Wydział Matematyki
i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

RPiESM, Omówienie laboratorium 2

Paweł Szymański, Julian Zalewski, Antoni Zasada

21 maja 2025

Zadanie 1. Wygenerować dwie próby losowe: 20 i 100 elementową z rozkładu standardowego normalnego. Narysować dla obu prób dystrybuanty empiryczne i porównać je z odpowiednią dystrybuantą teoretyczną.

Rozwiązanie: W języku R do wygenerowania próby losowej z rozkładu normalnego służy funkcja o nazwie `rnorm`. Pierwszym argumentem jest liczba elementów próby, drugim wartość oczekiwana, a trzecim odchylenie standardowe. Aby otrzymać rozkład normalny standardowy za wartość średnią należy przyjąć $\mu = 0$, a za odchylenie $\sigma = 1$.

Dystrybuanta empiryczna jest to estymator dystrybuanty rozkładu z którego pochodzi próba. Można ją określić za pomocą wzoru:

$$\hat{F}_n(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(X_i \leq t)$$

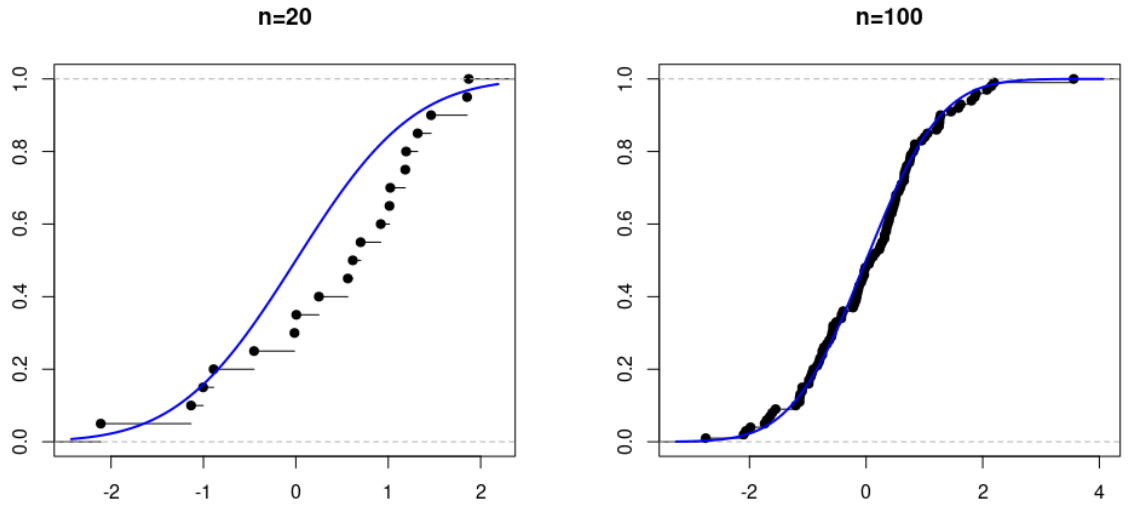
gdzie

$$\mathbb{1}(X_i \leq t) = \begin{cases} 1, X_i \leq t \\ 0, X_i > t \end{cases}$$

Do jej obliczania w R dostępna jest funkcja `ecdf`. Przyjmuje ona za argument wektor wartości próby.

Po obliczeniu dystrybuant rysujemy ich wykresy. W tym celu korzystamy z funkcji `plot`.

Ze względu na rozkład, który opisują, powinny być one zbliżone do dystrybuanty rozkładu normalnego standardowego. W celu porównania na wykresy nakładamy niebieskie krzywe obrazujące dokładną dystrybuantę rozkładu normalnego standardowego. Do uzyskania tego efektu służy funkcja `curve`, która potrzebuje wektora punktów według których ma być narysowana krzywa (w tym przypadku jest to wynik funkcji `pnorm` zwracającej dokładną dystrybuantę rozkładu normalnego) i dodatkowe argumenty, jak chociażby `col` w celu zmiany koloru linii.



Rysunek 1: Wykresy dystrybuant: po lewej z 20 elementami, po prawej z 100. Niebieska linia to spodziewany kształt.

Pełny kod rozwiązania:

```
# zad 2.1

mu = 0
sigma = 1
prob1 = rnorm(n=20, mean=mu, sd=sigma)
prob2 = rnorm(n=100, mean=mu, sd=sigma)

# to achieve two plots next to each other
par(mfrow=c(1,2))

plot(ecdf(prob1), main='n=20', xlab='', ylab='')
curve(pnorm(x, mu, sigma), add=TRUE, col='blue', lwd=2)

plot(ecdf(prob2), main='n=100', xlab='', ylab='')
curve(pnorm(x, mu, sigma), add=TRUE, col='blue', lwd=2)
```

Zadanie 2. Wygenerować $N = 1000$ obserwacji z rozkładu normalnego standardowego. Utworzyć histogram oraz estymator jądrowy dla tej próby. Nałożyć na uzyskany obraz wykres gęstości teoretycznej rozkładu normalnego.

Rozwiązanie: Dokładnie tak jak w zadaniu 1. generujemy próbę losową rozkładu normalnego standardowego, tyle że tym razem dla 1000 elementów.

Następnie tworzymy histogram. W R można to zrobić wywołując gotową funkcję `hist`, która przyjmuje wektor elementów, którego histogram ma zostać narysowany, wartość `br` (czyli *break length*) określającą grubość słupków histogramu i `freq` decydującą czy przedstawiona ma być gęstość czy częstotliwość rozkładu.

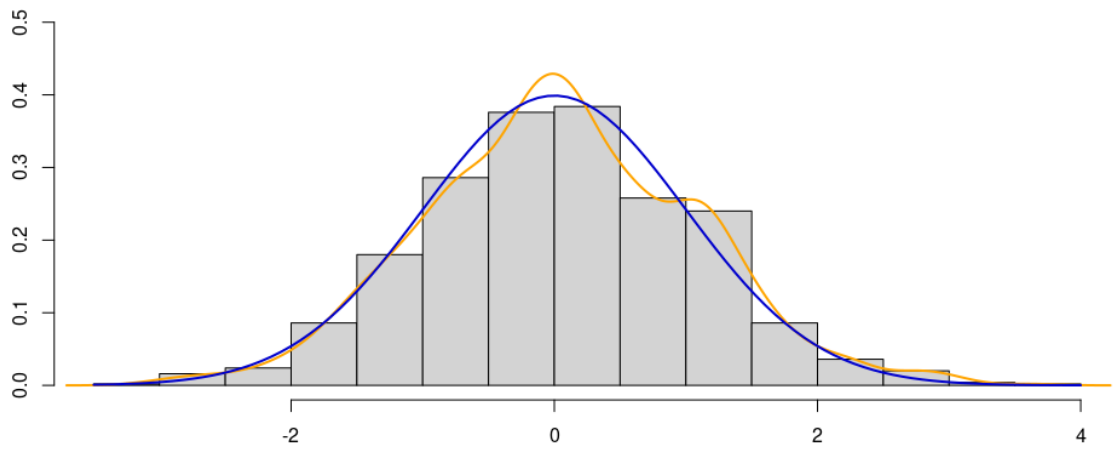
Estymator jądrowy jest to estymator służący wygładzaniu podanego rozkładu próby. Dla N elementowej realizacji próby losowej $x_1, x_2, ..., x_N$ jest on definiowany wzorem:

$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\Big(\frac{x - x_i}{h}\Big),$$

gdzie dodatni współczynnik h określa się mianem parametru wygładzania, a funkcja K jest mierzalną, symetryczną względem zera oraz posiadającą w tym punkcie słabe maksimum globalne funkcją $K : \mathbb{R} \to [0, \infty)$ spełniającą warunek $\int_{\mathbb{R}} K(x)dx = 1$ i jest ona nazywana jądrem.

W R do wyznaczania estymatora jądrowego istnieje funkcja `density`, która przyjmuje rozkład próby losowej oraz wartość `bw` (*smoothing bandwith*), czyli współczynnik wygładzenia (w poprzednim akapicie oznaczony jako h). Do jego narysowania można zastosować funkcji `lines`.

Do tego należy nakreślić wykres gęstości teoretycznej rozkładu normalnego. Podobnie jak w zadaniu 1. można do tego użyć funkcji `curve`, tyle że tym razem z funkcją `dnorm` dającą rozkład gęstości, a nie dystrybuantę.



Rysunek 2: Histogram losowej próby wraz z estymatorem jądrowym (`bw` $= 0.2$, na pomarańczowo) oraz teoretyczną gęstością rozkładu normalnego (na niebiesko).

Pełny kod rozwiązania:

```
# zad 2.2

mu = 0
sigma = 1
N = 1000
X = rnorm(N, mean=mu, sd=sigma)

(h = hist(X, br=20, freq=FALSE, ylim=c(0,0.5), main='Histogram losowej próby rozkładu
    normalnego standardowego', ylab='', xlab=''))
lines(density(X, bw=0.2), col='orange', lwd=2)
curve(dnorm(x, mu, sigma), add=TRUE, col='mediumblue', lwd=2)
```

Zadanie 3. Sporządzić wykresy funkcji prawdopodobieństwa następujących rozkładów dwumianowych: `binom(10, 0.5)`, `binom(10, 0.25)`, `binom(50, 0.25)`. Wyciągnąć wnioski.

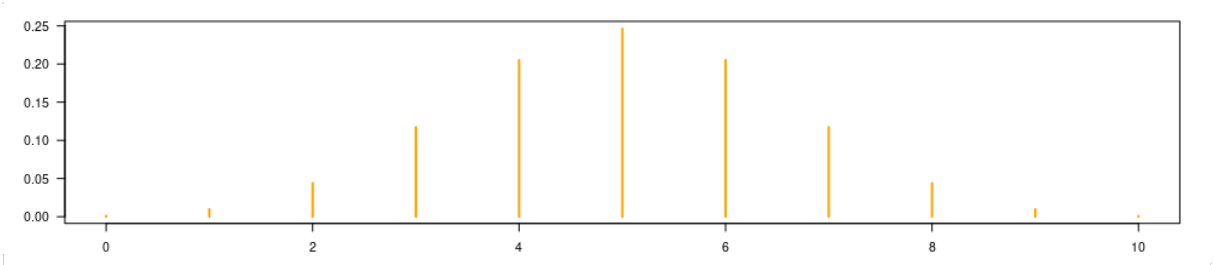
Rozwiązanie: W celu uzyskania trzech podanych rozkładów dwumianowych należy wykonać funkcję `dbinom` z podanymi w poleceniu parametrami.

Do narysowania wykresów tych rozkładów stosujemy funkcję `plot`. W celu uzyskania kilku wykresów jeden nad drugim przed użyciem plotów wywołujemy funkcję `par` z argumentem `mfrow` ustawionym na wartość dwuwymiarowego wektora $(3, 1)$.

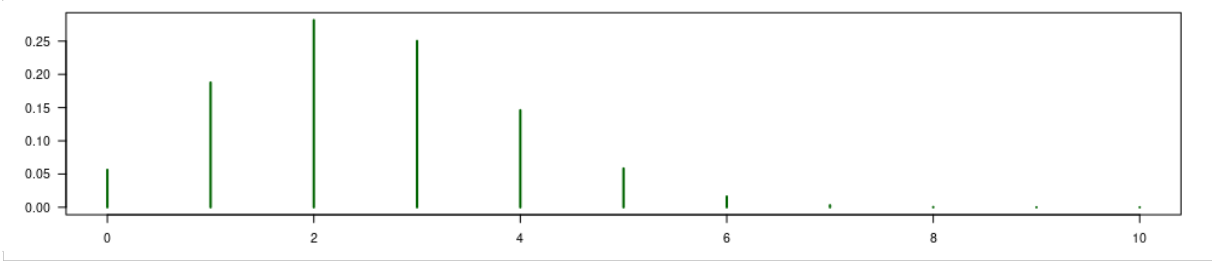
Jak można zauważyć na rysunku [5] wykres przypomina swoim wyglądem gęstość rozkładu normalnego. Można porównać go z tą funkcją rysując dodatkową krzywą na wykresie rozkładu dwumianowego korzystając z `curve`.

By otrzymać tą krzywą możemy obliczyć wartość oczekiwaną i odchylenie standardowe rozkładu dwumianowego i wziąć je jako te dla rozkładu normalnego. Wartość średnia rozkładu dwumianowego Z jest równa $EZ = np$, a odchylenie $\sigma_Z = \sqrt{np(1-p)}$, gdzie n jest liczbą elementów w rozkładzie, a p prawdopodobieństwem sukcesu.

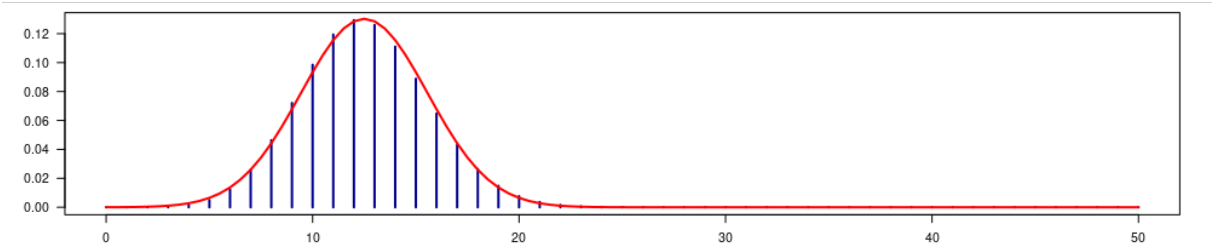
Dla naszego wykresu obliczenia wyglądają następująco: $\mu = 50 \cdot 0.25 = \frac{25}{2}$; $\sigma = \sqrt{50 \cdot 0.25 \cdot (1 - 0.25)} = \sqrt{\frac{150}{4}}$



Rysunek 3: Wykres funkcji prawdopodobieństwa rozkładu dwumianowego `binom(10, 0.5)`



Rysunek 4: Wykres funkcji prawdopodobieństwa rozkładu dwumianowego `binom(10, 0.25)`



Rysunek 5: Wykres funkcji prawdopodobieństwa rozkładu dwumianowego `binom(50, 0.25)` wraz z funkcją gęstości rozkładu normalnego $N(\frac{25}{2}, \frac{150}{4})$

Pełny kod rozwiązania:

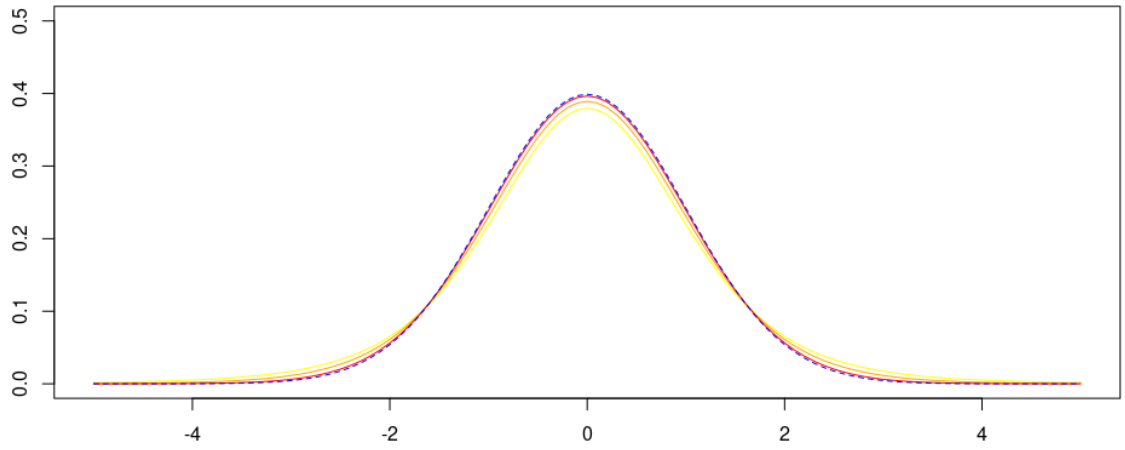
```
# zad 2.3

x = 0:10
y = 0:50

den_X = dbinom(x, 10, 0.5)
den_Y = dbinom(x, 10, 0.25)
den_Z = dbinom(y, 50, 0.25)

EZ = 50 * 0.25
VarZ = 50 * 0.25 * (1 - 0.25)

den_ZN = dnorm(y, mean=EZ, sd=sqrt(VarZ))
par(mfrow=c(3,1))
plot(x, den_X, pch=19, type='h', lwd=2, las=1, col='orange', lty=1, xlab='', ylab='',
    main='n=10, p=0.5')
plot(x, den_Y, pch=19, type='h', lwd=2, las=1, col='darkgreen', lty=1, xlab='', ylab='',
    main='n=10, p=0.25')
plot(y, den_Z, pch=19, type='h', lwd=2, las=1, col='darkblue', lty=1, xlab='', ylab='',
    main='n=50, p=0.25')
curve(dnorm(x, mean=EZ, sd=sqrt(VarZ)), add=TRUE, col='red', lwd=2, xlab='', ylab='')
```

Zadanie 4. Utworzyć wykresy gęstości zmiennych losowych o rozkładzie t-Studenta o 5, 10 oraz 40 stopniach swobody. Przeanalizować, jak zmienia się gęstość rozkładu t-Studenta wraz ze wzrostem liczby stopni swobody.

Rozwiązanie: W celu wygenerowania gęstości zmiennych losowych rozkładu t-Studenta w R istnieje funkcja `dt`. Przyjmuje ona argument `df`, który określa stopnie swobody rozkładu.

Aby narysować te rozkłady stosujemy funkcję `curve`. By uzyskać kilka wykresów na jednym wykresie należy ustawić argument `add` na wartość `TRUE`.

Wraz ze wzrostem stopni swobody rozkład t-Studenta zdaje się dążyć do rozkładu normalnego. Zaiste, faktycznie dla dużych ν t_ν ma w przybliżeniu rozkład $N(0,1)$. W celu porównania na wykres naniesiono niebieską, przerywaną linię reprezentującą rozkład normalny za pomocą `curve` z `dnorm`. Rzeczywiście, im większa liczba stopni, tym wykres bardziej się pokrywa z wykresem rozkładu normalnego.



Rysunek 6: Rozkłady t-Studenta o s stopniach swobody. Żółty: $s = 5$, pomarańczowy: $s = 10$, czerwony: $s = 40$.

Pełny kod rozwiązania:

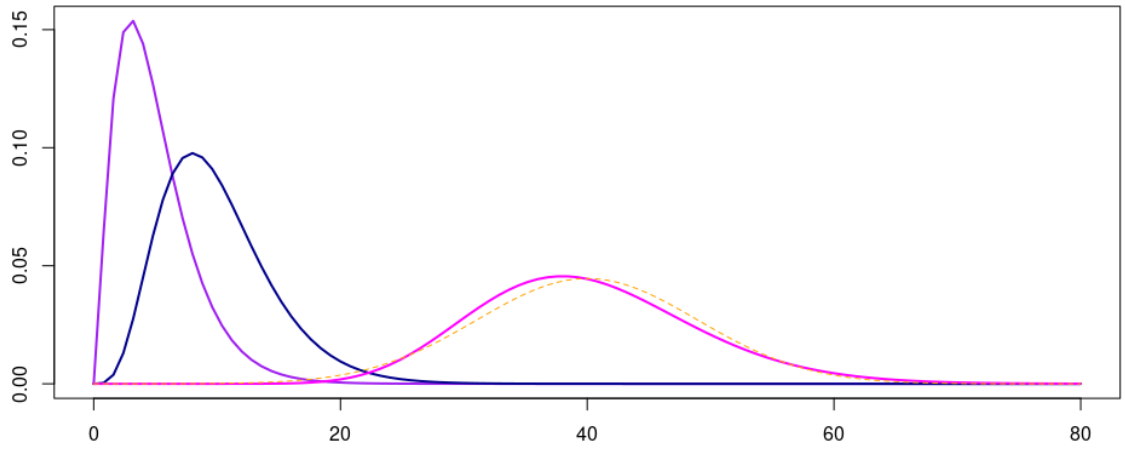
```
# zad 2.4

curve(dt(x, df=5), xlim=c(-5,5), ylim=c(0,0.5), col='yellow')
curve(dt(x, df=10), xlim=c(-5,5), ylim=c(0,0.5), col='orange', add=TRUE)
curve(dt(x, df=40), xlim=c(-5,5), ylim=c(0,0.5), col='red', add=TRUE)
curve(dnorm(x, mean=0, sd=1), xlim=c(-5,5), ylim=c(0,0.5), col='blue', add=TRUE, lty=2)
```

Zadanie 5. Utworzyć wykresy gęstości zmiennych losowych o rozkładzie chi-kwadrat o 5, 10 oraz 40 stopniach swobody. Przeanalizować, jak zmienia się gęstość rozkładu chi-kwadrat wraz ze wzrostem liczby stopni swobody.

Rozwiązanie: W celu wygenerowania gęstości zmiennych losowych rozkładu chi-kwadrat w R istnieje funkcja `dchisq`. Przyjmuje ona argument `df`, który określa stopnie swobody rozkładu.

Aby narysować te rozkłady tak samo jak w zadaniu 4 stosujemy funkcję `curve` z argumentem `add` ustawionym na wartość `TRUE`.

Na rysunku [7] można zauważyć, że wraz ze wzrostem stopni swobody wykres staje się coraz bardziej podobny do wykresu rozkładu normalnego. Rzeczywiście, dla dużych p χ_p ma w przybliżeniu rozkład $N(p, 2p)$. Rysujemy zatem pomarańczową, przerywaną linią rozkład normalny dla największej liczby stopni swobody, jakie mamy, czyli 40.



Rysunek 7: Rozkłady chi-kwadrat o s stopniach swobody. Na fioletowo: $s = 5$, niebiesko: $s = 10$, różowo: $s = 40$.

Pełny kod rozwiązania:

```
# zad 2.5

curve(dchisq(x, df=5), xlim=c(0,80), col='purple', lwd=2, xlab='', ylab='')
curve(dchisq(x, df=10), xlim=c(0,80), col='darkblue', lwd=2, add=TRUE)
curve(dchisq(x, df=40), xlim=c(0,80), col='magenta', lwd=2, add=TRUE)
curve(dnorm(x, mean=40, sd=sqrt(2*40)), xlim=c(0, 80), ylim=c(0,0.5), col='orange', add=
    TRUE, lty=2)
```

Zadanie 6. Zbiór `Cars93`, znajdujący się w bibliotece `MASS`, zawiera dane dotyczące różnych modeli samochodów osobowych.

a) Utworzyć nową zmienną o nazwie `zp.m` opisującą zużycie paliwa (mierzone w litrach na 100 km) podczas jazdy samochodu w mieście. Przyjąć, że 1 mila to 1.6 km; 1 galon amerykański to 3.8 litra. Odpowiednie dane wyrażone w milach na galon znajdują się w zmiennej `MPG.city`.

b) Wyznaczyć podstawowe statystyki próbkowe dla danych w zmiennej `zp.m`. Obliczyć kwantyl rzędu 0.95 dla tych danych i podać jego interpretację.

c) Sporządzić wykresy skrzynkowe dla zmiennej `zp.m` osobno dla samochodów amerykańskich i nieamerykańskich. Powtórzyć to samo dla zmiennej `MPG.city`.

d) Narysować wykres słupkowy i kołowy dla zmiennej `Type`. Ile spośród badanych samochodów zaliczono do kategorii sportowe?
Wskazówka: Aby uzyskać liczności poszczególnych grup użyć funkcji `table()`.

Rozwiązanie: Na początku ładujemy bibliotekę `MASS` za pomocą wcześniej poznanej funkcji `library`. Używając operatora `$` „doklejamy" zmienną `zp.m` do naszej zmiennej `x`. Kwantyl rzędu 0.95 liczymy za pomocą funkcji `quantile` - jego interpretacja to jakie jest minimalne zużycie paliwa dla 5% najbardziej ekonomicznych aut. Do zilustrowania danych używamy funkcji `boxplot` - stworzy ona wykres pudełkowy.

Zwraca ona listę z wartościami pozwalającymi przeanalizować sporządzony wykres:

- `stats` - każda kolumna zawiera: granicę dolną dolnego wąsa, dół pudełka, medianę, górę pudełka oraz granicę górną górnego wąsa
- `n` - wektor z obserwacjami (różnymi od NA) z danej grupy
- `conf` - macierz z granicami pudełek
- `out` - macierz, w której każda kolumna zawiera obserwacje odstające (`outliery`)
- `group` - wektor, który mówi do której grupy należą obserwacje odstające
- `names` - nazwy grup

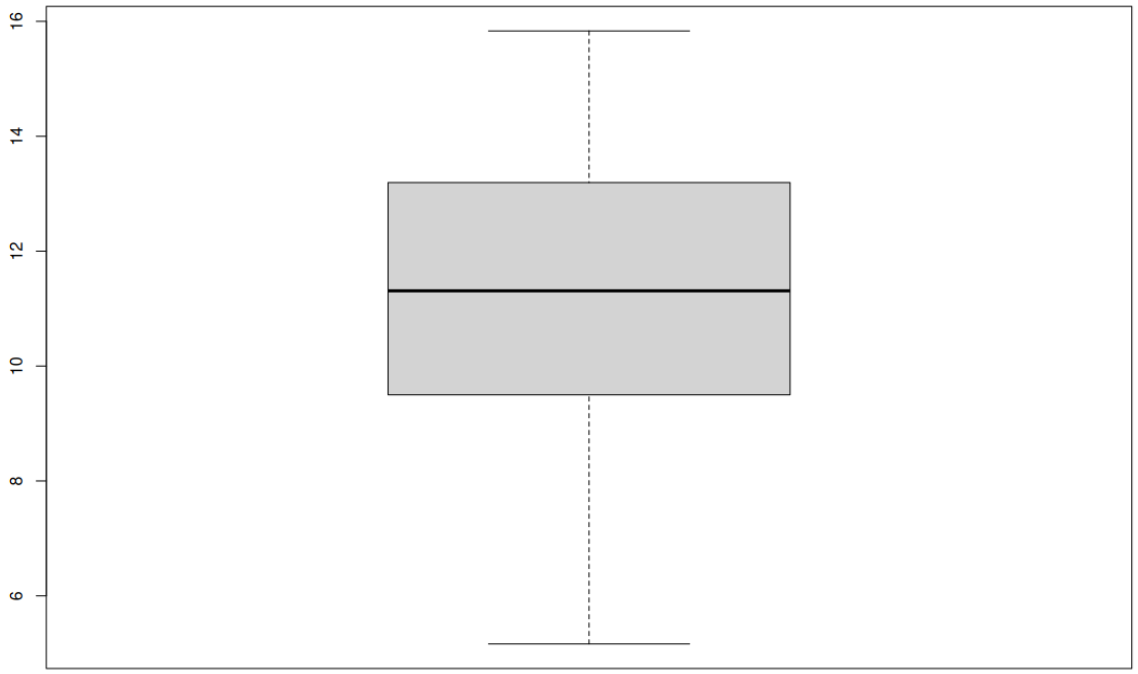
Wykres pudełkowy (przedstawiony poziomo) tworzy się odkładając na poziomej osi wartości niektórych parametrów rozkładu. Nad osią umieszczony jest prostokąt (pudełko), którego lewy bok jest wyznaczony przez pierwszy kwartyl, zaś prawy bok przez trzeci kwartyl. Szerokość pudełka odpowiada wartości rozstępu ćwiartkowego. Wewnątrz prostokąta znajduje się pionowa linia, określająca wartość mediany.

Po prawej i lewej stronie znajdują się odcinki zwane wąsami. Wąsy mają długość ostatniej wartości wewnątrz półtorej wartości rozstępu międzykwartylowego, zaś wartości leżące poza tym zakresem są reprezentowane przez punkty.

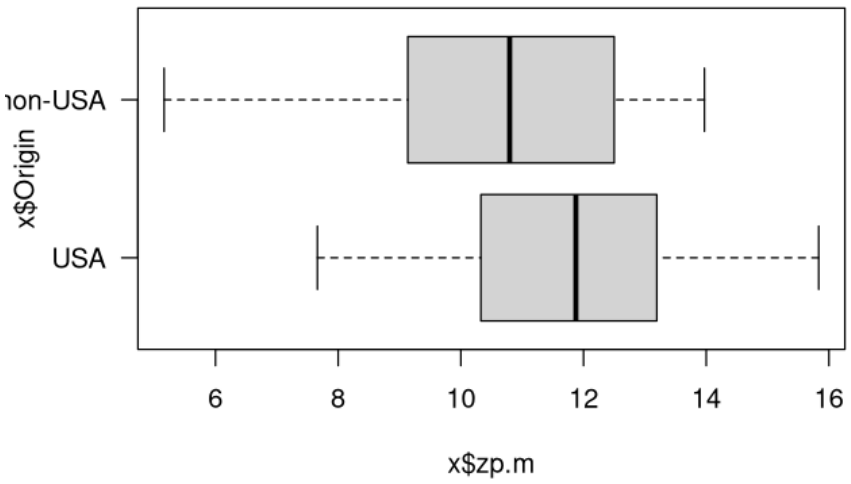
Następne 2 wywołania funkcji `boxplot` obrazują te same dane z podziałem na pochodzenie samochodów - służy do tego operator tyldy, który powoduje podział danych względem wszystkich kategorii typu czynnikowego `x$Origin` (typ czynnikowy to typ, który może przyjąć jedną z predefiniowanych wartości, w tym przypadku „USA" i „non-USA").

Funkcja ta wyznacza tablicę liczebności dla jednej, dwóch lub większej liczby zmiennych wyliczeniowych. W przypadku zmiennych jakościowych podobny efekt co funkcja `table()` ma funkcja `summary()`. Różnica polega na tym, że w razie występowania danych `NA` funkcja `table()` je ignoruje, a funkcja `summary()` wypisuje ich liczbę.

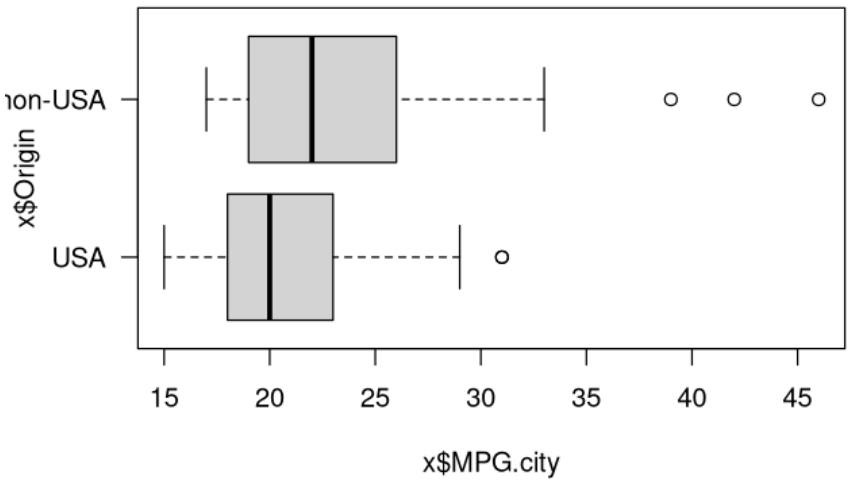
Dodatkowo możemy użyć `sort` aby posortować wartości, domyślnie rosnąco. Do narysowania wykresu słupkowego służy funkcja `barplot`, a kołowego `pie`.



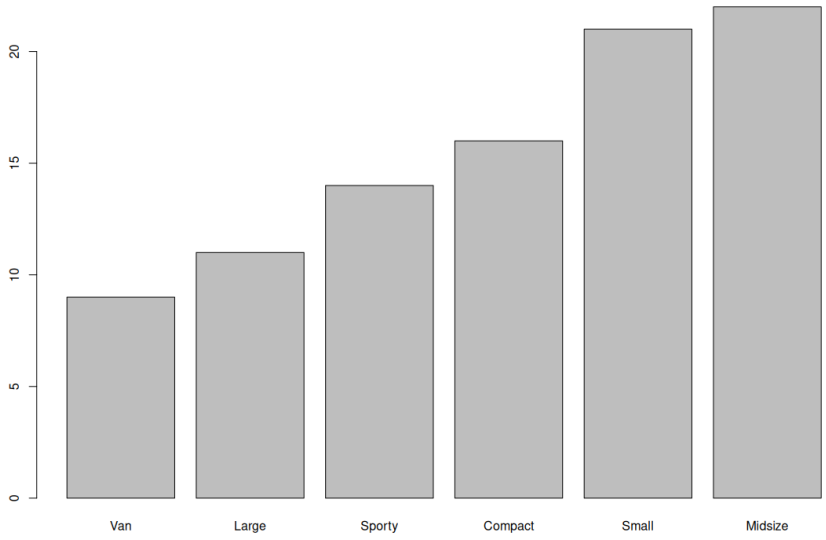
Rysunek 8: Wykres pudełkowy zużycia paliwa (w litrach na 100 km) wszystkich badanych aut



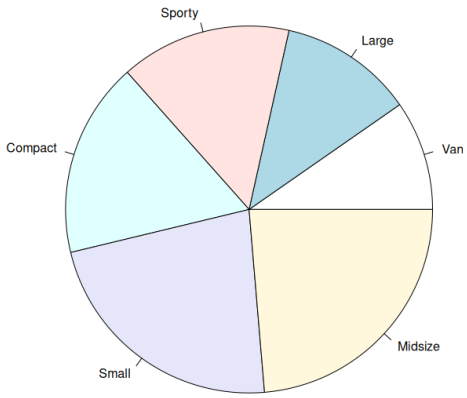
Rysunek 9: Wykres pudełkowy zużycia paliwa (w litrach na 100 km) wszystkich badanych aut z podziałem na pochodzenie aut



Rysunek 10: Wykres pudełkowy zużycia paliwa (w milach na galon) wszystkich badanych aut z podziałem na pochodzenie aut



Rysunek 11: Wykres słupkowy liczebności aut z poszczególnych kategorii



Rysunek 12: Wykres kołowy liczebności aut z poszczególnych kategorii

Pełny kod rozwiązania:

```
rm(list = ls())

library(MASS)
x = Cars93
dim(x)
head(x)
summary(x)

zp.m = 380/1.6/x$MPG.c
x$zp.m = 380/1.6/x$MPG.city

quantile(x$zp.m, 0.95)

boxplot(x$zp.m)

b = boxplot(x$zp.m~x$Origin, las=1, horizontal = TRUE)
b1 = boxplot(x$MPG.city~x$Origin, las=1, horizontal = TRUE)

t = table(x$Type)
t = sort(t)
barplot(t)

pie(t)
```

Zadanie 7. Wygenerować 10000 prób 10-elementowych z rozkładu normalnego. Następnie zakładając, iż o próbach wiemy tylko tyle, że pochodzą one z rozkładu normalnego o nieznanych parametrach, wyznaczyć dla każdej próby przedział ufności dla wartości oczekiwanej na poziomie ufności 0.95. Porównać frakcję pokryć przez przedziały ufności faktycznej wartości oczekiwanej z założonym poziomem ufności.

Rozwiązanie: W poniższym kodzie przedstawiono 2 sposoby rozwiązania tego zadania. W pierwszym, używamy `for`, w którym dla każdego `i` generujemy 10-elementową próbę z rozkładu normalnego używając funkcji `rnorm`, a następnie używamy `t.test`. Jednym z wyników tej funkcji jest pole `$conf`, które przechowuje lewy i prawy kraniec przedziału ufności. Jeżeli wartość oczekiwana znajduje się w przedziale ufności to zwiększamy zmienną `counter`. W drugim robimy to samo używając funkcji `replicate`, która jest znacząco szybsze. Funkcja `replicate` zwróci nam wektor wartości logicznych. Liczbę `TRUE` można policzyć poprzez użycie `sum` - traktuje ona `TRUE` jako 1, a `FALSE` jako 0.

Oba sposoby skutkują uzyskaniem wyniku 0.9523, co oznacza że około w 95.23% przypadków wartość oczekiwana μ trafiła do przedziału ufności. Jest to w dużej mierze zgodne z oczekiwanym wynikiem 95%.

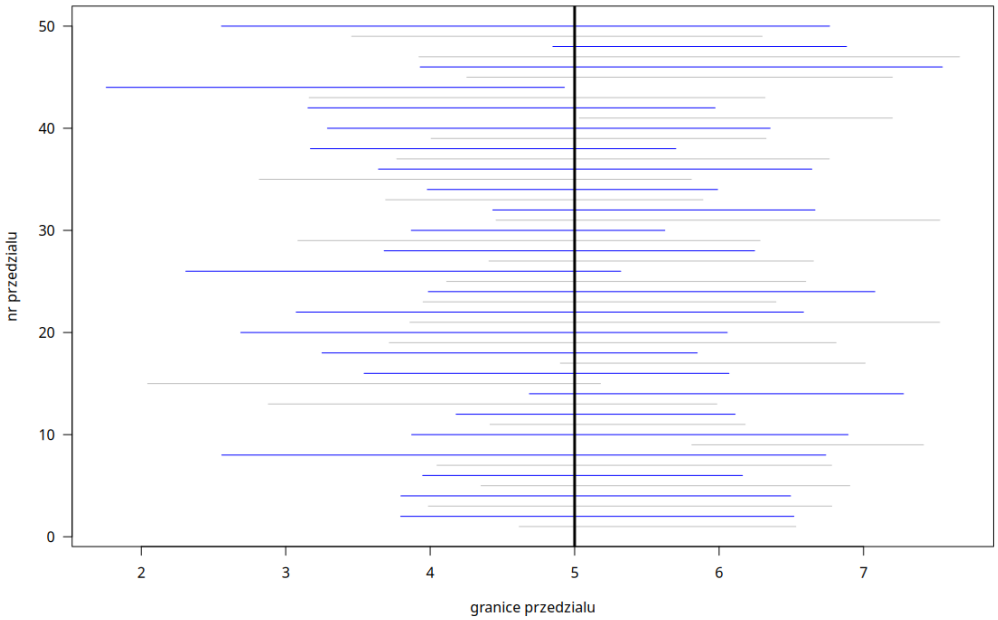
Pełny kod rozwiązania:

```
rm(list = ls())

N = 10^4
n = 10
mi = 5
sigma = 2
alpha = 0.05

counter = 0
for (i in 1:N) {
  x = rnorm(n, mean=mi, sd=sigma)
  range = t.test(x, conf.level=1-alpha)$conf
  if(mi < range[2] && mi > range[1]){
    counter = counter + 1
  }
}
result = counter/N

# alternative - use of replicate
hits = replicate(N,
                    {
                      x = rnorm(n, mean=mi, sd=sigma)
                      range = t.test(x, conf.level=1-alpha)$conf
                      mi < range[2] && mi > range[1]
                    })
result2 = sum(hits)/N
```

Zadanie 8. Wybrać $\mu \in \mathbb{R}$, $\sigma > 0$. Wygenerować $N = 50$ prób n-elementowych ($n = 10$) z rozkładu $N(\mu, \sigma)$ i dla każdej z nich utworzyć przedział ufności dla μ na poziomie ufności 0.95. Przedstawić na jednym wykresie uzyskane przedziały ufności. Ile z nich powinno zawierać μ?

Rozwiązanie: W tym zadaniu również używamy funkcji `replicate`, tym razem jednak zwróci ona wektor N przedziałów ufności dla 10-elementowych prób z rozkładu $N(\mu = 5, \sigma = 2)$. Warto zauważyć w jaki sposób wybieramy te wartości przedziałów - używamy operatora `$` oraz nazwy `conf` - odwoła się ona do `conf.int` - nie trzeba pisać pełnej nazwy komponentu, o ile `R` jest w stanie jednoznacznie stwierdzić o jaki komponent nam chodzi.

Dalej wizualizujemy wyniki przy pomocy funkcji `matplot`. Jako pierwszy argument (oś X) ustawiamy macierz $2 \times n$ pochodzącą z `result` (pierwszy wiersz to wektor początków przedziałów ufności, a drugi to wektor końców). Na osi Y odkładamy wektory $1 : N$. W ten sposób otrzymujemy poniższy wykres (kolory odcinków reprezentujących przedziały są naprzemienne dla czytelności - za co odpowiada ustawienie koloru (`col`) jako dwuelementowego wektora `('grey', 'blue')`):



Pionowa czarna linia odpowiada wartości $\mu = 5$. Spodziewamy się więc, że mniej-więcej 95% widocznych na wykresie poziomych odcinków przetnie się z tą linią. Patrząc na rysunek, widzimy, że 3 przedziały spośród 50 wygenerowanych nie zawierają wartości oczekiwanej, czyli 94% ją zawiera.

Pełny kod rozwiązania:

```
rm(list = ls())

N = 5e1
n = 1e1
mi = 5
sigma = 2
alpha = 5e-2

result = replicate(N,
                   {
                      x = rnorm(n, mi, sigma)
                      t.test(x, conf.level = 1-alpha)$conf
                   })
matplot(result, rbind(1:N, 1:N), type='l', lty=1,
        col=c('grey','blue'), las=1,
        xlab="granice przedzialu",
        ylab="nr przedzialu")
abline(v=mi, col=2)
```

Uwagi

- Opis wykresu pudełkowego wzięty z: https://pl.wikipedia.org/wiki/Wykres_pude%C5%82kowy
- Do generowania sekwencji można użyć funkcji **seq**, która jako parametry przyjmuje granice przedziału oraz opcjonalne argument **by** (krok) oraz **length.out** (pożądana długość sekwencji).