Programowanie w języku R. Typy danych

Aleksander Denisiuk
Uniwersytet Warmińsko-Mazurski
Olsztyn, ul. Słoneczna 54
denisjuk@matman.uwm.edu.pl

6 marca 2024

Typy danych

Najnowsza wersja tego dokumentu dostępna jest pod adresem

`http://wmii.uwm.edu.pl/~denisjuk/uwm`

Typy danych

Skale pomiarowe

- skala nominalna
  - brak uporządkowania, relacja: równość
- skala dychotomiczna
  - tylko dwie wartości, relacja: równość
- skala porządkowa
  - relacje: $=$, $<$, $>$, $\leqslant$, $\geqslant$
- skala interwałowa (przedziałowa)
  - różnice pomiędzy wartościami mają interpretację
- skala ilorazowa (stosunkowa)
  - różnice, ale także ilorazy wielkości mają interpretację
- skala absolutna
  - istnieje punkt zerowy, rośnie w jedną stronę

Dane interwałowe

- Dyskretne
- Ciągłe
- Testy parametryczne:
  - dane interwałowe, ciągłe
  - duża próba (ponad 30)
  - rozkład danych jest bliski do normalnego
- Testy nieparametryczne

Dane interwałowe w R

- Wektory numeryczne
- Jeden wektor $\longleftrightarrow$ jedna próba

```
x <- c(174, 162, 188, 192, 165, 168, 172.5)
str(x)
is.vector(x)
is.numeric(x)
```

- Dla obiektów unikać zajętych nazw

`c, T, F, NA, NaN, Inf, NULL, pi, ...`

Dane nominalne w R

- Wektor tekstowy
  - dane dychotomiczne: **TRUE** i **FALSE**

```
sex <-  c("male", "female", "male", "male",
   "female", "male", "male")
is.vector(sex)
is.character(sex)
str(sex)
table(sex)
plot(f)
```

Typ `factor`


```
sex.f <-  factor(sex)
plot(sex.f)
is.factor(sex.f)
is.character(sex.f)
str(sex.f)
```

Właściwości typu `factor`

- Podzbiór też jest typu `factor` o takiej samej ilości poziomów
  - dodatkowy argument `drop=TRUE`
  - konwersja typów

```
sex.f[5:6]
sex.f[6:7]
sex.f[6:7, drop=TRUE]
factor(as.character(sex.f[6:7]))
```

- Konwersja na liczby

```
as.numeric(sex.f)
```

Przykład

- Dane są wzrost `x` płeć `sex.f` oraz wagi pracowników `w`

```
w <- c(69, 68, 93, 87, 59, 82, 72)
```

- Wykres, na którym by się ukazały te wszystkie dane:

```
plot(x, w, pch=as.numeric(sex.f),
   col=as.numeric(sex.f))
legend("topleft", pch=1:2, col=1:2,
   legend=levels(sex.f))
```

  - kolory są zbędne

Dane porządkowe w R

- Kodowane liczbami
  - różnica liczb nie ma sensownej interpretacji
  - metody nieparametryczne

Uporządkowanie danych `factor`


```
m <- c("L", "S", "XL", "XXL", "S", "M", "L")
m.f <- factor(m)
m.f
```

- Zmiana uporządkowania

```
m.o <- ordered(m.f, levels=c("S", "M", "L",
      "XL", "XXL"))
m.o
```

Uwaga o danych `factor`

- Przy konwersji gubią się pierwotne wartości

```
a <- factor(3:5)
a
as.numeric(a)
```

- Rozwiązanie: poprzez tekst

```
as.numeric(as.character(a))
```

- Przy wczytywaniu tablic kolumna, w której jest nie liczba, zostanie przekonwertowana na `factor`
  - dodatkowy argument
    `read.table(..., as.is=TRUE)`

Dane wtórne

Częstość

- Częstość (częstość względna)
- Wuzualizacja
  - diagram słupkowy

```
rumianek.t <- read.table(
      "data/rumianek.txt",
      sep="\t")
rumianek <- rumianek.t$V2
names(rumianek) <- rumianek.t$V1
rumianek.plot <- barplot(rumianek,
      names.arg="")
text(rumianek.plot, par("usr")[3]-0.25,
      srt=45, adj=1, xpd=TRUE,
      labels=names(rumianek))
```

  - *dot plots*

```
dotchart(rumianek)
```

Ranga

- Numer kolejny obserwacji w próbie

```
a1 <- c(1,2,3,4,4,5,7,7,7,9,15,17)
a2 <- c(1,2,3,4,5,7,7,7,9,15,17)
names(a1) <- rank(a1)
a1
names(a2) <- rank(a2)
a2
```

  - mogą powstac problemy z jednakowymi wartościami

```
wilcox.test(a2)
```

Dane brakujące

- *unknown*
- *both*
- *not applicable*
  - ile godzin pracownicy śpią
    - *nie wiem*
    - *nie powiem*
    - na wagarach

```
h <- c(8, 10, NA, NA, 8, NA, 8)
mean(h)
```

- pominąć dane brakujące:

```
mean(h, na.rm=TRUE)
mean(na.omit(h))
```

Zastąpić dane brakujące

- Na średnią

```
h[is.na(h)] <- mean(h, na.rm=TRUE)
```

- Zrobić kopię zapasową

```
h.old <- h
```

- Pakiety `mice`, `cat`, `MissingDataGUI`

Obserwacja odstająca

- Literówka
  - minimum, maksimum dla ciągłych
  - częstości dla nominalnych
- Im więcej danych, tym lepiej

Przekształcenie danych

- Nominale $\rightarrow$ ciągłe
  - dodatkowa informacja
    - płeć $\rightarrow$ poziom testosteronu
  - często popełniany błąd: wprowadzenie uporządkowania

Przekształcenie danych ciągłych

- Logarytmiczne: `log(data + 1)`
  - asymetryczne prawe sprowadza do normalnego
  - linearyzuje zależności
  - wyrównuje wariancje
- Pierwiastkowe: `sqrt(data)`
  - podobne do logarytmicznego
- Odwrotne: `1/(data+1)`
  - stabilizuje wariancję
- Kwadratowe: `data^2`
  - asymetryczne lewe sprowadza do normalnego
  - linearyzuje zależności
  - wyrównuje wariancje

Przekształcenie logit

- Logit: `log(p/(1-p))`
  - stosowne do proporcji
- Arkus-sinusowe: `asin(sqrt(p))`

Dane wielowymiarowe

- Wszystkie w tych samych jednostkach
- Normalizacja (`scale()`):

```
a <- 1:10
b <- seq(100, 1000, 100)
d <- data.frame(a, b)
d
scale(d)
```

Macierze, Listy, Ramki

Macierz

- Wektor, który ma dodatkowe właściwości

```
m <- 1:4
m
ma <- matrix(m, ncol=2, byrow=TRUE)
ma
str(m)
str(ma)
mb <- m
mb
attr(mb, "dim") <- c(2,2)
mb
```

- Czemu macierz `mb` różni się od macierzy `ma`?
- Macierz trójwymiarowa:

```
m3 <- 1:8
dim(m3) <- c(2,2,2)
m3
```

Lista

- Składa się z elemetów różnych typów

```
l <- list("R", 1:3, TRUE, NA, list("r", 4))
l
```

- Wybór elementu:
  - wektor:

    ```
    h[3]
    ```
  - macierz:

    ```
    ma[2, 1]
    ```
  - lista:
    - wynik — lista

      ```
      l[1]
      str(l[1])
      ```
    - wynik — pierwszy element

      ```
      l[[1]]
      str(l[[1]])
      ```

Nazwy elementów

- Nazwy elementów listy

```
names(l) <- c("first", "second", "third",
      "fourth", "fifth")
l$first
str(l$first)
```

- Nazwy elementów wektora

```
names(w) <- c("Piotr", "Róża", "Czesław",
      "Łukasz", "Iwona", "Józef", "Filip")
w
w["Róża"]
```

- Nazwy elementów macierzy

```
rownames(ma) <- c("a1","a2")
colnames(ma) <- c("b1","b2")
ma
```

Ramki danych

- Lista wektorów tej samej długości

```
d <- data.frame(weight=w, height=x, size=m.o,
                sex=sex.f)
d
str(d)
```

- Wybór elementów jak w liście albo w macierzy:

```
d$weight
d[[1]]
d[,1]
d[,"weight"]
```

- Wybór kilka kolumn

```
d[,2:4]
d[,-1]
```

Wektory logiczne

- Wybrać tylko kobiety

```
d[d$sex=="female",]
```

  - ✦ wektor logiczny

```
d$sex=="female"
```

  - ✦ operacje logiczne: `&`, `|`, `!`

- Sortowanie

```
d[order(d$sex, d$height), ]
```

  - ✦ wektor indeksów

```
order(d$sex, d$height)
```

- jak posortować kolumny według alfabetu?
- jak posortować wiersze według alfabetu względem imion?

apply


```
M <- matrix(seq(1,16), 4, 4)
apply(M, 1, min)
[1] 1 2 3 4
apply(M, 2, max)
[1] 4 8 12 16
M <- array(seq(32), dim = c(4,4,2))
apply(M, 1, sum)
[1] 120 128 136 144
apply(M, c(1,2), sum)
[,1]  [,2]  [,3]  [,4]
[1,]  18  26  34  42
[2,]  20  28  36  44
[3,]  22  30  38  46
[4,]  24  32  40  48
```

lapply


```
x <- list(a = 1, b = 1:3, c = 10:100)
lapply(x, FUN = length)
$a
[1] 1
$b
[1] 3
$c
[1] 91
lapply(x, FUN = sum)
$a
[1] 1
$b
[1] 6
$c
[1] 5005
```

sapply


```
x <- list(a = 1, b = 1:3, c = 10:100)
sapply(x, FUN = length)
  a  b  c
  1  3 91
sapply(x, FUN = sum)
   a    b    c
   1    6 5005
```

tapply


```
x <- sample(1:4, size=50, replace=TRUE)
gr <- as.factor(sample(c("A","B","C","D"),
                       size=50, replace=T))
tapply(x, gr,sum)
  A  B  C  D
 25 35 41 27
```

- Oblicza sumę x dla każdego elementu grupy gr

Inne

- `aggregate`
- `replicate`
- `vapply`
- `mapply`
- `rapply`
- `by`
- `...`