Aleksander Jakóbczyk i Kacper Pasterniak

Sprawozdanie 1

Lista 1

Zad 1

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) %>% summar
```

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

```
Detergent.df %>% filter(Water_softness == "Medium") %>% group_by(Temperature) %>% summ
```

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

```
Detergent.df %>% filter(Water_softness == "Hard") %>% group_by(Temperature) %>% summar
```

```
## # 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) %>% summari
```

```
## # 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) %>% summa
```

```
## # 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) %>% summari
```

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

Zad 2

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

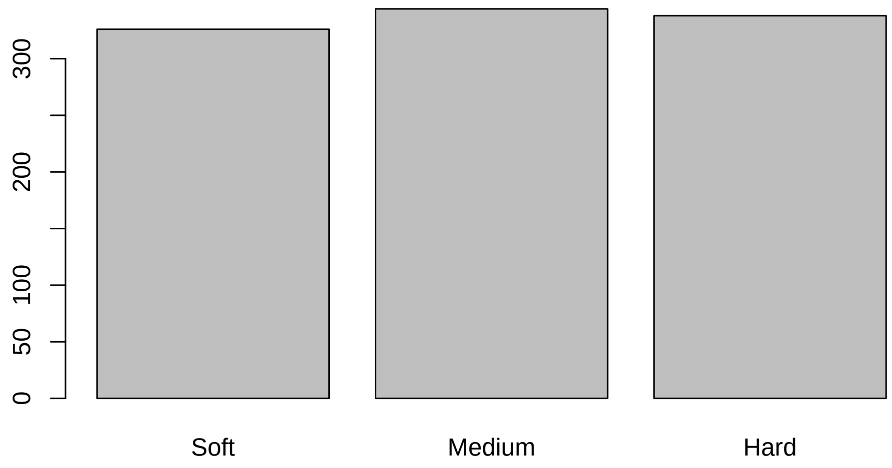
```
##                Temperature High Low
## Water_softness
## Soft                        104 222
## Medium                      126 218
## Hard                        139 199
```

```
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
```

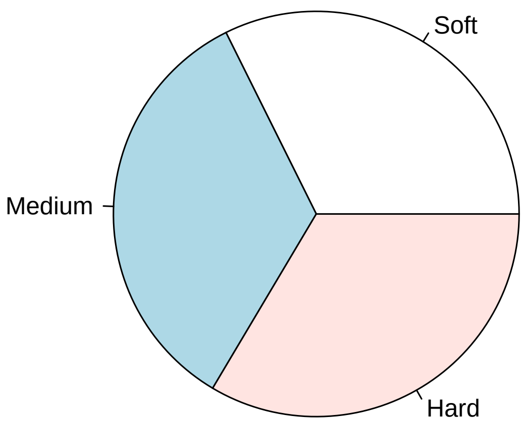
Zad 3

```
A <- apply(Detergent, "Water_softness", sum)
barplot(A)
```



Rysunek 1. Wykresy supkowy dla zmiennej Water Softness.

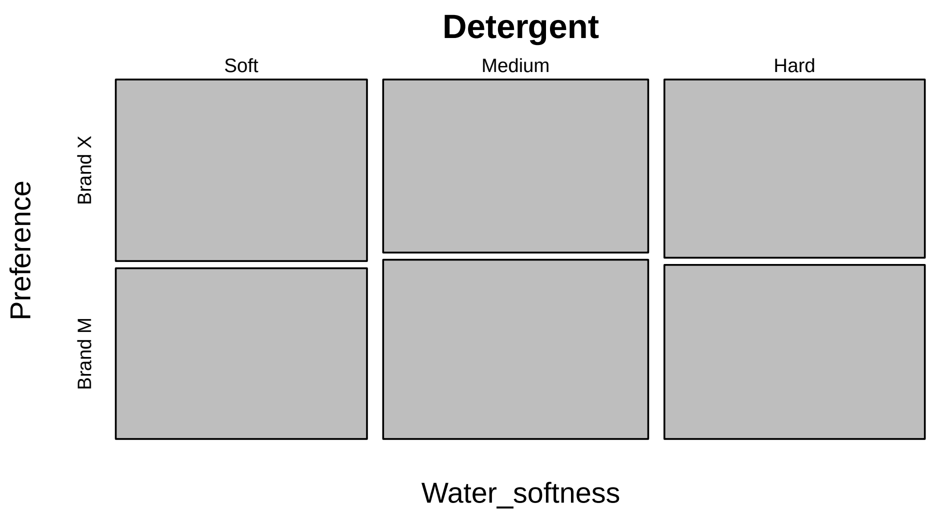
```r
par(mar = c(2, 2, 2, 2))
pie(A)
```



Rysunek 2. Wykresy koowy dla zmiennej Water Softness.

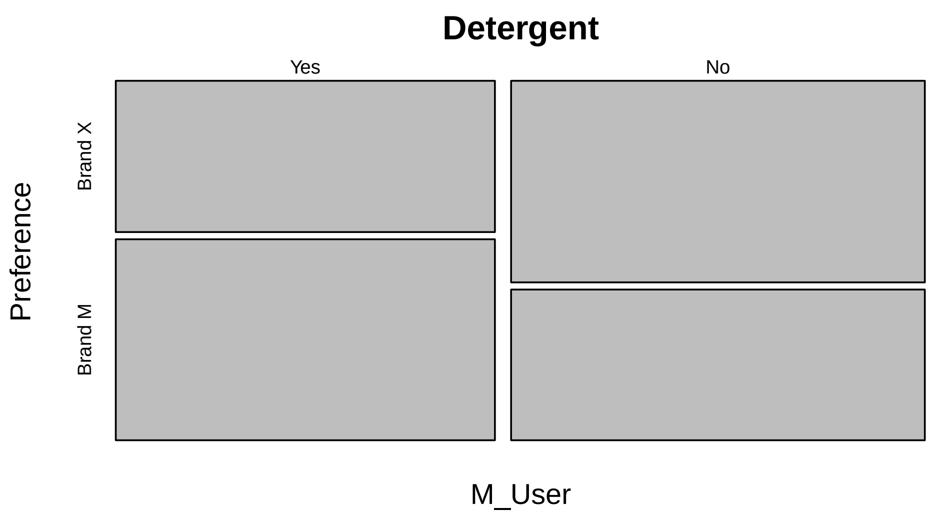
Zad 4

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



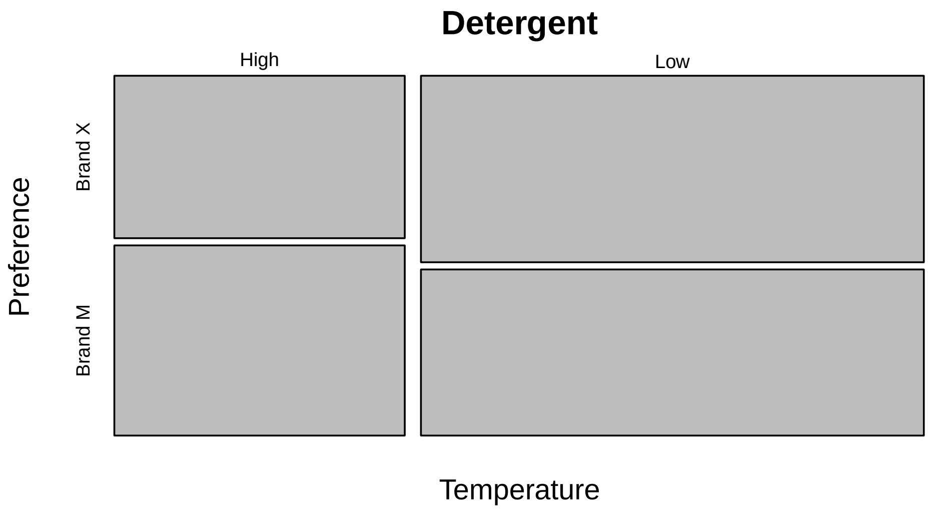
Rysunek 3. Wykres mozaikowy dla Preference i Water softness

```
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

Losowanie ze zwracaniem:

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

Wylosowane indeksy:

```r
ind

## [1] 12 26 11
```

Wylosowane elementy z bazy danych:

```r
mtcars[ind,]

##              mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Merc 450SE  16.4   8 275.8 180 3.07 4.070 17.4  0  0    3    3
## Fiat X1-9   27.3   4  79.0  66 4.08 1.935 18.9  1  1    4    1
## Merc 280C   17.8   6 167.6 123 3.92 3.440 18.9  1  0    4    4
```

Losowanie bez zwracania:

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

Wylosowane indeksy:

```
ind

## [1] 4 1 9
```

Wylosowane elementy z bazy danych:

```
mtcars[ind,]

##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
```

Zad 2

Propozycja algorytmu:

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

Algorytm opisany za pomoc 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 - prawdopodobiestwo, N - ilo wywoa

Przykadowe uycie:

```
bin(10,0.4,5)

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

Sprawdzenie poprawnoci:
Dla zmiennej losowej $X \sim \mathcal{B}(n,p)$ wiemy, e:
$\mathbb{E}(X) = np$,
$\text{Var}(X) = np(1-p)$,

Sprawdmy zatem dziaanie naszej funkcji dla $n = 1000$ i $p = 0.4$:

```
test <- bin(1000,0.4,10000)
mean(test)

## [1] 400.1427

var(test)

## [1] 238.761
```

Wartoci teoretyczne redniej i wariancji dla takich parametrów powinny wynosi kolejno 400 i 240. Nasze wyniki s bardzo bliskie co wskazuje na poprawno metody.

Zad 3

Chcemy wygenerowa generowa zmienna losowa z rozkadu wielomianowego o parametrach n i p, gdzie p jest wektorem wag prawdopodobiestw o dugoci k którego elementy sumuj si do jedynki.

Propozycja algorytmu:

1. Generujemy wektor zer o dugoci k.
2. Generuj wektor prób o dugoci n, przy czym w kadej próbie mamy do czynienia z wylosowaniem jednego z k zdarze o poszczególnym prawdopodobiestwem.
3. Sumuj ilo wystpowania kadego zdarzenia i zapisz je do wektora.
4. Krok 1 i 3 powtarzamy N razy.

Algorytm opisany za pomoc 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)
  }
```

Przykadowe uycie:

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

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

Sprawdzenie poprawnoci:
Niech zmienne losowe $X_1, X_2, \ldots, X_k$ oznaczaj liczby zaj poszczególnych zdarze 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, e:

$\mathbb{E}(X_i) = np_i$.

$\mathrm{Var}(X_i) = np_i(1 - p_i)$.

Sprawdmy zatem dziaanie 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] 19.9781 29.9578 50.0641
```

Widzimy zatem e symulowane wartoci s bardzo bliskie wartoci empirycznych: $20, 30, 50$.

```
rowVars(test)

## [1] 16.07543 21.13133 24.41563
```

Widzimy zatem e symulowane wartoci s bardzo bliskie wartoci empirycznych: $16, 21, 25$.
W obu powyszych przypadkach nasze wyniki s bardzo bliskie co wskazuje na poprawno metody.

Lista 3

W szymulacji wykorzystalismy przedziaów ufnoci Cloppera-Pearsona, Walda i Asymptotycznych. Symulacje przeporwadzimy na posdsawie ralizaji zmiennej loswej $X \sim \mathcal{B}(n, p)$.

```
symulation <-function(n = 10, dp= 0.2,MCs = 1000){
MCs <- MCs
n <- n
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)
```

```r
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:

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

Stwórzmy teraz ramki danych prawdopodobiestw pokrycia

```r
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
```

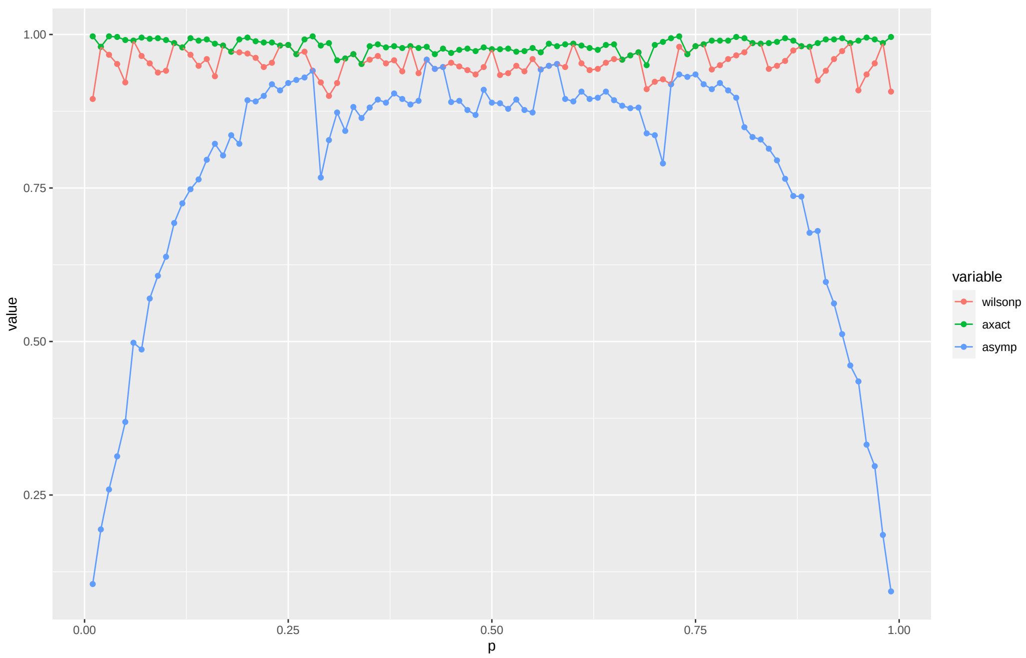
Stwórzmy teraz ramki danych redniej dugoci przedziaów

```
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
```

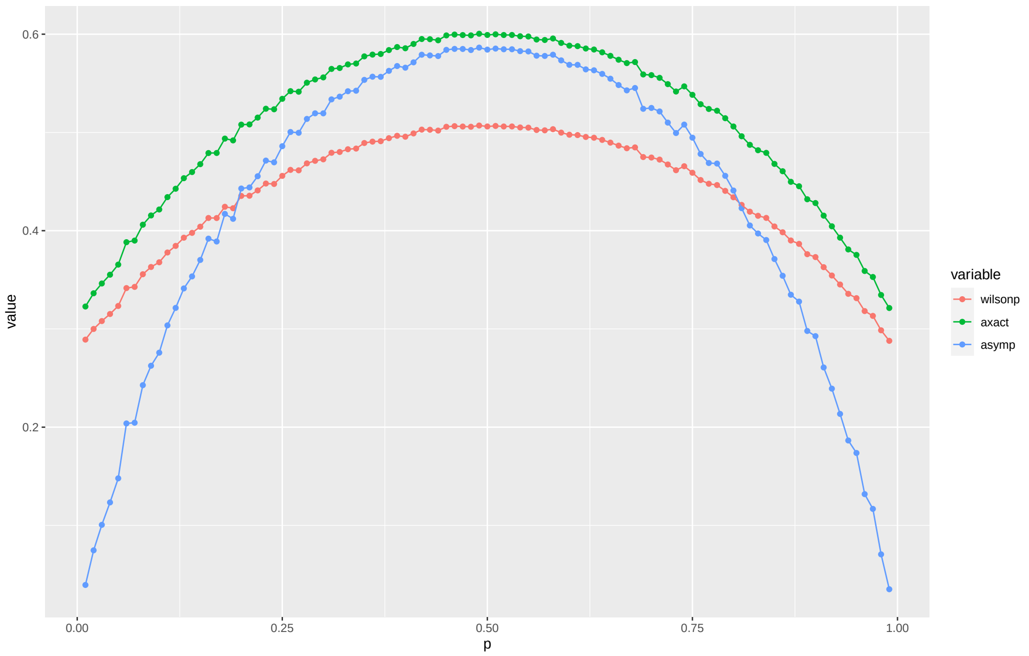
Spozadzmy równie wykresy

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



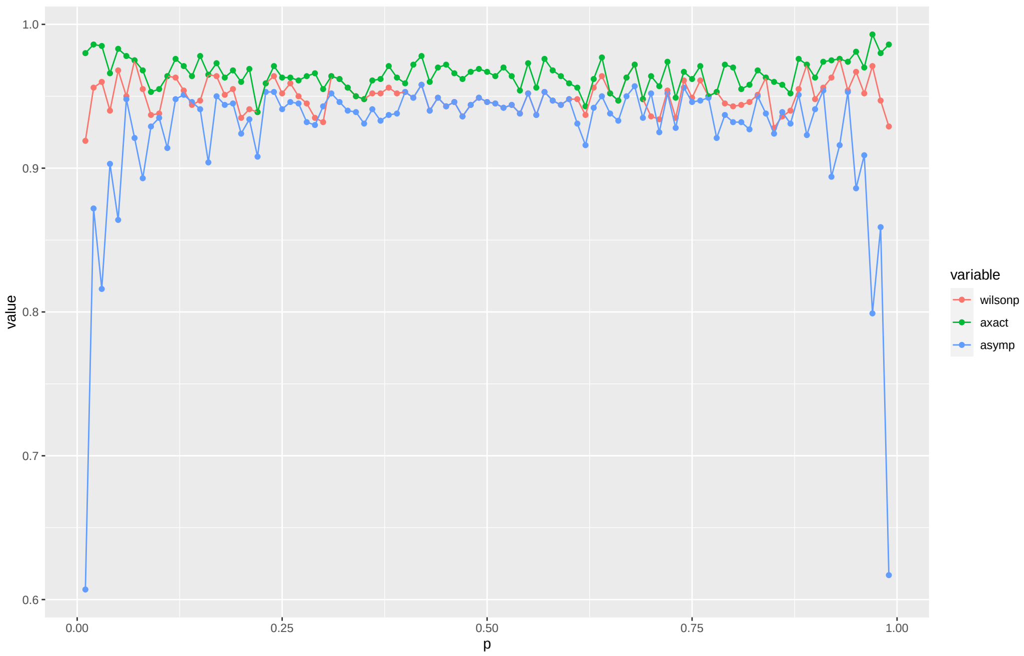
Rysunek 6. Wykres prawdopodobiestwa pokrycia dla poszczególnych metod i n równego 10.

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

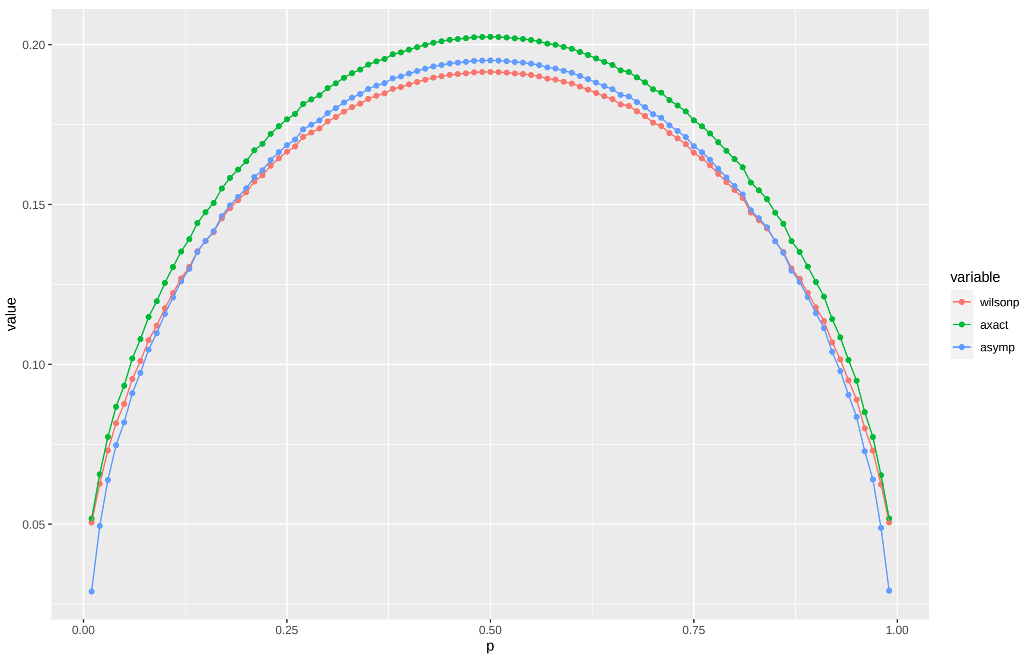


Rysunek 7. Wykres redniej dugoci przedziaów dla poszczególnych metod i n równego 10.

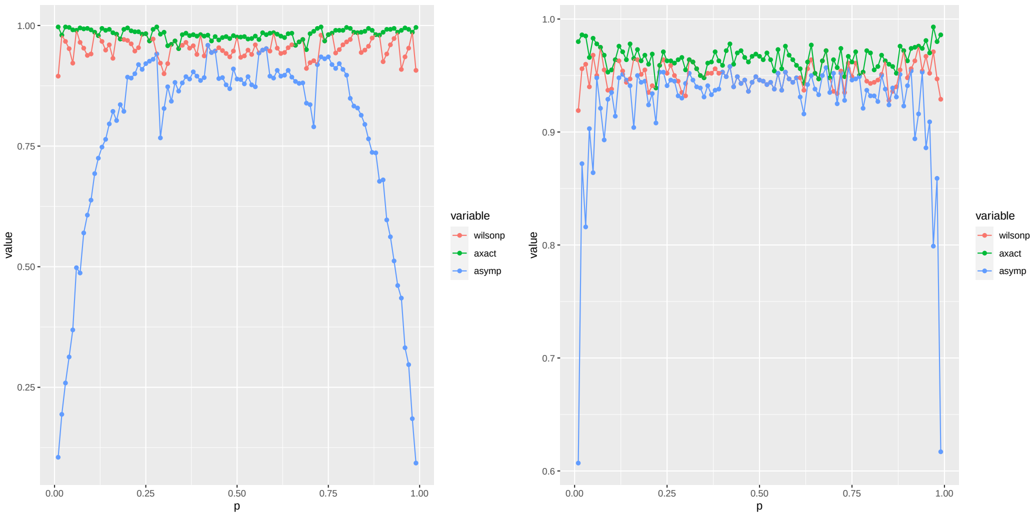
Zobaczmy równie jak wygladaja powyzsze wykresy ale dla tym razem dla n równego 100



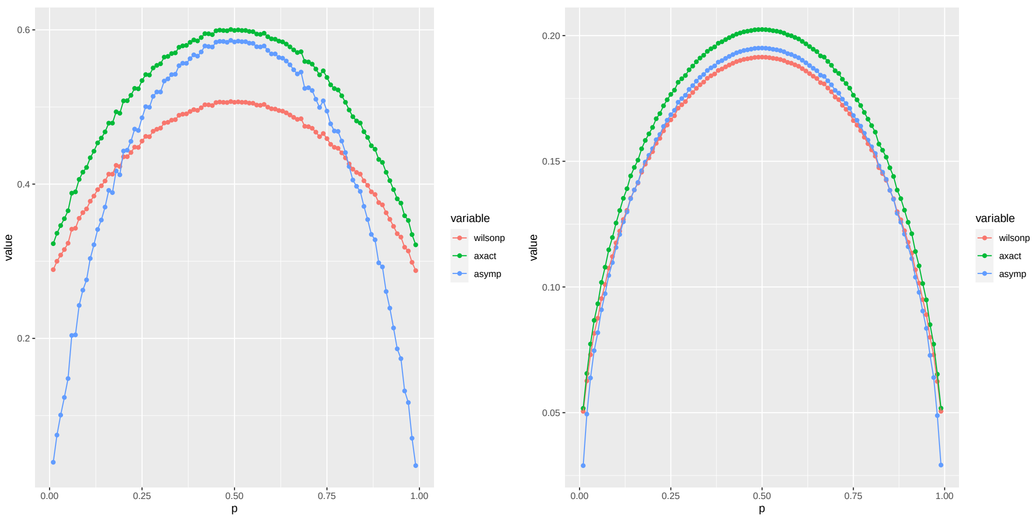
Rysunek 8. Wykres prawdopodobiestwa pokrycia dla poszczególnych metod i n równego 10.



Rysunek 9. Wykres redniej dugoci przedziaów dla poszczególnych metod i n równego 10.



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



Rysunek 11. Wykresy redniej dugoci przedziaów dla poszczególnych metod i n równego 10 i 100.

Zad 2

a)

```
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
```

Najlepszy

```
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)

```
conf <- binom.confint(0,200)
ls <- conf$"upper"- conf$"lower"
cbind(conf,ls)
```

```
##            method x   n        mean        lower       upper          ls
## 1  agresti-coull 0 200 0.000000000 -0.003840065 0.022685391 0.026525456
## 2     asymptotic 0 200 0.000000000  0.000000000 0.000000000 0.000000000
## 3          bayes 0 200 0.002487562  0.000000000 0.009545787 0.009545787
## 4        cloglog 0 200 0.000000000  0.000000000 0.018275340 0.018275340
## 5          exact 0 200 0.000000000  0.000000000 0.018275340 0.018275340
## 6          logit 0 200 0.000000000  0.000000000 0.018275340 0.018275340
## 7         probit 0 200 0.000000000  0.000000000 0.018275340 0.018275340
## 8        profile 0 200 0.000000000  0.000000000 0.016677710 0.016677710
## 9            lrt 0 200 0.000000000  0.000000000 0.009573900 0.009573900
## 10     prop.test 0 200 0.000000000  0.000000000 0.023490044 0.023490044
## 11        wilson 0 200 0.000000000  0.000000000 0.018845326 0.018845326
```

Najlepszy

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

```
##          method x   n mean        lower      upper         ls
## 1 agresti-coull 0 200    0 -0.003840065 0.02268539 0.02652546
```

c)

```r
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
```

Najlepszy

```r
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)

```r
conf <- binom.confint(22,200)
ls <- conf$"upper"- conf$"lower"
cbind(conf,ls)
```

```
##           method  x   n      mean      lower     upper         ls
## 1  agresti-coull 22 200 0.1100000 0.07316852 0.1615308 0.08836232
## 2     asymptotic 22 200 0.1100000 0.06663649 0.1533635 0.08672702
## 3          bayes 22 200 0.1119403 0.07001320 0.1560624 0.08604920
## 4        cloglog 22 200 0.1100000 0.07144094 0.1578266 0.08638562
## 5          exact 22 200 0.1100000 0.07023316 0.1617962 0.09156305
## 6          logit 22 200 0.1100000 0.07353070 0.1614059 0.08787522
## 7         probit 22 200 0.1100000 0.07253867 0.1596458 0.08710711
## 8        profile 22 200 0.1100000 0.07166319 0.1582594 0.08659619
## 9            lrt 22 200 0.1100000 0.07165897 0.1582584 0.08659941
## 10     prop.test 22 200 0.1100000 0.07173658 0.1637902 0.09205366
## 11        wilson 22 200 0.1100000 0.07377244 0.1609269 0.08715447
```

Najlepszy

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

```
##       method  x   n mean      lower     upper         ls
## 10 prop.test 22 200 0.11 0.07173658 0.1637902 0.09205366
```