Projekt_koncowy_pd_4657

Małgorzata_Bujak

2025-04-19

1. Importowanie i wstępna inspekcja danych

- Wczytaj dane wykorzystując odpowiednie funkcje importu

```
# Wczytanie biblioteki
library(readr)

# Import danych z pliku i zapisanie do zmiennej dane
dane <- read_tsv("C:\\Users\\magor\\OneDrive\\Desktop\\2025_PJATK_Programowanie_R\\Projekt_koncowy\\GSE
```

- Sprawdź i pokaż strukturę danych

```
# Sprawdzenie struktury danych
struktura_danych <- str(dane)
```

- Określ wymiar danych (liczba wierszy, kolumn)

```
# Wymiary data frame (odpowiednio wiersze, kolumny)
dim(dane)
```

- Wyświetl pierwsze 10 wierszy

```
# Pierwsze 10 wierszy
head(dane, 10)
```

- Wyświetl ostatnie 5 wierszy

```
# Ostatnie 5 wierszy
tail(dane, 5)
```

2. Czyszczenie i przekształcanie danych

- Zidentyfikuj brakujące wartości (NA) - nawet jeśli w wybranym zbiorze danych nie ma brakujących wartości, to usuń przynajmniej 3 wartości z dowolnie wybranych przez siebie pozycji

```r
# Sprawdzenie, czy są brakujące dane wszystkich typów
is.na(dane)

# Zliczenie brakujących danych w kolumnie
sum(is.na(dane))

# Usunięcie wierszy z brakującymi danymi
dane_kompletne <- na.omit(dane)

# Wyświetlenie wyniku
print(dane_kompletne)
```

- Przekształć chodź jeden typ danych w inny np. w faktory, daty, zmienne numeryczne itp.

```r
# Zamiana wartości numerycznej na faktor
dane$p_value_jako_faktor <- cut(dane$pvalue,
                                breaks = c(-Inf, 0.05, Inf),
                                labels = c("istotna", "nieistotna"),
                                right = TRUE)

# Zamiana kolumn liczbowych na znakowe
dane$baseMean_znaki <- as.character(dane$baseMean)

# Sprawdzenie typu danych w kolumnach
class(dane$p_value_jako_faktor)  # Powinno zwrócić "factor"
class(dane$baseMean_znaki)       # Powinno zwrócić "character"
```

- Zmień nazwę kolumn lub wierszy na zapis dużymi literami

```r
# Zmiana wszystkich nazw kolumn na wielkie litery
colnames(dane) <- toupper(colnames(dane))

# Sprawdzenie nowych nazw kolumn
print(colnames(dane))
```

- Posortuj dane według wybranej kolumny malejąco

```r
# Sortownie malejąco po kolumnie log2foldChange
dane_posortowane <- dane[order(dane$LOG2FOLDCHANGE, decreasing = TRUE), ]

# Sprawdzenie pierwszych kilku wierszy posortowanych danych
head(dane_posortowane)
```

- Wyfiltruj i zapisz do nowej zmiennej obserwacje według określonego kryterium np. numerycznego $>$ 20 itp.

```r
# Przefiltorwanie danych, aby odrzucić geny, których zmiana ekspresji jest niewiarygodna
dane_filtorwane <- dane[dane$LFCSE > 1.00, ]

# Sprawdzenie pierwszych kilku wierszy
head(dane_filtorwane)
```

3. Imputacja danych

- Zastąp wartość NA wartością średnią dla danej kolumny

```r
# Sprawdzenie, w której kolumnie mam NA
colSums(is.na(dane))

# Zastapienie NA wartością średnią dla kolumny padj
dane$PADJ <- ifelse(is.na(dane$PADJ), mean(dane$PADJ, na.rm = TRUE), dane$PADJ)

# Sprawdzenie, czy kolumna PADJ zawiera jakiekolwiek NA po zamianie
sum(is.na(dane$PADJ))
```

4. Analiza eksploracyjna danych

- Na ramce danych wykorzystaj przynajmniej 4 różne funkcje z pakietu dplyr służące np. do: filtrowania, wybierania kolumn i ich modyfikowania, agregowania, sortowania, grupowania itp.

```r
# Wczytanie pakietu dplyr
library(dplyr)

# Filtrowanie danych, wybieramy wiersze, gdzie pvalue < 0.05
dane_filtrowane <- dane %>%
  filter(PVALUE < 0.05)

# Wyświetlenie pierwszych kilku wierszy po filtrze
head(dane_filtrowane)
```

5. Statystyka opisowa

- Oblicz podstawowe statystyki opisowe dla zmiennych numerycznych: minimum, maksimum, średnia, mediana, moda, odchylenie standardowe, kwartyle

```r
# Funkcja summary() poda dla wszytkich kolumn liczbowych: minimum, I kwartyl, mediane, średnią, III kwa
summary(dane)

# Obliczenie oddzielnymi funkcjami
dane_num <- dane[, sapply(dane, is.numeric)]

moda <- function(x) {
  names(sort(table(x), decreasing = TRUE))[1]
}

statystyki <- data.frame(
  zmienna = names(dane_num),
  minimum = sapply(dane_num, min, na.rm = TRUE),
  maksimum = sapply(dane_num, max, na.rm = TRUE),
  srednia = sapply(dane_num, mean, na.rm = TRUE),
  mediana = sapply(dane_num, median, na.rm = TRUE),
  odch_std = sapply(dane_num, sd, na.rm = TRUE),
  Q1 = sapply(dane_num, quantile, probs = 0.25, na.rm = TRUE),
```

```
  Q3 = sapply(dane_num, quantile, probs = 0.75, na.rm = TRUE),
  moda = sapply(dane_num, moda)
)

print(statystyki)
```

- Przygotuj tabele częstości dla zmiennych kategorycznych, jeżeli ich nie masz to utwórz takie dane

```
# Tworzenie kolumny ze zmienną kategoryczną
dane$Płeć <- rep(c("kobieta", "mężczyzna"), length.out = nrow(dane))

# Tabela częstości
tabela_czestosci <- table(dane$Płeć)

# Wyświetlenie wyników
print(tabela_czestosci)
```

6. Testowanie statystyczne

- Wykonaj dowolny test parametryczny lub nieparametryczny na wybranych danych i zinterpretuj jego wynik

```
# Test parametryczny, ANOVA jednoczynnikowa: porównanie poziomów ekspresji dla płci
wynik_ANOVA <- aov(LOG2FOLDCHANGE ~ Płeć, data = dane)

# Wyświetlenie wyniku testu ANOVA
print(wynik_ANOVA)

# Wyświetlenie podsumowania
summary(wynik_ANOVA)
```

Interpretacja wyniku testu ANOVA: Na podstawie pvalue (0.758) w teście ANOVA, możemy stwierdzić, że brak istotnych różnic w średnich wartości zmiany ekspresji pomiędzy kobietami a mężczyznami. Wynik ten sugeruje, że płeć nie ma wpływu na poziom zmiennej zależnej w analizowanych danych.

7. Wizualizacja danych

- Stwórz co najmniej 3 różne typy wykresów odpowiednie dla danych (np. wykresy punktowe, pudełkowe, słupkowe, liniowe, heatmapy), przynajmniej dwa z nich utwórz za pomocą pakietu ggplot2. Dostosuj wygląd wykresów (etykiety, tytuły, legendy, motywy) dla estetycznego przedstawienia danych

```
# Histogram tworzony funkcja wbudowaną hist()
wykres <- hist(dane$BASEMEAN,
     main = "Histogram średnich ekspresji genów",
     xlab = "Wartości",
     col = "skyblue",
     border = "white")
```

```r
# Wyświetlenie wykresu
print(wykres)
```

```r
# Wykres pudełkowy (ggplot2)
# Wczytanie odpowiednich pakietów
library(ggplot2)
library(RColorBrewer)

# Zamieniam kolumnę Płeć na faktor
dane$Płeć <- as.factor(dane$Płeć)

# Ustawienie palety kolorów
kolory <- c("lightblue", "lightgreen")

# Tworzenie wykresu pudełkowego
plot <- ggplot(dane, aes(x = Płeć, y = BASEMEAN, fill = Płeć)) +
  geom_boxplot() +
  scale_fill_manual(values = kolory) +
  ggtitle("Poziomy ekspresji genów wg płci") +
  xlab("Płeć") +
  ylab("Poziom ekspresji genów") +
  theme_minimal()

# Wyświetlenie wykresu
print(plot)
```

```r
# Histogram
plot_his <- ggplot(dane, aes(x = BASEMEAN)) +
  geom_histogram(binwidth = 10, fill = "steelblue", color = "white") +
  ggtitle("Rozkład ekspresji genów") +
  theme_classic()

# Wyświetlenie wykresu
print(plot_his)
```

8. Funkcje własne (1 punkt)

- Stwórz własną funkcję na danych z ramki danych, która pozwoli na automatyzację danych analiz np. filtrację danych na podstawie określonych wartości.

```r
# Funkcja filtrująca dane po kolumnie
filtruj_dane_p_istotne <- function(dane, wartosc = "istotna") {
  # Zamiana kolumny na faktor
  dane$P_VALUE_JAKO_FAKTOR <- as.factor(dane$P_VALUE_JAKO_FAKTOR)

  # Filtrowanie danych po wartości "istotna"
  dane_filtrowane <- dane[dane$P_VALUE_JAKO_FAKTOR == wartosc, ]

  return(dane_filtrowane)
}

# Użycie funkcji
```

```r
dane_przefiltrowane <- filtruj_dane_p_istotne(dane)

# Sprawdzenie pierwszych wierszy
head(dane_przefiltrowane)
```