Statistical Factor Model Strategy

I would like to present in this short essay the equity strategy implemented in the R code enclosed. I will go through briefly the theory and I will explain all the logical steps with the help of some snapshots directly from the script.

Theory

Factor model (or better, time series factor model) is a well-known technique used in finance in order to predict equity returns:

$$R_{j,t} = \beta_{0,j} + \beta_{1,j} F_{1,t} + ... + \beta_{p,j} F_{p,t} + \varepsilon_{j,t}$$

Where $R_{j,t}$ is the return of the asset j at the time t and $F_{1,t} .... F_{p,t}$ are the variables, called *factor* or *risk factor*, which can be macroeconomic quantities, financial measures related to the company or different kind of market proxies. As far as this strategy is concerned, I have used a particular type of statistical method (or if you want machine learning method), which is factor analysis, in order to derive a set of factors loadings. According to WIkipedia, factor analysis is a statistical method used to describe variability among observed, correlated variables in terms of a potentially lower number of unobserved variables called factors. The following step is to use daily returns for the universe of stocks picked in order to run a cross sectional regression to find the time series of the hidden factors for every single day. Basically, every single day the cross sectional regression is performed and the betas of this regression are the values of the hidden factors for that specific day. The ultimate step is to run a time series regression using stock returns inside the lookback period and the daily factors just calculated before in order to find the beta of the regression. This last regression should be predictive, i.e. the return of the next day should be regress against the factors of the day before. The last day of data inside the lookback period is used in order to predict the return of the next day for all the stocks. The last step is to form a long/short portfolio based on the sorted returns just calculated. This model could be included inside the category of the statistical factor models where both the factor and factor loadings are not known (see *Statistics and data analysis for financial engineering*, Ruppert and Matteson, 2015 for more insights on statistical factor models). I will explain in details the strategy on the following chapter.

Strategy

I will describe the strategy step by step with the help of the code.

Before starting, I want to say that this model is just a prototype! Further studies are required in order to make it works for live trading

- **The first step is to select the universe of stocks that we want to use**. The source for the tickers to download is contained into the Static folder. I used some free Python API to download these tickers and the respective information. From these tickers, I

selected only the ones listed on the NASDAQ. *This choice is purely casual and should be investigated*.

```
# Upload all the tickers based on a list that I scraped on the web
# the csv contains the ticker, name, industry group and the correspondent
# sic code
StockTickers <- fread("./Static/TickersList.csv")

# Select only NASDAQ stocks
Financials <- StockTickers[exchange == 'NASDAQ']
# remove duplicates
ListTickersFinancials <- unique(Financials$ticker)
```

- **We are now downloading stock prices from yahoo finance**. I am using the "stocks" package in R since quantmod package is now defunct. This step is quite straightforward
- **After that we are cleaning the data in order to have a clean dataset for the analysis.** In this passage I calculated the log returns and I cleaned the dataset. I remove NAs and all the stocks which have a percentage of NAs in the data bigger than 1%. This cleaning is arbitrary (also this is another step that should be discussed) and is performed in order to prepare the dataset for the PCA analysis

```
# Function to clean the data
CleaningData <- function(StockDataframeFinancials){

  # find log return dataframe
  StockDataframeFinancialsLogRet <- StockDataframeFinancials[,.(Date,
                                                                 LogRet = c(NA,log(Price[-1]/Price[-length(Price)]))),
                                                             by=Ticker]

  # remove log ret equal to 0, NAs
  StockFinancialsLogRetClean <- StockDataframeFinancialsLogRet[LogRet != 0]
  StockFinancialsLogRetClean <- StockFinancialsLogRetClean[!is.na(LogRet)]

  # flip the dataset and remove stock where we have more than 1% of NAs
  FinancialsLogRetDcast <- as.data.frame(dcast.data.table(StockFinancialsLogRetClean,Date~Ticker,value.var = "LogRet"))
  # take only column with a % of NAs less than 1%
  FinancialsLogRetDcastClean <- FinancialsLogRetDcast[, -which(colMeans(is.na(FinancialsLogRetDcast)) > 0.01)]
  # replace NAs with 0
  FinancialsLogRetDcastClean[is.na(FinancialsLogRetDcastClean)] <- 0

  return(FinancialsLogRetDcastClean)
}
```

- **Perform PCA and factor analysis.** In order to run PCA and factor analysis (and also cross-sectional regression and predictive regression later on), I selected a lookback period of 600 days in the data. This lookback period is used to calculate PCA and perform factor analysis. Every day, the lookback is moved ahead of one day and PCA and factor analysis are performed again. As far as the PCA is concerned, the first 10 PCA are explaining almost 42% of the variance. I select the first 10 components for my analysis. **This selection is obviously another part that should be optimized**. After that, I used the factanal function in R to find the factor loadings for 10 factors and I built up the dataframe with all the factor loadings which will be used to run the cross sectional regression in the next step

```
# Function to find the factor loadings using factor analysis
GetFactorLoadings <- function(FinancialsLogRetDcastClean,LookBackPeriod){

  # take one year for the PCA calculation
  FinancialsLogRetTrainSet <- FinancialsLogRetDcastClean[j:(j+LookBackPeriod),]

  PCAFinancials <- prcomp(FinancialsLogRetTrainSet[,-1], center = T,scale. = T,retx = TRUE)
  summary(PCAFinancials)

  ## 10 PCA explain 42% of the variance, try to apply a factor analysis
  ## Now performing factor analysis in order to find the factors
  ## please note that the selection of 10 factors is arbitrary
  FactorAnalysis <- factanal(FinancialsLogRetTrainSet[,-1], factors = 10, rotation = "none")
  Factorloadings <- data.table(FactorAnalysis$loadings[,])
  Factorloadings[,Stock:= attr(FactorAnalysis$loadings,"dimnames")[[1]]]

  return(Factorloadings)
}
```

- **Cross-sectional regression part.** Now that we have the 10 factors loadings for all the stocks, we can run a cross-sectional regression against the log returns of the stocks for that particular day in order to find the factors. Basically the hidden factors are the betas of this cross-sectional regression. This step is repeated every day (factor analysis → cross sectional regression) in order to find the time series for the 10 hidden factors.

```
# Function to find the time series of returns
GetFactorsTimeSeries <- function(Row,Factorloadings){

  # melt the dataframe
  RowMelt <- melt.data.table(Row,id.vars = "Date",measure.vars = colnames(Row)[-1],
                             variable.name = "Stocks",value.name = "LogRet")

  # match the returns into the Factorloadings dataset
  CrossSectionalDf <- merge(Factorloadings,RowMelt,by.x = "Stock",by.y = "Stocks")
  # remove the dates and the name of the stocks
  CrossSectionalDfClean <- CrossSectionalDf[,-c("Date","Stock")]
  setcolorder(CrossSectionalDfClean, c("LogRet", setdiff(names(CrossSectionalDfClean), "LogRet")))
  # perform regression
  FactorsforTheDay <- lm(formula = LogRet ~ . +0,data = CrossSectionalDfClean)

  # create the dataframe with the factors
  FactorsDf <- data.table(Date = unique(CrossSectionalDf$Date),
                          FactorValue = FactorsforTheDay$coefficients,
                          FactorName = names(FactorsforTheDay$coefficients))
  # dcast the table and sort the columns
  FactorFinal <- dcast.data.table(FactorsDf,Date ~ FactorName, value.var = "FactorValue")
```

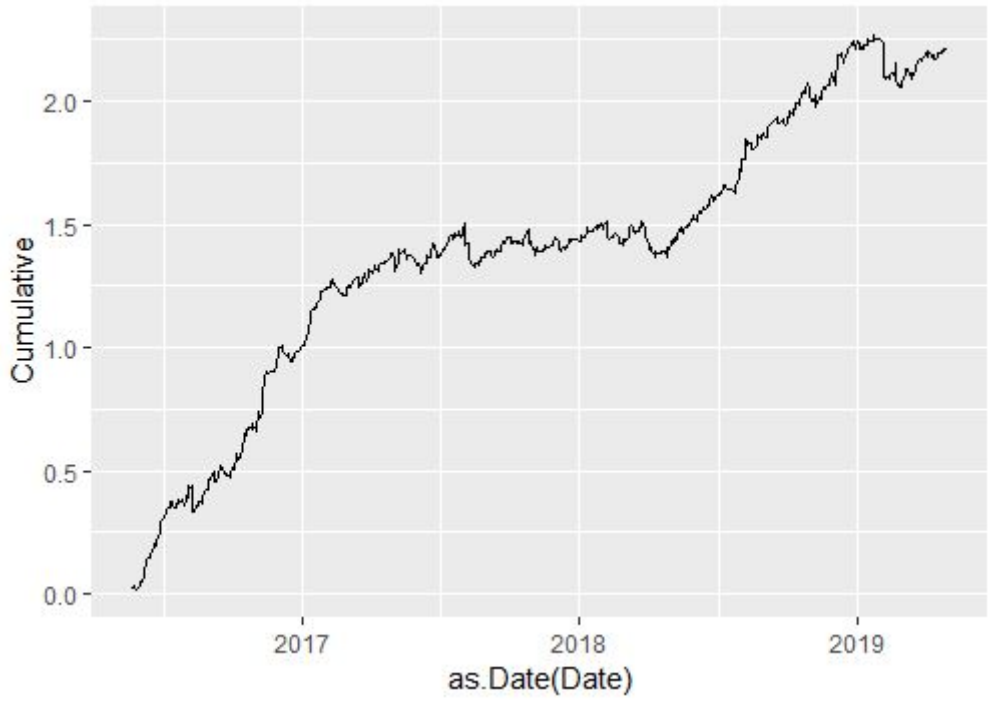
- **Predictive time series regression**. Now it is the time to run a predictive regression on the returns of every stock against the time series of our hidden factors that we just found. The regression is repeated for every stock and is predictive i.e. regress the factors value at time t against the returns at time t+1. In this way we can use the last values of the factors on the lookback period in order to calculate the predictive return (just multiplying the betas of the time series regression against the value of the factors). At the end, we have the predictive return for every stock on a given day

```
# Function for predicting regression
FindPredictingRegression <- function(FinancialsLogRetTrainSet,FactorDataframe){

  # find the dataframe with the stock to use (Use returns plus one and the factors dataframe)
  FinancialsLogRetTrainSet <- as.data.frame(FinancialsLogRetTrainSet)
  # dataframe with dates and the stock log returns
  Df <- FinancialsLogRetTrainSet[,c(1,i)]
  # Build up the dataframe for the regression, please remember that is a predictive time series regression
  # so I am removing the last factor observation and first return observation because I am running
  # the regression on the factor for the day t and the return on the day t+1
  NewDf <- data.frame(data.table(Df[-1,-1]),FactorDataframe[-nrow(FactorDataframe),-1])
  # regression
  RegressionDf <- lm(formula = V1 ~ . +0,data = NewDf)
  # predicted return -- I use the last factor observation in order to predict the return for this stock
  # for the next day
  PredRet <- sum(RegressionDf$coefficients * FactorDataframe[nrow(FactorDataframe),-1])
  # predicted return dataframe
  PredRetDf <- data.table(Date = Df$Date[length(Df$Date)],
                          Stock = colnames(Df)[2],
                          PredRet = PredRet)
```

- **The last part is related to find the strategy return for a particular day.** Once we have the dataframe with the predictive returns for a given day, we sort this dataframe and we pick the 10 best predictive returns and 10 worst predictive returns (the idea is to build up a long/short portfolio AQR style). The tickers for these returns are selected and used in order to find the actual return. After that, the returns of the 10 best stocks are added and divided by 10 (equally weighted) and the same is performed for the 10 worst stocks. The return of the portfolio is just the difference between the long leg and the short leg

```
# Get the strategy return for that day
GetStrategyReturn <- function(ListPredictedRet,FinancialsLogRetDcastClean,StockDataframeFinancials){

  # dataframe predicted returns for all the stocks
  PredRetFinalDf <- rbindlist(ListPredictedRet)
  # sort based on the return
  PredRetFinalDf <- PredRetFinalDf[order(PredRet,decreasing = T)]
  # Find the top 10 and last 10 stocks based on our predictive dataframe
  Top10 <- PredRetFinalDf[,head(.SD,10)]$Stock
  Last10 <- PredRetFinalDf[,tail(.SD,10)]$Stock

  FinancialsLogRetDcastClean <- data.table(FinancialsLogRetDcastClean)
  FinancialsLogClean <- melt.data.table(FinancialsLogRetDcastClean,id.vars = "Date",
                                        measure.vars = colnames(FinancialsLogRetDcastClean)[-1],
                                        variable.name = "Stocks",value.name = "LogRet")
  # calculate returns for the stocks
  StockDataframeFinancialsRet <- StockDataframeFinancials[,.(Date,
                                                             Ret = c(NA,(Price[-1] - Price[-length(Price)])/Price[-lengt
                                                          by=Ticker]

  # Get the actual returns based on the top and last 10 selected by the model
  # the idea is to go long the top 10 and short the last 10 (AQR style)
  LongReturn <- sum(StockDataframeFinancialsRet[Date == unique(PredRetFinalDf$Date)& Ticker%in%Top10]$Ret,na.rm = T)/le
  ShortReturn <- sum(StockDataframeFinancialsRet[Date == unique(PredRetFinalDf$Date)& Ticker%in%Last10]$Ret,na.rm = T)/

  # total return for the day
  Total <- LongReturn - ShortReturn
  # build the dataframe and append to the list
  TotalDf <- data.table(Date = unique(PredRetFinalDf$Date),StrategyReturn = Total)

  return(TotalDf)

}
```

If you repeat the process for every day in the sample (data start on “2014-01-03” and ends in “2019-04-30”) you can get the result of this strategy (performing backtest). Obviously the strategy used the first 600 days in order to estimate the model. Below you can find the equity curve



The strategy would have achieved a cumulated 200% over the last 3 years (from “2016-05-23” to “2019-04-30”). **Transaction costs are not included**.

I would like to repeat again that this model is just a prototype and several things should be investigated in order to make it works on live trading. All the parameters should be optimized (lookback period, numbers of factors used, number of stocks that you buy and sell on your long/short portfolio). Also, different types of forecasting techniques should be investigated in order to find the best one. For example, we could have used a support vector regression instead of a normal time series regression to perform prediction

London
01/08/2019