Vectors and Lists

David Gerard

2019-03-08

Learning Objectives

- Manipulating Vectors and Lists using base R syntax.
- Chapter 20 of RDS.

Vector Basics

- We'll use just a few tidyverse functions.

```
library(tidyverse)
```

- Recall the vector material from the R Basics Worksheet.
- A **vector** is a sequence of elements of the same type.
- "type" = `integer`, `double`, `character`, `logical`, `factor`, or `date`.
- `numeric` is used to represent vectors that are either integers or doubles.
- Use `c()` to create vectors.
- Use `typeof()` to see the type of vector and `is_*()` to check the type of vector.
- Double:

```
x <- c(1, 10, 2)
typeof(x)
```

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

```
is_double(x) ## From purrr package
```

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

- Integer (use `L` to tell R that a number is an integer):

```
x <- c(1L, 10L, 2L)
typeof(x)
```

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

```
is_integer(x) ## From purrr package
```

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

- Character:

```r
x <- c("hello", "good", "sir")
typeof(x)
```

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

```r
is_character(x) ## From purrr package
```

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

- Logical:

```r
x <- c(TRUE, FALSE, FALSE)
typeof(x)
```

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

```r
is_logical(x) ## From purrr package
```

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

- Factor: Factors are actually integers with extra attributes.

```r
x <- factor(c("A", "B", "B"))
typeof(x)
```

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

```r
is.factor(x)
```

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

```r
is_logical(x) ## From purrr package
```

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

- Dates: Dates are actually doubles with extra attributes.

```r
x <- lubridate::ymd(20150115, 20110630, 20130422)
typeof(x)
```

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

```r
lubridate::is.Date(x)
```

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

```r
is_double(x) ## From purrr package
```

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

- Each element of a vector can have a name

```r
x <- c(horse = 7, man = 1, dog = 8)
x
```

```
## horse   man   dog
##     7     1     8
```

- You can see and change the names with the `names()` function

```r
names(x)
```

```
## [1] "horse" "man"   "dog"
```

```r
names(x)[1] <- "cat"
x
```

```
## cat man dog
##   7   1   8
```

- Subset with brackets `[`

```r
x <- c("I", "like", "dogs")
x[2:3]
```

```
## [1] "like" "dogs"
```

```r
lvec <- c(TRUE, FALSE, TRUE)
x[lvec]
```

```
## [1] "I"    "dogs"
```

- Substitute while subsetting

```r
x[1] <- "You"
x
```

```
## [1] "You"  "like" "dogs"
```

```r
x[lvec] <- "We"
x
```

```
## [1] "We"   "like" "We"
```

- Subset with negative values to drop elements

```
x[-3]
```

```
## [1] "We"   "like"
```

- Subset a named vector with the name

```
x <- c(horse = 7, man = 1, dog = 8)
x["man"]
```

```
## man
##   1
```

- Two brackets `[[` only returns a single elements and drops the name.

```
x[3]
```

```
## dog
##   8
```

```
x[[3]]
```

```
## [1] 8
```

- **Exercise**: Consider the following vector:

```
x <- c(Yoshi = 10L,
       Mario = 31L,
       Luigi = 72L,
       Peach = 11L,
       Toad  = 38L)
```

Extract Yoshi and Peach from the above vector using:

1. Integer subsetting.
2. Negative integer subsetting.
3. Logical subsetting.
4. Name subsetting.

- **Exercise**: In the vector above, substitute Yoshi's number with `19L`.
- You are used to doing vectorized operations.

```
x <- c(1, 4, 1, 5)
x + 10
```

```
## [1] 11 14 11 15
```

- This is called "recycling", because what R is internally doing is thinking this is the same as

```
x + c(10, 10, 10, 10)
```

```
## [1] 11 14 11 15
```

- You can recycle non-scalars (but it's almost never a good idea):

```r
x + c(10, 20)
```

```
## [1] 11 24 11 25
```

```r
x + c(10, 20, 10, 20)
```

```
## [1] 11 24 11 25
```

Lists

- **Lists are vectors** whose elements can be of different types.
- Use `list()` to make a list.

```r
my_first_list <- list(x = "a", y = 1, z = c(1L, 2L, 3L), list("a", 1))
my_first_list
```

```
## $x
## [1] "a"
##
## $y
## [1] 1
##
## $z
## [1] 1 2 3
##
## [[4]]
## [[4]][[1]]
## [1] "a"
##
## [[4]][[2]]
## [1] 1
```

- The above is a *named* list that contains a character, a numeric, a logical vector, and another list. The internal list is *unnamed*.
- Use `str()` (for **str**ucture) to see the internal properties of a list.

```r
str(my_first_list)
```

```
## List of 4
##  $ x: chr "a"
##  $ y: num 1
##  $ z: int [1:3] 1 2 3
##  $  :List of 2
##   ..$ : chr "a"
##   ..$ : num 1
```

- Single brackets `[` return a sublist. You can use the same subsetting strategies as for vectors.

```
my_first_list[1:2]
```

```
## $x
## [1] "a"
##
## $y
## [1] 1
```

```
my_first_list["y"]
```

```
## $y
## [1] 1
```

- Double brackets `[[` returns a single list element.

```
my_first_list[[1]]
```

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

```
my_first_list[["z"]]
```

```
## [1] 1 2 3
```

- Use dollar signs `$` (just like in data frames) to extract named list elements.

```
my_first_list$z
```

```
## [1] 1 2 3
```

- You can remove elements of a list by substituting them with `NULL`.

```
str(my_first_list)
```

```
## List of 4
##  $ x: chr "a"
##  $ y: num 1
##  $ z: int [1:3] 1 2 3
##  $  :List of 2
##   ..$ : chr "a"
##   ..$ : num 1
```

```
my_first_list$x <- NULL
str(my_first_list)
```

```
## List of 3
##  $ y: num 1
##  $ z: int [1:3] 1 2 3
##  $  :List of 2
##   ..$ : chr "a"
##   ..$ : num 1
```

- **Exercise**: Consider the following list:

```
wedding <- list(venue = "chick-fil-a",
                guest = tribble(~name,     ~meal, ~age,
                                ##--------/------/-----
                                "Yoshi",   "V",   29L,
                                "Wario",   "C",   27L,
                                "Bowser",  "V",   34L,
                                "Luigi",   "C",   36L,
                                "Toad",    "B",   34L),
                bride = "Peach",
                groom = "Mario",
                date  = parse_date("11/10/2020", "%d/%m/%Y"))
```

a. Wario can't actually make it. Remove his row from the data frame.

b. Add a new named vector called `meal` where `V` is `"Vegetarian"`, `C` is `"Chicken"`, and `B` is `"Beef"`.

c. Extract the venue and the date from `wedding`. Use three different techniques do this.

d. `"chick-fil-a"` should be capitalized. Capitalize the first `"c"`.