Manual - pl

Aby uruchomić program, należy w terminalu Windows, znajdując się w folderze zawierającym projekt wpisać frazę:
C:\"Program Files"\R\R-4.0.4\bin\Rscript.exe [nazwa pliku z projektem.R] [adres strony internetowej, z której pobierane są dane dla projektu]
Bardzo ważnym jest aby uruchamiać program poprzez Rscript.exe, a nie przez R.exe. W przypadku uruchamiania przez R.exe - nie wszystko wczyta się tak jak powinno i niektóre funkcje mogą nie działać.

Na samym początku paczki pobierane są ze strony internetowej i od razu ładowane są biblioteki.

TRYB WSADOWY

Najpierw mam krótki kod sprawdzający czy został podany chociaż jeden argument dla programu. Jeśli nie, to wyświetlany jest komunikat, że należy podać co najmniej 1 argument. Argument, który podawany jest przy trybie wsadowym wczytywany jest do pliku za pomocą funkcji `read.csv2()`. Wczytane dane wypisywane są na ekran.

GRUPY BADANE I ILOŚĆ PACJENTÓW

Tworzę dwa wektory - `grupy` oraz `ile badanych`, które będą przechowywały informacje o nazwach grup oraz ilości pacjentów w poszczególnej grupie.
Za pomocą funkcji `unique()` zapełniam wektor `grupy`.
Natomiast `ile_badanych` wypełniany jest za pomocą pętli, która iteruje po wektorze 'grupy' znajdując dla każdej grupy liczbę pacjentów z tą samą nazwą grupy. Wykorzystuję do tego funkcję `length(which(DANE$grupa == grupy[i]))`.

PRZYGOTOWANIE DANYCH WEJŚCIOWYCH -> ZASTĘPOWANIE BRAKÓW

Na początek tworzę dwie zmienne - `początek` równy 1 i `koniec` równy pierwszej pozycji z wektora `ile_badanych`. Następnie zaczynam iterować po moich `grupach` i po kolumnach tabeli z danymi. Sprawdzam, czy dla danej kolumny w przedziale wierszów od `początku` do `końca` znajduje się jakakolwiek komórka z wartością NA. Jeśli tak jest, to zapisuje indeks tej komórki do wektora `braki`. Natomiast tą komórkę z NA nadpisuję wartością średnią wyliczoną dla konkretnej grupy, w której się znajduje ta komórka. Ta wartość wyliczana jest na podstawie przedziału od `początku` do `końca` wierszy (ponieważ ten przedział zawiera dokładnie tyle wierszy, ile jest pacjentów w danej grupie).

Uznałam nadpisywanie średnimi za lepszy pomysł niż usuwanie danych z brakami, ponieważ spowodowałoby to powstanie niepełnych danych, a co za tym idzie mniej dokładną analizę.

Po nadpisaniu, na ekran wypisywana jest informacja, że zostały znalezione braki, dla jakiej grupy, w której kolumnie oraz jaki jest indeks tego braku. Komunikat informuje również z której grupy średnią został nadpisany brak. *Poniżej przykład.*

```
"Znaleziono braki w grupie: CHOR1 w kolumnie MON"
"Indeksy z brakami: "
5
"Braki w przedziale indeksow 1 - 25 zastapiono srednia z grupy CHOR1"
```

Po przejściu przez wszystkie kolumny danej grupy moje zmienne początek i koniec zmieniają wartości: początek staje się końcem+1 a koniec jest wynikiem dodania następnej pozycji z wektora ile_badanych do poprzedniej. W ten sposób mogę poruszać się po konkretnych wierszach danych należących do odpowiedniej grupy.

PRZYGOTOWANIE DANYCH WEJŚCIOWYCH -> WARTOŚCI ODSTAJĄCE

Tutaj również używam swoich zmiennych początek i koniec. Są one mi potrzebne przy wyliczaniu konkretnych średnich i odchyleń standardowych dla grup. Dodatkowo tworzę 4 pomocnicze puste wektory. Ponownie iteruję najpierw po grupach, a następnie po kolumnach. Tworzę pusty wektor Odstajace oraz sprawdzam, czy kolumna w której się aktualnie znajduję jest typu numeric. Jeśli tak, to tworzę zmienne - min równe średnia-odchylenie oraz max równe średnia+odchylenie. Zaczynam iterować po poszczególnych komórkach danej grupy w kolumnie. Jeśli znajduję wartość, która jest większa od max lub mniejsza od min, dodaję ją do wektora Odstajace.

Po przejściu przez wszystkie komórki z danej grupy z kolumny, sortuję moje Odstajace oraz do moich 4 pomocniczych wektorów dodaję po kolei: nazwę grupy, nazwę kolumny, najmniejszą wartość Odstajacych oraz największą wartość Odstajacych.

Po przejściu wszystkich grup oraz kolumn, tworzę z moich 4 pomocniczych wektorów ramkę danych, aby ułatwić czytanie wartości odstających.

Przykład

	grupa	parametr	Min_odstajaca	Max_odstajaca
1	CHOR1	wiek	17.000000	43.0000
2	CHOR1	hsCRP	16.406900	42.6499
3	CHOR1	ERY	33.000000	33.0000
4	CHOR1	PLT	128.000000	336.0000
5	CHOR1	HGB	9.504900	14.4990
6	CHOR1	HCT	0.280000	0.4050
7	CHOR1	MCHC	32.555600	36.8742
8	CHOR1	MON	0.480000	1.5200
9	CHOR1	LEU	6.790000	16.8100
10	CHOR2	wiek	21.000000	42.0000
11	CHOR2	hsCRP	0.335089	19.2124
12	CHOR2	ERY	3.250000	5.0400
13	CHOR2	PLT	91.000000	456.0000
14	CHOR2	HGB	9.827100	22.2318
15	CHOR2	HCT	0.042300	0.0423
16	CHOR2	MCHC	32.887800	38.8674
17	CHOR2	MON	7.000000	7.0000
18	CHOR2	LEU	7.950000	16.5900
19	KONTROLA	wiek	23.000000	48.0000
20	KONTROLA	hsCRP	0.758440	14.3951
21	KONTROLA	ERY	3.090000	5.0500
22	KONTROLA	PLT	147.000000	434.0000
23	KONTROLA	HGB	9.504900	13.2102
24	KONTROLA	HCT	0.279000	0.3890
25	KONTROLA	MCHC	32.057300	36.0437
26	KONTROLA	MON	0.350000	1.2500
27	KONTROLA	LEU	4.830000	17.4600

STATYSTYKA OPISOWA

Iteruję po moich kolumnach, sprawdzając przy tym czy nazwa kolumny nie jest nazwą grupy (ponieważ nie robię statystyki dla parametru "grupa"). Następnie sprawdzam, czy moja kolumna jest typu numeric. Jeśli nie, to wyświetlam komunikat, że parametr jest niemierzalny. Jeśli kolumna jest typu numeric, to wyświetlam, że parametr jest mierzalny i przechodzę do statystyki. Wykorzystuję tutaj funkcję summarise() z grupowaniem group_by(). Moje statystyki opisowe, które wybrałam to: średnia, mediana, rozstęp międzykwartalny, wariancja oraz odchylenie stndardowe.

```
podsumowanie <- group_by(DANE, grupa) %>%
  summarise(
    Ilosc_badanych = n(),
    Srednia = format(round(mean(DANE[,i]), 2), nsmall = 2),
    Mediana = format(round(median(DANE[,i]), 2), nsmall = 2),
    Rozstep_miedzykwartalowy = format(round(IQR(DANE[,i]), 2), nsmall = 2),
    Wariancja = format(round(var(DANE[,i]), 2), nsmall = 2),
    Odchylenie_standardowe = format(round(sd(DANE[,i]), 2), nsmall = 2)
  )
```

To podsumowanie wypisywane jest na ekran.

PORÓWNYWANIE GRUP

Funkcje

Dla porównywania grup jak i dla korelacji opisywanej później stworzyłam funkcje odpowiadające testom lub graficznym przedstawieniom tych testów.

Rozklad_normalny -> Funkcja przyjmuje jako argumenty numer kolumny, z której tabeli danych ma wyliczać oraz z którego wektora grup(ile ma być grup testowanych). Iterując po grupach wykonywany jest test Shapiro dla każdej z nich a wynik p.value zapisywany jest do wektora p.val. Funkcja na końcu zwraca wektor z wartościami.

Graficzny_rozklad_normalny -> Funkcja przyjmuje jako argumenty numer kolumny, oraz z której tabeli brać dane. Tworzy ona graficzną ocenę zgodności z rozkładem normalnym. Jeśli wykres ma kształt dzwonu, to można założyć, że występuje rozkład normalny.

Jednorodnosc_wariancji -> Funkcja przyjmuje jako argumenty numer kolumny oraz z której tabeli mają być pobierane dane. Wykorzystując grupowanie wykonywany jest test Levene a wartość p.value, którą dostajemy zapisywana jest do zmiennej p.value, a tą z kolei zwraca funkcja.

Funkcje dla testów dla więcej niż 2 grup (to jest: Test_Kruskala, Test_Dunna, Test_Anova, Test_Tukeya i Test_Chisq) jako argument przyjmują już tylko numer kolumny, z czego Testy Kruskala, Anova i Chisq opierają się na tym samym kodzie co test jednorodności. Wyświetlają test oraz zapisują wyniki p.value do zmiennej, która jest zwracana przez funkcję.

Test_Dunna i Test_Tukey-> W tych funkcjach tworzone są tabelę dla testów. Tabela zawiera wyniki testu (dunnTest(....)$res dla testu Dunna oraz TukeyHSD(aov(...))$grupa dla testu Tukey). Tabela jest wyświetlana a następnie iteruję po niej sprawdzając każdy wynik. Jeśli któraś wartość jest mniejsza od 0.05, to znaczy że istnieją istotne różnice w parametrze między grupami. Wtedy wypisywany jest parametr oraz między jakimi grupami jest różnica.

Funkcja dla dokładnie 2 grup (to jest: Test_Wilcoxona, Test_Studenta, Test_Welcha) jako argument również przyjmują tylko numer kolumny. Są już one tak ustawione żeby przyjmować konkretne dane dla dwóch grup, które tworzę później w kodzie. Każda funkcja działa na tej samej zasadzie, czyli do zmiennej przekazywana jest wartość p.value z testu. Test jest wyświetlany a wartość p.value jest zwracana.

O testach będzie jeszcze więcej napisane w części porównawczej.

Porównanie grup niezależnych			
Ilość porównywanych grup	Zgodność z rozkładem normalnym	Jednorodność wariancji	Wybrany test
2	TAK	TAK	test t-Studenta (dla gr. niezależnych)
		NIE	test Welcha
	NIE	-	test Wilcoxona (Manna-Whitneya)
>2	TAK	TAK	test ANOVA (*post hoc* Tukeya)
		NIE	test Kruskala-Wallisa (*post hoc* Dunna)
	NIE	-	

Posiłkując się powyższą tabelą, stworzyłam porównania dla >2 grup oraz 2 grup

Porównywanie więcej niż 2 grup

Mając do dyspozycji dane z kilkoma grupami dokonuję porównania dla więcej niż 2 grup niezależnych.

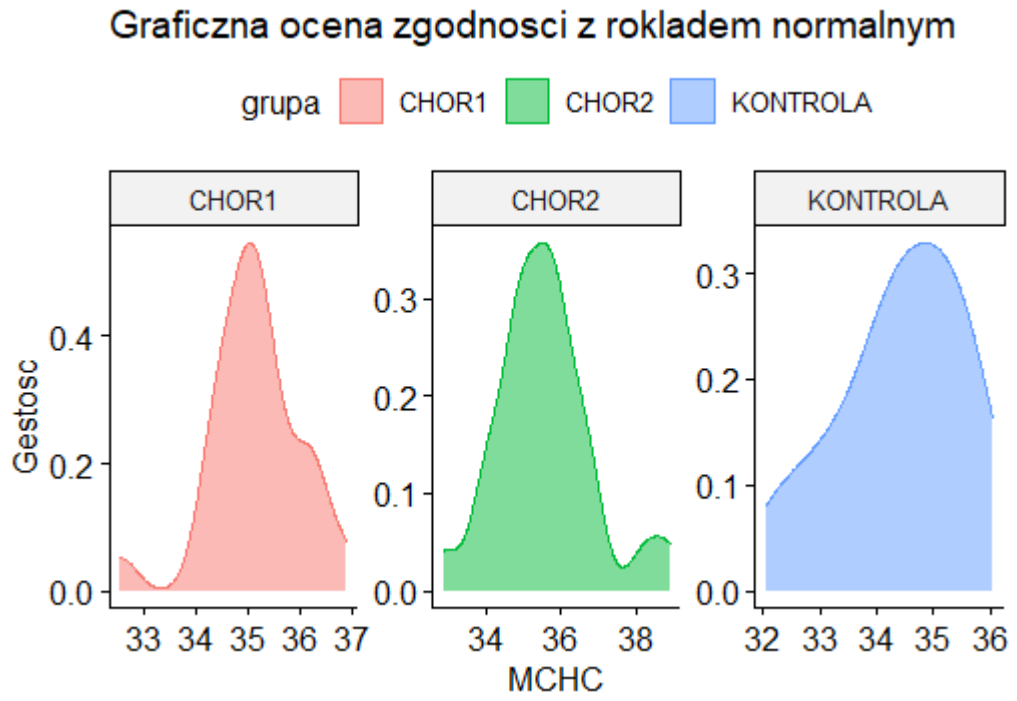
Na początku tworzę wektory numery_nieparametryczne_wiele i numery_parametryczne_wiele, które będą mi potrzebne dopiero w części związanej z korelacjami.

Moje porównanie polega na iterowaniu po kolumnach tabeli danych i sprawdzania czy dany parametr dla grup spełnia wymagania testów. Jeśli już na początku mój parametr okaże się nominalny to uruchamiana jest funkcja testu Chisq. Jeśli p.value > 0.05 to nie ma różnic.

```
        Pearson's Chi-squared test

data:  DANE$grupa and DANE[, numer]
X-squared = 0.42857, df = 2, p-value = 0.8071

[1] 0.8071177
```

Jeśli natomiast parametr jest mierzalny, to przechodzę do testu rozkładu normalnego. Zwracam wektor wartości p.value dla grup oraz wyświetlają się wykresy zgodności z rozkładem.

```
"DLA PARAMETRU MCHC"
"Wartosci rozkladu normalnego"
0.2266247 0.2832569 0.2741520
"Grupy sa zgodne z rozkladem normalnym"
```



Sprawdzam za pomocą funkcji sum() czy wartości p.value wszystkie są większe od 0.05. Jeśli tak jest to wartość sum() powinna wynieść tyle ile ma ilość grup (ponieważ liczymy dla wszystkich grup). W przykładzie wyżej sum() wynosi wartość 3, czyli tyle ile jest grup porównywanych. Oznacza to, że grupy dla parametru MCHC są zgodne z rozkładem normalnym. Dodatkowo, jak już wcześniej wspomniałam, można wnioskować zgodność po kształtach wykresów.

Jeśli test na rozkład normalny wykaże, że parametr jest zgodny - można przejść do testu jednorodności wariancji. Funkcja odpowiadająca za niego zwraca jedynie wartość p.value dla testu. Na jej podstawie określane są dalsze kroki. Jeśli p.value z testu jednorodności jest większe od 0.05, to możemy uznać, że występuje jednorodność. Przeprowadzany jest wtedy parametryczny test Anova.
Na poniższym zdjęciu przykład użycia testu Anova dla parametru MCHC:

```
[1] 0.00185981
            Df Sum Sq Mean Sq F value  Pr(>F)
grupa        2  16.90   8.448    6.87 0.00186 **
Residuals   72  88.55   1.230
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

[1] "Test ANOVA wykryl, ze sa roznice miedzy tymi grupami - TEST TUKEY"
```

Test Anova wykrywa, że są różnice między grupami, wtedy kiedy wartość p.value (w tym przykładzie - 0,00186) jest mniejsza od 0.05. Wtedy należy przeprowadzić jest tzw. post hoc test Tukey:

```
[1] "Test TUKEY:"
                     diff        lwr         upr       p adj
CHOR2-CHOR1     0.4232228 -0.3274109  1.17385653 0.372940393
KONTROLA-CHOR1 -0.7261892 -1.4768229  0.02444453 0.060043259
KONTROLA-CHOR2 -1.1494120 -1.9000457 -0.39877827 0.001352322
[1] "Sa istotne roznice statyczne w parametrze diff w wierszu 2"
[1] "Sa istotne roznice statyczne w parametrze diff w wierszu 3"
[1] "Sa istotne roznice statyczne w parametrze lwr w wierszu 1"
[1] "Sa istotne roznice statyczne w parametrze lwr w wierszu 2"
[1] "Sa istotne roznice statyczne w parametrze lwr w wierszu 3"
[1] "Sa istotne roznice statyczne w parametrze upr w wierszu 2"
[1] "Sa istotne roznice statyczne w parametrze upr w wierszu 3"
[1] "Sa istotne roznice statyczne w parametrze p adj w wierszu 3"
```

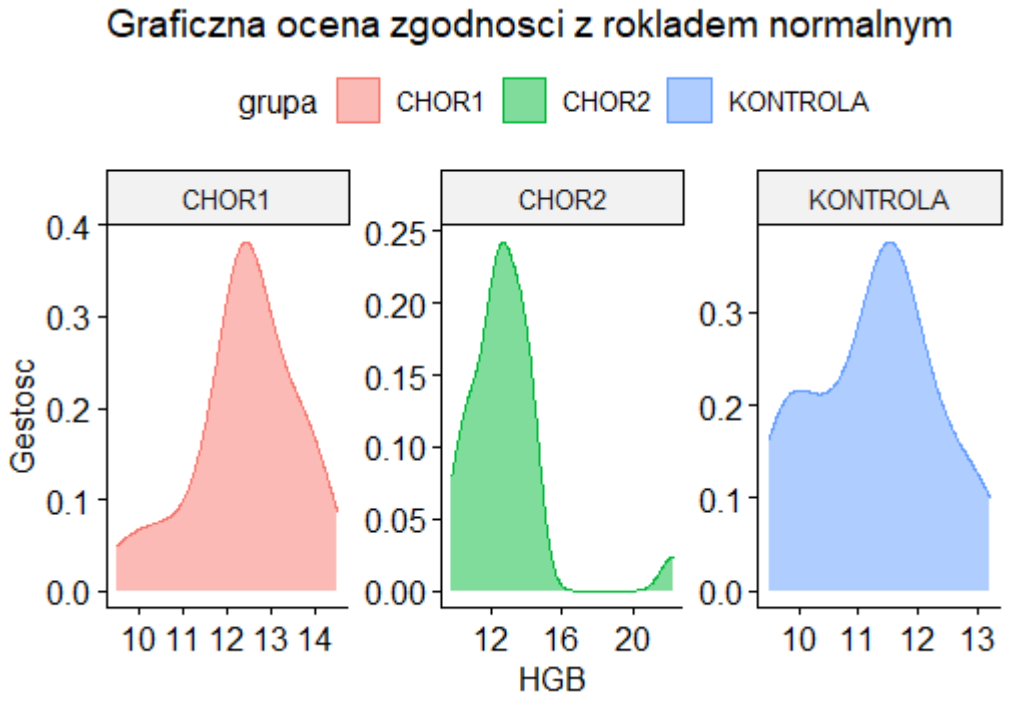
Test Tukey jest już ostatnim testem, po nim nie ma więcej w tym kierunku. Wyświetla on tablicę parametrów dla poszczególnych par grup. Jeśli wartość parametru dla którejś pary jest mniejsza od 0.05, to program informuje że są istotne różnice statystyczne.

Istnieją jeszcze inne opcje testów. Nieparametryczny test Kruskala wykonywany jest w dwóch sytuacjach:

- Gdy parametr jest zgodne z rozkładem normalnym, ale nie z jednorodnością wariancji
- Gdy parametr nie jest zgodny z rozkładem normalnym

Na przykładzie parametru HGB niezgodnego z rozkładem normalnym:

```
"DLA PARAMETRU HGB"
"Wartosci rozkladu normalnego"
5.073768e-01 3.885087e-05 3.944054e-01
"Grupy nie sa zgodne z rozkladem normalnym - TEST KRUSKALA"
```



Z przykładu wynika, że parametr HGB nie jest zgodny z rozkładem normalnym (druga wartość p.value < 0.05 / zielony wykres nie ma kształtu dzwonu). Przechodzimy zatem do nieparametrycznego testu Kruskala:

```
[1] "Test Kruskala:"

        Kruskal-Wallis rank sum test

data:  DANE[, numer] by grupa
Kruskal-Wallis chi-squared = 14.345, df = 2, p-value = 0.0007673

[1] 0.0007673315
```

Wartość p.value, którą dostajemy z testu sprawdzamy ponownie. Jeśli jest mniejsza od 0.05, to należy wykonać test post hoc Dunna:

```
[1] "Test KRUSKALA wykryl, ze sa roznice miedzy  grupami - TEST DUNNA"
[1] "Test DUNNA:"
        Comparison          Z      P.unadj       P.adj
1    CHOR1 - CHOR2 -0.1201601 0.9043562952 0.904356295
2 CHOR1 - KONTROLA  3.2183431 0.0012893348 0.002578670
3 CHOR2 - KONTROLA  3.3385033 0.0008423104 0.002526931
[1] "Roznica miedzy CHOR1 - CHOR2 w parametrze ' Z ' jest istotna statycznie"
[1] "Roznica miedzy CHOR1 - KONTROLA w parametrze ' P.unadj ' jest istotna statycznie"
[1] "Roznica miedzy CHOR2 - KONTROLA w parametrze ' P.unadj ' jest istotna statycznie"
[1] "Roznica miedzy CHOR1 - KONTROLA w parametrze ' P.adj ' jest istotna statycznie"
[1] "Roznica miedzy CHOR2 - KONTROLA w parametrze ' P.adj ' jest istotna statycznie"
```

Przy sprawdzaniu rozkładu normalnego i jednorodności, jeśli któryś parametr spełnia oba testy, to jego numer kolumny zapisywany jest w wektorze numery_parametryczne_wiele. Jeśli natomiast nie spełnia jednego testu lub obu, to wtedy trafia do wektora numery_nieparamteryczne_wiele.

Porównywanie dokładnie 2 grup

W przypadku porównywania 2 grup, trzeba najpierw określić, które 2 grupy będziemy porównywać. Najpierw program wyświetla komunikat, ile jest grup oraz jakie to grupy. Każe wybrać dwie grupy po numerach. Aby zapewnić interakcje użytkownika z programem została użyta funkcja:

```
con <- if (interactive()) stdin() else file('stdin')
message('Wybierz grupe pierwsza:')
grupa1 <- scan(file=con, sep=',', nlines=1, quiet=TRUE)
```

zarówno dla zmiennej o nazwie grupa1 jak i grupa2. Obie zmienne są wczytywane do programu.

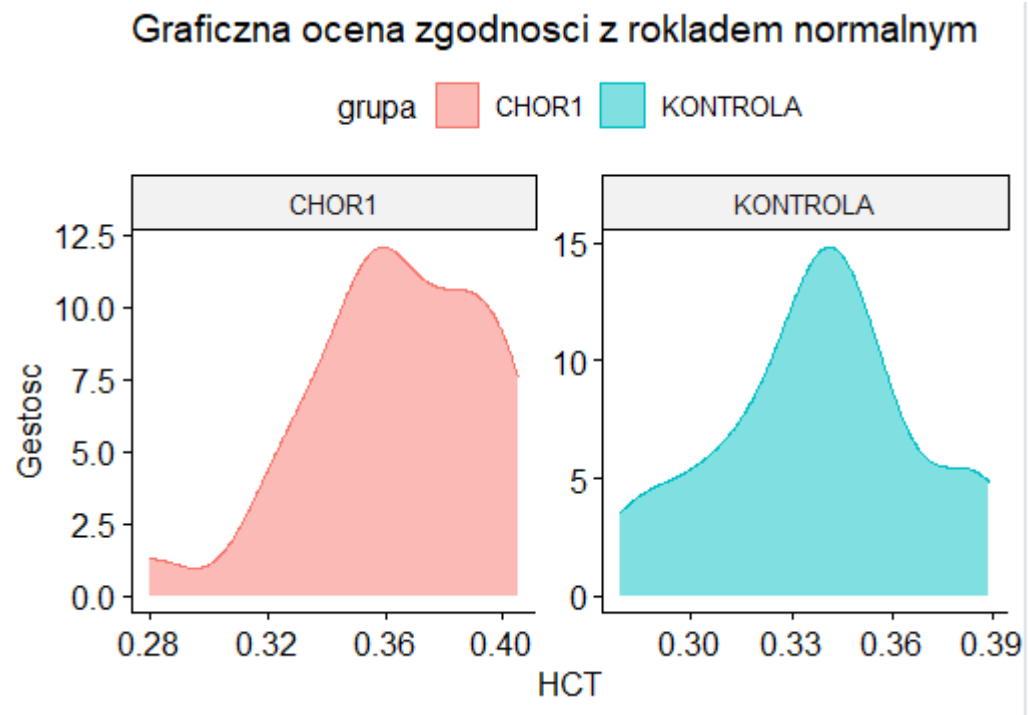
Następnie wczytana tabela z danymi jest dzielona w zależności od grup funkcją split(dane, dane$grupa). Tworzę nową zmienną o nazwie Dwie_grupy, a jej wartością jest tabela powstała ze złączonych ze sobą dwóch mniejszych tabel grupa1 i grupa2 funkcją full_join(). Aby jeszcze zachować porządek tworzę wektor grupy2 i nadaje mu nazwy grup z tabeli Dwie_grupy za pomocą funkcji unique().

Od teraz posługuję się więc danymi z Dwie_grupy oraz nazwami grup z grupy2 .

Tutaj również tworzę wektory potrzebne dla korelacji: numery_parametryczne_2, numery_nieparametryczne_2. Wypełniane są na tej samej zasadzie, co przy porównywaniu więcej niż 2 grup. To jest: w momencie, kiedy parametr jest równocześnie zgodny z rozkładem normalnym jak i jednorodnością wariancji - trafia do wektora numery_parametryczne_2. Jeśli jest inaczej - do wektora numery_nieparametryczne_2.

Kod nie różni się w tym przypadku od poprzedniego. Iteruję po kolumnach mojej tabeli Dwie_grupy. Sprawdzam czy parametr jest zgodny z rozkładem normalnym oraz jednorodnością warinacji.

```
"DLA PARAMETRU HCT"
"Wartosci rozkladu normalnego"
0.2126689 0.6084101
"Grupy sa zgodne z rozkladem normalnym"
```



Powyższy przykład pokazuje rozkład normalny dla parametru HCT. HCT jest również zgodny z jednorodnością wariancji, co oznacza że musi zostać wykonany parametryczny test T.Studenta. Przykład jego użycia na następnej stronie. Jeśli p.value uzyskane podczas tego testu jest mniejsze od 0.05, to można wnioskować, że istnieją różnice statystyczne.

```
[1] "Test T.STUDENTA"

        Two Sample t-test

data:  Dwie_grupy[, numer] by grupa
t = 3.0103, df = 48, p-value = 0.004152
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.008634214 0.043365786
sample estimates:
   mean in group CHOR1 mean in group KONTROLA
               0.36356                0.33756

[1] 0.004151741
```

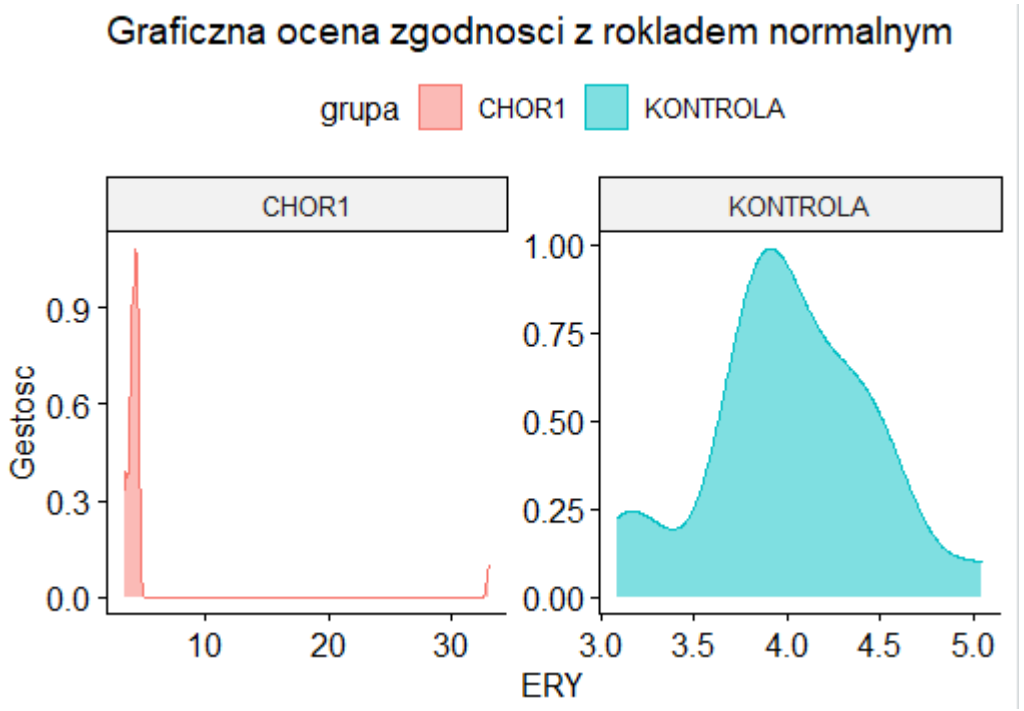
W przypadku, w którym parametr spełnia założenia rozkładu normalnego, ale jednorodności już nie, mamy do czynienia z nieparametrycznym testem Welcha.

```
        Welch Two Sample t-test

data:  Dwie_grupy[, numer] by grupa
t = 1.1664, df = 24.302, p-value = 0.2548
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.036866  3.736066
sample estimates:
   mean in group CHOR1 mean in group KONTROLA
                5.3628                 4.0132

[1] 0.2547663
```

Jeśli p.value jest mniejsze od 0.05 - są różnice statystyczne między grupami.

Jeśli parametr nie spełni już na początku założenia o rozkładzie normalnym:

```
"DLA PARAMETRU ERY"
"Wartosci rozkladu normalnego"
2.653537e-10 6.267790e-01
"Grupy nie sa zgodne z rozkladem normalnym"
```



Wtedy naszym naszym testem jest nieparametryczny test Wilcoxona:

```
[1] "Test WILCOXONA"

        Wilcoxon rank sum test with continuity correction

data:  Dwie_grupy[, numer] by grupa
W = 413, p-value = 0.05224
alternative hypothesis: true location shift is not equal to (

[1] 0.05224323
```

Jeśli p.value jest mniejsze od 0.05, to są różnice statystyczne.

TESTY KORELACJI

Tutaj mam utworzone 4 funkcje: dwie dotyczące testów korelacji, jedna do zwizualizowania testów oraz jedna do określania współczynnika.

Funkcje do testów korelacji przyjmują jako argumenty: która tabela z danymi, z którego pakietu grup, która grupa, parametr1, parametr2.
Przekazują one wyniki testu korelacji(p.value i R) do funkcji współczynnika korelacji. Natomiast funkcja współczynnika dostaje p i r z testu i poddaje je sprawdzeniu. Jeśli p jest większe od 0.05 to nie ma korelacji. Natomiast jeśli jest mniejsze od 0.05 to występuje korelacja pomiędzy parametrami. Wtedy funkcja sprawdza jeszcze rodzaj i siłę korelacji za pomocą if'ów:

- Rodzaj korelacji:
  - r > 0 - dodatnia korelacja
  - r = 0 - brak

    - r < 0 - ujemna korelacja
- Siła korelacji
    - -1 < r < -0.7 - bardzo silna ujemna
    - -0.7 < r < -0.5 - silna ujemna
    - -0.5 < r < -0.3 - ujemna średnia
    - -0.3 < r < -0.2 - słaba ujemna
    - -0.2 < r < 0.2 - brak
    - 0.2 < r < 0.3 - słaba dodatnia
    - 0.3 < r < 0.5 - dodatnia średnia
    - 0.5 < r < 0.7 - silna dodatnia
    - 0.7 < r < 1 - bardzo silna dodatnia

Funkcja wypisuje odpowiedni komunikat na ekran: Jaki jest rodzaj i siła.

Ostatnią funkcją dotyczącą korelacji jest funkcja graficzna. Jest ona zbudowana aby przyjmować argumenty dla korelacji zarówno Spearmana jak i Pearsona(argument metoda, który przyjmuje funkcja). Oprócz tego przyjmuje te same argumenty co funkcje testujące korelacje.

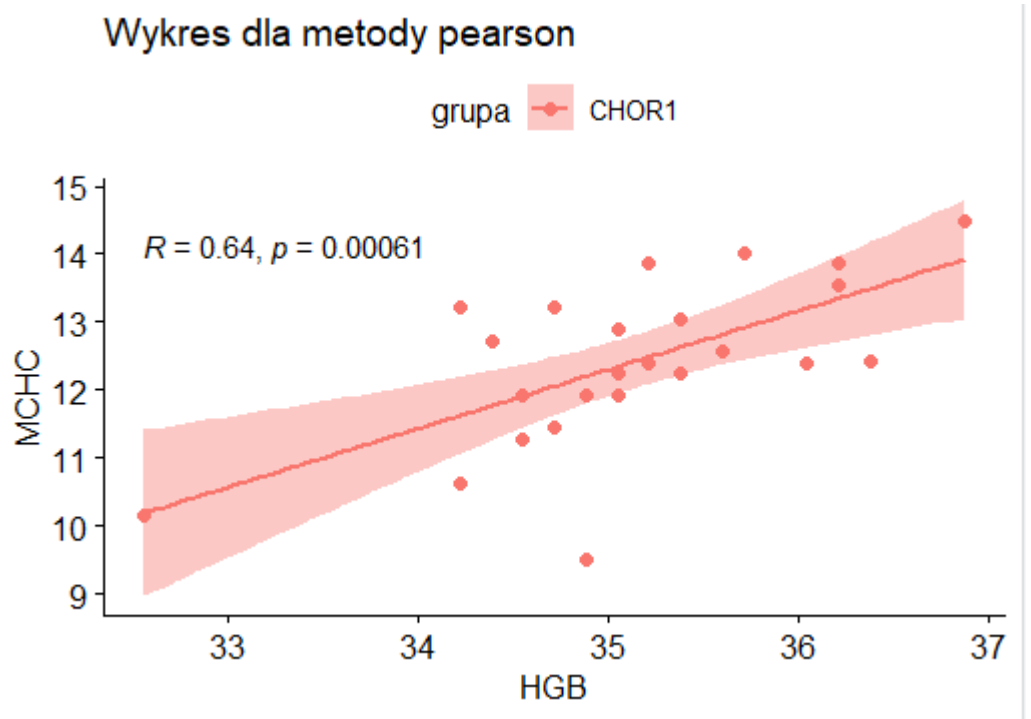
Testy korelacji wykonuję osobno dla wszystkich grup i dla konkretnych 2. Zrobiłam tak, ponieważ przy porównywaniu wszystkich grup do siebie, niektóre parametry wychodziły niezgodne z rozkładem normalnym, ale przy tylko dwóch grupach te same parametry spełniały już założenia rozkładu normalnego. Wykorzystuję tutaj stworzone wcześniej wektory przy porównywaniu grup.

Kody nie różnią się zbytnio od siebie, przyjmowane są tylko różne argumenty. Iteruję dwa razy po kolumnach mojej tabeli danych bez powtórzeń (a więc odwiedzam dwa różne parametry). Sprawdzam, czy parametry są typu numeric. Jeśli są to sprawdzam, czy oba te parametry znajdują się w wektorze numery_parametryczne_wiele(Ten wektor stosuję w przypadku testów dla >2 grup). Jeśli tak to uruchamiana jest funkcja testu parametrycznego korelacji Pearsona. Jeśli natomiast choć jeden parametr nie należy do wektorów parametrycznych, to wykonywany jest wtedy test nieparametryczny korelacji Spearmana.

W przypadku testów przeprowadzanych dla 2 grup jest tak samo, tylko wtedy brane są pod uwagę inne dane: tabela Dwie_grupy, grupy2 oraz sprawdzany wektor to numery_parametryczne_2.

Poniżej znajduje się przykład testu Pearsona dla parametrów MCHC - HGB dla grupy CHOR1, w przypadku brania pod uwagę tylko dwóch grup - CHOR1 i KONTROLA:

```
"Grupa - CHOR1 | Parametry: MCHC - HGB"
"Wyniki:"
"P.value = 0.000611749123444521 < 0.05 - Korelacja miedzy zmiennymi"
"Rodzaj korelacji:"
"r = 0.637305862411678 > 0 - Korelacja Dodatnia"
"Sila korelacji:"
"0.637305862411678 - Silna korelacja dodatnia"
```



Pragnę dodać, że na każdym etapie analizy program wyświetla komunikaty, co aktualnie jest sprawdzane. Wyniki są wyświetlane na ekran z dopiskami, dla jakiej grupy, jaki parametr itd.