# Data Science on Blockchain with R. Part 1: reading the blockchain

#### By Thomas de Marchin (Senior Manager Statistics and Data Sciences at Pharmalex) and Milana Filatenkova (Manager Statistics and Data Sciences at Pharmalex)

#### 4/11/2021



Cryptopunks are the earliest versions of NFTs. Image from Cryptopunks.

# Intro

**What is the Blockchain:** A blockchain is a growing list of records, called blocks, that are linked together using cryptography. It is used for recording transactions, tracking assets, and building trust between participating parties. Primarily known for Bitcoin and cryptocurrencies application, Blockchain is now used in almost all domains, including supply chain, healthcare, logistic, identity management,… Hundreds of blockchains exist with their own specifications and applications: Bitcoin, Ethereum, Tezos,…

**What are NFTs:** Non-Fungible Tokens are used to represent ownership of unique items. They let us tokenize things like art, collectibles, even real estate. They can only have one official owner at a time and they’re secured by the Blockchain, no one can modify the record of ownership or copy/paste a new NFT into existence. You’ve probably heard of the artist Beeple who sold one of his NFT art for $69 million. For more information, see <https://ethereum.org/en/nft>,

**What is R:** R language is widely used among statisticians and data miners for developing data analysis software.

**Why doing data science on blockchain:** As Blockchain technology is booming, there is a growing need in tools and people able to read, understand and summarize the information contained on the ledger.

**What is an API:** An API is a software intermediary that allows two applications to talk to each other. APIs are designed to help developers make requests to another software (i.e. downloading information) and getting results in predefined easy to read format, without having to understand how this software works.

There are already several articles dealing with blockchain data analysis inR, but most of them focus on price forecasting. Obtaining data on the cryptocurrencies price is quite straightforward, there are many databases available on internet. But how to actually read the blockchain? In this article, we will focus on reading blockchain transactions. Not all transactions but specifically, transactions related to NFTs. We will read the Ethereum blockchain, probably the top one used to trade NFTs. Several market places allowing to trade NFTs exist: OpenSea, Rarible,… We will focus here on OpenSea, the largest NFT trading places one at the moment.

Reading the raw blockchain is possible but it is hard. First of all, you have to setup a node and download the content of the blockchain (approximately 7TB at the time of writing), it is going to take some time to synchronize… Secondly, the data are stored sequentially which requires developing specific tools to follow a transaction. Third, the structure of the block is particularly difficult to read. Fortunately for us, there are APIs which facilitate our work.

# First, let's load a few useful packages

library(tidyverse)

library(httr)

library(jsonlite)

library(scales)

library(waffle)

# OpenSea API

OpenSea provides an API for fetching non-fungible ERC721 assets based on a set of query parameters. Let’s have a look:

# Retrieve sold NFTs

resOpenSea <- GET("https://api.opensea.io/api/v1/events",

query = list(limit=300, #number of events to retrieve

event\_type="successful", #retrieve only the sales

only\_opensea="true")) #retrieve only sales from the opensea website

# Convert the raw unicode (not human friendly) into JSON format

# Don't forget the option flatten=TRUE, otherwise the objects will be a complex list of list of list, impossible to work with

dataOpenSea <- fromJSON(rawToChar(resOpenSea$content), flatten=TRUE)[[1]]

# There are a lot of columns. We have to clean a bit.

# Let's start removing the one containing only NA values

dataOpenSea <- dataOpenSea %>%

select\_if(~!all(is.na(.)))

There is not a lot of explanations on the content of this dataset on the OpenSea website. I thus selected a few columns which seemed to contain interesting information (at least the ones I could understand).

# Let's select a few columns with interesting information

dataOpenSea <- dataOpenSea %>% select("collection\_slug",

"contract\_address",

"id", "quantity",

"payment\_token.name",

"total\_price",

"seller.address",

"transaction.timestamp",

"winner\_account.address",

"payment\_token.usd\_price",

"payment\_token.eth\_price",

"asset.asset\_contract.schema\_name")

#"asset.asset\_contract.address", "asset.asset\_contract.asset\_contract\_type", "asset.asset\_contract.created\_date", "asset.asset\_contract.name"

# Get a glimpse of the data

glimpse(dataOpenSea)

## Rows: 300

## Columns: 12

## $ collection\_slug <chr> "malai-collection", "from-physical-to~

## $ contract\_address <chr> "0x7be8076f4ea4a4ad08075c2508e481d6c9~

## $ id <int> 193766734, 193766381, 193764901, 1937~

## $ quantity <chr> "1", "10", "1", "1", "4", "1", "1", "~

## $ payment\_token.name <chr> "Ether", "Ether", "Ether", "Ether", "~

## $ total\_price <chr> "5000000000000000", "4000000000000000~

## $ seller.address <chr> "0x67b3ba3582292ae12890026904591b0cb8~

## $ transaction.timestamp <chr> "2021-06-06T15:18:20", "2021-06-06T15~

## $ winner\_account.address <chr> "0xc0fdd5d4d5cebd6e9d8f4b28071ff4b334~

## $ payment\_token.usd\_price <chr> "2682.800000000000182000", "2682.8000~

## $ payment\_token.eth\_price <chr> "1.000000000000000", "1.0000000000000~

## $ asset.asset\_contract.schema\_name <chr> "ERC1155", "ERC1155", "ERC1155", "ERC~

Based on my guess, we have:

* collection\_slug: The collection to which the item belong
* contract\_address: All the sales are managed by a contract (a piece of code / a software) which send the NFT to the winner of the bid. This is the address of the OpenSea contract. We see that there is only one address for all the sales, which means that all sales are managed by the same contract.
* id: A unique identifier for each sale
* quantity: The number of items sold per transaction (see fungible / semi fungible below). As in the supermarket, you can buy 1 apple or 20.
* payment\_token.name: The cryptocurrency used to buy the item.
* total\_price: The cost paid by the winner. For Ether, this is expressed in Wei, the smallest denomination of ether. 1 ether = 1,000,000,000,000,000,000 Wei (10^18).
* seller.address: The address of the seller
* transaction.timestamp: Date of the transaction
* winner\_account.address: The address of the buyer
* payment\_token.usd\_price: The price of one token used to make the transaction in USD

Let’s have a look at the distribution of currencies:

dataOpenSea %>%

group\_by(payment\_token.name) %>%

summarise(n=n())

## # A tibble: 6 x 2

## payment\_token.name n

## <chr> <int>

## 1 Decentraland MANA 1

## 2 Ether 278

## 3 Gala 1

## 4 REVV 1

## 5 USD Coin 2

## 6 Wrapped Ether 17

We see that most sales are made in Ether (note that Wrapped Ether can be considered the same as Ether), let’s focus on these Ether sales for the rest of the article.

# Change the format of some columns to something more adapted than character

dataOpenSea <- dataOpenSea %>%

mutate(quantity=as.numeric(quantity),

total\_price=as.numeric(total\_price),

transaction.timestamp=as.Date(transaction.timestamp),

payment\_token.usd\_price=as.numeric(payment\_token.usd\_price))

# filter on sales in ETH

dataOpenSea <- dataOpenSea %>%

filter(payment\_token.name %in% c("Ether", "Wrapped Ether"))

# Convert the price in Ether and then USD. We divide by the quantity as one sale can contain multiple items and then divide by 10^18 to convert the price from Wei to ETH (see above).

dataOpenSea <- dataOpenSea %>% mutate(priceUSD = total\_price / 10^18 \* payment\_token.usd\_price / quantity)

# Make a histogram of the price distribution (with a log scale as prices are quite spread)

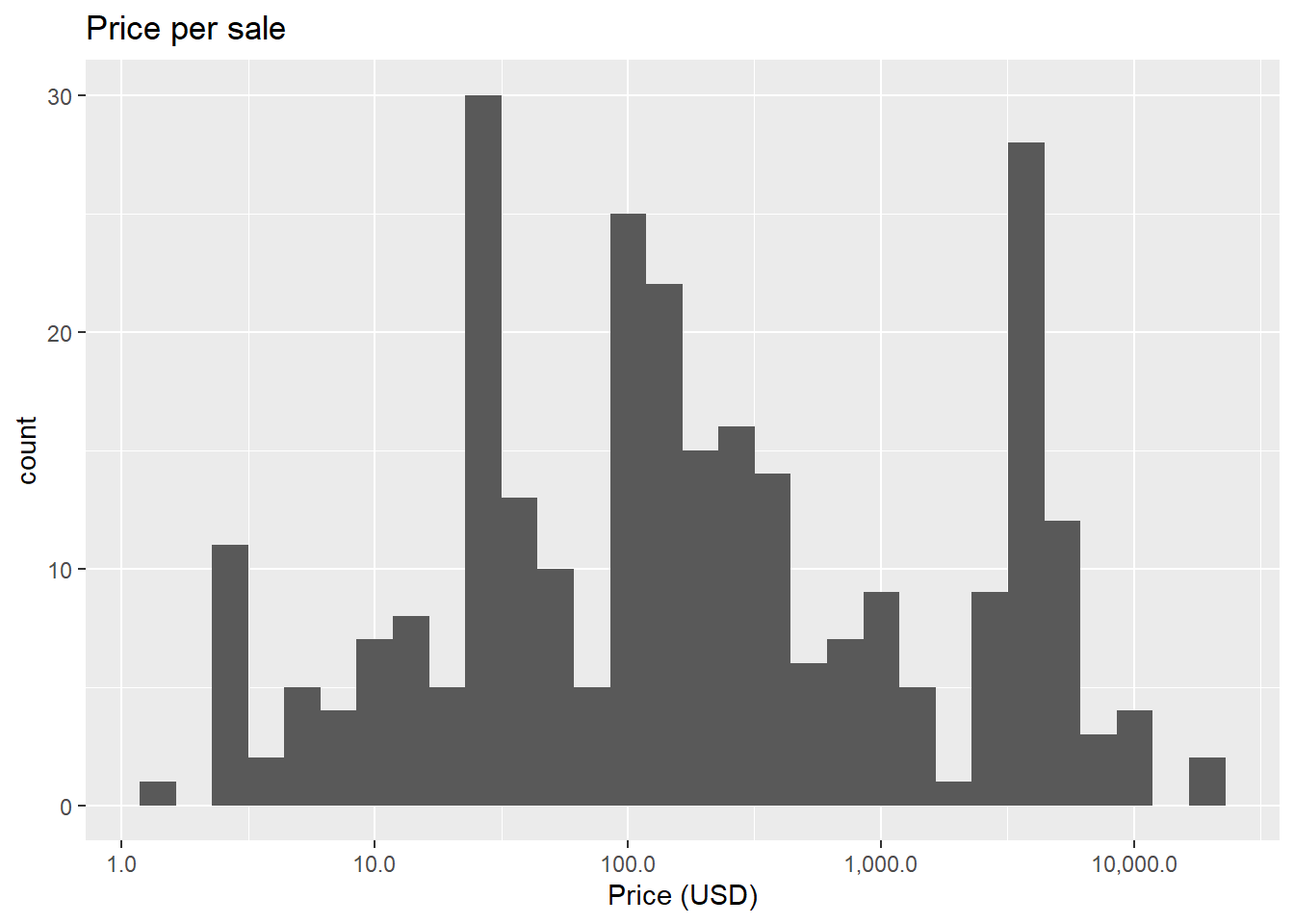
pHistoOpenSea <- ggplot(dataOpenSea, aes(priceUSD)) +

geom\_histogram() +

labs(x="Price (USD)", title="Price per sale") +

scale\_x\_log10(labels = comma)

pHistoOpenSea



# Make a pie chart

dataOpenSea$cut <- cut(dataOpenSea$priceUSD, breaks = c(0, 10, 100, 1000, 10000, 100000, 1000000),

labels = c("0-10USD", "10-100USD", "100-1000USD", "1000-10000USD", "10000-100000USD", "100000-1000000USD"), include.lowest = TRUE)

dataPieChartOpenSea <- dataOpenSea %>%

group\_by(cut) %>%

count() %>%

ungroup() %>%

mutate(percent=`n`/sum(`n`)) %>%

arrange(desc(cut)) %>%

mutate(label=scales::percent(percent))

pPieChartOpenSea <- ggplot(dataPieChartOpenSea, aes(x="", y=percent, fill=cut))+

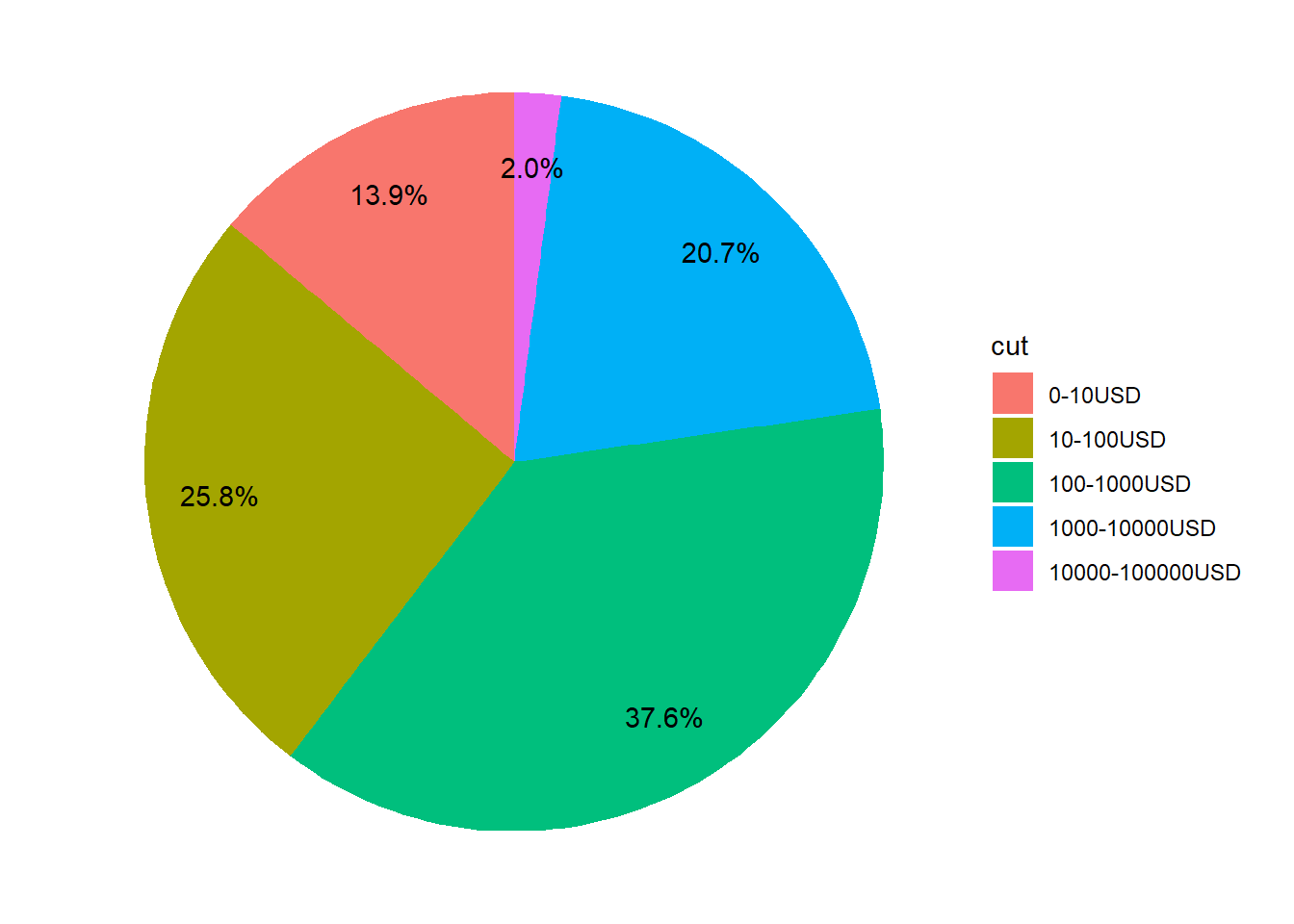
geom\_bar(width = 1, stat = "identity") +

coord\_polar("y", start=0) +

geom\_text(aes(x=1.3, y = cumsum(percent) - percent/2, label=label)) +

theme\_void()

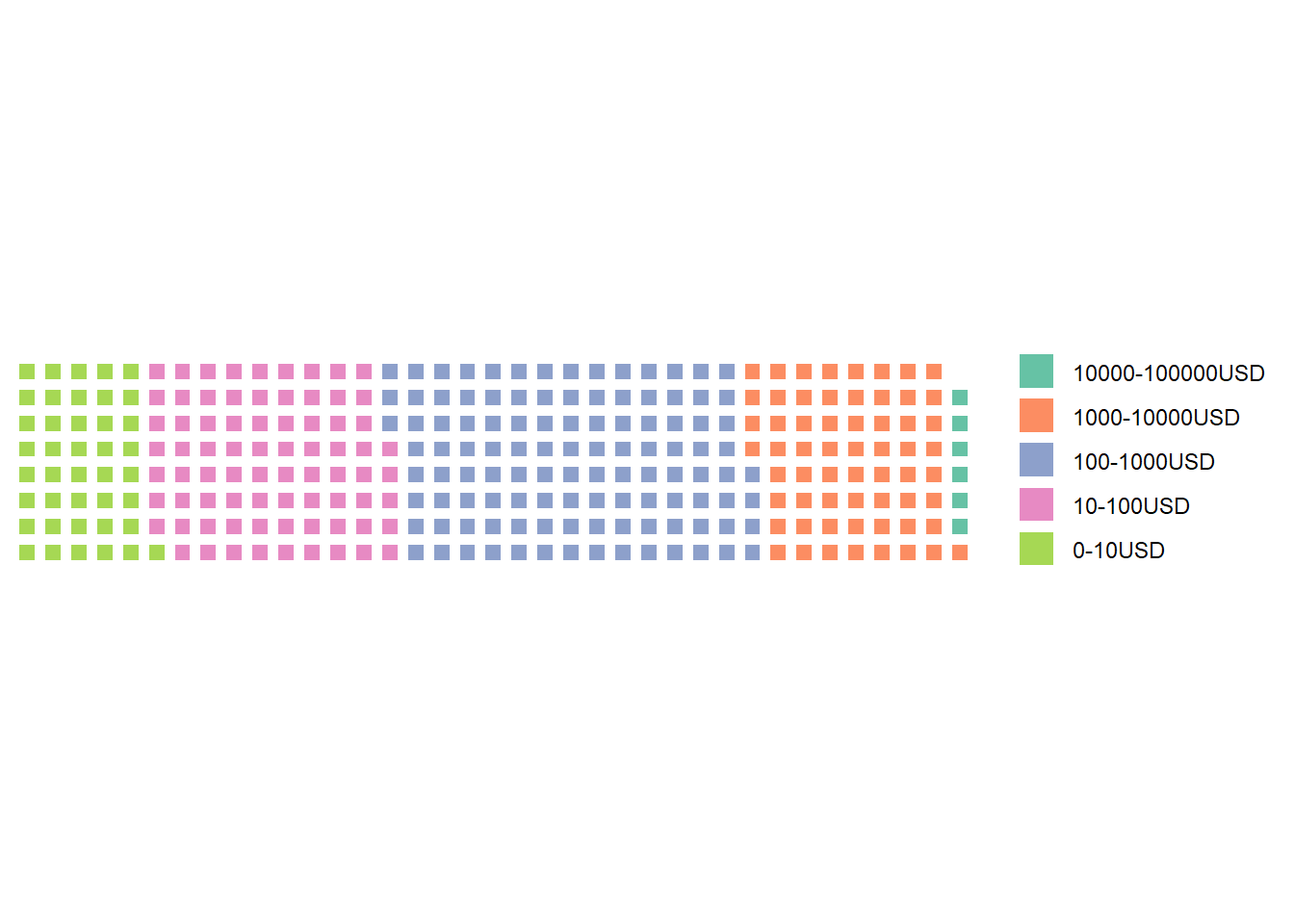
pPieChartOpenSea



# Make a waffle chart

pWaffle <- waffle(dataPieChartOpenSea, rows = 8, reverse=TRUE)

pWaffle



Well, all this is nice but there is a big drawback… OpenSea API limits the number of events to the last 300 transactions. There is not that much we can do about it if we use their API. We saw that retrieving data directly from the blockchain can be quite complex. There are hopefully services like Etherscan which allows you to explore Ethereum Blocks in an easy way and guess what? They also developed an API!

# EtherScan API

EtherScan is a block explorer, which allows users to view information about transactions that have been submitted to the Ethereum blockchain, verify contract code, visualize network data,… We can therefore use it to read transactions about OpenSea. Where do we start? From the data we extracted from OpenSea, we saw that the address of their contract is “0x7be8076f4ea4a4ad08075c2508e481d6c946d12b”. If we enter this address in EtherScan and filter on the completed transaction (i.e. the transactions validated by the network, not the one waiting to be approved), <https://etherscan.io/txs?a=0x7be8076f4ea4a4ad08075c2508e481d6c946d12b>, we see an incredible amount of them (848,965 at the time of writing.) Of course, not all of them are related to a sale. Let’s see what we can do:

# Retrieve the last 10000 transactions (maximum allowed by Etherscan) from the OpenSea contract

resEtherScan <- GET("https://api.etherscan.io/api",

query = list(module="account",

action="txlist",

address="0x7Be8076f4EA4A4AD08075C2508e481d6C946D12b",

sort="desc",

apikey=EtherScanAPIToken))

# Convert the raw unicode (not human friendly) into JSON format

# Don't forget the option flatten=TRUE, otherwise the objects will be a complex list of list of list, impossible to work with

dataEtherScan <- fromJSON(rawToChar(resEtherScan$content), flatten=TRUE)$result

# Change the format of some columns to something more adapted than character

dataEtherScan <- dataEtherScan %>%

mutate(value=as.numeric(value))

# There are many transactions with a value of 0 ether. Among them, some are not directly linked to a sale (actions on the contract such as publishing, maintenance,...) and some are but involve a token transfer (wrapped ETH for instance) instead of ether directly. To keep things simple, let's keep only transactions involving transfer of some ether.

dataEtherScan <- dataEtherScan %>%

filter(value>0)

# Convert ETH price in USD

# For this, we first need to obtain the last USD price

resEtherScanPrice <- GET("https://api.etherscan.io/api",

query = list(module="stats",

action="ethprice",

apikey=EtherScanAPIToken))

dataEtherScanPrice <- fromJSON(rawToChar(resEtherScanPrice$content), flatten=TRUE)$result$ethusd %>% as.numeric()

dataEtherScan <- dataEtherScan %>%

mutate(value=value/10^18) %>% #We divide by 10^18 to convert the price from Wei to ETH (see above).

mutate(priceUSD=value\*dataEtherScanPrice) # convert in USD

# Make a histogram of the price distribution (with a log scale as prices are quite spread)

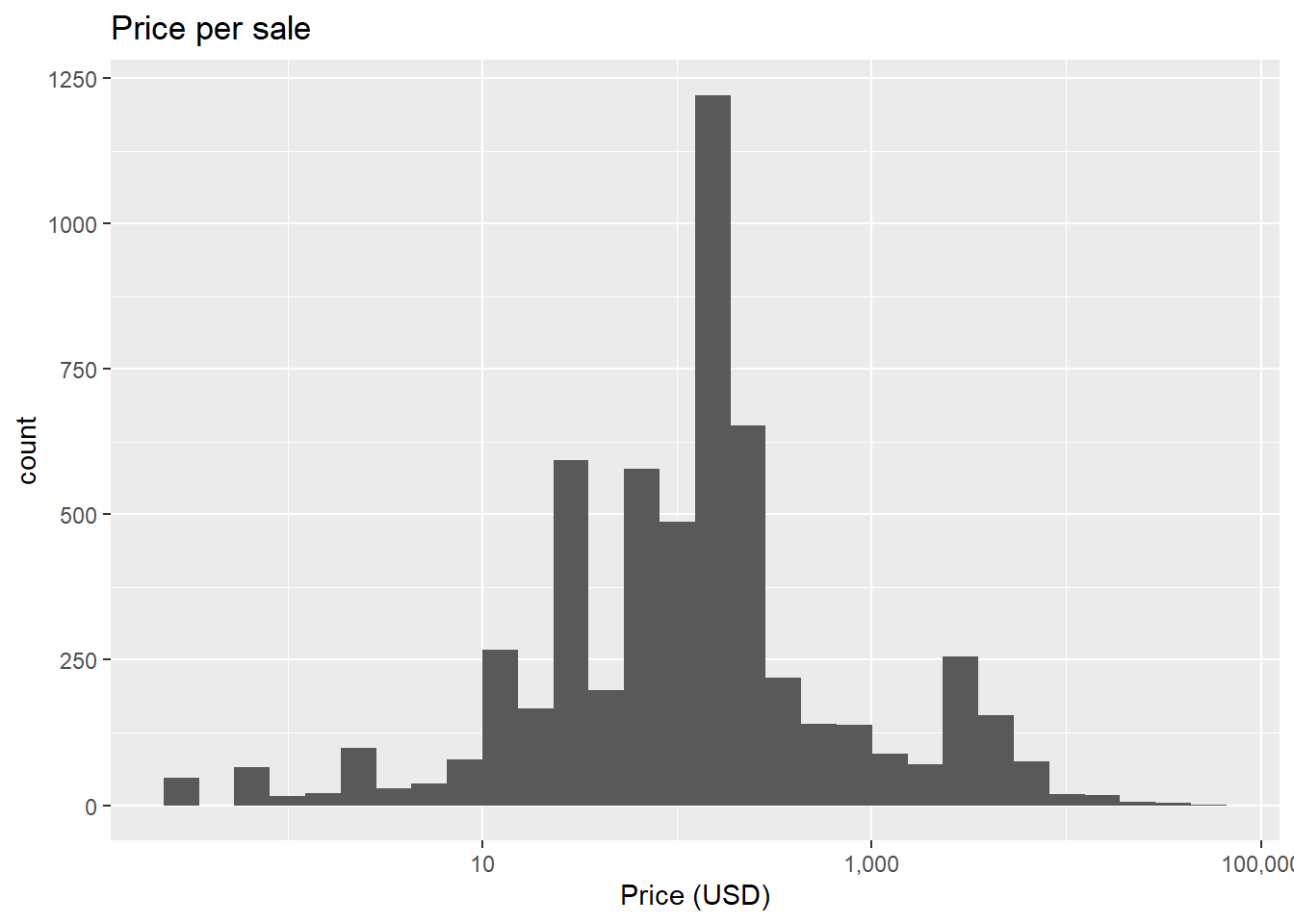
pHistoEtherScan <- ggplot(dataEtherScan, aes(priceUSD)) +

geom\_histogram() +

labs(x="Price (USD)", title="Price per sale") +

scale\_x\_log10(labels = comma)

pHistoEtherScan



# Make a pie chart

dataEtherScan$cut <- cut(dataEtherScan$priceUSD, breaks = c(0, 10, 100, 1000, 10000, 100000, 1000000),

labels = c("0-10USD", "10-100USD", "100-1000USD", "1000-10000USD", "10000-100000USD", "100000-1000000USD"), include.lowest = TRUE)

dataPieChartEtherScan <- dataEtherScan %>%

group\_by(cut) %>%

count() %>%

ungroup() %>%

mutate(percent=`n`/sum(`n`)) %>%

arrange(desc(cut)) %>%

mutate(label=scales::percent(percent))

pPieChartEtherScan <- ggplot(dataPieChartEtherScan, aes(x="", y=percent, fill=cut)) +

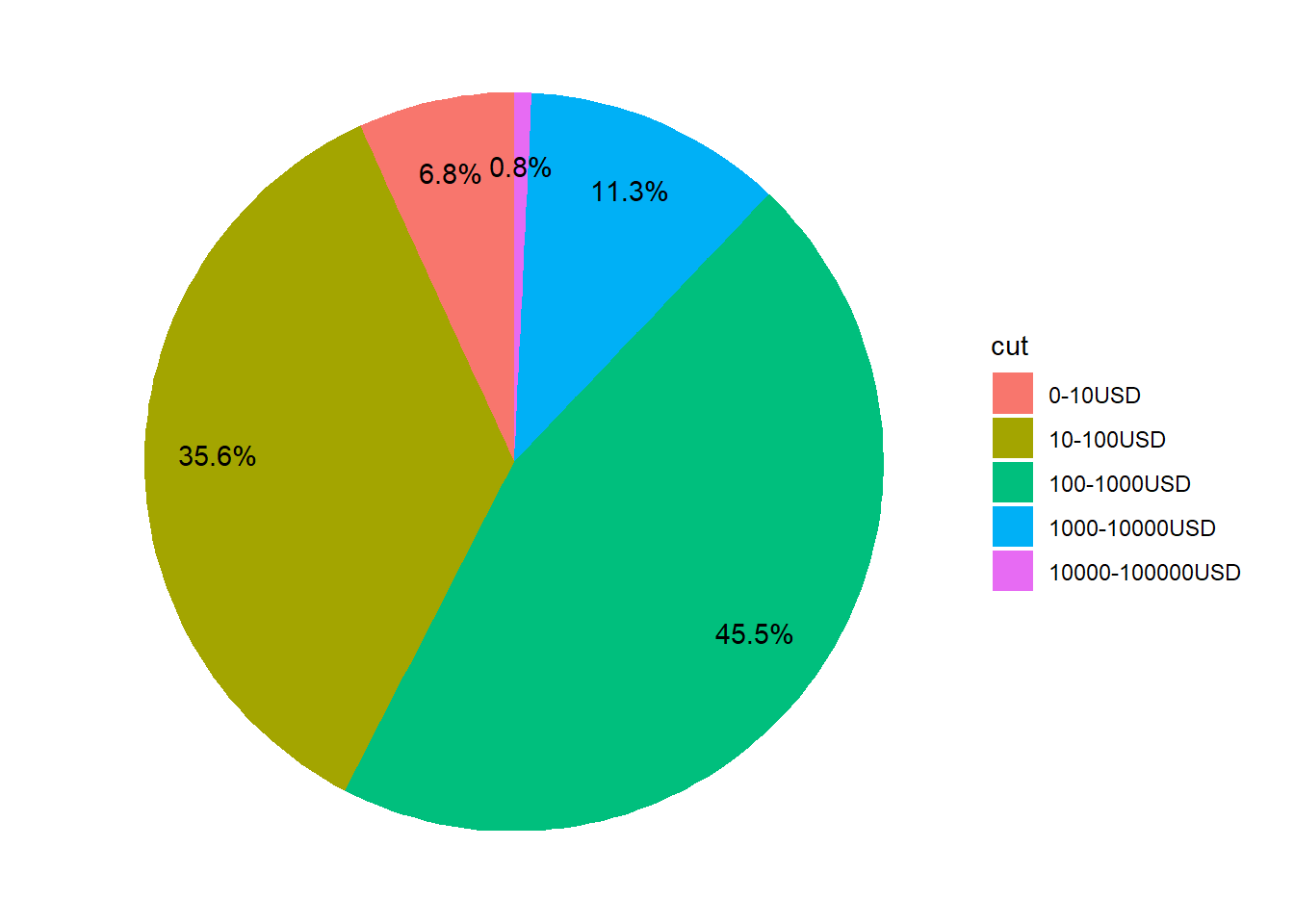
geom\_bar(width = 1, stat = "identity") +

coord\_polar("y", start=0) +

geom\_text(aes(x=1.3, y = cumsum(percent) - percent/2, label=label)) +

theme\_void()

pPieChartEtherScan



# Make a waffle chart

dataPieChartEtherScan <- dataPieChartEtherScan %>%

mutate(n=n/10)

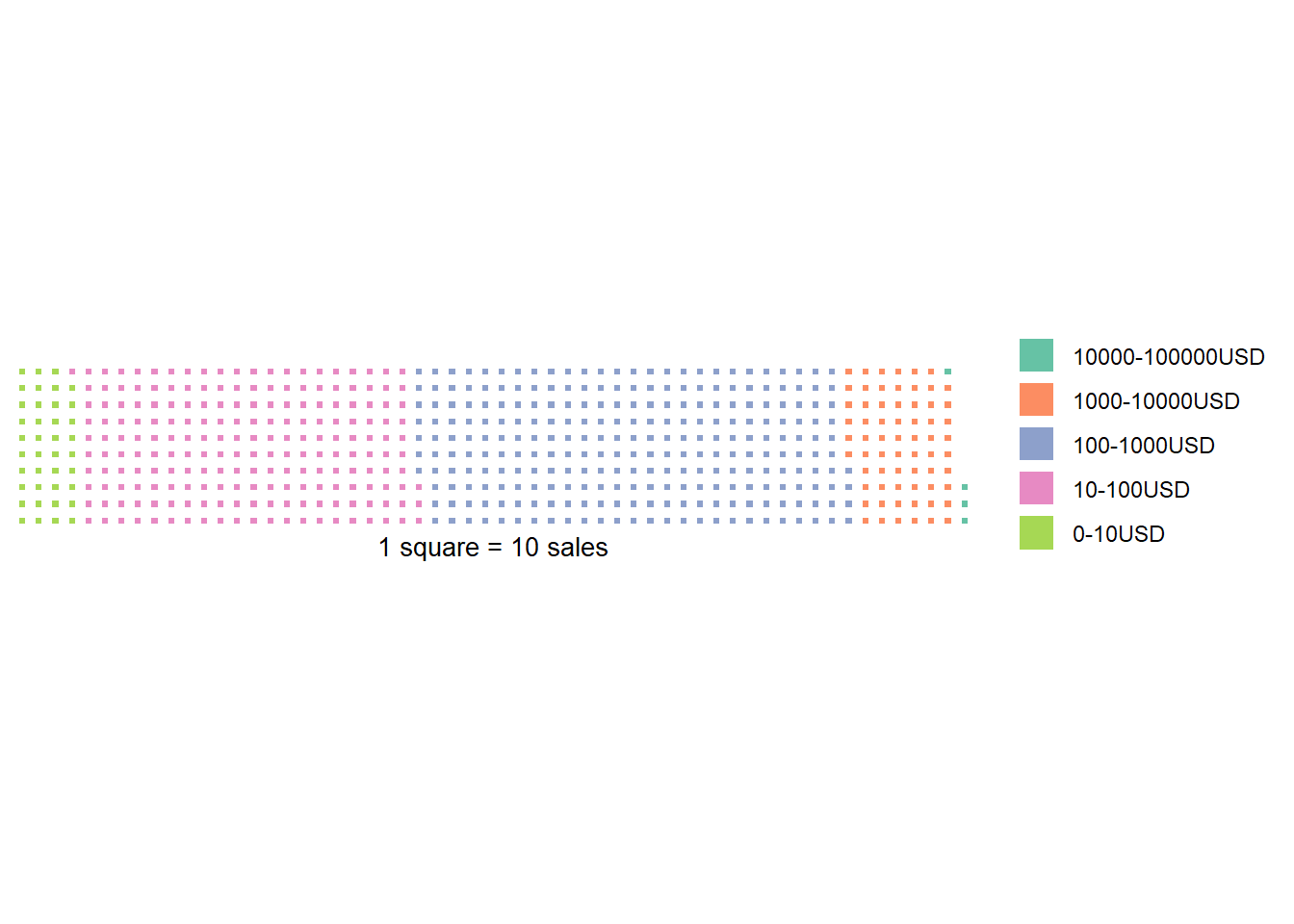
pWaffle <- waffle(dataPieChartEtherScan,

rows = 10,

reverse=TRUE,

xlab = "1 square = 10 sales")

pWaffle



# Conclusion

This article introduced how to read the blockchain and obtain transactions data that can be analyzed and plotted. We made here simple plot to represent sales price. That’s interesting but there a lot more we can do! In Part II, we will investigate how to go further. We can for example follow specific NFTs and see if there are sold after being purchased the first time or if people keep them,… Let me know by which statistics you would be interested!

# References

* <https://docs.opensea.io/reference>
* <https://www.dataquest.io/blog/r-api-tutorial/>
* <https://ethereum.org/en/nft>
* <https://influencermarketinghub.com/nft-marketplaces>
* <https://www.r-bloggers.com/>
* <https://etherscan.io/>
* <https://en.wikipedia.org/wiki/Blockchain>