Zarządzanie danymi z wykorzystaniem pakietu dplyr

Kacper Jagiełło

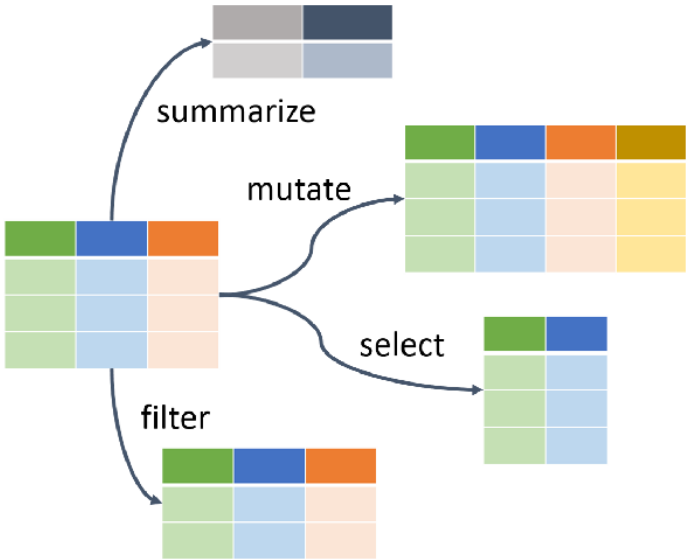
Zakład Prewencji i Dydaktyki
Gdański Uniwersytet Medyczny

9 lutego 2021

Spis treści

Wprowadzenie

Zarządzanie danymi to ważny proces poprzedzający analizę statystyczną. Umiejętne wykorzystanie dostępnych pakietów i funkcji w R pozwala na szybkie i efektywne działania w bazie danych. Takimi operacjami są między innymi: wybieranie i tworzenie zmiennych, filtrowanie i sortowanie obserwacji, tworzenie zestawień czy łączenie baz danych.



Pakiet *dplyr*

Doskonałym narzędziem do wykonania wspomnianych wcześniej działań jest stworzony przez Hadleya Wickhama pakiet *dplyr*. Oferuje on szereg funkcji, dzięki którym zarządzenie danymi staje się prostsze i bardziej przyjazne, ponieważ z wielu z nich możemy korzystać intuicyjnie.

Baza danych

Number	Name	Nation	Pos	Club	Age	Born	MP	Starts	Min
1	Issah Abbas	GHA	WB	Mainz	22	1998	2	0	18
2	David Abraham	ARG	CB	Eintracht	34	1986	14	14	1222
3	Amir Abrashi	ALB	DM	Freiburg	30	1990	5	0	60
4	Ragnar Ache	GER	FW	Eintracht	22	1998	2	0	36
5	Tyler Adams	USA	DM	Leipzig	21	1999	15	11	1051
6	Sargis Adamyan	ARM	W	Hoffenheim	27	1993	9	0	154
7	Felix Agu	GER	WB	Werder	21	1999	7	4	365
8	Manuel Akanji	SUI	CB	Dortmund	25	1995	17	16	1475
9	Kevin Akpoguma	NGA	CB	Hoffenheim	25	1995	13	11	942
10	David Alaba	AUT	CB	Bayern	28	1992	18	17	1542
11	Lucas Alario	ARG	FW	Leverkusen	28	1992	16	8	826
12	Omar Alderete	PAR	CB	Hertha	24	1996	10	10	884

Najważniejsze funkcje

Wybieranie zmiennych

Aby stworzyć podzbiór bazy danych z wybranych przez nas kolumn według ich nazw, korzystamy z funkcji *select*.

Przykładowy kod

```
data <- data %>% select(Number, Name, Club, Note)
```

Number	Name	Club	Note
1	Issah Abbas	Mainz	NA
2	David Abraham	Eintracht	3.32
3	Amir Abrashi	Freiburg	NA
4	Ragnar Ache	Eintracht	NA
5	Tyler Adams	Leipzig	3.50
6	Sargis Adamyan	Hoffenheim	NA
7	Felix Agu	Werder	NA
8	Manuel Akanji	Dortmund	3.50
9	Kevin Akpoguma	Hoffenheim	3.60
10	David Alaba	Bayern	3.38
11	Lucas Alario	Leverkusen	NA
12	Omar Alderete	Hertha	3.85

Najważniejsze funkcje

Filtrowanie obserwacji

W celu wyodrębnienia obserwacji, które spełniają wybrane przez nas kryteria możemy użyć funkcji *filter*.

Przykładowy kod

```
data <- data %>% filter(!is.na(Note))
```

Number	Name	Club	Note
2	David Abraham	Eintracht	3.32
5	Tyler Adams	Leipzig	3.50
8	Manuel Akanji	Dortmund	3.50
9	Kevin Akpoguma	Hoffenheim	3.60
10	David Alaba	Bayern	3.38
12	Omar Alderete	Hertha	3.85
13	Nadiem Amiri	Leverkusen	3.57
15	Robert Andrich	Union	3.09
16	Angelino	Leipzig	2.83
17	Waldemar Anton	Stuttgart	3.43
20	Maximilian Arnold	Wolfsburg	3.06
23	Ludwig Augustinsson	Werder	3.27

Najważniejsze funkcje

Tworzenie nowych zmiennych

Do tworzenia nowych zmiennych w bazie danych służy funkcja *mutate*.

Przykładowy kod

```
data <- data %>% mutate(NotePL = 7 - Note)
```

Number	Name	Club	Note	NotePL
2	David Abraham	Eintracht	3.32	3.68
5	Tyler Adams	Leipzig	3.50	3.50
8	Manuel Akanji	Dortmund	3.50	3.50
9	Kevin Akpoguma	Hoffenheim	3.60	3.40
10	David Alaba	Bayern	3.38	3.62
12	Omar Alderete	Hertha	3.85	3.15
13	Nadiem Amiri	Leverkusen	3.57	3.43
15	Robert Andrich	Union	3.09	3.91
16	Angelino	Leipzig	2.83	4.17
17	Waldemar Anton	Stuttgart	3.43	3.57
20	Maximilian Arnold	Wolfsburg	3.06	3.94
23	Ludwig Augustinsson	Werder	3.27	3.73

Najważniejsze funkcje

Grupowanie i statystyki

W celu grupowania i tworzenia zmiennych w ramach grup używamy funkcji *group_by*, którą możemy połączyć z poleceniem *summarise*, pozwalającą na wyliczanie różnych statystyk jak na przykład średnia, mediana czy odchylenie standardowe.

Przykładowy kod

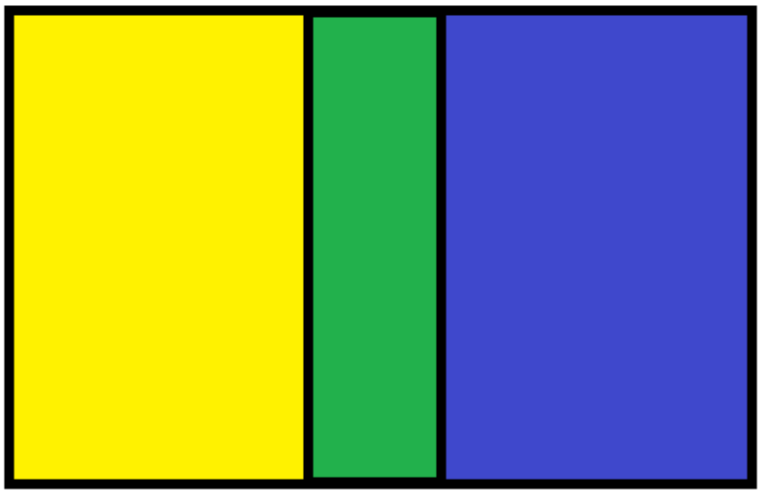
```
mNote <- data %>%
  group_by(Club) %>%
  summarise(mean_Note = round(mean(NotePL),2))
```

Najważniejsze funkcje

	Club	mean_Note
1	Arminia	3.22
2	Augsburg	3.27
3	Bayern	3.86
4	Dortmund	3.62
5	Eintracht	3.80
6	Freiburg	3.61
7	Hertha	3.30
8	Hoffenheim	3.73
9	Koeln	3.12
10	Leipzig	3.83
11	Leverkusen	3.74
12	Mainz	3.18

Łączenie baz danych

Pakiet *dplyr* oferuje również funkcje pozwalające łączyć ze sobą bazy danych według klucza.



```
Inner_join: ZIELONY

Left_join: ŻÓŁTY+ZIELONY

Right_join: NIEBIESKI+ZIELONY

Full_join: ŻÓŁTY+ZIELONY+NIEBIESKI

Semi_join: ZIELONY-NIEBIESKI

Anti_join: ŻÓŁTY-ZIELONY
```

Baza danych nr 2

Club	TabPos
Bayern	1
Leipzig	2
Wolfsburg	3
Eintracht	4
Leverkusen	5
Dortmund	6
Moenchengladbach	7
Freiburg	8
Union	9
Stuttgart	10
Werder	11
Hoffenheim	12

Łączenie baz danych

Przykładowy kod

```
mNote <- left_join(mNote, table, by = "Club")
```

	Club	mean_Note	TabPos
1	Arminia	3.22	16
2	Augsburg	3.27	13
3	Bayern	3.86	1
4	Dortmund	3.62	6
5	Eintracht	3.80	4
6	Freiburg	3.61	8
7	Hertha	3.30	15
8	Hoffenheim	3.73	12
9	Koeln	3.12	14
10	Leipzig	3.83	2
11	Leverkusen	3.74	5
12	Mainz	3.18	17

Najważniejsze funkcje

Sortowanie obserwacji

Obserwacje według zmiennych w bazie danych możemy posortować używając polecenia *arrange*.

Przykładowy kod

```
mNote <- mNote %>% arrange(desc(mean_Note))
```

	Club	mean_Note	TabPos
1	Bayern	3.86	1
2	Leipzig	3.83	2
3	Eintracht	3.80	4
4	Moenchengladbach	3.78	7
5	Stuttgart	3.78	10
6	Wolfsburg	3.76	3
7	Leverkusen	3.74	5
8	Hoffenheim	3.73	12
9	Union	3.69	9
10	Dortmund	3.62	6
11	Freiburg	3.61	8
12	Werder	3.41	11

Najważniejsze funkcje

Przykładowy kod

```
mNote <- mNote %>% mutate(NotePos = c(1:18))

mNote <- mNote %>%
  mutate(diff = NotePos - TabPos)
```

Club	mean_Note	TabPos	NotePos	diff
Bayern	3.86	1	1	0
Leipzig	3.83	2	2	0
Eintracht	3.80	4	3	-1
Moenchengladbach	3.78	7	4	-3
Stuttgart	3.78	10	5	-5
Wolfsburg	3.76	3	6	3
Leverkusen	3.74	5	7	2
Hoffenheim	3.73	12	8	-4
Union	3.69	9	9	0
Dortmund	3.62	6	10	4
Freiburg	3.61	8	11	3
Werder	3.41	11	12	1

Najważniejsze funkcje

Przypisanie kategorii

Dzięki funkcji *case_when* możemy kategoryzować obserwacje według ustalonych przez nas warunków.

Przykładowy kod

```
mNote <- mNote %>%
  mutate(
    Ocena = case_when(
      diff > 0  ~ "niedoceniani",
      diff < 0      ~ "przeceniani",
      TRUE                      ~ "bez zmian"
    )
  )
```

Club	mean_Note	TabPos	NotePos	diff	Ocena
Bayern	3.86	1	1	0	bez zmian
Leipzig	3.83	2	2	0	bez zmian
Eintracht	3.80	4	3	-1	przeceniani
Moenchengladbach	3.78	7	4	-3	przeceniani
Stuttgart	3.78	10	5	-5	przeceniani
Wolfsburg	3.76	3	6	3	niedoceniani
Leverkusen	3.74	5	7	2	niedoceniani
Hoffenheim	3.73	12	8	-4	przeceniani
Union	3.69	9	9	0	bez zmian
Dortmund	3.62	6	10	4	niedoceniani
Freiburg	3.61	8	11	3	niedoceniani
Werder	3.41	11	12	1	niedoceniani

Najważniejsze funkcje

Zmiana nazwy zmiennej

Operację zmiany nazwy zmiennej wykonujemy za pomocą funkcji *rename*.

Przykładowy kod

```
mNote <- mNote %>% rename(Klub = Club,
 ‘Średnia nota‘ = mean_Note,
‘Pozycja w tabeli‘ = TabPos,
‘Pozycja według not‘ = NotePos,
‘Różnica pozycji‘ = diff)
```

Klub	Średnia nota	Pozycja w tabeli	Pozycja według not	Różnica pozycji	Ocena
Bayern	3.86	1	1	0	bez zmian
Leipzig	3.83	2	2	0	bez zmian
Eintracht	3.80	4	3	-1	przeceniani
Moenchengladbach	3.78	7	4	-3	przeceniani
Stuttgart	3.78	10	5	-5	przeceniani
Wolfsburg	3.76	3	6	3	niedoceniani
Leverkusen	3.74	5	7	2	niedoceniani
Hoffenheim	3.73	12	8	-4	przeceniani
Union	3.69	9	9	0	bez zmian
Dortmund	3.62	6	10	4	niedoceniani
Freiburg	3.61	8	11	3	niedoceniani
Werder	3.41	11	12	1	niedoceniani

Najważniejsze funkcje

Generowanie tabeli

Z użyciem funkcji *flextable* tworzymy tabelę, którą możemy umieścić w raporcie.

Przykładowy kod

```
mNote2 %>%
  flextable() %>%
  autofit() %>%
  flextable::set_caption("Tabela 1. Średnie
  noty zawodników w poszczególnych klubach.")
```

Tabela 1. Średnie noty zawodników w poszczególnych klubach.

Klub	Średnia nota	Pozycja w tabeli	Pozycja według not	Różnica pozycji	Ocena
Bayern	3.86	1	1	0	bez zmian
Leipzig	3.83	2	2	0	bez zmian
Eintracht	3.80	4	3	-1	przeceniani
Moenchengladbach	3.78	7	4	-3	przeceniani
Stuttgart	3.78	10	5	-5	przeceniani
Wolfsburg	3.76	3	6	3	niedoceniani
Leverkusen	3.74	5	7	2	niedoceniani
Hoffenheim	3.73	12	8	-4	przeceniani
Union	3.69	9	9	0	bez zmian
Dortmund	3.62	6	10	4	niedoceniani
Freiburg	3.61	8	11	3	niedoceniani
Werder	3.41	11	12	1	niedoceniani
Hertha	3.30	15	13	-2	przeceniani
Augsburg	3.27	13	14	1	niedoceniani
Arminia	3.22	16	15	-1	przeceniani
Mainz	3.18	17	16	-1	przeceniani
Koeln	3.12	14	17	3	niedoceniani
Schalke	2.86	18	18	0	bez zmian

Najważniejsze funkcje

Pipes

Wszystkie poznane wcześniej funkcje możemy połączyć ze sobą za pomocą znaku %>%, zwanym *pipe operator*. Dzięki niemu możemy przekazywać wynik operacji do kolejnego wiersza kodu i kontynuować działanie na nowej zmiennej.

Najważniejsze funkcje

```
mNote2 <- data %>%
select(Number, Club, Note) %>%
filter(!is.na(Note)) %>%
mutate(NotePL = 7 - Note) %>%
group_by(Club) %>%
summarise(mean_Note = round(mean(NotePL),2)) %>%
left_join(table, by = "Club") %>%
arrange(desc(mean_Note)) %>%
mutate(NotePos = c(1:18)) %>%
mutate(diff = NotePos - TabPos) %>%
mutate(
  Ocena = case_when(
    diff > 0  ~ "niedoceniani",
    diff < 0      ~ "przeceniani",
    TRUE                      ~ "bez zmian"
  )
) %>%
```

Najważniejsze funkcje

Losowanie obserwacji

Aby wybrać losowe obserwacje z bazy danych korzystamy z funkcji *sample_n* (liczba) lub *sample_frac* (procent).

Przykładowy kod

```
sample_data <- sample_n(data, 50)

sample_data2 <- sample_frac(data, 0.11)
```

Najlepsze obserwacje

Funkcja *top_n* (liczba) pozwala nam wybrać z bazy najlepsze obserwacje według wybranej zmiennej.

Przykładowy kod

```
top_data <- top_n(data, 10, NotePL)
```

Najważniejsze funkcje

Inny sposób filtrowania

Znając już funkcję *filter* możemy użyć także operatora *%in%* i wybrać interesujące nas obserwacje z danej kolumny.

Przykładowy kod

```
data1<-data%>% filter(Club %in% c("Schalke","Union"))
```

Number	Name	Club	Note	NotePL
15	Robert Andrich	Union	3.09	3.91
24	Taiwo Awoniyi	Union	3.36	3.64
35	Sheraldo Becker	Union	3.43	3.57
120	Ralf Fährmann	Schalke	3.25	3.75
129	Marvin Friedrich	Union	2.84	4.16
167	Amine Harit	Schalke	4.54	2.46
194	Marcus Ingvartsen	Union	3.79	3.21
202	Ozan Kabak	Schalke	4.75	2.25
225	Robin Knoche	Union	3.47	3.53
245	Christopher Lenz	Union	3.31	3.69
254	Andreas Luthe	Union	3.16	3.84
262	Omar Mascarell	Schalke	4.04	2.96