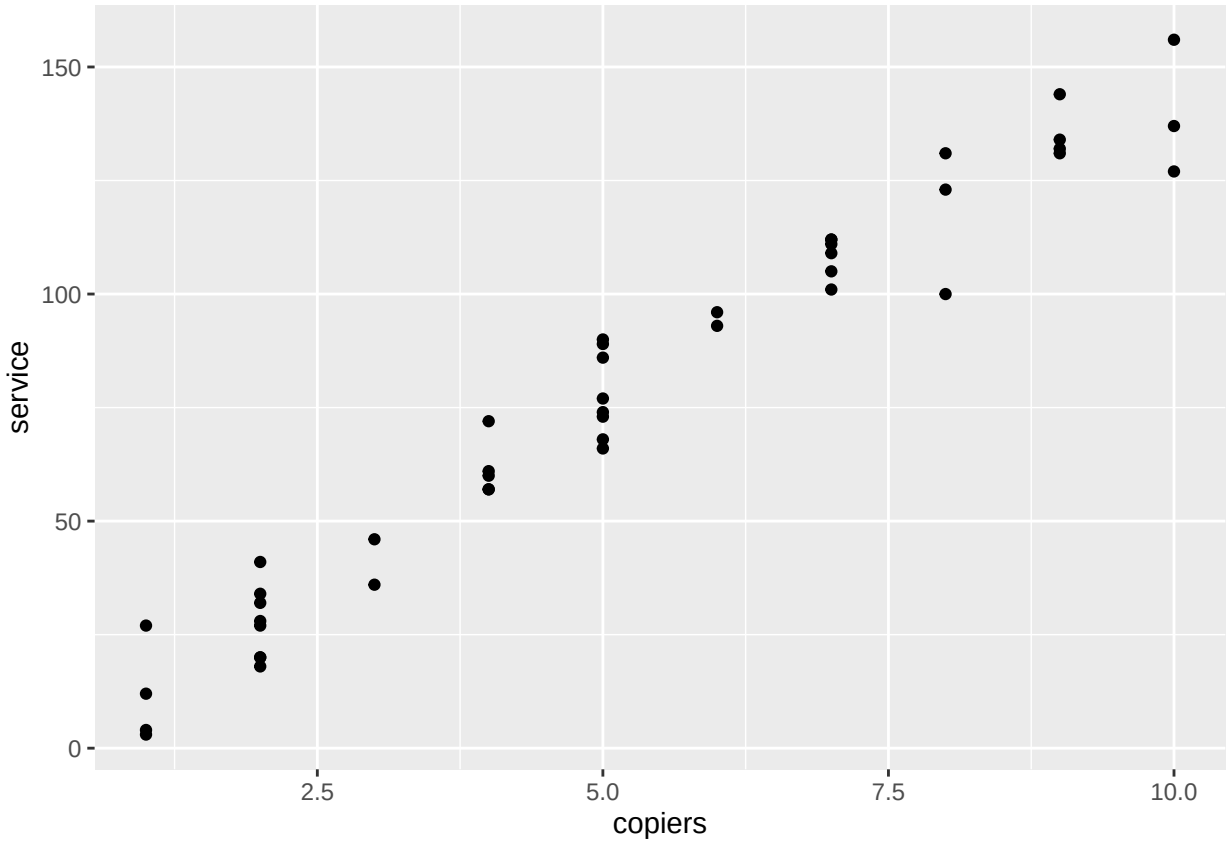
Raport 2 Modele liniowe

Magdalena Wawrzyniak

2022-11-28

Zadanie 1

W tym raporcie będziemy analizować dane z pliku `ch01pr20.txt`, który przedstawia czas serwisowania kopiarek. Aby łatwiej było zacząć analizować nasze dane obejrzmy je na wykresie. Wgrywamy dane do R przy użyciu komendy read.table, ustawiamy ziarno (ja wybrałam 17, bo nie pamiętałam na jakie się umawialiśmy na zajęciach), a następnie przy użyciu funkcji z pakietu `ggplot2` generujemy wykres, który widzimy poniżej.



Możemy zauważyć, że nasze dane wykazują pewną korelację. Na podstawie wykresu możemy stwierdzić, że dane wykazują zależność w przybliżeniu liniową. Jest to dobra podstawa do tego, aby rozważać model regresji liniowej prostej.

Zadanie 2

Rozważmy teoretyczny model regresji liniowej. W tym modelu zakładamy, że związek pomiędzy zmiennymi zależnymi Y (service) i odpowiadającymi im wartościami zmiennych niezależnych X (copiers) jest postaci:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i,$$

gdzie:

β_0 - intercept - wyraz wolny, parametr deterministyczny,

β_1 - slope - współczynnik kierunkowy, parametr deterministyczny,

ϵ_i - błąd pomiarowy, zmienna losowa z rozkładu $N(0, \sigma^2)$, i.i.d dla każdego $i \in 1..., n$

Znajomość parametrów β_0, β_1, σ^2 możemy generować wartości zmiennej Y_i dla dowolnych wartości X_i, co umożliwia przeprowadzanie symulacji i doświadczeń. My dysponujemy tylko zbiorem par postaci (X_i, Y_i), ale na ich podstawie możemy estymować parametry, na których nam zależy. Na wykładzie poznaliśmy dwie metody estymacji parametrów, tj. Metoda najmniejszych kwadratów oraz metoda największej wiarogodności. Za ich pomocą otrzymaliśmy estymatory postaci:

$$\hat{\beta_1} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

$$\hat{\beta_0} = \bar{Y} - \hat{\beta_1}\bar{X}$$

$$s^2 = \frac{1}{n-2}\sum_{i=1}^{n} Y_i - \hat{\beta_0} - \hat{\beta_1} X_i$$

gdzie s^2 jest nieobciążonym estymatorem wariancji błędów. Teoretyczne wyliczenie podanych powyżej estymatorów wygląda następująco (intercept, to wyraz wolny, a copiers odnosi się do współczynnika kierunkowego przy X).

Wyliczenie teoretyczne

Estymatory

```
slope = cov(X,Y)/var(X)
intercept = mean(Y) - slope*mean(X)
est = matrix(c(intercept, slope), nrow = 1, ncol = 2)
rownames(est) = " "
colnames(est) = c("intercept", "copiers")

n = nrow(copier_service)
df = n - 2
s_2 = (n-1)*var(Y - intercept - slope*X)
s_2 = s_2/df

est
```

```
##    intercept  copiers
##   -0.5801567 15.03525
```

```
s_2
```

```
## [1] 79.45063
```

Następną rzeczą jaką chcielibyśmy zrobić jest znalezienie przedziału ufności dla β_1 o współczynniku ufności 0.95, który wyznacza się na podstawie statystyki testowej T, która pochodzi z rozkładu studenta z $n-2$ stopniami swobody, danej wzorem:

$$T = \frac{\hat{\beta}_1 - \beta_1}{s(\hat{\beta}_1)},$$

gdzie

$$s^2(\hat{\beta}_1) = \frac{s^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}.$$

Wówczas przedział ufności dla parametru β_1 wyznaczamy następująco:

$$[\hat{\beta}_1 - t_c s(\hat{\beta}_1),\ \hat{\beta}_1 + t_c s(\hat{\beta}_1)],$$

tutaj t_c oznacza kwantyl rzędu $1-\frac{\alpha}{2}$ z rozkładu studenta z $n-2$ stopniami swobody. Dla naszego zagadnienia wyliczenia teoretyczne przedstawione są poniżej.

```
alfa = 0.05
t_c = qt(1 - alfa/2, df)
s_2_slope = s_2/((n-1)*var(X))
s_2_slope
```

```
## [1] 0.2333733
```

Wyliczenia teoretyczne

Przedzial ufnosci

```
lower_bound = slope - t_c*sqrt(s_2_slope)
upper_bound = slope + t_c*sqrt(s_2_slope)
conf_int = matrix(c(lower_bound, upper_bound), nrow = 1)
rownames(conf_int) = " "
colnames(conf_int) = c( "lower bound", "upper bound")
conf_int
```

```
##   lower bound upper bound
##      14.06101    16.00949
```

Na podstawie statystyki T możemy również testować czy β_1 jest różne od zera. Przeprowadzimy test istotności dla β_1. Testowane hipotezy:

$H_0:\ \beta_1 = 0,$

$H_1:\ \beta_1 \neq 0$

Statystyka testowa jest postaci:

$$T = \frac{\hat{\beta}_1 - 0}{s(\hat{\beta}_1)},$$

gdzie s jest jak wyżej. Będziemy odrzucali hipotezę zerową, gdy $|T| > t_c$, dla tego samego t_c jak wcześniej. Możemy również wyznaczyć p-wartość dla tego testu: $p = P(|z| > |T|)$, z ma rozkład $t(n-2)$. Poniżej przedstawione jest testowanie dla naszych danych.

Statystyka testowa

```
sqrt(s_2_slope)
```

```
## [1] 0.4830872
```

```
T_slope = slope / sqrt(s_2_slope)
T_slope
```

```
## [1] 31.12326
```

p-wartość

```
p_value = pt(-T_slope, df)*2
p_value
```

```
## [1] 4.009032e-31
```

```
T_slope > t_c
```

```
## [1] TRUE
```

Na podstawie naszych wyników odrzucamy hipotezę zerową i przyjmujemy hipotezę alternatywną, co oznacza, że na poziomie ufności 0.95 wartość współczynnika kierunkowego jest różna od zera.

Model z R

Wszystkie te wyniki można było uzyskać z wbudowanej funkcji z R, co jest dość wygodne, bo zajmuje dosłownie kilka linijek, co widać poniżej. Istnieje funkcja, która zwraca model regresji wyliczony dla naszych danych z potrzebnymi parametrami i innymi istotnymi dla nas informacjami. Parametry wyliczone przy pomocy wbudowanej funkcji wyznaczamy poniżej.

```
model <- lm(service ~ copiers, data = copier_service)
```

Estymatory

```
model$coefficient
```

```
## (Intercept)     copiers
##  -0.5801567  15.0352480
```

Widzimy, że estymatory zostały poprawnie wyliczone, bo niezależnie od tego czy użyliśmy wbudowanej funkcji z R czy liczyliśmy ręcznie, to doszliśmy do tych samych wyników.

Przedzial ufnosci

```
confint(model)
```

```
##                 2.5 %    97.5 %
## (Intercept) -6.234843  5.074529
## copiers     14.061010 16.009486
```

Wartość statystyki T

```
summary(model)$coefficients[2,3]
```

```
## [1] 31.12326
```

p-wartość

```
summary(model)$coefficients[2,4]
```

```
## [1] 4.009032e-31
```

W obu przypadkach wyniki są praktycznie takie same.

Zadanie 3

Teraz naszym celem jest podanie średniego czasu serwisu, jakiego możemy się spodziewać dla 11 maszyn. Jest to dość proste, gdy zastosujem funkcję `predict` dla naszego modelu i wygląda to następująco:

```
predict(model,
        newdata = data.frame(copiers = 11),
        interval = "confidence")
```

```
##        fit      lwr      upr
## 1 164.8076 158.4754 171.1397
```

Oczywiście to samo można liczyć samodzielnie, stosując odpowiednią teorię. To jak to zrobić w R pokazane jest poniżej. Pierwsze co robimy, to wyliczamy estymator wartości oczekiwanej dla konkretnego X_h, w tym zadaniu równego 11, gdzie:

$$\hat{\mu_h} = \hat{\beta}_0 + \hat{\beta}_1 X_h.$$

Następnie wyliczamy pierwiastek z nieobciążonego estymatora wariancji, gdzie estymator wariancji dla wartości oczekiwanej wyliczamy ze wzoru:

$$\sigma^2(\hat{\mu_h}) = \sigma^2\left(\frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}\right).$$

Kolejnym krokiem jest wyliczenie na podstawie statystyki testowej dla wartości oczekiwanej przedziałów ufności.

```
copiers_h = 11

mu_h = intercept + slope*copiers_h
s_mu = sqrt(s_2 * (1/n + (copiers_h - mean(X))^2 /((n-1)*var(X))))
# T_mu = mu_h/s_mu

lower_bound_mu = mu_h - t_c*s_mu
upper_bound_mu = mu_h + t_c*s_mu
conf_int_mu = matrix(c(mu_h, lower_bound_mu, upper_bound_mu), nrow = 1)
rownames(conf_int_mu) = " "
colnames(conf_int_mu) = c("fit", "lower bound", "upper bound")
conf_int_mu
```

```
##        fit lower bound upper bound
##   164.8076    158.4754    171.1397
```

Po porównaniu naszych wyników z funkcją z R widzimy, że wyszło nam to samo, czyli przedził $[158.4754,\ 171.1397]$.

Zadanie 4

W tym zadaniu mamy podać przewidywany rzeczywisty czas serwisu, jakiego można się spodziewać, gdyby obsługiwano 11 maszyn. Tak jak w poprzednim zadaniuemy wykorzysta funkcj `predikt`, co pokaemy poniżej.

```
predict(model,
        newdata = data.frame(copiers = 11),
        interval = "prediction")
```

```
##        fit      lwr     upr
## 1 164.8076 145.7491 183.866
```

Drugi sposób to wyliczanie tego krok po kroku i w zasadzie cała procedura będzie wyglądała jak poprzednio, z tą różnicą, że tym razem trzeba będzie wyliczyć wariancję błędu predykcji. Wzór na tę wariancję wygląda tak jak podano na dole i jest to suma wariancji Y_h oraz wariancji estymatora wartości oczekiwanej,

$$s^2(pred) = Var(Y_h - \hat{\mu_h}) = \sigma^2\left(1 + \frac{1}{n} + \frac{(X_h - \bar{X})^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}\right).$$

```
s_2_pred_mu = s_2*(1 + 1/n + (copiers_h - mean(X))^2/((n-1)*var(X)))

lower_bound_pred_mu = mu_h - t_c*sqrt(s_2_pred_mu)
upper_bound_pred_mu = mu_h + t_c*sqrt(s_2_pred_mu)
conf_int_pred_mu = matrix(c(mu_h, lower_bound_pred_mu, upper_bound_pred_mu), nrow = 1)
rownames(conf_int_pred_mu) = " "
colnames(conf_int_pred_mu) = c("fit", "lower bound", "upper bound")
conf_int_pred_mu
```

```
##        fit lower bound upper bound
##   164.8076    145.7491     183.866
```

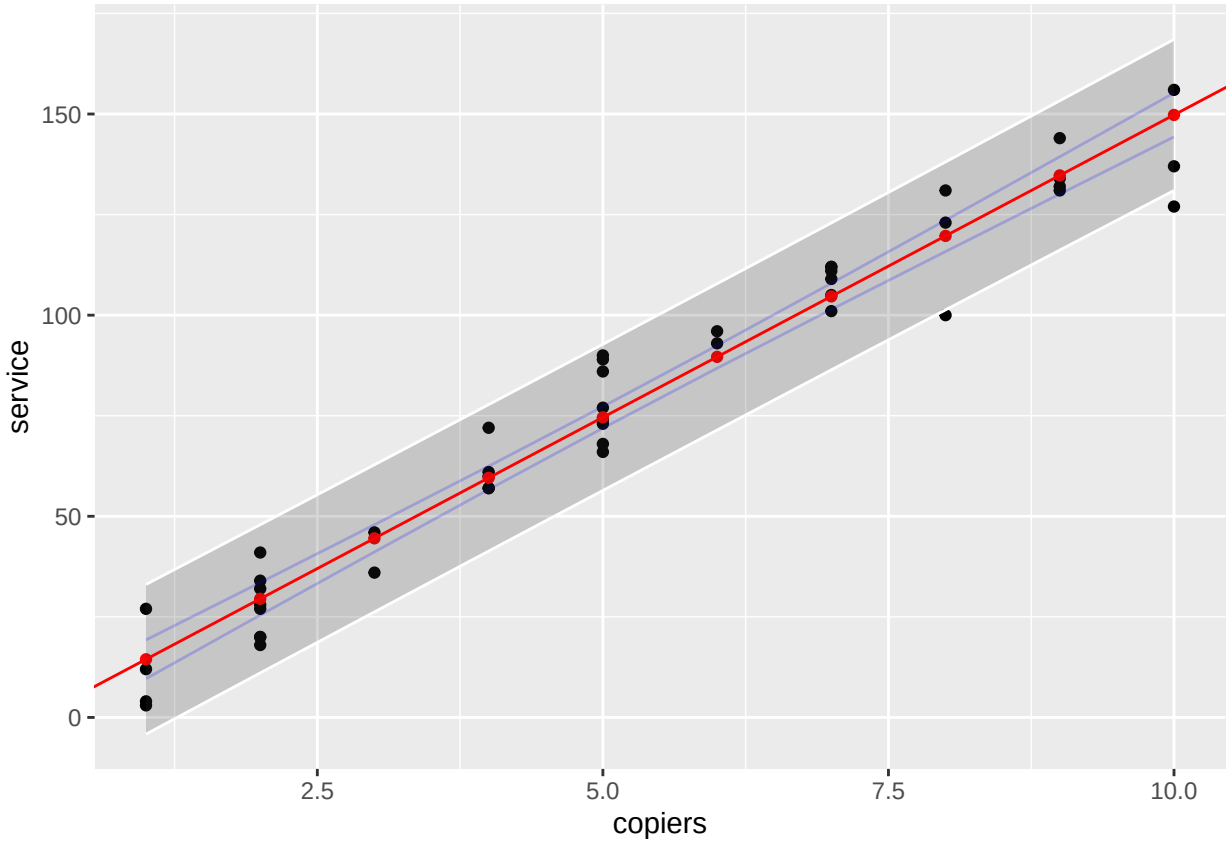
Tak jak można było się spodziewać przedział predykcji jest szerszy od przedziału z poprzedniego zadania. Jest to spowodowane tym, że w tym wypadku chcemy podać przedział, do któregoz prawdopodobieństwem 0.95 wpadłaby nowa wartość.

Zadanie 5

Dane z granicami predykcji 95% dla poszczególnych obserwacji zaprezentowane są poniżej na wykresie, na którym kolorem czrwonym zaznaczona jest prosta predykcji z przykładowymi przewidywanymi obesrwacjami, kolorem niebieskim zaznaczono przedział ufności, natomiast zacieniony obszer to przedział predykcyjny, na czarno zaznaczone są dane, na podstawie których tworzyliśmy model.

```
predictions <- data.frame(predict(model,
                                  newdata = data.frame(copiers = 1:10),
                                  interval = "prediction"),
                                  copiers = 1:10)
confidence <- data.frame(predict(model,
                                  newdata = data.frame(copiers = 1:10),
                                  interval = "confidence"),
                           copiers = 1:10)

ggplot() +
  geom_point(copier_service, mapping = aes(x = copiers, y = service)) +
  geom_point(predictions, mapping = aes(x = copiers, y = fit), col = "red") +
  geom_ribbon(predictions,
              mapping = aes(x = copiers, ymin = lwr, ymax = upr),
              alpha = 0.2, colour = "white") +
  geom_line(confidence,
              mapping = aes(x = copiers, y= lwr),
              alpha = 0.2, colour = "blue") +
  geom_line(confidence,
              mapping = aes(x = copiers, y= upr),
              alpha = 0.2,  colour = "blue") +
  geom_abline(slope = coef(model)[2], intercept = coef(model)[1], col = "red")
```



Zadanie 6

Szukamy mocy testu, czyli prawdopodobieństwa odrzucenia hipotezy zerowej, dla tego zadania $H_0 : \beta_1 = 0$, gdy prawdziwa jest hipoteza alternatywna, w tym zadaniu rzeczywiste $beta_1 = 1$. Przy tym założeniu statystyka T ma niecentralny rozkład studenta z 38 stopniami swobody i parametrem niecentralności $\delta = \frac{\beta_1}{\sigma\hat{\beta}_1}$, który wyliczymy poniżej przy pomocy R, wraz z mocą dla tego testu ($\pi(\beta_1 = 1)$).

```
# (a)
# moc testu dla beta1 = 1
n = 40
SSX = 1000
s2 = 120
s2_beta1 = s2 / SSX
df = n - 2
alfa = 0.05

delta_1 = 1 / sqrt(s2_beta1)
delta_1
```

```
## [1] 2.886751
```

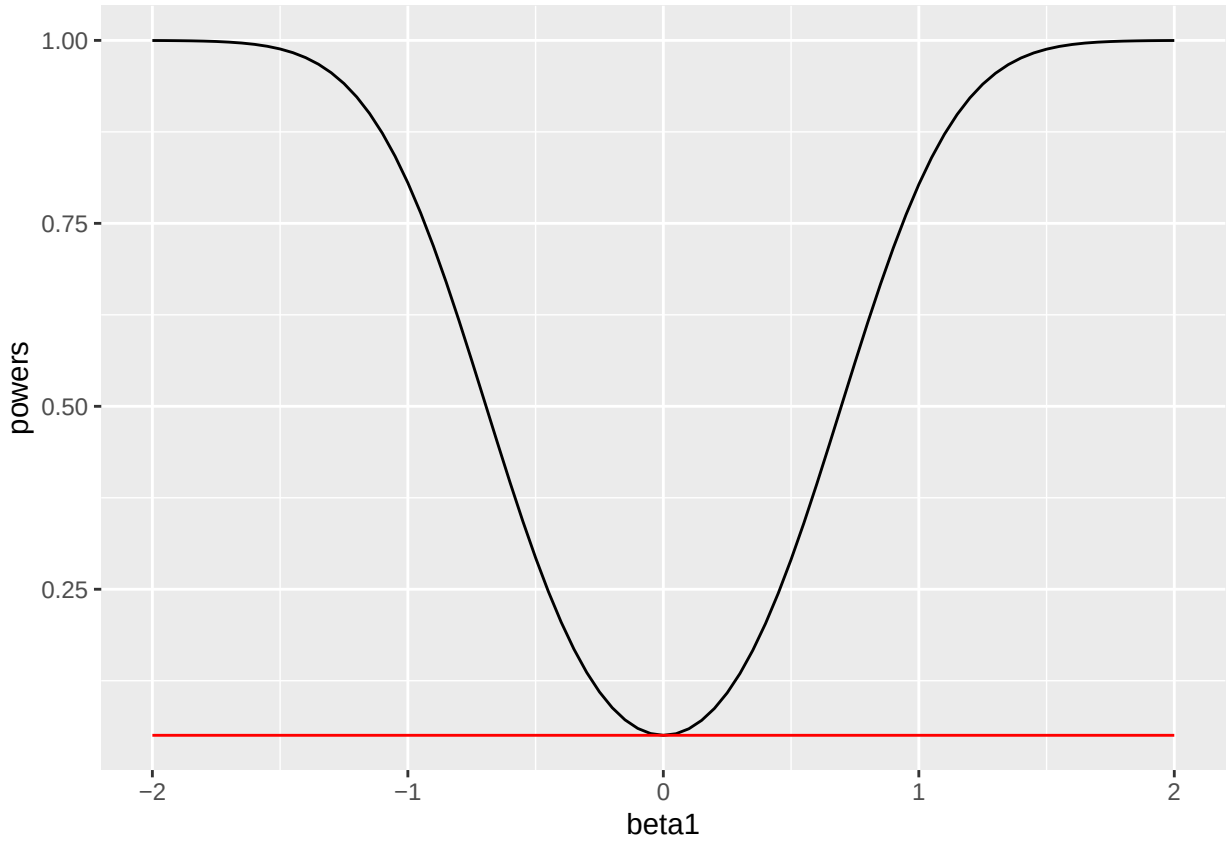
```
tc = qt(1 - alfa/2, df)
pi_1 = 1 - pt(tc, df, delta_1) + pt(-tc, df, delta_1)
pi_1
```

```
## [1] 0.8032105
```

W drugeij czeęści tego zadania chcemy narysować wykres funkcji mocy testu dla β_1 z przedziłu $[-2,\ 2]$.

```
# (b)

beta1 = seq(from=-2.0, to= 2.0, by= 0.05)
pi = 1 - pt(tc, df, beta1 / sqrt(s2_beta1)) + pt(-t_c, df, beta1 / sqrt(s2_beta1))
power_test = data.frame(beta1,pi)
colnames(power_test) = c("beta1", "powers")
ggplot() +
    geom_line(power_test, mapping = aes(x = beta1, y = powers)) +
    geom_line(data.frame(beta1, y = min(pi)), mapping = aes(x =     beta1, y),
              colour = "red")
```



Minimum krzywej wypada wtedy, gdy H0 jest prawdziwa i jest to równe poziomowi istotności testu.

Zadanie 7

Naszym zadaniem jest wygenerować wektor X o długości 200 z rozkładu wielowymiarowego normalnego $N(0, \frac{1}{200}I)$, a następnie 1000 wektorów Y z modelu przedstawiającego zleżność, że $Y = 5 + \beta_1 X + \epsilon$, przy podanych założeniach na β_1 i ϵ. Kolejną rzeczą jaką chcemy zrobić jest obliczenie prawdopodobieństwo odrzucenia hipotezy $H_0 : \beta_1 = 0$ na podstawie częstości odrzuceń występujących w naszej próbe, a następnie porównanie ich z teoretycznym prawdopodobieństwem błędu. Aby oszacować prawdopodobieństwo odrzucenia na podstawie naszej próby tworzymy funkcję o nazwie `empirical_result`, która zlicza ile razy odrzucamy hipotezę zerową, a następnie liczy średnią z tego wyniku.

```r
empirical_result <- function(beta1, epsilon){
  x = rnorm(200, 0, sqrt(1/200))
  res = numeric(1000)
  eps = matrix(epsilon, 200, 1000)
  for (i in 1:1000){
    y = 5 + beta1*x + eps[,i]
    reg = lm(y~x)
    res[i] = (summary(reg)$coefficients[2, 4] < 0.05)

  }
  return(mean(res))
}
```

Funkcje `theoretical_result` zlicza moce testów i liczy z nich średnią z wszystkich prób.

```r
theoretical_result <- function(beta1){
  x <- rnorm(200, 0, sqrt(1/200))
  powers <- numeric(1000)
  eps <- matrix(rnorm(200*1000), 200, 1000)
  for(i in 1:1000){
    y <- 5 + beta1*x + eps[,i]
    reg <- lm(y~x)
    s <- sd(reg$residuals)*sqrt((200-1)/(200-2))/(var(x)*199)
    delta <- beta1/s
    powers[i] <- 1 - pt(qt(1-0.05/2, 200 - 2), 200-2, delta) + pt(-qt(1-0.05/2, 200-2), 200-2, delta)
  }
  return(mean(powers))
}
```

Wyniki dla (a)-(b) - prawdopodobińśtwo popełnienia błędu I rodzaju

Wyniki na podstawie empirycznych doświadczeń:

(a)

```r
empirical_result(0, rnorm(200*1000, 0, 1))
```

```
## [1] 0.057
```

(b)

```r
empirical_result(0, rexp(200*100, 1))             #estimator of probability of type I error
```

```
## [1] 0.02
```

Teoretyczne prawdopodobieństwo dla tego problemu:

```r
theoretical_result(0)
```

```
## [1] 0.05
```

Prawdopodobieństwo popełnienia błędu pierwszego rodzaju w podpunktach (a)-(b) teoretycznie wynosi 0.05, natomiast z naszych doświadczeń wynika, że został on popełniony rzadziej niz byśmy się tego spodziewali.

Wyniki dla (c)-(d) - moc testu

Wyniki na podstawie empirycznych doświadczeń: (c)

```
1 - empirical_result(1.5, rnorm(200*1000, 0, 1))
```

```
## [1] 0.733
```

(d)

```
1 - empirical_result(1.5, rexp(200*1000, 1))
```

```
## [1] 0.619
```

Teoretyczne prawdopodobieństwo dla tego problemu:

```
1 - theoretical_result(1.5) #theoretical value of test power
```

```
## [1] 0.6949466
```

Moc testu liczona dla podpunktów (c)-(d) teoretycznie powinna być nieco wyższa, niż wyszło nam z doświadczeń, ale mimo to wyniki są do siebie zbliżone. W tym przypadku wyszło nam, że rzadziej popełniamy błąd I rodzaju, ale trochę częściej przytrafiał się nam błędy II rodzaju. Wynika to z tego, że rozkład statystyki przy H0 i H1 mało się różnił.