МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

МАТЕМАТИЧНІ МЕТОДИ І МОДЕЛІ РИНКОВОЇ ЕКОНОМІКИ

Методичні рекомендації до лабораторних робіт для студентів спеціальності 051 "Економіка" другого (магістерського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2019

УДК 330.4(07.034)
М34

Укладач О. В. Івахненко

Затверджено на засіданні кафедри економічної кібернетики.
Протокол № 9 від 28.12.2018 р.

Самостійне електронне текстове мережеве видання

Математичні методи і моделі ринкової економіки [Електрон-
М34 ний ресурс] : методичні рекомендації до лабораторних робіт для студентів спеціальності 051 "Економіка" другого (магістерського) рівня / уклад. О. В. Івахненко. – Харків : ХНЕУ ім. С. Кузнеця, 2019. – 65 с.

Розглянуто основні питання підгонки моделей статистичного навчання мовою R. Подано завдання до лабораторних робіт із навчальної дисципліни та наведено методичні рекомендації до їхнього виконання.

Рекомендовано для студентів спеціальності 051 "Економіка" другого (магістерського) рівня.

УДК 330.4(07.034)

Загальні положення

Лабораторні роботи призначені для закріплення теоретичного й практичного матеріалу, набуття навичок побудови статистичних моделей ринкової економіки в середовищі RStudio мовою програмування R.

Для виконання лабораторних робіт пропонується використовувати R. Це й мова програмування, й розрахункове середовище для статистичних обчислень, бізнес-аналізу та візуалізації даних. R доступний під ліцензією GNU GPL. Код було відкрито в 1995 році. Подальший розвиток функціональності R здійснюється за допомогою пакетів, які розробляють окремі дослідники. R-пакет – це колекція наборів даних, функцій мови R, документації та динамічно завантажувальних елементів.

Для швидкості та зручності побудови моделей пропонується використовувати безоплатне інтегроване середовище розробки для мови R – RStudio.

Кожна лабораторна містить мету, завдання та методичні рекомендації до її виконання. Методичні рекомендації до виконання окремої лабораторної роботи містять необхідний перелік команд та пакетів мовою R для вирішення поставленого завдання, а також коментарі та рисунки до основного алгоритму для його виконання.

Лабораторні роботи стосуються основних тем дисципліни та ґрунтуються на теоретичному матеріалі, тож їхній рекомендовано виконувати послідовно та відповідно до лекційного матеріалу для кращого розуміння матеріалу дисципліни.

Для захисту лабораторної роботи студенту потрібно оформити індивідуальний звіт, що повинен містити: постановку завдання; скрипт мовою R з коментарями та ілюстраціями того, як він працює; аналіз результатів і висновки. Слід оформлювати звіт у текстовому редакторі, можливо дублювати в pdf-форматі.

Оцінка за виконання роботи ставиться за результати виконання й захисту роботи студентом.

Змістовий модуль 1
Особливості мови R для аналізу даних та моделювання в економіці

Лабораторна робота 1. Знайомство з R та RStudio

Мета – ознайомитися з організацією роботи з даними в середовищі розробки RStudio, вивчити основні класи, об'єкти та синтаксис мови R.

Завдання – ознайомитися з текстом методичних рекомендацій і проілюструвати їх, створивши скрипт у середовищі RStudio, за всіма операціями з різними об'єктами в R, які наведені в методичних рекомендаціях.

Методичні рекомендації

Прикладні пакети мови R можна завантажувати з сайта проекту R чи із сайта CRAN (Comprehensive Archives Network for R – всеосяжна мережа архівів для *R)*.

Працювати з R більш зручно в *RStudio*. Це безоплатне інтегроване середовище розробки (IDE) для R. Інсталятор *RStudio* треба завантажити з офіційного сайта проекту.

У *RStudio* є чотири базових вікна (рис. 1.1).

"1" – Консоль (*Console*). Тут пишемо команди й відразу отримуємо результат їхньої реалізації.

Натискання клавіші *Enter* веде до виконання введеної команди. Команди відокремлюються новим рядком. Новий рядок починається з позначки >. Знайти нещодавно використовувану команду – Ctrl + Вгору.

"2" – Редактор коду (Source). Тут можна набирати й редагувати код на R, зберігати те, що отримали у вигляді скриптів або функцій для подальшого багаторазового використання.

Запустити на виконання можна як окремий рядок, так і виділений шматок коду, або безпосередньо скопіювавши його в консоль, або натиснувши кнопки *Run* або *Re-run* у верхньому правому куті вікна редактора.

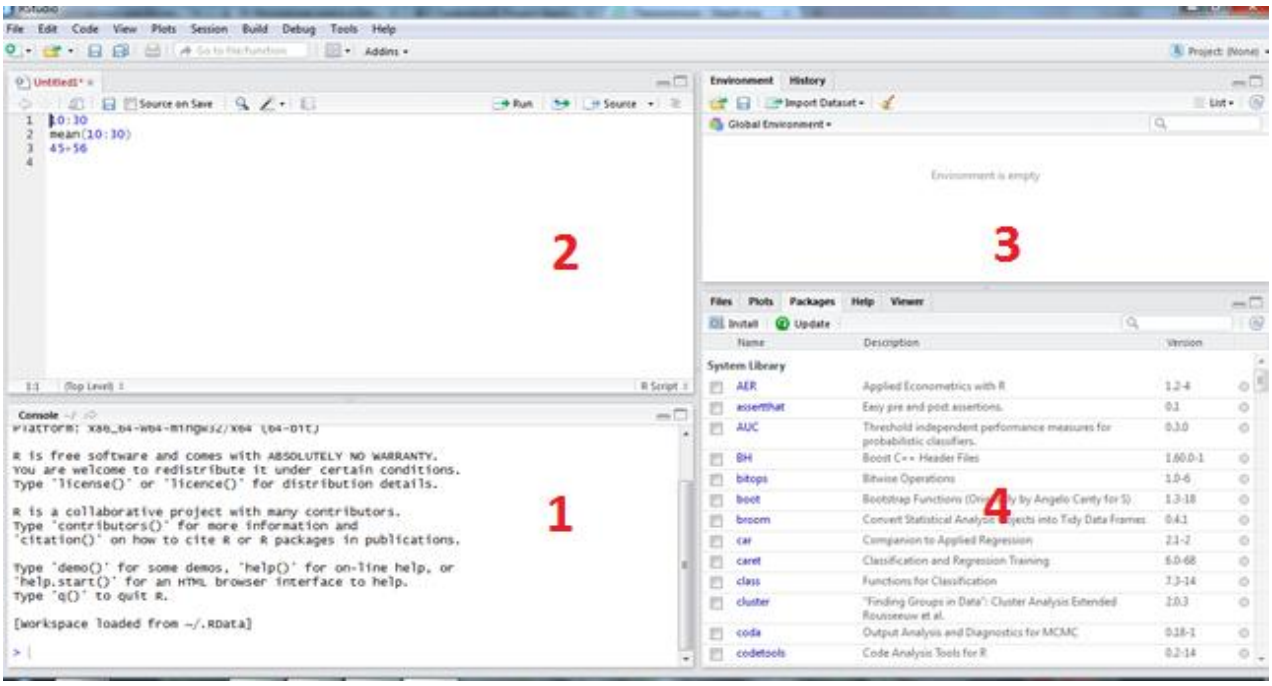


Рис. 1.1. **Вікна *RStudio***

Закінчений шматок коду можна зберегти як скрипт.

Автоматичне створення функцій можливо шляхом виділення шматка коду в редакторі та вибору *Extract function* (рис. 1.2). Усі безпосередньо не задані у виділеному шматку коду змінні автоматично розпізнаються як аргументи нової функції.

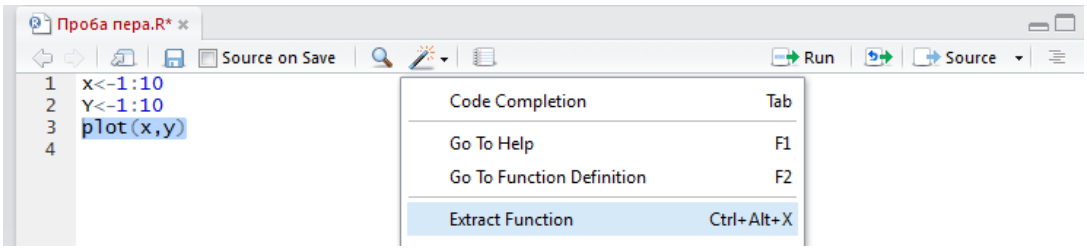


Рис. 1.2. **Збереження функцій**

Додаються коментарі до рядків коду після позначки #. Наприклад:

plot(x,y) #точковий графік двох змінних x та y.

"3" – Вікно з двома вкладеннями *Environment* та *History*.

В *Environment* зберігаються всі створені нами об'єкти: змінні, підвантажені дані, функції. Можна легко знайти потрібні об'єкти для повторного використання або перегляду значень.

У *History* зберігається вся історія нашої роботи в поточному сеансі. Також можна використовувати для навігації, надсилати виділені частини вже реалізованого коду в редактор або в консоль (відповідними кнопками *To Console*, *To Source*).

"4" – Вікно з п'ятьма вкладеннями *Files*, *Plots*, *Packages*, *Help*, *Viewer*.

Files – файл-менеджер, який дозволяє переходити з теки в теку, не залишаючи RStudio, відкривати потрібні файли даних.

Plots – тут будуть відображатися результати виконання всіх наших команд, пов'язаних із візуалізацією даних. Побудовані графіки можна відразу зберігати як картинку або pdf, збільшувати, публікувати.

Packages – вкладення бібліотек (пакетів), присвячених окремим групам команд в R. Щоб використовувати в *Console* якусь команду, необхідно спочатку довантажити пакет, що її містить. Довантажити пакет можна, або задавши відповідну команду *library ("назва пакета")* в консолі, або поставивши галочку навпроти пакета у вкладенні *Packages*.

Help – це довідка щодо різних функцій. Щоб викликати довідку щодо функції, можна або задати її у вікні пошуку безпосередньо на вкладці *Help*, або в консолі набрати її назву після знаку питання. Наприклад, довідка щодо функції plot може бути отримана як "*?Plot"*. У результаті отримаємо опис функції, її основні аргументи й приклади застосування.

Viewer дозволяє переглядати файли даних у вигляді звичних табличок.

Для зручності роботи слід установити робочу директорію, до якої буде звертатися R, коли йому знадобитися відкрити або зберегти певні файли (дані, скрипти ...).

Змінити робочу директорію в RStudio можливо:

через меню (Session -> Set Working Directory -> Choose Directory...);

командою setwd ("D: / R / РОБОЧА ДИРЕКТОРІЯ))"), аргументом якої буде адреса нової робочої директорії;

за допомогою гарячих клавіш Ctrl + Shift_H (рис. 1.3).

За замовчуванням R буде звертатися до цієї директорії. Якщо потрібний файл знаходиться не в ній, можна або перемістити його в цю директорію, або під час звернення до файла прописувати повну адресу доступу до нього. Наприклад, read.csv ("C: /Users/username/documents/evals.csv") – відкрити csv файл, який знаходиться за адресою C: /Users/username/documents/evals.csv.

Відкриття / закриття файлів можливо через меню *File* або командою в консолі.

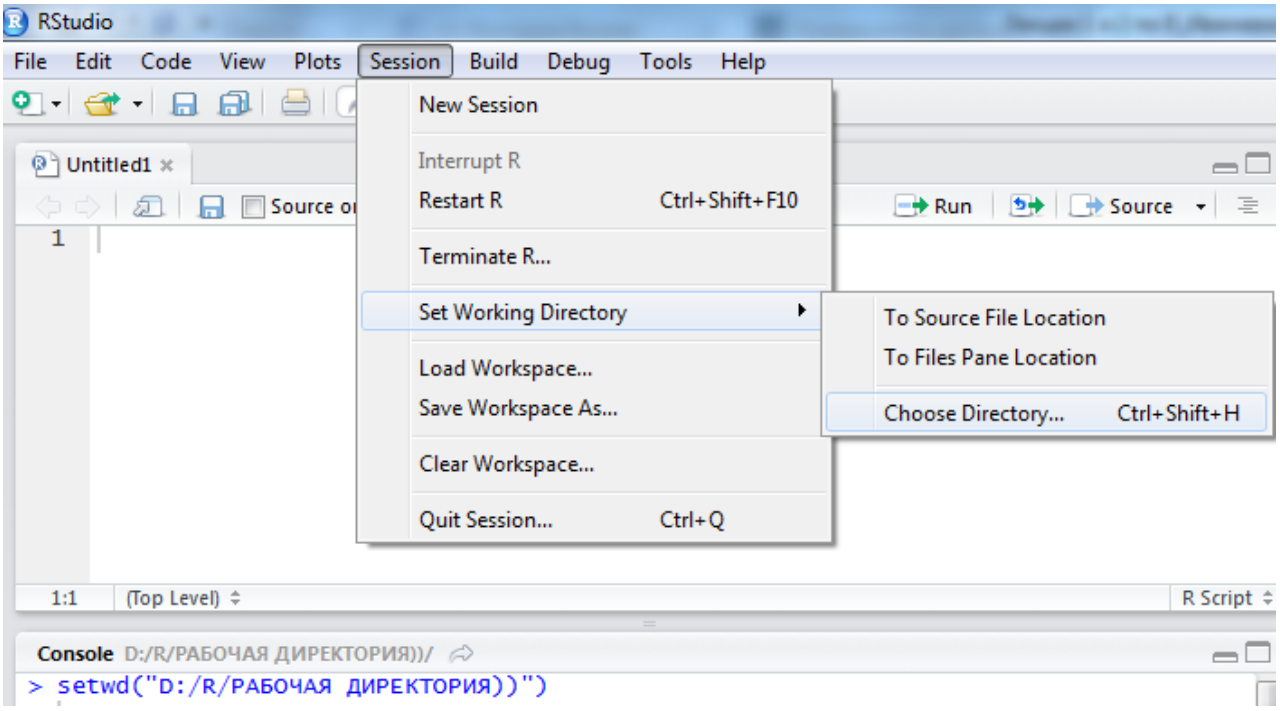


Рис. 1.3. **Встановити робочу директорію**

Й історія, й робочий простір зберігаються окремими файлами. А також їх можна відкрити з меню *File*. Однак частіше ми будемо відкривати й зберігати скрипти (готові до "експлуатації" закінчені шматки відредагованого коду) та файли даних. Скрипти зберігаються з розширенням *".R"*.

Файли даних R відкриває практично будь-які. Найчастіше використовуються текстові формати .csv (comma separated values) і .txt (tab-delimited files).

Файли даних можна відкривати за допомогою функцій: read.table (), read.csv () і подібних, і за допомогою меню.

Зверніть увагу, що під час використання команд ім'я файла треба брати в лапки (наприклад, read.csv ("evals.csv")). Також можна відразу зчитувати файл у змінну (d <-read.csv ("evals.csv")) для зручності подальшої роботи з ним.

Функція read.table () є базовою. Виклик довідки за нею "?Read.table" дозволяє побачити всі можливі її аргументи, з яких найцікавіші header = (TRUE – якщо в першому рядку містяться назви стовпців, FALSE – якщо немає, це значення за замовчуванням), sep = "," (в лапках задається роздільник стовпчиків даних), dec = "." (у лапках задається роздільник десяткових дробів).

Синтаксис R.

Як символ присвоєння в R використовується "<-". Вивести значення змінної на екран можна ввівши в консоль її назву й натиснувши *Enter*.

Наприклад, дамо змінній My_Var1 значення 45 і виведемо результат у консоль:

```
> My_var1 <- 45
> My_var1
[1] 45
```

Знак присвоєння <- буде R використовуватися всюди, практично в кожному рядку коду. Вставити ці символи в код можна комбінацією клавіш *Alt* -.

Для реалізації логічних операцій зі змінними в R використовуються символи:

<(Менше),

> (Більше),

== (дорівнює),

! = (Не дорівнює).

Символ = не є знаком рівності, але може бути використаний для присвоєння значення змінної, як і <-, хоча це й не прийнято.

Результат виконання логічної операції можна зберігати в нову змінну:

```
> Var1<-120 #привласнити змінній var1 значення 120
> Var2<-200 # привласнити змінній var2 значення 200
> Var3<-Var2!=Var1 # привласнити змінній var3 результат виконання логічної операції
> Var3 #вивести до консолі значення var3
[1] TRUE
```

Спецсимволи R:

NA – Not Available (наприклад, пропущені значення);

NaN – Not a Number ("не число", знак невизначеності, наприклад, 0/0);

Inf – нескінченість (1/0);

Inf – мінус нескінченість (-1/0).

Роздільником розрядів у R є точка. Аргументи функцій відокремлюються комою.

Об'єкти в R:

– *Vectors* – вектори (одновимірні масиви одного класу);

– *Factors* – фактори (вектори з визначеними рівнями);

– *Matrix* – матриці (двовимірні масиви одного класу);

– *Arrays* – масиви одного класу (можуть бути й більшої розмірності);

– *List* – список (масив із різних класів, вектор об'єктів у R);

– *Dataframe* – структура (специфічний вид списку *List*, усі елементи якого мають однакову довжину – грубо кажучи, звичайна таблиця різнотипних даних).

Усі об'єкти в R мають свій клас, який описує що вміщує цей об'єкт та що з ним може робити та чи інша функція:

> class(object) #визначити клас об'єкта;

> as._____(x) #змінити клас об'єкта (наприклад, as.numeric(x));

> is._____(x) #перевірити клас об'єкта (наприклад, is.numeric(x)).

```
Приклад 1:
> x<-1:6            #створили вектор числових значень
> x
[1] 1 2 3 4 5 6
> class(x)
[1] "integer"                 #клас довге ціле (number – коротке ціле)
> as.numeric(x)               #вивести як числовий вектор
 [1] 1 2 3 4 5 6
> as.character(x)
[1] "1" "2" "3" "4" "5" "6" #вивести як категоріальний вектор
> is.numeric(x)               #перевірити чи є клас об'єкта числовим
[1] TRUE
```

Не обов'язково клас – це numeric, character, logical і т. д., класом об'єкта може бути й *list*.

Цікаво усвідомити також різницю між класом і типом об'єкта в R:

> typeof(object) #визначити тип об'єкта.

Створення, індексація та особливості об'єктів у R.

Вектор. Задати зміну типу вектора можна за допомогою функції *c()* (від combine – скомбінувати) та *vector().*

Приклад 1: вектора числових значень, характеристик, комплексних чисел, логічно

```
> var1<-c(1,5,-2,5,-56)
> var1
[1]   1   5  -2   5 -56
> var2<-c("a","b","c")
> var2
[1] "a" "b" "c"
```

```
> var3<-c(2+0i,3+4i)
> var3
[1] 2+0i 3+4i
> var4<-c(T,F,T) #короткий та довгий запис TRUE і T, FALSE і F рівнозначні
> var4
[1]  TRUE FALSE  TRUE
```

Приклад 2: отримання послідовності даних від 5 до 25

```
> var1<-c(5:25) #запис x:y – задає послідовність від x до y
#чи
> var1<-5:25
#чи
> var1<-seq(5,25,by=1) #функція seq – задає послідовність із кроком, що дорівнює аргументу by
> var1
[1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

Приклад 3: створення "порожніх" векторів із заданим типом

```
> var1<-vector("numeric",length = 10)
> var1
 [1] 0 0 0 0 0 0 0 0 0 0
> var2<-vector("logical",length = 5)
> var2
[1] FALSE FALSE FALSE FALSE FALSE
> var1<-vector("integer",length = 0)
> var1
integer(0)
```

Найчастіше необхідно звернутися до окремого елемента вектора. Зробити це можна, використовуючи наступний запис Вектор [номер елемента].

Слід зазначити, що в R індексація елементів йде з одиниці, не з нуля, як в деяких інших мовах програмування.

Приклад 1: вивести на екран другий елемент вектора var1

```
> var1<-c(7:10)
> var1[2]
[1] 8
```

Приклад 2: змінити перший елемент вектора з 7 на 5

```
> var1
[1]  7  8  9 10
> var1[1]<-5
> var1
[1]  5  8  9 10
```

Приклад 3: вивести значення з другого по четверте

```
> var1[2:4]
[1]  8  9 10
              # ЧИ
> var1[c(2,3,4)]
[1]  8  9 10
> var1[2,3,4]      #так не можна
```

Приклад 4: нехай var2 складається з непарних елементів var1 та чисел з 20 по 25

```
var1<-1:10
> var2<-c(var1[c(1,3,5,7,9)], 20:25)
> var2
[1]  1  3  5  7  9 20 21 22 23 24 25
```

З векторами можна робити всі можливі арифметичні операції. Усі операції роблять поелементно, тобто якщо var1 – це вектор, то команда var1 + 10 додасть 10 до кожного елемента вектора (паралельні обчислення):

```
> var1<-1:10
> var1+10
[1] 11 12 13 14 15 16 17 18 19 20
```

Використання логічних операцій з векторами також можливо, та є основою для більш плідної роботи з даними. Розглянемо приклад: необхідно вивести з наявного вектора vector1 тільки позитивні значення. Якщо ми проведемо з вектором логічну операцію vector1>0, результатом буде вектор значень тієї ж розмірності, що й основний вектор, але складається зі значень TRUE або FALSE. Цей вектор зі свого боку можна використовувати для індексації виведення позитивних значень першого вектора:

```
> vector1<-c(1,-5,-6,5,7,10,-1)
> vector2<-vector1>0
> vector2
[1]  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE
> vector1[vector2]
[1]  1  5  7 10
```

або простіше

```
> vector1<-c(1,-5,-6,5,7,10,-1)
> vector1[vector1>0]
[1]  1  5  7 10
```

Іноді може знадобитися об'єднати кілька логічних умов в одне ціле. Це робиться за допомогою символу об'єднання &.

Приклад: потрібні тільки позитивні елементи вектора, та такі, що не перевищують 6

```
> vector1<-c(1,-5,-6,5,7,10,-1)
> vector1[vector1>0 & vector1<6]
[1] 1 5
```

Назви спостереженням, що утворюють вектор, можна задати функцією names():

```
> y<-1:4
> names(y) <- c("a","b","c","d")
> y
a b c d
1 2 3 4
```

Фактори – це вектори, що мають додатковий атрибут – рівні (*levels*). Зрозуміло, що в рівнів можуть бути назви (*labels*), і легше оперувати саме ними, а не кодуваннями змінної. Фактори також можуть бути впорядкованими (ordered).

```
Приклад: зробимо з вектора фактор
> var1<-c(2,4,5,4,2,2,4,4,4,4,5)          #створили вектор
> length(var1)                             #довжина вектора
[1] 11
> var2<-as.factor(var1)          #зберегли як фактор
> attributes(var2)               #подивились атрибути
$levels
[1] "2" "4" "5"
$class
[1] "factor"
```

За замовчуванням рівні втягне сам R, але якщо необхідно, щоб базовим був певний рівень, можна додатково це прописати – дивимося приклад. Базовим буде перший з описаних під час створення фактора рівнів.

Фактор можна згрупувати за рівнями за допомогою функції table().

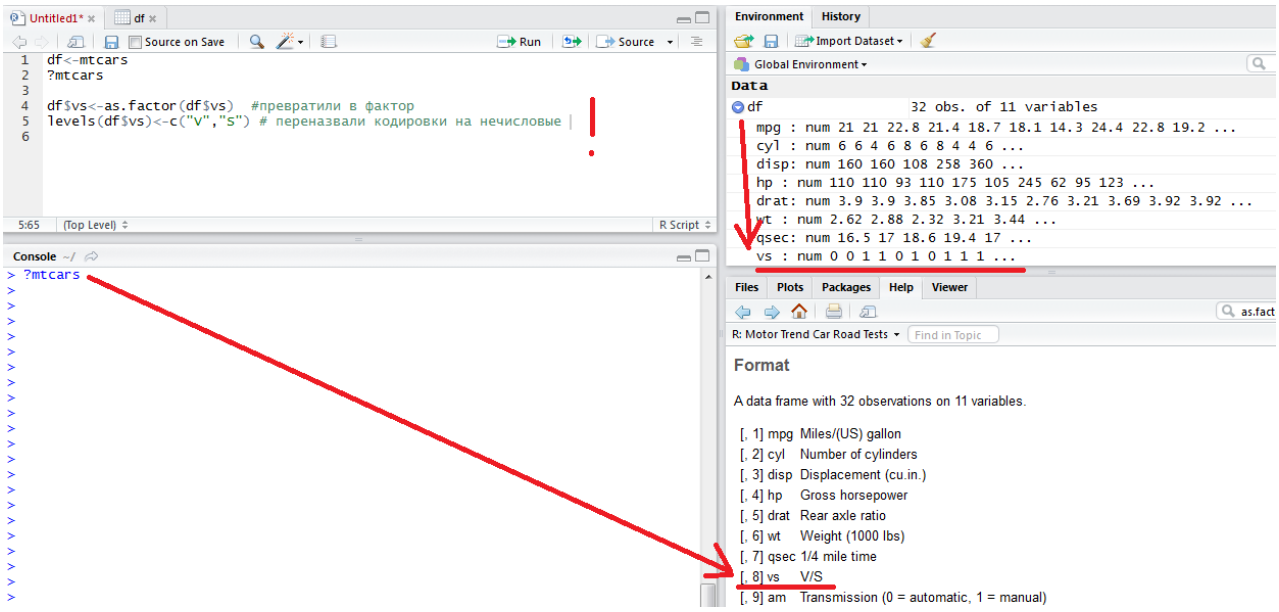
```
Приклад: levels
> x<-factor(c("yes","yes","no","yes","no"),levels=c("no","yes"))
> x[1] yes yes no  yes no
Levels: no yes
> table(x)
x
 no yes
  2   3
```

Бувають випадки, коли дані втягувалися в закодованому вигляді (наприклад, 0 і 1), а розшифрування кодування присутнє тільки в описі даних. Показник виглядає і є для R неякісним фактором, а звичайним числовим вектором. Для зручності роботи краще його "привести до порядку": призначити зрозумілі імена рівням, перевести в формат фактора інтерпретованих назв – (див. приклад на рис. 1.4 а і б).

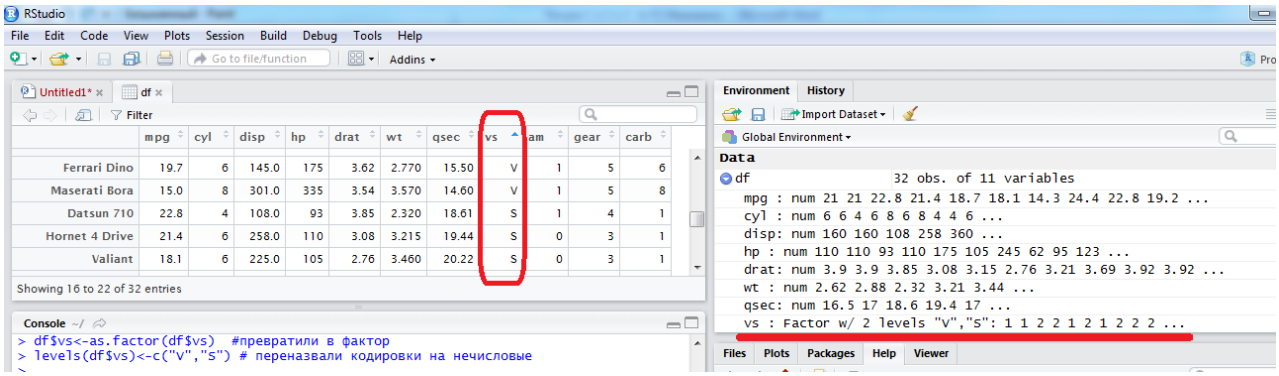
Приклад:

> df$vs<-as.factor(df$vs) #перетворили на фактор

> levels(df$vs)<-c("V","S") # перейменували кодування на нечислові



а



б

Рис. 1.4. **Перепишемо змінну як фактор**

Імена спостереженням для фактора також додаються функцією names().

Матриці *matrix.* Задати порожню матрицю можна за допомогою функції matrix (nrow =, ncol =), де nrow – число рядків, ncol – число стовпців.

Матриці мають додатковий атрибут – розмірність. Функція для виведення розмірності матриці (і будь-якого іншого об'єкта в R) – *dim (назва об'єкта)*.

```
> x<-matrix(1:6,nrow=2,ncol = 3) #створимо матрицю 2 на 3 з числами від 1 до 6
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> attributes(x)    #дізнатися атрибути об'єкта
$dim
[1] 2 3
> dim(x)           #розмірність об'єкта
[1] 2 3
```

Додавши атрибут розмірності до вектора, можна теж створити матрицю. Тому, як і вектор, і матриця – все це масиви різної розмірності, й, відповідно, змінюючи розмірність ми, міняємо тип об'єкта в R.

```
> x<-c(1:6)
> dim(x)<-c(2,3)          #додаємо атрибут розмірності
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> attributes(x)
$dim
[1] 2 3
```

Задати матрицю можна достроково та за стовпцями за допомогою функцій cbind та rbind, відповідно (як і будь-який інший об'єкт в R).

```
x<-1:3
> y<-10:12
> cbind(x,y)
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(y,x)
  [,1] [,2] [,3]
y   10   11   12
x    1    2    3
```

Обов'язкові атрибути функції *matrix* – це число її рядків та стовпців (*nrow=, ncol=*), також можна вказати джерело даних для заповнення (*data=назва об'єкта-джерела даних*), заповнювати за рядками чи стовпцями (*byrow = TRUE чи FALSE*), вказати джерело назв рядків та стовпців (*dimnames = List с назвами*).

```
> x<-c(1:6)
>
> m1<-matrix(data=x,nrow=2,ncol=3,byrow = FALSE)
> m1
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> m2<-matrix(data=x,nrow=2,ncol=3,byrow = TRUE)
> m2
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> name_1<-c("перший рядок", "другий рядок")
> name_2<-c("перший стовпець", "другий стовпець", "третій стовпець")
> y<-list(name_1,name_2)
> m3<-matrix(data=x,nrow=2,ncol=3,byrow = TRUE, dimnames = y)
> m3
             перший стовпець другий стовпець третій стовпець
перший рядок               1               2               3
другий рядок               4               5               6
```

Перевірити чи є об'єкт матрицею, чи вивести як матрицю можна за допомогою команд as._____ та is._____, відповідно (як і для інших об'єктів) – див. приклад.

```
> x<-c(1:6)
 > m<-as.matrix(x)  #створюємо з вектора матрицю
> m
     [,1]
[1,]    1
[2,]    2
```

```
[3,]    3
[4,]    4
[5,]    5
[6,]    6
> is.matrix(m)     #перевіряємо чи це матриця
[1] TRUE
```

Звернутися до елемента матриці можна так:

```
> m3              #сама матриця
          перший стовпець другий стовпець третій стовпець
перший рядок          1          2          3
другий рядок          4          5          6
> m3[2,1]         #елемент 2,1
[1] 4
> m3[2,] #другий рядок
перший стовпець другий стовпець третій стовпець
                      4          5          6
> m3[,3] #третій стовпець
перший рядок другий рядок
                      3          6
```

Нехай є дві матриці X та Y.

Поелементне множення і ділення елементів матриць проводитися за допомогою команд X*Y; X/Y, відповідно.

Перемноження й розподіл матриць математично правильно (матриці перемножуються рядок на стовпець, нагадуємо) здійснюється в такій спосіб X%*% , X%/%Y.

Транспонувати матрицю можна за командою t (X) або crossroad (X).

```
> X<-matrix(1:6,nrow=2,ncol=3)
> Y<-matrix(2:7,nrow=3,ncol=2)
> X;Y;
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
     [,1] [,2]
[1,]    2    5
[2,]    3    6
[3,]    4    7
> X%*%Y                    #перемноження матриц
     [,1] [,2]
[1,]   31   58
[2,]   40   76
> t(X)              #и crossroad(X) також транспонувати матрицю X
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

Масиви *Arrays* задаються командою *array*. Можна задавати кількість вимірювань більше 2. Індексуються відповідно. як і матриці – див. приклад.

```
> x<-1:8
> A<-array(x,c(2,4))                #створимо двовимірний масив
> A
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
> A[2,3]                    #виведемо елемент 2,3
[1] 6

> A1<-array(x,c(2,2,2)) #створимо тривимірний масив – (кубик 2*2*2)
> A1[2,,]                   #отримаємо грань куба
[,1] [,2]
[1,]    2    6
[2,]    4    8
> A1[2,2,2]                         #виведемо елемент 2.2.2
[1] 8
```

Імена вимірювань у масиві задаються аналогічно варіанту для матриць за допомогою атрибута dimnames.

```
> dimnames(A) <- list(c("a","b"),c("q","w","e","r"))
> A
  q w e r
a 1 3 5 7
b 2 4 6 8
```

Списки *List* – є своєрідними масивами, точніше вектором з інших об'єктів. Цими об'єктами можуть бути вектори, матриці, константи.

Задається список командою list().

```
> x<-1:6
> y<-c(T,F,T,T)
> z<-matrix(20:25,nrow=2,ncol=3)
> l<-list(x,y,z)
> l
[[1]]
[1] 1 2 3 4 5 6
[[2]]
[1]  TRUE FALSE  TRUE  TRUE
[[3]]
     [,1] [,2] [,3]
[1,]   20   22   24
[2,]   21   23   25
```

Індексація у списку дещо інша, оскільки слід вказати спочатку номер об'єкта, а потім внутрішню індексацію в ньому.

```
Приклад за даними попереднього
> l[[1]]            #вивести перший об'єкт списку
[1] 1 2 3 4 5 6
> l[[2]][3]         #вивести третій елемент другого об'єкта
[1] TRUE
> l[[3]][2,3]       #вивести елементи 2 та 3 третього об'єкта – матриці
[1] 25
```

Датафрейми – це таблиці даних *dataframe.*

Вершиною еволюції, з точки зору роботи з даними, будуть структури даних, вони ж таблиці даних – *dataframes*.

З одного боку, *dataframe* може бути інтерпретований як підклас списку *List*. Але тільки такий, у якому всі об'єкти списку мають однакову довжину. Як і список датафрейм може об'єднувати дані різних типів (кількісні, якісні, бінарні ...). І це його помітно відрізняє від масивів, матриць і векторів.

З іншого – це звична таблиця даних, в рядках якої записуються окремі випадки (cases), а стовпцями є якісь фактори.

Задати датафрейм командою *data.frame*, аргументи якої відповідно імена об'єктів, найчастіше векторів або факторів, які нам потрібні.

Приклад

```
> a<-c(20,16,45,34)
> n<-c("oleg","olga","nasty","stas")
> w<-c(45,46,60,100)
> d<-data.frame(name=n,age=a,Weight=w)
> d
   name age Weight
1  oleg  20     45
2  olga  16     46
3 nasty  45     60
4  stas  34    100
```

Переглянути таблицю даних у звичному вигляді можна за допомогою команди View (), або клацнувши у вікні Environment на відповідному об'єкті (в обох випадках показується перша 1000 спостережень). Дивіться рис. 1.5.

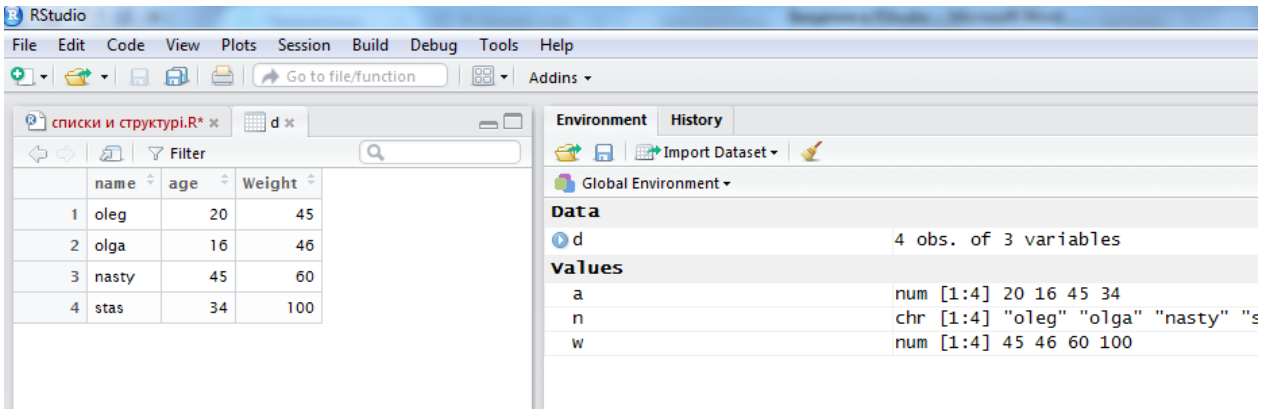


Рис. 1.5. **Перегляд таблиці даних**

У вікні перегляду даних натискання на заголовки стовпця дозволяє впорядкувати всі спостереження за зростанням / спаданням значень саме цього стовпчика.

Прочитати дані у вигляді матриці, а не таблиці даних допоможе функція *data.matrix ():*

```
> data.matrix(d)
     name age Weight
[1,]    2  20     45
[2,]    3  16     46
[3,]    1  45     60
[4,]    4  34    100
```

Лабораторна робота 2. Робота з таблицями даних у R

Мета – навчитися розраховувати описові характеристики даних у R, впорядковувати інформацію, створювати нові змінні у датафреймі та використовувати цикли щодо роботи з даними.

Завдання *(за курсом "Аналіз даних у R" від інституту Біоінформатики платформи Stepic):*

Завдання 1. У вбудованому в R датафреймі *mtcars* створіть нову змінну even_gear, заповнить її одиницями, якщо значення змінної gear для спостереження парне, та нулями, якщо непарне. Підказка – використовуйте оператор ділення із залишком: 5%%2==1 TRUE

Завдання 2. Створити змінну-вектор mpg_4 и зберегти в неї значення витрат палива (mpg) для машин із 4 циліндрами (cyl).

Завдання 3. Створить новий dataframe під назвою mini_mtcars, у якому будуть збережені тільки третій, сьомий, десятий, дванадцятий та останній рядок датафрейма mtcars.

Завдання 4. Яка команда створить сабсет mtcars, тільки для тих автомобілів, в яких число циліндрів (cyl) не дорівнює 3, та час розгону автомобіля (qsec) більше середнього за вибіркою?

Завдання 5. У датасеті оцінок викладачів створить нову змінну quality. Та, використовуючи комбінацію циклів if та for, заповнить цю змінну якісними оцінками діяльності викладачів. Якщо оцінка викладача score перевищує 4, якість викладання буде дорівнювати 'good', якщо менше, то 'bad'.

Завдання 6. Зробить ту ж операцію, але використовуючи функцію ifelse. Результат збережіть у нову змінну quality2. Переконайтесь, що змінні тотожні.

Завдання 7. Створить нову числову змінну new_var у даних mtcars, яка містить одиниці в рядках, якщо в машині не менше 4 карбюраторів (змінна "carb") чи більше 6 циліндрів (змінна "cyl"). У рядках, в яких умова не виконується, повинні бути нулі.

Завдання 8. В існуючій змінній my_vector зберегти вектор з 50 чисел. Вирішить завдання, використовуючи конструкцію: if () {} else {}. Якщо середнє значення вектора my_vector більше 20, у змінну result збережіть "My mean is great", якщо інше, то в змінну result збережіть рядок "My mean is not so great".

Завдання 9. Убудовані в R дані AirPassengers – це новий формат даних типу Time-Series. Вивчить структуру цих даних, перш ніж вирішувати завдання! Наприклад через команди:

```
> ?AirPassengers # справка щодо даних
> str(AirPassengers) # структура даних
```

Потрібно створити змінну good_months та зберегти в неї кількість пасажирів тільки в тих місяцях, у яких ця кількість більша, ніж показник попереднього місяця.

Методичні рекомендації

Будемо використовувати дані файла https://stepic.org/media/attachments/lesson/11481/evals.csv, які містять інформацію щодо факторів, що впливають на оцінювання студентом викладача.

Щоб почати роботу з даними їхній необхідно завантажити в робочу директорію та зберегти до змінної. Класом змінної є data frame:

```
> d<-read.csv('evals.csv')
> View(d)        #дозволяє побачити перші 1000 спостережень
> class(d)
[1] «data.frame»
```

Зручні команди для попереднього аналізу даних:

```
> head() #6 перших рядків та заголовки
> tail() #6 останніх рядків та заголовки
> head(,n) #n перших рядків та заголовки
```

```
> tail(,n) #n останніх рядків та заголовки
> str()   #структура даних
> summary(d) #узагальнені відомості щодо даних
> names(d) #повертає вектор назв
> nrow(d) #повертає кількість рядків
> ncol(d) #повертає кількість стовпців
```

Найбільш інформативною є функція (str ()). Вона виводить кількість спостережень і змінних у розглянутому data frame (та в будь-якому іншому об'єкті R), клас і тип кожної змінної, кількість рівнів для факторів і перші 10 значень кожної змінної в якості прикладу.

Зверніть увагу, що за замовчуванням функція read.csv буде зчитувати як фактор усі категоріальні змінні.

Також структуру не за допомогою функції можна подивитися у вікні Environment, натиснувши на синю позначку перед назвою data frame.

Приклад фрагмент структури даних датафрейму d

```
> str(d)   #структура даних
'data.frame':    463 obs. of  21 variables:
 $ score        : num  4.7 4.1 3.9 4.8 4.6 4.3 2.8 4.1 3.4 4.5 ...
 $ rank         : Factor w/ 3 levels "teaching","tenure track",..: 2 2 2 2 3 3 3 3 3 3
...
 $ ethnicity    : Factor w/ 2 levels "minority","not minority": 1 1 1 1 2 2 2 2 2 2
...
 $ gender       : Factor w/ 2 levels "female","male": 1 1 1 1 2 2 2 2 2 1 ...
 $ language     : Factor w/ 2 levels "english","non-english": 1 1 1 1 1 1 1 1 1 1
...
```

Ще одна функція, що дозволяє розглянути, що за дані знаходяться у відкритому файлі, це функція summary (). Вона дає нам узагальнені дані. Для чинників вона діє аналогічно функції table (), групуючи та підраховуючи кількість значень за рівнями; для кількісних даних обчислює основні описові статистики (мінімум, максимум, квартили, середнє).

```
> summary(d)
   score                rank           ethnicity      gender
Min.   :2.300   teaching    :102   minority    : 64   female:195
```

```
1st Qu.:3.800   tenure track:108   not minority:399   male  :268
Median :4.300   tenured     :253
Mean   :4.175
3rd Qu.:4.600
Max.   :5.000
      …
```

Щоб звернутися до змінної з датафрейма, пишеться його назва, потім знак $, а потім назва самої змінної. Натискання клавіші Tab, дозволяє вибрати зі списку існуючих змінних потрібну, щоб ми не помилилися в наборі.

Змінні датафрейма виводяться векторами, тому з ними можна робити всі операції, які з векторами можливі (складати, множити, зберігати в іншу змінну і т. д.). Наприклад, можна вивести узагальнюючі характеристики даних не в усьому датафреймі, а за окремою змінною:

```
> summary(d$score)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.300   3.800   4.300   4.175   4.600   5.000
```

Можна створити нову змінну в датафреймі, якщо скомбінувати дані з існуючої змінної, та задати її вручну:

```
> d$new<-d$score^2          #у новій змінній оцінки будуть за 10-бальною шкалою
> d$nol<-0                  #в новій змінній будуть всі 0
> d$number<-1:nrow(d)       #в новій змінній будуть значення від 1 до кінця таблиці
```

Розглянемо питання індексації докладніше. Датафрейм є двовимірним об'єктом, тому звернутися до будь-якого елемента можна класично: "*назва датафрейма[i, j]"* (слід тільки пам'ятати, що перший індекс – це завжди рядок).

Але можна, знаючи смислове навантаження даних, і використовувати звернення через змінні:

```
        Приклад: записи дають аналогічний результат
> d$score[3] #звертається до 3 спостережень змінної score (змінна перша)
[1] 3.9
> d[3,1]    #звертаємося до елемента 3,1
```

```
[1] 3.9

> d$score[c(5,8,9)] #5,8 і 9 спостереження змінної score
[1] 4.6 4.1 3.4
> d[c(5,8,9),1]     #5,8 і 9 спостереження змінної score
[1] 4.6 4.1 3.4

> d[1,]             #бажаємо вивести перший рядок (перший викладач)
  score         rank ethnicity gender language age cls_perc_eval cls_did_eval
1   4.7 tenure track  minority female  english  36      55.81395          24
…
> d[,1]             #вивести перший рядок – змінна оцінок
  [1] 4.7 4.1 3.9 4.8 4.6 4.3 2.8 4.1 3 …

> head(d[,c(3,6,9)]) #вивести 6 перших спостережень в  3, 6 і 9 стовпцях
     ethnicity age cls_students
1     minority  36           43
2     minority  36          125
3     minority  36          125
4     minority  36          123
5 not minority  59           20
6 not minority  59           40
```

Індексація має цінність не сама собою, а як основа формування вибірок – підмножин потрібних нам даних.

Наприклад, у нас є змінна – стать респондента. Відповідно, вибірку даних можна розділити на дві підвибірки за ознакою статі.

Збережемо в окрему таблицю всі дані за чоловіками та окремо за жінками:

```
> male_d<-d[d$gender=="male",] #збережемо окремо вибірку за чоловіками
> female_d<-d[d$gender!="male",] # збережемо окремо вибірку за жінками
```

Формування підвибірок легко здійснити за допомогою функції subset:

```
> subset(d,gender=="male")  # окремо вибірка за чоловіками
> v<-subset(d,score>"4.9")   # збережемо окремо вибірку викладачів з оцінкою вище 4,9
```

Найчастіше необхідно здійснити зворотну операцію – з'єднати вибірки в єдиний масив даних. Для цього використовуються функції cbind (,) і rbin (,). Склеїти вибірки за стовпцями необхідно, наприклад, коли проведено додаткове дослідження й отримані нові спостереження, на які потрібно розширити існуючу вибірку (використовуємо cbin). Функція rbin склеює датафрейми за рядками (наприклад, якщо ми вже створили підвибірки за викладачами чоловіками та жінками, можемо склеїти їх в загальну вибірку, аналогічну тій, що була спочатку). Зверніть увагу на розмірність датафреймів d, d_male, d_female, d_m в Environment на рис. 2.1: в d 463 спостереження, з яких 195 жінок викладачів і 268 чоловіків, вибірки d і d_m відрізняються тільки порядком проходження спостережень.

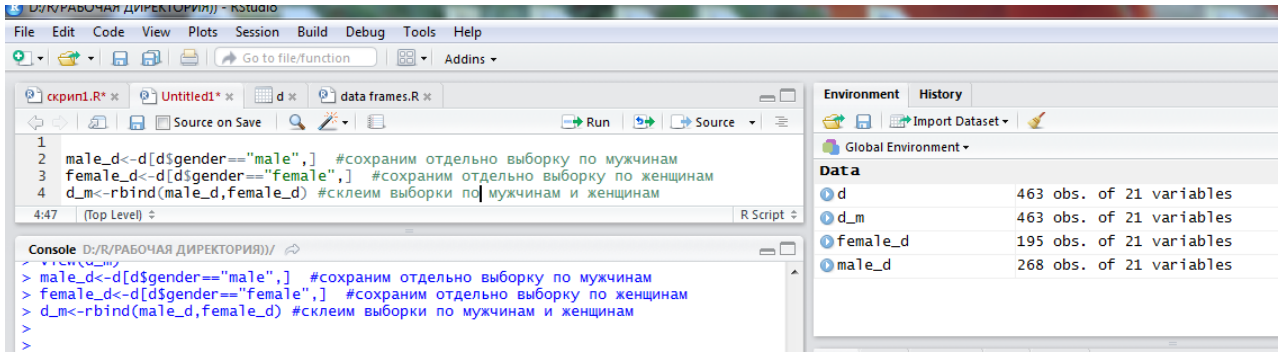


Рис. 2.1. **Об'єднання вибірок за рядками**

Одним із пакетів R є пакет datasets, що містить більше 80 датасетів (наборів даних), котрі характеризують різні сфери господарювання: від зростання апельсинового дерева, до кольору очей студентів, які вивчають статистику.

Подивитися, які дані є в базовому пакеті R, можна завдяки команді library (help = "datasets"). Звертатися до вбудованих наборів даних можна просто за найменуванням. Наприклад, щоб попрацювати з масивом даних, що характеризують машини cars, треба в консолі задати назву cars і натиснути enter.

Корисні команди для роботи з вбудованими даними:

```
> View(cars)      #проглядання датасета
```

```
> data(cars)      #додати датасет у робоче середовище (Environment)
> help(cars)      #вивести інформацію щодо датасета
```

Організація циклів у R. Під час роботи із циклами корисно пам'ятати наступні особливості роботи з R.

Змінні, які використовуються всередині циклів, краще задати до циклів. Наприклад, вектора можна задати порожніми й нульової довжини (x <-vector ("numeric", length = 0)), а можна заданої довжини (x<-numeric (10) # вектор довжини 10).

У циклах часто необхідно додати до змінної, найчастіше до вектора, нові значення. Це можна зробити за допомогою команди append (назва вектора, нові значення]), або за допомогою функції *c* (назва вектора, нові значення).

Команда break – перервати цикл. Команда next – пропустити ітерацію циклу. Команда return – вийти з функції.

```
> x<-1:3
> x<-append(x,4)
> x
[1] 1 2 3 4
> x<-c(x,5)
> x
[1] 1 2 3 4 5
```

Розглянемо наступні цикли:

– if ( ) { }else { } – умовний цикл;

– ifelse(, ,) – умовний цикл для векторів;

– for ( ) { } – цикл фіксованої кількості повторень;

– while ( ) {} – цикл до виконання умов;

– repeat {} – безкінечний цикл.

Реалізувати вибір дії за умовою можна за допомогою стандартного циклу if () {} else {}. У круглих дужках () пишуть умову, через виконання якої відбувається дія прописана у фігурних дужках перед else, інакше виконується дія в фігурних дужках після else.

Якщо ми хочемо виконати тільки одну дію в результаті вибору за умови можна фігурні дужки взагалі не ставити. Якщо хочемо зробити кілька умов послідовно – вкладаємо цикли один в інший.

```
        Приклад
> x<--5
> if (x>0){print('positive')}else{print('negative');x+3}  #дві дії за else
[1] "negative"
[1] -2
> if(x>0)print('positive')else print('negative')  #без фігурних дужок
[1] "negative"
> x<-0              #дві умови послідовно
> if (x>0){print('positive')} else if (x<0){print('negative')} else {print('0')}
[1] "0"
```

В R є більш зручний варіант циклу-умови – ifelse ().

По-перше, запис коротший на один рядок. Команду print можна не писати. Вкладеність циклів реалізується так само.

По-друге, команді на вхід можна подавати вектори:

```
> x<-0
> ifelse(x>0,'positive','not positive')
[1] "not positive"
> x<-c(1,-1,0)
> ifelse(x>0,'positive',ifelse(x<0,'negative','0'))
[1] "positive" "negative" "0"
```

3. Цикл for ( ) { } дозволяє повторювати дії, записані у фігурних дужках, доки виконується умова, задана в круглих дужках.

```
> for (i in 1:5) {print(i)}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
> d<-read.csv('evals.csv')  #та з масивами
> for (i in 1:5) {print('оцінка викладача); print(d$score[i])}
[1] "'оцінка викладача"
[1] 4.7
[1] "'оцінка викладача"
```

```
[1] 4.1
[1] "'оцінка викладача"
[1] 3.9
[1] "'оцінка викладача"
[1] 4.8
[1] "'оцінка викладача"
[1] 4.6
```

Можлива комбінація циклів for та if. Розв'яжемо завдання виведення оцінок підвибірки викладачів-чоловіків за даними масиву d за допомогою циклів та за допомогою формування підмножин (для порівняння):

```
> for (i in 1:nrow(d)){if (d$gender[i]=='male'){print(d$score[i])}}
[1] 4.6            # фрагмент вибірки
[1] 4.3
[1] 2.8
[1] 4.1
[1] 3.4
[1] 4.4 ….
# чи за допомогою subset
> m<-subset(d,gender=="male")  # окремо вибірка за чоловіками
> m$score
  [1] 4.6 4.3 2.8 4.1 3.4       # фрагмент вибірки
```

Наступний оператор циклу – while ( ) {}. Дія у фігурних дужках буде повторюватися до тих пір, доки виконується умова в круглих дужках.

Зверніть увагу, що для роботи цього циклу необхідно задати значення лічильника перед циклом. Крім того, це саме той оператор, який може створити безкінечний цикл – будьте уважні!

```
> i<-1            #важливо
> while(i<5){print(d$score[i])
+  i<-i+1
+ }
[1] 4.7
[1] 4.1
[1] 3.9
[1] 4.8
```

Лабораторна робота 3. Передоброблення даних, описові статистики та візуалізація

Мета – навчитися проводити первинну підготовку даних, розрахунок описових статистик та візуалізацію даних в R.

Завдання – за даними вбудованого в R пакету swiss потрібно візуалізувати дані за всіма соціально-економічними індикаторами у вигляді коробкових діаграм та графіків розсіювання, вивести основні описові статистики та зробити висновки.

Методичні рекомендації

Роботу з масивами даних потрібно починати з наступних кроків:

вивчити структуру, розмірність, правильність відображення даних (функції head, tail, str, names, nrow, ncol, view);

визначитися з підходом до пропущених даних (видалити чи замінити їхній);

трансформувати змінні, визначитися з признаковим простором, провести попередній аналіз даних (графіки та описові статистики);

прочистити дані на нелегітимні.

Робота з пропущеними і даними, що "не існують", Na і NaN може здійснюватися кількома способами. Найбільш поширений через функцію is.na – виводить логічний вектор, що відображає наявність значень в досліджуваних даних. Цей вектор можна використовувати для чищення даних.

```
Приклад
> x<-c(1,2,3,NA,4,NA,5,6,0/0,7,"l")
> x
 [1] "1"   "2"   "3"   NA    "4"   NA    "5"   "6"   "NaN" "7"   "l"
> is.na(x)     #логічний вектор, що перевіряє на наявність Na
 [1] FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE
> is.nan(x)    # логічний вектор, що перевіряє на наявність NaN
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
```

```
> x[!is.na(x)]    #тільки значення x, які не Na (Nan теж немає)
[1] 1 2 3 4 5 6 7
> x[!is.nan(x)]   #тільки значення x, які не NaN
 [1]  1  2  3 NA  4 NA  5  6  7 NA
> x[!is.na(x) & !is.nan(x)]        # x, які не Na та не NaN
[1] 1 2 3 4 5 6 7
> x<-x[!is.na(x)]          #оновимо вектор
> x
[1] 1 2 3 4 5 6 7
```

Найчастіше потрібно зберегти тільки ті дані, які є в наявності в декількох змінних одночасно. Для цього використовується функція complete.cases (). Її можна застосовувати як до окремих векторів, так і до датафрейма в цілому.

```
        Приклад для векторів
> x<-c(1,2,NA,4,5,NA,7)
> y<-c("a",NA,NA,"d","e","f","g")
>  good_cases<-complete.cases(x,y)        #обираємо спостереження, де є значення в обох векторах
> good_cases
[1]  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE
> x[good_cases]
[1] 1 4 5 7
> y[good_cases]
[1] "a" "d" "e" "g"

        Приклад для датафреймів
> str(airquality) #дивимося структуру вбудованого масиву даних airquality
'data.frame':    153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
> good_cases<-complete.cases(airquality) #обираємо номери спостережень, для яких є значення для всіх змінних датафрейму
> a<-airquality[good_cases,]    #збережемо в новий датафрейм очищені від пропусків дані
> str(a)
'data.frame':    111 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 23 19 8 16 11 14 ...
 $ Solar.R: int  190 118 149 313 299 99 19 256 290 274 ...
…
```

Варто відзначити, що неповні дані не завжди в аналізі не підлягають розгляду. Часто їх використовують, замінивши на якесь усереднене значення пропущені елементи цих даних (на середнє по вибірці, моду, медіану).

```
Приклад
> bad_ozone<-is.na(airquality$Ozone)
>                 airquality$Ozone[bad_ozone==TRUE]<-mean(airquality$Ozone) #замінимо пропущені дані на середні за вибіркою
```

Розглянемо найбільш вживані функції для описових статистик:

```
> mean()   #середнє
> median() #медіана
> sd()     #середньоквадратичне відхилення
> range()  # розмах (виводить максимум та мінимум)
```

Також корисні:

```
> max()          #максимум
> min()          #мінімум
> var()          #дисперсія (варіація)
> quantile ()    #квантилі
> fivenum()      #мінімальне значення та квартилі
> summary()      #базові описові статистики
```

Якщо ми бажаємо отримати деякі специфічні квантилі, вони задаються як другий аргумент функції вектором чи окремим значенням.

```
> df<-mtcars        #зберегли в змінну вбудований масив
> quantile (df$mpg, c(0.3,0.6,0.9))    #30%, 60% та 90% значення квантилів
  30%   60%   90%
15.98 21.00 30.09
> quantile (df$mpg, c(0.7))    #70% квантиль
# чи так
> quantile (df$mpg, 0.7)    #70% квантиль
  70%
21.47
```

Для того, щоб подивитися основні описові статистики щодо змінної датасета використовується функція summary (). На вхід функції за бажанням (як й інших, до речі) можна подавати не одну змінну, а кілька, й весь дата фрейм теж.

```
> summary(df$mpg)      #описові статистики
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.40   15.42   19.20   20.09   22.80   33.90
```

Зверніть увагу, що якщо дані заздалегідь не чистилися щодо пропущених значень і NA, можна прямо у функціях задати варіант роботи з пропущеними значеннями. Наприклад, mean (, na.rm = FALSE) – пропустити дані, значення яких відсутні. Аналогічно з іншими функціями.

Можна вивести потрібні статистики щодо спеціально організованих підвибірок нашого масиву даних, комбінуючи логічні умови:

```
> mean(df$mpg[df$cyl==6])    #середнє значення змінної mpg за вибіркою машин з 6 циліндрами
[1] 19.74286
> mean(df$mpg[df$cyl==6 & df$am=="0"]) #середнє значення змінної mpg за вибіркою для машин з 6 циліндрами та автоматичною коробкою передач
[1] 20.56667
```

Корисна функція, яка дозволяє розрахувати основні статистики щодо однієї чи декількох змінних з урахуванням їхнього попереднього

агрегування за будь-якою іншою змінною чи змінними того самого датасету – функція aggregate.

Приклад

```
> aggregate(x = d$hp, by = list(d$carb), FUN=mean) #розрахунок середніх значень змінної hp – кількість кінських сил за даними попереднього групування за змінною carb – тип карбюратора
  Group.1     x
1       1  86.0
2       2 117.2
3       3 180.0
4       4 187.0
5       6 175.0
6       8 335.0

> mean_hp<-aggregate(x = d$hp, by = list(d$carb), FUN=mean) #результати розрахунку в датафрейм з назвою mean_hp
> colnames(mean_hp) #подивились назви колонок таблиці результату [1] "Group.1" "x"
> colnames(mean_hp)<-c("carb","mean_hp") #перейменували колонки
> View(mean_hp)
```

Лабораторна робота 4. Перевірка гіпотез та закони розподілу в R

Мета – навчитися перевіряти гіпотези та працювати із законами розподілу в R.

Завдання – сформувати три власні гіпотези, щодо осіб, що вижили та загинули на Титаніку, та перевірити в R.

Методичні рекомендації

У вбудованих датасетах пакета datasets у R є дані пасажирів, що загибли на "Титаніку", – Titatic. Інакше його можливо завантажити наступним чином (рис. 4.1).

Нас будуть цікавити тільки ті пасажири, що вижили. Виберемо відповідні рядки з бази df. Подивимося, чому дорівнює середній вік пасажирів, що вижили.

Перевіримо гіпотезу, чи дорівнював середній вік тих пасажирів, що вижили, 30 рокам. Яку альтернативну гіпотезу обрати? Виходячи із цих даних, можна зупинитися на однобічній, а точніше, лівобічній, бо вибіркове середнє менше 30. Отже, отримуємо: H0: a = 30, H1: a <30.

Перевіримо цю гіпотезу в R, використовуючи критерій Стьюдента для однієї вибірки (припускаємо, що вік пасажирів, які вижили, має нормальний розподіл) за допомогою команди t.test(, ,). На вхід команді потрібно подати змінну, для якої перевіряємо гіпотезу, середнє значення змінної, що перевіряється, та тип альтернативи (less, greater, two.sided) – рис. 4.1.

```
df <- read.csv("https://raw.githubusercontent.com/allatambov/Py-programming-3/master/28-05/Titanic.csv")
df <- na.omit(df)       #видалити рядки з порожніми значеннями
surv <- df[df$Survived == 1, ]  #створимо датасет тих, хто вижив
mean(surv$Age)    #подивимося їхній середній вік
## [1] 28.34369

t.test(surv$Age, mu = 30, alternarive = "less") #тестуємо

##  One Sample t-test
##
## data:  surv$Age
## t = -1.8866, df = 289, p-value = 0.06022
## alternative hypothesis: true mean is not equal to 30
## 95 percent confidence interval:
##  26.61570 30.07168
## sample estimates:
## mean of x
##  28.34369
```

Рис. 4.1. **Код для тестування гіпотези**

Що ми бачимо у видачі? По-перше, спостережуване значення критерію t і число ступенів свободи до нього df (df = n − 1 = 290 − 1).

По-друге, mean of x – вибіркове середнє (середнє арифметичне) і 95 %-ний довірчий інтервал до нього. Нарешті, p-value, на основі якого ми зможемо зробити висновок про нульовій гіпотезі. Зафіксуємо рівень значущості. Домовимося, що ми перевіряємо гіпотезу, коли α = 5 %.

Статистичний висновок: на 5 %-му рівні значущості на наявних даних немає підстав відкинути нульову гіпотезу на користь альтернативної (p-value> α). Змістовий висновок: середній вік пасажирів, що вижили, можна вважати рівним 30.

Зверніть увагу: якщо б обрали рівень значущості рівний 10 %, то висновок був би протилежний, так як p-value = 0,06 менше α = 0,1. Нульова гіпотеза була б відкинута на користь альтернативи, й ми б зробили висновок про те, що середній вік пасажирів, що вижили менше 30 років. Тому у висновках (і в будь-якій інтерпретації) дуже важливо вказувати рівень значущості.

Тепер перейдемо до іншого завдання. Порівняймо середній вік чоловіків і жінок, що вижили на "Титаніку" (рис. 4.2). Гіпотеза про рівність середніх (у припущенні про нормальний розподіл генеральних сукупностей) перевіряється за допомогою критерію Стьюдента для двох вибірок. У якості альтернативної гіпотези виберемо правостороннім, тобто будемо вважати, що вік тих жінок, які вижили, більший. У тесті через "~" вказується змінна, яка ділить спостереження на групи.

```
t.test(surv$Age ~ surv$Sex, alternarive = "greater") #тест
##  Welch Two Sample t-test
## data:  surv$Age by surv$Sex
## t = 0.7909, df = 158.22, p-value = 0.4302
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.353227  5.496616
## sample estimates:
## mean in group female   mean in group male
##           28.84772            27.27602
```

Рис. 4.2. **Перевірка рівності середнього віку чоловіків та жінок, що вижили**

У цьому випадку ми бачимо вже два вибіркових середніх sample estimates, середній вік чоловіків і жінок. У R реалізується не класичний критерій Стьюдента, а його модифікація. P-value в нашому випадку досить високий, вищий за будь-який конвенційний рівень значущості (1 %, 5 %, 10 %), тому на будь-якому рівні значущості в нас немає підстав відкинути нульову гіпотезу. Середній вік жінок та чоловіків, що вижили, однаковий.

Лабораторна робота 5. Кореляційно-регресійний аналіз в R

Мета – навчитися будувати регресійні лінійні відносно факторів моделі в R у тому числі з кількісними та якісними факторами, модифікованими змінними, навчитися виводити коефіцієнти моделей, їхні якісні характеристики, довірчі та прогнозні інтервали.

Завдання – ретельно вивчити скрипт, що міститься в методичних рекомендаціях, і на базі отриманих знань побудувати регресійну модель на власних даних (офіційна статистика країни, можливо дані диплому чи з останнього місця практики). Побудована модель повинна містити щонайменше один кількісний регресор, одну якісну змінну та нелінійну складову (поліном, як приклад). Регресія повинна мати сенс!

Підказка: для реалізації завдання можна брати часовий ряд зі складною динамікою та dummy-змінними.

Методичні рекомендації

Проілюструємо виконання завдання на прикладі даних вбудованого в R набору даних cars (рис. 5.1).

```
>  d<-cars      #зберігаємо в змінну d
> attach(d)      # прикріпити d / додати у шлях пошуку, щоб змінні шукати в цьому файлі
```

Рис. 5.1. **Код для прикріплення масиву**

Лінійна регресія може бути розрахована в R за допомогою функцій lm (лінійна модель) та glm (узагальнена лінійна модель).

Побудуємо регресійну модель довжини гальмівного шляху від швидкості руху авто одним із наведених варіантів:

model1 = lm (dist~speed) #підганяємо модель лінійної регресії та зберігаємо її в об'єкт model1;

model2 <-glm(dist~speed) # підганяємо узагальнену модель лінійної регресії та зберігаємо її в об'єкт model2;

model1 = lm (dist~speed, data = d) # підганяємо модель лінійної регресії та зберігаємо її в об'єкт model1 для випадку, якщо масив даних не прикріплений заздалегідь.

Вивести основні характеристики побудованої моделі, звернутися до окремих атрибутів моделі, побудувати прогноз за моделлю можливо на основі команд, що подані в таблиці.

Таблиця

Корисні команди щодо роботи з регресією

Код на R	Пояснення
summary(model1)	Коефіцієнти, помилки, рівень значущості моделі
names(model1)	дивимось, які ще є атрибути в моделі
model1$coefficients	витягуємо за іменами атрибутів моделі
coef(model1)	аналогічно для виведення коефіцієнтів моделі
confint(model1)	довірчі інтервали для оцінювання коефіцієнтів
predict() predict.lm() predict.glm()	прогнозування за моделлю
predict(model1, data.frame(speed=c(5,7)), interval="confidence")	прогнозування з довірчими інтервалами
predict(model1, data.frame(speed=c(5,7)), interval="prediction")	прогнозування з прогнозними інтервалами

Візуалізувати кореляційне поле та побудовану модель регресії можна за допомогою наступних написів (рис. 5.2).

```
> plot(speed,dist, pch =20, col="blue") #наносимо блакитні крапки
> abline(model1, col='red',lwd=3)  #будуємо лінію регресії більш товсту
```

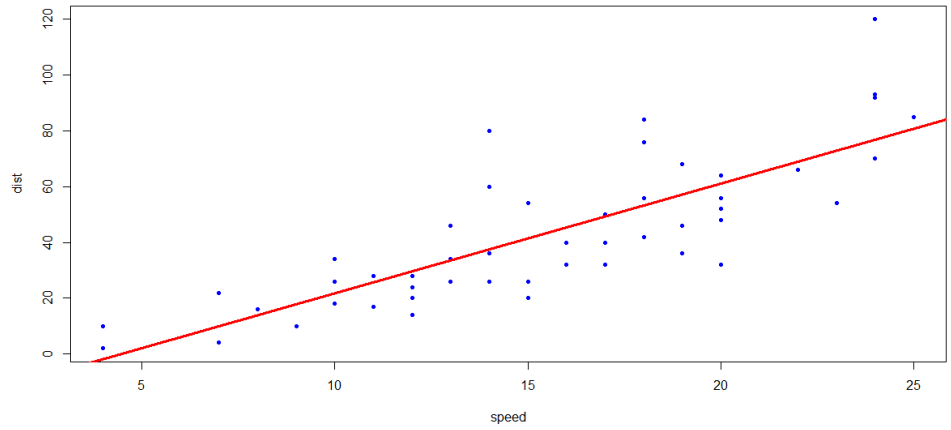


Рис. 5.2. **Візуалізація регресії**

Для того, щоб побудувати графік для прогнозу та помилок виводимо plot(predict(model1),residuals(model1)).

Аналогічно будується множинна регресія. Додамо новий стовпчик new у наш масив даних d та заповнимо його упорядкованою послідовністю даних для прикладу (рис. 5.3).

```
d$new<-c(1:50)  #додали стовпчик даних;
model3 = lm (dist~speed+new, data = d)  # побудували регресійну модель довжини гальмівного шляху від швидкості руху авто та нової змінної
```

Рис. 5.3. **Множинна регресія**

У регресійні моделі у R легко ввести ефекти взаємодії змінних. Є два варіанти побудови моделей зі взаємодією:

model4 = lm (dist~speed:new, data = d) # модель регресії від одного регресора, що відповідає спільному впливу змінних dist та new;

model5 = lm (dist~speed*new, data = d) # модель регресії від 3 регресорів: dist, new та їхній взаємодії.

За допомогою функцій lm та glm можна будувати нелінійні регресії, але такі, що лінійні відносно предикторів. Наприклад, поліноми. Вираз I() ізолює перетворення, що знаходиться в середині дужок. Побудуємо для залежної змінної *dist* поліноми 2 та 3 ступеня від *speed*:

model6 = lm (dist~speed+I(speed^2), data = d) #лінійна регресія від швидкості та її квадрату;

model7 = lm (dist~speed+I(speed^2)+I(speed^3), data = d) # регресія від швидкості, її квадрату та її кубу.

Побудувати поліном можна й за допомогою такого аргументу функції:

model8 = lm (dist~poly(speed,5), data = d) #будуємо поліном 5 ступеня.

Також є спеціальний атрибут для логлінійного перетворення як найбільш часто вживаного:

model9 = lm (dist~new+log(speed), data = d) #регресія від нової змінної та логаріфму швидкості.

В умовах, коли предикторів досить багато, зручно користуватися такими видами запису:

model10 = lm (dist~., data = d) #додати всі наявні регресори в модель;

model11 = lm (dist~.-new, data = d) #додати до регресії всі наявні в d регресори окрім new;

model12 = lm (dist~.+new:speed, data = d) #побудувати регресії від всіх регресорів, що є у d та від ефекту взаємодії змінних new та speed.

Якісні фактори в кодуються автоматично у фіктивні змінні. Можна одразу будувати регресію від якісних та кількісних даних, але потрібно вивести кодування на екран для подальшої інтерпретації результату.

Зверніть увагу, що якщо у якісного фактора n рівнів, R побудує для її перекодування n-1 фіктивну змінну.

Розглянемо підгонку регресійної моделі на даних убудованого пакета CO2, що містить дані щодо забруднення навколишнього середовища. Дивимося код (рис. 5.4).

```
>  a<-CO2   #збережемо дані пакета в робочій простір;
> model13<-lm(a$conc~a$Treatment+a$uptake+a$Plant) # будуємо регресію від кількісних значень змінної uptake, та двох якісних факторів
>  contrasts(a$Treatment)  #дивимось як закодована якісна змінна Treatment
>  contrasts(a$ Plant)  #дивимось як закодована якісна змінна Plant
>  summary(model13)  # дивимось результати регресії
```

Рис. 5.4. **Побудувати регресію за даними CO2 та вивести позначення фіктивних змінних**

Корисна функція anova() дозволяє провести перевірку гіпотези щодо рівної якості моделей. Якщо за результатами тестування значення статистики Фішера досить велике, а ймовірність низька – нульова гіпотеза

відкидається. Наприклад, зважимо, чи додає якості прогнозу введення до моделі model1 квадрата швидкості (model6):

anova(model1,model6) #порівнюємо дві регресії.

Лабораторна робота 6. Відбір та регуляризація лінійних моделей в R

Мета – вивчити особливості роботи з узагальненими лінійними моделями в R.

Завдання – побудувати логістичну регресію в R на власних даних та провести інтерпретацію результатів аналізу.

Методичні рекомендації

Почнемо з розгляду деяких кількісних і графічних зведень за даними Smarket, які входять до складу бібліотеки ISLR. Цей набір даних містить процентні зміни прибутковості біржового індексу S & P 500 для 1 250 днів – з початку 2001 до кінця 2005 року. Для кожної дати у нас є значення процентної зміни прибутковості для кожного з п'яти попередніх днів – з Lag1 по Lag5. У нас є також змінні Volume (кількість акцій (в мільярдах), проданих у попередній день), Today (процентна зміна руху ринку – вгору (Up) або вниз (Down) – у цей день) (рис. 6.1).

```
> library (ISLR)
>  names(Smarket)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
[7] "Volume"    "Today"     "Direction"
> dim(Smarket)
[1] 1250    9
> summary(Smarket)
      Year           Lag1                Lag2                Lag3                Lag4                Lag5         
 Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.92200  
 1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000   1st Qu.:-0.640000   1st Qu.:-0.64000  
 Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500   Median : 0.038500   Median : 0.03850  
 Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716   Mean   : 0.001636   Mean   : 0.00561  
 3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.59700  
 Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.73300  
     Volume          Today          Direction 
 Min.   :0.3561   Min.   :-4.922000   Down:602  
 1st Qu.:1.2574   1st Qu.:-0.639500   Up  :648  
 Median :1.4229   Median : 0.038500             
 Mean   :1.4783   Mean   : 0.003138             
 3rd Qu.:1.6417   3rd Qu.: 0.596750             
 Max.   :3.1525   Max.   : 5.733000             
```

Рис. 6.1. **Опис вихідних даних регресії**

Функція cor () створює матрицю зі значеннями всіх парних кореляцій між предикторами з деякого набору даних. Перша з наведених нижче команд видає повідомлення про помилку, оскільки змінна Direction є якісною.

```
> cor(Smarket)
Error in cor(Smarket) : 'x' must be numeric
> cor(Smarket[ ,-9])
```

	Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today
Year	1.00000000	0.029699649	0.030596422	0.033194581	0.035688718	0.029787995	0.53900647	0.030095229
Lag1	0.02969965	1.000000000	-0.026294328	-0.010803402	-0.002985911	-0.005674606	0.04090991	-0.026155045
Lag2	0.03059642	-0.026294328	1.000000000	-0.025896670	-0.010853533	-0.003557949	-0.04338321	-0.010250033
Lag3	0.03319458	-0.010803402	-0.025896670	1.000000000	-0.024051036	-0.018808338	-0.04182369	-0.002447647
Lag4	0.03568872	-0.002985911	-0.010853533	-0.024051036	1.000000000	-0.027083641	-0.04841425	-0.006899527
Lag5	0.02978799	-0.005674606	-0.003557949	-0.018808338	-0.027083641	1.000000000	-0.02200231	-0.034860083
Volume	0.53900647	0.040909908	-0.043383215	-0.041823686	-0.048414246	-0.022002315	1.00000000	0.014591823
Today	0.03009523	-0.026155045	-0.010250033	-0.002447647	-0.006899527	-0.034860083	0.01459182	1.000000000

```
> attach(Smarket)
> plot(Volume)
```

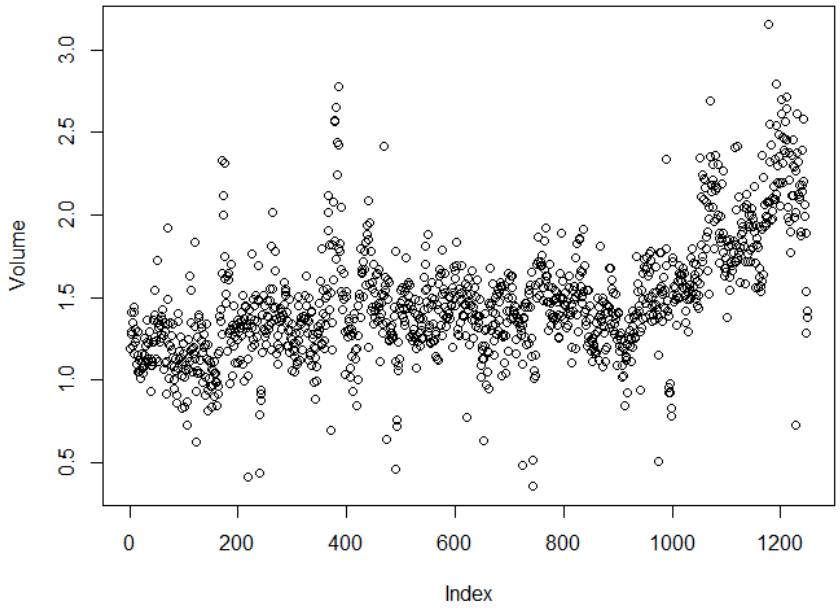


Рис. 6.2. **Кореляція між змінними**

Коефіцієнти кореляції між лаг-змінними й прибутковістю за "сьогодні" цілком очікувано близькі до нуля. Іншими словами, схоже, що прибутковість "сьогодні" дуже слабко пов'язана з прибутковістю в попередні дні. Єдина сильна кореляція між Year і Volume. Зобразивши дані графічно (рис. 6.2), ми побачимо, що Volume збільшується в часі. Іншими словами, в період з 2001 по 2005 р. середня кількість проданих щодня акцій зросла.

Далі побудуємо логістичну модель (рис. 6.3) з метою передбачення Direction на основі Lag1 – Lag5 і Volume. Функція glm () підганяє узагальнені лінійні моделі – клас моделей, які містять і логістичну регресію. Синтаксис

функції glm () схожий на синтаксис lm (), за тим винятком, що ми повинні подати аргумент family-binomial, який скаже R побудувати логістичну, а не яку-небудь іншу узагальнену лінійну модель.

```
> glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+
+ Volume, data=Smarket, family=binomial)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Smarket)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.446  -1.203   1.065   1.145   1.326

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000   0.240736  -0.523    0.601
Lag1        -0.073074   0.050167  -1.457    0.145
Lag2        -0.042301   0.050086  -0.845    0.398
Lag3         0.011085   0.049939   0.222    0.824
Lag4         0.009359   0.049974   0.187    0.851
Lag5         0.010313   0.049511   0.208    0.835
Volume       0.135441   0.158360   0.855    0.392

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1741.6

Number of Fisher Scoring iterations: 3
```

Рис. 6.3. **Код для побудови логістичної регресії та результат оцінювання її параметрів**

Найменше p-значення тут у змінної Lag1. Негативний коефіцієнт у цього предіктору передбачає, що якщо вчора спостерігалася позитивна прибутковість, то сьогодні вона навряд чи піде вгору. Однак p-значення, дорівнює 0,15, воно досить велике, й тому чіткого свідчення на користь залежності між Lag1 і Direction немає.

Для отримання доступу до коефіцієнтів цієї підігнаної моделі ми скористаємося функцією coef () (рис. 6.4). Для доступу до детального опису таких властивостей підігнаної моделі, як p-значення коефіцієнтів, ми також можемо застосувати функцію summary ().

Маючи ті або інші значення предикторів, можна скористатися функцією predict () і передбачити ймовірність того, що прибутковість індексу

піде вгору. Опція type = "response" повідомляє R, що необхідно виводити ймоправильності у вигляді P (Y = 1 | X), а не будь-яку іншу інформацію на кшталт логіт. Якщо на функцію predict () не подано якийсь певний набір даних, то ймоправильності розраховуються для навчальних даних, які були використані для підгонки логістичної моделі.

Нижче ми вивели тільки перші десять значень ймоправильності. Ми знаємо, що ці значення відповідають ймоправильності зростання, а не падіння прибутковості акцій, оскільки функція contrasts () показує, що програма створила індикаторну змінну, в якій 1 відповідає значенню Up.

```
> coef(glm.fit)
 (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5       Volume
-0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068  0.135440659
> summary(glm.fit)$coef
             Estimate Std. Error    z value  Pr(>|z|)
(Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
Lag3         0.011085108 0.04993854  0.2219750 0.8243333
Lag4         0.009358938 0.04997413  0.1872757 0.8514445
Lag5         0.010313068 0.04951146  0.2082966 0.8349974
Volume       0.135440659 0.15835970  0.8552723 0.3924004
> summary(glm.fit)$coef[ ,4]
(Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5      Volume
  0.6006983   0.1452272   0.3983491   0.8243333   0.8514445   0.8349974   0.3924004
> glm.probs=predict(glm.fit, type="response")
> glm.probs[1:10]
        1         2         3         4         5         6         7         8         9        10
0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292 0.5176135 0.4888378
> contrasts(Direction)
      Up
Down   0
Up     1
```

Рис. 6.4. **Коефіцієнти моделі**

Щоб передбачити напрямок росту прибутковості для конкретного дня, ми повинні конвертувати ці ймоправильності на помітки класів – Up або Down. Наведені нижче команди (рис. 6.5) створюють вектор із помітками класів, присвоєними залежно від того, чи перевищує передбачена можливість зростання прибутковості 0,5.

Перша команда створює вектор з 1 250 значеннями Down. Друга – перетворює на Up усі елементи вектора, у яких передбачена можливість

зростання прибутковості перевищує 0,5. Маючи ці передбачення, можна скористатися функцією table (), щоб отримати матрицю неточностей і визначити, скільки спостережень були класифіковані неправильно.

```
> glm.pred=rep("Down",1250)
> glm.pred[glm.probs>.5]="Up"

> table(glm.pred,Direction)
          Direction
glm.pred Down  Up
    Down  145 141
    Up    457 507

> (507+145)/1250
[1] 0.5216
> mean (glm.pred == Direction)
[1] 0.5216
```

Рис. 6.5. **Кодування в Up або Down та результати оцінювання**

Діагональні елементи матриці неточностей відповідають правильним прогнозам, а елементи, розташовані поза діагоналі, – помилковим прогнозам. Таким чином, наша модель для 507 днів правильно передбачила, що прибутковість зросте, й для 145 днів, що прибутковість знизиться. У цілому це 507 + 145 = 652 правильно передбачених випадків. Функцію mean () можна застосувати для обчислення частки правильно передбачених випадків. У цьому прикладі логістична регресія правильно передбачила напрям руху ринку в 52,2 % випадків.

На перший погляд може здатися, що логістична регресійна модель працює трохи краще, ніж випадкове вгадування. Однак цей результат вводить в оману, оскільки ми навчили і перевірили модель на тих же 1 250 спостереженнях. Іншими словами, 100 − 52,2 = 47,8 % – це частота помилок у навчальних даних. Як ми вже бачили раніше, частоти помилок на навчальних даних часто занадто оптимістична – вона має тенденцію недооцінювати частоту помилок на контрольній вибірці. Щоб краще оцінити правильність прогнозувань логістичної регресійної моделі в разі такого сценарію, ми можемо підігнати модель на основі певної частини даних, а потім досліджувати якість її прогнозувань на невикористаних для навчання даних. Це дасть більш реалістичну частоту помилок, у тому

сенсі, що на практиці нам буде цікаво якість прогнозів нашої моделі не для тих даних, які ми використовували для її підгонки, а для майбутніх днів, для яких напрямок руху ринку невідомий.

Для реалізації цієї стратегії ми спочатку створимо вектор, відповідний до спостережень за період з 2001 по 2004 р. Потім ми використовуємо цей вектор для створення контрольної вибірки, що містить спостереження за 2005 р.

Об'єкт train є вектором з 1 250 елементами, що відповідають спостереженнями з нашого набору даних. Елементам вектора, які відповідають спостереженням, отриманим до 2004 р., присвоєно значення TRUE, а елементам, які відповідають спостереженням за 2005 р., – значення FALSE. Об'єкт train – це логічний вектор, оскільки він містить значення TRUE і FALSE. Логічні вектори можна застосовувати для отримання певних підмножин рядків або стовпців матриці. Наприклад, команда Smarket [train,] відібрала б ту частину матриці з даними щодо прибутковості акцій, яка відповідає тільки датам до 2005 р. Оскільки це ті дані, для яких елементи вектора train дорівнюють TRUE. Символ "!" дозволяє звернути всі елементи логічного вектора. Іншими словами, вектор !Train схожий на train, за винятком того, що елементи зі значенням TRUE в train замінюються на FALSE в !Train, елементи зі значенням FALSE в train замінюються на TRUE в !Train. Отже, Smarket [!Train,] повертає частину матриці з даними щодо прибутковості акцій, які містять тільки train багатозначно FALSE, тобто спостереження за 2005 р.

Тепер, застосовуючи аргумент subset, ми підганяємо логістичну регресію на основі даних за 2005 р. Отримуємо передбачену ймовірність зростання прибутковості для кожного дня з конкретної вибірки, тобто для днів 2005 р.

Ми навчили модель на даних до 2005 р., а оцінювання якості робили не лише за даними 2005 р. Нарешті ми обчислюємо передбачення для 2005 р. і порівнюємо їх з напрямком руху ринку (рис. 6.6).

```
> train = (Year <2005)
> Smarket.2005 = Smarket [!train,]
> dim(Smarket.2005)
```

Рис. 6.6. **Результати побудови моделі за даними до 2005 р. та прогнозування для 2005 р.**

```
[1] 252   9
> Direction.2005 = Direction [!train]
>glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Smarket, family =
binomial, subset = train)
> glm.probs=predict(glm.fit,Smarket.2005,type="response")
> mean(glm.pred!=Direction.2005)
[1] 0.4648
> glm.pred=rep("Down",252)
> glm.pred[glm.probs>.5]="Up"
> table(glm.pred,Direction.2005)

        Direction.2005
glm.pred Down Up
    Down   77 97
    Up     34 44

> mean(glm.pred==Direction.2005)
```

Закінчення рис. 6.6

Вираз != означає не дорівнює. Тому остання команда обчислює частоту помилок на контрольних даних. Частота помилок на контрольних даних становить 52 %, що гірше ніж випадкове вгадування. Ми і без цього знали, що не можна передбачити майбутній розвиток ринку на підставі даних за попередній день.

Нагадаємо, що p-значення змінних були вражаючими. Й, видаливши змінні з найбільш низьким значенням, ми зможемо отримати більш ефективну модель. Нижче ми повторно підігнали логістичну регресію, використовуючи тільки лаги 1 і 2, які були найбільш перспективні щодо попередньої моделі (рис. 6.7).

```
> glm.fit=glm(Direction~Lag1+Lag2,data=Smarket, family = binomial, subset = train)
> glm.probs=predict(glm.fit,Smarket.2005,type="response")
> glm.pred=rep("Down",252)
> glm.pred[glm.probs>.5]="Up"
> table(glm.pred,Direction.2005)
        Direction.2005
glm.pred Down  Up
```

Рис. 6.7. **Підгонка моделі для лагів 1 та 2**

```
   Down   35  35
   Up     76 106
> mean(glm.pred==Direction.2005)
[1] 0.5595238
> 106/(106+76)
[1] 0.5824176
> predict(glm.fit,newdata = data.frame(Lag1=c(1.2,1.5),Lag2=c(1.1,-0.8)),type="response")
       1         2
0.4791462 0.4960939
```

Закінчення рис. 6.7

Тепер результати виглядають краще (56 % денних змін були передбачені правильно. Матриця неточностей показує, що у випадках, коли логістична регресія передбачає зростання прибутковості, частота правильних прогнозів становить 58 %.

Змістовий модуль 2
Прикладні аспекти використання мови R для аналізу даних та моделювання в економіці

Лабораторна робота 7. Кластерізація в R

Мета – навчитися будувати моделі кластерного аналізу в R та візуалізації результатів розрахунків.

Завдання – провести кластерний аналіз та визначити кластерну структуру даних. А саме:

провести стандартизацію даних;

побудувати графік каменистого осипу та зробити припущення щодо кількості кластерів, побудувати кластерне розбиття методом к-середніх та к-медоїдів з автоматичним перебором начальних центрів кластерів; оцінити якість розбиття та візуалізувати;

провести класифікацію ієрархічними процедурами та візуалізувати;

звести результати класифікації до єдиного датафрейма (варіанти класифікації стовпці).

Методичні рекомендації

Найчастіше для аналізу великих даних використовуються інтерактивні методи, такі як метод K-середніх (k-means clustering), k медоїдів або PAM (Partitioning Around Medoids), алгоритм CLARA (Clustering Large Applications) та їхні модифіції (наприклад k-medians).

Ієрархічні методи застосовувати на великих даних не завжди зручно, бо вони припускають графічний аналіз дендрограм, що не завжди є наочним за умови багатьох даних та груп.

Головною особливістю методів ітеративних є наявність гіпотези щодо кількості кластерів.

Якщо гіпотези немає, можна скористатися візуалізацією відстаней між об'єктами у вигляді графіка "кам'янистого осипу" (критерію "кам'янистого осипу" и Кеттела, графіка "ліктя"). Побудуємо його для наших даних об'єкта CAR (рис. 7.1).

```
> kl <- (nrow(CAR)-1)*sum(apply(CAR,2,var))
> for (i in 2:15) kl[i] <- sum(kmeans(CAR, centers=i)$withinss)
> plot(1:15, kl, type="b", xlab="Число кластерів", ylab="Сума квадратів відстаней усередині кластерів")
```

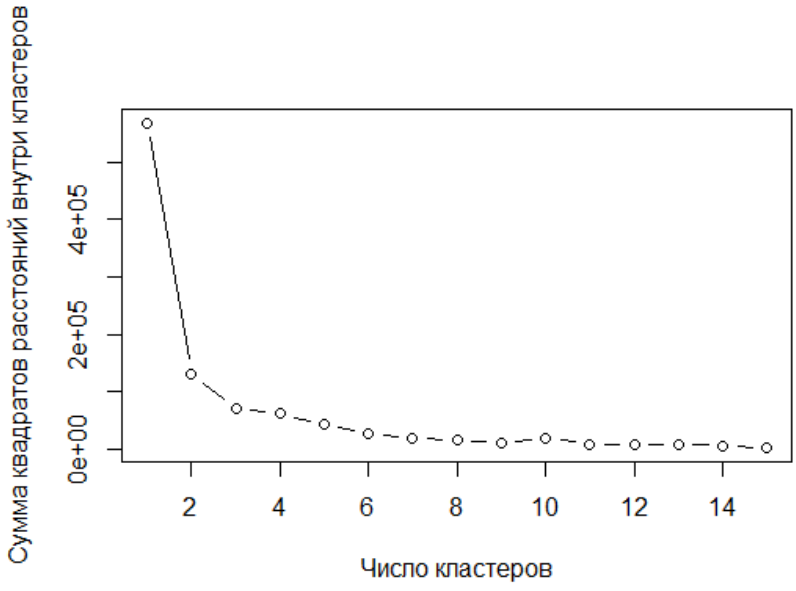


Рис. 7.1. **Графік "кам'янистого осипу"**

Сама функція класифікації має три аргументи: масив даних – X, число кластерів – k, максимум ітерацій – m) – kmean (X, k, m).

Об'єкт K2 є список з дев'яти елементів. Далі зручно розрахувати середні за кластерами (рис. 7.2).

Додати номери кластерів у вихідний дата фрейм можна за формулою: CAR<- data.frame(CAR, K2$cluster).

Можливо зробити різні візуалізації кластерів, наприклад такий, де центри кластерів помічені зірочками, кластери – кольорами в просторі ознак наступним чином (рис. 7.2).

```
>K2<-kmeans(CAR,2)
>aggregate(CAR,by=list(K2$cluster),FUN=mean)  #середні за кластерами
  Group.1      mpg      cyl     disp       hp
1       1 15.10769 8.000000 357.1077 199.53846
2       2 24.12353 4.823529 136.3941  97.41176

>CAR<- data.frame(CAR, K2$cluster)
>op <- par(mfrow = c(1, 2))
>plot(CAR[c("mpg", "hp")], col=K2$cluster)
>points(K2$centers[,c("mpg", "hp")], col=1:3, pch=8, cex=2)
>plot(CAR[c("cyl", "disp")], col=K2$cluster)
>points(K2$centers[,c("cyl", "disp")], col=1:3, pch=8, cex=2)
```

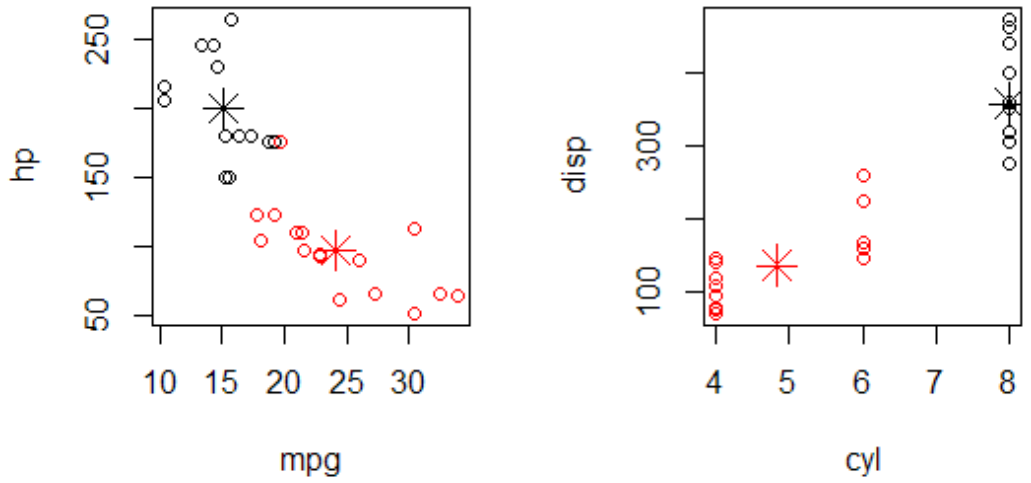


Рис. 7.2. **Код та результат візуалізації кластерного аналізу**

Більш поширений варіант візуалізації в просторі головних компонентів подано на рис. 7.3.

```
library(cluster) # підключили бібліотеку
clusplot(d, K2$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```

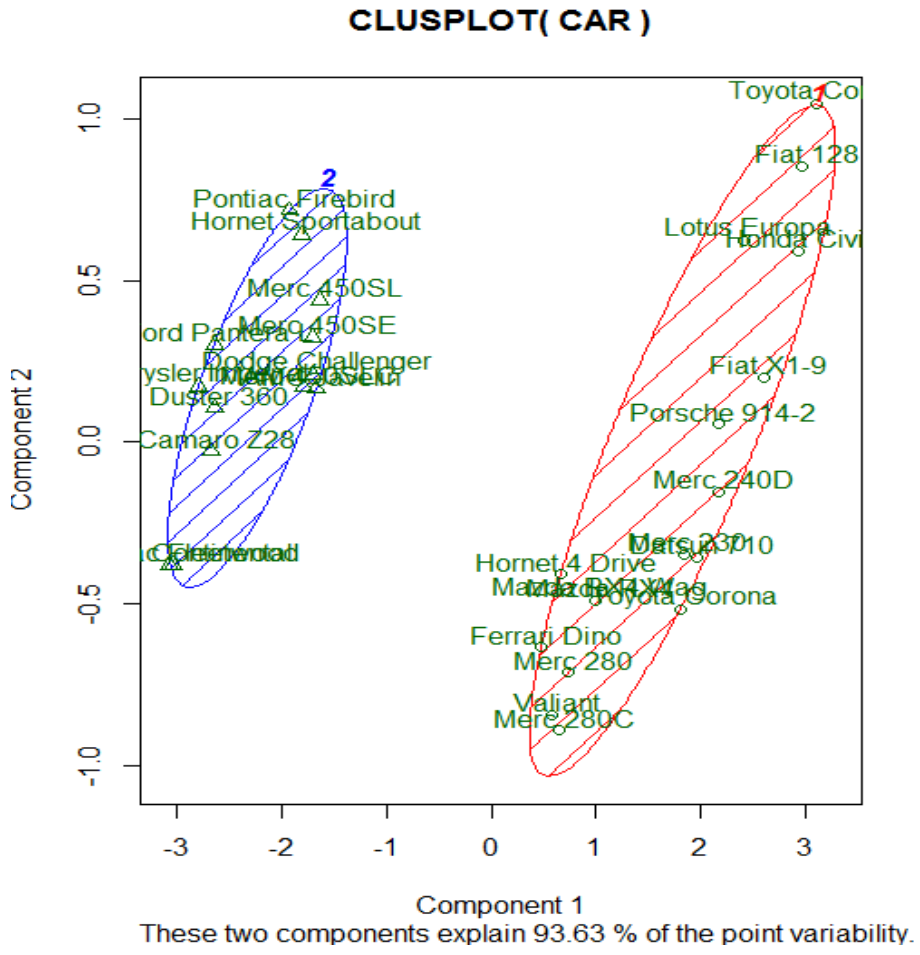


Рис. 7.3. **Код та результат візуалізації кластерного аналізу**

Більше про візуалізацію кластерного аналізу дивитися за посиланням: http://mymagictools.blogspot.com/2015/07/r.html.

Розглянемо ієрархічні процедури.

Для проведення стандартизації можна використовувати функцію preProcess пакета caret. Функція здійснює перетворення даних, але не змінює масив. Стандартизуємо дані CAR та записуємо в окремий масив (рис. 7.4).

Ієрархічна кластеризація проводиться функцією hclust, на вхід якої потрібно подати значення матриці відстаней і метод приєднання. Матриця відстаней розраховується функцією dist (). Якщо не вказати інше, то за замовчуванням буде евклідова: dist(StandCAR) чи dist(StandCAR, method = "euclidean").

```
> library("caret", lib.loc="C:/Program Files/R/R-3.2.5/library")
> ( transCAR = preProcess(CAR, method = c("center", "scale")) )
Created from 30 samples and 4 variables
Pre-processing:
  - centered (4)
  - ignored (0)
  - scaled (4)
> StandCAR = predict(transCAR, CAR)
# чи можливо так
scale(x, center = TRUE, scale = TRUE)
> SCAR=scale(CAR) #за замовчуванням буде шкалювання та центрування
```

Рис. 7.4. **Код для стандартизації**

Варіанти застосування кластерізації за різними методами та їхній візуалізації у вигляді дендрограм подано на рис. 7.5.

```
> k.complete<-hclust(dist(SCAR), method = "complete")  #повний зв'язок
> k.single<-hclust(dist(SCAR), method = "single")    # одиночного приєднання
> k.average<-hclust(dist(SCAR), method = "average")   #середнього приєднання
> k.Ward<-hclust(dist(StandCAR), method = "ward.D2")
        #візуалізація на одному графіку
> par(mfrow=c(1,3))  #задаємо сітку для виведення графіка
> plot(k.complete)
> plot(k.single)
> plot(k.average)
> plot(k.Ward)
```

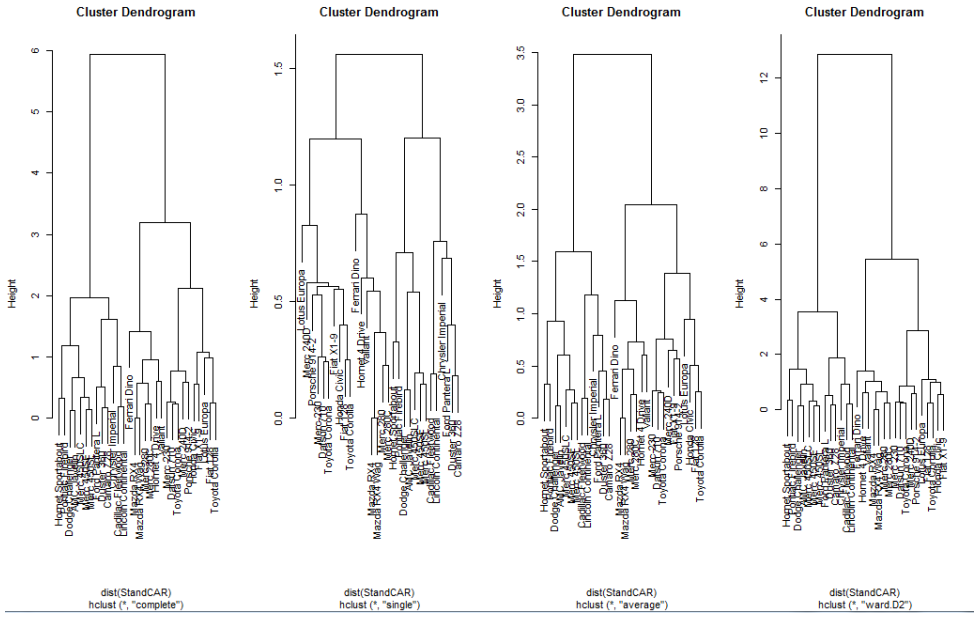


Рис. 7.5. **Код для кластерізації та візуалізації дендрограм**

Коли за дендрограмами виявили природне розбиття, можна вивести розбиття за допомогою функції cutree (2 – кількість кластерів, розбивку на які ми захотіли мати під час процедури повної відстані):

> CAR$cluss<-cutree(k.complete,2).

Хороша візуалізація робиться наступним чином (рис. 7.6).

```
> par(mfrow=c(1,2))
> k.Ward<-hclust(dist(SCAR), method = "ward.D2")
> grp <- cutree(k.Ward, k =2)  # Розріз дерева на 2 групи
> plot(k.Ward, cex = 0.7)
> rect.hclust(k.Ward, k = 2, border = 2:3)
>
> k.Ward<-hclust(dist(SCAR), method = "ward.D2")
> grp <- cutree(k.Ward, k =5)  # Розріз дерева на 5 груп
> plot(k.Ward, cex = 0.7)
> rect.hclust(k.Ward, k = 5, border = 2:6)
```

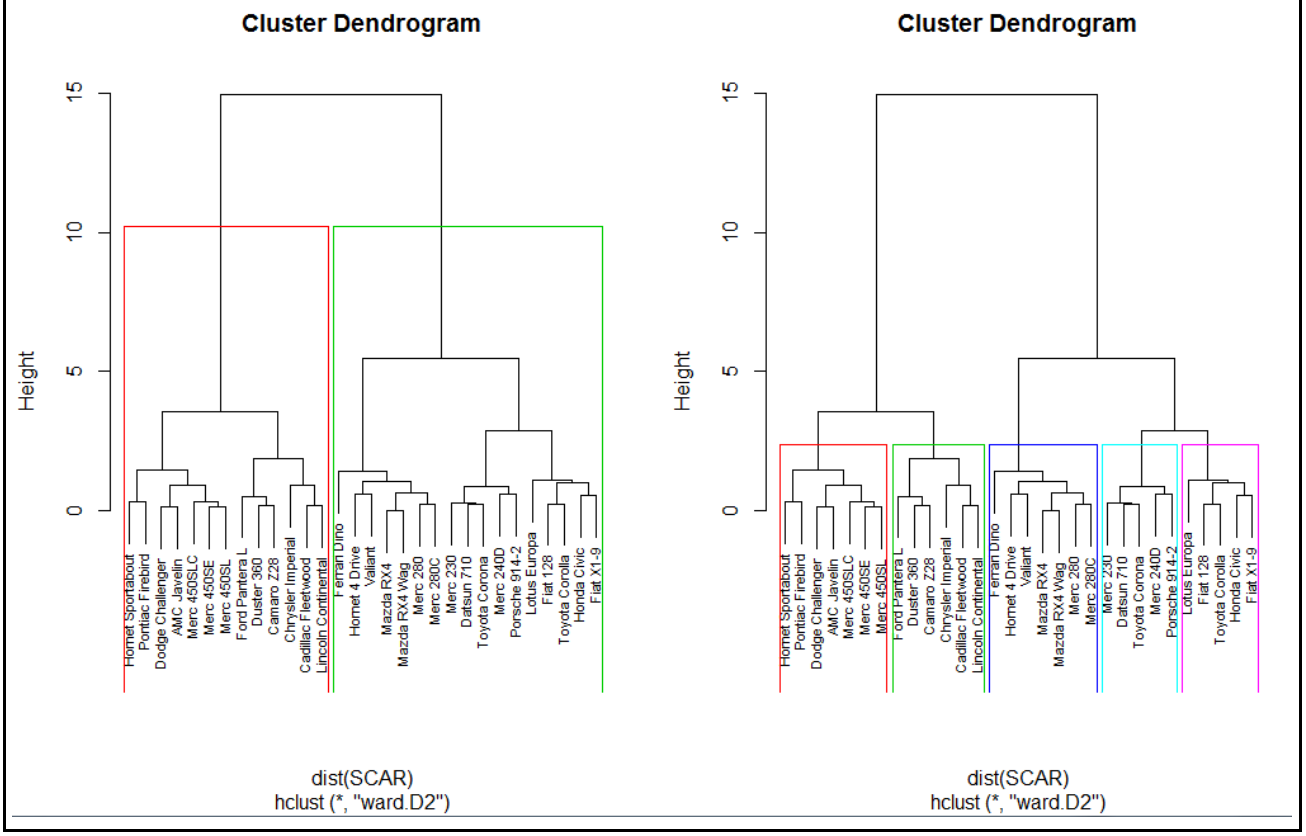


Рис. 7.6. **Візуалізація розбиття на кластери**

Більш просунуті варіанти дивиться за посиланням: https://ranalytics.github.io/data-mining/102-H-Clustering.html.

Лабораторна робота 8. Дерева класифікації, ансамблі дерев у R

Мета – навчитися будувати моделі, засновані на методах дерев класифікації (Classification Trees) у R та моделі випадкових лісів дерев (Random Forests).

Завдання. За власними даними побудувати дерево класифікації, провести обрізання гілок та оцінити рівень поліпшення моделі. Дані обов'язково бити на навчальну та тестову вибірки. Побудувати випадковий ліс дерев.

Методичні рекомендації

Дерева рішень будуються на основі функції tree з однойменного пакета. Побудуємо дерево рішень на вбудованому пакеті даних за характеристиками квітки ірису iris.

Для цього виводимо код (рис. 8.1).

```
> library(tree)  #підключимо пакет tree

> D<-iris          #збережемо дані в змінну D

> m1=tree(Species~.,D)  #будуємо дерево для змінної Species (види квіток ірису) від усіх наявних змінних

> summary(m1)  #виводимо опис, перелік змінних, які використані у класифікації, та якість класифікації

Classification tree:
tree(formula = Species ~ ., data = D)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width"  "Sepal.Length"
Number of terminal nodes:  6
Residual mean deviance:  0.1253 = 18.05 / 144
Misclassification error rate: 0.02667 = 4 / 150
```

Рис. 8.1. **Результати побудови дерева**

Частота помилок на навчальній вибірці (Misclassification error rate) 2.6 %. Використано три змінні із чотирьох для побудови дерева. Мале

значення показника залишкової середньої аномальності (Residual mean deviance) свідчить, що дерево гарно описує навчальну вибірку.

Візуалізація дерева здійснюється функцією plot, додавання вузлів – функцією text. Аргумент pretty вказує, що для будь-яких якісних предикторів необхідно навести назву відповідних категорій, а не тільки позначити літерами. Код та результат візуалізації на рис. 8.2.

```
> plot(m1)
> text(m1,pretty=0)
```

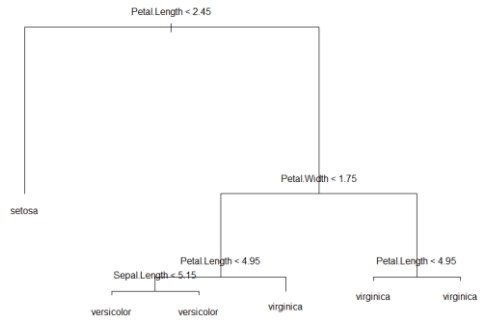


Рис. 8.2. **Дерево класифікацій**

Просте звернення до збудованого класифікатором дає нам опис усіх гілок дерева (критерій розбиття, кількість спостережень, що припадають на цю гілку, показник аномальності, передбачений клас для гілки, а також частку, що припадає на цю гілку спостережень зі значеннями так і ні (рис. 8.3).

```
> m1
node), split, n, deviance, yval, (yprob)
      * denotes terminal node
 1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
   2) Petal.Length < 2.45 50   0.000 setosa ( 1.00000 0.00000 0.00000 ) *
   3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
     6) Petal.Width < 1.75 54  33.320 versicolor ( 0.00000 0.90741 0.09259 )
      12) Petal.Length < 4.95 48   9.721 versicolor ( 0.00000 0.97917 0.02083 )
        24) Sepal.Length < 5.15 5   5.004 versicolor ( 0.00000 0.80000 0.20000 ) *
        25) Sepal.Length > 5.15 43   0.000 versicolor ( 0.00000 1.00000 0.00000 ) *
      13) Petal.Length > 4.95 6   7.638 virginica ( 0.00000 0.33333 0.66667 ) *  …
  …
```

Рис. 8.3. **Опис гілок дерева рішень**

Використовуємо функцію predict для виводу матриці класифікації. Процент випадків, що правильно класифіковано дорівнює 97 % (рис. 8.4).

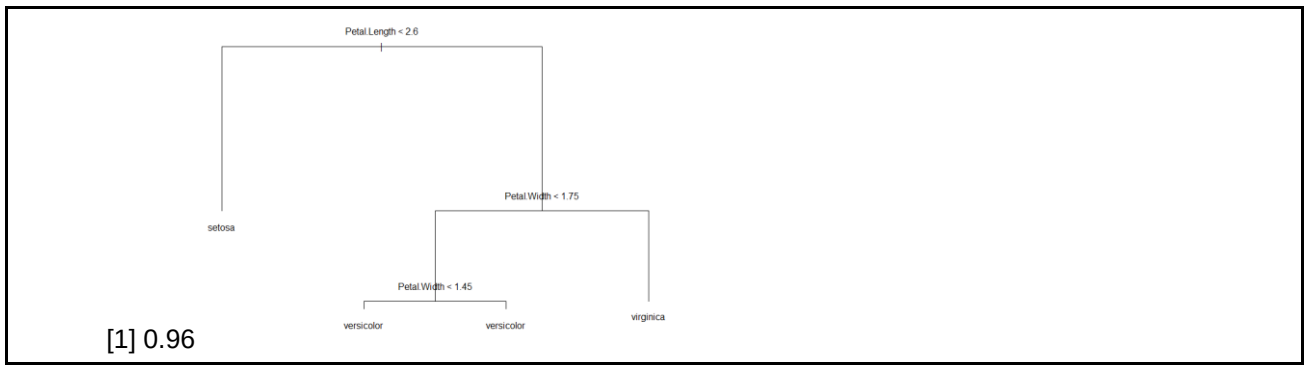
```
> m1.pred=predict(m1,D,type="class")
> table(m1.pred,D$Species)
  m1.pred      setosa versicolor virginica
  setosa         50         0         0
  versicolor      0        47         1
  virginica       0         3        49
> (50+47+49)/150
[1] 0.9733333
```

Рис. 8.4. **Прогнозування за деревом на навчальній вибірці**

Дерева чудово описують дані навчальної вибірки, але як переконатися, що вони опишуть і нові дані? Розподілимо дані на навчальну й тестову вибірки, будуємо дерево за навчальною. Оцінюємо якість дерева за контрольною, це знизить ризик перенавчання (рис. 8.5).

```
> set.seed(2)
> train=sample(1:nrow(D),75)
> D.test=D[-train,]
> m2=tree(Species~.,D,subset = train)
> summary(m2)
Classification tree:
tree(formula = Species ~ ., data = D, subset = train)
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width"
Number of terminal nodes:  4
Residual mean deviance:  0.1978 = 14.05 / 71
Misclassification error rate: 0.04 = 3 / 75
> plot(m2)
 > m2.pred=predict(m2,D.test,type = "class")
> table(m2.pred,D.test$Species)
  m2.pred      setosa versicolor virginica
  setosa         26         0         0
  versicolor      0        26         2
  virginica       0         1        20
> (26+26+20)/75
```

Рис. 8.5. **Побудова дерева**



Закінчення рис. 8.5

Таким чином, правильно класифіковані 96 % тестової вибірки, на 1 % менше, ніж за всіма даними.

Скрипт побудови ліса дерев та моделі XGBoost наведено в посиланні: https://drive.google.com/file/d/0B0JXHrPlIPHTRi0yVFdHdllpem8/view.

Лабораторна робота 9. Машини опорних векторів у R

Мета – навчитися проводити класифікацію методами машин опорних векторів у R.

Завдання. За власними даними побудувати класифікатор на два класи за методом машин опорних векторів, оцінити якість класифікації.

Методичні рекомендації

Розглянемо набір даних щодо зразків тканин чотирьох різних виглядів раку Khan. Кожне спостереження містить дані щодо вимірів експресії генів – 2 308 змінних (X) та результуючу змінну (У) – тип саркоми. Цей набір уже містить навчальні вибірки даних на 63 спостереження (xtrain та ytain) та контрольні на 20 спостережень (xtest та ytest).

```
> Library(ISLR)
>names(Khan)
>dim(Khan$xtrain)
>dim(Khan$xtest)
>dim(Khan$ytrain)
>dim(Khan$ytest)
```

Використаємо класифікатор на опорних векторах для прогнозування типу раку за рівнем генів.

Моделі класифікації машин опорних векторів будуються командою svm пакета e1071 у R. Функція svm має ключовий аргумент – тип ядра – kernel. Залежно від нього будуть різні інші аргументи. Ще один загальний аргумент – штраф за порушення межі класів cost.

Оскільки в прикладі кількість змінних значно перевищує кількість спостережень, тож задовільнимося лінійним ядром.

```
> d = data.frame(x=Khan#xtrain,y=as.factor(Khan$ytrain))  #створюємо єдиний датасет для даних, де обов'язково "y" буде фактором
> model1=svm(y ~.,data=d,kernel="linear",cost=10)  #будуємо класифікатор на опорних векторах із лінійним ядром та штрафом 10 за порушення границь зазору між видами раку
> summary (out)  #виводимо результат класифікації для навчальної вибірки
```

Бачимо повну відсутність неправильно класифікованих випадків на навчальній вибірці. Але, яка реальна якість класифікатора, ми дізнаємося тільки оцінивши його за тестовою вибіркою.

```
> d.test=data.frame(x=Khan$xtest,y=as.factor(khan$ytest))
> prediction=predict(model1,newdata=d.test)
>table(prediction,d.test$y)
```

Як бачимо, використання штрафу в cost =10 призводить до виникнення двох помилок на контрольних даних.

Можна побавитися змінюючи cost, тип ядра на "radial" (при цьому задаємо ще параметр gamma, для визначення параметру γ ядра на основі радіальної базисної функції) чи "polinomial" (потрібно буде задати аргумент degree для ступеня поліному ядра). Це буде мати наступний вигляд:

> model2=svm(y ~.,data=d,kernel="polynomial",degree=3,cost=10) #будуємо класифікатор на опорних векторах із поліноміальним ядром (ступеня 3) та штрафом 10 за порушення меж проміжку між видами раку

> model3=svm(y ~.,data=d,kernel=" radial ",gamma=1,cost=10) #будуємо класифікатор на опорних векторах з поліноміальним ядром (ступеня 3) та штрафом 10 за порушення меж проміжку між видами раку

Лабораторна робота 10. Звуження простору ознак у R

Мета – навчитися проводити звуження простору ознак методами ICA та PCA у R та візуалізувати результатами.

Завдання. За власними даними провести звуження простору ознак методами ICA та PCA у R.

Методичні рекомендації

Метод головних компонент дозволяє замінити кілька вихідних змінних на меншу кількість нових. Нові штучні змінні називаються головними компонентами.

Метод належить до методів навчання без учителя.

Спробуємо провести аналіз на даних з багатоборства (рис. 10.1).

```
>library("knitr")
>library("ggplot2")      # для побудови графіків
>library("HSAUR")        # потрібен набор даних з багатоборства
>data(heptathlon)        #завантажуємо набір даних з багатоборства
>head(heptathlon)        #дивимося структуру даних
##          hurdles highjump  shot run200m longjump javelin run800m score
## hurdles   1.0000  -0.8114 -0.65    0.77   -0.912 -0.0078    0.78 -0.92
## highjump -0.8114   1.0000  0.44   -0.49    0.782  0.0022   -0.59  0.77
## shot     -0.6513   0.4408  1.00   -0.68    0.743  0.2690   -0.42  0.80
## run200m   0.7737  -0.4877 -0.68    1.00   -0.817 -0.3330    0.62 -0.86
## longjump -0.9121   0.7824  0.74   -0.82    1.000  0.0671   -0.70  0.95
## javelin  -0.0078   0.0022  0.27   -0.33    0.067  1.0000    0.02  0.25
## run800m   0.7793  -0.5912 -0.42    0.62   -0.700  0.0200    1.00 -0.77
## score    -0.9232   0.7674  0.80   -0.86    0.950  0.2531   -0.77  1.00
```

Рис. 10.1. **Попередні дії з підключення пакетів та даних**

Восьмий стовпець, score, нам не потрібен, тому що це не результат змагання, а підсумковий бал з багатоборства. Застосовуємо метод головних компонент на стандартизованих змінних (рис. 10.2).

```
>h <- heptathlon[, -8]
>hept_pca <- prcomp(h, scale = TRUE) #стандартизуємо дані
>pc <- hept_pca$x   #збережемо головні компоненти в дата фрейм
>head(pc)     #подивимося на перші дані таблиці для уявлення про неї

      PC1        PC2        PC3        PC4        PC5        PC6        PC7
```

Рис. 10.2. **Попередні дії з підключення пакетів та даних**

```
Joyner-Kersee (USA) -4.121448 -1.24240435 -0.3699131 -0.02300174  0.4260062 -
0.33932922  0.3479213
John (GDR)          -2.882186 -0.52372600 -0.8974147  0.47545176 -0.7030659
0.23808730  0.1440158
…
> loadings <- hept_pca$rotation   #збережемо вагові коефіцієнти в датафрейм
>head(loadings)                          #подивимося шапку
PC1         PC2         PC3         PC4         PC5         PC6         PC7
hurdles   0.4528710 -0.15792058 -0.04514996  0.02653873 -0.09494792 -0.78334101
-0.3802471
highjump -0.3771992  0.24807386  0.36777902 -0.67999172 -0.01879888 -0.09939981
-0.4339311
shot     -0.3630725 -0.28940743 -0.67618919 -0.12431725 -0.51165201  0.05085983
-0.2176249
…
```

Закінчення рис. 10.2

Оцінимо якість отриманих компонент. Для цього подивимося, наприклад, як перша компонента пов'язана з результатом спортсмена – (рис. 10.3). Таким чином бачимо, що перша компонента вгадує майже повністю результат спортсмена. Подивимося також, яку долю дисперсії пояснює кожна з головних компонент та скільки їх потрібно для пояснення даних.

```
>qplot(x = heptathlon$score, y = pc[, 1]) +
  labs(x = "Сума балів з боротьби",
       y = "Перша головна компонента",
       title = "Зв'язок першої компоненти з результатом")
```

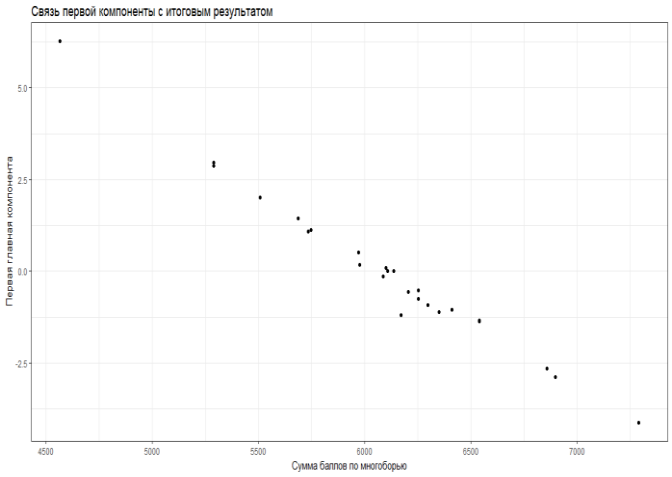


Рис. 10.3. **Важливість та пояснювальна спроможність компонент**

```
> summary(hept_pca)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6    PC7
Standard deviation     2.1119 1.0928 0.72181 0.67614 0.49524 0.27010 0.2214
Proportion of Variance 0.6372 0.1706 0.07443 0.06531 0.03504 0.01042 0.0070
Cumulative Proportion  0.6372 0.8078 0.88223 0.94754 0.98258 0.99300 1.0000
```

Закінчення рис. 10.3

Візуалізуємо важливість та пояснювальна спроможність компонент на графіку (рис. 10.4).

```
>exp_percent <- summary(hept_pca)$importance[2, ]
qplot(y = exp_percent, x = names(exp_percent)) +
  geom_bar(stat = 'identity') +
  labs(x = "Головні компоненти",
       y = "Долі дисперсії",
       title = "Долі дисперсіїи, що відповідають головним компонентам \n
(без стандартизації)")
```

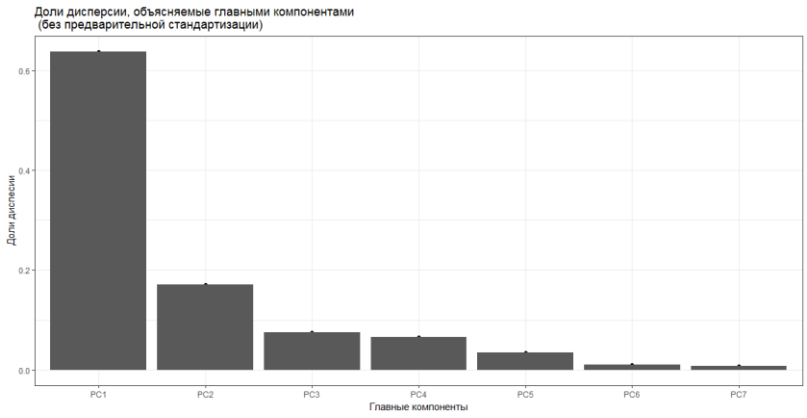


Рис. 10.4. **Діаграма важливості компонент**

Побудуємо графік перших двох компонент із внеском кожної змінної. Червоні вектори – це проекції одиничних векторів вихідних координат на площину перших двох головних компонент (рис. 10.5).

```
>biplot(hept_pca,
     xlim = c(-0.8, 0.8), ylim = c(-0.6, 0.6))
```

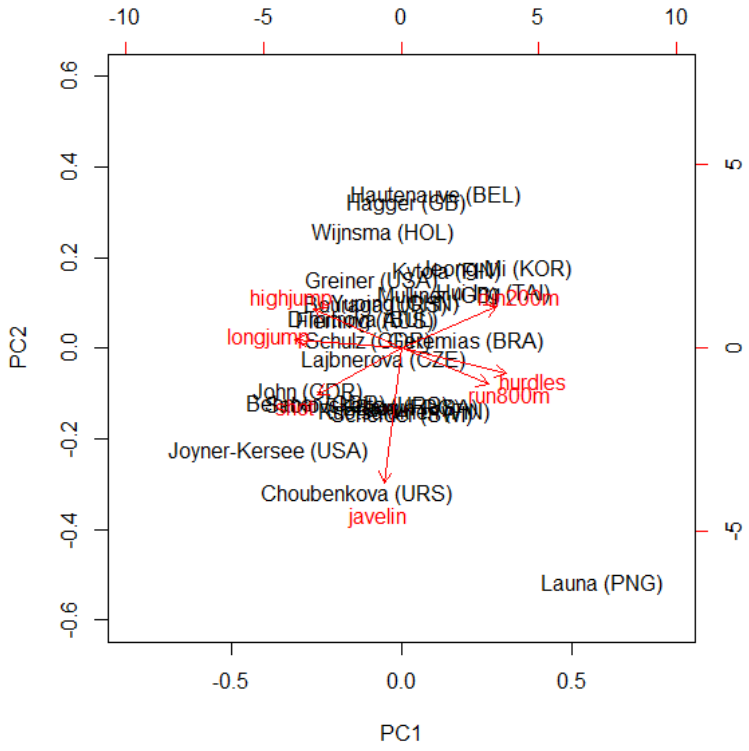


Рис. 10.5. **Спостереження в площині двох перших компонент**

Рекомендована література

1. Бабешко Л. О. Основы эконометрического моделирования / Л. О. Бабешко. – Москва : КомКнига, 2006. – 432 с.

2. Введение в статистическое обучение с примерами на языке R / Г. Джеймс, Д. Уиттон, Т. Хасти, Р. Тибширани ; пер. с англ. С. Э. Мастицкого. – Москва : ДМК Пресс, 2016. – 450 с.

3. Дубров А. М. Многомерные статистические методы / А. М. Дубров, В. С. Мхитарян, Л. И. Трошин. – Москва : Финансы и статистика, 1998. – 350 с.

4. Дюк В. Data Mining : учебный курс / В. Дюк, А. Самойленко. – Санкт-Петербург : б. и., 2001. – 368 с.

5. Зарядов И. С. Введение в статистический пакет R : типы переменных, структуры данных, чтение и запись информации, графика / И. С. Зарядов. – Москва : Изд-во РУДН, 2010. – 207 с.

6. Зарядов И. С. Статистический пакет R: теория вероятностей и математическая статистика / И. С. Зарядов. – Москва : Изд-во РУДН, 2010. – 141 с.

7. Кабаков Р. И. R в действии. Анализ и визуализация данных в программе R / Р. И. Кабаков ; пер. с англ. П. А. Волковой. – Москва : ДМК Пресс, 2014. – 588 с.

8. Магнус Я. Р. Эконометрика / Я. Р. Магнус, П. К. Катышев, А. А. Пересецкий. – Москва : Дело, 1997. – 248 с.

9. Методы и модели анализа данных : OLAP и Data Mining / А. А. Барсегян, М. С. Куприянов, В. В. Степаненко, И. И. Холод. – Санкт-Петербург : БХВ-Петербург, 2004. – 336 с.

10. Многомерный статистический анализ в экономике / Л. А. Сошникова, В. Н. Тамашевич, Г. Уебе, М. Шефер ; [под ред.проф. В. Н. Тамашевича. – Москва : ЮНИТИ-ДАНА, 1999. – 598 с.

11. Плюта В. Сравнительный многомерный анализ в экономических исследованиях / В. Плюта. – Москва : Статистика, 1980. – 151 с.

12. Статистический анализ данных в системе R : учеб. пособ. / А. Г. Буховец, П. В. Москалев, В. П. Богатова, Т. Я. Бирючинская ; под ред. проф. А. Г. Буховца. – Воронеж : ВГАУ, 2010. – 124 с.

13. Шипунов А. Б. и др. Наглядная статистика. Используем R!. – Москва : ДМК Пресс, 2014. – 296 с.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

МАТЕМАТИЧНІ МЕТОДИ І МОДЕЛІ РИНКОВОЇ ЕКОНОМІКИ

Методичні рекомендації до лабораторних робіт

для студентів спеціальності 051 "Економіка" другого (магістерського) рівня

Самостійне електронне текстове мережеве видання

Укладач **Івахненко** Ольга Володимирівна

Відповідальний за видання *Л. С. Гур'янова*

Редактор *О. І. Черненко*

Коректор *О. В. Анацька*

План 2019 р. Поз. № 89 ЕВ. Обсяг 65 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру ***ДК № 4853 від 20.02.2015 р.***