Advanced R by Hadley Wickham

Chapter 3: Vectors

Tony ElHabr

@TonyElHabr

2020-04-16

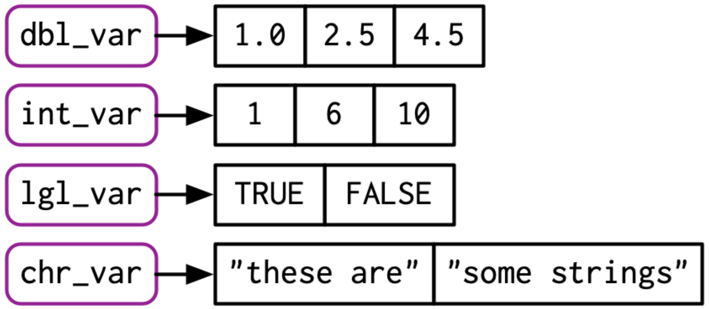
What's in Chapter 3

- Section 3.2: atomic vectors
- Section 3.3: attributes
- Section 3.4: "special" vectors (S3 atomic vectors)
- Section 3.5: lists
- Section 3.6: data frames and tibbles
- Section 3.7: `NULL`

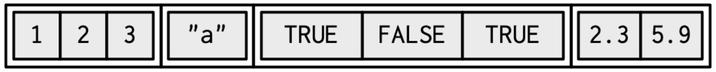
Vectors

- 2 types: atomic and list

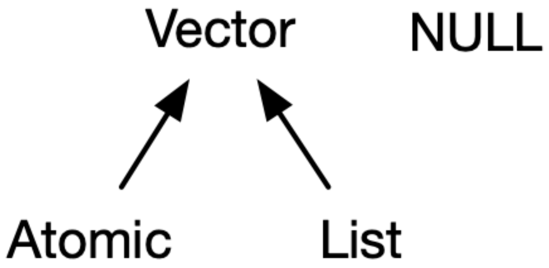
atomic



list

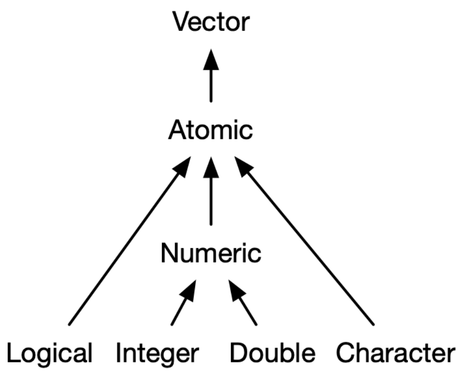


… and there is also `NULL`



Atomic Vectors

- **4** primary types: logical, integer, double, character (i.e. strings)



```
c(TRUE, FALSE, T, F)
c(1234L, 42L)
c(3.14, .314e1, 0xbada55)
c('single quote', "double quote")
```

… also raw and complex

```
raw(42)
complex(real = 0, imaginary = -1)
```

- Check type with `typeof()`

Coercion

- **Coercion** happens when you attempt to combine vectors with elements of different types
- Coercion order: character → double → integer → logical

```
c(1, 1.01) # to double
```

```
## [1] 1.00 1.01
```

```
c(1, '1') # to character
```

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

```
c(1, TRUE) # to integer
```

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

- Explicity coerce with `as.*()` functions

```
as.integer(c(1, 1.01))
```

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

- Failed coercion leads to warnings and `NA`

```
as.integer(c('1', '1.01', 'a'))
```

```
## Warning: NAs introduced by coercion
## [1]  1  1 NA
```

NA and NULL

- `NA` is a "sentinel" value for explicit missingness
- `NA` can be of any type, e.g. `NA_integer_`, `NA_character_`, etc.
- Calculations involving `NA`s usually result in more `NA`s

```
1 + NA
```

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

...although not always

```
1 | NA
```

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

- Test with `is.na()`

- `NULL` is its own vector type

```
typeof(NULL)
```

```
## [1] "NULL"
```

- Zero-length

```
length(NULL)
```

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

- Cannot have attributes

```
x <- NULL
attr(x, 'y') <- 1 # error
```

- Test with `is.null()`

Attributes

- Name-value pairs of metadata for R objects
- Get and set a single attribute with `attr()`

```r
x <- 'a'
attr(x, 'what') <- 'apple'
attr(x, 'what')
```

```
## [1] "apple"
```

- Get and set multiple attributes with `attributes()` and `structure()`

```r
x <- structure('a', what = 'apple', type = 'fruit')
attributes(x)
```

```
## $what
## [1] "apple"
##
## $type
## [1] "fruit"
```

- With the exception of `names()` and `dim()`, most attributes are lost with calculations

```r
attributes(x[1])
```

```
## NULL
```

`names()`

- `names()` can be assigned in multiple ways

```
x <- c(apple = 'a', banana = 'b') # 1
x
y <- c('a', 'b')
names(y) <- c('apple', 'banana') # 2
y
setNames(y, c('apple', 'banana')) # 3
```

```
##  apple banana
##    "a"    "b"
##  apple banana
##    "a"    "b"
##  apple banana
##    "a"    "b"
```

`dim()`

- `dim()` has the capability of turning a 1-d vector into a 2-d matrix or an n-d array

```
a <- matrix(1:6, nrow = 2, ncol = 3)
a
```

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

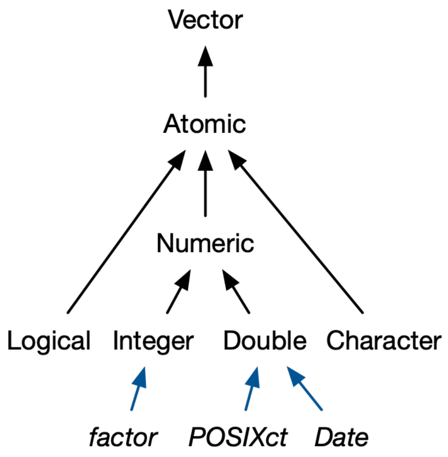
```
b <- array(1:6, dim = c(1, 3, 2))
b
```

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

- Weird things
    - 1-d vector without a `dim` attribute has `NULL` dimension
    - Matrices and arrays can be a single column or row vector

S3 atomic vectors

- Objects with a `class` attribute, making them **S3 objects**
- **4** important S3 vector types in base R: `factor` (categorical), `Date` (Date), `POSIXct` (date-time), `duration` (difftime).



Factors

- Vector that can only contain pre-defined values
- Has two attributes: `class` and `levels`
- Built on top of integers, not characters

```r
fruits <- factor(c('banana', 'apple', 'carrot'))
fruits
```

```
## [1] banana apple  carrot
## Levels: apple banana carrot
```

- Variation: **ordered** factors

```r
x <- ordered(c('two', 'three', 'one'), levels = c('one', 'two', 'three'))
x
```

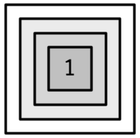
```
## [1] two   three one
## Levels: one < two < three
```

`Date`, `POSIXct`, and `duration`

- All built on top of doubles
- Dates have `class = "Date"`
- Date-times are trickier…
    - Represent seconds since Jan. 1, 1970
    - `POSIXct` isn't the only possible class; there's also `POSIXlt`
    - Also have a "parent" class of `POSIXt`
    - Have a `tzone` attribute
- Durations have 2 attributes: `class = "difftime"` and `units` corresponding to a temporal unit, e.g. `"day"`

Lists

- Each element can by of any atomic type, or even another list



- Each element is really a reference

```
x <- 1L
lobstr::obj_size(x)
```

```
## 56 B
```

```
lobstr::obj_size(rep(x, 3L))
```

```
## 64 B
```

- Combining with `c` is different than wrapping with `list()`

```
x <- list(a = 1, b = 2)
y <- list(c = -1, d = -2)
length(list(x, y))
```

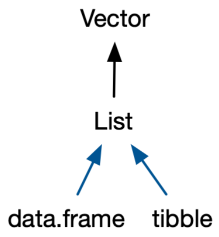
```
## [1] 2
```

```
length(c(x, y))
```

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

Data frames

- S3 vectors built on top of lists



```
df <- data.frame(col1 = 1:2, col2 = c('a', 'b'))
df
```

```
##   col1 col2
## 1    1    a
## 2    2    b
```

- Data frames have some undesireable default behavior

```
class(df$col2)
```

```
## [1] "character"
```

… which spawned tibbles (with the `{tibble}` package)

```
tbl <- tibble::tibble(col1 = 1:2, col2 = c('a', 'b'))
class(tbl$col2)
```

```
## [1] "character"
```

Data frame vs tibble behavior

- Tibble don't coerce strings to factors by default
- Tibbles discourage rownames, which are generally "bad"
- Tibbles have a "prettier" print method
- Tibbles have stricter subsetting rules

Non-your-typical column

- Data frame columns can be lists

```
data.frame(x = 1:2, y = I(list(1:3, 1:4)))
```

```
##   x          y
## 1 1    1, 2, 3
## 2 2 1, 2, 3, 4
```

- Easier list-column creation with tibbles

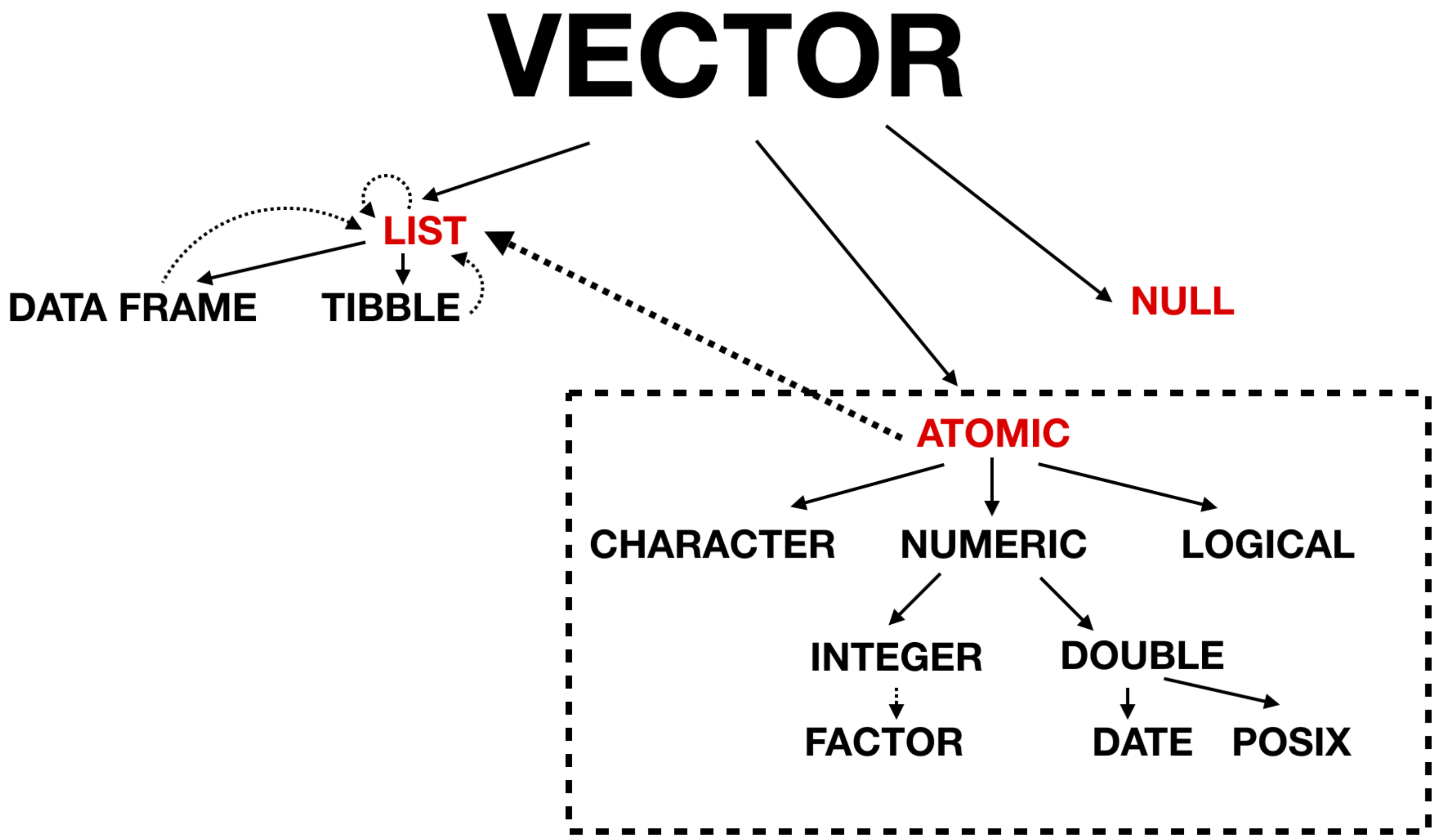
```
tibble::tibble(x = 1:2, y = list(1:3, 1:4))
```

```
## # A tibble: 2 x 2
##       x y
##   <int> <list>
## 1     1 <int [3]>
## 2     2 <int [4]>
```

- Columns can even be matrices and data frames

```
data.frame(x = 1:2, y = matrix(3:6, nrow = 2))
data.frame(x = 1:2, y = data.frame(a = 3:4, b = 5:6))
```

In Review


VECTOR
LIST
DATA FRAME
TIBBLE
NULL
ATOMIC
CHARACTER
NUMERIC
LOGICAL
INTEGER
DOUBLE
FACTOR
DATE
POSIX