Modele regresji i ich zastosowania
Labolatoria 7 i 8

Jan Solarz
243889

7 maja 2021

Spis treści

1 Zadania labolatoryjne

1.1 Zadanie 1

Generujemy $n = 100$ obserwacji $Y1, ..., Yn$ postaci

$$Y_i = \beta_1 + \beta_2 e^{-\beta_3 x} + \epsilon_i \quad (1)$$

gdzie $\epsilon_1, ..., \epsilon_n i.i.d.N(0, \sigma^2)$

```
n = 100
set.seed(100)
funkcja1 <- function(i) {
  x_i <- 10*i/n
  return(x_i)
}

x_100<-seq(1,100)

x_i<-funkcja1(x_100)


beta1 = 80
```

```r
beta2 = 100
beta3 = 0.005
sigma2 = 0.5

epsilon_i<-rnorm(100,0,sigma2)
exp(-beta3*0.1)

## [1] 0.9995001

y_i<-beta1+beta2*exp(-beta3*x_i)+epsilon_i

head(cbind(x_i,y_i))

##      x_i      y_i
## [1,] 0.1 179.6989
## [2,] 0.2 179.9658
## [3,] 0.3 179.8107
## [4,] 0.4 180.2436
## [5,] 0.5 179.8088
## [6,] 0.6 179.8598
```

1.2 Zadanie 2

Znajdujemy postać funkcji $g(x,\beta)$ oraz kolejno elementry gradientu G, które są pochodnymi cząstkowymi:

$$g(x,\beta) = \beta_1 + \beta_2 e^{-\beta_3 x} \tag{2}$$

$$\frac{\delta g(x,\beta_1)}{\delta\beta_1} = 1 \tag{3}$$

$$\frac{\delta g(x,\beta_2)}{\delta\beta_2} = e^{-\beta_3 x} \tag{4}$$

$$\frac{\delta g(x,\beta_3)}{\delta\beta_3} = -x\beta_2 e^{-\beta_3 x} \tag{5}$$

1.3 Zadanie 3 i 4

Implementujemy algorytm Gaussa- Newtona zadając parametry wstępne 79,101,0.004. Algorytm przerywa estymacje kolejnych parametrów po spełnieniu kryterium stopu. Tworzymy tabele ze wszystkimi wykonanymi krokami, gdzie liczona jest również sigma oraz wartość log-likelihood.

```r
install.packages("MASS")

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror

library(MASS)
matrix0<-matrix(rep(0,5000),ncol=5,nrow=1000)
beta0 <- c(79.0000000,101.0000000,0.0040000)
```

```r
for(i in 1:1000) {
  e = y_i - beta0[1] - beta0[2]*exp(-beta0[3]*x_i)
  sigma<-(1/n)*(sum(e^2))
  likehood<-(-n/2)*log(2*pi)-(n/2)*log(sigma)-n/2
  matrix0[i,]<-c(beta0,sigma,likehood)

  gradient = cbind(rep(1,n), exp(-beta0[3]*x_i), -beta0[2]*x_i*exp(-beta0[3]*x_i))
  beta1<-beta0 + ginv(t(gradient)%*%gradient)%*%t(gradient)%*%e


  if(norm(beta0-beta1, type="2")<=0.0001){
    matrix0[i+1,]<-c(beta1,sigma,likehood)
    break
  }
  else{
  beta0<-beta1
  }

}
head(matrix0)

##           [,1]     [,2]        [,3]      [,4]       [,5]
## [1,] 79.00000 101.0000 0.004000000 0.5676750 -113.58354
## [2,] 79.03417 101.0342 0.005079991 0.2563989  -73.84280
## [3,] 79.03462 101.0346 0.005085324 0.2563936  -73.84178
## [4,] 79.03462 101.0346 0.005085323 0.2563936  -73.84178
## [5,]  0.00000   0.0000 0.000000000 0.0000000    0.00000
## [6,]  0.00000   0.0000 0.000000000 0.0000000    0.00000

matrix0<-matrix0[1:4,1:5]
matrix0

##           [,1]     [,2]        [,3]      [,4]       [,5]
## [1,] 79.00000 101.0000 0.004000000 0.5676750 -113.58354
## [2,] 79.03417 101.0342 0.005079991 0.2563989  -73.84280
## [3,] 79.03462 101.0346 0.005085324 0.2563936  -73.84178
## [4,] 79.03462 101.0346 0.005085323 0.2563936  -73.84178

colnames(matrix0)<-c("beta_1","beta_2","beta_3","sigma","log-likelikhood")
row.names(matrix0)<-c(0,1,2,3)
matrix0

##     beta_1   beta_2      beta_3     sigma log-likelikhood
## 0 79.00000 101.0000 0.004000000 0.5676750      -113.58354
## 1 79.03417 101.0342 0.005079991 0.2563989       -73.84280
## 2 79.03462 101.0346 0.005085324 0.2563936       -73.84178
## 3 79.03462 101.0346 0.005085323 0.2563936       -73.84178
```

- Zostały wykonane 3 iteracje.
- Widzimy bardzo niskie różnice miedzy kolejnymi estymacjami
- Krok 3 nie ma już sensu

1.4 Zadanie 5

Porównanie parametrów, różnice

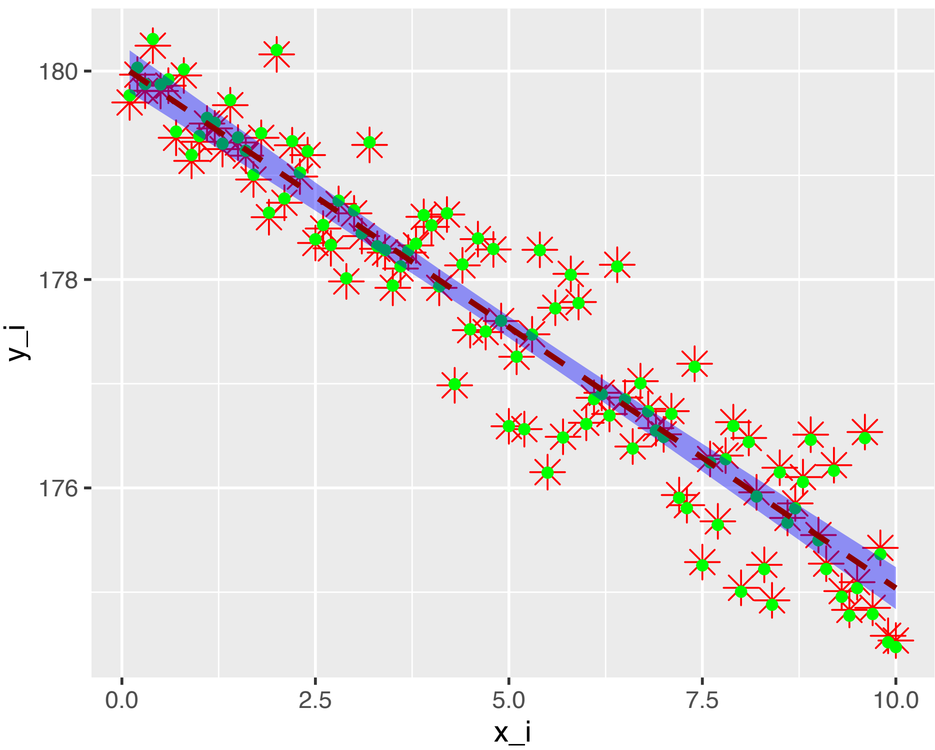
```
beta000 <- c(79.0000000,101.0000000,0.0040000)
matrix0[3,1:3]-beta000
```

```
##      beta_1      beta_2      beta_3
## 0.034617830 0.034622669 0.001085324
```

1.5 Zadanie 6

```
beta_00<-matrix0[4,1:3]
y_i_2<-beta_00[1]+beta_00[2]*exp(-beta_00[3]*x_i)+epsilon_i
y<-c(y_i,y_i_2)

library(ggplot2)
ggplot(data.frame(x_i,y_i), aes(x = x_i, y = y_i)) +
  geom_point(shape=8, color="red", lwd =4)+
  geom_point(data = data.frame(x_i,y_i_2), aes(x = x_i, y = y_i_2),col = "green")+
  geom_smooth(method=lm,  linetype="dashed",
              color="darkred", fill="blue")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



- Widzimy, że punkty niemal się całkowicie pokrywają na wykresie

1.6 Zadanie 7

Macierz kowariancji

```
cov_matrix<-data.frame(matrix0[1:3,1],matrix0[1:3,2],matrix0[1:3,3])
cov(cov_matrix)

##                  matrix0.1.3..1. matrix0.1.3..2. matrix0.1.3..3.
## matrix0.1.3..1.     3.943752e-04    3.944301e-04    1.241305e-05
## matrix0.1.3..2.     3.944301e-04    3.944851e-04    1.241478e-05
## matrix0.1.3..3.     1.241305e-05    1.241478e-05    3.907229e-07
```

Macierz kowariancji jest bliska macierzy zerowej.

1.7 Zadanie 8

Parametry nieznacznie różniące się od β początkowej

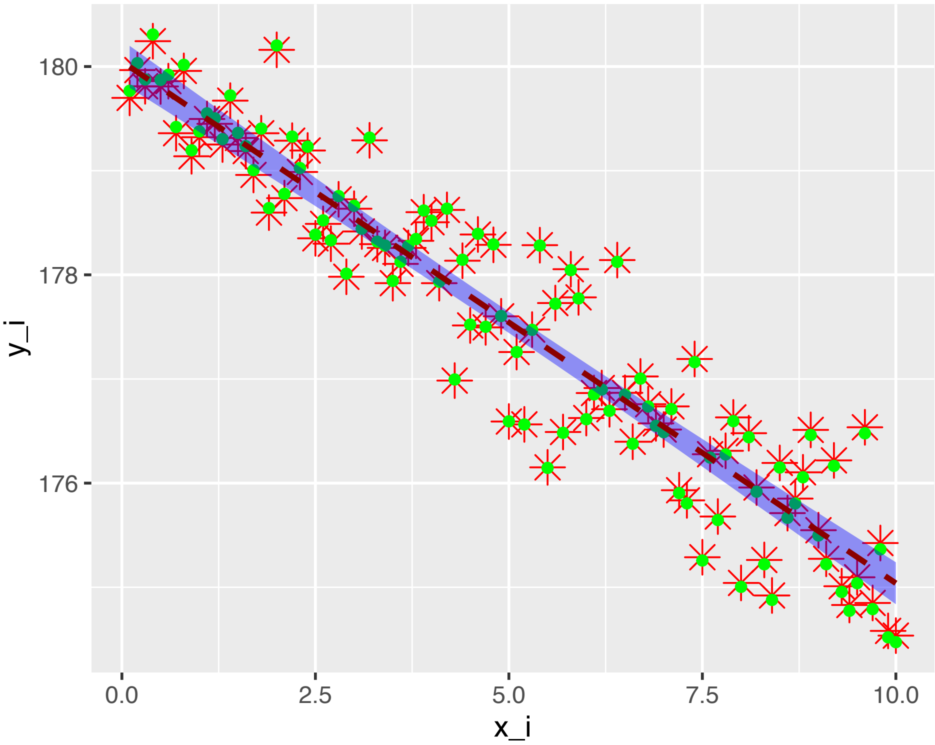
```
matrix0<-matrix(rep(0,5000),ncol=5,nrow=1000)
beta0 <- c(82,102,0.006)
for(i in 1:1000) {
  e = y_i - beta0[1] - beta0[2]*exp(-beta0[3]*x_i)
  sigma<-(1/n)*(sum(e^2))
  likehood<-(-n/2)*log(2*pi)-(n/2)*log(sigma)-n/2
  matrix0[i,]<-c(beta0,sigma,likehood)
```

```r
  gradient = cbind(rep(1,n), exp(-beta0[3]*x_i), -beta0[2]*x_i*exp(-beta0[3]*x_i))
  beta1<-beta0 + ginv(t(gradient)%*%gradient)%*%t(gradient)%*%e


  if(norm(beta0-beta1, type="2")<=0.0001){
    matrix0[i+1,]<-c(beta1,sigma,likehood)
    break
  }
  else{
  beta0<-beta1
  }

}
matrix0<-matrix0[1:4,1:5]
colnames(matrix0)<-c("beta_1","beta_2","beta_3","sigma","log-likelikhood")
row.names(matrix0)<-c(0,1,2,3)
matrix0
```

```
##      beta_1   beta_2      beta_3      sigma log-likelikhood
## 0 82.00000 102.0000 0.006000000 12.2473392      -267.15929
## 1 80.03468 100.0341 0.005150263  0.2564581       -73.85435
## 2 80.03503 100.0344 0.005137497  0.2563978       -73.84260
## 3 80.03503 100.0344 0.005137500  0.2563978       -73.84260
```

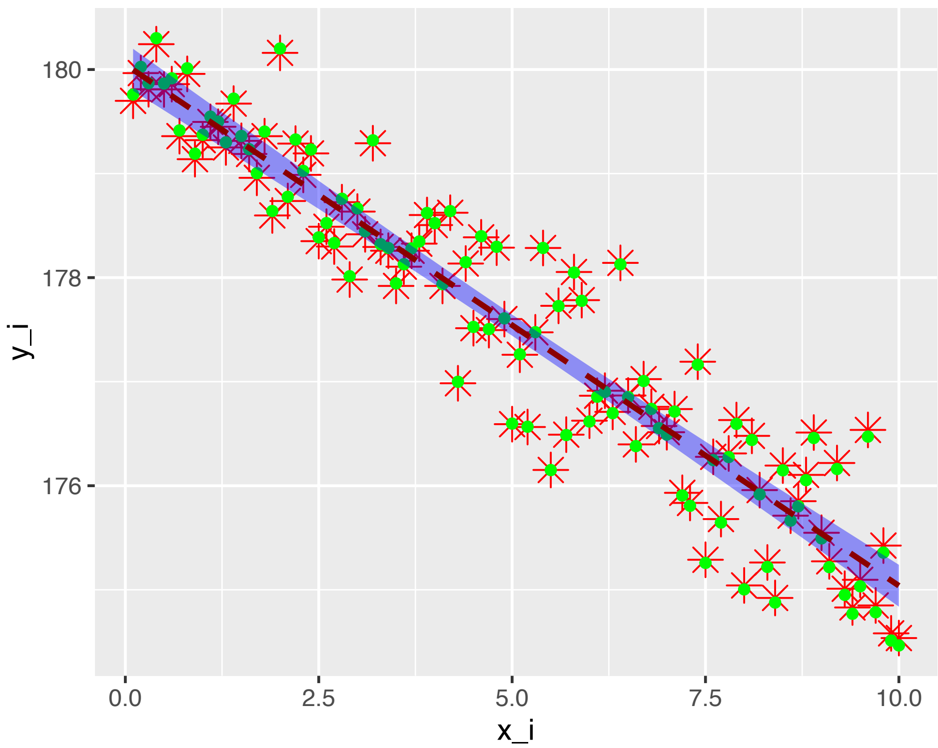
```r
beta_00<-matrix0[4,1:3]
y_i_2<-beta_00[1]+beta_00[2]*exp(-beta_00[3]*x_i)+epsilon_i
y<-c(y_i,y_i_2)

ggplot(data.frame(x_i,y_i), aes(x = x_i, y = y_i)) +
  geom_point(shape=8, color="red", lwd =4)+
  geom_point(data = data.frame(x_i,y_i_2), aes(x = x_i, y = y_i_2),col = "green")+
  geom_smooth(method=lm,  linetype="dashed",
              color="darkred", fill="blue")
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
matrix0<-matrix(rep(0,5000),ncol=5,nrow=1000)
beta0 <- c(43,180,0.01)
for(i in 1:1000) {
  e = y_i - beta0[1] - beta0[2]*exp(-beta0[3]*x_i)
  sigma<-(1/n)*(sum(e^2))
  likehood<-(-n/2)*log(2*pi)-(n/2)*log(sigma)-n/2
  matrix0[i,]<-c(beta0,sigma,likehood)

  gradient = cbind(rep(1,n), exp(-beta0[3]*x_i), -beta0[2]*x_i*exp(-beta0[3]*x_i))
  beta1<-beta0 + ginv(t(gradient)%*%gradient)%*%t(gradient)%*%e


  if(norm(beta0-beta1, type="2")<=0.0001){
    matrix0[i+1,]<-c(beta1,sigma,likehood)
    break
  }
  else{
  beta0<-beta1
  }

}
matrix0<-matrix0[1:5,1:5]
colnames(matrix0)<-c("beta_1","beta_2","beta_3","sigma","log-likelikhood")
row.names(matrix0)<-c(0,1,2,3,4)
matrix0

##      beta_1    beta_2      beta_3         sigma log-likelikhood
```

```
## 0 43.00000 180.0000 0.010000000 1358.5466047      -502.60239
## 1 21.50864 158.4902 0.003812235    0.6096497      -117.15031
## 2 21.53964 158.5212 0.003209045    0.2562568       -73.81509
## 3 21.53991 158.5214 0.003211053    0.2562549       -73.81473
## 4 21.53991 158.5214 0.003211053    0.2562549       -73.81473

beta_00<-matrix0[4,1:3]
y_i_2<-beta_00[1]+beta_00[2]*exp(-beta_00[3]*x_i)+epsilon_i
y<-c(y_i,y_i_2)

ggplot(data.frame(x_i,y_i), aes(x = x_i, y = y_i)) +
  geom_point(shape=8, color="red", lwd =4)+
  geom_point(data = data.frame(x_i,y_i_2), aes(x = x_i, y = y_i_2),col = "green")+
  geom_smooth(method=lm,  linetype="dashed",
              color="darkred", fill="blue")

## `geom_smooth()` using formula 'y ~ x'
```



- Widzimy, że zarówno dla małych jak i dużych rozbieżności parametrów wstępnych zauważyć możemy szybkie dopasowanie estymatorów (w 3 i 4 krokach) oraz bardzo bliskie dopasowanie na wykresach.