Final_Project

Nicola Cortinovis, Roberta Lamberti, Marta Lucas, Stefano Tumino

2024-02-22

Introduction

The aim of this project is to predict the price of a smartphone based on its features. The dataset used for this analysis is the mobile phones dataset, which contains 13 columns and 1224 rows:

Variables	Description	Type
X	Index of the phone	int
Brand_Name	Name of the phone brand	chr
Model_Name	Name of the phone model	chr
Os	Operating system	chr
Popularity	The popularity of the phone in range 1-1224	int
Best_Price	Best price of the price-range in (UAH)	num
Lowest_Price	Highest price of the price-range in (UAH)	num
Highest_Price	Lowest price of the price-range in (UAH)	num
Sellers_Amount	The amount sellers sold the phone	num
Screen_Size	The size of phone's screen (inches).	num
Memory_Size	The size of the phone's memory (GB)	num
Battery_Size	The size of the phone's battery (mAh)	num
Release_Date	The launch date of the product on the market	chr

```
df <- read.csv("phones_data.csv", header=T)
head(df)
```

```
##   X brand_name                                    model_name      os popularity
## 1 0    ALCATEL         1 1/8GB Bluish Black (5033D-2JALUAA) Android        422
## 2 1    ALCATEL 1 5033D 1/16GB Volcano Black (5033D-2LALUAF) Android        323
## 3 2    ALCATEL 1 5033D 1/16GB Volcano Black (5033D-2LALUAF) Android        299
## 4 3    ALCATEL 1 5033D 1/16GB Volcano Black (5033D-2LALUAF) Android        287
## 5 4      Nokia                          1.3 1/16GB Charcoal Android       1047
## 6 5      Honor                              10 6/64GB Black Android         71
##   best_price lowest_price highest_price sellers_amount screen_size memory_size
## 1       1690         1529          1819             36        5.00           8
## 2       1803         1659          2489             36        5.00          16
## 3       1803         1659          2489             36        5.00          16
## 4       1803         1659          2489             36        5.00          16
## 5       1999           NA            NA             10        5.71          16
## 6      10865        10631         11099              2        5.80          64
##   battery_size release_date
## 1         2000      10-2020
## 2         2000       9-2020
## 3         2000       9-2020
```

```
## 4          2000       9-2020
## 5          3000       4-2020
## 6          3400       6-2018
```

Specifically, our objective is to predict the `best_price` variable. Our approach consists of the following steps:

Data exploration

Data preprocessing

Firstly we remove from the dataset the index column `X`

```
df$X <- NULL
```

Our next step is to briefly explore the `chr` variables and transform the appropriate ones into factors. - `model_name` has not been transformed into a factor because it has too many levels, almost a unique model_name for each row; - `brand_name` has not been transformed into a factor obtaining 64 classes; - `os` has not been transformed into a factor obtaining 3 classes; - `release_date` has not been transformed into a factor because it has been used to create two new variables: `month` and `year`.

```
length(unique(df$model_name))
```

```
## [1] 1068
```

```
df$brand_name <- factor(df$brand_name)
df$os <- factor(df$os)
```

```
df$month <- as.numeric(sapply(df$release_date, FUN = function(x) {strsplit(x, split = '[-]')[[1]][1]}))
df$year <- as.numeric(sapply(df$release_date, FUN = function(x) {strsplit(x, split = '[-]')[[1]][2]}))-1
```

For clarity's sake we convert the ukrainian currency (UAH) into euros (€) and rename the blank " " `os` class as "other".

```
df$best_price <- df$best_price*0.024
df$lowest_price <- df$lowest_price*0.024
df$highest_price <- df$highest_price*0.024

levels(df$os)[1] <- "other"
```

```
summary(df)
```

```
##         brand_name   model_name                   os        popularity
##  Samsung     :168   Length:1224        other       :197   Min.   :   1.0
##  Xiaomi      :111   Class :character   Android     :915   1st Qu.: 306.8
##  Apple       :102   Mode  :character   EMUI        :  2   Median : 612.5
##  Motorola    : 62                      iOS         :103   Mean   : 612.5
##  Sigma mobile: 52                      KAIOS       :  1   3rd Qu.: 918.2
##  HUAWEI      : 49                      OxygenOS    :  3   Max.   :1224.0
##  (Other)     :680                      WindowsPhone:  3
##    best_price        lowest_price      highest_price      sellers_amount
##  Min.   :   5.136   Min.   :   4.752   Min.   :   5.496   Min.   :  1.00
##  1st Qu.:  62.394   1st Qu.:  57.576   1st Qu.:  69.288   1st Qu.:  2.00
##  Median : 113.472   Median : 109.776   Median : 127.812   Median :  8.00
##  Mean   : 190.589   Mean   : 185.184   Mean   : 237.202   Mean   : 16.74
##  3rd Qu.: 223.752   3rd Qu.: 222.294   3rd Qu.: 304.170   3rd Qu.: 26.00
##  Max.   :1345.968   Max.   :1199.976   Max.   :1679.976   Max.   :125.00
##                     NA's   :260        NA's   :260
##   screen_size     memory_size        battery_size   release_date
```

```
##  Min.   :1.400   Min.   :3.20e-03   Min.   :  460   Length:1224
##  1st Qu.:5.162   1st Qu.:3.20e+01   1st Qu.: 2900   Class :character
##  Median :6.000   Median :6.40e+01   Median : 3687   Mode  :character
##  Mean   :5.394   Mean   :9.57e+01   Mean   : 3608
##  3rd Qu.:6.400   3rd Qu.:1.28e+02   3rd Qu.: 4400
##  Max.   :8.100   Max.   :1.00e+03   Max.   :18800
##  NA's   :2       NA's   :112        NA's   :10
##      month            year
##  Min.   : 1.000   Min.   :13.00
##  1st Qu.: 4.000   1st Qu.:18.00
##  Median : 8.000   Median :19.00
##  Mean   : 7.016   Mean   :18.85
##  3rd Qu.:10.000   3rd Qu.:20.00
##  Max.   :12.000   Max.   :21.00
##
```

From the summary we notice the presence of some NAs in the `battery_size`, `screen_size`, `memory_size`, `lowest_price` and `highest_price` variables. Given the small amount of NAs in the first two we decide to remove the rows with NAs. Further investigation into the `memory_size` shows that the NAs are present only for the "other" `os` class, so we decide to fill them with the median of the `memory_size` for the "other" `os` class. The choice of the median is dictated by the presence of a few outliers.

```
df <- df[- which(is.na(df$battery_size)),]
df <- df[- which(is.na(df$screen_size)),]
```

```
tmp <- df[which(df$os == "other"),]$memory_size
df$memory_size[which(is.na(df$memory_size))] <- median(tmp[-which(is.na(tmp))])
```

The variables `lowest_price` and `highest_price` also show some outliers so their NAs have been filled with their median.

```
tmp <- which(is.na(df$lowest_price))
df$lowest_price[tmp] <- median(df$lowest_price[-tmp])
tmp <- which(is.na(df$highest_price))
df$highest_price[tmp] <- median(df$highest_price[-tmp])

summary(df)
```

```
##         brand_name   model_name                  os        popularity
##  Samsung     :168   Length:1212        other       :196   Min.   :   1.0
##  Xiaomi      :111   Class :character   Android     :910   1st Qu.: 305.8
##  Apple       : 96   Mode  :character   EMUI        :  2   Median : 610.5
##  Motorola    : 62                      iOS         : 97   Mean   : 611.7
##  Sigma mobile: 51                      KAIOS       :  1   3rd Qu.: 918.2
##  HUAWEI      : 48                      OxygenOS    :  3   Max.   :1224.0
##  (Other)     :676                      WindowsPhone:  3
##    best_price        lowest_price      highest_price      sellers_amount
##  Min.   :   5.136   Min.   :   4.752   Min.   :   5.496   Min.   :  1.00
##  1st Qu.:  62.370   1st Qu.:  72.264   1st Qu.:  83.976   1st Qu.:  2.00
##  Median : 113.160   Median : 108.468   Median : 127.176   Median :  8.00
##  Mean   : 189.163   Mean   : 167.963   Mean   : 211.801   Mean   : 16.65
##  3rd Qu.: 216.996   3rd Qu.: 167.976   3rd Qu.: 210.150   3rd Qu.: 25.00
##  Max.   :1328.112   Max.   :1099.176   Max.   :1559.976   Max.   :125.00
##
##   screen_size     memory_size         battery_size   release_date
##  Min.   :1.400   Min.   :   0.0032   Min.   :  460   Length:1212
```

```
##  1st Qu.:5.200   1st Qu.:  16.0000   1st Qu.: 2900   Class :character
##  Median :6.000   Median :  64.0000   Median : 3687   Mode  :character
##  Mean   :5.396   Mean   :  86.5264   Mean   : 3610
##  3rd Qu.:6.400   3rd Qu.: 128.0000   3rd Qu.: 4400
##  Max.   :8.100   Max.   :1000.0000   Max.   :18800
##
##      month            year
##  Min.   : 1.000   Min.   :13.00
##  1st Qu.: 4.000   1st Qu.:18.00
##  Median : 8.000   Median :19.00
##  Mean   : 7.001   Mean   :18.86
##  3rd Qu.:10.000   3rd Qu.:20.00
##  Max.   :12.000   Max.   :21.00
##
```

The dataset contains several duplicates, where phones share the same characteristics but have different popularity levels. Therefore, we eliminate these duplicate observations as they provide redundant information. This process involves retaining only the first occurrence of each duplicate and replacing its popularity with the average popularity of the duplicates. An example is given below.

```
df[2:4,]
```

```
##   brand_name                                    model_name      os popularity
## 2    ALCATEL 1 5033D 1/16GB Volcano Black (5033D-2LALUAF) Android        323
## 3    ALCATEL 1 5033D 1/16GB Volcano Black (5033D-2LALUAF) Android        299
## 4    ALCATEL 1 5033D 1/16GB Volcano Black (5033D-2LALUAF) Android        287
##   best_price lowest_price highest_price sellers_amount screen_size memory_size
## 2     43.272       39.816        59.736             36           5          16
## 3     43.272       39.816        59.736             36           5          16
## 4     43.272       39.816        59.736             36           5          16
##   battery_size release_date month year
## 2         2000       9-2020     9   20
## 3         2000       9-2020     9   20
## 4         2000       9-2020     9   20
```

```
# Find the indices of duplicate rows
idxs <- which(duplicated(df[,-c(2, 4)]))

# Check if each index is succeeded by the next one in the sequence
succ <- c(idxs[-1] - idxs[-length(idxs)] == 1, FALSE)

i = 1
while (i <= length(idxs)){
  start = idxs[i]
  sum <- c(df$popularity[start])
  while (succ[i]){
    i = i + 1
    sum <- c(sum, df$popularity[idxs[i]])
  }
  df$popularity[start] <- mean(sum)
  i = i + 1
}

# Remove the duplicate rows
df <- df[-idxs, ]
```

```r
# Remove the model_name column
df$model_name <- NULL
```

The `popularity` variable is unique for each row, therefore we decided to create a new variable `popularity_levels` which divides the popularity into 4 classes: "low", "medium", "high" and "very high" based on the quartiles of the `popularity` variable.

```r
df$popularity <- as.numeric(df$popularity)

tag <- quantile(df$popularity)

df$popularity_levels <- cut(df$popularity, breaks = tag,
labels=c("low", "medium", "high", "very high"), include.lowest=TRUE)

df$popularity <- NULL
```

Data visualization

```r
corr <- cor(df[, c("battery_size", "memory_size", "screen_size", "best_price", "highest_price", "lowest_

ggcorrplot(corr, hc.order = TRUE, lab = TRUE, colors=c("yellow", "white", "purple"))
```

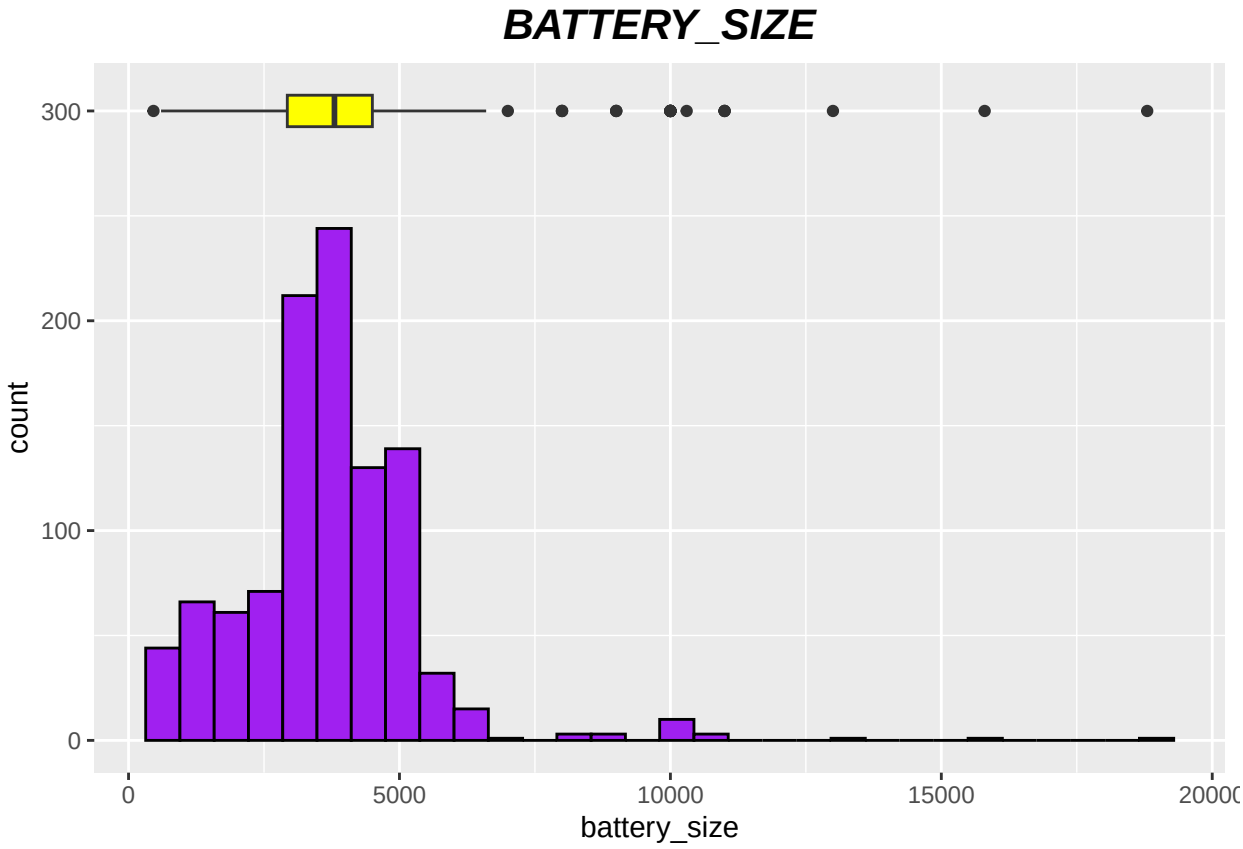
From the correlation plot, we observe a strong positive linear correlation between best_price, highest and lowest price. For this reason we don't consider them in our analysis.

```r
df$highest_price <- NULL
df$lowest_price <- NULL
```

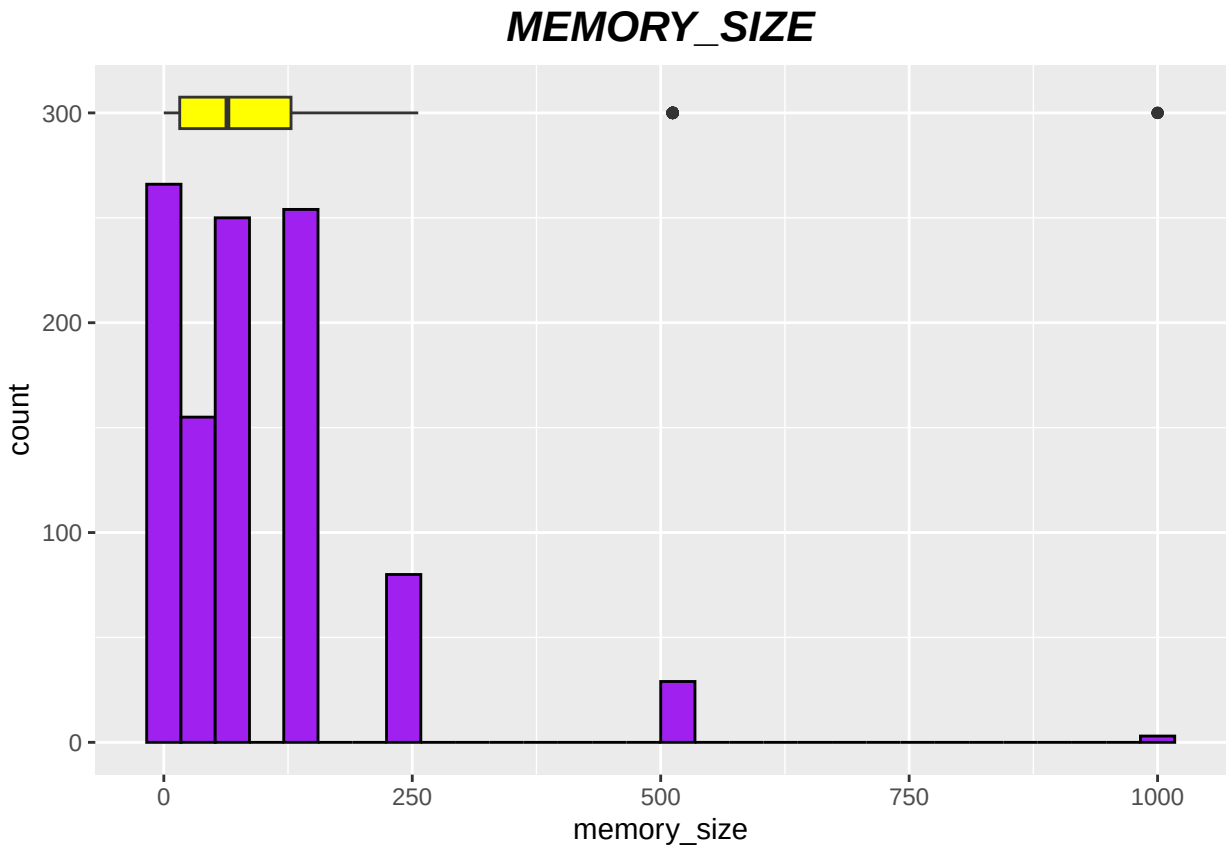
```
cols <- c("battery_size", "memory_size", "screen_size", "best_price", "sellers_amount")

for (i in cols) {
  p1 <- ggplot(df, aes_string(x=i)) + geom_boxplot(fill="yellow", width= 15, position = position_nudge(y
  geom_histogram(fill="purple", color = "black") + ggtitle(toupper(i)) + theme(plot.title = element_text
  plot(p1)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

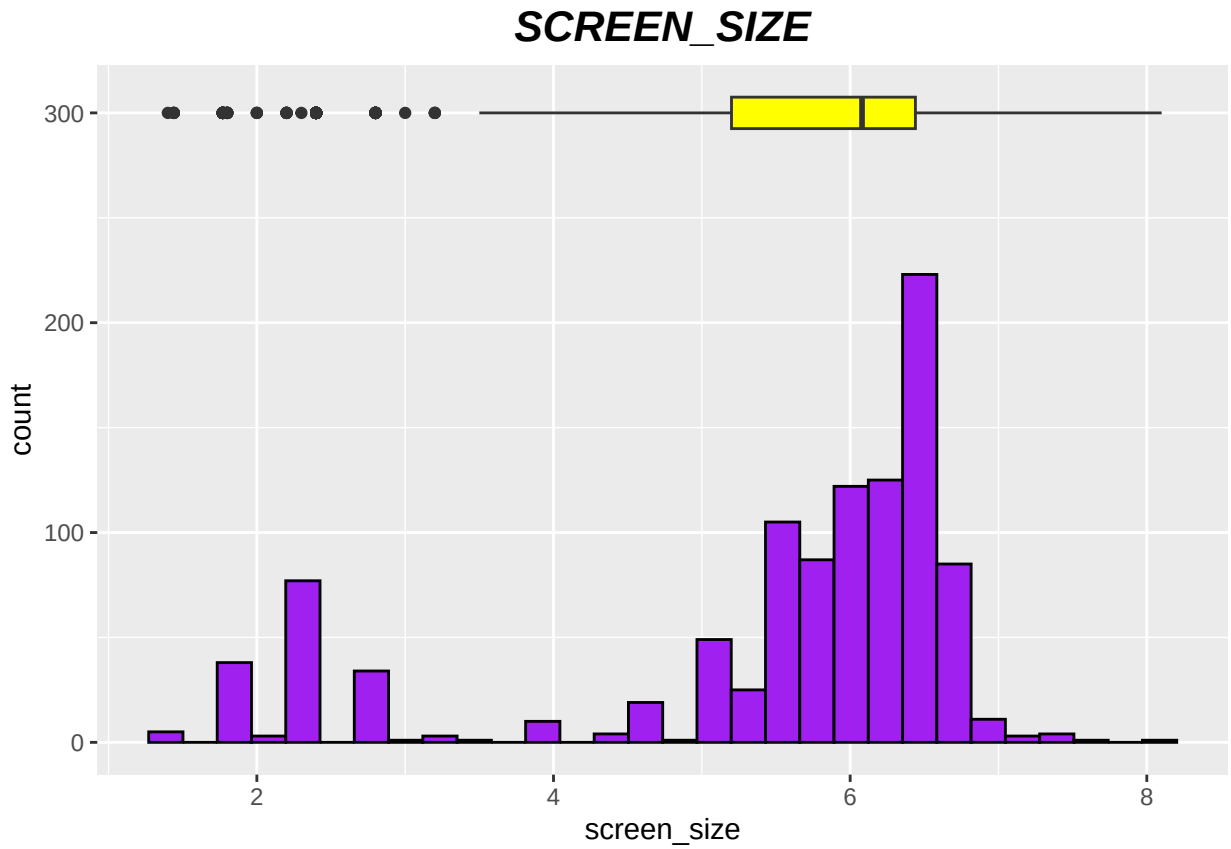
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

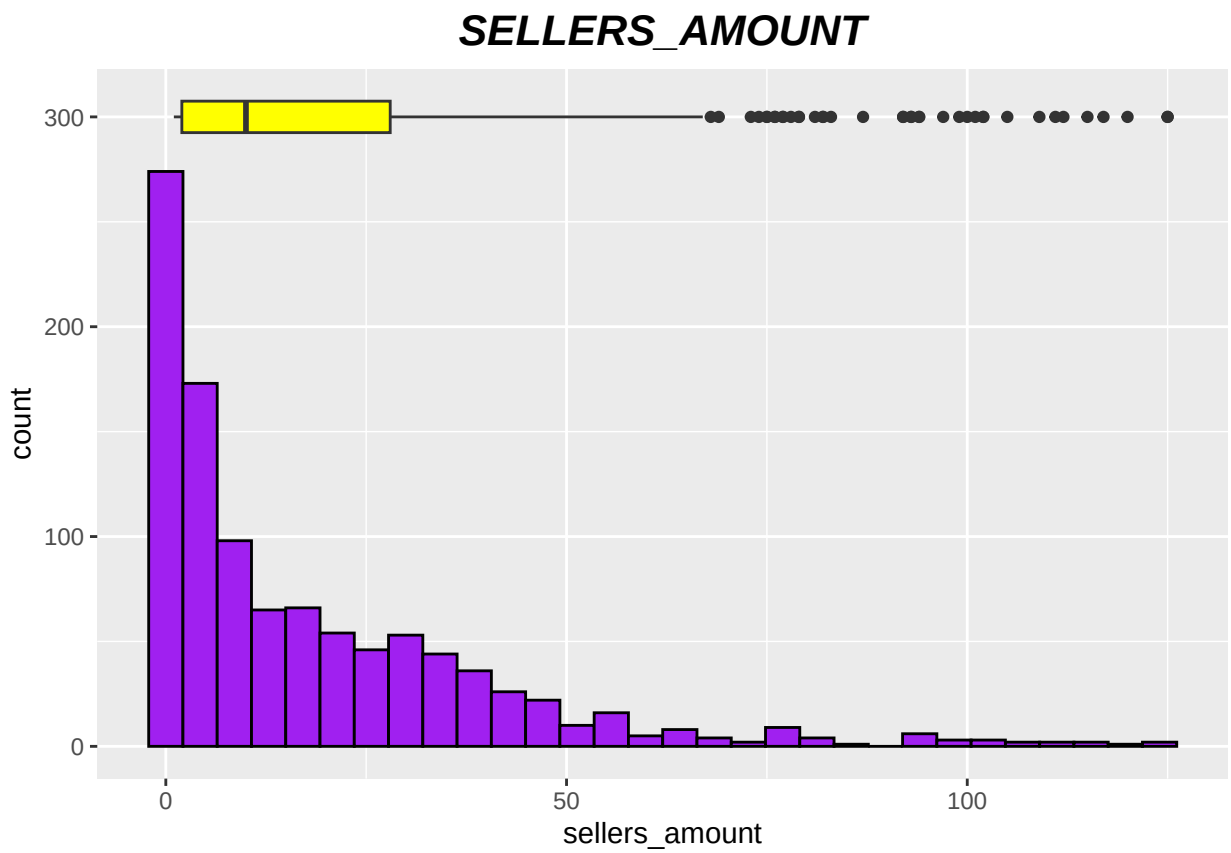



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


SELLERS_AMOUNT
300
200
100
0
count
0
50
100
sellers_amount