Programowanie w języku R
Zestaw zadań

Marek Gągolewski

Autorem zadań oznaczonych przez [AO] jest dr Anna Olwert, a tych oznaczonych przez [BT] – p. Bartłomiej Tartanus. Rozwiązania wybranych zadań znajdują się na stronie rksiazka.rexamine.com.

Spis treści

2 Podstawowe typy atomowe

Zadanie 2.1. Wymień wszystkie poznane funkcje, które mogą służyć do tworzenia wektorów liczbowych.

Zadanie 2.2. Co będzie wynikiem rzutowania stałych (każdej oddzielnie) `FALSE`, `TRUE`, `1L`, `1.5`, `-0.5`, `'123'` do omówionych w tym rozdziale typów wektorów atomowych? Czy jeśli wszystkie umieścimy w jednym wektorze, to zauważymy jakąś różnicę?

Zadanie 2.3. Przeczytaj na stronie dokumentacji `?rep` na temat zachowania się tej funkcji, gdy podany zostanie jej `NULL` jako jeden z argumentów.

Zadanie 2.4. Jaka postać wywołania funkcji `seq()` odpowiada standardowemu zachowaniu się funkcji `seq_along()` i `seq_len()` (które poznasz studiując R-ową dokumentację).

Zadanie 2.5. Co się stanie, gdy jako argument funkcji `cat()` podamy wektor składający się z trzech napisów? Przeczytaj w dokumentacji, jak można zmodyfikować zachowanie się tej funkcji.

3 Operacje na wektorach

Uwaga. Wszystkie poniższe zadania da się rozwiązać nie używając pętli. Jeśli więc już znasz te wyrażenia sterujące R-a – spróbuj pomyśleć, jak obyć się bez nich.

Zadanie 3.1. Dla danego wektora liczbowego `x` wykonaj następujące operacje.

- Wypisz na ekran wszystkie wartości ze zbioru $[-2, -1] \cup [1, 2]$.
- Wypisz na ekran liczbę wszystkich wartości nieujemnych.
- Wyznacz średnią arytmetyczną wartości bezwzględnych elementów.
- Wyznacz wartość najbliższą i najdalszą od 0 (zachowując jej znak).
- Wyznacz wartość najbliższą i najdalszą od 2 (zachowując jej znak).
- Wypisz na ekran część ułamkową wszystkich elementów.
- Wypisz na ekran wektor powstały w wyniku przekształcenia liniowego wartości z `x` na przedział $[0, 1]$ (najmniejsza wartość staje się równa 0, a największa 1).
- Utwórz wektor napisów `y` o długości takiej samej, jaką ma `x`, dla którego y_i przyjmuje wartość `'nieujemna'`, jeśli x_i jest nieujemne oraz `'ujemna'` w przeciwnym przypadku.
- Utwórz wektor liczbowy `y` o długości takiej samej, jaką ma `x`, dla którego y_i przyjmuje wartość $k/2$ wtedy i tylko wtedy, gdy $x_i \in [k, k+1)$, gdzie $k \in \mathbb{Z}$ (prosty histogram).

W celach testowych użyj np. losowego wektora:

```
x <- round(rnorm(20, 0, 1), 2)
```

Zadanie 3.2. [AO] Odgadnij, w wyniku jakiego przekształcenia wektora `x` otrzymano dany wektor `y`.

- `x : -5 -4 -3 -2 -1 0 1 2 3 4 5`,
  `y : 0 1 0 1 0 1 0 1 0 1 0`,

- x : 1 3 -2 -1 0 -3 -5 4 -4 2,
  y : 2 8 4 10 -3 0 -1 4 3 -2,
- x : 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16,
  y : 0 8 24 48 80 120 168 224.

Zadanie 3.3. [AO] W statystycznej kontroli jakości, gdy obserwacje sterowanego procesu zbierane są w równych odstępach czasu, tzw. składnik losowy zmienności szacuje się często za pomocą współczynnika

$$\ddot{s}_{\mathtt{x}}^2 = \frac{1}{2(n-1)} \sum_{i=1}^{n-1} (x_{i+1} - x_i)^2,$$

gdzie x_i jest obserwacją pobraną w chwili i, $i = 1, 2, \ldots, n$. Dla danego wektora liczbowego `x` wyznacz wartość $\ddot{s}_{\mathtt{x}}$.

Zadanie 3.4. Mamy dane dwa wektory liczbowe `x` i `y` tej samej długości równej n. Wypisz na ekran (1 linijka kodu) wartość współczynnika korelacji r Pearsona, będącego miarą liniowej zależności między poszczególnymi parami obserwacji (x_i, y_i) dla $i = 1, \ldots, n$.

$$r(\mathtt{x}, \mathtt{y}) = \frac{1}{n-1} \sum_{i=1}^{n} \frac{x_i - \bar{\mathtt{x}}}{s_{\mathtt{x}}} \frac{y_i - \bar{\mathtt{y}}}{s_{\mathtt{y}}},$$

gdzie $\bar{\mathtt{x}}, \bar{\mathtt{y}}$ oznacza średnią arytmetyczną, a $s_{\mathtt{x}}, s_{\mathtt{y}}$ – odchylenie standardowe, odpowiednio, wektorów `x` i `y`. Warto zauważyć, że $r(\mathtt{x}, \mathtt{y}) \in [-1, 1]$.
W celach testowych użyj następujących wektorów:

- `x <- rnorm(20, 0, 1); y <- 10*x+2`,
- `x <- rnorm(20, 0, 1); y <- -4*x+1`,
- `x <- rnorm(2000, 0, 1); y <- rnorm(2000, 5, 2)`.

Zadanie 3.5. [AO] Dla danego wektora liczbowego `x` o długości n wyznacz wartość współczynnika asymetrii danego wzorem $\frac{n_1 - n_2}{n}$, gdzie n_1 i n_2 oznaczają liczbę obserwacji z `x`, odpowiednio, mniejszych i większych od $\bar{\mathtt{x}}$. Na marginesie, jeśli średnia arytmetyczna jest równa medianie, to wartość tego współczynnika wynosi 0.

Zadanie 3.6. Dla danego wektora liczbowego `x` o parzystej długości, utwórz wektor składający się z mediany, minimum i maksimum z `x`, nie korzystając z wbudowanych funkcji agregujących `median()`, `min()`, `max()` oraz `quantile()`.

Zadanie 3.7. [AO] Obserwacje w próbie zwykliśmy nazywać odstającymi, jeśli przyjmują wartości większe niż $Q_3 + 1{,}5\,\mathrm{IQR}$ lub mniejsze niż $Q_1 - 1{,}5\,\mathrm{IQR}$, gdzie Q_1 i Q_3 oznaczają kwantyle rzędu $0{,}25$ i $0{,}75$ (pierwszy i trzeci kwartyl), a $\mathrm{IQR} = Q_3 - Q_1$ jest tzw. rozstępem międzykwartylowym. Dla danego wektora liczbowego `x` wyznacz liczbę wartości odstających.

Zadanie 3.8. Dla danego wektora liczbowego `x` o nieparzystej długości n i wartości $\mathtt{k} \leqslant \frac{n-1}{2}$ wyznacz tzw. średnią k-uciętą, tj. średnią arytmetyczną z podwektora `x`, w którym pomijanych jest `k` najmniejszych i `k` największych elementów.

Zadanie 3.9. Dla danego wektora liczbowego `x` o nieparzystej długości n i wartości $\mathtt{k} \leqslant \frac{n-1}{2}$ wyznacz tzw. średnią k-winsorowską, tj. średnią arytmetyczną z podwektora `x`, w którym `k` najmniejszych i `k` największych elementów zostaje zastąpionych przez, odpowiednio, (`k+1`)-szą wartość najmniejszą i największą.

Zadanie 3.10. Dla danego wektora liczbowego `x` o długości n i n-elementowego wektora wag `w` takiego, że $w_i \geqslant 0$ i $\sum_{i=1}^{n} w_1 = 1$ wyznacz wartość tzw. operatora OWA (*ordered weighted averaging*), wg wzoru:

$$\mathrm{OWA}_{\mathtt{w}} = \sum_{i=1}^{n} x_{(i)} w_i,$$

gdzie $x_{(i)}$ jest i-tym najmniejszym elementem w `x`.

Zadanie 3.11. Dla danego wektora liczbowego `x` o długości n i n-elementowego wektora `c` wyznacz wartość tzw. operatora OWMax (*ordered weighted maximum*), wg wzoru:

$$\mathrm{OWMax}_{\mathtt{c}} = \bigvee_{i=1}^{n} x_{(n-i+1)} \wedge c_{(i)},$$

gdzie $\vee$, $\wedge$ oznaczają, odpowiednio, operatory maksimum i minimum.

Zadanie 3.12. Niech dany będzie wektor `x` o wartościach całkowitych. Wypisz na ekran wszystkie indeksy i, dla których $x_i = x_{i+1}$. Na przykład dla `x == c(1,1,0,1,1,1,0,0,1,1)` wynikiem powinno być 1, 4, 5, 7, 9.

Zadanie 3.13. [AO] W wektorze liczbowym `x`, zawierającym braki danych, zastąp wszystkie wartości `NA`:

- średnią arytmetyczną wszystkich dobrze określonych wartości z `x`,
- wartością średnią dwóch najbliższych sąsiednich wartości nie będących brakami danych (możesz założyć, że wartość `NA` sąsiaduje zawsze z dwoma elementami dobrze określonymi).

W celach testowych użyj np. `x=c(5, NA, 6, 2, 3, 5, 6, 4, NA, 2, NA, 5)`.

Zadanie 3.14. [AO] Dla danej wartości całkowitej `n` $\leqslant 16$ wyznacz wektor składający się z kolejnych `n` cyfr znaczących rozwinięcia dziesiętnego stałej wbudowanej `pi`. Na przykład dla `n == 10` powinniśmy otrzymać `c(3, 1, 4, 1, 5, 9, 2, 6, 5, 3)`.

Zadanie 3.15. Dane są dwa n-elementowe wektory liczbowe: `x` (posortowany rosnąco) i `p` (dla którego $p_i \geqslant 0$, $i = 1, \ldots, n$ oraz $\sum_{i=1}^{n} p_i = 1$). Para $(\mathtt{x}, \mathtt{p})$ określa rozkład prawdopodobieństwa dyskretnej zmiennej losowej X, jeśli założymy, że $\mathrm{P}(X = x_i) = p_i$.

Wartość oczekiwana zmiennej losowej X określona jest jako $\mathbb{E}\, X = \sum_{i=1}^{n} x_i\, p_i$, wariancja $\mathrm{Var}\, X = \mathbb{E}\, X^2 - (\mathbb{E}\, X)^2$, gdzie $\mathbb{E}\, X^2 = \sum_{i=1}^{n} x_i^2\, p_i$, a odchylenie standardowe $\sigma_X = \sqrt{\mathrm{Var}\, X}$. Ponadto dystrybuanta rozkładu zmiennej losowej X określona jest wzorem $F(y) = \mathrm{P}(X \leqslant y) = \sum_{i: x_i \leqslant y} p_i$.

Rozwiąż za pomocą R-a następujące zadanie:

Przemek bierze udział w loterii „Sukces", w której można zdobyć nagrody pieniężne o wartościach 10, 25 i 100 zł. Okazuje się, że prawdopodobieństwa wyciągnięcia poszczególnych losów wynoszą:

x_i	*0*	*10*	*25*	*100*
p_i	*80%*	*15%*	*4%*	*1%*

Wyznacz oczekiwaną wygraną, jej wariancję i odchylenie standardowe. Podaj wartości dystrybuanty rozkładu w punktach 0, 10, 25 *i* 100.

4 Listy

Zadanie 4.1. Dany jest wektor liczbowy `x`. Napisz kod, który utworzy listę zawierającą trzy, być może puste, wektory. Pierwszy element listy składa się ze wszystkich obserwacji z `x` mniejszych od 0, drugi – z obserwacji równych 0, a trzeci – z obserwacji większych od 0.

Zadanie 4.2. Napisz kod, który złączy wszystkie napisy z listy wektorów napisów w jeden napis.

Zadanie 4.3. Dana jest lista składająca się z wektorów atomowych. Znajdź indeks wektora o największej sumie elementów.

⋆ **Zadanie 4.4.** Dana jest lista składająca się z niepustych wektorów liczbowych. Napisz kod, który uporządkuje tę listę niemalejąco względem wartości pierwszych elementów tych wektorów.

5 Funkcje

Zadanie 5.1. [AO] Napisz funkcję `kwantyl()`, która dla danego n-elementowego wektora liczbowego `x` oraz wartości `p` $\in [0,1]$ zwróci wartość kwantyla rzędu `p` < 1. Wartość kwantyla rzędu `p` dana jest wzorem $x_{(\lfloor h\rfloor)} + (h - \lfloor h\rfloor)(x_{(\lfloor h\rfloor+1)} - x_{(\lfloor h\rfloor)})$, gdzie $h = (n-1)p + 1$, a $x_{(i)}$ oznacza i-tą najmniejszą wartość z `x`.

Zadanie 5.2. Napisz zwektoryzowaną wersję funkcji `kwantyl()` o nazwie `kwantyle()`, która pozwala na to, by argument `p` miał być może więcej niż jeden element. Zwracany powinien być wektor kwantyli zadanych rzędów, tj. taki, że `wynik[i] == kwantyl(x, p[i])`. Aby zyskać na wydajności, nie wywołuj wcale funkcji `kwantyl()` z zad. 5.1, tylko napisz ją od nowa.

Zadanie 5.3. Napisz funkcję, która obliczy wartość statystyki testowej W w teście rangowanych znaków Wilcoxona dla dwóch wektorów liczbowych `x` i `y` o tej samej liczbie elementów. Algorytm:

1. Usuń z wektorów wszystkie elementy leżące na indeksach j, dla których zachodzi $|y_j - x_j| = 0$.
2. Wyznacz wektor rang `r` elementów wektora $|$`y` $-$ `x`$|$.
3. Zwróć wartość $W = |\sum_i \operatorname{sign}(y_i - x_i) r_i|$.

Zadanie 5.4. Indeksem h Hirscha dla wektora `x` o n elementach nieujemnych nazywamy funkcję $\mathsf{H}(\mathbf{x}) = \max\{i = 0, 1, \ldots, n : x_{(n-i+1)} \geqslant i\}$, gdzie $x_{(i)}$ oznacza i-tą statystykę pozycyjną z `x`, tj. jego i-tą najmniejszą wartość, przy założeniu, że $x_{(n+1)} = x_{(n)}$. Można pokazać, że

$$\begin{aligned}
\mathsf{H}(\mathbf{x}) &= \left\lfloor \max\left\{ \min\{x_{(n)}, 1\}, \min\{x_{(n-1)}, 2\}, \ldots, \min\{x_{(1)}, n\} \right\} \right\rfloor \\
&= \left\lfloor \bigvee_{i=1}^{n} x_{(n-i+1)} \wedge i \right\rfloor.
\end{aligned}$$

Napisz funkcję `index.h()`, która wyznacza wartość indeksu h dla danego wektora.

Ciekawostka: Indeks h to „bombowy" pomysł chemika J.E. Hirscha z 2005 r., który miał rozwiązać problem oceny jakości dorobku naukowego. Autor(ka) ma indeks h równy i, jeśli i spośród n jego/jej publikacji uzyskało co najmniej i cytowań, a pozostałe $n - i$ prac jest cytowanych co najwyżej i razy.

6 Modyfikacja przepływu sterowania

Zadanie 6.1. Napisz własne implementacje funkcji `cumsum()`, `diff()`, `which()`, `which.min()` oraz `range()` działające dla argumentów będących niepustymi wektorami atomowymi. Za pomocą funkcji `microbenchmark2()` porównaj czas działania funkcji wbudowanych i odpowiadających im Twoich implementacji.

Zadanie 6.2. Wyznacz wartość współczynnika korelacji rangowej τ Kendalla dla dwóch wektorów liczbowych `x` i `y` o tej samej liczbie elementów n. Zakładamy, że żadne wartości w wektorze `x` ani w `y` nie powtarzają się. Mamy:

$$\tau = \frac{2c}{n(n-1)/2} - 1,$$

gdzie c oznacza liczbę wszystkich par zgodnych, czyli takich, że dla $1 \leqslant i < j \leqslant n$ zachodzi $x_i > x_j$ oraz $y_i > y_j$ bądź $x_i < x_j$ oraz $y_i < y_j$.

Zadanie 6.3. Napisz funkcję, która dla danej wartości całkowitej $k > 0$ i wektora całkowitego `x` o n elementach ze zbioru $\{1, 2, \ldots, k\}$ (jeśli taki nie jest podany, należy zgłosić błąd) zwróci wektor o długości k, w którym i-ty element jest równy liczbie wystąpień wartości i w `x`, $i = 1, \ldots, k$.

Zadanie 6.4. Wykorzystaj funkcję z zadania 6.3 (bez żadnych zmian w jej kodzie) do posortowania wektora liczb całkowitych `x`. Tak zwane *sortowanie kubełkowe* (ang. *bucket sort*) polega na zliczeniu występujących w ciągu powtarzających się wartości, a następnie odtworzeniu danego wektora przez powtórzenie odpowiednich liczb określoną liczbę razy.

I tak dla `c(5,3,2,3,1,2,3)` wiemy, że 1 występuje raz, 2 – dwa razy, 3 – trzy razy, 4 – zero razy, a 5 – raz. Na podstawie tej informacji od razu możemy wygenerować posortowany wektor: `c(1,2,2,3,3,3,5)`.

Tym razem nie zakładaj, że wartości w `x` należą koniecznie do zbioru $\{1, 2, \ldots, k\}$ – dostarcz funkcji z zad. 6.3 odpowiednio przekształcony wektor oraz poprawną wartość `k`. Opisany algorytm zaimplementuj w postaci funkcji o nazwie `bucketsort()`.

Zadanie 6.5. Napisz funkcję `podziel()`, która dla danego wektora liczbowego `x` o n elementach oraz wektora `a` o k-elementach zliczy, ile wartości z `x` wpada do każdego z przedziałów P_i, gdzie $P_1 = (-\infty, a_1)$, $P_{i+1} = [a_i, a_{i+1})$ dla $i = 1, \ldots, k-1$ oraz $P_{k+1} = [a_k, \infty)$. Wynik działania funkcji przedstaw w postaci $(k+1)$-elementowego wektora. Jeśli wektor `a` nie jest posortowany, należy użyć funkcji `sort()` przed przystąpieniem do zliczania.

Zadanie 6.6. Napisz funkcję `liniowa()`, która jako argumenty przyjmuje: (a) posortowany rosnąco n-elementowy wektor liczbowy `x`, (b) n-elementowy wektor liczbowy `y` (dowolny), (c) k-elementowy wektor liczbowy `z` o wartościach z przedziału $[x_1, x_n)$.

Funkcja ta powinna zwracać k-elementowy wektor, którego i-ty element jest wynikiem obliczenia wartości funkcji kawałkami liniowej (tj. łamanej) interpolującej liniowo punkty $(x_1, y_1), \dots, (x_n, y_n)$ w punkcie z_i.

Formalnie, jeśli $j \in \{1, \dots, n-1\}$ jest taki, że $x_j \leqslant z_i < x_{j+1}$, to i-tą wartością wynikową będzie $y_j + (y_{j+1} - y_j)(z_i - x_j)/(x_{j+1} - x_j)$.

Zadanie 6.7. Napisz funkcję `kombinuj()`, która jako parametr przyjmuje `k`-elementową listę zawierającą wektory liczbowe, wszystkie o tej samej długości `n`. Jeśli użytkownik dostarcza niepoprawne dane, zwracana jest wartość `NULL`. W przeciwnym przypadku zwracana jest lista zawierająca `n` wektorów liczbowych, każdy o `k` elementach. i-ty wektor powstaje przez złączenie i-tych elementów ze wszystkich wektorów wejściowych. Na przykład wynikiem wywołania `kombinuj(list(c(1,2), c(3,4), c(5,6)))` powinno być `list(c(1,3,5), c(2,4,6))`.

Zadanie 6.8. Napisz funkcję `podziel1()`, która dla danego wektora liczbowego `x` zwróci listę wektorów liczbowych składającą się z obserwacji z `x` podzielonych na klasy wg następującego algorytmu.

Niech $[a, b)$, $a, b \in \mathbb{Z}$ będzie najmniejszym przedziałem takim, że $(\forall i)\ x_i \in [a, b)$. j-ty element listy, $j = 1, \dots, b - a$, jest wektorem składającym się ze wszystkich elementów x_i takich, że $x_i \in [a + j - 1, a + j)$.

Zadanie 6.9. Liczby rozmyte czasem używane są do reprezentacji nieprecyzyjnie określonych danych ilościowych typu „około 3”, „prawie 10”, czy „co najmniej 100”. Trapezoidalna liczba rozmyta (TLR) $T = \mathrm{T}(a_1, a_2, a_3, a_4)$ określona jest za pomocą czterech liczb rzeczywistych $a_1 < a_2 \leqslant a_3 < a_4$. Jej funkcja przynależności jest postaci:

$$\mu_T(x) = \begin{cases} 0 & \text{dla } x \leqslant a_1 \text{ lub } x \geqslant a_4, \\ (x - a_2)/(a_2 - a_1) & \text{dla } x \in (a_1, a_2), \\ (x - a_4)/(a_3 - a_4) & \text{dla } x \in (a_3, a_4), \\ 1 & \text{dla } x \in [a_2, a_3]. \end{cases}$$

Wartość $\mu_T(x) = 1$ interpretuje się jako „x na pewno należy do T”, a $\mu_T(x) = 0$ jako „x na pewno nie należy do T”. Wartości pośrednie postrzega się jako częściowe przekonanie co do przynależności. Dla przykładu liczba rozmyta $\mathrm{T}(29, 30, 35, 40)$ może reprezentować wiek pewnej kobiety, która mówi o sobie, że „ma około 30 lat”.

Mnożenie TLR $T = \mathrm{T}(a_1, a_2, a_3, a_4)$ przez skalar $k \in \mathbb{R}$ określa się najczęściej wzorem:

$$kT = \begin{cases} \mathrm{T}(ka_1, ka_2, ka_3, ka_4) & \text{dla } k \geqslant 0, \\ \mathrm{T}(ka_4, ka_3, ka_2, ka_1) & \text{dla } k < 0. \end{cases}$$

Napisz funkcję `mnozskal.trap()`, która jako argumenty przyjmuje listę `T` o długości n składającą się z 4-elementowych wektorów liczbowych oraz m-elementowy wektor liczbowy `k`. Wynikiem działania tej funkcji ma być $\max\{n, m\}$-elementowa lista W, dla której W_i ma wartość `NULL`, jeśli T_i nie określa TLR w sposób poprawny. W przeciwnym przypadku, W_i jest 4-elementowym wektorem liczbowym reprezentującym TLR $k_{((i-1) \bmod m)+1} T_{((i-1) \bmod n)+1}$, czyli wynik mnożenia przez skalar zgodny z regułą zawijania.

Zadanie 6.10. Niech $T = \mathrm{T}(a_1, a_2, a_3, a_4)$ oraz $T' = \mathrm{T}(a'_1, a'_2, a'_3, a'_4)$. Sumę dwóch TLR (por. zad. 6.9) określa się najczęściej wzorem:

$$T + T' = \mathrm{T}(a_1 + a'_1, a_2 + a'_2, a_3 + a'_3, a_4 + a'_4).$$

Napisz funkcję `srednia.trap()`, która jako argumenty przyjmuje listę `T` o długości n składającą się z 4-elementowych wektorów liczbowych reprezentujących TLR. Wynikiem działania tej funkcji ma być 4-elementowy wektor liczbowy reprezentujący TLR będącą średnią arytmetyczną danych TLR, tj. $\frac{1}{n}(T_1 + \cdots + T_n)$.

$\star\star$ **Zadanie 6.11.** Zmodyfikuj algorytm wyboru Bluma, Floyda, Pratta, Rivesta i Tarjana (słynny *selection / median of medians algoritm* z 1973 r. – musisz poszperać w literaturze) w taki sposób, by wyznaczał wartość indeksu h w czasie $O(n)$ (tzn. bez sortowania elementów), gdzie n jest długością wektora wejściowego.

7 Atrybuty obiektów

Zadanie 7.1. Napisz funkcję `taSamaKlasa()`, która dla danej listy sprawdza, czy wszystkie jej elementy mają taką samą wartość atrybutu `class`. Jeśli tak jest, zwróć napis określający nazwę klasy. W przeciwnym przypadku zwróć `FALSE`.

Zadanie 7.2. Napisz funkcję `zakres()`, która dla danego niepustego wektora liczbowego zwraca listę o elementach nazwanych: `x` – kopia wektora, `min` – minimalna wartość z podanego wektora, `max` – obserwacja maksymalna. Nadaj atrybutowi `class` zwracanej listy wartość `zakres`. Następnie utwórz metodę `print.zakres()`, która wypisuje obiekt klasy `zakres` w postaci podobnej do:

```
## x   = 1, 3, 4, 5, 2
## min = 1
## max = 5
```

Zadanie 7.3. Utwórz funkcję `textHist()` o argumentach `x` i `z`, która na podstawie danych zwróconych przez wywołanie `hist(x, plot=FALSE)` wypisze na ekran tekstową, poziomą wersję histogramu. Każdy wiersz powinien zawierać granice przedziałów oraz tyle powtórzeń napisu `z` (domyślnie – gwiazdka, czyli „*”), ile obserwacji wpada do danej klasy. Przykładowo, efektem wywołania `textHist(c(1, 1.2, 1.3, 1.6, 2.1, 2.3, 2.6)), "o")` może być:

```
## ooo 1.0-1.5
## o   1.5-2.0
## oo  2.0-2.5
## o   2.5-3.0
```

Możesz założyć, że maksymalna liczba obserwacji w każdej klasie jest mniejsza niż 30. Postaraj się ustawić granice przedziałów w jednej kolumnie.

Zadanie 7.4. Napisz funkcję o nazwie `combapply()`, która dla danej listy wektorów liczbowych oraz dla danej listy funkcji wywołuje każdą funkcję na każdym wektorze. Zwracaj wynik w postaci listy list, których ewentualne nazwy elementów są kopiowane z list wejściowych. Dla przykładu, w wyniku wywołania `combapply(list(a=1:5, b=2:6), list(f=mean, g=sum, h=prod))` powinniśmy otrzymać obiekt `list(f=list(a=3, b=4), g=list(a=15, b=20), h=list(a=120, b=720))`.

Zadanie 7.5. Twierdzenie Darboux głosi, że dla funkcji $f : \mathbb{R} \to \mathbb{R}$, która jest ciągła na przedziale $[a, b]$ takim, że $f(a) < 0$ i $f(b) > 0$ (lub odwrotnie), istnieje $c \in [a, b]$ takie, że $f(c) = 0$. Napisz funkcję `bisekcja()`, która przyjmuje następujące argumenty:

– jednoargumentową funkcję `f()` zwracającą obiekt typu wektor liczbowy,
– wartość liczbową a (ściślej: wektor liczbowy o długości jeden),

– wartość `b` $>$ `a`,
– wartość dodatnią `eps` (domyślnie: 10^{-16}),
– wartość całkowitą dodatnią `maxiter` (domyślnie: 100).

Prócz podanych warunków poprawności należy sprawdzać, czy spełnione jest założenie twierdzenia Darboux o znaku funkcji w podanych punktach.

Funkcja `bisekcja()` ma znajdować przybliżone położenie miejsca zerowego `f()` za pomocą metody bisekcji, tj. według następującego algorytmu. Dla $i = 1, 2, \dots,$ `maxiter`:

1. Niech $x_i := (a + b)/2$.
2. Jeśli $|f(x_i)| <$ `eps`, to zakończ obliczenia.
3. Jeśli $f(x_i) < 0$, ustal $a := x_i$, a w przeciwnym przypadku ustal $b := x_i$ (dla $f(b) < 0$ i $f(a) > 0$ odwrotnie).

W przypadku braku zbieżności metody w zadanej liczbie iteracji wygeneruj dodatkowo ostrzeżenie za pomocą funkcji `warning()`. W wyniku działania funkcji `bisekcja()` zwracana jest lista zawierająca następujące elementy nazwane (por. `?uniroot`):

1. `root` – przybliżone położenie miejsca zerowego (z ostatnio wykonywanej iteracji algorytmu),
2. `f.root` – wartość funkcji `f()` w powyższym punkcie,
3. `iter` – liczba wykonanych iteracji,
4. `estim.prec` – błąd oszacowania (tutaj: połowa długości przedziału $[a, b]$ z ostatnio wykonywanej iteracji).

Przykładowe wywołanie: `bisekcja(function(x) x^2-1, -0.5, 7.81)`.

Zadanie 7.6. Napisz funkcję `newton.raphson()` implementującą metodę Newtona-Raphsona znajdowania miejsca zerowego różniczkowalnej funkcji $f : \mathbb{R} \to \mathbb{R}$. Funkcja ta powinna przyjmować następujące argumenty wejściowe:

– funkcję `f()` zwracającą obiekt typu wektor liczbowy,
– funkcję `fp()` zwracającą obiekt typu wektor liczbowy (pochodna `f()` – użytkownik musi sam dostarczyć odpowiednią funkcję),
– punkt startowy `x0` (liczba rzeczywista),
– wartość dodatnią `eps` (domyślnie: 10^{-16}),
– wartość całkowitą dodatnią `maxiter` (domyślnie: 100).

Zastosuj następujący algorytm. Niech $x_0 =$ `x0`. Dla $i = 1, 2, \dots,$ `maxiter` wykonuj:

1. Jeśli $f'(x_{i-1}) \simeq 0$, zgłoś błąd (funkcja `stop()`).
2. Niech $x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$.
3. Jeśli $|f(x_i)| <$ `eps`, to zakończ obliczenia.

W przypadku braku zbieżności metody w zadanej liczbie iteracji wygeneruj dodatkowo ostrzeżenie za pomocą funkcji `warning()`. W wyniku działania funkcji `newton.raphson()` zwracana jest lista zawierająca następujące elementy nazwane: `root` – jak w zad. 7.5, `f.root` – jw., `iter` – jw., `estim.prec` – zawsze `NA`. Przykładowe wywołanie: `newton.raphson(function(x) x^2-1, function(x) 2*x, 10)`,

Zadanie 7.7. Napisz funkcję `zloty.podzial()`, która implementuje algorytm złotego podziału wyznaczania minimum lokalnego funkcji ciągłej $f : \mathbb{R} \to \mathbb{R}$ na przedziale $[a, b]$. Funkcja ta powinna przyjmować następujące argumenty wejściowe:

– jednoargumentową funkcję `f()` zwracającą obiekt typu wektor liczbowy,
– wartość liczbową `a`,
– wartość `b` $>$ `a`,
– wartość dodatnią `eps` (domyślnie: 10^{-16}),
– wartość całkowitą dodatnią `maxiter` (domyślnie: 100).

Dla $i = 1, 2, \ldots,$ `maxiter` wykonuj:

1. Niech $x_L := b - \varphi(b-a)$ oraz $x_P := a + \varphi(b-a)$, gdzie $\varphi = \frac{\sqrt{5}-1}{2}$ (złoty podział).
2. Jeśli $f(x_L) > f(x_P)$, to $a := x_L$. W przeciwnym przypadku ustal $b := x_P$.
3. Jeśli $b - a <$ `eps`, to zakończ obliczenia.

W przypadku braku zbieżności metody w zadanej liczbie iteracji wygeneruj ostrzeżenie za pomocą funkcji `warning()`. W wyniku działania funkcji `zloty.podzial()` zwracana jest lista zawierająca następujące elementy nazwane (por. `?optim`):

1. `par` – przybliżone położenie minimum: $(a+b)/2$,
2. `value` – wartość funkcji `f()` w powyższym punkcie,
3. `counts` – liczba wykonanych iteracji,
4. `convergence` – `0` gdy osiągnięto zbieżność, `1` w przeciwnym przypadku,
5. `message` – zawsze `NULL`.

8 Typy złożone

Zadanie 8.1. Napisz funkcję służącą do wyznaczania iloczynu elementów leżących na przekątnej danej macierzy kwadratowej, ale bez użycia `diag()`. Jeśli obiekt przekazany funkcji na wejściu nie jest macierzą kwadratową, należy zgłosić błąd.

Zadanie 8.2. [AO] W poniższej tabeli podane są dane o wadze (w kg) i wzroście (w cm) 10 losowo wybranych mieszkańców Czarnego Lasu. Na podstawie tych danych wyznacz wskaźnik prawidłowej masy ciała (BMI, ang. *body mass index*). Wskaźnik BMI wyznacza się ze wzoru: waga w kilogramach / (wzrost w metrach do kwadratu).

waga [kg]	87	64	62	50	64	83	62	84	66	64
wzrost [cm]	148	162	160	162	170	172	169	162	162	159

Dane o wadze, wzroście i wartości BMI zapisz w postaci ramki danych. Dodaj do ramki danych kolumnę określającą kategorie wagowe: niedowaga (BMI < 18,5), prawidłowa waga (18,5 $\leqslant$ BMI < 25), nadwaga (25 $\leqslant$ BMI < 30), otyłość (BMI $\geqslant$ 30).

Zadanie 8.3. [AO] Ramka danych `nlschools` (pakiet MASS) zawiera dane o 2287 uczniach ze 131 holenderskich szkół tamtejszej ósmej klasy. Wśród nich znajdują się informacje o wynikach testu językowego (kolumna `lang`) oraz statusie społeczno-ekonomicznym (`SES`). Na podstawie zmiennej `SES` utwórz zmienną typu czynnikowego opisującą status społeczno-ekonomiczny w skali 1–5 (przyjmij dowolny sensowny podział na grupy), a następnie utwórz ramkę danych zawierającą dla każdej z tych grup informacje o medianie oraz minimalnej i maksymalnej wartości liczby punktów uzyskanych z testu językowego.

Zadanie 8.4. [AO] W ramce danych `waders` (pakiet MASS) podane są zaobserwowane latem liczności 19 gatunków ptaków brodzących występujących w różnych miejscach południowej Afryki. Utwórz ramkę danych, której kolumny uporządkowane są w kolejności od sumarycznie najczęściej do najrzadziej występujących osobników.

Zadanie 8.5. [AO] Ramka danych `wine` z pakietu gamair (wywołaj `data(wine)`, by uzyskać do niej dostęp), zawiera informacje o cenach i cechach win Bordeaux roczników 1952–1998.

(a) Wyznacz średnią temperaturę w porze letniej i porze zbiorów oraz średnią wysokość opadów w porze zimowej i porze zbiorów w poszczególnych latach.
(b) Utwórz ramkę danych zawierającą tylko wyniki dziesięciu najgorętszych roczników (temperatura w lecie). Jej obserwacje powinny przyjąć nazwy (`row.names`) odpowiadające rocznikom win.
(c) Wyznacz wartość współczynnika korelacji Kendalla między ceną a oceną roczników win. Zbadaj także korelacje między ceną a innymi zmiennymi.

Zadanie 8.6. [AO] Celem pewnego szwedzkiego eksperymentu było zbadanie liczby wypadków samochodowych w zależności od obowiązywania bądź nie ograniczenia prędkości na autostradach (zmienna `limit`). W wybranych 92 dniach roku zmierzono liczbę wypadków (zmienna `y`). Eksperyment powtórzono w odpowiadających dniach kolejnego roku. Wyniki zawarte są w ramce danych `Traffic` z pakietu MASS. Utwórz tablice kontyngencji w postaci macierzy o dwóch wierszach i dwóch kolumnach informujące o:

(a) liczbie dni w jednym i drugim roku podczas których obowiązywało ograniczenie prędkości,
(b) łącznej liczbie wypadków,
(c) średniej liczbie wypadków na dzień.

Zadanie 8.7. Dana jest macierz $P \geqslant 0$ rozmiaru $n \times m$ taka, że $\sum_{i=1}^{n}\sum_{j=1}^{m} p_{i,j} = 1$ oraz posortowane rosnąco wektory liczbowe $\mathtt{x}$ (n-elementowy) i $\mathtt{y}$ (m-elementowy). Trójka $(\mathtt{x}, \mathtt{y}, P)$ opisuje rozkład prawdopodobieństwa pewnej dwuwymiarowej zmiennej losowej dyskretnej (X, Y), tak jak w poniższym podzadaniu.

W pewnej szkole rozkład prawdopodobieństwa uzyskania ocen z Filozofii Bytu i Wychowania Fizycznego przez tego samego studenta przedstawia się następująco.

		WF 2	3	4	5
FB	2	*0*	*0,01*	*0,1*	*0,2*
	3	*0,01*	*0,05*	*0,03*	*0,1*
	4	*0,1*	*0,03*	*0,05*	*0,01*
	5	*0,2*	*0,1*	*0,01*	*0*

a) Zmienne losowe X i Y są niezależne wtedy i tylko wtedy, gdy dla każdego i, j zachodzi $p_{i,j} = (\sum_{k=1}^{n} p_{k,j})(\sum_{l=1}^{m} p_{i,l})$. Napisz funkcję `niezaleznosc()`, która sprawdza, czy zachodzi ta własność dla danych $(\mathtt{x}, \mathtt{y}, P)$ (zwracamy wartość logiczną).
b) Ponadto napisz funkcję, która dla $(\mathtt{x}, \mathtt{y}, P)$ zwróci wektor liczbowy (z ustawionym atrybutem `names` – dowolne, lecz czytelne dla użytkownika etykiety) zawierający wartości podstawowych charakterystyk (X, Y):

- Wartości oczekiwane: $\mathbb{E}\,X = \sum_{i=1}^n x_i \sum_{j=1}^m p_{i,j}$, $\mathbb{E}\,Y = \sum_{j=1}^m y_j \sum_{i=1}^n p_{i,j}$,
- Wariancje: $\mathrm{Var}\,X = \mathbb{E}\,X^2 - (\mathbb{E}\,X)^2$, gdzie $\mathbb{E}\,X^2 = \sum_{i=1}^n x_i^2 \sum_{j=1}^m p_{i,j}$ oraz $\mathrm{Var}\,Y = \mathbb{E}\,Y^2 - (\mathbb{E}\,Y)^2$, gdzie $\mathbb{E}\,X^2 = \sum_{j=1}^m y_j^2 \sum_{i=1}^n p_{i,j}$,
- Kowariancję: $\mathrm{Cov}(X,Y) = \mathbb{E}\,(XY) - \mathbb{E}\,X\,\mathbb{E}\,Y$ dla $\mathbb{E}\,(XY) = \sum_{i=1}^n \sum_{j=1}^m x_i y_j p_{i,j}$,
- Współczynnik korelacji: $\varrho(X,Y) = \mathrm{Cov}(X,Y)/\sqrt{\mathrm{Var}\,X\,\mathrm{Var}\,Y}$.

Zadanie 8.8. Napisz własną implementację funkcji służącej do „algebraicznego" mnożenia macierzy liczbowych działającą tak samo, jak operator „%*%". Dla macierzy `A` rozmiaru $n \times p$ i macierzy `B` rozmiaru $p \times m$ zwracana będzie macierz C rozmiaru $n \times m$ taka, że $c_{i,j} = \sum_{k=1}^p a_{i,k}\, b_{k,j}$ dla $i = 1,\dots,n$ i $j = 1,\dots,m$. Jeśli macierze podane na wejściu nie spełniają powyższych założeń, należy zgłosić błąd. Porównaj wydajność swojej funkcji i operatora `%*%` za pomocą funkcji `microbenchmark2()` dla macierzy o różnych rozmiarach.

Zadanie 8.9. Napisz własną implementację funkcji służącej do agregacji danych pewnego wektora liczbowego `x` w podgrupach wyznaczonych przez czynnik `g` za pomocą danej funkcji `f()` (por. `aggregate()` i `by()`, `x` i `g` muszą być takiej samej długości). Wynikiem jej działania powinien być wektor liczbowy z ustawionym atrybutem `names` (odpowiadającym nazwom poziomów czynnika).

Wbudowana przykładowa ramka danych `ToothGrowth` (wpisz po prostu jej nazwę, aby ją wyświetlić) zawiera wyniki badań przeprowadzonych na świnkach morskich. Mierzono długość zębów w grupach gryzoni przyjmujących różne dawki witaminy C [mg] w jednej z dwóch postaci: soku pomarańczowego (`OJ`) i bezpośrednio, czyli kwasu askorbinowego (`VC`). Celem sprawdzenia działania swojej funkcji wyznacz średnią długość zębów oddzielnie dla (a) każdej dawki, (b) postaci preparatu witaminowego oraz (c) dawki i postaci preparatu razem.

Zadanie 8.10. Napisz funkcję, która dla danego szeregu czasowego `x` o n elementach oraz nieparzystej liczby naturalnej `k` wyznaczy `k`-średnią ruchomą, $k < n$, tj. szereg czasowy $(w_1,\dots,w_{n-k+1})$, dla którego $w_i = \sum_{j=1}^k x_{i+j-1}/k$. Jednostki czasu dla wynikowego szeregu dobierz wedle uznania.

Wbudowany szereg czasowy `UKgas` zawiera dane na temat kwartalnej konsumpcji gazu w Zjednoczonym Królestwie. Użyj go do przetestowania zaimplementowanego algorytmu. Szereg możesz narysować, wywołując `plot(UKgas)`.

Zadanie 8.11. [BT] Napisz własną implementację podstawowej funkcjonalności oferowanej przez funkcję `diag()` w postaci jednoargumentowej funkcji `mydiag()`. Jeśli argumentem jest wektor o długości większej niż 1, funkcja zwraca diagonalną macierz kwadratową z danymi elementami na przekątnej (uwaga: wektor nie musi być koniecznie liczbowy). Jeśli argumentem jest liczba naturalna, funkcja zwraca macierz identycznościową podanego rozmiaru. Jeśli zaś argumentem jest macierz, należy zwrócić elementy występujące na przekątnej (macierz nie musi koniecznie być kwadratowa). W przeciwnym przypadku należy zgłosić błąd.

Zadanie 8.12. Graf skończony jest to para $G = (V, E)$, gdzie $V = \{v_1,\dots,v_n\}$ (zbiór wierzchołków) oraz $E \subseteq V \times V$ (zbiór krawędzi, czyli relacja binarna określona na V). Każdy graf możemy reprezentować w postaci zerojedynkowej macierzy kwadratowej K, gdzie $k_{i,j} = 1$ oznacza, że wierzchołek v_i jest połączony z wierzchołkiem v_j oraz $k_{i,j} = 0$ gdy $(v_i, v_j) \notin E$, $i,j \in \{1,\dots,n\}$.

Graf skończony nazwiemy *prostym*, jeśli nie ma on pętli, tj. $(v_i, v_i) \notin E$ oraz gdy jest nieskierowany, tzn. $(v_i, v_j) \in E \Rightarrow (v_j, v_i) \in E$ dla każdego $i, j \in \{1, \ldots, n\}$.

Napisz funkcję `prosty()`, która dla danej kwadratowej macierzy zerojedynkowej (jeśli argument nie jest takim obiektem, zwróć błąd) sprawdzi, czy reprezentuje ona graf prosty.

$\star$ **Zadanie 8.13.** [BT] Napisz funkcję, która dla danej zerojedynkowej macierzy kwadratowej reprezentującej graf prosty $G = (V, E)$ o n wierzchołkach, por. zad. 8.12, oraz wartości całkowitych $i, j \in \{1, \ldots, n\}$, zwróci wektor całkowitoliczbowy $(d_1, \ldots, d_k)$ reprezentujący drogę między wierzchołkami v_i i v_j, tj. $d_1 = i$, $d_k = j$, $(v_{d_l}, v_{d_{l+1}}) \in E$ dla $l = 1, \ldots, k-1$ bądź `NA`, jeśli droga nie istnieje.

Zadanie 8.14. [BT & Project Euler] Dana jest macierz trójkątna dolna A rozmiaru $n \times n$ o elementach dodatnich (możemy założyć, że elementy nad przekątną są np. równe zero – nie będą nas interesować). Każdy element $a_{i,j}$, $i \geqslant j$, reprezentuje pewną liczbę cukierków, które może zagarnąć Jaś do swojego puzderka. Zbieranie rozpoczyna on od $a_{1,1}$. Z każdego elementu $a_{i,j}$ i może iść dalej albo w dół (do $a_{i+1,j}$) albo w dół-prawo (do $a_{i+1,j+1}$, o ile to możliwe). Naszym celem jest napisanie funkcji, która ustali, ile najwięcej cukierków może zdobyć Jaś postępując zgodnie z regułami tej okrutnej gry.

Napisz funkcję `jas.zachlanny()`, która implementuje „zachłanny” (w ogólności nieoptymalny) sposób wyboru kolejnych kroków. Decyzję o wyborze drogi podejmujemy lokalnie na podstawie tego, gdzie dostaniemy większą ich liczbę. Na przykład dla macierzy podanej obok wynikiem powinno być 21, ponieważ Jaś wybrałby tutaj drogę $4 + 7 + 4 + 6$.

```
4 . . .
7 5 . .
3 4 6 .
9 6 2 3
```

Zadanie 8.15. [BT] Napisz funkcję `jas.dynamiczny()` która tym razem rozwiązuje problem zbierania cukierków (por. zad. 8.14) za pomocą sposobu uzyskanego przez metodę tzw. programowania dynamicznego (rezultat będzie tym razem optymalny).

Rozwiązanie wyznaczane jest z wykorzystaniem pomocniczej macierzy B (trójkątnej dolnej rozmiaru $n \times n$), w której $b_{i,j} > 0$ reprezentuje informację o maksymalnej liczbie cukierków, która może być zebrana przez dojście optymalną drogą w dół poczynając od $a_{i,j}$. Zachodzi $b_{n,j} = a_{n,j}$ oraz $b_{i,j} = a_{i,j} + \max\{b_{i+1,j}, b_{i+1,j+1}\}$, $i < n$.

Zadanie 8.16. [AO] Napisz funkcję `rozwin()`, która przekształca daną macierz rozmiaru $n \times m$ (niekoniecznie liczbową) z ustawionym atrybutem `dimnames` na ramkę danych zawierającą nm obserwacji i trzy kolumny o nazwach zadanych za pomocą odpowiedniego argumentu funkcji. Wartości z macierzy mają znajdować się w pierwszej kolumnie, a w kolejnych dwóch – kombinacje nazw wierszy i kolumn odpowiadające podanym poziom czynnika.

Dla przykładu, obiekt `WorldPhones` (wbudowany) zawiera dane o liczbie telefonów (w tysiącach) w różnych regionach świata w wybranych latach. Wynikiem wywołania `rozwin(WorldPhones, c("ile", "gdzie", "kiedy"))` może być:

```
     ile    gdzie kiedy
...
2   60423   N.Amer   1956
3   64721   N.Amer   1957
...
9   29990   Europe   1956
10  32510   Europe   1957
...
```

Zadanie 8.17. Napisz funkcję odwrotną do funkcji z zad. 8.16. Dana jest ramka danych zawierająca nm wierszy oraz 3 kolumny (pierwsza – dowolnego typu, druga i trzecia – typu czynnikowego, odpowiednio o n i m poziomach). Obserwacje zawierają wszystkie możliwe kombinacje poziomów dwóch czynników, ale nie możemy założyć, że są one koniecznie ułożone w jakimś określonym porządku (funkcja ma działać dla dowolnej permutacji obserwacji). Wynikiem ma być macierz rozmiaru $n \times m$ o elementach pochodzących z pierwszej kolumny ramki danych. Atrybut `dimnames` ustawiamy na podstawie wartości poziomów pierwszego i drugiego czynnika.

9 Napisy

Zadanie 9.1. [AO] Załóżmy, że mamy dany wektor napisów zawierający informacje o datach urodzenia losowo wybranych osób. Dane te zapisane są w formacie *rrrr-mm-dd*. Napisz funkcję, która zwróci ramkę danych o czterech kolumnach: oddzielnie rok, miesiąc i dzień oraz informacja, czy dana osoba jest dziś pełnoletnia, czy nie (na podstawie aktualnej daty systemowej). Dla niepoprawnych dat (np. *2011-02-29*) wstawiaj wartości `NA`.

Zadanie 9.2. Napisz funkcję `wyluskajLiczby()`, która z danego wektora napisów „wyłuska” wszystkie liczby, np. 12.1, -14, 0.000001. Uwaga: w jednym napisie może występować tekst i różne liczby – należy wydobyć wszystkie z nich. Wynik przedstaw w postaci listy, której i-ty element to wektor liczbowy składający się z wartości odczytanych z i-tego napisu. Jeśli wynikowa lista składa się z wektorów tej samej długości, przekształć ją w ramkę danych o i wierszach.

Zadanie 9.3. [BT] Najprostszy (i jednocześnie niezapewniający bezpieczeństwa) ze znanych sposobów szyfrowania tekstu to tzw. szyfr Cezara z przesunięciem `h`. Przekształca on każdą i-tą literę alfabetu łacińskiego na $(i + \mathtt{h})$-tą (z ewentualnym „zawijaniem”). Napisz funkcję `cezar()`, która dla danego `h` zaszyfruje każdy napis z danego wektora napisów.

```
cezar(c("abcABCąśćxyzXYZ0123!@$", "Ala ma Ferrari."), 1)
## [1] "bcdBCDąśćyzaYZA0123!@$" "Bmb nb Gfssbsj."
cezar("F bodfokrę aw gwę dcrcpo!", -14) odszyfruj
## [1] "R naprawdę mi się podoba!"
```

Zadanie 9.4. Dany jest wektor napisów, w którym każdy napis reprezentuje akapit tekstu. Napisz funkcję `wrap_greedy()`, która implementuje zachłanny algorytm podziału akapitów na wiersze (ang. *word wrap*) składające się z maksymalnie `h` (argument wejściowy, domyślnie 76) znaków.

Z każdego akapitu należy wydobyć wszystkie „słowa”, tj. najdłuższe spójne ciągi znaków drukowanych niebędących białymi znakami. Zakładamy, że każde „słowo” jest nie dłuższe niż `h`. Wszystkie podciągi składające się z białych znaków zostaną zastąpione pojedynczymi spacjami (oddzielanie „słów”) bądź znakami nowej linii, `\n` (oddzielanie „wierszy”). W algorytmie zachłannym należy po prostu wypisywać do napisu wynikowego kolejne słowa z napisu wejściowego, wstawiając między nimi znaki nowej linii wtedy, gdy stwierdzimy, że dane słowo nie „zmieściłoby” się w tworzonym wierszu.

```
tekst <- c("Marcin Karolina Adam Bartłomiej Karolina Kamil Dawid Anna
Paulina Joanna Jakub Monika Paulina Piotr Marek Artur Katarzyna Alicja
Michał Sylwia Paweł Hong Aleksandra Adam Monika Michał Bartłomiej
```

```
Paweł Joanna Joanna Artur",
"Tatoooooooo nieeeeeee wraaaaaaaca; ranki i wieeeeeeeczory        \n
We łzaaaaaach goooooooo czeeeeeeeeeeeeeekam")
h <- 55
cat(wrap_greedy(tekst, h), sep=str_join("\n|", str_dup("-",h-2), "|\n"))
```

```
## Marcin Karolina Adam Bartłomiej Karolina Kamil Dawid
## Anna Paulina Joanna Jakub Monika Paulina Piotr Marek
## Artur Katarzyna Alicja Michał Sylwia Paweł Hong
## Aleksandra Adam Monika Michał Bartłomiej Paweł Joanna
## Joanna Artur
## |-----------------------------------------------------|
## Tatoooooooo nieeeeeee wraaaaaaaca; ranki i
## wieeeeeeeczory We łzaaaaaach goooooooo
## czeeeeeeeeeeeeeekam
```

⋆ **Zadanie 9.5.** Zaimplementuj dynamiczny algorytm zawijania wierszy (por. zad. 9.4) w postaci funkcji `wrap_dynamic()`, minimalizujący sumę „kosztów” wypisania każdego wiersza, gdzie koszt określony jest jako kwadrat liczby pustych kolumn po ostatnim słowie w danym wierszu (sięgnij do dostępnej literatury w poszukiwaniu wskazówek). Można zaobserwować, że taka metoda powoduje bardziej równomierne rozłożenie się słów we wszystkich wierszach. Spróbuj też zaimplementować wersję, w której koszt ostatniego wiersza się nie liczy (można tam pozostawić więcej pustych kolumn). Możesz też wstawiać więcej niż jedną spację między słowami, by poprawić estetykę wyniku.

```
cat(wrap_dynamic(tekst, h), sep=str_join("\n|", str_dup("-",h-2), "|\n"))
## Marcin Karolina Adam Bartłomiej Karolina
## Kamil Dawid Anna Paulina Joanna Jakub Monika
## Paulina Piotr Marek Artur Katarzyna Alicja
## Michał Sylwia Paweł Hong Aleksandra Adam Monika
## Michał Bartłomiej Paweł Joanna Joanna Artur
## |-----------------------------------------------------|
## Tatoooooooo nieeeeeee wraaaaaaaca;
## ranki i wieeeeeeeczory We łzaaaaaach
## goooooooo czeeeeeeeeeeeeeekam
```

Zadanie 9.6. Napisz funkcję, która usunie z danego wektora napisów wszystkie niecenzuralne wyrazy (ich listę musisz sam określić, każdy napis może składać się z wielu wyrazów). Wszystkie „wewnętrzne” litery zastąp gwiazdkami, np. „wziąść” → „w****ć”, „poszłem” → „p*****m”,

Zadanie 9.7. Napisz funkcję `zlicz()`, która dla danego napisu zliczy liczbę wystąpień każdej litery z alfabetu łacińskiego (i tylko takiej) i zwróci wynik w postaci wektora liczb całkowitych z ustawionym atrybutem `names` (jaka litera). Wynikowy wektor powinien być posortowany nierosnąco względem częstości występowania każdej z liter.

Zadanie 9.8. [AO] Wartości wektora napisów opisują 10-cyfrowe kody terytorialne gmin w pewnym województwie. Wiedząc, że pierwsze 4 cyfry tego kodu oznaczają kod powiatu, do którego należy dana gmina, stwórz wykaz gmin przynależnych do każdego poszczególnego powiatu w tym województwie. Rozwiązanie napisz w postaci funkcji `kody()`. Zwracaj listę napisów.

Zadanie 9.9. Dane są trzy wektory o równych długościach, w których i-te elementy opisują, odpowiednio, godzinę, minutę i sekundę zajścia pewnego zdarzenia. Napisz funkcję, która zwróci $(n-1)$-elementowy wektor informujący o liczbie sekund, które upłynęły między *kolejnymi* (w czasie; wektory trzeba odpowiednio posortować) zdarzeniami.

Zadanie 9.10. Palindrom to napis, który zapisany wprost i wspak jest identyczny. Napisz funkcję `palindrom()`, która zwraca tylko te napisy z danego wektora napisów, które są palindromami. Np. `palindrom(c("Łał", "bala", "Madam, I'm Adam")) == c("Łał", "Madam, I'm Adam")`. *Uwaga. Przy sprawdzaniu należy pomijać wszystkie znaki niebędące literami i ignorować wielkość liter.*

Zadanie 9.11. Napisz, nie korzystając z wyrażeń regularnych ani z funkcji `match()` bądź `match.arg()`, własną implementację funkcji `pmatch()`.

⋆ **Zadanie 9.12.** Napisz prosty kalkulator dla wyrażeń w tzw. odwrotnej notacji polskiej (beznawiasowej). Funkcja `onp()` przyjmuje jako argument wektor napisów i zwraca wektor liczbowy tej samej długości, w którym znajdują się wyniki obliczeń. W napisach wejściowych dopuszczalne są tylko operatory (binarne): `+`, `-`, `*`, `-` oraz liczby (naturalne). Każdy „składnik” oddzielony jest spacją. Np. `"2 3 +"` powinno dać w wyniku 5, `"3 2 - 1 + 7 *"` to $((3-2)+1)*7 = 14$, `"2 7 + 3 / 14 3 - 4 * + 2 /"` to $((2+7)/3+(14-3)*4)/2 = 23{,}5$.

10 Przetwarzanie plików

Uwaga. Przez „napisz funkcję, która dla danego pliku zrobi coś” mamy poniżej na myśli „dla danej ścieżki dostępu do pliku”.

Zadanie 10.1. Strona internetowa *cran.rstudio.com/src/base/R-2/* udostępnia przykładowy listing plików standardowo generowany przez serwer Apache (w formie dokumentu HTML). Napisz funkcję `getApacheDirListing()`, która dla danego adresu URL (I argument) zwraca ramkę danych zawierającą informacje udostępnione w listingu. Wynikowy obiekt powinien składać się z czterech kolumn: `url` – URL pliku (napis), `name` – nazwa pliku (napis), `modtime` – czas ostatniej modyfikacji pliku (`POSIXct`) oraz `size` – przybliżony rozmiar w bajtach (liczba rzeczywista, założenie: 1 KB to 1000 B).

Zadanie 10.2. Napisz funkcję `mergeAll()`, która ze wszystkich plików o rozszerzeniu `ext` (domyślnie `.txt`, III argument) ze wskazanego katalogu `dir` (I argument) utworzy jeden plik o nazwie `outfname` (II argument) będący ich złączeniem.

Zadanie 10.3. Napisz funkcję `zdarzenia()`, która dla danego pliku (I argument, `fname`) zawierającego czas nastąpienia pewnych zdarzeń i ich rodzaj (rodzaje zdarzeń mogą się powtarzać), w formacie `rrrr-mm-dd<spacja>gg:mm:ss<spacja>opis zdarzenia`:
1. sprawdzi, czy kolejne czasy są posortowane rosnąco (jeśli nie – zgłaszany jest błąd);
2. wyłuska wszystkie rodzaje zdarzeń i zliczy, ile ich jest;
3. wyznaczy minimalny, średni i maksymalny czas między zdarzeniami tego samego rodzaju.

Wyniki należy zachować w ramce danych (kolumny: `zdarzenie`, `liczba`, `dt.min`, `dt.sr`, `dt.max`). Przykładowa zawartość pliku:

```
2012-12-01 00:00:01 hamburger
2012-12-01 00:01:12 hamburger
2012-12-01 00:04:21 frytki
2012-12-01 00:04:22 cola
```

Zadanie 10.4. Napisz funkcję `removeHtmlTags()`, która utworzy plik o nazwie `outfname` (II argument) będący tekstową reprezentacją pliku HTML o nazwie `infname` (I argument). Interesuje nas zawartość zamieszczona między `<body>` a `</body>` i w której wszystkie tagi HTML są pominięte. Na przykład dla pliku o zawartości:

```
<html><head><title>Moja strona</title></head>
<body><h1>Moja pierwsza strona</h1>
<p>Za górami, <em>za lasami</em> żył sobie mały leśny ludek....</p>
</body></html>
```

powinniśmy uzyskać:

```
Moja pierwsza strona
Za górami, za lasami żył sobie mały leśny ludek...
```

Zadanie 10.5. Napisz funkcję `extractUrls()`, która z danego pliku tekstowego `infname` (I argument) wyłuska wszystkie adresy internetowe postaci `http://*` lub `www.*`. Rezultaty przedstaw jako wektor napisów.

Zadanie 10.6. Napisz funkcję `massDownload()`, która dla danego wektora napisów `urls` (I argument) zawierającego n adresów URL różnych stron internetowych pobierze i zapisze je w oddzielnych plikach `1.html`, ..., n`.html` we wskazanym katalogu `outdir` (II argument).

Zadanie 10.7. Napisz funkcję `words()`, która dla danego pliku tekstowego (I argument, `fname`) zwróci ramkę danych o dwóch kolumnach zawierającą wszystkie występujące słowa (`word`) oraz ich liczby wystąpień (`count`). Ramka danych powinna być posortowana nierosnąco względem liczby wystąpień słów.

Zadanie 10.8. Napisz funkcję `extractRcodeSweave()`, która dla danego pliku `.Rnw` (I argument, `infname`) zawierającego kod dokumentu knitr dla LaTeX-a utworzy plik `.R` (II argument, `outfname`) zawierający wszystkie polecenia R-a znajdujące się we wszystkich wstawkach (*chunks*).

Zadanie 10.9. Napisz funkcję `diskusage()`, która narysuje wykres kołowy reprezentujący sumę rozmiarów wszystkich plików znajdujących się w danym katalogu `directory` (I argument) i jego podkatalogach. Każdy podkatalog przeskanuj rekurencyjnie.

Dla przykładu, skanowanie katalogu `katalog`, zawierającego następujące elementy:

```
katalog/plik1.txt 10 MB
katalog/plik2.txt 2 MB
katalog/katA/plik3.R 5 MB
katalog/katB/katB1/plik4.html 20 MB
katalog/katB/katB2/plik5.html 30 MB
```

powinno narysować w wyniku wykres kołowy zawierający trzy części: „`.`” (12 MB), „`katA`” (5 MB), „`katB`” (50 MB).

⋆ **Zadanie 10.10.** Napisz funkcję `CSVToCSV2()`, która zamieni separatory pól „`,`” na „`;`” oraz separatory części ułamkowej liczb „`.`” na „`,`” bez wywoływania `read.table()`, `write.table()` i ich pochodnych. Wejście: plik `infname` (I argument), wyjście: `outfname` (II argument). Uważaj, na to, by nie zmieniać zawartości napisów (zawartych w cudzysłowie).

11 Niskopoziomowe operacje graficzne

Zadanie 11.1. Wysokopoziomowa funkcja `pie()` służy do rysowania wykresów kołowych dla danych jakościowych. Przykład:

```
dzieci <- c("krasnale"=48, "zuchy"=69, "wesolki"=32)
pie(dzieci)
```

Napisz swoją własną implementację `pie()` w postaci funkcji `mypie()`, wzorując się na jej R-owej implementacji. Przyjmij, że dopuszczalnymi danymi wejściowymi będą wektory liczbowe z ustawionym atrybutem `names` (liczby obserwacji przypadające na daną klasę) oraz zmienne typu `factor`.

Zadanie 11.2. Utwórz funkcję `myboxplot()`, która zawiera Twoją własną implementację funkcji rysującej wykres skrzynkowy dla danych ilościowych, por. `?boxplot`. Pamiętaj o poprawnym wyróżnianiu obserwacji odstających (*outliers*).

Zadanie 11.3. Narysuj, korzystając tylko z funkcji niskopoziomowych, wykres funkcji sinus i cosinus na przedziale $[0, 2\pi]$. Układ współrzędnych narysuj (ręcznie) tak, by na osi OX oznaczone były wartości $0, \frac{\pi}{2}, \pi, \frac{3}{2}\pi, 2\pi$ (skorzystaj z symboli opisanych na stronie podręcznika `?plotmath`), a na OY tylko $-1, 0, 1$. Dodaj odpowiednią legendę podobną do tej, którą wygenerowałaby funkcja `legend()` (lub ładniejszą).

Zadanie 11.4. Napisz funkcję `myaxis()` (por. `?axis`), która narysuje układ współrzędnych (osie i ich etykiety) na Twój własny, ulubiony sposób. Skorzystaj m.in. z wartości parametru graficznego `usr`.

Zadanie 11.5. [AO] Napisz funkcję `mymultiboxplot()`, która dla danego wektora liczbowego `x` i czynnika `g` o `k` poziomach (wektor i czynnik są równoliczne) narysuje jeden wykres składający się z ustawionych obok siebie `k` „skrzynek” (por. zad. 11.2) dla wartości z wektora `x` odpowiadających kolejnym poziomom czynnika `g`. Możesz uzyskać efekt podobny do poniższego wywołania funkcji wysokopoziomoej `boxplot()`.

```
x <- rnorm(30)
g <- factor(rep(1:3, each=10))
boxplot(split(x, g)) # u nas: mymultiboxplot(x, g)
```

Dodatkowo zaznacz na wykresach przedziały $[\bar{x}_i - cs_i, \bar{x}_i + cs_i]$, gdzie $\bar{x}_i$ i s_i oznaczają, odpowiednio, średnią i odchylenie standardowe wartości z wektora `x` dla i-tego poziomu czynnika. Dla parametru $c \in \mathbb{N}$ przyjmij domyślnie wartość 2. Co więcej, wartości średnie połącz za pomocą łamanej.

Zadanie 11.6. Zaimplementuj funkcję `rectplot()`, służącą do rysowania tego, co tutaj roboczo nazwiemy wykresem prostokątnym dla danych jakościowych (por. zad. 11.1). Prostokąt wypełniający cały obszar rysowania podziel dowolnie na podprostokąty o polach proporcjonalnych do liczby obserwacji w każdej z klas. Każdy podobszar powinien być wypełniony innym kolorem. Ponadto wewnątrz niego powinna znajdować się etykieta poziomu czynnika i informacja o procentowym udziale obserwacji z tej klasy w zbiorze wejściowym.

Zadanie 11.7. Wysokopoziomowa funkcja `barplot()` służy do rysowania wykresów słupkowych dla danych jakościowych. Przykład:

```
dzieci <- c("krasnale"=48, "zuchy"=69, "wesolki"=32)
barplot(dzieci)
```


Napisz swoją własną implementację `barplot()` w postaci funkcji `mybarplot()`.

Zadanie 11.8. W rozdz. 3 analizowaliśmy, jakiego typu obiekt zwraca funkcja `hist()`, która służy do wyznaczania i, opcjonalnie, rysowania histogramu:

```
x <- c(...) # Twoje dane
h <- hist(x, plot = FALSE) # nie rysuj, tylko wyznacz histogram
```

Napisz funkcję `myhist1()`, która dla danego jako argument wektora liczbowego `x` wyznaczy histogram jak wyżej (obiekt klasy `histogram`, który jest listą) i narysuje go na Twój ulubiony, samodzielnie opracowany sposób.

Zadanie 11.9. Utwórz funkcję `myhist2()`, która zawiera Twoją własną implementację najważniejszych działań wykonywanych przez `hist()`, włącznie z wyznaczaniem histogramu, zwracaniem go w postaci listy klasy `histogram` i ewentualnym rysowaniem. Możesz skorzystać z części kodu napisanego w ramach rozwiązania zadania 11.8. Tym razem jakiekolwiek wywoływanie `hist()` jest niedopuszczalne.

Zadanie 11.10. Napisz funkcję `Fn()`, która dla danego wektora liczbowego $\mathtt{x} = (x_1, \ldots, x_n)$ o unikalnych wartościach narysuje za pomocą funkcji niskopoziomowych (bez użycia wbudowanych funkcji `ecdf()` oraz `approxfun()`) jego dystrybuantę empiryczną daną wzorem

$$\hat{F}_n(y) = \begin{cases} 0 & \text{dla } y < x_{(1)}, \\ i/n & \text{dla } i < n \text{ takiego, że } x_{(i)} \leqslant y < x_{(i+1)}, \\ 1 & \text{dla } y \geqslant x_{(n)}, \end{cases}$$

gdzie $x_{(i)}$ oznacza i-tą najmniejszą wartość z `x`.

Zadanie 11.11. [BT] Napisz funkcję rysującą graf prosty na podstawie podanej zerojedynkowej, symetrycznej macierzy sąsiedztwa. Wierzchołki (sposób rysowania dowolny) mają być rozlokowane równomiernie na okręgu. Krawędzie reprezentowane są przez odcinki łączące wierzchołki.

Zadanie 11.12. [BT] Za pomocą poniższych poleceń możemy poznać aktualny czas systemowy:

```
h <- as.integer(format(Sys.time(), "%H")) # godziny
m <- as.integer(format(Sys.time(), "%M")) # minuty
s <- as.integer(format(Sys.time(), "%S")) # sekundy
print(c(h, m, s))
## [1] 14 43  3
```

Utwórz funkcję `zegarek()`, która narysuje tarczę zegarka (w formie np. okręgu) z oznaczonymi godzinami (minimum to 3, 6, 9 i 12) oraz wskazówki (godzinową, minutową i sekundową) ustawione na odczytanym powyżej czasie.

Zadanie 11.13. Napisz funkcję `heart()`, która za pomocą funkcji niskopoziomowych narysuje przybliżony kształt krzywej zamkniętej danej równaniem parametrycznym:

$$\begin{cases} x(t) & = & 16 \sin^3 t, \\ y(t) & = & 13 \cos t - 5 \cos 2t - 2 \cos 3t - \cos 4t, \end{cases}$$

dla $t \in [0, 2\pi)$. Ponadto wypełnij jej wnętrze danym kolorem (domyślnie czerwonym).

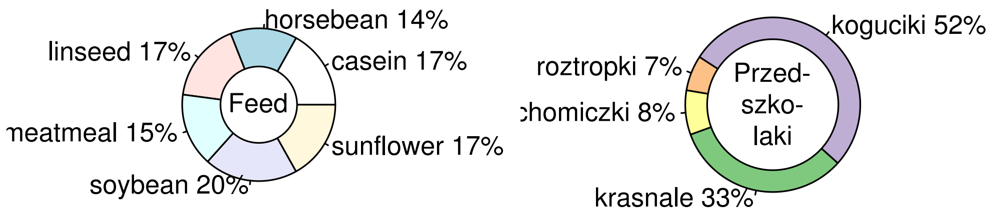
12 Wysokopoziomowe operacje graficzne

Uwaga 1. Tym razem celem ćwiczeń jest dostosowywanie działania funkcji wysokopoziomowych za pomocą funkcji niskopoziomowych. Do każdego zadania podajemy ilustracje, które dla przykładowych danych powinny być ***wiernie*** *odtworzone.*

Uwaga 2. Wszystkie z implementowanych funkcji powinny udostępniać możliwość przekazania dodatkowych parametrów graficznych bezpośrednio wywoływanej funkcji wysokopoziomowej za pomocą argumentu specjalnego „...”.

Zadanie 12.1. Napisz funkcję `nicepie()`, która tworzy poniższy wykres kołowy za pomocą wywołania funkcji `pie()` i funkcji niskopoziomowych. Funkcja powinna przyjmować następujące argumenty: `x` (zbiór danych), `main` (tytuł wykresu), `r1` (promień „dużego” koła, domyślnie `0.7`), `r2` (promień wypełnionego kolorem białym „małego” koła o czarnym brzegu, domyślnie `r1/2`).

```
nicepie(table(chickwts$feed), 'Feed') # wbudowana ramka danych chickwts
library('RColorBrewer')
dane <- c(krasnale=20, koguciki=32, roztropki=4, chomiczki=5)
nicepie(dane, 'Przed-\nszko-\nlaki', 0.8, 0.6,
     init.angle=200, col=brewer.pal(4, 'Accent'))
```

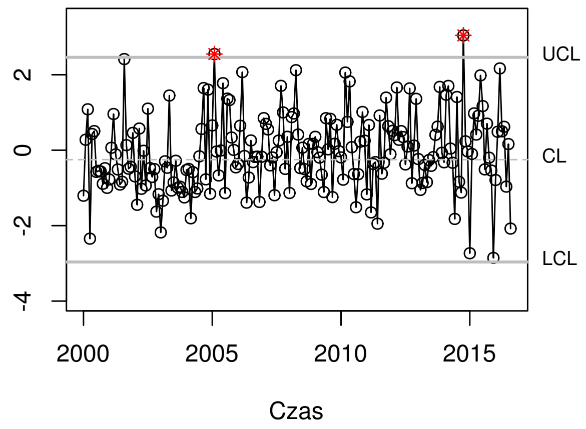


Zadanie 12.2. [AO] Napisz funkcję `ctrlchart()`, która za pomocą metody `plot.ts()` narysuje przebieg wartości danego szeregu czasowego y składającego się z n obserwacji.

Dodatkowo zaznacz na wykresie trzy poziome linie: $\mathrm{CL} = \bar{y}$, $\mathrm{UCL} = \bar{y} + 3\frac{\mathrm{MR}}{1{,}128}$ i $\mathrm{LCL} = \bar{y} - 3\frac{\mathrm{MR}}{1{,}128}$ (spraw, by były zawsze widoczne), gdzie $\bar{y}$ jest średnią arytmetyczną z pierwszych m (argument wejściowy, domyślnie równy $\lfloor n/10 \rfloor$) obserwacji szeregu, zaś MR jest średnim ruchomym rozstępem z m pierwszych obserwacji szeregu wyznaczonym ze wzoru $\mathrm{MR} = \sum_{i=1}^{m} |y_{i+1} - y_i|/m$. Wyróżnij te obserwacje z y, które znajdą się poza przedziałem $[\mathrm{LCL}, \mathrm{UCL}]$ innym symbolem i kolorem. Pamiętaj, by poprawnie uwzględniać atrybuty `frequency`, `start` itp. szeregu czasowego, por. `?time`.

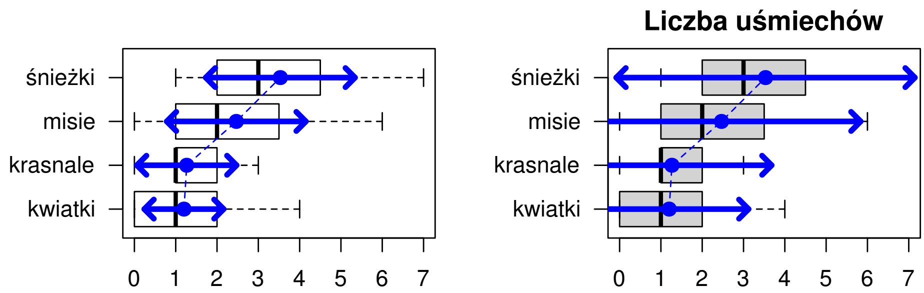
```
set.seed(1234)
y <- ts(c(rnorm(200, 0, 1)), frequency = 12, start=2000)
ctrlchart(y, 20, type = "o")
```

Ciekawostka. W przypadku, gdy obserwacje pochodzą z rozkładu normalnego, tak skonstruowany wykres jest tzw. kartą kontrolną pojedynczych pomiarów stosowaną często w statystycznej kontroli jakości. Obserwacje poza przedziałem $[\mathrm{LCL}, \mathrm{UCL}]$ *interpretuje się wówczas jako sygnały alarmowe świadczące o potencjalnym rozregulowaniu się procesu produkcji.*



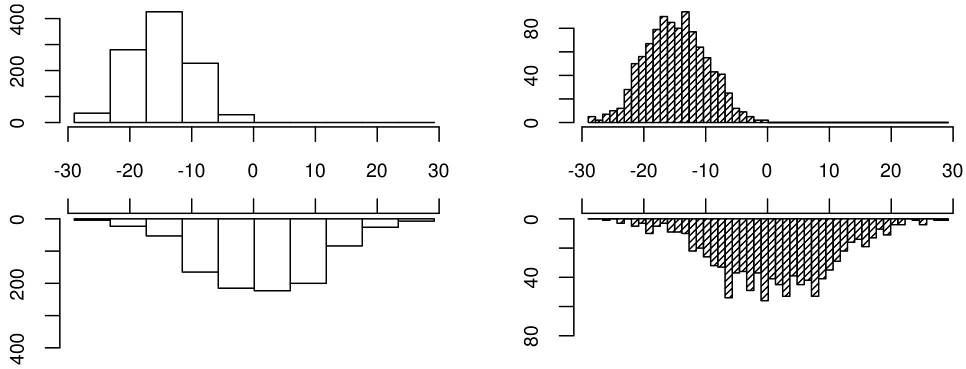
Zadanie 12.3. [AO] Napisz funkcję `myboxplots()`, która dla danego wektora liczbowego `x` i czynnika `g` o takiej samej długości narysuje za pomocą funkcji `boxplot()` wykresy skrzynkowe dla `x` podzielonego na odpowiednie klasy. Dodatkowo na rysunku należy zaznaczyć średnią wartość obserwacji z każdej klasy oraz przedział średnia $\pm$ $d\times$odchylenie standardowe, gdzie d jest argumentem wejściowym domyślnie równym 1. Średnie należy połączyć za pomocą łamanej. Kolejność rysowania „pudełek" – tak, by średnie były w kolejności rosnącej.

```
set.seed(123)
x <- as.integer(c(rpois(15, 1), rpois(15, 2),
   rpois(15, 1.4), rpois(15, 4)))
g <- factor(rep(c('krasnale', 'misie', 'kwiatki', 'śnieżki'), each=15))
myboxplots(x, g) # lewy rysunek
myboxplots(x, g, 2, main='Liczba uśmiechów', col='lightgray') # prawy
```



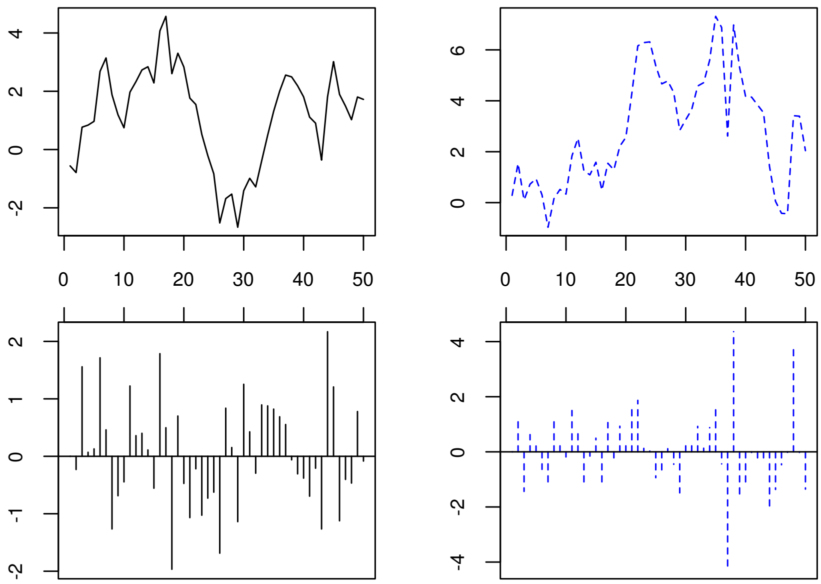
Zadanie 12.4. [AO] Napisz funkcję `hist2()`, która dla danych dwóch wektorów liczbowych `x` i `y` o długości, odpowiednio, n_x i n_y narysuje dwa histogramy na jednym rysunku. Granice klas histogramów oraz zakresy rysunków na OX i OY powinny być w obydwu przypadkach takie same. Liczbę klas kontroluje (nie wprost) argument wejściowy `h`, domyślnie równy $\lceil \log_2(\max\{n_x, n_y\}) + 1\rceil$.

```
set.seed(321)
x <- rnorm(1000, -15, 5); y <- rnorm(1000, 1, 9)
hist2(x, y) # h domyslne, lewy rysunek
hist2(x, y, 50, density=30) # h=50, prawy rysunek
```



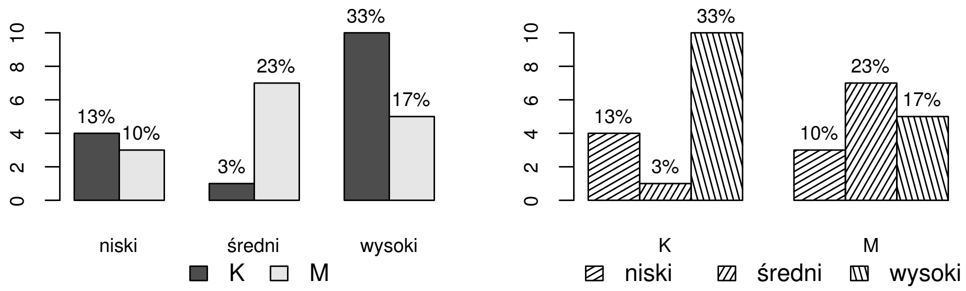
Zadanie 12.5. Napisz funkcję `plotdiff()`, która za pomocą dwóch wywołań funkcji `plot()` z odpowiednimi parametrami oraz kilku funkcji niskopoziomowych utworzy dwa rysunki na jednej stronie. Na górnym rysunku ma znaleźć się wykres x_i w zależności od i dla danego wektora liczbowego x, a na dolnym – wykres przyrostów, tj. $x_i - x_{i-1}$. Rysunki powinny mieć jedną, wspólną etykietowaną oś OX, tak jak na poniższych ilustracjach.

```
set.seed(123)
plotdiff(cumsum(rnorm(50))) # rysunek po lewej stronie
plotdiff(cumsum(rt(50,5)), col=4, lty=2) # rysunek po prawej stronie
```



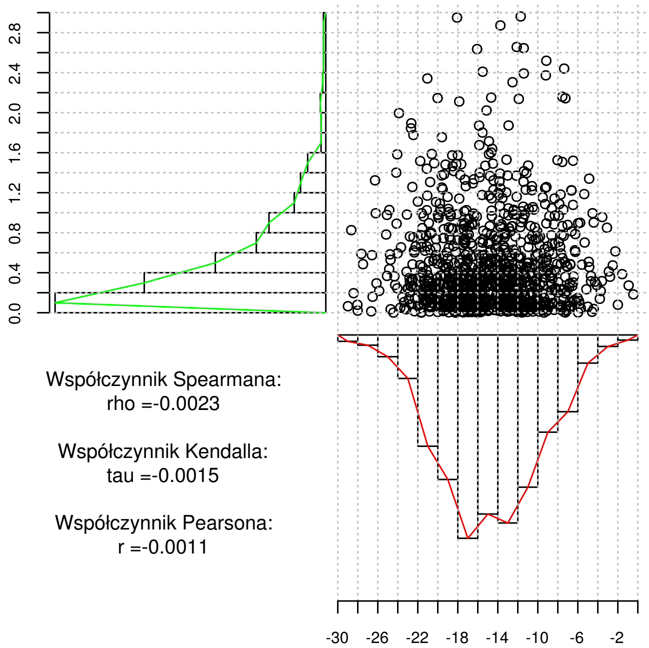
Zadanie 12.6. [AO] Napisz funkcję `mymultibarplot()`, która dla dwóch czynników g i `h` o tej samej długości skonstruuje wykres słupkowy zawierający liczności wystąpień obserwacji dla każdej kombinacji poziomów czynników. Argument `col` określa kolor wypełnienia słupków dla każdego poziomu czynnika g (domyślnie – poziomy szarości, zob. `?gray.colors`).

```
set.seed(1234)
g <- factor(c(rep('M', 15), rep('K', 15)))
h <- factor(sample(c('wysoki', 'średni', 'niski'), replace=TRUE, 30))
table(g, h)

##    h
## g   niski średni wysoki
##   K     4      1     10
##   M     3      7      5

mymultibarplot(g, h) # lewy rysunek
mymultibarplot(h, g, density=25, col=1, angle=c(30, 60, 105)) # prawy
```



Zadanie 12.7. [AO] Napisz funkcję `multi2d()`, która dla dwóch wektorów liczbowych równej długości narysuje wykres rozrzutu, histogramy oraz wartości różnych współczynników korelacji (zob. `?cor`).

```
set.seed(321); x <- rnorm(1000, -15, 5); y <- rexp(1000, 2)
multi2d(x, y)
```



Przetwarzanie obrazów rastrowych

Uwaga. Jeśli nie zaznaczono inaczej, poniżej zakładamy, że operacje mają być wykonywane na bitmapach w skali szarości. Wybrane zadania warto jednak spróbować także rozwiązać korzystając z bitmap w modelu RGB.

Zadanie 12.8. Najczęściej używanym przekształceniem obrazu jest zmiana jasności i kontrastu. Napisz funkcję, która dla danej bitmapy oraz parametrów `b` $\in [-1, 1]$ i `c` $\in [-1, 1]$ zwróci nową bitmapę, w której każdy piksel p został przekształcony zgodnie ze wzorami:

$$
\begin{aligned}
p &:= (1 + \mathtt{b})p \text{ jeśli } \mathtt{b} < 0, \\
p &:= p + \mathtt{b}(1 - p) \text{ jeśli } \mathtt{b} \geqslant 0, \\
p &:= \left(\left((p - 0{,}5)\tan\left(\tfrac{\pi}{4}(\mathtt{c} + 1)\right) + 0{,}5\right) \wedge 1\right) \vee 0.
\end{aligned}
$$

Zadanie 12.9. Efekt „posteryzacji" polega na odpowiednim zdyskretyzowaniu „ciągłej" jasności pikseli z przedziału $[0, 1]$ na zbiór o ograniczonej liczbie wartości. Napisz funkcję `posterize()`, która dla danej bitmapy i parametru $k \in \mathbb{N}$ zwraca bitmapę składającą się z pikseli o jasności $\{0/k, 1/k, \ldots, k/k\}$ (sposób przekształcenia: dowolny sensowny).

Zadanie 12.10. Napisz dwie funkcje dokonujące poziomego oraz pionowego odbicia lustrzanego (ang. *flip*) danej bitmapy. Np. dla odbicia lustrzanego pionowego pierwszy wiersz zamieniany jest z ostatnim, drugi z przedostatnim itd.

Zadanie 12.11. Napisz funkcję, która obraca daną bitmapę o 90, 180 lub 270 stopni (kąt obrotu podany za pomocą odpowiedniego parametru).

Zadanie 12.12. Napisz funkcję, która dla danej kolorowej bitmapy w modelu RGB zwróci odpowiadającą jej bitmapę w skali szarości. W praktyce najczęściej korzysta się nie ze średniej arytmetycznej wartości kanałów, lecz ze średniej ważonej: $0{,}2126R + 0{,}7152G + 0{,}0722B$, co ponoć daje bardziej przyjazny oczom wynik. Zaimplementuj obydwie metody konwersji i porównaj rezultaty.

Zadanie 12.13. Napisz funkcję implementującą efekt „pikselizacji" stopnia $\mathbb{N} \ni k \geqslant 2$ danej bitmapy. Możesz założyć, że wysokość i szerokość bitmapy dzielą się bez reszty przez k (jeśli tak nie jest, zgłoś błąd). Przekształcenie to polega na podzieleniu bitmapy na kwadraty o boku k pikseli i wypełnieniu ich kolorem powstającym przez uśrednienie wartości pikseli znajdujących się tamże.

Zadanie 12.14. Napisz funkcję `zmienrozmiar()`, która zmienia rozmiar danej bitmapy o rozmiarze $w \times h$ pikseli do danego (parametry funkcji) rozmiaru $w' \times h'$ pikseli, $w', h' \in \mathbb{N}$, $w' \geqslant w, h' \geqslant h$. Wyjściowy obraz powinien zostać wypełniony kolorem czarnym, a następnie należy zamieścić w nim oryginalny obraz „prawie" na jego środku (możliwy błąd: ± 1 piksel w kierunku poziomym i pionowym).

Zadanie 12.15. Napisz funkcję `skaluj()`, która skaluje daną bitmapę rozmiaru $w \times h$ pikseli do rozmiaru $sw \times sh$, gdzie $s \in \mathbb{N}$ jest danym (parametr funkcji) współczynnikiem skalowania. Bloki o rozmiarze $s \times s$ odpowiadające oryginalnym pikselom wypełniamy jednym, tym samym kolorem.

$\star$ **Zadanie 12.16.** Napisz funkcję `skaluj2()`, która zmienia rozmiar danej bitmapy do $w \times h$ pikseli, $w, h \in \mathbb{N}$. Zastosuj biliniową interpolację barw.

Zadanie 12.17. Napisz funkcję implementującą efekt typu *blue box*. Dla danych bitmap `przod` i `tyl` o tym samym rozmiarze oraz nasycenia kanału szarości $k \in [0, 1]$ wynikiem powinno być „złączenie" bitmap: wszystkie piksele o kolorze k w `przod` są zastępowane odpowiadającymi im pikselami z `tyl`.

$\star$ **Zadanie 12.18.** Uzupełnij funkcję wyznaczającą splot bitmapy z macierzą transformacji tak, by uwzględniała także piksele leżące na obrzeżach bitmapy wejściowej, zgodnie z informacją podaną w rozdz. 4.

$\star\star$ **Zadanie 12.19.** Dana jest bitmapa, punkt (i, j) i nasycenie kanału szarości $k \in (0, 1)$. Zamień na k jasność wszystkich pikseli, które tworzą maksymalny spójny obszar sąsiadujący z pikselem na pozycji (i, j) w sensie posiadania tej samej jasności, co jasność piksela (i, j). W literaturze taki algorytm zwany jest *flood fill*.

13 Generowanie raportów przy użyciu pakietu knitr

Zadanie 13.1. Ramka danych `Cars93` z pakietu MASS zawiera informacje na temat 93 modeli samochodów dostępnych w sprzedaży w roku 1993.

```
library(MASS) # ładowanie pakietu
data(Cars93)  # ładowanie ramki danych
```

Przeprowadź wstępną (eksploracyjną) analizę tego zbioru danych. Wygeneruj raport w knitr.

Zadanie 13.2. Stwórz raport stanowiący „materiały dydaktyczne” na temat funkcji `apply()`, `lapply()`, `sapply()` i `mapply()`. Opowiedz o argumentach, które może przyjmować każda z ww. funkcji. Zaprezentuj kilkanaście przykładów na różnych danych.

Zadanie 13.3. Utwórz raport zawierający wykaz (w postaci rysunków) znaczenia różnych parametrów graficznych funkcji `lines()` i `points()`, w tym `pch`, `lwd`, `lty`, `cex` oraz `type`.

Zadanie 13.4. Funkcja `colors()` zwraca wektor nazw wszystkich barw obsługiwanych przez R-a. Wygeneruj raport, którego spis treści zawiera nazwy wszystkich barw z odnośnikami do innych miejsc dokumentu, w którym można zobaczyć wygląd danej barwy. Wygląd każdej barwy zilustruj rysując prostokąt, w którego wnętrzu zostanie wypisana jej nazwa (tak by dało się ją odczytać).

Zadanie 13.5. Napisz funkcję, która dla podanej na wejściu ramki danych wygeneruje kod LaTeX-a albo HTML5 tabelki reprezentującej „tekstowo” tę ramkę danych. Następnie stwórz raport, w którym zademonstrujesz, jak działa Twoja funkcja na wbudowanych ramkach danych `cars`, `as.data.frame(WorldPhones)`, `Orange`, `Puromycin` oraz `iris`.

Zadanie 13.6. Wywołując `installed.packages()[,1]` otrzymasz wektor napisów zawierający nazwy wszystkich zainstalowanych na Twoim komputerze pakietów R-a. Funkcja `packageDescription()` zwraca listę zawierającą podstawowe informacje na temat danego pakietu, dla przykładu:

```
str(packageDescription("FuzzyNumbers")[1:7], strict.width="wrap")
## List of 7
## $ Package : chr "FuzzyNumbers"
## $ Title : chr "Tools to deal with fuzzy numbers"
## $ Type : chr "Package"
## $ Authors@R : chr "c(\n person(\"Marek\", \"Gagolewski\", role = c(\"aut\", \"cre\"),\n
##    email = \"gagolews@examine.com\"),\n person(\"Ja"| __truncated__
## $ Description: chr "The FuzzyNumbers package provides S4 classes and methods\n to deal
##    with Fuzzy Numbers that allow for computations of arithme"| __truncated__
## $ Version : chr "0.3-3"
## $ Date : chr "2014-01-03"
```

Po załadowaniu pakietu, za pomocą wywołania `ls("package:NazwaPakietu")` poznasz nazwy wszystkich obiektów, które zostały zdefiniowane w danym pakiecie (funkcji, ramek danych itp.).


```r
library("FuzzyNumbers")
head(ls("package:FuzzyNumbers"))
## [1] "alphacut"                        "alphaInterval"
## [3] "ambiguity"                       "approxInvert"
## [5] "as.FuzzyNumber"                  "as.PiecewiseLinearFuzzyNumber"
```

Utwórz estetyczny raport, w którym zawarte są podstawowe informacje na temat każdego zainstalowanego pakietu (w tym jego nazwa, informacje o licencji, twórcy, do czego pakiet służy) oraz wszystkich zdefiniowanych w nim obiektach.

15 Symulacje i wnioskowanie statystyczne

Zadanie 15.1. Rozważmy zagadnienie obliczania granic dwustronnych przedziałów ufności dla parametru p rozkładu $\mathrm{Bern}(p)$. Przedział dokładny jest postaci:

$$\left[\texttt{qbeta}\left(\frac{\beta}{2}, m\hat{p}, m(1-\hat{p})+1\right), \texttt{qbeta}\left(1-\frac{\beta}{2}, m\hat{p}+1, m(1-\hat{p})\right)\right],$$

z kolei przedział przybliżony (oparty na CTG), dość często podawany w literaturze, dany jest wzorem:

$$\left[\hat{p} - \texttt{qnorm}(1-\beta/2)\frac{\sqrt{\hat{p}(1-\hat{p})}}{m}, \hat{p} + \texttt{qnorm}(1-\beta/2)\frac{\sqrt{\hat{p}(1-\hat{p})}}{m}\right],$$

gdzie $1-\beta$ jest zadanym poziomem ufności.

Oszacuj symulacyjnie prawdopodobieństwa pokrycia parametru p 90% przedziałem ufności. Wyniki przedstaw w postaci jednego wykresu (dla różnych $p \in (0,1)$).

Zadanie 15.2. Wyznacz metodą całkowania Monte Carlo przybliżoną wartość:

1. pola koła o promieniu 1 $(=\pi)$.
2. $\int_{-2}^{-1} x^3\,dx$ $(=-3{,}75)$.

Zadanie 15.3. Zbadaj moc testu Shapiro-Wilka dla $n = 10, 25, 100$-elementowych próbek z rozkładów: (a) Cauchy'ego, (b) t o 5 stopniach swobody, (c) jednostajnego oraz (d) χ^2 o 10 stopniach swobody. Wyniki zapisz w postaci pojedynczej ramki danych (nazwa wiersza – opis rozkładu, nazwa kolumny – wartość n, wartości wewnątrz – oszacowana moc). Program zaprojektuj tak, by dane wejściowe (rozkłady i liczności prób) można było łatwo modyfikować w jednym miejscu oraz by cała wynikowa ramka danych była generowana za pomocą jednej funkcji.

Zadanie 15.4. Porównaj moc testu Shapiro-Wilka z mocą różnych testów normalności z pakietu `nortest` (np. `ad.test()`, `cvm.test()` oraz `pearson.test()`) dla różnych n i różnych rozkładów. Wyniki dla każdego n przedstaw w postaci oddzielnej ramki danych.

```r
install.packages("nortest") # tylko raz
library("nortest")
```

Zadanie 15.5. Sprawdź, czy test Kołmogorowa-Smirnowa zachowuje zadany poziom istotności, jeśli estymujemy nieznany parametr λ rozkładu wykładniczego za pomocą estymatora $\hat{\lambda} = n/\sum_{i=1}^{n} X_i$ (jak jest zalecane w niektórych podręcznikach). Porównaj oszacowane prawdopodobieństwo odrzucenia prawdziwej H_0 z wersją dla parametru λ podanego w sposób jawny. Rozpatrz $n = 10, 100, 1000$ oraz $\lambda = 10, 1$ i $0{,}1$.

⋆ **Zadanie 15.6.** Zbadaj rozkład liczby prób w grze *Memo* zakładając, że gracz po odsłonięciu kolejnej karty zapomina każdy zapamiętany typ obrazka z pewnym małym prawdopodobieństwem p (fakt wystąpienia „zaniku pamięci” jest niezależny dla każdej z pamiętanych kart).

⋆ **Zadanie 15.7.** Zbadaj rozkład liczby prób w grze *Memo* zakładając, że gracz cechuje się „pamięcią” ograniczoną tylko do r ostatnio „widzianych” typów obrazków.

17 Środowiska

Zadanie 17.1. Użyj obiektu typu środowisko do zliczenia liczby wystąpień wszystkich niepowtarzalnych słów znalezionych w danym pliku tekstowym. Porównaj wydajność takiej implementacji z równoważną, ale używającą zwykłej listy oraz z taką, która korzysta z czynników (`factor`).

⋆ **Zadanie 17.2.** Napisz funkcję, która dla danej listy środowisk zwróci definicję grafu w postaci pliku źródłowego programu Graphviz[1]. Wierzchołkami grafu są podane środowiska i wszystkie ich środowiska otaczające (cała ścieżka wyszukiwania), a krawędzie reprezentują informacje o tym, że jedno środowisko jest otaczane przez drugie.

18 Syntaktyka i semantyka języka R

Zadanie 18.1. Poznaj zasadę działania funkcji `local()` studiując jej kod źródłowy, zob. `print(local)`.

Zadanie 18.2. Poznaj zasadę działania funkcji `with()` studiując jej kod źródłowy, zob. `print(with.default)`.

Zadanie 18.3. Poznaj zasadę działania funkcji `within()` studiując jej kod źródłowy, zob. `print(within.data.frame)`.

Zadanie 18.4. Poznaj zasadę działania funkcji `transform()` studiując jej kod źródłowy, zob. `print(transform.data.frame)`.

Zadanie 18.5. Poznaj zasadę działania funkcji `aggregate()` studiując jej kod źródłowy, zob. `print(aggregate.data.frame)`.

Wskazówki do ćwiczeń

Wskazówka do zadania 3.6. Skorzystaj z funkcji `sort()` lub `order()`.

Wskazówka do zadania 3.12. Możesz skorzystać z funkcji `diff()` oraz `which()`. Alternatywnie, możesz zapisać rozwiązanie z wykorzystaniem tylko `which()` i operatora indeksowania „`[`”. Ta ostatnia będzie też działać dla wektorów złożonych z napisów.

[1] Zob. *www.graphviz.org* – narzędzie dot, zob. też R-owy pakiet Rgraphviz.

Wskazówka do zadania 3.14. Skorzystaj z operatora modulo i np. funkcji `trunc()`.

Wskazówka do zadania 4.4. Skorzystaj z funkcji `"["()`.

Wskazówka do zadania 6.4. Przed wywołaniem funkcji z zadania 6.3 przydać się może funkcja `min()` i `max()`, zaś potem – `rep()`.

Wskazówka do zadania 7.3. Przydatne funkcje: `format()`, `cat()` i `paste()`.

Wskazówka do zadania 8.1. Funkcję da się zaimplementować nie używając jawnie żadnej pętli, korzystając z odpowiedniego użycia funkcji `seq()` i operatora indeksowania.

Wskazówka do zadania 8.8. Funkcję da się zaimplementować przy użyciu tylko dwóch pętli `for` albo bez jawnych pętli za pomocą `outer()` (sugerujemy porównać wydajność obydwu rozwiązań).

Wskazówka do zadania 8.9. W przypadku (c) możesz skorzystać z funkcji `paste()`, by wygenerować jeden wektor czynnikowy na podstawie dwóch zmiennych grupujących.

Wskazówka do zadania 8.17. Dwie ostatnie kolumny należy posortować.

Wskazówka do zadania 9.1. Skorzystaj z `strptime()`. Sprawdzenie pełnoletniości najwygodniej dokonać przy użyciu obiektu typu `POSIXlt`.

Wskazówka do zadania 9.2. Skorzystaj z `str_extract_all()` i wyrażeń regularnych.

Wskazówka do zadania 9.3. Tutaj konieczna będzie konwersja napisu do wektora liczb całkowitych.

Wskazówka do zadania 9.6. Ciekawym pomysłem byłoby rozważenie użycia `agrep()`.

Wskazówka do zadania 9.9. Skorzystaj z `strftime()` i obiektów klasy `POSIXct`.

Wskazówka do zadania 9.10. Zwróć uwagę na poprawną obsługę polskich „ogonków”.

Wskazówka do zadania 10.1. Jeśli Twój R używa polskich ustawień lokalizacyjnych, trzeba je tymczasowo zmienić na angielskojęzyczne:

```
lok <- Sys.getlocale('LC_TIME')
Sys.setlocale('LC_TIME', 'C')
... przetwarzanie daty ...
Sys.setlocale('LC_TIME', lok)
```

Wskazówka do zadania 10.5. Musisz poszukać w internecie, jakie napisy definiują poprawne URL stron internetowych. Jeśli znajdziesz adres, w którym protokół `http://` nie został podany, dołącz go do wynikowego napisu.

Wskazówka do zadania 11.1. W przypadku argumentu typu `factor` użyj funkcji `table()`.

Wskazówka do zadania 11.9. Pomocne mogą być funkcje `cut()`, `range()` oraz `pretty()`, choć niekoniecznie.

Wskazówka do zadania 12.3. Przydatnym może być obiekt zwracany przez funkcję `boxplot()`

Wskazówka do zadania 12.4. Godnym uwagi jest parametr `ylim` metody `plot.histogram()`. Możemy założyć, że zawsze będzie rysowany histogram dla liczby obserwacji, a nie ich proporcji.

Wskazówka do zadania 12.6. Warto wykorzystać wartość zwracaną przez `barplot()`. Ponadto jeden z parametrów (jaki?) tej funkcji służy do sprawienia, by słupki wyświetlały się w „grupach”.

Wskazówka do zadania 12.7. Do rysowania histogramów być może przydadzą się funkcje `hist()` oraz `barplot()`.

Wskazówka do zadania 12.9. Spróbuj skorzystać z funkcji `cut()`.

Wskazówka do zadania 12.11. Zauważ, że niektóre obroty wymagają zwrócenia bitmapy o innym rozmiarze niż wejściowa.

Wskazówka do zadania 13.4. Do generowania „interaktywnego” spisu treści wykorzystaj na przykład:

1. w LaTeX-u: polecenia \label, \pageref,
2. w HTML5: znaczniki <a id='...' />, ...

Wskazówka do zadania 13.5. LaTeX: zob. otoczenie `tabular`, `longtable` albo `tabularx`. HTML5: zob. znacznik `table`.

Poczytaj w dokumentacji knitr-a (najlepiej na stronie internetowej pakietu) na temat opcji `results` (*chunk options*).

Wskazówka do zadania 13.6. Wygenerowany dokument z pewnością będzie dość pokaźnych rozmiarów. Na pewno przyda się spis treści. Informacje o pakiecie najlepiej wyświetlać w postaci tabelki, a informacje o obiektach – np. za pomocą listy wypunktowanej.