```
library(keras)
library(Hmisc)
library(lubridate)
library(gtools)
library(data.table)
library(onehot)
library(pracma)
library(abind)


###
### Data Preprocessing step
###


setwd("C:/Users/nhche/Development/stat-430")

es_dir="datasets/algoseek_ES_M1/"

nq_dir="datasets/algoseek_NQ_M1/"

es_files = list.files(es_dir)

nq_files = list.files(nq_dir)

num_cols = 20

mavg_periods = as.integer(c(2:num_cols)**1.8)

es_train = array(0,dim=c(0,num_cols,2))


es_min_counts = c(1)

eval_windows = c(1,5,15,30,60) #for different time periods of
prediction

es_Y = array(NA, dim=c(0, length(eval_windows)))



for(i in c(1:length(es_files))){
  fname = es_files[i]

  dat = read.csv(paste(es_dir,fname, sep=''))
```

```
  for (w in mavg_periods){
    ema = tail(movavg(mid_price, w, type='e'), kept)
    sma = tail(movavg(mid_price, w, type='s'), kept)
    temp[,j,1] = ema
    temp[,j,2] = sma
    j = j+1
  }
  print(dim(es_train))
  es_train = abind(es_train,temp, along=1)


  temp_Y = array(0, dim = c(kept, length(eval_windows)))

  for(i in c(1:length(eval_windows))){
    w = eval_windows[i]
    avgMprice <- c(rep(NA, w-1), zoo::rollmean(tail(mid_price,
kept), k=w, align="left"))
    preMP <- avgMprice
    postMP <- c(avgMprice[-(1:w)], rep(NA,w))
    price_change <- postMP - preMP
    temp_Y[,i]= price_change
  }
  es_Y = abind(es_Y, temp_Y, along=1)

}

nq_train = array(0,dim=c(0,num_cols,2))
nq_min_counts = c(1)
nq_Y = array(NA, dim=c(0, length(eval_windows)))

for(i in c(1:length(nq_files))){
  fname = nq_files[i]

  dat = read.csv(paste(nq_dir,fname, sep=''))
  if(dim(dat)[1] < 1000){
    next
  }

  kept = dim(dat)[1]-mavg_periods[num_cols-1]

  nq_min_counts=c(nq_min_counts,
kept+nq_min_counts[length(nq_min_counts)])

  temp=array(0,dim=c(kept, num_cols, 2))
```

```
  for(i in c(1:length(eval_windows))){
    w = eval_windows[i]
    avgMprice <- c(rep(NA, w-1), zoo::rollmean(tail(mid_price,
kept), k=w, align="left"))
    preMP <- avgMprice
    postMP <- c(avgMprice[-(1:w)], rep(NA,w))
    price_change <- postMP - preMP
    temp_Y[,i] = price_change
  }
  nq_Y = abind(nq_Y, temp_Y, along=1)


}

hist(na.omit(nq_Y[,1]), breaks=100,
xlim=range(-10,10),main='Histogram of NQ Price Changes')
hist(na.omit(es_Y[,1]), breaks=100, xlim=range(-5,5),
main='Histogram of ES Price Changes')

sd(nq_Y, na.rm=T)*sqrt(2/pi)
sd(es_Y, na.rm=T)*sqrt(2/pi)

for(k in c(1:length(es_min_counts)-1)){
  s=es_min_counts[k]
  for(i in c(0:mavg_periods[num_cols-1])){
    for(j in c(1:dim(es_Y)[2])){
      es_Y[s+i,j] = NA
    }
  }
}

for(k in c(1:length(nq_min_counts)-1)){
  s=nq_min_counts[k]
  for(i in c(0:mavg_periods[num_cols-1])){
    for(j in c(1:dim(nq_Y)[2])){
      nq_Y[s+i,j] = NA
    }
  }
}

fracDiff = function(datc, d=.5, tau=.0005){
  w_n = 1
  w = 1
  while (abs(w_n) > tau && length(w) < length(datc)){
```

```
  fracD
}

fracD_es = es_train
fracD_nq = nq_train

for(i in c(1:num_cols)){
  print(i)
  fracD_es[,i,1] = fracDiff(es_train[,i,1])
  fracD_es[,i,2] = fracDiff(es_train[,i,2])

  fracD_nq[,i,1] = fracDiff(nq_train[,i,1])
  fracD_nq[,i,2] = fracDiff(nq_train[,i,2])
}

norm_es = fracD_es
norm_nq = fracD_nq

for(i in c(1:num_cols)){
  norm_es[1:69,i,1]=0
  norm_es[70:dim(fracD_es)[1],i,1]=(fracD_es[70:dim(fracD_es)
[1],i,1] - mean(fracD_es[70:dim(fracD_es)[1],i,1], na.rm=T))/
sd(fracD_es[70:dim(fracD_es)[1],i,1], na.rm=T)
  norm_es[1:69,i,2]=0
  norm_es[70:dim(fracD_es)[1],i,2]=(fracD_es[70:dim(fracD_es)
[1],i,2] - mean(fracD_es[70:dim(fracD_es)[1],i,2], na.rm=T))/
sd(fracD_es[70:dim(fracD_es)[1],i,2], na.rm=T)

  norm_nq[1:69,i,1]=0
  norm_nq[70:dim(fracD_nq)[1],i,1]=(fracD_nq[70:dim(fracD_nq)
[1],i,1] - mean(fracD_nq[70:dim(fracD_nq)[1],i,1], na.rm=T))/
sd(fracD_nq[70:dim(fracD_nq)[1],i,1], na.rm=T)
  norm_nq[1:69,i,2]=0
  norm_nq[70:dim(fracD_nq)[1],i,2]=(fracD_nq[70:dim(fracD_nq)
[1],i,2] - mean(fracD_nq[70:dim(fracD_nq)[1],i,2], na.rm=T))/
sd(fracD_nq[70:dim(fracD_nq)[1],i,2], na.rm=T)
}

norm_es[is.na(norm_es)]=0
norm_nq[is.na(norm_nq)]=0


###
###Models and Training
###
```

```
  {
    rows_with_Y <- intersect(c(w:dim(Y_data)[1]), which(!
is.na(Y_data[,dim(Y_data)[2]])) )

    rows <- sample( rows_with_Y, batch_size, replace = TRUE )

    X = array(dim=c(batch_size, w, 40))
    Y = array(dim=c(batch_size))

    for(i in 1:length(rows)){
      X[i,,]=X_data[(rows[i]-w+1):rows[i],]
      Y[i] = Y_data[rows[i],1]
    }
    list(X, Y)
  }
}

names = c('nq','es')

nq_train_x = norm_nq[1:as.integer(.6*dim(nq_Y)[1]),,]
nq_train_y = nq_Y[1:as.integer(.6*dim(nq_Y)[1]),]

es_train_x = norm_es[1:as.integer(.6*dim(es_Y)[1]),,]
es_train_y = es_Y[1:as.integer(.6*dim(es_Y)[1]),]

es_val_x = norm_es[as.integer(.6*dim(es_Y)[1]):as.integer(.
8*dim(es_Y)[1]),,]
es_val_y = es_Y[as.integer(.6*dim(es_Y)[1]):as.integer(.
8*dim(es_Y)[1]),]

nq_val_x = norm_nq[as.integer(.6*dim(nq_Y)[1]):as.integer(.
8*dim(nq_Y)[1]),,]
nq_val_y = nq_Y[as.integer(.6*dim(nq_Y)[1]):as.integer(.
8*dim(nq_Y)[1]),]

es_test_x = norm_es[-c(1:as.integer(.8*dim(es_Y)[1])),,]
es_test_y = es_Y[-c(1:as.integer(.8*dim(es_Y)[1])),]

nq_test_x = norm_nq[-c(1:as.integer(.8*dim(nq_Y)[1])),,]
nq_test_y = nq_Y[-c(1:as.integer(.8*dim(nq_Y)[1])),]

train_xlist = list(nq_train_x, es_train_x)
val_xlist = list(nq_val_x, es_val_x)
train_ylist = list(nq_train_y, es_train_y)
val_ylist = list(nq_val_y, es_val_y)
```

```
    model %>% compile(
      loss = "mean_absolute_error",
      optimizer = optimizer_adam(lr = 2e-4),
      metrics = c("mae")
    )


    earlyStop <- callback_early_stopping(monitor = "val_loss",
patience = 5)

    checkPoint <- callback_model_checkpoint(filepath =
file.path(paste("C:/Users/nhche/Development/stat-430/
project/",names[i],names[j],"-simple.h5", sep='')),
                                            monitor = "val_loss",
save_best_only = T)

    schedule <- function(epoch,lr) (lr)*(0.8^(epoch))
    schedulLr <- callback_learning_rate_scheduler(schedule)

    batch_size = 64
    epochs = 15

    train_x = train_xlist[[i]]
    train_y = train_ylist[[i]]

    val_x = val_xlist[[j]]
    val_y = val_ylist[[j]]

    his <- model %>% fit_generator(sampling_generator(train_x,
train_y, batch_size = batch_size, w=w),
                                   steps_per_epoch = 250, epochs
= epochs, callbacks = list(earlyStop,checkPoint, schedulLr),
                                   validation_data =
sampling_generator(val_x, val_y, batch_size = batch_size, w=w),
                                   validation_steps = 100)
  }
}

for(i in c(1,2)){
  for(j in c(1,2)){

    k_clear_session()

    w=60
    model <- keras_model_sequential() %>%
```

```
    checkPoint <- callback_model_checkpoint(filepath =
file.path(paste("C:/Users/nhche/Development/stat-430/
project/",names[i],names[j],"-lb1.h5", sep='')),
                                            monitor = "val_loss",
save_best_only = T)

    schedule <- function(epoch,lr) (lr)*(0.8^(epoch))
    schedulLr <- callback_learning_rate_scheduler(schedule)

    batch_size = 64
    epochs = 15

    train_x = train_xlist[[i]]
    train_y = train_ylist[[i]]

    val_x = val_xlist[[j]]
    val_y = val_ylist[[j]]

    his <- model %>% fit_generator(sampling_generator(train_x,
train_y, batch_size = batch_size, w=w),
                                   steps_per_epoch = 250, epochs
= epochs, callbacks = list(earlyStop,checkPoint, schedulLr),
                                   validation_data =
sampling_generator(val_x, val_y, batch_size = batch_size, w=w),
                                   validation_steps = 100)
  }
}

for(i in c(1,2)){
  for(j in c(1,2)){

    k_clear_session()
    model <- keras_model_sequential() %>%
      layer_conv_1d(filters = 1, kernel_size = 4, activation =
"relu",
                    input_shape = c(w, 40)) %>%
      layer_gru(unit=8, return_sequences = TRUE) %>%
      layer_gru(unit=16) %>%
      layer_dense(units = 12, activation = "linear") %>%
      layer_dense(units = 1, activation = "linear")

    summary(model)

    model %>% compile(
```

```
    val_x = abind(val_x[,,1],val_x[,,2],along=2)
    val_y = val_ylist[[j]]

    his <- model %>% fit_generator(rnn_generator(train_x,
train_y, batch_size = batch_size, w=w),
                                   steps_per_epoch = 250, epochs
= epochs, callbacks = list(earlyStop,checkPoint, schedulLr),
                                   validation_data =
rnn_generator(val_x, val_y, batch_size = batch_size, w=w),
                                   validation_steps = 100)
  }
}

###
### Model Evaluation
###

eses <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/eses-lb1.h5"))
esnq <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/esnq-lb1.h5"))
nqes <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/nqes-lb1.h5"))
nqnq <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/nqnq-lb1.h5"))

esesS <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/eses-simple.h5"))
esnqS <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/esnq-simple.h5"))
nqesS <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/nqes-simple.h5"))
nqnqS <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/nqnq-simple.h5"))

esesR <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/eses-rnn.h5"))
esnqR <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/esnq-rnn.h5"))
nqesR <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/nqes-rnn.h5"))
nqnqR <- load_model_hdf5(file.path("C:/Users/nhche/Development/
stat-430/project/nqnq-rnn.h5"))

x_test = nq_test_x
```

```
barplot(c(NQ_NQ_Deep$loss, NQ_NQ_Shallow$loss, ES_NQ_Deep$loss,
ES_NQ_Shallow$loss, NQ_NQ_RNN$loss, ES_NQ_RNN$loss), names.arg =
c("NQ_NQ_Deep", "NQ_NQ_Shallow", "ES_NQ_Deep", "ES_NQ_Shallow",
"NQ_NQ_RNN", "ES_NQ_RNN"))
```

```
x_test = es_test_x
y_test = es_test_y

NQ_ES_Deep <- nqes %>%
evaluate_generator(sampling_generator(x_test, y_test, batch_size
= batch_size, w=w),
                                          steps = 100)
ES_ES_Deep <- eses %>%
evaluate_generator(sampling_generator(x_test, y_test, batch_size
= batch_size, w=w),
                                          steps = 100)
NQ_ES_Shallow <- nqesS %>%
evaluate_generator(sampling_generator(x_test, y_test, batch_size
= batch_size, w=w),
                                            steps = 100)
ES_ES_Shallow <- esesS %>%
evaluate_generator(sampling_generator(x_test, y_test, batch_size
= batch_size, w=w),
                                            steps = 100)
rnn_x = abind(x_test[,,1],x_test[,,2],along=2)

NQ_ES_RNN <- nqesR %>% evaluate_generator(rnn_generator(rnn_x,
y_test, batch_size = batch_size, w=w),
                                          steps = 100)
ES_ES_RNN <- esesR %>% evaluate_generator(rnn_generator(rnn_x,
y_test, batch_size = batch_size, w=w),
                                          steps = 100)

NQ_ES_Deep$loss

NQ_ES_Shallow$loss

ES_ES_Deep$loss
```