Statystyka i programowanie w R

Aleš Kozubík
Uniwersytet Żyliński w Żylinie

Project: Innovative Open Source Courses for Computer Science

31. 5. 2021

Co-funded by the
Erasmus+ Programme
of the European Union

Innovative Open Source Courses for Computer Science

This teaching material was written as one of the outputs of the project
"Innovative Open Source Courses for Computer Science",
funded by the Erasmus+ grant no. 2019-1-PL01-KA203-065564.
The project is coordinated by West Pomeranian University of Technology in Szczecin (Poland)
and is implemented in partnership with Mendel University in Brno (Czech Republic)
and University of Žilina (Slovak Republic).
The project implementation timeline is September 2019 to December 2022.

Innovative Open Source Courses for Computer Science

Project was implemented under the Erasmus+.
Project name: "Innovative Open Source courses for Computer Science curriculum"
Project no.: 2019-1-PL01-KA203-065564
Key Action: KA2 – Cooperation for innovation and the exchange of good practices
Action Type: KA203 – Strategic Partnerships for higher education

Consortium: Zachodniopomorski uniwersytet technologiczny w Szczecinie
Mendelova univerzita v Brně
Žilinská univerzita v Žiline

Erasmus+ Disclaimer: This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Statystyka i programowanie w R

I. Wprowadzenie do R

Charakterystyka języka R

- jest językiem i wolnym środowiskiem programistycznym specjalizowanym do obliczeń statystycznych i wizualizacji danych,
- jest dostępny dla wszystkich popularnych systemów operacyjnych: platform UNIX, Win i MacOS,
- jest alternatywą dla komercyjnego narzędzia S lub S-plus (język i środowisko) opracowanego przez AT&T.

Dlaczego R

- jest darmowy, większość platform oprogramowania statystycznego kosztuje tysiące dolarów,
- dla programu dostępna jest duża liczba pakietów rozszerzających,
- R może łatwo importować dane z różnych źródeł,
- w R zaimplementowano wiele zaawansowanych narzędzi statystycznych,
- R zapewnia interaktywną platformę do analizy danych,
- środowisko R oferuje wizualizację danych w postaci wysokiej jakości i estetycznych wykresów,
- jest niezależny od platformy, kompatybilny z większością najczęściej używanych systemów operacyjnych,
- jest kompatybilny z językami programowania takimi jak C, C++, Python, Java.

Instalacja R

Jest on dostępny za darmo w Comprehensive R Archive Network (w skrócie CRAN)..

Jest on dostępny w Internecie pod adresem https://cran.r-project.org.

Prekompilowane binaria są dostępne dla popularnych platform Linux, Mac OS i Windows.

Możemy wybrać odpowiedni serwer lustrzany, aby pobrać pakiet instalacyjny.

Instalacja pakietów R

Istnieje bogaty zestaw pakietów, które rozszerzają funkcjonalność jądra R.

Pakiety zwiększają wydajność R.

zainstalować pakiety, używamy funkcji `install.packages()`

Pierwsze uruchomienie R

eśli mamy zainstalowany R, możemy sprawdzić jego funkcjonalność.

Prostredie R spustíme jednoducho z príkazového riadku zadaním typing:

```
username@host:~$ R
```

Wyświetlane jest krótkie wprowadzenie, po którym następuje

```
>
```

Ten symbol jest znakiem wiersza poleceń środowiska R.

Opuszczenie środowiska R

Środowisko R jest teraz gotowe do pracy..

Aby wyjść ze środowiska R, wystarczy wpisać

```
> q()
```

R odpowiada pytaniem:

```
Save workspace image? [y/n/c]:
```

Jeśli wybrano `y`, cała historia wykonanych poleceń jest zapisywana w pliku `.Rhistory`, który jest zapisywany w katalogu roboczym.

Obszar roboczy i nawigacja

Wszystkie polecenia są wprowadzane interaktywnie w wierszu poleceń.

Po historii poleceń poruszamy się za pomocą klawiszy kursora, strzałek w górę i w dół.

Dzięki temu możemy wracać do starszych komend bez konieczności pisania ich od nowa. Wystarczy wybrać żądane polecenie i wysłać je ponownie za pomocą klawisza Enter.

Jeśli po opuszczeniu środowiska zapiszemy naszą historię, możemy również powrócić do poleceń z poprzedniej sesji.

Komunikacja z OS

Domyślnym katalogiem roboczym jest katalog, w którym został uruchomiony program R. W tym bieżącym katalogu roboczym R odczytuje i zapisuje pliki i wyniki. Sprawdzamy aktualny katalog roboczy za pomocą funkcji `getwd()`.

Bieżący katalog roboczy może zostać zmieniony przy użyciu funkcji `setwd()`.

Aby wykonać polecenia systemu operacyjnego, należy użyć funkcji `system()`.

twórz nowy katalog używając

```
> system("mkdir new")
```

Szukanie pomocy

Funkcja uzyskiwania pomocy ma zazwyczaj prostą formę `help()` lub jest skracana przez operator `?`.

Jeśli chcemy uzyskać informacje o pakietach rozszerzeń, używamy

```
> help(package="nazwa pakietu")
```

Niektóre pakiety zawierają również próbki kodu, które uruchamiamy za pomocą funkcji `demo()`, na przykład

```
> demo(package="stats")
```

R jak kalkulator

Konsola wiersza poleceń umożliwia interaktywne obliczanie wyników operacji i funkcji

```
> 5+3
[1] 8
```

Jeśli nie widzimy początkowego znaku linii poleceń, może to być spowodowane tym, że nie wpisaliśmy kompletnego polecenia

```
> 5-
+
```

Musimy dokończyć resztę polecenia, a następnie wcisnąć Enter lub anulować polecenie wciskając Esc.

Operacje arytmetyczne w R

+	Liczenie.
-	Odejmowanie.
*	Mnożenie.
/	Dzielenie.
^	Potęgowanie.
%%	modulo (reszta z dzielenia liczb całkowitych).
%/%	ułamek całkowity.

Operatory relacji R

<	Mniejszy
>	Większy
<=	Mniejszy lub równy
>=	Większy lub równy
==	Równa się
!=	Nie jest równy

Często używane funkcje matematyczne

`exp()`	Eksponencjalna e^x
`log()`	Logarytm (domyślnie naturalny)
`log10()`	Logarytm o podstawie 10 10
`sin()`	Sínus
`cos()`	Kosínus
`tan()`	Tangens
`round()`	Zaokrąglanie (domyślnie liczba całkowita)
`sqrt()`	Pierwiastek kwadratowy
`abs()`	Absolutna wartość
`asin()`	Arkussínus
`acos()`	Arkuskosínus
`atan()`	Arkustangens

Obiekty

R jest językiem zorientowanym obiektowo

W R wszystko jest obiektem i reprezentuje jakieś dane, które zostały zapisane w pamięci

Obiekty mogą mieć dowolne nazwy, ale należy przestrzegać następujących zasad:

- Nazwa artykułu składa się wyłącznie z dużych lub małych liter, cyfr, podkreśleń i kropek„
- Nazwa artykułu zaczyna się od dużej lub małej litery,
- W języku R rozróżniana jest wielkość liter (to znaczy, że A i a są dwoma różnymi obiektami),
- Nazwa elementu nie może zawierać żadnego z zastrzeżonych słów języka R (możesz zobaczyć ich listę wpisując `help(reserved)`).

Tworzenie obiektu

Nowy obiekt tworzy się w prosty sposób za pomocą operatora przypisania.

Operator przypisania ma dwie możliwe formy: <- lub =.

Zaleca się używanie formy <-, ponieważ forma = może czasami powodować błędy:

```
> log(x=25,base=5)
[1] 2
> x
Error: object 'x' not found
> log(x<-25,base=5)
[1] 2
> x
[1] 25
```

Listowanie i usuwanie obiektów

Listę wszystkich istniejących obiektów otrzymujemy jako wyjście funkcji `ls()`.

Obiekty, które nie będą używane w przyszłości, mogą zostać usunięte z pamięci za pomocą funkcji `rm()`.

Statystyka i programowanie w R

II. Struktury danych w R

Typy i struktury danych

Istnieje 5 elementarnych typów danych

- numeric,
- integer,
- complex,
- logical,
- character.

Mogą one być agregowane w struktury danych:

- vector,
- matrix,
- list,
- frame.

Typ danych `numeric`

Typ danych `numeric` reprezentuje rzeczywiste liczby dziesiętne.

Jest to domyślny typ każdego nowego obiektu.

Powstaje ona, gdy do dowolnej zmiennej przypiszemy liczbę rzeczywistą.

Typ dowolnego obiektu jest sprawdzany za pomocą funkcji `class()`.

Typ danych `numeric` – przykład

Przyjrzyjmy się przykładowi.

```
1 > x<-12.35
2 > class(x)
3 [1] "numeryczny"
```

Note

Liczba ta jest reprezentowana jako wektor o długości 1. Znak `[1]` na początku linii oznacza pierwszą pozycję w tym wektorze.

Typ danych integer

Aby utworzyć obiekt typu integer, używamy funkcji `as.integer()`.

Przykład

```
> a <- as.integer(12)
> a
[1] 12
> class(a)
[1] "integer"
> is.integer(a)
[1] TRUE
```

Zmiana typu zmiennej

Przy wszelkich obliczeniach należy pamiętać, że typ zmiennej może się zmieniać.

Przykład

```
> x<-as.integer(20)
> class(x)
[1] "integer"
> x<-x/3+1
> x
[1] 7.666667
> class(x)
[1] "numeric"
```

Typ danych `complex`

Środowisko R pozwala na pracę z liczbami złożonymi.

Wartość złożona jest określona w R przez jednostkę urojoną `i`

Przykład

```
> z<-1+2i
> class(z)
[1] "complex"
```

Typ danych `logical`

Może przyjmować dwie wartości logiczne `TRUE` lub `FALSE`.

Często tworzone przez porównywanie zmiennych.

```
1  > x<-10;y<-20
2  > z<-x<y
3  > z
4  [1] TRUE
5  > class(z)
6  [1] "logical"
```

Typ danych `logical`

Zdefiniowane są dla nich wszystkie standardowe operacje logiczne

`&`	Logiczne `AND`
`|`	Logicczne `OR`
`+!+`	Negacja

Typ danych character

Służy do przechowywania ciągów znaków, ciągi są wprowadzane z użyciem cudzysłowów

```
> x<-"facina"
> class(x)
[1] "character"
#Ale i
> x<-as.character(3.1415926)
> x
[1] "3.1415926"
> class(x)
[1] "character"
```

Typ danych character

Łańcuchy znaków mogą być łączone za pomocą paste().

```
1  > name<-"Donald"
2  > surname<-"Knuth"
3  > paste(name,surname)
4  [1] "Donald␣Knuth"
5  # Jesli chcemy
6  > paste(iname,surname,sep=","))
7  [1] "Donald,Knuth"
```

Wektory

Wektor jest najprostszą strukturą danych.

Można ją scharakteryzować jako dostępność elementów tego samego typu danych.

Poszczególne wartości zawarte w wektorze są określane jako składowe.

Liczba składowych wektora jest określana jako jego długość.

Wektory

Wektor `v` jest tworzony za pomocą funkcji `c()`.

Jego długość jest znajdowana przy użyciu funkcji `length()`.

```
1  > v<-c(1,3,5,7,9)
2  > length(v)
3  [1] 5
```

Wektory

Wektory mogą być łączone przy użyciu “combine” `c()`.

```
1  >a<-c(1,2,3)
2  >b<-c(4,5,6)
3  >c(b,a)
4  [1] 4 5 6 1 2 3
5  # Patrz nadpisywanie komponentow
6  > a<-c("a", "b", "c")
7  > c(a,b)
8  [1] "a" "b" "c" "4" "5" "6"
```

Wektory – arytmetyka

Arytmetyka wektorowa jest implementowana komponent po komponencie.

Operacje arytmetyczne są implementowane komponent po komponencie.

- `+` dodawanie liczby do wszystkich składowych lub dodawanie wektorów po składowych
- `-` odejmuje liczbę od wszystkich składowych lub odejmuje wektory składowa po składowej,
- `*` pomnożyć wszystkie składowe przez liczbę lub pomnożyć wektory przez składowe,
- `/` dzielenie wszystkich składowych przez liczbę lub dzielenie wektorów przez składowe.

Wektory – arytmetyka

```
> v<-c(1,3,5,7,9)
> u<-c(10,20,30,40,50)
> u+v
[1] 11 23 35 47 59
> u-v
[1] 9 17 25 33 41
> 5*v
[1] 5 15 25 35 45
> u*v
[1] 10 60 150 280 450
> u/5
[1] 2 4 6 8 10
> u/v
[1] 10.000000 6.666667 6.000000 5.714286 5.555556
```

Macierz

Macierz to dwuwymiarowa tablica danych tego samego typu ułożonych w prostokątny schemat.

Tworzymy ją za pomocą funkcji `matrix()` z następującymi argumentami

`vector`	zawiera elementy macierzy,
`nrow`	jest wartością całkowitą, która wskazuje liczbę wierszy w mactierzy,
`ncol`	jest wartością całkowitą określającą liczbę kolumn macirzy,
`byrow`	jest wartością logiczną, która określa, czy macierz powinna być wypełniona wierszami (`byrows=TRUE`) czy kolumnami (`byrows=FALSE`), jej domyślną wartością jest `FALSE`,
`dimnames`	jest listą wektorów typu `character`, które zawierają opcjonalne etykiety wierszy i kolumn.

Macierz – przykładowe zadanie

vfill

```
> A<-matrix(3:8,nrow=3,ncol=2,byrow=TRUE)
> A
     [,1] [,2]
[1,]    3    4
[2,]    5    6
[3,]    7    8
> B<-matrix(3:8,nrow=3,ncol=2,byrow=FALSE)
> B
     [,1] [,2]
[1,]    3    6
[2,]    4    7
[3,]    5    8
```

Macierz – wybór podmacierzy

Wiersze i kolumny definiuje się za pomocą funkcji `c()`.

```
> C<-matrix(1:12,nrow=3)
> C
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> C[c(1,3),c(2,4)]
     [,1] [,2]
[1,]    4   10
[2,]    6   12
```

Macierz – przypisanie nazwy

Nazwy przypisujemy do wierszy i kolumn za pomocą funkcji `dimnames()` i `list()`.

```
> dimnames(A)<-list(c("row1","row2","row3"),
+ c("col1","col2"))
> A
     col1 col2
row1    3    4
row2    5    6
row3    7    8

> A["row2","col1"]
[1] 5
```

Macierz – transpozycja

Możemy transponować macierz za pomocą funkcji `t()`.

```
> B<-matrix(3:8,nrow=3,ncol=2,byrow=FALSE)
> t(B)
     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    7    8
```

Pozostałe funkcje są zdefiniowane w pakiecie `matlib`.

Macierz – operacje

Są one określane dla poszczególnych komponentów

Jest to ważne w mnożeniu macierzy. Wspólna operacja erb+*+ oznacza mnożenie elementów na tych samych pozycjach.

Standardowe mnożenie macierzy z algebry liniowej jest definiowane jako operacja %*%.

Array

Tablice są uogólnieniem macierzowej struktury danych.

W rzeczywistości są to macierze więcej niż dwuwymiarowe.

Tablice można tworzyć za pomocą funkcji `array()`.

Składnia tej funkcji jest następująca

```
name<-array(vector, dimensions,dimnames)
```

Pole – tworzenie

Utwórzmy tablicę o wymiarach $3 \times 4 \times 3$.

Aby lepiej poruszać się po tablicy, stwórzmy najpierw nazwy poszczególnych wymiarów.

```
1 > dim1<-c("A1", "A2", "A3")
2 > dim2<-c("B1", "B2", "B3", "B4")
3 > dim3<-c("C1", "C2", "C3")
```

Struktura data frame

Data frame jest najbardziej popularną strukturą do przechowywania danych.

Umożliwia przechowywanie wektorów kolumnowych różnych typów danych.

Data framy tworzone są za pomocą funkcji `data.frame()`, której ogólna składnia jest następująca

```
1 > name<-data.frame(col1,col2,col3, ...)
```

Data frame – tworzenie

.

Utwórz krótką ramkę danych zawierającą dane o strzałach koszykarzy.

```
> playerID<-c(1,2,3,4)
> position<-c("forward","guard","forward","center")
> attempted<-c(12,6,10,15)
> made<-c(7,4,6,12)
> players<-data.frame(playerID,position,attempted,made)
> players
  playerID position attempted made
1        1  forward        12    7
2        2    guard         6    4
3        3  forward        10    6
4        4   center        15   12
```

Data frame – łączenie

.

Często zachodzi potrzeba połączenia danych z dwóch lub więcej zbiorów danych.

Używamy funkcji `merge()`.

Argumenty są nazwami dwóch ramek danych, które mają zostać połączone.

Trzeci argument, `by=''column_name''`, określa zmienną, która ma zostać połączona z danymi.

Data frame – łączenie danych

Aby zademonstrować łączenie, najpierw tworzymy nową ramkę danych `rebounds`.

```
1 > offensive<-c(5,2,3,10)
2 > defensive<-c(6,3,8,12)
3 > rebounds<-data.frame(playerID,defensive,offensive)
4 > row.names(rebounds)<-c("Player1","Player2","Player3",
5 + "Player4")
```

Listy- `List`

Listy są najbardziej złożoną strukturą danych.

Reprezentują one uporządkowane kolekcje obiektów.

Aby utworzyć listę, użyj funkcji `list()`. Jego składnia jest prosta:

```
\list(object1,object2,...)
```

Jego argumentami są nazwy istniejących obiektów

Listy

Opcjonalnie nadaj nazwy obiektom na tworzonej liście:

```
\list(name1=object1,name2=object2,...)
```

Wejście z klawiatury

Najprostsza metoda (ale również najbardziej czasochłonna dla dużych próbek)

Pracujemy w dwóch etapach

- Utwórz pustą ramkę danych z nazwami i typami zmiennych, które chcemy przechowywać.
- Otwórz prosty edytor danych za pomocą funkcji *()+, której argumentem jest nazwa ramki danych, którą chcemy edytować.*

Wejście z klawiatury

Tworzymy pustą ramkę danych o nazwie `mydata` z czterema zmiennymi: `name`, która ma typ `character`, oraz trzy zmienne numeryczne age, `height` i `weight`.

```
1 > mydata<-data.frame(name=character(0),age=numeric(0),
2 + height=numeric(0),weight=numeric(0))
3 > mydata<-edit(mydata)
```

Uwaga

Zauważ, że przypisania takie jak `numeric(0)` i `character(0)` utworzą zmienną tego typu, ale bez danych.

Dane wejściowe z pliku .csv

Opcjonalne argumenty do `read.csv()`.

- `header` Wartość logiczna, określa czy plik wejściowy zawiera nazwy zmiennych jako pierwszą linię, wartość domyślna `TRUE`.
- `sep` określa znak oddzielający wpisy, domyślną wartością jest przecinek,
- `dec` określa znak używany w pliku dla dziesiętnych, domyślną wartością jest `.`, należy również wspomnieć o funkcji `read.csv2()`, która używa przecinka dla dziesiętnych i średnika jako separatora.
- `skip=n` określa liczbę linii do pominięcia przed odczytem danych. Opcja ta jest przydatna dla tabel danych z pustymi wierszami lub opisów tekstowych na początku plików.
- `stringsAsFactors`, która jest wartością logiczną określającą czy ciągi są konwertowane na czynniki, ustaw ją na `FALSE` jeśli chcesz zapobiec konwersji.
- `row.names` jest wektorem nazw wierszy.

Zapis danych do pliku `.csv`

R może utworzyć plik `csv` z istniejących data framu.

Używamy funkcji `write.csv()` lub `write.csv2()`, która używa przecinka jako kropki dziesiętnej i średnika jako separatora.

Wspólna składnia

```
write.csv(object,file="filename",...options)
```

`object` jest obowiązkowym argumentem zawierającym nazwę data framu, którą chcemy zapisać, a `filename` jest nazwą (lub pełną ścieżką) pliku.

Zapisywanie danych do pliku `.csv`

Wybrane opcje programu `write.csv()`.

- `append`, który jest wartością logiczną wskazującą, czy dane wyjściowe są dołączane do końca pliku. Domyślną wartością jest `FALSE` i wszystkie istniejące pliki o tej nazwie zostaną nadpisane.
- `sep` Znak separatora elementów jest definiowany przez czasownik `sep`. Wartości w każdej linii `object` są oddzielone tym znakiem.
- `dec` łańcuch, który ma być używany dla punktów dziesiętnych w kolumnach numerycznych lub złożonych, musi to być pojedynczy znak. Domyślną wartością jest kropka dziesiętna.
- `row.names` Wartość logiczna określająca, czy zapisywać nazwy wierszy`object`.

Dane wejściowe z plików Excela

Istnieje kilka pakietów, które pozwalają na import danych bezpośrednio z plików Excela. Przyjrzyjmy się niektórym z nich:

- `xlsx`,
- `XLconnect`
- `readxl`

Excel 2007 i późniejsze wersje używają formatu `xlsx`, więc tutaj wymienimy pakiet `xlsx`.

Dane wejściowe z plików Excela

Instalujemy pakiet za pomocą zwykłej komendy:

```
install.packages("xlsx")
```

Jeśli chcemy go użyć w bieżącym obszarze roboczym, ładujemy go w standardowy sposób:

```
library("xlsx")
```

Dane wejściowe z plików Excela

Pakiet ten dostarcza dwie funkcje do ładowania zawartości skoroszytu Excela do R data.frame: `read.xlsx()` i `read.xlsx2()`.

Różnica pomiędzy tymi dwoma funkcjami jest następująca:

- `read.xlsx()` zachowuje typ danych, typ zmiennej odpowiada każdej kolumnie w arkuszu, ale jest powolny dla dużych zestawów danych (arkusz z więcej niż 100 000 komórek). `read.xlsx2()` jest szybszy dla dużych plików.

Dane wejściowe z plików Excela

Obie funkcje mają podobną składnię:

```
read.xlsx(file, sheetIndex, header=TRUE, colClasses=NA)
read.xlsx2(file, sheetIndex, header=TRUE, colClasses="character")
```

Ich argumenty mają następujące znaczenie:

- `file` to nazwa pliku, który zawiera tabelę. Jeżeli plik nie znajduje się w katalogu roboczym, musi być podana pełna ścieżka.
- `sheetIndex` jest liczbą wskazującą indeks arkusza, który ma zostać odczytany. Można go zastąpić argumentem `sheetname`, podanym jako łańcuch znaków z nazwą arkusza.
- `header` wartość logiczna. Jeśli `header=TRUE`, to pierwsza linia jest używana jako konwencja nazewnictwa zmiennych.
- `colClasses` wektor znaków reprezentujący klasę każdej kolumny.
- `startRow`,liczby określające indeks początkowego i ostatniego załadowanego wiersza.

Zapisywanie danych do plików Excela

Pakiet `xlsx` udostępnia dwie funkcje zapisu `write.xlsx()` oraz `write.xlsx2()`.

Ogólna składnia

```
write.xlsx(x, file, sheetName="Sheet1", col.names=TRUE,
row.names=TRUE, append=FALSE)

write.xlsx2(x, file, sheetName="Sheet1",col.names=TRUE,
row.names=TRUE, append=FALSE)
```

Zapisywanie danych do plików Excela

Ich argumenty mają następujące znaczenie:

- `x` data frame, która jest zapisywana do skoroszytu.
- `file` nazwa (lub ścieżka) pliku wyjściowego.
- `sheetName` łańcuch znaków z nazwą arkusza.
- `col.names` Wartość logiczna określająca, czy zapisywać nazwy kolumn do pliku.
- `row.names` Wartość logiczna, określa czy zapisywać nazwy wierszy do pliku.
- `append` Wartość logiczna, określa czy `x` powinien być dołączony do istniejącego pliku, jeśli `FALSE`, nadpisuje istniejący plik w ten sam sposób.

Odczytywanie danych z plików JSON

JSON (JavaScript Object Notation) jest prostym formatem wymiany danych.

Aby móc czytać pliki JSON w R, musimy najpierw zainstalować lub załadować pakiet `rjson`.

Możemy użyć funkcji `fromJSON()`.

Sposób użycia zależy od lokalizacji pliku `.json`.

```
dane<-fromJSON(file="filename.json")
dane<-fromJSON(file="URL do pliku json")
```

W obu przypadkach, obiekt `data` jest przechowywany jako lista. Do dalszej analizy możemy przekonwertować dane za pomocą funkcji `as.data.frame()`.

Zapisywanie danych do plików JSON

Musi być wykonany w dwóch etapach.

W pierwszym kroku musimy przygotować obiekt JSON, a w drugim zapisać go do pliku.

Aby utworzyć obiekt JSON, używamy funkcji `toJSON()`:

```
dataJSON<-toJSON(dane)
```

Następnie używamy funkcji `write()`.

```
write(dataJSON, "filename.json")
```

Statystyka i programowanie w R

III. Rozkłady prawdopodobieństwa w R

Wdrożone funkcje

R pozwala na pracę z dużą liczbą rozkładów prawdopodobieństwa.

Dla każdej z dystrybucji, z którymi pracuje R, zaimplementowane są cztery funkcje.

Ich nazwy składają się z nazwy głównej i przedrostka:

- `p` dla dystrybuanty,
- `d` dla funkcji gęstości prawdopodobieństwalub funkcji masy prawdopodobieństwa,
- `q` dla funkcji kwantylowej, odwrotna dystrybuanta,
- `r` zmienna losowa o określonym rozkładzie (generator wartości losowych).

Rozkład dyskretny

Prawdopodobieństwa są określane przez listę prawdopodobieństw dyskretnych wyników, zwaną funkcją prawdopodobieństwa.

Jeżeli zbiór wszystkich możliwych wartości dyskretnej zmiennej losowej X oznaczymy jako H , to możemy wprowadzić funkcję prawdopodobieństwa $p(x)$ według wzoru

$$p(x) = \mathbb{P}(X = x), \; x \in H. \tag{1}$$

Rozkład dyskretny

Wymieńmy niektóre z nich:

- rozkład Bernoulliego,
- rozkład dwumianowy,
- rozkład geometryczny pozycji,
- rozkład hipergeometryczny,
- rozkład dwumianowy ujemny,
- rozkład Poissona.

Rozkład Bernoulliego

Najprostszy rozkład prawdopodobieństwa

Jest to rozkład, który ma tylko dwie możliwe wartości.

Można ją interpretować jako zmienną wskaźnikową, która określa, czy zdarzenie losowe miało miejsce, czy też nie.

Niech A będzie zdarzeniem losowym, $\mathbb{P}(A) = p$ oraz zmienną losową $X = 1$ jeśli A zachodzi, a $X = 0$ w przeciwnym przypadku. Wówczas funkcja prawdopodobieństwa ma postać

$$p(x) = p^x(1-p)^{1-x} \text{ dla } x = 0 \text{ lub } 1$$

.

Jest on zaimplementowany w pakiecie `Rlab` jako `bern` z parametrem `prob`.

Rozkład Bernoulliego

Mamy cztery funkcje:

- `rbern(n,prob)`, gdzie Ž200+ to liczba obserwacji, a `prob` to prawdopodobieństwo zdarzenia losowego A (sukcesu w eksperymencie). Generuje on wektor `0` oraz `1` wybrany z rozkładu Bernoulliego z zadanym prawdopodobieństwem.
- `pbern(q, prob, lower.tail = TRUE, log.p = FALSE)`
- `dbern(x, prob, log = FALSE)`
- `qbern(p, prob, lower.tail = TRUE, log.p = FALSE)`

Rozkład dwumianowy

Rozkład dwumianowy jest rozkładem dyskretnym, który opisuje liczbę sukcesów w serii prób z dwoma możliwymi wynikami.

Formalnie, niech p oznacza prawdopodobieństwo sukcesu w jednej próbie, n - liczbę niezależnych prób, a x - liczbę sukcesów w sekwencji n niezależnych prób. Zmienna losowa X ma rozkład dwumianowy, jeżeli jej funkcja prawdopodobieństwa ma postać:

$$\mathbb{P}(X = x) = \binom{n}{x} p^x q^{n-x}, \quad x = 0, 1, 2, \ldots, n, \tag{2}$$

gdzie $q + p = 1$.

Rozkład dwumianowy

W programie zaimplementowane są 4 funkcje do pracy z rozkładem dwumianowym:

- `rbinom(n,prob)`, gdzie `n` to liczba obserwacji, `p` to prawdopodobieństwo sukcesu. Funkcja ta generuje n zmiennych losowych z zadanym prawdopodobieństwem.
- `pbinom(x,n k)`, gdzie `n` jest całkowitą liczbą prób, `p` jest prawdopodobieństwem sukcesu, `x` jest wartością, dla której ma być wyznaczone prawdopodobieństwo.
- `dbinom(x,n,p)`, gdzie `n` jest całkowitą liczbą prób, `p` jest prawdopodobieństwem sukcesu, `x` jest wartością, dla której ma być wyznaczone prawdopodobieństwo.
- `qbinom(prob,n,p)`, gdzie `prob` jest prawdopodobieństwem, `n` jest całkowitą liczbą prób, a `p` jest prawdopodobieństwem sukcesu na próbę. Funkcja ta służy do wyznaczania kwantyla n-tego, tzn. wyznacza k takie, że $P(X \leq k)$.

Rozkład hipergeometryczny

Rozkład hipergeometryczny jest dyskretnym rozkładem prawdopodobieństwa, który opisuje prawdopodobieństwo x sukcesów dla n selekcji bez zastąpienia ze skończonej populacji o rozmiarze N zawierającej dokładnie K obiektów z obserwowaną właściwością.

Oznaczmy przez N wielkość populacji, K liczbę obiektów o danej własności w populacji, n liczbę selekcji, a x liczbę zaobserwowanych sukcesów. Zmienna losowa X ma rozkład hipergeometryczny, jeśli jej funkcja prawdopodobieństwa ma postać

$$\mathbb{P}(X = x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}} \quad x = 0, 1, 2, \ldots, n. \tag{3}$$

Rozkład hipergeometryczny

Cztery funkcje do pracy z rozkładem hipergeometrycznym w R:

- `rhyper(N, m, n, k)` Ogólnie rzecz biorąc, odnosi się do funkcji generowania liczb losowych przy danych parametrach i wielkości próby,
  item `phyper(x, m, n, k)` definiuje funkcję rozkładu hipergeometrycznego,
- `dhyper(x, m, n, k)` definiuje funkcję prawdopodobieństwa rozkładu hipergeometrycznego,
- `qhyper(N, m, n, k)` jest funkcją kwantyla rozkładu hipergeometrycznego, która służy do wyznaczania ciągu prawdopodobieństw od 0 do 1.

Tutaj, `x` reprezentuje zbiór wartości, `m` wielkość populacji, `n` liczbę próbek, `k` liczbę elementów w populacji, a `N` wartości o rozkładzie hipergeometrycznym.

Rozkład Pascala

Rozkład Pascala jest dyskretnym rozkładem prawdopodobieństwa, który modeluje liczbę sukcesów w sekwencji niezależnych i identycznie rozłożonych prób Bernoulliego przed wystąpieniem pewnej (nielosowej) liczby porażek (oznaczonej n).

Oznaczając liczbę sukcesów x i prawdopodobieństwo sukcesu p, funkcja prawdopodobieństwa rozkładu dwumianowego ujemnego ma postać:

$$\mathbb{P}(X = x) = \binom{n+x-1}{n-1} p^x q^n, \quad x = 0, 1, 2, \ldots \tag{4}$$

gdzie $q + p = 1$, $p > 0$, $q > 0$.

Rozkład Pascala

Cztery funkcje do pracy z ujemnym rozkładem Pascalowym w R:

- `rnbinom(N,n,prob)`, gdzie Ž00+ to liczba prób, `N` to wielkość próby, `prob` to prawdopodobieństwo sukcesu. Funkcja ta generuje N zmiennych losowych z zadanym prawdopodobieństwem.
- `pnbinom(x, n, p)` służy do obliczania wartości dysrybuanty rozkładu Pascala. Tutaj `x` jest liczbą niepowodzeń przed `n`tym sukcesem, a `p` jest prawdopodobieństwem sukcesu.
- `dnbinom(x, n, p)` jest prawdopodobieństwem niepowodzenia `x` przed `n`-tym sukcesem (zauważ różnicę), gdy prawdopodobieństwo sukcesu jest równe `p`.
- `qnbinom(x, n, p)` służy do obliczania wartości funkcji kwantyla rozkładu pascala. Tutaj `x` jest wektorem wymaganych poziomów kwantyli, `n` jest całkowitą liczbą prób, a `p` jest prawdopodobieństwem sukcesu na próbę.

Rozkład Poissona

Rozkład Poissona jest dyskretnym rozkładem prawdopodobieństwa, który wyraża prawdopodobieństwo wystąpienia pewnej liczby zdarzeń w ustalonym przedziale czasowym lub przestrzennym, przy założeniu, że zdarzenia te występują w znanym stałym średnim tempie i niezależnie od czasu, jaki upłynął od ostatniego zdarzenia.

Jeżeli zmienna losowa X ma rozkład Poissona z parametrem $\lambda > 0$ (średnia liczba zdarzeń), to jej funkcja prawdopodobieństwa ma postać

$$\mathbb{P}(X = x) = \frac{\mathrm{e}^{-\lambda}\lambda^x}{x!}, \quad x = 0, 1, 2, \ldots \tag{5}$$

Rozkład Poissona

Cztery funkcje do pracy z rozkładem Poissona w R:

- `dpois(x,l)` oblicza wartość funkcji prawdopodobieństwa $\mathbb{P}(X = x)$ rozkładu Poissona z parametrem λ wprowadzonym jako argument `l`.
- `ppois(x,l)` oblicza dystrybuanty zmiennej losowej o rozkładzie Poissona. Wyznacza prawdopodobieństwo $\mathbb{P}(X \leq x)$, argument `l` jest parametrem rozkładu. Biorąc pod uwagę inny argument `lower.tail=FALSE`, otrzymujemy prawdopodobieństwo $\mathbb{P}(X > x)$.
- `rpois(k,l)` jest używany do generowania liczb losowych z danego rozkładu Poissona, `k` jest liczbą potrzebnych liczb losowych, a `l` jest parametrem rozkładu.
- `qpois(q,l)` służy do generowania kwantyli danego rozkładu Poissona, `q` jest wektorem potrzebnych poziomów kwantyli, a `l` jest parametrem rozkładu.

Rozkład ciągły

Jest to rozkład prawdopodobieństwa, który niesie zbiór nieprzeliczalny.

Rozkład ten jest jednoznacznie scharakteryzowany przez dystrybuantu

$$F(x) = \mathbb{P}(x \leq x) = \int_{-\infty}^{x} f(t)\,\mathrm{d}t.$$

Rozkłady ciągłe

Wymieńmy niektóre z nich:

- rozkład jednostajny,
- rozkład wykładniczy,
- rozkład normalny,
- rozkład Studenta t,
- rozkład Chi-kwadrat,
- rozkład Fishera F.

Wiele innych dystrybucji jest zaimplementowanych w R.

Rozkłady ciągłe

Wymieńmy niektóre z nich:

- rozkład jednostajny,
- rozkład wykładniczy,
- rozkład normalny,
- rozkład Studenta t,
- rozkład Chi-kwadrat,
- rozkład Fishera F.

Wiele innych dystrybucji jest zaimplementowanych w R.

Zajmiemy się, trzema "błękitnymi" dystrybucjami.

Rozkład jednostajny

Ciągły rozkład jednostajny opisuje eksperyment, w którym każdy wynik, który leży pomiędzy pewnymi granicami jest możliwy.

Dokładniej, zmienna losowa X ma rozkład jednostajny z parametrami a i b, $a < b$, jeśli jej funkcja gęstości ma postać:

$$f(x) =, \begin{cases} \frac{1}{b-a} & x \in \langle a; b \rangle \\ 0 & \text{indziej} \end{cases} \tag{6}$$

Rozkład jednostajny

Cztery funkcje do pracy z rozkładem jednostajnym w R:

- `dunif()`, który definiuje funkcję gęstości, której argumentami są wektor `x` oraz parametry `min` i `max` rozkładu,
- `punif()`, która definiuje dystrybuantu rozkładu, jej argumentami są wektor `x` oraz parametry `min` i `max` rozkładu,
- `qunif()`, która dostarcza funkcję kwantyla, której argumentami są kwantyle `q` oraz parametry `min` i `max` rozkładu,
- `runif()`, który generuje losowe wartości zmiennej, jego argumentami są wielkość próby `n` oraz parametry `min` i `max` rozkładu.

Rozkład wykładniczy

Rozkład wykładniczy opisuje czas oczekiwania pomiędzy zdarzeniami w procesie Poissona, tj. procesie, w którym zdarzenia występują w sposób ciągły i niezależny ze stałą średnią częstotliwością.

Zmienna losowa X ma rozkład wykładniczy z parametrem *lambda* > 0 (zwykle podawanym jako częstość), jeśli jej funkcja gęstości ma postać:

$$f(x) = \begin{cases} \lambda \mathrm{e}^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{7}$$

Jeżeli parametr λ jest interpretowany jako częstotliwość, to średni czas oczekiwania wynosi $\frac{1}{lambda}$.

Rozkład wykładniczy

Cztery funkcje do pracy z rozkładem wykładniczym w R:

- `dexp()`, która jest funkcją gęstości, której argumentami są wektor `x` i parametr `rate` rozkładu,
- `pexp()`, która jest dystrybuanta rozkładu, której argumentami są wektor `x` oraz parametr `rate` rozkładu,
- `qexp()`, określający funkcję kwantylową, której argumentami są kwantyle `q` oraz parametr `rate` rozkładu,
- `rexp()`, który generuje losowe wartości zmiennej, jego argumentami są wielkość próby `n` oraz parametr `rate` rozkładu.

Rozkład normalny

Rozkład normalny (lub Gaussowski) jest rodzajem ciągłego rozkładu prawdopodobieństwa, którego funkcja gęstości ma następującą postać.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{8}$$

Parametr μ jest średnią rozkładu (a także jego medianą i modusem), parametr σ jest jego odchyleniem standardowym.

Rozkład normalny jest ważny ze względu na centralne twierdzenie graniczne, które mówi, że populacja wszystkich możliwych próbek o rozmiarze n z populacji o średniej mu i wariancji σ^2 zbiega do rozkładu normalnego o średniej μ i $\frac{\sigma^2}{n}$, gdy n rośnie do nieskończoności.

rozkład normalny

Cztery funkcje do pracy z rozkładem normalnym w R:

- `dnormf()`, która jest funkcją gęstości, której argumentami są wektor `x` oraz parametry `mean` i `sd` rozkładu,
- `pnorm()`, który jest dystrybuanta rozkładu, której argumentami są wektor `x` oraz parametry `mean` i `sd` rozkładu
- `qnorm()`, który reprezentuje funkcję kwantylową, której argumentami są kwantyle `q` oraz parametry `mean` i `sd` rozkładu,
- `rnorm()`, który generuje wartości losowe zmiennej, jego argumentami są wielkość próby `n` oraz parametry `mean` i `sd` rozkładu.

Statystyka i programowanie w R

IV. Programowanie w R

Funkcje

Prawie wszystkie działania w R są wykonywane za pomocą funkcji.

Zaimplementowany jest bogaty zestaw funkcji wbudowanych.

Użytkownik może zdefiniować dodatkowe funkcje

Wbudowane funkcje można podzielić na

- Funkcje matematyczne,
- Funkcje łańcuchów elementów,
- Wyspecjalizowane funkcje statystyczne i probabilistyczne,
- Inne przydatne funkcje.

Funkcje matematyczne

O niektórych z nich wspomnieliśmy już w lekcji 1.

Tutaj podamy kilka dodatkowych szczegółów

Funkcja logarytmiczna `log()` oblicza logarytm naturalny jako wartość domyślną.

Aby otrzymać logarytm o dowolnej podstawie, należy zadeklarować argument `base` funkcji `log()`.

```
1 > log(4)
2 [1] 1.386294
3 > log(4,base=2)
4 [1] 2
```

Funkcje tekstowe

Funkcja nchar()+ określa rozmiar każdego elementu wektora znaków.

```
1  > z<-c("yellow","black","white")
2  > nchar(z)
3  [1] 6 5 5
4  > str<-"This␣is␣a␣long␣string"
5  > nchar(str)
6  [1] 21
```

Elementarne funkcje statystyczne

- `mean()` Średnia próbna.
- `median()` Mediana próby.
- `sd()` Odchylenie standardowe.
- `var()` Wariancja próby .
- `mad()` Odchylenie bezwzględne mediany.
- `quantile()` Kwantyle próbky, kwartyle są domyślne.
- `range()` Zakres wartości.
- `sum()` Suma elementów wektora.
- `min()` Minimum.
- `max()` Maximum.

Elementarne funkcje statystyczne – `mean()` argumenty opcjonalne

`trim`, który określa procent najwyższych i najniższych wartości, które są pomijane w obliczeniach, zwracając w ten sposób przyciętą średnią.

Drugi opcjonalny argument `na.rm` jest wartością logiczną, która określa, czy usunąć wartości `NA` przed kontynuowaniem obliczeń.

```
1  > x<-c(1,3,5,10,12)
2  > mean(x)
3  [1] 6.2
4  > mean(x,trim=0,2)
5  [1] 6
```

```
1  > x<-c(1,5,2,12,NA,3,6)
2  > mean(x)
3  [1] NA
4  > mean(x,na.rm=TRUE)
5  [1] 4.833333
6  > mean(x,na.rm=TRUE,trim=0.17)
7  [1] 4
```

Elementarne funkcje statystyczne – `mad()`

Bezwzględne odchylenie mediany jest solidną miarą zmienności jednoczynnikowej próbki danych ilościowych.

Dla próbki $X_1, \ldots, X_n$ jest ona określona wzorem:

$$\mathrm{MAD}(X) = \mathrm{median}\{|X_i - \overline{X}|\}$$

```
1  > mad(delay)
2  [1] 13.3434
```

Użyteczne funkcje – seq()

Funkcja `seq()` generuje ciąg liczb zaczynający się od `from` i kończący się `to`. Ostatni argument `by` określa krok sekwencji.

```
> seq(10)
 [1]  1  2  3  4  5  6  7  8  9 10
> seq(5,15)
 [1]  5  6  7  8  9 10 11 12 13 14 15
> seq(5,15,2)
[1]  5  7  9 11 13 15
```

Użyteczne funkcje – rep()

Funkcja rep() przyjmuje dwa argumenty, wektor x do powtórzenia oraz liczbę cykli powtórzeń n.

```
> rep(1,10)
 [1] 1 1 1 1 1 1 1 1 1 1
> rep(c(1,3),4)
[1] 1 3 1 3 1 3 1 3
> rep("hello",3)
[1] "hello" "hello" "hello"
```

Użyteczne funkcje – `sort()` i `order()`

.

Funkcje `sort()` i `order()` związane są z porządkowaniem elementów wektora `x`.

`sort()` dostarcza posortowanych rosnąco wartości, podczas gdy `order()` dostarcza indeksów posortowanych komponentów w oryginalnym wektorze.

```
1  > x<-c(5,2,10,3,7,8)
2  > sort(x)
3  [1]  2  3  5  7  8 10
4  > order(x)
5  [1]  2  4  1  5  6  3
```

Użyteczne funkcje – `rev()`

Daje wektor `x` w odwrotnej kolejności elementów

```
> rev(x)
[1]  8  7  3 10  2  5
> rev(sort(x))
[1] 10  8  7  5  3  2
```

Komendy warunkowe – `if`

`if()` wykonuje operacje na podstawie prostego warunku

```
if (warunek) {polecenie do wykonania, jeśli warunek jest spełniony}
```

Więcej niż jedno stwierdzenie musi być w nawiasie

```
1  > x<-5
2  > if(x%%2){print("Odd␣number")}
3  [1] "Odd␣number"
4  > x<-6
5  > if(x%%2){print("Odd␣number")}
6  >
```

Komendy warunkowe – switch

`switch()` testuje wyrażenie względem elementów listy. Każda wartość na liście jest nazywana case.

Składnia funkcji `switch()`:

```
switch (wyrażenie, lista)
```

```
1  > x<-10
2  > switch(x%%2+1, "even", "odd")
3  [1] "even"
4  > x<-9
5  > switch(x%%2+1, "even", "odd")
6  [1] "odd"
```

Komendy warunkowe – switch

Jeśli wyrażenie jest łańcuchem znaków, switch() zwraca wartość opartą na nazwie elementu.

```
> x <- "a"
> switch(x, "a"="apple", "b"="banana", "c"="cherry")
[1] "apple"
> x <- "c"
> switch(x, "a"="apple", "b"="banana", "c"="cherry")
[1] "cherry"
```

Pętla – `for`

Pętla \`for` pozwala nam na ustaloną liczbę powtórzeń danej instrukcji lub bloku instrukcji.

Ogólna składnia pętli `for` jest następująca:

```
for (val in sequence)
{
polecenie
}
```

gdzie `sequence` jest wektorem, a `val` przyjmuje każdą z wartości sekwencji podczas pętli.

Funkcja zdefiniowana przez użytkownika

Ogólna struktura funkcji jest następująca

```
myfunction_name <- function(arg1, arg2, ... ){
polecenia
return(obiekt)
}
```

Funkcja zdefiniowana przez użytkownika

Poszczególnymi składnikami funkcji są:

- **Nazwa funkcji**, który jest rzeczywistą nazwą funkcji. Jest on przechowywany w środowisku R jako obiekt o tej nazwie.
- **Argumenty**, które są symbolami wieloznacznymi. Kiedy wywołujemy funkcję, przekazujemy wartości argumentów. Argumenty są opcjonalne, to znaczy, że funkcja nie musi zawierać żadnych argumentów. Argumenty mogą mieć również wartości domyślne.
- **Ciało funkcji**, który zawiera zestaw poleceń określających działanie funkcji. Ciało funkcji znajduje się wewnątrz nawiasów złożonych.
- **Wartość zwracana**, który jest ostatnim obliczanym wyrażeniem w treści funkcji.

Uruchamianie skryptów w R

Skrypt R jest po prostu plikiem tekstowym zawierającym (prawie) te same polecenia, które wpisałbyś pisząc

Można go utworzyć w dowolnym prostym edytorze tekstu i zapisać z rozszerzeniem `.R`.

Istnieją zasadniczo dwa polecenia do uruchomienia skryptu w Linuksie.

```
Rscript filename.R
```

co jest preferowane. Starsze polecenie to

```
R CMD BATCH filename.R
```

Statystyka i programowanie w R

V. Podstawy grafiki w R

Wykresy punktowe

Wykres punktowy (także korelacyjny) przedstawia wartości dwóch różnych zmiennych liczbowych.

Położenie każdego punktu odpowiada poszczególnym wartościom danych na osi poziomej i pionowej.

Najczęstsze zastosowania i sposoby wykorzystania wykresów punktowych to:

1. Demonstracja związku między dwiema zmiennymi.
2. Identyfikacja związków korelacyjnych.
3. Identyfikacja modeli danych.

Wykresy punktowe

Tworzymy je po prostu za pomocą funkcji `plot()`.

W najprostszym ujęciu funkcja przyjmuje dwa argumenty `x` oraz `y`.

Zmienne te są wektorami, które zawierają wartości, które chcemy wykreślić.

Długość wektorów musi być taka sama.

Jak zapisać obraz

Alternatywnie, możemy przekierować wyjście z ekranu do pliku.

Możemy użyć funkcji

`pdf()`	Vector `pdf` format, najlepszy wybór, gdy jest używany z `pdflatex`.
`svg()`	Format wektorowy `svg`; łatwy do zmiany rozmiaru.
`postscript()`	Format wektorowy postscriptowy, łatwa zmiana rozmiaru.
`png()`	Format bitmap o wysokiej rozdzielczości, nie może być zmieniany bez utraty rozmiaru.
`jpeg()`	Skompresowany format bitmapy, bez bezstratnej zmiany rozmiaru.
`bmp()`	Format bitmap o wysokiej rozdzielczości, nie zmienia rozmiaru bezstratnie.
`tiff()`	format bitmap o wysokiej rozdzielczości, nie zmienia rozmiaru bezstratnie.

Opcje przechowywania wykresów

`filename`	Nazwa zapisywanego pliku, w razie potrzeby z pełną ścieżką.
`width`	Szerokość wynikowego wykresu, wartość domyślna 7 in.
`height`	Wysokość wynikowego wykresu, wartość domyślna 7 in.
`res`	rozdzielczość obrazu, ważne dla formatów bitmapowych, domyślnie 72 dpi.
`units`	Jednostki miary.
`bg`	Kolor tła.
`fg`	Kolor pierwszego planu.
`family`	Używana czcionka (domyślnie Helvetica).

Modyfikacja wykresu – punkty znacznikowe

Znacznik punktu jest podawany przez wartość argumentu `pch` funkcji `plot()`.

Możliwe wartości

□	pch=0	○	pch=1	△	pch=2	+	pch=3	×	pch=4
◇	pch=5	▽	pch=6	⊠	pch=7	✳	pch=8	⊕	pch=9
⊕	pch=10	✡	pch=11	⊞	pch=12	⊗	pch=13	◪	pch=14
■	pch=15	●	pch=16	▲	pch=17	◆	pch=18	●	pch=19
•	pch=20	○	pch=21	□	pch=22	◇	pch=23	△	pch=24

Modyfikacja wykresu – typ łącza

Typ linii punktowej ustawia się za pomocą argumentu `type` funkcji `plot()`.

Możliwe wartości

`p`	Wykres punktowy, wartość domyślna.
`l`	Linia ciągła.
`b`	Linia ciągła z punktami.
`c`	Części linii ciągłych z pominiętymi punktami.
`o`	Części linii ciągłych, punkty przerysowane.
`h`	Wykres podobny do histogramu.
`s`	Schemat schodów.

Modyfikacja wykresu – styl łącza

Styl linii jest ustawiany za pomocą argumentu `lty` funkcji `plot()`.

Możliwe wartości

1	Pogrubiona linia (domyślnie).	2	Pogrubiona linia.
3	Kreska z kropkami.	4	Kreski i przecinki.
5	Długie kreski.	6	D Długie i krótkie podwójne linie przerywane.

Szerokość linii ustawia się za pomocą argumentu `lwd` funkcji `plot()`.

Modyfikacja grafu - kolorowanie

Kolory możemy modyfikować za pomocą

- przez nazwy koloru elementu, na przykład `col=red`.
- numer koloru elementu, na przykład `col=636`
- według kodu szesnastkowego (w trybie RGB), na przykład `col="#FFCC00"`.

Listę dostępnych kolorów można uzyskać na wyjściu funkcji `colors()`.

Modyfikacja grafu - kolorowanie

Inne opcje kolorystyczne to

`col.axis`	Kolor etykiet osi.	`col.lab`	Kolor oznaczeń osi.
`col.main`	Kolor nagłówka głównego	`col.sub`	Kolor podtytułu
`bg`	Kolor wypełnienia znaków.	`fg`	Kolor tła wykresu.

Modyfikacja grafu - kolorowanie

Kolory jako wektory

Wartość argumentu `col` może być ustawiona jako wektor.

Kolory z wektora są następnie regularnie przeplatane.

Możemy również użyć funkcji `rainbow()` z predefiniowaną sekwencją kolorów.

Edytowanie ramki – tytuły i napisy

Podstawowe funkcje rysowania w R zawierają argument `main`, który umożliwia dodanie tytułu do wykresu.

Możemy również użyć argumentu `sub`, aby dodać podtytuł, który zostanie umieszczony pod wykresem.

Alternatywnym sposobem na dodanie tytułu i podtytułu do wykresu jest użycie funkcji `title()`.

Edycja wykresu – dodaj tekst do wykresu

Do narysowanego wykresu możemy dodać dowolny tekst za pomocą funkcji `text()` oraz `mtext()`.

Funkcja `text()` umieszcza podany tekst w dowolnym miejscu obszaru rysowania, funkcja `mtext()` umieszcza tekst na krawędziach.

Funkcja `text()` przyjmuje dwa dodatkowe argumenty:

- `location` definiuje współrzędne `x` i `y`, w których zostanie umieszczony tekst. Współrzędne muszą być podane jako dwa pierwsze argumenty funkcji.
- `pos` określa położenie względem aktualnej pozycji, 1=w dół, 2=w lewo, 3=w górę i 4=w prawo. Zdefiniowanie pozycji jako `locator(1)` pozwala na pozycjonowanie tekstu za pomocą myszy.

Edycja wykresu – wyrównanie osi

Aby usunąć ramkę wykresu, ustaw argument funkcji rysowania `axes=FALSE`.

Dodaj nowe osie za pomocą funkcji `axes()`.

Argument funkcji `osie()` określa stronę wykresu, do której zostanie dodana oś.

Jak zwykle, liczby określają boki: 1=dolny, 2=lewy, 3=górny i 4=prawy.

Spróbujmy

```
1 > plot(sort(x),y[order(x)],pch=17,type="b",col=30,axes=FALSE)
2 > axis(1)
3 > axis(2)
```

Dalsze ustawienia osi

Możemy również:

- Określ liczbę znaczników o podanych wartościach początkowych i końcowych,
- aby dostosować długość i orientację znaków,
- obracać etykiety znaczników,
- Dostosuj opisy tagów,
- Usuń tagi,
- dodać mniejsze tagi za pomocą `Hmisc` .

Dalsze ustawienia osi – długość i orientacja markerów

.

Argument `tck` pozwala na ustawienie długości i orientacji znaczników podziału.

Jego dodatnia wartość orientuje znaki wewnątrz obszaru rysowania, podczas gdy wartości ujemne orientują znaki poza obszarem rysowania. Im większa jest wartość bezwzględna, tym dłuższe są znaki. Domyślną wartością jest `tck=-0.05`.

Obrót jest włączany przez argument `las`, który może przyjąć jedną z czterech wartości:

- `las=0` etykiety są równoległe do osi (domyślnie),
- `las=1` wszystkie etykiety są poziome,
- `las=2` etykiety są prostopadłe do osi,
- `las=3` wszystkie etykiety są pionowe.

Zakres osi i dostosowanie do potrzeb użytkownika

Zakres wartości dla osi można zdefiniować za pomocą opcjonalnych argumentów `xlim` i `ylim` funkcji `plot()`.

Granice są określone jako wektory postaci `c(start,end)`.

Możemy również przekształcić osie do skali logarytmicznej, ustawiając argument `log` na wartość osi, którą planujemy dopasować.

`log="x "` ustawia skalę logarytmiczną na osi `x`,
`log="y "` ustawia skalę logarytmiczną na osi `y`, oraz
`log="xy "` przekształca obie osie na skalę logarytmiczną.

Podwójna oś pionowa

Príklad.
Chcemy nanieść na jeden wykres dwie charakterystyki stanu zdrowia pacjentów, temperaturę i ciśnienie krwi.

Podwójna oś pionowa

Príklad.
Chcemy nanieść na jeden wykres dwie charakterystyki stanu zdrowia pacjentów, temperaturę i ciśnienie krwi.

W zmiennych `y` i `z` mamy dane dotyczące 100 pacjentów, a zmienna `x` zawiera ciąg identyfikatorów pacjentów, liczb od 1 do 100.

Krzywe

Jedną z wielu przydatnych funkcji w R jest `curve()`.

Jest to poręczna, mała funkcja, która pozwala na wykreślanie krzywych, takich jak wykresy funkcji.

Funkcja `curve()` przyjmuje jako pierwszy argument wyrażenie w składni R.

Na przykład,

```
curve(x^2)
curve(x^2,xlim=c(-2,2),col="red",lwd=2)
```

Wyświetl dwie lub więcej krzywych na jednym wykresie

Używamy funkcji `curve()` z argumentem `add=TRUE`.

Na przykład

```
curve(x^2)
curve(sqrt(x),col="red",lwd=2,add=TRUE)
```

Dodawanie legendy

Funkcja legenda()+ pozwala na dodanie legendy do wykresów.

Kilka argumentów:

- verb+x,y+ pozycja w obszarze rysowania określona przez współrzędne na wykresie,
- `legend` wektor ciągów znaków dla opisu legendy,
- `col` wektor kolorów użytych na wykresie,
- `pch` wektor kształtów markerów użytych w wykresie,
- `lty` wektor typów linii użytych w wykresie,
- `ncol` liczba kolumn używanych w legendzie, wartość domyślna to jedna kolumna.

Wykresy słupkowe

Wykres słupkowy wyświetla dane kategoryczne za pomocą prostokątnych kolumn, których wysokość lub długość jest proporcjonalna do reprezentowanych przez nie wartości.
R używa funkcji do tworzenia wykresów słupkowych

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

Parametry wykorzystywane w funkcji są następujące:

- `H` jest wektorem lub macierzą zawierającą wartości liczbowe używane w wykresie słupkowym,
- `xlab` to etykieta osi x,
- `ylab` jest etykietą osi y,
- `title` to tytuł wykresu słupkowego,
- `names.arg` to wektor etykiet, które pojawiają się pod każdą kolumną,
- `col` służy do przypisywania kolorów kolumnom na wykresie.

Wykresy kolumnowe – kolorowanie i etykiety kolumn

Używamy parametru `names.arg` wykresu słupkowego, aby przypisać nazwy do kolumn.

Następnie określamy wartości parametrów

- `xlab` a `ylab` dla nazw osi,
- `col` a `border` do kolorowania pasków, a także
- `main` aby zdefiniować tytuł wykresu.

Jest to funkcja podobna do funkcji `plot()`.

Histogramy

Histogram to przedstawienie przybliżonego rozkładu danych liczbowych.

Pokazuje częstości występowania wartości podzielonych na przedziały.

Histogram jest podobny do wykresu słupkowego, z tą różnicą, że grupuje wartości w ciągłe przedziały.

Histogramy dają przybliżony obraz gęstości rozkładu danych.

Histogramy

Histogram można utworzyć za pomocą funkcji `hist()` w programie R.

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

Parametry wykorzystywane w funkcji są następujące:

- `data` to wektor zawierający wartości numeryczne używane w histogramie,
- `main` to tytuł wykresu,
- `col` służy do ustawiania koloru pasków,
- `border` służy do ustawiania koloru obramowania każdej kolumny,
- `xlab` określa opis osi x,
- `xlim` określa zakres wartości na osi x,
- `ylim` określa zakres wartości na osi y,
- `breaks` służy do określenia szerokości każdej kolumny.

Diagramy kołowe

Diagram kołowy jest wykresem dla jednej zmiennej kategorycznej i jest alternatywą dla wykresu słupkowego.

Diagram kołowy (lub wykres/diagram tortowy) to wykres w kształcie koła podzielony na części, które pokazują zależności między zmiennymi.

W wykresie kołowym długość łuku każdego segmentu (a zatem jego kąt środkowy i powierzchnia) jest proporcjonalna do wielkości, którą reprezentuje.

Diagramy kołowe

Podstawowa składnia do tworzenia wykresu kołowego w R jest następująca:

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

Znaczenie argumentów:

- `data` jest wektorem zawierającym wartości liczbowe użyte w wykresie kołowym,
- `labels`Do opisania części używa się przysłówka + znaków+,
- `radius` określa promień diagramu kołowego (wartość pomiędzy -1 a $+1$),
- verb+main+
- wskazuje tytuł wykresu,
- `col` wskazuje paletę kolorów,
- `clockwise` jest to wartość logiczna określająca, czy plasterki są rysowane zgodnie czy przeciwnie do ruchu wskazówek zegara.

Diagram kołowy – regulacja koloru

Aby zmienić kolory na wykresie, używamy funkcji `rainbow()`, która definiuje paletę kolorów. Jego argumenty są następujące:

- `n` liczba kolorów (*geq*1), które mają być w palecie,
- `s,v` "nasycenie koloru" i "wartośći", aby dodać opisy do kolorów
- `start` (zmodyfikowany) odcień w $\langle 0; 1\rangle$, od którego zaczyna się wybrana tęcza,
- `end` (dostosowany) odcień w miejscu, gdzie kończy się tęcza,
- `gamma` Korekcja gamma dla każdego koloru (r,g,b)+ w przestrzeni RGB (ze wszystkimi wartościami w przestrzeni $\langle 0; 1\rangle$), kolor wynikowy odpowiada $(r^\gamma, g^\gamma, b^\gamma)$,
- `alpha`transparency, liczba w postaci $\langle 0; 1\rangle$, (0 oznacza przezroczyste, a 1 oznacza nieprzezroczyste).

Diagram kołowy - użycie rainbow()

```
1 description<-paste(labels,"\n",data,sep="")
2 pie(data,description,main="Monthly␣expenses",
3 col=rainbow(length(data)))
```

Uwaga

Zmieniliśmy również etykiety. Do ich nazw dodaliśmy wartości liczbowe.

Wachlarzowy diagram

Użyteczną alternatywą dla wykresów kołowych jest funkcja `fan.plot()` zdefiniowana w pakiecie `plotrix`.

Umożliwia wizualne porównanie sektorów wykresu kołowego.

Wykres wachlarzowy może być dostosowany do potrzeb użytkownika poprzez podanie dodatkowych argumentów:

- `max.span` Kąt maksymalnej rozpiętości sektora w radianach. Domyślnie, `data` jest skalowane tak, aby suma była równa $2pi$.
- `ticks` liczba łatek, które pojawiłyby się, gdyby sektory znajdowały się na wykresie kołowym. Domyślnie nie ma skrzynek.

Wykres pudełkowy

W statystyce opisowej, wykres pudełkowy lub boxplot jest rodzajem wykresu, który jest często używany do objaśniania analizy danych.

Wykresy pudełkowe pokazują rozrzut danych liczbowych i skośność poprzez wyświetlanie kwartyli (lub percentyli) i średnich danych.

Jest on również przydatny do porównywania rozkładu danych w różnych zestawach danych poprzez rysowanie wykresu pudełkowego dla każdego z nich.

Wykres pudełkowy

Wykres pudełkowy składa się z dwóch części, pudełka i zestawu wiskerów.

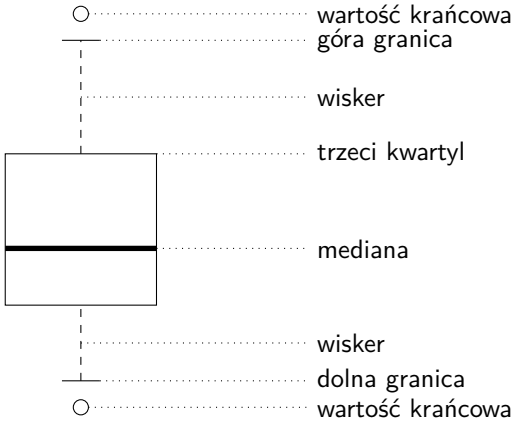
Ramka pokazuje zakres od dolnego kwartyla do górnego kwartyla, a pozioma linia narysowana wewnątrz ramki wskazuje medianę.

Wiskery może kilka wartości odstających między obserwowanymi danymi według jednej z alternatyw:

- Podać minimum i maksimum wszystkich danych,
- Pozycja z jednym odchyleniem standardowym powyżej i poniżej średniej danych,
- Pozycja 9. percentyl i 91. percentyl,
- 2. percentyl i 98. percentyl.

Wszystkie dane, które nie mieszczą się w wachlarzach, powinny być wykreślone jako wartości odstające, oznaczone kropką, małym kółkiem lub gwiazdką, ale czasami jest to pomijane.

Wykres pudełkowy



Wykresy Q-Q

Wykres kwantylowy (w skrócie wykres Q-Q) jest narzędziem graficznym, które pomaga nam ocenić, czy zbiór danych wiarygodnie pasuje do pewnego teoretycznego rozkładu, takiego jak rozkład normalny lub wykładniczy.

Na przykład, jeśli przeprowadzamy analizę statystyczną, która zakłada, że nasza zmienna zależna ma rozkład normalny, możemy użyć normalnego wykresu Q-Q, aby sprawdzić to założenie.

Jest to tylko wizualna kontrola, a nie dokładny dowód, ale pozwala nam zobaczyć na pierwszy rzut oka, czy nasze założenie jest wiarygodne, a jeśli nie, to w jaki sposób założenie jest naruszone i które wartości danych przyczyniają się do naruszenia.

Wykresy Q-Q

Wykres Q-Q jest zasadniczo wykresem punktowym utworzonym przez wykreślenie dwóch zestawów kwantyli względem siebie.

Jeśli oba zestawy kwantyli pochodzą z tego samego rozkładu, punkty leżą w przybliżeniu na linii prostej.

Wykresy Q-Q biorą nasze próbki, sortują je w porządku rosnącym, a następnie wykreślają je względem kwantyli proponowanego rozkładu teoretycznego.

Liczba kwantyli jest tak dobrana, aby odpowiadała wielkości naszej próby.

Wielość wykresów w jednym obrazie

W R możemy łączyć wykresy za pomocą parametrów graficznych `mfrow` i `mfcol`.

Wystarczy podać wektor określający liczbę wierszy i liczbę kolumn, które zamierzamy utworzyć.

Decyzja o tym, który parametr wykresu zastosować zależy od tego, jak chcemy ułożyć wykresy:

- `mfrow` wykresy będą ułożone w rzędach,
- `mfcol` wykresy będą ułożone według kolumn.

To ustawienie jest używane jako argument funkcji `par()`, która modyfikuje parametry urządzenia graficznego.

Statystyka i programowanie w R

VI. Charakterystyky doboru próby

Średnia

Średnia podanego zestawu wartości $x_1, x_2, \ldots, x_n$ jest określona zależnością:

$$\overline{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = \frac{\sum_{i=1}^{n} x_i}{n}. \tag{9}$$

W R jest zaimplementowana jako funkcja `mean()`.

Jego użycie jest bardzo proste.

Średnia

Bazę danych makroekonomicznych można znaleźć na stronie Słowackiego Banku Centralnego https://www.nbs.sk/en/monetary-policy/macroeconomic-database.

Możemy pobrać np. plik `.csv`, który zawiera liczbę rejestracji nowych samochodów osobowych w wybranym okresie.

Ten plik jest niejawnie zapisywany jako `makrostat.csv`.

Ze względu na użycie przecinka jako separatora dziesiętnego dane odczytujemy za pomocą funkcji `read.csv2()`.

Średnia

Musimy użyć funkcji `as.numeric()` do obliczenia średniej, ponieważ `cars[1,]` podaje wartości w formacie listy.

Aby więc uzyskać średnią wartość nowych samochodów osobowych rejestrowanych miesięcznie (w tysiącach) w latach 2017-18, użyjemy kodu:

```
1 cars<-read.csv2("macrostat.csv",header=FALSE,sep=";")
2 mean(as.numeric(cars[1,]))
3 [1] 8.090125
```

Średnia

Często zdarza się, że interesujące nas wartości o charakterze statystycznym układają się w ciąg bezwzględnych obfitości.

W tym przypadku modyfikujemy relację (9), aby obliczyć średnią wartość dla kształtu:

$$\overline{x} = \frac{x_1 \cdot n_1 + x_2 \cdot n_2 + \cdots + x_k \cdot n_k}{n_1 + n_2 + \cdots + n_k} = \frac{\sum_{i=1}^{k} x_i \cdot n_i}{\sum_{i=1}^{k} n_i}. \tag{10}$$

gdzie x_i oznaczają wartości zmiennej, a n_i ich bezwzględne liczby.

Średnia

W takim przypadku musimy zdefiniować funkcję niestandardową, aby obliczyć wartość średnią.

Wprowadzamy dwa wektory jako wartości wejściowe. Pierwszy wektor zawiera wartości uzyskane przez zmienną losową, a drugi to wektor ich częstości.

Przed wykonaniem obliczeń według relacji (10) należy sprawdzić, czy oba wektory mają taką samą długość.

Mediana, wartość środkowa

Medianę możemy scharakteryzować jako wartość środkową, jeśli obserwowane wartości uszeregujemy od najmniejszej do największej.

Formalnie możemy powiedzieć, że prawdopodobieństwo, że wartość zmiennej jest większa od mediany, jest równe prawdopodobieństwu, że jej wartość jest mniejsza od mediany, a zatem prawdopodobieństwo to jest równe $\frac{1}{2}$.

Jeśli mamy próbkę wartości $x_1, x_2, \ldots, x_n$, to mediana $\tilde{x}$ dana jest przez relacje

$$\tilde{x} = \begin{cases} \left(\frac{n+1}{2}\right) \text{ta wartość porządkowa} & n \text{ jest nieparzystą} \\ \frac{1}{2}\left(\left(\frac{n}{2}\right) \text{ta} + \left(\frac{n}{2}\right) \text{ta wartość porządkowa}\right) & n \text{ jest parzystą} \end{cases} \tag{11}$$

Kwantyle

Funkcja `quantile()` jest zaimplementowana w języku R w celu znalezienia kwantyli. Bez określania parametrów opcjonalnych wynikiem jest minimum próby, pierwszy kwartyl, mediana, trzeci kwartyl i maksimum próby.

Możemy to zilustrować danymi na temat COVID-19, pobranymi z oficjalnej strony słowackiego rządu https://korona.gov.sk.

```
1  data<-read.csv("https://mapa.covid.chat/export/csv",
2     header=T,sep=";")
3  > quantile(data[,4])
4     0%    25%    50%    75%   100%
5      0     30    232   1737  15278
6  >
```

Rozstęp

Rozstęp zależy tylko od dwóch wartości w próbce — największej i najmniejszej wartości.

Definiuje się ją jako różnicę

$$R = \max\{x_1, \ldots, x_n\} - \min\{x_1, \ldots, x_n\} \tag{12}$$

Jest to przydatne do szybkiej orientacji w zakresie zmienności danej próbki.

Rozstęp

Dane wyjściowe funkcji `range()` w środowisku języka R to zakres wariacji.

Jego wyjście to dwie wartości - największa i najmniejsza wartość w próbce.

Aby wyrazić zakres odchylenia jako z definicji jedną wartość (12), używamy funkcji `max()` i `min()`.

Rozstęp międzykwartylowy

Rozstęp międzykwartylowy jest miarą rozproszenia statystycznego.

Definiuje się go jako różnicę między górnym i dolnym kwartylem próby.

Przypisując te kwartyle jako Q_3 i Q_1 , możemy wyrazić rozstęp międzykwartylowy IQR jako

$$IQR = Q_3 - Q_1. \tag{13}$$

Średnie odchylenie bezwzględne

Odchylenie bezwzględne średniej próbki *MAD* to średnia odległość między każdym punktem danych a średnią.

Dla próbki $x_1, \ldots, x_n$ jest zdefiniowany przez formułę

$$MAD = \frac{1}{n} \sum_{i=1}^{n} |x_i - \overline{x}| . \tag{14}$$

Wariancja i odchylenie standardowe

Wariancja jest najpopularniejszą i najczęściej stosowaną miarą zmienności próby.

Jeśli mamy próbkę $x_1, \ldots, x_n$, definiujemy wariancję próbki s^2 przez

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2 . \tag{15}$$

Wariancja i odchylenie standardowe

W raziedane ułożone w ciąg obfitości bezwzględnych $n_1, \ldots, n_k$, $\sum_{i=1}^{k} n_i = n$, poszczególne wartości $x_1, \ldots, x_k$, dopasowujemy wzór (15) do postaci

$$s^2 = \frac{1}{n} \sum_{i=1}^{k} n_i \left(x_i - \overline{x}\right)^2 . \tag{16}$$

Wariancja i odchylenie standardowe

Funkcji `var()` i `sd()` należy używać ostrożnie.

Ich wynikiem są obiektywne oszacowania wariancji i odchylenia standardowego całej populacji.

Jeśli chcemy obliczyć wariancję próbki zgodnie z relacją (15), musimy zdefiniować własną funkcję, którą zilustrujemy w poniższym kodzie źródłowym.

Wariancja i odchylenie standardowe

```
> variance<-function(x) sum((x-mean(x))^2)/length(x)
> stdev<-function(x) sqrt(variance(x))
> variance(x)
[1] 14.40816
> stdev(x)
[1] 3.795809
> var(x) # porownaj wyniki
[1] 16.80952
> sd(x)
[1] 4.099942
```

Współczynnik zmienności

Współczynnik zmienności jest statystyczną miarą względnego rozrzutu danych w stosunku do wartości średniej.

Współczynnik zmienności CV jest zdefiniowany jako stosunek odchylenia standardowego s do średniej $\overline{x}$

$$CV = \frac{s}{\overline{x}}. \tag{17}$$

Współczynnik zmienności jest często wyrażany w procentach.

Współczynnik zmienności

Współczynnik zmienności nie jest zaimplementowany jako funkcja w języku R

Możemy to obliczyć za pomocą znanych funkcji lub zdefiniować własną funkcję

```
1 > cv<-function(x) variance(x)/mean(x) * 100
2 > cv(x)
3 [1] 157.5893
```

Skośność

Skośność jest miarą asymetrii rozkładu lub zbioru danych.

Skośność γ_1 definiujemy jako

$$\gamma_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^3}{s^3}. \tag{18}$$

Skośność

W zależności od wartości γ_1 istnieją 3 rodzaje skośności.

Jeśli $\gamma_1 > 0$, mówimy o skośności właściwej. Oznacza to, że większość danych jest mniejsza od średniej.

Z drugiej strony, gdy $\gamma_1 < 0$ mówimy o lewostronnej skośności. W tym przypadku większość wartości w zbiorze danych jest większa niż średnia.

I wreszcie zerowa skośność $\gamma_1 = 0$ reprezentuje symetryczny rozkład danych.

Ostrość

Ostrość mierzy punktowość piku w rozkładzie danych.

ostrość γ_2 definiujemy jako

$$\gamma_2 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^4}{s^4}. \tag{19}$$

Ostrość

Podobnie jak w przypadku skośności, istnieją trzy rodzaje ostrości.

Jeśli $\gamma_2 = 3$, mówimy o normalnym wzroście rozkładu. Wartość γ_2 jest porównywana z wartością 3, ponieważ kurtoza rozkładu normalnego jest równa 3. Porównujemy pik próbki z krzywą Gaussa.

W przypadku $\gamma_2 < 3$ rozkład jest bardziej płaski niż rozkład normalny.

W przeciwnym razie, gdy $\gamma_2 > 3$, analizowany rozkład ma ostrzejszy pik niż rozkład normalny.

Statystyka i programowanie w R

VII. Szacunki parametrów

Szacunki parametrów

Jednym z celów analizy statystycznej jest oszacowanie parametrów pierwotnego rozkładu, z którego pochodzi losowy wybór.

Wyróżniamy dwa rodzaje szacunków:

- szacunki punktowe, które zapewniają oszacowanie dokładnych wartości parametrów,
- przedziały ufności, które reprezentują przedziały zawierające wartość parametru z określonym prawdopodobieństwem (podanym jako poziom ufności).

Szacunki punktowe

Aby oszacować wartość parametru, staramy się wybrać taką charakterystykę, która najlepiej przybliża parametr Θ.

Jakość oszacowania zależy od jego właściwości:

- Bezstronne oszacowanie parametru θ to takie oszacowanie T, że zachodzi równość $\mathbb{E}(T) = \theta$.
- Spójne oszacowanie może charakteryzować się rosnącą precyzją wraz ze wzrostem wielkości próby. Formalnie możemy napisać $\lim_{n\to\infty} T_n = \Theta$, gdzie indeksy dolne reprezentują wielkość próby użytej do oszacowania.
- Efektywne oszacowanie może być interpretowane jako najlepsze możliwe oszacowanie. Niedokładność jest mierzona jako pierwiastek błędu średniokwadratowego T, tj. o wartość $MSE(T) = \mathbb{E}\left((T - \Theta)^2\right)$. Efektywne oszacowanie minimalizuje tę wartość.

Szacunki punktowe

Właściwości te wpływają na implementację charakterystyki próbki w R.

Spójrzmy na średnią próbki. Jeśli mamy losową próbkę $X_1, \ldots, X_n$ z rozkładu ze średnią μ, możemy obliczyć:

$$\mathbb{E}(\overline{x}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}(X_i) = \frac{1}{n} n\mu = \mu.$$

Pokazaliśmy więc, że średnia jest bezstronnym oszacowaniem średniej próbki.

Szacunki punktowe

Przyjrzyjmy się teraz wariancji próbki. Wykorzystamy nierówności

$$\sum_{i=1}^{n}(X_i - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu + \mu - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu)^2 - n(\overline{X} - \mu)^2,$$

otrzymujemy

$$\begin{aligned}\mathbb{E}\left(\sum_{i=1}^{n}(X_i - \overline{X})^2\right) &= \mathbb{E}\left(\sum_{i=1}^{n}(X_i - \mu)^2 - n(\overline{X} - \mu)^2\right) \\ &= \sum_{i=1}^{n}\mathbb{D}\left(\overline{X_i}\right) - n\mathbb{D}\left(\overline{X}\right) \\ &= n\sigma^2 - \frac{n\sigma^2}{n} = (n-1)\sigma^2,\end{aligned}$$

Szacunki punktowe

Przyjrzyjmy się teraz wariancji próbki. Wykorzystamy nierówności

$$\sum_{i=1}^{n}(X_i - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu + \mu - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu)^2 - n(\overline{X} - \mu)^2,$$

a więc

$$\mathbb{E}\left(s^2\right) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2\right) = \frac{n-1}{n}\sigma^2.$$

Widzimy zatem, że wariancja próbki nie jest bezstronnym oszacowaniem wariancji losowej.

Szacunki punktowe

Przyjrzyjmy się teraz wariancji próbki. Wykorzystamy nierówności

$$\sum_{i=1}^{n}(X_i - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu + \mu - \overline{X})^2 = \sum_{i=1}^{n}(X_i - \mu)^2 - n(\overline{X} - \mu)^2,$$

Aby uzyskać obiektywne oszacowanie, musimy pomnożyć wariancję próbki przez $\frac{n}{n-1}$. Otrzymujemy w ten sposób nieobciążone oszacowanie wariancji

$$s_{n-1}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

Ten wynik wyjaśnia, dlaczego funkcja `var()` jest zaimplementowana inaczej niż wariancja próbki. Stanowi nieobciążone oszacowanie oryginalnej zmiennej losowej.

Szacunki punktowe

Metody

W tym kursie przedstawimy dwie metody konstruowania Szacunków punktowych:

- metoda momentów,
- metoda największej wiarygodności.

Załóżmy, że mamy próbkę $X_1, \ldots, X_n$ z rozkładu, który zależy od wektora parametrów $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$.

Metoda momentów

Załóżmy dalej, że istnieją wszystkie momenty $\nu_k = \mathbb{E}\left(X_i^k\right)$, $k = 1, \ldots, m$.

Momenty próbki v_k są zdefiniowane jako $v_k = \frac{1}{n}\sum_{i=1}^{n} X_i^k$ dla $k = 1, \ldots, m$.

Zasadą metody momentów jest równość momentu teoretycznego i momentu próbki.

Metoda momentów

Oznacza oszacowanie momentów dla $\theta_1, \theta_2, \ldots, \theta_k$, etykieta $\widehat{\theta}_1, \widehat{\theta}_2, \ldots, \widehat{\theta}_k$, które są zdefiniowane jako rozwiązanie (jeśli istnieją) równań:

$$\nu_k = v_k, \quad k = 1, \ldots, m.$$

Alternatywnie, zamiast dowolnego r-tego momentu, możemy użyć zdefiniowanego r-tego momentu centralnego jako $\mu_r = \mathbb{E}\left((X - \mathbb{E}(X))^r\right)$ i r-ty centralny moment próbki $m_r = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^r$.

Metoda momentów

Zilustrujemy metodę na przypadku rozkładu jednostajnego parametrami a i b. Wiadomo, że momenty zmiennej losowej X o rozkładzie jednostajnym są równe

$$\mathbb{E}(X) = \frac{a+b}{2} \quad \text{i} \quad \mathbb{D}(X) = \frac{(b-a)^2}{12}.$$

Określ szacunki pametry kwadratowe a i b odpowiadają rozwiązaniu równań:

$$\overline{x} = \frac{a+b}{2},$$
$$s^2 = \frac{(b-a)^2}{12}.$$

Metoda momentów

Rozwiązując te równania otrzymujemy:

$$a = \overline{x} - \sqrt{3}s,$$
$$b = \overline{x} + \sqrt{3}s.$$

Jesteśmy teraz gotowi do zaimplementowania tych estymatorów w R. Użyjemy predefiniowanych funkcji `variation()` resp. `stdev()`.

Metoda momentów

```
1 > x<-runif(1000,1,3) #generating the random sample
2 > a<-mean(x)-sqrt(3)*stdev(x)
3 > b<-mean(x)+sqrt(3)*stdev(x)
4 > a
5 [1] 1.018323 #can differ for other samples
6 > b
7 [1] 3.033395 #can differ for other samples
```

Metoda największej wiarygodności

Metoda ta opiera się na maksymalizacji funkcji wiarygodności.

Punkt w przestrzeni parametrów, który maksymalizuje funkcję wiarygodności, jest uważany za oszacowanie maksymalnego prawdopodobieństwa.

Formalnie załóżmy, że $X_1, \ldots, X_n$ są niezależnymi zmiennymi losowymi o tym samym rozkładzie i gęstości $f(x, \boldsymbol{\theta})$. Ujednolicona gęstość jest iloczynem tych jednowymiarowych funkcji gęstości:

$$L(x_1, \ldots, x_n; \boldsymbol{\theta}) = \prod_{i=1}^{n} f(x, \boldsymbol{\theta}).$$

Metoda największej wiarygodności

Wprowadzona funkcja $L(x_1, \ldots, x_n; \boldsymbol{\theta})$ jest nazywana funkcją prawdopodobieństwa.

Aby uzyskać oszacowania parametrów, maksymalizujemy tę funkcję za pomocą standardowego procesu znanego z analizy matematycznej.

Dla uproszczenia pracy zamiast funkcji $L(x_1, \ldots, x_n; \boldsymbol{\theta})$ maksymalizujemy jej logarytm $\ln L(x_1, \ldots, x_n; \boldsymbol{\theta})$.

Logarytm naturalny jest funkcją rosnącą, więc ma ekstrema, a ponadto zamienia iloczyn na sumę funkcji.

Metoda największej wiarygodności

Ilustrujemy metodę na problemie oszacowania prawdopodobieństwa p pewnego zdarzenia losowego A.

Możemy zinterpretować tę sytuację jako wynik alternatywnej zmiennej losowej, która jest wskaźnikiem zdarzenia losowego A.

Następnie mamy $\mathbb{P}(X = 1) = p$ i $\mathbb{P}(X = 0) = 1 - p$.

Wykonujemy losowe próby n i obserwujemy wystąpienie zdarzenia A.

Otrzymujemy więc próbkę $X_1, \ldots, X_n$ zmiennych losowych o gęstościach $f(x_i, p) = p^{x_i}(1-p)^{1-x_i}$, gdzie $x_i \in \{0, 1\}$.

Metoda największej wiarygodności

Odpowiednia funkcja prawdopodobieństwa ma postać

$$L(x,p) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^{\sum_{i=1}^{n} x_i}(1-p)^{n-\sum_{i=1}^{n} x_i}.$$

Aby wyznaczyć oszacowanie p , maksymalizujemy funkcję $L(x,p)$ względem parametru p . Aby to osiągnąć, używamy logarytmu naturalnego funkcji wiarygodności

$$\ln(L(x,p)) = \sum_{i=1}^{n} x_i \ln p + \left(n - \sum_{i=1}^{n} x_i\right) \ln(1-p),$$

Metoda największej wiarygodności

Ustalamy jego pochodną względem p równą 0:

$$\frac{\mathrm{d} \ln L(x,p)}{\mathrm{d}x} = \frac{\sum_{i=1}^{n} x_i}{p} - \frac{n - \sum_{i=1}^{n} x_i}{1-p} = 0.$$

Mnożąc równanie $\frac{1}{n}$ otrzymujemy

$$\overline{x} \cdot \frac{1}{p} - (1 - \overline{x}) \cdot \frac{1}{1-p} = 0,$$

a rozwiązaniem jest

$$p = \overline{x}.$$

Metoda największej wiarygodności

Aby potwierdzić, że $p = \overline{x}$ rzeczywiście maksymalizuje funkcję wiarygodności, musimy zweryfikować pochodną drugiego rzędu.

$$\frac{\mathrm{d}^2 \ln L(x,p)}{\mathrm{d}x^2} = -\frac{\overline{x}}{p} + (1-\overline{x}) \cdot \frac{1}{(1-p)^2}.$$

Podstawiając $p = \overline{x}$ otrzymujemy

$$\left(\frac{\mathrm{d}^2 \ln L(x,p)}{\mathrm{d}x^2} \right)_{p=\overline{x}} = -\frac{1}{1-\overline{x}} < 0.$$

Wtedy mamy najbardziej prawdopodobne oszacowanie $p = \overline{x}$.

Metoda największej wiarygodności

Możemy użyć tego podejścia, aby określić, jak sprawiedliwy jest rzut monetą.

Najpierw generujemy próbkę $0i1$, co oznacza orzeł lub reszka. Aby uzyskać próbkę nieuczciwej monety, deklarujemy wektor prawdopodobieństw `prob`, jak widać w kodzie źródłowym:

```
1 > x<-sample(c(0,1),1000,replace=TRUE,prob=c(2/3,1/3))
2 > mean(x)
3 [1] 0.304
```

Przedziały ufności

Przedział ufności można zdefiniować jako zakres oszacowań dla nieznanego parametru, który zawiera skubieżąca wartość parametru z zadanym prawdopodobieństwem.

To prawdopodobieństwo, że parametr mieści się w danym przedziale, jest określane jako poziom ufności.

Najczęściej stosowanym w praktyce poziomem ufności jest 95 %, ale często stosuje się inne poziomy (takie jak 90 % lub 99 %).

Przedziały ufności

Formalnie niech $\mathbf{X} = (X_1, \ldots, X_n)$ będzie próbą losową o rozkładzie zależnym od nieznanego parametru θ.

Przedział ufności dla parametru θ z poziomem ufności α to przedział z losowymi punktami końcowymi $(u(\mathbf{X}); v(\mathbf{X}))$ określonymi przez parę losowych zmienne $u(\mathbf{X})$ i $v(\mathbf{X})$ z właściwością:

$$\mathbb{P}\left(u(\mathbf{X}) < \theta < v(\mathbf{X})\right) = \alpha.$$

Przedziały ufności

Ilustrujemy wyprowadzenie przedziału ufności na rozkładzie normalnym.

Załóżmy najpierw, że $X_1, \ldots, X_n$ jest próbą losową z rozkładu normalnego $N(\mu, \sigma^2)$, którego odchylenie standardowe σ jest znane.

Chcemy znaleźć przedział ufności dla średniej μ. Ponieważ $\overline{X}$ ma rozkład normalny $N\left(\mu, \frac{\sigma^2}{n}\right)$, mamy

$$\mathbb{P}\left(\left|\frac{\overline{X}-\mu}{\frac{\sigma}{\sqrt{n}}}\right| < c\right) = 2\Phi(c) - 1, \text{ dla wszystkich } c > 0,$$

Przedziały ufności

To jest równoważne

$$\mathbb{P}\left(\overline{X} - c\frac{\sigma}{\sqrt{n}} < \mu < \overline{X} + c\frac{\sigma}{\sqrt{n}}\right) = 2\Phi(c) - 1, \text{ dla wszystkich } c > 0.$$

Z ostatniej zależności wynika, że przedział ufności dla średniej z poziomem ufności $\alpha = 2\Phi(c) - 1$ ma postać

$$\left(\overline{X} - c\frac{\sigma}{\sqrt{n}}; \overline{X} + c\frac{\sigma}{\sqrt{n}}\right).$$

Jeśli weźmiemy pod uwagę, że $c = \Phi^{-1}\left(\frac{\alpha+1}{2}\right)$, możemy zapisać przedział ufności w postaci

$$\left(\overline{X} - \Phi^{-1}\left(\frac{\alpha+1}{2}\right)\frac{\sigma}{\sqrt{n}}; \overline{X} + \Phi^{-1}\left(\frac{\alpha+1}{2}\right)\frac{\sigma}{\sqrt{n}}\right).$$

Przedziały ufności

Używamy funkcji kwantylowej qnorm(), aby znaleźć granice przedziału ufności.

```
x<-rnorm(100,1,2) # we generate some sample
> n<-length(x)
> sample.mean<-mean(x)
> sample.sd<-2 #  standard deviation is known
> alpha<-0.95 # seting the confidence level 95\%
> c<-qnorm((alpha+1)/2,0,1)
> margin<-c*sample.sd/sqrt(n)
> lower.bound<-sample.mean-margin
> upper.bound<-sample.mean+margin
> print(c(lower.bound,upper.bound))
[1] 0.8149884 1.5989740
```

Przedziały ufności

Zobaczmy teraz, jak zmienia się przedział ufności, jeśli odchylenie standardowe jest nieznane.

Następnie musimy użyć nieobciążonego oszacowania odchylenia standardowego $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\overline{X} - X_i)^2$, $s = \sqrt{s^2}$. Następnie zmienna losowa

$$T = \frac{\overline{X} - \mu}{s}\sqrt{n},$$

ma rozkład t-Studenta z $n-1$ stopniami swobody. Otrzymujemy wtedy zadany przedział ufności

$$\left(\overline{X} - c\frac{s}{\sqrt{n}}; \overline{X} + c\frac{s}{\sqrt{n}}\right).$$

Przedziały ufności

Wartość c jest odpowiednim kwantylem rozkładu Studenta, więc użyjemy funkcji `qt()` w R.

```
> n<-length(x) # we use previously generated sample
> sample.mean<-mean(x)
> sample.sd<-sd(x) # estimate of the standard deviation
> alpha<-0.95
> c<-qt((alpha+1)/2,df=n-1)
> margin<-c*sample.sd/sqrt(n)
> lower.bound<-sample.mean-margin
> upper.bound<-sample.mean+margin
> print(c(lower.bound,upper.bound))
[1] 0.8118763 1.6020861
```

Przedziały ufności

Przyjrzyjmy się teraz przedziałowi ufności dla wariancji rozkładu normalnego.

Jeśli s^2 jest nieobciążonym oszacowaniem wariancji, to zmienną losową

$$Y = \frac{(n-1)s^2}{\sigma^2} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{\sigma^2} = \sum_{i=1}^{n}\left(\frac{x_i - \overline{x}}{\sigma}\right)^2$$

jest zgodne z rozkładem $\chi^2(n-1)$.

Przedziały ufności

Dla przedziału ufności dostajemy

$$\mathbb{P}(c_1 < \chi^2 < c_2) = \alpha$$
$$\mathbb{P}\left(c_1 < \frac{(n-1)s^2}{\sigma^2} < c_2\right) = \alpha$$
$$\mathbb{P}\left(\sigma \in \left(\frac{(n-1)s^2}{c_2}; \frac{(n-1)s^2}{c_1}\right)\right) = \alpha,$$

gdzie c_1 i c_2 są wartościami krytycznymi rozkładu $\chi^2(n-1)$.

Przedziały ufności

Podczas obliczeń musimy wziąć pod uwagę, że rozkład χ^2 nie jest symetryczny, co jest ważne dla wartości krytycznych c_1 i c_2.

```
1  > n<-length(x) # the sample already generated sooner
2  > sample.var<-var(x)
3  > c1<-qchisq(1-(alpha+1)/2,df=n-1)#consequent of asymmetry
4  > c2<-qchisq((alpha+1)/2,df=n-1)#consequent of asymmetry
5  > lower.bound<-sample.var*(n-1)/c2
6  > upper.bound<-sample.var*(n-1)/c1
7  > print(c(lower.bound,upper.bound))
8  [1] 3.056626 5.350767
```

Przedziały ufności

W ostatnim przykładzie pokażemy przedział ufności dla prawdopodobieństwa zdarzenia losowego.

Nieobciążone oszacowanie prawdopodobieństwa p wystąpienia zdarzenia losowego to $\hat{p} = \frac{m}{n}$, gdzie m to liczba wystąpień obserwowanego zdarzenia w serii z n prób losowych.

Wiemy, że $\mathbb{E}(\hat{p}) = p$ i $\mathbb{D}(\hat{p}) = \frac{pq}{n}$, gdzie $q = 1 - p$.

Dla dużych n, zachodzi przybliżona równość $\frac{pq}{n} \approx \frac{\hat{p}\hat{q}}{n}$.

Przedziały ufności

Następnie zmienna losowa

$$Z = \frac{\frac{m}{n} - p}{\sqrt{\frac{\hat{p}\hat{q}}{n}}}$$

jest zgodny ze standardowym rozkładem normalnym $N(0, 1)$. Wtedy przedział ufności dla prawdopodobieństwa p z poziomem ufności α można zapisać jako

$$\left(\hat{p} - \Phi^{-1}\left(\frac{\alpha+1}{2}\right) \cdot \sqrt{\frac{\hat{p}\hat{q}}{n}}; \hat{p} + \Phi^{-1}\left(\frac{\alpha+1}{2}\right) \cdot \sqrt{\frac{\hat{p}\hat{q}}{n}} \right).$$

Przedziały ufności

Example

Załóżmy, że przeprowadzana jest ankieta wśród 250 losowo wybranych osób w celu ustalenia, czy posiadają one tablet. Spośród 250 respondentów 98 stwierdziło, że posiada tablet. Korzystając z poziomu ufności 95 %, oblicz szacunkowy przedział ufności dla prawdziwego odsetka osób posiadających tablet.

Dziękujemy za uwagę.

Statystyka i programowanie w R

Aleš Kozubík