Rstudio main code

```r
# packages used
library("ggplot2")
library("tidyverse")
library("fpp3")
library("ggthemes")
library("psych")
library("jtools")
library("huxtable")
library("ggrepel")
library("directlabels")

# NOTE: some code for loading and cleaning data is not shown for brevity

#setting market cap limit
m_cap_lim = 100


# binding fundementals/accounting data from Nor, Swe, Dnk, and FIN
acc_nordic = rbind(acc_nor, acc_swe, acc_dnk, acc_fin)
acc_nordic = data.frame(
  f_year = acc_nordic$fyear,        # fiscal year
  f_date = acc_nordic$datadate,     # fiscal year end date
  a_currency = acc_nordic$curcd,    # accounting reporting currency
  name = acc_nordic$conm,           # company name
  gvkey = acc_nordic$gvkey,         # gvkey
  naics = acc_nordic$naics,         # NAICS code
  sales = acc_nordic$sale,          # sales
  COGS = acc_nordic$cogs,           # cost of goods sold
  ebitda = acc_nordic$ebitda,       # EBITDA
  ebit = acc_nordic$ebit,           # EBIT
  ni = acc_nordic$nicon,            # net income (used for computing EPS)
  cf_o = acc_nordic$oancf,          # cash flow from operations
  cf_i = acc_nordic$ivncf,          # cash flow from investing activities
  tot_assets = acc_nordic$at,       # total assets
  lt_debt = acc_nordic$dltt +
    acc_nordic$dlc,                 # total long-term debt
  book_equity = acc_nordic$ceq,     # common equity
  cash = acc_nordic$ch,             # cash
  cur_assets = acc_nordic$act,      # current assets
  cur_liab = acc_nordic$lct         # current liabilities
)

# binding securities market data from Nor, Swe, Dnk, and FIN
prices_nordic = rbind(prices_nor, prices_swe, prices_dnk, prices_fin)
prices_nordic = data.frame(
  p_date = prices_nordic$datadate,     # price date
  name = prices_nordic$conm,           # company name
  gvkey = prices_nordic$gvkey,         # gvkey
  iid = prices_nordic$iid,             # iid - issue identifier
  prirow = prices_nordic$prirow,       # Primary issue tag
  isin = prices_nordic$isin,           # isin
  issue_type = prices_nordic$tpci,     # issue type (0 = common, 1 =
preference, 2 = warrant...)
  exchange = prices_nordic$exchange,   # Nor, Swe, Dnk, or Fin
  price_close = prices_nordic$prccd,   # stock price at closing
  price_high = prices_nordic$prchd,    # stock price high
  price_low = prices_nordic$prcld,     # stock price low
  ajexdi = prices_nordic$ajexdi,       # adjustment for stock splits
  trfd = prices_nordic$trfd,           # adjustment for cash distributions
  trading_vol = prices_nordic$cshtrd, # daily trading volume (in stocks)
  shares = prices_nordic$cshoc,        # outstanding shares
  exchg = prices_nordic$exchg,         # stock exchange code
  dlrsn = prices_nordic$dlrsn,         # deletion reason (02=bankruptcy)
  p_currency = prices_nordic$curcdd    # currency for the issue
)

# formatting date variable
acc_nordic$f_date = parse_date_time(acc_nordic$f_date, orders = "%Y/%m/%d")
prices_nordic$p_date = parse_date_time(prices_nordic$p_date, orders =
"%Y/%m/%d")

# Agregating currency data to monthly from daily
fx$year_month = paste(year(fx$date), month(fx$date))
fx_monthly = fx %>% group_by(year_month, base) %>%
  summarize(date = last(date),
            quote = last(quote),
            units = last(units))
fx_monthly = fx_monthly %>% arrange(date)

# Agregating currency data to yearly from daily
fx$year = year(fx$date)
fx_annual = fx %>% group_by(year, base) %>%
  summarize(date = last(date),
            quote = last(quote),
            units = last(units))


#filtering out primary issues and common stock
prices_nordic$is_primary = rep(0, 7840348)
prices_nordic$is_primary[prices_nordic$iid == prices_nordic$prirow] = 1
prices_nordic_clean = prices_nordic %>% filter(is_primary == 1)
prices_nordic_clean = prices_nordic_clean %>% filter(issue_type == 0)

# Estimating bid-ask spreads
prices_nordic_clean = prices_nordic_clean %>% group_by(gvkey) %>%
  mutate(
    price_l_l1 = lag(price_low, n=1),
    price_h_l1 = lag(price_high, n=1),
    price_c_l1 = lag(price_close, n=1))

#    creating two day high and low prices
prices_nordic_clean = prices_nordic_clean %>% group_by(gvkey) %>%
  mutate(
    two_d_h = pmax(price_high, price_h_l1),
    two_d_l = pmin(price_low, price_l_l1))

#    creating overnight price increase and decrease
prices_nordic_clean = prices_nordic_clean %>% group_by(gvkey) %>%
  mutate(
    o.nght_pd = pmax(0, price_c_l1 - price_high),
    o.nght_pi = pmax(0, price_low - price_c_l1))
```

```
#    adjusting Low/High price
prices_nordic_clean = prices_nordic_clean %>% group_by(gvkey) %>%
  mutate(
    price_l_adj = price_low - o.nght_pi + o.nght_pd,
    price_h_adj = price_high - o.nght_pi + o.nght_pd )
prices_nordic_clean = prices_nordic_clean %>% group_by(gvkey) %>%
  mutate(
    two_d_h_adj = pmax(price_h_adj, price_h_l1),
    two_d_l_adj = pmin(price_l_adj, price_l_l1))

#    computing bid-ask spreads
prices_nordic_clean = prices_nordic_clean %>%
  mutate(
    b = (log(price_h_l1/price_l_l1))^2 + (log(price_h_adj/price_l_adj))^2 )
prices_nordic_clean = prices_nordic_clean %>%
  mutate(
    gamma = (log(two_d_h_adj/two_d_l_adj))^2 )
prices_nordic_clean = prices_nordic_clean %>%
  mutate(
    alpha = (sqrt(2*b) - sqrt(b)) / (3-2*sqrt(2)) -sqrt((gamma)/(3-
2*sqrt(2))) )
prices_nordic_clean = prices_nordic_clean %>%
  mutate(
    S = (2*(exp(alpha)-1)) / (1+exp(alpha)) )
prices_nordic_clean = prices_nordic_clean %>%
  mutate(
    bid_ask = pmax(0, S) )

prices_nordic_clean = prices_nordic_clean %>%
  mutate(
    y_m = paste(year(p_date), month(p_date)))

#    removing estimates of bid-ask when price_high = price_low = Price_close
prices_nordic_clean$one_trade = rep(0, 5898007)
prices_nordic_clean$one_trade[prices_nordic_clean$price_high ==
                              prices_nordic_clean$price_low &
                              prices_nordic_clean$price_low ==
                              prices_nordic_clean$price_close] = 1
prices_nordic_clean$bid_ask[prices_nordic_clean$one_trade == 1] = NA
prices_nordic_clean$bid_ask[lag(prices_nordic_clean$one_trade, n=1) == 1] =
NA

# agregating prices data
prices_nordic_clean = prices_nordic_clean %>% group_by(gvkey, y_m) %>%
  summarize(
    p_date = last(p_date),
    name = last(name),
    exchange = last(exchange),
    price_close = last(price_close),
    ajexdi = last(ajexdi),
    trfd = last(trfd),
    shares = last(shares),
    exchg = last(exchg),
    dlrsn = last(dlrsn),
    p_currency = last(p_currency),
    bid_ask_monthly.avg = mean(bid_ask, na.rm = T) )
prices_nordic_clean$bid_ask_monthly.avg[
  prices_nordic_clean$bid_ask_monthly.avg =="NaN"] = NA

# in the code below we adjust sales for currency changes
# creating a function that finds the mode (most frequent value)
calculate_mode <- function(x) {
  uniqx <- unique(na.omit(x))
  uniqx[which.max(tabulate(match(x, uniqx)))]}

# extracting the most frequent used repporting currency for each company
most_freq_currencies_nordic = acc_nordic %>%
  group_by(gvkey) %>%
  summarize(a_currency_most = calculate_mode(a_currency))

# creating a column for the most used currency
acc_nordic = acc_nordic %>% left_join(most_freq_currencies_nordic, by =
"gvkey")

# joining fx data for reported currency (in NOK)
fx_monthly01 = data.frame(
  f_year_month = fx_monthly$year_month,
  fx_date = fx_monthly$date,
  a_currency = fx_monthly$base,
  NOK_per_reported = fx_monthly$units)

acc_nordic$f_year_month = paste(year(acc_nordic$f_date),
month(acc_nordic$f_date))
acc_nordic = acc_nordic %>%
  left_join(fx_monthly01, by = c("f_year_month", "a_currency"))
acc_nordic$NOK_per_reported[acc_nordic$a_currency == "NOK"] = 1

# joining fx for most used reporting currency (in NOK)
fx_monthly02 = data.frame(
  f_year_month = fx_monthly01$f_year_month,
  fx_date = fx_monthly01$fx_date,
  a_currency_most = fx_monthly01$a_currency,
  NOK_per_most_reported = fx_monthly01$NOK_per_reported)

acc_nordic = acc_nordic %>% left_join(fx_monthly02,
                                      by = c("f_year_month",
"a_currency_most"))
acc_nordic$NOK_per_most_reported[acc_nordic$a_currency_most == "NOK"] = 1

# adding a factor that changes financials to most used currency
acc_nordic$currency_chg_adj = acc_nordic$NOK_per_reported /
acc_nordic$NOK_per_most_reported
acc_nordic$sales_crcy_adj = acc_nordic$sales * acc_nordic$currency_chg_adj
acc_nordic$ni_crcy_adj = acc_nordic$ni * acc_nordic$currency_chg_adj
acc_nordic$book_equity_crcy_adj = acc_nordic$book_equity *
acc_nordic$currency_chg_adj
acc_nordic$cf_o_crcy_adj = acc_nordic$cf_o * acc_nordic$currency_chg_adj
acc_nordic$cf_i_crcy_adj = acc_nordic$cf_i * acc_nordic$currency_chg_adj
acc_nordic$ebit_crcy_adj = acc_nordic$ebit * acc_nordic$currency_chg_adj


# joining FX-data with securities market data (monthly)
prices_nordic_clean$year_month = paste(year(prices_nordic_clean$p_date),
                                       month(prices_nordic_clean$p_date))
```

```
fx_monthly = data.frame(
  fx_date = fx_monthly$date,
  year_month = fx_monthly$year_month,
  p_currency = fx_monthly$base,
  nok_per_units = fx_monthly$units)
prices_nordic_clean = prices_nordic_clean %>%
  left_join(fx_monthly, by = c("year_month", "p_currency")) # joining fx data
prices_nordic_clean$nok_per_units[prices_nordic_clean$p_currency == "NOK"] =
1

# adjusting prices for stock-splits, dividends, and FX, and calculating
market capitalization
prices_nordic_clean$price_adj_1 =
  (prices_nordic_clean$price_close/prices_nordic_clean$ajexdi) *
  prices_nordic_clean$trfd
prices_nordic_clean$price_adj =
  prices_nordic_clean$price_adj_1 *
  prices_nordic_clean$nok_per_units
prices_nordic_clean$m_cap =
  (prices_nordic_clean$price_close * prices_nordic_clean$shares) /
  1000000
prices_nordic_clean$m_cap_NOK =
  prices_nordic_clean$m_cap * prices_nordic_clean$nok_per_units

# identifying bankrupt companies and setting returns to -100%
bankruptcies_nordic = prices_nordic_clean %>%
  group_by(gvkey) %>%
  filter(dlrsn == 2) %>%
  summarize(name = last(name), gvkey = last(gvkey),
            price_adj = last(price_adj), p_date = last(p_date))
bankruptcies_nordic$p_date_f1 = bankruptcies_nordic$p_date + days(x=20)
bankruptcies_nordic$price_bankrupt = rep(0, 47)
bankruptcies_nordic$year_month = paste(year(bankruptcies_nordic$p_date_f1),
                                       month(bankruptcies_nordic$p_date_f1))
bankruptcies_nordic = data.frame(
  gvkey = bankruptcies_nordic$gvkey,
  name = bankruptcies_nordic$name,
  p_date = bankruptcies_nordic$p_date_f1,
  price_adj = bankruptcies_nordic$price_bankrupt,
  year_month = bankruptcies_nordic$year_month)
prices_nordic_clean = bind_rows(prices_nordic_clean, bankruptcies_nordic)
prices_nordic_clean = prices_nordic_clean %>% arrange(by = p_date)
prices_nordic_clean = prices_nordic_clean %>% arrange(by = gvkey)

# preparing the securities market (prices) data to merge with fundemental
prices_nordic_clean$year = year(prices_nordic_clean$p_date)
prices_nordic_clean$month = month(prices_nordic_clean$p_date)
prices_nordic_merge = prices_nordic_clean
prices_nordic_merge = data.frame(
  price_date = prices_nordic_merge$p_date,
  f_year_month = prices_nordic_merge$year_month,
  gvkey = prices_nordic_merge$gvkey,
  m_cap = prices_nordic_merge$m_cap,
  m_cap_crcy = prices_nordic_merge$p_currency,
  m_cap_NOK = prices_nordic_merge$m_cap_NOK)
```

```r
# "balancing" the fundamental dataset such that missing rows = NA (important
for lagged variables)
time = data.frame(f_year = c(1987:2021))
time <- time %>% dplyr::arrange(f_year)
gvkeys_nordic = acc_nordic %>%
  group_by(gvkey) %>%
  summarize()
acc_nordic_fill = data.frame(gvkey = rep(gvkeys_nordic$gvkey, 35))
acc_nordic_fill <- acc_nordic_fill %>% dplyr::arrange(gvkey)
acc_nordic_fill$f_year = rep(time$f_year, length(gvkeys_nordic$gvkey))
acc_nordic_clean = acc_nordic_fill %>% left_join(acc_nordic,
                                                 by = c("gvkey", "f_year"))

# merging the agregated price data with the acounting data
nordic = acc_nordic_clean %>% left_join(prices_nordic_merge,
                                        by = c("gvkey", "f_year_month"))

# converting market cap to the same currency as financials
nordic$NOK_per_m_cap_crcy = nordic$m_cap_NOK / nordic$m_cap
nordic$currency_chg_m_cap = nordic$NOK_per_m_cap_crcy /
nordic$NOK_per_reported
nordic$m_cap_in_a_currency = nordic$m_cap * nordic$currency_chg_m_cap

# extracting relevant coloumns
nordic = nordic %>% select(
  gvkey, name, f_year, f_date, naics, sales, COGS, ebitda, ebit,
  ni, cf_o, cf_i, tot_assets, lt_debt, book_equity,
  cash, cur_assets, cur_liab, a_currency, a_currency_most, m_cap_crcy,
  currency_chg_adj, sales_crcy_adj, ni_crcy_adj, book_equity_crcy_adj,
  cf_o_crcy_adj, cf_i_crcy_adj, ebit_crcy_adj, price_date, fx_date.y,
  m_cap_NOK, m_cap, m_cap_in_a_currency, currency_chg_m_cap)

#changing fiscal year variable to the end of the fiscal period
nordic$f_year = year(nordic$f_date)

# generating variables
nordic = nordic %>%
  group_by(gvkey) %>%
  dplyr::mutate(sale_crcy_adj_lag10 = dplyr::lag(sales_crcy_adj,
                                                 n = 9,
                                                 default = NA)) %>%
  as.data.frame()

nordic$sales_g_10 =
  ((nordic$sales_crcy_adj / nordic$sale_crcy_adj_lag1) ^(1/9) - 1)
nordic$ebit_m = nordic$ebit / nordic$sales
nordic$solidity = (nordic$book_equity / (nordic$tot_assets - nordic$cash))
nordic$book_price = (nordic$book_equity) / nordic$m_cap_in_a_currency

# ------------------------------
# --- FIRST VERITAS BACKTEST ---
# ------------------------------

# creating 10 year moving variables (average and variance)
fv_nordic <- nordic %>%
  dplyr::group_by(gvkey) %>%
  dplyr::mutate(
    ni_10 = zoo::rollsum(ni_crcy_adj,
                         k = 10, fill = NA, align = "right"),
    b_e_10 = zoo::rollsum(book_equity_crcy_adj,
                          k = 10, fill = NA, align = "right"),
    cf_o_10 = zoo::rollsum(cf_o_crcy_adj,
                           k = 10, fill = NA, align = "right"),
    cf_i_10 = zoo::rollsum(cf_i_crcy_adj,
                           k = 10, fill = NA, align = "right"),
    margin_var_0 = zoo::rollapply(ebit_m,
                                  width = 10, align = "right",
                                  FUN = sd, fill = NA),
    sales_10 = zoo::rollsum(sales_crcy_adj,
                            k = 10, fill = NA, align = "right"),
    sales_3 = zoo::rollsum(sales_crcy_adj,
                           k = 3, fill = NA, align = "right"),
    ebit_10 = zoo::rollsum(ebit_crcy_adj,
                           k = 10, fill = NA, align = "right"),
    ebit_3 = zoo::rollsum(ebit_crcy_adj,
                          k = 3, fill = NA, align = "right"))

# computing averages
fv_nordic = fv_nordic %>% mutate(
  ROE_10 = ni_10 / b_e_10,
  CCR_10 = (cf_o_10 + cf_i_10) / ni_10,
  ebit_m_3 = ebit_3 / sales_3,
  ebit_m_10 = ebit_10 / sales_10 )

# computing "margin variance" and replacing negative values with NA
fv_nordic = fv_nordic %>% mutate(
  margin_var = margin_var_0 / ebit_m_10)
fv_nordic$margin_var = replace(fv_nordic$margin_var,
                               which(fv_nordic$margin_var < 0),
                               NA)

#creating cyclical phase variable
fv_nordic$cyclical_phase_raw =
  fv_nordic$ebit_m / (0.5*fv_nordic$ebit_m_10 + 0.5*fv_nordic$ebit_m_3)
fv_nordic$cyclical_phase = replace(fv_nordic$cyclical_phase_raw,
                                   which(fv_nordic$cyclical_phase_raw < 0),
                                   NA)

#generating a PE variable
fv_nordic$PE_raw = (fv_nordic$m_cap_in_a_currency) / (fv_nordic$ni)
fv_nordic$PE = replace(fv_nordic$PE_raw,
                       which(fv_nordic$PE_raw < 0),
                       NA)

#extracting the relevant data
fv_nordic = data.frame(
  f_year = fv_nordic$f_year,
  f_date = fv_nordic$f_date,
  name = fv_nordic$name,
  gvkey = fv_nordic$gvkey,
  m_cap_NOK = fv_nordic$m_cap_NOK,
  sales_g_10 = fv_nordic$sales_g_10,
  ROE_10 = fv_nordic$ROE_10,
  CCR_10 = fv_nordic$CCR_10,
  margin_var = fv_nordic$margin_var,
  solidity = fv_nordic$solidity,
  PE = fv_nordic$PE,
  cyclical_phase = fv_nordic$cyclical_phase,
  price_date = fv_nordic$price_date)

# filtering out companies with missing data
fv_nordic = fv_nordic %>% filter(sales_g_10 != "NA")
fv_nordic = fv_nordic %>% filter(sales_g_10 != "Inf")
fv_nordic = fv_nordic %>% filter(sales_g_10 != "NaN")
fv_nordic = fv_nordic %>% filter(PE != "NA")
fv_nordic = fv_nordic %>% filter(CCR_10 != "NA")
fv_nordic = fv_nordic %>% filter(ROE_10 != "NA")
fv_nordic = fv_nordic %>% filter(margin_var != "NA")
fv_nordic = fv_nordic %>% filter(margin_var != "NaN")
fv_nordic = fv_nordic %>% filter(cyclical_phase != "NA")
fv_nordic = fv_nordic %>% filter(cyclical_phase != "NaN")
fv_nordic = fv_nordic %>% filter(cyclical_phase != "Inf")

# Filtering out companies with m_cap below limit
fv_nordic = fv_nordic %>% filter(m_cap_NOK >= m_cap_lim)

# Counting the number of companies with sufficient data and plotting it by
year
fv_nordic_univ = fv_nordic %>%
  group_by(f_year)
N_nordic_count = data.frame(N = count(fv_nordic_univ, "year_q"))
N_nordic_count = data.frame(
  f_year = N_nordic_count[, 1],
  N = N_nordic_count[, 3])
N_nordic_count %>% ggplot() +
  geom_col(aes(x=f_year, y=N), fill="#009177", col="black") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  labs(title = "Number of companies in the Nordics with sufficient 10 year
historic
fundamental data and market capitalization above 100 MNOK") +
  theme_wsj()+
  theme(title = element_text(size = 12),
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  scale_x_continuous(breaks = c(2003:2021))

fv_nordic = fv_nordic %>% left_join(N_nordic_count, by = "f_year")

# saving the original dataset before screening on P/E
fv_nordic_orig = fv_nordic

# PE hardcap
fv_nordic = fv_nordic %>% filter(PE <= 200)

# adding a column for relative rank
# decreasing = F when high is good
# decreasing = T when low is good
fv_nordic = fv_nordic %>%
  group_by(f_year) %>%
  mutate(rank_sales_g = order(order(sales_g_10, decreasing=F)),
         rank_ROE = order(order(ROE_10, decreasing=F)),
         rank_CCR = order(order(CCR_10, decreasing=F)),
         rank_margin_var = order(order(margin_var, decreasing=T)),
         rank_solidity = order(order(solidity, decreasing=F)),
         rank_PE = order(order(PE, decreasing=T)),
         rank_cyclical_phase = order(order(cyclical_phase, decreasing=T))
  )

# calculating the weighted avg rank
fv_nordic$rank_total = (0.175 * fv_nordic$rank_sales_g +
                        0.175 * fv_nordic$rank_ROE +
                        0.100 * fv_nordic$rank_CCR +
                        0.175 * fv_nordic$rank_margin_var +
                        0.075 * fv_nordic$rank_solidity +
                        0.200 * fv_nordic$rank_PE +
                        0.100 * fv_nordic$rank_cyclical_phase)

# Top N highest values by group
nordic_top15 = fv_nordic %>% filter(f_year >= 2007) %>%
  arrange(desc(rank_total)) %>%
  group_by(f_year) %>%
  slice(1:15)

# creating returns variable
prices_nordic_clean = prices_nordic_clean %>% arrange(p_date)
prices_nordic_clean = prices_nordic_clean %>% arrange(gvkey)
prices_nordic_clean = prices_nordic_clean %>%
  group_by(gvkey) %>%
  dplyr::mutate(price_adj_lag1 = dplyr::lag(price_adj, n = 1, default = NA))
%>%
  as.data.frame()
prices_nordic_clean$return =
  (prices_nordic_clean$price_adj / prices_nordic_clean$price_adj_lag1) - 1
prices_nordic_clean$return[is.infinite(prices_nordic_clean$return)==T] = NA

# lagging the year variable so it matches end of june t+1 to end of june t+2
which is the holding period
prices_nordic_clean$year_match = ifelse(
  month(prices_nordic_clean$p_date) >= 7,
  year(prices_nordic_clean$p_date)-1,
  year(prices_nordic_clean$p_date)-2)

# creating year_match variable for the FV portfolio
nordic_top15$year_match = year(nordic_top15$price_date)

# extracting only the First-veritas winners
nordic_top15_monthly = prices_nordic_clean %>%
  inner_join(nordic_top15, by = c("year_match", "gvkey"))

# extracting relevant variable
FV_portfolio = data.frame(
  name = nordic_top15_monthly$name.x,
  gvkey = nordic_top15_monthly$gvkey,
  price_date = nordic_top15_monthly$p_date,
  bid_ask.avg = nordic_top15_monthly$bid_ask_monthly.avg,
  p_year_month = nordic_top15_monthly$year_month,
  price_adj = nordic_top15_monthly$price_adj,
  return = nordic_top15_monthly$return,
  fiscal_date = nordic_top15_monthly$f_date,
```

```r
  fiscal_year = nordic_top15_monthly$f_year,
  N = nordic_top15_monthly$N,
  tot_rank = nordic_top15_monthly$rank_total)

# computing the monthly weights during the holding period
# indexing returns for each company during the holding period
FV_portfolio$hprd = FV_portfolio$fiscal_year-2006
FV_portfolio = FV_portfolio %>% group_by(gvkey, hprd) %>%
  mutate(cum_return = cumprod(return + 1))

#creating a dataset with all the relevant years
fv_time = FV_portfolio %>% group_by(p_year_month) %>%
  summarize(date = last(price_date)) %>% arrange(date)

#extracting all the gvkeys for the FV strategy
gv_fv = FV_portfolio %>%
  group_by(gvkey) %>%
  summarize()

#replicating the gvkeys to the number of months
FV_p_fill = data.frame(gvkey = rep(gv_fv$gvkey, 162))
FV_p_fill <- FV_p_fill %>%
  dplyr::arrange(gvkey)

#inserting the years
FV_p_fill$p_year_month = rep(fv_time$p_year_month, length(gv_fv$gvkey))

# merging the repeated time data rows with the actual data (making the panel
data "balanced")
FV_p_fill = FV_p_fill %>% left_join(FV_portfolio,
                                    by = c("gvkey", "p_year_month"))

# creating a dummy vartiable that checks if the company is deleted
FV_del = FV_p_fill %>%
  group_by(gvkey) %>%
  select(gvkey, p_year_month, name) %>%
  filter(substr(p_year_month, 6, 6) !=6 & is.na(lead(name, n=1)) == T
         & is.na(name) ==F & p_year_month != "2021 12") %>%
  mutate(del = 1)
FV_p_fill = FV_p_fill %>% left_join(FV_del, by = c("gvkey", "p_year_month"))
FV_p_fill$del[is.na(FV_p_fill$del) == T] = 0

# creating a dummy vartiable that checks if the company was in the portfolio
in the previous month
FV_new = FV_p_fill %>% select(gvkey, p_year_month, name.x) %>%
  group_by(gvkey) %>%
  filter(is.na(lag(name.x, n=1)) == T & is.na(name.x) == F)
FV_new$new = rep(1, length(FV_new$gvkey))
FV_p_fill = FV_p_fill %>% left_join(FV_new, by = c("gvkey", "p_year_month"))
FV_p_fill$new = ifelse(is.na(FV_p_fill$new) ==T, 0, FV_p_fill$new)

# creating a dummy variable for companies that go out of the portfolio
FV_out = FV_p_fill  %>% select(gvkey, p_year_month, name.x.x) %>%
  group_by(gvkey) %>%
  filter(is.na(lead(name.x.x, n=1))==T & substr(p_year_month, 6, 6) == 6
         & is.na(name.x.x) == F)
FV_out$out = rep(1, length(FV_out$gvkey))
```

```r
FV_p_fill = FV_p_fill %>% left_join(FV_out, by = c("gvkey", "p_year_month"))
FV_p_fill$out = ifelse(is.na(FV_p_fill$out) ==T, 0, FV_p_fill$out)

FV_p_fill = FV_p_fill %>% filter(is.na(name.x.x.x) == F)

# creating weights
FV_p_fill = FV_p_fill %>% group_by(p_year_month) %>%
  mutate(n_companies = n(),
         sum = sum(cum_return))
FV_p_fill = FV_p_fill %>% group_by(p_year_month) %>%
  mutate(sum_del.adj = sum(cum_return) - sum(cum_return * del))
FV_p_fill = FV_p_fill %>%
  mutate(weight = (cum_return * (1-del)) / sum_del.adj)
FV_p_fill = FV_p_fill %>% arrange(price_date) %>% group_by(gvkey) %>%
  mutate(weight_l1 = lag(weight, n=1))
FV_p_fill$weight_l1 = ifelse(month(FV_p_fill$price_date) == 7,
                             1/FV_p_fill$n_companies, FV_p_fill$weight_l1)

# controlling the weights adds up to 100% each month
test_sum_FV = FV_p_fill %>% group_by(p_year_month) %>%
  summarize(sum = sum(weight_l1))

# computing turnover
FV_p_fill = FV_p_fill %>% mutate(
  del_turn = (del * cum_return) / sum)
FV_p_fill = FV_p_fill %>% arrange(price_date) %>%
  group_by(gvkey) %>%
  mutate(del_turn = (del * cum_return) / sum,
         keep_turn = (weight_l1 - lag(weight, n=1)) * (1-new),
         out_turn = -weight * out,
         new_turn = new * weight_l1)
FV_p_fill$keep_turn_buy = ifelse(FV_p_fill$keep_turn >= 0,
                                 FV_p_fill$keep_turn, 0)
FV_p_fill$keep_turn_sold = ifelse(FV_p_fill$keep_turn <= 0,
                                  FV_p_fill$keep_turn, 0)
FV_turnover = FV_p_fill %>% arrange(price_date) %>% group_by(p_year_month)
%>%
  summarize(date = last(price_date),
            del_turn = sum(del_turn, na.rm = T),
            out_turn = sum(out_turn, na.rm = T),
            new_turn = sum(new_turn, na.rm = T),
            keep_turn_buy = sum(keep_turn_buy, na.rm = T),
            keep_turn_sold = sum(keep_turn_sold, na.rm = T))
FV_turnover = FV_turnover %>% arrange(date) %>%
  mutate(new_turn = lead(new_turn, n=1),
         keep_turn_buy = lead(keep_turn_buy, n=1),
         keep_turn_sold = lead(keep_turn_sold, n=1) )

# summarising portfolio turnover and avg bid-ask
FV_trans_costs = FV_turnover %>% group_by(p_year_month) %>%
  summarize(turnover_FV = sum(del_turn + new_turn + keep_turn_buy))
FV_trans_costs = FV_p_fill %>% group_by(p_year_month) %>%
  summarize(
    date = last(price_date),
    bid_ask_FV = mean(bid_ask.avg, na.rm =T)) %>%
  inner_join(FV_trans_costs, by = "p_year_month") %>%
  arrange(date)
```

```r
FV_yearly_turn = FV_trans_costs %>%
  mutate(p_constr_yr = ifelse(month(date) >= 6,
                              year(date), year(date)-1)) %>%
  group_by(p_constr_yr) %>%
  summarize(yrly_turn_FV = sum(turnover_FV))

# computing weighted average returns (lagged weights)
FV_monthly_returns = FV_p_fill %>%
  group_by(p_year_month) %>%
  summarize(return = sum(return * weight_l1),
            date = last(price_date)) %>% arrange(date)


# --- First Veritas quartile portfolios ---
# adding a column for relative rank
fv_nordic_orig = fv_nordic_orig %>%
  group_by(f_year) %>%
  mutate(rank_sales_g = order(order(sales_g_10, decreasing=F)),
         rank_ROE = order(order(ROE_10, decreasing=F)),
         rank_CCR = order(order(CCR_10, decreasing=F)),
         rank_margin_var = order(order(margin_var, decreasing=T)),
         rank_solidity = order(order(solidity, decreasing=F)),
         rank_PE = order(order(PE, decreasing=T)),
         rank_cyclical_phase = order(order(cyclical_phase, decreasing=T))
  )

# calculating the weighted avg rank
fv_nordic_orig$rank_total = (0.175 * fv_nordic_orig$rank_sales_g +
                             0.175 * fv_nordic_orig$rank_ROE +
                             0.100 * fv_nordic_orig$rank_CCR +
                             0.175 * fv_nordic_orig$rank_margin_var +
                             0.075 * fv_nordic_orig$rank_solidity +
                             0.200 * fv_nordic_orig$rank_PE +
                             0.100 * fv_nordic_orig$rank_cyclical_phase)

fv_nordic_orig = fv_nordic_orig %>% group_by(f_year) %>% mutate(
  rank_quartile = ntile(rank_total, 4))

fv_1qrt = fv_nordic_orig %>% filter(rank_quartile == 1 & f_year >= 2007) %>%
  mutate(year_match = year(price_date))
fv_2qrt = fv_nordic_orig %>% filter(rank_quartile == 2 & f_year >= 2007) %>%
  mutate(year_match = year(price_date))
fv_3qrt = fv_nordic_orig %>% filter(rank_quartile == 3 & f_year >= 2007) %>%
  mutate(year_match = year(price_date))
fv_4qrt = fv_nordic_orig %>% filter(rank_quartile == 4 & f_year >= 2007) %>%
  mutate(year_match = year(price_date))

# computing equal-weighted returns
fv_1qrt = prices_nordic_clean %>%
  inner_join(fv_1qrt, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(date = last(p_date), FV_1q_return = mean(return, na.rm = T)) %>%
  arrange(date)
fv_2qrt = prices_nordic_clean %>%
  inner_join(fv_2qrt, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(FV_2q_return = mean(return, na.rm = T))
```

```r
fv_3qrt = prices_nordic_clean %>%
  inner_join(fv_3qrt, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(FV_3q_return = mean(return, na.rm = T))
fv_4qrt = prices_nordic_clean %>%
  inner_join(fv_4qrt, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(FV_4q_return = mean(return, na.rm = T))

fv_quartiles = fv_1qrt %>% left_join(fv_2qrt, by = "year_month")
fv_quartiles = fv_quartiles %>% left_join(fv_3qrt, by = "year_month")
fv_quartiles = fv_quartiles %>% left_join(fv_4qrt, by = "year_month")

# --------------------------------
# --- "Magic Formula" backtest ---
# --------------------------------

# removing utilities
Greenblatt_nordic = nordic %>% filter(substr(naics, 1, 2) != 22)

# generating variables
Greenblatt_nordic = Greenblatt_nordic %>%
  mutate(E_yld = ebit / (lt_debt - cash + m_cap_in_a_currency),
         ROCE = ebit / (tot_assets - cur_liab) )

# extracting the relevant data
Greenblatt_nordic = data.frame(
  name = Greenblatt_nordic$name,
  gvkey = Greenblatt_nordic$gvkey,
  f_year = Greenblatt_nordic$f_year,
  ROCE = Greenblatt_nordic$ROCE,
  E_yld = Greenblatt_nordic$E_yld,
  price_date = Greenblatt_nordic$price_date,
  m_cap_NOK = Greenblatt_nordic$m_cap_NOK)

# filtering out companies with missing data
Greenblatt_nordic = Greenblatt_nordic %>% filter(ROCE != "NA")
Greenblatt_nordic = Greenblatt_nordic %>% filter(ROCE != "Inf")
Greenblatt_nordic = Greenblatt_nordic %>% filter(ROCE != "NaN")
Greenblatt_nordic = Greenblatt_nordic %>% filter(E_yld != "NA")
Greenblatt_nordic = Greenblatt_nordic %>% filter(E_yld != "Inf")
Greenblatt_nordic = Greenblatt_nordic %>% filter(E_yld != "NaN")

Greenblatt_nordic = Greenblatt_nordic %>% filter(m_cap_NOK != "NA")
Greenblatt_nordic = Greenblatt_nordic %>% filter(m_cap_NOK >= m_cap_lim)

# ranking the companies by Greenblatts variables
Greenblatt_nordic = Greenblatt_nordic %>%
  group_by(f_year) %>%
  mutate(rank_ROCE = order(order(ROCE, decreasing=F)),
         rank_E_yld = order(order(E_yld, decreasing=F))
  )

#calculating the total rank
Greenblatt_nordic$rank_total = (Greenblatt_nordic$rank_ROCE +
                                  Greenblatt_nordic$rank_E_yld)
```

```r
# removing duplicates
Greenblatt_nordic = aggregate(x = Greenblatt_nordic,
                              by = list(Greenblatt_nordic$f_year,
                                        Greenblatt_nordic$gvkey),
                              FUN = "last")

# extracting the top 25 companies
Greenblatt_nordic_top25 = Greenblatt_nordic %>% filter(f_year >= 2000) %>%
  arrange(desc(rank_total)) %>%
  group_by(f_year) %>%
  slice(1:25)

# calculating returns for the portfolio
# creating year_match variable for the GB portfolio
Greenblatt_nordic_top25$year_match = Greenblatt_nordic_top25$f_year

# extracting only the GB winners from return data
Greenblatt_nordic_top25_monthly = prices_nordic_clean %>%
  inner_join(Greenblatt_nordic_top25, by = c("year_match", "gvkey"))

# extracting relevant variable
GB_nordic_portfolio = data.frame(
  price_date = Greenblatt_nordic_top25_monthly$p_date,
  name = Greenblatt_nordic_top25_monthly$name.x,
  gvkey = Greenblatt_nordic_top25_monthly$gvkey,
  price_adj = Greenblatt_nordic_top25_monthly$price_adj,
  bid_ask.avg = Greenblatt_nordic_top25_monthly$bid_ask_monthly.avg,
  p_year_month = paste(Greenblatt_nordic_top25_monthly$year,
                       Greenblatt_nordic_top25_monthly$month),
  return = Greenblatt_nordic_top25_monthly$return,
  f_year = Greenblatt_nordic_top25_monthly$f_year,
  rank = Greenblatt_nordic_top25_monthly$rank_total)

# generating weight
# indexing returns for companies by each year
GB_nordic_portfolio$hprd = GB_nordic_portfolio$f_year-2006
GB_nordic_portfolio = GB_nordic_portfolio %>% group_by(gvkey, hprd) %>%
  mutate(cum_return = cumprod(return + 1))

#extracting all the gvkeys for the FV strategy
gv_GB = GB_nordic_portfolio %>%
  group_by(gvkey) %>%
  summarize()

#replicating the gvkeys to the number of months
GB_p_fill = data.frame(gvkey = rep(gv_GB$gvkey, 162))
GB_p_fill <- GB_p_fill %>%
  dplyr::arrange(gvkey)

#inserting the years
GB_p_fill$p_year_month = rep(fv_time$p_year_month, length(gv_GB$gvkey))

# merging the repeated time data rows with the actual data (making the panel
data "balanced")
GB_p_fill = GB_p_fill %>% left_join(GB_nordic_portfolio,
                                    by = c("gvkey", "p_year_month"))
```

```r
# creating a dummy vartiable that checks if the company is deleted
GB_del = GB_p_fill %>%
  group_by(gvkey) %>%
  select(gvkey, p_year_month, name) %>%
  filter(substr(p_year_month, 6, 6) !=6 & is.na(lead(name, n=1)) == T &
           is.na(name) ==F & p_year_month != "2021 12") %>%
  mutate(del = 1)
GB_p_fill = GB_p_fill %>% left_join(GB_del, by = c("gvkey", "p_year_month"))
GB_p_fill$del[is.na(GB_p_fill$del) == T] = 0

# creating a dummy vartiable that checks if the company was in the portfolio
in the previous month
GB_new = GB_p_fill %>% select(gvkey, p_year_month, name.x) %>%
  group_by(gvkey) %>%
  filter(is.na(lag(name.x, n=1)) == T & is.na(name.x) == F)
GB_new$new = rep(1, length(GB_new$p_year_month))
GB_p_fill = GB_p_fill %>% left_join(GB_new, by = c("gvkey", "p_year_month"))
GB_p_fill$new = ifelse(is.na(GB_p_fill$new) ==T, 0, GB_p_fill$new)

# creating a dummy variable for companies that go out of the portfolio
GB_out = GB_p_fill  %>% select(gvkey, p_year_month, name.x.x) %>%
  group_by(gvkey) %>%
  filter(is.na(lead(name.x.x, n=1))==T & substr(p_year_month, 6, 6) == 6 &
           is.na(name.x.x) == F)
GB_out$out = rep(1, length(GB_out$p_year_month))
GB_p_fill = GB_p_fill %>% left_join(GB_out, by = c("gvkey", "p_year_month"))
GB_p_fill$out = ifelse(is.na(GB_p_fill$out) ==T, 0, GB_p_fill$out)

GB_p_fill = GB_p_fill %>% filter(is.na(name.x.x.x) == F)

# creating weights
GB_p_fill = GB_p_fill %>% group_by(p_year_month) %>%
  mutate(n_companies = n(),
         sum = sum(cum_return))
GB_p_fill = GB_p_fill %>% group_by(p_year_month) %>%
  mutate(sum_del.adj = sum(cum_return) - sum(cum_return * del))
GB_p_fill = GB_p_fill %>%
  mutate(weight = (cum_return * (1-del)) / sum_del.adj)
GB_p_fill = GB_p_fill %>% arrange(price_date) %>% group_by(gvkey) %>%
  mutate(weight_l1 = lag(weight, n=1))
GB_p_fill$weight_l1 = ifelse(month(GB_p_fill$price_date) == 7,
                             1/GB_p_fill$n_companies, GB_p_fill$weight_l1)

# controlling the weights adds up to 100% each month
test_sum_GB = GB_p_fill %>% group_by(p_year_month) %>%
  summarize(sum = sum(weight_l1))

# computing portfolio turnover
# columns for monthly transactions
GB_p_fill = GB_p_fill %>% mutate(
  del_turn = (del * cum_return) / sum)
GB_p_fill = GB_p_fill %>% arrange(price_date) %>%
  group_by(gvkey) %>%
  mutate(del_turn = (del * cum_return) / sum,
         keep_turn = (weight_l1 - lag(weight, n=1)) * (1-new),
         out_turn = -weight * out,
         new_turn = new * weight_l1)
```

```
GB_p_fill$keep_turn_buy = ifelse(GB_p_fill$keep_turn >= 0,
                                 GB_p_fill$keep_turn, 0)
GB_p_fill$keep_turn_sold = ifelse(GB_p_fill$keep_turn <= 0,
                                  GB_p_fill$keep_turn, 0)

GB_turnover = GB_p_fill %>% arrange(price_date) %>% group_by(p_year_month)
%>%
  summarize(date = last(price_date),
            del_turn = sum(del_turn, na.rm = T),
            out_turn = sum(out_turn, na.rm = T),
            new_turn = sum(new_turn, na.rm = T),
            keep_turn_buy = sum(keep_turn_buy, na.rm = T),
            keep_turn_sold = sum(keep_turn_sold, na.rm = T))
GB_turnover = GB_turnover %>% arrange(date) %>%
  mutate(new_turn = lead(new_turn, n=1),
         keep_turn_buy = lead(keep_turn_buy, n=1),
         keep_turn_sold = lead(keep_turn_sold, n=1) )
GB_turnover = GB_turnover %>% mutate(
  ctrl = out_turn + new_turn + keep_turn_buy + keep_turn_sold)

# summarizing portfolio turnover and avg bid-ask
GB_trans_costs = GB_turnover %>% group_by(p_year_month) %>%
  summarize(turnover_GB = sum(del_turn + new_turn + keep_turn_buy))
GB_trans_costs = GB_p_fill %>% group_by(p_year_month) %>%
  summarize(
    date = last(price_date),
    bid_ask_GB = mean(bid_ask.avg, na.rm =T)) %>%
  inner_join(GB_trans_costs, by = "p_year_month") %>%
  arrange(date)

# computing value weighted returns (lagged weights)
GB_monthly_returns = GB_p_fill %>%
  group_by(p_year_month) %>%
  summarize(return_GB = sum(return * weight_l1),
            date = last(price_date)) %>% arrange(date)
GB_monthly_returns = GB_monthly_returns[, c(1,2)]

# -------------------------------------------
# --- "Magic Formula" quartile portfolios ---
# -------------------------------------------

# creating a quartile rank column
Greenblatt_nordic = Greenblatt_nordic %>% group_by(f_year) %>% mutate(
  rank_quartile = ntile(rank_total, 4))

gb_1qrt = Greenblatt_nordic %>% filter(rank_quartile == 1 & f_year >= 2007)
%>%
  mutate(year_match = year(price_date))
gb_2qrt = Greenblatt_nordic %>% filter(rank_quartile == 2 & f_year >= 2007)
%>%
  mutate(year_match = year(price_date))
gb_3qrt = Greenblatt_nordic %>% filter(rank_quartile == 3 & f_year >= 2007)
%>%
  mutate(year_match = year(price_date))
gb_4qrt = Greenblatt_nordic %>% filter(rank_quartile == 4 & f_year >= 2007)
%>%
  mutate(year_match = year(price_date))
```

```r
# computing equal-weighted avg returns
gb_1qrt = prices_nordic_clean %>%
  inner_join(gb_1qrt, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(date = last(p_date), GB_1q_return = mean(return, na.rm = T)) %>%
  arrange(date)
gb_2qrt = prices_nordic_clean %>%
  inner_join(gb_2qrt, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(GB_2q_return = mean(return, na.rm = T))
gb_3qrt = prices_nordic_clean %>%
  inner_join(gb_3qrt, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(GB_3q_return = mean(return, na.rm = T))
gb_4qrt = prices_nordic_clean %>%
  inner_join(gb_4qrt, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(GB_4q_return = mean(return, na.rm = T))

gb_quartiles = gb_1qrt %>% left_join(gb_2qrt, by = "year_month")
gb_quartiles = gb_quartiles %>% left_join(gb_3qrt, by = "year_month")
gb_quartiles = gb_quartiles %>% left_join(gb_4qrt, by = "year_month")

# ------------------------------
# --- Back testing Piotroski ---
# ------------------------------

# extracting relevant data
piotroski_nordic = nordic %>%
  select(gvkey, name, f_year, f_date, ni, cf_o,
         tot_assets, lt_debt, cash, cur_assets, cur_liab,
         sales, COGS, book_price, m_cap_NOK) %>% filter(is.na(name) == F)
piotroski_nordic$f_y_m = paste(piotroski_nordic$f_year,
                               month(piotroski_nordic$f_date))
shares_join_nordic = data.frame(
  f_y_m = prices_nordic_clean$year_month,
  gvkey = prices_nordic_clean$gvkey,
  shares = prices_nordic_clean$shares,
  ajexdi = prices_nordic_clean$ajexdi)
piotroski_nordic = piotroski_nordic %>% left_join(shares_join_nordic,
                                                   by = c("f_y_m", "gvkey"))

# generating F_SCORE
piotroski_nordic = piotroski_nordic %>%
  dplyr::mutate(
    roa = ni / tot_assets,
    cfo = cf_o / tot_assets,
    lever = (lt_debt - cash) / tot_assets,
    liquid = cur_assets / cur_liab,
    margin = (sales - COGS) / sales,
    turn = sales / tot_assets  )
piotroski_nordic = piotroski_nordic %>%
  group_by(gvkey) %>%
  dplyr::mutate(
    d_roa = roa - lag(roa, n = 1, default = NA),
    accrual = cfo - roa,
    d_lever = lever - lag(lever, n = 1, default = NA),
    d_liquid = liquid - lag(liquid, n = 1, default = NA),
    d_margin = margin - lag(margin, n = 1, default = NA),
    d_turn = turn - lag(turn, n = 1, default = NA)    )
piotroski_nordic = piotroski_nordic %>%
  group_by(gvkey) %>%
  dplyr::mutate(
    shares_offered = shares - (lag(shares, n=1, default = NA) * (
      lag(ajexdi, n=1, default = NA) / ajexdi)))
piotroski_nordic = piotroski_nordic %>%
  group_by(gvkey) %>%
  dplyr::mutate(
    pct_shares_offered = shares_offered / lag(shares, n=1, default = NA))

#    creating dummy variables
piotroski_nordic$F_roa = rep(0, 26410)
piotroski_nordic$F_roa[piotroski_nordic$roa >= 0] = 1
piotroski_nordic$F_cfo = rep(0, 26410)
piotroski_nordic$F_cfo[piotroski_nordic$cfo >= 0] = 1
piotroski_nordic$F_d_roa = rep(0, 26410)
piotroski_nordic$F_d_roa[piotroski_nordic$d_roa >= 0] = 1
piotroski_nordic$F_accrual = rep(0, 26410)
piotroski_nordic$F_accrual[piotroski_nordic$accrual >= 0] = 1
piotroski_nordic$F_d_lever = rep(0, 26410)
piotroski_nordic$F_d_lever[piotroski_nordic$d_lever <= 0] = 1
piotroski_nordic$F_d_liquid = rep(0, 26410)
piotroski_nordic$F_d_liquid[piotroski_nordic$d_liquid >= 0] = 1
piotroski_nordic$F_d_margin = rep(0, 26410)
piotroski_nordic$F_d_margin[piotroski_nordic$d_margin >= 0] = 1
piotroski_nordic$F_d_turn = rep(0, 26410)
piotroski_nordic$F_d_turn[piotroski_nordic$d_turn >= 0] = 1
piotroski_nordic$eq_offer = rep(0, 26410)
piotroski_nordic$eq_offer[piotroski_nordic$pct_shares_offered <= 0.05] = 1

piotroski_nordic = piotroski_nordic %>%
  dplyr::mutate(
    F_score = F_roa + F_cfo + F_d_roa + F_accrual + F_d_lever +
      F_d_liquid + F_d_margin + F_d_turn + eq_offer)

# dropping NA's
piotroski_nordic = piotroski_nordic %>% filter(is.na(book_price) == F)
piotroski_nordic = piotroski_nordic %>% filter(is.na(d_roa) == F)
piotroski_nordic = piotroski_nordic %>% filter(is.na(cfo) == F)
piotroski_nordic = piotroski_nordic %>% filter(is.na(d_lever) == F)
piotroski_nordic = piotroski_nordic %>% filter(is.na(d_liquid) == F)
piotroski_nordic = piotroski_nordic %>% filter(is.na(d_margin) == F)
piotroski_nordic = piotroski_nordic %>% filter(is.na(d_turn) == F)
piotroski_nordic = piotroski_nordic %>% filter(is.na(shares_offered) == F)
piotroski_nordic = piotroski_nordic %>% filter(m_cap_NOK >= m_cap_lim)

# ranking the companies on book-to-price
piotr_univ_nord = piotroski_nordic %>%
  group_by(f_year)
piotr_univ_nord = data.frame(N = count(piotr_univ_nord, "gvkey"))
piotr_univ_nord = data.frame(
  f_year = piotr_univ_nord[, 1],
  N = piotr_univ_nord[, 3])
```

```r
piotroski_nordic = piotroski_nordic %>%
  left_join(piotr_univ_nord, by = "f_year")
piotroski_nordic = piotroski_nordic %>%
  group_by(f_year) %>%
  mutate(pct_rank_b_p = (order(order(book_price, decreasing=F)) / N))

# extracting piotroski winners
piotr_w_nordic = piotroski_nordic %>% filter(pct_rank_b_p >= 0.7)
piotr_w_nordic = piotr_w_nordic %>%
  select(gvkey, f_year, book_price, F_score) %>%
  filter(F_score >= 8)

piotr_w_nordic$year_match = piotr_w_nordic$f_year

# lagging the m_cap_nok variable
prices_nordic_clean = prices_nordic_clean %>%
  dplyr::group_by(gvkey) %>%
  dplyr::mutate(m_cap_NOK_l1 = lag(m_cap_NOK, n = 1))

# extracting monthly securities data for the PT portfolio
piotr_w_nordic_monthly = prices_nordic_clean %>%
  inner_join(piotr_w_nordic, by = c("year_match", "gvkey"))

# extracting relevant variables
piotr_w_portfolio = data.frame(
  price_date = piotr_w_nordic_monthly$p_date,
  p_year_month = piotr_w_nordic_monthly$year_month,
  name = piotr_w_nordic_monthly$name,
  gvkey = piotr_w_nordic_monthly$gvkey,
  price_adj = piotr_w_nordic_monthly$price_adj,
  return = piotr_w_nordic_monthly$return,
  f_year = piotr_w_nordic_monthly$f_year,
  book_price = piotr_w_nordic_monthly$book_price,
  F_score = piotr_w_nordic_monthly$F_score)

# computing weights
# indexing companies' returns by holding period
piotr_w_portfolio$hprd = piotr_w_portfolio$f_year-2006
piotr_w_portfolio = piotr_w_portfolio %>% group_by(gvkey, hprd) %>%
  mutate(cum_return = cumprod(return + 1))

#extracting all the gvkeys for the Piotroski strategy
gv_PT = piotr_w_portfolio %>%
  group_by(gvkey) %>%
  summarize()

#replicating the gvkeys to the number of months
PT_p_fill = data.frame(gvkey = rep(gv_PT$gvkey, 162))
PT_p_fill <- PT_p_fill %>%
  dplyr::arrange(gvkey)

#inserting the years
PT_p_fill$p_year_month = rep(fv_time$p_year_month, length(gv_PT$gvkey))

# merging the repeated time data rows with the actual data (making the panel
data "balanced")
PT_p_fill = PT_p_fill %>%
  left_join(piotr_w_portfolio, by = c("gvkey", "p_year_month"))

# creating a dummy variable that checks if the company is deleted
PT_del = PT_p_fill %>%
  group_by(gvkey) %>%
  select(gvkey, p_year_month, name) %>%
  filter(substr(p_year_month, 6, 6) !=6 & is.na(lead(name, n=1)) == T &
           is.na(name) ==F & p_year_month != "2021 12") %>%
  mutate(del = 1)
PT_p_fill = PT_p_fill %>% left_join(PT_del, by = c("gvkey", "p_year_month"))
PT_p_fill$del[is.na(PT_p_fill$del) == T] = 0

PT_p_fill = PT_p_fill %>% filter(is.na(name.x) == F)

# creating weights
PT_p_fill = PT_p_fill %>% group_by(p_year_month) %>%
  mutate(n_companies = n(),
         sum = sum(cum_return),
         sum_del.adj = sum(cum_return) - sum(cum_return * del))
PT_p_fill = PT_p_fill %>%
  mutate(weight = (cum_return * (1-del)) / sum_del.adj)
PT_p_fill = PT_p_fill %>% arrange(price_date) %>% group_by(gvkey) %>%
  mutate(weight_l1 = lag(weight, n=1))
PT_p_fill$weight_l1 = ifelse(month(PT_p_fill$price_date) == 7,
                             1/PT_p_fill$n_companies, PT_p_fill$weight_l1)

# controlling the weights adds up to 100% each month
test_sum_PT = PT_p_fill %>% group_by(p_year_month) %>%
  summarize(sum = sum(weight_l1))

# computing weighted returns (lagged weights)
piotr_nordic_monthly_returns = PT_p_fill %>%
  group_by(p_year_month) %>%
  summarize(return_PT = sum(return * weight_l1))

# -----------------------------------------
# --- Piotroski winner/loser portfolios ---
# -----------------------------------------

piotroski_nordic$year_match = piotroski_nordic$f_year
pt_h_7_9 = piotroski_nordic %>%
  filter(f_year >= 2007 & pct_rank_b_p >= 0.7 & F_score >= 7)
pt_h_4_6 = piotroski_nordic %>%
  filter(f_year >= 2007 & pct_rank_b_p >= 0.7 & F_score >= 4 & F_score <= 6)
pt_h_0_3 = piotroski_nordic %>% filter(f_year >= 2007 & pct_rank_b_p >= 0.7 &
                                         F_score <= 3)

pt_l_7_9 = piotroski_nordic %>%
  filter(f_year >= 2007 & pct_rank_b_p <= 0.3 & F_score >= 7)
pt_l_4_6 = piotroski_nordic %>%
  filter(f_year >= 2007 & pct_rank_b_p <= 0.3 & F_score >= 4 & F_score <= 6)
pt_l_0_3 = piotroski_nordic %>%
  filter(f_year >= 2007 & pct_rank_b_p <= 0.3 & F_score <= 3)

pt_h_7_9 = prices_nordic_clean %>%
  inner_join(pt_h_7_9, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(pt_h_7_9_return = mean(return, na.rm = T))
pt_h_4_6 = prices_nordic_clean %>%
  inner_join(pt_h_4_6, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(pt_h_4_6_return = mean(return, na.rm = T))
pt_h_0_3 = prices_nordic_clean %>%
  inner_join(pt_h_0_3, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(pt_h_0_3_return = mean(return, na.rm = T))
pt_l_7_9 = prices_nordic_clean %>%
  inner_join(pt_l_7_9, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(pt_l_7_9_return = mean(return, na.rm = T))
pt_l_4_6 = prices_nordic_clean %>%
  inner_join(pt_l_4_6, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(pt_l_4_6_return = mean(return, na.rm = T))
pt_l_0_3 = prices_nordic_clean %>%
  inner_join(pt_l_0_3, by = c("year_match", "gvkey")) %>%
  group_by(year_month) %>%
  summarize(pt_l_0_3_return = mean(return, na.rm = T))

pt_split = pt_h_7_9 %>% left_join(pt_h_4_6, by = "year_month")
pt_split = pt_split %>% left_join(pt_h_0_3, by = "year_month")
pt_split = pt_split %>% left_join(pt_l_7_9, by = "year_month")
pt_split = pt_split %>% left_join(pt_l_4_6, by = "year_month")
pt_split = pt_split %>% left_join(pt_l_0_3, by = "year_month")
pt_split$p_year_month = pt_split$year_month


# joining results
all_returns = FV_monthly_returns %>%
  left_join(VINX, by = "p_year_month")
all_returns = all_returns %>%
  left_join(FV_actual_returns, by = "p_year_month")
all_returns = all_returns %>%
  left_join(GB_monthly_returns, by = "p_year_month")
all_returns = all_returns %>%
  left_join(piotr_nordic_monthly_returns, by = "p_year_month")

# plotting HPR vs index
all_returns$FV_backtest = cumprod((all_returns$return + 1)) * 100
all_returns$VINX_index = cumprod((all_returns$vinx_return + 1)) * 100
all_returns$GB_backtest = cumprod((all_returns$return_GB + 1)) * 100
all_returns$PT_backtest = cumprod((all_returns$return_PT + 1)) * 100
all_returns %>%
  mutate(F.Veritas = FV_backtest,
         'Magic Formula' = GB_backtest,
         Piotroski = PT_backtest,
         'Market (VINX)' = VINX_index) %>%
  select(date, F.Veritas, 'Magic Formula', 'Market (VINX)', Piotroski) %>%
  pivot_longer(-date, names_to = "portfolio", values_to = "index") %>%
  mutate(label = if_else(date == max(date),
                         as.character(portfolio), NA_character_)) %>%
  ggplot(aes(x=date, y=index, group = portfolio, color=portfolio)) +
  geom_line(size = 1.1, alpha = 1) +
  scale_color_manual(values=c("#009177", "#8f0015", "Black", "#0058a6")) +
  labs(title = "Backtest of strategies in the Nordics",
       subtitle = "Backtest of First Veritas, Greenblatt's Magic formula, and
Piotroski's F_SCORE. (Market cap > 100 MNOK)") +
  theme_wsj() +
  theme(title = element_text(size = 12, family="Times"),
        axis.text.x = element_text(angle = 90, vjust = 0.5,
                                   hjust=1,  family="Times"),
        legend.position = "none",
        plot.margin = unit(c(0.5,2.2,0.5,0.5), "cm")) +
  geom_dl(aes(label = portfolio), method = list("last.points",
                                                 rot = 0,
                                                 fontface = "bold")) +
  coord_cartesian(clip="off") +
  scale_x_datetime(date_breaks = "1 year", date_labels = c("%Y")) +
  scale_y_continuous(name="index", breaks = c(100, 300, 500, 700,
                                               900, 1100, 1300))


# -----------------------------------
# --- generating FF3F variables ---
# -----------------------------------

#extracting relevant data
carhart02 = data.frame(
  gvkey = nordic$gvkey,
  name = nordic$name,
  f_year = nordic$f_year,
  f_date = nordic$f_date,
  m_cap_nok = nordic$m_cap_NOK,
  book_price = nordic$book_price)

# removing NA's and negative b-to-p
carhart02 = carhart02 %>% filter(is.na(m_cap_nok) == F)
carhart02 = carhart02 %>% filter(is.na(book_price) == F)
carhart02 = carhart02 %>% filter(book_price >= 0)

# counting the total number of companes and joinng the data
carhart_univ = carhart02 %>%
  group_by(f_year)
carhart_univ = data.frame(N = count(carhart_univ, "gvkey"))
carhart_univ = data.frame(
  f_year = carhart_univ[, 1],
  N = carhart_univ[, 3])
carhart02 = carhart02 %>% left_join(carhart_univ, by = "f_year")

# ranking the companies on size
carhart02 = carhart02 %>%
  group_by(f_year) %>%
  mutate(rank_m_cap = order(order(m_cap_nok, decreasing=F)))  # decreasing

# create rank in % of total
carhart02$pct_m_cap = carhart02$rank_m_cap / carhart02$N

big = carhart02 %>% filter(pct_m_cap >= 0.5)
small = carhart02 %>% filter(pct_m_cap <= 0.5)

# counting the companies in the small and big portfolio
```

```r
big_univ = big %>%
  group_by(f_year)
big_univ = data.frame(N = count(big_univ, "gvkey"))
big_univ = data.frame(
  f_year = big_univ[, 1],
  N = big_univ[, 3])
big = big %>% left_join(big_univ, by = "f_year")

small_univ = small %>%
  group_by(f_year)
small_univ = data.frame(N = count(small_univ, "gvkey"))
small_univ = data.frame(
  f_year = small_univ[, 1],
  N = small_univ[, 3])
small = small %>% left_join(small_univ, by = "f_year")

# ranking the companies on book-to-price
big = big %>%
  group_by(f_year) %>%
  mutate(pct_rank_b_p = (order(order(book_price, decreasing=F)) / N.y))
small = small %>%
  group_by(f_year) %>%
  mutate(pct_rank_b_p = (order(order(book_price, decreasing=F)) / N.y))

# extracting the six portfolios
small_value = small %>% filter(pct_rank_b_p >= 0.7) %>%
  select(gvkey, name, f_year, f_date, m_cap_nok, book_price)
small_neutral = small %>% filter(pct_rank_b_p <= 0.7 & pct_rank_b_p >= 0.3)
%>%
  select(gvkey, name, f_year, f_date, m_cap_nok, book_price)
small_growth = small %>% filter(pct_rank_b_p <= 0.3) %>%
  select(gvkey, name, f_year, f_date, m_cap_nok, book_price)

big_value = big %>% filter(pct_rank_b_p >= 0.7) %>%
  select(gvkey, name, f_year, f_date, m_cap_nok, book_price)
big_neutral = big %>% filter(pct_rank_b_p <= 0.7 & pct_rank_b_p >= 0.3) %>%
  select(gvkey, name, f_year, f_date, m_cap_nok, book_price)
big_growth = big %>% filter(pct_rank_b_p <= 0.3) %>%
  select(gvkey, name, f_year, f_date, m_cap_nok, book_price)

# extracted value weighted averages of the portfolios
returns_join = prices_nordic_clean %>% filter(return != "Inf") %>%
  select(gvkey, p_date, return, year_match, year_month, m_cap_NOK_l1)
returns_join$f_year = returns_join$year_match

small_value = small_value %>%
  inner_join(returns_join, by = c("gvkey", "f_year"))
small_value_r = small_value %>% group_by(year_month) %>%
  summarize(date = last(p_date),
            SV_r = weighted.mean(return, m_cap_NOK_l1, na.rm = T))

small_neutral = small_neutral %>%
  inner_join(returns_join, by = c("gvkey", "f_year"))
small_neutral_r = small_neutral %>% group_by(year_month) %>%
  summarize(SN_r = weighted.mean(return, m_cap_NOK_l1, na.rm = T))

small_growth = small_growth %>%
  inner_join(returns_join, by = c("gvkey", "f_year"))
small_growth_r = small_growth %>% group_by(year_month) %>%
  summarize(SG_r = weighted.mean(return, m_cap_NOK_l1, na.rm = T))

big_value = big_value %>%
  inner_join(returns_join, by = c("gvkey", "f_year"))
big_value_r = big_value %>% group_by(year_month) %>%
  summarize(BV_r = weighted.mean(return, m_cap_NOK_l1, na.rm = T))

big_neutral = big_neutral %>%
  inner_join(returns_join, by = c("gvkey", "f_year"))
big_neutral_r = big_neutral %>% group_by(year_month) %>%
  summarize(BN_r = weighted.mean(return, m_cap_NOK_l1, na.rm = T))

big_growth = big_growth %>%
  inner_join(returns_join, by = c("gvkey", "f_year"))
big_growth_r = big_growth %>% group_by(year_month) %>%
  summarize(BG_r = weighted.mean(return, m_cap_NOK_l1, na.rm = T))

# computing SMB and HML
FF3_nordic = small_value_r %>% left_join(small_neutral_r, by = "year_month")
FF3_nordic = FF3_nordic %>% left_join(small_growth_r, by = "year_month")
FF3_nordic = FF3_nordic %>% left_join(big_value_r, by = "year_month")
FF3_nordic = FF3_nordic %>% left_join(big_neutral_r, by = "year_month")
FF3_nordic = FF3_nordic %>% left_join(big_growth_r, by = "year_month")
FF3_nordic = FF3_nordic %>% arrange(date)
FF3_nordic$SMB = (1/3) * (FF3_nordic$SV_r + FF3_nordic$SN_r +
FF3_nordic$SG_r) -
  (1/3) * (FF3_nordic$BV_r + FF3_nordic$BN_r + FF3_nordic$BG_r)
FF3_nordic$HML = (1/2) * (FF3_nordic$SV_r + FF3_nordic$BV_r) -
  (1/2) * (FF3_nordic$SG_r + FF3_nordic$BG_r)

# ----------------------
# --- generating WML ---
# ----------------------

# generating momentum factor
carhart01 = prices_nordic_clean %>%
  dplyr::group_by(gvkey) %>%
  dplyr::mutate(price_adj_l11 = lag(price_adj, n = 11))
carhart01$l1_return_11m = lag(
  ((carhart01$price_adj / carhart01$price_adj_l11)-1), n = 1)
carhart01 = carhart01 %>% filter(is.infinite(l1_return_11m) == F)
carhart01 = carhart01 %>% filter(is.na(l1_return_11m) == F)
carhart01 = carhart01 %>% filter(is.infinite(return) == F)
carhart01 = carhart01 %>% filter(is.na(return) == F)
carhart01 = carhart01 %>% filter(is.na(m_cap_NOK_l1) == F)

# dropping some irrelevant columns
carhart01 = carhart01 %>% select(p_date, name, gvkey, p_currency, year_month,
                                 price_adj, return, nok_per_units, m_cap,
                                 m_cap_NOK, m_cap_NOK_l1, l1_return_11m,
year)

# counting the companies by year_month
WML_univ = carhart01 %>%
  group_by(year_month)
```

```r
WML_univ = data.frame(N = count(WML_univ, "gvkey"))
WML_univ = data.frame(
  year_month = WML_univ[, 1],
  N = WML_univ[, 3])
carhart01 = carhart01 %>% left_join(WML_univ, by = "year_month")

# ranking the companies monthly on momentum
carhart01 = carhart01 %>%
  group_by(year_month) %>%
  mutate(pct_rank_momentun = (order(order(l1_return_11m, decreasing=F)) / N))

# weighted avg. returns for top/bottom 30%
winners = carhart01 %>% group_by(year_month) %>%
  filter(pct_rank_momentun >= 0.7) %>%
  summarize(date = last(p_date),
            W_r = weighted.mean(return, m_cap_NOK_l1, na.rm = TRUE))

losers = carhart01 %>% filter(pct_rank_momentun <= 0.3) %>%
  group_by(year_month) %>%
  summarize(date = last(p_date),
            L_r = weighted.mean(return, m_cap_NOK_l1, na.rm = TRUE))
WML = winners %>% left_join(losers, by = "year_month")
WML$WML = WML$W_r - WML$L_r

WML = data.frame(
  p_year_month = WML$year_month,
  WML = WML$WML)

# joining SMB, HML, and WML
carhart = data.frame(
  p_year_month = FF3_nordic$year_month,
  SMB = FF3_nordic$SMB,
  HML = FF3_nordic$HML)
carhart = carhart %>% left_join(WML, by = "p_year_month")

# -------------------
# --- regressions ---
# -------------------

# joining risk-free data
all_returns = all_returns %>% left_join(nibor_1m, by = "p_year_month")

# computing excess returns
all_returns$excess_fv = all_returns$return - all_returns$nibor_1m
all_returns$excess_GB = all_returns$return_GB - all_returns$nibor_1m
all_returns$excess_PT = all_returns$return_PT - all_returns$nibor_1m
all_returns$mp = all_returns$vinx_return - all_returns$nibor_1m
all_returns = all_returns %>% left_join(carhart, by = "p_year_month")

# computing reg models for gross returns
fv_capm <- lm(excess_fv ~ mp, data = all_returns)
fv_ff3 <- lm(excess_fv ~ mp + SMB + HML, data = all_returns)
fv_carhart <- lm(excess_fv ~ mp + SMB + HML + WML, data = all_returns)
gb_capm <- lm(excess_GB ~ mp, data = all_returns)
gb_ff3 <- lm(excess_GB ~ mp + SMB + HML, data = all_returns)
gb_carhart <- lm(excess_GB ~ mp + SMB + HML + WML, data = all_returns)
PT_capm <- lm(excess_PT ~ mp, data = all_returns)
```

```r
PT_ff3 <- lm(excess_PT ~ mp + SMB + HML, data = all_returns)
PT_carhart <- lm(excess_PT ~ mp + SMB + HML + WML, data = all_returns)

# creating table
reg_nordic =
  export_summs(fv_capm, fv_ff3, fv_carhart,
               gb_capm, gb_ff3, gb_carhart,
               PT_capm, PT_ff3, PT_carhart,
               model.names = c("FV CAPM", "FV FF3", "FV Carhart",
                               "GB CAPM", "GB FF3", "GB Carhart",
                               "PT CAPM", "PT FF3", "PT Carhart"),
               digits = 5,
               coefs = c("Alpha" = "(Intercept)",
                         "Market" = "mp",
                         "HML" = "HML",
                         "SMB" = "SMB",
                         "WML" = "WML"),
               stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.10),
               statistics = c("adj. R squared" = "adj.r.squared",
                              "SE residuals" = "sigma",
                              "N" = "nobs"),
               error_format = "({p.value})")

reg_nordic %>%  print_latex() # printing latex code for reg-table

# adjusting returns for transaction costs
comission = 0.0005

all_returns = GB_trans_costs %>%
  select(p_year_month, bid_ask_GB, turnover_GB) %>%
  inner_join(all_returns, by = "p_year_month")
all_returns = FV_trans_costs %>%
  select(p_year_month, bid_ask_FV, turnover_FV) %>%
  inner_join(all_returns, by = "p_year_month")

all_returns$turnover_FV[all_returns$p_year_month == "2021 12"] = 0
all_returns$turnover_GB[all_returns$p_year_month == "2021 12"] = 0

all_returns = all_returns %>% mutate(
  excess_fv_adj =
    (return+1) * (1 - 2*turnover_FV*((bid_ask_FV/2) + comission)) -
    1 - nibor_1m,
  excess_gb_adj =
    (return_GB+1) * (1 - 2*turnover_GB*((bid_ask_GB/2) + comission)) -
    1 - nibor_1m )

fv_capm_adj <- lm(excess_fv_adj ~ mp, data = all_returns)
fv_ff3_adj <- lm(excess_fv_adj ~ mp + SMB + HML, data = all_returns)
fv_carhart_adj <- lm(excess_fv_adj ~ mp + SMB + HML + WML, data =
all_returns)

gb_capm_adj <- lm(excess_gb_adj ~ mp, data = all_returns)
gb_ff3_adj <- lm(excess_gb_adj ~ mp + SMB + HML, data = all_returns)
gb_carhart_adj <- lm(excess_gb_adj ~ mp + SMB + HML + WML, data =
all_returns)

reg_nordic_adj =
  export_summs(fv_capm_adj, fv_ff3_adj, fv_carhart_adj,
               gb_capm_adj, gb_ff3_adj, gb_carhart_adj,
               model.names = c("FV CAPM", "FV FF3", "FV Carhart",
                                "GB CAPM", "GB FF3", "GB Carhart"),
               digits = 5,
               coefs = c("Alpha" = "(Intercept)",
                         "Market" = "mp",
                         "HML" = "HML",
                         "SMB" = "SMB",
                         "WML" = "WML"),
               stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.10),
               statistics = c("adj. R squared" = "adj.r.squared",
                              "SE residuals" = "sigma",
                              "N" = "nobs"),
               error_format = "({p.value})")

reg_nordic_adj %>%  print_latex() # printing latex code for reg-table


# regression on winner/loser portfolios - MF and FV
split_returns = fv_quartiles %>%
  select(year_month, FV_1q_return,FV_2q_return, FV_3q_return, FV_4q_return)
%>%
  left_join(gb_quartiles, by = "year_month")
split_returns$p_year_month = split_returns$year_month
split_returns = all_returns %>%
  select(p_year_month, vinx_return, mp, nibor_1m, SMB, HML, WML) %>%
  left_join(split_returns, by = "p_year_month")
split_returns = split_returns %>% left_join(pt_split, by = "p_year_month")

split_returns = split_returns %>% mutate(
  excess_fv_1q = FV_1q_return - nibor_1m,
  excess_fv_2q = FV_2q_return - nibor_1m,
  excess_fv_3q = FV_3q_return - nibor_1m,
  excess_fv_4q = FV_4q_return - nibor_1m,
  excess_gb_1q = GB_1q_return - nibor_1m,
  excess_gb_2q = GB_2q_return - nibor_1m,
  excess_gb_3q = GB_3q_return - nibor_1m,
  excess_gb_4q = GB_4q_return - nibor_1m,
  excess_pt_h_7_9 = pt_h_7_9_return - nibor_1m,
  excess_pt_h_4_6 = pt_h_4_6_return - nibor_1m,
  excess_pt_h_0_3 = pt_h_0_3_return - nibor_1m,
  excess_pt_l_7_9 = pt_l_7_9_return - nibor_1m,
  excess_pt_l_4_6 = pt_l_4_6_return - nibor_1m,
  excess_pt_l_0_3 = pt_l_0_3_return - nibor_1m)

fv_1q <- lm(excess_fv_1q ~ mp + SMB + HML + WML, data = split_returns)
fv_2q <- lm(excess_fv_2q ~ mp + SMB + HML + WML, data = split_returns)
fv_3q <- lm(excess_fv_3q ~ mp + SMB + HML + WML, data = split_returns)
fv_4q <- lm(excess_fv_4q ~ mp + SMB + HML + WML, data = split_returns)

gb_1q <- lm(excess_gb_1q ~ mp + SMB + HML + WML, data = split_returns)
gb_2q <- lm(excess_gb_2q ~ mp + SMB + HML + WML, data = split_returns)
gb_3q <- lm(excess_gb_3q ~ mp + SMB + HML + WML, data = split_returns)
gb_4q <- lm(excess_gb_4q ~ mp + SMB + HML + WML, data = split_returns)
```

```r
reg_02 =
  export_summs(fv_1q, fv_2q, fv_3q, fv_4q,
               gb_1q, gb_2q, gb_3q, gb_4q,
               model.names = c("FV 1Q", "FV 2Q", "FV 3Q", "FV 4Q",
                               "GB 1Q", "GB 2Q", "GB 3Q", "GB 4Q"),
               digits = 5,
               coefs = c("Alpha" = "(Intercept)",
                         "Market" = "mp",
                         "HML" = "HML",
                         "SMB" = "SMB",
                         "WML" = "WML"),
               stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.10),
               statistics = c("adj. R squared" = "adj.r.squared",
                              "SE residuals" = "sigma",
                              "N" = "nobs"),
               error_format = "({p.value})")
reg_02

# regression on winner/loser portfolios - PT
pt_h_7_9 <- lm(excess_pt_h_7_9 ~ mp + SMB + HML + WML, data = split_returns)
pt_h_4_6 <- lm(excess_pt_h_4_6 ~ mp + SMB + HML + WML, data = split_returns)
pt_h_0_3 <- lm(excess_pt_h_0_3 ~ mp + SMB + HML + WML, data = split_returns)
pt_l_7_9 <- lm(excess_pt_l_7_9 ~ mp + SMB + HML + WML, data = split_returns)
pt_l_4_6 <- lm(excess_pt_l_4_6 ~ mp + SMB + HML + WML, data = split_returns)
pt_l_0_3 <- lm(excess_pt_l_0_3 ~ mp + SMB + HML + WML, data = split_returns)

reg_03 =
  export_summs(pt_h_7_9, pt_h_4_6, pt_h_0_3,
               pt_l_7_9, pt_l_4_6, pt_l_0_3,
               model.names = c("PT.h79", "Pt.h46", "PT.h03",
                               "PT.l79", "Pt.l46", "PT.hl3"),
               digits = 5,
               coefs = c("Alpha" = "(Intercept)",
                         "Market" = "mp",
                         "HML" = "HML",
                         "SMB" = "SMB",
                         "WML" = "WML"),
               stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.10),
               statistics = c("adj. R squared" = "adj.r.squared",
                              "SE residuals" = "sigma",
                              "N" = "nobs"),
               error_format = "({p.value})")
reg_03


# ---------------------
# --- FV after fees ---
# ---------------------
# Note: this code is only run when market cap limit = 2000 and vinx =
benchmark cap
FV_fees = all_returns %>%
  select(p_year_month, date, excess_fv_adj, nibor_1m, VINX_index)
FV_fees = FV_fees %>% mutate(
  fv_r_fees1 = (excess_fv_adj + nibor_1m) - 0.0125/12,
  vinx = VINX_index / 100)

# Calculating yearly return for FV after mngmt fee and vinx
```

```r
FV_fees = FV_fees %>% mutate(
  fv_r_fees_sumprod = cumprod(fv_r_fees1 + 1))
FV_fees$y_return = ifelse(
  month(FV_fees$date) == 6,
  FV_fees$fv_r_fees_sumprod /
    ifelse(is.na(lag(FV_fees$fv_r_fees_sumprod, n=12)) == F,
           lag(FV_fees$fv_r_fees_sumprod, n=12),1) - 1, NA)
FV_fees$vinx_y_return = ifelse(
  month(FV_fees$date) == 6,
  FV_fees$vinx /
    ifelse(is.na(lag(FV_fees$vinx, n=12)) == F,
           lag(FV_fees$vinx, n=12), 1) - 1, NA)

FV_fees$y_return[FV_fees$p_year_month == "2021 12"] = 5.1761052 / 4.7059891 -
1
FV_fees$vinx_y_return[FV_fees$p_year_month == "2021 12"] = 4.9950946 /
4.6189892 - 1

FV_fees = FV_fees %>% select(p_year_month, y_return, vinx_y_return) %>%
  filter(is.na(y_return)==F)
FV_fees$excess_r = FV_fees$y_return - FV_fees$vinx_y_return

# variable fee is calculated in excel
write_excel_csv(FV_fees, file = "C:/Users/Erik Årstad
Gilje/Dropbox/Rstudio/Master_data/FV_fees.csv")
FV_fees_2 = read.csv("C:/Users/Erik Årstad
Gilje/Dropbox/Rstudio/Master_data/FV_fees_2.csv")
FV_fees = data.frame(
  p_year_month = FV_fees_2$ï..p_year_month,
  v_fee = FV_fees_2$v_fee)

all_returns = all_returns %>% left_join(FV_fees, b = "p_year_month")
all_returns$v_fee[is.na(all_returns$v_fee)==T] = 0
all_returns = all_returns %>% mutate(
  excess_fv_a_fees = excess_fv_adj - (0.0125/12) - v_fee)

# running regression
fv_fee_capm <- lm(excess_fv_a_fees ~ mp, data = all_returns)
fv_fee_ff3 <- lm(excess_fv_a_fees ~ mp + SMB + HML, data = all_returns)
fv_fee_c4f <- lm(excess_fv_a_fees ~ mp + SMB + HML + WML, data = all_returns)

reg_03 = export_summs(fv_fee_capm, fv_fee_ff3, fv_fee_c4f,
                      model.names = c("CAPM", "FF3", "C4F"),
                      digits = 5,
                      coefs = c("Alpha" = "(Intercept)",
                                "Market" = "mp",
                                "HML" = "HML",
                                "SMB" = "SMB",
                                "WML" = "WML"),
                      stars = c(`***` = 0.01, `**` = 0.05, `*` = 0.10),
                      statistics = c("adj. R squared" = "adj.r.squared",
                                     "SE residuals" = "sigma",
                                     "N" = "nobs"),
                      error_format = "({p.value})")
reg_03
```