

10
10
23

essencemediacom
business science

Marketing Mix Modeling



UNIwersYTET WARSZAWSKI
Wydział Nauk Ekonomicznych



UNIwersYTET
WARSZAWSKI

Wykład nr 2
Intro do R





Today's agenda

- 01** Set-up R & basic tips
- 02** Praktyka
- 03** Useful tips & links

01



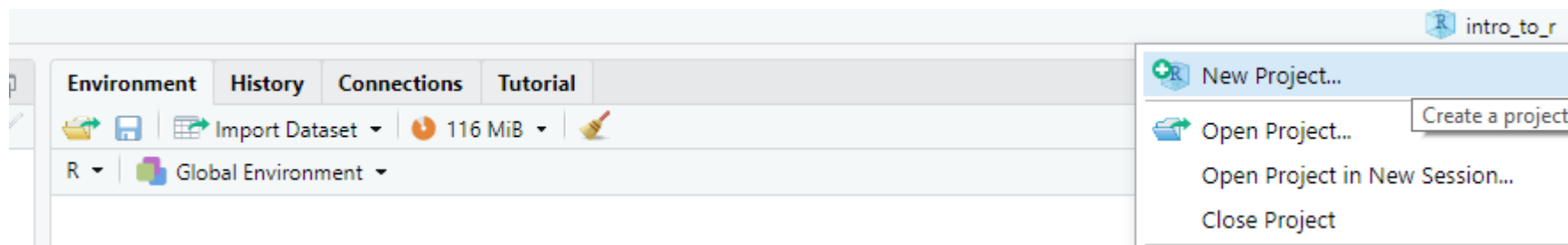
Section one
Set-up R and
basic tips

Set-up środowiska

1. Instalacja R i Rstudio

1. <https://cran.r-project.org/bin/windows/base/>
2. <https://posit.co/download/rstudio-desktop/>
3. Rtools (opcjonalnie)
<https://cran.r-project.org/bin/windows/Rtools/>

2. Tworzenie projektów



RStudio – przydatne ustawienia

1. Odstęp:

Tools -> Global Options -> Code -> Tab Width -> 4

2. Encoding:

File -> Reopen with encoding -> UTF-8 -> Set as default encoding

3. Zawijanie kodu:

Tools -> Global Options -> Code -> Soft-wrap R source files

4. Zmiana wielkości czcionki:

1. Tools -> Global Options... -> Appearance

2. Ctrl + + / Ctrl + -

5. Zmiana motywu

Tools -> Global Options -> Appearance -> Editor theme
(polecam wybrać jakiś czarny motyw, np. Idle Fingers)

RStudio panels

The screenshot displays the RStudio interface with four main panels highlighted by red boxes:

- Source:** The top-left panel shows the R script editor with the following code:

```
1 library(ggplot2)
2 mpg_plot <- ggplot(mpg, aes(x = displ, y = hwy)) +
3   geom_point(aes(colour = class))
4
5 mpg_plot
6
```
- Environment:** The top-right panel shows the Global Environment with a table of objects:

Name	Type	Len...	Size	Value
mpg_plot	gg	9	29.1...	List of 9
- Console:** The bottom-left panel shows the R console output:

```
> library(ggplot2)
> mpg_plot <- ggplot(mpg, aes(x = displ, y = hwy)) +
+   geom_point(aes(colour = class))
>
> mpg_plot
>
```
- Output:** The bottom-right panel shows a scatter plot of highway mileage (hwy) versus engine displacement (displ), colored by car class. The legend indicates the following classes: 2seater (red), compact (yellow), midsize (green), minivan (teal), pickup (cyan), subcompact (blue), and suv (magenta).

RStudio – obsługa

1. Podejrzenie historii kodu w zakładce History (Environment pane)
2. zapisywanie i wgrywanie utworzonej wcześniej bazy danych z poziomu Environment
3. Czyszczenie bazy miotłąką
4. odpalanie wcześniejszej linijki kodu w konsoli przez strzałkę w górę (Console pane)
5. “>” R jest gotowy do nowej komendy
6. “+” R czeka na zakończenie komendy

RStudio – skróty klawiszowe

1. **Ctrl + Shift + C** - comment/uncomment linijki
2. **F1** (gdy kursor stoi wewnątrz nazwy funkcji) - przekierowanie do Helpa
3. **F2** (gdy kursor stoi wewnątrz nazwy funkcji) - zakładka z kodem danej funkcji
4. **Ctrl + F** - znajdź / znajdź i zamień, można też wyszukać/wyszukać i zamienić w zaznaczeniu
5. **Ctrl + Enter** - wywołanie linijki kodu, w której znajduje się kursor
6. **Ctrl + Shift + Enter** - wywołanie całego skryptu
7. **Ctrl + 1** - zmiana pozycji kursora do okna skryptu
8. **Ctrl + 2** - zmiana pozycji kursora do okna konsoli
9. **Ctrl + Alt + Arrow up/down/LPM** - powielenie kursora w edytorze
10. **Ctrl + Shift + R** – nowy chunk kodu

Najważniejsze typy i struktury danych

Typ

dla pojedynczej wartości

Typ	Opis	Przykład
Logical	Wartości logiczne	TRUE, FALSE, T, F
Integer	Liczby całkowite	1, -5, 997, -2010, 3000L
Numeric	Liczby rzeczywiste (default typ dla liczb)	1, 0.5, -3.33, 0.2136
Complex	Liczby zespolone	1+0i, 1+4i
Character	Tekst, string	„A”, „MBS”, „Przyroda”

Struktura

Zbiorka dla wielu wartości

Struktura	Opis
Vector	Wektor danych, najprostszy 1-elementowy jest po prostu pojedynczą wartością, może składać się z wielu elementów, ale wszystkie elementy muszą być tego samego typu
List	Lista, podobnie jak wektor, ale umożliwia przechowywanie elementów różnych typów
Factor	Faktory to specjalne wektory do danych, zawierających kategorie, mogą zawierać tylko pre-definiowane wartości (kategorie)
Matrix	macierz to wektor składający się z wektorów
Data frame	Data frame to tabela danych, składa się z kolumnowych wektorów, z których każdy ma swoją nazwę. Bardziej elegancka forma macierzy
Tibble	Data frame z pakietu <i>tidyr</i> „na sterydach”. Więcej na następnych zajęciach ☺

Braki danych i inne „błędne” wartości

- **Wartości NA** (ang. Not Available) to braki danych, które występują w zbiorze danych, gdy nie ma informacji na temat wartości danego atrybutu dla pewnych obserwacji.
- **NA** najczęściej występują z powodu:
 - Braku danych (np. brak odpowiedzi u respondenta)
 - Błąd pomiarowy
 - Błąd obliczeń
- Każdy przypadek braku danych trzeba zbadać i podjąć stosowne działania (np. imputacja bądź usunięcie obserwacji)
- Przydatne funkcje:
 - **is.na()** ← sprawdzenie czy występują NA
 - **colSums(is.na(zbior.danych.df))** ← identyfikacja w której kolumnie znajdują się NA
 - Wiele funkcji ma parametr **na.rm**, który przyjmuje wartość logiczną TRUE lub FALSE. Sprawdź jak zmieni się poniższy wynik w zależności od tego parametru: **sum(c(1, 2, NA, 4), na.rm = F)**

Rodzaje braków danych

Symbol	Znaczenie	Opis
NA	Not Available	Brak danych
NaN	Not a Number	Na przykład wynik działania 0/0 <u>Każde NaN to NA, ale nie każde NA to NaN</u>
Inf / -Inf	Nieskonczoność	Na przykład wynik działania 1/0 lub -1/0
NULL	Null	Nic, najczęściej występuje gdy funkcja nie potrafi zwrócić wyniku (np. ma źle zdefiniowane argumenty)

Podstawowe operacje i funkcje

Podstawowe operacje na liczbach

Oznaczenie	Działanie
+	Dodawanie
-	Odejmowanie
*	Mnożenie
/	Dzielenie
^	Potęga
sqrt()	Pierwiastek
%%	Reszta z dzielenia
%/%	Część całkowita z dzielenia
log()	Logarytm (domyślnie naturalny)
exp()	Eksponenta (e^x)
abs()	Wartość bezwzględna
round()	Zaokrąglenie
sum()	Suma
mean()	Średnia
min() / max()	Minimum / maksimum

Oznaczenie	Działanie
paste()	Łączenie tekstów
substr()	Wyciąganie fragmentu tekstu
grepl()	Sprawdzenie czy jeden string zawiera się w drugim
grep()	Sprawdzenie w których elementach wektora zawiera się dany string
regexr()	sprawdzanie na której pozycji w danym stringu znajduje się szukany string, podaje pierwsze wystąpienie
gregexpr()	to samo co regexr tylko podaje wszystkie pozycje na których szukany string się znajduje
sub()	znajdź i zamień, dla pierwszej pozycji z przeszukiwanym stringu
gsub()	to samo co sub, ale zamienia wszystkie wystąpienia
tolower()	zamiana na małe litery
toupper()	zamiana na wielkie litery
nchar()	liczba znaków
strsplit()	rozdzielenie stringa na fragmenty wg podanego ogranicznika

Najważniejsze typy i struktury danych

Typ	Opis
<code>colSums()</code> , <code>rowSums()</code>	Suma kolumn / wierszy
<code>colMeans()</code> , <code>rowMeans()</code>	Średnia kolumn / wierszy
<code>sort()</code> , <code>order()</code>	Sortowanie
<code>unique()</code>	Unikalne wartości
<code>t()</code>	Transpozycja
<code>cor()</code>	Korelacja
<code>ifelse()</code>	Wykonaj instrukcję nr1 jeżeli jest spełniony warunek, w przeciwnym przypadku wykonaj instrukcję nr2 (skrócony if)
<code>crossprod()</code>	Suma iloczynów

02



Section two
Praktyka

03



Section three

Useful tips & links

Zasady pisania kodu

1. Czytelność

- Opisowe nazwy zmiennych, data frame, wektorów itd.
- Funkcje opisane na konkretnym poziomie

2. Przejrzystość

- Kod, który jest zakomentowany powinniśmy usuwać
- Stosowanie spacji, chunków, pipeline'ów

3. Dokumentacja

- Poszczególne kroki analizy powinny być wyjaśnione (wystarczy nawet krótki komentarz)

Zasady pisania kodu

4. Ustrukturyzowany

- Fragmenty kodu w kolejności ich wykonywania
- Podział na chunki

5. Zasada DRY

- **Don't repeat yourself**
- Tworzenie funkcji lub pętli zamiast kopiowania tego samego kodu kilkakrotnie

6. Projekty

- Dzięki projektom nie trzeba ustawiać *working directory* dla poszczególnych skryptów

Cheatsheety

Krótkie, przedstawione w przejrzysty sposób (w formie "plakatuwej") opisy poszczególnych bibliotek z przedstawionymi najważniejszymi funkcjami, uwagami i przykładami ich użycia. Ich ideą jest przedstawienie jak najwięcej w jak najbardziej kompaktowej formie.

Najłatwiej znaleźć je w Google hasłem ***nazwa_pakietu cheatsheet***.

Oficjalna lista cheatsheetów:

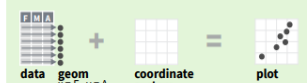
<https://posit.co/resources/cheatsheets/>

Data visualization with ggplot2 : : CHEAT SHEET

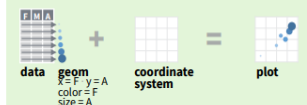


Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),  
    stat = <STAT>, position = <POSITION>) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <SCALE_FUNCTION> +  
  <THEME_FUNCTION>
```

ggplot(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

last_plot() Returns the last plot.

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5" x 5" file named "plot.png" in working directory. Matches file type to file extension.

Aes

Common aesthetic values.

color and fill - string ("red", "#RRGGBB")

linetype - integer or string (0 = "blank", 1 = "solid", 2 = "dashed", 3 = "dotted", 4 = "dotteddash", 5 = "longdash", 6 = "shortdash")

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemployment))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank() and **a + expand_limits()**
Ensure limits include values across all plots.
b + geom_curve()(aes(yend = lat + 1, xend = long + 1), curvature = 1) - x, yend, y, yend, alpha, angle, color, curvature, linetype, size
a + geom_path()(lineend = "butt", linejoin = "round", linemitre = 1) - x, y, alpha, color, group, linetype, size
a + geom_polygon()(aes(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, linetype, size
b + geom_rect()(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
a + geom_ribbon()(aes(ymin = unemployment - 900, ymax = unemployment + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size
b + geom_abline()(aes(intercept = 0, slope = 1))
b + geom_hline()(aes(yintercept = lat))
b + geom_vline()(aes(xintercept = long))
b + geom_segment()(aes(yend = lat + 1, xend = long + 1))
b + geom_spoke()(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg, aes(hwy))
c + geom_area()(stat = "bin") - x, y, alpha, color, fill, linetype, size
c + geom_density()(kernel = "gaussian") - x, y, alpha, color, fill, group, linetype, size, weight
c + geom_dotplot() - x, y, alpha, color, fill
c + geom_freqpoly() - x, y, alpha, color, group, linetype, size
c + geom_histogram()(binwidth = 5) - x, y, alpha, color, fill, linetype, size, weight

TWO VARIABLES both continuous

e <- ggplot(mpg, aes(cty, hwy))
e + geom_label()(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
e + geom_point() - x, y, alpha, color, fill, shape, size, stroke
e + geom_quantile() - x, y, alpha, color, group, linetype, size, weight
e + geom_rug()(sides = "bl") - x, y, alpha, color, linetype, size
e + geom_smooth()(method = lm) - x, y, alpha, color, fill, group, linetype, size, weight
e + geom_text()(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

f <- ggplot(mpg, aes(class, hwy))
f + geom_col() - x, y, alpha, color, fill, group, linetype, size
f + geom_boxplot() - x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
f + geom_dotplot()(binaxis = "y", stackdir = "center") - x, y, alpha, color, fill, group
f + geom_violin()(scale = "area") - x, y, alpha, color, fill, group, linetype, size, weight

both discrete

g <- ggplot(diamonds, aes(cut, color))
g + geom_count() - x, y, alpha, color, fill, shape, size, stroke
e + geom_jitter()(height = 2, width = 2) - x, y, alpha, color, fill, shape, size

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))
h + geom_bin2d()(binwidth = c(0.25, 500)) - x, y, alpha, color, fill, linetype, size, weight
h + geom_density_2d() - x, y, alpha, color, group, linetype, size
h + geom_hex() - x, y, alpha, color, fill, size

continuous function

i <- ggplot(economics, aes(date, unemployment))
i + geom_area() - x, y, alpha, color, fill, linetype, size
i + geom_line() - x, y, alpha, color, group, linetype, size
i + geom_step()(direction = "hv") - x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
j + geom_crossbar()(fatten = 2) - x, y, ymax, ymin, alpha, color, fill, group, linetype, size
j + geom_errorbar() - x, ymax, ymin, alpha, color, group, linetype, size, width
Also **geom_errorbarh()**.
j + geom_linerange() - x, ymin, ymax, alpha, color, group, linetype, size
j + geom_pointrange() - x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
k + geom_map()(aes(map_id = state), map = map) + **expand_limits**(x = map\$long, y = map\$lat)
map_id, alpha, color, fill, linetype, size

Przydatne linki

- **R for Data Science**, H. Wickham, G. Grolemund
<https://r4ds.had.co.nz/>
- **Różnego rodzaju oficjalne manuały, od podstaw do zaawansowanych rzeczy**
<https://cran.r-project.org/manuals.html>
- **Cheatsheety**
<https://web.sgh.waw.pl/~jmuck/R/ListaNkomendR.pdf>
- **Galeria wykresów w R razem z kodami**
<https://r-graph-gallery.com/index.html>
- **Jak wygląda rzeczywista analiza danych w R – projekt tidytuesday**
<https://github.com/rfordatascience/tidytuesday>
<https://www.youtube.com/@safe4democracy/videos> (przykładowy workflow)

Gdzie szukać pomocy?

- Google
- Stack Overflow
- Help do funkcji / pakietu
- Cheatsheety
- ChatGPT / inne chatboty
- Tutoriale na YT



**Nauka pisania kodu polega na ...
pisaniu jak największej ilości
różnorodnych kodów.**

**Jeśli czegoś nie wiesz lub nie rozumiesz to korzystaj
z pomocy internetu. Na 99% ktoś miał taki sam lub
podobny problem i rozwiązanie już gdzieś istnieje.**



essencemediacom
business science


Jarek Dejneka, Managing Partner
jaroslaw.dejneka@essencemediacom.com

Bartek Kowalski, Business Science Director
bartosz.kowalski@essencemediacom.com

 mbs@mediacom.com

 [@MBSWarsaw](https://www.instagram.com/MBSWarsaw)

 [@MBSWarsaw](https://twitter.com/MBSWarsaw)

 business-science.pl

***„An investment in knowledge
always pays the best interest”***

Benjamin Franklin

