EDA (Exploratory Data Analysis) is one of the key steps in any Data Science Project. The better the EDA is the better the Feature Engineering could be done. From Modelling to Communication, EDA has got much more hidden benefits that aren’t often emphasised while beginners start while teaching Data Science for beginners.

**The Problem**

That said, EDA is also one of the areas of the Data Science Pipeline where a lot of manual code is written for different types of plots and different types for inference. Let’s you’d want to visualize a bar plot of a categorical variable and you’d want to visualize a histogram of a continuous variable to understand their distribution. All these things increase the number of lines of code and also there by number of lines of code which could be time consuming if you’re participating in Hackathons or Online Competitions like Kaggle where time-bound response is usually required to move ahead in the leaderboard.

**The Solution**

That’s where the tools of Automated EDA comes very handy and one such popular tool for Automated EDA in R is DataExplorer by **Boxuan Cui**.

**DataExplorer**

The stable version of DataExplorer can be installed from CRAN.

install.packages("DataExplorer")

And if you’d like to try on the development version:

if (!require(devtools)) install.packages("devtools")

devtools::install\_github("boxuancui/DataExplorer", ref = "develop")

**Automating EDA – Get started**

Before we start with EDA, We should first get the data that we would like explore. In this case, [We’ll use data generated by fakir](https://www.programmingwithr.com/how-to-generate-meaningful-fake-data-for-learning-experimentation-and-teaching/)

library(fakir)

library(tidyverse)

library(DataExplorer)

web <- fakir::fake\_visits()

glimpse(web)

## Observations: 365

## Variables: 8

## $ timestamp 2017-01-01, 2017-01-02, 2017-01-03, 2017-01-04, 2017-…

## $ year 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, 2017, …

## $ month 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …

## $ day 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,…

## $ home 352, 203, 103, 484, 438, NA, 439, 273, 316, 193, 322, …

## $ about 176, 115, 59, 113, 138, 75, 236, 258, 206, 260, NA, 29…

## $ blog 521, 492, 549, 633, 423, 478, 364, 529, 320, 315, 578,…

## $ contact NA, 89, NA, 331, 227, 289, 220, 202, 367, 369, 241, 28…

# year,month,day to factor

web$year <- as.factor(web$year)

web$month <- as.factor(web$month)

web$day <- as.factor(web$day)

To go with glimpse(), DataExplorer itself has got a function called introduce()

introduce(web)

## # A tibble: 1 x 9

## rows columns discrete\_columns continuous\_colu… all\_missing\_col…

##

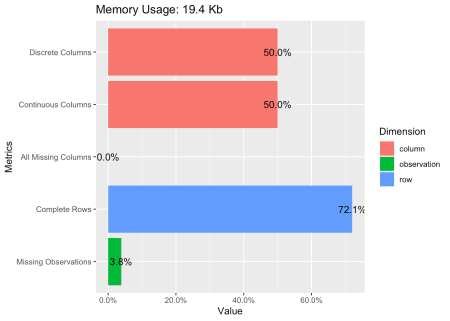
## 1 365 8 4 4 0

## # … with 4 more variables: total\_missing\_values ,

## # complete\_rows , total\_observations , memory\_usage

The same introduce() could also be plotted in a pretty graph.

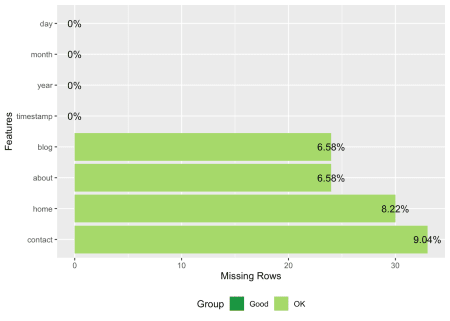
plot\_intro(web)



**Automating EDA – Missing**

Personally, The most useful function of DataExplorer is to plot\_missing() values.

plot\_missing(web)



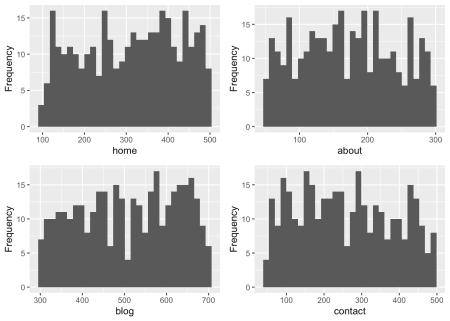
That’s so handy that I don’t have to copy paste any custom function from SO or my previous code.

**Automating EDA – Continuous**

As with most EDA on Continuous variables (numbers), We’ll start of with Histogram that can help us understand the underlying distributions.

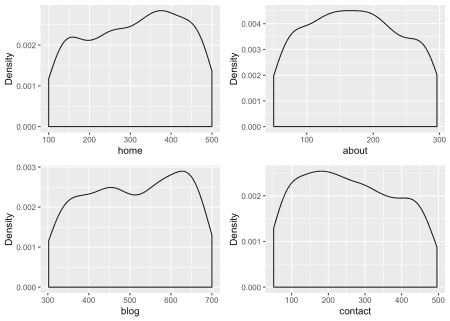
And that’s just one function plot\_histogram()

DataExplorer::plot\_histogram(web)



And a similar function for density plot plot\_density()

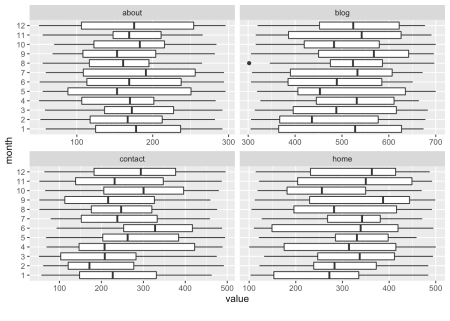
plot\_density(web)



That’s all univariate and if we get on with `bivariate1, we can start off with boxplots with respect to a categorical variable.

plot\_boxplot(web, by= 'month', ncol = 2)

## Warning: Removed 111 rows containing non-finite values (stat\_boxplot).



And, the super-useful correlation plot.

plot\_correlation(web, cor\_args = list( 'use' = 'complete.obs'))

## 2 features with more than 20 categories ignored!

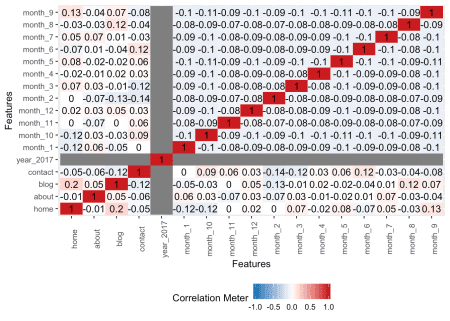
## timestamp: 365 categories

## day: 31 categories

## Warning in cor(x = structure(list(home = c(352L, 203L, 103L, 484L, 438L, :

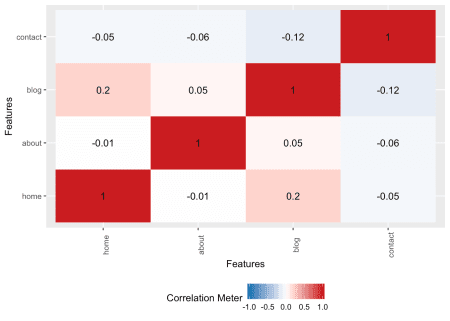
## the standard deviation is zero

## Warning: Removed 32 rows containing missing values (geom\_text).



If in case, you want the correlation plot to be plotted only for continuous variables:

plot\_correlation(web, type = 'c',cor\_args = list( 'use' = 'complete.obs'))



Well, that’s how simple it’s to make a bunch of plots for continous variables.

**Automating EDA – Categorical**

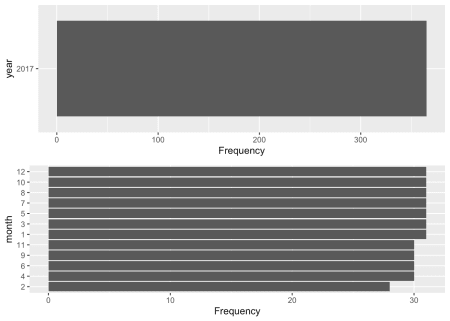
A bar plot to combine a categorical and a continuous variable. By default (with no with value), plot\_bar() plots the categorical variable against the frequency/count.

plot\_bar(web,maxcat = 20, parallel = TRUE)

## 2 columns ignored with more than 20 categories.

## timestamp: 365 categories

## day: 31 categories



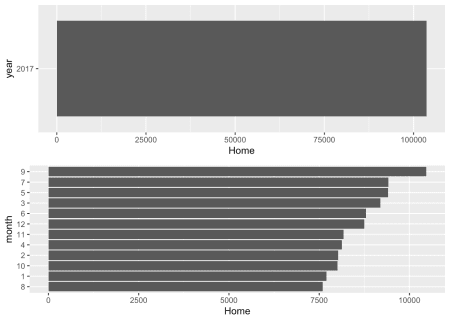
Also, We’ve got an option to specify the name of the continuous variable to be summed up.

plot\_bar(web,with = c("home"), maxcat = 20, parallel = TRUE)

## 2 columns ignored with more than 20 categories.

## timestamp: 365 categories

## day: 31 categories



**EDA Report**

While those above ones are specific functions for a specific type of plot (but plotted for the whole dataset) making EDA a very quick process.

create\_report()

create\_report() helps us in generating an output report combining all the required plots for different types of variables.

**Plot Aesthetics**

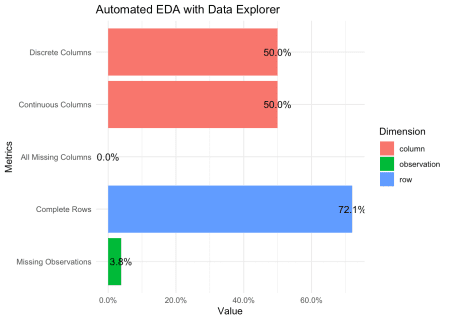
It’s worthy enough to mention that these ggplots that are built aren’t the final version as DataExplorer allows us to supply ggtheme theme name and theme\_config to pass on the theme paramaters. Also functions like plot\_box() or plot\_histgram() also takes in the plot-specific arguments. For more details on this check out relevant help files.

plot\_intro(web,

ggtheme = theme\_minimal(),

title = "Automated EDA with Data Explorer",

)



**Summary**

DataExplorer is extremely handy for automating EDA in a lot of use-cses like Missing Values reporting in an ETL process or Basic EDA in a Hackathon. It’s definitely another generalist-tool that could be customized for better usage.