Problem Set 5

Jiaqi Li

May 12, 2019

A:Set up data for analysis amenable to cross-validation routines

```
options(warn = -1)
suppressMessages(library(data.table))
suppressMessages(library(dplyr))
suppressMessages(library(readxl))
suppressMessages(library(ggplot2))

####################################################################
#                              Part A                              #
####################################################################

#---------------------------question 1----------------------------#
Fport = read_excel("French_Portfolio_Returns.xlsx", skip = 1,
                   sheet = 3) %>% as.data.table()
Fport[, Date := as.Date(as.character(Date*100+1),format="%Y%m%d")]
setkey(Fport,Date)
sample = Fport[Date>="1963-07-01" & Date <= "2016-12-01",]
sample[,`:=`(Month = .I,Date = NULL)]

#---------------------------question 2---------------------------#
names(sample)[1:138] = c(1:138)
Data = reshape(sample,varying = names(sample)[1:138],
               v.names = "ExRet", timevar = "Portfolio",
               times = names(sample)[1:138],
               direction = "long")
Data$id=NULL
Data$Month = as.numeric(Data$Month)
Data$Portfolio = as.numeric(Data$Portfolio)
Data$ExRet = as.numeric(Data$ExRet)

#---------------------------question 3---------------------------#
Data[,`:=`(lag1Ret = shift(ExRet),
           lag2Ret = shift(ExRet,2)),
     by = Portfolio]
Data[,sumlagRet := Reduce(`+`,shift(ExRet,3:12)),by=Portfolio]

Data[,`:=`(lag1Ret_2 = shift(ExRet)^2,
           lag2Ret_2 = shift(ExRet,2)^2,
           sumlagRet_2 = sumlagRet^2),by = Portfolio]
Data
```

```
##        Month Portfolio     ExRet   lag1Ret   lag2Ret sumlagRet
##     1:     1         1  0.027616        NA        NA        NA
##     2:     2         1 -0.005684  0.027616        NA        NA
##     3:     3         1 -0.022155 -0.005684  0.027616        NA
```

```
##     4:     4         1 -0.004219 -0.022155 -0.005684        NA
##     5:     5         1 -0.033259 -0.004219 -0.022155        NA
##    ---
## 88592:   638       138  0.006435  0.013729  0.029512 -0.116215
## 88593:   639       138  0.005681  0.006435  0.013729 -0.024476
## 88594:   640       138 -0.024368  0.005681  0.006435  0.069347
## 88595:   641       138  0.060264 -0.024368  0.005681  0.017455
## 88596:   642       138  0.009861  0.060264 -0.024368  0.029956
##           lag1Ret_2    lag2Ret_2  sumlagRet_2
##     1:           NA           NA           NA
##     2: 7.626435e-04           NA           NA
##     3: 3.230786e-05 7.626435e-04           NA
##     4: 4.908440e-04 3.230786e-05           NA
##     5: 1.779996e-05 4.908440e-04           NA
##    ---
## 88592: 1.884854e-04 8.709581e-04 0.0135059262
## 88593: 4.140922e-05 1.884854e-04 0.0005990746
## 88594: 3.227376e-05 4.140922e-05 0.0048090064
## 88595: 5.937994e-04 3.227376e-05 0.0003046770
## 88596: 3.631750e-03 5.937994e-04 0.0008973619
```

B: Decision Trees

Inputs	value
number of trees	500
max nodes is	30
mtry is	2

```
####################################################################
#                             Part B                               #
####################################################################
suppressMessages(library(glmnet))
suppressMessages(library(foreign))
suppressMessages(library(randomForest))
suppressMessages(library(lfe))
suppressMessages(library(xgboost))

#---------------------------question 1----------------------------#
train = Data[complete.cases(Data),]
setkey(train,Month)
y_train = as.vector(as.matrix(train[.(13:(642-(2016-2009)*12)),list(ExRet)]))
x_train = as.matrix(train[.(13:(642-(2016-2009)*12)),list(Month, Portfolio,
          lag1Ret,lag2Ret,sumlagRet,lag1Ret_2,lag2Ret_2,sumlagRet_2)])
y_test = as.vector(as.matrix(train[.((642-(2016-2009)*12+1):642),list(ExRet)]))
x_test = as.matrix(train[.((642-(2016-2009)*12+1):642),list(Month, Portfolio,
          lag1Ret,lag2Ret,sumlagRet,lag1Ret_2,lag2Ret_2,sumlagRet_2)])

RF_train = randomForest(x_train,y_train,ntree = 500,maxnodes = 30,mtry = 2)

RF_train_pred = predict(RF_train,x_train)
```

```r
train_felm = as.data.frame(cbind(y_train,RF_train_pred,
                                 Month = train[.(13:(642-(2016-2009)*12))]$Month))
summary(felm(y_train~RF_train_pred |0|0|Month, data = train_felm))
```

```
##
## Call:
##    felm(formula = y_train ~ RF_train_pred | 0 | 0 | Month, data = train_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35662 -0.03150  0.00028  0.03143  0.69429
##
## Coefficients:
##                Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)   -0.004939     0.002209  -2.236   0.0254 *
## RF_train_pred  1.993301     0.165240  12.063   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05481 on 75274 degrees of freedom
## Multiple R-squared(full model): 0.1062   Adjusted R-squared: 0.1062
## Multiple R-squared(proj model): 0.1062   Adjusted R-squared: 0.1062
## F-statistic(full model, *iid*): 8943 on 1 and 75274 DF, p-value: < 2.2e-16
## F-statistic(proj model): 145.5 on 1 and 545 DF, p-value: < 2.2e-16
```

```r
RF_test_pred = predict(RF_train,x_test)
test_felm = as.data.frame(cbind(y_test,RF_test_pred,
                                Month = train[.((642-(2016-2009)*12+1):642)]$Month))
summary(felm(y_test~RF_test_pred |0|0|Month, data = test_felm))
```

```
##
## Call:
##    felm(formula = y_test ~ RF_test_pred | 0 | 0 | Month, data = test_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41297 -0.02854  0.00286  0.02975  0.33208
##
## Coefficients:
##              Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)  0.004548     0.008727   0.521    0.602
## RF_test_pred 0.227197     0.388212   0.585    0.558
##
## Residual standard error: 0.05101 on 11590 degrees of freedom
## Multiple R-squared(full model): 0.002356   Adjusted R-squared: 0.00227
## Multiple R-squared(proj model): 0.002356   Adjusted R-squared: 0.00227
## F-statistic(full model, *iid*):27.38 on 1 and 11590 DF, p-value: 1.705e-07
## F-statistic(proj model): 0.3425 on 1 and 83 DF, p-value: 0.56
```

```r
port_ret = NULL
row_counter_end = 0
for (i in (642-(2016-2009)*12+1):642)
{
  month_length <- Data[Month==i,]$ExRet %>% length()
  row_counter_start = row_counter_end + 1
```

```r
  row_counter_end = row_counter_end + month_length
  x_temp <- RF_test_pred[row_counter_start:row_counter_end]
  y_temp <- y_test[row_counter_start:row_counter_end]
  fit_yr <- lm(y_temp ~ x_temp)
  temp <- coefficients(fit_yr)
  port_ret = rbind(port_ret,temp[2])
}
fm_RF_output = list(SR_Return = mean(port_ret)/sd(port_ret),
                    tstat_MeanRet = sqrt((2016-2009)*12-1)*
                      mean(port_ret)/sd(port_ret))
fm_RF_output
```

```
## $SR_Return
## [1] -0.04472687
##
## $tstat_MeanRet
## [1] -0.4074812
```

```r
#---------------------------question 2----------------------------#
params1 <- list(booster = "gbtree", objective = "reg:linear",
               eta = 0.1, gamma = 0, max_depth = 1)
params2 <- list(booster = "gbtree", objective = "reg:linear",
               eta = 0.1, gamma = 0, max_depth = 6)
params3 <- list(booster = "gbtree", objective = "reg:linear",
               eta = 0.3, gamma = 0, max_depth = 1)
params4 <- list(booster = "gbtree", objective = "reg:linear",
               eta = 0.3, gamma = 0, max_depth = 6)

for(i in 1:4){
  sink('NUL')
  xgb_train <- xgb.DMatrix(data = x_train, label = y_train)
  xgbcv <- xgb.cv(params = get(paste("params",i,sep="")), data = xgb_train,
                  nfold = 10, nrounds = 200, showsd = T, print_every_n = 10)
  cv_nrounds = which.min(xgbcv$evaluation_log$test_rmse_mean)
  xgb_optb <- xgboost(params = get(paste("params",i,sep="")), data = xgb_train,
                      nround = cv_nrounds)
  xgb_train_pred <- predict(xgb_optb, xgb_train)
  train_felm = as.data.frame(cbind(y_train,xgb_train_pred,
                     Month = train[.(13:(642-(2016-2009)*12))]$Month))
  assign(paste("XGB_train",i,sep=""),
       summary(felm(y_train~xgb_train_pred|0|0|Month,data = train_felm)))
  xgb_test <- xgb.DMatrix(data = x_test, label = y_test)
  xgb_test_pred <- predict(xgb_optb, xgb_test)
  test_felm <- as.data.frame(cbind(y_test,xgb_test_pred,
                     Month = train[.((642-(2016-2009)*12+1):642)]$Month))
  assign(paste("XGB_test",i,sep=""),
       summary(felm(y_test~xgb_test_pred|0|0|Month,data = test_felm)))

  port_ret = NULL
  row_counter_end = 0
  for (j in (642-(2016-2009)*12+1):642)
  {
    month_length <- Data[Month==j,]$ExRet %>% length()
    row_counter_start = row_counter_end + 1
```

```r
    row_counter_end = row_counter_end + month_length
    x_temp <- xgb_test_pred[row_counter_start:row_counter_end]
    y_temp <- y_test[row_counter_start:row_counter_end]
    fit_yr <- lm(y_temp ~ x_temp)
    temp <- coefficients(fit_yr)
    port_ret = rbind(port_ret,temp[2])
  }
  assign(paste("fm_xgb_output",i,sep=""),list(SR_Return = mean(port_ret)/sd(port_ret),
                    tstat_MeanRet = sqrt((2016-2009)*12-1)*mean(port_ret)/sd(port_ret)))
  sink()
}
#set 1
XGB_train1
```

```
##
## Call:
##    felm(formula = y_train ~ xgb_train_pred | 0 | 0 | Month, data = train_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35741 -0.03146  0.00069  0.03146  0.76086
##
## Coefficients:
##                 Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)    -0.002695     0.002180  -1.236    0.216
## xgb_train_pred  1.540546     0.174114   8.848   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05548 on 75274 degrees of freedom
## Multiple R-squared(full model): 0.08421   Adjusted R-squared: 0.0842
## Multiple R-squared(proj model): 0.08421   Adjusted R-squared: 0.0842
## F-statistic(full model, *iid*): 6922 on 1 and 75274 DF, p-value: < 2.2e-16
## F-statistic(proj model): 78.29 on 1 and 545 DF, p-value: < 2.2e-16
```

```r
XGB_test1
```

```
##
## Call:
##    felm(formula = y_test ~ xgb_test_pred | 0 | 0 | Month, data = test_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33877 -0.02854  0.00233  0.02951  0.32970
##
## Coefficients:
##               Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)    0.03340      0.03315   1.007    0.314
## xgb_test_pred -0.63114      0.86958  -0.726    0.468
##
## Residual standard error: 0.05101 on 11590 degrees of freedom
## Multiple R-squared(full model): 0.002209   Adjusted R-squared: 0.002123
## Multiple R-squared(proj model): 0.002209   Adjusted R-squared: 0.002123
## F-statistic(full model, *iid*):25.66 on 1 and 11590 DF, p-value: 4.136e-07
## F-statistic(proj model): 0.5268 on 1 and 83 DF, p-value: 0.47
```

```r
fm_xgb_output1
```

```
## $SR_Return
## [1] 0.04846428
##
## $tstat_MeanRet
## [1] 0.4415306
```

```r
#set 2
XGB_train2
```

```
##
## Call:
##    felm(formula = y_train ~ xgb_train_pred | 0 | 0 | Month, data = train_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.27974 -0.01657 -0.00046  0.01580  0.35996
##
## Coefficients:
##                   Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)    -0.0015594    0.0005753  -2.711  0.00672 **
## xgb_train_pred  1.3127675    0.0211773  61.989  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03087 on 75274 degrees of freedom
## Multiple R-squared(full model): 0.7164   Adjusted R-squared: 0.7164
## Multiple R-squared(proj model): 0.7164   Adjusted R-squared: 0.7164
## F-statistic(full model, *iid*):1.902e+05 on 1 and 75274 DF, p-value: < 2.2e-16
## F-statistic(proj model):  3843 on 1 and 545 DF, p-value: < 2.2e-16
```

```r
XGB_test2
```

```
##
## Call:
##    felm(formula = y_test ~ xgb_test_pred | 0 | 0 | Month, data = test_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42201 -0.02826  0.00334  0.03036  0.33436
##
## Coefficients:
##                 Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)   -0.002817     0.009366  -0.301    0.764
## xgb_test_pred  0.309091     0.250609   1.233    0.217
##
## Residual standard error: 0.05083 on 11590 degrees of freedom
## Multiple R-squared(full model): 0.009383   Adjusted R-squared: 0.009297
## Multiple R-squared(proj model): 0.009383   Adjusted R-squared: 0.009297
## F-statistic(full model, *iid*):109.8 on 1 and 11590 DF, p-value: < 2.2e-16
## F-statistic(proj model): 1.521 on 1 and 83 DF, p-value: 0.2209
```

```r
fm_xgb_output2
```

```
## $SR_Return
```

```
## [1] -0.05352264
##
## $tstat_MeanRet
## [1] -0.4876144
```

```r
#set 3
XGB_train3
```

```
##
## Call:
##    felm(formula = y_train ~ xgb_train_pred | 0 | 0 | Month, data = train_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35010 -0.03087  0.00021  0.03041  0.75523
##
## Coefficients:
##                 Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)    -0.002297     0.002037  -1.128    0.259
## xgb_train_pred  1.460720     0.128609  11.358   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05441 on 75274 degrees of freedom
## Multiple R-squared(full model): 0.1193   Adjusted R-squared: 0.1193
## Multiple R-squared(proj model): 0.1193   Adjusted R-squared: 0.1193
## F-statistic(full model, *iid*):1.019e+04 on 1 and 75274 DF, p-value: < 2.2e-16
## F-statistic(proj model):   129 on 1 and 545 DF, p-value: < 2.2e-16
```

```r
XGB_test3
```

```
##
## Call:
##    felm(formula = y_test ~ xgb_test_pred | 0 | 0 | Month, data = test_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37115 -0.02877  0.00214  0.02950  0.32833
##
## Coefficients:
##               Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)    0.01326      0.02309   0.574    0.566
## xgb_test_pred -0.07361      0.67291  -0.109    0.913
##
## Residual standard error: 0.05106 on 11590 degrees of freedom
## Multiple R-squared(full model): 5.207e-05   Adjusted R-squared: -3.421e-05
## Multiple R-squared(proj model): 5.207e-05   Adjusted R-squared: -3.421e-05
## F-statistic(full model, *iid*):0.6035 on 1 and 11590 DF, p-value: 0.4373
## F-statistic(proj model): 0.01197 on 1 and 83 DF, p-value: 0.9132
```

```r
fm_xgb_output3
```

```
## $SR_Return
## [1] 0.03754614
##
## $tstat_MeanRet
```

```
## [1] 0.3420616
```

```
#set 4
XGB_train4
```

```
##
## Call:
##    felm(formula = y_train ~ xgb_train_pred | 0 | 0 | Month, data = train_felm)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.186619 -0.013299 -0.000331  0.012779  0.219724
##
## Coefficients:
##                  Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)    -0.0006948    0.0002610  -2.662  0.00778 **
## xgb_train_pred  1.1391945    0.0078482 145.154  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02496 on 75274 degrees of freedom
## Multiple R-squared(full model): 0.8146   Adjusted R-squared: 0.8146
## Multiple R-squared(proj model): 0.8146   Adjusted R-squared: 0.8146
## F-statistic(full model, *iid*):3.307e+05 on 1 and 75274 DF, p-value: < 2.2e-16
## F-statistic(proj model): 2.107e+04 on 1 and 545 DF, p-value: < 2.2e-16
```

```
XGB_test4
```

```
##
## Call:
##    felm(formula = y_test ~ xgb_test_pred | 0 | 0 | Month, data = test_felm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.39527 -0.02876  0.00235  0.02964  0.32807
##
## Coefficients:
##               Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)   0.007560     0.005553   1.361    0.173
## xgb_test_pred 0.080580     0.157742   0.511    0.609
##
## Residual standard error: 0.05104 on 11590 degrees of freedom
## Multiple R-squared(full model): 0.001062   Adjusted R-squared: 0.0009755
## Multiple R-squared(proj model): 0.001062   Adjusted R-squared: 0.0009755
## F-statistic(full model, *iid*):12.32 on 1 and 11590 DF, p-value: 0.0004504
## F-statistic(proj model): 0.261 on 1 and 83 DF, p-value: 0.6108
```

```
fm_xgb_output4
```

```
## $SR_Return
## [1] -0.1308379
##
## $tstat_MeanRet
## [1] -1.19199
```

C: Asymptotic PCA

```
#######################################################################
#                              Part C                                  #
#######################################################################
suppressMessages(library(MTS))
suppressMessages(library(zoo))
suppressMessages(library(sandwich))

#--------------------------question 1----------------------------#
#names(sample)[1:138] = names(Fport)[2:139]
sample2 = Fport[Date>="1963-07-01" & Date <= "2016-12-01",]
sample2 = apply(sample2[,!"Date"],2,as.numeric) %>% as.data.table() %>% na.omit()
N = length(sample2$Agric)

APCA_loading = function(data,start,end,K){
  if(start < 0){
    return(list(NA))
  }else{
    sink('NUL')
    temp = apca(data[start:end,],K)
    sink()
    return(list(temp$loadings))
  }
}

apca_routine = list()
for(i in 1:N){
  test = APCA_loading(sample2,i-60,i,5)
  apca_routine = c(apca_routine,test)
}

N_loading = length(apca_routine)
for(i in 1:5){
  assign(paste("fact",i,"_ret",sep = ""),rep(NA,60))
}
for(i in 61:N_loading){
  fact1_ret[i] = sum(sample2[i,]*apca_routine[[i-1]][,1])
  fact2_ret[i] = sum(sample2[i,]*apca_routine[[i-1]][,2])
  fact3_ret[i] = sum(sample2[i,]*apca_routine[[i-1]][,3])
  fact4_ret[i] = sum(sample2[i,]*apca_routine[[i-1]][,4])
  fact5_ret[i] = sum(sample2[i,]*apca_routine[[i-1]][,5])
}
fact_ret = cbind(fact1_ret,fact2_ret,fact3_ret,fact4_ret,fact5_ret)

#part a----------------------------------------------------------
avgAnnFactRet = c(); AnnFactSD = c(); AnnFactSR = c(); Corr1mon = c()
for(i in 1:5){
  avgAnnFactRet[i] = mean(get(paste("fact",i,"_ret",sep="")),na.rm = T)*12
  AnnFactSD[i] = sd(get(paste("fact",i,"_ret",sep="")),na.rm = T)*sqrt(12)
  AnnFactSR[i] = avgAnnFactRet[i]/AnnFactSD[i]
  Corr1mon[i] = acf(na.omit(get(paste("fact",i,"_ret",sep=""))),plot=F)$acf[2]
}
```

```r
avgAnnFactRet
```

```
## [1] -0.27617791 -0.03828889 -0.07150723 -0.05448349  0.04781366
```

```r
AnnFactSD
```

```
## [1] 1.0232744 0.5017037 0.2723979 0.2132336 0.1728707
```

```r
AnnFactSR
```

```
## [1] -0.26989624 -0.07631772 -0.26251019 -0.25551080  0.27658622
```

```r
Corr1mon
```

```
## [1] -0.11138271  0.07052808  0.04940312  0.04846115  0.01700756
```

```r
#part b-----------------------------------------------------------
FactTstat = c()
for(i in 1:5){
  FactTstat[i] = AnnFactSR[i]*sqrt((N-61+1)/12)
  assign(paste("se",i,sep=""),
         sqrt(vcovHAC(lm(get(paste("fact",i,"_ret",sep=""))~1))))
}
FactTstat
```

```
## [1] -1.7595082 -0.4975307 -1.7113571 -1.6657266  1.8031215
```

```r
#part c-----------------------------------------------------------
```

Based on the average factor return and the Sharpe Ratio, these 5 factors do not seem to be very useful since 4 of these factor generates negative returns and the last factor only generate very small positive return.
Based on the t-stat, none of these 5 factors is very significant
Based on the autocorrelation, no factor suggests significantly high monthly first autocorrelation.
Therefore, there is not enough evidence that shows the 5 factors are good to use

```r
#part d-----------------------------------------------------------
signal = sample2
lambda = NULL
for (i in 61:N){
  x_temp <- apca_routine[[i-1]]
  x_signal <- t(signal[i-1,])
  y_temp <- t(sample2[i,])
  fit_yr <- lm(y_temp ~ x_temp + x_signal)
  temp <- coefficients(fit_yr)
  lambda = rbind(lambda,temp)
}
SignalMean = mean(lambda[,6])
SignalSD = sd(lambda[,6])
SignalTstat = sqrt(N-62+1)*SignalMean/SignalSD
SignalTstat
```

```
## [1] 1.271584
```

Procedure explanation:
Lagged returns (x_{signal}) used returns at time t
Loadings ($x_{loading}$) are also at time t
Regression returns (y_{return}) used returns at time t+1
Regress y_{return} on $x_{loading}$ and x_{signal} cross sectionally to get Lambdas for each of the 5 factors and the

signal
Generate lambdas through all time at compute mean and standard deviation of the signal lambda cross time. Compute t-stat by dividing signal lambda mean by signal lambda standard deviation and times square root of number of observations

The t-stat is actually quite small, which indicates that the lagged return may not be a significant signal.