```
##########################################################################
# The following script reads the submission, numerical, and tag data     #
# to construct a list of standard tags that are commonly used throughout #
# the sample period.                                                     #
##########################################################################
for (year in c(2012:2018)) {

  dir <- "/XBRL/sub/"
  sub <- do.call(rbind,
                 lapply(list.files(paste(dir,year,sep=""),full.names=T),
                        read.table,
                        sep="\t",
                        header=T,
                        fill=T,
                        quote=""))

  dir <- "/XBRL/num/"
  num <- do.call(rbind,
                 lapply(list.files(paste(dir,year,sep=""),full.names=T),
                        read.table,
                        sep="\t",
                        header=T,
                        fill=T,
                        quote=""))

  dir <- "/XBRL/tag/"
  tag <- do.call(rbind,
                 lapply(list.files(paste(dir,year,sep=""),full.names=T),
                        read.table,
                        sep="\t",
                        header=T,
                        fill=T,
                        quote=""))

  sub_num <- merge(sub, num, by = "adsh", all.x = TRUE, all.y = FALSE)
  sub_num <- sub_num[sub_num$coreg == "",]
  sub_num <- sub_num[sub_num$form == "10-K" | sub_num$form == "10-K/A",]
  sub_num <- sub_num[grepl("us-gaap", sub_num$version),]

  sub_num_tag <- merge(sub_num, tag[,c("tag","datatype")], all = F)

  if (year==2012) {
    common_tax <- unique(sub_num_tag$tag)

  } else {
    common_tax <- common_tax[which(common_tax %in%
unique(sub_num_tag$tag))]
  }
}

write.table(common_tax, "common_standard_tags.csv", sep = ",", col.names =
F, row.names = F)

##########################################################################
# The following script constructs a list of XBRL filings that need to    #
# be removed from the sample to retain only those that would have been   #
# available as of portfolio formation date.                              #
##########################################################################
```

```
library(dplyr)
library(lubridate)

dir <- "/XBRL/sub"
sub <- do.call(rbind,
               lapply(list.files(dir, full.names= T),
                      read.table,
                      sep = "\t",
                      header = T,
                      fill = T,
                      quote = ""))

sub <- sub[sub$fp == "FY",]
sub$start_date <- as.Date(as.character(sub$period), format="%Y%m%d") %m+%
months(3)

# Filings submitted after portfolio formation
late <- as.data.frame(sub[as.Date(as.character(sub$filed), format="%Y%m%d")
> sub$start_date,])

# Stale filings (i.e., filings that are subsequently amended)
stale <- as.data.frame(sub[as.Date(as.character(sub$filed),
format="%Y%m%d") <= sub$start_date,])
stale <- stale %>%
  ungroup() %>%
  group_by(cik, fy) %>%
  mutate(amended = sum(prevrpt))
stale <- stale[stale$amended != 0,]
stale <- stale %>%
  ungroup() %>%
  group_by(cik, fy) %>%
  arrange(cik, fy, desc(filed)) %>%
  mutate(most_recent = first(filed))
stale <- stale[stale$filed != stale$most_recent,]
stale$amended <- NULL
stale$most_recent <- NULL

remove <- rbind(late, as.data.frame(stale))
remove <- rbind(remove, sub[!grepl("^10-K", sub$form) & sub$fp == "FY",])
remove <- remove[!duplicated(remove[c("adsh")]),]

write.csv(remove, "point_in_time.csv", row.names = FALSE)

#####################################################
# The following script constructs the OP variables #
# by following Ou and Penman (1989)                #
#####################################################
library(plm)
op <- read.csv("compustat_op.csv", header = T, stringsAsFactors = F)
op <- pdata.frame(op, index = c("GVKEY", "fyear"), drop.index = F,
row.names = F)

# 1. Current ratio
op$cr <- op$act/op$lct

# 2. % change in current ratio
op$cr_d <- lag(op$cr, k = 1)
op$pd_cr <- (op$cr - op$cr_d) / abs(op$cr_d)
```

```
# 3. Quick ratio
op$qr <- (op$che + op$rect) / op$lct

# 4. % change in quick ratio
op$qr_d <- lag(op$qr, k = 1)
op$pd_qr <- (op$qr - op$qr_d) / abs(op$qr_d)

# 5. Days sales in accounts receivable
op$rect_d <- lag(op$rect, k = 1)
op$days_sales_rect <- (((op$rect + op$rect_d)/2) / abs(op$sale)) * 365

# 6. % change in days sales in AR
op$days_sales_rect_d <- lag(op$days_sales_rect, k = 1)
op$pd_days_sales_rect <- (op$days_sales_rect - op$days_sales_rect_d) /
abs(op$days_sales_rect_d)

# 7. Inventory turnover
op$invt_d <-lag(op$invt, k = 1)
op$it <- op$cogs / ((op$invt_d + op$invt)/2)

# 8. % change in inventory turnover
op$it_d <- lag(op$it, k = 1)
op$pd_it <- (op$it - op$it_d) / abs(op$it_d)

# 9. Inventory/total assets
op$invt_at <- op$invt / op$at

# 10. % change in (inventory/assets)
op$invt_at_d <- lag(op$invt_at, k = 1)
op$pd_invt_at <- (op$invt_at - op$invt_at_d) / abs(op$invt_at_d)

# 11. % change in inventory
op$pd_invt <- (op$invt - op$invt_d) / abs(op$invt_d)

# 12. % change in sales
op$sale_d <- lag(op$sale, k = 1)
op$pd_sale <- (op$sale - op$sale_d) / abs(op$sale_d)

# 13. % change in depreciation
op$dp_d <- lag(op$dp, k = 1)
op$pd_dp <- (op$dp - op$dp_d) / abs(op$dp_d)

# 14. Change in dividend per share
op$dvpsp_f_d <- lag(op$dvpsp_f, k = 1)
op$d_dvpsp_f <- (op$dvpsp_f - op$dvpsp_f_d) / op$dvpsp_f_d

# 15. Depreciation/plant assets
op$dp_ppent <- op$dp / op$ppent

# 16. % change in (depreciation/plant assets)
op$dp_ppent_d <- lag(op$dp_ppent, k = 1)
op$pd_dp_ppent <- (op$dp_ppent - op$dp_ppent_d) / abs(op$dp_ppent_d)

# 17. Return on opening equity
op$seq_d <- lag(op$seq, k = 1)
op$roe_begin <- op$ni / op$seq_d
```

```
# 18. Change in return on beginning equity
op$roe_begin_d <- lag(op$roe_begin, k = 1)
op$d_roe_begin <- op$roe_begin - op$roe_begin_d

# 19. % change in (capital expenditure/total assets)
op$capx_at <- op$capx / op$at
op$capx_at_d <- lag(op$capx_at, k = 1)
op$pd_capx_at <- (op$capx_at - op$capx_at_d) / abs(op$capx_at_d)

# 20. One-year lag of % change in (capital expenditure/total assets)
op$pd_capx_at_d <- lag(op$pd_capx_at, k = 1)

# 21. Debt-equity ratio
op$debt_seq <- (op$dltt + op$dlc) / op$seq

# 22. % change in debt-equity ratio
op$debt_seq_d <- lag(op$debt_seq, k = 1)
op$pd_debt_seq <- (op$debt_seq - op$debt_seq_d) / abs(op$debt_seq_d)

# 23. LT debt to equity
op$dltt_seq <- op$dltt / op$seq

# 24. % change in (LT debt to equity)
op$dltt_seq_d <- lag(op$dltt_seq, k = 1)
op$pd_dltt_seq <- (op$dltt_seq - op$dltt_seq_d) / abs(op$dltt_seq_d)

# 25. Equity to fixed assets
op$seq_ppent <- op$seq / op$ppent

# 26. % change in (equity to fixed assets)
op$seq_ppent_d <- lag(op$seq_ppent, k = 1)
op$pd_seq_ppent <- (op$seq_ppent - op$seq_ppent_d) / abs(op$seq_ppent_d)

# 27. Times interest earned
op$times_interest_earned <- op$ebit / op$xint

# 28. % change in (times interest earned)
op$times_interest_earned_d <- lag(op$times_interest_earned, k = 1)
op$pd_times_interest_earned <- (op$times_interest_earned -
op$times_interest_earned_d) / abs(op$times_interest_earned_d)

# 29. Sales/total assets
op$sale_at <- op$sale / op$at

# 30. % change in (sales/total assets)
op$sale_at_d <- lag(op$sale_at, k = 1)
op$pd_sale_at <- (op$sale_at - op$sale_at_d) / abs(op$sale_at_d)

# 31. Return on total assets
op$at_d <- lag(op$at, k = 1)
op$roa <- op$ni / ((op$at + op$at_d) / 2)

# 32. Return on closing equity
op$roe_ending <- op$ni / op$seq

# 33. Gross margin ratio
op$gp_revt <- op$gp / op$revt
```

```
# 34. % change in gross margin ratio
op$gp_revt_d <- lag(op$gp_revt, k = 1)
op$pd_gp_revt <- (op$gp_revt - op$gp_revt_d) / abs(op$gp_revt_d)

# 35. Operating profit (before depreciation) to sales
op$oibdp_sale <- op$oibdp / op$sale

# 36. % change in (operating profit (before depreciation) to sales)
op$oibdp_sale_d <- lag(op$oibdp_sale, k = 1)
op$pd_oibdp_sale <- (op$oibdp_sale - op$oibdp_sale_d) /
abs(op$oibdp_sale_d)

# 37. Pretax income to sales
op$pi_sale <- op$pi / op$sale

# 38. % change in (pretax income to sales)
op$pi_sale_d <- lag(op$pi_sale, k = 1)
op$pd_pi_sale <- (op$pi_sale - op$pi_sale_d) / abs(op$pi_sale_d)

# 39. Net profit margin
op$ni_revt <- op$ni / op$revt

# 40. % change in net profit margin
op$ni_revt_d <- lag(op$ni_revt, k = 1)
op$pd_ni_revt <- (op$ni_revt - op$ni_revt_d) / abs(op$ni_revt_d)

# 41. Sales to total cash
op$sale_ch <- op$sale / op$ch

# 42. Sales to accounts receivable
op$sale_rect <- op$sale / op$rect

# 43. Sales to inventory
op$sale_invt <- op$sale / op$invt

# 44. % change in (sales to inventory)
op$sale_invt_d <- lag(op$sale_invt, k = 1)
op$pd_sale_invt <- (op$sale_invt - op$sale_invt_d) / abs(op$sale_invt_d)

# 45. Sales to working capital
op$sale_wcap <- op$sale / op$wcap

# 46. % change in (sales to working capital)
op$sale_wcap_d <- lag(op$sale_wcap, k = 1)
op$pd_sale_wcap <- (op$sale_wcap - op$sale_wcap_d) / abs(op$sale_wcap_d)

# 47. Sales to fixed assets
op$sale_ppent <- op$sale / op$ppent

# 48. % change in production
op$production <- op$invt - op$invt_d + op$cogs
op$production_d <- lag(op$production, k = 1)
op$pd_production <- (op$production - op$production_d) /
abs(op$production_d)

# 49. % change in R&D
op$xrd_d <- lag(op$xrd, k = 1)
op$pd_xrd <- (op$xrd - op$xrd_d) / abs(op$xrd_d)
```

```
# 50. % change in (R&D/sales)
op$xrd_sale <- op$xrd / op$sale
op$xrd_sale_d <- lag(op$xrd_sale, k = 1)
op$pd_xrd_sale <- (op$xrd_sale - op$xrd_sale_d) / abs(op$xrd_sale_d)

# 51. % change in advertising expense
op$xad_d <- lag(op$xad, k = 1)
op$pd_xad <- (op$xad - op$xad_d) / abs(op$xad_d)

# 52. % change in (advertising/sales)
op$xad_sale <- op$xad / op$sale
op$xad_sale_d <- lag(op$xad_sale, k = 1)
op$pd_xad_sale <- (op$xad_sale - op$xad_sale_d) / abs(op$xad_sale_d)

# 53. % change in total assets
op$pd_at <- (op$at - op$at_d) / abs(op$at_d)

# 54. Cash flow to total debt
op$lt_d <- lag(op$lt, k = 1)
op$cf <- op$ni - (op$at - op$at_d) + (op$lt - op$lt_d) + (op$seq -
op$seq_d)
op$cf_debt <- op$cf / (op$dltt + op$dlc)

# 55. Working capital/total assets
op$wcap_at <- op$wcap / op$at

# 56. % change in (working capital/total assets)
op$wcap_at_d <- lag(op$wcap_at, k = 1)
op$pd_wcap_at <- (op$wcap_at - op$wcap_at_d) / abs(op$wcap_at_d)

# 57. Operating income/total assets
op$oibdp_at <- op$oibdp / op$at

# 58. % change in (operating income/total assets)
op$oibdp_at_d <- lag(op$oibdp_at, k = 1)
op$pd_oibdp_at <- (op$oibdp_at - op$oibdp_at_d) / abs(op$oibdp_at_d)

# 59. % change in total uses of funds # Not used
#op$fuset_d <- lag(op$fuset, k = 1)
#op$pd_fuset <- (op$fuset - op$fuset_d) / abs(op$fuset_d)

# 60. % change in total sources of funds # Not used
#op$fsrct_d <- lag(op$fsrct, k = 1)
#op$pd_fsrct <- (op$fsrct - op$fsrct_d) / abs(op$fsrct_d)

# 61. Repayment of LT debt as % of total LT debt
op$dltt_d <- lag(op$dltt, k = 1)
op$repay_dltt <- op$dltt_d + op$dltis - op$dltt
op$p_repay_dltt <- op$repay_dltt / op$dltt

# 62. Issuance of LT debt as % of total LT debt
op$p_dltis_dltt <- op$dltis / op$dltt

# 63. Purchase of treasury stock as % of stock
op$treasury <- op$tstkc + op$tstkp
op$treasury_d <- lag(op$treasury, k = 1)
op$d_treasury <- op$treasury - op$treasury_d
```

```
op$p_d_treasury_stock <- op$d_treasury / (op$ceq + op$pstk)

# 64. % change in funds # Not used
#op$utfdoc_d <- lag(op$utfdoc, k = 1)
#op$pd_utfdoc <- (op$utfdoc - op$utfdoc_d) / abs(op$utfdoc_d)

# 65. % change in LT debt
op$pd_dltt <- (op$dltt - op$dltt_d) / abs(op$dltt_d)

# 66. Cash dividend as % of cash flows
op$dv_cf <- op$dv / op$cf

# 67. % change in working capital
op$wcap_d <- lag(op$wcap, k = 1)
op$pd_wcap <- (op$wcap - op$wcap_d) / abs(op$wcap_d)

# 68. Net income over cash flows
op$ni_cf <- op$ni / op$cf

op_var <- c("cr","pd_cr","qr","pd_qr",
            "days_sales_rect","pd_days_sales_rect","it",
            "pd_it","invt_at","pd_invt_at","pd_invt",
            "pd_sale","pd_dp","d_dvpsp_f","dp_ppent",
            "pd_dp_ppent","roe_begin","d_roe_begin",
            "pd_capx_at","pd_capx_at_d","debt_seq",
            "pd_debt_seq","dltt_seq","pd_dltt_seq",
            "seq_ppent","pd_seq_ppent",
            "times_interest_earned",
            "pd_times_interest_earned","sale_at",
            "pd_sale_at","roa","roe_ending","gp_revt",
            "pd_gp_revt","oibdp_sale","pd_oibdp_sale",
            "pi_sale","pd_pi_sale","ni_revt","pd_ni_revt",
            "sale_ch","sale_rect","sale_invt",
            "pd_sale_invt","sale_wcap","pd_sale_wcap",
            "sale_ppent","pd_production","pd_xrd",
            "pd_xrd_sale","pd_xad","pd_xad_sale","pd_at",
            "cf_debt","wcap_at","pd_wcap_at","oibdp_at",
            "pd_oibdp_at","p_repay_dltt","p_dltis_dltt",
            "p_d_treasury_stock","pd_dltt","dv_cf",
            "pd_wcap","ni_cf")

write.csv(op[,c("cik","fyear",op_var)], "op_var.csv", row.names = F)

#####################################################
# The following script constructs the DuPont       #
# variables by following Nissim and Penman (2001). #
#####################################################
np <- read.csv("compustat_np.csv", header = T)

np[is.na(np$ivao),"ivao"] <- 0
np[is.na(np$dltt),"dltt"] <- 0
np[is.na(np$dlc),"dlc"] <- 0

np$msr <- NA
np[!is.na(np$mii) & np$mii<0, "msr"] <- 1
np[!is.na(np$mib) & np$mib<0, "msr"] <- 1
```

```
np <- plm::pdata.frame(np, index = c("LPERMNO","fyear"), drop.index = F,
row.names = F)
np$msa_1L <- plm::lag(np$msa, 1)
np$csa <- np$msa - np$msa_1L

np$cni <- np$ni - np$dvp + np$csa

np[!is.na(np$cni) & np$cni<0, "msr"] <- 1

np[is.na(np$ni),"ni"] <- 0
np[is.na(np$dvp),"dvp"] <- 0
np[is.na(np$csa),"csa"] <- 0
np[is.na(np$cni), "cni"] <- np[is.na(np$cni),]$ni - np[is.na(np$cni),]$dvp
+ np[is.na(np$cni),]$csa
np[!is.na(np$cni) & np$cni<0, "msr"] <- 1

np[is.na(np$pstk),"pstk"] <- 0

np[is.na(np$tstkp),"tstkp"] <- 0

np$cse <- np$ceq + np$tstkp - np$dvpa

np[!is.na(np$cse) & np$cse<0, "msr"] <- 1

np[is.na(np$mib),"mib"] <- 0

np[is.na(np$msr), "msr"] <- (np[is.na(np$msr),]$cni/(np[is.na(np$msr),]$cni
+ np[is.na(np$msr),]$mii))/(np[is.na(np$msr),]$cse/(np[is.na(np$msr),]
$cse+np[is.na(np$msr),]$mib))

np[is.na(np$xint),"xint"] <- 0
np[is.na(np$idit),"idit"] <- 0
np[is.na(np$dvp),"dvp"] <- 0
np[is.na(np$msa),"msa"] <- 0
np[is.na(np$esub),"esub"] <- 0
np[is.na(np$mii),"mii"] <- 0
np[is.na(np$recta),"recta"] <- 0

np$mtr <- ifelse(np$fyear==2018, 0.22, 0.37)

np$nfo <- np$dlc + np$dltt + np$pstk - np$tstkp + np$dvpa - np$che -
np$ivao

np$core_nfe <- ((np$xint - np$idit) * (1-np$mtr)) + np$dvp

np$msa_1L <- plm::lag(np$msa, 1)
np$d_msa <- np$msa - np$msa_1L

np$nfe <- np$core_nfe - np$d_msa

np$oa <- np$at - np$che - np$ivao
np$oa_1L <- plm::lag(np$oa, 1)
np$avg_oa <- (np$oa + np$oa_1L)/2
np$noa <- np$nfe + np$cni + np$mii
np$ol <- np$avg_oa - np$noa
np$ol_1L <- plm::lag(np$ol, 1)
np$avg_ol <- (np$ol + np$ol_1L)/2
np$ollev <- np$avg_ol/np$noa
```

```
np$ollev_1L <- plm::lag(np$ollev, 1)
np$d_ollev <- np$ollev - np$ollev_1L

np$cse_1L <- plm::lag(np$cse, 1)
np$avg_cse <- (np$cse + np$cse_1L)/2

np$nfo_1L <- plm::lag(np$nfo, 1)
np$avg_nfo <- (np$nfo + np$nfo_1L)/2

np$flev <- np$avg_nfo/np$avg_cse

np$nbc <- np$nfe/np$avg_nfo

np$noa_1L <- plm::lag(np$noa, 1)

np <- np[!is.na(np$noa_1L),]

np$oi <- np$nfe + np$cni + np$mii
np$rnoa <- np$oi/((np$noa+np$noa_1L)/2)

np$spread <- np$rnoa - np$nbc

np <- np[!is.na(np$rnoa),]

np$oi_from_sales <- np$oi - np$esub
np$pm <- np$oi_from_sales/np$sale

np$ato <- np$sale/((np$noa+np$noa_1L)/2)

np <- np[!is.na(np$pm),]

np$pm_L1 <- plm::lag(np$pm, 1)

np$ato_L1 <- plm::lag(np$ato, 1)

np$d_pm <- np$pm - np$pm_L1

np$d_ato <- np$ato - np$ato_L1

np$rnoa_l <- plm::lead(np$rnoa, 1)

np$d_rnoa <- np$rnoa_l - np$rnoa

np$d_rnoa_L1 <- plm::lag(np$d_rnoa, 1)

np$d_noa <- (np$noa - np$noa_1L)/np$noa_1L

np$other_noa <- np$esub/np$noa

np$flev_1L <- plm::lag(np$flev, 1)
np$spread_1L <- plm::lag(np$spread, 1)
np$nbc_1L <- plm::lag(np$nbc, 1)
np$other_noa_1L <- plm::lag(np$other_noa, 1)

np$d_flev <- np$flev - np$flev_1L
np$d_spread <- np$spread - np$spread_1L
np$d_nbc <- np$nbc - np$nbc_1L
np$d_other_noa <- np$other_noa - np$other_noa_1L
```

```
np$msr_1L <- plm::lag(np$msr, 1)
np$d_msr <- np$msr - np$msr_1L

np_var <- c("msr","d_msr","rnoa","pm","d_pm",
            "ato","d_ato","other_noa","d_other_noa",
            "d_rnoa_L1","nbc","d_nbc","flev","d_flev",
            "ollev","d_ollev")

write.csv(np[,c("cik","fyear",np_var)], "np_var.csv", row.names = F)

##########################################################################
# The following script reads the submission and numerical data to        #
# to construct the final sample matrix containing the current, lagged,   #
# percentage change predictors.                                          #
##########################################################################
library(lubridate)
library(zoo)
library(data.table)
library(dplyr)
library(SparkR)

# Step 1: Reading the submission and numerical data, and retaining only the
# observations with a non-missing reporting period ending on after June 15,
2012
dir <- "/XBRL/sub"
sub <- do.call(rbind,
               lapply(list.files(dir, full.names= T),
                      read.table,
                      sep = "\t",
                      header = T,
                      fill = T,
                      quote = ""))

dir <- "/XBRL/num"
num <- do.call(rbind,
               lapply(list.files(dir, full.names= T),
                      read.table,
                      sep = "\t",
                      header = T,
                      fill = T,
                      quote = ""))
sub <- sub[sub$period>=20120615,]

# Step 2: Merging the submission and numerical data, retaining 10-K and 10-
K/A,
# and retaining only numerical items with a U.S. GAAP tag and that are
# attributed to a consolidated entity
sub_num <- merge(num, sub, by = "adsh", all = FALSE)
sub_num <- sub_num[sub_num$form == "10-K" | sub_num$form == "10-K/A",]
sub_num <- sub_num[grepl("us-gaap", sub_num$version),]
sub_num <- sub_num[sub_num$coreg == "",]

mtrx <- data.table(adsh= character(length(unique(sub_num$adsh))),
                   cik = integer(length(unique(sub_num$adsh))),
                   name = character(length(unique(sub_num$adsh))),
                   sic = integer(length(unique(sub_num$adsh))),
```

```
                    period =
as.Date(character(length(unique(sub_num$adsh))),format="%Y%m%d"),
                    fy = integer(length(unique(sub_num$adsh))),
                    stringsAsFactors = FALSE)
mtrx$adsh <- sub_num[!duplicated(sub_num[c("adsh")]),]$adsh
mtrx$cik <- sub_num[!duplicated(sub_num[c("adsh")]),]$cik
mtrx$name <- sub_num[!duplicated(sub_num[c("adsh")]),]$name
mtrx$sic <- sub_num[!duplicated(sub_num[c("adsh")]),]$sic
mtrx$period <- sub_num[!duplicated(sub_num[c("adsh")]),]$period
mtrx$fy <- sub_num[!duplicated(sub_num[c("adsh")]),]$fy

setkey(mtrx, adsh, physical = TRUE)
sub_num$time_diff <-
as.yearmon(strptime(as.character(sub_num$period),format="%Y%m%d"))-
as.yearmon(strptime(as.character(sub_num$ddate),format="%Y%m%d"))
sub_num <- sub_num[sub_num$time_diff>=0,]
sub_num <- sub_num[sub_num$time_diff%%1==0,]
sub_num <- sub_num[sub_num$time_diff<=1,]
sub_num$tag <- paste(sub_num$tag,sub_num$time_diff,sep="_")

for (i in 1:nrow(sub_num)) {
  if (sub_num$tag[i] %in% names(mtrx)) {
    mtrx[adsh ==
as.character(sub_num$adsh[i]),c(sub_num$tag[i]):=sub_num$value[i]]
  }
}

# Step 3: Importing and merging the pro forma earnings data from IBES
ibes <- read.csv("ibes_pro_forma.csv", header = T, stringsAsFactors = F)
mtrx <- merge(mtrx, ibes, by = c("TICKER","period"), all = F)

# Step 4: Importing and merging with the Compustat data, analyst forecasts,
and CRSP data
op <- read.csv("op_var.csv", header = T, stringsAsFactors = F)
mtrx <- merge(mtrx, op, by = c("cik","period"), all = F)
np <- read.csv("np_var.csv", header = T, stringsAsFactors = F)
mtrx <- merge(mtrx, np, by = c("cik","period"), all = F)
af <- read.csv("af.csv", header = T)
mtrx$forecast_period <- as.Date(as.character(mtrx$period), format =
"%Y%m%d") %m+% months(12)
mtrx <- merge(mtrx, af, by = c("TICKER","forecast_period"), all = F)
crsp <- read.csv("crsp.csv", header = T)
mtrx <- merge(mtrx, crsp, by = c("cik","period"), all = F)

# Step 5: Requiring non-zero total assets
mtrx <- mtrx[!is.na(mtrx$Assets_0) & !is.na(mtrx$Assets_1),]
mtrx <- mtrx[mtrx$Assets_0 != 0 & mtrx$Assets_1 != 0,]

# Step 6: Merging 10-Ks and 10-K/As
mtrx <- mtrx %>% ungroup() %>% group_by(cik, fy) %>% arrange(cik, fy,
desc(filed)) %>%
  fill(names(mtrx)[which(grepl("_0$|_1$", names(mtrx)))], .direction =
"up")
pit <- read.csv("point_in_time.csv", header = T, stringsAsFactors = F)
mtrx <- mtrx[!(mtrx$adsh %in% pit$adsh),]

# Step 7: Keeping only those tags that are used at least once each year
from 2012 to 2018
```

```
tax <- read.csv("common_standard_tags.csv", header = F, stringsAsFactors =
F)
mtrx[,names(mtrx)[which(grepl("_0$|_1$", names(mtrx)) & (!names(mtrx) %in%
tax$tag))]] <- NULL

# Step 8:
# Selecting and scaling the current and lagged predictors by total assets
scols <- names(mtrx)[which(grepl("_0$", names(mtrx)) & !grepl("Assets_0",
names(mtrx)))]
scols <- scols[which(!substr(scols, 1, nchar(scols)-2) %in%
tax[tax$datatype=="perShare",]$tag)]
mtrx[, scols] <- mtrx[, scols]/mtrx[,"Assets_0"]
scols <- names(mtrx)[which(grepl("_1$", names(mtrx)) & !grepl("Assets_1",
names(mtrx)))]
scols <- scols[which(!substr(scols, 1, nchar(scols)-2) %in%
tax[tax$datatype=="perShare",]$tag)]
mtrx[, scols] <- mtrx[, scols]/mtrx[,"Assets_1"]

# Creating the annual percentage changes
feature_name <- names(mtrx)[!names(mtrx) %in% c("adsh", "cik", "name",
"sic", "period", "fy")]
feature_name <- feature_name[grepl("_0$", feature_name)]
pd_feature_name <- character()
for (feature in feature_name) {
  pd_feature_name <- append(pd_feature_name, paste("pd_", feature, sep =
""))
}
feature_prior_name <- character()
for (pd_feature in pd_feature_name) {
  feature <- substr(pd_feature, 4, nchar(pd_feature))
  feature_prior_name <- append(feature_prior_name,
                               paste(substr(pd_feature, 4, regexpr("_0",
pd_feature)),
                                     as.character(1), sep = ""))
}
for (i in 1:length(pd_feature_name)) {
  pd_feature <- c(pd_feature_name[i])
  feature <- c(feature_name[i])
  feature_prior <- c(feature_prior_name[i])
  mtrx[,pd_feature] <- (mtrx[,feature]-mtrx[,feature_prior])/
((mtrx[,feature]+mtrx[,feature_prior])/2)
}

# Step 9: Filling in zeros for the missing values
for (j in names(mtrx)) {
  set(mtrx,which(is.nan(mtrx[[j]])),j,0)
  set(mtrx,which(is.na(mtrx[[j]])),j,0)
}

mtrx <- mtrx[,c("adsh","inc_eps","sar","af",
                names(mtrx)[which(grepl("_0$|_1$", names(mtrx)))],
                op_var,np_var)]

mtrx$inc_eps <- as.factor(mtrx$inc_eps)
mtrx <- mtrx %>% mutate(inc_eps = factor(inc_eps, labels =
make.names(levels(inc_eps))))

write.csv(mtrx, "finalMatrix.csv", row.names = F)
```

```
##########################################################################
# The following script runs the machine learning models.                 #
# Note that some parts are run in parallel to expedite the process.      #
##########################################################################
library(gbm)
library(randomForest)
library(caret)
library(dplyr)
library(pROC)
library(tidyverse)

# Note that the following codes are run using parallel computing for each
model
tid <- as.integer(Sys.getenv("SLURM_ARRAY_TASK_ID"))

mtrx <- read.csv("finalMatrix.csv", header = TRUE, stringsAsFactors = F)

# Splitting the sample into training, validation, and testing sample
training <-
mtrx[grepl(paste("-",paste(tid,tid+1,tid+2,sep="-|-"),"-",sep=""),
mtrx$adsh),]
rownames(training) <- NULL

testing <- mtrx[grepl(paste("-",tid+3,"-",sep=""), mtrx$adsh),]

training_index <- list()
validation_index <- list()
training_index[[1]] <-
as.integer(rownames(training[grepl(paste("-",paste(tid,tid+1,sep="-|-"),"-",sep=""),
training$adsh),]))
validation_index[[1]] <-
as.integer(rownames(training[grepl(paste("-",tid+2,"-",sep=""),
training$adsh),]))
names(training_index) <- c("Fold1")
names(validation_index) <- c("Fold1")

# Setting the training control
ctrl <- trainControl(method = "cv",
                     number = 1,
                     index = training_index,
                     indexOut = validation_index,
                     summaryFunction= twoClassSummary,
                     classProbs = TRUE)

################## Stochastic Gradient Boosting ####################
# Setting up the hyperparameter search grid
gbmGrid <- expand.grid(interaction.depth = c(1:4),
                       n.trees = seq(500,2000,100),
                       shrinkage = c(0.005,0.01,0.05),
                       n.minobsinnode = c(10))

# Running the model
set.seed(135)
mod <- train(inc_eps ~. -adsh -sar -op_var -np_var -af,
             data = training,
             method = "gbm",
             trControl = ctrl,
```

```
             metric = "ROC",
             bag.fraction = 0.5,
             verbose = FALSE,
             tuneGrid = gbmGrid)

mod$xlevels <- NULL

# Variable importance
coeflist <- data.frame(summary(mod, method = permutation.test.gbm,
normalize = FALSE))
##################################################################

##################### Random Forest ##############################
# Setting up the hyperparameter search grid
mtry_grid <- expand.grid(.mtry = c(110, 115, floor(sqrt(sum(grepl("_\\d{1}
$",names(mtrx))))), 120))

# Note that the randomForest package does not offer a way to tune ntree and
nodesize.
# Hence, the following script needs to be run multiple times with different
values of ntree.
mod_perf <- data.frame()
for (size in c(1:4)) {

  set.seed(135)
  mod <- train(inc_eps ~. -adsh -sar -op_var -np_var -af,
               data = training,
               method = "rf",
               trControl = ctrl,
               metric = "ROC",
               ntree = 500,
               replace = F,
               sampsize = 0.5*nrow(training),
               nodesize = size,
               importance = T,
               tuneGrid = mtry_grid)

  mod_perf <- rbind(mod_perf, (cbind(data.frame(nodesize = rep(size, 6)),
mod$results)))
}

mod_perf <- mod_perf %>% arrange(desc(ROC))

set.seed(135)
mod <- train(inc_eps ~. -adsh -sar -op_var -np_var -af,
             data = training,
             method = "rf",
             trControl = ctrl,
             metric = "ROC",
             nodesize = mod_perf[1,"nodesize"],
             ntree = 500,
             replace = F,
             sampsize = 0.5*nrow(training),
             importance = T,
             tuneGrid = expand.grid(.mtry = c(mod_perf[1,"mtry"])))

mod$xlevels <- NULL
```

```
# Variable importance
# Note that the following caret function for variable importance calls
# on the underlying randomForest function importance(,type=1,scale=F)
coeflist <- varImp(mod,type=1,scale=F)$importance %>%
mutate(var=row.names(.)) %>% arrange(desc(Overall))
##################################################################

final <- data.frame("adsh" = testing$adsh,
                    "actual_pos" = testing$inc_eps,
                    "pred_prob" = predict(mod, newdata = testing, type =
"prob")[,2],
                    "sar" = testing$sar)

final$pred_pos_55 <- ifelse(final$pred_prob>0.5, 1, 0)
final$pred_pos_64 <- NA
final[final$pred_prob>=0.6,"pred_pos_64"] <- 1
final[final$pred_prob<=0.4,"pred_pos_64"] <- 0

final %>% ungroup() %>% group_by(pred_pos_55) %>% summarize(mean(sar))
final %>% ungroup() %>% group_by(pred_pos_64) %>% summarize(mean(sar))

# Bootstrapped pseudo excess returns (n = 10,000)
max_iteration <- 10000

bootstrap <- data.frame(iteration = c(1:max_iteration),
                        bootstrap_55_sar = numeric(max_iteration),
                        bootstrap_64_sar = numeric(max_iteration))

set.seed(10)
for (i in 1:max_iteration) {

  for (threshold in c("pred_pos_55","pred_pos_64")) {

    random_pos <- sample(1:nrow(final), sum(final[,threshold]==1, na.rm =
TRUE), replace = TRUE)
    random_neg <- sample(1:nrow(final), sum(final[,threshold]==0, na.rm =
TRUE), replace = TRUE)

    if (grepl("_55", threshold)) {
      random_dist[i, "bootstrap_55_sar"] <- mean(final[random_pos, "sar"])
- mean(final[random_neg, "sar"])

    } else if (grepl("_64", threshold)) {
      random_dist[i, "bootstrap_64_sar"] <- mean(final[random_pos, "sar"])
- mean(final[random_neg, "sar"])
    }
  }
}

# For each ML, import all the yearly models and the bootstrapped pseudo
excess returns and aggregate across the testing years
verification::roc.area(as.integer(gsub("X", "",
as.character(final$actual_pos))), final$pred_prob)
sum(bootstrap$mean_bootstrap_55_sar > mean(final$hedge_55_sar))/
nrow(bootstrap)
sum(bootstrap$mean_bootstrap_64_sar > mean(final$hedge_64_sar))/
nrow(bootstrap)
```

```
# Import all the comparison models (e.g. OP/Logit, OP/RF, DuPont/SGB)
temp <- merge(final, compare, by="adsh", all = T)
roc.test(roc(temp$actual_pos, temp$pred_prob, direction = "<"),
         roc(temp$actual_pos, temp$pred_prob_compare, direction = "<"),
         alternative = "great", method = "boot", boot.n = 10000, paired =
T)
```