Code ▾

Unsupervised learning for Consumer behaviour and marketing analysis

Stephanie Omwanda

Introduction

> Kira Plastinina is a Russian brand that is sold through a defunct chain of retail stores in Russia, Ukraine, Kazakhstan, Belarus, China, Philippines, and Armenia. The brand's Sales and Marketing team would like to understand their customer's behavior from data that they have collected over the past year. More specifically, they would like to learn the characteristics of customer groups.

Research Question

> Perform clustering and dimensionality reduction stating insights drawn from your analysis and visualizations. Upon implementation, provide comparisons between K-Means clustering vs Hierarchical clustering, highlighting the strengths and limitations of each approach in the context of your analysis. Your findings should help inform the team in formulating the marketing and sales strategies of the brand.

Understanding the context

> The dataset can be found here [http://bit.ly/EcommerceCustomersDataset (http://bit.ly/EcommerceCustomersDataset)]. The dataset consists of 10 numerical and 8 categorical attributes. The ‘Revenue’ attribute can be used as the class label. “Administrative”, “Administrative Duration”, “Informational”, “Informational Duration”, “Product Related” and “Product Related Duration” represents the number of different types of pages visited by the visitor in that session and total time spent in each of these page categories. The values of these features are derived from the URL information of the pages visited by the user and updated in real-time when a user takes an action, e.g. moving from one page to another. The “Bounce Rate”, “Exit Rate” and “Page Value” features represent the metrics measured by “Google Analytics” for each page in the e-commerce site. The value of the “Bounce Rate” feature for a web page refers to the percentage of visitors who enter the site from that page and then leave (“bounce”) without triggering any other requests to the analytics server during that session The value of the “Exit Rate” feature for a specific web page is calculated as for all pageviews to the page, the percentage that was the last in the session. The “Page Value” feature represents the average value for a web page that a user visited before completing an e-commerce transaction. The “Special Day” feature indicates the closeness of the site visiting time to a specific special day (e.g. Mother’s Day, Valentine’s Day) in which the sessions are more likely to be finalized with the transaction. The value of this attribute is determined by considering the dynamics of e-commerce such as the duration between the order date and delivery date. For example, for Valentina’s day, this value takes a nonzero value between February 2 and February 12, zero before and after this date unless it is close to another special day, and its maximum value of 1 on February 8. The dataset also includes the operating system, browser, region, traffic type, visitor type as returning or new visitor, a Boolean value indicating whether the date of the visit is weekend, and month of the year

Metric of Success

> The model’s accuracy score is what will be used to measure the model’s predictive power.

Loading and Cleaning the dataset

Hide

```
 # load libraries
library(readr) # provides a faster and friendly way to read rectangular data like
csvs
library(dplyr) # provides a flexible grammar of data manipulation library(tinytex)
library(knitr) # for dynamic report generation
options(warn = -1)
```

Hide

```
# Loading the csv file
df = read_csv('online_shoppers_intention.csv')
```

```
Parsed with column specification:
cols(
  Administrative = [32mcol_double()[39m,
  Administrative_Duration = [32mcol_double()[39m,
  Informational = [32mcol_double()[39m,
  Informational_Duration = [32mcol_double()[39m,
  ProductRelated = [32mcol_double()[39m,
  ProductRelated_Duration = [32mcol_double()[39m,
  BounceRates = [32mcol_double()[39m,
  ExitRates = [32mcol_double()[39m,
  PageValues = [32mcol_double()[39m,
  SpecialDay = [32mcol_double()[39m,
  Month = [31mcol_character()[39m,
  OperatingSystems = [32mcol_double()[39m,
  Browser = [32mcol_double()[39m,
  Region = [32mcol_double()[39m,
  TrafficType = [32mcol_double()[39m,
  VisitorType = [31mcol_character()[39m,
  Weekend = [33mcol_logical()[39m,
  Revenue = [33mcol_logical()[39m
)
```

Hide

```
# Previewing the first five rows of the dataframe
head(df)
```

Administrative <dbl>	Administrative_Duration <dbl>	Informational <dbl>	Informational_Duration <dbl>
0	0	0	0

0	0	0	0
0	-1	0	-1
0	0	0	0
0	0	0	0
0	0	0	0

6 rows | 1-5 of 18 columns

Hide

```
# show structure of  dataset
str(df)
```

```
Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame':    12330 obs. of  18 vari
ables:
 $ Administrative         : num  0 0 0 0 0 0 0 1 0 0 ...
 $ Administrative_Duration: num  0 0 -1 0 0 0 -1 -1 0 0 ...
 $ Informational          : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Informational_Duration : num  0 0 -1 0 0 0 -1 -1 0 0 ...
 $ ProductRelated         : num  1 2 1 2 10 19 1 1 2 3 ...
 $ ProductRelated_Duration: num  0 64 -1 2.67 627.5 ...
 $ BounceRates            : num  0.2 0 0.2 0.05 0.02 ...
 $ ExitRates              : num  0.2 0.1 0.2 0.14 0.05 ...
 $ PageValues             : num  0 0 0 0 0 0 0 0 0 0 ...
 $ SpecialDay             : num  0 0 0 0 0 0 0.4 0 0.8 0.4 ...
 $ Month                  : chr  "Feb" "Feb" "Feb" "Feb" ...
 $ OperatingSystems       : num  1 2 4 3 3 2 2 1 2 2 ...
 $ Browser                : num  1 2 1 2 3 2 4 2 2 4 ...
 $ Region                 : num  1 1 9 2 1 1 3 1 2 1 ...
 $ TrafficType            : num  1 2 3 4 4 3 3 5 3 2 ...
 $ VisitorType            : chr  "Returning_Visitor" "Returning_Visitor" "Returnin
g_Visitor" "Returning_Visitor" ...
 $ Weekend                : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
 $ Revenue                : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 - attr(*, "spec")=
  .. cols(
  ..   Administrative = [32mcol_double()[39m,
  ..   Administrative_Duration = [32mcol_double()[39m,
  ..   Informational = [32mcol_double()[39m,
  ..   Informational_Duration = [32mcol_double()[39m,
  ..   ProductRelated = [32mcol_double()[39m,
  ..   ProductRelated_Duration = [32mcol_double()[39m,
  ..   BounceRates = [32mcol_double()[39m,
  ..   ExitRates = [32mcol_double()[39m,
  ..   PageValues = [32mcol_double()[39m,
  ..   SpecialDay = [32mcol_double()[39m,
  ..   Month = [31mcol_character()[39m,
  ..   OperatingSystems = [32mcol_double()[39m,
  ..   Browser = [32mcol_double()[39m,
  ..   Region = [32mcol_double()[39m,
  ..   TrafficType = [32mcol_double()[39m,
  ..   VisitorType = [31mcol_character()[39m,
  ..   Weekend = [33mcol_logical()[39m,
  ..   Revenue = [33mcol_logical()[39m
  .. )
```

Hide

Hide

```
# catch a glimpse of the dataset
glimpse(df)
```

```
Observations: 12,330
Variables: 18
$ Administrative          [3m[38;5;246m<dbl>[39m[23m 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ Administrative_Duration [3m[38;5;246m<dbl>[39m[23m 0.0, 0.0, -1.0, 0.0, 0.0, 0.0
, -1.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.…
$ Informational           [3m[38;5;246m<dbl>[39m[23m 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ Informational_Duration  [3m[38;5;246m<dbl>[39m[23m 0, 0, -1, 0, 0, 0, -1, -1, 0,
0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, -1, 0, 0…
$ ProductRelated          [3m[38;5;246m<dbl>[39m[23m 1, 2, 1, 2, 10, 19, 1, 1, 2,
3, 3, 16, 7, 6, 2, 23, 1, 13, 2, 20, 8, 1, 3, …
$ ProductRelated_Duration [3m[38;5;246m<dbl>[39m[23m 0.000000, 64.000000, -1.00000
0, 2.666667, 627.500000, 154.216667, -1.000000…
$ BounceRates             [3m[38;5;246m<dbl>[39m[23m 0.200000000, 0.000000000, 0.2
00000000, 0.050000000, 0.020000000, 0.01578947…
$ ExitRates               [3m[38;5;246m<dbl>[39m[23m 0.200000000, 0.100000000, 0.2
00000000, 0.140000000, 0.050000000, 0.02456140…
$ PageValues              [3m[38;5;246m<dbl>[39m[23m 0.00000, 0.00000, 0.00000, 0.
00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.0…
$ SpecialDay              [3m[38;5;246m<dbl>[39m[23m 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.4, 0.0, 0.8, 0.4, 0.0, 0.4, 0.0, 0.0, 0.0, …
$ Month                   [3m[38;5;246m<chr>[39m[23m "Feb", "Feb", "Feb", "Feb", "
Feb", "Feb", "Feb", "Feb", "Feb", "Feb", "Feb"…
$ OperatingSystems        [3m[38;5;246m<dbl>[39m[23m 1, 2, 4, 3, 3, 2, 2, 1, 2, 2,
1, 1, 1, 2, 3, 1, 1, 1, 2, 2, 2, 3, 3, 2, 2, …
$ Browser                 [3m[38;5;246m<dbl>[39m[23m 1, 2, 1, 2, 3, 2, 4, 2, 2, 4,
1, 1, 1, 5, 2, 1, 1, 1, 2, 4, 2, 3, 2, 4, 2, …
$ Region                  [3m[38;5;246m<dbl>[39m[23m 1, 1, 9, 2, 1, 1, 3, 1, 2, 1,
3, 4, 1, 1, 3, 9, 4, 1, 1, 4, 5, 1, 1, 1, 4, …
$ TrafficType             [3m[38;5;246m<dbl>[39m[23m 1, 2, 3, 4, 4, 3, 3, 5, 3, 2,
3, 3, 3, 3, 3, 3, 3, 4, 3, 4, 1, 3, 5, 3, 1, …
$ VisitorType             [3m[38;5;246m<chr>[39m[23m "Returning_Visitor", "Returni
ng_Visitor", "Returning_Visitor", "Returning_V…
$ Weekend                 [3m[38;5;246m<lgl>[39m[23m FALSE, FALSE, FALSE, FALSE, T
RUE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, …
$ Revenue                 [3m[38;5;246m<lgl>[39m[23m FALSE, FALSE, FALSE, FALSE, F
ALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE…
```

Hide

```
# checking for the statistical summary
summary(df)
```

```
 Administrative   Administrative_Duration Informational    Informational_Duration
ProductRelated
 Min.   : 0.000   Min.   :  -1.00         Min.   : 0.000   Min.   :  -1.00
Min.   :  0.00
 1st Qu.: 0.000   1st Qu.:   0.00         1st Qu.: 0.000   1st Qu.:   0.00
1st Qu.:  7.00
 Median : 1.000   Median :   8.00         Median : 0.000   Median :   0.00
Median : 18.00
 Mean   : 2.318   Mean   :  80.91         Mean   : 0.504   Mean   :  34.51
Mean   : 31.76
 3rd Qu.: 4.000   3rd Qu.:  93.50         3rd Qu.: 0.000   3rd Qu.:   0.00
3rd Qu.: 38.00
 Max.   :27.000   Max.   :3398.75         Max.   :24.000   Max.   :2549.38
Max.   :705.00
 NA's   :14       NA's   :14              NA's   :14       NA's   :14
NA's   :14
 ProductRelated_Duration  BounceRates         ExitRates         PageValues
SpecialDay
 Min.   :   -1.0         Min.   :0.000000   Min.   :0.00000   Min.   :  0.000   Mi
n.   :0.00000
 1st Qu.:  185.0         1st Qu.:0.000000   1st Qu.:0.01429   1st Qu.:  0.000   1s
t Qu.:0.00000
 Median :  599.8         Median :0.003119   Median :0.02512   Median :  0.000   Me
dian :0.00000
 Mean   : 1196.0         Mean   :0.022152   Mean   :0.04300   Mean   :  5.889   Me
an   :0.06143
 3rd Qu.: 1466.5         3rd Qu.:0.016684   3rd Qu.:0.05000   3rd Qu.:  0.000   3r
d Qu.:0.00000
 Max.   :63973.5         Max.   :0.200000   Max.   :0.20000   Max.   :361.764   Ma
x.   :1.00000
 NA's   :14              NA's   :14         NA's   :14
    Month            OperatingSystems    Browser           Region        TrafficType
VisitorType
 Length:12330       Min.   :1.000    Min.   : 1.000   Min.   :1.000   Min.   : 1.0
0   Length:12330
 Class :character   1st Qu.:2.000    1st Qu.: 2.000   1st Qu.:1.000   1st Qu.: 2.0
0   Class :character
 Mode  :character   Median :2.000    Median : 2.000   Median :3.000   Median : 2.0
0   Mode  :character
                    Mean   :2.124    Mean   : 2.357   Mean   :3.147   Mean   : 4.0
7
                    3rd Qu.:3.000    3rd Qu.: 2.000   3rd Qu.:4.000   3rd Qu.: 4.0
0
                    Max.   :8.000    Max.   :13.000   Max.   :9.000   Max.   :20.0
0

  Weekend          Revenue
 Mode :logical   Mode :logical
 FALSE:9462      FALSE:10422
 TRUE :2868      TRUE :1908
```

Hide

```
 # determine the dimensions of the dataset
dim(df)
```

```
[1] 12330    18
```

> From the chunk above, it is evident that the dataset contains 12,330 observations of 18 variables

Hide

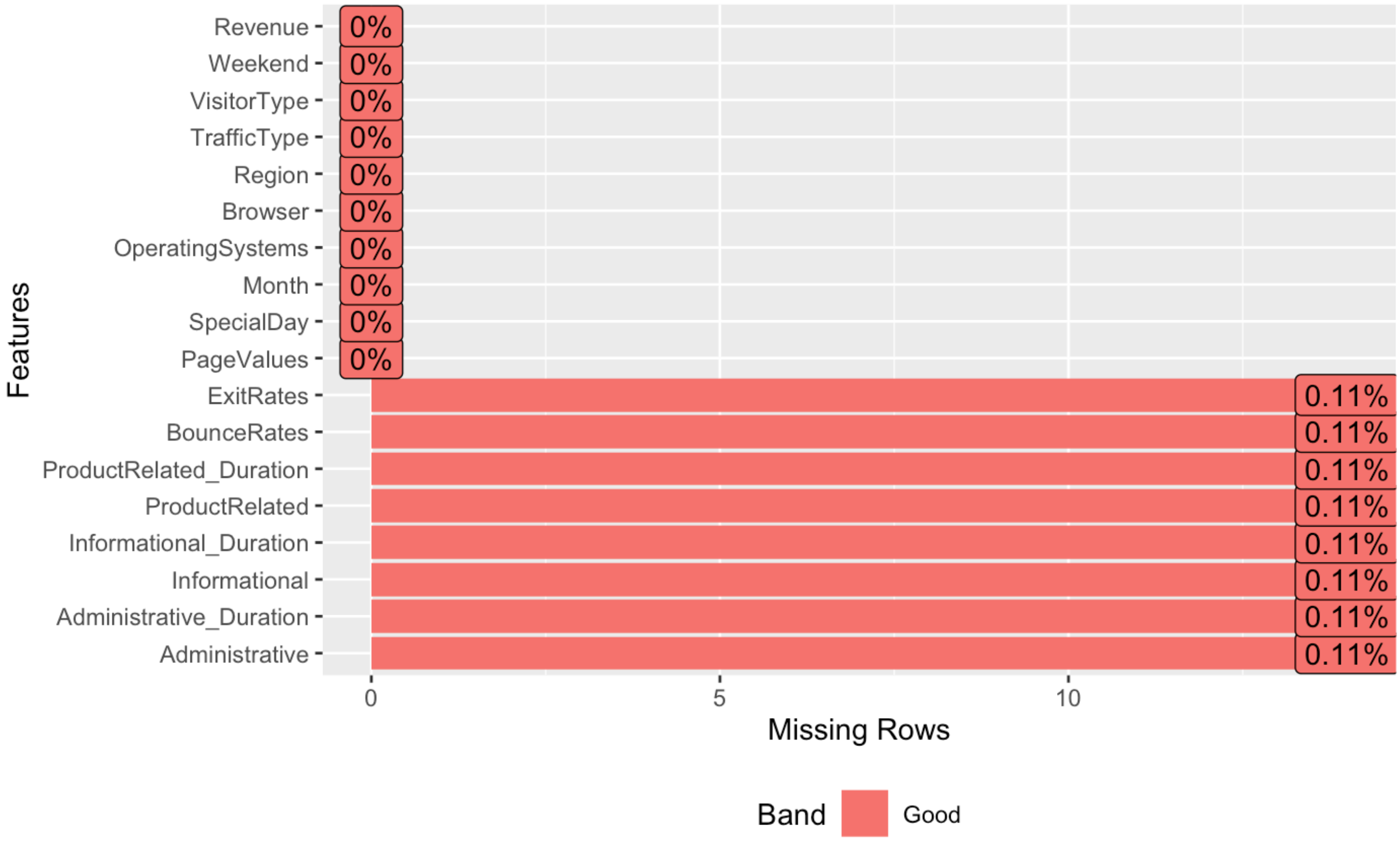
```
# checking if there exists null values by calculating the sum of the null values p
er column
colSums((is.na(df)))
```

```
         Administrative Administrative_Duration           Informational  Informati
onal_Duration
                     14                      14                      14
14
         ProductRelated ProductRelated_Duration             BounceRates
ExitRates
                     14                      14                      14
14
             PageValues              SpecialDay                   Month         Ope
ratingSystems
                      0                       0                       0
0
                Browser                  Region             TrafficType
VisitorType
                      0                       0                       0
0
                Weekend                 Revenue
                      0                       0
```

> We can see that about 8 variables have the same number of missing values

Hide

```
# a plot showing missing values
library(DataExplorer) # simplifies and automates EDA processes and aids in report
generation
plot_missing(df)
```



Hide

```
# Dropping missing values
df = na.omit(df)
```

Hide

```
colSums((is.na(df)))
```

```
        Administrative Administrative_Duration           Informational  Informati
onal_Duration
                     0                       0                       0
0
        ProductRelated ProductRelated_Duration             BounceRates
ExitRates
                     0                       0                       0
0
            PageValues              SpecialDay                   Month         Ope
ratingSystems
                     0                       0                       0
0
               Browser                  Region             TrafficType
VisitorType
                     0                       0                       0
0
               Weekend                 Revenue
                     0                       0
```

> The missing values have been successfully omitted

Hide

```
# Checking for duplicated data
anyDuplicated(df)
```

```
[1] 159
```

> The dataset is seen to contain a number of duplicates which will mess up with analysis and the prediction model, they will therefore be dealt with in th next chunk

Hide

```
# Dropping duplicates
df = distinct(df)

# confirming whether the drop was successful
anyDuplicated(df)
```

```
[1] 0
```

> Our dataset if now free of redundant data.

Hide

```
# Checking the type of the dataset
class(df)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

Hide

```
# Changing the type of the loaded dataset to a dataframe
df = as.data.frame(df)
class(df)
```

```
[1] "data.frame"
```

Hide

```
library(magrittr) # offers a set of operators that provide semantics that will imp
rove code
# Checking the datatypes for each column
columns = colnames(df)
for (column in seq(length(colnames(df)))){
    print(columns[column])
    print(class(df[, column]))
    cat('\n')
}
```

```
[1] "Administrative"
[1] "numeric"

[1] "Administrative_Duration"
[1] "numeric"

[1] "Informational"
[1] "numeric"

[1] "Informational_Duration"
[1] "numeric"

[1] "ProductRelated"
[1] "numeric"

[1] "ProductRelated_Duration"
[1] "numeric"
```

```
[1] "BounceRates"
[1] "numeric"

[1] "ExitRates"
[1] "numeric"

[1] "PageValues"
[1] "numeric"

[1] "SpecialDay"
[1] "numeric"

[1] "Month"
[1] "character"

[1] "OperatingSystems"
[1] "numeric"

[1] "Browser"
[1] "numeric"

[1] "Region"
[1] "numeric"

[1] "TrafficType"
[1] "numeric"

[1] "VisitorType"
[1] "character"

[1] "Weekend"
[1] "logical"

[1] "Revenue"
[1] "logical"
```

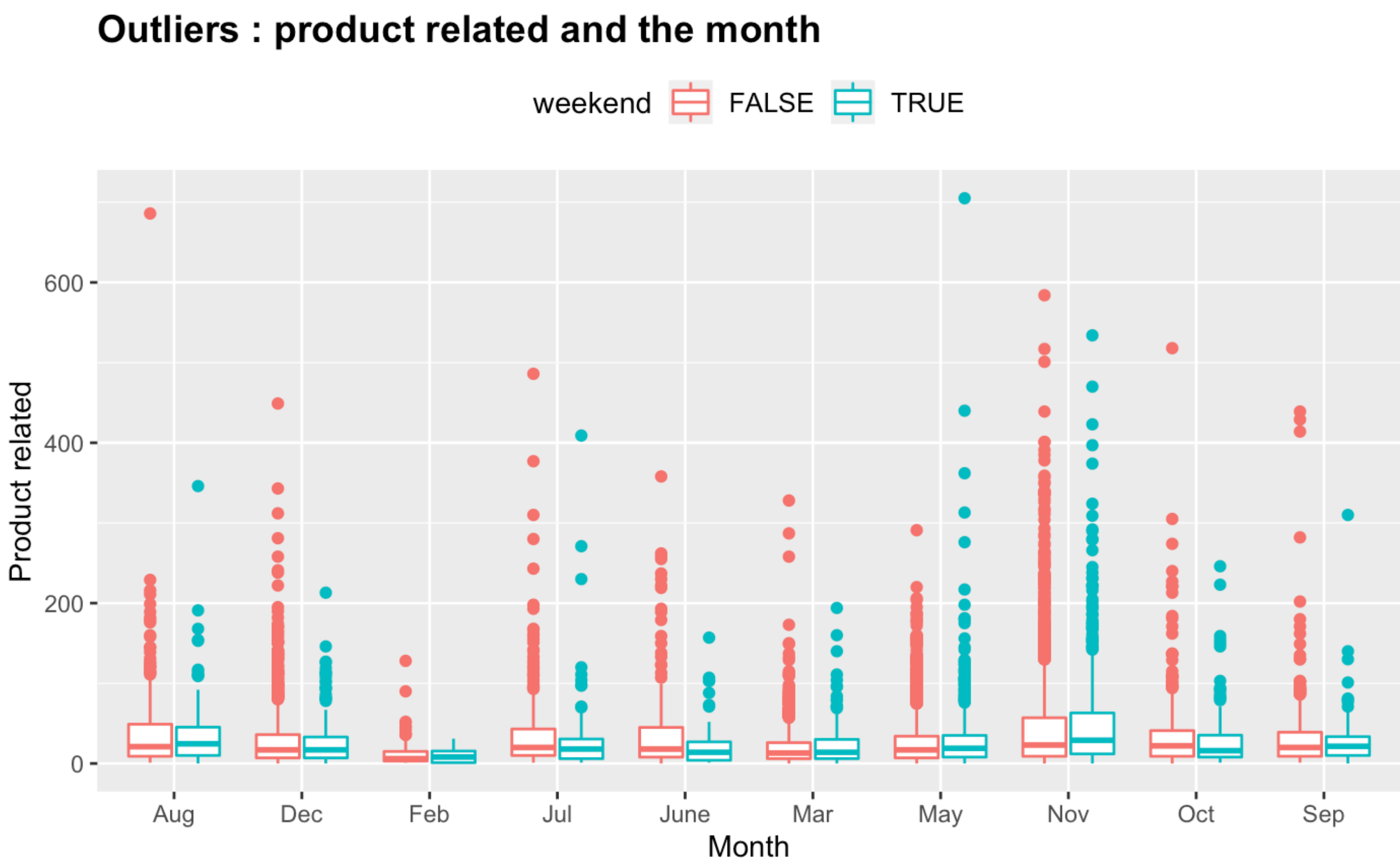
Hide

```
# Cleaning column names, by making them uniform
colnames(df) = tolower(colnames(df))
```

> Now that our dataset is free of missing values and duplicates, we will go ahead and do a visual check on outliers in the dataset

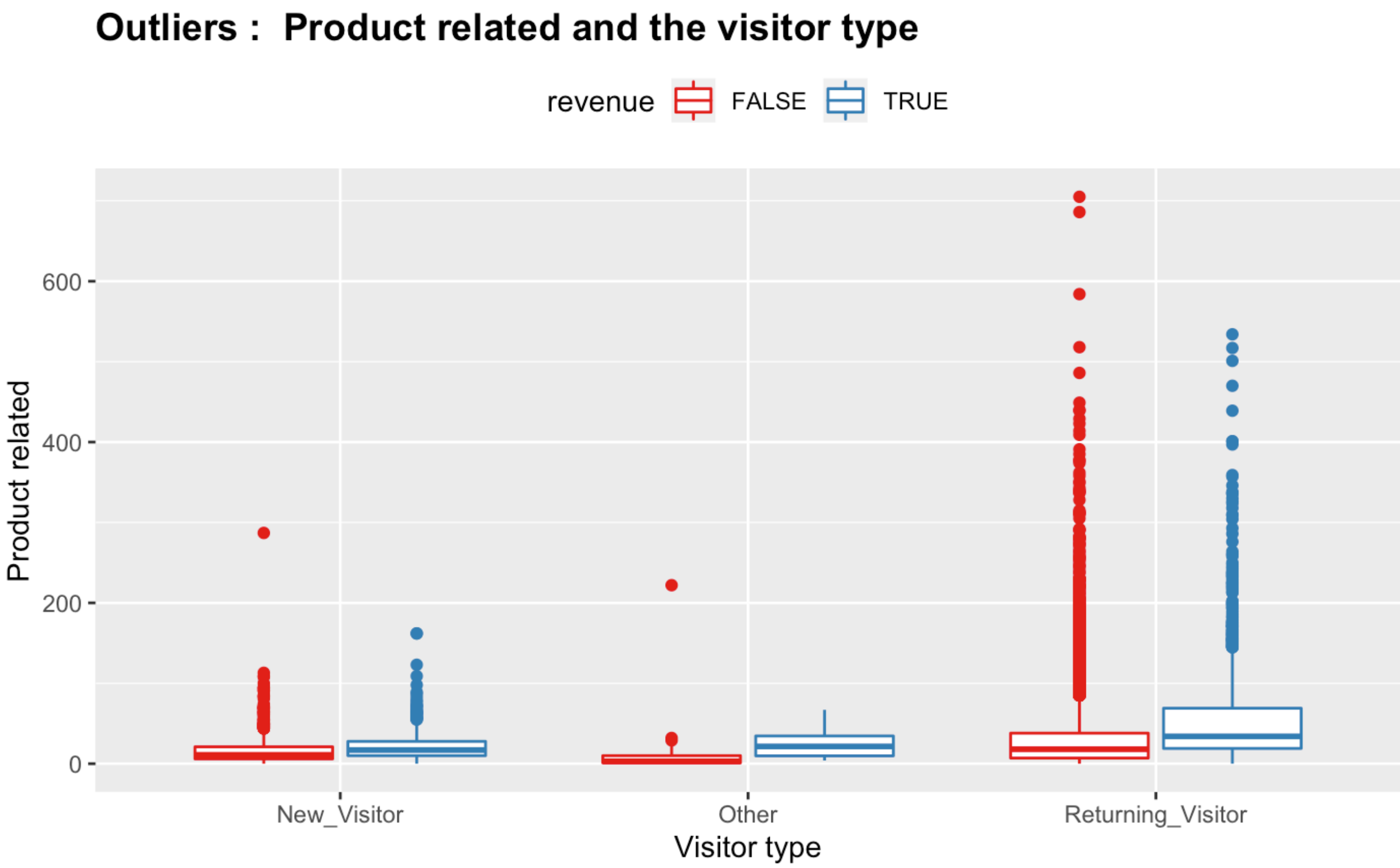
Hide

```
library(ggplot2) # aids in creation of data visuals with the help of grammar of gr
aphics
# Plotting boxplots
options(repr.plot.width = 11, repr.plot.height = 5)
ggplot(df, aes(month, productrelated, col = weekend)) +
  geom_boxplot() +
  labs(x = 'Month', y = 'Product related', title = 'Outliers : product related and
the month') +
  theme(legend.position = 'top', legend.text = element_text(size = 10),
        plot.title = element_text(size = 14, color = 'black', face = 'bold'))
```



Hide

```
# Plotting boxplots to check for outliers
options(repr.plot.width = 7, repr.plot.height = 5)
ggplot(df, aes(visitortype, productrelated, col = revenue)) +
  geom_boxplot() +
  labs(x = 'Visitor type', y = 'Product related', title = 'Outliers :  Product rel
ated and the visitor type') +
  scale_color_brewer(palette = 'Set1') +
  theme(legend.position = 'top',
        plot.title = element_text(size = 14, color = 'black', face ='bold'))
```



Exploraroty data analyisis

> This is where we explore the data so as to: * maximize insights on the data set * uncover underlying structure * extract important variables * test underlying assumptions * develop models with great explanatory predictive power * determine optimal factor settings

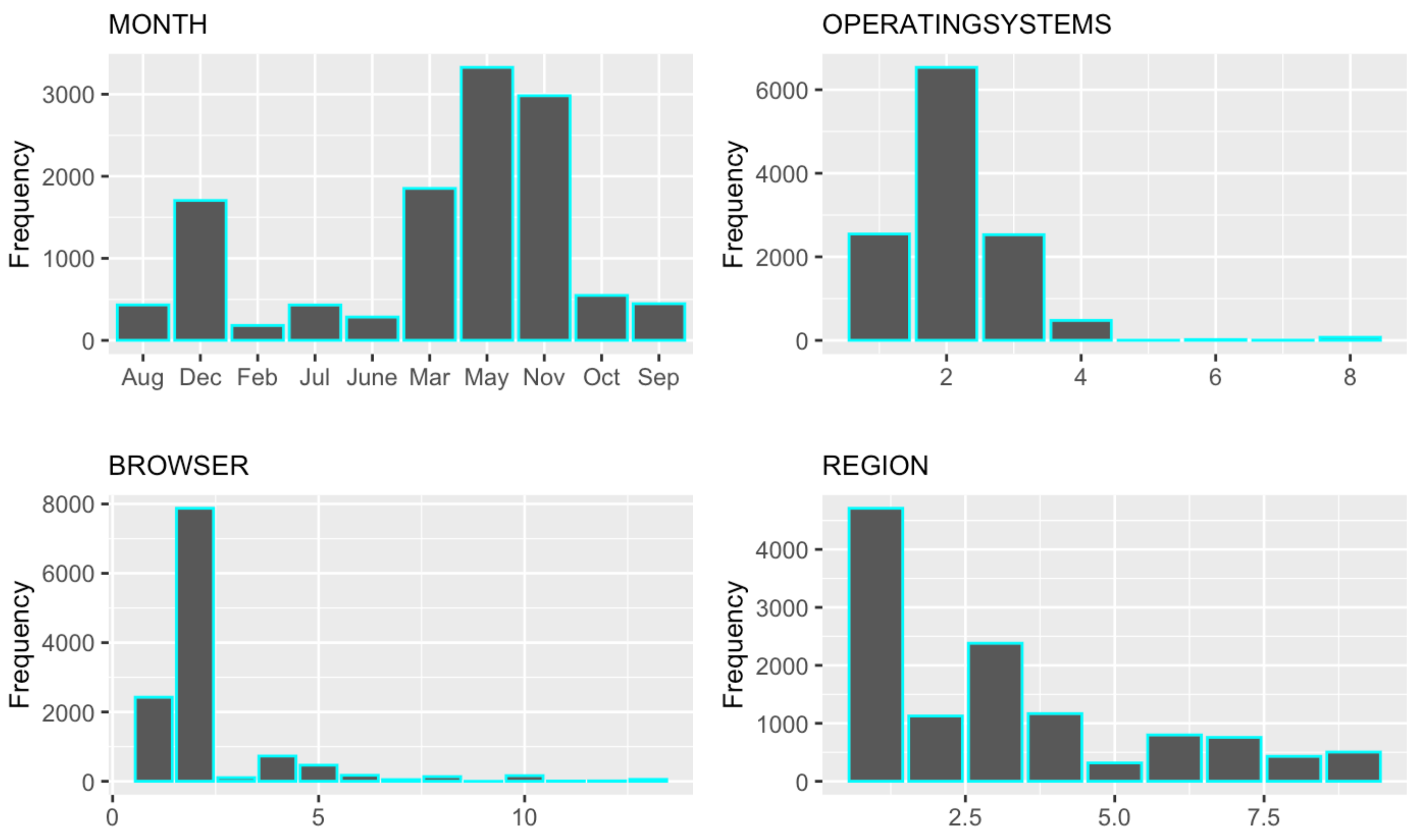
> Here we will perform : * univariate analysis * bivariate analysis * multivariate analysis

> In EDA we have both graphical and non-graphical anaylyis , where non graphical contains Measures of central tendancies :(Mean, mode and median for numerical data and Mode for categorical data) and Measures of dispersion. Because the summary function did most of the non graphical analysis, this section will be densley populated with visaulizations and their analyses

Univariate analysis

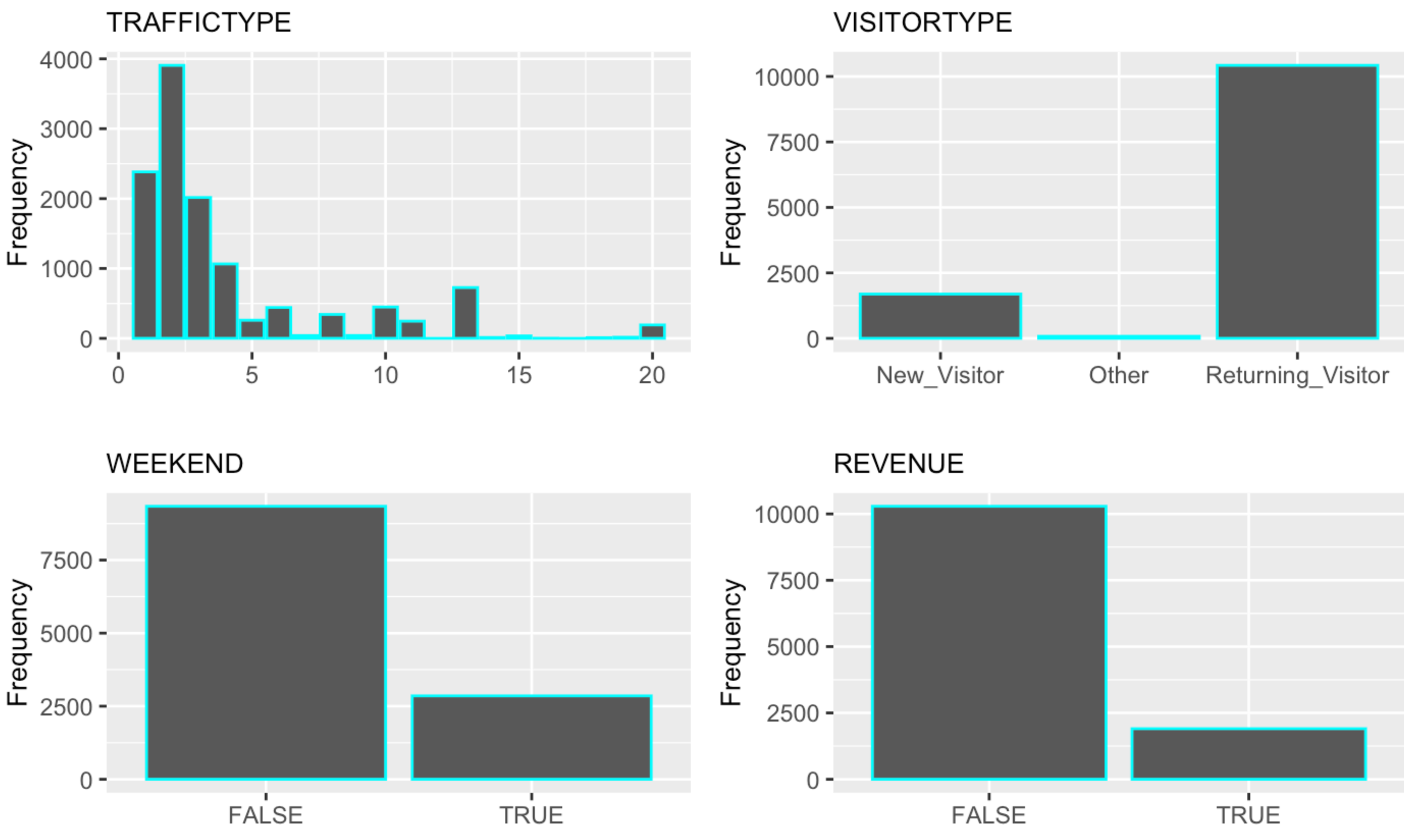
Hide

```
library(gridExtra) # provides a number of user level functions to work with grid g
raphics

# Plotting bar plots showing frequency for each of variables
# we will plot the first 4  in this current chunk then the next for on the next ch
unk to avoid crampling up and untidyness

fac_cols = c('month', 'operatingsystems',   'browser',  'region')

columns = colnames(select(df, fac_cols))

p = list()
options(repr.plot.width = 10, repr.plot.height = 6)
for (i in 1:4){
  p[[i]] = ggplot(df, aes_string(columns[i])) + geom_bar(color = 'cyan') + labs(y
= 'Frequency', x = '', title = toupper(columns[i])) +
  theme(plot.title = element_text(size = 10),
        axis.title.y = element_text(size = 10))
}

do.call(grid.arrange, p)
```



Hide

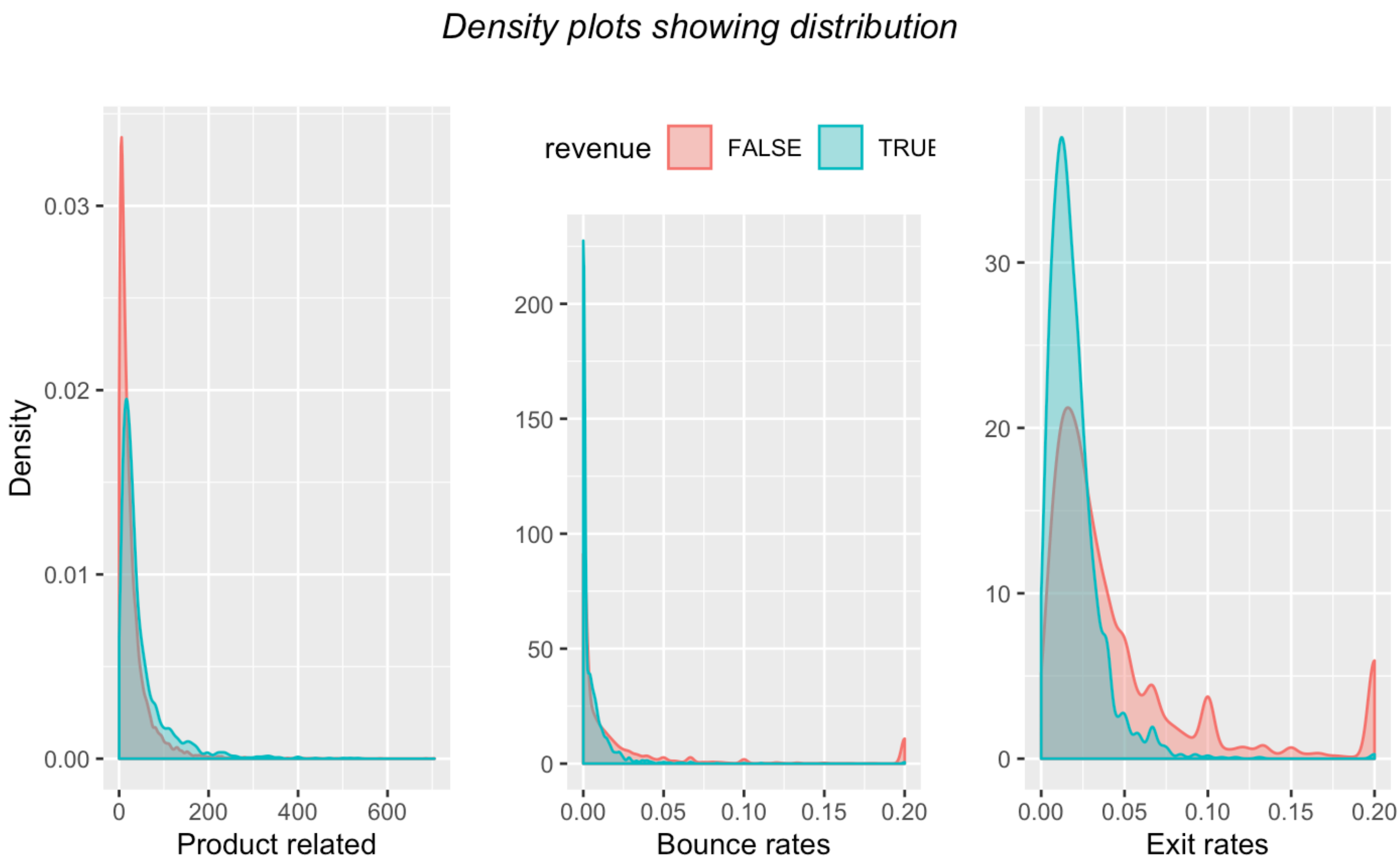
```
fac_cols_2 = c('traffictype',   'visitortype',  'weekend',  'revenue')

columns_2 = colnames(select(df, fac_cols_2))

p_2 = list()
options(repr.plot.width = 10, repr.plot.height = 6)
for (i in 1:4){
  p_2[[i]] = ggplot(df, aes_string(columns_2[i])) + geom_bar(color = 'cyan') + lab
s(y = 'Frequency', x = '', title = toupper(columns_2[i])) +
  theme(plot.title = element_text(size = 10),
        axis.title.y = element_text(size = 10))
}

do.call(grid.arrange, p_2)
```



Now that we have seen the frequency of each column in the dataset, we will check for their distribution and determine whether they are negatively or positively skewed

Hide

```
library(grid)
# Plotting density plots to check for distributions
options(repr.plot.width = 11, repr.plot.height = 5)
p1 = ggplot(df, aes(productrelated, col = revenue)) +
  geom_density(aes(fill = revenue), alpha = 0.4) +
  labs(x = 'Product related', y = 'Density', title = '') +
  theme(legend.position = 'none',
        plot.title = element_text(size = 12))

p2 = ggplot(df, aes(bouncerates, col = revenue)) +
  geom_density(aes(fill = revenue), alpha = 0.4) +
  labs(x = 'Bounce rates', y = '', title = '') +
  theme(legend.position = 'top')

p3 = ggplot(df, aes(exitrates, col = revenue)) +
  geom_density(aes(fill = revenue), alpha = 0.4) +
  labs(x = 'Exit rates', y = '', title = '') +
  theme(legend.position = 'none',
        plot.title = element_text(size = 12))

grid.arrange(p1, p2, p3, ncol = 3, top = textGrob("Density plots showing distribut
ion",gp=gpar(fontsize=13,font=3, color = 'black')))
```

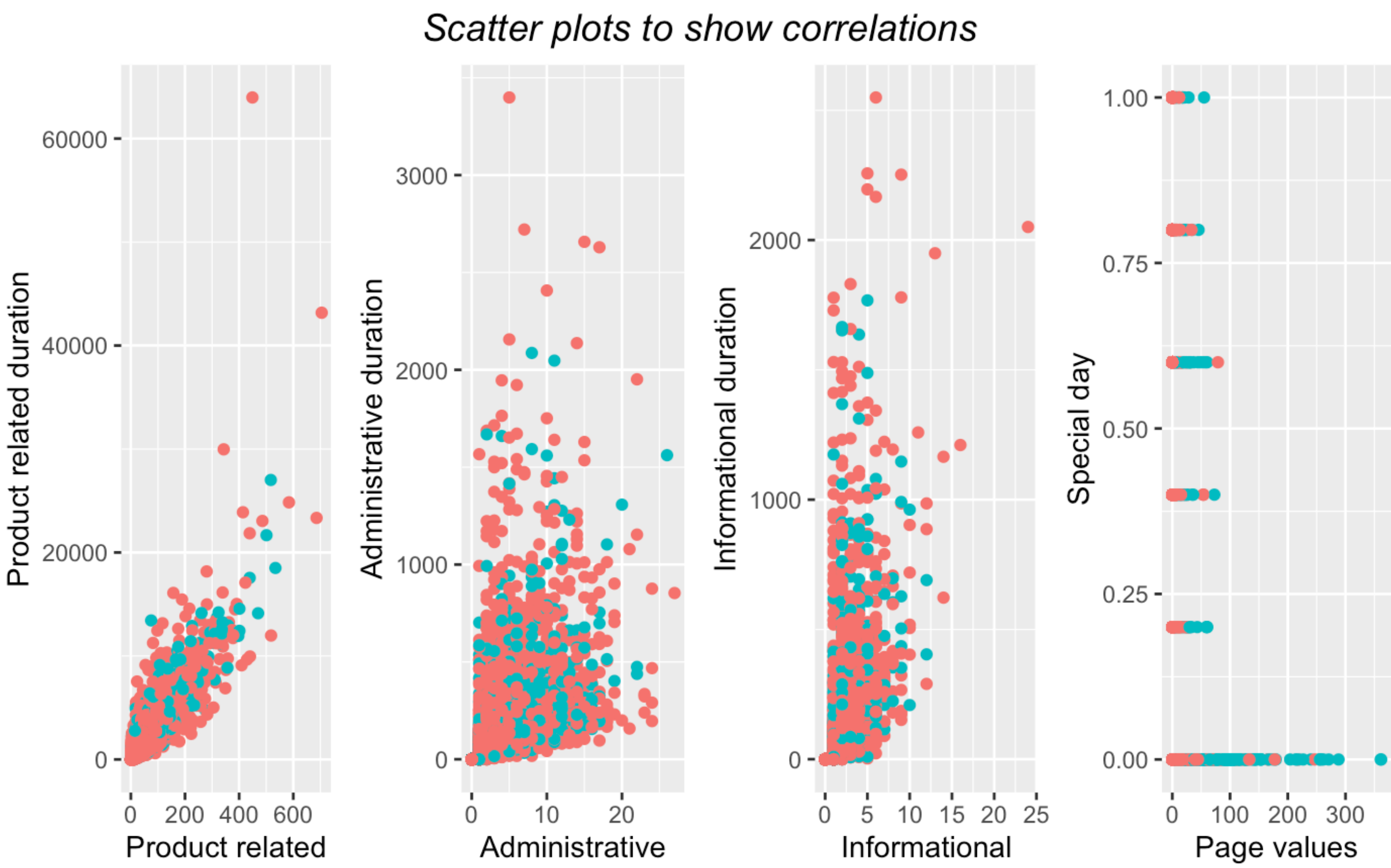


So far, the variables we are working with are all positvely skewed

Bivariate analayisis

Hide

```
# Plotting scatter plots to check for correlations
options(repr.plot.width = 11, repr.plot.height = 5)

p1 = ggplot(df, aes(productrelated, productrelated_duration, col = revenue)) +
    geom_point() + theme(legend.position = 'none') +
    labs(x='Product related', y ='Product related duration')

p2 = ggplot(df, aes(administrative, administrative_duration, col = revenue)) +
    geom_point() + theme(legend.position = 'none') +
    labs(x = 'Administrative', y = 'Administrative duration')

p3 = ggplot(df, aes(informational, informational_duration, col = revenue)) +
    geom_point() + theme(legend.position = 'none') +
    labs(x = 'Informational', y = 'Informational duration')

p4 = ggplot(df, aes(pagevalues, specialday  , col = revenue)) +
    geom_point() + theme(legend.position = 'none') +
    labs(x = 'Page values', y = 'Special day')

grid.arrange(p1, p2, p3, p4, ncol = 4,
             top = textGrob("Scatter plots to show correlations",gp=gpar(fontsize=
14,font=3, color = 'darkmagenta')))
```

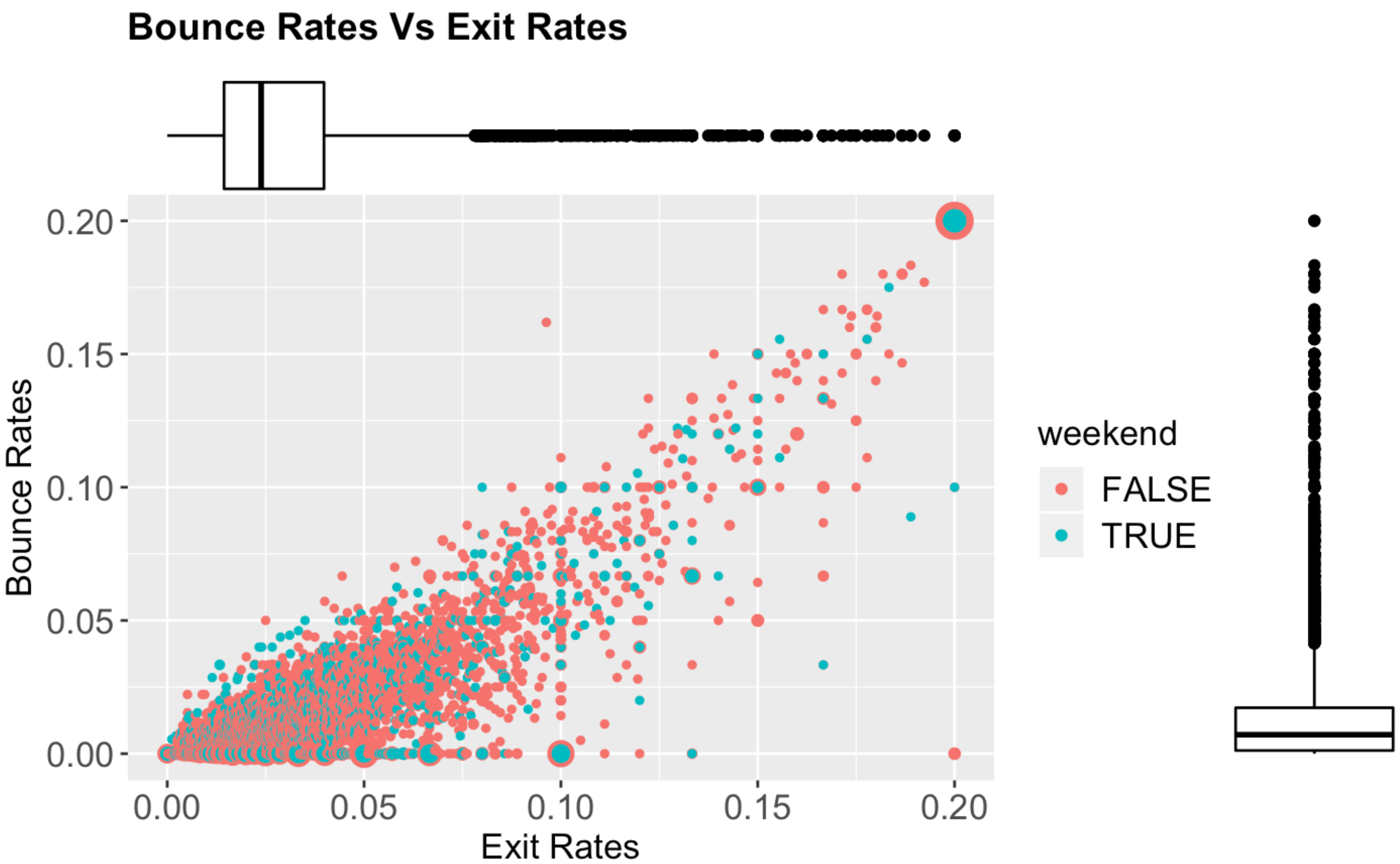


Hide

```
library (ggExtra) # used to add marginal histograms, densityplots and boxplots sca
tter plots

# Plotting scatterplot with marginal density plots (default) or histograms using g
gMarginal

options(repr.plot.width = 7, repr.plot.height = 5)

g = ggplot(data =df, aes(x =exitrates, y = bouncerates, col = weekend)) +
    geom_count(show.legend=c(size=FALSE)) +
    labs(title = 'Bounce Rates Vs Exit Rates', y = 'Bounce Rates', x = 'Exit Rates
') +
    theme(plot.title = element_text(size = 14, face = 'bold'),
             axis.title.x = element_text(size = 13),
             axis.title.y = element_text(size = 13),
             axis.text.x = element_text(size = 13),
             axis.text.y = element_text(size = 13),
             legend.title = element_text(size = 13),
             legend.text = element_text(size = 13))

ggMarginal(g, type = "histogram", fill="transparent")
```

Hide

```
ggMarginal(g, type = "boxplot", fill="transparent")
```



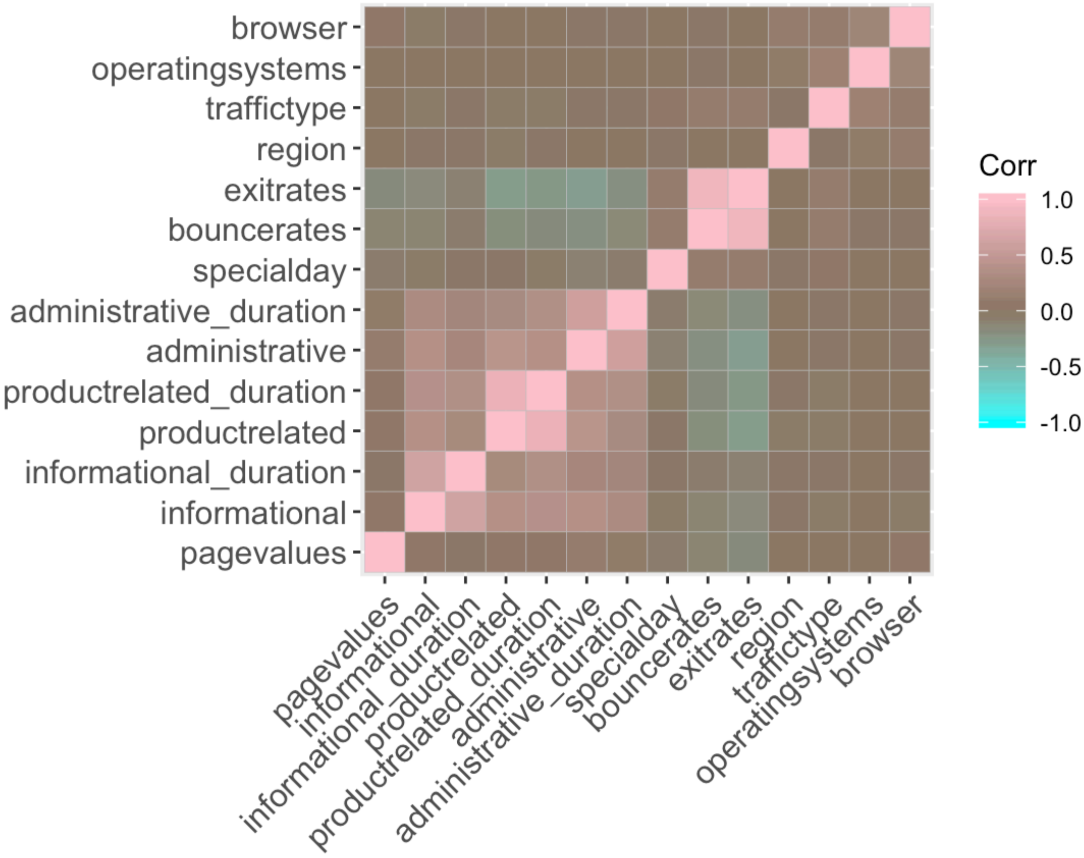
Hide

```
ggMarginal(g, type = "density", fill="transparent")
```

Multivariate analysis

Hide

```
library(ggcorrplot)
# Plotting a correlogram to check for correlations among all th features
options(repr.plot.width = 6, repr.plot.height = 5)

corr = round(cor(select_if(df, is.numeric)), 2)
ggcorrplot(corr, hc.order = T, ggtheme = ggplot2::theme_gray,
   colors = c("cyan", "peachpuff4", "pink"), lab = F)
```



> Now that we have created visual representations of how the dataset looks like we will go ahead and perfom analyses and create the models we need to make the best predictions

K-Means

> Because K-means is a type unsupervised-learning, the class attribute is not needed for the execution of the algorithm. In the 'Understanding the Context' section, we mentioned that the 'Revenue' attribute will be used as the class label. It will be removed and stored in another variable. Afterwards we can normalize the attributes left.

Hide

```
# creating a new dataframe
df.new <- df[,c(1:17)]
colnames(df.new)
```

```
 [1] "administrative"          "administrative_duration" "informational"
"informational_duration"
 [5] "productrelated"          "productrelated_duration" "bouncerates"
"exitrates"
 [9] "pagevalues"              "specialday"              "month"
"operatingsystems"
[13] "browser"                 "region"                  "traffictype"
"visitortype"
[17] "weekend"
```

Hide

```
# storing the class attribute in another variable
df.class <- df["revenue"]
colnames(df.class)
```

```
[1] "revenue"
```

Hide

```
# Normalising the data so that no particular attribute has more impact on clusteri
ng than others
normalize <- function(x){
  return ((x-min(x)) / (max(x)-min(x)))
}
df.new$administrative <- normalize(df.new$administrative)
df.new$administrative_duration <- normalize(df.new$administrative_duration)
df.new$infromational <- normalize(df.new$informational)
df.new$infromational_duration <- normalize(df.new$informational_duration)
df.new$productrelated <- normalize(df.new$productrelated)
df.new$productrelated_duration <- normalize(df.new$productrelated_duration)
df.new$bouncerates<- normalize(df.new$bouncerates)
df.new$exitrates<- normalize(df.new$exitrates)
df.new$pagevalues<- normalize(df.new$pagevalues)
df.new$specialday<- normalize(df.new$specialday)
df.new$opertaingsystem<- normalize(df.new$operatingsystems)
df.new$browser<- normalize(df.new$browser)
df.new$region<- normalize(df.new$region)
df.new$traffictype<- normalize(df.new$traffictype)

head(df)
```

	administrative <dbl>	administrative_duration <dbl>	informational <dbl>	informational_duration <dbl>
1	0	0	0	0
2	0	0	0	0

3	0	-1	0	-1
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0

6 rows | 1-6 of 18 columns

Hide

```
NA
```

Hide

```
# Storing numeric values in a new variable
df.clust <- df.new [,c("administrative","administrative_duration","informational",
"informational_duration","productrelated","productrelated_duration","bouncerates",
"exitrates","pagevalues","specialday","operatingsystems","browser","region","traff
ictype")]

# Apply k means clustering algorithm with a number of centroids k
result<- kmeans(df.clust,3)
```

Hide

```
# preview the number of records in each cluster

result$size
```

```
[1]   116 11530   553
```

Hide

```
# get the value of cluster center datapoint value(k centers for k)
result$centers
```

```
  administrative administrative_duration informational informational_duration prod
uctrelated
1     0.22701149              0.08298299     4.4655172            1156.40264
0.14458303
2     0.07877357              0.02160253     0.3204683               8.72358
0.04184225
3     0.22182037              0.06867505     3.6057866             344.04175
0.10038347
  productrelated_duration bouncerates exitrates pagevalues specialday operatingsys
tems    browser    region
1              0.08241265  0.04112800 0.1182392 0.01846520 0.03793103         2.12
0690 0.10991379 0.2198276
2              0.01701258  0.10601490 0.2132804 0.01603853 0.06374675         2.12
5412 0.11389853 0.2709128
3              0.04471927  0.03621331 0.1053481 0.02469705 0.03001808         2.10
3074 0.09885473 0.2429928
  traffictype
1   0.1279492
2   0.1632081
3   0.1400019
```

Hide

```
# getting cluster vector that shows the cluster where each record falls
result$cluster
```

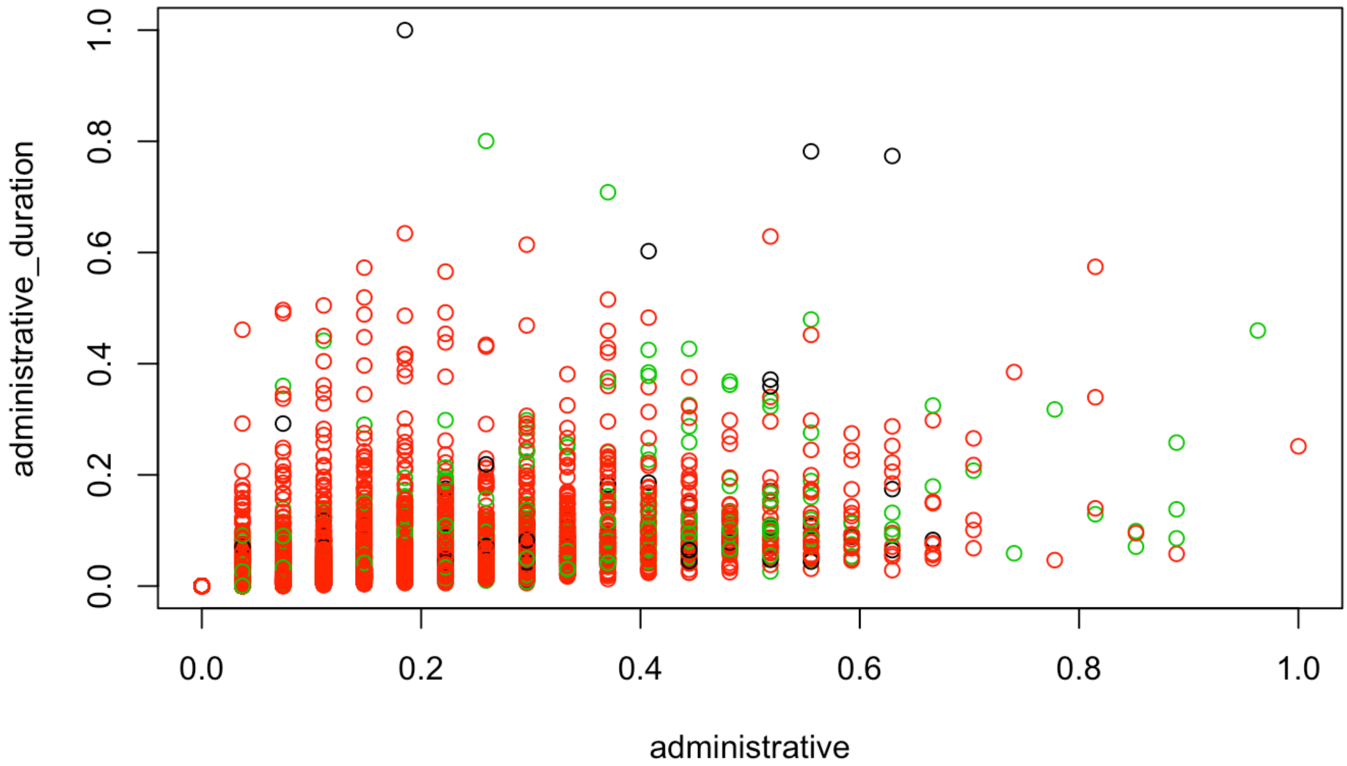
```
   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
  [53] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [105] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [157] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [209] 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2 2 2 2 1 2 2 2 2
2 3 2 2 2 2 2 2 2 2 2 2 2 2
 [261] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [313] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [365] 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [417] 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [469] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 3
2 2 3 2 2 2 2 2 2 2 2 2 2 2
 [521] 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 3 2 2 2 2
 [573] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [625] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [677] 2 2 2 2 3 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 3
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [729] 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 3 2 2 3 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [781] 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [833] 2 2 1 2 2 2 3 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 3 2
 [885] 2 2 2 2 2 2 2 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [937] 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [989] 2 2 2 2 2 2 2 2 2 2 2 2
 [ reached getOption("max.print") -- omitted 11199 entries ]
```

Hide

```
# visualize the clustering results
par(mfrow = c(2,2), mar = c(5,4,2,2))
```

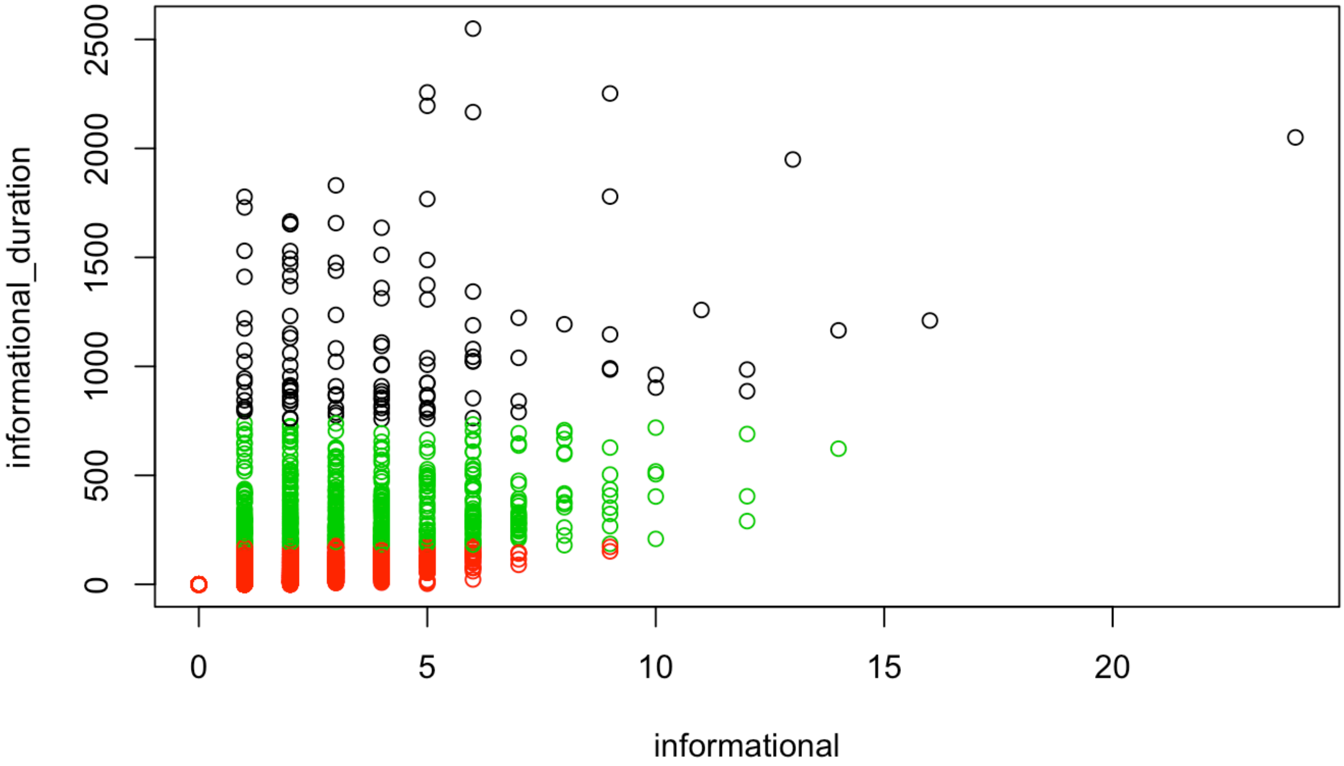
Hide

```
# Plot dist in clusters
plot(df.clust[c(1,2)], col = result$cluster)
```



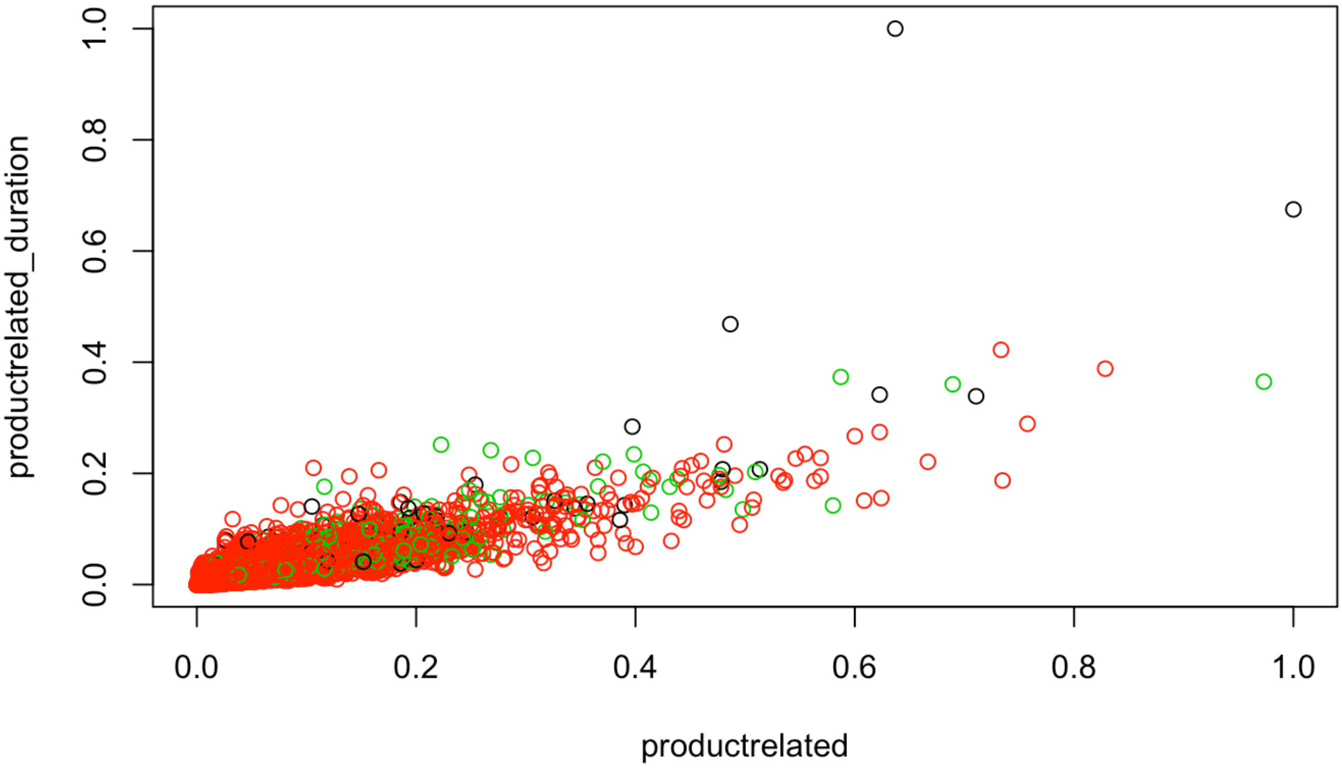
Hide

```
# Plot dist in clusters
plot(df.clust[c(3,4)], col = result$cluster)
```



Hide

```
# Plot dist in clusters
plot(df.clust[c(5,6)], col = result$cluster)
```



Hide

```
NA
NA
```

Hide

```
df.class[] <- lapply(df.class, as.numeric)
sapply(df.class,class)
```

```
  revenue
"numeric"
```

Hide

```
#install.packages('useful')
#library(useful)

#df.clust[] <- lapply(df.clust, as.numeric)
#plot(df.clust[c(1,2)], col = df.class)
```

Hide

```
# result table
#table(result$cluster,df.class)
```

Hide

```
# accuracy score
mean(df.new == result$cluster)
```

```
[1] 0.02897779
```

Hierachical Clustering

Hide

```
# Compute descriptive stats
desc_stats <- data.frame(
  Min = apply(df.clust, 2, min),    # minimum
  Med = apply(df.clust, 2, median), # median
  Mean = apply(df.clust, 2, mean),  # mean
  SD = apply(df.clust, 2, sd),      # Standard deviation
  Max = apply(df.clust, 2, max)     # Maximum
)
desc_stats <- round(desc_stats, 1)
head(desc_stats)
```

	Min <dbl>	Med <dbl>	Mean <dbl>	SD <dbl>	Max <dbl>
administrative	0	0	0.1	0.1	1.0
administrative_duration	0	0	0.0	0.1	1.0
informational	0	0	0.5	1.3	24.0
informational_duration	-1	0	34.8	141.5	2549.4
productrelated	0	0	0.0	0.1	1.0
productrelated_duration	0	0	0.0	0.0	1.0

6 rows

Hide

```
# scale the datafarm to make the variables comparable
# entails transforming variables such that they have a mean of 0 and std dev of 1
# because we dont want the hierachical clustering result to depend on an arbitrary
variable unit

df.clust <- scale(df.clust)
head(df.clust)
```

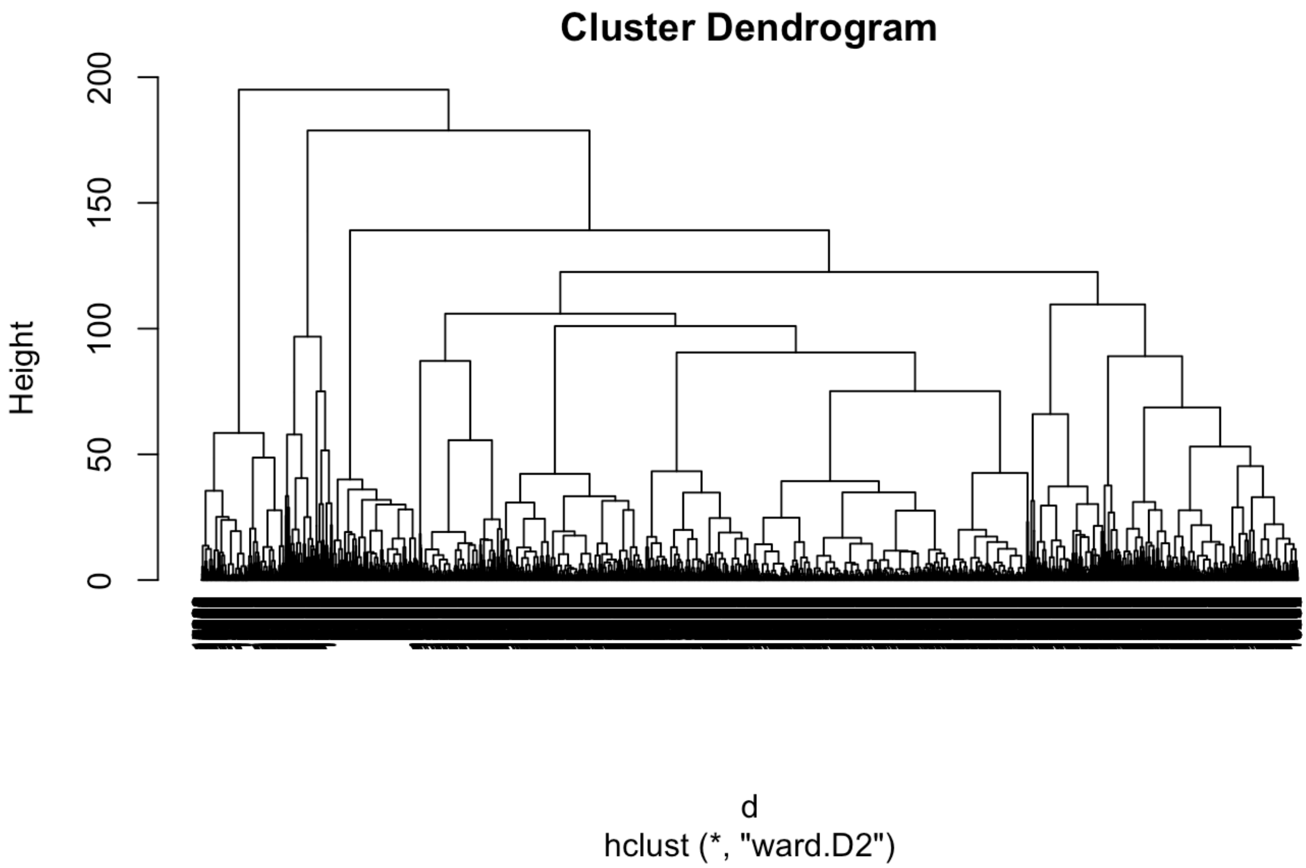
```
     administrative administrative_duration informational informational_duration p
roductrelated
[1,]     -0.7025315              -0.4601081    -0.3988128             -0.2462725
-0.6963635
[2,]     -0.7025315              -0.4601081    -0.3988128             -0.2462725
-0.6739424
[3,]     -0.7025315              -0.4657410    -0.3988128             -0.2533417
-0.6963635
[4,]     -0.7025315              -0.4601081    -0.3988128             -0.2462725
-0.6739424
[5,]     -0.7025315              -0.4601081    -0.3988128             -0.2462725
-0.4945739
[6,]     -0.7025315              -0.4601081    -0.3988128             -0.2462725
-0.2927843
     productrelated_duration  bouncerates  exitrates pagevalues specialday operati
ngsystems    browser
[1,]              -0.6289343  3.954699721  3.4273070 -0.3190356 -0.3103105       -
1.2396607 -0.7939682
[2,]              -0.5955997 -0.450343788  1.2650121 -0.3190356 -0.3103105       -
0.1371074 -0.2093703
[3,]              -0.6294551  3.954699721  3.4273070 -0.3190356 -0.3103105        
2.0679992 -0.7939682
[4,]              -0.6275453  0.650917089  2.1299300 -0.3190356 -0.3103105        
0.9654459 -0.2093703
[5,]              -0.3020990 -0.009839437  0.1838646 -0.3190356 -0.3103105        
0.9654459  0.3752276
[6,]              -0.5486101 -0.102577188 -0.3661929 -0.3190356 -0.3103105       -
0.1371074 -0.2093703
         region trafficctype
[1,] -0.8962939 -0.76562243
[2,] -0.8962939 -0.51660683
[3,]  2.4336556 -0.26759123
[4,] -0.4800502 -0.01857564
[5,] -0.8962939 -0.01857564
[6,] -0.8962939 -0.26759123
```

Hide

```
# clustering
# use the R function hclust() for hierarchical clustering
# 1st use dist() function to compute the Euclidean Distance btwn observs
# d will be the 1st argument in the hclust() function dissimilarity matrix

d<-dist(df.clust,method = "euclidean")
res.hc <- hclust (d,method = "ward.D2")
```

Hide

```
# plot obtained dendogram
plot(res.hc,cex = 0.6,hang = -1)
```



Hide

```
# checking for accuracy
mean(df.clust== result$cluster)
```

```
[1] 0
```