Krzysztof Trajkowski

Przeglad pakietów do analizy zmiennych niezależnych

7 października 2012

1. Wprowadzenie

Niniejsze opracowanie ma na celu pokazanie kilka ciekawych alternatyw dla klasycznych testów (parametrycznych i nieparametrycznych) które są dostępne w środowisku R. Jest ono napisane w formie poradnika dla osób, które chcą szybko wdrożyć wybrane tradycyjne lub nowoczesne metody statystyczne w R. W pierwszej części każdego rozdziału będą prezentowane klasyczne procedury analizy danych. Natomiast w drugiej zostanie przedstawione praktyczne wprowadzenie do alternatywnych nowoczesnych metod. Współczesne metody są przeznaczone do stosowania w przypadku, gdy pewne założeniach są spełnione lub są one naruszone (normalność, homogeniczność) oraz gdy występują pewne problemy takie jak: obserwacje odstające, zbyt mała liczebność próby. Warto podkreślić, że wyżej wymienione założenia są rzadko spełnione przy analizie prawdziwych danych. Dlatego analitycy mają niewiele do stracenia i wiele do zyskania stosując nowoczesne metody statystyczne zamiast tradycyjnych technik. Dostępność profesjonalnego oprogramowania (R jest pakietem wieloplatformowym na licencji open source) daje nadzieję, że autorzy podręczników do statystyki będą częściej przedstawiać nowe metody pomimo iż konwencjonalne programy nauczania statystyki niewiele oferują poza klasycznymi testami.

2. Dwie próby niezależne

2.1. Test t-Studenta

Najbardziej powszechnym sposobem oceny różnic między dwoma niezależnymi grupami jest stosowanie klasycznego testu t-Studenta. Należy pamiętać, że w omawianym teście muszą być spełnione pewne założenia: normalność rozkładu oraz równość wariancji.

Hipotezy badawcze:

$$\begin{aligned} H_0 &: \quad \mu_1 = \mu_2 \\ H_1 &: \quad \mu_1 \neq \mu_2 \end{aligned} \tag{2.1}$$

Statystyka testu:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{2.2}$$

gdzie:

$\bar{x}_k$ to średnia arytmetyczna k-tej próby:

$$\bar{x}_k = \frac{1}{n_k} \sum_{i=1}^{n_k} x_i \tag{2.3}$$

s_k^2 to wariancja k-tej próby:

$$s_k^2 = \frac{1}{n_k - 1} \sum_{i=1}^{n_k} (x_i - \bar{x}_k)^2 \tag{2.4}$$

ma rozkład t-Studenta ze stopniami swobody:

$$df = n_1 + n_2 - 2 \tag{2.5}$$

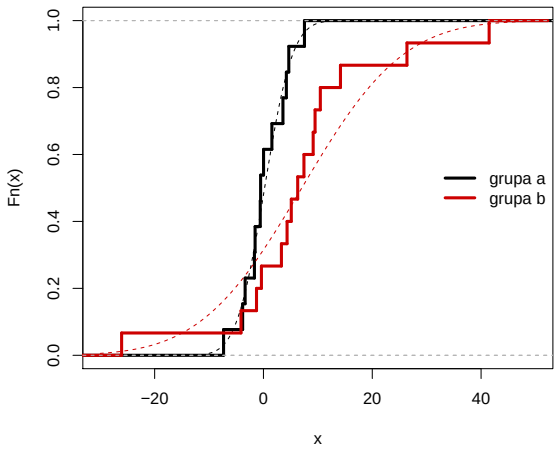
Jeśli wariancje nie są równe to stopnie swobody są przybliżane za pomocą formuły Welcha:

$$df_W = \frac{\left(s_1^2/n_1 + s_2^2/n_2\right)^2}{\frac{\left(s_1^2/n_1\right)^2}{n_1-1} + \frac{\left(s_2^2/n_2\right)^2}{n_2-1}} \tag{2.6}$$

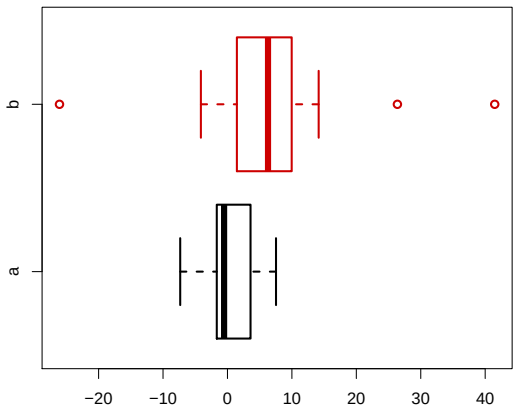
Jest to typowe rozwiązanie problemu Behrensa-Fishera (brak równości wariancji w grupach) dla dwóch prób niezależnych. Test t–Studenta ze zmodyfikowanymi stopniami swobody (2.6) nazywany jest też jako Welch–Aspin test [1938].

```
# przykładowe dane:
> set.seed(243)
> a= rnorm(13, mean= 0, sd= 5)
> set.seed(15)
> b= c(rnorm(10, mean= 5, sd= 5), rnorm(5, mean= 5, sd= 25))
```

```
# wykresy dystrybuant:
> plot(ecdf(a),verticals=T,do.points=F,col.h="black",col.v="black",↩
   lwd=3,xlim=c(-30,50))
> curve(pnorm(x,mean(a),sd(a)),add=T,lty=2,from=-10,to=20,lwd=1)
> lines(ecdf(b),verticals=T,do.points=F,col.h="red3",col.v="red3",↩
   lwd=3)
> curve(pnorm(x,mean(b),sd(b)),add=T,lty=2,col="red3",lwd=1)
> legend("right",c("grupa a","grupa b"),bty='n',lty=1,lwd=3,col=c("↩
   black","red3"))
# wykresy pudełkowe:
> boxplot(a,b,border=c("black","red3"),names=c("a","b"),horizontal=T↩
   ,lwd=2)
```



Rys. 2.1. Dystrybuanty.



Rys. 2.2. Wykresy pudełkowe.

```
> y=c(a,b)
> g=factor(rep(letters[1:2],c(length(a),length(b))))
# p-value Shapiro-Wilk:
> tapply(y,g,function(x) shapiro.test(x)$p.value)
        a         b
0.9858983 0.1110795
# p-value F-test:
> var.test(a,b)$p.value
[1] 7.137497e-05
```

Według powyższych obliczeń można dojść do wniosku, że do porównania dwóch średnich powinniśmy skorzystać z testu Welcha.

```
# test Welcha:
> t.test(a,b)

        Welch Two Sample t-test

data:  a and b
t = -1.7339, df = 16.424, p-value = 0.1017
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -15.184639   1.505355
sample estimates:
mean of a mean of b
0.2093677 7.0490098
```

Jednak na podstawie testu normalności Shapiro-Francia hipotetę zerową zakładającą rozkład normalny zmiennej `b` należy odrzucić.

```
# p-value Shapiro-Francia:
> library(nortest)
> tapply(y,g,function(x) sf.test(x)$p.value)
         a          b
0.93495930 0.03733246
```

```
# moc testu Shapiro-Wilka:
> out1= replicate(10000,{
+  q1= sample(b,length(b),T)
+  shapiro.test(q1)$p.value })
> mean( out1 <= 0.05 )
[1] 0.676
# moc testu Shapiro-Francia:
> out1= replicate(10000,{
+  q1= sample(b,length(b),T)
+  sf.test(q1)$p.value })
> mean( out1 <= 0.05 )
[1] 0.7453
```

Wyniki testu Shapiro-Francia potwierdzają również inne testy normalności:

```
# p-value E-test - bootstrap:
> library(energy)
> tapply(y,g,function(x) mvnorm.etest(x, R=199)$p.value)
         a          b
0.87437186 0.02512563
# p-value odporny Jarque-Bery - sumulacja monte-carlo:
> library(lawstat)
> tapply(y,g,function(x) rjb.test(x,option="RJB",crit.values="↩
   empirical",N=10000)$p.value)
         a          b
0.98033025 0.02334094
# p-value Neyman:
> library(ddst)
> tapply(y,g,function(x) ddst.norm.test(x,compute.p=TRUE,B=10000)$p.↩
   value)
     a      b
0.8688 0.0423
```

Ciekawą propozycją jest test Yuena [1974] dla obciętej średniej który jest nieco lepszym testem od testu Welcha. Średnia obcięta w większości przypadków lepiej opisuje tendencję centralną niż średnia arytmetyczna.

Hipotezy badawcze:

$$\begin{aligned} H_0 &: \quad \mu_{t1} = \mu_{t2} \\ H_1 &: \quad \mu_{t1} \neq \mu_{t2} \end{aligned} \tag{2.7}$$

Do obliczenia statystyki testu zostanie wykorzystana średnia obcięta oraz wariancja winsorowska:

$$t_Y = \frac{\bar{x}_{t1} - \bar{x}_{t2}}{\sqrt{d_1 + d_2}} \tag{2.8}$$

gdzie:

$$d_k = \frac{(n_k - 1)s^2_{wk}}{h_k(h_k - 1)} \tag{2.9}$$

n_k to wielkość k-tej próby,
h_k to wielkość k-tej próby po obcięciu,
$\bar{x}_t$ to średnia obcięta,

```
# średnia obcięta:
> c(mean(a,trim=0.2),mean(b,trim=0.2))
[1] 0.1873219 6.1325711
```

s^2_{wk} to wariancja winsorowska k-tej próby. Inaczej mówiąc jest to wariancja z próby poddanej procesowi winsoryzacji – w_i . Winsoryzację można przeprowadzić za pomocą funkcji `win(asbio)`.

$$s^2_{wk} = \frac{1}{n_k - 1} \sum_{i=1}^{n_k} (w_i - \bar{x}_{wk})^2 \tag{2.10}$$

$$\bar{x}_{wk} = \frac{1}{n_k} \sum_{i=1}^{n_k} w_i \tag{2.11}$$

```
> library(asbio)
# wariancja winsorowska:
> c(var(win(a)),var(win(b)))
[1]  8.941441 19.678954
# średnia winsorowska:
> c(mean(win(a)),mean(win(b)))
[1] 0.259148 5.695315
```

Do obliczeń wyżej wymienionych statystyk można wykorzystać gotowe funkcje z pakietu `WRS`. Średnia ucięta: `tmean(WRS)`, wariancja winsorowska: `winvar(WRS)`, średnia winsorowska: `win(WRS)`.

Statystyka testu (2.8) ma rozkład t-Studenta ze stopniami swobody:

$$df_Y = \frac{(d_1 + d_2)^2}{d_1^2/(h_1 - 1) + d_2^2/(h_2 - 1)} \tag{2.12}$$

```
> library(PairedData)
> yuen.t.test(a,b)

        Two-sample Yuen test, trim=0.2

data:  x and y
t = -2.5784, df = 13.411, p-value = 0.02247
alternative hypothesis: true difference in trimmed means is not ↩
   equal to 0
95 percent confidence interval:
 -10.9111616  -0.9793367
sample estimates:
trimmed mean of a trimmed mean of b
        0.1873219         6.1325711
```

Jeśli dodamy do funkcji `yuen.t.test` opcję `tr=0.0` to otrzymamy wyniki identyczne jak w teście Welcha. Domyślna wartość to `tr=0.2`.

```
> yuen.t.test(a, b, tr= 0.0)$p.value
[1] 0.1017
```

Na podstawie testu Welcha (2.6) brak jest podstaw do odrzucenia hipotezy zerowej (równość średnich arytmetycznych). Natomiast $p - value$ dla testu Yuena (2.8) jest równe 0.02247 a zatem dla $\alpha = 0.05$ hipotezę zerową (równość średnich obciętych) należy odrzucić. Gdy porównamy moc obu testów to możemy dojść do wniosku, że lepszym testem jest test Yuena.

```
> m=10000
> statystyka = NULL
> alpha=0.05
# moc testu Welcha:
> for (i in 1:m) {
+ q1=sample(a,length(a),T)
+ q2=sample(b,length(b),T)
+ statystyka[i] = t.test(q1,q2,var.equal=F)$p.value
+ }
> mean(statystyka < alpha)
[1] 0.4234
# moc testu Yuena:
> for (i in 1:m) {
+ q1=sample(a,length(a),T)
+ q2=sample(b,length(b),T)
+ statystyka[i] = yuen.t.test(q1,q2)$p.value
+ }
> mean(statystyka < alpha)
[1] 0.5866
```

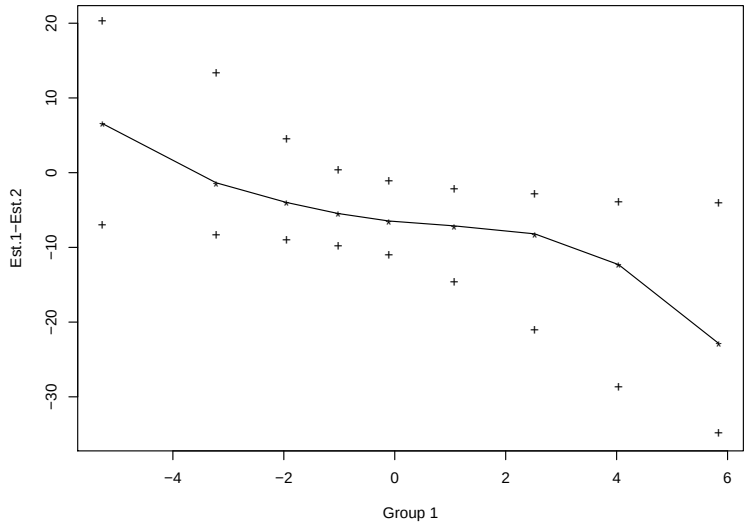
Wiele ciekawych alternatyw dla tradycyjnych testów (z wykorzystaniem pakietu `WRS` – Wilcoxon Robust Statistics) jest przedstawionych w książce *Introduction to Robust Estimation and Hypothesis Testing* której autorem jest Rand Wilcox. Możemy tam znaleźć np. test Yuena oraz jego wersję bootstrapową. Ten ostatni jest szczególnie polecany w sytuacji gdy rozmiary próbek są małe np. $n < 30$.

```
# test Yuena:
> yuen(a, b, tr= 0.2, alpha= 0.05)$p.value
[1] 0.02247492
```

```
# test Yuena - wersja bootstrapowa:
> yuenbt(a, b, tr= 0.2, alpha= 0.05, nboot= 9999, side= T)
[1] "Taking bootstrap samples. Please wait."
$ci
[1] -10.714808  -1.175691

$test.stat
[1] -2.522524

$p.value
[1] 0.01720172
```

Przy porównywaniu dwóch prób niezależnych warto zwrócić uwagę także na funkcję `qcomhd(WRS)`. Porównuje ona wybrane kwantyle z wykorzystaniem estymatora Harrella-Davisa [1982]. Generowany jest też wykres (Rys. 2.3).

```
> qcomhd(a, b, q= seq(0.1, 0.9, by= 0.1))
    q n1 n2      est.1      est.2 est.1_-_est.2     ci.low     ci.up
1 0.1 13 15 -5.2714521 -11.846568      6.575116  -7.099096 20.149444
2 0.2 13 15 -3.2191970  -1.862641     -1.356556  -8.438903 13.127821
3 0.3 13 15 -1.9492028   2.043682     -3.992884  -9.210532  4.329309
4 0.4 13 15 -1.0201145   4.449692     -5.469806 -10.029803  0.261833
5 0.5 13 15 -0.1060665   6.362018     -6.468084 -11.231118 -1.207709
6 0.6 13 15  1.0730481   8.196137     -7.123089 -14.818982 -2.377140
7 0.7 13 15  2.5241900  10.713495     -8.189305 -21.168448 -3.005859
8 0.8 13 15  4.0358749  16.342897    -12.307023 -28.877169 -4.073614
9 0.9 13 15  5.8403580  28.695959    -22.855601 -35.029182 -4.241833
       p_crit p.value signif
1 0.025000000   0.695     NO
2 0.050000000   0.710     NO
3 0.016666667   0.257     NO
4 0.012500000   0.063     NO
5 0.010000000   0.016     NO
6 0.008333333   0.005    YES
7 0.007142857   0.001    YES
8 0.006250000   0.001    YES
9 0.005555556   0.000    YES
```



Rys. 2.3. Graficzna prezentacja różnic oszacowanych kwantyli z przedziałami ufności.

2.2. Test Wilcoxona–Manna–Whitneya

Przy porównywaniu dwóch zmiennych które pochodzą z innych rozkładów niż normalny (niejednokrotnie także o różnych wariancjach) stosowanie testu Wilcoxona–Manna–Whitneya jest częstym wyborem.

Statystyka testu Wilcoxona–Manna–Whitneya:

$$W = S_k - \frac{n_k(n_k+1)}{2} \tag{2.13}$$

gdzie: S_k – to mniejsza suma rang. Rangi przypisujemy wszystkim obserwacjom z połączonej próby: pierwszej i drugiej, n_k – to liczebność próbki dla której suma rang jest mniejsza.

W przypadku testu Wilcoxona–Manna–Whitneya hipotezy badawcze mogą dotyczyć równości rozkładów:

$$\begin{aligned} H_0: &\quad F_1(x) = F_2(x) \\ H_1: &\quad F_1(x) \neq F_2(x) \end{aligned} \tag{2.14}$$

Można też wykonać test dla parametru przesunięcia lokalizacji.

$$\begin{aligned} H_0: &\quad F_1(x) = F_2(x-d) \\ H_1: &\quad F_1(x) \neq F_2(x-d) \end{aligned} \tag{2.15}$$

gdzie d to estymator mediany Hodgesa-Lehmanna (2.16) wszystkich sparowanych różnic obserwacji w dwóch próbkach, który może być zapisany jako:

$$\hat{d} = mediana(X_i - Y_j) \quad \text{gdzie } i = 1, ...n_1 \text{ oraz } j = 1, ...n_2 \tag{2.16}$$

```
# estymator mediany Hodgesa-Lehmanna:
> d= median(outer(a,b,"-"))
> d
[1] -5.997955
```

W funkcjach `wilcox.test(stats)` i `wilcox_test(coin)` opcjami domyślnymi są `alternative="two.sided"` oraz `mu=0`. A zatem hipotezy badawcze mają następującą postać:

$$\begin{aligned} H_0: &\quad d = 0 \\ H_1: &\quad d \neq 0 \end{aligned} \tag{2.17}$$

Poniżej przykład w którym wartości liczbowe nie powtarzają się.

```
> Y= c(a,b); G= factor(rep(1:2,c(length(a),length(b))))
# dokładne p-value - biblioteka stats:
> wilcox.test(Y~G,conf.int=T)

        Wilcoxon rank sum test

data:  Y by G
W = 48, p-value = 0.02224
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -10.9758268  -0.7589586
sample estimates:
```

```
difference in location
             -5.997955
# dokładne p-value - biblioteka coin:
> library(coin)
> wilcox_test(Y~G, conf.int=T,  distribution ="exact")

        Exact Wilcoxon Mann-Whitney Rank Sum Test

data:  Y by G (1, 2)
Z = -2.2802, p-value = 0.02224
alternative hypothesis: true mu is not equal to 0
95 percent confidence interval:
 -10.9758268  -0.7589586
sample estimates:
difference in location
             -5.997955
```

W przypadku gdy liczebność przynajmniej jednej z dwóch prób jest duża (tzn. wynosi przynajmniej 50) funkcja `wilcox.test(stats)` oblicza $p-value$ z wykorzystaniem rozkładu normalnego z korektą na ciągłość (2.18) o ile nie zaznaczymy w opcjach inaczej. Korektę na ciągłość stosujemy po to, aby obliczenia na podstawie rozkładu normalnego były dokładniejsze.

$$z_{Wc} = \frac{|W - E(W)| - 0.5}{\sqrt{V(W)}} \tag{2.18}$$

gdzie: $E(W) = \frac{n_1 n_2}{2}$ - to średnia oraz $V(W) = \frac{n_1 n_2 (n_1+n_2+1)}{12}$ - to wariancja.

```
> wilcox.test(Y~G,exact=F)

        Wilcoxon rank sum test with continuity correction

data:  Y by G
W = 48, p-value = 0.024
alternative hypothesis: true location shift is not equal to 0
```

Można także obliczyć wartość $p - value$ bez korekty (2.19). To rozwiązanie jest sugerowane w przypadku dużych prób – powyżej 50 obserwacji.

$$z_W = \frac{W - E(W)}{\sqrt{V(W)}} \tag{2.19}$$

```
> wilcox.test(Y~G,exact=F,correct=F)

        Wilcoxon rank sum test

data:  Y by G
W = 48, p-value = 0.02259
alternative hypothesis: true location shift is not equal to 0

> wilcox_test(Y~G)

        Asymptotic Wilcoxon Mann-Whitney Rank Sum Test

data:  Y by G (1, 2)
Z = -2.2802, p-value = 0.02259
alternative hypothesis: true mu is not equal to 0
```

Należy podkreślić że funkcja `wilcox.test(stats)` działa dobrze tylko w przypadku rozpatrywania rozkładów ciągłych. Natomiast nie jest ona zalecana, gdy mamy do czynienia z dyskretnymi rozkładami tzn. takimi w których wartości liczbowe mogą się powtarzać.

```
# dane z rangami wiązanymi:
> set.seed(312)
> y= sample(1:10,20,T); u=factor(rep(1:2,c(11,9)))
# przybliżona wartość p-value:
> wilcox.test(y~u)

        Wilcoxon rank sum test with continuity correction

data:  y by u
W = 69, p-value = 0.1452
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(x = c(8L, 7L, 10L, 5L, 9L, 7L, 3L, 5L, 5L,  :
  cannot compute exact p-value with ties
```

Ponieważ w danych występują rangi wiązane funkcja `wilcox.test(stats)` nie może obliczyć dokładnej wartości $p-value$. Informuje nas o tym komunikat: *cannot compute exact p-value with ties*. Z kolei funkcja `wilcox_test(coin)` radzi sobie bez problemu.

$$z_{Wt} = \frac{W - E(W)}{\sqrt{V(W) - \frac{n_1 n_2 \sum_{i=1}^{c}(t_i^3 - t_i)}{12(n_1+n_2)(n_1+n_2-1)}}} \tag{2.20}$$

```
# dokładna wartość p-value:
> wilcox_test(y~u, distribution= "exact")

        Exact Wilcoxon Mann-Whitney Rank Sum Test

data:  y by u (1, 2)
Z = 1.495, p-value = 0.1426
alternative hypothesis: true mu is not equal to 0
# symulacja monte-carlo:
> wilcox_test(y~u, distribution= approximate(B=100000))

        Approximative Wilcoxon Mann-Whitney Rank Sum Test

data:  y by u (1, 2)
Z = 1.495, p-value = 0.143
alternative hypothesis: true mu is not equal to 0
```

Często zdarza się, że jesteśmy zainteresowani czy wartości jednej grupy wydają się być większe niż z drugiej grupy. W takiej sytuacji należy zweryfikować hipotezę o prawdopodobieństwie z wykorzystaniem testu statystycznego Brunnera-Munzela. Jest on szczególnie przydatny gdy dane pochodzą z rozkładu skośnego, wariancje w grupach są różne a wielkość próby wynosi co najmniej 10 w grupie.

Statystyka testu Brunnera-Munzela:

$$BM = \frac{n_1 n_2(\bar{r}_2 - \bar{r}_1)}{(n_1+n_2)\sqrt{n_1 s_1^2 + n_2 s_2^2}} \tag{2.21}$$

Wariancje w k grupach s_k^2 (k = 1,2) są zdefiniowane w następujący sposób:

$$s_k^2 = \frac{1}{n_k - 1} \sum_{i=1}^{n_k} \left(r_{ki} - w_{ki} - \bar{r}_k + \frac{n_k + 1}{2} \right)^2 \tag{2.22}$$

gdzie n_k ($k = 1, 2$) oznacza liczebność k-tej próby, $\bar{r}_k$ ($k = 1, 2$) oznacza średnią rangę k-tej próby z próbki zbiorczej, r_k ($k = 1, 2$) to rangi dla k-tej próby z próbki zbiorczej, w_k to rangi dla k-tej próby. Statystyka BM ma asymptotyczny rozkład t-Studenta z df_{BM} stopniami swobody (2.23) obliczane za pomocą formuły Satterthwaite:

$$df_{BM} = \frac{\left(n_1 \cdot s_1^2 + n_2 \cdot s_2^2 \right)^2}{\frac{\left(n_1 \cdot s_1^2 \right)^2}{n_1 - 1} + \frac{\left(n_2 \cdot s_2^2 \right)^2}{n_2 - 1}} \tag{2.23}$$

Hipotezę zerową o równości stochastycznej można sformułować w poniższy sposób:

$$\begin{array}{ll} H_0: & P(X < Y) = P(X > Y) \\ H_1: & P(X < Y) \neq P(X > Y) \end{array} \tag{2.24}$$

Względna wielkość efektu:

$$p = P(X < Y) + \frac{1}{2} P(X = Y) \tag{2.25}$$

to prawdopodobieństwo, że obserwacje w grupie pierwszej X mają tendencję do mniejszej wartości niż w grupie drugiej Y. Estymator tego prawdopodobieństwa (2.26) może być wyrażony wzorem:

$$\hat{p} = \frac{\bar{r}_2 - (n_2 + 1) \cdot 0.5}{n_1} \tag{2.26}$$

gdzie $\bar{r}_2$ oznacza średnią rangę z drugiej próby z próbki zbiorczej

lub

$$\hat{p} = 1 - \frac{W}{n_1 n_2} \tag{2.27}$$

gdzie W oznacza statystykę Wilcoxona–Manna–Whitneya (2.13).

Z wykorzystaniem funkcji `brunner.munzel.test(lawstat)` możemy testować następujące hipotezy:

- obserwacje w grupie 1 są zazwyczaj takie same jak w grupie 2:

$$\begin{array}{ll} H_0: & p = 0.5 \\ H_1: & p \neq 0.5 \end{array} \tag{2.28}$$

- obserwacje w grupie 1 są zazwyczaj większe niż w grupie 2:

$$\begin{array}{ll} H_0: & p = 0.5 \\ H_1: & p < 0.5 \end{array} \tag{2.29}$$

- obserwacje w grupie 1 są zazwyczaj mniejsze niż w grupie 2:

$$\begin{array}{ll} H_0: & p = 0.5 \\ H_1: & p > 0.5 \end{array} \tag{2.30}$$

```
> library(lawstat)
# test Brunnera-Munzela:
> brunner.munzel.test(a,b)

        Brunner-Munzel Test

data:  a and b
Brunner-Munzel Test Statistic = 2.5943, df = 19.338, p-value = ↩
   0.01764
95 percent confidence interval:
 0.5492865 0.9584059
sample estimates:
P(X<Y)+.5*P(X=Y)
       0.7538462
```

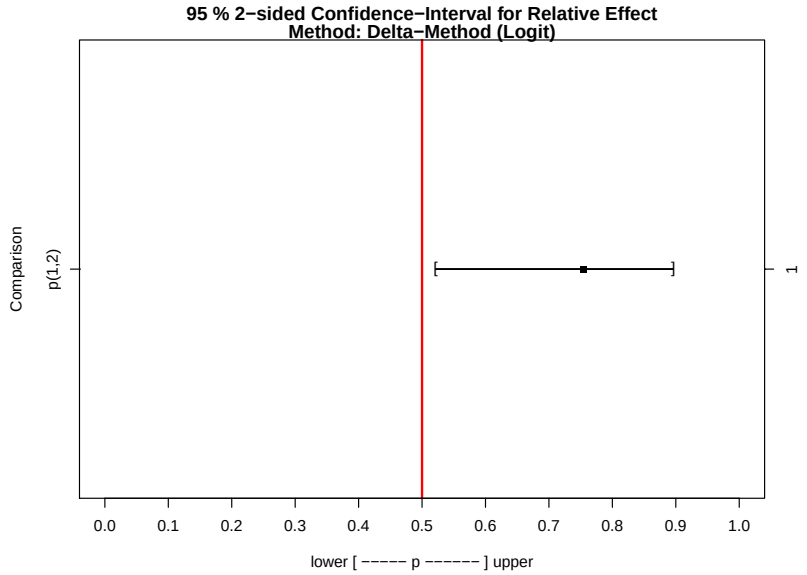
Poniżej wyniki tego samego testu ale z wykorzystaniem pakietu `WRS`:

```
> bmp(a,b)
$test.stat
[1] 2.59425

$phat
[1] 0.7538462

$dhat
[1] -0.5076923

$sig.level
[1] 0.01764079

$ci.p
[1] 0.5492865 0.9584059

$df
[1] 19.33755
```

Podobne wyniki otrzymamy wykorzystując funkcję `npar.t.test(nparcomp)`.

```
# dane:
> d=data.frame(Y,G)
> library(nparcomp)
# test:
> npar.t.test(Y~G, data=d, alternative= "two.sided")

  Nonparametric Behrens-Fisher Problem
 NOTE:
 *-----------Analysis of relative effects-----------*
 - Confidence interval for relative effect p(i,j)
      with confidence level 0.95
 - Method =  Delta-Method (Logit)
 - p.perm = p-value of the Neubert-Brunner permutation test
 - p-Values for  H_0: p(i,j)=1/2

 *----------------Interpretation--------------------*
 p(a,b) > 1/2 : b tends to be larger than a
 *------------------Wilcox.Test---------------------*
 - Asymptotic Wilcoxon Test
 - In this setup you can only test H_0:F_i = F_j
```

```
 *------------------------------------------------*
$Data.Info
  Sample Size
1      1   13
2      2   15

$Analysis.of.relative.effects
  Comparison rel.effect confidence.interval  t.value p.value p.perm
1     p(1,2)      0.754   [ 0.521 ; 0.896 ] 2.122514   0.034  0.024

$Wilcoxon.Test
  Comparison rel.effect    p.value
1     p(1,2)      0.754 0.02399546
```



Rys. 2.4. Przdział ufności dla oszacowanego prawdopodobieństwa $\hat{p} = 0.754$.

Wartość $p - value$ jest równa 0.034 ponieważ domyślną opcją jest metoda transformacji logitowej – `asy.method="logit"`. Jeśli zostałaby użyta funkcja `asy.method="t.app"` to $p - value$ byłoby równe 0.018 ponieważ wartość 0.01764 została zaokrąglona do trzech miejsc po przecinku. Dodatkowo podany jest wynik testu Wilcoxona (2.18) z wykorzystaniem funkcji `wilcox.test(stats)`. W ekstremalnych warunkach tzn. gdy liczbność próby wynosi mniej niż 10 w grupie jest zalecane stosowanie testu permutacyjnego Neuberta–Brunnera. Opcja `p.permu=TRUE` jest opcją domyślną, zatem jeśli jej nie zmienimy na `p.permu=FALSE` to będzie także podany wynik testu permutacyjnego. Funkcje z pakietu `nparcomp` mogą być wykorzystywane np. do oceny skuteczności leczenia.

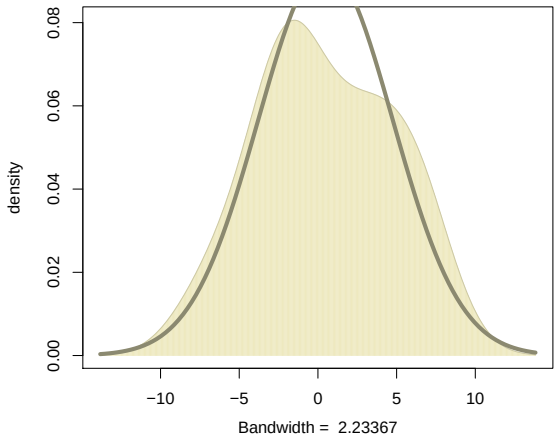
3. Kilka prób niezależnych

3.1. Test ANOVA

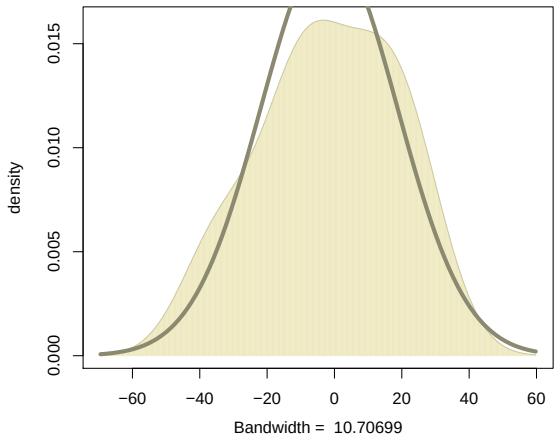
W tej części opracowania zostanie przedstawiony krótki przegląd kilku sposobów porównania więcej niż dwóch zmiennych niezależnych. Gdy zmienne w grupach mają rozkład normalny oraz równe wariancje w grupach można stosować klasyczny test

ANOVA. Natomiast gdy warunek o jednorodności wariancji nie jest spełniony należy wykorzystać test ANOVA-Welch.

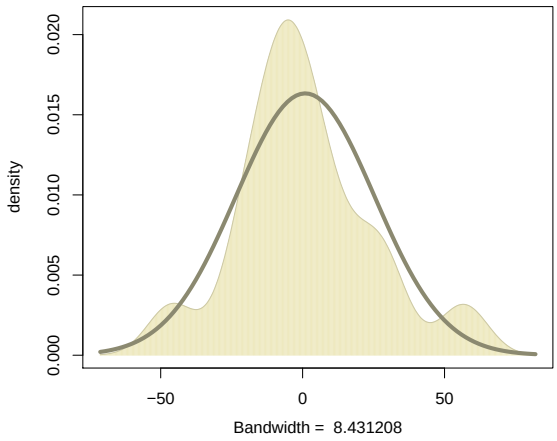
```
> set.seed(4489)
> y=c(rnorm(14,0,3),rnorm(12,5,18),rnorm(15,9,25),rnorm(11,11,1))
> g=factor(rep(LETTERS[1:4],c(14,12,15,11)))
```



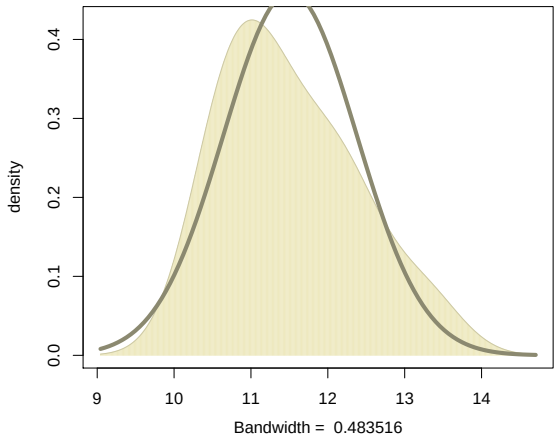
Rys. 3.1. Grupa A.



Rys. 3.2. Grupa B.



Rys. 3.3. Grupa C.

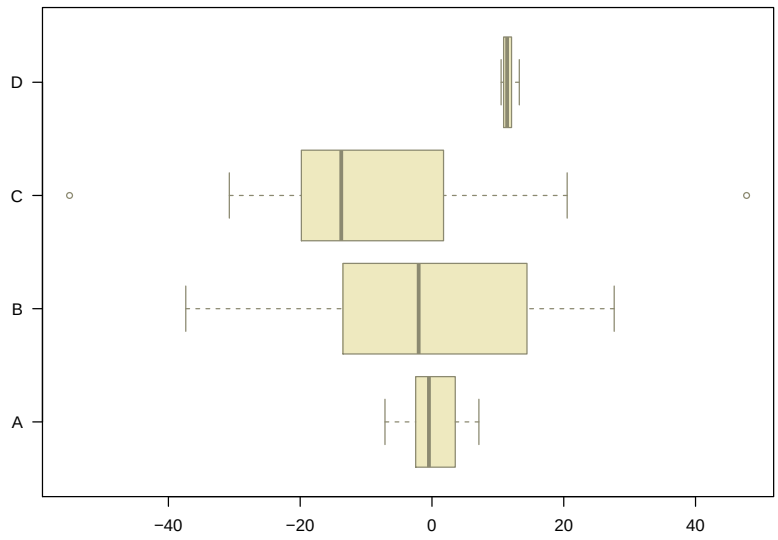


Rys. 3.4. Grupa D.

Wyniki wybranych testów normalności:

```
# p-value Shapiro-Wilk:
> tapply(y,g,function(x) shapiro.test(x)$p.value)
        A         B         C         D
0.7523433 0.8314132 0.7737059 0.4740445
# p-value E-test - bootstrap:
> library(energy)
> tapply(y,g,function(x) mvnorm.etest(x, R=199)$p.value)
        A         B         C         D
0.6926927 0.8748749 0.5365365 0.6056056
# p-value odporny Jarque-Bery - sumulacja monte-carlo:
> library(lawstat)
> tapply(y,g,function(x) rjb.test(x,option="RJB",crit.values="↩
   empirical",N=10000)$p.value)
```

```
         A          B          C          D
0.6620026  0.6964029  0.2059058  0.3455334
# p-value Neyman:
> library(ddst)
> tapply(y,g,function(x) ddst.norm.test(x,compute.p=TRUE)$p.value)
    A     B     C     D
0.919 0.639 0.404 0.313
```



Rys. 3.5. Wykres pudełkowy.

```
# p-value test Bartletta:
> bartlett.test(y~g)$p.value
[1] 2.18835e-16
# p-value test Flignera-Killeena:
> fligner.test(y~g)$p.value
[1] 4.656776e-05
# p-value test levene
> levene.test(y,g,location="trim.mean",trim.alpha=0.2,bootstrap=T,↩
   num.bootstrap=1000)$p.value
[1] 0
```

Ponieważ wariancje w grupach nie są równe zastosujemy test ANOVA–Welch.

```
# test Welch-ANOVA:
> oneway.test(y~g)

        One-way analysis of means (not assuming equal variances)

data:  y and g
F = 32.1491, num df = 3.000, denom df = 21.598, p-value = 3.786e-08
```

W przypadku gdy chcemy porównać różnice średnich (lub ilorazy) a dane pochodzą z rozkładu normalnego (nie koniecznie o równych wariancjach w grupach) to możemy skorzystać z pakietu `SimComp`. Liczebność grup nie musi być równoliczna. Warto w tym miejscu podkreślić, że testy wielokrotnych porównań z pakietu `SimComp` można stosować także w wielowymiarowej analizie wariancji – MANOVA. Dostępne testy do porównań wielokrotnych które możemy wybrać to: `"Dunnett"`,

"Tukey", "Sequen", "AVE", "GrandMean", "Changepoint", "Marcus", "McDermott", "Williams", "UmbrellaWilliams". Dodatkowo za pomocą funkcji Margin mamy możliwość określać wielkości różnic (lub ilorazów) średnich w testowanych hipotezach zerowych. Z kolei opcja ContrastMat służy do testowania własnych kontrastów.

```
# różnice średnich:
> d=data.frame(y,g)
> SimTestDiff(data= d, grp= "g", resp= "y", covar.equal= F, type= "↩
   Tukey")

Test for differences of means of multiple endpoints
Assumption: Heterogeneous covariance matrices for the groups
Alternative hypotheses: True differences not equal to the margins

comparison endpoint marg estimate statistic p.value.raw p.value.adj
     B - A        y    0  -2.0636   -0.3462      0.7352      0.9820
     C - A        y    0   0.4408    0.0688      0.9461      0.9999
     D - A        y    0  11.0286    9.4201      0.0000      0.0000
     C - B        y    0   2.5044    0.2911      0.7733      0.9894
     D - B        y    0  13.0922    2.2358      0.0470      0.1491
     D - C        y    0  10.5878    1.6775      0.1155      0.3309

# przedziały ufności:
> SimCiDiff(data= d, grp= "g", resp= "y", covar.equal= F, type= "↩
   Tukey")

Simultaneous 95% confidence intervals for differences of means of ↩
   multiple endpoints
Assumption: Heterogeneous covariance matrices for the groups

  comparison endpoint estimate lower.raw upper.raw   lower upper
1      B - A        y  -2.0636  -15.0693     10.94 -19.302 15.18
2      C - A        y   0.4408  -13.2257     14.11 -17.485 18.37
3      D - A        y  11.0286    8.5233     13.53   7.753 14.30
4      C - B        y   2.5044  -15.2130     20.22 -20.371 25.38
5      D - B        y  13.0922    0.2101     25.97  -3.840 30.02
6      D - C        y  10.5878   -2.9450     24.12  -7.069 28.24
```

Alternatywą dla testu ANOVA-Welch może być test dla średnich uciętych. Jest on dostępny w pakiecie asbio:

```
> library(asbio)
> trim.test(y,g)
$Results
  df1      df2      F*        P(>F)
1   3 13.57546 26.2426 6.469284e-06
```

Analizując powyższe wyniki możemy stwierdzić, że średnie obcięte także różnią się między sobą.

W bibliotece WRS znajdziemy więcej opcji dla testu ANOVA-trim:

```
# test dla uciętych średnich:
> w= list(A=y[g=="A"],B=y[g=="B"],C=y[g=="C"],D=y[g=="D"])
> t1way(w, tr =0.2)
$TEST
[1] 25.97634

$nu1
```

```
[1] 3

$nu2
[1] 13.57546

$n
$n$A
[1] 14

$n$B
[1] 12

$n$C
[1] 15

$n$D
[1] 11


$p.value
[1] 6.858018e-06
# test dla uciętych średnich - wersja bootstrapowa:
> t1waybt(w, tr =0.2, nboot= 2000)
$test
[1] 25.97634

$p.value
[1] 0.001
# porównania wielokrotne - wersja bootstrapowa:
> mcppb20(w, con= 0, tr= .2, nboot= 2000)

$psihat
     con.num       psihat        se   ci.lower  ci.upper p-value
[1,]       1   0.6636305 6.128106 -13.902365 20.638427   0.870
[2,]       2  10.7660538 5.168214  -7.343735 23.275691   0.115
[3,]       3 -10.8585650 1.353275 -14.255806 -7.243723   0.000
[4,]       4  10.1024233 7.795210 -17.134249 28.739353   0.294
[5,]       5 -11.5221956 5.990511 -30.819857  3.211495   0.053
[6,]       6 -21.6246188 5.004295 -33.116443 -3.450914   0.002

$crit.p.value
[1] 0.009

$con
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    1    1    0    0    0
[2,]   -1    0    0    1    1    0
[3,]    0   -1    0   -1    0    1
[4,]    0    0   -1    0   -1   -1
```

Dzięki opcji `con` mamy możliwość konstruowania własnych kontrastów. W ustawieniach domyślnych są porównywane wszystkie kombinacje dwóch średnich – `$con`.

3.2. Test Kruskala–Wallisa

Nieparametrycznym odpowiednikiem testu ANOVA jest test Kruskala–Wallisa,

$$H = \frac{12}{n(n+1)} \sum^{k} \left(\frac{R_j^2}{n_j} \right) - 3(n_j + 1) \tag{3.1}$$

gdzie: R_j – to suma rang w k-tej grupie.

W przypadku gdy występują rangi wiązane stosowany jest wzór:

$$H_t = \frac{H}{1 - \frac{\sum_{i=1}^{c}(t_i^3 - t_i)}{n^3 - n}} \tag{3.2}$$

gdzie: c – to liczba grup pomiarów wiązanych, t_i – to liczba pomiarów wiązanych w i-tej grupie pomiarów wiązanych.

```
# przykładowe dane:
> set.seed(7700)
> y=c(rexp(14,2),rexp(12,6),rexp(15,3),rexp(11,11))
> g=factor(rep(LETTERS[1:4],c(14,12,15,11)))
# asymptotyczny test:
> kruskal_test(y~g)

        Asymptotic Kruskal-Wallis Test

data:  y by g (A, B, C, D)
chi-squared = 12.0797, df = 3, p-value = 0.007115
# symulacja Monte Carlo:
> kruskal_test(y~g, distribution= approximate(B= 100000))

        Approximative Kruskal-Wallis Test

data:  y by g (A, B, C, D)
chi-squared = 12.0797, p-value = 0.00499
```

Jednym z bardziej popularnych nieparametrycznych metod porównań wielokrotnych jest przeprowadzenie serii testów Wilcoxona–Manna–Withneya (2.13) modyfikujac poziom istotnosci (np. stosując poprawkę Benjaminiego-Hochberga) dla kazdego z nich.

```
# asymptotyczny test:
> pairwise.wilcox.test(y,g,p.adjust.method="BH")

        Pairwise comparisons using Wilcoxon rank sum test

data:  y and g

  A     B     C
B 0.062 -     -
C 0.316 0.274 -
D 0.013 0.316 0.013

P value adjustment method: BH
```

Często proponowanymi rozwiązaniami w literaturze są metody podobne do procedury Tukeya. Poniżej zostanie przedstawionych kilka takich procedur. Pierwsza z nich to

konserwatywna metoda wielokrotnego porównania oparta na procedurze Bonferroniego. Jest ona dostępna w bibliotece `asbio`.

$$|\bar{R}_i - \bar{R}_j| \geqslant z_{\alpha/k(k-1)} \sqrt{\frac{n(n+1)}{12}\left(\frac{1}{n_i} + \frac{1}{n_j}\right)} \tag{3.3}$$

```
> KW.multi.comp(y, g, conf=.95)
                          Diff     Lower    Upper  Decision A.p-value
Avg.rankA-Avg.rankB  12.77381  -2.95508  28.5027     FTR H0  0.192875
Avg.rankA-Avg.rankC   4.52381 -10.33402 19.38164     FTR H0         1
Avg.rankB-Avg.rankC     -8.25 -23.73502  7.23502     FTR H0  0.959072
Avg.rankA-Avg.rankD   19.4026   3.29333 35.51186  Reject H0   0.00891
Avg.rankB-Avg.rankD   6.62879  -10.0607 23.31828     FTR H0         1
Avg.rankC-Avg.rankD  14.87879  -0.99245 30.75003     FTR H0  0.080327
```

Ta sama procedura jest zaimplementowana także w pakiecie `pgirmess`:

```
> library(pgirmess)
> kruskalmc(y, g, probs= 0.05)
Multiple comparison test after Kruskal-Wallis
p.value: 0.05
Comparisons
      obs.dif critical.dif difference
A-B 12.773810     15.72889      FALSE
A-C  4.523810     14.85783      FALSE
A-D 19.402597     16.10927       TRUE
B-C  8.250000     15.48502      FALSE
B-D  6.628788     16.68949      FALSE
C-D 14.878788     15.87124      FALSE
```

Kolejna metoda do porównań wielokrotnych znajduje się w pakiecie `agricolae`. Jest to test Conovera-Inmana.

$$|\bar{R}_i - \bar{R}_j| \geqslant t_{(N-k,\alpha/2)} \sqrt{\frac{n(n+1)}{12} \cdot \frac{n-1-H}{n-k} \cdot \left(\frac{1}{n_i} + \frac{1}{n_j}\right)} \tag{3.4}$$

```
> library(agricolae)
> kruskal(y, g, alpha= 0.05, group= F)

Study:
Kruskal-Wallis test's
Ties or no Ties

Value: 12.07972
degrees of freedom: 3
Pvalue chisq  : 0.007115012

g,  means of the ranks

         y replication
A 34.85714          14
B 22.08333          12
C 30.33333          15
D 15.45455          11
```

```
Comparison between treatments mean of the ranks

      Difference   pvalue sig       LCL      UCL
A - B  12.773810 0.021362   *  1.979830 23.56779
A - C   4.523810 0.376806     -5.672403 14.72002
A - D  19.402597 0.000932 ***  8.347586 30.45761
C - B   8.250000 0.125102     -2.376621 18.87662
B - D   6.628788 0.250298     -4.824403 18.08198
C - D  14.878788 0.008452  **  3.987124 25.77045
```

Ponieważ w analizowanych danych nie występują rangi wiązane to uzyskamy identyczne wyniki za pomocą funkcji `pairwise.t.test(stats)`:

```
> pairwise.t.test(rank(y),g,p.adjust.method="none")

        Pairwise comparisons using t tests with pooled SD

data:  rank(y) and g

  A       B       C
B 0.02136 -       -
C 0.37681 0.12510 -
D 0.00093 0.25030 0.00845

P value adjustment method: none
```

W domyślnych ustawieniach w funkcji `kruskal` nie jest stosowana żadna poprawka – opcja `p.adj="none"`. Dostępne poprawki jakie możemy zastosować to: `"holm"`, `"hochberg"`, `"bonferroni"`, `"BH"`, `"BY"`, `"fdr"`. Warto też podkreślić, że gdy zaznaczymy opcję `group=T` to będziemy mieli możliwość wyodrębnienia podobnych grup średnich.

Poniżej zostanie przedstawiony test Steel-Dwass-Chritchlow-Fligner. Kod w języku R do jego wykonania wraz z krótkim opisem testu jest dostępny w dokumencie: *Le test de Kruskal-Wallis et différents tests post hoc de comparaison par paires* w materiałach edukacyjnych opublikowanych przez F.-G. Carpentier na stronie: `http://geai.univ-brest.fr/`. Są tam również zaprezentowane już wcześniej wymienione w tym opracowaniu testy: Siegela-Castellana (3.3) oraz Conovera-Inmana (3.4).

```
# kod:
Steel.Dwass <- function(data, group)
        {
OK <- complete.cases(data, group)
data <- data[OK]
group <- group[OK]
n.i <- table(group)
ng <- length(n.i)
t <- combn(ng, 2, function(ij) {
        i <- ij[1]
        j <- ij[2]
        r <- rank(c(data[group == i], data[group == j]))
        R <- sum(r[1:n.i[i]])
        N <- n.i[i]+n.i[j]
        E <- n.i[i]*(N+1)/2
        V <- n.i[i]*n.i[j]/(N*(N-1))*(sum(r^2)-N*(N+1)^2/4)
return(abs(R-E)/sqrt(V))
})
p <- ptukey(t*sqrt(2), ng, Inf, lower.tail=FALSE)
```

```
result <- cbind(t, p)
rownames(result) <- combn(ng, 2, paste, collapse=":")
return(result)
        }
# Steel-Dwass-Chritchlow-Fligner test:
> Steel.Dwass(y,as.numeric(g))
           t          p
1:2 2.160247 0.13447671
1:3 1.047446 0.72144611
1:4 2.791989 0.02690258
2:3 1.366260 0.52062644
2:4 1.046278 0.72214477
3:4 2.776646 0.02812482
```

Po załadowaniu biblioteki `coin` i wpisaniu komendy `?kruskal_test` zostanie wyświetlony plik pomocy w którym jest przedstawiony kod za pomocą którego można wykonać test Nemenyi-Damico-Wolfe-Dunn:

```
# Nemenyi-Damico-Wolfe-Dunn test:
 if (require("multcomp")) {
    NDWD <- oneway_test(y~g, data= data.frame(y,g),
      ytrafo = function(data) trafo(data, numeric_trafo = rank),
      xtrafo = function(data) trafo(data, factor_trafo = function(x)
          model.matrix(~x - 1) %*% t(contrMat(table(x), "Tukey"))),
      teststat = "max", distribution = approximate(B = 90000))
    print(pvalue(NDWD))
    print(pvalue(NDWD, method = "single-step"))
  }
[1] 0.006555556
99 percent confidence interval:
 0.005883202 0.007280641

B - A 0.118822222
C - A 0.890366667
D - A 0.006488889
C - B 0.500588889
D - B 0.787088889
D - C 0.087477778
```

Alternatywą dla klasycznego testu Kruskala-Wallisa jest test Brunner-Dette-Munk, który jest odporny na brak normalności oraz heterogeniczność wariancji. Jest on traktowany jako heteroskedastyczna wersja testu Kruskala-Wallisa.

```
> library(lawstat)
> BDM.test(y,g)
$Q
  Levels Rel.effects
1      A   0.6607143
2      B   0.4150641
3      C   0.5737179
4      D   0.2875874

$BDM.Table
       df1      df2       F*        P(>F)
X 2.866344 45.11767 5.389113 0.003345328
```

Test Brunner-Dette-Munk charakteryzuje się także większą mocą niż test Kruskala-Wallisa.

```
> m= 10000
> statystyka = NULL
> alpha=0.05
# moc testu Kruskal-Wallis:
> for (i in 1:m) {
+ q1=sample(y[g=="A"],length(y[g=="A"]),T)
+ q2=sample(y[g=="B"],length(y[g=="B"]),T)
+ q3=sample(y[g=="C"],length(y[g=="C"]),T)
+ q4=sample(y[g=="D"],length(y[g=="D"]),T)
+ statystyka[i] = kruskal.test(c(q1,q2,q3,q4)~g)$p.value
+ }
> mean(statystyka < alpha)
[1] 0.8983
# moc testu Brunner-Dette-Munk:
> for (i in 1:m) {
+ q1=sample(y[g=="A"],length(y[g=="A"]),T)
+ q2=sample(y[g=="B"],length(y[g=="B"]),T)
+ q3=sample(y[g=="C"],length(y[g=="C"]),T)
+ q4=sample(y[g=="D"],length(y[g=="D"]),T)
+ statystyka[i] = BDM.test(c(q1,q2,q3,q4),g)$BDM.Table[4]
+ }
> mean(statystyka < alpha)
[1] 0.9159
```

Poniżej wyniki testu Brunner-Dette-Munk z wykorzystaniem funkcji `bdm(WRS)`:

```
> w=list(A=y[g=="A"],B=y[g=="B"],C=y[g=="C"],D=y[g=="D"])
> library(WRS)
> bdm(w)
$F
          [,1]
[1,] 5.389113

$nu1
[1] 2.866344

$nu2
[1] 45.11767

$q.hat
           [,1]
[1,] 0.6607143
[2,] 0.4150641
[3,] 0.5737179
[4,] 0.2875874

$p.value
             [,1]
[1,] 0.003345328
```

Jak już wcześniej zostało podkreślone (przy omawianiu testu Brunnera-Munzela) użytecznym sposobem do porównania różnic w grupach (np. badanie skuteczności leczenia) jest ocena względnych efektów. W przypadku gdy analizujemy więcej niż dwie grupy możemy skorzystać z kilku funkcji z pakietu `WRS`. Pierwsza z nich weryfikuje następującą hipotezę zerową:

$$H_0 : \quad p_{12} = p_{13} = \ldots p_{ij} = 0.5 \tag{3.5}$$

```
> library(WRS)
> wmwaov(w)
[1] 0.114
```

Do porównań wielokrotnych zostanie wykorzystana funkcja `cidM(WRS)`.

```
> cidM(w)
$test
Group Group p-value      p.crit    P(X<Y) P(X=Y)    P(X>Y)     p.hat
    1     2   0.018 0.012500000 0.2500000      0 0.7500000 0.2500000
    1     3   0.312 0.050000000 0.3857143      0 0.6142857 0.3857143
    1     4   0.004 0.010000000 0.1688312      0 0.8311688 0.1688312
    2     3   0.166 0.016666667 0.6555556      0 0.3444444 0.6555556
    2     4   0.298 0.025000000 0.3712121      0 0.6287879 0.3712121
    3     4   0.000 0.008333333 0.1757576      0 0.8242424 0.1757576
```

W przypadku gdy zmienne w grupach charakteryzują się rozkładami dyskretnymi to lepszym rozwiązaniem jest wykorzystanie funkcji `cidmulv2(WRS)`.

```
> cidmulv2(w)
$n
     [,1] [,2] [,3] [,4]
[1,]   14   12   15   11

$test
Group Group      p.hat p.ci.lower p.ci.uppper p-value      p.crit
    1     2 0.2500000 0.10817488   0.4780881   0.032 0.012500000
    1     3 0.3857143 0.20303796   0.6074695   0.330 0.025000000
    1     4 0.1688312 0.06094419   0.3886593   0.004 0.008333333
    2     3 0.6555556 0.42798727   0.8288031   0.190 0.016666667
    2     4 0.3712121 0.17188068   0.6267539   0.340 0.050000000
    3     4 0.1757576 0.06261763   0.4050001   0.007 0.010000000

$summary.dvals
     Group Group    P(X<Y) P(X=Y)    P(X>Y)
[1,]     1     2 0.2500000      0 0.7500000
[2,]     1     3 0.3857143      0 0.6142857
[3,]     1     4 0.1688312      0 0.8311688
[4,]     2     3 0.6555556      0 0.3444444
[5,]     2     4 0.3712121      0 0.6287879
[6,]     3     4 0.1757576      0 0.8242424
```

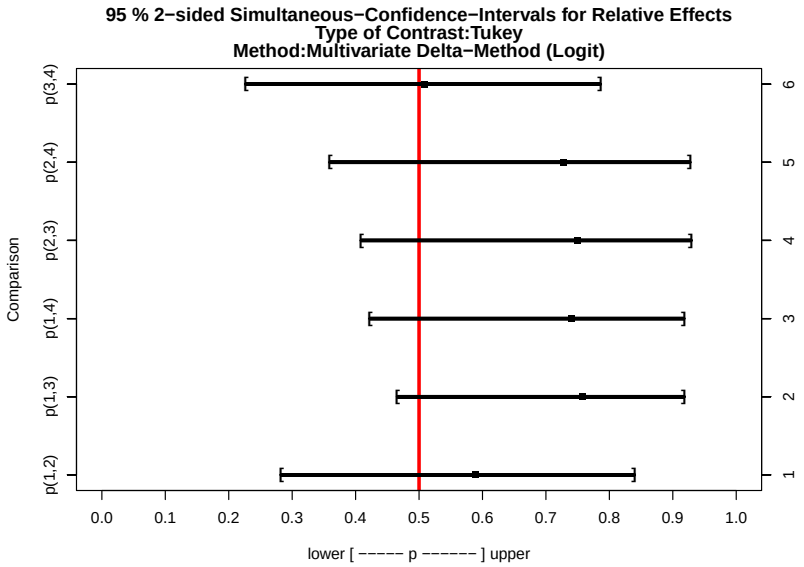
Więcej informacji można znaleźć w dokumencie *Inferences about a Probabilistic Measure of Effect Size When Dealing with More Than Two Groups* którego autorem jest Rand R. Wilcox.

Warto także zwrócić uwagę na procedurę dostarczoną w pakiecie `nparcomp` która jest rozszrzeniem metody `npar.t.test(nparcomp)` na więcej niż dwie niezależne próbki.

```
> d=data.frame(y,g)
> library(nparcomp)
> nparcomp(y~g, data= d, conflevel=0.95, alternative= "two.sided", ↩
   type= "Tukey")

  Nonparametric Multiple Comparison Procedure based on relative ↩
     contrast effects , Type of Contrast : Tukey
 NOTE:
```

```
 *-------------------Weight Matrix------------------*
 - Weight matrix for choosen contrast based on all-pairs comparisons

 *-----------Analysis of relative effects-----------*
 - Simultaneous Confidence Intervals for relative effects p(i,j)
      with confidence level 0.95
 - Method = Multivariate Delta-Method (Logit)
 - p-Values for  H_0: p(i,j)=1/2

 *----------------Interpretation--------------------*
 p(a,b) > 1/2 : b tends to be larger than a
 *--------------Mult.Distribution-------------------*
 - Equicoordinate Quantile
 - Global p-Value
 *------------------------------------------------*
$weight.matrix
    p(1,2) p(1,3) p(1,4) p(2,3) p(2,4) p(3,4) p(2,1) p(3,1) p(4,1)
C 1      1      0      0      0      0      0      0      0      0
C 2      0      1      0      0      0      0      0      0      0
C 3      0      0      1      0      0      0      0      0      0
C 4      0      0      0      1      0      0      0      0      0
C 5      0      0      0      0      1      0      0      0      0
C 6      0      0      0      0      0      1      0      0      0
    p(3,2) p(4,2) p(4,3)
C 1      0      0      0
C 2      0      0      0
C 3      0      0      0
C 4      0      0      0
C 5      0      0      0
C 6      0      0      0

$Data.Info
  Sample Size
1      1   14
2      2   12
3      3   15
4      4   11

$Analysis.of.relative.effects
  Comparison rel.effect confidence.interval    t.value p.value.adj
1     p(1,2)      0.589   [ 0.283 ; 0.839 ] 0.71980906       0.924
2     p(1,3)      0.757   [ 0.466 ; 0.918 ] 2.30089092       0.101
3     p(1,4)      0.740   [ 0.423 ; 0.917 ] 1.98599819       0.201
4     p(2,3)      0.750   [ 0.409 ; 0.929 ] 1.92929570       0.224
5     p(2,4)      0.727    [ 0.36 ; 0.927 ] 1.62242903       0.384
6     p(3,4)      0.509   [ 0.227 ; 0.785 ] 0.07428372       1.000
  p.value.unadjusted
1              0.472
2              0.021
3              0.047
4              0.054
5              0.105
6              0.941

$Mult.Distribution
  Quantile p.Value.global
1 2.575063          0.101
```

```
$Correlation
[1] NA
```



Rys. 3.6. Przdziały ufności dla oszacowanych prawdopodobieństw.

Więcej informacji na temat praktycznego wykorzystania funkcji `nparcomp(nparcomp)` można znaleźć w dokumencie *Nonparametric Evaluation of Quantitative Traits in Population-Based Association Studies when the Genetic Model is Unknown.*