R Programming and Application to Finance

Mikhail Mironov, Mikhail Frolov

Topic 6. Wash trading on cryptocurrency exchanges

PS: Hopefully, we will match some results of the paper. We attempted to do our best at replicating what the authors of the original paper did. We have collected data for the following three exchanges. Essentially it is very easy to scale to other exchanges as they all have copy-paste data websites.

- **Binance**
- **OKX**
- **Gateio**

Task 0. Data Collection

We decided to go for LTC which is traded on all exchanges and doesn't have huge trading activity as compared to BTC.

Download the tick trade order data for the latest 6 month for a crypto-asset of your choice for 3 crypto-exchanges using (if possible) 3 unregulated crypto-exchanges listed on Panel B and C in Appendix E on page 72 of the previous paper.

```
library(httr)
library(jsonlite)
library(lubridate)
library(readr)
library(ggplot2)
library(dplyr)
library(data.table)
```

Binance

https://data.binance.vision/

Download tick data from binance exchange using their API (data.vision api). Data is already aggregated there by months.

```
parse_binance = function(ticker, date) {

  # Collect data from binance
  year = lubridate::year(date)
  month = sprintf("%02d", lubridate::month(date))

  BINANCE_API_ENDPOINT = sprintf(
    "https://data.binance.vision/data/spot/monthly/trades/%s/%s-trades-%s-%s.zip",
    ticker, ticker, year, month
```

```
  )

  resp = httr::GET(
    url=BINANCE_API_ENDPOINT,
    write_disk("data.zip", overwrite = TRUE)
  )

  if (resp$status_code != 200) {
    stop(content(resp, "text"))
  }

  columns = c(
    "trade_id", "price", "qty", "quoteQty", "time",
    "isBuyerMaker", "isBestMatch"
  )

  short_cols = c(
    "price", "qty", "quoteQty", "time"
  )

  file = unzip("data.zip")

  # df = read.csv(
  #   file, header=FALSE, col.names=columns
  # )

  # Only load necessary columns (we will only need price and qty fields => c(2, 3))
  df = fread(file, select=c(2, 3, 4, 5))
  colnames(df) = short_cols

  # Cleanup files after unzipping and reading to csv
  file.remove("data.zip")
  file.remove(file)

  df$time = df$time = format(
    as.POSIXct(as.numeric(df$time) / 1000), format="%Y-%m-%d %H:%M:%OS3"
  )

  return(df)
}
```

```
# Collect data for these months and years for Binance
end_date = as.Date("2022-12-01")
start_date = end_date - months(6)

date_seq = seq(from=start_date, to=end_date, by="month")
# collect data from Binance
df_binance = data.frame()

pbar <- txtProgressBar(
  min = 0,
  max = length(date_seq),
  style = 3,
  width = 50,
```

```
  char = "="
)
```

```
for (i in 1:length(date_seq)) {
  df_trades = parse_binance(ticker="LTCUSDT", date=date_seq[i])
  df_binance = rbind(df_binance, df_trades)

  setTxtProgressBar(pbar, i)
}
```

```
close(pbar)
```

OKX

OKX provides historical data in zip files

https://www.okx.com/ru/data-download

Trade data is stored in zip files by each day of the month, therefore the procedure is the following, we should query each zip file, unzip it and extract the ticker we want:

https://www.okx.com/cdn/okex/traderecords/trades/daily/20240310/CELO-USDT-trades-2024-03-10.zip

```
parse_okx = function(ticker, date) {
  # Return dataframe for a given date and ticker
  year = lubridate::year(date)
  # add a leading 0 to month and day
  month = sprintf("%02d", lubridate::month(date))
  day = sprintf("%02d", lubridate::day(date))

  OKX_API_ENDPOINT = sprintf(
    "https://www.okx.com/cdn/okex/traderecords/trades/daily/%s%s%s/%s-trades-%s-%s-%s.zip",
    year, month, day, ticker, year, month, day
  )

  resp = httr::GET(
    url=OKX_API_ENDPOINT,
    httr::write_disk("data.zip", overwrite=TRUE)
  )

  if (resp$status_code != 200) {
    stop(content(resp, "text"))
  }

  file = unzip("data.zip")

  cols = c(
     "symbol", "side", "qty", "price", "time"
  )

  df = read.csv(
    file, col.names=cols
  )
  # Clean up files
  file.remove("data.zip")
```

```
  file.remove(file)
  # convert time from timestamp ms to datetime
  df$time = format(
    as.POSIXct(as.numeric(df$time) / 1000), format="%Y-%m-%d %H:%M:%OS3"
  )

  return(df)
}
```

Run the function above in the for loop for each date in the last six months

```
end_date = as.Date("2022-12-01")
start_date = end_date - months(6)

date_seq = seq(from=start_date, to=end_date, by="day")

df_okx = data.frame()

# add a progress bar to the for loop
pb <- txtProgressBar(
  min = 0,
  max = length(date_seq),
  style = 3,
  width = 50,
  char = "="
)
```

```
for (i in 1:length(date_seq)) {
  df_date = parse_okx(
    ticker="LTC-USDT", date=date_seq[i]
  )
  df_okx = rbind(df_okx, df_date)

  setTxtProgressBar(pb, i)
}
```

```
close(pb)
```

GATEIO

Find href on download button, we will use this href to download data from R. Data is aggregated by months

https://download.gatedata.org/spot/deals/202402/AVAX_USDT-202402.csv.gz

```
parse_gateio = function(ticker, date) {
  # Parse GATEIO exchange, get tick data for a ticker for a particular month
  year = lubridate::year(date)
  month = sprintf("%02d", lubridate::month(date))

  GATEIO_API_ENDPOINT = sprintf(
    "https://download.gatedata.org/spot/deals/%s%s/%s-%s%s.csv.gz", year, month, ticker
    year, month
  )

  resp = httr::GET(
```

```
    url=GATEIO_API_ENDPOINT,
    httr::write_disk("data.csv.gz", overwrite=TRUE)
  )

  if (resp$status_code != 200) {
    stop(content(resp, "text"))
  }

  cols = c(
    "time", "trade_id", "price", "qty", "side"
  )
  df = readr::read_csv(
    "data.csv.gz", col_names=cols, show_col_types=FALSE
  )

  df$time = format(
    as.POSIXct(as.numeric(df$time)), format="%Y-%m-%d %H:%M:%OS3"
  )

  file.remove("data.csv.gz")

  return(df)
}
```

Start collecting data from GATEIO exchange for a given ticker

```
# For gateio there is no trades data yet for March as a signle file
so I will download everything until March
end_date = as.Date("2022-12-01")
start_date = end_date - months(6)

date_seq = seq(from=start_date, to=end_date, by="month")

df_gateio = data.frame()

# add a progress bar to the for loop
pb <- txtProgressBar(
  min = 0,
  max = length(date_seq),
  style = 3,
  width = 50,
  char = "="
)
```

```
i = 0

for (i in 1:length(date_seq)) {
  df_date = parse_gateio(
    ticker="LTC_USDT", date=date_seq[i]
  )
  df_gateio = rbind(df_gateio, df_date)

  setTxtProgressBar(pb, i)

  i = i + 1
```

```
}
```

```
close(pb)
```

Now we have tick data for all 3 exchanges for the last 6 months. Now, let's proceed with analysis.

Task 1. Benford's Law

Investigate whether the first-significant-digit distribution of transactions on each exchange conforms to the pattern implied by Benford's law (see section 3.1 of the previous paper). There are several tests for this law that are very easy to implement.

Binance

We should check if the first digit of quantities follows a logarithmic distribution. Authors suggest that first digits should be the most prevalent as starting digits of trade quantities, if otherwise it could be indicative of trades happening sort of in non-natural way, thus wash-trading or other manipulations.

```
# aggreate ticks by time and sum qty and compute quoteQty which is USDT value of the trade
df_binance = df_binance %>%
  group_by(time) %>%
  summarize(
    qty = sum(qty), quoteQty = sum(quoteQty)
  )
```

```
# Create a column for qty first digit
df_binance["qty_first_digit"] = as.numeric(
  substr(as.character(df_binance$qty), 1, 1)
)

df_binance = df_binance %>% filter(qty_first_digit != 0)
```

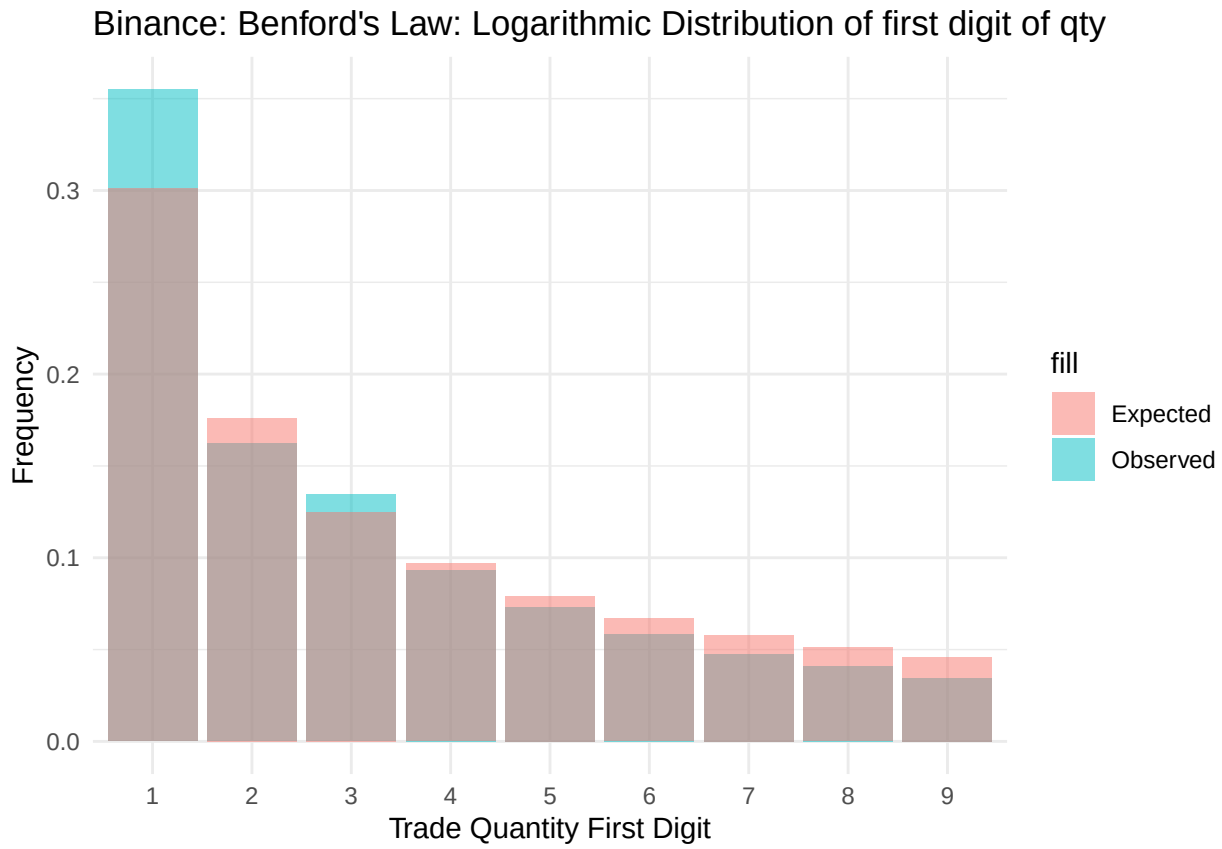
We observe the following frequencies of the leading digits in the data. Let's see how it compares to population frequencies:

```
# Observed frequencies
N = nrow(df_binance)
first_digits_freqs = table(df_binance$qty_first_digit) / N

df_freq = as.data.frame(first_digits_freqs)
colnames(df_freq) = c("digit", "observed_freq")

# Expected frequencies
digits = seq(1, 9, by=1)

expected_freq = sapply(
  digits, function(i) log10(1 + 1/i)
)

df_freq["expected_freq"] = expected_freq
```

```
# define plotting function to use further down the line
plot_benford = function(df_freq, exchange) {
  ggplot(
    data = df_freq, aes(x = factor(digit))
```

```
  ) + geom_bar(
    aes(y = observed_freq, fill="Observed"), stat = "identity", alpha=0.5
  ) + geom_bar(
    aes(y = expected_freq, fill = "Expected"), stat = "identity", alpha=0.5
  ) + labs(
    title = sprintf("%s: Benford's Law: Logarithmic Distribution of first digit of qty", exchange),
    x = "Trade Quantity First Digit", y = "Frequency"
  ) + theme_minimal()
}
```

```
plot_benford(df_freq=df_freq, exchange="Binance")
```



Calculate **chi_statistic** defined as follows:

$$\chi^2 = \sum_{i=1}^{9} \frac{(O_i - E_i)^2}{E_i}$$

The null hypothesis is that the first-significant-digit distribution observed in an exchange's trading data is consistent with that of Benford's law:

```
chi2_stat = sum(
  (df_freq$observed_freq - df_freq$expected_freq)^2 / df_freq$expected_freq
)

1 - pchisq(chi2_stat, 8)
```

```
## [1] 1
```

Run chisq.test:

```
chisq.test(
  x=df_freq$observed_freq, p=df_freq$expected_freq, rescale.p=TRUE
)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  df_freq$observed_freq
## X-squared = 0.0198, df = 8, p-value = 1
```

OKX

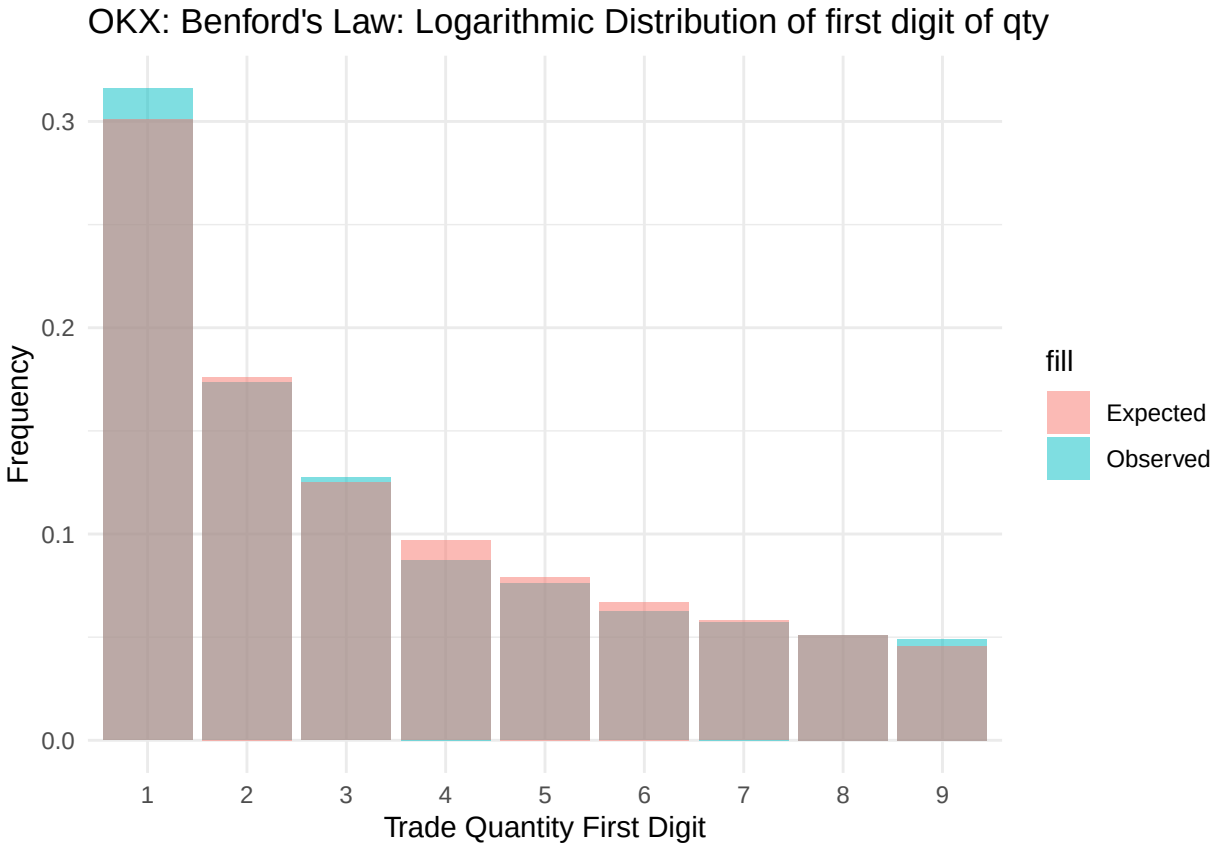
Now perform the same steps of OKX exchange, aggregate trades by ms, calculate qty volume, quoteQty volume and use these for Benford's Law:

```
# aggreate ticks by time and sum qty and compute quoteQty which is USDT value of the trade
df_okx = df_okx %>%
  group_by(time) %>%
  summarize(
    qty = sum(qty), quoteQty = sum(qty * price)
  )
```

```
# Create a column for qty first digit
df_okx["qty_first_digit"] = as.numeric(
  substr(as.character(df_okx$qty), 1, 1)
)

df_okx = df_okx %>% filter(qty_first_digit != 0)
```

```
# Observed frequencies
N = nrow(df_okx)
first_digits_freqs = table(df_okx$qty_first_digit) / N

df_freq = as.data.frame(first_digits_freqs)
colnames(df_freq) = c("digit", "observed_freq")

# Expected frequencies
digits = seq(1, 9, by=1)

expected_freq = sapply(
  digits, function(i) log10(1 + 1/i)
)

df_freq["expected_freq"] = expected_freq
```

```
plot_benford(df_freq=df_freq, exchange="OKX")
```




```
# check frequencies for OKX
chisq.test(
  x=df_freq$observed_freq, p=df_freq$expected_freq, rescale.p=TRUE
)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  df_freq$observed_freq
## X-squared = 0.0024686, df = 8, p-value = 1
```

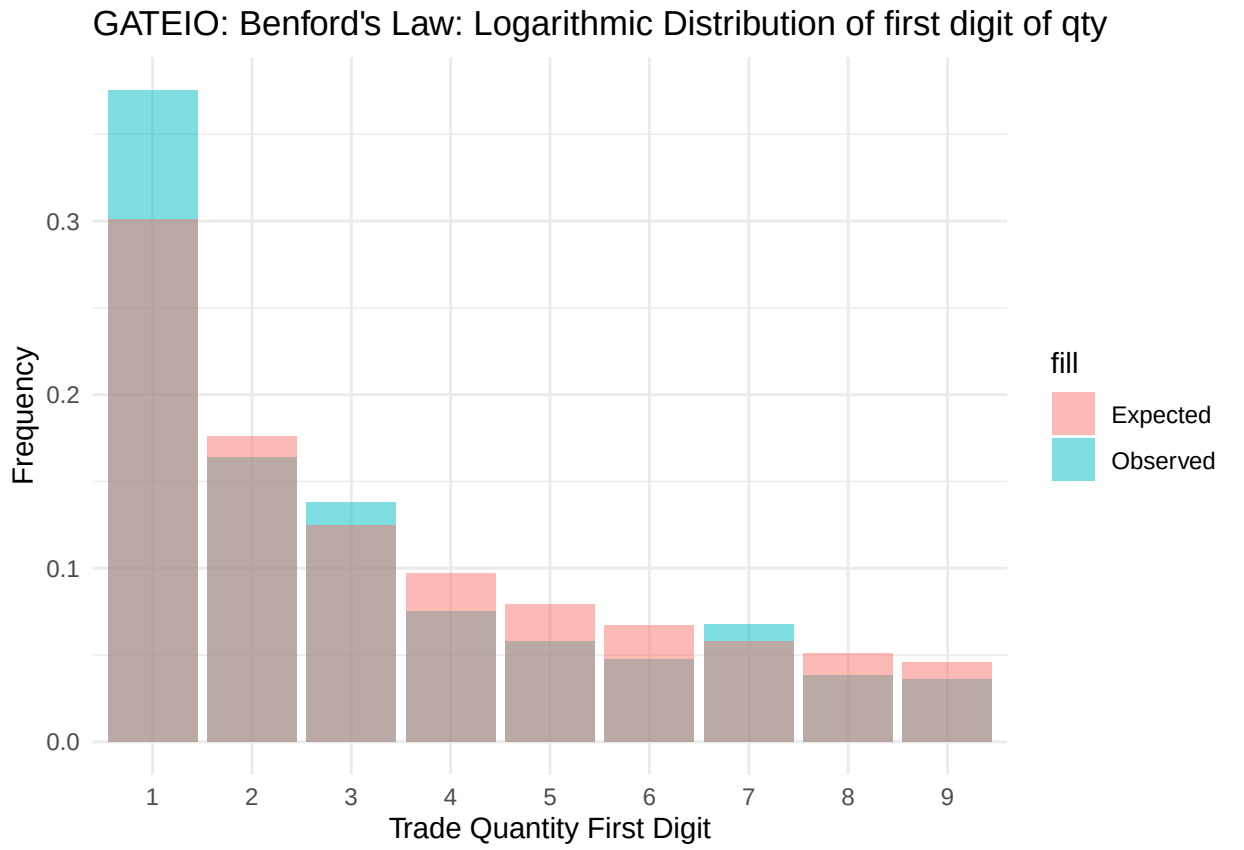
GATEIO

```
# aggreate ticks by time and sum qty and compute quoteQty which is USDT value of the trade
df_gateio = df_gateio %>%
  group_by(time) %>%
  summarize(
    qty = sum(qty), quoteQty = sum(qty * price)
  )
```

```
# Create a column for qty first digit
df_gateio["qty_first_digit"] = as.numeric(
  substr(as.character(df_gateio$qty), 1, 1)
)

df_gateio = df_gateio %>% filter(qty_first_digit != 0)
```

```
# Observed frequencies
N = nrow(df_gateio)
first_digits_freqs = table(df_gateio$qty_first_digit) / N

df_freq = as.data.frame(first_digits_freqs)
colnames(df_freq) = c("digit", "observed_freq")

# Expected frequencies
digits = seq(1, 9, by=1)

expected_freq = sapply(
  digits, function(i) log10(1 + 1/i)
)

df_freq["expected_freq"] = expected_freq
```

```
plot_benford(df_freq=df_freq, exchange="GATEIO")
```




```
# check frequencies for Gateio
chisq.test(
  x=df_freq$observed_freq, p=df_freq$expected_freq, rescale.p=TRUE
)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  df_freq$observed_freq
```

```
## X-squared = 0.04395, df = 8, p-value = 1
```

Results

As we can see all tests show that we can't almost for certain reject the null hypothesis that observed frequencies follow Benford's Law since in all cases pvalues is close 1. This is honestly strange as we can clearly see some deviation for Binance and Gateio for "1" as a leading digit. I would say that results obtained, especially what we observe on the barplots is consistent with real life, as historically OKX has been one of the most regulated exchanges, Binance at a time used to somewhat unregulated, Gateio I would say is still unregulated to this day, it is a go-to exchange to buy shady meme coins. Therefore, everything more or less checks out, OKX should not have much shady trading, Binance is somewhere in the middle and Gateio is the last.

Task 2. Trade-size clustering

Conduct a Student's t-test for each crypto exchange to quantify the effect of trade-size clustering by comparing trade frequencies at round trade sizes with the highest frequency of nearby unrounded trades. The null hypothesis of the test is that the difference between frequencies at rounded numbers and nearby unrounded trades is zero.

We should check the following claim from the paper: "Because wash traders use machine-based automated trading programs to save manpower, especially when fake orders feature small trade sizes but large total amounts (Vigna and Osipovich, 2018; Rodgers, 2019), wash trading naturally reduces the proportion of authentic volume, and thus clustering."

We should also aggregate data based on time, because trades executed at the same millisecond are likely associated with one person and one order, it is split up into multiple trades as it matches multiple other orders in the orderbook. Therefore, we are good to aggregate the data and sum the qty and quoteQty. We also, as I believe slightly improved on the approach used in the paper, as we check both asset qty and quoteQty for roundness at the same time. Either the amount of asset is somewhat round or the amount of quote spent. We define roundness by performing modulo division.

```
create_binned_df = function(df, bin_size, bin_start, bin_end) {
  # creates a dataframe with quantities aggregated buy bins of a given size
  breaks = seq(bin_start, bin_end, by=bin_size)

  df_bin = df %>%
    mutate(bin = cut(quoteQty, breaks=breaks, ordered_result=TRUE)) %>%
    group_by(bin) %>%
    summarize(bin_qty = sum(quoteQty))

  labs = df_bin$bin

  df_bin = cbind(
    df_bin,
    lower = as.numeric( sub("\\((.+),.*", "\\1", labs) ),
    upper = as.numeric( sub("[^,]*,([^]]*)\\]", "\\1", labs) )
  )
  return(df_bin)
}
```

```
# plot barplot of quantities vs volume traded at this quote quantity
plot_qty_bar = function(df_bin, exchange, lb=0, ub=2000) {
  ggplot(
    data=df_bin %>% filter(between(lower, lb, ub)),
```

```
    aes(x=lower)
  ) + geom_bar(
    aes(y=bin_qty), stat="identity", alpha=0.7, color="blue"
  ) + labs(
    title=sprintf("LTCUSDT - %s. Aggregation of trades around round quote amounts", exchange),
    x="Quote amount USDT", y="Volume USDT"
  ) + theme_minimal()
}
```

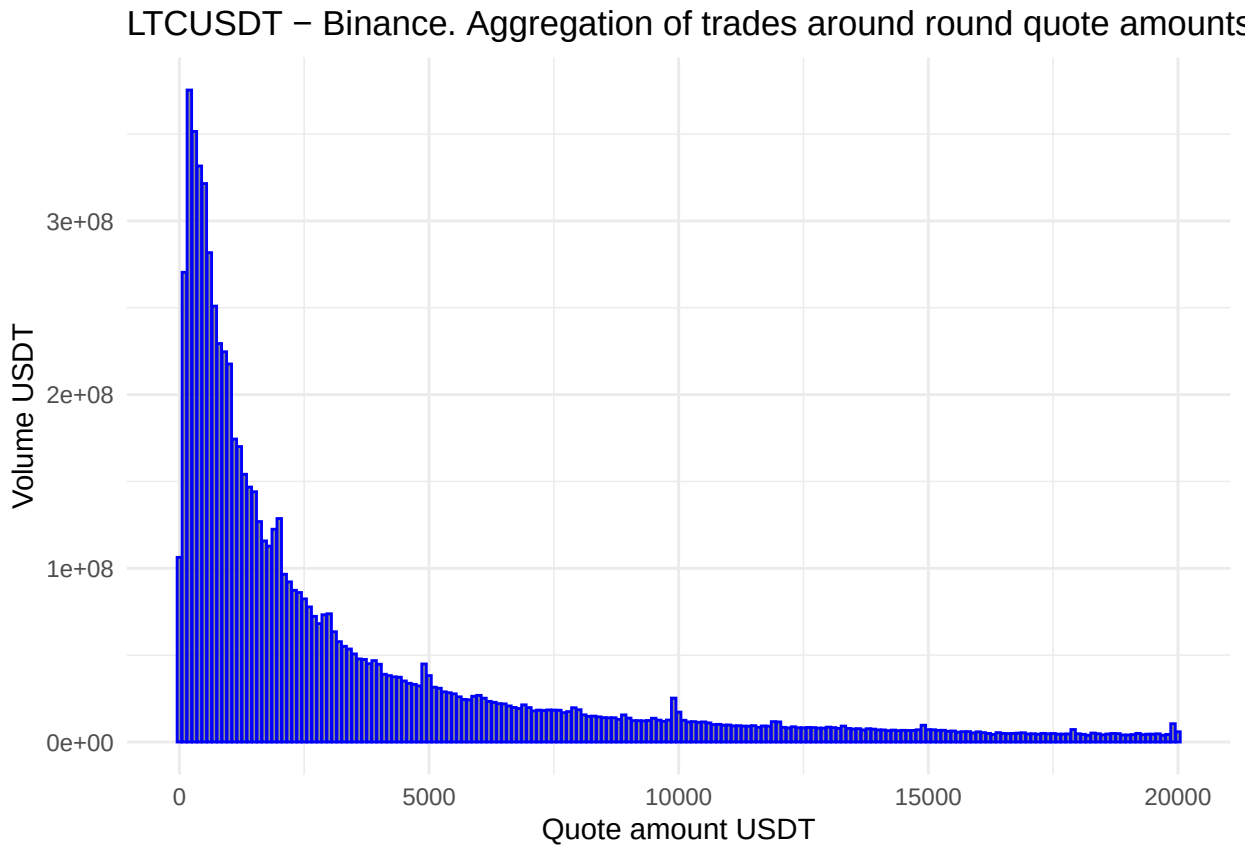
Binance

People might buy 0.01LTC, 0.1LTC, 1LTC and etc, therefore checking with mod division by 0.01 will help us to identify such round amounts of LTC bought, at the same time we check for quantities of quote spent (USDT) which are a multiple of 100.

```
# Calculate binned overall volume up until 99 quantile, exclude heavy outliers
df_bin_100 = create_binned_df(
  df=df_binance, bin_size=100, bin_start=0, bin_end=quantile(df_binance$quoteQty, .999)
)
df_bin_10 = create_binned_df(
  df=df_binance, bin_size=10, bin_start=0, bin_end=quantile(df_binance$quoteQty, .999)
)
```

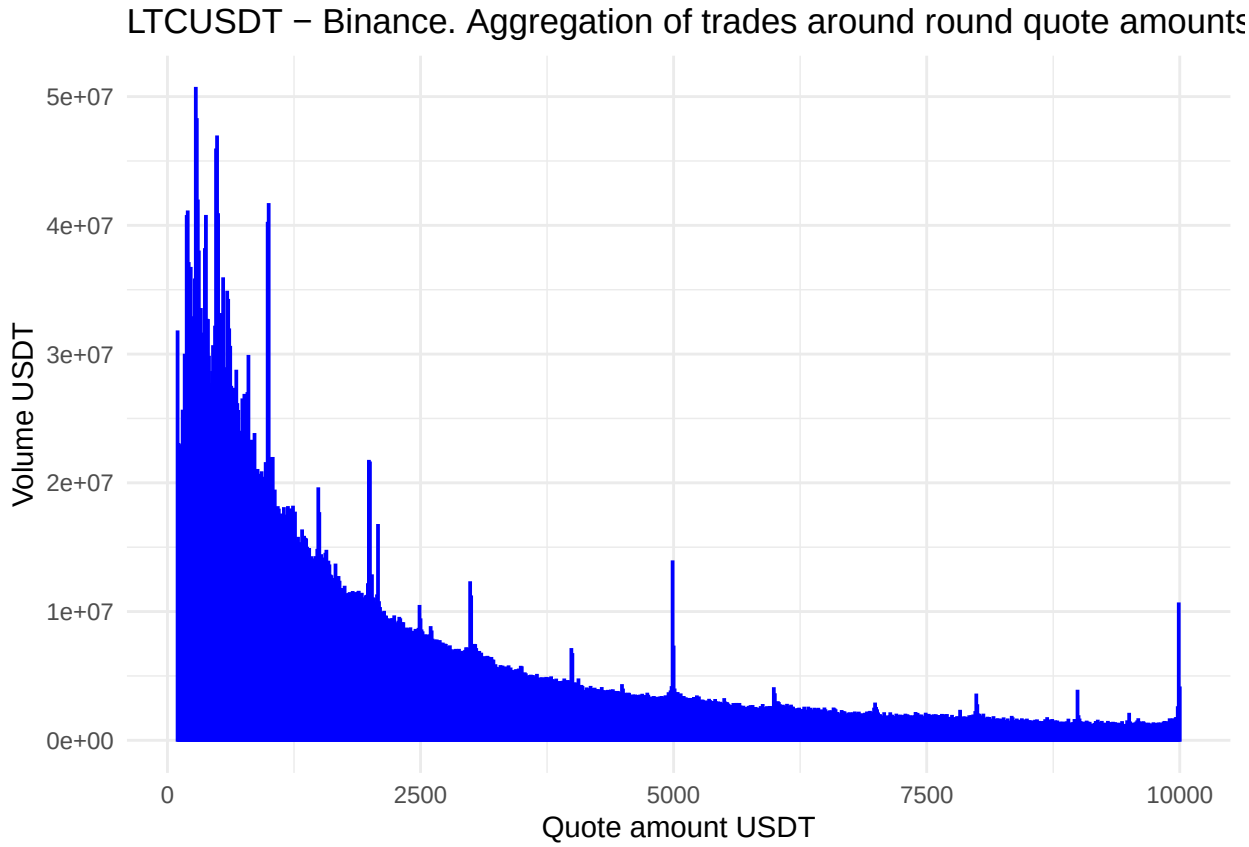
Below we get similar graphs as in the paper, we calculated overall trading volume in USDT calculated over bins of size 10 USDT. As we can, for Binance there is steady decline in overall volume as trade sizes increase which is consistent with legit exchanges. Also the barplot is quite spiky, especially around round quote_asset quantities like 100, 200, 500, 1000 USDT and etc which indicates according to the authors the prevalence of people trading rather than bots which typically operate with not round quantities.

```
plot_qty_bar(df_bin=df_bin_100, exchange="Binance", ub=20000)
```



The plot below is the same but with higher granularity of bins, now it is down to 10, this allows us to see how volumes like 100, 150, 200 dominate other quantities in terms trading volume in USDT

```
plot_qty_bar(df_bin=df_bin_10, exchange="Binance", lb=100, ub=10000)
```



For each value multiple of 100, find volume traded at +-epsilon this amount and compare it to the highest bin of size 2*eps within +-10 USD interval:

```
calculate_df_round_unround_vol = function(df_bin, df_agg) {
  # calculate trade volumes at rounded trades - break points multiples of 100
  # I know we should have vectorized
  vols_at_round = c()
  vols_max_unround = c()

  eps = 1

  pb <- txtProgressBar(
    min = 0,
    max = nrow(df_bin),
    style = 3,
    width = 50,
    char = "="
  )

  N = nrow(df_bin) - 1

  for (i in 1:N) {

    round_qty = df_bin$lower[i]
    lb = round_qty - eps*10
    ub = round_qty + eps*10
```

```
    # leave only observations (100X-eps*10, 100X-eps) | (100X+eps, 100X+eps*10)
    df_proxima = df_agg %>% filter(
      between(quoteQty, lb, round_qty-eps) | between(quoteQty, round_qty+eps, ub)
    )

    # find max volume within (100X-eps*10, 100X+eps*10)
    df_binned = create_binned_df(
      df=df_proxima, bin_size=2*eps, bin_start=lb, bin_end=ub
    )
    if (all(is.na(df_binned$bin_qty))) {
      max_not_round_vol = 0
    } else {
      max_not_round_vol = max(df_binned$bin_qty, na.rm=TRUE)
    }

    # Calculate volume for close to round multiple of 100X (100X-eps, 100X+eps)
    rounded_vol = df_agg %>% filter(
      between(quoteQty, round_qty-eps, round_qty+eps)
    ) %>% summarize(sum(quoteQty)) %>% pull()

    vols_at_round = append(vols_at_round, rounded_vol)
    vols_max_unround = append(vols_max_unround, max_not_round_vol)

    setTxtProgressBar(pb, i)
  }

  df_ttest = data.frame(
    lower=df_bin$lower[1:N],
    bin=df_bin$bin[1:N],
    round_qty_vol=vols_at_round,
    eps_interval_vol=vols_max_unround
  )

  return(df_ttest)
}
```

```
# Calculate trade volumes for round and max unround quoteQty
df_binance_ttest = calculate_df_round_unround_vol(
  df_bin=df_bin_100, df_agg=df_binance
)
```

The null hypothesis of the test is that the difference between frequencies at rounded numbers and nearby unrounded trades is zero.

$$H_0 : \mu_1 = \mu_2$$

$$H_a : \mu_1 \neq \mu_2$$

We have pvalue of 0.0172 therefore, there is enough evidence to reject the null hypothesis about equality of the means. Therefore, there is statistically significant difference between volume traded at clustered values of quote (multiples of 10 USDT) and closest highest bins. Essentially, such test is a way to sort of quantify how smooth the histogram of quoteQty against volume traded. If it is smooth enough, which is according to the author is indicative of bot trading, then there is wash trading of this ticker on the exchange, otherwise we observe high spikes at round quantities of the quote asset (USDT), therefore there is more actual people activity on the exchange (not much wash trading)

```r
# perform t-test for two means if they are equal
res = t.test(
  df_binance_ttest$round_qty_vol, df_binance_ttest$eps_interval_vol, alternative="two.sided"
)

res$p.value
```

```
## [1] 0.01717863
```

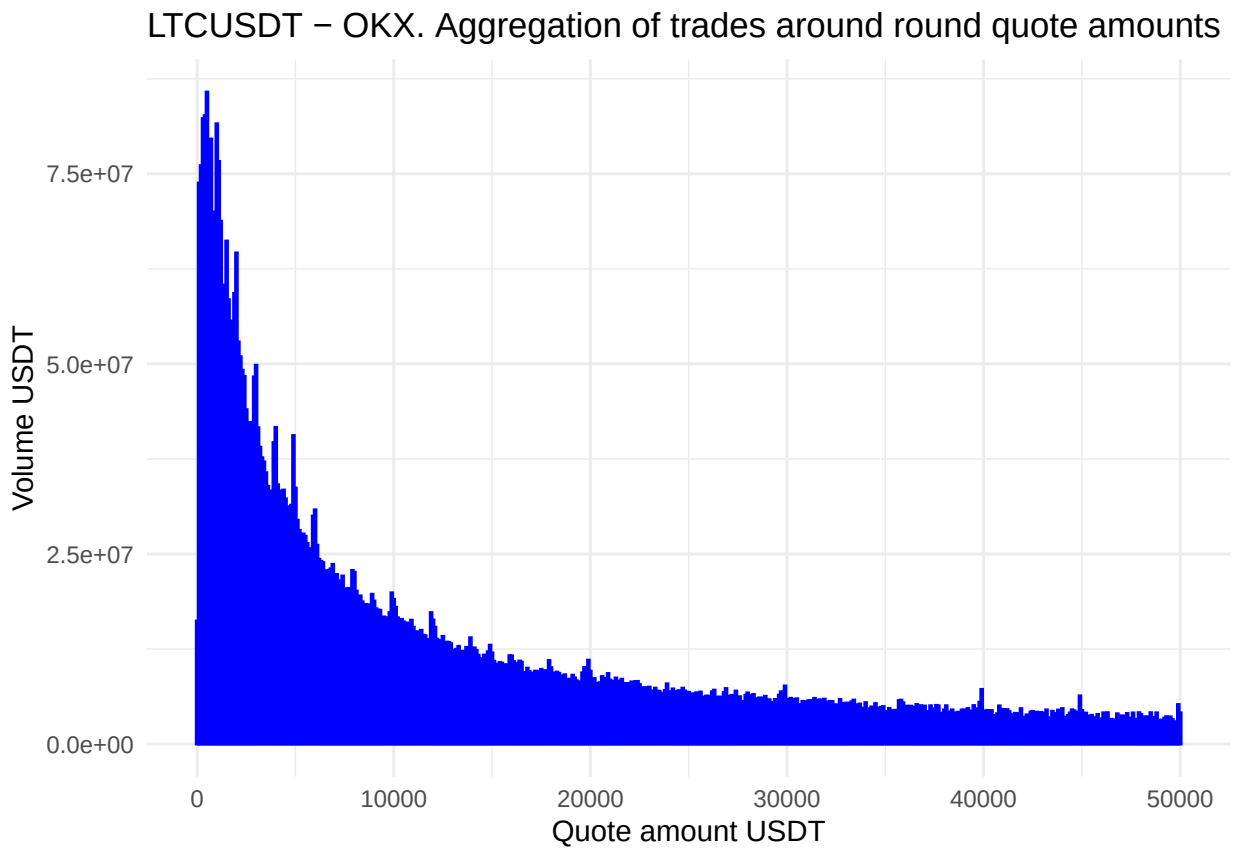
OKX

Perform the same steps for OKX exchange:

```r
# Calculate binned overall volume up until 99 quantile, exclude heavy outliers
df_bin_100 = create_binned_df(
  df=df_okx, bin_size=100, bin_start=0, bin_end=quantile(df_okx$quoteQty, .999)
)
df_bin_10 = create_binned_df(
  df=df_okx, bin_size=10, bin_start=0, bin_end=quantile(df_okx$quoteQty, .999)
)
```

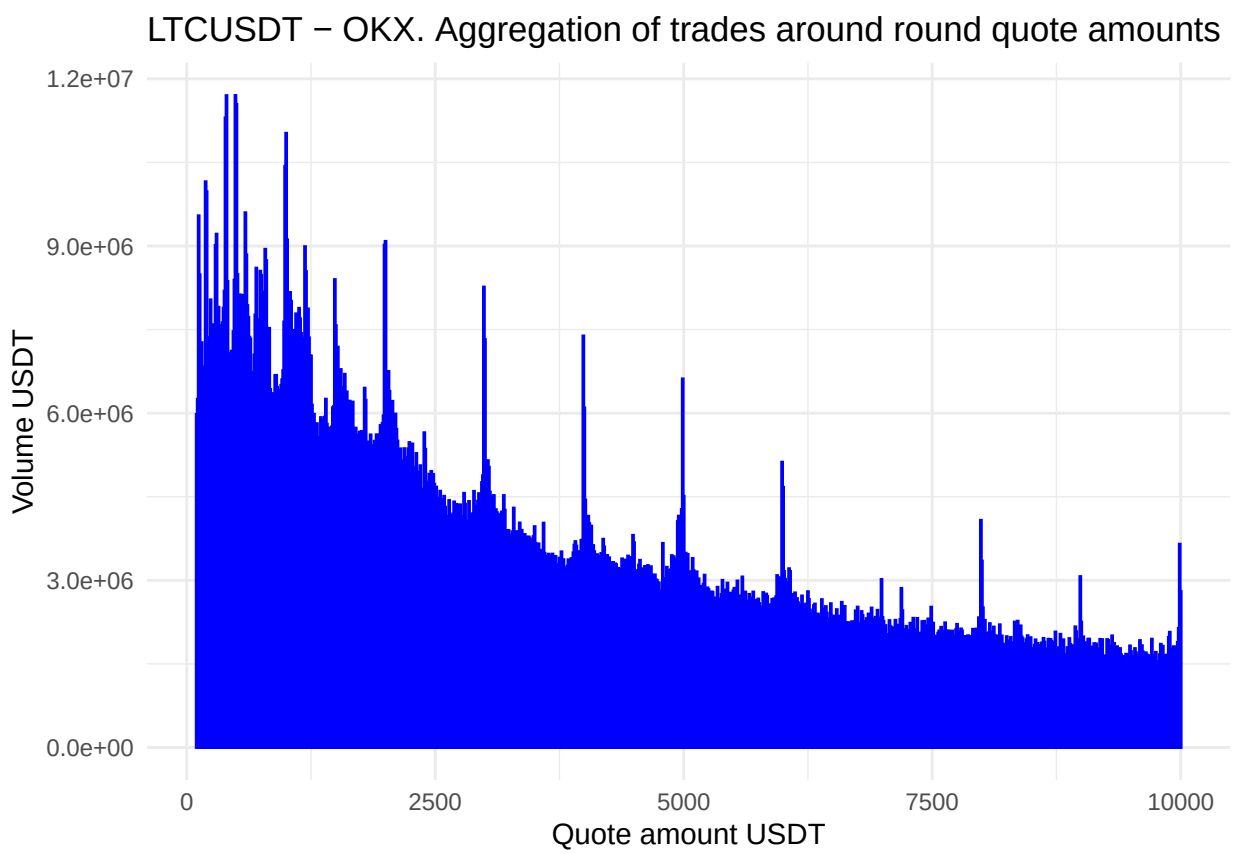
We observe the same behavior as on Binance, volume goes down slower as Quote amount increases, this might be attributed to the fact the Binance is the first exchange you interact with when diving into crypto => more kinda first time trades of smaller quantity. Here it is almost a linear decreasing relationship

```r
plot_qty_bar(df_bin=df_bin_100, exchange="OKX", ub=50000)
```



This one is more spiky as compared to Binance, this is exactly what we want to see as we anticipate less bot trading here as OKX has been more regulated

```
plot_qty_bar(df_bin=df_bin_10, exchange="OKX", lb=100, ub=10000)
```




```
# Calculate trade volumes for round and max unround quoteQty
df_okx_ttest = calculate_df_round_unround_vol(
  df_bin=df_bin_100, df_agg=df_okx
)
```

```
# perform t-test for two means if they are equal
res = t.test(
  df_okx_ttest$round_qty_vol, df_okx_ttest$eps_interval_vol, alternative="two.sided"
)

res$p.value
```

```
## [1] 4.163015e-87
```

We definitely reject the null, that means are the same, therefore there is indeed trade clustering at round quote asset amounts of 100X USDT. And the there is this striking difference in p-value, it is orders of magnitude lower than one on Binance which correlates with the fact that graph is visually way more spikey.

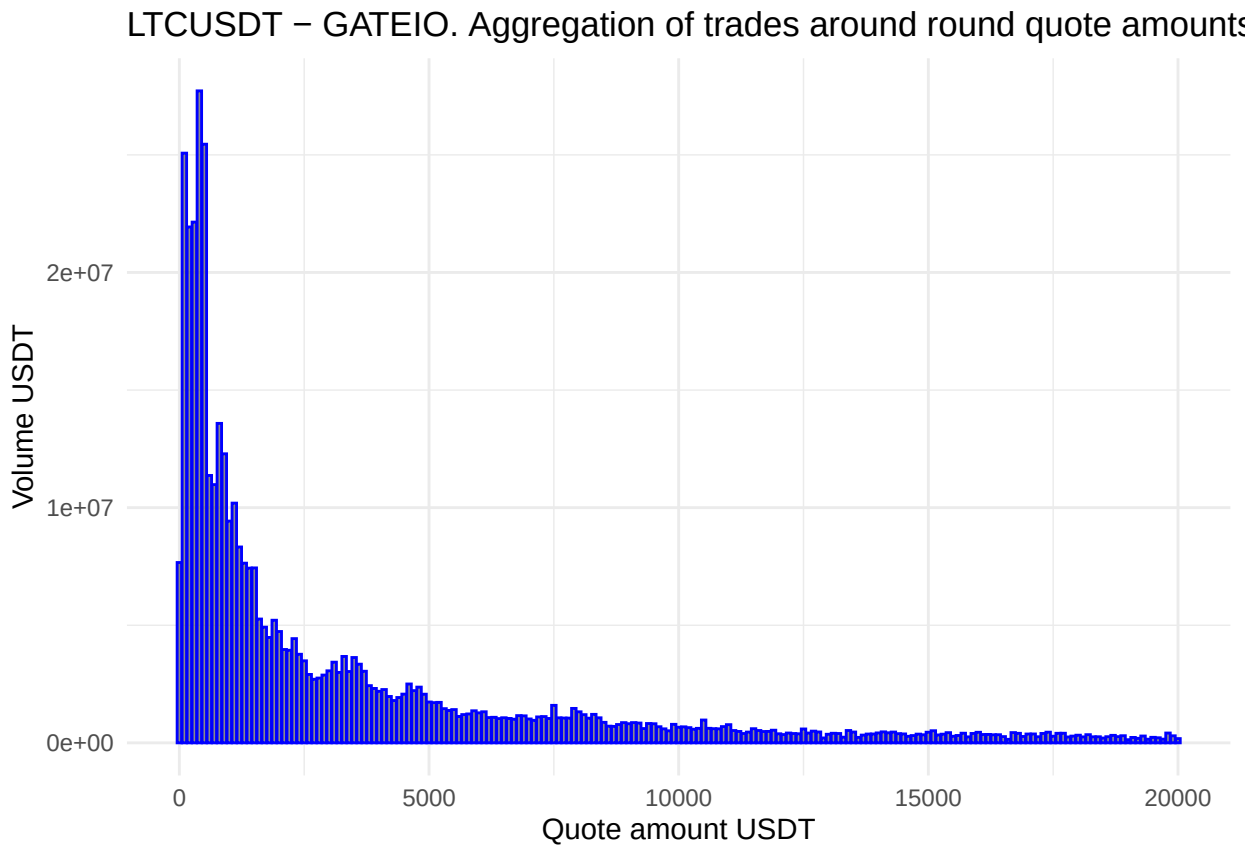
GATEIO

Now we perform the same steps for the most shady out of all exchanges, let's see what happens:

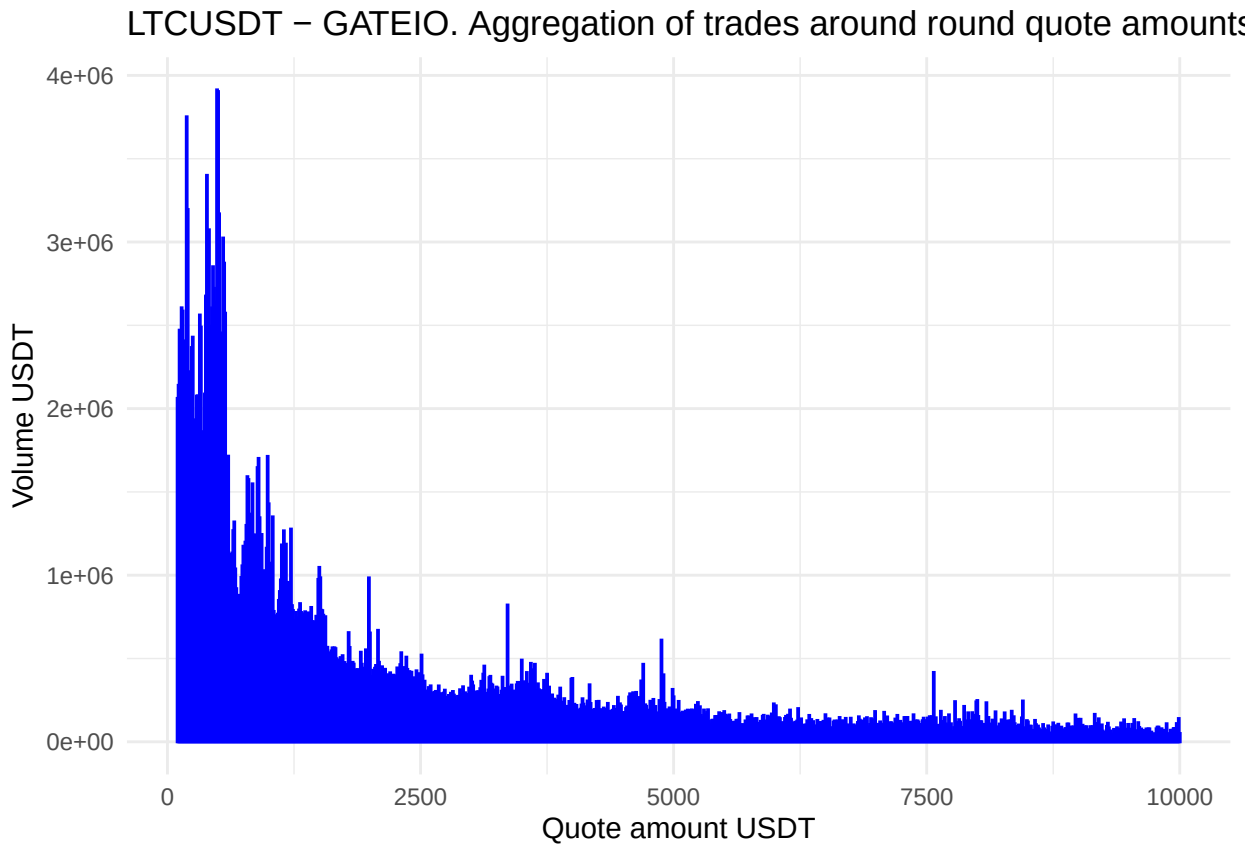
```
# Calculate binned overall volume up until 99 quantile, exclude heavy outliers
df_bin_100 = create_binned_df(
  df=df_gateio, bin_size=100, bin_start=0, bin_end=quantile(df_gateio$quoteQty, .999)
)
df_bin_10 = create_binned_df(
  df=df_gateio, bin_size=10, bin_start=0, bin_end=quantile(df_gateio$quoteQty, .999)
)
```

Here we observe less pronounces spikes at round numbers as expected

```
plot_qty_bar(df_bin=df_bin_100, exchange="GATEIO", ub=20000)
```



```
plot_qty_bar(df_bin=df_bin_10, exchange="GATEIO", lb=100, ub=10000)
```




```
# Calculate trade volumes for round and max unround quoteQty
df_gateio_ttest = calculate_df_round_unround_vol(
  df_bin=df_bin_100, df_agg=df_gateio
)
```

```
# perform t-test for two means if they are equal
res = t.test(
  df_gateio_ttest$round_qty_vol, df_gateio_ttest$eps_interval_vol, alternative="two.sided"
)

res$p.value
```

```
## [1] 0.3511844
```

And this is very interesting, **we fail to reject the null**, therefore means are more or less the same, hence there is not much clustering of trades near round numbers, therefore this could be indicative of washtrading. Visually, the barplot also looks less spikey.

Task 3. Tail Distribution

Examine the tails of trade-size distributions on each crypto exchange. To examine the trade size distribution tails, use two widely adopted techniques: the first is to take the logarithm of the empirical probability density function and fit the log-log data to power-law distribution by Ordinary Least Square (OLS). The second is to apply the Maximum Likelihood Estimation approach (MLE) and use the Hill estimator for the data fitting.

OLS estimate

$$\mathbb{P}(X > x) = x^{-\alpha}$$

$$\alpha_{\text{pdf}} = \alpha + 1$$

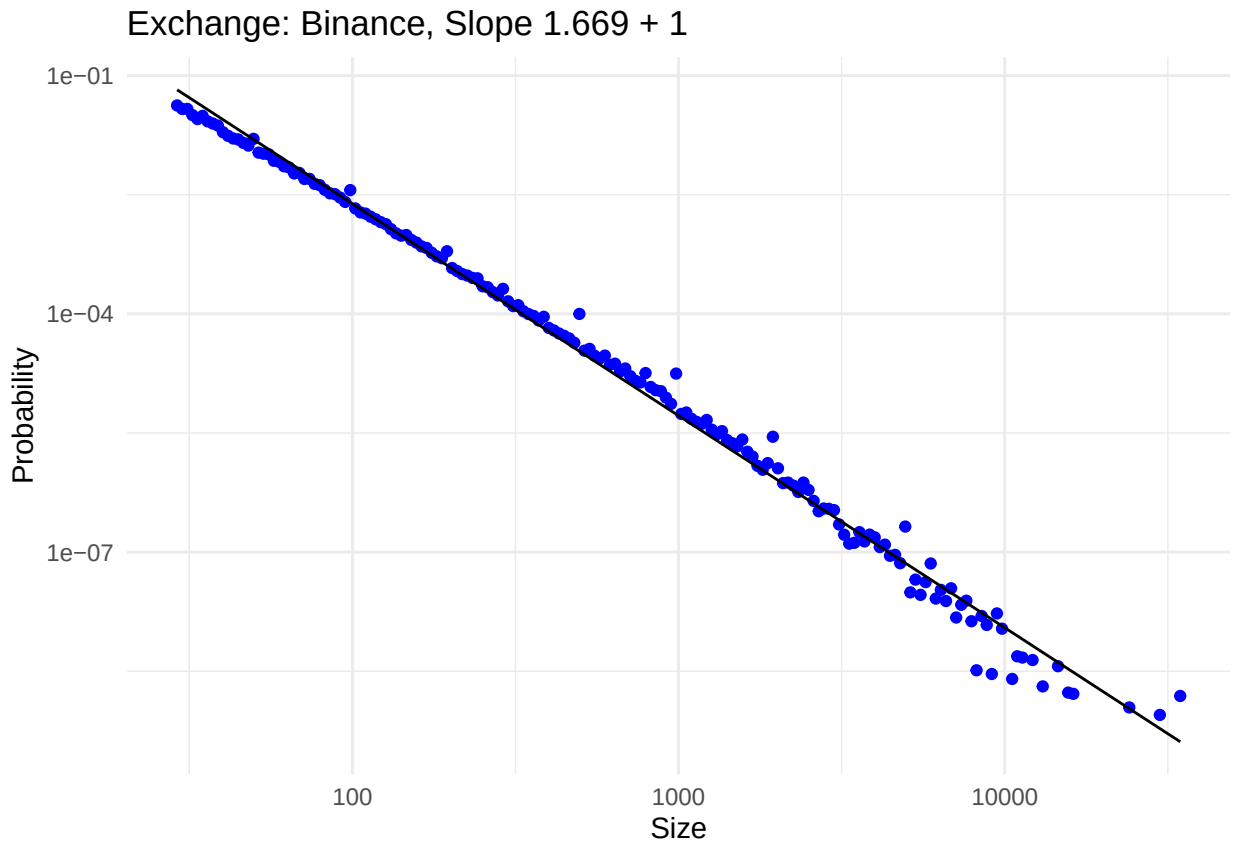
```
# OLS estimate of alpha_pdf
OLS_estimate = function(df_trades, exchange, nbins=200) {

  cutoffpercentile = 0.9

  # Calculate cutoff point
  cutoff = quantile(df_trades$qty, cutoffpercentile)

  # Filter tails
  df_pick = df_trades[df_trades$qty >= cutoff, ]

  # Create log space bins
  logbins2 = 10^seq(log10(cutoff), log10(max(df_trades$qty)), length=nbins)

  # Calculate PDF
  hist1 = hist(df_pick$qty, breaks = logbins2, plot = FALSE)
  df_hist_pdf = data.frame(size = hist1$breaks[-length(hist1$breaks)], p = hist1$density)
  df_hist_pdf = df_hist_pdf[df_hist_pdf$size > cutoff & df_hist_pdf$p != 0, ]

  # Calculate CDF
  hist2 = hist(df_pick$qty, breaks = logbins2, plot = FALSE)
  df_hist_cdf = data.frame(size = hist2$breaks[-length(hist2$breaks)], p = cumsum(hist2$density * diff(
  df_hist_cdf = df_hist_cdf[df_hist_cdf$size > cutoff & df_hist_cdf$p != 0, ]

  # Linear fit
  fit_lm = lm(log10(df_hist_pdf$p) ~ log10(df_hist_pdf$size))
  slope = -coef(fit_lm)[2]
  intercept = coef(fit_lm)[1]

  fit1 = data.frame(size = numeric(198), p = numeric(198), c = numeric(198))
  fit1$size = df_hist_cdf$size
  fit1$p = 10^intercept * fit1$size^(-slope)  # pdf
  fit1$c = (fit1$size / cutoff)^(-slope + 1)  # cdf

  # Create a scatter plot with linear and MLE fits
  plot = ggplot() +
    geom_point(data = df_hist_pdf, aes(x = size, y = p), color="blue") +
    geom_line(data = fit1, aes(x = size, y = p), color = "black", linetype = "solid") +
    scale_x_log10() +
    scale_y_log10() +
    labs(x = "Size", y = "Probability", title=sprintf("Exchange: %s, Slope %s + 1", exchange, round(slo
    theme_minimal() +
    theme(axis.text.x = element_text(), axis.text.y = element_text(), legend.position = "top") +
    guides(color=guide_legend(title="Legend")) +
    theme(legend.title=element_text(), legend.text=element_text())

  return(plot)
}
```

Calculate OLS estimates for each of the three exchanges. As we can see for all exchanges

$$\alpha_{\text{OLS}} \in (1, 2)$$

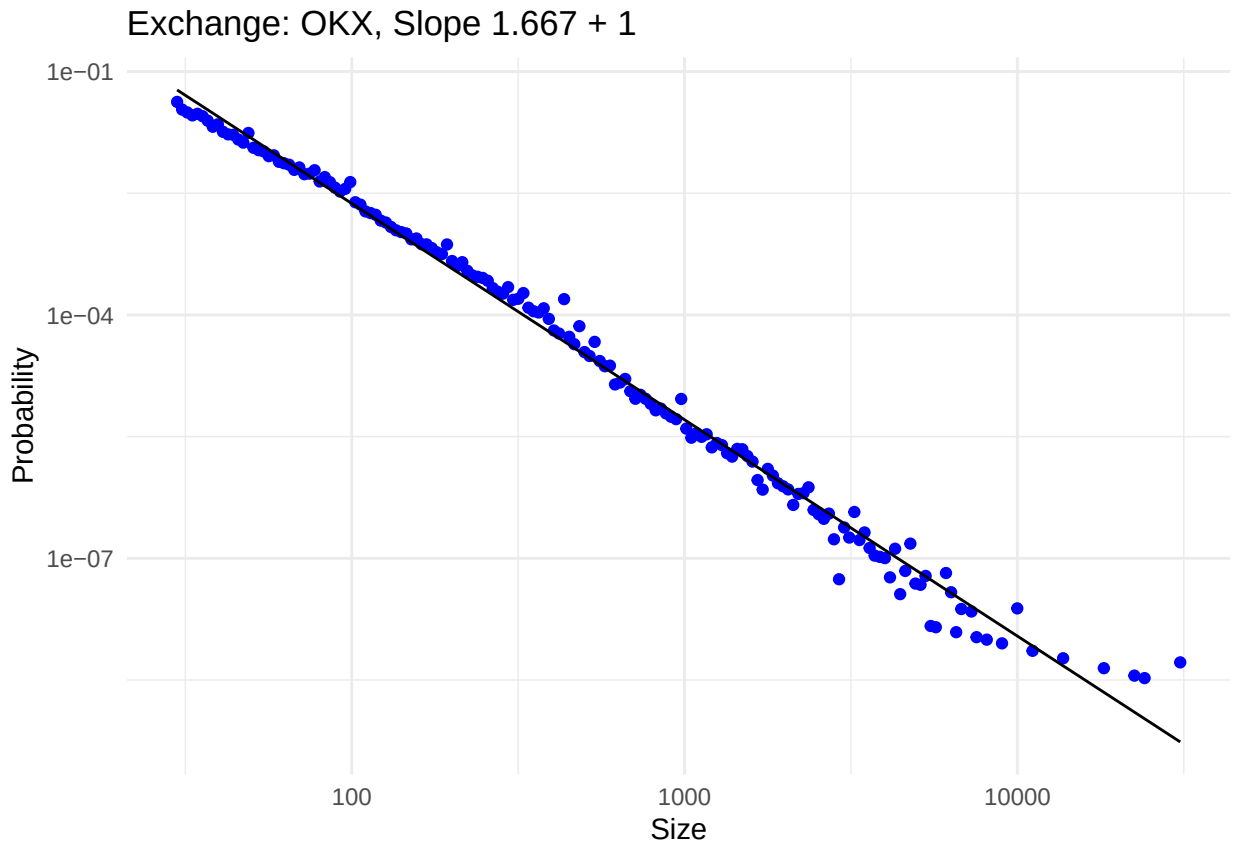
Binance

```
# Binance
OLS_estimate(df_trades=df_binance, exchange="Binance")
```



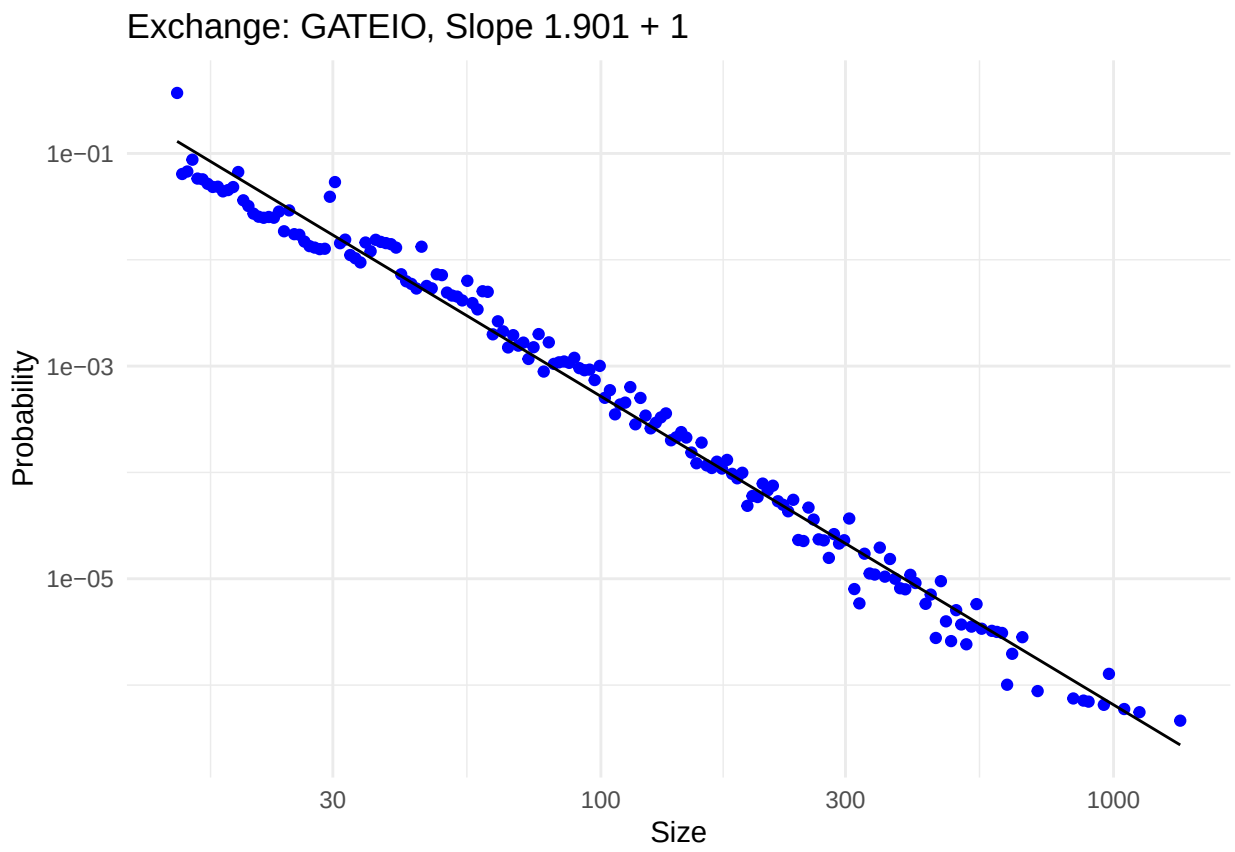
OKX

```
# OKX
OLS_estimate(df_trades=df_okx, exchange="OKX")
```



GATEIO

```
# GATEIO
OLS_estimate(df_trades=df_gateio, nbins=199, exchange="GATEIO")
```



MLE estimate

$$\hat{\alpha}_{\text{Hill}} = 1 + n(\sum_{i=1}^{N} ln(\frac{x_i}{x_{\min}}))^{-1}$$

```
hill_estimator = function(df) {
  # Hill estimator
  X = df %>% filter(qty > quantile(qty, 0.9)) %>% select(qty)
  hill_estimate = nrow(X)*(sum(log(X / min(X))))^(-1)
  return(hill_estimate)
}

# Binance
hill_estimator(df=df_binance)
```

```
## [1] 1.405276
```

```
# OKX
hill_estimator(df=df_okx)
```

```
## [1] 1.366678
```

```r
# GATEIO
hill_estimator(df=df_gateio)
```

```
## [1] 1.662912
```