Introduction to R for data analysis

2. data types in R

Carl Herrmann & Carlos Ramirez
R4SC - Freiburg June 2024

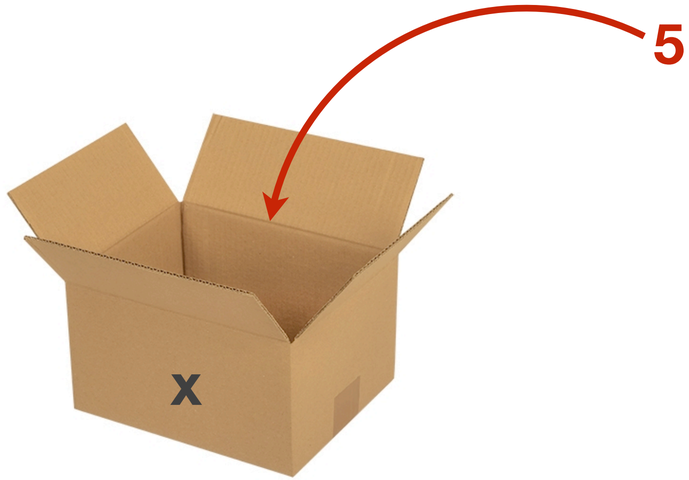
IPMB
Institut für Pharmazie und Molekulare Biotechnologie

UNIVERSITÄT HEIDELBERG
ZUKUNFT SEIT 1386

Data types in R

- One of the most important first steps when working with R is to understand the **different data types** used
- The data type indicates **what type of data is stored in a variable**, and in what format
- Think of a variable as a type of storage, which can contain something (numbers / text / lists / more complex objects)

`x = 5`



Simple data types

- With the sign "=" we can assign a value to a variable
- In R, we can also use the symbol "<-"
- Meaning: the value right of the symbol is assigned to the variable on the left of the symbol
- Depending on the type of the value on the right hand side, the variable will have a different data type

```
> x = 5
> class(x)
[1] "numeric"
>
> x = 5L
> class(x)
[1] "integer"
>
> x = TRUE
> class(x)
[1] "logical"
>
> x ="french"
> class(x)
[1] "character"
>
```

Converting data types

- Some data types can be converted from one to the other
- The is done with functions such as `as.numeric as.integer as.character`
- Not all convertions are possible!

  - numeric → character ✔
  - character → numeric ✘ / ✔

  - boolean → numeric ✔
  - numeric → boolean ✔

  - boolean → character ✔
  - character → boolean ✘ / ✔

```
> x = 2
> as.character(x)
[1] "2"
>
> x = 'blue'
> as.numeric(x)
[1] NA
Warning message:
NAs introduced by coercion
>
> x = TRUE
> as.numeric(x)
[1] 1
>
> x= FALSE
> as.numeric(x)
[1] 0
>
> x = 1
> as.logical(x)
[1] TRUE
>
> x = 4
> as.logical(x)
[1] TRUE
>
> x = 0
> as.logical(x)
[1] FALSE
```

```
> x = TRUE
> as.character(x)
[1] "TRUE"
>
> x = 'blue'
> as.logical(x)
[1] NA
>
> x='TRUE'
> as.logical(x)
[1] TRUE
```

Vectors

- Certain types of variables allow to **store multiple values** in the same object
- Using a vector, you can store multiple values
- Vectors are created using the `c( )` function

```
> x = c(5,3,4,2)
> class(x)
[1] "numeric"
>
> x[2]
[1] 3
>
> x = c('tom',5,'joe',7,TRUE)
> class(x)
[1] "character"
> x
[1] "tom"   "5"     "joe"   "7"     "TRUE"
> x[4]
[1] "7"
```

Note: vectors can only contain one type of data types

Matrices

- We can extend the idea of the vector two object with 2 dimensions (rows and columns) : **matrices**

```
> x
            [,1]        [,2]       [,3]
[1,]  0.76875195 -0.03627775 -0.4476004
[2,] -1.48748472 -1.49096738  2.6160260
[3,]  0.08930188 -0.78898598  0.7977257
[4,]  2.81901539 -1.63473071  1.6165126
[5,] -0.96473245 -0.67277273  1.1914672
> class(x)
[1] "matrix"
>
> x[4,2]
[1] -1.634731
>
> dim(x)
[1] 5 3
>
> x[,2]
[1] -0.03627775 -1.49096738 -0.78898598 -1.63473071 -0.67277273
>
> x[1,]
[1]  0.76875195 -0.03627775 -0.44760037
>
```

Note: a column or row of a matrix is a vector

Data frames

- A matrix can only contain one type of variables: numeric or character or boolean
- However, sometimes, we want to store different types of data:
  - patient name : a column of characters
  - alive? a column of boolean
  - age : a column of numeric values
- This can be done using a more general data type: the **data frame**
- A matrix can be converted into a data frame
- The reverse is (obviously) not always possible!

Data frames

```
> x
    chol age gender alive
285  226  20 female  TRUE
128  248  34   male  TRUE
136  170  65   male  TRUE
156  227  25   male  TRUE
249  245  47 female FALSE
189  277  63 female  TRUE
158  236  62   male FALSE
98   228  24 female FALSE
174  199  25   male FALSE
38   268  38 female FALSE
>
> class(x)
[1] "data.frame"
>
> rownames(x)
 [1] "285" "128" "136" "156" "249" "189" "158" "98"  "174" "38"
>
> colnames(x)
[1] "chol"   "age"    "gender" "alive"
>
```

Data frame

- We can add new columns to an existing data fra

```
> x
   chol age gender alive
1   203  46 female  TRUE
2   165  29 female FALSE
3   228  58 female FALSE
4    78  67   male FALSE
5   249  64   male  TRUE
6   248  34   male  TRUE
7   195  30   male  TRUE
8   227  37   male FALSE
9   177  45   male FALSE
10  263  55 female  TRUE
>
> x$over50 = x$age > 50
>
> x
   chol age gender alive over50
1   203  46 female  TRUE  FALSE
2   165  29 female FALSE  FALSE
3   228  58 female FALSE   TRUE
4    78  67   male FALSE   TRUE
5   249  64   male  TRUE   TRUE
6   248  34   male  TRUE  FALSE
7   195  30   male  TRUE  FALSE
8   227  37   male FALSE  FALSE
9   177  45   male FALSE  FALSE
10  263  55 female  TRUE   TRUE
>
> x$ageMonth = x$age*12
>
> x
   chol age gender alive over50 ageMonth
1   203  46 female  TRUE  FALSE      552
2   165  29 female FALSE  FALSE      348
3   228  58 female FALSE   TRUE      696
4    78  67   male FALSE   TRUE      804
5   249  64   male  TRUE   TRUE      768
6   248  34   male  TRUE  FALSE      408
7   195  30   male  TRUE  FALSE      360
8   227  37   male FALSE  FALSE      444
9   177  45   male FALSE  FALSE      540
10  263  55 female  TRUE   TRUE      660
```

Differences matrix/data frame

- The difference between matrices and data frames is a huges source of error messages...

```
> X
             a          b          c           d
[1,]  1.2637459 -0.1206988  0.8030185 -1.10323409
[2,]  0.4371218  4.0146618  1.9417579 -0.22278767
[3,]  0.9385465  0.4433365  0.3961076 -0.08211448
[4,] -1.0822598  0.1024311 -0.1462680  0.87965382
[5,]  0.4454623  0.3852503  0.7132976 -1.06716402
[6,]  0.5689450  1.4346889  0.7152038 -0.84166319
>
> class(X)
[1] "matrix"
>
> X$a
Error in X$a : $ operator is invalid for atomic vectors
>
> Y = as.data.frame(X)
> Y
           a          b          c           d
1  1.2637459 -0.1206988  0.8030185 -1.10323409
2  0.4371218  4.0146618  1.9417579 -0.22278767
3  0.9385465  0.4433365  0.3961076 -0.08211448
4 -1.0822598  0.1024311 -0.1462680  0.87965382
5  0.4454623  0.3852503  0.7132976 -1.06716402
6  0.5689450  1.4346889  0.7152038 -0.84166319
>
> Y$a
[1]  1.2637459  0.4371218  0.9385465 -1.0822598  0.4454623  0.5689450
```

X is a matrix

we cannot extract columns from matrices with this syntax!!

Convert X to a data frame

Now it works!

Complex data types

- Besides these "simple" built-in data types, certain packages create specific storage types (or data types) adapted to the data that is created
- These allow to store multiple type of information in the same variable
- Example: gene expression data
  - raw numeric expression values
  - normalized numeric values
  - names of the genes
  - names of the samples
  - metadata (sequencer / ...)
- The different storage areas within these complex objects are called **slots**

Example: Seurat object

```
> pbmc
An object of class Seurat
13714 features across 2700 samples within 1 assay
Active assay: RNA (13714 features)
>
> str(pbmc)
Formal class 'Seurat' [package "Seurat"] with 12 slots
  ..@ assays      :List of 1
  .. ..$ RNA:Formal class 'Assay' [package "Seurat"] with 8 slots
  .. .. .. .. ..@ counts       :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
  .. .. .. .. .. .. ..@ i       : int [1:2282976] 29 73 80 148 163 184 186 227 229 230 ...
  .. .. .. .. .. .. ..@ p       : int [1:2701] 0 779 2131 3260 4220 4741 5522 6304 7094 7626 ...
  .. .. .. .. .. .. ..@ Dim     : int [1:2] 13714 2700
  .. .. .. .. .. .. ..@ Dimnames:List of 2
  .. .. .. .. .. .. .. ..$ : chr [1:13714] "AL627309.1" "AP006222.2" "RP11-206L10.2" "RP11-206L10.9" ...
  .. .. .. .. .. .. .. ..$ : chr [1:2700] "AAACATACAACCAC" "AAACATTGAGCTAC" "AAACATTGATCAGC" "AAACCGTGCTTCCG" ...
  .. .. .. .. .. .. ..@ x       : num [1:2282976] 1 1 2 1 1 1 1 41 1 1 ...
  .. .. .. .. .. .. ..@ factors : list()
  .. .. .. ..@ data         :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
  .. .. .. .. .. .. ..@ i       : int [1:2282976] 29 73 80 148 163 184 186 227 229 230 ...
  .. .. .. .. .. .. ..@ p       : int [1:2701] 0 779 2131 3260 4220 4741 5522 6304 7094 7626 ...
  .. .. .. .. .. .. ..@ Dim     : int [1:2] 13714 2700
  .. .. .. .. .. .. ..@ Dimnames:List of 2
  .. .. .. .. .. .. .. ..$ : chr [1:13714] "AL627309.1" "AP006222.2" "RP11-206L10.2" "RP11-206L10.9" ...
  .. .. .. .. .. .. .. ..$ : chr [1:2700] "AAACATACAACCAC" "AAACATTGAGCTAC" "AAACATTGATCAGC" "AAACCGTGCTTCCG" ...
  .. .. .. .. .. .. ..@ x       : num [1:2282976] 1 1 2 1 1 1 1 41 1 1 ...
  .. .. .. .. .. .. ..@ factors : list()
  .. .. .. ..@ scale.data   : num[0 , 0 ]
  .. .. .. ..@ key          : chr "rna_"
  .. .. .. ..@ assay.orig   : NULL
  .. .. .. ..@ var.features : logi(0)
  .. .. .. ..@ meta.features:'data.frame':      13714 obs. of  0 variables
  .. .. .. ..@ misc         : NULL
  ..@ meta.data   :'data.frame':        2700 obs. of  3 variables:
  .. ..$ orig.ident  : Factor w/ 1 level "pbmc3k": 1 1 1 1 1 1 1 1 1 1 ...
  .. ..$ nCount_RNA  : num [1:2700] 2419 4903 3147 2639 980 ...
  .. ..$ nFeature_RNA: int [1:2700] 779 1352 1129 960 521 781 782 790 532 550 ...
  ..@ active.assay: chr "RNA"
```