

Marketing Mix Modeling

Wprowadzenie do tidyverse – część I

17 | 10 | 2023



UNIwersYTET WARSZAWSKI
Wydział Nauk Ekonomicznych



UNIwersYTET
WARSZAWSKI

essencemedia.com
business science

Agenda

1. Czym jest tidyverse?
2. Wczytywanie danych
3. Parsowanie danych
4. Grupowanie i sumowanie danych



01

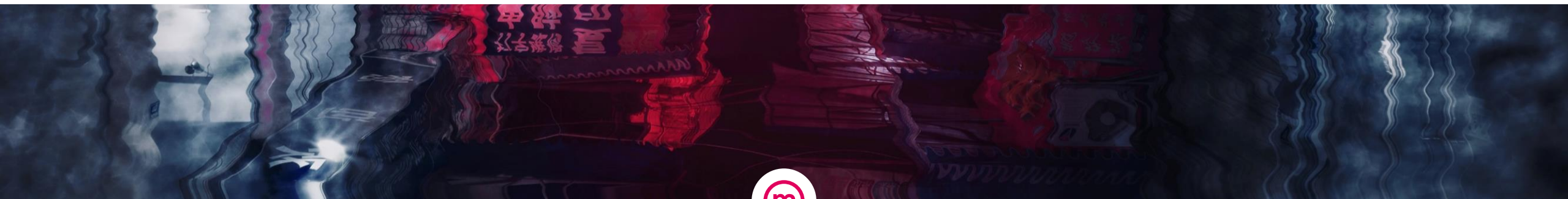
Czym jest tidyverse?



Tidyverse - intro

Czym jest tidyverse:

- Jedno z najbardziej znanych zbiorów bibliotek w R, napisana przez [Hadleya Wickhama](#) wraz z zespołem
- Tidyverse zawiera biblioteki, które umożliwiają: import, uporządkowanie, przetwarzanie, analizę i wizualizację danych.
- Cechą charakterystyczną tidyversa, jest stosowanie iteracyjnego sposobu pracy z wykorzystaniem danych poprzez stosowanie pipeline'ów (%>%)
- <https://www.tidyverse.org/>



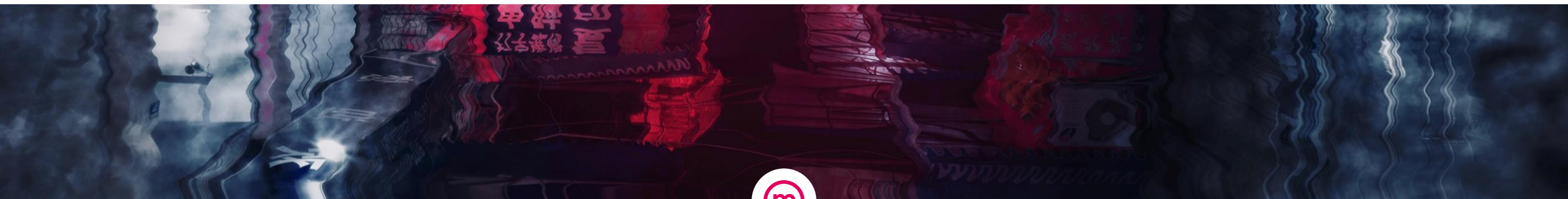
「Czym są uporządkowane dane (tidy data)

Czym są uporządkowane dane (tidy data):

- Uzyskanie danych w formacie tidy wymaga wcześniejszej pracy, ale jest o bardzo opłacalna praca w dłuższej perspektywie czasu,
- Uporządkowanie zarówno danych jak i narzędzi zmniejsza szanse na popełnienie błędu w trakcie przetwarzania danych oraz pozwoli na spędzenie większej ilości czasu na pytaniach analitycznych.

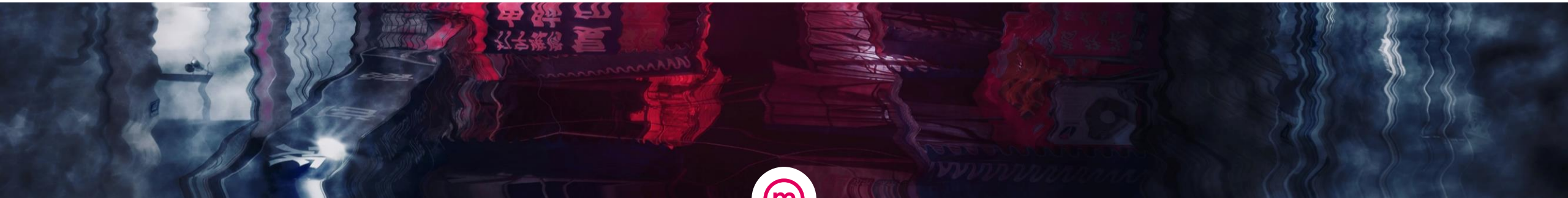
Po co dbać o porządek w swoich danych? Istnieją dwie główne zalety:

1. Istnieje ogólna zaleta wybrania jednego spójnego sposobu przechowywania danych. Jeśli masz spójną strukturę danych, łatwiej jest nauczyć się narzędzi, które z nią współpracują,
2. Umieszczanie zmiennych w kolumnach ma szczególną zaletę - większość wbudowanych funkcji języka R działa z wektorami wartości, a operacje zorientowane na wektory są najbardziej wydajne w języku R. Dzięki temu przekształcanie i uporządkowanych danych jest szczególnie proste.



「The tidy tools manifesto

<https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html>



Tidy data – 3 warunki do spełnienia

Każda zmienna musi mieć własną kolumnę

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	128042583

variables

Każda obserwacja musi mieć swój własny wiersz

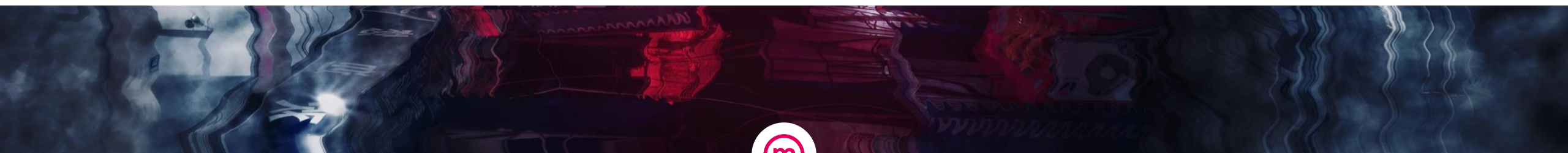
country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	128042583

observations

Każda wartość musi mieć własną komórkę

country	year	cases	population
Afghanistan	99	745	19987071
Afghanistan	00	2666	20595360
Brazil	99	37737	172006362
Brazil	00	80488	174504898
China	99	212258	1272915272
China	00	213766	128042583

values



Przykłady tidy & untidy data

table1

```
#> # A tibble: 6 x 4
#>   country    year cases population
#>   <chr>    <int> <int>      <int>
#> 1 Afghanistan 1999     745  19987071
#> 2 Afghanistan 2000    2666  20595360
#> 3 Brazil      1999   37737  172006362
#> 4 Brazil      2000   80488  174504898
#> 5 China       1999  212258 1272915272
#> 6 China       2000  213766 1280428583
```

table2

```
#> # A tibble: 12 x 4
#>   country    year type      count
#>   <chr>    <int> <chr>    <int>
#> 1 Afghanistan 1999 cases      745
#> 2 Afghanistan 1999 population 19987071
#> 3 Afghanistan 2000 cases      2666
#> 4 Afghanistan 2000 population 20595360
#> 5 Brazil      1999 cases      37737
#> 6 Brazil      1999 population 172006362
#> # ... with 6 more rows
```

table3

```
#> # A tibble: 6 x 3
#>   country    year rate
#>   * <chr>    <int> <chr>
#> 1 Afghanistan 1999 745/19987071
#> 2 Afghanistan 2000 2666/20595360
#> 3 Brazil      1999 37737/172006362
#> 4 Brazil      2000 80488/174504898
#> 5 China       1999 212258/1272915272
#> 6 China       2000 213766/1280428583
```

Spread across two tibbles

table4a # cases

```
#> # A tibble: 3 x 3
#>   country    `1999` `2000`
#>   * <chr>    <int> <int>
#> 1 Afghanistan     745     2666
#> 2 Brazil          37737  80488
#> 3 China          212258 213766
```

table4b # population

```
#> # A tibble: 3 x 3
#>   country    `1999`    `2000`
#>   * <chr>    <int>    <int>
#> 1 Afghanistan 19987071  20595360
#> 2 Brazil      172006362 174504898
#> 3 China      1272915272 1280428583
```



Biblioteki w ramach tidyverse

Import
danych



Uporządkowanie
danych



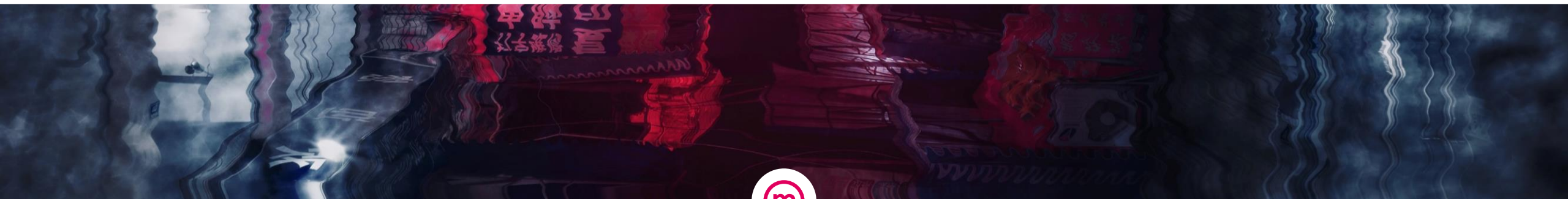
Przetwarzanie
danych



Programowanie



Wizualizacja



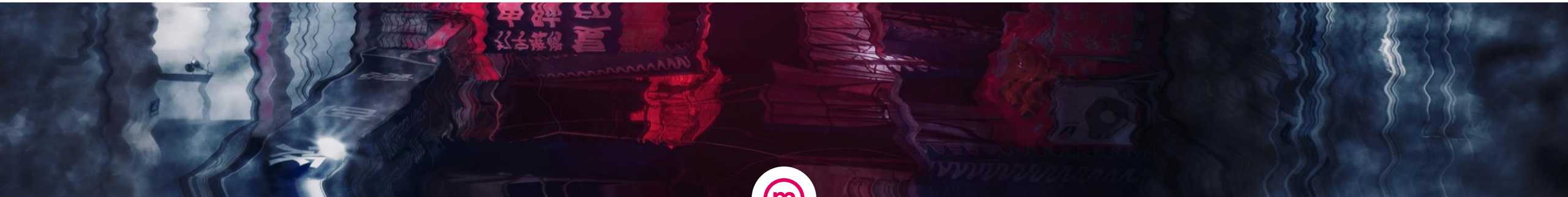
Pipelines

Pipelines (%>%) :

- Pipe'y („%>%“) są automatycznie łądowane przez bibliotekę tidyverse i służą do sekwencyjnej manipulacji zbiorem danych.
- Służą do wykonywania wielu operacji w jednym kroku.
- dane (argument) przekazywane są do kolejnej funkcji po zastosowaniu „%>%“

Pipelines – przykład:

```
data.tidy.df %>%  
  select(Date, Brand, TRP, Cost) %>%  
  filter(Brand == "Brand2") %>%  
  mutate(Cost_EUR = Cost * 4.5,  
         Cost_x_TRP = Cost * TRP,  
         TRP_new = TRP * 100) %>%  
  rename(Cost_EUR_new = Cost_EUR) %>%  
  arrange(desc(Date))|
```



02

Wczytywanie danych



Import danych - **readr**

Import
danych



Uporządkowanie
danych



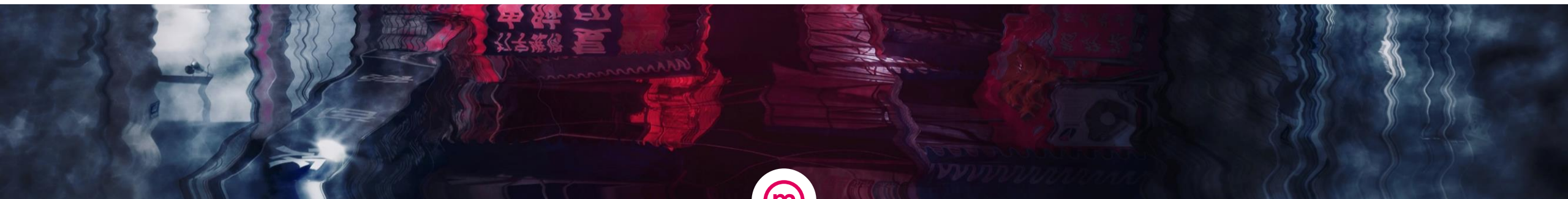
Przetwarzanie
danych



Programowanie



Wizualizacja



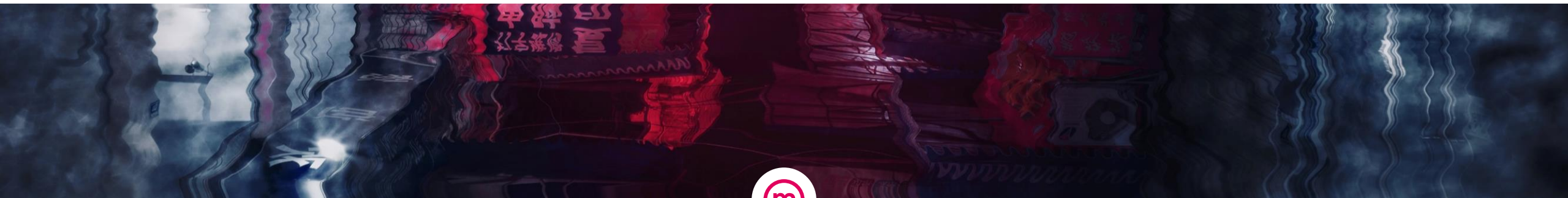
Wczytanie danych do środowiska (pakiety **readr** i **readxl**)

readr:

- Pakiet readr wykorzystywany jest to wczytania i zapisania niemal wszystkich formatów plików.
- Składnia funkcji: `read_*`, `write_*`, gdzie `*` zastępujemy formatem pliku, np. `csv`, `tsv`
- Importowane pliki mają strukturę tibble – o tym w dalszej części wykładu
- Przykład:
`read_csv2(file, ...)`

readxl:

- Dodatkowy pakiet ze środowiska tidyverse,
- Funkcje w pakiecie umożliwiają import danych do środowiska R zarówno z formatu `.xls` jak i `.xlsx`.
- Najbardziej podstawowa funkcja do importu danych to **`read_excel()`**, która rozpoznaje format importowanego pliku po rozszerzeniu.
- Przykład:
`read_excel(path, sheet = 1, range = NULL, n_max = Inf, skip = 0)`



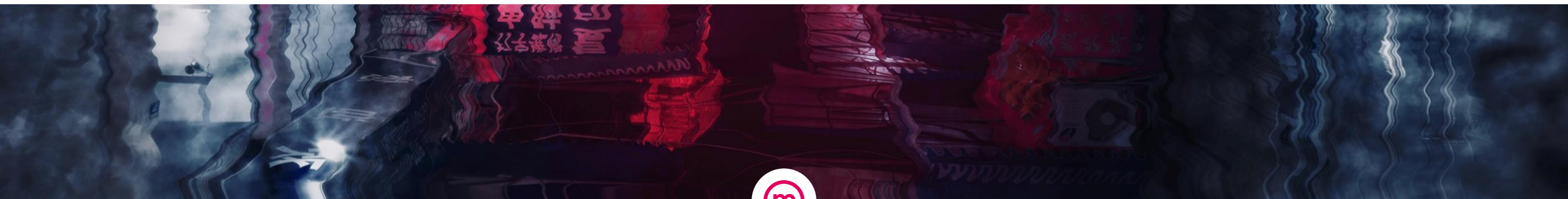
Tibble vs dataframe - różnice

Tibble vs data frame

- Tibble (klasa o nazwie `tbl_df`) jest nowoczesnym, bardziej użytecznym type `data.frame`,
- Tibble robią mniej niż klasyczny `data.frame` – nie zmieniają nazw, typów zmiennych, nie dokonując częściowego dopasowania oraz bardziej narzekają np. gdy zmienna nie istnieje. Skutkuje to wcześniejszym zmierzeniem się z problemami jakie istnieją w plikach co najczęściej prowadzi do czytelniejszego kodu,
- Tibble wyświetlają tylko 10 pierwszych wierszy oraz wszystkie kolumny, które mieszczą się na ekranie – ułatwia to pracę z dużymi danymi. Dodatkowo, wyświetlony jest typ kolumny pod jego nazwą, stosowane są czcionki kolory do podświetlania.

Największą różnicą pomiędzy tibble a `data.frame` – tibble wykonują znacznie mniej pracy:

- nigdy nie zmienia typu danych wejściowych (np. nigdy nie konwertuje stringów na factor)
- nigdy nie zmienia nazw zmiennych (np. wstawia `.` zamiast spacji między słowami)
- nigdy nie tworzy nazw wierszy (`row.names()`) - Celem uporządkowanych danych jest przechowywanie zmiennych w spójny sposób



Przykład tibble vs dataframe

```
> example.data3.df
# A tibble: 1,906 x 5
  Date           Brand `Ratecard` Duration` Metric      Value
  <dtm>          <chr>    <dbl>    <chr>    <dbl>
1 2017-01-11 00:00:00 Brand1      30 Cost    2655323.
2 2017-01-11 00:00:00 Brand1      30 TRP      14.2
3 2017-05-15 00:00:00 Brand2      30 Cost   111461286.
4 2017-05-15 00:00:00 Brand2      30 TRP      2182.
5 2017-05-16 00:00:00 Brand2      30 Cost   212519311.
6 2017-05-16 00:00:00 Brand2      30 TRP      3942.
7 2017-05-17 00:00:00 Brand2      30 Cost   146992783.
8 2017-05-17 00:00:00 Brand2      30 TRP      2972.
9 2017-05-18 00:00:00 Brand2      30 Cost   178000107.
10 2017-05-18 00:00:00 Brand2      30 TRP      1874.
# ... with 1.896 more rows
```

```
> example.data3.data.frame.df
      Date Brand Ratecard Duration Metric      Value
1 2017-01-11 Brand1      30 Cost    2655323.35
2 2017-01-11 Brand1      30 TRP      14.22
3 2017-05-15 Brand2      30 Cost  111461285.81
4 2017-05-15 Brand2      30 TRP    2181.98
5 2017-05-16 Brand2      30 Cost  212519310.62
6 2017-05-16 Brand2      30 TRP    3942.10
7 2017-05-17 Brand2      30 Cost  146992782.74
8 2017-05-17 Brand2      30 TRP    2971.98
9 2017-05-18 Brand2      30 Cost  178000107.05
10 2017-05-18 Brand2      30 TRP    1873.88
11 2017-05-19 Brand2      30 Cost  170921840.40
12 2017-05-19 Brand2      30 TRP    3600.82
13 2017-05-20 Brand2      30 Cost  137344848.34
14 2017-05-20 Brand2      30 TRP    3455.46
15 2017-05-21 Brand2      30 Cost  132937478.78
16 2017-05-21 Brand2      30 TRP    4266.88
```



03

Parowanie danych

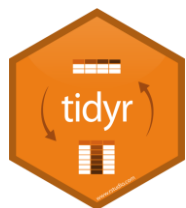


Biblioteki w ramach tidyverse

Import
danych



Uporządkowanie
danych



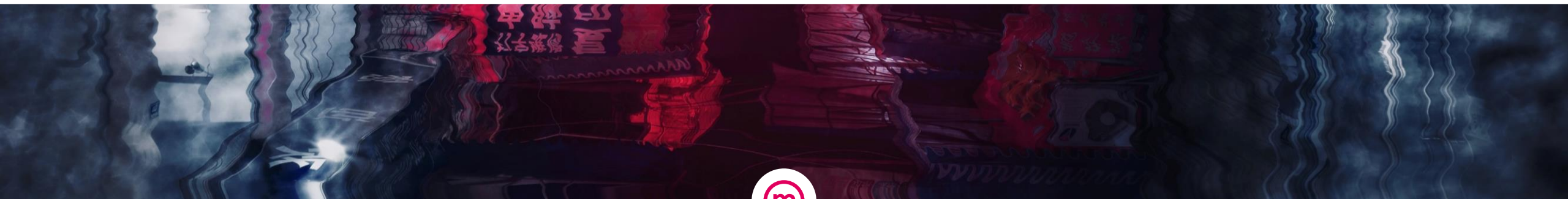
Przetwarzanie
danych



Programowanie



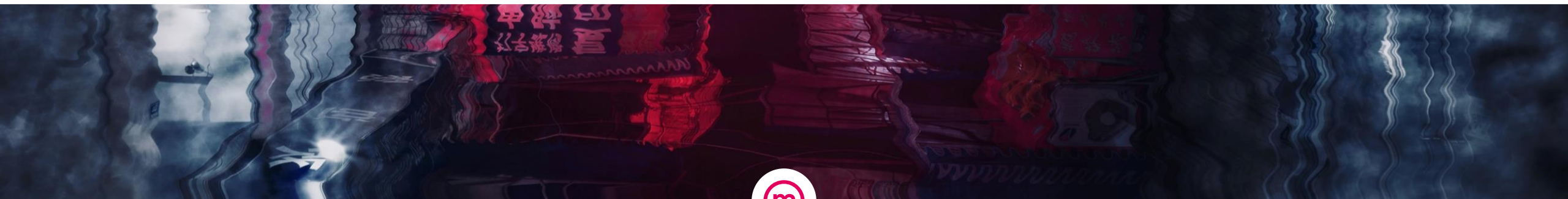
Wizualizacja



Tworzenie uporządkowanego zbioru danych

Funkcje w ramach biblioteki **tidyr**:

Funkcje	Opis
pivot_wider()	Funkcja „rozszerzają” wiele kolumn ze zbioru danych i konwertuje je na pary klucz-wartość — POZIOMO
pivot_longer()	Funkcja zajmuje dwie kolumny i „wydłuża” je w kilka kolumn - PIONOWO
separate()	Oddziela / dzieli pojedynczą kolumnę na wiele kolumn
unite()	W przeciwieństwie do separate() - łączy dwie lub więcej kolumn w jedną



04

Grupowanie i sumowanie danych



Biblioteki w ramach tidyverse

Import
danych



Uporządkowanie
danych



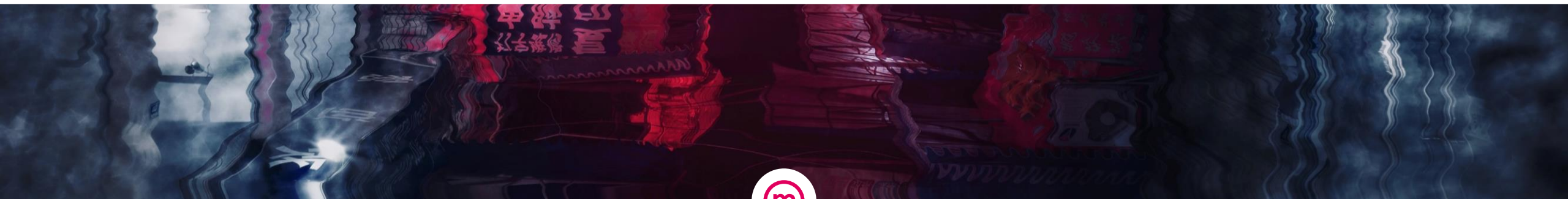
Przetwarzanie
danych



Programowanie



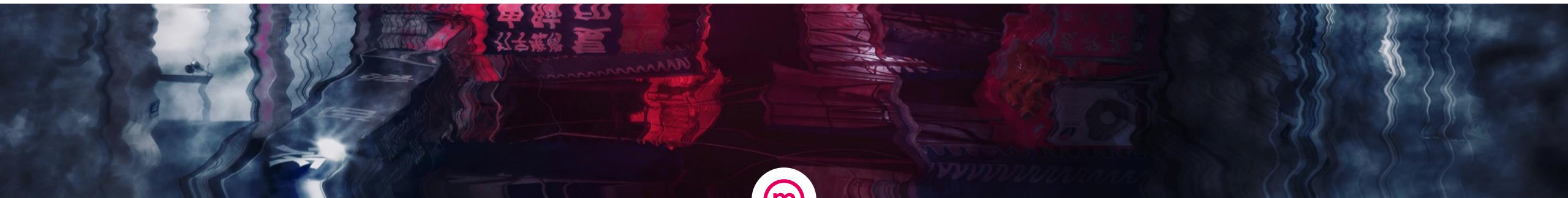
Wizualizacja



Przetwarzanie danych z wykorzystaniem biblioteki **dplyr**

Uporządkowany zbiór czasowników:

Verbs:	Description
<code>mutate()</code>	dodaje nową zmienną przy pomocy dodatkowych funkcji
<code>select()</code>	wybiera zmienną na podstawie jej nazwy
<code>filter()</code>	filtruje zmienne na podstawie ich wartości
<code>arrange()</code>	zmienia kolejność wierszy
<code>group_by()</code>	grupuje na podstawie zadanych zmiennych
<code>summarise()</code>	na podstawie <code>group_by</code> tworzy agregaty zmiennych
<code>rename()</code>	zmienia nazwy kolumn



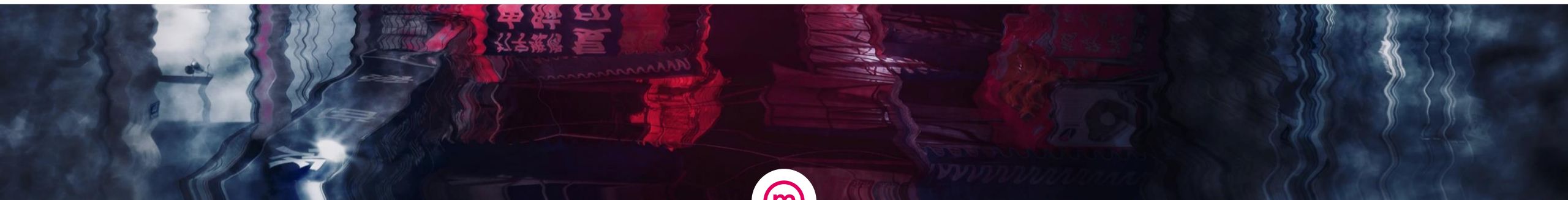
Przykład wykorzystania wszystkich czasowników

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa

```
iris.new.df <- iris %>%  
  select(Species , Sepal.Length, Sepal.Width ) %>%  
  mutate(sum = Sepal.Length + Sepal.Width) %>%  
  filter(Sepal.Length >= 4.5) %>%  
  arrange(Species) %>%  
  rename(Sepal.sum = sum) %>%  
  group_by(Species) %>%  
  summarise(Sepal.Length.sum = sum(Sepal.Length),  
            Sepal.Width.mean = mean(Sepal.Width),  
            Sepal.sum.max = max(Sepal.sum))
```



	Species	Sepal.Length.sum	Sepal.Width.mean	Sepal.sum.max
1	setosa	232.8	3.463043	10.1
2	versicolor	296.8	2.770000	10.2
3	virginica	329.4	2.974000	11.7





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

