

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

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

print(.start_time)

## [1] "2026-02-02 22:09:21 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(strings)
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 ---
## ✓ forecast 1.10.0 ✓ tidyr 1.3.0
## ✓ readr 2.1.5

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

library(hntr)
library(hovm)
library(fixest)
library(roadr)
library(fabricatr)
theme_set(theme_bw())

path = "~/Users/qiangianli/Dropbox/RDS_code/"
```

with sticky

1. sample

```
item_1A = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_item_1A.RDS")) %>%
  dplyr::select(id, OVKEY, LPERHMO, datadate, fqtr, rdate, fdate, fname, cik)
item_7 = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_item_7.RDS")) %>%
  dplyr::select(id, OVKEY, LPERHMO, datadate, fqtr, rdate, fdate, fname, cik)
cc = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_cc.RDS")) %>%
  dplyr::select(id, OVKEY, LPERHMO, datadate, fqtr, companyid, keydevid, mostimportantdateutc)
pr = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_pr.RDS")) %>%
  dplyr::select(id, OVKEY, LPERHMO, datadate, fqtr, rdate, fdate, fname, cik)
sl = readRDS(paste0(path, "1_data_processing/data/sample_selection/final_sample/sample_sl.RDS"))

# nrow(item_1A) # 105,795
# nrow(item_7) # 139,456
# nrow(cc) # 149,690
# nrow(pr) # 189,647
# nrow(sl) # 5,420
```

2. GenScore

```
df_gptzero = readRDS(paste0(path, "1_data_processing/data/gptzero_full_sample/gptzero_full_sample_2025_06_02_clean.RDS"))
df_gptzero_sl = readRDS(paste0(path, "1_data_processing/data/gptzero_full_sample/gptzero_full_sample_sl_2025_06_02_clean.RDS")) %>% rename("clean" = "clean_or_not")

# dat: in percentage points
df_gptzero = df_gptzero %>% mutate(GenScore = GenScore * 100)
df_gptzero_sl = df_gptzero_sl %>% mutate(GenScore = GenScore * 100)

df_item_1A = item_1A %>% inner_join(df_gptzero %>% filter(ITEM == 1, stickiness == "sticky"), by = c("id", "OVKEY", "datadate"))
df_item_7 = item_7 %>% inner_join(df_gptzero %>% filter(ITEM == 2, stickiness == "sticky"), by = c("id", "OVKEY", "datadate"))
df_cc = cc %>% inner_join(df_gptzero %>% filter(ITEM == 3, stickiness == "sticky"), by = c("id", "OVKEY", "datadate"))
df_pr = pr %>% inner_join(df_gptzero %>% filter(ITEM == 4, stickiness == "sticky"), by = c("id", "OVKEY", "datadate"))
df_sl = sl %>% inner_join(df_gptzero_sl, by = c("filing_name" = "filename"))

length_sl = readRDS(paste0(path, "1_data_processing/data/linguistic_properties/sl_number_of_sentences_clean.RDS")) %>% dplyr::select(filename, type, clean_or_not, number_of_sentences)
df_sl = df_sl %>% inner_join(length_sl, by = c("filing_name" = "filename", "type", "clean" = "clean_or_not"))

# combine GenScore for clean + non-clean text, weighted by number of sentences
df_sl_weighted = df_sl %>% filter(type != "Business", "MGA", "Prospectus Summary", "Risk Factors") %>%
  mutate(weight = number_of_sentences * GenScore) %>%
  group_by(fname, filing_name, cik, fdate, group, type) %>%
  mutate(number_of_sentences = sum(number_of_sentences), weight = sum(weight)) %>%
  mutate(GenScore = weight / number_of_sentences) %>% slice() %>% ungroup() %>% dplyr::select(-clean, -weight) %>%
  dplyr::select(fname|GenScore)

# nrow(df_item_1A) # 105,795
# nrow(df_item_7) # 139,456
# nrow(df_cc) # 149,690
# nrow(df_pr) # 189,647
# nrow(df_sl_weighted) # 20,582
```

3. control variables

```
control_10k_l0q = readRDS(paste0(path, "1_data_processing/data/control_variables/control_variables_10k_l0q_20250707.RDS"))
control_pr_cc = readRDS(paste0(path, "1_data_processing/data/control_variables/control_variables_pr_cc_20250707.RDS"))

df_item_1A = df_item_1A %>% inner_join(control_10k_l0q, by = c("id", "OVKEY", "LPERHMO", "datadate", "fqtr"))
df_item_7 = df_item_7 %>% inner_join(control_10k_l0q, by = c("id", "OVKEY", "LPERHMO", "datadate", "fqtr"))
df_cc = df_cc %>% inner_join(control_pr_cc, by = c("id", "OVKEY", "LPERHMO", "datadate", "fqtr"))
df_pr = df_pr %>% inner_join(control_pr_cc, by = c("id", "OVKEY", "LPERHMO", "datadate", "fqtr"))

# nrow(df_item_1A) # 105,795
# nrow(df_item_7) # 139,456
# nrow(df_cc) # 149,690
# nrow(df_pr) # 189,647
```

4. time indicators

```
df_item_1A = df_item_1A %>% dplyr::select(ITEM, disclosure, id|datadate, fdate, fyear, fqtr, GenScore, size|audit_fees)
df_item_7 = df_item_7 %>% dplyr::select(ITEM, disclosure, id|datadate, fdate, fyear, fqtr, GenScore, size|audit_fees)
df_cc = df_cc %>% dplyr::select(ITEM, disclosure, id|datadate, fdate, fyear, fqtr, GenScore, size|audit_fees) %>% rename("fdate" = "mostimportantdateutc")
df_pr = df_pr %>% dplyr::select(ITEM, disclosure, id|datadate, fyear, fdate, fqtr, GenScore, size|audit_fees)
df = rbind(df_item_1A, df_item_7, df_cc, df_pr)

df = df %>% mutate(post = ifelse(datadate > "2022-11-30", 1, 0),
  year_2024 = ifelse(year(datadate) == 2024, 1, 0),
  year_2023 = ifelse(year(datadate) == 2023, 1, 0),
  late_2022 = ifelse(year(datadate) == 2022 & datadate > "2022-11-30", 1, 0),
  year = year(datadate),
  month = month(datadate),
  qtr = quarter(datadate))

df_sl_weighted = df_sl_weighted %>% mutate(post = ifelse(fdate > "2022-11-30", 1, 0),
  year_2024 = ifelse(year(fdate) == 2024, 1, 0),
  year_2023 = ifelse(year(fdate) == 2023, 1, 0),
  late_2022 = ifelse(year(fdate) == 2022 & fdate > "2022-11-30", 1, 0),
  year = year(fdate),
  month = month(fdate),
  qtr = quarter(fdate))
```

5. size, return and audit fees into deciles, loss variable

size

```
df = df %>% group_by(ITEM, year) %>%
  mutate(tercile = split_quantile(size, 3)) %>% ungroup() %>%
  mutate(small = ifelse(tercile == 1, 1, 0)) %>% dplyr::select(-tercile)
```

return

```
df_large_returns_by_decile = df %>% filter(!is.na(abnormal_return_by_decile)) %>%
  group_by(ITEM, year, fqtr) %>%
  mutate(tercile = split_quantile(abs(abnormal_return_by_decile), 3)) %>% ungroup() %>%
  mutate(large_returns_by_decile = ifelse(tercile == 3, 1, 0)) %>% dplyr::select(ITEM, id, large_returns_by_decile)

df = df %>% left_join(df_large_returns_by_decile, by = c("ITEM", "id"))
```

audit fees

```
df_adjusted_audit_fees = df %>% group_by(ITEM, year) %>%
  mutate(ranktemp_size = ntile(size, 10)) %>% ungroup() %>%
  filter(!is.na(audit_fees)) %>%
  mutate(log_audit_fees = ifelse(audit_fees != 0, log(audit_fees), 0)) %>%
  group_by(ITEM, year, ranktemp_size) %>% mutate(lnaudit_sizeadj = log_audit_fees - mean(log_audit_fees)) %>% ungroup() %>% dplyr::select(-ranktemp_size)

df_small_audit_adjusted = df_adjusted_audit_fees %>% group_by(ITEM, year) %>%
  mutate(tercile = split_quantile(lnaudit_sizeadj, 3)) %>% ungroup() %>%
  mutate(small_audit_adj = ifelse(tercile == 1, 1, 0)) %>% dplyr::select(ITEM, id, small_audit_adj)

df = df %>% left_join(df_small_audit_adjusted, by = c("ITEM", "id"))
```

loss

```
df = df %>% mutate(loss = ifelse(net_income < 0, 1, 0))
```

6. linguistic properties (document level)

```
length_fog = readRDS(paste0(path, "1_data_processing/data/linguistic_properties/length_fog_clean.RDS"))
sentiment = readRDS(paste0(path, "1_data_processing/data/linguistic_properties/sentiment_clean.RDS"))
specificity = readRDS(paste0(path, "1_data_processing/data/linguistic_properties/specificity_clean.RDS"))

df_linguistic = length_fog %>% inner_join(specificity) %>% inner_join(sentiment) %>%
  dplyr::select(ITEM|fog, tone, specificity) %>% filter(stickiness == "sticky") %>% dplyr::select(-stickiness)

## Joining with `by` = join_by(ITEM, disclosure, stickiness, id, OVKEY, datadate)
## Joining with `by` = join_by(ITEM, disclosure, stickiness, id, OVKEY, datadate)
```

```
df = df %>% inner_join(df_linguistic, by = c("ITEM", "disclosure", "id", "OVKEY", "datadate"))
```

7. gics - hgsector

```
gics = readRDS(paste0(path, "1_data_processing/data/control_variables/wrds/RDS/gics_hgsector.RDS"))
df = df %>% inner_join(gics, by = c("OVKEY", "datadate"))
```

save

```
df = df %>% rename("abnormal_return" = "abnormal_return_by_decile",
  "large_returns" = "large_returns_by_decile",
  "low_audit_fee" = "small_audit_adj",
  "item" = "ITEM")

write_dta(df, paste0(path, "3_regression/data/data_with_sticky.dta"))

# pivot_wider
df_sl_weighted_wider = df_sl_weighted %>% dplyr::select(fname|year|post|qtr, GenScore) %>%
  pivot_wider(names_from = type, values_from = GenScore) %>%
  rename("GenScore_pr" = "Prospectus Summary",
    "GenScore_cc" = "Risk Factors",
    "GenScore_bus" = "Business",
    "GenScore_mf" = "MGA")

write_dta(df_sl_weighted_wider, paste0(path, "3_regression/data/data_sl.dta"))
```

non sticky

only need to update GenScore & linguistic properties

2. GenScore

```
df_sticky = read_dta(paste0(path, "3_regression/data/data_with_sticky.dta"))

df_gptzero = readRDS(paste0(path, "1_data_processing/data/gptzero_full_sample/gptzero_full_sample_2025_06_02_clean.RDS"))
df_gptzero_non_sticky = df_gptzero %>% filter(stickiness == "NonSticky") %>% dplyr::select(-stickiness) %>% mutate(GenScore = GenScore * 100)

df_non_sticky = df_sticky %>% dplyr::select(-GenScore, -number_of_words, -number_of_sentences, -fog, -tone, -specificity) %>%
  inner_join(df_gptzero_non_sticky, by = c("item" = "ITEM", "disclosure", "id", "OVKEY", "datadate"))
# nrow(df_non_sticky) # 553,100
```

6. linguistic properties (document level)

```
length_fog = readRDS(paste0(path, "1_data_processing/data/linguistic_properties/length_fog_clean.RDS"))
sentiment = readRDS(paste0(path, "1_data_processing/data/linguistic_properties/sentiment_clean.RDS"))
specificity = readRDS(paste0(path, "1_data_processing/data/linguistic_properties/specificity_clean.RDS"))

df_linguistic = length_fog %>% inner_join(specificity) %>% inner_join(sentiment) %>%
  dplyr::select(ITEM|fog, tone, specificity) %>% filter(stickiness == "NonSticky") %>% dplyr::select(-stickiness)

## Joining with `by` = join_by(ITEM, disclosure, stickiness, id, OVKEY, datadate)
## Joining with `by` = join_by(ITEM, disclosure, stickiness, id, OVKEY, datadate)
```

```
df_non_sticky = df_non_sticky %>% inner_join(df_linguistic, by = c("item" = "ITEM", "disclosure", "id", "OVKEY", "datadate"))
```

save

```
df_non_sticky = df_non_sticky %>% dplyr::select(item|fqtr, GenScore, size|loss, number_of_words|specificity, gics2)
# nrow(df_non_sticky) # 553,100

write_dta(df_non_sticky, paste0(path, "3_regression/data/data_non_sticky.dta"))

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

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

print(.end_time)

## [1] "2026-02-02 22:10:27 PST"

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

## Elapsed: 66.0198 seconds
```