FHK replication

Ian Gow

2020-10-13

Introduction

This document reproduces the results found in Table 15 and 16 of *Reply to "The Reg SHO Reanalysis Project: Reconsidering Fang, Huang and Karpoff (2016) on Reg SHO and Earnings Management" by Black et al. (2019)* (available on SSRN here). The tables provide results for regressions of performance-matched discretionary accruals on a treatment indicator, *PILOT*, two post-treatment indicators, *DURING* and *POST*, and interactions, $PILOT \times DURING$ (*pilot_during*) and $PILOT \times POST$ (*pilot_post*). The coefficient of primary interest is that on $PILOT \times DURING$.

To run this code, you will need:

1. An internet connection. (We get data from WRDS and a website.)
2. A WRDS ID. Before running the code, tell R your WRDS ID and password by running the following line:

```
Sys.setenv(PGUSER="yanhdong", PGPASSWORD="1qaz@WSX")
```

3. To install the libraries used in the following chunk of code. For example, `install.packages("haven")`.
4. **Optional** To have LaTeX software installed (if you want to compile as a PDF). Compiling the document as HTML should not require LaTeX.

```
library(haven)
library(lfe)
library(stargazer)
library(dplyr, warn.conflicts = FALSE)
library(lubridate)
library(stringr)
library(tidyr)
library(broom)
library(DBI)
```

```r
library(ggplot2)


library(knitr)
hook_output = knit_hooks$get('output')
knit_hooks$set(output = function(x, options) {
  # this hook is used only when the linewidth option is not NULL
  if (!is.null(n <- options$linewidth)) {
    x = knitr:::split_lines(x)
    # any lines wider than n should be wrapped
    if (any(nchar(x) > n)) x = strwrap(x, width = n)
    x = paste(x, collapse = '\n')
  }
  hook_output(x, options)
})

Sys.setenv(PGHOST="wrds-pgdata.wharton.upenn.edu", PGDATABASE="wrds", PGPORT=9737)
```

Getting the data

FHK use three data sources:

1. SHO data, which combines data from the SEC with analysis by the authors
2. Compustat
3. Fama-French industry data

SHO data

We can gain insight into the construction of the SHO indicator by using two SAS data files that embed the data used by FHK. The following code reproduces the analysis by FHK. There are effectively two elements: `pilot` provides the `PILOT` indicator and `pmda` identifies the firm-years (`gvkey-fyear` values) to be considered. The file `pilot.sas7bdat` includes variables such as `lpermno` and `linkdt`, suggesting that the WRDS table `crsp.stocknames` was used in its construction.

```r
pilot <- read_sas("data/pilot.sas7bdat")
pmda <- read_sas("data/pmda.sas7bdat")
```

```r
sho_data <-
    pilot %>%
    select(gvkey1, SHO) %>%
    distinct() %>%
    group_by(gvkey1) %>%
    filter(n() == 1) %>%
    ungroup() %>%
    inner_join(pilot, by = c("gvkey1", "SHO")) %>%
    rename(sho = SHO) %>%
    mutate(gvkey = str_pad(gvkey1, width = 6, side = "left", pad = "0")) %>%
    select(gvkey, sho)

# Find firms with more than one 'gvkey'
sho_data2 <-
    pilot %>%
    group_by(gvkey1) %>%
    filter(n() > 1)

sho_firm_years <-
    pmda %>%
    mutate(gvkey = str_pad(gvkey1, width = 6, side = "left", pad = "0")) %>%
    select(-gvkey1) %>%
    filter(fyear >= 2000) %>%
    inner_join(sho_data, by = "gvkey") %>%
    select(gvkey, fyear, datadate, sho)
```

- In constructing the `PILOT` indicator, FHK omit cases (`gvkey1` values) where there is more than one distinct value for the indicator. A question is: Who are these firms? Why is there more than one value for `PILOT` for these firms? And does omission of these make sense?

  A: As the dataframe 'sho_dupes' shows, when one 'gvkey' has multiple tickers, and SEC chose pilot firms based on tickers,there will be more than one distinct value for the indicator. For example, in 2001, USX, the holding company that owned United States Steel and Marathon, spun off the steel business and, in 2002, USX renamed itself Marathon Oil Corporation. The ticker of United States Steel is 'X', and the ticker of Marathon Oil Corporation is 'MRO', but they have the same 'gvkey' '007017'. SEC chose Marathon Oil Corporation as the pilot firm, not United States Steel. It does not make sense to omit these observations. Instead, FHK should choose observations by strictly following SEC's way, i.e., choosing the right ticker and company.

```r
sho_dupes <-
    pilot %>%
    select(gvkey1, SHO) %>%
```

```r
    distinct() %>%
    group_by(gvkey1) %>%
    filter(n() > 1) %>%
    ungroup() %>%
    inner_join(pilot, by = c("gvkey1", "SHO")) %>%
    rename(sho = SHO) %>%
    mutate(gvkey = str_pad(gvkey1, width = 6, side = "left", pad = "0")) %>%
    select(gvkey, lpermno, rsticker, sho) %>%
    arrange(gvkey)

sho_dupes
```

```
## # A tibble: 6 x 4
##   gvkey  lpermno rsticker   sho
##   <chr>    <dbl> <chr>    <dbl>
## 1 007017   15069 MRO          1
## 2 007017   15069 X            0
## 3 030146   46392 ITG          0
## 4 030146   46392 JEF          1
## 5 141400   26382 MEE          0
## 6 141400   26382 FLR          1
```

From the analysis above, we see that tickers `X`, `ITG`, and `MEE` have zero for `sho`, but are associated with `gvkey` (and `permno`) values where `sho` is one for a different ticker. The ticker (`rsticker`) values in `sho.sas7bdat` seem to match those available on the SEC's website.

```r
pg <- dbConnect(RPostgres::Postgres())

stocknames <- tbl(pg, sql("SELECT * FROM crsp.stocknames"))
ccmxpf_linktable <- tbl(pg, sql("SELECT * FROM crsp.ccmxpf_linktable"))
```

Using the tables `crsp.stocknames` and `crsp.ccmxpf_linktable`, my conclusion is that the GVKEY matches for the tickers `JEF`, `FLR`, and `MRO` are simply wrong. Based on my analysis, these are the correct GVKEY matches and all three cases should be included with `sho` equal to one.

Ticker	GVKEY
JEF	006239
FLR	004818
MRO	010970

- Picking one of these tickers, how can you match that ticker to the GVKEY value I have provided? Do you agree with my approach?

  A: I searched Compustat with Ticker and gvkey, and found the Column 1-5 of the table below. I think your match of the ticker 'FLR' is correct, but the matches of 'JEF' and 'MRO' are wrong.

RSNAME	Ticker	GVKEY	First_date	Last_date	GVEKY_FHK	GVKEY_Ian	SEC
Marathon Oil Corp	MRO	007017	1950	2019	007017	010970	Yes
USX Corp-Consolidated	MROX.CM	010970	1950	2000			
(Spun to MRO and X in 2001)							
UNITED STATES STEEL CORP	X	023978	1950	2019	007017		
————	———	———	————	———-	———-	———-	—-
MASSEY ENERGY CORP	MEE	141400	1996	2010	141400		
FLUOR CORP	FLR	004818	1950	2018	141400	004818	Yes
(Spun off MEE in 1999)							
————	———	———	————	———-	———-	———-	—-
INVESTMENT TECHNOLOGY	ITG	030146	1993	2018	030146		
(INVESTMENT TECHNOLOGY GP INC							
JEFFERIES GROUP INC	JEF	006682	1950	2019	030146	006239	Yes
(JEFFERIES FINANCIAL GRP INC)							
JEFFERIES GROUP LLC	LUK2	006239	1982	2019			
(a wholly subsidiary of JEF,							
spun off ITG in 1999)							

Fixing the issue with mismatched GVKEYs (and issues related to which firms should be used as controls) is complicated, so we won't attempt that here.

But another issue is implicit in the code below:

```
sho_data %>%
  count(gvkey) %>%
  arrange(desc(n))
```

```
## # A tibble: 2,992 x 2
##    gvkey      n
##    <chr>  <int>
##  1 001076     2
##  2 002008     2
##  3 002220     2
```

```
##  4 002435     2
##  5 002710     2
##  6 003581     2
##  7 003662     2
##  8 003708     2
##  9 004842     2
## 10 005284     2
## # ... with 2,982 more rows
```

```r
# Delete the duplicated gvkey
sho_data_del<-
    sho_data %>%
    group_by(gvkey) %>%
    filter(row_number() == 1) %>%
    ungroup()

sho_data <- sho_data_del
```

- What is the issue implied in the above? How would you fix this issue?
- Does fixing this issue affect the results in any way? Why or why not?

  A: 'gvkey' is the company-level identifier, but 'lpermno' is the security-level identifier. If a company issues multiple securities, it may have one 'gvkey' and multiple 'lpmermno', and then the count of 'gvkey' will be larger than 1. For example, '001076'(AARON RENTS INC) and '002220'(BIO RAD LABORATORIES INC) both have duall class shares.

  I think FHK should delete the duplicates because FHK studies firm-level earnings management. The duplicates will increase the total observations, but it is uncertain how they will influence the results because it is uncertain about whether the treatment group or the non-treatment group includes more duplicates and whether these duplicates will drive the results.

  I find that fixing this issue does not affect the results.

Fama-French data

I grab this data set directly from Ken French's website.

```r
get_ff_ind <- function(num = 48) {

  # fileext gives the tempfile extension"
    t <- tempfile(fileext = ".zip")
```

```
# If we set the num as 48, then the link is https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/Siccodes48.zip

url <- paste0("https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/Siccodes", num, ".zip")

# Download a zip file from the link"
download.file(url, t)

# The number of industries 1-48, has 3 spaces. The name of the industries has 7 spaces. The characters after the name are defined as '

ff_data <-
    readr::read_fwf(unzip(t),
             col_positions = readr::fwf_widths(c(3, 7, NA),
                                         c("ff_ind", "ff_ind_short_desc", "sic_range")),
            col_types = "icc") %>%

    # Add a column, ff_inde_desc. If the short name of the industries is not null, ff_inde_dec is defined as sic_range, otherwise def
    mutate(ff_ind_desc = if_else(!is.na(ff_ind_short_desc), sic_range, NA_character_)) %>%

  # Fill missing values. Here, ff_ind, ff_ind_short_desc, ff_ind_desc are missing except for the first line of 48 industries. Fill the
    tidyr::fill(ff_ind, ff_ind_short_desc, ff_ind_desc) %>%

  # Choose lines in which sic_range startes with a number, 0 to 9. Thus,the first line of each industry is deleted because sic_range c
  filter(grepl("^[0-9]", sic_range)) %>%

  # Extract sic_range into three columns "sic_min", "sic_max", "sic_desc". For example, '0100-0199 Agricultural production - crops', (
  tidyr::extract(sic_range,
            into = c("sic_min", "sic_max", "sic_desc"),
            regex = "^([0-9]+)-([0-9]+)(.*)$",
            convert = TRUE) %>%

  # Remove white space before sic_desc, but I  do not think we need this line of code because we do not use sic_desc later.
    mutate(sic_desc = trimws(sic_desc)) %>%

  # Add a column ff_ind_category, ff_48, but I  do not think we need this line of code because we do not use ff_ind_category later and
    mutate(ff_ind_category = paste0("ff_", num)) %>%
    select(ff_ind_category, everything())

ff_data
```

```r
}


# Get Fama-French 48-industry data
ff_data <-
    get_ff_ind(48) %>%
    # Define that the sic_min and sic_max correspond to each specific row.
    rowwise() %>%
    # Add a column 'sich', which is a list from sic_min to  sic_max
    mutate(sich = list(seq(from = sic_min, to = sic_max))) %>%
    # Make each element of the list its own row.
    unnest(sich) %>%
    rename(ff48 = ff_ind) %>%
    select(ff48, sich)



# Change code to replicate Fama-French 49 industry data

get_ff_ind2 <- function(num = 49) {
    t <- tempfile(fileext = ".zip")

    url <- paste0("https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/Siccodes", num, ".zip")

    download.file(url, t)

    ff_data <-
        readr::read_fwf(unzip(t),
                         col_positions = readr::fwf_widths(c(3, 7, NA),
                                                           c("ff_ind", "ff_ind_short_desc", "sic_range")),
                        col_types = "icc") %>%
        mutate(ff_ind_desc = if_else(!is.na(ff_ind_short_desc), sic_range, NA_character_)) %>%
        tidyr::fill(ff_ind, ff_ind_short_desc, ff_ind_desc) %>%
        filter(grepl("^[0-9]", sic_range)) %>%
        tidyr::extract(sic_range,
                        into = c("sic_min", "sic_max", "sic_desc"),
                        regex = "^([0-9]+)-([0-9]+)(.*)$") %>%
        mutate(sic_desc = trimws(sic_desc)) %>%
```

```
        #mutate(ff_ind_category = paste0("ff_", num)) %>%
        #select(ff_ind_category, everything())
        select(everything())

    ff_data
}


# Get Fama-French 49-industry data
ff_data49 <-
    get_ff_ind2(49) %>%
    rowwise() %>%
    mutate(sich = list(seq(from = sic_min, to = sic_max))) %>%
    unnest(sich) %>%
    rename(ff49 = ff_ind) %>%
    select(ff49, sich)
```

- Set `num <- 48` and work through the body of the `get_ff_ind` function line by line? Do you understand what the code is doing?

  A: See the notes that I put above each line of code.

- What are these lines doing?

  A: These linse set a list from sic_min to sic_max for each row, and then make each element of the list its own row, producing multiple rows, the number of which equals to the number of elements of the list.

```
    mutate(sich = list(seq(from = sic_min, to = sic_max))) %>%
        unnest(sich) %>%
```

Compustat data

We can get the data we need from the WRDS PostgreSQL database.

```
pg <- dbConnect(RPostgres::Postgres())

comp.funda <- tbl(pg, sql("SELECT * FROM comp.funda"))

compustat_annual <-
```

```r
    comp.funda %>%
    filter(indfmt == 'INDL', datafmt == 'STD', popsrc == 'D', consol == 'C',
           between(fyear, 1999, 2012)) %>%
    select(gvkey, fyear, datadate, fyr, sich, dltt, dlc, seq, oibdp,
            ib, ibc, oancf, xidoc, at, ppegt, sale, rect, ceq, csho, prcc_f) %>%
    mutate(fyear = as.integer(fyear)) %>%
    collect()
```

```r
controls_a <-       # 78301 observations
    compustat_annual %>%
    # Delete Financial Institutions 6000-6499, Insurance and Real Estate 6500-6999, Utilities 4900-4999.
    # I think the operator '|' should be '&'. Otherwise, we can find 6000 - 6999 and 4900 - 4949 after filtering. I change the code and ge
    # filter(!between(sich, 6000, 6999) | !between(sich, 4900, 4949)) %>%
    filter(!between(sich, 6000, 6999) & !between(sich, 4900, 4949)) %>%
    group_by(gvkey) %>%
    arrange(fyear) %>%
    mutate(lag_fyear = lag(fyear),
           mtob = if_else(lag(ceq) != 0,
                          lag(csho) * lag(prcc_f)/lag(ceq), NA_real_),
           # common shares outstanding * price close / common equity total
           leverage = if_else(dltt + dlc + seq != 0,
                              (dltt + dlc) / (dltt + dlc + seq) * 100, NA_real_),
           roa = if_else(lag(at) > 0, oibdp/lag(at), NA_real_)) %>%
    # filter(fyear == lag(fyear) + 1) %>%
    select(gvkey, datadate, fyear, at, mtob, leverage, roa, sich)

controls_a_all<-       # 92016 observations
    compustat_annual %>%
    filter(!between(sich, 6000, 6999) & !between(sich, 4900, 4949)) %>%
    group_by(gvkey) %>%
    arrange(fyear) %>%
    mutate(lag_fyear = lag(fyear),
           mtob = if_else(lag(ceq) != 0,
                          lag(csho) * lag(prcc_f)/lag(ceq), NA_real_),
           leverage = if_else(dltt + dlc + seq != 0,
                              (dltt + dlc) / (dltt + dlc + seq) * 100, NA_real_),
           roa = if_else(lag(at) > 0, oibdp/lag(at), NA_real_)) %>%
    #filter(fyear == lag(fyear) + 1) %>%
```

```r
    select(gvkey, datadate, fyear, at, mtob, leverage, roa, sich)


controls_a_dele<-    # 228 observations. 228 is not the difference between 92016 and 7830. In the case of 'gveky' ' 001618', year 2000 is
    compustat_annual %>%
    filter(!between(sich, 6000, 6999) & !between(sich, 4900, 4949)) %>%
    group_by(gvkey) %>%
    arrange(fyear) %>%
    mutate(lag_fyear = lag(fyear),
           mtob = if_else(lag(ceq) != 0,
                          lag(csho) * lag(prcc_f)/lag(ceq), NA_real_),
           leverage = if_else(dltt + dlc + seq != 0,
                              (dltt + dlc) / (dltt + dlc + seq) * 100, NA_real_),
           roa = if_else(lag(at) > 0, oibdp/lag(at), NA_real_)) %>%
    filter(fyear != lag(fyear) + 1) %>%
    select(gvkey, datadate, fyear, at, mtob, leverage, roa, sich)

# Check missing sich

controls_a_all_na<-        # I find that 'NA' is between(sich, 6000, 6999), so when we filter filter(!between(sich, 6000, 6999)), 'NA' is
    compustat_annual %>%
    filter(between(sich, 6000, 6999)) %>%
    group_by(gvkey) %>%
    arrange(fyear) %>%
    mutate(lag_fyear = lag(fyear),
           mtob = if_else(lag(ceq) != 0,
                          lag(csho) * lag(prcc_f)/lag(ceq), NA_real_),
           leverage = if_else(dltt + dlc + seq != 0,
                              (dltt + dlc) / (dltt + dlc + seq) * 100, NA_real_),
           roa = if_else(lag(at) > 0, oibdp/lag(at), NA_real_)) %>%
    #filter(fyear == lag(fyear) + 1) %>%
    select(gvkey, datadate, fyear, at, mtob, leverage, roa, sich)

#controls_b <-
#    controls_a %>%
#    group_by(gvkey) %>%
#    arrange(fyear) %>%
#    fill(at, mtob, leverage, roa) %>%
```

```
#   ungroup()


#controls_fyear <-
#    controls_b %>%
#    group_by(fyear) %>%
#    summarize_at(vars(at, mtob, leverage, roa), ~ mean(., na.rm=TRUE))

#df_controls <-     # 92016 observations
#  controls_b %>%
#  inner_join(controls_fyear, by = "fyear", suffix=c("", "_avg")) %>%
#  mutate(at = coalesce(at, at_avg),
#         mtob = coalesce(mtob, mtob_avg),
#         leverage = coalesce(leverage, leverage_avg),
#         roa = coalesce(roa, roa_avg)) %>%
#  select(gvkey, fyear, at, mtob, leverage, roa)

# Delete the missing values
df_controls <- # 69448 observations
  controls_a %>%
  filter(!is.na(at) & !is.na(mtob) & !is.na(leverage) & !is.na(roa))
  #select(gvkey, datadate, fyear, at, mtob, leverage, roa, sich)
```

- Why is `filter(fyear == lag(fyear) + 1)` required?

  A: Because when we generate mtob and roa, we use lag 1 data, and then for the first year of each firm 'gvkey', mtob and roa are missing. FHK want to delete the first year missing values and then fill missing values of later years.

  I think there is an issue in this step. Some firms have different sich in different years. After deleting sich 6000 - 6999 and 4900 - 4949, these firms do not have continuous years observations. For example, sich of 'gveky' ' 001618' in 1999, 2006, and 2007 is '6552', and sich in 2000 - 2005, 2008 - 2012 is 7389. After '6552' in 1999, 2006, and 2007 is deleted, year 2000 and 2008 do not have lag 1 year, so FHK delete 2000 and 2008. However, it is unnecessary to delete 2008 because FHK can fill 2008 with their methods.

  The unnecessary deletion also happens when sich 'NA' is deleted.

  I think FHK can do so by deleting the first row of each gvkey range by year.

  A related question is when sich or NA is deleted, and we use lag ib/at later, we may use not lag 1, but lag 2, 3, 4 or more.

  I solve the unnecessary deletion issue by omit the line " # filter(fyear == lag(fyear) + 1) %>%" and then 'filter(!is.na(at) & !is.na(mtob) & !is.na(leverage) & !is.na(roa))' when producing 'df_controls'.

- What are the authors doing in the creation of `controls_b`? (Hint: The key "verb" is `fill`.) Does this seem appropriate?

A: `controls_b` fill the missing values of ‘at, mtob, leverage, roa’ with the values in previous years, which have values (e.g., gvkey 129441 roa). This way is inappropriate because it implicitly assumes that ROA does not change every year, or follows a random walk, but this assumption is lack of evidence. FHK should check if they delete missing values, whether the resulsts will change.

- What are the authors doing in the creation of `df_controls` from `controls_fyear`? Does this seem appropriate?

  A: The authors fill the missing values with industry mean in the same year. For example, gvkey 001010 mtob is missing in 2000 to 2003, and is filled with industry average. After checking the original data, I find that prcc_f, i.e. price close, is missing in the four years. FHK should analyze the reason (e.g., M&A, bankruptcy, late reporting etc.), and then decide whether it is appropriate to use the industry average. If this company is in a special situation, it is inappropriate to do so.

- How would you change the code to skip the two steps above? Does doing so make a difference?

  A: If we skip the two steps, df_controls will be ‘filter(!is.na(at) & !is.na(mtob) & !is.na(leverage) & !is.na(roa))’ from ‘controls_a’. I find that doing so makes the results more significant in ‘CL_2=TRUE’., and the magnitudes of interaction coefficients bigger in ‘CL_2=FALSE’.

```
ind_data <-
    compustat_annual %>%
    select(gvkey, fyear, sich) %>%
    inner_join(ff_data, by="sich") %>%
    # Do not choose sich
    select(-sich)

# Check why some observations are deleted after sich is matched. 2271 observations have sich, but cannot match ff48.
ind_data_left <-
    compustat_annual %>%
    select(gvkey, fyear, sich) %>%
    left_join(ff_data, by="sich") %>%
    filter(is.na(ff48))


for_disc_accruals_a <-
    compustat_annual %>%
    filter(!between(sich, 6000, 6999), !between(sich, 4900, 4949)) %>%
    inner_join(ind_data, by = c("gvkey", "fyear")) %>%
    select(gvkey, fyear, fyr, ib, ibc, oancf, xidoc, at, ppegt, sale,
           rect, ceq, csho, prcc_f, ff48)

for_disc_accruals_b <-
    for_disc_accruals_a %>%
    group_by(gvkey, fyr) %>%
```

```
    arrange(fyear) %>%
    filter(lag(at) > 0) %>%
    mutate(lag_fyear = lag(fyear),
           # Nominator: income before extraordinary items - (operationg activities NCF - Extraordinary Items and Discontinued Operations (
           ta_at = (ibc - (oancf - xidoc)) / lag(at),
           one_at = 1/lag(at),
           ppe_at = ppegt / lag(at),
           sale_c_at = (sale - lag(sale)) / lag(at),
           salerect_c_at = ((sale - lag(sale)) - (rect - lag(rect))) / lag(at),
           bm_lr = if_else(csho * prcc_f > 0, ceq / (csho * prcc_f), NA_real_),
           mb_lr = if_else(ceq != 0, csho * prcc_f / ceq, NA_real_)) %>%
    ungroup() %>%
    filter(lag_fyear == fyear - 1,
           abs(ta_at) <= 1,  # Is this criterion implicitly assumed?
           !is.na(salerect_c_at), !is.na(ta_at), !is.na(ppe_at)) %>%
    select(ff48, gvkey, fyear, ta_at, one_at, ppe_at,
           sale_c_at, salerect_c_at, bm_lr, mb_lr)

ind_years <-
    for_disc_accruals_b %>%
    group_by(ff48, fyear) %>%
      summarize(num_obs = n(), .groups="drop") %>%
    filter(num_obs >= 10)

for_disc_accruals <-
    for_disc_accruals_b %>%
    semi_join(ind_years, by = c("ff48", "fyear")) %>%
    arrange(ff48, fyear, gvkey)
```

Discretionary accruals

The following code estimates the discretionary-accrual models.

```
fm_da1 <-
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    do(model = tidy(lm(ta_at ~ one_at + sale_c_at + ppe_at - 1, data = .))) %>%
```

```r
  # - 1 means omitting intercept. The Jones Model also does not include intercept, but FHK Model 3 and 4 on Page 1263 include an intercept
    unnest(model) %>%
    select(ff48, fyear, term, estimate) %>%
    pivot_wider(names_from = "term", values_from = "estimate",
                names_prefix = "b_")

# Check whether not omitting intercept makes a difference
fm_da1_noomit <-
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    # do(model = tidy(lm(ta_at ~ one_at + sale_c_at + ppe_at - 1, data = .))) %>%
    do(model = tidy(lm(ta_at ~ one_at + sale_c_at + ppe_at, data = .))) %>%
    unnest(model) %>%
    select(ff48, fyear, term, estimate) %>%
    pivot_wider(names_from = "term", values_from = "estimate",
                names_prefix = "b_")

df_da1 <-
    for_disc_accruals %>%
    left_join(fm_da1, by = c("ff48", "fyear")) %>%
    mutate(hat = one_at * b_one_at + ppe_at * b_ppe_at + salerect_c_at * b_sale_c_at,
           da1 = ta_at - hat) %>%
    select(gvkey, fyear, da1)

# Check whether not omitting intercept makes a difference
df_da1_noomit <-
    for_disc_accruals %>%
    left_join(fm_da1_noomit, by = c("ff48", "fyear")) %>%
    mutate(hat = one_at * b_one_at + ppe_at * b_ppe_at + salerect_c_at * b_sale_c_at,
           da1 = ta_at - hat) %>%
    select(gvkey, fyear, da1)
# I find that df_da1 changes when we do not omit intercept, but the regression results do not change.


df_da2 <-
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    do(model = augment(lm(ta_at ~ one_at + salerect_c_at + ppe_at - 1, data = .),
```

```
                       data = select(., -ff48, -fyear))) %>%
    unnest(model) %>%
    select(gvkey, fyear, .resid) %>%
    dplyr::rename(da2 = .resid)


fm_da3 <-
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    do(model = tidy(lm(ta_at ~ one_at + sale_c_at + ppe_at, data = .))) %>%
    unnest(model) %>%
    select(ff48, fyear, term, estimate) %>%
    pivot_wider(names_from = "term", values_from = "estimate",
                names_prefix = "b_")

df_da3 <- for_disc_accruals %>%
    left_join(fm_da3, by = c("ff48", "fyear")) %>%
    mutate(hat = `b_(Intercept)` + one_at * b_one_at + ppe_at * b_ppe_at + salerect_c_at * b_sale_c_at,
           da3 = ta_at - hat) %>%
    select(gvkey, fyear, da3)

df_da4 <-
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    do(model = augment(lm(ta_at ~ one_at + salerect_c_at + ppe_at, data = .),
                       data = select(., -ff48, -fyear))) %>%
    unnest(model) %>%
    select(gvkey, fyear, .resid) %>%
    dplyr::rename(da4 = .resid)

fm_da5 <-
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    do(model = tidy(lm(ta_at ~ one_at + sale_c_at + ppe_at + bm_lr, data = .))) %>%
    unnest(model) %>%
    select(ff48, fyear, term, estimate) %>%
    pivot_wider(names_from = "term", values_from = "estimate",
                names_prefix = "b_")
```

```
df_da5 <-
    for_disc_accruals %>%
    left_join(fm_da5, by = c("ff48", "fyear")) %>%
    mutate(hat = `b_(Intercept)` + one_at * b_one_at + ppe_at * b_ppe_at + salerect_c_at * b_sale_c_at +
               bm_lr * b_bm_lr,
           da_lr1 = ta_at - hat) %>%
    select(gvkey, fyear, da_lr1) %>%
    arrange(gvkey, fyear)

df_da6 <-
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    do(model = augment(lm(ta_at ~ one_at + salerect_c_at + ppe_at + bm_lr,
                          data = ., na.action = na.exclude),
                          data = select(., -ff48, -fyear))) %>%
    unnest(model) %>%
    select(gvkey, fyear, .resid) %>%
    dplyr::rename(da_lr2 = .resid)

fm_da7 <-
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    do(model = tidy(lm(ta_at ~ one_at + sale_c_at + ppe_at + mb_lr, data = .))) %>%
    unnest(model) %>%
    select(ff48, fyear, term, estimate) %>%
    pivot_wider(names_from = "term", values_from = "estimate",
                names_prefix = "b_")

df_da7 <-
    for_disc_accruals %>%
    left_join(fm_da7, by = c("ff48", "fyear")) %>%
    mutate(hat = `b_(Intercept)` + one_at * b_one_at + ppe_at * b_ppe_at + salerect_c_at * b_sale_c_at +
               mb_lr * b_mb_lr,
           da_lr3 = ta_at - hat) %>%
    select(gvkey, fyear, da_lr3) %>%
    arrange(gvkey, fyear)

df_da8 <-
```

```r
    for_disc_accruals %>%
    group_by(ff48, fyear) %>%
    do(model = augment(lm(ta_at ~ one_at + salerect_c_at + ppe_at + mb_lr,
                          data = ., na.action = na.exclude),
                          data = select(., -ff48, -fyear))) %>%
    unnest(model) %>%
    select(gvkey, fyear, .resid) %>%
    dplyr::rename(da_lr4 = .resid)

merged <-
    for_disc_accruals %>%
    left_join(df_da1, by=c("gvkey", "fyear")) %>%
    left_join(df_da2, by=c("gvkey", "fyear")) %>%
    left_join(df_da3, by=c("gvkey", "fyear")) %>%
    left_join(df_da4, by=c("gvkey", "fyear")) %>%
    left_join(df_da5, by=c("gvkey", "fyear")) %>%
    left_join(df_da6, by=c("gvkey", "fyear")) %>%
    left_join(df_da7, by=c("gvkey", "fyear")) %>%
    left_join(df_da8, by=c("gvkey", "fyear")) %>%
    arrange(gvkey, fyear)
```

- Why does the code look different for the odd-numbered datasets (e.g., `df_da1`) and the even-numbered datasets (e.g., `df_da2`).

  A: df_da1 is obtained by ta_at - hat, and hat is the fitted value of normal accruals, but FHK use `salerect_c_at * b_sale_c_at` to get the fitted value.

  df_da2 is the residual in the regression of ta_at on salerect_c_at and other same variables.

- Does the argument for using `salerect_c_at * b_sale_c_at` make sense to you?

  A: No, it is fine to use the modified Jones Model in Dechow, Sloan, and Sweeney (1995), but FHK should keep the regression model and the fitted model consistent.

Performance-matching code

The performance-matching code here is not to generate matched pilot firms and controlled firms, but to adjust discretionary accruals by matching each sample firm with the firm from the same fiscal year-industry that has the closest ib_at (FHK).

```r
perf <-
    merged %>%
    select(gvkey, fyear, ff48) %>%
    inner_join(
        compustat_annual %>%
            mutate(fyear = fyear + 1), by = c("gvkey", "fyear")) %>%
    # Income Before Extraordinary Items / total assets
    mutate(ib_at = if_else(at > 0, ib/at, NA_real_)) %>%
    select(gvkey, fyear, ff48, ib_at)

# Check what 'fyear = fyear + 1' is doing by omitting this line
perf2 <-
    merged %>%
    select(gvkey, fyear, ff48) %>%
    inner_join(
        compustat_annual,
          by = c("gvkey", "fyear")) %>%
    mutate(ib_at = if_else(at > 0, ib/at, NA_real_)) %>%
    select(gvkey, fyear, ff48, ib_at)

# Match two datasets and then use lag
perf3 <-
    merged %>%
    select(gvkey, fyear, ff48) %>%
    inner_join(
        compustat_annual,
          by = c("gvkey", "fyear")) %>%
    mutate(ib_at = if_else(at > 0, ib/at, NA_real_)) %>%
    mutate(ib_at_lag = lag(ib_at)) %>%
    select(gvkey, fyear, ff48, ib_at, ib_at_lag)

# Use lag in 'compustat_annual' and then match

perf4 <-
    merged %>%
    select(gvkey, fyear, ff48) %>%
    inner_join(
        compustat_annual %>%
```

```
        group_by(gvkey) %>%
        arrange(fyear) %>%
        mutate(ib_at = if_else(at > 0, ib/at, NA_real_)) %>%
        mutate(ib_at = lag(ib_at)), by = c("gvkey", "fyear")) %>%
    # Income Before Extraordinary Items / total assets
    select(gvkey, fyear, ff48, ib_at)


perf_match <-
    perf %>%
    inner_join(perf, by = c("fyear", "ff48"),
               suffix = c("", "_other")) %>%
    filter(gvkey != gvkey_other) %>%
    mutate(perf_diff = abs(ib_at - ib_at_other)) %>%
    group_by(gvkey, fyear) %>%
    filter(perf_diff == min(perf_diff)) %>%
    select(gvkey, fyear, ff48, gvkey_other, perf_diff)

# In Session 9, we match samples by predicting the probability of treatment. That method is different from how FHK match.
#fm_match <- matchit(treat ~ mean_x,
  #                  method = "nearest", data = fm_psm$data,
   #                 caliper = 0.02)


perf_matched_accruals <-
    merged %>%
    rename(gvkey_other = gvkey) %>%
    select(gvkey_other, fyear, matches("^da"))

pm_disc_accruals <-
    merged %>%
   # I think it is unnecessary to match by 'ff48' because ('gvkey' and 'fyear') have defined a unique value. I use by = c("gvkey", "fyear"
    inner_join(perf_match, by = c("ff48", "gvkey", "fyear")) %>%
    inner_join(perf_matched_accruals, by = c("fyear", "gvkey_other"),
               suffix = c("", "_other")) %>%
    mutate(da1_adj = da1 - da1_other,
           da2_adj = da2 - da2_other,
           da3_adj = da3 - da3_other,
```

```r
          da4_adj = da4 - da4_other,
          da_lr1_adj = da_lr1 - da_lr1_other,
          da_lr2_adj = da_lr2 - da_lr2_other,
          da_lr3_adj = da_lr3 - da_lr3_other,
          da_lr4_adj = da_lr4 - da_lr4_other) %>%
    select(gvkey, fyear, matches("_adj"))


# Check whether a firm is used as a control just once
pm_disc_accruals2 <-
    merged %>%
    inner_join(perf_match, by = c("ff48", "gvkey", "fyear")) %>%
    inner_join(perf_matched_accruals, by = c("fyear", "gvkey_other"),
               suffix = c("", "_other")) %>%
    mutate(da1_adj = da1 - da1_other,
          da2_adj = da2 - da2_other,
          da3_adj = da3 - da3_other,
          da4_adj = da4 - da4_other,
          da_lr1_adj = da_lr1 - da_lr1_other,
          da_lr2_adj = da_lr2 - da_lr2_other,
          da_lr3_adj = da_lr3 - da_lr3_other,
          da_lr4_adj = da_lr4 - da_lr4_other) %>%
    select(everything())

pm_disc_accruals2 %>%
  count(gvkey_other) %>%
  arrange(desc(n)) %>%
  filter(n >= 2)
```

```
## # A tibble: 7,857 x 2
##    gvkey_other     n
##    <chr>       <int>
##  1 010846         25
##  2 010853         25
##  3 019565         22
##  4 012384         21
##  5 024475         20
##  6 064166         20
```

```
##  7 005210         19
##  8 007346         19
##  9 008169         19
## 10 014794         19
## # ... with 7,847 more rows
```

- What does the line `mutate(fyear = fyear + 1)` effectively do? Would it be possible to create `perf` using an alternative approach? (Hint: Use `lag`?) Does this give the same result?

  A: The line `mutate(fyear = fyear + 1)` produces ib_at, which is ib / at with the value in the previous year (lag 1).

  It is possible to use 'lag'. If we match 'merged' and 'compustat_annual', calculate ib_at, and then use lag, we will cause missing values because the first year of a firm does not have its lag (as 'perf 3' shows). However, if we calculat ib_at, use lag, and then match 'merged' and 'compustat_annual', the reuslts are the same (as 'perf4' shows).

- Does the code above ensure that a performance-matched control firm is used as a control just once? If so, which aspect of the code ensures this is true? If not, how might you ensure this? (Just describe the approach in general; no need to do this.)

  A: No, I find that 7,857 firms are used as a control more than once.

  When generating 'perf_match', we can use the code of 'perf-match', and then if we find one firm is used as a control more than once, we can choose the smallest perf_diff. For example, '001004' is used as a control for 13 firms, but we can choose the smallest perf_diff among the 13 pairs, so that '001004' is only used as a control once. Then, the residual 12 firms are matched again with all other firms except for '001004'. We can iterate this process until all firms are matched and a firm is used as a control just once.

  'perf' has 59,305 observations, but why 'perf_match' has 59,363 observations? 'merged' also has 59,305 observations. Why after inner_join 'perf_match' and 'merged', 'pm_disc_accruals' has 59,363 observations, more than 59,305?

  In the Chunk below, 'pm_disc_accruals_sorted' 'filter(row_number() == 1)' deletes duplicates of (gvkey, fyear). I find that 'perf_match' chooses more than one controls for a firm. For example, 'gvkey' '008838' in fyear 2004 has three matched 'gvkey_other' '009267', '009655', and '012384', and perf_diff are all 0.0013764522. '009267', '009655', and '012384' indeed have the same ib_at (ib/at) in 2003, the lag of 2004.

gvkey	ib	at	ib/at
009267	7344.6	100864.6	0.07281642915353850000
009655	4896.4	67236.4	0.07282364909483550000
012384	12241	168091	0.07282364909483550000

Then, the match process may be more complicated because when we ensure that a firm is only used as a control once, we should not waste a firm as a control when we delete the duplicates of match with 'filter(row_number() == 1)'.

Data preparation

Merge data sets

```
pm_disc_accruals_sorted <-
    pm_disc_accruals %>%
    group_by(gvkey, fyear) %>%
    filter(row_number() == 1) %>%
    ungroup()
# After 'filter(row_number() == 1)', 'pm_disc_accruals_sorted' has 59,305 observations.


# Check which firms have more than 1 rows
pm_disc_accruals_sorted2 <-
    pm_disc_accruals %>%
    group_by(gvkey, fyear) %>%
    mutate(rownum = row_number()) %>%
    ungroup()


sho_accruals_prewin <-   # 21,880 observations
  sho_firm_years %>%     # 21,880 observations
  #left_join(df_controls,                          # Why left_join, not inner_join?
  #          by = c("gvkey", "fyear")) %>%
  # Because 'sho_firm_years' and 'df_controls' both have "datadate", I need to include "datadate" in match criteria. Otherwise, 'sho_accru
  left_join(df_controls,
            by = c("gvkey", "fyear", "datadate")) %>%
  left_join(pm_disc_accruals_sorted,
            by = c("gvkey", "fyear"))
```

Winsorize the data

```
winsorize <- function(x, prob = 0.01, p_low = prob, p_high = 1-prob) {
    cuts <- quantile(x, probs = c(p_low, p_high),
                     type = 2, na.rm = TRUE)
    x[x < cuts[1]] <- cuts[1]
```

```
    x[x > cuts[2]] <- cuts[2]
    x
}

win_vars <- c("at", "mtob", "leverage", "roa", "da1_adj", "da2_adj",
                 "da3_adj", "da4_adj", "da_lr1_adj", "da_lr2_adj",
                 "da_lr3_adj", "da_lr4_adj")

# Winsorize "within each year" at the 1%/99% level
sho_accruals <-              # 21,880 observations
    sho_accruals_prewin %>%  # 21,880 observations
    group_by(fyear) %>%
    mutate_at(all_of(win_vars), winsorize, prob=0.01) %>%
    ungroup()

# Check how 'sho_accruals_prewin' changes after winsorization.

# summary(sho_accruals_prewin),the output is too long, I treat summary as a comment here.

# summary(sho_accruals)

# Winsorize "across sample years" at the 1%/99% level
sho_accruals_sam <-              # 21,880 observations
    sho_accruals_prewin %>%  # 21,880 observations
    #group_by(fyear) %>%
    mutate_at(all_of(win_vars), winsorize, prob=0.01) %>%
    ungroup()

# summary(sho_accruals_sam)

# Winsorize "within each year" at the 2%/98% level
sho_accruals_2per <-              # 21,880 observations
    sho_accruals_prewin %>%  # 21,880 observations
    group_by(fyear) %>%
    mutate_at(all_of(win_vars), winsorize, prob=0.02) %>%
    ungroup()
# summary(sho_accruals_2per)
```

```r
# Winsorize "across sample years" at the 2%/98% level
sho_accruals_sam_2per <-              # 21,880 observations
    sho_accruals_prewin %>%  # 21,880 observations
    #group_by(fyear) %>%
    mutate_at(all_of(win_vars), winsorize, prob=0.02) %>%
    ungroup()
# summary(sho_accruals_sam_2per)
```

- In an online appendix BDLYY say "FHK winsorize covariates for their covariate balance table at 1/99%. We inferred that they also winsorized accruals at this level. Whether they winsorize across sample years or within each year, they do not specify." The code above winsorized within each year. How would you modify the code to winsorize "across sample years"? Does it make a difference?

  A: If we delete the line 'group_by(fyear) %>%', the code will winsorize "across sample years". I find that the min after winsorizing "across sample years" is greater than after winsoring within each year, and the max of the former is smaller than the latter. That is, winsorizing "across sample years" makes the sample more centralized than winsoring within each year, i.e. the range is smaller. The reason may be that some years have values that are more extreme. The coefficients of the interaction term are more significant and with greater magnitude.

- How would you modify the code to winsorize at the 2%/98% level? Does this make a difference?

  A: Define prob = 0.02 in the line 'mutate_at(all_of(win_vars), winsorize, prob=0.02)'. Winsorizing at the 2%/98% level makes the sample more centralized. The magnitude of the interaction term coefficients are smaller than that in winsorizing at 1%/99% level.

- How would you modify the code to not winsorize at all? Does this make a difference?

  A: The code will be ' sho_accruals <-
  sho_accruals_prewin '.

  No winsorization makes the magnitudes of the interaction coefficients greater.

```r
reg_data <-                           # 21880 observations
    sho_accruals %>%
    mutate(leverage = leverage/100,
           year = year(datadate),
           during = year %in% c(2005, 2006, 2007),
           post = year %in% c(2008, 2009, 2010),
           log_at = log(at),
           pilot = sho,
           pilot_during = pilot * during,
           pilot_post = pilot * post) %>%
    select(gvkey, fyear, log_at, leverage, mtob, roa,
           da1_adj, da2_adj, da3_adj, da4_adj,
```

```
         da_lr1_adj, da_lr2_adj, da_lr3_adj, da_lr4_adj,
         year, during, post, pilot, pilot_during, pilot_post)

# Exclude 2004 data from the sample
reg_data_exc <-                          # 19791 observations
    sho_accruals %>%
    mutate(leverage = leverage/100,
           year = year(datadate),
           during = year %in% c(2005, 2006, 2007),
           post = year %in% c(2008, 2009, 2010),
           log_at = log(at),
           pilot = sho,
           pilot_during = pilot * during,
           pilot_post = pilot * post) %>%
    filter (year != 2004) %>%
    select(gvkey, fyear, log_at, leverage, mtob, roa,
           da1_adj, da2_adj, da3_adj, da4_adj,
           da_lr1_adj, da_lr2_adj, da_lr3_adj, da_lr4_adj,
           year, during, post, pilot, pilot_during, pilot_post)
```

- Some of the studies discussed by BDLYY exclude 2004 data from the sample. How would you modify the above code to do this here?

  A: Add a line of code 'filter (year != 2004) %>%', as shown in 'reg_data_exc' above, but if we want to exclude 2004, we may need to exclude 2004 before winsorization because winsorization usually happens at the last step?

- Would excluding 2004 here make a significant difference?

  A: The coefficients of pilot_during are less significant and with smaller magnitudes.

Regression analysis

```
lhs <- c("da1_adj", "da2_adj", "da3_adj", "da4_adj",
         "da_lr1_adj", "da_lr2_adj", "da_lr3_adj", "da_lr4_adj")
controls <- c("log_at", "mtob", "roa", "leverage")

reg_year_fe <- function(y, firm_fe = FALSE, cl_2 = TRUE) {
    model <- paste0(y, " ~ pilot_during + pilot_post + pilot + ",
                    # I find that if we use " ~ pilot_during + pilot_post + pilot + during + post + " by strictly following FHK's model, t
```

```r
                    paste(controls, collapse = " + "),
                    if_else(firm_fe, "| gvkey + year ", "| year "),
                  "| 0 ",
                  if_else(!cl_2, "| gvkey ", "| year + gvkey"))
    fm <- felm(as.formula(model), data = reg_data)
    fm
}

# Do not define the interaction term, but use multiply operator directly

# Three multiply
reg_year_fe_al1 <- function(y, firm_fe = FALSE, cl_2 = TRUE) {
    model <- paste0(y, " ~ pilot * during * post + ",
                    paste(controls, collapse = " + "),
                    if_else(firm_fe, "| gvkey + year ", "| year "),
                  "| 0 ",
                  if_else(!cl_2, "| gvkey ", "| year + gvkey"))
    fm_al <- felm(as.formula(model), data = reg_data)
    fm_al
}

# Two multiply
reg_year_fe_al2 <- function(y, firm_fe = FALSE, cl_2 = TRUE) {
    model <- paste0(y, " ~ pilot * during + pilot * post + ",
                    paste(controls, collapse = " + "),
                    if_else(firm_fe, "| gvkey + year ", "| year "),
                  "| 0 ",
                  if_else(!cl_2, "| gvkey ", "| year + gvkey"))
    fm_al <- felm(as.formula(model), data = reg_data)
    fm_al
}
```

- Code in the previous subsection creates variables `pilot_during` and `pilot_post`. Is it necessary to do so to estimate the regressions here? If not, how would you modify the string `" ~ pilot_during + pilot_post + pilot + "`,in the code above to not use these variables?

  A: Compared to Model (5) on FHK Page 1268, 'reg_year_fe' does not have 'during' and 'post'. The regressions run by FHK have year fixed effects, so POST and DURING will fall out.

  There are two ways of modifying the string. We can use either '" ~ pilot * during * post + " or " ~ pilot * during + pilot * post + ". Of course, the latter is more consistent with FHK.

Table 4: Change filtering code in producing 'controls(a)' with cl2

	Dependent variable:							
	model							
	da1_adj	da2_adj	da3_adj	da4_adj	da_lr1_adj	da_lr2_adj	da_lr3_adj	da_lr4_adj
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
pilot_during	-0.010^{***}	-0.010^{***}	-0.010^{***}	-0.010^{***}	-0.013^{***}	-0.013^{***}	-0.012^{***}	-0.012^{***}
	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)
pilot_post	0.008^{**}	0.008^{**}	0.008^{**}	0.008^{**}	0.006	0.007	0.009^{**}	0.009^{**}
	(0.003)	(0.003)	(0.003)	(0.003)	(0.004)	(0.004)	(0.003)	(0.003)
pilot	0.001	0.001	0.0001	-0.0002	0.002	0.001	0.001	0.001
	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)
Observations	19,274	19,274	19,274	19,274	18,456	18,456	18,454	18,454
R^2	0.002	0.003	0.003	0.003	0.002	0.002	0.002	0.003
Note:								*p<0.1; **p<0.05; ***p<0.01

Table 5: Three multiply with cl2'

	Dependent variable:							
	model							
	da1_adj	da2_adj	da3_adj	da4_adj	da_lr1_adj	da_lr2_adj	da_lr3_adj	da_lr4_adj
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
pilot	0.001	0.001	0.0001	−0.0002	0.002	0.001	0.001	0.001
	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)
during								
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
post								
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
pilot:during	−0.010***	−0.010***	−0.010***	−0.010***	−0.013***	−0.013***	−0.012***	−0.012***
	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)
pilot:post	0.008**	0.008**	0.008**	0.008**	0.006	0.007	0.009**	0.009**
	(0.003)	(0.003)	(0.003)	(0.003)	(0.004)	(0.004)	(0.003)	(0.003)
duringTRUE:post								
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
pilot:duringTRUE:post								
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
Observations	19,274	19,274	19,274	19,274	18,456	18,456	18,454	18,454
R^2	0.002	0.003	0.003	0.003	0.002	0.002	0.002	0.003
Note:								*p<0.1; **p<0.05; ***p<0.01

Table 6: Two multiply with cl2'

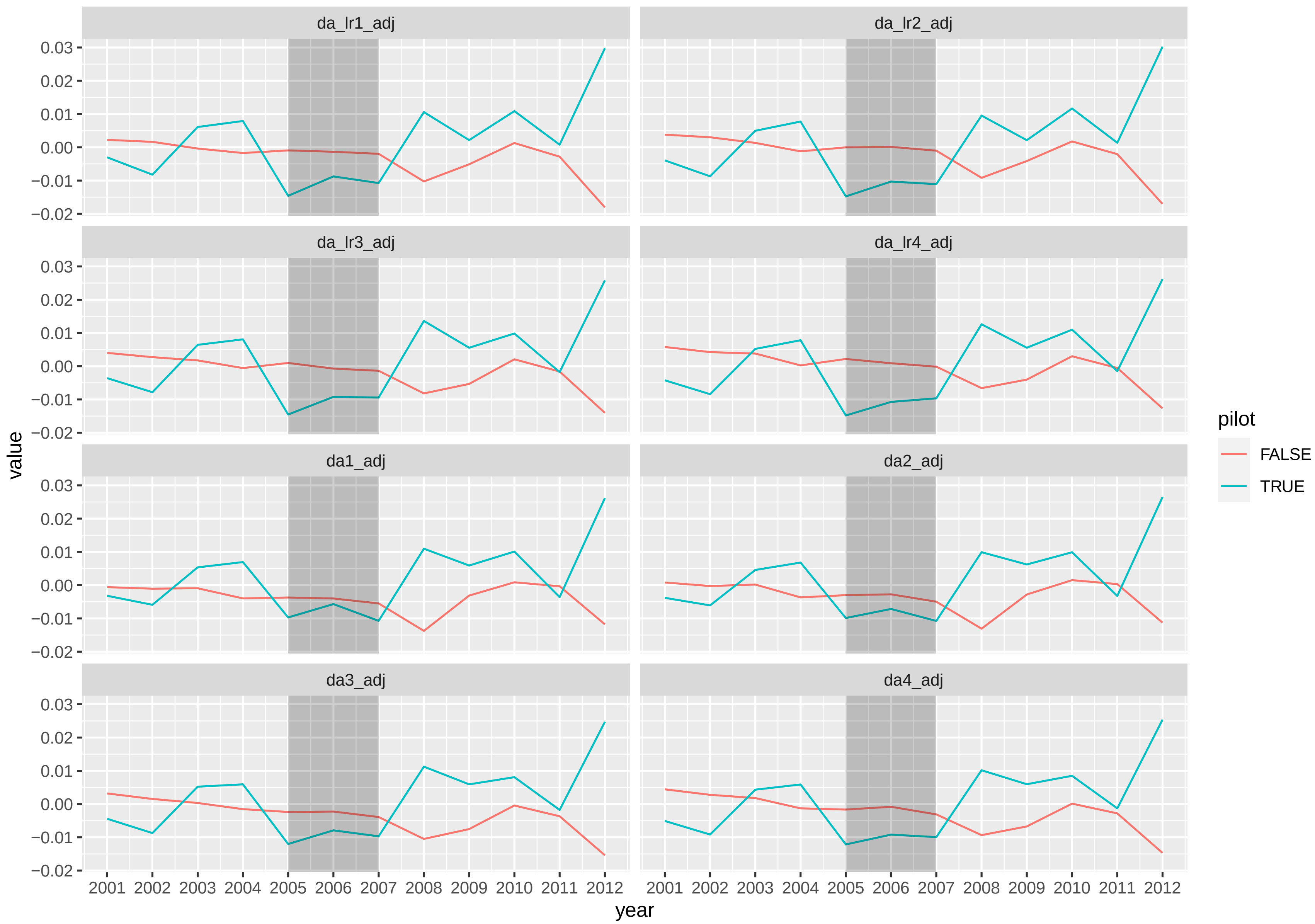
	Dependent variable:							
	model							
	da1__adj	da2__adj	da3__adj	da4__adj	da__lr1__adj	da__lr2__adj	da__lr3__adj	da__lr4__adj
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
pilot	0.001	0.001	0.0001	−0.0002	0.002	0.001	0.001	0.001
	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)
during								
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
post								
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
pilot:during	−0.010***	−0.010***	−0.010***	−0.010***	−0.013***	−0.013***	−0.012***	−0.012***
	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)	(0.003)
pilot:post	0.008**	0.008**	0.008**	0.008**	0.006	0.007	0.009**	0.009**
	(0.003)	(0.003)	(0.003)	(0.003)	(0.004)	(0.004)	(0.003)	(0.003)
Observations	19,274	19,274	19,274	19,274	18,456	18,456	18,454	18,454
R^2	0.002	0.003	0.003	0.003	0.002	0.002	0.002	0.003
Note:								*p<0.1; **p<0.05; ***p<0.01

Table 7: Change filtering code in producing 'controls(a)' without cl2

	Dependent variable:							
	model							
	da1_adj	da2_adj	da3_adj	da4_adj	da_lr1_adj	da_lr2_adj	da_lr3_adj	da_lr4_adj
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
pilot_during	-0.014^{***}	-0.014^{***}	-0.014^{***}	-0.014^{***}	-0.016^{***}	-0.016^{***}	-0.016^{***}	-0.016^{***}
	(0.005)	(0.005)	(0.005)	(0.005)	(0.005)	(0.005)	(0.005)	(0.005)
pilot_post	0.006	0.006	0.006	0.006	0.004	0.004	0.006	0.006
	(0.006)	(0.006)	(0.006)	(0.006)	(0.006)	(0.006)	(0.006)	(0.006)
Observations	19,274	19,274	19,274	19,274	18,456	18,456	18,454	18,454
R^2	0.176	0.176	0.171	0.171	0.175	0.175	0.176	0.176
Note:								*p<0.1; **p<0.05; ***p<0.01

Plot coefficients

- **Stretch exercise** Produce plots like those below, but using total accruals instead of discretionary accruals and excluding controls (so the coefficients will be simple conditional sample means).



da_lr1_adj
da_lr2_adj
da_lr3_adj
da_lr4_adj
da1_adj
da2_adj
da3_adj
da4_adj
value
year
0.03
0.02
0.01
0.00
−0.01
−0.02
2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012
pilot
FALSE
TRUE

```
results_by_year_total %>%
    ggplot(aes(x = year, y=value, color = pilot)) +
    geom_line() +
    scale_x_continuous(breaks=2000:2012L) +
    geom_rect(xmin=2005, xmax=2007, ymin=-Inf, ymax=Inf,
              color = NA, alpha=0.01) +
    facet_wrap(~ dv, ncol = 2)
```