Sequences and Repetition

Brett Presnell

Equally Spaced Sequences

R has a variety of ways of generating equally spaced sequences. Here are a few. See the documentation for *Sequence Generation* for more information.

```
1:10                                                    # Integers from 1 to 10
##  [1]  1  2  3  4  5  6  7  8  9 10

5:(-2)                                                  # Integers from 5 to -2
## [1]  5  4  3  2  1  0 -1 -2

seq(from = 3, to = 11, by = 2)      # Specifying start (3), end (11), and increment (2)
## [1]  3  5  7  9 11

seq(from = 1.5, to = 3.5, by = 0.25)                  # Specifying the increment (0.25)
## [1] 1.50 1.75 2.00 2.25 2.50 2.75 3.00 3.25 3.50

seq(from = -pi, to = pi, length = 7)        # Specifying the length of the sequence (7)
## [1] -3.141593 -2.094395 -1.047198  0.000000  1.047198  2.094395  3.141593
```

Repetition

Previously we created a vector containing two copies of `x` with the command `c(x, x)`. This is ok, but what if we wanted 6 copies of `x`?

```
x <- 1:5
x
## [1] 1 2 3 4 5

rep(x, 6)        # Equivalent to rep(x, times = 6)
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

- `rep()` has optional arguments `times`, `length.out`, and `each`.
  - `times` is an integer vector (of length 1 or of the length of `x`).
  - `length.out` is a single integer.
  - `each` is a single integer.
  - Defaults to `rep(x, times = 1, length.out = NA, each = 1)`.

- Here are some examples:

```
x <- c("blah", "woof")
x
## [1] "blah" "woof"

rep(x, times = 3)
## [1] "blah" "woof" "blah" "woof" "blah" "woof"

rep(x, each = 3)
## [1] "blah" "blah" "blah" "woof" "woof" "woof"

rep(x, length.out = 5)
## [1] "blah" "woof" "blah" "woof" "blah"

rep(x, times = c(3, 2))                    # Here times has the same length (2) as x.
## [1] "blah" "blah" "blah" "woof" "woof"
```

A Puzzle

Can you see that these should be the same?

```
rep(rep(1:3, times = 4), each = 2)
##  [1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3

rep(rep(1:3, each = 2),  times = 4)
##  [1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3
```

Because they are the same, the following is unambiguous:

```
rep(1:3, each = 2, times = 4)
##  [1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3
```

- More generally, if `each` is specified, then it is applied before `times` and `length.out`.