Łagodne i praktyczne wprowadzenie do R

Paweł Kleka na podstawie materiałów openstax.org

2025-12-12

Spis treści

1 Ciekawostki i tło historyczne

Język **R** jest dialektem języka **S** (S od *statistics*). S został opracowany przez Johna Chambersa w 1976 w Bell Labs jako interaktywne środowisko do analiz statystycznych. S w wersji 4 (S4, 1998) używany jest do dziś, a komercyjną implementacją była **S-PLUS**.

Ponieważ S był płatny, **Ross Ihaka** i **Robert Gentleman** w 1991 r. na Uniwersytecie w Auckland metodą inżynierii wstecznej stworzyli darmowy odpowiednik – R. W 1995 **Martin Mächler** przekonał twórców do użycia licencji GNU GPL, a w 1997 powstała **R Core Team** czuwająca nad rozwojem języka.

Zalety R:

1. Coroczne (październik) wydania + szybkie poprawki błędów.
2. Bardzo bogate możliwości graficzne (base, lattice, ggplot2).
3. Język wyspecjalizowany w analizie statystycznej, ale pełnoprawny do programowania.
4. Integracje: Python, Java, C/C++, Hadoop, bazy danych, Spark.

R używają m.in. Google, Meta, Airbnb, NY Times, Microsoft (R Open).

1.1 Po co się uczyć R i programowania?

- Rozwijanie myślenia algorytmicznego i umiejętności rozwiązywania problemów.
- Automatyzacja powtarzalnych zadań.
- Prowadzenie eksperymentów i symulacji.
- Rzetelność, powtarzalność i transparentność analiz.
- Elastyczne tworzenie nowych narzędzi na potrzeby badań.

Programowanie wymusza precyzję – diagnozowanie błędów doskonali strategię analityczną.

1.2 Czy R jest trudny?

Historycznie składnia bywała nieintuicyjna; ogromny progres przyniósł ekosystem **tidyverse** (pakiety ujednolicające sposoby pracy z danymi). R współpracuje z nowoczesnymi metodami ML (pakiety `keras`, `tensorflow`, `xgboost`) i z innymi językami oraz silnikami danych (Spark, Hadoop).

1.2.1 Słabe wymówki

- “Lepiej policzyć w czymś innym” – praca z kodem pozwala na poprawianie swojej analizy zamiast robienia jej od nowa.
- “Zlecę to komuś” – a jak zweryfikujesz wynik i zakomunikujesz potrzeby?
- “Mam dwie lewe ręce” – tutaj zamiast sekwencji kliknięć uczysz się powtarzalnych przepisów.

1.3 AI a nauka R

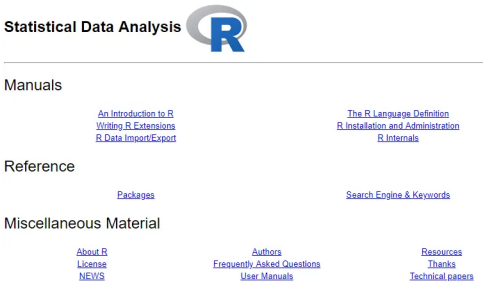
AI potrafi pisać kod R, ale całkowita zależność od sztucznej inteligencji spowalnia zrozumienie i debugowanie. Znajomość podstaw R pozwala precyzyjnie formułować pytania i oceniać wygenerowane przez AI rozwiązania.

R to narzędzie do analizy statystycznej, które jest szeroko stosowane w nauce, gdzie wiedza jest ukryta w danych. R oferuje wiele narzędzi do eksploracji danych, wizualizacji i analizy statystycznej. Jest dostępny jako bezpłatny program typu open source i zapewnia zintegrowany zestaw funkcji do analizy danych, tworzenia wykresów i programowania statystycznego. R jest bardzo często wykorzystywany przez naukowców zajmujących się danymi jako narzędzie do analizy danych i statystyki, częściowo dlatego, że R jest językiem open source, a społeczność użytkowników stale dodaje do niego dodatkowe funkcje. Narzędzie to może być używane na wielu systemach komputerowych i można je pobrać ze strony internetowej projektu R.

Po zainstalowaniu i uruchomieniu programu R na komputerze, w dolnej części konsoli R powinien pojawić się symbol „>”, który oznacza, że program R jest gotowy do przyjmowania poleceń.

Jeśli to Twój pierwszy kontakt z R, napisz w konsoli `help.start()` a otworzy się interaktywny przewodnik po programie wyglądający trochę staromodnie

Ryc. 1. Wygenerowana poleceniem: `help.start()`



R oferuje wiele wbudowanych zasobów pomocy. Po wpisaniu polecenia `help()` w wierszu poleceń R wyświetlana jest lista zasobów pomocy. Aby uzyskać konkretny przykład, należy wpisać polecenie `help(median)`, co spowoduje wyświetlenie różnych dokumentów dotyczących wbudowanej funkcji mediany w języku R.

Ponadto, jeśli użytkownik wpisze `demo()` w wierszu poleceń R, wyświetlą się różne opcje demonstracyjne. Na przykład wpisanie `demo(graphics)` spowoduje wyświetlenie kilku przykładów różnych wykresów graficznych.

R jest językiem sterowanym poleceniami, co oznacza, że użytkownik wprowadza polecenia w wierszu poleceń, a R wykonuje je pojedynczo. R może również wykonywać programy zawierające wiele poleceń. Istnieją sposoby dodania graficznego interfejsu użytkownika (GUI) do języka R. Przykładem najpopularniejszego narzędzia GUI dla języka R jest RStudio.

2 Uruchomienie i środowisko pracy

Do komfortowej pracy:

1. Samo **R** jest tu: https://cran.r-project.org
2. **RStudio** (od Posit) Desktop: https://posit.co
3. **Positron** (też od Posit), które jest dostosowanym VSCode do pracy z R

Można używać też samego VSCode (+ rozszerzenie R), ale RStudio jest najstabilniejszą drogą w tej chwili.

Pakiety rozszerzające funkcjonalność programu instalujemy poleceniem:

```
install.packages("nazwa_pakietu")
```

A ładujemy (udostępniamy w sesji):

```
library(nazwa_pakietu)
```

Przydatne na start: `formatR`, `ggplot2`, `dplyr`, `tidyr`, `psych`.

---

3 Pierwsze kroki w konsoli R

Po uruchomieniu R/RStudio w konsoli pojawia się znak zachęty `>` – tu wpisujemy pojedyncze polecenia lub uruchamiamy cały skrypt.

Pomoc:

- `help.start()` – interaktywny indeks pomocy.
- `help(funkcja)` lub `?funkcja` – dokumentacja jednej funkcji.
- `??wzorzec` – szukanie wśród tematów pomocy.
- `demo()` / `demo(graphics)` – pokaz zastosowań.
- `citation()` – sposób cytowania R w pracy naukowej.

3.1 R jako kalkulator

```
2 + 3    # dodawanie
2 - 3    # odejmowanie
2 * 3    # mnożenie
2 / 3    # dzielenie
2 ^ 3    # potęgowanie
4 ^ 3 - 2 * 3
(4 ^ 3) - (2 * 3)
4 ^ (3 - 2) * 3
```

3.2 Obiekty i przypisania

Większość obliczeń w języku R jest wykonywana za pomocą funkcji. W przypadku analizy statystycznej w języku R dostępnych jest wiele gotowych funkcji do obliczania średniej, mediany, odchylenia standardowego, kwartyli itp. Zmienne można nazwać i przypisać im wartości za pomocą operatora przypisania `<-`.

```
a <- 13
b <- a / 2
cc <- a / b
print(cc)
```

```
[1] 2
```

```
# print(Cc)  # Wygeneruje błąd, ponieważ wielkość liter ma znaczenie
```

Na przykład poniższe polecenia języka R przypisują wartość 20 do obiektu o nazwie x i wartość 30 do obiektu o nazwie y:

```
# Przypisanie wartości do zmiennych
x <- 20
y <- 30
# Mnożenie
x * y
```

```
[1] 600
```

Typową metodą wykorzystania funkcji w aplikacjach statystycznych jest najpierw utworzenie wektora wartości danych. Istnieje kilka sposobów tworzenia wektorów w języku R. Na przykład funkcja `c()` jest często używana do łączenia wartości w wektor. Poniższe polecenie R wygeneruje wektor o nazwie ***salaries***, zawierający wartości danych 40000, 50000, 75000 i 92000:

```
# Tworzenie wektora
salaries <- c(40000, 50000, 75000, 92000)
```

Wartości te można następnie wykorzystać w funkcjach statystycznych, takich jak średnia, mediana, min, max itp., jak pokazano poniżej.:

```
# Statystyki opisowe
mean(salaries)    # Średnia
```

```
[1] 64250
```

```
median(salaries)  # Mediana
```

```
[1] 62500
```

```
min(salaries)     # Minimum
```

```
[1] 40000
```

```r
max(salaries)     # Maksimum
```

```
[1] 92000
```

Inną opcją generowania wektora w języku R jest użycie funkcji `seq()`, która automatycznie generuje sekwencję liczb. Na przykład możemy wygenerować sekwencję liczb od 1 do 5, zwiększaną o 0,5, i nazwać ten wektor *Przykład1*, w następujący sposób:

```r
# Sekwencja liczb
Przykład1 <- seq(1, 5, by = 0.5)
```

Jeśli następnie wpiszemy nazwę wektora i naciśniemy klawisz Enter, R wyświetli listę wartości liczbowych dla tej nazwy wektora.

```r
# Wyświetlanie wektorów
Przykład1
```

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

Często jesteśmy zainteresowani szybkim sporządzeniem podsumowania statystycznego zbioru danych w postaci średniej, mediany, kwartyli, wartości minimalnej i maksymalnej. Polecenie R o nazwie `summary()` zapewnia takie wyniki.

```r
# Podsumowanie statystyczne
summary(salaries)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  40000   47500   62500   64250   79250   92000
```

W przypadku miar rozrzutu R zawiera polecenie dotyczące odchylenia standardowego, zwane `sd()`, oraz polecenie dotyczące wariancji, zwane `var()`. Odchylenie standardowe i wariancja są obliczane przy założeniu, że zbiór danych został zebrany z próby.

```r
# Miary rozrzutu
sd(salaries)   # Odchylenie standardowe
```

```
[1] 23641.42
```

```r
var(salaries)  # Wariancja
```

```
[1] 558916667
```

Funkcje pozwalające zobaczyć fragment zbioru danych:

```r
head(poziom, n = 10)      # pierwsze 10 wartości
tail(sekwencja)           # ostatnie
```

3.3 Przestrzeń robocza (workspace)

Pracując w R tworzymy różne obiekty. Za pomocą funkcji `ls()` możemy zobaczyć ich listę.

```r
ls()            # lista obiektów
```

```
[1] "a"         "b"         "cc"        "Przykład1" "salaries"  "x"
[7] "y"
```

```r
rm(a)           # usuń obiekt a
ls()
```

```
[1] "b"         "cc"        "Przykład1" "salaries"  "x"         "y"
```

```r
rm(list = ls()) # usuń wszystko
ls()
```

```
character(0)
```

Zapisywanie / wczytywanie sesji:

```r
save.image("~/Desktop/MyWorkspace.RData")
rm(list = ls(all = TRUE))
load("~/Desktop/MyWorkspace.RData")
```

(Uwaga na różnice ścieżek między systemami – w Windows użyj np. `C:/Users/...`)

---

4 Typy danych i wartości specjalne

R ma kilka podstawowych typów atomowych:

- logical: `TRUE`, `FALSE`
- integer: liczby całkowite
- double (numeric): liczby zmiennoprzecinkowe (domyślne)
- character: ciągi tekstowe w cudzysłowach `"tekst"`
- complex: liczby zespolone `1+2i`
- raw: rzadko używane bajty

Wartości specjalne:

- `NA` brakująca wartość (istnieją też `NA_integer_`, `NA_real_`, itd.)
- `NaN` not-a-number (np. `0/0`)
- `Inf`, `-Inf` (np. `1/0`, `-1/0`)

Sprawdzenia i konwersje:

```
x <- c(1, 2, NA, 5)
is.na(x)            # które elementy to NA?
anyNA(x)            # czy istnieje NA w wektorze x?
as.character(1:3)   # konwersja z liczb na znaki
```

Faktory (czynniki) – typ do przechowywania zmiennych kategorialnych z poziomami:

```
plec_chr <- c("K", "M", "K", "K")
plec_fac <- factor(plec_chr,
                   levels = c("K", "M"),
                   labels = c("Kobieta", "Mężczyzna"))
print(plec_fac)
```

```
[1] Kobieta   Mężczyzna Kobieta   Kobieta
Levels: Kobieta Mężczyzna
```

```
levels(plec_fac)
```

```
[1] "Kobieta"   "Mężczyzna"
```

Zmienne porządkowe (*ordered factors*):

```r
ocena <- factor(c("dostateczny","bardzo dobry","dobry"),
                levels = c("dostateczny","dobry","bardzo dobry"),
                ordered = TRUE)
ocena > "dobry"   # TRUE dla "bardzo dobry"
```

```
[1] FALSE  TRUE FALSE
```

---

5 Struktury danych w R

Zmienne różnego typy można łączyć w struktury

5.1 Wektory

Podstawowy kontener jednowymiarowy tej samej klasy.

```r
pory.roku <- c(-10, 25, 16, 5)
names(pory.roku) <- c("zima", "wiosna", "lato", "jesień")
pory.roku

# Alternatywnie nadawanie nazw wprost:
pory.roku <- c(zima = -10, wiosna = 25, lato = 16, jesien = 5)
length(pory.roku)
length(names(pory.roku))
```

5.2 Macierze

Jednorodna pod względem typu danych tablica 2D:

```r
mat <- matrix(c(1,2,3,4,5,6,7,8,9), nrow = 3, ncol = 3)
mat
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

5.3 Ramki danych (data.frame)

Tablica kolumn o (potencjalnie) różnych klasach + atrybuty.

```
df <- data.frame(var1 = c(1,2,3,4,5),
                 var2 = c("k","m","k","k","m"),
                 var3 = c(TRUE, TRUE, FALSE, FALSE, TRUE))
attributes(df)

names(df) <- c("Lp", "Płeć", "Fakt")
attributes(df)
df
```

5.4 Listy

Heterogeniczne (składające się z różnych typów) kolekcje dowolnych obiektów:

```
moja.lista.1 <- list(a = 1, b = "ala")
moja.lista.2 <- list(a = 1, b = c("ala", "kot"))
length(moja.lista.2)

[1] 2
```

Lista popularnych poleceń statystycznych języka R znajduje się w Tabela 1.

tab. 1: Lista popularnych poleceń statystycznych programu R

Polecenie R	**Działanie**
`mean()`	Oblicza średnią arytmetyczną
`median()`	Oblicza medianę
`min()`	Oblicza minimalną wartość
`max()`	Oblicza wartość maksymalną
`weighted.mean()`	Oblicza średnią ważoną
`sum()`	Oblicza sumę wartości
`summary()`	Oblicza średnią, medianę, kwartyle, minimalną i maksymalną wartość.
`sd()`	Oblicza odchylenie standardowe próby.
`var()`	Oblicza wariancję próby
`IQR()`	Oblicza rozstęp międzykwartylowy
`barplot()`	Wykreśla wykres słupkowy danych nie numerycznych.

Polecenie R	Działanie
`boxplot()`	Wykreśla wykres pudełkowy danych liczbowych.
`hist()`	Wykreśla histogram danych liczbowych.
`plot()`	Wykreśla różne wykresy, w tym wykres punktowy.
`freq()`	Tworzy tabelę rozkładu częstotliwości
`unique()`	Lista niepowtarzalnych wartości

5.5 Tworzenie i używanie wektorów do analiz statystycznych

Poniżej przedstawiono przykład obliczenia średniej ważonej w języku R:

5.5.1 Przykład P.1

Problem

Załóżmy, że portfel finansowy zawiera 1000 akcji spółki XYZ Corporation, zakupionych w trzech różnych terminach, jak pokazano w tabeli Tabela 2. Użyj języka R do obliczenia średniej ważonej ceny zakupu, gdzie wagi są oparte na liczbie akcji w portfelu.

tab. 2: Portfolio akcji XYZ

Data zakupu	Cena ($)	Liczba kupionych akcji
Styczeń 17	78	200
Luty 10	122	300
Marzec 23	131	500
Razem		**1000**

Rozwiązanie

Oto jak utworzyć dwa wektory w języku R: wektor ceny będzie zawierał cenę zakupu, a wektor akcji będzie zawierał liczbę akcji. Następnie należy wykonać polecenie R `weighted.mean(price, shares)` w następujący sposób:

```
> price <- c(78, 122, 131)

> shares <- c(200, 300, 500)

> weighted.mean(price, shares)
```

```
[1] 117.7
```

6 Podstawowa wizualizacja i tworzenie wykresów przy użyciu R

R oferuje wiele wbudowanych funkcji do wizualizacji danych, takich jak wykresy słupkowe, histogramy, wykresy punktowe i inne. Poniżej przedstawiono spójny opis i przykłady najważniejszych typów wykresów.

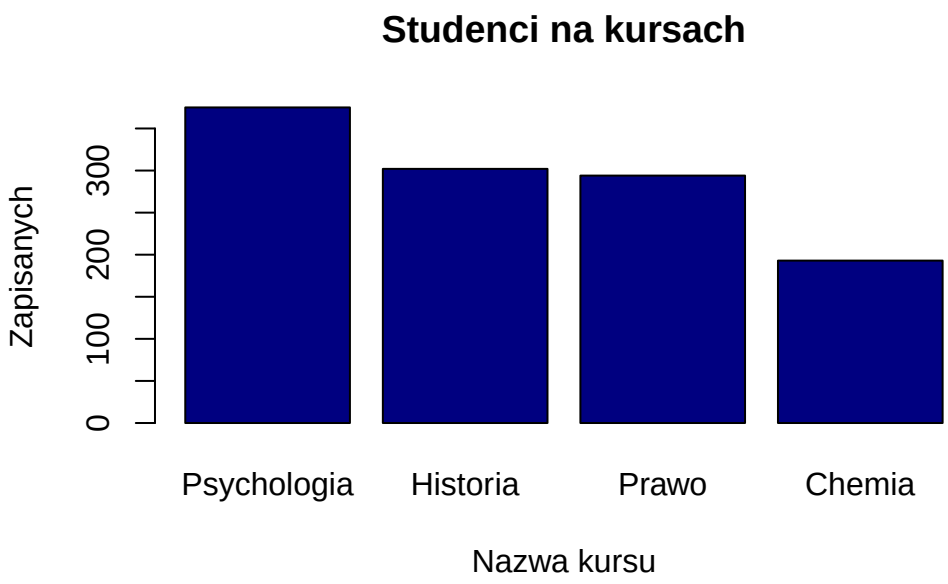
6.1 Wykres słupkowy (barplot)

Wykresy słupkowe służą do prezentacji danych kategorialnych Przykład: liczba zapisanych studentów na różne kursy.

tab. 3: Liczba zapisanych studentów na kursy

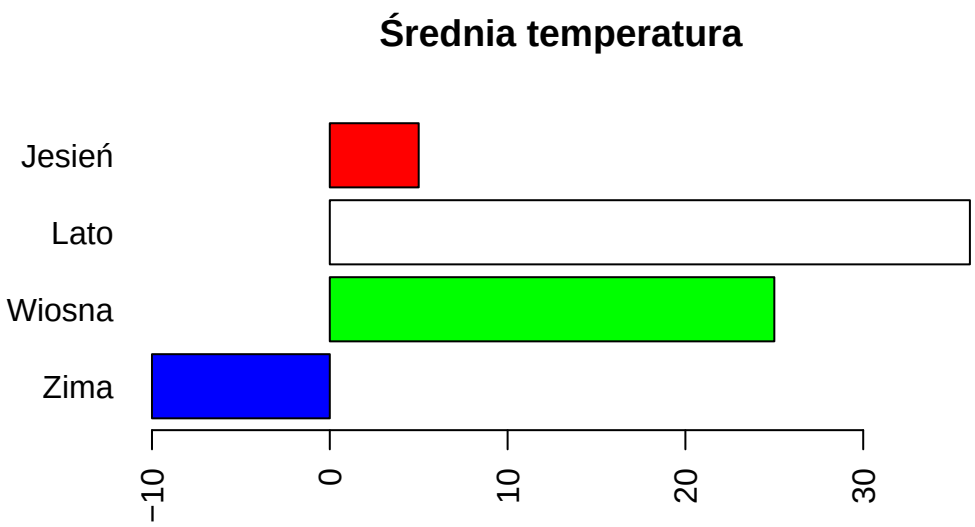
Kurs	Liczba zapisanych
Psychologia	375
Historia	302
Prawo	294
Chemia	193

```
enrollment <- data.frame(
  course = c("Psychologia", "Historia", "Prawo", "Chemia"),
  enrolled = c(375, 302, 294, 193)
)
barplot(
  enrollment$enrolled,
  names.arg = enrollment$course,
  main = "Studenci na kursach",
  xlab = "Nazwa kursu",
  ylab = "Zapisanych",
  col = "navyblue"
)
```



Możliwości dostosowywania wykresów są szerokie, np. zmiana kolorów, orientacji, etykiet:

```r
# Przykłady z innym zbiorem danych
pory.roku <- c(zima = -10, wiosna = 25, lato = 36, jesien = 5)
barplot(
  pory.roku,
  main = "Średnia temperatura",
  col = c("blue", "green", "white", "red"),
  names.arg = c("Zima", "Wiosna", "Lato", "Jesień"),
  horiz = TRUE,
  las=c(2)
)
```



6.2 Wykres punktowy

Podstawowym poleceniem do tworzenia wykresu punktowego w R jest `plot(x, y)`, gdzie x i y to wektory danych liczbowych. Wykresy punktowe są szczególnie przydatne do wizualizacji zależności między dwiema zmiennymi.

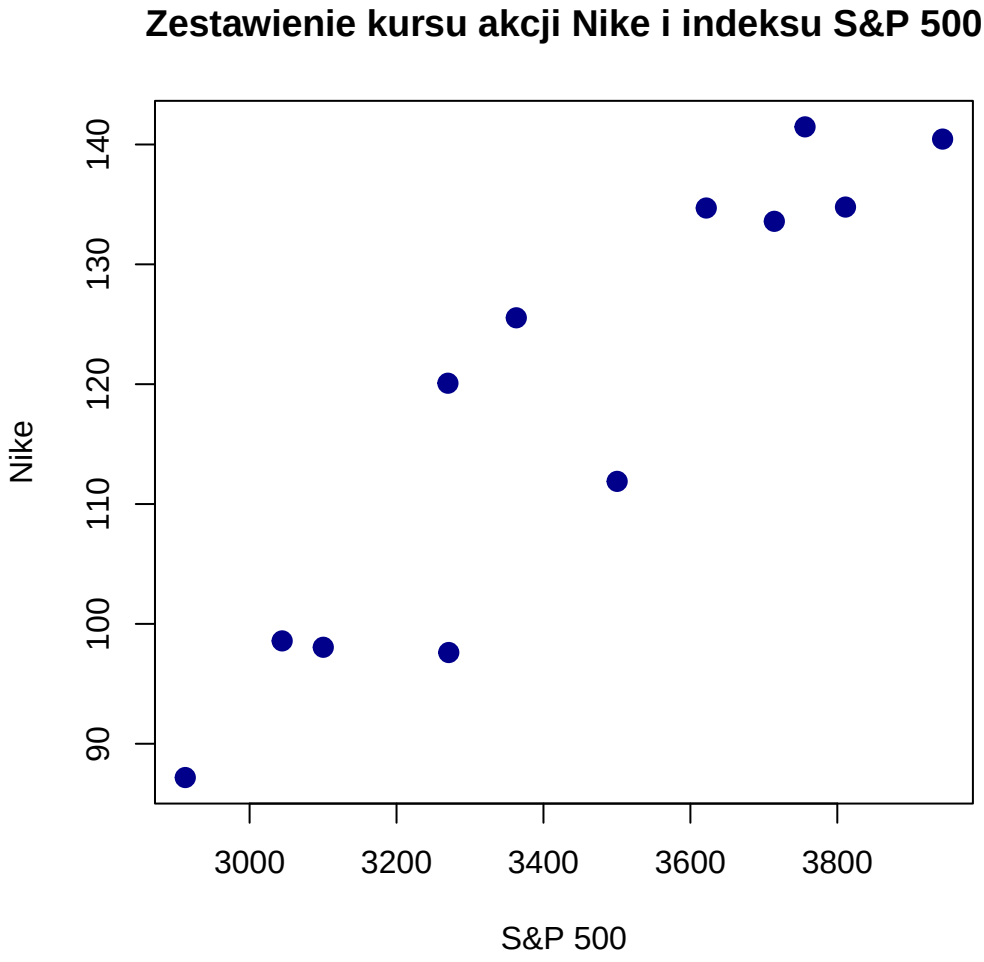
Przykład: Zależność kursu akcji Nike od indeksu S&P 500

tab. 4: Dane miesięczne: S&P 500 i kurs akcji Nike

Data	S&P 500	Nike ($)
4/1/2020	2912.43	87.18
5/1/2020	3044.31	98.58
6/1/2020	3100.29	98.05
7/1/2020	3271.12	97.61
8/1/2020	3500.31	111.89
9/1/2020	3363.00	125.54
10/1/2020	3269.96	120.08
11/1/2020	3621.63	134.70
12/1/2020	3756.07	141.47
1/1/2021	3714.24	133.59
2/1/2021	3811.15	134.78

Data	S&P 500	Nike ($)
3/1/2021	3943.34	140.45

```
# Wykres punktowy
plot(SP500, Nike,
     main = "Zestawienie kursu akcji Nike i indeksu S&P 500",
     xlab = "S&P 500", ylab = "Nike",
     pch = 20, col = "darkblue", cex = 2)
```



7 Wprowadzenie do rozkładów prawdopodobieństwa

Zanim przejdziemy do rozkładów prawdopodobieństwa, warto przypomnieć sobie podstawowe pojęcia:

- **Zmienne losowe** – to wielkości, których wartości zależą od przypadku (np. liczba wyrzuconych oczek na kostce, wzrost losowo wybranej osoby).
- **Rozkład prawdopodobieństwa** – opisuje, z jakim prawdopodobieństwem zmienna losowa przyjmuje określone wartości. Rozkłady mogą być dyskretne (np. rzut monetą) lub ciągłe (np. wzrost, masa ciała).

W praktyce rozkłady prawdopodobieństwa pozwalają: - przewidywać częstość występowania określonych wyników, - modelować zjawiska losowe, - obliczać prawdopodobieństwa zdarzeń, - wyznaczać przedziały ufności i testować hipotezy.

Najczęściej spotykane rozkłady to: rozkład normalny (ciągły), dwumianowy (dyskretny), Poissona (dyskretny), wykładniczy (ciągły).

Poniżej przedstawiam praktyczne zastosowania rozkładów w R.

7.1 Rozkłady prawdopodobieństwa

Jak wspomniano wcześniej, szybkie podsumowanie statystyczne można wygenerować za pomocą polecenia `summary()` w języku R, którego użycie wygląda następująco: `summary(data_vector)`.

R posiada rozbudowane wbudowane biblioteki do obliczania przedziałów ufności, testowania hipotez i pracy z różnymi rozkładami prawdopodobieństwa. Analitycy danych często interesują się różnymi rozkładami prawdopodobieństwa, takimi jak rozkład normalny, rozkład dwumianowy i rozkład Poissona.

R używa polecenia `pnorm` do znalezienia pola pod krzywą normalną po lewej stronie określonej wartości:

Użycie: `pnorm(x_value, mean, standard_deviation)`

gdzie `pnorm` zwraca prawdopodobieństwo, że zmienna losowa o danej średniej i odchyleniu standardowym jest mniejsza niż `x_value`.

7.1.1 Przykład P.2

Problem

Masa urodzeniowa noworodków w Stanach Zjednoczonych ma rozkład normalny o średniej 3400 gramów i odchyleniu standardowym 500 gramów.

1. Użyj programu R, aby obliczyć prawdopodobieństwo, że losowo wybrany noworodek waży mniej niż 4000 gramów.
2. Użyj programu R, aby obliczyć prawdopodobieństwo, że losowo wybrany noworodek waży więcej niż 3000 gramów.

Rozwiązanie

Dla części (a), użyjemy funkcji `pnorm` w ten sposób:

```
pnorm(4000, 3400, 500)
```

```
[1] 0.8849303
```

Dla części (b), musimy dodać opcję `lower.tail=FALSE`, aby obliczyć powierzchnię pod krzywą rozkładu prawdopodobieństwa po prawej stronie:

```
pnorm(3000, 3400, 500, lower.tail=FALSE)
```

```
[1] 0.7881446
```

R udostępnia również wbudowaną funkcję rozkładu dwumianowego, która wygląda następująco: `pbinom(k, n, p)`.

gdzie n to liczna prób, p to prawdopodobieństwo jednego sukcesu, a k jest liczbą sukcesów, dla których pożądane jest prawdopodobieństwo.

7.1.2 Przykład P.3

Problem

Załóżmy, że przeprowadzamy ankietę wśród próby 20 osób i zadajemy pytanie: „Czy uważasz, że strona internetowa firmy ABC jest łatwa w nawigacji?". Na podstawie danych zebranych w przeszłości wiemy, że 65% osób uznało stronę za łatwą w nawigacji. Użyj języka R, aby obliczyć prawdopodobieństwo, że 13 z 20 respondentów uznało stronę za łatwą w nawigacji.

Rozwiązanie

Użyjemy funkcji `pbinom` w taki sposób:

```r
pbinom(13, 20, 0.65)
```

```
[1] 0.5833746
```

8 Podstawowa analiza korelacji i regresji przy użyciu języka R

Przypomnij sobie omówienie korelacji i analizy regresji w sekcji Statystyka wnioskowa i analiza regresji. Pierwszym krokiem w analizie korelacji jest obliczenie współczynnika korelacji (r) dla danych (x, y). R udostępnia wbudowaną funkcję `cor()` do obliczania współczynnika korelacji dla danych dwuwymiarowych.

Jako przykład rozważmy dane przedstawione w tabeli poniżej

tab. 5: Miesięczne zwroty akcji Coca-Cola i indeksu S&P 500

Miesiąc	**S&P 500 zwrot (%)**	**Coca-Cola zwrot (%)**
Jan	8	6
Feb	1	0
Mar	0	-2
Apr	2	1
May	-3	-1
Jun	7	8
Jul	4	2

Aby obliczyć współczynnik korelacji dla tego zbioru danych, należy najpierw utworzyć dwa wektory w języku R, jeden wektor dla zwrotów z indeksu S&P 500, a drugi wektor dla zwrotów z Coca-Coli:

```r
SP500 <- c(8,1,0,2,-3,7,4)
CocaCola <- c(6,0,-2,1,-1,8,2)
```

Polecenie R o nazwie `cor` zwraca współczynnik korelacji dla wektora danych x i wektora danych y:

```r
cor(SP500, CocaCola)
```

```
[1] 0.9123872
```

Czyli współczynnik korelacji dla tego zbioru danych wynosi około 0,912.

8.1 Modele regresji liniowej z wykorzystaniem języka R

Aby utworzyć model liniowy w języku R, zakładając, że korelacja jest istotna, użyjemy polecenie `lm()` (dla modelu liniowego), co zapewni nam obliczenie nachylenia i punkt przecięcia z osią y dla równania regresji liniowej.

Format polecenia R jest następujący:

```
lm(dependent_variable_vector ~ independent_variable_vector)
```

Zwróć uwagę na użycie symbolu tyldy (to taki wężyk, przeważnie na klawiaturze koło jedynki) jako separatora między wektorem zmiennej zależnej a wektorem zmiennej niezależnej.

Jako zmienną zależną wykorzystujemy zwroty z akcji Coca-Coli, a jako zmienną niezależną zwroty z indeksu S&P 500, więc polecenie R będzie wyglądało następująco

```
lm(CocaCola ~ SP500)
```

```
Call:
lm(formula = CocaCola ~ SP500)

Coefficients:
(Intercept)        SP500
    -0.3453       0.8641
```

Wynik w R podaje wartość punktu przecięcia z osią y jako -0,3453, a wartość nachylenia jako 0,8641. Na tej podstawie model liniowy będzie wyglądał następująco:

$$\hat{y} = a + b \cdot x = -0.3453 + 0.8641 \cdot x$$

gdzie x oznacza wartość zwrotu z indeksu S&P 500, a y oznacza wartość zwrotu z akcji Coca-Coli.

Wyniki można również zapisać jako formułę i nazwać „modelem” za pomocą następującego polecenia R. Aby uzyskać bardziej szczegółowe wyniki regresji liniowej, można użyć polecenia `summary` w następujący sposób:

```
model <- lm(CocaCola ~ SP500)

summary(model)
```

```
Call:
lm(formula = CocaCola ~ SP500)

Residuals:
      1       2       3       4       5       6       7
-0.5672 -0.5187 -1.6547 -0.3828  1.9375  2.2969 -1.1109

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.3453     0.7836  -0.441  0.67783
SP500         0.8641     0.1734   4.984  0.00416 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.658 on 5 degrees of freedom
Multiple R-squared:  0.8325,    Adjusted R-squared:  0.7989
F-statistic: 24.84 on 1 and 5 DF,  p-value: 0.004161
```

W tym wyniku podano jak wcześniej punkt przecięcia z osią y i nachylenie, a także reszty dla każdej wartości x. Wynik zawiera dodatkowe szczegóły statystyczne dotyczące analizy regresji.

W programie R można również generować wartości prognozowane i przedziały dla tych prognoz. Najpierw możemy utworzyć w programie R strukturę zwaną ramką danych, która będzie przechowywać wartości zmiennej niezależnej, dla której chcemy wygenerować prognozę. Na przykład, chcielibyśmy wygenerować przewidywany zwrot z akcji Coca-Coli, biorąc pod uwagę, że zwrot z indeksu S&P 500 wynosi 6. Do tego celu używamy polecenia R o nazwie `predict()`.

Aby wygenerować prognozę dla równania regresji liniowej o nazwie *model*, używając ramki danych, w której wartość indeksu S&P 500 wynosi 6, polecenia R będą następujące

```
a <- data.frame(SP500 = 6)
predict(model, a)
```

```
       1
4.839063
```

Wynik polecenia `predict` wskazuje, że przewidywany zwrot z akcji Coca-Cola wyniesie 4,8%, podczas gdy zwrot z indeksu S&P 500 wyniesie 6%.

Możemy rozszerzyć tę analizę, aby wygenerować 95% przedział ufności dla prognozy tego wyniku, używając następującej opcji do polecenia `predict`:

```
predict(model, a, interval="predict")
```

```
       fit        lwr     upr
1 4.839063 0.05417466 9.62395
```

W związku z tym 95-procentowy przedział prognozy dla zwrotu z inwestycji w Coca-Colę wynosi (0,05%, 9,62%), gdy zwrot z inwestycji w indeks S&P 500 wynosi 6%. Inaczej mówiąc, gdy zwrot z S&P 500 będzie wynosił 6%, to zwrot z akcji Coca-Cola na 95% będzie w przedziale od 0,05% do 9,62%.

8.2 Modele regresji wielokrotnej z wykorzystaniem języka R

R zawiera również wiele narzędzi, które pozwalają przeprowadzać regresję wielokrotną, w której zmienna zależna jest przewidywana na podstawie więcej niż jednej zmiennej niezależnej, co jest bardziej realistyczną sytuacją w prawdziwej sytuacji. Na przykład, możemy uzyskać lepszy model prognozowania miesięcznego zwrotu z akcji Coca-Coli, jeśli weźmiemy pod uwagę nie tylko miesięczny zwrot z indeksu S&P500, ale również miesięczną sprzedaż produktów Coca-Coli.

Oto kilka przykładów, w których model regresji wielokrotnej może zapewnić lepszy model prognozowania w porównaniu z modelem regresji z tylko jedną zmienną niezależną:

1. Wynagrodzenia pracowników można prognozować na podstawie stażu pracy i poziomu wykształcenia.

2. Ceny nieruchomości można prognozować na podstawie powierzchni domu, liczby sypialni i liczby łazienek.

Ogólna postać modelu regresji wielokrotnej jest następująca:

$$\hat{y} = a + b_1 \cdot x_1 + b_2 \cdot x_2 + b_3 \cdot x_3 + ... + b_n \cdot x_n$$

gdzie:

x1, x2, x3, ..., xn są zmiennymi niezależnymi; b1, b2, b3, ..., bn są współczynnikami regresji, gdzie każdy współczynnik jest wielkością zmiany w y, gdy przyjmiemy, że dany x_i wzrośnie o 1 jednostkę, a pozostałe zmienne niezależne pozostaną niezmienne. a jest wartością przecięcia linii regresji z osią y (ang. *intercept*), czyli wartością y, gdy wszystkie wartości $x_i = 0$.

Przypomnijmy sobie z poprzedniego przykładu, że format analizy regresji liniowej w języku R, gdy występuje tylko jedna zmienna niezależna, wyglądał następująco:

```
> model <- lm(y ~ x)
```

Aby „dodać” dodatkowe zmienne niezależne do modelu regresji wielokrotnej, stosujemy następujący format:

```
> model <- lm(y ~ x1 + x2 + x3)
```

gdzie x1, x2, x3 są zmiennymi niezależnymi.

Przykład:

Użyj języka R do stworzenia modelu regresji wielokrotnej w celu przewidzenia ceny domu na podstawie zmiennych niezależnych, takich jak powierzchnia i liczba pokoi, w oparciu o poniższy zestaw danych. Następnie użyj modelu regresji wielokrotnej do przewidzenia ceny domu o powierzchni 300 metrów kwadratowych i 3 pokojach

tab. 6: Ceny domów w oparciu o powierzchnię i liczbę pokoi

Cena domu (y) w $	**Powierzchnia** (x1) w m2	**Liczba pokoi** (x2)
466000	466,8	6
355000	319,6	5
405900	399,8	5
415000	402,2	5
206000	183,4	2
462000	466,8	6
290000	265,0	3

Najpierw utwórz trzy wektory w języku R, po jednym dla ceny domu, powierzchni i liczby pokoi. Następnie uruchom model regresji wielokrotnej za pomocą znanego już Ci polecenia `lm`:

```
model <- lm(cena ~ powierzchnia + pokoje)
summary(model)
```

```
Call:
lm(formula = cena ~ powierzchnia + pokoje)

Residuals:
      1       2       3       4       5       6       7
-1704.6  1438.4  -606.9  6908.7 -6747.4 -5704.6  6416.5
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  57739.89    9529.69   6.059  0.00375 **
powierzchnia   660.17      89.24   7.398  0.00178 **
pokoje       16966.54    6283.18   2.700  0.05408 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6563 on 4 degrees of freedom
Multiple R-squared:  0.9968,    Adjusted R-squared:  0.9953
F-statistic: 630.6 on 2 and 4 DF,  p-value: 9.996e-06
```

W wynikach programu R należy zwrócić uwagę na kolumnę „Estimate” (Szacunki), która zawiera szacunki współczynników i punkt przecięcia z osią y.

Punkt przecięcia z osią y wynosi 57740 (zaokrąglone do najbliższej liczby całkowitej).

Współczynnik zmiennej „powierzchnia” wynosi 660.

Współczynnik zmiennej „sypialnie” wynosi 16967.

Na podstawie tych wartości model regresji wielokrotnej wygląda następująco (wartości przybliżone, zaokrąglone):

$$\hat{y} = 57740 + 66 \cdot x1 + 16967 \cdot x2$$

Teraz możemy użyć modelu regresji wielokrotnej do przewidzenia ceny domu o powierzchni 300 metrów kwadratowych i 3 pokojach, ustawiając x1 = 300 i x2 = 3, w następujący sposób:

$$\hat{y} = 57\,740 + 660 \times 300 + 16\,967 \times 3 \approx 306\,641$$

Prognoza (powierzchnia = 300, pokoje = 3):

```r
predict(model, newdata = data.frame(powierzchnia = 300, pokoje = 3), interval = "prediction")
```

W ten sposób wiemy, że przewidywana cena domu o powierzchni 300 metrów kwadratowych i 3 pokojach wynosi około 306 641 dolarów.

---

9 Dalsze zasoby

- *Learning Statistics with R*: https://learningstatisticswithr.com/
- YaRrr! *The Pirate's Guide to R*: https://bookdown.org/ndphillips/YaRrr/
- R code best practices: https://www.r-bloggers.com/2018/09/r-code-best-practices/
- Dokumentacja pakietów: https://www.rdocumentation.org
- Społeczność: https://stackoverflow.com/questions/tagged/r

10 Zadanie dla chętnych

10.1 Ćwiczenia – wizualizacja

1. Wejdź na: http://awp.diaart.org/km/surveyresults.html
2. Wybierz dowolny kraj i pytanie z odpowiedziami procentowymi (wykres kołowy).
3. Stwórz wektor nazwanych wartości (kategorie + procenty).
4. Znajdź liczbę respondentów i przelicz procenty na wartości bezwzględne.
5. Narysuj wykres słupkowy – każda kategoria innym kolorem, dodaj tytuł.
6. Udostępnij kod i wykres (w Classroom).

10.2 Ćwiczenia – rozkłady prawdopodobieństwa

1. Oblicz prawdopodobieństwo uzyskania co najmniej 4 orłów w 10 rzutach monetą (przyjmij sprawiedliwą monetę, gdzie p = 0.5).
2. Wygeneruj 100 liczb z rozkładu normalnego o średniej 10 i odchyleniu 2, narysuj histogram wyników.
3. Sprawdź, jak zmienia się prawdopodobieństwo w rozkładzie normalnym przy różnych wartościach średniej i odchylenia.

10.3 Ćwiczenia – regresja

1. Stwórz prosty model regresji liniowej dla dowolnych danych (np. wzrost, waga znajomych i wielkość stopy).
2. Oblicz współczynnik korelacji dla dwóch wektorów liczbowych.
3. Przeprowadź predykcję dla nowej wartości zmiennej niezależnej.

Powodzenia!

11 Słowniczek

Glossarium – kluczowe pojęcia

Wektor – jednowymiarowa struktura danych w R, zawiera elementy tego samego typu.

Macierz – dwuwymiarowa tablica danych o tej samej klasie.

Ramka danych (data.frame) – tabela, w której kolumny mogą mieć różne typy danych.

Lista – kolekcja obiektów dowolnego typu i długości.

Faktor – zmienna kategorialna z określonymi poziomami.

Rozkład prawdopodobieństwa – opisuje, z jakim prawdopodobieństwem zmienna losowa przyjmuje określone wartości.

Regresja liniowa – metoda statystyczna do modelowania zależności między zmienną zależną a jedną lub wieloma niezależnymi.

Korelacja – miara siły i kierunku związku między dwiema zmiennymi.

Histogram – wykres przedstawiający rozkład częstości wartości zmiennej liczbowej.

Wykres słupkowy – wykres do prezentacji danych kategorycznych.

Wykres punktowy – wykres do wizualizacji zależności między dwiema zmiennymi liczbowymi.

Odchylenie standardowe – miara rozrzutu wartości wokół średniej.

Wariancja – średnia kwadratów odchyleń od średniej.

Przedział ufności – zakres wartości, w którym z określonym prawdopodobieństwem znajduje się prawdziwy parametr populacji.

Polecenia

Podstawy `x <- 5` przypisanie; `print(x)` wypis; `rm(x)` usuń; `ls()` lista obiektów; `?nazwa` pomoc; `sessionInfo()` informacje o sesji.

Operatory `+ - * / ^` potęgowanie, `%%` modulo, `%/%` dzielenie całkowite, `== != < <= > >=`, `&` / `&&` logiczne AND, `|` / `||` OR, `!` negacja.

Typy i brakujące dane `is.na(x)`, `anyNA(x)`, `as.numeric`, `as.factor`, `factor(x, levels=...)`.

Struktury danych - Wektor: `c(1,2,3)` - Macierz: `matrix(1:9, nrow=3)` - Data frame: `data.frame(a=1:3, b=c("x","y","z"))` - Lista: `list(num=1, txt="a")` - Faktor: `factor(c("A","B"))`

Indeksowanie i filtrowanie `x[3]`, `x[ x>0 ]`, `df$kolumna`, `df[ , "kolumna"]`, `df[1:2, ]`, `list_obj$name`.

Statystyki opisowe `mean(x)`, `median(x)`, `sd(x)`, `var(x)`, `summary(x)`, `IQR(x)`, `quantile(x, 0.9)`, `weighted.mean(v,w)`.

Agregacje (base) `tapply(x, g, mean)`, `aggregate(x ~ g, data=df, mean)`.

Wizualizacja (base) `plot(x,y)`, `hist(x)`, `boxplot(x)`, `barplot(tab)`, `abline(model)`.

Wizualizacja (ggplot2, skrót) `library(ggplot2) ggplot(df, aes(x,y)) + geom_point()`

Rozkłady Losowanie: `rnorm(n, m, sd)`, `rbinom(n, size, prob)` Gęstość: `dnorm(x, m, sd)` Dystrybuanta: `pnorm(q, m, sd)` Kwantyl: `qnorm(p, m, sd)` (analogicznie: d/p/q/r + nazwa rozkładu)

Regresja `model <- lm(y ~ x1 + x2, data=df) summary(model) predict(model, newdata=..., interval="prediction")` Reszty: `residuals(model)`; dopasowania: `fitted(model)`.

Zapisywanie `saveRDS(obj, "plik.rds"); readRDS("plik.rds") write.csv(df, "dane.csv", row.names=FALSE)`

Częste pułapki - Użycie `=` zamiast `<-` (technicznie działa w wielu miejscach, ale nie zawsze a znak `<-` jest powszechną konwencją)
- Literówki w nazwach obiektów (*case-sensitive*)
- Zapomniane przecinki w `data.frame()`
- Mieszanie typów w wektorze (wszystko stanie się `character`)
- Brak obsługi NA: użyj `na.rm=TRUE` np.: (`mean(x, na.rm=TRUE)`).

Powodzenia w dalszej pracy z R!