Analysis of Facebook Digital Marketing advertsiment

Ibeh Amaka

2023-10-21

This is a dataset that was retrived from kaggel on Measuring Facebook Advertising ROI. Objectives: Company wants to maximise the amount of revenue, while minimising the amount it spends on advertising.

Firstly, we load the packages needed

library(tidyr)

## Warning: package 'tidyr' was built under R version 4.1.3

library(dplyr)

## Warning: package 'dplyr' was built under R version 4.1.3

##   
## Attaching package: 'dplyr'

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

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

library(readxl)

## Warning: package 'readxl' was built under R version 4.1.3

library(readr)

Load the dataset

data <- read\_csv("C:/Users/ibeha/OneDrive/Desktop/MY R DATASET/Facebook ad dataset/KAG\_conversion\_data.csv")

## Rows: 1143 Columns: 11  
## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## chr (2): age, gender  
## dbl (9): ad\_id, xyz\_campaign\_id, fb\_campaign\_id, interest, Impressions, Clic...  
##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

glimpse(data)

## Rows: 1,143  
## Columns: 11  
## $ ad\_id <dbl> 708746, 708749, 708771, 708815, 708818, 708820, 70~  
## $ xyz\_campaign\_id <dbl> 916, 916, 916, 916, 916, 916, 916, 916, 916, 916, ~  
## $ fb\_campaign\_id <dbl> 103916, 103917, 103920, 103928, 103928, 103929, 10~  
## $ age <chr> "30-34", "30-34", "30-34", "30-34", "30-34", "30-3~  
## $ gender <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", ~  
## $ interest <dbl> 15, 16, 20, 28, 28, 29, 15, 16, 27, 28, 31, 7, 16,~  
## $ Impressions <dbl> 7350, 17861, 693, 4259, 4133, 1915, 15615, 10951, ~  
## $ Clicks <dbl> 1, 2, 0, 1, 1, 0, 3, 1, 1, 3, 0, 0, 0, 0, 7, 0, 1,~  
## $ Spent <dbl> 1.43, 1.82, 0.00, 1.25, 1.29, 0.00, 4.77, 1.27, 1.~  
## $ Total\_Conversion <dbl> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~  
## $ Approved\_Conversion <dbl> 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,~

The documenation describes the columns in the data as follows:

1.) ad\_id: unique ID for each ad.

2.) xyz\_campaign\_id: an ID associated with each ad campaign of XYZ company.

3.) fb\_campaign\_id: an ID associated with how Facebook tracks each campaign.

4.) age: age of the person to whom the ad is shown.

5.) gender: gender of the person to whom the add is shown

6.) interest: a code specifying the category to which the person’s interest belongs (interests are as mentioned in the person’s Facebook public profile).

7.) Impressions: the number of times the ad was shown.

8.) Clicks: number of clicks on for that ad.

9.) Spent: Amount paid by company xyz to Facebook, to show that ad.

10.) Total conversion: Total number of people who enquired about the product after seeing the ad.

11.) Approved conversion: Total number of people who bought the product after seeing the ad.

We see age is in character data type, and not having a specific numbers for age, so I will take the mean of age. in addition, I will create a copy of the dataset set.

unique(data$age)

## [1] "30-34" "35-39" "40-44" "45-49"

new\_data <- data  
new\_data$age[new\_data$age == "30-34"] <- 32  
new\_data$age[new\_data$age == "35-39"] <- 37  
new\_data$age[new\_data$age == "40-44"] <- 42  
new\_data$age[new\_data$age == "45-49"] <- 47

Convert the variables to integers

new\_data$age <- as.integer(new\_data$age)  
str(new\_data$age)

## int [1:1143] 32 32 32 32 32 32 32 32 32 32 ...

change gender to 1 and 0

new\_data$gender[new\_data$gender == "M"] <- 0  
new\_data$gender[new\_data$gender == "F"] <- 1  
new\_data$gender <- as.integer(new\_data$gender)  
glimpse(new\_data)

## Rows: 1,143  
## Columns: 11  
## $ ad\_id <dbl> 708746, 708749, 708771, 708815, 708818, 708820, 70~  
## $ xyz\_campaign\_id <dbl> 916, 916, 916, 916, 916, 916, 916, 916, 916, 916, ~  
## $ fb\_campaign\_id <dbl> 103916, 103917, 103920, 103928, 103928, 103929, 10~  
## $ age <int> 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32~  
## $ gender <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~  
## $ interest <dbl> 15, 16, 20, 28, 28, 29, 15, 16, 27, 28, 31, 7, 16,~  
## $ Impressions <dbl> 7350, 17861, 693, 4259, 4133, 1915, 15615, 10951, ~  
## $ Clicks <dbl> 1, 2, 0, 1, 1, 0, 3, 1, 1, 3, 0, 0, 0, 0, 7, 0, 1,~  
## $ Spent <dbl> 1.43, 1.82, 0.00, 1.25, 1.29, 0.00, 4.77, 1.27, 1.~  
## $ Total\_Conversion <dbl> 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~  
## $ Approved\_Conversion <dbl> 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,~

abbreviate some variable name

library(heatmaply)

## Loading required package: plotly

## Warning: package 'plotly' was built under R version 4.1.3

## Loading required package: ggplot2

##   
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':  
##   
## last\_plot

## The following object is masked from 'package:stats':  
##   
## filter

## The following object is masked from 'package:graphics':  
##   
## layout

## Loading required package: viridis

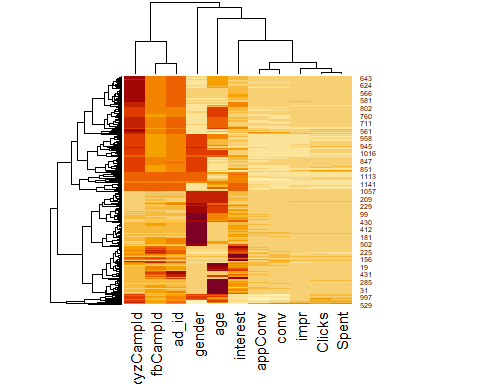
## Loading required package: viridisLite

##   
## ======================  
## Welcome to heatmaply version 1.5.0  
##   
## Type citation('heatmaply') for how to cite the package.  
## Type ?heatmaply for the main documentation.  
##   
## The github page is: https://github.com/talgalili/heatmaply/  
## Please submit your suggestions and bug-reports at: https://github.com/talgalili/heatmaply/issues  
## You may ask questions at stackoverflow, use the r and heatmaply tags:   
## https://stackoverflow.com/questions/tagged/heatmaply  
## ======================

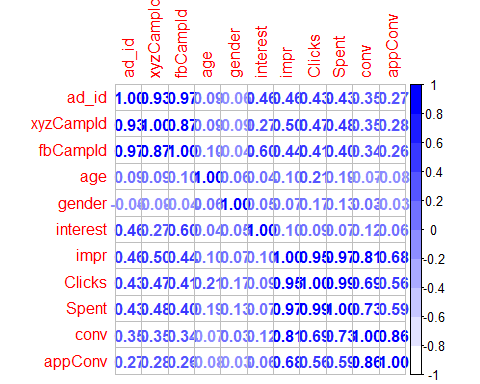
library(corrplot)

## corrplot 0.92 loaded

dataMatNorm <- as.matrix(normalize(new\_data, method = "standardize"))  
correlation\_matrix <- cor(dataMatNorm)  
heatmap(dataMatNorm)



corrplot(correlation\_matrix, method = "number", col = colorRampPalette(c("white", "blue"))(10))

 The results shows that there is a strong corrolation between impression and clicks, spents, conv and appconv.

Lets talk about the terms use in this analysis for better understanding Click-through rate (CTR) is a metric used in online advertising and digital marketing to measure the effectiveness of an online advertising campaign or a specific advertisement. CTR is expressed as a percentage and represents the proportion of people who click on an ad or a specific link within a webpage, email, or other digital content, relative to the total number of people who were exposed to the ad (impressions) or had the opportunity to click (clicks).The formula to calculate CTR is:CTR=CTR=( Number of Clicks/ Number of Impressions or Opportunities)×100

Cost Per Click (CPC) is a common digital advertising metric used by advertisers and marketers to measure the cost-effectiveness of online advertising campaigns, particularly in pay-per-click (PPC) advertising models. CPC represents the average cost that an advertiser pays each time a user clicks on their ad. It is a crucial metric for assessing the efficiency and return on investment (ROI) of online advertising efforts. CPC= Total Cost of Campaign/ Total Number of Click

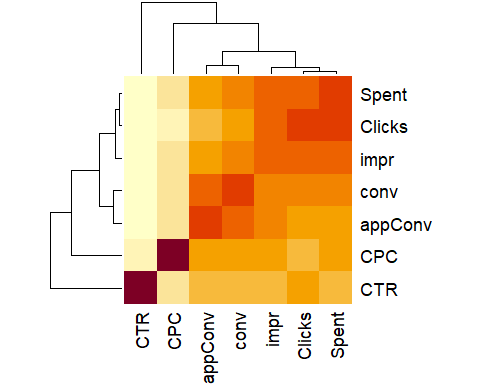
Conversion Rate (CR) is a key performance indicator (KPI) used in marketing and e-commerce to measure the effectiveness of a marketing campaign, website, or online sales funnel. It represents the percentage of visitors or users who take a desired action, referred to as a “conversion,” out of the total number of visitors or users who had the opportunity to take that action. Conversion Rate is expressed as a percentage.

The formula to calculate Conversion Rate is: CR=( Number of Conversions//Total Number of Visitors or Users)×100

Create the additional KPIs with dplyr create the CTR and CPC figures

new\_data <- new\_data %>%  
 mutate(CTR = ((Clicks/impr)\* 100), CPC = Spent/Clicks)  
#round to 4 decemail place  
new\_data$CTR <- round(new\_data$CTR, 4)  
new\_data$CPC <- round(new\_data$CPC, 2)

# create trimmed dataset  
dataTfTrim <- new\_data %>%  
 select(CTR, CPC, appConv, conv, impr, Spent, Clicks)  
  
# omit missing values, normalise data, calculate correlations and plot heatmap  
heatmap(cor(normalize(na.omit(dataTfTrim))))



# Install and load reshape2 package  
library(reshape2)

## Warning: package 'reshape2' was built under R version 4.1.3

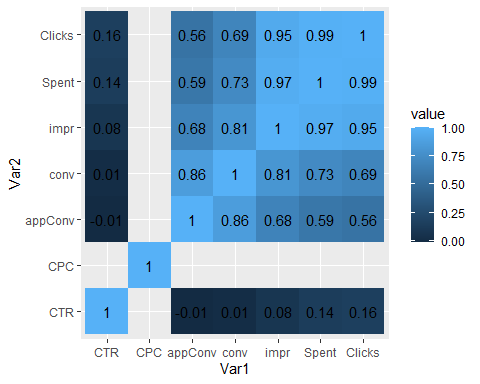
##   
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':  
##   
## smiths

# creating correlation matrix  
corr\_mat <- round(cor(dataTfTrim),2)  
# reduce the size of correlation matrix  
melted\_corr\_mat <- melt(corr\_mat)  
#remove NA  
melted\_corr\_mat <- na.omit(melted\_corr\_mat)  
head(melted\_corr\_mat)

## Var1 Var2 value  
## 1 CTR CTR 1.00  
## 3 appConv CTR -0.01  
## 4 conv CTR 0.01  
## 5 impr CTR 0.08  
## 6 Spent CTR 0.14  
## 7 Clicks CTR 0.16

library(ggplot2)  
ggplot(data = melted\_corr\_mat, aes(x=Var1, y=Var2,  
 fill=value)) +   
 geom\_tile() +  
 geom\_text(aes(Var2, Var1, label = value),   
 color = "black", size = 4)



From our broad overview of the data, we can see that the more we spend, the more clicks and conversions we seem to get. That’s quite reassuring to know, but doesn’t really give us the ‘actionable insight’ we were looking for.

For our next stage in the analysis, we’ll look at a specific campaign in more detail and see what we can pull out of it. First of all, let’s choose a campaign, the one on which we regularly spend the most money and regularly get the most conversions Group by Campaign Name and calculate total spending and conversions

campaign\_summary <- new\_data %>%  
 group\_by(xyzCampId) %>%  
 summarise(TotalSpending = sum(Spent, na.rm = TRUE),  
 TotalConversions = sum(appConv, na.rm = TRUE))

Find the campaign with the highest spending and the most conversions

top\_campaign <- campaign\_summary %>%  
 filter(TotalSpending == max(TotalSpending) & TotalConversions == max(TotalConversions))  
print(top\_campaign)

## # A tibble: 1 x 3  
## xyzCampId TotalSpending TotalConversions  
## <dbl> <dbl> <dbl>  
## 1 1178 55662. 872

we can see that the 1178 has the most spending and impression

we use our prevous names

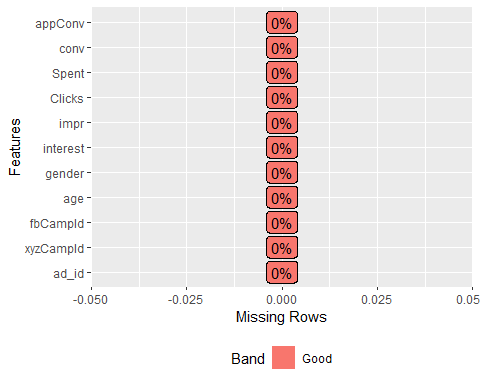
data1178 <- data %>%  
 rename(xyzCampId = xyz\_campaign\_id, fbCampId = fb\_campaign\_id, impr = Impressions,  
 conv = Total\_Conversion, appConv = Approved\_Conversion) %>%  
 filter(xyzCampId == 1178)

check for missing values

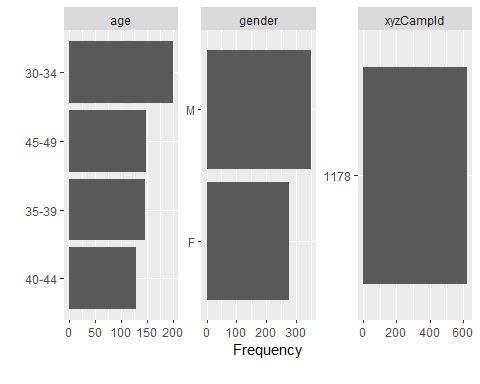
library(DataExplorer)

## Warning: package 'DataExplorer' was built under R version 4.1.3

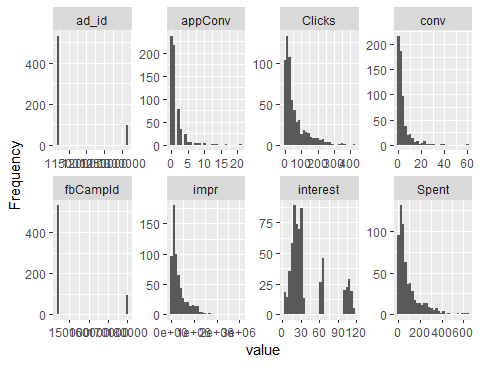
plot\_missing(data1178)

 Non is missing, Let’s look at the distributions of our data, variable by variable:

options(repr.plot.width=4, repr.plot.height=4)  
plot\_bar(data1178)



options(repr.plot.width=8, repr.plot.height=4)  
plot\_histogram(data1178)

 Lets see if our Return on Advertising Spend (ROAS) or the money spent is generating a revenue Return on Advertising Spend (ROAS) is a metric used in digital marketing and advertising to measure the effectiveness and efficiency of advertising campaigns. It quantifies the revenue generated for each dollar spent on advertising. ROAS is typically expressed as a ratio or percentage and is calculated using the following formula:

ROAS = (Revenue Generated from Advertising) / (Cost of Advertising)

An enquiry (Total conversion, conv) is worth £5, and a sale (Approved conversion, appConv),is worth £100. We can now create our conversion value-based variables.

data1178 <- data1178 %>%  
 mutate(totConv = conv + appConv,  
 conVal = conv \* 5,  
 appConVal = appConv \* 100) %>%  
 mutate(totConVal = conVal + appConVal) %>%  
 mutate(costPerCon = round(Spent / totConv, 2),  
 ROAS = ifelse(Spent == 0, 0, round(totConVal / Spent, 2)))

summary(data1178$ROAS)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 0.330 1.470 3.685 3.330 70.470

ROAS Greater than 1: If the ROAS is greater than 1, it indicates that for every dollar spent on advertising, the campaign generated more than one dollar in revenue. This is generally considered a positive outcome, indicating a profitable campaign.

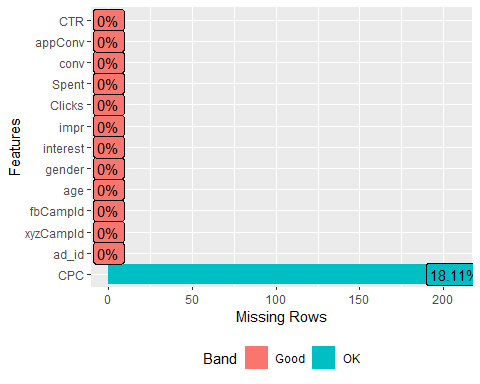
The mean of ROAS is 3.685 this means every dollar spent on advertisement, the campeign generated 3.685 revenue

high\_ROAS\_campaigns <- data1178 %>%  
 filter(ROAS > 2)  
high\_ROAS\_campaigns

## # A tibble: 258 x 17  
## ad\_id xyzCampId fbCampId age gender interest impr Clicks Spent conv  
## <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1121091 1178 144531 30-34 M 10 1194718 141 254. 28  
## 2 1121092 1178 144531 30-34 M 10 637648 67 122. 13  
## 3 1121095 1178 144531 30-34 M 10 459690 50 86.3 5  
## 4 1121097 1178 144532 30-34 M 15 30068 1 1.82 1  
## 5 1121098 1178 144532 30-34 M 15 1267550 123 237. 24  
## 6 1121100 1178 144532 30-34 M 15 3052003 340 640. 60  
## 7 1121101 1178 144532 30-34 M 15 29945 1 1.59 2  
## 8 1121102 1178 144532 30-34 M 15 357856 30 53.0 7  
## 9 1121104 1178 144533 30-34 M 16 2080666 202 360. 40  
## 10 1121105 1178 144533 30-34 M 16 145999 9 16.5 5  
## # i 248 more rows  
## # i 7 more variables: appConv <dbl>, totConv <dbl>, conVal <dbl>,  
## # appConVal <dbl>, totConVal <dbl>, costPerCon <dbl>, ROAS <dbl>

We have been working on the 1178 has the most spending and impression, we can also see if others too has the high ROAS or less so we can decide to stop that perticular add

library(DataExplorer)  
plot\_missing(new\_data)



unique(data$xyz\_campaign\_id)

## [1] 916 936 1178

all\_data <- new\_data %>%   
 mutate(totConv = conv + appConv,  
 conVal = conv \* 5,  
 appConVal = appConv \* 100) %>%  
 mutate(totConVal = conVal + appConVal) %>%  
 mutate(costPerCon = round(Spent / totConv, 2),  
 ROAS = ifelse(Spent == 0, 0, round(totConVal / Spent, 2)))

kk = all\_data %>%   
 filter(xyzCampId == 916) %>%   
 summarise(mean\_ros = mean(ROAS))  
kk

## # A tibble: 1 x 1  
## mean\_ros  
## <dbl>  
## 1 14.3

kk2 = all\_data %>%   
 filter(xyzCampId == 936) %>%   
 summarise(mean\_ros = mean(ROAS))  
kk2

## # A tibble: 1 x 1  
## mean\_ros  
## <dbl>  
## 1 12.9

we see that though we spend less and have low impressions on the add 916 and 936 this shows that mean of add 916 is 14.3 while that of 936 is 12.9. This means that it has larger returns than add 1178 even if it has the highest impressions and spenings

Our objective here is to find the cost of impressions in other to increase brand awareness, now lets work with our data1178

Cost Per Mille (CPM). This number is the cost of one thousand impressions.

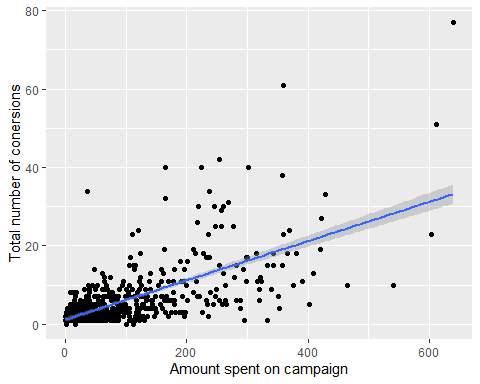
data1178 <- data1178 %>%  
 mutate(CPM = round((Spent / impr) \* 1000, 2))  
sum(data1178$CPM)

## [1] 156.46

The result, 156.46, represents the total CPM for all the data in your dataset. This means that, on average, you are spending $156.46 to reach one thousand impressions across all the campaigns or data points in your dataset. This metric is useful for understanding the efficiency of your advertising spend in terms of impressions generated for a given cost. Lower CPM values typically indicate more efficient advertising spending.

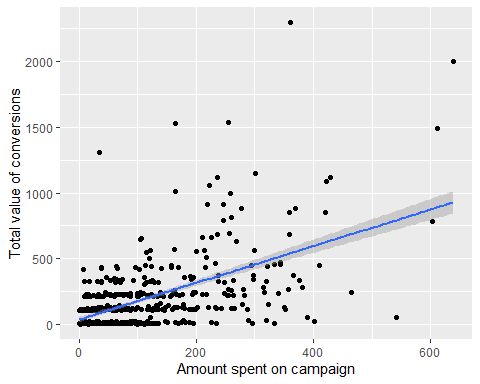
options(repr.plot.width=6, repr.plot.height=3)  
ggplot(data1178, aes(Spent, totConv)) + geom\_point() + geom\_smooth(method = "lm") +  
 labs(x = "Amount spent on campaign", y = "Total number of conersions")

## `geom\_smooth()` using formula = 'y ~ x'



ggplot(data1178, aes(Spent, totConVal)) + geom\_point() + geom\_smooth(method = "lm") +  
 labs(x = "Amount spent on campaign", y = "Total value of conversions")

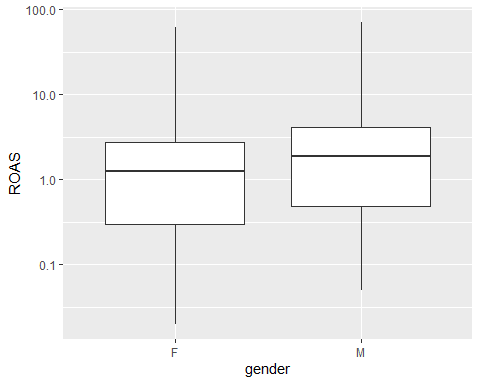
## `geom\_smooth()` using formula = 'y ~ x'



options(repr.plot.width=4, repr.plot.height=3)  
ggplot(data1178, aes(gender, ROAS)) + geom\_boxplot() + scale\_y\_log10()

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 16 rows containing non-finite values (`stat\_boxplot()`).

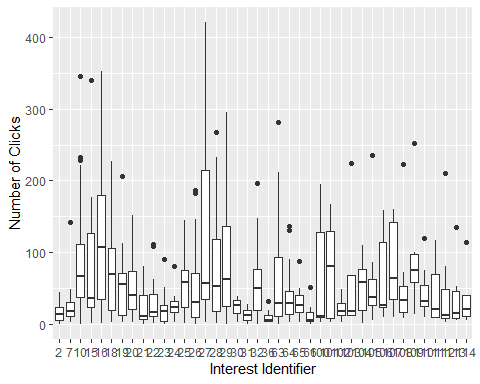
 The data look quite symmetrical with a log-transformed axis, but without the log-transformation, it doesn’t fit the normal distribution, so we’ll look to see if this difference is significant using a non-parametric test:

wilcox.test(ROAS ~ gender, data=data1178)

##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: ROAS by gender  
## W = 39672, p-value = 0.0001523  
## alternative hypothesis: true location shift is not equal to 0

The small p-value (0.0001523) indicates that there is strong evidence to reject the null hypothesis. Therefore, you can conclude that there is a statistically significant difference in ROAS between the two gender groups.

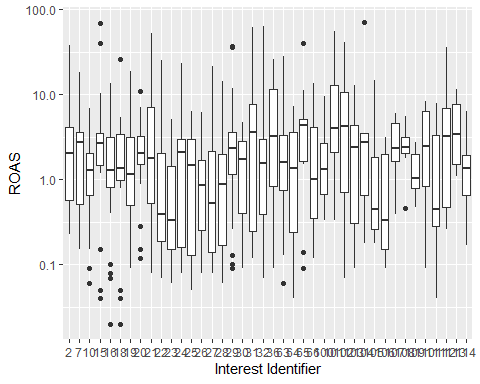
options(repr.plot.width=8, repr.plot.height=3)  
ggplot(data1178, aes(as.factor(interest), Clicks)) + geom\_boxplot() +  
 labs(x = "Interest Identifier", y = "Number of Clicks")



options(repr.plot.width=8, repr.plot.height=3)  
data1178 %>%  
 ggplot(aes(as.factor(interest), ROAS)) + geom\_boxplot() + scale\_y\_log10() +  
 labs(x = "Interest Identifier", y = "ROAS")

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 16 rows containing non-finite values (`stat\_boxplot()`).



data1178 %>%  
 select(interest, ROAS, Clicks) %>%  
 group\_by(interest) %>%  
 filter(ROAS != 'Inf') %>%  
 summarise(medianROAS = round(median(ROAS) ,2),   
 meanROAS = round(mean(ROAS), 2), clicks = sum(Clicks)) %>%  
 arrange(desc(meanROAS)) %>%  
 head(n = 10)

## # A tibble: 10 x 4  
## interest medianROAS meanROAS clicks  
## <dbl> <dbl> <dbl> <dbl>  
## 1 104 2.75 15.5 265  
## 2 101 3.96 15.0 524  
## 3 102 4.27 10.4 150  
## 4 112 3.21 8.06 339  
## 5 15 2.68 7.89 1554  
## 6 31 3.4 7.71 189  
## 7 36 3.21 7.38 126  
## 8 65 4.38 7 343  
## 9 21 1.63 6.06 493  
## 10 113 3.43 5.01 233

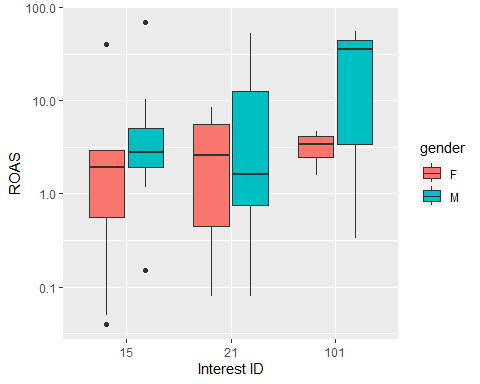
From the result above, we can see that people that fall under interest 104, 101, and 102, has the highest ROAS. if we can find a way to target those group of people it will be beneficial to us.

Lets filter the intrest based on the number of clicks by gender

options(repr.plot.width=8, repr.plot.height=3)  
data1178 %>%  
 filter(interest == 101 | interest == 15 | interest == 21) %>%  
 ggplot(aes(x = as.factor(interest), y = ROAS, fill = gender)) + geom\_boxplot() + scale\_y\_log10() +  
 labs(x = 'Interest ID', y = 'ROAS')

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 1 rows containing non-finite values (`stat\_boxplot()`).



data1178 %>%  
 select(interest, gender, ROAS, Clicks) %>%  
 group\_by(interest, gender) %>%  
 filter(ROAS != 'Inf', interest == 101 | interest == 15 | interest == 21) %>%  
 summarise(medianROAS = round(median(ROAS), 2),  
 meanROAS = round(mean(ROAS) ,2), clicks = sum(Clicks)) %>%  
 arrange(desc(meanROAS))

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

## # A tibble: 6 x 5  
## # Groups: interest [3]  
## interest gender medianROAS meanROAS clicks  
## <dbl> <chr> <dbl> <dbl> <dbl>  
## 1 101 M 35.5 30.5 17  
## 2 15 M 2.79 8.89 827  
## 3 21 M 1.47 8.75 200  
## 4 15 F 1.98 6.38 727  
## 5 21 F 2.6 3.36 293  
## 6 101 F 3.41 3.28 507

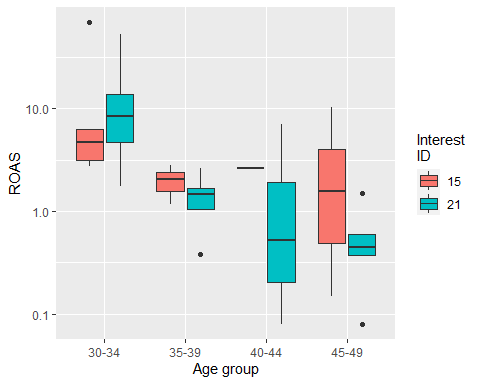
from the result above we can see that males has more ROAS returns than females

Analysis by age We’ve been able to break apart this campaign by interest and gender, with each split allowing us to pull out groups with the highest ROAS. What about the age variable though, we haven’t used that yet, so let’s see if we can be even more targeted:

options(repr.plot.width=8, repr.plot.height=4)  
data1178 %>%  
 filter(interest == 21 | interest == 15 & gender == 'M') %>%  
 group\_by(age, interest) %>%   
 ggplot(aes(x = as.factor(age), y = ROAS, fill = as.factor(interest))) + geom\_boxplot() + scale\_y\_log10() +  
 labs(x = 'Age group', y = 'ROAS') + scale\_fill\_discrete(name="Interest\nID")

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 1 rows containing non-finite values (`stat\_boxplot()`).



data1178 %>%  
 select(age, interest, gender, ROAS, Clicks) %>%  
 group\_by(age, interest) %>%  
 filter(ROAS != 'Inf', interest == 21 | interest == 15, gender == 'M') %>%  
 summarise(medianROAS = round(median(ROAS), 2),  
 meanROAS = round(mean(ROAS) ,2), clicks = sum(Clicks)) %>%  
 arrange(desc(meanROAS))

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

## # A tibble: 8 x 5  
## # Groups: age [4]  
## age interest medianROAS meanROAS clicks  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 30-34 21 13.9 18.4 58  
## 2 30-34 15 4.73 17.2 495  
## 3 45-49 15 1.57 3.97 138  
## 4 40-44 15 2.62 2.62 26  
## 5 35-39 15 2.07 2.02 168  
## 6 35-39 21 1.47 1.47 44  
## 7 45-49 21 0.45 0.38 98  
## 8 40-44 21 0 0 0

Age 30-32 has the highest mean returns of ROAS