Problem Set 4 Solution

Denis Mokanov

April 26, 2019

Question 1: Automatic stock picking algorithm

```
# Clean workspace
rm(list = ls())

# Load packages
library(foreign)
library(data.table)
library(lfe)
library(glmnet)
library(Metrics)
library(ggplot2)

# Load data
StockRetAcct_DT = as.data.table(read.dta("StockRetAcct_insample.dta"))
```

a.

Download the data. The firm-level characteristics you will use are lnIssue, lnProf, lnInv, and lnME. For each of these four characteristics, create new, additional characteristics as the squared value of the original characteristic. Name the new characteristics the same as the orignal, but with a "2" at the end. For instance, for lnProf, the squared value should be lnProf2. Further, create additional characteristics by multiplying each characteristic with lnME (except for lnME itself, which you already have squared). To name these, add _ME at the end. Thus, lnProf interacted with lnME is named lnProf_ME. You should have now gone from 4 to 11 characteristics.

```
# Clean up data
StockRetAcct_DT[, `:=`(ExRet, exp(lnAnnRet) - exp(lnRf))]
StockRetAcct_DT = StockRetAcct_DT[, .(FirmID, year, ExRet, ff_ind, MEwt, lnIssue,
    lnProf, lnInv, lnME)]

# Create squared variables
list_of_squared = colnames(StockRetAcct_DT)[-(1:5)]
for (i in list_of_squared) {
    StockRetAcct_DT[, `:=`(paste0(i, "2"), get(i)^2)]
}

# Create interaction with ME variable
for (i in list_of_squared[-4]) {
    StockRetAcct_DT[, `:=`(paste0(i, "_ME"), get(i) * lnME)]
}

# Create a constant
StockRetAcct_DT[, `:=`(Constant, 1)]
```

(i)

For each year in the sample, cross-sectionally demean each of the 11 characteristics. That is, for each characteristic and each year subtract the average value of that characteristic across stocks. Then add as final characteristic a column of 1's to the dataset. This effectively inserts an intercept in the relation between the MVE portfolio weight and the characteristcs.

Next calculate the factor portfolio returns corresponding to each of these 12 characteristics, as explained at the end of the Topic 4 note. Note that the factor corresponding to the constant is simply an equal-weighted portfolio of all stocks (the "market"). The overall idea is that with this approach you have a market factor and long-short characteristics factors. We do not normalize characteristics to behave unit variance, as (as an empiricaly observation) the magnitude of the spread in characteristics across stocks is informative for the portfolio weights.

Calculate and report the factor sample means and sample variance-covariance matrix for these 12 annual factor returns, as well as the factors' sample Sharpe ratios.

```r
# Identify NA observations
standardized_StockRetAcct_DT = copy(StockRetAcct_DT)
for (i in colnames(standardized_StockRetAcct_DT)[-1]) {
    standardized_StockRetAcct_DT = standardized_StockRetAcct_DT[!is.na(get(i))]
}

# Create demeaned dataset
for (i in colnames(standardized_StockRetAcct_DT)[-c(1:5, 17)]) {
    standardized_StockRetAcct_DT[, `:=`(paste0(i), (get(i) - mean(get(i)))),
        by = year]
}

# Mark sample used for estimation
standardized_StockRetAcct_DT[, `:=`(Training_data, year <= 2004)]

# Factor portfolios
N_factors = ncol(standardized_StockRetAcct_DT) - 6
Factor_matrix = matrix(NA, nrow = length(unique(standardized_StockRetAcct_DT$year)),
    ncol = N_factors)
for (i in sort(unique(standardized_StockRetAcct_DT$year))) {
    Factor_matrix[(i - 1979), 1:N_factors] = t(data.matrix(standardized_StockRetAcct_DT[year ==
        i, -c("FirmID", "year", "ExRet", "ff_ind", "MEwt", "Training_data")])) %*%
        data.matrix(standardized_StockRetAcct_DT[year == i, .(ExRet)])
}
colnames(Factor_matrix) = colnames(standardized_StockRetAcct_DT[, -c("FirmID",
    "year", "ExRet", "ff_ind", "MEwt", "Training_data")])
rownames(Factor_matrix) = sort(unique(standardized_StockRetAcct_DT$year))

# Report factor sample means and sample variance-covariance matrix
Factor_avg_ret = colMeans(Factor_matrix[1:25, 1:N_factors])
Factor_var_matrix = var(Factor_matrix[1:25, 1:N_factors])
Factor_sharpe = colMeans(Factor_matrix[1:25, 1:N_factors])/(diag(var(Factor_matrix[1:25,
    1:N_factors]))^0.5)
Factor_avg_ret
```

```
##     lnIssue      lnProf       lnInv        lnME    lnIssue2     lnProf2
##  -11.214723   13.299770  -16.991948  -10.565218   -9.202694   -9.971779
##      lnInv2       lnME2  lnIssue_ME   lnProf_ME    lnInv_ME    Constant
##  -22.827767 -290.790558 -152.064519  172.201583 -234.065558  166.326097
```

```
Factor_var_matrix
```

```
##                lnIssue      lnProf       lnInv         lnME   lnIssue2
## lnIssue       640.7192   -994.5461    624.9896     496.1655   352.6523
## lnProf       -994.5461   2071.3625   -868.2150   -1949.0740  -474.5086
## lnInv         624.9896   -868.2150    880.9365     742.7593   360.2347
## lnME          496.1655  -1949.0740    742.7593   10435.7774   203.9477
## lnIssue2      352.6523   -474.5086    360.2347     203.9477   269.9992
## lnProf2      1383.7085  -2983.0591   1141.9666    3028.4673   683.3212
## lnInv2        953.7910  -1443.2588   1297.8309    1424.3177   565.2194
## lnME2       15354.3826 -58530.0579  22918.9028  297633.4683  6705.9302
## lnIssue_ME   8972.6926 -14087.2883   8842.0190    8364.6787  4920.6331
## lnProf_ME  -13410.7773  27638.2922 -11712.5300  -24172.1252 -6387.3214
## lnInv_ME     9344.9478 -13498.7708  12897.0435   13392.4044  5344.2368
## Constant     1840.1187   -369.8943   2591.4172   -9846.1339  1285.0342
##                lnProf2      lnInv2       lnME2   lnIssue_ME   lnProf_ME
## lnIssue      1383.7085    953.7910    15354.38     8972.693  -13410.777
## lnProf      -2983.0591  -1443.2588   -58530.06   -14087.288   27638.292
## lnInv        1141.9666   1297.8309    22918.90     8842.019  -11712.530
## lnME         3028.4673   1424.3177   297633.47     8364.679  -24172.125
## lnIssue2      683.3212    565.2194     6705.93     4920.633   -6387.321
## lnProf2      4495.3851   1903.9075    91094.52    19584.810  -39726.708
## lnInv2       1903.9075   1987.7943    43495.22    13565.461  -19397.242
## lnME2       91094.5206  43495.2240  8522765.40   255411.705 -729700.060
## lnIssue_ME  19584.8101  13565.4613   255411.71   125976.333 -189731.988
## lnProf_ME  -39726.7085 -19397.2416  -729700.06  -189731.988  369318.567
## lnInv_ME    17889.2510  19207.8273   407897.76   132615.849 -181628.915
## Constant      165.9178   3246.0576  -255818.13    24152.449   -8014.256
##               lnInv_ME    Constant
## lnIssue       9344.948   1840.1187
## lnProf      -13498.771   -369.8943
## lnInv        12897.044   2591.4172
## lnME         13392.404  -9846.1339
## lnIssue2      5344.237   1285.0342
## lnProf2      17889.251    165.9178
## lnInv2       19207.827   3246.0576
## lnME2       407897.759 -255818.1326
## lnIssue_ME  132615.849  24152.4486
## lnProf_ME  -181628.915  -8014.2557
## lnInv_ME    189957.250  35318.2211
## Constant     35318.221  88995.8396
```

```
Factor_sharpe
```

```
##     lnIssue      lnProf       lnInv        lnME    lnIssue2     lnProf2
## -0.44305199  0.29222414 -0.57249390 -0.10342274 -0.56005894 -0.14872679
##      lnInv2       lnME2  lnIssue_ME   lnProf_ME    lnInv_ME    Constant
## -0.51200914 -0.09960704 -0.42843349  0.28335871 -0.53704364  0.55753938
```

(ii)

Next, you are to use the Elastic Net procedure (alpha = 0.5 in glmnet) to estimate the b coefficients. Here, we cannot use the pre-programmed cross-validation procedure in cv.glmnet (so, use glmnet to estimate elastic net). The reason is that the right and left hand side variables depend on the sample. You are to run a cross-sectional regression of average returns to the factors on the covariances of each factor with itself and the other factors (see slide 48 in Topic 4). A 5-fold cross-validation would tell you to first find the sample factor averages and sample factor covariance matrix in a 20-year subperiod, and then see how well the estimated b coefficients do in the 5-year out of sample period. In the out of sample period, the average returns are the 5-year average factor returns for this period and the covariances are the 5-year covariances in this period. Thus, due to the combination of time series info (average returns and sample covariance matrix) and the cross-sectional regression, our setting is a little more complicated than the standard cv.glmnet code.

```
# Run 5 seperate elastic net regressions
in_sample_Factor_matrix = Factor_matrix[1:25, 1:N_factors]
for (i in 1:5) {
    # Define sample
    Factor_sample_avg_ret = colMeans(in_sample_Factor_matrix[-((5 * (i - 1) +
        1):(5 * i)), ])
    Factor_sample_var_matrix = var(in_sample_Factor_matrix[-((5 * (i - 1) +
        1):(5 * i)), ])

    # Run elastic net
    assign(paste0("Elastic_net_", i), glmnet(Factor_sample_var_matrix, Factor_sample_avg_ret,
        family = "gaussian", alpha = 0.5, standardize = TRUE))

    # Define a range of lambdas
    if (i == 1) {
        S_min = min(Elastic_net_1$lambda)
        S_max = max(Elastic_net_1$lambda)
    } else {
        S_min = min(S_min, get(paste0("Elastic_net_", i))$lambda)
        S_max = max(S_max, get(paste0("Elastic_net_", i))$lambda)
    }
}

# Predict factor returns and compare
range_of_lambda = seq(S_min - 1, S_max + 1, 0.01)
MSE_lambda = matrix(NA, nrow = 6, ncol = length(range_of_lambda))
for (i in 1:5) {
    b_matrix = as.matrix(predict(get(paste0("Elastic_net_", i)), type = "coef",
        s = range_of_lambda))[-1, ]
    for (j in 1:length(range_of_lambda)) {
        MSE_lambda[i, j] = mse(b_matrix[, j] %*% var(in_sample_Factor_matrix[(5 *
            (i - 1) + 1):(5 * i), ]), colMeans(in_sample_Factor_matrix[(5 *
            (i - 1) + 1):(5 * i), ]))
    }
}
MSE_lambda[6, ] = colMeans(MSE_lambda[1:5, ])
Best_lambda = range_of_lambda[which(MSE_lambda[6, ] == min(MSE_lambda[6, ]),
    arr.ind = TRUE)]
Best_lambda
```

```
## [1] 22.38716
```

```r
# Calculate b_vector for Best_lambda
b_vector = glmnet(Factor_var_matrix, Factor_avg_ret, family = "gaussian", alpha = 0.5,
    standardize = TRUE, lambda = Best_lambda)$beta
b_vector
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## lnIssue    -2.751982e-03
## lnProf      .
## lnInv      -2.602896e-03
## lnME        .
## lnIssue2   -8.773452e-03
## lnProf2     .
## lnInv2     -4.786693e-04
## lnME2       .
## lnIssue_ME -1.262971e-04
## lnProf_ME   .
## lnInv_ME   -6.702161e-05
## Constant    1.964595e-04
```

(iii)

With the final b-vector in hand, calculate the out-of-sample average return, standard deviation, and Sharpe ratio for the corresponding estimated "ex ante" MVE portfolio with return b'F_t in the period 2005-2014.

```r
b_vector = data.matrix(b_vector)
Out_of_sample_ret = Factor_matrix[-(1:25), ] %*% b_vector
mean(Out_of_sample_ret)  # Mean
```

```
## [1] 0.06117857
```

```r
sd(Out_of_sample_ret)  #Standard deviation
```

```
## [1] 0.09430877
```

```r
mean(Out_of_sample_ret)/sd(Out_of_sample_ret)  # Sharpe ratio
```

```
## [1] 0.648705
```

(iv)

Plot the cumulative return on this portfolio relative to that on the market (get market return using the value-weights in the sample, MEwt) over the 2005-2014 period, where you normalize the "MVE" portfolio's standard deviation to be the same as the market over this period. Compare. Note that one should really redo the estimation each year to get proper out of sample results that would mimick what you would do in the real world. Also, you could experiment in the in-sample cross-validation with different values for alpha to see what works best.

```r
# I define market as the universe of stocks available to me in the previous
# parts of the problem

# Throw out observations from StockRetAcct_DT that I threw out in
# standardized_StockRetAcct_DT
for (i in colnames(StockRetAcct_DT)[-(1)]) {
    StockRetAcct_DT = StockRetAcct_DT[!is.na(get(i))]
```

```
}

# Calculate 'market' return
StockRetAcct_DT[, `:=`(MEwt_adj, MEwt/sum(MEwt)), by = year]
MktRet = StockRetAcct_DT[year > 2004, .(Mkt_Ret = sum(ExRet * MEwt_adj)), by = year]
setorder(MktRet, year)
mean(MktRet$Mkt_Ret)  # Mean
```

```
## [1] 0.08122235
```

```
sd(MktRet$Mkt_Ret)  # Standard deviation
```

```
## [1] 0.1831969
```

```
mean(MktRet$Mkt_Ret)/sd(MktRet$Mkt_Ret)  # Sharpe ratio
```

```
## [1] 0.443361
```

```
# Scale MVE portfolio to match market
Scaled_Out_of_sample_ret = data.table(Out_of_sample_ret * sd(MktRet$Mkt_Ret)/sd(Out_of_sample_ret))
setnames(Scaled_Out_of_sample_ret, "s0", "Elastic_Ret")

# Calculate cumulative returns
Scaled_Out_of_sample_ret[, `:=`(Cum_Ret, cumprod(1 + Elastic_Ret))]
MktRet[, `:=`(Cum_Ret, cumprod(1 + Mkt_Ret))]


# Plot
qplot(MktRet$year, MktRet$Cum_Ret, geom = "line", xlab = "Year", ylab = "Excess return",
    color = I("blue"), size = I(1.5), main = "Value-weighted portfolio (blue) vs. MVE portfolio (red)") +
    geom_line(aes(y = Scaled_Out_of_sample_ret$Cum_Ret), color = I("red"), size = I(1.5)) +
    theme_bw()
```