ANALIZA DANYCH ANKIETOWYCH
Sprawozdanie 1

Adriana Naumczuk, Kostiantyn Skopych, Katarzyna Zalewska

Lista 1

Zadanie 1

- Polecenie:

Zadanie polega na opracowaniu danych z pliku „Choroba.csv”. Plik zawiera informacje na temat 196 osób przypisując je do grup ze względu na wiek, status ekonomiczny, sektor, oszczędności oraz zdrowie.

- Kod:

Do wykonania poleceń potrzebne są pakiety: tidyverse, sjmisc, vcd.

```
install.packages('tidyverse')
install.packages('sjmisc')
install.packages("vcd")
```

```
library(tidyverse)
library(sjmisc)
library(vcd)
```

Najpierw wczytujemy dane pamiętając aby były one w odpowiednim folderze i sprawdzamy ich typy. Trzeba przygotować dane aby wykonać wszystkie polecenia. Chcemy aby w ostateczności wszystkie zmienne oprócz wieku były zmiennymi jakościowymi, ponieważ wtedy wartości ich nie będą musiały być binarne.

```
dane <- read.csv2("Choroba.csv")
dane <- dane %>% mutate(CHORY_ZD=as.factor(CHORY_ZD), OSZCZED=as.factor(OSZCZED),
SEKTOR=as.factor(SEKTOR), STATUS=as.factor(STATUS))
```

Po zmianie zmiennych trzeba zmienić ich wartości na tekstowe wg propozycji w poleceniu czyli np. Status: ‘1’ = ‘Wysoki’.

```
dane <- dane %>% mutate(STATUS=fct_recode(STATUS, "Wysoki"="1", "Średni"="2", "Niski"="3"),
                        SEKTOR=fct_recode(SEKTOR, "Osoba z sektora 1"="1", "Osoba z sektora 2"="2"),
        OSZCZED=fct_recode(OSZCZED, "Ma oszczędności"="1", "Nie ma oszczędności"="0"),
        CHORY_ZD=fct_recode(CHORY_ZD, "Osoba jest chora"="1", "Osoba jest zdrowa"="0"))
```

Gdy nasze dane wyglądają już prawidłowo, możemy przejść do wykonywania poleceń.

A) Wyświetlamy tabele liczności dla zmiennych OSZCZED i CHORY_ZD (kod dla CHORY_ZD analogicznie):

1) Ze względu na wszystkie dane

```
dane %>% count(OSZCZED) %>% mutate (prop=n/sum(n))
```

2) Ze względu na zmienną status

```
dane %>% filter(STATUS=="Wysoki") %>% count (OSZCZED) %>% mutate (prop=n/sum(n))
dane %>% filter(STATUS=="Średni") %>% count (OSZCZED) %>% mutate (prop=n/sum(n))
dane %>% filter(STATUS=="Niski") %>% count (OSZCZED) %>% mutate (prop=n/sum(n))
```

3) Ze względu na zmienną sektor

```
dane %>% filter(SEKTOR==" Osoba z sektora 1") %>% count (OSZCZED) %>% mutate (prop=n/sum(n))
dane %>% filter(SEKTOR==" Osoba z sektora 2") %>% count (OSZCZED) %>% mutate (prop=n/sum(n))
```

4) Ze względu na obie zmienne status i sektor

```
dane %>% filter(STATUS=="Wysoki") %>% filter(SEKTOR=="Osoba z sektora 1") %>% count (OSZCZED)
%>% mutate (prop=n/sum(n))
dane %>% filter(STATUS=="Średni") %>% filter(SEKTOR=="Osoba z sektora 1") %>% count (OSZCZED)
%>% mutate (prop=n/sum(n))
dane %>% filter(STATUS=="Niski") %>% filter(SEKTOR=="Osoba z sektora 1") %>% count (OSZCZED)
%>% mutate (prop=n/sum(n))
dane %>% filter(STATUS=="Wysoki") %>% filter(SEKTOR=="Osoba z sektora 2") %>% count (OSZCZED)
%>% mutate (prop=n/sum(n))
dane %>% filter(STATUS=="Średni") %>% filter(SEKTOR=="Osoba z sektora 2") %>% count (OSZCZED)
%>% mutate (prop=n/sum(n))
dane %>% filter(STATUS=="Niski") %>% filter(SEKTOR=="Osoba z sektora 2") %>% count (OSZCZED)
%>% mutate (prop=n/sum(n))
```

B) Sporządzamy tabelę wielodzielczą uwzględniającą zmienne Chory/Zdrowy i Sektor.

```
dane %>% group_by(SEKTOR) %>% count(CHORY_ZD) %>% pivot_wider(names_from=CHORY_ZD,
values_from=n)
```

C) Sporządzamy tabelę wielodzielczą uwzględniającą zmienne Chory/Zdrowy i Status.

```
dane %>% group_by(STATUS) %>% count(CHORY_ZD) %>% pivot_wider(names_from=CHORY_ZD,
values_from=n)
```

D) Kategoryzujemy zmienną wiek dodając nową kolumnę WIEK_KAT

```
dane <- dane %>% mutate(WIEK_KAT = cut(WIEK,
                                       breaks = c(0,15,30,45,60,75,Inf),
                                       labels = c("0-15","15-30","30-45","45-60","60-75","75-Inf")))
```

E) Najpierw liczymy osoby z kategorii STATUS i przypisujemy wyniki do zmiennej. Następnie wyświetlamy wykres słupkowy oraz kołowy.

```
counts <- count(dane, STATUS)
barplot(height = counts$n, names = counts$STATUS, col = rainbow(3), main='Wykres słupkowy dla
zmiennej STATUS', ylim=range(pretty(c(0, counts$n))))
pie(counts$n, col = rainbow(3), main='Wykres kołowy dla zmiennej STATUS', labels=c('Wysoki',
'Średni','Niski'))
```

F) Najpierw wyliczamy liczność osób z kategorii CHORY_ZD z sektora 1 i tworzymy wykres słupkowy wyników, następnie postępujemy analogicznie dla osób z sektora 2.

```
num1 <- dane %>% filter(SEKTOR=="Osoba z sektora 1") %>% count(CHORY_ZD)
barplot(height = num1$n,col = rainbow(2), names = num1$CHORY_ZD, main='Wykres słupkowy dla
zmiennej CHORY_ZD dla SEKTORA 1', ylim=range(pretty(c(0, num1$n))), ylab='Liczba osób')

num2 <- dane %>% filter(SEKTOR=="Osoba z sektora 2") %>% count(CHORY_ZD)
barplot(height = num2$n, col = rainbow(2), names = num2$CHORY_ZD, main='Wykres słupkowy dla
zmiennej CHORY_ZD dla SEKTORA 2', ylim=range(pretty(c(0, num2$n))), ylab='Liczba osób')
```

- Wyniki:

Po wczytaniu danych wyglądają one następująco:

```
Rows: 196
Columns: 5
$ WIEK     <int> 33, 35, 6, 60, 18, 26, 6, 31, 26, 37, 23, 23, 27, 9~
$ STATUS   <int> 1, 1, 1, 1, 3, 3, 3, 2, 2, 2, 1, 1, 1, 1, 1, 1, ~
$ SEKTOR   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, ~
$ CHORY_ZD <int> 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, ~
$ OSZCZED  <int> 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, ~
```

Natomiast gdy zmienimy ich typy na jakościowe i zmienimy zmienne na tekstowe to wyglądają tak:

```
$ WIEK     <int> 33, 35, 6, 60, 18, 26, 6, 31, 26, 37, 23, 23, 27, 9~
$ STATUS   <fct> Wysoki, Wysoki, Wysoki, Wysoki, Niski, Niski, Niski~
$ SEKTOR   <fct> Osoba z sektora 1, Osoba z sektora 1, Osoba z sekto~
$ CHORY_ZD <fct> Osoba jest zdrowa, Osoba jest zdrowa, Osoba jest zd~
$ OSZCZED  <fct> Ma oszczędności, Ma oszczędności, Nie ma oszczędnoś~
```

A)

1) Teraz można zobaczyć tabelę liczności osób mających i niemających oszczędności:

```
             OSZCZED   n      prop
1 Nie ma oszczędności  90 0.4591837
2     Ma oszczędności 106 0.5408163
```

oraz chorych i zdrowych:

```
             CHORY_ZD   n      prop
1 Osoba jest zdrowa 140 0.7142857
2  Osoba jest chora  56 0.2857143
```

2) Poniżej znajdują się tablice liczności ze względu na status kolejno: wysoki, średni i niski:

```
             OSZCZED  n      prop            CHORY_ZD  n      prop
1 Nie ma oszczędności 17 0.2207792  1 Osoba jest zdrowa 53 0.6883117
2     Ma oszczędności 60 0.7792208  2  Osoba jest chora 24 0.3116883

             OSZCZED  n      prop            CHORY_ZD  n      prop
1 Nie ma oszczędności 25 0.5102041  1 Osoba jest zdrowa 36 0.7346939
2     Ma oszczędności 24 0.4897959  2  Osoba jest chora 13 0.2653061

             OSZCZED  n      prop            CHORY_ZD  n      prop
1 Nie ma oszczędności 48 0.6857143  1 Osoba jest zdrowa 51 0.7285714
2     Ma oszczędności 22 0.3142857  2  Osoba jest chora 19 0.2714286
```

3) Następnie przedstawiamy tablice liczności ze względu na sektor kolejno: osoba z sektora 1, osoba z sektora 2:

```
             OSZCZED  n      prop            CHORY_ZD  n      prop
1 Nie ma oszczędności 65 0.5555556  1 Osoba jest zdrowa 95 0.8119658
2     Ma oszczędności 52 0.4444444  2  Osoba jest chora 22 0.1880342

             OSZCZED  n      prop            CHORY_ZD  n      prop
1 Nie ma oszczędności 25 0.3164557  1 Osoba jest zdrowa 45 0.5696203
2     Ma oszczędności 54 0.6835443  2  Osoba jest chora 34 0.4303797
```

4) Wyniki ostatniego polecenia w punkcie a) znajdują się poniżej. Są to tablice liczności dla OSZCZED oraz CHORY_ZD ze względu na zmienne STATUS i SEKTOR. Pierwsze trzy wiersze są dla osób z sektora 1 ze statusem kolejno: wysokim, średnim, niskim. Następne trzy są dla osób z sektora 2 w tej samej kolejności.

```
             OSZCZED  n      prop            CHORY_ZD  n      prop
1 Nie ma oszczędności 11 0.2894737  1 Osoba jest zdrowa 31 0.8157895
2     Ma oszczędności 27 0.7105263  2  Osoba jest chora  7 0.1842105

             OSZCZED  n      prop            CHORY_ZD  n      prop
1 Nie ma oszczędności 15 0.5769231  1 Osoba jest zdrowa 23 0.8846154
2     Ma oszczędności 11 0.4230769  2  Osoba jest chora  3 0.1153846
```

```
             OSZCZED  n      prop
1 Nie ma oszczędności 39 0.7358491
2     Ma oszczędności 14 0.2641509

             CHORY_ZD  n      prop
1 Osoba jest zdrowa 41 0.7735849
2  Osoba jest chora 12 0.2264151
```

```
             OSZCZED  n      prop
1 Nie ma oszczędności  6 0.1538462
2     Ma oszczędności 33 0.8461538

             CHORY_ZD  n      prop
1 Osoba jest zdrowa 22 0.5641026
2  Osoba jest chora 17 0.4358974
```

```
             OSZCZED  n      prop
1 Nie ma oszczędności 10 0.4347826
2     Ma oszczędności 13 0.5652174

             CHORY_ZD  n      prop
1 Osoba jest zdrowa 13 0.5652174
2  Osoba jest chora 10 0.4347826
```

```
             OSZCZED n      prop
1 Nie ma oszczędności 9 0.5294118
2     Ma oszczędności 8 0.4705882

             CHORY_ZD  n      prop
1 Osoba jest zdrowa 10 0.5882353
2  Osoba jest chora  7 0.4117647
```

B) Tabela wielodzielcza uwzględniająca zmienne CHORY_ZD i SEKTOR.

```
  SEKTOR            `Osoba jest zdrowa` `Osoba jest chora`
  <fct>                           <int>              <int>
1 Osoba z sektora 1                  95                 22
2 Osoba z sektora 2                  45                 34
```

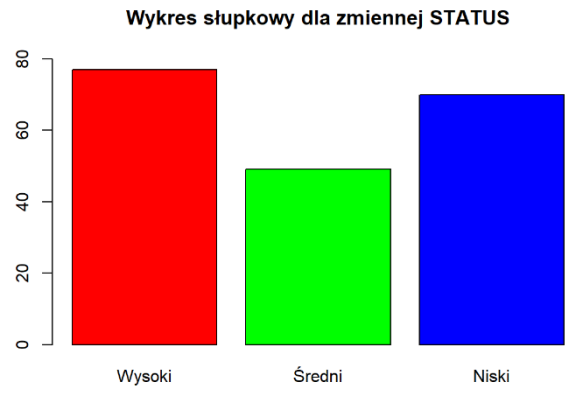
C) Tabela wielodzielcza uwzględniająca zmienne CHORY_ZD i STATUS.

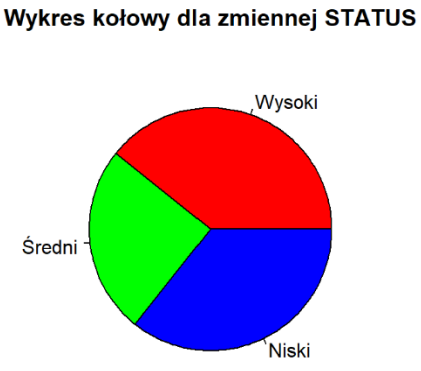
```
  STATUS `Osoba jest zdrowa` `Osoba jest chora`
  <fct>                <int>              <int>
1 Wysoki                  53                 24
2 Średni                  36                 13
3 Niski                   51                 19
```

D) Po kategoryzacji nasze dane wyglądają następująco:

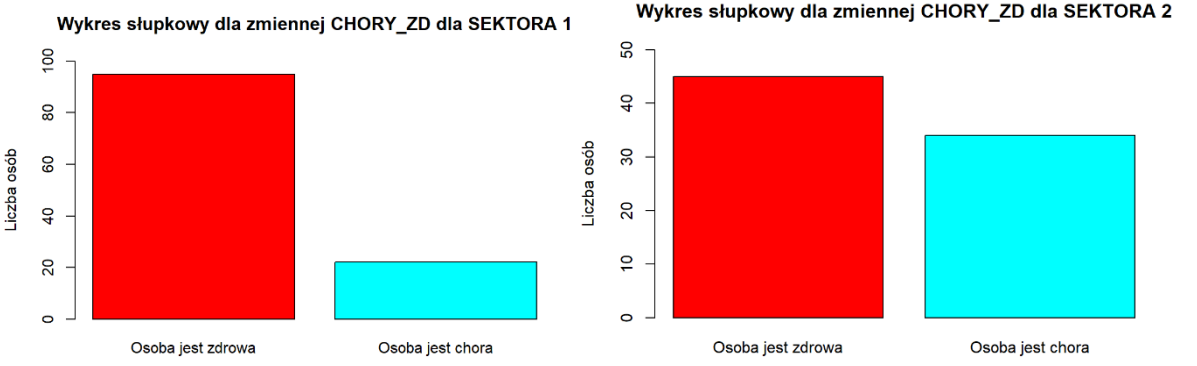
```
Rows: 196
Columns: 6
$ WIEK     <int> 33, 35, 6, 60, 18, 26, 6, 31, 26, 37, 23, 23, 27, 9~
$ STATUS   <fct> Wysoki, Wysoki, Wysoki, Wysoki, Niski, Niski, Niski~
$ SEKTOR   <fct> Osoba z sektora 1, Osoba z sektora 1, Osoba z sekto~
$ CHORY_ZD <fct> Osoba jest zdrowa, Osoba jest zdrowa, Osoba jest zd~
$ OSZCZED  <fct> Ma oszczędności, Ma oszczędności, Nie ma oszczędnoś~
$ WIEK_KAT <fct> 30-45, 30-45, 0-15, 45-60, 15-30, 15-30, 0-15, 30-4~
```

E) Wykres słupkowy i kołowy dla zmiennej STATUS.





F) Skategoryzowane wykresy słupkowe dla zmiennej CHORY_ZD ze względu na sektor.



- Podsumowanie:

Przygotowanie danych jest bardzo istotne w analizach statystycznych, ponieważ daje nam możliwości przedstawienia danych w wygodny i łatwy do interpretowania sposób.

Lista 1

Zadanie 2

- Polecenie

  W zadaniu mamy zapoznać się z plikiem „Reakcja.csv” oraz opracować wybrane wykresy przedstawiające zależności objawów wśród pacjentów.

- Kod:

Do wykonania polecenia potrzebne są dwa pakiety: tidyverse oraz vcd. Pobieramy je i wczytujemy dane z podanego pliku.

```
install.packages('tidyverse')
install.packages(‘vcd’)

library(tidyverse)
library(vcd)

dane <- read.csv2("Reakcja.csv")
```

Należy przygotować dane do dalszej interpretacji.

```
dane <- dane %>% mutate(Reakcja=as.factor(Reakcja), Miejsce=as.factor(Miejsce),
Rodzaj=as.factor(Rodzaj))

dane <- dane %>% mutate(Reakcja=fct_recode(Reakcja, "Nie nastąpiła poprawa"="0", "Nastąpiła
                poprawa"="1"), Miejsce=fct_recode(Miejsce, "Leczony w domu"="0", "Leczony
                w szpitalu"="1"), Rodzaj=fct_recode(Rodzaj, "Firma nr I"="0", "Firma nr II"="1"))
```

Liczymy osoby z kategorii Reakcja i przypisujemy do zmiennej aby wykonać wykres słupkowy i kołowy dla całej badanej grupy.

```
counts <- count(dane, Reakcja)

barplot(height = counts$n,names = counts$Reakcja, col = rainbow(3), main='Wykres słupkowy dla
zmiennej Reakcja', ylim=range(pretty(c(0, 160))))
```

```
pie(counts$n,col = rainbow(3), main='Wykres kołowy dla zmiennej Reakcja', labels=c('Nie nastąpiła poprawa', 'Nastąpiła poprawa')) # wykres kołowy statusu
```

Następnie wykonujemy wykresy dla zmiennej Reakcja w podgrupach.

1) Ze względu na zmienną Miejsce

```
num1 <- dane %>% filter(Miejsce=="Leczony w domu") %>% count(Reakcja) #nazywamy zmienna osoby z sektora 1 liczymy chorych i zdrowych
barplot(height = num1$n, names = num1$Reakcja, ylim=c(0,100), col=rainbow(5), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych w domu')

num2 <- dane %>% filter(Miejsce=="Leczony w szpitalu") %>% count(Reakcja) #nazywamy zmienna osoby z sektora 1 liczymy chorych i zdrowych
barplot(height = num2$n, names = num2$Reakcja, ylim=c(0,70), col=rainbow(5), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych w szpitalu')
```

2) Ze względu na zmienną Rodzaj

```
num3 <- dane %>% filter(Rodzaj=="Firma nr I") %>% count(Reakcja)
barplot(height = num3$n, names = num3$Reakcja, ylim=c(0,90), col=rainbow(3), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych w Firmie nr 1')

num4 <- dane %>% filter(Rodzaj=="Firma nr II") %>% count(Reakcja)
barplot(height = num4$n, names = num4$Reakcja, ylim=c(0,90), col=rainbow(3), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych w Firmie nr 2')
```

3) Ze względu na zmienną Dawka

```
num5 <- dane %>% filter(Dawka=="-3.204") %>% count(Reakcja)
barplot(height = num5$n, names = num5$Reakcja, ylim=c(0,40), col=rainbow(3), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych dawką -3.204')

num6 <- dane %>% filter(Dawka=="-2.903") %>% count(Reakcja)
barplot(height = num6$n, names = num6$Reakcja, ylim=c(0,40), col=rainbow(3), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych dawką -2.903')

num7 <- dane %>% filter(Dawka=="-2.602") %>% count(Reakcja)
barplot(height = num7$n, names = num7$Reakcja, ylim=c(0,40), col=rainbow(3), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych dawką -2.602')

num8 <- dane %>% filter(Dawka=="-2.301") %>% count(Reakcja)
barplot(height = num8$n, names = num8$Reakcja, ylim=c(0,30), col=rainbow(3), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych dawką -2.301')

num9 <- dane %>% filter(Dawka=="-2") %>% count(Reakcja)
barplot(height = num9$n, names = num9$Reakcja, ylim=c(0,25), col=rainbow(3), main='Wykres słupkowy dla zmiennej Reakcja', xlab='Reakcja', ylab='Liczba osób leczonych dawką -2')
```

- Wyniki:

Po wczytaniu danych wyglądają one następująco:

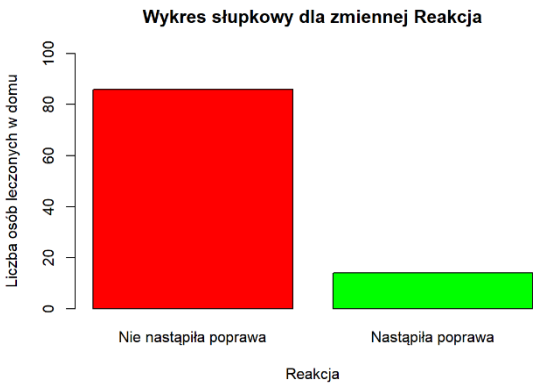
```
Rows: 200
Columns: 4
$ Reakcja <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0~
$ Dawka   <dbl> -3.204, -3.204, -3.204, -3.204, -3.204, -3.204, -3.2~
$ Rodzaj  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
$ Miejsce <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1~
```

Natomiast gdy zmienimy ich typy na jakościowe i zmienimy zmienne na tekstowe to wyglądają tak:

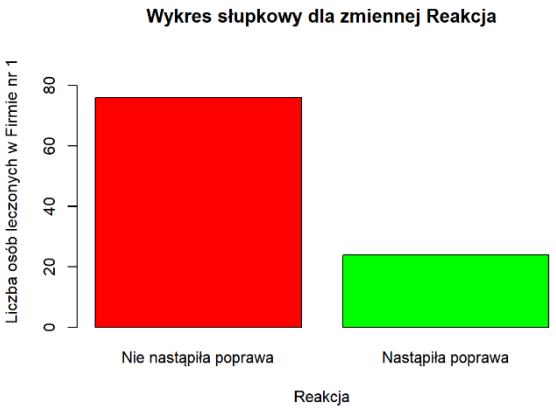
```
Rows: 200
Columns: 4
$ Reakcja <fct> Nie nastąpiła poprawa, Nie nastąpiła poprawa, Nie na~
$ Dawka   <dbl> -3.204, -3.204, -3.204, -3.204, -3.204, -3.204, -3.2~
$ Rodzaj  <fct> Firma nr I, Firma nr I, Firma nr I, Firma nr I, Firm~
$ Miejsce <fct> Leczony w domu, Leczony w domu, Leczony w domu, Lecz~
```

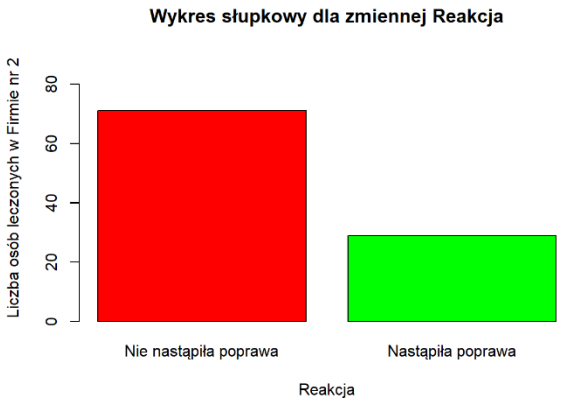
Przedstawimy teraz wykresy pokazujące jakie są warunki aby stan zdrowia się polepszył. Pozwolą one odpowiedzieć na pytania: która dawka daje lepsze wyniki? w której firmie więcej osób wyzdrowiało? czy ludzie zdrowieją częściej w domu czy w szpitalu?

1)



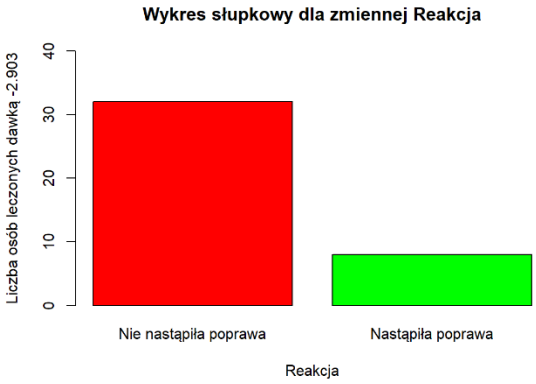
2)

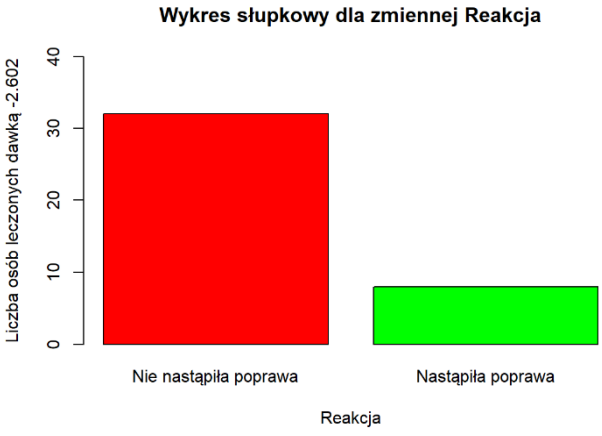

Wykres słupkowy dla zmiennej Reakcja
Liczba osób leczonych w Firmie nr 1
0
20
40
60
80
Nie nastąpiła poprawa
Nastąpiła poprawa
Reakcja

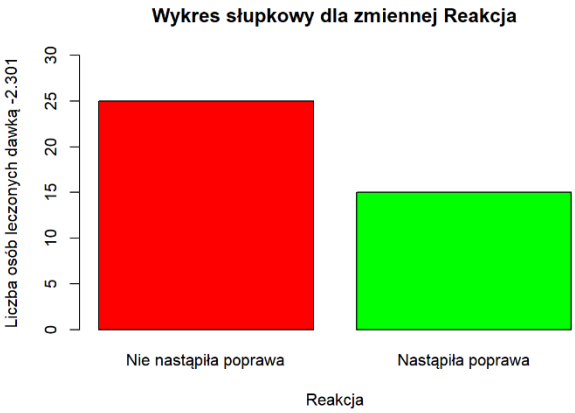

Wykres słupkowy dla zmiennej Reakcja
Liczba osób leczonych w Firmie nr 2
0
20
40
60
80
Nie nastąpiła poprawa
Nastąpiła poprawa
Reakcja

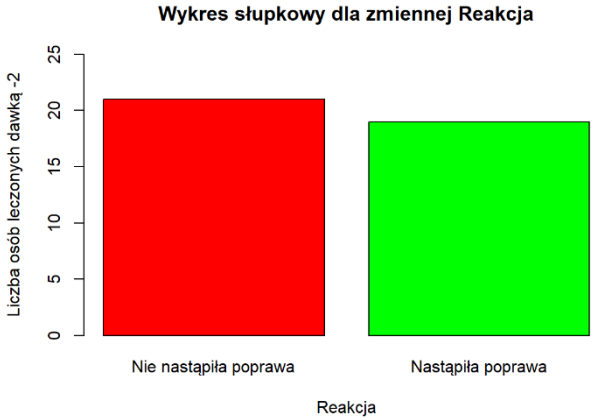
3)


Wykres słupkowy dla zmiennej Reakcja
Liczba osób leczonych dawką -3.204
0
10
20
30
40
Nie nastąpiła poprawa
Nastąpiła poprawa
Reakcja


Wykres słupkowy dla zmiennej Reakcja
Liczba osób leczonych dawką -2.903
0
10
20
30
40
Nie nastąpiła poprawa
Nastąpiła poprawa
Reakcja


Wykres słupkowy dla zmiennej Reakcja
Liczba osób leczonych dawką -2.602
0
10
20
30
40
Nie nastąpiła poprawa
Nastąpiła poprawa
Reakcja


Wykres słupkowy dla zmiennej Reakcja
Liczba osób leczonych dawką -2.301
0
5
10
15
20
25
30
Nie nastąpiła poprawa
Nastąpiła poprawa
Reakcja


Wykres słupkowy dla zmiennej Reakcja
Liczba osób leczonych dawką -2
0
5
10
15
20
25
Nie nastąpiła poprawa
Nastąpiła poprawa
Reakcja

- Podsumowanie:

Patrząc na powyższe wykresy możemy określić ogólne prawdopodobieństwo poprawy dzieląc liczbę osób, których zdrowie się poprawiło na ogólną liczbę osób, które przyjęły lek. Natomiast najwięcej można z nich uzyskać informacji na temat, w jaki sposób przyjąć lek, aby prawdopodobieństwo poprawy były największe. Jest to m.in. pójście do Firmy nr II, na dawkę '-2' i przyjąć lek w szpitalu. Te wszystkie czynniki sprawią, że prawdopodobieństwo będzie największe.

Lista 2

Zadanie 1

- Polecenie:

W tym zadaniu mamy zapoznać się z funkcją $sample$ z pakietu $stats$ oraz napisać program, którego celem będzie wylosowanie próbki rozmiaru około $1/10$ liczby przypadków danej bazy danych, ze zwracaniem oraz bez zwracania. Jako naszą bazę danych przyjmiemy plik „Choroba.csv" użyty w zadaniu 1 z listy 1.

- Teoria:

Funkcja $sample(x, size, replace, prob)$ służy do losowania próbki o określonym rozmiarze z zadanej bazy danych. Losowanie może się odbywać ze zwracaniem lub bez.

Parametry funkcji:

x to wektor, z którego będziemy losowali; może to też być liczba całkowita – w tym przypadku losujemy z wektora $1{:}\,n$, gdzie n to liczba całkowita;

$size$ definiuje ile próbek losujemy;

$replace$ z opcjami $True/False$ określa czy losujemy odpowiednio ze zwracaniem lub bez;

$prob$ jest parametrem, który określa jakie mają być wartości prawdopodobieństwa wylosowania konkretnych elementów z zadanego wektora.

- Kod:

Gdy już zapoznaliśmy się z działaniem funkcji $sample$ możemy użyć jej do wykonania dalszej części tego zadania.

Poniżej prezentujemy program do losowania próbki z naszej bazy danych:

```
dane <- read.csv2("Choroba.csv") # wczytanie danych
wektor <- 1:nrow(dane) # ustalenie wektora (ilość danych w kolumnie)
rozmiar_proby <- round(nrow(dane)/10) #ustalenie rozmiaru próby - zaokrąglona 1/10 całości

losowanie_ze_zwracaniem <- sample(wektor, size=rozmiar_proby, replace=TRUE)
losowanie_bez_zwracania <- sample(wektor, size=rozmiar_proby, replace=FALSE)
```

Stworzymy funkcję pomocniczą do odczytu danych na wylosowanych miejscach:

```
odczytanie_danych <- function(losowanie){
wylosowane_dane <- as.numeric()
```

```
for (i in 1:length(losowanie)){
nr <- losowanie[i]
wylosowane_dane[i] = dane[nr, 1]}
return (wylosowane_dane)}
```

Teraz możemy zamieścić otrzymane dane w tabeli:

```
wylosowane_dane1 <- odczytanie_danych(losowanie_ze_zwracaniem)
wylosowane_dane2 <- odczytanie_danych(losowanie_bez_zwracania)
data.frame(losowanie_ze_zwracaniem,wylosowane_dane1,losowanie_bez_zwracania,wylosowa
ne_dane2)
```

- Wyniki:

	losowanie_ze_zwracaniem	wylosowane_dane1	losowanie_bez_zwracania	wylosowane_dane2
1	176	22	165	13
2	145	3	11	23
3	118	7	128	70
4	140	30	145	3
5	72	27	44	70
6	165	13	166	21
7	1	33	66	22
8	38	6	143	32
9	57	39	48	65
10	19	6	157	79
11	70	28	114	15
12	29	61	1	33
13	196	24	2	35
14	119	2	170	73
15	69	46	174	7
16	34	4	178	13
17	73	28	149	13
18	44	70	136	35
19	31	16	65	59
20	176	22	134	39

- Podsumowanie:

W pliku „Choroba.csv” jest 196 wierszy z podziałem na 5 kategorii. Naszym celem było wylosowanie próbki o rozmiarze około $1/10$ z 196 przy użyciu funkcji $sample$. Stworzyliśmy więc program, który losuje liczby z wektora $1{:}\,196$, a następnie odczytuje dane z pierwszej kolumny zatytułowanej „WIEK” na wylosowanych miejscach.

Lista 2

Zadanie 2

- Polecenie:

Zadanie polega na zaproponowaniu algorytmu do generowania liczb z rozkładu dwumianowego, napisaniu programu na podstawie wspomnianego algorytmu oraz pokazaniu, że jest on poprawny.

- Teoria

Rozkład dwumianowy to dyskretny rozkład prawdopodobieństwa opisujący liczbę sukcesów k w ciągu N niezależnych prób, z których każda ma stałe prawdopodobieństwo sukcesu równe p. Teoretyczna wartość oczekiwana tego rozkładu wynosi: Np, natomiast wariancja: $Np(1-p)$.

- Algorytm

Niech:

p – prawdopodobieństwo sukcesu w próbie Bernoulliego;

N – liczba prób Bernoulliego.

1. Tworzymy licznik sukcesów $sukcesy = 0$.
2. Generujemy zmienną losową X z rozkładu jednostajnego $U(0,1)$.
3. Jeśli $X > (1-p)$ to powiększamy $sukcesy$ o 1.
4. Powtarzamy podpunkty 1. i 2. N razy.
5. Otrzymany wynik $sukcesy$ to liczba sukcesów k w ciągu N niezależnych prób.

- Kod:

Poniżej prezentujemy program napisany na podstawie zaproponowanego algorytmu.

```
binomial <- function(n, N, p) {
 lista_k <- as.numeric()
 for (i in 1:n){ # powtarzamy calosc n razy
  X <- runif(N, min=0, max=1)  # losujemy N liczb od 0 do 1
  sukcesy <- 0
  for (j in 1: N){
    if (X[j]>(1-p)){sukcesy = sukcesy + 1}}
 lista_k[i]=sukcesy}
return (lista_k)}
```

Funkcja $binomial$ generuje n liczb z rozkładu dwumianowego $B(N,p)$.

Stworzymy teraz dwa przykładowe histogramy (dla $n = 10$ oraz $n = 100$), na których porównamy liczby wygenerowane przez funkcję napisaną przez nas oraz funkcję wbudowaną $rbinom$.

Poniżej znajduje się kod do wygenerowania liczb z rozkładu dwumianowego przy użyciu funkcji $binomial$ oraz $rbinom$ dla parametrów $n = 10, N = 1000, p = 0.5$.

```
nasz_dwumianowy <- binomial(10,1000,0.5)
wbud_dwumianowy <- rbinom(10,1000,0.5)

hist_info1 <- hist(nasz_dwumianowy, plot=FALSE) # informacje o histogramie binomial
hist_info2 <- hist(wbud_dwumianowy, plot=FALSE) # informacje o histogramie rbinom

hist_counts1 <- hist_info1$counts # liczności z histogramu binomial
hist_counts2 <- hist_info2$counts # liczności z histogramu rbinom
```

Teraz tworzymy histogram dla wygenerowanych danych.

```
hist(nasz_dwumianowy, col="red", xlim = range(pretty(range(nasz_dwumianowy,
wbud_dwumianowy))), ylim = c(0,max(hist_counts1,hist_counts2)), main = "Histogram rozkładu
dwumianowego", xlab = "Liczba sukcesów", ylab = "Liczność")

hist(wbud_dwumianowy, col= "blue", density = 20, add=TRUE, lty=1)

legend("topright", c("binomial", "rbinom"), lty=c(1, 2), col=c("red", "blue"))
```

Sprawdzimy również jak wyglądają wygenerowane dane oraz jaka jest ich średnia i wariancja.

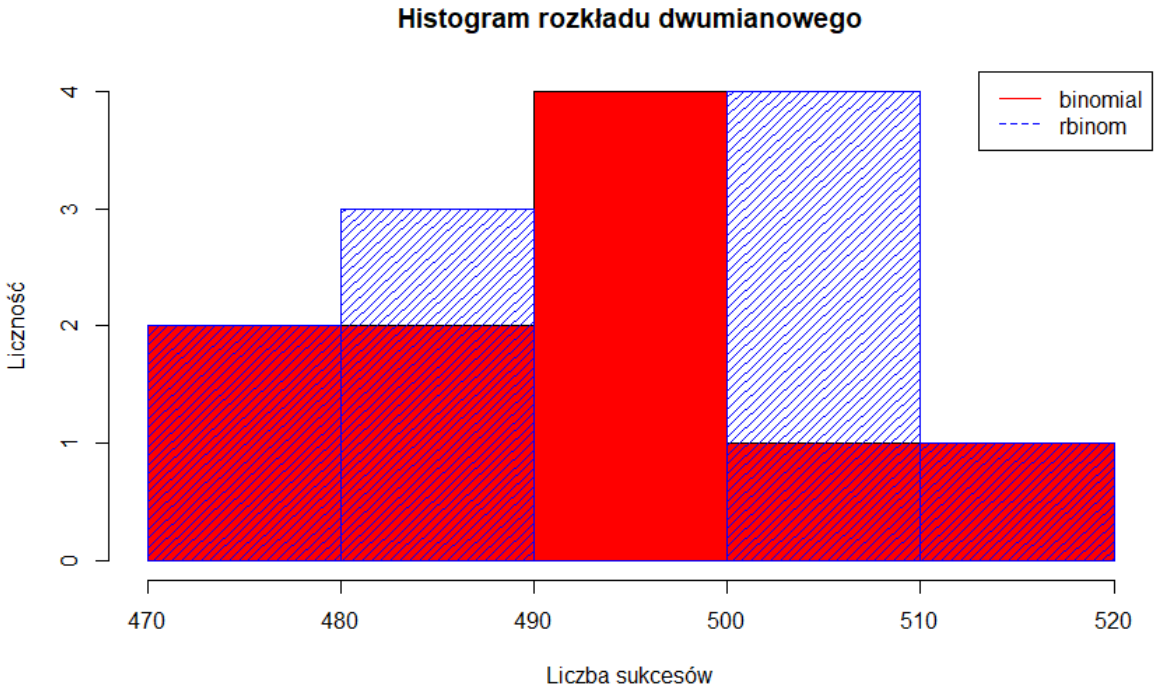
```
glimpse(nasz_dwumianowy)
glimpse(wbud_dwumianowy)
glimpse(mean(nasz_dwumianowy))
glimpse(mean(wbud_dwumianowy))
glimpse(var(nasz_dwumianowy))
glimpse(var(wbud_dwumianowy))
```

- Wyniki:

Dane wygenerowane dla parametrów $n = 10, N = 1000, p = 0.5$ wyglądają następująco:

```
> glimpse(nasz_dwumianowy)
 num [1:10] 492 491 486 520 493 479 472 496 503 486
> glimpse(wbud_dwumianowy)
 int [1:10] 510 503 482 477 505 471 504 489 487 517
```

Poniżej znajduje się histogram wygenerowanych wcześniej danych.



Teraz porównamy teoretyczne wartości średniej oraz wariancji rozkładu dwumianowego z tymi otrzymanymi przez nas.

Teoretyczna wartość oczekiwana wynosi: $Np = 1000 \cdot 0.5 = 500$.

Teoretyczna wariancja wynosi: $Np(1-p) = 500 \cdot 0.5 = 250$.

Dla pierwszego zestawu danych średnia i wariancja prezentują się następująco:

Dla danych wygenerowanych za pomocą funkcji $binomial$:

Średnia wynosi 492.

Wariancja jest równa 174.

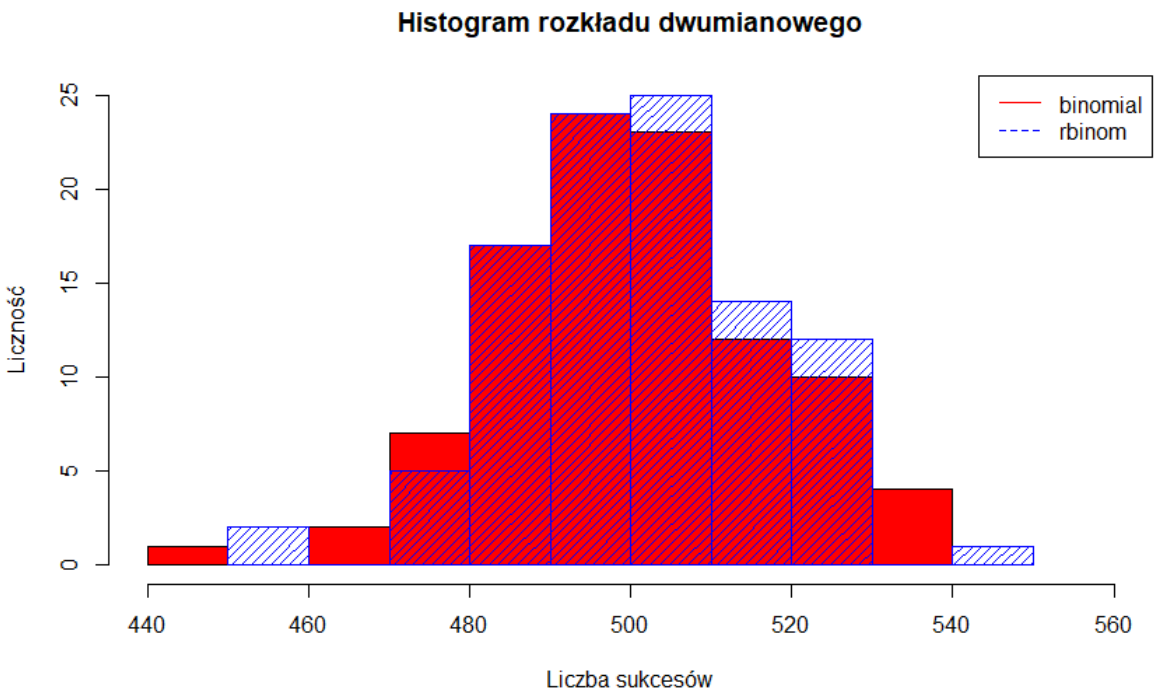
Dla danych wygenerowanych za pomocą funkcji $rbinom$:

Średnia wynosi 494.

Wariancja jest równa 236.

Analogicznie generujemy dane dla parametrów $n = 100, N = 1000, p = 0.5$.

Histogram dla tych danych wygląda następująco:



Tak jak dla poprzedniego zestawu danych liczymy średnią oraz wariancję:

Dla danych wygenerowanych za pomocą funkcji $binomial$:

Średnia wynosi 501.

Wariancja jest równa 275.

Dla danych wygenerowanych za pomocą funkcji $rbinom$:

Średnia wynosi 502.

Wariancja jest równa 240.

- Podsumowanie:

Zaproponowaliśmy algorytm do generowania liczb z rozkładu dwumianowego oraz napisaliśmy program na bazie tego algorytmu. Pokazaliśmy, że program działa poprawnie porównując histogramy danych wygenerowanych przy pomocy stworzonej przez nas funkcji oraz funkcji wbudowanej w R - $rbinom$. Sprawdziliśmy też jakie są wartości średniej oraz wariancji dla wygenerowanych danych i porównaliśmy je z wartościami teoretycznymi. Możemy zauważyć, że dla większej ilości danych (dla $n = 100$) wartości średniej oraz wariancji dla wygenerowanych przez nas danych są bliższe wartościom teoretycznym tych statystyk dla rozkładu dwumianowego. Ze względu na losowość występującą w generowaniu liczb z rozkładu jednostajnego wartości te zawsze będą nieznacznie się różnić.

Lista 2

Zadanie 3

- Polecenie:

Zadanie polega na zaproponowaniu algorytmu generowania wektora z rozkładu wielomianowego. Należy również napisać program do generowania tych wektorów na podstawie wspomnianego algorytmu oraz pokazać, że zaproponowany algorytm jest poprawny.

- Teoria:

W teorii prawdopodobieństwa rozkład wielomianowy jest uogólnieniem rozkładu dwumianowego. Modeluje on na przykład prawdopodobieństwo wystąpień każdej ze stron k-bocznej matrycy rzucanej n razy. Poniższy wzór określa rozkład wielomianowy:

Prawdopodobieństwo, że $X = x$ jest postaci:

$$P(X = x) = \frac{n!}{x_1!x_2!\cdots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k},$$

gdzie $x_i \in \{0,\dots,n\}, \mathrm{i} = 1,\dots,\mathrm{k}, \sum_{i=1}^{k} x_i = n$, a $p = (p_1,\cdots,p_k)$ jest wektorem prawdopodobieństw p_i wystąpienia *i-tego* zdarzenia.

Teoretyczna wartość oczekiwana tego rozkładu wynosi: $\mathrm{E}(X_i) = Np_i$, natomiast wariancja: $Var(X_i) = Np_i(1 - p_i)$.

- Algorytm:

Niech:

$prob = (p_1,\cdots,p_k)$ jest wektorem prawdopodobieństw p_i wystąpienia *i-tego* zdarzenia;

N – liczba powtórzeń eksperymentu.

1. Tworzymy listę $przedzialy$ zawierającą prawe krańce kolejnych przedziałów, gdzie długości przedziałów równe są kolejnym wartościom z zadanej listy.
2. Normalizujemy listę $przedzialy$, tak aby najwyższą wartością było 1.
3. Tworzymy listę $wektor$ z samymi zerami o długości równej długości listy $przedzialy$.
4. Generujemy zmienną losową X z rozkładu jednostajnego $U(0,1)$.
5. Jeśli $X \leq przedzialy[k]$ to powiększamy $wektor[k]$ o 1, gdzie $\mathrm{k} = 1,\dots,\mathrm{l}$, l – długość listy $przedzialy$. Oznacza to, że przechodzimy przez kolejne przedziały do momentu, gdy

liczba X znajdzie się w aktualnie sprawdzanym. Gdy tak się stanie – powiększamy odpowiedni licznik o 1 i przerywamy pętlę.

6. Powtarzamy podpunkty 4. i 5. N razy.
7. Otrzymany wynik $wektor$ to wektor zawierający liczbę wystąpień zmiennej losowej X w każdym z przedziałów w ciągu N niezależnych prób.

- Kod:

Najpierw tworzymy funkcję odpowiedzialną za stworzenie listy z krańcami kolejnych przedziałów, gdzie długości przedziałów równe są kolejnym wartościom z zadanej listy.

```
stworz_przedzialy <- function(lista){
 przedzialy <- as.numeric()
 przedzialy[1] = lista[1]
 for (i in 2:length(lista)){
  przedzialy[i] = przedzialy[i-1] + lista[i]}
return (przedzialy)}
```

Tworzymy również funkcję, która zwraca listę znormalizowanych wartości z zadanej listy, tak aby sumowały się one do 1.

```
normalizacja <- function(lista){
 norm_przedzialy <- lista*(1/lista[length(lista)])
return (norm_przedzialy)}
```

Teraz możemy przejść do funkcji $multinomial$ odpowiedzialnej za tworzenie pojedynczego wektora wielomianowego.

```
multinomial <- function(N, prob){
```

Poniżej tworzymy listę ze znormalizowanymi krańcami kolejnych przedziałów, gdzie długości przedziałów równe są kolejnym wartościom z zadanej listy prób.

```
  przedzialy <- normalizacja(stworz_przedzialy(prob))
  wektor <- as.numeric(rep(0,length(przedzialy))) # tworzymy listę liczników
  X <- runif(N, min=0, max=1)  # losujemy N liczb od 0 do 1

  for (j in 1:N){
   for (k in 1:length(przedzialy)){
```

Jeśli wylosowana liczba jest w przedziale to dodajemy 1 do odpowiedniego licznika i przerywamy pętlę.

```
     if (X[j]<=przedzialy[k]){
wektor[k] = wektor[k] + 1
break}}}
return (wektor)}
```

A na końcu do stworzenia funkcji $n\_multinomials$, która tworzy macierz zawierającą n wektorów wielomianowych podanych w kolejnych kolumnach.

```
n_multinomials <- function(n, N, prob){
```

Tworzymy macierz z zerami, do której dodawane będą wektory wielomianowe.

```
n_wektorow <- matrix(rep(0,length(prob)))
for (i in 1:n){ # powtarzamy całośc n razy
wektor <- multinomial(N, prob) # tworzymy wektor za pomocą funkcji multinomial
n_wektorow <- cbind(n_wektorow, wektor) } # dodajemy wektor do macierzy
n_wektorow <- n_wektorow[,-1] # usuwamy zbędną kolumnę z zerami
return (n_wektorow) }  # zwracamy macierz z wektorami wielomainowymi
```

Przykładowe wywołania funkcji $n\_multinomials$ oraz funkcji wbudowanej $rmultinom$:

```
p1 = c(1,2,5)
p2 = c(1,2,3,4)

wynik1 <- n_multinomials(10, 1000, p1)
print(wynik1)

rmultinom(10, size = 1000, prob = p1)
wynik2 <- n_multinomials(10, 100, p2)

print(wynik2)
rmultinom(10, size = 100, prob = p2)
```

- Wyniki:

Dla danych wejściowych $n = 10, size = N = 1000, prob = [1,2,5]$ macierze z wektorami wielomianowymi prezentują się następująco:

Macierze stworzone za pomocą funkcji $n\_multinomials$ oraz funkcji wbudowanej $rmultinom$:

```
> wynik1 <- n_multinomials(10, 1000, p1)
> print(wynik1)
     wektor wektor wektor wektor wektor wektor wektor wektor wektor wektor
[1,]    133    134    126    131    116    131    134    134    122    118
[2,]    238    274    247    245    246    236    232    267    245    242
[3,]    629    592    627    624    638    633    634    599    633    640
> rmultinom(10, size = 1000, prob = p1)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]  108  118  132  123  137  115  125  107  135   129
[2,]  245  237  245  247  228  250  258  245  250   249
[3,]  647  645  623  630  635  635  617  648  615   622
```

W powyższych macierzach wartości w kolejnych wierszach oznaczają:

1. Liczba wystąpień zmiennej losowej X w przedziale $[0, 0.125]$ w ciągu $N = 1000$ prób.
2. Liczba wystąpień zmiennej losowej X w przedziale $(0.125, 0.375]$ w ciągu $N = 1000$ prób.
3. Liczba wystąpień zmiennej losowej X w przedziale $(0.375, 1.0]$ w ciągu $N = 1000$ prób.

W kolejnych kolumnach umieszczone są wartości pojedynczych wektorów wielomianowych.

Prawdopodobieństwa wystąpienia zmiennej X w kolejnych przedziałach wynoszą więc odpowiednio: $p_1 = 0.125$ dla pierwszego przedziału, $p_2 = 0.25$ dla drugiego przedziału oraz $p_3 = 0.625$ dla trzeciego przedziału.

Porównamy również wartości oczekiwane oraz wariancje wartości otrzymanych dla odpowiednich przedziałów z ich teoretycznymi odpowiednikami:

Przedział	Średnia z danych	Wartość teoretyczna $\mathrm{E}(X_i) = Np_i$	Wariancja z danych	Wartość teoretyczna $Var(X_i) = Np_i(1-p_i)$
$[0, 0.125]$	127.9	125	48.32222	109.375
$(0.125, 0.375]$	247.2	250	176.6222	187.5
$(0.375, 1.0]$	624.9	625	265.4333	234.375

Sprawdzimy te wartości również dla danych wygenerowanych dla tych samych parametrów N oraz $prob$, ale dla $n = 100$.

Przedział	Średnia z danych dla n=100	Wartość teoretyczna $\mathrm{E}(X_i) = Np_i$	Wariancja z danych dla n=100	Wartość teoretyczna $Var(X_i) = Np_i(1-p_i)$
$[0, 0.125]$	125.55	125	104.7348	109.375
$(0.125, 0.375]$	249.1	250	169.6465	187.5
$(0.375, 1.0]$	625.35	625	229.1187	234.375

Dla danych wejściowych $n = 10, size = N = 100, prob = [1,2,3,4]$ macierze z wektorami wielomianowymi prezentują się następująco:

```
> wynik2 <- n_multinomials(10, 100, p2)
> print(wynik2)
     wektor wektor wektor wektor wektor wektor wektor wektor wektor wektor
[1,]     14     10      7     16     11     12      6     13      7      9
[2,]     18     22     21     14     24     22     26     16     22     15
[3,]     31     31     28     20     31     32     23     33     27     33
[4,]     37     37     44     50     34     34     45     38     44     43
> rmultinom(10, size = 100, prob = p2)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]   12    7    9   11    9   10   13    8    8     8
[2,]   24   18   23   14   15   26   19   17   19    22
[3,]   25   30   21   30   38   26   36   31   29    35
[4,]   39   45   47   45   38   38   32   44   44    35
```

W powyższych macierzach wartości w kolejnych wierszach oznaczają:

1. Liczba wystąpień zmiennej losowej X w przedziale $[0, 0.1]$ w ciągu $N = 100$ prób.
2. Liczba wystąpień zmiennej losowej X w przedziale $(0.1, 0.3]$ w ciągu $N = 100$ prób.
3. Liczba wystąpień zmiennej losowej X w przedziale $(0.3, 0.6]$ w ciągu $N = 100$ prób.
4. Liczba wystąpień zmiennej losowej X w przedziale $(0.6, 1.0]$ w ciągu $N = 100$ prób.

W kolejnych kolumnach umieszczone są wartości pojedynczych wektorów wielomianowych.

Prawdopodobieństwa wystąpienia zmiennej X w kolejnych przedziałach wynoszą odpowiednio: $p_1 = 0.1$ dla pierwszego przedziału, $p_2 = 0.2$ dla drugiego przedziału, $p_3 = 0.3$ dla trzeciego przedziału oraz $p_4 = 0.4$ dla czwartego przedziału.

Porównamy również wartości oczekiwane oraz wariancje wartości otrzymanych dla odpowiednich przedziałów z ich teoretycznymi odpowiednikami:

Przedział	Średnia z danych	Wartość teoretyczna $\mathrm{E}(X_i) = Np_i$	Wariancja z danych	Wartość teoretyczna $Var(X_i) = Np_i(1-p_i)$
$[0, 0.1]$	10.5	10	10.94444	9
$(0.1, 0.3]$	20	20	16.22222	16
$(0.3, 0.6]$	28.9	30	19.43333	21
(0.6, 1.0]	40.6	40	28.48889	24

Sprawdzimy te wartości również dla danych wygenerowanych dla tych samych parametrów N oraz $prob$, ale dla $n = 1000$.

Przedział	Średnia z danych	Wartość teoretyczna $\mathrm{E}(X_i) = Np_i$	Wariancja z danych	Wartość teoretyczna $Var(X_i) = Np_i(1-p_i)$
$[0, 0.1]$	9.905	10	9.805781	9
$(0.1, 0.3]$	20.049	20	15.66426	16
$(0.3, 0.6]$	30.078	30	20.89681	21
(0.6, 1.0]	39.968	40	23.65663	24

- Podsumowanie:

Napisaliśmy program do generowania macierzy zawierającej wektory wielomianowe. Pokazaliśmy, że zaproponowany przez nas algorytm jest poprawny poprzez porównanie wygenerowanych przez nas danych z danymi wygenerowanymi przez wbudowaną funkcję $rmultinom$. Sprawdziliśmy również jak mają się wartości średniej oraz wariancji z odpowiednich danych do ich teoretycznych odpowiedników. Dla $n = 10$ możemy zauważyć pewne odchylenia od wartości teoretycznej w przypadku wariancji. Patrząc jednak na inne otrzymane wyniki umieszczone w tabelach, możemy stwierdzić, że dla większych n (oraz większych N) wartości sprawdzanych statystyk uzyskane z naszych danych są bliższe tym teoretycznym.

Lista 2

Zadanie 4

- Polecenie:

Należy wybrać własny temat badania ankietowego i dogłębnie go przedstawić. Następnie zaproponować sposób generowania wyników zaproponowanej ankiety.

- Kod:

1. Zdefiniowanie celu badania.
   Ocena warunków w toaletach publicznych w celu ich poprawy.
2. Grupa docelowa.
   Grupa ludzi korzystających z toalet publicznych.
3. Sposób zbierania danych.
   Ankieta internetowa.
4. Propozycje kwestionariusza.
   Metryczka: płeć, wiek, miejsce zamieszkania.
   A) Czy korzystasz/korzystałeś z toalety publicznej? TAK / NIE.
   B) Jak oceniasz czystość toalet publicznych? Bardzo źle/ źle/ nie mam zdania/ czysto/ bardzo

czysto.
C) Jak często zdarzył się brak papieru toaletowego/mydła itd.? Bardzo rzadko/ rzadko/ nie mam zdania/ często/ bardzo często.
D) Jak bardzo przeszkadza Ci obecność innych osób w toalecie? Przeszkadza/ nie mam zdania/ nie przeszkadza.
E) Co najbardziej przeszkadza Tobie w korzystaniu z toalet publicznych? (zaznacz max. 3 odp.) Brak papieru / brak mydła / brak ręczników papierowych, suszarki / brak zamka w drzwiach / brak światła / brudna toaleta / tłok ludzi / mokra podłoga.
F) Który z poniższych problemów najczęściej występuje w toaletach publicznych? (zaznacz max. 3 odp.) Brak papieru / brak mydła / brak ręczników papierowych, suszarki / brak zamka w drzwiach / brak światła / brudna toaleta / tłok ludzi / mokra podłoga.

5. Sposób wygenerowania wyników ankiety.
   W tym badaniu nie wykonujemy losowania indywidualnych jednostek badania.
   Do pytania A generujemy wyniki za pomocą rozkładu dwumianowego. Do pozostałych pytań generujemy wyniki za pomocą rozkładu wielomianowego.

- Podsumowanie:

Zdefiniowany cel badania i zadane pytania pozwalają nam określić chęć do korzystania z toalet w zależności od ich warunków sanitarnych. Można także wywnioskować co należy zmienić, aby wizyta w lokalu z toaletą publiczną była przyjemniejsza. W tym celu należy zwrócić uwagę na wyniki ankiety, mówiącej o tym co najbardziej przeszkadza w korzystaniu z tej usługi oraz jakie problemy najczęściej występują.

Lista 3_4

Zadanie 1

- Polecenie:

W zadaniu mamy przeprowadzić symulacje, której celem jest porównanie prawdopodobieństwa pokrycia i długości przedziałów ufności Cloppera-Pearsona, Walda i dowolnego przedziału (wybraliśmy Wilsona). Uwzględniamy poziom ufności 0.95, różne rozmiary próby i różne wartości prawdopodobieństwa.

Teoria:

Przedział ufności daje informacje na ile możemy ufać danej własności. Pokazuje nam, że poszukiwana przez nas rzeczywista wartość mieści się w pewnym przedziale z założonym prawdopodobieństwem.

Prawdopodobieństwo pokrycia jest prawdopodobieństwem pokrycia docelowym, czyli tym, które staramy się osiągnąć, gdy wyprowadzamy metodę zapewniającą przedział ufności.

- Kod:

W kodzie korzystamy z funkcji wbudowanych:

> Binom.confint(x, n, conf.level = 0.95, methods = "exact") - uzyskuje przedział ufności dla prawdopodobieństwa dwumianowego, gdzie x – liczba sukcesów w eksperymencie dwumianowym, n – liczba realizacji, conf.level - poziom ufności stosowany w przedziale ufności, methods – metody stotoswane (exact - Cloppera-Pearsona, asymptotic – Walda, wilson - Wilsona).
> Rbinom(n, size, prob) - generuje rozkład dwumianowy o ‘n’ realizacjach, ‘size’ próbach oraz ‘prob’ prawdopodobienstwie sukcesu w próbie Bernoulliego.

Początkowo instalujemy potrzebne paczki.

```
install.packages('binom')
install.packages('tidyverse')

library(binom)
library(tidyverse)
```

Zaczynamy od **Metody Cloppera-Pearsona (exact)**. Ustalamy początkowe parametry: liczbę realizacji, liczbę prób Bernoulliego i prawdopodobieństwo sukcesu w próbie Bernoulliego (do wykresów ze stałym p) oraz listy prawdopodobienstwa i listy prób Bernoulliego (do wykresów ze zmiennymi). Korzystamy z rozkładu dwumianowego.

```
N=10000
n=100
p_0 = 0.5

prawd=seq(0.01,0.99,0.01)
nlist = seq(100, 1000, 10)
```

Następnie przechodzimy do przedstawienia prawdopodobieństwa pokrycia w zależności od prawdopodobieństwa sukcesu w próbie Bernoulliego.

```
 y1_1 <- as.numeric()
for (p in prawd){
        xk <- rbinom(N, n, prob=p)
                p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='exact')
                tmp1 <- sum(p>p_uf[,5] & p<p_uf[,6])/N
                        y1_1 <- append(y1_1, tmp1) }
```

Następujący fragment kodu przedstawia prawdopodobieństwo pokrycia w zależności od liczby prób Bernoulliego.

```
y1_2 <- as.numeric()
for (n in nlist){
                xk <- rbinom(N, n, prob=p_0)
                        p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='exact')
                        tmp2 <- sum(p_0>p_uf[,5] & p_0<p_uf[,6])/N
                        y1_2 <- append(y1_2, tmp2)}
```

Poniższy fragment kodu przedstawia średnią długość przedziału ufności w zależności od prawdopodobieństwa sukcesu w próbie Bernoullego.

```
y1_3 <- as.numeric()
for (p in prawd){
        tmp3 <- as.numeric()
                xk <- rbinom(N, n, prob=p)
                   p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='exact')
                        l = length(p_uf[,5])
        for (i in 1:l){
                if (p>p_uf[i,5] & p<p_uf[i,6]){tmp3 <- append(tmp3, abs(p_uf[i,5] - p_uf[i,6]))}}
                sr <- mean(tmp3)
                        y1_3 <- append(y1_3, sr)}
```

Kolejnym zadaniem jest napisać kod na średnią długość przedziału ufności w zależności od liczby prób Bernoulliego.

```
y1_4 <- as.numeric()
for (n in nlist){
                tmp4 <- as.numeric()
                xk <- rbinom(N, n, prob=p_0) #N realizacji rozkladu 2mianowego
                p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='exact') #przedzial ufnosci
                        l = length(p_uf[,5])
        for (i in 1:l){
                if (p_0>p_uf[i,5] & p_0<p_uf[i,6]){tmp4 <- append(tmp4, abs(p_uf[i,5] - p_uf[i,6]))}}
                sr <- mean(tmp4)
                        y1_4 <- append(y1_4, sr)}
```

Przechodzimy do **Metody Walda** (asymptotic). Poniższy kod przedstawia prawdopodobieństwo pokrycia w zależności od prawdopodobieństwa sukcesu w próbie Bernoulliego. Tu również korzystamy z rozkładu dwumianowego.

```
y2_1 <- as.numeric()
for (p in prawd){
        xk <- rbinom(N, n, prob=p)
                p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='asymptotic')
                tmp1 <- sum(p>p_uf[,5] & p<p_uf[,6])/N
                        y2_1 <- append(y2_1, tmp1) }
```

Następnie szukamy rozwiązania dla prawdopodobieństwa pokrycia w zależności od liczby prób Bernoulliego.

```
y2_2 <- as.numeric()
for (n in nlist){
                xk <- rbinom(N, n, prob=p_0)
                        p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='asymptotic')
                        tmp2 <- sum(p_0>p_uf[,5] & p_0<p_uf[,6])/N
                        y2_2 <- append(y2_2, tmp2) }
```

Przechodzimy do znajdywania średniej długości przedziału ufności w zależności od prawdopodobieństwa sukcesu w próbie Bernoulliego.

```
y2_3 <- as.numeric()
for (p in prawd){
        tmp3 <- as.numeric()
                xk <- rbinom(N, n, prob=p) #N realizacji rozkladu 2mianowego
                        p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='asymptotic')
                        l = length(p_uf[,5])
        for (i in 1:l){
                if (p>p_uf[i,5] & p<p_uf[i,6]){tmp3 <- append(tmp3, abs(p_uf[i,5] - p_uf[i,6]))}}
                sr <- mean(tmp3)
                        y2_3 <- append(y2_3, sr)}
```

Poniżej znajduje się kod do średniej długości przedziału ufności w zależności od liczby prób Bernoullego.

```
y2_4 <- as.numeric()
for (n in nlist){
          tmp4 <- as.numeric()
                xk <- rbinom(N, n, prob=p_0) #N realizacji rozkladu 2mianowego
```

```
                p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='asymptotic')
                    l = length(p_uf[,5])
for (i in 1:l){
        if (p_0>p_uf[i,5] & p_0<p_uf[i,6]){tmp4 <- append(tmp4, abs(p_uf[i,5] - p_uf[i,6]))}}
                sr <- mean(tmp4)
                y2_4 <- append(y2_4, sr)}
```

Ostatecznie robimy wszystko analogicznie dla **Metody Wilsona** (wilson). Zaczynając od prawdopodobieństwa pokrycia w zależności od prawdopodobieństwa sukcesu w próbie Bernoulliego.

```
y3_1 <- as.numeric()
for (p in prawd){
xk <- rbinom(N, n, prob=p)
p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='wilson')
tmp1 <- sum(p>p_uf[,5] & p<p_uf[,6])/N
y3_1 <- append(y3_1, tmp1)}
```

Przechodzimy do kodu na prawdopodobieństwo pokrycia w zależności od liczby prób Bernoulliego.

```
y3_2 <- as.numeric()
for (n in nlist){
xk <- rbinom(N, n, prob=p_0)
p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='wilson')
tmp2 <- sum(p_0>p_uf[,5] & p_0<p_uf[,6])/N
y3_2 <- append(y3_2, tmp2)}
```

Następnie liczymy średnią długość przedziału ufności w zależności od prawdopodobieństwa sukcesu w próbie Bernoulliego.

```
y3_3 <- as.numeric()
for (p in prawd){
        tmp3 <- as.numeric()
                xk <- rbinom(N, n, prob=p) #N realizacji rozkladu 2mianowego
                        p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='wilson')
                        l = length(p_uf[,5])
for (i in 1:l){
        if (p>p_uf[i,5] & p<p_uf[i,6]){tmp3 <- append(tmp3, abs(p_uf[i,5] - p_uf[i,6]))}}
        sr <- mean(tmp3)
                y3_3 <- append(y3_3, sr)}
```

Na koniec przedstawiamy kod na średnią długość przedziału ufności w zależności od liczby prób Bernoulliego.

```
y3_4 <- as.numeric()
for (n in nlist){
                tmp4 <- as.numeric()
                xk <- rbinom(N, n, prob=p_0)
                        p_uf <- binom.confint(xk, n, conf.level = 0.95, methods='wilson')
                                l = length(p_uf[,5])
for (i in 1:l){
        if (p_0>p_uf[i,5] & p_0<p_uf[i,6]){tmp4 <- append(tmp4, abs(p_uf[i,5] - p_uf[i,6]))}}
                sr <- mean(tmp4)
                        y3_4 <- append(y3_4, sr)}
```

Z poniższych kodów tworzymy wykresy dla w.w. metod.

Wykres 1 - Wykres prawdopodobieństwa pokrycia w zależności od prawdopodobieństwa sukcesu w próbie Bernoulliego.

```
plot(prawd, y1_1, ylim = c(0.93,1), type = 'l', col = "black", main='Wykres prawdopodobieństwa
        pokrycia w zależności p', xlab='p - Prawdopodobieństwo sukcesu w próbie Bernoulliego',
ylab='Prawdopodobieństwo pokrycia')

lines(prawd, y2_1, type = "l", col = "blue")
lines(prawd, y3_1, type = "l", col = "green")
abline(h=0.95, col='red')

legend("topright", legend = c("exact", "asymptotic", "wilson", "poziom 0.95"), col = c("black", "blue",
"green", "red"), lty = 1)
```

Wykres 2 - Wykres prawdopodobieństwa pokrycia w zależności od liczby prób Bernoulliego.

```
plot(nlist, y1_2, ylim = c(0.93,1), type = 'l', col = "black", main='Wykres prawdopodobieństwa
pokrycia w zależności od n', xlab='n - Liczba prób Bernoulliego', ylab='Prawdopodobieństwo
pokrycia')

lines(nlist, y2_2, type = "l", col = "blue")
lines(nlist, y3_2, type = "l", col = "green")
abline(h=0.95, col='red')

legend("topright", legend = c("exact", "asymptotic", "wilson", "poziom 0.95"), col = c("black", "blue",
"green", "red"), lty = 1)
```

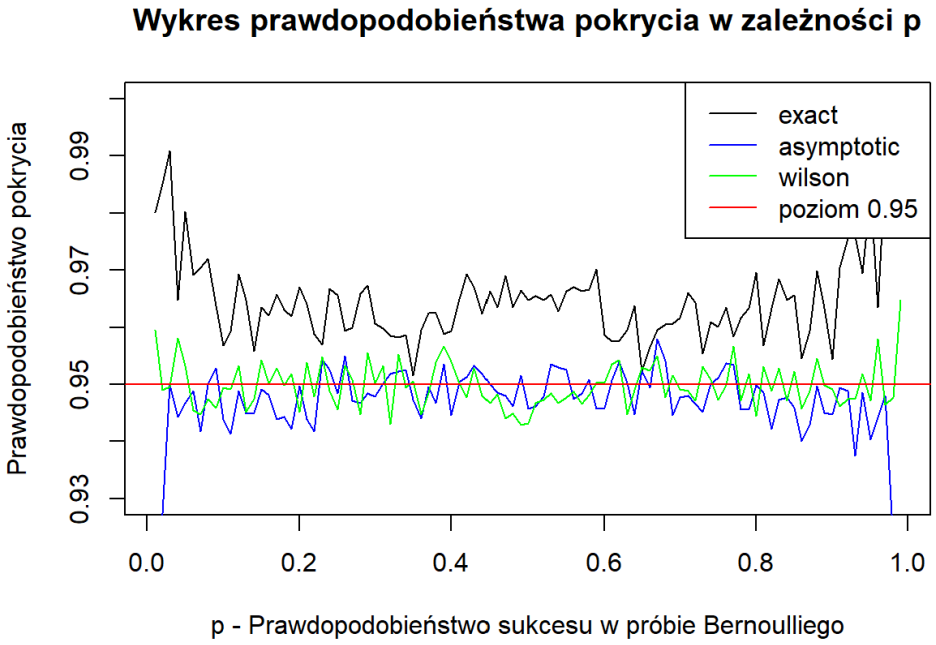
Wykres 3 - Wykres średniej długości przedziału ufności w zależności od prawdopodobieństwa sukcesu w próbie Bernoullego

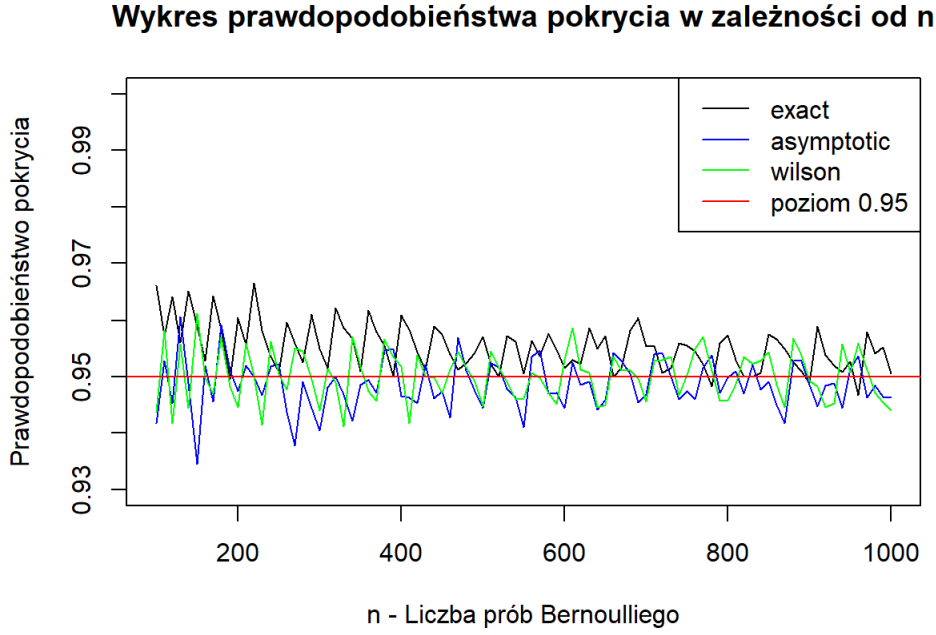
```
plot(prawd, y1_3, type = 'l', col = "black", main='Wykres średniej długości przedziału ufności w
zależności od p', xlab='p - Prawdopodobieństwo sukcesu w próbie Bernoulliego', ylab='Średnia
długość przedziału ufności')

lines(prawd, y2_3, type = "l", col = "blue")
lines(prawd, y3_3, type = "l", col = "green")

legend("topright", legend = c("exact", "asymptotic", "wilson"), col = c("black", "blue", "green"), lty =
1)
```

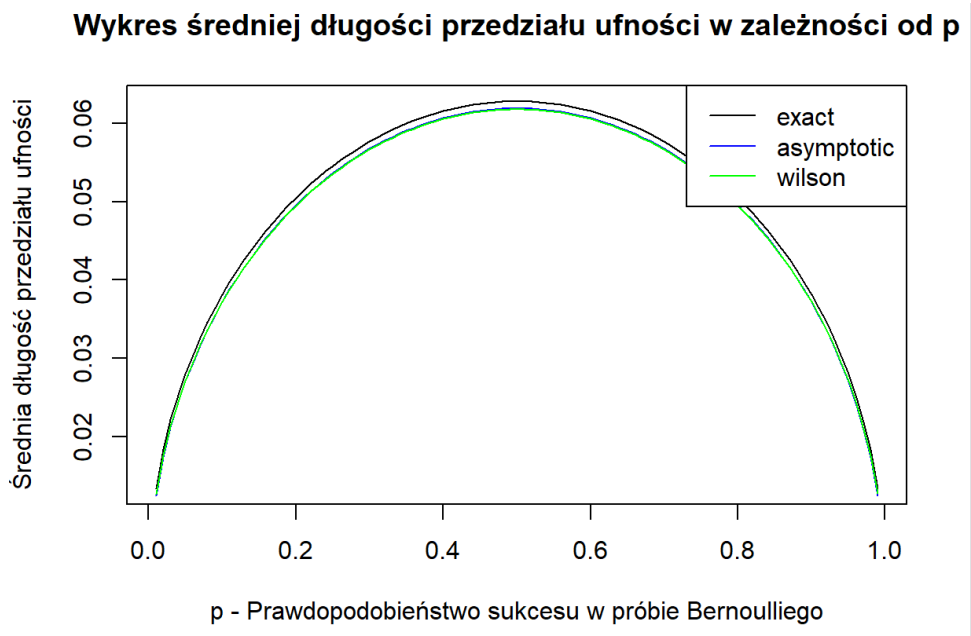
Wykres 4 - Wykres średniej długości przedziału ufności w zależności od liczby prób Bernoulliego

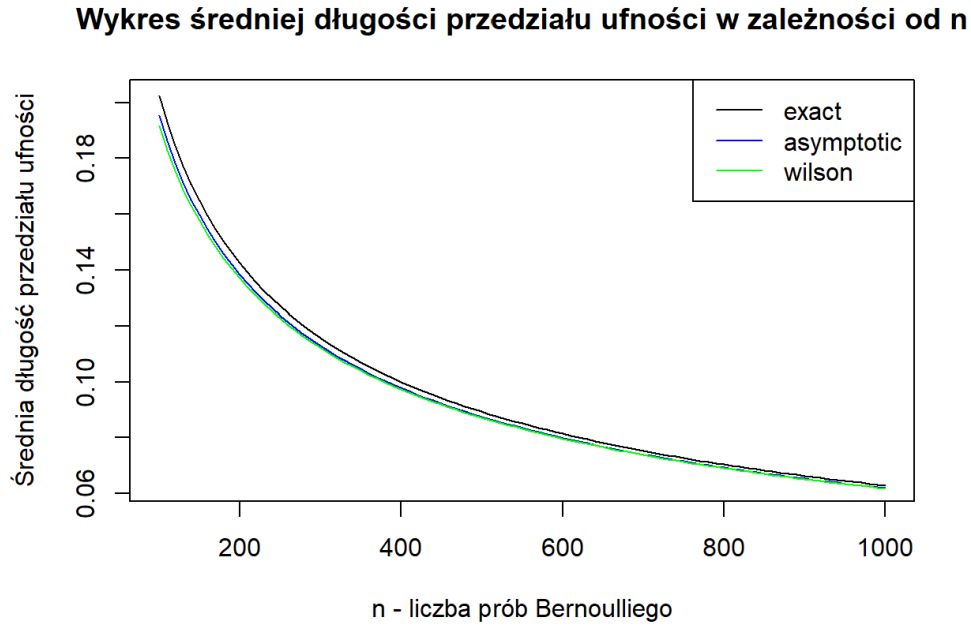
```
plot(nlist, y1_4, type = 'l', col = "black", main='Wykres średniej długości przedziału ufności w
zależności od n', xlab='n - liczba prób Bernoulliego', ylab='Średnia długość przedziału ufności')

lines(nlist, y2_4, type = "l", col = "blue")
lines(nlist, y3_4, type = "l", col = "green")

legend("topright", legend = c("exact", "asymptotic", "wilson"), col = c("black", "blue", "green"), lty =
1)
```

- Wyniki:


Wykres prawdopodobieństwa pokrycia w zależności p
Prawdopodobieństwo pokrycia
0.93
0.95
0.97
0.99
0.0
0.2
0.4
0.6
0.8
1.0
p - Prawdopodobieństwo sukcesu w próbie Bernoulliego
exact
asymptotic
wilson
poziom 0.95


Wykres prawdopodobieństwa pokrycia w zależności od n
Prawdopodobieństwo pokrycia
0.93
0.95
0.97
0.99
200
400
600
800
1000
n - Liczba prób Bernoulliego
exact
asymptotic
wilson
poziom 0.95





- Podsumowanie:

Patrząc na dwa ostatnie wykresy (średniej długości przedziału ufności) można stwierdzić, że metoda Wilsona jest najlepsza, metoda Cloppera-Pearsona wypada najdalej od prawidłowych wyników, natomiast metoda Walda jest bardzo przybliżona do metody Wilsona i prawie jej dorównuje. Interpretując pozostałe dwa wykresy (prawdopodobieństwa pokrycia) można stwierdzić to samo.

Lista 3_4

Zadanie 2

- Polecenie:

Na podstawie danych z pliku Reakcja.csv trzeba było wyznaczyć przedziały ufności (na poziomie ufności 0.95) dla prawdopodobieństwa wystąpienia poprawy stanu zdrowia w różnych podgrupach ze względu na wielkość dawki. Należało skorzystać z funkcji binom.confint oraz wyznaczyć te przedziały na kilka możliwych sposobów. Następnie porównać te przedziały i wybrać najlepszą metodę.

- Kod:

Podłączamy biblioteki:

```
library (tidyverse)

library (binom)
```

Ładujemy dane, grupujemy je według wielkości dawki oraz zliczamy ilość sukcesów (popraw zdrowia) dla każdej grupy.

```
dane <- read.csv2("Reakcja.csv")

groups <- dane %>% group_by(Dawka) %>% count(Reakcja)

glimpse(groups)
```

Zapisujemy policzone sukcesy w poszczególne zmienne. Tworzymy zmienną z ogólną ilością sukcesów dla porównania.

```
s1 <- c(groups[2,3])[[1]]
s2 <- c(groups[4,3])[[1]]
s3 <- c(groups[6,3])[[1]]
s4 <- c(groups[8,3])[[1]]
s5 <- c(groups[10,3])[[1]]

soverall <- s1+s2+s3+s4+s5
glimpse(soverall)
```

Dalej kod będzie podawany tylko dla pierwszej grupy, ponieważ kod dla pozostałych grup jest taki sam, tylko ze zmianą jednego parametru.

Obliczamy przedział ufności dla poszczególnych grup za pomocą metody Cloppera-Pearsona.

```
p_uf1a <- binom.confint(s1, 40, conf.level = 0.95, methods='exact')
glimpse(p_uf1a[,5:6])
```

Obliczamy przedział ufności dla poszczególnych grup za pomocą metody Walda (asymptotycznej).

```
p_uf1b <- binom.confint(s1, 40, conf.level = 0.95, methods='asymptotic')
glimpse(p_uf1b[,5:6])
```

Obliczamy przedział ufności dla poszczególnych grup za pomocą metody Wilsona.

```
p_uf1c <- binom.confint(s1, 40, conf.level = 0.95, methods='wilson')
glimpse(p_uf1c[,5:6])
```

- Wyniki:

W tabeli poniżej są zamieszczone wyniki. Grupy są oznaczane wielkością dawki. Ostatni wiersz był liczony dla całej statystyki. W poszczególnych kratkach znajdują się przedziały ufności dla odpowiedniej grupy wyliczone odpowiednią metodą.

Grupa	Cloppera-Pearsona	Walda	Wilsona
-3.204	0.016, 0.204	-0.007, 0.157	0.025, 0.199
-2.903	0.091, 0.356	0.076, 0.324	0.105, 0.348
-2.602	0.091, 0.356	0.076, 0.324	0.105, 0.348
-2.301	0.227, 0.542	0.225, 0.525	0.242, 0.530
-2	0.315, 0.639	0.320, 0.630	0.329, 0.625
Wszystkie Razem	0.205, 0.332	0.204, 0.326	0.209, 0.330

W tabeli poniżej są długości odpowiednich przedziałów z poprzedniej tabeli.

Grupa	Cloppera-Pearsona	Walda	Wilsona
-3.204	0.188	0.164	0.174
-2.903	0.265	0.248	0.243
-2.602	0.265	0.248	0.243
-2.301	0.315	0.300	0.288
-2	0.324	0.310	0.296
Wszystkie Razem	0.127	0.122	0.121

- Podsumowanie:

Znaleźliśmy przedziały ufności prawdopodobieństwa nastąpienia poprawy w zależności od wielkości dawki. Widać z naszych wyników, że im większa jest dawka, tym większa jest szansa na poprawienie się zdrowia pacjenta. Możemy też porównać znalezione przydziały różnymi metodami i wywnioskować na przykład, że metoda Walda jest lepsza od metody Cloppera-Pearsona ze względu na długość przedziału. Metoda Wilsona jest średnia pod tym względem jeśli patrzeć na najmniejszą dawkę. Jednak we wszystkich pozostałych grupach ta metoda jest najlepsza spośród trzech.

Lista 5

Zadanie 1

- Polecenie:

Zapoznać się z funkcjami binom.test oraz prop.test.

- Treść:

Obydwie funkcje służą do testowania hipotez statystycznych. Funkcja binom.test testuje hipotezy o prawdopodobieństwie w rozkładzie Bernoulli. Jako argumenty przyjmuje: liczbę sukcesów lub listę sukcesów i porażek, ilość prób (niepotrzebna jeśli w pierwszym argumencie jest lista), hipotezę o prawdopodobieństwie sukcesu, hipotezy alternatywne, poziom ufności. Tylko pierwsze dwa argumenty są niezbędne do uruchomienia funkcji, pozostałe mają domyślne wartości. Funkcja zwraca dużo różnych wartości. Najważniejszymi spośród nich są p-wartość, przedział ufności oraz wyestymowane prawdopodobieństwo. Na podstawie tych wyników podejmujemy decyzję o przyjęciu lub odrzuceniu hipotezy zerowej. Na przykład możemy odrzucać hipotezę zerową, kiedy p-wartość jest mniejsza od 0.05.

Funkcja prop.test służy do testowania czy zgadzają się prawdopodobieństwa. Mogą tutaj występować dwa przypadki. Pierwszy: kiedy hipotezą zerową jest jakieś prawdopodobieństwo w rozkładzie i

testujemy czy nasza realizacja zmiennej losowej lub statystyka odpowiada temu prawdopodobieństwu. Drugi: kiedy porównujemy prawdopodobieństwa w dwóch różnych rozkładach. Jako argumenty przyjmuje: wektor liczb sukcesów lub dwuwymiarowy wektor z porażkami i sukcesami, wektor liczb prób (niepotrzebny jeśli w pierwszym argumencie mamy wektor dwuwymiarowy), wektor prawdopodobieństw sukcesu, hipotezy alternatywne, poziom ufności oraz argument correct, na którym nie będziemy skupiali się. Zwraca data-frame z wartościami. Najważniejszymi spośród nich są: wartość statystyki testu Pearsona chi-kwadrat, stopień swobody rozkładu chi-kwadrat, p-wartość, przedziały ufności. Jeśli p jest NULL i jest więcej niż jedna grupa podana jako argument, to funkcja testuje zgodność prawdopodobieństw sukcesu dla podanych grup. W przeciwnym przypadku testuje, czy odpowiada prawdopodobieństwo próby temu z hipotezy. Tylko pierwsze dwa argumenty są niezbędne, pozostałe mają wartości domyślne. Na podstawie wyników z tej funkcji możemy podejmować decyzję o przyjęciu lub odrzuceniu hipotezy zerowej. Zwykle robi się to na podstawie wartości chi-kwadrat lub p-wartości.

- Podsumowanie:

Funkcję binom.test warto używać kiedy mamy jeden zestaw parametrów i chcemy przetestować prostą hipotezę o prawdopodobieństwie rozkładu. Funkcję prop.test warto używać kiedy mamy więcej niż jeden zestaw parametrów lub kiedy chcemy przetestować zgodność prawdopodobieństw między zestawami. Przykłady użycia tych funkcji będą podane w zadaniu 2.

Lista 5

Zadanie 2

- Polecenie:

Na podstawie danych z pliku Reakcja.csv trzeba przetestować 4 hipotezy zerowe: a) prawdopodobieństwo poprawy stanu zdrowia pacjenta leczonego w domu najmniejszą dawką leku jest mniejsze bądź równe ½, b) prawdopodobieństwo poprawy stanu zdrowia pacjenta leczonego w domu najmniejszą dawką leku jest równe prawdopodobieństwu poprawy stanu zdrowia pacjenta leczonego najmniejszą dawka leku, ale w szpitalu, oraz c) to samo, tylko dla pacjenta leczonego największą dawką leku. W każdym teście należy podać p-wartość i sformułować odpowiedź.

- Kod:

Wczytujemy dane z pliku .csv, grupujemy je według wielkości dawki oraz miejsca leczenia się. Obliczamy ilość popraw w każdym szczególnym przypadku.

```
dane <- read.csv2("Reakcja.csv")
groups <- dane %>% group_by(Dawka) %>% count(Reakcja, Miejsce)
glimpse(groups)
```

Bierzemy odpowiednie ilości sukcesów i dokonujemy odpowiednich testów.

```
na1 <- 0
testa1 <- binom.test(na1, 20, alternative="l")
glimpse(testa1)

na2 <- c(groups[18,4])[[1]]
glimpse(na2)

testa2 <- binom.test(na2,20, alternative="l")
glimpse(testa2)
```

```
nb1 <- c(groups[3,4])[[1]]
glimpse(nb1)

testb1 <- prop.test(c(na1,nb1),c(20,20))
glimpse(testb1)

nb2 <- c(groups[19,4])[[1]]
glimpse(nb2)

testb2 <- prop.test(c(na2,nb2),c(20,20))
glimpse(testb2)
```

- Wyniki:

Test a)1 (najmniejsza dawka): p-wartość = 9.54e-07

Test a)2 (największa dawka): p-wartość = 0.0577

Test b)1 (najmniejsza dawka): p-wartość = 0.23, chi2 = 1.44

Test b)2 (największa dawka): p-wartość = 0.0575, chi2 = 3.61

- Podsumowanie:

Ponieważ funkcja binom.test przyjmuje jako hipotezę alternatywną to, że prawdopodobieństwo jest mniejsze od argumentu, otrzymana p-wartość (<<0.05) w teście a)1 daje nam podstawę do stwierdzenia, że prawdopodobieństwo poprawy stanu zdrowia pacjenta leczonego w domu najmniejszą dawką leku jest mniejsze niż ½. W teście a)2 dostaliśmy p-wartość > 0.05, co nie pozwala nam na odrzucenie hipotezy zerowej. W testach b)1 oraz b)2 p-wartości też nie pozwalają nam na odrzucenie hipotezy zerowej. To samo widać z wartości chi-kwadrat. Porównując między sobą testy dla najmniejszej i największej dawki możemy wywnioskować, że leczenie największą dawkę jest skuteczniejsze, ponieważ otrzymane p-wartości były znacznie bliżej do odrzucenia hipotezy zerowej niż dla najmniejszej dawki.