Regression Model Projekt - Kacper Ludwiczak, 221303

Temat: Kształtowanie się oczekiwanej długości życia dla większości państw świata na podstawie modelu regresji i czynniki wpływające na jego wysokość.

Zaczynając projekt pracowałem na danych związanych z nowoczesnymi technologiami. Dokonałem czyszczenia danych, obliczeń w excelu oraz obliczeń w języku R. Projekty znajdują się pod nazwą “Stary projekt - Reggresion Models”, “Stary projekt – Zestawienie", “Stary projekt w R”. Jednakże stwierdziłem, że wyniki nie są zadowalające i porzuciłem ten projekt.

Następnie zacząłem prace nad tym modelem. Dane pozyskałem ze strony kaggle.com. Według strony dane pochodzą ze strony internetowej WHO i ONZ z pomocą Deeksha Russella i Duana Wanga. Dane dotyczą większości państw świata, ich różnych zbiorów informacji, np. Długość życia. Plik z oryginałem danych jest pod nazwą “Life Expectancy Data”.

Po pierwsze chciałem skupić się na próbie 30 krajów. Wybrałem zatem 30 krajów, metodą losową. Dane obliczyłem zarówno w excelu jak i w języku R. Dane i obliczenia znajdują się w pliku “Projekt w R próba” oraz “Projekt w Excel próba”.

Uznałem, że lepszym pomysłem będzie stworzenie modelu z większości krajów na jakich dano było mi pracować. Zacząłem od pozostawienia tylko krajów z 2015 jako najbardziej aktualnych z danej bazy. Dane są pod nazwą “Dane z 2015”. Następnie zamieniłem dane za pomocą funkcji “Tekst jako kolumny”. Usunąłem kolumny “Year”, “Status”, ponieważ kolumny są mi niepotrzebne. Usunąłem kolumny “Alcohol”, “percentage expenditure”, “Total expenditure”, ponieważ były zauważające braki danych. Usunąłem kolumny “BMI”, “HIV/AIDS”, “thinness 1-19 years”, “thinness 5-9 years”, “Schooling”, ponieważ pojawiła mi się pewna anomalia. Podczas zamiany na kolumny niektóre dane są formacie daty, po zamienieniu ich na liczby pojawiały się błędne liczby. Również nie które liczby zamienione automatycznie są błędne. Po wielu nieudanych próbach rozwiązania tego problemu, dokonałem eliminacji tych kolumn. Również musiałem dokonać zamiany kropek na przecinki w liczbach oraz wypełnić puste pola średnią z reszty kolumny. Dane znajdują się w pliku “Dane zrobione”. Również stworzyłem osobny plik dla importowania do RStudio, pod nazwą “Dane do R”.

Szczegóły dotyczące danych:

- Country - Kraj
- Year - Rok
- Status - Stan rozwinięty lub rozwijający się
- Life expectancy - Oczekiwana długość życia w latach
- Adult Mortality - Wskaźniki śmiertelności dorosłych obu płci (prawdopodobieństwo śmierci w wieku od 15 do 60 lat na 1000 mieszkańców)
- Infant deaths - Liczba zgonów niemowląt na 1000 ludności
- Alcohol - Spożycie alkoholu na mieszkańca (15+) (w litrach czystego alkoholu)
- Hepatitis B - Zasięg szczepień przeciw wirusowemu zapaleniu wątroby typu B (HepB) wśród 1-latków (%)
- Measles – Odra, liczba zgłoszonych przypadków na 1000 ludności
- BMI - Średni wskaźnik masy ciała całej populacji
- Under-five deaths - Liczba zgonów poniżej piątego roku życia na 1000 mieszkańców
- Polio - Zasięg szczepień przeciw polio (Pol3) wśród 1-latków (%)
- Total expenditure - Wydatki sektora instytucji rządowych i samorządowych na zdrowie jako odsetek wydatków sektora instytucji rządowych i samorządowych ogółem (%)

- Diphtheria - Odsetek szczepień przeciwko anatoksynie błonicy i tężcowi oraz krztuścowi (DTP3) wśród 1-latków (%)
- HIV/AIDS - Zgony na 1000 żywych urodzeń HIV/AIDS (0-4 lata)
- GDP - Produkt Krajowy Brutto per capita (w USD)
- Population - Ludność kraju
- Thinness 1-19 years - Rozpowszechnienie szczupłości wśród dzieci i młodzieży w wieku od 10 do 19 lat (%)
- Thinness 5-9 years - Występowanie szczupłości wśród dzieci w wieku od 5 do 9 lat (%)
- Income - Wskaźnik rozwoju społecznego pod względem struktury dochodów zasobów (wskaźnik w zakresie od 0 do 1)
- Schooling- Liczba lat nauki (lata)

Projekt w RStudio jest pod nazwą “Projekt w R”

Zainstalowałem pakiet “readxl”, oraz uruchomiłem go przez funkcje “library”.

```
#Instalacja pakietu 'readxl', oraz zaznaczenie pakietu przez funkcje library
library(readxl)
```

Kod wczytuje plik excel "Dane do R.xlsx" i zapisuje jego zawartość w zmiennej "Dane". Następnie wyświetla zawartość tej zmiennej za pomocą funkcji "View()", wyświetla nazwy kolumn danych za pomocą funkcji "names()", a na końcu wyświetla wartości kolumny "Life expectancy" za pomocą funkcji "print()".

```
#Przydzielenie bazy danych do wektora 'Dane'
Dane <-  read_excel("C:/Users/Kacper/Desktop/Projekt Regression/Dane do R.xlsx")
View(Dane)
names(Dane)
print(Dane$"Life expectancy" )
```

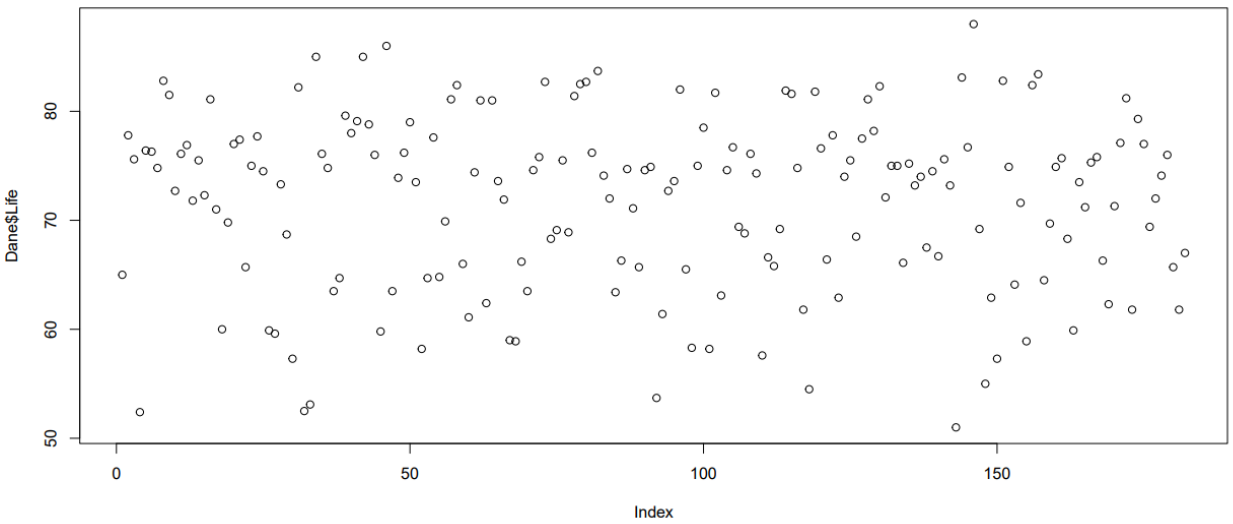
```
> print(Dane$"Life expectancy" )
  [1] 65.0 77.8 75.6 52.4 76.4 76.3 74.8 82.8 81.5 72.7 76.1 76.9 71.8 75.5 72.3 81.1 71.0 60.0 69.8
 [20] 77.0 77.4 65.7 75.0 77.7 74.5 59.9 59.6 73.3 68.7 57.3 82.2 52.5 53.1 85.0 76.1 74.8 63.5 64.7
 [39] 79.6 78.0 79.1 85.0 78.8 76.0 59.8 86.0 63.5 73.9 76.2 79.0 73.5 58.2 64.7 77.6 64.8 69.9 81.1
 [58] 82.4 66.0 61.1 74.4 81.0 62.4 81.0 73.6 71.9 59.0 58.9 66.2 63.5 74.6 75.8 82.7 68.3 69.1 75.5
 [77] 68.9 81.4 82.5 82.7 76.2 83.7 74.1 72.0 63.4 66.3 74.7 71.1 65.7 74.6 74.9 53.7 61.4 72.7 73.6
 [96] 82.0 65.5 58.3 75.0 78.5 58.2 81.7 63.1 74.6 76.7 69.4 68.8 76.1 74.3 57.6 66.6 65.8 69.2 81.9
[115] 81.6 74.8 61.8 54.5 81.8 76.6 66.4 77.8 62.9 74.0 75.5 68.5 77.5 81.1 78.2 82.3 72.1 75.0 75.0
[134] 66.1 75.2 73.2 74.0 67.5 74.5 66.7 75.6 73.2 51.0 83.1 76.7 88.0 69.2 55.0 62.9 57.3 82.8 74.9
[153] 64.1 71.6 58.9 82.4 83.4 64.5 69.7 74.9 75.7 68.3 59.9 73.5 71.2 75.3 75.8 66.3 62.3 71.3 77.1
[172] 81.2 61.8 79.3 77.0 69.4 72.0 74.1 76.0 65.7 61.8 67.0
```

Tutaj za pomocą funkcji “colnames” dokonałem zmiany nazw kolumn w celu szybszej pracy w dalszych etapach.

```
#Zmianna nazw zmiennych
colnames(Dane)<-c("Life","Adult_M","Infant_D",
```

Użyłem funkcji “plot” w celu zobaczeniu na wykresie dane z kolumny “Life”. Kolumna “Life” zostaje moją zmienną zależną, natomiast reszta kolumn zmiennymi niezależnymi.

```
#Przedstawienie zmiennej zaleznej na wykresie
plot(Dane$"Life")
```



Sprawdziłem następujące wartości danych:

- Minimalna wartość = 51.00
- Pierwszy kwartal = 65.85
- Mediana = 73.95
- Średnia = 71.72
- Trzeci kwartal = 76.97
- Maksymalna wartość = 88.00

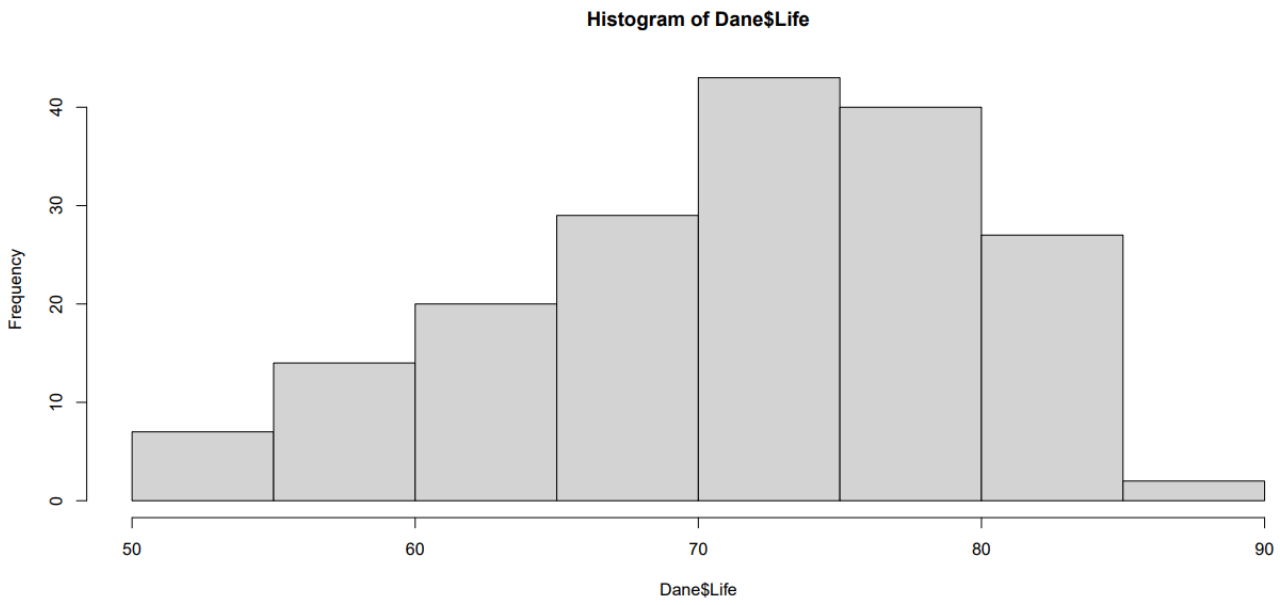
, dla zmiennej zależnej.

```
#Przedstawienie nastepujacych danych ze zmiennej zaleznej:
summary(Dane$"Life" )
```

```
> summary(Dane$"Life" )
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  51.00   65.85   73.95   71.72   76.97   88.00
```

W celu zobrazowania graficznego użyłem funkcji “hist”. Według tego można zauważyć dominacje wartości w przedziale od 70 do 80.

```
#Przedstawienie histogramy zmiennej zaleznej
hist(Dane$"Life" )
```



Za pomocą funkcji “cor” przedstawiłem korelacje wszystkich zmiennych. Im dana liczba jest większa tym większa jest korelacja między odpowiadającymi danymi w wierszu i kolumnie.

Można z niej ustalić, że korelacja między zmienną zależną “Life” a zmienną:

- “Adult_M” jest duża ujemnie
- “Infant_D” jest bardzo mała ujemnie
- “H_B” jest mała dodatnio
- “Measles” jest bardzo mała ujemnie
- “Under_D” jest bardzo mała ujemnie
- “Polio” jest średnia dodatnio
- “Dipht” jest średnia dodatnio
- “GDP” jest mała dodatnio
- “Popl” jest bardzo mała ujemnie
- “Income” jest bardzo mała ujemnie

```
#Przedstawienie korelacji wszystkich zmiennych
cor(Dane)
```

```
> cor(Dane)
              Life     Adult_M    Infant_D         H_B      Measles     Under_D       Polio        Dipht
Life      1.00000000 -0.77215206 -0.23984305  0.40450050 -0.077810464 -0.27495588  0.52186030  0.506553583
Adult_M  -0.77215206  1.00000000  0.18604690 -0.23139939  0.054570792  0.21446481 -0.37773277 -0.327386992
Infant_D -0.23984305  0.18604690  1.00000000 -0.08681527  0.801663443  0.99348313 -0.12958871 -0.117880715
H_B       0.40450050 -0.23139939 -0.08681527  1.00000000  0.015656414 -0.11001898  0.59020367  0.909255374
Measles  -0.07781046  0.05457079  0.80166344  0.01565641  1.000000000  0.76499012 -0.02835947 -0.001622602
Under_D  -0.27495588  0.21446481  0.99348313 -0.11001898  0.764990118  1.00000000 -0.15154462 -0.143729736
Polio     0.52186030 -0.37773277 -0.12958871  0.59020367 -0.028359472 -0.15154462  1.00000000  0.661623315
Dipht     0.50655358 -0.32738699 -0.11788072  0.90925537 -0.001622602 -0.14372974  0.66162332  1.000000000
GDP       0.43089742 -0.31610931 -0.11953200  0.13188477 -0.076143290 -0.12620291  0.22349572  0.213332775
Popl     -0.04721323  0.03756295  0.26624483 -0.05672244  0.127925269  0.30372017 -0.22107193 -0.065038775
Income   -0.13196089  0.13031897  0.02150182 -0.08670694  0.008493532  0.05419071 -0.06262976 -0.100745256
                GDP          Popl        Income
Life      0.43089742 -0.047213234 -0.131960889
Adult_M  -0.31610931  0.037562946  0.130318973
Infant_D -0.11953200  0.266244826  0.021501816
H_B       0.13188477 -0.056722444 -0.086706940
Measles  -0.07614329  0.127925269  0.008493532
Under_D  -0.12620291  0.303720175  0.054190707
Polio     0.22349572 -0.221071926 -0.062629761
Dipht     0.21333277 -0.065038775 -0.100745256
GDP       1.00000000  0.046037030 -0.047562832
Popl      0.04603703  1.000000000 -0.007033844
Income   -0.04756283 -0.007033844  1.000000000
```

Kod ten oblicza korelację między kolumnami w obiekcie "Dane", z wyjątkiem pierwszej kolumny, która jest pomijana za pomocą wyrażenia "[, -1]". Jest to kolumna “Life”. Funkcja "cor" oblicza korelację Pearsona, czyli stopień zależności liniowej między dwoma zmiennymi. Wynik jest następnie zaokrąglany do trzech miejsc po przecinku za pomocą funkcji "round".

```
#współliniowość
round(cor(Dane[,-1]),3)
```

Kod ten tworzy macierz z danych zawartych w obiekcie "Dane". Wiersze z obiektu "Dane" są wczytywane za pomocą wyrażenia "[, -1]", co oznacza, że pierwsza kolumna jest pomijana. Jest to kolumna “Life”. Następnie dane są konwertowane na macierz za pomocą funkcji "as.matrix". Ostatecznie macierz jest przypisywana do obiektu "matrix".

```
matrix <- as.matrix(Dane[,-1])
```

Kod oblicza własne wartości i wektory dla macierzy "matrix". Macierz jest najpierw transponowana za pomocą funkcji "t", a następnie mnożona przez siebie za pomocą operatora "%*%", co daje macierz kowariancji. Otrzymaną macierz kowariancji jest analizowana przez funkcję "eigen", która oblicza własne wartości i wektory. Wynik jest przypisywany do obiektu "matrix_eigen". Dzięki temu mogę zobaczyć własne wartości macierzy.

```
matrix_eigen <- eigen(t(matrix) %*% matrix)
matrix_eigen$val
```

```
> matrix_eigen$val
 [1] 1.365373e+17 2.887223e+10 1.122436e+10 6.501808e+06 1.227256e+06 7.446202e+05
 [7] 4.555312e+04 1.774313e+04 7.863299e+03 5.388979e+03
```

Kod wylicza pierwiastek kwadratowy elementów ilorazu dwóch wektorów za pomocą “sqrt” “`matrix_eigen$val[1]`” jest pierwszym elementem wektora “`matrix_eigen$val`”, a “`matrix_eigen$val`” jest całym wektorem. Dostając następujące wyniki. Wyniki powyżej 30 wskazują na brak współliniowości.

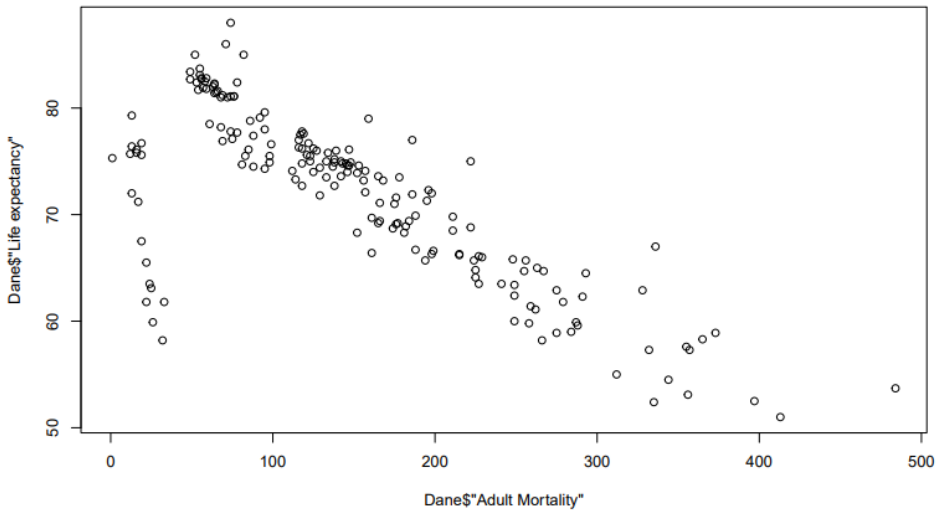
W moim przypadku można zauważyć, że współliniowość nie występuje. Na tym etapie można wykluczyć jako by mój model był dobrym modelem.

```
sqrt(matrix_eigen$val[1]/matrix_eigen$val)
```

```
> sqrt(matrix_eigen$val[1]/matrix_eigen$val)
  [1]       1.000    2174.631    3487.746  144913.411  333547.800  428211.415
  [7] 1731277.043 2774026.198 4166997.596 5033527.140
```

Sprawdziłem na wykresie jaka jest korelacje zmiennej zależnej i zmiennej “Adult_M” w postaci graficznej.

```
#Przedstawienie wykresu ze zmienna zaleznom wertykalnie i zmienna
plot(Dane$"Life"~Dane$"Adult_M")
```



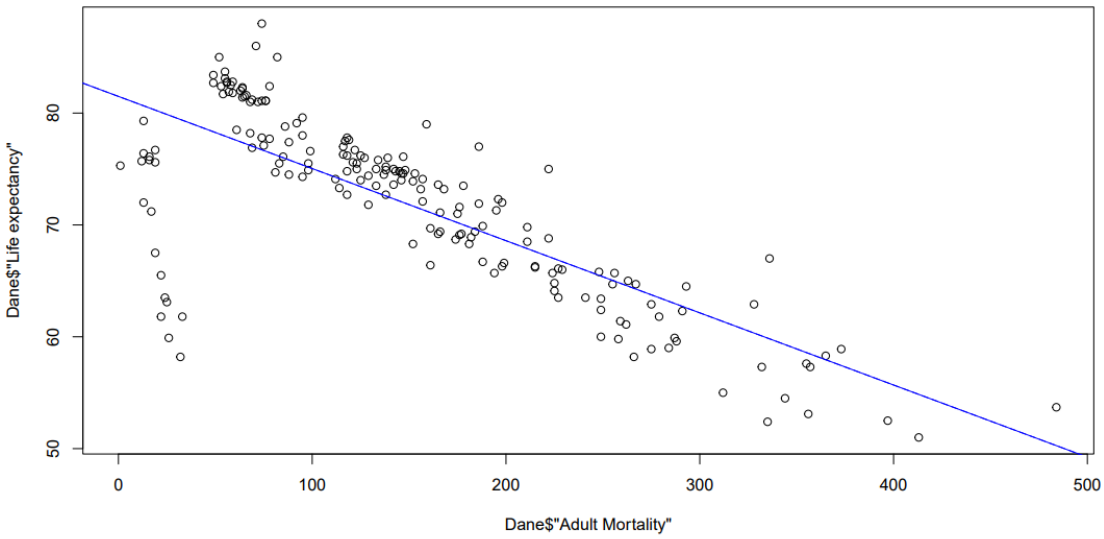
Tutaj przetestowałem funkcje “lm” za pomocą stworzenia modelu prostej regresji liniowej z użyciem jednej zmiennej niezależnej. Wynik p-value wyszedł bardzo dobry. Jednakże wartość R-squared zbyt mała.

```
#Tworzenie modelu simple regression za pomoca funkcji lm(linear model).
SimpleModel<- lm(Dane$"Life"~Dane$"Adult_M")
summary(SimpleModel)
```

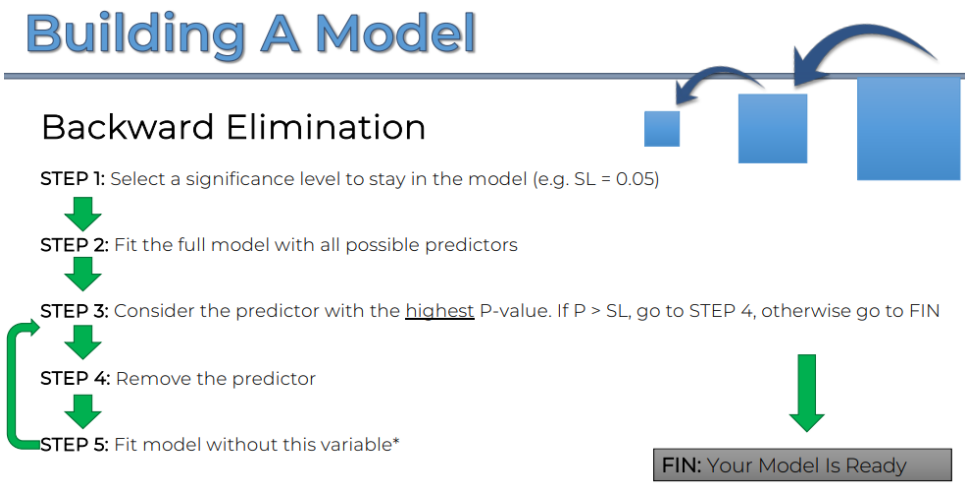
```
> summary(SimpleModel)

Call:
lm(formula = Dane$"Life expectancy" ~ Dane$"Adult Mortality")

Residuals:
     Min       1Q   Median       3Q      Max
-21.2282  -1.6823   0.7664   2.9876  11.2813

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           81.492535   0.709494   114.9   <2e-16 ***
Dane$"Adult Mortality" -0.064512   0.003957   -16.3   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.117 on 180 degrees of freedom
Multiple R-squared:  0.5962,    Adjusted R-squared:  0.594
F-statistic: 265.8 on 1 and 180 DF,  p-value: < 2.2e-16
```

Funkcja “abline” umożliwiła mi dodanie niebieskiej linii regresji, która najdokładniej próbuje dopasować się do istniejących danych. Można zauważyć, że linia jest przekrzywiona w dół po stronie lewej, wynika to z pojawiających się wartości odstających, poniżej wartości 100 dla zmiennej niezależnej.

```
#Przedstawienie na wykresie niebieskiej lini
abline(SimpleModel$coef,col="blue")
```



Stworzenie pierwszego modelu. Używam metody “Backward” ręcznie. Eliminując zmienne o najwyższym wskaźniku p-value, tak długo aż model będzie zadawalający. Zakładam dla modelu, że poziom p-value poniżej 0.05 jest odpowiedni.




```
#Tworzenie modelu multiple regression, 'Life' jako zmienna zalezna.
Model <- lm(Life~Adult_M+Infant_D+H_B+Measles+Under_D+Polio+Dipht+GI
summary(Model)
```

Podsumowaniem modelu regresji liniowej wygląda następująco. Zawiera on informacje na temat tego, jak dobrze modelem można wyjaśnić zmienną zależną (Life), korzystając z 7 zmiennych objaśniających (Adult_M, Infant_D, H_B, Measles, Under_D, Polio, Dipht, GDP, Popl, Income).

Sekcja "Residuals" pokazuje rozkład reszt modelu regresji liniowej, czyli różnic między wartościami faktycznymi a wartościami przewidywanymi. "Min", "1Q", "Median", "3Q", i "Max" to odpowiednio najmniejsza, pierwszy kwartyl, medianę, trzeci kwartyl, i największą wartość reszt.

Sekcja "Coefficients" pokazuje wartości współczynników regresji dla każdej zmiennej objaśniającej, wraz z błędem standardowym i wartością statystyki t. Wartość p dla każdej zmiennej określa, czy jest ona istotna statystycznie ($p < 0,05$ oznacza, że zmienna jest istotna).

Wartość "Residual standard error" to średni kwadrat błędu reszty. "Multiple R-squared" to współczynnik determinacji, który określa, jak dobrze modelem można wyjaśnić zmienną zależną. Wartość R-kwadratu zawsze znajduje się w zakresie od 0 do 1, a wartość bliska 1 oznacza dobre dopasowanie modelu. "Adjusted R-squared" to współczynnik determinacji uwzględniający liczbę zmiennych objaśniających. Wartość "F-statistic" i "p-value" służą do testowania hipotezy zerowej, że wszystkie współczynniki regresji są równe 0.

```
> summary(Model)

Call:
lm(formula = Life ~ Adult_M + Infant_D + H_B + Measles + Under_D +
    Polio + Dipht + GDP + Popl + Income, data = Dane)

Residuals:
     Min       1Q   Median       3Q      Max
-16.3833  -2.4605   0.2957   2.3275  11.8063

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.931e+01  1.761e+00  39.356  < 2e-16 ***
Adult_M     -4.718e-02  3.959e-03 -11.919  < 2e-16 ***
Infant_D     9.576e-02  4.644e-02   2.062 0.040728 *
H_B         -9.270e-03  3.379e-02  -0.274 0.784131
Measles      1.253e-05  7.851e-05   0.160 0.873375
Under_D     -8.295e-02  3.440e-02  -2.412 0.016940 *
Polio        5.024e-02  1.887e-02   2.662 0.008500 **
Dipht        6.339e-02  3.886e-02   1.631 0.104718
GDP          1.202e-04  3.239e-05   3.712 0.000278 ***
Popl         2.531e-08  1.461e-08   1.732 0.085120 .
Income       1.122e-02  3.531e-02   0.318 0.751148
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.346 on 171 degrees of freedom
Multiple R-squared:  0.7233,    Adjusted R-squared:  0.7071
F-statistic: 44.69 on 10 and 171 DF,  p-value: < 2.2e-16
```

W pierwszym modelu największy poziom p-value miała kolumna “Measles”. W drugim modelu już się nie pojawiła.

```
#W modelu drugim usuwam 'Measles', dajac spadek wartosci p dla zmiennej 'Under_D'.
Model2 <- lm(Life~Adult_M+Infant_D+H_B+Under_D+Polio+Dipht+GDP+Popl+Income,data=Dane)
summary(Model2)
```

```
> summary(Model2)

Call:
lm(formula = Life ~ Adult_M + Infant_D + H_B + Under_D + Polio +
    Dipht + GDP + Popl + Income, data = Dane)

Residuals:
     Min       1Q   Median       3Q      Max
-16.3917  -2.4905   0.3158   2.3285  11.8053

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.930e+01  1.755e+00  39.481  < 2e-16 ***
Adult_M     -4.721e-02  3.944e-03 -11.968  < 2e-16 ***
Infant_D     9.971e-02  3.918e-02   2.545 0.011806 *
H_B         -9.209e-03  3.369e-02  -0.273 0.784901
Under_D     -8.530e-02  3.099e-02  -2.752 0.006554 **
Polio        5.024e-02  1.882e-02   2.670 0.008305 **
Dipht        6.345e-02  3.875e-02   1.637 0.103391
GDP          1.201e-04  3.229e-05   3.720 0.000269 ***
Popl         2.538e-08  1.457e-08   1.743 0.083161 .
Income       1.204e-02  3.483e-02   0.346 0.730039
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.334 on 172 degrees of freedom
Multiple R-squared:  0.7232,    Adjusted R-squared:  0.7087
F-statistic: 49.94 on 9 and 172 DF,  p-value: < 2.2e-16
```

W drugim modelu największy poziom p-value miała kolumna “H_B”. W trzecim modelu już się nie pojawiła.

```
#w modelu trzecim usuwam 'H_B', dajac spadek wartosci p dla zmiennej 'Dipht'.
Model3 <- lm(Life~Adult_M+Infant_D+Under_D+Polio+Dipht+GDP+Popl+Income,data=Dane)
summary(Model3)
```

```
> summary(Model3)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP + Popl + Income, data = Dane)

Residuals:
     Min       1Q   Median       3Q      Max
-16.4264  -2.4996   0.3435   2.3178  11.8929

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.933e+01  1.747e+00  39.684  < 2e-16 ***
Adult_M     -4.735e-02  3.899e-03 -12.143  < 2e-16 ***
Infant_D     9.943e-02  3.906e-02   2.546 0.011780 *
Under_D     -8.510e-02  3.090e-02  -2.754 0.006521 **
Polio        5.018e-02  1.876e-02   2.674 0.008208 **
Dipht        5.437e-02  1.988e-02   2.735 0.006881 **
GDP          1.211e-04  3.200e-05   3.785 0.000211 ***
Popl         2.531e-08  1.452e-08   1.743 0.083134 .
Income       1.203e-02  3.474e-02   0.346 0.729621
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.322 on 173 degrees of freedom
Multiple R-squared:  0.7231,    Adjusted R-squared:  0.7103
F-statistic: 56.47 on 8 and 173 DF,  p-value: < 2.2e-16
```

W trzecim modelu największy poziom p-value miała kolumna “Income”. W czwartym modelu już się nie pojawiła.

```
#W modelu czwartym usuwam 'Income'.
Model4 <- lm(Life~Adult_M+Infant_D+Under_D+Polio+Dipht+GDP+Popl,data=Dane)
summary(Model4)
```

```
> summary(Model4)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP + Popl, data = Dane)

Residuals:
     Min       1Q   Median       3Q      Max
-16.4750  -2.5123   0.3676   2.3051  11.8873

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.935e+01  1.742e+00  39.819  < 2e-16 ***
Adult_M     -4.730e-02  3.887e-03 -12.169  < 2e-16 ***
Infant_D     9.561e-02  3.737e-02   2.558 0.011365 *
Under_D     -8.206e-02  2.955e-02  -2.777 0.006095 **
Polio        5.021e-02  1.872e-02   2.683 0.008003 **
Dipht        5.425e-02  1.982e-02   2.737 0.006843 **
GDP          1.211e-04  3.192e-05   3.795 0.000203 ***
Popl         2.472e-08  1.439e-08   1.718 0.087513 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.312 on 174 degrees of freedom
Multiple R-squared:  0.7229,    Adjusted R-squared:  0.7118
F-statistic: 64.85 on 7 and 174 DF,  p-value: < 2.2e-16
```

W czwartym modelu największy poziom p-value miała kolumna “Popl”. W piątym modelu już się nie pojawiła.

```
#W modelu piatym usuwam 'Popl', daje wzrost wartosci 'Under_D' i 'Polio'
Model5 <- lm(Life~Adult_M+Infant_D+Under_D+Polio+Dipht+GDP,data=Dane)
summary(Model5)
```

```
> summary(Model5)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP, data = Dane)

Residuals:
    Min      1Q  Median      3Q     Max
-16.951  -2.437   0.477   2.551  11.735

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.980e+01  1.731e+00  40.319  < 2e-16 ***
Adult_M     -4.821e-02  3.872e-03 -12.451  < 2e-16 ***
Infant_D     7.326e-02  3.523e-02   2.080  0.03901 *
Under_D     -6.291e-02  2.753e-02  -2.286  0.02347 *
Polio        4.199e-02  1.819e-02   2.308  0.02218 *
Dipht        5.995e-02  1.965e-02   3.051  0.00264 **
GDP          1.264e-04  3.194e-05   3.957  0.00011 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.336 on 175 degrees of freedom
Multiple R-squared:  0.7182,    Adjusted R-squared:  0.7086
F-statistic: 74.34 on 6 and 175 DF,  p-value: < 2.2e-16
```

W piątym modelu po usunięciu kolumny “Popl”, nastąpił duży wzrost wartości p-value dla kolumny “Under_D” oraz “Polio”. Jest efekt niepożądany dla mojego modelu, dlatego wybranym modelem

zostaje model czwarty. Pomimo wartości p-value dla kolumny “Popl” w okolicy 0.08. Również poziom R-squared wynosi 0.7229, możemy interpretować to jako dobry model, choć na granicy złego (zmienność oczekiwanej długości życia jest wyjaśniony w 72% przez model). Dodatkowo Adjusted R-squared jest na poziomie 0.7118, jest to najlepszy wynik ze wszystkich pięciu policzonych.

```
#Wybrany model
Model <- lm(Life~Adult_M+Infant_D+Under_D+Polio+Dipht+GDP+Popl,data=Dane)
summary(Model)
```

```
> summary(Model)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP + Popl, data = Dane)

Residuals:
     Min       1Q   Median       3Q      Max
-16.4750  -2.5123   0.3676   2.3051  11.8873

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.935e+01  1.742e+00  39.819  < 2e-16 ***
Adult_M     -4.730e-02  3.887e-03 -12.169  < 2e-16 ***
Infant_D     9.561e-02  3.737e-02   2.558 0.011365 *
Under_D     -8.206e-02  2.955e-02  -2.777 0.006095 **
Polio        5.021e-02  1.872e-02   2.683 0.008003 **
Dipht        5.425e-02  1.982e-02   2.737 0.006843 **
GDP          1.211e-04  3.192e-05   3.795 0.000203 ***
Popl         2.472e-08  1.439e-08   1.718 0.087513 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.312 on 174 degrees of freedom
Multiple R-squared:  0.7229,    Adjusted R-squared:  0.7118
F-statistic: 64.85 on 7 and 174 DF,  p-value: < 2.2e-16
```

Model	list [12] (S3: lm)	List of length 12
coefficients	double [8]	69.3515 -0.0473 0.0956 -0.0821 0.0502 0.0543 ...
residuals	double [182]	4.239 1.127 -4.302 -3.714 -3.897 -0.065 ...
effects	double [182]	-967.5 -83.4 10.6 14.1 24.7 -14.4 ...
rank	integer [1]	8
fitted.values	double [182]	60.8 76.7 79.9 56.1 80.3 76.4 ...
assign	integer [8]	0 1 2 3 4 5 ...
qr	list [5] (S3: qr)	List of length 5
df.residual	integer [1]	174
xlevels	list [0]	List of length 0
call	language	lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht + GDP + Po ...
terms	formula	Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht + GDP + Popl
model	list [182 x 8] (S3: data.frame)	A data.frame with 182 rows and 8 columns

W celu potwierdzenia wybrania dobrego modelu, użyłem funkcji “step” za równo direction “backward” jak i “both”. Obie funkcje dały mi wynik identyczny dla mojego. Kod tworzy nowy model regresji liniowej "Model_nowy_backward" przy użyciu procedury backward selection. Procedura ta polega na usuwaniu najmniej istotnych zmiennych objaśniających, aż pozostaną tylko te, które są istotne statystycznie. Na początku tworzony jest pełny model regresji liniowej. Następnie jest on przekształcany za pomocą funkcji "step" z argumentem "direction = 'backward'". W końcowym kroku wyświetlane jest podsumowanie modelu "Model_nowy_backward", a następnie "Model_nowy" jest ustawiany na "Model_nowy_backward" i wyświetlany jest ponownie podsumowanie modelu.

```r
#W celu potwierdzenia wyboru uzylem funkcji backward i both, ktore daly identyczny wynik.
Model_backward<-step(lm(Life~Adult_M+Infant_D+H_B+Measles+Under_D+Polio+Dipht+GDP+Popl+Income,data=Dane),direction="backward")
summary(Model_backward)
Model_both<-step(lm(Life~Adult_M+Infant_D+H_B+Measles+Under_D+Polio+Dipht+GDP+Popl+Income,data=Dane),direction="both")
summary(Model_both)
```

```
> summary(Model_backward)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP + Popl, data = Dane)

Residuals:
     Min       1Q   Median       3Q      Max
-16.4750  -2.5123   0.3676   2.3051  11.8873

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.935e+01  1.742e+00  39.819  < 2e-16 ***
Adult_M     -4.730e-02  3.887e-03 -12.169  < 2e-16 ***
Infant_D     9.561e-02  3.737e-02   2.558 0.011365 *
Under_D     -8.206e-02  2.955e-02  -2.777 0.006095 **
Polio        5.021e-02  1.872e-02   2.683 0.008003 **
Dipht        5.425e-02  1.982e-02   2.737 0.006843 **
GDP          1.211e-04  3.192e-05   3.795 0.000203 ***
Popl         2.472e-08  1.439e-08   1.718 0.087513 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.312 on 174 degrees of freedom
Multiple R-squared:  0.7229,    Adjusted R-squared:  0.7118
F-statistic: 64.85 on 7 and 174 DF,  p-value: < 2.2e-16

> summary(Model_both)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP + Popl, data = Dane)

Residuals:
     Min       1Q   Median       3Q      Max
-16.4750  -2.5123   0.3676   2.3051  11.8873

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.935e+01  1.742e+00  39.819  < 2e-16 ***
Adult_M     -4.730e-02  3.887e-03 -12.169  < 2e-16 ***
Infant_D     9.561e-02  3.737e-02   2.558 0.011365 *
Under_D     -8.206e-02  2.955e-02  -2.777 0.006095 **
Polio        5.021e-02  1.872e-02   2.683 0.008003 **
Dipht        5.425e-02  1.982e-02   2.737 0.006843 **
GDP          1.211e-04  3.192e-05   3.795 0.000203 ***
Popl         2.472e-08  1.439e-08   1.718 0.087513 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.312 on 174 degrees of freedom
Multiple R-squared:  0.7229,    Adjusted R-squared:  0.7118
F-statistic: 64.85 on 7 and 174 DF,  p-value: < 2.2e-16
```

Kodu przedstawia analizę reszt statystycznych (residuals) na podstawie oszacowania opartego na estymatorach.

"RS <- rstudent(Model)" - definiuje nową zmienną "RS" jako wartości R-student reszt modelu.

"plot(RS, ylab="R-Student residual", main="R-Student residual")" - tworzy wykres R-Student reszt.

"RS[abs(RS)==max(abs(RS))]" - znajduje największą wartość absolutną w zmiennej "RS".

"dim(Dane)" - zwraca wymiary macierzy "Dane".

"dim(Dane)[1]" - zwraca liczbę wierszy macierzy "Dane".

"0.05/dim(Dane)[1]*2" - oblicza prawdopodobieństwo dwustronnego testu.

"dim(Dane)[1]-4-1" - oblicza stopień swobody.

"qt(0.05/(dim(Dane)[1]*2),(dim(Dane)[1]-4-1))" - oblicza kwantyl dla dwustronnego testu.

"(-abs(RS[abs(RS)==max(abs(RS))])<qt(0.05/(dim(Dane)[1]*2),(dim(Dane)[1]-4-1)))" - sprawdza, czy największa wartość absolutna w "RS" jest mniejsza niż kwantyl dwustronnego testu.

Wynik jest “TRUE”, czyli największa wartość absolutna w "RS" jest mniejsza niż kwantyl dwustronnego testu.

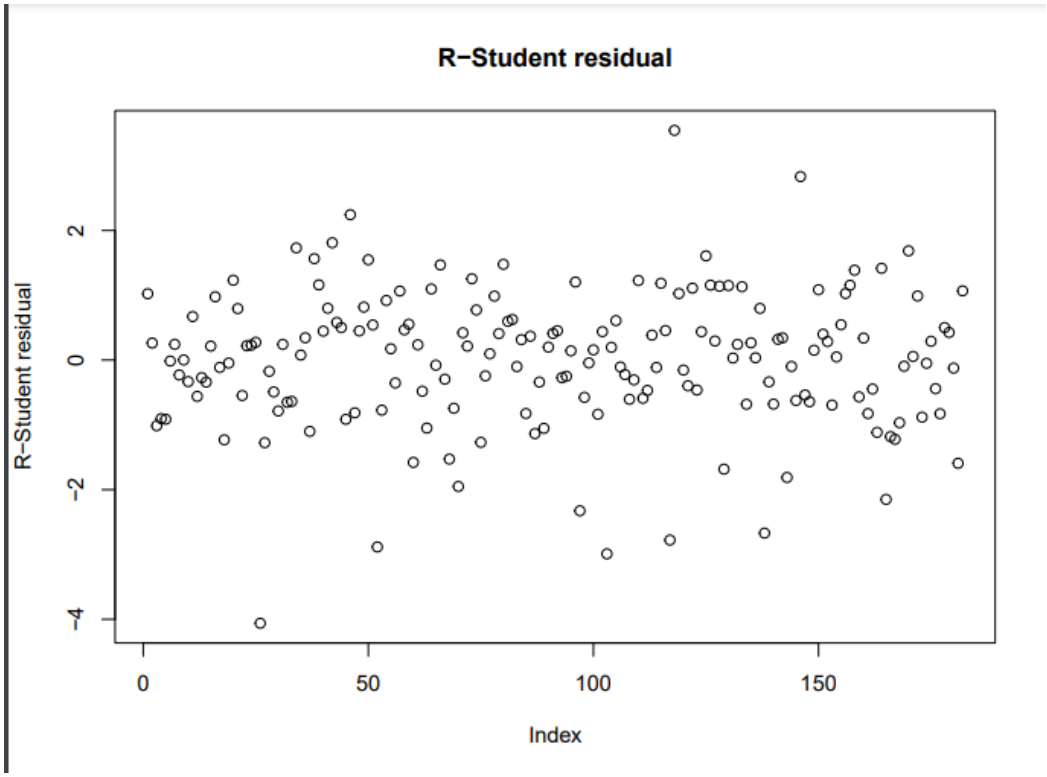
```
(RS <- rstudent(Model))
```

```
> (RS <- rstudent(Model))
          1           2           3           4           5           6
 1.023334203  0.262412754 -1.013383482 -0.903483486 -0.913156781 -0.015213520
          7           8           9          10          11          12
 0.242952203 -0.229492019  0.001261876 -0.332966301  0.671634892 -0.561060502
         13          14          15          16          17          18
-0.270781679 -0.339882040  0.214714241  0.976191332 -0.113823635 -1.230626553
         19          20          21          22          23          24
-0.046351605  1.231480550  0.793732960 -0.547296389  0.221400169  0.229791685
         25          26          27          28          29          30
 0.273818513 -4.061306753 -1.275146284 -0.173241386 -0.491721295 -0.785892147
         31          32          33          34          35          36
 0.241631341 -0.651209707 -0.638224944  1.730399228  0.076582343  0.344153583
         37          38          39          40          41          42
-1.100803597  1.564525379  1.159958707  0.445940067  0.799042649  1.809026896
         43          44          45          46          47          48
 0.580420746  0.501151287 -0.914841163  2.242821108 -0.813938531  0.449812365
         49          50          51          52          53          54
 0.817504905  1.547108690  0.541282555 -2.883727628 -0.773276590  0.918851445
         55          56          57          58          59          60
 0.173950471 -0.353599366  1.064451492  0.466158405  0.552824600 -1.578945019
         61          62          63          64          65          66
 0.234763297 -0.480943724 -1.050109286  1.096449459 -0.079188624  1.467786615
         67          68          69          70          71          72
-0.295376009 -1.528178983 -0.744291906 -1.950954126  0.420017839  0.214408844
         73          74          75          76          77          78
 1.254123467  0.773044602 -1.270032847 -0.243978236  0.096259248  0.986343745
         79          80          81          82          83          84
 0.408143258  1.477295155  0.596832705  0.626567232 -0.100794361  0.315345557
         85          86          87          88          89          90
-0.824873415  0.367743613 -1.134733535 -0.339728572 -1.053871009  0.199180650
```

```
          91           92           93           94           95           96
 0.410473037  0.452682429 -0.271775293 -0.250182884  0.142002855  1.203761565
          97           98           99          100          101          102
-2.325931428 -0.575460598 -0.044116138  0.156755345 -0.836429828  0.438391304
         103          104          105          106          107          108
-2.990215824  0.196806713  0.607760674 -0.106705083 -0.223317263 -0.605692702
         109          110          111          112          113          114
-0.306043882  1.227272336 -0.589963621 -0.469085169  0.384342768 -0.114487775
         115          116          117          118          119          120
 1.184321370  0.455125603 -2.777418278  3.543884752  1.026745667 -0.154999832
         121          122          123          124          125          126
-0.396854086  1.109353454 -0.462503565  0.437154698  1.607319731  1.155742908
         127          128          129          130          131          132
 0.293728314  1.138540223 -1.683538561  1.148938315  0.031105432  0.241567069
         133          134          135          136          137          138
 1.131679473 -0.682442145  0.266168378  0.035215507  0.796320550 -2.669991967
         139          140          141          142          143          144
-0.337072015 -0.679764799  0.320581874  0.342732468 -1.811657272 -0.098739108
         145          146          147          148          149          150
-0.624008836  2.830762674 -0.539010518 -0.642895473  0.151362342  1.085328421
         151          152          153          154          155          156
 0.398716556  0.285371202 -0.694530663  0.047931289  0.544787550  1.027560398
         157          158          159          160          161          162
 1.153994165  1.386293181 -0.568225813  0.339170801 -0.826027842 -0.446803060
         163          164          165          166          167          168
-1.115845717  1.415353364 -2.149828953 -1.180955476 -1.222909747 -0.967192297
         169          170          171          172          173          174
-0.095609992  1.684867748  0.055532210  0.988162446 -0.880918068 -0.050072552
         175          176          177          178          179          180
 0.289617284 -0.441532692 -0.828668802  0.501050943  0.425020876 -0.123891612
         181          182
-1.592907350  1.067742383
```

Przedstawienie na wykresie

```
plot(RS,ylab="R-Student residual",main="R-Student residual")
```



Uzyskanie wartości odstających.

```
RS[abs(RS)==max(abs(RS))]
```

```
> RS[abs(RS)==max(abs(RS))]
       26
-4.061307
```

```
#test statistics
dim(Dane)
dim(Dane)[1]
0.05/dim(Dane)[1]*2
dim(Dane)[1]-4-1
qt(0.05/(dim(Dane)[1]*2),(dim(Dane)[1]-4-1))
```

```
> dim(Dane)
[1] 182  11
> dim(Dane)[1]
[1] 182
> 0.05/dim(Dane)[1]*2
[1] 0.0005494505
> dim(Dane)[1]-4-1
[1] 177
> qt(0.05/(dim(Dane)[1]*2),(dim(Dane)[1]-4-1))
[1] -3.712521
```

```
(-abs(RS[abs(RS)==max(abs(RS))])<qt(0.05/(dim(Dane)[1]*2),(dim(Dane)[1]-4-1)))
```

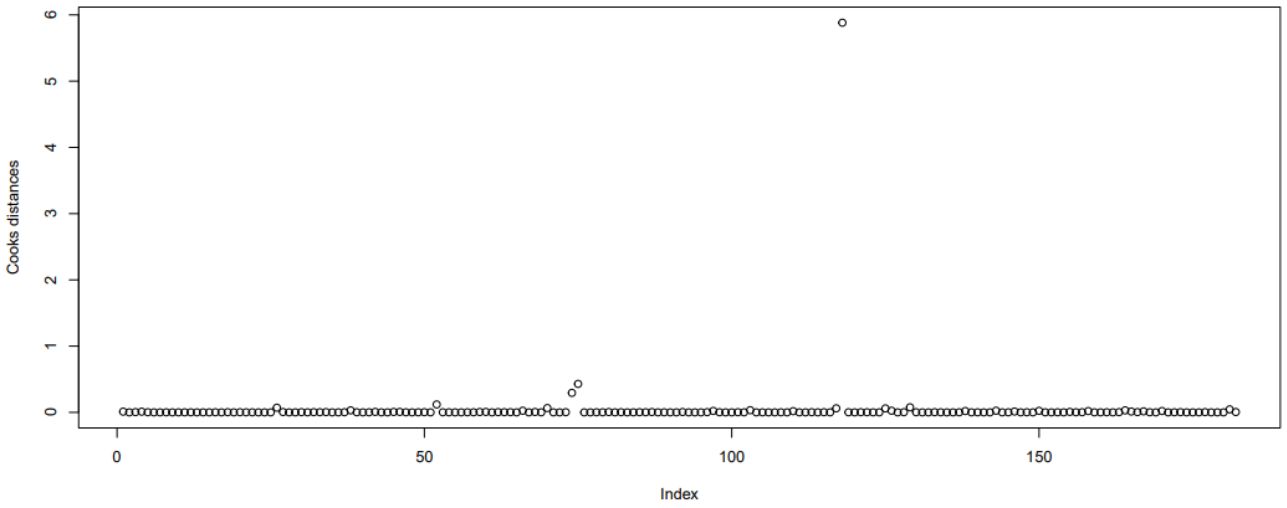
```
> (-abs(RS[abs(RS)==max(abs(RS))])<qt(0.05/(dim(Dane)[1]*2),(dim(Dane)[1]-4-1)))
   26
TRUE
```

Dzięki odległości cooka, mogę usunąć wartości odstające i stworzyć nowy model o nazwie “Model_no_outliers”. Kod wykorzystuje funkcję `cooks.distance` do obliczenia odległości Cooka dla każdego punktu danych. Odległość Cooka jest miarą, jak bardzo punkt danych wpływa na wynik modelu regresji liniowej. Wartość odległości Cooka jest wyznaczana jako iloczyn między zmienną zależną (Life) i skorygowaną estymatą modelu bez tego punktu danych. Wartości odległości Cooka powyżej 1 są uważane za "outliers", tj. punkty danych, które mają nieproporcjonalnie duży wpływ na wynik modelu. Następnie wykonywane jest wykreślenie wartości odległości Cooka, a następnie tworze nowy model regresji liniowej bez punktów danych, które są outliers. W rezultacie, wynik końcowy jest wynikiem modelu regresji liniowej bez wartości odstających.

```
#zidentyfikowanie wpływowych danych, stosujemy statystyki Cooka.
(cooks_dis <- cooks.distance(Model))
plot(cooks_dis,ylab="Cooks distances")

Model_no_outliers <-lm(Life~Adult_M+Infant_D+Under_D+Polio+Dipht+GDP+Popl,data=Dane,subset=(cooks_dis<max(cooks_dis)))
```

```
> (cooks_dis <- cooks.distance(Model))
           1            2            3            4            5
1.084358e-02 1.158679e-04 4.041842e-03 1.033076e-02 2.245975e-03
           6            7            8            9           10
6.714757e-07 8.708425e-05 9.492557e-04 1.517303e-08 1.796033e-04
          11           12           13           14           15
4.433065e-04 7.350522e-04 1.569365e-04 1.499690e-04 7.685861e-05
          16           17           18           19           20
1.557227e-03 1.463874e-05 2.546840e-03 4.061338e-06 2.160770e-03
          21           22           23           24           25
9.745246e-04 7.601425e-04 6.239910e-05 8.403537e-05 6.198951e-05
          26           27           28           29           30
7.059110e-02 5.146900e-03 3.544015e-05 3.074601e-04 3.036962e-03
          31           32           33           34           35
5.586022e-04 2.971755e-03 2.307590e-03 4.006873e-03 3.053602e-05
          36           37           38           39           40
4.743459e-04 2.171488e-03 3.032421e-02 2.001798e-03 2.217774e-04
          41           42           43           44           45
1.148955e-03 6.854474e-03 4.264796e-04 2.664669e-04 6.059988e-03
          46           47           48           49           50
6.830475e-03 1.129106e-03 1.611891e-04 1.043956e-03 2.760880e-03
          51           52           53           54           55
3.194454e-04 1.194298e-01 1.451466e-03 1.059199e-03 1.317752e-04
          56           57           58           59           60
1.898576e-04 1.694939e-03 1.352787e-03 5.445132e-03 6.684669e-03
          61           62           63           64           65
5.857327e-05 3.574510e-03 2.628266e-03 2.569498e-03 6.967644e-06
          66           67           68           69           70
2.505006e-02 3.121076e-04 5.797469e-03 8.629252e-04 6.519839e-02
          71           72           73           74           75
2.265100e-04 5.534666e-05 2.819776e-03 2.942617e-01 4.297374e-01
          76           77           78           79           80
7.765277e-05 3.144929e-05 1.405190e-03 1.037284e-03 4.876417e-03
          81           82           83           84           85
3.273077e-04 2.235635e-03 1.675840e-05 2.216640e-04 2.464354e-03
          86           87           88           89           90
1.889457e-03 4.743857e-03 1.333189e-04 1.179606e-03 4.804916e-05
          91           92           93           94           95
2.710941e-04 5.348596e-03 2.247925e-04 6.615451e-05 2.661073e-05
```

```
          96           97           98           99          100
3.096057e-03 2.188943e-02 1.773639e-03 2.081422e-06 3.897792e-05
         101          102          103          104          105
2.152629e-03 5.399223e-04 3.302102e-02 4.206278e-05 3.321832e-04
         106          107          108          109          110
1.328257e-05 1.027624e-04 9.295949e-04 1.390226e-04 1.826201e-02
         111          112          113          114          115
4.237468e-04 4.554140e-04 2.388011e-03 1.274920e-04 2.201075e-03
         116          117          118          119          120
3.247714e-04 6.103057e-02 5.882092e+00 1.626442e-03 3.471825e-05
         121          122          123          124          125
2.344821e-03 1.830146e-03 5.171257e-04 2.358737e-04 5.984783e-02
         126          127          128          129          130
2.495201e-02 2.016877e-04 2.352226e-03 7.660300e-02 2.083046e-03
         131          132          133          134          135
8.080530e-07 6.191294e-05 2.459306e-03 1.123319e-03 8.086594e-05
         136          137          138          139          140
1.403479e-06 1.259203e-03 2.029298e-02 1.730489e-04 5.820324e-04
         141          142          143          144          145
1.073002e-04 1.683776e-04 2.505885e-02 1.529818e-04 9.258432e-04
         146          147          148          149          150
1.282444e-02 4.503165e-04 1.881848e-03 8.713101e-05 2.343858e-02
         151          152          153          154          155
7.590964e-04 1.035859e-04 1.344452e-03 2.241222e-06 7.078227e-03
         156          157          158          159          160
1.847264e-03 2.119699e-03 1.997196e-02 4.627543e-04 1.352773e-04
         161          162          163          164          165
1.739250e-03 2.357439e-04 3.401139e-03 3.024241e-02 1.269319e-02
         166          167          168          169          170
4.242884e-03 1.544484e-02 1.758608e-03 2.480695e-05 2.067676e-02
         171          172          173          174          175
5.010622e-06 1.404717e-03 3.072742e-03 6.328392e-06 1.041605e-04
         176          177          178          179          180
3.131750e-04 3.372494e-03 2.059599e-04 1.914949e-04 2.645504e-05
         181          182
4.527136e-02 5.129188e-03
```



“Model_no_outliers” ma lepsze wyniki niż Model

```
summary(Model_no_outliers)
```

```
> summary(Model_no_outliers)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP + Popl, data = Dane, subset = (cooks_dis < max(cooks_dis)))

Residuals:
     Min       1Q   Median       3Q      Max
-14.2143  -2.3349   0.1912   2.2708  11.9048

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.042e+01  1.714e+00  41.099  < 2e-16 ***
Adult_M     -4.508e-02  3.816e-03 -11.813  < 2e-16 ***
Infant_D     2.787e-01  6.307e-02   4.418 1.75e-05 ***
Under_D     -2.341e-01  5.158e-02  -4.539 1.05e-05 ***
Polio        4.282e-02  1.824e-02   2.347 0.020060 *
Dipht        5.017e-02  1.923e-02   2.609 0.009875 **
GDP          1.194e-04  3.091e-05   3.863 0.000158 ***
Popl         1.072e-08  1.448e-08   0.740 0.460100
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.175 on 173 degrees of freedom
Multiple R-squared:  0.7349,    Adjusted R-squared:  0.7242
F-statistic: 68.51 on 7 and 173 DF,  p-value: < 2.2e-16
```

Kod tworzy wykres diagnostycznych reszt dla modelu statystycznego.

"par(mfrow=c(2,1))" - ustawia parametr okna, aby rysować dwa wykresy na jednej stronie.

"plot(Model$res, ylab="Residuals", main="Index plot of residuals")" - tworzy wykres indexu reszt dla modelu "Model".

"abline(h=0, col="red")" - dodaje linie na wykresie, reprezentującą wartość 0, w kolorze czerwonym.

"plot(Model$fit, Model$res, xlab="Fitted", ylab="Residuals")" - tworzy wykres reszt, gdzie na osi x jest "Fitted", a na osi y jest "Residuals".

"abline(h=0, col="red")" - dodaje linie na wykresie, reprezentującą wartość 0, w kolorze czerwonym.

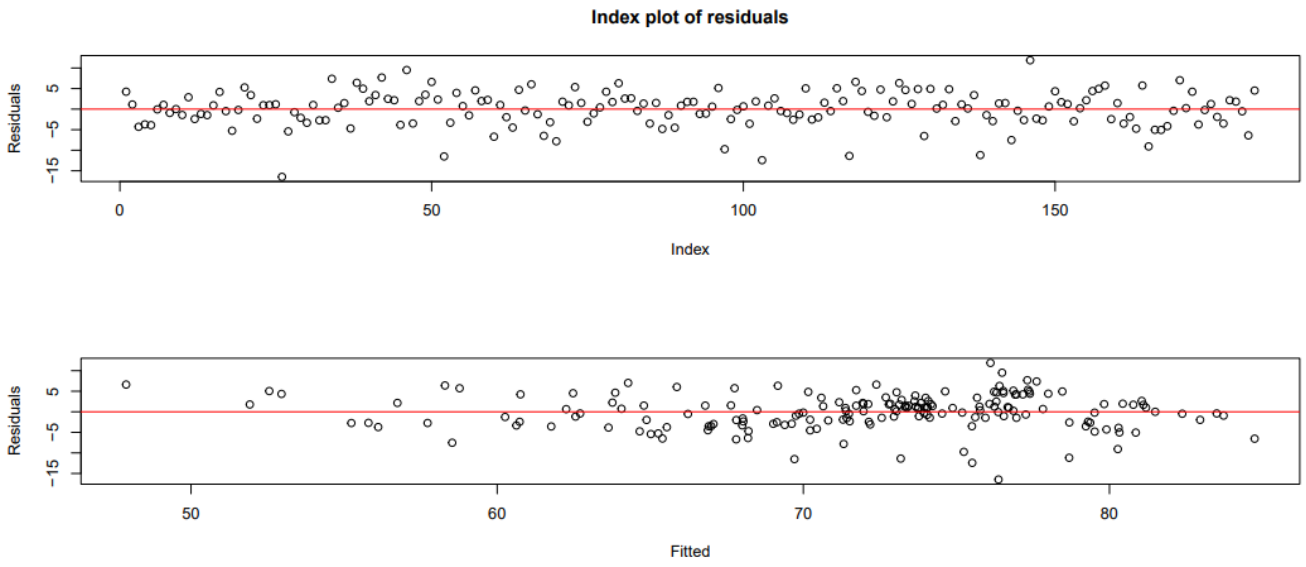
"plot(Model_no_outliers$res, ylab="Residuals", main="Index plot of residuals")" - tworzy wykres indexu reszt dla modelu "Model_no_outliers".

"abline(h=0, col="red")" - dodaje linie na wykresie, reprezentującą wartość 0, w kolorze czerwonym.

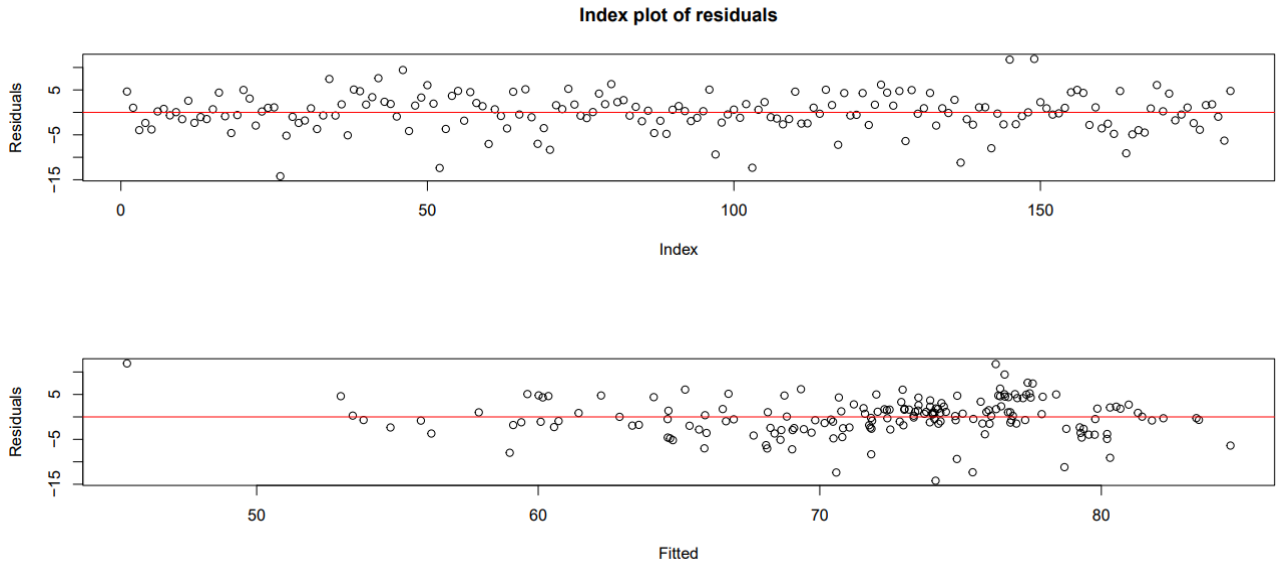
"plot(Model_no_outliers$fit, Model_no_outliers$res, xlab="Fitted", ylab="Residuals")" - tworzy wykres skojarzony reszt, gdzie na osi x jest "Fitted", a na osi y jest "Residuals", dla modelu "Model_no_outliers".

"abline(h=0, col="red")" - dodaje linie na wykresie, reprezentującą wartość 0, w kolorze czerwonym.

```
#Diagnostic
par(mfrow=c(2,1))
plot(Model$res,ylab="Residuals",main="Index plot of residuals")
abline(h=0,col="red")
plot(Model$fit,Model$res,xlab="Fitted",ylab="Residuals")
abline(h=0,col="red")
```




```
par(mfrow=c(2,1))
plot(Model_no_outliers$res,ylab="Residuals",main="Index plot of residuals")
abline(h=0,col="red")
plot(Model_no_outliers$fit,Model_no_outliers$res,xlab="Fitted",ylab="Residuals")
abline(h=0,col="red")
```



Test rozkładu normalnego zacząłem od graficznego przedstawienia wszystkich zmiennych w modelu. Za pomocą funkcji “boxplot” oraz “qqnorm”.

Kod przedstawia.

"boxplot(Dane$Life)" - rysuje wykres typu boxplot, który pokazuje statystyki centralne i rozrzut danych dla zmiennej "Life".

"qqnorm(Dane$"Life")" - rysuje wykres typu Q-Q plot, który pokazuje, jak bardzo dane odpowiadają rozkładowi normalnemu.

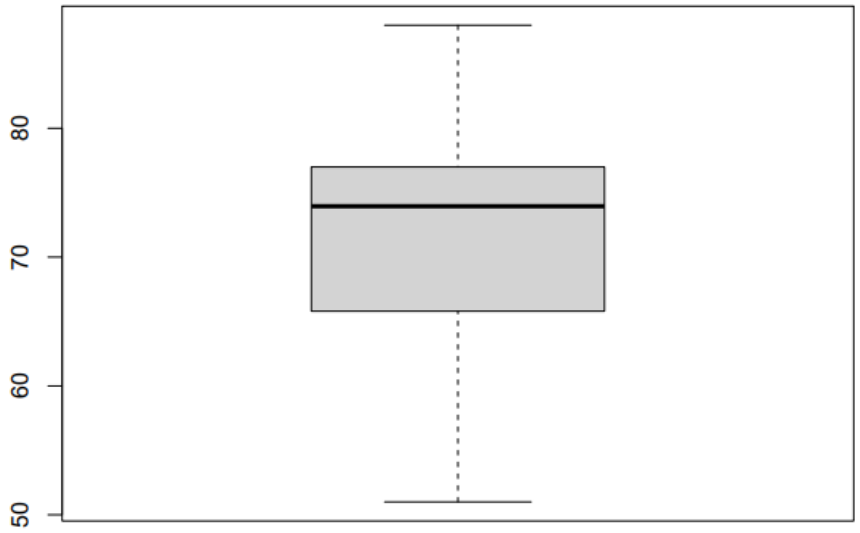
"qqline(Dane$"Life")" - dodaje linię do wykresu typu Q-Q plot, która pokazuje, jak dane powinny odpowiadać rozkładowi normalnemu.

Jeśli dane są zgodne z rozkładem normalnym, to na wykresie typu Q-Q plot powinny być one rozłożone wokół linii. Jeśli nie są, oznacza to, że dane nie odpowiadają rozkładowi normalnemu.

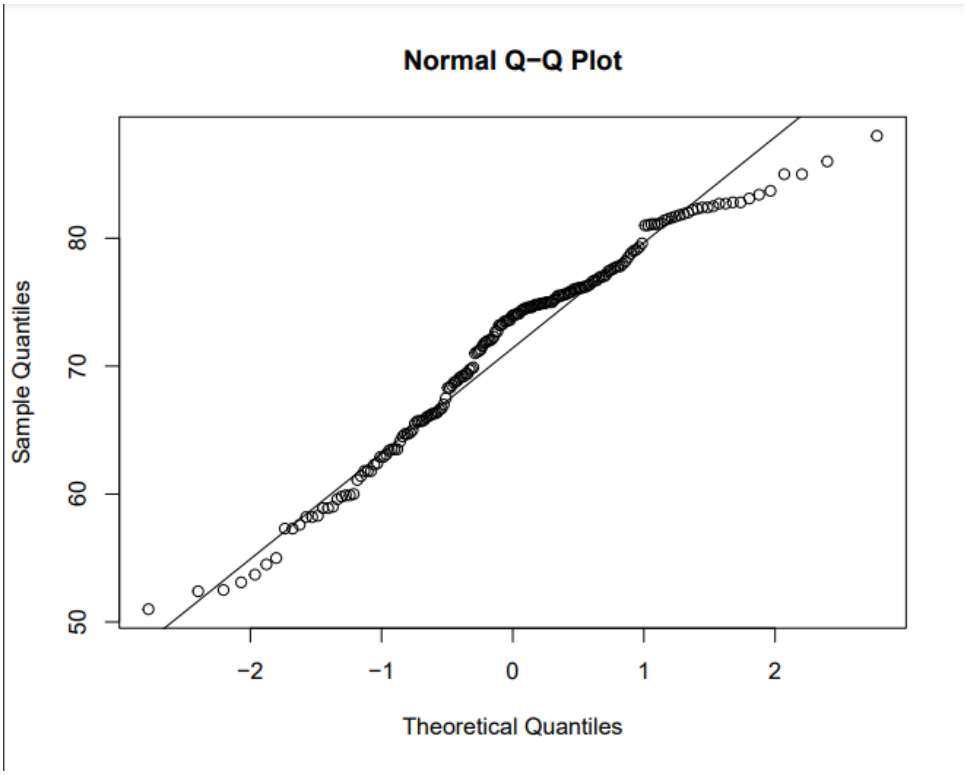
Wykres pudełkowy składa się z osi. Nad osią umieszczony jest prostokąt (pudełko), którego lewy bok jest wyznaczony przez pierwszy kwartyl, zaś prawy bok przez trzeci kwartyl. Szerokość pudełka odpowiada wartości rozstępu ćwiartkowego. Wewnątrz prostokąta znajduje się pionowa linia, określająca wartość mediany.

W pierwszym przykładzie można zauważyć, że dane z kolumny “Life” całkiem dobrze wypadają na rozkładzie normalnym.

```
boxplot(Dane$Life)
```

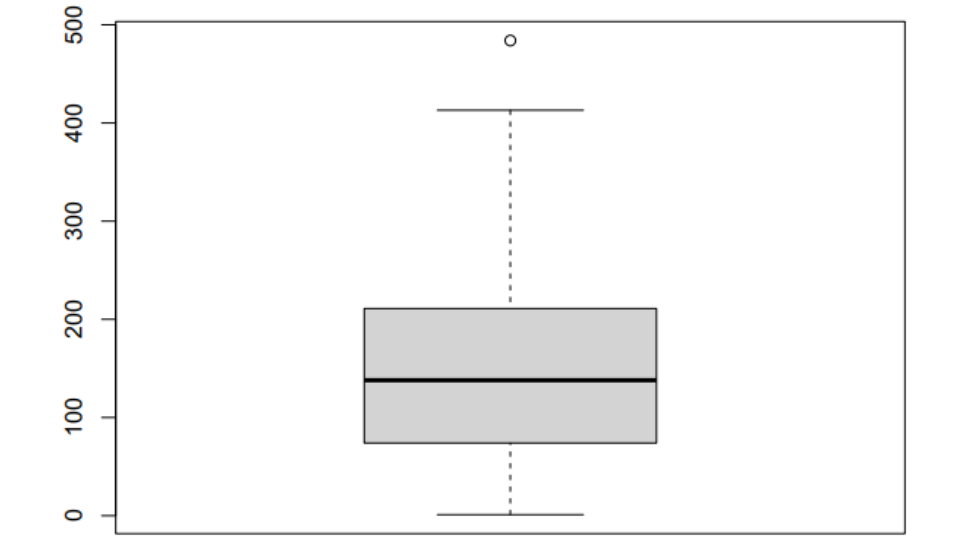



```
qqnorm(Dane$"Life" )
qqline(Dane$"Life" )
```

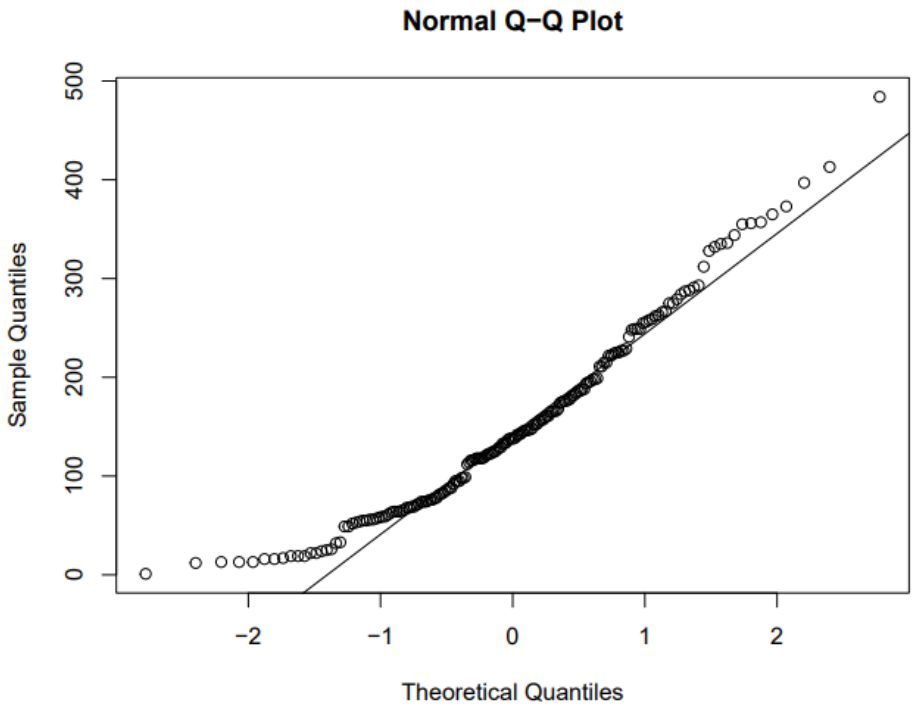


Rozkład normlany jest gorszy niż w pierwszym przykładzie, ale również dobry.

```
boxplot(Dane$Adult_M)
```

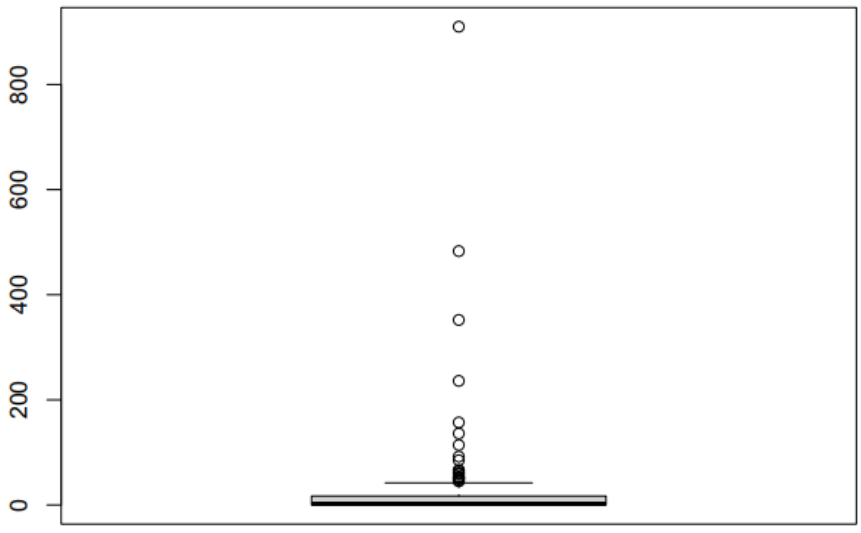



```
qqnorm(Dane$"Adult_M" )
qqline(Dane$"Adult_M" )
```

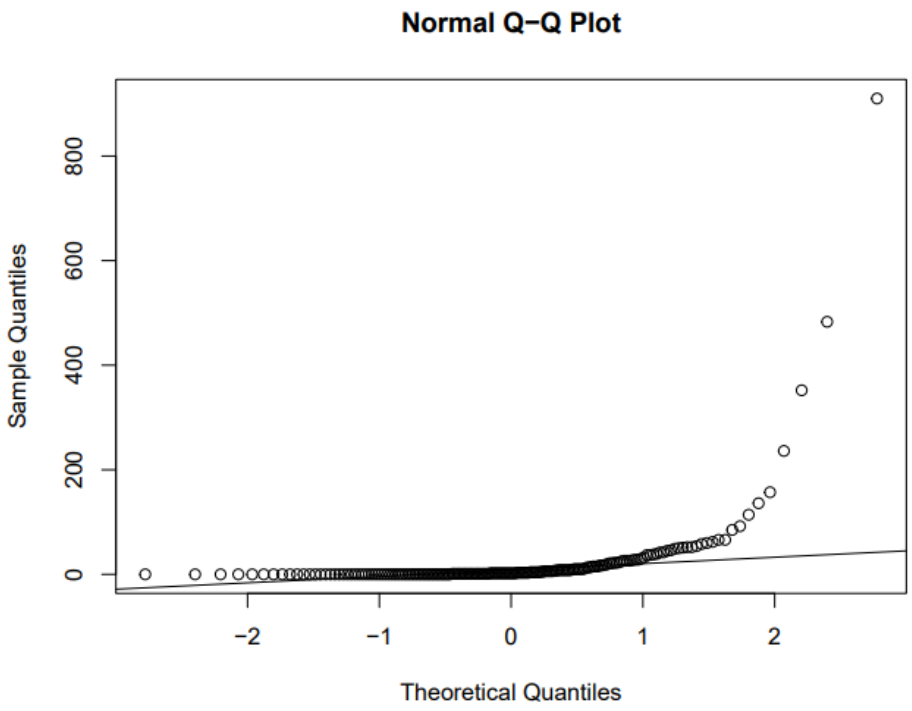


Brak rozkładu normalnego.

```
boxplot(Dane$Infant_D)
```

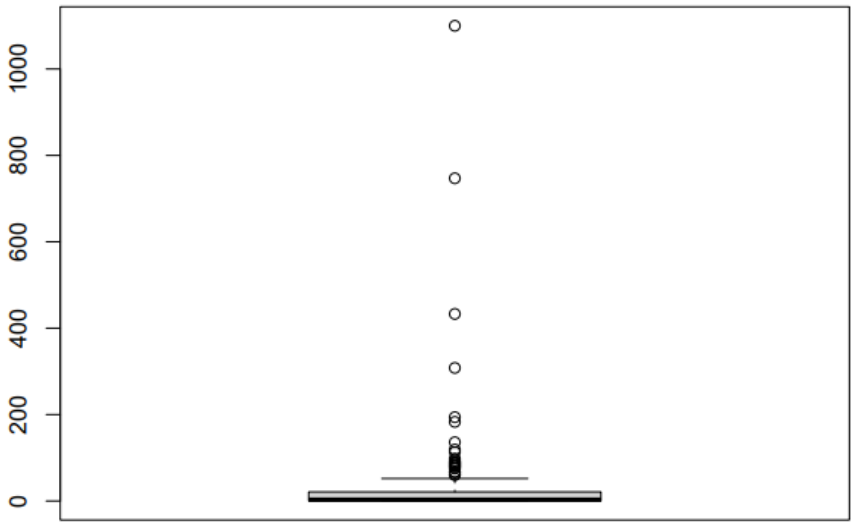



```
qqnorm(Dane$"Infant_D")
qqline(Dane$"Infant_D")
```

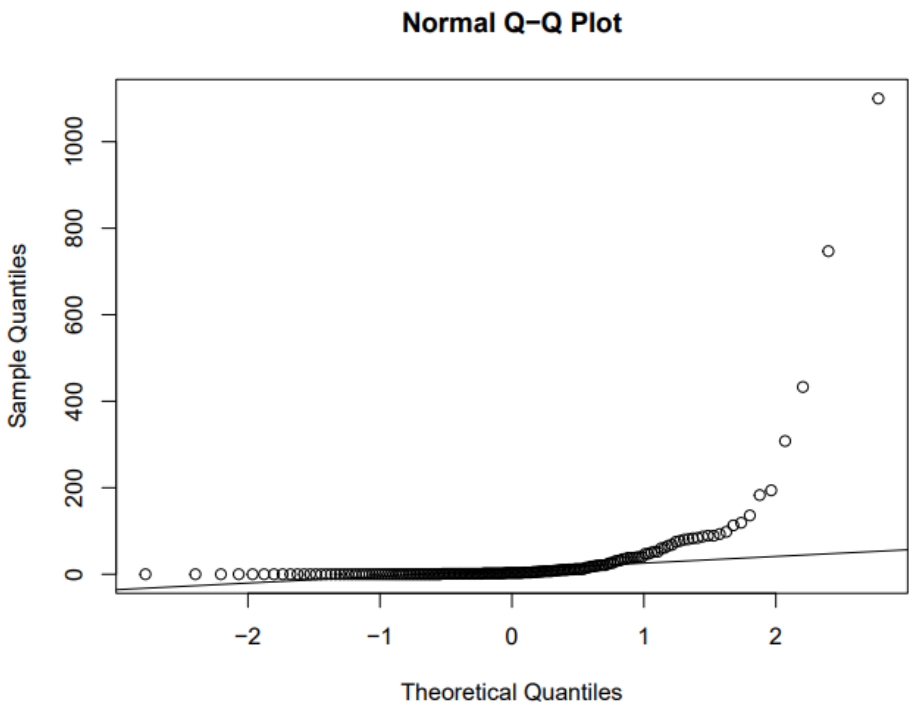


Brak rozkładu normlanego.

```
boxplot(Dane$Under_D)
```

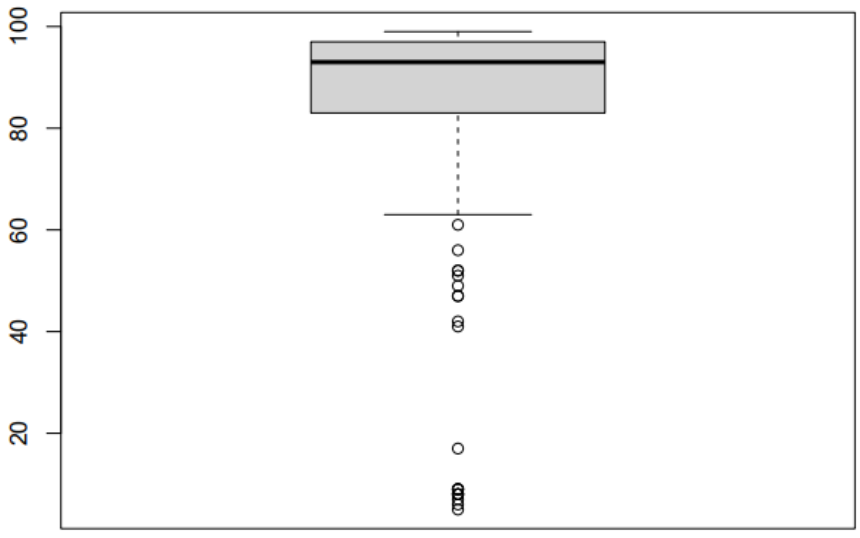



```
qqnorm(Dane$"Under_D")
qqline(Dane$"Under_D")
```

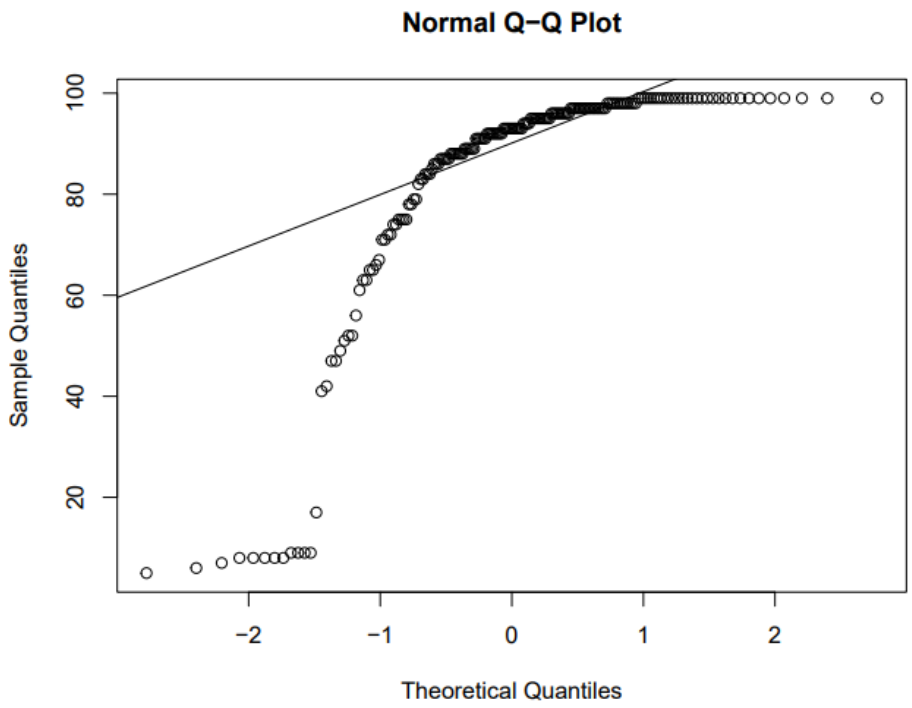


Brak rozkładu normlanego.

```
boxplot(Dane$Polio)
```

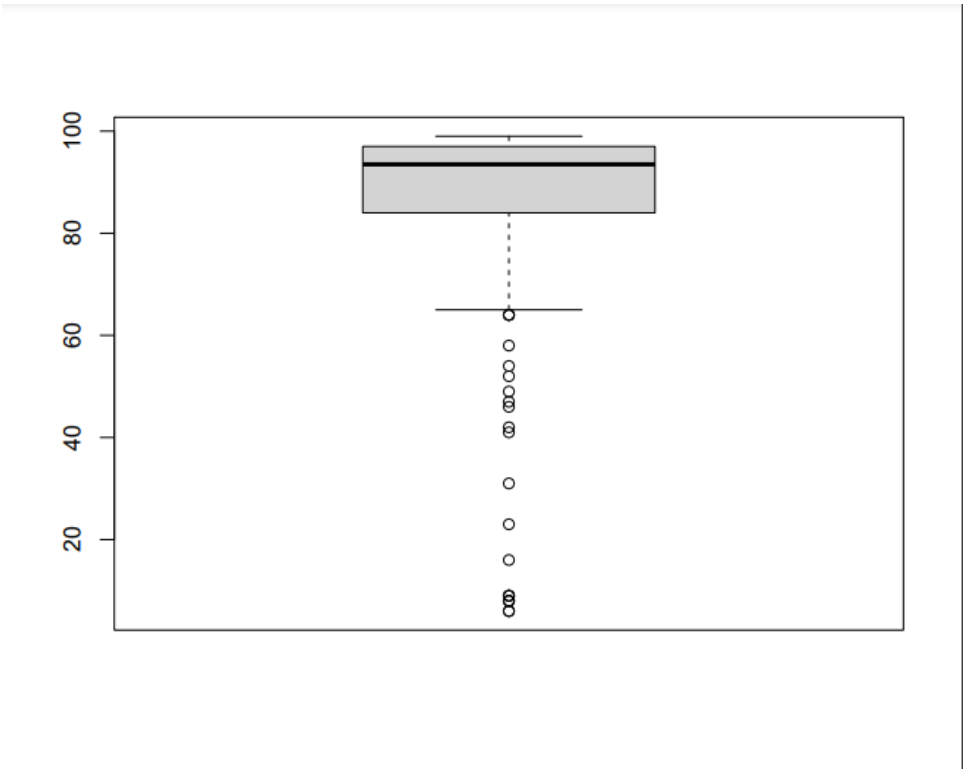



```
qqnorm(Dane$"Polio")
qqline(Dane$"Polio")
```

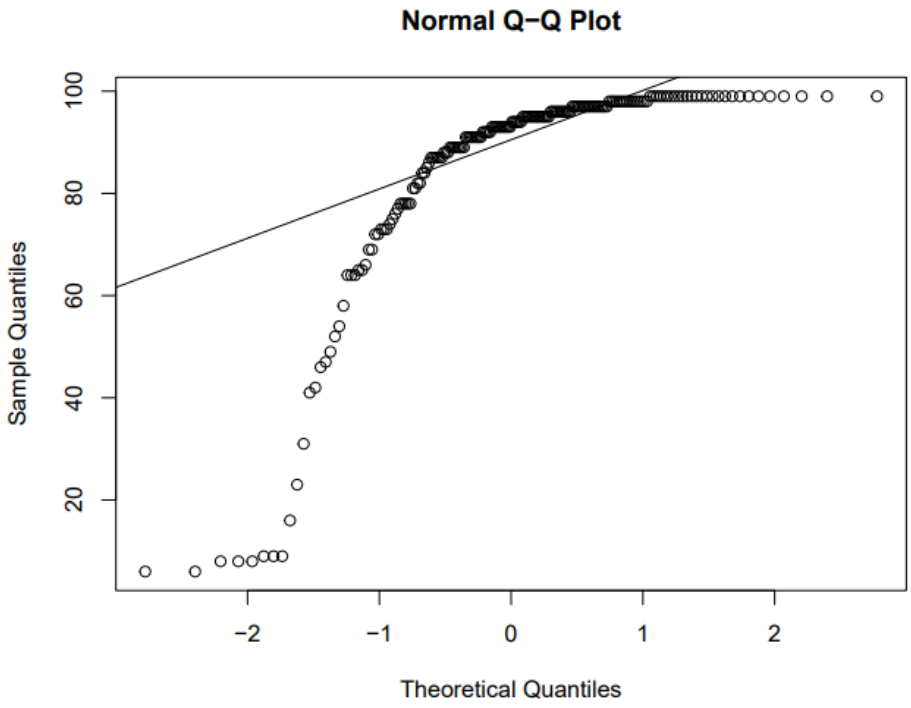


Brak rozkładu normalnego.

```
boxplot(Dane$Dipht)
```

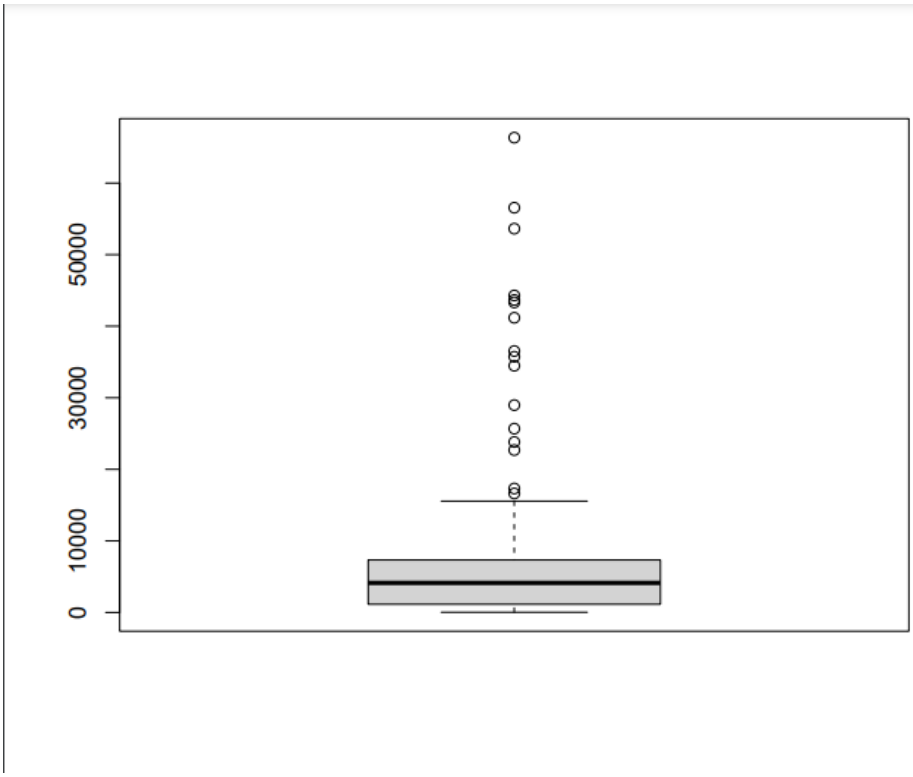



```
qqnorm(Dane$"Dipht")
qqline(Dane$"Dipht")
```

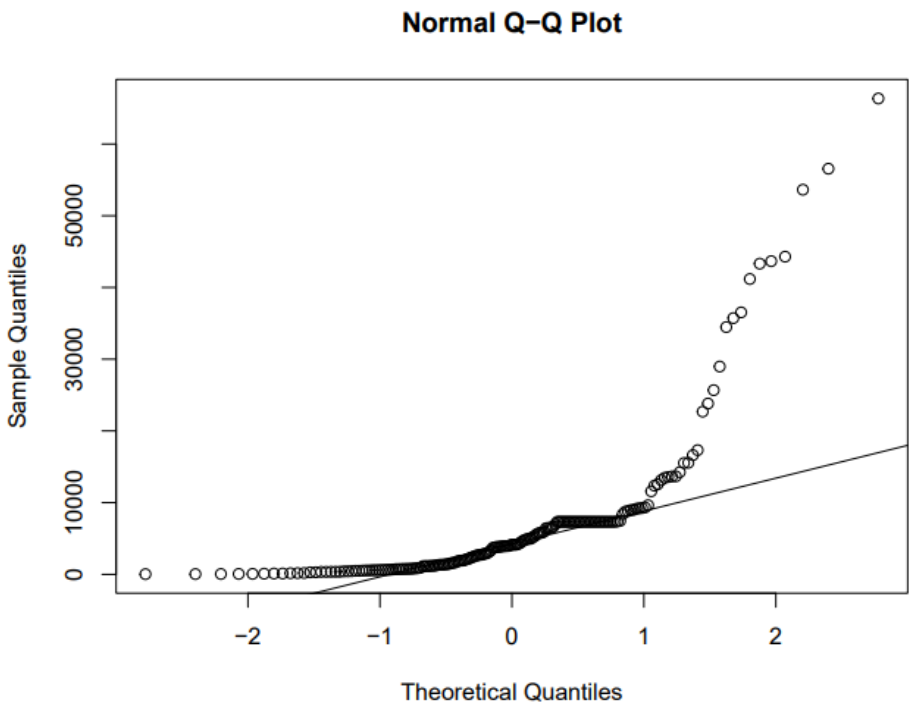


Brak rozkładu normalnego.

```
boxplot(Dane$GDP)
```

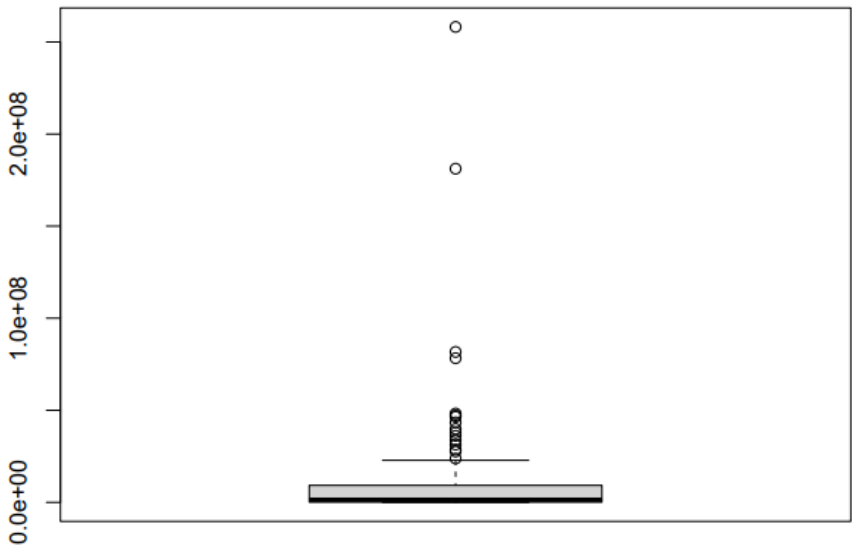



```
qqnorm(Dane$"GDP")
qqline(Dane$"GDP")
```

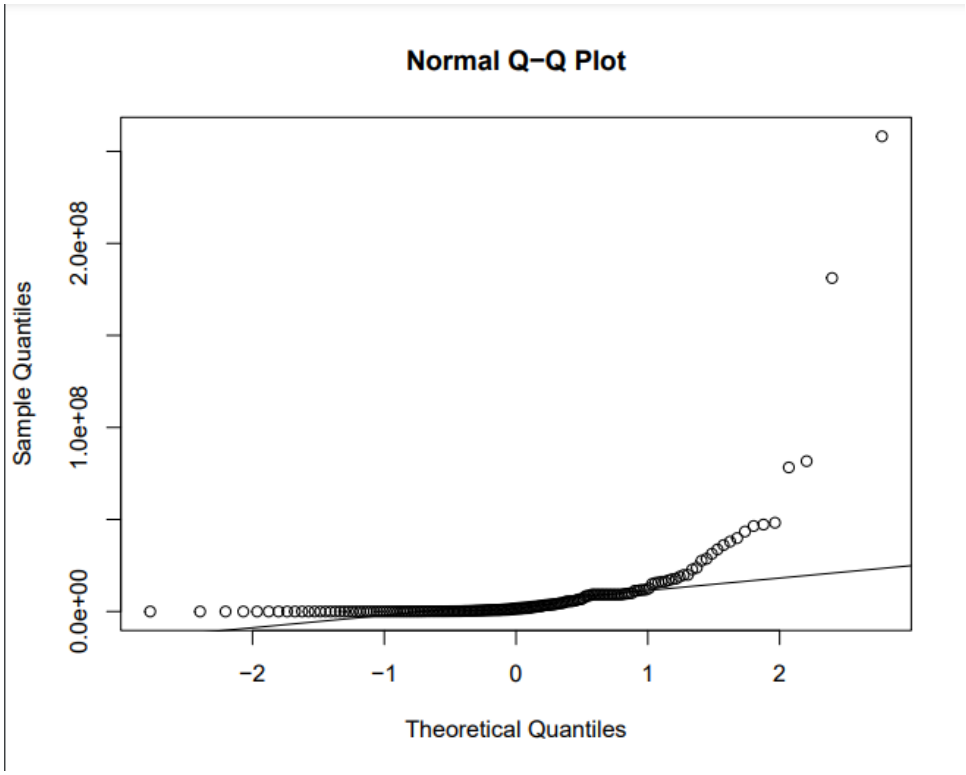


Brak rozkładu normalnego.

```
boxplot(Dane$Popl)
```

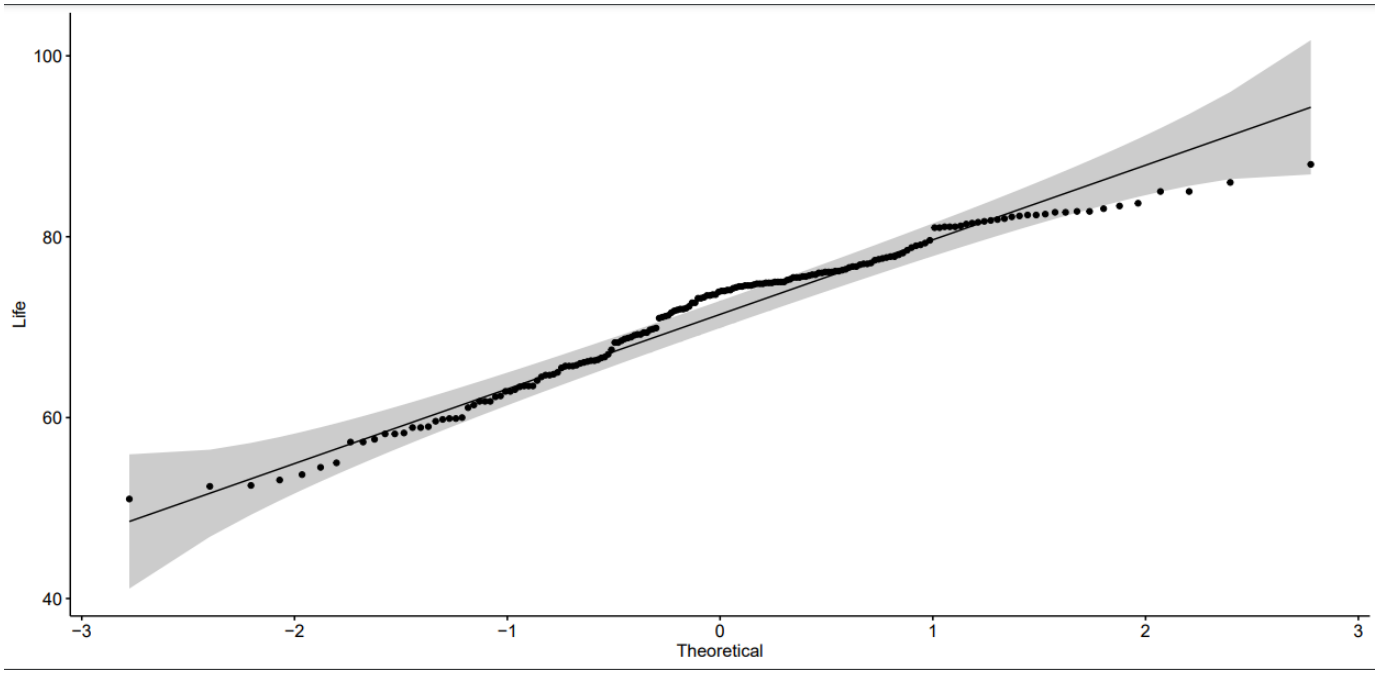


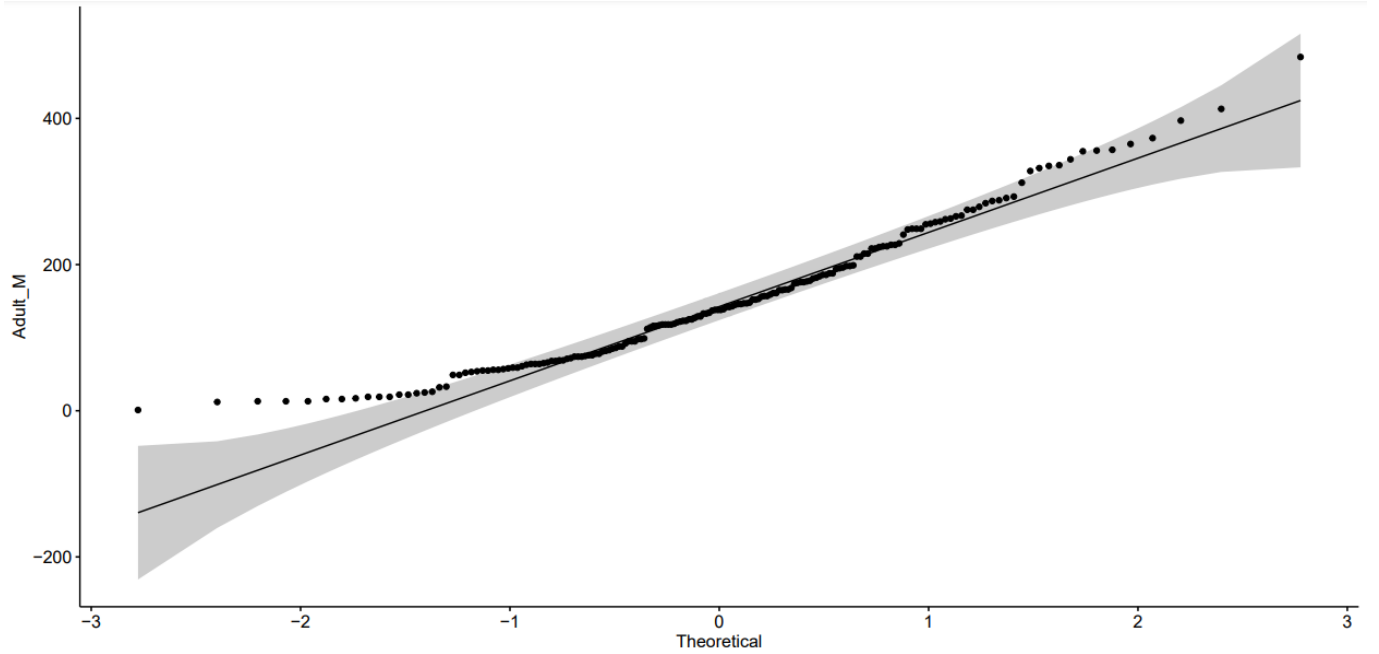

```
qqnorm(Dane$"Popl")
qqline(Dane$"Popl")
```

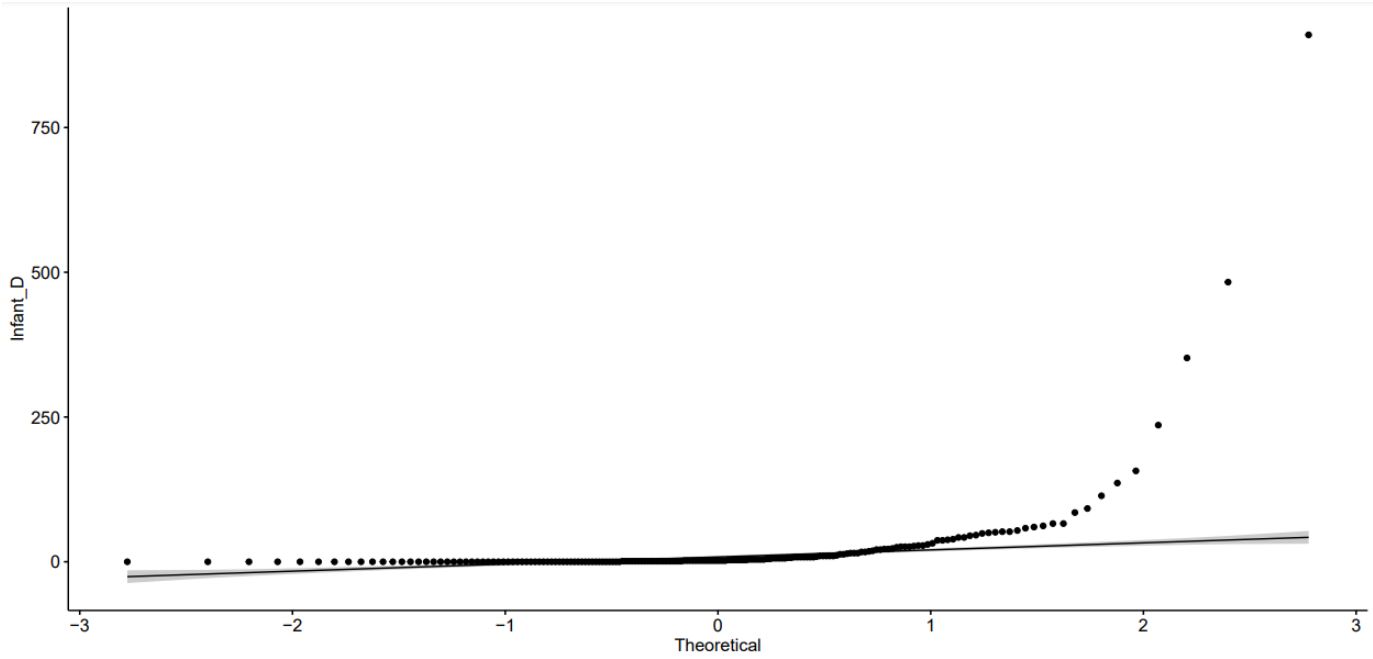


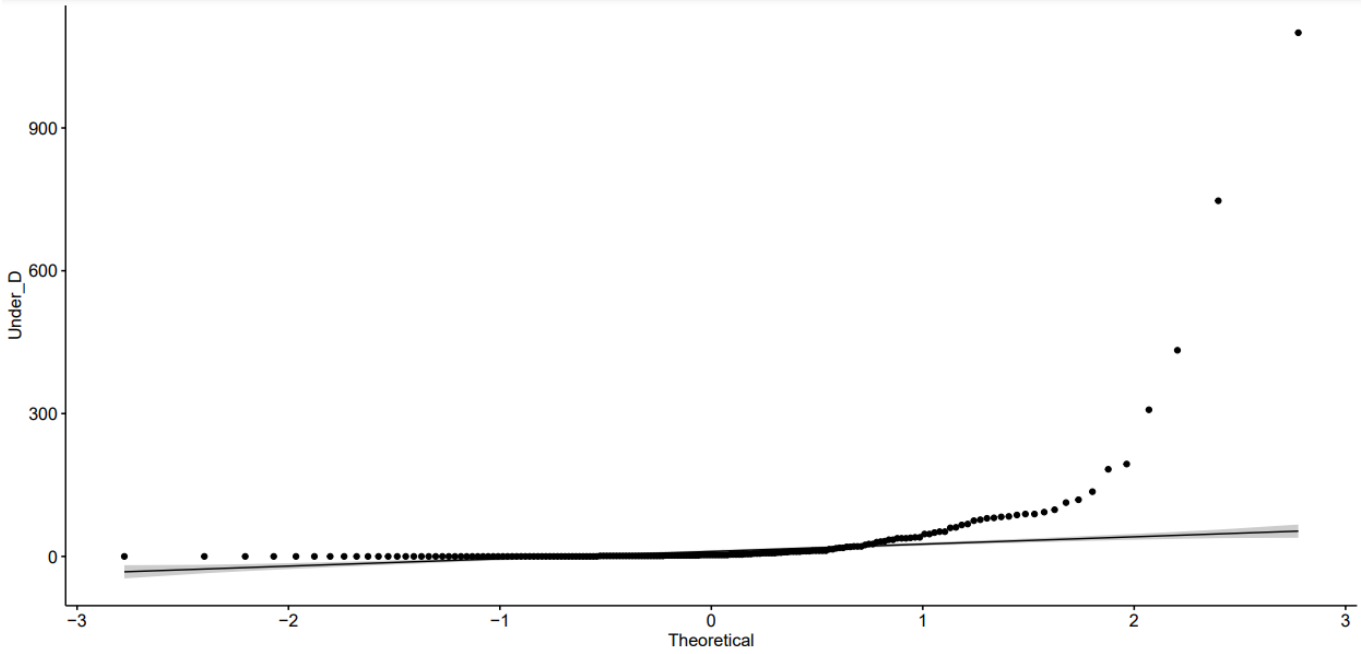
Również przedstawiłem graficznie to samo za pomocą pakietu “ggpubr”, oraz funkcji “ggqqplot”. Wykres Q-Q jest rysowany między daną próbą a rozkładem normalnym. Wykreślana jest również 45-stopniowa linia odniesienia, aby ocenić, jak bliskie są wartości próbki rozkładowi normalnemu.

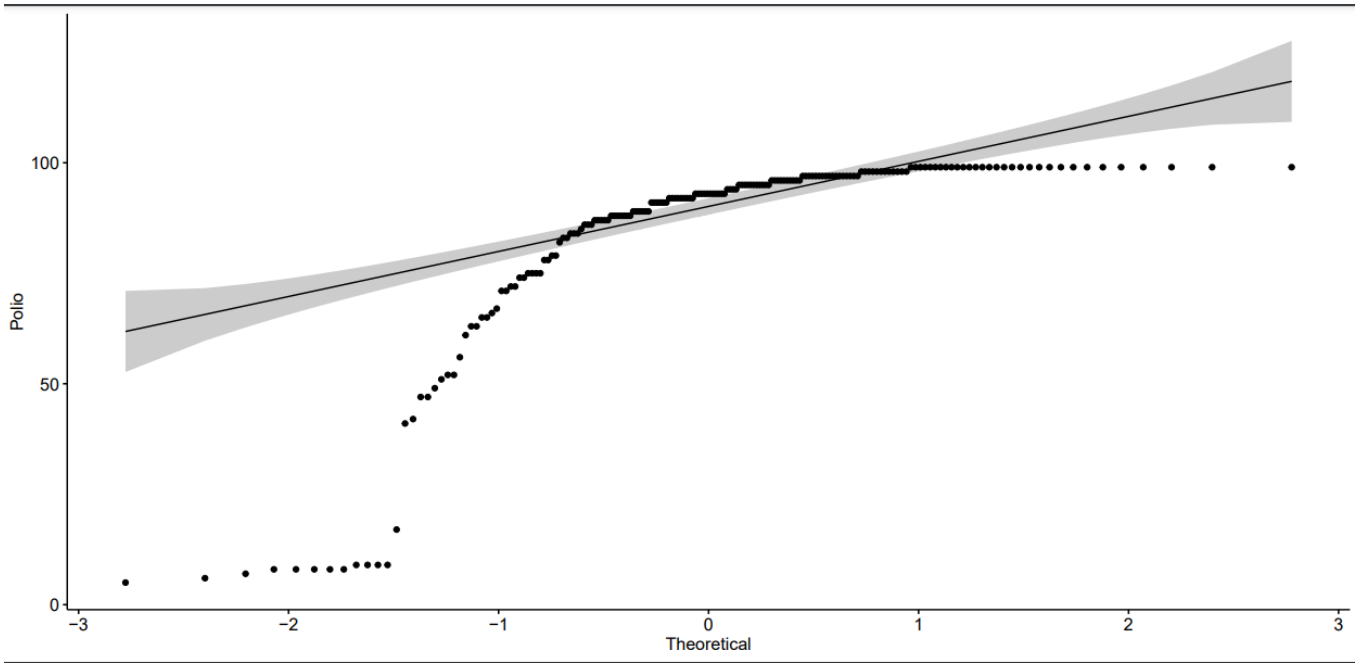
```
library("ggpubr")
ggqqplot(Dane$Life, ylab = "Life")
ggqqplot(Dane$Adult_M, ylab = "Adult_M")
ggqqplot(Dane$Infant_D, ylab = "Infant_D")
ggqqplot(Dane$Under_D, ylab = "Under_D")
ggqqplot(Dane$Polio, ylab = "Polio")
ggqqplot(Dane$Dipht, ylab = "Dipht")
ggqqplot(Dane$GDP, ylab = "GDP")
ggqqplot(Dane$Popl, ylab = "Popl")
```

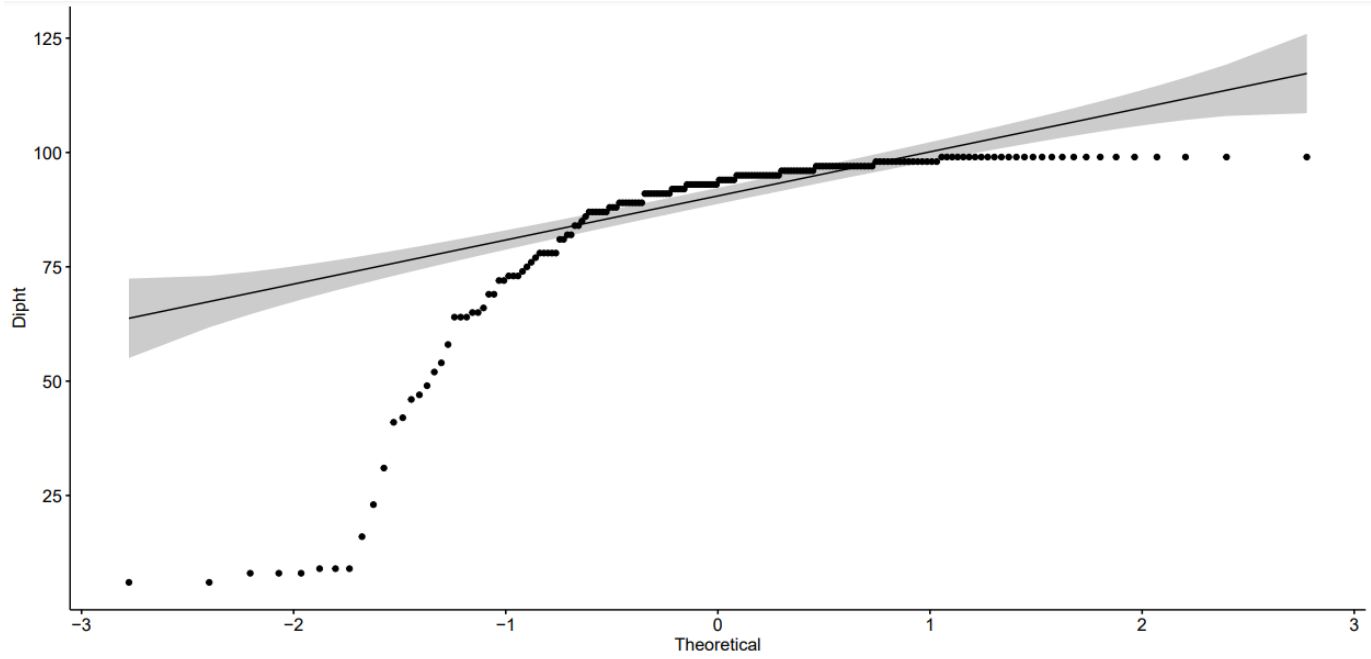

Life
100
80
60
40
−3
−2
−1
0
1
2
3
Theoretical

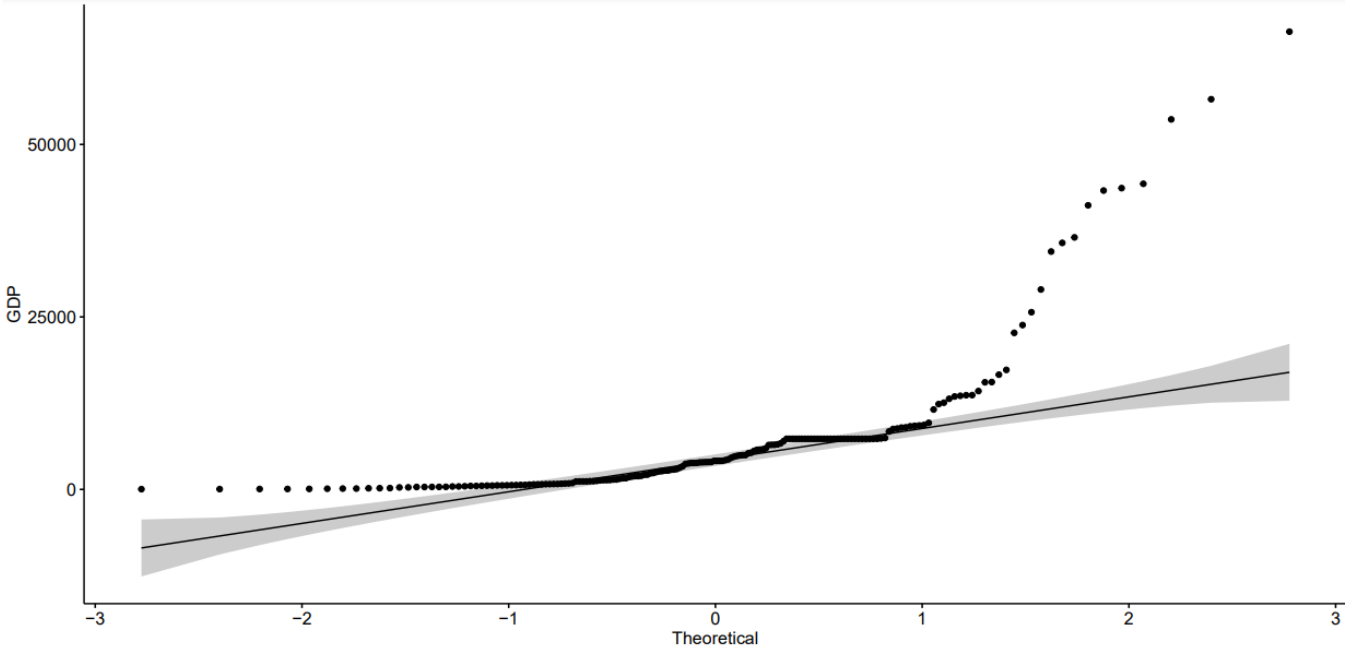

Adult_M
400
200
0
−200
−3
−2
−1
0
1
2
3
Theoretical

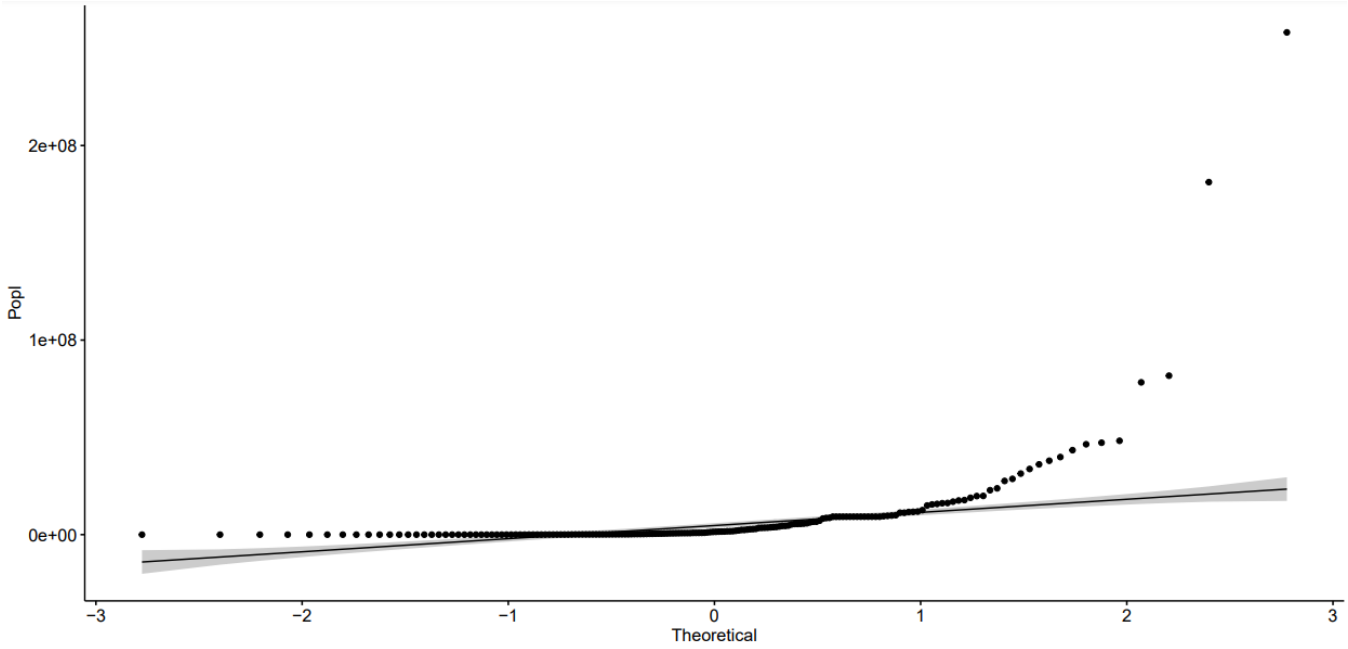

750
500
250
0
Infant_D
−3
−2
−1
0
1
2
3
Theoretical


900
600
300
0
Under_D
−3
−2
−1
0
1
2
3
Theoretical


Polio
100
50
0
−3
−2
−1
0
1
2
3
Theoretical


Dipht
125
100
75
50
25
−3
−2
−1
0
1
2
3
Theoretical





Przedstawienie na wykresie rozkładu normlanego modeli “Model” oraz “Model_no_outliers”.

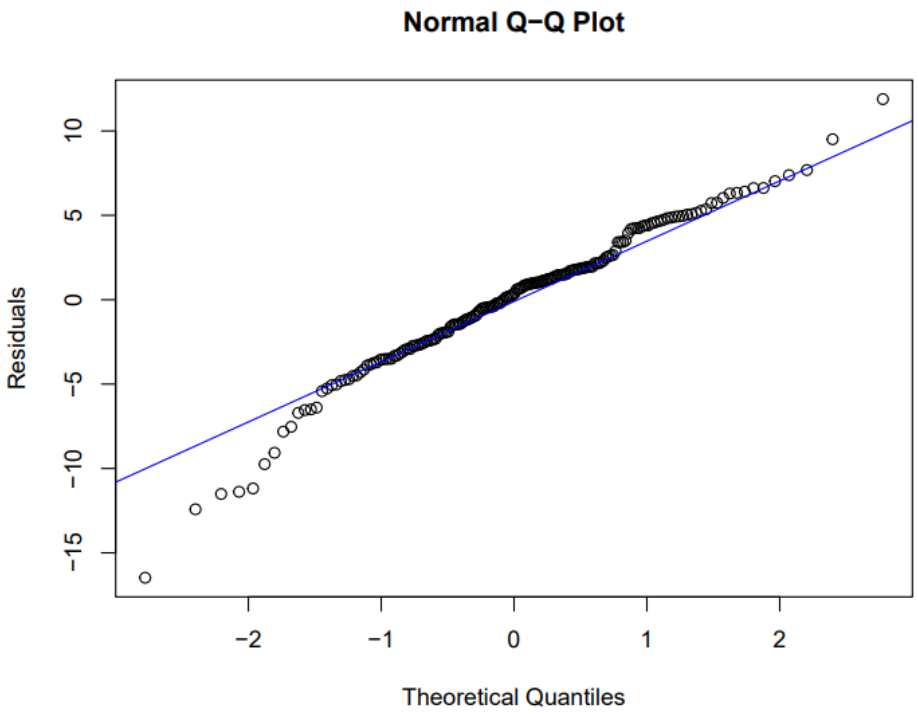
Kod pokazuje wizualizację danych i sprawdza, czy reszty regresji są zgodne z rozkładem normalnym.

"par(mfrow=c(1,1))" oznacza, że na każdej stronie wyników będzie tylko jeden wykres.

"qqnorm(Model$res,ylab="Residuals")" - rysuje wykres typu Q-Q plot dla reszt regresji Model, z opisem osi y jako "Residuals".

"qqline(Model$res, col="blue")" - dodaje linię do wykresu typu Q-Q plot, która pokazuje, jak dane powinny odpowiadać rozkładowi normalnemu. Kolor linii to niebieski.

```
par(mfrow=c(1,1))
qqnorm(Model$res,ylab="Residuals")
qqline(Model$res, col="blue")
```



Przedstawienie na wykresie rozkładu normlanego modeli “Model” oraz “Model_no_outliers” z użyciem funkcji “rstudent”.

"qqnorm(Model_no_outliers$res,ylab="Residuals")" - rysuje wykres typu Q-Q plot dla reszt regresji Model_no_outliers, z opisem osi y jako "Residuals".

"qqline(Model_no_outliers$res, col="blue")" - dodaje linię do wykresu typu Q-Q plot, która pokazuje, jak dane powinny odpowiadać rozkładowi normalnemu. Kolor linii to niebieski.

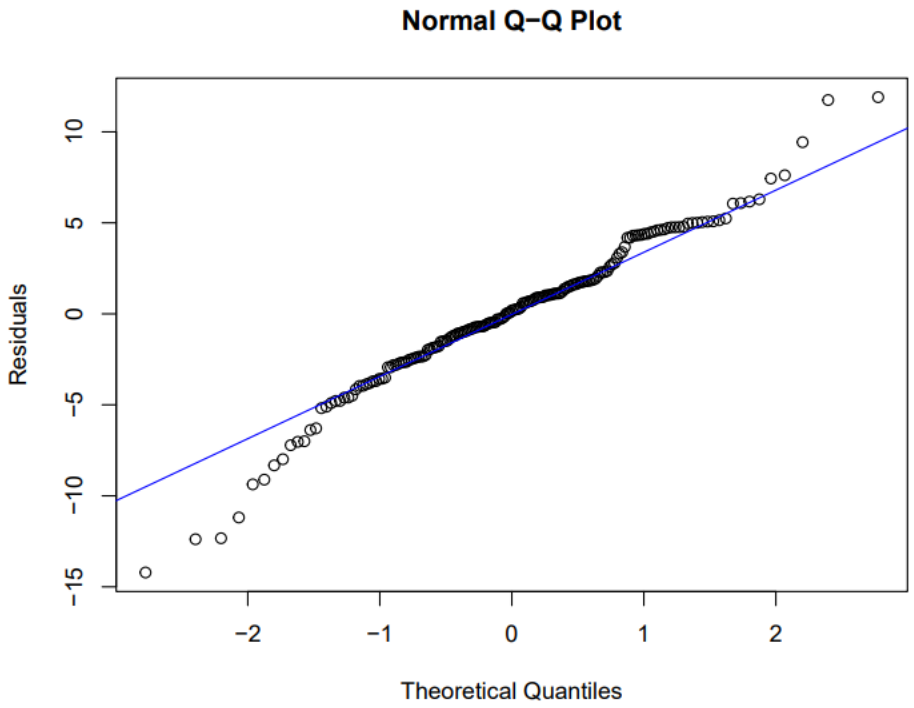
"qqnorm(rstudent(Model),ylab="Studentized residuals")" - rysuje wykres typu Q-Q plot dla studentyzowanych reszt regresji Model, z opisem osi y jako "Studentized residuals".

"abline(0,1)" - dodaje prostą do wykresu typu Q-Q plot, która pokazuje, jak dane powinny odpowiadać rozkładowi normalnemu.

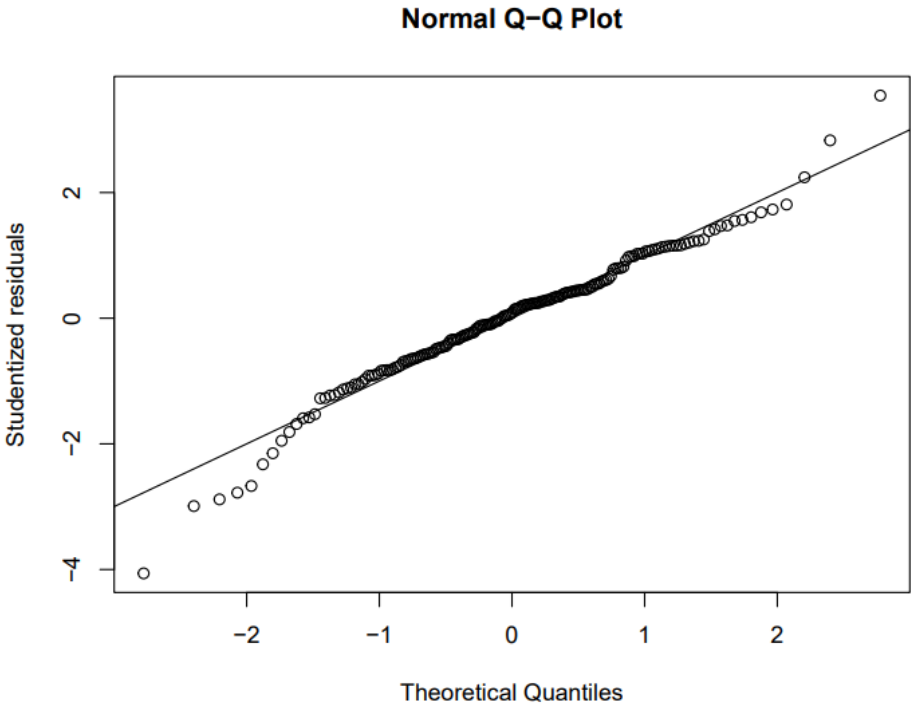
"qqnorm(rstudent(Model_no_outliers),ylab="Studentized residuals")" - rysuje wykres typu Q-Q plot dla studentyzowanych reszt regresji Model_no_outliers, z opisem osi y jako "Studentized residuals".

"abline(0,1)" - dodaje prostą do wykresu typu Q-Q plot, która pokazuje, jak dane powinny odpowiadać rozkładowi normalnemu.

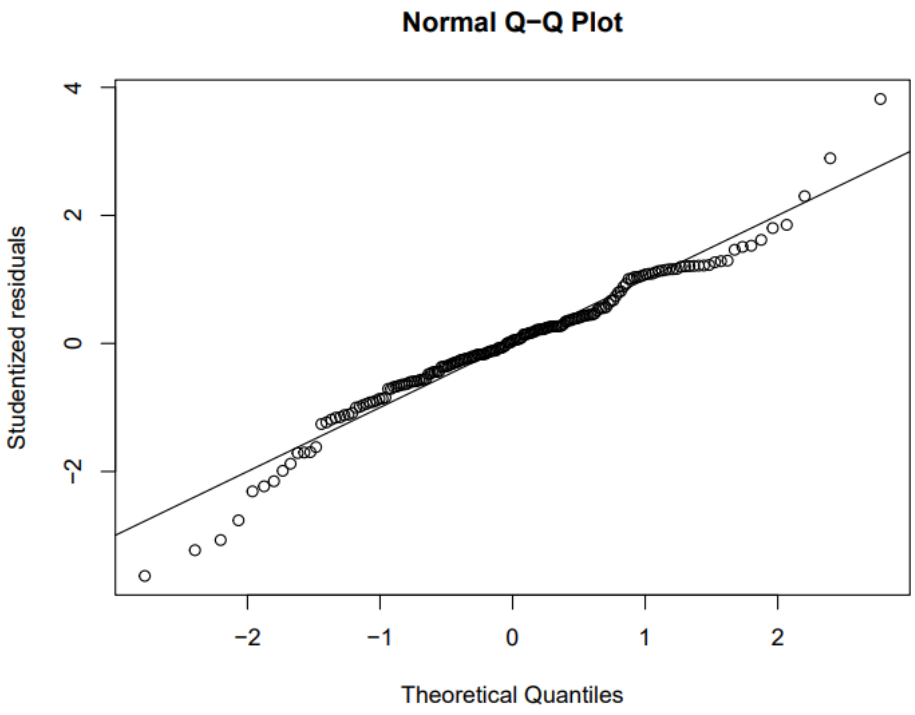
```
par(mfrow=c(1,1))
qqnorm(Model_no_outliers$res,ylab="Residuals")
qqline(Model_no_outliers$res, col="blue")
```




```
qqnorm(rstudent(Model),ylab="Studentized residuals")
abline(0,1)
```




```
qqnorm(rstudent(Model_no_outliers),ylab="Studentized residuals")
abline(0,1)
```



W celu sprawdzenia rozkładu normlanego użyje testu Shapiro-Wilka.

P-value w teście Shapiro-Wilka jest miarą siły dowodu przeciwko hipotezie, że dane pochodzą z rozkładu normalnego. Im mniejsze p-value, tym mocniejszy jest dowód przeciwko hipotezie normalności.

Jeśli p-value jest mniejsze niż poziom istotności (np. 0,05), to odrzucamy hipotezę normalności i stwierdzamy, że dane nie pochodzą z rozkładu normalnego. W przeciwnym razie przyjmujemy hipotezę normalności.

Za pomocą “shapiro.test” mogłem uzyskać następujące wyniki

- Dla modelu “Model” wynik wyszedł W = 0.97386, p-value = 0.001701. Poniżej 0,05 p-value rozkład nie jest normalny.
- Dla modelu “Model2” wynik wyszedł W = 0.97416, p-value = 0.001852. Poniżej 0,05 p-value rozkład nie jest normalny.
- Dla modelu “Model3” wynik wyszedł W = 0.97468, p-value = 0.002148. Poniżej 0,05 p-value rozkład nie jest normalny.
- Dla modelu “Model4” wynik wyszedł W = 0.97386, p-value = 0.001701. Poniżej 0,05 p-value rozkład nie jest normalny.
- Dla modelu “Model5” wynik wyszedł W = 0.97026, p-value = 0.0006324. Poniżej 0,05 p-value rozkład nie jest normalny.
- Dla modelu “Model_no_outlires” wynik wyszedł W = 0.97233, p-value = 0.001155. Poniżej 0,05 p-value rozkład nie jest normalny.

Wszystkie modele uzyskały wynik p-value poniżej 0,05, co czynni je niepochodzących z rozkładu normlanego. Jednocześnie wyniki są bardzo zbliżone. Nie zmieniam modelu, a modelem jest model “Model4”.

```
shapiro.test(Model$res)
shapiro.test(Model_no_outliers$res)
```

```
> shapiro.test(Model$res)

        Shapiro-Wilk normality test

data:  Model$res
W = 0.97386, p-value = 0.001701

> shapiro.test(Model2$res)

        Shapiro-Wilk normality test

data:  Model2$res
W = 0.97416, p-value = 0.001852

> shapiro.test(Model3$res)

        Shapiro-Wilk normality test

data:  Model3$res
W = 0.97468, p-value = 0.002148

> shapiro.test(Model4$res)

        Shapiro-Wilk normality test

data:  Model4$res
W = 0.97386, p-value = 0.001701

> shapiro.test(Model5$res)

        Shapiro-Wilk normality test

data:  Model5$res
W = 0.97026, p-value = 0.0006324
```

```
> shapiro.test(Model_no_outliers$res)

        Shapiro-Wilk normality test

data:  Model_no_outliers$res
W = 0.97233, p-value = 0.001155
```

Test niezależności

Kod ładuje bibliotekę "randtests" i wywołuje funkcję "runs.test" na zmiennej "Model$res" oraz Model_no_outliers$res". Funkcja "runs.test" jest jednym z testów dla oceny losowości w sekwencji wartości. Test jest wykonywany na “residulas” z modelu statystycznego.

Wynik testu przedstawia czy wartości "residuals” są losowe, czy są skorelowane.

Jeśli wartości danych byłyby losowe, można oczekiwać, że liczba ciągów wzrostów i spadków wartości byłaby podobna. Jeśli natomiast wartości danych byłyby skorelowane, można oczekiwać, że będzie więcej ciągów wzrostów niż spadków, lub na odwrót.

W obu przypadkach moje model są powyżej 0.05 p-value, oznacza to, że nie odrzucaja hipotezę zerową i wartości są losowe.

```r
#Testing of independence
library(randtests)

runs.test(Model$res)
runs.test(Model_no_outliers$res)
```

```
> runs.test(Model$res)

        Runs Test

data:  Model$res
statistic = 0.14866, runs = 93, n1 = 91, n2 = 91, n = 182, p-value =
0.8818
alternative hypothesis: nonrandomness

> runs.test(Model_no_outliers$res)

        Runs Test

data:  Model_no_outliers$res
statistic = 0, runs = 91, n1 = 90, n2 = 90, n = 180, p-value = 1
alternative hypothesis: nonrandomness
```

Kod oznacza wykonanie analiz regresji dla zmiennych wyjaśniających w zbiorze danych Dane.

W pierwszej linijce tworzy się macierz explanatory z wykluczeniem pierwszej kolumny Dane.

Następnie tworzy się model regresji dla pierwszej kolumny explanatory jako zmiennej zależnej i pozostałych kolumn explanatory jako zmiennych niezależnych. Podsumowanie modelu jest wyświetlane za pomocą funkcji summary().

Wartość współczynnika determinacji dla tego modelu jest pobierana z jego podsumowania.

```r
#Calculate VIF for the first variable
explanatory<-as.matrix(Dane[,-1])
summary(lm(explanatory[,1]~explanatory[,-1]))
summary(lm(explanatory[,1]~explanatory[,-1]))$r.squared
```

```
> summary(lm(explanatory[,1]~explanatory[,-1]))

Call:
lm(formula = explanatory[, 1] ~ explanatory[, -1])

Residuals:
     Min       1Q   Median       3Q      Max
-232.418  -50.191    0.923   43.754  246.229

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)               2.636e+02  2.733e+01   9.647  < 2e-16 ***
explanatory[, -1]Infant_D -1.762e+00  8.844e-01  -1.992  0.04791 *
explanatory[, -1]H_B       1.131e+00  6.450e-01   1.753  0.08145 .
explanatory[, -1]Measles  -7.701e-04  1.511e-03  -0.510  0.61098
explanatory[, -1]Under_D   1.557e+00  6.518e-01   2.389  0.01799 *
explanatory[, -1]Polio    -1.039e+00  3.547e-01  -2.930  0.00385 **
explanatory[, -1]Dipht    -1.291e+00  7.421e-01  -1.739  0.08384 .
explanatory[, -1]GDP      -1.698e-03  6.102e-04  -2.783  0.00600 **
explanatory[, -1]Popl     -4.823e-07  2.791e-07  -1.728  0.08575 .
explanatory[, -1]Income    3.595e-01  6.796e-01   0.529  0.59749
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 83.72 on 172 degrees of freedom
Multiple R-squared:  0.2791,    Adjusted R-squared:  0.2414
F-statistic:   7.4 on 9 and 172 DF,  p-value: 4.237e-09

> summary(lm(explanatory[,1]~explanatory[,-1]))$r.squared
[1] 0.2791155
```

W tej części kodu zaimportowana jest biblioteka “car” i wykonane jest obliczenie Variance Inflation Factor (VIF) dla modelu “Model” automatycznie. VIF jest miarą multicollinearity (wzajemnej korelacji) pomiędzy zmiennymi niezależnymi w modelu regresji. Wartość VIF powyżej 10 sugeruje występowanie wzajemnej korelacji i konieczność usunięcia jednej z kolumn explanatory. W moim modelu takimi wartościami są “Infant_D” oraz “Under_D”.

```
#Automatic calculation of the VIF values
library(car)
vif(Model)
```

```
> vif(Model)
  Adult_M  Infant_D   Under_D     Polio     Dipht       GDP      Popl
 1.359195 95.660047 99.401120  2.015937  1.889497  1.147135  1.347827
```

Za pomocą “Model”, mogłem dostać współczynniki dla wszystkich zmiennych.

```
#Przedstawienie wspołczynnikow wartosci.
Model$coefficients
```

```
> Model$coefficients
  (Intercept)       Adult_M      Infant_D       Under_D         Polio         Dipht           GDP
 6.935148e+01 -4.730148e-02  9.560769e-02 -8.205823e-02  5.021274e-02  5.425434e-02  1.211242e-04
         Popl
 2.472045e-08
```

Przykładowa interpretacja współczynnika dla “Adult_M”:

Jeśli zwiększymy oczekiwane prawdopodobieństwo śmierci między 15 a 60 rokiem życia na 1000 mieszkańców o rok, to oczekiwana długość życia w kraju zmniejszy się o 4,73 lata, przy założeniu niezmienionych wartości pozostałych zmiennych objaśniających tzn. zgodnie z zasadą ceteris Paribus.

Stworzyłem nowe zmienne do których przypisałem współczynniki. Dostając bety z odpowiadającymi współczynnikami.

```
#Przedstawienie współczynnikow wartosci.
Model$coefficients

#Przypisanie wspolczynnikow do nowych zmiennych
beta_0<-Model$coefficients['(Intercept)']
beta_0
beta_adultm<-Model$coefficients['Adult_M']
beta_adultm
beta_infantd<-Model$coefficients['Infant_D']
beta_infantd
beta_underd<-Model$coefficients['Under_D']
beta_underd
beta_polio<-Model$coefficients['Polio']
beta_polio
beta_dipht<-Model$coefficients['Dipht']
beta_dipht
beta_gdp<-Model$coefficients['GDP']
beta_gdp
beta_popl<-Model$coefficients['Popl']
beta_popl
```

Dzięki stworzeniu bet, mogę dokonać predykcji. Stworzyłem 3 różne predykcje z różnymi liczbami. Przykładowo w “forecast” zakładając, że:

- “Adult Mortality” jest na poziomie 100
- “Infant deaths” jest na poziomie 56
- “Under-five deaths” jest na poziomie 80
- “Polio” jest na poziomie 70
- “Diphtheria” jest na poziomie 80
- “GDP” jest na poziomie 300
- “Population” jest na poziomie 10000000

Wynik oczekiwanej długości życia według modelu wynosi 71.54948.

```
#Forecast

(forecast<-beta_0+beta_adultm*100+beta_infantd*56+beta_underd*80+beta_polio*70+beta_dipht*80+beta_gdp*300+beta_popl*10000000)
(forecast2<-beta_0+beta_adultm*500+beta_infantd*86+beta_underd*90+beta_polio*73+beta_dipht*94+beta_gdp*456+beta_popl*9080000)
(forecast3<-beta_0+beta_adultm*60+beta_infantd*47+beta_underd*52+beta_polio*64+beta_dipht*45+beta_gdp*291+beta_popl*13768000)
```

```
> (forecast<-beta_0+beta_adultm*100+beta_infantd
a_popl*10000000)
(Intercept)
   71.54948
```

W już końcowym etapie projektu dostrzegłem możliwość innego wprowadzenia danych. Użyłem sposobu zmiany danych w RStudio. Musiałem użyć nowego pakietu “tidyverse”, aby odczytało moje dane w formacie “csv”. Tak dostałem dane z prawidłowym odczytem kolumn, które w excelu nie działały, dzięki temu dostałem dostęp do danych, które bardzo mnie ciekawiły i chciałem użyć wcześniej. Postanowiłem nie usuwać poprzedniej pracy w celu sprawozdania pracy i dostania możliwych ciekawych wyników.

Instalacja pakietu i uruchomienie go, aby użyć funkcji “read_csv”

```
#Package do czytania "read_csv"
library(tidyverse)
```

Stworzenie nowego wektora z nową bazą danych, o nazwie “Dane_nowe”

```
#Przydzielenie bazy danych do wektora 'Dane_nowe'
Dane_nowe <-  read_csv("C:/Users/Kacper/Desktop/Projekt Regression/Dane z 2015.csv")
View(Dane_nowe)
names(Dane_nowe)
```

Działania kodu:

Ładuje bibliotekę "dplyr", która zapewnia zestaw narzędzi do pracy z ramkami danych w R.

Używa biblioteki "dplyr" do modyfikowania ramki danych "Dane_nowe". Funkcja "select" jest używana do wybierania kolumn z ramki danych, a argument "-c(Country, Year, Status, Alcohol, **`percentage expenditure`**, **`Total expenditure`**)" jest używany, aby wykluczyć określone kolumny z ramki danych. Rezultat jest przypisywany z powrotem do "Dane_nowe" za pomocą operatora rury "%>%".

Wywołuje funkcję "View", aby wyświetlić zawartość zmodyfikowanej ramki danych "Dane_nowe" w formacie tabeli.

Wywołuje funkcję "names", aby wyświetlić nazwy kolumn w zmodyfikowanej ramce danych "Dane_nowe".

```
library("dplyr")
Dane_nowe <- Dane_nowe %>% select(-c(Country, Year, Status, Alcohol,
View(Dane_nowe)
names(Dane_nowe)
```

Przydzielanie nowych nazw kolumną.

```
#Nowe nazwy
colnames(Dane_nowe)<-c("Life","Adult_M",
```

Działania kodu:

Tworzy nową kolumnę w ramce danych "Dane_nowe" o nazwie "H_B".

Używa funkcji "ifelse" do wypełnienia brakujących wartości (reprezentowanych przez "NA") w kolumnie "H_B".

Jeśli wartość w danej komórce kolumny "H_B" jest "NA", oblicza się średnią wszystkich nie-brakujących wartości w kolumnie "H_B" za pomocą funkcji "mean" z argumentem "na.rm = TRUE", który usuwa brakujące wartości z obliczeń.

Jeśli wartość w danej komórce kolumny "H_B" nie jest "NA", wartość pozostaje niezmieniona.

Rezultat jest przypisywany z powrotem do kolumny "Dane_nowe$H_B".

```
#Zastepowanie ubytków
Dane_nowe$H_B = ifelse(is.na(Dane_nowe$H_B),
                       ave(Dane_nowe$H_B, FUN = function(x) mean(x, na.rm = TRUE)),
                       Dane_nowe$H_B)
```

Działania te są wykonywane na reszcie kolumn z brakującymi danymi.

```
Dane_nowe$BMI = ifelse(is.na(Dane_nowe$BMI ),
                       ave(Dane_nowe$BMI , FUN = function(x) mean(x, na.rm = TRUE)),
                       Dane_nowe$BMI )
Dane_nowe$GDP = ifelse(is.na(Dane_nowe$GDP ),
                       ave(Dane_nowe$GDP  , FUN = function(x) mean(x, na.rm = TRUE)),
                       Dane_nowe$GDP  )
Dane_nowe$Popl = ifelse(is.na(Dane_nowe$Popl  ),
                       ave(Dane_nowe$Popl   , FUN = function(x) mean(x, na.rm = TRUE)),
                       Dane_nowe$Popl   )
Dane_nowe$`t_1-19`  = ifelse(is.na(Dane_nowe$`t_1-19`   ),
                        ave(Dane_nowe$`t_1-19`   , FUN = function(x) mean(x, na.rm = TRUE)),
                        Dane_nowe$`t_1-19`   )
Dane_nowe$`t_5-9` = ifelse(is.na(Dane_nowe$`t_5-9`   ),
                        ave(Dane_nowe$`t_5-9`    , FUN = function(x) mean(x, na.rm = TRUE)),
                        Dane_nowe$`t_5-9`    )
Dane_nowe$Income = ifelse(is.na(Dane_nowe$Income    ),
                          ave(Dane_nowe$Income    , FUN = function(x) mean(x, na.rm = TRUE)),
                          Dane_nowe$Income    )
Dane_nowe$Schooling = ifelse(is.na(Dane_nowe$Schooling   ),
                         ave(Dane_nowe$Schooling    , FUN = function(x) mean(x, na.rm = TRUE)),
                         Dane_nowe$Schooling    )
```

Sprawdzanie współliniowości.

Działanie kodu:

Ten kod tworzy nową macierz "matrix_nowy" z danymi "Dane_nowe" bez pierwszej kolumny. Następnie oblicza macierz wartości własnych "matrix_eigen_nowy" jako iloczyn transponowanej macierzy "matrix_nowy" i "matrix_nowy". W końcowym kroku wartości pierwszej wartości własnej dzielone są przez wszystkie wartości własne, a następnie wynik jest pierwiastkowany.

```
#współliniowość
matrix_nowy <- as.matrix(Dane_nowe[,-1])

matrix_eigen_nowy <- eigen(t(matrix_nowy) %*% matrix_nowy)
matrix_eigen_nowy$val

sqrt(matrix_eigen_nowy$val[1]/matrix_eigen_nowy$val)
```

Wyniki powyżej 30 pokazują brak współliniowości.

```
> sqrt(matrix_eigen_nowy$val[1]/matrix_eigen_nowy$val)
 [1] 1.000000e+00 2.231714e+03 3.544788e+03 1.476668e+05 3.349387e+05 4.270232e+05 1.426854e+06
 [8] 1.775193e+06 4.208999e+06 4.858363e+06 6.555410e+06 1.114377e+07 2.786175e+07 4.429972e+07
[15] 3.791074e+08
```

Kod tworzy nowy model regresji liniowej "Model_nowy" z zmienną objaśnianą "Life" i kilkoma zmiennymi objaśniającymi: "Adult_M", "Infant_D", "H_B", "Measles", "BMI", "Under_D", "Polio", "Dipht", "H/A", "GDP", "Popl", "t_1-19", "t_5-9", "Income", "Schooling". Dane są wczytywane z data frame'u "Dane_nowe". W końcowym kroku jest wyświetlany podsumowanie modelu regresji liniowej "Model_nowy".

```
#Model
Model_nowy <- lm(Life~Adult_M+
summary(Model_nowy)
```

```
> summary(Model_nowy)

Call:
lm(formula = Life ~ Adult_M + Infant_D + H_B + Measles + BMI + 
    Under_D + Polio + Dipht + `H/A` + GDP + Popl + `t_1-19` + 
    `t_5-9` + Income + Schooling, data = Dane_nowe)

Residuals:
    Min      1Q  Median      3Q     Max 
-9.0242 -1.4732  0.0279  1.5076  7.6316 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.325e+01  2.112e+00  25.214  < 2e-16 ***
Adult_M     -2.512e-02  3.473e-03  -7.233 1.65e-11 ***
Infant_D     9.068e-02  3.411e-02   2.659  0.00862 ** 
H_B          2.476e-02  2.359e-02   1.050  0.29526    
Measles     -4.671e-05  5.591e-05  -0.835  0.40471    
BMI         -2.990e-03  1.395e-02  -0.214  0.83062    
Under_D     -6.832e-02  2.482e-02  -2.753  0.00656 ** 
Polio        1.469e-02  1.332e-02   1.103  0.27170    
Dipht        1.143e-02  2.743e-02   0.417  0.67733    
`H/A`       -4.438e-01  2.287e-01  -1.940  0.05402 .  
GDP          5.588e-06  2.414e-05   0.231  0.81721    
Popl         4.571e-09  1.029e-08   0.444  0.65730    
`t_1-19`    -1.883e-01  2.496e-01  -0.754  0.45171    
`t_5-9`     -2.071e-02  2.462e-01  -0.084  0.93307    
Income       2.713e+01  4.619e+00   5.872 2.28e-08 ***
Schooling    4.810e-02  2.037e-01   0.236  0.81364    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.97 on 166 degrees of freedom
Multiple R-squared:  0.8746,    Adjusted R-squared:  0.8633 
F-statistic: 77.19 on 15 and 166 DF,  p-value: < 2.2e-16
```

Użycie metody “backward”, aby ustalić najlepszy model.

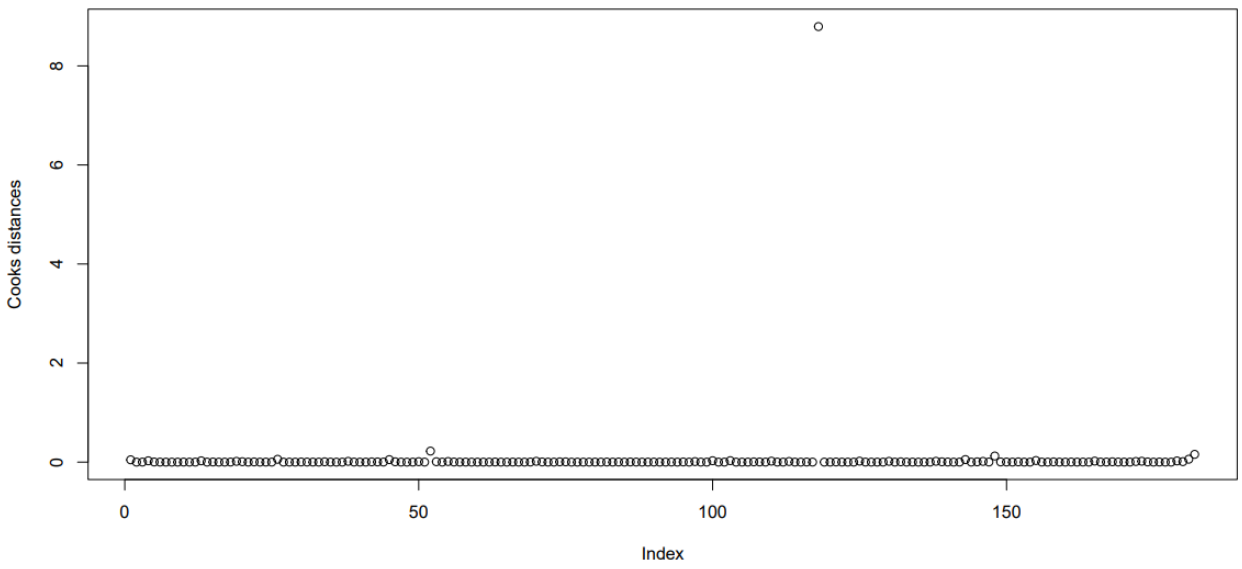
```
Model_nowy_backward<-step(lm(Life~Adult_M+
summary(Model_nowy_backward)
Model_nowy <- Model_nowy_backward
summary(Model_nowy)
```

Finalny wynik jest bardzo optymistyczny, ponieważ wynik R-squared wynosi 0.8726 (zmienność oczekiwanej długości życia jest wyjaśniony w 87% przez model), a “Adjusted R-squared" wynosi 0.8675.

```
> Model_nowy <- Model_nowy_backward
> summary(Model_nowy)

Call:
lm(formula = Life ~ Adult_M + Infant_D + H_B + Under_D + `H/A` +
    `t_1-19` + Income, data = Dane_nowe)

Residuals:
    Min      1Q  Median      3Q     Max
-9.7571 -1.4736 -0.1144  1.6313  8.0334

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 53.331379   1.970747  27.062  < 2e-16 ***
Adult_M     -0.025318   0.003364  -7.525 2.71e-12 ***
Infant_D     0.069914   0.025657   2.725 0.007088 **
H_B          0.040848   0.010313   3.961 0.000109 ***
Under_D     -0.054905   0.019797  -2.773 0.006154 **
`H/A`       -0.458897   0.219038  -2.095 0.037614 *
`t_1-19`    -0.192444   0.072761  -2.645 0.008919 **
Income      29.053282   2.182283  13.313  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.923 on 174 degrees of freedom
Multiple R-squared:  0.8726,    Adjusted R-squared:  0.8675
F-statistic: 170.3 on 7 and 174 DF,  p-value: < 2.2e-16
```

Sprawdzenie outlierów oraz stworzenie nowego modelu z usunietymi wartościami odstającymi.

```
cooks_dis <- cooks.distance(Model_nowy)
plot(cooks_dis,ylab="Cooks distances")
Model_nowy_no_outliers <-lm(Life~Adult_M+
summary(Model_nowy_no_outliers)
summary(Model_nowy)
```



Wynik R-squared wynosi poprawił się z 0.8726 na 0.8773 (zmienność oczekiwanej długości życia jest wyjaśniony w 88% przez model), a “Adjusted R-squared" z 0.8675 na 0.8723.

```
> summary(Model_nowy_no_outliers)

Call:
lm(formula = Life ~ Adult_M + Infant_D + H_B + Under_D + `H/A` +
    `t_1-19` + Income, data = Dane_nowe, subset = (cooks_dis <
    max(cooks_dis)))

Residuals:
    Min      1Q  Median      3Q     Max
-8.7930 -1.5945 -0.0056  1.5528  8.2047

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 55.166752   1.991738  27.698  < 2e-16 ***
Adult_M     -0.024734   0.003274  -7.554 2.33e-12 ***
Infant_D     0.243635   0.057474   4.239 3.65e-05 ***
H_B          0.038465   0.010047   3.829  0.00018 ***
Under_D     -0.198851   0.047024  -4.229 3.80e-05 ***
`H/A`       -0.485941   0.213008  -2.281  0.02375 *
`t_1-19`    -0.185696   0.070736  -2.625  0.00943 **
Income      26.920332   2.213949  12.159  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.841 on 173 degrees of freedom
Multiple R-squared:  0.8773,    Adjusted R-squared:  0.8723
F-statistic: 176.6 on 7 and 173 DF,  p-value: < 2.2e-16
```

Sprawdzenie czy rozkład jest normalny za pomocą Testu Shapiro. W przypadku “Model_nowy” model nie ma rozkładu normalnego. Natomiast w modelu “Model_nowy_no_outliers” p-value wynosi ponad 0.05, oznacza to, że istnieje rozkład normalny.

```
#Test normalności
shapiro.test(Model_nowy$res)
shapiro.test(Model_nowy_no_outliers$res)
```

```
> #Test normalności
> shapiro.test(Model_nowy$res)

        Shapiro-Wilk normality test

data:  Model_nowy$res
W = 0.98429, p-value = 0.03897

> shapiro.test(Model_nowy_no_outliers$res)

        Shapiro-Wilk normality test

data:  Model_nowy_no_outliers$res
W = 0.98966, p-value = 0.2134
```

Na końcu porównałem moje wszystkie modele.

```
#Porównanie
summary(Model)
summary(Model_nowy)
```

“Model_nowy” ma lepszy “R-squared” niż “Model” - 0.72 < 0.87.

“Model_nowy” ma lepszy “Adjusted R-squared” niż “Model” - 0.71 < 0.86.

“Model_nowy” ma lepsze zmienne o wartości “p-value” mniejszych niż 0.05, niż “Model”:

- “Model” trzy wartości z przedziału 0 – 0,001
- “Model_nowy” cztery wartości z przedziału 0 – 0,001
- “Model” trzy wartości z przedziału 0,001 – 0,01
- “Model_nowy” trzy wartości z przedziału 0,001 – 0,01
- “Model” jedną wartość z przedziału 0,01 – 0,05
- “Model_nowy” jedną wartość z przedziału 0,01 – 0,05
- “Model” jedną wartość z przedziału 0,05 – 0,1
- “Model_nowy” zero wartości z przedziału 0,05 – 0,1

```
> summary(Model)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP + Popl, data = Dane)

Residuals:
     Min       1Q   Median       3Q      Max
-16.4750  -2.5123   0.3676   2.3051  11.8873

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.935e+01  1.742e+00  39.819  < 2e-16 ***
Adult_M     -4.730e-02  3.887e-03 -12.169  < 2e-16 ***
Infant_D     9.561e-02  3.737e-02   2.558 0.011365 *
Under_D     -8.206e-02  2.955e-02  -2.777 0.006095 **
Polio        5.021e-02  1.872e-02   2.683 0.008003 **
Dipht        5.425e-02  1.982e-02   2.737 0.006843 **
GDP          1.211e-04  3.192e-05   3.795 0.000203 ***
Popl         2.472e-08  1.439e-08   1.718 0.087513 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.312 on 174 degrees of freedom
Multiple R-squared:  0.7229,    Adjusted R-squared:  0.7118
F-statistic: 64.85 on 7 and 174 DF,  p-value: < 2.2e-16
```

```
> summary(Model_nowy)

Call:
lm(formula = Life ~ Adult_M + Infant_D + H_B + Under_D + `H/A` +
    `t_1-19` + Income, data = Dane_nowe)

Residuals:
    Min      1Q  Median      3Q     Max
-9.7571 -1.4736 -0.1144  1.6313  8.0334

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 53.331379   1.970747  27.062  < 2e-16 ***
Adult_M     -0.025318   0.003364  -7.525 2.71e-12 ***
Infant_D     0.069914   0.025657   2.725 0.007088 **
H_B          0.040848   0.010313   3.961 0.000109 ***
Under_D     -0.054905   0.019797  -2.773 0.006154 **
`H/A`       -0.458897   0.219038  -2.095 0.037614 *
`t_1-19`    -0.192444   0.072761  -2.645 0.008919 **
Income      29.053282   2.182283  13.313  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.923 on 174 degrees of freedom
Multiple R-squared:  0.8726,    Adjusted R-squared:  0.8675
F-statistic: 170.3 on 7 and 174 DF,  p-value: < 2.2e-16
```

```
summary(Model_no_outliers)
summary(Model_nowy_no_outliers)
```

“Model_nowy_no_outliers” ma lepszy “R-squared” niż “Model_no_outliers” - 0.73 < 0.87.

“Model_nowy_no_outliers” ma lepszy “Adjusted R-squared” niż “Model_no_outliers” - 0.72 < 0.87.

“Model_nowy_no_outliers” ma lepsze zmienne o wartości “p-value” mniejszych niż 0.05, niż “Model_no_outliers”:

- “Model_no_outliers” pięć wartości z przedziału 0 – 0,001
- “Model_nowy_no_outliers” sześć wartości z przedziału 0 – 0,001
- “Model_no_outliers” jedną wartość z przedziału 0,001 – 0,01
- “Model_nowy_no_outliers” jedną wartość z przedziału 0,001 – 0,01
- “Model_no_outliers” jedną wartość z przedziału 0,01 – 0,05
- “Model_nowy_no_outliers” jedną wartość z przedziału 0,01 – 0,05
- “Model_no_outliers” zero wartości z przedziału 0,05 – 0,1
- “Model_nowy_no_outliers” zero wartości z przedziału 0,05 – 0,1
- “Model_no_outliers” ma jedną wartość z przedziału 0,1 – 1

```
> summary(Model_no_outliers)

Call:
lm(formula = Life ~ Adult_M + Infant_D + Under_D + Polio + Dipht +
    GDP + Popl, data = Dane, subset = (cooks_dis < max(cooks_dis)))

Residuals:
     Min       1Q   Median       3Q      Max
-14.2143  -2.3349   0.1912   2.2708  11.9048

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.042e+01  1.714e+00  41.099  < 2e-16 ***
Adult_M     -4.508e-02  3.816e-03 -11.813  < 2e-16 ***
Infant_D     2.787e-01  6.307e-02   4.418 1.75e-05 ***
Under_D     -2.341e-01  5.158e-02  -4.539 1.05e-05 ***
Polio        4.282e-02  1.824e-02   2.347 0.020060 *
Dipht        5.017e-02  1.923e-02   2.609 0.009875 **
GDP          1.194e-04  3.091e-05   3.863 0.000158 ***
Popl         1.072e-08  1.448e-08   0.740 0.460100
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.175 on 173 degrees of freedom
Multiple R-squared:  0.7349,    Adjusted R-squared:  0.7242
F-statistic: 68.51 on 7 and 173 DF,  p-value: < 2.2e-16

> summary(Model_nowy_no_outliers)

Call:
lm(formula = Life ~ Adult_M + Infant_D + H_B + Under_D + `H/A` +
    `t_1-19` + Income, data = Dane_nowe, subset = (cooks_dis <
    max(cooks_dis)))

Residuals:
    Min      1Q  Median      3Q     Max
-8.7930 -1.5945 -0.0056  1.5528  8.2047

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 55.166752   1.991738  27.698  < 2e-16 ***
Adult_M     -0.024734   0.003274  -7.554 2.33e-12 ***
Infant_D     0.243635   0.057474   4.239 3.65e-05 ***
H_B          0.038465   0.010047   3.829  0.00018 ***
Under_D     -0.198851   0.047024  -4.229 3.80e-05 ***
`H/A`       -0.485941   0.213008  -2.281  0.02375 *
`t_1-19`    -0.185696   0.070736  -2.625  0.00943 **
Income      26.920332   2.213949  12.159  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.841 on 173 degrees of freedom
Multiple R-squared:  0.8773,    Adjusted R-squared:  0.8723
F-statistic: 176.6 on 7 and 173 DF,  p-value: < 2.2e-16
```

W tym etapie z ciekawości użyłem metody “Machine Learning”, aby zobaczyć jakie wyniki mogę uzyskać działając na moich danych. Dokonałem podziału danych na część “test”, którą będę testować na podstawie części “training”. Podział jest 80% dla “training” i 20% dla “test” z całości danych. Zainstalowałem pakiet “caTools”, oraz go uruchomiłem za pomocą funkcji “library”. Następnie uruchomienie funkcji “set.seed”, która odpowiada za generowanie losowych liczb. Funkcja “sample.split”, która podzieliła moją bazą danych według ustalonego wcześniejszego podziału.

Kod przedstawia:

Pierwsza linia - ładowanie biblioteki caTools

Druga linia - ustawianie seeda losowego na wartość 123

Trzecia linia - dzielenie danych na zbiór treningowy i testowy z proporcją 80/20

Czwarta linia - tworzenie zbioru treningowego na podstawie danych i dzielenia

Piąta linia - tworzenie zbioru testowego na podstawie danych i dzielenia

```
#Podzial danych na training i test
library(caTools)
set.seed(123)
split = sample.split(Dane$Life, SplitRatio = 0.8)
training_set = subset(Dane, split == TRUE)
test_set = subset(Dane, split == FALSE)
```

Stworzenie modelu składającego z bazy danych “training_set”.

Kod przedstawia:

Pierwsza linia - tworzenie modelu regresji liniowej na zbiorze treningowym, gdzie zmienna objaśniana jest Life a pozostałe zmienne są uwzględniane jako zmienne objaśniające

Druga linia - wyświetlenie podsumowania modelu

```
regressor = lm(formula = Life ~ .,
               data = training_set)

summary(regressor)
```

```
> summary(regressor)

Call:
lm(formula = Life ~ ., data = training_set)

Residuals:
     Min       1Q   Median       3Q      Max
-12.8832  -2.3786   0.2148   2.2319  11.7683

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.095e+01  2.035e+00  34.871  < 2e-16 ***
Adult_M     -4.333e-02  4.427e-03  -9.788  < 2e-16 ***
Infant_D     3.582e-01  8.639e-02   4.146 5.96e-05 ***
H_B         -2.775e-03  4.473e-02  -0.062  0.95064
Measles      4.439e-05  9.019e-05   0.492  0.62341
Under_D     -3.029e-01  6.925e-02  -4.373 2.44e-05 ***
Polio        2.729e-02  2.103e-02   1.297  0.19670
Dipht        6.012e-02  4.980e-02   1.207  0.22947
GDP          1.214e-04  3.811e-05   3.186  0.00179 **
Popl         3.350e-09  1.539e-08   0.218  0.82795
Income       8.854e-02  4.126e-02   2.146  0.03367 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.311 on 134 degrees of freedom
Multiple R-squared:  0.7194,    Adjusted R-squared:  0.6985
F-statistic: 34.36 on 10 and 134 DF,  p-value: < 2.2e-16
```

Stworzyłem przewidziane y, dla “regressor”

Kod przedstawia:

Pierwsza linia - prognozowanie wartości Life na zbiorze testowym

Druga linia - wyświetlenie prognozowanych wartości.

```
y_pred = predict(regressor, newdata = test_set)
y_pred
```

```
> y_pred
       1        2        3        4        5        6        7        8        9       10
76.670226 54.765098 80.158901 83.354213 73.581444 76.470558 72.056293 74.566890 76.538567 81.451813
      11       12       13       14       15       16       17       18       19       20
56.818147 77.422626 73.219638 68.548345 64.410685 73.990682 60.204301 66.129969 69.850911 79.137619
      21       22       23       24       25       26       27       28       29       30
73.871221 70.555095 74.053635 70.754243 71.222304 68.897333 82.096634  8.478208 63.420292 74.277451
      31       32       33       34       35       36       37
71.569561 75.935574 79.217138 80.146481 62.553392 74.047578 68.050284
```

Stworzenie “training_set_nowy” i “test_set_nowy” na podstawie “Dane_nowe”, według tych samych zasad.

```
split_nowy = sample.split(Dane_nowe$Life, SplitRatio = 0.8)
training_set_nowy = subset(Dane_nowe, split == TRUE)
test_set_nowy = subset(Dane_nowe, split == FALSE)
```

Stworzenie modelu regresji liniowej “regressor_nowy” na zbiorze treningowym “training_set_nowy”, gdzie zmienna objaśniana jest Life a pozostałe zmienne są uwzględniane jako zmienne objaśniające

```
regressor_nowy = lm(formula = Life ~ .,
                    data = training_set_nowy)
```

Porównałem dwa modele.

“regressor_nowy” ma lepszy “R-squared” niż “regressor” - 0.71 < 0.87.

“regressor_nowy” ma lepszy “Adjusted R-squared” niż “regressor” - 0.69 < 0.86.

```
summary(regressor_nowy)
summary(regressor)
```

```
> summary(regressor_nowy)

Call:
lm(formula = Life ~ ., data = training_set_nowy)

Residuals:
    Min      1Q  Median      3Q     Max
-9.1604 -1.3642 -0.1081  1.5677  7.6245

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.454e+01  2.398e+00  22.748  < 2e-16 ***
Adult_M     -2.577e-02  3.856e-03  -6.684 6.34e-10 ***
Infant_D     1.996e-01  6.846e-02   2.916  0.00418 **
H_B          3.779e-02  3.070e-02   1.231  0.22056
Measles     -3.437e-05  6.371e-05  -0.539  0.59053
BMI         -1.023e-02  1.462e-02  -0.699  0.48554
Under_D     -1.599e-01  5.467e-02  -2.924  0.00408 **
Polio        1.100e-04  1.446e-02   0.008  0.99394
Dipht        1.486e-03  3.413e-02   0.044  0.96534
`H/A`       -3.577e-01  2.432e-01  -1.471  0.14378
GDP          6.981e-06  2.874e-05   0.243  0.80849
Popl        -7.987e-09  1.065e-08  -0.750  0.45476
`t_1-19`     7.336e-03  3.198e-01   0.023  0.98173
`t_5-9`     -1.954e-01  3.153e-01  -0.620  0.53647
Income       2.683e+01  5.226e+00   5.133 1.02e-06 ***
Schooling    8.662e-02  2.353e-01   0.368  0.71343
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.911 on 129 degrees of freedom
Multiple R-squared:  0.8768,    Adjusted R-squared:  0.8625
F-statistic: 61.22 on 15 and 129 DF,  p-value: < 2.2e-16
```

```
> summary(regressor)

Call:
lm(formula = Life ~ ., data = training_set)

Residuals:
     Min       1Q   Median       3Q      Max
-12.8832  -2.3786   0.2148   2.2319  11.7683

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.095e+01  2.035e+00  34.871  < 2e-16 ***
Adult_M     -4.333e-02  4.427e-03  -9.788  < 2e-16 ***
Infant_D     3.582e-01  8.639e-02   4.146 5.96e-05 ***
H_B         -2.775e-03  4.473e-02  -0.062  0.95064
Measles      4.439e-05  9.019e-05   0.492  0.62341
Under_D     -3.029e-01  6.925e-02  -4.373 2.44e-05 ***
Polio        2.729e-02  2.103e-02   1.297  0.19670
Dipht        6.012e-02  4.980e-02   1.207  0.22947
GDP          1.214e-04  3.811e-05   3.186  0.00179 **
Popl         3.350e-09  1.539e-08   0.218  0.82795
Income       8.854e-02  4.126e-02   2.146  0.03367 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.311 on 134 degrees of freedom
Multiple R-squared:  0.7194,    Adjusted R-squared:  0.6985
F-statistic: 34.36 on 10 and 134 DF,  p-value: < 2.2e-16
```

Prognozowanie wartości Life na zbiorze testowym “test_set_nowy” i modelu “regressor_nowy”.

```
y_pred_nowy = predict(regressor_nowy, newdata = test_set_nowy)
y_pred_nowy
```

```
> y_pred_nowy = predict(regressor_nowy, newdata = test_set_nowy)
> y_pred_nowy
       1        2        3        4        5        6        7        8        9       10       11       12       13
77.35855 58.70143 79.16632 82.99941 75.57473 80.84683 72.02276 75.67873 79.42603 80.71335 52.37017 79.52112 72.87348
      14       15       16       17       18       19       20       21       22       23       24       25       26
61.31732 66.19483 74.75147 58.20621 60.07876 69.03170 77.48792 71.30342 67.42520 75.22256 70.60603 72.00059 65.47484
      27       28       29       30       31       32       33       34       35       36       37
82.47891 34.28289 66.12623 73.64672 72.81358 78.07606 80.81572 81.14668 66.82476 71.37072 66.73114
```

Porównanie “y_pred” i “y_pred_nowy” z “Life”.

1. “y_pred_nowy” jest bardziej przybliżony, różnica 0,45
2. “y_pred” jest bardziej przybliżony, różnica 2,37
3. “y_pred_nowy” jest bardziej przybliżony, różnica 2,77
4. “y_pred_nowy” jest bardziej przybliżony, różnica 0,19
5. “y_pred_nowy” jest bardziej przybliżony, różnica 0,53
6. “y_pred_nowy” jest bardziej przybliżony, różnica 0,25
7. “y_pred” jest bardziej przybliżony, różnica 4,95
8. “y_pred_nowy” jest bardziej przybliżony, różnica 1,72
9. “y_pred” jest bardziej przybliżony, różnica 1,16
10. “y_pred” jest bardziej przybliżony, różnica 0,75
11. “y_pred_nowy” jest bardziej przybliżony, różnica 0,13
12. “y_pred_nowy” jest bardziej przybliżony, różnica 5,48
13. “y_pred” jest bardziej przybliżony, różnica 5,78
14. “y_pred_nowy” jest bardziej przybliżony, różnica 3,38
15. “y_pred_nowy” jest bardziej przybliżony, różnica 0,19
16. “y_pred” jest bardziej przybliżony, różnica 0,39
17. “y_pred_nowy” jest bardziej przybliżony, różnica 0,8
18. “y_pred_nowy” jest bardziej przybliżony, różnica 1,18
19. “y_pred_nowy” jest bardziej przybliżony, różnica 2,83
20. “y_pred_nowy” jest bardziej przybliżony, różnica 2,79
21. “y_pred_nowy” jest bardziej przybliżony, różnica 0,20
22. “y_pred_nowy” jest bardziej przybliżony, różnica 1,73
23. “y_pred” jest bardziej przybliżony, różnica 0,55
24. “y_pred_nowy” jest bardziej przybliżony, różnica 1,2
25. “y_pred” jest bardziej przybliżony, różnica 2,42
26. “y_pred_nowy” jest bardziej przybliżony, różnica 1,13
27. “y_pred” jest bardziej przybliżony, różnica 0,20
28. “y_pred_nowy” jest bardziej przybliżony, różnica 20,22, anomalia
29. “y_pred_nowy” jest bardziej przybliżony, różnica 2,37
30. “y_pred” jest bardziej przybliżony, różnica 0,72
31. “y_pred_nowy” jest bardziej przybliżony, różnica 1,19
32. “y_pred” jest bardziej przybliżony, różnica 1,44
33. “y_pred” jest bardziej przybliżony, różnica 2,52
34. “y_pred_nowy” jest bardziej przybliżony, różnica 1,65
35. “y_pred” jest bardziej przybliżony, różnica 0,75
36. “y_pred” jest bardziej przybliżony, różnica 1,95
37. “y_pred_nowy” jest bardziej przybliżony, różnica 4,93

“y_pred_nowy” miał lepsze przybliżenie wyników w 23/37. W punkcie 28 doszło do anomalii, a różnica wyniku przypadku “y_pred_nowy” okazało się oddalone od wartości ze zbioru testującej w 20,22, a “y_pred” w 46,02.

```
y_pred = predict(regressor, newdata = test_set)
y_pred
y_pred_nowy = predict(regressor_nowy, newdata = test_set_nowy)
y_pred_nowy
```

```
> y_pred = predict(regressor, newdata = test_set)
> y_pred
       1        2        3        4        5        6        7        8        9
76.670226 54.765098 80.158901 83.354213 73.581444 76.470558 72.056293 74.566890 76.538567
      10       11       12       13       14       15       16       17       18
81.451813 56.818147 77.422626 73.219638 68.548345 64.410685 73.990682 60.204301 66.129969
      19       20       21       22       23       24       25       26       27
69.850911 79.137619 73.871221 70.555095 74.053635 70.754243 71.222304 68.897333 82.096634
      28       29       30       31       32       33       34       35       36
 8.478208 63.420292 74.277451 71.569561 75.935574 79.217138 80.146481 62.553392 74.047578
      37
68.050284
> y_pred_nowy = predict(regressor_nowy, newdata = test_set_nowy)
> y_pred_nowy
       1        2        3        4        5        6        7        8        9       10
77.35855 58.70143 79.16632 82.99941 75.57473 80.84683 72.02276 75.67873 79.42603 80.71335
      11       12       13       14       15       16       17       18       19       20
52.37017 79.52112 72.87348 61.31732 66.19483 74.75147 58.20621 60.07876 69.03170 77.48792
      21       22       23       24       25       26       27       28       29       30
71.30342 67.42520 75.22256 70.60603 72.00059 65.47484 82.47891 34.28289 66.12623 73.64672
      31       32       33       34       35       36       37
72.81358 78.07606 80.81572 81.14668 66.82476 71.37072 66.73114
```

	Life
1	77.8
2	52.4
3	76.4
4	82.8
5	76.1
6	81.1
7	77.0
8	77.4
9	77.7
10	82.2
11	52.5
12	85.0
13	79.0
14	64.7
15	66.0
16	73.6
17	59.0
18	58.9
19	66.2
20	74.7
21	71.1
22	65.7
23	74.6
24	69.4
25	68.8
26	66.6
27	81.9
28	54.5
29	68.5
30	75.0
31	74.0
32	74.5
33	76.7
34	82.8
35	61.8
36	76.0
37	61.8