ЛЕКЦІЯ.
СТАТИСТИКА В R

3.1. Типи об’єктів в R

R це об’єктно-орієнтована мова, в якій основними об’єктами виступають так звані змінні. В статистичному аналізі зазвичай використовуються такі об’єкти:

– вектор;
– список;
– матриця;
– масив (array);
– фактор;
– набір даних (data frame).

Найпростішим та базовим із цих об'єктів є вектор, з якого складається шість типів даних або класів векторів: логічні [logical] (TRUE , FALSE), числові [numeric] (12.3, 5, 999), цілі числа [integer] (2, 34, 0), комплексні [complex] (3 + 2i), символьні [character] ("a", "добре", "TRUE", "23.4"), первинні дані [raw] ("Hello" зберігається як 48 65 6c 6c 6f). Всі інші R-об'єкти будуються на базі цих векторів.

В соціологічних дослідженнях найчастіше використовуються такі типи R-об’єктів як вектор, список, масив, фактор та набір даних.

Вектор

Якщо ви хочете створити вектор з кількома елементами, ви повинні використовувати функцію **c ()**, яка означає об'єднання елементів у вектор.

```
# Створення вектору під назвою «apple»
apple <- c('red','green',"yellow")
print(apple)
# Визначення класу вектору «apple»
print(class(apple))
```

Коли ми виконуємо вищезазначений код, він видає такий результат:

```
[1] "red"    "green"  "yellow"
[1] "character"
```

Список

Список - це R-об'єкт, який може складатися з багатьох різних типів елементів: векторів, функцій та навіть інших списків.

```
# Створення списку під назвою «list1»
list1 <- list(c(2,5,3),21.3,sin)
# Виведення результату на монітор
print(list1)
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
[[1]]
[1] 2 5 3

[[2]]
[1] 21.3

[[3]]
function (x)  .Primitive("sin")
```

Фактор

Фактор – це R-об'єкт, створений за допомогою вектору. Він зберігає вектор разом з різними значеннями елементів вектору як мітки. Мітки завжди є символом, незалежно від того, чи це числовий, символьний або інший тип вектору. Фактор використовується для статистичного моделювання. Фактори створюються за допомогою функції **factor ()**.

```
# Створення вектору «apple_colors»
apple_colors <- c('green','green','yellow','red','red','red','green')
# Створення фактору «factor_apple»
factor_apple <- factor(apple_colors)
# Вивід результату на монітор
print(factor_apple)
print(nlevels(factor_apple))
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
[1] green  green  yellow red    red    red    green
Levels: green red yellow
[1] 3
```

Набір даних

Набір даних - це такий об’єкт даних, що представлено у вигляді таблиці (як у таблицях MS Office Excel). На відміну від матриці, в наборі даних кожен стовпець може містити різні типи даних. Перший стовпець може бути числовим, тоді як другий стовпець може бути символьним, а третій – логічним. При цьому кожний вектор цього набору має однакову довжину. Набір даних створюються за допомогою функції **data.frame ()**.

```
# Створення набору даних «BMI»
BMI <- data.frame(
     gender = c("Male", "Male","Female"),
     height = c(152, 171.5, 165),
     weight = c(81,93, 78),
     Age =c(42,38,26)
 )
print(BMI)
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
  gender height weight Age
1   Male  152.0     81  42
2   Male  171.5     93  38
3 Female  165.0     78  26
```

3.2. Візуалізація даних в R

Стовпчикова діаграма

Стовпчикова діаграма представляє дані в прямокутних смужках з довжиною стрижня, пропорційного значенню відповідної змінної. R використовує функцію barplot () для створення таких діаграм. R може малювати як вертикальні, так і горизонтальні смуги на стовпчиковій діаграмі. У стовпчиковій діаграмі кожен з стовпчиків може бути заданий різними кольорами.

Синтаксис

Основний синтаксис створення стовпчикової діаграми в R:

```
barplot (H, xlab, ylab, main, names.arg, col)
```

Нижче наведено опис використовуваних параметрів:

– **H** - це вектор або матриця, що містить числові значення, що використовуються в гістограмі.

– **xlab** є міткою для осі x;

– **ylab** - це мітка для осі y;

– **main** - це заголовок діаграми;

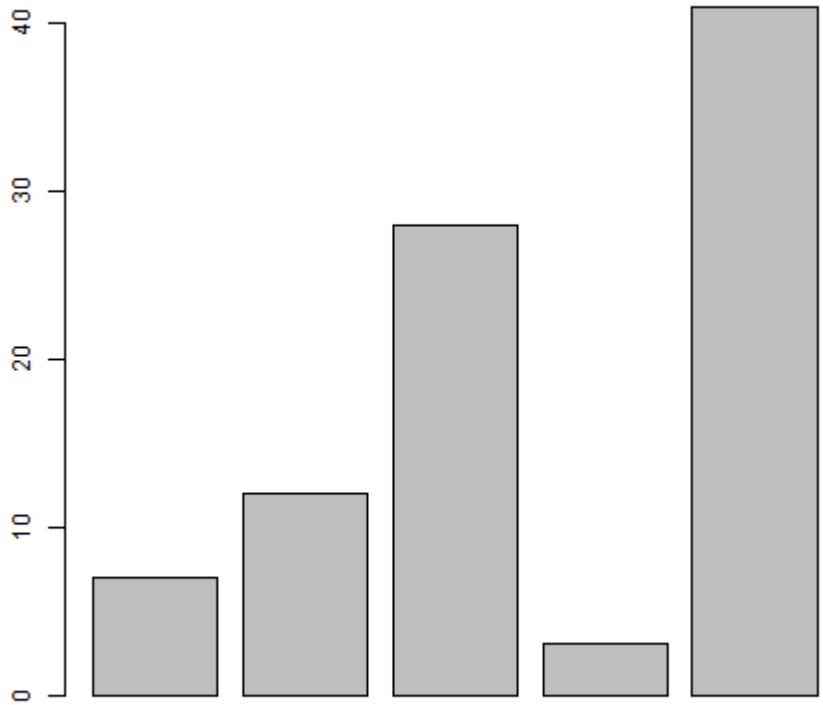
– **names.arg** - це вектор імен, що з'являються під кожним рядком.

– **col** використовується для відтворення кольорів на графах.

Приклад

Проста стовпчикова діаграма створюється лише за допомогою вхідного вектора та назви кожного рядка. Наступний скрипт створить і збереже стовпчикову діаграму в поточну робочу теку R.

```
# Створення даних для діаграми у вигляді вектору «H»
H <- c(7,12,28,3,41)
# Визначення назви файлу діаграми
png(file = "barchart.png")
# Побудова діаграми
barplot(H)
# Збереження у файл
dev.off()
```



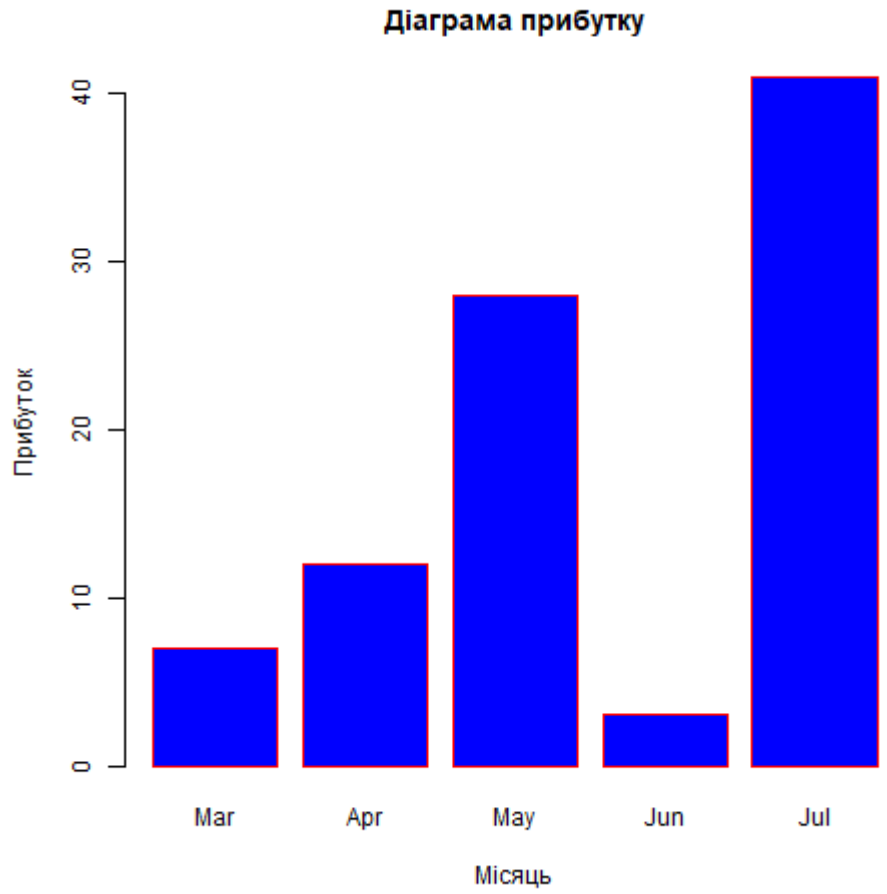
Мітки з написом, назви та кольорів

Функції стовпчикової діаграми можна розширити, додавши більше параметрів. Основний параметр використовується для додавання заголовку. Параметр col використовується для додавання кольорів до смуг. Args.name – вектор, що має однакову кількість значень як у вхідного вектору для опису значення кожного рядка.

Приклад

Наступний скрипт створить і збереже стовпчикову діаграму в поточну робочу теку R.

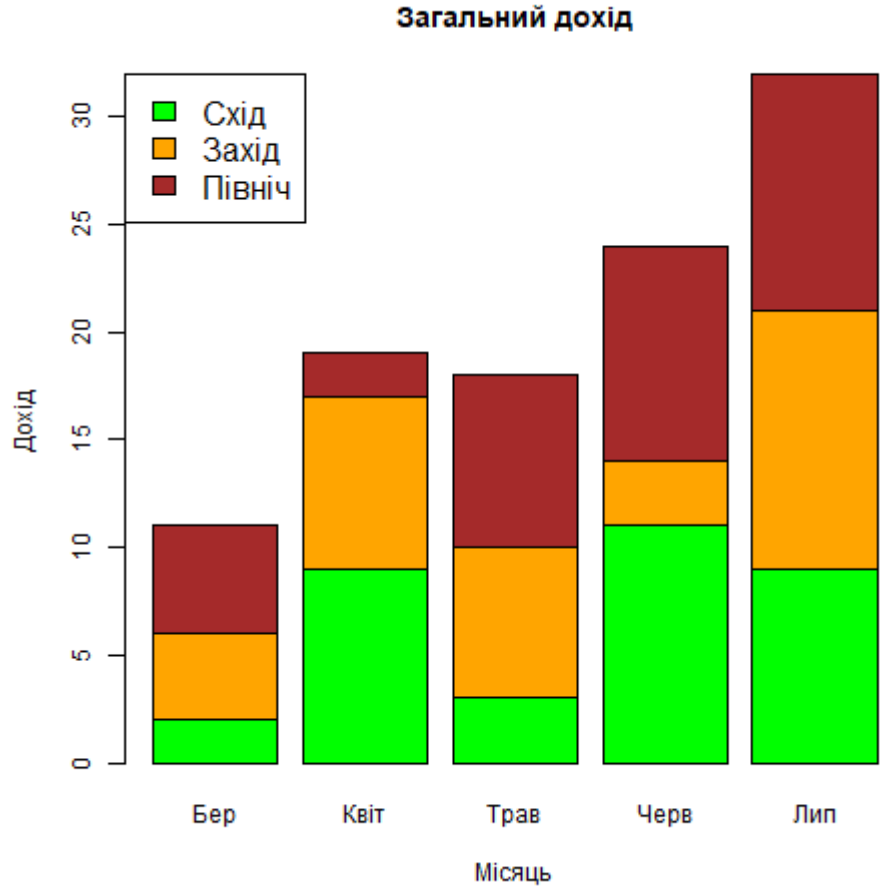
```
# Створення даних для діаграми у форматі звичайних векторів «H» та «M»
H <- c(7,12,28,3,41)
M <- c("Mar","Apr","May","Jun","Jul")
# Визначення назви файлу діаграми
png(file = "barchart_months_revenue.png")
# Побудова діаграми
barplot(H,names.arg=M,xlab="Місяць",ylab="Прибуток",col="blue",
main="Діаграма прибутку",border="red")
# Збереження зображення у файл
dev.off()
```



Групова стовпчикова діаграма стовпчикова діаграма з наповненням

Ми можемо створити гістограму з групами смуг і стеків у кожному рядку, використовуючи матрицю як вхідні значення. Більш ніж дві змінні представлені у вигляді матриці, яка використовується для створення групової стовпчикової діаграми та стовпчикової діаграми з наповненням.

```
# Створення вхідних векторів «colors», «months» та «regions»
colors <- c("green","orange","brown")
months <- c("Бер","Квіт","Трав","Черв","Лип")
regions <- c("Схід","Захід","Північ")
# Створення матриці значень «Values»
Values <- matrix(c(2,9,3,11,9,4,8,7,3,12,5,2,8,10,11),nrow=3,ncol=5,byrow=TRUE)
# Визначення назви файлу діаграми
png(file = "barchart_stacked.png")
# Побудова стовпчикової діаграми
barplot(Values,main="Загальний дохід",names.arg=months,xlab="Місяць",ylab="Дохід",col=colors)
# Додавання легенди
legend("topleft", regions, cex=1.3, fill=colors)
# Збереження зображення у файл
dev.off()
```



Секторна (кругова) діаграма

Секторна (кругова) діаграма - це представлення значень як скибочки кола з різними кольорами. У R кругова діаграма створюється за допомогою функції **pie ()**, яка приймає лише позитивні числа у формі вектору. Додаткові параметри використовуються для контролю підписів, кольору, назви тощо.

Синтаксис

Основний синтаксис для створення кругової діаграми з використанням R:

```
pie (x, labels, radius, main, col, clockwise)
```

Нижче наведено опис використовуваних параметрів:

– **x** – це вектор, що містить числові значення, які використовуються в круговій діаграмі;

– **labels** використовуються для опису секторів;

– **radius** вказує радіус кола кругової діаграми (значення між -1 та +1);

– **main** позначає назву графіка;

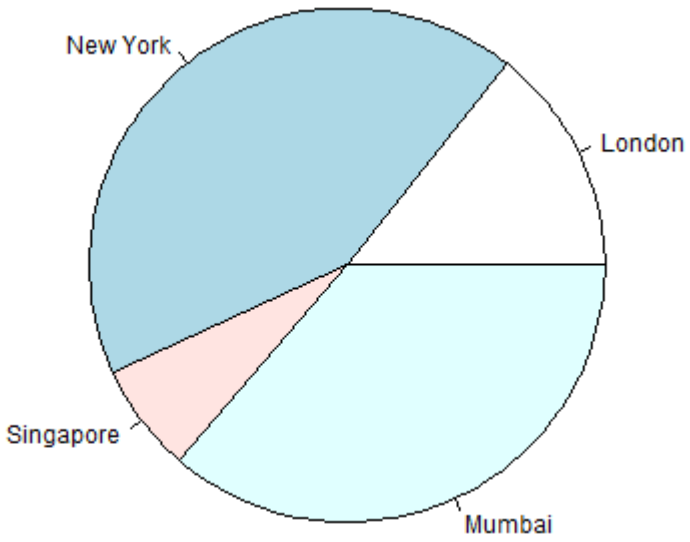
– **col** показує колірну палітру;

– **clockwise** - це логічне значення, яке вказує на те, чи зрізуються шматочки за годинниковою стрілкою або проти годинникової стрілки.

Приклад

Дуже проста кругова діаграма створюється лише за допомогою вхідного вектору та міток. Скрипт, приведений нижче, створить і збереже кругову діаграму в поточній робочий теці R.

```
# Створення даних для діаграми у вигляді вектору «x»
x <- c(21, 62, 10, 53)
labels <- c("London", "New York", "Singapore", "Mumbai")
# Визначення назви файлу діаграми
png(file = "city.jpg")
# Побудова діаграми
pie(x,labels)
# Збереження зображення у файл
dev.off()
```



Підпис та кольори кругової діаграми

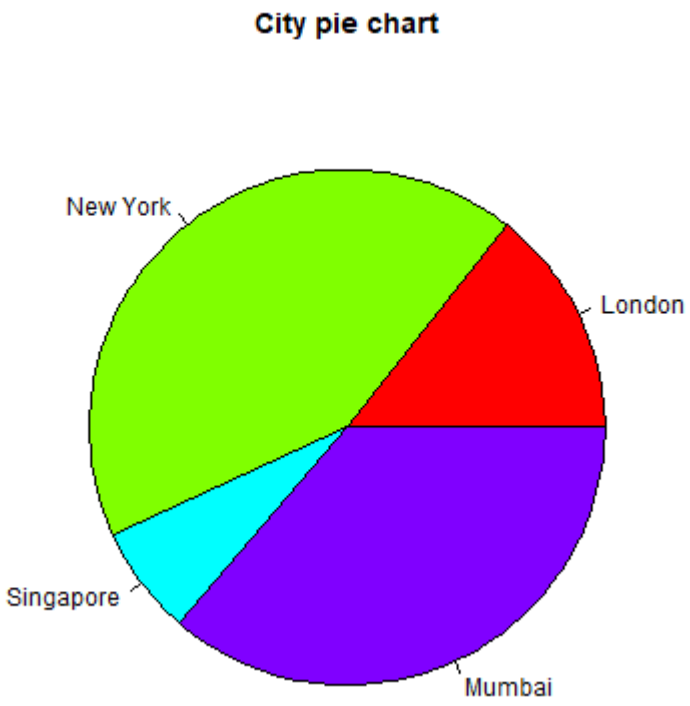
Ми можемо розширити можливості діаграми, додавши до функції більше параметрів. Ми будемо використовувати параметр **main**, щоб додати назву до

діаграми, а інший параметр - **col**, який буде використовувати райдужну палітру під час малювання діаграми. Довжина палітри повинна бути такою ж, як і кількість значень для діаграми. Отже, ми використовуємо **довжину (x)**.

Приклад

Нижній скрипт створить і збереже кругову діаграму в поточну робочу теку R.

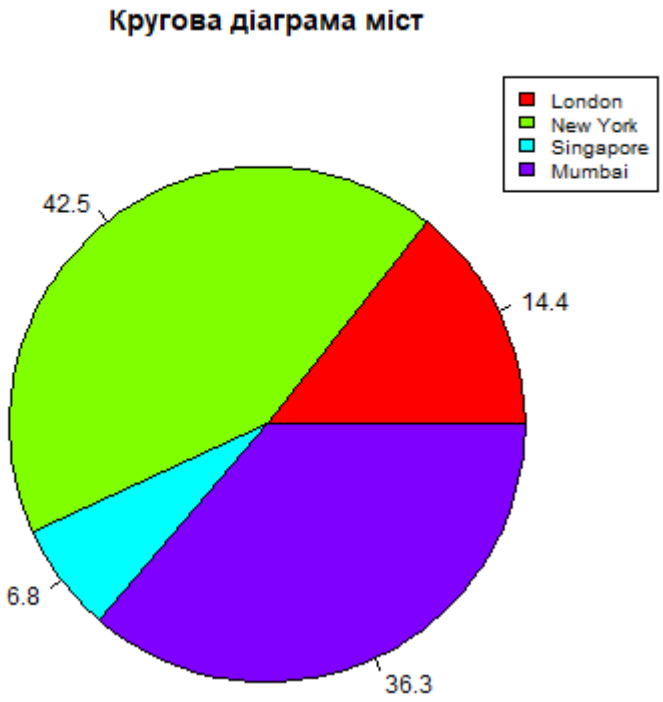
```
# Створення даних для діаграми.
x <- c(21, 62, 10, 53)
labels <- c("London", "New York", "Singapore", "Mumbai")
# Визначення назви файлу діаграми.
png(file = "city_title_colours.jpg")
# Plot the chart with title and rainbow color pallet.
pie(x, labels, main="City pie chart", col=rainbow(length(x)))
# Збереження зображення у файл.
dev.off()
```



Відсотки фрагментів та легенда

Ми можемо додати підписи відсотків та легенду діаграми.

```
# Створення даних для діаграми.
x <- c(21, 62, 10,53)
labels <- c("London","New York","Singapore","Mumbai")
piepercent<- round(100*x/sum(x), 1)
# Визначення назви файлу діаграми.
png(file = "city_percentage_legends.jpg")
# Побудова діаграми.
pie(x, labels=piepercent, main="Кругова діаграма міст",col=rainbow(length(x)))
legend("topright", c("London","New York","Singapore","Mumbai"), cex=0.8,
fill=rainbow(length(x)))
# Збереження зображення у файл.
dev.off()
```



Діаграми розсіювання

Діаграма розсіювання показують багато точок, побудованих в декартовій площині. Кожна точка представляє значення двох змінних. Одна змінна вибирається в горизонтальній осі, а інша - у вертикальній осі. Проста діаграма розсіювання створюється за допомогою функції **plot ()**.

Синтаксис

Основний синтаксис для створення розсіювання в R:

```
plot (x, y, main, xlab, ylab, xlim, ylim, axes)
```

Нижче наведено опис використовуваних параметрів:

– **x** - це набір даних, значення якого є горизонтальними координатами;

– **y** - це набір даних, значення якого є вертикальними координатами;

– **main** - це назва діаграми;

– **xlab** є міткою горизонтальної осі;

– **ylab** - це мітка вертикальної осі;

– **xlim** є межі значень x, які використовуються для побудови діаграми;

– **ylim** - це межа значень y, використаних для побудови діаграми;

– **axes** показує, чи повинні бути накреслені обидві осі на діаграмі.

Приклад

Ми використовуємо набір даних "mtcars", доступний у середовищі R, для створення базового розсіювання. Використовуємо стовпці "wt" та "mpg" у mtcars.

```
input <- mtcars[,c('wt','mpg')]
print(head(input))
```

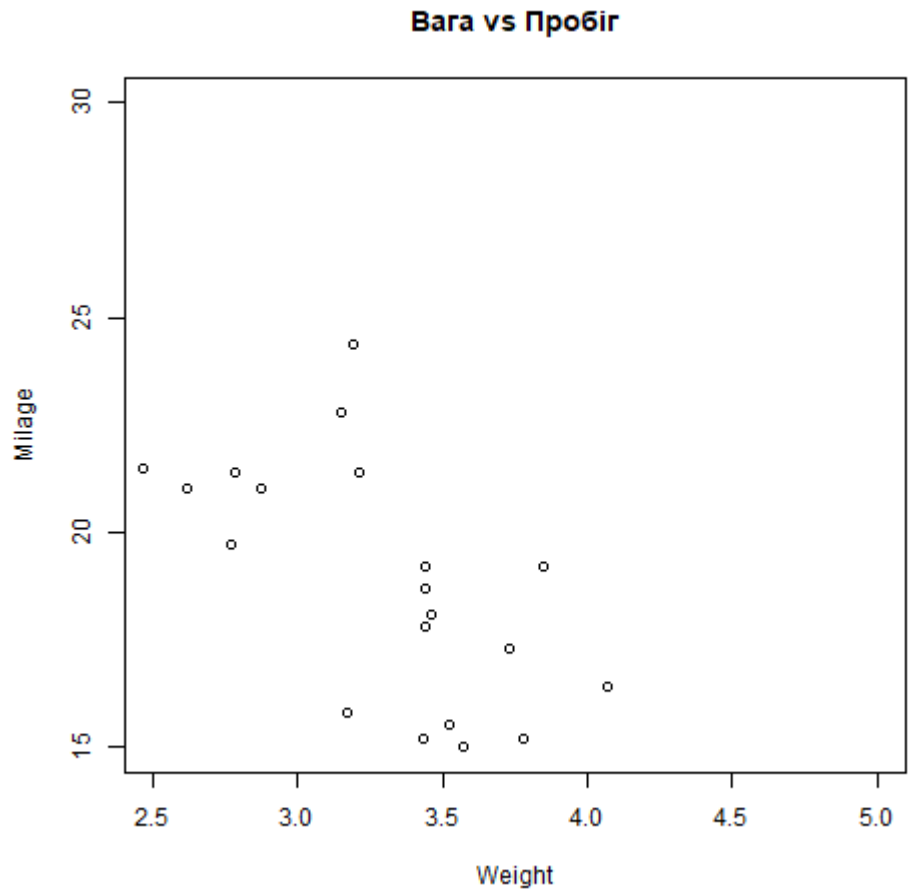
Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
                   wt  mpg
Mazda RX4         2.620 21.0
Mazda RX4 Wag     2.875 21.0
Datsun 710        2.320 22.8
Hornet 4 Drive    3.215 21.4
Hornet Sportabout 3.440 18.7
Valiant           3.460 18.1
```

Створення діаграми розсіювання

Нижній скрипт створить графік розсіювання для співвідношення між **wt (вага)** та **mpg (пробіг)**.

```
# Отримання значення для діаграми.
input <- mtcars[,c('wt','mpg')]
# Визначення назви файлу діаграми.
png(file = "scatterplot.png")
# Побудова діаграми для машин вагою від 2.5 до 5 та пробігом від 15 до 30.
plot(x=input$wt,y=input$mpg,
      xlab="Weight",
      ylab="Milage",
      xlim=c(2.5,5),
      ylim=c(15,30),
      main="Вага vs Пробіг")
# Збереження діаграми у файл.
dev.off()
```



Матриці розсіювання

Коли ми маємо більше двох змінних і ми хочемо знайти кореляцію між однією змінною та іншими, ми використовуємо матрицю розсіювання. Для цього використовуємо функцію **pairs ()**.

Синтаксис

Основний синтаксис для створення матриць розсіювання в R:

```
pairs (formula, data)
```

Нижче наведено опис використовуваних параметрів:

– **formula** представляє собою ряд змінних, що використовуються парами;

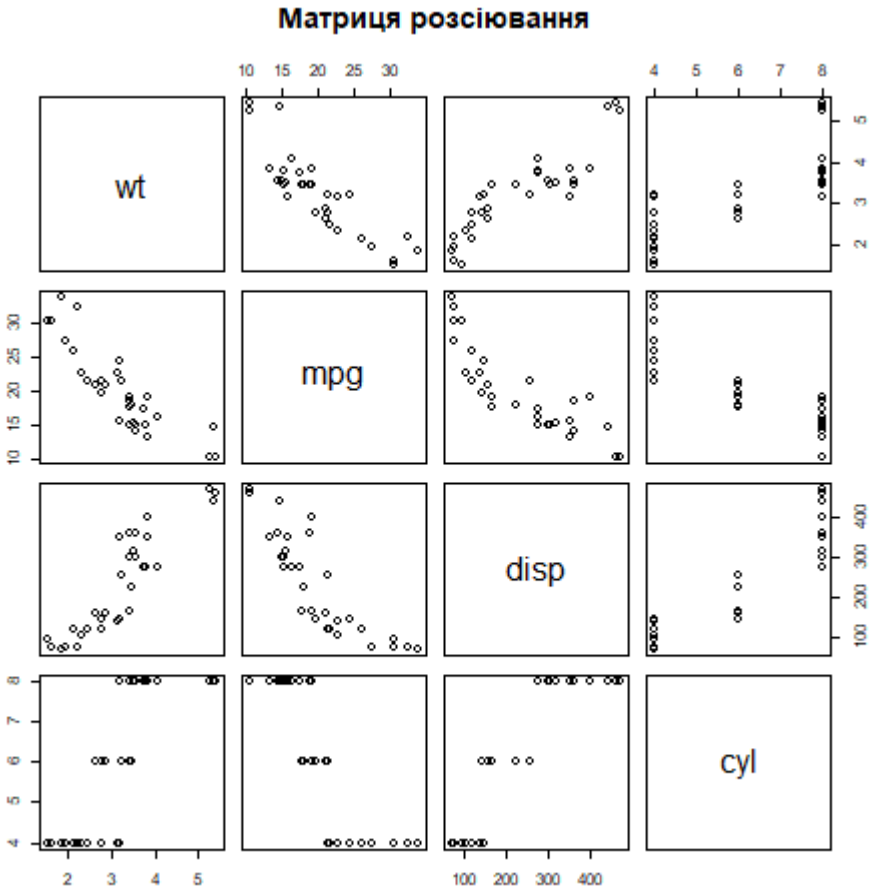
– **data** представляє набір даних, з якого будуть використовуватися змінні.

Приклад

Кожна змінна спарована з іншою змінною. Діаграма будується для кожної пари.

```
# Визначення назви файлу діаграми.
png(file = "scatterplot_matrices.png")
# Побудувати діаграму для 4 змінних у 12 графіках.
# Одна змінна з 3 іншими – в цілому 4 змінних.
pairs(~wt+mpg+disp+cyl,data=mtcars,main="Матриця розсіювання")
# Збереження діаграми у файл.
dev.off()
```

Коли виконується вищезгаданий код, ми отримуємо такий результат.



Лінійна діаграма

Лінійна діаграма – це графік, який з'єднує ряд точок шляхом нанесення сегментів лінії між ними. Ці точки впорядковані за однією їх координатою (зазвичай, x-координатою). Лінійні діаграми зазвичай використовуються для визначення тенденцій в даних. Функція **plot ()** в R використовується для створення лінійного графа.

Синтаксис

Основний синтаксис для створення діаграми ліній в R:

```
plot (v,type,col,xlab,ylab)
```

Нижче наведено опис використовуваних параметрів:

– **v** - це вектор, що містить числові значення;

– **type** отримує значення "p", щоб намалювати лише точки, "l" тільки лінії та "o", щоб намалювати як точки, так і лінії;

– **xlab** - мітка для осі x;

– **ylab** - мітка для осі y;

– **main** - назва діаграми;

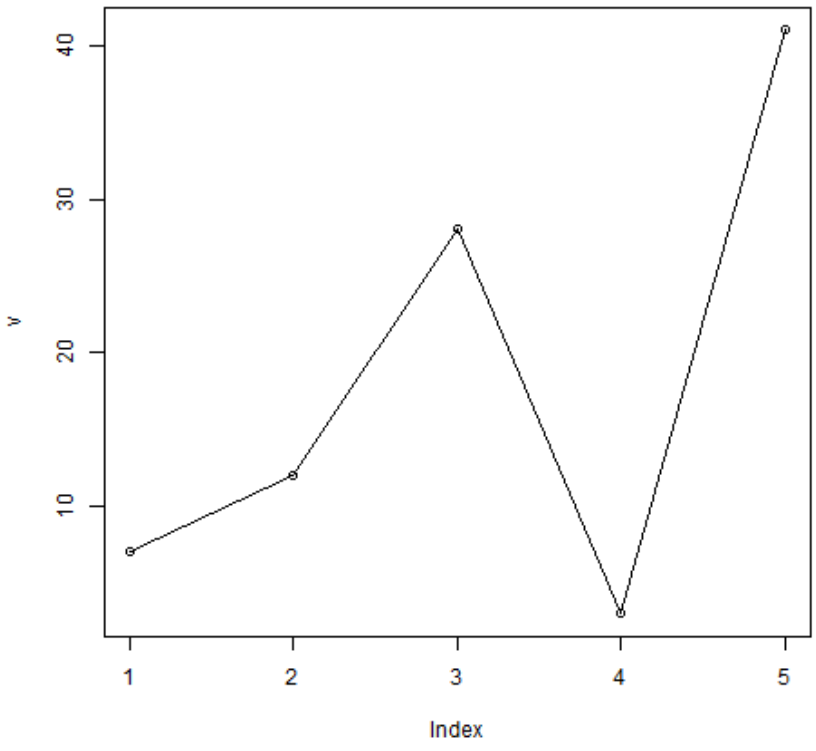
– **col** використовується для надання кольорів як точкам, так і лініям.

Приклад

Проста лінійна діаграма створюється за допомогою вхідного вектора та параметра типу як "O". Нижній сценарій створить та збереже лінійну діаграму у поточній робочій теці R.

```
# Створення даних для діаграми.
v <- c(7,12,28,3,41)
# Визначення назви файлу діаграми.
png(file = "line_chart.jpg")
# Побудова діаграми.
plot(v,type="o")
# Збереження діаграми у файл.
dev.off()
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

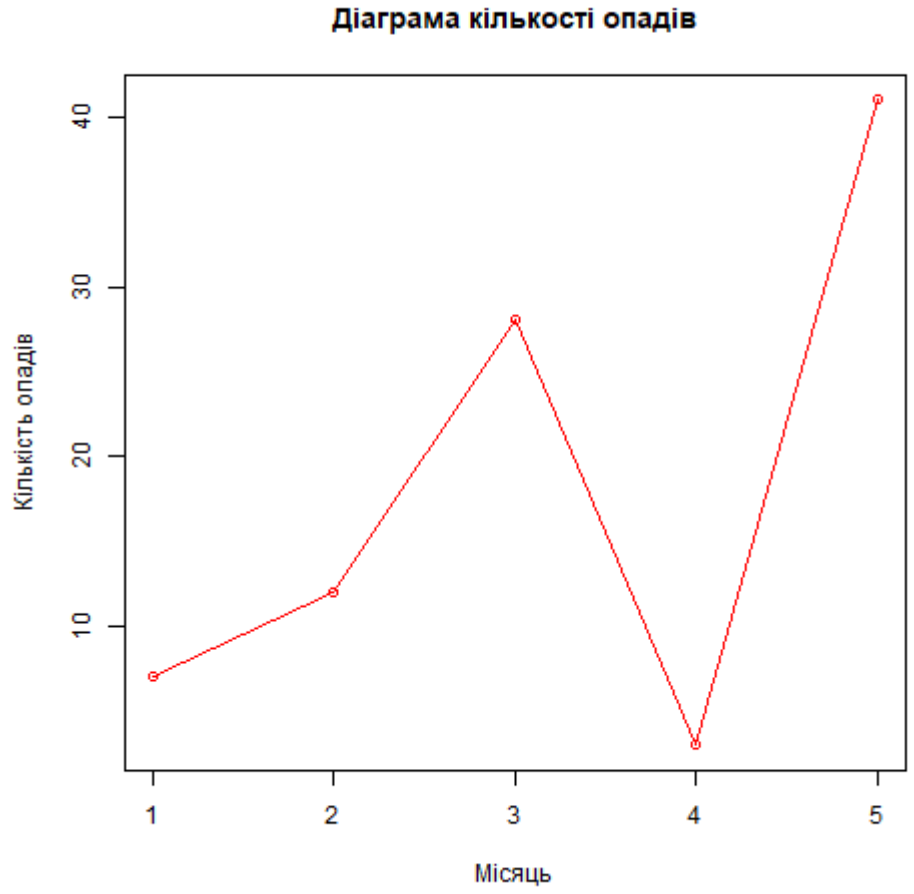


Заголовок діаграми, колір та мітки лінії

Функції лінійної діаграми можна розширити за допомогою додаткових параметрів. Ми додаємо колір до точок і ліній, даємо назву до діаграми та додаємо підписи до осей.

Приклад

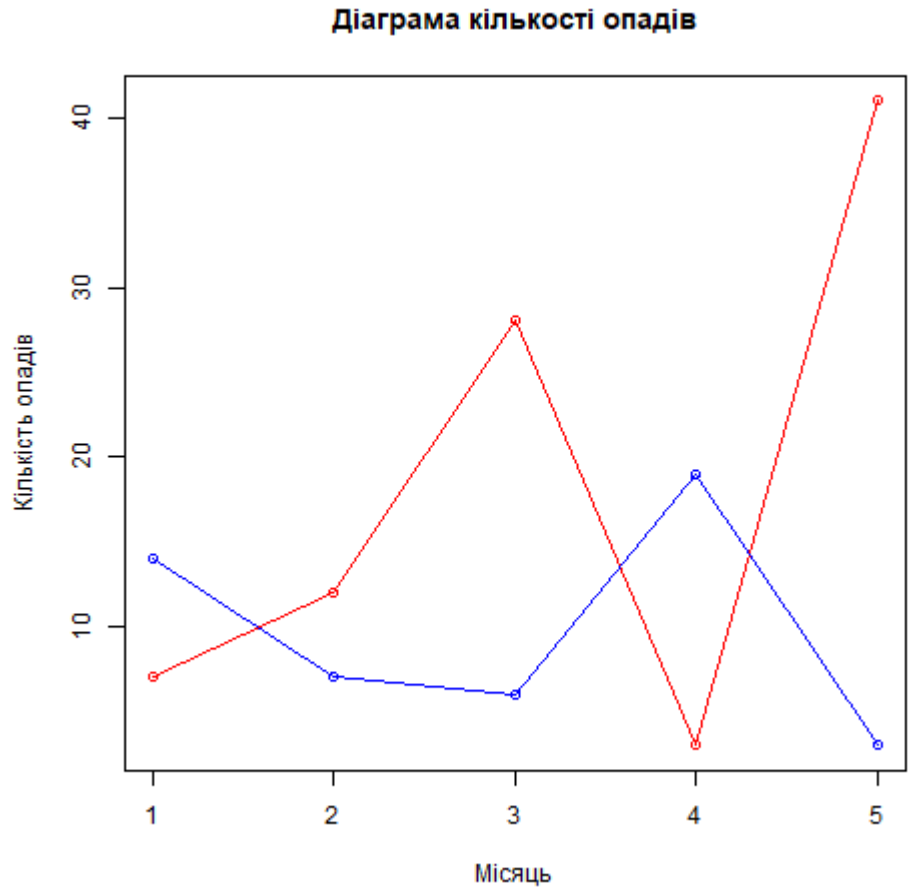
```
# Створення даних для діаграми.
v <- c(7,12,28,3,41)
# Визначення назви файлу діаграми.
png(file = "line_chart_label_colored.jpg")
# Побудова діаграми.
plot(v,type="o",col="red",xlab="Місяць",ylab="Кількість опадів",main="Діаграма
кількості опадів")
# Збереження діаграми у файл.
dev.off()
```



Лінійна діаграма з декількома лініями

На одному графіку можна зобразити більше однієї лінії, використовуючи функцію **lines ()**. Після того, як побудовано графік для першого рядка, функція **lines ()** може використовувати додатковий вектор як основу для малювання другої лінії в діаграмі.

```
# Створення даних для діаграми.
v <- c(7,12,28,3,41)
t <- c(14,7,6,19,3)
# Визначення назви файлу діаграми.
png(file = "line_chart_2_lines.jpg")
# Побудова діаграми.
plot(v,type="o",col="red",xlab="Місяць",ylab="Кількість опадів",main="Діаграма
кількості опадів")
lines(t, type="o", col="blue")
# Збереження діаграми у файл.
dev.off()
```



Вусикова діаграма (boxplot)

Вусикова діаграма - це показник того, яким чином розкидані значення в наборі даних. Він розподіляє дані на три квартилі. Діаграма показує одночасно 5 базових статистик: мінімальне значення, максимальне, середнє, медіану, перший та третій квартилі. Графік також корисний для порівняння розподілу даних за різними наборами даних. Вусикова діаграма створюються в R за допомогою функції **boxplot ()**.

Синтаксис

Основний синтаксис для створення boxplot в R:

```
boxplot (x, data, notch, varwidth, names, main)
```

Нижче наведено опис використовуваних параметрів:

– **x** – вектор або формула;

– **data** – це назва масиву даних;

– **notch** – логічне значення для побудови вусиків;

– **varwidth** – логічне значення для побудови графіку пропорційно розміру вибірки;

– **names** – це групове позначення, яке буде надруковано під кожним боксом;

– **main** – назва діаграми.

Приклад

Ми використовуємо набір даних "mtcars", доступний у середовищі R, для створення вусикової діаграми. Давайте подивимося на стовпці "mpg" та "cyl" у mtcars.

```
input <- mtcars[,c('mpg','cyl')]
print(head(input))
```

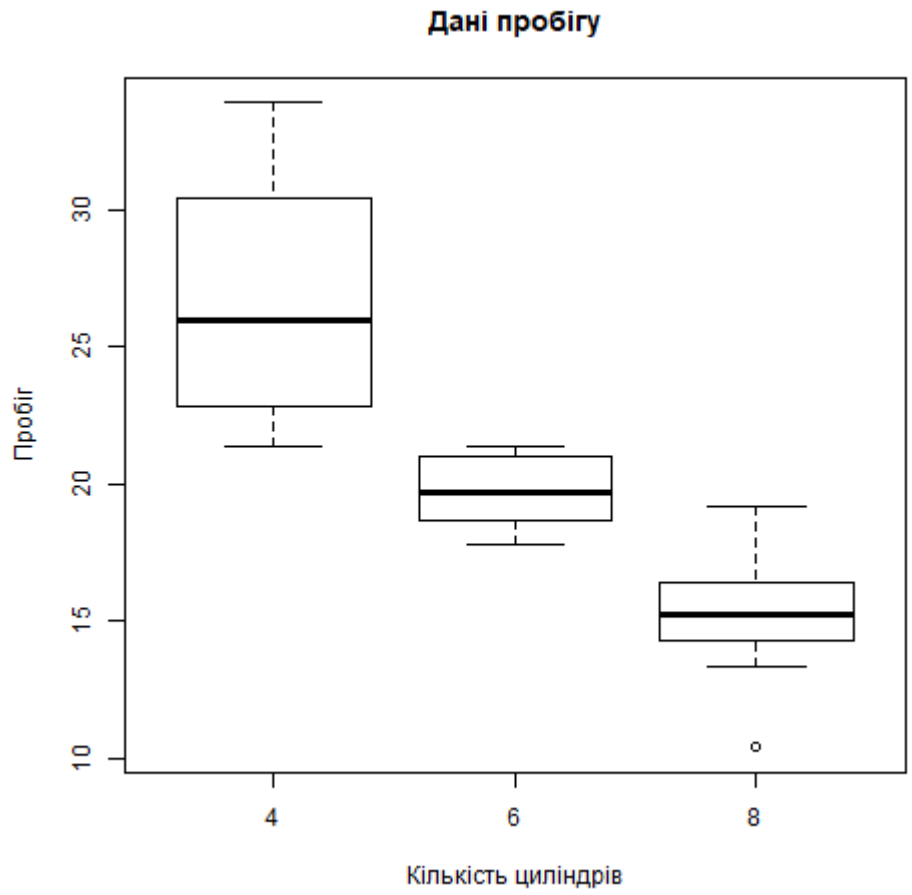
Коли ми виконуємо вищезазначений код, він призводить до наступного результату:

```
                   mpg cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
```

Нижній скрипт створить вусикову діаграму для співвідношення між mpg (пробіг) та cyl (кількість циліндрів).

```
# Визначення назви файлу діаграми.
png(file = "boxplot.png")
# Побудова діаграми.
boxplot(mpg ~ cyl, data=mtcars,
         xlab="Кількість циліндрів",
         ylab="Пробіг",
         main="Дані пробігу")
# Збереження діаграми у файл.
dev.off()
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:



Гістограма

Гістограма – це зображення частоти певних значень змінної. Гістограма схожа на стовпчикову діаграму, але різниця полягає в тому, що вона групує значення в безперервні діапазони. Кожен рядок в гістограмі відображає висоту числа значень, що знаходяться в цьому діапазоні. R створює гістограму за допомогою функції **hist ()**. Ця функція приймає вектор як вхід і використовує ще кілька параметрів для складання гістограм.

Синтаксис

Основний синтаксис для створення гістограми за допомогою R:

```
hist (v, main, xlab, xlim, ylim, break, col, border)
```

Нижче наведено опис використовуваних параметрів:

– **v** - вектор, що містить числові значення, які використовуються в гістограмі;

– **main** – назва діаграми;

– **col** використовується для встановлення кольору стрижнів.

– **border** використовується для встановлення кольору межі кожного стовбця;

– **xlab** – опис осі x.

– **xlim** використовується для визначення діапазону значень на осі x.

– **ylim** використовується для визначення діапазону значень на осі y.

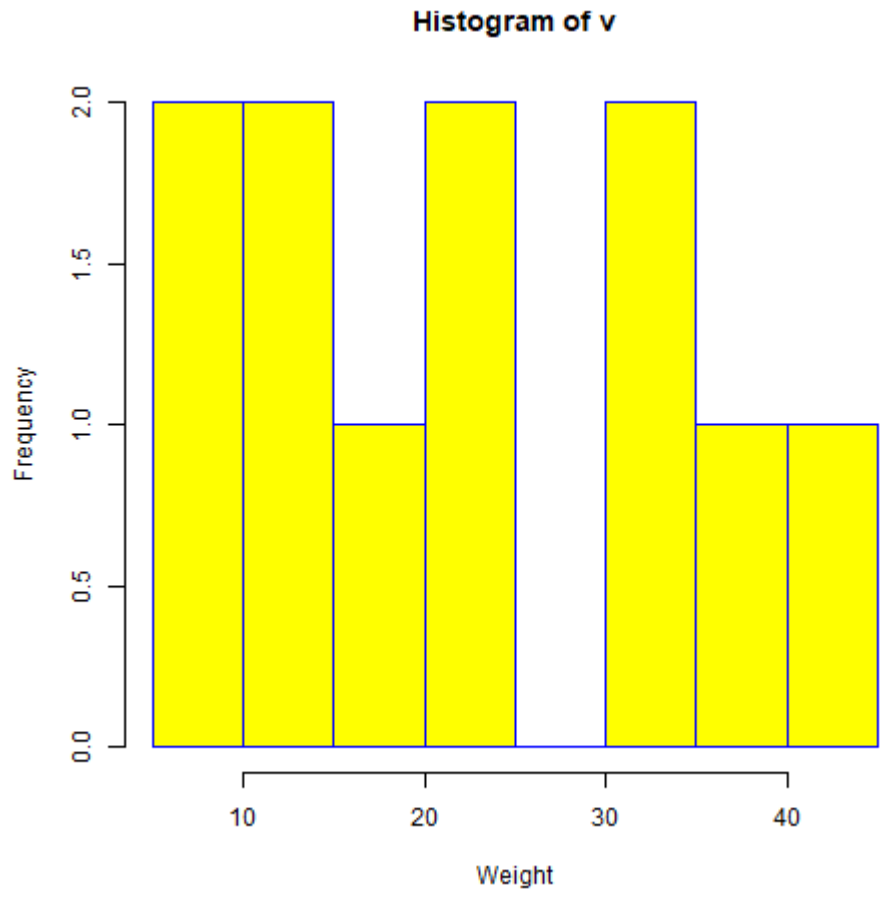
– **breaks** визначає ширину кожного стовпчика.

Приклад

Наведений нижче скрипт створить та збереже гістограму у поточному робочому каталозі R.

```
# Створення даних для діаграми.
v <- c(9,13,21,8,36,22,12,41,31,33,19)
# Визначення назви файлу діаграми.
png(file = "histogram.png")
# Побудова гістограми.
hist(v,xlab="Weight",col="yellow",border="blue")
# Збереження діаграми у файл.
dev.off()
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

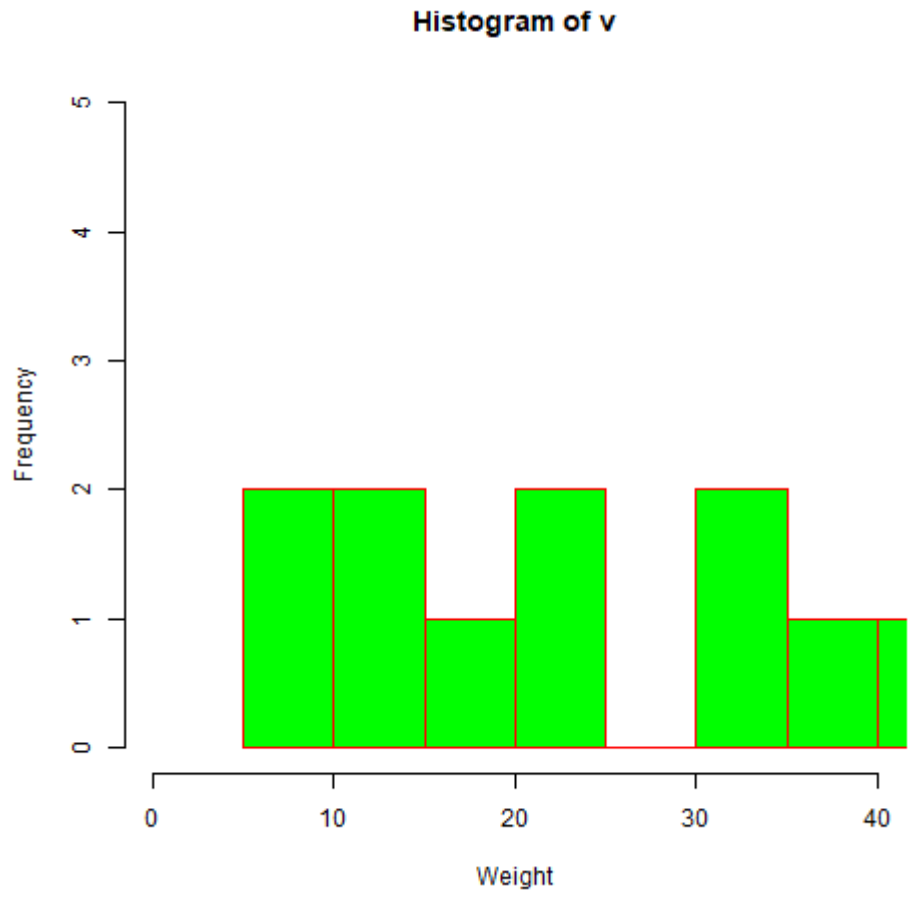


Діапазон значень X і Y

Щоб вказати діапазон значень, дозволених в осі X і осі Y, ми можемо використовувати параметри **xlim** та **ylim**. Ширина кожного стовбця може бути зазначена за допомогою параметра **breaks**.

```
# Створення даних для діаграми.
v <- c(9,13,21,8,36,22,12,41,31,33,19)
# Визначення назви файлу діаграми.
png(file = "histogram_lim_breaks.png")
# Побудова гістограми.
hist(v,xlab="Weight",col="green",border="red",xlim = c(0,40), ylim = c(0,5),
      breaks = 5 )
# Збереження діаграми у файл.
dev.off()
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:



3.3. Розрахунок описових статистик в R

Статистичний аналіз у R виконується за допомогою багатьох вбудованих функцій. Більшість цих функцій є частиною базового пакету R. Ці функції беруть

вектор R як вхідні значення разом з аргументами, які дають відповідний результат.

Середнє арифметичне

Розраховується шляхом сумування значень вектору та поділу на кількість значень у серії даних. Функція **mean ()** використовується для розрахунку цього в R.

Синтаксис

Основний синтаксис обчислення середнього значення в R:

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Нижче наведено опис використовуваних параметрів:

– **x** є вхідним вектором.

– **trim** використовується для видалення деяких спостережень з обох кінців сортованого вектора.

– **na.rm** використовується для видалення пропущених значень з вхідного вектора.

Приклад

```
# Створення вектору.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
# Визначення середнього арифметичного.
result.mean <- mean(x)
print(result.mean)
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
[1] 8.22
```

Застосування параметру Trim

При використанні параметру **Trim**, відбувається сортування всіх значень вектора, а потім необхідне число спостережень видаляється з розрахунку середнього значення. Коли Trim = 0.3, це значить що 3 значення з кожного кінця будуть виключені з розрахунків, щоб знайти середнє арифметичне. У цьому випадку у векторі (-21, -5, 2, 3, 4.2, 7, 8, 12, 18, 54) видаляються з обчислення середнього значення (-21, -5,2) з лівого і (12, 18, 54) з правого боку.

```
# Створення вектору.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
# Визначення середнього арифметичного.
result.mean <- mean(x,trim=0.3)
print(result.mean)
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
[1] 5.55
```

Застосування варіанта NA

Якщо у векторі є відсутні значення, то при визначенні середного арифметичного ми отримуємо значення NA. Щоб викинути відсутні значення з розрахунку, треба використовувати параметр **na.rm = TRUE**.

```
# Створення вектору.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
# Визначення середнього арифметичного.
result.mean <- mean(x)
print(result.mean)
# Визначення середенього арифметичного з видалення пропущених значень.
result.mean <- mean(x,na.rm=TRUE)
print(result.mean)
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
[1] NA
[1] 8.22
```

Медіана

Середнє значення у серії даних називається медіаною. Функція **median ()** використовується в R для обчислення цього значення.

Синтаксис

Основний синтаксис для обчислення медіана в R:

```
median(x, na.rm = FALSE)
```

Нижче наведено опис використовуваних параметрів:

– **x** є вхідним вектором.

– **na.rm** використовується для видалення відсутніх значень з вхідного вектора.

Приклад

```
# Створення вектору.
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
# Визначення медіани.
median.result <- median(x)
print(median.result)
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
[1] 5.6
```

Мода

Мода – це значення з найбільшою кількість випадків у наборі даних. На відміну від середнього арифметичного та медіани, мода може вираховуватися як

для цифрових, так і символьних даних. R не має стандартної вбудованої функції для обчислення моди. Можна, наприклад, створити функцію для обчислення моди в наборі даних в R. Ця функція приймає вектор як вхід і дає значення моди як результат.

Приклад

```
# Створення функції.
getmode <- function(v) {
uniqv <- unique(v)
uniqv[which.max(tabulate(match(v, uniqv)))]
}
# Створення числового вектору.
v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)
# Визначення моди за допомогою створеної функції.
result <- getmode(v)
print(result)
# Створення символьного вектору.
charv <- c("o","it","the","it","it")
# Визначення моди за допомогою створеної функції.
result <- getmode(charv)
print(result)
```

Коли ми виконуємо вищезазначений код, він видає наступний результат:

```
[1] 2
[1] "it"
```

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Andrea Cirillo (2016) RStudio for R Statistical Computing Cookbook, Packt Publishing, 247 p.
2. R programming (2016), Tutorials Point, 196 p.
3. James E. Monogan III (2015) Political Analysis Using R, Springer, 248 p.
4. Winston Chang (2013) R Graphics Cookbook, O'Reilly, 413 p.