



(<https://www.bigdatauniversity.com>)

VECTORS AND FACTORS | ВЕКТОРИ І ФАКТОРИ

Welcome! | Вітаємо!

By the end of this notebook, you will have learned about **vectors and factors**, two very important data types in R. /

До кінця цього зошита ви дізнаєтесь про **вектори і фактори**, два дуже важливі типи даних в R.

Table of Contents | Зміст

- [About the Dataset](#) | [Про датасет](#)
- [Vectors](#) | [Вектори](#)
- [Vector Operations](#) | [Векторні операції](#)
- [Subsetting Vectors](#) | [Поділ векторів](#)
- [Factors](#) | [Фактори](#)

Estimated Time Needed: **25 min** Очікувана тривалість: **25 хв**

About the Dataset | Про датасет

You have received many movie recommendations from your friends and compiled all of the recommendations into a table, with information about each movie.

This table has one row for each movie and several columns.

Ви отримали багато рекомендацій про фільми від своїх друзів і внесли усю інформацію до таблиці з інформацією про кожен фільм.

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

- **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

Here's what the data looks like:

Ось так виглядають дані:

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

Remember: To run the grey code cells in this exercise, click on the code cell, and then press Shift + Enter.

Лайфхак: Щоб запустити зелену ячейку з кодом нижче, клікніть по ній і натисніть Shift + Enter.

Vectors | Вектори

Vectors are strings of numbers, characters or logical data (one-dimension array). In other words, a vector is a simple tool to store your grouped data.

In R, you create a vector with the combine function `c()`. You place the vector elements separated by a comma between the brackets. Vectors will be very useful in the future as they allow you to apply operations on a series of data easily.

Note that the items in a vector must be of the same class, for example all should be either number, character, or logical.

Вектори - це ряди чисел, символів або логічних даних (одновимірний масив). Іншими словами, вектор - це простий інструмент для зберігання ваших згрупованих даних.

В мові R, ви можете створити вектор за допомогою функції `c()`. Просто розмістіть елементи вектора, відокремлені комою, між круглими дужками. Вектори стануть нам дуже корисними згодом, так як вони дають змогу дуже просто застосовувати операції для рядів даних.

Зауважте, що елементи у векторі повинні бути одного класу, наприклад, всі повинні бути числами, символами або логічними даними.

Numeric, Character and Logical Vectors | Числові, Символьні і Логічні Вектори

Let's say we have four movie release dates (1985, 1999, 2015, 1964) and we want to assign them to a single variable, `release_year`. This means we'll need to create a vector using `c()`.

Using numbers, this becomes a **numeric vector**.

Скажімо, ми маємо чотири дати релізів фільмів (1985, 1999, 2015, 1964), і ми хочемо присвоїти їм одну змінну, `release_year`. Це означає, що нам потрібно буде створити вектор, використовуючи `c()`.

Так як тут використовуються номери, він стає **числовим вектором**.

In []:

```
release_year <- c(1985, 1999, 2015, 1964)
```

In []:

```
release_year
```

What if we use quotation marks? Then this becomes a **character vector**.

Якщо ж ми застосуємо лапки, то вектор стане **символьним вектором**.

In []:

```
# Create genre vector and assign values to it
titles <- c("Toy Story", "Akira", "The Breakfast Club")
titles
```

There are also **logical vectors**, which consist of TRUE's and FALSE's. They're particular important when you want to check its contents

Існують також **логічні вектори**, які складаються з вірних (TRUE) і невірних (FALSE) елементів. Вони особливо важливі, якщо ви хочете перевірити його вміст

In []:

```
titles == "Akira" # which item in `titles` is equal to "Akira"?
```

[Tip] TRUE and FALSE in R

Did you know? R only recognizes TRUE, FALSE, T and F as special values for true and false. That means all other spellings, including *True* and *true*, are not interpreted by R as logical values.

[Tip] TRUE і FALSE в мові R

Чи знаєте ви, що R розрізняє лише TRUE, FALSE, T і F як спеціальні значення вірного і невірного? Це означає, що всі інші написання, включаючи *True* and *true*, не інтерпретуються R як логічні значення.

Vector Operations | Векторні операції

Adding more elements to a vector

You can add more elements to a vector with the same `c()` function you use the create vectors:

In []:

```
release_year <- c(1985, 1999, 2015, 1964)
release_year
```

In []:

```
release_year <- c(release_year, 2016:2018)
release_year
```

Length of a vector | Довжина вектора

How do we check how many items there are in a vector? We can use the **length()** function:

Як перевірити скільки елементів має вектор? Ми можемо скористатися функцією **length()**:

In []:

```
release_year
length(release_year)
```

Head and Tail of a vector | "Голова" та "Хвіст" вектора

We can also retrieve just the **first few items** using the **head()** function:

Ми можемо також отримати лише **кілька перших елементів** вектора, використовуючи функцію **head()**:

In []:

```
head(release_year) #first six items
```

In []:

```
head(release_year, n = 2) #first n items
```

In []:

```
head(release_year, 2)
```

We can also retrieve just the **last few items** using the **tail()** function:

Ми можемо також отримати лише **кілька останніх елементів** вектора за допомогою функції **tail()**:

In []:

```
tail(release_year) #last six items
```

In []:

```
tail(release_year, 2) #last two items
```

Sorting a vector | Сортювання вектора

We can also sort a vector:

Ми можемо також сортувати наш вектор:

In []:

```
sort(release_year)
```

We can also **sort in decreasing order**:

Можна **сортувати у порядку спадання**:

In []:

```
sort(release_year, decreasing = TRUE)
```

But if you just want the minimum and maximum values of a vector, you can use the `min()` and `max()` functions

Але якщо ви просто хочете отримати мінімальне і максимальне значення вектора, можна використовувати функції `min()` та `max()`

In []:

```
min(release_year)
max(release_year)
```

Average of Numbers | Середнє арифметичне

If you want to check the average cost of movies produced in 2014, what would you do? Of course, one way is to add all the numbers together, then divide by the number of movies:

Якщо ви хочете розрахувати середню вартість фільмів, створених у 2014 році, що б ви зробили? Звичайно, один із способів - це додати всі числа разом, а потім розділити суму на кількість фільмів:

In []:

```
cost_2014 <- c(8.6, 8.5, 8.1)

# sum results in the sum of all elements in the vector
avg_cost_2014 <- sum(cost_2014)/3
avg_cost_2014
```

You also can use the **mean** function to find the average of the numeric values in a vector:

Також ви можете використовувати функцію **mean**, щоб знайти середнє арифметичне числових значень у векторі:

In []:

```
mean_cost_2014 <- mean(cost_2014)
mean_cost_2014
```

Giving Names to Values in a Vector | Як присвоїти назви значенням вектора?

Suppose you want to remember which year corresponds to which movie.

With vectors, you can give names to the elements of a vector using the **names()** function:

Припустімо, ви хочете пам'ятати, який рік відповідає якому фільму.

За допомогою векторів ви можете присвоїти назви елементам вектора за допомогою функції **names()**:

In []:

```
#Creating a year vector
release_year <- c(1985, 1999, 2010, 2002)

#Assigning names
names(release_year) <- c("The Breakfast Club", "American Beauty", "Black Swan", "Chicago")

release_year
```

Now, you can retrieve the values based on the names:

Тепер ви можете отримати значення за заданими іменами:

In []:

```
release_year[c("American Beauty", "Chicago")]
```

Note that the values of the vector are still the years. We can see this in action by adding a number to the first item:

Зауважте, що значення вектора все ще є роками. Ми можемо бачити це в дії, додавши номер до першого елемента:

In []:

```
release_year[1] + 100 #adding 100 to the first item changes the year
```

And you can retrieve the names of the vector using **names()**

І ви можете отримати назви елементів вектора, використовуючи **names()**

In []:

```
names(release_year)[1:3]
```

Summarizing Vectors | Підсумовування Векторів

You can also use the **"summary"** function for simple descriptive statistics: minimum, first quartile, mean, third quartile, maximum:

Ви також можете використовувати функцію **"summary"** для простої описової статистики: мінімум, перший кuartиль, середнє, третій кuartиль, максимум:

In []:

```
summary(cost_2014)
```

Using Logical Operations on Vectors | Логічні Операції з Векторами

A vector can also be comprised of **TRUE** and **FALSE**, which are special **logical values** in R. These boolean values are used to indicate whether a condition is true or false.

Let's check whether a movie year of 1997 is older than (**greater in value than**) 2000.

Вектор може також складатися з **TRUE** та **FALSE**, які є спеціальними логічними операторами в R. Ці логічні оператори використовуються для позначення того, чи є умова істинною або невірною.

Давайте перевіримо, чи є 1997 рік більшим за 2000. (а фільм 1997, відповідно, новішим за 2000)

In []:

```
movie_year <- 1997  
movie_year > 2000
```

You can also make a logical comparison across multiple items in a vector. Which movie release years here are "greater" than 2014?

Ви також можете зробити логічне порівняння кількох елементів у векторі. Які роки випуску фільму тут "більші"/"свіжіші", ніж 2014 рік?

In []:

```
movies_years <- c(1998, 2010, 2016)  
movies_years > 2014
```

We can also check for **equivalence**, using **==**. Let's check which movie year is equal to 2015.

Ми також можемо звірити **еквівалентність**, використовуючи `==`. Давайте перевіримо, який рік фільму дорівнює 2015 року.

In []:

```
movies_years == 2015 # is equal to 2015?
```

If you want to check which ones are **not equal** to 2015, you can use `!=`

Якщо ви хочете перевірити, які з них **не рівні** 2015 року, ви можете використовувати `!=`

In []:

```
movies_years != 2015
```

[Tip] Logical Operators in R

You can do a variety of logical operations in R including:

- Checking equivalence: `1 == 2`
- Checking non-equivalence: `TRUE != FALSE`
- Greater than: `100 > 1`
- Greater than or equal to: `100 >= 1`
- Less than: `1 < 2`
- Less than or equal to: `1 <= 2`

[Лайф-хак] Логічні оператори в R

Ви можете виконувати різні логічні операції в R, включаючи:

- Перевірити рівність змінних: `1 == 2`
- Перевірити на не-рівність: `TRUE != FALSE`
- Більше: `100 > 1`
- Більше або дорівнює: `100 >= 1`
- Менше: `1 < 2`
- Менше або дорівнює: `1 <= 2`

Subsetting Vectors | Поділ векторів

What if you wanted to retrieve the second year from the following **vector of movie years**?

Що робити, якщо ви хочете отримати другий рік з даного **вектору років фільмів**?

In []:

```
movie_years <- c(1985, 1999, 2002, 2010, 2012)
movie_years
```

To retrieve the **second year**, you can use square brackets []:

Щоб отримати **другий рік**, ви можете використовувати квадратні дужки []:

In []:

```
movie_years[2] #second item
```

To retrieve the **third year**, you can use:

Щоб отримати **третій рік** у векторі, ви можете скористатися:

In []:

```
movie_years[3]
```

And if you want to retrieve **multiple items**, you can pass in a vector:

Якщо ви хочете отримати **кілька елементів**, ви можете дати таку команду:

In []:

```
movie_years[c(1,3)] #first and third items
```

Retrieving a vector without some of its items | Вивід вектора без деяких його елементів

To retrieve a vector without an item, you can use negative indexing. For example, the following returns a vector slice **without the first item**.

Щоб отримати вектор без елемента, можна використовувати негативну індексацію. Наприклад, наступна команда повертає частину вектора **без першого елемента**.

In []:

```
titles <- c("Black Swan", "Jumanji", "City of God", "Toy Story", "Casino")
titles[-1]
```

You can save the new vector using a variable:

Ви можете зберегти новий вектор за допомогою змінної:

In []:

```
new_titles <- titles[-1] #removes "Black Swan", the first item
new_titles
```

Missing Values (NA) | Відсутні значення (NA)

Sometimes values in a vector are missing and you have to show them using NA, which is a special value in R for "Not Available". For example, if you don't know the age restriction for some movies, you can use NA.

Іноді значення у векторі відсутні, і ви повинні показати їх за допомогою NA, що є особливим значенням у R для відображення «Немає даних». Наприклад, якщо ви не знаєте обмеження за віком для деяких фільмів, ви можете використовувати NA.

In []:

```
age_restric <- c(14, 12, 10, NA, 18, NA)
age_restric
```

[Tip] Checking NA in R

You can check if a value is NA by using the **is.na()** function, which returns TRUE or FALSE.

- Check if NA: **is.na(NA)**
- Check if not NA: **!is.na(2)**

[Tip] Перевірка на наявність NA в мові R

Ви можете перевірити, чи є наявні NA у векторі, використовуючи функцію **is.na()**, яка повертає значення TRUE або FALSE.

- Перевірити чи є NA: **is.na(NA)**
- Перевірити чи нема NA: **!is.na(2)**

Subsetting vectors based on a logical condition | Поділ векторів на основі логічної умови

What if we want to know which movies were created after year 2000? We can simply apply a logical comparison across all the items in a vector:

Що робити, якщо ми хочемо знати, які фільми були створені після 2000 року? Ми можемо просто застосувати логічне порівняння всіх елементів у векторі:

In []:

```
release_year > 2000
```

To retrieve the actual movie years after year 2000, you can simply subset the vector using the logical vector within **square brackets "[]"**:

Щоб отримати перелік років виходу фільмів після 2000 року, ви можете просто поділити вектор, використовуючи логічний вектор у **квадратних дужках "[]"** :

In []:

```
release_year[movie_years > 2000] #returns a vector for elements that returned TRUE for the
```

As you may notice, subsetting vectors in R works by retrieving items that were TRUE for the provided condition. For example, `year[year > 2000]` can be verbally explained as: *"From the vector year, return only values where the values are TRUE for year > 2000"*.

You can even manually write out TRUE or T for the values you want to subset:

Як ви помітили, поділ векторів в R працює шляхом витягування саме тих елементів, які виявились істинними (TRUE) для заданої нами умови. Наприклад, `year[year > 2000]` можна усно пояснити так: *"З вектору year повертайте тільки ті значення, які є істинними (TRUE) для year > 2000"*.

Ви можете навіть вручну вписати значення TRUE або T для значень, які потрібно витягнути в окрему підмножину:

In []:

```
release_year  
release_year[c(T, F, F, F)] #returns the values that are TRUE
```

Factors | Фактори

Factors are variables in R which take on a limited number of different values; such variables are often referred to as **categorical variables**. The difference between a categorical variable and a continuous variable is that a categorical variable can belong to a limited number of categories. A continuous variable, on the other hand, can correspond to an infinite number of values. For example, the height of a tree is a continuous variable, but the titles of books would be a categorical variable.

One of the most important uses of factors is in statistical modeling; since categorical variables enter into statistical models differently than continuous variables, storing data as factors insures that the modeling functions will treat such data correctly.

Фактори - це змінні в R, які приймають обмежену кількість різних значень; такі змінні часто називають **категоріальними змінними**. Різниця між категоріальною змінною та неперервною змінною полягає в тому, що категоріальна змінна може належати до обмеженої кількості категорій. З іншого боку, неперервна змінна може відповідати нескінченній кількості значень. Наприклад, висота дерева є неперервною змінною, але назви книг будуть категоріальною (або дискретною) змінною.

Одним з найважливіших застосувань факторів є статистичне моделювання; оскільки категоріальні змінні входять в статистичні моделі інакше, ніж неперервні змінні, зберігаючи їх в вигляді **факторів** ви забезпечите правильну обробку цих даних функціями моделювання.

Let's start with a **vector** of genres:

Розпочнемо з **вектору** жанрів:

In []:

```
genre_vector <- c("Comedy", "Animation", "Crime", "Comedy", "Animation")
genre_vector
```

As you may have noticed, you can theoretically group the items above into three categories of genres: *Animation*, *Comedy* and *Crime*. In R-terms, we call these categories "**factor levels**".

The function **factor()** converts a vector into a factor, and creates a factor level for each unique element.

Як ви помітили, ви теоретично можете групувати елементи за трьома категоріями жанрів: *Animation*, *Comedy* і *Crime*. У термінології R ми називаємо ці категорії "**факторні рівні**".

Функція **factor()** перетворює вектор у фактор, і створює "factor level" для кожного унікального елемента.

In []:

```
genre_factor <- factor(genre_vector)
genre_factor
```

In []:

```
levels(genre_factor) # factor levels
```

Summarizing Factors | Підсумовування Факторів

When you have a large vector, it becomes difficult to identify which levels are most common (e.g., "How many 'Comedy' movies are there?").

To answer this, we can use **summary()**, which produces a **frequency table**, as a named vector.

Якщо у вас є великий вектор, стає важко визначити, які рівні є найбільш поширеними (наприклад, "скільки" комедійних фільмів міститься в векторі?).

Щоб відповісти на це питання, ми можемо використовувати функцію **summary()**, яка видає **частотну таблицю**, у вигляді іменованого вектора.

In []:

```
summary(genre_factor)
```

And recall that you can sort the values of the table using **sort()**.

І нагадаємо, що ви можете сортувати значення таблиці за допомогою **sort()**.

In []:

```
sort(summary(genre_factor)) #sorts values by ascending order
```

Ordered factors | Впорядковані фактори

There are two types of categorical variables: a **nominal categorical variable** and an **ordinal categorical variable**.

A **nominal variable** is a categorical variable for names, without an implied order. This means that it is impossible to say that 'one is better or larger than the other'. For example, consider **movie genre** with the categories *Comedy, Animation, Crime, Comedy, Animation*. Here, there is no implicit order of low-to-high or high-to-low between the categories.

In contrast, **ordinal variables** do have a natural ordering. Consider for example, **movie length** with the categories: *Very short, Short, Medium, Long, Very long*. Here it is obvious that *Medium* stands above *Short*, and *Long* stands above *Medium*.

Існує два типи дискретних змінних: **номінальна категоріальна/дискретна змінна** і **порядкова категоріальна/дискретна змінна**.

номінальна змінна є дискретною змінною для назв без заданого порядку. Це означає, що неможливо сказати, що "одна категорія краща або більша за іншу". Наприклад, розглянемо **жанр фільму (movie genre)** з категоріями *Comedy, Animation, Crime, Comedy, Animation*. Тут відсутній будь-який внутрішній порядок від низького до високого чи високого до низького між категоріями.

Натомість **порядкові змінні** мають природний порядок. Розглянемо, наприклад, змінну **тривалість фільму (movie length)** з категоріями: *Дуже Короткий, Короткий, Середній, Довгий, Дуже Довгий* (*Very short, Short, Medium, Long, Very long*). Тут очевидно, що *Medium* знаходиться вище *Short*, а *Long* знаходиться вище *Medium*.

In []:

```
movie_length <- c("Very Short", "Short", "Medium","Short", "Long",  
                 "Very Short", "Very Long")  
movie_length
```

movie_length should be converted to an ordinal factor since its categories have a natural ordering. By default, the function **factor()** transforms **movie_length** into an unordered factor.

To create an **ordered factor**, you have to add two additional arguments: **ordered** and **levels**.

- **ordered**: When set to **TRUE** in **factor()**, you indicate that the factor is ordered.
- **levels**: In this argument in **factor()**, you give the values of the factor in the correct order.

movie_length слід перетворити на порядковий фактор, оскільки його категорії мають природний порядок. За замовчуванням функція **factor()** перетворює **movie_length** у неупорядкований (номінальний) фактор.

Щоб створити **порядковий фактор**, вам слід додати ще два додаткові аргументи: **ordered** та **levels**.

- **ordered:** Якщо зазначити TRUE у рамках функції `factor()`, ви вкажете, що дана змінна є порядковим фактором.
- **levels:** у цьому аргументі функції `factor()` ви можете задати значення змінної у правильному порядку.

In []:

```
movie_length_ordered <- factor(movie_length, ordered = TRUE ,
                               levels = c("Very Short" , "Short" , "Medium",
                                           "Long", "Very Long"))
movie_length_ordered
```

Now, lets look at the summary of the ordered factor, **factor_mvlength_vector**:

Тепер, давайте розглянемо підсумок нашого порядкового фактора, **factor_mvlength_vector**:

In []:

```
summary(movie_length_ordered)
```

About the Author: | Про автора:

Hi! It's Helly Patel (<https://ca.linkedin.com/in/helly-patel-90344750>), 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 Helly Patel (<https://ca.linkedin.com/in/helly-patel-90344750>), 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>), 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 © 2016 Big Data University (https://bigdatauniversity.com/?utm_source=bducopyrightlink&utm_medium=dswb&utm_campaign=bdu).

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

This notebook and its source code are released under the terms of the MIT License (<https://bigdatauniversity.com/mit-license/>).

