Regresja - diagnostyka

Zbiór danych

W niniejszych ćwiczeniach nadal pracujemy z danymi o krabach.

```
crabs <- read.csv("C:/Users/Ewci/Desktop/wnioskowanie II/Dane/crabs.csv")
```

Rozważymy również nowy zbiór danych, który przedstawia stopień rozpadu pewnego organicznego materiału.

```
decay <- read.csv("C:/Users/Ewci/Desktop/wnioskowanie II/Dane/decay.csv")
```

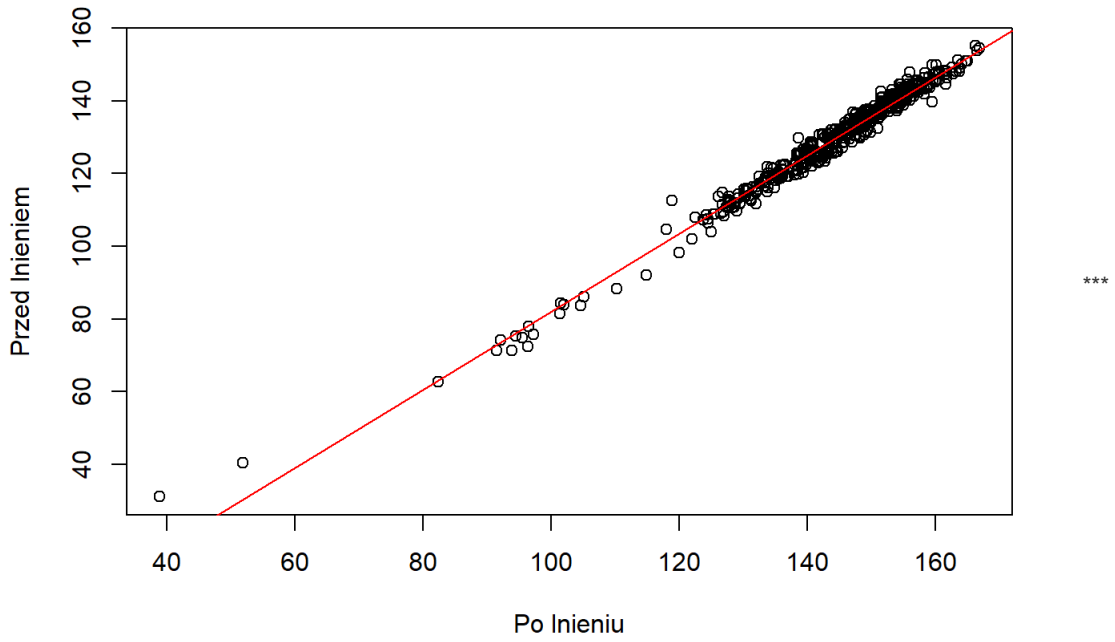
Ćwiczenia

Dopasowujemy model liniowy do danych

```
x<-crabs$postsz
y<-crabs$presz
model.crabs<-lm(y~x)
```

Zadanie: jak myślisz, które punkty danych mają największy wpływ na współczynniki regresji (posłuż się wykresem punktowym, dodaj do niego linię regresji)

```
plot(x,y, xlab="Po lnieniu",ylab="Przed lnieniem")
abline(lm(y~x), col="red")
```



***

Na wykresie regresji możemy zaobserwować trzy punkty, które mają istotny przebieg na linię regresji. Jednak trzy z nich istotnie wpływają na współczynnik kierunkowy. Jest tak dlatego, że punkty te mają dużą dźwignię (ang. leverage).

Jako dźwignię punktu intuicyjnie rozumiemy odległość punktu, mierzoną na osi x, od średniej zmiennej x. Typową miarą dźwigni dla obserwacji o indeksie i, jest wartość h_{ii} leżąca na diagonalnej macierzy rzutu H na obraz macierzy obserwacji (ang. hat matrix).

Zadanie: wyraź h_{ii} jako pochodną pewnej wielkości

$\hat{y_{i}}=\sum\limits_{j=1}^{n}{h_{ij}y_j}$ $h_{ii}=\frac{\delta{\hat{y_i}}}{\delta{y_i}}$ *** Wiedząc, że $H = X(X^T X)^{-1} X^T$, możemy udowodnić, że $$0 < h_{ii} = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_j (x_j - \bar{x})^2} < 1$$

Zadanie: Oblicz dźwignię z powyższego wzoru i porównaj wyniki ze zwracanymi przez funkcję `lm.influence`

```
x <- crabs$postsz
n <- length(x)
m <- mean(x)
hii<-(1/n) + ((x-m)^2) / (sum((x-m)^2))
```

Obliczona dźwignia ze wzoru

```
head(hii)
```

```
## [1] 0.004717405 0.003252191 0.002850832 0.002122182 0.002328025 0.002254075
```

```
tail(hii)
```

```
## [1] 0.005257815 0.002524655 0.002646827 0.002120210 0.002310611 0.004258370
```

Obliczona dźwignia przez funkcje

```
lm.influence
```

```
## function (model, do.coef = TRUE)
## {
##     wt.res <- weighted.residuals(model)
##     e <- na.omit(wt.res)
##     if (model$rank == 0) {
##         n <- length(wt.res)
##         sigma <- sqrt(deviance(model)/df.residual(model))
##         res <- list(hat = rep(0, n), coefficients = matrix(0,
##             n, 0), sigma = rep(sigma, n))
##     }
##     else {
##         e[abs(e) < 100 * .Machine$double.eps * median(abs(e))] <- 0
##         mqr <- qr.lm(model)
##         n <- as.integer(nrow(mqr$qr))
##         if (is.na(n))
##             stop("invalid model QR matrix")
##         if (NROW(e) != n)
##             stop("non-NA residual length does not match cases used in fitting")
##         do.coef <- as.logical(do.coef)
##         tol <- 10 * .Machine$double.eps
##         res <- .Call(C_influence, mqr, do.coef, e, tol)
##         if (!is.null(model$na.action)) {
##             hat <- naresid(model$na.action, res$hat)
##             hat[is.na(hat)] <- 0
##             res$hat <- hat
##             if (do.coef) {
##                 coefficients <- naresid(model$na.action, res$coefficients)
##                 coefficients[is.na(coefficients)] <- 0
##                 res$coefficients <- coefficients
##             }
##             sigma <- naresid(model$na.action, res$sigma)
##             sigma[is.na(sigma)] <- sqrt(deviance(model)/df.residual(model))
##             res$sigma <- sigma
##         }
##     }
##     res$wt.res <- naresid(model$na.action, e)
##     res$hat[res$hat > 1 - 10 * .Machine$double.eps] <- 1
##     names(res$hat) <- names(res$sigma) <- names(res$wt.res)
##     if (do.coef) {
##         rownames(res$coefficients) <- names(res$wt.res)
##         colnames(res$coefficients) <- names(coef(model))[!is.na(coef(model))]
##     }
##     res
## }
## <bytecode: 0x0000000013b549a8>
## <environment: namespace:stats>
```

```
head(lm.influence(model.crabs)$hat)
```

```
##           1           2           3           4           5           6
## 0.004717405 0.003252191 0.002850832 0.002122182 0.002328025 0.002254075
```

```
tail(lm.influence(model.crabs)$hat)
```

```
##         467         468         469         470         471         472
## 0.005257815 0.002524655 0.002646827 0.002120210 0.002310611 0.004258370
```

Możemy powiedzieć, że wyniki w obu przypadkach są takie same. ***

Praktyczną zasadą jest uznawanie punktów za bardzo wpływowe, gdy \[h_{ii} > \frac{2p}{n}\]

Zadanie: Oblicz ile punktów danych zostanie uznanych za bardzo wpływowe według tego wzoru

```
p = 2
w = 2*p/n
suma=0
for (i in 1:n){
  if (hii[i] > w){
    suma=suma+1
  }
}
```

Liczba wpływowych danych jest równa

```
suma
```

```
## [1] 19
```

Można udowodnić, że \[Var(e_i) = (1-h_{ii}) \sigma^2\] gdzie \(e_i\) oznacza resztę dla \(i\)-tej obserwacji.

Poniższy kod tworzy funkcję `rand.df`, która tworzy zbiór danych ze zmiennymi `x` i `y` wygenerowanymi z modelu regresji liniowej. Zmienna `x` przyjmuje wartości od 1 do 6 z dodanym jedym punktem odstającym o dużej dźwigni.

```
k<-6
rand.df <-function(){
  e <-rnorm(k+1,mean = 0,sd = 30)
  x<-c(1:k, 7*k)
  y<-2*x+3+e
  data.frame(x=x,y=y)
}
```

Zadanie: z poniższego modelu wygeneruj dane, dokonaj regresji liniowej i zapisz reszty. Proces powtórz odpowiednią liczbę razy, aby móc zwizualizować jak dźwignia wpływa na wariancję reszt.

```
x=rand.df()[,1]
y=rand.df()[,2]
model=lm(y~x)
reszty=model$resid
```

Reszty modelu z pierwszej generacji danych

```
reszty
```

```
##          1          2          3          4          5          6
##  13.951281 -13.249054  57.228513 -30.748454 -39.041066   9.416897
##          7
##   2.441882
```

```
x1=rand.df()[,1]
y1=rand.df()[,2]
model1=lm(y1~x1)
reszty1=model1$resid
```

Reszty modelu z drugiej generacji danych

```
reszty1
```

```
##          1          2          3          4          5          6
## -15.796618  12.669840   6.130374  29.244671  -2.052879 -31.487719
##          7
##   1.292329
```

```
x2=rand.df()[,1]
y2=rand.df()[,2]
model2=lm(y2~x2)
reszty2=model2$resid
```

Reszty modelu z trzeciej generacji danych

```
reszty2
```

```
##           1           2           3           4           5           6
##  12.6900339  12.9352104 -58.7585362  -0.7724109  -1.5253339  37.2131132
##           7
##  -1.7820765
```

```
x3=rand.df()[,1]
y3=rand.df()[,2]
model3=lm(y3~x3)
reszty3=model3$resid
```

Reszty modelu z czwartej generacji danych

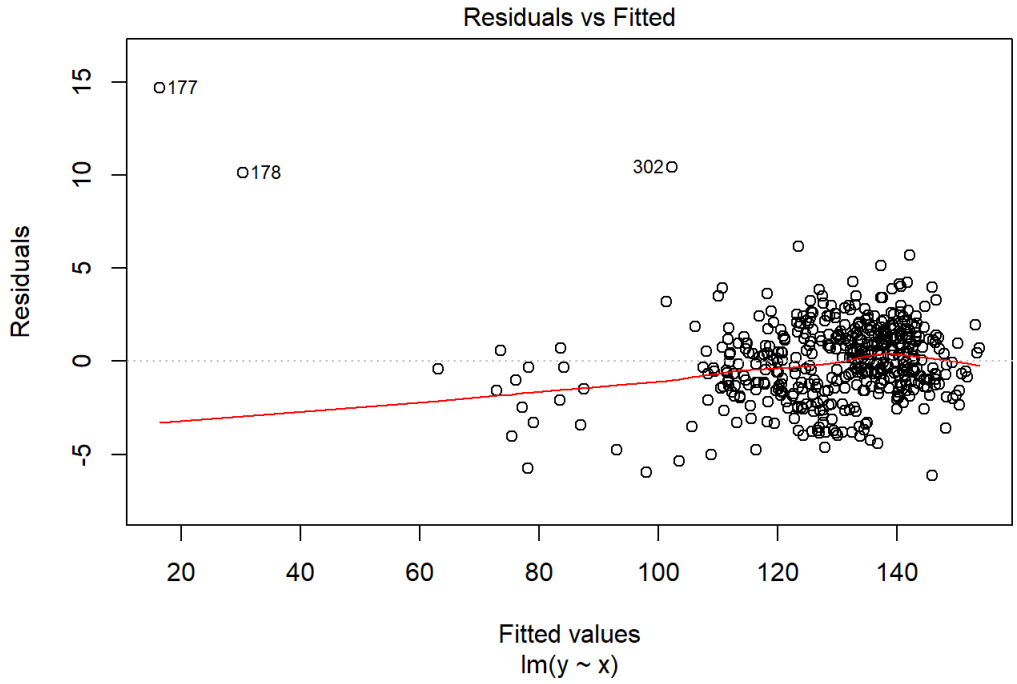
```
reszty3
```

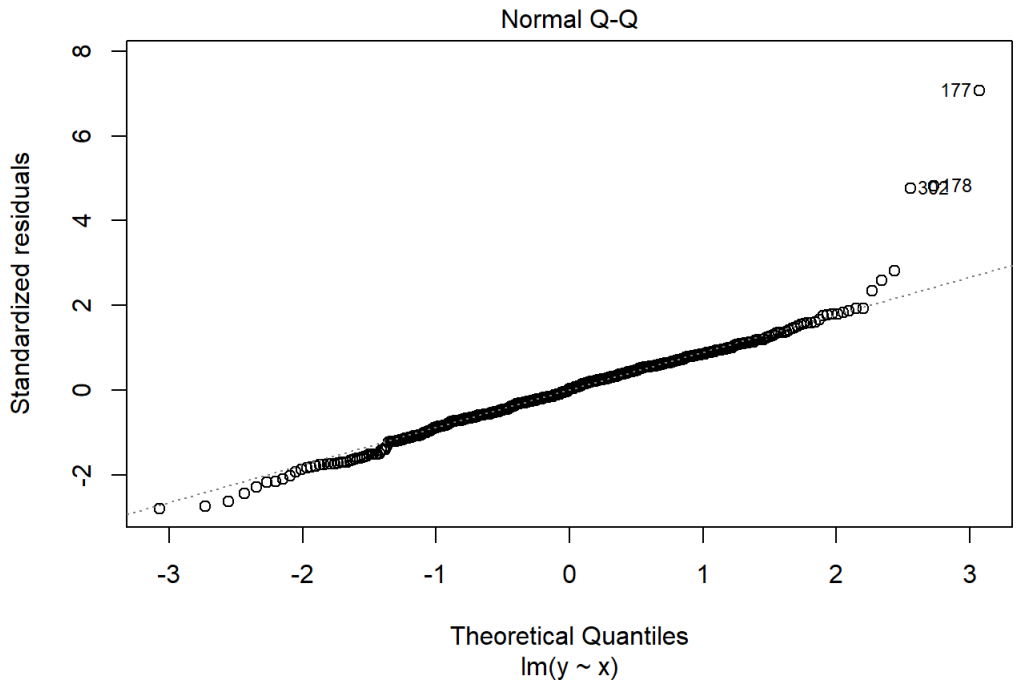
```
##         1         2         3         4         5         6         7
##  49.19034  12.16015 -49.04750 -47.76494  15.97823  17.59766   1.88606
```

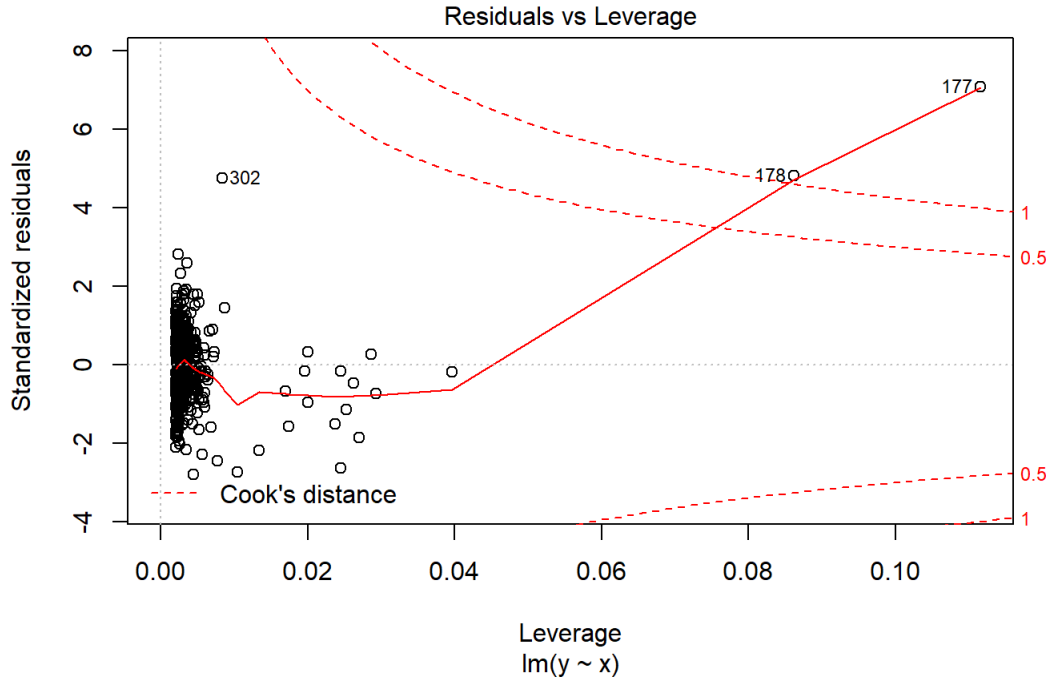
Wykresy diagnostyczne

Funkcja `plot` wyrysowuje wykresy diagnostyczne dla regresji.

```
plot(model.crabs)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x)
177
178
302


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x)
177
178
302



Pierwszy wykres przedstawia reszty wykreślone względem wartości dopasowanych. Punkty powinny być równomiernie rozrzucone względem osi `x`. Czerwona linia (wygładzenie) wskazuje na lekkie niedopasowane modelu dla mniejszych wartości.

Drugim wykresem jest wykres kwantyl-kwantyl standaryzowanych reszt. Reszty standaryzowane, są to reszty podzielone przez odchylenie standardowe: $$t_i = \frac{e_i}{s \sqrt{1 - h_{ii}} }$$ Wykres kwantyl-kwantyl powinien przypominać linię prostą. Tutaj widzimy pewne niedopasowanie w przypadku bardziej ekstremalnych obserwacji.

Trzeci wykres jest wygodny do badania homoskedastyczności (czerwona krzywa powinna być jak najbardziej pozioma).

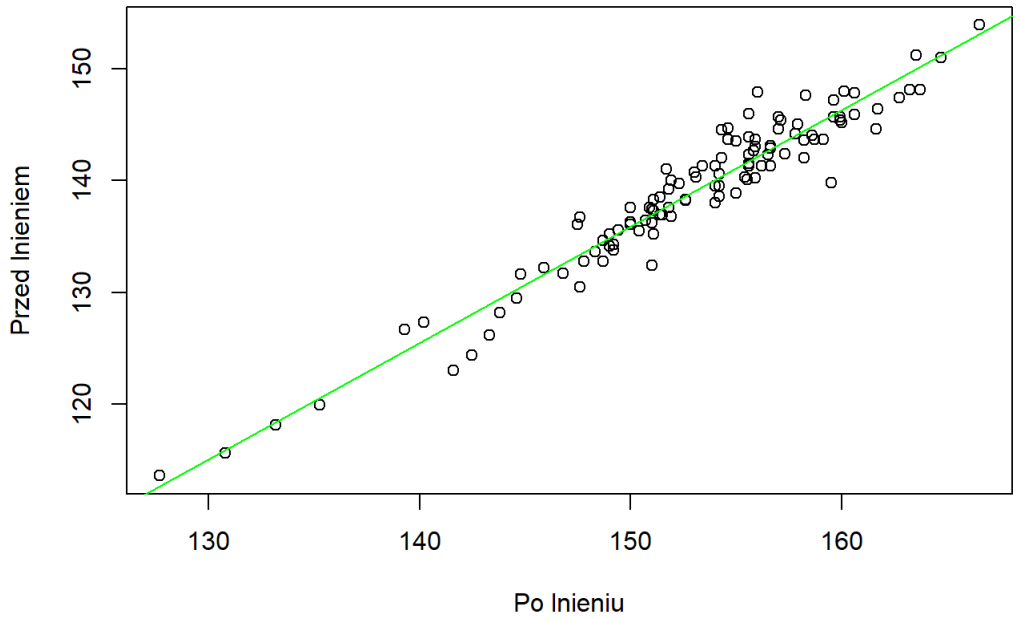
Czwarty wykres służy do badania, które obserwacje są wpływowe. Uwidoczniona jest na nim odległość Cooka, która bada różnicę predykcji modelu wyjściowego z modelem po usunięciu obserwacji. Można udowodnić, że wyraża się ona wzorem $$C_i = e_i \left(\frac{n-p}{p} \cdot\frac{h_{ii}}{1 - h_{ii}} \right)^{1/2} $$

Jak widać obserwacje 177 i 178 są najbardziej wpływowe.

Zadanie: dopasuj model, który uwzględnia tylko obserwacje z połowów a nie z laboratorium. ***

```
polow =crabs[crabs$lf == 0,]
x<-polow$postsz
y<-polow$presz
model.polow<-lm(y~x)

plot(polow$postsz,polow$presz, xlab="Po lnieniu", ylab="Przed lnieniem")
abline(lm(y~x), col=("green"))
```



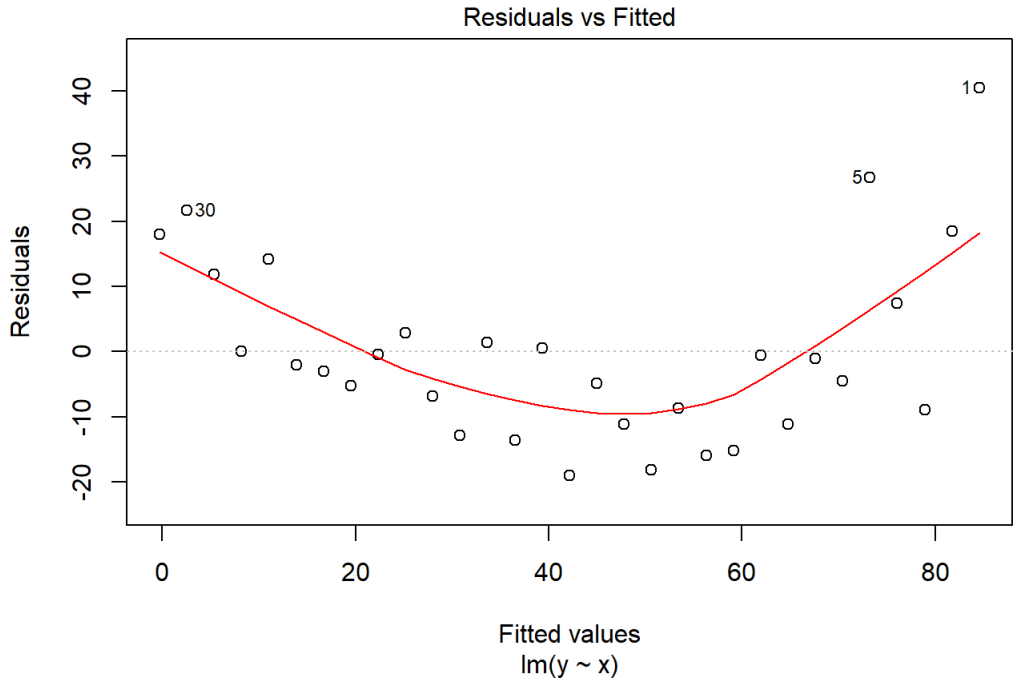
***

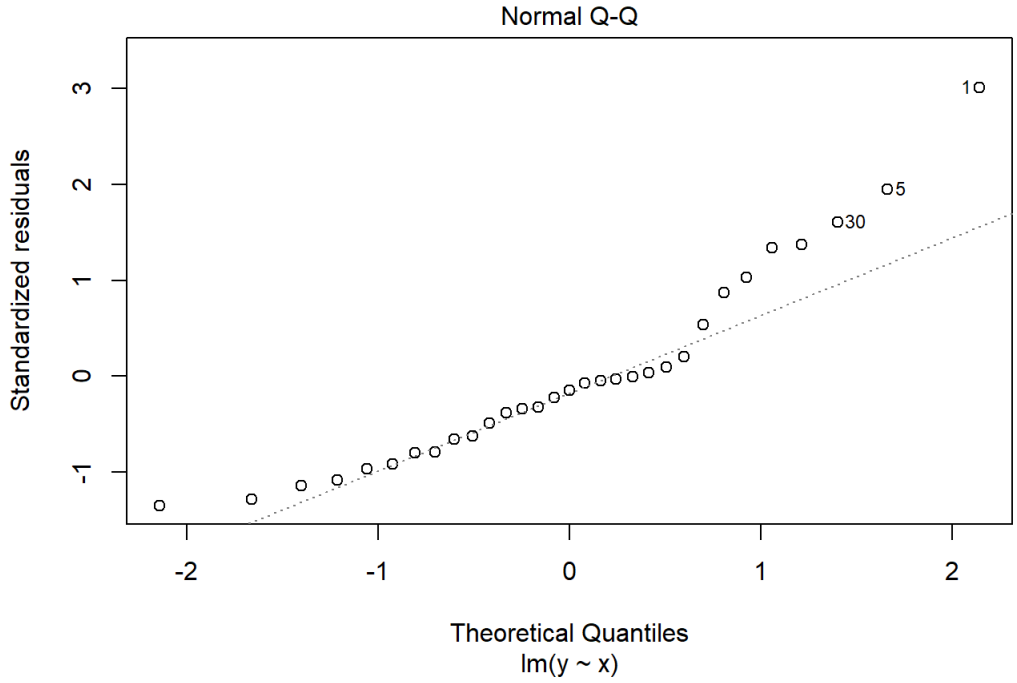
Rozpad materiału (dopasowanie modelu)

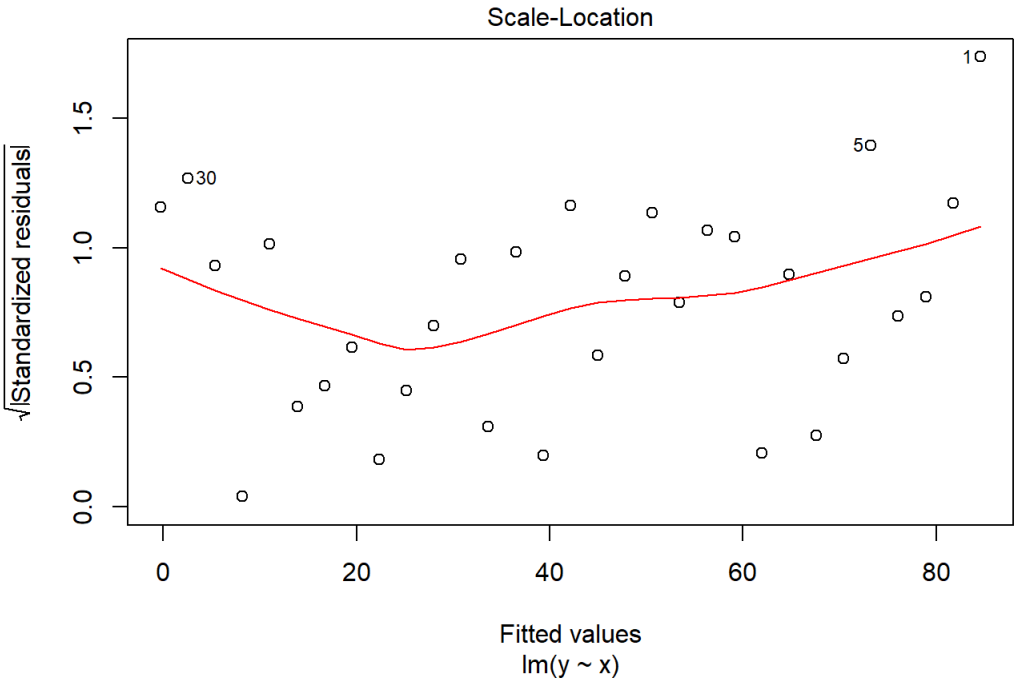
W kolejnych paragrafach zajmiemy się drugim zbiorem danych (rozpad materiału w zależności od czasu).

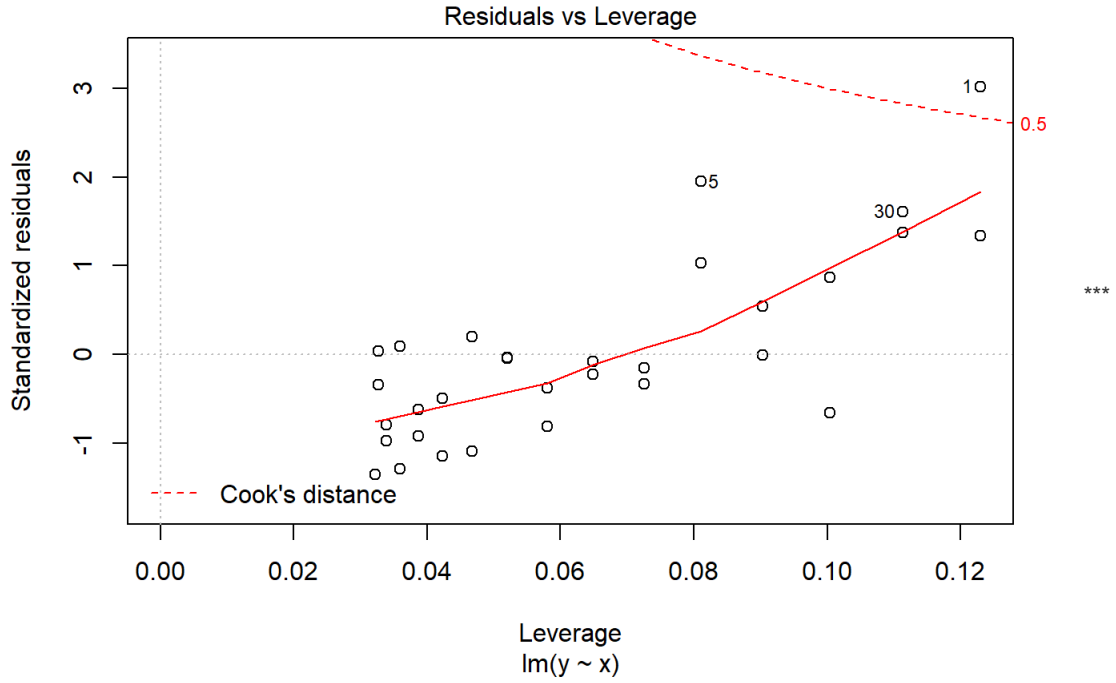
Zadanie: dopasuj model liniowy, który wyjaśnia zależność ilości materiału od czasu. Jak wyglądają wykresy diagnostyczne?

```
y<-decay$amount
x<-decay$time
model.dec<-lm(y~x)
plot(model.dec)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x)
30
5
1


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x)
30
5
1



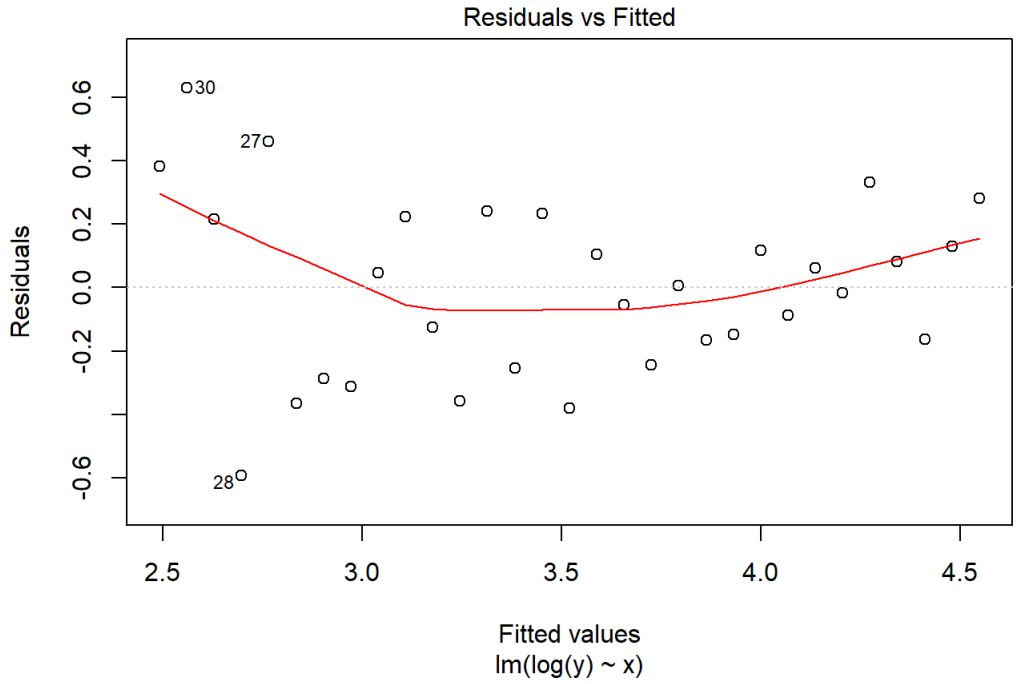


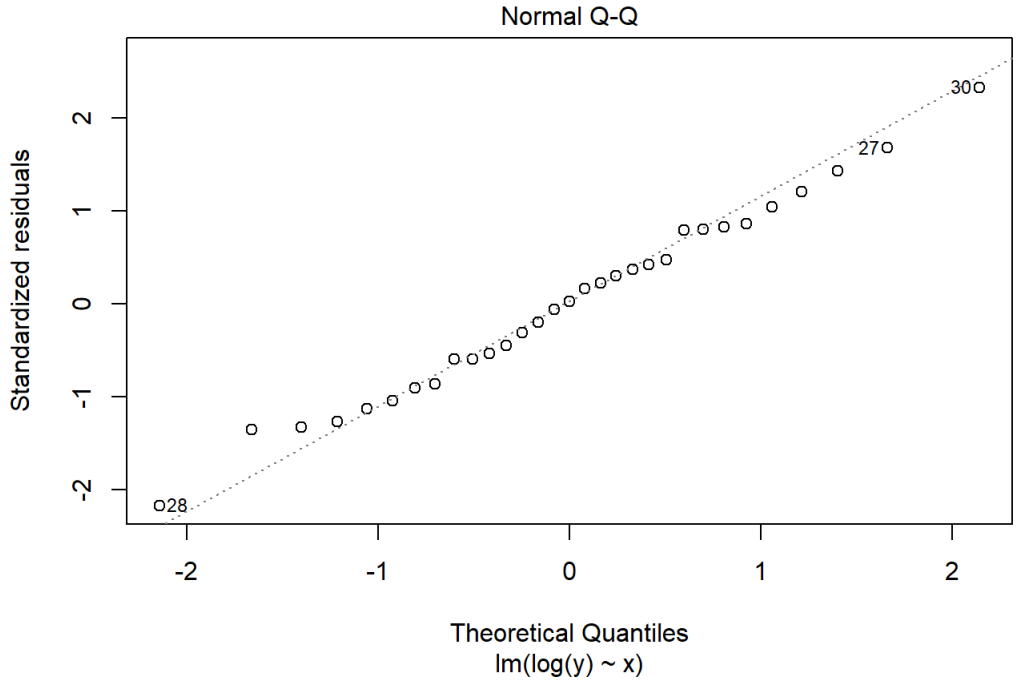
Rozpad materiału (model fizyczny)

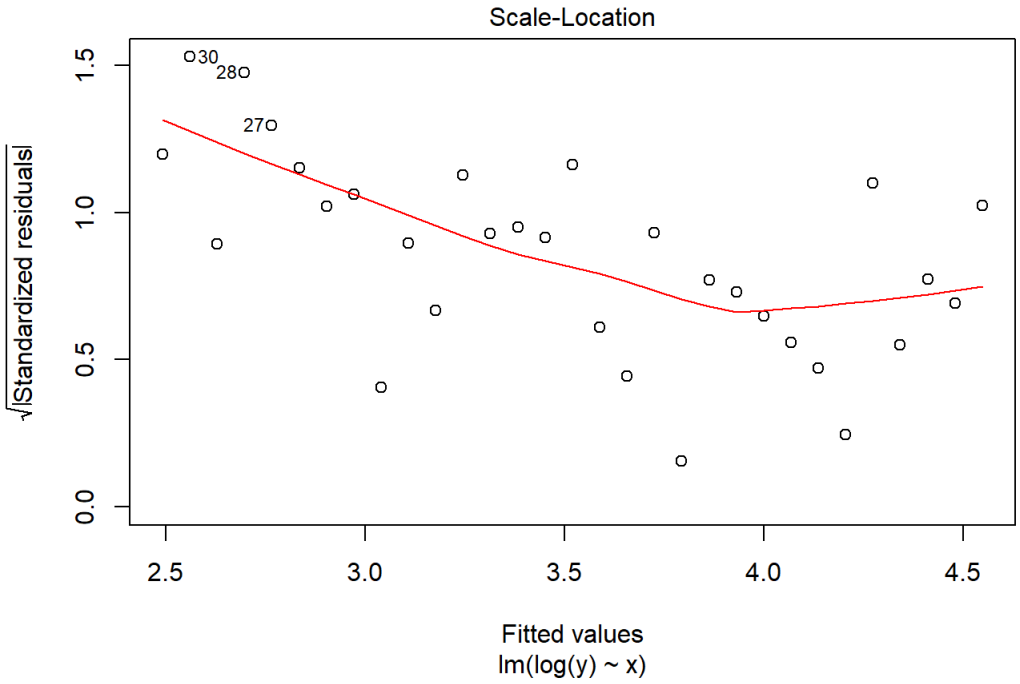
Materiały ulegające rozpadowi podlegają prawu, które powiada, że rozpad materiału w czasie jest proporcjonalny do ilości materiału. Z tego wynika, że ilość materiału zmienia się w czasie zgodnie z prawem wykładniczym \[y = y_0 \cdot e^{-bt}\] Zatem możemy otrzymać następującą zależność \[\log y = \log y_o + b t \] Możemy więc przypuszczać, że zlogarytmowanie zmiennej zależnej mooże poprawić nasz model.

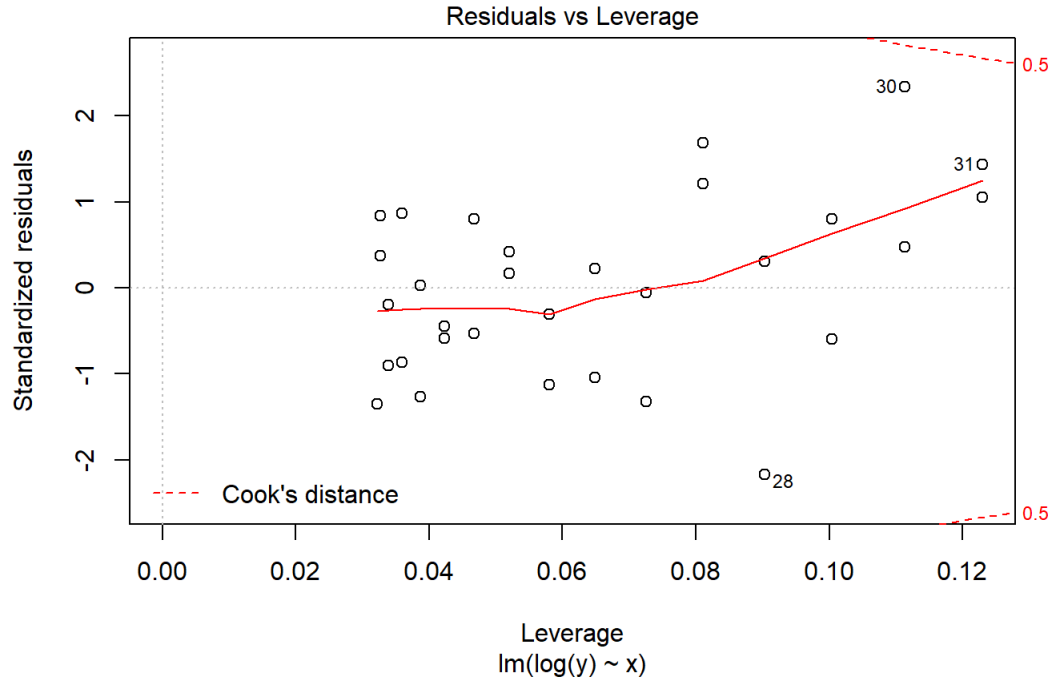
Zadanie: Dopasuj model do danych z zlogarytmowaną zmienną zależną. Jak teraz wyglądają wykresy diagnostyczne?

```
model.declog <- lm(log(y)~x)
plot(model.declog)
```


Residuals vs Fitted
Residuals
Fitted values
lm(log(y) ~ x)
30
27
28
-0.6
-0.2
0.0
0.2
0.4
0.6
2.5
3.0
3.5
4.0
4.5


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(log(y) ~ x)
30
27
28
-2
-1
0
1
2





Wykresy diagnostyczne są podobne. ***