Advanced R in Korean

Jsang

2020-02-08

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` ?

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

2. , `y` ?

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

3. `a` ?

```
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 .

```
library(lobstr)
```

2.2 Binding basics

.

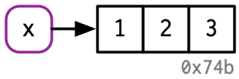
```
x <- c(1, 2, 3)
```

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

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

, . .

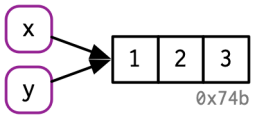
, .



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()` ?

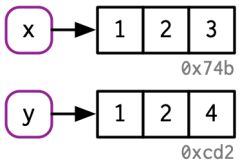
5. syntactic . `.123e1` syntactic ? `?makes.names` .

2.3 Copy-on-modify

. x y . y .

```
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() . , .

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

tracemem() , .

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

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

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

untracemem(x)
```

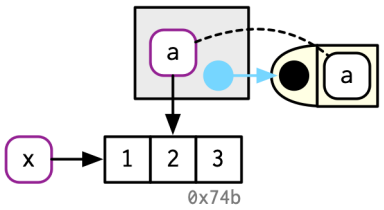
untracemem() tracemem() , .

2.3.2 Function calls

. .

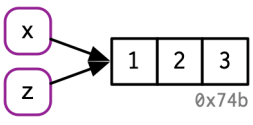
```
f <- function(a) {
  a
}

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 .

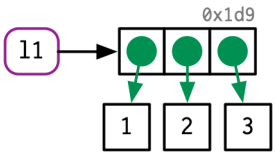


2.3.3 Lists

(,) . . , .

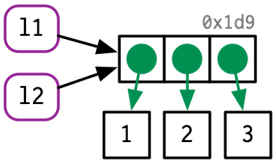
```
l1 <- list(1, 2, 3)
```

, , ( ) .

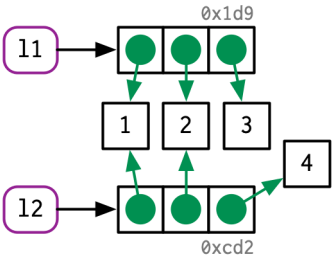


.

```
l2 <- l1
```




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



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

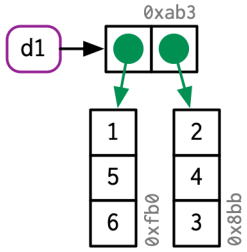
, `lobstr::ref()` . `ref()` ID ,
.

```
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

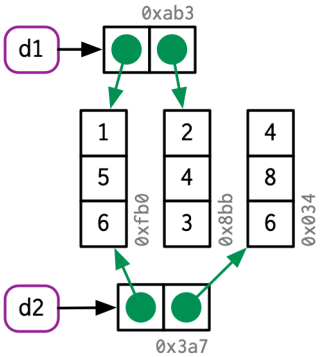
. copy-on-modify . .

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



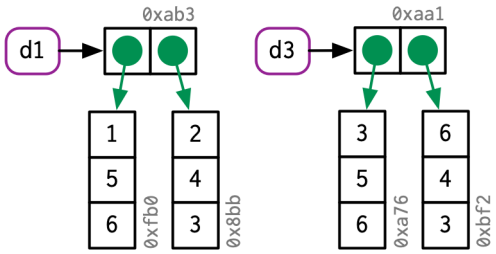
, . .

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



, . .

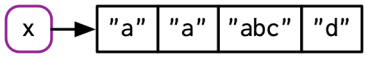
```
d3 <- d1
d3[1, ] <- d3[1, ] * 3
```



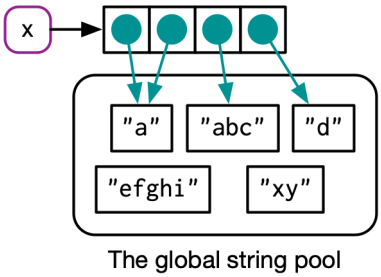
2.3.5 Character vectors

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

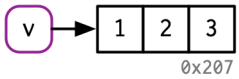
, R . .

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

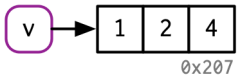
2.5.1 Objects with a single binding

, 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]:
```

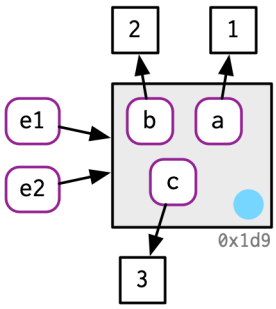
, . , Chapter 25 C++
.

2.5.2 Environments

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

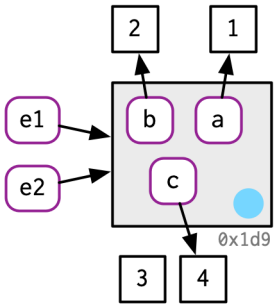
. `e1` `e2` .

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



, modified in place .

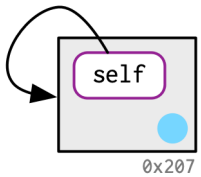
```
e1$c <- 4
e2$c
#> [1] 4
```



“ ” . 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
```

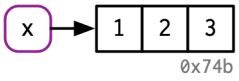
2. , 'bench' . ?

3. `tracemem()` ?

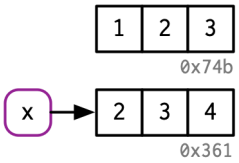
2.6 Unbinding and the garbage collector

.

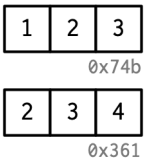
```
x <- 1:3
```




```
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 , .

```
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()` , .

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

, .

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

2.7 Quiz answers

1. non-syntactic backticks(`) .

```
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 .

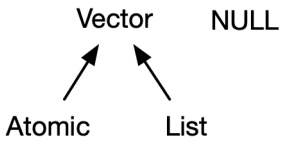
Chapter 3

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 ? ?

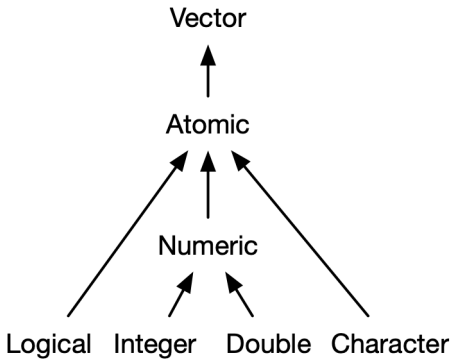
2. (attributes) ? ?
3. atomic ? ?
4. ? ?
5. (tibbles) ?

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()

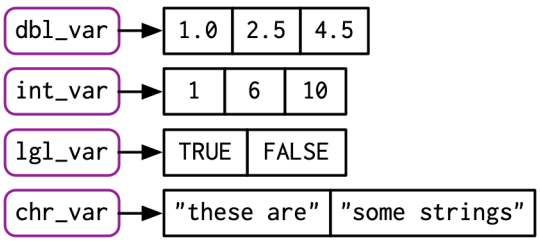
, c() .

```
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()` ?

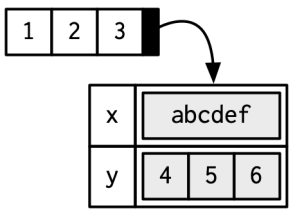
3.3 Attributes

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

3.3.1 Getting and setting

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

```
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 .

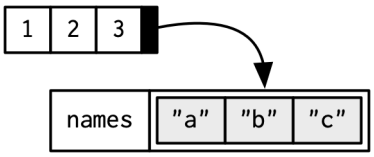
3.3.2 Names

.

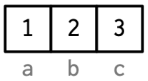
```
# When creating it:
x <- c(a = 1, b = 2, c = 3)

# 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()` .

```
# 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

3.5 Lists

3.6 Data frames and tibbles

3.7 NULL

3.8 Quiz answers

3.9 Summary