Data types in R

Colin Rundel

2019-01-17

Generic Vectors

Lists

Lists are *generic vectors*, in that they are 1 dimensional (i.e. have a length) and can contain any type of R object.

```
list("A", c(TRUE,FALSE), (1:4)/2, function(x) x^2)
```

```
## [[1]]
## [1] "A"
##
## [[2]]
## [1]  TRUE FALSE
##
## [[3]]
## [1] 0.5 1.0 1.5 2.0
##
## [[4]]
## function(x) x^2
```

Structure

Often we want a more compact representation of a complex object, the `str` function is useful for this particular task

```
str( list("A", c(TRUE,FALSE), (1:4)/2, function(x) x^2) )
```

```
## List of 4
##  $ : chr "A"
##  $ : logi [1:2] TRUE FALSE
##  $ : num [1:4] 0.5 1 1.5 2
##  $ :function (x)
##   ..- attr(*, "srcref")= 'srcref' int [1:8] 1 40 1 54 40 54 1 1
##   .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7f91004120c8>
```

Recursive lists

Lists can contain other lists, meaning they don't have to be flat

```
str( list(1, list(2, list(3, 4), 5)) )
```

```
## List of 2
##  $ : num 1
##  $ :List of 3
##   ..$ : num 2
##   ..$ :List of 2
##   .. ..$ : num 3
##   .. ..$ : num 4
##   ..$ : num 5
```

List Coercion

By default a vector will be coerced to a list (as a list is more generic) if needed

```
str( c(1, list(4, list(6, 7))) )
```

```
## List of 3
##  $ : num 1
##  $ : num 4
##  $ :List of 2
##   ..$ : num 6
##   ..$ : num 7
```

List Coercion

By default a vector will be coerced to a list (as a list is more generic) if needed

```
str( c(1, list(4, list(6, 7))) )
```

```
## List of 3
##  $ : num 1
##  $ : num 4
##  $ :List of 2
##   ..$ : num 6
##   ..$ : num 7
```

We can coerce a list into an atomic vector using `unlist` - the usual type coercion rules then apply to determine its type.

```
unlist(list(1:3, list(4:5, 6)))
```

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

```
unlist( list(1, list(2, list(3, "Hello"))) )
```

```
## [1] "1"     "2"     "3"     "Hello"
```

Named lists

Because of their more complex structure we often want to name the elements of a list (we can also do this with vectors). This can make reading and accessing the list more straight forward.

```
str(list(A = 1, B = list(C = 2, D = 3)))
```

```
## List of 2
##  $ A: num 1
##  $ B:List of 2
##   ..$ C: num 2
##   ..$ D: num 3
```

```
list("knock knock" = "who's there?")
```

```
## $`knock knock`
## [1] "who's there?"
```

```
names(list(ABC=1, DEF=list(H=2, I=3)))
```

```
## [1] "ABC" "DEF"
```

Exercise 2

Represent the following JSON data as a list in R.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address":
  {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumber":
  [
    {
      "type": "home",
      "number": "212 555-1239"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ]
}
```

Acknowledgments

Acknowledgments

Above materials are derived in part from the following sources:

- Hadley Wickham - Advanced R
- R Language Definition