Thesis

Jesse Keränen

11/4/2023

Loading the data that was cleaned and made tidy in file "Data_collection.Rmd".

```
load(file = "Data/data.Rdata")
set.seed(1) # Initialize a pseudorandom numbers
```

Instead of using raw accounting data, usually some kind of ratios are used in anomaly literature. In the next chunck stock characteristics used in this study are calculated.

```
independend_variables <- c("L.CashToAsset", "L.CapitalTurnOver","L.INV", "L.BEME", "L.CashFlowToPrice",

data[, CashToAsset := WC02001 / WC02999, by = Company]

data[, CapitalTurnOver := WC01001 / shift(WC02999), by = Company]

data[, BE := WC03501 + WC03263, by = Company]
data[, BEME := BE / DEC.MV, by = Company]

data[, BEME.M := BE / L.MV]

data[, CashFlowToPrice := WC04860 / DEC.MV, by = Company]

data[, Leverage := (WC02999 - WC03501) / DEC.MV, by = Company]

data[, SalesToPrice := WC01001 / DEC.MV, by = Company]

data[, EP := WC01551 / DEC.MV, by = Company]

data[, ReturnOnAssets := WC01551 / L.WC02999, by = Company]

data[, ReturnOnEquity := WC01551 / shift(BE), by = Company]

data[, MOM7 := shift(RI, 7, type = "lag")/shift(RI, 12, type = "lag") - 1,
   by = Company]

data[, MOM12 := shift(RI, 2, type = "lag")/shift(RI, 12, type = "lag") - 1,
   by = Company]

data[, MOM24 := shift(RI, 2, type = "lag")/shift(RI, 24, type = "lag") - 1,
   by = Company]

data[, MOM2 := shift(MOM12)]
```

Idea of this study is to mimic actual investment decision setting. Therefore we use lagged variables in the regressions because this information would have been available for an investor in a given time. To make running regressions simpler we add for each variable used in the regression lagged version (monthly variables).

```
data[, `:=` (L.CashToAsset = shift(CashToAsset), L.CapitalTurnOver = shift(CapitalTurnOver), L.INV = shi

data <- data[is.finite(rowSums(data[, -c("Company", "Date", "Country")]))] # Remove infinite values
```

Next step is to run separate cross-sectional regression for each month. In "cross_sectional" function first these regressions are run for each data. Results of these regressions are stored in a list format together with the corresponding date to a data.table called "regression_results". To have results in readable and usable format another lambda function is used. This lambda function takes one row at (based on a date) the time from the "regression_results" data.table. Inside of the function the list stored to the second column of the "regression_results" is turned to a data.table and corresponding date is added as a new column to the created data.table. Finally data.tables lambda function has returned are appended to one data.table. Ready data.table contains regression coefficients, standard deviations of coefficients and their t-values and p-values for each date and variable.

```
cross_sectional <- function(data, date, formula) {
  #browser()
  regression_results <- data[Date >= date, .(summary(lm(formula, data = .SD, na.action = na.omit))[4]),
  regression_results <- rbindlist(lapply(regression_results[, unique(Date)], function(x){
      temp <- as.data.table(regression_results[Date == x, V1], keep.rownames = "variable")
      temp[, Date := x]
      return(temp)
    }))
  return(regression_results)
}
formula <-as.formula(paste("RET", "~", paste(independend_variables, collapse = " + "))) # Regression fo

dates <- sort(data[Date >= "2000-01-31", unique(Date)]) # Get the first date of the dataset
FM_regressions <- cross_sectional(data, dates[1], formula)

# write_xlsx(data[Date == dates[250]], "test.xlsx")
```

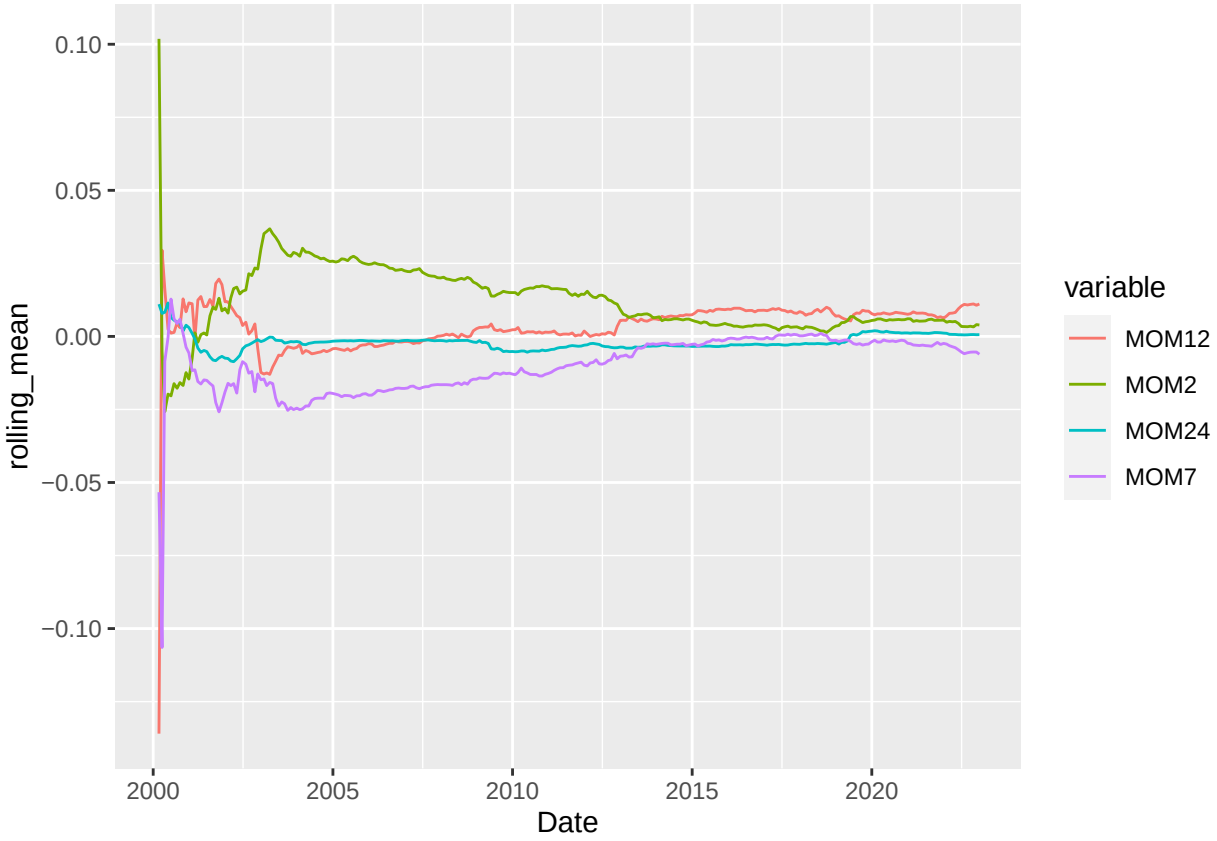
For final coefficients we need to calculate rolling means of the coefficients obtained from separate regressions. Lewellen uses average of 120 months. I want to follow method of Lewellen, but since our dataset is smalller we would lose almost half of the observations if we blindly would use rolling means of 120 months. Instead we use compromise and start from rolling window of 1 and increase the window until we reach 120 months. Then we will continue with window of 120 months. This will result hihgly volatile coefficients at the beginning, but I still find it better than wasting big portion of the dataset.

Finally rolling means are shifted one step for each variable. This is done because we need to exclude for each month most recent regression from the means. E.g. for date "2000-01-31" to derive the expected return we can take directly the characteristics, because they were already lagged earlier. Our logic for calculation of the mean regression coefficients for "2000-01-31" will consider regression using "2000-01-31" and up to 120 previous regressions. But when investor would make prediction of return "2000-01-31" return for corresponding time is not yet available and therefore it must be excluded.

```
rolling_window <- 120
setorder(FM_regressions, Date)
FM_regressions[, rolling_mean := frollmean(Estimate, n = c(seq.int(rolling_window), rep(rolling_window,
              length(.SD[, Estimate])-rolling_window)), na.rm = TRUE, align = "right", adaptive = T), by
FM_regressions[, rolling_mean := shift(rolling_mean), by = variable]
```

```
ggplot(FM_regressions[variable %in% c("MOM12", "MOM7", "MOM24", "MOM2")], aes(Date, rolling_mean, color=
  geom_line()
```

```
## Warning: Removed 4 rows containing missing values (`geom_line()`).
```



Finally we can derive the expected returns by multiplying the mean regression coefficients by the most recent available stock characteristics.

```
variables <- c("Date", "Company", independend_variables)

help_dt <- data[, ..variables]
help_dt[, `(intercept)` := 1]
help_dt <- melt(help_dt, id.vars = c("Date", "Company"))
help_dt <- merge(help_dt, FM_regressions, by = c("Date", "variable"))
help_dt[, product := value * rolling_mean, by = c("Company", "Date", "variable")]
help_dt <- help_dt[, .(FM.EXP.RET = sum(product)), by=c("Company", "Date")]

data <- merge(data, help_dt, by = c("Company", "Date"))

data[!is.na(FM.EXP.RET), summary(lm(RET ~ FM.EXP.RET))]
```

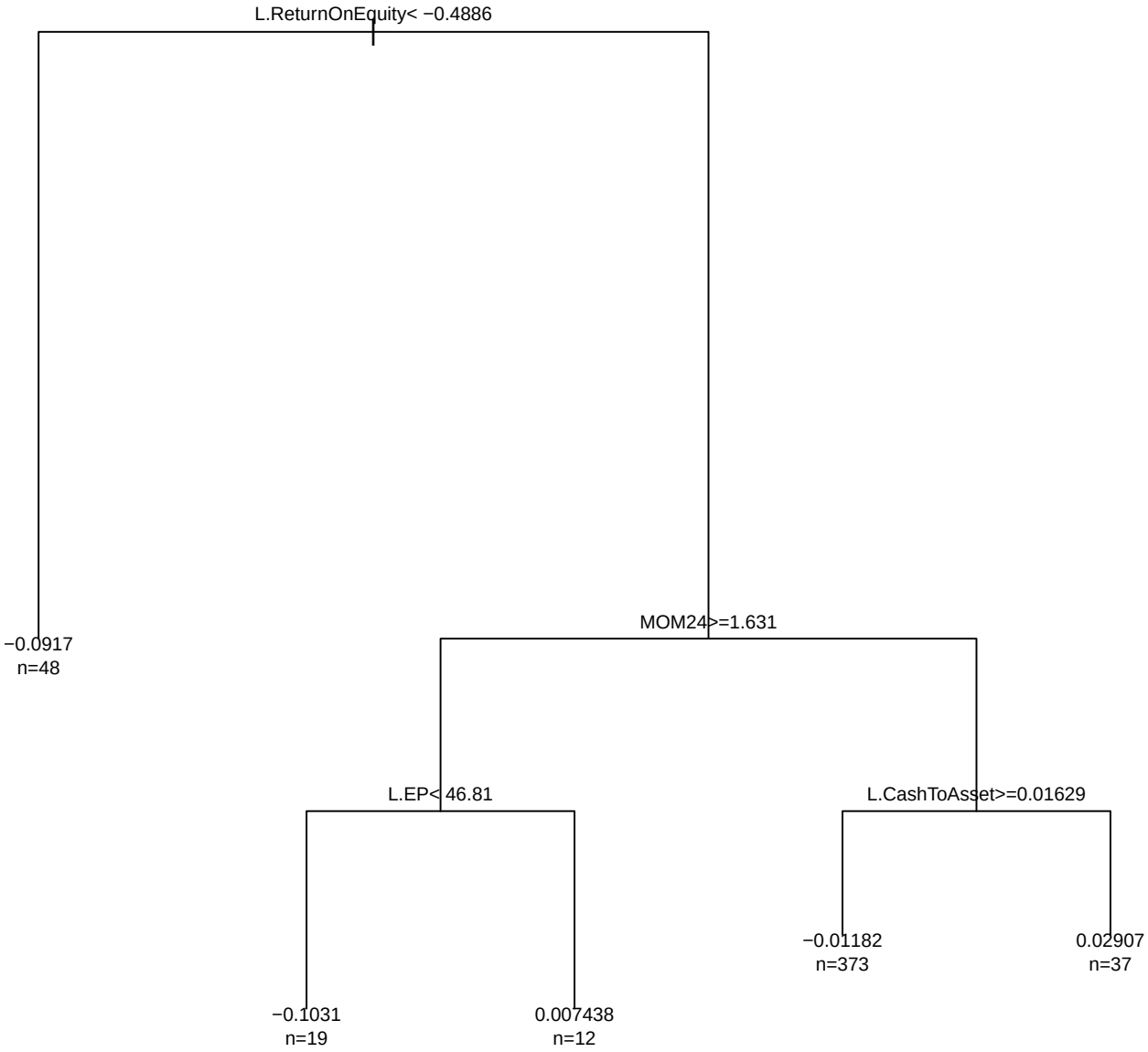
```
##
## Call:
## lm(formula = RET ~ FM.EXP.RET)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.64854 -0.06304 -0.00759  0.05440  1.22300
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0077450  0.0003113  24.876  < 2e-16 ***
## FM.EXP.RET  0.0306845  0.0070725   4.339 1.44e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1299 on 174967 degrees of freedom
## Multiple R-squared:  0.0001076,  Adjusted R-squared:  0.0001019
## F-statistic: 18.82 on 1 and 174967 DF,  p-value: 1.435e-05
```

```
library(rpart)

# Example Random Forest tree, just for illustration purposes
example_tree <- rpart(formula, method = "anova", data = na.omit(data[Date == as.POSIXct("2004-07-30", t
                                                                     control = rpart.control(minsplit = 2000
example_tree <- prune.rpart(example_tree, cp = 0.02)

plot(example_tree)
text(example_tree, use.n = TRUE, cex = .7)
```




```
data[, RF.EXP.RET := 0]
trainings <- data[Date > dates[50] & month(Date) == 7, unique(Date)]

grid <- expand.grid(dates = trainings, ntree = 200, mtry = c(9, 11, 14), maxnodes = c(10, 20, 30), mse =
grid <- setorder(as.data.table(grid), dates)
#grid[, `:=` (ntree = as.numeric(ntree), mtry = as.numeric(mtry))]


random_forest <- function(dt, grid, date_index, formula, independend_variables) {
  #browser()
  mse_lowest <- .Machine$integer.max
  for (j in 1:9) {
    index <- (date_index - 1) * 9 + j
    rf <- randomForest(formula, data = dt[Date < trainings[date_index]], ntree = as.numeric(grid[i, 2])
                       mtry = as.numeric(grid[i, 3]), maxnodes = 30)
    mse <- dt[Date >= trainings[date_index] & Date < trainings[date_index + 1],
              sum((RET - predict(rf, .SD[, ..independend_variables]))^2)]
```

```
    #browser()
    grid[index , 5] <- mse
    if (mse < mse_lowest) {
      dt[Date >= trainings[date_index + 1] & Date < trainings[date_index + 2],
         RF.EXP.RET := predict(rf, .SD[, ..independend_variables])]
    }
    mse_lowest <- ifelse(mse < mse_lowest, mse, mse_lowest)
    print(index)
  }
  return(list(dt, grid))
}

for (i in seq(1, length(trainings) - 2)) {
  #lista <- random_forest(data, grid, i, formula, independend_variables)
  #data <- lista[[1]]
  #grid <- lista[[2]]
}


#summary(data[ RF.EXP.RET != 0, lm(RET ~ RF.EXP.RET)])
```

```
#save(data2, file = "Data/EXP_RET.Rdata")
```