

Eltecon Data Science Course by Emarsys

EDA and Data Visualization

András Bérczi

September 15, 2021

About me

- Background in Economics
- Works as Data Scientist @ Emarsys

Today's topics

- Coding guidelines
- Exploratory Data Analysis (EDA)
- Data visualization with ggplot

Why writing good code is important?

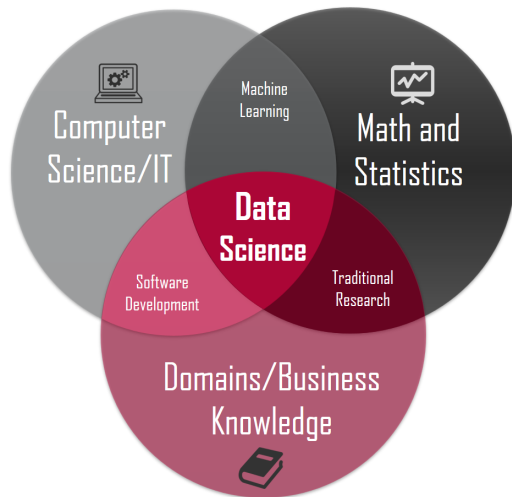


Figure 1: <https://www.inf.elte.hu/en/content/data-science-in-computer-science-msc.t.1732?m=361>

Try to make your code 'clean'!

- **D**(on't) **R**(epeat) **Y**(ourself)!
- Make your code readable!
- Use style guides!

Keep it DRY!

```
average_of_A <- sum(data['A_column']) / nrow(data['A_column'])  
variance_of_A <- sum((data['A_column'] - average_of_A) ^ 2) /  
  (nrow(data['A_column']) - 1)
```

... some line of codes ...

```
average_of_B <- sum(data['B_column']) / nrow(data['B_column'])  
variance_of_B <- sum((data['B_column'] - average_of_B) ^ 2) /  
  (nrow(data['B_column']) - 1)
```

... some line of codes ...

```
average_of_C <- sum(data['C_column']) / nrow(data['C_column'])  
variance_of_C <- sum((data['C_column'] - average_of_C) ^ 2) /  
  (nrow(data['C_column']) - 1)
```

Move repeating code into functions!

```
# Define a function
variance <- function(values) {
  N <- length(values)
  average <- sum(values) / N

  return ((values - N) ^ 2) / N
}
```

Use the function instead!

```
variance_of_A <- variance(data['A_column'])
```

... some line of codes ...

```
variance_of_B <- variance(data['B_column'])
```

... some line of codes ...

```
variance_of_C <- variance(data['C_column'])
```


Watch out for messy code!

```
1 dataset <- "customers"
2 if (isDatasetUpToDate(dataset, max_hour_delay = 14 * 24, filepath = "emconnect_data/customers.csv")) {
3   customers <- getListOfAllCustomers()
4 } else {
5   customers <- getListOfAllCustomers(refetch = TRUE)
6   saveRefreshTimeOfDataset(dataset)
7 }
8
9 effect_dt <- sto_performance %>%
10   .[, versions := segment] %>%
11   .[, segment := NULL] %>%
12   calculateEffect("open_rate", by_cols = c("customer_id", "customer_name")) %>%
13
14 if (metric == "first_2_hour_open_rate") {
15   metric <- "open_rate"
16 }
17
18 num_col_name <- paste0("num_", gsub("_rate", "", "open_rate"))
19
20 effect_dt %>%
21   copy() %>%
22   .[, num_delivered := as.numeric(num_delivered_control) + as.numeric(num_delivered_treatment)] %>%
23   .[, total_num := get(str_c(num_col_name, "_control")) + get(str_c(num_col_name, "_treatment"))] %>%
24   setnames(paste0("open_rate", c("_control", "_treatment")), c("control_rate", "treatment_rate")) %>%
25   setnames("open_rate_uplift", "treatment_uplift") %>%
26   .[, sto_uplift := round(treatment_uplift, 4)] %>%
27   .[, .(customer_id, sto_uplift)]
28
29 adoption_metrics_dt <- merge(customers, sto_uplift, by = "customer_id", all.x = TRUE)
30
31 id_col <- ifelse("customer_id" %in% names(adoption_metrics_dt), "customer_id", "customer_name")
32 real_customers <- getFreshListOfAllCustomers(exclude_test = TRUE)[, .SD, .SDcols = id_col]
33
34 lifecycle_for_relevant_customers <- merge(adoption_metrics_dt, real_customers, by = id_col) %>%
35   .[, lifecycle := dplyr::case_when(
36     is.na(num_sto_campaign) | num_sto_campaign == 0 ~ "Never used (product is on, but never used)",
37     num_sto_campaign <= 20 | is.na(sto_uplift) ~ "Testing",
38     days_since_last_sto_campaign > 15 ~ "Abandoned use",
39     sto_uplift > 0 ~ "Seeing value",
40     TRUE ~ "Using"
41   )] %>%
42   .[, .(customer_id, customer_name, lifecycle)] %>%
43   .[order(customer_name)]
44
```

Make it easy(er) to read!

```
1 customers <- getFreshListOfAllCustomers(exclude_test = FALSE)
2
3 sto_uplift <- sto_performance %>%
4   .[, versions := segment] %>%
5   .[, segment := NULL] %>%
6   calculateEffect("open_rate", by_cols = c("customer_id", "customer_name")) %>%
7   transformColsForSimplePlot("open_rate") %>%
8   .[, sto_uplift := round(treatment_uplift, 4)] %>%
9   .[, .(customer_id, sto_uplift)]
10
11 adoption_metrics_dt <- merge(customers, sto_uplift, by = "customer_id", all.x = TRUE)
12
13 lifecycle_for_relevant_customers <- adoption_metrics_dt %>%
14   filterInternalAccounts() %>%
15   calculateLifecycleForAdoptionMetrics_()
```

Quick recap

Structure of functions:

- name the function
- define input parameters / arguments
 - set default values if needed
- write down the logic
- provide a return value

How to write functions

```
name_of_the_function <- (argument_1, argument_2, ...) {  
  Body of the function  
  
  last line is the return value  
}
```

Your turn!

Write a function to calculate the average ...

Check prerequisites for more info on functions!