Aleksander Jakóbczyk i Kacper Pasterniak

Sprawozdanie 1

Lista 1

Zad 1

Sporządzić tablice liczności dla zmiennych *Temperature* oraz *Preference* biorąc pod uwagę wszystkie dane, jak również w podgrupach ze względu na zmienną *Water Softness*:

Detergent

```
Detergent.df <- data.frame(Detergent)
Detergent.df %>% group_by(Temperature) %>% summarise(n = sum(Freq))
```

```
## # A tibble: 2 x 2
##   Temperature     n
##   <fct>       <dbl>
## 1 High          369
## 2 Low           639
```

```
Detergent.df %>% filter(Water_softness == "Soft") %>%
  group_by(Temperature) %>% summarise(n = sum(Freq))
```

```
## # A tibble: 2 x 2
##   Temperature     n
##   <fct>       <dbl>
## 1 High          104
## 2 Low           222
```

```
Detergent.df %>% filter(Water_softness == "Medium") %>%
  group_by(Temperature) %>% summarise(n = sum(Freq))
```

```
## # A tibble: 2 x 2
##   Temperature     n
##   <fct>       <dbl>
## 1 High          126
## 2 Low           218
```

```
Detergent.df %>% filter(Water_softness == "Hard") %>%
  group_by(Temperature) %>% summarise(n = sum(Freq))
```

```
## # A tibble: 2 x 2
##   Temperature     n
##   <fct>       <dbl>
## 1 High          139
## 2 Low           199
```

Preference

```
# Detergent.df %>% group_by(Preference) %>% summarise(n = sum(Freq))
Detergent.df %>% filter(Water_softness == "Soft") %>%
  group_by(Preference) %>% summarise(n = sum(Freq))
```

```
## # A tibble: 2 x 2
##   Preference     n
##   <fct>      <dbl>
## 1 Brand X      168
## 2 Brand M      158
```

```
Detergent.df %>% filter(Water_softness == "Medium") %>%
  group_by(Preference) %>% summarise(n = sum(Freq))
```

```
## # A tibble: 2 x 2
##   Preference     n
##   <fct>      <dbl>
## 1 Brand X      169
## 2 Brand M      175
```

```
Detergent.df %>% filter(Water_softness == "Hard") %>%
  group_by(Preference) %>% summarise(n = sum(Freq))
```

```
## # A tibble: 2 x 2
##   Preference     n
##   <fct>      <dbl>
## 1 Brand X      171
## 2 Brand M      167
```

Zad 2

Sporządzić tabelę wielodzielczą uwzględniającą zmienną *Temperature* i *Water Softness*:

```
ftable(Detergent, col.vars = "Temperature", row.vars = "Water_softness")
```

```
##                 Temperature High Low
## Water_softness
## Soft                          104 222
## Medium                        126 218
## Hard                          139 199
```

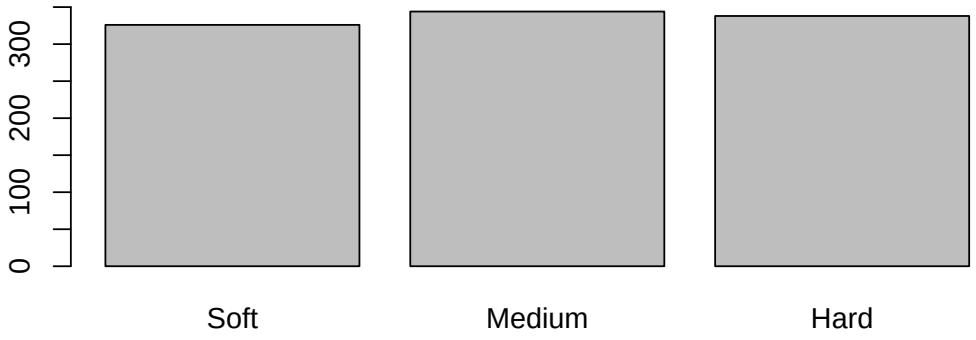
```
structable(Temperature ~ Water_softness, Detergent) %>% addmargins()
```

```
##                Temperature
## Water_softness High Low  Sum
##          Soft    104 222  326
##          Medium  126 218  344
##          Hard    139 199  338
##          Sum     369 639 1008
```

Zad 3

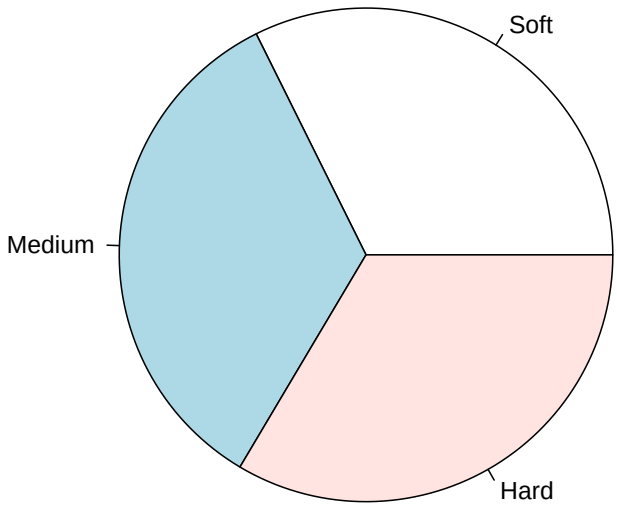
Sporządzić wykres kołowy i słupkowy dla zmiennej *Water Softness*:

```
par(mar = c(2, 2, 0.5, 1))
A <- apply(Detergent, "Water_softness", sum)
barplot(A,ylim=c(0,350))
```



Rysunek 1. Wykres słupkowy dla zmiennej *Water Softness*.

```
par(mar = c(0,0,0,0))
pie(A)
```

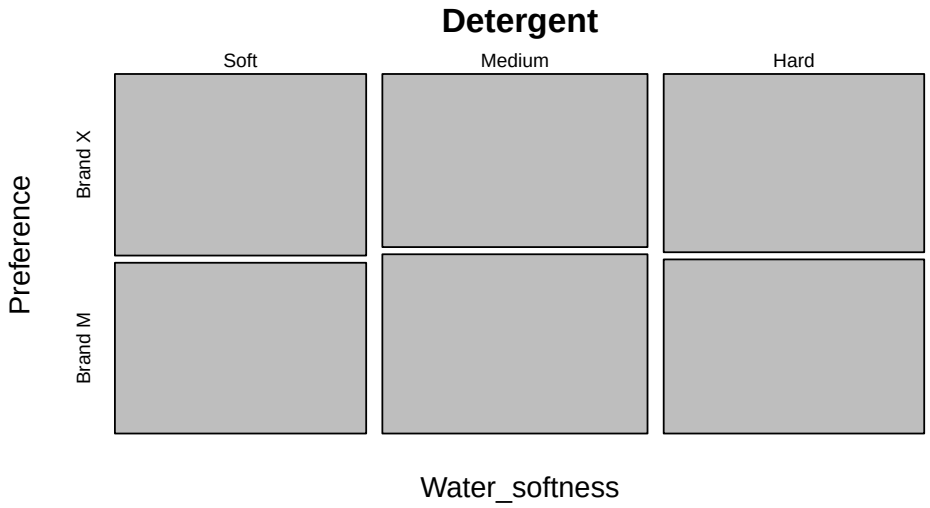


Rysunek 2. Wykresy kołowy dla zmiennej *Water Softness*.

Zad 4

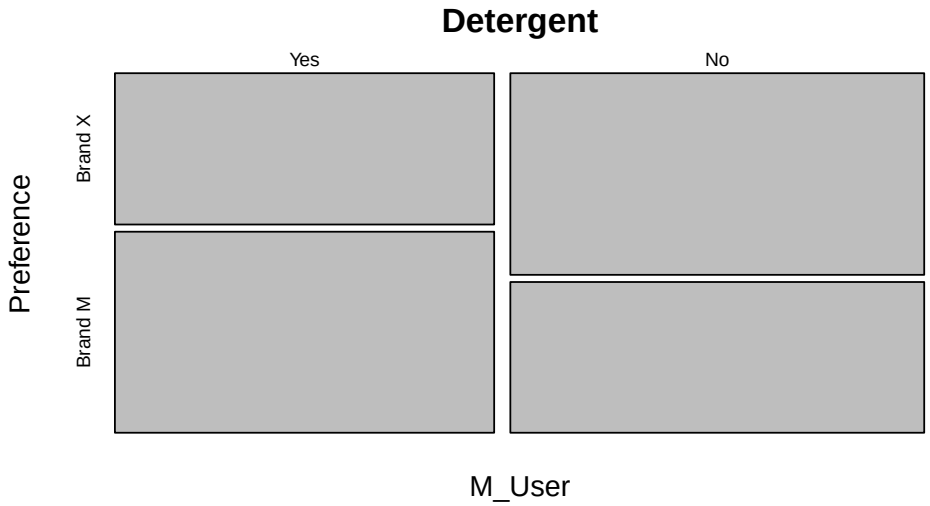
Sporządzić wykresy mozaikowe odpowiadające rozpatrywanym danym.

```
par(mar = c(2, 2, 2, 2))
mosaicplot(~Water_softness+Preference, data = Detergent)
```



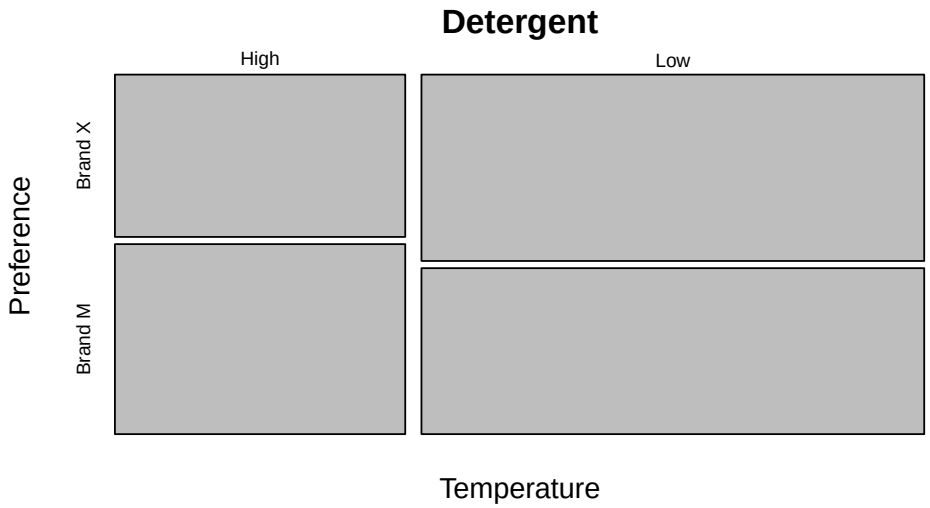
Rysunek 3. Wykres mozaikowy dla *Preference* i *Water softness*

```
par(mar = c(2, 2, 2, 2))
mosaicplot(~M_User+Preference, data = Detergent)
```



Rysunek 4. Wykres mozaikowy dla *Preference* i *M User*.

```r
par(mar = c(2, 2, 2, 2))
mosaicplot(~Temperature+Preference, data = Detergent)
```



Rysunek 5. Wykres mozaikowy dla *Preference* i *Temperature*.

Wykresy mozaikowe są bardzo proste w analizie. Wykres tworzy nam prostokąty, dzięki czemu łatwo zauważyć które kombinacje zmiennych są najliczniejsze. Wysokość takiego prostokąta odpowiada liczności zmiennej na osi OX, a szerokość odpowiada liczności zmiennej na osi OY.

Lista 2

Zad 1

Zapoznać się z funkcją *sample* (w pakiecie *stats*). Napisać fragment programu, którego celem jest wylosowanie próbki rozmiaru około 1/10 liczby przypadków danej bazy danych (pewnej hipotetycznej), ze zwracaniem oraz bez zwracania.

Będziemy korzystać z funkcji *sample*. Wylosujemy elementy z bazy danych o nazwie *mtcars*. Jest to zbiór informacji na temat specyfikacji różnych samochodów, poniżej widzimy pięć pierwszych wierszy:

```r
head(mtcars,5)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

Losowanie ze zwracaniem:

```
ind <- sample(x=nrow(mtcars),size=nrow(mtcars)/10,replace=TRUE)
```

Wylosowane indeksy:

```
ind
```

```
## [1] 25 21  2
```

Wylosowane elementy z bazy danych:

```
mtcars[ind,]
```

```
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Toyota Corona    21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Mazda RX4 Wag    21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
```

Losowanie bez zwracania:

```
ind <- sample(x=nrow(mtcars),size=nrow(mtcars)/10,replace=FALSE)
```

Wylosowane indeksy:

```
ind
```

```
## [1]  9  6 30
```

Wylosowane elementy z bazy danych:

```
mtcars[ind,]
```

```
##                mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## Merc 230      22.8   4 140.8  95 3.92 3.15 22.90  1  0    4    2
## Valiant       18.1   6 225.0 105 2.76 3.46 20.22  1  0    3    1
## Ferrari Dino 19.7   6 145.0 175 3.62 2.77 15.50  0  1    5    6
```

Zad 2

Zaproponować algorytm generowania liczb z rozkładu dwumianowego i udowodnić, że jest poprawny. Napisać program do generowania tych liczb zgodnie z zaproponowanym algorytmem. (W pakiecie R dostępna jest funkcja rbinom.)

Propozycja algorytmu:

1. Generujemy wektor zer o rozmiarze n .
2. Dla każdego elementu tego wektora losujemy u z rozkładu jednostajnego U(0,1), jeśli $u \leqslant p$ to dodajemy 1 do tego elementu.
3. Krok 2 powtarzamy N razy.

Algorytm opisany za pomocą funkcji w R:

```
bin <- function(n,p,N){
  X <- rep(0,N)
  for (i in 1:N) {
    r = sum(runif(n) < p)
    X[i] = r
  }
  return(X)
}
```

gdzie: n - rozmiar próby, p - prawdopodobieństwo sukcesu, N - ilość wywołań

Przykładowe użycie:

```
bin(10,0.4,5)
```

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

Sprawdzenie poprawności:

Dla zmiennej losowej $X \sim \mathcal{B}(n,p)$ wiemy, że:

$\mathbb{E}(X) = np$,

$\mathrm{Var}(X) = np(1-p)$,

Sprawdźmy zatem działanie naszej funkcji dla $n = 100$ i $p = 0.4$:

```
test <- bin(100,0.4,10000)
mean(test)
```

```
## [1] 39.9916
```

```
var(test)
```

```
## [1] 23.96893
```

Wartości teoretyczne średniej i wariancji dla takich parametrów powinny wynosić kolejno 40 i 24. Nasze wyniki są bardzo bliskie co wskazuje na poprawność metody.

Zad 3

Zaproponować algorytm generowania wektora z rozkładu wielomianowego i udowodnić, że jest poprawny. Napisać program do generowania tych wektorów zgodnie z zaproponowanym algorytmem. (W pakiecie R dostępna jest funkcja *rmultinom*.)

Propozycja algorytmu:
Chcemy generować zmienna losowa z rozkładu wielomianowego o parametrach n i p, gdzie p jest wektorem wag prawdopodobieństw o długości k którego elementy sumują się do jedynki.

1. Generujemy wektor zer o długości k.
2. Generuj wektor prób o długości n, przy czym w każdej próbie mamy do czynienia z wylosowaniem jednego z k zdarzeń o poszczególnym prawdopodobieństwem.
3. Sumuj ilość występowania każdego zdarzenia i zapisz je do wektora.
4. Krok 1 i 3 powtarzamy N razy.

Algorytm opisany za pomocą funkcji w R:

```
multinom.rv <-function(n, p, N){
  k <- length(p)
  X <- matrix(0, nrow = k, ncol = N)
  for (j in 1:N) {
    ind <- sample(1:k, n, replace = TRUE, prob = p)
    for (i in 1:n) {
      X[ind[i],j] = 1 + X[ind[i],j]
    }
  }
  return(X)
}
```

Przykładowe użycie:

```
multinom.rv(10,c(0.2,0.3,0.5),5)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    0    3    2    2
## [2,]    3    5    0    3    4
## [3,]    5    5    7    5    4
```

Sprawdzenie poprawności:
Niech zmienne losowe $X_1, X_2, \ldots, X_k$ oznaczają liczby zajść poszczególnych zdarzeń w n próbach, przy czym $X_1 + X_2 + \cdots + X_k = n$. Dla zmiennej losowej $X \sim \mathcal{W}(n, \{p_1, p_2, \ldots, p_k\})$ wiemy, że:
$\mathbb{E}(X_i) = np_i$.
$\mathrm{Var}(X_i) = np_i(1 - p_i)$.
Sprawdźmy zatem działanie naszej funkcji dla $n = 100$ i $p = \{0.2, 0.3, 0.5\}$:

```r
test <- multinom.rv(100,c(0.2,0.3,0.5),10000)
rowMeans(test)
```

```
## [1] 20.0341 29.9371 50.0288
```

Widzimy zatem że symulowane wartości są bardzo bliskie wartości teoretycznych: $20, 30, 50$.

```r
rowVars(test)
```

```
## [1] 16.03394 20.68761 25.20669
```

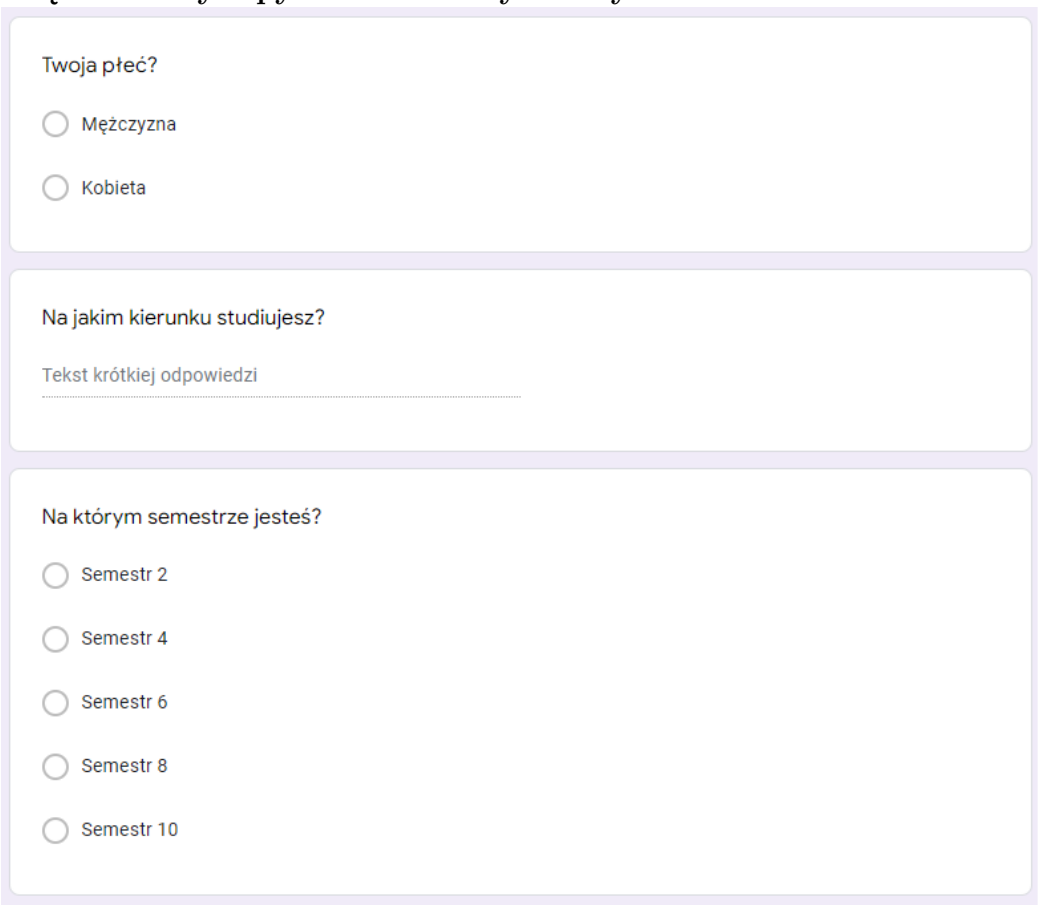
Widzimy zatem że symulowane wartości są bardzo bliskie wartości teoretycznych: $16, 21, 25$.
W obu powyższych przypadkach nasze wyniki są bardzo bliskie co wskazuje na poprawność metody.

Zad 4

Propozycja badania ankietowego

Celem badania będzie zebranie informacji na temat zorganizowanego wcześniej Webinaru wydziałowego. Będziemy chcieli dowiedzieć się jak wydarzenie zostało odebrane przez uczestników. Naszą grupą docelową stanowią oczywiście studenci, którzy brali udział w naszym Webinarze. Warunkiem przystąpienia do wydarzenia było wypełnienie formularza zgłoszeniowego, w którym studenci musieli podać adresy e-mail. Dzięki temu posiadamy adres e-mail każdego uczestnika, więc łatwo możemy wysłać im link do ankiety. Poniżej znajduję się fragment przykładowego kwestionariusza:

Część ankiety z pytaniami metryczkowymi:



Przykładowe pytania:

Pytanie:	Bardzo źle	Źle	Średnio	Dobrze	Bardzo dobrze
Czas trwania webinaru					
Przydatność wiedzy zdobytej podczas webinaru					
Przygotowanie merytoryczne prowadzących					
Obsługa techniczna wydarzenia					

Taką ankietę możemy stworzyć za pomocą darmowych narzędzi dostępnych w internecie, na przykład Formularze Google.

Lista 3

Zad 1

Przeprowadzić symulacje, których celem jest porównanie prawdopodobieństwa pokrycia i długości przedziałów ufności Cloppera-Pearsona, Walda i trzeciego dowolnego typu przedziału ufności zaimplementowanego w funkcji *binom.confint* pakietu *binom*. Uwzględnić poziom ufności 0.95, różne rozmiary próby i różne wartości prawdopodobieństwa p. Wyniki zamieścić w tabelach i na rysunkach. Sformułować wnioski, które umożliwią praktykowi wybór konkretnego przedziału ufności do wyznaczenia jego realizacji dla konkretnych danych.

W symulacji wykorzystamy przedziały ufności Cloppera-Pearsona, Walda i Asymptotyczne.Jako trzecią metodę wybraliśmy test asymptotyczny, ponieważ znacząco różni się od dwóch pozostałych. Wykorzystuje on Centralne Twierdzenie Graniczne, więc powinien bardzo dobrze działać dla próbki o dużym rozmiarze. Symulacje przeprowadzimy na podstawie realizacji zmiennej losowej: $X \sim \mathcal{B}(n, p)$. Wyznaczymy prawdopodobieństwa pokrycia oraz długości przedziałów ufności, a wykorzystamy do tego symulacje Monte Carlo.

Poniżej znajduje się kod z symulacją:

```
symulation <-function(n = 10, dp= 0.2, MCs = 1000){
  ps <- seq(0.01, 0.99, dp)
  N <- length(ps)
  data <- matrix(0,N,6)

  for (k in 1:N) {
    p <- ps[k]
    wilson_ok <- 0
    axact_ok  <- 0
    asymp_ok  <- 0
    wilson_l <- rep(0,MCs)
    axact_l  <- rep(0,MCs)
    asymp_l  <- rep(0,MCs)

    for (i in 1:MCs){
      x <- rbinom(1, n, p)
      wilson <- binom.wilson(x, n)
      exact  <- binom.exact(x, n)
      asymp  <- binom.asymp(x, n)
      wilson_l[i] <- wilson$upper - wilson$lower
      axact_l[i] <- exact$upper - exact$lower
```

```
      asymp_l[i] <- asymp$upper - asymp$lower

      if (wilson["lower"]<p && p < wilson["upper"]) {
        wilson_ok <- 1 + wilson_ok
      }
      if (exact["lower"]<p && p < exact["upper"]) {
        axact_ok <- 1 + axact_ok
      }
      if (asymp["lower"]<p && p < asymp["upper"]) {
        asymp_ok <- 1 + asymp_ok
      }
    }

    data[k,]<- c( wilson_ok/MCs, axact_ok/MCs, asymp_ok/MCs,
                  mean(wilson_l), mean(axact_l), mean(asymp_l))
  }
  return(data)
}
```

Symulacje przeprowadzamy dla $n = 10$ z krokiem równym $dp = 0.01$. Wyniki zapisujemy w pliku csv, dzięki czemu nie musimy za każdym razem czekać na wyniki.

```
data <- symulation(n = 10, dp = 0.01)
write.csv(df_l, "data_n_10.csv")
```

Stwórzmy teraz ramki danych prawdopodobieństw pokrycia:

```
data <- read.csv("data_n_10.csv")
ps <- seq(0.01, 0.99, 0.01)
df1 <- data.frame(wilsonp = data[,1], axact = data[,2],
                  asymp = data[,3], p = ps)
head(df1)
```

```
##   wilsonp axact asymp    p
## 1   0.895 0.997 0.105 0.01
## 2   0.980 0.980 0.194 0.02
## 3   0.967 0.997 0.259 0.03
## 4   0.952 0.996 0.313 0.04
## 5   0.922 0.991 0.369 0.05
## 6   0.990 0.990 0.498 0.06
```

Stwórzmy teraz ramki danych średniej długości przedziałów:

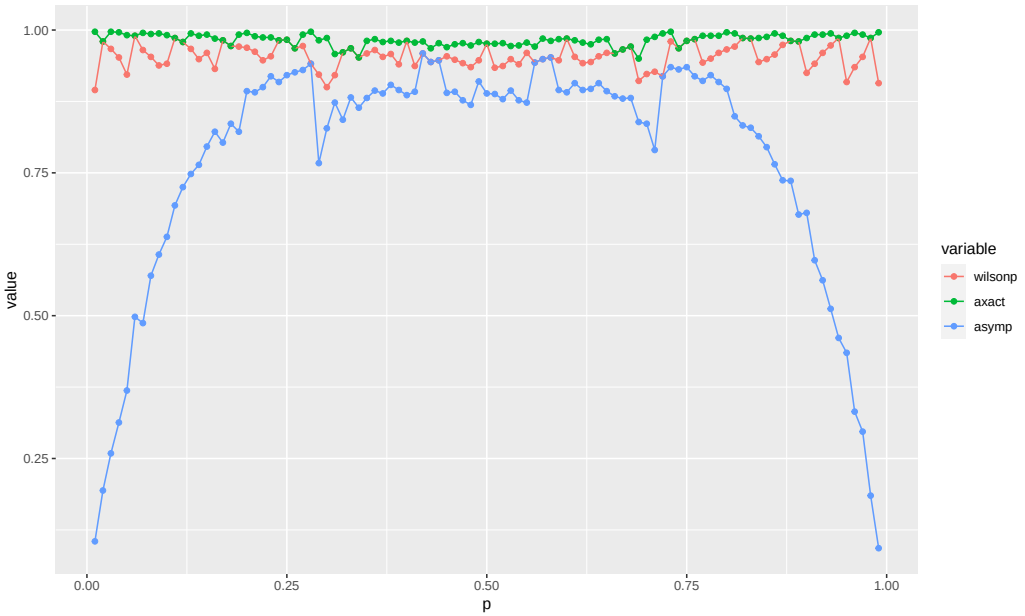
```
df2 <- data.frame(wilsonp = data[,4], axact = data[,5],
                  asymp = data[,6], p = ps)
head(df2)
```

```
##     wilsonp     axact      asymp    p
```

```
## 1 0.2891513 0.3228313 0.03941896 0.01
## 2 0.3000085 0.3363140 0.07469554 0.02
## 3 0.3080306 0.3462822 0.10062344 0.03
## 4 0.3152131 0.3552272 0.12345860 0.04
## 5 0.3233879 0.3654924 0.14795246 0.05
## 6 0.3416031 0.3883001 0.20384545 0.06
```

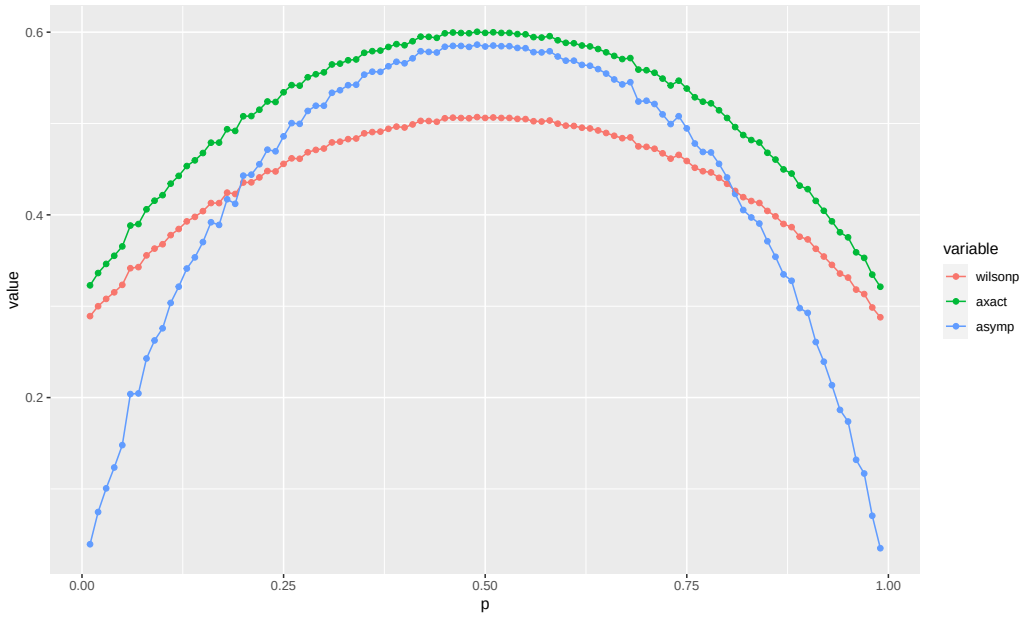
Sprawdzamy również wykresy:

```
df_p <- melt(df1,id.vars="p")
ggplot(df_p,aes(p, value,col=variable))+
  geom_point()+ geom_line()
```



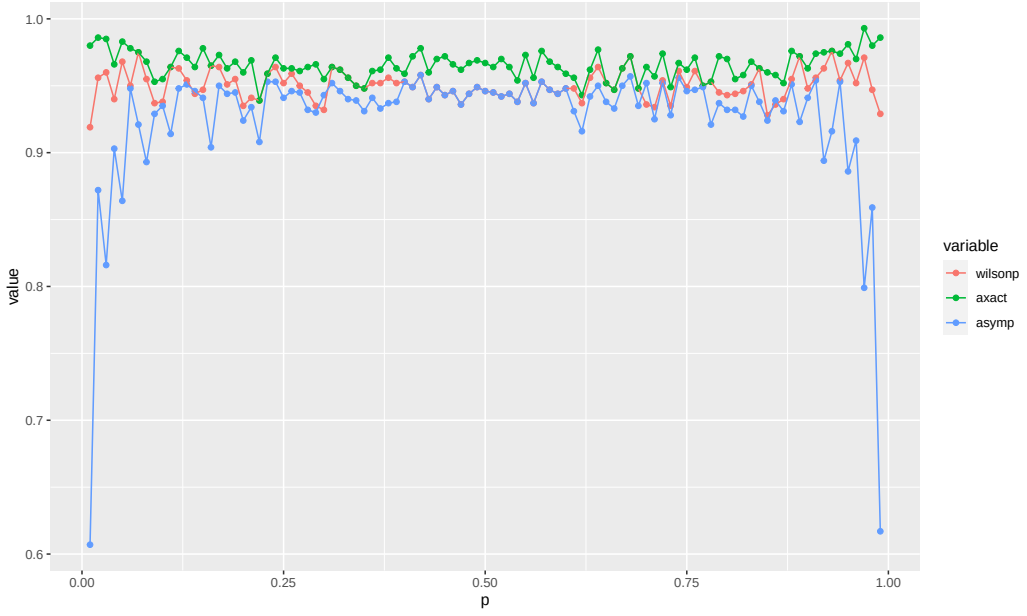
Rysunek 6. Wykres prawdopodobieństwa pokrycia dla poszczególnych metod i n równego 10.

```
df_l <- melt(df2,id.vars="p")
ggplot(df_l,aes(p, value,col=variable))+
  geom_point()+ geom_line()
```

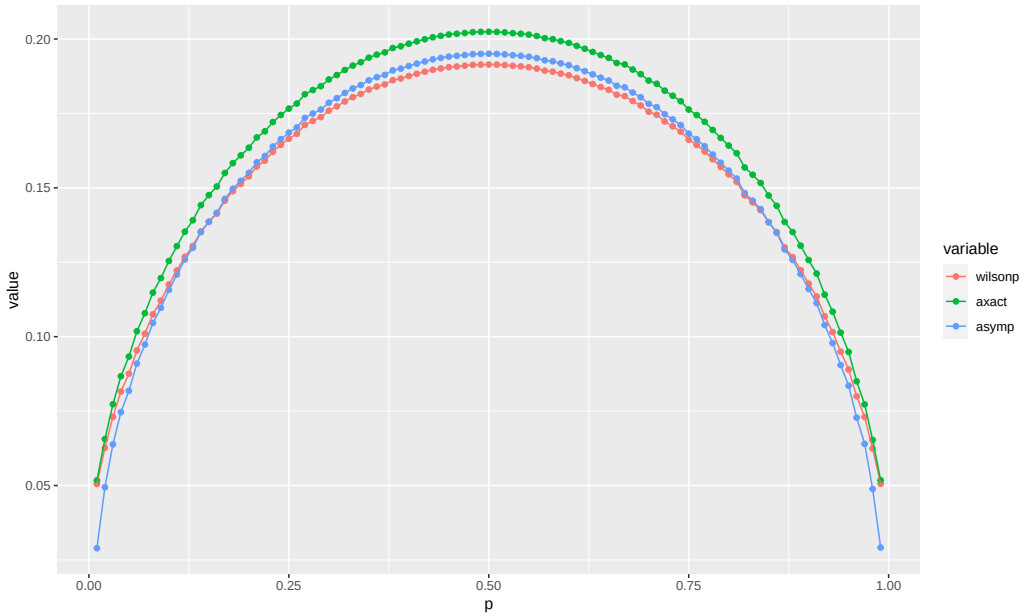


Rysunek 7. Wykres średniej długości przedziałów dla poszczególnych metod i n równego 10.

Zobaczmy również jak wygladają powyższe wykresy, ale tym razem dla n równego 100:

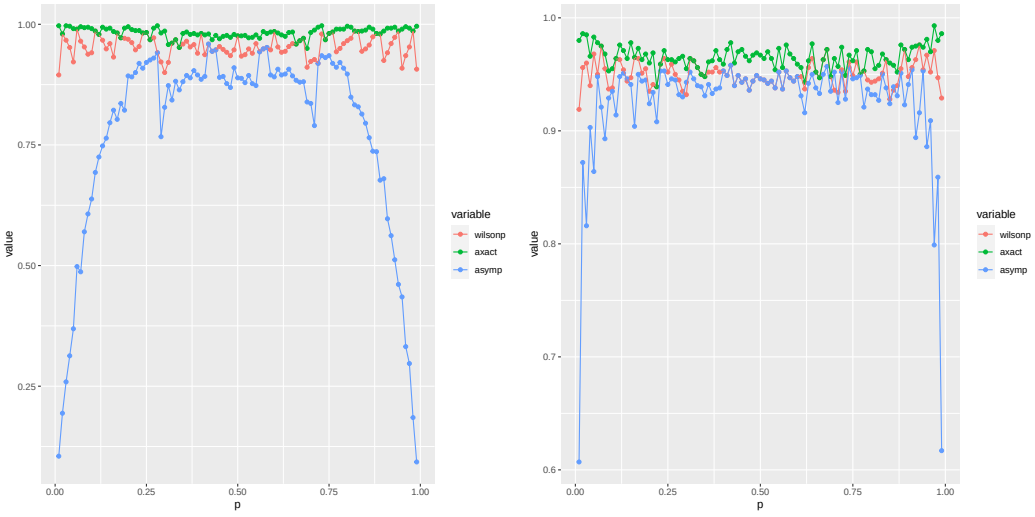


Rysunek 8. Wykres prawdopodobieństwa pokrycia dla poszczególnych metod i n równego 100.



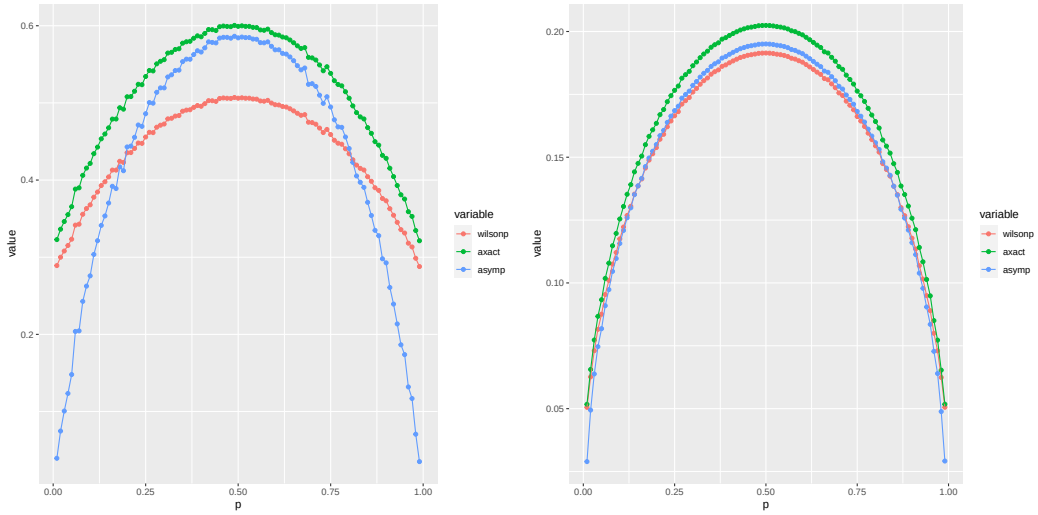
Rysunek 9. Wykres średniej długości przedziałów dla poszczególnych metod i n równego 100.

Porównanie Rysunku 6 z Rysunkiem 8:



Rysunek 10. Wykresy prawdopodobieństwa pokrycia dla poszczególnych metod i n równego odpowiednio 10 i 100.

Porównanie Rysunku 7 z Rysunkiem 9:



Rysunek 11. Wykresy średniej długości przedziałów dla poszczególnych metod i n równego 10 i 100.

Wnioski:

Jak możemy zobaczyć z powyższych wykresów, przedziały Cloppera-Pearsona mają największe prawdopodobieństwo pokrycia oraz największą średnią długość dla każdego p, świadczy to, że metoda Cloppera-Pearsona będzie najlepszym wyborem spośród testowanych metod. Zdecydowanie najgorszym okazał się test asymptotyczny, okazało się, że ta metoda wymaga jeszcze większych rozmiarów próbki.

Zad 2

Załóżmy, że 200 losowo wybranych klientów (w różnym wieku) kilku (losowo wybranych) aptek zapytano, jaki lek przeciwbólowy zwykle stosują. Zebrane dane zawarte są w tablicy 1. Na podstawie tych danych, wyznaczyć realizacje przedziałów ufności, na poziomie ufności 0.95.

Do wyboru najlepszych przedziałów ufności stosujemy funkcję $binom.confint$. Funkcja ta zwraca tabelkę z porównaniem 11 metod. Przedziały będziemy porównywać na podstawie ich długości, im węższy tym metoda jest lepsza. Długość takiego przedziału będzie opisana w tabelce w kolumnie ls.

a) Prawdopodobieństwo stosowania leku ibuprofen (bez względu na grupę wiekową)

```
conf <- binom.confint(50,200)
ls <- conf$"upper"- conf$"lower"
cbind(conf,ls)
```

```
##           method  x   n      mean     lower     upper        ls
## 1  agresti-coull 50 200 0.2500000 0.1948993 0.3145233 0.1196240
## 2     asymptotic 50 200 0.2500000 0.1899886 0.3100114 0.1200228
## 3          bayes 50 200 0.2512438 0.1923105 0.3115641 0.1192536
## 4        cloglog 50 200 0.2500000 0.1923621 0.3116476 0.1192856
## 5          exact 50 200 0.2500000 0.1916072 0.3159628 0.1243557
## 6          logit 50 200 0.2500000 0.1948697 0.3146322 0.1197625
## 7         probit 50 200 0.2500000 0.1939760 0.3136105 0.1196346
## 8        profile 50 200 0.2500000 0.1934176 0.3129498 0.1195322
## 9            lrt 50 200 0.2500000 0.1934316 0.3129489 0.1195173
## 10     prop.test 50 200 0.2500000 0.1928239 0.3169864 0.1241625
## 11        wilson 50 200 0.2500000 0.1950817 0.3143410 0.1192593
```

Dla tych danych najlepsza okazała się $metodaBayes'a$.

```
cbind(conf,ls)[ls == min(ls),]
```

```
##   method  x   n      mean     lower     upper        ls
## 3  bayes 50 200 0.2512438 0.1923105 0.3115641 0.1192536
```

b) Prawdopodobieństwo stosowania leku ibuprofen przez klienta w wieku do 35 lat

```
conf <- binom.confint(0,90)
ls <- conf$"upper"- conf$"lower"
cbind(conf,ls)
```

```
##           method x  n        mean        lower      upper         ls
## 1  agresti-coull 0 90 0.000000000 -0.008180285 0.04911591 0.05729620
## 2     asymptotic 0 90 0.000000000  0.000000000 0.00000000 0.00000000
## 3          bayes 0 90 0.005494505  0.000000000 0.02105727 0.02105727
## 4        cloglog 0 90 0.000000000  0.000000000 0.04015892 0.04015892
## 5          exact 0 90 0.000000000  0.000000000 0.04015892 0.04015892
## 6          logit 0 90 0.000000000  0.000000000 0.04015892 0.04015892
## 7         probit 0 90 0.000000000  0.000000000 0.04015892 0.04015892
## 8        profile 0 90 0.000000000  0.000000000 0.03652208 0.03652208
## 9            lrt 0 90 0.000000000  0.000000000 0.02111561 0.02111561
## 10     prop.test 0 90 0.000000000  0.000000000 0.05101162 0.05101162
## 11        wilson 0 90 0.000000000  0.000000000 0.04093563 0.04093563
```

Najlepszy w tym przypadku okazał się test asymptotyczny.

```
cbind(conf,ls)[ls == min(ls),]
```

```
##       method x  n mean lower upper ls
## 2 asymptotic 0 90    0     0     0  0
```

c) Prawdopodobieństwo stosowania leku apap (bez względu na grupę wiekową)

```
conf <- binom.confint(44,200)
ls <- conf$"upper"- conf$"lower"
cbind(conf,ls)
```

```
##           method  x   n     mean     lower     upper        ls
## 1  agresti-coull 44 200 0.220000 0.1679267 0.2826267 0.1147000
## 2     asymptotic 44 200 0.220000 0.1625894 0.2774106 0.1148211
## 3          bayes 44 200 0.221393 0.1651366 0.2792052 0.1140686
## 4        cloglog 44 200 0.220000 0.1654772 0.2795930 0.1141158
## 5          exact 44 200 0.220000 0.1646361 0.2838612 0.1192252
## 6          logit 44 200 0.220000 0.1679499 0.2827004 0.1147504
## 7         probit 44 200 0.220000 0.1670005 0.2815308 0.1145304
## 8        profile 44 200 0.220000 0.1663740 0.2807561 0.1143821
## 9            lrt 44 200 0.220000 0.1663832 0.2807552 0.1143720
## 10     prop.test 44 200 0.220000 0.1659406 0.2850661 0.1191255
## 11        wilson 44 200 0.220000 0.1681654 0.2823880 0.1142226
```

Po raz drugi test *Bayes'a* okazuję się najlepszy.

```
cbind(conf,ls)[ls == min(ls),]
```

```
##   method  x   n     mean     lower     upper        ls
## 3  bayes 44 200 0.221393 0.1651366 0.2792052 0.1140686
```

d) Prawdopodobieństwo stosowania leku apap przez klienta w wieku do 35 lat

```
conf <- binom.confint(22,90)
ls <- conf$"upper"- conf$"lower"
cbind(conf,ls)
```

```
##           method  x  n      mean     lower     upper        ls
## 1  agresti-coull 22 90 0.2444444 0.1667306 0.3430809 0.1763503
## 2     asymptotic 22 90 0.2444444 0.1556573 0.3332316 0.1775743
## 3          bayes 22 90 0.2472527 0.1612799 0.3363365 0.1750565
## 4        cloglog 22 90 0.2444444 0.1615228 0.3366897 0.1751669
## 5          exact 22 90 0.2444444 0.1599693 0.3463767 0.1864074
## 6          logit 22 90 0.2444444 0.1667000 0.3435007 0.1768006
## 7         probit 22 90 0.2444444 0.1648158 0.3411605 0.1763447
## 8        profile 22 90 0.2444444 0.1636309 0.3396167 0.1759858
## 9            lrt 22 90 0.2444444 0.1636231 0.3396152 0.1759921
## 10     prop.test 22 90 0.2444444 0.1626454 0.3484391 0.1857937
## 11        wilson 22 90 0.2444444 0.1673278 0.3424837 0.1751559
```

W podpunkcie *d*) najlepszym testem jest po raz trzeci test *Bayes'a*.

```
cbind(conf,ls)[ls == min(ls),]
```

```
##   method  x  n      mean     lower     upper        ls
## 3  bayes 22 90 0.2472527 0.1612799 0.3363365 0.1750565
```

Wnioski:
Zauważamy, że im mamy większą liczbę sukcesów tym metody zwracają bardziej zbliżone wyniki. Największą rozbieżność mamy przy testowaniu dla liczby sukcesów równej 0, wtedy niektóre przedziały mają nawet długość 0, a inne około 0.02. Test *Bayes'a* okazał się najlepszy w 3 podpunktach, więc możemy uznać, że jest to najlepsza metoda, którą możemy wykorzystać konstruując przedziały ufności dla tych danych.