Statistical methods for spatial data analysis

Introduction to R

Joaquin Cavieres

1. Introduction to `R`

Notation

- The matrices are noted generally with bold font and capital letter : $\boldsymbol{A}$, $\boldsymbol{B}$, etc.
- The column vectors are denoted with bold font but with lower case: $\boldsymbol{a}$, $\boldsymbol{b}$, etc.
- The elements of a vector $\boldsymbol{x}$ are denoted as x_1, $x_2, ..., x_m$, etc.
- The elements of a matrix $\boldsymbol{X}$ are denoted as x_{11}, $x_{12}, ..., x_{1m}$, or x_{11}, $x_{21}, ..., x_{mn}$, etc.

Vectors

- A vector $\boldsymbol{x}$ of size m is a column of m elements (numbers):

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

- The numbers x_i are elements of the vector $\boldsymbol{x}$
- The transpose of the vector $\boldsymbol{x}$ can be expressed as $\boldsymbol{x}^T = (x_1, ..., x_n)$ (a row vector).
- The addition and subtraction of vectors of equals dimension is done element by element:

$$\boldsymbol{x} + \boldsymbol{y} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_m + y_m \end{bmatrix}$$

It is not possible add or subtract vectors with different dimension!.

- The scalar multiplication of a vector is element by element:

$$\lambda \boldsymbol{x} = \begin{bmatrix} \lambda x_1 \\ \lambda x_2 \\ \vdots \\ \lambda x_m \end{bmatrix}$$

for any scalar λ (a real number).

- Two vectors $\boldsymbol{x}$ e $\boldsymbol{y}$ are equals if they have the same dimension and each pair of elements are equals, for example $x_i = y_i$ for $i = 1, ..., n$.
- - The scalar product (inner product) is $\langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{n} x_i y_i$ (commonly written so by the mathematicians). The statisticians denote this operation as $\boldsymbol{x}^T \boldsymbol{y}$.

Example in `R`: We have the vector $\boldsymbol{x} = (1,2,3)^T$ and a vector $\boldsymbol{y} = (4,5,6)^T$. The scalar product is:

$1 * 4 + 2 * 5 + 3 * 6 = 32$.

Building vectors in `R`

The easy way to create a vector is using the function `c(x_1, x_2,..., x_n)`, that is:

```
x = c(1,2,3)
x
```

```
## [1] 1 2 3
```

```
y = c(4,5,6)
y
```

```
## [1] 4 5 6
```

`x` y `y` will be interpreted as a column vector or a row vector depending on the context. We can avoid any ambiguity or error in the declaration of the vector in `R` using the function `matrix (.,.,.,.)`, that is:

```
c = matrix(c(6,5,4), nrow=3, ncol=1,byrow=T)
d = matrix(c(3,2,1),3,1, byrow=T)
```

```
c
```

```
##      [,1]
## [1,]    6
## [2,]    5
## [3,]    4
```

```
d
```

```
##      [,1]
## [1,]    3
## [2,]    2
## [3,]    1
```

If we do not use the function `matrix (.,.,.)` then the vector have a `numeric()` class (not a matrix expressed as a column vector). Anyway, we can convert the numeric vector to a matrix as:

```
b = c(4,5,6)
b
```

```
## [1] 4 5 6
```

```
class(b)
```

```
## [1] "numeric"
```

```
b = as.matrix(b)
b
```

```
##      [,1]
## [1,]    4
## [2,]    5
## [3,]    6
```

```
class(b)
```

```
## [1] "matrix" "array"
```

It should be noted that the arguments `nrow()` and `ncol()` are assumed by default, but we can change the order of the elements in the matrix:

```
u = matrix(c( 3, 2, 1), 1, 3, byrow=T)
u
```

```
##      [,1] [,2] [,3]
## [1,]    3    2    1
```

```
v = matrix(c(6,5,4), ncol=1, nrow=3, byrow=T)
v
```

```
##      [,1]
## [1,]    6
## [2,]    5
## [3,]    4
```

Additional

Other ways to create vectors in R:

```
x = 1:5
x
```

```
## [1] 1 2 3 4 5
```

```
x = seq(1,5, by =0.25)
x
```

```
##  [1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50
## [16] 4.75 5.00
```

```
x = seq(10,1, by =-1)
x
```

```
##  [1] 10  9  8  7  6  5  4  3  2  1
```

```
x = rep(1, times = 5)
x
```

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

```
length(x)
```

```
## [1] 5
```

```
rep(x, 2)
```

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

```
set.seed(1)                          # "seed"
x = sample(1:10, 5)
x
```

```
## [1] 9 4 7 1 2
```

Operations with vectors:

```
u = 1:5                             # To create the vector
v = rep(0.5, times = length(u))
u + v
```

```
## [1] 1.5 2.5 3.5 4.5 5.5
```

```
u * v
```

```
## [1] 0.5 1.0 1.5 2.0 2.5
```

```
u^v
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

Funcionts applied to vectors

```
x = 1:10
sqrt(x)            # square root
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
##  [9] 3.000000 3.162278
```

```
sum(x)             # sum
```

```
## [1] 55
```

```
prod(x)            # product
```

```
## [1] 3628800
```

```r
mean(x)            # mean of the vector
```

```
## [1] 5.5
```

```r
var(x)             # variance of the vector
```

```
## [1] 9.166667
```

Access to elements of a vector

```r
x=c(3, 2, 1, 7, 5)           # to create a vector
x[4]                         # access to element 4
```

```
## [1] 7
```

```r
x[2];x[5]                    # access to element 2 and 5
```

```
## [1] 2
```

```
## [1] 5
```

```r
x[1:3]; x[c(2,4,5)]          # access to sub-vectors
```

```
## [1] 3 2 1
```

```
## [1] 2 7 5
```

```r
x[4]= 4                      # changing the value of an element
x
```

```
## [1] 3 2 1 4 5
```

```r
x[-5]                        # remove the element 5
```

```
## [1] 3 2 1 4
```

```r
x = c( x[1:3],16, x[4:5] ) # insert the number 16 between the position 3 and 4
```

Matrices

A matrix is a rectangular structure of real numbers, thus a matrix $\boldsymbol{X}_{m\times n}$ is an rectangular "array" of scalar numbers such that:

$$\boldsymbol{X}_{m\times n} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

with m rows and n columns. Commonly the matrix is called of "dimension" or "order" $m \times n$. For the above we can say that $\boldsymbol{X}$ have m and n dimensions. Sometimes the matrix also ca be denoted as $\boldsymbol{X} = (x_{ij})$. For example, $\boldsymbol{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ is a matrix of dimension 2×3. The notation for x_{ij} indicates the elements of $\boldsymbol{X}$. Other types of characterics are:

- A matrix is **square** if $m = n$ (same number of rows and columns).
- A matrix with elements "0" is denoted as **0**
- A matrix is **symmetric** if $\boldsymbol{X} = \boldsymbol{X}^T$.
- A square matrix with all elements not on the diagonal equal to 0 is a **matriz diagonal**, for example $x_{ij} = 0$ for all $i \neq j$ (and $x_{ii} \neq 0$ for at least i).
- A diagonal matrix with 1's in the diagonal and all others not on the diagonal 0 is denoted as $\boldsymbol{I}_n$. This matrix is also kwon as **idetidy matrix**.

$$\boldsymbol{I}_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Summary of functions in `R` for matrices:

- Access to a column of a matrix $\boldsymbol{U}$, the value $j - th \Rightarrow$ `U[,j]`
- Access to a sub-set of a rows in a matrix $\boldsymbol{U} \Rightarrow$ $\texttt{U}[\texttt{i}_1 : \texttt{i}_2,]$
- Access to a sub-set of a columns in a matrix $\boldsymbol{U} \Rightarrow$ $\texttt{U}[,\texttt{j}_1 : \texttt{j}_2]$
- Access to a sub-matrix of a matrix $\boldsymbol{U} \Rightarrow$ $\texttt{U}[\texttt{i}_1 : \texttt{i}_2, \texttt{j}_1 : \texttt{j}_2]$
- Sum of $\boldsymbol{U} + \boldsymbol{V} \Rightarrow$ `U + V`
- Subtract of $\boldsymbol{U} + \boldsymbol{V} \Rightarrow$ `U - V`
- Multiplication of $\boldsymbol{UV} \Rightarrow$ `U %*% V`
- Transpose of $\boldsymbol{U}^T \Rightarrow$ `t(U)`

- Matrix-vector product $\boldsymbol{U}^T\boldsymbol{V}$ $\Rightarrow$ `crossprod(U, V)`
- Inverse of a matrix $\boldsymbol{U}^{-1}$, `solve(U)`
- Determinant of a matrix $\boldsymbol{U}$, det(A) or denoted also as |`U`| $\Rightarrow$ `det(A)`
- Diagonal fo a matrix $\boldsymbol{U}$ $\Rightarrow$ `diag(U)`
- Union of matrices by columns $\boldsymbol{U}$ y $\boldsymbol{V}$ $\Rightarrow$ `cbind(U, V)`
- Union de matrices by rows $\boldsymbol{U}$ y $\boldsymbol{V}$ $\Rightarrow$ `rbind(U, V)`
- Length of a vector $\boldsymbol{x}$ $\Rightarrow$ `length(x)`
- Dimension fo a matrix $\boldsymbol{U}$ $\Rightarrow$ `dim(U)`

Examples:

```r
A = matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=F)
B = matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=T)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```r
B
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```r
A[1, ]   # Access to the first row of A
```

```
## [1] 1 3 5
```

```r
B[2, ]   # Access to the second row of B
```

```
## [1] 4 5 6
```

```r
A[1, 3]   # Access to the element of the row 1 and column 3 of the matrix A
```

```
## [1] 5
```

```r
B[2, 3]   # Access to the element of the row 2 and column 3 of the matrix B
```

```
## [1] 6
```

Operations with matrices

```
A + B
```

```
##      [,1] [,2] [,3]
## [1,]    2    5    8
## [2,]    6    9   12
```

```
A - B
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    2
## [2,]   -2   -1    0
```

```
2*B
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    8   10   12
```

Note: If $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices and we want to multiply $\boldsymbol{A}$ with $\boldsymbol{B}$, it could be done when the number of columns in $\boldsymbol{A}$ is equal to the number of rows in $\boldsymbol{B}$.

```
t(A)  # Transpose of A
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

```r
t(A)%*%B  # Multiply the transpose of A with B
```

```
##      [,1] [,2] [,3]
## [1,]    9   12   15
## [2,]   19   26   33
## [3,]   29   40   51
```

Dimension and length of the vectors in matrices

```r
U = matrix(c(1,2,3,4,5,6),2,3)
```

```r
dim(U)       # Dimension of U
```

```
## [1] 2 3
```

```r
dim(t(U))    # Dimensión of the transpose of U
```

```
## [1] 3 2
```

```r
length(U)    # Length of the elements in U
```

```
## [1] 6
```

For loops in R

A for loop essentially iterates a process a number of times, indexed by a counter variable. The simplest example of a for loop is one which prints the numbers in a vector successively

```
u <- seq(0, 1, length=10)
for(i in 1:10)
{
print(u[i])
}
```

```
## [1] 0
## [1] 0.1111111
## [1] 0.2222222
## [1] 0.3333333
## [1] 0.4444444
## [1] 0.5555556
## [1] 0.6666667
## [1] 0.7777778
## [1] 0.8888889
## [1] 1
```

To loop over a process that is indexed by more than one number we need to use one loop within another, commonly referred to as a nested loop. For example,

```
A = matrix(0, nrow = 4, ncol = 4)

   for(i in 1:4){
   for(j in 1:4){
       A[i,j] = j^i
   }
}
print(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    1    4    9   16
## [3,]    1    8   27   64
## [4,]    1   16   81  256
```