Raport 4

Magdalena Wawrzyniak

Zadanie 1

a)

Na początek generujemy macierz X, wektor błędów losowych i wektor zmiennej odpowiedzi zgodnie z wytycznymi z zadania. Dla pierwszych trzech podpunktów usatalam ziarno równe 27.

```
library(MASS)

## funkcja generująca dane:

generate <- function(seed = 27, Sigma = diag(0.1, 2) + 0.9){

  set.seed(seed)

  X  <- mvrnorm(n = 100, c(0, 0), Sigma = Sigma/100)
  X1 <- X[, 1]
  X2 <- X[, 2]

  epsilon <- rnorm(100)

  Y <- 3*X1 + epsilon
  return(data.frame(Y, X1, X2))
}


## dane dla podpunktów a-c

dane <-  generate()
Y <- dane[, 1]
X1 <- dane[, 2]
X2 <- dane[, 3]
```

b)

Tworzymy dwa modele, model 1 z jedną zmienną objaśniającą X_1 i model 2 z dwiema zmiennymi X_1 oraz X_2

```
## tworzenie modeli

model1 <- lm(Y~X1)
model2 <- lm(Y~X1+X2)
```

Tworzę funkcje, która wypisze mi w tabelce najpotrzebniejsze informacje dla tego podpunktu (i podpunktu d)

```
## funkcja wypisująca potrzebne informacje

summary_model <- function(model, alpha = 0.05){

  confidance_interval <- confint(model)[2,]
  p_value <- summary(model)$coefficient[2,4]
  summary_data <- data.frame(c(confidance_interval, p_value))
  rownames(summary_data) = c("2.5%", "97.5%", "p-wartość dla beta1")
  colnames(summary_data) = c(" ")

  return(summary_data)

}

## informacje o modelach

{
info_b <- data.frame(summary_model(model1), summary_model(model2))
colnames(info_b) <- c("Model 1", "Model 2")
info_b
}
```

```
##                         Model 1     Model 2
## 2.5%                0.898546444 -0.25395259
## 97.5%               4.810470796  9.58073697
## p-wartość dla beta1 0.004659438  0.06280319
```

Widzimy, że przedział ufności w modelu 2 jest około dwa razy szerszy, niż w modelu 1. Dostajemy również w oparciu o p-wartość, że wyniki testu na niezależność danych od X_1 są różne dla obu modeli, gdzie w przypadku modelu 1 możemy odrzucić hipotezę zerową, która mówi o tym że Y nie zależy od X_1, natomiast dla modelu 2 nie mamy podstaw do odrzucenia hipotezy zerowej. Możemy stąd wnioskować, że model 2 gorzej odzwierciedla nasze dane.

c)

Obliczam ręcznie odchylenia standardowe, i błędy standardowy dla parametru β_1 dla modelu 1.

```
## wyliczenia ręczne odchylenia standardowego dla modelu 1

n <- length(X1)
beta0_m1 <-summary(model1)$coefficients[1]
beta1_m1 <- summary(model1)$coefficients[2]

sigma2_by_hand_m1 <- sum((Y - (beta0_m1 + beta1_m1*X1))^2)/(n-2)
sigma_by_hand_m1 <- sqrt(sigma2_by_hand_m1)
sigma_m1 <- summary(model1)$sigma
su <- summary(model1)
std_error1 <- summary(model1)$coefficients[2,2]
std_error_by_hand_m1 <- sqrt(sigma2_by_hand_m1/sum((X1 - mean(X1))^2))
```

Następnie wyznaczam funkcję mocy testu, z której będę korzystać przy wyliczeniach dla obu modeli. Funkcja przyjmuje błąd standardowy i parametr β_1, a także wielkość próby i liczbę stopni swobody.

```
# funkcja mocy testu

#beta1/s_beta1 #parametr niecentralności

power <- function(n, df, beta1, std_error){

  tc_model1 <- qt(1-0.05/2, n - df)
  1 - pt(tc_model1, n-df, beta1/std_error) +
      pt(-tc_model1, n-df, beta1/std_error)

}

power_m1 <- power(100, 2, beta1_m1, std_error_by_hand_m1)
```

Obliczam ręcznie odchylenia standardowe, i błędy standardowy dla parametru β_1 dla modelu 1.

```
## wyliczenia ręczne odchylenia standardowego dla modelu 2

beta0_m2 <- summary(model2)$coefficients[1]
beta1_m2 <- summary(model2)$coefficients[2]
beta2_m2 <- summary(model2)$coefficients[3]
sigma2_by_hand_m2 <- sum((Y - (beta0_m2 + beta1_m2*X1 + beta2_m2*X2))^2)/(n-3)
sigma_by_hand_m2 <- sqrt(sigma2_by_hand_m2)
sigma_m2 <- summary(model2)$sigma

std_error2 <- summary(model2)$coefficients[2,2]
std_error_by_hand_m2 <- sqrt(sigma2_by_hand_m2/
                               sum((X1 - mean(X1))^2)) # tu jest gdzieś błąd


power_m2 <- power(100, 3, beta1_m2, std_error1)

power_m2 <- power(100, 3, beta1_m2, std_error2)
std_deviation_2_date <- data.frame(c(sigma_by_hand_m2, sigma_m2,
                                     std_error_by_hand_m2, std_error2,
                                     power_m2))
rownames(std_deviation_2_date) <- c("odchylenie standardowe wyliczony ręcznie",
                                    "odchylenie standardowe z R",
                                    "błąd wyliczony ręcznie", "błąd z R",
                                    "moc testu" )
colnames(std_deviation_2_date) <- c("Model 2")
```

W związku z tym, że w wyliczeniach błędu standardowego ręcznie jest błąd dla modelu 2, to moc testu wyliczyłam na podstawie tego co otrzymałam z funkcji z R. Poniżej mamy tabelą z wynikami dla obu modeli.

```
# wyniki
std_deviation_1_date <- data.frame(c(sigma_by_hand_m1, sigma_m1, std_error_by_hand_m1,
                                     std_error1, power_m1))
rownames(std_deviation_1_date) <- c("odchylenie standardowe wyliczony ręcznie",
```

```
                                    "odchylenie standardowe z R",
                                    "błąd wyliczony ręcznie", "błąd z R", "moc testu")
colnames(std_deviation_1_date) <- c("Model 1")

short_summary <- data.frame(std_deviation_1_date, std_deviation_2_date)
short_summary
```

```
##                                          Model.1   Model.2
## odchylenie standardowe wyliczony ręcznie 1.0552875 1.0572652
## odchylenie standardowe z R               1.0552875 1.0572652
## błąd wyliczony ręcznie                   0.9856358 0.9874830
## błąd z R                                 0.9856358 2.4775973
## moc testu                                0.8179700 0.4616727
```

Odchylenia standardowe dla obu modeli są sobie bardzo bliskie, ale dla modelu 1 jest odrobinę mniejsze.

Widzimy, że moc testu dla modelu 2 nie jest zadowalająca. Odrzucanie wyniku na podstawie tego testu jest prawie tak skuteczny jak odrzucanie go na podstawie wyniku rzutu monetą.

d)

```
v_beta1_m1 = c()
v_beta1_m2 = c()
p_value_m1 = c()
p_value_m2 = c()
std_error_m1 = c()
std_error_m2 = c()
power_m1 = c()
power_m2 = c()

for (i in 1:1000){

# generujemy dane
 dane <- generate(seed = i)
  Y <- dane[, 1]
  X1 <- dane[, 2]
  X2 <- dane[, 3]

# tworzymy modele
  model1 <- lm(Y~X1)
  model2 <- lm(Y~X1+X2)

# parametr beta1 modelu 1
  v_beta1_m1[i] <- summary(model1)$coefficients[2]

# p-wartość dla modelu 1
  p_value_m1[i] = summary(model1)$coefficient[2,4]

# parametr beta1 modelu 2
  v_beta1_m2[i] <- summary(model2)$coefficients[2]

# p-wartość dla modelu 2
```

```
  p_value_m2[i] = summary(model2)$coefficient[2,4]

# odchylenie standardowe dla beta 1 i moc testu dla modelu 1
  std_error_m1[i] <- summary(model1)$coefficients[2,2]
  power_m1[i] = power(100, 2, v_beta1_m1[i], std_error_m1[i])

# odchylenie standardowe dla beta 1 i moc testu dla modelu 2
  std_error_m2[i] <- summary(model2)$coefficients[2,2]
  power_m2[i] = power(100, 3, v_beta1_m2[i], std_error_m2[i])
}
est_beta1_m1 <- mean(v_beta1_m1)
est_std_error_m1 <- mean(std_error_m1)
est_p_value_m1 <- mean(p_value_m1)
est_power_m1 <- mean(power_m1)

est_beta1_m2 <- mean(v_beta1_m2)
est_std_error_m2 <- mean(std_error_m2)
est_p_value_m2 <- mean(p_value_m2)
est_power_m2 <- mean(power_m2)
```

```
teor_m1 <- data.frame(c(beta1_m1, short_summary$Model.1[4], info_b$`Model 1`[3],
                        short_summary$Model.1[5]))
teor_m2 <- data.frame(c(beta1_m2, short_summary$Model.2[4], info_b$`Model 2`[3],
                        short_summary$Model.2[5]))
est_m1 <- data.frame(c(est_beta1_m1, est_std_error_m1, est_p_value_m1, est_power_m1))
est_m2 <- data.frame(c(est_beta1_m2, est_std_error_m2, est_p_value_m2, est_power_m2))
results <- data.frame(teor_m1, est_m1, teor_m2, est_m2)
colnames(results) = c("Model 1 teor.", "Model 1 est.", "Model 2 teor.", "Model 2 est.")
rownames(results) = c("beta 1", "s dla beta 1", "p-wartość", "moc testu")
results
```

```
##              Model 1 teor. Model 1 est. Model 2 teor. Model 2 est.
## beta 1         2.854508620   3.01894309    4.66339219    3.0312025
## s dla beta 1   0.985635804   1.01098365    2.47759729    2.3341361
## p-wartość      0.004659438   0.03622942    0.06280319    0.2864812
## moc testu      0.817969955   0.76108317    0.46167267    0.3366815
```

Dla modelu 1 wyniki teoretyczne i wyestymowane są do siebie zbliżone, niewystępują bardzo duże odchylenia. Wprzypadku wyników dla modelu 2 różnice są trochę większe, może być to rezultat kiepskiego dopasowywania modelu do danych, co generowało więcej błędów.

Zadanie 2

a)

Na początek generujemy macierz X i wektor beta. Ustalamy ziarno równe 27.

```
set.seed(27)

X <- matrix(rnorm(950000, 0, 0.1), nrow = 1000)

beta <- rep(0,1000)
beta[1:5] <- 3
```

b)

Tworzymy funkcje, która wyliczy wartości SSE, MSE, AIC, p-wartości dla pierwszych odpowiadające dwóm pierwszym zmiennym objaśniającym oraz liczba fałszywych odkryć. Wewnątrz funkcji generujemy Y i wyniki zapisujemy w tabeli. Na ich podstawie tworzymy modele i wyliczamy potrzebna w tym zadaniu dane.

```
p = 950
n = c(1, 2, 5, 10, 50, 100, 500, 950)

compiut = function(n){
  Y = 0
  p_val_2 = NA
  for(i in 1:n){
    Y = Y + X[i,]*beta[i]
  }
  Y = Y + rnorm(1000)


  data = data.frame(Y,X)
  lin_m = lm(Y~X[, 1:n] - 1, data=data)
  summ_m = summary(lin_m)

  est_beta = lin_m$coefficients

  sse = sum(lin_m$residuals^2)
  mse = sse/(1000-p)
  aic = AIC(lin_m)
  p_val_1 = summ_m$coefficients[1,4]
  if (i > 2){
    p_val_2 = summ_m$coefficients[2,4]
  }

  f_discovery = rep(0,n)
  for (i in c(1:n)){
    if (summ_m$coefficients[i,4] < 0.05){
      f_discovery[i] = 1
    }
  }
  false_disc = sum(f_discovery[5:n])
  if (n < 6){
    false_disc = 0
  }
  result = c(sse, mse, aic, p_val_1, p_val_2, false_disc)
  for (i in 1:length(result)){
    result[i] = round(result[i],3)
  }
  return(result)
}

tabelka = function(nn){
  results <- data.frame(
    k=double(), sse=double(), mse=double(), aic=double(),
    p_val_1=double(), p_val_2=double(), false_disc=double())
```

```
  for (i in n) {

    res <- compiut(i)

    results <- rbind(results, data.frame(k=i, sse=res[1], mse=res[2], aic=res[3],
                          p_val_1=res[4], p_val_2=res[5], false_disc=res[6]))
  }
  colnames(results) = c('k', 'SSE', 'MSE', 'AIC',
                        'p-wartość 1', 'p-wartść 2', 'false discoveries')
  rownames(results) = c(1:length(n))
  return(results)
}
results1 = tabelka(n)
```

Wyniki dla poszczególnych k-pierwszych kolumn macierzy wyglądają u nas następująco:

```
results1
```

```
##     k      SSE    MSE      AIC p-wartość 1 p-wartść 2 false discoveries
## 1   1 1071.528 21.431 2910.963       0.669         NA                 0
## 2   2 1146.344 22.927 2980.455       0.358         NA                 0
## 3   5 1411.466 28.229 3194.506       0.034      0.553                 0
## 4  10 1459.019 29.180 3237.642       0.216      0.486                 0
## 5  50 1402.303 28.046 3277.993       0.032      0.069                 2
## 6 100 1373.758 27.475 3357.427       0.086      0.990                 1
## 7 500  793.683 15.874 3608.806       0.426      0.293                12
## 8 950   73.757  1.475 2132.903       0.669      0.185                64
```

Kryteria AIC jest modyfikacją metody największej wiarogodności i sąskonstruowaną w taki sposób, by znaleźć balans pomiędzy dopasowaniem modelu do danych, a nadmierną złożonością modelu. W przypadku AIC istotne znaczenie ma statystyka SSE. Model, który należy wybrać powinien charakteryzować się jak najniższą wartością statystyki AIC. Na podstawie tych danych, model, który zostałby wybrany na podstawie wartości AIC to ten z największą ilością kolumn $k = 950$ z wartością AIC 2132.903.

(c)

Powtarzamy podpunkt (b), ale korzystając z największych (a nie pierwszych) oszacowanych współczynnikach regresji.

W tym celu modyfikujemy funkcję `compiut`, tak że porządkujemy nasze dane od największych do najmniejszych i następnie na ich podstawie tworzymy model.

```
compiut_c = function(n){
  Y =0
  p_val_2 = NA
  for(i in 1:n){
    Y = Y + X[i,]*beta[i]
  }
  Y = Y + rnorm(1000)
  data = data.frame(Y,X)

  lin_m = lm(Y~X[,1:n]-1, data=data)
```

```
  est_beta = lin_m$coefficients


  lin_m = lm(Y~X[, order(abs(est_beta),decreasing = TRUE)[1:n]]-1, data=data)
  summ_m = summary(lin_m)

  p_val_2 = NA
  sse = sum(lin_m$residuals^2)
  mse = sse/(1000-p)
  aic = AIC(lin_m)
  p_val_1 = summ_m$coefficients[1,4]
  if (i > 2){
    p_val_2 = summ_m$coefficients[2,4]
  }

  f_discovery = rep(0,n)
  for (i in c(1:n)){
    if (summ_m$coefficients[i,4] < 0.05){
      f_discovery[i] = 1
    }
  }
  false_disc = sum(f_discovery[5:n])
  if (n < 6){
    false_disc = 0
  }
  result = c(sse, mse, aic, p_val_1, p_val_2, false_disc)
  for (i in 1:length(result)){
    result[i] = round(result[i],3)
  }
  return(result)
}

tabelka_c = function(n){
  results <- data.frame(
    k=double(), sse=double(), mse=double(), aic=double(),
    pwartx1=double(), pwartx2=double(), false_disc=double())

  for (i in n) {

    res <- compiut_c(i)

    results <- rbind(
      results, data.frame(k=i, sse=res[1], mse=res[2], aic=res[3],
                          p_val_1=res[4], p_val_2=res[5], false_disc=res[6]))
  }
  colnames(results) = c('k', 'SSE', 'MSE', 'AIC',
                        'p-wartość 1', 'p-wartość 2', 'false discoveries')
  rownames(results) = c(1:length(n))
  return(results)
}

results2 = tabelka_c(n)
results2
```

```
##     k      SSE    MSE      AIC p-wartość 1 p-wartość 2 false discoveries
## 1   1 1075.545 21.511 2914.704       0.445          NA                 0
## 2   2 1154.796 23.096 2987.801       0.043          NA                 0
## 3   5 1383.580 27.672 3174.551       0.024       0.201                 0
## 4  10 1416.522 28.330 3208.081       0.189       0.206                 0
## 5  50 1344.260 26.885 3235.721       0.016       0.065                 0
## 6 100 1304.524 26.090 3305.715       0.012       0.023                 1
## 7 500  681.669 13.633 3456.666       0.002       0.011                17
## 8 950   63.779  1.276 1987.538       0.001       0.000                99
```

Tak jak można było się tego spodziewać, na podstawie wartości AIC modelem, który powinnniśmy wybrać jest znów ten z największą ilością kolumn $k = 950$ i wartością AIC 1987.538.