Code for: Does Floor Trading Matter?

Jonathan Brogaard
University of Utah

Matthew C. Ringgenberg
University of Utah

Dominik Roesch
State University of New York at Buffalo

ABSTRACT

This document contains code to produce the results of "Does Floor Trading Matter?" that were obtained using data available on Wharton Research Data Services (WRDS), i.e., Tables 1, 2, 3, 7, and Figure 1. This document is an R Sweave file. To execute the code and produce a PDF, it is easiest to use a local version of RStudio, alternatively, one could try RStudio as provided by WRDS. On execution, the code downloads and saves locally all required data from WRDS. For that to work, you first need to configure access to WRDS, as described below. Then, you need to download "Linking Suite, Daily TAQ CRSP" manually. Note: after the article was accepted, WRDS made changes to some of the data which may lead to slight differences between the results produced by this file and the published results.

1 Discussion

This code produces the results that were calculated using data on WRDS (i.e., Tables 1, 2, 3, 7, and Figure 1). In particular, we do not include data that require TAQ XDP (parity volume and d-orders). We note that using the most recent data will lead to slight differences from the published paper, because of changes in the WRDS data that occurred after the paper was accepted. For example, in the CRSP Stock Header Information, PERMNO 10051 changed Industry from HSICCD 8093 to 3448. Moreover,

- Permno 91953 (a trust) disappeared
- Permno 12141 with share code 10 disappeared
- Permno 12355 changed share code from 11 to 14

We also noticed changes in Effective Spreads with the release of the second version of "Intraday Indicators by WRDS" in July 2021.

2 Setup

This section contains various code blocks to setup the analysis such as environment variables (like where to store the data); variables related to the analysis (such as when the Floor closed); and variables determining how the compiled PDF will look (e.g., whether it will contain the underlying R code.)

2.1 Setup variable names

- First, the code contains variables that determine how the PDF will look, e.g., with ("echo=T") or without the underlying R code ("echo=F".)

```
knitr::opts_knit$set(progress = TRUE, verbose = TRUE)
opts_knit$set(self.contained=FALSE)

# cache.lazy = F (helps error when caching large data sets)
opts_chunk$set(echo=T,warning=F,message=F,tidy=T,tidy.opts=list(width.cutoff=50),
```

- Second, the code sets important environment variables (which need to be adjusted by the user):

```
wrds_username <- "droschub"

setwd("C:/Users/drosch/Documents/repositories/floor_trading/R")

data_dir <- "C:/Users/drosch/Dropbox/Research/data/floor_trades/replicate"

if (!file.exists(data_dir)) {
    stop(sprintf("Could not find data_dir '%s' to store data, first create this d
        data_dir))
}
```

- Third, the code creates variables related to the analysis (which do not need to be adjusted by the user):

```
# Variables related to summary statistics
in_end_sample_long <- "2020-04-30"

# Variables related to propensity score matching
in_end_match <- "2020-03-22"
listing_cov <- c("CPrc", "total_dollar_m", "mktcap")

# Variables related to Floor closing analysis
in_close_date <- "2020-03-23"

in_start_close_sample <- "2020-03-15"
in_end_close_sample <- "2020-03-27"

# Variables related to Floor re-opening analysis
in_open1_date <- "2020-05-26"
in_open2_date <- "2020-06-17"
```

```r
in_start_open1_sample <- "2020-05-18"
in_end_open1_sample <- "2020-06-01"

in_start_open2_sample <- "2020-06-10"
in_end_open2_sample <- "2020-06-23"
```

2.2 Setup Data connection

If required data cannot be found, automatically download it from WRDS and industry specifications from French's website. For that first establish a connection to WRDS. Note, as explained at WRDS website[1] the file

- %APPDATA%/postgresql/pgpass.conf

must contain the username and password in the following way:

- wrds-pgdata.wharton.upenn.edu:9737:wrds:wrds_username:wrds_password

```r
# Setup see:
# https://wrds-www.wharton.upenn.edu/pages/support/programming-wrds/programming-r/r-
library(RPostgres)

wrds <- dbConnect(Postgres(), host = "wrds-pgdata.wharton.upenn.edu",
    port = 9737, dbname = "wrds", sslmode = "require",
    user = wrds_username)
```

2.3 Setup R packages and formats

[1] https://wrds-www.wharton.upenn.edu/pages/support/programming-wrds/programming-r/r-from-your-computer/

```r
library(plyr)
library(dplyr)
library(RODBC)
library(sqldf)
library(reshape2)
library(ggplot2)
# for  na.locf
library(xts)

# for R tidy in knitr
library(formatR)

library(MatchIt)
library(lubridate)
library(plm)
require(lmtest)

# for fwrite, fread
library( data.table)
library(anytime)
library(lubridate)



# Setup default number layouts
format_numeric    <- "%.2f";
format_integer    <- "%.0f";
format_percent    <- "%.2f";
format_significance <- "(%.2f)";

pretty_numbers <- function(format, number, p_value = -1, ...)
{
    result <- sprintf(format, number, ...)
```

```
    if (format == format_integer || format == format_numeric)
    {
        result <- prettyNum(result, big.mark=",",  scietific=F)
    }

    if (p_value > 0) {

        stars <- ""
        if (p_value < 0.10) {  stars <- "*" }
        if (p_value < 0.05) {  stars <- "**" }
        if (p_value < 0.01) {  stars <- "***" }
        result <- sprintf("%1.2f%2$-3s", number, stars)
    }

    return(result)
}

sanitize <- function(str) {
  result <- str
  result <- gsub("&", "\\&", result, fixed = TRUE)
  result <- gsub("_", "\\_", result, fixed = TRUE)

  return(result)
}

# Setup default charting layout
theme_report <- function (base_size = 30, base_family = "", ...)
{
  theme_bw(base_size = base_size, base_family = base_family) +
    theme(
      line = element_line(colour = "black", size = 4, linetype = "solid"),
      legend.position = "top",
```

```
      legend.key.height = unit(3,"line"),
      legend.justification = c("right", "top"),
      strip.text.x = element_text(size = base_size * 1.0, colour = "black", angle = 0)
      text         = element_text(colour="black", size = base_size * 1.0),
      axis.text    = element_text(angle=0,size = base_size * 1.0),
      axis.ticks = element_line(colour = "black", size = 2),
      legend.text  = element_text(angle=0,size = base_size * 1.0),
      # axis.title.x = element_text(size = base_size * 1.0),
      axis.title.y = element_text(size = base_size * 0.5),
      legend.title   =element_blank(),
      legend.key.width = unit(3, "cm"),
      panel.grid.minor = element_blank(),
      panel.grid = element_blank(),
      strip.background = element_rect(color = "black", size = 4),
      panel.background = element_rect(fill = NA, colour = "black", size=4)
    )
}
```

2.4 Setup Data

If the data can be found in the previously defined data directory, read it. Otherwise download it. We are using five different data files, of these we can automatically get four. But one—the linking suite—is not available via a database query and needs to be manually downloaded.

- Linking Suite from Daily TAQ to CRSP from WRDS
- Intraday Indicators using TAQ data from WRDS
- Industry Identification from French's website
- CRSP Header files from WRDS
- CRSP Daily files from WRDS

In the following section we read in the "Linking Suite, Daily TAQ CRSP", if available and otherwise stop with an error.

```
data_file_name <- paste(data_dir, "data_wrds_link_crsp_taq_202001_202002.csv",
    sep = "/")
if (!file.exists(data_file_name) & file.exists(data_dir)) {
    stop(paste("could not get file,", data_file_name,
        " need to download from WRDS (Linking Suite, Daily TAQ CRSP). As mentioned in
        sep = ""))
}
data_stock_match <- fread(data_file_name)

setnames(data_stock_match, "date", "DATE")
```

The data in the linking suite should look like:

```
head(data_stock_match, n = 2)

##    SYM_ROOT     DATE SYM_SUFFIX PERMNO    CUSIP   NCUSIP match_lvl
## 1:        A 20200102             87432 00846U10 00846U10         1
## 2:        A 20200103             87432 00846U10 00846U10         1
```

The next four code sections download the data, if necessary.

```
data_file_name <- paste(data_dir, "data_wrds_intraday_taq.csv",
    sep = "/")

if (!file.exists(data_file_name) & file.exists(data_dir)) {

    res <- dbSendQuery(wrds, "SELECT date, sym_root, sym_suffix, oprc, cprc,  total_do
    data_wrds_intraday <- dbFetch(res, n = -1)
    dbClearResult(res)

    data_wrds_intraday <- as.data.table(data_wrds_intraday)
```

```
    setnames(data_wrds_intraday, "date", "DATE")
    setnames(data_wrds_intraday, "sym_root", "SYM_ROOT")
    setnames(data_wrds_intraday, "sym_suffix", "SYM_SUFFIX")

    setnames(data_wrds_intraday, "quotedspread_percent_tw",
        "QuotedSpread_Percent_tw")
    setnames(data_wrds_intraday, "quotedspread_dollar_tw",
        "QuotedSpread_Dollar_tw")
    setnames(data_wrds_intraday, "effectivespread_percent_dw",
        "EffectiveSpread_Percent_DW")

    setnames(data_wrds_intraday, "oprc", "OPrc")
    setnames(data_wrds_intraday, "cprc", "CPrc")

    fwrite(data_wrds_intraday, file = data_file_name)

}

data_wrds_intraday <- fread(data_file_name)

# from PostregSQL Date is Date object from web it
# isnt, hence check
if (class(data_wrds_intraday[1, DATE])[1] == "integer") {
    data_wrds_intraday[, DATE := as.Date(as.character(DATE),
        format = "%Y%m%d")]
}
```

```
data_file_name <- paste(data_dir, "ff_sic_classification.csv",
    sep = "/")

if (!file.exists(data_file_name) & file.exists(data_dir)) {
    ff_file_name <- paste(data_dir, "Siccodes48.txt",
```

```
        sep = "/")
    if (!file.exists(ff_file_name)) {
        # https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/det_4
        url <- "https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/Siccodes48
        temp <- paste(data_dir, "ff_sic_classification.csv.zip",
            sep = "/")
        download.file(url, temp)
        unzip(temp, exdir = data_dir, files = "Siccodes48.txt")
    }

    pat <- "(?:^ *(\\d{1,2}) (.+) +(.+)$)|(?:^( +)(\\d{4})-(\\d{4}) +(.+)$)"
    res <- read.pattern(file = ff_file_name, pattern = pat,
        perl = TRUE, quote = NULL)

    res <- as.data.table(res)
    setnames(res, "V2", "industry")
    setnames(res, "V5", "sic_from")
    setnames(res, "V6", "sic_to")

    res[, sic_from := as.numeric(sic_from)]
    res[, sic_to := as.numeric(sic_to)]

    res[industry == "", industry := NA]

    res <- transform(res, industry = na.locf(industry))

    res <- res[!is.na(sic_to)]

    fwrite(res, file = data_file_name)
}

data_ff_industry <- read.csv(data_file_name)
```

```
no_match <- data.table(sic_from = 1, sic_to = 99999,
    industry = NA)

data_ff_industry <- data_ff_industry %>%
    add_row(no_match)
```

```
data_file_name <- paste(data_dir, "data_wrds_crsp_stock_header_20200501.csv",
    sep = "/")
if (!file.exists(data_file_name) & file.exists(data_dir)) {

    res <- dbSendQuery(wrds, "SELECT * FROM crsp.dsfhdr")
    data_stock_header <- dbFetch(res, n = -1)
    dbClearResult(res)

    data_stock_header <- as.data.table(data_stock_header)

    setnames(data_stock_header, "permno", "PERMNO")
    setnames(data_stock_header, "hshrcd", "HSHRCD")
    setnames(data_stock_header, "hprimexc", "HPRIMEXC")
    setnames(data_stock_header, "hsiccd", "HSICCD")

    fwrite(data_stock_header, file = data_file_name)
}
data_stock_header <- fread(data_file_name)

data_stock_header <- subset(data_stock_header, HSICCD >
    0)

data_stock_header <- ddply(data_stock_header, .(HSICCD),
    function(x) {
        x[, "ff_ind"] <- data_ff_industry[data_ff_industry[,
            "sic_from"] <= x[1, "HSICCD"] & x[1, "HSICCD"] <=
```

```
            data_ff_industry[, "sic_to"], "industry"][[1]]
        return(x)
    })
```

```
data_file_name <- paste(data_dir, "data_wrds_crsp_201912.csv",
    sep = "/")

if (!file.exists(data_file_name) & file.exists(data_dir)) {
    res <- dbSendQuery(wrds, "SELECT * FROM crsp.dsf where date > '2019-11-30' and dat
    data <- dbFetch(res, n = -1)
    dbClearResult(res)

    data_stock_2019 <- as.data.table(data)

    setnames(data_stock_2019, "prc", "PRC")
    setnames(data_stock_2019, "shrout", "SHROUT")
    setnames(data_stock_2019, "permno", "PERMNO")

    fwrite(data_stock_2019, file = data_file_name)
}

data_stock_2019 <- fread(data_file_name)

data_stock_2019[, mktcap := abs(PRC) * SHROUT]

data_stock_2019 <- sqldf("select PERMNO, avg(mktcap) as mktcap from data_stock_2019 gr
```

2.5 Setup Data 2 (merge)

Now we can merge all data into one data.table:

```
# total_dollar_m: \tTotal Trade Value in Dollars
# during market hours (Open to Close)
data_wrds_intraday_s <- data_wrds_intraday[, c("DATE",
    "SYM_ROOT", "SYM_SUFFIX", "total_dollar_m", "CPrc",
    "OPrc", "QuotedSpread_Percent_tw", "EffectiveSpread_Percent_DW")]

data_stock_match <- merge(data_stock_match, data_stock_header[,
    c("PERMNO", "HSHRCD", "HPRIMEXC", "HSICCD", "ff_ind")],
    by = ("PERMNO"))
data_stock_match <- merge(data_stock_match, data_stock_2019[,
    c("PERMNO", "mktcap")], by = c("PERMNO"))

data_stock_match[, DATE := as.Date(as.character(DATE),
    format = "%Y%m%d")]
data <- merge(data_wrds_intraday_s, data_stock_match,
    by = c("DATE", "SYM_ROOT"), all.x = T)

data <- ddply(data, .(SYM_ROOT), function(x) {
    y <- na.locf(x, na.rm = T)
    return(y)
})

data <- as.data.table(data)



data[DATE >= in_close_date, after := 1]
data[DATE < in_close_date, after := 0]
data[HPRIMEXC == "N", treated := 1]
data[HPRIMEXC != "N", treated := 0]
```

2.6 Setup Data 3 (filter)

As we describe in our paper we use U.S. common stocks with a CRSP share code of 10 or 11, that are listed on the NYSE or NASDAQ in December 2019; have only one share class; and a market capitalization above $500 million as of December 2019.

```
sample <- subset(data, HSHRCD == 10 | HSHRCD == 11)
sample <- subset(sample, HPRIMEXC == "N" | HPRIMEXC ==
    "Q")

sample <- subset(sample, mktcap > 5e+05)
sample <- subset(sample, SYM_SUFFIX.x == "")

sample_stocks <- length(row.names(unique(sample[, "PERMNO"])))
```

"The resulting sample has approximately 1,600 equities" and with the most recent data we now start with 1,614 stocks.

2.7 Setup Data 4 (scale)

We scale Effective Spreads to basis points and market capitalization to million USD.

```
sample[, "EffectiveSpread_Percent_DW"] <- sample[,
    "EffectiveSpread_Percent_DW"] * 10000

sample[, "total_dollar_m"] <- sample[, "total_dollar_m"]/1000
sample[, "mktcap"] <- sample[, "mktcap"]/1000
```

3 Results

3.1 Generate Table I

```
data_by_symbol <- subset(sample, DATE <= in_end_sample_long)

# to be consistent with paper, scale to PESPR in
# per cent for summary stats
data_by_symbol <- sqldf("select PERMNO,
                               count(*) as days,
                               avg(EffectiveSpread_Percent_DW)/100 as pespr,
                               avg(total_dollar_m) as volume,
                               avg(mktcap) as mktcap,
                               avg(CPrc) as close
                          from data_by_symbol group by PERMNO, HPRIMEXC")

data_means <- colMeans(data_by_symbol[, -1], na.rm = T)
obs_fct <- function(x) sum(!is.na(x))
data_obs <- colwise(obs_fct)(data_by_symbol)
data_sd <- sqrt(colwise(var)(data_by_symbol[, -1],
    na.rm = T))
data_min <- colwise(min)(data_by_symbol[, -1], na.rm = T)
data_median <- colwise(median)(data_by_symbol[, -1],
    na.rm = T)
data_max <- colwise(max)(data_by_symbol[, -1], na.rm = T)

probs <- c(0.05, 0.95)
quantiles <- colwise(quantile)(data_by_symbol[, -1],
    probs = probs, na.rm = T)
```

Table 1 – Cross-sectional summary statistics of time-series averages, Jan 2020 - Apr 2020

This table reports cross-sectional averages, standard deviations, and 5, 50, and 95 percentiles of daily time-series averages by stock. The sample consists of all U.S. common stocks listed on the NYSE or Nasdaq, with a single share class, and with a market capitalization of more than USD 500 million as of December 2019. PESPR is the dollar-weighted average of effective quoted spread (two times the difference between the signed trade price and the prevailing midpoint price, scaled by the midpoint price); Price is the closing price; Volume is trade volume in million dollars; Market Cap. is the stock price times shares outstanding in millions of U.S. dollars in December 2019. All variables (except Price and Market Cap.) are estimated over the continuous trading session only, i.e., from 9:30 ET to 16:00 ET.

	#Stocks	Mean	SD	5%	Median	95%
PESRP	1,614	0.16	0.13	0.05	0.13	0.38
Price	1,614	72.74	143.96	7.00	40.24	218.07
Volume	1,614	140,701	530,670	2,602	30,473	535,636
Mktcap	1,614	17,391	62,344	622	3,309	67,701

3.2 Propensity score matchings between treated and control stocks

```
sub_sample_psm <- subset(sample, DATE >= in_start_close_sample &
    DATE <= in_end_match)

query <- paste(sprintf("avg(%s) as %s", listing_cov,
    listing_cov), collapse = ", ")

# to be consistent with paper, scale to PESPR in
# per cent for summary stats
query <- sprintf("select PERMNO, avg(treated) as treated, ff_ind as ff_ind,
                    avg(EffectiveSpread_Percent_DW)/100 as EffectiveSpread_Percent_DW,
                    %s
                    from sub_sample_psm group by PERMNO, ff_ind",
    query)
query <- gsub("[\r\n]", "", query)

sub_sample_psm <- sqldf(query)
```

```
# see https://sejdemyr.github.io/r-tutorials/statistics/tutorial8.html

formula_did <- function(lhs, rhs) {
  formula <- as.formula(paste(lhs, " ~  ", paste(rhs, collapse = " + "), sep=""))
  return(formula)
}

sample_nomiss <- sub_sample_psm %>%  # MatchIt does not allow missing values
  select(PERMNO, EffectiveSpread_Percent_DW, ff_ind, treated, one_of(listing_cov)) %>%
  na.omit()

  mod_match <- matchit(formula_did("treated", c('ff_ind', listing_cov)),
```

```
                    method = "nearest", data = sample_nomiss);

  # Alternatively, match separately within industry
  # mod_match <- matchit(formula_did("treated", listing_cov), method = "nearest", ex

  dta_m <- match.data(mod_match);

sample_did <- merge(sample, dta_m[,c("PERMNO","weights")], by=c("PERMNO"))
```

```
sub_sample_psm <- subset(sample_did, DATE >= in_start_close_sample &
    DATE <= in_end_match)

sub_sample_psm <- sqldf(query)
```

3.3 Generate Table II

```
# Panel A
data_by_symbol <- subset(sample_did, DATE <= in_end_sample_long)

# to be consistent with paper, scale to PESPR in
# per cent for summary stats
data_by_symbol <- sqldf("select PERMNO,
                            count(*) as days,
                            avg(EffectiveSpread_Percent_DW)/100 as pespr,
                            avg(total_dollar_m) as volume,
                            avg(mktcap) as mktcap,
                            avg(CPrc) as close
                        from data_by_symbol group by PERMNO, HPRIMEXC")

data_means <- colMeans(data_by_symbol[, -1], na.rm = T)
obs_fct <- function(x) sum(!is.na(x))
```

```r
data_obs <- colwise(obs_fct)(data_by_symbol)
data_sd <- sqrt(colwise(var)(data_by_symbol[, -1],
    na.rm = T))
data_min <- colwise(min)(data_by_symbol[, -1], na.rm = T)
data_median <- colwise(median)(data_by_symbol[, -1],
    na.rm = T)
data_max <- colwise(max)(data_by_symbol[, -1], na.rm = T)


probs <- c(0.05, 0.95)
quantiles <- colwise(quantile)(data_by_symbol[, -1],
    probs = probs, na.rm = T)

# Panel B/C

m_ps <- glm(formula_did("treated", listing_cov), family = binomial(),
    data = sub_sample_psm)
prs_df <- data.frame(pr_score = predict(m_ps, type = "response"),
    treated = m_ps$model$treated)

psm_post <- ddply(prs_df, .(treated), function(x) {
    quantile(x[, "pr_score"], c(0.01, 0.05, 0.5, 0.95,
        0.99), na.rm = T)
})
row.names(psm_post) <- psm_post[, "treated"]

avg_covariates <- dta_m %>%
    group_by(treated) %>%
    select(one_of(c("EffectiveSpread_Percent_DW", listing_cov))) %>%
    summarise_all(mean)

avg_treated <- subset(avg_covariates, treated == 1)
```

```r
avg_control <- subset(avg_covariates, treated == 0)
avg_diff <- avg_control - avg_treated

t_tests <- lapply(c(listing_cov, "EffectiveSpread_Percent_DW"),
    function(v) {
        t.test(dta_m[, v] ~ dta_m$treated)
    })
```

Table 2 – Overview of propensity score matching
This table reports the propensity score matching (PSM) of treated stocks (NYSE listed stocks) and control stocks (Nasdaq listed stocks). We match stocks using one-to-one nearest neighbor PSM, without replacement, using average price and trading volume from one week before the event (i.e., from March 16, 2020 to March 20, 2020), Market Cap. as of December 2019, and industry using Fama and French 48 industry classifications. The event date is the closure of the NYSE floor on March 23, 2020. Panel A reports summary statistics for the matched sample of treatment and control firms. Panel B reports the 1, 5, 50, 95, and 99 percentiles of estimated propensity scores for the matched sample. Panel C reports univariate comparisons between stock characteristics of the matched sample of treatment and control stocks.

	#Stocks	Mean	SD	5%	Median	95%
PESRP	1,238	0.17	0.14	0.05	0.13	0.41
Price	1,238	71.12	150.27	6.60	40.20	202.63
Volume	1,238	142,068	587,070	2,369	27,489	479,205
Mktcap	1,238	17,895	68,888	609	3,004	65,880

Panel B: Estimated Propensity Score Distributions

	1%	5%	50%	95%	99%
Treatment	0.38	0.45	0.50	0.58	0.82
Control	0.17	0.42	0.50	0.52	0.64

Panel C: Differences in covariates pre-even

	Treatment	Control	Diff	t-stat
PESPR	0.28	0.40	0.12	7.52
CPrc	51.61	58.15	6.54	0.98
total_dollar_m	179,571	223,457	43,886	0.86
mktcap	19,122	16,667	-2,456	-0.63

```
sub_sample_did <- subset(sample_did, DATE >= in_start_close_sample &
    DATE <= in_end_close_sample)
```

3.4 Generate Table III

```
sample_p <- pdata.frame(subset(sub_sample_did, QuotedSpread_Percent_tw >
    0), index = c("PERMNO", "DATE"), row.names = TRUE)

model1 <- (plm(EffectiveSpread_Percent_DW ~ treated *
    after, data = sample_p, model = "within"))
model2 <- (plm(EffectiveSpread_Percent_DW ~ treated *
    after, data = sample_p, model = "within", effect = "twoways"))
model3 <- (plm(EffectiveSpread_Percent_DW ~ treated *
    after + CPrc + log(total_dollar_m), data = sample_p,
    model = "within"))
model4 <- (plm(EffectiveSpread_Percent_DW ~ treated *
    after + CPrc + log(total_dollar_m), data = sample_p,
    model = "within", effect = "twoways"))

# using cluster robust standard errors (see
# https://stackoverflow.com/ Question 33155638)
cluster1 <- coeftest(model1, vcovHC(model1, type = "sss",
    cluster = "group"))
cluster2 <- coeftest(model2, vcovHC(model2, type = "sss",
    cluster = "group"))
cluster3 <- coeftest(model3, vcovHC(model3, type = "sss",
    cluster = "group"))
cluster4 <- coeftest(model4, vcovHC(model4, type = "sss",
    cluster = "group"))

reg1 <- summary(model1)
```

```
reg2 <- summary(model2)
reg3 <- summary(model3)
reg4 <- summary(model4)

days1 <- length(unique(as.character(sample_p[rownames(reg1[["model"]]),
    "DATE"])))
days2 <- length(unique(as.character(sample_p[rownames(reg2[["model"]]),
    "DATE"])))
days3 <- length(unique(as.character(sample_p[rownames(reg3[["model"]]),
    "DATE"])))
days4 <- length(unique(as.character(sample_p[rownames(reg4[["model"]]),
    "DATE"])))
firms1 <- length(unique(as.character(sample_p[rownames(reg1[["model"]]),
    "PERMNO"])))
firms2 <- length(unique(as.character(sample_p[rownames(reg2[["model"]]),
    "PERMNO"])))
firms3 <- length(unique(as.character(sample_p[rownames(reg3[["model"]]),
    "PERMNO"])))
firms4 <- length(unique(as.character(sample_p[rownames(reg4[["model"]]),
    "PERMNO"])))

# Alternatively use feols for faster estimation
# library(fixest) model1 <-
# feols(EffectiveSpread_Percent_DW ~
# treated*after | PERMNO , data=sample_p) reg1 <-
# summary( model1, se='cluster',
# cluster=~PERMNO)[['coeftable']]
# reg1[['coefficients']] <- reg1
# reg1[['r.squared']][['rsq']] <- 1 -
# model1£ssr/model1£ssr_fe_only cluster1 <- reg1
```

Table 3 – Difference-in-differences Regression: PESPR

As Table III of the paper.
This table shows results from a difference-in-differences regression of proportional effective spreads (*PESPR*) before versus after the closure of the NYSE floor. Panel A compares NYSE listed stocks to a matched sample of Nasdaq stocks. We estimate the following fixed-effect panel regression using data from one-week before to one-week after the closure of the NYSE floor (March 16, 2020 through March 27, 2020):

$$PESPR_{i,e,t} = \beta_1(Treated_{i,e,t} \times After_{i,e,t}) + \beta_2(Treated_{i,e,t}) + \beta_3(After_{i,e,t}) + \gamma(C_{i,e,t}) + FE + \epsilon_{i,e,t},$$

where $PESPR_{i,e,t}$ is the effective spread of stock i on day t trading on exchange e; $After_{i,e,t}$ is an indicator variable equal to one after the closure of the NYSE floor (March 23, 2020) and zero otherwise; $Treated_{i,e,t}$ is an indicator variable equal to one if the stock is listed on the NYSE (Panel A) or equal to one if $PESPR$ is estimated on the NYSE (Panel B) and zero otherwise; and $C_{i,e,t}$ is a vector of additional control variables including the natural log of trading volume ($Log(Volume)$) and parity trading volume ($Parityvolume$). *PESPR* is estimated as the trade price across all exchanges in excess of the prevailing midpoint price using NBBO bid and ask prices. t-statistics calculated using standard errors clustered by stock are shown below the estimates. Statistical significance at the 1%, 5%, and 10% level is indicated by ***, **, and *.

	PESPR (1)	PESPR (2)	PESPR (3)	PESPR (4)
Panel A: NYSE- ($Treated = 1$) versus Nasdaq- ($Treated = 0$) listed stocks				
$treated \times after$	5.18***	5.18***	5.14***	5.18***
	(6.16)	(6.16)	(6.11)	(6.19)
$after$	-7.33***		-7.33***	
	(-10.60)		(-10.54)	
$Price$			-0.05**	0.05
			(-2.15)	(1.42)
$log(volume)$			-0.53	-2.83***
			(-0.57)	(-2.70)
Within R^2 [%]	1.53	0.39	1.59	0.55
#Stocks	1,238	1,238	1,238	1,238
#Days	10	10	10	10
Stock FE	Yes	Yes	Yes	Yes
Day FE	No	Yes	No	Yes

```
window_open1 <- interval(ymd(as.Date(in_start_open1_sample)),
    ymd(as.Date(in_end_open1_sample)))
window_open2 <- interval(ymd(as.Date(in_start_open2_sample)),
    ymd(as.Date(in_end_open2_sample)))

sample_did[, open1 := ifelse(DATE >= in_open1_date,
    1, 0)]
sample_did[, open2 := ifelse(DATE >= in_open2_date,
    1, 0)]

sample_did[, event := ifelse(DATE %within% window_open1,
    2, 0)]
sample_did[event == 0, event := ifelse(DATE %within%
    window_open2, 3, 0)]


sub_sample_did <- subset(sample_did, DATE %within%
    window_open1 | DATE %within% window_open2)
```

3.5 Generate Table VII

```
# Using WRDS calculations
diff_data_taq <- sub_sample_did
diff_data_taq[, SYMBOL := paste(SYM_ROOT, as.character(event))]

sample_p <- pdata.frame(subset(diff_data_taq, QuotedSpread_Percent_tw >
    0), index = c("SYMBOL", "DATE"), row.names = TRUE)

model1 <- (plm(EffectiveSpread_Percent_DW ~ treated *
    open1 + treated * open2, data = sample_p, model = "within"))
model2 <- (plm(EffectiveSpread_Percent_DW ~ treated *
    open1 + treated * open2, data = sample_p, model = "within",
```

```
    effect = "twoways"))
model3 <- (plm(EffectiveSpread_Percent_DW ~ treated *
    open1 + treated * open2 + CPrc + log(total_dollar_m),
    data = sample_p, model = "within"))
model4 <- (plm(EffectiveSpread_Percent_DW ~ treated *
    open1 + treated * open2 + CPrc + log(total_dollar_m),
    data = sample_p, model = "within", effect = "twoways"))

# using cluster robust standard errors (see
# https://stackoverflow.com/questions/33155638/clustered-standard-errors-in-r-using-
cluster1 <- coeftest(model1, vcovHC(model1, type = "sss",
    cluster = "group"))
cluster2 <- coeftest(model2, vcovHC(model2, type = "sss",
    cluster = "group"))
cluster3 <- coeftest(model3, vcovHC(model3, type = "sss",
    cluster = "group"))
cluster4 <- coeftest(model4, vcovHC(model4, type = "sss",
    cluster = "group"))

reg1 <- summary(model1)
reg2 <- summary(model2)
reg3 <- summary(model3)
reg4 <- summary(model4)

days1 <- length(unique(as.character(sample_p[rownames(reg1[["model"]]),
    "DATE"])))
days2 <- length(unique(as.character(sample_p[rownames(reg2[["model"]]),
    "DATE"])))
days3 <- length(unique(as.character(sample_p[rownames(reg3[["model"]]),
    "DATE"])))
days4 <- length(unique(as.character(sample_p[rownames(reg4[["model"]]),
    "DATE"])))
```

```
firms1 <- length(unique(as.character(sample_p[rownames(reg1[["model"]]),
    "PERMNO"])))
firms2 <- length(unique(as.character(sample_p[rownames(reg2[["model"]]),
    "PERMNO"])))
firms3 <- length(unique(as.character(sample_p[rownames(reg3[["model"]]),
    "PERMNO"])))
firms4 <- length(unique(as.character(sample_p[rownames(reg4[["model"]]),
    "PERMNO"])))
```

Table 7 – Difference-in-differences Regression around Re-openings: PESPR

As Panel A of Table VII of the paper.
This table shows results from a difference-in-differences regression of proportional effective spreads (*PESPR*) before versus after the partial re-openings of the NYSE floor. Panel A compares NYSE listed stocks to a matched sample of Nasdaq stocks, while Panel B compares *PESPR* of NYSE listed stocks trading on the NYSE to *PESPR* in those same stocks trading elsewhere (i.e., off the NYSE). We estimate the following fixed-effect panel regression using data from one-week before to one-week after both times (events) the NYSE Floor partially re-opened (May 18, 2020 through June 1, 2020 and June 10, 2020 through June 23, 2020):

$$\begin{aligned} PESPR_{i,e,t} =& \beta_1(Treated_{i,e,t} \times Open1_{i,e,t}) + \beta_2(Treated_{i,e,t} \times Open2_{i,e,t}) + \beta_3(Treated_{i,e,t}) \\ &+ \beta_4(Open1_{i,e,t}) + \beta_5(Open2_{i,e,t}) + \gamma(C_{i,e,t}) + FE + \epsilon_{i,e,t}, \end{aligned}$$

where $PESPR_{i,e,t}$ is the effective spread of stock i on day t trading on exchange e; $Open1_{i,e,t}$ and $Open2_{i,e,t}$ are indicator variables equal to one, respectively, after the first and second partial re-opening of the NYSE floor (May 26, 2020 and June 17, 2020, respectively) and zero otherwise; $Treated_{i,e,t}$ is an indicator variable equal to one if the stock is listed on the NYSE (Panel A) or equal to one if $PESPR$ is estimated on the NYSE (Panel B) and zero otherwise; and $C_{i,e,t}$ is a vector of additional control variables including the natural log of trading volume ($Log(Volume)$) and parity trading volume ($Parityvolume$). In Panel A, *PESPR* is estimated as the trade price across all exchanges in excess of the prevailing midpoint price using NBBO bid and ask prices. In Panel B, *PESPR* on the NYSE is estimated from trades on the NYSE and midpoint prices using bid and ask prices on the NYSE. *PESPR* on other exchanges is estimated from trades off the NYSE and midpoint prices using the NBBO excluding NYSE bid and ask prices (i.e., using the best bid and ask prices across all exchanges excluding the NYSE). t-statistics calculated using standard errors clustered by stock are shown below the estimates. Statistical significance at the 1%, 5%, and 10% level is indicated by ***, **, and *.

	PESPR (1)	PESPR (2)	PESPR (3)	PESPR (4)
$treated \times open1$	0.18	0.18	0.31	0.19
	(0.41)	(0.41)	(0.69)	(0.43)
$treated \times open2$	-2.73***	-2.73***	-2.48***	-2.78***
	(-5.09)	(-5.09)	(-4.66)	(-5.25)
$open1$	-0.38		-1.47***	
	(-1.15)		(-3.69)	
$open2$	2.95***		3.19***	
	(6.25)		(6.47)	
$Price$			-0.06***	-0.02*
			(-3.24)	(-1.80)
$log(volume)$			4.99***	-0.39
			(6.56)	(-0.50)
Within R^2 [%]	0.35	0.17	2.22	0.18
#Stocks	1,233	1,233	1,233	1,233
#Days	20	20	20	20
Stock FE	Yes	Yes	Yes	Yes
Day FE	No	Yes	No	Yes

3.6 Generate Figure 1

```
diff_data_taq <- sample_did[DATE <= "2020-06-30", .(pespr = mean(EffectiveSpread_Perce
    by = .(DATE, PERMNO)]

diff_data_taq <- pdata.frame(diff_data_taq, index = c("PERMNO",
    "DATE"), row.names = TRUE, stringsAsFactors = F)

diff_data_taq[, "pespr"] <- 100 * log(diff_data_taq[,
    "pespr"]/lag(diff_data_taq[, "pespr"]))

diff_data_taq[is.na(diff_data_taq[, "pespr"]), "pespr"] <- 0
diff_data_taq <- ddply(diff_data_taq, .(PERMNO), function(x) {
    x[, "sum_pespr"] <- cumsum(x[, "pespr"])
    return(x)
})
diff_data_taq[, "pespr"] <- diff_data_taq[, "sum_pespr"]
diff_data_taq[, "DATE"] <- as.Date(diff_data_taq[,
    "DATE"])



diff_data_taq <- reshape2::melt(diff_data_taq, id.vars = c("DATE",
    "PERMNO"), measure.vars = c("pespr"))

diff_data_taq <- merge(diff_data_taq, sample_did[,
    c("PERMNO", "DATE", "HPRIMEXC", "total_dollar_m",
        "CPrc", "mktcap")], by = c("PERMNO", "DATE"))

names(diff_data_taq)[names(diff_data_taq) == "value"] <- "pespr"

# diff_data_taq <- sqldf('select DATE, HPRIMEXC,
# avg(pespr) as pespr from diff_data_taq group by
# DATE, HPRIMEXC')
```

```
diff_data_taq <- sqldf("select DATE, HPRIMEXC, sum(mktcap * pespr) as pespr, sum(mktca
diff_data_taq[, "pespr"] <- diff_data_taq[, "pespr"]/diff_data_taq[,
    "mktcap"]

diff_data_taq[diff_data_taq[, "HPRIMEXC"] == "N", "HPRIMEXC"] <- "NYSE"
diff_data_taq[diff_data_taq[, "HPRIMEXC"] == "Q", "HPRIMEXC"] <- "Nasdaq"


plot <- ggplot(diff_data_taq, aes(x = DATE, y = pespr)) +
    geom_line(aes(color = as.factor(HPRIMEXC))) + geom_jitter(aes(color = as.factor(HP
    geom_vline(xintercept = as.Date(in_close_date)) +
    geom_vline(xintercept = as.Date(in_open1_date)) +
    geom_vline(xintercept = as.Date(in_open2_date)) +
    ylab("Percent change in PESPR relative to Jan. 1, 2020") +
    xlab("") + theme_report()
```

Figure 1 – Percent change in daily PESPR for NYSE stocks relative to NASDAQ stocks

This figure shows the change in proportional effective spreads (*PESPR*) for all matched NYSE and Nasdaq listed stocks in our sample. Each date, for each stock, we calculate the percent change in *PESPR* relative to January 1, 2020 and plot the daily market cap-weighted average *PESPR* across all stocks in our sample. *PESPR* is estimated as the trade price across all exchanges in excess of the prevailing midpoint price using NBBO bid and ask prices. Vertical lines indicate the closing (March 23, 2020), first partial reopening (May 26, 2020), and second partial reopening (June 17, 2020) of the NYSE trading Floor.

```
print(plot)
```

