



LISTS and DATAFRAMES in R | СПИСКИ і ФРЕЙМИ ДАНИХ

Table of Contents | Зміст

- [About the Dataset | Про базу даних](#)
- [Lists | Списки](#)
- [Data frames | Фрейми даних](#)

Estimated Time Needed: **15 min** | Необхідний час: **15 хв**

About the Dataset | Про базу даних

Imagine you got many movie recommendations from your friends and compiled all of the recommendations in a table, with specific info about each movie.

The table has one row for each movie and several columns

Уявіть, що у вас є багато рекомендацій фільмів від ваших друзів і зібрані вони в таблиці з конкретною інформацією про кожен фільм.

У цій таблиці одному рядкові відповідає один фільм а також кілька колонок.

- **name** - The name of the movie | Назва фільму
- **year** - The year the movie was released | Рік релізу фільму
- **length_min** - The lenght of the movie in minutes | Тривалість фільму (хвилини)
- **genre** - The genre of the movie | Жанр фільму
- **average_rating** - Average rating on Imdb | Середній рейтинг на сайті IMDB
- **cost_millions** - The movie's production cost in millions | Вартість фільму (млн доларів США)
- **sequences** - The amount of sequences | Кількість частин
- **foreign** - Indicative of whether the movie is foreign (1) or domestic (0) | Фільм є зарубіжним (1) чи вітчизняним (0)?
- **age_restriction** - The age restriction for the movie | Вікові обмеження для перегляду фільму

Part of the dataset can be seen below

Ви можете побачити частину бази даних нижче

name	year	length_min	genre	average_rating	cost_millions	foreign	age_restriction
Toy Story	1995	81	Animation	8.3	30	0	0
Akira	1998	125	Animation	8.1	10.4	1	14
The Breakfast Club	1985	97	Drama	7.9	1	0	14
The Artist	2011	100	Romance	8	15	1	12
Modern Times	1936	87	Comedy	8.6	1.5	0	10
Fight Club	1999	139	Drama	8.9	63	0	18
City of God	2002	130	Crime	8.7	3.3	1	18
The Untouchables	1987	119	Drama	7.9	25	0	14
Star Wars	1977	121	Action	8.7	11	0	10
American Beauty	1999	122	Drama	8.4	15	0	14
Room	2015	118	Drama	8.3	13	1	14
Dr. Strangelove	1964	94	Comedy	8.5	1.8	1	10
The Ring	1998	95	Horror	7.3	1.2	1	18
Monty Python and the Holy Grail	1975	91	Comedy	8.3	0.4	1	18
High School Musical	2006	98	Comedy	5.2	4.2	0	0
Shaun of the Dead	2004	99	Horror	8	6.1	1	18
Taxi Driver	1976	113	Crime	8.3	1.3	1	14
The Shawshank Redemption	1994	142	Crime	9.3	25	0	16
Interstellar	2014	169	Adventure	8.6	165	0	10
Casino	1995	178	Biography	8.2	50	0	18
The Goodfellas	1990	145	Biography	8.7	25	0	14
Blue is the Warmest Colour	2013	179	Romance	7.8	4.5	1	18
Black Swan	2010	108	Thriller	8	13	0	16
Back to the Future	1985	116	Sci-fi	8.5	19	0	0
The Wave	2008	107	Thriller	7.6	5.5	1	16
Whiplash	2014	106	Drama	8.5	3.3	1	12
The Grand Hotel Budapest	2014	100	Crime	8.1	25.5	0	14
Jumanji	1995	104	Fantasy	6.9	65	0	12
The Eternal Sunshine of the Spotless Mind	2004	108	Drama	8.3	20	0	14
Chicago	2002	113	Comedy	7.2	45	0	12

Lists | Списки

First of all, we're gonna take a look at lists in R. A list is a sequenced collection of different objects of R, like vectors, numbers, characters, other lists as well, and so on. You can consider a list as a container of correlated information, well structured and easy to read. A list accepts items of different types, but a vector (or a matrix, which is a multidimensional vector) doesn't. To create a list just type **list()** with your content inside the parenthesis and separated by commas. Let's try it!

Перш за все, ми розглянемо списки в R. Список - це послідовний набір різних об'єктів R, наприклад векторів, чисел, символів, інших списків тощо. Ви можете уявити список як контейнер з інформацією, що є добре структурованим і легко читається. Список може містити елементи різних типів, тоді як вектор (або матриця, що є багатовимірним вектором) не має. Щоб створити список, просто введіть **list()** зі своїм вмістом у дужках і розділіть їх комами. Спробуємо це!

In []:

```
movie <- list("Toy Story", 1995, c("Animation", "Adventure", "Comedy"))
```

In the code above, the variable `movie` contains a list of 3 objects, which are a string, a numeric value, and a vector of strings. Easy, eh? Now let's print the content of the list. We just need to call its name.

У наведеному вище коді змінна `movie` містить список з 3 об'єктів, які є символом, числом та символьним вектором. Легко, правда? Тепер надрукуємо вміст списку. Нам просто потрібно вказати його назву.

In []:

```
movie
```

A list has a sequence and each element of a list has a position in that sequence, which starts from 1. If you look at our previous example, you can see that each element has its position represented by double square brackets `"[[ ]]"`.

Список має певну послідовність, і кожен елемент списку має місце в цій послідовності, що починається з 1. Якщо подивитися на наш попередній приклад, ви можете побачити, що кожен елемент має свою позицію, представлену подвійними квадратними дужками `"[[ ]]"`.

Accessing items in a list | Доступ до елементів списку

It is possible to retrieve only a part of a list using the **single_square** bracket operator `"[ ]"`. This operator can be also used to get a single element in a specific position. Take a look at the next example:

Можна отримати лише частину списку, використовуючи оператор **одиничні квадратні дужки** `"[ ]"`. Цей оператор також може використовуватися для отримання одного елемента у певній позиції. Погляньте на наступний приклад:

The index number 2 returns the second element of a list, if that element exists:

Індекс №2 витягує другий елемент списку, якщо такий елемент існує:

In []:

```
movie[2]
```

Or you can select a part or interval of elements of a list. In our next example we are retrieving the 1st, 2nd, and 3rd elements:

Або ви можете вибрати частину чи інтервал елементів списку. У нашому наступному прикладі ми витягнемо елементи 2 і 3:

In []:

```
movie[2:3]
```

It looks a little confusing, but lists can also store names for its elements.

Це дещо збиває з пантелику, але списки також можуть містити назви для своїх елементів.

Named lists | Іменовані списки

The following list is a named list:

Даний список - це іменований список:

In []:

```
movie <- list(name = "Toy Story",  
             year = 1995,  
             genre = c("Animation", "Adventure", "Comedy"))
```

Let me explain that: the list **movie** has some named objects within it. **name**, for example, is an object of type **character**, **year** is an object of type **number**, and **genre** is a vector with objects of type **character**.

Now take a look at this list. This time, it's full of information and well organized. It's clear what each element means. You can see that the elements have different types, and that's ok because it's a list.

Дозвольте мені пояснити, що в списку **movie** є деякі іменовані об'єкти всередині нього. **name**, наприклад, є об'єктом типу **character(текст)**, **year** є об'єктом типу **number(число)**, а **genre** - це вектор з об'єктами типу **character(текст)**

Тепер подивіться на цей список. Цього разу він містить інформацію та добре організований. Тепер зрозуміло, що означає кожен елемент. Як бачите, елементи мають різні типи, і це ок, тому що це список.

In []:

```
movie
```

You can also get separated information from the list. You can use **listName\$selectorName**. The *dollar-sign operator* **\$** will give you the block of data that is related to selectorName.

Let's get the genre part of our movies list, for example.

Ви також можете отримати часткову інформацію зі списку. Ви можете використовувати **listName\$selectorName**. Оператор знак долара **\$** дасть вам блок даних, пов'язаний з selectorName.

Давайте отримаємо, наприклад, частину нашого списку фільмів про жанри.

In []:

```
movie$genre
```

Another way of selecting the genre column:

Інший спосіб витягнути колонку жанрів:

In []:

```
movie["genre"]
```

You can also use numerical selectors like an array. Here we are selecting elements from 2 to 3.

Ви також можете використовувати числові селектори, як і в масивах. Тут ми відбираємо елементи від 2 до 3.

In []:

```
movie[2:3]
```

The function **class()** returns the type of a object. You can use that function to retrieve the type of specific elements of a list:

Функція **class()** показує нам тип об'єкта. Ви можете використовувати цю функцію, щоб отримати тип конкретних елементів списку:

In []:

```
class(movie$name)  
class(movie$foreign)
```

Adding, modifying, and removing items | Додавання, модифікація і видалення елементів

Adding a new element is also very easy. The code below adds a new field named **age** and puts the numerical value 5 into it. In this case we use the double square brackets operator, because we are directly referencing a list member (and we want to change its content).

Додати новий елемент у список також дуже легко. Наведений нижче код додає нове поле з назвою **age** і вводить в нього числове значення 5. У цьому випадку ми використовуємо оператор подвійних квадратних дужок, оскільки ми безпосередньо посилаємося на частину списку (і ми хочемо змінити її вміст).

In []:

```
movie[["age"]] <- 5  
movie
```

In order to modify, you just need to reference a list member that already exists, then change its content.

Для того, щоб змінити частину списку, потрібно просто послатися на ту частину, яка вже існує, а потім змінити її вміст.

In []:

```
movie[["age"]] <- 6
# Now it's 6, not 5
movie
```

And removing is also easy! You just put **NULL**, which means missing value/data, into it.

Видаляти елементи також просто! Треба лиш зазначити **NULL**, що означає відсутність даних.

In []:

```
movie[["age"]] <- NULL
movie
```

Concatenating lists | Зчеплення списків

Concatenation is the process of putting things together, in sequence. And yes, you can do it with lists. Just call the function **c()**. Take a look at the next example:

Зчеплення - це процес послідовного об'єднання речей. І так, ви можете зробити це зі списками. Просто виконайте функцію **c()**. Погляньте на наступний приклад:

In []:

```
# We split our previous list in two sublists
movie_part1 <- list(name = "Toy Story")
movie_part2 <- list(year = 1995, genre = c("Animation", "Adventure", "Comedy"))

# Now we call the function c() to put everything together again
movie_concatenated <- c(movie_part1, movie_part2)

# Check it out
movie_concatenated
```

Lists are really handy for organizing different types of elements in R, and also easy to use. Additionally, lists are also important since this type of data structure is essential to create data frames, our next covered topic.

Списки дійсно зручні для організації елементів різних типів у R, а також дуже прості у використанні. Крім того, списки також важливі, оскільки цей тип структури даних є базовим для створення фреймів даних, яким присвячена наступна тема.

Data Frames | Фрейми даних (Датафрейми)

Data frame is a structure that is used for storing data tables. Underneath it all, a data frame is a list of vectors of same length, exactly like a table (each vector is a column). We call a function called **data.frame()** to create a

data frame and pass vector, which are our columns, as arguments. It is required to name the columns that will compose the data frame.

Фрейм даних - це структура, яка використовується для зберігання таблиць даних. Перш за все, датафрейм - це список векторів такої ж довжини як таблиця (кожен вектор - це одна її колонка). Ми застосовуємо функцію **data.frame()** для створення фрейму даних з аргументом у вигляді вектору, що визначає назви його колонок. Потрібно дати назви колонкам, які разом становлять фрейм даних.

In []:

```
movies <- data.frame(name = c("Toy Story", "Akira", "The Breakfast Club", "The Artist",  
                             "Modern Times", "Fight Club", "City of God", "The Untouchable",  
                             year = c(1995, 1998, 1985, 2011, 1936, 1999, 2002, 1987),  
                             stringsAsFactors=F)
```

Let's print its content of our recently created data frame:

Давайте роздрукуємо вміст нашого нещодавно створеного датафрейму:

In []:

```
movies
```

It's very easy! You can note how it looks like a table.

We can also use the "\$" selector to get some type of information. This operator returns the content of a specific column of a data frame (that's why we have to choose a name for each column).

Це дуже просто! Ви можете помітити, що фрейм виглядає як таблиця.

Ми також можемо використовувати селектор "\$" для отримання певного типу інформації. Цей оператор витягує вміст певної колонки датафрейму (саме тому ми повинні вибрати назву для кожної колонки).

In []:

```
movies$name
```

You retrieve data using numeric indexing, like in lists:

Ви отримуєте дані за допомогою цифрової індексації, як у списках:

In []:

```
# This returns the first (1st) column  
movies[1]
```

The function called **str()** is one of most useful functions in R. With this function you can obtain textual information about an object. In this case, it delivers information about the objects within a data frame. Let's see what it returns:

Функція **str ()** є однією з найбільш корисних функцій у R. За допомогою цієї функції ви можете отримати текстову інформацію про об'єкт. У цьому випадку він передає інформацію про об'єкти, що знаходяться в датафреймі. Давайте подивимося, що він витягне:

In []:

```
str(movies)
```

It shows this data frame has 8 observations, for 2 columns, so called **name** and **year**. The "name" column is a factor with 8 levels and "year" is a numerical column.

У цьому датафреймі є 8 спостережень, для 2 колонок, які мають назви **name** і **year**. Колонка "name" є фактором із 8 рівнями, а "year" - це числова колонка.

The class() function works for data frames as well. You can use it to determine the type of a column of a data frame.

Функція class() також працює для датафреймів. Ви можете використовувати її для визначення типу кожної колонки датафрейму.

In []:

```
class(movies$year)
```

You can use numerical selectors to reach information inside the table.

Ви можете використовувати числові селектори, щоб витягнути потрібну інформацію з таблиці.

In []:

```
movies[1,2] #1-Toy Story, 2-1995
```

The **head()** function is very useful when you have a large table and you need to take a peek at the first elements. This function returns the first 6 values of a data frame (or even a list).

Функція **head()** дуже корисна, коли у вас є велика таблиця, і вам потрібно побачити перші елементи. Ця функція повертає перші 6 рядків датафрейму (або спостережень списку).

In []:

```
head(movies)
```

Similar to the previous function, **tail()** returns the last 6 values of a data frame or list.

Подібно до попередньої функції, **tail()** повертає останні 6 рядків датафрейму або списку даних.

In []:

```
tail(movies)
```

Now let's try to add a new column to our data frame with the length of each movie in minutes.

Тепер давайте спробуємо додати новий стовпець до нашого датафрейму з тривалістю кожного фільму в хвилинах.

In []:

```
movies['length'] <- c(81, 125, 97, 100, 87, 139, 130, 119)
movies
```

A new column was included into our data frame with just one line of code. We just needed to add a vector to data frame, then it will be our new column.

Новий стовпець був включений у наш датафрейм лише одним рядком коду. Нам було просто потрібно додати ще один вектор до датафрейму, який став нашою новою колонкою.

Now let's try to add a new movie to our data set.

Тепер спробуємо додати новий фільм до набору даних.

In []:

```
movies <- rbind(movies, c(name="Dr. Strangelove", year=1964, length=94))
movies
```

REMEMBER!!!! You can't add a list with more variables than the data frame, and vice-versa.

ПАМ'ЯТАЙТЕ!!!! Ви не можете додати список з більшою кількістю змінних(колонок), ніж має датафрейм, і навпаки.

We don't need this movie anymore, let's delete it. Here we are deleting row 12

Якщо нам якийсь фільм більше не потрібен, давайте його видалимо. Так, наприклад, ми видаляємо рядок 12

In []:

```
movies <- movies[-12,]  
movies  
tail(movies)
```

To delete a column you can just set it as **NULL**.

Щоб видалити колонку, ви можете просто присвоїти їй значення **NULL**.

In []:

```
movies[["length"]] <- NULL  
movies
```

That is it! You learned a lot about data frames and how easy it is to work with them.

Це все! Ви багато чого навчилися про фрейми даних і про те, як з ними можна легко працювати.

Scaling R with big data | Масштабування R з використанням великих даних

As you learn more about R, if you are interested in exploring platforms that can help you run analyses at scale, you might want to sign up for a free account on [IBM Data Science Experience \(http://cocl.us/dsx_rp0101en\)](http://cocl.us/dsx_rp0101en), which allows you to run analyses in R with two Spark executors for free.

About the Author: | Про автора:

Hi! It's [Thiago Felipe Correa Borges \(https://www.linkedin.com/in/thiago-felipe-corr%C3%AAa-borges-a932bb114?trk=nav_responsive_tab_profile\)](https://www.linkedin.com/in/thiago-felipe-corr%C3%AAa-borges-a932bb114?trk=nav_responsive_tab_profile), the author of this notebook. I hope you found R easy to learn! There's lots more to learn about R but you're well on your way. Feel free to connect with me if you have any questions.

Привіт! Це [Thiago Felipe Correa Borges \(https://www.linkedin.com/in/thiago-felipe-corr%C3%AAa-borges-a932bb114?trk=nav_responsive_tab_profile\)](https://www.linkedin.com/in/thiago-felipe-corr%C3%AAa-borges-a932bb114?trk=nav_responsive_tab_profile), автор цього зошиту. Я сподіваюся, що ви побачили наскільки легко вивчати R! Є ще багато чого дізнатись про R, але ви вже рухаєтесь у вірному напрямку. Якщо у вас виникли запитання, можете без проблем зв'язатися зі мною.

About the Translator: | Про перекладача:

Hello! It's [Roman Kornyliuk \(https://www.linkedin.com/in/kornelli\)](https://www.linkedin.com/in/kornelli), the translator of this notebook in Ukrainian. Feel free to connect with me if you have any questions too.

Привіт! Мене звуть Roman Kornyliuk (<https://www.linkedin.com/in/kornelli>), перекладач цього зошиту на українську мову. Без вагань пишіть мені також, якщо у вас виникли запитання з приводу R.

Copyright © 2018 Roman Kornyliuk (<https://www.linkedin.com/in/kornelli>).

Copyright © IBM Cognitive Class (<https://cognitiveclass.ai>). This notebook and its source code are released under the terms of the MIT License (<https://cognitiveclass.ai/mit-license/>).