Biostatistics I: Introduction to R

Indexing and subsetting

Eleni-Rosalina Andrinopoulou

Department of Biostatistics, Erasmus Medical Center

e.andrinopoulou@erasmusmc.nl

@erandrinopoulou

In this Section

- Indexing
- Subsetting
- A lot of practice

Indexing/Subsetting

- When transforming and analyzing data we often need to select specific observations or variables
  - Examples: Select …
    - the 3rd element of vector age
    - the 3rd column of the pbc data set
    - the sex of the 10th patient
    - all information of the 5th patient
    - the serum cholesterol for all males
    - the age for male patients or patients that have serum bilirubin > 3
    - the first measurement per patient

Indexing/Subsetting

- This can be done using square bracket (**[ ]**) notation and indices.
- Three basic types
  - position indexing
  - logical indexing
  - name indexing

Vectors

Indexing with `vector`

- For position indexing, use a **positive** value to select an element

```
x <- c(6:17)
x
```

```
 [1]  6  7  8  9 10 11 12 13 14 15 16 17
```

```
x[2]
```

```
[1] 7
```

- Use multiple positive values to select multiple elements

```
x[c(2,3,4)]
```

```
[1] 7 8 9
```

Vectors

Indexing with `vector`

- For position indexing, use duplicated **positive** values to select the same elements

```
x <- c(6:17)
x
```

```
 [1]  6  7  8  9 10 11 12 13 14 15 16 17
```

```
x[c(2,2,2)]
```

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

Vectors

Indexing with `vector`

- For position indexing, use a **negative** value to remove an element

```
x <- c(6:17)
x
```

```
 [1]  6  7  8  9 10 11 12 13 14 15 16 17
```

```
x[-5]
```

```
 [1]  6  7  8  9 11 12 13 14 15 16 17
```

- **Positive and negative indices cannot be combined**

Vectors

Indexing with `vector`

- Use logical index of the same length to select elements where the value is **TRUE**

```
x <- c(6:10)
y <- c(TRUE, FALSE, FALSE, FALSE, FALSE)
x[y]
```

```
[1] 6
```

Vectors

Indexing with `vector`

- Use logical indexing in combination with conditions

```
x <- c(6:10)
x[x > 7]
```

```
[1]  8  9 10
```

```
x[x > 7 & x > 9]
```

```
[1] 10
```

```
x[x > 7 | x > 9]
```

```
[1]  8  9 10
```

Vectors

Indexing with `vector`

- For name/character indexing, use the name of the element

```
x <- c(foo=5, bar=4, one=7, two=12, three=2)
x[c('foo', 'one')]
```

```
foo one
  5   7
```

- Use the function `names` to obtain the names

Matrices

Indexing with `matrix`

- Indexing matrices is similar to indexing vectors but with double index
  - The first position denotes the rows **["index",]**
  - The first position denotes the columns **[, "index"]**

Matrices

Indexing with `matrix`

- Indexing matrices is similar to indexing vectors but with double index
  - The first position denotes the rows **["index",]**
  - The first position denotes the columns **[, "index"]**

```
mat <- matrix(data = 1:4,
              nrow = 2, ncol = 2)
mat
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

- Use position indexing as:

```
mat <- matrix(data = 1:4,
              nrow = 2, ncol = 2)
mat[2, 2]
```

```
[1] 4
```

Matrices

Indexing with `matrix`

- Be cautious, it also works with a single index. In this case, it selects the particular element of the vector that will be included in the matrix

```
mat <- matrix(data = 1:4,
              nrow = 2, ncol = 2)
mat[2]
```

```
[1] 2
```

```
mat[[2]]
```

```
[1] 2
```

Matrices

Indexing with `matrix`

- When we leave a position blank all elements are selected

```
mat <- matrix(data = 1:4,
              nrow = 2, ncol = 2)
mat
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
mat[2, ]
```

```
[1] 2 4
```

Arrays

Indexing with `array`

```
ar <- array(data = 1:4,
            dim = c(1,2,2))
ar
```

```
, , 1

     [,1] [,2]
[1,]    1    2

, , 2

     [,1] [,2]
[1,]    3    4
```

```
ar[1, 1, ]
```

```
[1] 1 3
```

Data Frames

Indexing with `data.frame`

- Works with single and double index

```
DF <- data.frame(x = 1:3,
      y = c("male", "male", "female"))
DF
```

```
  x      y
1 1   male
2 2   male
3 3 female
```

Data Frames

Indexing with `data.frame`

- Works with single and double index

```
DF <- data.frame(x = 1:3,
      y = c("male", "male", "female"))
DF
```

```
  x      y
1 1   male
2 2   male
3 3 female
```

- Use position single indexing

```
DF[2]
```

```
       y
1   male
2   male
3 female
```

```
DF[[2]]
```

```
[1] "male"   "male"   "female"
```

Data Frames

Indexing with `data.frame`

- When using double index, indexing works like a matrix

```
DF <- data.frame(x = 1:3,
      y = c("male", "male", "female"))
DF
```

```
  x      y
1 1   male
2 2   male
3 3 female
```

- Use position indexing

```
DF[2, ]
```

```
  x    y
2 2 male
```

- Use logical indexing

```
DF[DF$x < 2, ]
```

```
  x    y
1 1 male
```

Data Frames

Indexing with `data.frame`

- **$** provides a convenient notation to extract an element by name

```
head(pbc$time)
```

```
[1]  400 4500 1012 1925 1504 2503
```

```
head(pbc[ ,"time"])
```

```
[1]  400 4500 1012 1925 1504 2503
```

Data Frames

Indexing with `data.frame`

- Combine logical and position indexing in data frame

```
head(pbc[pbc$sex == "m", 1:7])
```

```
   id time status trt      age sex ascites
3   3 1012      2   1 70.07255   m       0
14 14 1217      2   2 56.22177   m       1
21 21 3445      0   2 64.18891   m       0
24 24 4079      2   1 44.52019   m       0
48 48 4427      0   2 49.13621   m       0
52 52 2386      2   1 50.54073   m       0
```

Data Frames

Indexing with `data.frame`

- Combine logical and position indexing in data frame

```
head(pbc[pbc$age > 30 | pbc$sex == "f", 1:7])
```

```
  id time status trt      age sex ascites
1  1  400      2   1 58.76523   f       1
2  2 4500      0   1 56.44627   f       0
3  3 1012      2   1 70.07255   m       0
4  4 1925      2   1 54.74059   f       0
5  5 1504      1   2 38.10541   f       0
6  6 2503      2   2 66.25873   f       0
```

Data Frames

Indexing with `data.frame`

- Combine logical and position indexing in data frame

```
head(pbc[pbc$age > 30 & pbc$sex == "f", 1:7])
```

```
  id time status trt      age sex ascites
1  1  400      2   1 58.76523   f       1
2  2 4500      0   1 56.44627   f       0
4  4 1925      2   1 54.74059   f       0
5  5 1504      1   2 38.10541   f       0
6  6 2503      2   2 66.25873   f       0
7  7 1832      0   2 55.53457   f       0
```

Lists

Indexing with `list`

- Lists can be subsetted in the same way as vectors using single brackets - Note that the output is a list

- Use position indexing

```
mylist <- list(y = c(14, 45), z = c("m", "f", "f"))
mylist[2]
```

```
$z
[1] "m" "f" "f"
```

Lists

Indexing with `list`

- Double square brackets can be also used - Note that the output is a vector

- Use position indexing

```
mylist <- list(y = c(14, 45), z = c("m", "f", "f"))
mylist[[2]]
```

```
[1] "m" "f" "f"
```

Lists

Indexing with `list`

- **$** provides a convenient notation to extract an element by name - Note that the output is a vector

```
mylist <- list(y = c(14, 45), z = c("m", "f", "f"))
mylist
```

```
$y
[1] 14 45

$z
[1] "m" "f" "f"
```

```
mylist$y
```

```
[1] 14 45
```

Summary

Vectors

- `[]`
- `[""]` - for categorical variables

Matrices

- `[,]`
- `[[]], []`

Arrays

- `[ , , ]`

Data frames

- `[,]`
- `[[]], []`
- `$`

Lists

- `[]`
- `[[]]`
- `$`

Practice

- Use the following webpage to further investigate indexing and subsetting
  https://emcbiostatistics.shinyapps.io/indexing/