STATS 782

R Programming

Patterned Vectors

Generating Sequences

- The `:` operator can be used to generate sequences.

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

- The operator can accept `start` and `end` values which are non-integer.

```
> 1.5:7.5
[1] 1.5 2.5 3.5 4.5 5.5 6.5 7.5

> 1.5:7
[1] 1.5 2.5 3.5 4.5 5.5 6.5
```

More General Sequences

- The stepsize in sequences created by the `:` operator is always 1.
- The `seq` function makes it possible to produce more general sequences.
- Sequences are generated according to the values of the `from`, `to`, `by` and `length` arguments.
- Any three of these arguments can be used to create a sequence.
- In addition, the `along` argument can be used to produce the full set of indices for a vector.

Sequence Examples

```
> seq(0, 1, by = .1)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
[11] 1.0

> seq(0, 1, length = 11)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
[11] 1.0

> seq(0, by = .1, length = 11)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
[11] 1.0

> seq(to = 1, by = .1, length = 11)
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
[11] 1.0
```

Sequence Examples

```
> x = 1:10
> seq(along = x)
 [1]  1  2  3  4  5  6  7  8  9 10

> x = numeric(0)
> seq(along = x)
integer(0)

> 1:length(x)
[1] 1 0
```

Repeating Values

- The `rep` function makes it possible to repeat the values which occur in a vector.
- The first argument to `rep` gives the values to be repeated and the second specifies how many times to repeat them.
- The second argument can be a single value which specifes how many times to repeat the sequence of values in the first argument.
- It can also be a vector (with the same length as the first argument) which specifies how many times to repeat each value in the first argument.

Repetition Examples

The effect of scalar second argument.

```
> rep(1:4, 3)
 [1] 1 2 3 4 1 2 3 4 1 2 3 4
```

The effect of a vector second argument.

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

Note the first argument can be any kind of vector.

```
> rep(c("A", "B", "C"), 3)
[1] "A" "B" "C" "A" "B" "C" "A" "B" "C"
```

The `each` Argument

It is also possible to specify the repetitions with an optional scalar `each` argument which specifies how many times to repeat each value in the first argument.

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

This extra argument isn't strictly necessary because the same result can be obtained with the expression

```
> rep(1:4, rep(3, 4))
 [1] 1 1 1 2 2 2 3 3 3 4 4 4
```

but it makes an important special case simpler to type and easier to read.