03. Praca z danymi - zadania

3.0 Należy pobrać zbiór danych Movie Lens spakowany w pliku ml-1m.zip dostępny tutaj: http://grouplens.org/datasets/movielens/1m/
Następnie należy zaznajomić się z danymi czytając plik README. Później należy wczytać wszystkie trzy pliki jako osobne ramki danych do R przy pomocy funkcji `read.table`. Zwróć uwagę na separator i zastanów się jak rozwiązać ten problem. Po pomyślnym wczytaniu danych przy pomocy funkcji stri_sub z pakietu stringi stwórz nową kolumnę w ramce danych, która zawierajać będzie rok powstania filmu. Wywołaj `stri_sub(title,-5,-2)` w funkcji `mutate()`. Następnie odpowiedz na poniższe pytania

A. ile jest wszystkich filmów
B. ile filmów powstało w poszczególnych latach
C. jak wygląda rozkład płci oraz grup wiekowych wśród użytkowników
D. jaki gatunek filmowy jest najczęstszy
E. jaki jest najlepszy film wszechczasów, (najlepszy, czyli ma najwyższą średnią ocenę) - to zadanie możesz rozwiązać wykonując złączenie (join) zbioru movies i ratings
F. wykonaj poprzedni punkt, odrzucając wcześniej filmy które nie uzyskały wystarczająco dużo głosów (np 100)
G. jaki jest najlepszy film według kobiet i według mężczyzn
H. jaki jest średni rok oglądanego filmu w poszczególnych grupach wiekowych
I. jaka jest najpopularniejsza grupa wiekowa dla każdego gatunku filmowego
J. jakie trzy gatunki filmowe są najczęściej oglądane przez kobiety i mężczyzn

3.1 Napisz funkcję rozwin(), która przekształca daną macierz rozmiaru n x m (niekoniecznie liczbową) z ustawionym atrybutem dimnames na ramkę danych zawierającą nm obserwacji i trzy kolumny o nazwach zadanych za pomocą odpowiedniego argumentu funkcji.
Wartości z macierzy mają znajdować się w pierwszej kolumnie, a w kolejnych dwóch - kombinacje nazw wierszy i kolumn odpowiadające podanym poziom czynnika.
Dla przykładu, obiekt WorldPhones (wbudowany) zawiera dane o liczbie telefonów (w tysiącach) w różnych regionach świata w wybranych latach.
Wynikiem wywołania rozwin(WorldPhones, c("ile", “gdzie”, “kiedy")) może być:

```
    ile    gdzie kiedy
...
2  60423   N.Amer  1956
3  64721   N.Amer  1957
...
9  29990   Europe  1956
```

10 32510 Europe 1957
...

3.2 Napisz funkcję odwrotną do funkcji z zad. 3.1. Dana jest ramka danych zawierająca nm wierszy oraz 3~kolumny (pierwsza -- dowolnego typu, druga i~trzecia -- typu czynnikowego, odpowiednio o n i m poziomach). Obserwacje zawierają wszystkie możliwe kombinacje poziomów dwóch czynników, ale nie możemy założyć, że są one koniecznie ułożone w~jakimś określonym porządku (funkcja ma działać dla dowolnej permutacji obserwacji). Wynikiem ma być macierz rozmiaru n x m o elementach pochodzących z pierwszej kolumny ramki danych. Atrybut dimnames ustawiamy na podstawie wartości poziomów pierwszego i drugiego czynnika.

3.3 Napisz funkcję uniqueRows() (bez korzystania z funkcji duplicated()), która jako argument przyjmuje ramkę danych. Funkcja ta usuwa z ramki danych duplikaty (całych) wierszy i zwraca tak otrzymaną ramkę danych. Jeśli wiersze w wejściowej ramce danych są nazwane (atrybut row.names) kolejnymi liczbami całkowitymi, zadbaj o to, aby wyjściowa ramka danych również miała tę własność.

3.4 Nazwijmy macierzą prawieortogonalną taką macierz $A \in \mathbb{R}^{n x m}$, że wszystkie jej kolumny są wektorami takimi, że są one do siebie parami prostopadłe (ze świecą szukać definicji takiej macierzy w mądrych książkach do matematyki). Zwróćmy uwagę, że nie zakładamy tego, że macierz jest kwadratowa. Nie zakładamy też, że wektory tej macierzy są wektorami jednostkowymi.

Napisz funkcję isNearlyOrthoMatrix(), która przyjmuje jako argument macierz A, a następnie zwraca wartość TRUE, jeśli macierz jest macierzą prawieortogonalną, a FALSE, gdy nią nie jest.

```
M1 <- matrix(c(7, 0, 0, 0, 5, 0), ncol = 2)
print(M1)
## [,1] [,2]
## [1,] 7 0
## [2,] 0 5
## [3,] 0 0
isNearlyOrthoMatrix(M1)
## [1] TRUE
M2 <- matrix(c(0.96, 0.28, -0.28, 0.96), ncol = 2)
print(M2)
```

```
## [,1] [,2]
## [1,] 0.96 -0.28
## [2,] 0.28 0.96
isNearlyOrthoMatrix(M2)
## [1] TRUE
M3 <- matrix(c(0.96, -0.28, -0.28, 0.96), ncol = 2)
print(M3)
## [,1] [,2]
## [1,] 0.96 -0.28
## [2,] -0.28 0.96
isNearlyOrthoMatrix(M3)
## [1] FALSE
```

3.5 Prawą macierzą stochastyczną nazywamy macierz kwadratową,
której elementami są nieujemne liczby rzeczywiste i~w~której każdy wiersz
sumuje się do jedynki. Przykładem prawej macierzy stochastycznej jest macierz:

R =
0.5 & 0.3 & 0.2 \\
0.2 & 0.8 & 0 \\
0.3 & 0.3 & 0.4

Napisz funkcję doStochastycznej(), która przyjmuje kwadratową macierz
(niekoniecznie prawą stochastyczną) i~wykonuje następujące czynności:

1. Sprawdza, czy wszystkie elementy są nieujemne (jeśli nie, użyj \func{stop()}),
2. Sprawdza, czy w każdym wierszu jest co najmniej jeden elementwiększy od zera (jeśli nie, użyj \func{stop()}),
3. Tworzy nową macierz na bazie wejściowej macierzy, w której wiersze są ,,unormowane'', czyli sumują się do jedynki. Dla przykładu, dla następującego wiersza: 5, 3, 2, w macierzy zwracanej ten wierszy powinien wyglądać tak: 0.5, 0.3, 0.2.

Przykład:

INPUT =

5 & 3 & 2 \\
20 & 80 & 0 \\
0 & 0 & 1

OUTPUT =
0.5 & 0.3 & 0.2 \\
0.2 & 0.8 & 0 \\
0 & 0 & 1

3.6

Napisz funkcję aggregation(), która daną ramkę danych rozszerzy o dodatkową kolumnę zawierającą dane zagregowane względem wskazanego czynnika, definiującego podział obserwacji na podgrupy. Funkcja ma przyjmować jako argumenty wejściową ramkę danych, nazwę kolumny podlegającej agregacji, nazwę wskazanej kolumny typu factor, funkcję agregującą oraz opcjonalne dodatkowe argumenty przesyłane do danej funkcji agregującej przy użyciu ,,...".

Nowo powstała kolumna powinna być nazwana zgodnie ze schematem Czynnik_zmienna_funkcja (nazwę użytej funkcji możesz poznać wywołując deparse(substitute(Nazwa-Argumentu)). Przy próbie agregacji danych nieliczbowych bądź względem zmiennej innego typu niż factor zwróć błąd.

(df <- data.frame(Customer=c("A", "B", "B", "C", "C", "C"), Profit=c(1:4, NA, 6)))

Przykład zastosowania funkcji na powyższej ramce danych wygląda następująco:

```
agregation(df, "Profit", "Customer", mean, na.rm = TRUE)
##   Customer Profit Customer_Profit_mean
## 1        A      1                  1.0
## 2        B      2                  2.5
## 3        B      3                  2.5
## 4        C      4                  5.0
## 5        C     NA                  5.0
## 6        C      6                  5.0
```