

Podstawowe wykresy w środowisku R

Agnieszka Goroncy



**UNIWERSYTET
MIKOŁAJA KOPERNIKA
W TORUNIU**

Wydział Matematyki
i Informatyki

Wykres służy do graficznej prezentacji danych.

Istnieje bardzo wiele typów wykresów, najpopularniejszymi są:

- punktowy,
- liniowy,
- słupkowy,
- kołowy,
- rozrzutu.

Środowisko R pozwala tworzyć zarówno podstawowe wykresy statystyczne, jak i mniej znane takie jak:

- skrzynkowy ("pudełko z wąsami"),
- łodyga - liście,
- kwantyl - kwantyl (poprzedni temat),
- skrzypcowy.

Wiele poleceń generujących wykresy posiada wspólne argumenty, m.in.

- `height`, `width` - wysokość i szerokość wykresu,
- `col` - określa kolor elementów wykresu (tj. słupków, wycinków koła itp.),
- `main`, `sub` - tytuł i podtytuł wykresu,
- `axes`, `axisnames` - odpowiada za wyświetlanie i opisanie osi wykresu, domyślnie: TRUE,
- `xlab`, `ylab` - umożliwia wprowadzenie własnego tekstowego opisu osi x i y wykresu,
- `add` - umożliwia wyświetlenie wykresu wspólnie z innym, wcześniej wygenerowanym wykresem, domyślnie: FALSE.

Kolory

W środowisku R dostępna jest bogata gama kolorystyczna. Lista 657 predefiniowanych kolorów jest dostępna po wywołaniu funkcji

```
> colors()
```

Do konkretnej barwy można się odwoływać poprzez jej nazwę, np. "cyan" lub numer, np. 68. Tworząc wykres, mamy możliwość zdefiniowania kolorystyki jego składowych (słupków, wycinków koła), np. ustawiając argument `col` odpowiedniej funkcji generującej wykres:

- `col=rainbow(5)` - elementy wykresu będą kolorowane naprzemiennie pięcioma kolorami tęczy,
- `col=45` - wszystkie elementy wykresu będą w jednakowym odcieniu koloru niebieskiego ("cadetblue3"),
- `col=c(93,2,"darkgrey")` - elementy wykresu będą kolorowane naprzemiennie trzema kolorami.

Legenda wykresu

Dodanie legendy do wykresu jest możliwe poprzez wywołanie funkcji **legend()**.

Położenie legendy na wykresie może być zrealizowane poprzez:

- argument `x` (lub `x,y`) zawierający współrzędne położenia lub tekst: `"bottomright"`, `"bottom"`, `"bottomleft"`, `"left"`, `"topleft"`, `"top"`, `"topright"`, `"right"`, `"center"`.
- argument `locator(1)` - umożliwia umieszczenie legendy na wykresie we wskazanym myszką miejscu.

Pozostałe ważne parametry:

- `bty` - określa czy legenda ma być obramowana; możliwe wartości: `"o"` - ramka (domyślnie), lub `"n"` - brak ramki,
- `bg` - jeżeli `bty="o"`, to pozwala ustawić tło ramki z legendą,
- `cex` - określa wielkość czcionki w legendzie,
- `fill` - pozwala wyświetlić kwadraciki przy opisie i przypisać im wybrane kolory, np. `fill=rainbow(4)`.

Uwaga: Funkcję `legend()` należy wywołać **po** tym, jak R wyświetli wykres.

Funkcja plot()

Podstawową i najbardziej uniwersalną funkcją generującą wykresy w R jest funkcja **plot()**. Może ona dawać różne rezultaty, w zależności od podanych parametrów.

Najważniejszym parametrem funkcji plot jest **type** - typ wykresu, jeden z:

- "p" - punktowy,
- "l" - liniowy,
- "b" - punktowo-liniowy,
- "s" - schodkowy,
- "S" - schodkowy, odwrócony,
- "h" - kreskowy ("histogramowy").

Funkcja `plot()` - przykłady wykresów statystycznych

Przeanalizujemy bazę danych `mcycle` dołączoną do pakietu `MASS`.

```
> library(MASS)
```

Zawiera ona pomiary przyspieszenia głowy w symulowanym wypadku motocyklowym, wykorzystywane do testowania kasków.

Jednowymiarowy wykres punktowy czasu dla każdej obserwacji:

```
> plot(mcycle$times)
```

Dwuwymiarowy wykres punktowy (rozrzutu) zależności czasu od przyspieszenia

```
> plot(mcycle$times, mcycle$accel)
```

```
> plot(mcycle$times, mcycle$accel, type="p")
```

Dwuwymiarowy wykres liniowy zależności czasu od przyspieszenia:

```
> plot(mcycle$times, mcycle$accel, type="l")
```

Dwuwymiarowy wykres schodkowy zależności czasu od przyspieszenia:

```
> plot(mcycle$times, mcycle$accel, type="s")
```


Funkcja `par()`

Funkcja **`par()`** wywołana przed funkcją `plot()` umożliwia przekazanie bądź pobranie parametrów graficznych. Często używane argumenty:

- `mfcol`, `mfrow` - wektor określający liczbę wierszy i kolumn, które określają tablicę, na której rysowane będą poszczególne wykresy.

Przykład: Zanim wygenerujemy kolejne wykresy, wywołajmy polecenie

```
> par(mfcol=c(1,2))
```

Funkcja `plot()` - wykresy dowolnych funkcji

Funkcja `plot()` może być wykorzystywana również do generowania wykresów funkcji, np.

```
> plot(cos,-2*pi,2*pi)
> plot(tan,-pi/3,pi/3)
```

jak również wykresów dystrybuant i gęstości znanych rozkładów.

Wykres słupkowy

Funkcja **barplot()** pozwala wygenerować pionowy (domyślnie) lub poziomy wykres słupkowy.

W zależności od postaci argumentu głównego, możliwe są następujące **rodzaje wykresów słupkowych**:

- **obrazujący wartości poszczególnych obserwacji** - jeden słupek odpowiada jednej obserwacji; jako argument główny podajemy **nazwę zmiennej**, dla której chcemy zilustrować wartości obserwacji,
- **obrazujący grupy obserwacji** - jeden słupek odpowiada jednej kategorii; jako argument główny podajemy **szereg rozdzielnicy** (wynik działania funkcji `table()`) kategoryzujący zmienną, dla której chcemy zilustrować grupy obserwacji.

Możliwe jest uzyskanie również wykresów **zestawionych/zgrupowanych w podziale na inną zmienną**. Wówczas argumentem funkcji jest tablica krzyżowa dla dwóch zmiennych.

Funkcja `barplot()`: argumenty

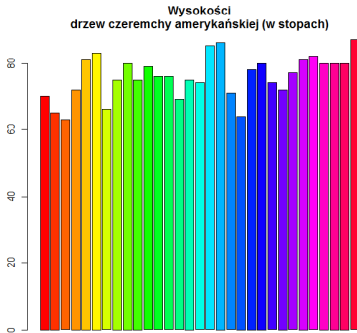
Parametry specyficzne dla funkcji `barplot`:

- `horiz` - poziome położenie słupków, domyślnie: `FALSE`,
- `beside` - określa sposób ułożenia słupków w przypadku podziału ze względu na kategorie innej zmiennej. Jeżeli `beside=TRUE`, to słupki są zgrupowane obok siebie, jeżeli `beside=FALSE` (domyślnie), to słupki zestawione są jeden na drugim.

Wykres słupkowy wartości obserwacji: przykład

Baza danych *trees* zawiera informacje dotyczące pomiarów 31 powalonych drzew czeremchy amerykańskiej. Poniższe polecenie umożliwia wygenerowanie wykresu słupkowego dla wysokości poszczególnych drzew:

```
> barplot(trees$Height, col=rainbow(31), main="Wysokości drzew czeremchy  
+ amerykańskiej (w stopach)")
```



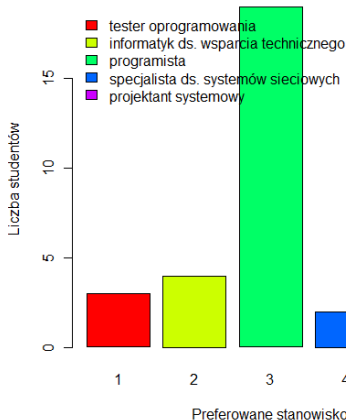
Wykres słupkowy dla grup obserwacji: przykład

Plik `ankieta_studenci_2011.csv` zawiera informacje dotyczące studentów 3-go roku informatyki WMil, zebrane latem 2011 roku. Poniższe polecenia umożliwiają wczytanie pliku i wygenerowanie wykresu słupkowego dla grup obserwacji wyznaczonych przez zmienną `stanowisko` (preferencje dotyczące stanowiska, które studenci chcieliby objąć w przyszłości).

```
> studenci<-read.csv("ankieta_studenci_2011.csv",  
+ header=TRUE, sep=";")  
> attach(studenci)  
> arg<-table(stanowisko)  
> barplot(arg, col=rainbow(5), xlab="Preferowane stanowisko",  
+ ylab="Liczba studentów")  
> legend("topleft", c("tester oprogramowania","informatyk  
+ ds. wsparcia technicznego","programista","specjalista ds.  
+ systemów sieciowych","projektant systemowy"), cex=1, bty="n",  
+ fill=rainbow(5))
```

Wykres słupkowy dla grup obserwacji

W rezultacie otrzymujemy taki oto wykres:



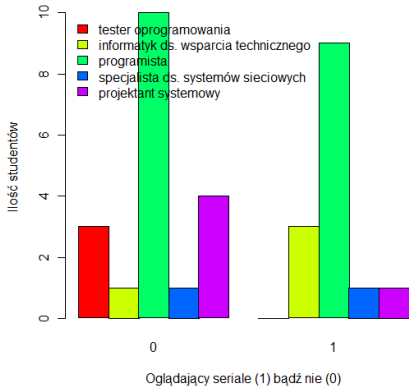
Wykres słupkowy zgrupowany: przykład

Zilustrujemy liczbę studentów preferujących poszczególne stanowiska, **zgrupowaną** ze względu na to, czy student ogląda seriale, czy nie.

```
> arg<-table(stanowisko, seriale)
> barplot(arg, col=rainbow(5),beside=T,xlab="Oglądający
+ seriale (1) bądź nie (0)",ylab="Ilość studentów")
> legend("topleft", c("tester oprogramowania","informatyk
+ ds. wsparcia technicznego","programista","specjalista ds.
+ systemów sieciowych","projektant systemowy"), cex=1, bty="n",
+ fill=rainbow(5))
```


Wykres słupkowy zgrupowany

W rezultacie otrzymujemy taki oto wykres:



Wykres słupkowy zestawiony: przykład

Ponownie zilustrujemy liczbę studentów preferujących poszczególne stanowiska, ale teraz **zestawioną** ze względu na to, czy student ogląda seriale, czy nie. Zamienimy też miejscami zmienne w funkcji `table()`.

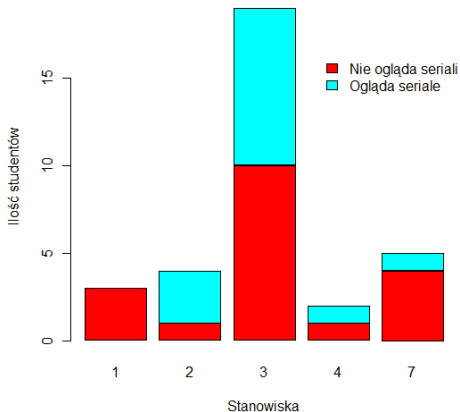
```
> arg<-table(seriale,stanowisko)
> barplot(arg, col=rainbow(2), xlab="Stanowiska")
> legend(locator(1), c("Nie ogląda seriali", "Ogląda seriale"),
+ cex=1, bty="n",fill=rainbow(2))
```

Uwaga: Położenie legendy wskazujemy myszką klikając w odpowiednie miejsce na wykresie.

Zadanie: Wygeneruj analogiczny wykres zgrupowany.

Wykres słupkowy zestawiony

W rezultacie otrzymujemy taki oto wykres (numeracja pod słupkami odpowiada numerom możliwych stanowisk w ankiecie):



Histogram

Histogram jest szczególnym rodzajem wykresu słupkowego, który umożliwia podgląd rozkładu empirycznego badanej cechy, która jest odpowiednio kategoryzowana. Słupków jest tyle, ile kategorii (klas). Są one zawsze tej samej szerokości i odkładane w równoodległych punktach na osi poziomej. Wysokości słupków mogą odpowiadać liczebnościom poszczególnych klas, bądź ich prawdopodobieństwom empirycznym (częstościom względnym).

Funkcja `hist()` pozwala wygenerować histogram.

Ważne parametry:

- `breaks` - parametr określający sposób wyznaczania klas na histogramie, może to być jeden z następujących argumentów:
 - wektor zawierający punkty podziału,
 - liczba określająca ilość klas,
 - nazwa algorytmu, który wylicza liczbę klas,
 - funkcja obliczająca liczbę klas.

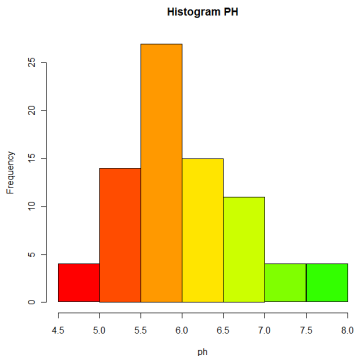
Parametr ten domyślnie ustawiony jest na "Sturges" (nazwa algorytmu),

- `freq` - na osi pionowej odkładane są liczebności poszczególnych klas (domyślnie TRUE),
- `prob` - przeciwieństwo `freq`: na osi pionowej odkładane są częstości względne (prawdopodobieństwa empiryczne) dla poszczególnych klas (domyślnie FALSE).

Histogram - przykład

Pakiet `boot` zawiera m.in. bazę danych `urine`. Histogram dla zmiennej `ph` z tej bazy możemy wygenerować następująco:

```
> library(boot)
> hist(urine$ph,main="Histogram PH",col=rainbow(20))
```



Wykres łodyga-liście

Funkcja **stem()** pozwala wygenerować wykres łodyga-liście. Jest to wykres tekstowy ilustrujący wartości zmiennej.

Ważne parametry:

- `scale` - określa skalę wykresu (domyślnie 1).
- `width` - szerokość wykresu (domyślnie 80),

Przykład:

```
> a=c(1.2, 2.4, 3.4, 3.6, 4.4, 4.5, 5.4, 5.5,  
+ 5.5, 6.3, 6.4)  
> stem(a)  
> stem(a, scale=2)
```

Porównaj wyniki działania dwóch ostatnich poleceń.

Wykres kołowy

Wykres kołowy pozwala **porównać proporcje** poszczególnych frakcji w stosunku do całości. Jest to bardzo popularny typ wykresu, z ponad dwusetletnią historią.

Funkcja **pie()** pozwala wygenerować wykres kołowy.

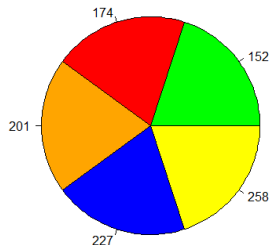
Podobnie jak w przypadku wykresu słupkowego, głównym argumentem może być albo zmienna (wówczas wycinki koła będą odpowiadały **wartościom poszczególnych obserwacji**), albo szereg rozdzielnicy, czyli wynik funkcji `table()` (wówczas wycinki koła będą odpowiadały **kategoriom wyznaczonym przez zmienną**).

Uwaga: Jeżeli chcemy porównywać **proporcje frakcji między sobą**, bardziej odpowiednim jest np. wykres słupkowy. Wykres kołowy nie nadaje się również do przedstawiania nieskategoryzowanych danych (gdy niemal każda obserwacja ma inną wartość).

Wykres kołowy - przykład

Przykład: Zobrazujemy na wykresie kołowym czas pomiaru (w dniach) pewnej odmiany świerku (pakiet MASS, zmienna Time w bazie Sitka):

```
> time<-Sitka$Time  
> czas<-table(time)  
> pie(czas,col=c("green","red","orange",  
+ "blue","yellow"))
```



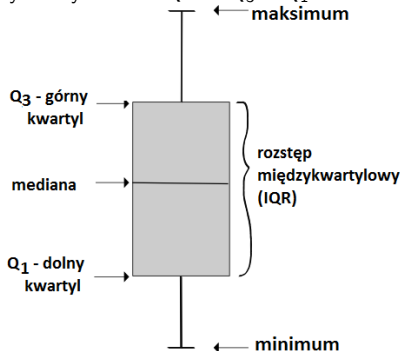
"Na oko" wydaje się, że wycinki wykresu kołowego są jednakowe. Czy na pewno? Jeżeli chcemy porównać częstości pomiarów między sobą, lepszy rezultat w tym przypadku da nam wykres słupkowy:

```
> barplot(czas)
```

Wniosek: Wycinki koła są rzeczywiście jednakowe.

Wykres pudełkowy

Wykres skrzynkowy (tzw. "pudełko z wąsami") pozwala przedstawić parametry położenia i rozproszenia rozkładu empirycznego badanej cechy. Składa się on z pudełka i "wąsów". Dolna część pudełka rysowana jest na wysokości dolnego kwartyła ($Q_1 = \text{quantile}(0.25)$), górna część na wysokości górnego kwartyła ($Q_3 = \text{quantile}(0.75)$), a linia w środku odpowiada medianie ($Q_2 = \text{Med}$). Wysokość pudełka zawsze odpowiada więc rozstępowi międzykwartylowemu $IQR = Q_3 - Q_1$.



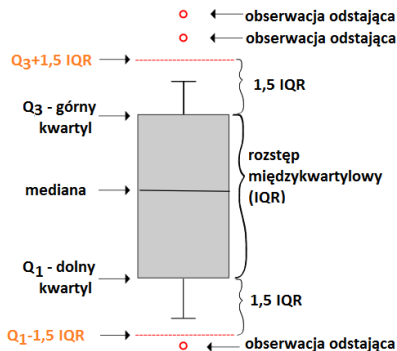
Długości wąsów mogą być różne, w zależności od tego, czy bierzemy pod uwagę obserwacje odstające (skrajne, czyli podejrzanie niskie bądź wysokie), czy nie. **Pudełko z wąsami nie uwzględniające obserwacji odstających wygląda tak:**

Wykres pudełkowy (2)

Wykres pudełkowy może **uwzględniać obserwacje odstające**. W tym przypadku budowa pudełka jest identyczna jak wcześniej, jednak wąsy są odpowiednio obcinane.

Górny wąs: do górnego kwartyla dodajemy 1,5 (standardowo) rozstępu międzykwartylowego (czyli IQR) i wszystkie obserwacje przekraczające tę wielkość oznaczamy jako odstające (kółko). Wąs ucinamy na wysokości największej obserwacji, która znajduje się poniżej poziomu $Q_3 + 1,5 \cdot \text{IQR}$.

Dolny wąs: od dolnego kwartyla odejmujemy 1,5 (standardowo) rozstępu międzykwartylowego i wszystkie obserwacje znajdujące się poniżej tej wielkości oznaczamy jako odstające (kółko). Wąs ucinamy na wysokości najmniejszej obserwacji, która znajduje się powyżej poziomu $Q_1 - 1,5 \cdot \text{IQR}$.



Wykres pudełkowy: parametry i przykład

Funkcja **boxplot()** pozwala wygenerować wykres skrzynkowy.

Ważne parametry funkcji:

- **range** - określa zakres długości wąsów. Dodatnia wartość range (domyślnie wynosi ona 1,5) pomnożona przez IQR wyznacza granicę długości wąsów. Wartość 0 powoduje wygenerowanie wykresu skrzynkowego bez uwzględnienia obserwacji odstających.
- **horizontal** - wartość logiczna określająca, czy wykres ma być ułożony poziomo (domyślnie FALSE).

Przykład:

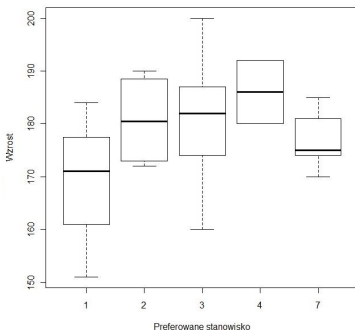
```
> x<-c(1,-3,-1,2,4,6,7,9,12,15,25,30,-16)
> summary(x)
> boxplot(x)
> boxplot(x, range=0)
```

Wykresy pudełkowe w rozbiciu na kategorie wyznaczone przez inną zmienną

Aby narysować wykresy pudełkowe dla zmiennej liczbowej w rozbiciu na kategorie wyznaczone przez inną zmienną (np. jakościową), należy użyć znaku `~`.

Przykład: Zbadajmy wzrost studentów z pliku `ankieta_studenci_2011.csv` w rozbiciu na stanowisko.

```
> boxplot(studenci$wzrost~studenci$stanowisko)
```



Agregacja danych

Agregacja danych polega na wyliczeniu statystyki (jednej lub wielu, np. średniej) dla grup obserwacji wyznaczonych przez kategorie zmiennej grupującej (lub wielu zmiennych grupujących). Do agregacji danych służy funkcja **aggregate()**, której pierwszym argumentem jest obiekt, który będzie agregowany (wektor, ramka danych itp.). Kolejne ważne argumenty to:

- **by** - lista (!) elementów, względem których odbędzie się grupowanie (zmiennych grupujących),
- **FUN** - funkcja podsumowująca dane, np. **mean** - średnia, **weighted.mean** - średnia ważona, **max** - maksimum, itp.

Przykład: Baza danych `mtcars` zawiera dane dotyczące samochodów. Chcemy obliczyć średnie zużycie paliwa (zmienna `mpg`) dla samochodów w grupach wyznaczonych przez liczbę cylindrów (zmienna `cyl`):

```
> aggregate(mtcars$mpg, by=list(mtcars$cyl), FUN=mean)
```

Aby wyznaczyć maksimum wszystkich zmiennych z bazy danych, w grupach wyznaczonych zarówno przez liczbę cylindrów, jak i typ skrzyni biegów (zmienna `am`), należy wywołać polecenie

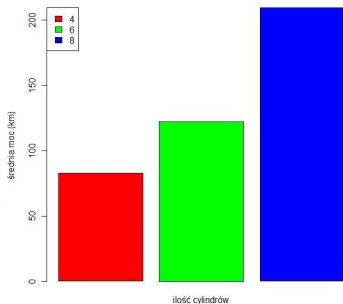
```
> aggregate(mtcars, by=list(mtcars$cyl, mtcars$am), FUN=max)
```

Wykresy dla danych zagregowanych

Przykład: Chcemy zobrazować na wykresie słupkowym średnią moc samochodu (zmienna `hp`) w zależności od liczby cylindrów w samochodzie. W tym celu najpierw agregujemy dane.

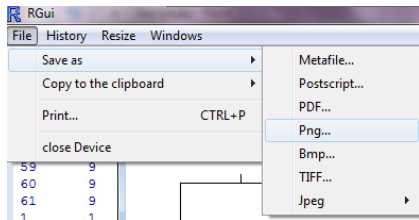
```
> dane=aggregate(mtcars$hp, by=list(cyl), FUN=mean)
> barplot(dane$x, col=rainbow(length(dane$x)))
> legend("topleft", names(table(mtcars$cyl)),
+ fill=rainbow(length(dane$x)))
```

Z wykresu wynika, że średnia moc samochodu jest najwyższa wśród samochodów posiadających 8 cylindrów.



Eksport wykresów do plików






Najprostszy sposób na zapisanie uzyskanego wykresu w pliku to wybór odpowiedniego elementu z menu:



Można to zrobić również z poziomu konsoli, wywołując odpowiednią funkcję, która zapisuje aktualnie wyświetlaną grafikę w wybranym formacie, w pliku o nazwie podanym jako jej argument:

- `dev.print("nazwa.jpeg",device=jpeg,width=600),`
- `dev.print("nazwa.png",device=png,width=600),`
- `dev.print("nazwa.pdf",device=pdf,width=6).`

Literatura i przydatne zasoby w internecie

-  Grzegorz Skalski (Uniwersytet w Łodzi), **Wizualizacja danych: odkrycie pierwszych zależności**
-  Paul Murrell (Uniwersytet w Auckland, Nowa Zelandia), **Grafika w R**
-  Hans Rosling, **The Joy of Stats**
-  Przemysław Biecek, **Przewodnik po pakiecie R**, Oficyna Wydawnicza GiS, Wrocław, 2011
-  Joseph Adler, **R in a Nutshell**, O'Reilly Media, 2009