Lab2 - Estymacja parametrów

..

..

Zadanie 1. *Przypomnienie arytmetyki wektorowej.*

a) Dany jest wektor liczb zmiennoprzecinkowych `x`. Wyjaśnij, co oblicza poniższy kod:

```
v <- mean((x-mean(x))^2) # Ten kod oblicza wariancję wektora x, czyli średnią kwadratów odchyleń wartoś
```

b) Utwórz zmienną całkowitoliczbową `n` z wybraną przez siebie wartością od 100 do 5000. Wylosuj `n` obserwacji z rozkładu normalnego z wybranymi przez siebie parametrami μ i σ, korzystając z funkcji `rnorm`. Następnie, wykorzystując jedynie funkcję `sum`, oblicz nieobciążony estymator wariancji oraz estymatory największej wiarygodności wariancji i odchylenia standardowego. Twój kod powinien zająć co najwyżej 7 linijek. Porównaj swoje wyniki z funkcjami `var` oraz `sd`. Jaki estymator wariancji jest zaimplementowany domyślnie w R?

```
n <- 2000
x <- rnorm(n, 0,1)

unbiased_var <- sum((x - mean(x))^2) / (n - 1) # estymator nieobciążony (mianownik n-1) - uwzględnia st
mle_var <- sum((x - mean(x))^2) / n # estymator największej wiarygodności (obciążony)
mle_sd <- sqrt(mle_var)

cat("Nieobciążony estymator wariancji:", unbiased_var, "\n")
```

```
## Nieobciążony estymator wariancji: 1.014798
```

```
cat("Estymator największej wiarygodności wariancji:", mle_var, "\n")
```

```
## Estymator największej wiarygodności wariancji: 1.014291
```

```
cat("Estymator największej wiarygodności odchylenia standardowego:", mle_sd, "\n")
```

```
## Estymator największej wiarygodności odchylenia standardowego: 1.00712
```

```
cat("Wartość wariancji obliczona za pomocą funkcji var():", var(x), "\n")
```

```
## Wartość wariancji obliczona za pomocą funkcji var(): 1.014798
```

```
cat("Wartość odchylenia standardowego obliczona za pomocą funkcji sd():", sd(x), "\n")
```

```
## Wartość odchylenia standardowego obliczona za pomocą funkcji sd(): 1.007372
```

W R domyślnie stosowany jest nieobciążony estymator wariancji jako `var(x)`.

Macierze

W następnym zadaniu skorzystamy z nowego typu danych, czyli **macierzy**. Mając wektor `x`, możemy przekształcić go w macierz o `n` wierszach i `m` kolumnach komendą `matrix(x, nrow=n, ncol=m)`. Wypełnianie macierzy domyślnie odbywa się kolumna po kolumnie. Możemy również wypełniać ją wiersz po wierszu podając argument `byrow=TRUE`. Jeśli wektor ma mniej niż `nm` elementów, to nastąpi jego *recykling* - po wyczerpaniu wartości z `x` wracamy do jego początku i wypełniamy macierz dalej. Jeśli podamy wyłącznie argument `nrow` (lub `ncol`), liczba kolumn (wierszy) zostanie dobrana automatycznie. Przetestuj działanie komendy `matrix`, wpisując w konsolę `matrix(1:9, ncol=3)` oraz `matrix(1:6, ncol=3)`.

```r
matrix(1:9, ncol=3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
matrix(1:6, ncol=3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Mając macierz `M`, możemy przyłożyć dowolną funkcję kolejno do wszystkich wierszy lub kolejno do wszystkich kolumn. Pozwala to na przykład w prosty sposób otrzymać średnią z każdej kolumny. W tym celu wykorzystujemy funkcję `apply(X, n, f)`, gdzie `X` to macierz wejściowa, `F` to funkcja, a `n` oznacza czy chcemy przyłożyć `F` do wierszy (`n=1`) czy kolumn (`n=2`) macierzy `X`. Przetestuj działanie komendy `apply`, wpisując w konsolę `M <- matrix(1:9, ncol=3)`, a następnie `apply(M, 2, mean)` oraz `apply(M, 1, sum)`.

```r
M <- matrix(1:9, ncol=3)
M
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
apply(M, 2, mean) # 2 - kolumnami
```

```
## [1] 2 5 8
```

```r
apply(M, 1, sum)  # 1 - wierszami
```

```
## [1] 12 15 18
```

Macierze i komenda `apply` będą jednymi z najczęściej wykorzystywanych przez nas narzędzi pakietu R. Warto zatem dobrze zrozumieć ich działanie i wiedzieć, w jakich sytuacjach ich używać.

Zadanie 2. *Trzy estymatory wariancji*

Korzystając z funkcji `rnorm`, wylosuj 5000 obserwacji z rozkładu normalnego o średniej 0 i wybranym przez siebie odchyleniu standardowym σ. Przekształć otrzymany wektor w macierz o wymiarach 10 x 500. W każdej kolumnie wyestymuj wariancję korzystając z funkcji `apply` oraz `var`. Funkcja `var` zwraca nieobciążony estymator wariancji:

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

Dodatkowo, w każdej kolumnie wyestymuj wariancję korzystając z estymatorów

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

oraz

$$\hat{S}_2^2 = \frac{1}{n+1}\sum_{i=1}^{n}(X_i - \bar{X})^2.$$

Wskazówka. Mając wartości estymatora $\hat{S}^2$ w wektorze `S`, estymator $\hat{S}_1^2$ możesz łatwo otrzymać jako `S1 <- S*9/10`.

Wyestymuj i porównaj obciążenia i odchylenia standardowe wszystkich trzech estymatorów wariancji. Następnie wyestymuj i porównaj porównaj błędy średniokwadratowe estymatorów:

$$RMSE = \sqrt{\mathbb{E}(\hat{\theta} - \theta)^2},$$

$$R\hat{M}SE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{\theta}_i - \theta)^2}.$$

Błąd średniokwadratowy mówi nam, jak daleko, średnio rzecz biorąc, wartość wyestymowana parametru θ znajduje się od jego wartości prawdziwej. Estymujemy go korzystając z prawdziwej wartości parametru oraz n *realizacji* estymatora, $\hat{\theta}_i$, czyli po prostu tego co wyliczamy z macierzy. W naszym przypadku parametr θ to wariancja σ^2 naszego rozkładu normalnego.

Na podstawie przeprowadzonych symulacji, który estymator zdaje się mieć najmniejsze obciążenie? A który najmniejszy błąd średniokwadratowy?

```
sigma <- 10
x <- rnorm(5000, 0, sigma)
M <- matrix(x, ncol=500)
var1 <- apply(M, 2, var)
var2 <- var1*9/10
var3 <- var1*11/10

# porównanie wyników estymatorów
bias <- c(mean(var1-sigma^2), mean(var2-sigma^2), mean(var3-sigma^2)) # porównanie obciążeń
sd_est <- c(sd(var1), sd(var2), sd(var3)) # porównanie std
rmse <- c(sqrt(mean((var1-sigma^2)^2)), sqrt(mean((var2-sigma^2)^2)), sqrt(mean((var3-sigma^2)^2))) # b

# wyniki
print(bias)
```

```
## [1]  1.996541 -8.203113 12.196195
```

```
print(sd_est)
```

```
## [1] 47.35528 42.61975 52.09081
```

```
print(rmse)
```

```
## [1] 47.35001 43.36014 53.44879
```

Wyniki pokazują, że estymator nieobciążony var zwracał mniejsze obciążenie niż oba estymatory obciążone. Odchylenie standardowe estymatora obciążonego o mianowniku $n + 1$ miał nieco mniejsze odchylenie niż estymator z mianownikiem n. Pod względem błędu średniokwadratowego, estymator nieobciążony var był najdokładniejszy.

Funkcje

W następnym zadaniu zbadamy obciążenie estymatora odchylenia standardowego w zależności od liczebności próby, z jakiej estymujemy odchylenie standardowe. W tym celu wykorzystamy kolejne ważne narzędzie pakietu R, czyli **funkcje**. Spotkaliśmy się już z gotowymi funkcjami w R, na przykład `var`. Własne funkcje natomiast piszemy w następujący sposób:

```r
S2var <- function(X){
  # Funkcja obliczajaca estymator wariancji S2 na podstawie wektora X.
  m <- mean(X)
  v <- sum((x-m)^2)/(length(X)+1)  # funkcja length(X) zwraca długość wektora X
  return(v)
}
```

Powyższy kawałek kodu utworzy funkcję o nazwie `S2var`, która przyjmuje wektor `X` i zwraca wartość estymatora wariancji $\hat{S}_2^2$. Zwróćmy uwagę, że w R przypisujemy funkcję na zmienną, tak samo jak dowolny inny typ danych.

Bardzo odradzam pisanie własnych funkcji w konsoli. Na ogół są to już nieco dłuższe kawałki kodu i zbyt łatwo popełnić w nich błąd. Funkcje należy pisać albo w skryptach, albo w notatnikach Rmarkdown. Wyjątkiem wobec tej reguły jest pisanie funkcji bezpośrednio w `apply`, na przykład:

```r
M <- matrix(1:9, ncol=3)
M
apply(M, 2, function(x) x^x[1])
```

Powyższy kawałek kodu utworzy macierz M, wypisze ją w konsoli, a następnie obliczy macierz, w której każda kolumna zostanie podniesiona do potęgi o wykładniku równym pierwszemu elementowi tej kolumny. Czyli kolumna druga zostanie podniesiona do potęgi czwartej, a kolumna trzecia do potęgi siódmej.

Przyda się nam również funkcja `sapply`. Jest to po prostu uproszczone `apply`, które działa nie na macierzach, a na wektorach. Na przykład, `sapply(X, function(x) x^2)` podniesie każdy element wektora `X` do kwadratu, czyli zwróci to samo, co napisanie `X^2`.

Zadanie 3. *Estymator odchylenia standardowego.* Domyślny estymator odchylenia standardowego w R to pierwiastek z nieobciążonego estymatora wariancji. Czy taki estymator jest obciążony? Dlaczego? Czy średnio rzecz biorąc zwraca wartości niższe, czy wyższe od prawdziwych?

Zbadamy empirycznie zależność obciążenia od liczebności próby z której estymujemy odchylenie. Celem tego zadania jest utworzenie wykresu punktowego, który przedstawi wartość estymowanego odchylenia standardowego w zależności od liczebności próby.

Napisz funkcję o nazwie `sample_sd`, która przyjmie dwa argumenty, `N` oraz `n`, wylosuje `N` prób rozmiaru `n` z rozkładu normalnego (o wybranym przez Ciebie prawdziwym odchyleniu standardowym) i zwróci `N` estymowanych odchyleń standardowych. Wykorzystaj funkcje `matrix` i `apply` tak jak w poprzednim zadaniu. Ciało funkcji powinno mieć co najwyżej 4 linijki kodu.

```r
sample_sd <- function(N, n) {
  x <- rnorm(N*n, 0, 1)
  M <- matrix(x, nrow=n)
  return (apply(M, 2, sd))
}
```

Następnie utwórz wektor `n <- 2:100`. Korzystając ze swojej funkcji, dla każdej wartości z wektora `n` otrzymaj 100 estymowanych odchyleń standardowych. Właśnie tutaj przyda Ci się funkcja `sapply`, którą możesz wykorzystać w następujący sposób: `sapply(n, sample_sd, N=100)`. Wszystkie argumenty znajdujące się za nazwą funkcji wywoływanej w `sapply` zostaną przekazane tejże funkcji, tak jak w tym przypadku argument `N`.

```r
n <- 2:100
SD_mat <- sapply(n, sample_sd, N=100)
```

Korzystając w ten sposób z funkcji `sapply`, otrzymasz macierz, która w każdej kolumnie będzie zawierała po sto odchyleń standardowych (funkcja `sapply` składa otrzymane wyniki kolumna do kolumny). Żeby otrzymać wykres będący celem tego zadania, należy taką macierz przekształcić do ramki danych o dwóch kolumnach. Jedna z nich musi zawierać estymator, a druga - liczebność próby.

Żeby otrzymać pierwszą kolumnę, musimy „spłaszczyć” naszą macierz. Najprostszy sposób żeby spłaszczyć macierz `M` to wywołać komendę `c(M)`, która połączy kolejne kolumny jedna za drugą (dla przypomnienia, funkcja `c` służy do łączenia wektorów). W ten sposób na pierwszych stu współrzędnych otrzymamy odchylenia estymowane z 2 obserwacji, na następnych stu z 3 obserwacji itd.

Wektor zawierający liczebności próby odpowiadajace kolejnym obserwacjom otrzymamy komendą `rep(n, each=100)`. Funkcja `rep` służy do powtarzania wartości z pewnego wektora. Wywołanie `rep(1:3, 3)` powtórzy trzykrotnie cały wektor `1:3`, a wywołanie `rep(1:3, each=3)` powtórzy trzykrotnie każdą z wartości wektora `1:3`. Porównaj oba wyniki, wywołując te dwie komendy w konsoli.

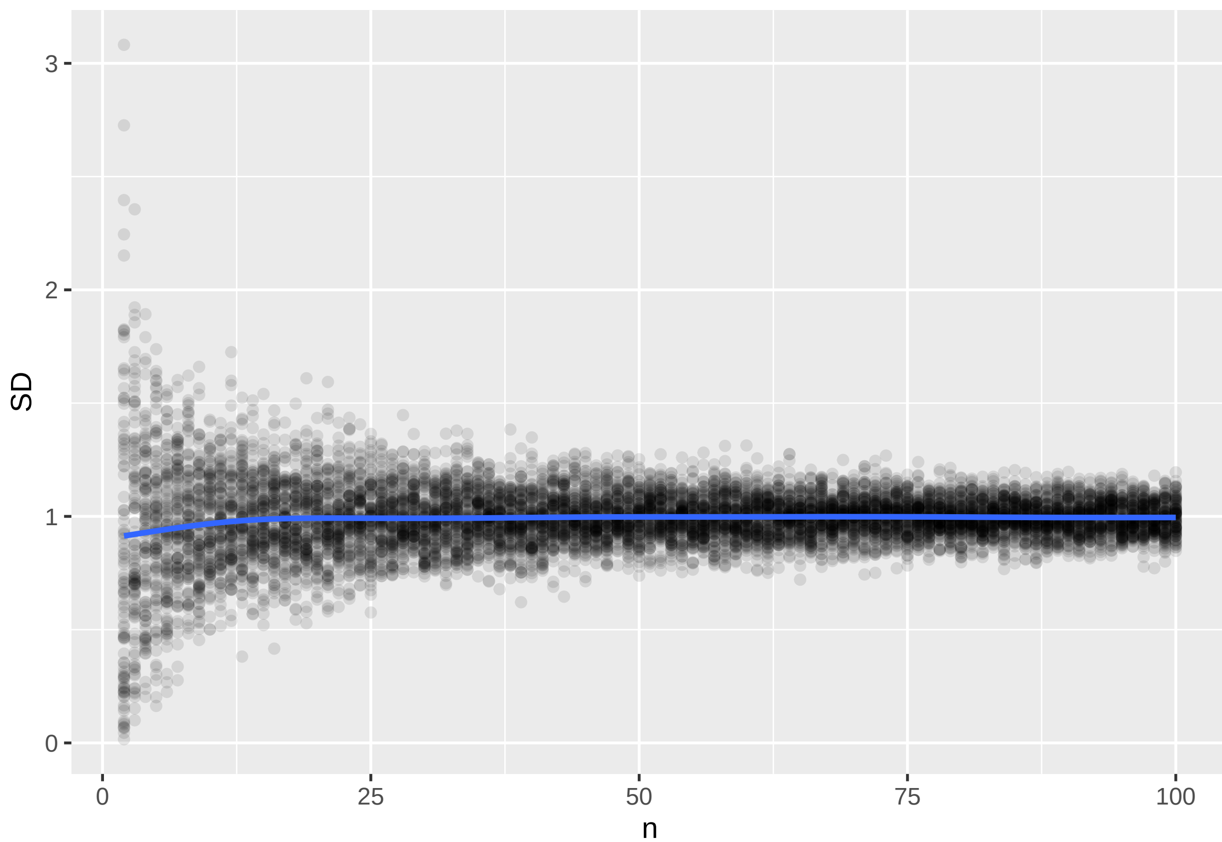
Na koniec, utwórz wykres punktowy obrazujący zależność estymowanego odchylenia od liczności próby. Zaraz za estetykami dodaj argument `alpha=0.1`, aby punkty były nieco przezroczyste. Wykres stanie się w ten sposób bardziej czytelny. Dodatkowo, nałóż na wykres warstwę `geom_smooth(aes(x=n, y=SD))`, która wykreśli krzywą obrazującą (w pewnym sensie) typowe wartości na wykresie. Na tym przedmiocie niestety nie mamy czasu aby dokładnie omówić w jaki sposób wyliczana jest ta krzywa.

```
# spłaszczamy macierz i tworzymy wektor liczebności prób
SD <- c(SD_mat)
n_repeated <- rep(n, each=100)

# tworzymy ramkę danych
data <- data.frame(n=n_repeated, SD=SD)

# wykres
ggplot(data, aes(x=n, y=SD)) +
  geom_point(alpha=0.1) +
  geom_smooth(aes(x=n, y=SD), se=FALSE)
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



Jakie wnioski możesz wyciągnąć z otrzymanego wykresu?

Na wykresie widać, że estymowane odchylenie standardowe maleje wraz ze wzrostem liczności próby. Im większa liczność próby, tym bardziej dokładna jest szacowana wartość odchylenia standardowego.

Zadanie 4. Załaduj dane `iris`. Wybierz dane dotyczące gatunku *setosa*. W tym celu najlepiej skorzystać z jeszcze jednego typu indeksowania, czyli *indeksowania logicznego* (a jakie dwa inne typy poznaliśmy na poprzednich zajęciach?). Indeksowanie logiczne wygląda następująco: `setosa_data <- iris[iris$Species == 'setosa', ]`. To, co tu się dzieje, to po kolei:

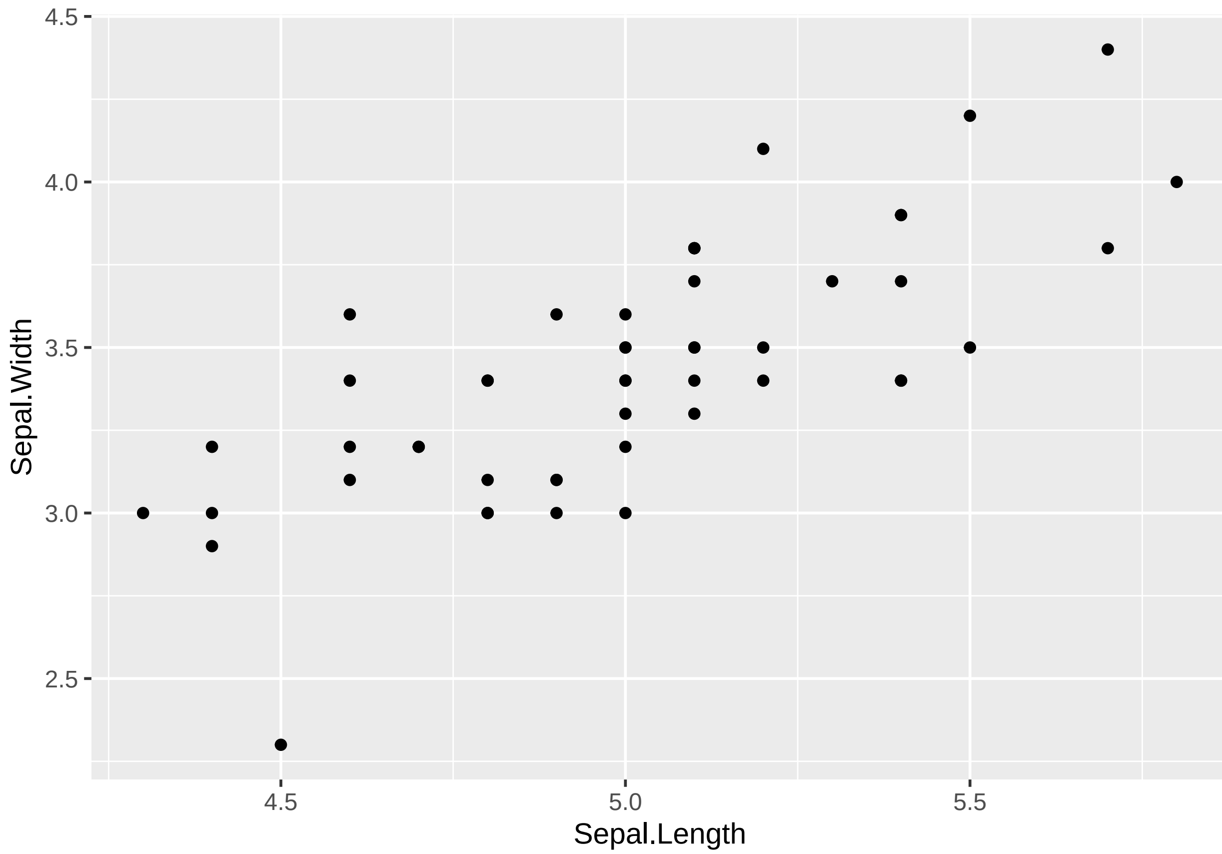
- Komenda `iris$Species` wybiera kolumnę o nazwie `Species`.
- Komenda `iris$Species == 'setosa'` porównuje wartości w tej kolumnie z napisem 'setosa' i zwraca wektor logiczny. Jeżeli chcemy, to możemy wynik tej komendy przypisać na nową zmienną: `is_this_setosa <- iris$Species == 'setosa'`.
- Komenda `iris[iris$Species == 'setosa', ]` wybiera wszystkie kolumny tabeli `iris` oraz te wiersze, w których w wektorze logicznym z poprzedniego punktu znajduje się wartość `TRUE`.

Po wybraniu danych dotyczących gatunku *setosa*, oblicz średnią oraz odchylenie standardowe zmiennych `Sepal.Width` oraz `Sepal.Length`. Następnie oblicz korelację tych zmiennych korzystając z funkcji `cor`. Porównaj ją z korelacją zmiennych `Petal.Length` oraz `Sepal.Length`.

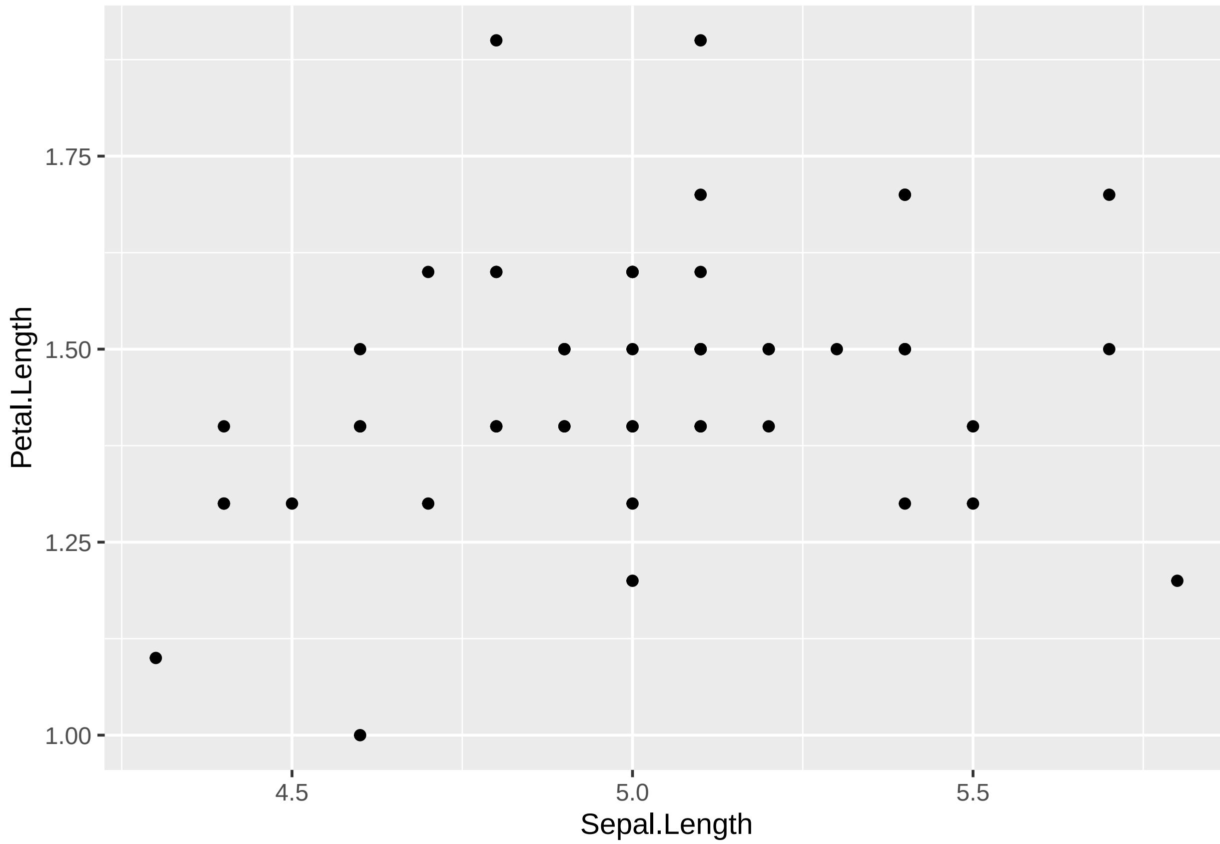
Korelacja zmiennych losowych mierzy, na ile silna jest zależność liniowa pomiędzy tymi zmiennymi. Korelacja pomiędzy zmiennymi X, Y jest równa ± 1 wtedy i tylko wtedy, gdy $X = aY + b$ dla pewnych liczb rzeczywistych a, b. Co możesz wywnioskować o zależności pomiędzy zmiennymi `Sepal.Width` a `Sepal.Length` na podstawie ich korelacji? A pomiędzy `Petal.Length` a `Sepal.Length`? Korzystając z biblioteki `ggplot2`, utwórz wykresy punktowe obrazujące zależność pomiędzy `Sepal.Width` a `Sepal.Length` oraz pomiędzy `Petal.Length` a `Sepal.Length` i sprawdź, czy Twoje wnioski były poprawne.

```
data(iris)
setosa_data <- iris[iris$Species == 'setosa', ]
mean_width <- mean(setosa_data$Sepal.Width)
std_width <- sd(setosa_data$Sepal.Width)
mean_length <- mean(setosa_data$Sepal.Length)
std_length <- sd(setosa_data$Sepal.Length)
cor_sepal <- cor(setosa_data$Sepal.Length, setosa_data$Sepal.Width)
cor_petal <- cor(setosa_data$Petal.Length, setosa_data$Sepal.Length)

ggplot(setosa_data, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point()
```




```
ggplot(setosa_data, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point()
```



Zadania dodatkowe.

Zadanie 1. Rozegraj jedną rundę w Guess the Correlation. Jeśli zamierzasz wykonać to zadanie w laboratorium, wyłącz głośniki.

Zadanie 2. Wyestymuj pole koła o promieniu 1 metodą Monte Carlo. W tym celu:

- Wylosuj n punktów z kwadratu [0, 1] x [0, 1]. Ustaw je w macierz o wymiarach $2 \times n$.
- Napisz funkcję, która przyjmie punkt (wektor długości 2) i sprawdzi, czy należy on do koła o promieniu 1.
- Wykorzystaj tę funkcję, funkcję `apply` oraz funkcję `mean` do sprawdzenia, jaka proporcja wylosowanych punktów należy do koła.
- Wykorzystaj obliczoną proporcję do estymacji pola koła.
- Porównaj wyniki dla różnych wartości `n`.

Wskazówka. Całe to zadanie można zrealizować w 8 linijkach kodu.

Zadanie 3.. Zbadaj obciążenie estymatora z poprzedniego zadania.