

## Instrukcja do przeprowadzenia prostej analizy statystycznej w środowisku R

### Spis treści

Instrukcja do przeprowadzenia prostej analizy statystycznej w środowisku R.....	1
Wstęp .....	2
Część I .....	2
Instalacja R.....	2
Instalowanie pakietów .....	3
Pomoc.....	4
Proste obliczenia w R.....	4
Zmienne.....	5
Obiekty .....	5
Wektory i macierze.....	5
Odczytywanie z plików .....	7
Statystyki opisowe dla zmiennych.....	7
Graficzne statystyki opisowe .....	8
Popularne rozkłady zmiennych losowych .....	12
Standaryzacja zmiennych .....	13
Podstawy regresji liniowej.....	14
Testowanie zgodności .....	15
Testy parametryczne .....	16
Część II. Ćwiczenie praktyczne. ....	17

## Wstęp

R to nazwa języka programowania, platformy programistycznej jak i całego projektu, w ramach którego rozwijane jest zarówno środowisko jak i język. R bardzo często jest nazywany pakietem statystycznym, ponieważ dostępnych jest szereg pakietów i funkcji wykorzystywanych do zastosowań statystycznych. Okazuje się, że R znajduje znacznie więcej zastosowań: automatyczne generowanie raportów, wysyłanie maili, renderowanie trójwymiarowych animacji.

R jest projektem opartym o licencję GNU GPL, wyposażonym w bardzo szczegółową dokumentację dostępną w Internecie. Język R jest językiem interpretowanym, korzystanie z niego sprowadza się do podania szeregu komend, które mają zostać kolejno wykonane. Komendy mogą być zebrane w formie skryptu, który zostaje następnie wykonywany krok po kroku.

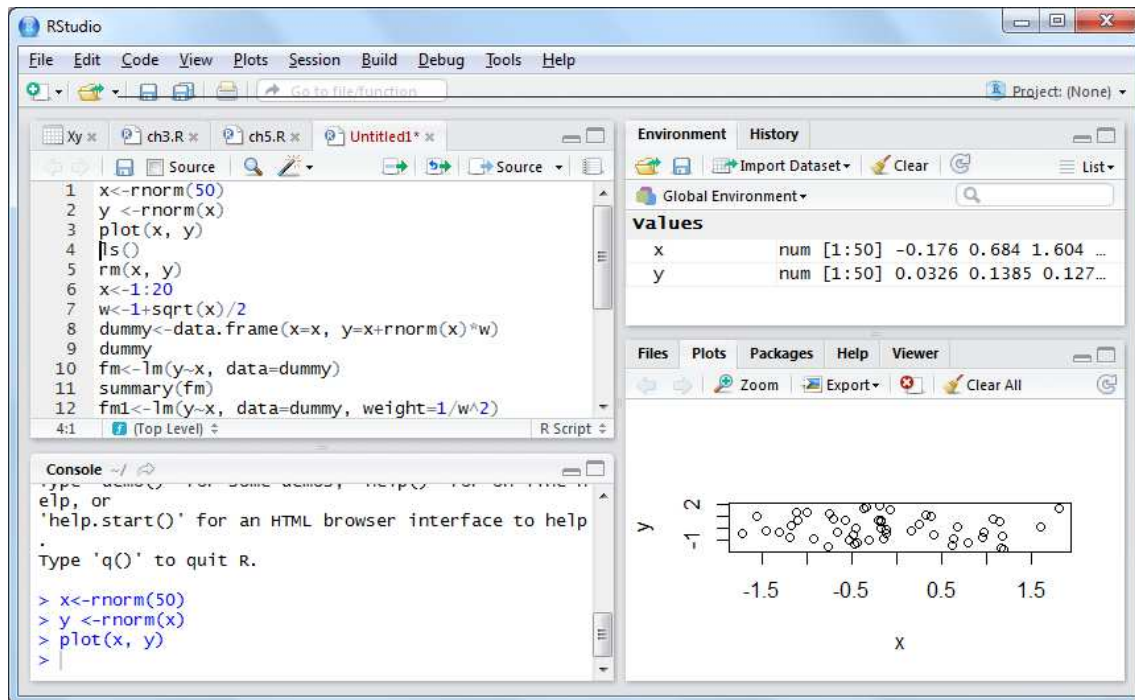
Instrukcja składa się z 2 części: teoretycznej, w której opisane są przydatne do wykonania prostej analizy statystycznej funkcje R, a także praktycznej, w której należy te funkcje wykorzystać do analizy danych. W części teoretycznej wypisywane są przykładowe komendy – najlepiej czytając instrukcję wpisywać je bezpośrednio do edytora R i sprawdzać, jaki dają wynik. W instrukcji opisano głównie te argumenty funkcji, które będą istotne w trakcie wykonywania części praktycznej. Aby dowiedzieć się o każdej z nich więcej, odsyłam do pomocy (rozd. Pomoc) i do wypisanych w instrukcji stron internetowych.

## Część I

### Instalacja R

Dla większości systemów operacyjnych pakiet R jest dostępny zarówno w postaci źródłowej jak i skompilowanej. Najlepiej skorzystać ze skompilowanego pliku instalacyjnego, który można pobrać z jednego z serwerów, których lista znajduje się na stronie: <http://cran.r-project.org/mirrors.html>.

Do tworzenia i edytowania skryptów w R służy specjalne edytory, przykładem może być RStudio. Jego instalacja jest możliwa na systemie Windows, Linux oraz MacOS. Można go pobrać ze strony: [www.rstudio.com](http://www.rstudio.com). Edytor umożliwia wykonywanie pojedynczych komend wpisywanych prosto do konsoli, albo wykonywanie całych skryptów przygotowanych uprzednio w oknie skryptowym. Wygląd edytora jest przedstawiony na rys. 1.



**Rys. 1** Przykładowy wygląd paneli RStudio. W lewym górnym rogu znajduje się okno do edycji skryptów, w lewym dolnym konsola, bezpośrednio do której można wpisywać polecenia, w prawym górnym rogu znajduje się okno podglądu zmiennych znajdujących się w pamięci, a w prawym dolnym rogu okno, w którym wyświetlane są pliki pomocy oraz wykresy.

Po uruchomieniu RStudio w konsoli wyświetla się znak zachęty `>` do wprowadzania kolejnych poleceń. Jeśli nowa linia rozpoczyna się od znaku `+` oznacza to, że polecenie wpisane w poprzedniej linii nie zostało jeszcze zakończone i platforma czeka na dalszą jego część.

### Instalowanie pakietów

Podstawowy zbiór bibliotek R pozwala wykonać szereg analiz. Jednak może się zdarzyć, że brakuje jakiegoś pakietu. Instalowanie pakietów odbywa się poprzez wpisanie komendy `install.packages()`. Poniższe polecenie instaluje pakiet o nazwie „Rcmdr” wraz z pakietami zależnymi, które są niezbędne do jego działania.

```
install.packages("Rcmdr", dependencies = TRUE)
```

Po zainstalowaniu pakietu wszystkie funkcje z nim związane zostają zapisane na dysku twardym komputera, jako podkatalogi katalogu library. Aby móc skorzystać z zainstalowanych funkcja należy włączyć odpowiedni pakiet. Włączenie wykonuje się poprzez instrukcję `library()`:

```
library(Rcmdr)
```

wpisując poniższe komendy, można teraz uruchomić kilka demonstracji, prezentujących możliwości R:

```
demo(persp)
demo(graphics)
```

oraz przykłady grafik trójwymiarowych:

```
library(rgl)
```

```
demo(rgl)
```

## Pomoc

Funkcja `help()` – wyświetla stronę system pomocy R, na której znajdują się szczegółowe opisy poniższych funkcji oraz komend:

<code>help(„nazwaFunkcji”)</code> lub <code>?nazwaFunkcji</code>	Wyświetlenie strony z pomocą dla funkcji określonej przez <i>nazwaFunkcji</i> . W ten sam sposób można uzyskać opisy poszczególnych pakietów;
<code>args(„nazwaFunkcji”)</code>	Wyświetlenie listy argumentów <i>nazwaFunkcji</i> .
<code>apropos(slowo)</code> lub <code>find(slowo)</code>	Wypisanie listy funkcji i obiektów, które w nazwie zawierają <i>slowo</i> .
<code>example(„nazwaFunkcji”)</code>	Wyświetlenie skryptu z przykładami dla <i>nazwaFunkcji</i> .
<code>help.search(„slowoKluczowe”)</code>	Wyświetlenie listy funkcji, w których znajduje się <i>slowoKluczowe</i> .

W Internecie znajdują się podręczniki i wiele materiałów poświęconych R. Przydatne strony:

- <http://cran.r-project.org/manuals.html>
- [www.r-bloggers.com](http://www.r-bloggers.com)
- [www.r-project.org/doc/bib/R-books.html](http://www.r-project.org/doc/bib/R-books.html)
- <http://stats.stackexchange.com>
- <http://stackoverflow.com/questions/tagged/r>
- <http://cran.r-project.org/faqs.html>

Chcąc wyszukać informacji na temat R w wyszukiwarce internetowej, dobrym hasłem wyszukiwania będzie “R CRAN”, CRAN to skrót pochodzący od nazwy *Comprehensive R Archive Network*.

## Proste obliczenia w R

RStudio może służyć jako bardziej zaawansowany kalkulator. Podstawowe operatory arytmetyczne, logiczne i trygonometryczne dostępne są w pakiecie `base`. Aby wyznaczyć wynik prostych operacji matematycznych wystarczy wpisać do linii komend któreś z poniższych poleceń:

```
2+2
```

```
2^10-1
```

```
1/5
```

```
log(1024,2)
```

Funkcje trygonometryczne:

```
sin(pi/2)
```

Symbol Newtona:

```
choose(6,2)
```

Operacje na liczbach zespolonych:

```
sqrt(-17+0i)
```

## Zmienne

Zmienne przechowują wprowadzone dane lub wyniki przeprowadzonych operacji matematycznych. Do wartości przechowywanych przez zmienne odwołujemy się poprzez podanie nazwy zmiennej. Przypisanie wartości do zmiennych:

```
A <- 4  
B <- 2*6  
C <- A/B
```

## Obiekty

Wszystko, z czym mamy do czynienia w R to obiekty, dzielimy je na kilka typów:

- Liczbowy;
- Czynniki;
- Znakowy;
- Logiczny;
- Wektor elementów;
- Lista;
- Macierz;
- Ramka danych;
- Typ funkcyjny.

## Wektory i macierze

Jednym z częściej wykorzystywanych typów danych są wektory. Konstruowanie wektora przebiega w prosty sposób:

```
wektor <- c(1, 2, 3, -4, 7, -8)  
wektor <- 1:10
```

Na wektorach również można wykonywać operacje matematyczne:

```
wektor^2  
wektor-2
```

Aby połączyć dwa wektory, podobnie jak przy budowaniu pojedynczego wektora wykorzystuje się komendę `c()`:

```
c(wektor, 2:4, 9, wektor)
```

Można operować na wektorze wartości logicznych. Po wpisaniu następującej komendy:

```
wektor>0
```

Wynikiem będzie wektor składający się z wartości logicznych „TRUE” lub „FALSE” w zależności od tego, czy dany element wektora `wektor` był większy czy mniejszy lub równy 0. Można się odwoływać do wartości poszczególnych elementów wektora. Pierwszą wartość wektora wyświetlimy wpisując następującą komendę:

```
wektor[1]
```

Chcąc wyświetlić wartości pierwszą trzecią i piątą wpisujemy:

```
wektor[c(1,3,5)]
```

Przypisanie do zmiennej A wartości wektora od trzeciej do piątej wykonamy wpisując następującą komendę:

```
A <- wektor[3:5]
```

Wartości brakujące są oznaczane jako „NA” (od ang. *not available*). Aby obliczyć parametr bez uwzględniania „NA”, na przykład wartość średnią, należy do komendy dopisać jako argument `na.rm=TRUE`:

```
mean(wektor, na.rm=TRUE)
```

Przydatną funkcją do operacji na wektorach jest funkcja `which()`, której wynikiem są indeksy elementów wektora spełniających dany warunek. Gdy wektor przyjmie wartości:

```
wektor <- c(10, 12, 9.5, -2 11 -2)
```

to komenda:

```
which(wektor == 9.5)
```

powinna zwrócić wartość 3.

Drugim ważnym rodzajem zmiennych są macierze. Chcąc zbudować macierz o wymiarach 2 na 3, wypełnioną samymi zerami należy wpisać następującą komendę:

```
macierz <- matrix(0, 2, 3)
```

Z kolei wpisanie poniższej komendy:

```
macierz <- matrix(1:6, 2, 3)
```

Spowoduje powstanie macierzy o wymiarach 2 na 3 o wpisanych kolejno liczbach od 1 do 6. Aby wyświetlić pojedyncze kolumny lub wiersze należy wykorzystać następujące komendy – do wyświetlenia drugiej kolumny:

```
macierz[,2]
```

Aby wyświetlić drugi wiersz:

```
macierz[2,]
```

Na macierzach również można wykonywać operacje. Po określeniu zdefiniowaniu zmiennych A i B jako macierzy 2 na 2 zawierających wartości od 1 do 4:

```
A <- B <- matrix(1:4, 2, 2)
```

Mnożenie poszczególnych elementów macierzy przez siebie wykonuje się w następujący sposób:

```
A*B
```

Mnożenie macierzowe można natomiast wykonać tak:

```
A%*%B
```

Pełną listę operatorów można znaleźć na stronie:

<http://cran.r-project.org/doc/manuals/r-release/R-lang.html#Operators>

### Odczytywanie z plików

W R dostępnych jest wiele funkcji umożliwiających wczytanie danych z pliku. Funkcje te posiadają szereg argumentów, przy pomocy których można określić rodzaj kodowania, znak separatora, znak dziesiętny, typ odczytywanych danych itd.

Aby otworzyć plik tekstowy, w którym dane zapisane są w postaci tabelarycznej można wykorzystać funkcje `read.table()`. Natomiast zapis do pliku tekstowego można wykonać przy pomocy funkcji `write.table()`. Jeśli plik zawiera w pierwszym wierszu nazwy kolumn, a separatorem jest tabulator, to otwarcie takiego pliku można wykonać wpisując poniższą komendę:

```
dane <- read.table("nazwaPliku", header=TRUE, sep="\t")
```

Dane często są zapisywane w formacie .csv. Otwarcie pliku z takiego formatu można wykonać na kilka sposobów, jednym z nich jest wpisanie następującej komendy:

```
dane <- read.csv("nazwaPliku", header=TRUE, sep = ",")
```

Ścieżka do pliku podawana jako argument funkcji `read.table()` może być również adresem URL – wtedy nastąpi otwarcie danych ściągniętych bezpośrednio z sieci.

Po wczytaniu danych można je podglądnąć, wykorzystując komendę:

```
view(dane)
```

A także wczytać do pamięci wszystkie zmienne znajdujące się w pliku, wykonując następującą komendę:

```
attach(dane)
```

Jeśli plik z danymi zawierał nazwy zmiennych zapisane w pierwszym wierszu, to po wykonaniu komendy `attach(dane)` do każdej z tych zmiennych można się odwoływać po jej nazwie. Jeśli natomiast w pliku nie było tych nazw, to po wykonaniu komendy `attach(dane)` do poszczególnych zmiennych można się odwoływać, wykorzystując symbole `V1`, `V2`, `V3` itd.

### Statystyki opisowe dla zmiennych

Pakiet `base` zawiera szereg funkcji umożliwiających wyliczenie prostych statystyk opisowych dla wybranej zmiennej:

<code>Range(V1)</code>	Zakres wartości, jakie przyjmuje zmienna <code>V1</code> ;
------------------------	--

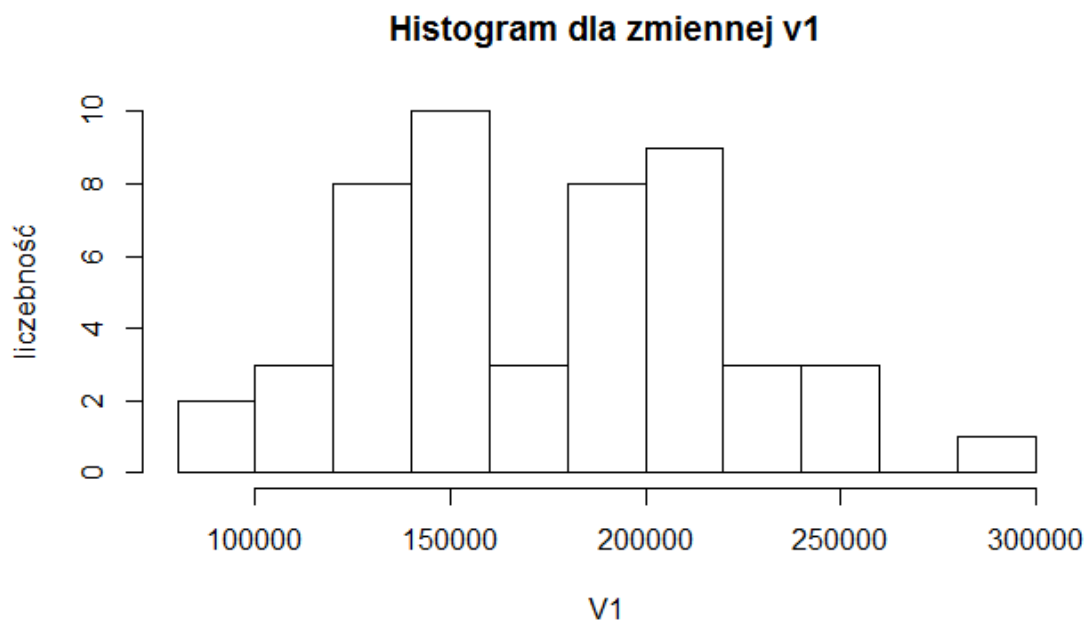
IQR(V1)	Rozstęp międzykwartylowy dla zmiennej V1;
Mean(V1)	Wartość średnia zmiennej V1;
Median(V1)	Mediana zmiennej V1;
Sd(V1)	Odchylenie standardowe zmiennej V1;
Quantile(V1, c(0.1, 0.25, 0.75, 0.9))	Wybranej kwantyle dla zmiennej V1;
Cor(dane[c(1,4,7,9)])	Macierz korelacji dla zmiennych 1, 4, 7 oraz 9;
Summary(V1)	Podsumowanie statystyk opisowych dla zmiennej V1

### Graficzne statystyki opisowe

Aby wykonać histogram dla wybranej zmiennej należy użyć funkcji `hist()`. Pomijając elementy graficzne deklaracja tej funkcji może wyglądać w następujący sposób:

```
hist(V1,10,main="Histogram dla zmiennej V1", ylab="Liczebności")
```

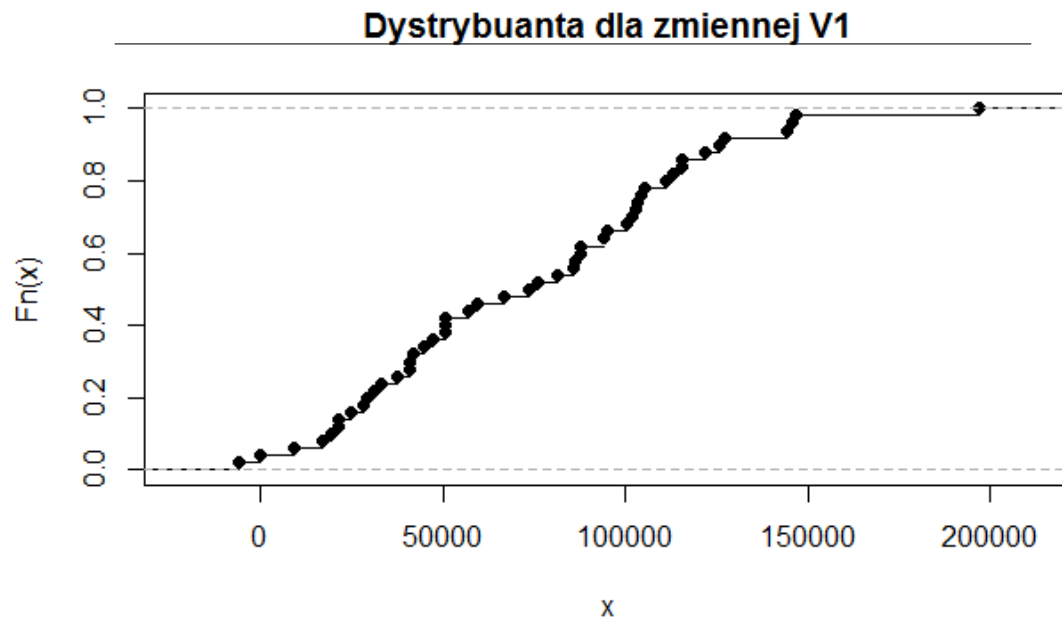
występująca po zmiennej liczba „10” określa liczbę klas, dla których ma powstać histogram. Wpisując w to miejsce argument `breaks="Sturges"` albo `"Scott"` albo `"FD"` albo `"Freedman-Diaconis"` dokonamy wyboru algorytmu, który automatycznie sam wyznaczy liczbę klas histogramu.



Dla każdej zmiennej można też wyznaczyć funkcję dystrybuanty, wykorzystując funkcję `ecdf()`, a także narysować jej wykres:

```
plot(ecdf(V1), main="Dystrybuanta dla zmiennej V1")
```



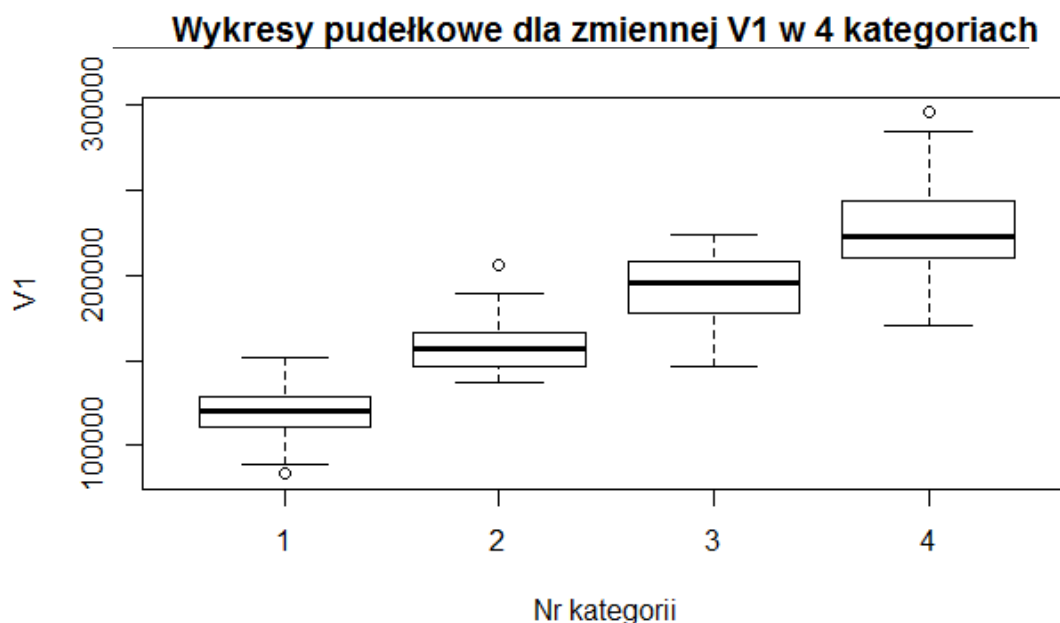


Bardzo przydatną funkcją jest również funkcja `boxplot()`, która umożliwia narysowanie wykresu pudełkowego:

```
boxplot(V1, range=1.5)
```

Często się zdarza, że chcemy porównać wartości jakie dana zmienna przyjmuje w różnych kategoriach. Załóżmy, że kategorie są określone poprzez zmienną V2 jako 1, 2, 3 oraz 4. Aby porównać przy pomocy wykresów pudełkowych rozkłady zmiennej V1 w różnych kategoriach zdefiniowanych w zmiennej V2, można wykorzystać następującą komendę:

```
boxplot(V1~V2, range=1.5, main="Wykresy pudełkowe dla zmiennej V1 w 4 kategoriach", ylab="V1", xlab="Nr kategorii")
```

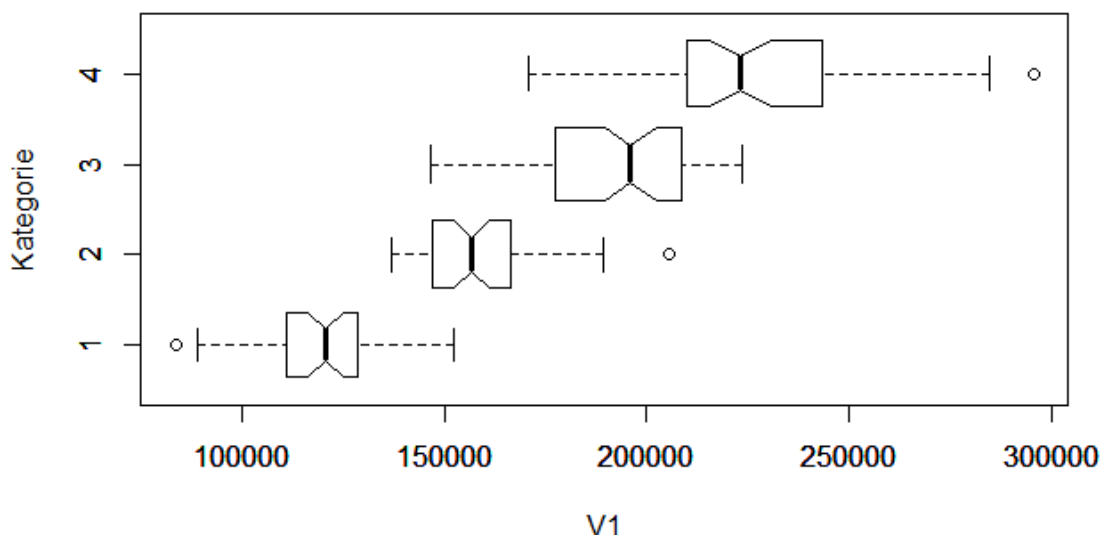


Argument `range=1.5` oznacza, że obserwacje są uznawane za odstające, gdy odstają od kwartyli bardziej niż o  $1,5 \cdot \text{IQR}$ . W wyglądzie wykresów pudełkowych można modyfikować więcej elementów:

```
boxplot(V1~V2, range=1.5, varwidth=TRUE, notch=TRUE, outline=TRUE, horizontal=TRUE, main="Wykresy pudełkowe dla zmiennej V1 w 4 kategoriach", ylab="Kategorie", xlab="V1")
```

Wynikiem tak zadeklarowanej komendy są wykresy pudełkowe przedstawione poniżej. Wcięcia przy medianie oznaczają w tym przypadku 95% przedział ufności dla mediany (`notch=TRUE`), wyświetlane są wartości odstające (`outline=TRUE`), szerokość pudełek jest proporcjonalna do pierwiastka z liczby obserwacji w wektorze (`varwidth=TRUE`), a wykresy są przedstawione w poziomie (`horizontal=TRUE`).

### Wykresy pudełkowe dla zmiennej V1 w 4 kategoriach

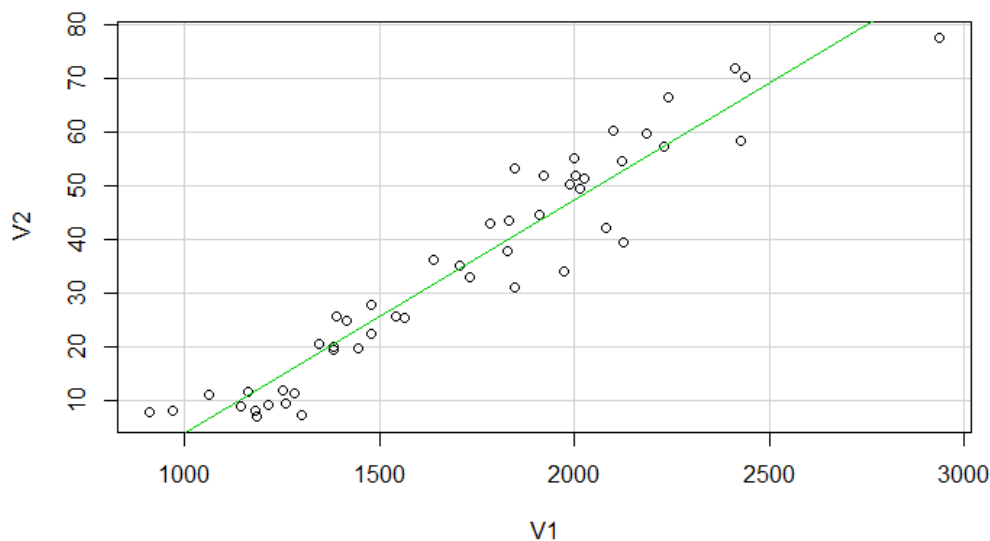


Kolejnym przydatnym wykresem jest wykres rozrzutu, który jest generowany na podstawie komendy `scatterplot()` dostępny w pakiecie `car`. Wykres rozrzutu przedstawia zależności pomiędzy parą zmiennych. Wywołanie wykresu rozrzutu wykonuje się w następujący sposób:

```
scatterplot(V1, V2, smoother=NULL, reg.line=lm, boxplots=FALSE, main="Wykres  
s rozrzutu dla zmiennych V1 i V2")
```

Tak zadeklarowana funkcja spowoduje powstanie wykresu rozrzutu zmiennej V2 względem zmiennej V1 wraz z naniesioną linią regresji liniowej (`reg.line=lm`), bez dopasowania krzywej wygładzonej (`smoother=NULL`) i bez wykresów pudełkowych, które domyślnie rysowane dla obydwu osi (`boxplots=FALSE`).

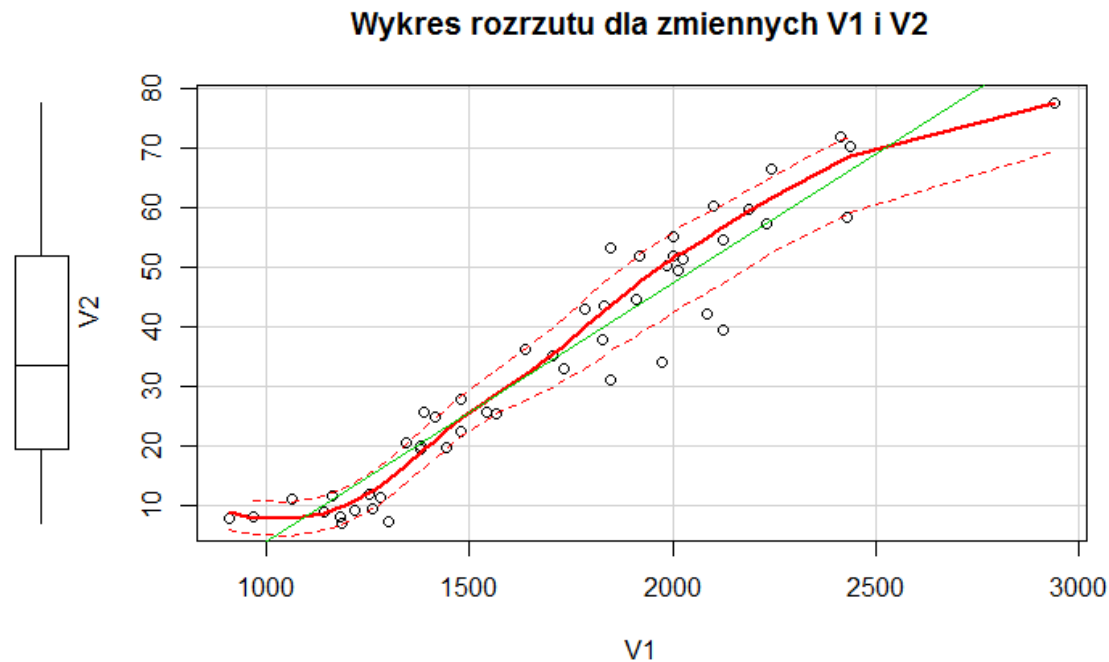
### Wykres rozrzutu dla zmiennych V1 i V2



Z kolei poniższa deklaracja:

```
scatterplot(V1, V2, smoother=loessLine, reg.line=lm, boxplots="y", main="Wykres rozrzutu dla zmiennych V1 i V2")
```

spowoduje dodanie wygładzonej krzywej regresji oraz wykresu pudełkowego przy osi Y:



### Popularne rozkłady zmiennych losowych

Narysujemy rozkład i dystrybucję rozkładu normalnego. W pierwszym kroku definiujemy punkty, w których chcemy wyznaczyć wartości dystrybucji i gęstości prawdopodobieństwa, następnie rysujemy wykres funkcji gęstości prawdopodobieństwa, wykorzystując funkcję `dnorm()` :

```
x <- seq(-4,4,by=0.01)
plot(x, dnorm(x), type="l", lwd=2)
```

na tym samym wykresie chcemy dodać wykres funkcji gęstości, jednocześnie chcemy, aby teraz os y przyjmowała wartości od -0.04 do 1.04:

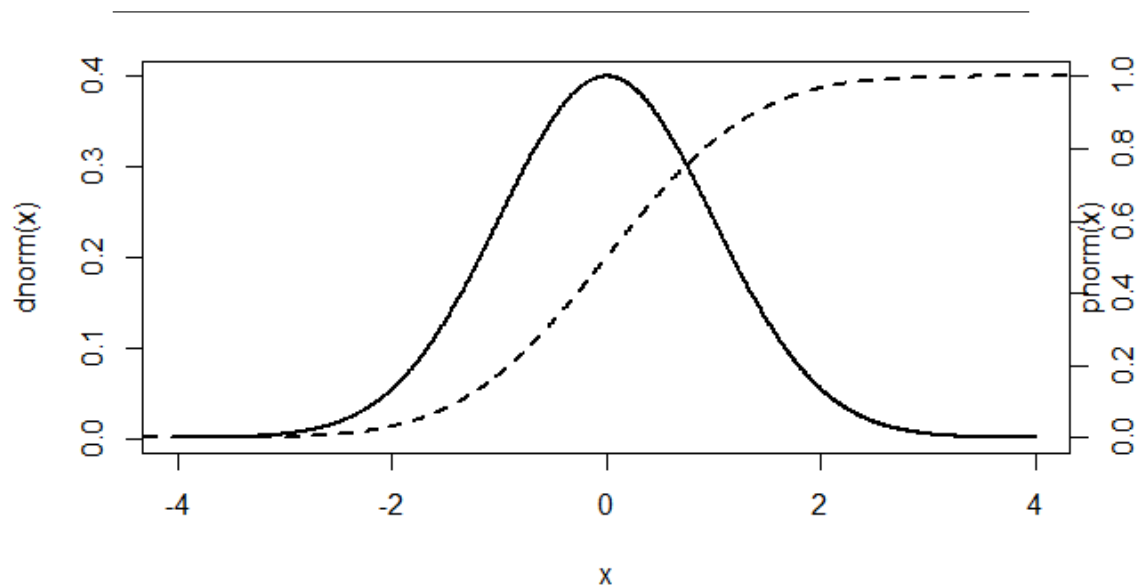
```
par(usr=c(-4, 4, -0.04, 1.04))
```

Dorysowujemy dystrybucję, wykorzystując funkcję `pnorm()`, dla nowych współrzędnych:

```
lines(x, pnorm(x), lty=2, lwd=3)
```

Dodajemy oś y po prawej stronie ze współzrędnymi dla dystrybucji:

```
axis(side=4)
mtext(side=4, "pnorm(x)")
```



Przydatne funkcje do generowania parametrów związanych z rozkładem normalnym o zadanej wartości średniej ( $\text{mean}=0$ ) i odchyleniu standardowym ( $\text{sd}=1$ ) to:

```
dnorm(x, mean=0, sd=1, log=FALSE)
pnorm(x, mean=0, sd=1, lower.tail=TRUE, log.p=FALSE)
qnorm(x, mean=0, sd=1, lower.tail=TRUE, log.p=FALSE)
rnorm(n, mean=0, sd=1)
```

Kolejno funkcje te generują: ( $\text{dnorm}$ ) gęstość prawdopodobieństwa w punktach określonych przez wektor  $x$ , ( $\text{pnorm}$ ) dystrybucję w punktach określonych wektorem  $x$ , ( $\text{qnorm}$ ) kwantyle w punktach określonych przez wektor  $x$ , ( $\text{rnorm}$ )  $n$  losowych wartości. Z innych rozkładów korzysta się w podobny sposób, analogiczne funkcje dla rozkładu t-Studenta to:  $\text{dt}()$ ,  $\text{pt}()$ ,  $\text{qt}()$  oraz  $\text{rt}()$ , dla rozkładu F to  $\text{df}()$ ,  $\text{pf}()$ ,  $\text{qf}()$  oraz  $\text{rf}()$ , a dla Chi-kwadrat  $\text{dchisq}()$ ,  $\text{pchisq}()$ ,  $\text{qchisq}()$  oraz  $\text{rchisq}()$ .

### Standaryzacja zmiennych

Standaryzacja, czyli inaczej skalowanie, polega na odjęciu od każdej wartości zmiennej wartości średniej i podzieleniu przez odchylenie standardowe. Operację tą można wykonać przy pomocy funkcji  $\text{scale}()$ . Przypisanie do zmiennej  $\text{standX}$  standaryzowanych wartości zmiennej  $x$  odbywa się w następujący sposób:

```
standX <- scale(x)
```

## Podstawy regresji liniowej

Funkcją służącą do budowy modelu liniowego w R jest funkcja `lm()`. Jeśli wartość zmiennej V1 chcemy szacować na podstawie zmiennych V2 oraz V4, to funkcja `lm()` dokona dopasowania modelu liniowego, wyznaczając wiele parametrów dla tego modelu.

```
model <- lm(V1~V2+V4, data = dane)
```

Aby wydobyć wartości współczynników równania liniowego można wykorzystać następującą komendę:

```
model$coeff
```

Uzyskamy wtedy następujący wynik:

(Intercept)	V2	V4
1028.85437	22.74417	-31.55056

Z którego można odczytać współczynniki modelu liniowego:

$$V1 = 22.74 \cdot V2 - 31.55 \cdot V4 + 1028.85$$

Aby zobaczyć wszystkie parametry tak powstałego modelu należy wykorzystać funkcję `summary()`:

```
summary(model)
```

W wyniku której uzyskamy następujące podsumowanie parametrów modelu:

```
Call:
lm(formula = V1 ~ V2 + V4, data = dane)

Residuals:
    Min       1Q   Median       3Q      Max
-234.88  -85.70   -8.51   52.29  321.29

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1028.854    99.860   10.303 1.21e-13 ***
V2           22.744     3.007    7.563 1.15e-09 ***
V4          -31.551    54.729   -0.576  0.567
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 131.3 on 47 degrees of freedom
Multiple R-squared:  0.9167,    Adjusted R-squared:  0.9132
F-statistic: 258.7 on 2 and 47 DF,  p-value: < 2.2e-16
```

W jaki sposób interpretować otrzymane wyniki? Oprócz oszacowanych wartości dla współczynników równania liniowego, otrzymujemy tutaj też dla każdego błąd standardowy, wartość statystyki t oraz wynik testu istotności dla danego współczynnika. Jeśli przez  $\beta_i$  oznaczymy wartości kolejnych współczynników równania modelu liniowego, to w teście istotności dla współczynników modelu liniowego testowana hipoteza zerowa dla  $i$ -tego współczynnika wygląda następująco:

$$H_0: \beta_i = 0$$

natomiast hipoteza alternatywna:

$$H_1: \beta_i \neq 0$$

Wyniki testu są podawane jako  $Pr(> |t|)$ , co jest też oznaczane jako p-wartość. Na zadanym poziomie istotności  $\alpha = 0.05$  decyzję o odrzuceniu lub nieodrzuceniu hipotezy zerowej dla danego testu podejmujemy porównując p-wartość z parametrem  $\alpha$ . **Jeśli  $p < \alpha$ , to odrzucamy hipotezę zerową, w przeciwnym wypadku nie mamy podstaw do jej odrzucenia.** Interpretując wyniki otrzymane dla modelu przedstawionego powyżej, dla  $\alpha = 0.05$  możemy odrzucić hipotezy zerowe dla współczynnika Intercept, czyli dla wyrazu wolnego, oraz dla współczynnika stojącego przed zmienną V2. Oznacza to, że ich wartości są istotnie różne od zera. Nie możemy tego natomiast powiedzieć o współczynniku stojącym przed zmienną V4. W wyniku testu istotności nie mamy podstaw do odrzucenia hipotezy zerowej mówiącej o tym, że jest on równy zero, a więc równanie dla modelu powinno wyglądać jednak tak:

$$V1 = 22.74 \cdot V2 + 1028.85$$

Do graficznego przedstawienia tej zależności można wykorzystać funkcję `scatterplot()`, gdzie argument `reg.line=lm`:

```
scatterplot(V2, V1, smoother=NULL, reg.line=lm, boxplots=NULL, main="Wykres
rozrzutu dla zmiennych V1 i V2")
```

Model liniowy w regresji liniowej mogą też tworzyć zmienne jakościowe. Pakiet R sam zmienia ich wartości na wartości liczbowe. Gdyby zmienna V3 przyjmowała wartości „mały”, „średni” oraz „duży”, to pakiet R domyślnie utworzyłby dwie sztuczne zmienne: V3.średni oraz V3.duży, zmienna V3.mały byłaby zmienną referencyjną, zawsze przyjmującą wartość 0. Pozostałe zmienne w zależności od przypadku przyjmowałyby wartości 0 lub 1. Zmienne jakościowe można kodować na różne sposoby, np. wykorzystując funkcje `level()` oraz `reorder()`. Wywołanie funkcji

```
plot(model)
```

wygeneruje szereg wykresów diagnostycznych dla modelu liniowego. Jednak te wykresy pozostają do interpretacji tylko dla osób bardziej zainteresowanych modelami liniowymi. Aby przy pomocy zbudowanego modelu przewidzieć wartości zmiennej V1 dla nowych przypadków, opisanych przez zmienną V2 wystarczy wykorzystać funkcję `predict()`. Przed wywołaniem tej funkcji trzeba zbudować ramkę danych testowych. Jeśli chcielibyśmy przewidzieć wartości zmiennej V1 dla przypadków, dla których zmienna V2 przyjmuje wartości 34 i 65, to taką ramkę danych stworzymy w następujący sposób:

```
daneTest <- data.frame(V2=c(34, 65))
```

Gdyby model liniowy zawierał więcej zmiennych o różnych nazwach, to ramkę tworzylibyśmy w następujący sposób:

```
daneTest2 <- data.frame(V2=c(34, 65), plec=c("Kobieta", "Mężczyzna"))
```

Następnie do przewidzenia wartości dla nowych przypadków opisanych w ramce, należałoby wykorzystać komendę `predict()`:

```
predict(model, newdata=daneTest)
```

### Testowanie zgodności

Polega na testowaniu zgodności rozkładów, najczęściej testuje się zgodność z rozkładem normalnym. Przy testowaniu normalności rozkładu hipoteza zerowa wygląda tak:

$$H_0: F \in \{N(\mu, \sigma): \mu \in R, \sigma \in R_+\}$$

Gdzie jako  $F$  oznaczony jest nieznany rozkład, z którego pochodzą obserwowane wartości. Badamy, czy można przyjąć, że rozkład ten jest rozkładem normalnych o nieznanych parametrach  $\mu$  oraz  $\sigma$ .

Funkcje do testowania normalności dostępne są w pakiecie `nortest`. Przykładami testów, jakie można przeprowadzić przy sprawdzaniu normalności rozkładu jest test Shapiro-Wilka albo test chi-kwadrat Pearsona. W zależności od tego, który test chcemy przeprowadzić wykorzystujemy jedną z funkcji `shapiro.test()` lub `pearson.test()`. Lista komend do przeprowadzenia testu normalności dla zmiennej `V1` została przedstawiona poniżej:

```
shapiro.test(V1)

(pwynik <- shapiro.test(V1)$p.value)

ifelse(pwynik < 0.05,
      "Odrzucamy hipotezę zerową na poziomie istotności 0.05",
      "Nie ma podstaw do odrzucenia hipotezy zerowej na poziomie istotności 0.05")
```

W ten sam sposób przeprowadza się test chi-kwadrat, podstawiając w odpowiednie miejsce funkcję `pearson.test()` zamiast `shapiro.test()`. Każdy z testów daje na wyjściu wartość wyliczonej statystyki testowej oraz p-wartość.

### Testy parametryczne

Jednym z testów parametrycznych jest test wartości średniej, w którym hipoteza przyjmuje następującą postać:

$$H_0: \mu_x = \mu_0$$

W teście tym chcemy sprawdzić, czy wartość oczekiwana populacji  $\mu_x$ , z której pochodzi próba  $x$  będzie wynosiła  $\mu_0$ . Aby przeprowadzić taki test należy wykorzystać funkcję `t.test()` i jako argumenty podać wektor liczb pochodzących z próby  $x$  oraz określić argument `mu`. Dla hipotezy zerowej o postaci:

$$H_0: \mu_x = 0$$

Przeprowadzenie testu parametrycznego dla zmiennej  $x$  w R odbywa się w następujący sposób:

```
t.test(x, mu=0)

t.test(x)$p.value
```

Podobnie jak przy teście zgodności można dodać komentarz:

```
ifelse(pwynik < 0.05,
      "Odrzucamy hipotezę zerową na poziomie istotności 0.05",
      "Nie ma podstaw do odrzucenia hipotezy zerowej na poziomie istotności 0.05")
```

W wyniku testu otrzymujemy nie tylko wartość statystyki testowej, p-wartość ale także 95% przedział ufności dla średniej. Możemy określić, czy hipoteza alternatywna ma być jedno- czy dwustronna.



Wykorzystujemy do tego argument `alternative`, który domyślnie przyjmuje wartość `"two.sided"`:

`alternative="two.sided"`       $H_1: \mu_x \neq 0$

`alternative="less"`       $H_1: \mu_x < 0$

`alternative="greater"`       $H_1: \mu_x > 0$

Gdy chcemy porównać średnie pomiędzy dwoma próbami niezależnymi,  $x$  i  $y$ , hipoteza zerowa wygląda w następujący sposób:

$$H_0: \mu_x - \mu_y = 0$$

Hipotezę alternatywną określamy jak powyżej, a test przeprowadza się w następujący sposób:

```
t.test(x,y)
t.test(x,y)$p.value
```

Dla prób powiązanych test ten przeprowadza się dopisując do funkcji argument `"paired=T"`

```
t.test(x, y, paired=T)
```

Dla funkcji `t.test()` można określić dodatkowo argument `var.equal`. Może on przyjąć wartość `TRUE`, co oznacza, że zakładamy równość wariancji pomiędzy próbami albo `FALSE` (wartość domyślna), która oznacza, że nie jesteśmy pewni tej równości. W drugim przypadku zostanie wprowadzona do specjalna korekta (korekta Welcha), która umożliwi przeprowadzenie testu i interpretację wyników. Można jednak samemu sprawdzić równość wariancji wykorzystując `F test` i umożliwiającą na jego przeprowadzenie funkcję `var.test()`.

## Część II. Ćwiczenie praktyczne.

Zarządzasz danymi w firmie zajmującej się sprzedażą nieruchomości we Wrocławiu. Wczytaj dane `mieszkania.csv` i wykonaj następujące polecenia:

1. Opisz dane. Wyznacz parametry statystyki opisowej dla zmiennych ilościowych oraz liczebności w poszczególnych kategoriach dla zmiennych jakościowych.
2. Narysuj histogramy dla każdej ze zmiennych ilościowych. Pamiętaj o odpowiednim podpisaniu osi.
3. Sprawdź, czy na poziomie istotności  $\alpha = 0.05$  można przyjąć, że zmienne cena i powierzchnia charakteryzują się rozkładami normalnymi. Zapisz hipotezy zerowe i alternatywne, p-wartość oraz komentarz w sprawie odrzucenia lub nie hipotezy zerowej dla każdej ze zmiennych.
4. Na poziomie istotności  $\alpha = 0.05$  zweryfikuj hipotezę zerową mówiącą o tym, że średnia cena mieszkania we Wrocławiu wynosi 180 000 PLN. Zapisz hipotezę zerową i alternatywną, p-wartość oraz komentarz w sprawie odrzucenia lub nie hipotezy zerowej. Wyznacz 95% przedział ufności dla średniej ceny mieszkania we Wrocławiu.
5. Narysuj wykresy pudełkowe dla zmiennych cena i powierzchnia uwzględniając podział na kategorie określone przez zmienne dzielnica oraz typ budynku. Co można powiedzieć o wartościach ceny i powierzchni w tych różnych dzielnicach? A co w różnych typach budynków? Skomentuj wykresy.

6. Dodaj nową zmienną, która będzie zawierała informację na temat ceny za metr kwadratowy. Sporządź wykresy pudełkowe dla ceny za metr dla kategorii określonych przez liczbę pokoi. Skomentuj otrzymane wykresy.
7. Wykonaj test istotności dla średnich: sprawdź, czy można przyjąć, że średnia cena mieszkań w dzielnicy Biskupin jest wyższa od średniej ceny mieszkań w Śródmieściu. Przyjmij poziom istotności  $\alpha = 0.05$ . W razie braku równości pomiędzy wariancjami wykonaj test t z uwzględnieniem poprawki Welcha. Zapisz wszystkie testowane hipotezy zerowe i alternatywne oraz wszystkie niezbędne komentarze.
8. Wykonaj test istotności dla średnich: sprawdź, czy średnia powierzchnia mieszkań w wieżowcach jest mniejsza od powierzchni mieszkań w kamienicach. Przyjmij poziom istotności  $\alpha = 0.05$ . W razie braku równości pomiędzy wariancjami wykonaj test t z uwzględnieniem poprawki Welcha. Zapisz wszystkie testowane hipotezy zerowe i alternatywne oraz wszystkie niezbędne komentarze.
9. Sporządź model regresji liniowej, który umożliwi wyliczenie ceny mieszkania na podstawie jego powierzchni i liczby pokoi. Zapisz równanie modelu liniowego. Które współczynniki równania są istotnie różne od zera? W jaki sposób cena mieszkania zależy od tych dwóch zmiennych?
10. Narysuj wykres rozrzutu dla zmiennych cena i powierzchnia, uwzględniając prostą będącą wizualizacją modelu liniowego z punktu 8 (bez modelu wygładzonego).
11. Sporządź model regresji liniowej, który umożliwi wyliczenie ceny mieszkania na podstawie powierzchni, liczby pokoi, dzielnicy, typu budynku. Które współczynniki równania są istotnie różne od zera? Zapisz równanie modelu liniowego.
12. Do biura nieruchomości przyszło trzech nowych klientów którzy chcą sprzedać mieszkania we Wrocławiu, parametry nowych mieszkań do sprzedaży są opisane w poniższej tabeli.

L.p.	Liczba pokoi	Powierzchnia	Dzielnica	Typ budynku
1.	2	54.3	Krzyki	niski blok
2.	4	70.2	Śródmieście	kamienica
3.	1	22.5	Biskupin	wieżowiec

Wykorzystując model liniowy wyznaczony w punkcie 10 oszacuj i doradź im za jaką cenę mogą sprzedać swoje mieszkania.