Exploratory Data Analysis

Matheus C. Andrade

6/15/2021

Stage 0: Collecting the data

```
## Setting the working directory
setwd("~/Development/DataScienceAcademy/FCD/BigDataRAzure/ProjetoFinal/TalkingData-AdTracking-Fraud-Dete
getwd()
```

```
## [1] "/home/matheus/Development/DataScienceAcademy/FCD/BigDataRAzure/ProjetoFinal/TalkingData-AdTrack
```

```
## Libraries
library(data.table)
library(Amelia)
```

```
## Loading required package: Rcpp
```

```
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.8.0, built: 2021-05-26)
## ## Copyright (C) 2005-2021 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
```

```
##     combine
```

```r
library(corrplot)
```

```
## corrplot 0.88 loaded
```

```r
## Loading dataset
df_original <- fread("../Datasets/train_sample.csv", header=T)
# df <- fread(file.choose(), header=T)
df <- df_original
```

Stage 1: Knowing the data as they are

```r
str(df) #glipmse(df)
```

```
## Classes 'data.table' and 'data.frame':   100000 obs. of  8 variables:
##  $ ip             : int  87540 105560 101424 94584 68413 93663 17059 121505 192967 143636 ...
##  $ app            : int  12 25 12 13 12 3 1 9 2 3 ...
##  $ device         : int  1 1 1 1 1 1 1 1 2 1 ...
##  $ os             : int  13 17 19 13 1 17 17 25 22 19 ...
##  $ channel        : int  497 259 212 477 178 115 135 442 364 135 ...
##  $ click_time     : POSIXct, format: "2017-11-07 09:30:38" "2017-11-07 13:40:27" ...
##  $ attributed_time: POSIXct, format: NA NA ...
##  $ is_attributed  : int  0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
dim(df)
```

```
## [1] 100000      8
```

```r
summary(df)
```

```
##        ip              app             device              os
##  Min.   :     9   Min.   :  1.00   Min.   :   0.00   Min.   :  0.00
##  1st Qu.: 40552   1st Qu.:  3.00   1st Qu.:   1.00   1st Qu.: 13.00
##  Median : 79827   Median : 12.00   Median :   1.00   Median : 18.00
##  Mean   : 91256   Mean   : 12.05   Mean   :  21.77   Mean   : 22.82
##  3rd Qu.:118252   3rd Qu.: 15.00   3rd Qu.:   1.00   3rd Qu.: 19.00
##  Max.   :364757   Max.   :551.00   Max.   :3867.00   Max.   :866.00
##
##     channel        click_time                    attributed_time
##  Min.   :  3.0   Min.   :2017-11-06 16:00:00   Min.   :2017-11-06 17:19:04
##  1st Qu.:145.0   1st Qu.:2017-11-07 11:34:09   1st Qu.:2017-11-07 11:50:27
##  Median :258.0   Median :2017-11-08 07:07:50   Median :2017-11-08 06:43:39
##  Mean   :268.8   Mean   :2017-11-08 06:29:52   Mean   :2017-11-08 07:04:12
##  3rd Qu.:379.0   3rd Qu.:2017-11-09 02:06:01   3rd Qu.:2017-11-09 01:42:52
##  Max.   :498.0   Max.   :2017-11-09 15:59:51   Max.   :2017-11-09 15:28:15
##                                                NA's   :99773
##  is_attributed
##  Min.   :0.00000
##  1st Qu.:0.00000
##  Median :0.00000
##  Mean   :0.00227
##  3rd Qu.:0.00000
##  Max.   :1.00000
##
```

```
## Duplicated rows analysis
any(duplicated(df))
```

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

```
# what are this rows?
df[duplicated(df), ] #df %>% !distinct() # There is one
```

```
##     ip app device os channel          click_time attributed_time is_attributed
## 1: 871  12      1 13     178 2017-11-08 10:00:05            <NA>             0
```

```
# removing duplicated rows
df <- df[!duplicated(df), ]
any(duplicated(df))
```

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

```
## quantity of null values in each columns
any(is.na(df))
```

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

```
# missmap(df, main = "Missing Values Map", col = c("red", "black"), legend = FALSE)
sapply(df, function(x) sum(is.na(x)))
```

```
##              ip             app          device              os         channel
##               0               0               0               0               0
##      click_time attributed_time   is_attributed
##               0           99772               0
```

```
# just 'attributed_time' has null values: 99773
sum(is.na(df[,"attributed_time"]))/length(df$attributed_time)*100
```

```
## [1] 99.773
```

```
# as for the users who didn't download the app, the time has not been recorded
# and the column wasn't filled with any value

## quantity of unique values in each columns
sapply(df, function(x) length(unique(x)))
```

```
##              ip             app          device              os         channel
##           34857             161             100             130             161
##      click_time attributed_time   is_attributed
##           80350             228               2
```

```
# the label column is a factor with two levels
df$is_attributed <- as.factor(df$is_attributed)

# "ip", "app", "device", "os", "channel" are also factor type
df[,1:5] <- lapply(df[,1:5], factor)#######
#str(df)

## High class imbalance problem
summary(df$is_attributed) #table(df$is_attributed)
```

```
##     0     1
## 99772   227
```

```r
prop.table(table(df$is_attributed))*100
```

```
##
##          0          1
## 99.7729977  0.2270023
```

```r
# it means that the models will be overfitted about no-downloads

## when "attributed_time" is NULL "is_attributed" is 0.
# there isn't attributed_time when there wasn't made download

## Generating weekday and hour from click_time intending to explore days and hours
# 'weekdays' to names, 'wday' to numbers (it starts with 0 = Sunday)
df$weekday <- weekdays(df$click_time)
df$hour <- hour(df$click_time)
#glimpse(df)
# quantity of unique values in the new columns
sapply(df[, 9:10], function(x) length(unique(x)))
```

```
## weekday    hour
##       4      24
```

```r
## changing the columns order
# names(df)
df <- df[, c(6,9,10,1,2,3,4,5,7,8)]

# changing for factor the new columns
df$weekday <- as.factor(df$weekday)
df$hour <- as.factor(df$hour)

# after data munging, just confirming whether the data integrity is as before
sapply(df, function(x) sum(is.na(x)))
```

```
##      click_time         weekday            hour              ip             app
##               0               0               0               0               0
##          device              os         channel attributed_time   is_attributed
##               0               0               0           99772               0
```

```r
sapply(df, function(x) length(unique(x)))
```

```
##      click_time         weekday            hour              ip             app
##           80350               4              24           34857             161
##          device              os         channel attributed_time   is_attributed
##             100             130             161             228               2
```

```r
## Creating subsets from is_attributed classes
df_IsAttributed0 <- df %>%
  filter(is_attributed == '0')

df_IsAttributed1 <- df %>%
  filter(is_attributed == '1') %>%
  mutate(wday_IsAttributed1 = weekdays(attributed_time),
         hour_IsAttributed1 = hour(attributed_time))

# as attributed_time represents the event time we want predict, it's deleted
df$attributed_time = NULL
```

```r
summary(df)
```

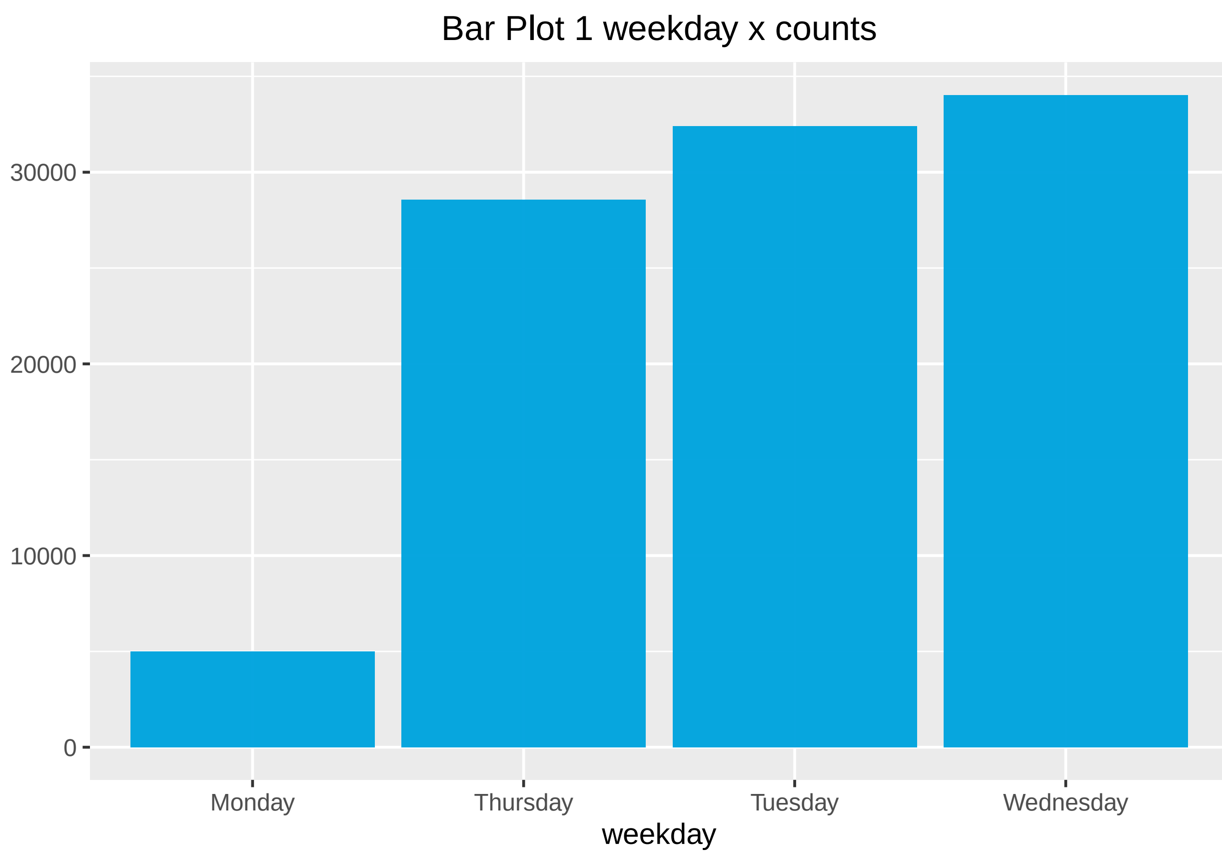
```
##    click_time                        weekday           hour      
##  Min.   :2017-11-06 16:00:00   Monday   : 5011   4      : 6039  
##  1st Qu.:2017-11-07 11:34:09   Thursday :28561   0      : 5654  
##  Median :2017-11-08 07:07:49   Tuesday  :32393   13     : 5619  
##  Mean   :2017-11-08 06:29:52   Wednesday:34034   14     : 5561  
##  3rd Qu.:2017-11-09 02:06:01                     10     : 5510  
##  Max.   :2017-11-09 15:59:51                     5      : 5400  
##                                                  (Other):66216  
##        ip             app            device            os       
##  5348   :  669   3      :18279   1      :94337   19     :23870  
##  5314   :  616   12     :13197   2      : 4345   13     :21222  
##  73487  :  439   2      :11737   0      :  541   17     : 5232  
##  73516  :  399   9      : 8992   3032   :  371   18     : 4830  
##  53454  :  280   15     : 8595   3543   :  151   22     : 4039  
##  114276 :  219   18     : 8315   3866   :   93   10     : 2816  
##  (Other):97377   (Other):30884   (Other):  161   (Other):37990  
##     channel      is_attributed
##  280    : 8114   0:99772      
##  245    : 4802   1:  227      
##  107    : 4543                
##  477    : 3960                
##  134    : 3224                
##  259    : 3130                
##  (Other):72226                
```

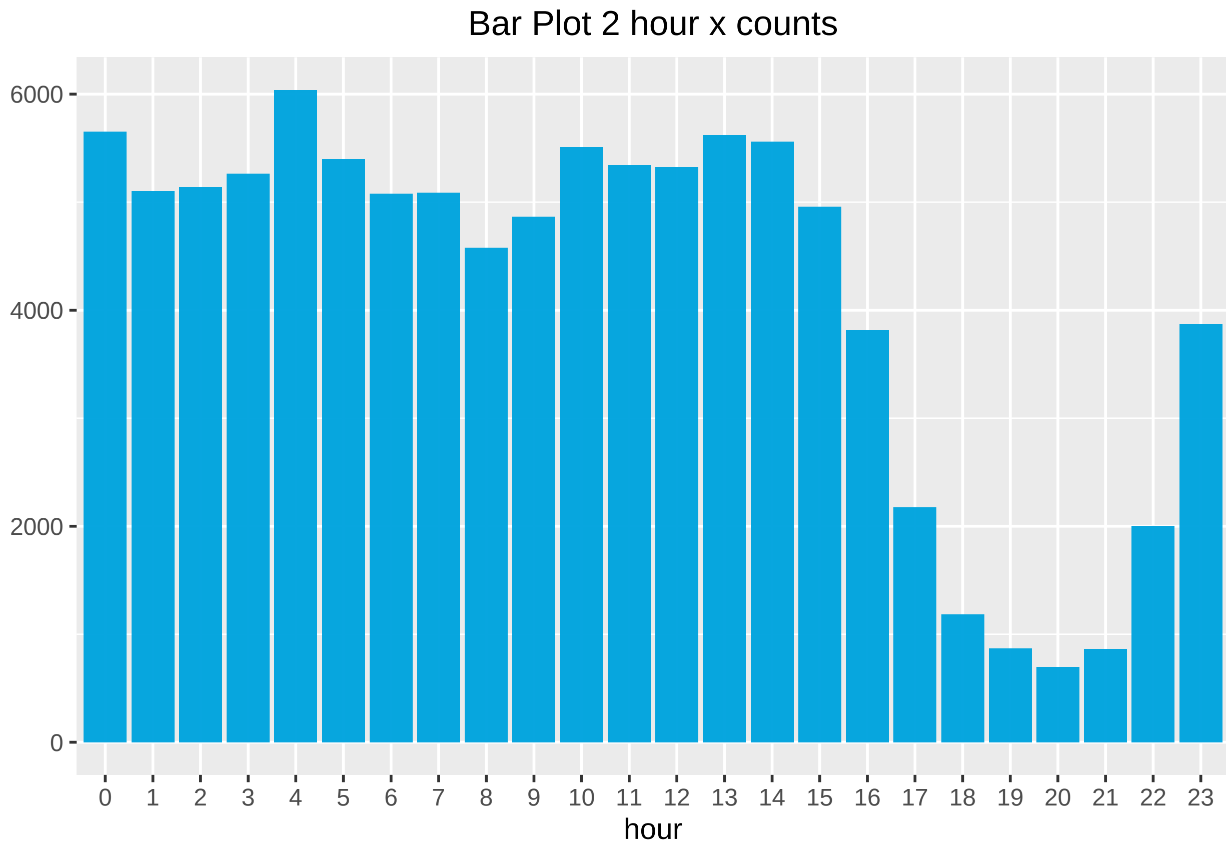
Stage 2: Exploratory Data Analysis

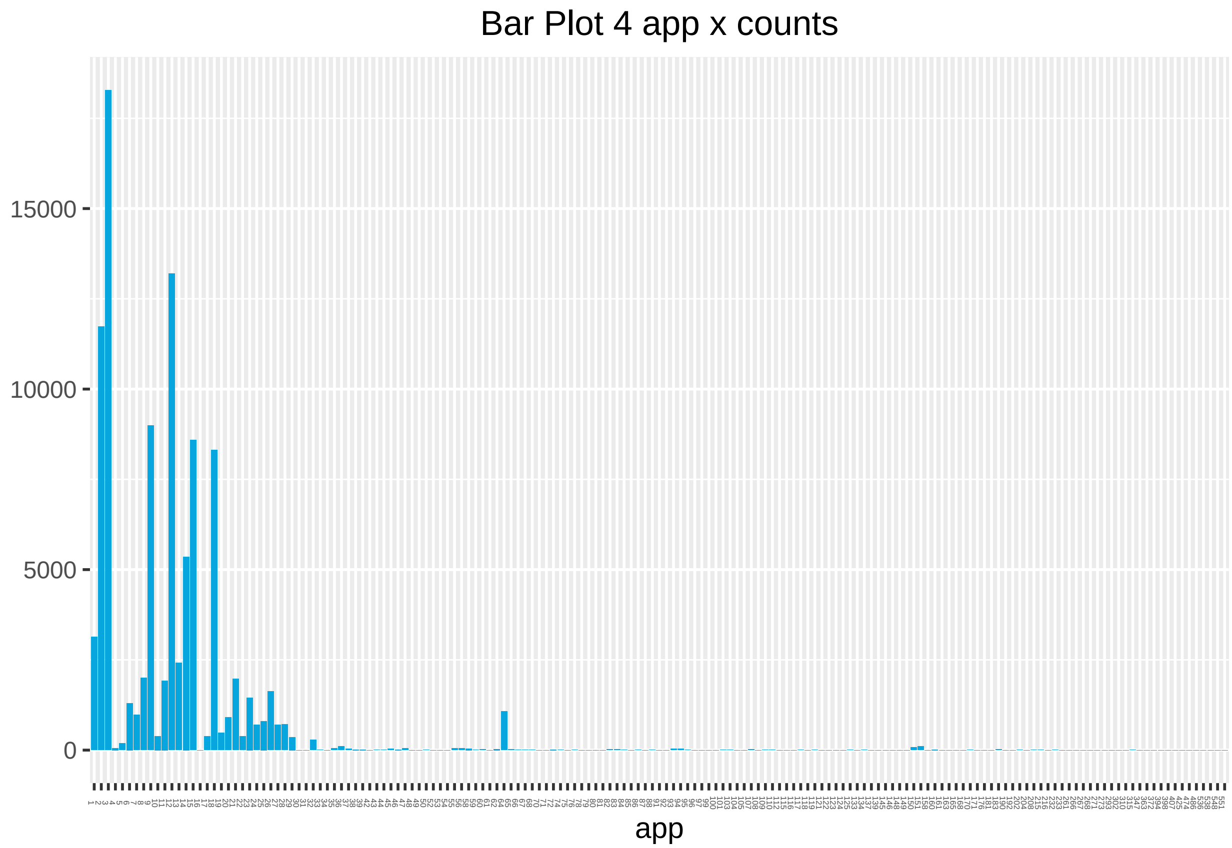
2.1 Bar plots

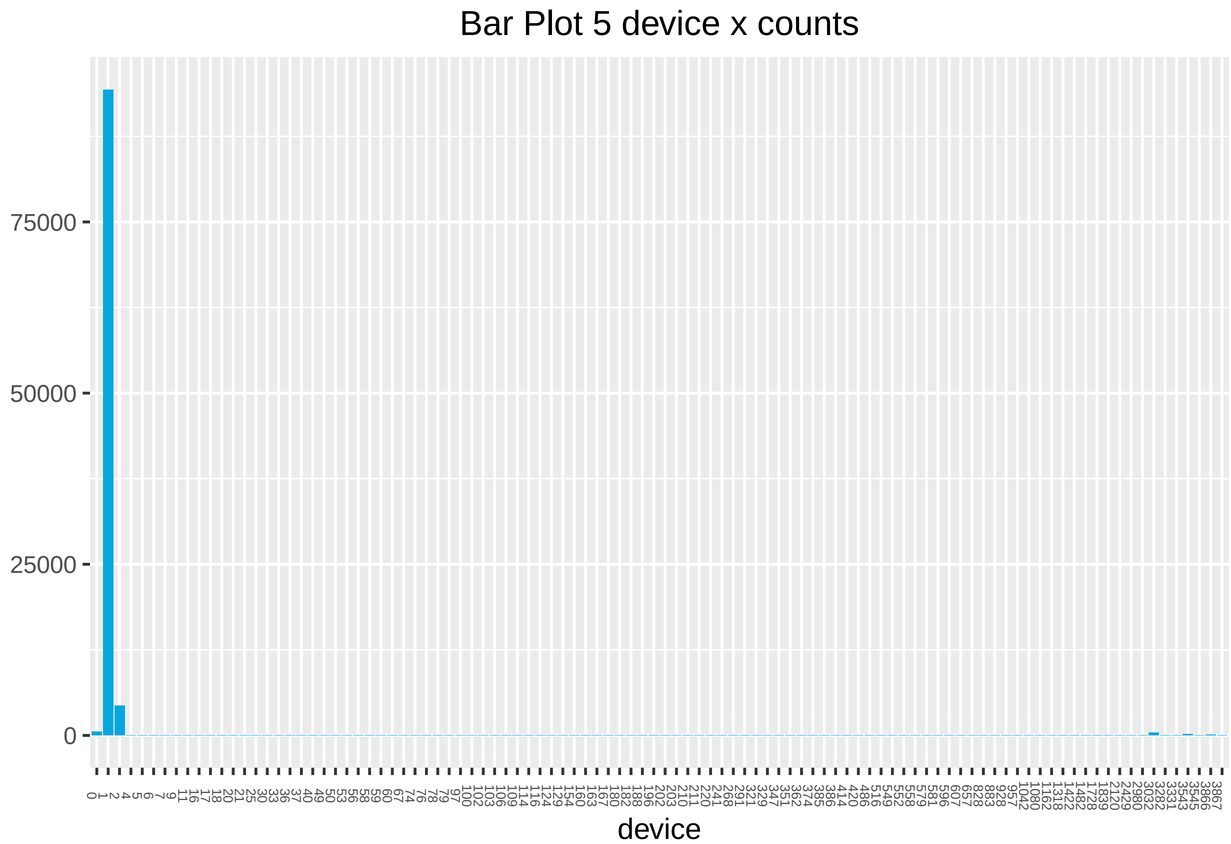
```r
# For categorical variables (or grouping variables).
# the count of categories is visualized using a bar plot, pie chart or dot charts to show the proportio

# creating a list with the same dataset rows length
df_subsets <- list(1:nrow(df))
#df_subsets[[1]][1]

# creting a aux dataset with only columns in df_subsets
df_aux <- df[, 2:8] # without click_time, which is represented by hour and weekday columns


### 2.1.1 Creating dataset for each variable counting its frequency
for(i in 1:7){
  #print(names(df_dataSets[i]) )
  df_subsets[[i]] <- df_aux %>%
    group_by_at(i) %>%
    summarise(counts = n())
}
#View(df_subsets)

### 2.1.2 Creating bar plots for each variable counting its frequency from its datasets
```

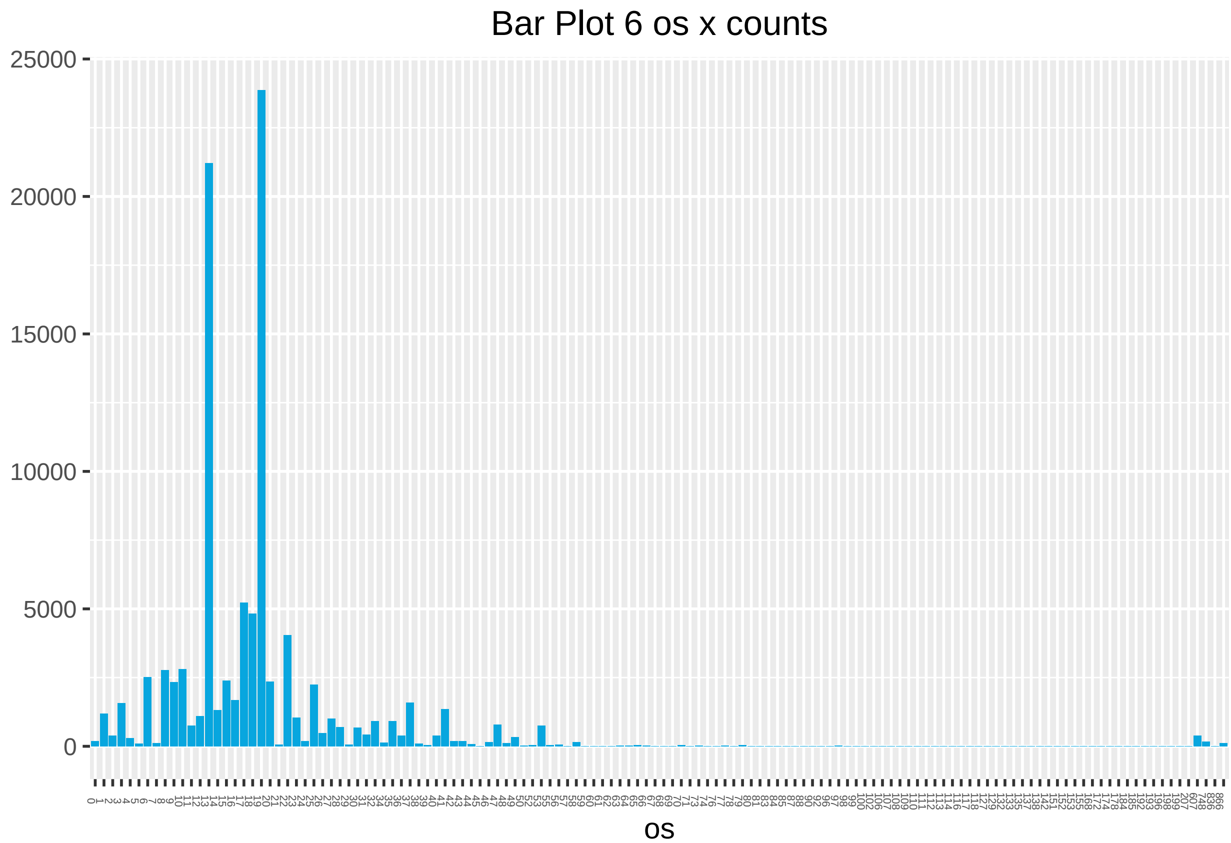
```r
# automatizing visualization for bar plots
for (i in 1:length(df_subsets)){

  if(i==3)
    next # as the IP bar plot is heavy for load, this bar is omitted

  x_column <- unlist(df_subsets[[i]][,1])
  title_x <- names(df_subsets[[i]][,1])
  title_y <- names(df_subsets[[i]][,2])


  barplot <- ggplot(df_subsets[[i]],aes(x = x_column , y = counts)) +
    geom_bar(fill = "#00A4DEF7", stat = "identity") +
    #geom_text(aes(label = counts), vjust = -0.3, size = 3) +
    ggtitle(paste("Bar Plot", i, title_x,"x", title_y)) +
    theme(plot.title = element_text(hjust = 0.5),
          #axis.title.x=element_blank(),
          axis.title.y=element_blank()) +
    xlab(title_x)

  print(barplot +
          if (i == 3)
            theme(axis.text.x = element_text(angle=-90, size=0))
          else if (i == 4)
            theme(axis.text.x = element_text(angle=-90, size=3))
          else if (i == 5)
            theme(axis.text.x = element_text(angle=-90, size=5))
          else if (i == 6)
            theme(axis.text.x = element_text(angle=-90, size=4))
          else if (i == 7)
            theme(axis.text.x = element_text(angle=-90, size=2.5)))
}
```

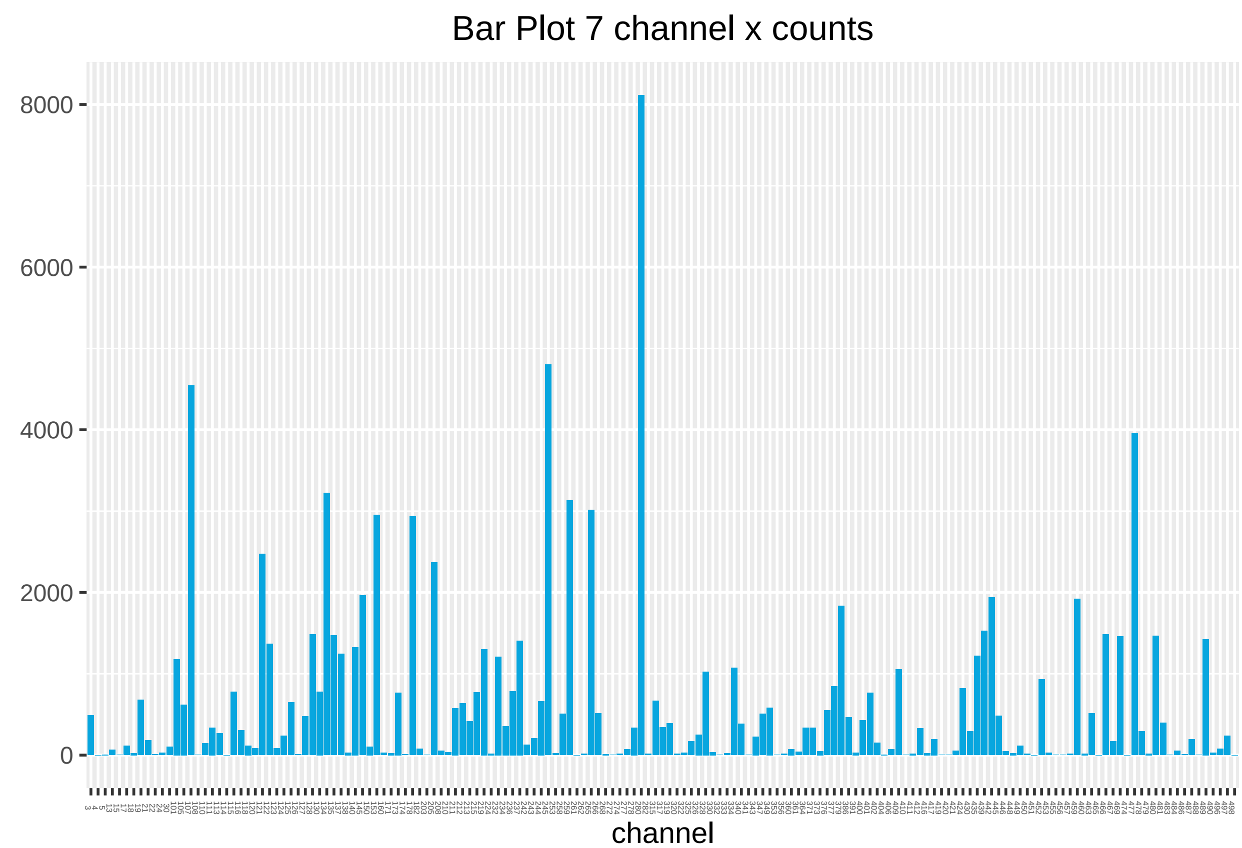

Bar Plot 1 weekday x counts
30000
20000
10000
0
Monday
Thursday
Tuesday
Wednesday
weekday


Bar Plot 2 hour x counts
6000
4000
2000
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
hour


Bar Plot 4 app x counts
15000
10000
5000
0
app


Bar Plot 5 device x counts
75000
50000
25000
0
device


Bar Plot 6 os x counts
25000
20000
15000
10000
5000
0
os



Day: wednesday > tuesday > thursday > monday (a lot less)

Hour: between 17 and 22 hour is made less download; great peak: 0h to 15h; peak 1 is 0h to 7h; peak 2 is 9h to 15h; curiously 8 has the minor value compared with its proximities; @ that's interesting to relate day and hour together;

IP: there are 5 great used IP; there are also more +-20 less used than this 5 and more than others; @ to relate IP with (all) others variables?

App: in general, the firsts are more downloaded, but there is some whose highlight a tiny and one who highlight a lot more;

Device: there is one who highlights in a huge way;

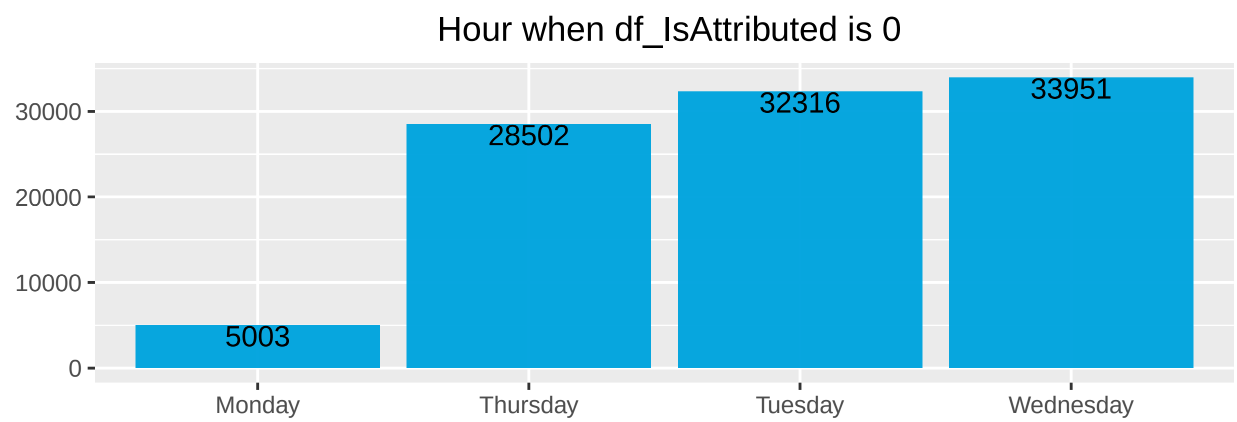
OS: there are 2 who highlight a lot;

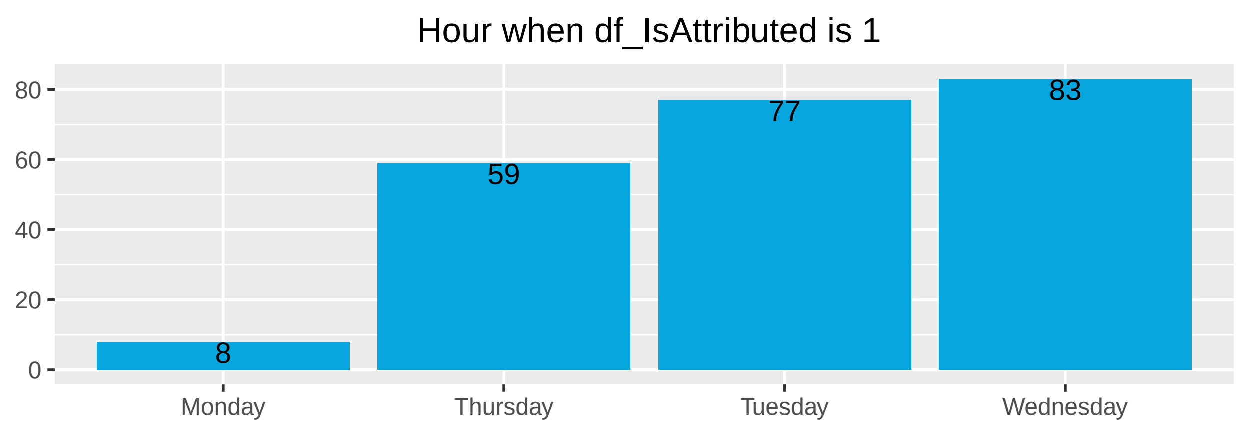
Channel: there is one that stands out a lot and others who highlights less;

2.2 Exploring hour and weekday

```r
### 2.2.1 is_attributed frequency: comparison between download and no-download
# is_attributed0 weekday
df_IsAttributed0Day <- df_IsAttributed0 %>%
  group_by(weekday) %>%
  summarise(counts = n())

p1_IsA_Day <- ggplot(df_IsAttributed0Day, aes(x = weekday, y = counts)) +
  geom_bar(fill = "#00A4DEF7", stat = "identity") +
  geom_text(aes(label = counts), vjust = 1) +
```

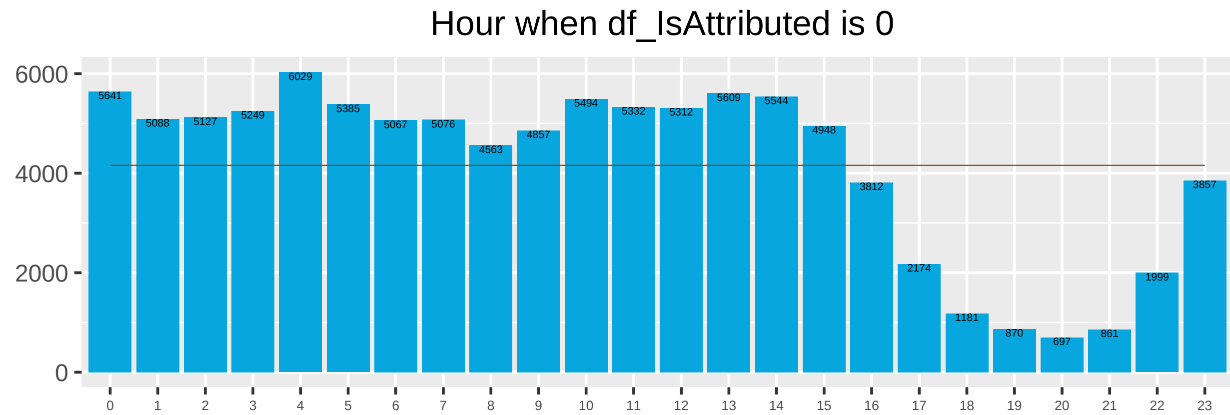
```
  ggtitle(" Hour when df_IsAttributed is 0") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())

# is_attributed1 weekday
df_IsAttributed1Day <- df_IsAttributed1 %>%
  group_by(weekday) %>%
  summarise(counts = n())

p2_IsA_Day <- ggplot(df_IsAttributed1Day, aes(x = weekday, y = counts)) +
  geom_bar(fill = "#00A4DEF7", stat = "identity") +
  geom_text(aes(label = counts), vjust = 1) +
  ggtitle(" Hour when df_IsAttributed is 1") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())

grid.arrange(p1_IsA_Day, p2_IsA_Day, ncol = 1)
```

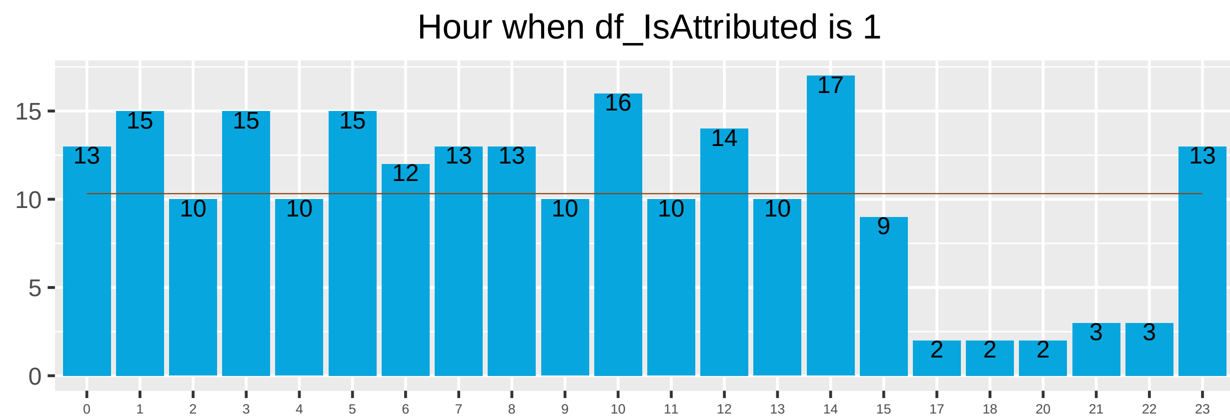




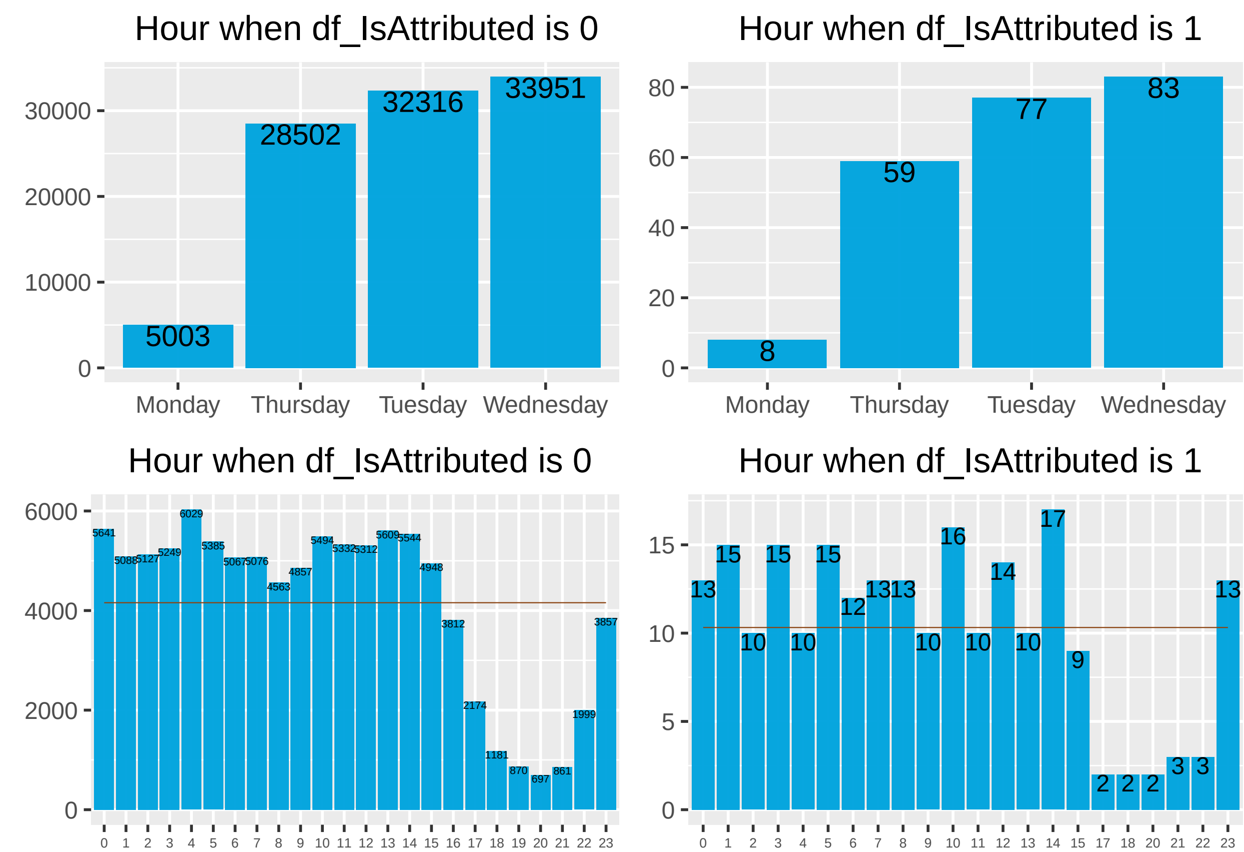

```
# Apparently there isn't no significant difference

# is_attributed0 hour
df_IsAttributed0Hour <- df_IsAttributed0 %>%
  group_by(hour) %>%
  summarise(counts = n())
```

```r
p1_IsA_Hour <- ggplot(df_IsAttributed0Hour, aes(x = hour, y = counts)) +
  geom_bar(fill = "#00A4DEF7", stat = "identity") +
  geom_text(aes(label = counts), vjust = 1, size = 1.3) +
  ggtitle(" Hour when df_IsAttributed is 0") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        axis.text.x = element_text(size=5)) +
  geom_line(aes(x = hour, y = mean(counts)), size = 0.2, color="#8B4513", group = 1)

# is_attributed1 hour
df_IsAttributed1Hour <- df_IsAttributed1 %>%
  group_by(hour) %>%
  summarise(counts = n())

p2_IsA_Hour <- ggplot(df_IsAttributed1Hour, aes(x = hour, y = counts)) +
  geom_bar(fill = "#00A4DEF7", stat = "identity") +
  geom_text(aes(label = counts), vjust = 1, size = 3) +
  ggtitle(" Hour when df_IsAttributed is 1") +
  theme(plot.title = element_text(hjust = 0.5),
        axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        axis.text.x = element_text(size=5)) +
  geom_line(aes(x = hour, y = mean(counts)), size = 0.2, color="#8B4513", group = 1)

grid.arrange(p1_IsA_Hour, p2_IsA_Hour, ncol = 1)
```






```
grid.arrange(p1_IsA_Day, p2_IsA_Day, p1_IsA_Hour, p2_IsA_Hour, ncol = 2)
```



There is a significative difference between that hour download was made.

It's interesting to use statistics to know more about this difference, but basically, the no-downloads happened more between 0 to 15h.

In this sample, there isn't download at the 16h, but there is a lot no-download.

.

.

```r
### 2.2.2 investigating more the relations between day and time variables
# is_attributed0 hour x day
df_IsAttributed0HourDay <- df_IsAttributed0 %>%
  group_by(weekday,hour) %>%
  summarise(counts = n())
```

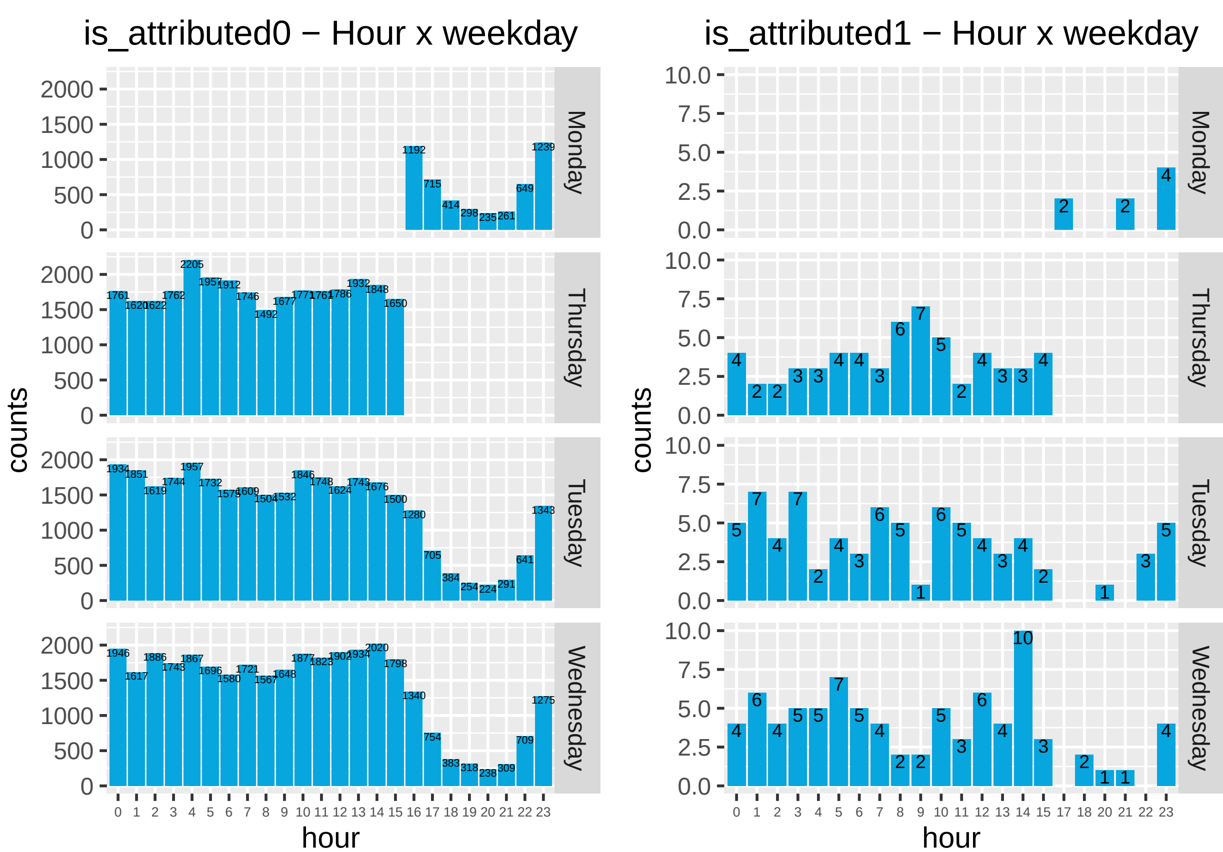
```
## `summarise()` has grouped output by 'weekday'. You can override using the `.groups` argument.
```

```r
a <- ggplot(df_IsAttributed0HourDay, aes(x = hour, y = counts)) +
  geom_bar(fill = "#00A4DEF7", stat = "identity") +
  facet_grid(weekday ~ .) +
  geom_text(aes(label = counts), vjust = 1, size = 1.5) +
  ggtitle("is_attributed0 - Hour x weekday") +
  theme(axis.text.x = element_text(size=5),
        plot.title = element_text(hjust = 0.5))

# is_attributed1 hour x day
df_IsAttributed1HourDay <- df_IsAttributed1 %>%
```

```r
  group_by(weekday,hour) %>%
  summarise(counts = n())
```

```
## `summarise()` has grouped output by 'weekday'. You can override using the `.groups` argument.
```

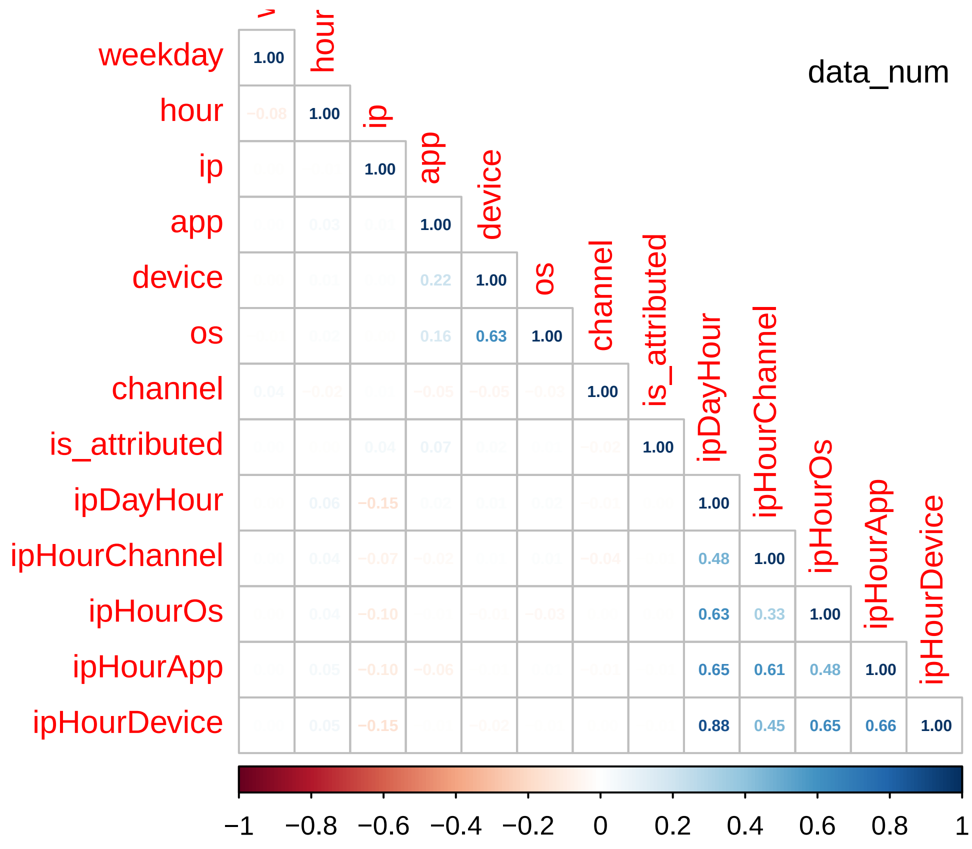
```r
b <- ggplot(df_IsAttributed1HourDay, aes(x = hour, y = counts)) +
  geom_bar(fill = "#00A4DEF7", stat = "identity") +
  facet_grid(weekday ~ .) +
  geom_text(aes(label = counts), vjust = 1, size = 2.5) +
  ggtitle("is_attributed1 - Hour x weekday") +
  theme(axis.text.x = element_text(size=5),
        plot.title = element_text(hjust = 0.5))

# there wasn't downloads at Thursday in 17 to 03h
grid.arrange(a, b, ncol = 2)
```



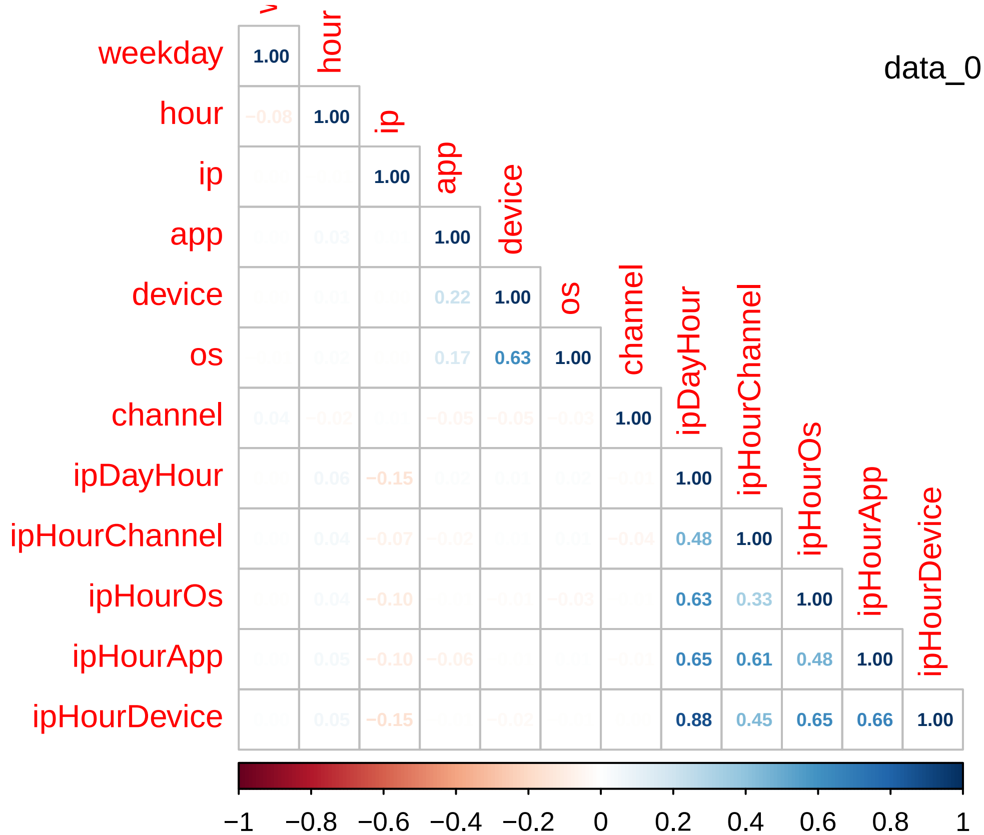
Stage 3: Correlation

```r
# Generating new columns from 'IP' relating it with another columns
data <- df %>%
  add_count(ip, weekday, hour) %>% rename("ipDayHour"     = n) %>%
  add_count(ip, hour, channel) %>% rename("ipHourChannel" = n) %>%
  add_count(ip, hour, os)      %>% rename("ipHourOs"      = n) %>%
  add_count(ip, hour, app)     %>% rename("ipHourApp"     = n) %>%
  add_count(ip, hour, device)  %>% rename("ipHourDevice"  = n)%>%
```

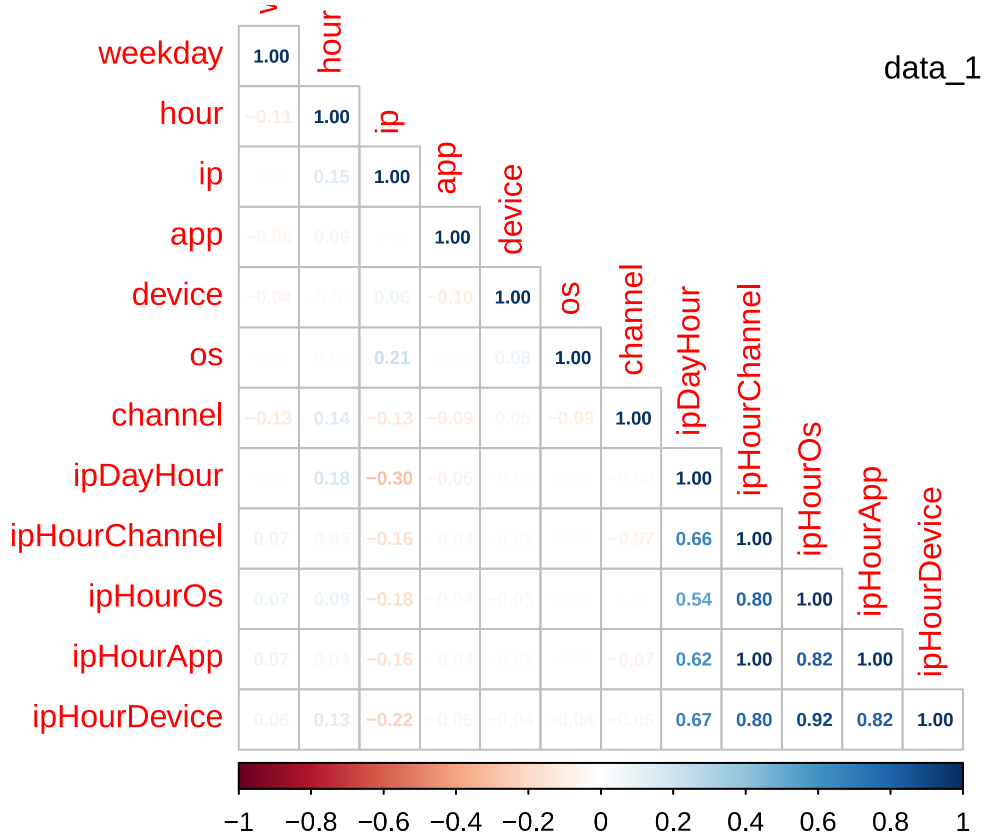
```r
  select(-click_time)
# 'click_time' isn't necessary as we already have created weekday and hour columns

## Creating a dataset for doesn't alter the original
data_num <- data
# to observe how much each variable correlates depending if it was made download or no
data_0 <- data %>%
  filter(is_attributed == '0') %>%
  select(-is_attributed)
data_1 <- data %>%
  filter(is_attributed == '1') %>%
  select(-is_attributed)

# converting to numeric to correlate variables
data_num[,1:ncol(data_num)] = lapply(data_num[,1:ncol(data_num)], as.numeric)
data_0[,1:ncol(data_0)] <- lapply(data_0[,1:ncol(data_0)], as.numeric)
data_1[,1:ncol(data_1)] <- lapply(data_1[,1:ncol(data_1)], as.numeric)

## correlations
corrplot(cor(data_num, method="pearson"), method = 'number',
         number.cex= 7/ncol(data_num), type="lower")
mtext("data_num", at=12, line=2, cex=1)
```




```r
corrplot(cor(data_0, method="pearson"), method = 'number',
         number.cex= 7/ncol(data_0), type="lower")
mtext("data_0", at=12, line=2, cex=1)
```




```
corrplot(cor(data_1, method="pearson"), method = 'number',
         number.cex= 7/ncol(data_1), type="lower")
mtext("data_1", at=12, line=2, cex=1)
```



data_num: only ‘ip’ and ‘app’ are very weakly and positively correlated with the target variable;‘device’ and ‘os’ correlate positively with ‘app’ and with each other.

data_0: particularly, for no-downloads ‘OS’ is high positive correlated with ‘device’, and weakly is ‘app’ with them.

data_1: there are some weakly and very weakly correlations; ‘ip’, ‘app’, and ‘os’ aren’t correlated as ‘data_num’ and ‘data_0’