Statystyka opisowa w środowisku R

Agnieszka Goroncy

UNIWERSYTET MIKOŁAJA KOPERNIKA W TORUNIU
Wydział Matematyki i Informatyki

Kategoryzacja danych: szereg rozdzielczy punktowy

Analizę danych często ułatwia ich skategoryzowanie poprzez stworzenie tzw. **szeregu rozdzielczego**.

Jeżeli dane mają charakter **dyskretny**, tzn. w obrębie tej samej zmiennej wartości obserwacji tworzą zbiór skończony, to rozsądnie jest stworzyć szereg rozdzielczy **punktowy**, np.:

liczba dzieci w rodzinie	0	1	2	3	4	5
liczba rodzin	1	15	81	244	388	271

W R służą do tego funkcje `table()` oraz `xtabs()`. Aby zademonstrować ich działanie, wczytamy plik `liczba_dzieci.txt` i skategoryzujemy dane w nim zawarte:

Kategoryzacja danych: szereg rozdzielczy punktowy

Analizę danych często ułatwia ich skategoryzowanie poprzez stworzenie tzw. **szeregu rozdzielczego**.

Jeżeli dane mają charakter **dyskretny**, tzn. w obrębie tej samej zmiennej wartości obserwacji tworzą zbiór skończony, to rozsądnie jest stworzyć szereg rozdzielczy **punktowy**, np.:

liczba dzieci w rodzinie	0	1	2	3	4	5
liczba rodzin	1	15	81	244	388	271

W R służą do tego funkcje `table()` oraz `xtabs()`. Aby zademonstrować ich działanie, wczytamy plik `liczba_dzieci.txt` i skategoryzujemy dane w nim zawarte:

```
> dane=read.table("liczba_dzieci.txt", header=TRUE)
> t1=table(dane)
> t2=xtabs(dane)
```

Kategoryzacja danych: szereg rozdzielczy punktowy

Analizę danych często ułatwia ich skategoryzowanie poprzez stworzenie tzw. **szeregu rozdzielczego**.

Jeżeli dane mają charakter **dyskretny**, tzn. w obrębie tej samej zmiennej wartości obserwacji tworzą zbiór skończony, to rozsądnie jest stworzyć szereg rozdzielczy **punktowy**, np.:

liczba dzieci w rodzinie	0	1	2	3	4	5
liczba rodzin	1	15	81	244	388	271

W R służą do tego funkcje `table()` oraz `xtabs()`. Aby zademonstrować ich działanie, wczytamy plik `liczba_dzieci.txt` i skategoryzujemy dane w nim zawarte:

```
> dane=read.table("liczba_dzieci.txt", header=TRUE)
> t1=table(dane)
> t2=xtabs(dane)
> names(t1)
```

Kategoryzacja danych: szereg rozdzielczy przedziałowy

Jeżeli dane mają charakter **ciągły**, tzn. wartości obserwacji w obrębie tej samej zmiennej raczej się nie powtarzają lub powtarzają się rzadko, wówczas konstrukcja szeregu przedziałowego punktowego nie ma sensu, bo liczba klas będzie w przybliżeniu taka sama jak liczba obserwacji. W takim przypadku rozsądnie jest skategoryzować wartości obserwacji w **szereg przedziałowy**, np.:

napięcie w sieci	(210, 213]	(213,216]	(216,219]	(219,222]	(222,225
liczba obserwacji	1	1	8	9	6

Szeregi przedziałowe zazwyczaj tworzymy dzieląc zakres zmiany zmiennej na przedziały równej długości. Liczbę przedziałów k można wybrać korzystając np. ze wzoru $k \simeq \sqrt{n}$, gdzie n oznacza liczbę obserwacji, lub samemu określając granice przedziałów. Funkcją przydatną w tej sytuacji jest `cut()`. Przykład:

```
> x = c(225, 223, 224, 220, 221, 218, 215, 219, 220, 221, 222, 220, 222,
+ 220, 219, 223, 224, 217, 218, 219, 216, 210, 218, 221, 225)
```

Kategoryzacja danych: szereg rozdzielczy przedziałowy

Jeżeli dane mają charakter **ciągły**, tzn. wartości obserwacji w obrębie tej samej zmiennej raczej się nie powtarzają lub powtarzają się rzadko, wówczas konstrukcja szeregu przedziałowego punktowego nie ma sensu, bo liczba klas będzie w przybliżeniu taka sama jak liczba obserwacji. W takim przypadku rozsądnie jest skategoryzować wartości obserwacji w **szereg przedziałowy**, np.:

napięcie w sieci	(210, 213]	(213,216]	(216,219]	(219,222]	(222,225
liczba obserwacji	1	1	8	9	6

Szeregi przedziałowe zazwyczaj tworzymy dzieląc zakres zmiany zmiennej na przedziały równej długości. Liczbę przedziałów k można wybrać korzystając np. ze wzoru $k \simeq \sqrt{n}$, gdzie n oznacza liczbę obserwacji, lub samemu określając granice przedziałów. Funkcją przydatną w tej sytuacji jest `cut()`. Przykład:

```
> x = c(225, 223, 224, 220, 221, 218, 215, 219, 220, 221, 222, 220, 222,
+ 220, 219, 223, 224, 217, 218, 219, 216, 210, 218, 221, 225)
> n = length(x)
> y1=cut(x, sqrt(n))
```

Kategoryzacja danych: szereg rozdzielczy przedziałowy

Jeżeli dane mają charakter **ciągły**, tzn. wartości obserwacji w obrębie tej samej zmiennej raczej się nie powtarzają lub powtarzają się rzadko, wówczas konstrukcja szeregu przedziałowego punktowego nie ma sensu, bo liczba klas będzie w przybliżeniu taka sama jak liczba obserwacji. W takim przypadku rozsądnie jest skategoryzować wartości obserwacji w **szereg przedziałowy**, np.:

napięcie w sieci	(210, 213]	(213,216]	(216,219]	(219,222]	(222,225
liczba obserwacji	1	1	8	9	6

Szeregi przedziałowe zazwyczaj tworzymy dzieląc zakres zmiany zmiennej na przedziały równej długości. Liczbę przedziałów k można wybrać korzystając np. ze wzoru $k \simeq \sqrt{n}$, gdzie n oznacza liczbę obserwacji, lub samemu określając granice przedziałów. Funkcją przydatną w tej sytuacji jest `cut()`. Przykład:

```
> x = c(225, 223, 224, 220, 221, 218, 215, 219, 220, 221, 222, 220, 222,
+ 220, 219, 223, 224, 217, 218, 219, 216, 210, 218, 221, 225)
> n = length(x)
> y1=cut(x, sqrt(n))
> table(y1)
```

Kategoryzacja danych: szereg rozdzielczy przedziałowy

Jeżeli dane mają charakter **ciągły**, tzn. wartości obserwacji w obrębie tej samej zmiennej raczej się nie powtarzają lub powtarzają się rzadko, wówczas konstrukcja szeregu przedziałowego punktowego nie ma sensu, bo liczba klas będzie w przybliżeniu taka sama jak liczba obserwacji. W takim przypadku rozsądnie jest skategoryzować wartości obserwacji w **szereg przedziałowy**, np.:

napięcie w sieci	(210, 213]	(213,216]	(216,219]	(219,222]	(222,225
liczba obserwacji	1	1	8	9	6

Szeregi przedziałowe zazwyczaj tworzymy dzieląc zakres zmiany zmiennej na przedziały równej długości. Liczbę przedziałów k można wybrać korzystając np. ze wzoru $k \simeq \sqrt{n}$, gdzie n oznacza liczbę obserwacji, lub samemu określając granice przedziałów. Funkcją przydatną w tej sytuacji jest `cut()`. Przykład:

```
> x = c(225, 223, 224, 220, 221, 218, 215, 219, 220, 221, 222, 220, 222,
+ 220, 219, 223, 224, 217, 218, 219, 216, 210, 218, 221, 225)
> n = length(x)
> y1=cut(x, sqrt(n))
> table(y1)
> y2=cut(x,breaks=c(205,215,225))
```

Kategoryzacja danych: szereg rozdzielczy przedziałowy

Jeżeli dane mają charakter **ciągły**, tzn. wartości obserwacji w obrębie tej samej zmiennej raczej się nie powtarzają lub powtarzają się rzadko, wówczas konstrukcja szeregu przedziałowego punktowego nie ma sensu, bo liczba klas będzie w przybliżeniu taka sama jak liczba obserwacji. W takim przypadku rozsądnie jest skategoryzować wartości obserwacji w **szereg przedziałowy**, np.:

napięcie w sieci	(210, 213]	(213,216]	(216,219]	(219,222]	(222,225
liczba obserwacji	1	1	8	9	6

Szeregi przedziałowe zazwyczaj tworzymy dzieląc zakres zmiany zmiennej na przedziały równej długości. Liczbę przedziałów k można wybrać korzystając np. ze wzoru $k \simeq \sqrt{n}$, gdzie n oznacza liczbę obserwacji, lub samemu określając granice przedziałów. Funkcją przydatną w tej sytuacji jest `cut()`. Przykład:

```
> x = c(225, 223, 224, 220, 221, 218, 215, 219, 220, 221, 222, 220, 222,
+ 220, 219, 223, 224, 217, 218, 219, 216, 210, 218, 221, 225)
> n = length(x)
> y1=cut(x, sqrt(n))
> table(y1)
> y2=cut(x,breaks=c(205,215,225))
> levels(y2)=c("niskie", "wysokie")
```

Kategoryzacja danych: szereg rozdzielczy przedziałowy

Jeżeli dane mają charakter **ciągły**, tzn. wartości obserwacji w obrębie tej samej zmiennej raczej się nie powtarzają lub powtarzają się rzadko, wówczas konstrukcja szeregu przedziałowego punktowego nie ma sensu, bo liczba klas będzie w przybliżeniu taka sama jak liczba obserwacji. W takim przypadku rozsądnie jest skategoryzować wartości obserwacji w **szereg przedziałowy**, np.:

napięcie w sieci	(210, 213]	(213,216]	(216,219]	(219,222]	(222,225
liczba obserwacji	1	1	8	9	6

Szeregi przedziałowe zazwyczaj tworzymy dzieląc zakres zmiany zmiennej na przedziały równej długości. Liczbę przedziałów k można wybrać korzystając np. ze wzoru $k \simeq \sqrt{n}$, gdzie n oznacza liczbę obserwacji, lub samemu określając granice przedziałów. Funkcją przydatną w tej sytuacji jest `cut()`. Przykład:

```
> x = c(225, 223, 224, 220, 221, 218, 215, 219, 220, 221, 222, 220, 222,
+ 220, 219, 223, 224, 217, 218, 219, 216, 210, 218, 221, 225)
> n = length(x)
> y1=cut(x, sqrt(n))
> table(y1)
> y2=cut(x,breaks=c(205,215,225))
> levels(y2)=c("niskie", "wysokie")
> table(y2)
```

Tabele krzyżowe

W R można również tworzyć **tabele krzyżowe** (kontyngencji), które **pozwalają przedstawić rozkład dwóch bądź więcej zmiennych losowych**. Z reguły prezentowane są one w postaci macierzowej, np.

ID	Płeć	Typ programu
1	K	Kabarety
2	M	Kabarety
3	M	Teleturnieje
4	K	Teleturnieje
5	M	Kabarety
6	M	Teleturnieje
7	K	Teleturnieje
8	K	Teleturnieje
9	K	Teleturnieje
10	M	Kabarety

$\rightsquigarrow$

	Typ programu		
Płeć	Teleturnieje	Kabarety	**Razem**
Mężczyźni	2	3	5
Kobiety	4	1	5
Razem	6	4	10

Tabele krzyżowe: przykład

Tworzenie tabel krzyżowych umożliwia funkcja **table()**.

Przykład: Wykorzystamy bazę `MathAchieve` dostępną w pakiecie `nlme`, która zawiera wyniki osiągnięć matematycznych. Interesuje nas rozkład płci względem przynależności do mniejszości rasowych (zmienne `Sex, Minority`).

Tabele krzyżowe: przykład

Tworzenie tabel krzyżowych umożliwia funkcja **table()**.

Przykład: Wykorzystamy bazę `MathAchieve` dostępną w pakiecie `nlme`, która zawiera wyniki osiągnięć matematycznych. Interesuje nas rozkład płci względem przynależności do mniejszości rasowych (zmienne `Sex`, `Minority`).

```
> library("nlme")
> table(MathAchieve$Sex, MathAchieve$Minority)
          No     Yes
  Male    2481   909
  Female  2730   1065
```

Z uzyskanej tablicy kontyngencji można odczytać np., że 909 spośród ankietowanych mężczyzn należy do mniejszości rasowych.

Analizowana baza danych

Na potrzeby omówienia statystyk opisowych i pracy z nimi, **będziemy analizować słynną bazę danych Edgara Andersona**, który badał trzy gatunki irysów na Półwyspie Gaspé. Baza wygląda następująco:

```
> iris
```

Zmienne się w niej znajdujące możemy uzyskać po wywołaniu polecenia:

```
> names(iris)
```

Na początek przeanalizujmy dane dotyczące długości płatków kwiatów (zmienna `Petal.Length`):

```
> petal<-iris$Petal.Length
```

Funkcja length() pozwala wyznaczyć wielkość analizowanej próby:

```
> length(petal)
```

Są to dane typu numerycznego, zatem łatwo możemy uzyskać z nich liczbowe statystyki opisowe.

Przydatne funkcje

Funkcje, które pozwalają wyznaczać najpopularniejsze statystyki opisowe są następujące:

> `mean(petal)` - średnia arytmetyczna; dodatkowy parametr `trim`$\in (0, 0.5)$ pozwala na obcięcie po trim*100% skrajnych obserwacji z każdego końca próbki, przed obliczeniem średniej (przydatne, gdy występują obserwacje odstające),

> `var(petal)` - wariancja próbkowa (wersja nieobciążona: $var = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$),

> `sd(petal)` - odchylenie standardowe (pierwiastek z wariancji próbkowej),

> `min(petal)` - minimum z próby,

> `max(petal)` - maksimum z próby,

> `range(petal)` - przedział zmienności próby: (min, max),

Kwantyle próbkowe w R

Funkcje, które pozwalają liczyć statystyki opisowe związane z kwantylami empirycznymi to:

> `median()` - mediana próbkowa,

> `quantile()` - kwantyle próbkowe, ich rzędy podawane są w wektorze będącym drugim argumentem funkcji,

> `IQR()` - rozstęp międzykwartylowy.

Przykład:

```
> quantile(petal,c(0.1,0.6,0.9))
10% 60% 90%
1.40 4.64 5.80
```

Kwantyle próbkowe w R

Funkcje, które pozwalają liczyć statystyki opisowe związane z kwantylami empirycznymi to:

> `median()` - mediana próbkowa,

> `quantile()` - kwantyle próbkowe, ich rzędy podawane są w wektorze będącym drugim argumentem funkcji,

> `IQR()` - rozstęp międzykwartylowy.

Przykład:

```
> quantile(petal,c(0.1,0.6,0.9))
10% 60% 90%
1.40 4.64 5.80
```

Oznacza to np. że 10% płatków irysów ma długość poniżej 1.40, z kolei 40% płatków irysów ma długość powyżej 4.64 (bo 60% płatków ma długość poniżej tej wielkości), zaś tylko 10% płatków ma długość większą niż 5.80.

Dominanta (moda)

W pakietach, które standardowo są załadowane do `R`, nie ma funkcji, która umożliwiałaby policzenie **dominanty** (wartości modalnej), czyli takiej wielkości, która w zbiorze danych występuje **najczęściej**. Aby policzyć najliczniej reprezentowaną kategorię w bazie danych, można zrobić to w następujący sposób:

- skategoryzować dane w szereg rozdzielczy,
- uporządkować szereg rozdzielczy względem liczebności kategorii, w kolejności od najbardziej licznej do najmniej licznej,
- zwrócić nazwę kategorii, która znajduje się na pierwszym miejscu.

Przykład: Posłużymy się danymi z pliku `liczba_dzieci.txt`:

Dominanta (moda)

W pakietach, które standardowo są załadowane do R, nie ma funkcji, która umożliwiałaby policzenie **dominanty** (wartości modalnej), czyli takiej wielkości, która w zbiorze danych występuje **najczęściej**. Aby policzyć najliczniej reprezentowaną kategorię w bazie danych, można zrobić to w następujący sposób:

- skategoryzować dane w szereg rozdzielczy,
- uporządkować szereg rozdzielczy względem liczebności kategorii, w kolejności od najbardziej licznej do najmniej licznej,
- zwrócić nazwę kategorii, która znajduje się na pierwszym miejscu.

Przykład: Posłużymy się danymi z pliku `liczba_dzieci.txt`:

```
> dane=read.table("liczba_dzieci.txt")
> tablica<-table(dane)
> tablica_s<-names(sort(tablica,decreasing=T))
> dominanta = tablica_s[1]
> dominanta
```

Funkcja summary()

Funkcja summary() umożliwia podsumowanie charakterystyk obiektu, który jest argumentem funkcji. Może ona dać inny wynik dla zmiennej numerycznej, inny dla jakościowej, a inny np. dla ramki danych.

Jeżeli argumentem funkcji jest wektor **danych liczbowych**, wówczas funkcja zwraca podstawowe statystyki opisowe dotyczące położenia:
minimum, dolny kwartyl, medianę, górny kwartyl, maksimum.

Przykład: Wykorzystamy ponownie zmienną `Petal.Length` z bazy danych `iris`:

```
> summary(petal)
Min. 1st Qu. Median Mean 3rd Qu. Max.
 1.000 1.600 4.350 3.758 5.100 6.900
```

Funkcja summary(), c.d.

Jeżeli argumentem funkcji jest wektor **danych jakościowych**, wówczas funkcja zwraca liczebności poszczególnych kategorii zmiennej, np.

```
> summary(iris$Species)
setosa versicolor virginica
    50         50        50
```

W tym przypadku rezultat działania funkcji `summary()` jest analogiczny jak funkcji `table()`.

Jeżeli argumentem funkcji jest **ramka danych**, wówczas funkcja zwraca podsumowania każdej zmiennej z ramki, np.

```
> summary(iris)
```

Momenty próbkowe w R

Dla obliczenia momentów empirycznych służy pakiet *moments*. Zawiera on też inne przydatne statystyki, m.in.

- **moment(x, order = 1, central = FALSE, absolute = FALSE, na.rm = FALSE)** - umożliwia obliczenie momentów zwykłych (domyślnie), centralnych (jeżeli `central=TRUE`), absolutnych (jeżeli `absolute=TRUE`), oraz centralnych absolutnych (jeżeli oba parametry jednocześnie: `absolute` oraz `central` ustawione są na `TRUE`), ustalonego rzędu (parametr `order`)
- **skewness()** - umożliwia policzenie współczynnika skośności rozkładu wg wzoru $sk = \frac{m_3}{(m_2)^{3/2}} = \frac{m_3}{s^3}\left(\frac{n}{n-1}\right)^{3/2}$,
- **kurtosis()** - umożliwia policzenie kurtozy rozkładu wg wzoru $K = \frac{m_4}{(m_2)^2} = \frac{m_4}{s^4}\left(\frac{n}{n-1}\right)^2$.

Momenty próbkowe w R: przykład

Przykład: Oblicz trzeci absolutny moment centralny, kurtozę i skośność dla zmiennej `Petal.Length` z bazy danych `iris`.

Najpierw należy zainstalować pakiet *moments*:

```
> install.packages("moments")
```

Dalej załadować go:

```
> library(moments)
```

I teraz:

```
> moment(petal,order=3,central=T,absolute=T)
6.731268
```

Momenty próbkowe w R: przykład

Przykład: Oblicz trzeci absolutny moment centralny, kurtozę i skośność dla zmiennej `Petal.Length` z bazy danych `iris`.

Najpierw należy zainstalować pakiet *moments*:

```
> install.packages("moments")
```

Dalej załadować go:

```
> library(moments)
```

I teraz:

```
> moment(petal,order=3,central=T,absolute=T)
6.731268
> kurtosis(petal)
1.604464
```

Momenty próbkowe w R: przykład

Przykład: Oblicz trzeci absolutny moment centralny, kurtozę i skośność dla zmiennej `Petal.Length` z bazy danych `iris`.

Najpierw należy zainstalować pakiet *moments*:

```
> install.packages("moments")
```

Dalej załadować go:

```
> library(moments)
```

I teraz:

```
> moment(petal,order=3,central=T,absolute=T)
6.731268
> kurtosis(petal)
1.604464
> skewness(petal)
-0.2721277
```

Momenty próbkowe w R: przykład

Przykład: Oblicz trzeci absolutny moment centralny, kurtozę i skośność dla zmiennej `Petal.Length` z bazy danych `iris`.

Najpierw należy zainstalować pakiet *moments*:

```
> install.packages("moments")
```

Dalej załadować go:

```
> library(moments)
```

I teraz:

```
> moment(petal,order=3,central=T,absolute=T)
6.731268
> kurtosis(petal)
1.604464
> skewness(petal)
-0.2721277
```

Uwaga: Kurtoza liczona wg powyższego wzoru dla danych z rozkładu normalnego wynosi ok. 3.

Podsumowanie analizy

Funkcja `summary()` podsumowująca długość płatków irysów dała następujący rezultat:

```
Min. 1st Qu. Median Mean 3rd Qu. Max.
 1.000 1.600 4.350 3.758 5.100 6.900
```

Jak widać, średnia jest mniejsza od mediany, zatem więcej obserwacji znajduje się po prawej stronie od wartości przeciętnej (ponad połowa płatków ma długość większą od średniej).

Czyżby rozkład empiryczny miał cechy lewostronnej asymetrii? Współczynnik skośności wynosi `-0.2721277` i jest ujemny, co rzeczywiście oznacza lewostronną asymetrię.

Kurtoza (koncentracja rozkładu empirycznego) wynosi `1.604464` i jest mniejsza niż 3, co oznacza, że dane są mniej skoncentrowane niż w przypadku wzorcowego rozkładu normalnego.

Uwaga: do kompletnej analizy opisowej danych potrzeba jeszcze analizy wykresów statystycznych, w szczególności histogramu oraz wykresu pudełkowego.

Literatura

- Łukasz Komsta, **Dokumentacja pakietu 'moments'**
- Robert I. Kabacoff, **Quick R - statystyki opisowe**
- Phil Spector, **Data Manipulation with** R, Use R!, Springer, 2008
- Przemysław Biecek, **Przewodnik po pakiecie** R, Oficyna Wydawnicza GiS, Wrocław, 2011
- Łukasz Komsta, **Wprowadzenie do środowiska** R
- Joseph Adler, **R in a Nutshell**, O'Reilly Media, 2009