For Loop explained in detail

by Jiajia Li

Research School of Biology

31 July 2025

Australian
National
University

For loop

Here, I will explain more in detail about the for loop we used to import .h5 files, create Seurat objects, and store all Seurat objects to a list.

First, we created a **vector** called "**sample_names**", and this vector contains **6 values** which are the names of our sample.

```
sample_names <- c("Rep1_ICBdT", "Rep3_ICBdT", "Rep5_ICBdT",
                  "Rep1_ICB", "Rep3_ICB", "Rep5_ICB")
```

`**c()**` is a function that **combines value into a Vector or List**.

Here, it combines 6 texts into a vector.

Data Types

In R, there are **many data types**. Here are some that are **often used**.

Type	Description	Example
numeric	Real numbers (double precision)	`3.14`, `42`, `-5.2`
integer	Whole numbers	`2L`, `100L`
character	Text/string	`"hello"`, `"R"`
logical	Boolean values	`TRUE`, `FALSE`

Texts wrapped in double quotes "" are characters.

Variables

I will introduce a new concept called **variables**.

In R and many other programming languages, we create different types of variables constantly to **store values and information**, so we can use it later.

Especially for longer values, such as a list of our sample names, we don't want to type all 6 sample names every time when we need to use it.

In this case, we can store them in a list and give this list a name, such as "sample_names".

The next time we use it we just type "sample_names" instead of the actual names of the 6 samples.

So, if we type "sample_names" in the R Console and hit Enter to run it, what do we get?

Variables

```
> sample_names
[1] "Rep1_ICBdT" "Rep3_ICBdT" "Rep5_ICBdT" "Rep1_ICB"
[5] "Rep3_ICB"   "Rep5_ICB"
>
```

If we run "sample_names" in the R console, it returns us what's been saved inside this variable.

Create an empty list

```
sample.data <- list()
```

List is a **flexible data structure** that can hold **elements of different types and structures**.

You can **name the elements** for easier access.

To create a list without name:

```
my_list <- list(42, "hello", TRUE)
```

```
> print(my_list)
[[1]]
[1] 42

[[2]]
[1] "hello"

[[3]]
[1] TRUE
```


Create a list with name

```
my_list <- list(name = "Alice", age = 30, married = TRUE)
```

```
> print(my_list)
$name
[1] "Alice"

$age
[1] 30

$married
[1] TRUE
```

To access the elements in the list, we can use double quotes "" and the name of that element.

```
> my_list[["name"]]
[1] "Alice"
> my_list[["age"]]
[1] 30
> my_list[["married"]]
[1] TRUE
```


Create a list with name

Or:

```
> my_list$name
[1] "Alice"
> my_list$age
[1] 30
> my_list$married
[1] TRUE
> 
```

If you only use **one pair of square brackets []**, you can still access the element, but with **both the name and actual value**.

```
> my_list["name"]
$name
[1] "Alice"

> my_list["age"]
$age
[1] 30
```


Syntax of For Loop

For loop is a type of loop statement is often used in programming.

In different programming languages, loop can be written in different syntax, but they all work with the same logic.

In R, it's written like:

```
for (variable in sequence) {
  # Code to execute each time
}
```

- `**variable**`: gets the value of each item in the sequence, one at a time.
- `**sequence**`: can be a vector, list, or range like `1:5`.

A simple for loop

This is an example of a simple for loop which loop through numbers.

```
> for (i in 1:5) {
+     print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
> 
```


A simple for loop

If we loop through the variable "**sample_names**" we created earlier:

```
for (i in sample_names) {
  print(i)
}
```

```
> for (i in sample_names) {
+   print(i)
+ }
[1] "Rep1_ICBdT"
[1] "Rep3_ICBdT"
[1] "Rep5_ICBdT"
[1] "Rep1_ICB"
[1] "Rep3_ICB"
[1] "Rep5_ICB"
>
```


The for loop

Now, let's break down the complicated for loop we use to create Seurat objects.

```
for (sample in sample_names) {
  cat("\nWorking on sample: ", sample, "\n")
  path <- paste("data/", sample, "-sample_filtered_feature_bc_matrix.h5", sep="")
  cat("Input file path: ", path, "\n")
  data <- Read10X_h5(path)
  seurat_obj <- CreateSeuratObject(counts = data, project = sample, min.cells = 10,
                                   min.features = 100)
  sample.data[[sample]] <- seurat_obj
}
```


The `cat()` function

The `**cat()**` function in R is used to **concatenate and print objects** (especially strings) to the console, without quotes and without line breaks by default.

Basic usage:

```
cat("Hello", "world!")
```

```
> cat("Hello", "world!")
Hello world!
>
```

The **default separator** is a space. We can use a different separator, such as **dash**.

```
> cat("Hello", "world!", sep = "-")
Hello-world!
>
```


The for loop

So, if we run the for loop with only the **first line of code**:

```
for (sample in sample_names) {
  cat("\nWorking on sample: ", sample, "\n")
}
```


The for loop

```
> for (sample in sample_names) {
+   cat("\nWorking on sample: ", sample, "\n")
+ }

Working on sample:  Rep1_ICBdT

Working on sample:  Rep3_ICBdT

Working on sample:  Rep5_ICBdT

Working on sample:  Rep1_ICB

Working on sample:  Rep3_ICB

Working on sample:  Rep5_ICB
>
```


The for loop

`\n` is a **new line** character. Things printed behind a `\n` will be returned to a new line.

Actually, we don't need the first "\n" here in our for loop. We can delete the extra empty line so it looks cleaner.

```
for (sample in sample_names) {
  cat("Working on sample:", sample, "\n")
}
```


The for loop

```
> for (sample in sample_names) {
+   cat("Working on sample:", sample, "\n")
+ }
Working on sample: Rep1_ICBdT
Working on sample: Rep3_ICBdT
Working on sample: Rep5_ICBdT
Working on sample: Rep1_ICB
Working on sample: Rep3_ICB
Working on sample: Rep5_ICB
```


The for loop

Now, let's try run two lines of code in our for loop.

```
for (sample in sample_names) {
  cat("Working on sample:", sample, "\n")
  path <- paste("data/", sample, "-sample_filtered_feature_bc_matrix.h5", sep="")
}
```

The `paste()` function.

The paste() function in R is used to concatenate strings or other objects into a single character string. It is super helpful for **building labels, messages, filenames**, etc.

The for loop

```
> for (sample in sample_names) {
+   cat("Working on sample:", sample, "\n")
+   path <- paste("data/", sample, "-sample_filtered_feature_bc_matrix.h5", sep="")
+ }
Working on sample: Rep1_ICBdT
Working on sample: Rep3_ICBdT
Working on sample: Rep5_ICBdT
Working on sample: Rep1_ICB
Working on sample: Rep3_ICB
Working on sample: Rep5_ICB
>
```

If we see this, there is no difference on the screen than last time. That is because the loop only run the code but we didn't ask it to print anything out to the screen.

It still create a "path" in every loop, we can visualise it by asking the loop to print out "path".

The for loop

```r
for (sample in sample_names) {
  cat("Working on sample:", sample, "\n")
  path <- paste("data/", sample, "-sample_filtered_feature_bc_matrix.h5", sep="")
  print(path)
}
```

```
Working on sample: Rep1_ICBdT
[1] "data/Rep1_ICBdT-sample_filtered_feature_bc_matrix.h5"
Working on sample: Rep3_ICBdT
[1] "data/Rep3_ICBdT-sample_filtered_feature_bc_matrix.h5"
Working on sample: Rep5_ICBdT
[1] "data/Rep5_ICBdT-sample_filtered_feature_bc_matrix.h5"
Working on sample: Rep1_ICB
[1] "data/Rep1_ICB-sample_filtered_feature_bc_matrix.h5"
Working on sample: Rep3_ICB
[1] "data/Rep3_ICB-sample_filtered_feature_bc_matrix.h5"
Working on sample: Rep5_ICB
[1] "data/Rep5_ICB-sample_filtered_feature_bc_matrix.h5"
> 
```

We can see that in every loop, it does create a path to store the file path and name.

The for loop

Similarly, if we use the **cat()** function to print texts to console.

```
for (sample in sample_names) {
  cat("Working on sample:", sample, "\n")
  path <- paste("data/", sample, "-sample_filtered_feature_bc_matrix.h5", sep="")
  cat("Input file path:", path, "\n")
}
```

```
Working on sample: Rep1_ICBdT
Input file path: data/Rep1_ICBdT-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep3_ICBdT
Input file path: data/Rep3_ICBdT-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep5_ICBdT
Input file path: data/Rep5_ICBdT-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep1_ICB
Input file path: data/Rep1_ICB-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep3_ICB
Input file path: data/Rep3_ICB-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep5_ICB
Input file path: data/Rep5_ICB-sample_filtered_feature_bc_matrix.h5
>
```


The for loop

Similarly, if we add another line of code.

To read in count matrix from 10X CellRanger hdf5 file.

```
for (sample in sample_names) {
  cat("Working on sample:", sample, "\n")
  path <- paste("data/", sample, "-sample_filtered_feature_bc_matrix.h5", sep="")
  cat("Input file path:", path, "\n")
  data <- Read10X_h5(path)
}
```

In every loop, it reads in a hdf5 file and stored it in the "**data**" variable.

Since in every loop, the variable names are the same. The variables will be **overwritten** in the next loop.

The for loop

```
Working on sample: Rep1_ICBdT
Input file path: data/Rep1_ICBdT-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep3_ICBdT
Input file path: data/Rep3_ICBdT-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep5_ICBdT
Input file path: data/Rep5_ICBdT-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep1_ICB
Input file path: data/Rep1_ICB-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep3_ICB
Input file path: data/Rep3_ICB-sample_filtered_feature_bc_matrix.h5
Working on sample: Rep5_ICB
Input file path: data/Rep5_ICB-sample_filtered_feature_bc_matrix.h5
> 
```

We get the same result as the last time.

But it takes time to print the next file name, that is because in between, it's running the `**Read10X_h5()**` function.

The for loop

```
for (sample in sample_names) {
  cat("Working on sample:", sample, "\n")
  path <- paste("data/", sample, "-sample_filtered_feature_bc_matrix.h5", sep="")
  cat("Input file path:", path, "\n")
  data <- Read10X_h5(path)
  seurat_obj <- CreateSeuratObject(counts = data, project = sample, min.cells = 10,
                                   min.features = 100)
  sample.data[[sample]] <- seurat_obj
}
```

Then, for the last two lines.

The line for creating Seurat object has the same idea.

The last line is to save each Seurat object to a list, with the name to be the sample name.

Thank you

Contact us

Jiajia Li
Research School of Biology

RN Robertson Building, 46 Sullivan's Creek Rd
The Australian National University
Canberra ACT 2600

E jiajia.li1@anu.edu.au

Australian National University

TEQSA PROVIDER ID: PRV12002 (AUSTRALIAN UNIVERSITY)
CRICOS PROVIDER CODE: 00120C