Programowanie w R - Lab 2

Dariusz Kopka gr.1 DS19

Import bibliotek

```
suppressPackageStartupMessages(library(stringr))
suppressPackageStartupMessages(library(tibble))
suppressPackageStartupMessages(library(dplyr))
```

Funkcje pomocnicze do drukowania tabel.

```
# Wydrukuj całe tibble
print_tib <- function(x, y = 5) {
    x_mod <- x %>%
        mutate(No = row_number(), .before = 1)
    x_mod %>% head(y) %>% knitr::kable(row.names = FALSE)
}

# Wydrukuj 'y' pierwszych i ostatnich rzędów tibble
print_tib_sym <- function(x, y = 5, digits = NULL,
                          decimals = NULL, custom_col_names = NULL) {
    x_mod <- x %>%
        mutate(No = row_number(), .before = 1)

    if (!is.null(decimals)) {
        x_mod <- x_mod %>%
            mutate(across(where(is.numeric) & !matches("^(No|year)$"),
                          ~ format(round(., decimals), nsmall = decimals)))
    }

    if (nrow(x) < 2 * y) {
        combined_tibble <- x_mod
    } else {
        head_rows <- x_mod %>% head(y) %>% mutate(across(everything(), as.character))
        tail_rows <- x_mod %>% tail(y) %>% mutate(across(everything(), as.character))
        etc_row <- x_mod %>% head(1) %>% mutate(across(everything(),
                                                       ~ as.character('...')))
        combined_tibble <- bind_rows(head_rows, etc_row, tail_rows)
    }

    kable_args <- list(combined_tibble, row.names = FALSE, align = 'r')
    if (!is.null(custom_col_names)) {
        kable_args$col.names <- custom_col_names
    }

    do.call(knitr::kable, kable_args)
}
```

1 Zadanie 1

Plik `crypto.rds` zawiera notowania wybranych kryptowalut.

1. Wczytaj dane crypto.rds do R i zapoznaj się z nimi.
2. Wybierz z danych tylko te wiersze, które dotyczą Bitcoina.
3. Pozostaw w danych tylko kolumny Date i Close.
4. Popraw kolumnę Date w taki sposób, aby była typu Date.
5. Stwórz kolumnę Rate (r_t) na podstawie kolumny Close (p_t) zgodnie z następującą definicją: $r_t = \frac{p_t - p_{t-1}}{p_{t-1}}$
6. Posortuj dane według kolumny Rate w porządku malejącym.

1.1 Rozwiązania

1.1.1 Ad. 1. Wczytaj dane crypto.rds do R i zapoznaj się z nimi.

```
readr::read_rds("./crypto.rds") -> dane
dane[1:10, ]
```

```
## # A tibble: 10 x 8
##    Currency Date          Open  High   Low Close Volume     Market.Cap
##    <chr>    <chr>        <dbl> <dbl> <dbl> <dbl> <chr>      <chr>
##  1 0x       Feb 11, 2018 1.09   1.09 0.934 0.979 4,888,770  555,363,000
##  2 0x       Feb 10, 2018 1.14   1.2  0.986 1.1   10,828,700 576,535,000
##  3 0x       Feb 09, 2018 1.08   1.15 1.01  1.14  5,979,420  545,842,000
##  4 0x       Feb 08, 2018 0.989  1.13 0.989 1.07  12,992,800 501,142,000
##  5 0x       Feb 07, 2018 1.01   1.16 0.902 0.993 13,476,600 513,163,000
##  6 0x       Feb 06, 2018 0.822  1.03 0.640 1.02  18,000,300 415,693,000
##  7 0x       Feb 05, 2018 1.15   1.16 0.736 0.817 16,094,600 579,817,000
##  8 0x       Feb 04, 2018 1.39   1.39 1.05  1.15  13,450,500 703,996,000
##  9 0x       Feb 03, 2018 1.3    1.5  1.15  1.39  14,554,900 655,192,000
## 10 0x       Feb 02, 2018 1.61   1.61 1.05  1.3   26,332,600 807,909,000
```

1.1.2 Ad. 2. Wybierz z danych tylko te wiersze, które dotyczą Bitcoina.

```
tib <- dane %>%
    filter(Currency == "bitcoin")
print_tib_sym(tib, 3, decimals = 2)
```

No	Currency	Date	Open	High	Low	Close	Volume	Market.Cap
1	bitcoin	Feb 11, 2018	8616.13	8616.13	7931.10	8129.97	6,122,190,000	145,245,000,000
2	bitcoin	Feb 10, 2018	8720.08	9122.55	8295.47	8621.90	7,780,960,000	146,981,000,000
3	bitcoin	Feb 09, 2018	8271.84	8736.98	7884.71	8736.98	6,784,820,000	139,412,000,000
...	...	...	...	...	...	...	...	...

No	Currency	Date	Open	High	Low	Close	Volume	Market.Cap
1749	bitcoin	Apr 30, 2013	144.00	146.93	134.05	139.00	NA	1,597,780,000
1750	bitcoin	Apr 29, 2013	134.44	147.49	134.00	144.54	NA	1,491,160,000
1751	bitcoin	Apr 28, 2013	135.30	135.98	132.10	134.21	NA	1,500,520,000

1.1.3 Ad. 3. Pozostaw w danych tylko kolumny Date i Close.

```
tib %>%
    filter(Currency == "bitcoin") %>%
    select(Date, Close) -> tib
print_tib_sym(tib, 3, decimals = 2)
```

No	Date	Close
1	Feb 11, 2018	8129.97
2	Feb 10, 2018	8621.90
3	Feb 09, 2018	8736.98
...	...	...
1749	Apr 30, 2013	139.00
1750	Apr 29, 2013	144.54
1751	Apr 28, 2013	134.21

1.1.4 Ad. 4. Popraw kolumnę Date w taki sposób, aby była typu Date

```
tib %>%
    mutate(Date = as.Date(tib$Date, format = "%b %d, %Y")) -> tib
print_tib_sym(tib, 3, decimals = 2)
```

No	Date	Close
1	2018-02-11	8129.97
2	2018-02-10	8621.90
3	2018-02-09	8736.98
...	...	...
1749	2013-04-30	139.00
1750	2013-04-29	144.54
1751	2013-04-28	134.21

1.1.5 Ad. 5. Stwórz kolumnę Rate (r_t) na podstawie kolumny Close (p_t) zgodnie z następującą definicją: $r_t = \frac{p_t - p_{t-1}}{p_{t-1}}$

```
tib %>%
    arrange(Date) %>%
    mutate(CloseDayBefore = lag(.$Close, 1)) %>%
    mutate(Rate = (Close - CloseDayBefore)/CloseDayBefore) -> tib
# moje funkcje pritnowania się niedoskonałe i pozwalają określić
# ilość miejsc dziesiętnych dla wszystkich wartości 'numerical'
# jednocześnie.
print_tib_sym(tib, 5, decimals = 4)
```

No	Date	Close	CloseDayBefore	Rate
1	2013-04-28	134.2100	NA	NA
2	2013-04-29	144.5400	134.2100	0.0770
3	2013-04-30	139.0000	144.5400	-0.0383
4	2013-05-01	116.9900	139.0000	-0.1583
5	2013-05-02	105.2100	116.9900	-0.1007
...	...	...	...	...
1747	2018-02-07	7621.3000	7754.0000	-0.0171
1748	2018-02-08	8265.5900	7621.3000	0.0845
1749	2018-02-09	8736.9800	8265.5900	0.0570
1750	2018-02-10	8621.9000	8736.9800	-0.0132
1751	2018-02-11	8129.9700	8621.9000	-0.0571

1.1.6 Ad. 6. Posortuj dane według kolumny Rate w porządku malejącym.

```
tib %>%
    arrange(desc(Rate)) -> tib
print_tib_sym(tib, 10, decimals = 4)
```

No	Date	Close	CloseDayBefore	Rate
1	2013-11-18	703.5600	492.1100	0.4297
2	2013-12-19	691.9600	522.7000	0.3238
3	2017-12-07	17899.7000	14291.5000	0.2525
4	2017-07-20	2817.6000	2273.4300	0.2394
5	2013-11-21	722.4300	590.8300	0.2227
6	2017-12-06	14291.5000	11916.7000	0.1993
7	2014-03-03	667.7600	559.7900	0.1929
8	2015-01-15	209.8400	178.1000	0.1782
9	2013-11-26	928.1000	799.1100	0.1614
10	2017-07-17	2228.4100	1929.8200	0.1547
...	...	...	...	...
1742	2013-11-19	584.6100	703.5600	-0.1691
1743	2014-04-10	365.1800	442.7300	-0.1752
1744	2015-08-18	211.0800	257.9800	-0.1818
1745	2017-09-14	3154.9500	3882.5900	-0.1874
1746	2014-03-27	471.2400	580.8300	-0.1887
1747	2013-12-16	705.9700	876.1200	-0.1942
1748	2013-12-06	829.4500	1045.1100	-0.2064
1749	2015-01-14	178.1000	225.8600	-0.2115

No	Date	Close	CloseDayBefore	Rate
1750	2013-12-18	522.7000	682.1200	-0.2337
1751	2013-04-28	134.2100	NA	NA

2 Zadanie 2

Plik `albums.csv` zawiera następujące dane dotyczące albumów muzycznych:

- `artist_id` – identyfikator artysty,
- `album_title` – tytuł albumu,
- `genre` – gatunek muzyczny,
- `year_of_pub` – rok publikacji,
- `num_of_tracks` – liczba piosenek na płycie,
- `num_of_sales` – liczba sprzedanych płyt,
- `rolling_stone_critic` – ocena Rolling Stone Magazine,
- `mtv_critic` – ocena MTV,
- `music_maniac_critic` – ocena Music Maniac.

Przeprowadź analizę tej próbki stosując dplyr.

2.1 Rozwiązanie

```
suppressPackageStartupMessages(library(dplyr))
suppressPackageStartupMessages(library(ggplot2))

# Wczytywanie danych
albums <- read.csv("albums.csv")

# Najlepiej sprzedające się albumy
top_sold_albums <- albums %>%
    group_by(artist_id, album_title) %>%
    reframe(artist_id, album_title, genre, year_of_pub, num_of_sales) %>%
    arrange(desc(num_of_sales))
knitr::kable(top_sold_albums[1:10, ], booktabs = TRUE,
             row.names = TRUE, caption = "Najlepiej sprzedające się albumy")
```

Table 6: Najlepiej sprzedające się albumy

	artist_id	album_title	genre	year_of_pub	num_of_sales
1	12047	Decent Waterbuck	Heavy Metal	2010	999994
2	4842	Little Of Pahari	Hard Rock	2009	999991
3	25396	Gabby Malay American Boy Sudden	K-Pop	2019	999977
4	37484	Every Country Toys Chameleon	Rock	2010	999976
5	8123	cuddly Trick Sudden Javanese Kia Hills	Unplugged	2012	999969
6	9202	Algerian Color Sufficient	Rap	2012	999958
7	5330	Move Like a Seat	Heavy Metal	2004	999950
8	23827	Competitive Severe Inside My Girls Lion	Heavy Metal	2016	999925
9	730	Cow Matteo High Computer Systems	Indie	2009	999924
10	4073	Try Rich Marketing	Jazz	2009	999915

```
# Top 10 gatunków muzycznych wg sprzedaży
top_genres_by_sales <- albums %>%
    group_by(genre) %>%
    summarise(total_sales = sum(num_of_sales)) %>%
    arrange(desc(total_sales))
knitr::kable(top_genres_by_sales[1:10, ], booktabs = TRUE,
             row.names = TRUE, caption = "Top 10 gatunków wg sprzedaży")
```

Table 7: Top 10 gatunków wg sprzedaży

	genre	total_sales
1	Indie	4699805459
2	Pop	3916637999
3	Rap	2888914186
4	Latino	1953132011
5	Pop-Rock	1939002474
6	Punk	1914131397
7	Rock	1899856670
8	Dance	1882029421
9	Compilation	1009992493
10	Gospel	1005454870

```
# Top 10 gatunków muzycznych wg ilości albumów
top_genres_by_number <- albums %>%
    group_by(genre) %>%
    summarise(total_albums = n()) %>%
    arrange(desc(total_albums))
knitr::kable(top_genres_by_number[1:10, ], booktabs = TRUE,
             row.names = TRUE, caption = "Top 10 gatunków wg ilości albumów")
```

Table 8: Top 10 gatunków wg ilości albumów

	genre	total_albums
1	Indie	9384
2	Pop	7755
3	Rap	5788
4	Latino	3898
5	Pop-Rock	3880
6	Rock	3804
7	Punk	3787
8	Dance	3775
9	Gospel	2008
10	Compilation	2003

```
# Top 10 tytułów albumów
top_album_names <- albums %>%
    group_by(album_title) %>%
    summarize(album_count = n()) %>%
    arrange(desc(album_count))
```

```
knitr::kable(top_album_names[1:10, ], booktabs = TRUE,
             row.names = TRUE, caption = "Top 10 tytułów albumów muzycznych")
```

Table 9: Top 10 tytułów albumów muzycznych

	album_title	album_count
1	Of Love	211
2	In the Sky	124
3	Loves	111
4	Next to	88
5	Of	87
6	Noodles Moves In the Sky Ultra-Maximus	85
7	The	84
8	Vampire Chip A Quiet	83
9	China	82
10	Afrodite Potion	70

```
# Najbardziej płodni artyści
top_artists <- albums %>%
    group_by(artist_id) %>%
    summarize(sum_tracks = sum(num_of_tracks), sum_albums = n(), sum_sales = sum(num_of_sales))

# - po ilości sprzedanych płyt
top_artists %>%
    arrange(desc(sum_sales)) %>%
    reframe(artist_id = artist_id, sum_sales = sum_sales) %>%
    head(10) %>%
    knitr::kable(booktabs = TRUE, row.names = TRUE,
                 caption = "Najlepiej sprzedający się artyści")
```

Table 10: Najlepiej sprzedający się artyści

	artist_id	sum_sales
1	11290	6267844
2	33896	6072907
3	44614	5643206
4	7889	5573891
5	49450	5401584
6	5508	5329169
7	27648	5267296
8	40017	5214678
9	22799	5196851
10	15528	5193117

```
# - po ilości wydanych albumów
top_artists %>%
    arrange(desc(sum_albums)) %>%
    reframe(artist_id = artist_id, sum_albums = sum_albums) %>%
    head(10) %>%
```

```
    knitr::kable(booktabs = TRUE, row.names = TRUE,
                 caption = "Artyści, którzy wydali najwięcej płyt")
```

Table 11: Artyści, którzy wydali najwięcej płyt

	artist_id	sum_albums
1	10708	11
2	1608	9
3	4044	9
4	8330	9
5	11106	9
6	11290	9
7	22058	9
8	29669	9
9	33896	9
10	36534	9

```
# Szukamy korelacji między ocenami krytyków a wynikamie sprzedaży
korelacja <- cor(albums[, c("num_of_sales", "rolling_stone_critic",
                            "mtv_critic", "music_maniac_critic")])
knitr::kable(korelacja)
```

	num_of_sales	rolling_stone_critic	mtv_critic	music_maniac_critic
num_of_sales	1.0000000	-0.0024935	0.0005231	-0.0011949
rolling_stone_critic	-0.0024935	1.0000000	0.0011983	-0.0012693
mtv_critic	0.0005231	0.0011983	1.0000000	-0.0018090
music_maniac_critic	-0.0011949	-0.0012693	-0.0018090	1.0000000

Próbka zawiera 100000 albumów 43225 artystów wydanych od 2000 do 2019 roku w 38 gatunkach.

Macież korelacji nie wykazała liniowych zależności ani pomiędzy ocenami a ilością sprzedanych płyt, ani między samymi ocenami.

Dane wyglądają na wygenerowane losowo.

---

3 Zadanie 3

Plik `suicides.rds` zawiera informacje na temat liczby popełnionych samobójstw w 101 krajach świata na przestrzeni lat 1985 – 2016 z uwzględnieniem podziału na płeć oraz grupę wiekową.

1. Wskaż pięć krajów w których na przestrzeni lat 1985 – 2016 popełniono najwięcej / najmniej samobójstw na 100k mieszkańców.
2. Dla każdego roku badania wyznacz łączną liczbę samobójstw popełnionych na 100k mieszkańców na świecie.
3. Ustal łączną liczbę samobójstw popełnionych na 100k mieszkańców na przestrzeni całej próby w podziale na płeć oraz wiek.
4. Dla każdego roku badania wskaż trzy kraje, w których odnotowano największą liczbę samobójstw.
5. Znajdź kraj w którym nastąpiła największa / najmniejsza zmiana pomiędzy liczbą samobójstw na 100k mieszkańców w najgorszym roku (najwięcej samobójstw) i najlepszym roku (najmniej samobójstw).

3.1 Rozwiązanie

```
suicides <- readRDS('suicides.rds')
```

3.1.1 1. Wskaż pięć krajów w których na przestrzeni lat 1985 – 2016 popełniono najwięcej / najmniej samobójstw na 100k mieszkańców.

```
suicides %>%
    group_by( country) %>%
    summarise(avg_suicides = mean(suicides.100k.pop, .groups = "drop_last")) %>%
    arrange(desc(avg_suicides)) %>%
    mutate(across(matches("^avg_suicides$"),
                  ~ format(round(., 4), nsmall = 2))) %>%
    print_tib_sym(5)
```

No	country	avg_suicides
1	Lithuania	40.4156
2	Sri Lanka	35.2952
3	Russian Federation	34.8924
4	Hungary	32.7615
5	Belarus	31.0759
...	...	...
97	Oman	0.7361
98	Antigua and Barbuda	0.5529
99	Jamaica	0.5218
100	Dominica	0.0000
101	Saint Kitts and Nevis	0.0000

3.1.2 2. Dla każdego roku badania wyznacz łączną liczbę samobójstw popełnionych na 100k mieszkańców na świecie.

```
suicides %>%
    group_by(year) %>%
    summarise(avg_suicides = mean(suicides.100k.pop, .groups = "drop_last")) %>%
    arrange(desc(avg_suicides)) %>%
    mutate(across(matches("^avg_suicides$"),
                  ~ format(round(., 4), nsmall = 2))) %>%
    print_tib_sym(5)
```

No	year	avg_suicides
1	1995	15.6627
2	1996	15.3054
3	1997	14.9544
4	1998	14.9269
5	1999	14.5320
…	…	…
28	2010	11.2159
29	2013	11.1080
30	2015	11.0941
31	2011	11.0153
32	2014	11.0115

3.1.3 3. Ustal łączną liczbę samobójstw popełnionych na 100k mieszkańców na przestrzeni całej próby w podziale na płeć oraz wiek.

```
suicides %>%
    group_by(sex, age) %>%
    summarise(avg_suicides = mean(suicides.100k.pop), .groups = "drop_last") %>%
    arrange(sex, age) %>%
    mutate(across(matches("^avg_suicides$"),
                  ~ format(round(., 4), nsmall = 2))) %>%
    print_tib(100)
```

No	sex	age	avg_suicides
1	female	15-24 years	4.3280
2	female	25-34 years	4.5732
3	female	35-54 years	5.9165
4	female	5-14 years	0.4623
5	female	55-74 years	7.1234
6	female	75+ years	9.9198
1	male	15-24 years	13.5663
2	male	25-34 years	19.8006
3	male	35-54 years	23.9784
4	male	5-14 years	0.7778
5	male	55-74 years	25.1877
6	male	75+ years	37.9910

No	sex	age	avg_suicides

3.1.4 4. Dla każdego roku badania wskaż trzy kraje, w których odnotowano największą liczbę samobójstw.

```
suicides %>%
    group_by(year, country) %>%
    summarise(suicides_no = sum(suicides_no), .groups = "drop_last") %>%
    arrange(year, desc(suicides_no)) %>%
    slice_head(n = 3) %>%
    ungroup() %>%
    print_tib_sym(6)
```

No	year	country	suicides_no
1	1985	United States	29446
2	1985	Japan	23257
3	1985	France	12501
4	1986	United States	30892
5	1986	Japan	25484
6	1986	France	12529
...	...	...	...
91	2015	United States	44189
92	2015	Russian Federation	25432
93	2015	Japan	23092
94	2016	Thailand	4117
95	2016	Romania	1953
96	2016	Netherlands	1886

3.1.5 5. Znajdź kraj w którym nastąpiła największa / najmniejsza zmiana pomiędzy liczbą samobójstw na 100k mieszkańców w najgorszym roku (najwięcej samobójstw) i najlepszym roku (najmniej samobójstw).

```
avg_suicides <- suicides %>%
    group_by(year, country) %>%
    summarise(avg_suicides_100k_pop_annual = mean(suicides.100k.pop),
              .groups = "drop")

avg_suicides %>%
    group_by(country) %>%
    summarise(number_of_years_considered = n(),
              min_annual_avg_per_100k_pop = min(avg_suicides_100k_pop_annual),
              max_annual_avg_per_100k_pop = max(avg_suicides_100k_pop_annual),
              .groups = "drop") %>%
    mutate(change = max_annual_avg_per_100k_pop - min_annual_avg_per_100k_pop) %>%
    mutate(across(matches("annual_avg") | matches("change"),
                  ~ format(round(., 4), nsmall = 2))) %>%
    arrange(desc(change)) -> suicides_change
```

```
col_names <- c("No", "Kraj", "Ilość lat branych pod uwagę",
               "Minimalna średnia", "Maksymalna średnia", "Zmiana")

suicides_change %>%
    print_tib_sym(5, custom_col_names = col_names)
```

No	Kraj	Ilość lat branych pod uwagę	Minimalna średnia	Maksymalna średnia	Zmiana
1	Republic of Korea	31	9.5483	43.2933	33.7450
2	Suriname	28	7.0225	39.2367	32.2142
3	Guyana	25	5.4050	33.5892	28.1842
4	Estonia	21	17.1150	45.2658	28.1508
5	Montenegro	10	0.0000	27.9608	27.9608
…	…	…	…	…	…
97	Cabo Verde	1	11.1533	11.1533	0.0000
98	Dominica	1	0.0000	0.0000	0.0000
99	Macau	1	14.3117	14.3117	0.0000
100	Mongolia	1	18.4390	18.4390	0.0000
101	Saint Kitts and Nevis	3	0.0000	0.0000	0.0000

4 Zadanie 4

Katalog `gapps` zawiera trzy pliki z informacji o aplikacjach z Google Play Store:

- free_apps.rds – ocenione aplikacje darmowe,
- paid_apps.rds – ocenione aplikacje płatne,
- norat_apps.rds – nie ocenione aplikacje płatne i darmowe.

Połącz ten dane w jeden wspólny plik i zapisz wynik pracy w postaci pliku CSV.

4.1 Rozwiązanie

```
free_apps <- readr::read_rds('gapps/free_apps.rds')
paid_apps <- readr::read_rds('gapps/paid_apps.rds')
norat_apps <- readr::read_rds('gapps/norat_apps.rds')

bind_rows(free_apps, paid_apps, norat_apps) %>%
    readr::write_csv('gapps/merged.csv')
```

---

5 Zadanie 5

Katalog movies zawiera trzy pliki dotyczące filmów:

- movies.rds – podstawowe dane na temat filmu,
- ratings.rds – jednostkowe oceny jakie filmom przyznali użytkownicy,
- tags.rds – tagi jakie do poszczególnych filmów przypisali użytkownicy.

Wykonaj następujące zadania:

1. Wyznacz średnie oceny filmów oraz liczbę osób, które oceniły każdy film i dołącz te informacje do informacji na temat filmu.
2. Ustal czas dodania ostatniego tagu do każego z filmów i dodaj tą informację do informacji na temat filmu.
3. Dokonaj agregacji wszystkich różnych tagów dotyczących filmu, a następnie dołącz je o informacji na temat filmu.

5.1 Rozwiązanie

5.1.1 1. Wyznacz średnie oceny filmów oraz liczbę osób, które oceniły każdy film i dołącz te informacje do informacji na temat filmu.

```
movies <- readr::read_rds('movies/movies.rds')
ratings <- readr::read_rds('movies/ratings.rds')
tags <- readr::read_rds('movies/tags.rds')

# zamieniam "Adventure|Animation" na "Adventure Animation" dla czytelności tabel,
# ponieważ długie stringi nie chcą się 'wrapować'
movies <- movies %>%
        mutate(genres = str_replace_all(genres, "\\|", ", "))

movies_rated <- ratings %>%
    group_by(movieId) %>%
    summarise(avg_score = mean(rating), users_rated = n()) %>%
    mutate(across(matches("avg_score"),
                  ~ format(round(., 2), nsmall = 2))) %>%
    left_join(movies, by = c("movieId"))
print_tib_sym(movies_rated, 5)
```

No	movieId	avg_score	users_rated	title	genres
1	1	3.92	215	Toy Story (1995)	Adventure, Animation, Children, Comedy, Fantasy
2	2	3.43	110	Jumanji (1995)	Adventure, Children, Fantasy
3	3	3.26	52	Grumpier Old Men (1995)	Comedy, Romance
4	4	2.36	7	Waiting to Exhale (1995)	Comedy, Drama, Romance
5	5	3.07	49	Father of the Bride Part II (1995)	Comedy
…	…	…	…	…	…

No	movieId	avg_score	users_rated	title	genres
9720	193581	4.00	1	Black Butler: Book of the Atlantic (2017)	Action, Animation, Comedy, Fantasy
9721	193583	3.50	1	No Game No Life: Zero (2017)	Animation, Comedy, Fantasy
9722	193585	3.50	1	Flint (2017)	Drama
9723	193587	3.50	1	Bungo Stray Dogs: Dead Apple (2018)	Action, Animation
9724	193609	4.00	1	Andrew Dice Clay: Dice Rules (1991)	Comedy

5.1.2 2. Ustal czas dodania ostatniego tagu do każego z filmów i dodaj tą informację do informacji na temat filmu.

```
movies_rated_ts <- tags %>%
    group_by(movieId) %>%
    summarise(last_tag_timestamp = max(timestamp)) %>%
    right_join(movies_rated, by = c("movieId"))

movies_rated_ts %>%
#    select(movieId, title, last_tag_timestamp) %>%
    print_tib_sym(5)
```

No	movieId	last_tag_timestamp	avg_score	users_rated	title	genres
1	1	1525286013	3.92	215	Toy Story (1995)	Adventure, Animation, Children, Comedy, Fantasy
2	2	1528843932	3.43	110	Jumanji (1995)	Adventure, Children, Fantasy
3	3	1143424860	3.26	52	Grumpier Old Men (1995)	Comedy, Romance
4	5	1137373903	3.07	49	Father of the Bride Part II (1995)	Comedy
5	7	1137375642	3.19	54	Sabrina (1995)	Comedy, Romance
…	…	…	…	…	…	…
9720	193581	NA	4.00	1	Black Butler: Book of the Atlantic (2017)	Action, Animation, Comedy, Fantasy
9721	193583	NA	3.50	1	No Game No Life: Zero (2017)	Animation, Comedy, Fantasy
9722	193585	NA	3.50	1	Flint (2017)	Drama
9723	193587	NA	3.50	1	Bungo Stray Dogs: Dead Apple (2018)	Action, Animation
9724	193609	NA	4.00	1	Andrew Dice Clay: Dice Rules (1991)	Comedy

5.1.3 3. Dokonaj agregacji wszystkich różnych tagów dotyczących filmu, a następnie dołącz je o informacji na temat filmu.

```
tags_aggregated <- tags %>%
    group_by(movieId) %>%
    summarise(tags = str_c(tag, collapse = ", "))
```

```
movies_rated_ts_tags <- left_join(movies_rated_ts, tags_aggregated, by = c("movieId"))
print_tib_sym(movies_rated_ts_tags, 5)
```

No	movieId	last_tag_timestamp	avg_score	users_rated	title	genres	tags
1	1	1525286013	3.92	215	Toy Story (1995)	Adventure, Animation, Children, Comedy, Fantasy	pixar, pixar, fun
2	2	1528843932	3.43	110	Jumanji (1995)	Adventure, Children, Fantasy	fantasy, magic board game, Robin Williams, game
3	3	1143424860	3.26	52	Grumpier Old Men (1995)	Comedy, Romance	moldy, old
4	5	1137373903	3.07	49	Father of the Bride Part II (1995)	Comedy	pregnancy, remake
5	7	1137375642	3.19	54	Sabrina (1995)	Comedy, Romance	remake
…	…	…	…	…	…	…	…
9720	193581	NA	4.00	1	Black Butler: Book of the Atlantic (2017)	Action, Animation, Comedy, Fantasy	NA
9721	193583	NA	3.50	1	No Game No Life: Zero (2017)	Animation, Comedy, Fantasy	NA
9722	193585	NA	3.50	1	Flint (2017)	Drama	NA
9723	193587	NA	3.50	1	Bungo Stray Dogs: Dead Apple (2018)	Action, Animation	NA
9724	193609	NA	4.00	1	Andrew Dice Clay: Dice Rules (1991)	Comedy	NA