Problem Set 5 Solution

Denis Mokanov

May 13, 2019

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

1. Upload this data to a data.table in R, delete the "date" column and instead create a column called "Month" which counts the observations over time. I.e., 196307 is 1, 196308 is 2, etc.
2. Reshape the data so that each each observation (row) has Month, Portfolio, ExRet in the columns. "Portfolio" is a number from 1 to 138 - corresponding to the previous columns in the data set.
3. For each portfolio, add the three following columns: one month lagged return, two months lagged return, sum of months 3-12 lagged return. Then, add the following additional three columns: Do the same for squared return - one month lagged squared return, two months lagged squared return, squared sum of lags 3-12 of returns.

```
# remove all variables
rm(list = ls())

# Load packages
library(xlsx)
library(data.table)
library(dplyr)
library(lfe)
library(randomForest)
library(xgboost)
library(MTS)
library(sandwich)

# Load data
french_portfolio_returns <- as.data.table(read.xlsx(file = "French_Portfolio_Returns.xlsx",
    sheetIndex = 3, startRow = 440, colnames = TRUE, stringsAsFactors = FALSE))

# Delete Date column
french_port_rets <- french_portfolio_returns[, `:=`(c("X196306", "colnames"),
    NULL)]

# change column headers for the portfolios
colnames(french_port_rets) <- paste0("p", 1:ncol(french_port_rets))

# add a Month column
french_port_rets <- cbind(Month = 1:nrow(french_port_rets), french_port_rets)

# we need to make sure all portfolio returns are numeric (this is related to
# the APCA exercise)
french_port_rets <- french_port_rets[, lapply(.SD, as.numeric), by = Month]

#'melt' the data table to arrive at the format required in the question
characteristics <- melt.data.table(french_port_rets, id.vars = "Month")

# finish data preparation by renaming the columns and making sure that ExRet
# and Portfolio are numeric
```

```r
colnames(characteristics) <- c("Month", "Portfolio", "ExRet")
characteristics$Portfolio <- as.numeric(sub(".", "", characteristics$Portfolio))
characteristics$ExRet <- as.numeric(characteristics$ExRet)

# create the lagged variables; in my experience the dplyr lag function works
# bettter than the stats lag function
characteristics <- characteristics[, `:=`(ExRet_lag1, dplyr::lag(ExRet, 1)),
    by = "Portfolio"]
characteristics <- characteristics[, `:=`(ExRet_lag2, dplyr::lag(ExRet, 2)),
    by = "Portfolio"]
characteristics <- characteristics[, `:=`(ExRet_lag312, dplyr::lag(ExRet, 3) +
    dplyr::lag(ExRet, 4) + dplyr::lag(ExRet, 5) + dplyr::lag(ExRet, 6) + dplyr::lag(ExRet,
    7) + dplyr::lag(ExRet, 8) + dplyr::lag(ExRet, 9) + dplyr::lag(ExRet, 10) +
    dplyr::lag(ExRet, 11) + dplyr::lag(ExRet, 12)), by = "Portfolio"]
characteristics <- characteristics[, `:=`(ExRet_lag1_sqrd, ExRet_lag1^2)]
characteristics <- characteristics[, `:=`(ExRet_lag2_sqrd, dplyr::lag(ExRet,
    2)^2), by = "Portfolio"]
characteristics <- characteristics[, `:=`(ExRet_lag312_sqrd, (dplyr::lag(ExRet,
    3) + dplyr::lag(ExRet, 4) + dplyr::lag(ExRet, 5) + dplyr::lag(ExRet, 6) +
    dplyr::lag(ExRet, 7) + dplyr::lag(ExRet, 8) + dplyr::lag(ExRet, 9) + dplyr::lag(ExRet,
    10) + dplyr::lag(ExRet, 11) + dplyr::lag(ExRet, 12))^2), by = "Portfolio"]

# keep data pertaining to months 13+
characteristics <- characteristics[complete.cases(characteristics)]
```

B. Decision Trees.

1. Using data from the beginning of your sample (now, first row should correspond to 196407) until 200912, estimate RandomForest as in the class notes.

a. Report the inputs you chose for the routine (number of trees, max nodes) and report a linear panel regression of excess returns on the predicted value, the latter obtained from the Random Forest predict function as in the notes. Cluster the standard errors by time to account for contemporaneous correlation across firms each month.

```
# we split the overall sample into training and test samples

# define training sample
ExRet_train <- as.matrix(characteristics[Month < 559, Month, ExRet])
lagged_rets_train <- as.matrix(characteristics[Month < 559, list(ExRet_lag1,
    ExRet_lag2, ExRet_lag312, ExRet_lag1_sqrd, ExRet_lag2_sqrd, ExRet_lag312_sqrd)])

# define test sample
ExRet_test <- as.matrix(characteristics[Month > 528, Month, ExRet])
lagged_rets_test <- as.matrix(characteristics[Month > 528, list(ExRet_lag1,
    ExRet_lag2, ExRet_lag312, ExRet_lag1_sqrd, ExRet_lag2_sqrd, ExRet_lag312_sqrd)])

# estimate the random forest I use maxnodes of 10 and ntree of 500; higher
# maxnodes leads to too much overfitting; trial and error is a good approach
RF <- randomForest(lagged_rets_train, ExRet_train[, 2], ntree = 500, maxnodes = 10)

# in sample test using clustered standard errors
RF_in_sample_pred <- predict(RF, lagged_rets_train)
RF_in_sample_test <- felm(ExRet_train[, 2] ~ RF_in_sample_pred | 0 | 0 | ExRet_train[,
    1])
summary(RF_in_sample_test)
```

```
##
## Call:
##    felm(formula = ExRet_train[, 2] ~ RF_in_sample_pred | 0 | 0 |      ExRet_train[, 1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -391.98 -135.81   -0.27  135.96  285.90
##
## Coefficients:
##                    Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)       -514.7784      47.9615  -10.73   <2e-16 ***
## RF_in_sample_pred    2.8016       0.1678   16.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 157.3 on 75274 degrees of freedom
## Multiple R-squared(full model): 0.002889   Adjusted R-squared: 0.002876
## Multiple R-squared(proj model): 0.002889   Adjusted R-squared: 0.002876
## F-statistic(full model, *iid*):218.1 on 1 and 75274 DF, p-value: < 2.2e-16
## F-statistic(proj model): 278.8 on 1 and 57962 DF, p-value: < 2.2e-16
```

b. Now, in the true out-of-sample period 201001-201612, get predicted values using the tree you estimated in a. Run a linear panel regression of realized returns on the predicted values, again clustering standard errors by time.

```
# out of sample test

RF_out_of_sample_pred <- predict(RF, lagged_rets_test)
RF_out_of_sample_test <- felm(ExRet_test[, 2] ~ RF_out_of_sample_pred | 0 |
    0 | ExRet_test[, 1])
summary(RF_out_of_sample_test)
```

```
##
## Call:
##    felm(formula = ExRet_test[, 2] ~ RF_out_of_sample_pred | 0 |      0 | ExRet_test[, 1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.274 -25.906  -0.093  25.847  57.495
##
## Coefficients:
##                       Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)           614.2054      18.4473  33.295   <2e-16 ***
## RF_out_of_sample_pred  -0.1196       0.0645  -1.854   0.0638 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29.73 on 14212 degrees of freedom
## Multiple R-squared(full model): 0.0003588   Adjusted R-squared: 0.0002885
## Multiple R-squared(proj model): 0.0003588   Adjusted R-squared: 0.0002885
## F-statistic(full model, *iid*):5.101 on 1 and 14212 DF, p-value: 0.02393
## F-statistic(proj model): 3.436 on 1 and 12308 DF, p-value: 0.0638
```

c. Run Fama-MacBeth regressions in the true out of sample periods to get the Sharpe ratio and t-statistic of the implied trading strategy that uses the predicted value as the signal.

```
# put precdicted variable and predictor in a single data table
fmb.data.RF <- as.data.table(cbind(ExRet_test, RF_out_of_sample_pred))

# run Fama-MacBeth using the standard approach we saw in week 1
port_ret_RF = fmb.data.RF[, .(lambda = felm(ExRet ~ RF_out_of_sample_pred)$coef[2]),
    by = Month]
fm_output_RF = list(MeanReturn = mean(port_ret_RF$lambda), StdReturn = sqrt(var(port_ret_RF$lambda)),
    SR_Return = mean(port_ret_RF$lambda)/sqrt(var(port_ret_RF$lambda)), tstat_MeanRet = sqrt(1 +
        631 - 545) * mean(port_ret_RF$lambda)/sqrt(var(port_ret_RF$lambda)))
fm_output_RF
```

```
## $MeanReturn
## [1] -4.04407e-05
##
## $StdReturn
## [1] 0.001237372
##
## $SR_Return
```

```
## [1] -0.03268274
##
## $tstat_MeanRet
## [1] -0.3048443
```

2. Repeat question B1, but instead use XGBoost. Try and report results for the following four parameter configurations in your exercise: {eta = 0.1, maxdepth = 1}, {eta = 0.1, maxdepth = 6}, eta = 0.3, maxdepth = 1}, {eta = 0.3, maxdepth = 6}. For each run, set nrounds using the cross-validation exercise in the notes.

```
# I report the solution for only one combination of eta and
# maxdepth;applying the methodology to different combinations is a purely
# mechanical exercise

params <- list(booster = "gbtree", objective = "reg:linear", eta = 0.1, gamma = 0,
    max_depth = 6)

# define the xgb.DMatrix as in the lecture notes; lagged_rets_train and
# lagged_rets_test are defined in the previous question
xgb_train <- xgb.DMatrix(data = lagged_rets_train, label = ExRet_train[, 2])

# xgb.cv is the cross validation procedure described in the lecture notes
xgbcv <- xgb.cv(params = params, data = xgb_train, nfold = 10, nrounds = 100)
cv_nrounds = which.min(xgbcv$evaluation_log$test_rmse_mean)
xgb_optb <- xgboost(params = params, data = xgb_train, nround = cv_nrounds)
```

Tests using the predicted values of XGBoost are presented below.

```
# in sample regression
xgb_pred_in_sample <- predict(xgb_optb, lagged_rets_train)
xgb_test_in_sample <- felm(ExRet_train[, 2] ~ xgb_pred_in_sample | 0 | 0 | ExRet_train[,
    1])
summary(xgb_test_in_sample)
```

```
##
## Call:
##    felm(formula = ExRet_train[, 2] ~ xgb_pred_in_sample | 0 | 0 |      ExRet_train[, 1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -427.16 -130.52   -1.21  129.81  410.10
##
## Coefficients:
##                      Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)        -876.05324     17.52578  -49.99   <2e-16 ***
## xgb_pred_in_sample    4.09828      0.06178   66.33   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 153.7 on 75274 degrees of freedom
## Multiple R-squared(full model): 0.04844   Adjusted R-squared: 0.04843
## Multiple R-squared(proj model): 0.04844   Adjusted R-squared: 0.04843
## F-statistic(full model, *iid*): 3832 on 1 and 75274 DF, p-value: < 2.2e-16
## F-statistic(proj model):  4400 on 1 and 57962 DF, p-value: < 2.2e-16
```

```r
# out of sample tests define true out of sample data
xgb_test <- xgb.DMatrix(data = lagged_rets_test, label = ExRet_test[, 2])

# get predicted values in the true out of sample period
xgb_pred_out_of_sample <- predict(xgb_optb, lagged_rets_test)

# run out-of-sample regression analogous to those in B1
xgb_test_out_of_sample <- felm(ExRet_test[, 2] ~ xgb_pred_out_of_sample | 0 |
    0 | ExRet_test[, 1])
summary(xgb_test_out_of_sample)
```

```
##
## Call:
##    felm(formula = ExRet_test[, 2] ~ xgb_pred_out_of_sample | 0 |      0 | ExRet_test[, 1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -52.549 -25.915   0.015  25.955  51.388
##
## Coefficients:
##                          Estimate Cluster s.e. t value Pr(>|t|)
## (Intercept)            573.05882      8.29854  69.055   <2e-16 ***
## xgb_pred_out_of_sample   0.02447      0.02925   0.837    0.403
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29.73 on 14212 degrees of freedom
## Multiple R-squared(full model): 6.256e-05   Adjusted R-squared: -7.801e-06
## Multiple R-squared(proj model): 6.256e-05   Adjusted R-squared: -7.801e-06
## F-statistic(full model, *iid*):0.8891 on 1 and 14212 DF, p-value: 0.3457
## F-statistic(proj model): 0.7001 on 1 and 12308 DF, p-value: 0.4028
```

```r
# put precdicted variable and predictor in a single data table
fmb.data.XGB <- as.data.table(cbind(ExRet_test, xgb_pred_out_of_sample))

# run Fama-MacBeth using the standard approach we saw in week 1
port_ret_XGB = fmb.data.XGB[, .(lambda = felm(ExRet ~ xgb_pred_out_of_sample)$coef[2]),
    by = Month]
fm_output_XGB = list(MeanReturn = mean(port_ret_XGB$lambda), StdReturn = sqrt(var(port_ret_XGB$lambda))
    SR_Return = mean(port_ret_XGB$lambda)/sqrt(var(port_ret_XGB$lambda)), tstat_MeanRet = sqrt(1 +
        631 - 545) * mean(port_ret_XGB$lambda)/sqrt(var(port_ret_XGB$lambda)))
fm_output_XGB
```

```
## $MeanReturn
## [1] -1.947811e-06
##
## $StdReturn
## [1] 0.0003738875
##
## $SR_Return
## [1] -0.005209617
##
## $tstat_MeanRet
## [1] -0.04859207
```

C. Asymptotic PCA

Using the same data, run the APCA routine over rolling 5-year windows through the sample, setting K = 5 (i.e., get 5 biggest PCs). I.e., your sample is the excess returns from 196307 to 201612 on the 138 portfolios. At each time t, have the APCA routine estimate the 5 factors using the last 60 months of data on the 138 portfolios. Record at each time t the loadings of each of the 138 portfolios on the 5 factors. Now, obtain the next month's out of sample return to these factors by for each factor taking the 138 loadings and multiplying them with next month's return to the 138 portfolios. Roll through the sample so you have a time series of "out-of-sample" returns for each of the 5 factors.

```
# we cannot have NaN in any columns if we are to implement APCA; we can
# either remove P11 or move the starting sample to 1969. I chose to move the
# starting point of the sample
french_port_rets <- as.data.table(french_port_rets[complete.cases(french_port_rets)])

# define a matrix in which we will save the factor loadings
Factor_loadings <- matrix(0, nrow = 69000, ncol = 5)

# implement APCA for rolling 5 year windows
for (i in 1:500) {
    pca_data_rolling <- french_port_rets[Month >= 72 + i & Month <= 131 + i]
    Factors <- apca(pca_data_rolling[, !"Month"], 5)
    for (k in 1:138) {
        for (j in 1:5) {
            Factor_loadings[(i - 1) * 138 + k, j] <- Factors$loadings[k, j]
        }
    }
}

# remove the first five year window from the portfolios: the remaining
# portfolios can be used to compute out-of-sample returns
french_port_rets <- french_port_rets[!c(1:60), !1]

# define a matrix which will be populated with the time series of out of
# sample returns
f_p_out_of_sample <- matrix(0, nrow = nrow(french_port_rets), ncol = 5)

# compute out of sample returns
for (l in 1:nrow(french_port_rets)) {
    f_p_out_of_sample[l, ] <- as.matrix(french_port_rets[l, ]) %*% Factor_loadings[((l -
        1) * 138 + 1):(l * 138), ]
}
```

a. What is the average annualized excess return, standard deviation, and Sharpe ratio of each of the factors. Also report the monthly first autocorrelation of the factors.

The statistics of interest are reported below. All definitions are standard.

```
# annualize mean returns, standard deviations and Sharpe ratios
f_p_out_of_sample <- as.data.table(f_p_out_of_sample)
mean_ret <- 12 * f_p_out_of_sample[, lapply(.SD, mean)]
sd_ret <- sqrt(12) * f_p_out_of_sample[, lapply(.SD, sd)]
Sharpe_ratio <- mean_ret/sd_ret
```

```r
# find autocorrelations
ac1 <- acf(f_p_out_of_sample[, 1], lag.max = 1, plot = FALSE)$acf
ac2 <- acf(f_p_out_of_sample[, 2], lag.max = 1, plot = FALSE)$acf
ac3 <- acf(f_p_out_of_sample[, 3], lag.max = 1, plot = FALSE)$acf
ac4 <- acf(f_p_out_of_sample[, 4], lag.max = 1, plot = FALSE)$acf
ac5 <- acf(f_p_out_of_sample[, 5], lag.max = 1, plot = FALSE)$acf

# display required statistics; first order autocorrelations only
mean_ret
```

```
##            V1           V2          V3          V4          V5
## 1: -0.1785579 -0.006076559 0.003684183 -0.03028142 -0.03636207
```

```r
sd_ret
```

```
##          V1        V2        V3        V4        V5
## 1: 1.024109 0.4923788 0.2735024 0.2161391 0.1724598
```

```r
Sharpe_ratio
```

```
##            V1          V2         V3         V4         V5
## 1: -0.1743545 -0.01234123 0.01347039 -0.1401015 -0.2108437
```

```r
as.data.table(cbind(ac1[2], ac2[2], ac3[2], ac4[2], ac5[2]))
```

```
##            V1         V2         V3            V4         V5
## 1: 0.05844194 0.02732326 -0.0147419 -0.0002419484 -0.1199029
```

b. What are the t-statistics of the average return estimates. Estimate the standard errors controlling for heteroscedasticity and one autocorrelation of the factors.

```r
# obtain the mean out of sample return
mean_ret_est <- f_p_out_of_sample[, lapply(.SD, mean)]

# find the HAC standard errors
se1 <- sqrt(vcovHAC(lm(f_p_out_of_sample$V1 ~ 1)))
se2 <- sqrt(vcovHAC(lm(f_p_out_of_sample$V2 ~ 1)))
se3 <- sqrt(vcovHAC(lm(f_p_out_of_sample$V3 ~ 1)))
se4 <- sqrt(vcovHAC(lm(f_p_out_of_sample$V4 ~ 1)))
se5 <- sqrt(vcovHAC(lm(f_p_out_of_sample$V5 ~ 1)))

se <- cbind(se1, se2, se3, se4, se5)

# find t stats
t_stats <- mean_ret_est/se
t_stats
```

```
##           V1          V2         V3         V4        V5
## 1: -1.114492 -0.07866683 0.08689681 -0.9036358 -1.524077
```

c. Given your results in a and b, what do you conclude about the use of these portfolios as hedges? E.g., do they affect average excess returns? Could anything go wrong in the rolling APCA you did in terms of interpreting the factors as risk factors? I have in mind a discussion of conditional versus unconditional expected returns here.

The factors have, over the sample, excess returns that unconditionally are not statistically significant. Thus, it seems that hedging out these factor risks is a good idea - you can reduce volatility (as Numeric is doing in their portfolio risk management) but not at the expense of mean returns. (Note: we are ignoring trading costs here). That said, the factors may conditionally have time-variation in expected returns. E.g., sometimes conditional expected returns are +2%, sometimes conditional expected returns are -2%. For instance, there may be factor momentum (some papers have documented this). If so, you should consider this in your hedging strategy.

d. You want to see if lagged return is a useful signal in a Fama-MacBeth regression (for predicting next month's return). However, you also want control for the loadings on the five factors you estimated in the APCA. Make sure you use the loadings in a rolling out-of-sample fashion - i.e.., so the Fama-MacBeth regressions indeed correspond to real-time tradeable portfolios. Explain your procedure (ie, what regressions you run and what their purpose is) and report the results.

```
# We need to remove all observations prior to the start of the 'APCA sample'
# (see above for discussion) and additional observations due to missing data
# generated by the 5-year windows.
APCA_test_data <- characteristics[Month >= 132]

# Use only lagged returns as a benchmark
FMB_benchmark = APCA_test_data[, .(lambda = felm(ExRet ~ ExRet_lag1)$coef[2]),
    by = "Month"]

fm_output_benchmark = list(MeanReturn = mean(FMB_benchmark$lambda), StdReturn = sqrt(var(FMB_benchmark$
    SR_Return = mean(FMB_benchmark$lambda)/sqrt(var(FMB_benchmark$lambda)),
    tstat_MeanRet = sqrt(500) * mean(FMB_benchmark$lambda)/sqrt(var(FMB_benchmark$lambda)))

fm_output_benchmark
```

```
## $MeanReturn
## [1] 0.06742763
##
## $StdReturn
## [1] 0.3252371
##
## $SR_Return
## [1] 0.2073184
##
## $tstat_MeanRet
## [1] 4.63578
```

```
# Merge and prepare data for Fama-MacBeth regressions
APCA_FMB_data <- as.data.table(cbind(APCA_test_data$Month, APCA_test_data$ExRet,
    APCA_test_data$ExRet_lag1, Factor_loadings))

# Rename columns
names(APCA_FMB_data)[1] <- "Month"
```

```r
names(APCA_FMB_data)[2] <- "ExRet"
names(APCA_FMB_data)[3] <- "Lag1Ret"
names(APCA_FMB_data)[4] <- "L1"
names(APCA_FMB_data)[5] <- "L2"
names(APCA_FMB_data)[6] <- "L3"
names(APCA_FMB_data)[7] <- "L4"
names(APCA_FMB_data)[8] <- "L5"

APCA_FMB = APCA_FMB_data[, .(lambda = felm(ExRet ~ Lag1Ret + L1 + L2 + L3 +
    L4 + L5)$coef[2]), by = "Month"]

fm_output_APCA = list(MeanReturn = mean(APCA_FMB$lambda), StdReturn = sqrt(var(APCA_FMB$lambda)),
    SR_Return = mean(APCA_FMB$lambda)/sqrt(var(APCA_FMB$lambda)), tstat_MeanRet = sqrt(500) *
        mean(APCA_FMB$lambda)/sqrt(var(APCA_FMB$lambda)))

fm_output_APCA
```

```
## $MeanReturn
## [1] 0.06568876
##
## $StdReturn
## [1] 0.3237022
##
## $SR_Return
## [1] 0.2029296
##
## $tstat_MeanRet
## [1] 4.537643
```