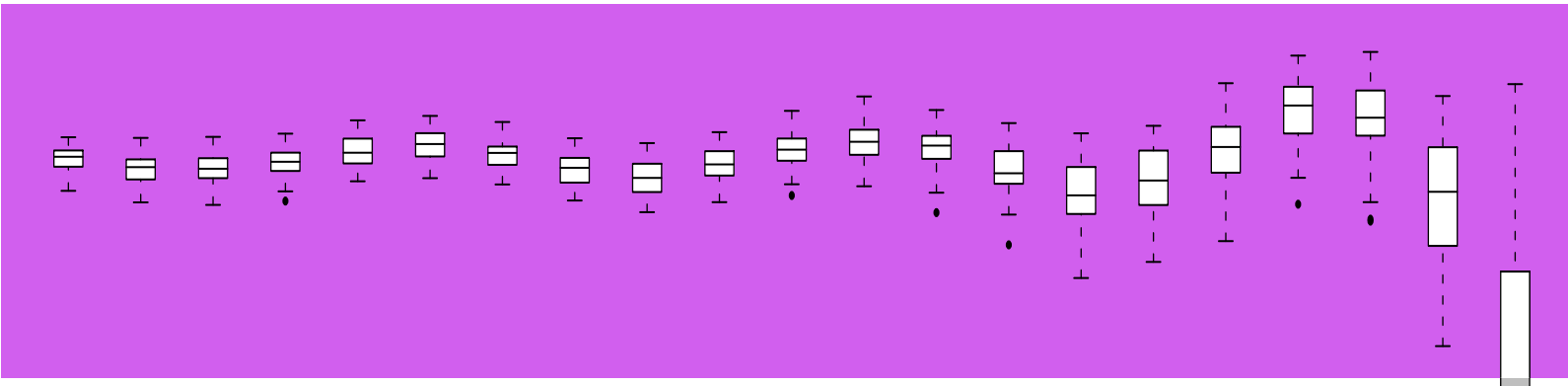
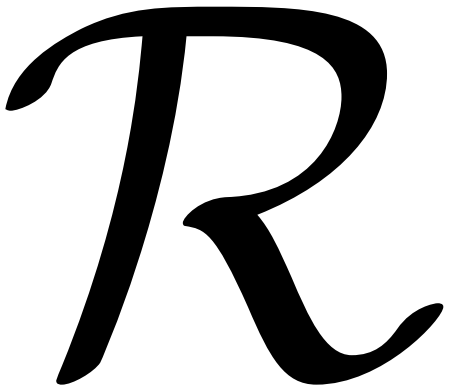
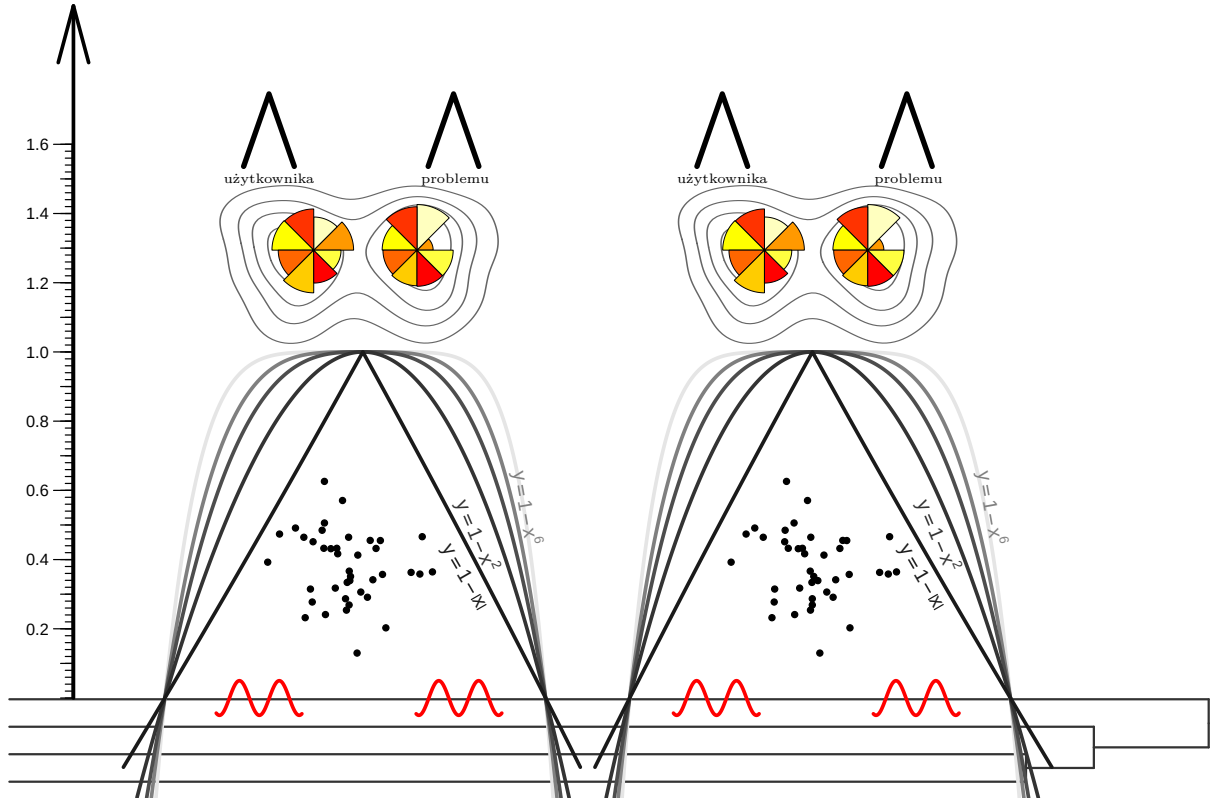


Na przełaj przez Data Mining z pakietem



wersja 0.3, robocza



Spis treści

Rozdział 1

Kilka słów zamiast wstępu

na przełaj „najkrótszą drogą, nie trzymając się wytyczonej trasy”
Słownik Języka Polskiego PWN

„Kto chodzi na skróty, ten nie śpi w domu”
Mądrość ludowa ;-)

„Nie ma drogi na skróty, do miejsca do którego, warto dojść”
Edith

W połowie roku 2010 te notatki zostały „otwarte”. Osoby chcące je rozwijać są do tego gorąco zachęcane. Poniżej znajduje się aktualna lista autorów kolejnych rozdziałów

- Rozdziały 2 do 6 przygotowuje Przemek Biecek, `przemyslaw.biecek@gmail.com`.
- Rozdziały 7 do 13 przygotowuje Krzysiek Trajkowski, `seaproject@poczta.onet.pl`.

Notatki zatytułowane „na przełaj” przygotowane są jako materiały pomocnicze. Każdy może z nich korzystać, pamiętając jednak że:

- Notatki zostały przygotowane tak, bym ułatwić wykonywanie pewnych analiz w R, nacisk został położony na poznanie pakietu R jako narzędzia do wykonywania danych analiz.
- Notatki NIE ZOSTAŁY przygotowane tak by uczyć się z nich metodologii. NIE SĄ tutaj dyskutowane zagadnienia teoretyczne, nie ma tu wyprowadzeń, nie ma sprawdzenia niektórych założeń. Od tego są książki, staram się w bibliografii zamieszać lepsze pozycje które udało mi się znaleźć.

- Notatki przygotowuję ucząc się danych metod, NIE SĄ więc to materiały zebrane przez eksperta, czasami nie są to nawet materiały czytane po raz drugi. Chętnie usłyszę co w takich notatkach powinno się znaleźć dodatkowo, co jest niejasne lub co jest błędem tak merytorycznym jak i językowym.

Analiza danych to nie tylko klasyczna statystyka z zagadnieniami estymacji i testowania (najczęściej wykładanymi na standardowych kursach statystyki). Znaczny zbiór metod analizy danych nazywany technikami eksploracji danych lub data mining dotyczy zagadnień klasyfikacji, identyfikacji, analizy skupień oraz modelowania złożonych procesów. Właśnie te metody będą nas interesować w poniższym rozdziale.

To paraphrase provocatively, 'machine learning is statistics minus any checking of models and assumptions'.

Brian D. Ripley (about the difference between machine learning and statistics) fortune(50)

Data mining to szybko rosnąca grupa metod analizy danych rozwijana nie tylko przez statystyków ale głównie przez biologów, genetyków, cybernetyków, informatyków, ekonomistów, osoby pracujące nad rozpoznawaniem obrazów, myśli i wiele innych grup zawodowych. Podobnie jak w poprzednich rozdziałach nie będziemy dokładnie omawiać poszczególnych algorytmów ani szczegółowo dyskutować kontekstu aplikacyjnego danej metody. Zakładamy, że czytelnik zna te metody, zna ich założenia i podstawy teoretyczne ale jest zainteresowany w jakich funkcjach pakietu R są one zaimplementowane. Stąd też skrótowe traktowanie wielu zagadnień. Osoby szukające więcej informacji o sposobie działania poszczególnych algorytmów znajdą z pewnością wiele pozycji poświęconym konkretnym metodom, szczególnie polecam książki [26] oraz [25].

Przedstawione metody zostały podzielone na trzy grupy. Pierwsza z nich to metody związane z zagadnieniem redukcji wymiaru. Te metody są również wykorzystywane do ekstrakcji cech oraz do wstępnej obróbki danych, stanowią więc często etap pośredni w analizach. Druga grupa metod związana jest z zagadnieniem analizy skupień (w innej nomenklaturze nazywanym uczeniem bez nadzoru). Przedstawiane będą metody hierarchiczne oraz grupujące. Ostatnia grupa metod związana jest z zagadnieniem dyskryminacji (w innej nomenklaturze - uczeniem pod nadzorem lub klasyfikacją).

Rozdział 2

Redukcja wymiaru

Metody redukcji wymiaru umożliwiają przedstawienie obserwacji w przestrzeni o zadanym wymiarze, niższym niż w przypadku oryginalnych danych.

Przedstawimy wybrane metody na przykładzie danych z różnych województw (zbiór danych daneGUS, patrz ostatni rozdział). Dla każdego województw zebrano informację o liczbie studentów w tych województwach w rozbiciu na 16 grup kierunków. Każde województwo jest więc opisane wektorem 16 liczb.

Przypuśćmy, że chcemy przedstawić graficznie te województwa. Jak uwzględnić wszystkie 16 parametrów? Jednym z rozwiązań jest użycie metod redukcji wymiaru i liczby zmiennych z 16 do 2. Opisanie każdego z województw dwoma liczbami ułatwi przedstawianie graficzne tych województw.

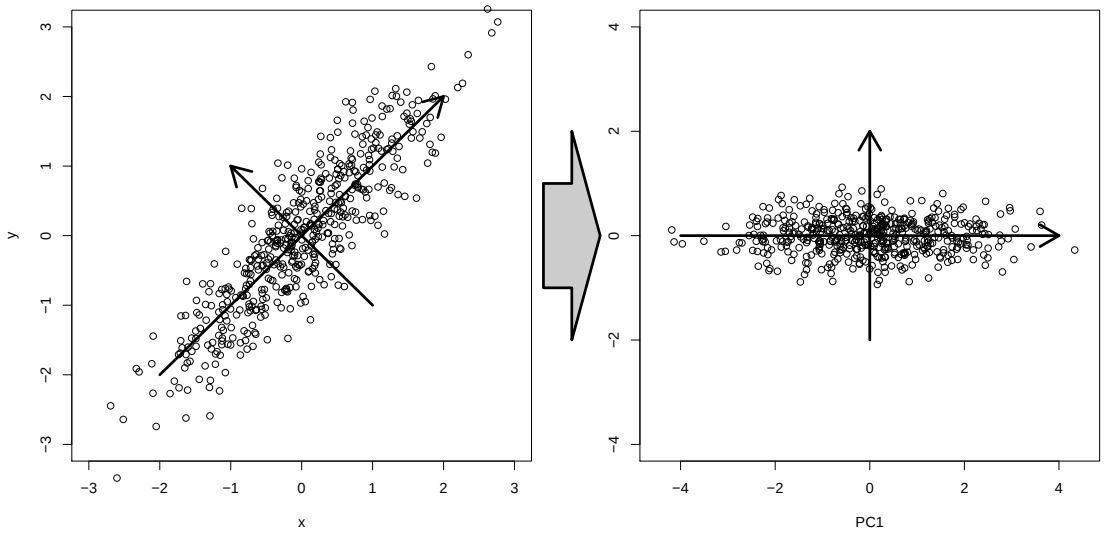
Więc do dzieła!

Redukcja wymiaru często jest pośrednim etapem w zagadnieniu klasyfikacji, analizy skupień czy regresji. W określonych sytuacjach pozwala na poprawę skuteczności tych metod, zwiększa stabilność a czasem pozwala na uwzględnienie w analizach dużej liczby zmiennych. Jest też popularnie wykorzystywaną metodą do wizualizacji wielowymiarowych zmiennych, dane są redukowane do przestrzeni dwuwymiarowej, w której już łatwo je przedstawić na wykresie. Metody z tej grupy są również nazywane metodami ekstrakcji cech, ponieważ w wyniku redukcji wymiaru tworzone są nowe cechy, które mogą być wykorzystane do innych zagadnień.

2.1 Analiza składowych głównych (PCA, ang. Principal Components Analysis)

Analiza składowych głównych służy do wyznaczania nowych zmiennych, których możliwie mały podzbiór będzie mówił możliwie dużo o całej zmienności w zbiorze danych. Nowy zbiór zmiennych będzie tworzył bazę ortogonalną w przestrzeni cech. Zmienne będą wybierane w ten sposób by pierwsza zmienna odwzorowywała możliwie dużo zmienności w danych (po zrzutowaniu obserwacji na ten wektor, chcemy by wariancja rzutów była najwyższa). Po wyznaczeniu pierwszej zmiennej wyznaczamy drugą, tak by była ortogonalna do pierwszej, i wyjaśniała możliwie dużo pozostałej zmienności, kolejną zmienną wybieramy tak by była ortogonalna do dwóch pierwszych itd.

Tak uzyskany zbiór wektorów tworzy bazę ortogonalną w przestrzeni cech, a co więcej pierwsze współrzędne wyjaśniają większość zmienności w obserwacjach. Celem metody składowych głównych jest więc znalezienie transformacji układu współrzędnych, która lepiej opisze zmienność pomiędzy obserwacjami. Przykład takiej transformacji pokazujemy na rysunku 2.1. Przedstawiamy obserwacje w oryginalnym układzie współrzędnych (lewy rysunek) i w nowym układzie współrzędnych (prawy rysunek).



Rysunek 2.1: Przykład transformacji zmiennych z użyciem metody PCA.

W pakiecie R dostępnych jest kilka implementacji metody składowych głównych. Przedstawione poniżej najpopularniejsze wersje znajdują się w funkcjach `prcomp(stats)` i `princomp(stats)` z pakietu `stats`. Inne popularne wersje znaleźć można w funkcjach `PCA(FactoMineR)`, `cmdscale(MASS)` lub `pca(pcurve)`.

Poszczególne implementacje różnią się metodami wykorzystanymi do znalezienia nowego układu zmiennych. W przypadku funkcji `prcomp()` nowe zmienne wyznaczane są z z użyciem dekompozycji na wartości osobliwe SVD. Ten sposób wyznaczania składowych głównych jest zalecany z uwagi na dużą dokładność numeryczną. W funkcji `princomp()` składowe główne są wyznaczane poprzez wektory własne macierzy kowariancji pomiędzy zmiennymi, używana jest więc dekompozycja spektralna. Teoretyczne właściwości wyznaczonych składowych głównych będą identyczne, jednak w określonych sytuacjach otrzymane wyniki dla poszczególnych funkcji mogą się różnić.

Poniższy kod pokazuje w jaki sposób działa funkcja `princomp()`. Wyznaczane są wektory własne macierzy kowariancji, tworzą one macierz przekształcenia dla danych.

```
kowariancja = cov(dane)
eig = eigen(kowariancja)
noweDane = dane %*% eig$vectors
```

Poniższy kod pokazuje w jaki sposób działa funkcja `prcomp()`. Wykorzystywana jest dekompozycja SVD.

```
svdr = svd(dane)
noweDane = dane %*% svdr$v
```

Obu funkcji do wyznaczania składowych głównych używa się w podobny sposób, kolejne argumenty określają zbiór danych (można wskazać macierz, ramkę danych, lub formułę określającą które zmienne mają być transformowane) oraz informacje czy zmienne powinny być uprzednio wycentrowane i przeskalowane. To czy dane przed wykonaniem analizy składowych głównych mają być przeskalowane zależy od rozwiązywanego problemu, w większości sytuacji skalowanie pozwala usunąć wpływ takich artefaktów jak różne jednostki dla poszczególnych zmiennych. W obiektach przekazywanych jako wynik przez obie funkcje przechowywane są podobne informacje, choć w polach o różnej nazwie. Wynikiem funkcji `prcomp()` jest obiekt klasy `prcomp`, którego pola są wymienione w tabeli 2.1.

Tabela 2.1: Pola obiektu klasy *prcomp*.

`$sdev`	Wektor odchyleń standardowych dla obserwacji. Kolejne zmienne odpowiadają odchyleniom standardowym liczonym dla kolejnych składowych głównych.
`$rotation`	Macierz obrotu przekształcająca oryginalny układ współrzędnych w nowy układ współrzędnych.
`$center`	Wektor wartości wykorzystanych przy centrowaniu obserwacji.
`$scale`	Wektor wartości wykorzystanych przy skalowaniu obserwacji.
`$x`	Macierz współrzędnych kolejnych obserwacji w nowym układzie współrzędnych, macierz ta ma identyczne wymiary co oryginalny zbiór zmiennych.

Dla obiektów klasy `prcomp` dostępne są przeciążone wersje funkcji `plot()`, `summary()`, `biplot()`. Poniżej przedstawiamy przykładowe wywołanie tych funkcji. Graficzny wynik ich działania jest przedstawiony na rysunku 2.2. Lewy rysunek przedstawia wynik dla funkcji `plot()` a prawy przedstawia wynik funkcji `biplot()` wykonanych dla argumentu klasy `prcomp`. Na lewym rysunku przedstawione są wariancje wyjaśnione przez kolejne wektory nowego układu współrzędnych. Ta informacja jest podawana również przez funkcje `summary()`. Prawy rysunek przedstawia biplot, na którym umieszczone są dwa wykresy. Jeden przedstawia indeksy obserwacji przedstawione na układzie współrzędnych określonych przez dwie pierwsze składowe główne (w tym przypadku dwie współrzędne wyjaśniają około 50% całej zmienności). Drugi rysunek przedstawia kierunki w których działają oryginalne zmienne, innymi słowy przedstawiają jak wartość danej zmiennej wpływa na wartości dwóch pierwszych składowych głównych.

Jeżeli wektory mają przeciwne zwroty to dane zmienne są ujemnie skorelowane (nie można jednak ocenić wartości korelacji), jeżeli zwroty są prostopadłe to zmienne

są nieskorelowane, a jeżeli zwroty są bliskie to zmienne są dodatnio skorelowane.

```
> # przygotowujemy dane, usuwamy zmienne jakościowe i brakujące przypadki
> dane  = na.omit(daneO[,-c(4,5,6,7)])
> # wykonujemy analizę składowych głównych, normalizując wcześniej zmienne
> wynik = prcomp(dane, scale=T)
>
> # jak wygląda obiekt z wynikami od środka
> str(wynik)
List of 5
 $ sdev    : num [1:5] 1.153 1.068 0.963 0.916 0.873
 $ rotation: num [1:5, 1:5]  0.5016  0.0935 -0.4244  0.4878 -0.5671 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5] "Wiek" "Rozmiar.guza" "Wezly.chlonne" "Okres.bez.
     wznowy" ...
  .. ..$ : chr [1:5] "PC1" "PC2" "PC3" "PC4" ...
 $ center  : Named num [1:5]   45.417    1.271    0.417   37.406 2640.896
  ..- attr(*, "names")= chr [1:5] "Wiek" "Rozmiar.guza" "Wezly.chlonne" "
     Okres.bez.wznowy" ...
 $ scale   : Named num [1:5]    6.206    0.447    0.496    9.527 3616.045
  ..- attr(*, "names")= chr [1:5] "Wiek" "Rozmiar.guza" "Wezly.chlonne" "
     Okres.bez.wznowy" ...
 $ x       : num [1:96, 1:5] -1.5446  0.0105 -1.4565 -1.2352 -1.2541 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:96] "1" "2" "3" "4" ...
  .. ..$ : chr [1:5] "PC1" "PC2" "PC3" "PC4" ...
 - attr(*, "class")= chr "prcomp"
> # ten wykres przedstawia ile wariancji jest wyjaśnione przez kolejne
    zmienne
> plot(wynik)
> # zamiast obrazka możemy tę informację mieć przedstawioną jako ramkę
    danych
> summary(wynik)
Importance of components:
                          PC1   PC2   PC3   PC4   PC5
Standard deviation     1.153 1.068 0.963 0.916 0.873
Proportion of Variance 0.266 0.228 0.185 0.168 0.153
Cumulative Proportion  0.266 0.494 0.680 0.847 1.000
> # narysujmy biplot dla tych wyników
> biplot(wynik)
```

I jeszcze przykład dla danych GUSowskich

```
> # przygotowujemy dane, wybieramy tylko kolumny dotyczące studentów
> dane  = daneGUS[,5:19]

> # wykonujemy analizę składowych głównych
> wynik = prcomp(dane, scale=T)
```

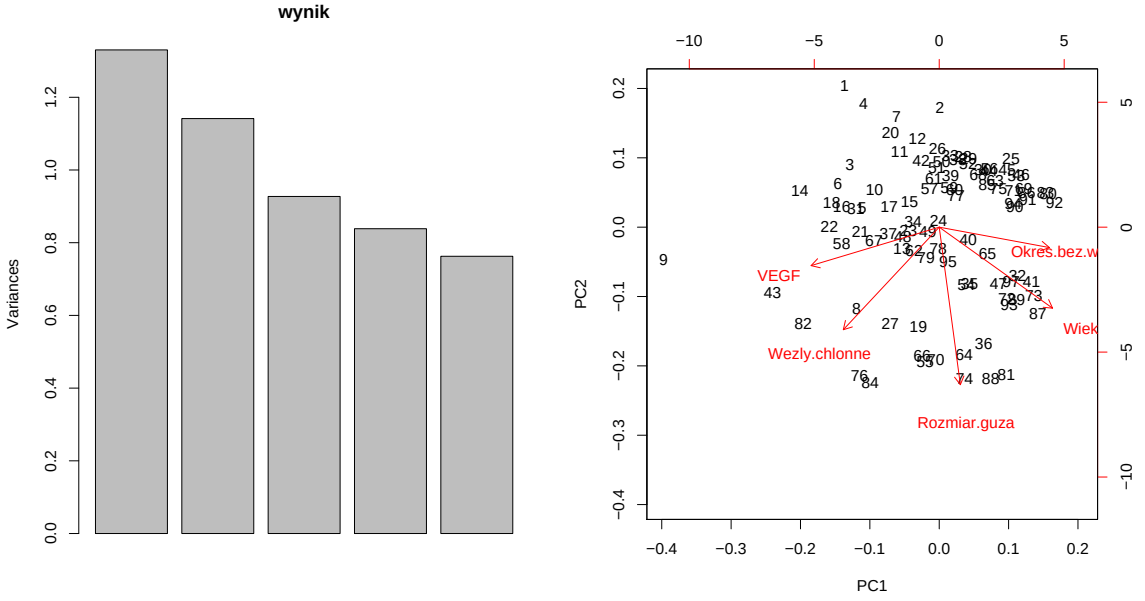
```
> # ten wykres przedstawia ile wariancji jest wyjaśnione przez kolejne
    zmienne
> plot(wynik)

> # zamiast obrazka możemy tę informację mieć przedstawioną jako ramkę
    danych
> summary(wynik)
Importance of components:
                          PC1   PC2    PC3    PC4    PC5
Standard deviation     3.327 1.199 0.8443 0.7418 0.5666 ...
Proportion of Variance 0.738 0.096 0.0475 0.0367 0.0214 ...
Cumulative Proportion  0.738 0.834 0.8815 0.9182 0.9396 ...

> # narysujmy biplot dla tych wyników
> biplot(wynik)
```

Zobaczmy jak wygląda macierz przekształcenia. Można z niej odczytać w jaki sposób poszczególne współrzędne wpływają na kolejne składowe.

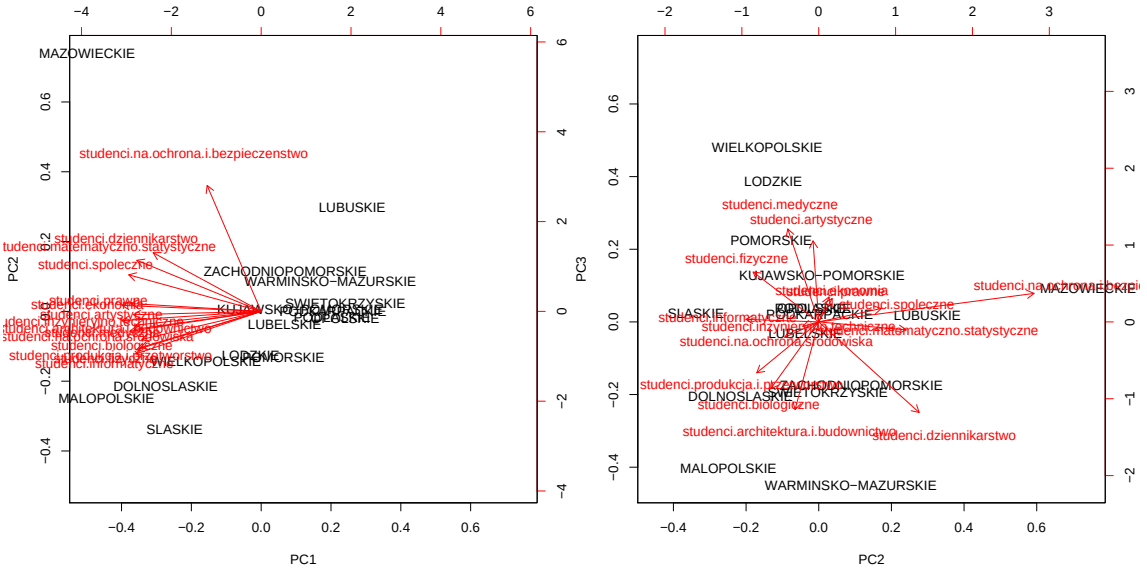
W przypadku pierwszej składowej współczynniki przy praktycznie każdej zmiennej wynoszą około -0.25. W przybliżeniu oznacza to, że pierwsza składowa będzie odpowiadała -łącznej liczbie studentów w danym województwie. A więc im więcej studentów tym mniejsza wartość pierwszej składowej.

Druga składowa jest już ciekawsza, ponieważ różnym województwom odpowiadają różne współczynniki. Województwa o dużych wartościach na drugiej składowej to województwa z „nadreprezentacją" studentów z kierunków społecznych, dziennikarstwa, matematyki i ochrony a „niedomiarze" studentów z nauk biologicznych, fizycznych, informatycznych i produkcji.

```
> # macierz przekształcenia
> wynik$rotation[,1:4]
                                       PC1      PC2      PC3      PC4
studenci.artystyczne                -0.2668 -0.01935  0.39012  0.06339
studenci.spoleczne                  -0.2769  0.21209  0.06445 -0.18107
studenci.ekonomia                   -0.2908  0.03508  0.12031  0.01692
studenci.prawne                     -0.2724  0.04551  0.11257  0.27527
studenci.dziennikarstwo             -0.2258  0.34018 -0.43605  0.37212
studenci.biologiczne                -0.2496 -0.16285 -0.32214  0.40296
studenci.fizyczne                   -0.2604 -0.22242  0.24023  0.28846
studenci.matematyczno.statystyczne  -0.2599  0.29657 -0.03841  0.13851
studenci.informatyczne              -0.2621 -0.24703  0.01225 -0.33688
studenci.medyczne                   -0.2654 -0.10490  0.44763  0.09636
studenci.inzynieryjno.techniczne    -0.2913 -0.05414 -0.02117 -0.08996
studenci.produkcja.i.przetworstwo   -0.2521 -0.20996 -0.24515 -0.39126
studenci.architektura.i.budownictwo -0.2621 -0.08025 -0.42188 -0.15234
studenci.na.ochrona.srodowiska      -0.2744 -0.11530 -0.07394 -0.29601
studenci.na.ochrona.i.bezpieczenstwo -0.1129  0.72998  0.13728 -0.29845
```



Rysunek 2.2: Graficzna reprezenacja wyników funkcji *prcomp()*.



Rysunek 2.3: Graficzna reprezenacja wyników PCA dla danych GUSowych.

2.2 Nieliniowe skalowanie wielowymiarowe (Sammon Mapping)

W przypadku metody PCA nowe współrzędne konstruowano tak, by były one kombinacjami liniowymi oryginalnych danych. To oczywiście nie jest jedyna możliwość

konstrukcji nowych zmiennych. Postawmy zagadnienie skalowania następująco.

Dane: mamy n obiektów oraz macierz odległości pomiędzy każdą parą obiektów. Oznaczmy przez d_{ij} odległość pomiędzy obiektem i-tym i j-tym.

Szukane: reprezentacja obiektów w przestrzeni k wymiarowej, tak by zminimalizować

$$stress = \frac{\sum_{i,j}(d_{ij} - \tilde{d_{ij}})^2/d_{ij}}{\sum_{i,j} d_{ij}},$$

gdzie $\tilde{d_{ij}}$ to odległość pomiędzy obiektami i i j w nowej k-wymiarowej przestrzeni.

Innymi słowy, szukamy (niekoniecznie liniowego) przekształcenia, które możliwie najwierniej (w sensie ważonego błędu kwadratowego) zachowa odległości pomiędzy obiektami. Takie przekształcenie poszukiwane jest iteracyjnie.

Jak to zrobić w R? Można np. używając funkcji `sammon(MASS)`. Pierwszym argumentem tej funkcji powinna być macierz odległości pomiędzy obiektami (np. wynik funkcji `dist()`) a argument k określa na iluwymiarową przestrzeń chcemy skalować dane (domyślnie $k = 2$).

```
> # wyznaczamy macierz odległości
> odleglosci = dist(dane)
> # wyznaczamy nowe współrzędne w przestrzeni dwuwymiarowej
> noweSammon = sammon(odleglosci, k=2, trace=FALSE)
> # jak wyglądają nowe współrzędne
> head(noweSammon$points)
                         [,1]       [,2]
DOLNOSLASKIE        15211.180 -3403.2348
KUJAWSKO-POMORSKIE  -7063.770 -3448.9048
LUBELSKIE           -4912.805 -1058.6175
LUBUSKIE           -12775.224 -3090.0001
LODZKIE              1251.418   916.5612
MALOPOLSKIE         21526.869 -1963.5498
```

Tabela 2.2: Pola obiektu będącego wynikiem funkcji *sammon()*.

`$points`	Macierz współrzędnych obiektów w nowej k-wymiarowej przestrzeni.
`$stress`	Uzyskana wartość optymalizowanego parametru stresu.

2.3 Skalowanie wielowymiarowe Kruskalla (MDS, ang. Multidimensional Scaling)

Jak już wspominaliśmy, metody redukcji wymiaru są często wykorzystywane do wizualizacji danych. W przypadku analizy składowych głównych po znalezieniu współrzędnych obiektów w nowej bazie wystarczy wziąć dwie pierwsze współrzędne by móc

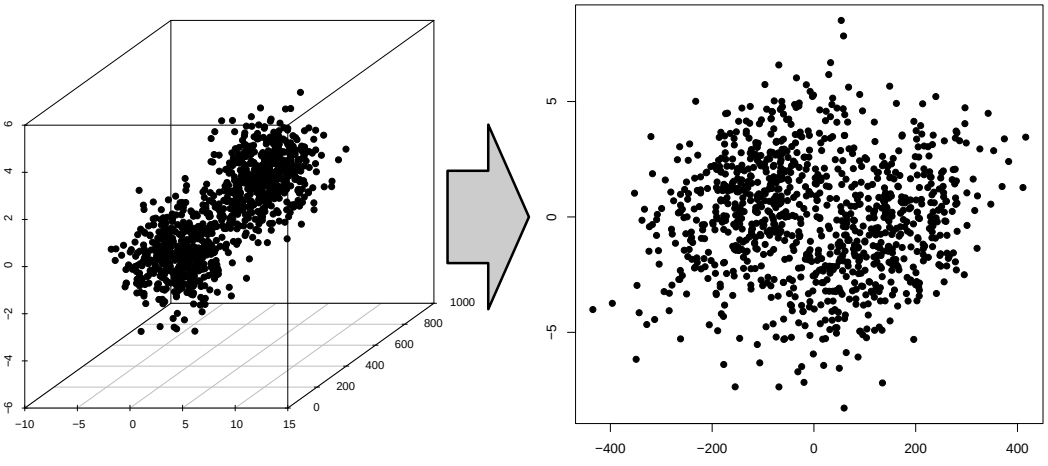
przedstawiać zbiór obserwacji na wykresie dwuwymiarowym, trzy pierwsze by móc przedstawić zbiór obserwacji na wykresie trójwymiarowym itp. Wadą analizy składowych głównych jest uwzględnianie wyłącznie zmiennych ilościowych. Kolejnym minusem jest konieczność posiadania wartości pomiarów dla kolejnych zmiennych, nie można tej metody użyć w sytuacji gdy mamy wyłącznie informacje o podobieństwie lub odległości pomiędzy obiektami.

Metody skalowania Sammona i Kruskala nie mają tych wad. Są to metody ekstrakcji cech, na podstawie macierzy odległości lub macierzy niepodobieństwa pomiędzy obiektami. Celem tych metod jest wyznaczenie współrzędnych w nowym układzie współrzędnych, w taki sposób by odległości pomiędzy obiektami w nowym układzie współrzędnych były podobne do oryginalnych odległości pomiędzy obiektami. Przykład skalowania wielowymiarowego przedstawiliśmy na rysunku 2.4.

W przypadku skalowania Kruskala minimalizowana jest wartość

$$stress = \frac{\sum_{i,j}(f(d_{ij}) - \tilde{d_{ij}})^2}{\sum_{i,j} f(d_{ij})^2},$$

gdzie $\tilde{d_{ij}}$ to odległość pomiędzy obiektami i i j w nowej k-wymiarowej przestrzeni a d_{ij} to oryginalne odległości pomiędzy obiektami przekształcone przez pewną monotoniczną funkcję $f()$ (więc d_{ij} i $\tilde{d}_{ij}$ mogą być w różnych skalach!).



Rysunek 2.4: Przykład skalowania wielowymiarowego.

Skalowanie wielowymiarowe jest w R dostępne w kilku funkcjach:

- funkcja `isoMDS(MASS)` wyznacza niemetryczne skalowanie Kruskala,
- funkcja `sammon(MASS)` wyznacza niemetryczne skalowanie Sammona (patrz poprzedni podrozdział [TODO: uspójnić!!]),
- funkcja `cmdscale(stats)` wyznacza skalowanie metryczne inaczej PCA (patrz poprzedni podrozdział [TODO: uspójnić!!]).

Poniżej przedstawimy przykłady użycia dla funkcji `isoMDS()`, z pozostałych korzysta się podobnie.

This has been discussed before in this list, and Ripley said „no, no!". I do it all the time, but only in secrecy.

Jari Oksanen (about replacing zero distances with tiny values for isoMDS()) fortune(163)

Najważniejszym argumentem wejściowym do algorytmu skalowania wielowymiarowego jest macierz odległości pomiędzy obserwacjami. Wyznaczyć ją można np. funkcją `dist(stats)`. W funkcji `dist()` zaimplementowane są wszystkie popularne metody liczenia odległości pomiędzy obserwacjami, w tym odległość euklidesowa (najpopularniejsza, odpowiadająca sumie kwadratów różnic poszczególnych współrzędnych, tej odległości odpowiada argument `method="euclidean"`), odległość maksimum (argument `method="maximum"`), odległość Manhattan, nazywana też odległością taksówkową lub miejską (suma modułów różnic pomiędzy współrzędnymi, argument `method="manhattan"`), odległość Mińkowskiego (argument `method="minkowski"`) oraz kilka innych mniej popularnych odległości. Jeżeli w zbiorze danych znajdują się zmienne jakościowe to funkcja `dist()` sobie z nimi nie poradzi. W takiej sytuacji lepiej wykorzystać funkcję `daisy(cluster)` wyznaczającą macierz niepodobieństwa pomiędzy obiektami. Funkcja `daisy()` uwzględnia również zmienne jakościowe (poniżej przedstawiamy przykład użycia). Macierz odległości jest obiektem klasy `dist()` i nie jest pamiętana jako macierz, a jedynie jako połowa macierzy (ponieważ odległość jest symetryczna szkoda pamięci na przechowywanie nadmiarowych danych). Jeżeli potrzebujemy przekształcić obiekt `dist()` na macierz to możemy wykorzystać funkcję `as.matrix()`.

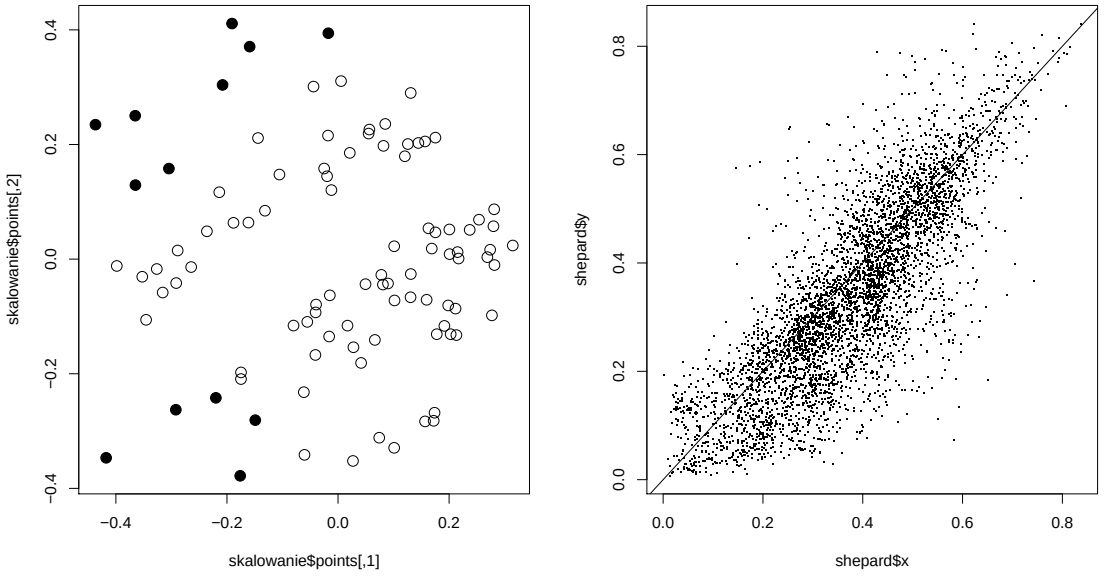
Wynikiem algorytmu skalowania wielowymiarowego są współrzędne obserwacji w pewnym nowym układzie współrzędnych. Możemy wybrać wymiar przestrzeni na jaką mają być przeskalowane dane (argument `k` funkcji `isoMDS`). Z pewnością po wykonaniu skalowania interesować nas będzie na ile skalowanie zachowało odległości pomiędzy obiektami, czy dużo jest znacznych zniekształceń. Do oceny wyników skalowania wykorzystać można wykres Sheparda przedstawiający na jednej osi oryginalne odległości pomiędzy obiektami a na drugiej osi odległości w nowym układzie współrzędnych. Do wyznaczenia obu wektorów odległości służy funkcja `Shepard(MASS)`, można też skorzystać z wrappera na tę funkcję, czyli z funkcji `stressplot(vegan)`.

Wrapper to funkcja agregująca i udostępniająca funkcjonalności innej funkcji.

Poniżej przedstawiamy przykład skalowania wielowymiarowego. Wykorzystamy tę metodę do przedstawienia za pomocą dwuwymiarowego wykresu podobieństw pomiędzy pacjentkami ze zbioru danych `daneO`. Graficzny wynik tych analiz jest przedstawiony na rysunku 2.5. Lewy rysunek przedstawia pacjentki w nowym dwuwymiarowym układzie współrzędnych, w tym przypadku pacjentki przedstawiane są jako punkty. Wypełniony punkt oznacza dla niepowodzenie leczenia a więc wznowę, a pusty w środku w oznacza wyleczenie pozytywne (widzimy, że pacjentki z niepowodzeniami grupują się blisko siebie). Ciekawym było by naniesienie na ten wykres nazwisk pacjentek i porównanie, które pacjentki pod względem zmierzonych wartości były do siebie podobne. Prawy rysunek przedstawia dokładność skalowania, a więc jak oryginalne odległości mają się do odległości w nowym układzie współrzędnych.

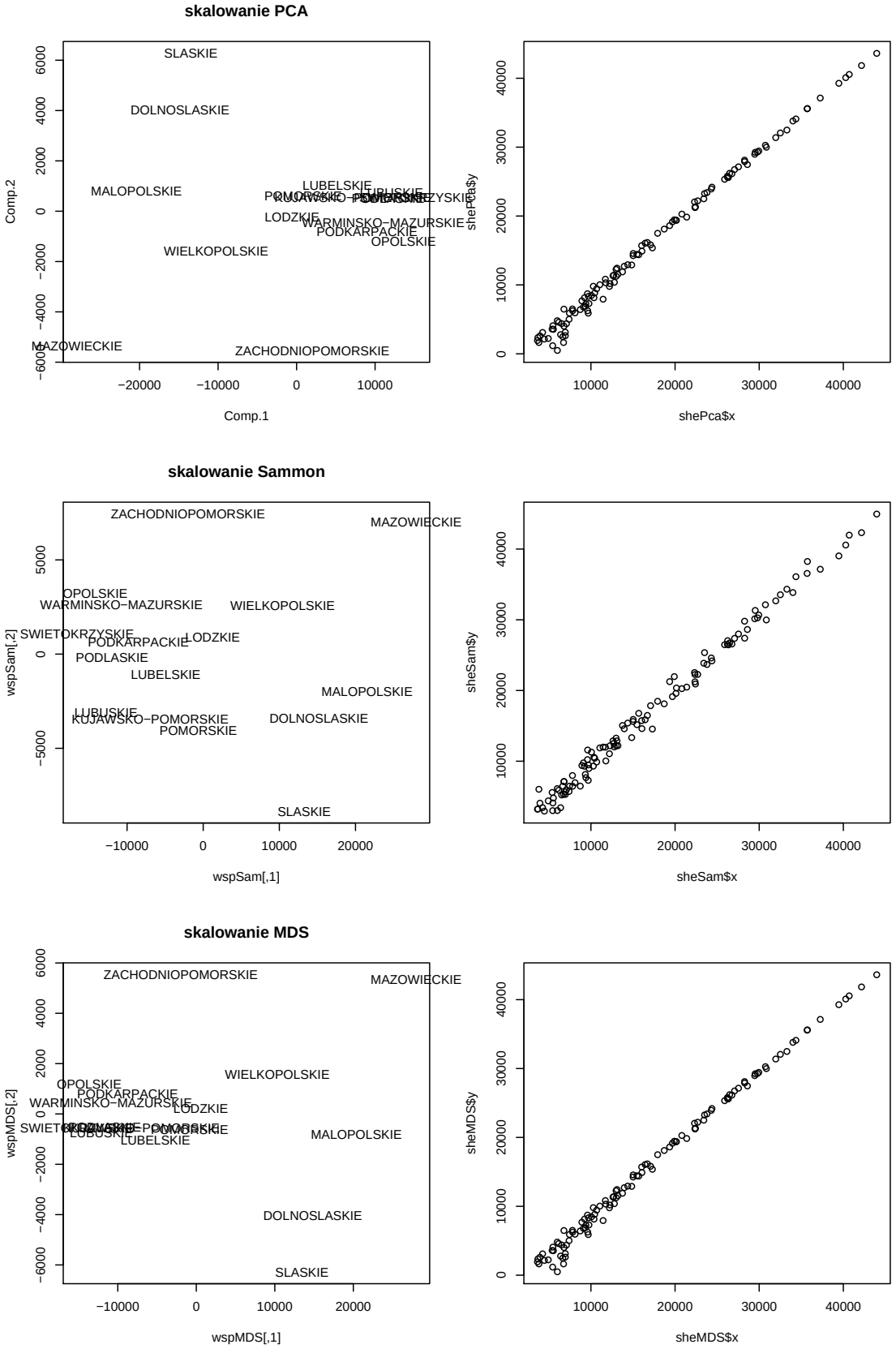
```
> # konstruujemy macierz niepodobieństwa pomiędzy pacjentkami, również
    zmienne jakościowe są uwzględnione
> niepodobienstwa = daisy(daneO)
Warning message:
In daisy(daneO) : binary variable(s) 2, 3 treated as interval scaled
> # przeprowadzamy skalowanie niemetryczne, skalujemy do przestrzeni o~
```

```
    dwóch wymiarach
> skalowanie = isoMDS(niepodobienstwa, k=2)
initial  value 30.138174
iter   5 value 25.846808
final  value 25.531983
converged
> # obiekt wynikowy zawiera współrzędne obserwacji w~nowym układzie
    współrzędnych
> str(skalowanie)
List of 2
 $ points: num [1:97, 1:2]  0.1011  0.3147 -0.1055  0.2811 -0.0604 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:97] "1" "2" "3" "4" ...
  .. ..$ : NULL
 $ stress: num 25.5
> # konstruujemy wektor pomocniczy do rysowania
> ksztalty = ifelse(daneO$Niepowodzenia=="brak", 21, 19)
> # rysujemy pacjentki w~nowym układzie współrzędnych
> plot(skalowanie$points, type = "p", pch=ksztalty, cex=1.5)
> # rysujemy również diagram Sheparda
> shepard <- Shepard(niepodobienstwa, skalowanie$points)
> plot(shepard, pch = ".")
> abline(0,1)
```



Rysunek 2.5: Graficzna reprezentacja wyników funkcji *isoMDS()* i *Shepard()*.

Zobaczmy jak wyglądają dane o województwach po przeskalowaniu różnymi metodami. Na wykresie 2.6 po lewej stronie przedstawiamy przeskalowane dane a po prawej wykresy Sheparda.



Rysunek 2.6: Graficzna reprezentacja wyników różnych funkcji skalowania, na przykładach danych GUS.

Rozdział 3

Analiza skupień

Analiza skupień to zbiór metod pozwalających na wyróżnienie zbiorów obserwacji (nazywanych skupieniami lub klastrami) podobnych do siebie. Proces szukania podziału na grupy, nazywany jest czasem klastrowaniem. W pakiecie R dostępnych jest bardzo wiele metod do przeprowadzania analizy skupień. Poniżej omówimy jedynie kilka wybranych funkcji z pakietów `cluster` i `stats`. Osoby zainteresowane tym tematem powinny przyjrzeć się również funkcjom z pakietów `flexclust` oraz `mclust02`.

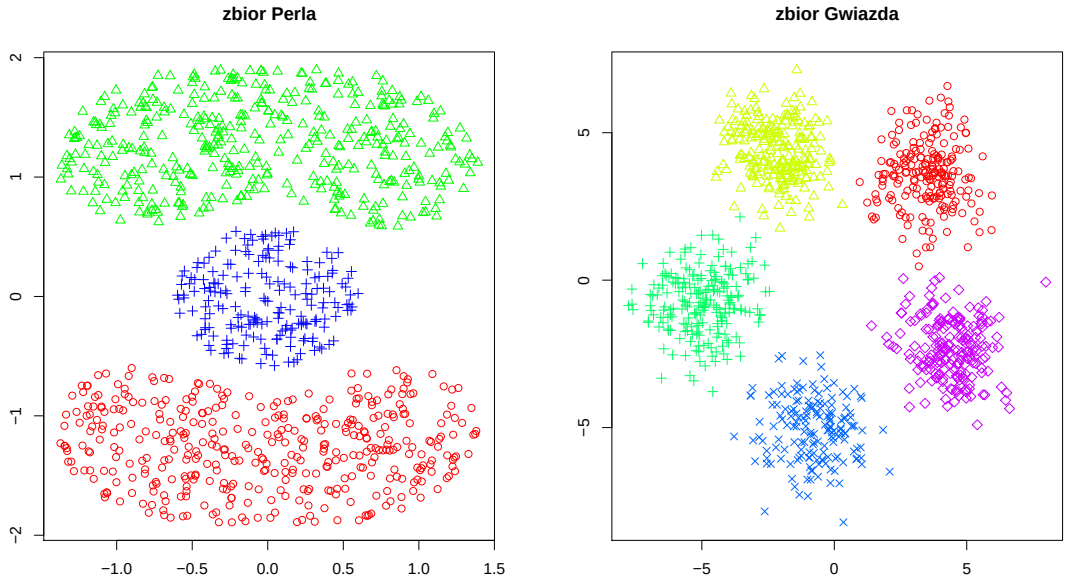
Wyniki działania wybranych procedur analizy skupień przedstawimy ma przykładzie zbioru danych benchmarkowych. W pakiecie `mlbench` (skrót od Machine Learning Benchmark Problems) umieszczonych jest wiele ciekawych zbiorów danych wykorzystywanych do testowania właściwości algorytmów dyskryminacji lub analizy skupień. W tym pakiecie znajdują się zbiory rzeczywistych danych, jak również funkcje do generowania zbiorów danych o określonych kształtach lub właściwościach. Dwa zbiory wygenerowanych danych na których będziemy testować metody analizy skupień przedstawione są na rysunku 3.1. Zostały one wygenerowane funkcjami `mlbench.cassini(mlbench)` i `mlbench.2dnormals(mlbench)`.

Do wyznaczania skupisk wystarczy macierz odległości pomiędzy obiektami. Domyślnie wyznaczane są odległości euklidesowe (może więc warto skalować dane?) ale można te odległości liczyć korzystając z innych funkcji `dist.BC(clusterSim)`, `dist.GDM(clusterSim)`, `dist.SM(clusterSim)`, `dist(stats)`, `dist.binary(ade4)`.

3.1 Metoda k-średnich

Celem tej metody jest podział zbioru danych na k klastrów. Dobry podział to taki, w którym suma odległości obserwacji należących do klastra jest znacznie mniejsza od sumie odległości obserwacji pomiędzy klastrami. Metoda k-średnich polega na wyznaczeniu współrzędnych k punktów, które zostaną uznane za środki klastrów. Obserwacja będzie należała do tego klastra, którego środek jest najbliżej niej.

Metoda k-średnich jest zaimplementowana w funkcji `kmeans(stats)`. Pierwszym argumentem tej funkcji jest ramka danych określająca wartości zmiennych dla kolejnych obserwacji. Drugim argumentem może być pojedyncza liczba określająca ile klastrów chcemy identyfikować (w tym przypadku środki klastrów będą wyznaczone iteracyjnym algorytmem) lub wektor środków klastrów. Algorytm wyboru środków



Rysunek 3.1: Dane, na których będziemy przedstawiać metody analizy skupień.

klastrów jest algorytmem zrandomizowanym, może też dawać różne wyniki nawet na tym samym zbiorze danych! Dlatego też zalecane jest uruchomienie kilkukrotne tego algorytmu oraz wybranie najlepsze go podziału na klastry. Można to zrobić też automatycznie, określając argument `nstart` funkcji `kmeans()`.

Algorytm k-średnich minimalizuje $tr(W)$ gdzie W to macierz kowariancji wewnątrz klas. Opisuje go poniższa sekwencja

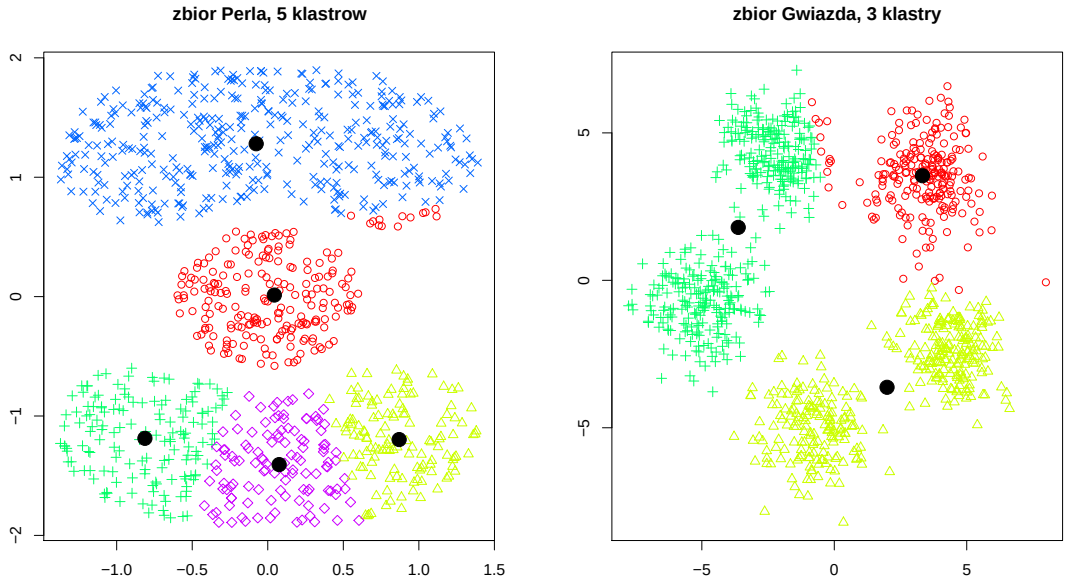
1. wybierany jest wstępny podział (w razie potrzeby można ten wybór powtórzyć wielokrotnie by znaleźć globalne maksimum),
2. przypisuje się obiekty do klasy z najbliższym środkiem ciężkości
3. przelicza się środki ciężkości dla nowych klas
4. kroki 2-3 powtarza się tak długo aż nie będą zachodziły żadne zmiany.

Poniżej prezentujemy przykład użycia funkcji `kmeans()`. Wyniki analizy skupień przedstawione są graficznie na rysunku 3.2. Różne klastry zaznaczono punktami o różnych kształtach. Czarne pełne punkty wskazują na środki znalezionych klastrów. Oczywiście właściwszym byłoby dopasowanie do lewego przykładu 3 klastrów, a do prawego 5 klastrów. Na przedstawionych przykładach możemy prześledzić ci się dzieje, jeżeli źle określimy liczbę klastrów (czytelnik powinien spróbować powtórzyć te analizy, wyniki najprawdopodobniej otrzyma inne!).

```
> # szukamy 5 klastrow, nie trafiło się nam najlepsze dopasowanie
> klaster = kmeans(zbiorPerla,5)
> # jak wygląda wynik w środku?
```

```
> # pole $cluster określa numer klastra dla kolejnych punktów, $centers
    określa współrzędne środków klastrów
> str(klaster)
List of 4
 $ cluster : int [1:1000] 3 3 3 3 3 3 3 3 3 3 ...
 $ centers : num [1:5, 1:2]  0.03203 -0.00749 -0.08380 -0.81601  0.91808
     ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:5] "1" "2" "3" "4" ...
  .. ..$ : NULL
 $ withinss: num [1:5]  22.4  10.9 126.9  11.5   9.8
 $ size    : int [1:5] 103 69 197 70 61
 - attr(*, "class")= chr "kmeans"
> # rysujemy punkty, różne klastry oznaczamy innymi kształtami punktów
> plot(zbiorPerla, pch=klaster$cluster)
> # dorysujmy środki klastrów
> points(klaster$centers, cex=2, pch=19)

> klaster = kmeans(zbiorGwiazda,2)
> plot(zbiorGwiazda, pch=klaster$cluster)
> points(klaster$centers, cex=2, pch=19)
```

Na powyższym przykładzie przedstawiliśmy pola w obiektach przekazanych przez funkcję `kmeans()`. Pole `$cluster` określa do jakiego klastra została przyporządkowana dana obserwacja, a pole `$centers` to współrzędne środków poszczególnych klastrów.



Rysunek 3.2: Graficzna prezentacja działania funkcji *kmeans()*.

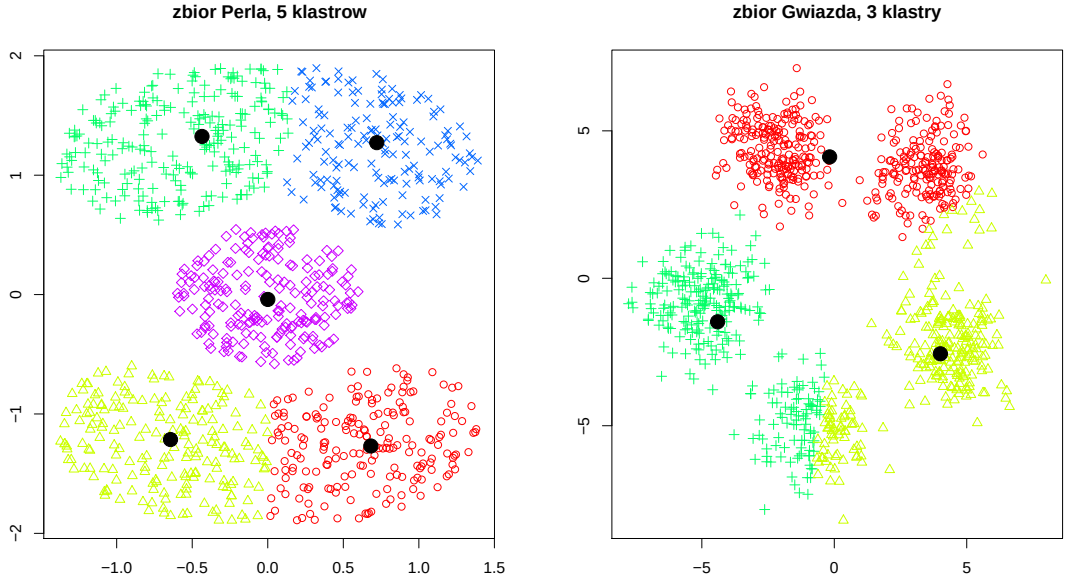
3.2 Metoda grupowania wokół centroidów (PAM, ang. Partitioning Around Medoids)

Metoda PAM działa na podobnej zasadzie jak k-średnich, z tą różnicą, że środkami klastrów są obserwacje ze zbioru danych (nazywane centroidami lub centrami klastrów). W metodzie PAM zbiór możliwych środków klastrów jest więc znacznie mniejszy, niż w metodzie k-średnich, zazwyczaj też wyniki działania metody PAM są stabilniejsze. Na rysunku 3.3 przedstawiony jest wynik działania poniższego przykładowego wywołania tej funkcji. Podobnie jak poprzednio różne klastry zaznaczono punktami o różnych kształtach. Czarne pełne punkty to środki klastrów (w tej metodzie odpowiadają przypadkom ze zbioru danych).

```
> # ponownie szukamy 5 klastrow
> klaster = pam(zbiorPerla,5)
> # jak wygląda wynik w środku?
> # pole $medoids określa współrzędne środków klastrów (wybranych
    przypadków), $in.med określa indeksy obserwacji, które są środkami
    klastrów, $clustering to wektor indeksów kolejnych klastrów, $silinfo
    to informacje o dopasowaniu danego obiektu do klastra w którym się
    znajduje (wartość silhouette)
> str(klaster)
List of 10
 $ medoids   : num [1:5, 1:2]  6.47e-01 -6.26e-01 -6.38e-01  5.87e-01
    1.59e-05 ...
 $ id.med    : int [1:5] 24 126 230 267 464
 $ clustering: int [1:1000] 1 1 2 1 1 2 1 1 1 2 ...
 $ objective : Named num [1:2] 0.526 0.461
  ..- attr(*, "names")= chr [1:2] "build" "swap"
 $ isolation : Factor w/ 3 levels "no","L","L*": 1 1 1 1 1
  ..- attr(*, "names")= chr [1:5] "1" "2" "3" "4" ...
 $ clusinfo  : num [1:5, 1:5]  87 113 101  99 100 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:5] "size" "max_diss" "av_diss" "diameter" ...
 $ silinfo   :List of 3
  ..$ widths         : num [1:1000, 1:3] 1 1 1 1 1 1 1 1 1 1 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:500] "12" "96" "133" "155" ...
  .. .. ..$ : chr [1:3] "cluster" "neighbor" "sil_width"
  ..$ clus.avg.widths: num [1:5] 0.434 0.440 0.482 0.443 0.508
  ..$ avg.width      : num 0.462
 $ diss      : NULL
 $ call      : language pam(x = zb1$x, k = 5)
 $ data      : num [1:1000, 1:2]  0.0964  0.6938 -0.5325  1.2839  0.1743
    ...
 - attr(*, "class")= chr [1:2] "pam" "partition"
> # rysujemy punkty, różne klastry oznaczamy innymi kształtami punktów
> plot(zbiorPerla, pch=klaster$clustering)
```

```
> # dorysujmy środki klastrów
> points(klaster$meoids, cex=2, pch=19)
```

Obiekt, będący wynikiem funkcji `pam()` ma sporo pól, najistotniejsze to `$medoids` ze współrzędnymi medoidów, `$id.med` z indeksami medoidów i `$clustering` z indeksami klastrów, do których zostały przypisane kolejne obserwacje.



Rysunek 3.3: Graficzna prezentacja działania funkcji *pam()*.

Metoda PAM jest obliczeniowo złożona. Może być niemożliwym obliczeniowo wykonanie jej na dużym zbiorze danych. Do klastrowania dużych zbiorów polecana jest metoda clara (Clustering Large Applications) zaimplementowana w funkcji `clara(cluster)`. Wykonuje ona wielokrotnie metodę PAM na mniejszych zbiorach danych i scala wyniki w jeden podział na klastry.

Algorytm PAM (k-medoidów) opisuje następująca sekwencja

1. wybiera się początkowy zbiór medoidów,
2. przepisuje się obiekty do klas o najbliższym medoidzie,
3. zmienia się medoidy o ile jest taka potrzeba, w takiej sytuacji wraca się do kroku 2.

Minimalizowana jest $\sum_{r=1}^{u} d(r)$, gdzie $d(r)$ to najmniejsza z sum odległości jednego punktu z klasy r do wszystkich pozostałych punktów z tej klasy.

3.3 Metoda aglomeracyjnego klastrowania hierarchicznego

Klastrowanie hierarchiczne różni się od przedstawionych powyżej metod tym, że zamiast dzielić obserwacje na określoną liczbę klastrów, określa się stopień podobieństwa poszczególnych obiektów i wyznacza się drzewo odpowiadające tym podobieństwom. Do budowy takich drzew wykorzystywane są różne algorytmy.

Algorytm AGglomerative NESting (AGNES) jest metodą aglomeracyjną, co oznacza, że w pierwszym kroku każda obserwacja traktowana jest jak osobny klaster. W kolejnych krokach klastry najbardziej podobne do siebie są łączone w coraz większe klastry, tak długo aż nie powstanie tylko jeden klaster. Algorytm aglomeracyjnego klastrowania hierarchicznego jest dostępny w funkcji `agnes(cluster)`. Pierwszym argumentem może być macierz danych (podobnie jak w przypadku innych algorytmów klastrowania) określająca współrzędne poszczególnych obserwacji lub też macierz odległości pomiędzy obserwacjami, a więc obiekt klasy `dist` (jak tworzyć takie obiekty pisaliśmy w poprzednim podrozdziale). Duże znaczenie ma metoda liczenia odległości pomiędzy obserwacjami. Z reguły zmiana metody liczenia odległości (np. z euklidesowej na taksówkową) prowadzi do otrzymania zupełnie innego wyniku.

Algorytm grupowania hierarchicznego można opisać następująco:

1. Każdą obserwacje traktujemy jako osobne skupienie.
2. Znajdujemy dwa skupiska najbliższe sobie. Odległość pomiędzy skupiskami można wyznaczać na różne sposoby. Trzy najpopularniejsze opisane są poniżej.
3. Łączymy dwa najbliższe skupiska w jedno.
4. Jeżeli pozostało więcej niż jedno skupisko to wracamy do kroku 2.

Kolejnym istotnym argumentem jest argument `method`. Określa on kolejność łączenia małych klastrów w coraz większe klastry. W każdym kroku łączone są najbliższe klastry, ale odległość pomiędzy dwoma klasterami można liczyć na trzy sposoby:

- `method="single"`, liczona jest odległość pomiędzy najbliższymi punktami każdego z klastrów, do jednego klastra dołączany jest klaster którego dowolny element jest najbliżej. Ta metoda odpowiada zachłannemu dodawania do skupiska obiektów bliskich brzegowi skupiska, możliwe jest tworzenie się tzw. łańcuchów kolejno dołączanych obiektów, być może już nie tak podobnych do całego skupiska, coś w stylu „przyjaciele naszych przyjaciół są naszymi przyjaciółmi",
- `method="average"`, liczona jest średnia odległość pomiędzy punktami każdego z klastrów, łączone są więc klastry średnio podobnych obserwacji, to jedna z popularniejszych metod ([unweighted pair-]group average method, UPGMA),
- `method="complete"`, liczona jest odległość pomiędzy najdalszymi punktami każdego z klastrów, jeżeli dwa skupiska są w odległości d oznacza to, że że każda para punktów w tych skupiskach jest nie bardziej odległa niż d.

- `method="ward"` skupiska o minimalnej wariancji, w wyniku otrzymuje się zwarte skupiska.
- `method="flexible"`, elastyczność tej metody polega na możliwości określenia jak liczona ma być odległość pomiędzy łączonymi klastrami. Tą odległość sparametryzowano czterema współczynnikami (więcej informacji znaleźć można w [1], p.237 lub w opisie funkcji `agnes()`). Korzystanie z tej opcji polecane jest bardziej doświadczonym użytkownikom, działa zasada: nie wiesz jak to działa nie używaj.
- `method="weighted"` odpowiada metodzie elastyczne z parametrem `par.method = 0.5`.

Dla funkcji `agnes()` domyślnie stosowana jest metoda `average`, a dla funkcji `hclust()` domyślnie stosowana jest `complete`. Funkcja `agnes()` w porównaniu do innych implementacji ma dwie dodatkowe cechy: wyznacza „agglomerative coefficient" i umożliwia rysowanie „banner.plot".

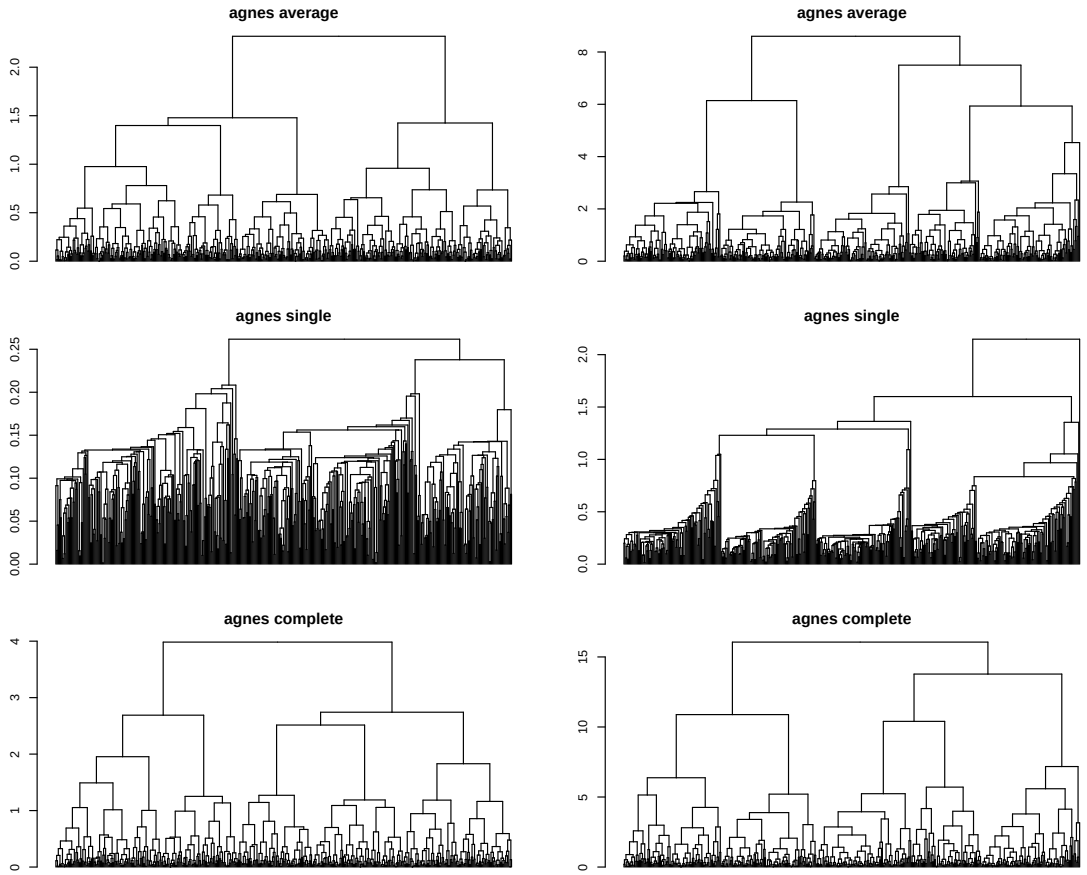
Użycie każdej z tych metod prowadzi do wygenerowania innego drzewa. Wyniki dla każdej z tych trzech wymienionych metod łączenia klastrów oraz dla obu zbiorów danych przedstawiamy na rysunku 3.4. Na tym rysunku przedstawione są wyniki analizy skupień dla 1000 obiektów, jeżeli analizujemy mniejszą liczbę obiektów, to na osi poziomej można odczytać nazwy poszczególnych obiektów a tym samym wizualizować, które obiekty są do siebie bardziej, a które mniej podobne (przykład takiego drzewa przedstawiliśmy na rysunku 3.5).

Wracając do rysunku 3.4 w przypadku zbioru Gwiazda sensowniejsze wyniki otrzymuje się dla metod łączenia `average` i `complete` (na drzewie można wydzielić 5 podgałęzi odpowiadającym spodziewanym skupiskom). Dla zbioru Perła najlepiej radzi sobie metoda łączenia `single` wyodrębniająca dosyć szybko trzy rozłączne skupiska. Najczęściej wykorzystywaną metodą łączenia jest `average`, nie oznacza to że zawsze daje najlepsze wyniki.

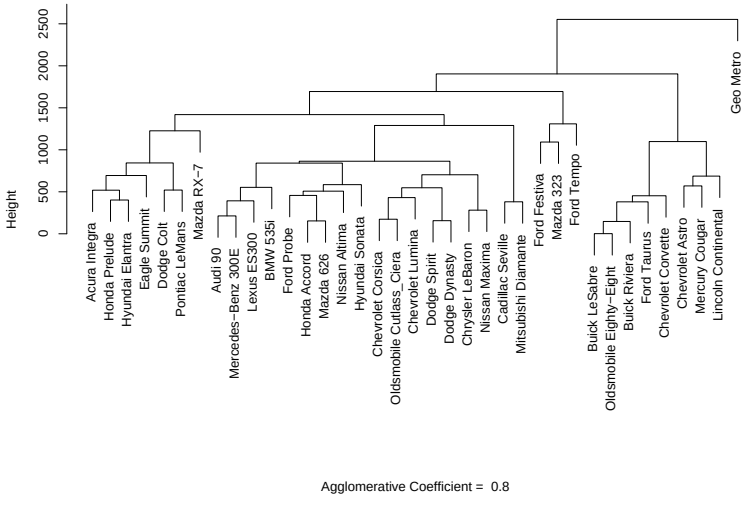
Aby na podstawie hierarchicznego klastrowania przypisać obserwacje do określonej liczby klastrów należy drzewo przyciąć na pewnej wysokości. Do przycinania drzewa służy funkcja `cutree(stats)`, jej pierwszym argumentem jest obiekt będący wynikiem metody hierarchicznej. Kolejnym argumentem, który należy wskazać jest `k` (określa do ilu klastrów chcemy przyciąć drzewo) lub `h` (określa na jakiej wysokości chcemy przyciąć drzewo). Wysokość drzewa na której chcemy odciąć klastry można odczytać z rysunków wygenerowanych dla tego drzewa.

Poniżej przedstawiamy przykład wywołania funkcji `agnes()` i `cuttree()`.

```
> # wywołanie funkcji AGNES
> klaster = agnes(zbiorPerla, method="average")
> # wynik możemy narysować przeciążoną funkcją plot
> plot(klaster)
> # otrzymane drzewo możęmy przyciąć do określonej liczby klastrów
> etykietkiKlastrow = cutree(klaster, k=2)
```



Rysunek 3.4: Graficzny przykład wyników funkcji *agnes()*.



Rysunek 3.5: Drzewo dla wybranych modeli samochodów na bazie zbioru danych *(*Cars93(MASS)).

3.4 Ile skupień wybrać?

Krótka odpowiedź na to pytanie brzmi 5. Długa wymaga odpowiedzi na dodatkowe pytanie, na czym nam zależy wybierając liczbę skupień? Co chcemy z tymi skupie-

niami robić i po co je wyznaczmy.

Popularną techniką wyboru liczby skupień jest rysowanie pewnej statystyki jako funkcji liczby skupień i wybór takiej liczby skupień dla której dana statystyka spełnia nasze oczekiwania (np. osiąga maksimum lub najszybciej maleje). Wiele przydatnych statystyk opisujących jakość podziału na grupy znajduje się w pakiecie `clusterSim`.

Popularne indeksy służące do wyboru liczby klastrów to:

- $tr(B)/tr(W)$ gdzie B to macierz kowariancji wewnątrzklasowej a W to macierz kowariancji miedzyklasowej,
- $B_u/(u-1)/W_u/(n-u)$ czyli miara zaproponowana przez Calińskiego i Harabasza (viva Poznań),
- sylwetka (silhouette) $S(u) = 1/n \sum_{i=1}^{n}(b(i) - a(i))/max(a(i), b(i))$ średnie podobieństwo obiektów do klastrów w których się znajdują, $a(i)$ to średnia odległość pomiędzy obiektem i a pozostałymi w tym samym klastrze, $b(i)$ to średnia odległość obiektu i od obiektów z najbliższego skupiska do i (do którego i nie należy).

Wiele indeksów wyznaczyć można korzystając z funkcji `index.G1(clusterSim)`, `index.G2(clusterSim)`, `index.G3(clusterSim)`, `index.S(clusterSim)`, `index.KL(clusterSim)`, `index.H(clusterSim)`, `index.Gap(clusterSim)`, `index.DB(clusterSim)`.

Funkcja `index.G1(clusterSim)` wyznacza indeks Calińskiego i Harabasza, funkcja `index.S(clusterSim)` wyznacza średnią sylwetkę (silhouette).

TODO: opisać funkcję `cluster.Description(clusterSim)` pozwalającą na opisywanie znalezionych klas.

3.5 Inne metody analizy skupień

Poza wymienionymi powyżej trzema metodami, w pakiecie R dostępnych jest wiele innych metod do analizy skupień. Poniżej wymienimy inne popularne.

- **Metoda hierarchiczej analizy skupień przez dzielenie.** Metoda hierarchicznego klastrowania przez łączenie działała w ten sposób, że zaczynając od małych, jednopunktowych klastrów w procesie łączenia klastrów otrzymywało się hierarchiczną zależność. Metoda analizy skupień przez dzielenie działa w przeciwnym kierunku. Zaczynając od jednego dużego klastra, w kolejnych krokach ten klaster jest dzielony na mniejsze klastry, aż do otrzymania jednoelementowych klastrów. Ta metoda jest zaimplementowana w funkcji `diana(cluster)`.

  Algorytm działania metody `diana()` opisany jest poniżej

  1. dla każdego skupiska wyznaczamy maksymalną odległość pomiędzy dwoma obserwacjami w skupisku,
  2. wybieramy skupisko o największej średnicy, w tym skupisku szukamy obiektu o największej średniej odległości od pozostałych, to będzie zalążek nowego skupiska,

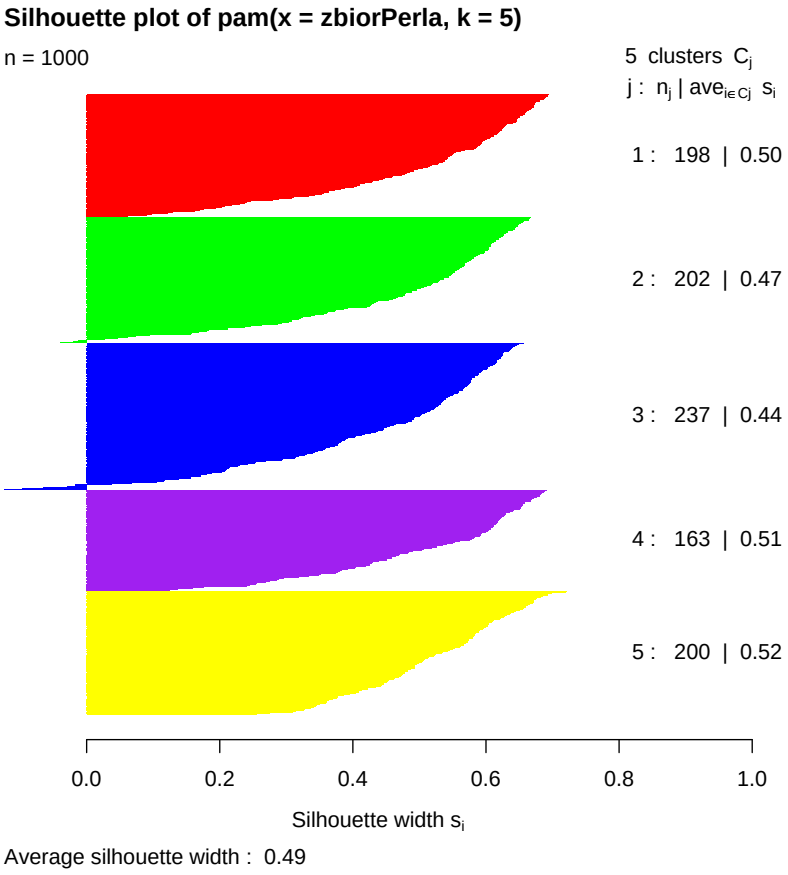
3. dla każdego obiektu sprawdzamy czy nie jest średnio bliżej obiektom z klasy nowej niż obiektom z klasy starej, w razie potrzeby zmieniana jest klasa
4. punkt 3 powtarzany jest tak długo aż nie będzie potrzeby zmieniać przynależności żadnego punktu

- **Metoda klastrowania rozmytego.** Jeżeli w procesie klastrowania dopuszczamy rozmytą przynależność do klastra (np. obserwacja może z pewnymi współczynnikami przynależeć do różnych klastrów) to uzasadnionym jest użycie metody klastrowania rozmytego. Ta metoda jest zaimplementowana w funkcji `fanny(cluster)`.
- **Inne metody hierarchiczego klastrowania.** Podobna w działaniu do `agnes()` metoda klastrowania hirarchicznego dostępna w funkcji `hclust(stats)`. Umożliwia ona większy wybór metody łączenia klastrów. Argumentem `method` należy wskazać jeden z wielu dostępnych indeksów do określania odległości pomiędzy klastrami, dostępne są indeksy znane z funkcji `agnes()` oraz `"mcquitty"`, `"ward"` i `"centroid"`.

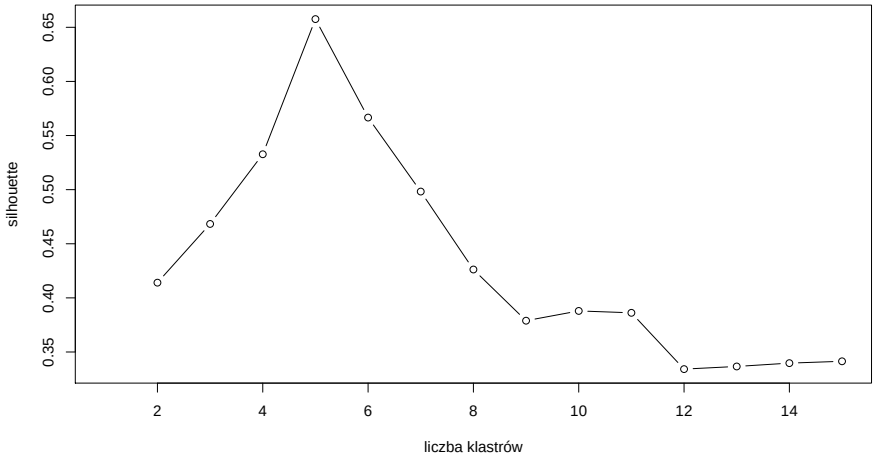
Wykonać klastrowanie jest stosunkowo prosto, jednak to jeszcze nie jest koniec pracy, potrzebne są metody oceny jakości podziału na skupiska. Do oceny jakości klastrowania można wykorzystać współczynnik silhouette, określający podobieństwo obiektu do innych obiektów w tym samym klastrze. Ten indeks jest wyznaczany przez funkcje `silhouette(cluster)`. Poniżej przedstawiamy przykład użycia tej funkcji, a na rysunku 3.6 przedstawiamy dopasowania poszczególnych punktów do klastrów. Wiele innych metod do oceny wyników klastrowania oraz badania zgodności dwóch podziałów na klastry jest dostępnych w pakiecie `clv`.

```
> kluster <- pam(zbiorPerla, 5)
> sil <- silhouette(kluster)
> summary(sil)
Silhouette of 1000 units in 5 clusters from pam(x = zbiorPerla, k = 5) :
Cluster sizes and average silhouette widths:
      198       202       237       163       200
0.5005513 0.4711742 0.4402300 0.5146160 0.5240964
Individual silhouette widths:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.1238  0.3895  0.5330  0.4873  0.6111  0.7220
> plot(sil, col = c("red", "green", "blue", "purple","yellow"))
```

Miara silhouette może użyta do wyznaczeniu liczby klastrów, na które należy podzielić dane. Na rysunku 3.7 przedstawiono zależność pomiędzy średnim współczynnikiem silhouette a liczbą klastrów wyznaczonych algorytmem PAN dla zbioru danych Gwiazda. W tym przypadku wyraźnie najlepszym wyborem jest 5 klastrów. Niestety w rzeczywistych problemach wybór klastrów nie jest tak prosty.



Rysunek 3.6: Wykres dopasowania punktów do poszczególnych klastrów z użyciem miary silhouette.



Rysunek 3.7: Wartości funkcji silhouette dla różnych liczb klastrów wyznaczonych algorytmem PAM na zbiorze danych Gwiazda.

3.6 Case study

```
#
# cztery skupiska, dla pewności inicjujemy 25 razy
#

dane   = daneGUS[,c(22:25)]
gus.k4 = kmeans(dane, 4, nstart=25)

silhouette(gus.k4$clust, dist(dane))[,3]


gus.p4 = pam(dane, 4)
plot(gus.p4)


#
# analiza skupień
#

par(mfrow=c(1,3))
h = hclust(dist(dane), method="average")
plot(h)

h = hclust(dist(dane), method="single")
plot(h)

h = hclust(dist(dane), method="complete")
plot(h)


#
# cztery przykładowe indeksy
#

mNiep       = daisy(dane)

clustGrid   = 2:14
indeksy = matrix(0,length(clustGrid),4)
indeksy[,1] = sapply(clustGrid, function(x) {index.G1(dane, pam(dane, x)$
    clustering)})
indeksy[,2] = sapply(clustGrid, function(x) {index.Gap(dane, cbind(pam(
    dane, x)$clustering, pam(dane, x+1)$clustering))$diffu})
indeksy[,3] = sapply(clustGrid, function(x) {index.G2(mNiep, pam(dane, x)$
    clustering)})
indeksy[,4] = sapply(clustGrid, function(x) {index.S(mNiep, pam(dane, x)$
    clustering)})

matplot(clustGrid, scale(indeksy), type="l",lwd=3)
legend("right",c("G1","Gap","G2","S"),lwd=3,bg="white",col=1:4)
```

```
#
# cztery metody porównane indeksem G1
#

clustGrid = 2:10
indeksy = matrix(0,length(clustGrid),5)
indeksy[,1] = sapply(clustGrid, function(x) {index.G1(dane, pam(dane, x)$
    clustering)})
indeksy[,2] = sapply(clustGrid, function(x) {index.G1(dane, kmeans(dane, x
    )$cluster)})
indeksy[,3] = sapply(clustGrid, function(x) {index.G1(dane, cutree(hclust(
    dist(dane),"average"), k = x))})
indeksy[,4] = sapply(clustGrid, function(x) {index.G1(dane, cutree(hclust(
    dist(dane),"single"), k = x))})
indeksy[,5] = sapply(clustGrid, function(x) {index.G1(dane, cutree(hclust(
    dist(dane),"complete"), k = x))})

matplot(clustGrid, indeksy, type="l",lwd=3)
legend("right",c("PAM","kmeans","hclust av","hclust si","hclust co"),lwd
    =3,bg="white",col=1:5)


#
# profile skupień
#

gus.p4 = pam(dane,4)$clustering

desc = cluster.Description(dane, gus.p4)
dimnames(desc)[[2]] = colnames(dane)
dimnames(desc)[[3]] = c("srednia", "sd", "median", "mad", "mode")

# wartosci srednie
desc[,,1]
```

Rozdział 4

Analiza dyskryminacji

W wielu dziedzinach potrzebne są metody, potrafiące automatycznie przypisać nowy obiekt do jednej z wyróżnionych klas. W medycynie interesować nas może czy pacjent jest chory, a jeżeli tak to na co (zbiorem klas do którego chcemy przypisać mogą być możliwe choroby, lub tylko informacja czy jest chory czy nie). W analizie kredytowej dla firm chcemy przewidzieć czy firma spłaci kredyt czy nie. W analizie obrazów z fotoradarów policyjnych będzie nas interesowało określenie numeru rejestracji samochodu który przekroczył prędkość a również typu pojazdu (w końcu ograniczenia dla ciężarówek są inne niż dla samochodów).

W rozdziale **??** używaliśmy regresji logistycznej do znalezienia parametrów, które można by wykorzystać w określeniu ryzyka pojawienia się wznowienia choroby u operowanych pacjentek. Okazuje się wiec, że regresja logistyczna jest klasyfikatorem, pozwalającym na przypisanie nowego obiektu do jednej z dwóch klas. Poniżej przedstawimy szereg funkcji implementujących inne popularne metody analizy dyskryminacji.

Celem procesu dyskryminacji (nazywanego też klasyfikacją, uczeniem z nauczycielem lub uczeniem z nadzorem) jest zbudowanie reguły, potrafiącej przypisywać możliwie dokładnie nowe obiekty do znanych klas. W przypadku większości metod możliwe jest klasyfikowanie do więcej niż dwie klasy.

4.1 Dyskryminacja liniowa i kwadratowa

Dyskryminacja liniowa, a więc metoda wyznaczania (hiper)płaszczyzn separujących obiekty różnych klas, jest dostępna w funkcji `lda(MASS)`. Rozszerzeniem tej metody jest dyskryminacja kwadratowa, umożliwiająca dyskryminacje powierzchniami, w opisie których mogą pojawić się człony stopnia drugiego. Metoda klasyfikacji kwadratowej jest dostępna w funkcji `qda(MASS)`. Wynikami obu funkcji jest klasyfikator, wyznaczony na zbiorze uczącym. Aby użyć go do predykcji klas dla nowych obiektów możemy wykorzystać przeciążoną funkcje `predict()`.

Poniżej przedstawiamy przykład użycia funkcji `lda()`. Z funkcji `qda()` korzysta się w identyczny sposób. Na potrzeby przykładu wykorzystaliśmy zbiór danych dotyczących występowania cukrzycy u Indian Pima, ten zbiór danych jest dostępny w zbiorze `PimaIndiansDiabetes2(mlbench)`. Interesować nas będą dwie zmienne z tego zbioru danych, opisujące poziom glukozy i insuliny, w zbiorze danych jest

znacznie więcej zmiennych, ale na potrzeby wizualizacji wybraliśmy tę parę. Na bazie tych dwóch zmiennych będziemy badać skuteczność oceny czy dana osoba jest cukrzykiem z wykorzystaniem obu algorytmów klasyfikacji. Zbiór danych podzielimy na dwie części, uczącą i testową. Klasyfikator zostanie „nauczony" na zbiorze uczących, a później będziemy weryfikować jego właściwości na zbiorze testowym.

Pierwszym argumentem funkcji `lda()` jest zbiór zmiennych na bazie których budowany będzie klasyfikator (ramka danych lub macierz). Drugim argumentem `grouping` jest wektor określający klasy kolejnych obiektów. Kolejnym wykorzystanym poniżej argumentem jest `subset`, określający indeksy obiektów, na bazie których budowany ma być klasyfikator. Jeżeli argument `subset` nie będzie podany, to do konstrukcji klasyfikatora wykorzystane będą wszystkie obiekty. Jako pierwszy argument funkcji `lda()` można również podać również formułę, określającą, które zmienne mają być użyte do konstrukcji klasyfikatora a która zmienna opisuje klasy.

```
> # wczytujemy zbiór danych z pakietu mlbench, usuwamy brakujące dane i
    logarytmujemy poziom insuliny
> data(PimaIndiansDiabetes2)
> dane = na.omit(PimaIndiansDiabetes2)[,c(2,5,9)]
> dane[,2] = log(dane[,2])
> # zbiór danych chcemy podzielić na dwie części, uczącą i~testową,
    funkcją sample wylosujemy indeksy obiektów, które trafia do zbioru
    uczącego
> zbior.uczacy    = sample(1:nrow(dane), nrow(dane)/2, F)
> # wywołujemy funkcję lda
> klasyfikatorLDA = lda(dane[,1:2], grouping = dane[,3], subset=zbior.
    uczacy)
> # jak wygląda wynik w~środku?
> str(klasyfikatorLDA)
List of 8
 $ prior  : Named num [1:2] 0.643 0.357
  ..- attr(*, "names")= chr [1:2] "neg" "pos"
 $ counts : Named int [1:2] 126 70
  ..- attr(*, "names")= chr [1:2] "neg" "pos"
 $ means  : num [1:2, 1:2] 110.42 146.50   4.57   5.20
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "neg" "pos"
  .. ..$ : chr [1:2] "glucose" "insulin"
 $ scaling: num [1:2, 1] 0.0316 0.3227
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "glucose" "insulin"
  .. ..$ : chr "LD1"
 $ lev    : chr [1:2] "neg" "pos"
 $ svd    : num 9
 $ N      : int 196
 $ call   : language lda(x = dane[, 1:2], grouping = dane[, 3], subset =
    zbior.uczacy)
 - attr(*, "class")= chr "lda"
```

Zbudowanie klasyfikatora to dopiero pierwszy krok, kolejnym jest jego ocena. Na bazie obserwacji niewykorzystanych do budowania klasyfikatora zbadamy jaka była zgodność klasyfikatora z rzeczywistymi danymi (ponieważ zbiór uczący wybraliśmy losowo, to dla różnych powtórzeń otrzymalibyśmy inny błąd klasyfikacji). Do klasyfikacji nowych obiektów użyjemy funkcji `predict()`. Jest to funkcja przeciążona, działająca dla większości klasyfikatorów. Wynikiem tej funkcji mogą być prognozowane klasy dla nowych obserwacji, lub też prawdopodobieństwa a posteriori przynależności do danej klasy.

```
> # używając metody predict wykonujemy klasyfikacje obiektów ze zbioru
    testowego
> oceny = predict(klasyfikatorLDA, newdata=dane[-zbior.uczacy,1:2])
> # jak wyglądają wyniki? Pole $class wskazuje na przewidzianą klasę, pole
     $posterior określa wyznaczone prawdopodobieństwo przynależności do
    każdej z klas, na podstawie tej wartości obiekt był przypisywany do
    bardziej prawdopodobnej dla niego klasy
> str(oceny)
List of 3
 $ class    : Factor w/ 2 levels "neg","pos": 1 1 2 1 1 1 1 1 2 1 ...
 $ posterior: num [1:196, 1:2] 0.5302 0.9480 0.0506 0.8663 0.7852 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:196] "5" "7" "9" "19" ...
  .. ..$ : chr [1:2] "neg" "pos"
 $ x        : num [1:196, 1]  0.540 -1.533  2.814 -0.762 -0.336 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:196] "5" "7" "9" "19" ...
  .. ..$ : chr "LD1"

> # porównajmy macierzą kontyngencji oceny i rzeczywiste etykietki dla
    kolejnych obiektów
> table(predykcja = oceny$class, prawdziwe = dane[-zbior.uczacy,3])
         prawdziwe
predykcja neg pos
      neg 115  29
      pos  21  31
```

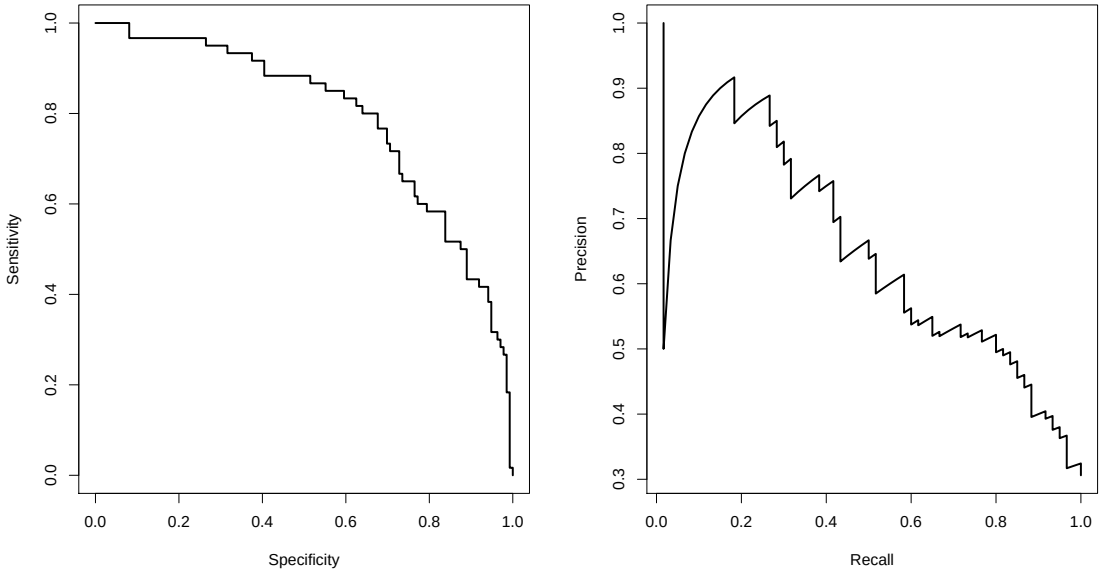
W powyższym przykładzie w ostatnim poleceniu wyznaczyliśmy tablice kontyngencji dla wyników. Na bazie tak otrzymanej tablicy kontyngencji można wyznaczyć błąd predykcji. Jest wiele wskaźników opisujących błąd predykcji, począwszy od popularnych: czułość (ang. sensitivity `TP/(TP+FN)` opisuje jaki procent chorych zostanie poprawnie zdiagnozowanych jako chorzy), specyficzność (ang. (Specifity) `TN/(TN+FP)` określa jaki procent zdrowych zostanie poprawnie zdiagnozowanych jako zdrowi), przez mniej popularne aż po takie o których mało kto słyszał (np. mutual information, współczynnik ϕ itp.). Bogata lista takich współczynników wymieniona jest w opisie funkcji `performance(ROCR)` (definiując błąd klasyfikacji używa się oznaczeń `TP`, `TN`, `FP`, `FN` określających kolejno liczbę poprawnie wykrytych sygnałów pozytywnych, poprawnie wykrytych braków sygnału, fałszywie wykrytych sygnałów pozytywnych oraz fałszywie wykrytych braków sygnałów. W powyższym przypadku

czułość wyniosła $31/(31+29) \approx 0.517$ a specyficzność $115/(115+21) \approx 0.846$.

Jeżeli już jesteśmy przy pakiecie `ROCR` to na przykładzie przedstawimy w jaki sposób wyznaczać krzywe ROC dla klasyfikatorów. Proszę zauważyć, że funkcja `predict()` poza ocenionymi klasami jako wynik przekazuje również prawdopodobieństwo przynależności do jednej z klas (pole `posterior` wyniku funkcji `predict()`). Krzywa ROC to zbiór punktów wyznaczonych dla różnych poziomów odcięcia (ang. threshold) dla wspomnianego prawdopodobieństwa przynależności do jednej z klas. Współrzędne każdego punktu to czułość i specyficzność (dokładniej rzecz biorąc 1-specyficzność) otrzymana dla zadanego punktu odcięcia.

Poniżej przykład użycia funkcji z pakietu `ROCR`, służącej do rysowania krzywych ROC i innych o podobnych właściwościach. Wynik graficzny przedstawiony jest na rysunku 4.1. Na osiach tego wykresu mogą być przedstawiane różne miary dokładności klasyfikacji, w zależności od argumentów funkcji `performance()`.

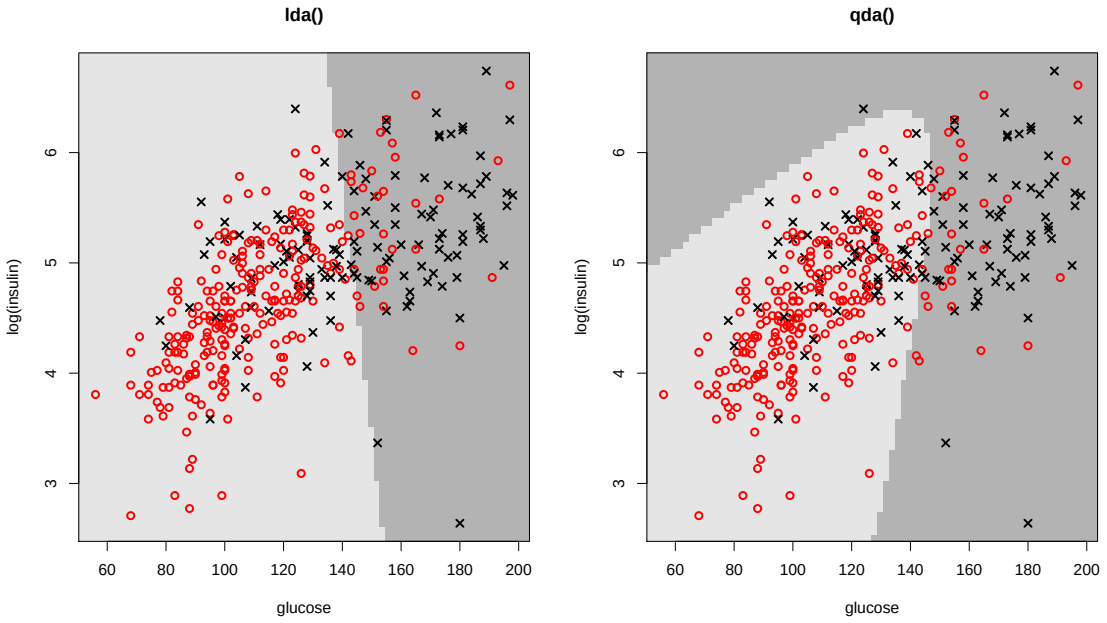
```
# wyznaczamy obiekt klasy prediction, zawierający informacje o prawdziwych
    klasach, i prawdopodobieństwu przynależności do wybranej klasy
pred <- prediction(oceny$posterior[,2], dane[-zbior.uczacy,3])
# wyznaczamy i wyrysowujemy wybrane miary dobroci klasyfikacji
perf <- performance(pred, "sens", "spec")
plot(perf)
perf <- performance(pred, "prec", "rec")
plot(perf)
```



Rysunek 4.1: Wykres zależności czułości od specyficzności oraz miary prediction od recall dla klasyfikacji z użyciem funkcji *lda()*.

Na rysunku 4.2 przedstawiamy kształty obszarów decyzyjnych dla obu klasyfikatorów (do wyznaczania obszarów decyzyjnych można się posłużyć funkcją `partimat(klaR)`

lub `drawparti(klaR)`). Obszary decyzyjne wyznaczane są dla wytrenowanego klasyfikatora, przedstawiają do której klasy zostałby przypisany punkt o określonych współrzędnych. Osoby chore na cukrzyce oznaczane są czarnymi krzyżykami, osoby zdrowe czerwonymi okręgami. Punkty w których przewidzianą klasą była by cukrzyca zaznaczone są ciemnoszarym kolorem a punkty dla których klasyfikowalibyśmy do grona osób zdrowych zaznaczono jaśniejszym szarym kolorem.



Rysunek 4.2: Przykładowe obszary decyzyjne dla liniowej i kwadratowej dyskryminacji dostępne w funkcjach *lda()* o *qda()*.

4.2 Metoda najbliższych sąsiadów

Bardzo popularną metodą klasyfikacji jest metoda k-sąsiadów. Idea jej działania jest prosta i intuicyjna. Nowemu obiektowi przypisuje się klasę, która występuje najczęściej wśród jego k sąsiadów (k najbliższych obiektów znajdujących się w zbiorze uczącym, najbliższych w sensie określonej miary odległości). Ten klasyfikator dostępny jest w różnych funkcjach, począwszy od `knn(class)` (zwykły klasyfikator najbliższych sąsiadów), przez `kknn(kknn)` (ważony klasyfikator k-sąsiadów) oraz `knncat(knncat)` (klasyfikator k-sąsiadów, również dla zmiennych jakościowych).

Poniżej przedstawimy implementację metody k-sąsiadów z funkcji `ipredknn(ipred)`. Na rysunku 4.3 prezentujemy obszary decyzyjne wyznaczone dla różnej liczby sąsiadów (odpowiednio k=3 i k=21) w omawianym zagadnieniu klasyfikacji na osoby zdrowe i chore na cukrzycę. Ponieważ metoda k-sąsiadów bazuje silnie na odległościach pomiędzy obiektami (jakoś trzeba mierzyć odległość od sąsiadów), przed rozpoczęciem obliczeń wykonamy skalowanie danych.

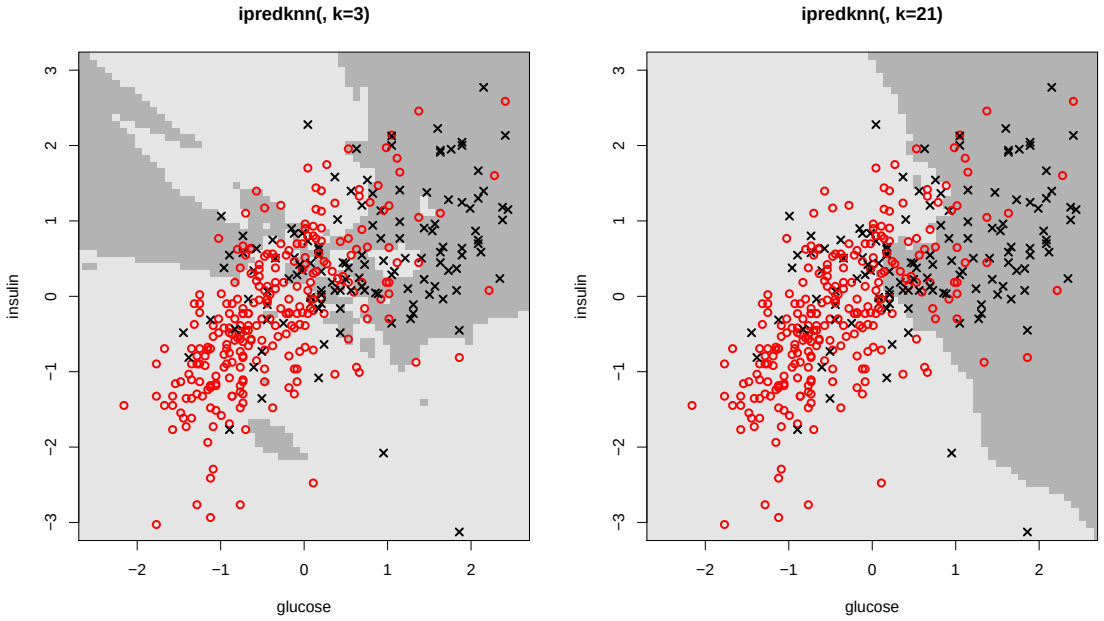
```
> # zaczynamy od przeskalownia danych
```

```
> dane[,1:2] = scale(dane[,1:2])
> # budujemy klasyfikator k-sąsiadów, dla 3 sąsiadów
> klasyfikatorKNN = ipredknn(diabetes~glucose+insulin, data = dane, subset
   =zbior.uczacy, k=3)
> # wykonujemy predykcję klas i wyświetlamy macierz kontyngencji
> oceny = predict(klasyfikatorKNN, dane[-zbior.uczacy, ], "class")
> table(predykcja = oceny, prawdziwe = dane[-zbior.uczacy,3])
         prawdziwe
predykcja neg pos
      neg  98  25
      pos  38  35
```

Błąd klasyfikacji można liczyć na palcach (powyższą procedurę należało by uśrednić po kilku wstępnych podziałach na zbiór uczący i testowy). Można też błąd klasyfikacji wyznaczyć wykorzystać funkcję `errorest(ipred)`. Wylicza ona błąd klasyfikacji (określony jako procent źle zaklasyfikowanych obiektów) używając różnych estymatorów tego błędu, w tym opartego na walidacji skrośnej (ocenie krzyżowej, ang. cross validation, domyślnie z podziałem na 10 grup), metodzie bootstrap lub estymatorze 632+. Estymator błędu możemy wybrać określając argument `estimator`. W funkcji `errorest()` jako kolejne argumenty należy wskazać zbiór danych, metodę budowy klasyfikatora (argument `model`) oraz metodę wyznaczania ocen dla zbioru testowego (argument `predict`). Funkcja `errorest()` pozwala na jednolity sposób wyznaczenia błędu dla dowolnej metody klasyfikacji. Przedstawimy poniżej przykład dla metody najbliższych sąsiadów.

```
> # wyznaczmy błąd klasyfikacji dla metody 3 sąsiadów
> errorest(diabetes~glucose+insulin, data = dane, model=ipredknn, k=3,
+          estimator = "632plus", predict= function(ob, newdata)  predict(
    ob, newdata, "class"))

Call:
errorest.data.frame(formula = diabetes ~ glucose + insulin, data = dane,
    model = ipredknn, predict = function(ob, newdata)  predict(ob, newdata
        , "class")
    , estimator = "632plus", k = 3)

         .632+ Bootstrap estimator of misclassification error
         with 25 bootstrap replications

Misclassification error:  0.2874

>
> # wyznaczmy błąd klasyfikacji dla metody 21 sąsiadów, powinna być
    stabilniejsza
> blad = errorest(diabetes~glucose+insulin, data = dane, model=ipredknn, k
   =21,
+          estimator = "632plus", predict= function(ob, newdata)  predict(
    ob, newdata, "class"))
```

```
> blad$error
[1] 0.2599233
```



Rysunek 4.3: Przykładowe obszary decyzyjne dla metody k-sąsiadów z parametrami k=3 i k=21.

4.3 Naiwny klasyfikator Bayesowski

Do klasyfikacji wykorzystać można szeroką grupę metod bazujących na ocenie prawdopodobieństwa przynależności do określonej grupy. Dla każdej z klas ocenia się częstość (w przypadku ciągłym gęstość) występowania obiektów o określonych parametrach. Następnie dla nowego obiektu wyznacza się częstości występowania obiektów poszczególnych klas i wybiera się klasę występującą dla tych parametrów najczęściej.

Do tej grupy metod należy naiwny klasyfikator Bayesowski. Bayesowski, ponieważ bazuje na regule Bayesa użytej do wyznaczenia prawdopodobieństwa a posteriori należenia do poszczególnych klas. Naiwność w tym kontekście oznacza przyjęte założenie, że łączna gęstość występowania obiektów jest iloczynem gęstości brzegowych. Naiwny klasyfikator Bayesowski jest dostępny w funkcjach `naiveBayes(e1071)` i `NaiveBayes(klaR)`. Poniżej przedstawimy tą drugą implementację.

Sposób użycia tej funkcji jest podobny do użycia innych opisanych powyżej klasyfikatorów. Na rysunku 4.4 przestawione są warunkowe brzegowe oceny gęstości dla obu zmiennych dla każdej z klas, na bazie tych gęstości wykonywana jest kalsyfikacja. Na rysunku 4.5 przedstawiamy przykładowe obszary decyzyjne dla naiwnego klasyfikatora Bayesowskiego.

```
> # konstruujemy naiwny klasyfikator Bayesowski
```

```
> mN    <- NaiveBayes(diabetes~glucose+insulin, data=dane, subset=zbior.
   uczacy)
> # wyznaczamy oceny
> oceny <- predict(mN, dane[-zbior.uczacy,])$class
> table(predykcja = oceny, prawdziwe = dane[-zbior.uczacy,3])
         prawdziwe
predykcja neg pos
      neg 113  27
      pos  23  33
> plot(mN)
```

4.4 Drzewa decyzyjne

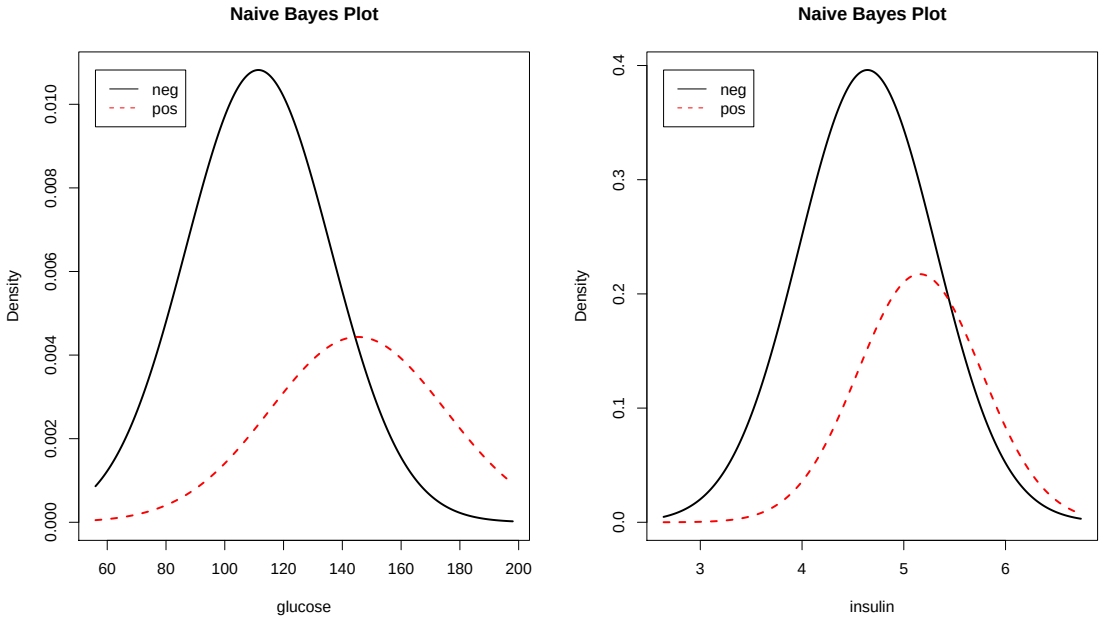
Inną klasą klasyfikatorów są cieszące się dużą popularnością metody bazujące na drzewach decyzyjnych (klasyfikacyjnych). W tym przypadku klasyfikator jest reprezentowany przez drzewo binarne, w którego węzłach znajdują się pytania o wartości określonej cechy, a w liściach znajdują się oceny klas. Przykład drzewa klasyfikacyjnego przedstawiamy na rysunku 4.6. Dodatkowo w liściach przedstawiono proporcję obiektów z obu klas, które znalazły się w danym węźle, a wiec spełniły warunki określone w poszczególnych węzłach drzewa. Jeżeli nowe obiekty będziemy przypisywać do klasy, która w danym liściu występowała najczęściej, to dla trzech liści (czwartego, szóstego i siódmego) klasyfikować będziemy do grupy osób chorych, a w pozostałych liściach będziemy klasyfikować do grupy osób zdrowych.

Dlaczego nas to nie dziwi?

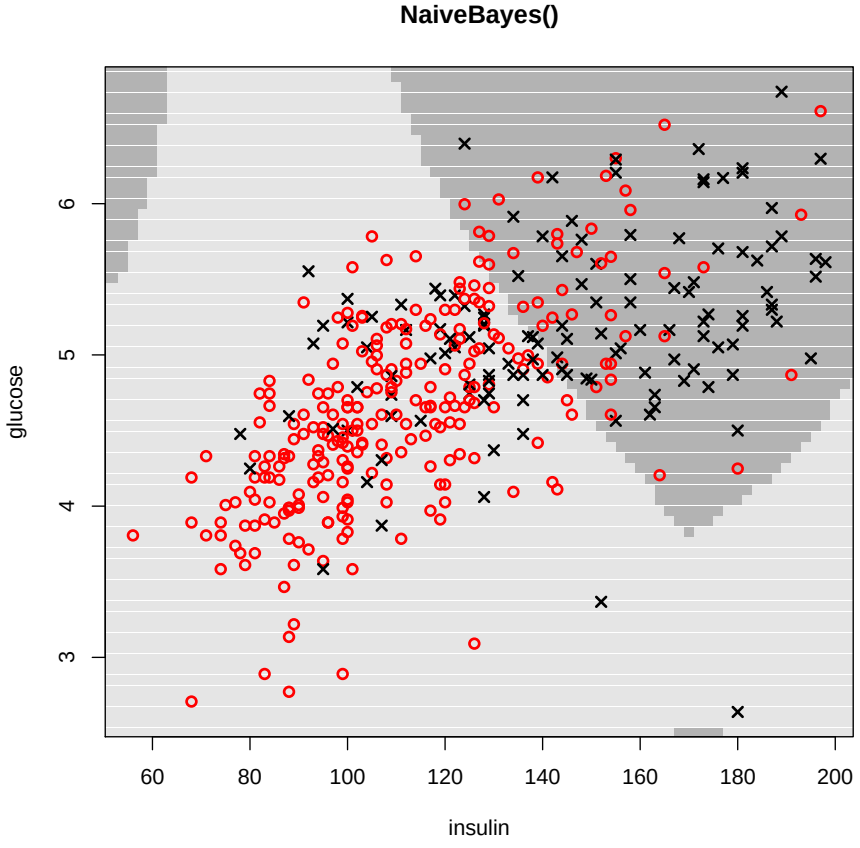
W pakiecie R metoda wyznaczania drzew decyzyjnych dostępna jest w wielu różnych funkcjach. Popularnie wykorzystywane są funkcje `tree(tree)`, `rpart(rpart)` oraz `cpart(party)`. Poniżej przedstawimy tylko tą ostatnią, ponieważ są dla niej opracowane najbardziej atrakcyjne funkcje do prezentacji graficznej. Z wszystkich wymienionych funkcji do konstrukcji drzew korzysta się podobnie. Wymienione funkcje mogą służyć zarówno do wyznaczania drzew regresyjnych jak i klasyfikacyjnych, ale w tym miejscu przedstawimy tylko ich klasyfikacyjną naturę. Do wizualizacji drzew klasyfikacyjnych można wykorzystać (w zależności od tego jaką funkcją wyznaczyliśmy drzewo) funkcje `plot.BinaryTree(party)`, `plot.tree(tree)`, `text.tree(tree)`, `draw.tree(maptree)`, `plot.rpart(rpart)` oraz `text.rpart(rpart)`.

Aby zbudować drzewo klasyfikacyjne należy określić kryterium podziału, a więc na jaka wartość ma być minimalizowana przy tworzeniu kolejnych gałęzi (najczęściej jest to błąd klasyfikacji) oraz kryterium stopu (a wiec jak długo drzewo ma być dzielone). Różne warianty drzew umożliwiają kontrolę różnych kryteriów, w przypadku metody `ctree()` zarówno kryterium stopu jak i podziału można określić argumentem `control` (patrz opis funkcji `ctree_control(party)`). Poniżej przedstawiamy przykładową sesję z budową drzewa klasyfikacyjnego.

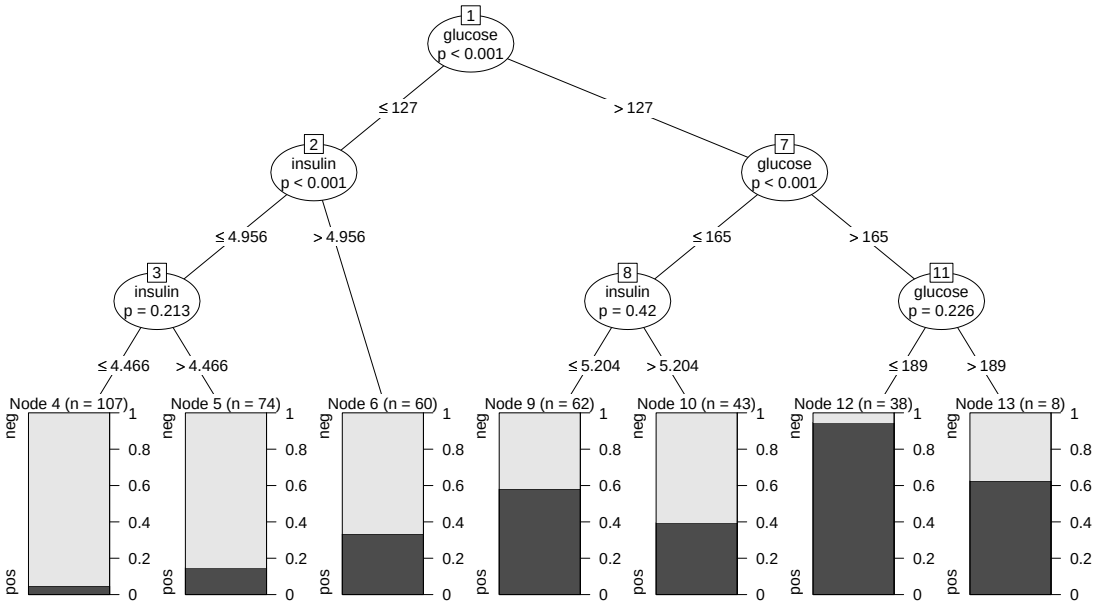
```
> # określamy kryteria budowy drzewa klasyfikayjnego
> ustawienia <- ctree_control(mincriterion = 0.5, testtype = "
   Teststatistic")
> # uczymy drzewo
```



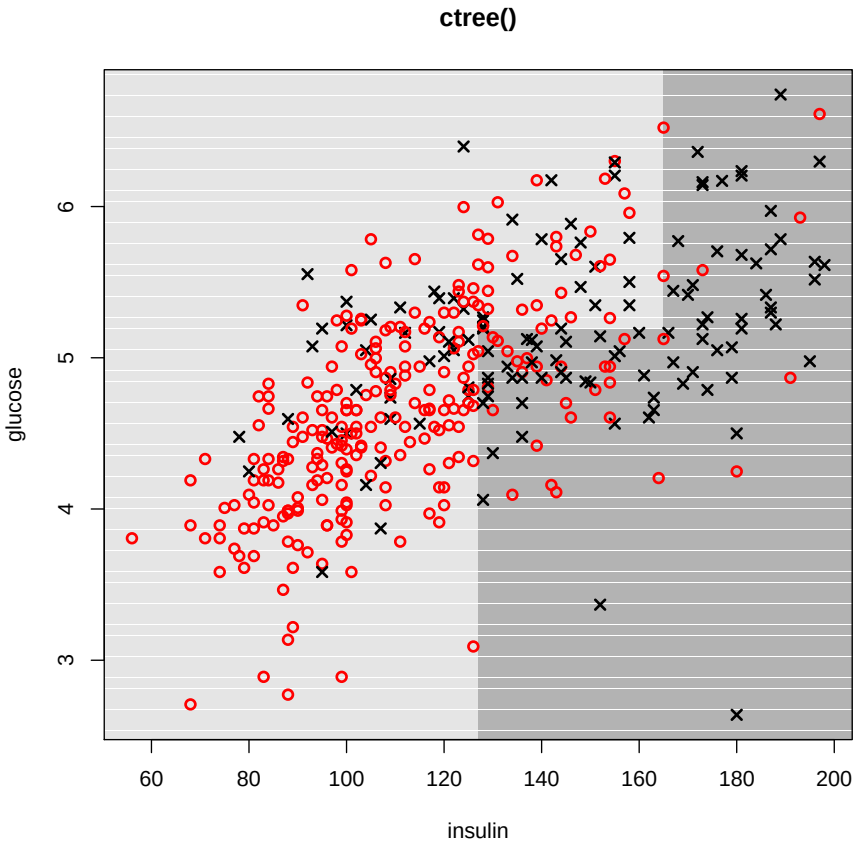
Rysunek 4.4: Warunkowe brzegowe oceny gęstości, na ich bazie funkcjonuje naiwny klasyfikator Bayesowski.



Rysunek 4.5: Przykładowe obszary decyzyjne dla naiwnego klasyfikatora Bayesowskiego.



Rysunek 4.6: Przykładowe drzewo klasyfikacyjne wyznaczone funkcją *ctree()*.



Rysunek 4.7: Przykładowe obszary decyzyjne dla drzewa klasyfikacyjnego.

```
> drzewo     <- ctree(diabetes~glucose+insulin, data=dane, subset = zbior.
   uczacy, controls = ustawienia)
> # narysujmy je
> plot(drzewo)
> # w standardowy sposób przeprowadzamy klasyfikacje
> oceny = predict(drzewo, dane[-zbior.uczacy,])
> table(predykcja = oceny, prawdziwe = dane[-zbior.uczacy,3])
         prawdziwe
predykcja neg pos
      neg 111  26
      pos  25  34
```

Zaletą drzew jest ich łatwość w interpretacji. Narysowany klasyfikator może być oceniony i poprawiony, przez eksperta z dziedziny, której problem dotyczy.

Dla drzew zapisanych jako obiekty klasy `tree` lub `rpart` dostępne są dwie przydatne funkcje umożliwiające modyfikację drzewa. Pierwsza to `prune.tree(tree)` (`prune.rpart(rpart)`). Pozwala ona na automatyczne przycinanie drzewa, poszczególnymi argumentami tej funkcji możemy ustalić jak drzewo ma być przycięte. Można określić pożądaną liczbę węzłów w drzewie, maksymalny współczynnik błędu w liściu drzewa oraz inne kryteria. Nie zawsze przycinanie automatyczne daje satysfakcjonujące rezultaty. W sytuacji gdy drzewo potrzebuje ludzkiej ingerencji można wykorzystać funkcję `snip.tree(tree)` (`snip.rpart(rpart)`) pozwalającą użytkownikowi na wskazanie myszką które węzły drzewa maja być usunięte. Inne ciekawe funkcje to `misclass.tree(tree)` (wyznacza błąd klasyfikacji dla każdego węzła z drzewa) oraz `partition.tree(tree)` (wyznacza obszary decyzyjne dla drzew).

4.5 Lasy losowe

Wadą drzew klasyfikacyjnych jest ich mała stabilność. Są jednak sposoby by temu zaradzić. Takim sposobem jest konstruowanie komitetu klasyfikatorów a wiec użycie metody bagging lub boosting. Nie wystarczyło tu miejsca by przedstawić te metody w ich ogólnej postaci, wspomnimy o nich na przykładzie lasów losowych.

Idea, która przyświeca metodzie lasów losowych możne być streszczona w zdaniu „Niech lasy składają się z drzew". Jak pamiętamy w metodzie bootstrap generowano replikacje danych, by ocenić zachowanie statystyki dla oryginalnego zbioru danych. Odmianą metody bootstrap w zagadnieniu klasyfikacji jest bagging. Na bazie replikacji zbioru danych konstruowane są klasyfikatory, które na drodze głosowania większością wyznaczają ostateczną klasą dla danej obserwacji. W przypadku lasów losowych komitet klasyfikatorów składa się z drzew klasyfikacyjnych, które są trenowane na replikacjach zbioru danych, dla ustalonego podzbioru zmiennych.

Ponieważ drzew w lesie jest dużo, do komitet głosujący demokratycznie charakteryzuje się większą stabilnością. Dodatkową zaletą drzew jest naturalny nieobciążony estymator błędu klasyfikacji. Generując replikacje losując metodą z powtórzeniami, średnio do replikacji nie trafia około jednej trzeciej obserwacji (w ramach ćwiczeń warto to sprawdzić). Te obserwacje, które nie trafiły do replikacji, można wykorzystać do oceny klasyfikatora nauczonego na danej replikacji. Taka ocena błędu okre-

ślana jest błędem OOB (ang. out-of-bag). Jeszcze inną zaletą drzew losowych jest możliwość oceny zdolności dyskryminujących dla poszczególnych zmiennych (dzięki czemu możemy wybrać najlepszy zbiór zmiennych).

Lasy losowe dostępne są w funkcji `randomForest(randomForest)`. Poniżej przedstawiamy przykład jej użycia. Ponieważ przy konstrukcji lasów losowych generowane są losowe replikacje zbioru danych, to aby móc odtworzyć uzyskane wyniki warto skorzystać z funkcji `set.seed()`.

```
> # ustawiamy ziarno generatora, by można było odtworzyć te wyniki
> set.seed(1)
> # ćwiczymy las
> klasyfikatorRF <- randomForest(diabetes~glucose+insulin, data=dane,
    subset=zbior.uczacy, importance=TRUE, proximity=TRUE)
> # podsumowanie lasu
> print(klasyfikatorRF)

Call:
 randomForest(formula = diabetes ~ glucose+insulin, data = dane,
     importance = TRUE,      proximity = TRUE, subset = zbior.uczacy)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 21.94%
Confusion matrix:
    neg pos class.error
neg 106  20   0.1587302
pos  23  47   0.3285714
> # podobnie jak dla innych klasyfikatorów funkcją predict() możemy ocenić
     etykietki na zbiorze testowym i porównać błąd klasyfikacji z błędem z
     innych metod
> oceny = predict(klasyfikatorRF, dane[-zbior.uczacy,])
> table(predykcja = oceny, prawdziwe = dane[-zbior.uczacy,"diabetes"])
         prawdziwe
predykcja neg pos
      neg 114  25
      pos  22  35
```

Lasy losowe są uznawane za jedną z najlepszych metod klasyfikacji. Ich dodatkową zaletą jest możliwość użycia nauczonego lasu losowego do innych zagadnień niż tylko do klasyfikacji. Przykładowo na podstawie drzew z lasu można wyznaczyć ranking zmiennych, a tym samym określić które zmienne mają lepsze właściwości predykcyjne a które gorsze. Taką informację przekazuje funkcja `importance(randomForest)`, można tę informacje również przedstawić graficznie z użyciem funkcji `varImpPlot(randomForest)` (przykład dla naszego zbioru danych jest przedstawiony na rysunku 4.9).

Używając lasów losowych (regresyjnych) można również wykonać imputacje, a wiec zastąpić brakujące obserwacje w zbiorze danych. Do tego celu można posłużyć się funkcją `rfImpute(randomForest)`. Można również zdefiniować na podstawie

lasów losowych miarę odstępstwa i wykrywać nią obserwacje odstające (do tego celu służy funkcja `outlier(randomForest)`). Można rónież używając lasów losowych przeprowadzać skalowanie wielowymiarowe, osoby zainteresowane tym tematem powinny zaznajomić się z funkcją `MDSplot(randomForest)`

4.6 Inne klasyfikatory

Powyżej wymienione metody to zaledwie początek góry lodowej funkcji do klasyfikacji dostępnych w pakiecie R. Poniżej wymienimy jeszcze kilka nazw popularnych klasyfikatorów, oraz pokażemy w których funkcjach i pakietach dane metody są dostępne.

- **Sieci neuronowe**.

  Sieci neuronowe gotowe do użycia w zagadnieniach klasyfikacji są dostępne w funkcjach `nnet(nnet)` (prosta w użyciu sieć z jedną warstwą neuronów ukrytych) i `train(AMORE)` (bardziej zaawansowana funkcja do uczenia sieci neuronowych).

- **Metoda wektorów podpierających (SVM, ang. Support Vector Machines)**.

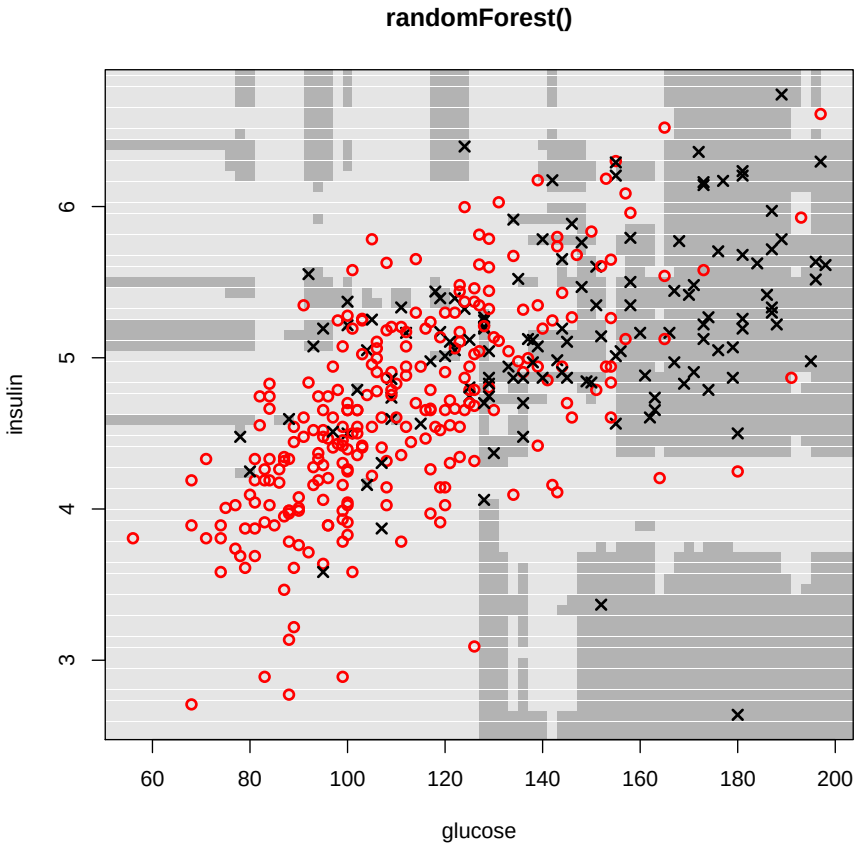
  Ideą wykorzystywaną w tej technice jest zwiększenie wymiaru przestrzeni obserwacji (przez dodanie nowych zmiennych np. transformacji zmiennych oryginalnych) tak by obiekty różnych klas można było rozdzielić hiperpłaszczyznami. Ta technika zyskała wielu zwolenników, funkcje pozwalające na jej wykorzystanie znajdują się w kilku pakietach np. `ksvm(kernlab)`, `svm(1071)`, `svmlight(klaR)`, `svmpath(svmpath)`.

- **Nearest mean classification i Nearest Shrunken Centroid Classifier** .

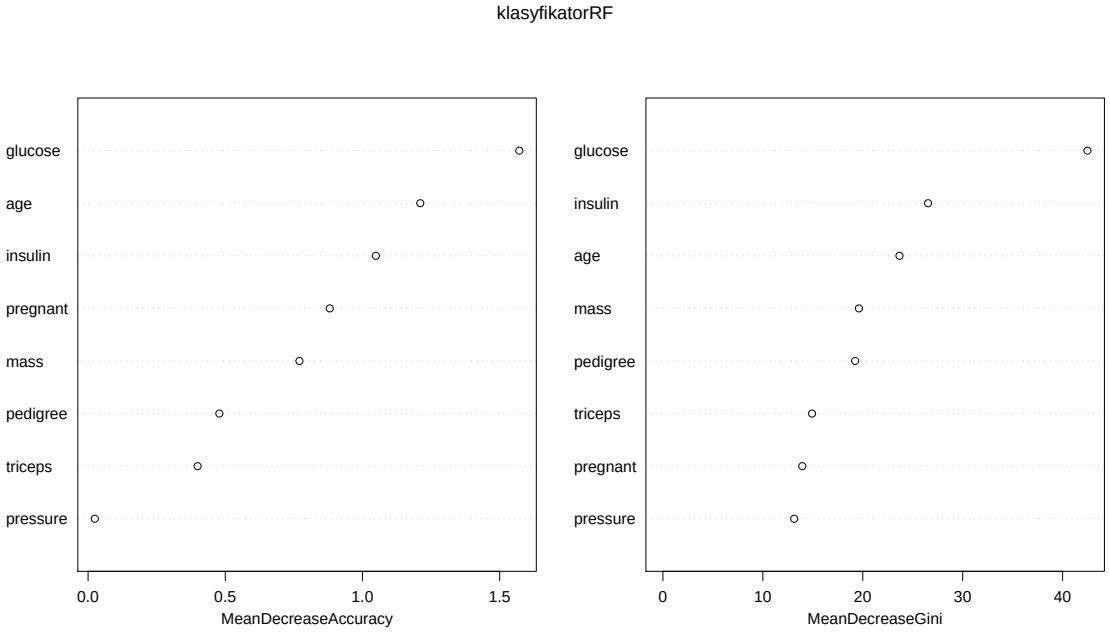
  Zasady działania podobne jak dla metod analizy skupień k-średnich i PAM. Metoda Nearest mean classification dostępna w funkcji `nm(klaR)` wyznacza średnie dla obiektów z danej klasy i nowe obiekty klasyfikuje do poszczególnych klas na podstawie wartości odległości nowej obserwacji od wyznaczonych średnich. Metoda Nearest Shrunken Centroid Classifier dostępna w funkcji `pamr.train(pamr)` działa podobnie, tyle, że zamiast średnich wyznacza centroidy.

- **Metody bagging i boosting**.

  Idea obu metod opiera się na tworzeniu replikacji zbioru danych, na których uczone są klasyfikatory. Wiele funkcji ze wsparciem dla tych metod znajduje się w pakiecie `boost` (boosting) i `ipred` (bagging), np. funkcje `adaboost(boost)`, `bagboost(boost)`, `l2boost(boost)`, `logitboost(boost)`, `ipredbagg(ipred)`.



Rysunek 4.8: Przykładowe obszary decyzyjne dla lasów losowych.



Rysunek 4.9: Ranking zmiennych wykonany przez las losowy.

Rozdział 5

Analiza kanoniczna

Podstawowe problemy i wyniki analizy kanonicznej zostały sformułowane przez Harolda Hotellinga (wybitny ekonomista, matematyk, statystyk) w latach 1935-36.

Powstała jako metoda do badania zależności pomiędzy dwoma zbiorami zmiennych.

Do dziś doczekała się wielu uogólnień i rozszerzeń, np. na badanie relacji pomiędzy wieloma zbiorami zmiennych, na badane relacji w obecności współliniowych zmiennych (przez regularyzację) itp.

5.1 Problem

Mamy dwa zbiory zmiennych $\{X_1, ..., X_p\}$ i $\{Y_1, ..., Y_q\}$.

Chcemy znaleźć taką kombinację liniową zmiennych z pierwszego zbioru, aby korelowała ona możliwie najsilniej ze zmiennymi z drugiego zbioru.

Innymi słowy, szukamy wektorów współczynników a i b, takich, że

$$cor(a'X, b'Y)$$

jest możliwie największa.

5.2 Rozwiązanie

Wektor współczynników a to wektor własny odpowiadający największej wartości własnej macierzy

$$S_{22}^{-1} S_{21} S_{11}^{-1} S_{12} \tag{5.1}$$

a wektor współczynników b to wektor własny odpowiadający największej wartości własnej macierzy

$$S_{11}^{-1} S_{12} S_{22}^{-1} S_{21}. \tag{5.2}$$

Korelacja $cor(a'X, b'Y)$ to wartość największa wartość własna z powyższych macierzy.

[Wyprowadzenie na tablicy]

Nowe zmienne $u_1 = a'X$ i $v_1 = b'Y$ wyjaśniają największą część korelacji pomiędzy zbiorami wektorów X i Y, ale nie całą.

Kolejnym krokiem jest znalezienie kolejnych zmiennych $u_i = a_i X$ i $v_i = b_i Y$, tak by:

- wektory u_i są nieskorelowane pomiędzy sobą,
- wektory v_i są nieskorelowane pomiędzy sobą,
- korelacje $cor(u_i, v_i)$ tworzą nierosnący ciąg odpowiadający możliwie największym cząstkowym korelacjom.

Jeżeli obserwacje pochodzą z wielowymiarowego modelu normalnego $\mathcal{N}(\mu, \Sigma)$ to możemy testować:

$$H_0 : R_i = 0 \forall_i$$

Statystyka testowa dla testu ilorazu wiarogodności

$$LRT = -n \sum_{i=1}^{s} \log(1 - R_i^2)$$

ma asymptotyczny rozkład χ^2_{pq}.

$$H_0 : R_i = 0 \forall_{i>k}$$

Statystyka testowa dla testu ilorazu wiarogodności

$$LRT = -n \sum_{i=k+1}^{s} \log(1 - R_i^2)$$

ma asymptotyczny rozkład $\chi^2_{(p-k)(q-k)}$.

Wartość n w statystykach testowych zamienia się czasem na $n - \frac{1}{2}(p + q + 3)$, co poprawia test.

5.3 Założenia

- wielowymiarowa normalność,
- brak obserwacji odstających (miara Cooka, Leverage, test Grubbsa, test Dixona)
- brak współliniowości (reguła kciuka, wyznacznik $> 10^{-5}$)

Liczba obserwacji powinna być większa od około 20$*$liczba zmiennych.

5.4 Jak to zrobić w R

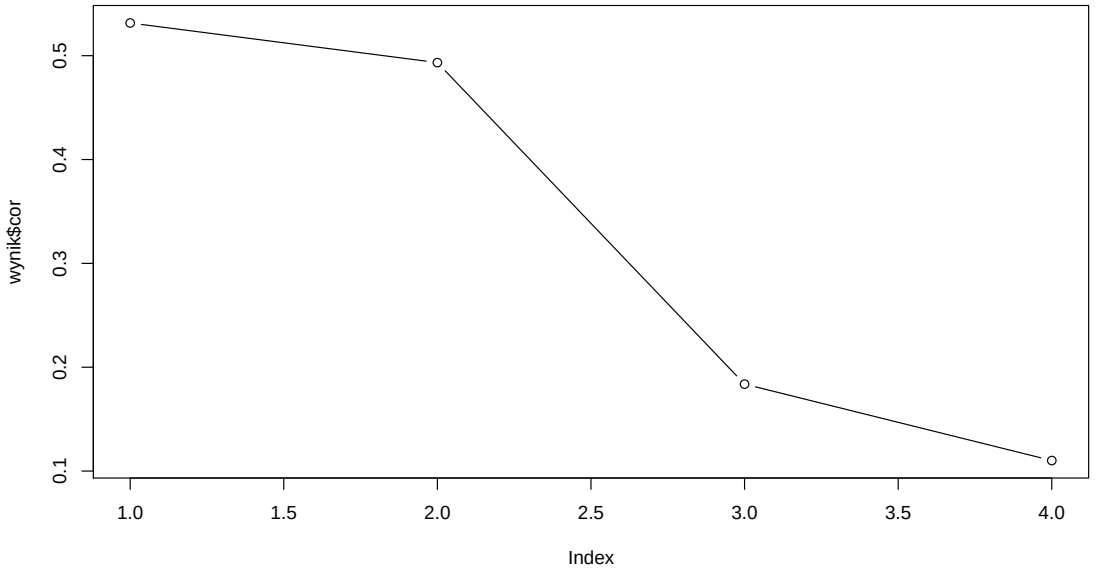
Analiza kanoniczna jest zaimplementowana między innymi w pakiecie CCA w funkcji $cc()$.

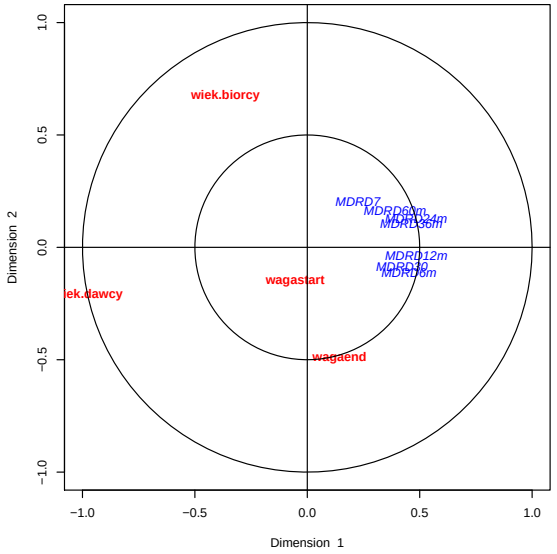
Prześledźmy poniższy kod

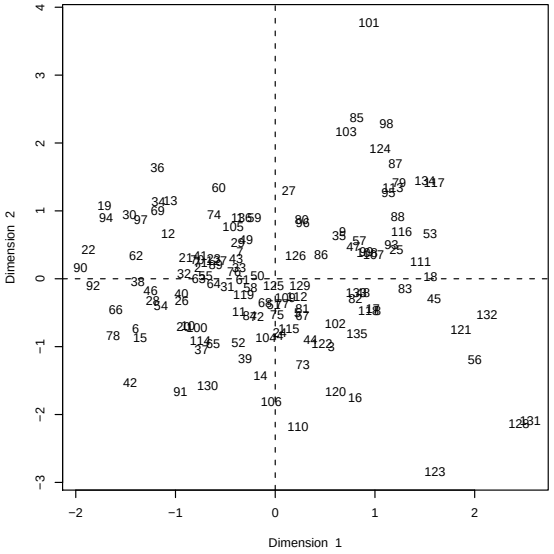
```
> library(CCA)
> dane = read.table("dane.csv",header=T,sep=";")
> X = dane[,c(9:10)]      # kolumny z waga
> Y = dane[,c(11:17)]     # kolumny z MDRD
> wynik = cc(X,Y)
> wynik$cor
[1] 0.3754946 0.1907164
```

```
> wynik$xcoef
                [,1]        [,2]
wagastart  0.1047822 -0.09276486
wagaend   -0.1154909  0.01404359
> wynik$ycoef
                [,1]        [,2]
MDRD7    0.056059823  0.05799373
MDRD30  -0.059196976 -0.03981322
MDRD6m  -0.006987328  0.02870234
MDRD12m -0.094082377  0.07732582
MDRD24m  0.119735985 -0.09688825
MDRD36m -0.024980200 -0.01744831
MDRD60m -0.007345604  0.04083270
> plot(wynik$cor,type="b")
> plt.cc(wynik,var.label=T)
```

5.5 Przykładowe wyniki







5.6 Studium przypadku

Rozdział 6

Analiza korespondencji (odpowiedniości)

6.1 Problem

Obserwujemy dwie zmienne jakościowe. Pierwsza zmienna przyjmuje wartości z n_1 poziomów, druga z n_2 poziomów.

Interesuje nas, czy zmienne te są od siebie niezależne, a jeżeli nie są niezależne to które kombinacje poziomów występują ze sobą znacznie częściej.

Dane z którymi mamy do czynienia to macierz kontyngencji (tabela wielodzielcza) o wymiarach $k \times l$ wyznaczona dla dwóch zmiennych o odpowiednio k i l poziomach. Jeżeli k i l są małe, to taką macierz można ogarnąć rzutem oka. Jednak dla zmiennych występujących na wielu poziomach potrzebne są już dodatkowe narzędzia.

6.2 Rozwiązanie

Aby ocenić niezależność dwóch zmiennych można wykorzystać test χ^2 z funkcji `chisq.test()` lub inny test dla macierzy kontyngencji (jeżeli poziomy są uporządkowane to dobrym rozwiązaniem będzie test Cochrana-Armitage). Testem tym zweryfikujemy hipotezę o niezależności częstości występowań poszczególnych zmiennych.

Jeżeli jednak odrzucimy tę hipotezę zerową, czyli przyjmiemy że jest JAKAŚ zależność, to naturalnym pytaniem będzie jaka to zależność i pomiędzy którymi poziomami.

Aby ocenić, które zmienne występują częściej ze sobą można wykonuje się tzw. analizę korespondencji. Jeżeli zmienne były by niezależne od siebie, to zachodziło by równanie

$$p_{ij} = p_{i\cdot} p_{\cdot j}, \qquad i \in \{1...k\},\ j \in \{1...l\}.$$

gdzie p_{ij} to prawdopodobieństwo zaobserwowania pierwszej zmiennej na poziomie i i jednocześnie drugiej na poziomie j, $p_{i\cdot}$ to prawdopodobieństwo zaobserwowania zmiennej pierwszej na poziomie i a $p_{\cdot j}$ to prawdopodobieństwo zaobserwowania zmiennej drugiej na poziomie j.

Żeby ocenić, które zmienne występują częściej lub rzadziej niż wynikało by to z niezależności, wyznaczymy standaryzowane reszty Pearsonowskie, zastąpimy też prawdopodobieństwa ich częstościowymi ocenami (czyli $\hat{p}$ to liczba obserwowanych zdarzeń podzielona przez liczbę wszystkich obserwowanych zdarzeń)

$$\hat{e}_{ij} = \frac{\hat{p}_{ij} - \hat{p}_{i\cdot}\hat{p}_{\cdot j}}{\hat{p}_{i\cdot}\hat{p}_{\cdot j}}.$$

Duże dodatnie wartości $\hat{e}_{ij}$ odpowiadają wysokiemu współwystępowaniu, ujemne wartości odpowiadają występowaniu rzadszemu niż losowe. Możemy teraz macierz $E = [\hat{e}_{ij}]$ przedstawić w postaci graficznej używając tzw. biplotu. Innymi słowy wyznaczamy dekompozycję SVD macierzy E

$$E_{k\times l} = U_{k\times k}\Sigma_{k\times l}V^T_{l\times l}.$$

Kolumny macierzy $U_{k\times k}$ to wektory własne macierzy E^TE a kolumny macierzy V to wektory własne macierzy EE^T. Na przekątnej macierzy diagonalnej Σ znajdują się tzw. wartości singularne (osobliwe?) równe pierwiastkom z wartości własnych macierzy E^TE i EE^T. „Przy okazji" kolumny macierzy U rozpinają ortonormalną bazę na kolumnach macierzy E a kolumny macierzy V rozpinają ortonormlaną bazę na wierszach macierzy E. Można więc przedstawić macierz E (lub możliwie dużo informacji z tej macierzy) we nowej przestrzeni określonej na bazie współrzędnych wierszy i kolumn macierzy E.

6.3 Jak to zrobić w R?

Przedstawimy funkcje `ca(ca)` oraz `corresp(MASS)`. Obie służą do wykonania analizy korespondencji. Wykonują ją jednak w odrobinę odmienny sposób przez co wyniki końcowe też są inne.

6.4 Studium przypadku

Przedstawmy przykład bazujący na danych o zatrudnieniu w poszczególnych województwach. Zmienne które chcemy porównać to Województwo (16 poziomów) i Sektor pracy (4 poziomy: rolnictwo, przemysł, usługi, bezrobotni). Podejrzewamy, że struktura zatrudnienia różni się pomiędzy województwami.

```
> library(MASS)
> library(ca)
> library(RColorBrewer)
> # przygotujmy wektor kolorow
> kolory = rev(brewer.pal(11,"Spectral"))

> # wybierzmy interesujące nas kolumny
> # konwertujemy tabele na macierz, takiego formatu spodziewa się funkcja
    heatmap()
> dane = as.matrix(daneGUS[,c(22:25)])
```

```
> colSums(dane)
  pr.rolnictwo  pr.przemysl    pr.uslugi   bezrobotni
          2249         4680         8309          743
> rowSums(dane)
       DOLNOSLASKIE  KUJAWSKO-POMORSKIE             LODZKIE
               1218                  791                1304
          LUBELSKIE            LUBUSKIE         MALOPOLSKIE
               1018                  451                1336
        MAZOWIECKIE            OPOLSKIE        PODKARPACKIE
               2373                  379                 852
          PODLASKIE           POMORSKIE             SLASKIE
                478                  792                1845
     SWIETOKRZYSKIE WARMINSKO-MAZURSKIE       WIELKOPOLSKIE
                622                  573                1371
 ZACHODNIOPOMORSKIE
                578

> # jak wygląda macierz z danymi?
> head(dane)
                   pr.rolnictwo pr.przemysl pr.uslugi bezrobotni
DOLNOSLASKIE                 74         415       651         78
KUJAWSKO-POMORSKIE          128         245       369         49
LODZKIE                     220         384       636         64
LUBELSKIE                   328         197       447         46
LUBUSKIE                     43         151       244         13
MALOPOLSKIE                 205         381       689         61
```

Macierz 64 liczb jest trudno ogarnąć nieuzborojonym okiem. Możemy zauważyć, że biorąc pod uwagę, że najwięcej ludzi pracuje w sektorze usługowym (8 309 tys.) również łatwo zauważyć, że najludniejsze jest województwo Mazowieckie (2 373 tys.) ale z tych surowych danych ciężko wyciągnąć głębszy wniosek.

```
> # tę macierz można przedstawić w postaci graficznej
> # czerwony odpowiada dużym liczbom, niebieski małym
> heatmap(dane,scale="col",Colv=NA, col= kolory)

> # zobaczmy czy sa zaleznosci pomiedzy kolumnami i wierszami, wygląda na
    to że nie są to niezależne cechy
> chisq.test(dane)

        Pearson's Chi-squared test

data:  dane
X-squared = 1031.905, df = 45, p-value < 2.2e-16
```

Wykonaliśmy test χ^2 i otrzymaliśmy bardzo małą p-wartość, która upewniła nas w przekonaniu, że województwa mają inną strukturę zatrudnienia.

```
> # zobaczmy które kombinacje występują częściej niż w przypadku
    niezależności
> # policzymy residua Pearsonowskie
> P = dane/sum(dane)
> # macierz czestosci oczekiwanych
> PP = outer(rowSums(P),colSums(P))
> # macierz residuow Pearsonowskich
> E = (P-PP)/sqrt(PP)
> head(E)
                  pr.rolnictwo pr.przemysl pr.uslugi   bezrobotni
DOLNOSLASKIE           -0.05885     0.02442   0.00557     0.022465
KUJAWSKO-POMORSKIE      0.01250     0.00694  -0.01648     0.015945
LODZKIE                 0.02130     0.00086  -0.01275     0.003427
LUBELSKIE               0.12209    -0.04632  -0.02829    -0.001528
LUBUSKIE               -0.02032     0.01302   0.00491    -0.013765
MALOPOLSKIE             0.00979    -0.00409  -0.00168    -0.001118
> # tę macierz również można przedstawić za pomocą mapy ciepła
> heatmap(E,scale="none",Colv=NA,col= kolory)
```

Wyznaczyliśmy macierz reszt Pearsonowskich E (którą też przedstawiliśmy graficznie z użyciem mapy ciepła) i z tej macierzy możemy już odczytać w których województwach poszczególne sektory są popularniejsze. Największe reszty obserwujemy dla sektora rolnictwa, zarówno duże dodatnie wartości (w województwie lubelskim, podlaskim, podkarpackim i świętokrzyskim) jak i duże (co do modułu) ujemne wartości (w województwie śląskim i dolnośląskim).

Możemy teraz przedstawić graficznie macierz E z użyciem biplotu (z wykorzystaniem dekompozycji SVD).

```
> # wykonujemy dekompozycje na wartosci osobliwe (singularne/szczegolne)
> A = svd(E)
> X = t(apply(A$u, 1, "*", sqrt(A$d)))
> Y = t(apply(A$v, 1, "*", sqrt(A$d)))
# zwykle współrzędne liczone są ze wzorow, w ktorych A jest dekompozycja
    innej macierzy
# [TODO: uzupełnić albo pominąć]
# A = Dr^(-1/2) A$u A$d
# B = Dc^(-1/2) A$v a$d
> plot(rbind(X[,1:2], Y[,1:2]), xlab="", ylab="", main="Bramka nr. 1", lwd
    =3)

> # analiza korespondencji z użyciem pakietu MASS
> biplot(corresp(dane, nf = 2))
> # analiza korespondencji z użyciem pakietu ca
> # argument mass powoduje że liczebności komórek odpowiadają wielkościom
    punktów na wykresie
> plot(ca(dane), mass=c(T,T))
> summary(ca(dane))
```

```
Principal inertias (eigenvalues):

     dim    value      %   cum%   scree plot
[1,] 1      0.054859  85.0  85.0  ************************
[2,] 2      0.007352  11.4  96.3  **
[3,] 3      0.002360   3.7 100.0
[4,]        -------- -----
[5,] Total: 0.064571 100.0


Rows:
     name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
1  |  DOL |   76  918   71 | -225 837  70 |  -70  81  51 |
2  |  KUJ |   49  848   11 |   65 286   4 |  -91 563  56 |
3  |  LOD |   82 1000   10 |   80 838  10 |  -35 162  14 |
4  | LUBE |   64 1000  277 |  527 990 322 |   53  10  24 |
5  | LUBU |   28  723   12 | -142 719  10 |  -11   4   0 |
6  |  MAL |   84  995    2 |   37 960   2 |    7  34   1 |
7  |  MAZ |  148 1000   88 |  -95 238  25 |  171 762 587 |
8  |  OPO |   24  496    3 |   -5   2   0 |  -68 494  15 |
9  | PODK |   53  926   78 |  296 925  85 |  -12   2   1 |
10 | PODL |   30  994   56 |  342 974  64 |   49  20  10 |
11 |  POM |   50  939   24 | -171 919  26 |   25  20   4 |
12 |  SLA |  115  995  187 | -319 970 214 |  -52  25  42 |
13 |  SWI |   39  968  128 |  443 926 139 |  -94  41  47 |
14 |  WAR |   36  515    4 |  -42 246   1 |  -44 269  10 |
15 |  WIE |   86  808   15 |   -1   0   0 |  -97 808 109 |
16 |  ZAC |   36  806   33 | -203 702  27 |   78 103  30 |

Columns:
    name   mass  qlt  inr    k=1 cor ctr    k=2 cor ctr
1 | pr.r |  141  999  720 |  574 999 847 |   -4   0   0 |
2 | pr.p |  293  967  128 | -120 509  77 | -114 458 514 |
3 | pr.s |  520 1000  111 |  -90 587  77 |   75 413 402 |
4 |  bzr |   46  236   42 |   21   7   0 | -115 228  83 |
```

Przy opisie analizy korespondencji często wymienianym czynnikiem jest inercja, nazywana czasem bezwładnością (przez podobieństwo do inercji w fizyce). Aby opisać formalnie czym jest ta miara musimy zacząć od kilku innych pojęć.

Niech masa wiersza i masa kolumny będzie określona jako brzegowe częstości macierzy kontyngencji.

```
> (masa.kolumny = colSums(prop.table(dane)))
pr.rolnictwo  pr.przemysl    pr.uslugi   bezrobotni
  0.14072962   0.29284776   0.51992992   0.04649271
> (masa.wiersza = rowSums(prop.table(dane)))
       DOLNOSLASKIE  KUJAWSKO-POMORSKIE             LODZKIE
         0.07621551          0.04949628          0.08159690
```

```
          LUBELSKIE            LUBUSKIE         MALOPOLSKIE
         0.06370064          0.02822101          0.08359927
       MAZOWIECKIE            OPOLSKIE        PODKARPACKIE
         0.14848883          0.02371566          0.05331331
          PODLASKIE           POMORSKIE             SLASKIE
         0.02991052          0.04955885          0.11544960
    SWIETOKRZYSKIE WARMINSKO-MAZURSKIE       WIELKOPOLSKIE
         0.03892122          0.03585508          0.08578937
 ZACHODNIOPOMORSKIE
         0.03616795
```

Profilem wiersza (kolumny) będą wartości w wierszu (kolumnie) unormowane przez masę kolumny (wiersza).

```
> head(profil.kolumny <- dane/rowSums(dane))
                    pr.rolnictwo pr.przemysl pr.uslugi bezrobotni
DOLNOSLASKIE          0.06075534   0.3407225 0.5344828 0.06403941
KUJAWSKO-POMORSKIE    0.16182048   0.3097345 0.4664981 0.06194690
LODZKIE               0.16871166   0.2944785 0.4877301 0.04907975
LUBELSKIE             0.32220039   0.1935167 0.4390963 0.04518664
LUBUSKIE              0.09534368   0.3348115 0.5410200 0.02882483
MALOPOLSKIE           0.15344311   0.2851796 0.5157186 0.04565868
```

Tak unormowane profile wierszy i kolumn mogą być ze sobą porównywane. Im większa odległość pomiędzy profilami kolumn tym mniejsza zależność pomiędzy czynnikami opisanymi w tych kolumnach (oczywiście dla wierszy jest tak samo). Średni profil kolumnowy to masa wiersza a średni profil wierszowy to masa kolumn.

Inercja to odległość (w mierze χ^2) pomiędzy daną kolumną (wierszem, punktem) a średnią wartością dla kolumn (wierszy). Całkowita inercja to suma odległości dla wszystkich kolumn (wierszy). **Im większa inercja tym punkty są oddalone dalej od średniego profilu wiersza/kolumny.**

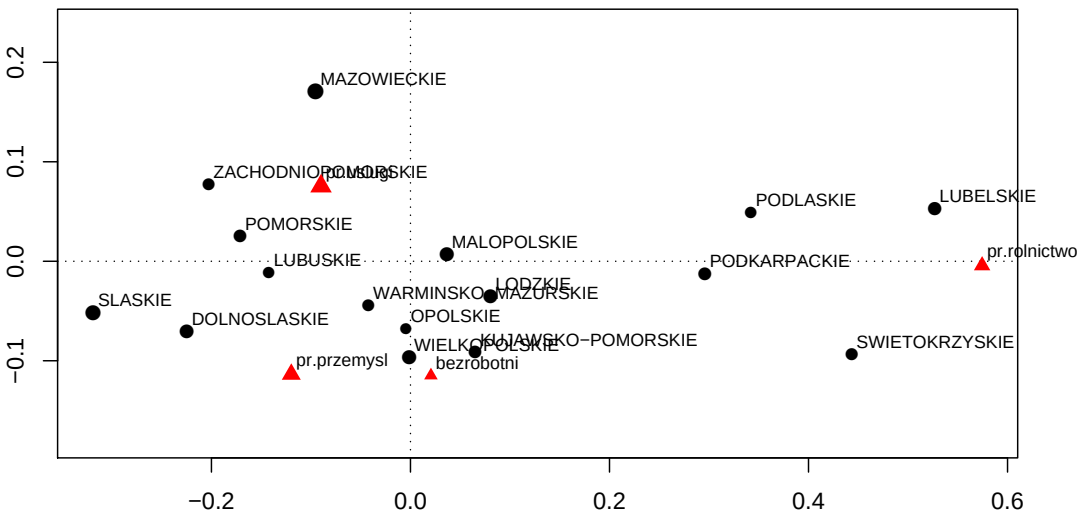
[TODO: uzupełnić. nawiązać do wyników funkcji summary.ca()]

Wyniki powyższych instrukcji oglądać można na wykresie 6.1 i kolejnych.

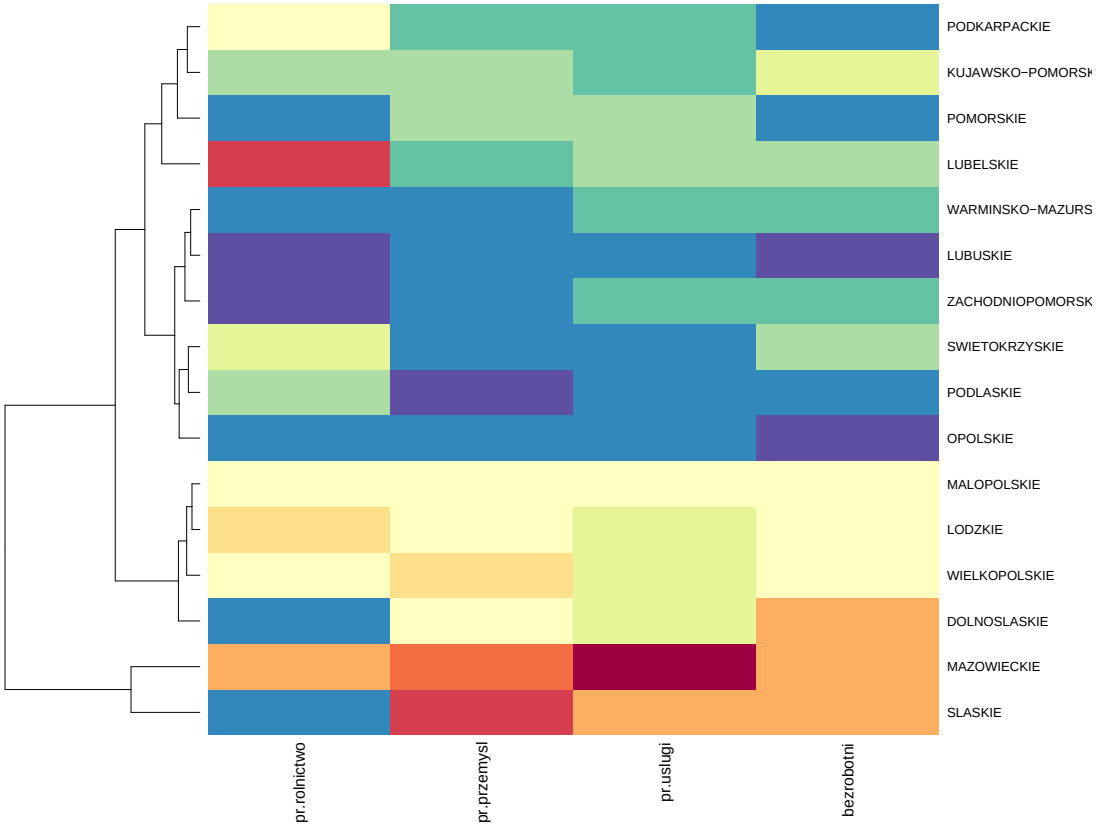
Analizując ten wykres można zauważyć ciekawe zróżnicowanie województw. Na osi X największą współrzędną ma zmienna `pr.rolnictwo`, a więc to ta zmienna odpowiada największemu zróżnicowaniu województw. Wysoki wartości na tej osi osiągają województwa gdzie zatrudnienie w rolnictwie było wyższe niż średnie. Druga oś rozróżnia województwa w których dominuje zatrudnienie w sektorze usług (dodatnie wartości) versus zatrudnieniu w przemyśle lub braku zatrudnienia (niestety profil zatrudnienia w przemyśle i braku zatrudnienia jest podobny).

Osoby w województwach Mazowieckim i Zachodniopomorskim częściej pracują w sektorze usług, w województwach Lubelskim, Podlaskim, Świętokrzyskim i Podkarpackim dominuje zatrudnienie w sektorze rolniczym, w województwach Śląskim i Dolnośląskim dominuje zatrudnienie w przemyśle te województwa mają też większe problemy z bezrobociem.

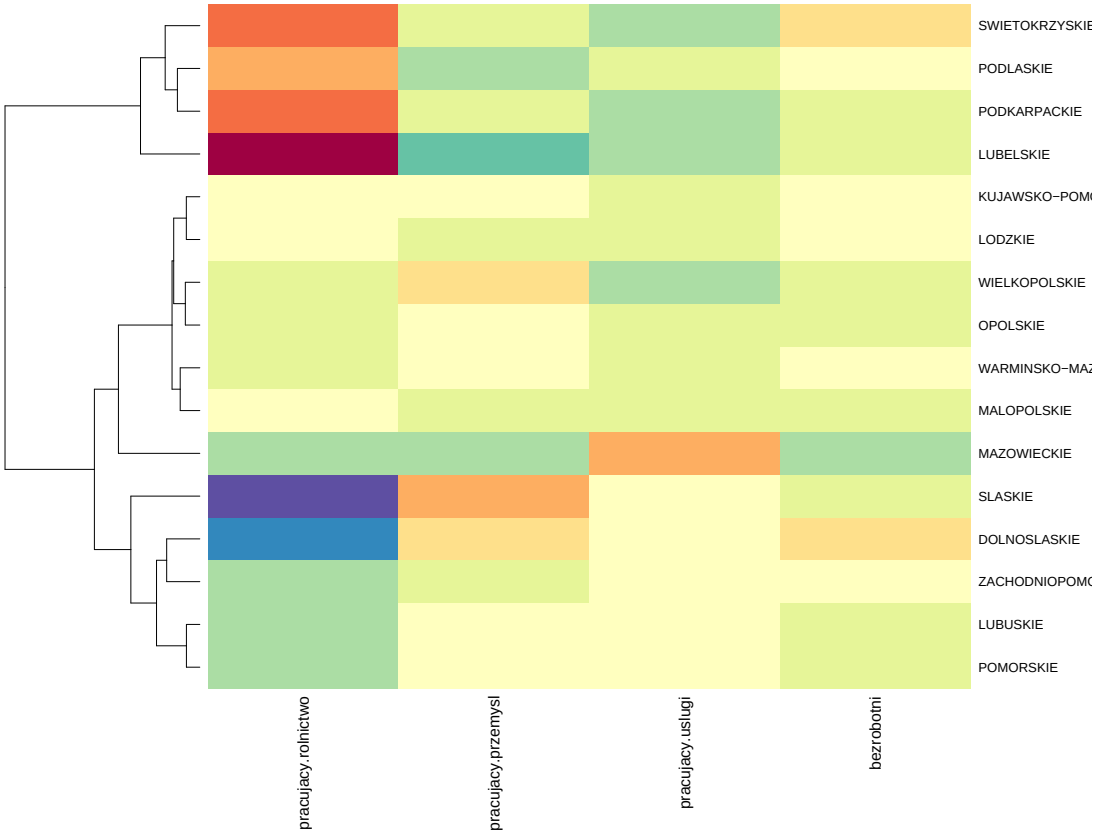
[TODO: opisać podobieństwa i różnice do PCA]



Rysunek 6.1: Przykład analizy korespondencji.



Rysunek 6.2: Mapa ciepła dla danych oryginalnych (kolory normalizowane po kolumnach).



Rysunek 6.3: Mapa ciepła dla reszt Pearsonowskich.

Rozdział 7

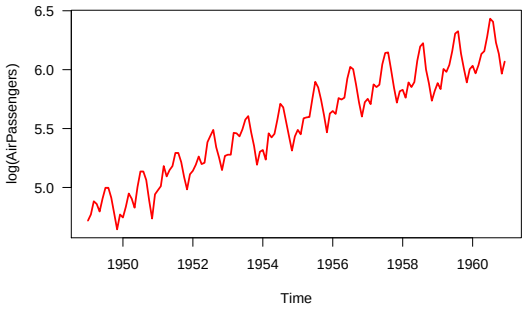
Przykład analizy szeregów czasowych

7.1 Wprowadzenie

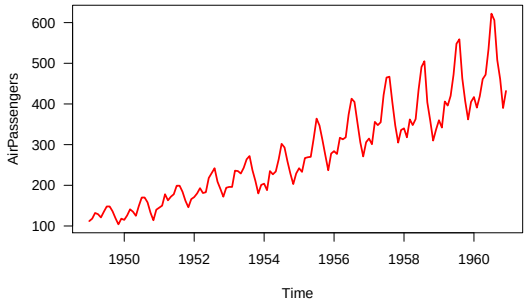
W skład szeregu czasowego mogą wchodzić następujące elementy:

- trend
  - deterministyczny
  - stochastyczny
- wahania sezonowe
  - addytywne
  - multiplikatywne

```
> plot(log(AirPassengers),col=2,las=1,lwd=2) # Rys. 7.1
> plot(AirPassengers,col=2,las=1,lwd=2)      # Rys. 7.2
```



Rysunek 7.1: Szereg addytywny.



Rysunek 7.2: Szereg multiplikatywny.

Wśród szeregów czasowych rozróżniamy dwa rodzaje procesów stochastycznych: stacjonarne (w których nie występuje trend) oraz niestacjonarne (w których występuje trend). Aby wyeliminować tendencję rozwojową z danego procesu można wykorzystać do tego celu różnicowanie:

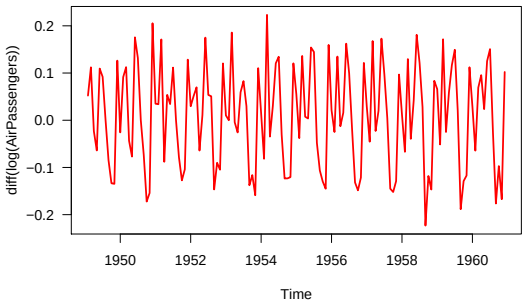
$$\Delta y_t = y_t - y_{t-1}$$

```
> dy=diff(y,lag=1,differences=1) # jednokrotne różnicowanie
```

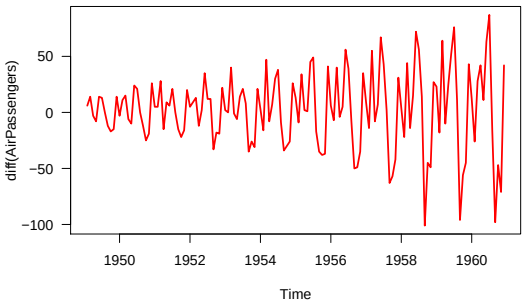
$$\Delta\Delta y_t = \Delta y_t - \Delta y_{t-1}$$

```
> ddy=diff(y,lag=1,differences=2) # dwukrotne różnicowanie
```

```
> plot(diff(log(AirPassengers)),col=2,las=1,lwd=2) # Rys. 7.3
> plot(diff(AirPassengers),col=2,las=1,lwd=2)      # Rys. 7.4
```



Rysunek 7.3: Brak trendu, sezonowość addytywna.



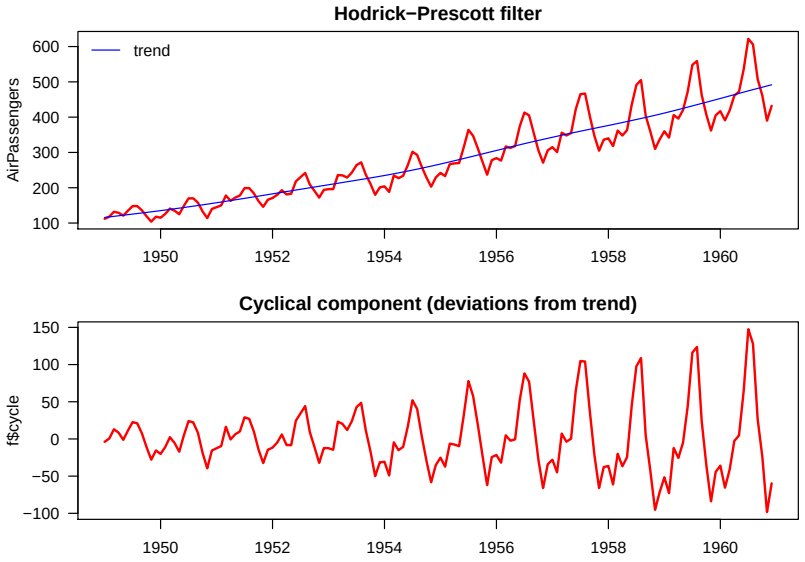
Rysunek 7.4: Brak trendu, sezonowość multiplikatywna.

W środowisku R dostępne są także funkcje dotyczące filtrowania szeregów czasowych. Jest to takie przekształcenie danych które doprowadza do oczyszczenia szeregu czasowego z wahań periodycznych. W paczce `mFilter` dostępnych jest kilka takich filtrów:

- Baxter-King — `bkfilter(mFilter)`,
- Christiano-Fitzgerald — `cffilter(mFilter)`,
- Butterworth — `bwfilter(mFilter)`,
- Trigonometric regression — `trfilter(mFilter)`,
- Hodrick-Prescott — `hpfilter(mFilter)`.

Stosując filtr HP należy pamiętać o odpowiednim doborze parametru λ. Hodrick oraz Prescott zalecają, aby wartość współczynnika λ była równa 400, 1600 i 14400 odpowiednio dla danych rocznych, kwartalnych i miesięcznych.

```
> library(mFilter)
# filtr Hodrick-Prescott dla danych miesięcznych
> f=mFilter(AirPassengers,filter="HP",freq=14400,drift=FALSE)
# wykresy
> par(mfrow=c(2,1),mar=c(3,4,2,1),cex=.8)
> plot(AirPassengers,las=1,col="red",lwd=2,main="Hodrick-Prescott filter")
> lines(f$trend,col='blue')
> legend('topleft','trend',col='blue',lty=1,bty="n")
> plot(f$cycle,las=1,col='red',lwd=2,
main='Cyclical component (deviations from trend)')
```

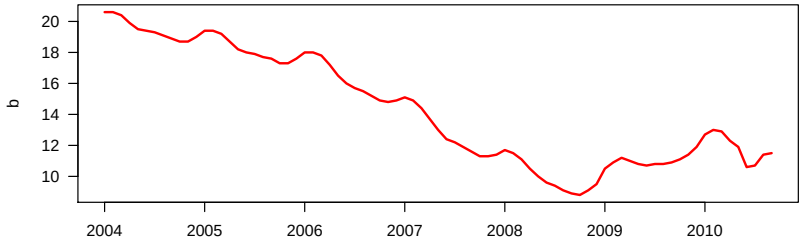


Rysunek 7.5: Filtr Hodrick-Prescott.

7.2 Identyfikacja trendu i sezonowości

7.2.1 Analiza wariancji — ANOVA

Ocena występowania trendu oraz wahań periodycznych zostanie dokonana na przykładzie miesięcznej stopy bezrobocia w Polsce w okresie od 01-2004 do 10-2010 roku.



Rysunek 7.6: Stopa bezrobocia w Polsce od 01-2004 do 10-2010 roku.

```
# wektor danych:
> b=c(
20.6, 20.6, 20.4, 19.9, 19.5, 19.4, 19.3, 19.1, 18.9, 18.7, 18.7, 19.0,
19.4, 19.4, 19.2, 18.7, 18.2, 18.0, 17.9, 17.7, 17.6, 17.3, 17.3, 17.6,
18.0, 18.0, 17.8, 17.2, 16.5, 16.0, 15.7, 15.5, 15.2, 14.9, 14.8, 14.9,
15.1, 14.9, 14.4, 13.7, 13.0, 12.4, 12.2, 11.9, 11.6, 11.3, 11.3, 11.4,
11.7, 11.5, 11.1, 10.5, 10.0,  9.6,  9.4,  9.1,  8.9,  8.8, 9.1,  9.5,
10.5, 10.9, 11.2, 11.0, 10.8, 10.7, 10.8, 10.8, 10.9, 11.1, 11.4, 11.9,
12.7, 13.0, 12.9, 12.3, 11.9, 10.6, 10.7, 11.4, 11.5)
# przekształcenie wektora danych na szereg czasowy:
> b=ts(b,freq=12,start=c(2004,1))
```

```
> T=rep(c(2004,2005,2006,2007,2008,2009,2010),c(12,12,12,12,12,12,9))
```

```
> T=factor(T)
> S=rep(1:12,7);S=S[-c(82:84)]
> S=factor(S)
```

Identyfikację szeregu czasowego można przeprowadzić na kilka sposobów. Jednym z nich jest analiza wariancji ANOVA.

```
> anova(lm(b~T+S))
Analysis of Variance Table

Response: b
          Df Sum Sq Mean Sq F value    Pr(>F)
T          6 990.99 165.165 523.410 < 2.2e-16 ***
S         11  51.67   4.697  14.886 1.130e-13 ***
Residuals 63  19.88   0.316
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Na podstawie przeprowadzonej analizie wariancji, należy stwierdzić, że w omawianym szeregu czasowym występuje trend (p-value $= 2.2e-16$). W celu dokonania dalszej identyfikacji szeregu (oceny występowania sezonowości) należy wyeliminować tendencję rozwojową (np. poprzez różnicowanie).

```
> anova(lm(diff(b)~T[-1]+S[-1]))
Analysis of Variance Table

Response: diff(b)
          Df Sum Sq Mean Sq F value    Pr(>F)
T[-1]      6 1.7879 0.29798  8.1625 1.617e-06 ***
S[-1]     11 6.8836 0.62578 17.1420 6.600e-15 ***
Residuals 62 2.2634 0.03651
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Jednokrotne różnicowanie szeregu nie przyniosło porządanego efektu (p-value $= 1.617e-06$). Zatem powinniśmy badany szereg poddać dwukrotnemu różnicowaniu. Należy tę procedurę powtarzać, aż do skutku czyli wyeliminowamia trendu.

```
> anova(lm(diff(b,1,2)~T[-c(1,2)]+S[-c(1,2)]))
Analysis of Variance Table

Response: diff(b, 1, 2)
            Df Sum Sq Mean Sq F value    Pr(>F)
T[-c(1, 2)]  6 0.0383 0.00639  0.1030    0.9958
S[-c(1, 2)] 11 4.3495 0.39541  6.3775 6.621e-07 ***
Residuals   61 3.7820 0.06200
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Dopiero dwukrotne różnicowanie przyniosło porządany efekt (p-value $= 0.9958$). Tak więc teraz możemy stwierdzić, że w badanym szeregu czasowym występuje sezonowość (p-value $= 6.621e-07$).

```
> summary(lm(b~T+S))

Call:
lm(formula = b ~ T + S)

Residuals:
     Min       1Q   Median       3Q      Max
-1.75126 -0.20245  0.02639  0.19755  1.45139

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 20.75959    0.26044  79.710  < 2e-16 ***
T2005       -1.31667    0.22933  -5.741 2.91e-07 ***
T2006       -3.30000    0.22933 -14.390  < 2e-16 ***
T2007       -6.74167    0.22933 -29.397  < 2e-16 ***
T2008       -9.57500    0.22933 -41.752  < 2e-16 ***
T2009       -8.50833    0.22933 -37.101  < 2e-16 ***
T2010       -7.87546    0.25064 -31.422  < 2e-16 ***
S2           0.04286    0.30026   0.143 0.886958
S3          -0.14286    0.30026  -0.476 0.635883
S4          -0.67143    0.30026  -2.236 0.028894 *
S5          -1.15714    0.30026  -3.854 0.000275 ***
S6          -1.61429    0.30026  -5.376 1.18e-06 ***
S7          -1.71429    0.30026  -5.709 3.29e-07 ***
S8          -1.78571    0.30026  -5.947 1.31e-07 ***
S9          -1.91429    0.30026  -6.375 2.42e-08 ***
S10         -2.16931    0.31386  -6.912 2.85e-09 ***
S11         -2.08598    0.31386  -6.646 8.23e-09 ***
S12         -1.80265    0.31386  -5.744 2.88e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5617 on 63 degrees of freedom
Multiple R-squared: 0.9813,     Adjusted R-squared: 0.9762
F-statistic: 194.4 on 17 and 63 DF,  p-value: < 2.2e-16
```

Oszacowane współczynniki regresji liniowej wskazują jakie są różnice między średnimi stapami bezrobocia w poszczególnych latach oraz miesiącach. Punktem referencyjnym dla czynnika `T` jest 2004 rok, dla `S` styczeń. Aby zmienić punkt odniesienia np. na rok 2005 wystarczy wykonać komendę:

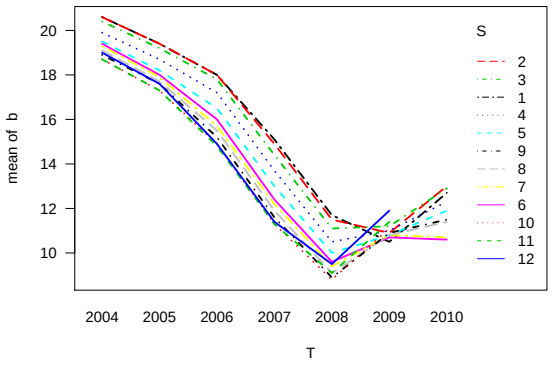
```
> T = relevel(T, ref="2005")
```

a następnie przeprowadzić regresję. Jeśli chcemy wyznaczyć średnie stopy bezrobocia w poszczególnych latach trzeba skorzystać z funkcji `tapply(base)`:

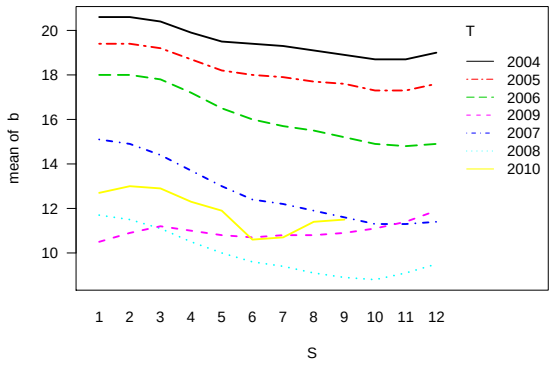
```
> tapply(b,T,mean)
     2004      2005      2006      2007      2008      2009      2010
19.508333 18.191667 16.208333 12.766667  9.933333 11.000000 11.888889
```

Prezentację graficzną kształtowania się średnich możemy wykonać w następujący sposób:

```
> interaction.plot(T,S,b,col=1:12,las=1,lwd=2) # Rys. 7.7
> interaction.plot(S,T,b,col=1:7,las=1,lwd=2)  # Rys. 7.8
```



Rysunek 7.7: Średnie dla miesięcy.

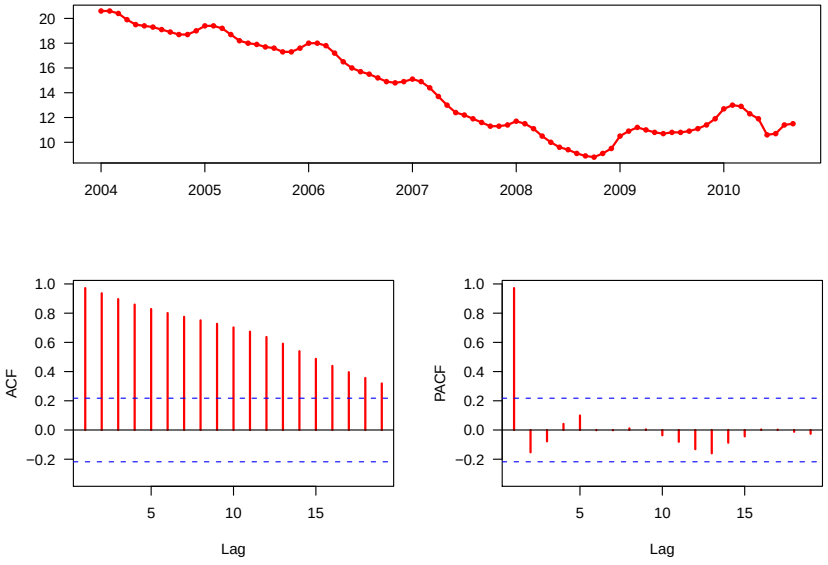


Rysunek 7.8: Średnie dla lat.

7.2.2 Funkcja autokorelacji — ACF

Do identyfikacji składowych szeregu czasowego (trendu oraz sezonowości) można również wykorzystać funkcję autokorelacji — ACF. W środowisku R jest dostępna funkcja graficzna `tsdisplay(forecast)`, za pomocą której zostają wygenerowane trzy wykresy: krzywa badanego zjawiska, funkcja autokorelacji ACF oraz funkcja częściowej autokorelacji PACF.

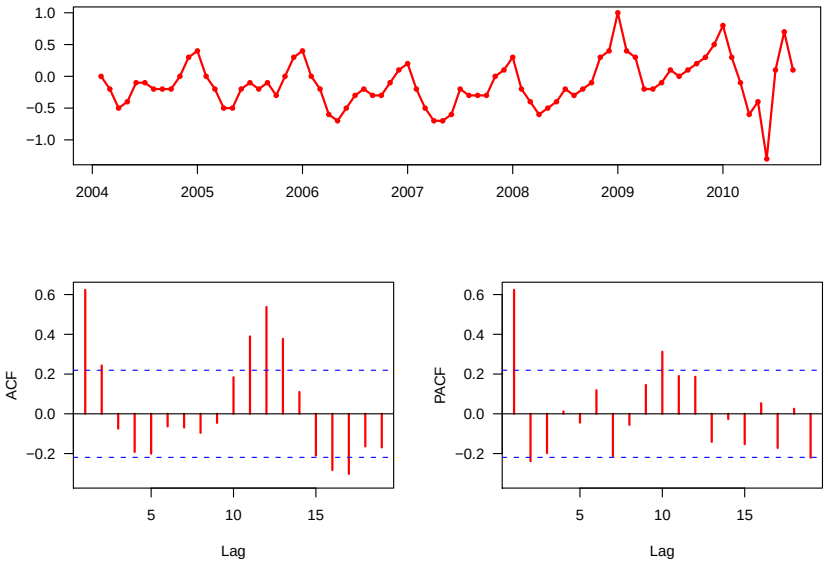
```
> tsdisplay(b,col=2,lwd=2,las=1)
```



Rysunek 7.9: Funkcja ACF oraz PACF.

Ponieważ funkcja autokorelacji ACF maleje wykładniczo wraz ze wzrostem parametru p (rys.7.9) należy sądzić, że w badanym procesie występuje trend.

```
> tsdisplay(diff(b),col=2,lwd=2,las=1)
```

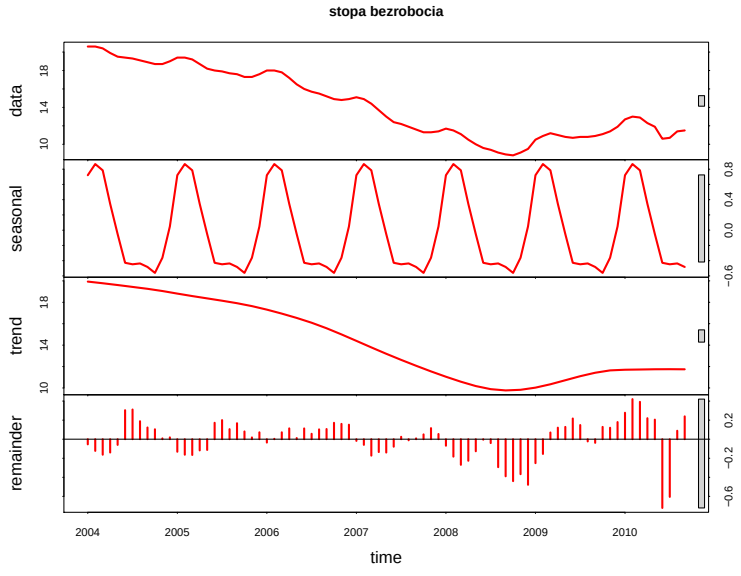


Rysunek 7.10: Funkcja ACF oraz PACF dla pierwszych różnic.

Po zastosowaniu funkcji różnicowania szeregu czasowego trend został wyeliminowany a funkcja ACF wskazuje na występowanie miesięcznych wahań sezonowych (rys.7.10).

Zatem na podstawie analizy ANOVA oraz funkcji ACF doszliśmy do wniosku, że badany szereg czasowy charakteryzuje się trendem oraz sezonowością. Również dzięki funkcji `stl(stats)` za pomocą której dokonaliśmy dekompozycji badanego procesu (rys. 7.11) możemy potwierdzić występowanie tendencji rozwojowej oraz wahań periodycznych.

```
> plot(stl(b,s.window="periodic"),col=2,lwd=2)
```



Rysunek 7.11: Dekompozycja szeregu czasowego.

7.3 Modele autoregresyjne $ARIMA$

Modele $ARIMA$ służą do analizy stacjonarnych szeregów czasowych. W przypadku gdy szereg jest niestacjonarny należy sprowadzić go do stacjonarności np. poprzez różnicowanie. W skład modelu $ARIMA(p, d, q)$ mogą wchodzić następujące elementy:

- $AR(p)$ — autoregresja (rząd opóźnienia p)
- $I(d)$ — stopień integracji szeregu (krotność różnicowania d)
- $MA(q)$ — średnia ruchoma (rząd opóźnienia q)

Z kolei gdy w badanym procesie występuje sezonowość należy estymować model o postaci $SARIMA(p, d, q)(P, D, Q)_m$ gdzie:

- m — sezonowość (np. dla $m = 4$ sezonowość kwartalna)
- P — autoregresja sezonowa
- D — integracja sezonowa
- Q — średnia ruchoma sezonowa

Prawie każdy model $ARIMA$ można zapisać w dwojaki sposób np dla procesu y_t:

- $ARIMA(2, 0, 0)$ czyli $AR(2)$:

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \varepsilon_t$$

- $ARIMA(2, 0, 1)$ czyli $ARMA(2, 1)$:

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} - \beta_1 \varepsilon_{t-1} + \varepsilon_t$$

- $ARIMA(2, 1, 0)$ czyli $ARI(2, 1)$:

$$\Delta y_t = \alpha_1 \Delta y_{t-1} + \alpha_2 \Delta y_{t-2} + \varepsilon_t$$

- $SARIMA(1, 0, 0)(2, 0, 0)_4$ czyli $SAR(1)(2)_4$:

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-4} + \alpha_3 y_{t-8} + \varepsilon_t$$

Środowisko R dostarcza wiele funkcji za pomocą których możemy estymować modele typu $ARIMA$. Przykładowo komenda `ar(stats)` daje możliwość estymacji modeli autoregresyjnych $AR(p)$. Opcja `method` oferuje kilka sposobów szacowania parametrów modelu: `"burg"`, `"ols"`, `"mle"`, `"yule-walker"`, `"yw"`. Z kolei funkcja `arima(stats)` służy do estymacji modeli $ARIMA$ lub $SARIMA$. Warto także zwrócić uwagę na pakiet `forecast` który dostarcza szereg funkcji do analizy szeregów czasowych.

7.3.1 Estymacja

Wykorzystując funkcje `auto.arima(forecast)` proces estymacji modelu $ARIMA$ przebiega w sposób całkowicie automatyczny. Spośród wielu estymowanych modeli zostaje wybrany ten, który charakteryzuje się najmniejszą wartością kryterium informacyjnego AIC.

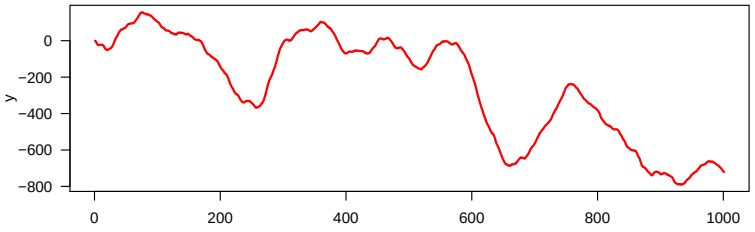
```
> m=auto.arima(b)
> summary(m)
Series: b
ARIMA(1,1,0)(1,0,0)[12]

Call: auto.arima(x = b)

Coefficients:
         ar1    sar1
      0.5580  0.7313
s.e.  0.0912  0.0834

sigma^2 estimated as 0.04792:  log likelihood = 3.24
AIC = -0.48   AICc = -0.16   BIC = 6.67

In-sample error measures:
          ME         RMSE          MAE          MPE         MAPE
-0.007606623  0.217557172  0.137980842  0.021537486  1.097890864
        MASE
 0.450549687
```

Oczywiście można także samemu wyznaczać liczbę parametrów p, d, q, P, D, Q.

```
> m=arima(b,order=c(1,1,0),seasonal=list(order=c(1,0,0),period=12))
> summary(m)
Series: b
ARIMA(1,1,0)(1,0,0)[12]

Call: arima(x = b, order = c(1, 1, 0), seasonal = list(order = c(1, 0, 0),
    period = 12))

Coefficients:
         ar1    sar1
      0.5580  0.7313
s.e.  0.0912  0.0834

sigma^2 estimated as 0.04792:  log likelihood = 3.24
AIC = -0.48   AICc = -0.16   BIC = 6.67

In-sample error measures:
          ME         RMSE          MAE          MPE         MAPE
-0.007606623  0.217557172  0.137980842  0.021537486  1.097890864
        MASE
 0.450549687
```

Warto również zaznaczyć, że w programie R mamy możliwość symulowania procesów autoregresji.

```
> y=arima.sim( n=1000,
        innov=rnorm(1000,0,2), # składnik losowy ma rozkład N(0,2)
        model=list(
        order = c(2,1,1),  # ilość parametrów
        ar = c(0.3, 0.6),  # wartości parametrów p
        ma = c( 0.2),      # wartości parametrów q
        sd = sqrt(0.1)))
> y=ts(y)
```



Rysunek 7.12: Prezentacja graficzna symulowanego procesu.

```
> s=arima(y,order=c(2,1,1));s
Series: y
ARIMA(2,1,1)

Call: arima(x = y, order = c(2, 1, 1))

Coefficients:
         ar1     ar2     ma1
      0.3596  0.5568  0.1741
s.e.  0.0710  0.0628  0.0878

sigma^2 estimated as 4.57:  log likelihood = -2179.67
AIC = 4367.34   AICc = 4367.38   BIC = 4386.97
```

Dzięki zastosowaniu funkcji `ndiffs(forecast)` mamy możliwość sprawdzenia czy rzeczywiście proces `y` jest zintegrowany w stopniu pierwszym tzn. $I(1)$. Za pomocą opcji `test` możemy wybrać procedurę z jakiej chcemy skorzystać: `pp` — test Phillipsa-Perrona,`adf` — test Dickeya-Fullera lub `kpss` — test Kwiatkowski-Phillips-Schmidt-Shin. Poziom istotności `alpha` także możemy zmieniać.

```
> ndiffs(y,test="pp",alpha=0.05)
[1] 1
```

Wynikiem funkcji `ndiffs(forecast)` zawsze jest stopień zintegrowania badanego procesu. Zatem należy stwierdzić, że w zmiennej `y` występuje pierwiastek jednostkowy $d = 1$.

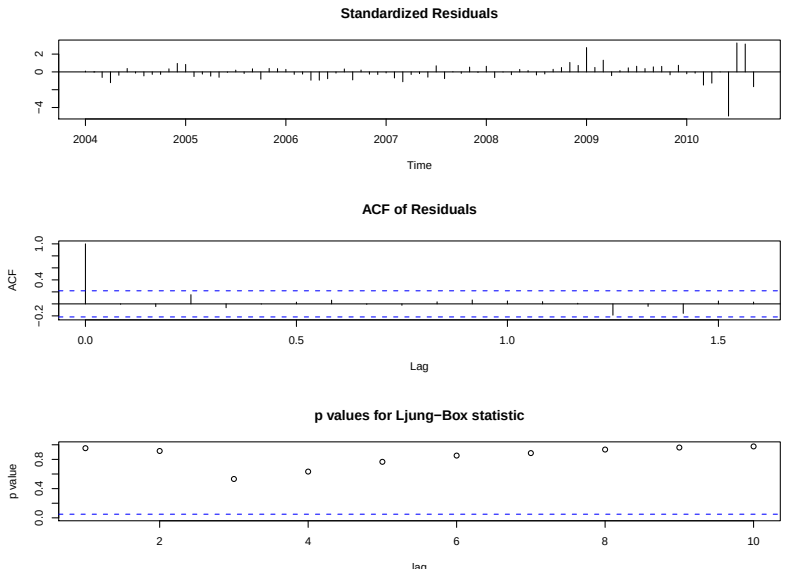
7.3.2 Weryfikacja

Etap weryfikacji oszacowanego modelu ARIMA polega na sprawdzeniu hipotezy zerowej o braku zjawiska autokorelacji w procesie reszt. Do tego celu możemy wykorzystać test Ljunga-Boxa.

```
# wyznaczenie p-value dla testu na autokorelację reszt Ljunga-Boxa:
> r=resid(m)
> k <- 10  # maksymalny stopień autokorelacji
> p <- rep(NA, k)
  for (i in 1:k) {
    p[i] <- Box.test(r, i, type = "Ljung-Box")$p.value
  }
> p=round(p, 8) # p-value
> p
 [1] 0.9535022 0.9152974 0.5318782 0.6333194 0.7667281 0.8522917 0.8863058
 [8] 0.9348725 0.9625130 0.9775571
```

Tak więc na podstawie testu Ljunga-Boxa brak jest podstaw do odrzucenia hipotezy zerowej zakładającej brak autokorelacji. Wniosek ten potwierdzają również wykresy wykonane za pomocą komendy `tsdiag(stats)` (rys. 7.13).

```
> tsdiag(m)
```



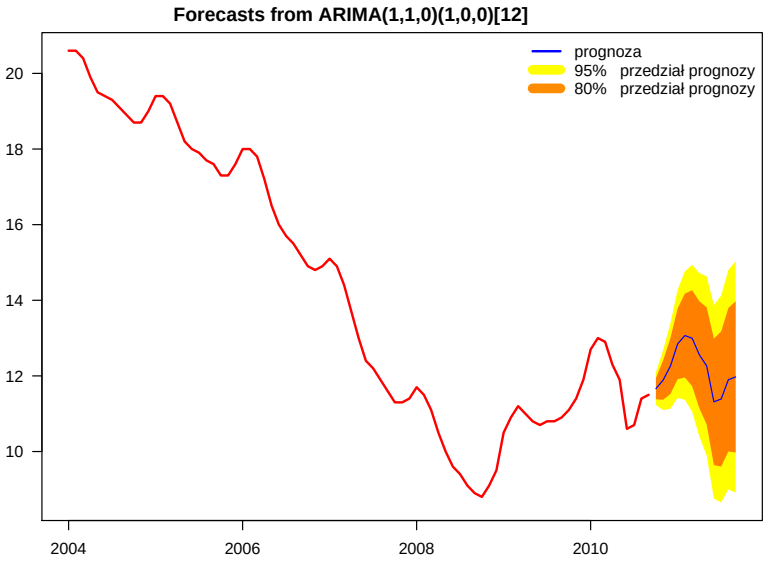
Rysunek 7.13: Diagnostyka reszt modelu.

7.3.3 Prognozowanie

Funkcja `forecast(forecast)` służy do obliczania prognoz na podstawie danego modelu. W naszym przykładzie liczbę okresów do prognozowania została ustalona na 12 miesięcy. Natomiast przedziały predykcji zostały ustalone na poziomie 80% i 95% — są to wartości domyślne. Gdybyśmy chcieli je zmienić na 99% wystarczy użyć polecenia `level=99`.

```
> forecast(m,h=12)
         Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
Oct 2010       11.66125 11.380716 11.94178 11.232211 12.09029
Nov 2010       11.88899 11.369636 12.40835 11.094703 12.68329
Dec 2010       12.25929 11.521215 12.99738 11.130499 13.38809
Jan 2011       12.84691 11.912406 13.78141 11.417711 14.27611
Feb 2011       13.06774 11.956998 14.17848 11.369006 14.76648
Mar 2011       12.99543 11.725520 14.26533 11.053272 14.93758
Apr 2011       12.55712 11.142176 13.97207 10.393148 14.72110
May 2011       12.26487 10.716515 13.81323  9.896864 14.63288
Jun 2011       11.31437  9.642230 12.98652  8.757050 13.87170
Jul 2011       11.38758  9.599679 13.17548  8.653222 14.12194
Aug 2011       11.89950 10.002628 13.79638  8.998482 14.80053
Sep 2011       11.97266  9.972585 13.97273  8.913812 15.03150
```

```
> par(mar=c(3,4,2,1),cex=.8)
> plot(forecast(m,h=12),las=1,col="red",lwd=2)
> legend('topright',c('prognoza',
'95%   przedział prognozy','80%   przedział prognozy'),
col=c('blue','yellow','darkorange'),lwd=c(2,8,8),bty="n")
```



Rysunek 7.14: Prognoza stopy bezrobocia od 10.2010–09.2011.

Dodatkowo możemy wyznaczyć błędy prognoz za pomocą funkcji `predict(stats)`:

```
> p=predict(m,n.ahead=12)
> ts( cbind(pred=p$pred, se=p$se, error=p$se/p$pred),
start=c(2010,10),freq=12)
             pred        se      error
Oct 2010 11.66125 0.2189006 0.01877162
Nov 2010 11.88899 0.4052582 0.03408683
Dec 2010 12.25929 0.5759270 0.04697881
Jan 2011 12.84691 0.7291963 0.05676045
Feb 2011 13.06774 0.8667176 0.06632497
```

```
Mar 2011 12.99543 0.9909135 0.07625094
Apr 2011 12.55712 1.1040891 0.08792532
May 2011 12.26487 1.2081893 0.09850810
Jun 2011 11.31437 1.3047811 0.11532066
Jul 2011 11.38758 1.3951054 0.12251115
Aug 2011 11.89950 1.4801411 0.12438678
Sep 2011 11.97266 1.5606634 0.13035232
```

Otrzymane wartości prognostyczne wskazują, iż należy się spodziewać wzrostu stopy bezrobocia od października 2010 — $11,66\%$ do lutego 2011 — $13,07\%$. Od marca należy się spodziewać spadku tego wskaźnika do poziomu $11,31\%$ w czerwcu 2011 roku. Następnie będzie on znów powoli wzrastał i w miesiącu wrześniu osiągnie wartość $11,97\%$. Należy także zwrócic uwagę na oszacowane błędy otrzymanych prognoz. Dla października 2010 wyniósł on $1,9\%$ i cały czas wzrastał, aż do poziomu $13,04\%$ dla września 2011 roku.

7.4 Modele adaptacyjne

7.4.1 Estymacja

Podobnie jak w przypadku modelu $ARIMA$ także estymacja modelu adaptacyjnego może przebiegać w sposób całkowicie automatyczny. Wykorzystamy do tego celu funkcje `ets(forecast)`.

```
> n=ets(b,model="ZZZ",damped=NULL)
> summary(n)
ETS(A,Ad,A)

Call:
 ets(y = b)

  Smoothing parameters:
    alpha = 0.9631
    beta  = 0.1318
    gamma = 0.0368
    phi   = 0.9747

  Initial states:
    l = 19.9195
    b = -0.0852
    s = 0.1033 -0.2757 -0.482 -0.3991 -0.3688 -0.4184
           -0.4601 -0.1605 0.275 0.6922 0.8065 0.6876

  sigma:  0.1994

     AIC     AICc      BIC
128.7737 138.4880 169.4794

In-sample error measures:
```

```
           ME         RMSE          MAE          MPE         MAPE
-0.006008925  0.199448035  0.123892225  0.032931026  0.996058963
        MASE
 0.404546040
```

```
> logLik(n)
'log Lik.' -47.38687 (df=17)  # logarytm wiarygodności
```

Spośród wszystkich modeli, które były estymowane — `model="ZZZ"` z wykorzystaniem optymalizacji logarytmu wiarygodności — `opt.crit="lik"`, został wybrany model o najmniejszym kryterium informacyjnym AIC — `ic="aic"`. Dostępne są także inne metody optymalizacyjne do szacowania parametrów: `mse`, `amse`, `nmse`, `sigma`. Także wybór kryterium informacyjnego (za pomocą którego dokonujemy wyboru najlepszego modelu) możemy wybrać samodzielnie: `aic`, `aicc`, `bic`. Tak przeprowadzona estymacja doprowadziła do otrzymania modelu `"AAdA"`, który charakteryzuje się następującymi cechami: `A` — addytywny składnik losowy (pierwsza litera), `Ad` — trend addytywny (druga litera), gdzie mała litera `d` oznacza, że została wybrana również opcja `damped=T` czyli „przygaszenie" trendu, `A` — addytywne wachania sezonowe (trzecia litera). Zatem otrzymaliśmy model Wintersa z addytywnymi wahaniami periodycznymi:

```
> n=ets(b, model="AAA", damped=T)
> n
ETS(A,Ad,A)

Call:
 ets(y = b, model = "AAA", damped = T)

  Smoothing parameters:
    alpha = 0.9631
    beta  = 0.1318
    gamma = 0.0368
    phi   = 0.9747

  Initial states:
    l = 19.9195
    b = -0.0852
    s = 0.1033 -0.2757 -0.482 -0.3991 -0.3688 -0.4184
           -0.4601 -0.1605 0.275 0.6922 0.8065 0.6876

  sigma:  0.1994

     AIC     AICc      BIC
128.7737 138.4880 169.4794
```

Oczywiście korzystając z funkcji `ets` oraz opcji `model` możemy od razu zawęzić grupę modeli, spośród których zostanie wybrany najlepszy z nich: modele Browna — `model="ZNN"`, modele Browna lub Holta — `model="ZZN"`, modele Browna, Holta lub Wintersa — `model="ZZZ"`. Jest również możliwość estymacji modeli o ustalonych parametrach: `alpha`, `beta`, `gamma`, `phi`. Natomiast gdy ich wartości zawierają się

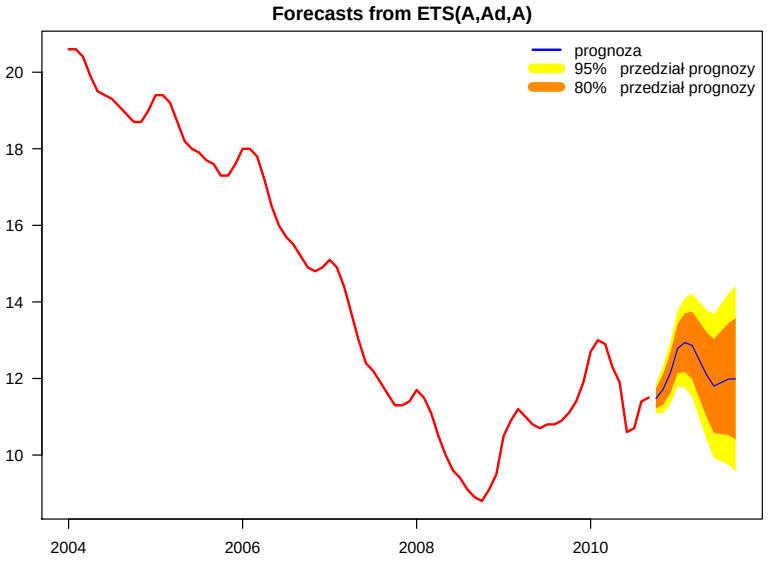
w pewnych przedziałach liczbowych, przykładowo jeśli $\alpha = (0,11 - 0,53)$, $\beta = (0,41 - 0,68)$, $\gamma = (0,65 - 0,78)$ to należy użyć opcji `level=c(0.11,0.41,0.65)` oraz `upper=c(0.53,0.68,0.78)`.

7.4.2 Prognozowanie

Do procesu prognozowania z wykorzystaniem modelu adaptacyjnego możemy również wykorzystać funkcję `forecast(forecast)`.

```
> forecast(n,h=12)
         Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
Oct 2010       11.47485 11.21925 11.73045 11.083940 11.86576
Nov 2010       11.71822 11.31436 12.12208 11.100564 12.33587
Dec 2010       12.14519 11.61424 12.67614 11.333168 12.95721
Jan 2011       12.78291 12.13239 13.43344 11.788021 13.77780
Feb 2011       12.93763 12.17068 13.70459 11.764672 14.11060
Mar 2011       12.86345 11.98128 13.74561 11.514292 14.21261
Apr 2011       12.47567 11.47856 13.47277 10.950730 14.00060
May 2011       12.09394 10.98166 13.20622 10.392858 13.79502
Jun 2011       11.80244 10.57448 13.03041  9.924430 13.68045
Jul 2011       11.89412 10.54981 13.23844  9.838176 13.95007
Aug 2011       11.98331 10.52193 13.44469  9.748317 14.21830
Sep 2011       11.98932 10.40653 13.57211  9.568652 14.40998
```

```
> par(mar=c(3,4,2,1),cex=.8)
> plot(forecast(n,h=12),las=1,col="red",lwd=2)
> legend('topright',c('prognoza',
'95%   przedział prognozy','80%   przedział prognozy'),
col=c('blue','yellow','darkorange'),lwd=c(2,8,8),bty="n")
```



Rysunek 7.15: Prognoza stopy bezrobocia od 10.2010–09.2011.

W przypadku gdy wykonaliśmy dwie prognozy za pomocą różnych modeli (np. modelu ARIMA i adaptacyjnego) warto skorzystać z testu Diebold-Mariano. Hipo-

teza zerowa tego tesu zakłada, że dokładność predykcyjna dwóch modeli jest jednakowa. Założenie `alt="two.sided"` (test dwustronny) jest opcją domyślną, możemy ją zmienić dodając polecenie `alt="g"` (test prawostronny) lub `alt="l"` (test lewostronny).

```
> dm.test(resid(m),resid(n))

        Diebold-Mariano Test

data:  resid(m) resid(n)
DM = 1.0885, Forecast horizon = 1, Loss function power = 2, p-value =
0.2764
alternative hypothesis: two.sided
```

Rozdział 8

Przykład analizy tabel wielodzielczych

8.1 Wprowadzenie

Zbiór danych, który zostanie wykorzystany do przeprowadzenia analiz jest dostępny w paczce `Titanic(datasets)`. Zawiera on informacje na temat losu 2201 pasażerów oceanicznego liniowca „Titanic". Do dyspozycji mamy cztery zmienne nominalne:

- **Class** — klasy podróżowania: 1st (pierwsza), 2nd (druga), 3rd (trzecia), Crew (załoga),
- **Sex** — płeć: Male (mężczyzna), Female (kobieta),
- **Age** — wiek: Adult (dorosły), Child (dziecko),
- **Survived** — przeżycie: No (nie), Yes (tak).

Oryginalne dane zostały zapisane w formie tabeli o nazwie `t`. Fragment naszych danych jest widoczny poniżej:

```
> t[1:5,]
  Class  Sex   Age Survived
1   3rd Male Child       No
2   3rd Male Child       No
3   3rd Male Child       No
4   3rd Male Child       No
5   3rd Male Child       No
```

Celem naszych badań będzie próba opisu zależności między przeżyciem, a klasą podróżowania, płcią podróżnych oraz ich wiekiem.

8.2 Test chi-kwadrat

Do analizy zależności między dwiema zmiennymi zostanie wykorzystamy test χ^2. W pakiecie R jest dostępna funkcja `assocstats(vcd)` za pomocą której wykonywane

są dwa testy badające niezależność: test χ^2 oraz test G:

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(O_{ij}-E_{ij})^2}{E_{ij}}$$

$$G = 2\cdot\sum_{i=1}^{r}\sum_{j=1}^{c}O_{ij}\cdot\ln\left(\frac{O_{ij}}{E_{ij}}\right)$$

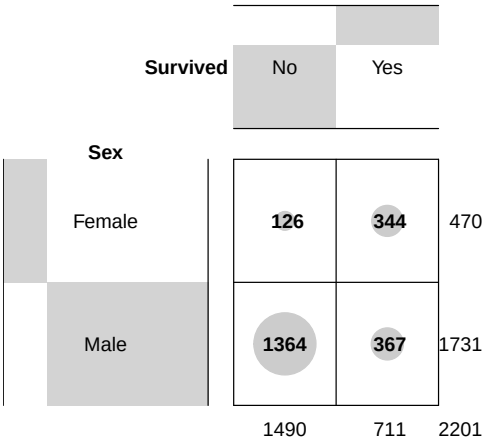
gdzie:

- r — liczba wierszy,
- c — liczba kolumn,
- O_{ij} — empiryczna liczebność i-tego wiersza oraz j-tej kolumny,
- E_{ij} — oczekiwana liczebność i-tego wiersza oraz j-tej kolumny.

Dodatkowo podawane są również współczynniki korelacji: ϕ-Yula, V-Pearsona i V-Cramera. A zatem spróbujmy odpowiedzieć na następujące pytania:

1. Czy na przeżycie katastrofy miała wpływ płeć?

```
> s=xtabs(~Survived+Sex)  # tablica kontygencji
> library(gplots)         # załadowanie paczki gplots
> balloonplot(s,dotcolor="gray80")
```



Rysunek 8.1: Przeżycie i płeć.

```
> library(vcd)            # załadowanie paczki vcd
> assocstats(s)
                    X^2 df P(> X^2)
Likelihood Ratio 434.47  1        0
Pearson          456.87  1        0

Phi-Coefficient   : 0.456
Contingency Coeff.: 0.415
Cramer's V        : 0.456
```

Na podstawie testu χ^2 oraz wykresu (rys. 8.1) możemy wnioskować, że występuje istotne powiązanie między przeżyciem katastrofy, a płcią pasażerów (p-value= 0, $V = 0,456$).

```
> ps=prop.table(s,2);ps
        Sex
Survived    Female      Male
     No  0.2680851 0.7879838
     Yes 0.7319149 0.2120162
```

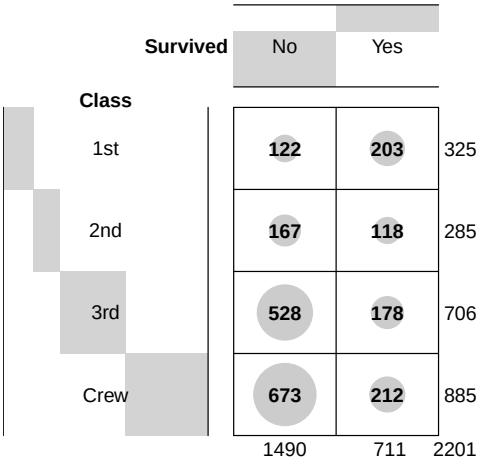
Obliczenia procentowe wskazują, że przeżyło 73% kobiet oraz tylko 21% mężczyzn.

```
> ns=ps/(1-ps);ns  # szanse przeżycia
        Sex
Survived    Female      Male
     No  0.3662791 3.7166213
     Yes 2.7301587 0.2690616
> ns[,1]/ns[,2]    # ilorazy szans przeżycia
         No         Yes
 0.09855163 10.14696596
```

Obliczony iloraz szans $OR = 10,15$ wskazuje, że szansa przeżycia katastrofy na Titanicu wśród kobiet $(2,7301587)$ była ponad dziesięć razy większa niż wśród mężczyzn $(0,2690616)$.

2. Czy na przeżycie katastrofy miała wpływ klasa w której podróżowano?

```
> c=xtabs(~Survived+Class)  # tablica kontygencji
> balloonplot(c,dotcolor="gray80")
```



Rysunek 8.2: Przeżycie i klasa podróżowania.

```
> assocstats(c)
                    X^2 df P(> X^2)
Likelihood Ratio 180.9  3        0
Pearson          190.4  3        0

Phi-Coefficient   : 0.294
Contingency Coeff.: 0.282
Cramer's V        : 0.294
```

Ponieważ p-value = 0 to należy sądzić, że istnieje zależność między przeżyciem a klasą w której podróżowali uczestnicy rejsu. O wysokim powiązaniu tych zmiennych świdczy także wysoki współczynnik korelacji $V = 0,29$.

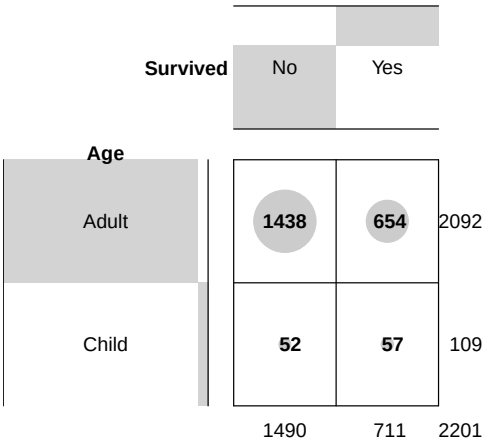
```
> pc=prop.table(c,2);pc
        Class
Survived       1st       2nd       3rd      Crew
     No  0.3753846 0.5859649 0.7478754 0.7604520
     Yes 0.6246154 0.4140351 0.2521246 0.2395480
```

Po obliczeniu wartości procentowych dochodzimy do wniosku, że najwięcej przeżyło tych osób, które podróżowały w pierwszej klasie — 62%. Natomiast wśród załogi przeżyło najmniej osób, tylko 24%. Należy także podkreślić, że im wyższy standard podróży tym większa szansa przeżycia.

```
> nc=pc/(1-pc);nc  # szanse przeżycia
        Class
Survived       1st       2nd       3rd      Crew
     No  0.6009852 1.4152542 2.9662921 3.1745283
     Yes 1.6639344 0.7065868 0.3371212 0.3150074
> nc[,1]/nc[,2]    # porównanie szans 1 z 2 klasą
       No       Yes
0.4246482 2.3548902
> nc[,2]/nc[,3]    # porównanie szans 2 z 3 klasą
       No       Yes
0.4771122 2.0959429
> nc[,3]/nc[,4]    # porównanie szans 3 z załogą
      No      Yes
0.934404 1.070201
```

3. Czy na przeżycie katastrofy miał wpływ wiek?

```
> a=xtabs(~Survived+Age)  # tablica kontygencji
> balloonplot(a,dotcolor="gray80")
```



Rysunek 8.3: Przeżycie i wiek.

```
> assocstats(a)
                    X^2 df   P(> X^2)
Likelihood Ratio 19.561  1 9.7458e-06
Pearson          20.956  1 4.7008e-06

Phi-Coefficient   : 0.098
Contingency Coeff.: 0.097
Cramer's V        : 0.098
```

Także i w tym przypadku istnieje zależność między zmiennymi przeżycie oraz wiek, choć współczynnik korelacji nie jest zbyt wysoki i wynosi $V = 0,098$.

```
> pa=prop.table(a,2);pa
        Age
Survived     Adult     Child
     No  0.6873805 0.4770642
     Yes 0.3126195 0.5229358
```

Po obliczeniu wartości procentowych należy stwierdzić, że katastrofę przeżyła ponad połowa dzieci (52%) oraz $\frac{1}{3}$ osób dorosłych (31%).

8.3 Model logitowy

Model logitowy daje dużo większe możliwości analizy tabel wielodzielczych niż test χ^2. Omawiane w poprzenim rozdziale testy niezależności są przeznaczone do badania zależności dwóch zmiennych jakościowych. Natomiast za pomocą modelu logitowego mamy możliwość analizy wielowymiarowych tabel kontyngencji. Jdnak warunek jaki musi zostać spełniony jest taki, aby zmienna zależna była zmienną binarną. W programie R dostępna jest funkcja `glm(stats)` za pomocą której możemy estymować parametry modelu logitowego.

```
# punkt odniesienia: Male
> tt=transform(t,Sex=relevel(Sex,ref="Male"))
```

```
> logit1=glm(Survived~.,tt,family=binomial)
> summary(logit1)

Call:
glm(formula = Survived ~ ., family = binomial, data = tt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0812  -0.7149  -0.6656   0.6858   2.1278

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.3762     0.1362  -2.763  0.00573 **
Class2nd     -1.0181     0.1960  -5.194 2.05e-07 ***
Class3rd     -1.7778     0.1716 -10.362  < 2e-16 ***
ClassCrew    -0.8577     0.1573  -5.451 5.00e-08 ***
SexFemale     2.4201     0.1404  17.236  < 2e-16 ***
```

```
AgeChild      1.0615     0.2440   4.350 1.36e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2769.5  on 2200  degrees of freedom
Residual deviance: 2210.1  on 2195  degrees of freedom
AIC: 2222.1

Number of Fisher Scoring iterations: 4
```

Ten model jest dobrze dopasowany do danych. Potwierdzają to poniższe testy:

```
# test logarytmu wiarygodności:
> library(lmtest)
> lrtest(logit1)
Likelihood ratio test

Model 1: Survived ~ Class + Sex + Age
Model 2: Survived ~ 1
   Df  LogLik Df Chisq Pr(>Chisq)
1   6 -1105.0
2   1 -1384.7 -5 559.4  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# test reszt deviance:
> anova(logit1,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: Survived

Terms added sequentially (first to last)


      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                  2200     2769.5
Class  3   180.90      2197     2588.6 < 2.2e-16 ***
Sex    1   359.64      2196     2228.9 < 2.2e-16 ***
Age    1    18.85      2195     2210.1 1.413e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

# adekwatność modelu:
> 1-pchisq(deviance(logit1),logit1$df.residual)
[1] 0.4063859
# dyspersja:
> sum(resid(logit1,type="pearson")^2)/df.residual(logit1)
```

```
[1] 1.023531
```

W kolejnym kroku obliczymy ilorazy wiarygodności na podstawie oszacowanych parametrów modelu.

```
> cbind(OR=exp(coef(logit1)))
                    OR
(Intercept)  0.6864493
Class2nd     0.3612825
Class3rd     0.1690159
ClassCrew    0.4241466
SexFemale   11.2465380
AgeChild     2.8908263
```

Interpretacja jest następująca:

- podróżowanie w drugiej klasie dało o 64% mniej szans na przeżycie niż w pierwszej klasie,
- podróżowanie w trzeciej klasie dało o 83% mniej szans na przeżycie niż w pierwszej klasie,
- załoga statku miała o 58% mniej szans na przeżycie niż w pierwszej klasie,
- kobiety miały ponad 11 razy większe szanse na przeżycie od mężczyzn,
- dzieci miały prawie $2,9$ razy większe szanse na przeżycie od dorosłych.

Uogólnione modele liniowe dają także możliwość badania bardziej złożonych zależności między zmiennymi tzn. interakcji.

```
> logit2=glm(Survived~.^2,tt,family=binomial)  # interakcje rzędu 2.
> logit3=glm(Survived~.^3,tt,family=binomial)  # interakcje rzędu 2 i 3.
```

Po estymacji kilku modeli, zawsze jesteśmy zainteresowani, który z nich wybrać. Takiego wyboru możemy dokonać np. na podstawie kryterium informacyjnego AIC.

```
> AIC(logit1,logit2,logit3)
       df      AIC
logit1  6 2222.061
logit2 12 2121.495
logit3 14 2125.495
```

Zatem do dalszych analiz powinniśmy wybrać model `logit2` (z podwójnymi interakcjami) ponieważ charakteryzuje się najmiejszą wartością AIC. Naszą decyzję potwierdzają także inne statystyki np. badające adekwatność modelu:

```
> 1-pchisq(deviance(logit1),logit1$df.residual)
[1] 0.4063859
> 1-pchisq(deviance(logit2),logit2$df.residual)
[1] 0.9181299
> 1-pchisq(deviance(logit3),logit3$df.residual)
[1] 0.913425
```

oraz oceniające istotność poszczególnych interakcji:

```
> anova(logit3,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: Survived

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                         2200     2769.5
Class         3   180.90      2197     2588.6 < 2.2e-16 ***
Sex           1   359.64      2196     2228.9 < 2.2e-16 ***
Age           1    18.85      2195     2210.1 1.413e-05 ***
Class:Sex     3    66.67      2192     2143.4 2.206e-14 ***
Class:Age     2    44.21      2190     2099.2 2.507e-10 ***
Sex:Age       1     1.69      2189     2097.5    0.1942
Class:Sex:Age 2     0.00      2187     2097.5    1.0000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Tak więc najlepszym modelem okazał się `logit2`.

```
> summary(logit2)

Call:
glm(formula = Survived ~ .^2, family = binomial, data = tt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6771  -0.7099  -0.5952   0.2374   2.2293

Coefficients: (1 not defined because of singularities)
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)          -0.72763    0.16130  -4.511 6.45e-06 ***
Class2nd             -1.67026    0.32240  -5.181 2.21e-07 ***
Class3rd             -0.91330    0.20478  -4.460 8.20e-06 ***
ClassCrew            -0.52215    0.18088  -2.887  0.00389 **
SexFemale             4.28298    0.53213   8.049 8.36e-16 ***
AgeChild             16.99213  920.38639   0.018  0.98527
Class2nd:SexFemale   -0.06801    0.67120  -0.101  0.91929
Class3rd:SexFemale   -2.79995    0.56875  -4.923 8.52e-07 ***
ClassCrew:SexFemale  -1.13608    0.82048  -1.385  0.16616
Class2nd:AgeChild     0.84881 1005.81951   0.001  0.99933
Class3rd:AgeChild   -16.34159  920.38646  -0.018  0.98583
ClassCrew:AgeChild         NA         NA      NA       NA
SexFemale:AgeChild   -0.68679    0.52541  -1.307  0.19116
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2769.5  on 2200  degrees of freedom
Residual deviance: 2097.5  on 2189  degrees of freedom
AIC: 2121.5

Number of Fisher Scoring iterations: 15
```

Teraz obliczymy ilorazy wiarygodności aby je zinterpretować. Wyniki zostały zaokrąglone do dwóch miejsc po przecinku za pomocą funkcji `round(base)`.

```
> cbind(OR=round(exp(coef(logit2)),2))
                             OR
(Intercept)                0.48
Class2nd                   0.19
Class3rd                   0.40
ClassCrew                  0.59
SexFemale                 72.46
AgeChild            23965569.60
Class2nd:SexFemale         0.93
Class3rd:SexFemale         0.06
ClassCrew:SexFemale        0.32
Class2nd:AgeChild          2.34
Class3rd:AgeChild          0.00
ClassCrew:AgeChild           NA
SexFemale:AgeChild         0.50
```

- szansa przeżycia osób, które podróżowały w drugiej klasie była o 81% mniejsza niż w pierwszej klasie,
- szansa przeżycia osób, które podróżowały w trzeciej klasie była o 60% mniejsza niż w pierwszej klasie,
- szansa przeżycia załogi statku była o 41% mniejsza niż w pierwszej klasie,
- kobiety miały ponad 72 razy większe szanse na przeżycie od mężczyzn,
- dzieci miały również dużo większe szanse na przeżycie od dorosłych,
- szansa przeżycia kobiet, które podróżowały w trzeciej klasie była mniejsza o 94% niż w pierwszej klasie,
- szansa przeżycia dzieci, które podróżowały w drugiej klasie była ponad 2 razy większa niż w pierwszej klasie.

Funkcja `glm` umożliwia także wybór odpowiednich zmiennych do modelu. Jeśli np. chcielibyśmy oszacować model `logit2` ale tylko z istotnymi statystycznie parametrami to kod wyglądałby następująco:

```
> myl=glm(Survived~Class+Sex+Age+I(Class=="3rd"&Sex=="Female"),
  tt,family=binomial)
> summary(myl)
```

```
Call:
glm(formula = Survived ~ Class + Sex + Age + I(Class == "3rd" &
    Sex == "Female"), family = binomial, data = tt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5435  -0.7070  -0.5798   0.3842   2.0293

Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                        -0.6337     0.1506  -4.208 2.57e-05 ***
Class2nd                           -1.2888     0.2385  -5.404 6.52e-08 ***
Class3rd                           -1.0642     0.1935  -5.500 3.80e-08 ***
ClassCrew                          -0.6252     0.1708  -3.661 0.000252 ***
SexFemale                           3.8283     0.2715  14.098  < 2e-16 ***
AgeChild                            1.0522     0.2306   4.563 5.05e-06 ***
I(Class=="3rd"&Sex=="Female")TRUE  -2.4578     0.3302  -7.443 9.81e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2769.5  on 2200  degrees of freedom
Residual deviance: 2145.1  on 2194  degrees of freedom
AIC: 2159.1

Number of Fisher Scoring iterations: 5
```

8.4 Model Poissona

Innym sposobem analizy wielowymiarowych tabel kontyngencji jest zastosowanie modelu Poissona. Procedura estymacji i diagnostyki przebiega w taki sam sposób jak w przypadku modelu logitowego. Należy jednak pamiętać o tym, że w tego typu modelach zmienna zależna ma rozkład Poissona. A zatem nasz zbiór danych należy przekształcić.

```
> f=as.data.frame(xtabs(~Survived+Class+Sex+Age,tt))
```

```
> f[1:3,]
  Survived Class  Sex   Age Freq
1       No   1st Male Adult  118
2      Yes   1st Male Adult   57
3       No   2nd Male Adult  154
```

```
> p1=glm(Freq~.,f,family=poisson)
> p2=glm(Freq~.^2,f,family=poisson)
> p3=glm(Freq~.^3,f,family=poisson)
> p4=glm(Freq~.^4,f,family=poisson)
```

```
# kryteria informacyjne:
> AIC(p1,p2,p3,p4)
   df       AIC
p1  7 1385.0654
p2 19  281.9902
p3 29  185.4021
p4 32  191.4021
# adekwatność modelu:
> 1-pchisq(deviance(p1),p1$df.residual)
[1] 0
> 1-pchisq(deviance(p2),p2$df.residual)
[1] 0
> 1-pchisq(deviance(p3),p3$df.residual)
[1] 1
> 1-pchisq(deviance(p4),p4$df.residual)
[1] 0
# istotność poszczególnych interakcji:
> anova(p4,test="Chisq")
Analysis of Deviance Table

Model: poisson, link: log

Response: Freq

Terms added sequentially (first to last)

                     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                                    31     4953.1
Survived              1   281.78        30     4671.4 < 2.2e-16 ***
Class                 3   475.81        27     4195.5 < 2.2e-16 ***
Sex                   1   768.32        26     3427.2 < 2.2e-16 ***
Age                   1  2183.56        25     1243.7 < 2.2e-16 ***
Survived:Class        3   180.90        22     1062.8 < 2.2e-16 ***
Survived:Sex          1   434.47        21      628.3 < 2.2e-16 ***
Survived:Age          1    19.56        20      608.7 9.746e-06 ***
Class:Sex             3   337.77        17      271.0 < 2.2e-16 ***
Class:Age             3   154.35        14      116.6 < 2.2e-16 ***
Sex:Age               1     0.02        13      116.6 0.8882240
Survived:Class:Sex    3    65.30        10       51.3 4.336e-14 ***
Survived:Class:Age    3    29.34         7       22.0 1.900e-06 ***
Survived:Sex:Age      1    12.17         6        9.8 0.0004857 ***
Class:Sex:Age         3     9.78         3        0.0 0.0205004 *
Survived:Class:Sex:Age  3     0.00         0        0.0 1.0000000
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
# ilorazy wiarygodności:
> cbind(OR=round(exp(coef(p3)),1))
                                  OR
(Intercept)                1.180000e+02
```

```
SurvivedYes                       5.000000e-01
Class2nd                          1.300000e+00
Class3rd                          3.300000e+00
ClassCrew                         5.700000e+00
SexFemale                         0.000000e+00
AgeChild                          0.000000e+00
SurvivedYes:Class2nd              2.000000e-01
SurvivedYes:Class3rd              4.000000e-01
SurvivedYes:ClassCrew             6.000000e-01
SurvivedYes:SexFemale             7.250000e+01
SurvivedYes:AgeChild              9.354350e+10
Class2nd:SexFemale                2.500000e+00
Class3rd:SexFemale                6.800000e+00
ClassCrew:SexFemale               1.000000e-01
Class2nd:AgeChild                 4.000000e-01
Class3rd:AgeChild                 9.644408e+10
ClassCrew:AgeChild                0.000000e+00
SexFemale:AgeChild                2.000000e-01
SurvivedYes:Class2nd:SexFemale    9.000000e-01
SurvivedYes:Class3rd:SexFemale    1.000000e-01
SurvivedYes:ClassCrew:SexFemale   3.000000e-01
SurvivedYes:Class2nd:AgeChild     2.550000e+01
SurvivedYes:Class3rd:AgeChild     0.000000e+00
SurvivedYes:ClassCrew:AgeChild    0.000000e+00
SurvivedYes:SexFemale:AgeChild    5.000000e-01
Class2nd:SexFemale:AgeChild       2.500000e+00
Class3rd:SexFemale:AgeChild       1.310000e+01
ClassCrew:SexFemale:AgeChild      4.034000e+02
```

Na podstawie otrzymanych ilorazów wiarygodności możemy również sformułować kilka wniosków:

- szanse zgonu w poszczególnych klasach były większe niż w pierwszej klasie o: 30% w klasie drugiej, ponad 3 razy większe w klasie trzeciej oraz prawie sześciokrotnie większe wśród załogi.

- szanse przeżycia w poszczególnych klasach były mniejsze niż w pierwszej klasie o: 80% w klasie drugiej, 60% w klasie trzeciej i 40% wśród załogi. Z kolei szansa przeżycia kobiet była ponad 72 razy większa niż mężczyzn, także dzieci miały wielokrotnie większe szanse na przeżycie niż dorośli.

- szanse przeżycia w poszczególnych klasach wśród kobiet były mniejsze niż dla kobiet podróżujących w pierwszej klasie o: 10% w klasie drugiej, 90% w klasie trzeciej i 70% wśród żeńskiej załogi statku.

W środowisku R dostępne są także inne funkcje za pomocą których możemy analizować wielowymiarowe tabele wielodzielcze np. `loglm(MASS)` lub `loglin(stats)`.

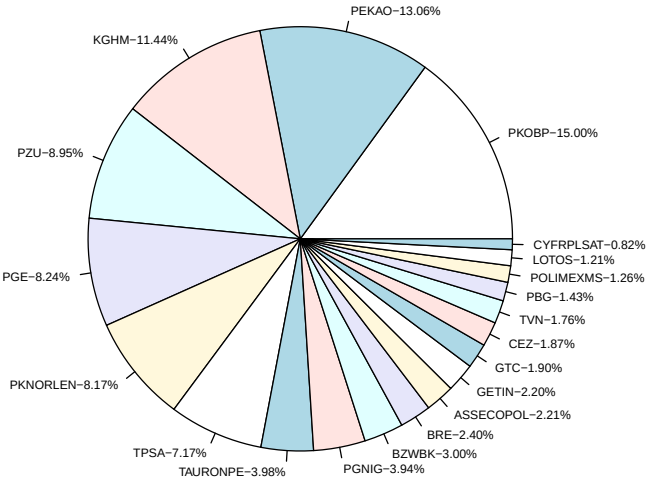
Rozdział 9

Przykład badania rozkładu stopy zwrotu z akcji

9.1 Wprowadzenie

W tym opracowaniu będziemy analizować dzienne stopy zwrotu z akcji spółki PKOBP, która wchodzi w skład indeksu giełdowego WIG20. Indeks WIG20 to portfel akcji dwudziestu największych i najbardziej płynnych spółek notowanych na GPW w Warszawie. Jego skład jest ustalany po ostatniej sesji w styczniu (korekta roczna), kwietniu, lipcu i październiku (korekty kwartalne). Akutalny stan portfela WIG20 przedstawia poniższy wykres 9.1.

```
> spolki1=c(15.00,13.06,11.44,8.95,8.24,8.17,7.17,3.98,3.94,3.00,2.40,
2.21,2.20,1.90,1.87,1.76,1.43,1.26,1.21,0.82)
> names(spolki1)=c('PKOBP-15.00%','PEKAO-13.06%','KGHM-11.44%','PZU-8.95%'
'PGE-8.24%','PKNORLEN-8.17%','TPSA-7.17%','TAURONPE-3.98%','PGNIG-3.94%',
'BZWBK-3.00%','BRE-2.40%','ASSECOPOL-2.21%','GETIN-2.20%','GTC-1.90%',
'CEZ-1.87%','TVN-1.76%','PBG-1.43%','POLIMEXMS-1.26%','LOTOS-1.21%',
'CYFRPLSAT-0.82%');  pie(spolki1,radius=1,cex=0.6)
```



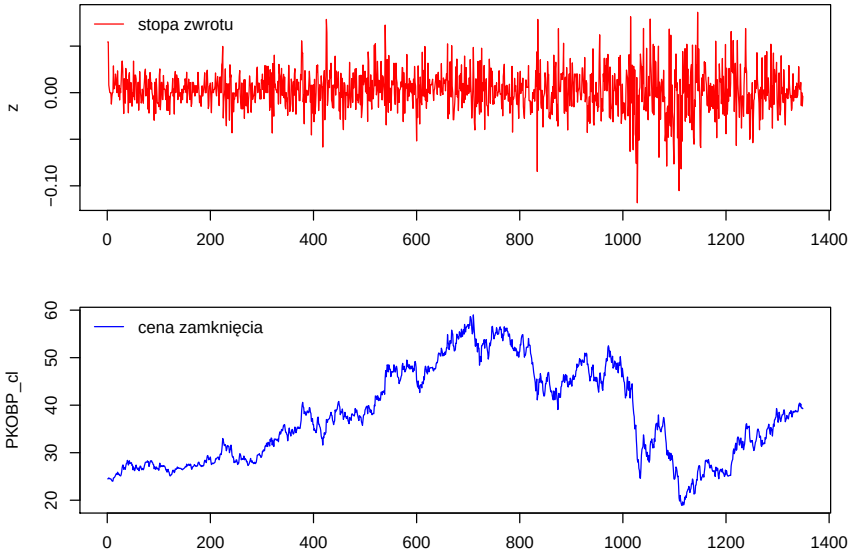
Rysunek 9.1: Struktura indeksu WIG20 — stan na 17 grudnia 2010 r.

9.2 Statystyki opisowe

Do naszych analiz zostanie wykorzystana stopa zwrotu spółki PKOBP, która ma największy udział w strukturze portfela WIG20. W pierwszym kroku wyznaczymy logarytmiczne stopy zwrotu:

$$z = \ln(Close) - \ln(Open) = \ln\left(\frac{Close}{Open}\right)$$

gdzie *Close* oznacza cenę zamknięcia, natomiast *Open* cenę otwarcia.



Rysunek 9.2: Stopy zwrotu i ceny zamknięcia od 2.01.1995 r. do 29.01.2010 r.

Następnie dla zmiennej `z` zostaną obliczone miary położenia takie jak: średnia, mediana oraz kwartyle:

```
> summary(z)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.118200 -0.009693  0.001889  0.002219  0.014330  0.086310
```

a także odchylenie standardowe według wzoru:

$$s = \sqrt{\frac{1}{n-1} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2}$$

```
> sd(z)
[1] 0.02191796
```

Z powyższych obliczeń wynika, że średnia stopa zwrotu w badanym okresie wyniosła $0,22\%$, maksymalna $8,63\%$ a minimalna $-11,82\%$. Natomiast w wypadku 25% sesji stopy zwrotu były mniejsze niż $-0,97\%$, podczas 50% sesji wartości były mniejsze niż $0,19\%$. Z kolei w przypadku 75% sesji zysk z akcji był mniejszy niż $1,43\%$, a w przypadku pozostałych 25% sesji zysk był większy niż $1,43\%$ (rysunek 9.3).

Przy omawianiu różnych statystyk opisowych wato także wspomnieć o funkcji `quantile(stats)`. Za pomocą komendy `quantile` mamy możliwość obliczenia wartość dowolnego kwantyla wykorzystując jedną z dziewięciu dostępnych metod. W tym opracowaniu zaprezentujemy sposób obliczania kwantyli za pomocą metody siódmej. W pierwszym kroku uporządkujemy nasz wektor danych w sposób rosnący:

```
> sz=sort(z)                    # wektor danych uporządkowany rosnąco
```

Następnie wyznaczymy numer kwantyla $Q = 80\%$ według wzoru:

$$h = (n-1) \cdot Q + 1$$

```
> h=(length(z)-1)*0.8+1; h      # numer kwantyla
[1] 1079.4
```

Kolejnym krokiem będzie obliczenie wartości kwantyla na podstawie wzoru:

$$x_{\lfloor h \rfloor} + (h - \lfloor h \rfloor)(x_{\lceil h \rceil} - x_{\lfloor h \rfloor})$$

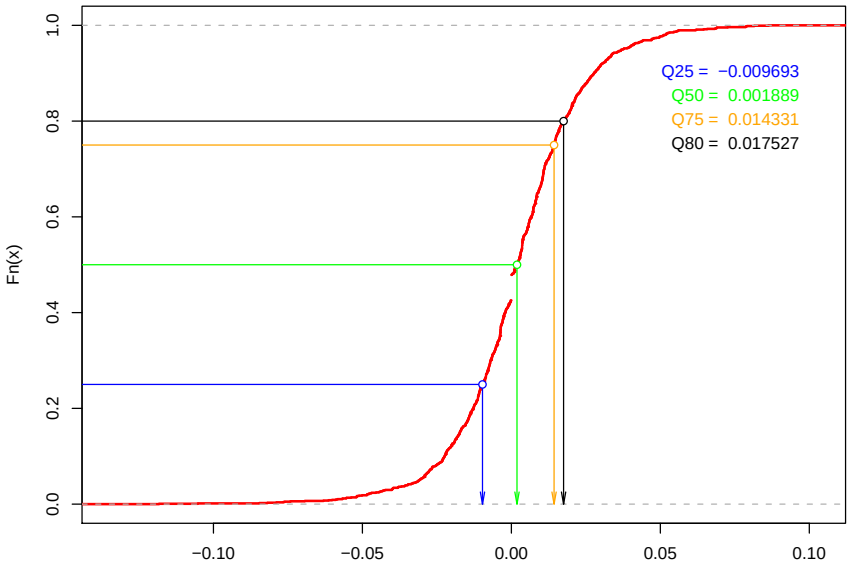
gdzie: $\lfloor h \rfloor$ to największa liczba całkowita nie większa od `h` oraz $\lceil h \rceil$ to najmniejsza liczba całkowita nie mniejsza od `h`.

```
> xL=sz[floor(h)]; xP=sz[ceiling(h)]
> xL+(h-floor(h))*(xP-xL)       # wartość kwantyla
[1] 0.01752695
```

Teraz wykorzystamy gotową funkcję z pakietu `stats`:

```
> quantile(z,0.80,type=7)
       80%
0.01752695
```

Otrzymane wartości kwantyli możemy również zaznaczyć na wykresie (rysunek 9.3).



Rysunek 9.3: Dystrybuanta – prezentacja graficzna kwantyli.

Wyznaczmy jeszcze wzory dla pozostałych metod szcowania kwantyli. Dla metod od 4 do 9 algorytmy różnią się tylko sposobem wyznaczenia numeru kwantyla. Wartość kwantyla jest obliczana tak jak to zostało przedstawione w powyższym przykładzie czyli według wzoru:

$$x_{\lfloor h \rfloor} + (h - \lfloor h \rfloor)(x_{\lceil h \rceil} - x_{\lfloor h \rfloor})$$

- typ 4:

$$h = n \cdot Q$$

- typ 5:

$$h = n \cdot Q + \frac{1}{2}$$

- typ 6:

$$h = (n+1) \cdot Q$$

- typ 8:

$$h = \left(n + \frac{1}{3}\right) \cdot Q + \frac{1}{3}$$

- typ 9:

$$h = \left(n + \frac{1}{4}\right) \cdot Q + \frac{3}{8}$$

W przypadku metod od 1 do 3 do wyznaczenia numeru kwantyla oraz jego wartości korzystamy z następujących wzorów:

- typ 1:

$$h = n \cdot Q + \frac{1}{2}$$
$$x_{\lceil h - \frac{1}{2} \rceil}$$

- typ 2:

$$h = n \cdot Q + \frac{1}{2}$$
$$\frac{x_{\lceil h - \frac{1}{2} \rceil} + x_{\lfloor h + \frac{1}{2} \rfloor}}{2}$$

- typ 3:

$$h = n \cdot Q$$
$$x_{[h]}$$

Za pomocą komendy `quantile` można także wyznaczyć jednocześnie kilka wartości dowolnych kwantyli:

```
> quantile(z,c(0.17,0.60,0.83),type=7)
         17%          60%          83%
-0.015043711  0.006710013  0.020240888
```

W różnych opracowaniach często spotykamy się z następującymi wzorami dotyczącymi skośności:

$$S_1 = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^3}{\sqrt{\left(\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2\right)^3}}$$

```
> library(e1071)
> S1=skewness(z,type=1)
> S1
[1] -0.2016978
```

$$S_2 = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^3 \cdot \frac{\sqrt{n(n-1)}}{n-2}}{\sqrt{\left(\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2\right)^3}}$$

```
> S2=skewness(z,type=2)
> S2
[1] -0.2019224
```

$$S_3 = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3}$$

gdzie $\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^3$ oraz $\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2$ to odpowiednio trzeci i drugi moment centralny oraz s to odchylenie standardowe z próby.

```
> S3=skewness(z,type=3)
> S3
[1] -0.2014735
```

Ponieważ obliczony wskaźnik skośności jest równy $-0,20$ możemy sądzić, że rozkład stopy zwrotu charakteryzuje się niewielką lewostronną skośnością.

```
> library(moments)
> agostino.test(z)

        D'Agostino skewness test

data:  z
skew = -0.2017, z = -1.9845, p-value = 0.0472
alternative hypothesis: data have a skewness
```

Również w przypadku miar koncentracji możemy spotkać się kilkoma wzorami na kurtozę:

$$K_1 = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2\right)^2} - 3$$

```
> K1=e1071::kurtosis(z,type=1)
> K1
[1] 2.402519
```

$$K_2 = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \cdot \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s} \right)^4 - \frac{3 \cdot (n-1)^2}{(n-2)(n-3)}$$

```
> K2=e1071::kurtosis(z,type=2)
> K2
[1] 2.415909
```

$$K_3 = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^4}{s^4} - 3$$

gdzie $\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^4$ oraz $\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^2$ to odpowiednio czwarty i drugi moment centralny oraz s to odchylenie standardowe z próby.

```
> K3=e1071::kurtosis(z,type=3)
> K3
[1] 2.394512
```

Wysoka wartość kurtozy świadczy o tym, że rozkład stóp zwrotu charakteryzuje się spiczastością (rozkład leptokurtyczny) w stosunku do rozkładu normalnego (rozkład mesokurtyczny).

```
> anscombe.test(z)

        Anscombe-Glynn kurtosis test

data:  z
kurt = 5.4025, z = 8.5673, p-value < 2.2e-16
alternative hypothesis: kurtosis is not equal to 3
```

9.3 Rozkład normalny

Funkcja gęstości rozkładu normalnego $\mathcal{N}(\mu, \sigma)$ dla $\mu \in \mathcal{R}$ oraz $\sigma > 0$ jest przedstawiona poniżej:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2} \right)$$

gdzie μ to średnia, natomiast σ to odchylenie standardowe.

```
> dnorm(x,a,b)    # funkcja gęstości rozkładu normalnego
```

Do estymacji nieznanych parametrów rozkładu normalnego wykorzystamy następujące funkcję:

```
> a=mean(z); a   # średnia
[1] 0.002218751
> b=sd(z); b     # odchylenie standardowe
[1] 0.02191796
```

W przypadku gdybyśmy chcieli estymować parametry μ oraz σ za pomocą funkcji `nlm(stats)` kod wyglądałby następująco:

```
# logarytm funkcji wiarygodności:
> f1=function(theta, z) {
+ sum(-dnorm(z, mean=theta[1], sd=theta[2], log=TRUE))
+ }
# parametry startowe:
> p.start=c(mean(z), sd(z))
# optymalizacja funkcji f1:
> e1=nlm(f1, p.start, z=z); e1
$minimum
[1] -3240.138

$estimate
[1] 0.002218512 0.021909709

$gradient
[1] -0.672284 -2.848160

$code
[1] 3

$iterations
[1] 2
```

Jak można zauważyć otrzymane parametry są dokładnie takie same jak te, które oszacowaliśmy za pomocą funkcji `mean(base)` i `sd(stats)`.

W środowisku R można oszacować nieznane parametry również za pomocą funkcji `fitdistr(MASS)`. Jednak jest ona mniej elastyczna od poprzedniej, gdyż korzysta z już wbudowanych funkcji prawdopodobieństwa. A więc nie możemy estymować parametrów dla dowolnego rozkładu.

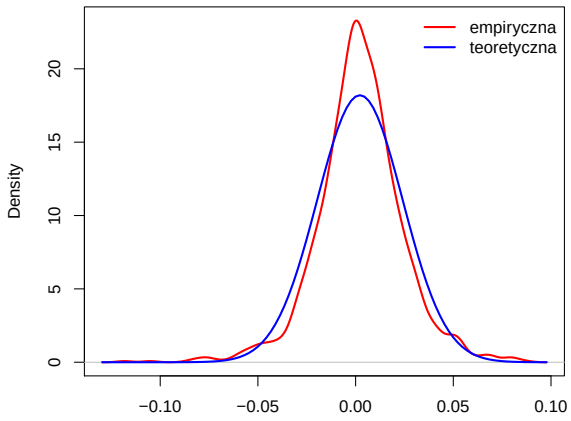
```
> library(MASS)
> fitdistr(z,'normal')
       mean             sd
  0.0022187508   0.0219098351
 (0.0005965312) (0.0004218112)
```

Znając już parametry μ oraz σ możemy teraz przeprowadzić test zgodności Andersona-Darlinga i odpowiedzieć na pytanie: czy stopa zwrotu PKOBP pochodzi z rozkładu normalnego?

```
> library(ADGofTest)
> ad.test(z,pnorm,mean(z),sd(z))

        Anderson-Darling GoF Test

data:  z  and  pnorm
AD = 7.889, p-value = 0.0001265
alternative hypothesis: NA
```

Ponieważ $p - value = 0,0001265$ to hipotetę zerową należy odrzucić. Zatem na poziomie istotności $\alpha = 0,05$ należy stwierdzić, że stopa zwrotu PKOBP nie ma rozkładu normalnego. Naszą decyzję potwierdza także rysunek 9.4.

```
> plot(density(z),col='red',lwd=2)
> curve(dnorm(x,mean(z),sd(z)),add=T,col='blue')
> legend("topright",bg='white',bty="n",cex=1,lty=1,lwd=2,
  c('empiryczna','teoretyczna'),col=c('red','blue'))
```



Rysunek 9.4: Porównanie wykresu gęstości z rozkładem normalnym.

9.4 Rozkład Cauchy'ego

Funkcja gęstości rozkładu Cauchy'ego $\mathcal{C}(a, b)$ dla $a \in \mathcal{R}$ oraz $b > 0$ jest przedstawiona poniżej:

$$f(x) = \frac{1}{\pi b \cdot \left(1 + \left(\frac{x-a}{b}\right)^2\right)}$$

gdzie a to parametr położenia, natomiast b to parametr skali.

```
> dcauchy(x,a,b)  # funkcja gęstości rozkładu Cauchy'ego
```

Tak samo jak w przypadku rozkładu normalnego, aby wykorzystać funkcję `nlm` należy oszacować parametry startowe. Dla rozkładu Cauchy'ego można to zrobić w następujący sposób:

```
> a=median(z); a      # parametr położenia: a
[1] 0.001888575
> b=IQR(z)/2; b       # parametr skali: b
[1] 0.01201212
```

Teraz metodą największej wiarygodności oszacujemy parametry rozkładu Cauche'go.

```
> f2=function(theta, z) {
+ sum(-dcauchy(z, location=theta[1], scale=theta[2], log=TRUE))
+ }
> p.start=c(median(z), IQR(z)/2)
> e2=nlm(f2, p.start, z=z); e2
$minimum
[1] -3158.958
```

```
$estimate
[1] 0.00205985 0.01105443

$gradient
[1] -0.3534797 -0.0820578

$code
[1] 2

$iterations
[1] 5
```

Prawie identyczne wyniki otrzymamy przy wykorzystaniu funkcji `fitdistr`.

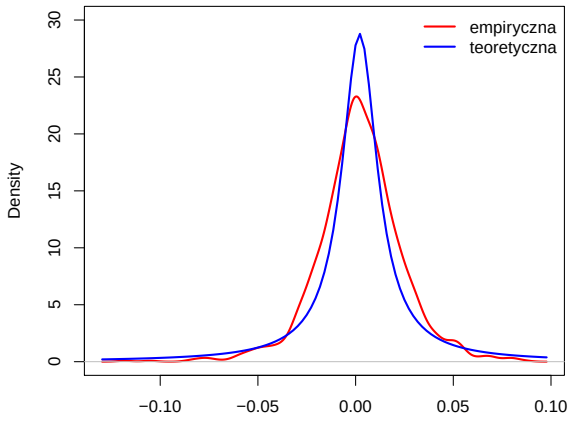
```
> fitdistr(z,'cauchy')
     location         scale
  0.0020690541   0.0110547804
 (0.0004695788) (0.0003874817)
```

Do sprawdzenia zgoności rozkładu empirycznego z rozkładem Cauchy'ego zastosujemy test Andersona-Darlinga.

```
> ad.test(z,pcauchy,0.0020690541, 0.0110547804)

        Anderson-Darling GoF Test

data:  z  and  pcauchy
AD = 11.0569, p-value = 1.028e-06
alternative hypothesis: NA
```

Także i w tym razem na poziomie istotności $\alpha = 0,05$ odrzucamy hipotezę zerową, która zakłada, że stopa zwrotu ma rozkład Cauchy'ego.

```
> plot(density(z),ylim=c(0,30),col='red',lwd=2)
> curve(dcauchy(x,0.0020690541, 0.0110547804),add=T,col='blue',lwd=2)
> legend("topright",bg='white',bty="n",cex=1,lty=1,lwd=2,
c('empiryczna','teoretyczna'),col=c('red','blue'))
```



Rysunek 9.5: Porównanie wykresu gęstości z rozkładem Cauchy'ego.

9.5 Rozkład Laplace'a

Funkcja gęstości rozkładu Laplace'a $\mathcal{L}(a, b)$ dla $a \in \mathcal{R}$ oraz $b > 0$ jest przedstawiona poniżej:

$$f(x) = \frac{1}{2b} \exp\left(\frac{-|x-a|}{b}\right)$$

gdzie a to parametr położenia, natomiast b to parametr skali.

```
> library(VGAM)
> dlaplace(x,a,b) # funkcja gęstości rozkładu Laplace'a
```

Parametry startowe oszacujemy w następujący sposób:

```
> a=median(z); a      # parametr położenia: a
[1] 0.001888575
> b=sd(z)/sqrt(2); b  # parametr skali: b
[1] 0.01549834
```

Mając do dyspozycji funkcję gęstości `dlaplace` oraz parametry startowe dokonamy estymacji parametrów `a` oraz `b` dla rozkładu Laplace'a.

```
> f3=function(theta, z) {
+ sum(-dlaplace(z, location=theta[1], scale=theta[2], log=TRUE))
+ }
> p.start=c(median(z), sd(z)/sqrt(2))
> e3=nlm(f3, p.start, z=z); e3
$minimum
[1] -3292.208

$estimate
[1] 0.001888338 0.016024837

$gradient
[1] -49.258835  -1.182744

$code
[1] 3

$iterations
[1] 6
```

Jak już wcześniej zostało podkreślone, za pomocą funkcji `fitdistr(MASS)` nie możemy estymować parametrów dowolnego rozkładu prawdopodobieństwa. Jednak w paczce `VGAM` jest dostępna funkcja `vglm` dzięki której oszacujemy parametry omawianego rozkładu.

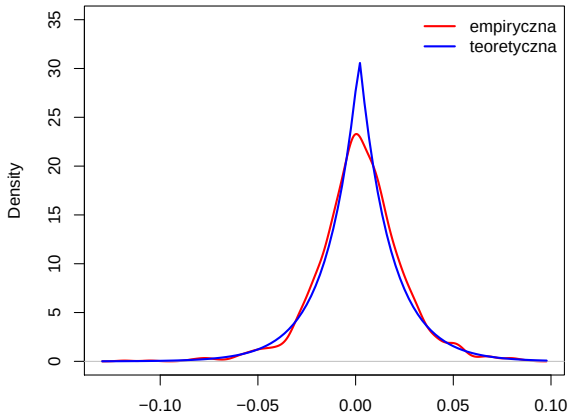
```
> fit=vglm(z  ~ 1, laplace, data.frame(z), trace=TRUE, crit="l")
> Coef(fit)
VGLM    linear loop  1 :  loglikelihood = 3292.207
VGLM    linear loop  2 :  loglikelihood = 3292.206
Taking a modified step...........
Warning message:
In eval(expr, envir, enclos) :
```

```
  iterations terminated because half-step sizes are very small
   location       scale
0.001900460 0.016024649
```

```
> ad.test(z,plaplace,0.001900460, 0.016024649)

        Anderson-Darling GoF Test

data:  z  and  plaplace
AD = 1.3817, p-value = 0.2074
alternative hypothesis: NA
```

Ponieważ $p - value = 0,2074$, więc nie ma podstaw do odrzucenia hipotezy zerowej. Zatem można przyjąć, że rozkład stóp zwrotu spółki PKOBP ma rozkład Laplace'a.

```
> plot(density(z),ylim=c(0,35),col='red',lwd=2)
> curve(dlaplace(x,0.001900460, 0.016024649),add=T,col='blue',lwd=2)
> legend("topright",bg='white',bty="n",cex=1,lty=1,lwd=2,
c('empiryczna','teoretyczna'),col=c('red','blue'))
```



Rysunek 9.6: Porównanie wykresu gęstości z rozkładem Laplace'a.

9.6 Rozkład Stabilny

Rozkład stabilny $\mathcal{S}(\alpha, \beta, \gamma, \delta)$, gdzie $\alpha \in (0; 2\rangle$ to parametr kształtu, $\beta \in \langle -1; 1\rangle$ to indeks skośności, $\gamma > 0$ to parametr skali, oraz $\delta \in \mathcal{R}$ to parametr położenia, nie ma ściśle określonej funkcji gęstości. W zależności od wartości parametrów α oraz β możemy otrzymać rozkłady, takie jak: rozkład normalny: $\alpha = 2$, rozkład Cauchy'ego: $\alpha = 1$ i $\beta = 0$ oraz rozkład Levy'ego: $\alpha = \frac{1}{2}$ i $\beta = -1$.

```
> library(fBasics)
> dstable(x,alpha,beta,gamma,delta) # gęstość rozkładu stabilnego
```

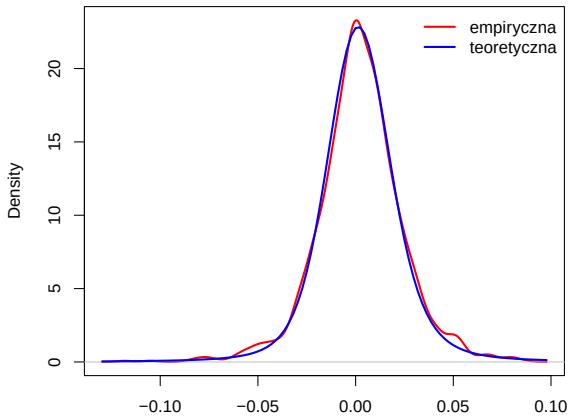
Do oszacowania parametrów rozkładu stabilnego wykorzystamy funkcję `stableFit(fBasics)`. Dzięki opcji `type` mamy możliwość wyboru jednej z dwóch dostępnych metod estymacji: metoda kwantyli — `type="q"` lub metoda największej wiarygodności — `type="mle"`.

```
> stableFit(z)

Title:
 Stable Parameter Estimation

Call:
 .qStableFit(x = x, doplot = doplot, title = title, description =
    description)

Model:
 Stable Distribution

Estimated Parameter(s):
      alpha        beta       gamma       delta
1.640000000 0.125000000 0.012460300 0.001564710
```

```
> ad.test(z,pstable,1.64 ,0.125, 0.012460300, 0.001564710)

        Anderson-Darling GoF Test

data:  z  and  pstable
AD = 0.6095, p-value = 0.639
alternative hypothesis: NA
```

Test Andersona-Darlinga wykazał, że badana zmienna ma rozkład stabilny o parametrach: $\alpha = 1,64$, $\beta = 0,125$, $\gamma = 0,0124603$ oraz $\delta = 0,00156471$. Warto zwrócić uwagę na wysokość p-value dla rozkładu stabilnego, która wynosi $0,639$ i jest większa niż dla rozkładu Laplace'a $(0,2074)$. A zatem należy sądzić, że właśnie ten rozkład lepiej opisuje badaną zmienną (rysunek 9.7).

```
> plot(density(z),col='red',lwd=2)
> curve(dstable(x,1.64 ,0.125, 0.012460300, 0.001564710),
  add=T,col='blue',lwd=2)
> legend("topright",bg='white',bty="n",cex=1,lty=1,lwd=2,
c('empiryczna','teoretyczna'),col=c('red','blue'))
```



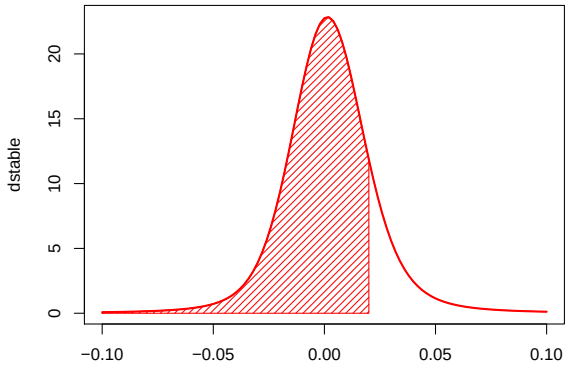
Rysunek 9.7: Porównanie wykresu gęstości z rozkładem stabilnym.

Znając rozkład badanej zmiennej można także obliczyć prawdopodobieństwo wystąpienia określonej stopy zwrotu. W tym celu wykorzystamy funkcję `pstable(fBasics)` czyli dystrybuantę rozkładu stabilnego.

1. Jakie jest prawdopodobieństwo, że stopa zwrotu będzie mniejsza niż 2%?

$$P(z < 0,02)$$

```
> a=1.64;b=0.125;g=0.012460300;d=0.001564710
> pstable(0.02,a,b,g,d)
[1] 0.8345802
attr(,"control")
   dist alpha  beta     gamma      delta pm
 stable  1.64 0.125 0.0124603 0.00156471  0
```

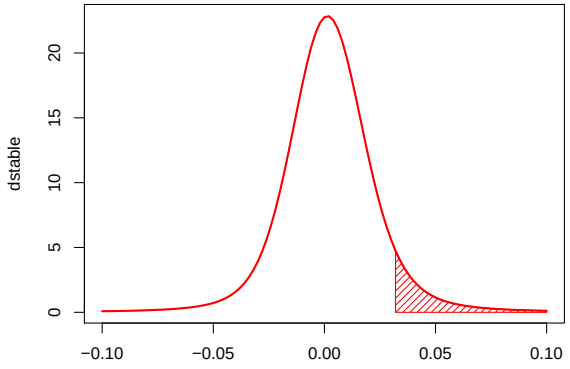


Rysunek 9.8: Obszar prawdopodobieństwa $P(z < 0,02) = 0,8345802$.

2. Jakie jest prawdopodobieństwo, że stopa zwrotu będzie większa niż $3,2\%$?

$$P(z > 0,032)$$

```
> 1-pstable(0.032,a,b,g,d)[1]
[1] 0.07052998
```

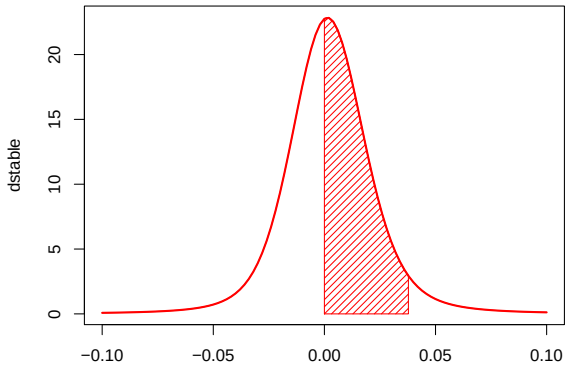


Rysunek 9.9: Obszar prawdopodobieństwa $P(z > 0,032) = 0,07052998$.

3. Jakie jest prawdopodobieństwo, że stopa zwrotu będzie się zawierać w przedziale $(0\%; 3,78\%)$?

$$P(0 < z < 0,0378)$$

```
> pstable(0.0378,a,b,g,d)[1]-pstable(0,a,b,g,d)[1]
[1] 0.494233
```

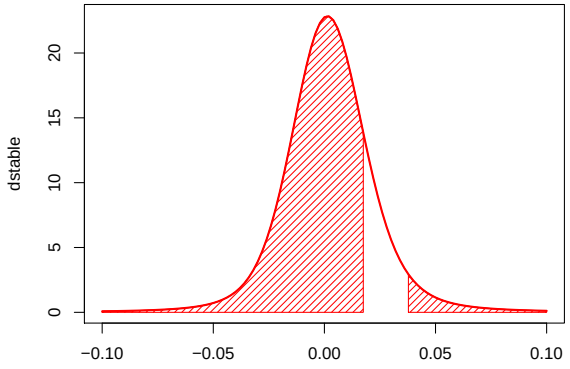


Rysunek 9.10: Obszar prawdopodobieństwa $P(0 < z < 0,0378) = 0,494233$.

4.Jakie jest prawdopodobieństwo, że stopa zwrotu będzie mniejsza niż $1,75\%$ lub większa niż $3,78\%$?

$$P(0,0175 > z > 0,0378)$$

```
> pstable(0.0175,a,b,g,d)[1]+(1-pstable(0.0378,a,b,g,d)[1])
[1] 0.8511473
```



Rysunek 9.11: Obszar prawdopodobieństwa $P(0,0175 > z > 0,0378)) = 0,8511473$.

Na zakończenie warto jeszcze wspomnieć, że środowisko R oferuje także wiele innych rozkładów prawdopodobieństwa wraz z funkcjami umożliwiającymi szacowanie ich parametrów. W paczce `fBasics` oprócz rozkładu stabilnego mamy do dyspozycji takie rozkłady jak:

- odwrotny rozkład normalny (*Normal Inverse Gaussian Distribution*)
  gdzie:
  `dnig` – funkcja gęstości, `pnig` – dystrybuanta, `qnig` – kwantyle, `rnig` – zmienne losowe, `nigFit` – estymacja parametrów.
- rozkład hiperboliczny (*Hyperbolic Distribution*)
  gdzie:
  `dhyp` – funkcja gęstości, `phyp` – dystrybuanta, `qhyp` – kwantyle, `rhyp` – zmienne losowe, `hypFit` – estymacja parametrów.

- uogolniony rozkład hiperboliczy (*Generalized Hyperbolic Distribution*)
  gdzie:
  `dgh` – funkcja gęstości, `pgh` – dystrybuanta, `qgh` – kwantyle, `rgh` – zmienne losowe, `ghFit` – estymacja parametrów.

- uogolniony rozkład hiperboliczy t-Studenta (*Generalized Hyperbolic Student-t*)
  gdzie:
  `dght` – funkcja gęstości, `pght` – dystrybuanta, `qght` – kwantyle, `rght` – zmienne losowe, `ghtFit` – estymacja parametrów.

- uogolniony rozkład lambda (*Generalized Lambda Distribution*)
  gdzie:
  `dgld` – funkcja gęstości, `pgld` – dystrybuanta, `qgld` – kwantyle, `rgld` – zmienne losowe, `gldFit` – estymacja parametrów.

Z kolei w bibliotece `fGarch` możemy skorzystać z następujących rozkładów:

- uogólniony rozkład błędów (*Generalized Error Distribution – GED*)
  gdzie:
  `dged` – symetryczna funkcja gęstości, `gedFit` – estymacja parametrów,
  `dsged` – skośna funkcja gęstości, `sgedFit` – estymacja parametrów.

- rozkład t-Studenta (*Student-t Distribution*)
  gdzie:
  `dstd` – symetryczna funkcja gęstości, `stdFit` – estymacja parametrów,
  `dsstd` – skośna funkcja gęstości, `sstdFit` – estymacja parametrów.

- rozkład normalny (*Normal Distribution*)
  gdzie:
  `dnorm` – symetryczna funkcja gęstości, `normFit` – estymacja parametrów,
  `dsnorm` – skośna funkcja gęstości, `snormFit` – estymacja parametrów.

Rozdział 10

Przykład budowy dynamicznego modelu liniowego

10.1 Wprowadzenie

W skład dynamicznego modelu liniowego wchodzą następujące elementy:

- trend
- składnik sezonowości
- składnik autoregresji

Do estymacji takiego modelu zostaną wykorzystane dodatkowe pakiety czyli takie które nie są instalowane wraz z oprogramowaniem R. Tak więc jeśli wcześniej nie zostały one zainstalowane w systemie, należy to zrobić w następujący sposób:

```
# instalacja pakietu tseries
> install.packages("tseries")
```

Instalację pakietu wykonujemy tylko raz niezależnie od tego ile razy uruchomimy środowisko R. Wykonując powyższe polecenie zostanie zainstalowany pakiet wraz z wymaganymi zależnościami (dodatkowe pakiety) jeśli takie są wymagane. Następnie należy taką bibliotekę załadować:

```
# wczytywanie pakietu tseries:
> library("tseries")    # diagnostyka szeregów czasowych
```

Komendę ładowania pakietu (oczywiście jeśli chcemy skorzystać z funkcji dostępnych w tym pakiecie) musimy powtarzać przy każdym uruchomieniu programu R. W dalszej kolejności wprowadzamy dane dotyczące miesięcznej stopy bezrobocia w Polsce w okresie od 05.2004 do 05.2010.

```
# wprowadzamy dane do programu R:
> b=c(
19.5, 19.4, 19.3, 19.1, 18.9, 18.7, 18.7, 19.0, 19.4, 19.4, 19.2, 18.7,
18.2, 18.0, 17.9, 17.7, 17.6, 17.3, 17.3, 17.6, 18.0, 18.0, 17.8, 17.2,
16.5, 16.0, 15.7, 15.5, 15.2, 14.9, 14.8, 14.9, 15.1, 14.9, 14.4, 13.7,
13.0, 12.4, 12.2, 11.9, 11.6, 11.3, 11.3, 11.4, 11.7, 11.5, 11.1, 10.5,
10.0,  9.6,  9.4,  9.1,  8.9,  8.8,  9.1,  9.5, 10.5, 10.9, 11.2, 11.0,
```

```
10.8, 10.7, 10.8, 10.8, 10.9, 11.1, 11.4, 11.9, 12.7, 13.0, 12.9, 12.3,
11.9)
```

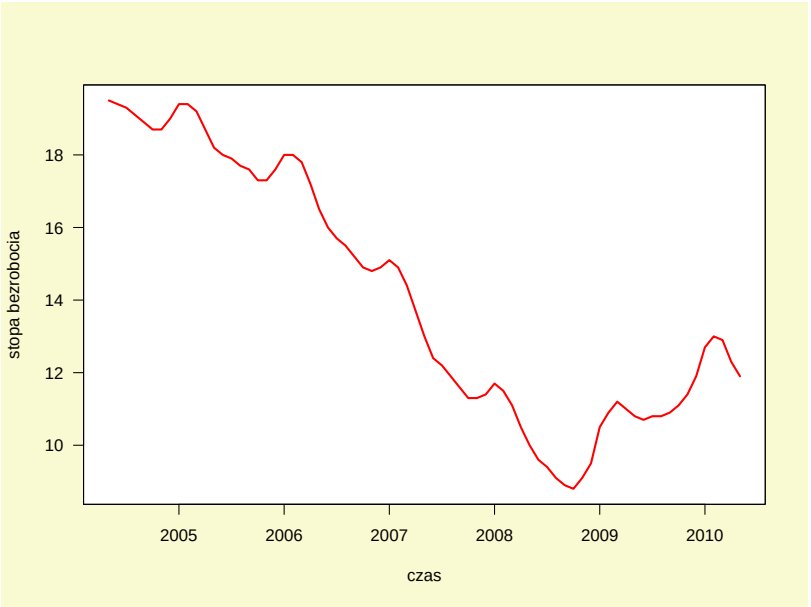
Oczywiście dane można importować do programu R (lub eksportować) za pomocą różnych funkcji:

```
# dane w formie tabeli:
> daneBez=data.frame(b)
# zapisywanie danych:
> write.table(daneBez,"/home/.../Pulpit/daneBez")
# odczytywanie danych:
> naszeDane=read.table("/home/.../Pulpit/daneBez")
```

Istnieje także możliwość odczytywania i zapisywania danych do takich typów plików jak np. .csv.

Aby móc zastosować funkcje `dyn$lm` konieczne jest przekształcenie zmiennej `b` (zapisnej w postaci wektora zmiennych) w szereg czasowy. Do budowy szeregu czasowego zostanie wykorzystana funkcja `ts(stats)`.

```
# budowa szeregu czasowego:
> bezrob=ts(b,start=c(2004,5),freq=12)
# wykres szeregu czasowego:
> pdf("r1a.pdf",width=8,height=6,paper="special")
> par(bg="lightgoldenrodyellow")
> plot(bezrob,axes=F,xlab="",ylab="",main="", col="white")
> u=par("usr")
> rect(u[1], u[3], u[2], u[4], col="white")
> par(new=plot)
> plot(bezrob,xlab="czas",ylab="stopa bezrobocia",col="red",lwd=2,las=1)
> dev.off()
```



Rysunek 10.1: Stopa bezrobocia w Polsce od 0.5.2004 do 0.5.2010.

10.2 Badanie wewnętrznej struktury procesu

10.2.1 Trend

W pierwszym kroku należy ustalić, czy w szeregu czasowym (Rysunek 10.1) występuje tendencja rozwojowa czyli trend. Jeśli dojdziemy do wniosku, że występuje trend należy zbudować kilka wielomianowych modeli trendu a następnie dokonać wyboru najlepszego modelu. Wybór wielomianu możemy dokonać za pomocą analizy wariancji ANOVA lub kryterium informacyjnego AIC.

```
# zmienne trendu:
> t1=ts(1:length(bezrob),start=c(2004,5),freq=12)
> t2=ts(t1^2,start=c(2004,5),freq=12)
> t3=ts(t1^3,start=c(2004,5),freq=12)
> t4=ts(t1^4,start=c(2004,5),freq=12)
```

```
> library(dyn)
# estymacja wielomianowych modeli trendu:
> m1=dyn$lm(bezrob~t1)          # wielomian stopnia pierwszego
> m2=dyn$lm(bezrob~t1+t2)       # wielomian stopnia drugiego
> m3=dyn$lm(bezrob~t1+t2+t3)    # wielomian stopnia trzeciego
> m4=dyn$lm(bezrob~t1+t2+t3+t4) # wielomian stopnia czwartego
```

```
# porównanie modelu trendu m1 i m2:
> anova(m1,m2)
Analysis of Variance Table

Model 1: bezrob ~ t1
Model 2: bezrob ~ t1 + t2
  Res.Df    RSS Df Sum of Sq     F    Pr(>F)
1     71 176.96
2     70 109.76  1    67.202 42.86 8.284e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
# porównanie modelu trendu m2 i m3:
> anova(m2,m3)
Analysis of Variance Table

Model 1: bezrob ~ t1 + t2
Model 2: bezrob ~ t1 + t2 + t3
  Res.Df     RSS Df Sum of Sq   F    Pr(>F)
1     70 109.756
2     69  26.572  1    83.183 216 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
# porównanie modelu trendu m3 i m4:
> anova(m3,m4)
Analysis of Variance Table
```

```
Model 1: bezrob ~ t1 + t2 + t3
Model 2: bezrob ~ t1 + t2 + t3 + t4
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     69 26.572
2     68 26.569  1 0.0034355 0.0088 0.9256
```

Tak więc najlepszym modelem trendu okazał się wielomian stopnia trzeciego. Do takiego samego wniosku dochodzimy porównując kryteria informacyjne AIC wszystkich modeli.

```
> AIC(m1,m2,m3,m4)
   df      AIC
m1  3 277.8030
m2  4 244.9342
m3  5 143.3919
m4  6 145.3825
```

10.2.2 Sezonowość

Aby ocenić czy w badanym szeregu czasowym występują wahania sezonowe trzeba oszacować model trendu wraz ze zmiennymi sezonowymi. Jeśli przynajmniej jeden parametr przy zmiennej zero-jedynkowej okaże się istotny możemy sądzić, że w szeregu występuje sezonowość.

```
# zmienne sezonowe: zero-jedynkowe:
> library(forecast)
> month = ts(seasonaldummy(bezrob),start=c(2004,5),freq=12)
```

```
# model z trendem i sezonowością:
> s.dyn=dyn$lm(bezrob~t1+t2+t3+month)
```

```
# podsumowanie modelu z trendem i sezonowością:
> summary(s.dyn)

Call:
lm(formula = dyn(bezrob ~ t1 + t2 + t3 + month))

Residuals:
     Min       1Q   Median       3Q      Max
-1.23729 -0.31580 -0.01533  0.36128  0.82255

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.903e+01  3.153e-01  60.363  < 2e-16 ***
t1           1.258e-01  2.853e-02   4.410 4.54e-05 ***
t2          -1.280e-02  8.917e-04 -14.357  < 2e-16 ***
t3           1.372e-04  7.927e-06  17.311  < 2e-16 ***
month1       5.935e-01  2.815e-01   2.109   0.0393 *
month2       7.138e-01  2.816e-01   2.535   0.0140 *
month3       5.934e-01  2.818e-01   2.106   0.0396 *
month4       1.149e-01  2.821e-01   0.407   0.6854
```

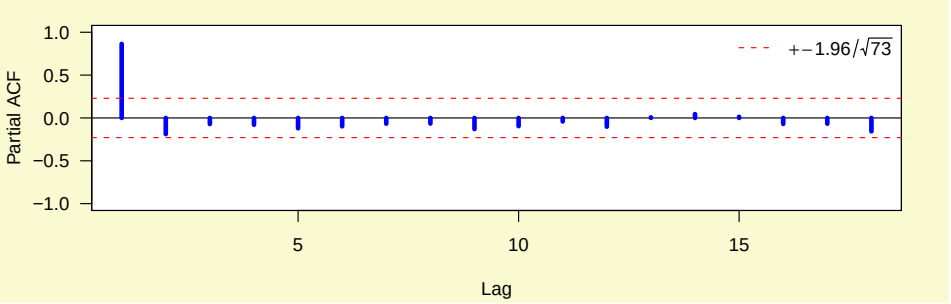
```
month5      -2.402e-01  2.721e-01  -0.883   0.3811
month6      -2.513e-01  2.829e-01  -0.888   0.3780
month7      -2.861e-01  2.824e-01  -1.013   0.3152
month8      -3.891e-01  2.820e-01  -1.380   0.1730
month9      -4.612e-01  2.818e-01  -1.637   0.1071
month10     -5.365e-01  2.816e-01  -1.905   0.0617 .
month11     -3.658e-01  2.815e-01  -1.300   0.1988
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4874 on 58 degrees of freedom
Multiple R-squared: 0.9842,     Adjusted R-squared: 0.9803
F-statistic: 257.4 on 14 and 58 DF,  p-value: < 2.2e-16
```

Ponieważ kilka zmiennych sezonowych jest istotnych statystycznie należy stwierdzić, że w badanym procesie występują wahania sezonowe.

10.2.3 Stopień autoregresji — funkcja PACF

Aby ocenić stopień autoregresji dynamicznego modelu liniowego należy sprawdzić czy w procesie reszt (modelu z trendem i sezonowością) występuje autokorelacja reszt. Można tego dokonać na podstawie funcji PACF (Rysunek 10.2) na poziomie istotności $\alpha = 0,05$ (przerywana linia pozioma).

```
# wyznaczenie współczynników autokorelacji cząstkowej:
> pacf(rs,plot=F)[[1]]
```



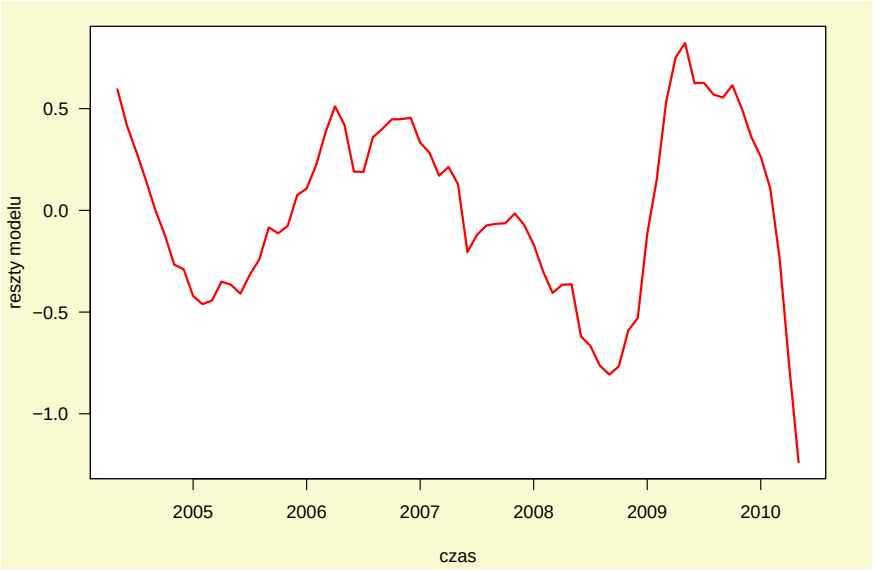
Rysunek 10.2: Funkcja autokorelacji cząstkowej.

Jak widać na rysunku 10.2 występuje rząd autokorelacji $p = 1$.

10.2.4 Stopień integracji — test ADF/PP

Jeśli otrzymane reszty (Rysunek 10.3) na podstawie modelu z trendem i sezonowością są niestacjonarne (występuje pierwiastk jednostkowy) należy wówczas rząd autoregresji (w dynamicznym modelu liniowym) powiększyć o liczbę d czyli liczbę pierwiastków jednostkowych. Do sprawdzenia hipotezy o występowaniu pierwiastka jednostkowego możemy posłużyć się rozszerzonym testem Dickey'a-Fullera.

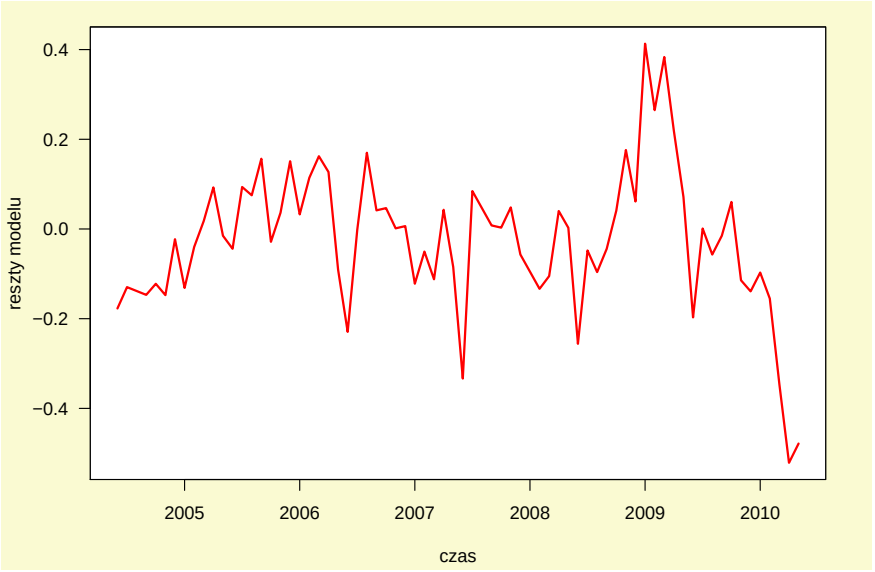
```
> adf.test(rs)
```



Rysunek 10.3: Reszty modelu z trendem i sezonowością rs_t.

```
        Augmented Dickey-Fuller Test

data:  rs
Dickey-Fuller = -2.6101, Lag order = 4, p-value = 0.3268
alternative hypothesis: stationary
```

Ponieważ w teście ADF p-value jest równe $0,3268$ należy przyjąć, że w procesie rs_t występuje pierwiastk jednostkowy $d = 1$. Skoro szereg rs_t jest niestacjonarny należy sprawdzić czy w procesie $\Delta rs_t = rs_t - rs_{t-1}$ również występuje pierwiastek jednostkowy. A więc czy szereg rs_t jest zintegrowany w stopniu drugin tzn. $I(2)$.

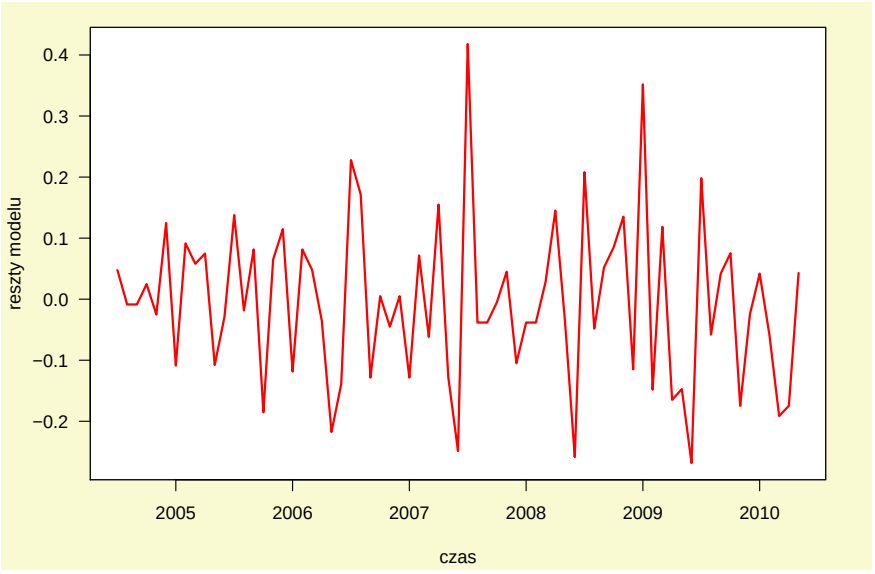


Rysunek 10.4: Reszty modelu po jednokrotnym różnicowaniu Δrs_t.

```
> adf.test(diff(rs))
```

```
        Augmented Dickey-Fuller Test

data:  diff(rs)
Dickey-Fuller = -1.7534, Lag order = 4, p-value = 0.6757
alternative hypothesis: stationary
```

Tak więc szereg rs_t jest zintegrowany w stopniu drugim (p-value $= 0,6757$), badamy więc czy jest również zintegrowany w stopniu trzecim.



Rysunek 10.5: Reszty modelu po dwukrotnym różnicowaniu $\Delta\Delta rs_t$.

```
> adf.test(diff(diff(rs)))

        Augmented Dickey-Fuller Test

data:  diff(diff(rs))
Dickey-Fuller = -4.4611, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary
```

Zatem dla H_0: $I(3)$ p-value jest równe 0.01.Czyli szereg $\Delta\Delta rs_t$ jest stacjonarny i na tym kończymy procedurę oceny zintegrowania szeregu rs_t. Rząd autoregresji w dynamicznym modelu liniowym będzie więc wynosił: $p + d = 1 + 2 = 3$. Innym testem do oceny istnienia pierwiastka jednostkowego jest test Phillipsa-Perrona.

```
> pp.test(rs)

        Phillips-Perron Unit Root Test

data:  rs
Dickey-Fuller Z(alpha) = -8.8842, Truncation lag parameter = 3, p-value
= 0.5915
alternative hypothesis: stationary
```

```
> pp.test(diff(rs))

        Phillips-Perron Unit Root Test

data:  diff(rs)
Dickey-Fuller Z(alpha) = -22.3947, Truncation lag parameter = 3,
p-value = 0.02967
alternative hypothesis: stationary

> pp.test(diff(diff(rs)))

        Phillips-Perron Unit Root Test

data:  diff(diff(rs))
Dickey-Fuller Z(alpha) = -80.4955, Truncation lag parameter = 3,
p-value = 0.01
alternative hypothesis: stationary
```

W oprogramowaniu R możemy również skorzystać z szeregu innych testów dotyczących pierwiastka jednostkowego. Są one dostępne w następujących paczkach: `uroot`, `urca`, `fUnitRoots`.

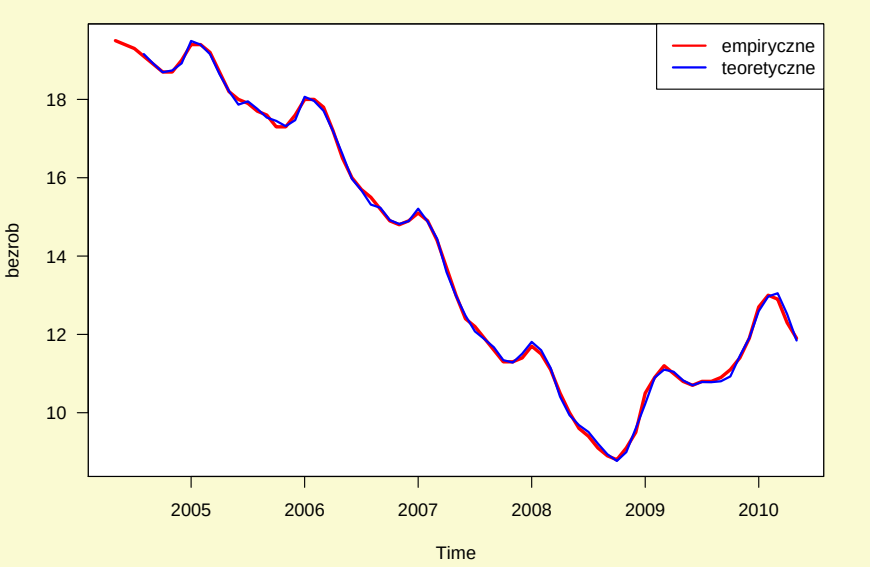
10.3 Weryfikacja modelu

10.3.1 Estymacja dynamicznego modelu liniowego

```
# estymacja modelu:
> m.dyn=dyn$lm(bezrob~t1+t2+t3+
                    month+
                    lag(bezrob,-1)+lag(bezrob,-2)+lag(bezrob,-3))
```

```
# podsumowanie modelu
> summary(m.dyn)

Call:
lm(formula = dyn(bezrob ~ t1 + t2 + t3 + month + lag(bezrob,
    -1) + lag(bezrob, -2) + lag(bezrob, -3)))

Residuals:
       Min         1Q     Median         3Q        Max
-0.2181171 -0.0556411  0.0008272  0.0517680  0.2725021

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.716e+00  6.131e-01   2.799 0.007172 **
t1               8.245e-03  1.169e-02   0.705 0.483721
t2              -8.612e-04  5.818e-04  -1.480 0.144833
t3               9.069e-06  5.615e-06   1.615 0.112309
month1           6.025e-02  6.328e-02   0.952 0.345409
```

```
month2          -5.624e-01  7.525e-02  -7.474 8.69e-10 ***
month3          -6.115e-01  1.129e-01  -5.417 1.58e-06 ***
month4          -6.728e-01  8.569e-02  -7.852 2.18e-10 ***
month5          -3.977e-01  9.862e-02  -4.033 0.000181 ***
month6          -1.633e-01  8.482e-02  -1.925 0.059724 .
month7          -4.991e-02  7.443e-02  -0.671 0.505471
month8          -3.025e-01  6.263e-02  -4.829 1.25e-05 ***
month9          -3.062e-01  7.022e-02  -4.361 6.14e-05 ***
month10         -3.064e-01  6.484e-02  -4.725 1.79e-05 ***
month11         -7.425e-02  6.629e-02  -1.120 0.267842
lag(bezrob, -1)  1.478e+00  1.297e-01  11.400 9.31e-16 ***
lag(bezrob, -2) -1.947e-01  2.453e-01  -0.794 0.430944
lag(bezrob, -3) -3.605e-01  1.444e-01  -2.497 0.015719 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1006 on 52 degrees of freedom
  (6 observations deleted due to missingness)
Multiple R-squared: 0.9993,     Adjusted R-squared: 0.9991
F-statistic:  4551 on 17 and 52 DF,  p-value: < 2.2e-16
```



Rysunek 10.6: Wykres wartości empirycznych i teoretycznych.

10.3.2 Ocena jakości modelu

Dokonując oceny jakości modelu należy zwrócić uwagę na wysoką wartość R^2 który wynosi $0,9993$. Oznacza to, że w $99,93\%$ została wyjaśniona zmienność zmiennej zależnej (stopa bezrobocia) przez model. Na podstawie oszacowanego błędu standardowego reszt Se, który jest równy $0,1006$ możemy stwierdzić, że średnio o $0,1006\%$ odchylają się wartości rzeczywiste stopy bezrobocia od wartości teoretycznych — oszacowanych na podstawie modelu. Także test F wskazuje na dobre dopasowanie modelu (p-value $= 2.2e-16$).

10.4 Diagnostyka modelu

10.4.1 Normalność procesu resztowego

Ocena normalności składnika resztowego została dokonana za pomocą testu Shapiro-Wilka oraz testu χ^2 w oparciu o procedurę zaproponowaną przez Doornika oraz Hansena.

```
# reszty modelu:
> r.dyn=resid(m.dyn)

# test Shapiro-Wilka:
> shapiro.test(r.dyn)

        Shapiro-Wilk normality test

data:  r.dyn
W = 0.9862, p-value = 0.6382
```

Wysoka wartość p-value, która wynosi $0,6382$ pozwala nam wnioskować, że na podstawie testu Shapiro-Wilka reszty mają rozkład normalny.

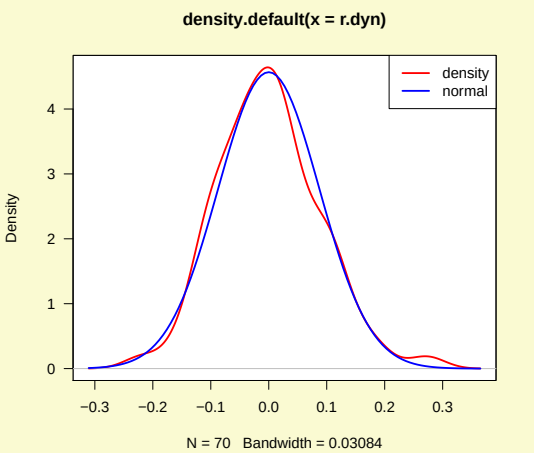
```
# test Doornika-Hansena:
> library(normwhn.test)
> normality.test1(as.matrix(r.dyn))

[1] "H0: data are normally distributed"
[1] "Ep"
         [,1]
[1,] 2.727479
[1] "dof"
[1] 2
[1] "sig.Ep"
          [,1]
[1,] 0.2557027
```

Również wynik testu χ^2 wskazuje, że nie ma podstaw do odrzucenia hipotezy zerowej przy poziomie istotności $\alpha = 0,05$. Należy więc stwierdzić, że proces resztowy ma rozkład normalny. Wyniki obu testów potwierdzają też wykresy (Rysunek 10.7) oraz (Rysunek 10.8).

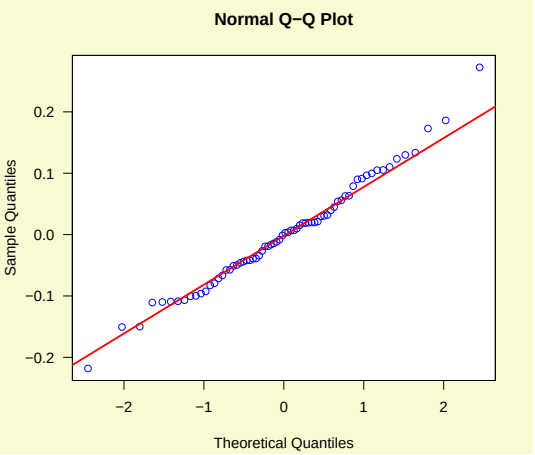
Do badania normalności zmiennych można wykorzystać także inne testy, które są dostępne w środowisku R. Oto niektóre z nich: `jarque.test(moments)` — test normalności Jarque-Bera, `agostino.test(moments)` — test skośności D'Agostino, `anscombe.test(moments)` — test kurtozy Anscombe-Glynn.

10.4.2 Autokorelacja procesu resztowego

Do oceny autokorelacji reszt można wykorzystać test autokorelacji Ljunga-Boxa, który jest dostępny w paczce `tseries`. W tym teście dzięki opcji `lag` możemy badać stopień autokorelacji dowolnego rzędu.



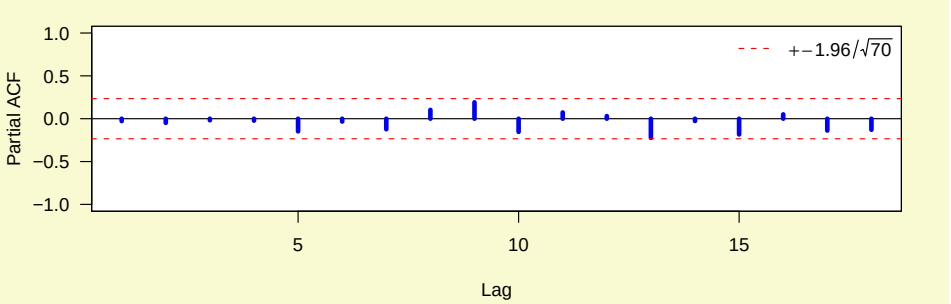
Rysunek 10.7: Gęstość prawdopodobieństwa.



Rysunek 10.8: Wykres kwantylowy.

```
> Box.test (r.dyn, lag = 1, type="Ljung")

        Box-Ljung test

data:  r.dyn
X-squared = 0.0592, df = 1, p-value = 0.8078
```

Ponieważ w teście Ljunga-Boxa wartość p-value jest równa $0,8078$ to na poziomie istotności $\alpha = 0,05$ można przyjąć, że w procesie resztowym autokorelacja nie występuje.



Rysunek 10.9: Funkcja autokorelacji cząstkowej reszt modelu.

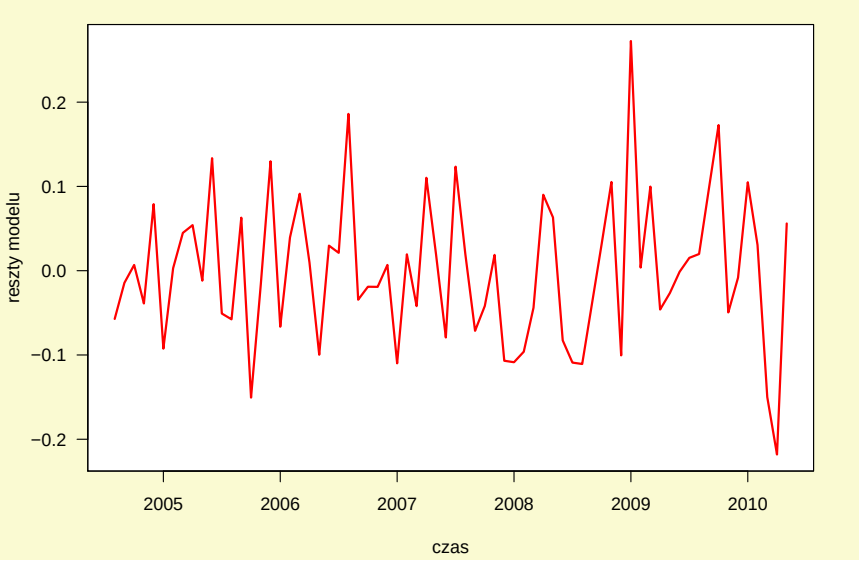
Także test Quenouille'a na poziomoe istotności $\alpha = 0,05$ (przerywane poziome linie — Rysunek 10.9) potwierdza brak zjawiska autokorelacji reszt. W oprogramowaniu R dostępne są też inne testy do badania tego zjawiska np. `bgtest(lmtest)` — test Breuscha-Godfreya oraz `dwtest(lmtest)` — test Durbina-Watsona. Jednak ten ostatni umożliwia badanie tylko autokorelacji rzędu pierwszego.

10.4.3 Heteroskedastyczność procesu resztowego

Ponieważ niejednorodność wariancji jest zjawiskiem nieporządanym należy więc zbadać czy występuje heteroskedastyczność w procesie reszt. Do tego celu można wykorzystać test Breusha-Pagana.

```
# test Koenkera:
> bptest(m.dyn)

        studentized Breusch-Pagan test

data:  m.dyn
BP = 22.7745, df = 17, p-value = 0.1567
# test Breusha-Pagana:
> bptest(m.dyn,studentize=F)

        Breusch-Pagan test

data:  m.dyn
BP = 28.2902, df = 17, p-value = 0.04166
# Cook i Weisberg [1983]:
> bptest(m.dyn,studentize=F,varformula =~fitted(m.dyn))

        Breusch-Pagan test

data:  m.dyn
BP = 1.7564, df = 1, p-value = 0.1851
```

Na podstawie przeprowadzonych testów Breusha-Pagana możemy wnioskować, że występuje homoskedastyczność reszt. Dzięki środowisku R mamy możliwość przeprowadzenia także innych testów np. `hmctest(lmtest)` — test Harrisona-McCabea lub `gqtest(lmtest)` — test Goldfelda-Quandta.



Rysunek 10.10: Reszty modelu.

10.4.4 Stabilność parametrów modelu

Do zbadania stabilności parametrów modelu można wykorzystać test Chowa. Jednak aby móc zastosować ten test należy określić punkt zwrotny, który podzieli cały

analizowany proces na dwie podpróby. Jeśli w tych dwóch podokresach parametry modeli nie będą się różniły, można wtedy przyjąć, że są one stabilne. W teście `sctest(strucchange)` jako punkt zwrotny należy podać ostatnią datę z pierwszej podpróby. Tzn. jeśli za punkt zwrotny przyjmiemy 10.2008 (pierwsza data drugiej podpróby) to dla opcji `point` trzeba podać 09.2008 (ostatnia data pierwszej podpróby).

```
# przygotowanie danych:
> d=cbind(bezrob,t1,t2,t3,month,
lag(bezrob,k=-1),lag(bezrob,k=-2),lag(bezrob,k=-3))
> d=na.omit(d)
# test Chowa:
> library(strucchange)
> sctest(bezrob~., data=d, type="Chow", point=c(2008,9))

        Chow test

data:  bezrob ~ .
F = 5.5201, p-value = 9.785e-06
```

Innym testem, który bada stabilność parametrów modelu jest test CUSUM (cumulated sum of residuals) zwany także testem Harvey'a-Colliera. Jest on dostępny w programie R po uprzednim wczytaniu biblioteki `lmtest`.

```
> harvtest(m.dyn)

        Harvey-Collier test

data:  m.dyn
HC = 0.37, df = 51, p-value = 0.713
```

Na podstawie przeprowadzonego testu Chowa należy odrzucić hipotezę zerową o stabilności parametrów modelu. Z kolei na podstawie testu CUSUM należy stwierdzić, że parametry modelu są stabilne.

10.4.5 Postać analityczna modelu

Aby zweryfikować hipotezę o poprawnym doborze postaci analitycznej modelu można skorzystać z testu RESET — `resettest(lmtest)`. Możemy go przeprowadzić w kilku wersjach:

```
# wartości wyrównane podniesione do potęgi drugiej i trzeciej:
> resettest(m.dyn,power=2:3)

        RESET test

data:  m.dyn
RESET = 0.0605, df1 = 2, df2 = 50, p-value = 0.9414

# wartości wyrównane podniesione do potęgi drugiej:
> resettest(m.dyn,power=2)
```

```
        RESET test

data:  m.dyn
RESET = 0.0598, df1 = 1, df2 = 51, p-value = 0.8078

# wartości wyrównane podniesione do potęgi trzeciej:
> resettest(m.dyn,power=3)

        RESET test

data:  m.dyn
RESET = 0.0468, df1 = 1, df2 = 51, p-value = 0.8297
```

Otrzymane wartości p-value wskazują, że brak jest podstaw do odrzucenia hipotezy zerowej zakładającej poprawną specyfikację modelu.

Rozdział 11

Przegląd wybranych testów statystycznych

11.1 Testy normalności

W celu przedstawienia procedury obliczeniowej testu normalności Doornika-Hansena wykorzystamy dane dotyczące długości płatka kosaćca z gatunku setosa.

```
> attach(iris)
> f= Petal.Length[Species=="setosa"]
```

Przed przystąpieniem do obliczeń statystyki jednowymiarowego testu normalności Doornika-Hansena, należy przekształcić nasz wektor danych `f` w następujący sposób:

```
> ff= f-mean(f)                    # przekształcenie
> n= length(ff)                    # liczebność wektora danych
```

Teraz dokonamy transformacji skośności (D'Agostino) czyli obliczymy statystykę `z1`:

$$\beta = \frac{3(n^2+27n-70)(n+1)(n+3)}{(n-2)(n+5)(n+7)(n+9)} \tag{11.1}$$

$$\omega^2 = -1 + \left[2(\beta-1)\right]^{\frac{1}{2}} \tag{11.2}$$

$$\delta = \frac{1}{\left[\ln\left(\sqrt{\omega^2}\right)\right]^{\frac{1}{2}}} \tag{11.3}$$

$$S_1 = \frac{\frac{1}{n}\cdot\sum_{i=1}^{n}(x_i-\bar{x})^3}{\sqrt{\left(\frac{1}{n}\cdot\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^3}} \tag{11.4}$$

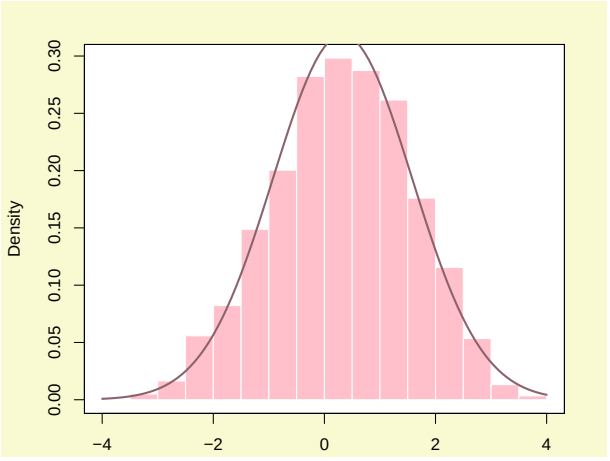
$$y = S_1\left[\frac{\omega^2-1}{2}\cdot\frac{(n+1)(n+3)}{6(n-2)}\right]^{\frac{1}{2}} \tag{11.5}$$

$$z_1 = \delta\ln\left[y+(y^2+1)^{\frac{1}{2}}\right] \tag{11.6}$$

```
# obliczenia dla z1:
> f1= function(x){
 DNAME= deparse(substitute(x))
 x= sort(x[complete.cases(x)])
 n= length(x)
 beta= (3*(n^2+27*n-70)*(n+1)*(n+3)) / ((n-2)*(n+5)*(n+7)*(n+9))
 w2= -1+(2*(beta-1))^(1/2)
 del= 1/sqrt(log(sqrt(w2)))
 S1= e1071::skewness(x,type=1)  # parametr skośności
 y= S1*sqrt(((w2-1) / (2))*(((n+1)*(n+3))/(6*(n-2))))
 z1= del*log(y+sqrt(y^2+1))
}
> z1=f1(ff); z1
[1] 0.3315036
```

Należy zaznaczyć, że statystyka `z1` ma rozkład zbliżony do rozkładu normalnego.

```
# rozkład statystyki z1 dla 10000 replikacji
> m=10000
> statS=NULL
> for (i in 1:m) {
        statS[i] = f1(sample(ff,n,T))
}
# rysunek:
> par(bg="lightgoldenrodyellow")
> hist(statK,prob=T,axes=F,xlab="",ylab="",main="", border="white")
> u=par("usr")
> rect(u[1], u[3], u[2], u[4], col="white")
> par(new=plot)
> hist(statS,prob=T,main="",border="white",col="pink",60)
> curve(dnorm(x,mean(statS),sd(statS)),add=T,lwd=3,col="darkred")
```



Rysunek 11.1: Rozkład statystyki $z1$ dla 10000 replikacji.

Otrzymaną wartość `z1` możemy wykorzystać do zweryfikowania następujących hipotez dotyczących skośności (wzór 11.4):

$$\begin{array}{ll} H_0: & S_1 = 0 \\ H_1: & S_1 \neq 0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & S_1 \geqslant 0 \\ H_1: & S_1 < 0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & S_1 \leqslant 0 \\ H_1: & S_1 > 0 \end{array}$$

```
> 2*(1-pnorm(abs(z1)))  # p-value dla H1: S1!=0
[1] 0.7402641
> pnorm(z1)             # p-value dla H1: S1<0
[1] 0.6298679
> 1-pnorm(z1)           # p-value dla H1: S1>0
[1] 0.3701321
```

Do weryfikacji tego typu hipotez możemy skorzystać z testu skośności D'Agostino wpisując następującą komendę:

```
> library(moments)
> agostino.test(f)

        D'Agostino skewness test

data:  f
skew = 0.1032, z = 0.2185, p-value = 0.827
alternative hypothesis: data have a skewness
```

Wartość statystyki `z` jest inna niż `z1`. Wynika to z tego, że we wzorze 11.3 oraz 11.6 zastosowano logarytm naturalny – ln, natomiast w omawianym teście D'Agostino zastosowano logarytm dziesiętny – log.

W kolejnym kroku obliczymy statystykę `z2` czyli przeprowadzimy transformację kurtozy (Wilson-Hilferty) według poniższych wzorów:

$$\delta = (n-3)(n+1)(n^2+15n-4) \tag{11.7}$$

$$a = \frac{(n-2)(n+5)(n+7)(n^2+27n-70)}{6\delta} \tag{11.8}$$

$$c = \frac{(n-7)(n+5)(n+7)(n^2+2n-5)}{6\delta} \tag{11.9}$$

$$k = \frac{(n+5)(n+7)(n^3+37n^2+11n-313)}{12\delta} \tag{11.10}$$

$$S_1 = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^3}{\sqrt{\left(\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2\right)^3}} \tag{11.11}$$

$$\alpha = a + {S_1}^2 c \tag{11.12}$$

$$K_1 = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2\right)^2} \tag{11.13}$$

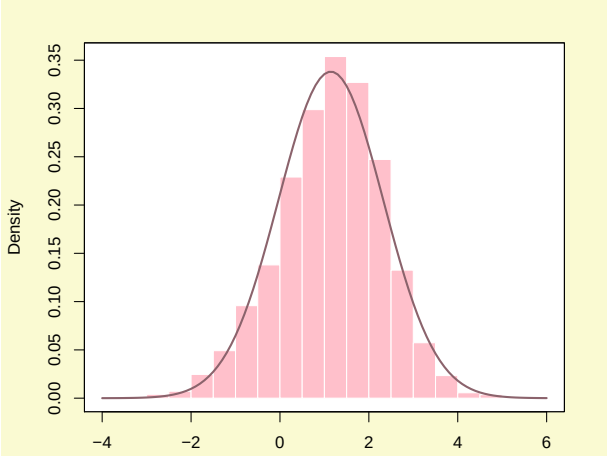
$$\chi = (K_1 - 1 - {S_1}^2)2k \tag{11.14}$$

$$z_2 = \left[\left(\frac{\chi}{2\alpha}\right)^{\frac{1}{3}} - 1 + \frac{1}{9\alpha}\right](9\alpha)^{\frac{1}{2}} \tag{11.15}$$

```
# obliczenia dla z2:
> f2=function(x){
 DNAME= deparse(substitute(x))
 x= sort(x[complete.cases(x)])
 n= length(x)
 delta= (n-3)*(n+1)*(n^2+15*n-4)
 a= ((n-2)*(n+5)*(n+7)*(n^2+27*n-70)) / (6*delta)
 c= ((n-7)*(n+5)*(n+7)*(n^2+2*n-5)) / (6*delta)
 k= ((n+5)*(n+7)*(n^3+37*n^2+11*n-313)) / (12*delta)
 S1= e1071::skewness(x,type=1)          # parametr skośności
 alpha= a+S1^2*c
 K1= e1071::kurtosis(x,type=1)+3        # parametr kurtozy
 chi= (K1-1-S1^2)*2*k
 z2= ( (chi/(2*alpha))^(1/3)-1+(1/(9*alpha)) )*sqrt(9*alpha)
}
> z2=f2(ff); z2
[1] 2.042610
```

Także parametr `z2` ma rozkład zbliżony do rozkładu normalnego.

```
# rozkład statystyki z2 dla 10000 replikacji
> m=10000
> statK = NULL
> for (i in 1:m) {
        statK[i] = f2(sample(ff,n,T))
}
# rysunek:
> par(bg="lightgoldenrodyellow")
> hist(statK,prob=T,axes=F,xlab="",ylab="",main="", border="white")
> u=par("usr")
> rect(u[1], u[3], u[2], u[4], col="white")
> par(new=plot)
> hist(statK,prob=T,main="",border="white",col="pink");box()
> curve(dnorm(x,mean(statK),sd(statK)),add=T,lwd=2,col="pink4")
```



Rysunek 11.2: Rozkład statystyki $z2$ dla 10000 replikacji.

Na podstawie obliczonej wartości `z2` dokonamy wefyfikacji następujących hipotez statystycznych dotyczących kurtozy (wzór 11.13):

$$\begin{array}{ll} H_0: & K_1 = 3 \\ H_1: & K_1 \neq 3 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & K_1 \geqslant 3 \\ H_1: & K_1 < 3 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & K_1 \leqslant 3 \\ H_1: & K_1 > 3 \end{array}$$

```
> 2*(1-pnorm(abs(z2)))  # p-value dla H1: K1!=3
[1] 0.04109101
> pnorm(z2)             # p-value dla H1: K1<3
[1] 0.9794545
> 1-pnorm(z2)           # p-value dla H1: K1>3
[1] 0.02054551
```

W przypadku weryfikacji powyższych hipotez statystycznych możemy zastosować test kurtozy Anscombe-Glynn:

$$a = \frac{3(n-1)}{(n+1)} \tag{11.16}$$

$$b = \frac{24n(n-2)(n-3)}{(n+1)^2(n+3)(n+5)} \tag{11.17}$$

$$c = \frac{6(n^2-5n+2)}{(n+7)(n+9)} \cdot \sqrt{\frac{6(n+3)(n+5)}{n(n-2)(n-3)}} \tag{11.18}$$

$$d = 6 + \frac{8}{c}\left(\frac{2}{c} + \sqrt{1+\frac{4}{c}}\right) \tag{11.19}$$

$$K_1 = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^4}{\left(\frac{1}{n} \cdot \sum_{i=1}^{n}(x_i - \bar{x})^2\right)^2} \tag{11.20}$$

$$\chi = \frac{K_1 - a}{\sqrt{b}} \tag{11.21}$$

$$z = \frac{1 - \frac{2}{9d} - \left(\frac{1-2/d}{1+\chi\cdot\sqrt{2/(d-4)}}\right)^{\frac{1}{3}}}{\sqrt{2/9d}} \tag{11.22}$$

```
> anscombe.test(f)

        Anscombe-Glynn kurtosis test

data:  f
kurt = 3.8046, z = 1.4562, p-value = 0.1453
alternative hypothesis: kurtosis is not equal to 3
```

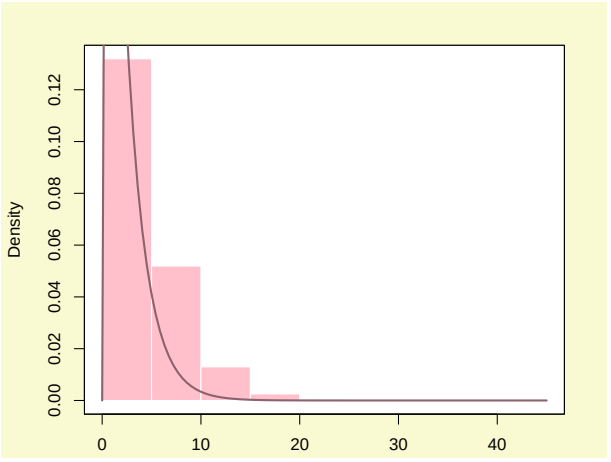
Po wyznaczeniu wartości z_1 oraz z_2 możemy obliczyć statystykę testu normalności *ep* według wzoru:

$$ep = {z_1}^2 + {z_2}^2 \tag{11.23}$$

```
> ep= z1^2+z2^2; ep
[1] 4.282152
```

Ponieważ statystyki `z1` oraz `z2` mają rozkłady zbliżone do normalnych (rysunek 11.1 i 11.2) to suma ich kwadratów będzie miała rozkład χ^2 z dwoma stopniami swobody.

```
# rozkład statystyki ep dla 10000 replikacji
> m=10000
> statC = NULL
> for (i in 1:m) {
        statC[i] = f1(sample(ff,n,T))^2+f2(sample(ff,n,T))^2
}
# rysunek:
> par(bg="lightgoldenrodyellow")
> hist(statK,prob=T,axes=F,xlab="",ylab="",main="", border="white")
> u=par("usr")
> rect(u[1], u[3], u[2], u[4], col="white")
> par(new=plot)
> hist(statC,prob=T,main="",border="white",col="pink");box()
> curve(dchisq(x,2),add=T,lwd=2,col="pink4")
```



Rysunek 11.3: Rozkład statystyki *ep* dla 10000 replikacji.

```
> 1-pchisq(ep,2)
[1] 0.1175283
```

Teraz zostaną przedstawione wyniki testu Doornika-Hansena z wykorzystaniem gotowej funkcji `normality.test1`.

```
> library(normwhn.test)
> normality.test1(as.matrix(f))
[1] "sk"
[1] 0.1031751
[1] "k"
[1] 3.804592
[1] "rtb1"
[1] 0.1031751
[1] "b2"
```

```
[1] 3.804592
[1] "z1"
[1] 0.3315036
[1] "z2"
[1] 2.042610
[1] "H0: data do not have skewness"
[1] "pvalsk"
[1] 0.7402641
[1] "H0: data do not have negative skewness"
[1] "pskneg"
[1] 0.6298679
[1] "H0: data do not have positive skewness"
[1] "pskpos"
[1] 0.3701321
[1] "H0: data do not have kurtosis"
[1] "pvalk"
[1] 0.04109101
[1] "H0: data do not have negative kurtosis"
[1] "pkneg"
[1] 0.9794545
[1] "H0: data do not have positive kurtosis"
[1] "pkpos"
[1] 0.02054551
[1] "H0: data are normally distributed"
[1] "Ep"
         [,1]
[1,] 4.282152
[1] "dof"
[1] 2
[1] "sig.Ep"
          [,1]
[1,] 0.1175283
```

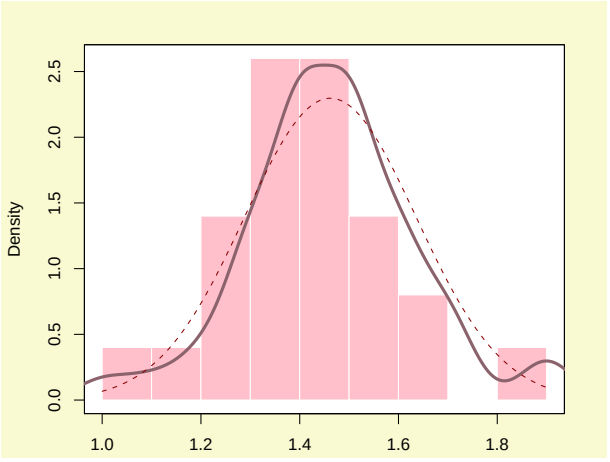
Ponieważ w teście Doornika-Hansena $p - value = 0,1175283$, należy stwierdzić, że zmienna `f` (długość płatka gatunku setosa) charakteryzuje się rozkładem normalnym. Również test skośności ($p - value = 0,7402641$) oraz kurtozy ($p - value = 0,04109101$) potwierdzają naszą decyzję.

```
> shapiro.test(f)

        Shapiro-Wilk normality test

data:  f
W = 0.955, p-value = 0.05481
```

Także test Shapiro-Wilka wskazuje na normalność rozkładu badanej zmiennej ($p - value = 0,05481$).

```
# histogram:
> par(bg="lightgoldenrodyellow")
> hist(statK,prob=T,axes=F,xlab="",ylab="",main="", border="white")
> u=par("usr")
```

```
> rect(u[1], u[3], u[2], u[4], col="white")
> par(new=plot)
> hist(f,prob=T,main="",border="white",col="pink");box()
> lines(density(f),lwd=3,col="pink4")
> curve(dnorm(x,mean(f),sd(f)),add=T,lty=2,col="darkred")
```



Rysunek 11.4: Histogram – długość płatka irysa, gatunek setosa.

Gdy badamy normalność za pomocą kilku testów np. Doornika-Hansena i Shapiro-Wilka zawsze warto porównać ich moc. Poniżej zostały przedstawione obliczenia dotyczące mocy obu testów.

```
# moc testu Doornika-Hansena:
> m= 10000
> statP= NULL
> for (i in 1:m) {
  statP[i] = 1-pchisq(f1(sample(ff,n,T))^2+f2(sample(ff,n,T))^2,2)
  }
> mean(statP< 0.05)
[1] 0.2696
# moc testu Shapiro-Wilka:
> m= 10000
> statSW= NULL
> for (i in 1:m) {
  statSW[i] = shapiro.test(sample(f,n,T))$p.value
  }
> mean(statSW< 0.05)
[1] 0.8699
```

Tak więc test Shapiro-Wilka charakteryzuje się zdecydowanie większą mocą $(0,8699)$ niż test Doornika-Hansena $(0,2696)$. Czyli do oceny normalności wektora danych `f` należy posłużyć się testem Shapiro-Wilka.

Za pomocą funkcji `normality.test1` mamy możliwość także badania wielowymiarowej normalności macierzy danych. Sposób przeprowadzania obliczeń dla wielowymiarowego testu Doornika-Hansena przedstawimy na przykładzie macierzy `X`.

```
# długość i szerokość płatka - setosa:
```

```
> X= iris[1:50,3:4]
> p= length(X[1,])
> n= length(X[,1])
> m= NULL       # wektor średnich dla kolumn macierzy X
> for (i in 1:p) m[i]= mean(X[,i])
```

W pierwszym kroku dokonamy przekształcenia macierzy `X` w poniższy sposób.

$$\check{X}_{p\times n} = \begin{bmatrix} X_{11} - \bar{X}_1 & \dots & X_{p1} - \bar{X}_p \\ \vdots & \ddots & \vdots \\ X_{1n} - \bar{X}_1 & \dots & X_{pn} - \bar{X}_p \end{bmatrix} \tag{11.24}$$

```
> M= matrix(rep(m, n), nrow=p)
> Xhat= X-t(M)
```

$$V = \operatorname{diag}\left({S_{11}}^{-\frac{1}{2}}, \dots, {S_{pp}}^{-\frac{1}{2}}\right) \tag{11.25}$$

```
> V= diag(1/sqrt(diag(cov(X))))
```

$$\Lambda = \operatorname{diag}\left(\lambda_1, \dots, \lambda_n\right) \tag{11.26}$$

```
> lambda= diag(eigen(cor(X))$values)
> H= eigen(cor(X))$vectors
```

$$R^T = H\Lambda^{-\frac{1}{2}}H^T V \check{X}^T \tag{11.27}$$

```
> RT= H %*% solve(lambda)^(1/2) %*% t(H) %*% V %*% t(Xhat)
> R= t(RT)
```

Po otrzymaniu macierzy `R` możemy teraz obliczyć wartości wektorów `z1` i `z2` w podobny sposób jak w przypadku testu jednowymiarowego.

```
# obliczenia dla z1:
> f1= function(x){
DNAME= deparse(substitute(x))
rS1= apply(x,2,FUN=function(x)e1071::skewness(x,type=1))
rK1= apply(x,2,FUN=function(x)e1071::kurtosis(x,type=1)+3)
beta= (3*(n^2+27*n-70)*(n+1)*(n+3)) / ((n-2)*(n+5)*(n+7)*(n+9))
w2= -1+(2*(beta-1))^(1/2)
del= 1/sqrt(log(sqrt(w2)))
y= rS1*sqrt(((w2-1) / (2))*(((n+1)*(n+3))/(6*(n-2))))
z1= del*log(y+sqrt(y^2+1))
}
> z1= f1(R); z1
[1] 0.3200522 3.1503286
# p-value dla jednowymiarowych testów skośności - H1: S1!=0
> 2*(1-pnorm(abs(z1)))
[1] 0.748928767 0.001630869
# obliczenia dla z2:
```

```
> f2= function(x){
delta= (n-3)*(n+1)*(n^2+15*n-4)
a= ((n-2)*(n+5)*(n+7)*(n^2+27*n-70)) / (6*delta)
c= ((n-7)*(n+5)*(n+7)*(n^2+2*n-5)) / (6*delta)
k= ((n+5)*(n+7)*(n^3+37*n^2+11*n-313)) / (12*delta)
rS1= apply(x,2,FUN=function(x)e1071::skewness(x,type=1))
alpha= a+rS1^2*c
rK1= apply(x,2,FUN=function(x)e1071::kurtosis(x,type=1)+3)
chi= (rK1-1-rS1^2)*2*k
z2= ( (chi/(2*alpha))^(1/3)-1+(1/(9*alpha)) )*sqrt(9*alpha)
}
> z2= f2(R); z2
[1]  2.145361 -1.685471
# p-value dla jednowymiarowych testów kurtozy - H1: K1!=3
> 2*(1-pnorm(abs(z2)))
[1] 0.03192402 0.09189779
```

$$Ep = {Z_1}^T Z_1 + {Z_2}^T Z_2 \tag{11.28}$$

```
> z1= matrix(z1,p,1)
> z2= matrix(z2,p,1)
> Ep= t(z1) %*% z1 + t(z2) %*% z2; Ep
         [,1]
[1,] 17.47039
```

$$Ep \longrightarrow {\chi^2}_{df=2p} \tag{11.29}$$

```
> 1-pchisq(Ep,2*p)
[1,] 0.001565663
```

Identyczne wyniki otrzymamy wykonując poniższą komendę.

```
> normality.test1(X)
[1] "sk"
[1] 0.1031751 1.2159276
[1] "k"
[1] 3.804592 4.434317
[1] "rtb1"
[1] 0.09959901 1.14396118
[1] "b2"
[1] 3.872623 4.324673
[1] "z1"
[1] 0.3200522 3.1503286
[1] "z2"
[1]  2.145361 -1.685471
[1] "H0: data do not have skewness"
[1] "pvalsk"
[1] 0.748928767 0.001630869
[1] "H0: data do not have negative skewness"
[1] "pskneg"
```

```
[1] 0.6255356 0.9991846
[1] "H0: data do not have positive skewness"
[1] "pskpos"
[1] 0.3744643837 0.0008154347
[1] "H0: data do not have kurtosis"
[1] "pvalk"
[1] 0.03192402 0.09189779
[1] "H0: data do not have negative kurtosis"
[1] "pkneg"
[1] 0.9840380 0.0459489
[1] "H0: data do not have positive kurtosis"
[1] "pkpos"
[1] 0.01596201 0.95405110
[1] "H0: data are normally distributed"
[1] "Ep"
         [,1]
[1,] 17.47039
[1] "dof"
[1] 4
[1] "sig.Ep"
            [,1]
[1,] 0.001565663
```

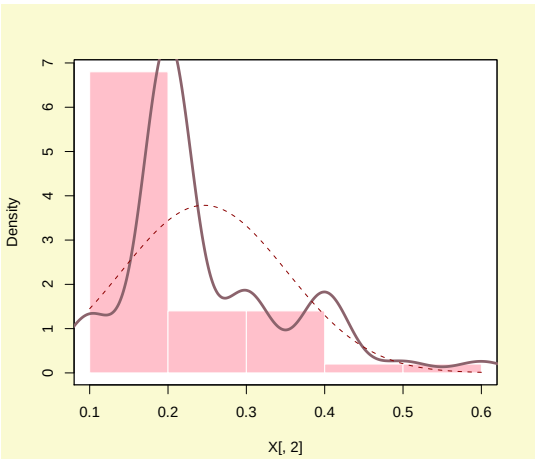
Wartość $p - value = 0,001565663$ jest mniejsza od $\alpha = 0,05$, a więc należy odrzucić hipotezę zerową zakładającą wielowymiarowy rozkład normalny macierzy `X`. Warto zwrócić uwagę na wysoki wskaźnik skośności $(1,2159276)$ dla drugiej zmiennej (szerokość płatka) co może sugerować prawostronną skośność rozkładu.

```
> library(mvnormtest)
> mshapiro.test(t(X))

        Shapiro-Wilk normality test

data:  Z
W = 0.8549, p-value = 2.108e-05
```

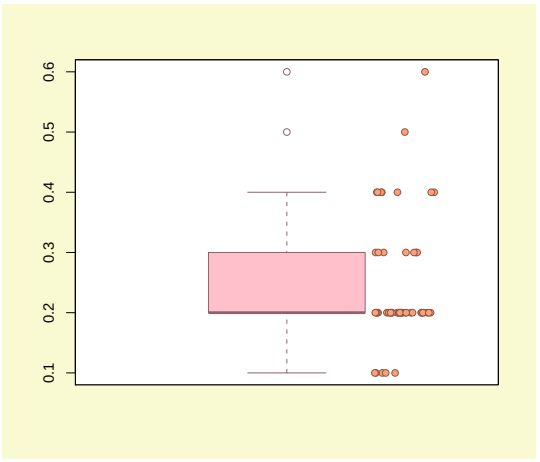
Na podstwie wielowymiarowego testu Shapiro-Wilka także należy odrzcić hipotezę zerową zakładającą dwuwymiarowy rozkład normalny macierzy `X`.

```
# histogram:
> par(bg="lightgoldenrodyellow")
> hist(X[,1],prob=F,axes=F,xlab="",ylab="",main="", col="white")
> u=par("usr")
> rect(u[1], u[3], u[2], u[4], col="white")
> par(new=plot)
> hist(X[,1],prob=T,main="",border="white",col="pink");box()
> lines(density(X[,1]),lwd=3,col="pink4")
> curve(dnorm(x,mean(X[,1]),sd(X[,1])),add=T,lty=2,col="darkred")
# boxplot:
> par(bg="lightgoldenrodyellow")
> boxplot(X[,1],axes=F,xlab="",ylab="",main="", col="white")
> u=par("usr")
```

```
> rect(u[1], u[3], u[2], u[4], col="white")
> par(new=plot)
> boxplot(X[,1],main="",border="pink4",col="pink")
> Gx=jitter(rep(1.3,length(X[,1])),factor=3)
> points(Gx,X[,1],pch=21,bg="lightsalmon",col="lightsalmon4")
```



Rysunek 11.5: Szerokość płatka.



Rysunek 11.6: Szerokość płatka.

W celu porównania obu wielowymiaroych testów normalności obliczymy ich moc.

```
# moc wielowymiarowego testu Doornika-Hansena:
> statP= NULL
> for (i in 1:10000) {
         R1= matrix(c(sample(R[,1],n,T),sample(R[,2],n,T)),n,2)
         z1= matrix(f1(R1))
         z2= matrix(f2(R1))
         Ep= t(z1) %*% z1 + t(z2) %*% z2
  statP[i] = 1-pchisq(Ep, 2*p)
  }
> mean(statP< 0.05)
[1] 0.8908
# moc wielowymiarowego testu Shapiro-Wilka:
> statP= NULL
> for (i in 1:10000) {
         X1= t(matrix(c(sample(X[,1],n,T),sample(X[,2],n,T)),n,2))
  statP[i] = mshapiro.test(X1)$p.value
  }
> mean(statP< 0.05)
[1] 0.8156
```

Tym razem okazało się, że większą moc ma wielowymiarowy test Doornika-Hansena niż wielowymiarowy test Shapiro-Wilka. Różnica między mocą obu testów nie jest zbyt wysoka i wynosi $0,0752$.

Kolejną grupa testów jaką przedstawimy to takie które badają wielowymiarową normalność w oparciu o wielowymiarowy parametr skośności lub kurtozy. Jednym z takich testów jest procedura zaproponowana przez Kanti V. Mardie. Obliczenia dla

tego testu są przedstawione poniżej.

$$J_n = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tag{11.30}$$

```
> Jn= matrix(1, n)
```

$$I_n \tag{11.31}$$

```
> In= matrix(0, n, n)
> diag(In)= 1
```

$$D_{p\times n} = \begin{bmatrix} d_{11} & \dots & d_{p1} \\ \vdots & \ddots & \vdots \\ d_{1n} & \dots & d_{pn} \end{bmatrix} \tag{11.32}$$

```
> Q= In - 1/n * Jn %*% t(Jn)
> X= as.matrix(X)
> D= Q %*% X %*% solve(var(X)) %*% t(X) %*% Q
```

Parametr wielowymiarowej skośności:

$$\hat{S}_1 = \frac{1}{n^2} \sum_{p=1}^{n} \sum_{n=1}^{n} d_{pn}^3 \tag{11.33}$$

```
> S_hat= 1/(n^2) * sum(D^3); S_hat
[1] 1.42242
```

Statystyka testu:

$$\kappa_1 = \frac{n\hat{S}_1}{6} \tag{11.34}$$

```
> kappa1= (n * S_hat)/6; kappa1
[1] 11.8535
```

$$\kappa_1 \longrightarrow \chi^2_{df=p(p+1)(p+2)/6} \tag{11.35}$$

```
> df=p*(p+1)*(p+2)/6; df
[1] 4
> 1-pchisq(kappa1,df)
[1] 0.01847459
```

Parametr wielowymiarowej kurtozy:

$$\hat{K}_1 = \frac{1}{n} \sum_{p=1}^{n} d_{pp}^2 \tag{11.36}$$

```
> K_hat= 1/n * sum(diag(D)^2); K_hat
[1] 9.305538
```

Statystyka testu:

$$\kappa_2 = \frac{\hat{K}_1 - p(p+2)}{\sqrt{8p(p+2/n)}} \tag{11.37}$$

```
> kappa2= (K_hat-p*(p+2)) / (8*p*(p+2)/n)^0.5; kappa2
[1] 1.153943
```

$$\kappa_2 \longrightarrow \mathcal{N}(0,1) \tag{11.38}$$

```
> 2*(1-pnorm(abs(kappa2)))
[1] 0.2485234
```

Wykorzystując funkcję `mult.norm(QuantPsyc)` mamy możliwość uzyskać jednocześnie wszystkie wartości, które zostały obliczone powyżej.

```
> library(QuantPsyc)
# wielowymiarowy test Mardii w oparciu o skośność i kurtozę:
> mult.norm(X)$mult.test
         Beta-hat     kappa      p-val
Skewness 1.422420 11.853500 0.01847459
Kurtosis 9.305538  1.153943 0.24852335
```

W programie R są dostępne także inne funkcje umożliwiające badanie wielowymiarowej normalności w oparciu o parametr skośności `mvnorm.skew.test(ICS)` lub kurtozy `mvnorm.kur.test(ICS)`

```
> library(ICS)
# wielowymiarowy test w oparciu o skośność:
> mvnorm.skew.test(X)

        Multivariate Normality Test Based on Skewness

data:  X
U = 8.2428, df = 2, p-value = 0.01622
# wielowymiarowy test w oparciu o kurtozę:
> mvnorm.kur.test(X)

        Multivariate Normality Test Based on Kurtosis

data:  X
W = 4.6747, w1 = 1.5, df1 = 2.0, w2 = 2.0, df2 = 1.0, p-value = 0.421
```

Badając wielowymiarowy rozkład macierzy `X` w oparciu o parametr skośności należy odrzucić hipotezę zerową (na poziomie istotności $\alpha = 0,05$), która zakłada rozkład normalny. Wartości $p - value$ są równe $0,01847$ (test Mardii) oraz $0,01622$. Z kolei na podstawie testu kurtozy $p-value$ wynoszą: $0,2485$ (test Mardii) i $0,421$, a zatem brak jest podstaw do odrzucenia hipotezy zerowej na poziomie istotności $\alpha = 0,05$.

Moc testów skośności:

```
# moc wielowymiarowego testu skośności (Mardia test):
> statP= NULL
> for (i in 1:10000) {
```

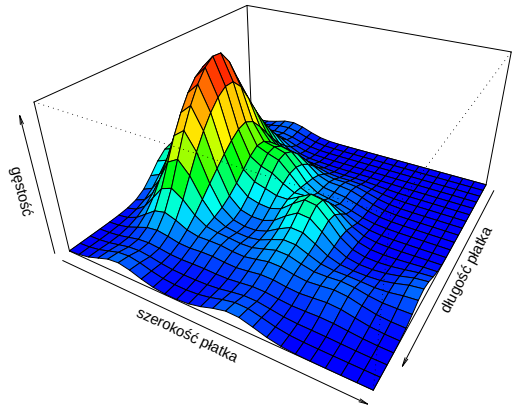
```
+         X1= matrix(c(sample(X[,1],n,T),sample(X[,2],50,T)),n,2)
+ statP[i] = mult.norm(X1)$mult.test[1,3]
+ }
> mean(statP< 0.05)
[1] 0.7239
# moc wielowymiarowego testu skośności:
> statP= NULL
> for (i in 1:10000) {
+         X1= matrix(c(sample(X[,1],n,T),sample(X[,2],50,T)),n,2)
+ statP[i] = mvnorm.skew.test(X1)$p.value
+ }
> mean(statP< 0.05)
[1] 0.6991
```

Moc testów kurtozy:

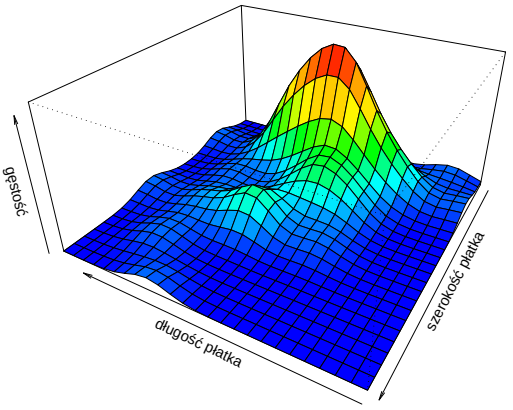
```
# moc wielowymiarowego testu kurtozy (Mardia test):
> statP= NULL
> for (i in 1:10000) {
+         X1= matrix(c(sample(X[,1],n,T),sample(X[,2],50,T)),n,2)
+ statP[i] = mult.norm(X1)$mult.test[2,3]
+ }
> mean(statP< 0.05)
[1] 0.2288
# moc wielowymiarowego testu kurtozy:
> statP= NULL
> for (i in 1:10000) {
+         X1= matrix(c(sample(X[,1],n,T),sample(X[,2],50,T)),n,2)
+ statP[i] = mvnorm.kur.test(X1)$p.value
+ }
> mean(statP< 0.05)
[1] 0.2786
```

Na koniec przedstawimy kilka dwuwymiarowych wykresów gęstości macierzy `X` za pomocą poniższych komend:

```
# przygotowanie zmiennych:
> library(MASS)
> x= kde2d(X[,1],X[,2])$x; y= kde2d(X[,1],X[,2])$y; z=kde2d(X[,1],X[,2])$z
# kolory:
> jet.colors= colorRampPalette(
c("blue", "cyan", "green", "yellow", "orange", "red") )
> nrz= nrow(z); ncz= ncol(z)
> nbcol= 100
> color= jet.colors(nbcol)
> zfacet= z[-1, -1] + z[-1, -ncz] + z[-nrz, -1] + z[-nrz, -ncz]
> facetcol= cut(zfacet, nbcol)
# wykresy - gęstość empiryczna:
> persp(x,y,z, expand= 0.5, col= color[facetcol],xlab= "długość płatka",
ylab= "szerokość płatka",zlab= "gęstość",theta= 210, phi= 30)
> persp(x,y,z, expand= 0.5, col= color[facetcol],xlab= "długość płatka",
ylab= "szerokość płatka",zlab= "gęstość",theta= 120, phi= 30)
```

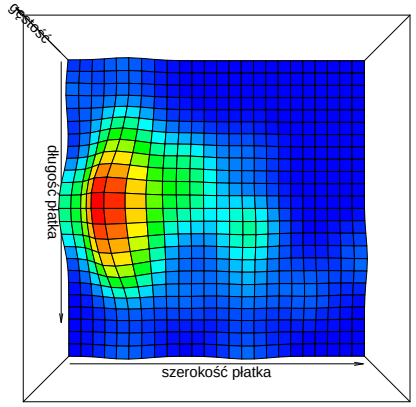


Rysunek 11.7: Gęstość empiryczna.

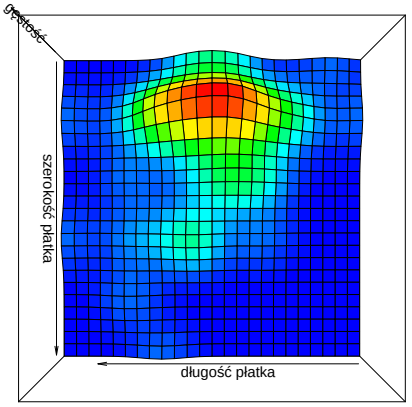


Rysunek 11.8: Gęstość empiryczna.

```
# tzw. mapy ciepła - gęstość empiryczna:
> persp(x,y,z, expand = 0.5, col = color[facetcol],xlab= "długość płatka",
ylab= "szerokość płatka",zlab= "gęstość",theta= 90, phi= 90)
> persp(x,y,z, expand = 0.5, col = color[facetcol],xlab= "długość płatka",
ylab= "szerokość płatka",zlab= "gęstość",theta= 180, phi= 90)
```



Rysunek 11.9: Gęstość empiryczna.

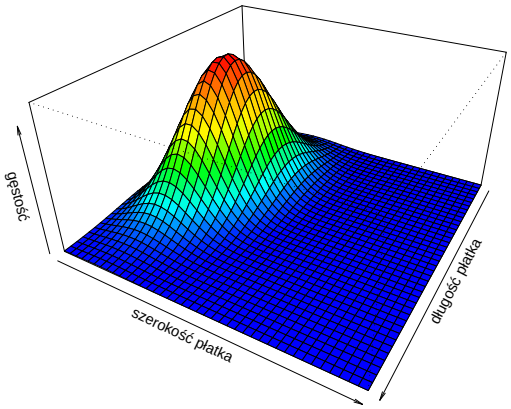


Rysunek 11.10: Gęstość empiryczna.

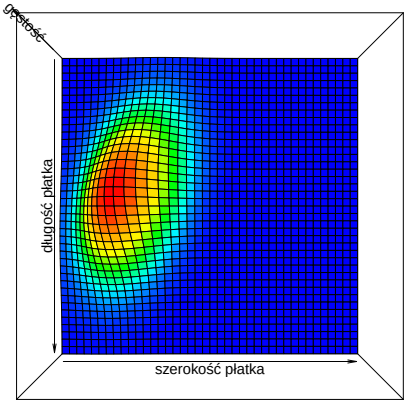
Możemy również przedstawić wykresy gęstości dwuwymiarowego rozkładu normalnego o określonych parametrach obliczonych na podstawie macierzy `X`.

```
> mu1= mean(X[,1])       # średnia dla X[,1]
> mu2= mean(X[,2])       # średnia dla X[,2]
> s11= var(X[,1])        # wariancja dla X[,1]
> s12= cov(X[,1],X[,2]) # kowariancja dla X[,1] i X[,2]
> s22= var(X[,2])        # wariancja dla X[,2]
> rho= cor(X)[1,2]       # korelacja między X[,1] i X[,2]
> x1= seq(1,2,length=41)         # zakres osi x
> x2= seq(0,1,length=41)         # zakres osi y
> f= function(x1,x2)
{
term1= 1/(2*pi*sqrt(s11*s22*(1-rho^2)))
```

```
term2= -1/(2*(1-rho^2))
term3= (x1-mu1)^2/s11
term4= (x2-mu2)^2/s22
term5= -2*rho*((x1-mu1)*(x2-mu2))/(sqrt(s11)*sqrt(s22))
term1*exp(term2*(term3+term4-term5))
}
> z= outer(x1,x2,f)
> persp(x1,x2,z, expand= 0.5, col= color[facetcol], xlab="długość płatka",
ylab= "szerokość płatka",zlab= "gęstość",theta= 120, phi= 30)
> persp(x1,x2,z, expand= 0.5, col= color[facetcol], xlab="długość płatka",
ylab= "szerokość płatka",zlab= "gęstość",theta= 90, phi= 90)
```



Rysunek 11.11: Gęstość teoretyczna.



Rysunek 11.12: Gęstość teoretyczna.

W środowisku R dostępnych jest jeszcze wiele innych testów badających normalność zmiennych. Wymienimy niektóre z nich: wielowymiarowy test Doornika-Hansena – `DH.test(asbio)`,Energy test – `mvnorm.etest(energy)`, Shapiro-Francia – `mvsf(mvsf)`. Warto także wspomnieć o gładkim teście adaptacyjnym Neymana, za pomocą którego mamy możliwość zbadania czy wektor danych pochodzi z rozkładu normalnego – `ddst.norm.test(ddst)`, jednostajnego – `ddst.unifrom.test(ddst)`, wykładniczego – `ddst.exp.test(ddst)` lub z rozkładu wartości ekstremalnych – `ddst.extr.test(ddst)`.

11.2 Testy asymptotyczne

W tym podrozdziale przedstawimy testy asymptotyczne `asymp.test(asympTest)`, które stanowią altenatywę dla takich testów jak: `z.test(BSDA)` oraz `var.test(stats)`. W celu przedstawienia tych testów skorzystamy ze zbioru danych, który dotyczy skuteczności leczenia niewydolności serca z rytmem zatokowym u 6800 pacjentów.

```
> library(asympTest)
> data(DIGdata)
```

Z całego zbioru danych `DIGdata` wyodrębnimy dwie zmienne: `DIABP` – ciśnienie rozkurczliwe oraz zmienną grupującą `TRTMT` – leczenie (0 – placebo, 1 – lek).

```
> t= data.frame(diabp= DIGdata$DIABP, l= DIGdata$TRTMT)
> length(which(is.na(t)))
[1] 5
```

Ponieważ nasz zbiór danych zawiera pięć obserwacji brakujących, zastąpimy je wartością mediany.

```
> library(e1071)
> t= e1071::impute(t,"median")
> t= as.data.frame(t)
```

Na podstawie danych zawartych w tabeli `t` możemy poddać weryfikacji następujące hipotezy:

1. Jedna śrenia:

$$\begin{array}{ll} H_0: & \mu = \mu_0 \\ H_1: & \mu \neq \mu_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & \mu \leqslant \mu_0 \\ H_1: & \mu > \mu_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & \mu \geqslant \mu_0 \\ H_1: & \mu < \mu_0 \end{array}$$

Zweryfikujemy hipotezę zerową która zakłada, że średnie ciśnienie rozkurczowe (u pacjentów którym podano placebo) wynosi 75 mmHg.

$$\begin{array}{ll} H_0: & \mu = 75 \text{ mmHg} \\ H_1: & \mu \neq 75 \text{ mmHg} \end{array}$$

W pierwszej kolejności obliczymy błąd dla śreniej według wzoru:

$$\hat{\sigma}_A(\bar{x}) = \sqrt{\frac{s^2(\bar{x})}{n}} \tag{11.39}$$

gdzie: $s^2(\bar{x}) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$.

```
> seMean(diabp[l==0])
[1] 0.1894600
```

Statystyka testu:

$$S = \frac{\bar{x} - \mu_0}{\hat{\sigma}_A(\bar{x})} \longrightarrow \mathcal{N}(0,1) \tag{11.40}$$

```
> S= (mean(diabp[l==0])-75) / seMean(diabp[l==0])
> S
[1] -0.4622072
# p-value dla H0: u=!75:
> 2*(1-pnorm(abs(S)))
[1] 0.6439328
```

95% przedział ufności dla średniej:

$$\bar{x} - z_{0,025} \cdot \hat{\sigma}_A(\bar{x}) < \mu < \bar{x} + z_{0,025} \cdot \hat{\sigma}_A(\bar{x}) \tag{11.41}$$

```
# dolny przedział ufności:
> mean(diabp[l==0])+qnorm(.025)*seMean(diabp[l==0])
[1] 74.5411
# górny przedział ufności:
> mean(diabp[l==0])-qnorm(.025)*seMean(diabp[l==0])
[1] 75.28377
```

A teraz zastosujemy gotową funkcje:

```
> asymp.test(diabp[l==0],par="mean",conf.level=.95,alt="two.sided",ref=75)

        One-sample asymptotic mean test

data:  diabp[l == 0]
statistic = -0.4622, p-value = 0.6439
alternative hypothesis: true mean is not equal to 75
95 percent confidence interval:
 74.54110 75.28377
sample estimates:
    mean
74.91243
```

Zatem możemy przyjąć, że średnie ciśnienie krwi (u pacjentów którym podano placebo) wynosi 75 mmHg.

Weryfikacja hipotezy $H_0 : \mu = 75$ mmHg dla pacjentów którym podano lek:

```
> asymp.test(diabp[l==1],par="mean",ref=75)$statistic   # statystyka testu
 statistic
-0.5417299
> asymp.test(diabp[l==1],par="mean",ref=75)$p.value     # p-value
[1] 0.5880046
```

Dla pacjentów którym podano lek, także można przyjąć, że średnie ciśnienie krwi wynosi 75 mmHg.

2. Różnica dwóch średnich:

$$\begin{array}{ll} H_0: & d_\mu = d_0 \\ H_1: & d_\mu \neq d_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & d_\mu \leqslant d_0 \\ H_1: & d_\mu > d_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & d_\mu \geqslant d_0 \\ H_1: & d_\mu < d_0 \end{array}$$

Sprawdźmy, czy różnica średnich jest równa 0 mmHg.

$$\begin{array}{ll} H_0: & d_\mu = 0 \text{ mmHg} \\ H_1: & d_\mu \neq 0 \text{ mmHg} \end{array}$$

Błąd dla różnicy średnich:

$$\hat{\sigma}_A(d_{\bar{x}}) = \sqrt{\frac{\hat{\sigma}^2(\bar{x}_{(1)})}{n_{(1)}} + \rho^2 \cdot \frac{\hat{\sigma}^2(\bar{x}_{(2)})}{n_{(2)}}} \tag{11.42}$$

```
> seDMean(diabp[l==0], diabp[l==1], rho=1)
[1] 0.2731128
```

Statystyka testu:

$$S = \frac{d_{\bar{x}} - d_0}{\hat{\sigma}_A(d_{\bar{x}})} \longrightarrow \mathcal{N}(0,1) \tag{11.43}$$

```
> asymp.test(diabp[l==0],diabp[l==1],par="dMean",ref=0)

        Two-sample asymptotic difference of means test

data:  diabp[l == 0] and diabp[l == 1]
statistic = 0.0695, p-value = 0.9446
alternative hypothesis: true difference of means is not equal to 0
95 percent confidence interval:
 -0.5162964  0.5542861
sample estimates:
difference of means
         0.01899482
```

3. Iloraz dwóch średnich:

$$\begin{array}{ll} H_0: & r_\mu = r_0 \\ H_1: & r_\mu \neq r_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & r_\mu \leqslant r_0 \\ H_1: & r_\mu > r_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & r_\mu \geqslant r_0 \\ H_1: & r_\mu < r_0 \end{array}$$

Sprawdźmy, czy iloraz dwóch średnich jest równy 1:

$$\begin{array}{ll} H_0: & r_\mu = 1 \\ H_1: & r_\mu \neq 1 \end{array}$$

Błąd estymacji

$$\hat{\sigma}_A(r_{\bar{x}}) = \frac{1}{|\bar{x}_{(2)}|}\sqrt{\frac{\hat{\sigma}^2(\bar{x}_{(1)})}{n_{(1)}} + r_{\bar{x}}^2 \cdot \frac{\hat{\sigma}^2(\bar{x}_{(2)})}{n_{(2)}}} \tag{11.44}$$

```
> seRMean(diabp[l==0], diabp[l==1])
[1] 0.003647165
```

Statystyka

$$S = \frac{r_{\bar{x}} - r_0}{\hat{\sigma}_A(r_{\bar{x}})} \longrightarrow \mathcal{N}(0,1) \tag{11.45}$$

```
> asymp.test(diabp[l==0],diabp[l==1],par="rMean",ref=1)

        Two-sample asymptotic ratio of means test

data:  diabp[l == 0] and diabp[l == 1]
statistic = 0.0695, p-value = 0.9446
alternative hypothesis: true ratio of means is not equal to 1
95 percent confidence interval:
 0.9931053 1.0074019
sample estimates:
ratio of means
      1.000254
```

Opcje testu `asymp.test` umożliwiają także weryfikację hipotez statystycznych dotyczących wariancji:

1. Jedna wariancja:

$$\begin{array}{ll} H_0: & \sigma^2 = \sigma_0^2 \\ H_1: & \sigma^2 \neq \sigma_0^2 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & \sigma^2 \leqslant \sigma_0^2 \\ H_1: & \sigma^2 > \sigma_0^2 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & \sigma^2 \geqslant \sigma_0^2 \\ H_1: & \sigma^2 < \sigma_0^2 \end{array}$$

Sprawdźmy, czy w grupie pacjentów którym podano placebo wariancja jest równa 120 mmHg:

$$\begin{array}{ll} H_0: & \sigma^2 = 120 \text{ mmHg} \\ H_1: & \sigma^2 \neq 120 \text{ mmHg} \end{array}$$

Błąd estymacji:

$$\hat{\sigma}_A(s^2) = \sqrt{\frac{\sum_{i=1}^{n} \left((x_i - \bar{x})^2 - s^2\right)^2}{n \cdot (n-1)}} \tag{11.46}$$

```
> seVar(diabp[l==0])
[1] 3.288122
```

Statystyka:

$$S = \frac{s^2 - \sigma_0^2}{\hat{\sigma}_A(s^2)} \longrightarrow \mathcal{N}(0,1) \tag{11.47}$$

```
> asymp.test(diabp[l==0],par="var",ref=120)

        One-sample asymptotic variance test

data:  diabp[l == 0]
statistic = 0.6542, p-value = 0.513
alternative hypothesis: true variance is not equal to 120
95 percent confidence interval:
 115.7065 128.5957
sample estimates:
variance
122.1511
```

Teraz powtórzymy ten sam test dla grupy pacjentów którym podano lek:

```
> asymp.test(diabp[l==1],par="var",ref=120)

        One-sample asymptotic variance test

data:  diabp[l == 1]
statistic = 2.2467, p-value = 0.02466
alternative hypothesis: true variance is not equal to 120
95 percent confidence interval:
 121.4609 141.4362
sample estimates:
variance
131.4486
```

2. Różnica dwóch wariancji:

$$\begin{array}{ll} H_0: & d_{\sigma^2} = d_0 \\ H_1: & d_{\sigma^2} \neq d_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & d_{\sigma^2} \leqslant d_0 \\ H_1: & d_{\sigma^2} > d_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & d_{\sigma^2} \geqslant d_0 \\ H_1: & d_{\sigma^2} < d_0 \end{array}$$

Błąd estymacji:

$$\hat{\sigma}_A(d_{s^2}) = \sqrt{\frac{\sum\limits_{i=1}^{n}\left((x_i - \bar{x}_{(1)})^2 - s^2{}_{(1)}\right)^2}{n_{(1)} \cdot (n_{(1)} - 1)} + \rho^2 \cdot \frac{\sum\limits_{i=1}^{n}\left((x_i - \bar{x}_{(2)})^2 - s^2{}_{(2)}\right)^2}{n_{(2)} \cdot (n_{(2)} - 1)}} \tag{11.48}$$

```
> seDVar(diabp[l==0],diabp[l==1])
[1] 6.06459
```

Statystyka:

$$S = \frac{d_{s^2} - d_0}{\hat{\sigma}_A(d_{s^2})} \longrightarrow \mathcal{N}(0,1) \tag{11.49}$$

```
> asymp.test(diabp[l==0],diabp[l==1],par="dVar",ref=0)$statistic
statistic
-1.533085
> asymp.test(diabp[l==0],diabp[l==1],par="dVar",ref=0)$p.value
[1] 0.1252548
```

3. Iloraz dwóch wariancji.

$$\begin{array}{ll} H_0: & r_{\sigma^2} = r_0 \\ H_1: & r_{\sigma^2} \neq r_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & r_{\sigma^2} \leqslant r_0 \\ H_1: & r_{\sigma^2} > r_0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & r_{\sigma^2} \geqslant r_0 \\ H_1: & r_{\sigma^2} < r_0 \end{array}$$

Błąd estymacji:

$$\hat{\sigma}_A(r_{s^2}) = \frac{1}{s^2{}_{(2)}} \sqrt{\frac{\sum\limits_{i=1}^{n}\left((x_i - \bar{x}_{(1)})^2 - s^2{}_{(1)}\right)^2}{n_{(1)} \cdot (n_{(1)} - 1)} + r_{s^2}^2 \cdot \frac{\sum\limits_{i=1}^{n}\left((x_i - \bar{x}_{(2)})^2 - s^2{}_{(2)}\right)^2}{n_{(2)} \cdot (n_{(2)} - 1)}} \tag{11.50}$$

```
> seRVar(diabp[l==0],diabp[l==1])
[1] 0.04385778
```

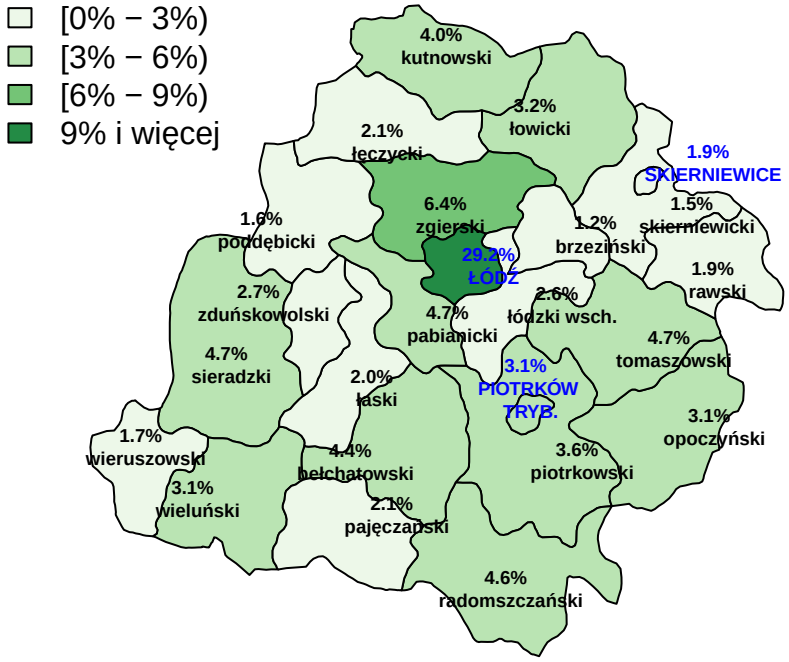
Statystyka:

$$S = \frac{r_{s^2} - r_0}{\hat{\sigma}_A(r_{s^2})} \longrightarrow \mathcal{N}(0,1) \tag{11.51}$$

```
> asymp.test(diabp[l==0],diabp[l==1],par="rVar",ref=1)$statistic
statistic
-1.612743
> asymp.test(diabp[l==0],diabp[l==1],par="rVar",ref=1)$p.value
[1] 0.1068003
```

11.3 Testy dla proporcji

Do prezentacji testów proporcji zostaną wykorzystane dane które są dostępne na stronie internetowej GUS. Dotyczą one liczby ludności w województwie łódzkim w 2009 r. z podziałem na powiaty (rys. 11.13) oraz grupy wiekowe i płeć (rys. 11.14).



Rysunek 11.13: Struktura liczby ludności w województwie łódzkim w 2009r.

Rysunek 11.13 został wygenerowany za pomocą następującego kodu:

```
> library(maptools)
> library("classInt")
> library("RColorBrewer")
> pow=readShapePoly('/home/.../POL_adm2.shp',
  proj4string=CRS("+proj=longlat+ellps=clrk80"))
# udział procentowy ludności w poszczególnych powiatach:
> zmienna=c(6.4, 4.4, 1.2, 4.0, 2.0, 2.1, 29.2, 2.6, 3.2, 3.1, 4.7,
  2.1, 3.6, 3.1, 1.6, 4.6, 1.9, 4.7, 1.5, 1.9, 4.7, 3.1, 1.7, 2.7)
# etykiety:
> NAM=c("6.4% \n zgierski","4.4% \n bełchatowski","1.2% \n brzeziński",
"4.0% \n kutnowski","2.0% \n łaski","2.1% \n łęczycki","29.2% \n ŁÓDŹ",
"2.6% \n łódzki wsch.","3.2% \n łowicki","3.1% \n opoczyński",
"4.7% \n pabianicki","2.1% \n pajęczański","3.6% \n piotrkowski",
"3.1% \n PIOTRKÓW \n TRYB.","1.6% \n poddębicki","4.6% \n radomszczański",
"1.9% \n rawski","4.7% \n sieradzki","1.5% \n skierniewicki",
"1.9% \n SKIERNIEWICE","4.7% \n tomaszowski","3.1% \n wieluński",
"1.7% \n wieruszowski","2.7% \n zduńskowolski")
# kolory:
> przedzialy=4
> kolory=brewer.pal(przedzialy,"Greens")
> klasy=classIntervals(zmienna,przedzialy,style="fixed",
```

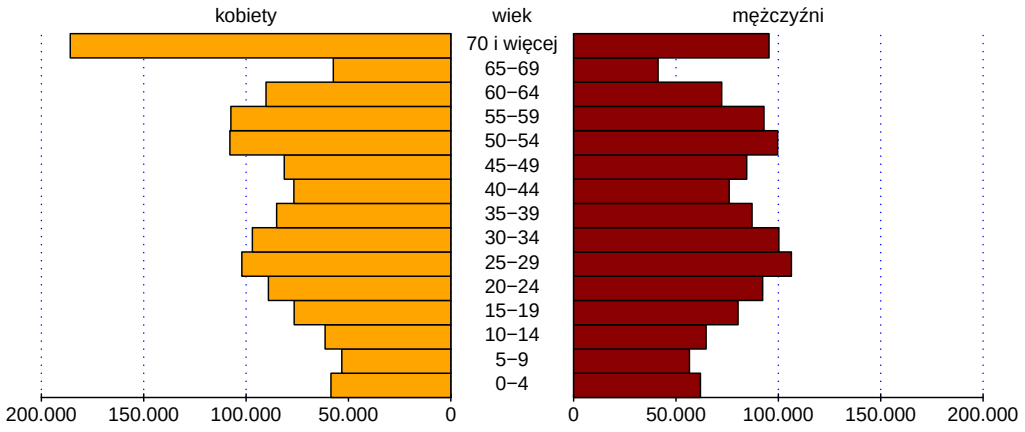
```
  fixedBreaks=c(0, 3, 6, 9, 30))
> tabela.kolorow=findColours(klasy,kolory)
# rysunek:
> plot(pow[c(136,224:246),],col=tabela.kolorow,main="w")
> wsp=coordinates(pow[c(136,224:246),]);colnames(wsp)=c("cx","cy")
> C=rep(c("black","blue","black","blue","black","blue","black"),
  c(6,1,6,1,5,1,4))
> pointLabel(wsp,NAM,cex=.6,col=C,font=2)
> legend("topleft",c("[0% - 3%)", "[3% - 6%)","[6% - 9%)","9% i więcej"),
  fill=attr(tabela.kolorow,"palette"),cex=1,bty="n")
```

Poniżej zostały przedstawione dane dotyczące liczby osób w województwie łódzkim w 2009 r. W tym podziale została uwzględniona również struktura wiekowa.

```
> K=c(58548, 53251, 61402, 76522, 89106, 102074, 96932, 85084, 76663,
  81355, 107911, 107430, 90269, 57402, 185898)
> M=c(61951, 56582, 64722, 80317, 92346, 106389, 100261, 87159, 75991,
  84553, 99678, 92989, 72334, 41261, 95452)
> w=data.frame(K, M)
> rownames(w)=c('0-4','5-9','10-14','15-19','20-24','25-29','30-34',
  '35-39','40-44','45-49','50-54','55-59','60-64','65-69','70 i więcej')
```

Za pomocą poniższego kodu dane zawarte w tabeli `w` zostały przedstawione na rysunku 11.14.

```
> library(pyramid)
> pyramid(w, Llab="kobiety", Rlab="mężczyźni", Clab="wiek",
  Laxis=seq(0,200000,len=5), AxisFM="d", AxisBM=".",
  Lcol="orange", Rcol="red4", Csize=1, Cadj=-0.01)
```

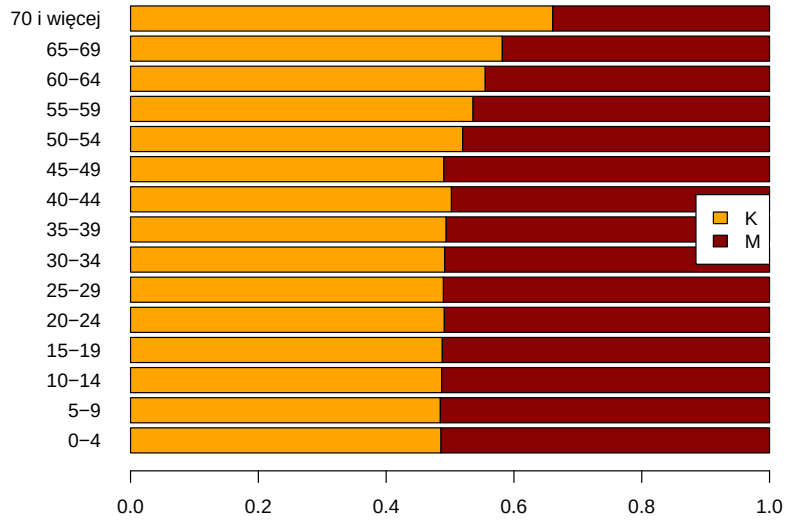


Rysunek 11.14: Liczebności w poszczególnych grupach wiekowych z podziałem na płeć.

Na podstawie danych zamieszczonych w tabeli `w` możemy sądzić, że odsetek kobiet i mężczyzn w grupie wiekowej 40-44 lat może wynosić po 50% (rys. 11.15).

```
> w[9,]
          K     M
40-44 76663 75991
```

```
> par(mar=c(5,5,4,2)+0.1)
> barplot(t(prop.table(as.matrix(w),1)),horiz=T,las=1,
  col=c("orange","red4"))
> legend('right',bg="white",fill=c('orange','red4'),c('k','m'))
```



Rysunek 11.15: Prezentacja proporcji w poszczególnych grupach wiekowych.

A zatem sprawdźmy czy te proporcje są równe. Hipotezy testowe będą następującej postaci:

$$\begin{aligned} H_0: &\quad p = 0,5 \\ H_1: &\quad p \neq 0,5 \end{aligned}$$

Estymacja badanej proporcji:

$$\hat{p} = \frac{m}{n} \tag{11.52}$$

gdzie m - liczba kobiet oraz n - liczba wszystkich osób w grupie wiekowej 40-44.

```
> m= 76663; n= 76663+75991
> hp= m/n
[1] 0.5022011
```

Prawdopodobieństwo badania:

$$\sum_{i=0}^{m} \binom{n}{m} p^m (1-p)^{n-m} \tag{11.53}$$

```
> (1-pbinom(m-1,n,.5))*2
[1] 0.08590797
```

Na podstawie otrzymanego wyniku możemy stwierdzić, że brak jest podstaw do odrzucenia hipotezy zerowej. Tak więc na poziomie istotności $\alpha = 0,05$ przyjmujemy, że proporcje w grupie wiekowej 40-44 lat wynoszą po 50%.

Przedział ufności Cloppera-Pearsona:
– z wykorzystaniem kwantyli rozkładu $\mathcal{F}$:

$$\left(1+\frac{n-m+1}{m\cdot\mathcal{F}_{\alpha/2,\,2m,\,2(n-m+1)}}\right)^{-1},\quad \left(1+\frac{n-m}{(m+1)\cdot\mathcal{F}_{1-\alpha/2,\,2(m+1),\,2(n-m)}}\right)^{-1} \tag{11.54}$$

```
> c( 1/(1+(n-m+1)/(m*qf(0.05/2,2*m,2*(n-m+1)))),
     1/(1+(n-m)/((m+1)*qf(1-0.05/2,2*(m+1),2*(n-m)))) )
[1] 0.4996896 0.5047125
```

– z wykorzystaniem kwantyli rozkładu $\mathcal{B}$:

$$\mathcal{B}_{[1-(1-\alpha)]/2,\,m,\,n-m+1},\quad \mathcal{B}_{[1+(1-\alpha)]/2,\,m+1,\,n-m} \tag{11.55}$$

```
> c( qbeta((1-(1-0.05))/2,m,n-m+1), qbeta((1+(1-0.05))/2,m+1,n-m) )
[1] 0.4996896 0.5047125
```

Wszystkie powyższe obliczenia otrzymamy wykonując poniższą komendę – test dla jednej proporcji:

```
> binom.test(m, n, p=0.5)

        Exact binomial test

data:  76663 and sum(76663, 75991)
number of successes = 76663, number of trials = 152654, p-value =
0.0859
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.4996896 0.5047125
sample estimates:
probability of success
             0.5022011
```

Powyższy test warto stosować w przypadku gdy dysponujemy małymi liczebnościami. Natomiast gdy nasze liczebności są duże możemy wykorzystać inną procedurę. Jest ona przedstawiona poniżej.

Statystyka testu bez korekty:

$$z^2=\left(\frac{\hat{p}-p}{\hat{\sigma}_{p(1)}}\right)^2\longrightarrow\chi^2_{df=1} \tag{11.56}$$

gdzie: $\hat{\sigma}_{p(1)}=\sqrt{\frac{p_1(1-p_1)}{n}}$

```
> prop.test(m, n, correct=F)$statistic
X-squared
 2.958219
```

Przedział ufności Wilsona:

$$\frac{m+z^2}{n+z^2}-z\sqrt{p(1-p)+\frac{z^2}{4n}},\quad \frac{m+z^2}{n+z^2}+z\sqrt{p(1-p)+\frac{z^2}{4n}} \tag{11.57}$$

gdzie: $z=z_{1-\alpha/2}$

```
> prop.test(m, n, correct=F)$conf.int
[1] 0.4996928 0.5047092
attr(,"conf.level")
[1] 0.95
```

Poniżej przedstawiamy wyniki otrzymane za pomocą komendy `prop.test`:

```
> prop.test(m, n, correct=F)

        1-sample proportions test without continuity correction

data:  m out of n, null probability 0.5
X-squared = 2.9582, df = 1, p-value = 0.08544
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.4996928 0.5047092
sample estimates:
        p
0.5022011
```

Gdy pominiemy argument `correct=F` to otrzymamy wyniki testu z korektą. Opcja `correct=T` jest stosowana domyślnie.

Statystyka z korektą:

$$z^2 = \left(\frac{|\hat{p} - p| - \frac{1}{2} \cdot \frac{1}{n}}{\hat{\sigma}_{p(1)}} \right)^2 \longrightarrow \chi^2_{df=1} \tag{11.58}$$

```
> prop.test(m, n, correct=T)$statistic
X-squared
 2.949422
> prop.test(m, n, correct=T)$p.value
[1] 0.0859083
```

Wśrodowisku R jest dostępnych jeszcze wiele innych przedziałow ufności dla jednej proporcji. Wymienimy niektóre z nich:

- przedział ufności Jeffreysa:

$$\mathcal{B}_{\alpha/2,\, m+1/2,\, n-m+1/2}, \quad \mathcal{B}_{1-\alpha/2,\, m+1/2,\, n-m+1/2} \tag{11.59}$$

```
> c( qbeta(0.05/2,m+1/2,n-m+0.5), qbeta(1-0.05/2,m+0.5,n-m+0.5) )
[1] 0.4996928 0.5047092
```

- przedział ufności (asymptotyczny) Walda:

$$\hat{p} - z_{1-\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \quad \hat{p} + z_{1-\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{11.60}$$

```
> c(hp-qnorm(1-0.05/2)*sqrt(hp*(1-hp)/n),
    hp+qnorm(1-0.05/2)*sqrt(hp*(1-hp)/n))
[1] 0.4996929 0.5047092
```

- przedział ufności Agresti-Coull:

$$\check{p} - z_{1-\alpha/2}\sqrt{\frac{\check{p}(1-\check{p})}{\check{n}}}, \quad \check{p} + z_{1-\alpha/2}\sqrt{\frac{\check{p}(1-\check{p})}{\check{n}}} \tag{11.61}$$

gdzie: $\check{p} = \frac{m+z_{\alpha/2}/2}{n+(z_{\alpha/2})^2}$ oraz $\check{n} = n + z_{1-\alpha/2}$

```
> cp=(m+(qnorm(1-0.05/2)^2)/2)/(n+qnorm(1-0.05/2)^2)
> cn=n+qnorm(1-0.05/2)^2
> c(cp-qnorm(1-0.05/2)*sqrt((cp*(1-cp))/cn),
    cp+qnorm(1-0.05/2)*sqrt((cp*(1-cp))/cn))
[1] 0.4996928 0.5047092
```

- przedział ufności arcsine:

$$\sin\left(\arcsin\sqrt{\check{p}} - \frac{z_{1-\alpha/2}}{2\sqrt{n}}\right)^2, \quad \sin\left(\arcsin\sqrt{\check{p}} + \frac{z_{1-\alpha/2}}{2\sqrt{n}}\right)^2 \tag{11.62}$$

gdzie: $\check{p} = \frac{m+0{,}375}{n+0{,}75}$

```
> cp= (m+0.375)/(n+0.75)
> c(sin(asin(sqrt(cp))+qnorm(1-0.05/2)/(2*sqrt(n)))^2,
  sin(asin(sqrt(cp))-qnorm(1-0.05/2)/(2*sqrt(n)))^2)
[1] 0.5047092 0.4996928
```

- przedział ufności logit:

$$\frac{\exp\left(\lambda - z_{1-\alpha/2}\sqrt{\frac{n}{m(n-m)}}\right)}{1-\exp\left(\lambda - z_{1-\alpha/2}\sqrt{\frac{n}{m(n-m)}}\right)}, \quad \frac{\exp\left(\lambda + z_{1-\alpha/2}\sqrt{\frac{n}{m(n-m)}}\right)}{1-\exp\left(\lambda + z_{1-\alpha/2}\sqrt{\frac{n}{m(n-m)}}\right)} \tag{11.63}$$

gdzie: $\lambda = \ln\frac{m}{n-m}$

```
> l= log(m/(n-m))
> c(exp(l-qnorm(1-0.05/2)*sqrt(n/(m*(n-m))))/
    (1+exp(l-qnorm(1-0.05/2)*sqrt(n/(m*(n-m))))),
    exp(l+qnorm(1-0.05/2)*sqrt(n/(m*(n-m))))/
    (1+exp(l+qnorm(1-0.05/2)*sqrt(n/(m*(n-m))))))
[1] 0.4996928 0.5047092
```

Wszystkie omówione powyżej przedziały ufności (oraz kilka innych przedziałów) możemy wyznaczyć dzięki funkcji `binomCI(MKmisc)`:

```
> library(MKmisc)
# przedział ufności logit:
> binomCI(m, n, method= "logit")
$estimate
[1] 0.5022011
$CI
[1] 0.4996928 0.5047092
attr(,"confidence level")
[1] 0.95
```

```
# przedział ufności Wittinga:
> binomCI(m, n, method= "witting", rand=231)
$estimate
[1] 0.5022011
$CI
[1] 0.5000651 0.5043002
attr(,"confidence level")
[1] 0.95
```

W tej części opracowania przedstawimy testy dla dwóch proporcji. Rozważane hipotezy oraz wzory dotyczące omawianego testu (bez korekty i z korektą) są przedstawione poniżej.

$$\begin{array}{ll} H_0: & p_1 - p_2 = 0 \\ H_1: & p_1 - p_2 \neq 0 \end{array} \quad \text{lub} \quad \begin{array}{ll} H_0: & p_1 = p_2 \\ H_1: & p_1 \neq p_2 \end{array}$$

Statystyka testu bez korekty:

$$z^2 = \left(\frac{(\hat{p}_1 - \hat{p}_2) - (p_1 - p_2)}{\hat{\sigma}_{p(1,2)}} \right)^2 \longrightarrow \chi^2_{df=1} \tag{11.64}$$

gdzie: $\hat{\sigma}_{p(1,2)} = \sqrt{\frac{m_1+m_2}{n_1+n_2}\left(1 - \frac{m_1+m_2}{n_1+n_2}\right)\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}$

Przedział ufności:

$$\hat{p}_1 - \hat{p}_2 - z_{1-\alpha/2} \cdot \hat{\sigma}_{p(1-2)}, \quad \hat{p}_1 - \hat{p}_2 + z_{1-\alpha/2} \cdot \hat{\sigma}_{p(1-2)} \tag{11.65}$$

gdzie: $\hat{\sigma}_{p(1-2)} = \sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2}}$

Do przeprowadzenia testu wykorzystamy dane dotyczące odsetka kobiet mieszkających w dwóch powiatach: piotrkowskim i mieście na prawach powiatu Piotrkowie Trybunalskim w grupie wiekowej: 70 lat i więcej - rys. 11.16.

```
# liczba kobiet w grupie wiekowej: 70 lat i więcej:
> k70 =c(10951,5295,1970,7178,3464,3980,64883,4096,5887,5361,8865,
  3856,6262,4922,3024,8276,3200,7734,3081,2676,8679,5521,2557,4180)
# liczba kobiet i mężczyzn w grupie wiekowej: 70 lat i więcej:
> k70o =c(16740,8196,3073,10922,5351,6282,94438,6296,9142,8291,13402,
  6061,9697,7556,4835,12883,5050,12129,4859,4140,13147,8345,3990,6525)
# tabela:
> d= data.frame(k70,k70o)
# nazwy równań tabeli:
> rownames(d)= c('zgierski','bełchatowski','brzeziński','kutnowski',
'łaski','łęczycki','ŁÓDŹ','łódzki wsch.','łowicki','opoczyński',
'pabianicki','pajęczański','piotrkowski','PIOTRKÓW TRYB.','poddębicki',
'radomszczański','rawski','sieradzki','skierniewicki','SKIERNIEWICE',
'tomaszowski','wieluński','wieruszowski','zduńskowolski')
# wybór dwóch badanych powiatów:
> d2=d[13:14,]; d2
               k70 k70o
piotrkowski    6262 9697
PIOTRKÓW TRYB. 4922 7556
```

```
# test dla dwóch proporcji:
> prop.test(d2[,1],d2[,2],correct=F)

        2-sample test for equality of proportions without continuity
        correction

data:  d2[, 1] out of d2[, 2]
X-squared = 0.5916, df = 1, p-value = 0.4418
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.01999111  0.00871886
sample estimates:
   prop 1    prop 2
0.6457667 0.6514029
```

Statystyka z korektą:

$$z^2 = \left(\frac{|\hat{p}_1 - \hat{p}_2| - \frac{1}{2}\left(\frac{1}{n_1} + \frac{1}{n_2}\right) - (p_1 - p_2)}{\hat{\sigma}_{p(1,2)}} \right)^2 \longrightarrow \chi^2_{df=1} \tag{11.66}$$

Przedział ufności:

$$|\hat{p}_1 - \hat{p}_2| - \frac{1}{2}\left(\frac{1}{n_1} + \frac{1}{n_2}\right) z_{1-\alpha/2} \cdot \hat{\sigma}_{p(1-2)}, \quad |\hat{p}_1 - \hat{p}_2| + \frac{1}{2}\left(\frac{1}{n_1} + \frac{1}{n_2}\right) z_{1-\alpha/2} \cdot \hat{\sigma}_{p(1-2)} \tag{11.67}$$

```
> prop.test(d2[,1],d2[,2])

        2-sample test for equality of proportions with continuity
           correction

data:  d2[, 1] out of d2[, 2]
X-squared = 0.5672, df = 1, p-value = 0.4514
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.020108848  0.008836595
sample estimates:
   prop 1    prop 2
0.6457667 0.6514029
```

Na poziomie istotności $\alpha = 0,05$ brak jest podstaw do odrzucenia hipotezy zerowej. Zatem należy stwierdzić, że proporcje są równe.

Za pomocą funkcji `prop.test(stats)` mamy także możliwość przeprowadzać testy dla kilku proporcji. Tym razem sprawdzimy czy odsetki kobiet (w grupie wiekowej 70 lat i więcej, mieszkających w powiatach: wieluńskim, pabianickim oraz tomaszowskim - rys. 11.16) są takie same.

$$\begin{aligned} H_0 &: \ p_1 = p_2 = ... = p_k \\ H_1 &: \ \text{nie wszystkie proporcje są równe} \end{aligned}$$

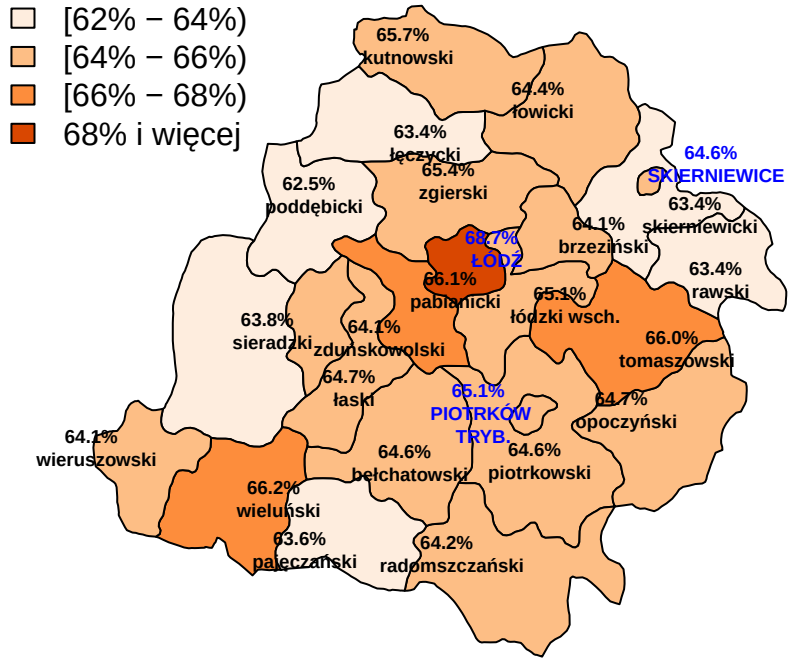
Statystyka testu:

$$c = \sum_{i=1}^{k} \left(\frac{\hat{p}_i - \hat{p}}{\hat{\sigma}_{p(i)}} \right)^2 \longrightarrow \chi^2_{df=k-1} \tag{11.68}$$

gdzie: $\hat{p}_i = \frac{m_i}{n_i}$, $\hat{p} = \sqrt{\frac{\sum_{i=1}^{k} m_i}{\sum_{i=1}^{k} n_i}}$, $\hat{\sigma}_{p(i)} = \sqrt{\frac{\hat{p}(1-\hat{p})}{n_i}}$ dla $i = 1, 2...k$

```
# wybór trzech badanych powiatów:
> d3=d[c(11, 21, 22),]; d3
             k70  k70o
pabianicki  8865 13402
tomaszowski 8679 13147
wieluński   5521  8345
# test dla trzech proporcji:
> prop.test(d3[,1],d3[,2])

        3-sample test for equality of proportions without continuity
        correction

data:  d3[, 1] out of d3[, 2]
X-squared = 0.0686, df = 2, p-value = 0.9663
alternative hypothesis: two.sided
sample estimates:
   prop 1    prop 2    prop 3
0.6614684 0.6601506 0.6615938
```

Także i w tym przypadku brak jest podstaw do odrzucenia hipotezy zerowej. A zatem (na poziomie istotności $\alpha = 0,05$) można przyjąć, że odsetki są równe.



Rysunek 11.16: Wskaźniki struktury liczby kobiet w grupie wiekowej 70 lat i więcej.

Powyższy rysunek został utworzony za pomocą następującego kodu:

```
> pow=readShapePoly('/.../POL_adm2.shp',
  proj4string=CRS("+proj=longlat+ellps=clrk80"))
```

```
> NAM=c("65.4% \n zgierski","64.6% \n bełchatowski","64.1% \n brzeziński",
"65.7% \n kutnowski","64.7% \n łaski","63.4% \n łęczycki","68.7% \n ŁÓDŹ",
"65.1% \n łódzki wsch.","64.4% \n łowicki","64.7% \n opoczyński",
"66.1% \n pabianicki","63.6% \n pajęczański","64.6% \n piotrkowski",
"65.1% \n PIOTRKÓW \n TRYB.","62.5% \n poddębicki",
"64.2% \n radomszczański","63.4% \n rawski","63.8% \n sieradzki",
"63.4% \n skierniewicki","64.6% \n SKIERNIEWICE","66.0% \n tomaszowski",
"66.2% \n wieluński","64.1% \n wieruszowski","64.1% \n zduńskowolski")
> zmienna=c(65.4, 64.6, 64.1, 65.7, 64.7, 63.4, 68.7, 65.1, 64.4, 64.7,
  66.1, 63.6, 64.6, 65.1, 62.5, 64.2, 63.4, 63.8, 63.4, 64.6, 66.0, 66.2,
  64.1, 64.1)
> przedzialy=4
> kolory=brewer.pal(przedzialy,"Oranges")
> klasy=classIntervals(zmienna,przedzialy,style="fixed",
  fixedBreaks=c(62, 64, 66, 68, 70))
> tabela.kolorow=findColours(klasy,kolory)
> plot(pow[c(136,224:246),],col=tabela.kolorow,main="w")
> wsp=coordinates(pow[c(136,224:246),]);colnames(wsp)=c("cx","cy")
> C=rep(c("black","blue","black","blue","black","blue","black"),
  c(6,1,6,1,5,1,4))
> pointLabel(wsp,NAM,cex=.6,col=C,font=2)
> legend("topleft",c("[62% - 64%)", "[64% - 66%)","[66% - 68%)",
  "68% i więcej"),fill=attr(tabela.kolorow,"palette"),cex=1,bty="n")
```

Na zakończenie warto podkreślić, że funkcje `binom.test` oraz `prop.test` mają jeszcze kilka innych opcji:

- `alternative` – test lewostronny (`alt="less"`), prawostronny (`alt="greater"`) oraz dwustronny (`alt="two.sided"`).
- `p` – założona wartość dla proporcji. Domyślna wartość jest równa `0`
- `conf.level` – przedział ufności. Domyślna wartość jest równa `0.95`.

Rozdział 12

Przykład analizy liczby przestępstw w Polsce w 2009 r.

12.1 Wprowadzenie

Do analizy liczby przestępstw w Polsce wykorzystamy dane które są dostępne w bazie danych GUS. Tabela przedstawia liczbę przestępstw stwierdzonych w zakończonych postępowaniach przygotowawczych w 2009 roku.

```
> G
                 krym  gosp zyc_zdr  wol rodz_opie  mien  drog   suma
łódzkie         49457  5870    1957 2765      3191 34622 12362 110224
mazowieckie    103680 14045    3496 4225      5041 81870 21876 234233
małopolskie     62488 11824    2496 4009      2751 47333 10330 141231
śląskie        117557 19481    5050 6670      6853 93888 14934 264433
lubelskie       31104  4854    1604 1849      3086 22119 10011  74627
podkarpackie    23852  4551    1326 1597      1486 17535  7961  58308
podlaskie       16086  2561    1015  921       923 11940  5696  39142
świętokrzyskie  18725  9108     879  921      1267 16267  6180  53347
lubuskie        22182  5506    1073 1628      1109 16262  7403  55163
wielkopolskie   66610 12425    2410 3818      3123 45119 15134 148639
zach.pomorskie  39012  5758    1435 2160      1099 30558 10900  90922
dolnośląskie    77967 11169    2705 4498      3430 60385 15936 176090
opolskie        18582  2648     923 1421      1210 13896  5578  44258
kuj.pom         36815 10337    1396 1883      1669 33400  8931  94431
pomorskie       53998 26185    2437 3511      1723 46273  8341 142468
war.mazur       25482  4943    1506 1498      1805 19794  6575  61603
```

gdzie:

- `krym` – przestępstwa o charakterze kryminalnym,
- `gosp` – przestępstwa o charakterze gospodarczym,
- `zyc_zdr` – przestępstwa przeciwko życiu i zdrowiu,
- `wol` – przestępstwa przeciwko wolności, wolności sumienia i wyznania, wolności seksualnej i obyczjności,

- `rodz_opie` – przestępstwa przeciwko rodzinie i opiece,
- `mien` – przestępstwa przeciwko mieniu,
- `drog` – przestępstwa drogowe.

Zanim przystąpimy do analiz warto przekształcić tabelę `g` w tabelę kontyngencji. Można to zrobić w następujący sposób:

```
> g=G[,-8]
> m=array(
  c(g[,1],g[,2],g[,3],g[,4],g[,5],g[,6],g[,7]), # kolumny tabeli
  dim=c(length(g[,1]),length(g[1,])),           # liczba wierszy, kolumn
  dimnames = list(
  województwo=c(rownames(g)),           # nazwy wierszy
  przestępczość=c(colnames(g))))        # nazwy kolumn
```

W pierwszej kolejności zbadamy czy występuje zależność między województwami a rodzajem przestępstwa. Zrobimy to za pomocą testu niezależności χ^2.

```
> library(vcd)
> assocstats(m)
                    X^2 df P(> X^2)
Likelihood Ratio 46183 90        0
Pearson          50592 90        0

Phi-Coefficient   : 0.168
Contingency Coeff.: 0.166
Cramer's V        : 0.069
```

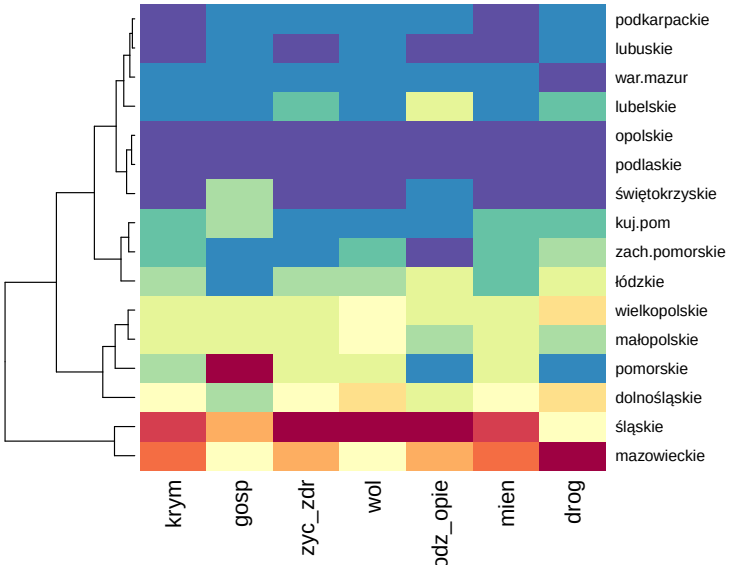
Wartość p-value jest równa 0. Zatem występuje pewna zależność między zmiennymi. Do oceny siły tego związku wykorzystamy współczynniki kontyngencji oraz Cramera. Wynoszą one odpowiednio: $0,166$ i $0,069$ czyli badana zależność nie jest zbyt wysoka.

12.2 Mapy

Gdy tabela kontyngencji jest dość sporych rozmiarów, warto liczebności w niej występujące przedstawić za pomocą różnych funkcji graficznych. Jedną z możliwości jest wykorzystanie komendy `heatmap(stats)` króra rysuje tzw. „mapa ciepła".

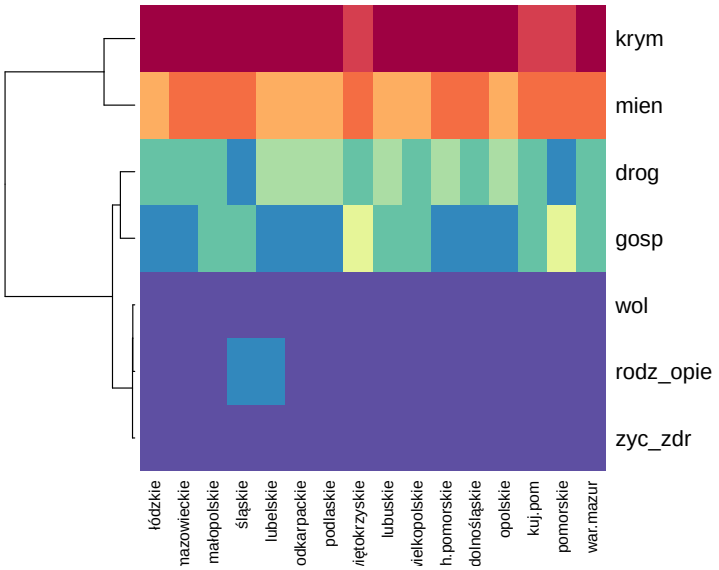
```
> library("RColorBrewer")
> heatmap(as.matrix(g), Colv=NA, scale="column",
  col= rev(brewer.pal(11,"Spectral")))
```

Na podstawie otrzymanej „tzw. mapy ciepła" (rysunek 12.1) możemy wyodrębnić kilka grup województw. W skład pierwszej z nich wchodzą dwa województwa: śląskie i mazowieckie. Charakteryzują się one dużymi liczebnościami przestępstw. Kolejna grupa to: wielkopolskie, małopolskie, pomorskie (duża liczeność przestępstw gospodarczych), dolnośląskie o średniej przestępczości. Ostatnią grupę stanowią głównie województwa ze wschodniej części kraju: podkarpackie, warmińsko-mazurskie, lubelskie, podlaskie, świętokrzyskie oraz lubuskie, zachodnio-pomorskie, łódzkie, kujawsko-pomorskie i opolskie. W tych województwach liczba przestępstw była najmniejsza.



Rysunek 12.1: Mapa ciepła.

```
> heatmap(as.matrix(t(g)), Colv=NA, scale="column",
  col= rev(brewer.pal(11,"Spectral")))
```



Rysunek 12.2: Mapa ciepła.

Po transformacji tabeli g i utworzeniu rysunku 12.2 nasuwają się kolejne wnioski. We wszystkich województwach przestępstwa o charakterze kryminalnym oraz przeciwko mieniu są popełniane najczęściej. Natomiast najrzadziej przestępstwa przeciwko wolności, wolności sumienia i wyznania, wolności seksualnej i obyczjności, rodzinie i opiece a także życiu i zdrowiu.

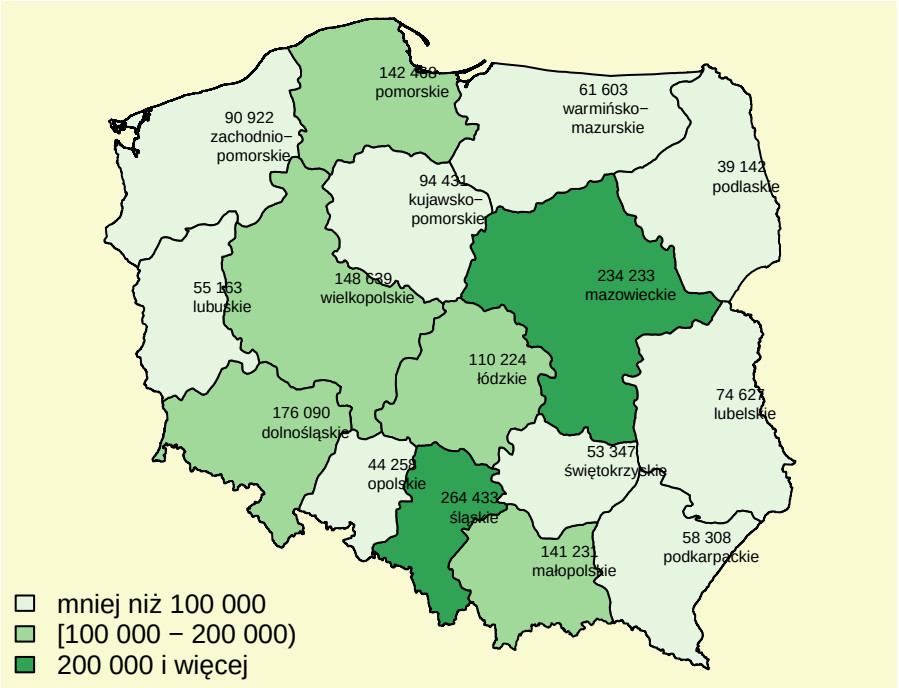
Inną graficzną formą przedstawienia naszych danych jest mapa Polski z podziałem na województwa (rysunki 12.3 i 12.4).

```
# przygotowanie danych:
```

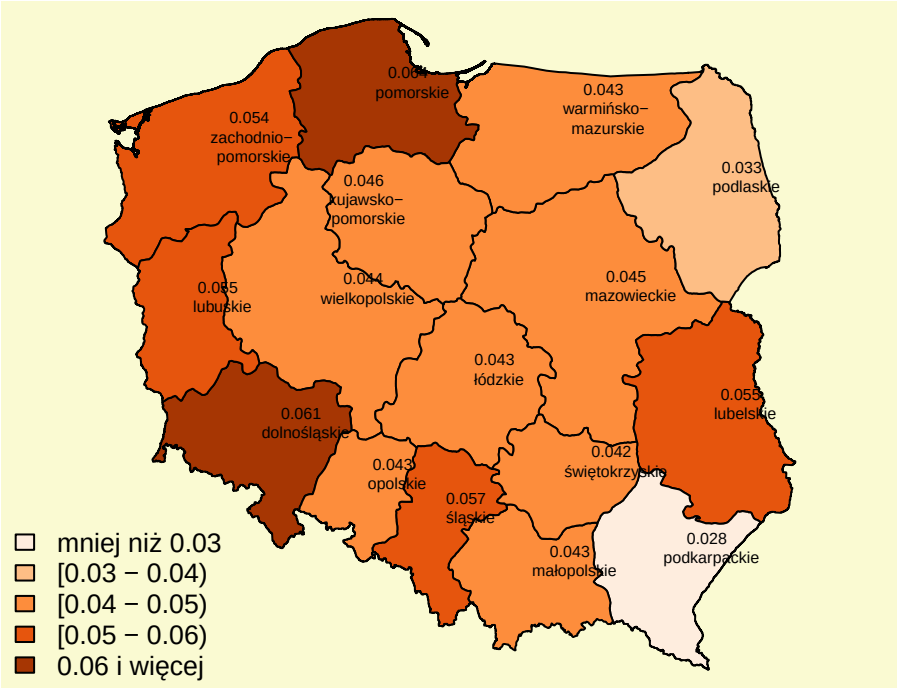
```
> library(maptools)
> woj=readShapePoly("/.../POL_adm1.shp",
  proj4string=CRS("+proj=longlat+ellps=clrk80"))
> library("classInt")
> library("RColorBrewer")
> NAM=c("148 639 \n wielkopolskie","94 431 \n kujawsko- \n pomorskie",
  "141 231 \n małopolskie","110 224 \n łódzkie","176 090 \n dolnośląskie",
  "74 627 \n lubelskie","55 163 \n lubuskie","234 233 \n mazowieckie",
  "44 258 \n opolskie","39 142 \n podlaskie","142 468 \n pomorskie",
 "264 433 \n śląskie","58 308 \n podkarpackie","53 347 \n świętokrzyskie",
 "61 603 \n warmińsko- \n mazurskie","90 922 \n zachodnio- \n pomorskie")
> zmienna=c(148639,94431,141231,110224,176090,74627,55163,234233,44258,
  39142,142468,264433,58308,53347,61603,90922)
> przedzialy=3
> kolory=brewer.pal(przedzialy,"Greens")
> klasy=classIntervals(zmienna,przedzialy,style="fixed",
  fixedBreaks=c(0,100000,200000,300000))
> tabela.kolorow=findColours(klasy,kolory)
# rysunek:
> par(bg = "lightgoldenrodyellow")
> plot(woj)
> u= par("usr"); rect(u[1], u[3], u[2], u[4],col = 'white')
> par(new=plot)
> plot(woj,col=tabela.kolorow,main="w")
> wsp=coordinates(woj)
> colnames(wsp)=c("cx","cy")
> pointLabel(wsp,NAM,cex=.5,col='black')
> legend("bottomleft",
  c("mniej niż 100 000","[100 000 - 200 000)","200 000 i więcej"),
  fill=attr(tabela.kolorow,"palette"),cex=.8,bty="n")
```

```
# liczba mieszkańców w danych województwach w 2009 r:
> l=c(2541832, 5222167, 3298270, 4640725, 2157202, 2101732,
  1189731, 1270120, 1010047, 3408281, 1693198, 2876627,
  1031097, 2069083, 2230099, 1427118)
> names(l)=c('łódzkie','mazowieckie','małopolskie','śląskie',
  'lubelskie','podkarpackie','podlaskie','świętokrzyskie',
  'lubuskie', 'wielkopolskie','zach.pomorskie','dolnośląskie',
  'opolskie','kuj.pom','pomorskie','war.mazur')
# fragment danych:
> T=cbind(na_1=G[,8]/l,na_100000=(G[,8]/l)*100000)
> T[1:3,]
                    na_1 na_100000
łódzkie        0.04336400  4336.400
mazowieckie    0.04485360  4485.360
małopolskie    0.04281972  4281.972
```

W przeliczeniu na jednego mieszkańca (rysunek 12.4) największą liczbę przestępstw odnotowano w dwóch województwach: dolnośląskim i pomorskim. Na 100000 mieszkańców popełniono tam odpowiednio 6121 oraz 6388 czynów zabronionych pra-



Rysunek 12.3: Liczba przestępstw w 2009 r.



Rysunek 12.4: Liczba przestępstw na jednego mieszkańca w 2009 r.

wem. Najmniej takich zachowań zanotowano w województwie podkarpackim: 2774 przestępstw na 100000 mieszkańców.

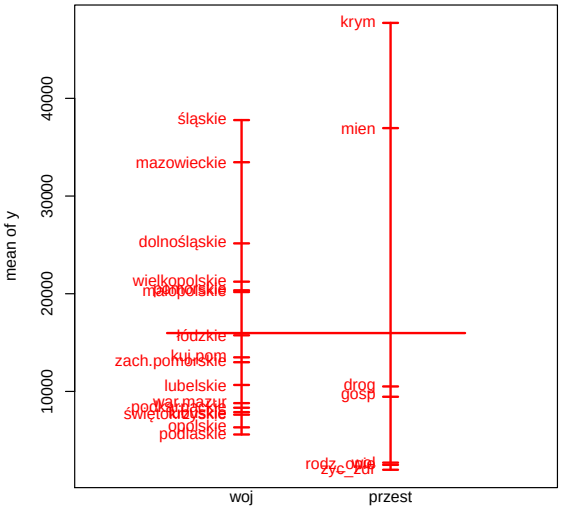
12.3 Analiza wariancji

Analiza wariancji to kolejna metoda którą wykorzystamy w naszych badaniach. Za pomocą komendy **anova** porównamy wartości średnie dla poszczególnych województw oraz kategorii przestępstw.

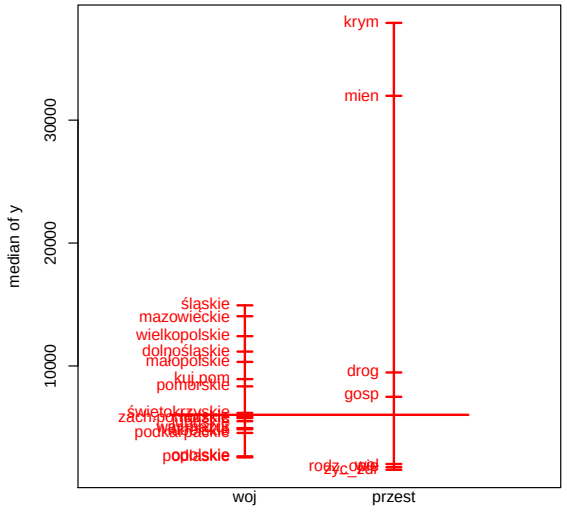
```
> y= c(g[,1],g[,2],g[,3],g[,4],g[,5],g[,6],g[,7])
> woj= rep(rownames(g),7); woj=factor(woj)
> przest= rep(colnames(g),each=16); przest=factor(przest)
> anova(lm(y~woj+przest))
Analysis of Variance Table

Response: y
           Df     Sum Sq    Mean Sq F value    Pr(>F)
woj        15 9.9360e+09  662403078  4.0995  1.18e-05 ***
przest      6 3.3188e+10 5531403342 34.2326 < 2.2e-16 ***
Residuals 90 1.4542e+10  161582725
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Analiza wariancji wskazuje, że średnie liczebności czynów zabronionych dla województw ($p - value = 1,18e - 05$) oraz ich charakteru ($p - value = 2,2e - 16$) różnią się na poziomie istotności $\alpha = 0,05$ (rysunki 12.5).

```
# średnie:
plot.design(data.frame(woj,przest,y),fun=mean,col='red',lwd=2)
# mediany:
plot.design(data.frame(woj,przest,y),fun=median,col='red',lwd=2)
```



Rysunek 12.5: Średnie.



Rysunek 12.6: Mediany.

Wartości średnie dla województw są następujące:

```
> tapply(y,woj,mean)
      śląskie   dolnośląskie        kuj.pom      lubelskie       lubuskie
    37776.143      25155.714      13490.143      10661.000       7880.429
```

```
       łódzkie    małopolskie    mazowieckie       opolskie   podkarpackie
     15746.286      20175.857      33461.857       6322.571       8329.714
     podlaskie      pomorskie świętokrzyskie      war.mazur  wielkopolskie
      5591.714      20352.571       7621.000       8800.429      21234.143
zach.pomorskie
     12988.857
```

Natomiast dla rodzaju przestępstwa:

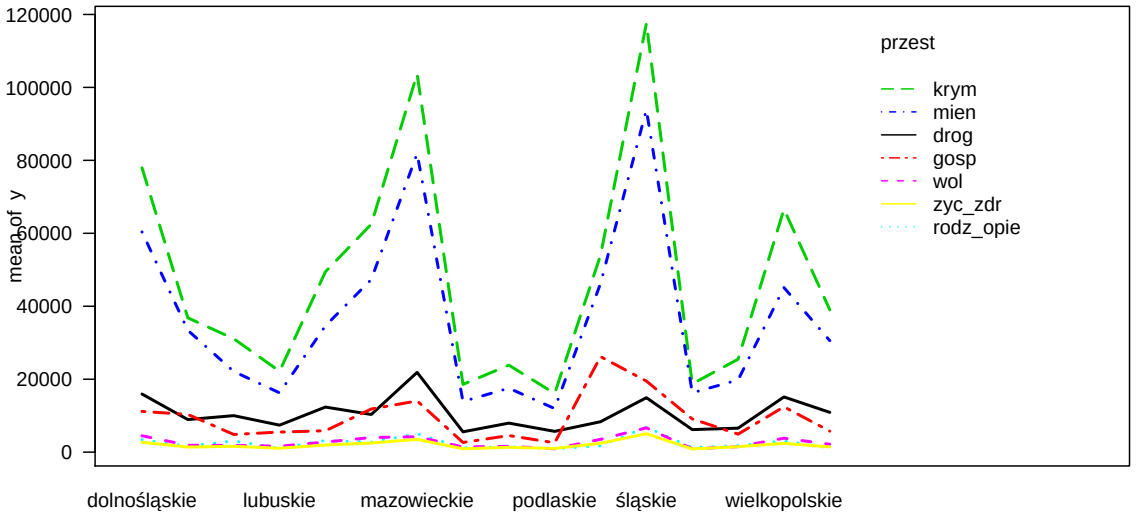
```
> tapply(y,przest,mean)
     krym      drog      gosp      mien rodz_opie       wol   zyc_zdr
47724.812 10509.250  9454.062 36953.812  2485.375  2710.875  1981.750
```

Ponieważ średnie różnią się na poziomie istotności $\alpha = 0,05$, możemy być zainteresowani odpowiedzią na następujące pytanie: Jak duże są różnice między badanymi wartościami w porównaiu z województwem ślaskim oraz przestępczością kryminalną. Odpowiedź na tak postawione pytanie otrzymamy wykonyjąc odpowiednią komendę:

```
# punkt odniesienia: województwo śląskie:
> woj= relevel(woj, ref="śląskie")
# punkt odniesienia: przestępstwa kryminalne:
> przest= relevel(przest, ref="krym")
> library(lmtest)
> coeftest(lm(y~woj+przest))

t test of coefficients:

                  Estimate Std. Error  t value  Pr(>|t|)
(Intercept)        69526.7     5633.8  12.3410 < 2.2e-16 ***
wojdolnośląskie   -12620.4     6794.6  -1.8574 0.0665213 .
wojkuj.pom        -24286.0     6794.6  -3.5743 0.0005671 ***
wojlubelskie      -27115.1     6794.6  -3.9907 0.0001341 ***
wojlubuskie       -29895.7     6794.6  -4.3999 2.967e-05 ***
wojłódzkie        -22029.9     6794.6  -3.2423 0.0016638 **
wojmałopolskie    -17600.3     6794.6  -2.5903 0.0111839 *
wojmazowieckie     -4314.3     6794.6  -0.6350 0.5270663
wojopolskie       -31453.6     6794.6  -4.6292 1.229e-05 ***
wojpodkarpackie   -29446.4     6794.6  -4.3338 3.807e-05 ***
wojpodlaskie      -32184.4     6794.6  -4.7368 8.060e-06 ***
wojpomorskie      -17423.6     6794.6  -2.5643 0.0119944 *
wojświętokrzyskie -30155.1     6794.6  -4.4381 2.566e-05 ***
wojwar.mazur      -28975.7     6794.6  -4.2645 4.933e-05 ***
wojwielkopolskie  -16542.0     6794.6  -2.4346 0.0168824 *
wojzach.pomorskie -24787.3     6794.6  -3.6481 0.0004424 ***
przestdrog        -37215.6     4494.2  -8.2808 1.073e-12 ***
przestgosp        -38270.7     4494.2  -8.5156 3.498e-13 ***
przestmien        -10771.0     4494.2  -2.3966 0.0186146 *
przestrodz_opie   -45239.4     4494.2 -10.0662 < 2.2e-16 ***
przestwol         -45013.9     4494.2 -10.0160 2.646e-16 ***
przestzyc_zdr     -45743.1     4494.2 -10.1782 < 2.2e-16 ***
---
```

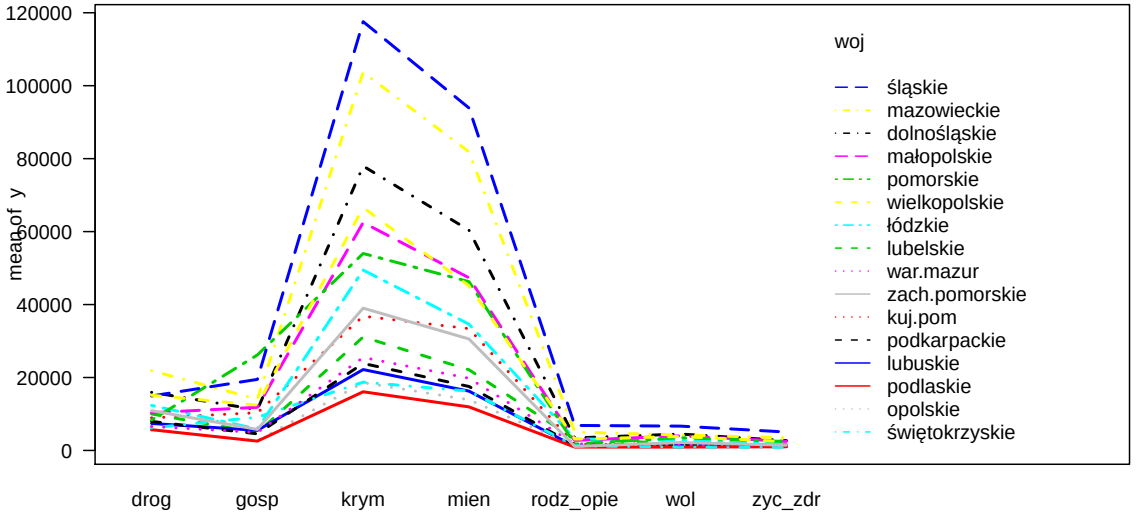
Na podstawie powyższych obliczeń możemy sformułować kilka wniosków. Największa różnica w średnich występuje między województwem śląskim a podlaskim (rysunek 12.5) i wynosi $-32184,4$. Natomiast najmniejsza różnica jest między województwem śląskim a mazowieckim i jest równa $-4314,3$. Jeśli porównamy wartość średnią przestępstw kryminalnych z pozostałymi to dojdziemy do wniosku, że największ różnica występuje między przestępstwami kryminalnymi i przeciwko zdrowiu i życiu a najmniejsza wynosi -10771 i dotyczy przestępstw kryminalnych oraz przeciwko mieniu. Na wykresach 12.7 i 12.8 zostały zaprezentowane różnice w liczebnościach zachowań niezgodnych z prawem.

```
> interaction.plot(woj,przest,y,col=1:7,las=1,lwd=2)
```



Rysunek 12.7: Rozkład liczebności.

```
> interaction.plot(przest,woj,y,col=1:16,las=1,lwd=2)
```



Rysunek 12.8: Rozkład liczebności.

12.4 Modele dla liczebności

Do modelowania liczebności wystąpień określonego zjawiska (np. liczba przestępstw) możemy wykorzystać model Piossona lub ujemny dwumianowy. W pierwszej kolejności zaprezentujemy model Poissona.

```
> pois=glm(y~woj+przest,family=poisson)
> summary(pois)

Call:
glm(formula = y ~ woj + przest, family = poisson)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-68.0044   -8.7900   -0.5255    9.8249  111.2952

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       11.633905   0.002129 5464.68   <2e-16 ***
wojdolnośląskie   -0.406593   0.003076 -132.19   <2e-16 ***
wojkuj.pom        -1.029719   0.003791 -271.62   <2e-16 ***
wojlubelskie      -1.265086   0.004145 -305.20   <2e-16 ***
wojlubuskie       -1.567295   0.004681 -334.84   <2e-16 ***
wojłódzkie        -0.875073   0.003585 -244.07   <2e-16 ***
wojmałopolskie    -0.627191   0.003296 -190.30   <2e-16 ***
wojmazowieckie    -0.121272   0.002837  -42.74   <2e-16 ***
wojopolskie       -1.787552   0.005136 -348.06   <2e-16 ***
wojpodkarpackie   -1.511849   0.004575 -330.45   <2e-16 ***
wojpodlaskie      -1.910392   0.005416 -352.75   <2e-16 ***
wojpomorskie      -0.618470   0.003286 -188.19   <2e-16 ***
wojświętokrzyskie -1.600770   0.004746 -337.27   <2e-16 ***
wojwar.mazur      -1.456877   0.004474 -325.65   <2e-16 ***
wojwielkopolskie  -0.576067   0.003242 -177.70   <2e-16 ***
wojzach.pomorskie -1.067586   0.003844 -277.69   <2e-16 ***
przestdrog        -1.513196   0.002694 -561.73   <2e-16 ***
przestgosp        -1.619007   0.002814 -575.27   <2e-16 ***
przestmien        -0.255783   0.001732 -147.65   <2e-16 ***
przestrodz_opie   -2.955028   0.005144 -574.50   <2e-16 ***
przestwol         -2.868180   0.004936 -581.06   <2e-16 ***
przestzyc_zdr     -3.181471   0.005731 -555.11   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2553250  on 111  degrees of freedom
Residual deviance:   46183  on  90  degrees of freedom
AIC: 47424
# Badanie istotności statystycznej całego modelu:
> lrtest(pois)
```

```
Likelihood ratio test

Model 1: y ~ woj + przest
Model 2: y ~ 1
   Df   LogLik  Df   Chisq Pr(>Chisq)
1  22   -23690
2   1 -1277223 -21 2507066  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Oszacowany model Poissona okazał się istotny statystycznie. Jednak założenie dotyczące równości wariancji i średniej nie zostało spełnione:

$$\begin{aligned} H_0: &\quad \phi = 1 \\ H_1: &\quad \phi \neq 1 \end{aligned}$$

```
> library(AER)
> dispersiontest(pois,alt="two.sided")

        Dispersion test

data:  pois
z = 2.8493, p-value = 0.004382
alternative hypothesis: true dispersion is not equal to 1
sample estimates:
dispersion
  451.6853
```

Tak więc na poziomie istotności $\alpha = 0,05$ odrzucamy hipotezę zerową, która zakłada, że dyspersja (czyli iloraz wariancji i średniej) jest równa 1. A zatem sprawdźmy czy występuje problem nadmiernego rozproszenia (wariancja jest większa od średniej):

$$\begin{aligned} H_0: &\quad \phi \leqslant 1 \\ H_1: &\quad \phi > 1 \end{aligned}$$

```
> dispersiontest(pois)

        Overdispersion test

data:  pois
z = 2.8493, p-value = 0.002191
alternative hypothesis: true dispersion is greater than 1
sample estimates:
dispersion
  451.6853
```

Na poziomie istotności $\alpha = 0,05$ przymujemy hipotezę alternatywną, która zakłada, że występuje nadmierne rozproszenie (overdispersion). W takiej syuacji możemy zastosować model ujemny dwumianowy.

```
> library(MASS)
> nb=glm.nb(y~woj+przest); summary(nb)

Call:
glm.nb(formula = y ~ woj + przest, init.theta = 21.74640825,
    link = log)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.68039  -0.55148  -0.08636   0.54318   3.96886

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       11.57767    0.09512 121.714  < 2e-16 ***
wojdolnośląskie   -0.45311    0.11477  -3.948 7.88e-05 ***
wojkuj.pom        -1.05428    0.11485  -9.180  < 2e-16 ***
wojlubelskie      -1.10428    0.11485  -9.615  < 2e-16 ***
wojlubuskie       -1.46951    0.11493 -12.786  < 2e-16 ***
wojłódzkie        -0.84128    0.11481  -7.327 2.35e-13 ***
wojmałopolskie    -0.63055    0.11479  -5.493 3.95e-08 ***
wojmazowieckie    -0.20431    0.11475  -1.780 0.074998 .
wojopolskie       -1.68246    0.11499 -14.632  < 2e-16 ***
wojpodkarpackie   -1.39677    0.11491 -12.155  < 2e-16 ***
wojpodlaskie      -1.81041    0.11503 -15.739  < 2e-16 ***
wojpomorskie      -0.57400    0.11478  -5.001 5.71e-07 ***
wojświętokrzyskie -1.49467    0.11494 -13.004  < 2e-16 ***
wojwar.mazur      -1.34852    0.11490 -11.736  < 2e-16 ***
wojwielkopolskie  -0.56953    0.11478  -4.962 6.98e-07 ***
wojzach.pomorskie -1.13722    0.11486  -9.901  < 2e-16 ***
przestdrog        -1.38273    0.07587 -18.224  < 2e-16 ***
przestgosp        -1.57685    0.07588 -20.780  < 2e-16 ***
przestmien        -0.25984    0.07584  -3.426 0.000612 ***
przestrodz_opie   -2.91336    0.07604 -38.312  < 2e-16 ***
przestwol         -2.83470    0.07603 -37.285  < 2e-16 ***
przestzyc_zdr     -3.11696    0.07609 -40.963  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for Negative Binomial(21.7464) family taken to be 1)

    Null deviance: 4061.80  on 111  degrees of freedom
Residual deviance:  112.78  on  90  degrees of freedom
AIC: 2002

Number of Fisher Scoring iterations: 1

              Theta:  21.75
          Std. Err.:  2.90
```

```
 2 x log-likelihood:  -1955.994
```

Znając wartość parametru rozproszenia $\theta = 21,7464$ oraz jego błąd standardowy $\sigma_\theta = 2,90$ możemy ocenić jego istotność statystyczną badając następujące hipotezy:

$$H_0: \ \theta = 0$$
$$H_1: \ \theta \neq 0$$

```
> pnorm(2*(1-abs(21.7464/2.9)))
[1] 6.318995e-39
# Badanie istotności statystycznej całego modelu:
> lrtest(nb)
Likelihood ratio test

Model 1: y ~ woj + przest
Model 2: y ~ 1
   Df  LogLik  Df  Chisq Pr(>Chisq)
1  23  -978.0
2   2 -1191.5 -21 426.95  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Model ujemny dwumianowy możemy estymować także z wykorzystaniem funkcji `glm(stats)` oraz opcji `negative.binomial(MASS)`:

```
> th=theta.ml(nb,fitted(nb))
> th
[1] 21.74641
attr(,"SE")
[1] 2.899898
# Estymacja modelu ujemnego dwumianowego:
> n=glm(y~woj+przest,family=negative.binomial(th))
# Porównanie modelu Poissona oraz ujemnego dwumianowego:
> lrtest(pois,nb)
Likelihood ratio test

Model 1: y ~ woj + przest
Model 2: y ~ woj + przest
   Df LogLik Df Chisq Pr(>Chisq)
1  22 -23690
2  23   -978  1 45424  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Przeprowadzony test ilorazu wiarygodności – `lrtest(lmtest)` wykazał, że model ujemny dwumianowy – `nb` ma znacznie wyższy logarytm funkcji wiarygodności. Zatem na podstawie modelu ujemnego dwumianowego należy stwierdzić, że we wszystkich województwach odnotowano mniejszą liczbę przestępstw (ujemne parametry) niż w województwie śląskim. Taka sama sytuacja występuje gdy porównamy wszystkie rodzaje czynów zabronionych z przestępstwami o charakterze kryminalnym. Aby dowiedzieć się jak duże są to różnice należy obliczyć ilorazy wiarygodności.

```
ORpois1=round((exp(coef(pois))-1)*100,4)
ORnb=round((exp(coef(nb))-1)*100,4)
cbind(ORpois1,ORnb)
                         ORpois           ORnb
(Intercept)       11285915.3797 10668768.3273
wojdolnośląskie         -33.4085      -36.4350
wojkuj.pom              -64.2893      -65.1558
wojlubelskie            -71.7785      -66.8550
wojlubuskie             -79.1391      -76.9962
wojłódzkie              -58.3169      -56.8840
wojmałopolskie          -46.5910      -46.7701
wojmazowieckie          -11.4207      -18.4792
wojopolskie             -83.2631      -81.4084
wojpodkarpackie         -77.9498      -75.2606
wojpodlaskie            -85.1978      -83.6413
wojpomorskie            -46.1232      -43.6730
wojświętokrzyskie       -79.8259      -77.5678
wojwar.mazur            -76.7037      -74.0376
wojwielkopolskie        -43.7895      -43.4210
wojzach.pomorskie       -65.6162      -67.9290
przestdrog              -77.9795      -74.9108
przestgosp              -80.1905      -79.3375
przestmien              -22.5690      -22.8826
przestrodz_opie         -94.7923      -94.5707
przestwol               -94.3198      -94.1264
przestzyc_zdr           -95.8475      -95.5708
```

Interpretacja otrzymanych ilorazów wiarygodności na podstawie modelu ujemnego dwumianowego jest następująca:

- Aż o $83,6\%$ mniej popełniono przestępstw w województwie podlaskim niż w województwie śląskim.
- W województwie mazowieckim popełniono tylko o $18,5\%$ mniej przestępstw niż w województwie ślaskim.
- Najmniejsze różnice w liczebności przestępstw w porównaniu z województwem śląskim zanotowano w następujących województwach: dolnośląskim ($-36,4\%$), małopolskim ($-46,8\%$), mazowieckim ($-18,5\%$), pomorskim ($-43,7\%$) i wielkopolskim ($-43,4\%$).
- Z kolei największe różnice zanotowano w województwach: kujawsko-pomorskim ($-65,2\%$), lubelskim ($-66,9\%$), lubuskim (-77%), łódzkim ($-56,9\%$), opolskim ($-81,4\%$), podkarpackim ($-75,3\%$), podlaskim ($-83,6\%$), świętokrzyskim ($-77,6\%$), warmińsko-mazurskim (-74%) oraz zachodnio-pomorskim (-68%).
- Przestępstw przeciwko mieniu popełniono tylko o $22,9\%$ mniej niż kryminalnych.
- Przestępstw o charakterze gospodarczym popełniono o $79,3\%$ mniej niż kryminalnych, natomiast w przypadku pozostałych było ich mniej o ponad 90%.

12.5 Modele dla częstości

Do modelowania częstości wystąpień określonego zjawiska (liczba przestępstw na jednego mieszkańca) wykorzystamy modele Poisona oraz modele logitowe. W pierwszej kolejności zostanie przedstawiony model Poissona w którym zostanie wykorzystana funkcja `offset`.

```
> L=rep(1,7) # liczebności
> W=y/L      # częstości
> pois1=glm(y~woj+przest,family=poisson,offset=log(L)) # model pois1
> summary(pois1)

Call:
glm(formula = y ~ woj + przest, family = poisson, offset = log(L))

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-68.0044   -8.7900   -0.5255    9.8249  111.2952

Coefficients:
                   Estimate Std. Error   z value Pr(>|z|)
(Intercept)       -3.716476   0.002129 -1745.704   <2e-16 ***
wojdolnośląskie    0.071659   0.003076    23.298   <2e-16 ***
wojkuj.pom        -0.221953   0.003791   -58.548   <2e-16 ***
wojlubelskie      -0.499027   0.004145  -120.390   <2e-16 ***
wojlubuskie       -0.042422   0.004681    -9.063   <2e-16 ***
wojłódzkie        -0.273088   0.003585   -76.170   <2e-16 ***
wojmałopolskie    -0.285719   0.003296   -86.692   <2e-16 ***
wojmazowieckie    -0.239313   0.002837   -84.342   <2e-16 ***
wojopolskie       -0.283304   0.005136   -55.163   <2e-16 ***
wojpodkarpackie   -0.719740   0.004575  -157.315   <2e-16 ***
wojpodlaskie      -0.549248   0.005416  -101.418   <2e-16 ***
wojpomorskie       0.114354   0.003286    34.796   <2e-16 ***
wojświętokrzyskie -0.305011   0.004746   -64.264   <2e-16 ***
wojwar.mazur      -0.277664   0.004474   -62.065   <2e-16 ***
wojwielkopolskie  -0.267405   0.003242   -82.486   <2e-16 ***
wojzach.pomorskie -0.059334   0.003844   -15.434   <2e-16 ***
przestdrog        -1.513196   0.002694  -561.726   <2e-16 ***
przestgosp        -1.619007   0.002814  -575.271   <2e-16 ***
przestmien        -0.255783   0.001732  -147.654   <2e-16 ***
przestrodz_opie   -2.955028   0.005144  -574.505   <2e-16 ***
przestwol         -2.868180   0.004936  -581.064   <2e-16 ***
przestzyc_zdr     -3.181471   0.005731  -555.108   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2048950  on 111  degrees of freedom
Residual deviance:   46183  on  90  degrees of freedom
```

```
AIC: 47424

Number of Fisher Scoring iterations: 4
```

W celu poprawnego oszacowania istotności poszczególnych zmiennych (występuje problem nadmiernego rozproszenia) skorzystamy z takiego modelu który uwzględni to zjawisko.

```
# model rpois1 z odpornymi błędami standardowymi:
> rpois1=glm(y~woj+przest,family=quasipoisson,offset=log(L))
> coeftest(rpois1)

z test of coefficients:

                   Estimate Std. Error  z value  Pr(>|z|)
(Intercept)       -3.716476   0.050476 -73.6289 < 2.2e-16 ***
wojdolnośląskie    0.071659   0.072926   0.9826 0.3257866
wojkuj.pom        -0.221953   0.089882  -2.4694 0.0135343 *
wojlubelskie      -0.499027   0.098278  -5.0777 3.820e-07 ***
wojlubuskie       -0.042422   0.110979  -0.3823 0.7022759
wojłódzkie        -0.273088   0.085005  -3.2126 0.0013153 **
wojmałopolskie    -0.285719   0.078142  -3.6564 0.0002558 ***
wojmazowieckie    -0.239313   0.067274  -3.5573 0.0003747 ***
wojopolskie       -0.283304   0.121767  -2.3266 0.0199862 *
wojpodkarpackie   -0.719740   0.108474  -6.6351 3.243e-11 ***
wojpodlaskie      -0.549248   0.128403  -4.2775 1.890e-05 ***
wojpomorskie       0.114354   0.077920   1.4676 0.1422180
wojświętokrzyskie -0.305011   0.112531  -2.7105 0.0067190 **
wojwar.mazur      -0.277664   0.106071  -2.6177 0.0088519 **
wojwielkopolskie  -0.267405   0.076862  -3.4790 0.0005032 ***
wojzach.pomorskie -0.059334   0.091151  -0.6509 0.5150809
przestdrog        -1.513196   0.063869 -23.6920 < 2.2e-16 ***
przestgosp        -1.619007   0.066726 -24.2634 < 2.2e-16 ***
przestmien        -0.255783   0.041072  -6.2276 4.735e-10 ***
przestrodz_opie   -2.955028   0.121952 -24.2310 < 2.2e-16 ***
przestwol         -2.868180   0.117032 -24.5077 < 2.2e-16 ***
przestzyc_zdr     -3.181471   0.135885 -23.4129 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Możemy także zastosować odporne błędy standardowe sandwich w modelu `pois1` które są dostępne w bibliotece `sandwich`.

```
# model pois1 z odpornymi błędami standardowymi sandwich:
> library(sandwich)
> coeftest(pois1,vcov=sandwich)
```

Jak można zauważyć model `pois1` nie oszacował poprawnie wszystkich błędów standardowych poszczególnych zmiennych. Natomiast model `rpois1` (który uwzględnił problem nadmiernego rozproszenia) wykazał kilka zmiennych które nie są statystycznie istotne.

Kolejną grupą modeli które przedstawimy to modele logitowe.

```
# model logit1:
> logit1=glm(cbind(y,L-y)~woj+przest,family=binomial)
> summary(logit1)

Call:
glm(formula = cbind(y, L - y) ~ woj + przest, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-68.4226   -8.7834   -0.4114    9.9514  111.2623

Coefficients:
                   Estimate Std. Error   z value Pr(>|z|)
(Intercept)       -3.692865   0.002150 -1717.792   <2e-16 ***
wojdolnośląskie    0.072989   0.003104    23.511   <2e-16 ***
wojkuj.pom        -0.225510   0.003820   -59.038   <2e-16 ***
wojlubelskie      -0.506041   0.004171  -121.334   <2e-16 ***
wojlubuskie       -0.043165   0.004721    -9.143   <2e-16 ***
wojłódzkie        -0.277357   0.003612   -76.792   <2e-16 ***
wojmałopolskie    -0.290159   0.003320   -87.387   <2e-16 ***
wojmazowieckie    -0.243116   0.002860   -85.011   <2e-16 ***
wojopolskie       -0.287712   0.005172   -55.631   <2e-16 ***
wojpodkarpackie   -0.728895   0.004599  -158.499   <2e-16 ***
wojpodlaskie      -0.556792   0.005446  -102.237   <2e-16 ***
wojpomorskie       0.116523   0.003318    35.118   <2e-16 ***
wojświętokrzyskie -0.309707   0.004779   -64.803   <2e-16 ***
wojwar.mazur      -0.281995   0.004506   -62.586   <2e-16 ***
wojwielkopolskie  -0.271597   0.003266   -83.150   <2e-16 ***
wojzach.pomorskie -0.060365   0.003877   -15.569   <2e-16 ***
przestdrog        -1.529639   0.002704  -565.681   <2e-16 ***
przestgosp        -1.635913   0.002824  -579.262   <2e-16 ***
przestmien        -0.260568   0.001748  -149.044   <2e-16 ***
przestrodz_opie   -2.974984   0.005149  -577.783   <2e-16 ***
przestwol         -2.888038   0.004942  -584.426   <2e-16 ***
przestzyc_zdr     -3.201647   0.005736  -558.160   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2064429  on 111  degrees of freedom
Residual deviance:   46562  on  90  degrees of freedom
AIC: 47802

Number of Fisher Scoring iterations: 4
```

Podobnie jak w przypadku modelu `pois1` model `logit1` wskazuje, że wszystkie zmienne są istotne. Jednak gdy wykorzystamy odporne błędy standardowe (`logit2`) jak w przypadku modelu `rpois1` niektóre z nich okażą się nieistotne statystycznie.

```
# model logit2:
```

```
> logit2=glm(W~woj+przest,family=quasibinomial)
> summary(logit2)

Call:
glm(formula = W ~ woj + przest, family = quasibinomial)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-0.0381820  -0.0062407  -0.0007905   0.0068514   0.0696409

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       -3.71320    0.07066 -52.551  < 2e-16 ***
wojdolnośląskie    0.07295    0.09326   0.782  0.43615
wojkuj.pom        -0.22541    0.10052  -2.243  0.02738 *
wojlubelskie      -0.50584    0.10894  -4.643 1.16e-05 ***
wojlubuskie       -0.04314    0.09589  -0.450  0.65386
wojłódzkie        -0.27724    0.10195  -2.719  0.00785 **
wojmałopolskie    -0.29003    0.10231  -2.835  0.00566 **
wojmazowieckie    -0.24301    0.10100  -2.406  0.01817 *
wojopolskie       -0.28759    0.10224  -2.813  0.00603 **
wojpodkarpackie   -0.72864    0.11691  -6.233 1.45e-08 ***
wojpodlaskie      -0.55658    0.11065  -5.030 2.49e-06 ***
wojpomorskie       0.11646    0.09234   1.261  0.21047
wojświętokrzyskie -0.30957    0.10287  -3.009  0.00340 **
wojwar.mazur      -0.28187    0.10208  -2.761  0.00698 **
wojwielkopolskie  -0.27148    0.10179  -2.667  0.00907 **
wojzach.pomorskie -0.06034    0.09630  -0.627  0.53257
przestdrog        -1.42112    0.06483 -21.922  < 2e-16 ***
przestgosp        -1.55866    0.06851 -22.750  < 2e-16 ***
przestmien        -0.25994    0.04375  -5.941 5.26e-08 ***
przestrodz_opie   -2.95441    0.12770 -23.135  < 2e-16 ***
przestwol         -2.84778    0.12144 -23.451  < 2e-16 ***
przestzyc_zdr     -3.14558    0.13986 -22.491  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for quasibinomial family taken to be 0.0002521464)

    Null deviance: 0.819052  on 111  degrees of freedom
Residual deviance: 0.021215  on  90  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 10
```

Tak samo jak w przypadku modelu `pois1` wykorzystamy odporne błędy standardowe sandwich w modelu `logit1`.

```
# model logit1 z odpornymi błędami standardowymi sandwich:
> coeftest(logit1,vcov=sandwich)
```

Wyznaczymy teraz ilorazy wiarygodności dla modeli `pois1`, `logit1` i `logit2`.

```
> ORpois1=round((exp(coef(pois1))-1)*100,4)
> ORlogit1=round((exp(coef(logit1))-1)*100,4)
> ORlogit2=round((exp(coef(logit2))-1)*100,4)
> cbind(ORpois,ORlogit1,ORlogit2)
                   ORpois1 ORlogit1 ORlogit2
(Intercept)       -97.5680 -97.5099 -97.5601
wojdolnośląskie     7.4289   7.5719   7.5678
wojkuj.pom        -19.9047 -20.1891 -20.1811
wojlubelskie      -39.2879 -39.7122 -39.7002
wojlubuskie        -4.1535  -4.2246  -4.2226
wojłódzkie        -23.8974 -24.2216 -24.2125
wojmałopolskie    -24.8526 -25.1855 -25.1761
wojmazowieckie    -21.2832 -21.5820 -21.5735
wojopolskie       -24.6710 -25.0023 -24.9929
wojpodkarpackie   -51.3121 -51.7558 -51.7433
wojpodlaskie      -42.2616 -42.6956 -42.6833
wojpomorskie       12.1149  12.3583  12.3514
wojświętokrzyskie -26.2885 -26.6338 -26.6241
wojwar.mazur      -24.2449 -24.5723 -24.5630
wojwielkopolskie  -23.4637 -23.7839 -23.7748
wojzach.pomorskie  -5.7608  -5.8579  -5.8551
przestdrog        -77.9795 -78.3386 -75.8557
przestgosp        -80.1905 -80.5226 -78.9582
przestmien        -22.5690 -22.9386 -22.8899
przestrodz_opie   -94.7923 -94.8952 -94.7891
przestwol         -94.3198 -94.4315 -94.2027
przestzyc_zdr     -95.8475 -95.9305 -95.6958
```

Wnioski (na podstawie modelu `logit2`) dotyczące ilości przestępstw przypadających na jednego mieszkańca są następujące:

- Liczba przestępstw przypadająca na jednego mieszkańca województwa pomorskiego była większa o $12,4\%$ niż w województwie śląskim. Natomiast najmniejsza ($-51,7\%$) w województwie podkarpackim.
- Liczba przestępstw przeciwko mieniu przypadająca na jednego mieszkańca była o $22,9\%$ mniejsza niż liczba przestępstw kryminalnych. Z kolei liczba przestępstw przeciwko życiu i zdrowiu była aż o $95,7\%$ mniejsza niż liczba przestępstw kryminalnych w przeliczeniu na jednego mieszkańca.

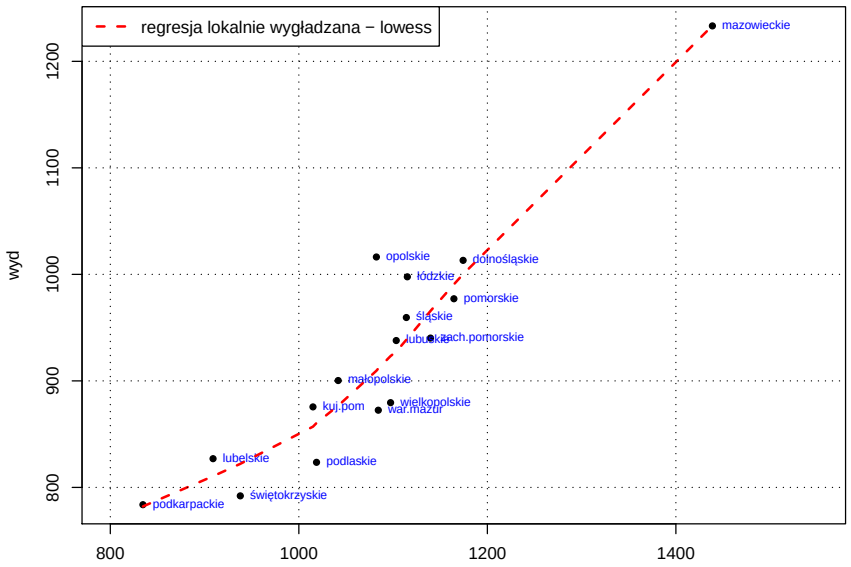
Rozdział 13

Modele regresji

13.1 Wprowadzenie

W tym opracowaniu zostanie przedstawionych kilka modeli regresji oraz metody za pomocą których można oszacować ich parametry. Dane jakie zostaną wykorzystane do budowy modeli regresji dotyczą przeciętnych miesięcznych wydatków i dochodów przypadających na jedną osobę w danym województwie. Informacje zostały zebrane i opublikowane przez GUS od 37302 gospodarstw domowych w 2009 roku.

```
# wydatki:
> wyd= c(997.73, 1233.28, 900.35, 959.51, 826.98, 783.78, 823.56, 792.07,
  937.9, 879.57, 940.2, 1013.17, 1016.33, 875.59, 977.11, 872.5)
# dochody:
> doch= c(1115.1, 1438.73, 1041.73, 1114.05, 908.99, 834.59, 1018.77,
  937.89, 1103.32, 1097.33, 1139.67, 1174.26, 1082.27, 1014.99, 1164.50,
  1084.28)
# liczba gospodarstw domowych objętych badaniem:
> gd=c(2735,5366,3104,4381,2205,1932,1208,1258,989,3143,1627,2939,1017,
  1965,2039,1394)
> t= data.frame(doch,wyd)
> rownames(t)= c('łódzkie','mazowieckie','małopolskie','śląskie',
  'lubelskie','podkarpackie','podlaskie','świętokrzyskie','lubuskie',
  'wielkopolskie','zach.pomorskie','dolnośląskie','opolskie','kuj.pom',
  'pomorskie','war.mazur')
# wykres korelacyjny:
> plot(doch,wyd,cex=1.2,pch=20,xlim=c(800,1550))
> text(doch,wyd,pos=4,rownames(t),cex=0.7,col="blue")
> lines(lowess(t),lwd=2,lty=2,col="red")
> grid(col="black")
> legend("topleft",c("regresja lokalnie wygładzana - lowess"),lty=2,lwd=2,
  col="red",bg="white")
```



Rysunek 13.1: Wykres korelacyjny.

13.2 Estymacja modelu liniowego

13.2.1 Metoda najmniejszych kwadratów

W programie R do estymacji modeli liniowych służy funkcja `lm(stats)`. Kryterium optymalizacji dla metody najmniejszych kwadratów jest przedstawione poniżej (wzory 13.1a oraz 13.1b).

$$\sum_{i=1}^{n} e_i^2 \rightarrow \min \tag{13.1a}$$

$$\sum_{i=1}^{n} (y_i - \alpha_0 - \alpha_1 x_{1i})^2 \rightarrow \min \tag{13.1b}$$

```
# model z wyrazem wolnym:
> mnk=lm(wyd~doch,data=t)
> summary(mnk)

Call:
lm(formula = wyd ~ doch)

Residuals:
     Min       1Q   Median       3Q      Max
-61.0675 -27.7128   0.3736  30.3636  87.2743

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 96.73935   90.08070   1.074    0.301
doch         0.76905    0.08285   9.282 2.33e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 43.1 on 14 degrees of freedom
Multiple R-squared: 0.8602,     Adjusted R-squared: 0.8502
F-statistic: 86.15 on 1 and 14 DF,  p-value: 2.332e-07
```

Ponieważ wyraz wolny nie jest istotny statystycznie ($p - value = 0,301$) zostanie on usunięty z modelu.

```
# model bez wyrazu wolnego:
> mnk0=lm(wyd~doch+0,data=t)
> summary(mnk0)

Call:
lm(formula = wyd ~ doch + 0)

Residuals:
    Min      1Q  Median      3Q     Max
-61.267 -25.223   2.034  15.802  88.405

Coefficients:
     Estimate Std. Error t value Pr(>|t|)
doch 0.857387   0.009962   86.07   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 43.32 on 15 degrees of freedom
Multiple R-squared: 0.998,      Adjusted R-squared: 0.9978
F-statistic:  7407 on 1 and 15 DF,  p-value: < 2.2e-16
```

Usunięcie wyrazu wolnego spowodowało podwyższenie skorygowanego współczynnika determinacji który teraz jest równy $0,998$. A więc model jest lepiej dopasowany. Potwierdza to także kryterium informacyjne AIC oraz BIC.

$$AIC = n + n\ln(2\pi) + n\ln\left(\frac{RSS}{n}\right) + 3 \cdot k \tag{13.2}$$

```
# kryterium AIC dla modelu z wyrazem wolnym:
> k= 2; n= length(doch)
> aic= n + n*log(2*pi) + n * log(sum(resid(mnk)^2) / n) + 3 * k
> aic
[1] 169.7045
# kryterium AIC dla modelu bez wyrazu wolnego:
> aic0= n + n*log(2*pi) + n * log(sum(resid(mnk0)^2) / n) + 2 * k
> aic0
[1] 168.9711
# z wykorzystaniem funkcji AIC oraz parametru k:
> AIC(mnk,mnk0,k=2)
     df      AIC
mnk   3 169.7045
mnk0  2 168.9711
```

```
# kryterium BIC dla modelu z wyrazem wolnym:
```

```
> k= log(n)
> bic= n + n*log(2*pi) + n * log(sum(resid(mnk)^2) / n) + 3 * k
> bic
[1] 172.0223
# kryterium BIC dla modelu bez wyrazu wolnego:
> bic0= n + n*log(2*pi) + n * log(sum(resid(mnk0)^2) / n) + 2 * k
> bic0
[1] 170.5162
# z wykorzystaniem funkcji AIC oraz parametru k:
> AIC(mnk,mnk0,k=log(n))
     df      AIC
mnk   3 172.0223
mnk0  2 170.5162
```

Porównamy teraz modele za pomocą funkcji `anova(stats)`. Badane hipotezy mają następującą postać:

$$\begin{aligned} H_0: &\quad y_i = 0,857387x_{1i} \\ H_1: &\quad y_i = 0,76905x_{1i} + 96,73935 \end{aligned} \tag{13.3}$$

```
# test oparty na statystyce F:
> anova(mnk0,mnk)
Analysis of Variance Table

Model 1: wyd ~ doch + 0
Model 2: wyd ~ doch
  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1     15 28153
2     14 26010  1    2142.7 1.1533  0.301
# test oparty na statystyce chi-kwadrat:
> anova(mnk0,mnk, test="Chisq")
Analysis of Variance Table

Model 1: wyd ~ doch + 0
Model 2: wyd ~ doch
  Res.Df   RSS Df Sum of Sq P(>|Chi|)
1     15 28153
2     14 26010  1    2142.7    0.2829
```

Takie same wyniki otrzymamy wykorzystując funkcję badającą restrykcje nałożone na parametry modelu liniowego. Czyli zweryfikujemy hipotezę zerową (wzór 13.4) o nieistotności wyrazu wolnego.

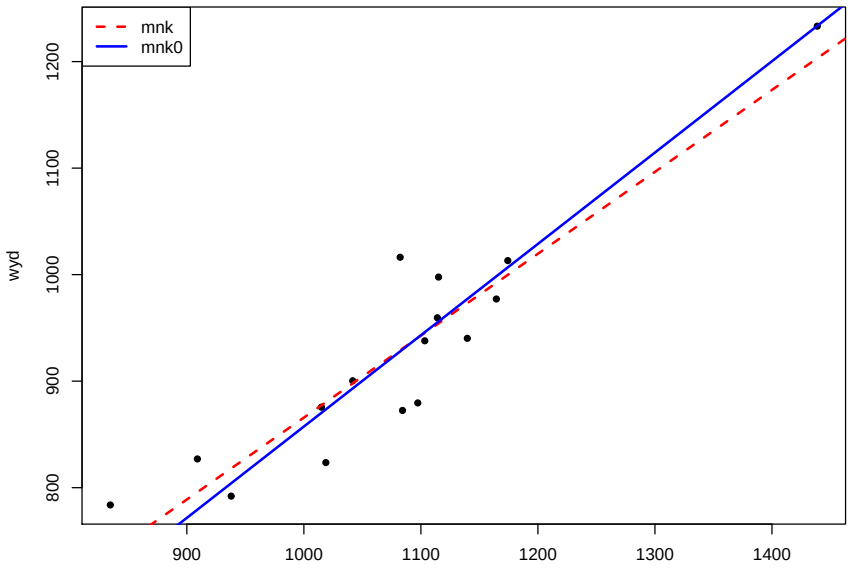
$$\begin{aligned} H_0: &\quad \alpha_0 = 0 \\ H_1: &\quad \alpha_0 \neq 0 \end{aligned} \tag{13.4}$$

```
> library(car)
# test oparty na statystyce F:
> linearHypothesis(mnk, c("(Intercept) = 0"))
Linear hypothesis test
```

```
Hypothesis:
(Intercept) = 0

Model 1: restricted model
Model 2: wyd ~ doch

  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1     15 28153
2     14 26010  1    2142.7 1.1533  0.301
# test oparty na statystyce chi-kwadrat:
> linearHypothesis(mnk, c("(Intercept) = 0"),test="Chisq")
Linear hypothesis test

Hypothesis:
(Intercept) = 0

Model 1: restricted model
Model 2: wyd ~ doch

  Res.Df   RSS Df Sum of Sq  Chisq Pr(>Chisq)
1     15 28153
2     14 26010  1    2142.7 1.1533     0.2829
```

Tak więc powyższe testy potwierdziły nasz wniosek, że lepszym modelem jest regresja liniowa bez wyrazu wolnego.

```
> plot(doch,wyd,cex=1.2,pch=20)
> abline(mnk,col='red',lwd=2,lty=2); abline(mnk0,col='blue',lwd=2)
> legend("topleft",c("mnk","mnk0"),col=c("red","blue"),lty=c(2,1),lwd=2)
```



Rysunek 13.2: Regresja liniowa.

13.2.2 Poprawność specyfikacji modelu

W tym podrozdziale omówimy dwa testy badające poprawną postać analityczną modelu: test RESET oraz test Rainbow. W programie R są one dostępne w paczce `lmtest`. W teście RESET należy oszacować model pomocniczy o równaniu:

$$y_i = \alpha_1 x_{1i} + \alpha_2 \hat{y_i}^2 + \alpha_3 \hat{y_i}^3 \tag{13.5}$$

```
> reset23=lm(wyd~doch+I(fitted(mnk0)^2)+I(fitted(mnk0)^3)+0)
```

a następnie zweryfikować hipotezę zerową o postaci:

$$H_0 : \begin{bmatrix} \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad H_1 : \begin{bmatrix} \alpha_2 \\ \alpha_3 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{13.6}$$

Do weryfikacji rozważanej hipotezy statystycznej (wzór 13.6) wykorzystamy funkcję `linearHypothesis(car)`. Umożliwia ona testowanie restrykcji nałożonych na parametry modelu liniowego.

```
> linearHypothesis(reset23,
  c("I(fitted(mnk0)^2)= 0","I(fitted(mnk0)^3)= 0"), test="F")
Linear hypothesis test

Hypothesis:
I(fitted(mnk0)^2) = 0
I(fitted(mnk0)^3) = 0

Model 1: restricted model
Model 2: wyd ~ doch + I(fitted(mnk0)^2) + I(fitted(mnk0)^3) + 0

  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1     15 28153
2     13 23446  2    4706.7 1.3049 0.3045
```

```
> library(lmtest)
> resettest(mnk0, power=2:3, type= "fitted")

        RESET test

data:  mnk0
RESET = 1.3049, df1 = 2, df2 = 13, p-value = 0.3045
```

Na podstawie testu RESET stwierdzamy, że brak jest podstaw do odrzucenia hipotezy zerowej zakładającej liniowość modelu. Poniżej zostaną przedstawione także inne rodzaje tego testu.

$$y_i = \alpha_1 x_{1i} + \alpha_2 \hat{y_i}^2 \tag{13.7}$$

```
> resettest(mnk0, power=2, type= "fitted")

        RESET test

data:  mnk0
RESET = 0.6134, df1 = 1, df2 = 14, p-value = 0.4465
```

$$y_i = \alpha_1 x_{1i} + \alpha_2 {x_i}^3 \tag{13.8}$$

```
> resettest(mnk0,power=3,type="regressor")

        RESET test

data:  mnk0
RESET = 0.4233, df1 = 1, df2 = 14, p-value = 0.5259
```

Kolejnym testem który przedstawimy to test Rainbow. Służy on także do badania postaci analitycznej modelu. Poniżej zostanie zaprezentowana jego procedura obliczeniowa.

```
> k=1           # liczba parametrów modelu razem z wyrazem wolnym
> n=length(doch)
> doch1=doch[order(doch)]       # sortowanie niemalejąco doch
> wyd1= wyd[order(doch)]        # sortowanie niemalejąco wyd względem doch
```

Z tak posortowanych danych wybieramy 50% (frakcja) środkowych (centrum) obserwacji:

```
> fraction= 0.5         # frakcja: 50 procent obserwacji czyli 8
> center= 0.5           # centrum podziału: 50 procent
> i= 1:n            # indeksy zmiennych
# początkowy indeks dla zmiennych doch1 i wyd1:
> from= ceiling(quantile(i, probs = (center - fraction/2)))
> from
25%
  5
# ostatni indeks dla zmiennych doch1 i wyd1:
> to= from + floor(fraction * n) - 1
> to
25%
 12
> sy= wyd1[from:to]     # podpróba dla wyd1 o indeksach od 5 do 12
> sx= doch1[from:to]    # podpróba dla doch1 o indeksach od 5 do 12
> n1= length(sy)        # liczebność podpróby
> e= resid(mnk0)        # reszty dla badanego modelu mnk0
> r= resid(lm(sy~sx+0)) # reszty dla modelu z podpróby
> ssr= sum(e^2)       # suma kwadratów reszt dla mnk0
> ssr1= sum(r^2)      # suma kwadratów reszt dla modelu z podpróby
```

Statystyka testu:

$$RAIN = \frac{\left(\sum_{i=1}^{n} e_i^2 - \sum_{i=1}^{n_1} r_i^2\right)/(n - n_1)}{\sum_{i=1}^{n_1} r_i^2/(n_1 - k)} \tag{13.9}$$

```
> rain= ((ssr - ssr1)/(n - n1))/(ssr1/(n1 - k))
> rain
[1] 0.4166865
> df= c((n - n1), (n1 - k))
> 1-pf(rain, df[1], df[2])
[1] 0.8783834
```

```
> raintest(mnk0,order.by=~doch,fraction=.5,center=.5)

        Rainbow test

data:  mnk0
Rain = 0.4167, df1 = 8, df2 = 7, p-value = 0.8784
```

13.2.3 Normalność

W R jest dostępnych wiele testów badających normalność rozkładu reszt. W tym opracowaniu zostanie przedstawiony jeden z nich. Test Andersona-Darlinga oraz szereg innych testów normalności możemy znaleźć w bibliotece `nortest`. Wzór na obliczenie statystyki tego testu jest przdstawiony poniżej.

$$A = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\Big(\ln(z_i) + \ln(1 - z_{n+1-i})\Big) \tag{13.10}$$

```
> z= pnorm(sort(e),mean(e),sd(e))
> n= length(e)
> i= 1:n
> A= -n-( (sum((2*i-1)*(log(z)+log(1-z[n+1-i]))))/n )
> A
[1] 0.3253687
```

Wzór poprawki na wielkość póby dla rozkładu normalnego jest podany poniżej:

$$A1 = \left(1 + \frac{0,75}{n} + \frac{2,25}{n^2}\right) \cdot A \tag{13.11}$$

```
> A1= (1+(0.75/n)+(2.25/n^2))*A
> A1
[1] 0.34348
```

Ponieważ wartość krytyczna dla poziomu istotności $\alpha = 0,05$ wynosi $A^* = 0,752$ (tabela 13.1) i jest większa od $A1 = 0,34348$ to należy stwierdzić, że jest brak podstaw do odrzucenia hipotezy zerowej zakładającej normalny rozkład reszt.

Tabela 13.1: Wartości krytyczne

α	10%	5%	2,5%	1%
A^*	0,631	0,752	0,873	1,035

W zależności od otrzymanej wartości $A1$ (wzór 13.11) należy użyć jednego z poniższych algorytmów w celu wyznaczenia wartości $p - value$:

- jeżeli $A1 < 0,2$ to:

$$p - value = 1 - \exp(-13,436 + 101,14 \cdot A1 - 223,73 \cdot A1^2) \tag{13.12a}$$

- jeżeli $0,2 \leqslant A1 < 0,34$ to:

$$p - value = 1 - \exp(-8,318 + 42,796 \cdot A1 - 59,938 \cdot A1^2) \tag{13.12b}$$

- jeżeli $0,34 \leqslant A1 < 0,6$ to:

$$p - value = \exp(0,9177 - 4,279 \cdot A1 - 1,38 \cdot A1^2) \tag{13.12c}$$

- jeżeli $A1 \geqslant 0,6$ to:

$$p - value = \exp(1,2937 - 5,709 \cdot A1 + 0,0186 \cdot A1^2) \tag{13.12d}$$

W omawianym przykładzie wartość statystyki $A1$ jest równa $0,34348$ a zatem należy skorzystać ze wzoru 13.12c.

```
> p.value= exp(0.9177 - 4.279 * A1 - 1.38 * A1^2 )
> p.value
[1] 0.48926
```

Teraz wykorzystamy gotową funkcje `ad.test`:

```
> library(nortest)
> ad.test(e)

        Anderson-Darling normality test

data:  e
A = 0.3254, p-value = 0.4893
```

Także na podstawie adaptacyjnego testu Neymana możemy sądzić, że reszty mają rozkład normalny. Ten test jest dostępny w paczce `ddst`.

```
> library(ddst)
> ddst.norm.test(e, compute.p=TRUE, B=1000)

        Data Driven Smooth Test for Normality

data:  e,   base: ddst.base.legendre,   c: 100
WT* = 0.5847, n. coord = 1, p-value = 0.455
```

13.2.4 Heteroskedastyczność

Badanie niejednorodności wariancji w procesie resztowym zostanie zaprezentowane za pomocą dwóch testów: Goldfelda-Quandta oraz Harrisona-McCabe'a. W obu przypadkach rozważane są następujące hipotezy:

$$\begin{aligned} H_0 &: \ \sigma_1^2 = \sigma_2^2 \\ H_1 &: \ \sigma_1^2 \neq \sigma_2^2 \end{aligned} \tag{13.13}$$

gdzie σ_2^2 i σ_1^2 to wariancja odpowiednio w pierwszej i drugiej podpróbie. Wariancje resztowe wyznaczamy według następujących wzorów:

$$s_1^2 = \frac{1}{(n_1 - k - 1)\sum_{i=1}^{n_1} e_i^2} \tag{13.14a}$$

$$s_2^2 = \frac{1}{(n_2 - k - 1)\sum_{i=1}^{n_2} e_i^2} \tag{13.14b}$$

W pierwszej kolejności należy posortować niemalejąco zmienne względem zmiennej niezależnej `doch`. Należy zaznaczyć, że w przypadku zmiennych przekrojowo-czasowych takie porządkowanie nie jest wymagane.

```
> doch1= sort(doch)      # sortowanie danych - niemalejąco
> wyd1= wyd[order(doch)] # sortowanie danych - niemalejąco względem doch
> t1= data.frame(wyd1,doch1)
> m1= lm(wyd1~doch1+0,data=t1[1:8,])   # regresja dla 1 podpróby
> m2= lm(wyd1~doch1+0,data=t1[9:16,])  # regresja dla 2 podpróby
> S1= 1/(8-1-1)*sum(resid(m1)^2)     # wariancja dla 1 podpróby
> S2= 1/(8-1-1)*sum(resid(m2)^2)     # wariancja dla 2 podpróby
```

Statystyka testowa w teście Goldfelda-Quandta jest następująca:

$$F = \frac{s_2^2}{s_1^2} \tag{13.15}$$

```
> F= S2/S1; F
[1] 0.3422651
```

```
> gqtest(mnk0,order.by=doch,fraction=0)

        Goldfeld-Quandt test

data:  mnk0
GQ = 0.3423, df1 = 7, df2 = 7, p-value = 0.9097
```

Z kolei w teście Harrisona-McCabe'a statystyka będzie miała postać:

$$HMC = \frac{\sum_{i=1}^{s} e_i^2}{\sum_{i=1}^{n} e_i^2} \tag{13.16}$$

```
> s= 8  # liczba (odsetek) obserwacji w podpróbie
> m= lm(wyd1~doch1+0,data=t1)
> rm= resid(m)
> hmc= sum(rm[1:s]^2)/sum(rm^2)
> hmc
[1] 0.7359933
```

```
> hmctest(mnk0,order.by=doch,point=0.5)

        Harrison-McCabe test

data:  mnk0
HMC = 0.736, p-value = 0.919
```

Otrzymane wyniki na podstawie omawianych testów wskazują, że brak jest podstaw do odrzucenia H_0 zakładającej homoskedastyczność reszt.

13.2.5 Obserwacje odstające

Obserwacje odstające dzielimy na: wpływowe (influential) oraz nietypowe (outliers). Pierwsze z nich wpływają na wartości ocen parametrów modelu (czyli na kąt nachylenia linii regresji) a drugie na wielkości reszt (czyli na dopasowanie modelu). W programie R dzięki funkcjom `influence.measures(stats)` i `outlierTest(car)` możemy zidentyfikować obserwacje odstające. W celu wykrycia obserwacji wpływowych zostaną wykorzystane następujące wartości:

wartości wpływu:

$$h_i = \text{diag}\Big(X(X^TX)^{-1}X^T\Big) \tag{13.17}$$

gdzie: dla $h_i > \frac{3 \cdot k}{n}$ i-tą obserwację uważa się za wpływową.

```
> X= as.matrix(doch)
> hi= dia((X %*% solve(t(X) %*% X) %*% t(X)))
# z wykorzystaniem gotowej funkcji:
> hi= hatvalues(mnk0)        # wartości hat values
```

odległości Cooka:

$$CD_i = \frac{e_i \cdot h_i}{k \cdot s^2 \cdot (1 - h_i)^2} \tag{13.18}$$

gdzie: dla $CD_i > F(0,05, k, n-k)$ i-tą obserwację uważa się za wpływową.

```
> i= 1:16    # indeksy obserwacji
> k= 1       # liczba parametrów łącznie z wyrazem wolnym
> s= summary(mnk0)$sigma       # błąd standardowy reszt
> s2= (summary(mnk0)$sigma)^2  # wariancja reszt
> CDi= (e[i]^2*hi[i])/(k*s2*(1-hi[i])^2) # odległości Cooka.
# z wykorzystaniem gotowej funkcji:
> cook= cooks.distance(mnk0)  # wartości Cook's distance
```

ilorazy kowariancji:

$$COVRIATO_i = \left(\frac{s_{(-i)}}{s}\right)^{2 \cdot k} \cdot \frac{1}{1 - hi} \tag{13.19}$$

gdzie: dla $|1 - COVRATIO_i| > \frac{3 \cdot k}{n-k}$ i-tą obserwację uważa się za wpływową.

```
covr= covratio(mnk0)      # wartości COVRATIO
```

miary $DFFITS$:

$$DFFITS_i = e_i \cdot \frac{\sqrt{h_i}}{s_{(-i)} \cdot (1 - h_i)} \tag{13.20}$$

gdzie: $s_{(-i)}$ oznacza błąd standardowy reszt po usunięciu i-tej obserwacji ze zbioru danych:

```
> si= influence(mnk0)$sigma
```

natomiast dla $|DFFITS_i| > 3 \cdot \sqrt{\frac{k}{n-k}}$ i-tą obserwację uważa się za wpływową.

```
dff= dffits(mnk0)         # wartości  DFFITS
```

miary $DFBETAS$:

$$DFBETAS_i = \frac{\beta - \beta_{(-i)}}{s_{(-i)}\sqrt{(X^T X)^{-1}}} \tag{13.21}$$

gdzie: β to oszacowany współczynnik z wykorzystaniem pełnego zbioru danych oraz $\beta_{(-i)}$ to oszacowany współczynnik po usunięciu i-tej obserwacji ze zbioru danych:

```
> dbetai= influence(mnk0)$coefficients
```

natomiast dla $|DFBETAS_i| > 1$ i-tą obserwację uważa się za wpływową.

```
dfb= dfbetas(mnk0)       # wartości DFBETAS
```

Wykorzystamy teraz gotową funkcję do wyznczenia obserwacji wpływowych.

```
> summary(influence.measures(mnk0))
Potentially influential observations of
         lm(formula = wyd ~ doch + 0, data = t) :

            dfb.doch dffit cov.r  cook.d hat
mazowieckie 0.00     0.00  1.20_* 0.00   0.11
```

Zatem obserwacja dotycząca województwa mazowieckiego okazała się wpływowa na podstawie ilorazu kowariancji. Poniżej zostały przedstawione obliczenia dla wszystkich obserwacji.

```
> influence.measures(mnk0)
Influence measures of
         lm(formula = wyd ~ doch + 0, data = t) :

               dfb.doch    dffit cov.r   cook.d    hat inf
łódzkie         0.26381  0.26381 1.071 6.96e-02 0.0657
mazowieckie    -0.00223 -0.00223 1.203 5.31e-06 0.1094   *
małopolskie     0.04075  0.04075 1.134 1.78e-03 0.0574
śląskie         0.02653  0.02653 1.146 7.54e-04 0.0656
lubelskie       0.24256  0.24256 1.026 5.77e-02 0.0437
podkarpackie    0.33300  0.33300 0.922 9.84e-02 0.0368
podlaskie      -0.28983 -0.28983 1.027 8.16e-02 0.0549
świętokrzyskie -0.06102 -0.06102 1.118 3.97e-03 0.0465
lubuskie       -0.04887 -0.04887 1.142 2.55e-03 0.0644
wielkopolskie  -0.39757 -0.39757 0.981 1.45e-01 0.0637
zach.pomorskie -0.23806 -0.23806 1.091 5.76e-02 0.0687
dolnośląskie    0.04143  0.04143 1.154 1.84e-03 0.0729
opolskie        0.62331  0.62331 0.804 2.93e-01 0.0619
kuj.pom         0.02947  0.02947 1.132 9.29e-04 0.0545
pomorskie      -0.13833 -0.13833 1.134 2.01e-02 0.0717
war.mazur      -0.36192 -0.36192 1.001 1.23e-01 0.0622
```

Do identyfikacji obserwacji nietypowych możemy wykorzystać reszty standaryzowane:

$$e_{i(stand)} = \frac{e_i}{s \cdot \sqrt{1 - h_i}} \tag{13.22}$$

gdzie: dla $|e_{i(stand)}| > 2$ i-tą obserwację można podejrzewać o nietypowość.

```
> es= rstandard(mnk0)    # reszty standaryzowane
```

lub reszty studentyzowane:

$$e_{i(stud)} = \frac{e_i}{s_{(-i)} \cdot \sqrt{1-h_i}} \tag{13.23}$$

gdzie: dla $|e_{i(stud)}| > 2$ i-tą obserwację można podejrzewać o nietypowość.

```
> et= rstudent(mnk0)     # reszty studentyzowane
```

Reszty `et` mają rozkład t-Studenta o stopniach swobody: $n-p-2$ (model z wyrazem wolnym) lub $n-p-1$ (model bez wyrazy wolnego). Zatem do identyfikacji obserwacji odstających wykorzystamy $p-value$ tego rozkładu weryfikując następukące hipotezy statystyczne:

$$\begin{aligned} H_0 &: \text{ i-ta obserwacja jest typowa} \\ H_1 &: \text{ i-ta obserwacja jest nietypowa} \end{aligned} \tag{13.24}$$

Dla obserwacji o największej wartości bezwzględnej `et` statystyka testu będzie miała postać:

$$\max|e_{i(stud)}| \tag{13.25}$$

```
# p-value z rozkładu t-Studenta:
> p.value= 2*(1-pt( max(abs(et)) ,16-1-1))
> p.value
[1] 0.02937727
# poprawka Bonferonniego:
> Bp.value= 16*p.value
> Bp.value
[1] 0.4700364
# wykorzystanie funkcji outlierTest:
> outlierTest(mnk0)

No Studentized residuals with Bonferonni p < 0.05
Largest |rstudent|:
         rstudent unadjusted p-value Bonferonni p
opolskie 2.425829           0.029377      0.47004
```

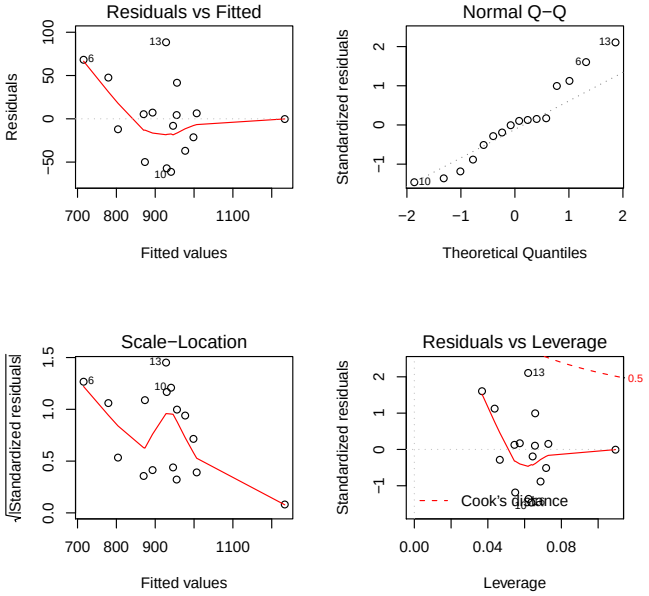
Ponieważ $p-value = 0,47004$ to na poziomie istotności $\alpha = 0,05$ brak jest podstaw do odrzucenia H_0. Gdy dodamy odpowiednie argumenty do funkcji `outlierTest`, to otrzymamy uporządkowane malejąco wartości $|e_{i(stud)}|$ oraz $p-value$.

```
> outlierTest(mnk0,order=T,cutoff=Inf,n.max=Inf)
                   rstudent unadjusted p-value Bonferonni p
opolskie         2.425829301           0.029377      0.47004
podkarpackie     1.702938479           0.110670           NA
wielkopolskie   -1.524648740           0.149610           NA
war.mazur       -1.405761385           0.181600           NA
podlaskie       -1.202794661           0.249000           NA
lubelskie        1.134839609           0.275500           NA
łódzkie          0.994454997           0.336880           NA
zach.pomorskie  -0.876672700           0.395460           NA
pomorskie       -0.497741566           0.626390           NA
```

```
świętokrzyskie -0.276280971          0.786370          NA
lubuskie       -0.186335919          0.854850          NA
małopolskie     0.165166434          0.871170          NA
dolnośląskie    0.147748037          0.884650          NA
kuj.pom         0.122770528          0.904030          NA
śląskie         0.100105764          0.921680          NA
mazowieckie    -0.006351306          0.995020          NA
```

W dignostyce modelu ekonometrycznego bardzo pomocnym narzędziem są również wykresy, które można wygenerować za pomocą komendy:

```
> par(mfrow=c(2,2))
> plot(mnk0)
```



Rysunek 13.3: Diagnostyka modelu.

13.2.6 Metoda najmniejszych wartości bezwzględnych

Jeśli w zbiorze danych występują obserwacje odstające to należy je wyeliminować lub zastosować regresję kwantylową. Dla parametru $\tau = 0,5$ (mediana) otrzymujemy estymator metody najmniejszego odchylenia bezwzględnego, którego kryterium optymalizacji jest przedstawione poniżej (wzory 13.26a oraz 13.26b).

$$\sum_{i=1}^{n} |e_i| \to \min \tag{13.26a}$$

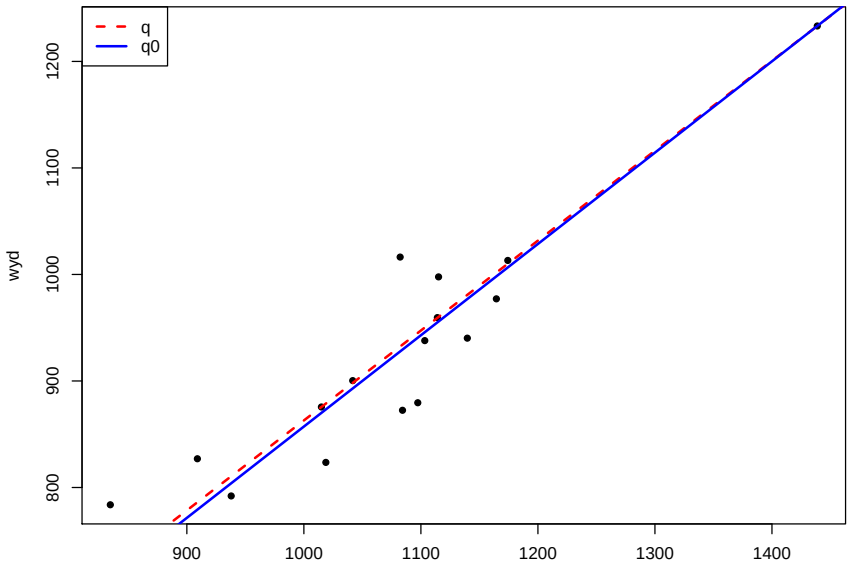
$$\sum_{i=1}^{n} |y_i - \alpha_0 - \alpha_1 x_{1i}| \to \min \tag{13.26b}$$

```
> library(quantreg)
# model z wyrazem wolnym:
```

```
> q=rq(wyd~doch,tau=0.5)
> summary(q,se='nid')

Call: rq(formula = wyd ~ doch, tau = 0.5)

tau: [1] 0.5

Coefficients:
            Value     Std. Error t value   Pr(>|t|)
(Intercept)  18.81043 138.88795    0.13544   0.89420
doch          0.84413   0.12837    6.57568   0.00001
```

```
# model bez wyrazu wolnego:
> q0=rq(wyd~doch+0,tau=0.5)
> summary(q0,se='nid')

Call: rq(formula = wyd ~ doch + 0, tau = 0.5)

tau: [1] 0.5

Coefficients:
     Value    Std. Error t value  Pr(>|t|)
doch  0.85720  0.01716   49.95465  0.00000
```

```
> plot(doch,wyd,cex=1.2,pch=20)
> abline(q,col='red',lwd=2,lty=2); abline(q0,col='blue',lwd=2)
> legend("topleft",c("q","q0"),col=c("red","blue"),lty=c(2,1),lwd=2)
```



Rysunek 13.4: Regresja kwantylowa.

13.3 Estymacja modelu nieliniowego

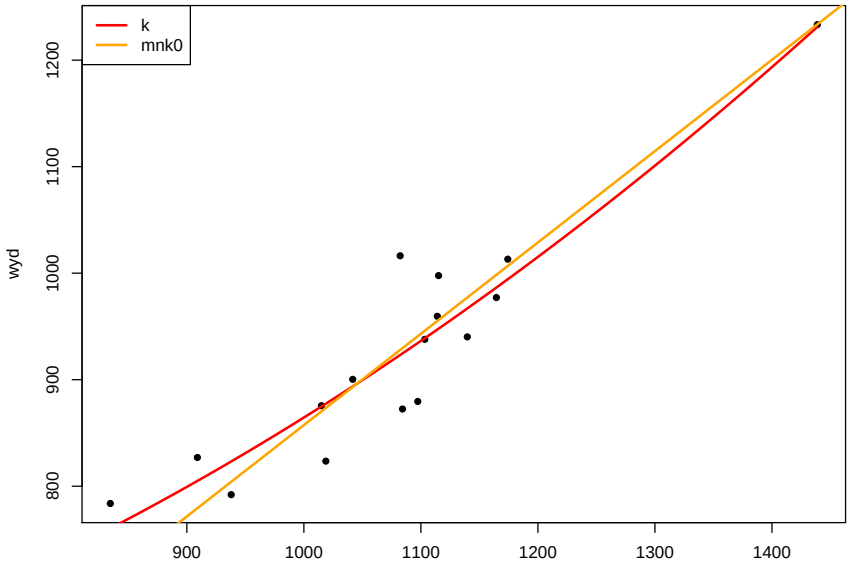
13.3.1 Model kwadratowy

Pierwszą funkcją nieliniową jaką spróbujemy dopasować do naszych danych będzie model kwadratowy:

$$y = \alpha_0 + \alpha_1 x^2 \tag{13.27}$$

Estymację parametrów funkcji o wzorze 13.27 dokonamy za pomocą poniższej komendy:

```
> k= lm(wyd~I(doch^2),data=t)
> summary(k)

Call:
lm(formula = wyd ~ I(doch^2), data = t)

Residuals:
    Min      1Q  Median      3Q     Max
-54.857 -27.803   1.452  19.653  93.152

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.218e+02  4.206e+01  12.404 6.11e-09 ***
I(doch^2)   3.427e-04  3.453e-05   9.924 1.03e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 40.67 on 14 degrees of freedom
Multiple R-squared: 0.8755,     Adjusted R-squared: 0.8667
F-statistic: 98.49 on 1 and 14 DF,  p-value: 1.026e-07
```



Rysunek 13.5: Regresja liniowa i kwadratowa.

```
> plot(t,cex=1.2,pch=20)
> curve(coef(k)[1]+coef(k)[2]*x^2,add=T,col="red",lwd=2)
> abline(mnk0,col="orange",lty=1,lwd=2)
> legend("topleft",c("k","mnk0"),col=c("red","orange"),lty=1,lwd=2)
```

```
> AIC(mnk0,k)
     df      AIC
mnk0  2 168.9711
k     3 167.8451  # funkcja kwadratowa jest lepszym modelem
```

```
> f=function(x){
  x=x
  0.000685408*x   # pierwsza pochodna funkcji kwadratowej
  }
> f(c(900,1000,1100,1200))
[1] 0.6168672 0.6854080 0.7539488 0.8224896
```

Otrzymane wartości można zinterpretować w następujący sposób. Wydatki w przeliczeniu na jedną osobę rosły średnio o $0,62$zł na jedną złotówkę przyrostu dochodów, przy poziomie dochodów równych 900zł. Natomiast prz poziomie dochodów równych 1200zł wydatki rosły średnio o $0,82$zł na złotówkę przyrostu dochodów.

13.3.2 Model wykładniczy

Analityczna postać funkcji wykładniczej jest podana poniżej:

$$y = \alpha_0 \cdot \alpha_1^x \tag{13.28}$$

gdzie: $\alpha_0 > 0$ oraz $\alpha_1 > 0$ i $\alpha_1 \neq 1$

Aby oszacować parametry modelu nieliniowego z wykorzystaniem funkcji `nls(stats)` (nieliniowa metoda najmniejszych kwadratów) należy znaleźć parametry startowe. Można je wyznaczyć sprowadzając funkcję nieliniową do postaci liniowej.

$$\ln y = \beta_0 \cdot \beta_1 \cdot x \tag{13.29}$$

```
> mw= lm(log(wyd)~doch)
> summary(mw)

Call:
lm(formula = log(wyd) ~ doch)

Residuals:
      Min        1Q    Median        3Q       Max
-0.063814 -0.029988  0.003799  0.022039  0.096239

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.971e+00  9.313e-02  64.115  < 2e-16 ***
doch        7.916e-04  8.566e-05   9.241 2.46e-07 ***
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.04456 on 14 degrees of freedom
Multiple R-squared: 0.8591,     Adjusted R-squared: 0.8491
F-statistic: 85.39 on 1 and 14 DF,  p-value: 2.462e-07
```

W kolejnym kroku obliczymy parametry funkcji wykładniczej według wzorów: $\alpha_0 = \exp(\beta_0)$ oraz $\alpha_1 = \exp(\beta_1)$.

```
> b0= exp(coef(mw)[1])
> b0
(Intercept)
   391.9122
> b1= exp(coef(mw)[2])
> b1
    doch
1.000792
```

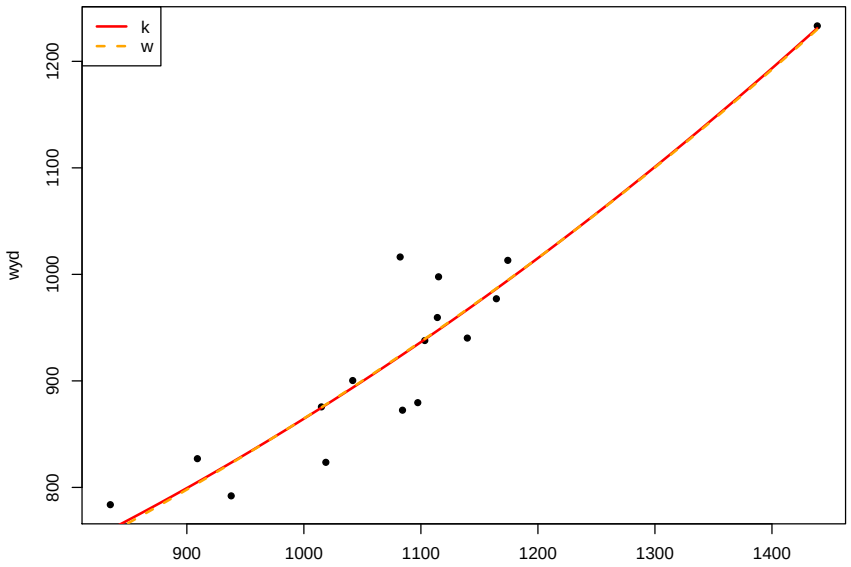
Otrzymane wartości można wykorzystać jako parametry startowe w nieliniowej metodzie najmniejszych kwadratów.

```
> w= nls(wyd~a0*a1^doch,start=list(a0=b0,a1=b1),data=t)
> summary(w)

Formula: wyd ~ a0 * a1^doch

Parameters:
    Estimate Std. Error  t value Pr(>|t|)
a0 3.873e+02  3.346e+01    11.58 1.48e-08 ***
a1 1.001e+00  7.733e-05 12942.47  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 40.84 on 14 degrees of freedom

Number of iterations to convergence: 2
Achieved convergence tolerance: 1.024e-06
```

```
> AIC(k,w)
  df      AIC
k  3 167.8451  # funkcja kwadratowa jest lepszym modelem
w  3 167.9795
```

```
> (coef(w)[2]-1)*100
        a1
0.08034964
```

Interpretacja otrzymanych parametrów jest następująca. Jeżeli dochód wzrośnie o 1zł to wydatki również wzrosną o $0,08\%$.



Rysunek 13.6: Regresja kwadratowa i wykładnicza.

13.3.3 Model hiperboliczny

Kolejną funkcją jaką będziemy badać to model hiperboliczny o postaci:

$$y = \frac{\alpha_0 x^2}{x + \alpha_1} \tag{13.30}$$

Parametry startowe wyznaczymy za pomocą następującego wzoru:

$$\frac{1}{y} = \beta_1 \frac{1}{x} + \beta_2 \frac{1}{x^2} \tag{13.31}$$

```
> mh= lm(I(1/wyd)~I(1/doch)+I(1/doch^2)+0)
> summary(mh)

Call:
lm(formula = I(1/wyd) ~ I(1/doch) + I(1/doch^2) + 0)

Residuals:
       Min         1Q     Median         3Q        Max
-9.731e-05 -3.285e-05 -1.201e-06  3.315e-05  7.973e-05

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
I(1/doch)      1.4011     0.1151  12.172 7.79e-09 ***
I(1/doch^2) -249.8795   118.3491  -2.111   0.0532 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.116e-05 on 14 degrees of freedom
Multiple R-squared: 0.9981,     Adjusted R-squared: 0.9978
F-statistic:  3682 on 2 and 14 DF,  p-value: < 2.2e-16
```

gdzie: $\alpha_0 = \frac{1}{\beta_1}$ oraz $\alpha_1 = \frac{\beta_2}{\beta_1}$.

```
> b1= 1/coef(mh)[1]
> b1
I(1/doch)
 0.713737
> b2= coef(mh)[2]/coef(mh)[1]
> b2
I(1/doch^2)
  -178.3482
```

```
> h= nls(wyd~(a0*doch^2)/(doch+a1),start=list(a0=b1,a1=b2))
> summary(h)

Formula: wyd ~ (a0 * doch^2)/(doch + a1)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
a0    0.77328    0.06732  11.487 1.63e-08 ***
a1 -107.24573   84.78731  -1.265    0.227
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 43 on 14 degrees of freedom

Number of iterations to convergence: 4
Achieved convergence tolerance: 4.592e-06
```

Ponieważ parametr α_1 jest nieistotny statystycznie, a więc gdy go pominiemy to funkcja hiperboliczna sprowadzi się do modelu liniowego.

$$y = \frac{\alpha_0 x^2}{x + \alpha_1} = \frac{\alpha_0 x^2}{x + 0} = \frac{\alpha_0 x^2}{x} = \frac{\alpha_0 x}{1} = \alpha_0 x$$

Rozdział 14

Zbiory danych

W tym miejscu przedstawimy zbiory danych wykorzystywane w poprzednich rozdziałach. Są one umieszczone w Internecie i można je ściągnąć podanymi poniżej poleceniami. Jeżeli nie mamy własnych danych, to warto na tych przećwiczyć omawiane w tej książce funkcje.

14.1 Zbiór danych GUSowskich

Korzystając z danych dostępnych na stronach GUSu `http://www.stat.gov.pl/` przygotowałem poniższy wybranych wskaźników w rozbiciu na województwa. Są to prawdziwe dane dotyczące roku 2007.

```
> daneGUS = read.table("http://www.biecek.pl/R/dane/Dane2007GUS.csv", sep=";",
      h=T, dec=",")
```

W tej ramce danych dla 16 województw zebrane są następujące dane:

1. ludnosc.do.15.lat - liczba mieszkańców w wieku do 15 lat
2. ludnosc.15.60.lat - liczba mieszkańców w wieku od 15 do 60 lat
3. ludnosc..60.lat + - liczba mieszkańców w wieku od 60 lat
4. mieszkan.wyprodukowanych - liczba mieszkań wyprodukowanych w 2007 roku
5. studenci.artystyczne - liczba studentów kierunków artystycznych
6. studenci.spoleczne - liczba studentów kierunków społecznych
7. studenci.ekonomia - liczba studentów kierunków ekonomicznych
8. studenci.prawne - liczba studentów kierunków prawniczych
9. studenci.dziennikarstwo - liczba studentów kierunku dziennikarstwo
10. studenci.biologiczne - liczba studentów kierunków biologicznych

11. studenci.fizyczne - liczba studentów kierunków fizycznych
12. studenci.matematyczno.statystyczne - liczba studentów kierunków matematycznych
13. studenci.informatyczne - liczba studentów kierunków informatycznych
14. studenci.medyczne - liczba studentów kierunków medycznych
15. studenci.inzynieryjno.techniczne - liczba studentów kierunków inżynieryjno technicznych
16. studenci.produkcja.i.przetworstwo - liczba studentów kierunków produkcja i przetwórstwo
17. studenci.architektura.i.budownictwo - liczba studentów kierunków architektura i budownictwo
18. studenci.na.ochrona.srodowiska - liczba studentów kierunku ochrona środowiska
19. studenci.na.ochrona.i.bezpieczenstwo - liczba studentów kierunków ochrona i bezpieczeństwo
20. nauczycieli.akademickich - liczba nauczycieli akademickich
21. dochod.budzetu.na.mieszkanca - średni dochód do budżetu na jednego mieszkańca
22. pracujacy.rolnictwo - liczba (w tysiącach) osób pracujących w sekorze rolniczym
23. pracujacy.przemysl - liczba (w tysiącach) osób pracujących w przemyśle
24. pracujacy.uslugi - liczba (w tysiącach) osób pracujących w sekorze usługowym
25. bezrobotni - liczba (w tysiącach) bezrobotnych
26. powierzchnia.na.mieszkanie - średnia powierzchnia mieszkania
27. powierzchnia.na.osobe - średnia powierzchnia mieszkania na osobe
28. mieszkancow.na.sklep - średnia liczba mieszkańsców na sklep

Bibliografia

[25] **„Statystyczne systemy uczące się”**, Jacek Koronacki i Jan Ćwik. WNT 2005, ISBN: 83-204-3157-3.

[26] **„The Elements of Statistical Learning: Data Mining, Inference, and Prediction”**, Trevor Hastie, Robert Tibshirani, Jerom Friedman. Springer-Verlag 2001.

[1] **„Finding Groups in Data: An Introduction to Cluster Analysis”**, Kaufman, L. and Rousseeuw, P.J. (1990) Wiley, New York.

Skorowidz