ADPS 2025Z — Laboratorium 4

Konrad Jędrzejewski

Przykład 1 – analiza wariancji

W plikach grX.txt, gdzie X jest numerem od 1 do 5, znajdują się dane dotyczące wzrostu [kg] i wagi [cm] studentów z pięciu grup. Wczytaj dane z pierwszego pliku i obejrzyj je:

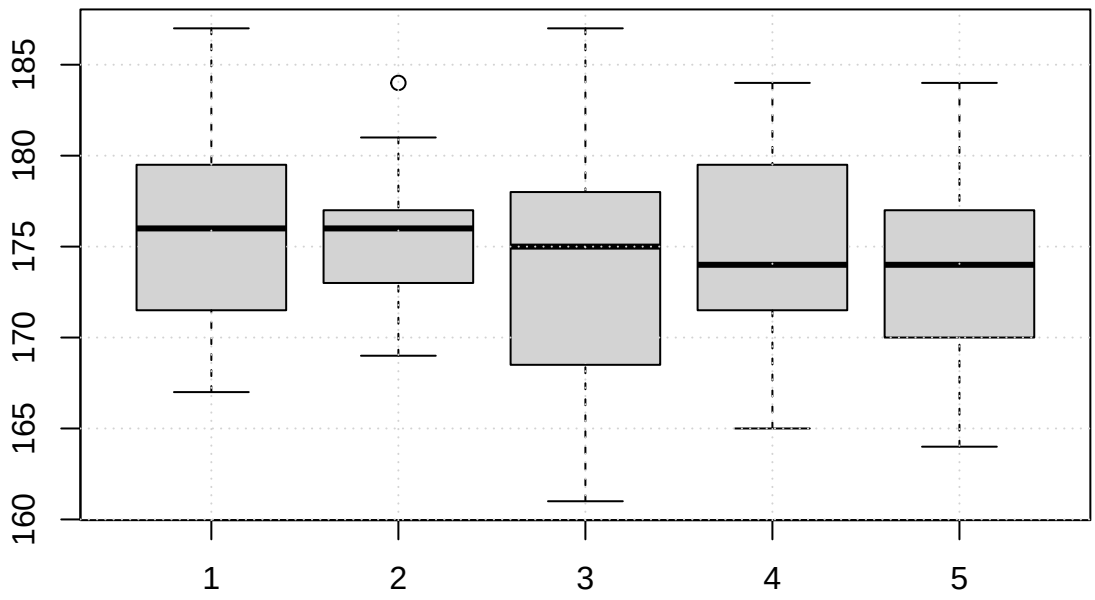
```
gr1 = read.csv('gr1.txt')
```

Wczytaj kolumny "Wzrost" z poszczególnych plików:

```
x1 = read.csv('gr1.txt')$Wzrost
x2 = read.csv('gr2.txt')$Wzrost
x3 = read.csv('gr3.txt')$Wzrost
x4 = read.csv('gr4.txt')$Wzrost
x5 = read.csv('gr5.txt')$Wzrost
```

Wyświetl ich wykres pudełkowy:

```
boxplot(x1, x2, x3, x4, x5)
grid()
```



Przygotuj dane na potrzeby funkcji aov():

```
dane_anova = data.frame( dane = c(x1, x2, x3, x4, x5),
          proba = rep( c('x1', 'x2', 'x3', 'x4', 'x5'),
                  times = c(length(x1), length(x2), length(x3), length(x4), length(x5))) )
```

Przeprowadź analizę wariancji przy założeniu normalności rozkładów:

```
aov_res = aov(dane~proba, data = dane_anova)
summary(aov_res)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## proba         4     58   14.51   0.513  0.726
## Residuals   116   3282   28.29
```

Przeprowadź analizę wariancji bez zakładania normalności rozkładów korzystając z testu Kruskala-Wallisa:

```
kruskal.test(dane~proba, dane_anova)
```

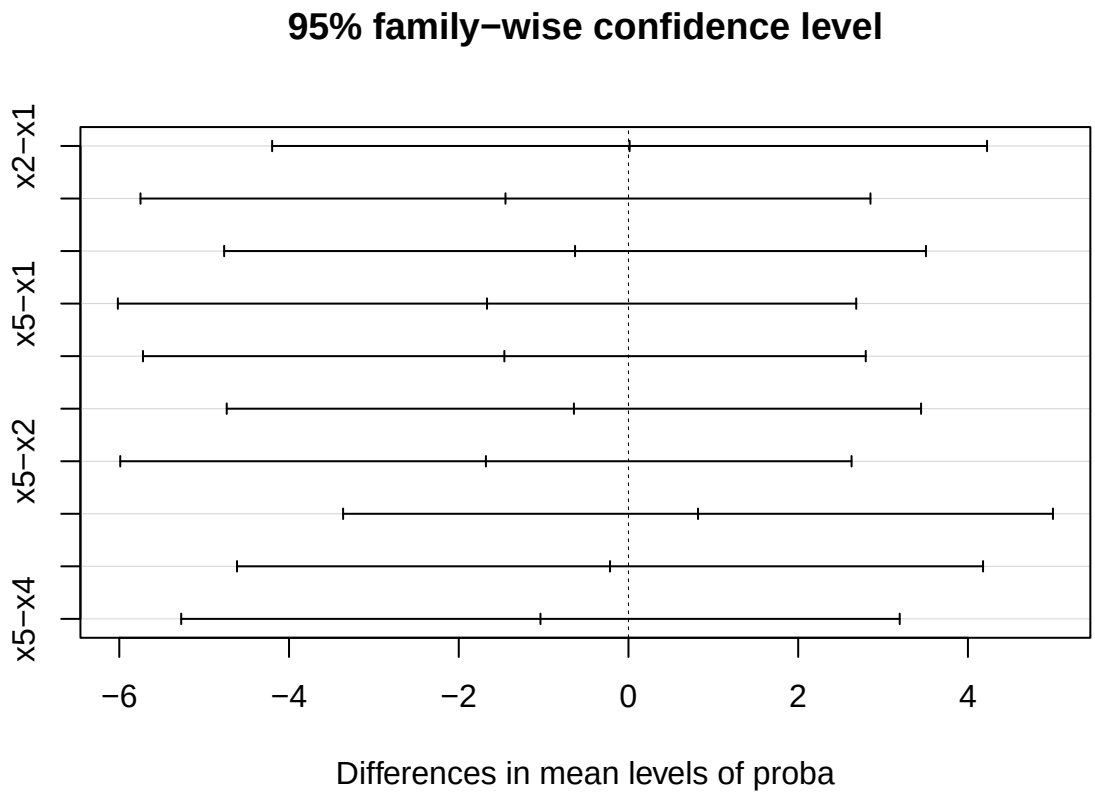
```
##
##  Kruskal-Wallis rank sum test
##
## data:  dane by proba
## Kruskal-Wallis chi-squared = 1.442, df = 4, p-value = 0.8369
```

Korzystając z metody Tukeya sprawdź, czy dla którejś z prób jej wartość średnia odbiega od wartości średnich w pozostałych próbach:

```
Tukey_res = TukeyHSD(aov_res)
print(Tukey_res)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = dane ~ proba, data = dane_anova)
##
## $proba
##              diff       lwr      upr     p adj
## x2-x1  0.01333333 -4.198781 4.225447 1.0000000
## x3-x1 -1.44927536 -5.750153 2.851602 0.8831820
## x4-x1 -0.62962963 -4.764627 3.505368 0.9932912
## x5-x1 -1.66666667 -6.017172 2.683838 0.8256799
## x3-x2 -1.46260870 -5.721185 2.795967 0.8758292
## x4-x2 -0.64296296 -4.733944 3.448018 0.9924276
## x5-x2 -1.68000000 -5.988690 2.628690 0.8162210
## x4-x3  0.81964573 -3.362671 5.001962 0.9825810
## x5-x3 -0.21739130 -4.612896 4.178113 0.9999194
## x5-x4 -1.03703704 -5.270371 3.196297 0.9606483
```

```
plot(Tukey_res)
```



Przeprowadź analizę za pomocą metody Bonferroniego:

```
pairwise_bonf = pairwise.t.test(dane_anova$dane, dane_anova$proba, p.adj = 'bonf')
pairwise_bonf
```

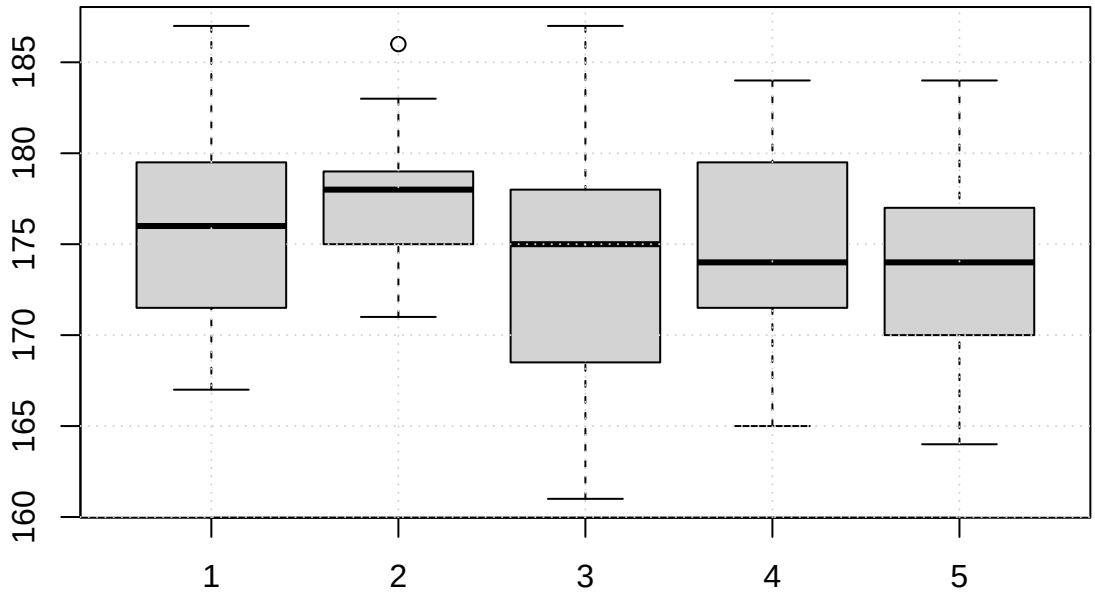
```
##
##  Pairwise comparisons using t tests with pooled SD
##
```

```
## data:  dane_anova$dane and dane_anova$proba
##
##    x1 x2 x3 x4
## x2 1  -  -  -
## x3 1  1  -  -
## x4 1  1  1  -
## x5 1  1  1  1
##
## P value adjustment method: bonferroni
```

Powtórz testy gdy zwiększymy wzrost wszystkich studentów w grupie drugiej o 2 i 5 cm.

2 cm

```
delta_mean = 2

x2 = read.csv('gr2.txt')$Wzrost + delta_mean

boxplot(x1, x2, x3, x4, x5)
grid()
```




```
dane_anova = data.frame( dane = c(x1, x2, x3, x4, x5),
          proba = rep( c('x1', 'x2', 'x3', 'x4', 'x5'),
          times = c(length(x1), length(x2),length(x3), length(x4), length(x5))) )

aov_res = aov(dane~proba, data = dane_anova)
summary(aov_res)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## proba         4    210   52.59   1.859  0.122
## Residuals   116   3282   28.29
```

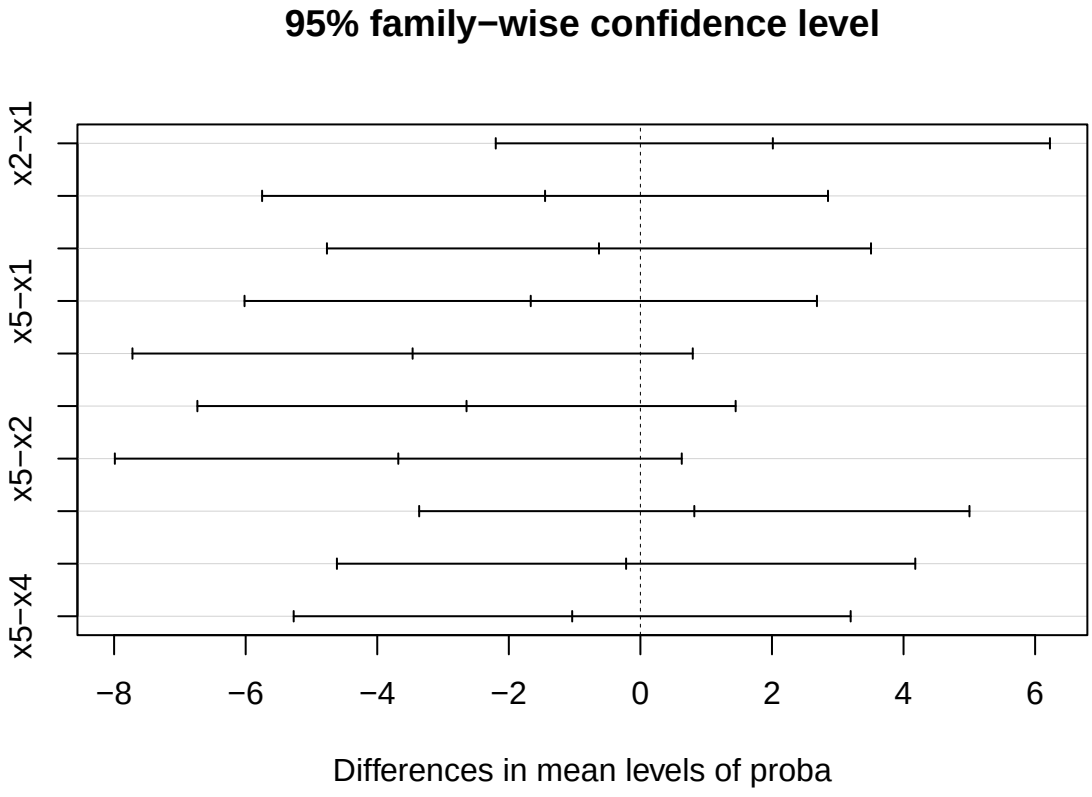
```
kruskal.test(dane~proba, dane_anova)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  dane by proba
## Kruskal-Wallis chi-squared = 7.2116, df = 4, p-value = 0.1251
```

```
Tukey_res = TukeyHSD(aov_res)
print(Tukey_res)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = dane ~ proba, data = dane_anova)
##
## $proba
##             diff       lwr       upr     p adj
## x2-x1  2.0133333 -2.198781 6.2254474 0.6765308
## x3-x1 -1.4492754 -5.750153 2.8516024 0.8831820
## x4-x1 -0.6296296 -4.764627 3.5053682 0.9932912
## x5-x1 -1.6666667 -6.017172 2.6838382 0.8256799
## x3-x2 -3.4626087 -7.721185 0.7959673 0.1679182
## x4-x2 -2.6429630 -6.733944 1.4480183 0.3841761
## x5-x2 -3.6800000 -7.988690 0.6286904 0.1319421
## x4-x3  0.8196457 -3.362671 5.0019623 0.9825810
## x5-x3 -0.2173913 -4.612896 4.1781129 0.9999194
## x5-x4 -1.0370370 -5.270371 3.1962968 0.9606483
```

```
plot(Tukey_res)
```

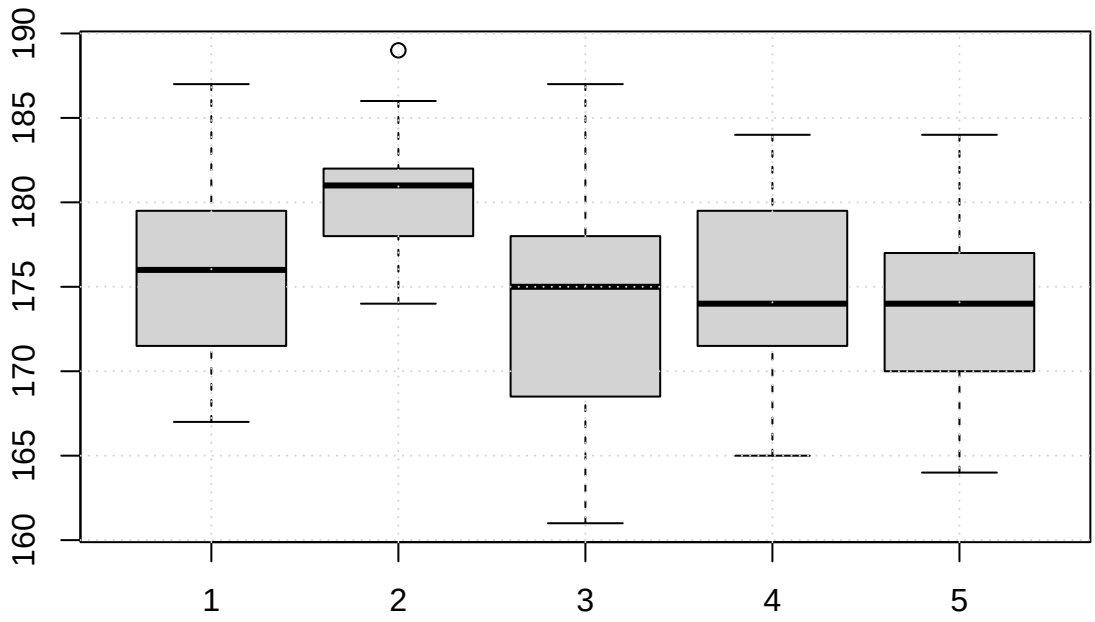



```
pairwaise_bonf = pairwise.t.test(dane_anova$dane, dane_anova$proba, p.adj = 'bonf')
pairwaise_bonf
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  dane_anova$dane and dane_anova$proba
##
##    x1   x2   x3   x4
## x2 1.00 -    -    -
## x3 1.00 0.26 -    -
## x4 1.00 0.76 1.00 -
## x5 1.00 0.20 1.00 1.00
##
## P value adjustment method: bonferroni
```

5 cm

```
delta_mean = 5

x2 = read.csv('gr2.txt')$Wzrost + delta_mean

boxplot(x1, x2, x3, x4, x5)
grid()
```




```
  dane_anova = data.frame( dane = c(x1, x2, x3, x4, x5),
            proba = rep( c('x1', 'x2', 'x3', 'x4', 'x5'),
            times = c(length(x1), length(x2),length(x3), length(x4), length(x5))) )

  aov_res = aov(dane~proba, data = dane_anova)
  summary(aov_res)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## proba         4    736  184.08   6.507 9.26e-05 ***
## Residuals   116   3282   28.29
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
  kruskal.test(dane~proba, dane_anova)
```

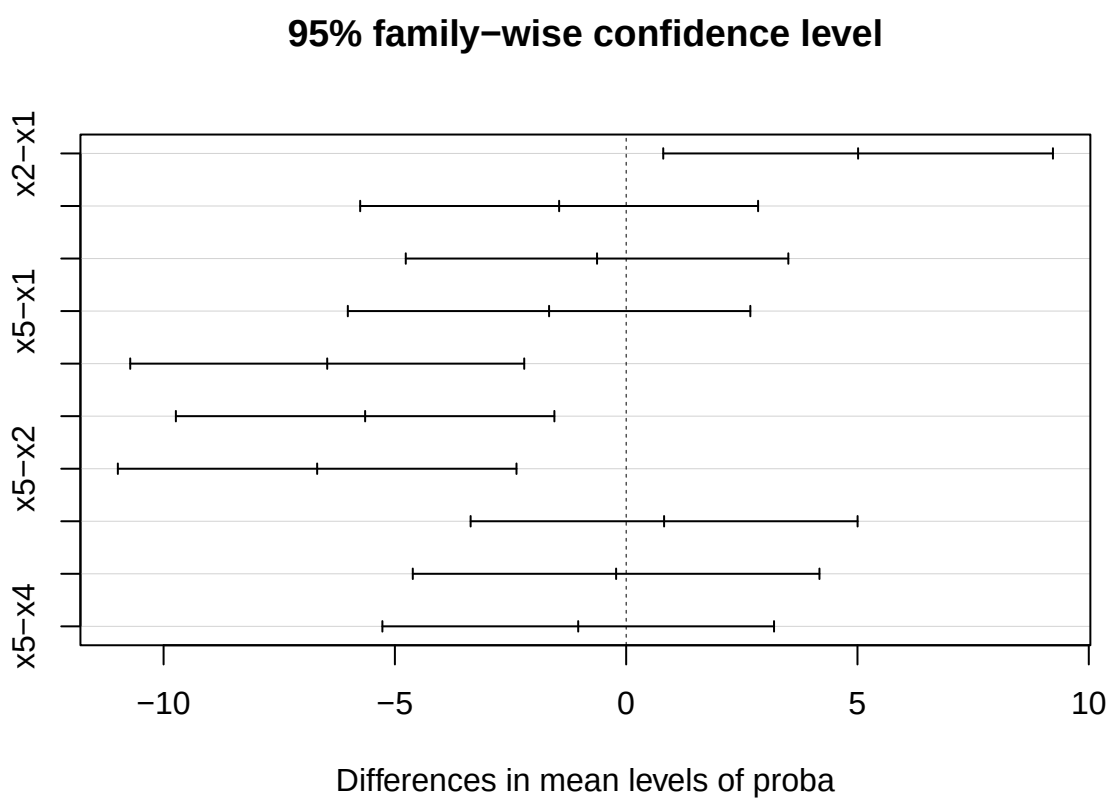
```
##
##  Kruskal-Wallis rank sum test
##
## data:  dane by proba
## Kruskal-Wallis chi-squared = 22.754, df = 4, p-value = 0.0001418
```

```
  Tukey_res = TukeyHSD(aov_res)
  print(Tukey_res)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = dane ~ proba, data = dane_anova)
```

```
##
## $proba
##              diff         lwr       upr     p adj
## x2-x1  5.0133333   0.8012193  9.225447 0.0111049
## x3-x1 -1.4492754  -5.7501531  2.851602 0.8831820
## x4-x1 -0.6296296  -4.7646275  3.505368 0.9932912
## x5-x1 -1.6666667  -6.0171715  2.683838 0.8256799
## x3-x2 -6.4626087 -10.7211847 -2.204033 0.0004878
## x4-x2 -5.6429630  -9.7339442 -1.551982 0.0019665
## x5-x2 -6.6800000 -10.9886904 -2.371310 0.0003453
## x4-x3  0.8196457  -3.3626709  5.001962 0.9825810
## x5-x3 -0.2173913  -4.6128955  4.178113 0.9999194
## x5-x4 -1.0370370  -5.2703709  3.196297 0.9606483
```

```
plot(Tukey_res)
```




```
pairwaise_bonf = pairwise.t.test(dane_anova$dane, dane_anova$proba, p.adj = 'bonf')
pairwaise_bonf
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  dane_anova$dane and dane_anova$proba
##
##    x1      x2      x3      x4
## x2 0.01292 -       -       -
## x3 1.00000 0.00052 -       -
```

```
## x4 1.00000 0.00214 1.00000 -
## x5 1.00000 0.00036 1.00000 1.00000
##
## P value adjustment method: bonferroni
```

Przykład 2 – regresja liniowa

Wczytaj kolumny „Wzrost” i „Waga” z poszczególnych plików:

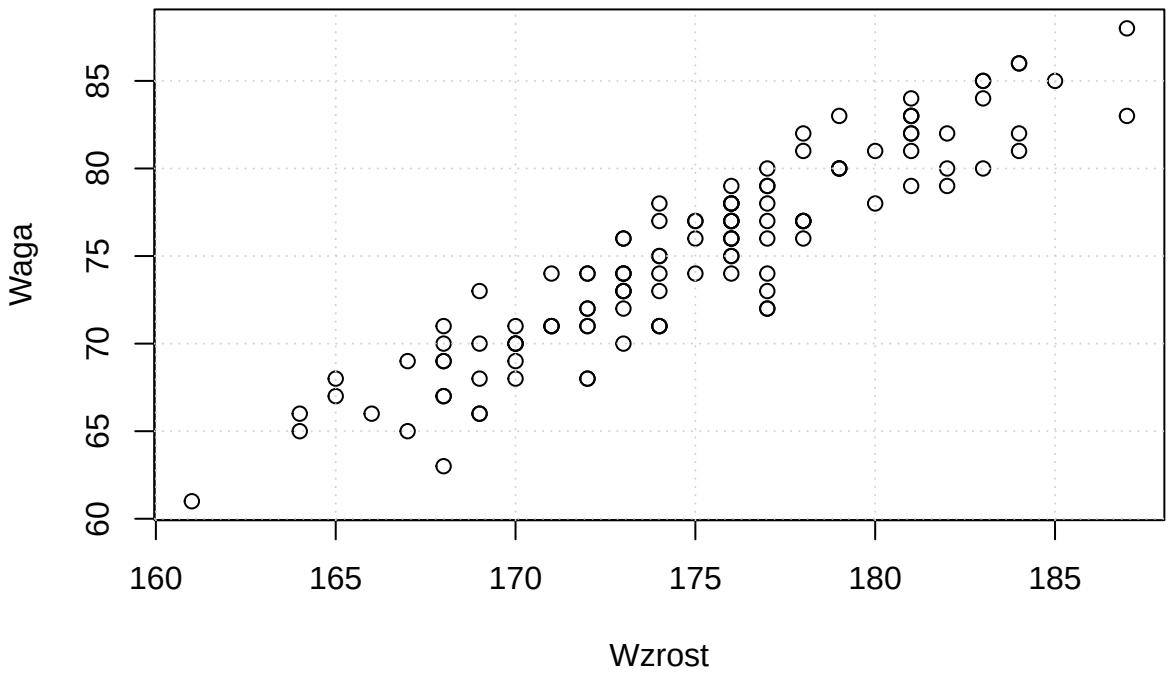
```
x1 = read.csv('gr1.txt')$Wzrost
x2 = read.csv('gr2.txt')$Wzrost
x3 = read.csv('gr3.txt')$Wzrost
x4 = read.csv('gr4.txt')$Wzrost
x5 = read.csv('gr5.txt')$Wzrost

y1 = read.csv('gr1.txt')$Waga
y2 = read.csv('gr2.txt')$Waga
y3 = read.csv('gr3.txt')$Waga
y4 = read.csv('gr4.txt')$Waga
y5 = read.csv('gr5.txt')$Waga
```

Połącz dane dot. wzrostu i wagi wszystkich studentów:

```
x = c(x1, x2, x3, x4, x5)
y = c(y1, y2, y3, y4, y5)
```

Narysuj wykres z danymi dot. wzrostu i wagi:

```
plot(x, y, xlab = 'Wzrost', ylab = 'Waga')
grid()
```



Wyznacz parametry prostej regresji $y = \beta_1 x + \beta_0$ i współczynnik determinacji R^2:

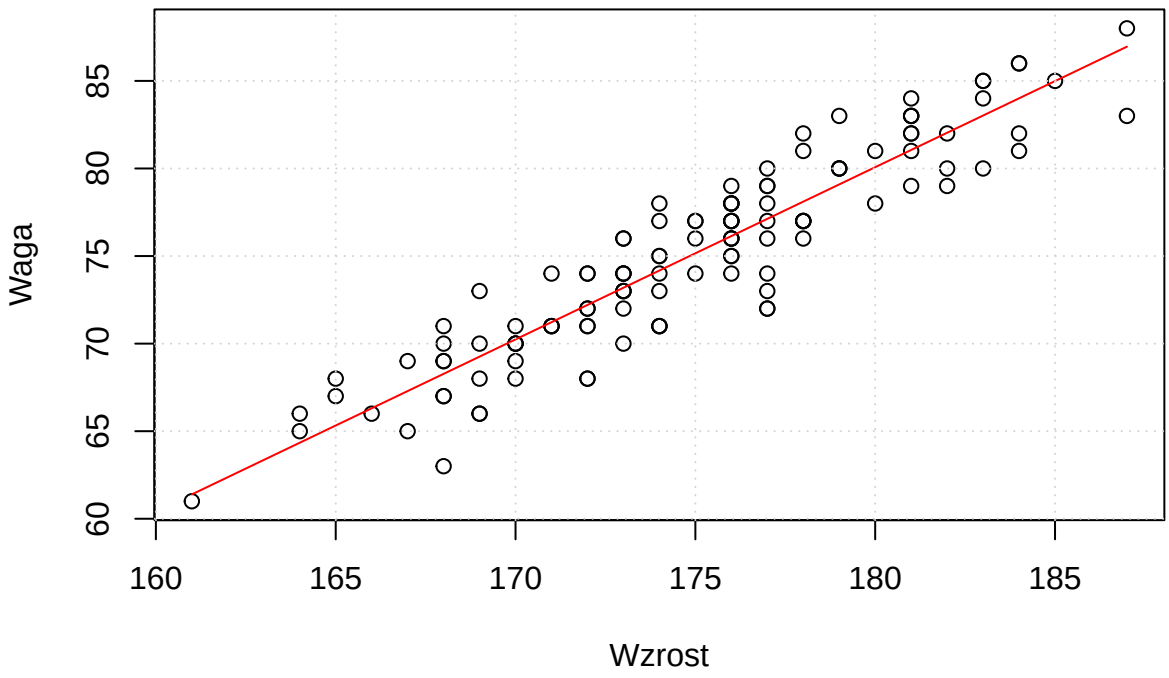
```
beta1_est = (mean(x*y) - mean(x)*mean(y)) / (mean(x^2) - (mean(x))^2)
beta0_est = mean(y) - beta1_est*mean(x)
y_est = beta1_est*x + beta0_est
R2 = 1 - sum((y - y_est)^2)/sum((y - mean(y))^2)
```

Prosta regresji $y = 0.9832x - 96.9064$.

Współczynnik $R^2 = 0.8578$.

Nanieś na rysunek prostą regresji:

```
plot(x, y, xlab = 'Wzrost', ylab = 'Waga')
arg = c(min(x), max(x))
out = beta1_est*arg + beta0_est
lines(arg, out, col = 'red')
grid()
```



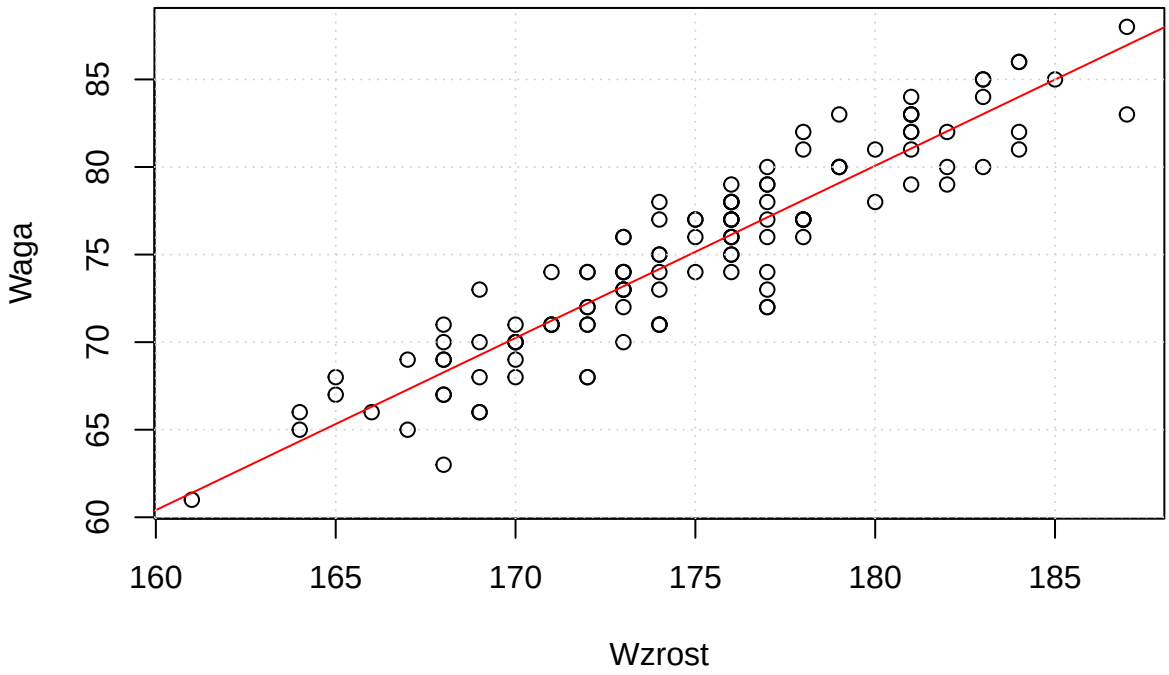
To samo wykonaj za pomocą funkcji lm:

```
lm_res = lm(y~x)
summary(lm_res)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2737 -1.2065 -0.0387  1.7935  3.9109
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -96.90638    6.42230  -15.09   <2e-16 ***
## x             0.98321    0.03669   26.80   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.12 on 119 degrees of freedom
## Multiple R-squared:  0.8578, Adjusted R-squared:  0.8566
## F-statistic:   718 on 1 and 119 DF,  p-value: < 2.2e-16
```

```
R2 = summary(lm_res)$r.squared
plot(x, y, xlab = 'Wzrost', ylab = 'Waga')
abline(lm_res, col='red')
```

```
grid()
```



Prosta regresji $y = 0.9832x - 96.9064$.

Współczynnik $R^2 = 0.8578$.

Zależność kwadratowa wzrostu od wagi: $y = \beta_1 x^2 + \beta_0$

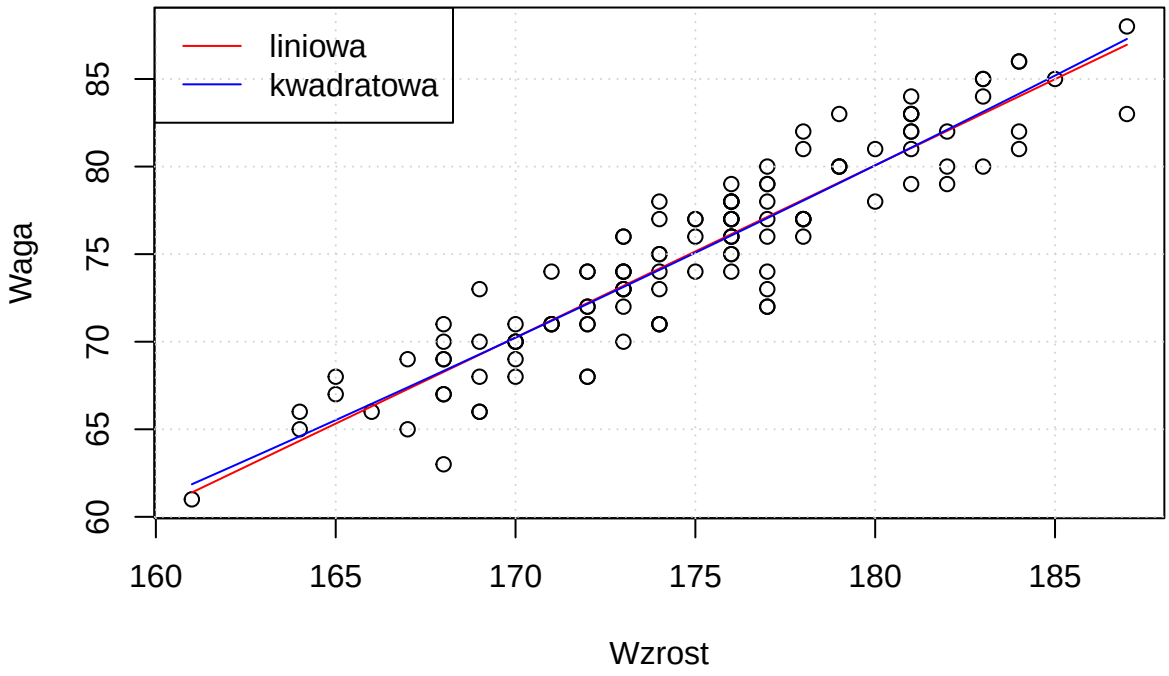
Metoda regresji liniowej za pomocą funkcji lm:

```
lm_res2 = lm(y~I(x^2))
summary(lm_res2)
```

```
##
## Call:
## lm(formula = y ~ I(x^2))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.332 -1.153 -0.065  1.842  3.946
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.978824   3.223153  -3.406 0.000899 ***
## I(x^2)        0.002810   0.000105  26.757  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 2.123 on 119 degrees of freedom
## Multiple R-squared:  0.8575, Adjusted R-squared:  0.8563
## F-statistic: 715.9 on 1 and 119 DF,  p-value: < 2.2e-16
```

```
R22 = summary(lm_res2)$r.squared

plot(x, y, xlab = 'Wzrost', ylab = 'Waga')
arg = seq(min(x), max(x), by = 1)
y_est = coef(lm_res)[2]*arg + coef(lm_res)[1]
y_est2 = coef(lm_res2)[2]*arg^2 + coef(lm_res2)[1]
lines(arg, y_est, col = 'red')
lines(arg, y_est2, col = 'blue')
grid()
legend('topleft', c('liniowa', 'kwadratowa'), col = c('red', 'blue'), lwd = 1)
```



Regresja kwadratowa: $y = 0.0028x^2 + -10.9788$.

Współczynnik $R^2 = 0.8575$.

Przykład 3 – predykcja

Oblicz prognozowaną wartość wagi studenta przy wzroście 184 cm:

```
x_new = 184
y_new = beta1_est*x_new + beta0_est
```

lub

```r
y_new = coef(lm_res)[2]*x_new + coef(lm_res)[1]
```

Prognozowana waga studenta przy wzroście 184 cm wynosi 84.01.

Oszacuj odchylenie standardowe błędu prognozy wagi studenta o wzroście 184 cm:

```r
n = length(x)
y_est = coef(lm_res)[2]*x + coef(lm_res)[1]
s2 = 1/(n - 2)*sum((y - y_est)^2)
s2_y_new = s2*( 1 + 1/n + (mean(x) - x_new)^2/(n*(mean(x^2) - mean(x)^2)) )
s_y_new = sqrt(s2_y_new)
```

Odchylenie standardowe błędu prognozy wagi studenta o wzroście 184 cm wynosi 2.15.

Prognozowana wartość wagi studenta i odchylenie standardowe za pomocą funkcji R predict():

```r
pred_res = predict(lm_res, data.frame(x = x_new), se=T)
pred_res$fit
```

```
##        1
## 84.00512
```

```r
s_y_new2 = sqrt(pred_res$se.fit^2 + pred_res$residual.scale^2)
```

Prognozowana waga studenta przy wzroście 184 cm wynosi 84.01, a odchylenie standardowe 2.15.

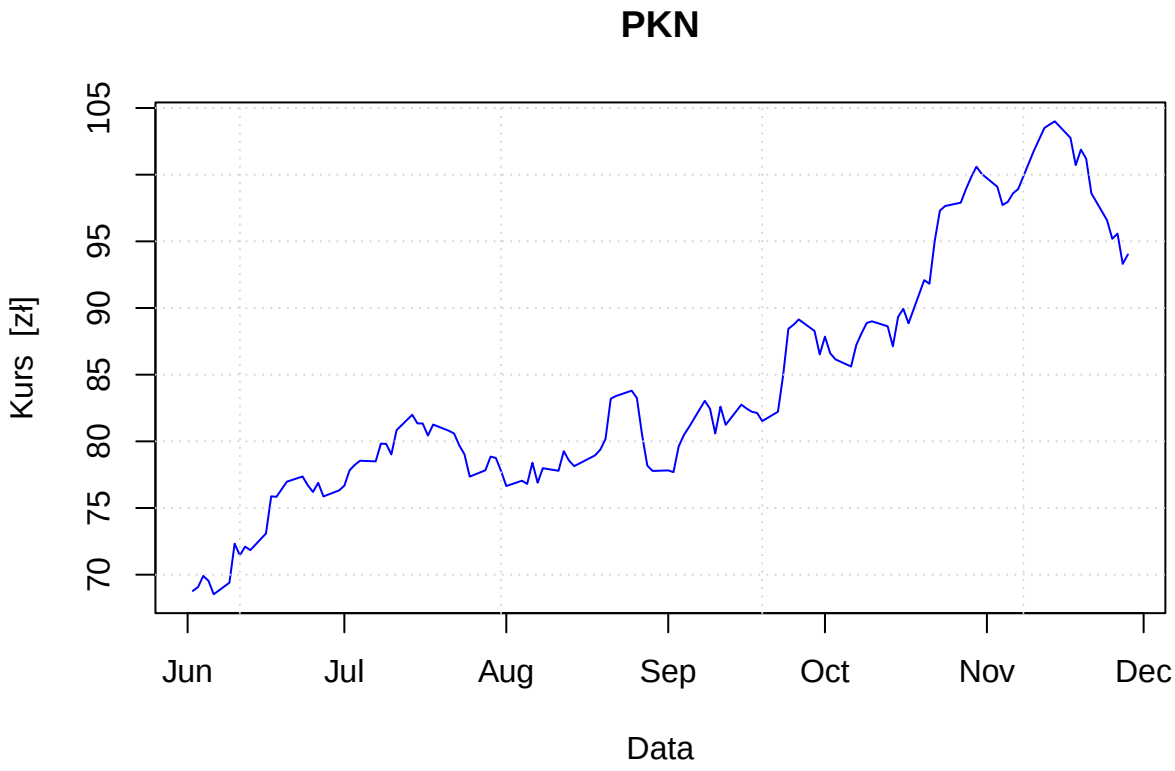
Przykład 4 – regresja, przypadek wielowymiarowy

Pomocnicza funkcja wczytująca kursy zamknięcia z plików z danymi ze storny https://stooq.pl/:

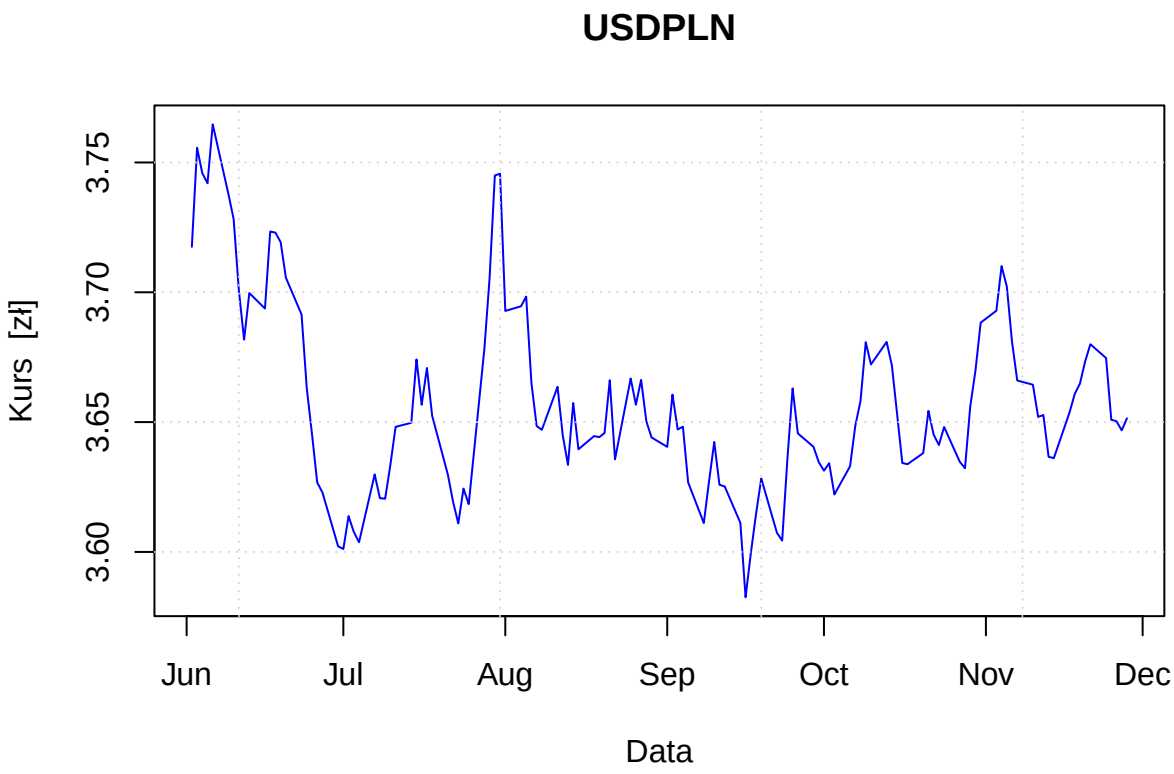
```r
read_kursy = function(Symbol, dateStart, dateEnd) {
  webLink = paste0('https://stooq.pl/q/d/l/?s=', Symbol, '&i=d')
  fileName = paste0(Symbol, '.csv')
  # if(!file.exists(fileName)) {
    download.file(webLink, fileName)
  # }
  df_Symbol = read.csv(fileName)
  df_Symbol$Data = as.Date(df_Symbol$Data)
  df_Symbol = df_Symbol[which(df_Symbol$Data >= dateStart & df_Symbol$Data <= dateEnd),]
  df_Symbol = subset(df_Symbol, select = c(Data, Zamkniecie))
  names(df_Symbol) = c('Data', Symbol)
df_Symbol }
```

Narysuj wykres kursów spółki ORLEN (PKN) i kursu USD w zależności od daty dla danych z ostatnich sześciu miesięcy.

```r
dateStart = '2025-06-01'
dateEnd = '2025-11-30'

Symbol = 'PKN'
df_Symbol = read_kursy(Symbol, dateStart, dateEnd)
  plot(PKN ~ Data, df_Symbol, type = 'l', col = 'blue',
       xlab = 'Data', ylab = 'Kurs  [zł]', main = Symbol )
  grid()
```




```
Symbol = 'USDPLN'
df_Symbol = read_kursy(Symbol, dateStart, dateEnd)
  plot(USDPLN ~ Data, df_Symbol, type = 'l', col = 'blue',
       xlab = 'Data', ylab = 'Kurs  [zł]', main = Symbol )
  grid()
```



Wyznacz parametry modelu regresji liniowej dla zależności kursu zamknięcia indeksu WIG20 od kursów zamknięcia spółek ASSECO (ASE) KGHM (KGH), ORLEN (PKN), PKOBP (PKO) dla danych z ostatnich sześciu miesięcy.

Pobieranie plików z danymi:

```
dateStart = '2025-06-01'
dateEnd = '2025-11-30'

df_WIG20 = read_kursy('WIG20', dateStart, dateEnd)
df_ASE = read_kursy('ASE', dateStart, dateEnd)
df_KGH = read_kursy('KGH', dateStart, dateEnd)
df_PKN = read_kursy('PKN', dateStart, dateEnd)
df_PKO = read_kursy('PKO', dateStart, dateEnd)
```

Metoda regresji liniowej za pomocą funkcji lm:

```
df_Dane = merge(df_WIG20, df_ASE, by = 'Data')
df_Dane = merge(df_Dane, df_KGH, by = 'Data')
df_Dane = merge(df_Dane, df_PKN, by = 'Data')
df_Dane = merge(df_Dane, df_PKO, by = 'Data')

lm_res = lm(WIG20 ~ ASE + KGH + PKN + PKO, data = df_Dane)
summary(lm_res)
```

```
##
## Call:
## lm(formula = WIG20 ~ ASE + KGH + PKN + PKO, data = df_Dane)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.382 -16.259  -1.661  13.404  56.711
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1231.7215    56.9321  21.635  < 2e-16 ***
## ASE            3.5881     0.5996   5.985 2.24e-08 ***
## KGH           -0.2335     0.1845  -1.265    0.208
## PKN            5.1354     0.6203   8.279 1.83e-13 ***
## PKO           13.5920     0.5781  23.513  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.37 on 122 degrees of freedom
## Multiple R-squared:  0.9271, Adjusted R-squared:  0.9247
## F-statistic:   388 on 4 and 122 DF,  p-value: < 2.2e-16
```

Wartości estymat parametrów modelu możemy też obliczyć na podstawie wzoru:

$$\hat{\beta} = (X^T X)^{-1} X^T Y.$$

Poniżej implementacja wzoru w R:

```
X = with(df_Dane, cbind(rep(1, length(WIG20)), ASE, KGH, PKN, PKO))
Y = df_Dane$WIG20
beta_est = solve(t(X)%*%X)%*%t(X)%*%Y
beta_est
```

```
##             [,1]
##     1231.7215298
## ASE    3.5881182
## KGH   -0.2335275
## PKN    5.1354200
## PKO   13.5919600
```