Beginings

Al

2026-01-06

R programming Course notes

Following the first course beginings. They are telling me how to begin to understand the syntax of R. For example

```
x <- 1 ## <- assignment of value to a variable. This line is assigning the value 1 to the variable x
print(x) ## outputs in the terminal the value of x
```

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

```
x ## Auto-prints the value of x
```

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

```
msg <- "Hello" ## Assigning a string
msg
```

```
## [1] "Hello"
```

```
x <- 1:20 ## Creating a vector populated with the values from 1 to 20 in their respective positions.
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Data Types

R has 5 basic or “atomic” classes of objects:

- character
- numeric
- integer
- complex
- logical (boolean)

Objects

The most basic object is a vector

- A vector can only contain objects of the same class
- *list* is a vector that can contain different classes

Empty vector is created by the vector() function.

Numbers

Numbers in R are always treated as doubles. If you want an integer specifically you need to use the L suffix (ex. 1 is double, 1L is an integer). Inf is a special number which represents infinity. Also represented as (1/0). NaN represents an undefined value or missing number.

Attributes

R objeccts can have attributes - names,dimnames
- dimensions
- class
- length
- other user-defined attributes/metadata

Attributes of an objeect can be accessed using the attributes() functions.

Creating Vectors

The c() function can be used to create vectors of objects

```
x <- c(.5,.6) ## numeric
x
```

```
## [1] 0.5 0.6
```

```
x<- c(TRUE, TRUE) ## boolean
x
```

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

```
x <- c(T,F)
x
```

```
## [1]  TRUE FALSE
```

```
x <- c("a","b","c")
x
```

```
## [1] "a" "b" "c"
```

```
x <- 9:29 ## integer
x
```

```
##  [1]  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

```
x <- c(1+0i,2+4i) ## Complex
x
```

```
## [1] 1+0i 2+4i
```

Or using the vector() function

```
x <- vector("numeric", length = 10)
x
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

What about mixing objects?

```
y <- c(1.7,"a") ## char
y
```

```
## [1] "1.7" "a"
```

```r
y <- c(T,2) ##numeric
y
```

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

```r
y<- c("a", TRUE) ##Char
y
```

```
## [1] "a"    "TRUE"
```

They get coerced to the common denominator class

Explicit coercion

Whay if you want to coerced something instead of letting the machine assume it? You can use the as." " () function.

```r
x <- 0:6
class(x)
```

```
## [1] "integer"
```

```r
as.numeric(x)
```

```
## [1] 0 1 2 3 4 5 6
```

```r
class(as.numeric(x))
```

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

```r
as.logical(x)
```

```
## [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```r
as.character(x)
```

```
## [1] "0" "1" "2" "3" "4" "5" "6"
```

Nonsensical coercion results in NA s

List

Vectors with different classes of opbjects

```r
x <- list(1,"a",TRUE,1+4i,2+0i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] 1+4i
##
## [[5]]
## [1] 2+0i
```

Matrices

matrices are vectors with an dimension attribute. The dimension attribute is itself an integer vector og length 2 (nrow,ncol)

```
m <- matrix(nrow=2,ncol = 3)
m
```

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

```
dim(m)
```

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

```
attributes(m)
```

```
## $dim
## [1] 2 3
```

Matrices are constructed column wise, so entries can thought of starting in the upper left and running down the column

They can alse be created directly from vectors by adding dimension atribute

```
m<-1:10
dim(m) <- c(2,5)
m
```

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

cbind-ing and rbind-ing

Matrices can be created by column binding or row binding with `cbind()` and `rbind()`

```
x <- 1:3
y <- 10:12
cbind (x,y)
```

```
##      x  y
## [1,] 1 10
## [2,] 2 11
## [3,] 3 12
```

```
rbind(x,y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y   10   11   12
```

Factors

Factors are used to represent categorical data. Factors can be ordered or unordered. one can think of a factor as an integer vector where each integer has a label.

- Factors are treated specially by modelling function lime `lm()` and `glm()`
- Using factors with labels is better than using integers because factors are self describing; having a variable that has values "Male" and "Female" is better than a variable that has value 1 and 2.

```r
x <- factor(c("yes", "yes","no", "yes", "no"))
x
```

```
## [1] yes yes no  yes no
## Levels: no yes
```

```r
table(x)
```

```
## x
##  no yes
##   2   3
```

```r
unclass(x)
```

```
## [1] 2 2 1 2 1
## attr(,"levels")
## [1] "no"  "yes"
```

The order of the levels can be set using the `levels` argument to `factor()`. This can be important in linear modelling because the first level is used as the baseline level.

```r
x <- factor(c("yes","yes","no","yes","no"),levels = c("yes","no"))
x
```

```
## [1] yes yes no  yes no
## Levels: yes no
```

```r
unclass(x)
```

```
## [1] 1 1 2 1 2
## attr(,"levels")
## [1] "yes" "no"
```

Missing values

Msiing values are denoted by `NA` or `NaN` for undefined mathematical operations.

- `is.na()` is used to test objects if they are NA
- `is.nan()` same for NaN
- NA values have a class also, so there are integer NA, char NA, etc
- A NaN value is also NA but the converse is not true

Data Frames

Data frames are used to store tabular data - They are represented as a special type opf list where every element of the list has to have the same length
- Each element of the list can be though of as a column and the length of each element of the list is the number of rows
- Unlike matrices, data frames can store different classes of objects in each column; matrices must have every element be the same class
- Data frames also have a special attribute called `row.names`
- Data frames are usually created by calling `read.table()` or `read.csv()`
- Can be converted to a matrix by calling `data.matrix()`

```r
x <- data.frame(foo = 1:4, bar = c(T,T,F,F))
x
```

```
##   foo   bar
## 1   1  TRUE
## 2   2  TRUE
## 3   3 FALSE
## 4   4 FALSE
```

```
nrow(x)
```

```
## [1] 4
```

```
ncol(x)
```

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

Name Attribute

R objects can also have names, which is very useful for writing readable code and self-describing objects

```
x<-1:3
names(x)
```

```
## NULL
```

```
names(x) <- c("foo","bar", "norf")
x
```

```
##  foo  bar norf
##    1    2    3
```

```
names(x)
```

```
## [1] "foo"  "bar"  "norf"
```

List can also have names

```
x<-list(a=1,b=2, c=3)
x
```

```
## $a
## [1] 1
##
## $b
## [1] 2
##
## $c
## [1] 3
```

and matrices

```
m<-matrix(1:4,nrow =2, ncol = 2)
dimnames(m) <- list(c("a","b"),c("c","d"))
m
```

```
##   c d
## a 1 3
## b 2 4
```

Summary

Data types - Atomic classes:numeric, logical,char,int,complex - vectors, list
- factors

- missing values
- data frames
- names