Loops in R

Daniel E. Weeks

Department of Human Genetics
School of Public Health
University of Pittsburgh

Location

This slide set is in the Lectures repository in the 06_loops_flow_control folder.

Learning objectives

- To learn about the different types of loops in R
  - `for` loops
  - `while` loops
  - `repeat` loops
  - Vectorize operations

Don't repeat yourself

```
print("i = 1")
```

```
## [1] "i = 1"
```

```
print("i = 2")
```

```
## [1] "i = 2"
```

```
print("i = 3")
```

```
## [1] "i = 3"
```

Loops in R

A 'for' loop, iterating through the values of i:

```
for (i in 1:3) {
    print(paste("i =", i))
}
```

```
## [1] "i = 1"
## [1] "i = 2"
## [1] "i = 3"
```

A while loop

```
i <- 1
while (i < 4) {
    print(paste("i =", i))
    i <- i + 1
}
```

```
## [1] "i = 1"
## [1] "i = 2"
## [1] "i = 3"
```

A repeat loop

```
i <- 1
repeat {
    print(paste("i =", i))
    i <- i + 1
    if (i > 3)
        break
}
```

```
## [1] "i = 1"
## [1] "i = 2"
## [1] "i = 3"
```

Vectorize where possible

```r
a <- 1:10
b <- 1:10
res <- numeric(length = length(a))
for (i in seq_along(a)) {
    res[i] <- a[i] + b[i]
}
res
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

Example from here, CC-BY 4.0

Vectorize where possible

```
a
```

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

```
b
```

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

```
res2 <- a + b
res2
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
all.equal(res, res2)
```

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

Example from here, CC-BY 4.0

Options for loops in R

- `for`, `while`, `repeat` loops
- Vectorized operations
- Use `apply` family of functions
  - Optimized to apply functions over rows or columns of a data frame.

Loops

- Loops can be slow in R.
  - Avoid `for` loops where possible to vectorize instead.
  - Don't grow objects within a loop (e.g., with `rbind`)
  - Pre-allocate a 'results' object and fill it in as you go.
- Fine to use loops if you are careful.

Example loop

```
test <- function(k) {
    print(paste0("file", k, "_snp1.txt"))
    print(paste0("file", k, "_snp2.txt"))
    print("---------")
}
```

Example loop

```
for (i in 1:2) {
    test(k = i)
}
```

```
## [1] "file1_snp1.txt"
## [1] "file1_snp2.txt"
## [1] "----------"
## [1] "file2_snp1.txt"
## [1] "file2_snp2.txt"
## [1] "----------"
```

What type of loop when?

- If you know in advance how many loops you'll need, use a `for` loop
  - Example: looping over N=5 subjects.
- If you know the loop exit criterion but not how many loops, use a `while` loop or a `repeat` loop
  - `while`: tests the condition at the start of the loop
  - `repeat`: tests the condition at the end of the loop
  - Example: Repeatedly iterate a function optimization step until it converges.
- Caution: When using a `while` or a `repeat` loop, be sure the exit condition will be satisfied to avoid infinite loops.

Loop speed

```
system.time({
    a <- NULL
    for (i in 1:1e+07) a[i] <- i
})
```

```
##    user  system elapsed
##   1.392   0.111   1.518
```

Example from https://stackoverflow.com/questions/2908822/speed-up-the-loop-operation-in-r

Loop speed

```
system.time({
    a <- rep(1, 1e+07)
    for (i in 1:1e+07) a[i] <- i
})
```

```
##    user  system elapsed
##   0.272   0.008   0.285
```

```
system.time(a <- 1:1e+07)
```

```
##    user  system elapsed
##       0       0       0
```

Timing example

Why does pre-allocating a vector speed things up so much?

Here is a nice blog post illustrating different ways to do things in R: *link*

See also the excellent 'R Inferno': *link*

Another reason some loops are slow

See the 'Loops' section at the bottom of this page of 'Advanced R': *link.*

R memory usage

A data frame is stored in memory in column order, looped around.

Adding a row onto the end can be slow because you have to shift all the data around.

Useful links

https://swcarpentry.github.io/r-novice-inflammation/15-supp-loops-in-depth.html

https://www.datacamp.com/community/tutorials/tutorial-on-loops-in-r

Questions

What questions do you have?