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
```

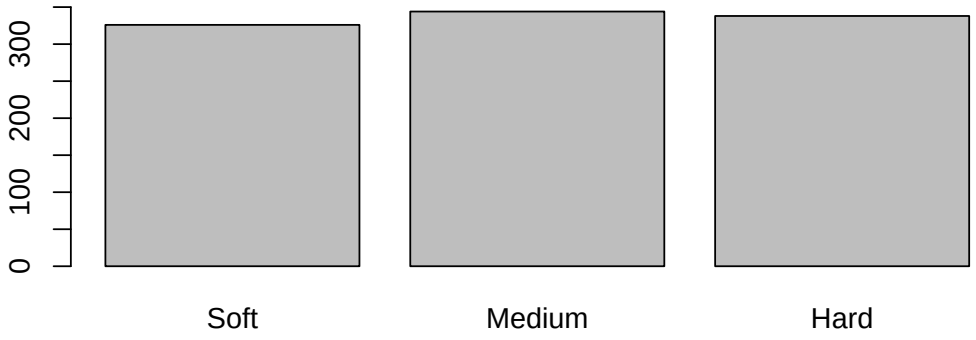
```
structable(Temperature ~ Water_softness, Detergent) %>% addmargins()
```

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

Zad 3

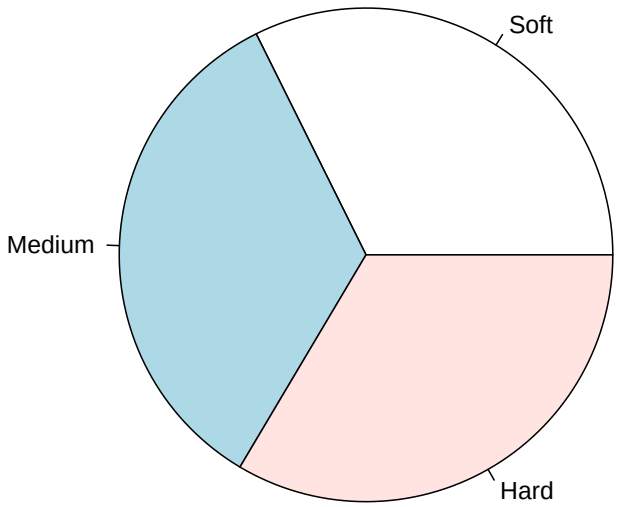
Sporządzić wykres kołowy i słupkowy dla zmiennej *Water Softness*:

```
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)
```

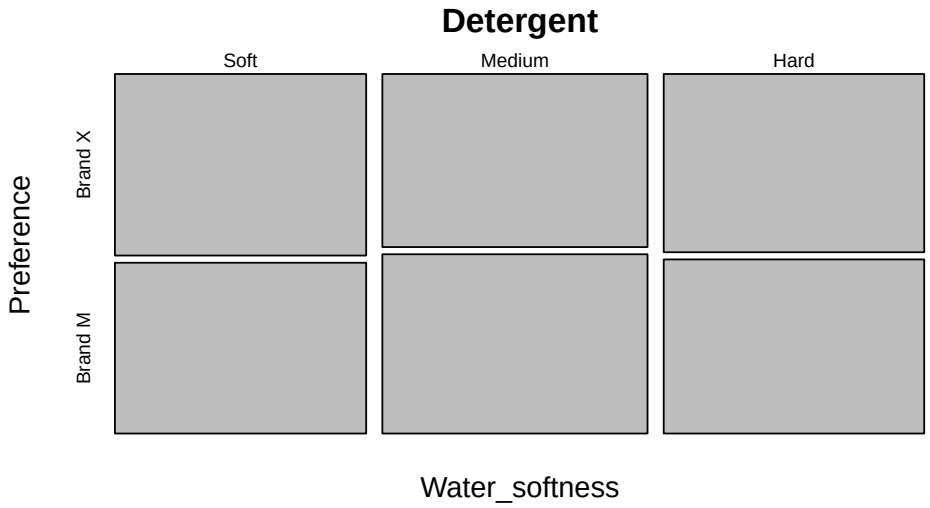


Rysunek 2. Wykresy kołowy dla zmiennej *Water Softness*.

Zad 4

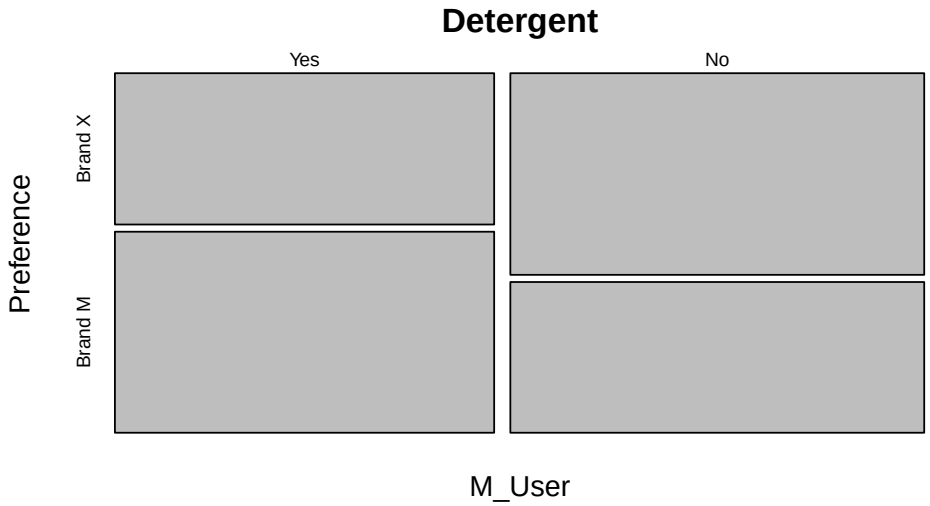
Sporządzić wykresy mozaikowe odpowiadające rozpatrywanym danym.

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



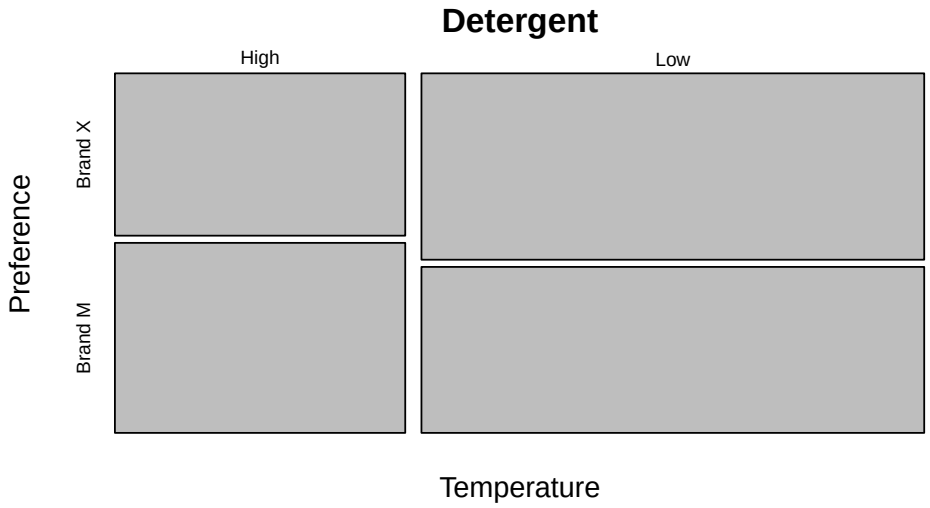
Rysunek 3. Wykres mozaikowy dla *Preference* i *Water softness*

```
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*.

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.

Losowanie ze zwracaniem:

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

Wylosowane indeksy:

```r
ind
```

```
## [1] 10 27 28
```

Wylosowane elementy z bazy danych:

```r
mtcars[ind,]
```

```
##                mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Merc 280      19.2   6 167.6 123 3.92 3.440 18.3  1  0    4    4
## Porsche 914-2 26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
## Lotus Europa  30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
```

Losowanie bez zwracania:

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

Wylosowane indeksy:

```
ind
```

```
## [1] 24 23 12
```

Wylosowane elementy z bazy danych:

```
mtcars[ind,]
```

```
##              mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Camaro Z28  13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## AMC Javelin 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Merc 450SE  16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
```

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] 2 4 2 4 3
```

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] 39.9826

var(test)

## [1] 23.93029
```

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,]    1    1    3    2    2
## [2,]    3    5    2    2    4
## [3,]    6    4    5    6    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 $p = \{0.2, 0.3, 0.5\}$:

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

```
## [1] 19.9791 29.9772 50.0437
```

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

```
rowVars(test)
```

```
## [1] 16.00966 20.93537 25.21491
```

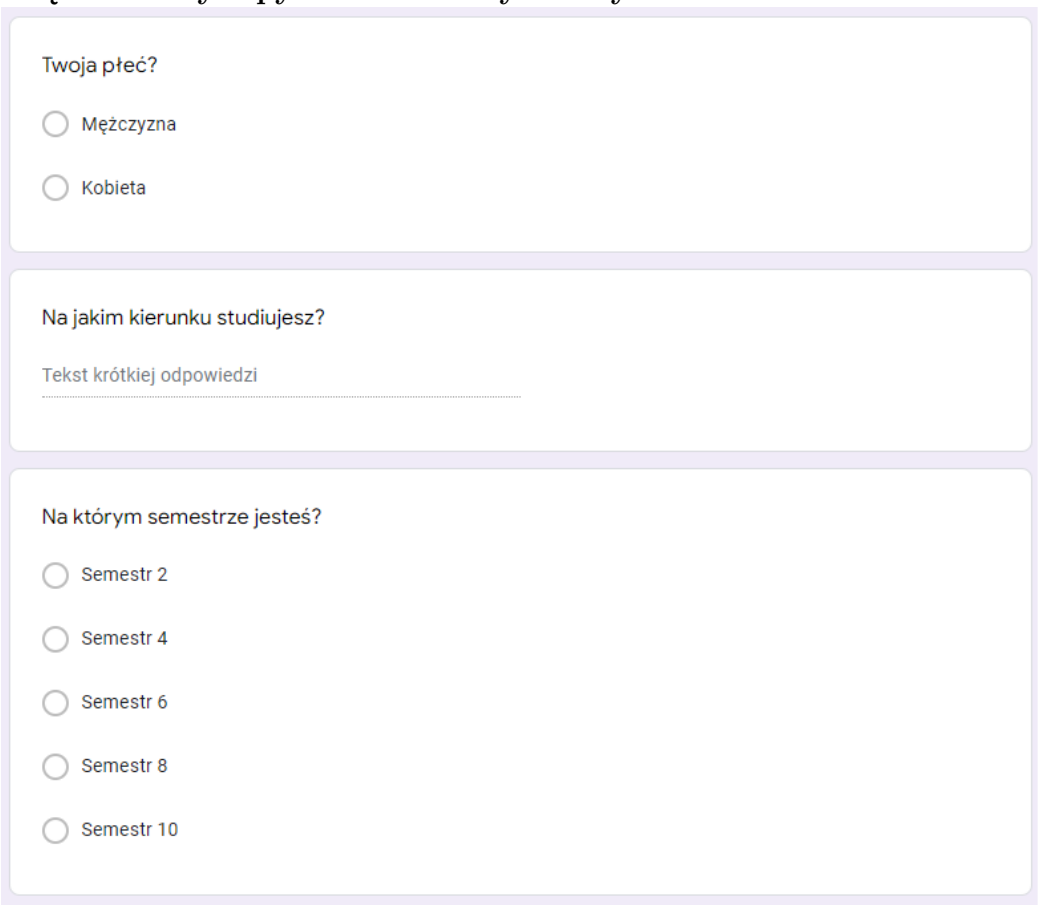
Widzimy zatem że symulowane wartości są bardzo bliskie wartości teoretycznych: $16, 21, 25$.
W obu powyższych przypadkach nasze wyniki są bardzo bliskie co wskazuje na poprawność metody.

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 ankiete 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 wykorzystaliśmy przedziały ufności Cloppera-Pearsona, Walda i Asymptotyczne. Symulacje przeprowadzimy na podstawie realizacji zmiennej loswej $X \sim \mathcal{B}(n, p)$.

```
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)
}
```

Odpolamy nasza szymulacje dla $n = 10$ i z krokiem $dp = 0.01$ i zapiszmy wynik w pliku csv:

```
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")
```

```
## Warning in file(file, "rt"): nie można otworzyć pliku 'data_n_10.csv':
No such file or directory
## Error in file(file, "rt"): nie można otworzyć połączenia
```

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

```
## Error in data[, 1]: obiekt typu 'closure' nie jest ustawialny
```

```
head(df1)
```

```
## Error in head(df1): nie znaleziono obiektu 'df1'
```

Stwórzmy teraz ramki danych średniej długości przedziałów

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

```
## Error in data[, 4]: obiekt typu 'closure' nie jest ustawialny
```

```
head(df2)
```

```
## Error in head(df2): nie znaleziono obiektu 'df2'
```

Spozadzmy również wykresy

```
df_p <- melt(df1,id.vars="p")
```

```
## Error in melt(df1, id.vars = "p"): nie znaleziono obiektu 'df1'
```

```
ggplot(df_p,aes(p, value,col=variable))+
  geom_point()+ geom_line()
```

```
## Error in ggplot(df_p, aes(p, value, col = variable)): nie znaleziono
obiektu 'df_p'
```

```
df_l <- melt(df2,id.vars="p")
```

```
## Error in melt(df2, id.vars = "p"): nie znaleziono obiektu 'df2'
```

```
ggplot(df_l,aes(p, value,col=variable))+
  geom_point()+ geom_line()
```

```
## Error in ggplot(df_l, aes(p, value, col = variable)): nie znaleziono
obiektu 'df_l'
```

Zobaczmy również jak wygladaja powyzsze wykresy ale tym razem dla n równego 100

```
## Warning in file(file, "rt"): nie można otworzyć pliku 'data_n_100.csv':
No such file or directory
## Error in file(file, "rt"): nie można otworzyć połączenia
## Error in data[, 1]: obiekt typu 'closure' nie jest ustawialny
## Error in data[, 4]: obiekt typu 'closure' nie jest ustawialny
## Error in melt(df1, id.vars = "p"): nie znaleziono obiektu 'df1'
## Error in melt(df2, id.vars = "p"): nie znaleziono obiektu 'df2'
```

```
## Error in ggplot(df_p2, aes(p, value, col = variable)): nie znaleziono
obiektu 'df_p2'
```

```
## Error in ggplot(df_l2, aes(p, value, col = variable)): nie znaleziono
obiektu 'df_l2'
```

Porównanie Rysunku ?? z Rysunkiem ?? oraz Rysunku ?? z Rysunkiem ??

```
## Error in ggplot(df_p, aes(p, value, col = variable)): nie znaleziono
obiektu 'df_p'
## Error in ggplot(df_p2, aes(p, value, col = variable)): nie znaleziono
obiektu 'df_p2'
```

```
## Error in ggplot(df_l, aes(p, value, col = variable)): nie znaleziono
obiektu 'df_l'
## Error in ggplot(df_l2, aes(p, value, col = variable)): nie znaleziono
obiektu 'df_l2'
```

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.

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 dłuższy tym metoda jest lepsza. Długość takiego przedziału bedzie 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 metoda Pearsona-Kloppera, która w tabelce występuję pod nazwą *exact*.

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

```
##   method  x   n mean     lower     upper        ls
## 5  exact 50 200 0.25 0.1916072 0.3159628 0.1243557
```

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 $Agresti - Coull'a$.

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

```
##          method x  n mean        lower      upper        ls
## 1 agresti-coull 0 90    0 -0.008180285 0.04911591 0.0572962
```

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 $Persona - Kloppera$ okazuję się najlepszy.

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

```
##   method  x   n mean     lower     upper        ls
## 5  exact 44 200 0.22 0.1646361 0.2838612 0.1192252
```

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 $Persona - Kloppera$.

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

```
##   method  x  n      mean     lower     upper        ls
## 5  exact 22 90 0.2444444 0.1599693 0.3463767 0.1864074
```

Wnioski:
Zauważamy, że im mamy wiekszą liczbę sukcesów tym metody zwracają bardziej zbliżone wyniki. Najwiekszą rozbieżność mamy przy testowaniu dla liczby sukcesów 0, wtedy niektóre przedziały mają nawet długość 0, a inne około 0.02. Test $Persona - Kloppera$, który w funckcji $binom.confint$ nazywa się $exact$, okazał się najlepszy w 3 podpunktach, więc możemy uznać, że jest to najlepsza metoda, którą możemy wykorzystać konstruując przedziały ufności dla tych danych.