# BVI Case Study

Karen Nershi

2025-04-08

## BVI Case Study

The following code generates output for Tables 5-6 in the main body of the paper; it also generates output for Tables 10-12 and Figures 3-4 in the Supplementary Material. This output matches the code included in the file "BVI Case Study.R."

```
################################################################################
################ PART 1: LOAD PACKAGES AND FUNCTIONS FOR ANLAYSIS ###############
################################################################################

################################################################################
### A. PACKAGES

# Packages used in analysis
pkg = c("tidyverse","stargazer","knitr","kableExtra","lmtest","sandwich")

# Checks if installed; if not, install
if (length(setdiff(pkg, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(pkg, rownames(installed.packages())))
}

# Load packages
suppressMessages(suppressWarnings(lapply(pkg, library, character.only = TRUE)))
```

```
## [[1]]
##  [1] "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"     "readr"
##  [7] "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"     "graphics"
## [13] "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[2]]
##  [1] "stargazer" "lubridate" "forcats"   "stringr"   "dplyr"     "purrr"
##  [7] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse" "stats"
## [13] "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
##
## [[3]]
##  [1] "knitr"     "stargazer" "lubridate" "forcats"   "stringr"   "dplyr"
##  [7] "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
## [13] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [19] "base"
##
## [[4]]
##  [1] "kableExtra" "knitr"      "stargazer"  "lubridate"  "forcats"
##  [6] "stringr"    "dplyr"      "purrr"      "readr"      "tidyr"
```

```
## [11] "tibble"     "ggplot2"    "tidyverse"  "stats"      "graphics"
## [16] "grDevices"  "utils"      "datasets"   "methods"    "base"
##
## [[5]]
##  [1] "lmtest"     "zoo"        "kableExtra" "knitr"      "stargazer"
##  [6] "lubridate"  "forcats"    "stringr"    "dplyr"      "purrr"
## [11] "readr"      "tidyr"      "tibble"     "ggplot2"    "tidyverse"
## [16] "stats"      "graphics"   "grDevices"  "utils"      "datasets"
## [21] "methods"    "base"
##
## [[6]]
##  [1] "sandwich"   "lmtest"     "zoo"        "kableExtra" "knitr"
##  [6] "stargazer"  "lubridate"  "forcats"    "stringr"    "dplyr"
## [11] "purrr"      "readr"      "tidyr"      "tibble"     "ggplot2"
## [16] "tidyverse"  "stats"      "graphics"   "grDevices"  "utils"
## [21] "datasets"   "methods"    "base"
```

```r
rm(list=ls())

################################################################################
### B. FUNCTIONS

# Create a function that will prepare data given range of interest

process_trading_data <- function(df1, lower_bound, upper_bound) {
  require(dplyr)

  # Filter data based on bounds
  df <- filter(df1, base > lower_bound & base < upper_bound)

  # Create a function to round the dollar values down to the nearest 10
  round_down <- function(numbers) {
    rounded_numbers <- floor(numbers / 10) * 10
    return(rounded_numbers)
  }

  # Apply function to dollar values to identify bins
  df$dollar_bins <- round_down(df$base)

  # Create an identifier for each exchange-trading pair
  df$identifier <- paste0(df$s, "_", df$exchange, "_", df$type)

  # Identify treated entities
  treated_entities <- unique(select(filter(df, type == "regulated"), identifier))

  # Summarize the total trades by day, bin, and exchange-trading pair
  df2 <- df %>%
    group_by(m_d, dollar_bins, identifier) %>%
    summarise(total_trades = n(), .groups = "drop")

  # Create a new dataframe that identifies all day-bin-exchange-trading pair combinations
  m_d <- unique(df$m_d)
  bins_in_range <- unique(df$dollar_bins)
  id_list <- unique(df$identifier)
```

```r
  # Create complete combination grid
  df3 <- expand.grid(m_d = m_d, dollar_bins = bins_in_range, identifier = id_list)

  # Merge this with your summarized data
  df4 <- df3 %>%
    left_join(df2, by = c("m_d", "dollar_bins", "identifier"))

  # Fills NAs in total_trades with zeros
  df4$total_trades <- ifelse(is.na(df4$total_trades), 0, df4$total_trades)

  # Create a new variable for all daily trades within the range for each exchange-trading pair
  df4 <- df4 %>%
    group_by(m_d, identifier) %>%
    mutate(daily_trades_range = sum(total_trades)) %>%
    ungroup()

  # Create a new variable for the proportion of trades within each bin divided by all daily trades
  # If all daily trades within the range for an exchange-trading pair is zero, set prop_trades to zero
  df4$prop_trades <- ifelse(df4$daily_trades_range == 0, 0,
                            df4$total_trades / df4$daily_trades_range)

  ### DID Specific variables
  # Create a variable for treated units
  df4$treat <- ifelse(df4$identifier %in% treated_entities$identifier, 1, 0)

  # Create a variable for the post-treatment period
  df4$time <- ifelse(df4$m_d > "07:09", 1, 0)

  # Create a variable to identify each bin for each exchange-trading pair
  df4$id_2 <- paste0(df4$identifier, "_", df4$dollar_bins)

  return(df4)
}

# Create function for estimating bunching

estimate_beta = function(thedata, z_vector, binv, zstar, bins_excl_l, bins_excl_r, rates) {
  z_vector = thedata$base

  # set parameters
  binwidth = 10 # 10 units for each bin
  bins_l = 25
  bins_r = 25
  poly = 3 # degree of polynomial
  n_boot = 1000 # number of iterations for caclulating bootstrapped standard errors
  rn = NA
  correct_above_zu = FALSE

  zmax <- zstar + (binwidth * bins_r)
  zmin <- zstar - (binwidth * bins_l)
  bins <- seq(zmin, zmax, by = binwidth)

  ##Cut the bins
```

```
thebin <- cut(z_vector, bins, right = FALSE, labels = FALSE)
thebin <- zmin + binwidth * (thebin - 1)
thedata <- data.frame(z = z_vector, bin = thebin)
thedata <- thedata %>% dplyr::group_by(bin) %>%
  dplyr::summarise(freq = n(), z = mean(z, na.rm = TRUE)) %>%
  dplyr::filter(!is.na(bin))
thedata$freq_orig <- thedata$freq
thedata <- as.data.frame(thedata)

data_binned = thedata

data_binned$z_rel = (data_binned$bin - zstar)/binwidth
data_binned$zstar <- ifelse(data_binned$bin == zstar, 1,
                            0)
extra_fe_vector <- ""

polynomial_vector <- c()
for (i in seq(poly)) {
  poly_varname <- paste0("poly_", i)
  data_binned[[poly_varname]] <- data_binned$z^i
  polynomial_vector <- c(polynomial_vector, poly_varname)
}
bins_excluded_all <- c()
if (bins_excl_l != 0) {
  bins_excl_l_vector <- c()
  for (i in seq(bins_excl_l)) {
    bins_excl_l_varname <- paste0("bin_excl_l_", i)
    data_binned[[bins_excl_l_varname]] <- ifelse(data_binned$z_rel ==
                                                   -i, 1, 0)
    bins_excl_l_vector <- c(bins_excl_l_vector, bins_excl_l_varname)
  }
  bins_excluded_all <- c(bins_excluded_all, bins_excl_l_vector)
}
if (bins_excl_r != 0) {
  bins_excl_r_vector <- c()
  for (i in seq(bins_excl_r)) {
    bin_excl_r_varname <- paste0("bin_excl_r_", i)
    data_binned[[bin_excl_r_varname]] <- ifelse(data_binned$z_rel ==
                                                  i, 1, 0)
    bins_excl_r_vector <- c(bins_excl_r_vector, bin_excl_r_varname)
  }
  bins_excluded_all <- c(bins_excluded_all, bins_excl_r_vector)
}
if (length(bins_excluded_all) > 0) {
  data_binned$bunch_region <- rowSums(data_binned[, c("zstar",
                                                      bins_excluded_all)])
}
data_binned$bin_above_excluded <- ifelse(data_binned$bin >
                                           zstar, 1, 0)
rn_vector <- ""
rhs_vars <- c("zstar", extra_fe_vector, polynomial_vector,
              rn_vector, bins_excluded_all)
rhs_vars <- setdiff(rhs_vars, "")
```

```
model_formula <- stats::as.formula(paste0("freq", " ~ ",
                                          paste(rhs_vars, collapse = " +")))

data_forreg <- list(data_binned = data_binned, model_formula = model_formula)

#######
thedata = data_forreg$data_binned
themodelformula = data_forreg$model_formula
notch = FALSE
zD_bin = NA

# Define model outcomes
model_fit <- stats::lm(themodelformula, thedata)
coefficients <- summary(model_fit)$coefficients
residuals <- stats::residuals(model_fit)
thedata$cf <- stats::predict(model_fit, thedata)
thedata$cf <- thedata$cf - (thedata$zstar * coefficients["zstar",
                                                         "Estimate"])
bins_excluded_in_reg <- rownames(coefficients)[grepl("bin_excl",
                                                     rownames(coefficients))]
for (i in bins_excluded_in_reg) {
  thedata$cf <- thedata$cf - (thedata[[i]] * coefficients[i,
                                                          "Estimate"])
}
bins_zstar_zu <- sum(grepl("bin_excl_r", rownames(coefficients)))
bins_zl_zstar <- sum(grepl("bin_excl_l", rownames(coefficients))) +
  1
zstarvalue <- thedata$bin[thedata$zstar == 1]
zstarvalue = zstarvalue[1]
binwidthvalue <- thedata$bin[2] - thedata$bin[1]

#Excluded bins below threshold
thedata$zl_zstar <- ifelse(((thedata$bin >=
                               (zstarvalue -
                                  (binwidthvalue * (bins_zl_zstar - 1)))) &
                              (thedata$bin <= zstarvalue)),
                           1, 0)
thedata$zstar_zu <- ifelse((thedata$bin <=
                              zstarvalue + (binwidthvalue *
                                              bins_zstar_zu)) &
                             (thedata$bin > zstarvalue), 1, 0)
thedata$bunch_region <- ifelse(thedata$zl_zstar == 1, "zl_zstar",
                               ifelse(thedata$zstar_zu == 1,
                                      "zstar_zu", "outside_bunching"))
bunching_region_count <- thedata %>% dplyr::group_by(bunch_region) %>%
  dplyr::summarize(actual = sum(freq_orig), cf = sum(cf),
                   excess = actual - cf)

B_zl_zstar <- as.numeric(subset(bunching_region_count, bunch_region ==
                                  "zl_zstar", select = 'excess'))
B_zstar_zu <- as.numeric(subset(bunching_region_count, bunch_region ==
                                  "zstar_zu", select = "excess"))
if (is.na(B_zstar_zu)) {
```

```
    B_zstar_zu <- 0
  }
  bunching_excess <- B_zl_zstar + B_zstar_zu
  cf_bunching <- sum(subset(bunching_region_count, bunch_region !=
                              "outside_bunching", select = "cf"))
  bins_bunching <- sum(thedata$bunch_region %in% c("zl_zstar",
                                                   "zstar_zu"))
  c0 <- cf_bunching/bins_bunching
  b_estimate <- as.numeric(sprintf("%.3f", bunching_excess/c0))

  ##Calculate standard errors
  data_for_boot <- thedata
  model <- themodelformula

  boot_results <- sapply(seq(1:n_boot), function(i) {
    data_for_boot$freq_orig <- data_for_boot$freq_orig +
      sample(residuals, replace = TRUE)
    data_for_boot$freq <- data_for_boot$freq_orig
    booted_firstpass <- bunching::fit_bunching(data_for_boot,
                                               model, binwidth, notch, zD_bin)
  })
  set.seed(99)
  ans = list()
  extract_me <- function(boot_results, ans) {
    thelist = seq(7, 12994, 13)
    for (i in thelist) {
      a = boot_results[[i]]
      ans <- append(ans, a)
    }
    return(ans)
  }

  betas = extract_me(boot_results = boot_results, ans = ans)
  betas = unlist(betas, use.names=FALSE)
  b_sd <- stats::sd(betas, na.rm = TRUE)

  #####Create df
  info = cbind.data.frame(b_estimate, b_sd)
  names(info) = c('beta', 'standard error')
  info$threshold = zstar
  info <- info[,c(3,1:2)]
  return(info)
}

# Create a function to round values within a column
rounding = function(x) {
  A = round(x,digits=3)
  return(A)
}


################################################################################
################## PART 2: ANALYSIS ############################################
```

```r
##############################################################################

##############################################################################
### TABLE 5
##############################################################################

# Set working directory to location of data files
# setwd("path to files")
setwd("/Users/knershi@middlebury.edu/Documents/Temporary/")

## Part 1: Data preparation

# Import data
df.btc <- read.csv("full.BVI.btc.csv")
df.eth <- read.csv("full.BVI.eth.csv")

# Subset data to pre-period
df.btc_pre <- df.btc %>%
  filter(m_d < "07:10")
df.eth_pre <- df.eth %>%
  filter(m_d < "07:10")

# Prep the data using the function
lower_bound = 749
upper_bound = 1250
p.btc <- process_trading_data(df.btc_pre, lower_bound = lower_bound, upper_bound = upper_bound)
p.eth <- process_trading_data(df.eth_pre, lower_bound = lower_bound, upper_bound = upper_bound)

## Part 2: Prepare descriptive statistics

# Identify Bitcoin vs. Ethereum transactions

# Calculate N exchanges
N_exchanges <- c(length(unique(df.btc[(df.btc$type=="regulated"),]$exchange)),
                 length(unique(df.btc[(df.btc$type=="unregulated"),]$exchange)),
                 length(unique(df.eth[(df.eth$type=="regulated"),]$exchange)),
                 length(unique(df.eth[(df.eth$type=="unregulated"),]$exchange))
)

# Calculate N trading pairs
N_trading_pairs <- c(length(unique(p.btc[(p.btc$treat==1),]$identifier)),
                     length(unique(p.btc[(p.btc$treat==0),]$identifier)),
                     length(unique(p.eth[(p.eth$treat==1),]$identifier)),
                     length(unique(p.eth[(p.eth$treat==0),]$identifier)))

# Calculate N transactions
N_transactions <- c(nrow(df.btc[(df.btc$type=="regulated"),]),
                    nrow(df.btc[(df.btc$type=="unregulated"),]),
                    nrow(df.eth[(df.eth$type=="regulated"),]),
                    nrow(df.eth[(df.eth$type=="unregulated"),]))


## Difference in means t-test
```

```r
# 1 bin below the threshold

# T.test difference in means
t1 <- t.test(p.btc[(p.btc$treat==1 & p.btc$dollar_bins==990),]$prop_trades, p.btc[(p.btc$treat==0 & p.b
means_t1 <- c(round(t1$estimate[1],3),round(t1$estimate[2],3))

t2 <- t.test(p.eth[(p.eth$treat==1 & p.eth$dollar_bins==990),]$prop_trades, p.eth[(p.eth$treat==0 & p.e
means_t2 <- c(round(t2$estimate[1],3),round(t2$estimate[2],3))

# 5 bins below the threshold

## T.test range of interest
t3 <- t.test(p.btc[(p.btc$treat==1 & p.btc$dollar_bins > 940 & p.btc$dollar_bins < 1000),]$prop_trades,
means_t3 <- c(round(t3$estimate[1],3),round(t3$estimate[2],3))

t4 <- t.test(p.eth[(p.eth$treat==1 & p.eth$dollar_bins > 940 & p.eth$dollar_bins < 1000),]$prop_trades,
means_t4 <- c(round(t4$estimate[1],3),round(t4$estimate[2],3))

## Bind this info into one data frame and display

# Combine means into rows
means_1 <- c(means_t1,means_t2)
means_2 <- c(means_t3,means_t4)

table_df <- rbind.data.frame(N_exchanges,
                             N_trading_pairs,
                             N_transactions,
                             means_1,
                             means_2)
names(table_df) <- c("BVI (btc)","Unregulated (btc)","BVI (eth)","Unregulated (eth)")
rownames(table_df) <- c("N_exchanges","N_trading_pairs","N_transactions",
                        "Mean Proportion of Trades 1 Bin Below Threshold",
                        "Mean Proportion of Trades 5 Bins Below Threshold")

table_df$`P-values (btc)`<- c("","","",round(t1$p.value,3),round(t3$p.value,3))
table_df$`P-values (eth)`<- c("","","",round(t2$p.value,3),round(t4$p.value,3))

print(table_df)
```

```
##                                                BVI (btc) Unregulated (btc)
## N_exchanges                                        2.000             4.000
## N_trading_pairs                                    4.000             4.000
## N_transactions                                1126824.000       1727534.000
## Mean Proportion of Trades 1 Bin Below Threshold       0.096             0.024
## Mean Proportion of Trades 5 Bins Below Threshold      0.038             0.016
##                                                BVI (eth) Unregulated (eth)
## N_exchanges                                        2.000             3.000
## N_trading_pairs                                    2.000             3.000
## N_transactions                                597561.000        925938.000
## Mean Proportion of Trades 1 Bin Below Threshold       0.027             0.031
## Mean Proportion of Trades 5 Bins Below Threshold      0.024             0.018
##                                               P-values (btc) P-values (eth)
## N_exchanges
```

```
## N_trading_pairs
## N_transactions
## Mean Proportion of Trades 1 Bin Below Threshold                0          0.727
## Mean Proportion of Trades 5 Bins Below Threshold               0          0.059
```

```r
################################################################################
### TABLE 6
################################################################################

# To avoid confusion, remove all objects created for previous table
# Define a list of prefixes you want to remove
prefixes_to_remove <- c("p.btc","p.eth",
                        "df.btc_pre","df.eth_pre")

# Create a pattern that matches any of these prefixes at the beginning of object names
pattern <- paste0("^(", paste(prefixes_to_remove, collapse="|"), ")")

# Remove objects matching the pattern
rm(list=ls(pattern=pattern))

# Prep the data using the function
lower_bound = 749
upper_bound = 1250
p.btc <- process_trading_data(df.btc, lower_bound = lower_bound, upper_bound = upper_bound)
p.eth <- process_trading_data(df.eth, lower_bound = lower_bound, upper_bound = upper_bound)

## Create two ranges of 50 and 10 dollars below the threshold for each
btc_range1 <- filter(p.btc, dollar_bins > 949 & dollar_bins < 1000)
btc_range2 <- filter(p.btc, dollar_bins > 989 & dollar_bins < 1000)

eth_range1 <- filter(p.eth, dollar_bins > 949 & dollar_bins < 1000)
eth_range2 <- filter(p.eth, dollar_bins > 989 & dollar_bins < 1000)

# Create difference-in-differences models for each range
btc_mod1 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),btc_range1)
btc_mod2 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),btc_range2)

eth_mod1 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),eth_range1)
eth_mod2 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),eth_range2)

## Extract R^2, adjusted R^2, and N for each model
# mod1
b_mod1_r2 <- round(summary(btc_mod1)$r.squared,3)
b_mod1_adj.r2 <- round(summary(btc_mod1)$adj.r.squared,3)
b_N1 <- nrow(btc_range1)

# mod2
b_mod2_r2 <- round(summary(btc_mod2)$r.squared,3)
b_mod2_adj.r2 <- round(summary(btc_mod2)$adj.r.squared,3)
b_N2 <- nrow(btc_range2)

# mod1
e_mod1_r2 <- round(summary(eth_mod1)$r.squared,3)
e_mod1_adj.r2 <- round(summary(eth_mod1)$adj.r.squared,3)
```

```r
e_N1 <- nrow(eth_range1)

# mod2
e_mod2_r2 <- round(summary(eth_mod2)$r.squared,3)
e_mod2_adj.r2 <- round(summary(eth_mod2)$adj.r.squared,3)
e_N2 <- nrow(eth_range2)

# Now calculate models for each range using robust standard errors
# clustered by exchange - trading pair
b_mod_robust <- coeftest(btc_mod1, vcov = vcovHC(btc_mod1, type="HC1"),
                         cluster = ~identitfier)
b_mod_robust2 <- coeftest(btc_mod2, vcov = vcovHC(btc_mod2, type="HC1"),
                          cluster = ~identitfier)

e_mod_robust <- coeftest(eth_mod1, vcov = vcovHC(eth_mod1, type="HC1"),
                         cluster = ~identitfier)
e_mod_robust2 <- coeftest(eth_mod2, vcov = vcovHC(eth_mod2, type="HC1"),
                          cluster = ~identitfier)

# Now use stargazer to create a Latex table using this information
stargazer(b_mod_robust,b_mod_robust2,
          e_mod_robust,e_mod_robust2,
          type = "text",
          omit=c("m_d","id_2"),
          star.cutoffs = c(0.05, 0.01, 0.001),
          add.lines = list(c("N",b_N1,b_N2,e_N1,e_N2),
                           c("R squared", b_mod1_r2, b_mod2_r2,
                             e_mod1_r2, e_mod2_r2),
                           c("Adjusted R squared", b_mod1_adj.r2,
                             b_mod2_adj.r2, e_mod1_adj.r2, e_mod2_adj.r2)))
```

```
##
## ================================================
##                          Dependent variable:
##                     --------------------------
##
##                       (1)      (2)      (3)     (4)
## ------------------------------------------------
## treat                -0.010  0.182*** -0.004  -0.026*
##                     (0.006)  (0.026)  (0.004) (0.011)
##
## time                 -0.012   -0.011  -0.009  -0.023
##                     (0.010)  (0.036)  (0.007) (0.017)
##
## treat:time          0.018*** 0.076*** 0.011** 0.028**
##                     (0.005)  (0.019)  (0.003) (0.010)
##
## Constant             0.014    -0.013  0.017*  0.034*
##                     (0.009)  (0.030)  (0.007) (0.017)
##
## ------------------------------------------------
## N                     2040     408     1275     255
## R squared             0.56    0.618    0.355   0.51
## Adjusted R squared    0.54    0.555    0.314   0.375
```

```
## ================================================
## Note:                  *p<0.05; **p<0.01; ***p<0.001
```

```
################################################################################
### TABLE 10
################################################################################

# To avoid confusion, remove all objects created for previous table
# Define a list of prefixes you want to remove
prefixes_to_remove <- c("p.btc","p.eth",
                        "b_","e_",
                        "btc_","eth_",
                        "lower_","upper_")

# Create a pattern that matches any of these prefixes at the beginning of object names
pattern <- paste0("^(", paste(prefixes_to_remove, collapse="|"), ")")

# Remove objects matching the pattern
rm(list=ls(pattern=pattern))

# Prep the data using the function
lower_bound = 249
upper_bound = 750
p.btc <- process_trading_data(df.btc, lower_bound = lower_bound, upper_bound = upper_bound)
p.eth <- process_trading_data(df.eth, lower_bound = lower_bound, upper_bound = upper_bound)

## Create two ranges of 50 and 10 dollars below the threshold for each
btc_range1 <- filter(p.btc, dollar_bins > 449 & dollar_bins < 500)
btc_range2 <- filter(p.btc, dollar_bins > 489 & dollar_bins < 500)

eth_range1 <- filter(p.eth, dollar_bins > 449 & dollar_bins < 500)
eth_range2 <- filter(p.eth, dollar_bins > 489 & dollar_bins < 500)

# Create difference-in-differences models for each range
btc_mod1 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),btc_range1)
btc_mod2 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),btc_range2)

eth_mod1 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),eth_range1)
eth_mod2 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),eth_range2)

## Extract R^2, adjusted R^2, and N for each model
# mod1
b_mod1_r2 <- round(summary(btc_mod1)$r.squared,3)
b_mod1_adj.r2 <- round(summary(btc_mod1)$adj.r.squared,3)
b_N1 <- nrow(btc_range1)

# mod2
b_mod2_r2 <- round(summary(btc_mod2)$r.squared,3)
b_mod2_adj.r2 <- round(summary(btc_mod2)$adj.r.squared,3)
b_N2 <- nrow(btc_range2)

# mod1
e_mod1_r2 <- round(summary(eth_mod1)$r.squared,3)
e_mod1_adj.r2 <- round(summary(eth_mod1)$adj.r.squared,3)
```

```r
e_N1 <- nrow(eth_range1)

# mod2
e_mod2_r2 <- round(summary(eth_mod2)$r.squared,3)
e_mod2_adj.r2 <- round(summary(eth_mod2)$adj.r.squared,3)
e_N2 <- nrow(eth_range2)

# Now calculate models for each range using robust standard errors
# clustered by exchange - trading pair
b_mod_robust <- coeftest(btc_mod1, vcov = vcovHC(btc_mod1, type="HC1"),
                         cluster = ~identitfier)
b_mod_robust2 <- coeftest(btc_mod2, vcov = vcovHC(btc_mod2, type="HC1"),
                          cluster = ~identitfier)

e_mod_robust <- coeftest(eth_mod1, vcov = vcovHC(eth_mod1, type="HC1"),
                         cluster = ~identitfier)
e_mod_robust2 <- coeftest(eth_mod2, vcov = vcovHC(eth_mod2, type="HC1"),
                          cluster = ~identitfier)

# Now use stargazer to create a Latex table using this information
stargazer(b_mod_robust,b_mod_robust2,
          e_mod_robust,e_mod_robust2,
          type="text",
          omit=c("m_d","id_2"),
          star.cutoffs = c(0.05, 0.01, 0.001),
          add.lines = list(c("N",b_N1,b_N2,e_N1,e_N2),
                           c("R squared", b_mod1_r2, b_mod2_r2, e_mod1_r2,
                             e_mod2_r2),
                           c("Adjusted R squared", b_mod1_adj.r2, b_mod2_adj.r2,
                             e_mod1_adj.r2, e_mod2_adj.r2)))
```

```
##
## ================================================
##                          Dependent variable:
##                     --------------------------
##
##                       (1)      (2)     (3)      (4)
## ------------------------------------------------
## treat              0.016*** 0.022** 0.015*** 0.013**
##                    (0.005)  (0.007) (0.004)  (0.005)
##
## time               -0.015**  0.002  -0.014** -0.001
##                    (0.005)  (0.012) (0.005)  (0.006)
##
## treat:time          -0.007  -0.011   -0.004  -0.001
##                    (0.004)  (0.007) (0.004)  (0.005)
##
## Constant           0.018*** 0.0003  0.019***  0.007
##                    (0.005)  (0.008) (0.004)  (0.004)
##
## ------------------------------------------------
## N                    2040     408     1275     255
## R squared           0.229    0.534   0.283    0.358
## Adjusted R squared  0.194    0.457   0.238    0.18
```

```
## ================================================
## Note:                  *p<0.05; **p<0.01; ***p<0.001
```

```
################################################################################
### TABLE 11
################################################################################

# To avoid confusion, remove all objects created for previous table
# Define a list of prefixes you want to remove
prefixes_to_remove <- c("p.btc","p.eth",
                        "b_","e_",
                        "btc_","eth_",
                        "lower_","upper_")

# Create a pattern that matches any of these prefixes at the beginning of object names
pattern <- paste0("^(", paste(prefixes_to_remove, collapse="|"), ")")

# Remove objects matching the pattern
rm(list=ls(pattern=pattern))

# Prep the data using the function
lower_bound = 1249
upper_bound = 1750
p.btc <- process_trading_data(df.btc, lower_bound = lower_bound, upper_bound = upper_bound)
p.eth <- process_trading_data(df.eth, lower_bound = lower_bound, upper_bound = upper_bound)

## Create two ranges of 50 and 10 dollars below the threshold for each
btc_range1 <- filter(p.btc, dollar_bins > 1449 & dollar_bins < 1500)
btc_range2 <- filter(p.btc, dollar_bins > 1489 & dollar_bins < 1500)

eth_range1 <- filter(p.eth, dollar_bins > 1449 & dollar_bins < 1500)
eth_range2 <- filter(p.eth, dollar_bins > 1489 & dollar_bins < 1500)

# Create difference-in-differences models for each range
btc_mod1 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),btc_range1)
btc_mod2 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),btc_range2)

eth_mod1 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),eth_range1)
eth_mod2 <- lm(prop_trades ~ treat*time + factor(id_2) + factor(m_d),eth_range2)

## Extract R^2, adjusted R^2, and N for each model
# mod1
b_mod1_r2 <- round(summary(btc_mod1)$r.squared,3)
b_mod1_adj.r2 <- round(summary(btc_mod1)$adj.r.squared,3)
b_N1 <- nrow(btc_range1)

# mod2
b_mod2_r2 <- round(summary(btc_mod2)$r.squared,3)
b_mod2_adj.r2 <- round(summary(btc_mod2)$adj.r.squared,3)
b_N2 <- nrow(btc_range2)

# mod1
e_mod1_r2 <- round(summary(eth_mod1)$r.squared,3)
e_mod1_adj.r2 <- round(summary(eth_mod1)$adj.r.squared,3)
```

```r
e_N1 <- nrow(eth_range1)

# mod2
e_mod2_r2 <- round(summary(eth_mod2)$r.squared,3)
e_mod2_adj.r2 <- round(summary(eth_mod2)$adj.r.squared,3)
e_N2 <- nrow(eth_range2)

# Now calculate models for each range using robust standard errors
# clustered by exchange - trading pair
b_mod_robust <- coeftest(btc_mod1, vcov = vcovHC(btc_mod1, type="HC1"),
                         cluster = ~identitfier)
b_mod_robust2 <- coeftest(btc_mod2, vcov = vcovHC(btc_mod2, type="HC1"),
                          cluster = ~identitfier)

e_mod_robust <- coeftest(eth_mod1, vcov = vcovHC(eth_mod1, type="HC1"),
                         cluster = ~identitfier)
e_mod_robust2 <- coeftest(eth_mod2, vcov = vcovHC(eth_mod2, type="HC1"),
                          cluster = ~identitfier)

# Now use stargazer to create a Latex table using this information
stargazer(b_mod_robust,b_mod_robust2,
          e_mod_robust,e_mod_robust2,
          type = "text",
          omit=c("m_d","id_2"),
          star.cutoffs = c(0.05, 0.01, 0.001),
          add.lines = list(c("N",b_N1,b_N2,e_N1,e_N2),
                           c("R squared", b_mod1_r2, b_mod2_r2,
                             e_mod1_r2, e_mod2_r2),
                           c("Adjusted R squared", b_mod1_adj.r2, b_mod2_adj.r2,
                             e_mod1_adj.r2, e_mod2_adj.r2)))
```

```
##
## ================================================
##                          Dependent variable:
##                     ----------------------------
##
##                       (1)      (2)     (3)     (4)
## ------------------------------------------------
## treat                 0.004   0.030*   0.003  0.015**
##                      (0.007)  (0.013) (0.010) (0.005)
##
## time                  -0.004  0.048*  -0.006   0.051
##                      (0.008)  (0.023) (0.013) (0.051)
##
## treat:time            0.006   -0.004   0.010  -0.006
##                      (0.004)  (0.011) (0.011) (0.006)
##
## Constant              0.013*  -0.008  0.015*  -0.0003
##                      (0.006)  (0.008) (0.007) (0.006)
##
## ------------------------------------------------
## N                      2040     408    1275     255
## R squared             0.122    0.31    0.092   0.247
## Adjusted R squared    0.081    0.195   0.035   0.039
```

```
## ================================================
## Note:                  *p<0.05; **p<0.01; ***p<0.001
```

```
################################################################################
### TABLE 12
################################################################################

## Part 1: Data preparation

# To avoid confusion, remove all objects created for previous table
# Define a list of prefixes you want to remove
prefixes_to_remove <- c("p.btc","p.eth",
                        "b_","e_",
                        "btc_","eth_",
                        "lower_","upper_")

# Create a pattern that matches any of these prefixes at the beginning of object names
pattern <- paste0("^(", paste(prefixes_to_remove, collapse="|"), ")")

# Remove objects matching the pattern
rm(list=ls(pattern=pattern))

## Separate by before and after regulatory change
df.btc_pre <- df.btc %>%
  filter(m_d < "07:10",
         type == "regulated")
df.btc_post <- df.btc %>%
  filter(m_d >= "07:10",
         type == "regulated")

df.eth_pre <- df.eth %>%
  filter(m_d < "07:10",
         type == "regulated")
df.eth_post <- df.eth %>%
  filter(m_d >= "07:10",
         type == "regulated")

## 2. Bitcoin analysis

## A. Pre

b_range1 = df.btc_pre[(df.btc_pre$base >= 250 & df.btc_pre$base < 750),]
b_range2 = df.btc_pre[(df.btc_pre$base >= 750 & df.btc_pre$base < 1250),]
b_range3 = df.btc_pre[(df.btc_pre$base >= 1250 & df.btc_pre$base < 1750),]

##Get estimates
b_r1 = estimate_beta(df.btc_pre, z_vector, binv, zstar=490, bins_excl_l=9, bins_excl_r=0)
b_r2 = estimate_beta(df.btc_pre, z_vector, binv, zstar=990, bins_excl_l=9, bins_excl_r=0)
b_r3 = estimate_beta(df.btc_pre, z_vector, binv, zstar=1490, bins_excl_l=9, bins_excl_r=0)

b_results = rbind.data.frame(b_r1,b_r2,b_r3)

b_results$t_statistic = abs(b_results$beta)/b_results$`standard error`
b_results[,c(2:4)] = sapply(b_results[,c(2:4)],rounding)
```

Table 1: Bitcoin Pre-Treatment Results

| threshold | beta | standard error | t_statistic | count_transactions |
|---|---|---|---|---|
| 490 | 5.051 | 2.326 | 2.172 | 15149 |
| 990 | 3.680 | 0.480 | 7.667 | 20903 |
| 1490 | -4.661 | 0.571 | 8.162 | 5948 |

[a] N exchanges = 2: N trading pairs = 4.

```
b_transactions = c(count(b_range1),count(b_range2),count(b_range3))
b_transactions = unlist(b_transactions)
b_results$count_transactions = b_transactions

# Total exchanges and trading pairs
b_N_exchanges <- sum(length(unique(df.btc_pre$exchange)))
b_N_trading_pairs <- sum(length(unique(paste0(df.btc_pre$exchange,df.btc_pre$s))))

# Create a footnote showing the number of exchanges and trading pairs
my_footnote1b <- paste0("N exchanges = ",b_N_exchanges,": N trading pairs = ",b_N_trading_pairs,".")

# Display Table
kable(b_results, caption = "Bitcoin Pre-Treatment Results", format = "latex", booktabs = T) %>%
  kable_styling() %>%
  add_footnote(my_footnote1b)
```

```
## B. Post

b2_range1 = df.btc_post[(df.btc_post$base >= 250 & df.btc_post$base < 750),]
b2_rangb2 = df.btc_post[(df.btc_post$base >= 750 & df.btc_post$base < 1250),]
b2_range3 = df.btc_post[(df.btc_post$base >= 1250 & df.btc_post$base < 1750),]

##Get estimates
b2_r1 = estimate_beta(df.btc_post, z_vector, binv, zstar=490, bins_excl_l=9, bins_excl_r=0)
b2_r2 = estimate_beta(df.btc_post, z_vector, binv, zstar=990, bins_excl_l=9, bins_excl_r=0)
b2_r3 = estimate_beta(df.btc_post, z_vector, binv, zstar=1490, bins_excl_l=9, bins_excl_r=0)

b2_results = rbind.data.frame(b2_r1,b2_r2,b2_r3)

b2_results$t_statistic = abs(b2_results$beta)/b2_results$`standard error`
b2_results[,c(2:4)] = sapply(b2_results[,c(2:4)],rounding)
b2_transactions = c(count(b2_range1),count(b2_rangb2),count(b2_range3))
b2_transactions = unlist(b2_transactions)
b2_results$count_transactions = b2_transactions

# Total exchanges and trading pairs
b2_N_exchanges <- sum(length(unique(df.btc_post$exchange)))
b2_N_trading_pairs <- sum(length(unique(paste0(df.btc_post$exchange,df.btc_post$s))))

# Create a footnote showing the number of exchanges and trading pairs
my_footnote2 <- paste0("N exchanges = ",b2_N_exchanges,": N trading pairs = ",b2_N_trading_pairs,".")

# Display Table
kable(b2_results, caption = "Bitcoin Post-Treatment Results", format = "latex", booktabs = T) %>%
```

Table 2: Bitcoin Post-Treatment Results

| threshold | beta | standard error | t_statistic | count_transactions |
|---|---|---|---|---|
| 490 | -0.088 | 2.040 | 0.043 | 101523 |
| 990 | 3.146 | 0.945 | 3.328 | 116790 |
| 1490 | -0.511 | 0.366 | 1.398 | 55521 |

[a] N exchanges = 2: N trading pairs = 4.

```
  kable_styling() %>%
  add_footnote(my_footnote2)
```

```
## 3. Ethereum analysis

## A. Pre

e_range1 = df.eth_pre[(df.eth_pre$base >= 250 & df.eth_pre$base < 750),]
e_range2 = df.eth_pre[(df.eth_pre$base >= 750 & df.eth_pre$base < 1250),]
e_range3 = df.eth_pre[(df.eth_pre$base >= 1250 & df.eth_pre$base < 1750),]

##Get estimates
e_r1 = estimate_beta(df.eth_pre, z_vector, binv, zstar=490, bins_excl_l=9, bins_excl_r=0)
e_r2 = estimate_beta(df.eth_pre, z_vector, binv, zstar=990, bins_excl_l=9, bins_excl_r=0)
e_r3 = estimate_beta(df.eth_pre, z_vector, binv, zstar=1490, bins_excl_l=9, bins_excl_r=0)

e_results = rbind.data.frame(e_r1,e_r2,e_r3)

e_results$t_statistic = abs(e_results$beta)/e_results$`standard error`
e_results[,c(2:4)] = sapply(e_results[,c(2:4)],rounding)
e_transactions = c(count(e_range1),count(e_range2),count(e_range3))
e_transactions = unlist(e_transactions)
e_results$count_transactions = e_transactions

# Total exchanges and trading pairs
e_N_exchanges <- sum(length(unique(df.eth_pre$exchange)))
e_N_trading_pairs <- sum(length(unique(paste0(df.eth_pre$exchange,df.eth_pre$s))))

# Create a footnote showing the number of exchanges and trading pairs
my_footnote1 <- paste0("N exchanges = ",e_N_exchanges,": N trading pairs = ",e_N_trading_pairs,".")

# Display Table
kable(e_results, caption = "Ethereum Pre-Treatment Results", format = "latex", booktabs = T) %>%
  kable_styling() %>%
  add_footnote(my_footnote1)
```

```
## B. Post

e2_range1 = df.eth_post[(df.eth_post$base >= 250 & df.eth_post$base < 750),]
e2_range2 = df.eth_post[(df.eth_post$base >= 750 & df.eth_post$base < 1250),]
e2_range3 = df.eth_post[(df.eth_post$base >= 1250 & df.eth_post$base < 1750),]

##Get estimates
e2_r1 = estimate_beta(df.eth_post, z_vector, binv, zstar=490, bins_excl_l=9, bins_excl_r=0)
```

Table 3: Ethereum Pre-Treatment Results

| threshold | beta | standard error | t_statistic | count_transactions |
|---|---|---|---|---|
| 490 | 5.684 | 1.751 | 3.247 | 16635 |
| 990 | 3.524 | 1.771 | 1.989 | 4189 |
| 1490 | 3.235 | 2.014 | 1.606 | 1609 |

[a] N exchanges = 2: N trading pairs = 2.

Table 4: Ethereum EUR Results

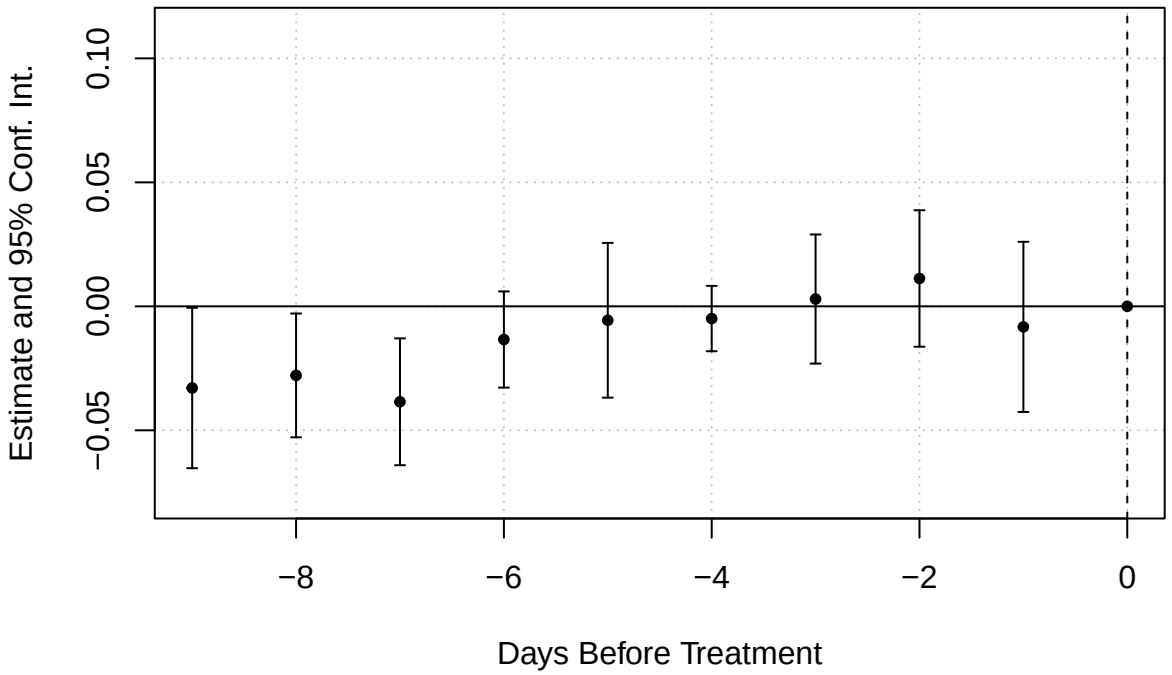
| threshold | beta | standard error | t_statistic | count_transactions |
|---|---|---|---|---|
| 490 | -0.383 | 0.782 | 0.490 | 107025 |
| 990 | 2.671 | 0.645 | 4.139 | 56844 |
| 1490 | -0.454 | 1.282 | 0.354 | 20489 |

[a] N exchanges = 2: N trading pairs = 2.

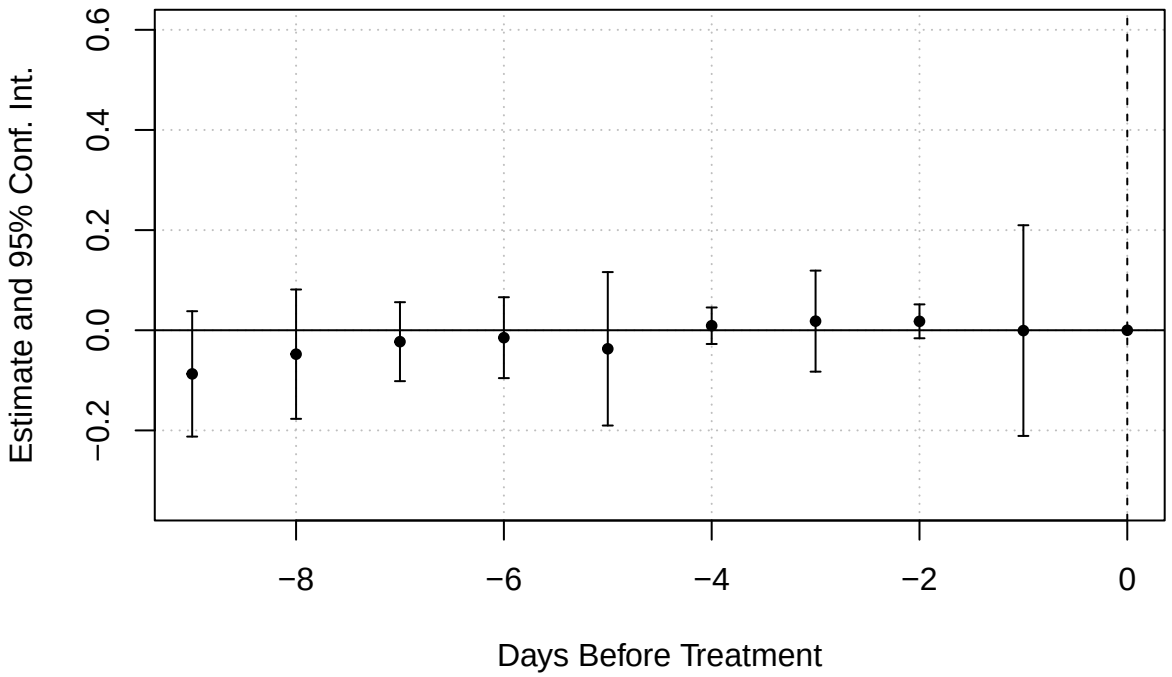
```
e2_r2 = estimate_beta(df.eth_post, z_vector, binv, zstar=990, bins_excl_l=9, bins_excl_r=0)
e2_r3 = estimate_beta(df.eth_post, z_vector, binv, zstar=1490, bins_excl_l=9, bins_excl_r=0)

e2_results = rbind.data.frame(e2_r1,e2_r2,e2_r3)

e2_results$t_statistic = abs(e2_results$beta)/e2_results$`standard error`
e2_results[,c(2:4)] = sapply(e2_results[,c(2:4)],rounding)
e2_transactions = c(count(e2_range1),count(e2_range2),count(e2_range3))
e2_transactions = unlist(e2_transactions)
e2_results$count_transactions = e2_transactions

# Total exchanges and trading pairs
e2_N_exchanges <- sum(length(unique(df.eth_post$exchange)))
e2_N_trading_pairs <- sum(length(unique(paste0(df.eth_post$exchange,df.eth_post$s))))

# Create a footnote showing the number of exchanges and trading pairs
my_footnote2 <- paste0("N exchanges = ",e2_N_exchanges,": N trading pairs = ",e2_N_trading_pairs,".")

# Display Table
kable(e2_results, caption = "Ethereum EUR Results", format = "latex", booktabs = T) %>%
  kable_styling() %>%
  add_footnote(my_footnote2)
```

```
################################################################################
### FIGURES 3 and 4
################################################################################

## Part 1: Data preparation

# To avoid confusion, remove all objects created for previous table
# Define a list of prefixes you want to remove
prefixes_to_remove <- c("p.btc","p.eth",
                        "b_","e_",
                        "b2_","e2_",
```

```
                        "btc_","eth_",
                        "my_",
                        "lower_","upper_")

# Create a pattern that matches any of these prefixes at the beginning of object names
pattern <- paste0("^(", paste(prefixes_to_remove, collapse="|"), ")")

# Remove objects matching the pattern
rm(list=ls(pattern=pattern))

# Prep the data using the function
lower_bound = 749
upper_bound = 1250
p.btc <- process_trading_data(df.btc, lower_bound = lower_bound, upper_bound = upper_bound)
p.eth <- process_trading_data(df.eth, lower_bound = lower_bound, upper_bound = upper_bound)


## Part 2: Bitcoin pre-trend plots

## A. 50 units below threshold
sub1 <- p.btc %>%
  filter(dollar_bins >= 950 & dollar_bins < 1000)


# Adjust time
sub1$time <- as.numeric(as.factor(sub1$m_d)) - 10
sub1$time_treat <- ifelse(sub1$time > -1,1,0)


# Create two-way fixed effects model
twfe_results <- fixest::feols(prop_trades ~ i(time, treat, ref = 0) | id_2 + time,
                              cluster = "id_2",
                              data = sub1)

# Generate plot
fixest::iplot(twfe_results,
              xlab="Days Before Treatment",
              main="",
              xlim=c(-9,0))
```

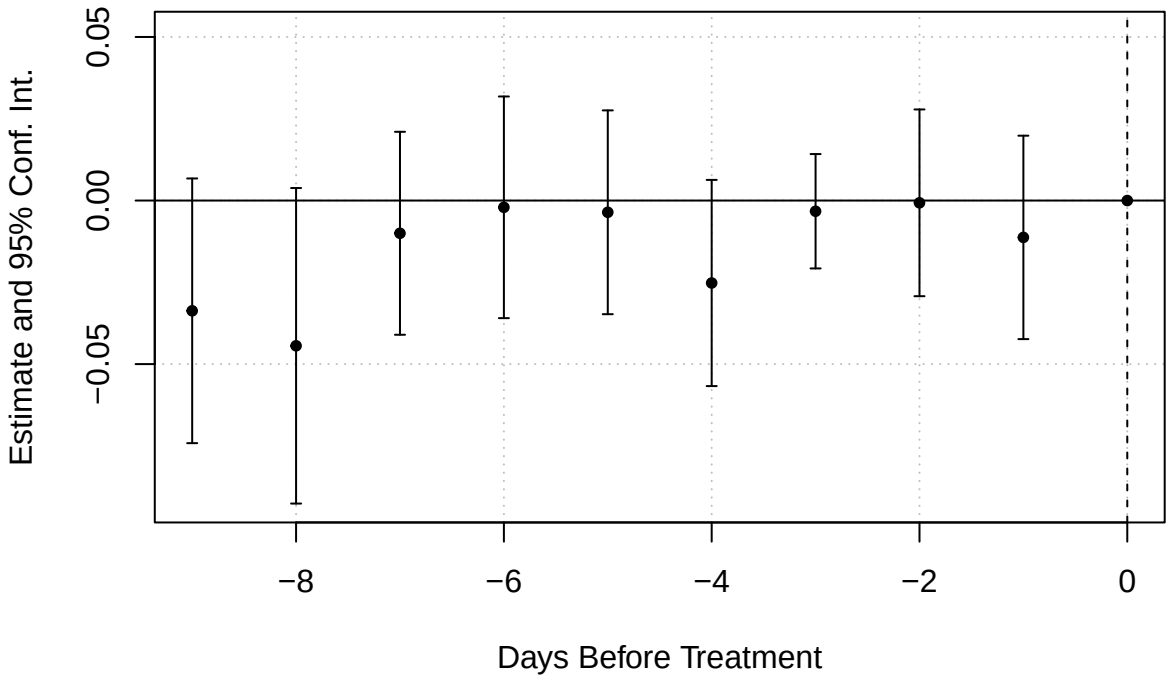



```
## B. 10 units below threshold

sub2 <- p.btc %>%
  filter(dollar_bins >= 990 & dollar_bins < 1000)

sub2$time <- as.numeric(as.factor(sub2$m_d)) - 10
sub2$time_treat <- ifelse(sub2$time > -1,1,0)

# Create two-way fixed effects model
twfe_results.2 <- fixest::feols(prop_trades ~ i(time, treat, ref = 0) | id_2 + time,
                                cluster = "id_2",
                                data = sub2)

# Generate plot
fixest::iplot(twfe_results.2,
              xlab="Days Before Treatment",
              main="",
              xlim=c(-9,0))
```

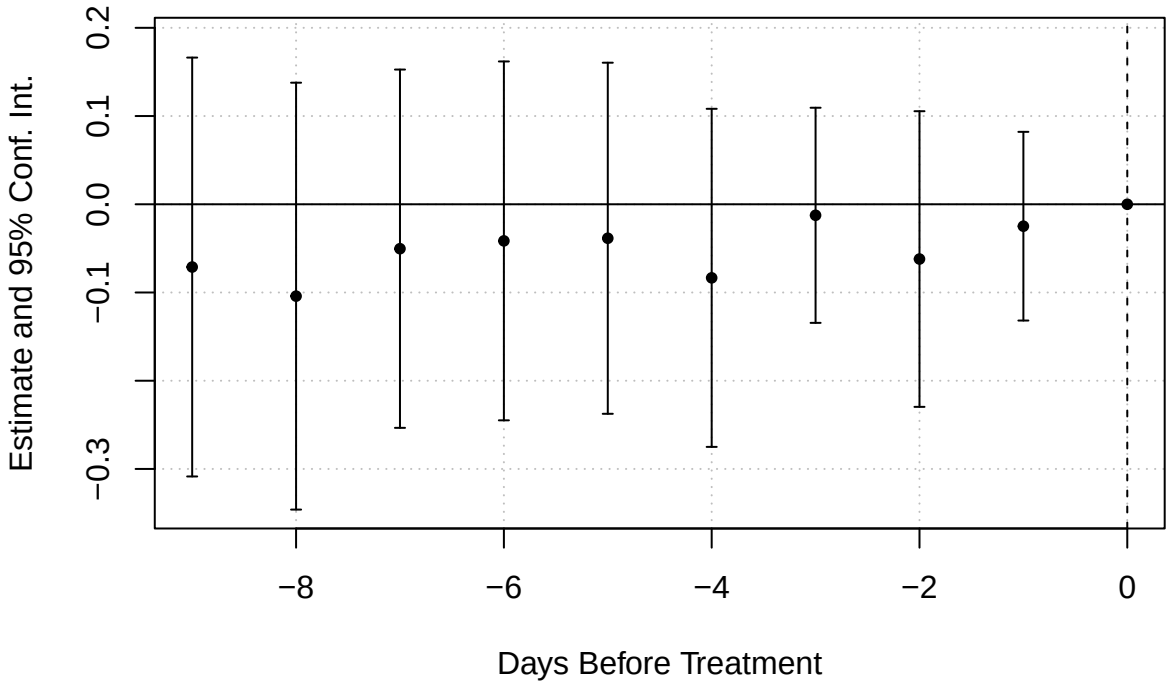



```
## Part 3: Ethereum pre-trend plots

# To avoid confusion, remove all objects created for previous plots
# Define a list of prefixes you want to remove
prefixes_to_remove <- c("sub","twfe_")

# Create a pattern that matches any of these prefixes at the beginning of object names
pattern <- paste0("^(", paste(prefixes_to_remove, collapse="|"), ")")

# Remove objects matching the pattern
rm(list=ls(pattern=pattern))

## A. 50 units below threshold

## First, do it for the 50 below the threshold
sub1 <- p.eth %>%
  filter(dollar_bins >= 950 & dollar_bins < 1000)


# Adjust time
sub1$time <- as.numeric(as.factor(sub1$m_d)) - 10
sub1$time_treat <- ifelse(sub1$time > -1,1,0)

# Create two-way fixed effects model
twfe_results <- fixest::feols(prop_trades ~ i(time, treat, ref = 0) | id_2 + time,
                              cluster = "id_2",
                              data = sub1)


fixest::iplot(twfe_results,
              xlab="Days Before Treatment",
              main="",
```

```
              xlim=c(-9,0))
```




```
## B. 10 units below threshold

sub2 <- p.eth %>%
  filter(dollar_bins >= 990 & dollar_bins < 1000)

sub2$time <- as.numeric(as.factor(sub2$m_d)) - 10
sub2$time_treat <- ifelse(sub2$time > -1,1,0)

# Create two-way fixed effects model
twfe_results.2 <- fixest::feols(prop_trades ~ i(time, treat, ref = 0) | id_2 + time,
                                cluster = "id_2",
                                data = sub2)

fixest::iplot(twfe_results.2,
              xlab="Days Before Treatment",
              main="",
              xlim=c(-9,0))
```


Estimate and 95% Conf. Int.
0.2
0.1
0.0
−0.1
−0.3
−8
−6
−4
−2
0
Days Before Treatment