

Преговор и въведение в R

Натали Георгиева

Седмица 1

1 Данни

Данни наричаме колекция от факти като числа, думи, измервания, наблюдения и даже описание на предмети. Множеството на изследване наричаме **генерална съвкупност**, например всички жители на страната. Категоризацията на данни става според техния вид:

- Числови (размер, цена, тегло ...) и нечислови данни (цвят, форма ...).
- Времеви редове - наблюдение на процес или явление във времето.
- Скали на измерване. Категориите могат да бъдат ненаредени, напримет цвят или наредени - оценки - слаб, среден ...
- Сувори и агрегирани данни. Агрегираните данни се правят на базата на сурови данни посредством таблици и групиране, числови характеристики, графики.

Аналитичната обработка на такъв тип данни и получаване на заключение е задача на математическата статистика.

2 R – среда за програмиране и език.

R е език и среда за статистически изчисления и графика. Той е разработен в Bell Laboratories, като GNU проект на езика C. Езикът R е криптов език за пресмятания в специфична среда, предимно на C/C++ и Фортран. По-късно са добавени функционалности на R. Първоначално R е бил предназначен основно за Unix, но в момента няма ограничение за операционната система.

Използването на R може да става чрез:

- Скриптове с файлове с разширение R.
- Използване на IDE - R Gui, R Studio, Eclipse, Visual Studio.
- Вграждане в програми от други езици, като C/C++ и Java.
- ООП с класове - обекти имащи атрибути и методи.

Средата R се инсталира с минимална функционалност налична в пакета base. Останалите статистически функционалности и примерни данни са разделени в библиотеки. Те са достъпни чрез mirror locations - CRANE(Comprehensive R Archive Network). Тяхното първоначално инсталиране става със специална команда или през специално меню в IDE. След това инсталираните библиотеки могат да се ползват, чрез зареждане през менюто на студио или команда. По същият начин може да бъде използван help за всяка функция и библиотека.

За програмиране и изпълнение на програмен код през shell се използва разширение на файлове '*.R'. Символ за отделяне на коментари е '#'.

3 Базови типове данни.

3.1 Вектор и скалар.

Основното предназначение на R е за статистическа обработка на данни. Затова структурите от данни са оптимизирани за работа с вектори. Инициализирането на вектор x за структура от данни или математическа функция става посредством функцията $x = c(x_1, x_2, \dots)$ и оператор за присвояване '='.

Операции за достъп до елементи:

- Скалар чрез индекс: $x[i]; i \geq 1$.
- Подмножество от i до j чрез индекс: $x[i : j]; i, j \geq 1$.
- Подмножество чрез изключване: $x[-i]; x[-(i : j)]$.
- Конкатенация на вектори: $x = c(x_1, x_2)$.
- Чрез операции за сравняване $>, <, ==, \&, |, \dots$: $x[x > 1]$.
- Достъп до индекси: $\text{which}(x == 3)$.
- Автоматизирано създаване на редица: $\text{seq}()$; $\text{rep}()$.
- Размер: $\text{length}()$.
- Операции със скалар/вектор: $+, -, *, /$.
- Основни функции за манипулации на вектори: $\text{sum}()$; $\text{mean}()$; $\text{sort}()$; $\text{min}()$; $\text{max}()$.

Основните типове променливи са числени (integer, numeric, double, complex), логически (TRUE/ FALSE/NULL), чар (character), категоризащи (factor). Всеки тип има нулев pointer NULL (включително логическите!! за разлика от C++). За разлика от C++. Съществува стойност за липсващи данни - NA и невъзможни числени резултати, като деление на 0 - NaN.

Проверките за типа на променливите става с помощта на функциите 'is.'. Дефинирането на тип на променливи с 'as.'.

- Проверка за нулев pointer става с помощта на функцията `is.null(x)`.
- Проверка за нулева стойност става с помощта на функцията `is.na(x)`.
- Проверка за тип - `is.numeric(x)`, `is.integer(x)`, `is.character(x)`...
- Дефиниране на тип - `as.numeric(x)`, `as.integer(x)`, `as.character(x)`...

3.2 Матрици

Матрици в R се създават чрез функцията `matrix(data, nrow, ncol, byrow)`. Например:

$$A = \text{matrix}(c(2, 4, 3, 1, 5, 7), nrow = 2, ncol = 3, byrow = TRUE)$$

Съществува възможност за начална инициализация чрез примитивната функция `as.matrix(data)` - създава по подразбиране обект от тип `matrix(ncol=1)`. Проверката за това дали даден обект е матрица става с функцията `is.matrix(M)`.

Операции за достъп до елементи:

- Скалар чрез индекс: `M[i,j]`; $i, j \geq 1$.
- Колона или стълб: `M[i,]`; `x[,i]`; $i \geq 1$.
- Селектиране на определени стълбове и колони: `M[,c(i,j)]`; $i, j \geq 1$.
- Добавяне на колони и редове - `cbind(M,col)`, `rbind(M,row)`.
- Размерност - `dim(M)`, `nrow(M)`, `ncol()`.
- `dimnames()` - функция и параметър в `matrix()` за поставяне на имена на колоните и стълбовете. Например, за създадената по-горе матрица A и поставянето едновременно на имена на редовете и колоните става по следния начин:

$$\text{dimnames}(A) = \text{list}(c(\text{row1}, \text{row2}), c(\text{col1}, \text{col2}, \text{col3})).$$

- Достъп до колони и редове с име - `x[row_name, col_name]`.
- Операции със скалар/вектор/матрица: `+`, `-`, `*`, `/`, `^`.
- Обръщане и транспониране - `solve(M)`, `t(M)`.
- Достъп до главния диагонал на матрицата: `diag(M)`.

4 Входни и изходни данни.

4.1 Входни данни.

Данните могат да съществуват под различни форми:

- Библиотеки в R. Зареждане на `dataset(s)` с функцията `data()`.
- Файлове на локална или отдалечена локация с разширения `.dat`, `txt`, `.csv`, `.xls`, `xlsx`,
- бази данни.
- Кеширани данни от средата R - `R.Data()`. Зареждане с функцията `load()`.

Тяхното зареждане в R става без ограничение, с единственото условие, че много от функционалностите са изнесени в специални библиотеки. Основните операции за четене от файлове е базирана на съответните C функции. Подобна функция е `scan()`. Когато трябва да бъде прочетен файл с подредени в таблица данни се използва `read.table()`. Подобни функции за четене са `read.csv()` и `read.csv2()` за четене на csv файлове.

4.2 Съхранение на данни.

Съхранение на файлове:

- `RData` - `save()`.
- Стандартната C функционалност: `file_handler = file(file_name, wa)`
добавяне на ред: `cat(list_of_elements, ..., file_handler)`
затваряне на връзката: `close(file_handler)`
конкатенация на файлове: `file.append()`.
- `write.table()`; `write.csv2()`.
- бази данни... .

4.3 Графично представяне.

За визуалното представяне на резултатите в R съществува богата функционалност. Основните видове графики в основния пакет са:

- `plot()` - Графика за съпоставяне последователно период на измерване и измерена стойност.
- `barplot()` - Графика за представяне на категоризирани данни с правоъгълни `bars` с височини и широчини пропорционална на стойностите, които те представят. Бар-графиките може да бъдат вертикални или хоризонтални.
- `pie()` - Сумаризирана графика за представяне на категоризирани данни представящи стойности на дадена променлива в процентно/дялово разпределение. Обобщените данни трябва да бъдат предварително подготвени таблично.

- `histogram()` - графически метод за представяне на брой измервания попадащи в определени интервали от стойности. Тези интервали се наричат класове или bins. Честота, с която данните попадат във всеки клас се описва с бар графика. Използва се за графическо обобщение на разпределението на данните.
- `boxplot()` - медиана, квантили, outliers.

5 Текст.

Основният тип (class of object) на текстови данни в R е `character`. За изписване на стринг от `character` се използват единични или двойни кавички:

`'a character string using single quotes'`

`"a character string using double quotes"`

Позволено е използването на единични кавички в стринг означен с двойни кавички или стринг от двойни кавички в стринг с единични:

`"The 'R' project for statistical computing"`

`'The "R"project for statistical computing'`

Но използването на подстринг с двойни кавички в стринг с двойни или такъв с единични в стринг с единични е грешно: `"This "is"totally unacceptable"`

`'This 'is' absolutely wrong'`

Функцията за създаване на вектор от стрингове е `character()`. За основните операции за работа със стрингове се използват следните 'C'-style функции:

- `paste(..., sep = , collapse = NULL)` - Функцията взима един или няколко обекта, превръща ги в `"character"` и ги конкатенира за получаването на един или няколко стринга:
`PI = paste("The value of pi is pi")`
`IR = paste("Just "Do "It sep = ". ")`
- `print(my_string)` - Принтиране на текст.
- `cat()` - Конкатенира обекти и ги записва или на екран или на файл.
- `format()` - Форматиране на R object, като ги интерпретира за стрингове: `format(pi,digits = 2)`.
- `sprintf()` - C-style format.
- `nchar()` - Брой символи в стринг.
- `tolower()`, `toupper()` - Конвертира от главни към малки букви и обратно.
- `substring()`, `substr()` - Връща субстринг:
`substr("abcdef", 2, 4)`
`substring("abcdef", 2, 4)`.
- `sort()` - Сортиране.

- `intersect()`, `setdiff()` - Съвпадения или разлики:
`set1 = ("some "random "few "words")`
`set2 = ("some "many "none "few")`
`intersect (set1, set2)`
`setdiff(set1, set2)`

6 Цикли и проверки.

Логическите проверки в R не се различават от останалите езици. За целта се използват операторите `if()` и `else()`, с единствено условие за проверка за наличие на NULL логическо състояние.

Масовото използване на цикли не е удачно в R поради неефективност. Но когато това се налага синтаксисът е следния:

```
for(i in 1:n)
{ ... Body ... }
while (true)
{ ... Body ...}
```

7 Функции.

Основната причина за създаването на собствени функции в R е дефинирането на функционалност за многократно използване и/или структуриране на кода. Синтаксисът е сравнително прост:

```
func_name<-function(input_params,...)
{...
BODY
....
return (ret)}
```

Важно е да се запомни, че локално дефинираната променлива предефинира глобалната такава само в тялото на функцията:

```
x=5
foo=function(a)
{
x=4
return (a+x)
}
foo(2)
x
```

8 Операции върху таблици.

За ефективен достъп и обработка на структурирани данни в R се използва специален клас функции. Те позволяват достъп и изпълнение на предварително дефинирана функция до елементите на вектор, list или data.frame. Такива функции са:

- `apply(array, margin, function, ...)`: Връща вектор от резултата на function върху margin.
`mat <- matrix(rep(seq(4), 4), ncol=4)`
`apply(mat, 1, sum)#`
редове `apply(mat, 2, sum)`
`apply (mat1, 1, function(x) sum (x) + 2).`
- `lapply(list, function, ...)`: Прилага function върху вектор или лист. Връща list. Особено полезна за работа с data.frame.
`mat.df <- lapply(mat.df, sum)`
`y <-lapply (mat.df, function(x, y) sum(x) + y, y = 5)`
`lapply(1:5, function(i) print(i)) # loop.`
- `sapply(list, function, ..., simplify)`: Прилага function върху вектор, матрица или лист. Връща вектор, data.frame, list. Когато `simplify=F` връща резултат като lapply.
`y2 <-sapply(mat1.df, function (x, y) sum(x) + y, y = 5).`
- `tapply(array, indices, function, ..., simplify)`: Прилага function върху елемент от array според групирането от indices. Връща вектор или матрица с размер според indices. Когато `simplify=F` връща list.
`x1 <- runif(16)`
`cat1 <- rep(1:4, 4)`
`cat2 <- c(rep(1, 8), rep(2, 8))`
`mat2.df <- data.frame(x1)`
`mat2.d$cat1 <- cat1`
`mat2.df$cat2 <- cat2`
`tapply(mat2.df$x1, mat2.d$cat1, mean).`

9 Упражнение.

Задача 1. Какво прави следната функция:

`v-matrix (c (a, b, c, 2, 2, 3), TRUE)`

Задача 2. Има две матрици: `mat1<-matrix(1:6,2)`

`mat2<-matrix(c(rep(1,3),rep (2,3), 2,byrow=F)`

Какво извеждат следните операции:

`solve(mat1[, 2:3])`

`mat1*mat2`

`mat1%*%t(mat 2).`

Задача 3. Нека е даден вектор $x=(8,3,8,7,15,9,12,4,9,10,5,1)$. Да се:

- създадете матрица 6 реда и 2 колони.
- да се поставят имена на колоните "c1" "c2". намерете максимум и минимум.
- добавете нов нов ред от случайни числа между 1:20 в края на матрицата. Използвайте функцията `rndi()`.
- умножете втората колона с 2 и след това всеки елемент съберете с 6.