AMS 580 Project: Time Series

```
# Load packages and data
library(keras) # for deep learning
library(tidyverse) # general utility functions
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret) # machine learning utility functions
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ggplot2)
library(dplyr)
library(tensorflow)
```

```
##
## Attaching package: 'tensorflow'

## The following object is masked from 'package:caret':
##
##     train
```

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(forecast)
```

```r
# read data
traindata <- read_csv("train.csv")[, c("date", "sales")]
```

```
## Rows: 3000888 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): family
## dbl  (4): id, store_nbr, sales, onpromotion
## 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.
```

```r
testdata <- read_csv("test.csv")[, c("date", "onpromotion")]
```

```
## Rows: 28512 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): family
## dbl  (3): id, store_nbr, onpromotion
## 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.
```

```r
str(traindata)
```

```
## tibble [3,000,888 x 2] (S3: tbl_df/tbl/data.frame)
##  $ date : Date[1:3000888], format: "2013-01-01" "2013-01-01" ...
##  $ sales: num [1:3000888] 0 0 0 0 0 0 0 0 0 0 ...
```

```r
str(testdata)
```

```
## tibble [28,512 x 2] (S3: tbl_df/tbl/data.frame)
##  $ date       : Date[1:28512], format: "2017-08-16" "2017-08-16" ...
##  $ onpromotion: num [1:28512] 0 0 2 20 0 12 0 25 45 18 ...
```

```r
sapply(traindata, function(x) sum(is.na(x)))
```

```
##  date sales
##     0     0
```

```r
sapply(testdata, function(x) sum(is.na(x)))
```

```
##        date onpromotion
##           0           0
```

```r
# processing data
traindata = aggregate.data.frame(traindata$sales, by = list(date = traindata$date), FUN = sum)
colnames(traindata)[2] <- "sales"
traindata$ts_preds <- NA
traindata$rnn_preds <- traindata$sales
traindata$lstm_preds <- traindata$sales
traindata$gru_preds <- traindata$sales
testdata = aggregate.data.frame(testdata$onpromotion, by = list(date = testdata$date), FUN = sum)
testdata = subset(testdata, select = -c(x))
str(traindata)
```

```
## 'data.frame':    1684 obs. of  6 variables:
##  $ date      : Date, format: "2013-01-01" "2013-01-02" ...
##  $ sales     : num  2512 496092 361461 354460 477350 ...
##  $ ts_preds  : logi  NA NA NA NA NA NA ...
##  $ rnn_preds : num  2512 496092 361461 354460 477350 ...
##  $ lstm_preds: num  2512 496092 361461 354460 477350 ...
##  $ gru_preds : num  2512 496092 361461 354460 477350 ...
```

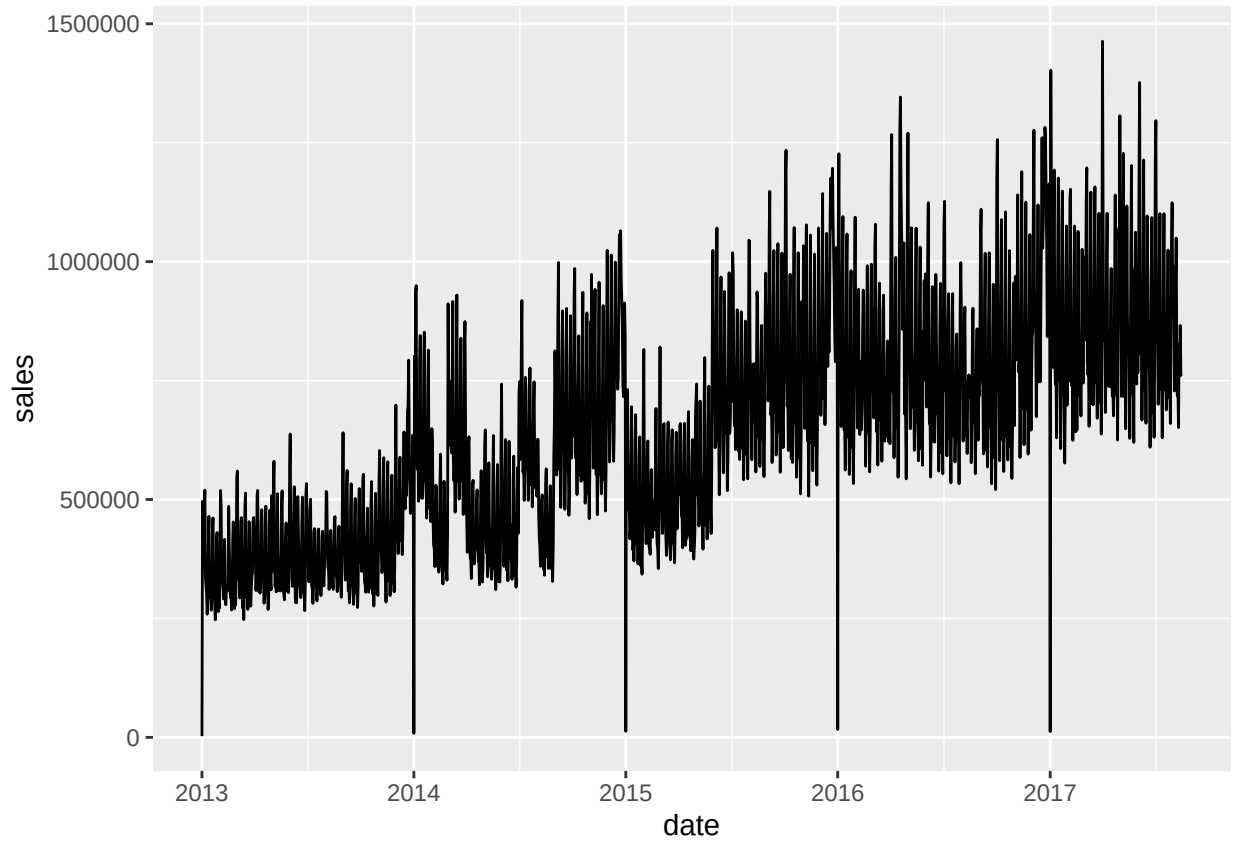
```r
str(testdata)
```

```
## 'data.frame':    16 obs. of  1 variable:
##  $ date: Date, format: "2017-08-16" "2017-08-17" ...
```

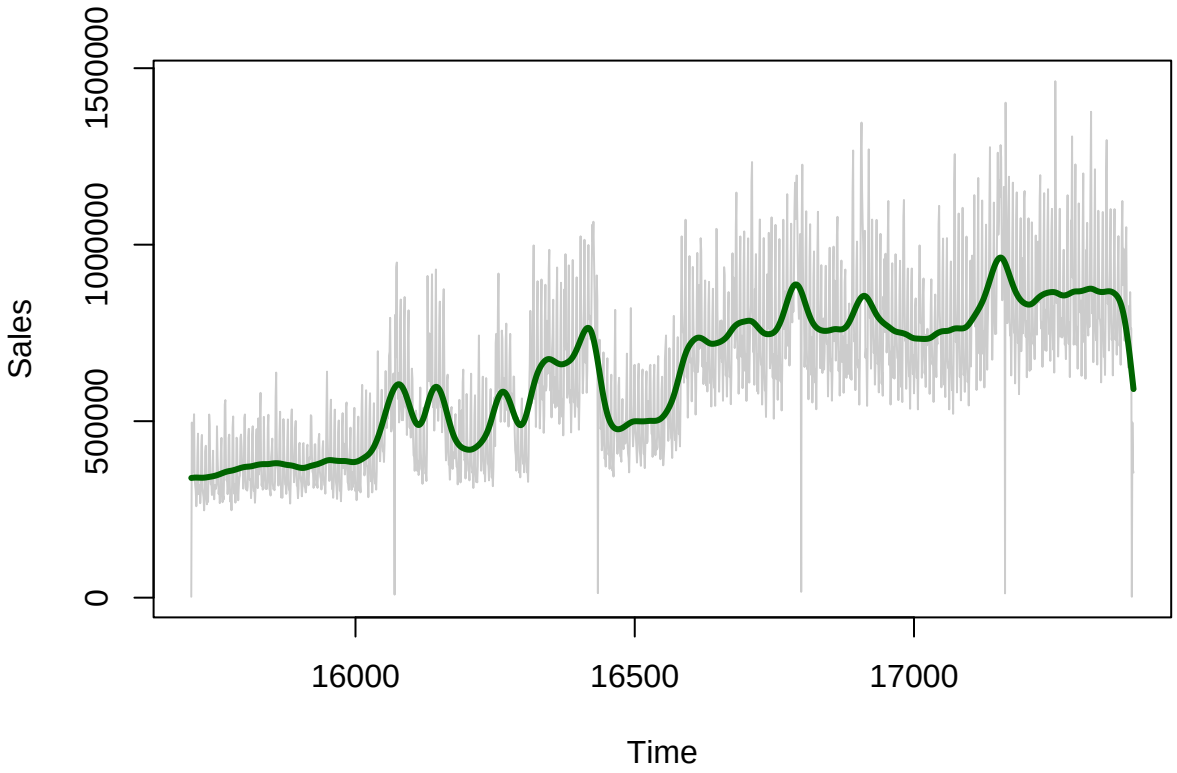
```r
# Visualization
knitr::kable(head(traindata))
```

date	sales	ts_preds	rnn_preds	lstm_preds	gru_preds
2013-01-01	2511.619	NA	2511.619	2511.619	2511.619
2013-01-02	496092.418	NA	496092.418	496092.418	496092.418
2013-01-03	361461.231	NA	361461.231	361461.231	361461.231
2013-01-04	354459.677	NA	354459.677	354459.677	354459.677
2013-01-05	477350.121	NA	477350.121	477350.121	477350.121
2013-01-06	519695.401	NA	519695.401	519695.401	519695.401

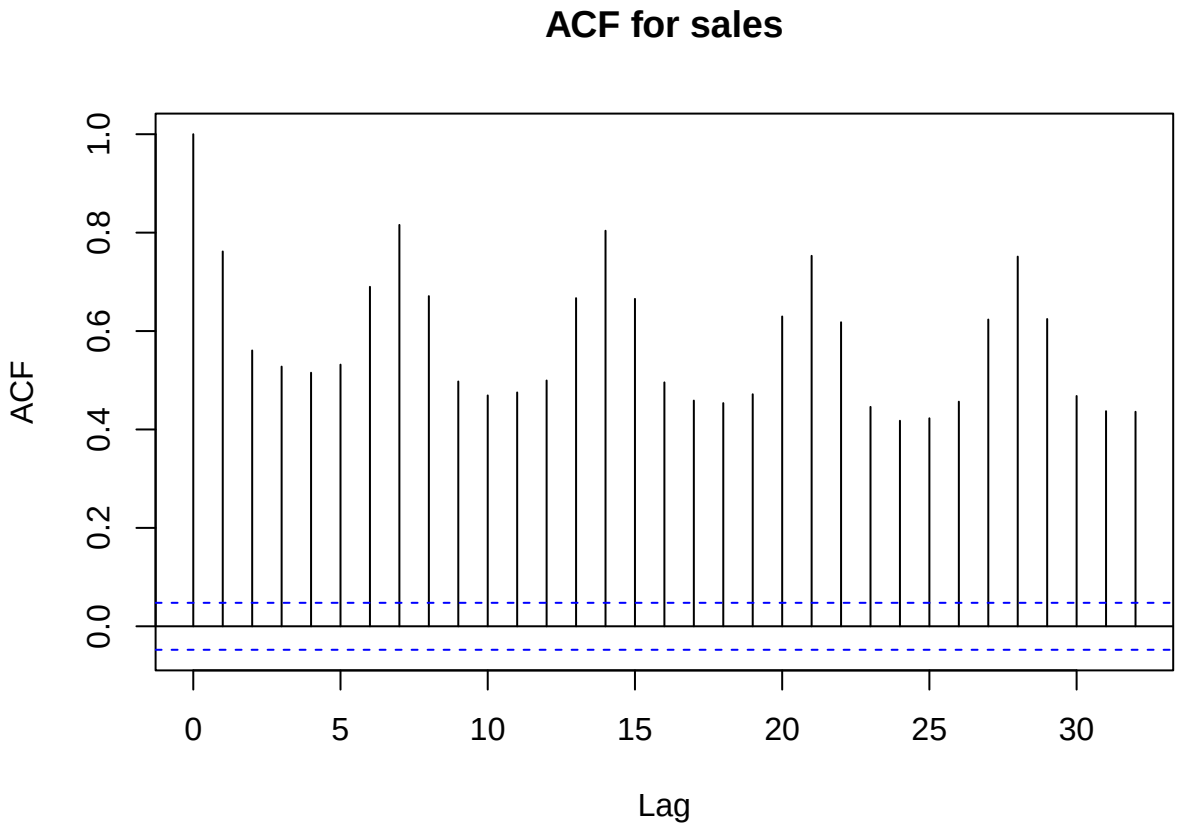
```r
ggplot(traindata, aes(x=date, y = sales)) + geom_line()
```



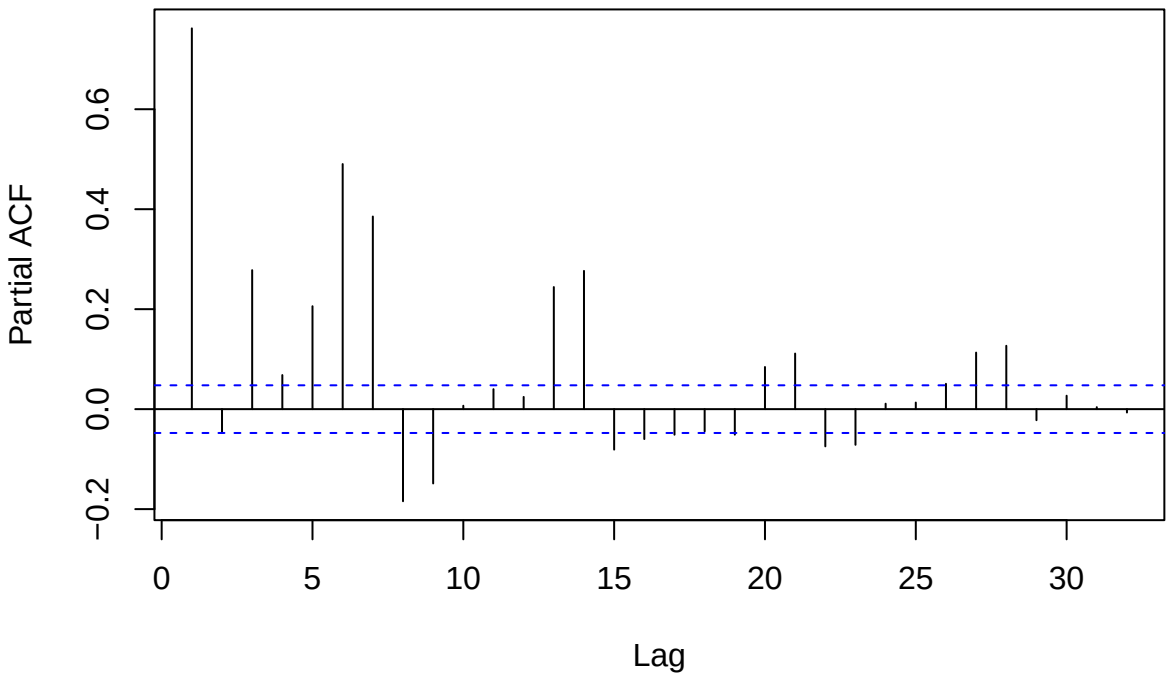

```
# ts object to analyze time series. ts() is often used for monthly, quarterly or yearly, but is still a
ots = ts(traindata[["sales"]], start=min(traindata[["date"]]),
         end=max(traindata[["date"]]), frequency=1)

# base R plotting with a smooth spline. Try if it can be plotted using ggplot2.
par(mar=c(4,4,4,2))  # margins: bottom, left, top, right
plot(
  ots, col=rgb(0,0,0,0.2),
  xlab="Time", ylab="Sales"
)
lines(
  predict(
    smooth.spline(ots~time(ots), spar=0.45)
  ),
  lwd=3, col="darkgreen"
)
```




```
par(mar=c(4,4,4,2))  # margins: bottom, left, top, right
acf(ots, main='ACF for sales')
```




```
pacf(ots, main='PACF for sales')
```




```
adf.test(ots)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ots
## Dickey-Fuller = -7.1184, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

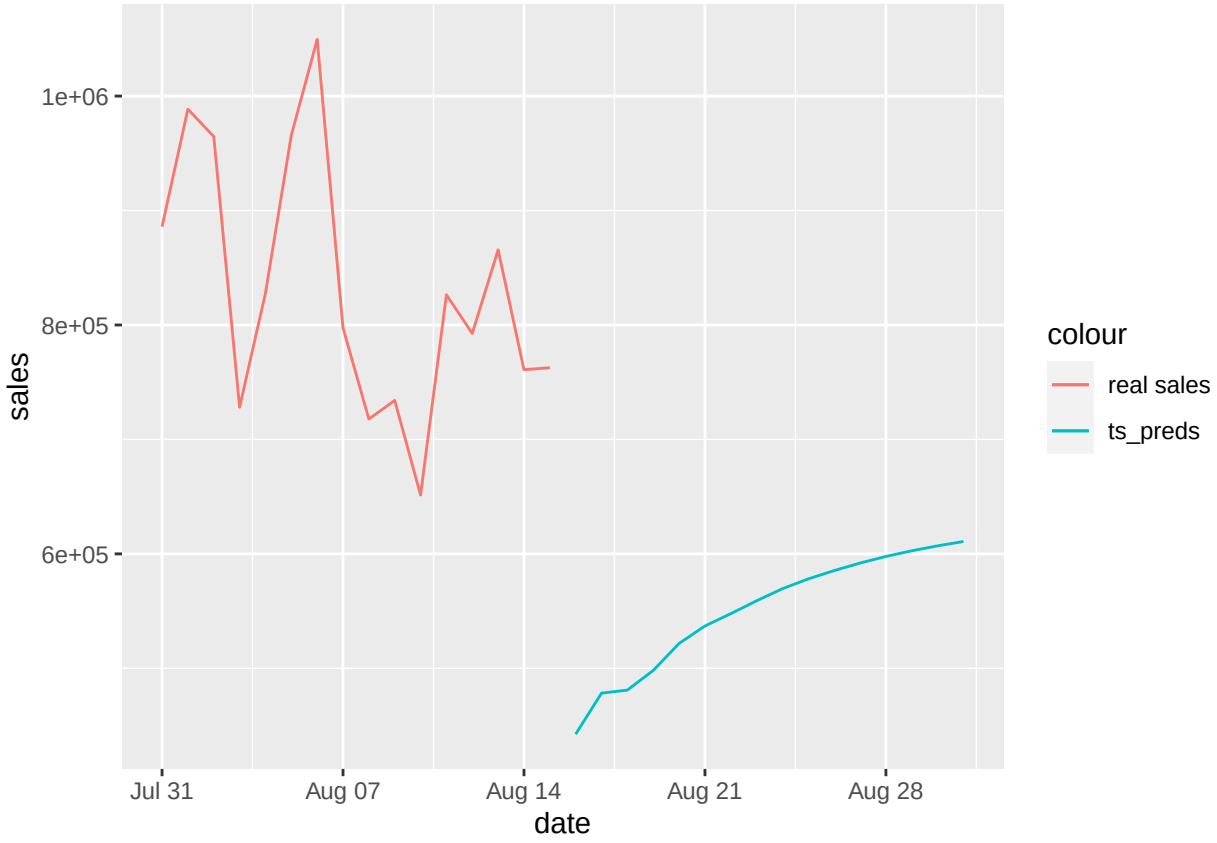
```
fit =
  auto.arima(
    ots,
    seasonal=FALSE,
    test="adf",
    ic="bic",  # I changed to BIC bcs it penalizes model complexity more than AIC and AICc
    lambda=NULL,
    stepwise=FALSE,
    approximation=FALSE,
    max.p=3
  )
summary(fit)
```

```
## Series: ots
```

```
## ARIMA(3,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3       mean
##       0.8200  -0.2808  0.2847  636705.12
## s.e.  0.0234   0.0301  0.0234   20034.97
##
## sigma^2 = 2.111e+10:  log likelihood = -22458.33
## AIC=44926.67   AICc=44926.7   BIC=44953.83
##
## Training set error measures:
##                     ME     RMSE      MAE     MPE     MAPE      MASE        ACF1
## Training set 53.55204 145135.6 111865.9 -46.128 59.52955 0.9571897 -0.02425064
```

```
# ts object of the future 16 days prediction
ts_prediction <- forecast(fit, 16)$mean

ts_data = traindata[c((nrow(traindata) - 15):nrow(traindata)), c(1,2,3)]
ts_data_preds = data.frame(date = testdata$date, sales = NA, ts_preds = ts_prediction)
ts_data <- rbind(ts_data, ts_data_preds)

# Plot
ggplot(data = ts_data, aes(x = date)) +
  geom_line(aes(y = sales, color = 'real sales')) +
  geom_line(aes(y = ts_preds, color = 'ts_preds'))
```

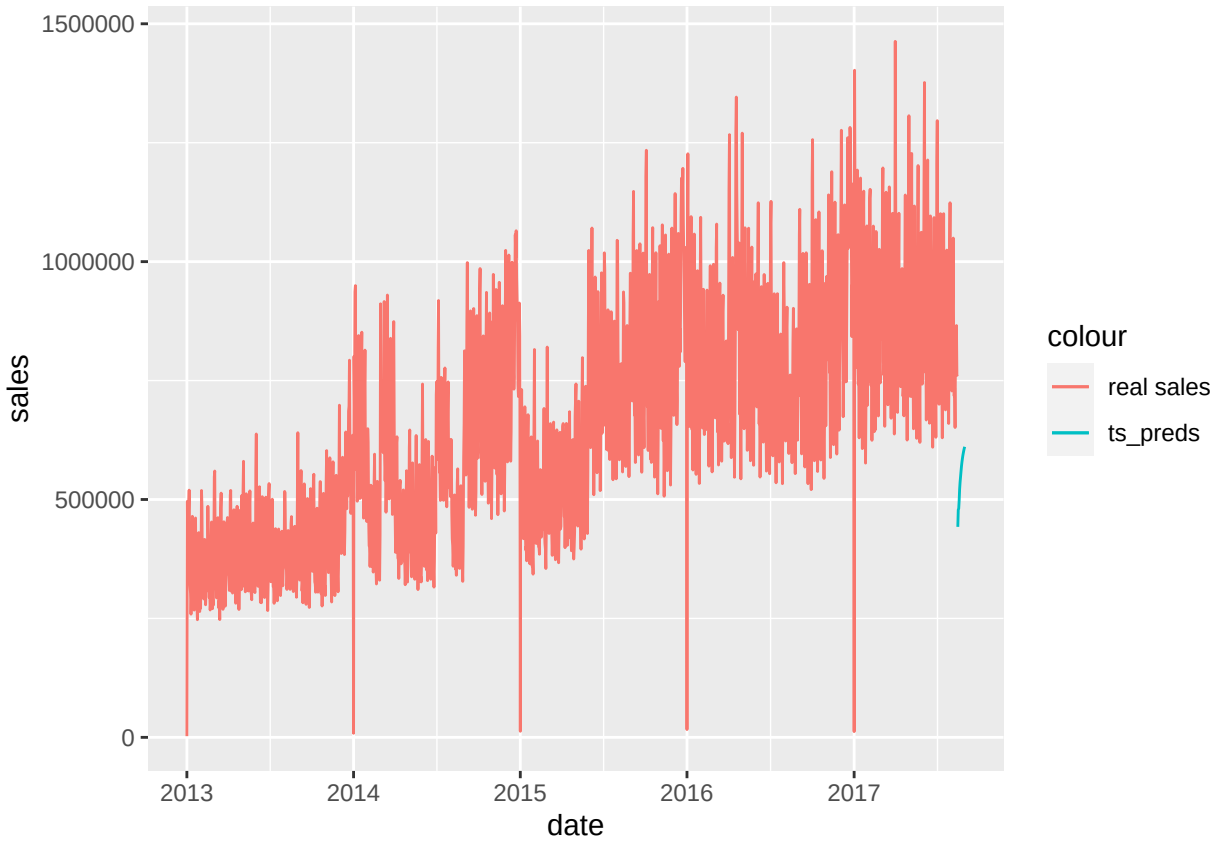
```
## Warning: Removed 16 row(s) containing missing values (geom_path).
## Removed 16 row(s) containing missing values (geom_path).
```




```
# Plot all data
ts_totaldata <- rbind(traindata[,c(1,2,3)], ts_data_preds)
ggplot(data = ts_totaldata, aes(x = date)) +
  geom_line(aes(y = sales, color = 'real sales')) +
  geom_line(aes(y = ts_preds, color = 'ts_preds'))
```

```
## Warning: Removed 16 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 1684 row(s) containing missing values (geom_path).
```




```
# Normalization
meansale = mean(traindata$sales)
sdsale = sd(traindata$sales)
traindata$sales_norm = scale(traindata$sales)
model_data = matrix(traindata$sales_norm)
knitr::kable(tail(model_data,5))
```

[1680,]	0.8054995
[1681,]	0.6615502
[1682,]	0.9730091
[1683,]	0.5262826
[1684,]	0.5337034

```
# Split traindata to train and test
# traindata time is from 2013-1-1 to 2017-8-15
# testdata time is from 2017-8-16 to 2017-8-31
# Each has 54 * 33 = 1782 samples
test_size = 2 * nrow(testdata)
train_data = head(model_data,-test_size)
test_data = tail(model_data, test_size)
cat(dim(train_data)[1], 'days are divided into the training set and', dim(test_data)[1], 'days are divi
```

```
## 1652 days are divided into the training set and 32 days are divided into the testing set.
```

```
prediction = nrow(testdata)
lag = prediction
# Training X
train_X = t(sapply(
    1:(length(train_data) - lag - prediction + 1),
    function(x) train_data[x:(x + lag - 1), 1]
  ))
# now we transform it into 3D form
train_X <- array(
    data = as.numeric(unlist(train_X)),
    dim = c(
        nrow(train_X),
        lag,
        1
    )
)
# Training y
train_y <- t(sapply(
    (1 + lag):(length(train_data) - prediction + 1),
    function(x) train_data[x:(x + prediction - 1)]
))
train_y <- array(
    data = as.numeric(unlist(train_y)),
    dim = c(
        nrow(train_y),
        prediction,
        1
    )
)
# Testing X
test_X = t(sapply(
    1:(length(test_data) - lag - prediction + 1),
    function(x) test_data[x:(x + lag - 1), 1]
  ))
test_X <- array(
    data = as.numeric(unlist(test_X)),
    dim = c(
        nrow(test_X),
        lag,
        1
    )
)
# Testing y
test_y <- t(sapply(
    (1 + lag):(length(test_data) - prediction + 1),
    function(x) test_data[x:(x + prediction - 1)]
))
test_y <- array(
    data = as.numeric(unlist(test_y)),
    dim = c(
        nrow(test_y),
        prediction,
        1
```

```
    )
)
dim(train_X)
```

```
## [1] 1621   16    1
```

```
dim(train_y)
```

```
## [1] 1621   16    1
```

```
dim(test_X)
```

```
## [1]  1 16  1
```

```
dim(test_y)
```

```
## [1]  1 16  1
```

RNN – Recurrent Neural Network

```
set_random_seed(123)
```

```
## Loaded Tensorflow version 2.8.0
```

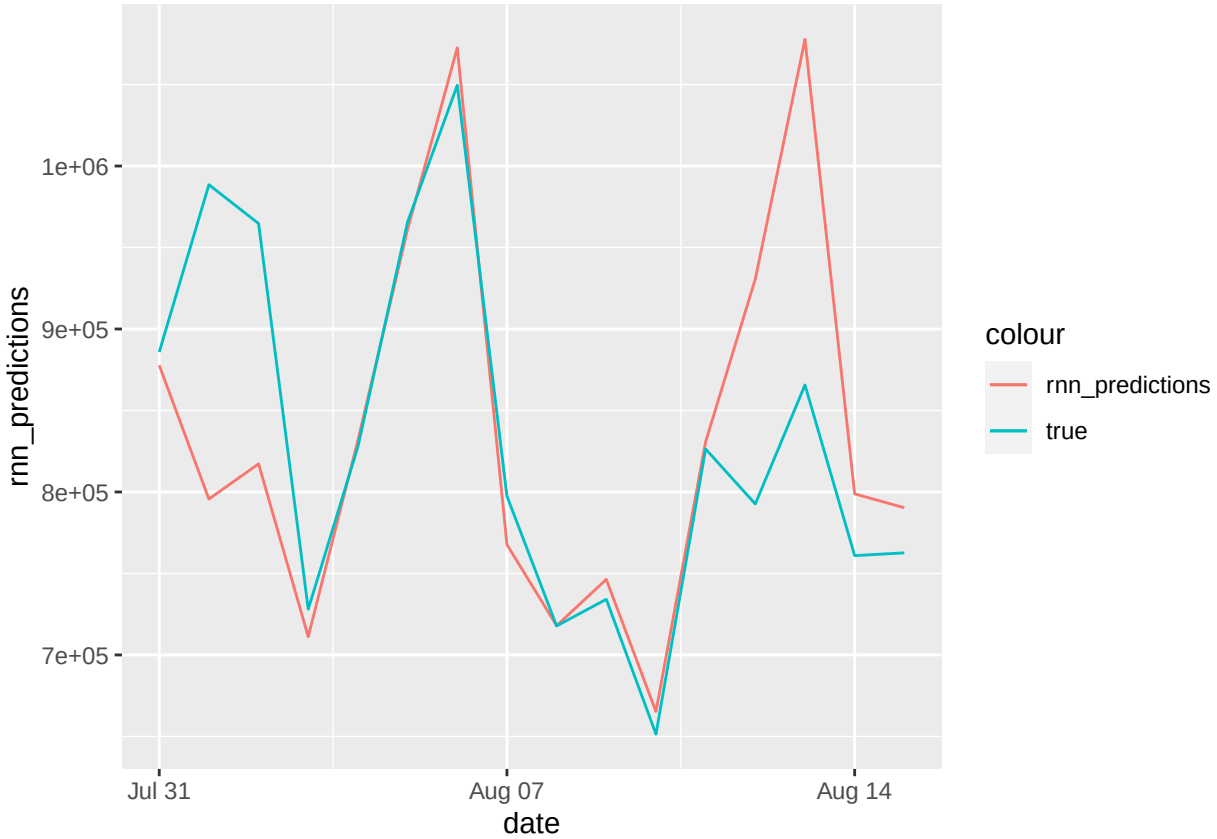
```
rnn_model <- keras_model_sequential()
rnn_model %>%
  layer_simple_rnn(units = 200, input_shape = dim(train_X)[2:3])
rnn_model %>%
  layer_dense(units = dim(test_y)[2])

summary(rnn_model)
```

```
## Model: "sequential"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  simple_rnn (SimpleRNN)             (None, 200)                     40400
##
##  dense (Dense)                      (None, 16)                      3216
##
## =========================================================================
## Total params: 43,616
## Trainable params: 43,616
## Non-trainable params: 0
## _________________________________________________________________________
```

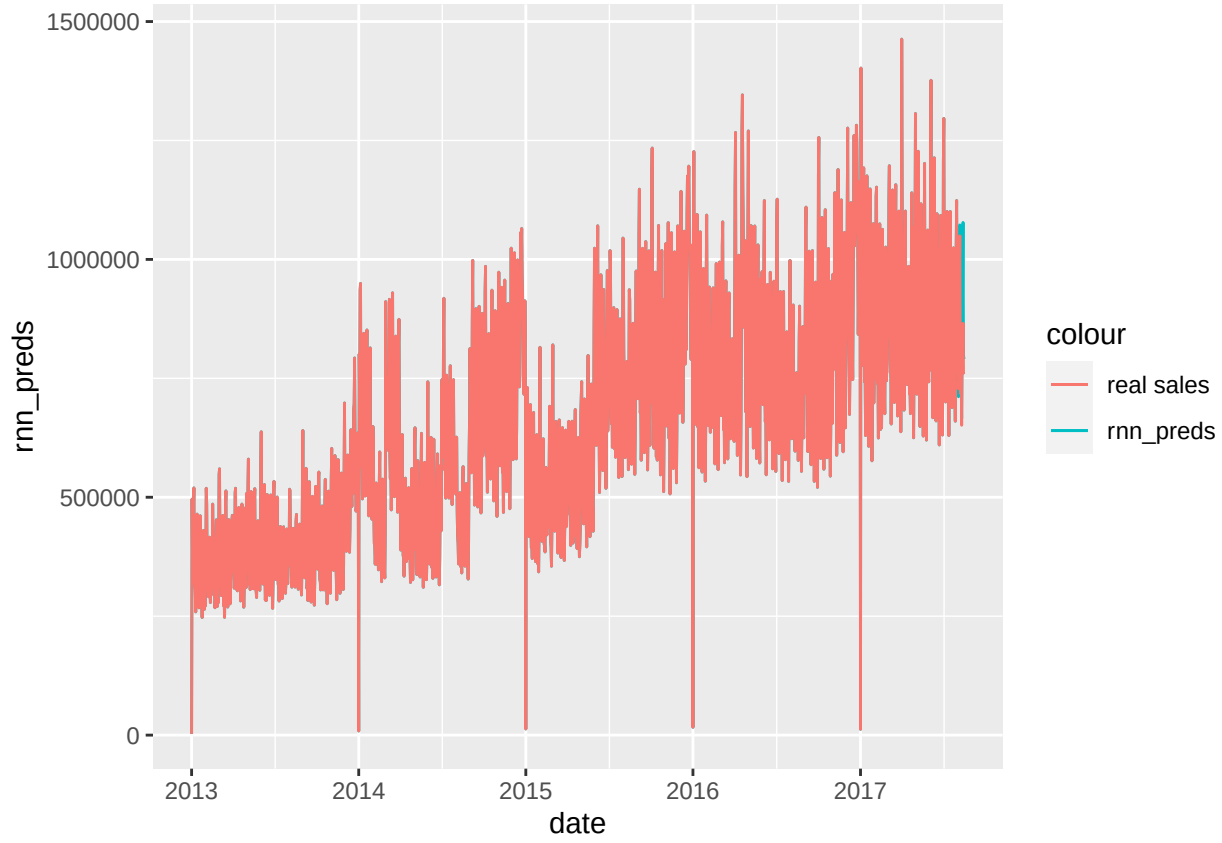
```
rnn_model %>% compile(loss = 'mse',
                      optimizer = 'adam',
                      metrics = c('mse'))
rnn_history <- rnn_model %>% fit(
  x = train_X,
  y = train_y,
  batch_size =16,
  epochs = 50,
  validation_split = 0.1,
  shuffle = FALSE
)

rnn_preds_norm = t(predict(rnn_model, test_X))
rnn_preds_complete = cbind(rnn_preds_norm, tail(traindata, prediction))
rnn_preds = rnn_preds_complete$rnn_preds_norm * sdsale + meansale
rnn_predictions = data.frame(rnn_predictions = rnn_preds, true = rnn_preds_complete$sales, date = rnn_p
# Test RMSE
(rnn_RMSE = RMSE(rnn_predictions$true, rnn_predictions$rnn_predictions))
```

```
## [1] 89213.54
```

```
# Plot
ggplot(data = rnn_predictions, aes(x = date)) +
  geom_line(aes(y = rnn_predictions, color = 'rnn_predictions')) +
  geom_line(aes(y = true, color = 'true'))
```



```
# Plot All data
traindata[c((nrow(traindata) - nrow(testdata) + 1):nrow(traindata)), c(4)] <- rnn_preds
ggplot(data = traindata, aes(x = date)) +
  geom_line(aes(y = rnn_preds, color = 'rnn_preds')) +
  geom_line(aes(y = sales, color = 'real sales'))
```



LSTM – Long-Short Term Memory

```
set_random_seed(123)
lstm_model <- keras_model_sequential()
lstm_model %>%
  layer_lstm(units = 200, input_shape = dim(train_X)[2:3])
lstm_model %>%
  layer_dense(units = dim(test_y)[2])

summary(lstm_model)
```

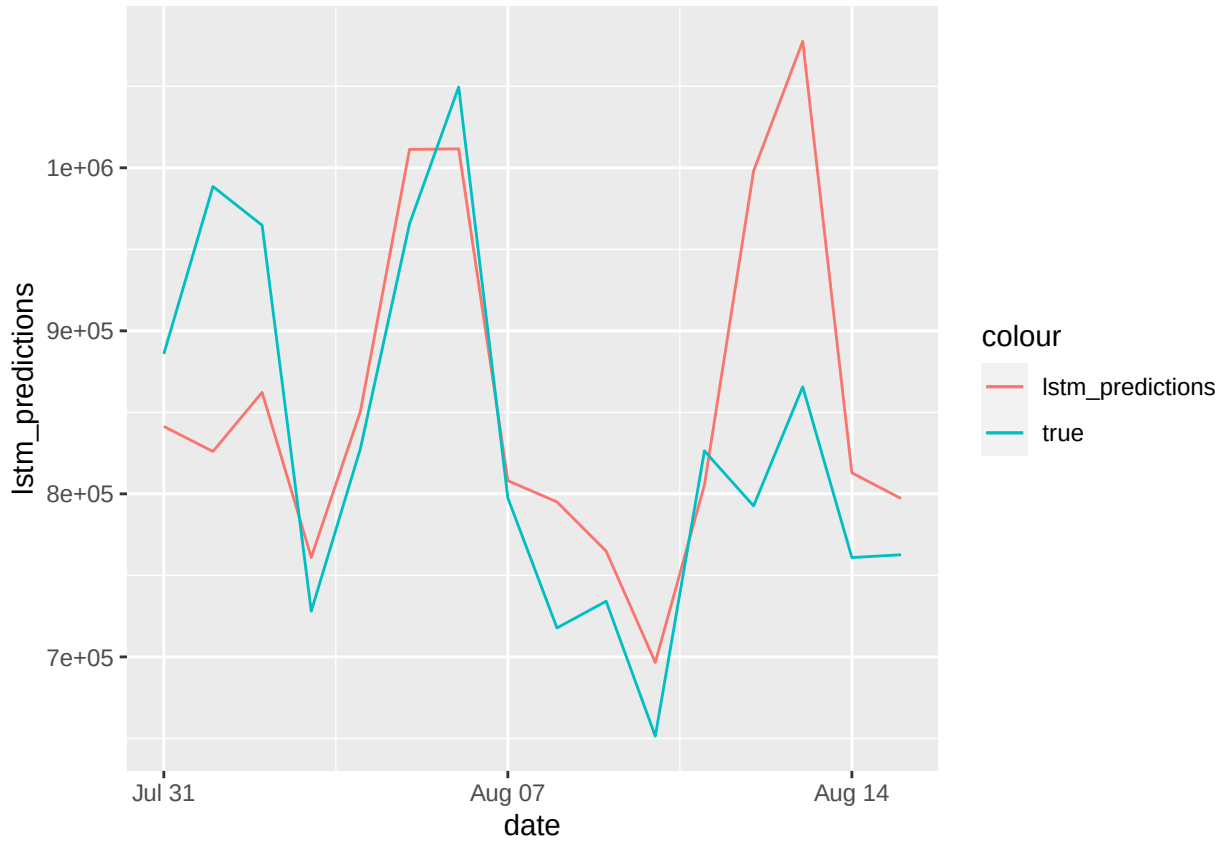
```
## Model: "sequential_1"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  lstm (LSTM)                        (None, 200)                     161600
##
##  dense_1 (Dense)                    (None, 16)                      3216
##
```

```
## ================================================================================
## Total params: 164,816
## Trainable params: 164,816
## Non-trainable params: 0
## _________________________________________________________________________________
```

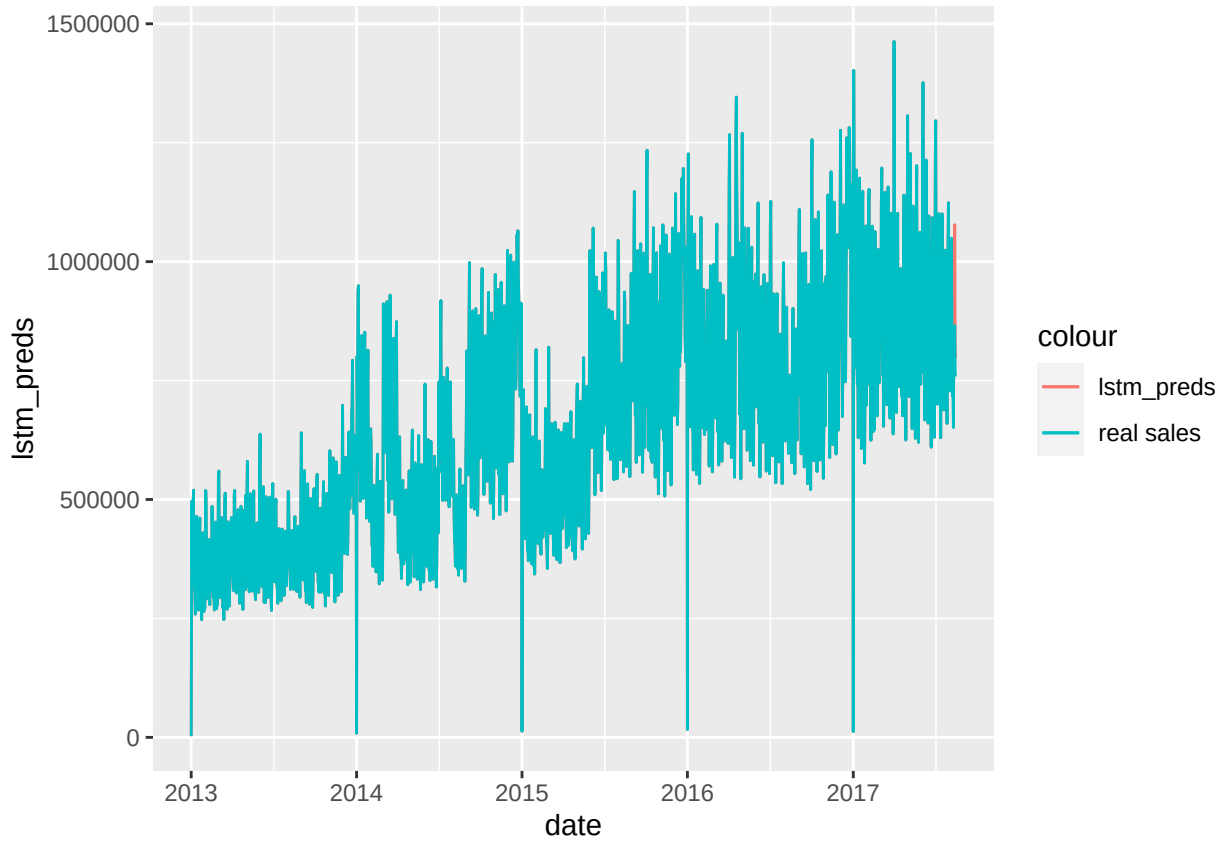
```
lstm_model %>% compile(loss = 'mse',
                  optimizer = 'adam',
                  metrics = c('mse'))
lstm_history <- lstm_model %>% fit(
  x = train_X,
  y = train_y,
  batch_size =16,
  epochs = 50,
  validation_split = 0.1,
  shuffle = FALSE
)

lstm_preds_norm = t(predict(lstm_model, test_X))
lstm_preds_complete = cbind(lstm_preds_norm, tail(traindata, prediction))
lstm_preds = lstm_preds_complete$lstm_preds_norm * sdsale + meansale
lstm_predictions = data.frame(lstm_predictions = lstm_preds, true = lstm_preds_complete$sales, date = ls
# Test RMSE
(lstm_RMSE = RMSE(lstm_predictions$true, lstm_predictions$lstm_predictions))
```

```
## [1] 95040.73
```

```
# Plot
ggplot(data = lstm_predictions, aes(x = date)) +
  geom_line(aes(y = lstm_predictions, color = 'lstm_predictions')) +
  geom_line(aes(y = true, color = 'true'))
```




```
# Plot All data
traindata[c((nrow(traindata) - nrow(testdata) + 1):nrow(traindata)), c(5)] <- lstm_preds
ggplot(data = traindata, aes(x = date)) +
  geom_line(aes(y = lstm_preds, color = 'lstm_preds')) +
  geom_line(aes(y = sales, color = 'real sales'))
```



GRU – Gated Recurrent Units

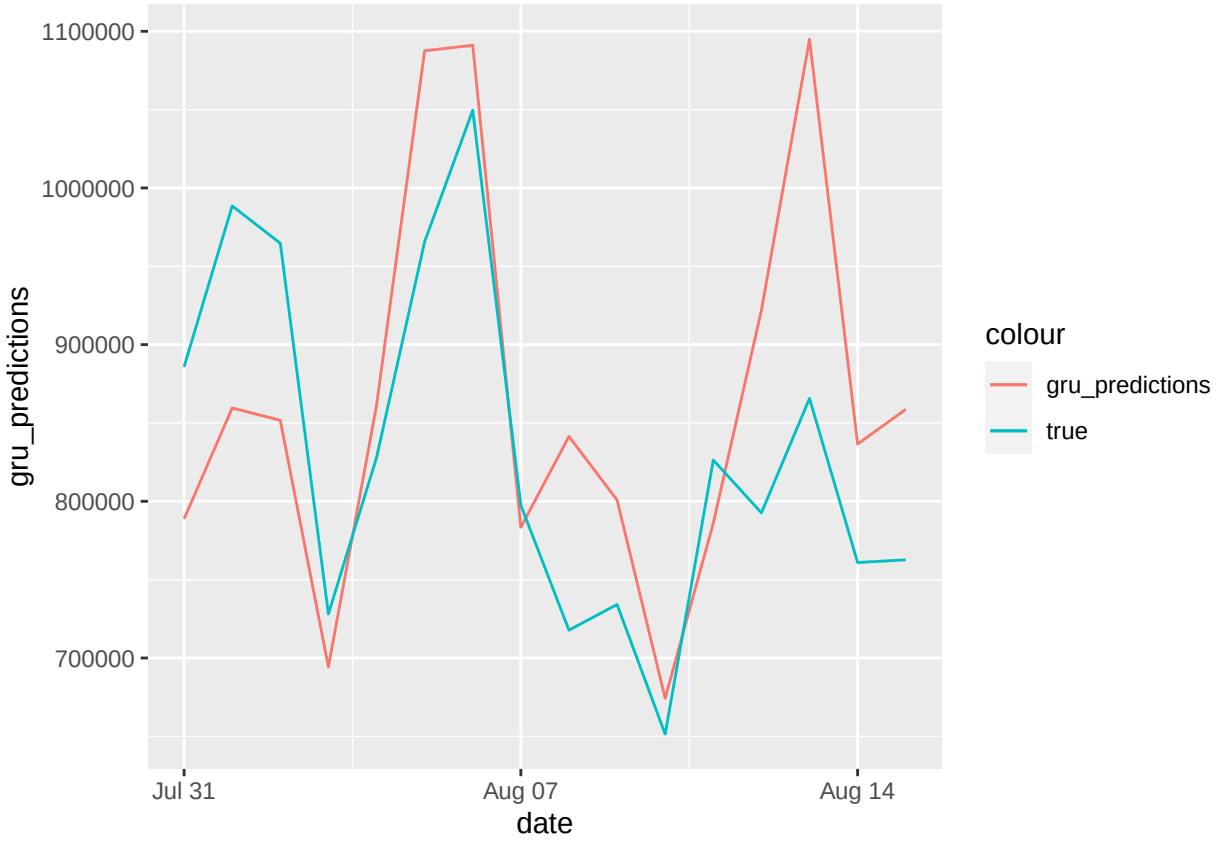
```
set_random_seed(123)
gru_model <- keras_model_sequential()
gru_model %>%
  layer_gru(units = 200, input_shape = dim(train_X)[2:3])
gru_model %>%
  layer_dense(units = dim(test_y)[2])

summary(gru_model)
```

```
## Model: "sequential_2"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  gru (GRU)                          (None, 200)                     121800
##
##  dense_2 (Dense)                    (None, 16)                      3216
##
## =========================================================================
## Total params: 125,016
## Trainable params: 125,016
## Non-trainable params: 0
## _________________________________________________________________________
```

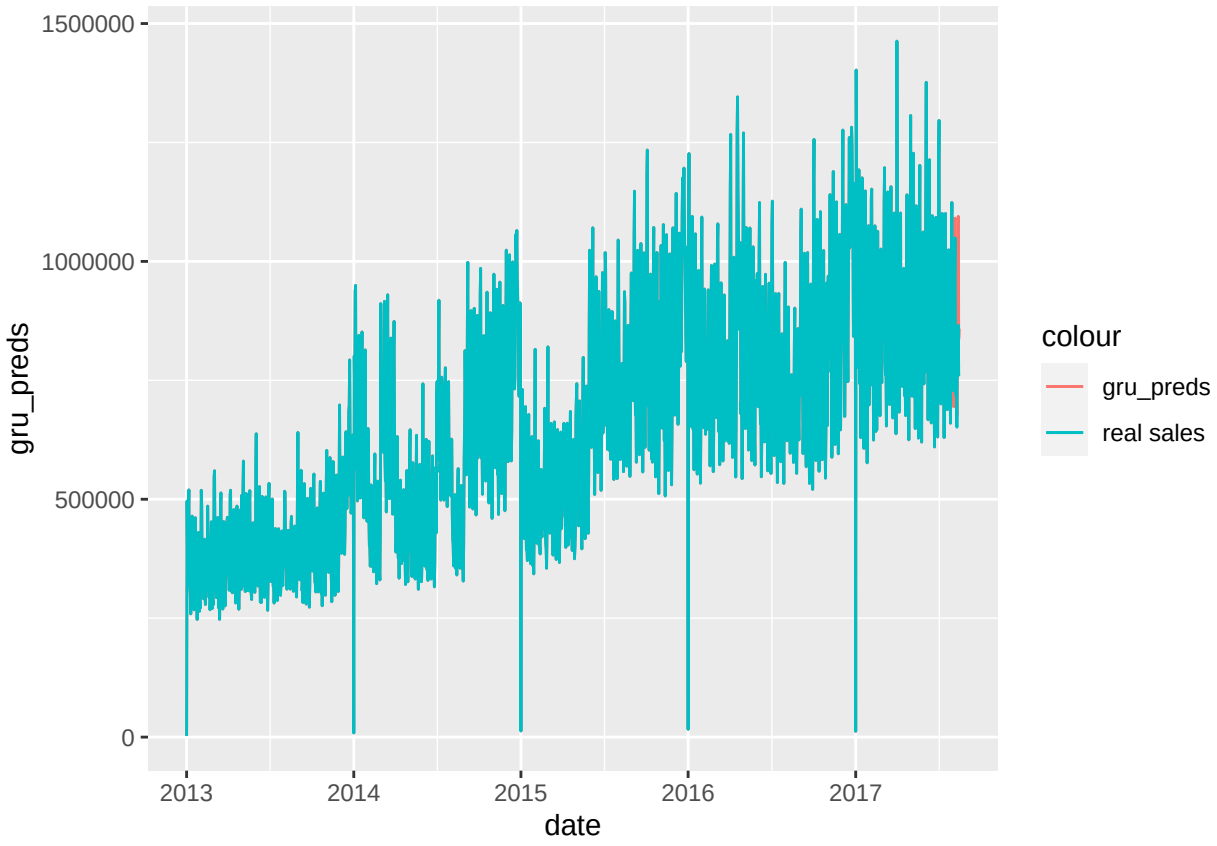
```
gru_model %>% compile(loss = 'mse',
                      optimizer = 'adam',
                      metrics = c('mse'))
gru_history <- gru_model %>% fit(
  x = train_X,
  y = train_y,
  batch_size =16,
  epochs = 50,
  validation_split = 0.1,
  shuffle = FALSE
)

gru_preds_norm = t(predict(gru_model, test_X))
gru_preds_complete = cbind(gru_preds_norm, tail(traindata, prediction))
gru_preds = gru_preds_complete$gru_preds_norm * sdsale + meansale
gru_predictions = data.frame(gru_predictions = gru_preds, true = gru_preds_complete$sales, date = gru_pr
# Test RMSE
(gru_RMSE = RMSE(gru_predictions$true, gru_predictions$gru_predictions))
```

```
## [1] 101219.8
```

```
# Plot
ggplot(data = gru_predictions, aes(x = date)) +
  geom_line(aes(y = gru_predictions, color = 'gru_predictions')) +
  geom_line(aes(y = true, color = 'true'))
```



```
# Plot All data
traindata[c((nrow(traindata) - nrow(testdata) + 1):nrow(traindata)), c(6)] <- gru_preds
ggplot(data = traindata, aes(x = date)) +
  geom_line(aes(y = gru_preds, color = 'gru_preds')) +
  geom_line(aes(y = sales, color = 'real sales'))
```



Compare Models

```
RMSEs <- data.frame(rnn_RMSE,lstm_RMSE, gru_RMSE)
knitr::kable(head(RMSEs))
```

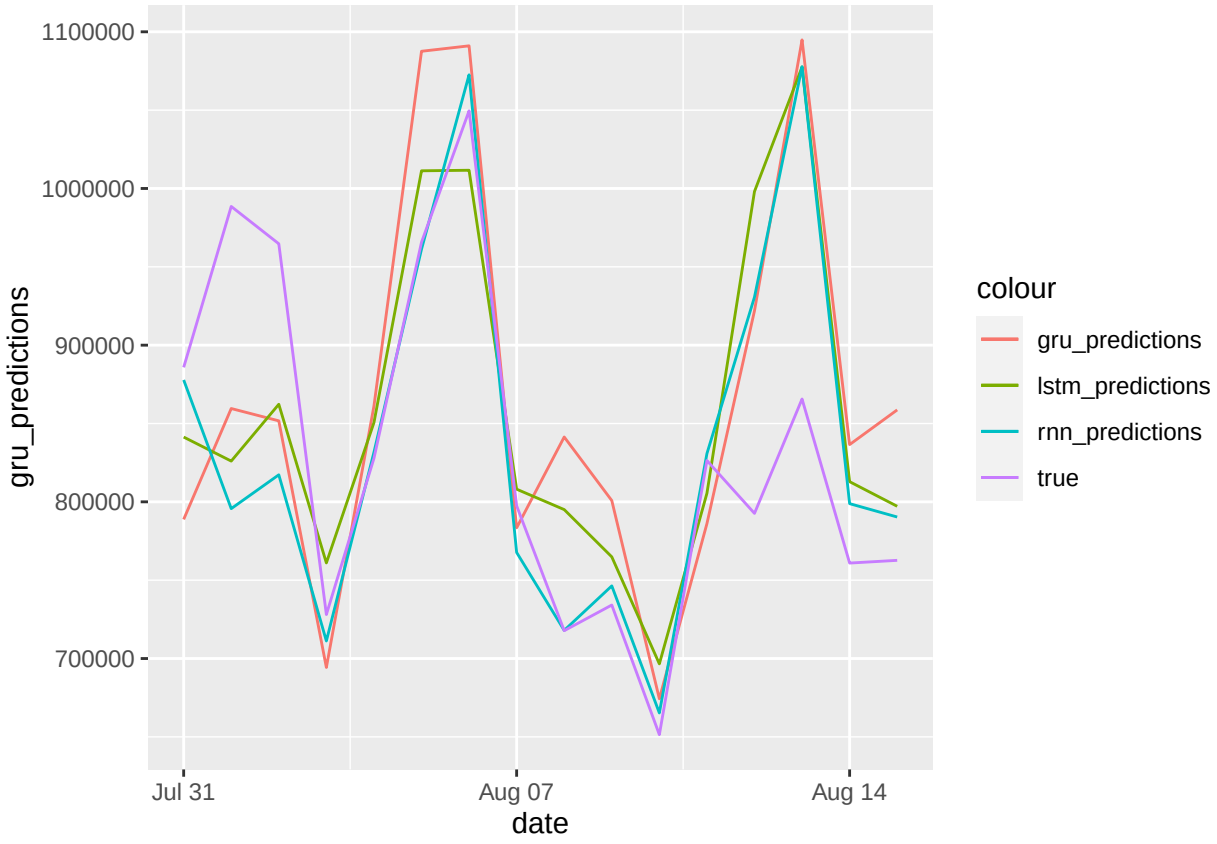
rnn_RMSE	lstm_RMSE	gru_RMSE
89213.54	95040.73	101219.8

```
cat('The minimum RMSE is', min(RMSEs), 'from RNN model.')
```

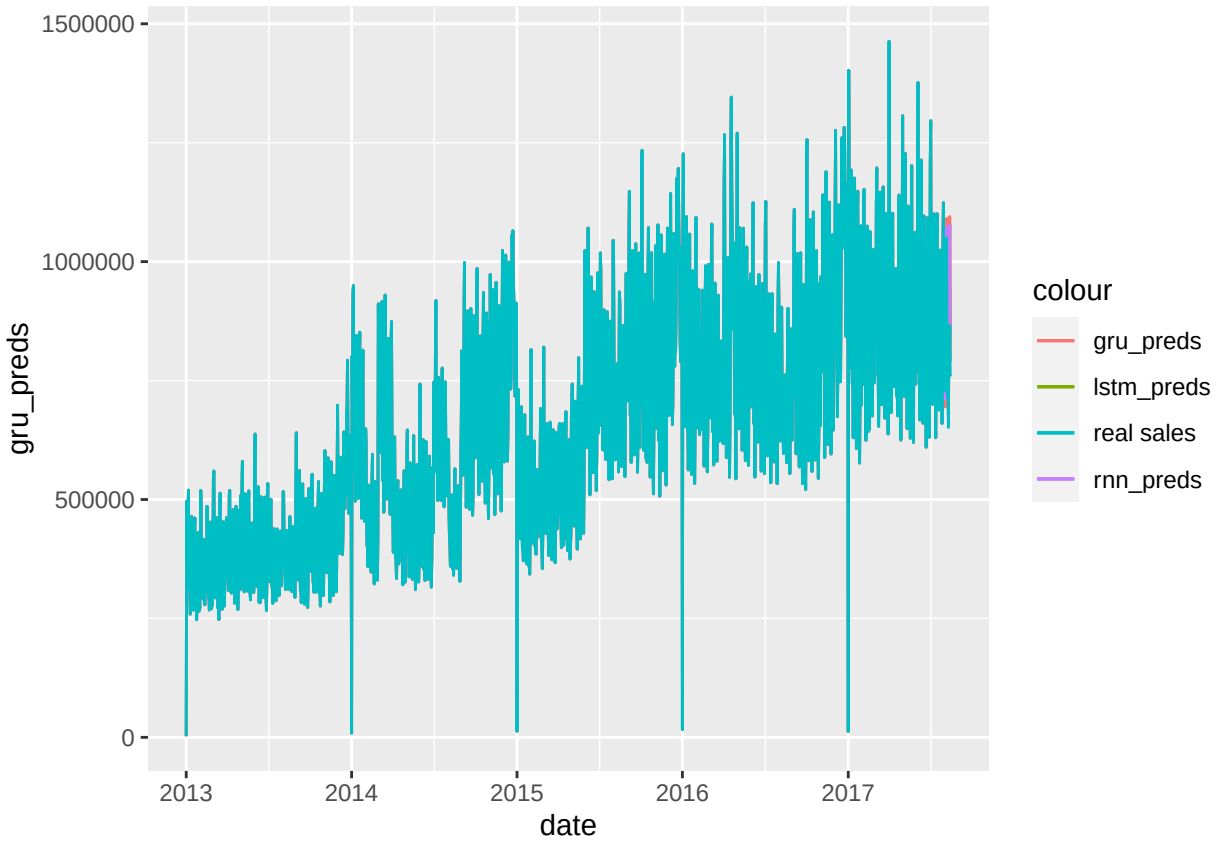
```
## The minimum RMSE is 89213.54 from RNN model.
```

```
# Plot
gru_predictions$lstm_predictions <- lstm_predictions$lstm_predictions
gru_predictions$rnn_predictions <- rnn_predictions$rnn_predictions
ggplot(data = gru_predictions, aes(x = date)) +
  geom_line(aes(y = gru_predictions, color = 'gru_predictions')) +
```

```
  geom_line(aes(y = lstm_predictions, color = 'lstm_predictions')) +
  geom_line(aes(y = rnn_predictions, color = 'rnn_predictions')) +
  geom_line(aes(y = true, color = 'true'))
```




```
# Plot All data
ggplot(data = traindata, aes(x = date)) +
  geom_line(aes(y = gru_preds, color = 'gru_preds')) +
  geom_line(aes(y = lstm_preds, color = 'lstm_preds')) +
  geom_line(aes(y = rnn_preds, color = 'rnn_preds')) +
  geom_line(aes(y = sales, color = 'real sales'))
```



Predict Test Data

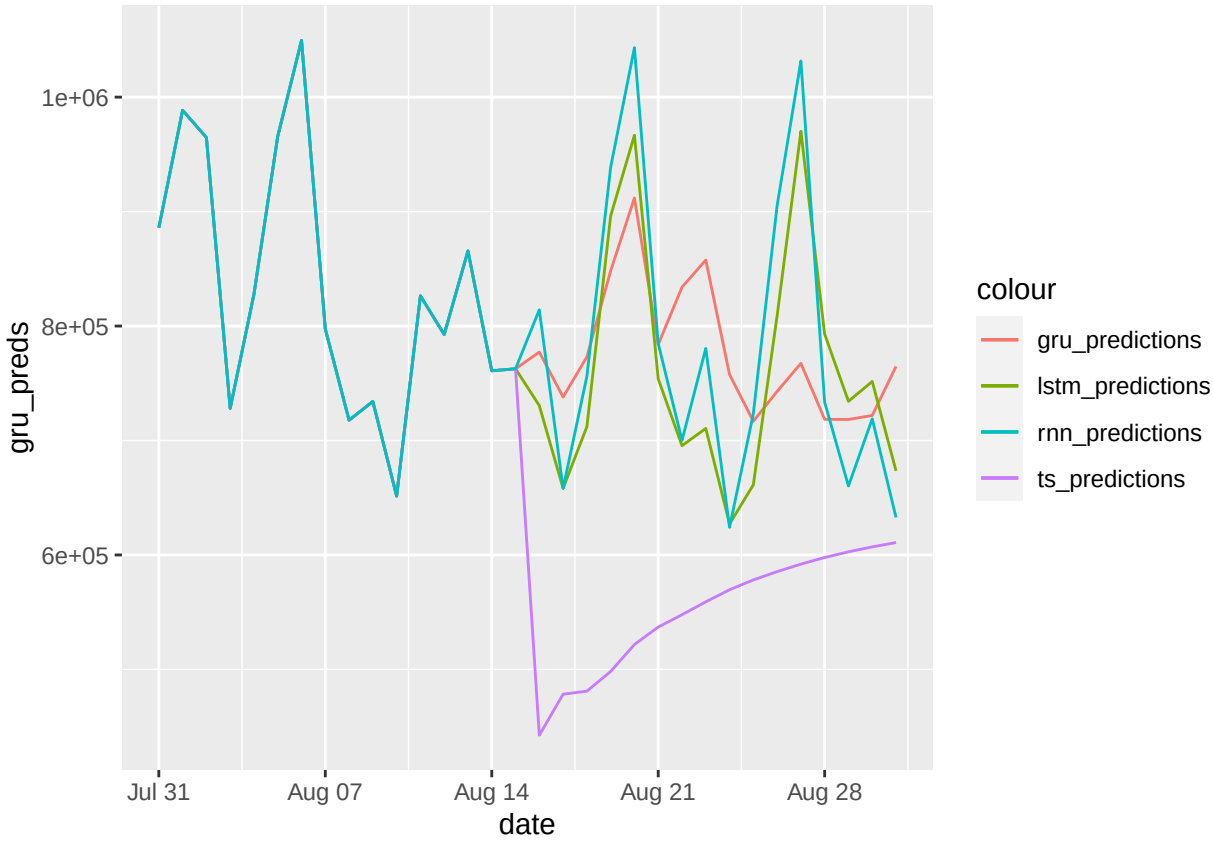
```
# Test data
testdata$sales <- NA
testdata$sales_norm <- NA
testdata$rnn_preds <- NA
testdata$lstm_preds <- NA
testdata$gru_preds <- NA
testdata$ts_preds <- NA
testdata <- rbind(traindata[c((nrow(traindata) - nrow(testdata) + 1):nrow(traindata)), ], testdata)
testdata$rnn_preds = testdata$lstm_preds = testdata$gru_preds = testdata$ts_preds = testdata$sales
model_data_x = matrix(traindata[c((nrow(traindata) - 16):nrow(traindata)), c(7)][-1])
# Test data X
test_data_X = t(sapply(
    1:(32 - lag - prediction + 1),
    function(x) model_data_x[x:(x + lag - 1), 1]
  ))
test_data_X <- array(
    data = as.numeric(unlist(test_data_X)),
    dim = c(
        nrow(test_data_X),
        lag,
        1
    )
)
dim(test_data_X)
```

```
## [1]  1 16  1
```

```
rnn_predstest_norm = t(predict(rnn_model, test_data_X))
rnn_predstest = rnn_predstest_norm * sdsale + meansale
lstm_predstest_norm = t(predict(lstm_model, test_data_X))
lstm_predstest = lstm_predstest_norm * sdsale + meansale
gru_predstest_norm = t(predict(gru_model, test_data_X))
gru_predstest = gru_predstest_norm * sdsale + meansale
testdata$rnn_preds[17:32] <- rnn_predstest
testdata$lstm_preds[17:32] <- lstm_predstest
testdata$gru_preds[17:32] <- gru_predstest
testdata$ts_preds[17:32] <- ts_data_preds$ts_preds
```

Compare the predictions and Visualization

```
ggplot(data = testdata, aes(x = date)) +
  geom_line(aes(y = gru_preds, color = 'gru_predictions')) +
  geom_line(aes(y = lstm_preds, color = 'lstm_predictions')) +
  geom_line(aes(y = ts_preds, color = 'ts_predictions')) +
  geom_line(aes(y = rnn_preds, color = 'rnn_predictions'))
```




```
totaldata <- subset(traindata, select = c(1))
totaldata$rnn_preds = totaldata$lstm_preds = totaldata$gru_preds = totaldata$ts_preds = traindata$sales
testdata <-subset(testdata, select = c("date","rnn_preds","lstm_preds","gru_preds","ts_preds"))
testdata <- testdata[c(17:32),]
```

```
totaldata <- rbind(totaldata, testdata)
ggplot(data = totaldata, aes(x = date)) +
  geom_line(aes(y = gru_preds, color = 'gru_predictions')) +
  geom_line(aes(y = lstm_preds, color = 'lstm_predictions')) +
  geom_line(aes(y = ts_preds, color = 'ts_predictions')) +
  geom_line(aes(y = rnn_preds, color = 'rnn_predictions'))
```