

1. R Foundations - Atomic Vectors

Data Science for OR - J. Duggan

R

- R's mission is to enable the best and most thorough exploration of data possible (Chambers 2008).
- It is a dialect of the S language, developed at Bell Laboratories
- ACM noted that S “will forever alter the way people analyze, visualize, and manipulate data”

```
v <- 1:10
```

```
v
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
summary(v)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.00   3.25   5.50   5.50   7.75   10.00
```


Atomic Vectors - Examples

```
dbl_var <- c(2.9, 3.1, 4.8)
typeof(dbl_var)
```

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

```
int_var <- c(0L, 1L, 2L)
typeof(int_var)
```

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

```
log_var <- c(TRUE, TRUE, FALSE, T, F)
typeof(log_var)
```

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

```
str_var <- c("Dublin", "London", "Edinburgh")
typeof(str_var)
```

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

str() function useful

```
str(dbl_var)
```

```
##  num [1:3] 2.9 3.1 4.8
```

```
str(int_var)
```

```
##  int [1:3] 0 1 2
```

```
str(log_var)
```

```
##  logi [1:5] TRUE TRUE FALSE TRUE FALSE
```

```
str(str_var)
```

```
##  chr [1:3] "Dublin" "London" "Edinburgh"
```

Creating Sequences : and seq() function

```
v1 <- 1:10
```

```
v1
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
v2 <- 10:20
```

```
v2
```

```
## [1] 10 11 12 13 14 15 16 17 18 19 20
```

```
v3 <- seq(20, 30, by=1)
```

```
v3
```

```
## [1] 20 21 22 23 24 25 26 27 28 29 30
```

Creating Vectors of fixed size (in advance)

```
v1 <- vector(mode="numeric", length=20)
```

```
v1
```

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

```
v2 <- vector(mode="logical", length=5)
```

```
v2
```

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

Coercion of atomic vectors

- All elements of an atomic vector MUST be of the same type
- When different type are combined, they will be coerced into the most flexible types

	logical	integer	numeric	character
logical	logical	integer	numeric	character
integer	integer	integer	numeric	character
numeric	numeric	numeric	numeric	character
character	character	character	character	character

Coercion Examples

```
v1 <- c(10, 20, TRUE)
v1
```

```
## [1] 10 20 1
```

```
typeof(v1)
```

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

```
v2 <- c(10, 20, "True")
v2
```

```
## [1] "10" "20" "True"
```

```
typeof(v2)
```

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

Challenge 1.1

Determine the types for each of the following vectors

```
v1 <- c(1L, T, FALSE)
v2 <- c(1L, T, FALSE, 2)
v3 <- c(T, FALSE, 2, "FALSE")
v4 <- c(2L, "FALSE")
v5 <- c(0L, 1L, 2.11)
```

Subsetting Atomic Vectors

- Subsetting data is a key activity in data science
- R's subsetting operators are powerful and fast
- For atomic vectors, the operator `[` is used
- In R, the index for a vector starts at 1

```
x <- c( 2.1, 4.2, 3.3, 5.4)
x
```

```
## [1] 2.1 4.2 3.3 5.4
```

```
x[1]
```

```
## [1] 2.1
```

```
x[c(1,4)]
```

```
## [1] 2.1 5.4
```

Subsetting Vectors - (1) Positive Integer

Positive integers return elements at the specified position

```
x <- 1:10
```

```
x
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
x[5]
```

```
## [1] 5
```

```
x[8:10]
```

```
## [1] 8 9 10
```

Subsetting Vectors - (2) Negative Integer

Negative integers omit elements at specified positions

```
x <- 1:10
```

```
x
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
x[-5]
```

```
## [1] 1 2 3 4 6 7 8 9 10
```

```
x[-(8:10)]
```

```
## [1] 1 2 3 4 5 6 7
```

```
x[-(2:10)]
```

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

Subsetting Vectors - (3) Logical Vectors

- Select elements where the corresponding logical value is TRUE.
- This approach supports recycling

```
x <- 1:5
```

```
x
```

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

```
x[c(F,T,T,T,T)]
```

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

```
x[c(F,T)]
```

```
## [1] 2 4
```

Logical Vectors - Can be formed with logical expressions

```
x <- 1:5
```

```
x
```

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

```
lx <- x < 2
```

```
lx
```

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

```
x[lx]
```

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

```
x[x>2]
```

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


Subsetting Vectors - (4) Using character vectors

Return elements with matching names

```
x <- 1:5  
names(x) <- c("a", "b", "c", "d", "e")  
x
```

```
## a b c d e  
## 1 2 3 4 5
```

```
x["a"]
```

```
## a  
## 1
```

```
x[c("a", "e")]
```

```
## a e  
## 1 5
```

Challenge 1.2

- Create an R vector of squares of 1 to 10
- Find the minimum
- Find the maximum
- Find the average
- Subset all those values greater than the average

Vectorisation

- A powerful feature of R is that it supports vectorisation
- Functions can operate on every element of a vector, and return the results of each individual operation in a new vector.

```
x <- c(1,4,9,16,25)
x
```

```
## [1] 1 4 9 16 25
```

```
y <- sqrt(x)
y
```

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

Vectorisation

Input Vector



Output Vector

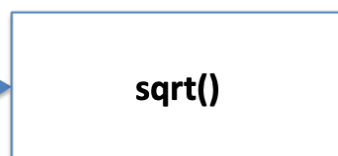


Figure 2: Vectorisation in R

Vectorised if/else

Vectors can also be processed using the vectorized `ifelse(b,u,v)` function, which accepts a boolean vector `b` and allocates the element-wise results to be either `u` or `v`.

```
v1 <- 1:5

ans <- ifelse(v1 %% 2 == 0, "Even", "Odd")
ans

## [1] "Odd"  "Even" "Odd"  "Even" "Odd"
```

Sample Function

`sample` takes a sample of the specified size from the elements of `x` using either with or without replacement.

Usage

```
sample(x, size, replace = FALSE, prob = NULL)
```

```
sample.int(n, size = n, replace = FALSE, prob = NULL)
```

Arguments

- | | |
|----------------------|---|
| <code>x</code> | Either a vector of one or more elements from which to choose, or a positive integer. See ‘Details.’ |
| <code>n</code> | a positive number, the number of items to choose from. See ‘Details.’ |
| <code>size</code> | a non-negative integer giving the number of items to choose. |
| <code>replace</code> | Should sampling be with replacement? |
| <code>prob</code> | A vector of probability weights for obtaining the elements of the vector being sampled. |

Figure 3: Sample function in Base R

```
s <- sample(c("Y", "N"), 10, prob=c(.2, .8), repl=T)
```

NA Symbol in R (Not available)

- In a project of any size, data is likely to be incomplete due to
 - Missed survey questions
 - Faulty equipment
 - Improperly coded data
- In R, missing data is represented by the symbol NA

```
x <- 1:5  
x[3] <- NA  
x
```

```
## [1] 1 2 NA 4 5
```

```
sum(x)
```

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

```
sum(x, na.rm=TRUE)
```

```
## [1] 12
```

Testing for NA? Need is.na() function

- The function is.na() indicates which elements are missing
- Returns a logical vector, the same size as the input vector

```
x
```

```
## [1] 1 2 NA 4 5
```

```
is.na(x)
```

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

```
which(is.na(x)) # get the location of NA
```

```
## [1] 3
```

```
x[!is.na(x)] # Exclude all NAs from result
```

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

Summary

- Atomic vectors, key type in R
- All elements are the same type (coercion)
- Different ways to filter, including logical vectors
- `is.na()` to check for symbol NA (not available)
- Vectors support vectorise operations. e.g., **`sqrt(1:10)`**