

```
.start_time <- Sys.time()
cat("==== START TIME =====\n")

## ===== START TIME =====

print(.start_time)

## [1] "2026-02-02 21:37:48 PST"

library(RPostgres)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##   filter, lag

## The following objects are masked from 'package:base':
##   intersect, setdiff, setequal, union

library(data.table)

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##   between, first, last

library(tibble)
library(stringr)
library(farr)

##
## Attaching package: 'farr'

## The following object is masked from 'package:base':
##   truncate

library(lubridate)

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##   hour, isoweek, mday, minute, month, quarter, second, wday, week,
##   yday, year

## The following objects are masked from 'package:base':
##   date, intersect, setdiff, union

library(ggplot2)
library(purrr)

##
## Attaching package: 'purrr'

## The following object is masked from 'package:data.table':
##   transpose

library(tidyverse)

## --- Attaching core tidyverse packages --- tidyverse 2.0.0 ---
## ✓ forcats 1.0.0      ✓ tidyr  1.3.0
## ✓ readr   2.1.5

## --- Conflicts --- tidyverse_conflicts() ---
## x data.table::between() masks dplyr::between()
## x dplyr::filter()       masks stats::filter()
## x data.table::first()  masks dplyr::first()
## x lubridate::hour()    masks data.table::hour()
## x lubridate::isoweek() masks data.table::isoweek()
## x dplyr::lag()         masks stats::lag()
## x data.table::last()   masks dplyr::last()
## x lubridate::mday()    masks data.table::mday()
## x lubridate::minute()  masks data.table::minute()
## x lubridate::month()   masks data.table::month()
## x lubridate::quarter() masks data.table::quarter()
## x purrr::transpose()   masks data.table::transpose()
## x lubridate::wday()     masks data.table::wday()
## x lubridate::week()     masks data.table::week()
## x lubridate::yday()     masks data.table::yday()
## x lubridate::year()     masks data.table::year()
## ! Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(knitr)
library(haven)
library(fixest)
library(readxl)
library(fabricatr)
theme_set(theme_bw())

path = "/Users/qianqianli/Dropbox/BDL_code/"

comphist = readRDS(paste0(path, "1_data_processing/data/control_variables/wrds/RDS/comphist_202508013.RDS"))

df_gics = comphist %>% dplyr::select(gvkey, hchgdt, hchgenddt, hgsector) %>%
  arrange(gvkey, hchgdt)

# fill NA with today's date
today = Sys.Date()
df_gics[which(is.na(df_gics$hchgenddt)), "hchgenddt"] = today

df_1 = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_10k_item_1A_skip_last_step.RDS"))
df_2 = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_10q_item_1A_skip_last_step.RDS"))
df_item_1A = rbind(df_1 %>% dplyr::select(id, GVKEY, datadate),
  df_2 %>% dplyr::select(id, GVKEY, datadate))

df_1 = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_10k_item_7_skip_last_step.RDS"))
df_2 = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_10q_item_7_skip_last_step.RDS"))
df_item_7 = rbind(df_1 %>% dplyr::select(id, GVKEY, datadate),
  df_2 %>% dplyr::select(id, GVKEY, datadate))

df_cc = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_cc_skip_last_step.RDS")) %>%
  dplyr::select(id, GVKEY, datadate)

df_pr = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_pr_skip_last_step.RDS")) %>%
  dplyr::select(id, GVKEY, datadate)

df = rbind(df_item_1A %>% mutate(item = 1),
  df_item_7 %>% mutate(item = 2),
  df_cc %>% mutate(item = 3),
  df_pr %>% mutate(item = 4))

df_indicator = df %>% dplyr::select(GVKEY, datadate) %>% distinct()

df_indicator_gics = df_indicator %>% left_join(df_gics, by = c("GVKEY" = "gvkey")) %>% filter(datadate >= hchgdt, datadate <= hchgenddt)

## Warning in left_join(., df_gics, by = c(GVKEY = "gvkey")): Detected an unexpected many-to-many relationship between `x` and `y`.
## ! Row 1 of `x` matches multiple rows in `y`.
## ! Row 20 of `y` matches multiple rows in `x`.
## ! If a many-to-many relationship is expected, set `relationship =`
## "many-to-many" to silence this warning.

df_indicator_gics = df_indicator %>% left_join(df_indicator_gics, by = c("GVKEY", "datadate"))

# fill in hgsector
df_indicator_gics = df_indicator_gics %>% mutate(hgsector_forward = hgsector,
  hgsector_backward = hgsector) %>%
  group_by(GVKEY) %>% arrange(datadate, .by_group = TRUE) %>%
  fill(hgsector_forward, .direction = "down") %>%
  fill(hgsector_backward, .direction = "up") %>% ungroup() %>%
  # forward fill first, then backward fill (all of them are the first entry so all backward filled)
  mutate(hgsector = ifelse(is.na(hgsector), ifelse(!is.na(hgsector_forward), hgsector_forward, hgsector_backward), hgsector)) %>%
  dplyr::select(GVKEY:hgsector)

# for those still missing, try to find the unique hgsector in the entire df_gics and fill in
df_indicator_gics_miss = df_gics %>% filter(gvkey != na & (df_indicator_gics %>% filter(is.na(hgsector)) %>% pull(GVKEY))) %>%
  filter(!is.na(hgsector)) %>%
  group_by(gvkey) %>% mutate(n = length(unique(hgsector))) %>% filter(n == 1) %>% slice(1) %>% ungroup() %>%
  dplyr::select(gvkey, hgsector)

df_indicator_gics = rbind(df_indicator_gics %>% filter(!is.na(hgsector)),
  df_indicator_gics %>% filter(is.na(hgsector)) %>% dplyr::select(-hgsector) %>%
  left_join(df_indicator_gics_miss, by = c("GVKEY" = "gvkey"))) %>% dplyr::select(GVKEY, datadate, hgsector)

# check: verify whether hgsector is missing for that GVKEY across the entire df_gics
# df_gics %>% filter(gvkey %in% (df_indicator_gics %>% filter(is.na(hgsector)) %>% pull(GVKEY))) %>% filter(!is.na(hgsector)) # 0 rows

df_indicator_gics = df_indicator_gics %>% rename("gics2" = "hgsector")

# save
saveRDS(df_indicator_gics, paste0(path, "1_data_processing/data/control_variables/wrds/RDS/gics_hgsector.RDS"))

.end_time <- Sys.time()
cat("==== END TIME =====\n")

## ===== END TIME =====

print(.end_time)

## [1] "2026-02-02 21:38:05 PST"

cat("Elapsed:", difftime(.end_time, .start_time, units = "secs"), "seconds\n")

## Elapsed: 17.26314 seconds
```