Advanced R in Korean

Jsang

2020-02-20

Contents

Preface

- Wickham Advanced R .
- .
- .
- , / .
- (@) .

Chapter 1

Introduction

Chapter 2

Names and values

2.1 Introduction

R (object) (name) . ,

- .
- , .
- R .

(names) (values) , R (object) .

Quiz

. Section 2.7 .

1. , `1` `2` “3” ? `[[` , `$` . `1`, `2`, `3` ?

```r
df <- data.frame(runif(3), runif(3))
names(df) <- c(1, 2)
```

2. , `y` ?

```r
x <- runif(1e6)
y <- list(x, x, x)
```

3. `a` ?

```r
a <- c(1, 5, 3, 2)
b <- a
b[[1]] <- 10
```

Outline

- Section 2.2 (names) (values) , `<-` (binding) (reference) .
- Section 2.3 R ' ' . , . `tracemem()` . , , , .
- Section 2.4 (object) , . , `utils::object.size()` , `lobstr::obj_size()` .
- Section 2.5 'copy-on-modify'(@ ' ' .) . (environments) , .
- Section 2.6 , garbage collector .

Prerequisites

R lobstr .

```r
library(lobstr)
```

2.2 Binding basics

.

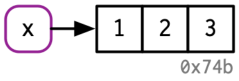
```r
x <- c(1, 2, 3)
```

" *x* , *1, 2, 3* . " . , R . .

- `c(1, 2, 3)` .
- `x` .

, . .

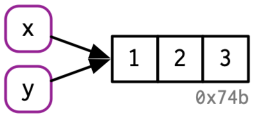
, .



x , . c(1, 2, 3) (,) .
<- , <- (binding) .

, (name) (value) (reference) .(@ ‘ ’ ?)
, c(1, 2, 3) . ’ ’ .(@ ,
)

```
y <- x
```



c(1, 2, 3) 0x74b . , . ,
(ID) . ID ’ ’ . .
, ID .

lobstr::obj_addr() ID . x y ID .

```
obj_addr(x)
#> [1] "0x18308880"
obj_addr(y)
#> [1] "0x18308880"
```

ID R .

. , . .

2.2.1 Non-syntactic names

R . **(syntactic)** , , ., _ , _ .
TRUE, NULL, if, function **(reserved words)** .(@ R
.) **non-syntactic** , .

```
_abc <- 1
#> Error: unexpected input in "_"

if <- 10
#> Error: unexpected assignment in "if <-"
```

. backticks .

```
`_abc` <- 1
`_abc`
#> [1] 1

`if` <- 10
`if`
#> [1] 10
```

, R , .

2.2.2 Exercises

1. `a`, `b`, `c`, `d` .

```
a <- 1:10
b <- a
c <- b
d <- 1:10
```

2. . ? `lobstr::obj_addr()` .

```
mean
base::mean
get("mean")
evalq(mean)
match.fun("mean")
```

3. `read.csv` R , non-syntactic syntactic . ? ?

4. non-syntactic syntactic `make.names()` ?

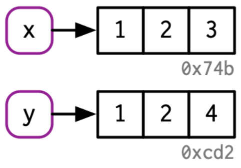
5. syntactic . `.123e1` syntactic ? `?makes.names` .

2.3 Copy-on-modify

. `x` `y` . `y` .

```r
x <- c(1, 2, 3)
y <- x

y[[3]] <- 4
x
#> [1] 1 2 3
```

y x . ? y , . , R 0x74b 0xcd2 , y .



copy-on-modify . R . , R (unchangeable), **(immutable)** . , Section 2.5 copy-on-modify .

copy-on-modify , RStudio . (environment pane) , . , . . , R , RMarkdown .

2.3.1 `tracemem()`

`base::tracemem()` . , .

```r
x <- c(1, 2, 3)
cat(tracemem(x), "\n")
#> <0x7f80c0e0ffc8>
```

`tracemem()` , .

```r
y <- x
y[[3]] <- 4L
#> tracemem[0x7f80c0e0ffc8 -> 0x7f80c4427f40]:
```

y , . , R modify-in-place . Section 2.5 .

```
y[[3]] <- 5L

untracemem(x)
```

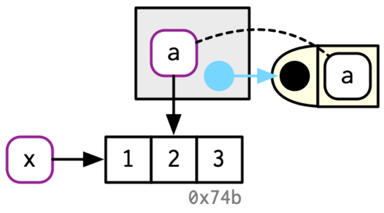
untracemem() tracemem() , .

2.3.2 Function calls

. .

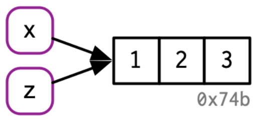
```
f <- function(a) {
  a
}

x <- c(1, 2, 3)
cat(tracemem(x), "\n")
#> <000000001A2688D0>

z <- f(x)
# there's no copy here!

untracemem(x)
```

f() , a x .



Section 7.4.4 . , f() . a
, ( ) ( ) .

f() , x z . 0x74b , . f() x , R
, z .

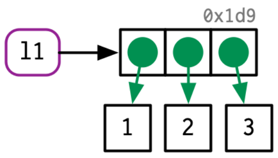


2.3.3 Lists

(,) . . , .

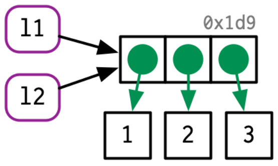
```r
l1 <- list(1, 2, 3)
```

, , ( ) .

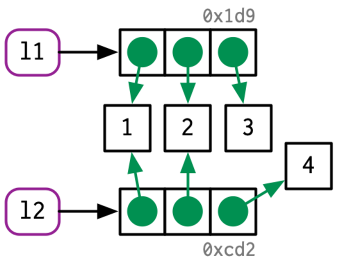


.

```r
l2 <- l1
```



```r
l2[[3]] <- 4
```



, copy-on-modify . , . .
, , . , . R 3.1.0
.

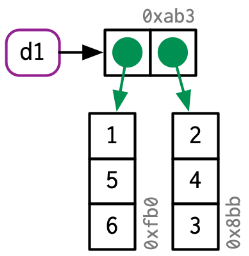
, `lobstr::ref()` . `ref()` ID ,
.

```
ref(l1, l2)
#> o [1:0x17fa8250] <list>
#> +-[2:0x17f958d8] <dbl>
#> +-[3:0x17f958a0] <dbl>
#> \-[4:0x17f95868] <dbl>
#>
#> o [5:0x18843c58] <list>
#> +-[2:0x17f958d8]
#> +-[3:0x17f958a0]
#> \-[6:0x18a12310] <dbl>
```

2.3.4 Data frames

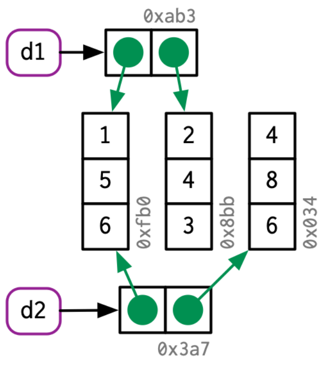
. copy-on-modify . .

```
d1 <- data.frame(x = c(1, 5, 6), y = c(2, 4, 3))
```



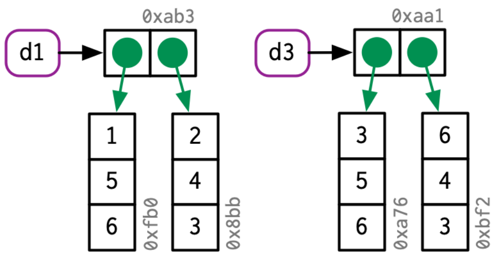
, . .

```
d2 <- d1
d2[, 2] <- d2[, 2] * 2
```



, . .

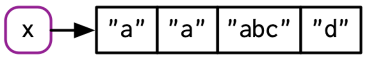
```
d3 <- d1
d3[1, ] <- d3[1, ] * 3
```



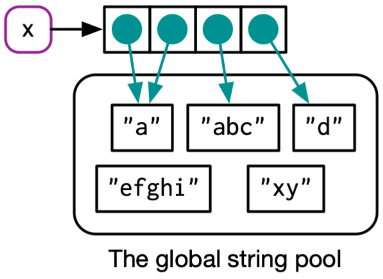
2.3.5 Character vectors

R . .

```
x <- c("a", "a", "abc", "d")
```



. R **global string pool** , .



`ref()` `character` `TRUE` .

```
ref(x, character = TRUE)
#> o [1:0x18fd9000] <chr>
#> +-[2:0x12f376d0] <string: "a">
#> +-[2:0x12f376d0]
#> +-[3:0x1994f480] <string: "abc">
#> \-[4:0x13135330] <string: "d">
```

, , .

2.3.6 Exercises

1. `tracemem(1:10)` ?
2. , `tracemem()` . : .

```
x <- c(1L, 2L, 3L)
tracemem(x)

x[[3]] <- 4
```

3. .

```
a <- 1:10
b <- list(a, a)
c <- list(b, a, 1:10)
```

4. ?

```
x <- list(1:10)
x[[2]] <- x
```

.

2.4 Object size

`lobstr::obj_size()` .

```
obj_size(letters)
#> 1,712 B
obj_size(ggplot2::diamonds)
#> 3,456,344 B
```

, .

```
x <- runif(1e6)
obj_size(x)
#> 8,000,048 B

y <- list(x, x, x)
obj_size(y)
#> 8,000,128 B
```

y x 80 , .

```
obj_size(list(NULL, NULL, NULL))
#> 80 B
```

, R global string pool . 100 100
.

```
banana <- "bananas bananas bananas"
obj_size(banana)
#> 136 B
obj_size(rep(banana, 100))
#> 928 B
```

. `obj_size(x) + obj_size(y)` `obj_size(x, y)` . , `x` `y` `y` .

```
obj_size(x, y)
#> 8,000,128 B
```

, 3.5.0 R ALTREP , **alternative representation** . R . : ,
. , .

```
obj_size(1:3)
#> 680 B
obj_size(1:1e3)
#> 680 B
obj_size(1:1e6)
#> 680 B
obj_size(1:1e9)
#> 680 B
```

2.4.1 Exercises

1. , `object.size(y)` `obj_size(y)` ? `object.size()` .

```
y <- rep(list(runif(1e4)), 100)

object.size(y)
#> 8005648 bytes
obj_size(y)
#> 80,896 B
```

2. , ?

```
funs <- list(mean, sd, var)
obj_size(funs)
#> 17,608 B
```

3. .

```
a <- runif(1e6)
obj_size(a)
#> 8,000,048 B

b <- list(a, a)
obj_size(b)
#> 8,000,112 B
obj_size(a, b)
#> 8,000,112 B

b[[1]][[1]] <- 10
obj_size(b)
#> 16,000,160 B
obj_size(a, b)
#> 16,000,160 B

b[[2]][[1]] <- 10
obj_size(b)
#> 16,000,160 B
obj_size(a, b)
#> 24,000,208 B
```

2.5 Modify-in-place

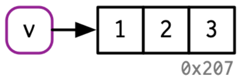
, R . .

- .
- (Environments) .(modified in place)

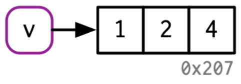
2.5.1 Objects with a single binding

, R .

```
v <- c(1, 2, 3)
```



```
v[[3]] <- 4
```



(ID , v 0bx207 .)

R .

- , R 0, 1 . , , , 1 . . R .
- , . “ ” C . R-core .

. R For , . . .

```
x <- data.frame(matrix(runif(5 * 1e4), ncol = 5))
medians <- vapply(x, median, numeric(1))

for (i in seq_along(medians)) {
  x[[i]] <- x[[i]] - medians[[i]]
}
```

loop , loop . tracemem() .

```
cat(tracemem(x), "\n")

#> <0x7f80c429e020>
for (i in 1:5) {
  x[[i]] <- x[[i]] - medians[[i]]
}
#> tracemem[0x7f80c429e020 -> 0x7f80c0c144d8]:
#> tracemem[0x7f80c0c144d8 -> 0x7f80c0c14540]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c14540 -> 0x7f80c0c145a8]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c145a8 -> 0x7f80c0c14610]:
#> tracemem[0x7f80c0c14610 -> 0x7f80c0c14678]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c14678 -> 0x7f80c0c146e0]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c146e0 -> 0x7f80c0c14748]:
#> tracemem[0x7f80c0c14748 -> 0x7f80c0c147b0]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c147b0 -> 0x7f80c0c14818]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c14818 -> 0x7f80c0c14880]:
#> tracemem[0x7f80c0c14880 -> 0x7f80c0c148e8]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c148e8 -> 0x7f80c0c14950]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c14950 -> 0x7f80c0c149b8]:
#> tracemem[0x7f80c0c149b8 -> 0x7f80c0c14a20]: [[<-.data.frame [[<-
#> tracemem[0x7f80c0c14a20 -> 0x7f80c0c14a88]: [[<-.data.frame [[<-

untracemem(x)
```

, , , . `[[.data.frame` ,
`[[.data.frame` `x` (regular function) .(@ ...)

. C , .

```
y <- as.list(x)
cat(tracemem(y), "\n")
#> <0x7f80c5c3de20>

for (i in 1:5) {
  y[[i]] <- y[[i]] - medians[[i]]
}
#> tracemem[0x7f80c5c3de20 -> 0x7f80c48de210]:
```

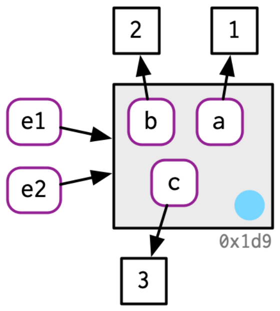
, . , Chapter 25 C++
.

2.5.2 Environments

Chapter 7 , . .
.(modified in place) **reference semantics** ,
.

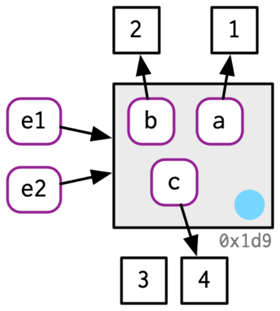
. `e1` `e2` .

```
e1 <- rlang::env(a = 1, b = 2, c = 3)
e2 <- e1
```



, modified in place .

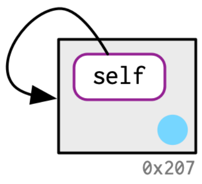
```
e1$c <- 4
e2$c
#> [1] 4
```



“ ” . Section 10.2.4 . Chapter 14
, R6 .

, .

```
e <- rlang::env()
e$self <- e

ref(e)
#> o [1:0x18631cf8] <env>
#> \-self = [1:0x18631cf8]
```



!

2.5.3 Exercises

1. .

```
x <- list()
x[[1]] <- x
```

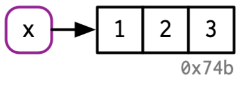
2. , ‘bench’ . ?

3. `tracemem()` ?

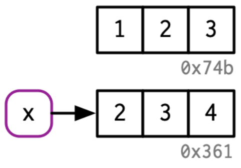
2.6 Unbinding and the garbage collector

.

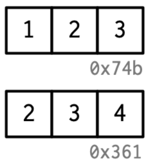
```
x <- 1:3
```



```
x <- 2:4
```



```
rm(x)
```



. , . ? **garbage collector** .(GC .) GC R , , .

R **(tracing)** GC . , .(, .) GC modify-in-place . , .(@ …)

GC R , . , GC . , . GC `gcinfo(TRUE)` , GC .

`gc()` garbage collection . `gc()` . `gc()` R , .

```r
gc()
#>          used (Mb) gc trigger (Mb) max used (Mb)
#> Ncells  581300 31.1    1245795 66.6  1245795 66.6
#> Vcells 1083544  8.3    8388608 64.0  2191249 16.8
```

`lobstr::mem_used()` `gc()` , .

```r
mem_used()
#> 41,258,096 B
```

, .

1. R R .
2. R . . R , OS .
3. R . . (fragmentation) .

2.7 Quiz answers

1. non-syntactic backticks(`) .

```r
df <- data.frame(runif(3), runif(3))
names(df) <- c(1, 2)

df$`3` <- df$`1` + df$`2`
```

2. 8MB .

```
x <- runif(1e6)
y <- list(x, x, x)
obj_size(y)
#> 8,000,128 B
```

3. `a` `b` `b[[1]] <- 10` .

2.8 Summary

- , .
- R , .
- , .
- modified-in-place .

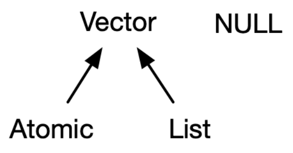
Chapter 3

Vectors

3.1 Introduction

Chapter R , . , . Chapter , . , R .

, (atomic) . , , . , `NULL` , 0 . . Chapter .



_ (attribute)_ , . . *(dimension)* (matrices) (array) , *(class)* S3 . S3 Chapter 13 , S3 (factors), (date and times), (data frames), (tibbles) . 2D , R .

Quiz

Chapter . , Chapter . Section 3.8 .

1. atomic ? ?

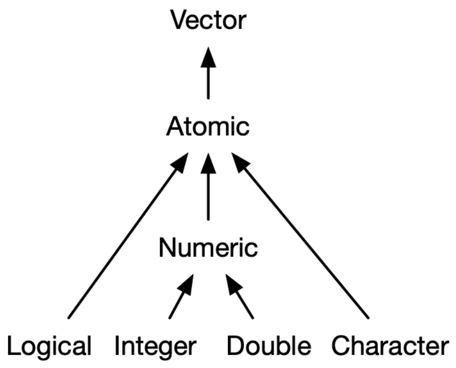
2. (attributes) ? ?
3. atomic ? ?
4. ? ?
5. (tibbles) ?

Outline

- Section 3.2 (logical), (integer), (double), (character) atomic . R .
- Section 3.3 R (attributes) . (names), (dimensions), (class) .
- Section 3.4 atomic . (factors), (dates), - (date-times), (durations) .
- Section 3.5 . atomic , . . .
- Section 3.6 . . , .

3.2 Atomic vectors

(logical), (integer), (double), (character) atomic . (numeric) . (complex) Raw . , , raw binary .



3.2.1 Scalars

, __scalar__ .

- (TRUE FALSE), (T F) .
- 10 (0.1234), (1.23e4), 16 (0xcafe) . . Inf, -Inf, NaN(not a number) , .
- L .(1234L, 1e4L, 0xcafeL) .
- "("hi") '('bye') . \ , ?Quotes .

3.2.2 Making longer vectors with c()

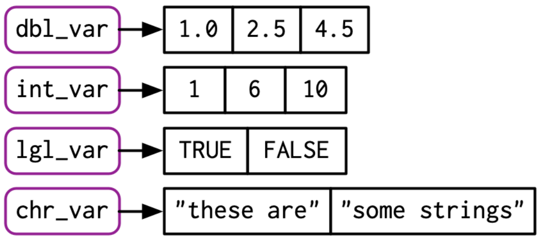
, c() .

```
lgl_var <- c(TRUE, FALSE)
int_var <- c(1L, 6L, 10L)
dbl_var <- c(1, 2.5, 4.5)
chr_var <- c("these are", "some strings")
```

atomic , c() atomic .

```
c(c(1, 2), c(3, 4))
#> [1] 1 2 3 4
```

. .



typeof() , length() .

```
typeof(lgl_var)
#> [1] "logical"
typeof(int_var)
#> [1] "integer"
```

```
typeof(dbl_var)
#> [1] "double"
typeof(chr_var)
#> [1] "character"
```

3.2.3 Missing values

R , NA(not applicable) . .
.

```
NA > 5
#> [1] NA
10 * NA
#> [1] NA
!NA
#> [1] NA
```

, .

```
NA ^ 0
#> [1] 1
NA | TRUE
#> [1] TRUE
NA & FALSE
#> [1] FALSE
```

.

```
x <- c(NA, 5, NA, 10)
x == NA
#> [1] NA NA NA NA
```

, . is.na() .

```
is.na(x)
#> [1]  TRUE FALSE  TRUE FALSE
```

3.2.4 Testing and coercion

is.*() _ _ . . is.logical(), is.integer(), is.double(), is.character() ,

. is.vector(), is.atomic(), is.numeric() . atomic
. .

atomic vectors , . . , __ (coerced)__ .

: → → →

, .

```
str(c("a", 1))
#>  chr [1:2] "a" "1"
```

Coercion . (+, log, abs) . coercion
, TRUE 1 FALSE 0 .

```
x <- c(FALSE, FALSE, TRUE)
as.numeric(x)
#> [1] 0 0 1

# Total number of TRUEs
sum(x)
#> [1] 1

# Proportion that are TRUE
mean(x)
#> [1] 0.333
```

as.logical(), as.integer(), as.double(), as.character() as.*()
. .

```
as.integer(c("1", "1.5", "a"))
#> Warning:           NA
#> [1]   1   1 NA
```

3.2.5 Exercises

1. raw complex scalar ?
2. coercion .

```
c(1, FALSE)
c("a", 1)
c(TRUE, 1L)
```

3. 1 =="1" ? -1 < FALSE ? "one" < 2 ?
4. NA ? ?(: c(FALSE, NA_character_))
5. is.atomic(), is.numeric(), is.vector() ?

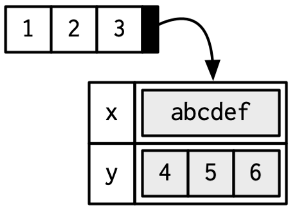
3.3 Attributes

atomic , , , - . atomic (attributes) . Section , (dim) . Section , , - S3 .

3.3.1 Getting and setting

. attr() , attributes() structure() .

```r
a <- 1:3
attr(a, "x") <- "abcdef"
attr(a, "x")
#> [1] "abcdef"

attr(a, "y") <- 4:6
str(attributes(a))
#> List of 2
#>  $ x: chr "abcdef"
#>  $ y: int [1:3] 4 5 6

# Or equivalently
a <- structure(
  1:3,
  x = "abcdef",
  y = 4:6
)
str(attributes(a))
#> List of 2
#>  $ x: chr "abcdef"
#>  $ y: int [1:3] 4 5 6
```



. , .

```
attributes(a[1])
#> NULL
attributes(sum(a))
#> NULL
```

.

- **(names)**: .
- **(dim)**: dimensions , .

, Chapter 13 , S3 .

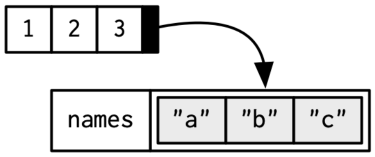
3.3.2 Names

.

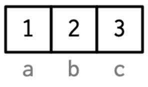
```
# When creating it:
x <- c(a = 1, b = 2, c = 3)

# By assigning a character vector to names()
x <- 1:3
names(x) <- c("a", "b", "c")

# Inline, with setNames():
x <- setNames(1:3, c("a", "b", "c"))
```

`attr(x, "names")` `names(x)` . `unname(x)` `names(x) <- NULL` .

, `x` .



, , .



, . R . , `""` `NA_character_` . , `names()` `NULL` .

3.3.3 Dimensions

`dim` 2 _ _ _ _ . , .
. Section 4.2.3 .

`matrix()` `array()` , `dim()` .

```r
# Two scalar arguments specify row and column sizes
a <- matrix(1:6, nrow = 2, ncol = 3)
a
#>      [,1] [,2] [,3]
#> [1,]    1    3    5
#> [2,]    2    4    6

# One vector argument to describe all dimensions
b <- array(1:12, c(2, 3, 2))
b
#> , , 1
#>
#>      [,1] [,2] [,3]
#> [1,]    1    3    5
#> [2,]    2    4    6
#>
#> , , 2
#>
#>      [,1] [,2] [,3]
#> [1,]    7    9   11
#> [2,]    8   10   12

# You can also modify an object in place by setting dim()
c <- 1:6
dim(c) <- c(3, 2)
c
#>      [,1] [,2]
#> [1,]    1    4
#> [2,]    2    5
#> [3,]    3    6
```

.

Vector	Matrix	Array
`names()`	`rownames()`, `colnames()`	`dimnames()`
`length()`	`nrow()`, `ncol()`	`dim()`
`c()`	`rbind()`, `cbind()`	`abind::abind()`
—	`t()`	`aperm()`
`is.null(dim(x))`	`is.matrix()`	`is.array()`

dim 1 , NULL . , 1 . , . , .(tapply()) , str() .

```
str(1:3)                   # 1d vector
#>  int [1:3] 1 2 3
str(matrix(1:3, ncol = 1)) # column vector
#>  int [1:3, 1] 1 2 3
str(matrix(1:3, nrow = 1)) # row vector
#>  int [1, 1:3] 1 2 3
str(array(1:3, 3))         # "array" vector
#>  int [1:3(1d)] 1 2 3
```

3.3.4 Exercises

1. setNames() unname() ? .
2. dim() 1 ? NROW() NCOL() ?
3. ? 1:5 ?

```
x1 <- array(1:5, c(1, 1, 5))
x2 <- array(1:5, c(1, 5, 1))
x3 <- array(1:5, c(5, 1, 1))
```

4. structure() .

```
structure(1:5, comment = "my attribute")
#> [1] 1 2 3 4 5
```

comment . ? ? ?(: help)

3.4 S3 atomic vectors

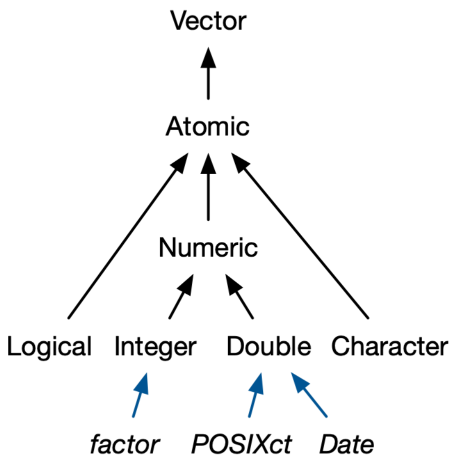
S3 class . **S3** . **(generic)** , . S3 , . Chapter 13 S3 S3 .

Section , R S3 .

- **(factor)**
- **(Date)** ( )

- **POSIXct** - ( )

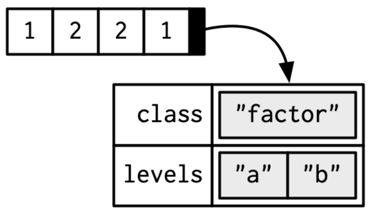
- **difftime**



3.4.1 Factors

. . .
“ ” class , levels .

```
x <- factor(c("a", "b", "b", "a"))
x
#> [1] a b b a
#> Levels: a b

typeof(x)
#> [1] "integer"
attributes(x)
#> $levels
#> [1] "a" "b"
#>
#> $class
#> [1] "factor"
```



, . , .

```
sex_char <- c("m", "m", "m")
sex_factor <- factor(sex_char, levels = c("m", "f"))

table(sex_char)
#> sex_char
#> m
#> 3
table(sex_factor)
#> sex_factor
#> m f
#> 3 0
```

(Ordered) . , levels (low, medium, high) .( )

```
grade <- ordered(c("b", "b", "a", "c"), levels = c("c", "b", "a"))
grade
#> [1] b b a c
#> Levels: c < b < a
```

Base R (`read.csv()`, `data.frame()`) R , . , levels . levels . , `stringAsFactors = FALSE` , " " . , Roger Peng *stringsAsFactors: An unauthorized biography* , Thomas Lumley *stringsAsFactors = <sigh>* .

, . . `gsub()` `grepl()` . `nchar()` , `c()` . , , .

3.4.2 Dates

(double) . "Date" class .

```r
today <- Sys.Date()

typeof(today)
#> [1] "double"
attributes(today)
#> $class
#> [1] "Date"
```

1970-01-01 .

```r
date <- as.Date("1970-02-01")
unclass(date)
#> [1] 31
```

3.4.3 Dates-times

Base R - . POSIXct POSIXlt . “POSIX” Portable Operating System Interface . “ct” calendar time(C `time_t`) , “lt” local time(C `struct tm`) . `POSIXct` . , atomic vector , . POSIXct 1970-01-01 .

```r
now_ct <- as.POSIXct("2018-08-01 22:00", tz = "UTC")
now_ct
#> [1] "2018-08-01 22:00:00 UTC"

typeof(now_ct)
#> [1] "double"
attributes(now_ct)
#> $class
#> [1] "POSIXct" "POSIXt"
#>
#> $tzone
#> [1] "UTC"
```

`tzone` - . . .

```r
structure(now_ct, tzone = "Asia/Tokyo")
#> [1] "2018-08-02 07:00:00 JST"
structure(now_ct, tzone = "America/New_York")
#> [1] "2018-08-01 18:00:00 EDT"
structure(now_ct, tzone = "Australia/Lord_Howe")
#> [1] "2018-08-02 08:30:00 +1030"
```

```
structure(now_ct, tzone = "Europe/Paris")
#> [1] "2018-08-02 CEST"
```

3.4.4 Durations

- Durations (difftimes) . Difftimes ,
units .

```
one_week_1 <- as.difftime(1, units = "weeks")
one_week_1
#> Time difference of 1 weeks

typeof(one_week_1)
#> [1] "double"
attributes(one_week_1)
#> $class
#> [1] "difftime"
#>
#> $units
#> [1] "weeks"

one_week_2 <- as.difftime(7, units = "days")
one_week_2
#> Time difference of 7 days

typeof(one_week_2)
#> [1] "double"
attributes(one_week_2)
#> $class
#> [1] "difftime"
#>
#> $units
#> [1] "days"
```

3.4.5 Exercises

1. `table()` ? , ? ?

2. levels ?

```
f1 <- factor(letters)
levels(f1) <- rev(levels(f1))
```

3. ? f2 f3 f1 ?

```
f2 <- rev(factor(letters))

f3 <- factor(letters, levels = rev(letters))
```

3.5 Lists

atomic . . , . Section 2.3.3 , ( ) _ _ .

3.5.1 Creating

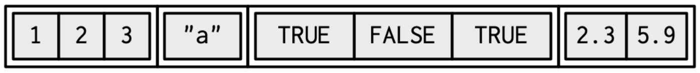
list() .

```
l1 <- list(
  1:3,
  "a",
  c(TRUE, FALSE, TRUE),
  c(2.3, 5.9)
)

typeof(l1)
#> [1] "list"

str(l1)
#> List of 4
#>  $ : int [1:3] 1 2 3
#>  $ : chr "a"
#>  $ : logi [1:3] TRUE FALSE TRUE
#>  $ : num [1:2] 2.3 5.9
```

, . .

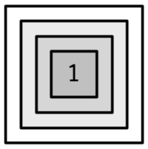
```
lobstr::obj_size(mtcars)
#> 7,208 B

l2 <- list(mtcars, mtcars, mtcars, mtcars)
lobstr::obj_size(l2)
#> 7,288 B
```

, . , .



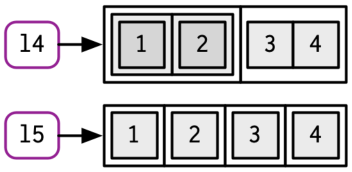
(reculsive) , . atomic .

```
l3 <- list(list(list(1)))
str(l3)
#> List of 1
#>  $ :List of 1
#>   ..$ :List of 1
#>   .. ..$ : num 1
```



`c()` . atomic , `c()` . `list()` `c()`
.

```
l4 <- list(list(1, 2), c(3, 4))
l5 <- c(list(1, 2), c(3, 4))
str(l4)
#> List of 2
#>  $ :List of 2
#>   ..$ : num 1
#>   ..$ : num 2
#>  $ : num [1:2] 3 4
str(l5)
#> List of 4
#>  $ : num 1
#>  $ : num 2
#>  $ : num 3
#>  $ : num 4
```



3.5.2 Testing and coercion

typeof() list . is.list() , as.list() .

```
list(1:3)
#> [[1]]
#> [1] 1 2 3
as.list(1:3)
#> [[1]]
#> [1] 1
#>
#> [[2]]
#> [1] 2
#>
#> [[3]]
#> [1] 3
```

unlist() atomic . , , c() .

3.5.3 Matrices and arrays

atomic . - - .

```
l <- list(1:3, "a", TRUE, 1.0)
dim(l) <- c(2, 2)
l
#>      [,1]      [,2]
#> [1,] Integer,3 TRUE
#> [2,] "a"       1

l[[1, 1]]
#> [1] 1 2 3
```

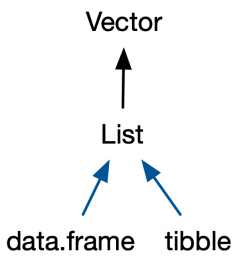
, grid-like . , grid , grid 3D .

3.5.4 Exercises

1. atomic .
2. atomic unlist() ? as.vector() ?
3. - , c() unlist() .

3.6 Data frames and tibbles

S3 , (tibbles) .



R , . ( )names, row.names() "data.frame" .

```r
df1 <- data.frame(x = 1:3, y = letters[1:3])
typeof(df1)
#> [1] "list"

attributes(df1)
#> $names
#> [1] "x" "y"
#>
#> $class
#> [1] "data.frame"
#>
#> $row.names
#> [1] 1 2 3
```

, , . , .

- rownames() colnames() . names() .
- nrow() ncol() . length() .

R . R .

20 , R , .

(tibble) . Tibbles are designed to be (as much as possible) drop-in replacements for data frames that fix those frustrations. , ' (lazy)' ' (surly)' . Section .

. , tbl_df .
. .

```
library(tibble)

df2 <- tibble(x = 1:3, y = letters[1:3])
typeof(df2)
#> [1] "list"

attributes(df2)
#> $names
#> [1] "x" "y"
#>
#> $row.names
#> [1] 1 2 3
#>
#> $class
#> [1] "tbl_df"     "tbl"        "data.frame"
```

3.6.1 Creating

data.frame() .

```
df <- data.frame(
  x = 1:3,
  y = c("a", "b", "c")
)
str(df)
#> 'data.frame':    3 obs. of  2 variables:
#>  $ x: int  1 2 3
#>  $ y: Factor w/ 3 levels "a","b","c": 1 2 3
```

. stringAsFactors = FALSE .

```
df1 <- data.frame(
  x = 1:3,
  y = c("a", "b", "c"),
  stringsAsFactors = FALSE
)
str(df1)
#> 'data.frame':    3 obs. of  2 variables:
#>  $ x: int  1 2 3
#>  $ y: chr  "a" "b" "c"
```

. .(‘ ’)

```
df2 <- tibble(
  x = 1:3,
  y = c("a", "b", "c")
)
str(df2)
#> Classes 'tbl_df', 'tbl' and 'data.frame':    3 obs. of  2 variables:
#>  $ x: int  1 2 3
#>  $ y: chr  "a" "b" "c"
```

non-syntactic (`check.names = FALSE`), .(non-syntactic `)

```
names(data.frame(`1` = 1))
#> [1] "X1"

names(tibble(`1` = 1))
#> [1] "1"
```

( ) , `data.frame()` `tibble()` . , 1 .

```
data.frame(x = 1:4, y = 1:2)
#>   x y
#> 1 1 1
#> 2 2 2
#> 3 3 1
#> 4 4 2
data.frame(x = 1:4, y = 1:3)
#> Error in data.frame(x = 1:4, y = 1:3): arguments imply differing
#> number of rows: 4, 3

tibble(x = 1:4, y = 1)
#> # A tibble: 4 x 2
#>       x     y
#>   <int> <dbl>
#> 1     1     1
#> 2     2     1
#> 3     3     1
#> 4     4     1
tibble(x = 1:4, y = 1:2)
#> Error: Tibble columns must have consistent lengths, only values of
#> length one are recycled:
```

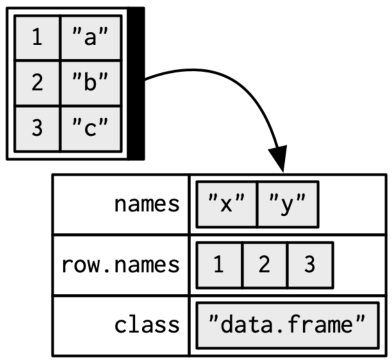
```
#> * Length 2: Column `y`
#> * Length 4: Column `x`
```

, tibble() .

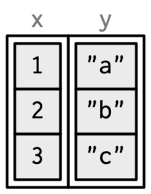
```
tibble(
  x = 1:3,
  y = x * 2
)
#> # A tibble: 3 x 2
#>       x     y
#>   <int> <dbl>
#> 1     1     2
#> 2     2     4
#> 3     3     6
```

(.)

, ,



. .



3.6.2 Row names

. .

```
df3 <- data.frame(
  age = c(35, 27, 18),
  hair = c("blond", "brown", "black"),
  row.names = c("Bob", "Susan", "Sam")
)
df3
#>       age  hair
#> Bob    35 blond
#> Susan  27 brown
#> Sam    18 black
```

`rownames()` , .

```
rownames(df3)
#> [1] "Bob"   "Susan" "Sam"

df3["Bob", ]
#>     age  hair
#> Bob  35 blond
```

2 . ( ) , ( ) . . . . . ’ (transpose)’ . . .( ) . .

, .

- . , . , .
- . . , ’ ’ , .
- . ( ) . .

```
df3[c(1, 1, 1), ]
#>       age  hair
#> Bob    35 blond
#> Bob.1  35 blond
#> Bob.2  35 blond
```

. rownames_to_column()
as_tibble() rownames .

```
as_tibble(df3, rownames = "name")
#> # A tibble: 3 x 3
#>   name    age hair 
#>   <chr> <dbl> <fct>
#> 1 Bob      35 blond
#> 2 Susan    27 brown
#> 3 Sam      18 black
```

3.6.3 Printing

. . dplyr
.

```
dplyr::starwars
#> # A tibble: 87 x 13
#>    name  height  mass hair_color skin_color eye_color birth_year
#>    <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl>
#>  1 Luke…    172    77 blond      fair       blue            19  
#>  2 C-3PO    167    75 <NA>       gold       yellow         112  
#>  3 R2-D2     96    32 <NA>       white, bl… red             33  
#>  4 Dart…    202   136 none       white      yellow          41.9
#>  5 Leia…    150    49 brown      light      brown           19  
#>  6 Owen…    178   120 brown, gr… light      blue            52  
#>  7 Beru…    165    75 brown      light      blue            47  
#>  8 R5-D4     97    32 <NA>       white, red red             NA  
#>  9 Bigg…    183    84 black      light      brown           24  
#> 10 Obi-…    182    77 auburn, w… fair       blue-gray       57  
#> # … with 77 more rows, and 6 more variables: gender <chr>,
#> #   homeworld <chr>, species <chr>, films <list>, vehicles <list>,
#> #   starships <list>
```

- 10 . .
- , .
- , ( ) .
- , , .

3.6.4 Subsetting

Chapter 4 , 1D ( ) , 2D ( ) .

, .

- df[, vars] , vars , . df[, vars, drop = FALSE] , [.
- df$x x , x . x , df$x NULL . .

. [, $.(surly)

```
df1 <- data.frame(xyz = "a")
df2 <- tibble(xyz = "a")

str(df1$x)
#>  Factor w/ 1 level "a": 1
str(df2$x)
#> Warning: Unknown or uninitialised column: 'x'.
#>  NULL
```

[(df[, "col"]) . df[["col"]] . , .

3.6.5 Testing and coercing

, is.data.frame() .

```
is.data.frame(df1)
#> [1] TRUE
is.data.frame(df2)
#> [1] TRUE
```

, , is_tibble() .

```
is_tibble(df1)
#> [1] FALSE
is_tibble(df2)
#> [1] TRUE
```

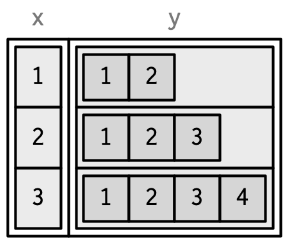
as.data.frame() , as_tibble() .

3.6.6 List columns

, . .
. , . *R for Data Science*, http:
//r4ds.had.co.nz/many-models.html “Many Models” chapter .

, , `I()` .

```
df <- data.frame(x = 1:3)
df$y <- list(1:2, 1:3, 1:4)

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



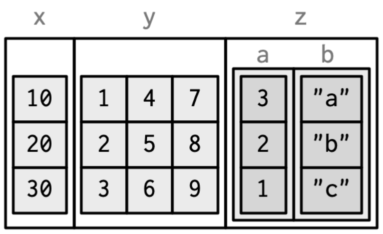
. `tibble()` , .

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

3.6.7 Matrix and data frame columns

, .(: length()
NROW()) , I() .

```
dfm <- data.frame(
  x = 1:3 * 10
)
dfm$y <- matrix(1:9, nrow = 3)
dfm$z <- data.frame(a = 3:1, b = letters[1:3], stringsAsFactors = FALSE)

str(dfm)
#> 'data.frame':	3 obs. of  3 variables:
#>  $ x: num  10 20 30
#>  $ y: int [1:3, 1:3] 1 2 3 4 5 6 7 8 9
#>  $ z:'data.frame':	3 obs. of  2 variables:
#>   ..$ a: int  3 2 1
#>   ..$ b: chr  "a" "b" "c"
```



. . .

```
dfm[1, ]
#>    x y.1 y.2 y.3 z.a z.b
#> 1 10   1   4   7   3   a
```

3.6.8 Exercises

1. 0 ?

2. ?

3. `df` `t(df)` `t(t(df))` ? .

4. `as.matrix()` ? `data.matrix()` ?

3.7 NULL

Chapter , . NULL . , 0 , .

```
typeof(NULL)
#> [1] "NULL"

length(NULL)
#> [1] 0

x <- NULL
attr(x, "y") <- 1
#> Error in attr(x, "y") <- 1: NULL
```

is.null() NULL .

```
is.null(NULL)
#> [1] TRUE
```

NULL .

- (0) . , c() NULL , NULL .

```
c()
#> NULL
```

- . NULL , , .(Section 6.5.3) NA _ _ .

SQL , NULL , R . NULL R NA .

3.8 Quiz answers

1. atomic , , , . raw .
2. . attr(x, "y") attr(x, "y") <- value . attribute() .
3. ( ) , atomic . , , .
4. - . df$x <- matrix() data.frame(x = I(matrix())) I() , .
5. , , .

3.9 Summary

- atomic (same type) (no matter)
- attributes
- Matrix, Array = Vector + < dimensions >
- Factor = Integer + < factor class, levels >
- Dataframe = List + < data.frame class, names >
- Tibble : strings as factors + non-syntactic variable name + recycling rule + varible references
- NULL : a vector is absent / NA : an element of a vector is absent