Quiz9_solutions

```
# install.packages(c("keras", "tensorflow"))
# install.packages("devtools")
# library(devtools)
#
# devtools::install_github("rstudio/keras", dependencies = TRUE)
# devtools::install_github("rstudio/tensorflow", dependencies = TRUE)

# install_keras()
# install_tensorflow()
library(keras)
library(tensorflow)
tinytex::install_tinytex()
```

```
## The directory /usr/local/bin is not writable. I recommend that you make it writable. See https://gitl

## tlmgr --repository http://www.preining.info/tlgpg/ install tlgpg

## tlmgr option repository 'https://ctan.math.illinois.edu/systems/texlive/tlnet'

## tlmgr update --list
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v ggplot2 3.4.0      v purrr   1.0.1
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.3.0      v stringr 1.5.0
## v readr   2.1.4      v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyquant)
```

```
## Loading required package: lubridate
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
##
```

```
## Loading required package: PerformanceAnalytics
## Loading required package: xts
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
##
## Attaching package: 'xts'
##
## The following objects are masked from 'package:dplyr':
##
##     first, last
##
##
## Attaching package: 'PerformanceAnalytics'
##
## The following object is masked from 'package:graphics':
##
##     legend
##
## Loading required package: quantmod
## Loading required package: TTR
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(zoo)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

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

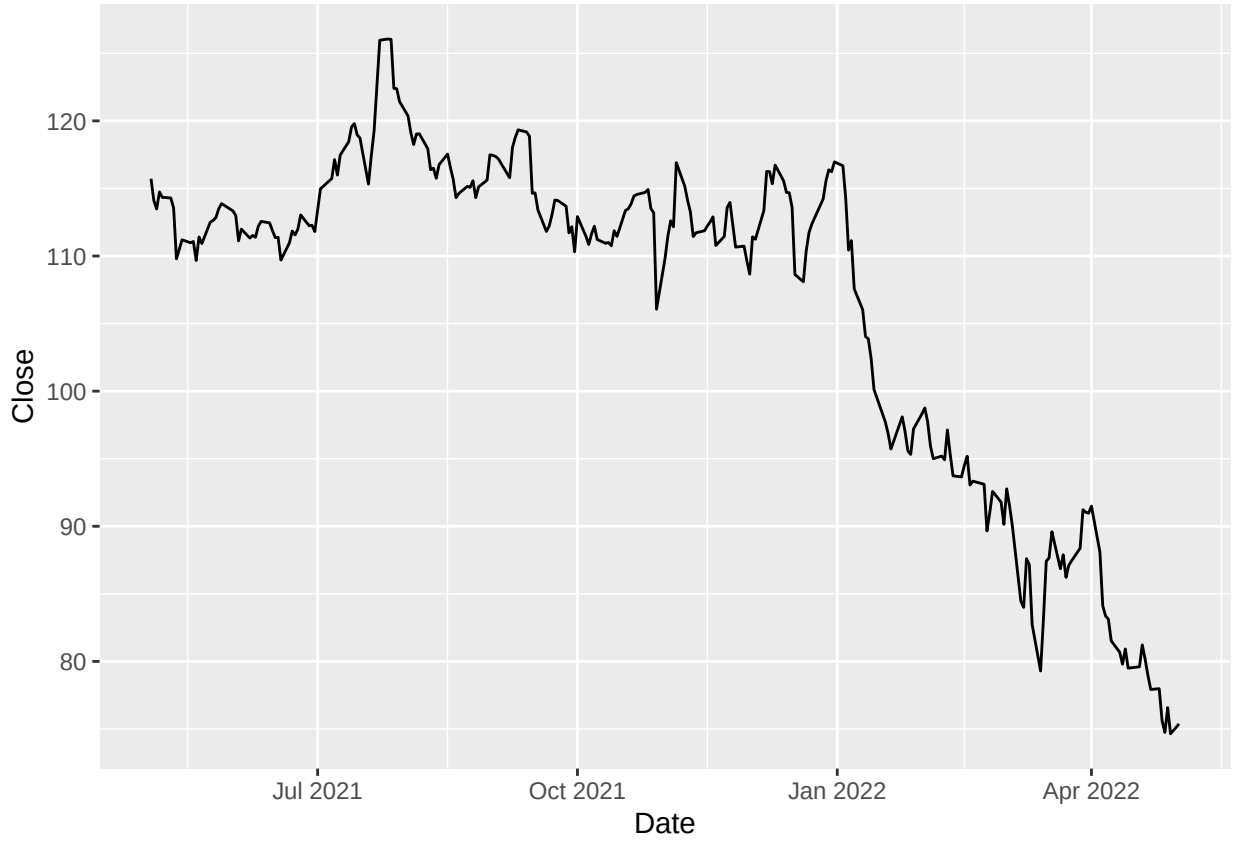
Q1.

```
data <- read.csv('SBUX.csv', header = T)
# transform the data type from 'chr' to 'Date'
data$Date = as.Date(data$Date)

# visualize our dataset
knitr::kable(head(data))
```

Date	Open	High	Low	Close	Adj.Close	Volume
2021-05-03	115.18	116.74	115.13	115.72	113.7461	5049800
2021-05-04	115.15	115.40	113.54	114.11	112.1635	6265100
2021-05-05	114.21	114.88	113.16	113.48	111.5443	4677700
2021-05-06	113.65	114.81	113.10	114.74	112.7828	4113300
2021-05-07	115.11	115.84	114.24	114.34	112.3896	4867500
2021-05-10	114.57	116.09	114.21	114.30	112.3503	5759500

```
ggplot(data, aes(x=Date, y = Close)) + geom_line()
```



```r
# normalize the stock price by using the 'min-max scaler'
data$min_lagged = lag(data$Low)
data$max_lagged = lag(data$High)
data$Close_norm = (data$Close - data$min_lagged) / (data$max_lagged - data$min_lagged)
model_data = matrix(data$Close_norm[-1])

# The last three scalded close prices
knitr::kable(tail(model_data,3))
```

[250,]	1.0857154
[251,]	-0.0982153
[252,]	0.3886473

Q2.

```r
train_data = head(model_data,-3)
test_data = tail(model_data, 6)
cat(dim(train_data)[1], 'days are divided into the training set.')
```

```
## 249 days are divided into the training set.
```

Q3.

```r
prediction = 3
lag = prediction
# Training X
# we lag the data 5 times and arrange that into columns
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] 244   3   1
```

```
dim(train_y)
```

```
## [1] 244   3   1
```

```
dim(test_X)
```

```
## [1] 1 3 1
```

```
dim(test_y)
```

```
## [1] 1 3 1
```

```
# [sample size, time steps, number of features (here only 1)]
```

Q4.

```
set_random_seed(123)
model <- keras_model_sequential()
model %>%
  layer_lstm(units = 150, input_shape = dim(train_X)[2:3])
model %>%
  layer_dense(units = dim(test_y)[2])

summary(model)
```

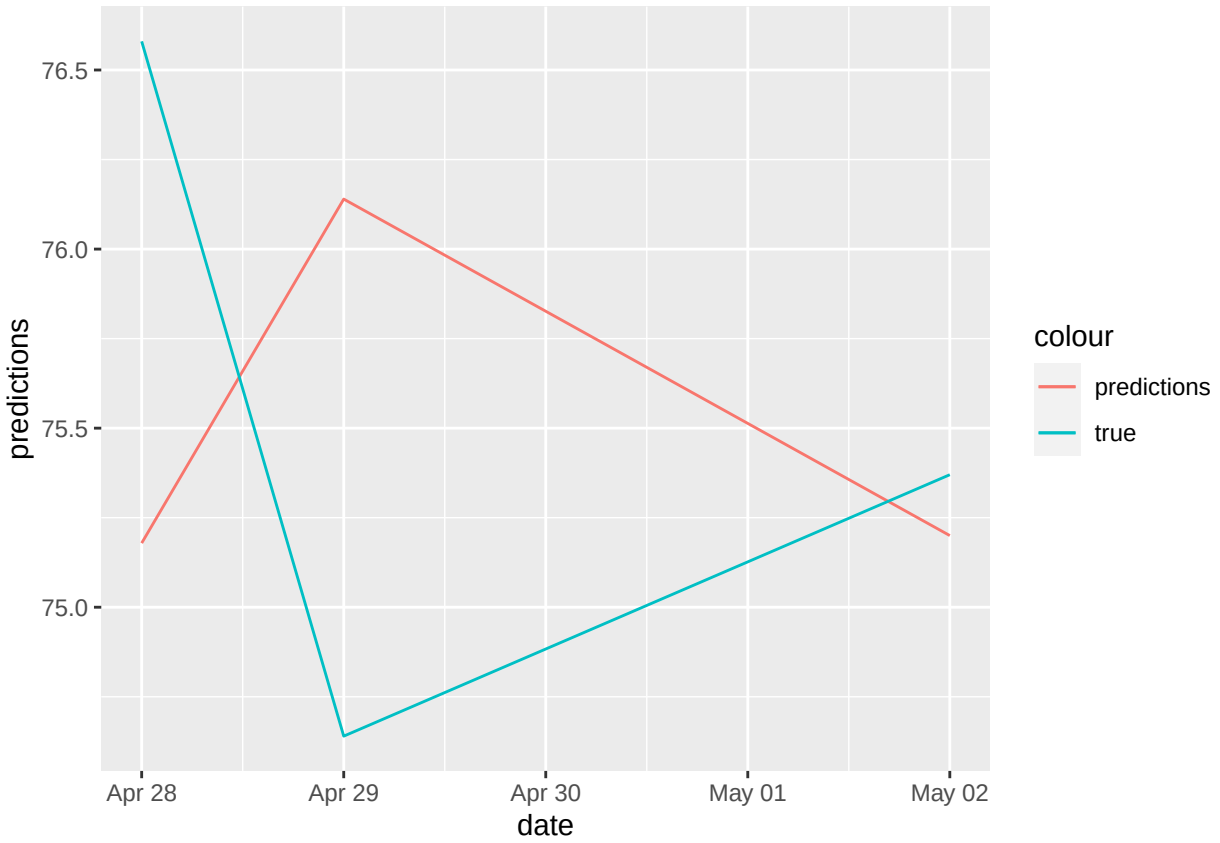
```
## Model: "sequential"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  lstm (LSTM)                        (None, 150)                     91200
##  dense (Dense)                      (None, 3)                       453
## =========================================================================
## Total params: 91,653
## Trainable params: 91,653
## Non-trainable params: 0
## _________________________________________________________________________
```

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

preds_norm = t(predict(model, test_X))
preds_complete = cbind(preds_norm, tail(data, prediction))
preds = preds_complete$preds_norm*(preds_complete$max_lagged - preds_complete$min_lagged) + preds_compl
predictions = data.frame(predictions = preds, true = preds_complete$Close, date = preds_complete$Date)
# Test MSE
(MSE.lstm = RMSE(predictions$true, predictions$predictions)^2)
```

```
## [1] 1.413504
```

```
# Plot 3-days forecast
ggplot(data = predictions, aes(x = date)) +
  geom_line(aes(y = predictions, color = 'predictions')) +
  geom_line(aes(y = true, color = 'true'))
```




```
# stock price of 2022-05-03
preds_norm1 = predict(model, test_y)[1]
(preds1 = preds_norm1*(data$High[253] - data$Low[253]) + data$Low[253])
```

```
## [1] 73.78454
```

Q5.

```
set_random_seed(123)
model <- keras_model_sequential()
model %>%
  layer_simple_rnn(units = 150, input_shape = dim(train_X)[2:3])
model %>%
  layer_dense(units = dim(test_y)[2])

summary(model)
```

```
## Model: "sequential_1"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  simple_rnn (SimpleRNN)             (None, 150)                     22800
##  dense_1 (Dense)                    (None, 3)                       453
## =========================================================================
```

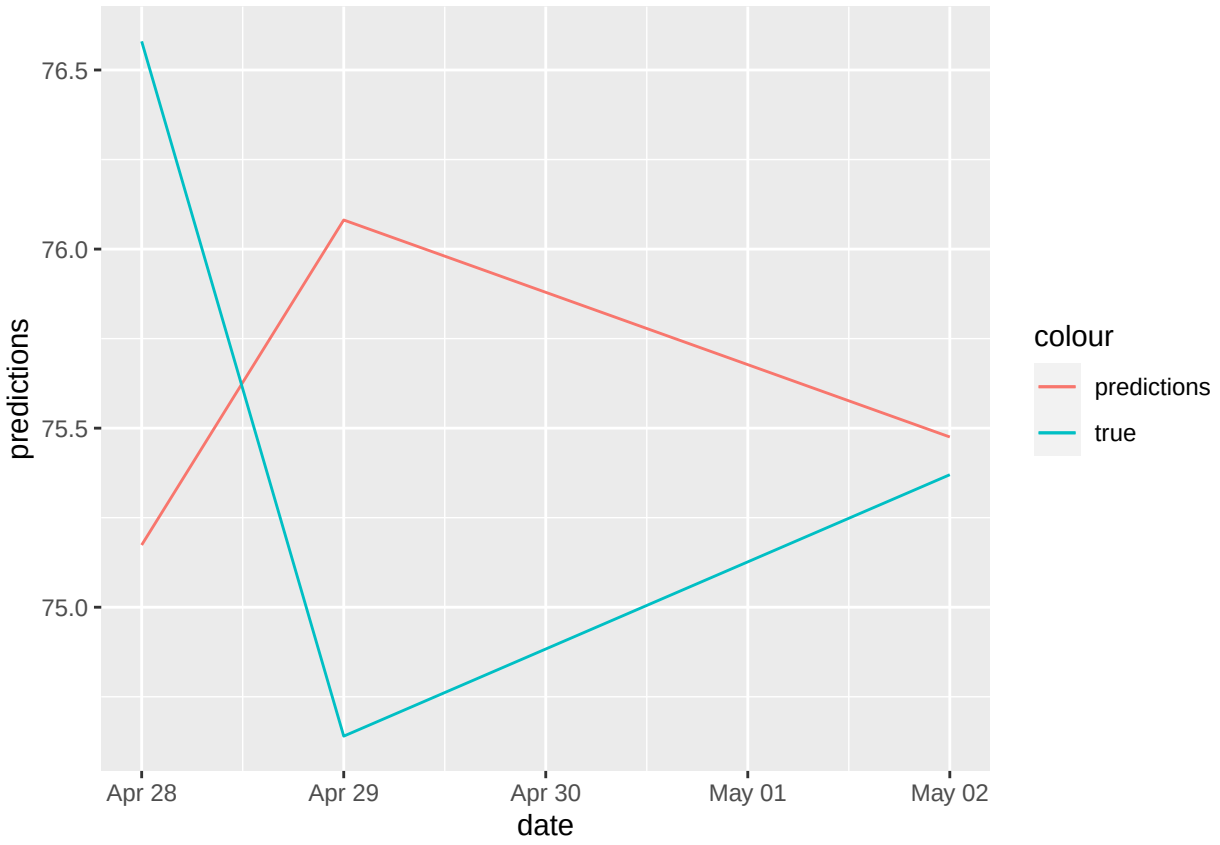
```
## Total params: 23,253
## Trainable params: 23,253
## Non-trainable params: 0
## _________________________________________________________________________________
```

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

preds_norm = t(predict(model, test_X))
preds_complete = cbind(preds_norm, tail(data, prediction))
preds = preds_complete$preds_norm*(preds_complete$max_lagged - preds_complete$min_lagged) + preds_compl
predictions = data.frame(predictions = preds, true = preds_complete$Close, date = preds_complete$Date)
# Test MSE
(MSE.rnn = RMSE(predictions$true, predictions$predictions)^2)
```

```
## [1] 1.355219
```

```
# Plot 3-days forecast
ggplot(data = predictions, aes(x = date)) +
  geom_line(aes(y = predictions, color = 'predictions')) +
  geom_line(aes(y = true, color = 'true'))
```




```
# stock price of 2022-05-03
preds_norm1 = predict(model, test_y)[1]
(preds1 = preds_norm1*(data$High[253] - data$Low[253]) + data$Low[253])
```

```
## [1] 73.8873
```

Q6.

Test MSE of LSTM and RNN:

```
MSE.lstm
```

```
## [1] 1.413504
```

```
MSE.rnn
```

```
## [1] 1.355219
```

Here the RNN algorithm in Question 4 is better for this dataset.