Table of Contents

R for Bioinformatics

by Sean Davis[1]

Replace with an introduction of your book.

[1]. seandavi@gmail.com ↩

Overview of R

What is R?

- A software package
- A programming language
- A toolkit for developing statistical and analytical tools
- An extensive library of statistical and mathematical software and algorithms
- A scripting language
- Many other things to other people

Why would I use R?

- R is cross-platform and runs on Windows, Mac, and Linux (as well as more obscure systems).
- R provides a vast number of useful statistical tools, many of which have been painstakingly tested.
- R produces publication-quality graphics in a variety of formats.
- R plays well with FORTRAN, C, C++, Java and many other languages.
- R scales, making it useful for small and large projects. It is NOT Excel.
- R eschews the GUI.

I can develop code for analysis on my Mac laptop. I can then install the *same* code on our 20k core cluster and run it in parallel on 100 samples, monitor the process, and then update a database with R when complete.

Why should I not use R?

- R cannot do everything.
- R is not always the "best"" tool for the job.

- R will *not* hold your hand.
- The R documentation can be opaque (but it does exist).
- R can drive you crazy (on a good day) or age you prematurely (on a bad one).
- Finding the right package to do the job you want to do can be challenging; worse, some contributed packages are unreliable.
- R eschews the GUI.

R License and the Open Source Ideal

- R is free!
- R is distributed under a GNU license.
    - You may download the source code.
    - You may modify the source code to your heart's content.
    - You may distribute the modified source code and even charge money for it, but you must distribute the modified source code under the original GNU license.

This license means that R will always be available, will always be open source, and can grow organically without constraint.

Getting Started with R

Installation

Data Types

Introduction to R Vectors

Learning objectives

- Understand that there are many ways to create vectors
- Understand how vectors can be subset
- Understand that vectors can be used as indexes
- Understand that vector operations in R are usually the fastest way to perform computation

Skills

- Creating vectors
- Using vector operations
- Subsetting and indexing vectors

Exercises

Constructing vectors

A vector is a sequence of data elements of the same basic type. Members in a vector are officially called components. Nevertheless, we will just call them members in this site.

Here is a vector containing three numeric values 2, 3 and 5.

```
c(2, 3, 5)
```

```
## [1] 2 3 5
```

And here is a vector of logical values.

```
c(TRUE, FALSE, TRUE, FALSE, FALSE)
```

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

A vector can contain character strings.

```
c("aa", "bb", "cc", "dd", "ee")
```

```
## [1] "aa" "bb" "cc" "dd" "ee"
```

Incidentally, the number of members in a vector is given by the length function.

```
length(c("aa", "bb", "cc", "dd", "ee"))
```

```
## [1] 5
```

Vectors can be combined via the function c. For examples, the following two vectors n and s are combined into a new vector containing elements from both vectors.

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc", "dd", "ee")
c(n, s)
```

```
## [1] "2"  "3"  "5"  "aa" "bb" "cc" "dd" "ee"
```

In the code snippet above, notice how the numeric values are being coerced into character strings when the two vectors are combined. This is necessary so as to maintain the same primitive data type for members in the same vector. *Remember that*

vectors can contain only one data type.

Vector indexing

We retrieve values in a vector by declaring an index inside a single square bracket "[]" operator.

For example, the following shows how to retrieve a vector member. Since the vector index is 1-based, we use the index position 3 for retrieving the third member.

```
s = c("aa", "bb", "cc", "dd", "ee")
s[3]
```

```
## [1] "cc"
```

Unlike other programming languages, the square bracket operator returns more than just individual members. In fact, the result of the square bracket operator is another vector, and s[3] is a vector slice containing a single member "cc".

If the index is negative, it would strip the member whose position has the same absolute value as the negative index. For example, the following creates a vector slice with the third member removed.

```
s[-3]
```

```
## [1] "aa" "bb" "dd" "ee"
```

If an index is out-of-range, a missing value will be reported via the symbol NA.

```
s[10]
```

```
## [1] NA
```

Numeric index vectors

A new vector can be sliced from a given vector with a numeric index vector, which consists of member positions of the original vector to be retrieved.

Here it shows how to retrieve a vector slice containing the second and third members of a given vector s.

```
s = c("aa", "bb", "cc", "dd", "ee")
s[c(2, 3)]
```

```
## [1] "bb" "cc"
```

The index vector allows duplicate values. Hence the following retrieves a member twice in one operation.

```
s[c(2, 3, 3)]
```

```
## [1] "bb" "cc" "cc"
```

The index vector can even be out-of-order. Here is a vector slice with the order of first and second members reversed.

```
s[c(2, 1, 3)]
```

```
## [1] "bb" "aa" "cc"
```

To produce a vector slice between two indexes, we can use the colon operator ":". This can be convenient for situations involving large vectors.

```
s[2:4]
```

```
## [1] "bb" "cc" "dd"
```

More information for the colon operator is available in the R documentation.

```
help(":")
```

Logical index vectors

A new vector can be sliced from a given vector with a logical index vector, which has the same length as the original vector. Its members are TRUE if the corresponding members in the original vector are to be included in the slice, and FALSE if otherwise.

For example, consider the following vector s of length 5.

```
s = c("aa", "bb", "cc", "dd", "ee")
```

To retrieve the the second and fourth members of s, we define a logical vector L of the same length, and have its second and fourth members set as TRUE.

```
L = c(FALSE, TRUE, FALSE, TRUE, FALSE)
s[L]
```

```
## [1] "bb" "dd"
```

The code can be abbreviated into a single line.

```
s[c(FALSE, TRUE, FALSE, TRUE, FALSE)]
```

```
## [1] "bb" "dd"
```

Named vector members

We can assign names to vector members.

For example, the following variable v is a character string vector with two members.

```
v = c("Mary", "Sue")
v
```

```
## [1] "Mary" "Sue"
```

We now name the first member as First, and the second as Last.

```
names(v) = c("First", "Last")
v
```

```
##  First   Last
## "Mary"  "Sue"
```

Then we can retrieve the first member by its name.

```
v["First"]
```

```
##  First
## "Mary"
```

Furthermore, we can reverse the order with a character string index vector.

```
v[c("Last", "First")]
```

```
##   Last  First
##  "Sue" "Mary"
```

Glossary

GNU

gnu definition

1. Overview of R

software

stuff to run a computer.

1. Overview of R