Figures

Karen Nershi

2025-04-08

Figures

The following code generates output for Figure 2 in the main body of the paper and Figures 6-12 in the Supplementary Material. This output matches the code included in the file "Figures.R."

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

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

# Packages used in analysis
pkg = c("tidyverse","bunching","ggplot2","ggridges")

# 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] "bunching"  "BB"        "lubridate" "forcats"   "stringr"   "dplyr"
##  [7] "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
## [13] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [19] "base"
##
## [[3]]
##  [1] "bunching"  "BB"        "lubridate" "forcats"   "stringr"   "dplyr"
##  [7] "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
## [13] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [19] "base"
##
## [[4]]
##  [1] "ggridges"  "bunching"  "BB"        "lubridate" "forcats"   "stringr"
##  [7] "dplyr"     "purrr"     "readr"     "tidyr"     "tibble"    "ggplot2"
```

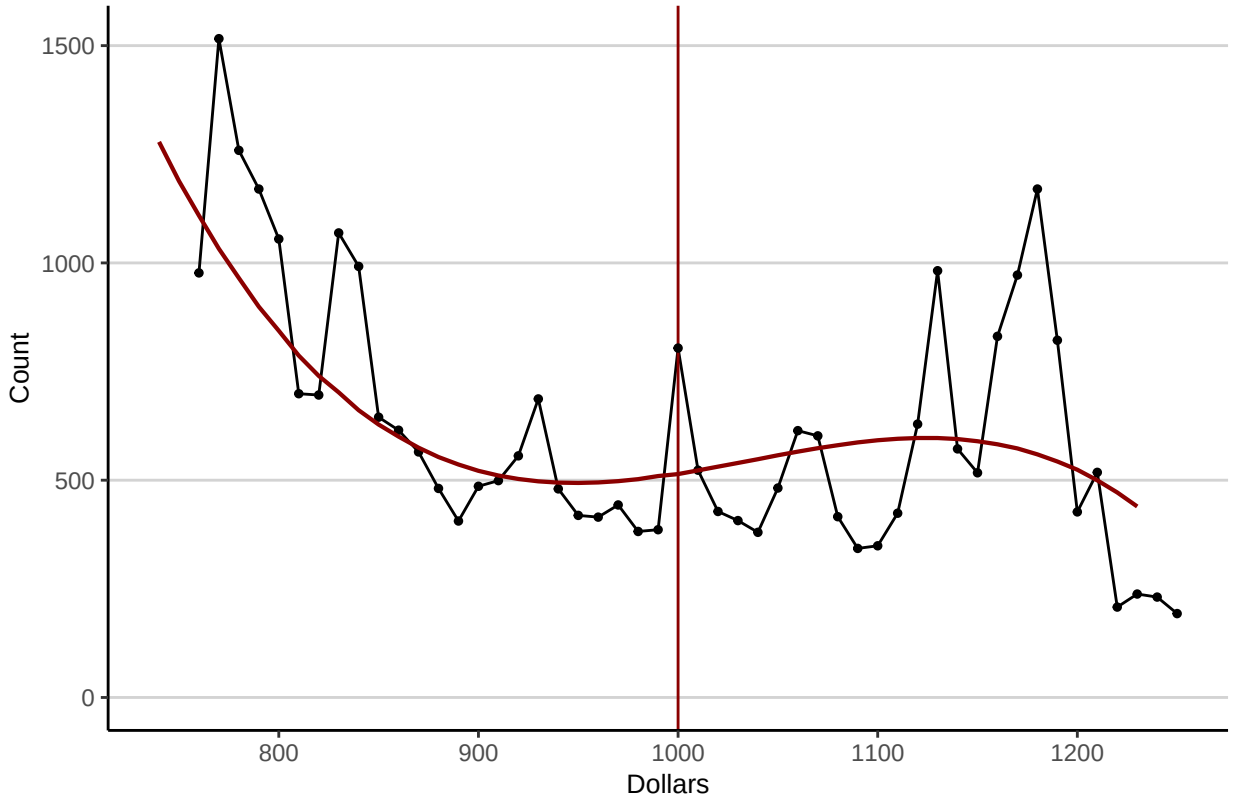
```
## [13] "tidyverse" "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [19] "methods"   "base"
```

```
rm(list=ls())

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

## Create a function and prepare data for graphing
data_for_graph = function(thedata) {
  z_vector = thedata$base

  # set parameters
  zstar = 990
  binwidth = 10 # 10 units for each bin
  bins_l = 25
  bins_r = 25
  poly = 3 # degree of polynomial

  bins_excl_l = 9
  bins_excl_r = 0

  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)
```

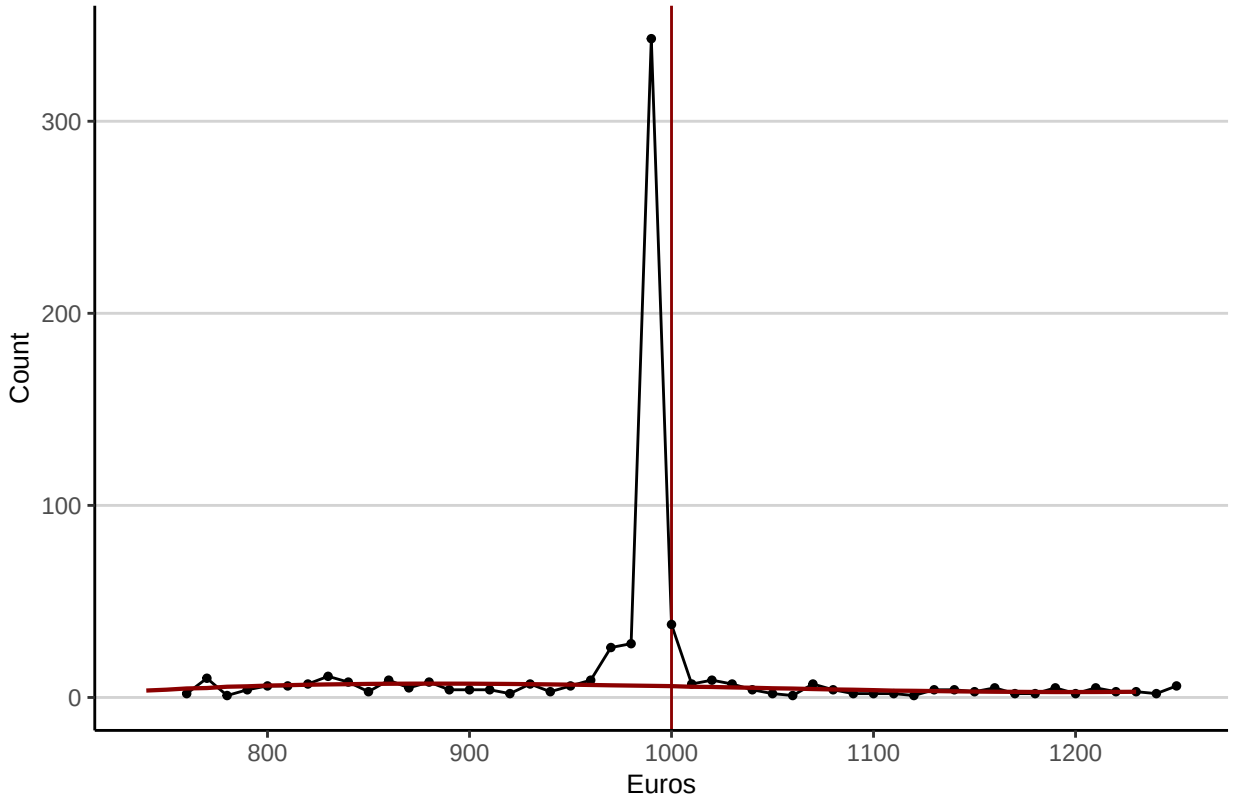
```
      data_binned[[bins_excl_l_varname]] <- ifelse(data_binned$z_rel ==
                                                     -i, 1, 0)
      bins_excl_l_vector <- c(bins_excl_l_vector, bins_excl_l_varname)
    }
    bins_excluded_all <- c(bins_excluded_all, bins_excl_l_vector)
  }
  if (bins_excl_r != 0) {
    bins_excl_r_vector <- c()
    for (i in seq(bins_excl_r)) {
      bin_excl_r_varname <- paste0("bin_excl_r_", i)
      data_binned[[bin_excl_r_varname]] <- ifelse(data_binned$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

  # 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"])
  }
  return(thedata)
}

################################################################################
```

```
################## PART 2: CREATE FIGURES ######################################
################################################################################

################################################################################
### FIGURE 2
################################################################################

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

## 1. Import and prepare data
df1 = read.csv("binanceus.csv")
df2 = read.csv("coinmetro.csv")

# apply function to data
thedata <- data_for_graph(df1)
thedata2 <- data_for_graph(df2)

## 2. Create graphs

# Binance US
graph_1 <- bunching::plot_hist(z_vector = df1$base, zstar = 1000, binv = "max",
                               binwidth = 10, bins_l = 25, bins_r = 25,
                               p_zstar_color = "darkred")$plot
graph_1 +
  geom_line(data=thedata,aes(x=bin,y=cf),color="darkred",size=0.75) +
  labs(x="Dollars")
```

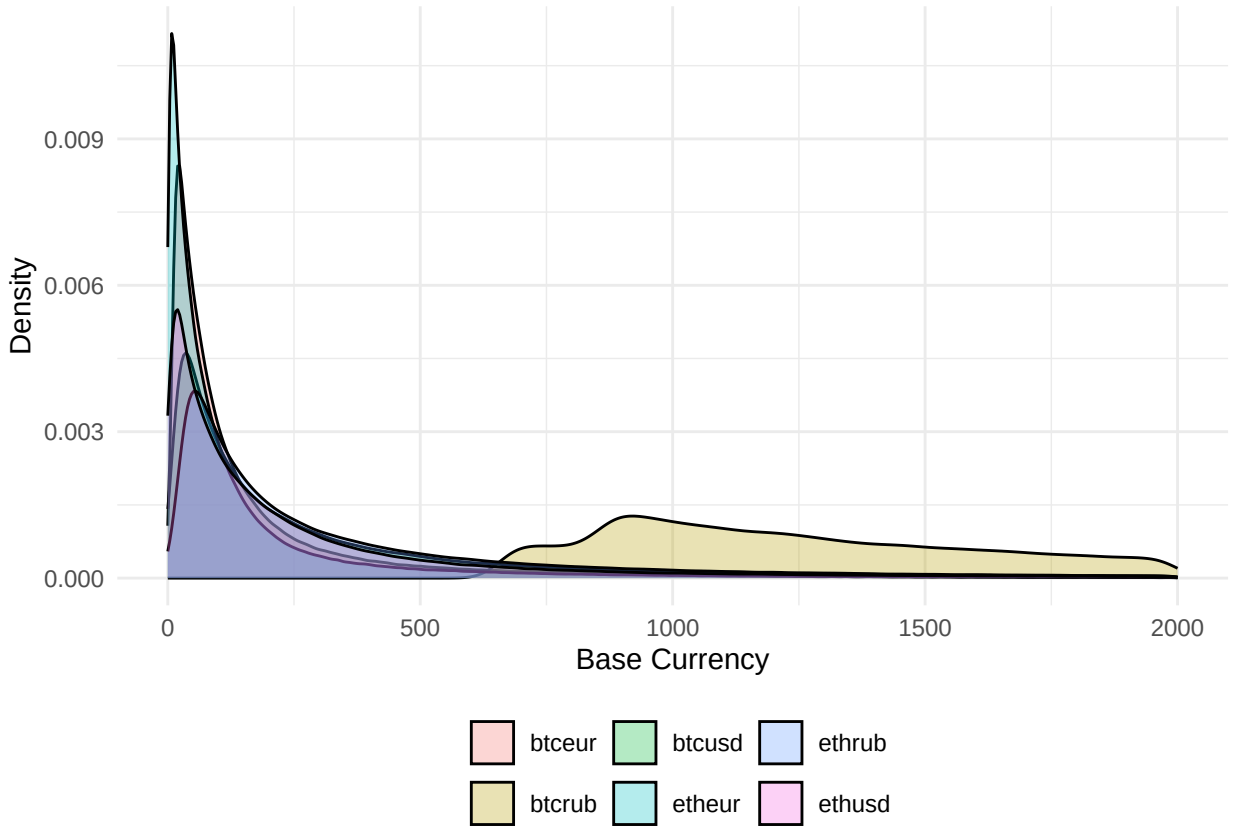
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



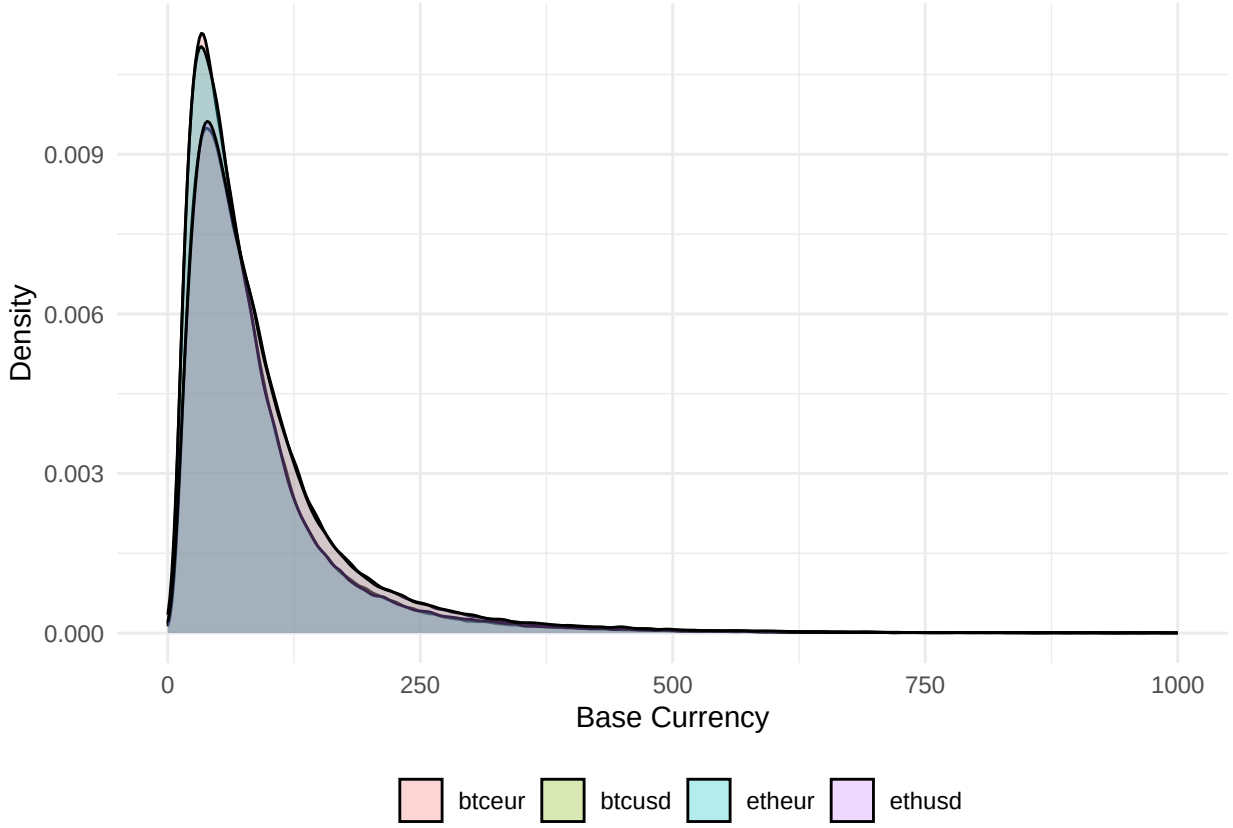

```
# Coinmetro
graph_2 <- bunching::plot_hist(z_vector = df2$base, zstar = 1000, binv = "max",
                               binwidth = 10, bins_l = 25, bins_r = 25,
                               p_zstar_color = "darkred")$plot
graph_2 +
  geom_line(data=thedata2,aes(x=bin,y=cf),color="darkred",size=0.75) +
  labs(x="Euros")
```



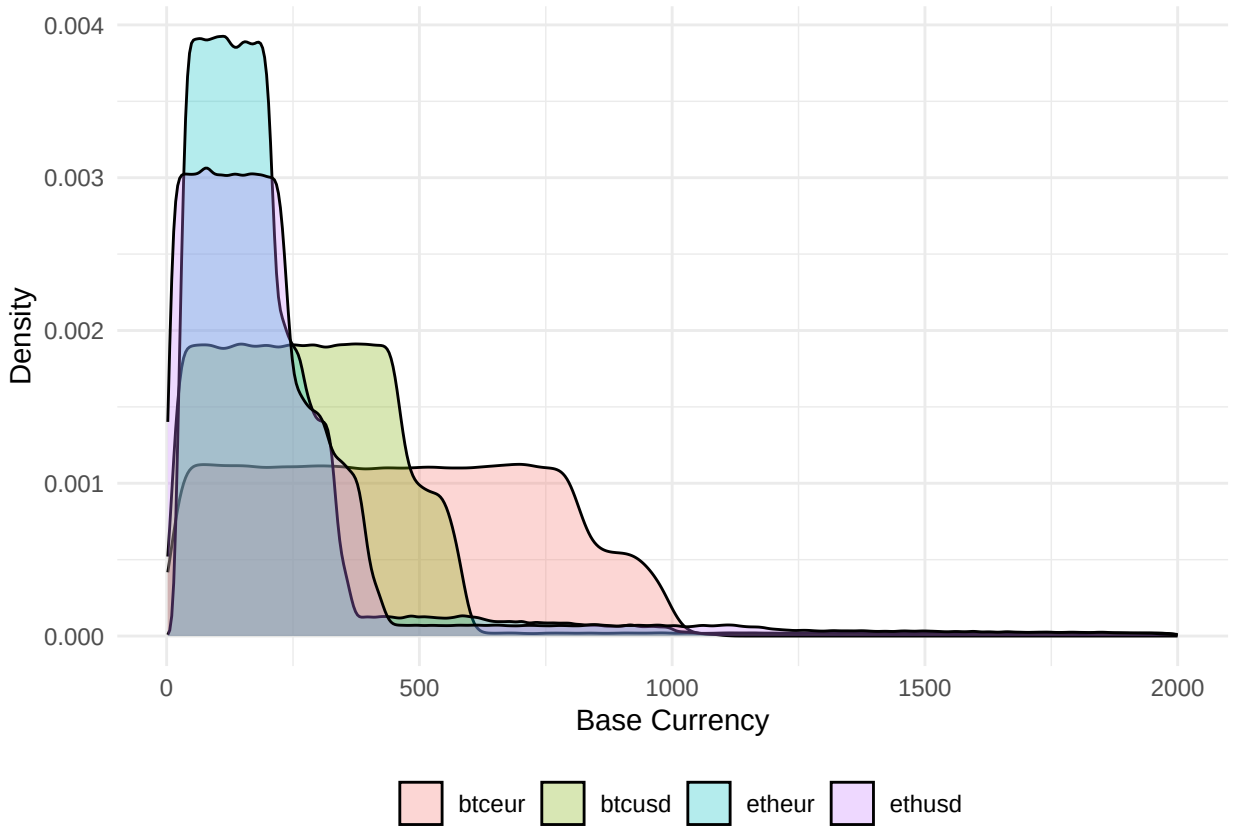

```
#################################################################################
### FIGURE 6
#################################################################################

## 1. Import data

coinsbit_df <- read.csv("ab_coinsbit.csv")

## 2. Graph

ggplot(coinsbit_df[(coinsbit_df$base < 2000),], aes(x=base)) + theme_minimal() +
  geom_density(aes(fill=s), alpha=0.3) +
  theme(legend.position='bottom') +
  labs(fill="",
       x="Base Currency",
       y="Density")
```



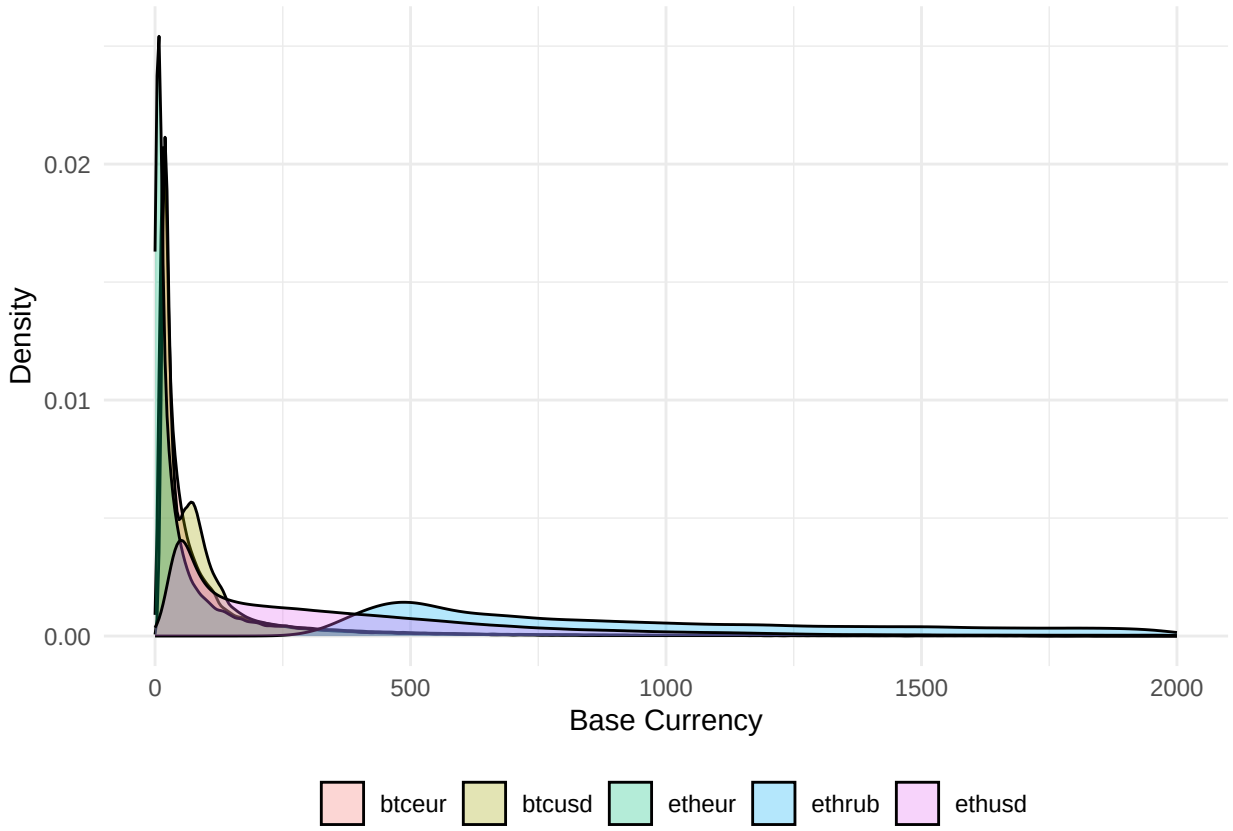

```
################################################################################
### FIGURE 7
################################################################################

## 1. Import data

cryptology_df <- read.csv("ab_cryptology.csv")

## 2. Graph

ggplot(cryptology_df[(cryptology_df$base < 1000),], aes(x=base)) + theme_minimal() +
  geom_density(aes(fill=s), alpha=0.3) +
  theme(legend.position='bottom') +
  labs(fill="",
       x="Base Currency",
       y="Density")
```



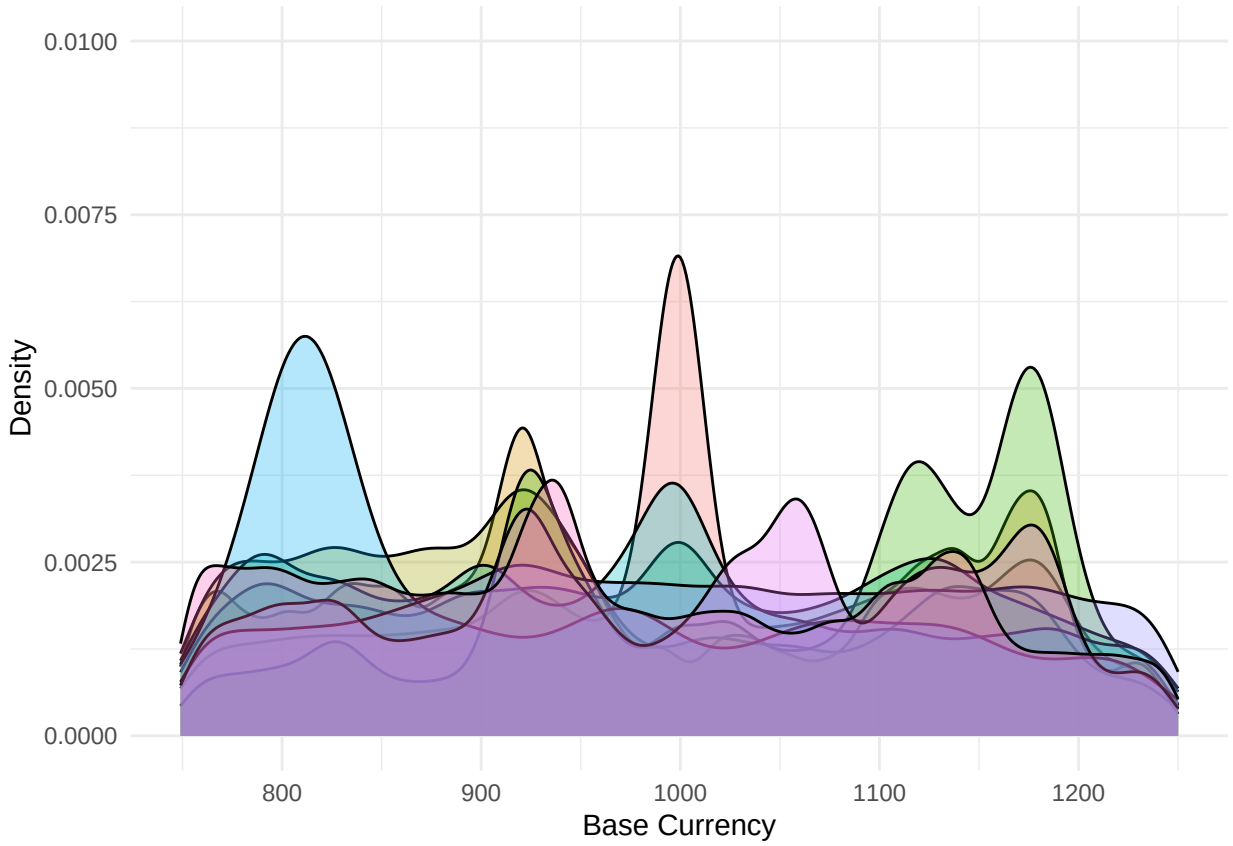

```
##############################################################################
### FIGURE 8
##############################################################################

## 1. Import data

folgory_df <- read.csv("ab_folgory.csv")

## 2. Graph

ggplot(folgory_df[(folgory_df$base < 2000),], aes(x=base)) + theme_minimal() +
  geom_density(aes(fill=s), alpha=0.3) +
  theme(legend.position='bottom') +
  labs(fill="",
       x="Base Currency",
       y="Density")
```



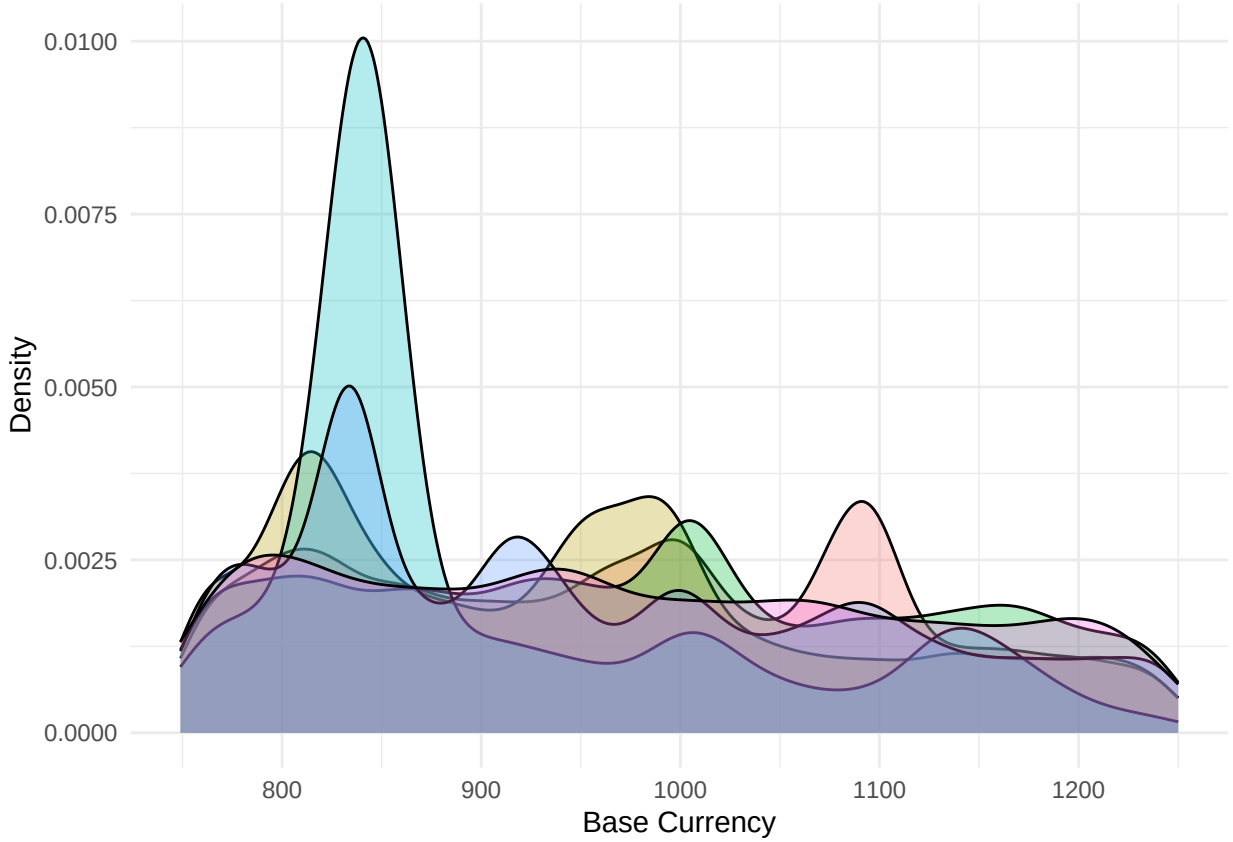

```
################################################################################
### FIGURE 9
################################################################################

## 1. Import data

whitebit_df <- read.csv("ab_whitebit.csv")

## 2. Graph

ggplot(whitebit_df[(whitebit_df$base < 2000),], aes(x=base)) + theme_minimal() +
  geom_density(aes(fill=s), alpha=0.3) +
  theme(legend.position='bottom') +
  labs(fill="",
       x="Base Currency",
       y="Density")
```



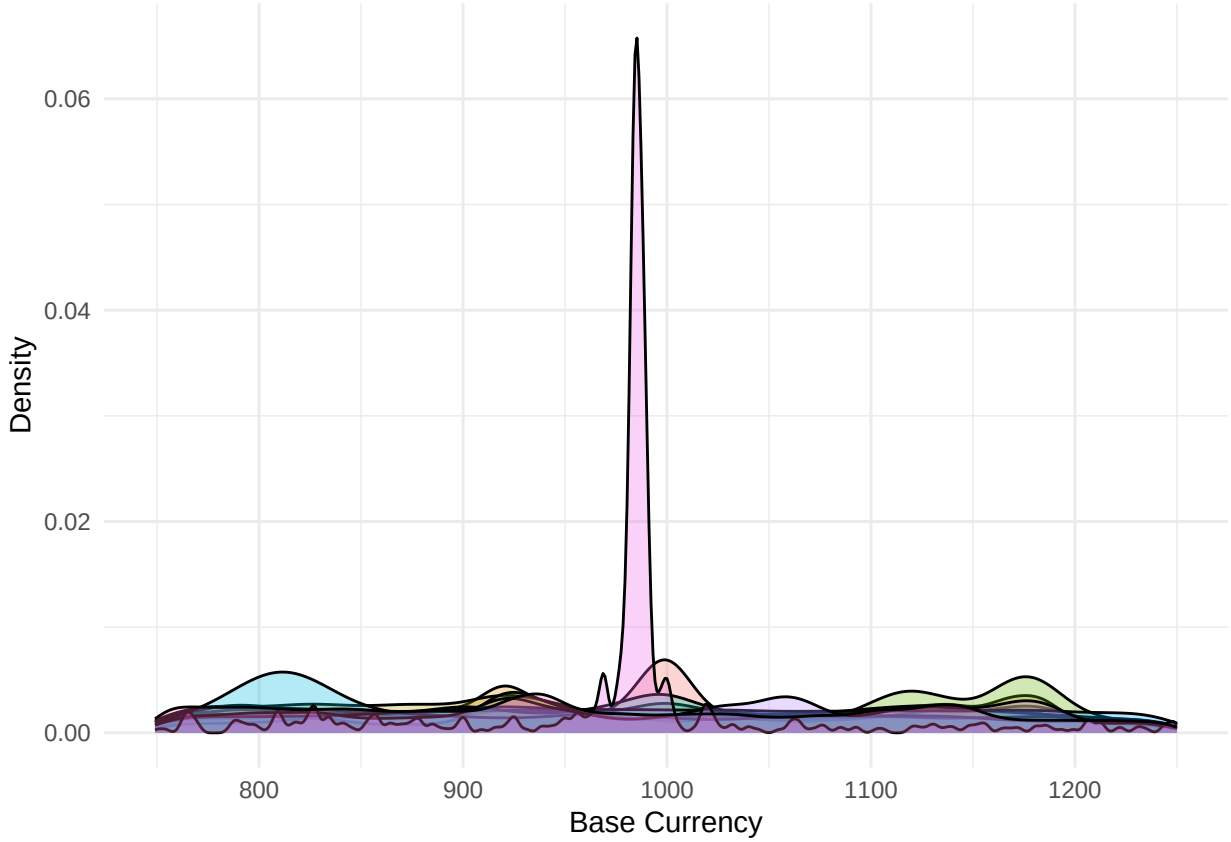

```
################################################################################
### FIGURE 10
################################################################################

## 1. Import data

reg_df <- read.csv("regulated_btc_transactions.csv")
unreg_df <- read.csv("unregulated_btc_transactions.csv")

## 2. Regulated exchanges

ggplot(reg_df[(reg_df$base > 749 & reg_df$base < 1250 & reg_df$exchange!="exchange_8"),], aes(x=base)) +
  geom_density(aes(fill=exchange), alpha=0.3) +
  theme(legend.position="none") +
  labs(x="Base Currency",
       y="Density") +
  ylim(min=0,max=0.01)
```



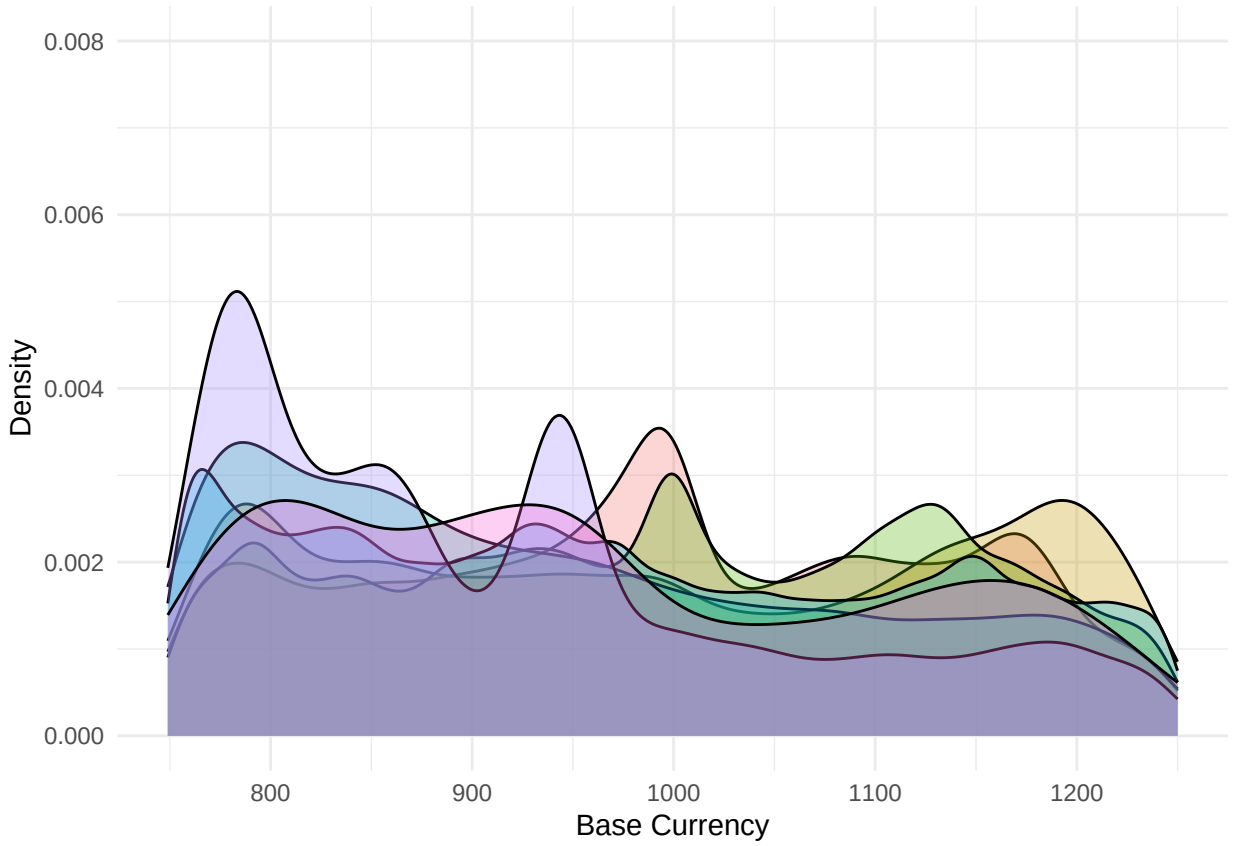

```
## 3. Unregulated exchanges

ggplot(unreg_df[(unreg_df$base > 749 & unreg_df$base < 1250),], aes(x=base)) + theme_minimal() +
  geom_density(aes(fill=exchange), alpha=0.3) +
  theme(legend.position="none") +
  labs(x="Base Currency",
       y="Density")
```



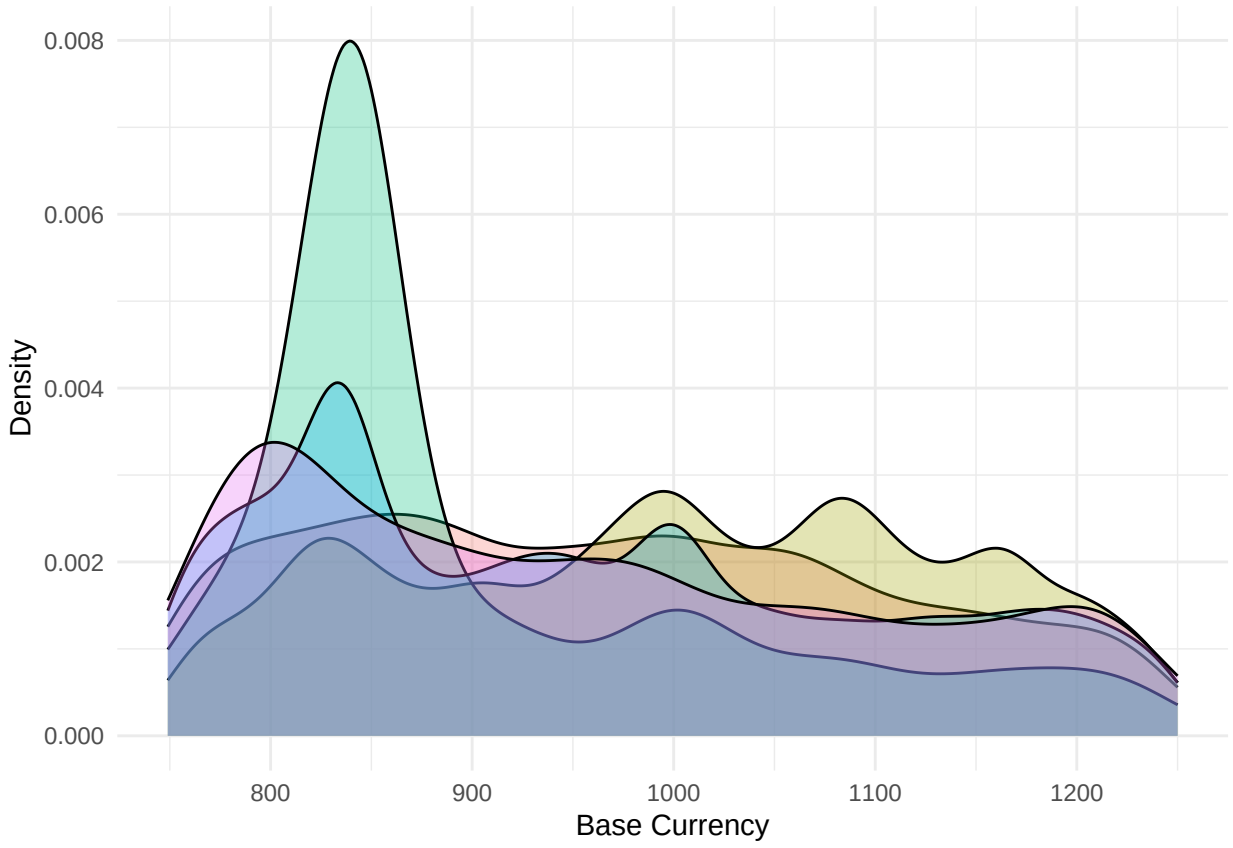

```
################################################################################
### FIGURE 11
################################################################################

ggplot(reg_df[(reg_df$base > 749 & reg_df$base < 1250),], aes(x=base)) + theme_minimal() +
  geom_density(aes(fill=exchange), alpha=0.3) +
  theme(legend.position="none") +
  labs(x="Base Currency",
       y="Density")
```




```
##############################################################################
### FIGURE 12
##############################################################################

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

# 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))

## 1. Import data

unreg_df <- read.csv("unregulated_eth_transactions.csv")
reg_df <- read.csv("regulated_eth_transactions.csv")

## 2. Regulated exchanges

ggplot(reg_df[(reg_df$base > 749 & reg_df$base < 1250),], aes(x=base)) + theme_minimal() +
  geom_density(aes(fill=exchange), alpha=0.3) +
  theme(legend.position='none') +
  labs(x="Base Currency",
       y="Density") +
  ylim(min=0,max=0.008)
```




```
## 3. Unregulated exchanges

ggplot(unreg_df[(unreg_df$base > 749 & unreg_df$base < 1250),], aes(x=base)) + theme_minimal() +
  geom_density(aes(fill=exchange), alpha=0.3) +
  theme(legend.position='none') +
  labs(x="Base Currency",
       y="Density")
```


0.008
0.006
0.004
0.002
0.000
Density
800
900
1000
1100
1200
Base Currency