Podstawy R: operacje na wektorach atomowych, listy oraz funkcje (cz. 2)

Zadanie 4.1.
[MG] Przestudiuj samodzielnie dokumentację funkcji `rle()`. Funkcja ta zwraca obiekt typu lista. Co najważniejsze, do każdego z dwóch komponentów tego obiektu (obydwa są zwykłymi wektorami atomowymi) możemy się odwołać przy użyciu wywołań:

```
rle(x)$lengths
rle(x)$values
```

Znajdź wartość, która powtarza się najczęściej (tj. modę) w danym wektorze liczb całkowitych `x`. Jeśli rozwiązanie nie jest jednoznaczne, zwróć dowolną modę.

Zadanie 4.2.
[MB] Napisz funkcję `calkaMonteCarlo()`. Funkcja przyjmuje cztery argumenty. Pierwszy to ściśle monotoniczna (na przedziale [`a`, `b`]), przyjmująca wartości nieujemne na przedziale [`a`, `b`] jednoargumentowa funkcja rzeczywista `f` (tego, czy jest monotoniczna, nie sprawdzaj, zakładamy, że tak jest; co do wartości nieujemnych, sprawdź je jedynie dla $f(a)$ i $f(b)$).

Kolejne argumenty, `a` i `b`, to przedział, na którym będziemy całkować funkcję `f`. Ostatnim argumentem jest liczba naturalna `n`, która będzie wyznaczać nam dokładność obliczeń (im większe `n`, tym dokładniejsze obliczenia). Niech wartość domyślna tego parametru wynosi 1000.

Funkcja ma całkować w następujący sposób:

1. niech $f_{min} = \min(f(a), f(b))$, a $f_{max} = \max(f(a), f(b))$;
2. wylosuj n punktów (x_1, x_2) z rozkładu jednostajnego na kwadracie $[a, b] \times [f_{min}, f_{max}]$ (wsk. `?runif`);
3. wyznacz frakcję punków leżących pod wykresem funkcji `f` (czyli takich że $x_2 \leqslant f(x_1)$);
4. uzyskany w punkcie 3. wynik przeskaluj mnożąc go przez miarę (powierzchnię) obszaru $[a, b] \times [f_{min}, f_{max}]$;
5. do wyniku dodaj $(b - a)f_{min}$.

Przykłady:

```
set.seed(123)
f <- function(x) sin(x)
a <- 0
b <- 1
n <- 10000
calkaMonteCarlo(f,a,b,n)
```

```
## [1] 0.461042
```

```
f <- function(x) x + 1
calkaMonteCarlo(f,a,b,n)
```

```
## [1] 1.5101
```

```
f <- function(x) x^2 + 2
calkaMonteCarlo(f,a,b,n)
```

```
## [1] 2.3304
```

Zadanie 4.3.
[MG] Dany jest wektor atomowy `x` oraz nieujemna liczba całkowita `k`. W wyniku wywołania `sample(x, k, replace=TRUE)` otrzymujemy wybrany pseudolosowo (ze zwracaniem) wektor składający się z `k` elementów z `x`. Napisz fragment kodu równoważny tej operacji. Przykład:

```
x <- c('a', 'b', 'c')
k <- 5
y <- ...
print(y)
## [1] "a" "c" "c" "a" "b"
```

Wskazówka: `runif(n, a, b)` generuje `n` obserwacji pseudolosowych z rozkładu jednostajnego na $(\texttt{a}, \texttt{b})$.

Zadanie 4.4.
[MG] Dany jest wektor liczb całkowitych `x`. Usuń z niego wszystkie wartości powtarzające się, nie korzystając z funkcji `unique()`, `duplicated()`, ani `anyDuplicated()`. Na przykład jeśli `x == c(1,2,1,4,3,4,1)`, to w wyniku powinniśmy otrzymać `c(1,2,4,3)`.

Zadanie 4.5.
[AC] Dane są dwa wektory `x` oraz `y` o elementach całkowitych dodatnich. Wykorzystując (koniecznie) funkcję `tabulate()`, napisz kod, który wypisze na konsolę wynik równoważny rezultatowi wywołania `intersect(x,y)`.

Zadanie 4.6.
[AC] Dany jest wektor liczbowy `x` oraz `y`. Napisz kod, który wyznaczy wartości zdefiniowanej poniżej funkcji $\widehat{F}_{\texttt{x}}(\cdot)$ we wszystkich punktach z danego wektora `y`. Dokładniej, k-ta współrzędna wynikowego wektora jest postaci:

$$\widehat{F}_{\texttt{x}}(y_k) = \frac{|\{x_i : x_i \leqslant y_k\}|}{|\texttt{x}|},$$

gdzie $|\texttt{x}|$ oznacza liczbę obserwacji w `x`. Możesz korzystać z funkcji `findInterval()` oraz `rle()` (por. zad. 4.1), lecz nie z `splinefun()` ani `approxfun()`. Zauważ, że wartości w `x` mogą się powtarzać.

Zadanie 4.7.
[MG] Napisz funkcję `merge_string_list()`, która złączy wszystkie napisy z danej jako argument `x` listy wektorów napisów w jeden napis. Argument `sep` (domyślnie: napis pusty) określa sposób rozdzielania poszczególnych elementów.

Zadanie 4.8.
[MB] Napisz funkcję `posortowanePunkty()`, która jako argumenty przyjmuje: (a) wartość naturalną n, (b) wartość naturalną m oraz (c) wektor nm wartości liczbowych, przy użyciu którego dane są współrzędne m punktów w $\mathbb{R}^n$. Na przykład dla $n = 2$ i $m = 3$ podany wektor interpretujemy jako $(x_1, x_2, x_3, y_1, y_2, y_3)$, a dla $n = 3$ i $m = 2$ jako $(x_1, x_2, y_1, y_2, z_1, z_2)$.

Funkcja `posortowanePunkty()` ma za zadanie policzyć odległości (euklidesowe) punktów od początku układu współrzędnych, tj. $(0, 0, \ldots, 0)$. Następnie sortuje ona te odległości rosnąco. W wyniku jej działania otrzymujemy wektor punktów (w takim samym formacie, co wektor wejściowy), przy czym współrzędne punktów są posortowane rosnąco względem odległości od środka układu współrzędnych. (wskazówka: użyj `order()`).

Zadanie 4.9.
[MG] Napisz funkcję `approxinvert()`, która jako argumenty przyjmuje: (a) zwektoryzowaną (tj. taką, która dla n-elementowego wektora liczbowego zwraca n-elementowy wektor liczbowy) funkcję `f`, o której wiemy, że jest ciągła i ściśle monotoniczna (tego warunku nie sprawdzamy) na pewnym przedziale $[\texttt{a}, \texttt{b}]$; (b) wektor liczbowy `y` o elementach z $[\texttt{f}(\texttt{a}), \texttt{f}(\texttt{b})]$; (c) wartość rzeczywistą `a`; (d) wartość rzeczywistą $\texttt{b} > \texttt{a}$; (e) wartość całkowitą dodatnią $\texttt{k} > 2$, domyślnie 100.

Funkcja ta ma wyznaczać przybliżenie odwrotności `f` w punktach z `y`, tj. $\texttt{f}^{-1}(\texttt{y})$ przy użyciu interpolacji (np. liniowej) `f` w `k` równoodległych punktach z przedziału $[\texttt{a}, \texttt{b}]$.

Wskazówka: możesz użyć funkcji `approxfun()` bądź `splinefun()` – zwracają one obiekty tupu funkcja.

Zadanie 4.10.
[JL] Napisz funkcję `wystarczy()`, która przyjmuje jako argument listę wektorów (dowolnej długości), liczby rzeczywiste `r` oraz `R` takie, że `r` $<$ `R` oraz funkcję agregującą `fun` (dowolną funkcję przyjmującą na wejściu wektor liczb i zwracającą wartość rzeczywistą, np. `mean`). Funkcja ta zwraca listę tych wektorów z listy podanej na wejściu, dla których wartość funkcji `fun` na wektorze wynosi co najmniej `r` i co najwyżej `R`. Przykład:

```
wystarczy(list(c(6, 3, 10), c(5, 10, 15), 4), 20, 30, sum)
## [[1]]  [1] 5 10 15
```

Zadanie 4.11.
[JL] Napisz funkcję `distance()`, która dla prostej o równaniu $\mathbf{a} \cdot \mathbf{x} + b = 0$ („$\cdot$” oznacza iloczyn skalarny) w przestrzeni $\mathbb{R}^n$ wyznaczy odległość punktu $p \in \mathbb{R}^n$ od tej prostej. Odległość ta wyraża się wzorem:

$$d = \frac{|\mathbf{a} \cdot \mathbf{p} + b|}{\|\mathbf{a}\|_2},$$

gdzie $\|\mathbf{x}\|_2$ oznacza normę euklidesową wektora $\mathbf{x} \in \mathbb{R}^n$. Funkcja ta przyjmuje argumenty w domyślnej kolejności $\mathbf{p}$, $\mathbf{a}$, b.

```
distance(c(2, -2, 0.5), c(2, 1, 4), b = -4)
## [1] 0
distance(c(0, 0), c(1, 1), b = -2.0)
## [1] 1.414214
```

Zadanie 4.12 (*AC; Generowanie obserwacji z zadanego rozkładu p-stwa).**
Napisz funkcję `gendyskr()`, która jako argumenty przyjmuje: (a) wartość całkowitą dodatnią `n`; (b) k-elementowy wektor liczbowy `x` o unikatowych wartościach; (c) k-elementowy wektor prawdopodobieństw `p` – jeśli wartości w `p` nie sumują się do 1, należy go przed wykonaniem obliczeń unormować i wygenerować ostrzeżenie.

Naszym zadaniem jest implementacja bodaj najprostszego algorytmu, który generuje `n`-elementową pseudolosową próbkę z rozkładu dyskretnego zmiennej losowej X określonej jako $\mathrm{P}(X = \mathtt{x}_i) = \mathtt{p}_i$ $(\forall i = 1, \ldots, k)$. Pojedynczą wartość otrzymujemy następująco:

1. Wygeneruj obserwację u z rozkładu jednostajnego na $(0, 1)$, por. `?runif`.
2. Znajdź $m \in \{1, \ldots, k\}$ takie, że $u \in (\sum_{j=1}^{m-1} \mathtt{p}_j, \sum_{j=1}^{m} \mathtt{p}_j]$, przy założeniu $\sum_{j=1}^{0} \cdot = 0$.
3. Zwróć $\mathtt{x}_m$ jako wynik.