Operatory i funkcje R

Po ukończeniu części 1 i 2 zakłada się, że następujące elementy są już znajome:

• Jak komunikować się z R;

• Jak zarządzać przestrzeniami roboczymi;

• Jak wykonywać proste zadania za pomocą R.

W tej części przyjrzymy się bliżej zachowaniu niektórych z najczęstszych

• Operatorzy R

• Funkcje R.

Operatory arytmetyczne

a) Przestudiuj użycie operatorów

+ Dodawanie

^ Potęgowanie

- Odejmowanie

%/% Dzielenie liczb całkowitych

* Mnożenie

%% modulus

/ Podział

: Sekwencja

%*% Mnożenie macierzy

- Jednoargumentowy minus

b) Reguły dla wyrażeń operatorowych z argumentami wektorowymi. Przestudiuj wyniki następujących instrukcji R

1. cars [,2] * 12 * 25.4 / 1000

Zinterpretuj wynik

2. 7%/%3

3. 7%%3

4. matrix(1,nrow=4,ncol=4) * matrix(3,nrow=4,ncol=4)

5. matrix(1,nrow=4,ncol=4) %*% matrix(3,nrow=4,ncol=4)

Wyjaśnij następujące instrukcje i dane wyjściowe z R:

> 1:12 + 1:3

[1] 2 4 6 5 7 9 8 10 12 11 13 15

> 1:10 + 1:2

[1] 2 4 4 6 6 8 8 10 10 12

> 1:10 + 1:3

[1] 2 4 6 5 7 9 8 10 12 11

Warning message:

In 1:10 + 1:3 :

longer object length is not a multiple of shorter object length

W powyższych przykładach zilustrowano, że R używa arytmetyki wektorowej, tj. operuje na wektorach jako całościach. Czasami zasada recyklingu jest stosowana z ostrzeżeniem lub bez ostrzeżenia. Dobrym nawykiem programowania w języku R jest korzystanie z obliczeń wektoryzacji tam, gdzie to możliwe. Należy pamiętać o skutkach zasady recyklingu, ponieważ może to prowadzić do niepożądanych rezultatów.

c) Brakujące wartości, nieskończoność i „nie liczba”. Brakująca wartość w R jest oznaczona przez NA. Wynikiem obliczeń z udziałem NA jest zawsze NA, np.

> mean(c(1,3,NA,12,5))

[1] NA

Wynik obliczenia, którego nie można przedstawić w postaci liczby, np. 0/0 jest oznaczane przez NaN:

> 0/0

[1] NaN

Uwaga: niektóre wyniki obliczeń są różnie przedstawiane przez R jako odpowiednie odpowiedniki algebraiczne, np.

> 5/0

[1] Inf

Niezdefiniowane w algebrze

> -5/0

[1] -Inf

> 5/(-0)

[1] –Inf

5/0 w R jest podane przez Inf, podczas gdy algebraicznie jest niezdefiniowane.

d) Notacja naukowa

R używa notacji dziesiętnej oraz notacji naukowej do obliczeń arytmetycznych:

> 60000000

[1] 6e+07

6 × 10⁷

> 1/6000000

[1] 1.666667e-07

1.666667 × 10⁻⁷

Notacji naukowej nie należy mylić z ex:

> exp(15)

[1] 3269017

> exp(-15)

[1] 3.059023e-07

e) Jak reprezentowane są liczby w pamięci komputera? Jakie są tego konsekwencje? Komputery używają przełączników ON/OFF (lub 1/0) do kodowania informacji. Pojedynczy przełącznik jest nazywany bitem, a grupa ośmiu bitów jest nazywana bajtem. Pojedyncza liczba całkowita jest dokładnie reprezentowana w komputerze przez stałą liczbę bajtów, tj. 32 lub 64 bity. Istnieje kilka schematów, według których w komputerze liczby całkowite są reprezentowane przez bity. Ta reprezentacja w komputerze odbywa się na poziomie, na którym R nie ma nad nim kontroli, ale R przechowuje informacje o środowisku obliczeniowym w obiekcie .Machine. Element .Machine$integer.max zwraca największą liczbę całkowitą, jaka może być reprezentowana na komputerze, na którym działa R, np.

> .Maszyna$liczba całkowita.max

[1] 2147483647

Chociaż powyższa metoda reprezentowania liczb całkowitych przez ciągi bitów zapewnia bardzo wydajny sposób przechowywania liczb całkowitych w komputerze, R zwykle traktuje liczby całkowite podobne do liczb rzeczywistych przy użyciu reprezentacji zmiennoprzecinkowej. W binarnym zapisie zmiennoprzecinkowym liczba x jest zapisywana jako ciąg zer i jedynek (mantysa) razy dwa z wykładnikiem m: x = b₀b₁b₂ … x 2^m gdzie b₀ = 1 z wyjątkiem sytuacji, gdy x = 0. jest tylko ograniczoną liczbą b, a wykładnik jest również ograniczony, dlatego na ogół nie wszystkie liczby rzeczywiste można dokładnie przedstawić w komputerze - można je co najwyżej przybliżyć. Najmniejsza liczba x taka, że 1 + x można odróżnić od 1 w komputerze, nazywana jest maszyną epsilon. W R można to uzyskać z . Machine$double.epsnp.

> .Machine$double.eps

[1] 2.220446e-16

Chociaż reprezentacja zmiennoprzecinkowa umożliwia obliczenia z bardzo małymi (w wielkości) i bardzo dużymi liczbami, powyższe ograniczenia mogą prowadzić do niedomiaru (wynik 0) lub przepełnienia (wynik ±∞), co w praktyce może mieć katastrofalne skutki. Pisanie dobrego kodu w R musi poważnie wziąć powyższe pod uwagę.

Operatory logiczne

Operatory logiczne dają TRUE, FALSE lub NA.

> Większe niż : & Elementwise and

< Mniej niż : | Elementa lub

<= Mniejszy lub równy : && AND

>= Większe lub równe : || LUB

== Równość : ! Jednoargumentowy nie

!= Nierówność

+ Dodawanie : ^ Potęgowanie

Ostrzeżenie: chociaż pisanie jest całkowicie uzasadnione

> x[x == -1] <- 0 lub > x[x == 1] <- 0

niepoprawne jest określenie

> x[x == NA] <- 0 lub > x[x == NaN] <- 0

Prawidłowy kod w tym drugim przypadku to

> x[is.na(x)] <- 0 lub > x[is.nan(x)] <- 0

Jakie są konsekwencje powyższego kodu? Zwróć także uwagę na funkcje any() i all(). Te dwie funkcje są przydatne przy łączeniu obiektów logicznych. Podaj niezbędne instrukcje aby wykonać następujące zadania:

a) Sprawdź, czy którykolwiek ze stanów w zbiorze danych state.x77 ma populacje ze wskaźnikiem analfabetyzmu nie większym niż 1,6 i wskaźnikiem morderstw powyżej 10,0.

b) Sprawdź, czy istnieje co najmniej jeden stan z dochodami większymi niż 5000 USD i oczekiwaną długością życia krótszą niż 70,0 lat.

c) Sprawdź, czy wszystkie stany o dochodach przekraczających 5000 USD mają analfabetyzm poniżej 2,0.

Co należy rozumieć przez sterujący operator logiczny?

d) Wykonaj instrukcje:

mata <- matrix(1:4,ncol=2)

matb <- matrix(c(10, 20,30,40), ncol=2)

mata; matb; mata>1 & matb>1; mata>1 | matb>1;

mata>1 && matb>1; mata>1 || matb>1

Skomentuj powyższe.

e) Jaki jest wynik

> sum(c(TRUE, !FALSE, FALSE, TRUE, TRUE)) ?

f) Jaki jest wynik

> sum(c(TRUE, !FALSE, FALSE, NA, TRUE)) ?

Wyjaśnij

Operatory <-, <<- i ~

Przed rozważeniem użycia tych operatorów odpowiedz na następujące pytania:

a) Co się stanie z obiektem aa w katalogu roboczym, jeśli w funkcji zadanie zostało wykonane

aa <- 20?

b) Teraz przestudiuj plik pomocy <<-, a następnie odpowiedz (a) czy operator <- został zastąpiony operatorem <<-.

Uwaga: używaj <<- bardzo ostrożnie.

c) Operator tyldy wykorzystywany jest w funkcjach modelowania, m.in.

lm(length ~ age).

Pierwszeństwo operatorów

Stosowane zasady przedstawiono poniżej od góry do dołu i od lewej do prawej. Zwróć uwagę na użycie

• nawiasy ( ) dla argumentów funkcji i zmiany pierwszeństwa,

• nawiasy klamrowe { } do rozgraniczania bloków instrukcji

• i nawiasy kwadratowe [ ] dla indeksów dolnych.

Prawidłowy sposób wyodrębnienia piątego elementu ciągu takiego jak 1:20 to

> (1:20)[5]

$: Subskrypcje listy i ramek danych

[], [[]] : Subskrypcje wektorowe i macierzowe; indeksowanie listy

^: Potęgowanie

%*%, %/%, %% : Mnożenie macierzy; dzielenie liczb całkowitych; moduł

*, / : Mnożenie i dzielenie

+, – : Dodawanie i odejmowanie

<, >, <=, >=, + =, != : Porównania logiczne

! : Nie jednoargumentowy

&, |, &&, || : Logiczne i; logiczne lub; kontrola i, kontrola lub

<–, <<- : Przypisanie

Wyjaśnij wyniki następujących instrukcji języka R:

> 20/4 * 12 ^2 – 6 + 14

> (20 / 4) * (12 ^2) + (-6 + 14)

> 20/4 * 12 ^(2 – 6 + 14)

> 20/4 * (12 ^2 – 6 + 14)

Wprowadzenie do funkcji w R

Funkcja w R składa się z

i. słowa kluczowego funtion, po którym następują

ii. nawiasy () zawierające argumenty funkcji z

iii. treścią funkcji (instrukcje lub linie kodu) pojawiające się w nawiasach klamrowych {}.

Argumenty funkcji można sprawdzić za pomocą polecenia

> args(nazwa funkcji)

Funkcja str(x) dostarcza informacji o obiekcie x. Jeśli x jest funkcją, jej wyjście jest podobne do tego z args(). Argumentom funkcji nadawane są wartości domyślne za pomocą konstrukcji (nazwa argumentu = wartość). Dobrą praktyką programistyczną jest intensywne używanie komentarzy do opisywania argumentów i/lub tego, co robi konkretny fragment kodu. Jaki jest pożytek z następującej funkcji:

> cube <- function(a) a^3

W powyższej funkcji argument a jest nazywany argumentem fikcyjnym. Co się stanie z obiektem a w katalogu roboczym?

Funkcje są wywoływane przez zastąpienie formalnych argumentów rzeczywistymi argumentami. Można to zrobić według stanowiska lub nazwiska. Wskazówka: Wywoływanie funkcji przy użyciu nazwanych argumentów jest mniej podatne na błędy. Utwórz następującą funkcję

> Demofunc <- function(vec = 1:10, m,k)

+ { # Function to subtract a specified constant from

+ # each element of a given vector and after subtraction

+ # divide each element by a second specified constant.

+ # The result of the above transformation is returned.

+ (vec – m)/ k

+ }

Wykonaj następujące wywołania funkcji i wyjaśnij wynik

> Demofunc(3, 2, 5)

> Demofunc(2,5)

> Demofunc(m = 2, k = 5)

> Demofunc(m = 2, k = 5, vec = 1:100)

Zwróć uwagę na użycie prompt() i package.skeleton() w celu udostępnienia nowej funkcji z plikiem pomocy.

Niektóre funkcje matematyczne

Ogólne funkcje matematyczne

abs(), exp(), log(x, base = exp(1)), log10(), gamma(), sign(), sqrt()

Funkcje trygonometryczne

cos() Cosinus : acos() arcus cosinus

sin() Sinus : asin() arc sinus

tan() Styczna : atan() arc tangens

cosh() cosinus hiperboliczny : acosh() arc cosinus hiperboliczny

sinh() hiperboliczny sinus : asinh() arc hiperboliczny sinus

tanh() tangens hiperboliczny : atanh() arc tangens hiperboliczny

Liczby zespolone

Arg(), Conj(), Mod(), Re(), Im()

Funkcje zaokrąglania i obcinania

round(), sufit(), floor(), trunc()

Zapoznaj się z plikami pomocy powyższych funkcji. Sprawdź wszystkie argumenty.

Funkcje dla macierzy

chol() : rozkład Choleskyego

crossprod() : iloczyn krzyżowy macierzy

diag() : Utwórz macierz tożsamości, macierz diagonalną lub wyodrębnij elementy diagonalne w zależności od argumentu

eigen() : Znajdowanie wektorów własnych i wartości własnych

kronecker() : Obliczanie iloczynu kroneckera dwóch macierzy

outer() : Iloczyn zewnętrzny dwóch wektorów

scale() : Centrowanie i skalowanie macierzy danych

solve() : Znajdowanie odwrotności macierzy nieosobliwej

svd() : Rozkład na wartości osobliwe macierzy prostokątnej

qr() : ortogonalizacja QR

t() : Transpozycja macierzy

Dwie inne funkcje, które odgrywają ważną rolę w obliczeniach macierzowych, to funkcje rbind() i cbind() służące do łączenia macierzy rzędowo lub kolumnowo. Popraw także funkcje matrix(), dim(), dimnames(), colnames(), rownames() oraz scan() i read.table().

a) Funkcja chol() dokonuje dekompozycji Cholesky'ego kwadratowej, symetrycznej, dodatnio określonej macierzy A =U'U, gdzie U jest górną macierzą trójkątną.

b) Funkcja crossprod (A, B) zwraca macierz A'B

c) Funkcja diag(arg) w zależności od argumentu wykonuje różne rzeczy: jeśli arg jest dodatnią liczbą całkowitą diag(age) zwraca macierz jednostkową o podanym rozmiarze; jeśli arg jest wektorem diag(arg) zwraca macierz diagonalną z elementami diagonalnymi odpowiednie elementy danego wektora; jeśli arg jest macierzą to diag(arg) zwraca wektor zawierający elementy diagonalne danej macierzy.

d) Jaka jest różnica między diag(A) a diag(diag(A))?

e) Funkcja eigen() operuje na macierzy kwadratowej i zwraca listę z nazwanymi wartościami elementów i wektorami zawierającymi odpowiednio wartości własne i wektory własne. Przestudiuj dokładnie plik pomocy eigen().

f) Funkcja kronecker() zwraca iloczyn Kroneckera A ⊗ B macierzy A i B.

g) Funkcja zewnętrzna (x, y, f) operuje na dwóch wektorach x: n × 1 i y: p × 1, aby zwrócić macierz o rozmiarze n × p z ij-tym elementem będącym wynikiem zastosowania funkcji f na x[i] i y[j]. Domyślną wartością f jest „*”.

h) Funkcja scale() ma trzy argumenty: macierz jako pierwszy argument; drugie centrum argumentów i trzecia skala argumentów. Jeśli środek = FALSE, nie jest wykonywane centrowanie kolumn argumentu macierzy, jeśli ustawione na TRUE, średnia wartość każdej kolumny jest odejmowana od odpowiednich kolumn, jeśli dany wektor wartości jest odejmowany od odpowiednich kolumn. Jeśli scale = FALSE, nie jest wykonywane skalowanie kolumn argumentu macierzy, jeśli ustawione na TRUE każda kolumna jest dzielona przez swoje odchylenie standardowe, jeśli podano wektor wartości, to każda kolumna jest dzielona przez odpowiednią wartość.

i) Funkcja solve (A, b) służy do rozwiązania równania Ax = b dla x, gdzie b może być wektorem lub macierzą, przy czym A jest macierzą kwadratową. Jeśli brakuje argumentu b, przyjmuje się, że jest to macierz jednostkowa, tak że zwracana jest odwrotność argumentu A.

j) Funkcja svd() zwraca osobliwą dekompozycję swojego argumentu macierzowego A = UDV'. Zwraca listę z trzema składnikami: u macierzą ortogonalną lub ortonormalną U; d wektor zawierający uporządkowane wartości osobliwe macierzy prostokątnej ; v ortogonalna lub macierz ortonormalna V.

k) Funkcja qr() dokonuje rozkładu QR dowolnej macierzy z Q i macierzą ortogonalną, a R jako macierz górnego trójkąta. Zapoznaj się z plikiem pomocy qr(), aby uzyskać szczegółowe informacje i zastosowania qr(). Zwróć uwagę, że macierze Q i R można uzyskać bezpośrednio przez wywołanie qr.Q(qr()) i qr.R(qr()).

l) Jakie jest znaczenie każdej z poniższych instrukcji?

rbid(a,b); rbind(1,x); rbind(a = 1:5,b = 10:14,c=20:24);

cbind(a= 1:5, b=10:14, c=20:24)

m) Napisz funkcję obliczającą wyznacznik macierzy kwadratowej. Nazwij tę funkcję det. own() w celu odróżnienia go od wbudowanej funkcji R det().

n) Gdy użytkownik jest zadowolony z funkcji, często konieczne jest udostępnienie jej we wszystkich projektach R. Przydatne jest przypisanie wszystkich takich funkcji do tej samej bazy danych lub katalogu. Użyj funkcji assign (x, object, pos = , envir =) do przechowywania funkcji det.own() we własnym katalogu funkcji R. Argument x w assign() jest ciągiem znaków służącym do przypisania nazwy do obiektu. Funkcja remove (lista nazw obiektów, pos = , envir =) może służyć do usuwania obiektów z własnej lub dowolnej innej bazy danych.

Wskazówka: Najpierw utwórz plik, a następnie użyj attach(), aby dodać go do ścieżki wyszukiwania języka R.

> save(file= " C:\\MyFunctions").

Study how save() works.

> attach("C:\\MyFunctions", pos=2).

Study how attach() works.

> assign("det.own", det.own, pos=2).

Study how assign() works.

> save(list=objects(2), file = "C:\\MyFunctions").

Wyjaśnij użycie argumentu list=objects(2).

Podsumowując: Budowa

> NAME <- object

to prosty sposób na przypisanie obiektu do nazwy. Ta forma przydziału zawsze ma miejsce w środowisku globalnym (obszarze roboczym). Przypisanie można również wykonać za pomocą funkcji save() i assign(), jak pokazano powyżej. Ta ostatnia forma przydziału jest bardziej skomplikowana, ale przypisanie nie jest ograniczone do środowiska globalnego.

o) Wynik funkcji gamma(x) to (x – 1)! Jeśli x jest nieujemną liczbą całkowitą. Teraz napisz funkcję fact(), aby obliczyć x!. Ta funkcja musi uwzględniać 0! Jak również dla liczby ujemnej lub ułamka, który został wczytany przez pomyłkę. Wskazówka: Najpierw przestudiuj użycie instrukcji if, prosząc o pomoc ?Kontrola. Zapisz tę funkcję w swoim katalogu funkcji R. Jak zamierzasz udostępnić fact() i det.own() dla dowolnego projektu R?

Funkcje sortowania

Zwróć uwagę na użycie funkcji sort(), order() i rank(). Najpierw skonstruuj MatX używając funkcji scan() i matrix(). Wyjaśnij szczegółowo, co robi order(), sortując wszystkie kolumny MatX zgodnie z wartościami w pierwszej kolumnie macierzy.

<img src=”images/matx.PNG”>

Niektóre funkcje do manipulacji danymi

apend() : łączenie wektorów; większa elastyczność niż c()

c() : Utwórz wektory

duplikat() : wyodrębnia zduplikowane wartości

match() : Dopasuj wartości i pary wektorów

pmatch() : Częściowe dopasowanie

replace() : Zamień określone wartości na wektory

unique() : Wyodrębnij unikalne wartości

a) Wstaw wektor (101, 102, 103, 104, 105) do wektora (10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20) po piątym elemencie, wykorzystując argument po funkcji append().

b) Funkcja replace() wymaga trzech argumentów x, listy i wartości. Wartości w x ze wskaźnikami podanymi w wykazie są zastępowane kolejnymi wartościami w wartościach wykorzystujących w razie potrzeby zasadę recyklingu. Wyjaśnij to, zastępując w wektorze (10, 2, 7, 20, 5, 8, 9, 20, 9, 1, 15) wartości 10, 20 i 15 zerami.

c) Znajdź unikalne wartości w wektorze (10, 2, 7, 20, 5, 8, 9, 20, 9, 1, 15).

d) Znajdź zduplikowane wartości w wektorze (10, 2, 7, 20, 5, 8, 9, 20, 9, 1, 15, 20, 20, 15).

e) Wyjaśnij użycie funkcji match(), biorąc pod uwagę różnicę między dopasowaniem (c(10,2,7,20,5,8,9,20,9,1,15), c(10,20,15)) mecz (c(10,20,15), c(10,2,7,20,5,8,9,20,9,1,15))

f) Zilustruj różnicę między match() i pmatch(), biorąc pod uwagę nazwy dni tygodnia.

Podstawowe funkcje statystyczne

cor() : Korelacja : Jeden lub dwa argumenty

cumsum() : Skumulowana suma elementów wektora :

mean() : średnia arytmetyczna : opcjonalny argument trim =

median() : Mediana : Akceptuje zmienną liczbę argumentów

min() : Minimalna wartość : Akceptuje zmienną liczbę argumentów

max() Maksymalna wartość : Akceptuje zmienną liczbę argumentów

prod() : Iloczyn elementów wektora : Akceptuje zmienną liczbę argumentów

cumprod() : Skumulowany iloczyn elementów wektora

quantile ( ) : Zwraca określone kwantyle

range() : Min i max wektora : Akceptuje zmienną liczbę argumentów

sample() : Próbka losowa : Z wymianą lub bez

sum() : Suma arytmetyczna : Używana również do liczenia

var() : Wariancja i kowariancja; używa n — 1 jako mianownika : akceptuje wektory lub macierze

sd() : odchylenie standardowe; używa n — 1 : przyjmuje wektor jako argument

Zwróć także uwagę na funkcje pmax() i pmin().

a) Znajdź średnią długość życia stanów w zestawie danych state.x77.

b) Znajdź 5% średnią obciętą dla analfabetyzmu stanów w zbiorze danych state.x77.

c) Znajdź korelację między analfabetyzmem a dochodem stanów w zbiorze danych state.x77.

d) Znajdź macierz kowariancji wszystkich zmiennych w zbiorze danych state.x77.

e) Znajdź zakres morderstwa w zestawie danych state.x77.

f) Uzyskaj szczegóły losowej próbki 10 stanów ze zbioru danych state.x77.

g) Uzyskaj dwie niezależne losowe permutacje liczb 1, 2, …, 10.

h) Napisz funkcję do obliczania współczynnika kurtozy dla próby losowej. Przetestuj swoją funkcję na zmiennej Frost w zestawie danych state.x77.

i) Napisz funkcję do obliczania współczynnika skośności dla próby losowej. Przetestuj swoją funkcję na zmiennej Murder w zestawie danych state.x77.

j) Napisz funkcję do obliczenia średniej harmonicznej wektora numerycznego. Przetestuj swoją funkcję na długości życia stanów w zestawie danych state.x77. Porównaj swoją odpowiedź z odpowiedzią w (a)

Rozkłady prawdopodobieństwa w R

Najpierw wykonaj instrukcję R

> help.search("distribution")

aby uzyskać listę dostępnych rozkładów statystycznych w R. Każdy rozkład ma nazwę identyfikacyjną poprzedzoną jedną z liter d, p, q lub r. Na przykład w przypadku rozkładu F identyfikatorem jest po prostu litera f, a dla rozkładu normalnego identyfikatorem jest norma. Poprzedzenie identyfikatora rozkładu jedną z liter d, p, q lub r zwraca wartość gęstości, prawdopodobieństwo, kwantyl lub próbę losową dla określonego rozkładu (funkcja gęstości prawdopodobieństwa lub funkcja masy prawdopodobieństwa). Zobacz rysunek dla wyjaśnienia.

<img src=”images/wyk.PNG”>

a) Znajdź losową próbkę o rozmiarze 10 z rozkładu normalnego (50, σ2 = 20).

b) Jak można zorganizować, aby wszyscy członkowie w klasie otrzymali tę samą losową próbkę? Zilustruj.

Podpowiedź: Poproś o pomoc dotyczącą funkcji set.seed().

Funkcje dla zmiennych kategorialnych

Oprócz tego, że są numeryczne lub logiczne, dane w R mogą być również kategoryczne (współczynnik w R) lub ciągami znaków. Przestudiuj szczegółowo funkcje operujące na danych czynnikowych:

cut() : Tworzy kategorie ze zmiennej ciągłej

factor() : Koduje wektor jako nominalną zmienną kategorialną

factor() : Koduje wektor jako porządkową zmienną kategorialną, gdy uporządkowany argument jest ustawiony na TRUE

levels() : Wyświetla lub ustawia poziomy zmiennej czynnika

pretty() : Tworzy wygodne punkty przerwania dla zmiennej kategorialnej

split() : Dzieli tablicę zgodnie z wartością zmiennej kategorialnej

table() : Zlicza obserwacje sklasyfikowane krzyżowo według kategorii

unclass() : Zwraca kody numeryczne reprezentujące poziomy zmiennej czynnika

a) Użyj cut(), aby utworzyć obiekt areagrp w celu podzielenia zestawu danych state.x77 na trzy grupy reprezentujące stany z obszarem w granicach odpowiednio przedziałów i . Podpowiedź: Najpierw przestudiuj argumenty cut().

b) Powtórz (a) z etykietami argumentów = ?? określić każdy stan jako Mały, Średni lub Duży w odniesieniu do jego obszaru.

c) Użyj unclass(), aby uzyskać kody numeryczne związane z każdym poziomem areagrp.

d) Powtórz (a), aby otrzymać areagrp2 zawierający pięć równo rozmieszczonych kategorii.

e) Powtórz (a), aby uzyskać aregrp3 zawierający pięć grup, z których każda zawiera 20% danych.

f) Użyj cut(), aby utworzyć obiekt ilitgrp w celu podzielenia zbioru danych state.x77 na pięć grup reprezentujących stany z analfabetyzmem w przedziale odpowiednio i .

g) Uzyskaj tabelę dwukierunkową zestawu danych state.x77 zgodnie z obszarem areagrp i ilitgrp.

Funkcje do manipulacji znakami

abbreviate( ): Generuje skróty wartości znaków

cat( ): Wyświetlaj wiadomości i/lub wartości na ekranie lub wyślij do pliku

grep( ): Szukaj wzorców w znakach

nchar( ) : Liczba znaków w ciągu

paste( ): Połącz wartości w ciągi znaków

strsplit ( ): Podziel elementy wektora znakowego x na podciągi

substring( ) : Wyodrębnia części ciągów znaków

a) Jaka jest zwrócona wartość grep („ia”, state.name)?

b) Omów użycie grep („ia”, nazwa.stanu).

c) Omów dane wyjściowe obiektów (pos = grep("stats", search())).

d) Użyj wklej, aby utworzyć nazwy zmiennych: var1, var2, …, var100.

e) Powtórz (d), aby utworzyć nazwy zmiennych: zm_1, zm_2, …, zm_100.

f) Omów wyniki

substring (paste(letters,collapse=""),

1:nchar(paste(letters,collapse="")),

1:nchar(paste(letters,collapse=""))).

g) Uzyskaj kopię drugiego akapitu we wstępie do tej książki w oknie poleceń języka R. Użyj tej kopii, aby obliczyć liczbę słów, a także całkowitą liczbę znaków (w tym spacje między słowami) we fragmencie.

Aby wykonać to zadanie krok po kroku, użyjemy kilku funkcji z początku . Najpierw otwórz Przedmowę w MS Word, a następnie skopiuj drugi akapit zaczynający się od „Centralny” do schowka. Postępuj w następujący sposób w R:

> TextPar <- scan(file="clipboard", what="")

Aby uzyskać wektor zawierający każde ze słów jako oddzielny element.

> TextPar <- paste(TextPar,collapse = " ")

Aby przekonwertować TextPar na wektor zawierający jeden element składający się ze wszystkich słów połączonych i oddzielonych spacjami w pojedynczy ciąg znaków. Dodaj poprawne podziały wiersza ("\n") w TextPar, używając np. fix().

> TextPar <- strsplit(x = TextPar, split = '\n')

> mode(TextPar)

[1] "list"

> mode(unlist(TextPar))

[1] "character"

> TextPar <- unlist(TextPar)

Aby zmienić TextPar na wektor znakowy

> nchar(TextPar)

> length(TextPar)

Różniczkowanie i całkowanie

Różniczkowanie symboliczne

Zapoznaj się z plikami pomocy D() i derive().

Całkowanie

Zapoznaj się z plikiem pomocy integrate().