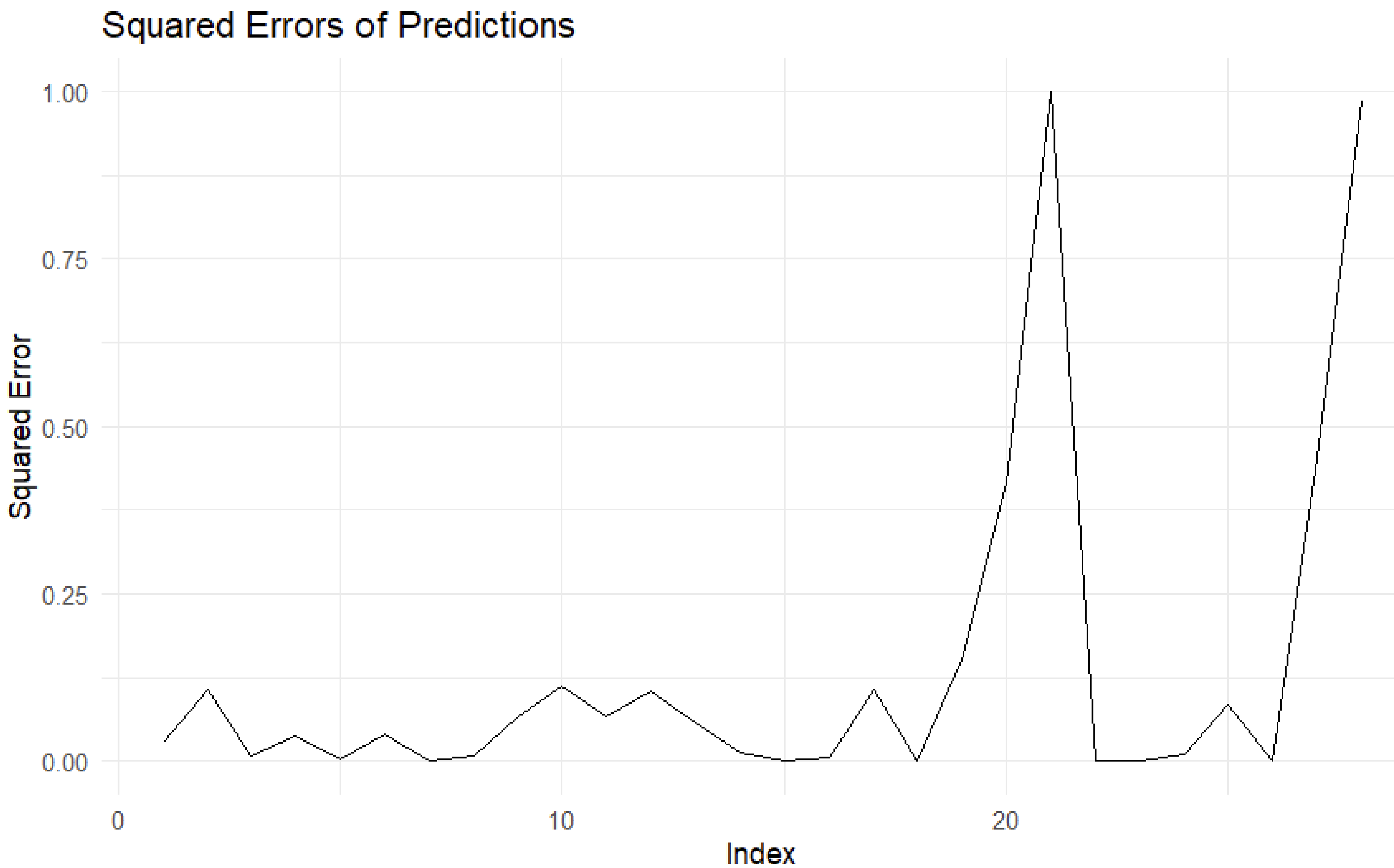
```
mse = mean((actual_y_ar - predictions_ar)^2, na.rm=TRUE)
rmse = sqrt(mse)
print(rmse) #1150.957 for California
```

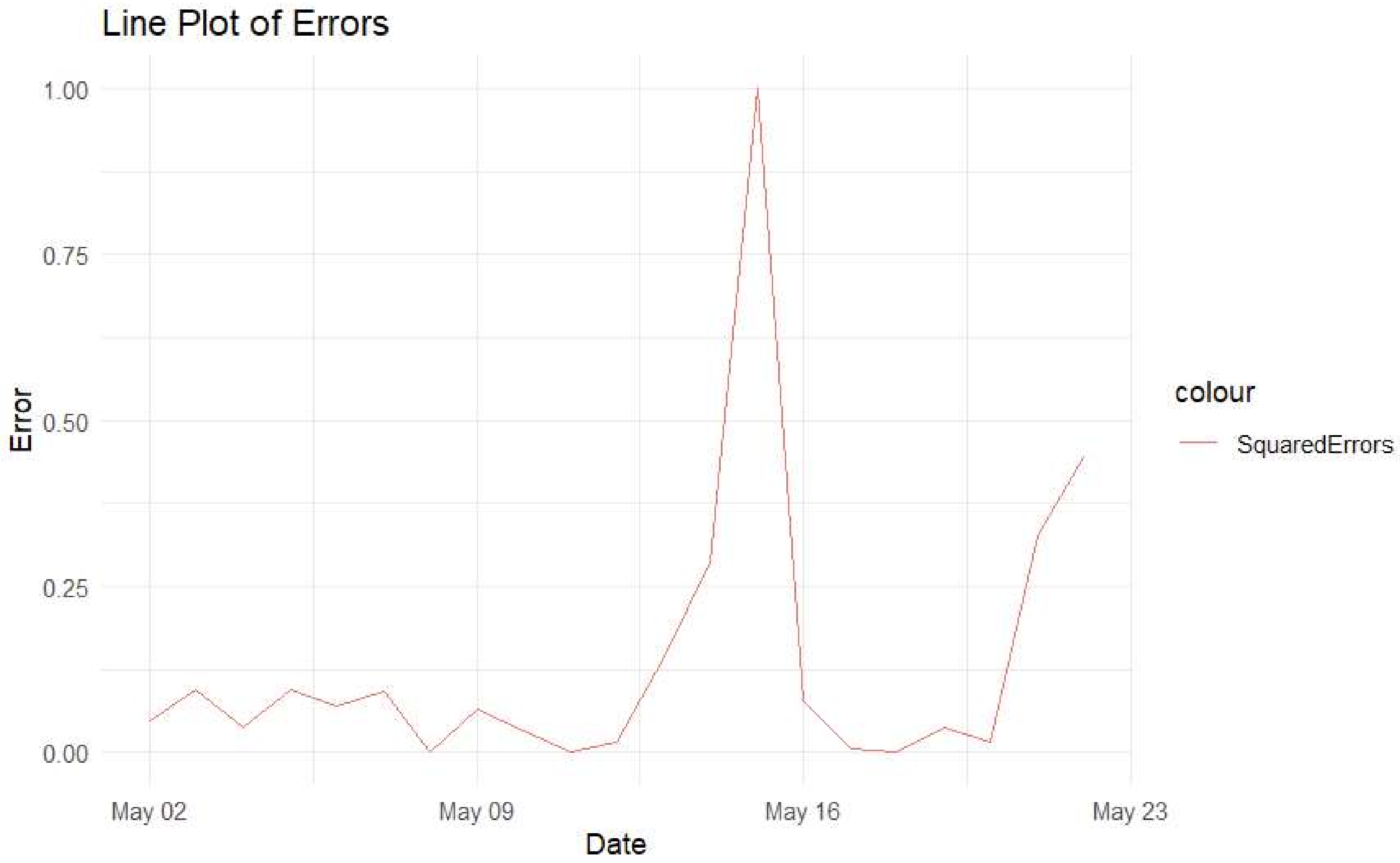
```
## [1] 1150.957
```

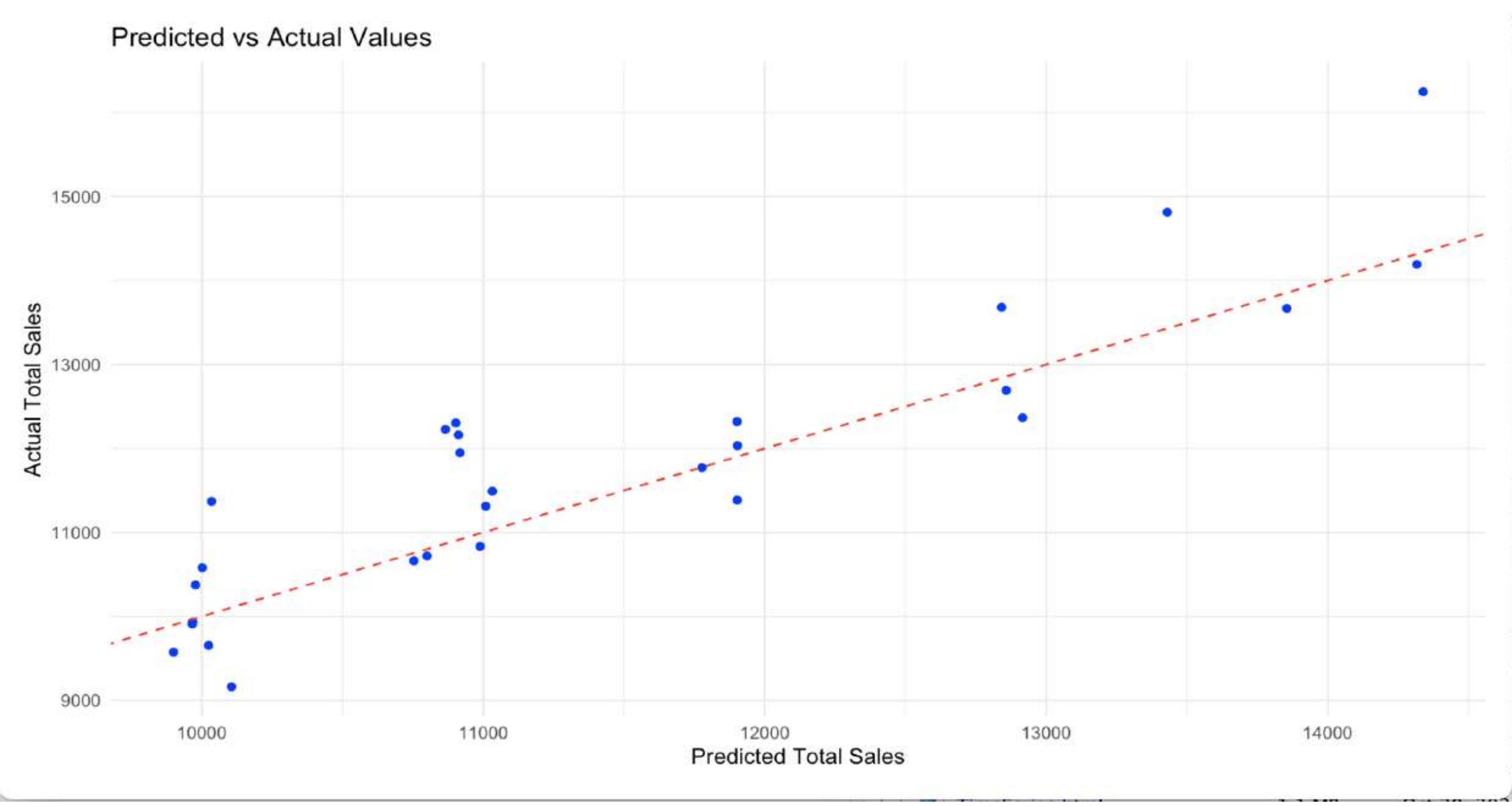
```
squared_errors = (actual_y_ar - predictions_ar)^2
errors = (actual_y_ar - predictions_ar)

plot_df_errors = data.frame(date = as.Date(rownames(df_ca_val_2)), errors = errors, squared_errors = sq

plot_df_errors <- plot_df_errors %>%
  mutate(
    normalized_errors = (errors - min(errors)) / (max(errors) - min(errors)),
    normalized_squared_errors = (squared_errors - min(squared_errors)) / (max(squared_errors) - min(squa
  )


ggplot(plot_df_errors, aes(x = date)) +
  geom_line(aes(y = normalized_squared_errors, color = "SquaredErrors"))+
  labs(title = "Line Plot of Errors", x = "Date", y = "Error") +
  theme_minimal()
```

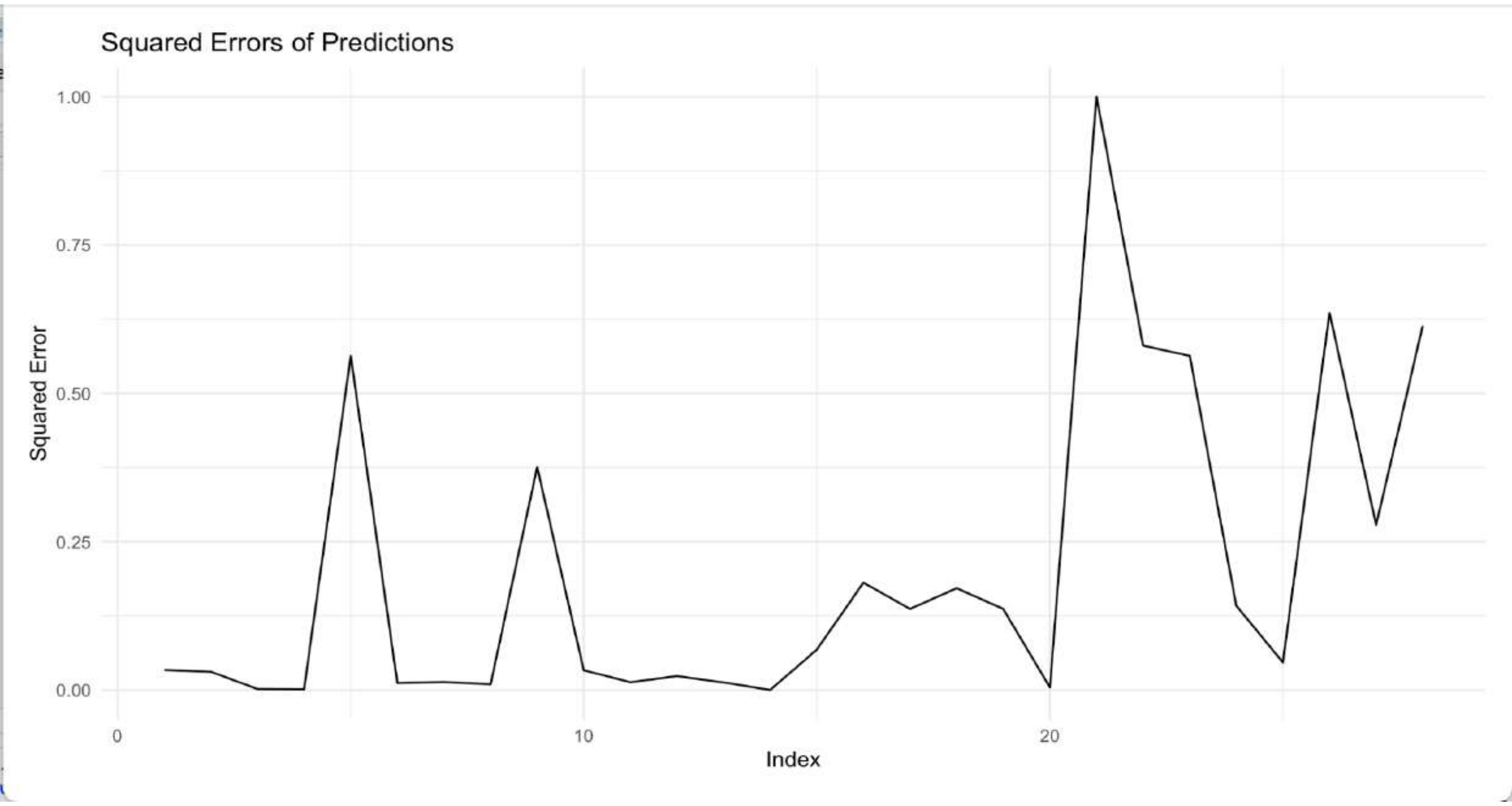


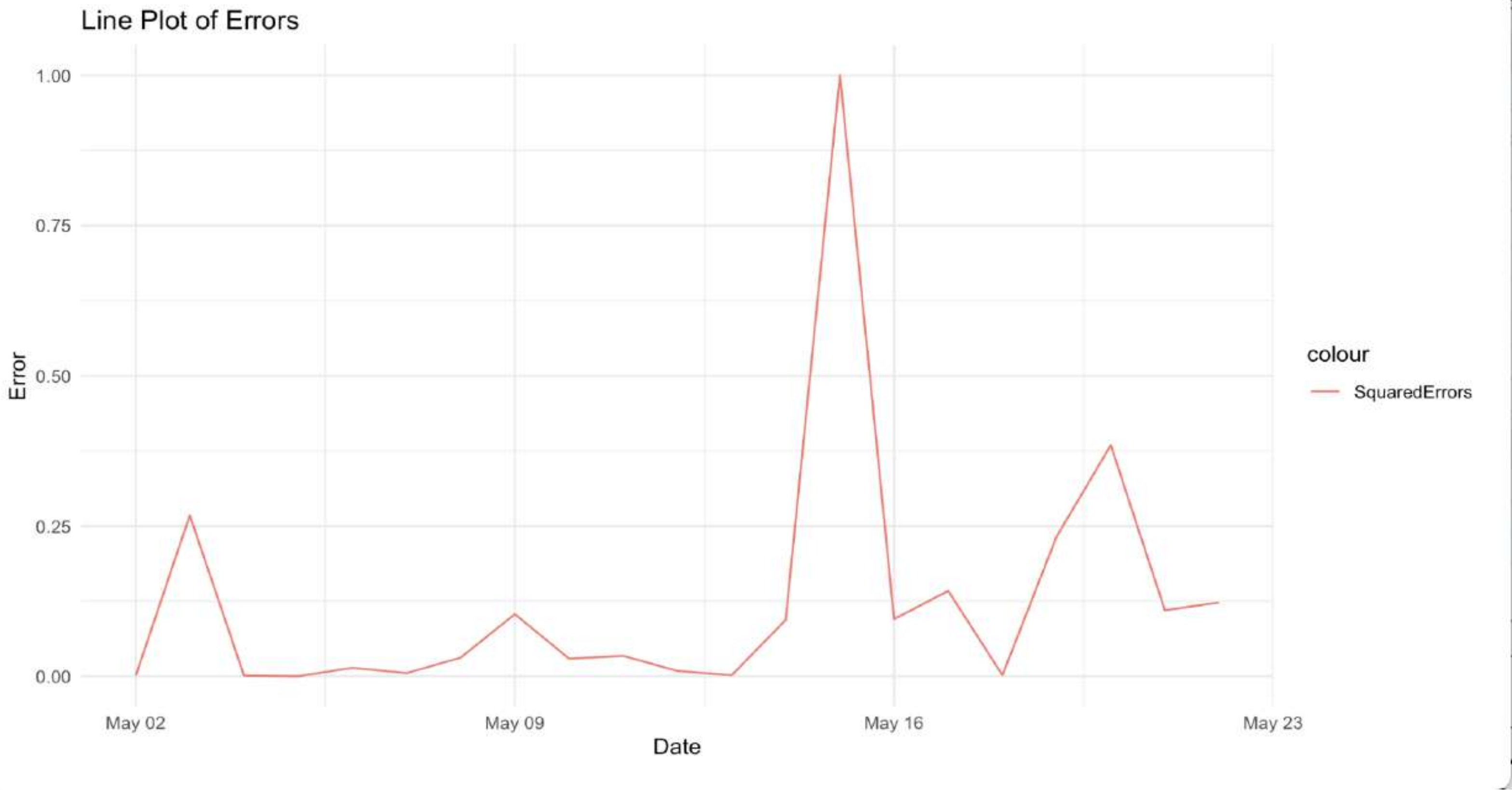

Predicted vs Actual Values
Actual Total Sales
Predicted Total Sales
25000
22500
20000
17500
15000
15000
17500
20000
22500

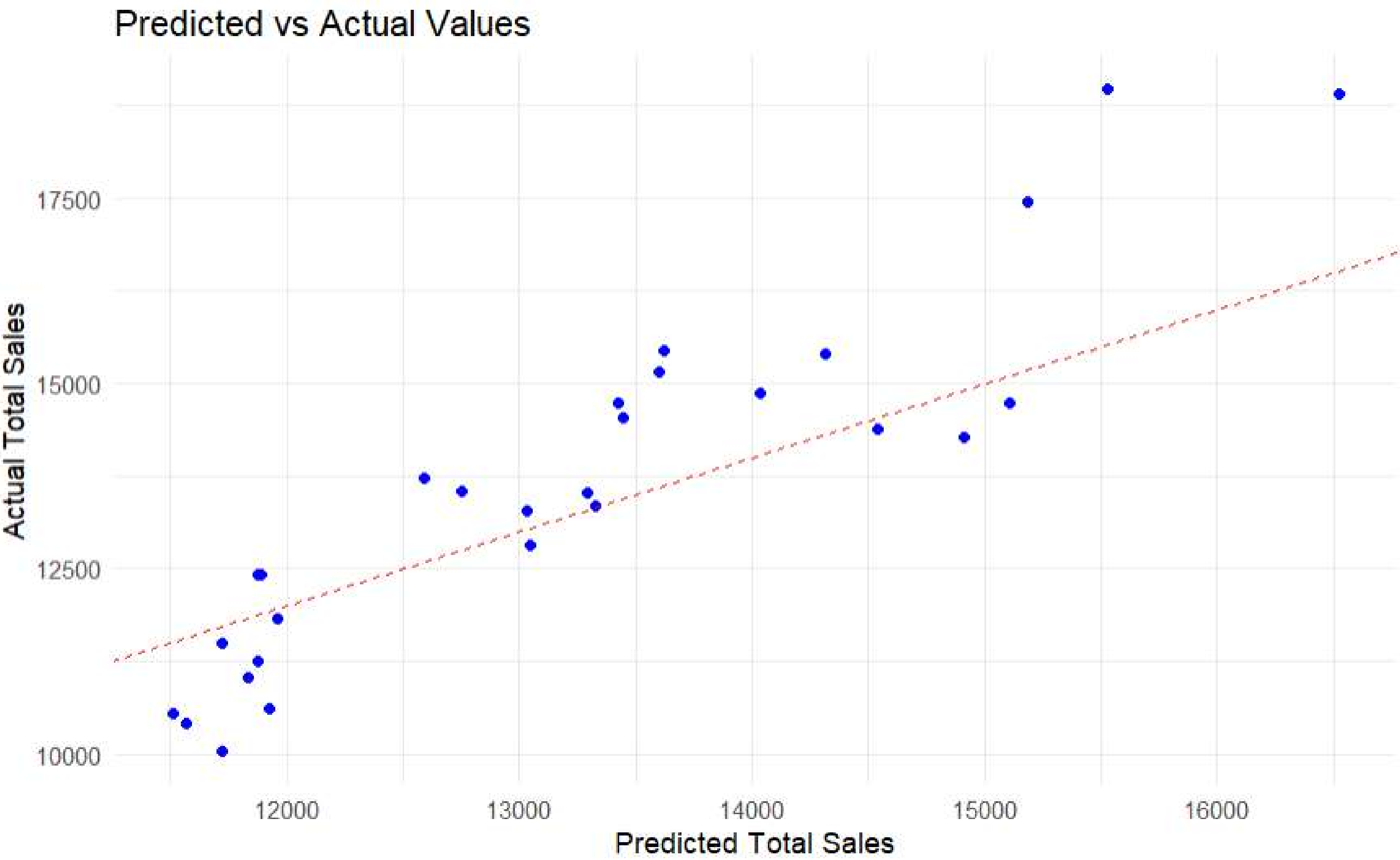

Squared Errors of Predictions
1.00
0.75
0.50
0.25
0.00
Squared Error
0
10
20
Index

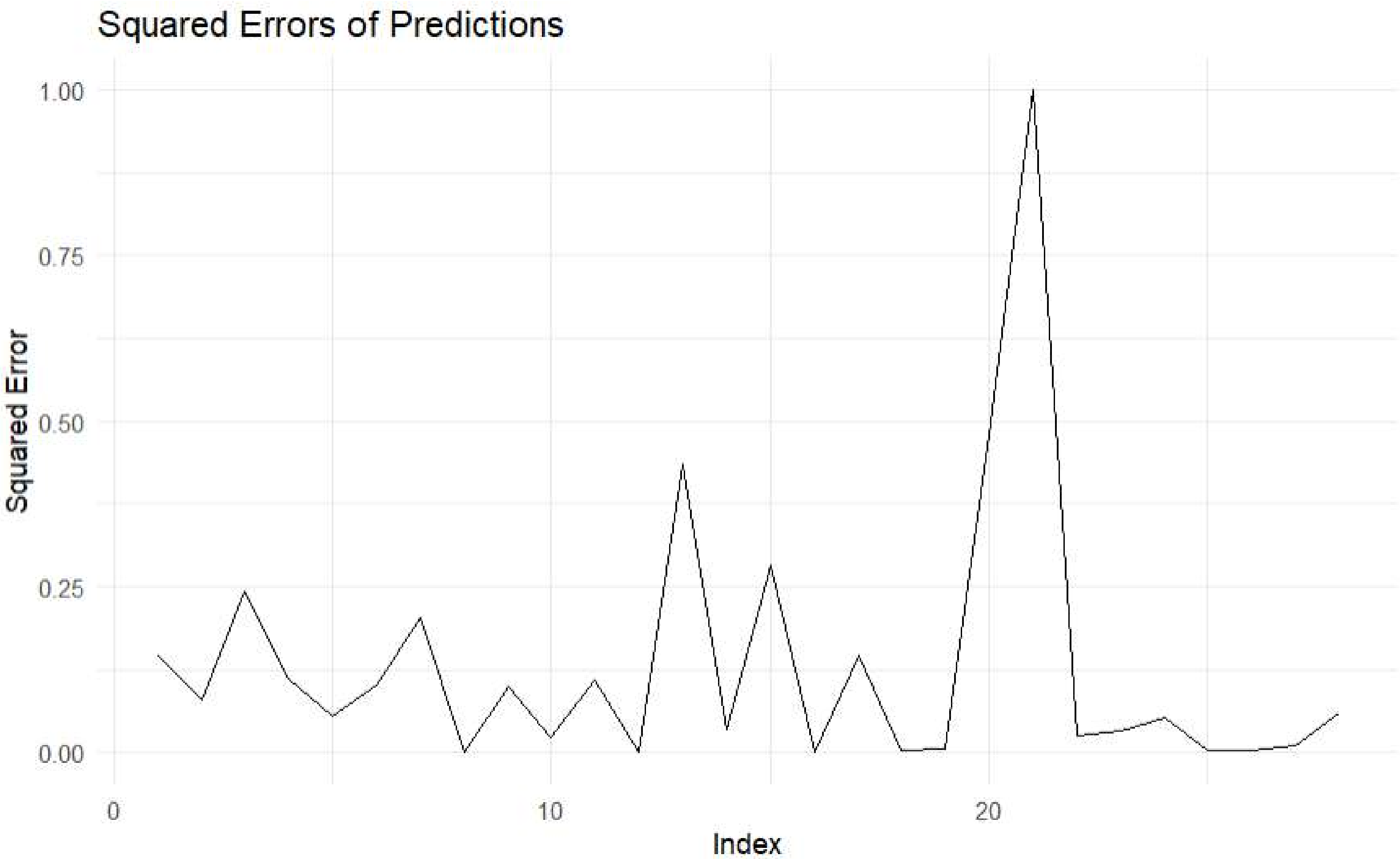

Line Plot of Errors
1.00
0.75
0.50
0.25
0.00
Error
May 02
May 09
May 16
May 23
Date
colour
SquaredErrors

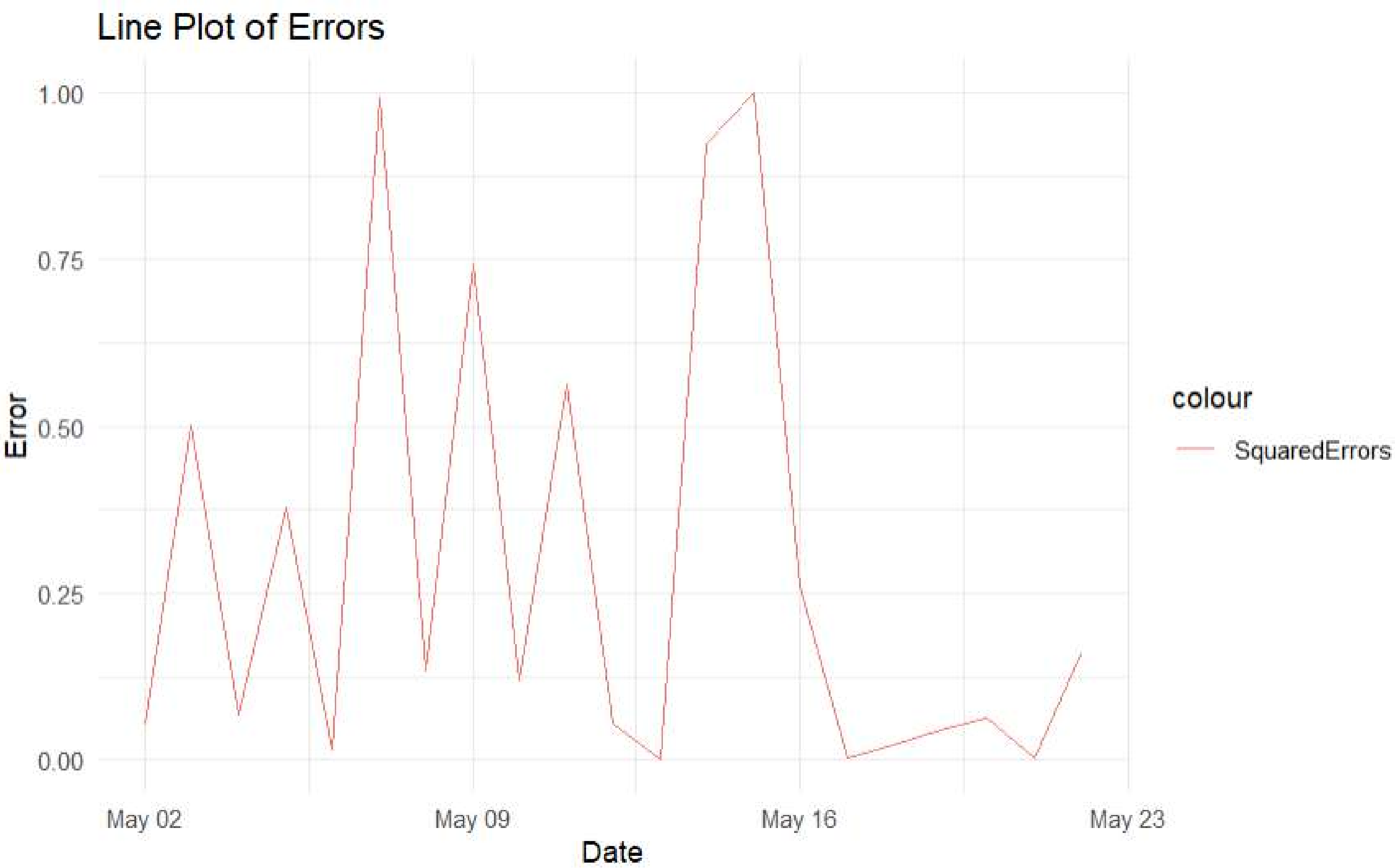

Predicted vs Actual Values
Actual Total Sales
Predicted Total Sales
15000
13000
11000
9000
10000
11000
12000
13000
14000


Squared Errors of Predictions
Squared Error
1.00
0.75
0.50
0.25
0.00
0
10
20
Index


Line Plot of Errors
1.00
0.75
0.50
0.25
0.00
Error
May 02
May 09
May 16
May 23
Date
colour
SquaredErrors


Predicted vs Actual Values
Actual Total Sales
Predicted Total Sales
17500
15000
12500
10000
12000
13000
14000
15000
16000


Squared Errors of Predictions
Squared Error
1.00
0.75
0.50
0.25
0.00
0
10
20
Index


Line Plot of Errors
1.00
0.75
0.50
0.25
0.00
Error
May 02
May 09
May 16
May 23
Date
colour
SquaredErrors