МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет менеджменту

Кафедра економічної кібернетики, комп'ютерних наук та інформаційних технологій

ШТУЧНИЙ ІНТЕЛЕКТ ТА МАШИННЕ НАВЧАННЯ

методичні рекомендації для практичних занять та самостійної роботи здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Комп'ютерні науки» спеціальності F3(122) «Комп'ютерні науки» денної форми здобуття вищої освіти

МИКОЛАЇВ

2025

УДК 004.8
Ш94

Друкується за рішенням науково-методичної комісії факультету менеджменту Миколаївського національного аграрного університету від 27 березня 2025 року, протокол № 7.

Укладачі:

О.В.Шебаніна — д-р екон. наук, професор, декан факультету менеджменту Миколаївського національного аграрного університету;

С. І. Тищенко — к.п.н., доцент, доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О. Ю. Пархоменко — к.ф.-м.н., доцент, доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О.О. Жебко — асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій, Миколаївського національного аграрного університету;

А.М.Коломієць — асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій, Миколаївського національного аграрного університету;

Рецензенти:

О. С. Садовий - канд. техн. наук, доцент, завідувач кафедри агроінженерії Миколаївського національного аграрного університету

Ю. В. Грицук - канд. техн. наук, доцент кафедри загальної інженерної підготовки Донбаської національної академії будівництва і архітектури

Штучний інтелект та машинне навчання : методичні рекомендації для Ш94 практичних занять та самостійної роботи здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Комп'ютерні науки» спеціальності F3(122) «Комп'ютерні науки» денної форми здобуття вищої освіти / уклад. О. В. Шебаніна, С. І. Тищенко, О. Ю. Пархоменко, О.О. Жебко, А. М. Коломієць. Миколаїв : МНАУ, 2025. 148 с.

УДК 004.8

ЗМІСТ

МЕТА, ЗАВДАННЯ КУРСУ, ВИМОГИ ДО ОСНОВНИХ ЗНАНЬ ЗДОБУВАЧІВ ВИЩОЇ ОСВІТИ

Дисципліна «Штучний інтелект» вивчається здобувачами вищої освіти спеціальності 122 «Комп'ютерні науки» на третьому курсі і є обов'язковою компонентою.

Покликана сформувати у здобувачів необхідний обсяг теоретичних знань та практичних навичок зі штучного інтелекту, навчити їх створювати моделі та розуміти методи навчання моделей, визначати методи обробки даних за допомогою навчання без нагляду та навчання з підкріпленням, ознайомити з використанням сучасної обчислювальної техніки і пакетів для роботи зі штучним інтелектом.

В процесі проходження курсу здобувачі навчаться створювати та оцінювати результат навчання різноманітних моделей навчання під наглядом за допомогою сучасних інструментів обчислювальної техніки створювати і навчати відповідні моделі як самостійно, так і за допомогою пакетів.

Мета дисципліни: сформувати у здобувачів вищої освіти необхідний обсяг теоретичних і практичних знань про навчання під наглядом, навчання без нагляду та навчання з підкріпленням, методи оцінки ефективності обраних моделей та алгоритмів.

Завдання дисципліни: оволодіння здобувачами вищої освіти знаннями і навичками для побудови моделей навчання, штучних нейронних мереж, вибору оптимального алгоритму для навчання без нагляду, визначення методів для роботи з навчанням з підкріпленням, оцінювання отриманого результату.

Предмет дисципліни: методи та засоби штучного інтелекту.

Кожна лабораторна робота містить методичні поради та хід порядку її виконання.

ЛАБОРАТОРНА РОБОТА 1-2

Тема: Введення в мову R. Базові типи даних та конструкції.

Приклад виконання роботи:

Для виконання роботи перейдіть за посиланням https://cloud.r-project.org/ та завантажте R. Встановлення підключення відображено на рис. 1.

Рисунок 1 – Завантаження R

Встановіть R у потрібне місце (рис. 2).

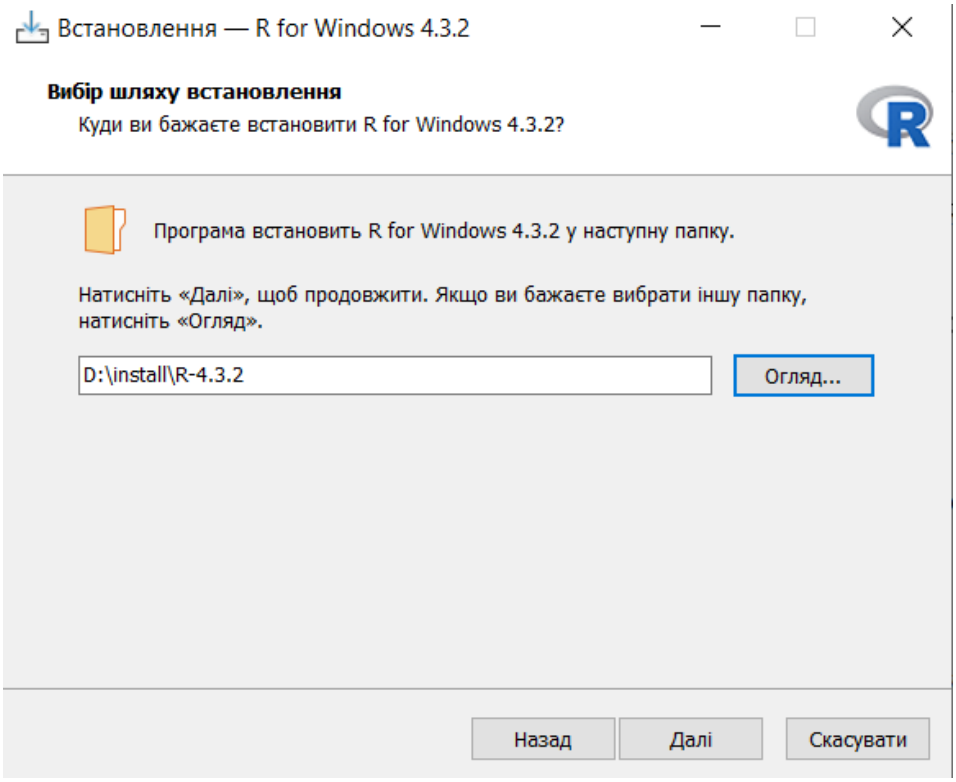


Рисунок 2 – Встановлення

В результаті отримуємо ярлик на робочому столі, що при натисканні запускає консоль (рис. 3).

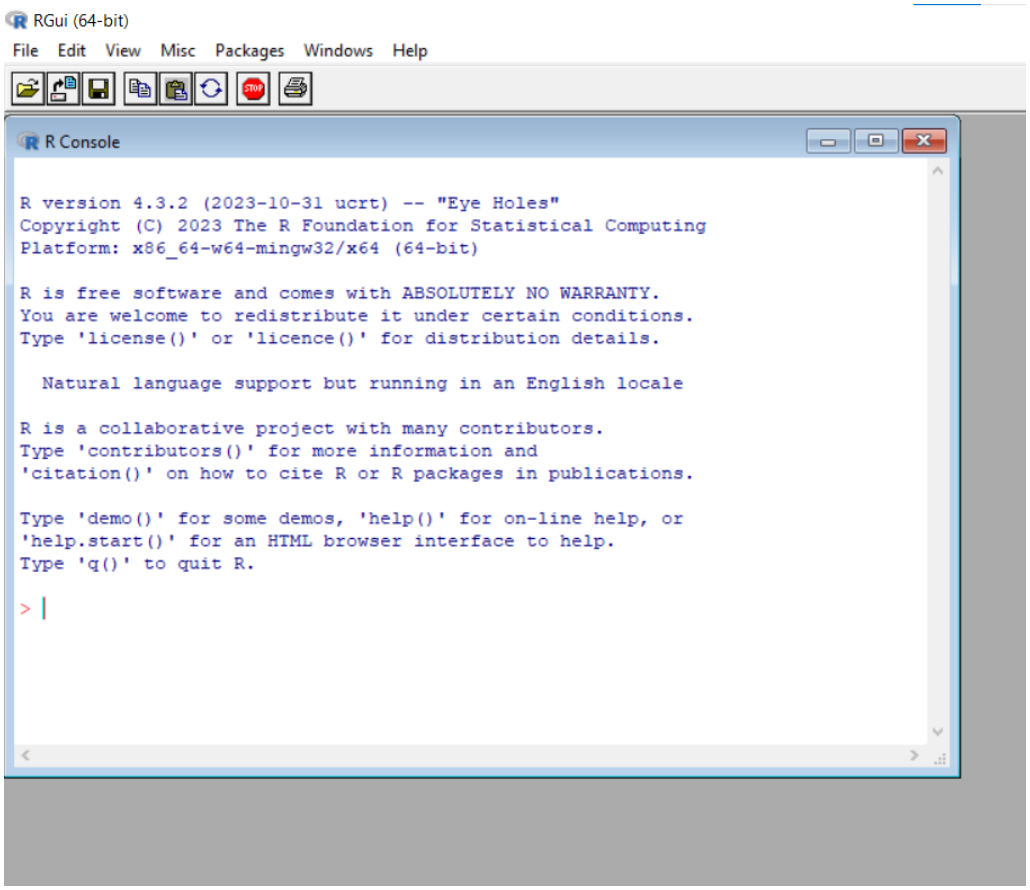


Рисунок 3 – Базова консоль після встановлення

При бажанні, консоль можна замінити на прийнятне IDE за посиланням https://posit.co/products/open-source/rstudio/. В подальшому для прикладів буде використовуватися саме консольна версія, але ніяких обмежень для виконання робіт нема.

Одразу переходимо до синтаксису. Звичайні типи даних (рядок, символ, числа чи операції над ними) можна виводити одразу в консоль без створення змінних. Рядкові змінні мають бути загорнуті в лапки (рис. 4).

```
> u
Error: object 'u' not found
> "u"
[1] "u"
> 5
[1] 5
> 5 + 5
[1] 10
```

Рисунок 4 – Приклад синтаксису R на базових типах даних

Функція print() також існує, але її частіше використовують у функціях, що загорнуті в {} (рис. 5). Без print() виводитись число не буде.

```
> for (x in 1:10) {
+   print(x)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

Рисунок 5 – Приклад функції з використанням print()

Коментарі виводяться за допомогою решітки # перед кожним рядком коментаря (нема багаторядкового коментування). У R немає команди для оголошення змінної. Змінна створюється, коли ви вперше присвоюєте їй значення. Щоб присвоїти значення змінній, використовуйте знак <-. Щоб вивести (або надрукувати) значення змінної, просто введіть її назву:

```
> name <- "John"
> age <- 40
> name
[1] "John"
> age
[1] 40
```

Рисунок 6 – Приклад використання змінних

Ви також можете об'єднати два або більше елементів за допомогою функції paste().

Для об'єднання тексту і змінної у R використовується кома (,):

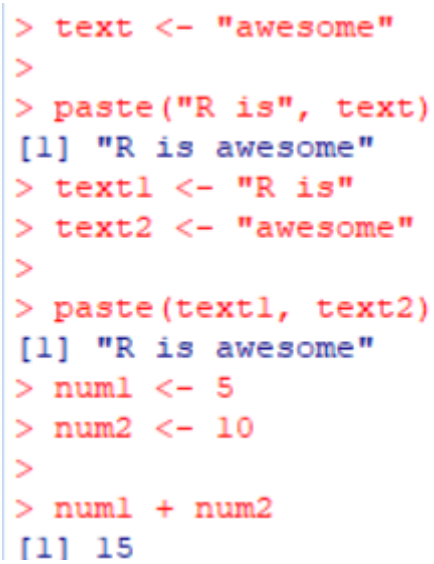


Рисунок 7 – Використання функції paste()

Змінна може мати коротку назву (наприклад, x та y) або більш описову назву (age, carname, total_volume). Правила для змінних R такі:

- Ім'я змінної повинно починатися з літери і може бути комбінацією літер, цифр, крапки(.)
- та підкреслення(_). Якщо ім'я починається з крапки(.), після неї не може йти цифра.
- Ім'я змінної не може починатися з цифри або символу підкреслення (_)
- Назви змінних чутливі до регістру (вік, вік і AGE - це три різні змінні)
- Зарезервовані слова не можна використовувати як змінні (TRUE, FALSE, NULL, if...).

У програмуванні тип даних є важливим поняттям.

Змінні можуть зберігати дані різних типів, а різні типи можуть виконувати різні функції.

У R змінні не обов'язково оголошувати з певним типом, вони навіть можуть змінювати тип після того, як їх було задано.

Основні типи даних у R можна розділити на наступні типи:

- числові - (10.5, 55, 787)

- цілі - (1L, 55L, 100L, де літера "L" позначає, що це ціле число)
- комплексний - (9 + 3i, де "i" - уявна частина)
- символьний (так званий рядок) - ("k", "R is exciting", "FALSE", "11.5")
- логічний (так званий булевий) - (TRUE або FALSE)

Ми можемо використовувати функцію class() для перевірки типу даних змінної:

```
> x <- 10.5
> class(x)
[1] "numeric"
> x <- TRUE
> class(x)
[1] "logical"
```

Рисунок 8 – Перевірка типу даних

Ви можете перетворити один тип в інший за допомогою наступних функцій:

as.numeric()

as.integer()

as.complex()

```
> x <- 1L
> y <- 2
> a <- as.numeric(x)
> b <- as.integer(y)
> x
[1] 1
> y
[1] 2
> class(a)
[1] "numeric"
> class(b)
[1] "integer"
```

Рисунок 9 – Перетворення числових типів

R має багато вбудованих математичних функцій, які дозволяють виконувати математичні задачі над числами.

Наприклад, функції min() і max() можна використовувати для знаходження найменшого або найбільшого числа у множині.

Функція sqrt() повертає квадратний корінь з числа.

Функція abs() повертає абсолютне (додатне) значення числа.

Функція ceil() округлює число в більшу сторону до найближчого цілого, а функція floor() округлює число в меншу сторону до найближчого цілого і повертає результат.

Ви можете присвоїти багаторядковий рядок такій змінній:

```
> str <- "Lorem ipsum dolor sit amet,
+ consectetur adipiscing elit,
+ sed do eiusmod tempor incididunt
+ ut labore et dolore magna aliqua."
> str
[1] "Lorem ipsum dolor sit amet,\nconsecte
```

Рисунок 10 – Спроба виведення багаторядкового тексту

Однак зауважте, що R додаватиме "\n" у кінці кожного розриву рядка. Це називається символ переходу на новий рядок, а символ n вказує на новий рядок.

Якщо ви хочете, щоб розриви рядків було вставлено у тій самій позиції, що і у коді, скористайтеся функцією cat().

У мові R існує багато корисних функцій для роботи з рядками.

Наприклад, щоб знайти кількість символів у рядку, скористайтеся функцією nchar().

Для перевірки наявності символу або послідовності символів у рядку використовуйте функцію grepl().

Для маніпулювання даними використовуються різні оператори:

Оператор	Опис	Приклад
:	Створює ряд чисел у послідовності	x <- 1:10
%in%	З'ясувати, чи належить елемент вектору	x %in% y
%*%	Матричне множення	x <- Matrix1 %*% Matrix2

Оператор if

Оператор if записується з ключовим словом if і використовується для вказівки блоку коду, який виконується, якщо умова має значення TRUE:

```
> a <- 33
> b <- 200
>
> if (b > a) {
+   print("b is greater than a")
+ }
[1] "b is greater than a"
```

Рисунок 11 – Приклад роботи оператора if

Ключове слово else if - це спосіб R сказати "якщо попередні умови не були істинними, то спробувати цю умову".

Ключове слово else перехоплює все, що не перехоплено попередніми умовами:

```
> a <- 200
> b <- 33
>
> if (b > a) {
+   print("b is greater than a")
+ } else if (a == b) {
+   print("a and b are equal")
+ } else {
+   print("a is greater than b")
+ }
[1] "a is greater than b"
```

Рисунок 12 – Комбінована конструкція

Цикли While

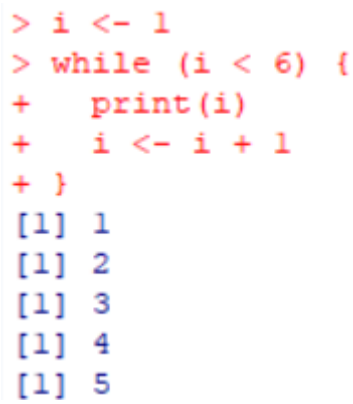


Рисунок 13 – Приклад циклу while

За допомогою оператора break ми можемо зупинити цикл, навіть якщо умова while має значення TRUE:

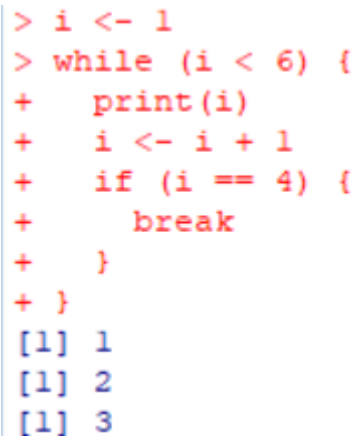


Рисунок 14 – Приклад використання оператора break

За допомогою оператора next ми можемо пропустити ітерацію, не завершуючи цикл:

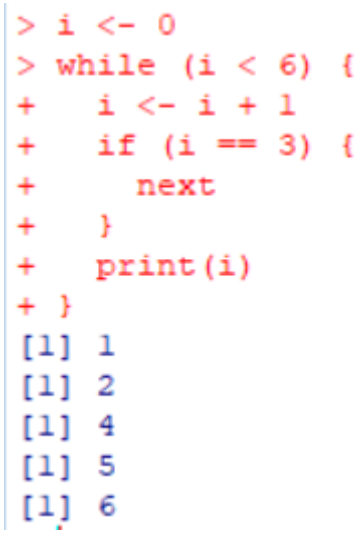


Рисунок 15 – Приклад використання оператора next

Цикли For

Цикл for використовується для перебору послідовності:

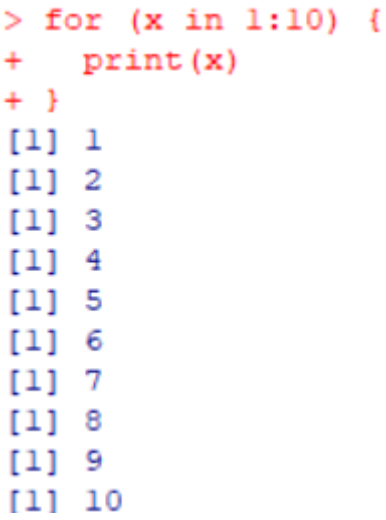


Рисунок 16 – Приклад використання циклу for

Це менше схоже на ключове слово for в інших мовах програмування, і працює більше як метод ітератора, як в інших об'єктно-орієнтованих мовах програмування.

За допомогою циклу for ми можемо виконати набір операторів, по одному разу для кожного елемента у векторі, масиві, списку тощо.

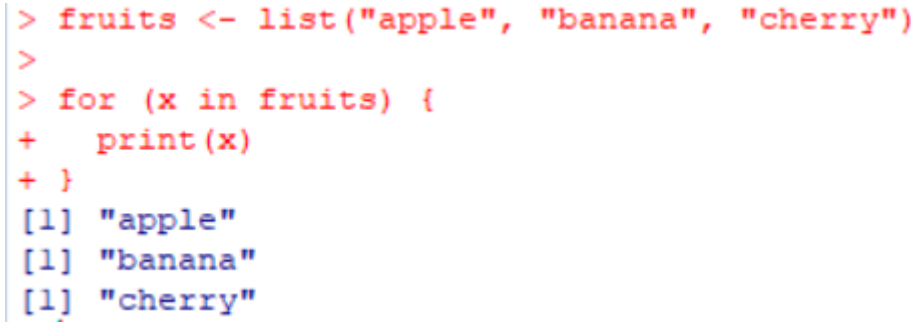


Рисунок 17 – Використання циклу для списку

R Функції

Щоб створити функцію, використовуйте ключове слово function().Щоб викликати функцію, використовуйте ім'я функції з круглими дужками, наприклад, my_function():

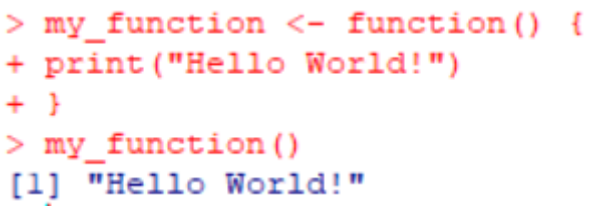


Рисунок 18 – Приклад створення та використання функції

У функції можна передавати інформацію як аргументи.

Аргументи вказуються після імені функції в круглих дужках. Ви можете додавати скільки завгодно аргументів, просто розділяйте їх комою.

За замовчуванням функція повинна викликатися з правильною кількістю аргументів. Це означає, що якщо ваша функція очікує 2 аргументи, ви повинні викликати функцію з 2 аргументами, не більше і не менше:

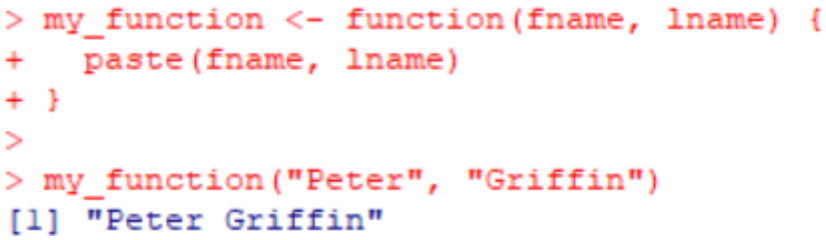


Рисунок 19 – Приклад створення та використання функції с кількома аргументами

Значення параметра за замовчуванням

У наступному прикладі показано, як використовувати значення параметра за замовчуванням.

Якщо ми викликаємо функцію без аргументу, вона використовує значення за замовчуванням:

```
> my_function <- function(country = "Norway") {
+   paste("I am from", country)
+ }
>
> my_function("Sweden")
[1] "I am from Sweden"
> my_function()
[1] "I am from Norway"
```

Рисунок 20 – Використання параметру за замовчуванням

Щоб дозволити функції повертати результат, використовуйте функцію return():

```
> my_function <- function(x) {
+   return (5 * x)
+ }
>
> print(my_function(3))
[1] 15
```

Рисунок 21 – Повернення значень у функції

Зазвичай, коли ви створюєте змінну всередині функції, ця змінна є локальною і може бути використана тільки всередині цієї функції.

Щоб створити глобальну змінну всередині функції, можна скористатися оператором глобального присвоювання <<-.

Також використовуйте оператор глобального присвоювання, якщо ви хочете змінити глобальну змінну всередині функції.

ЛАБОРАТОРНА РОБОТА 3-4

Тема: Введення в мову R. Структури даних. Графіки.

Приклад виконання роботи:

СТРУКТУРИ ДАНИХ

Вектор - це просто список елементів одного типу.

Щоб об'єднати список елементів у вектор, використовуйте функцію c() і розділяйте елементи комою.

У прикладі нижче ми створюємо векторну змінну з назвою fruits, яка об'єднує рядки:

```
> fruits <- c("banana", "apple", "orange")
> fruits
[1] "banana" "apple"  "orange"
```

Рисунок 1 – Вектор рядків

У цьому прикладі ми створюємо вектор, який поєднує числові значення:

```
> numbers <- c(1, 2, 3)
>
> numbers
[1] 1 2 3
```

Рисунок 2 – Вектор чисел

Щоб створити вектор з числовими значеннями в послідовності, використовуйте оператор :

```
> numbers <- 1:10
>
> numbers
 [1]  1  2  3  4  5  6  7  8  9 10
```

Рисунок 3 – Вектор чисел через оператор :

Щоб дізнатися, скільки елементів має вектор, використовуйте функцію length().

Щоб відсортувати елементи вектора за алфавітом або числом, використовуйте функцію sort().

Ви можете отримати доступ до елементів вектора, звертаючись до його індексу в дужках []. Перший елемент має індекс 1, другий елемент має індекс 2 і так далі.

Ви також можете отримати доступ до декількох елементів, посилаючись на різні позиції індексів за допомогою функції c():

```
> fruits <- c("banana", "apple", "orange", "mango", "lemon")
> fruits[c(1, 3)]
[1] "banana" "orange"
```

Рисунок 4 – Досутп до елементів вектор за позиціями

Ви також можете використовувати від'ємні індекси для доступу до всіх елементів, окрім вказаних. Щоб змінити значення конкретної позиції, зверніться до її індексного номера:

```
> fruits[1] <- "pear"
> fruits
[1] "pear"   "apple"  "orange" "mango"  "lemon"
```

Рисунок 5 – Зміна елементу вектора

Щоб повторити вектори, використовуйте функцію rep():

```
> repeat_indepent <- rep(c(1,2,3), times = c(5,2,1))
> repeat_indepent
[1] 1 1 1 1 1 2 2 3
> repeat_each <- rep(c(1,2,3), each = 3)
> repeat_each
[1] 1 1 1 2 2 2 3 3 3
```

Рисунок 6 – Повторення елементів вектору

Щоб зробити більші або менші кроки в послідовності, використовуйте функцію seq(). Функція seq() має три параметри: від - місце, де починається послідовність, до - місце, де закінчується послідовність, і по - інтервал послідовності.

Список у R може містити багато різних типів даних. Список – це колекція даних, яка є впорядкованою та змінною.

Щоб створити список, використовуйте функцію list():

```
> thislist <- list("apple", "banana", "cherry")
> thislist
[[1]]
[1] "apple"

[[2]]
[1] "banana"

[[3]]
[1] "cherry"
```

Рисунок 7 – Створення списку

Ви можете отримати доступ до елементів списку, посилаючись на його індекс, вказаний у дужках. Перший пункт має індекс 1, другий - 2 і так далі. Щоб змінити значення певного елемента, зверніться до його індексу. Щоб дізнатися, скільки елементів містить список, використовуйте функцію length(). Щоб дізнатися, чи присутній заданий елемент у списку, використовуйте оператор %in%. Щоб додати елемент в кінець списку, використовуйте функцію append(). Щоб додати елемент праворуч від заданого індексу, додайте у функції append() вираз "after=номер індексу":

```
> thislist <- list("apple", "banana", "cherry")
>
> append(thislist, "orange", after = 2)
[[1]]
[1] "apple"

[[2]]
[1] "banana"

[[3]]
[1] "orange"

[[4]]
[1] "cherry"
```

Рисунок 8 – Додавання елементу у список

Ви також можете видаляти елементи списку. У наступному прикладі створено новий, оновлений список без елемента "яблуко":

```
> thislist <- list("apple", "banana", "cherry")
>
> newlist <- thislist[-1]
> newlist
[[1]]
[1] "banana"

[[2]]
[1] "cherry"
```

Рисунок 9 – Видалення елементу зі списку

Ви можете вказати діапазон індексів, вказавши, де починається і де закінчується діапазон, за допомогою оператора :

```
> thislist <- list("apple", "banana", "cherry", "orange", "kiwi", "melon", "man$
>
> (thislist)[2:5]
[[1]]
[1] "banana"

[[2]]
[1] "cherry"

[[3]]
[1] "orange"

[[4]]
[1] "kiwi"
```

Рисунок 10 – Діапазон списку

Існує декілька способів об'єднання двох або більше списків у R.

Найпоширенішим способом є використання функції c(), яка об'єднує два елементи разом:

```
> list1 <- list("a", "b", "c")
> list2 <- list(1,2,3)
> list3 <- c(list1,list2)
>
> list3
[[1]]
[1] "a"

[[2]]
[1] "b"

[[3]]
[1] "c"

[[4]]
[1] 1

[[5]]
[1] 2

[[6]]
[1] 3
```

Рисунок 11 – Об’єднання списків

Матриця – це двовимірний набір даних зі стовпчиками та рядками.

Стовпчик - це вертикальне представлення даних, а рядок - горизонтальне представлення даних.

Матрицю можна створити за допомогою функції matrix(). Вкажіть параметри nrow і ncol, щоб отримати кількість рядків і стовпців:

```
> thismatrix <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)
> thismatrix
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

Рисунок 12 – Матриця

Ви також можете створити матрицю з рядків. Ви можете отримати доступ до елементів за допомогою дужок [ ]. Перша цифра "1" у дужці вказує на позицію рядка, а друга цифра "2" - на позицію стовпця. Доступ до всього рядка можна отримати, якщо вказати кому після числа в дужці. Доступ до всього стовпця можна отримати, якщо вказати кому перед числом у дужці. Доступ до декількох рядків можна отримати за допомогою функції c():

```
> thismatrix <- matrix(c("apple", "banana", "cherry", "orange","grape", "pineapple", "pear", "melon", "fig"), nrow = 3, ncol = 3)
>
> thismatrix[c(1,2),]
     [,1]     [,2]     [,3]
[1,] "apple"  "orange" "pear"
[2,] "banana" "grape"  "melon"
```

Рисунок 13 – Доступ до кількох рядків

Можна отримати доступ до більш ніж одного стовпця аналогічно. Використовуйте функцію cbind() для додавання додаткових стовпців (функція rbind() для додавання додаткових рядків) до матриці:

```
> thismatrix <- matrix(c("apple", "banana", "cherry", "orange","grape", "pineapple", "pear", "melon", "fig"), nrow = 3, ncol = 3)
> newmatrix <- rbind(thismatrix, c("strawberry", "blueberry", "raspberry"))
> newmatrix
     [,1]         [,2]         [,3]
[1,] "apple"      "orange"     "pear"
[2,] "banana"     "grape"      "melon"
[3,] "cherry"     "pineapple"  "fig"
[4,] "strawberry" "blueberry"  "raspberry"
```

Рисунок 14 – Додавання рядків/стовбців до матриці

Використовуйте функцію c() для видалення рядків і стовпців у матриці:

```
> thismatrix <- matrix(c("apple", "banana", "cherry", "orange", "mango", "pineapple"), nrow = 3, ncol =2)
> thismatrix <- thismatrix[-c(1), -c(1)]
> thismatrix
[1] "mango"     "pineapple"
```

Рисунок 15 – Видалення рядків/стовбців матриці

Щоб дізнатися, чи присутній заданий елемент у матриці, використовуйте оператор %in%. Використовуйте функцію dim(), щоб знайти кількість рядків і стовпців у матриці. Для знаходження розмірності матриці використовується функція length(). Загальна кількість комірок у матриці дорівнює кількості рядків, помноженій на кількість стовпців. Ви можете циклічно переглядати матрицю за допомогою циклу for. Цикл почнеться з першого рядка, рухаючись праворуч:

```
> thismatrix <- matrix(c("apple", "banana", "cherry", "orange"), nrow = 2, ncol = 2)
>
> for (rows in 1:nrow(thismatrix)) {
+   for (columns in 1:ncol(thismatrix)) {
+     print(thismatrix[rows, columns])
+   }
+ }
[1] "apple"
[1] "cherry"
[1] "banana"
[1] "orange"
```

Рисунок 16 – Приклад використання циклу for

Знову ж таки, ви можете використовувати функцію rbind() або cbind() для об'єднання двох або більше матриць разом. На відміну від матриць, масиви можуть мати більше двох вимірів.

Ми можемо використовувати функцію array() для створення масиву, а параметр dim - для вказівки розмірів:

```
> thisarray <- c(1:24)
> thisarray
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
> multiarray <- array(thisarray, dim = c(4, 3, 2))
> multiarray
, , 1

     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12

, , 2

     [,1] [,2] [,3]
[1,]   13   17   21
[2,]   14   18   22
[3,]   15   19   23
[4,]   16   20   24
```

Рисунок 17 – Тривимірний масив

Ви можете отримати доступ до елементів масиву, посилаючись на індексну позицію. Ви можете використовувати дужки [] для доступу до потрібних елементів масиву. Синтаксис наступний: array[позиція рядка, позиція стовпця, рівень матриці]. Ви також можете отримати доступ до цілого рядка або стовпця з матриці у масиві за допомогою функції c():

```
> multiarray[c(1),,1]
[1] 1 5 9
```

Рисунок 18 – Доступ до всіх елементів з першого рядка з першої матриці

Фрейм даних - це дані, що відображаються у форматі таблиці.

Таблиці даних можуть містити різні типи даних. Якщо перший стовпець може бути символьним, то другий і третій можуть бути числовими або логічними. Однак кожен стовпець повинен мати один і той самий тип даних.

Для створення фрейму даних використовуйте функцію data.frame():

```
> Data_Frame <- data.frame (
+   Training = c("Strength", "Stamina", "Other"),
+   Pulse = c(100, 150, 120),
+   Duration = c(60, 30, 45)
+ )
> Data_Frame
  Training Pulse Duration
1 Strength   100       60
2  Stamina   150       30
3    Other   120       45
```

Рисунок 19 – Створення фрейму даних

Використовуйте функцію summary() для підсумовування даних з фрейму даних (рис. 20).

Ми можемо використовувати одинарні дужки [ ], подвійні дужки [[ ]] або $ для доступу до стовпців з фрейму даних:

```
> summary(Data_Frame)
   Training             Pulse          Duration
 Length:3           Min.   :100.0   Min.   :30.0
 Class :character   1st Qu.:110.0   1st Qu.:37.5
 Mode  :character   Median :120.0   Median :45.0
                    Mean   :123.3   Mean   :45.0
                    3rd Qu.:135.0   3rd Qu.:52.5
                    Max.   :150.0   Max.   :60.0
```

```
> Data_Frame[1]
  Training
1 Strength
2  Stamina
3    Other
>
> Data_Frame[["Training"]]
[1] "Strength" "Stamina"  "Other"
>
> Data_Frame$Training
[1] "Strength" "Stamina"  "Other"
```

Рисунок 20 – Використання summary() та доступ до елементів

Використовуйте функцію rbind() для додавання нових рядків до фрейму даних. Використовуйте функцію cbind() для додавання нових стовпців до фрейму даних. Використовуйте функцію c() для видалення рядків і стовпців у

фреймі даних. Використовуйте функцію dim() для знаходження кількості рядків і стовпців у фреймі даних. Ви також можете використовувати функцію ncol() для знаходження кількості стовпців і nrow() для знаходження кількості рядків. Використовуйте функцію length(), щоб знайти кількість стовпців у фреймі даних (подібно до ncol()). Використовуйте функцію rbind(), щоб об'єднати два або більше фреймів даних у R по вертикалі. А функція cbind() - для об'єднання двох або більше фреймів даних у R по горизонталі.

Фактори використовуються для категоризації даних. Прикладами факторів є

Демографія: Чоловіки/жінки

Музика: Рок, поп, класика, джаз

Тренування: Сила, витривалість

Щоб створити фактор, використовуйте функцію factor() і додайте вектор як аргумент:

```
> music_genre <- factor(c("Jazz", "Rock", "Classic", "Classic", "Pop", "Jazz", "Rock", "Jazz"))
> music_genre
[1] Jazz    Rock    Classic Classic Pop     Jazz    Rock    Jazz
Levels: Classic Jazz Pop Rock
```

Рисунок 21 – Створення фактору

З наведеного вище прикладу видно, що фактор має чотири рівні (категорії): Класика, Джаз, Поп та Рок. Щоб вивести лише рівні, скористайтеся функцією levels(). Ви також можете задати рівні, додавши аргумент levels у функцію factor(). Використовуйте функцію length(), щоб дізнатися, скільки елементів у факторі. Щоб отримати доступ до елементів у факторі, зверніться до індексного номера, використовуючи дужки []. Щоб змінити значення конкретного елемента, зверніться до його індексного номера. **Зверніть увагу**, що ви не можете змінити значення певного елемента, якщо він ще не вказаний у факторі. Однак, якщо ви вже вказали його всередині аргументу levels, це спрацює.

ГРАФІКИ

Функція plot() використовується для малювання точок (маркерів) на діаграмі. Функція отримує параметри для задання точок на діаграмі. Параметр 1 задає точки на осі x. Параметр 2 задає точки на осі y.

У найпростішому випадку ви можете використовувати функцію plot() для побудови графіка залежності двох чисел одне від одного. Побудуйте по одній точці на діаграмі в позиції (1) і позиції (3): plot(1, 3).

Щоб намалювати більше точок, використовуйте вектори:

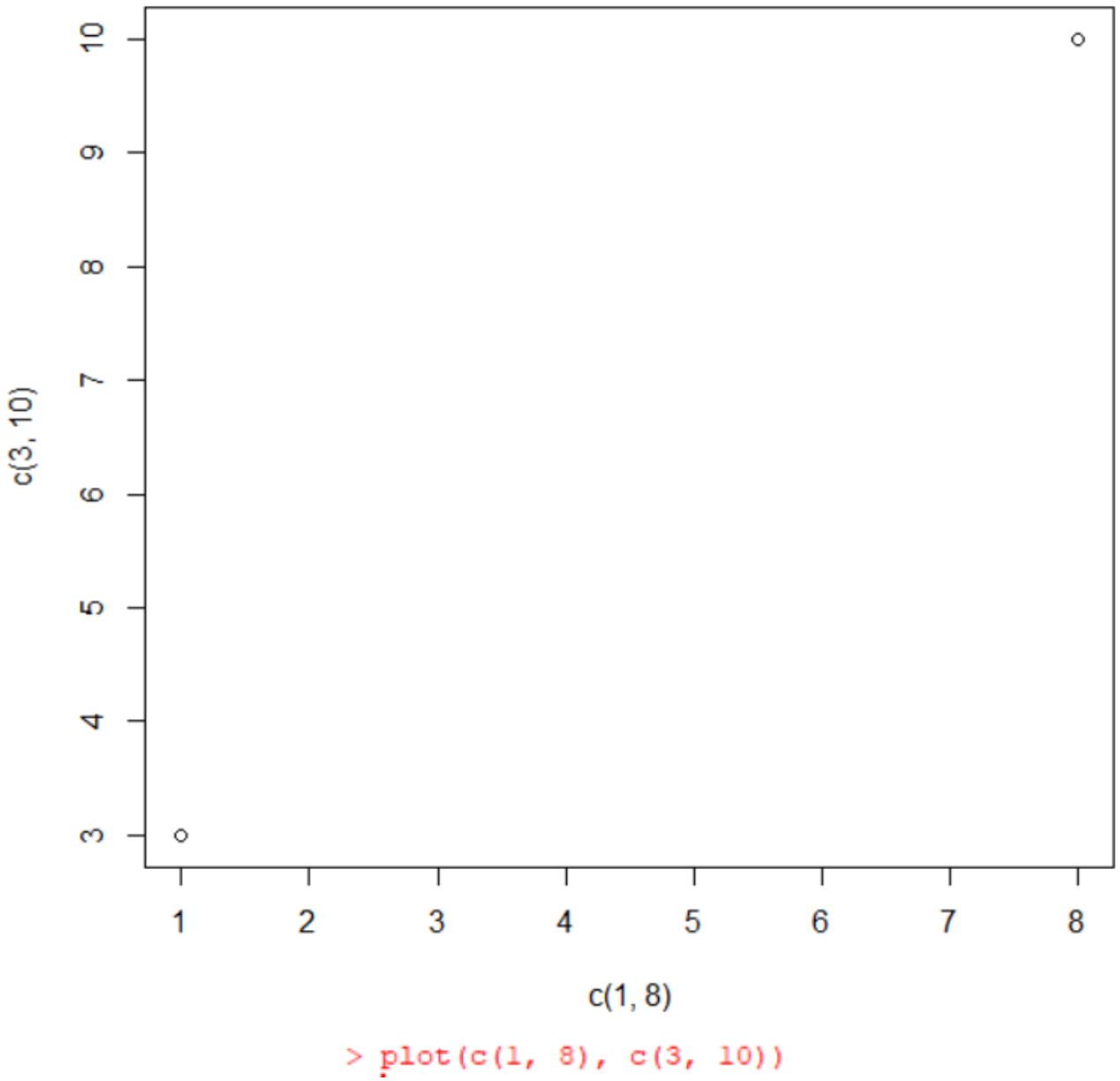


Рисунок 22 – Дві точки

Ви можете побудувати скільки завгодно точок, просто переконайтеся, що у вас є однакова кількість точок на обох осях. Для кращої організації, коли у вас багато значень, краще використовувати змінні:

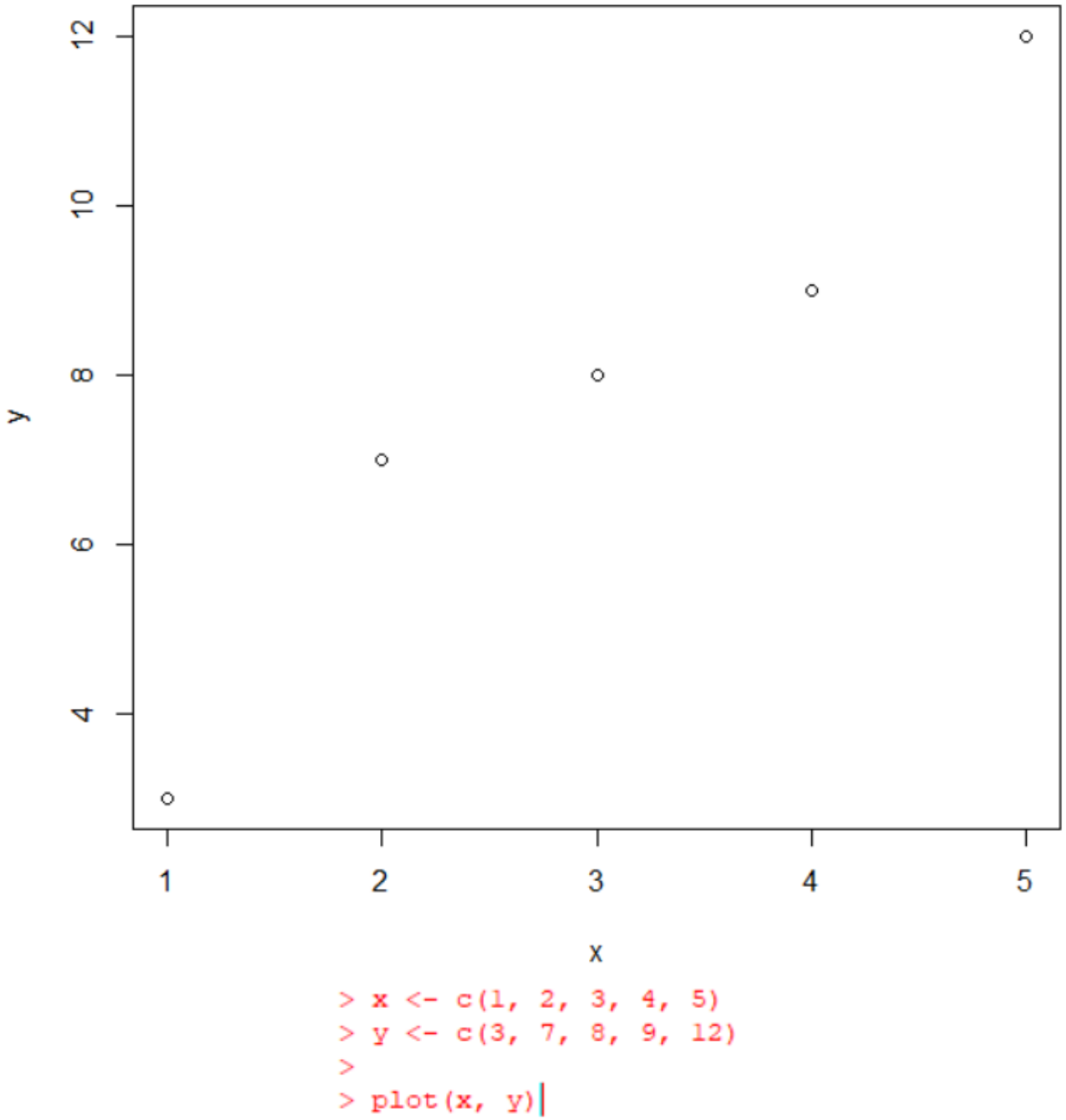


Рисунок 23 – Побудова точок зі змінних

Якщо ви хочете намалювати точки послідовно, як на осі x, так і на осі y, використовуйте оператор :. Функція plot() також приймає параметр типу зі значенням l, щоб намалювати лінію, яка з'єднає всі точки на діаграмі. Функція plot() також приймає інші параметри, такі як main, xlab і ylab, якщо ви хочете

налаштувати діаграму з головним заголовком і різними підписами для осей x і y. Існує багато інших параметрів, за допомогою яких можна змінити вигляд точок.

Використовуйте col="color", щоб додати колір до точок.

Використовуйте cex=number для зміни розміру точок (1 - за замовчуванням, 0.5 означає на 50% менше, а 2 - на 100% більше).

Використовуйте pch зі значенням від 0 до 25 для зміни формату форми точки. Значення параметра pch коливається від 0 до 25, що означає, що ми можемо вибрати до 26 різних типів форм точок:

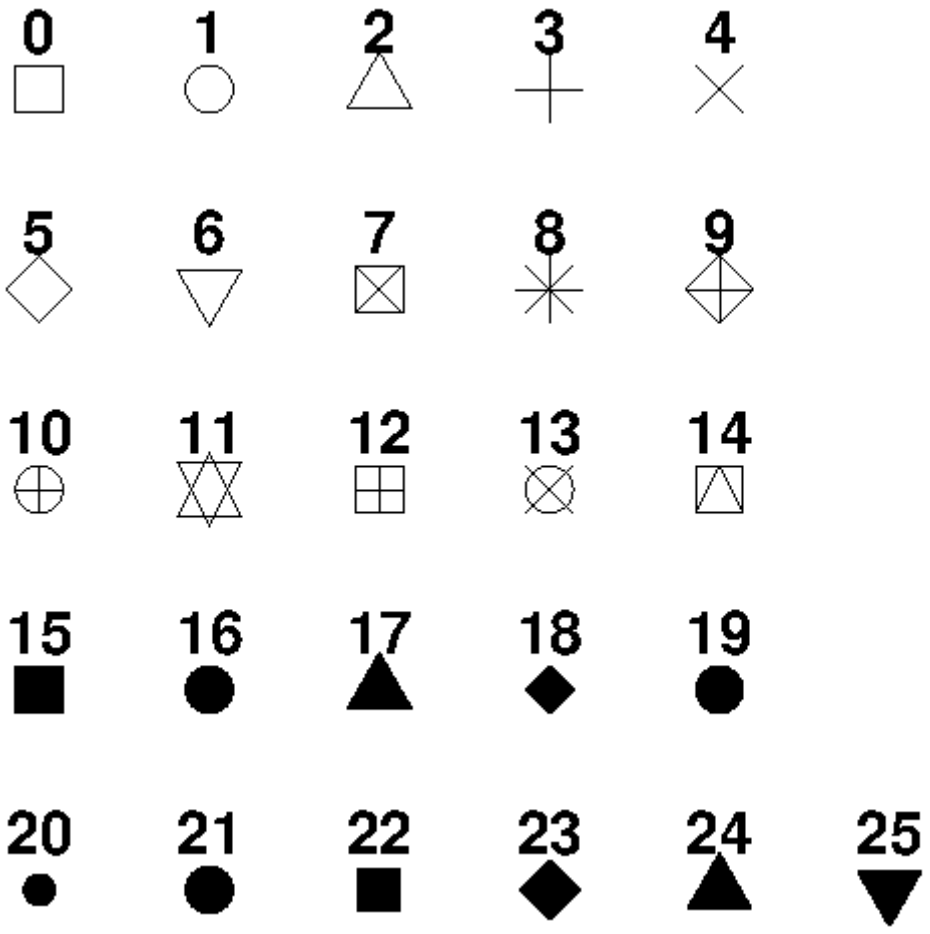


Рисунок 24 – Типи точок

Лінійна діаграма має лінію, яка з'єднує всі точки діаграми.

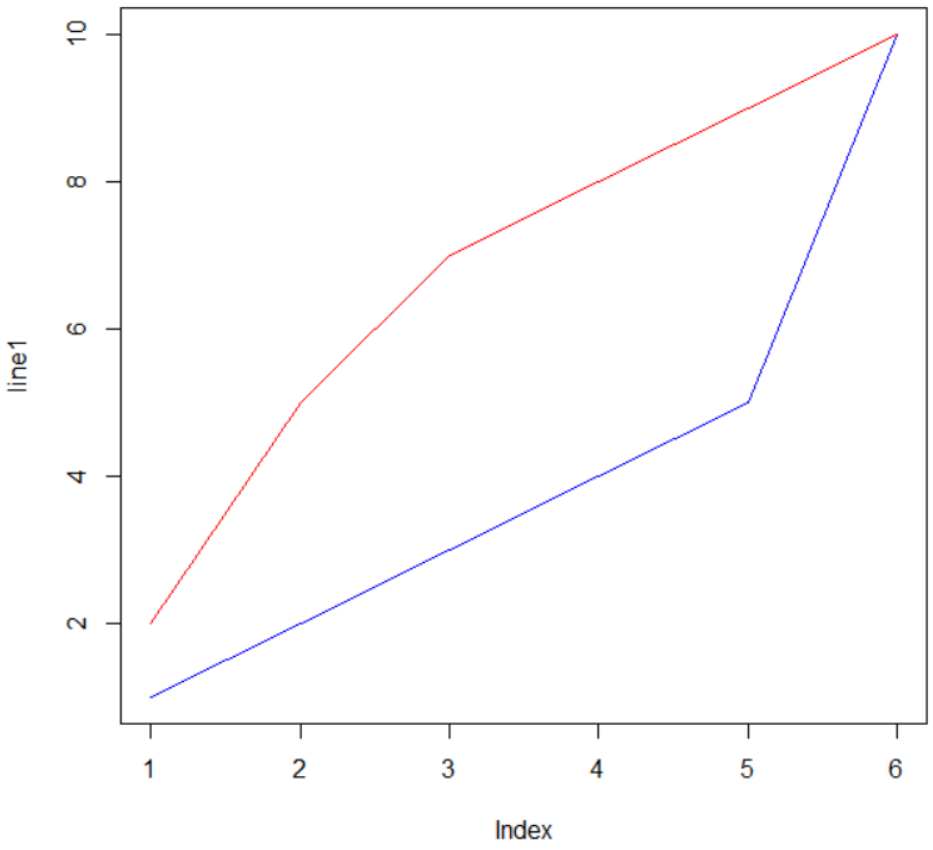
Щоб створити лінію, скористайтеся функцією plot() і додайте параметр type зі значенням "l". Колір лінії за замовчуванням чорний. Щоб змінити колір,

використовуйте параметр col. о Щоб змінити ширину лінії, використовуйте параметр lwd (1 - за замовчуванням, 0.5 означає на 50% менше, а 2 - на 100% більше). За замовчуванням лінія суцільна. Використовуйте параметр lty зі значенням від 0 до 6, щоб вказати формат рядка.

Наприклад, lty=3 відобразить пунктирну лінію замість суцільної. Доступні значення параметра lty:

- 0 вилучає лінію;
- 1 відображає суцільну лінію;
- 2 відображає пунктирну лінію;
- 3 відображає пунктирну лінію;
- 4 виводить "пунктирну" лінію;
- 5 відображає "довгу пунктирну" лінію;
- 6 відображає "дві пунктирні" лінії.

Щоб відобразити більше однієї лінії на графіку, використовуйте функцію plot() разом з функцією lines():



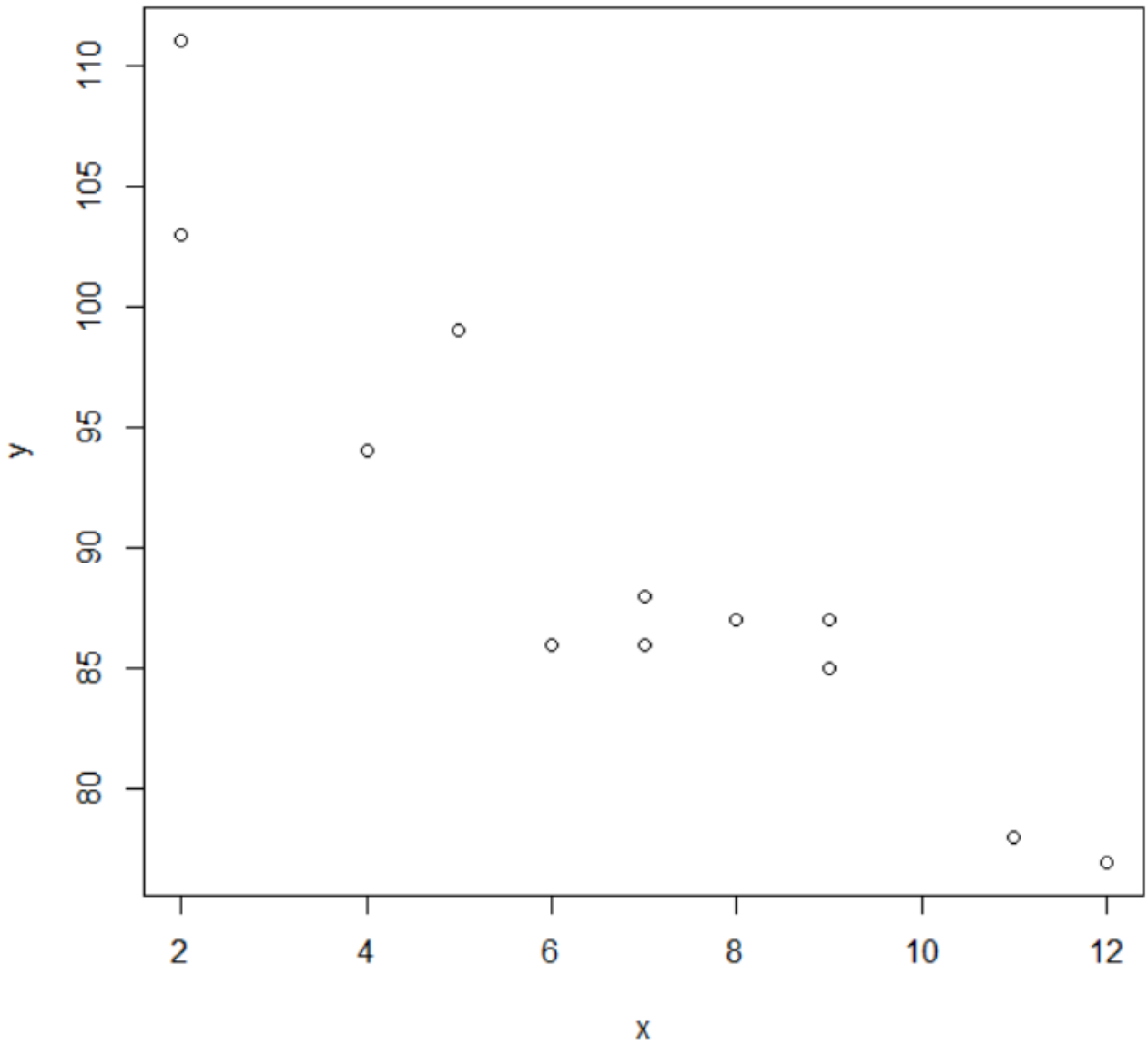
```
> line1 <- c(1,2,3,4,5,10)
> line2 <- c(2,5,7,8,9,10)
>
> plot(line1, type = "l", col = "blue")
> lines(line2, type="l", col = "red")
```

Рисунок 25 – 2 лінії

З розділу "Графік" ви дізналися, що функція plot() використовується для побудови графіків чисел відносно один одного.

"Точкова діаграма (діаграма розсіювання)" - це тип діаграми, який використовується для відображення зв'язку між двома числовими змінними і відображає по одній точці для кожного спостереження.

Вона потребує двох векторів однакової довжини, один для осі x (горизонтальний) і один для осі y (вертикальний):

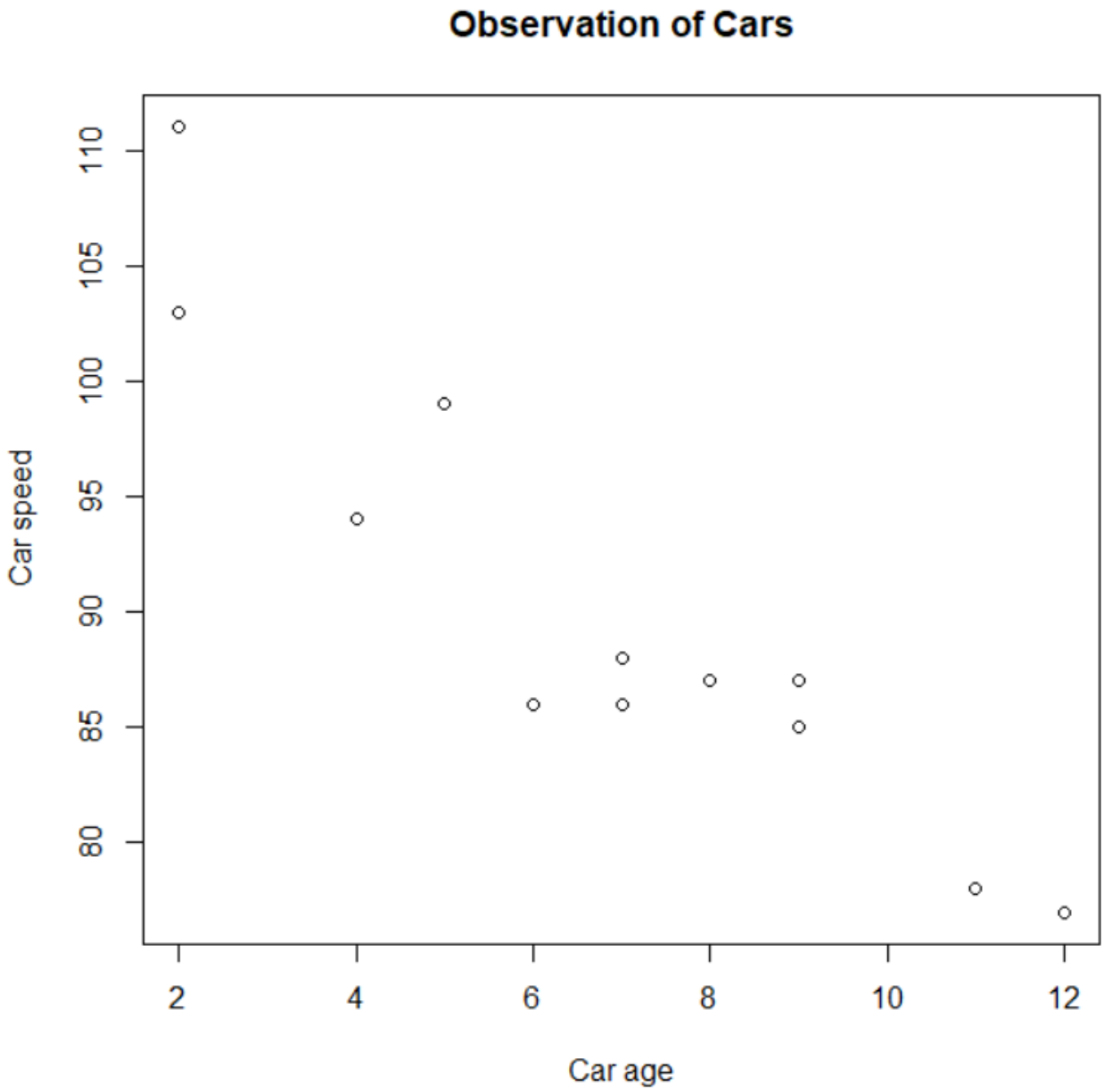



```
> x <- c(5,7,8,7,2,2,9,4,11,12,9,6)
> y <- c(99,86,87,88,111,103,87,94,78,77,85,86)
>
> plot(x, y)
```

Рисунок 26 – Точкова діаграма

Спостереження у наведеному вище прикладі повинно показати результат 12 автомобілів, що проїжджають повз.

Це може бути незрозуміло для тих, хто бачить графік вперше, тому додамо заголовок і різні підписи, щоб краще описати діаграму розсіювання:




```
> x <- c(5,7,8,7,2,2,9,4,11,12,9,6)
> y <- c(99,86,87,88,111,103,87,94,78,77,85,86)
>
> plot(x, y, main="Observation of Cars", xlab="Car age", ylab="Car speed")
```

Рисунок 27 – 12 автомобілів, що проїжджають повз

Нагадаємо, що спостереження у прикладі вище є результатом спостереження за 12 автомобілями, що проїжджають повз.

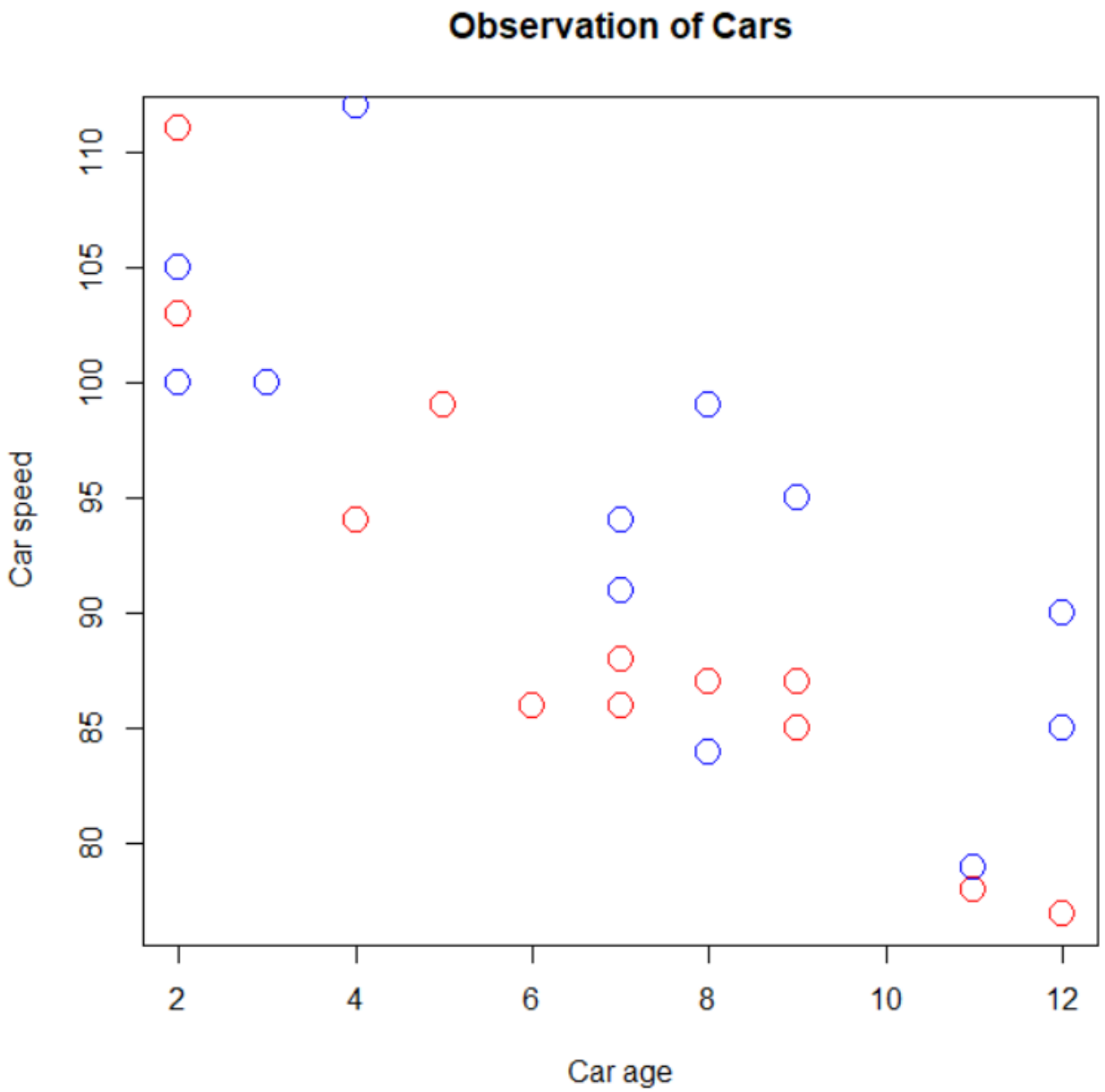
Вісь x показує вік автомобіля. Вісь y показує швидкість автомобіля, коли він проїжджав повз.

Чи існує взаємозв'язок між цими спостереженнями?

Здається, що чим новіший автомобіль, тим швидше він їздить, але це може бути збігом, адже ми зареєстрували лише 12 автомобілів.

У наведеному вище прикладі, здається, існує зв'язок між швидкістю автомобіля та віком, але що, якщо ми побудуємо графік спостережень за інший день? Чи покаже нам діаграма розсіювання щось інше?

Щоб порівняти графік з іншим графіком, скористайтеся функцією points():



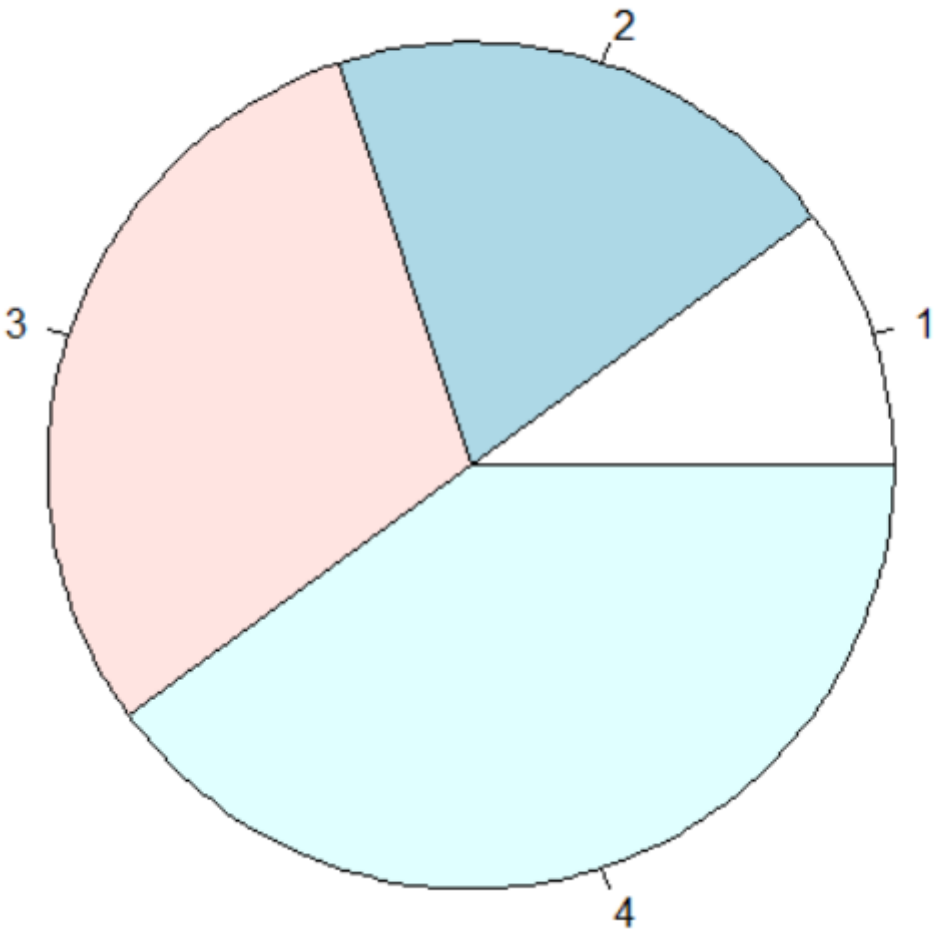

```
> x1 <- c(5,7,8,7,2,2,9,4,11,12,9,6)
> y1 <- c(99,86,87,88,111,103,87,94,78,77,85,86)
> x2 <- c(2,2,8,1,15,8,12,9,7,3,11,4,7,14,12)
> y2 <- c(100,105,84,105,90,99,90,95,94,100,79,112,91,80,85)
> plot(x1, y1, main="Observation of Cars", xlab="Car age", ylab="Car speed", col="red", cex=2)
> points(x2, y2, col="blue", cex=2)
```

Рисунок 28 – Діаграма розсіювання за 2 дні

Щоб мати змогу побачити різницю порівняння, ви повинні присвоїти графікам різні кольори (за допомогою параметра col). Червоним кольором позначено значення дня 1, а синім - дня 2. Зверніть увагу, що ми також додали параметр cex для збільшення розміру точок.

Висновок спостереження: Порівнюючи два графіки, я думаю, що можна з упевненістю сказати, що вони обидва дають нам той самий висновок: чим новіший автомобіль, тим швидше він їздить.

Кругова діаграма - це кругове графічне представлення даних. Використовуйте функцію pie() для побудови секторних діаграм:




```
x <- c(10,20,30,40)
pie(x)
```

Рисунок 29 – Кругова діаграма

Як ви можете бачити, кругова діаграма малює один круг для кожного значення вектора (в даному випадку 10, 20, 30, 40). За замовчуванням, побудова першого круга починається з осі x і рухається проти годинникової стрілки.

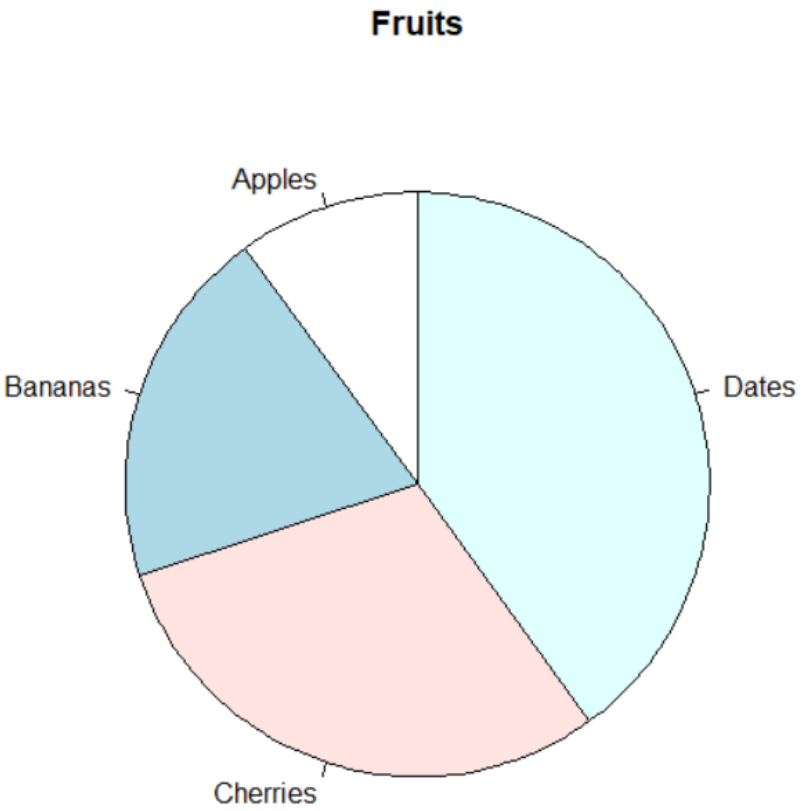
Розмір кожного пирога визначається шляхом порівняння значення з усіма іншими значеннями, за допомогою цієї формули:

Значення, поділене на суму всіх значень: x/sum(x).

Ви можете змінити початковий кут кругової діаграми за допомогою параметра init.angle.

Значення init.angle задається кутом у градусах, де за замовчуванням кут дорівнює 0 (pie(x, init.angle = 90)).

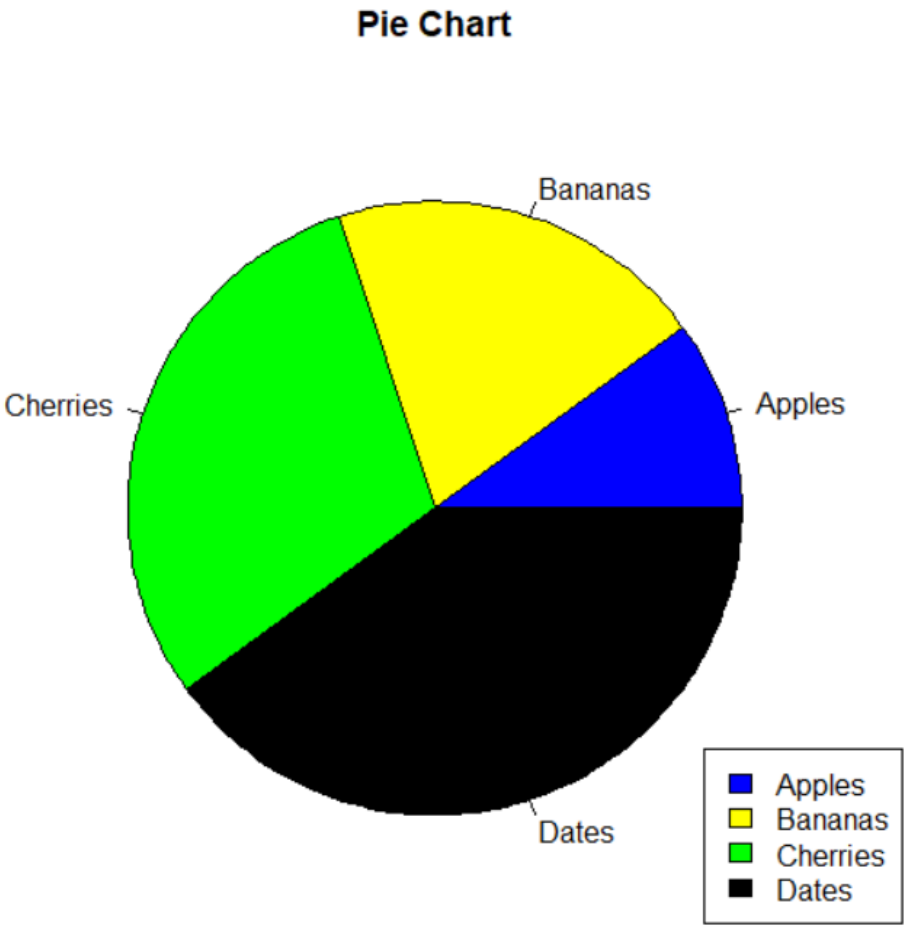
Використовуйте параметр label для додавання підписів до секторної діаграми, а параметр main - для додавання заголовка:




```
mylabel <- c("Apples", "Bananas", "Cherries", "Dates")
pie(x, label = mylabel, main = "Fruits", init.angle = 90)
```

Рисунок 30 – Підписана кругова діаграма з обертом

Ви можете додати колір до кожного пирога за допомогою параметра col. Щоб додати список пояснень до кожного пирога, використовуйте функцію legend():




```
colors <- c("blue", "yellow", "green", "black")
pie(x, label = mylabel, main = "Pie Chart", col = colors)
legend("bottomright", mylabel, fill = colors)
```

Рисунок 31 – Легенда до кругової діаграми

Легенда може бути розташована як завгодно: bottomright, bottom, bottomleft, left, topleft, top, topright, right, center.

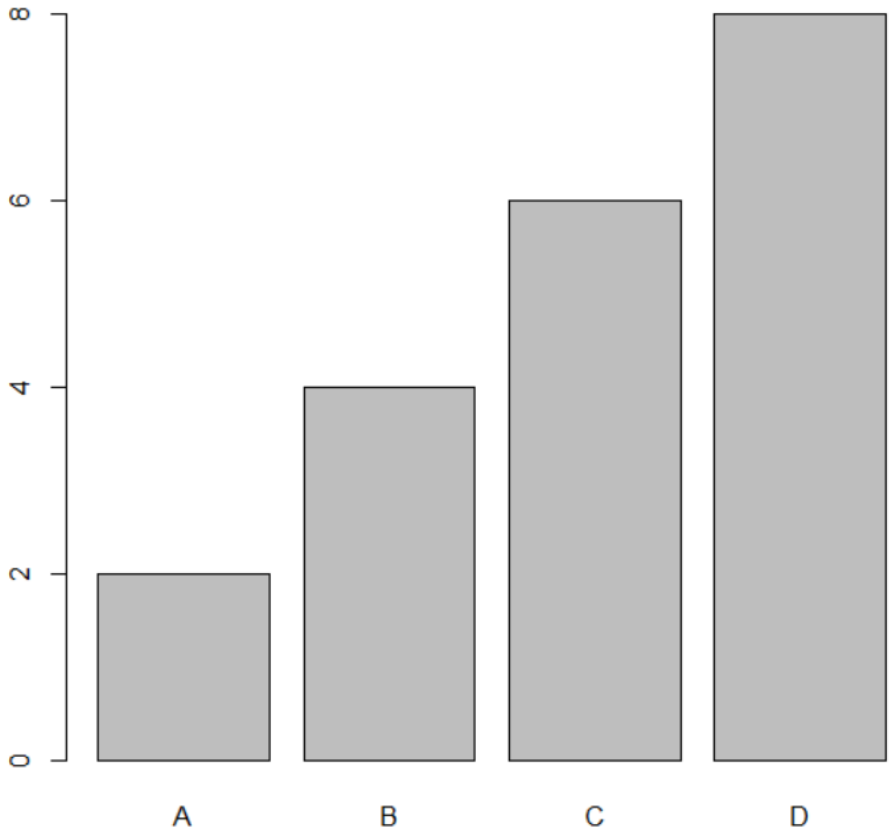
Гістограма використовує прямокутні стовпчики для візуалізації даних. Стовпчасті діаграми можуть відображатися горизонтально або вертикально. Висота або довжина стовпчиків пропорційна до значень, які вони представляють.

Використовуйте функцію barplot() для побудови вертикальної гістограми (рис.32). Змінна x представляє значення на осі x (A,B,C,D).

Змінна y представляє значення на осі y (2,4,6,8).

Потім ми використовуємо функцію barplot() для створення гістограми значень.

Змінна names.arg визначає назви кожного спостереження на осі x.




```
x <- c("A", "B", "C", "D")
y <- c(2, 4, 6, 8)
barplot(y, names.arg = x)
```

Рисунок 32 – Гістограма

Використовуйте параметр col для зміни кольору смуг. Для зміни текстури стовпчиків використовуйте параметр density: barplot(y, names.arg = x, density = 10). Використовуйте параметр width для зміни ширини смуг: barplot(y, names.arg = x, width = c(1,2,3,4)). Якщо ви хочете, щоб стовпчики відображалися горизонтально, а не вертикально, використовуйте horiz=TRUE.

ЛАБОРАТОРНА РОБОТА 5

Тема: Введення в мову R. Статистика.

Приклад виконання роботи:

Статистика - це наука про аналіз, огляд і висновки даних. Деякі основні статистичні числа включають:

- Середнє значення, медіана і мода
- Мінімальне та максимальне значення
- Процентилі
- дисперсія і стандартне відхилення
- Коваріація та кореляція
- Розподіл ймовірностей

Мова R була розроблена двома статистиками. Вона має багато вбудованих функцій, на додаток до бібліотек, призначених саме для статистичного аналізу.

Набір даних - це сукупність даних, часто представлених у вигляді таблиці.

Існує популярний вбудований набір даних у R під назвою "mtcars" (Motor Trend Car Road Tests), який отримано з журналу Motor Trend US за 1974 рік.

У наведених нижче прикладах (і в наступних розділах) ми будемо використовувати набір даних mtcars для статистичних цілей:

```
> mtcars
                     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
>
```

Рисунок 1 – Набір даних mtcars

Ви можете використовувати знак питання (?), щоб отримати інформацію про набір даних mtcars:

Format

A data frame with 32 observations on 11 (numeric) variables.

[, 1] mpg Miles/(US) gallon
[, 2] cyl Number of cylinders
[, 3] disp Displacement (cu.in.)
[, 4] hp Gross horsepower
[, 5] drat Rear axle ratio
[, 6] wt Weight (1000 lbs)
[, 7] qsec 1/4 mile time
[, 8] vs Engine (0 = V-shaped, 1 = straight)
[, 9] am Transmission (0 = automatic, 1 = manual)
[,10] gear Number of forward gears
[,11] carb Number of carburetors

```
> ?mtcars
starting httpd help server ... done
```

Рисунок 2 – Частина інформації про набір

Використовуйте функцію dim(), щоб знайти розмірність набору даних, і функцію names(), щоб переглянути назви змінних. Використовуйте функцію rownames(), щоб отримати назву кожного рядка в першому стовпчику, тобто назву кожного автомобіля. Тепер, коли ми маємо деяку інформацію про набір даних, ми можемо почати аналізувати його за допомогою статистичних чисел.

Наприклад, ми можемо використати функцію summary() для отримання статистичного зведення даних:

```
> summary(mtcars)
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

Рисунок 3 – Статистичне зведення даних

Ми можемо використовувати функції which.max() і which.min(), щоб знайти індексну позицію максимального і мінімального значення в таблиці:

```
> Data_Cars <- mtcars
>
> which.max(Data_Cars$hp)
[1] 31
> which.min(Data_Cars$hp)
[1] 19
```

Рисунок 4 – Індексна позиція максимального і мінімального значення в таблиці за кінськими силами

А ще краще об'єднати which.max() і which.min() з функцією rownames(), щоб отримати назву автомобіля з найбільшою і найменшою потужністю:

```
> rownames(Data_Cars)[which.max(Data_Cars$hp)]
[1] "Maserati Bora"
> rownames(Data_Cars)[which.min(Data_Cars$hp)]
[1] "Honda Civic"
```

Рисунок 5 – Назви автомобілів з максимальним і мінімальним значенням в таблиці за кінськими силами

У статистиці нас часто цікавлять три значення, які нас цікавлять:

- Mean - середнє значення
- Median - центральне значення
- Mode - найпоширеніше значення

Щоб обчислити середнє значення змінної з набору даних mtcars, знайдіть суму всіх значень і розділіть цю суму на кількість значень. На наше щастя, функція mean() в R може зробити це за вас. Медіанне значення - це значення посередині, після того, як ви відсортували всі значення. Якщо ми подивимося на значення змінної wt (з набору даних mtcars), то побачимо, що посередині є два числа. Якщо є два числа посередині, вам потрібно розділити суму цих чисел на два, щоб знайти медіану. На щастя, у R є функція, яка робить все це за вас: Просто використовуйте функцію median() для знаходження середнього значення. Модальне значення - це значення, яке з'являється найбільшу кількість разів. R не має функції для обчислення моди. Однак ми можемо створити власну функцію для її обчислення. Замість того, щоб обчислювати її самостійно, ми можемо використати наступний код для знаходження моди. R не має функції для обчислення моди. Однак ми можемо створити власну функцію для її знаходження.

```
> mean(Data_Cars$wt)
[1] 3.21725
> median(Data_Cars$wt)
[1] 3.325
> names(sort(-table(Data_Cars$wt)))[1]
[1] "3.44"
```

Рисунок 6 – Знаходження mean, median та mode

Перцентилі використовуються в статистиці, щоб дати вам число, яке описує значення, нижче якого знаходиться певний відсоток значень. Що таке 75-й процентиль ваги автомобілів? Відповідь: 3,61 або 3 610 фунтів, що означає, що 75% автомобілів важать 3 610 фунтів або менше:

```
> quantile(Data_Cars$wt, c(0.75))
 75%
3.61
```

Рисунок 7 – Перцентилі

Якщо запустити функцію quantile() без вказівки параметра c(), ви отримаєте процентилі 0, 25, 50, 75 і 100.

ЛАБОРАТОРНА РОБОТА 6-7

Тема: Методи попереднього дослідження часових рядів.

Пояснення

```
setwd('D:\\Studies\\501\\MMN\\.....')
cryptos_price <- read.csv("cryptos_price.csv")
fix(cryptos_price)
str(cryptos_price)
cryptos_price $ds=ymd(cryptos_price $ds)*
unique(cryptos_price$coin)
stellar_price <- dplyr::select(dplyr::filter(cryptos_price, coin == "stellar"), y,
ds)
summary(stellar_price)
```

* – Тут має виникнути помилка, адже немає підключеної бібліотеки, що мала б таку функцію. Для вирішення проблеми скористайтесь керівництвом в додатковому файлі.

```
> cryptos_price $ds=ymd(cryptos_price $ds)
Error in ymd(cryptos_price$ds) : could not find function "ymd"
> library(ymd)
Error in library(ymd) : there is no package called ‘ymd’
```

	y	ds	coin
4220	11.350000	2018-01-11	eos
4221	11.340000	2018-01-10	eos
4222	9.330000	2018-01-09	eos
4223	9.690000	2018-01-08	eos
4224	12.520000	2018-01-07	eos
4225	12.580000	2018-01-06	eos
4226	10.840000	2018-01-05	eos
4227	11.280000	2018-01-04	eos
4228	10.230000	2018-01-03	eos
4229	9.330000	2018-01-02	eos
4230	8.840000	2018-01-01	eos
4231	0.055028	2019-12-06	stellar
4232	0.055518	2019-12-05	stellar
4233	0.055419	2019-12-04	stellar
4234	0.055980	2019-12-03	stellar
4235	0.056078	2019-12-02	stellar
4236	0.058082	2019-12-01	stellar
4237	0.059641	2019-11-30	stellar
4238	0.059152	2019-11-29	stellar
4239	0.058225	2019-11-28	stellar

Рисунок 1 – Загальні дані

```
> str(cryptos_price)
'data.frame':   15510 obs. of  3 variables:
 $ y   : num  7547 7448 7252 7320 7322 ...
 $ ds  : chr  "2019-12-06" "2019-12-05" "2019-12-04" "2019-12-03" ...
 $ coin: chr  "bitcoin" "bitcoin" "bitcoin" "bitcoin" ...
> unique(cryptos_price$coin)
 [1] "bitcoin"   "ethereum"  "xrp"       "tether"    "litecoin"  "eos"       "stellar"   "cardano"   "tron"
[10] "monero"    "tezos"     "chainlink" "neo"       "iota"      "dash"      "maker"     "dogecoin"  "zcash"
[19] "decred"    "qtum"      "augur"     "nano"
> summary(stellar_price)
       y                  ds
 Min.   :0.05450   Min.   :2018-01-01
 1st Qu.:0.08797   1st Qu.:2018-06-26
 Median :0.13200   Median :2018-12-19
 Mean   :0.19069   Mean   :2018-12-19
 3rd Qu.:0.24514   3rd Qu.:2019-06-13
 Max.   :0.89623   Max.   :2019-12-06
```

Рисунок 2 – Деталі вибірки по stellar

stellar_price $ds=ymd(stellar_price $ds)

```
> str(stellar_price)
'data.frame':   705 obs. of  2 variables:
 $ y : num  0.055 0.0555 0.0554 0.056 0.0561 ...
 $ ds: Date, format: "2019-12-06" "2019-12-05" "2019-12-04" "2019-12-03" ...
```

Рисунок 3 – Трансформація в формат Дати

Формат tsibble

Сучасні набори даних часто характеризуються нерегулярною реєстрацією спостережень в часі, наявністю декількох змінних різних типів, кількох групуючих змінних і т. п. Крім того, традиційні формати представлення часових рядів суперечать принципам організації та зберігання "охайних даних" ("tidy data") і, як наслідок, ускладнюють аналіз і моделювання. Для вирішення цих проблем було використано новий формат, який реалізований в пакеті tsibble.

Формат даних tsibble характеризується наступними властивостями:

1. Дані зберігаються в табличному вигляді;

2. В таблиці повинні бути присутніми як мінімум два стовпці – зі значеннями спостерігається в часі кількісної змінної і з впорядкованими за зростанням (тобто від минулого до майбутнього) тимчасовими позначками (стовпець з тимчасовими позначками називається індексуються – index);

3. Крім того, в таблицю можуть входити одна або кілька групуючих змінних (key) - значення цих змінних вказують на приналежність кожного спостереження до відповідного тимчасового ряду;

4. Будь-які спостереження в таблиці можна унікально ідентифікувати по поєднанню значень індексучих і групуючих змінних.

Перетворимо розглянутий набір даних в формат tsibble за допомогою функції as_tsibble ().

```
attach(stellar_price)
as_tsibble(stellar_price, key=NULL, index=ds)
str(stellar_price)
glimpse(stellar_price, width = 60)
detach(stellar_price)
```

Змінна *y* – це вартість stellar (в доларах США), зазначена в день ds. Таблиця stellar містить тільки один часовий ряд. Тому при перетворенні цієї таблиці в формат tsibble аргументу key було присвоєно значення NULL – таким чином, ми повідомили програму, що в таблиці немає групуючих змінних.

Побудова графіків

Зобразимо на рисунку 4 динаміку вартості stellar в розглянутий період часу.

Рисунок 4 – Динаміка вартості валюти stellar

Дані про курс криптовалюти stellar, що представлені на рис. 4, є прикладом нестаціонарного часового ряду. На *нестаціонарність* процесу вказує одразу кілька ознак. По-перше, середнє значення змінної зменшується з часом, а не залишається постійним. По-друге, відстань між піками та спадами змінюється, що вказує на зміну дисперсії процесу з часом. По-третє, процес демонструє сезонну поведінку, не характерну для стаціонарних даних.

Декомпозиція ряду на складові компоненти

У загальному випадку одновимірний часовий ряд з спостережень y_t, врахованих в моменти часу t, можна розкласти на такі складові, або компоненти:

• *тренд* (T_t): характеризує довгострокову тенденцію в даних (зниження або зростання). Тренд може бути лінійним або нелінійним. У деяких

тимчасових рядах може також спостерігатися зміна напрямку тренду (наприклад, коли зростання змінюється спадом).

• *циклічна компонента* (C_t): довгострокові циклічні коливання, зазвичай займають не менше 2 років. Як правило, частота таких змін непостійна.

• *сезонна компонента* (S_t): короткочасні періодичні зміни, що володіють фіксованою частотою (наприклад, добові зміни кількості сонячного світла, що падає на одиницю поверхні Землі).

• *нерегулярне компонента* (E_t): ефекти випадкових факторів ("шум").

Існує кілька методів декомпозиції часових рядів. Одним з найбільш широко використовуваних є розкладання на тренд і сезонну складову за допомогою локальної полиномиальной регресії. Найпростіше з цим методом можна розібратися, застосувавши його до конкретних даних (рис. 5).

```
tsData  <- ts(data=stellar_price,
              start = c(2018,1),
              end = c(2020,1),
              frequency = 30)
stlData <- stl(tsData[,1],s.window="periodic")
plot(stlData)
```

Функція ts використовується для створення об'єктів часових рядів.

Сезонна складова визначається loess, що згладжує сезонні підсерії (ряди всіх значень січня, ...); якщо s.window = " periodic " згладжування ефективно замінено на середнє. Сезонні значення видаляються, а решта згладжуються, щоб знайти тенденцію. Загальний рівень вилучається із сезонної складової та додається до трендової складової. Цей процес повторюється кілька разів. Інша складова - це залишки від сезонної плюс тенденції.

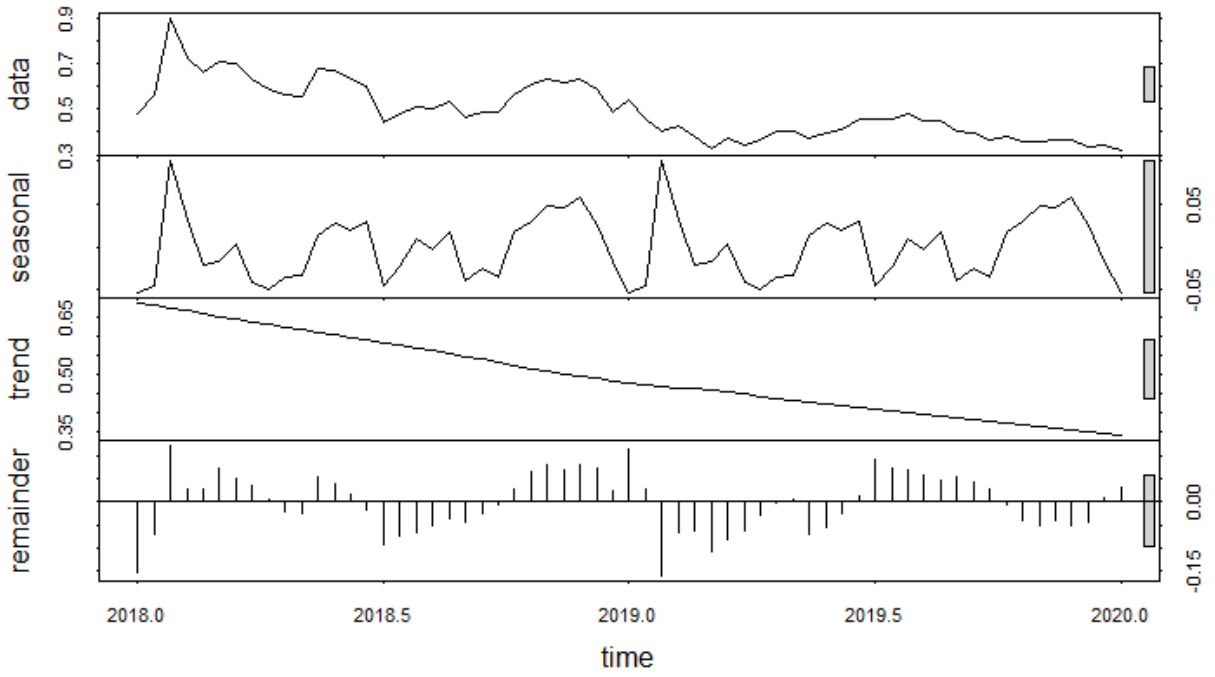


Рисунок 5 – Декомпозиція часового ряду

На рисунку 5, виділені за допомогою методу STL, компоненти часового ряду показані на трьох нижніх графіках (*trend* – тренд, *seasonal* – компонента сезонності, *remainder* – залишки). Якщо їх підсумувати, то отримаємо вихідний ряд y (наведено на верхньому графіку – *data*).

З наведених графіків видно, що вплив тренду домінує, це свідчить про те, що часовий ряд **нестаціонарний**.

Гістограми

У продовженні дослідження даних з вираженою часовою залежністю побудована гістограма часового ряду подібно до того, як це виконується в ході аналізу будь-яких інших даних (рис.). Для повнішого розуміння часового ряду побудована гістограма різниць вихідних даних, щоб отримати тимчасову залежність (рис. 6).

```
hist(stellar_price$y,main = "Histogram of stellar_price",xlab = "stellar_price")
hist(diff(stellar_price$y))
```

Гістограма різниць вихідних даних, що застосовна для часового ряду, представляє більший інтерес, ніж вона ж, побудована для необроблених даних. Оскільки, щодо часових рядів (особливо фінансових) частіше потрібно відстежувати зміни досліджуваної величини, а не її абсолютне значення. Гістограма даних, що мають виражений тренд, не має великого інформативного навантаження (верхня частина рис. 6). Гістограма різниці вказує на те, що значення ряду даних з часом збільшуються (позитивні значення різниць) і зменшуються (негативні значення різниць) приблизно на ту саму величину (права частина рис.6). Значення курсів валюти не показують тимчасове зростання і спад на однакову величину, оскільки мають тенденцію до постійного зменшення (рис. 4). Проте з гістограми різниць видно, що причиною такої тенденції є лише невелике усунення на користь негативних, а не позитивних відмінностей.

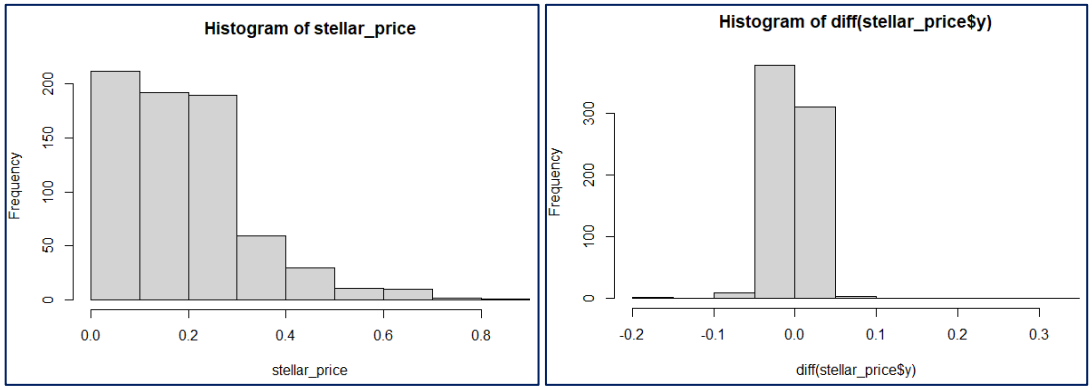


Рисунок 6 – Гістограма часового ряду та різниць початкових даних

Необработанные данные (ліва частина рис. 6) характеризуются большим разбросом значений и не подчиняются нормальному распределению. Этого следовало ожидать, учитывая наличие тренда в исходных данных. Для его удаления и приведения данных к нормальному распределению нужно работать с разностями значений (права частина рис. 6).

Спеціальні методи дослідження часового ряду. *Стаціонарність. Автокореляція.*

Важливою властивістю багатьох часових рядів є наявність автокореляції у значеннях, тобто. зв'язку між значеннями одного ряду, взятими з певним

зрушенням (“лагом”, або “затримкою”). Наявність автокореляції даних є важливою інформацією для побудови надійних прогнозних моделей.

Візуально кореляцію між значеннями одного часового ряду на різних зрушеннях (за замовчуванням від 1 до 9) можна подати за допомогою функції gg_lag() з пакету feasts():

```
stellar_price %>%
  gg_lag(geom = "point") + theme_minimal()
```

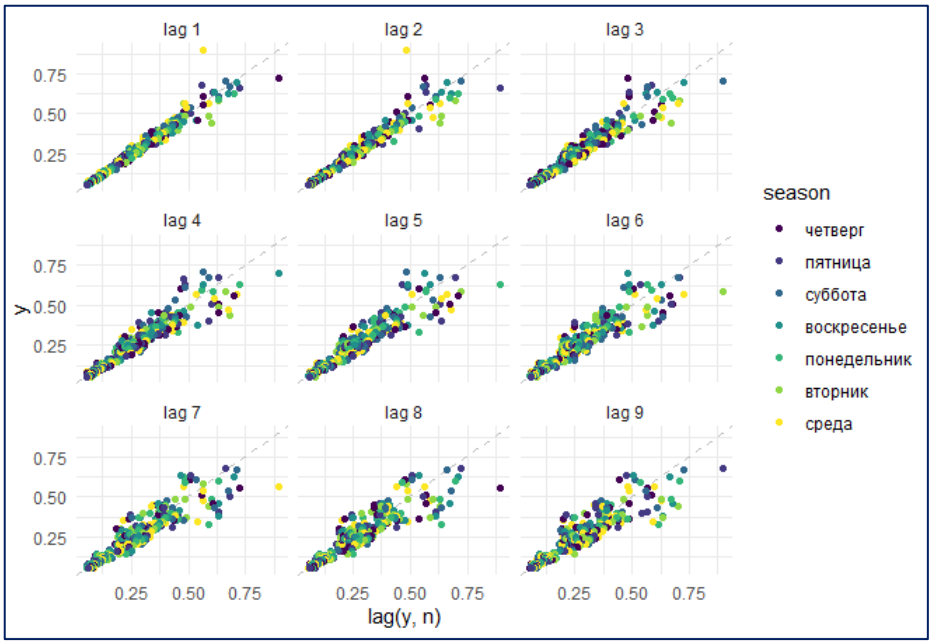


Рисунок 7 – Зв'язок між значеннями часового ряду stellar_price з таблиці cryptos_price та його копіями, зрушеними на кілька часових позначок (від 1 до 9)

На рис. 7 видно помірний зв'язок між значеннями тимчасового ряду stellar_price на всіх дев'яти зсувах. Існує кілька способів, що дозволяють висловити ступінь зв'язку між тимчасовим рядом та його зрушеними копіями (тобто *автокореляційну функцію*) кількісно. У найпростішому випадку при цьому розраховують звичайний лінійний коефіцієнт кореляції. У пакеті feasts розрахунку автокореляційної функції служить команда ACF(). За промовчанням ACF() виконує обчислення для зсувів від 1 до 23, проте максимальне зсув можна змінити за допомогою аргументу lag_max:

```
stellar_price %>%
  ACF(lag_max = 6)
```

```
# A tsibble: 6 x 2 [1D]
    lag   acf
  <lag> <dbl>
1    1D 0.982
2    2D 0.965
3    3D 0.942
4    4D 0.922
5    5D 0.905
6    6D 0.888
```

Отримані коефіцієнти кореляції зображені у вигляді графіка, який має назву *корелограма* (Рис. 8):

```
stellar_price %>%
  ACF() %>%
 autoplot() + theme_minimal()
```

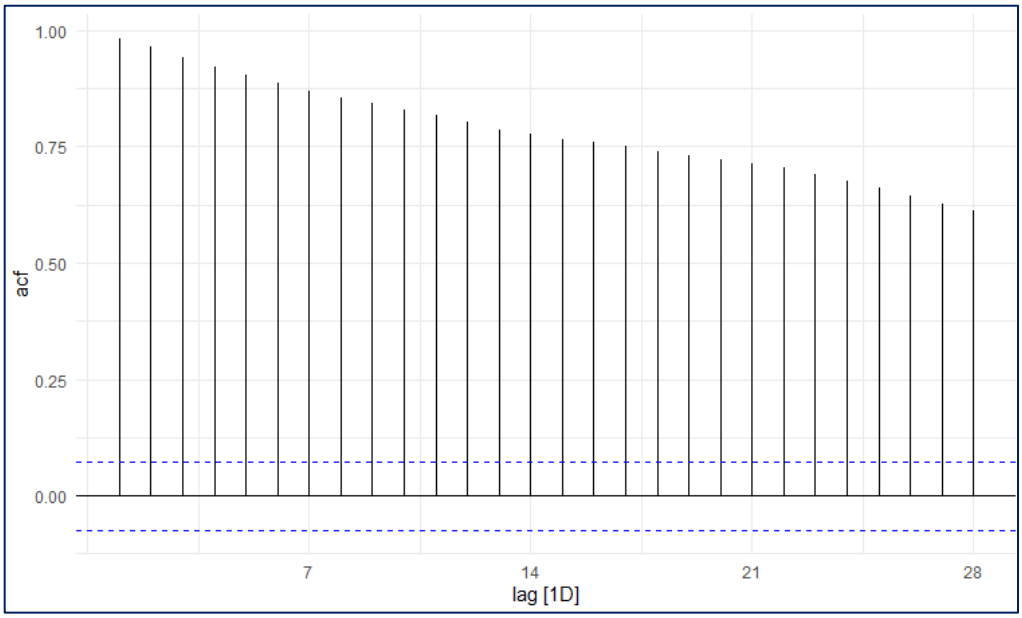


Рисунок 8 – Корелограма часового ряду stellar_price з таблиці cryptos_price для зрушень від 1 до 23

На отриманому графіку (рис. 8) видно, що це коефіцієнти автокореляції є помірковано позитивними, вказуючи на домінування в часі тренду (на зростання). Сині пунктирні лінії відповідають критичним значенням кореляції (на рівні значення 0.05). Якщо отримані коефіцієнти за модулем перевищують

ці критичні значення (як у нашому випадку), то відповідні кореляції можна вважати значущими від нуля.

Аналогічно виконані розрахунки часткової автокореляційної функції, цього служить команда PACF().

```
# A tsibble: 6 x 2 [1D]
    lag      pacf
  <lag>     <dbl>
1    1D  0.982
2    2D -0.00546
3    3D -0.149
4    4D  0.0574
5    5D  0.111
6    6D -0.0507
```

Отримані коефіцієнти кореляції зображені як графіка на рис. 8.

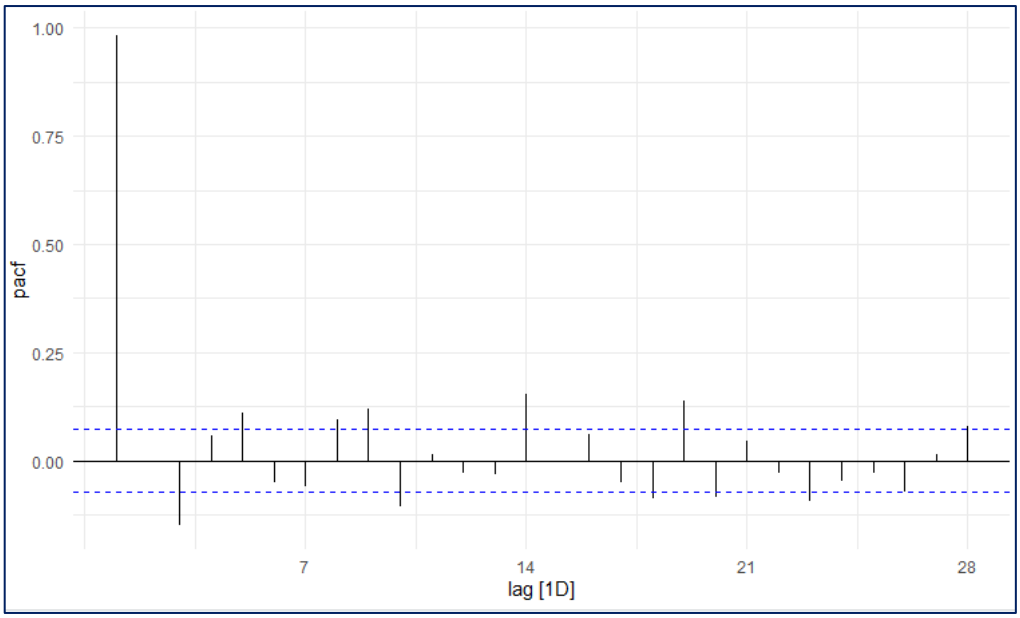


Рисунок 9 – PACF часового ряду stellar_price з таблиці cryptos_price для зрушень від 1 до 23

Причина «критичних» значень PACF, що спостерігаються на рис. 9 для деяких (великих) тимчасових зрушень у наявності сезонних циклів.

Визначення стаціонарності часового ряду за допомогою тестів

1. Тест Дікі-Фуллера (ADF)

У статистиці та економетриці розширений тест Дікі-Фуллера (ADF) перевіряє нульову гіпотезу про наявність одиничного кореня у вибірці часових рядів. Альтернативна гіпотеза відрізняється залежно від того, яка версія тесту

використовується, але, як правило, це стаціонарність або тенденція-стаціонарність. Це доповнена версія тесту Дікі-Фуллера для більшого і складного набору моделей часових рядів.

```
> adf.test(tsData[,1])

        Augmented Dickey-Fuller Test

data:  tsData[, 1]
Dickey-Fuller = -3.2411, Lag order = 3, p-value = 0.08938
alternative hypothesis: stationary
```

Якщо значення p> 0,05 під час тесту ADF (доповнений Дікі-Фуллер) часового ряду, тоді серія називається нестаціонарною і приймає гіпотезу NULL. Якщо значення p <= 0,05, воно відхиляє гіпотезу NULL, яка символізується як H0, і вона називається стаціонарною, коли дані не мають одиничного кореня. Альтернативна гіпотеза символізується як H1. З результату виконання тесту можна сказати, що часовий ряд **нестаціонарний,** так як p=0.09 > 0.05.

2. Тест KPSS

В econometrиці тести Квятковського – Філліпса – Шмідта – Шіна (KPSS) використовуються для перевірки нульової гіпотези про те, що спостережуваний часовий ряд є стаціонарним навколо детермінованої тенденції (тобто тенденції, стаціонарної) проти альтернативи одиничному кореню.

```
> kpss.test(tsData[,1])

        KPSS Test for Level Stationarity

data:  tsData[, 1]
KPSS Level = 1.2073, Truncation lag parameter = 3, p-value = 0.01
```

З результату виконання тесту можно сказати, що часовий ряд **стаціонарний**, так як p <= 0.05.

3. Тест PP

У статистиці тест Філіпса – Перрона (названий на честь Пітера С. Б. Філліпса та П'єра Перрона) є одиничним тестом. Тобто він використовується

для аналізу часових рядів для перевірки нульової гіпотези про те, що часовий ряд інтегрований із порядку.

```
> pp.test(tsData[,1])

        Phillips-Perron Unit Root Test

data:  tsData[, 1]
Dickey-Fuller Z(alpha) = -31.176, Truncation lag parameter = 3,
p-value = 0.01
alternative hypothesis: stationary
```

З результату виконання тесту можна сказати, що часовий ряд **нестаціонарний**, так як p <= 0.05.

ЛАБОРАТОРНА РОБОТА 8-9

Тема: Побудова першої прогнозної моделі для часового ряду.

Пояснення

Для завантаження даних необхідно скопіювати «cryptos_price.csv» в корінь проекту та виконати наступну команду.

```
setwd('D:\\Studie\\.....')
cryptos_price <- read.csv("cryptos_price.csv")
fix(cryptos_price)
str(cryptos_price)
cryptos_price $ds=ymd(cryptos_price $ds)
unique(cryptos_price$coin)
stellar_price <- dplyr::select(dplyr::filter(cryptos_price, coin == "stellar"), y, ds)
summary(stellar_price)
```

	y	ds	coin
4486	0.103379	2019-03-26	stellar
4487	0.102469	2019-03-25	stellar
4488	0.106012	2019-03-24	stellar
4489	0.108281	2019-03-23	stellar
4490	0.108693	2019-03-22	stellar
4491	0.107947	2019-03-21	stellar
4492	0.111403	2019-03-20	stellar
4493	0.113248	2019-03-19	stellar
4494	0.115914	2019-03-18	stellar
4495	0.109385	2019-03-17	stellar
4496	0.109619	2019-03-16	stellar
4497	0.107773	2019-03-15	stellar
4498	0.107272	2019-03-14	stellar
4499	0.108199	2019-03-13	stellar

Рисунок 1 – Загальні дані

```
> str(cryptos_price)
'data.frame':   15510 obs. of  3 variables:
 $ y   : num  7547 7448 7252 7320 7322 ...
 $ ds  : chr  "2019-12-06" "2019-12-05" "2019-12-04" "2019-12-03" ...
 $ coin: chr  "bitcoin" "bitcoin" "bitcoin" "bitcoin" ...
> unique(cryptos_price$coin)
 [1] "bitcoin"   "ethereum"  "xrp"       "tether"    "litecoin"  "eos"       "stellar"   "cardano"   "tron"
[10] "monero"    "tezos"     "chainlink" "neo"       "iota"      "dash"      "maker"     "dogecoin"  "zcash"
[19] "decred"    "qtum"      "augur"     "nano"
> summary(stellar_price)
       y                  ds
 Min.   :0.05450   Min.   :2018-01-01
 1st Qu.:0.08797   1st Qu.:2018-06-26
 Median :0.13200   Median :2018-12-19
 Mean   :0.19069   Mean   :2018-12-19
 3rd Qu.:0.24514   3rd Qu.:2019-06-13
 Max.   :0.89623   Max.   :2019-12-06
```

Рисунок 2 – Деталі вибірки по stellar

```
stellar_price $ds=ymd(stellar_price $ds)
```

```
> str(stellar_price)
'data.frame':   705 obs. of  2 variables:
 $ y : num  0.055 0.0555 0.0554 0.056 0.0561 ...
 $ ds: Date, format: "2019-12-06" "2019-12-05" "2019-12-04" "2019-12-03" ...
```

Рисунок 3 – Трансформація в формат Дати

Підготовка

Припустимо, що нам необхідно зробити прогноз вартості stellar на наступні 90 днів. Для зниження дисперсії виконаємо логарифмування значень вартості кріптовалюти *y*. Розіб'ємо вихідну вибірку на *навчальну* (всі спостереження за винятком останніх 90 днів) і *перевірочну* (останні 90 днів).

```
stellar_train <- stellar_price %>%
  mutate(y=log(y)) %>%
  slice(1:(n()-90)) %>%
  as.data.frame()

stellar_test <- stellar_price %>%
  mutate(y=log(y), ds=as.Date(ds)) %>%
  tail(90) %>%
  as.data.frame()

stellar_train %>%
  ggplot(., aes(ds, y))+geom_line()+theme_minimal()

stellar_test %>%
  ggplot(., aes(ds, y))+geom_line()+theme_minimal()
```

Підгонку моделі будемо виконувати на *навчальних даних* (stellar_train). *Перевірочна вибірка* (stellar_test) стане в нагоді в самому кінці процесу моделювання, щоб з'ясувати наскільки наші очікування щодо якості обраної оптимальної моделі відповідають дійсності.

На рисунку 4 та 5 відображена вибірка stellar після розбиття на тестову та перевірочну вибірку.

Рисунок 4 – Візуалізація *навчальних даних* stellar_train

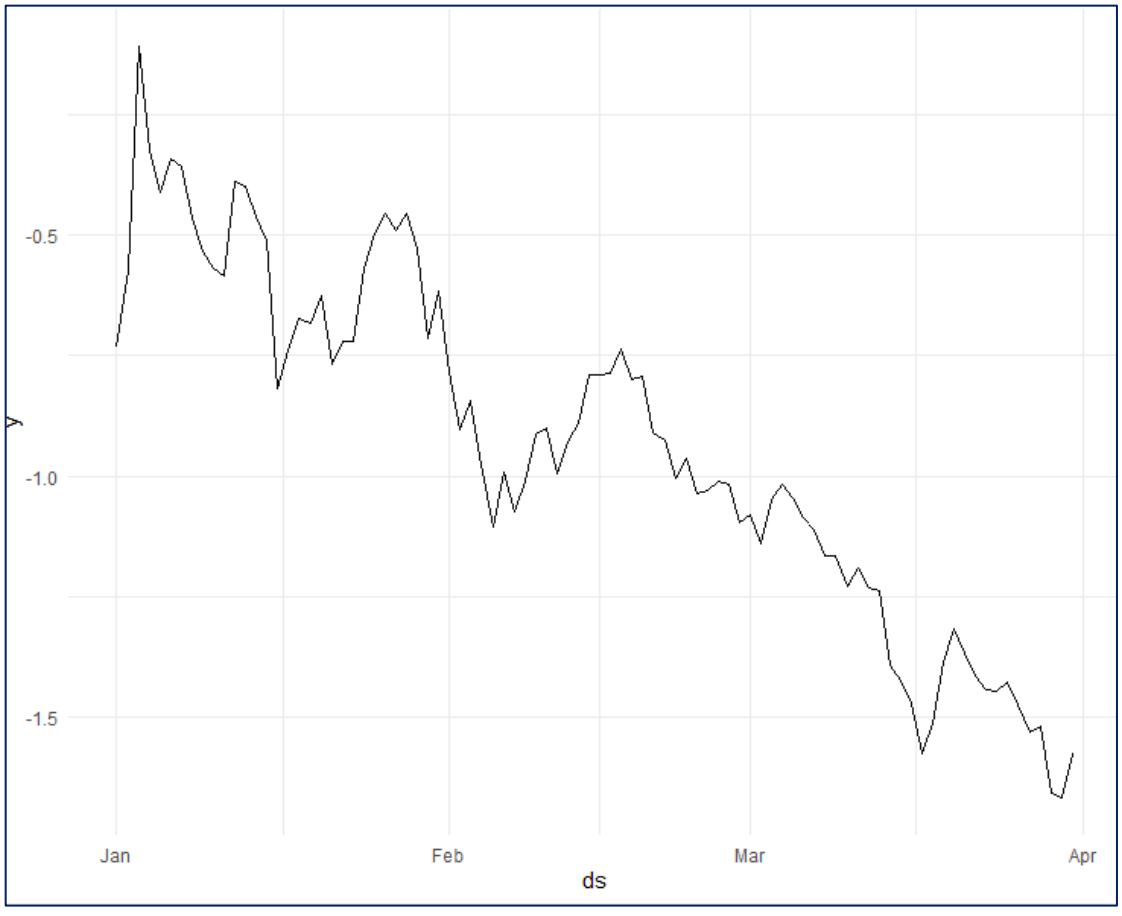


Рисунок 5 – Візуалізація *перевірочної вибірки* stellar_ test

Прогнозування

Побудуємо модель для прогнозування (позначимо її M0) з використанням параметрів, прийнятих в prophet за замовчуванням.

M0 <- prophet(stellar_train)

str(M0)

На рисунку 6 відображена структура моделі M0 за допомогою команди str(M0).

```
> str(M0)
List of 32
 $ growth                : chr "linear"
 $ changepoints          : POSIXct[1:25], format: "2018-03-13" "2018-04-03" "2018-04-24" "2018-05-15" ...
 $ n.changepoints        : num 25
 $ changepoint.range     : num 0.8
 $ yearly.seasonality    : chr "auto"
 $ weekly.seasonality    : chr "auto"
 $ daily.seasonality     : chr "auto"
 $ holidays              : NULL
 $ seasonality.mode      : chr "additive"
 $ seasonality.prior.scale: num 10
 $ changepoint.prior.scale: num 0.05
 $ holidays.prior.scale  : num 10
 $ mcmc.samples          : num 0
 $ interval.width        : num 0.8
 $ uncertainty.samples   : num 1000
 $ specified.changepoints : logi FALSE
 $ start                 : POSIXct[1:1], format: "2018-02-20"
 $ y.scale               : num 2.91
 $ logistic.floor        : logi FALSE
 $ t.scale               : num 56505600
 $ changepoints.t        : num [1:25] 0.0321 0.0642 0.0963 0.1284 0.1606 ...
 $ seasonalities         :List of 1
  ..$ weekly:List of 5
  .. ..$ period        : num 7
  .. ..$ fourier.order : num 3
  .. ..$ prior.scale   : num 10
  .. ..$ mode          : chr "additive"
  .. ..$ condition.name: NULL
 $ extra_regressors      : list()
 $ country_holidays      : NULL
 $ stan.fit              :List of 4
  ..$ par        :List of 6
  .. ..$ k         : num -0.339
  .. ..$ m         : num -0.4
  .. ..$ delta     : num [1:25(1d)] -7.59e-09 2.90e-09 -6.90e-05 -7.39e-04 -4.96e-08 ...
  .. ..$ sigma_obs: num 0.0599
  .. ..$ beta      : num [1:6(1d)] -0.001265 -0.002277 -0.000441 -0.000286 0.000365 ...
  .. ..$ trend     : num [1:655(1d)] -0.4 -0.4 -0.401 -0.402 -0.402 ...
  ..$ value      : num 1426
  ..$ return_code: int 0
  ..$ theta_tilde: num [1, 1:689] -3.39e-01 -4.00e-01 -7.59e-09 2.90e-09 -6.90e-05 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:689] "k" "m" "delta[1]" "delta[2]" ...
 $ params                :List of 6
  ..$ k         : num -0.339
  ..$ m         : num -0.4
  ..$ delta     : num [1, 1:25] -7.59e-09 2.90e-09 -6.90e-05 -7.39e-04 -4.96e-08 ...
  ..$ sigma_obs: num 0.0599
  ..$ beta      : num [1, 1:6] -0.001265 -0.002277 -0.000441 -0.000286 0.000365 ...
  ..$ trend     : num [1:655(1d)] -0.4 -0.4 -0.401 -0.402 -0.402 ...
```

```
$ history               :'data.frame':        655 obs. of  5 variables:
 ..$ y       : num [1:655] -0.911 -0.924 -1.004 -0.962 -1.035 ...
 ..$ ds      : POSIXct[1:655], format: "2018-02-20" "2018-02-21" "2018-02-22" "2018-02-23" ...
 ..$ floor   : num [1:655] 0 0 0 0 0 0 0 0 0 0 ...
 ..$ t       : num [1:655] 0 0.00153 0.00306 0.00459 0.00612 ...
 ..$ y_scaled: num [1:655] -0.313 -0.318 -0.345 -0.331 -0.356 ...
$ history.dates         : POSIXct[1:655], format: "2018-02-20" "2018-02-21" "2018-02-22" "2018-02-23" ...
$ train.holiday.names   : NULL
$ train.component.cols  :'data.frame':        6 obs. of  3 variables:
 ..$ additive_terms      : int [1:6] 1 1 1 1 1 1
 ..$ weekly              : int [1:6] 1 1 1 1 1 1
 ..$ multiplicative_terms: num [1:6] 0 0 0 0 0 0
$ component.modes       :List of 2
 ..$ additive      : chr [1:4] "weekly" "additive_terms" "extra_regressors_additive" "holidays"
 ..$ multiplicative: chr [1:2] "multiplicative_terms" "extra_regressors_multiplicative"
$ fit.kwargs            : list()
- attr(*, "class")= chr [1:2] "prophet" "list"
```

Рисунок 6 – str(M0)

Для отримання прогнозу на основі цієї моделі необхідно спочатку скористатися функцією make_future_dataframe() і створити таблицю з датами, які охоплюють необхідний часовий проміжок в майбутньому ("горизонт"), а потім подати цю таблицю разом з модельним об'єктом на функцію predict ():

future_df<-make_future_dataframe(M0,periods = 90)

forecast_M0<-predict(M0,future_df)

Об'єкт forecast_M0 – це звичайна таблиця, в якій зберігаються значення декількох розрахованих на основі моделі M0 величин, включаючи компоненти моделі, передбачені значення відгуку, а також верхні і нижні межі довірчих інтервалів відповідних величин. Перші кілька передбачених значень вартості stellar та їх (прийняті за замовчуванням) 80% (рис. 7.):

```
> head(forecast_M0)
        ds     trend additive_terms additive_terms_lower additive_terms_upper        weekly  weekly_lower  weekly_upper
1 2018-04-01 -1.184572   4.796346e-03         4.796346e-03         4.796346e-03  4.796346e-03  4.796346e-03  4.796346e-03
2 2018-04-02 -1.186328   3.465377e-03         3.465377e-03         3.465377e-03  3.465377e-03  3.465377e-03  3.465377e-03
3 2018-04-03 -1.188083   6.097395e-04         6.097395e-04         6.097395e-04  6.097395e-04  6.097395e-04  6.097395e-04
4 2018-04-04 -1.189839   4.183431e-05         4.183431e-05         4.183431e-05  4.183431e-05  4.183431e-05  4.183431e-05
5 2018-04-05 -1.191595  -7.592663e-03        -7.592663e-03        -7.592663e-03 -7.592663e-03 -7.592663e-03 -7.592663e-03
6 2018-04-06 -1.193351  -3.034339e-03        -3.034339e-03        -3.034339e-03 -3.034339e-03 -3.034339e-03 -3.034339e-03
  multiplicative_terms multiplicative_terms_lower multiplicative_terms_upper yhat_lower yhat_upper trend_lower trend_upper      yhat
1                    0                          0                          0  -1.392012 -0.9718633   -1.184572   -1.184572 -1.179775
2                    0                          0                          0  -1.391080 -0.9631867   -1.186328   -1.186328 -1.182862
3                    0                          0                          0  -1.398577 -0.9715889   -1.188083   -1.188083 -1.187474
4                    0                          0                          0  -1.416594 -0.9709057   -1.189839   -1.189839 -1.189797
5                    0                          0                          0  -1.408580 -0.9889872   -1.191595   -1.191595 -1.199188
6                    0                          0                          0  -1.408732 -0.9842326   -1.193351   -1.193351 -1.196385
>
```

Рисунок 7 – forecast_M0

Таблицю forecast_M0 і об'єкт M0 далі можна подати на функцію plot(), щоб зобразити підігнані модель і прогнозні значення на графіку (рис. 8):

plot(M0, forecast_M0)

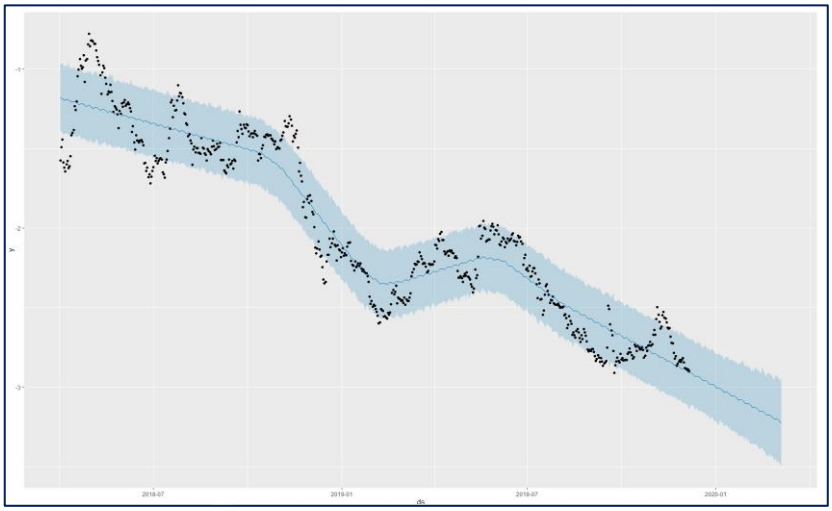


Рисунок 8 – Прогноз за допомогою моделі M0

Аналіз

Точки на рис. 8 відповідають (логорифмічному) значенням вартості stellar з навчальної вибірки. Темно-блакитна лінія – це *передбачені моделлю значення вартості*, а огинає цю лінію світло-блакитна "стрічка" позначає *80% -ві довірчі границі передбачених значень*. Прогнозні значення *y* на наступні 90 днів видно в правій частині графіка (після 2020-01).

Висновок. З графіку видно, що прогноз *незадовільний*, так як він недостатньо точно передає структуру вибірки. Вірогідно причина в тому, що вибірка недостатньо насичена даними.

Змінимо розмір тренувальної вибірки з 90 днів до 240 (рис. 9.). З графіку видно, що прогноз став дещо точніше повторювати структуру тренувальної вибірки але все одно *незадовільно*.

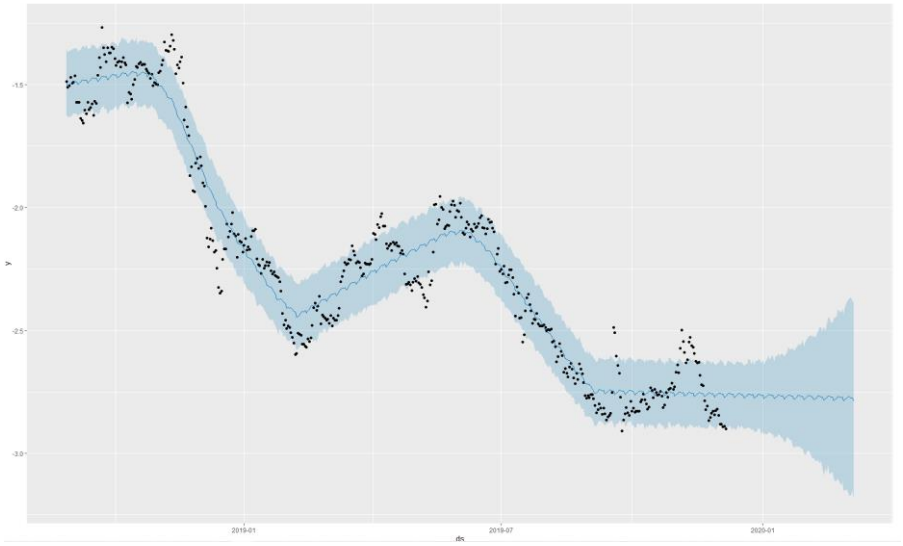


Рисунок 9 – Прогноз – 240 днів

Візуалізуємо окремі компоненти моделі (рис. 10):

prophet_plot_components(M0, forecast_M0)

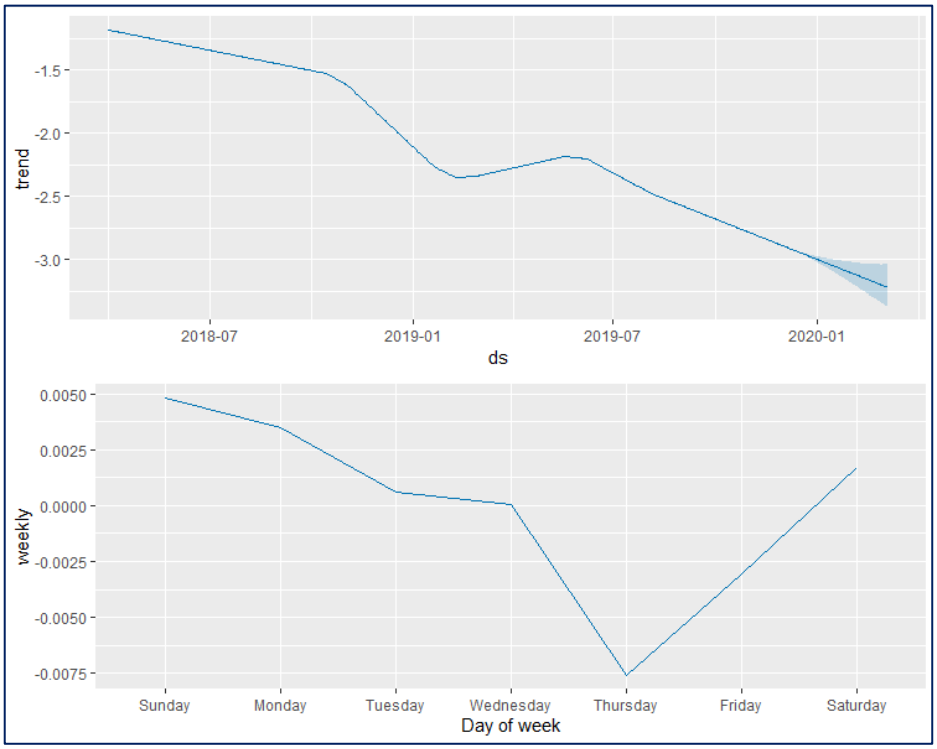


Рисунок 10 – Окремі компоненти моделі – 90 днів

На рис. 10 видно, що модель M0 добре передає наявний в даних складний тренд. Видно також, що в цьому часовому ряду є практично неіснуючі коливання в межах тижня. Шкали ординат цих двох графіків допомагають оцінити внесок кожної з компонент. Отримана модель досить добре передає властивості цього ряду. Проте, якість прогнозу M0 залишає бажати кращого. На даному етапі моделювання головною ознакою незадовільної якості прогнозів M0 є надмірно розширюються довірчі границі прогнозних значень (рис. 10).

ЛАБОРАТОРНА РОБОТА 10-11

Тема: Вивчення прогнозних моделей для часового ряду.

Пояснення

Зміна параметрів моделі M0

Змінюючи значення основних аргументів функції prophet () отримаємо альтернативні моделі для прогнозування. У зв'язку зі складністю лінії тренду часового ряду поставимо точки зламу тренда. Це завдання можна виконати двома способами: поставити самостійно або довіритися їх автоматичного виявлення.

Автоматичний режим виявлення точок зламу тренда

В автоматичному режимі при ініціалізації моделі 25 потенційних точок зламу рівномірно розподіляються в межах інтервалу, який охоплює перші 80% спостережень з *навчальної вибірки*. Це сталося, коли була побудована модель M0 в попередній лабораторній роботі. Однак ці 25 точок – лише передбачувані місця істотних змін в тренді: в більшості випадків на практиці тренд часового ряду не змінюється так часто. Тому в ході підгонки моделі спрацьовує *механізм регуляризації*, в результаті чого вибирається мінімально необхідну кількість точок зламу.

Зобразимо ці автоматично виявлені точки зламу за допомогою функції add_changepoints_to_plot (). Так, для моделі M0 отримуємо (рис. 9):

```
plot(M0,forecast_M0)+add_changepoints_to_plot(M0)
```

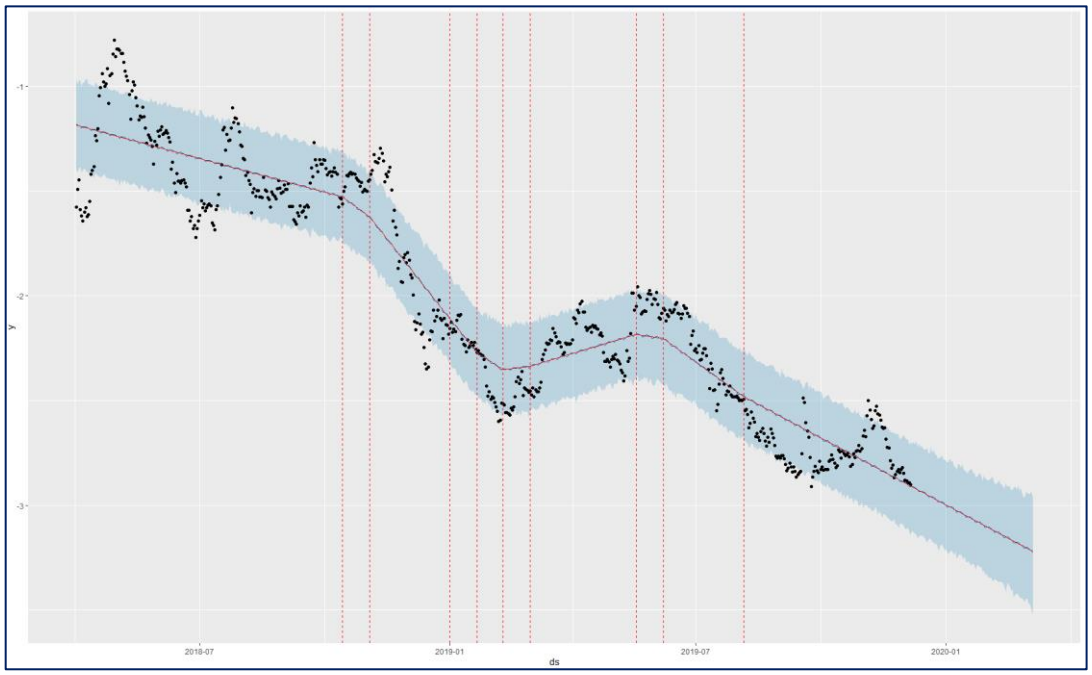


Рисунок 9 – Автоматично виявлені точки зламу тренда

Висновок: судячи по графіку, що отриманомо, модель M0 недооцінює кількість "переломних моментів" в тренді.

Побудуємо нову модель M1, яка буде инициализирована з меншим початковим кількістю потенційних точок зламу (**15** замість 25 за замовчуванням) (рис. 10):

```
M1<-prophet(stellar_train,n.changepoints = 15)
forecast_M1<-predict(M1,future_df)
plot(M1,forecast_M1)+add_changepoints_to_plot(M1)
```

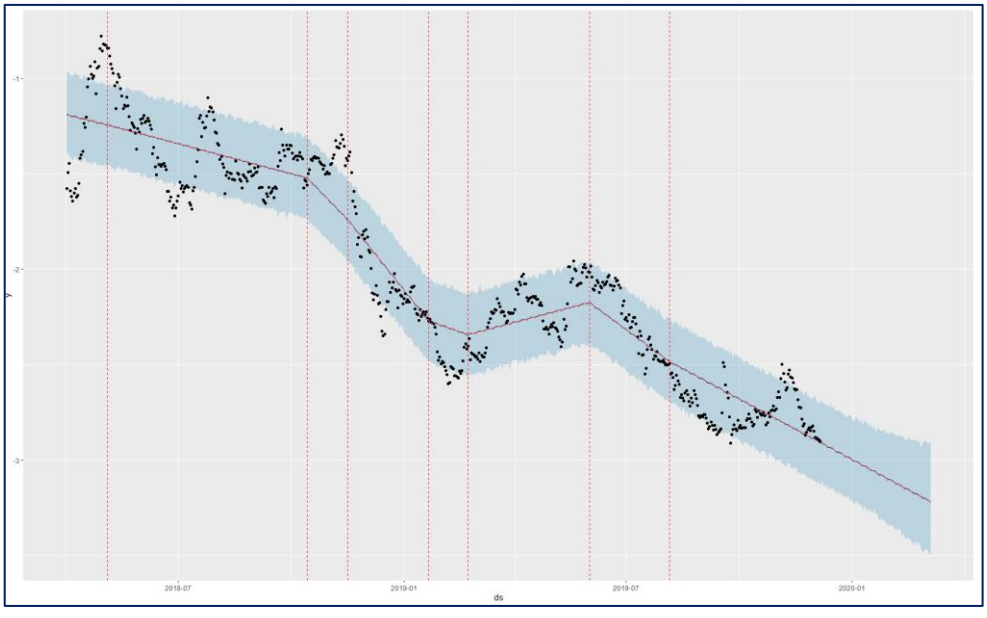


Рисунок 10 – Автоматично виявлені точки зламу з меншим початковим кількістю потенційних точок зламу

Висновок: знову оцінений тренд вийшов менш точним, ніж в моделі M0.

Побудуємо ще одну модель для прогнозування (позначимо її M2). Крім зміни початкового кількості потенційних точок зламу тренда змінимо також *часовий інтервал*, в межах якого відбувається їх оцінювання. Збільшимо інтервал до 90%, скориставшись аргументом changepoint.range і одночасно збільшимо кількість потенційних точок зламу з 15 до **20**, оскільки на більшій проміжку часу можна очікувати більше перепадів в тренді (рис. 11.):

```
M2<-prophet(stellar_train,n.changepoints = 20,changepoint.range = 0.9)
forecast_M2<-predict(M2,future_df)
plot(M2,forecast_M2)+add_changepoints_to_plot(M2)
```

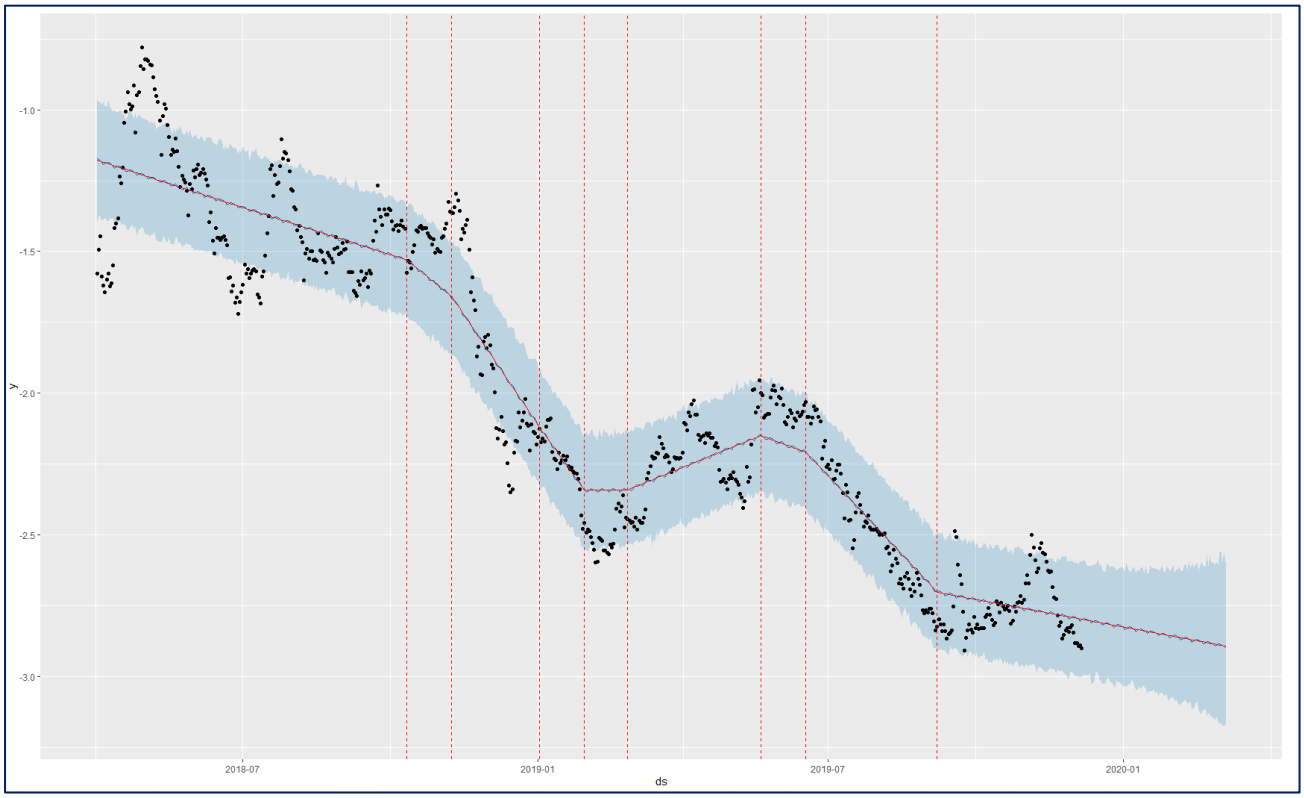


Рисунок 11 – Автоматично виявлені точки зламу з меншою початковою кількістю потенційних точок зламу та часовим інтервалом 90%

Висновок: з рис. 11 видно, що отримана модель M2 набагато краще передає властивості аналізованого часового ряду. Це стосується і одержуваного з її допомогою прогнозу (як з точки зору напрямку тренда, так і з точки зору ширини довірчих меж передбачених значень).

Побудуємо наступну альтернативну модель для прогнозування (M3), змінюючи параметр для налаштування гладкості тренда в часовому ряду, що моделюємо, – це changepoint.prior.scale. Чим більше значення цього параметра (в порівнянні з прийнятим за замовчуванням значенням 0.05), тим більше точок зламу залишиться в отриманої моделі. У моделі M3 збільшуємо інтервал, в межах якого оцінюються точки зламу тренда (до 90%), одночасно збільшуючи рівень регуляризації за допомогою параметра changepoint.prior.scale. Початкова кількість потенційних точок зламу залишимо рівним значенню, прийнятому за замовчуванням (25) (рис. 12):

```
M3<-prophet(stellar_train, changepoint.range = 0.9,
changepoint.prior.scale = 0.50)
forecast_M3<-predict(M3,future_df)
plot(M3,forecast_M3)+add_changepoints_to_plot(M3)
```

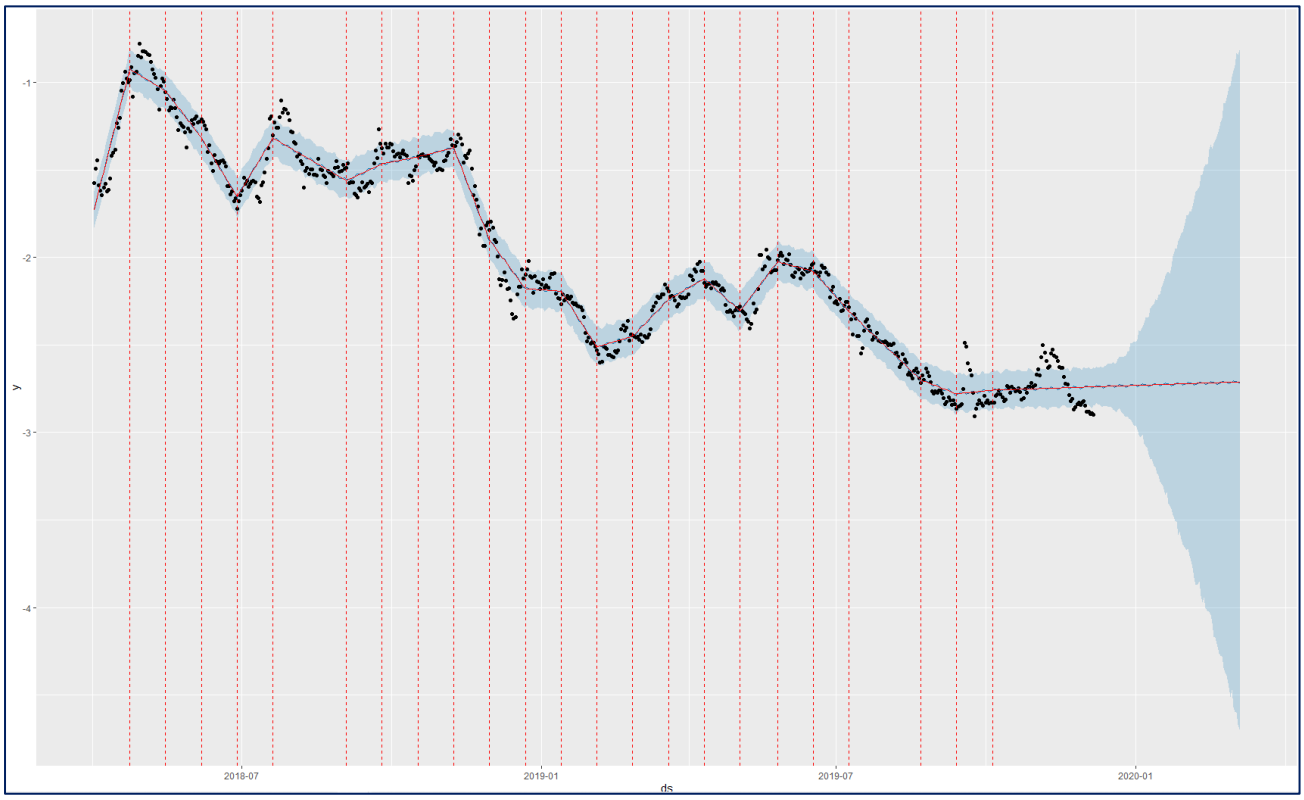


Рисунок 12 – Автоматично виявлені точки зламу з інтервалом 90% з більшим параметром гладкості

Висновок: з рис. 12 видно, що модель M3 дала більш кращі резільтати ніж попередні моделі, але ширина довірчих меж передбачених значень стала дуже великою.

Встановлення точок зламу тренда «вручну»

Побудуємо модель M4 і на цей раз поставимо точки зламу тренда «вручну», а не в автоматичному режимі. Для цього завдання скористаємося аргументом changepoints. Ставлячи точки зламу тренда самостійно врахуємо, що вибір дат, що подаються на аргумент changepoints, заснований на візуальному аналізі навчальних даних, а також змінемо параметр для налаштування гладкості тренда в моделируемом тимчасовому ряду (рис. 13.).

```
M4<-prophet(stellar_train, changepoint.prior.scale = 0.50, changepoints = c(
  "2018-05-01", "2018-06-01", "2018-07-01",
  "2018-08-01", "2018-11-17", "2019-02-04",
```

```
  "2019-04-04", "2019-05-04", "2019-07-04",
  "2019-09-21", "2019-11-07"
))
forecast_M4<-predict(M4,future_df)
plot(M4,forecast_M4)+add_changepoints_to_plot(M4)
```

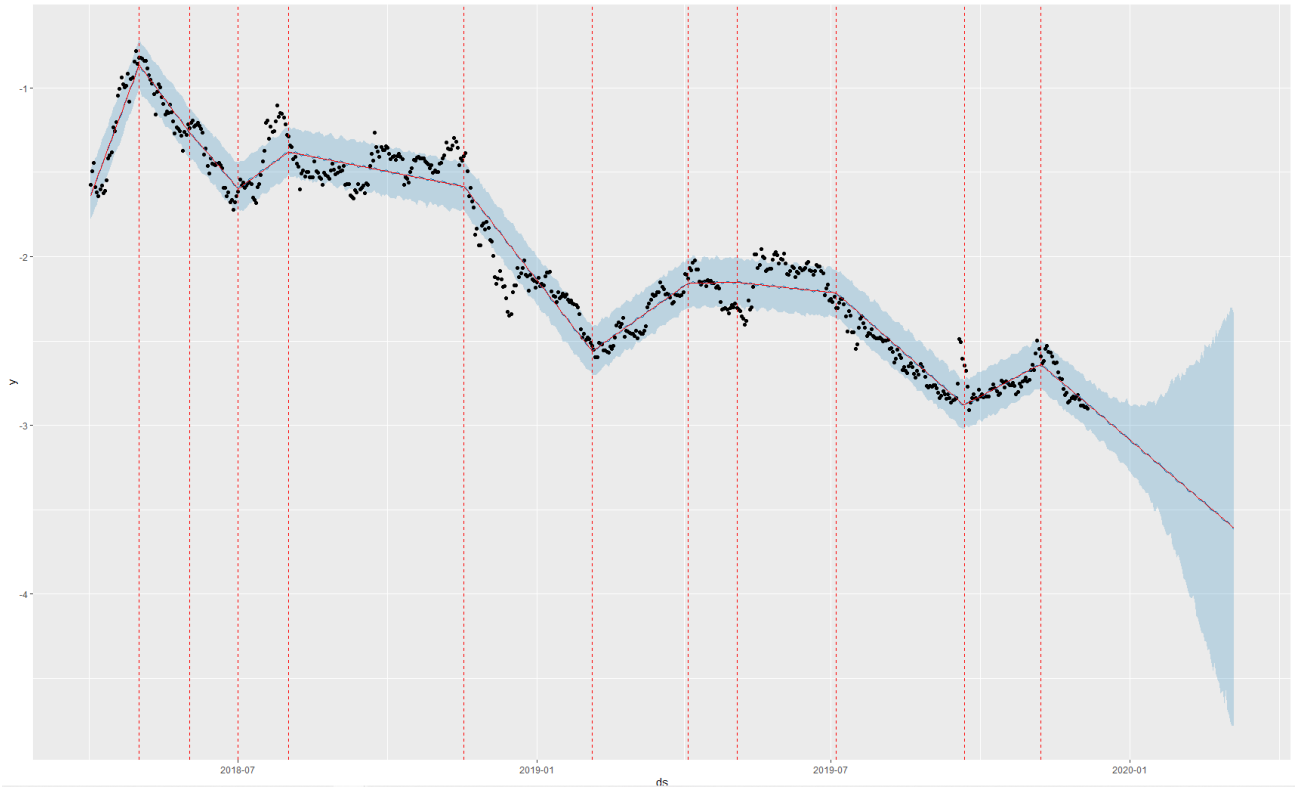


Рисунок 13 – Встановлення точок зламу тренда «вручну»

ЛАБОРАТОРНА РОБОТА 12-13

Тема: Підбір прогнозних моделей часового ряду. Сезонні компоненти.

Пояснення

Річна, тижнева і денна компоненти

Сезонна компонента є однією зі складових при підгонки часового ряду за допомогою *адитивних регресійних моделей*. Сезонні компоненти аппроксимируются за допомогою часткових сум ряду Фур'є, число членів якого (порядок) визначає гладкість відповідної функції.

Функція prophet () має три аргументи, за допомогою яких можна контролювати гладкість функцій річний, тижневої та денної сезонності: yearly.seasonality, weekly.seasonality і daily.seasonality. Збільшення значень цих аргументів призведе до підгонці менш гладких функцій відповідних компонент, що одночасно збільшить ризик перенавчання моделі. Збільшивши значення аргументу yearly.seasonality до 20, отримаємо наступну модель (рис. 9):

```
M3B <- prophet(stellar_train,
               yearly.seasonality = 20,
               changepoint.range = 0.9,
               changepoint.prior.scale = 0.02)
prophet:::plot_yearly(M3B)
```

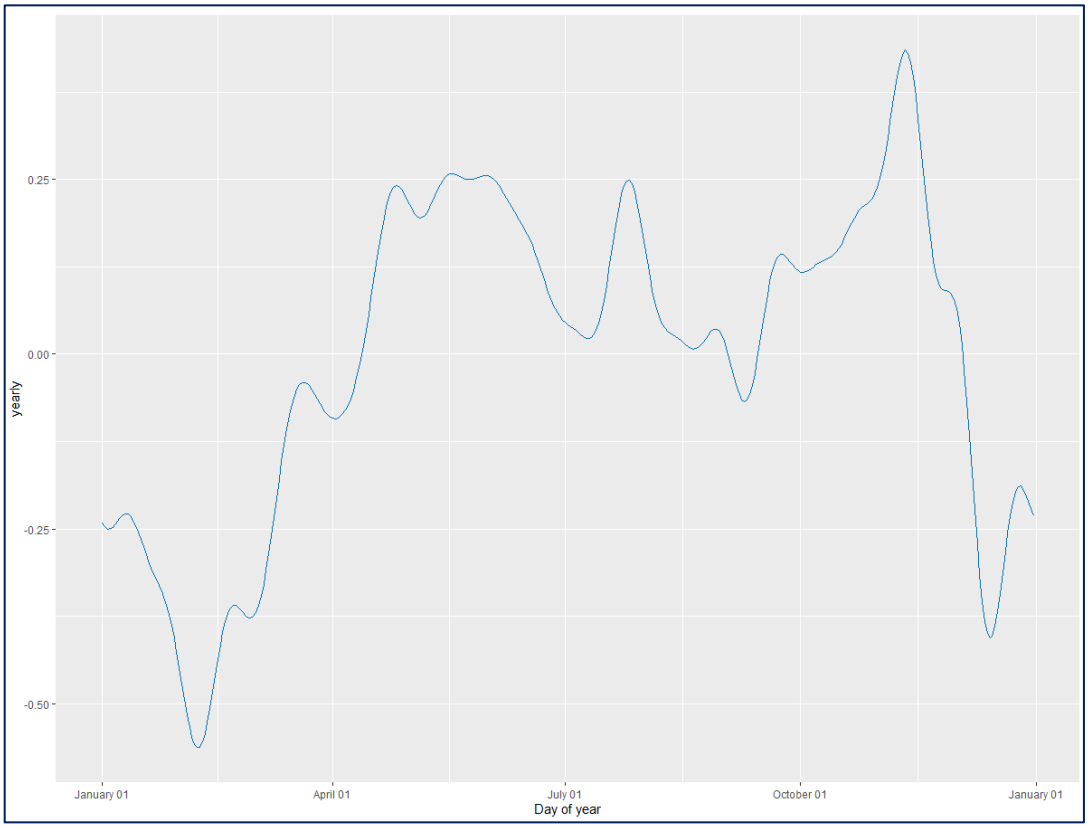


Рисунок 9 – Функція річний сезонності, оцінена за допомогою моделі M3B

Призначені для користувача сезонні компоненти

Для даних, що охоплюють як мінімум два роки, функція rophet () автоматично додасть в модель компоненти *річний* і *тижневої* сезонності. Якщо гранулярность даних перевищує денну (наприклад, коли є погодинні спостереження залежної змінної), то в модель автоматично буде додана також і компонента *денної* сезонності. Крім цього, є можливість додати і будь-які інші сезонні компоненти за допомогою функції add_seasonality () (наприклад, *годинну, місячну, квартальну* і т.п.).

У наведеному нижче коді спочатку відключаємо автоматично додається в модель *тижневу* сезонність і замість неї додаємо *місячну* (допустивши, що один місячний період становить 30.5 днів). На рис.10 представлені всі сезонні компоненти отриманої моделі.

M10 <- prophet(weekly.seasonality = FALSE)

```
M10 <- add_seasonality(m = M10,
                       name = "monthly",
                       period = 30.5,
                       fourier.order = 5)
M10 <- fit.prophet(M10, stellar_train)
forecast_M10 <- predict(M10, future_df)
prophet_plot_components(M10, forecast_M10)
```

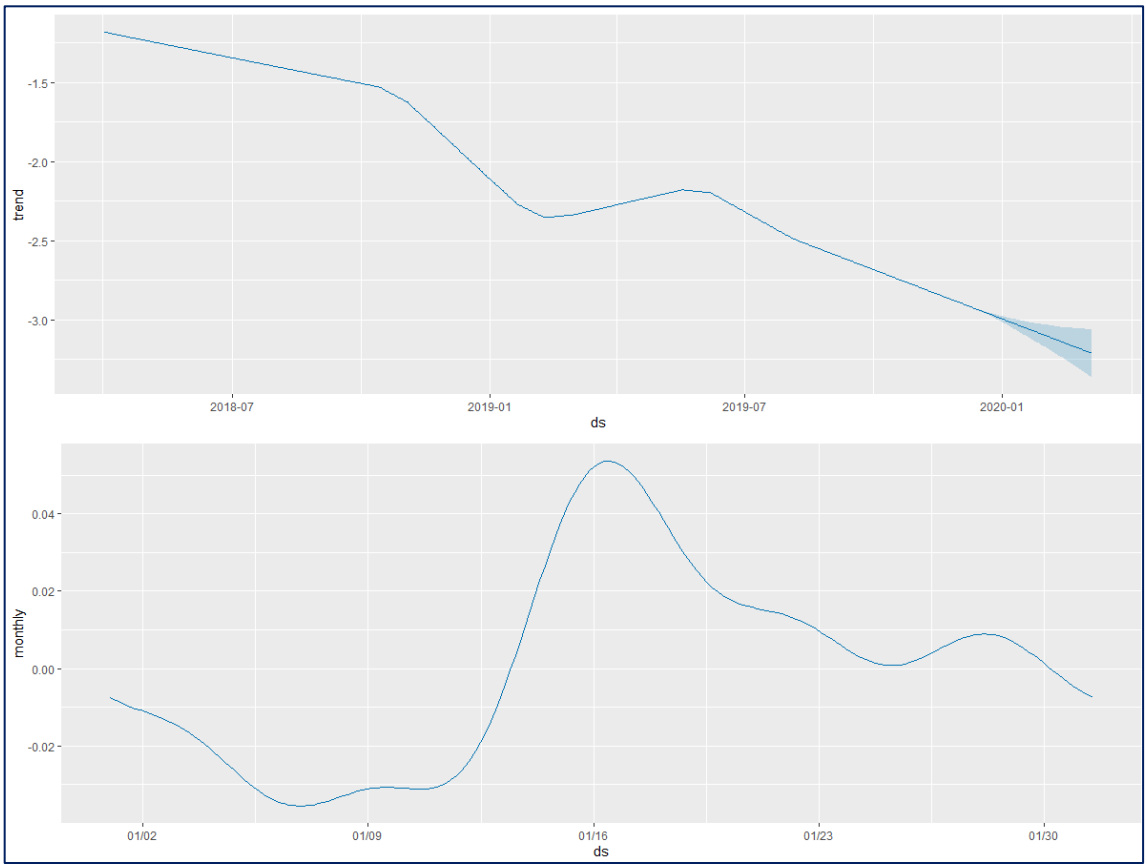


Рисунок 10 – Компоненти моделі M10

Висновок: Відповідно до отриманої моделі чітко видно місячні коливання та тренд. Проте, якість прогнозу M10 залишає бажати кращого. На даному етапі моделювання головною ознакою незадовільної якості прогнозів M10 є *надмірно розширюються довірчі кордону прогнозних значень* (рис. 10).

Аналогічним чином замість компоненти місячних коливань додамо компоненту *квартальної* сезонності (задавши період довжиною 365.25 / 4 днів). На рис.11 представлені всі сезонні компоненти отриманої моделі.

```
M11 <- prophet(weekly.seasonality = FALSE)
```

```
M11 <- add_seasonality(m = M11,
                       name = "quarter",
                       period = 365.25/4,
                       fourier.order = 2)
M11 <- fit.prophet(M11, stellar_train)
forecast_M11 <- predict(M11, future_df)
prophet_plot_components(M11, forecast_M11)
```

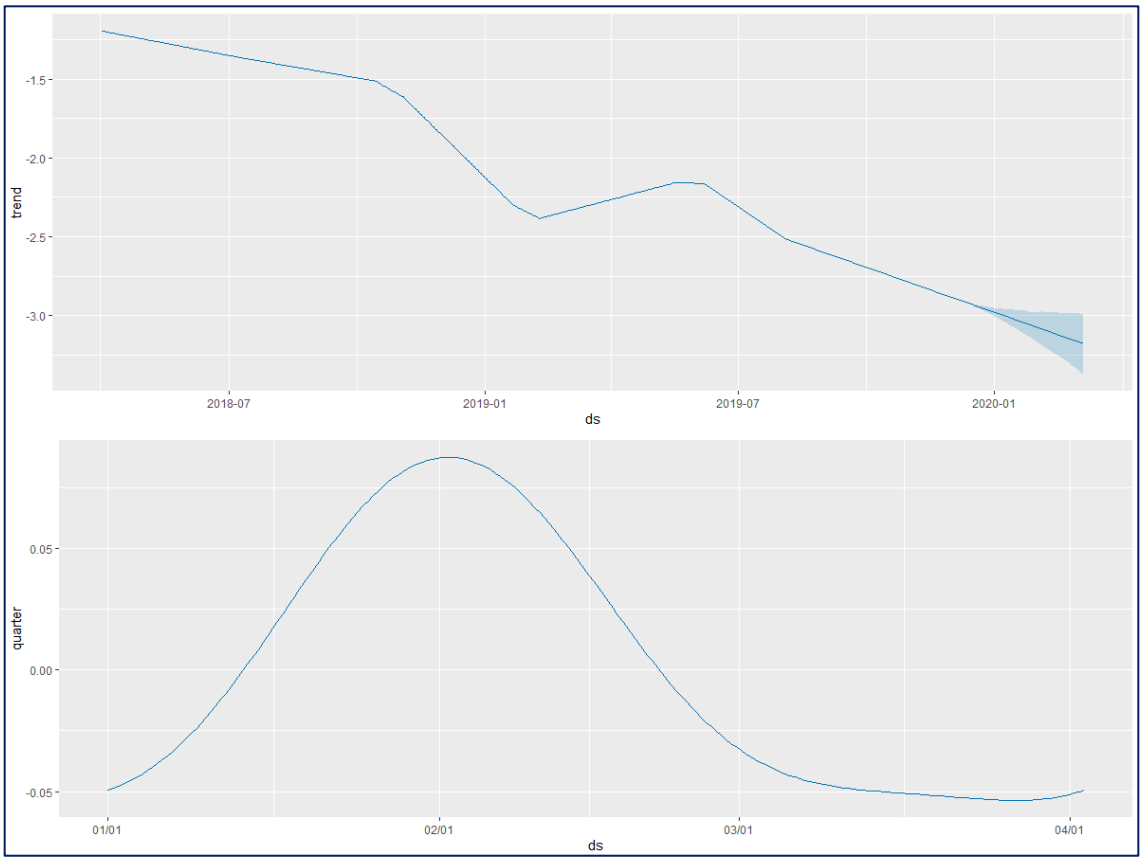


Рисунок 11 – Компоненти моделі M11

Висновок: Відповідно до отриманої моделі чітко видно коливання по *кварталу* яке дуже гладке і це недуже добре та тренд. В порівнянні з M10 тренд майже незмінився (рис. 11).

Умовні режими сезонності

У ряді випадків функція, що апроксимує ту чи іншу сезонну складову, може змінювати свої властивості в залежності від якихось сторонніх чинників. Наприклад, коливання протягом робочих днів можуть мати характер, сильно відрізняється від такого у вихідні дні. Пакет prophet дозволяє моделювати такі

умовні режими сезонності (тобто режими, які залежать від сторонніх чинників) за допомогою аргументу condition.name функції add_seasonality (). На цей аргумент подається ім'я (булевої) змінної, яка визначає відповідний режим. Такі змінні повинні зберігається в тій же таблиці, що і основні дані по тимчасовому ряду.

Як приклад припустимо, що тижневі коливання вартості stellar в літні місяці відрізняються від таких в інші місяці. Щоб змоделювати таке розходження додамо в таблицю з даними stellar_train дві нові індикаторні змінні: summer (приймає значення TRUE в літні місяці і FALSE в інші місяці) і not_summer (TRUE в нелітні місяці і FALSE влітку). Важливо пам'ятати, що такі ж змінні потрібно додати і в таблицю з майбутніми датами future_df - інакше прогнозні значення розрахувати не вийде:

```
is_summer <- function(ds) {
  month <- as.numeric(format(ds, '%m'))
  return(month > 5 & month < 9)
}
stellar_train$summer <- is_summer(stellar_train$ds)
stellar_train$not_summer <- !stellar_train$summer
future_df$summer <- is_summer(future_df$ds)
future_df$not_summer <- !future_df$summer
M12 <- prophet(weekly.seasonality = FALSE)
M12 <- add_seasonality(M12, name = 'weekly_summer',
                       period = 7,
                       fourier.order = 3,
                       condition.name = 'summer')
M12 <- add_seasonality(M12, name = "weekly_not_summer",
                       period = 7,
                       fourier.order = 3,
                       condition.name = "not_summer")
M12 <- fit.prophet(M12, stellar_train)
```

```
forecast_M12 <- predict(M12, future_df)
prophet_plot_components(M12, forecast_M12)
```

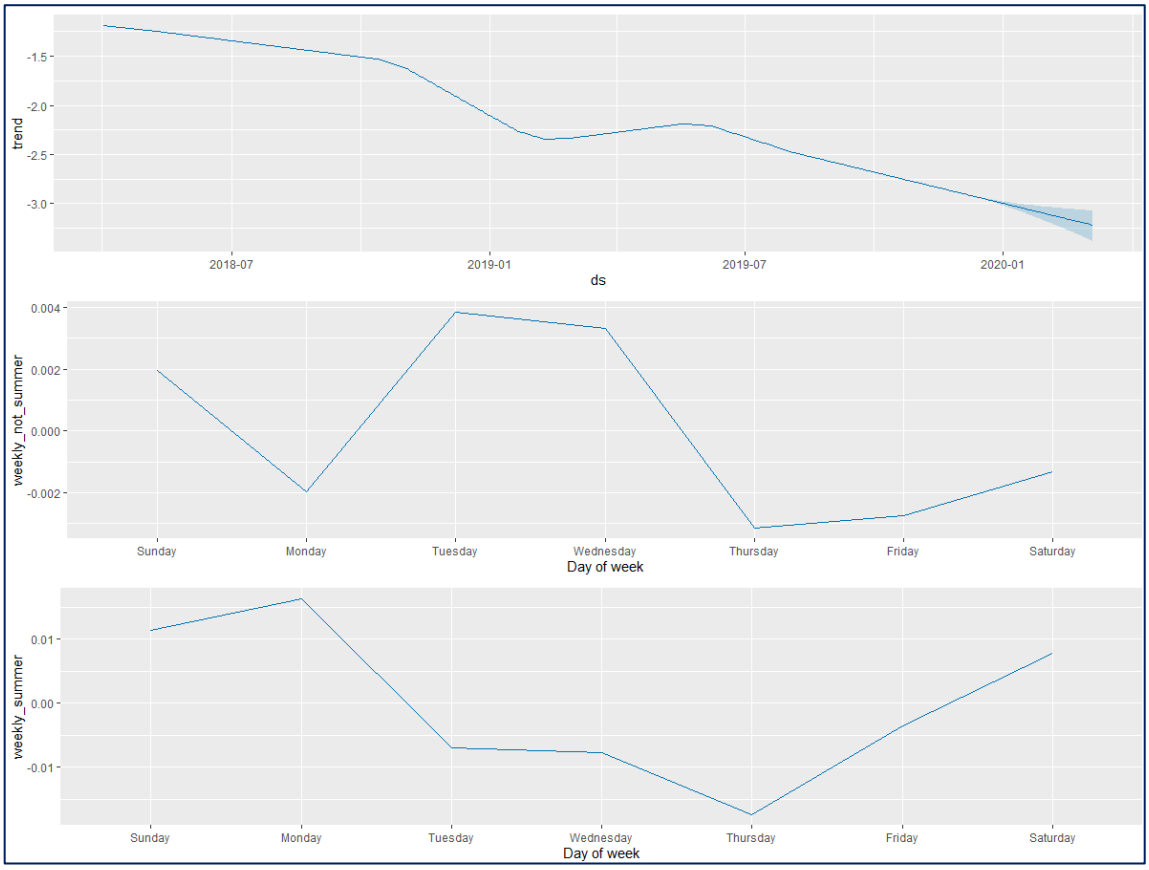


Рисунок 12 – Компоненти моделі M12

Висновок: Відповідно до отриманої моделі, в нелітні місяці вартість stellar протягом тижня зазвичай досягає максимуму по вівторкам та середах, тоді як в літні місяці по понеділкам, а вівторок та серела показують різке падіння (рис. 12).

Аддитивна і мультиплікативна сезонності

За характером функціонального зв'язку між своїми компонентами моделі часових рядів діляться на два основних типи – ***адитивні*** і ***мультиплікативні***. Перший з них застосовується у випадках, коли амплітуда сезонних коливань приблизно постійна. Якщо ж ця амплітуда помітно змінюється в часі (зазвичай зростає), то будують мультипликативную модель.

У пакеті prophet за замовчуванням підганяються адитивні моделі часових рядів. У мультиплікативний моделях, як випливає з їх назви, сезонна компонента множиться на тренд (в зв'язку з цим внесок сезонних коливань моделюється у вигляді частки (%) від рівня тренда).

Припустимо, що амплітуда всіх сезонних компонент істотно змінюється в часі. Для підгонки відповідних моделей скористаємося аргументом seasonality.mode функції prophet ():

```
M14 <- prophet(stellar_train, seasonality.mode = "multiplicative")
forecast_M14 <- predict(M14, future_df)
plot(M14, forecast_M14)
```

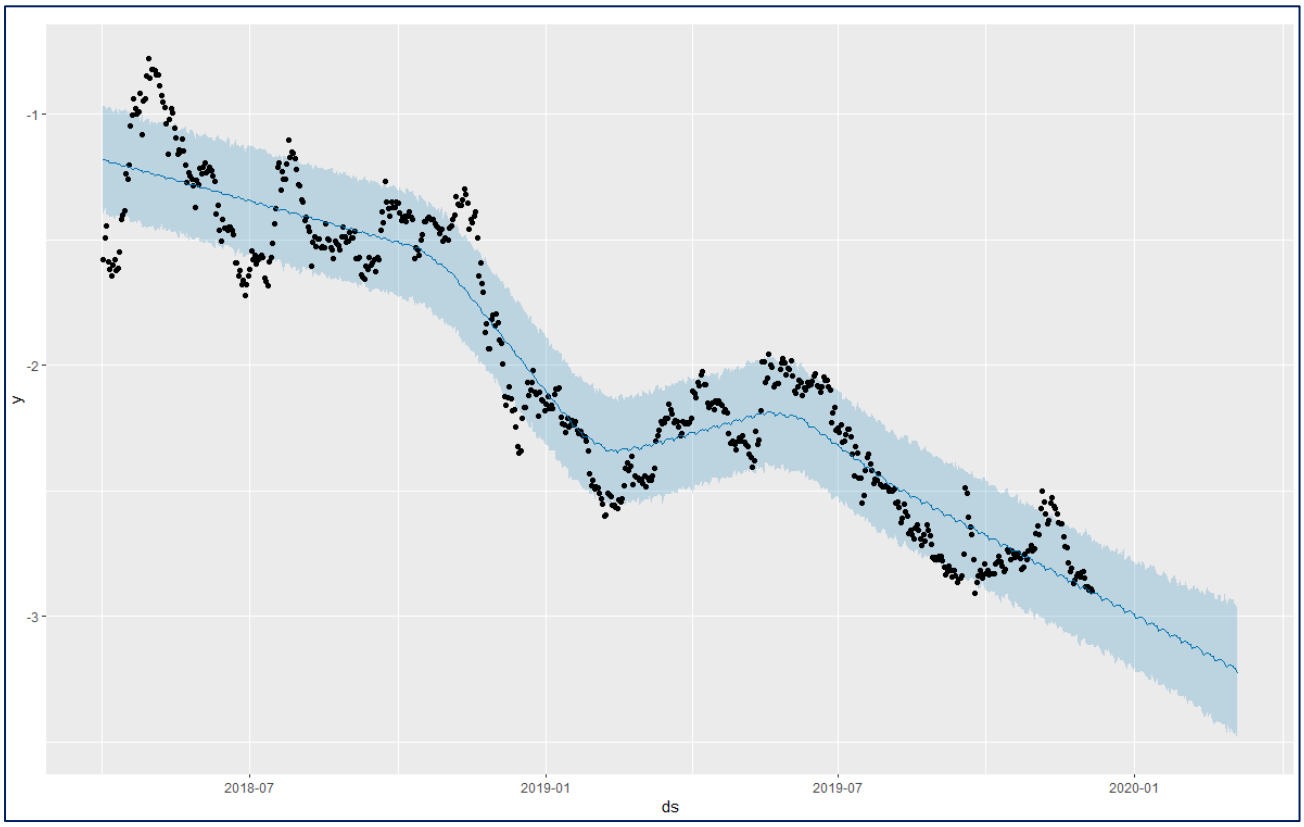


Рисунок 13 – Прогноз вартості stellar, отриманий на основі моделі M14

Висновок: З графіку видно, що прогноз незадовільний, так як він недостатньо точно передає структуру вибірки. Вірогідно прична в тому, що вибірка недостатньо насичена даними (рис. 13).

Побудуємо модель, в якій тижнева коливання представлені в **аддитивном** вигляді, а річні - в **мультипликативном**. Для цього застосовується функція add_seasonality ():

```
M15 <- prophet(yearly.seasonality = FALSE)
M15 <- add_seasonality(M15, name = 'yearly',
                       period = 365.25,
```

```
                fourier.order = 10,
                mode = "multiplicative")
M15 <- fit.prophet(M15, stellar_train)

forecast_M15 <- predict(M15, future_df)
prophet_plot_components(M15, forecast_M15)
```

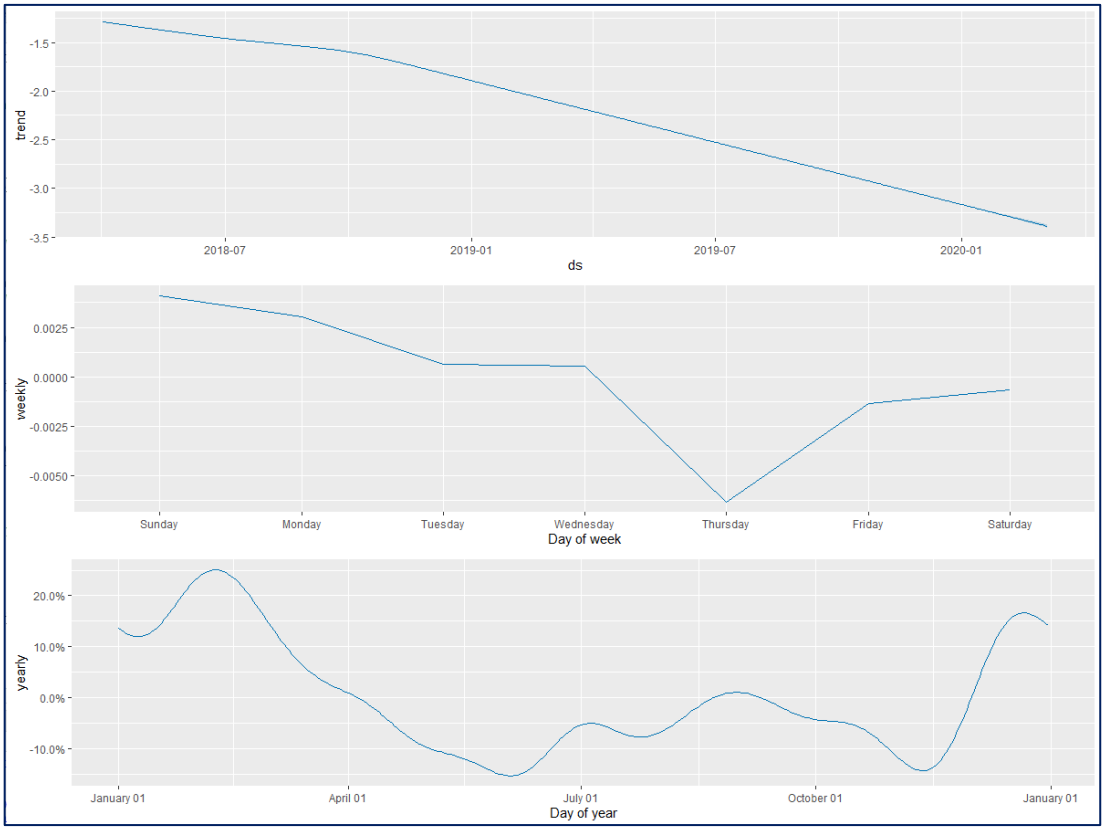


Рисунок 14 – Компоненти моделі M15

Висновок: Відповідно до отриманої моделі, видно, що тижнева компонента різко змінилась порівнянно з M12, а ось річна не дивлячись на те, що стала гладкою зберегла загальну форму з M12 (рис. 14).

ЛАБОРАТОРНА РОБОТА 14-15

Тема: Вибір оптимальної моделі.

Пояснення

Виконання перехресної перевірки

Перехресна перевірка за методом **Імітованих історичних прогнозів** виконується за допомогою функції cross_validation().

Функція cross_validation () повертає таблицю с спостереженим (*y*) і оціненими (*yhat*) значеннями моделюється змінної, а також довірчими границями передбачених значень (yhat_lower і yhat_upper) для кожної точки cutoff і кожної дати ds відповідного прогнозного періоду (для розрахунку було використано модель M3):

```
M3_cv <- cross_validation(M3, initial = 280,
                          period = 60,
                          horizon = 60,
                          units = "days")

head(M3_cv)
```

```
> M3_cv <- cross_validation(M3, initial = 280,
+                           period = 60,
+                           horizon = 60,
+                           units = "days")
Making 5 forecasts with cutoffs between 2019-02-09 and 2019-10-07
> head(M3_cv)
          y         ds      yhat yhat_lower yhat_upper     cutoff
1 -2.520828 2019-02-10 -2.565876  -2.649760  -2.486471 2019-02-09
2 -2.556252 2019-02-11 -2.588336  -2.676681  -2.499886 2019-02-09
3 -2.555982 2019-02-12 -2.604754  -2.681388  -2.519650 2019-02-09
4 -2.564080 2019-02-13 -2.623917  -2.706604  -2.540834 2019-02-09
5 -2.568701 2019-02-14 -2.636507  -2.723046  -2.546980 2019-02-09
6 -2.534820 2019-02-15 -2.648794  -2.744759  -2.550892 2019-02-09
```

Рисунок 9 – Перехресна перевірка за методом Імітованих історичних прогнозів

Метрики якості моделі

Для оцінки якості прогнозів моделей розглянемо наступні показники:

• Ефективне значення помилка (*mean squared error*, MSE);

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

• Квадратний корінь з середньоквадратичної помилки (*root mean squared error*, RMSE);

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$

• Середня абсолютна помилка (*mean absolute error*, MAE);

$$MAE = \sum_{i=1}^{n}|y_i - \hat{y}_i|.$$

• Середня абсолютна питома помилка (*mean absolute percentage error*, MAPE);

$$MAPE = \sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|.$$

• "Покриття" (*coverage*): частка істинних значень модельованої змінної, які знаходяться в межах довірчих границь прогнозу.

У наведених формулах y_i та $\hat{y}_i$ - це справжнє і передбачене значення модельованої змінної відповідно, а n - кількість спостережень.

Функція performance_metrics () має такі аргументи:

• df - таблиця, отримана за допомогою функції cross_validation ();

• metrics - вектор з назвами метрик якості моделі (за замовчуванням цей аргумент приймає значення NULL, що призводить до розрахунку всіх перерахованих вище метрик, тобто c ("mse", "rmse", "mae", "mape", " coverage "));

• rolling_window - розмір "ковзного вікна", в межах якого відбувається усереднення кожної метрики (за замовчуванням приймає значення 0.1, тобто 10% від довжини прогнозного горизонту).

Застосуємо функцію performance_metrics () для розрахунку середньоквадратичної помилки прогнозу моделі M3:

```
performance_metrics(M3_cv, metrics = "mse",
                    rolling_window = 0.1) %>% head()
```

```
> performance_metrics(M3_cv, metrics = "mse",
+                     rolling_window = 0.1) %>% head()
  horizon         mse
1  6 days 0.007967833
2  7 days 0.009237982
3  8 days 0.010147944
4  9 days 0.012561990
5 10 days 0.016915091
6 11 days 0.022698899
```

Рисунок 10 – Середньоквадратична помилка прогнозу моделі M3

Як видно з отриманого результату, перше усереднене значення MSE доводиться на 6-й день прогнозного горизонту, оскільки довжина цього горизонту для моделі M3 становить 60 днів, а 6 - це 10% від цієї довжини (розмір ковзаючого вікна, що задається аргументом rolling_window).

Якщо аргументу rolling_window привласнити значення 0, то запитувані метрики якості будуть розраховані для кожної дати прогнозного горизонту (тобто розмір ковзного вікна в даному випадку фактично дорівнює 1):

```
performance_metrics(M3_cv, metrics = "mse",
                    rolling_window = 0) %>% head()
```

```
> performance_metrics(M3_cv, metrics = "mse",
+                     rolling_window = 0) %>% head()
  horizon         mse
1  1 days 0.005491177
2  2 days 0.008038046
3  3 days 0.007180944
4  4 days 0.007916006
5  5 days 0.009528591
6  6 days 0.009652236
```

Рисунок 11 – Середньоквадратична помилка прогнозу моделі M3 для кожної дати прогнозного горизонту

Якщо ж аргументу rolling_window привласнити значення 1, то запитувані метрики якості будуть усереднені по усьому прогнозному горизонту:

```
performance_metrics(M3_cv, metrics = "mse",
                    rolling_window = 1) %>% head()
```

```
> performance_metrics(M3_cv, metrics = "mse",
+                     rolling_window = 1) %>% head()
  horizon       mse
1 60 days 0.2390802
```

Рисунок 12 – Середньоквадратична помилка прогнозу моделі M3 усереднені по усьому прогнозного горизонту

Метрики якості моделей, отримані в ході перехресної перевірки, можна візуалізувати за допомогою функції plot_cross_validation_metric(), яка має такі аргументи:

• df_cv - таблиця, отримана за допомогою функції cross_validation ();

• metric - назва метрики;

• rolling_window - розмір "ковзного вікна", в межах якого відбувається усереднення метрики.

Функція plot_cross_validation_metric () повертає об'єкт класу ggplot:

plot_cross_validation_metric(M3_cv, metric = "mse",rolling_window = 0.1)

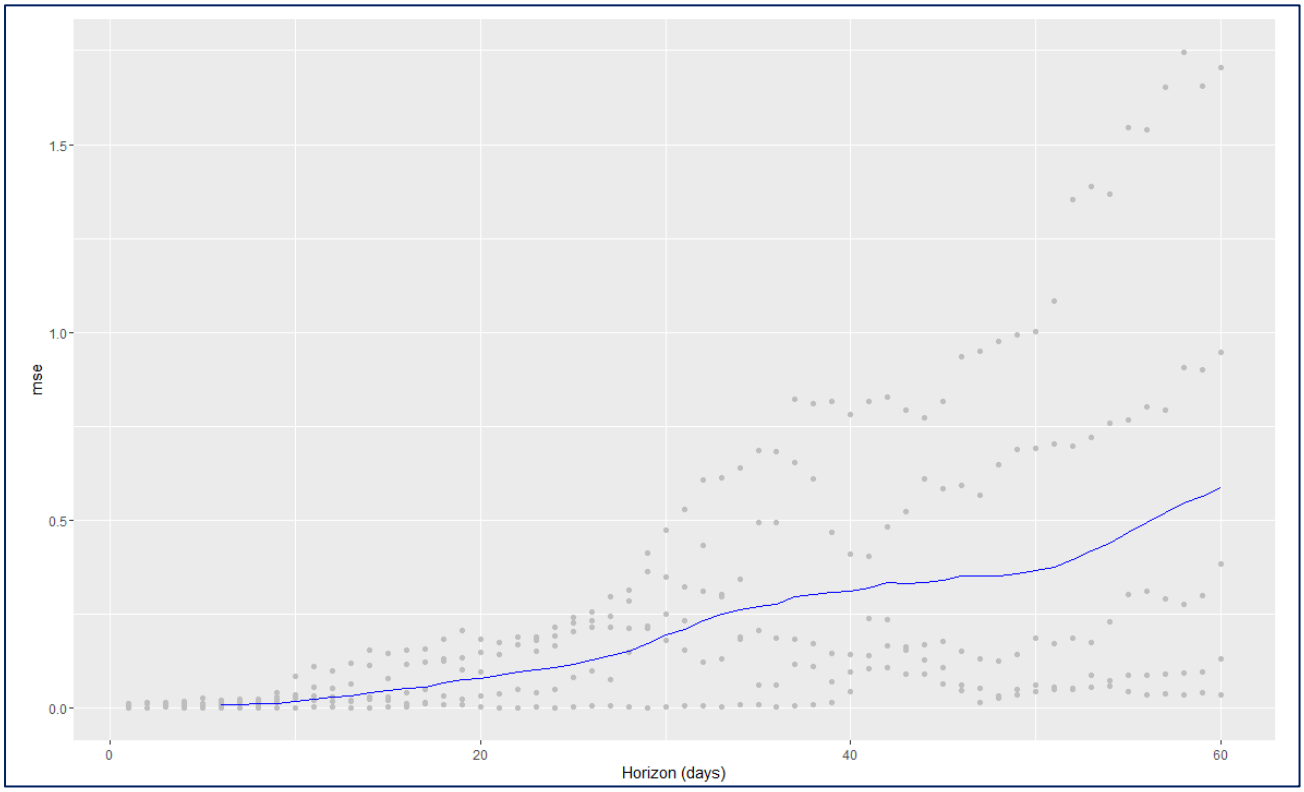


Рисунок 13 – Візуалізація метрики якості передбачень (MSE), отриманої за результатами перехресної перевірки моделі M3

Висновок: На рис. 13 наведені оцінки MSE для кожної з дат прогнозного горизонту (H = 60) кожного з K = 5 блоків даних, які брали участь в перехресній перевірці. Блакитна лінія відповідає усередненим значенням в межах кожного ковзного вікна розміром в 6 спостережень. Судячи з великого розкиду отриманих оцінок MSE, якість моделі M3 не найкраще (особливо це видно наприкінці моделі).

Вибір оптимальної моделі

Застосуємо методологію виконання перехресної перевірки для вибору оптимальної моделі з декількох альтернативних. Припустимо, що перед нами стоїть завдання вибрати оптимальну модель вартості stellar з побудованих раніше моделей M4, M12 і M15. Для опису якості цих моделей скористаємося всіма метриками: MSE, RMSE, MAE, MAPE і покриттям. Для спрощення прикладу припустимо також, що нас цікавить якість прогнозів в цілому для 60-денного прогнозного горизонту (тобто нам нецікаві окремі дати цього горизонту). Розрахуємо обидві метрики якості для кожної з моделей-кандидатів:

```
M4_cv <- cross_validation(M4, initial = 280,period = 120,horizon = 60,units = "days")
M12_cv <- cross_validation(M12, initial = 280,period = 120,horizon = 60,units = "days")
M15_cv <- cross_validation(M15, initial = 280,period = 120,horizon = 60,units = "days")

M4_perf <- performance_metrics(M4_cv,metrics = c("mse", "rmse", "mae", "mape", "coverage"), rolling_window = 1)
M12_perf <- performance_metrics(M12_cv,metrics = c("mse", "rmse", "mae", "mape", "coverage"),rolling_window = 1)
M15_perf <- performance_metrics(M15_cv,metrics = c("mse", "rmse", "mae", "mape", "coverage"),rolling_window = 1)
```

M4_perf

M12_perf

M15_perf

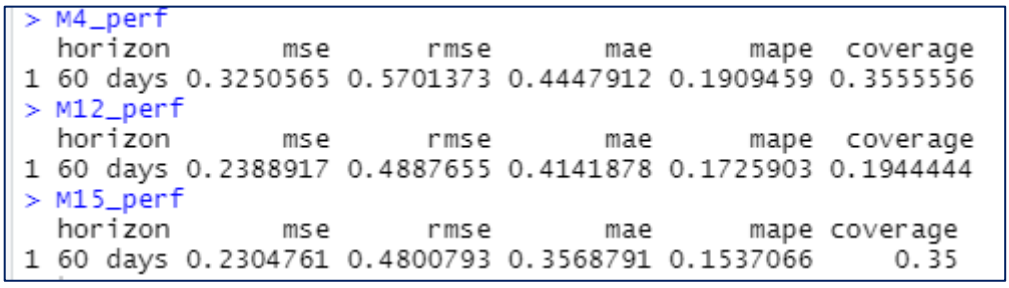


Рисунок 14 – Значення метрик якості прогнозу для моделей M4_perf, M12_perf та M15_perf

	horizon	mse	rmse	mae	mape	Coverage
M10	60 days	0.196	0.443	0.317	0.121	0.322
M14		0.276	0.526	0.457	0.181	0.205
M15		0.177	0.42	0.302	0.115	0.3

Як бачимо, M15 краще за інших моделей по всіх вибраних метрик якості. Це можна бачити також з графіків, побудованих за допомогою функції plot_cross_validation_metric() (рис. 15):

```
M4_cv_plot <- plot_cross_validation_metric(M4_cv,metric = "mape",rolling_window = 0.1) +
  ylim(c(0, 0.15)) + ggtitle("M4")
M12_cv_plot <- plot_cross_validation_metric(M12_cv,metric = "mape",rolling_window = 0.1) +
  ylim(c(0, 0.15)) + ggtitle("M12")
M15_cv_plot <- plot_cross_validation_metric(M15_cv,metric = "mape",rolling_window = 0.1) +
  ylim(c(0, 0.15)) + ggtitle("M15")

gridExtra::grid.arrange(M4_cv_plot, M12_cv_plot, M15_cv_plot, ncol = 3)
```

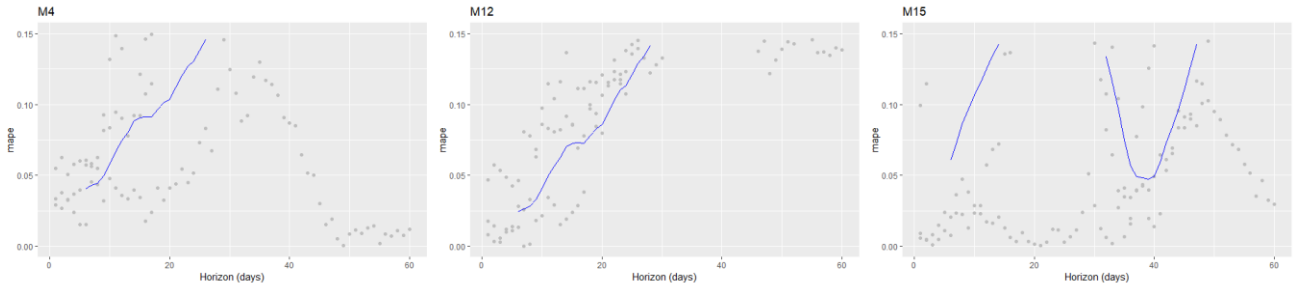


Рисунок 15 – Порівняння якості передбачень трьох моделей по метриці MAPE

Висновок: З рисунку 15 видно, що якість моделей недостатньо задовільна, скоріш за все із за малої кількості даних.

До сих пір ми будували всі моделі по навчальних даних з таблиці stellar _train. Однак у нас є і перевірки набір даних - stellar _test. Подивимося, як обрана оптимальна модель M15 спрацює на цій перевірочної вибірці. На рис. 16 представлені навчальні дані і істинні значення вартості stellar в прогнозному періоді. Блакитна суцільна лінія на цьому графіку відповідає передбаченим моделлю значень, а світло-блакитна смуга навколо неї - 80% -ної довірчою області передбачених значень:

```
plot(M15, forecast_M15) +
 coord_cartesian(xlim = c(as.POSIXct("2019-01-01"), as.POSIXct("2020-01-01"))) +
 geom_point(data = stellar_test, aes(as.POSIXct(ds), y), col = "red")
```

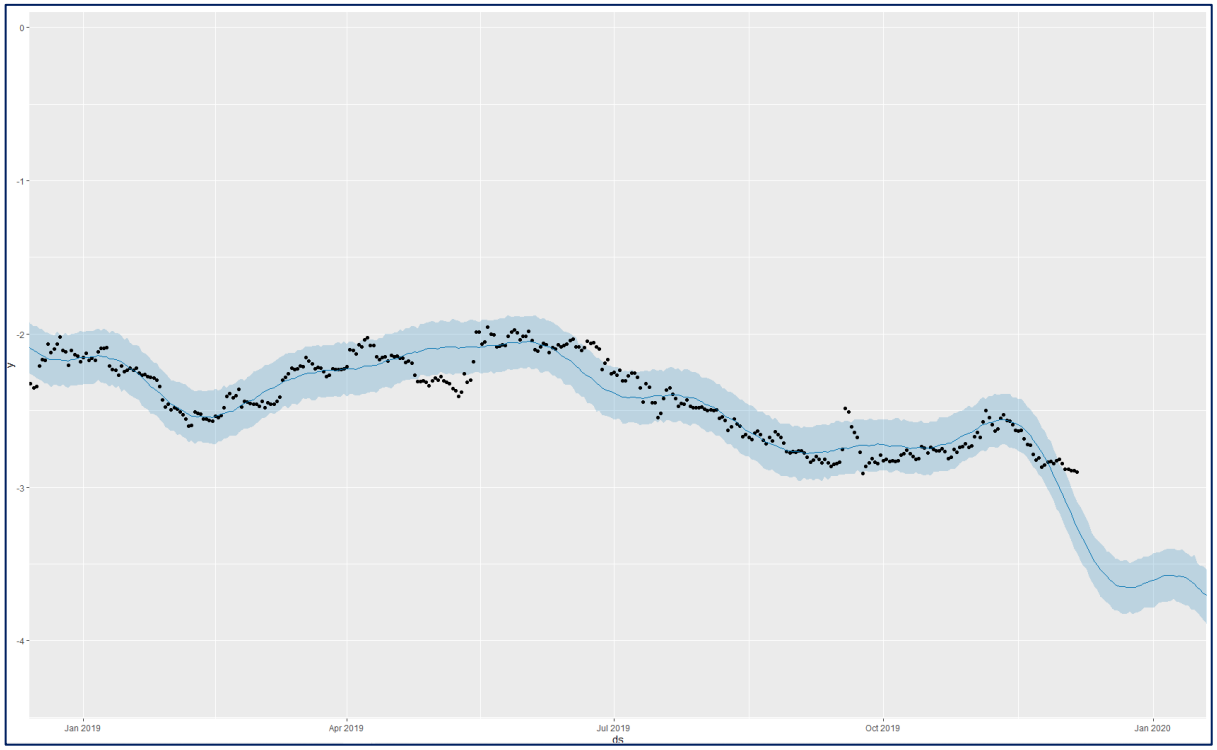


Рисунок 16 – Порівняння дійсних значень вартості stellar і прогнозних значень, отриманих за допомогою моделі M15

Висновок: Хоча обрана в якості оптимальної модель M15 не змогла вірно передбачити деякі локальні коливання вартості stellar в прогнозному періоді, в цілому вона недостатньо задовільний результат: більшість справжніх значень вартості виявилося в межах 80%-ої довірчою смуги.

ЛАБОРАТОРНА РОБОТА 16-17

Тема: Виявлення структурних змін у часовому ряді.

Пояснення

Під час аналізу часових рядів важливим аспектом є своєчасне виявлення та оповіщення про нестандартні структурні зміни. Наприклад, передаварійних і аварійних ситуацій. Зазвичай це пов'язано з істотними змінами параметрів, що описують часовий ряд. Одним із типів таких змін є зсув середнього рівня часового ряду, який може бути як дуже різким, так і поступовим (рис. 1). Існує кілька методів, що дають змогу автоматично виявляти подібні структурні зміни в часових рядах.

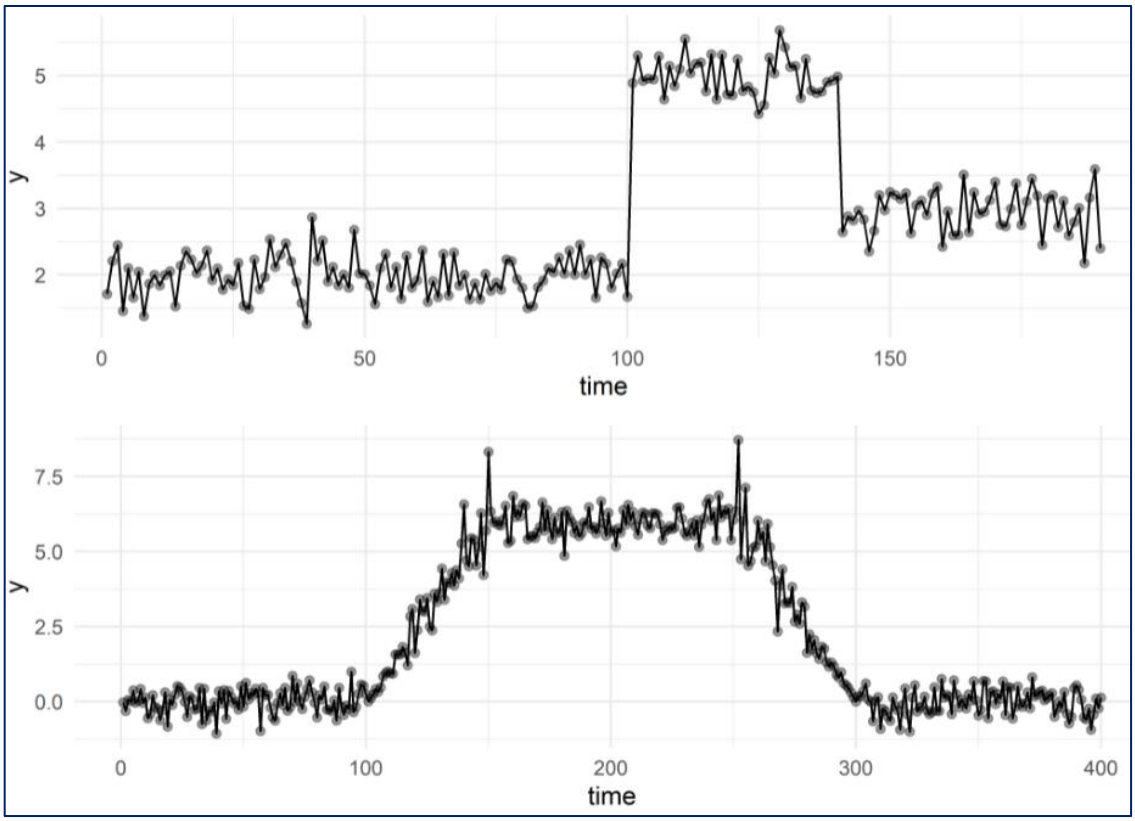


Рисунок 1 – Приклади структурних змін у часових рядах: різка (вгорі) і плавна (внизу)

Роботу із застосування методів виявлення структурних змін у часових рядах почніть зі встановлення пакета BreakoutDetection. Його можна встановити тільки з репозиторію GitHub, для чого достатньо скористатися

функцією install_github() з пакета devtools (який, звісно, теж потрібно спочатку встановити, якщо у вас його ще немає).:

```
install.packages("devtools")
devtools::install_github("twitter/BreakoutDetection")
```

Метод “E–Divisive with Medians” (EDM)

Цей метод ґрунтується на використанні т.зв. E-статистики (від "energy", тобто "енергія"), яка характеризує відстань між спостереженнями з випадкових вибірок. Що більша ця статистика, то більше підстав вважати, що порівнювані вибірки походять із різних генеральних сукупностей. Щоб уникнути впливу звичайних на практиці аномальних спостережень, метод EDM оперує не середніми значеннями, а медіанами.

Застосуємо функцію breakout() з прийнятими за замовчуванням параметрами до даних за вартістю stellar

```
require(BreakoutDetection)

BO0 <- breakout(log(stellar_price$y), plot = TRUE, ylab = "y")
BO0$plot
```

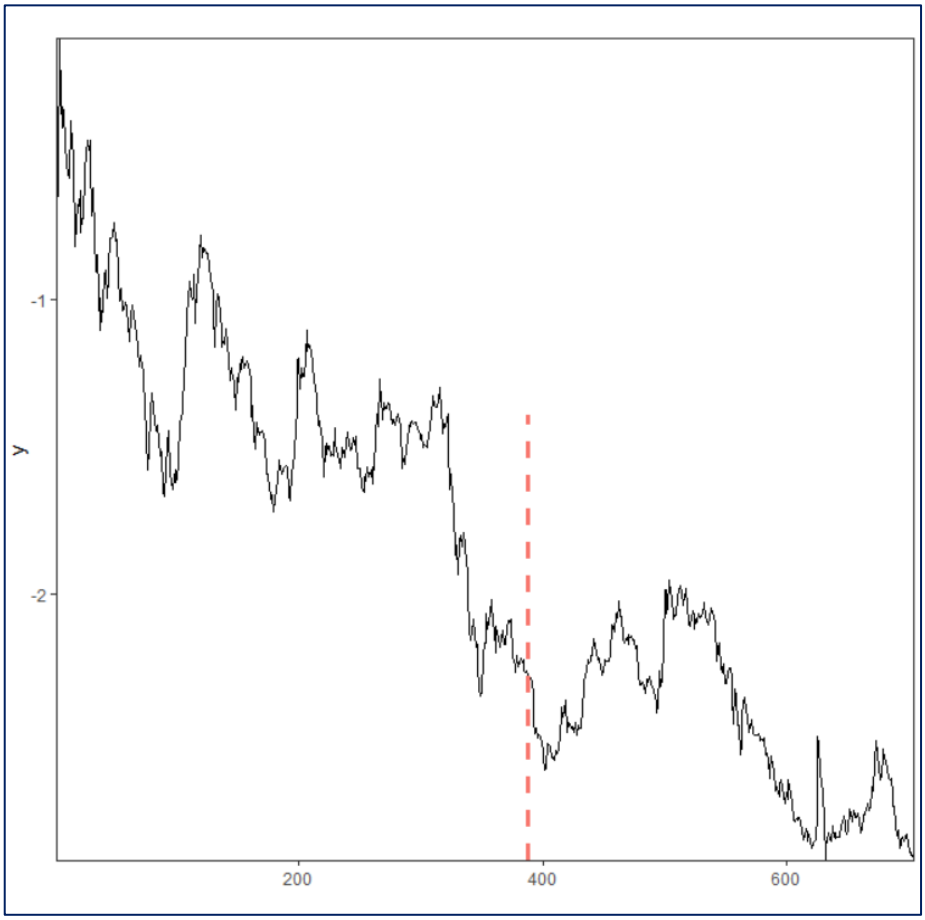


Рисунок 2 – Точка зламу тренда, виявлена в часовому ряду вартості stellar за допомогою моделі BO0

Висновок. Як видно на рис. 2, метод EDM виявив одну єдину точку зламу тренду (позначена вертикальною переривчастою лінією), що зумовлено прийнятим за замовчуванням аргументом **method = "amoc".**

Для оцінювання статистичної значущості цього зсуву в часовому ряду скористаємося аргументом nperm. Виконаємо оцінювання за допомогою перестановочного тесту з 1000 ітерацій:

BO0_perm <- breakout(log(stellar_price$y), nperm = 1000)

Об'єкт BO0_perm являє собою список із такими елементами:

```
> BO0_perm
$loc
[1] 388

$stat
[1] 24.44802

$time
[1] 8.08

$pval
[1] 0.000999001
```

де loc - це положення точки зламу (тобто її порядковий номер у часовому ряду), stat - значення E-критерію, time -

час виконання команди (сек.), a pval - p-значення, отримане за допомогою перестановочного тесту.

Видно, що виявлений зсув рівня часового ряду виявився статистично значущим (на рівні значущості 0.05)!

Запросимо тепер знаходження декількох точок зламу (рис. 3):

```
BO1 <- breakout(log(stellar_price$y), method = "multi",
                plot = TRUE, ylab = "y")
BO1$plot
```

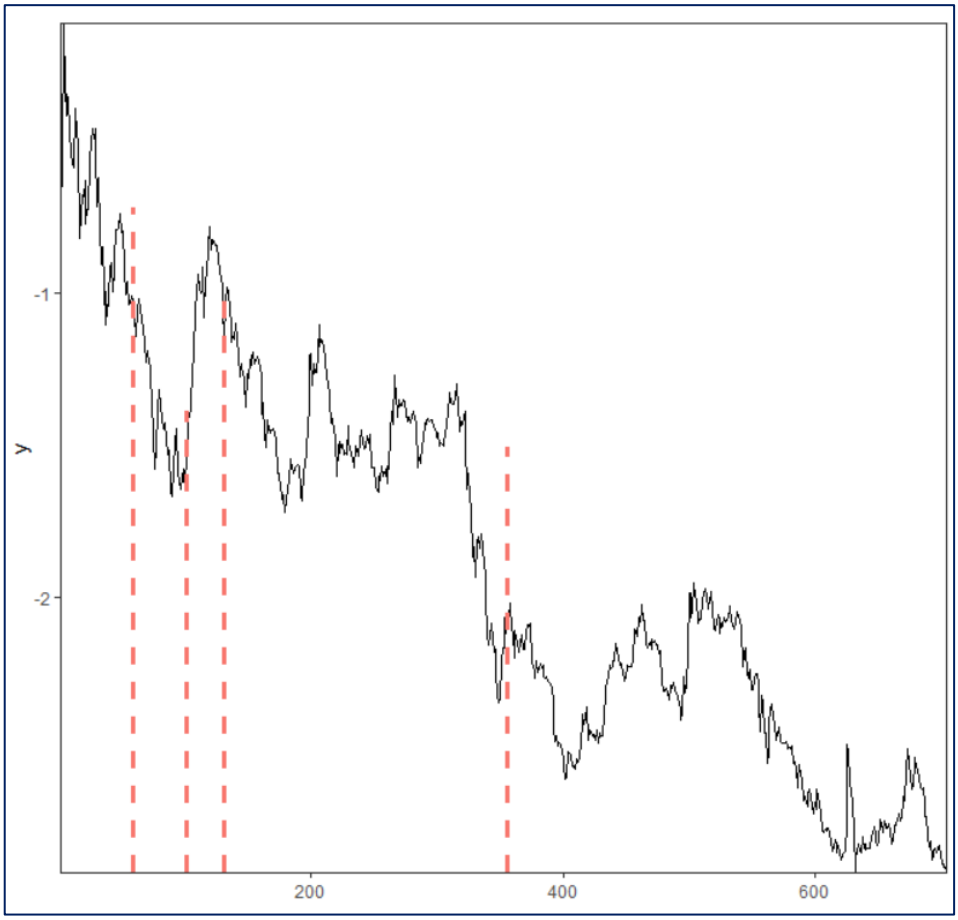


Рисунок 3 – Точки зламу тренда, виявлені в часовому ряду вартості stellar за допомогою моделі BO1

Розглянемо вплив параметрів регуляризації. Спочатку вимкнемо регуляризацію за допомогою параметра degree = 0 (рис.4):

```
BO2 <- breakout(log(stellar_price$y), method = "multi", degree = 0,
                plot = TRUE, ylab = "y")
BO2$plot
```

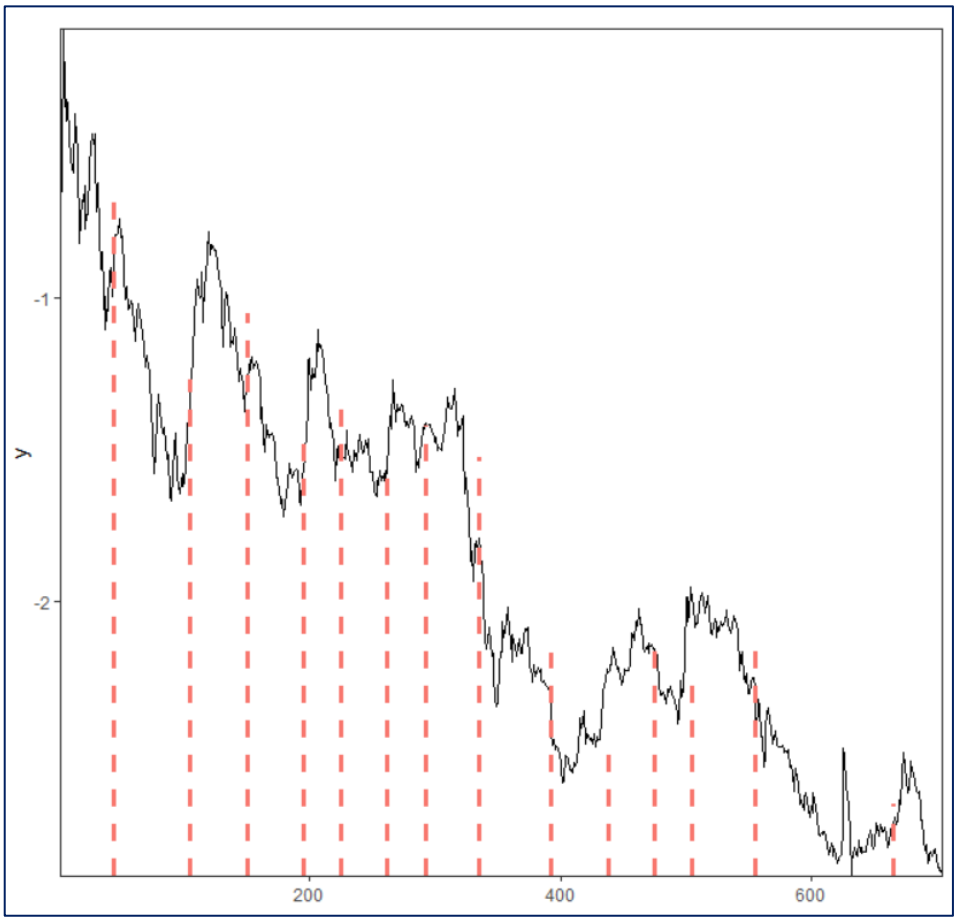


Рисунок 4 – Точки зламу тренда, виявлені в часовому ряду вартості stellar за допомогою моделі BO2

Порядкові номери спостережень, що відповідають виявленим точкам зламу, зберігаються в елементі loc одержуваного об'єкта-списку loc:

```
> BO2$loc
 [1]  43 104 150 195 225 262 293 335 393 438 475 505 556 666
```

Схожого ефекту (збільшення кількості виявлених точок зламу) можна домогтися також за допомогою низького значення параметра percent (рис.5):

```
BO3 <- breakout(log(stellar_price$y), method = "multi", percent = 0.05,
        plot = TRUE, ylab = "y")

BO3$plot
```

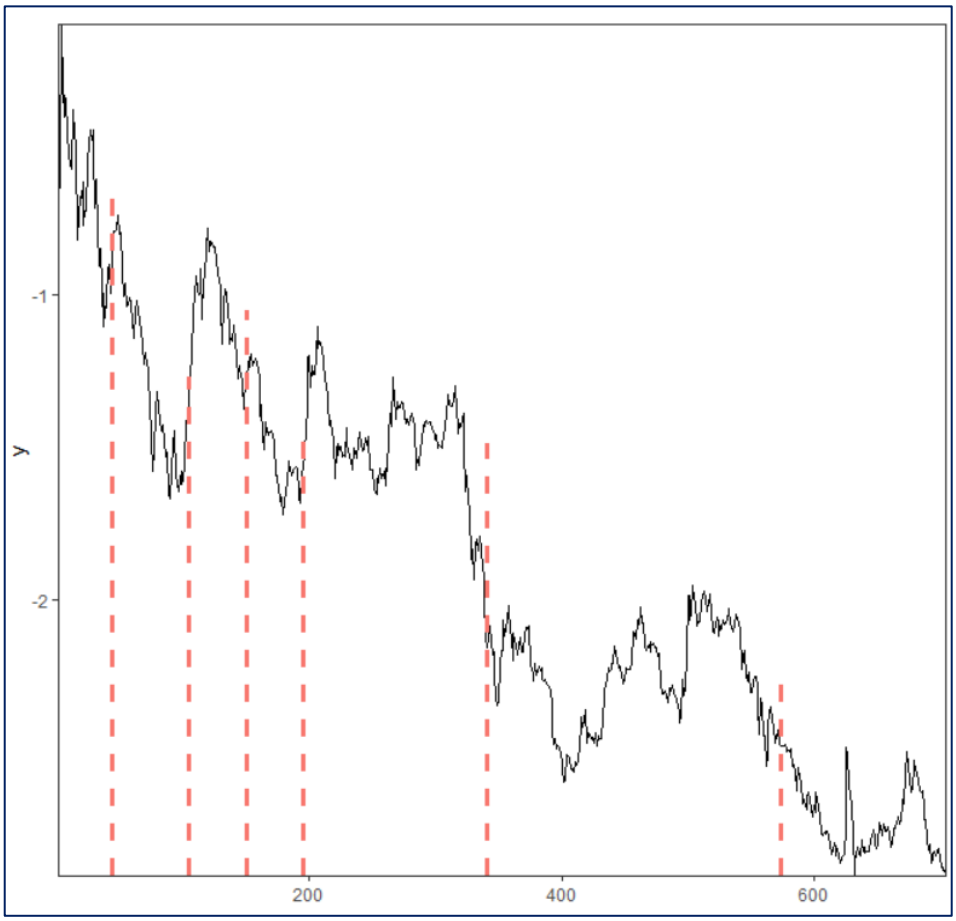


Рисунок 4 – Точки зламу тренда, виявлені в часовому ряду вартості stellar за допомогою моделі BO3

Нарешті, зниження параметра beta теж призведе до збільшення числа виявлених алгоритмом точок зламу (рис. 5):

```
BO4 <- breakout(log(stellar_price$y), method = "multi", beta = 0.0001,
                plot = TRUE, ylab = "y")
BO4$plot
```

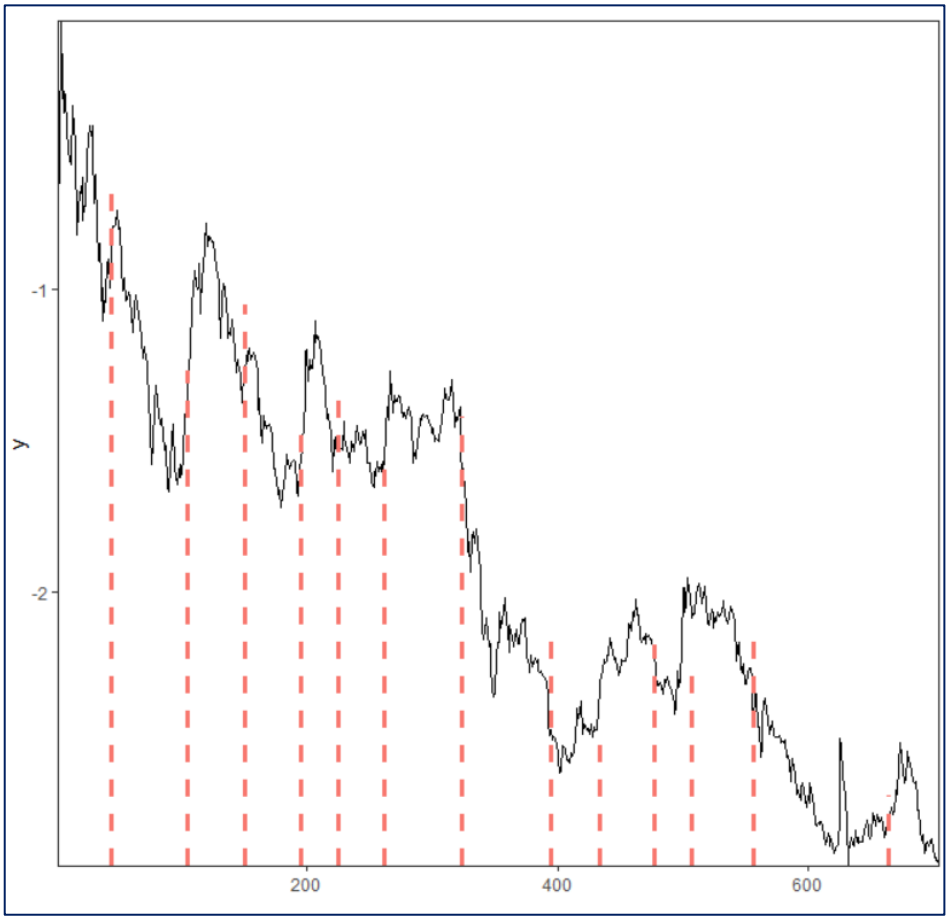


Рисунок 5 – Точки зламу тренда, виявлені в часовому ряду вартості stellar за допомогою моделі BO4

Висновок: У роботі було реалізовано один із найефективніших методів виявлення структурних змін у часових рядах - "E-Divisive with Medians". Вивчено можливості параметрів функції функції breakout() з пакета BreakoutDetection, у якій реалізовано метод EDM.

ЛАБОРАТОРНА РОБОТА 18-19

Тема: Виявлення аномалій в часових рядах.

Пояснення

Необхідність виявлення незвичайних спостережень (викидів, або аномалій) у часових рядах часто виникає в таких ситуаціях, як моніторинг стану обладнання, відстеження несподіваних коливань на ринку цінних паперів, облік показників стану здоров'я пацієнтів тощо. Один з інструментів для розв'язання подібних завдань - пакет anomalize.

Автоматичне виявлення аномалій (параметри функцій встановлені за замовчуванням)

Спробуємо виявити незвичайні спостереження в часовому ряду за вартістю криптовалюти stellar. Для цього до таблиці з даними послідовно застосуємо команди за участю таких функцій з пакета anomalize:

- time_decompose() — виконує декомпозицію часового ряду на окремі складові (сезонну, тренд і залишки);
- anomalize() — застосовує до залишків один із двох реалізованих у пакеті методів виявлення аномалій;
- time_recompose() — відновлює вихідний часовий ряд, паралельно обчислюючи верхню і нижню межі діапазону, до якого входять "нормальні" спостереження.

```
result_stellar <- stellar_price %>%
  time_decompose(y, merge = TRUE) %>%
  anomalize(remainder) %>%
  time_recompose()
```

```
result_stellar   705 obs. of 11 variables
 y : num [1:705] 0.48 0.565 0.896 0.724 0.663 ...
 ds : Date[1:705], format: "2018-01-01" "2018-01-02" "2018-01-03" "2018-01-04" ...
 observed : num [1:705] 0.48 0.565 0.896 0.724 0.663 ...
 season : num [1:705] 4.68e-04 7.03e-04 1.11e-04 -6.95e-04 -6.18e-05 ...
 trend : num [1:705] 0.579 0.575 0.571 0.567 0.563 ...
 remainder : num [1:705] -0.0997 -0.0112 0.3249 0.1575 0.0996 ...
 remainder_l1 : num [1:705] -0.0717 -0.0717 -0.0717 -0.0717 -0.0717 ...
 remainder_l2 : num [1:705] 0.0736 0.0736 0.0736 0.0736 0.0736 ...
 anomaly : Named chr [1:705] "Yes" "No" "Yes" "Yes" ...
 ..- attr(*, "names")= chr [1:705] "25%" "25%" "25%" "25%" ...
 recomposed_l1: num [1:705] 0.508 0.504 0.5 0.495 0.491 ...
 recomposed_l2: num [1:705] 0.653 0.65 0.645 0.64 0.637 ...
```

У результаті виконання наведених вище команд отримали:

1. Результатом застосування time_decompose() до вихідних даних став розрахунок 4х стовпців:

- **observed** (вихідні спостереження модельованої змінної),
- **season** (сезонна складова часового ряду),
- **trend** (тренд) та
- **remainder** (залишки). (Параметр merge = TRUE призвів до того, що ці нові стовпці були додані до вихідних даних).

2. Функція anomalize() була застосована до стовбця remainder і розрахувала 3 нові стовпчики:

- **remainder_l1**;
- **remainder_l2** (нижня і верхня межі, за межами яких спостереження вважаються аномаліями);
- змінну **anomaly**, яка набуває значення "No", якщо спостереження не є викидом, і "Yes" якщо є.

3. Функція time_recompose() відновила вихідний часовий ряд із використанням значень із season, trend, remainder_l1 і remainder_l2 і принагідно розрахувала два нові стовпчики: recomposed_l1 і recomposed_l2, які відповідають верхній і нижній межам діапазону "нормальних" спостережень..

Представимо результати графічно. Функція plot_anomaly_decomposition() зображує компоненти тимчасового ряду, які розраховує функція

time_decompose(). На одержуваному графіку аномалії показані у вигляді обведених колом червоних точок (рис.1):

result_stellar %>% plot_anomaly_decomposition()

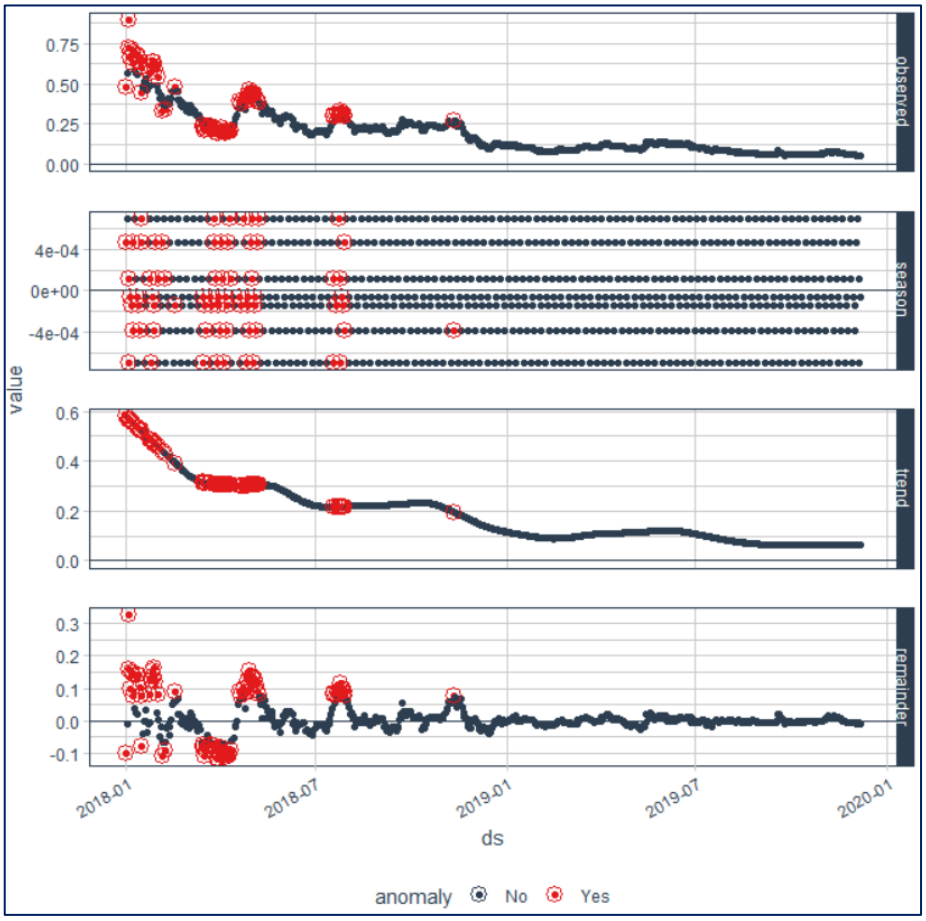


Рисунок 1 — Результат автоматичного виявлення аномалій у часовому ряду, виконаного за допомогою пакета anomalize

Функція - plot_anomalies() - дає змогу окремо зобразити вихідний часовий ряд і виявлені в ньому незвичайні спостереження (рис.2):

result_stellar %>% plot_anomalies()

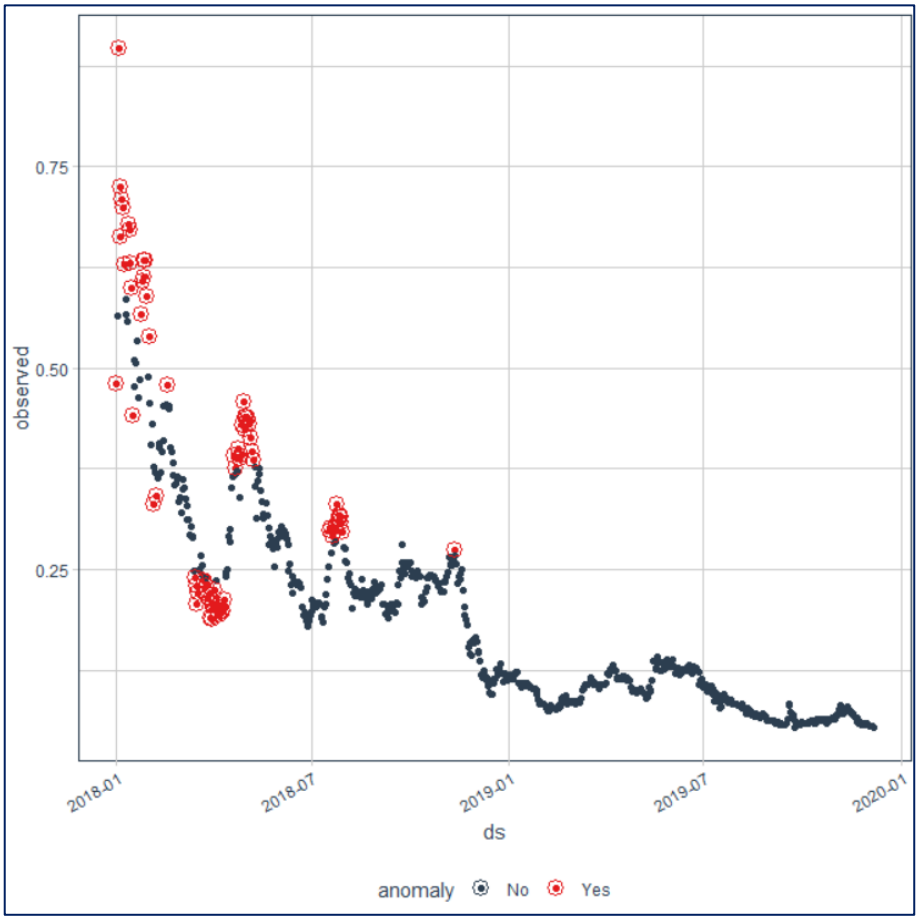


Рисунок 2 — Аномальні спостереження, автоматично виявлені в часовому ряду за допомогою пакета anomalize

Оскільки plot_anomalies() повертає графічний об'єкт класу ggplot, то можна додати до нього інші ggplot-елементи. Додамо лінію, що послідовно з'єднує всі точки (рис.3):

```
result_stellar %>%
  plot_anomalies() +
  geom_line()
```

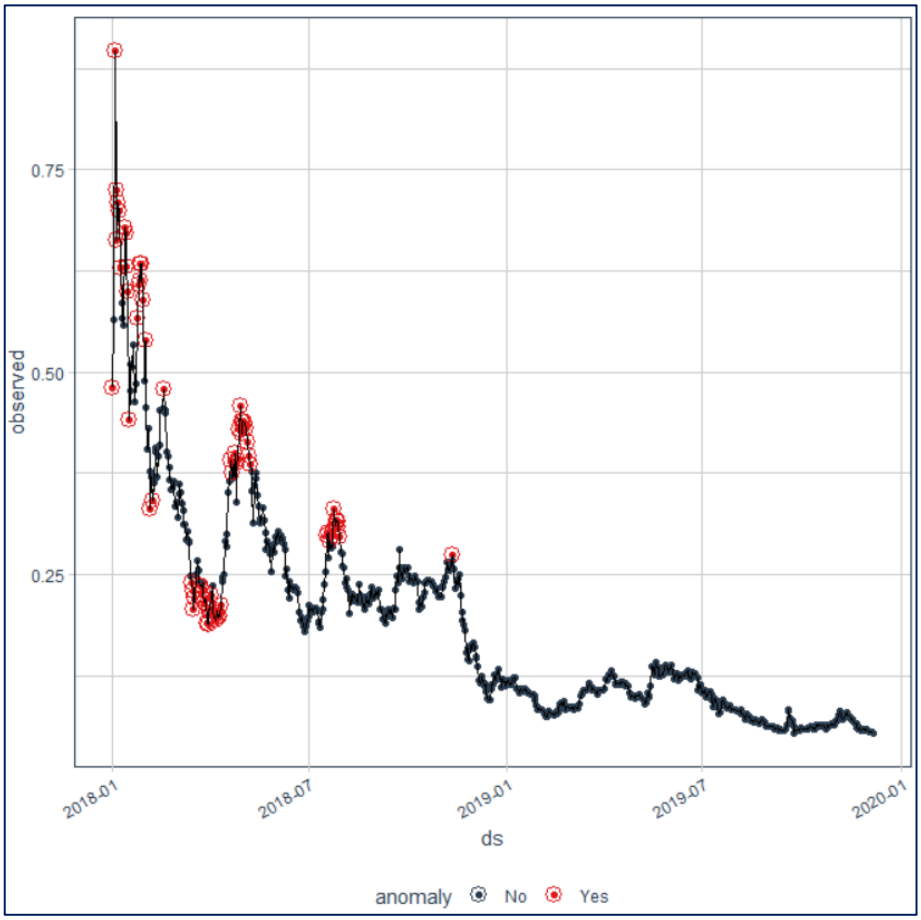


Рисунок 3 — Приклад поєднання функції plot_anomalies() з пакета anomalize і функції geom_line() з пакета ggplot2

Ручне налаштування параметрів для виявлення аномалій

При створенні об'єкта result_stellar всі обчислення були виконані з використанням налаштувань, заданих у пакеті anomalize за замовчуванням. Часто для отримання прийнятного результату цього буде достатньо. Однак, залежно від властивостей аналізованого часового ряду, знадобиться більш тонке налаштування відповідних параметрів.

Розглянемо спочатку параметри функції time_decompose(), яка виконує розкладання ряду на окремі компоненти. Результат роботи цієї функції залежить від значень таких її аргументів:

- frequency - визначає "частоту" сезонної компоненти, тобто кількість спостережень, що входять в один сезонний цикл (наприклад, для денних даних цей аргумент за замовчуванням дорівнюватиме "1 week", тобто одному тижню);

- trend - визначає кількість спостережень в окремих відрізках часового ряду, що використовуються для розрахунку тренда (наприклад, для денних даних за замовчуванням trend = "3 months", тобто три місяці);

- method - метод, який використовується для розкладання ряду на компоненти. У цього аргумента є два можливих значення: "stl" (прийнято за замовчуванням) і "twitter".

За замовчуванням аргументи frequency і trend приймають значення "auto" (автоматичний режим). У цьому разі для вибору конкретних значень frequency і trend, що підходять для ситуації, функція time_decompose() звертається до іншої функції - get_time_scale_template(), яка повертає "довідник" типових значень:

get_time_scale_template()

```
# A tibble: 8 x 3
  time_scale frequency trend
  <chr>      <chr>     <chr>
1 second     1 hour    12 hours
2 minute     1 day     14 days
3 hour       1 day     1 month
4 day        1 week    3 months
5 week       1 quarter 1 year
6 month      1 year    5 years
7 quarter    1 year    10 years
8 year       5 years   30 years
```

Висновок. Видно, що у випадку з даними, де проміжок між спостереженнями виражається в секундах, параметри frequency і trend виражатимуться в годинах і за замовчуванням становитимуть "1 hour" (1 година) і "12 hours" (12 годин) відповідно. Слід зазначити, що залежно від інтервалу між спостереженнями і довжини самого часового ряду, конкретні обрані програмою значення frequency і trend необов'язково виявляться рівними наведеним вище значенням із "довідника". Саме це сталося вище при створенні об'єкта result_stellar (frequency = 2 hours, trend = 61 hours). Крім строкових значень ("1 hour", "2 days", "2 weeks" тощо) обом параметрам можна привласнювати і числові значення. Це більш зручний підхід.

Зазначимо також, що "шаблонні" значення параметрів frequency і trend можна змінити глобально за допомогою функції set_time_scale_template().

Для скасування автоматичного вибору значень frequency і trend достатньо вказати бажані значення під час виклику функції time_decompose(). У наведеному нижче прикладі присвоїмо цим параметрам значення 2 і 6 (годин) відповідно:

```
stellar_price %>%
  time_decompose(y, frequency = 2, trend = 6) %>%
  anomalize(remainder) %>%
  time_recompose() %>%
  plot_anomaly_decomposition()
```

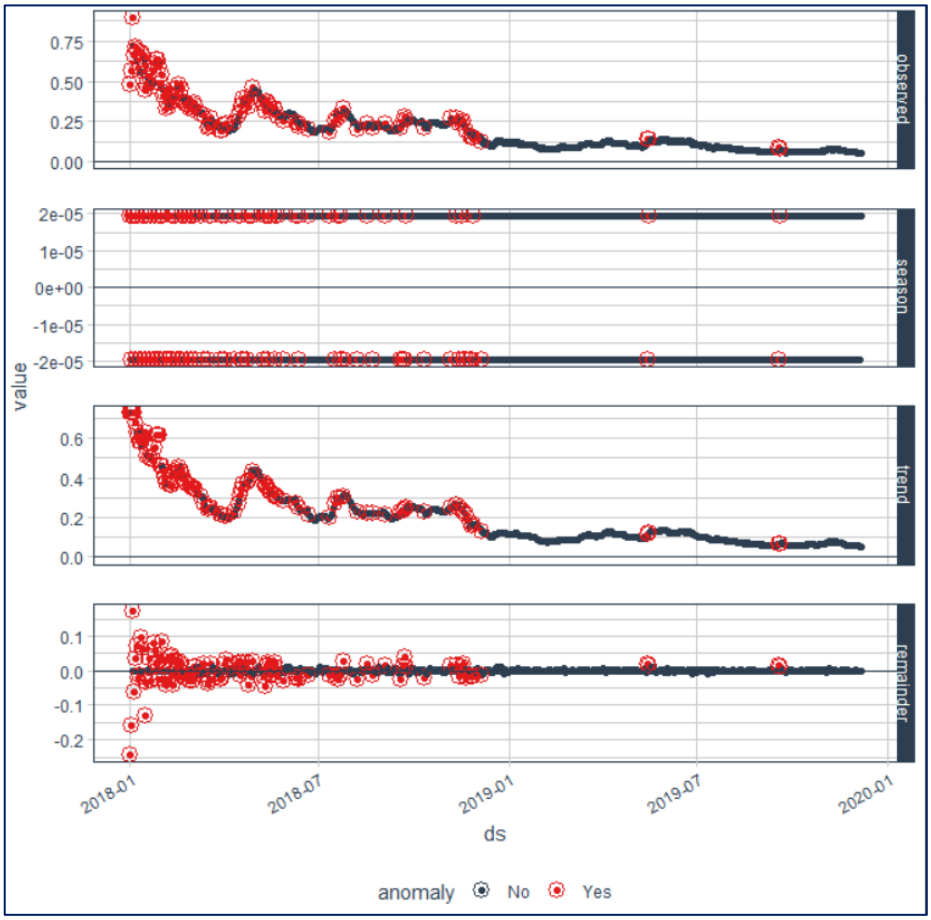


Рисунок 4 — Результат виявлення аномалій у часовому ряду, виконаного за допомогою користувацьких налаштувань функції time_decompose()

Висновок. Як видно на рис.4, зменшення параметрів frequency і trend призвело до класифікації більшої кількості спостережень як аномалій. Це звичайний результат у таких випадках, зумовлений перенавчанням Loess-моделі тренду (зверніть увагу на те, якою звивистою стала крива тренду на рис.4 порівняно з рис.1). На відміну від Loess, другий метод розкладання часових рядів - "twitter" - більш стійкий до локальних коливань і тому за тих самих значень frequency і trend зазвичай відносить до аномалій менше спостережень. Це легко перевірити, порівнюючи рис.4 і рис.5):

```
stellar_price %>%
  time_decompose(y, frequency = 2,
                 trend = 6, method = "twitter") %>%
  anomalize(remainder) %>%
  time_recompose() %>%
  plot_anomaly_decomposition()
```

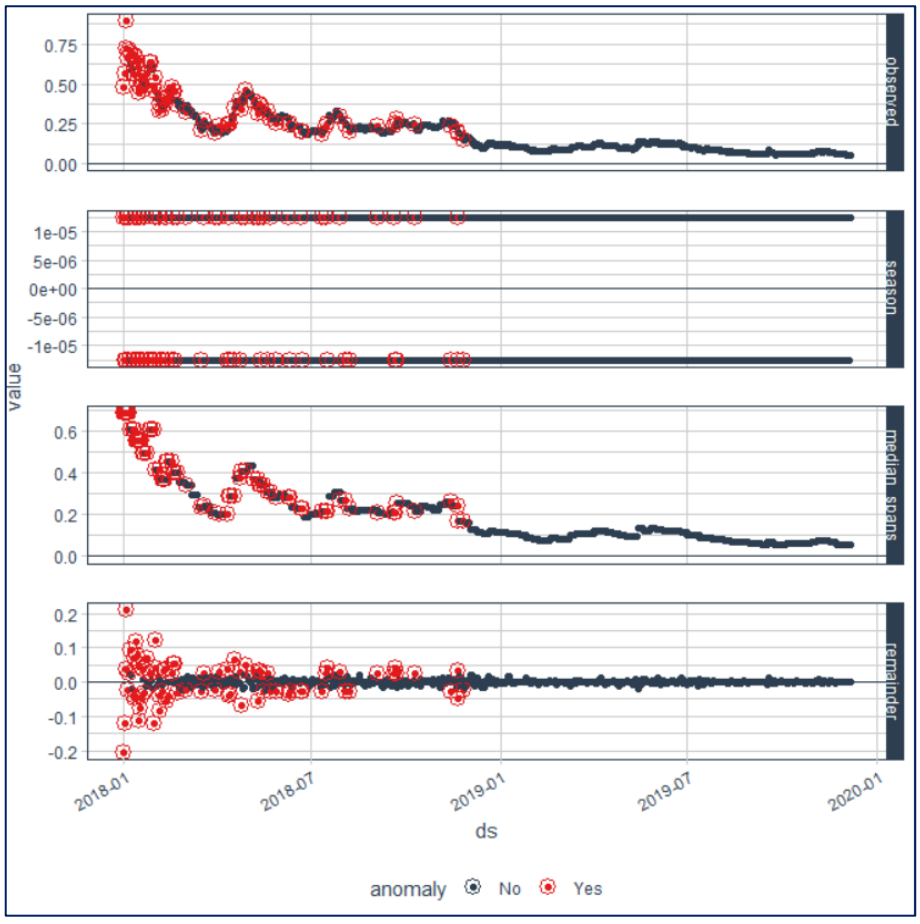


Рисунок 5 — Результат виявлення аномалій у часовому ряду, виконаного за методом twitter

Функція anomalize дає змогу вибрати один із двох методів виявлення аномальних спостережень, а також налаштувати чутливість цих методів. Для цього слугують такі аргументи функції:

- method - приймає два можливих значення відповідно до назви методу виявлення аномалій: "iqr" (прийнято за замовчуванням) і "gesd". Метод IQR (від "interquartile range", тобто "інтерквартильний розмах") працює швидше, але не такий точний, як GESD (Generalized Extreme Studentized Deviate test).
- alpha - контролює ширину діапазону "нормальних" значень (за замовчуванням дорівнює 0.05). Що меншим є цей параметр, то ширшим є діапазон значень, які розглядаються як "нормальні", і то менше спостережень буде класифіковано як аномальні.
- max_anoms - максимальна частка спостережень, які дозволено віднести до аномалій (0.2 за замовчуванням, тобто 20%).

Розглянемо вплив параметра alpha, знизивши його значення з прийнятого за замовчуванням 0.05 до 0.025. При цьому залишимо параметри frequency і trend функції time_decompose() такими самими, як на рис.4:

```
stellar_price %>%
  time_decompose(y,
                 frequency = 2,
                 trend = 6,
                 method = "twitter") %>%
  anomalize(remainder, alpha = 0.025) %>%
  time_recompose() %>%
  plot_anomaly_decomposition()
```

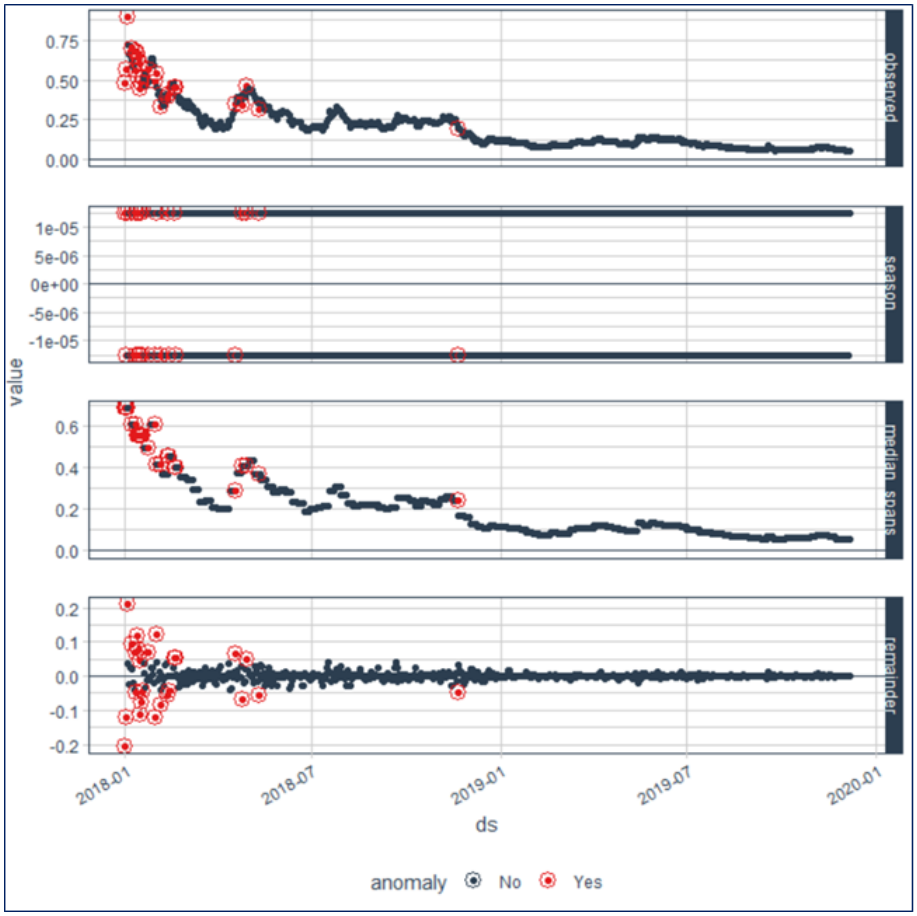


Рисунок 6 — Результат виявлення аномалій у часовому ряду, виконаного зі зниженим значенням параметра alpha функції anomalize()

Висновок. Як видно на рис.1, незважаючи на перенавчання Loess-моделі тренду, що все ще має місце, до класу аномальних було віднесено набагато меншу кількість спостережень.

Аналогічного ефекту можна було б досягти також шляхом зниження параметра max_anoms функції anomalize().

ЛАБОРАТОРНА РОБОТА 20-21

Тема: Побудова власної штучної нейронної мережі з сигмоїдною функцією активації.

Приклад виконання

Нижче наведені приклади функцій, що можуть бути використані при виконанні завдань.

```
# Activation Function
sigmoid <- function(x) {

  z = (1 / (1 + exp(-x)))

  return(z)

}
```

Рисунок 1 – Функція активації

```
# Defining the derivative of our acti
derivativeActivation <- function(x) {

  g = (sigmoid(x) * (1 - sigmoid(x)))

  return(g)

}
```

Рисунок 2 – Похідна функції активаці

```r
# Function to obtain predicted outputs
feedForward <- function(x, w1, w2, activation) {

  output <- rep(0, length(x))

  for (i in 1:length(x)) {

    a1 = w1 %*% matrix(rbind(1, x[i]), ncol=1)

    z1 = activation(a1)

    a2 = w2 %*% matrix(rbind(1, z1), ncol=1)

    output[i] = a2
  }


  return(output)
}
```

Рисунок 3 – Функція прямого розповсюдження

```r
#Function for computing the gradients
backPropagation <- function(x, y, w1, w2, activation, derivativeActivation) {

  preds <- feedForward(x, w1, w2, activation) #predicted values

  derivCost <- -2*(y - preds) #Derivative of the cost function (first term)

  dW1 <- matrix(0,ncol=2,nrow=nrow(w1)) #Gradient for w1
  dW2 <- matrix(rep(0,length(x)*(dim(w2)[2])),nrow=length(x)) #Gradient matrix for w2

  # Computing the Gradient for W2
  for (i in 1:length(x)) {

    a1 = w1 %*% matrix(rbind(1, x[i]), ncol=1)
    da2dW2 = matrix(rbind(1, activation(a1)), nrow=1)
    dW2[i,] = derivCost[i] * da2dW2

  }

  #Computing the gradient for W1
  for (i in 1:length(x)) {

    a1 = w1 %*% matrix(rbind(1, x[i]), ncol=1)
    da2da1 = derivativeActivation(a1) * matrix(w2[,-1], ncol=1)
    da2dW1 = da2da1 %*% matrix(rbind(1, x[i]), nrow=1)

    dW1 = dW1 + derivCost[i] * da2dW1
  }

  # Storing gradients for w1, w2 in a list
  gradient <- list(dW1, colSums(dW2))

  return (gradient)
}
```

Рисунок 4 – Функція зворотного розповсюдження

ЛАБОРАТОРНА РОБОТА 22-23

Тема: Побудова власної штучної нейронної мережі з сигмоїдною та софтмакс функціями активації.

Приклад виконання

Нижче наведені приклади функцій, що можуть бути використані при виконанні завдань.

```
sigmoid <- function(x) {
  1.0 / (1.0 + exp(-x))
}
```

Рисунок 1 – Функція активації сигмоїдна

```
sigmoid_derivative <- function(x) {
  x * (1.0 - x)
}
```

Рисунок 2 – Похідна функції активаці

```
softM <- function(x) {
  nDim = length(x)
  res = rep(0, nDim)
  res = matrix(res, nrow = nrow(x), byrow = FALSE)
  for (j in 1:ncol(x)) {
    for (i in 1:nrow(x)) {
      xDiff = x[,j] - x[i,j]
      sumExpVect = sum(exp(xDiff))
      res[i,j] = 1/sumExpVect
    }
  }
  return(res)
}
```

Рисунок 3 – Функція активації софтмакс

```
feedforward <- function(nn) {
  nn$layer1 <- sigmoid(t(nn$weights1) %*% nn$input)
  nn$output <- softM(t(nn$weights2) %*% nn$layer1)
  nn
}
```

Рисунок 4 – Функція прямого розповсюдження

```
backprop <- function(nn) {

  # application of the chain rule to find derivative of the loss function with
  # respect to weights2 and weights1
  d_weights2 <- ( (nn$y - nn$output) %*% t(nn$layer1) )

  d_weights1 <- t(nn$y - nn$output) %*%  t(nn$weights2)
  d_weights1 <- (t(d_weights1) * sigmoid_derivative(nn$layer1)) %*% t(nn$input)


  # update the weights using the derivative (slope) of the loss function
  nn$weights1 <- nn$weights1 + t(d_weights1)
  nn$weights2 <- nn$weights2 + t(d_weights2)

  nn
}
```

Рисунок 5 – Функція зворотного розповсюдження

ЛАБОРАТОРНА РОБОТА 24-26

Тема: Побудова власної штучної нейронної мережі типу RNN.

Приклад виконання

RNN розшифровується як "рекурентна нейронна мережа". У RNN ваше передбачення залежить не тільки від особливостей поточного моменту часу, але й від моментів часу, що передували поточному. Як наслідок, ШНМ найбільш корисні для моделювання закономірностей у послідовностях даних, де наступний елемент залежить від попередніх елементів у послідовності.

Ми використаємо RNN, щоб передбачити, чи буде дощ у Сіетлі, маючи інформацію про те, чи був дощ за попередні шість днів. Приблизно так:.

Рисунок 1 – Загальна схема прогнозу

Перш ніж приступити до підготовки даних та навчання моделі, ми повинні переконатися, що наше середовище налаштоване. По-перше, ми повинні переконатися, що завантажили пакет Keras R, який ми будемо використовувати для побудови нашої RNN. Я також збираюся завантажити кілька утиліт, щоб полегшити нам життя.

```
library(keras)
library(tidyverse)
library(caret)
```

Рисунок 2 – Підключення пакетів

Далі нам потрібно прочитати наші дані. Я завжди люблю дивитися на перші пару рядків нового фрейму даних, коли завантажую його, щоб переконатися, що дані виглядають розумно.

```
weather_data <- read_csv("D://download//seattleWeather_1948-2017.csv")
head(weather_data)
```

Рисунок 3 – Зчитування набору даних

Цей набір даних містить інформацію про погоду в Сіетлі за кожен день з 1 січня 1948 року по 12 грудня 2017 року. Для кожного дня ми маємо кількість опадів, максимальну та мінімальну температуру, а також те, чи були опади взагалі в цей день (ми припускаємо, що зазвичай йшов дощ, і тому колонка називається "дощ"). Давайте встановимо параметри нашої моделі, а потім візьмемося за підготовку даних.

Ми також встановимо деякі параметри для нашої моделі. Це гарна ідея - задати параметри на початку коду, а потім просто посилатися на них за іменами змінних. Таким чином, якщо ви захочете змінити їх пізніше, вам доведеться зробити це лише один раз, замість того, щоб намагатися запам'ятати кожне місце, де ви посилаєтеся на них у своєму коді, і неминуче забути хоча б одне.

Тут я встановлюю три параметри:

- Максимальна довжина послідовності, яку ми будемо розглядати, щоб спробувати передбачити наступний елемент
- Розмір партії, тобто кількість різних послідовностей, які потрібно переглянути за один раз під час навчання (таким чином, загальна кількість точок даних, які ви будете вводити в модель за один раз, дорівнює максимальній довжині партії). Зазвичай бажано, щоб це був степінь двійки (наприклад, 16, 32, 64 і т.д.), щоб полегшити собі життя в майбутньому, якщо ви захочете тренувати модель на графічному процесорі.
- Загальна кількість епох для навчання, тобто кількість разів, коли модель буде піддаватися впливу всього навчального набору (таким чином, кількість раундів навчання, які наша модель буде тренувати = (загальна кількість навчальних прикладів/розмір пакету) * загальна кількість епох навчання).

```
max_len <- 6
batch_size <- 32
total_epochs <- 15
set.seed(123)
```

Рисунок 4 – Встановлення параметрів

Ми поговоримо більше про ці параметри, коли дійдемо до відповідних етапів налаштування нашої моделі. Загалом, чим більші ваші параметри, тим довше ваша модель буде тренуватися і тим більша ймовірність того, що вона буде надмірно пристосована. З іншого боку, чим менші ваші параметри, тим більша ймовірність того, що ви не зможете налаштувати модель належним чином, і вона буде менш ефективною, ніж могла б бути.

Гаразд, тепер, коли ми встановили наші параметри, давайте підготуємо наші дані.

Хоча це гарний, чистий набір даних, нам потрібно виконати деяку попередню обробку, щоб підготувати їх до використання в нашій моделі. По-перше, виділимо стовпчик з послідовністю, яку ми хочемо передбачити, і підсумуємо його.

```
rain <- weather_data$RAIN
table(rain)
```

Рисунок 5 – Інформація про цільову змінну

Я люблю швидко поглянути на короткий опис мого набору даних перед початком роботи, щоб уникнути неприємних сюрпризів у майбутньому. Цей набір даних досить збалансований (жоден з класів не є надрідкісним), а це означає, що нам не потрібно надто турбуватися про особливі міркування щодо незбалансованих класів. У нас також немає пропущених значень, що також приємно.

Наше наступне завдання - взяти цей вектор, а потім розбити його на відрізки довжиною max_length + 1 (оскільки ми хочемо знати, який наступний елемент у послідовності, щоб побачити, наскільки добре ми його передбачили: ми будемо відрізати його перед тим, як передавати навчальні дані нашій моделі). Ми могли б просто почати з початку нашого вектора і розбити його на

шматки довжиною max_length + 1, що не перетинаються, але це дасть нам лише близько 3600 прикладів, що насправді недостатньо для навчання моделі глибокого навчання. (Чесно кажучи, ця проблема, ймовірно, занадто мала/проста, щоб взагалі використовувати глибоке навчання. Але в цьому уроці ми навчимося будувати RNN, а не обирати найкращу модель для цієї задачі, і зробимо вигляд, що це не так).

Щоб розтягнути наші дані, ми можемо використати так звану субдискретизацію з рухомим блоком, де ми розрізаємо наш вектор на частини, що перекриваються. Приблизно так:

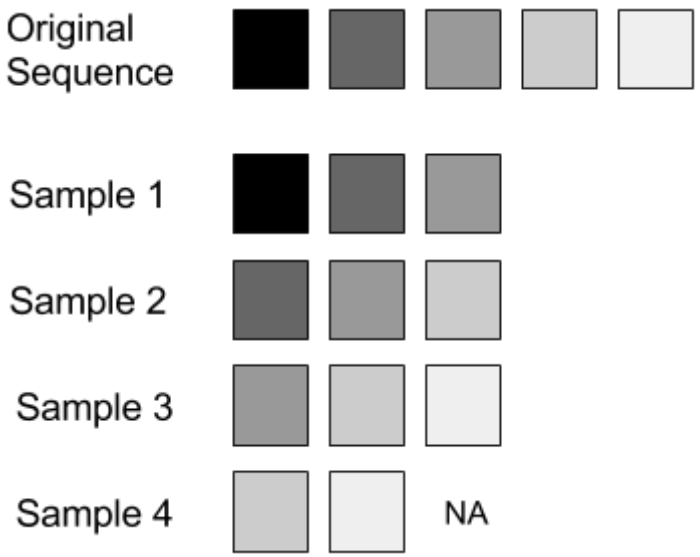


Рисунок 6 – Розділення вектора

На рисунку вище блоки мають довжину три одиниці і перекриваються на дві одиниці. У нас будуть блоки довжиною max_length + 1 і перекриваються на 3.

Важливо: Хоча це схоже на інші методи підвибірки, дуже важливо, щоб ми не змінювали порядок елементів у наших блоках! Оскільки ми будемо навчати рекурентну нейронну мережу, яка моделює взаємозв'язок елементів у послідовності, нам важливий порядок наших спостережень, і ми хочемо переконатися, що він буде збережений.

```
start_indexes <- seq(1, length(rain) - (max_len + 1), by = 3)
weather_matrix <- matrix(nrow = length(start_indexes), ncol = max_len + 1)
for (i in 1:length(start_indexes)){
  weather_matrix[i,] <- rain[start_indexes[i]:(start_indexes[i] + max_len)]
}
```

Рисунок 7 – Заповнення матриці

Тепер, коли у нас є дані вибірки у вигляді охайної матриці, нам потрібно лише зробити кілька дрібниць, щоб переконатися, що все готово до роботи.

Переконайтеся, що ваша матриця є числовою. Оскільки Keras очікує числову матрицю, ми перетворимо наші дані з логічних у числові, помноживши все на одиницю. Це не змінить жодних значень, але змінить їх на числовий тип даних і призведе до помилки, якщо ви випадково вставили текстовий рядок

Видаліть усі нулі: якщо ви випадково потрапили до даних з нулями, ваша модель буде скомпільована і навчатиметься чудово... але всі прогнози моделі будуть NaN (а не числа). Я люблю завжди включати цей крок про всяк випадок, але в даному випадку його потрібно виконати, оскільки, як показано на діаграмі вище, наш підхід до зрізу даних призвів до додавання нулів до набору даних.

```
weather_matrix <- weather_matrix * 1
if(anyNA(weather_matrix)){
  weather_matrix <- na.omit(weather_matrix)
}
```

Рисунок 8 – Перетворення матриці на числову з прибиранням пропусків

Отже, тепер, коли наші дані чисті і готові до роботи, ми можемо приступити до підготовки їх до введення в нашу модель. По-перше, нам потрібно розділити наш набір даних на вхідні (шість попередніх днів) і вихідні (один день, який ми хочемо спрогнозувати). Я буду дотримуватися домовленості і називатиму вхідні дані X, а вихідні - Y.

```
X <- weather_matrix[,-ncol(weather_matrix)]
y <- weather_matrix[,ncol(weather_matrix)]
```

Рисунок 9 – Вхідні та вихідні дані

Тепер нам просто потрібно розділити наші дані на тестовий і навчальний набір за допомогою функції createDataPartition() з пакета caret. З цього моменту ми навіть не будемо думати про те, щоб подивитися на наш тестовий набір даних, поки не прийде час оцінювати нашу фінальну модель.

```
training_index <- createDataPartition(y, p = .9,
                                      list = FALSE,
                                      times = 1)
X_train <- array(X[training_index,], dim = c(length(training_index), max_len, 1))
y_train <- y[training_index]
X_test <- array(X[-training_index,], dim = c(length(y) - length(training_index), max_len, 1))
y_test <- y[-training_index]
```

Рисунок 10 – Розділення набору даних на навчальний та тренувальний набори

Вказуємо, як має виглядати наша модель. На цьому кроці Keras дійсно блищить. Він був розроблений для того, щоб ми могли вказати, як має виглядати наша модель на дуже високому рівні, описуючи кожен шар нашої моделі в загальних рисах.

По-перше, нам потрібно вказати Keras, який тип моделі ми хочемо побудувати та ініціалізувати нашу модель. Я почну з послідовної моделі. "Послідовність", про яку йдеться, - це послідовність шарів у вашій моделі. (Зверніть увагу, що "послідовність" тут відноситься до архітектури моделі, а не до даних! Якщо ви хочете змоделювати дані, які знаходяться у послідовності, як ми збираємося зробити, вам потрібно буде вказати це в архітектурі моделі).

Інший варіант побудови моделі - це використання функціонального API. Це дозволить вам будувати більш складні типи моделей, і буде корисно, коли ви станете більш впевненими в глибокому навчанні і захочете досліджувати більш спеціалізовані моделі. Але поки що давайте зупинимося на послідовній моделі.

```
model <- keras_model_sequential()
```

Рисунок 11 – Створення моделі

Чудово, у нас є модель! Але ми ще не готові починати навчання: спочатку нам потрібно повідомити Керасу, скільки шарів повинна мати наша модель і яким має бути кожен шар. У глибокій нейронній мережі є три типи шарів:

- Вхідний шар
- Прихований шар (шари)
- Вихідний шар

У Keras ви створюєте архітектуру моделі, додаючи до неї додаткові шари. Спочатку вхідний шар, потім прихований шар (шари) і, нарешті, вихідний шар.

Скільки шарів повинна мати наша модель?

Вибір архітектури моделі є складною проблемою, особливо тому, що підхід "вгадай і протестуй" займає набагато більше часу, коли ви працюєте з моделями, на навчання яких можуть знадобитися години або дні. Існують деякі загальні рекомендації, які можна використати для вибору розумної початкової архітектури, але якщо вам потрібна модель з дуже високою продуктивністю, ви маєте бути готовими до того, що доведеться витратити чимало часу на тестування різних архітектур.

Однак, як мінімум, вам знадобиться два шари: вхідний і вихідний. Ви також можете створити один або декілька прихованих шарів. (Модель є "глибокою", якщо вона має принаймні один прихований шар між вхідним і вихідним шарами). Для цієї моделі я збираюся мати один вхідний шар, один прихований шар і один вихідний шар, ось так:

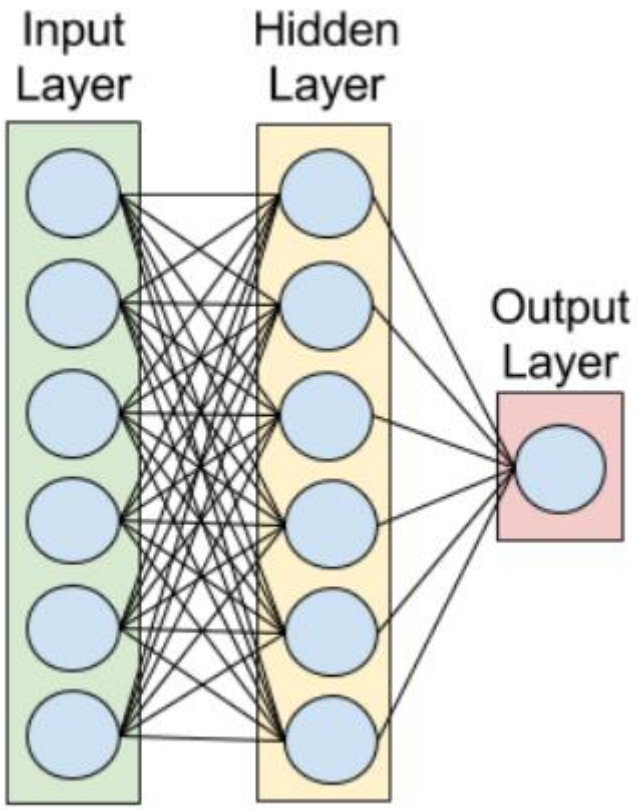


Рисунок 12 – Архітектура моделі

На цій діаграмі кола представляють нейрони в нашій моделі, а лінії - ваги. Якщо ви раніше не працювали з нейронними мережами, ідея полягає в тому, що кожен нейрон приймає всі ваги, які подаються на нього, а потім видає одне значення, яке потім передається далі і модифікується вагами, пов'язаними з цим нейроном. Коли ми навчаємо нашу модель, насправді змінюються ваги між окремими нейронами.

Вхідний шар - це перший шар, через який проходять ваші дані.

Оскільки він приймає ваші сирі дані як вхідні, вам потрібно вказати, якими будуть розміри ваших даних. Давайте подивимося на розміри матриці, яку ми будемо передавати в нашу модель, щоб зрозуміти, якою має бути наша input_shape.

```
dim(X_train)
```

Рисунок 13 – Розмірність матриці

У цьому випадку перший вимір - це кількість наших навчальних прикладів, другий - довжина вхідної послідовності (яку ми задаємо за допомогою max_len), а третій - кількість ознак, які ми маємо (у цьому випадку лише одна: був дощ чи ні). У цьому прикладі ми можемо проігнорувати перший вимір: ми будемо контролювати кількість відліків, що передаються до цього шару за один раз, за допомогою параметра batch_size, який ми задали в розділі параметрів налаштування. Щоб полегшити собі життя, якщо пізніше я вирішу змінити форму вхідних даних, я отримаю розміри вхідного масиву за допомогою функції dim() і передам їх як input_shape.

Аргумент "units" вказує нам, скільки нейронів (представлених колами на схемі вище) ми хочемо мати у цьому шарі. Для моделювання послідовностей має сенс мати стільки ж одиниць, скільки елементів у вхідній послідовності, що ми визначили за допомогою параметра max_len, тому я передам його до нашої моделі безпосередньо тут.

```
model %>%
  layer_dense(input_shape = dim(X_train)[2:3], units = max_len)
```

Рисунок 14 – Вхідний шар

Тепер, коли ми визначили, як виглядатиме наш вхідний шар, давайте розповімо Керасу про наш прихований шар. У цій моделі ми будемо використовувати лише один прихований шар, але для більших і складніших наборів даних вам може знадобитися більше шарів. (Зауважте, що додавання додаткових шарів означає, що навчання вашої моделі займе більше часу).

Як дізнатися, скільки потрібно мати прихованих шарів? Це залежить від складності об'єкта, який ви моделюєте. Для суто лінійних залежностей вам не потрібні жодні приховані шари (чесно кажучи, якщо ваші вхідні та вихідні дані мають чітку лінійну залежність один від одного, вам, ймовірно, не потрібна нейронна мережа з самого початку). Однак з одним прихованим шаром можна апроксимувати майже будь-яку залежність між входом і виходом. Якщо у вас не дуже великий або складний набір даних, я рекомендую почати з одного

прихованого шару. (Пізніше ви завжди можете спробувати додати більше і подивитися, чи це допоможе).

Для мого прихованого шару я використаю простий шар RNN. Це схоже на щільні шари, але всередині кожного нейрона знаходиться модель нашої вхідної послідовності, де другий елемент залежить від першого, третій - від першого і другого, четвертий - від першого, другого і третього, і так далі.

Наскільки великими мають бути мої приховані шари? Чим більше нейронів у ваших прихованих шарах, тим більша ймовірність того, що ви надмірно пристосуєтесь до ваших даних, і тим більше часу знадобиться на навчання вашої моделі. У той же час, якщо у вас занадто мало нейронів, ваші приховані шари не будуть дуже добре вловлювати закономірності у ваших даних, і ваша модель буде недостатньо пристосована. Почнемо з того, що ви навряд чи матимете надмірну кількість нейронів, якщо виберете розмір десь між розміром вхідного шару та вихідного шару.

Я зупинюся на шести нейронах. Оскільки це рекурентний шар мережі, це означає, що буде шість різних маленьких моделей наших вхідних послідовностей з шести елементів. Ми могли б мати лише одну, але якщо їх буде більше, то вони можуть розпізнавати різні шаблони в нашому наборі даних і допомагати нам робити більш точні передбачення. (Це буде більш імовірно, якщо ви працюєте з моделями, які мають багато функцій. Наприклад, якщо ми намагаємося передбачити, чи буде дощ на основі температури, атмосферного тиску, вологості, швидкості і напрямку вітру, в одному з наших нейронів швидкість і напрямок вітру можуть бути дуже важливими, в той час як в іншому нейроні більш важливими можуть бути вологість і атмосферний тиск).

```
model %>%
  layer_simple_rnn(units = 6)
```

Рисунок 15 – Прихований шар

На даний момент наша модель має 6 різних виходів, по одному для кожного нейрона в нашому прихованому шарі. Але ми хочемо передбачити лише один вихід: чи буде дощ наступного дня. Отже, нам потрібно додати вихідний шар, який візьме всю інформацію, яку ми маємо на даний момент, і зведе її до єдиного значення.

Якого розміру має бути мій вихідний шар? Ваш вихідний шар повинен бути розміром з кількість елементів, які ви прогнозуєте. Ви вказуєте це за допомогою аргументу "units". Оскільки в цьому випадку ми прогнозуємо, чи буде дощ протягом одного дня, одиницями будуть 1.

Ви можете помітити, що тут ми вказуємо, що хочемо використовувати сигмоїдну функцію активації. Функція активації бере всі вхідні дані в комірку і певним чином перетворює їх. У цьому випадку вона перетворює їх на число між нулем та одиницею, яке є ймовірністю того, що наступного дня піде дощ. (Ймовірність - це краще, ніж просто здогадка "так-ні", тому що якщо наша модель впевнено помиляється, нам потрібно знати, що ми повинні зробити більше коригувань, ніж якщо вона помиляється, але не дуже впевнена у своїй відповіді).

```
model %>%
  layer_dense(units = 1, activation = 'sigmoid')
```

Рисунок 16 – Вихідний шар

Тепер, коли ми створили нашу модель, давайте подивимось на нашу архітектуру. Ми можемо зробити це за допомогою функції summary(), щоб переглянути нашу модель після того, як ми закінчили додавання шарів.

```
summary(model)
```

Рисунок 17 – Інформація про модель

Це показує нам ту саму інформацію, яку ми бачили на нашій діаграмі вгорі цього розділу. Читаючи цей підсумок, ми бачимо, що наша модель має:

Вхідний шар з шістьма одиницями

Прихований (RNN) шар з шістьма одиницями

вихідний шар з однією одиницею

Це чудово, оскільки саме таку архітектуру ми вирішили використати для цієї задачі, і це говорить нам про те, що ми успішно налаштували нашу модель так, як ми хотіли.

Залишився останній крок у налаштуванні нашої моделі, перш ніж ми зможемо її навчати. Нам потрібно скомпілювати її та вказати, якими мають бути втрати та оптимізатор, а також яку метрику відстежувати (якщо вона є). Ваш вибір для кожного з цих параметрів буде залежати від вашого конкретного набору даних і проблеми.

Втрати

Під час навчання ваша модель намагатиметься вгадати, яким буде наступний елемент у послідовності. Оскільки в цьому випадку ми знаємо, що сталося далі, ми можемо порівняти наші передбачення з реальною подією. "Втрата" підказує вашій моделі, як зробити це порівняння, і яку міру використовувати, щоб зрозуміти, наскільки вона помилилася. Тут ми будемо використовувати "бінарну кросентропію", що означає, що ми хочемо, щоб наша модель точно відображала ймовірність того, що спостереження потрапить в одну з двох груп (в даному випадку, дощову або сонячну).

Оптимізатор

Оптимізатор підказує вашій моделі, як з'ясувати, які зміни потрібно зробити, щоб зменшити втрати. Тут ми використовуємо RMSprop, який оглядається на загальний напрямок змін, зроблених раніше, щоб вирішити, якою має бути наступна зміна. Він особливо добре підходить для RNN.

Метрики

Цей аргумент є необов'язковим. Він вказує нашій моделі, що, окрім втрат, вона повинна відстежувати під час навчання. Я буду запитувати точність, щоб ми могли подивитися, як змінюється точність під час навчання нашої моделі.

Як і при створенні нашої моделі, тут нам потрібно прийняти деякі рішення. (Ви можете помітити, що при створенні нейронної мережі вам

доводиться робити багато виборів. Це одна з причин, чому з ними складно працювати: у вас є багато речей, які ви можете змінити, і деякі з них можуть зіпсувати вашу модель... але ви не дізнаєтесь про це, поки вона не закінчить навчання!). Зокрема, нам потрібно вибрати розмір партії, кількість епох для навчання та часовий проміжок для валідації.

Епохи

Кількість епох - це просто кількість разів, коли наша модель побачить весь навчальний набір даних під час навчання. Чим більше епох тренується наша модель, тим більше шансів вона матиме вивчити закономірності в наших даних. Недоліком навчання на більшій кількості епох є те, що 1) навчання моделі займає більше часу і 2) ви з більшою ймовірністю можете перенавчити модель.

Крім того, в якийсь момент ваша модель навчиться всьому, що вона збирається робити з вашого набору даних, і довше навчання не допоможе вашій продуктивності. (Це станеться швидше, чим меншим/менш складним є ваш набір даних). Однак може бути важко визначити, коли це станеться, тому що іноді ваша модель може "застрягти". Тобто, вона не знатиме, які зміни потрібно внести, щоб підвищити точність.

Розподіл валідації

Це кількість даних, яку ми будемо зберігати для перевірки нашої моделі в процесі роботи. Ми встановили значення 0.1, що означає, що ми відкладаємо 10% наших навчальних даних і використовуємо їх для перевірки нашої моделі в процесі роботи. Зверніть увагу, що це відрізняється від тестових даних, про які ми говорили на початку: цей набір даних буде використано для оцінки фінальної моделі після того, як ми закінчимо її навчання. (Це допоможе нам переконатися, що наша навчена модель буде такою ж точною на нових даних, яких вона ще не бачила).

Тепер, коли ми все налаштували, настав час навчити нашу модель! Ми можемо зробити це за допомогою функції fiwith largert(). Зверніть увагу, що оскільки наша модель буде оновлюватися на місці, нам не потрібно зберігати її у змінну. Причина, по якій я зберігаю результат підбору нашої моделі у змінну,

полягає у тому, що ми можемо подивитися, як змінилася наша модель під час навчання.

Давайте потренуємо нашу модель! Це займе деякий час, і навіть більше, якщо ваша модель складніша або ви тренуєте більше епох. Оскільки це дуже маленький, іграшковий приклад, це не повинно зайняти більше двадцяти секунд або близько того.

```
trained_model <- model %>% fit(
  x = X_train, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = batch_size, # how many samples to pass to our model at a time
  epochs = total_epochs, # how many times we'll look @ the whole dataset
  validation_split = 0.1) # how much data to hold out for testing as we go along
trained_model
```

Рисунок 18 – Тренування моделі

Критерії, за якими ви оцінюватимете свою модель, залежатимуть від того, з якою метою ви вирішили її побудувати. У цьому прикладі ми хотіли передбачити, буде завтра дощ чи ні, тому нам, ймовірно, буде цікаво подивитися, наскільки точними є наші прогнози.

По-перше, ми можемо подивитися, наскільки добре наша модель впоралася з перевірочними даними під час тренування після тренування.

Це показує нам точність і втрати як для наших навчальних даних (перші два рядки), так і для валідаційних даних (другі два рядки) після останньої епохи навчання. Ми бачимо, що для наших валідаційних даних наша модель з точністю 74% передбачила, чи буде дощ наступного дня чи ні, на основі даних за попередні шість днів.

Ми також можемо бачити, як ці значення змінювалися під час навчання нашої моделі, будуючи графік на основі історії, яку ми зберегли раніше. Якщо все пройшло добре, ми побачимо, що точність зросла, а втрати зменшилися як для навчальних, так і для валідаційних даних з плином часу.

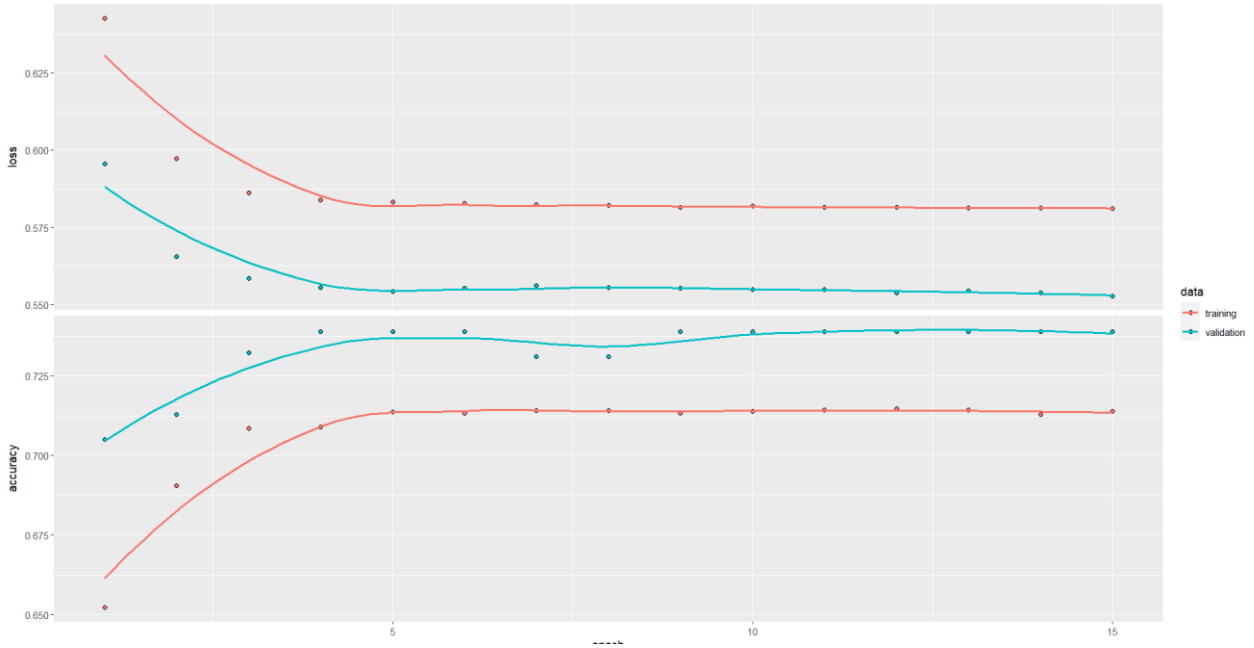


Рисунок 19 – Зміна продуктивності під час навчання

Виглядає досить непогано! Ми, мабуть, могли б припинити тренування на пару епох раніше, оскільки і точність, і втрати вирівнялися приблизно на 5 епосі. Важливо, що ми не бачимо, щоб втрати/точність на тренуванні та валідації віддалялися одне від одного. Якщо ви бачите, що втрати зменшуються на навчальних даних, але збільшуються на даних валідації, це великий червоний прапор, який вказує на те, що ваша модель надмірно пристосована!

Чому ми маємо вищу точність/нижчі втрати на валідаційних даних, ніж на навчальних? Частково це пов'язано з тим, коли проводяться ці вимірювання. Вимірювання для навчальних даних усереднюються по всій епосі, в той час як вимірювання для тестових даних проводяться в кінці епохи. Як наслідок, вимірювання для валідаційних даних потребують більше часу для покращення. Ці відмінності також можуть частково пояснюватися випадковими відмінностями між навчальними та тестовими даними: якщо валідаційні дані легше передбачити, ніж навчальні, це може призвести до вищої точності валідаційних даних.

ЛАБОРАТОРНА РОБОТА 27

Тема: Закони розподілу. Оцінка параметрів розподілу.

Приклад виконання роботи:

Для виконання роботи потрібно додати дві надбудови. Переходимо Файл -> Параметри -> Надбудови -> Вмикаємо пакет аналізу та пошук рішення:

Введіть на аркуші дані відповідно до таблиці 1. У таблиці 1 представлені дані щодо кількості справ, які на кінець звітного періоду в певному регіоні перебували у провадженні суду.

Таблиця 1 – Кількість відкритих справ

Область країни	Кількість позовів з питань охорони праці
Київська	28
Львівська	22
Миколаївська	25
Одеська	19
Хмельницька	18
Закарпатська	20
Харківська	21
Донецька	18
Тернопільська	16
Чернігівська	24

Для здійснення статистичного аналізу методом описової статистики відкрийте вкладку Дані/Data, у групі Аналіз/Analysis оберіть команду Аналіз даних/Data Analysis.

У вікні Аналіз даних/Data Analysis, що відкриється, оберіть Описова статистика/Descriptive Statistics та натисніть кнопку ОК.

У діалоговому вікні, що з’явилося, виберіть налаштування як на рисунку 2.

Інтерпретація результатів. Вихідні дані містять три узагальнюючі показники, які називаються типовими значеннями:

Середнє – це середня кількість відкритих справ, рівна 21,1. Виходить як результат розподілу суми всіх позовів на кількість об’єктів.

Медіана – значення, розташоване посередині впорядкованого набору даних. У нашому прикладі значення медіани дорівнює 20,5.

Мода – значення, яке найбільш часто зустрічається. Якщо зустрічається декілька значень, що часто зустрічаються, то Excel виводить перше з них. Якщо кожне значення зустрічається один раз, то Excel виводить запис #Н/Д. У такому випадку треба одержати таблицю розподілу частот, у якій інтервал з найбільшою частотою називається модальним інтервалом. Для визначення модального інтервалу рекомендується використати гістограми.

Вихідні дані містять довірчий інтервал для середнього = 2,64 з рівнем надійності 95%. Виходячи з цього знайдемо межі довірчого інтервалу: 21,1 – 2,64 = 18,46 21,1+2,64 = 23,74 В результаті можемо стверджувати, що розраховане середнє значення з ймовірністю 0,95 буде знаходитися у інтервалі (18,46; 23,74).

Гістограма або діаграма Парето (відсортована гістограма) – це стовпчикова діаграма, що показує частоту повторюваності значень.

Для створення гістограми в Excel скористаємося майстром діаграм із Пакету аналізу. Також додамо нову колонку:

F
Карман
16
18
20
22
24
26
28

Рисунок 3 – Нова колонка

Будуємо гістограму за допомогою аналізу даних. В результаті отримуємо таблицю та графік:

16	1	10,00%	18	2	20,00%
18	2	30,00%	20	2	40,00%
20	2	50,00%	22	2	60,00%
22	2	70,00%	16	1	70,00%
24	1	80,00%	24	1	80,00%
26	1	90,00%	26	1	90,00%
28	1	100,00%	28	1	100,00%
Еще	0	100,00%	Еще	0	100,00%

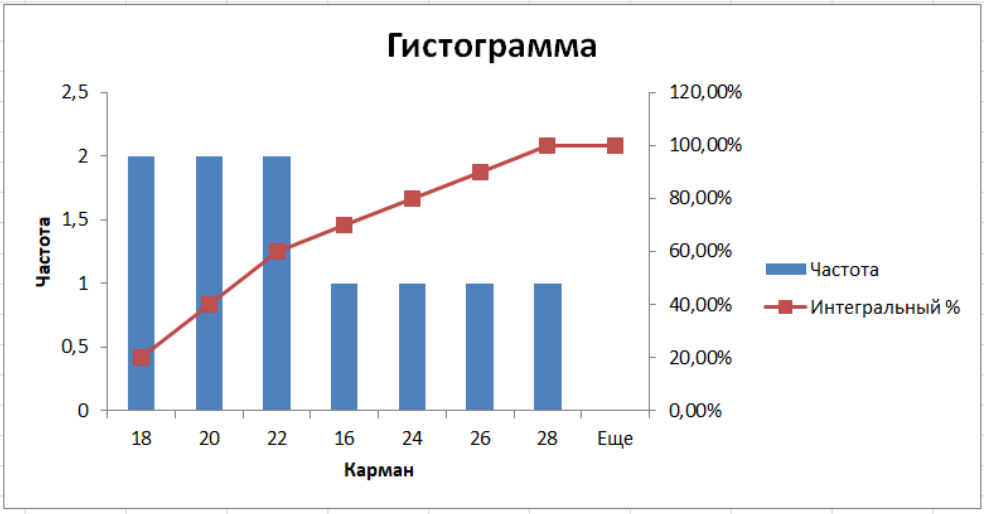


Рисунок 5 – Результат побудови гістограми

Щоб здійснити генерацію випадкових значень потрібно в надбудові «Аналіз даних» використати функцію генерація випадкових значень. Оберіть один із законів розподілу, налаштуйте його параметри та згенеруйте 20 випадкових чисел.

ЛАБОРАТОРНА РОБОТА 28

Тема: Закони розподілу. Оцінка параметрів розподілу.

Теоретичні відомості:

Асоціативним правилом (association rules) називається імплікація $X \Rightarrow Y$, де ***X*** та ***Y*** є наборами елементів множини ***I***: $X \subset I, Y \subset I$, серед яких немає однакових елементів $X \cap Y = \varnothing$.

Таким чином, **асоціативне правило** складається із двох наборів елементів (предметів) ***X*** та ***Y*** з назвами ***умова*** (antecedent)) та ***наслідок*** (consequent), які записують у вигляді $X \Rightarrow Y$. Формулюють асоціативне правило у так:

«***якщо*** умова, ***то*** наслідок» або «***якщо X, то Y***»

Задача пошуку асоціативних правил полягає у здійсненні аналізу множини транзакцій T, які мітяться у базі даних D, з метою знаходження таких правил, які мають високу ймовірність.

Асоціативні правила дозволяють знаходити закономірності між пов’язаними подіями.

Алгоритми пошуку асоціативних правил стали одним з популярних методів виявлення прихованих закономірностей та побудови знань. Вони дозволяють досліджувати взаємозв’язок між подіями, що відбуваються спільно.

Транзакція – деяка множина подій, що відбуваються спільно.

Предметний набір – не порожня множина об’єктів, що з’явилися в одній транзакції.

Аналіз ринкового кошика – це аналіз наборів даних для певної комбінації товарів, пов’язаних між собою.

Оцінки асоціативних правил

1. Підтримка S асоціативного правила – це відношення кількості транзакцій k, що містять умову і наслідок, до загальної кількості транзакцій m.

2. **Достовірність C асоціативного правила** $X \Rightarrow Y$ є мірою точності правила й визначається як відношення кількості транзакцій ***k***, що містять умову і наслідок, до кількості транзакцій ***l***, що містять тільки умову.

3. **Ліфт L** (lift – підвищення інтересу) **асоціативного правила** $X \Rightarrow Y$ є відношенням частоти появи умови в транзакціях, які містять і умову і наслідок, до частоти появи наслідку в цілому.

4. **Левередж Lv** (leverage – важіль, плече) **асоціативного правила** $X \Rightarrow Y$ – це різниця між спостережуваною частотою, з якою умова й наслідок з'являються спільно та добутком частоти появи умови й наслідку окремо.

5. **Поліпшення Im** (improvement) **асоціативного правила** $X \Rightarrow Y$ – це відношення частоти спостережуваних виконань правила до добутку частот появи умови й наслідку окремо.

Етапи алгоритму Apriori

Алгоритм Apriori визначає набори, які часто зустрічаються, за декілька етапів.

Кожен етап передбачає визначення усіх наборів елементів, які часто зустрічаються й складається із двох кроків:

1) формування кандидатів (candidate generation) ;

2) підрахунок підтримки кандидатів (candidate counting).

На і-му етапі:

1) на кроці формування кандидатів алгоритм створює множину кандидатів з і-елементних наборів, підтримка яких поки не обчислюється;

2) на кроці підрахунку кандидатів алгоритм сканує множину транзакцій:

— обчислюючи підтримку наборів-кандидатів;

— відкидаючи після сканування тих, підтримка яких менша за визначений користувачем мінімум;

— зберігаючи ті і-елементні набори, які часто зустрічаються. Під час 1-го етапу обрана множина наборів-кандидатів містить всі 1- елементні часті набори. Алгоритм обчислює їх підтримку під час кроку підрахунку кандидатів.

Опишемо алгоритм Apriori по крокам:

Крок 1. Привласнюється k = 1 і виконується відбір всіх 1- елементних

наборів, у яких підтримка більша мінімально заданої користувачем S_{min}.

Крок 2. Привласнюється k = k + 1.

Крок 3. Якщо не вдається створити k-елементні набори, то алгоритм завершується, інакше виконується наступний крок.

Крок 4. Створюється множина k-елементних наборів кандидатів у часті набори:

— для цього необхідно об'єднати в k-елементні кандидати (k – 1) - елементні часті набори;

— кожен кандидат m Mk формується шляхом додавання до (k – 1)-елементного частого набору p останнього елемента з іншого (k – 1)-елементного частого набору q;

— елемент, який додається з набору q, по порядку є вищим, ніж останній елемент набору p;

— усі k – 2 елементи обох наборів p і q однакові.

Крок 5. Для кожної транзакції T з множини D обираються кандидати з множини M_k , присутні у транзакції T – формується множина M_t.

Крок 6. У побудованій множині M_t видаляється кожен набір, підтримка якого менша за задану користувачем S_{min} – формується множина L_k . Повернення до кроку 2.

Результатом роботи алгоритму є об'єднання усіх множин L_k при усіх k: { L_1, L_2,..., L_k }.

Приклад виконання роботи:

Визначення оцінок асоціативних правил

Для набору транзакцій, заданих у таблиці 1, здійснити оцінку правил, визначивши підтримку, достовірність, ліфт, левередж та поліпшення.

Таблиця 1

№	Транзакція
1	Сливи, салат, помідори
2	Селера, цукерки
3	Цукерки
4	Яблука, морква, помідори, картопля, цукерки
5	Яблука, апельсини, салат, цукерки, помідори
6	Персики, апельсини, селера, помідори
7	Квасоля, салат, помідори
8	Апельсини, салат, морква, помідори, цукерки
9	Яблука, банани, сливи, морква, помідори, цибуля, цукерки
10	Яблука, картопля

Рисунок 1 – Таблиця для обробки

1. Визначити підтримку та достовірність правил цукерки -> помідори та салат -> помідори.

1.1. Розглянувши правило салат -> помідори для даних, представлених у таблиці 1, маємо:

$$S(салат \rightarrow помідори) = 4/10 = 0{,}4;$$
$$C(салат \rightarrow помідори) = 4/4 = 1.$$

Дане правило зустрічається у 40% транзакцій, тому його підтримка S = 0,4.

У всіх випадках, коли покупець купує салат, він купує й помідори, тому достовірність правила C = 1.

1.2. Розглянувши асоціацію цукерки -> помідори для даних, представлених у таблиці 1, отримуємо, що S = 4/10 = 0,4 також, однак C = 4/6 = 0,67.

2. Визначити ліфт правил помідори -> салат та помідори -> цукерки

З таблиці 1 для цих правил маємо:

P(салат) = 4/10 = 0,4 C(помідори -> салат) = 4/7=0,57

P(цукерки) = 6/10 = 0,6 C(помідори -> цукерки) = 4/7=0,57

Правила здаються однаково достовірними, однак після обчислення ліфта для кожного з правил виявляємо значно сильніший зв'язок для умови та наслідку у правила помідори салат:

L(помідори -> салат) = 0,57/0,4= 1,425

L(помідори -> цукерки) = 0,57/0,6= 0,95

3. Визначити левередж правил салат -> помідори та морква -> помідори.

З таблиці 1 маємо:

Для правила салат -> помідори: C= 1 L = 1/0,7 = 1,43

Для правила морква -> помідори: C= 1 L = 1/0,7 = 1,43

Як бачимо, ці правила мають однакові достовірність та ліфт. Проте перше правило становить більший інтерес, тому що воно зустрічається частіше, тобто, застосовується для більшого числа покупців. Левередж правил буде різним:

Lv(морква -> помідори) = 0,3 – 0,3*0,7 = 0,09

Lv(салат -> помідори) = 0,4 – 0,4*0,7 = 0,12

Тому правило салат помідори має більшу значимість і є більш цінним.

4. Визначити поліпшення правил салат помідори та морква помідори.

Im(морква -> помідори) = 0,3 /(0,3*0,7) = 1,43

Im(салат -> помідори) = 0,4 /(0,4*0,7) = 1,43

Поліпшення показує, у скільки разів правило забезпечує правильний прогноз краще, ніж випадкове вгадування.

Пошук асоціативних правил з допомогою алгоритму Apriori

Для множини об’єктів I = {шоколад, чіпси, кокоси, вода, пиво, горіхи}, які є товарами (табл. 2) та множини D транзакцій (табл. 3) з допомогою алгоритму Apriori здійснити пошук асоціативних правил при порогових мінімальній підтримці S_{min} = 0,5 та мінімальній достовірності C_{min} = 0,75

Таблиця 2. Прайс лист товарів

Ідентифікатор	Назва товару	Ціна
0	Шоколад	30.00
1	Чіпси	12.00
2	Кокоси	10.00
3	Вода	4.00
4	Пиво	14.00
5	Горіхи	15.00

Таблиця 3. Набори товарів у транзакціях множини ***D***

Номер транзакції	Номер товару	Найменування товару	Ціна
0	1	Чіпси	12.00
0	3	Вода	4.00
0	4	Пиво	14.00
1	2	Кокоси	10.00
1	3	Вода	4.00
1	5	Горіхи	15.00
2	5	Горіхи	15.00
2	2	Кокоси	10.00
2	1	Чіпси	12.00
2	2	Кокоси	10.00
2	3	Вода	4.00
3	2	Кокоси	10.00
3	5	Горіхи	15.00
3	2	Кокоси	10.00

Задача знаходження асоціативних правил розбивається на дві підзадачі:

1. Знаходження усіх наборів елементів, які задовольняють мінімальному порогу підтримки $S_{min} = 0,5$.

2. Генерація правил із знайдених наборів елементів з достовірністю, яка задовольняє заданому порогу $C_{min} = 0,75$.

Число транзакцій – 4.

Крок 1. Привласнюємо k = 1 та формуємо множину M_1 кандидатів (одноелементних наборів товарів), у яких підтримка більша за $S_{min} = 0,5$ (табл. 4).

Таблиця 4. Множина M_1 одноелементних наборів товарів

№	Набір	Підтримка S
1	{0}	0/4 = 0
2	{1}	2/4 = 0,5
3	{2}	3/4 = 0,75
4	{3}	3/4 = 0,75
5	{4}	1/4 = 0,25
6	{5}	3/4 = 0,75

Заданій мінімальній підтримці відповідають кандидати 2, 3, 4 та 6 з товарами 1, 2, 3 та 5 відповідно: L_1 = {{1}, {2}, {3}, {5}}.

Крок 2. Привласнюємо k = 2 та формуємо множину M_2 кандидатів (двоелементних наборів товарів), у яких підтримка більша за S_{min} = 0,5 (табл. 5).

Таблиця 5. Множина M_2 двоелементних наборів товарів

№	Набір	Підтримка S
1	{1,2}	0,25
2	{1,3}	0,5
3	{1,5}	0,25
4	{2,3}	0,5
5	{2,5}	0,75
6	{3,5}	0,5

Заданій мінімальній підтримці відповідають кандидати 2, 4, 5 та 6 з товарами відповідно:

L_2 = {{1,3}, {2,3}, {2,5}, {1,5}}.

Відсікаються кандидати 1 та 3: {{1,2}, {4}}.

Крок 3. Привласнюємо k = 3 та формуємо множину M_3 кандидатів (трьохелементних наборів товарів), у яких підтримка більша за Smin = 0,5 (табл. 6).

Таблиця 6. Множина M_3 трьохелементних наборів товарів

№	Набір	Підтримка S
1	{2,3,5}	0,5

На даному кроці отримуємо: L_3 = {{2,3,5}}.

Так як 4-х елементні набори створити не вдасться, то алгоритм завершується.

Результатом роботи алгоритму Apriori є множина наборів, серед яких буде здійснюватися пошук асоціативних правил:

$$L = L_1 \cup L_2 \cup L_3 = \{\{1\}, \{2\}, \{3\}, \{5\}, \{\{1,3\}, \{2,3\}, \{2,5\}, \{3,5\}, \{2,3,5\}\}.$$

На отриманій множині наборів можуть бути згенеровані правила, представлені у таблиці 6. Для кожного з цих правил знаходять достовірність.

Таблиця 6. Згенеровані асоціативні правила

№	Правило	Достовірність C
1	«*якщо* 1 *то* 3»	1
2	«*якщо* 2 *то* 3»	2/3=0,67
3	«*якщо* 2 *то* 5»	1
4	«*якщо* 2 *то* 3 і 5»	2/3=0,67
5	«*якщо* 3 *то* 1»	2/3=0,67
6	«*якщо* 3 *то* 2»	2/3=0,67
7	«*якщо* 3 *то* 5»	2/3=0,67
8	«*якщо* 3 *то* 2 і 5»	2/3=0,67
9	«*якщо* 5 *то* 2»	1
10	«*якщо* 5 *то* 3»	2/3=0,67
11	«*якщо* 5 *то* 2 і 3»	2/3=0,67

Із згенерованих асоціативних правил залишаємо ті, достовірність яких не менша за $C_{min} = 0,75$. Цій умові задовольняють три правила:

«якщо 1 то 3», «якщо 2 то 5», «якщо 5 то 2».

Або:

Правило 1: «якщо чіпси то вода»

Правило 2: «якщо кокоси то горіхи»

Правило 3: «якщо горіхи то кокоси»

ЛАБОРАТОРНА РОБОТА 29

Тема: Однофакторний дисперсійний аналіз.

Приклад виконання роботи:

Трьом групам студентів промовляли з різною швидкістю (низькою, середньою, високою) десять слів. Довести (або спростувати) припущення про те, що фактор швидкості пред'явлення слів впливає на показники їх відтворення.

Емпіричні дані наведено у таблиці.

№ п/пр	Швидкість пред'явлення		
	Низька	Середня	Висока
1	7	5	5
2	8	5	4
3	7	6	5
4	9	5	3
5	5	4	4
6	7	6	5
7		4	4
8			3

1.Формулюємо гіпотези:

— H0: Відмінності в обсязі відтворення слів не є більш вираженими, ніж випадкові відмінності в середині групи (фактор швидкість пред'явлення не впливає на показник відтворення).

— H1: Відмінності в обсязі відтворення слів є більш вираженими, ніж випадкові відмінності всередині групи (фактор швидкість пред'явлення впливає на показник відтворення).

2. Формуємо таблицю вхідних даних. Для цього об'єднуємо комірки A1:A2, куди вводимо назву «№» та комірки B1:D1, куди вводимо назву «Швидкість пред'явлення». В комірку B2 вводимо назву «Низька», в комірку

C2 вводимо назву «Середня», в комірку D2 вводимо назву «Висока». В сформовану таблицю вносимо дані задачі (рис. 1).

	A	B	C	D
1	№ п/пр	Швидкість пред'явлення		
2		Низька	Середня	Висока
3	1	7	5	5
4	2	8	5	4
5	3	7	6	5
6	4	9	5	3
7	5	5	4	4
8	6	7	6	5
9	7		4	4
10	8			3

Рисунок 1 – Таблиця для обробки

3. Формуємо таблиці для розрахунків.

3.1. Вводимо в комірку A11 “ni=” , а в комірки B11, C11, D11 – числа, що відповідають обсягам відповідних груп у стовпцях B, C та D, які обчислюємо за формулою =СЧЕТ(B3:B8)/COUNT(B3:B8) (аналогічно для стовпців B та D).

3.2. У наступних рядках розраховуємо величини, обернені до обсягів груп “1/ni=”, суми значень по групам, середні. У комірці C15 рахуємо загальний обсяг C15 = B11+C11+D11, у комірці C16 рахуємо загальне середнє =СРЗНАЧ(B14:D14) (рис. 2).

	A	B	C	D
1	№ п/пр	Швидкість пред'явлення		
2		Низька	Середня	Висока
3	1	7	5	5
4	2	8	5	4
5	3	7	6	5
6	4	9	5	3
7	5	5	4	4
8	6	7	6	5
9	7		4	4
10	8			3
11	ni =	6	7	8
12	1/ni =	0,166667	0,142857	0,125
13	Всього	43	35	33
14	Середні	7,166667	5	4,125
15	Загальний обсяг		21	
16	Загальне середнє		5,285714	5,430556

Рисунок 2 – Розрахунки середніх значень

3.3. Зліва від таблиці вхідних даних формуємо таблиці для обчислень.

— Об'єднуємо E1:G1, вводимо назву "Квадрати різниць по групах". У E2, F2, G2 вводимо назви «Низька», «Середня» і «Висока»

— Об'єднуємо H1:J1, вводимо назву "Квадрати різниць із заг. сер.". У H2, I2, J2 вводимо назви «Низька», «Середня» і «Висока».

3.4. У таблиці Квадрати різниць по групах розраховуємо квадрати різниць вхідних значень та середніх у стовпцях:

— У комірці E3 вводимо формулу =(B3-B$14)^2. Поширюємо її на стовпець E4:E8

— У комірці F3 вводимо формулу =(C3-C$14)^2. Поширюємо її на стовпець F4:F9

— У комірці G3 вводимо формулу =(D3-D$14)^2. Поширюємо її на стовпець G4:G10

3.5. У таблиці Квадрати різниць із заг. сер. розраховуємо квадрати різниць вхідних значень та загальних середніх:

— У H3 вводимо формулу =(B3-C16)^2. Поширюємо її на стовпець H4:H8

— У I3 вводимо формулу =(C3-C16)^2. Поширюємо її на стовпець I4:I9

— У J3 вводимо формулу =(D3-C16)^2. Поширюємо її на стовпець J4:J10

Результати обчислень будуть мати вигляд, представлений на рисунку 3.

	E	F	G	H	I	J
1	Квадрати різниць по групах			Квадрати різниць із заг. сер.		
2	низька	середня	висока	низька	середня	висока
3	0,111111	0	0,765625	3,014082	0,069637	0,069637
4	1,777778	0	0,015625	7,486304	0,069637	1,597415
5	0,111111	1	0,765625	3,014082	0,54186	0,069637
6	0,444444	0	1,265625	0,54186	0,069637	5,125193
7	0,111111	1	0,015625	3,014082	1,597415	1,597415
8	2,777778	1	0,765625	0,069637	0,54186	0,069637
9		1	0,015625		1,597415	1,597415
10			1,265625			5,125193

Рисунок 3 – Розрахунки квадратів різниць

4. Формуємо таблиці для обчислення результатів дисперсійного аналізу (рис. 4).

18	Вид дисперії	Сума квадратів відхилень		Ступені свободи	Статистичні оцінки дисперсій	
19	Внутрішньогрупова	Dвнут. =	14,20833	18	SSвнут. =	0,789352
20	Міжгрупова	Dміжгр. =	22,3631	2	SSміжгр. =	11,18155
21	Загальна	Dзаг. =	36,87905	20	SSзаг. =	1,843953

Рисунок 4 – Розрахунки дисперсій

Для розрахунку дисперсій:

— внутрішньогрупової Dвнутр: у комірку D19 вводимо формулу =СУММ(E3:E8;F3:F9;G3:G10)

— міжгрупової Dміжгруп: у комірку D20 вводимо формулу =B11*(B14-C16)^2+C11*(C14-C16)^2+D11*(D14-C16)^2

— загальної Dзаг: у комірку D21 вводимо формулу =СУММ(H3:H8;I3:I9;J3:J10)

Для розрахунку ступенів свободи:

— внутрішньогрупової: у комірку F19 вводимо формулу =C15-3 (загальний обсяг вибірок мінус 3 рівні фактору – низький, середній, високий)

— міжгрупової: у комірку F20 вводимо формулу =3-1 (кількість рівнів фактору мінус 1)

— загальної: у комірку F21 вводимо формулу =C15-1 (загальний обсяг вибірок мінус 1й)

Для розрахунку статистичних оцінок дисперсій:

— внутрішньогрупової SSвнутр: у комірку I19 вводимо формулу =D19/F19

— міжгрупової SSміжгруп: у комірку I20 вводимо формулу = D20/F20

— загальної SSзаг: у комірку I21 вводимо формулу = D21/F21 (відповідна дисперсія поділена на ступінь свободи

5. Формуємо таблицю для обчислення критерію Фішера (рис. 5).

— емпіричного Fемп: у комірку B24 вводимо формулу =I20/I19

— критичного на рівні значущості 0,01 F0.01: у комірку B25 вводимо формулу = FРАСПОБР(0,01;F20; F19)

— критичного на рівні значущості 0,05 F0.05: у комірку B26 вводимо формулу = FРАСПОБР(0,05;F20; F19)

23	Критерій Фішера	
24	Fемп =	14,16548
25	F0,01 =	6,012905
26	F0,05 =	3,554557

Рисунок 5 – Розрахунки критерію Фішера

6. Здійснюємо аналіз отриманих результатів. Якщо емпіричне значення критерію є більшим за критичне то нульову гіпотезу відхиляють (правобічна критична область).

Отримане емпіричне значення критерію Фішера більше за критичне на рівнях значущості 0,01 та 0,05. Тому нульову гіпотезу відхиляємо, приймаємо гіпотезу H_1, відповідно до якої фактор швидкість пред'явлення слів впливає на показник відтворення слів.

7. Розв'язання задачі однофакторного дисперсійного аналізу з використанням пакету «Анализ данных» MS Excel.

7.1.Задачу можна виконати значно швидше, застосувавши однофакторний дисперсійний аналіз пакету «Анализ данных». Для цього на вкладці Данные/Data обираємо групу Анализ/Analysis - Анализ данных/ Data Analysis та у вікні вибору інструменту аналізу обираємо Однофакторний дисперсійний аналіз .

7.2. У наступному вікні налаштовуємо параметри однофакторного дисперсійного аналізу як вказано у вікні нижче та натискаємо кнопку ОК.

7.3. У вказаному при налаштуванні вихідному діапазоні буде виведено результат здійсненого однофакторного дисперсійного аналізу.

Тут SS — сума квадратів різниць між групова та внутрішньогрупова;

df — число степенів свободи;

MS —дисперсія;

F — розрахункове значення критерію Фішера;

P-Значение — розрахункове значення мінімальної значущості;

F критическое — критичне значення критерію Фішера.

7.4. Аналіз результатів. Як бачимо, отримані з допомогою пакету «Анализ данных» результати співпадають з отриманими раніше. Розрахункове значення критерію Фішера 14,165 більше критичного 3,554. Тому нульова гіпотеза про відсутність впливу фактору швидкості пред'явлення слів на показник відтворення слів відкидається.

ЛАБОРАТОРНА РОБОТА 30

Тема: Регресійний аналіз. Лінійна регресія.

Приклад виконання роботи:

Побудувати регресійну модель, що описує залежність сумарних виробничих витрат Y (тис. грн.) від об'ємів виробництва X (тис. од.).

Емпіричні дані наведено у таблиці.

X	41	44	52	57	59	64	68	70	73	75
Y	670	657	713	736	778	812	833	876	911	932

1.1. Формування таблиці вхідних даних. Кількість пар емпіричних даних невелике – 10, тому для проведення регресійного аналізу їх можна не групувати. В Excel формуємо електронну таблицю:

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3	X	41	44	52	57	59	64	68	70	73	75
4	Y	670	657	713	736	778	812	833	876	911	932

Рисунок 1 – Таблиця для обробки

1.2. Побудова емпіричної лінії регресії. Для цього виділяємо вхідні дані та обраємо Вставка - Діаграми, обравши тип діаграми – Точечная.

Рисунок 2 – Емпірична лінія регресії

1.3. Вибір моделі регресії. Оскільки емпірична лінія регресії наближається до прямої лінії, висуваємо гіпотезу H_0 про лінійну залежність Y від X. Рівняння регресії будемо шукати у вигляді y=ax+b.

1.4. Знаходження параметрів a, b рівняння регресії. Для знаходження параметрів рівняння регресії y = ax + b скористаємося функцією Excel ЛИНЕЙН() / LINEST(), яка повертає параметри лінійного наближення по методу найменших квадратів:

а) підготуємо комірки для коефіцієнтів лінійної регресії, які будуть розраховуватися:

	L	M	N
1		Рівняння регресії	
2		*y = ax + b*	
3		*a*	*b*
4			

Рисунок 3 – Підготовка комірок

б) виділяємо комірки M4:N4, набираємо формулу =ЛИНЕЙН(B4:K4;B3:K3) / LINEST(B4:K4;B3:K3) та, оскільки це є функція масиву, натискаємо комбінацію клавіш Ctrl+Shift+Enter.

в) у комірках M4 та N4 з'являються розраховані коефіцієнти лінійної регресії:

	L	M	N
1		Рівняння регресії	
2		$y = ax + b$	
3		*a*	*b*
4		8,0592	305,832

Рисунок 4 – Коефіцієенти регресії

г) отже, шукане рівняння регресії має вигляд y=8,06x-305,83.

1.5. Перевірка правильності побудови моделі регресії. Для перевірки лінійної моделі на адекватність:

а) доповнимо електронну таблицю з вхідними даними рядками, які містять розраховані за знайденим рівнянням регресії значення $Y_{теор}$ та їх відхилення від емпіричних $Y - Y_{теор}$ при заданих значеннях X:

	A	B	C	D	E	F	G	H	I	J	K
2											
3	X	41	44	52	57	59	64	68	70	73	75
4	Y	670	657	713	736	778	812	833	876	911	932
5	$Y_{теор}$	636	660	725	765	781	822	854	870	894	910
6	$Y - Y_{теор}$	33,7	-3,4	-12	-29	-3,3	-9,6	-21	6,03	16,8	21,7

б) За даними утвореної електронної таблиці розрахуємо дисперсії, скориставшись функцією Excel ДИСП() / VAR().

$$\sigma_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

— для розрахунку загальної дисперсії введемо у комірку M10 формулу =ДИСП(B4:K4)

$$\sigma_p^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_{i,\text{теор}} - \bar{y})^2$$

— для розрахунку дисперсії регресії введемо у комірку M11 формулу =ДИСП(B5:K5)

— для розрахунку залишкової дисперсії

$$\sigma_e^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_{i,\text{теор}} - y_i)^2$$

введемо у комірку M12 формулу =ДИСП(B6:K6)

в) отримуємо розраховані значення дисперсій:

	L	M	N	O
9				
10		9508,84	загальна дисперсія σ_y^2	
11		9122,63	дисперсія регресії σ_p^2	
12		386,219	залишкова дисперсія σ_e^2	

Оскільки основне варіаційне рівняння для побудованої моделі має вигляд: 9508,84 = 9122,63 + 386,219 і є тотожністю, робимо висновок – рівняння регресії побудовано правильно.

1.6. Перевірка статистичної значущості рівняння регресії: а) розрахуємо критерій Фішера F, з врахуванням ступенів свободи:

$$F = \frac{\sigma_p^2 (10-2)}{\sigma_e^2 (2-1)} \approx 188,96$$

б) знайдемо $F_{кр} = FРАСПОБР(0,001; 2-1; 10-2) \approx 25,41$

в) розраховане значення $F > F_{kp}$, тому регресійна модель є статистично значущою на рівні 0,001.

1.7. Знаходимо коефіцієнт детермінації R^2 :

$$R^2 = \frac{\sigma_p^2}{\sigma_y^2} \approx 0,96$$

Значення коефіцієнта детермінації свідчить, що 96% варіації результативної ознаки Y пояснюється рівнянням регресії.

1.8. Знаходимо коефіцієнт кореляції

$$r_{xy} = \frac{\text{cov}(x,y)}{\sigma_x \sigma_y}, \quad \text{де} \quad \text{cov}(x,y) = \frac{1}{n}\sum (x_i - \bar{x})(y_i - \bar{y})$$

коваріація – сумісна варіація x та y.

Для розрахунку коефіцієнта кореляції Пірсона в Excel використовують функцію:

PEARSON / КОРРЕЛ(масив1; масив 2)

де масив1 - множина незалежних значень,

масив 2 - множина залежних значень.

Маємо $r_{xy} = 0,97948$;

Значення коефіцієнта кореляції свідчить, що між результативною ознакою Y та факторною ознакою X існує тісний лінійний зв’язок.

1.8. Здійснення регресійного аналізу за допомогою Пакету аналізу. Для вибору інструменту Регресія необхідно обрати вкладку Данные → групу Анализ → Анализ данных → Регресія. У вікні, що відкриється, обираємо наступні налаштування:

— вхідний інтервал Y: вказати

— вхідний інтервал X: вказати

— рівень надійності – 95%

— прапорець в: Мітки

— вихідний інтервал Новий рабочий лист

Важливо! Вхідні дані необхідно ввести в стопцях. Після вибору параметрів натискаємо ОК.

На новому робочому аркуші буде виведена інформація стосовно проведеного регресійного аналізу. Зробимо її аналіз.

— Коефіцієнт R = 0,979 свідчить про наявність суттєвого зв'язку між результативною та факторною ознаками

— коефіцієнт детермінації (R- квадрат = 0,959) показує, що 95,9% результативної ознаки обумовлена обраною факторною ознакою. 4,1% обумовлені факторами, які не включено до моделі

2) показники, що характеризують достовірність моделі регерсії:

Показник значущості F та показник F-статистики свідчить про достатній рівень достовірності результатів оцінювання.

3) таблиця коефіцієнтів

— для встановлення значущості коефіцієнтів регресії порівнюємо розраховані t-критерії з критичним, який знаходимо за встановленим рівнем значущості 0,05 за допомогою функції СТЬЮДРАСПОБР / TINV(рівень значущості, ступінь свободи) програмного пакету MS Excel t-критичне = 2,26. Умова tрозраховане > t-критичне виконується для факторної змінної X, яка для даної моделі буде значущою. (ступінь свободи k=n-m= 10-1 = 9)

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Звенігородський О. С., Зінченко О. В., Чичкарьов Є. А., Кисіль Т. М. Штучний інтелект. Вступний курс : навчальний посібник. Київ : ДУТ, 2022. 193 с. URL:https://duikt.edu.ua/uploads/l_492_92652604.pdf
2. Іванченко А. Види машинного навчання: посібник для початківців. Acer Corner. 2024. URL: https://blog.acer.com/ua/discussion/2054/vidi-mashinnogo-navchannya-posibnik-dlya-pochatkivciv
3. Кларк Е. Підручник зі штучного інтелекту для початківців: ознайомтеся з основами ШІ. Guru99. 2025. URL:https://www.guru99.com/uk/ai-tutorial.html
4. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. Харків: ХНУ імені В. Н. Каразіна, 2020. 301 с.
5. Лабораторний практикум з систем штучного інтелекту / уклад. О. Мосіюк. Житомир : Вид-во ЖДУ ім. Івана Франка, 2024. 104 с. URL: https://eprints.zu.edu.ua/40845/1/PraktykumAI.pdf
6. Машинне навчання : навчальний посібник / за науковою редакцією д.т.н., проф., В. В. Пасічника ; Т. М. Басюк, В. В. Литвин, Л. М. Захарія, Н. Е. Кунанець. 3-тє видання, стереотипне. Львів : «Новий Світ-2000», 2026. 330 с.
7. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних : електронний навчальний посібник. Вінниця : ВНТУ, 2024. 263 с. URL:https://pdf.lib.vntu.edu.ua/books/2024/Mokin_2024_263.pdf
8. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Системи штучного інтелекту : навчальний посібник. Львів : Магнолія 2006, 2024. 279 с.
9. Обчислювальний інтелект : навч. посібник / О. Ю. Заковоротний, Т. О. Орлова, Д. В. Гриньов ; Нац. техн. ун-т "Харків. політехн. ін-т". Харків : НТУ "ХПІ", 2024. 264 с. URI: https://repository.kpi.kharkov.ua/handle/KhPI-Press/85777.
10. Основи обчислювального інтелекту : лаб. практикум / уклад. О. Ю. Заковоротний, Т. О. Орлова, Д. В. Гриньов, В. М. Сергієнко ; Нац. техн. ун-т "Харків. політехн. ін-т". Харків : НТУ "ХПІ", 2023. 170 с. URI: https://repository.kpi.kharkov.ua/handle/KhPI-Press/74045.
11. Солодовник Г. В. Методи та системи штучного інтелекту. Харків : ТОВ «ДІСА ПЛЮС», 2021.177 с. https://surl.lu/ulafnc

12. Стратегія розвитку штучного інтелекту в Україні : монографія / за ред. А. І. Шевченка. Київ : Інститут проблем розвитку штучного інтелекту, 2023. 305 с. URI: https://jai.in.ua/archive/2023/ai_mono.pdf
13. Троцько В. В. Методи штучного інтелекту: навчально-методичний і практичний посібник. Київ : Університет економіки та права «КРОК», 2020. 86 с. URI: https://library.krok.edu.ua/media/library/category/navchalni-posibniki/trotsko_0001.pdf
14. Штучний інтелект. Нейромережева обробка інформації : архітектури, навчання, застосування : навчальний посібник у 2-х ч. : Ч. 1 / О. Г. Руденко, О. О. Безсонов, С. П. Євсеєв, О. Б. Ахієзер, Ю. І. Зайцев ; за заг. ред. С. П. Євсеєва. Харків : НТУ «ХПІ», Львів : «Новий Світ-2000», 2025. 426 с.

Додаткова

1. Bata, M., Carriveau, R., & Ting, D. S. K. (2020). Short-term water demand forecasting using hybrid supervised and unsupervised machine learning model. *Smart Water*, 5(1). https://doi.org/10.1186/s40713-020-00020-y
2. Jung, G., & Choi, S.-Y. (2021). Forecasting foreign exchange volatility using deep learning autoencoder-lstm techniques. *Complexity*, 2021, 1–16. https://doi.org/10.1155/2021/6647534
3. Liu, Q., Li, Z., Ji, Y., Martinez, L., Ul Haq, Z., Javaid, A., Lu, W., & Wang, J. (2019a). Forecasting the seasonality and trend of pulmonary tuberculosis in Jiangsu Province of China using advanced statistical time-series analyses. *Infection and Drug Resistance*, Volume 12, 2311–2322. https://doi.org/10.2147/idr.s207809
4. Petropoulos, F., Kourentzes, N., Nikolopoulos, K., & Siemsen, E. (2018). Judgmental selection of forecasting models. *Journal of Operations Management*, 60(1), 34–46. https://doi.org/10.1016/j.jom.2018.05.005
5. Rubio, L., & Alba, K. (2022). Forecasting selected colombian shares using a hybrid ARIMA-SVR model. *Mathematics*, 10(13), 2181. https://doi.org/10.3390/math10132181
6. Salah, O. M., Mahdi, G. J. M., & Al-Latif, I. A. A. (2021). A modified ARIMA model for forecasting chemical sales in the USA. *Journal of Physics: Conference Series*, 1879(3), 032008. https://doi.org/10.1088/1742-6596/1879/3/032008
7. Sun, J. (2021). Forecasting COVID-19 pandemic in Alberta, Canada using modified ARIMA models. *Computer Methods and Programs in Biomedicine Update*, 100029. https://doi.org/10.1016/j.cmpbup.2021.100029
8. Tripathi, M., Kumar, S., & Inani, S. K. (2020). Exchange rate forecasting using ensemble modeling for better policy implications. *Journal of Time Series Econometrics*. https://doi.org/10.1515/jtse-2020-0013

9. Wang, Y.-w., Shen, Z.-z., & Jiang, Y. (2018). Comparison of ARIMA and GM(1,1) models for prediction of hepatitis B in China. *Plos One*, 13(9), Стаття e0201987. https://doi.org/10.1371/journal.pone.0201987
10. Zhu, N., Zhang, D., Wang, W., Li, X., Yang, B., Song, J., Zhao, X., Huang, B., Shi, W., Lu, R., Niu, P., Zhan, F., Ma, X., Wang, D., Xu, W., Wu, G., Gao, G. F., & Tan, W. (2020). A novel coronavirus from patients with pneumonia in China, 2019. *New England Journal of Medicine*, 382(8), 727–733. https://doi.org/10.1056/nejmoa2001017

Навчальне видання

ШТУЧНИЙ ІНТЕЛЕКТ ТА МАШИННЕ НАВЧАННЯ

Методичні рекомендації

Укладачі: **Шебаніна** Олена В’ячеславівна
Тищенко Світлана Іванівна
Пархоменко Олександр Юрійович
Жебко Олександр Олегович
Коломієць Андрій Миколайович

Формат 60х84 1/16. Ум. друк. арк. 2.94.
Наклад 50 прим. Зам. № _____

Надруковано у видавничому відділі
Миколаївського національного аграрного університету
54020, м. Миколаїв, вул. Георгія Гонгадзе, 9

Свідоцтво суб’єкта видавничої справи ДК № 4490 від 20.02.2013