R Basics: Vectors and packages

Presenter: Steve Baskauf
steve.baskauf@vanderbilt.edu

DiSC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

CodeGraf landing page

- vanderbi.lt/codegraf

R objects

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

Common types of data

- **character**, e.g. "Fred" or "!@#ts23" (in quotes)
- **numeric**, e.g. 15 or 6.02 (no quotes)
- **logical**, TRUE or FALSE (all caps, no quotes)

Object name recommendations

- An **object** in R is a specialized data structure.
- We can use the term **variable** to refer to named objects
- R doesn't know what a name "means". A meaningful name helps human readers of the code.
- Be descriptive (what the object is or does)
- **snake_case** (underscores) is commonly used:
  - `ordinary_relational_processes`
- camelCase is sometimes used:
  - `bookList`, `alphabetizeParticipants`

Assigning a value to an object

- You can **assign** a value to an object using **<-** (similar to a left arrow)
- Examples:

  `name <- "Steve"` (creating a character object)

  `my_number <- 6.02` (creating a numeric object)
- Using the equals sign (=) is allowed, but not recommended.
- alt-minus is an RStudio shortcut to generate **<-**

Displaying the value of an object

- There is a "print" command in R, but it is not commonly used unless writing to a file.
- Entering the name of an object (or expression) in the console evaluates and displays its value

Function review

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

Using a function

- We don't have to know anything about the code that makes a function work. We just need to know:
  - What the function does
  - What arguments to put into it
  - What the function will output
- Examples:

`sqrt(2)` (evaluate and display)

`x <- sqrt(3)` (evaluate and assign to an object)

Sources of functions

- Functions can be:
  - **Built-in** to R and always available (examples in next section)
  - Imported by **loading a package** (more on this later in this lesson)
  - **Defined** as part of the code in the script (not covered in this module)

Loading an R script

DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

Ways to load an R script from GitHub

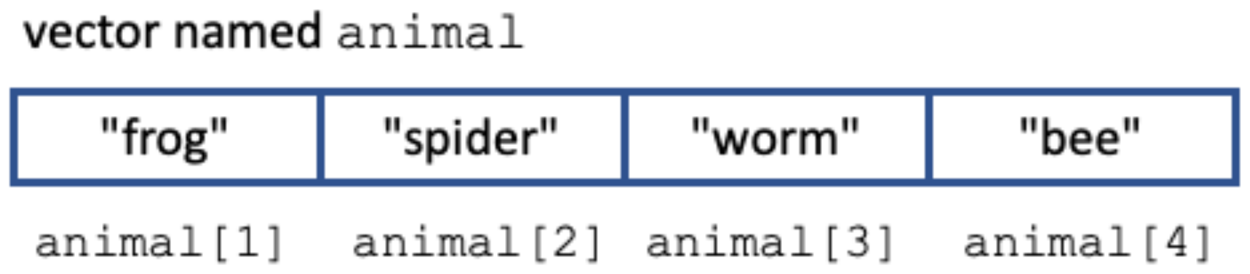
- Easy:
    1. Go to the script web page at GitHub.
    2. Left click on the Raw button.
    3. Copy all the text.
    4. Paste into a new RStudio editor window
    5. Save if desired.
- Harder, but generic:
    1. Go to the script web page at GitHub.
    2. Right click on the Raw button and select "Save Link As…"
    3. Save the file somewhere you can find it.
    4. In RStudio, select "Open File…" from the File menu and navigate to the file you saved, or click on the open file icon.
    5. Select the file and click Open.

Vectors

DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

Vectors are king in R



- A **vector** is the most common kind of data structure in R.
- Vectors contain a sequence of the **same type** of data.

Creating vectors

- We commonly use the **combine** function to make vectors:

```
number_vector <- c(1, 3, 6, 10, 15)
animal <- c("frog", "spider", "worm", "bee")
```

- We can also generate a **sequence** of numbers:

```
number_range <- 3:9
count_down <- 10:0
go_negative <- 5:-3
```

- The generated sequence is just another vector!
- (Python users: note the range includes the final value)

Knowing what's going on with a vector

- display it in console
- examine its value in the environment data pane
- examine its properties:

  `length(animal)` (how many items)

  `mode(animal)` (type of data in vector)

Referring to parts of vectors

- Referencing a **single item**:

  `animal[3]`   (displays the third item)

  `animal[2] <- "arachnid"` (assigns "arachnid" to the 2nd item)

- Referencing a **range of items** (subvector):

  `animal[2:4]`   (the range 2:4 is actually a vector itself)

- (Python users: R vectors are "1 based"; the first item is numbered 1, not 0. Also, the range includes the final value.)

Single item objects are vectors, too.

- Surprisingly, a single data item assigned to an object is also a vector. We can see this if we ask its length as if it were a vector:

```
an_item <- "some character string"
length(an_item)
```

- We can reference the single item using vector notation:

```
an_item[1]
```

Vectorized computing

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

Vectorized computing

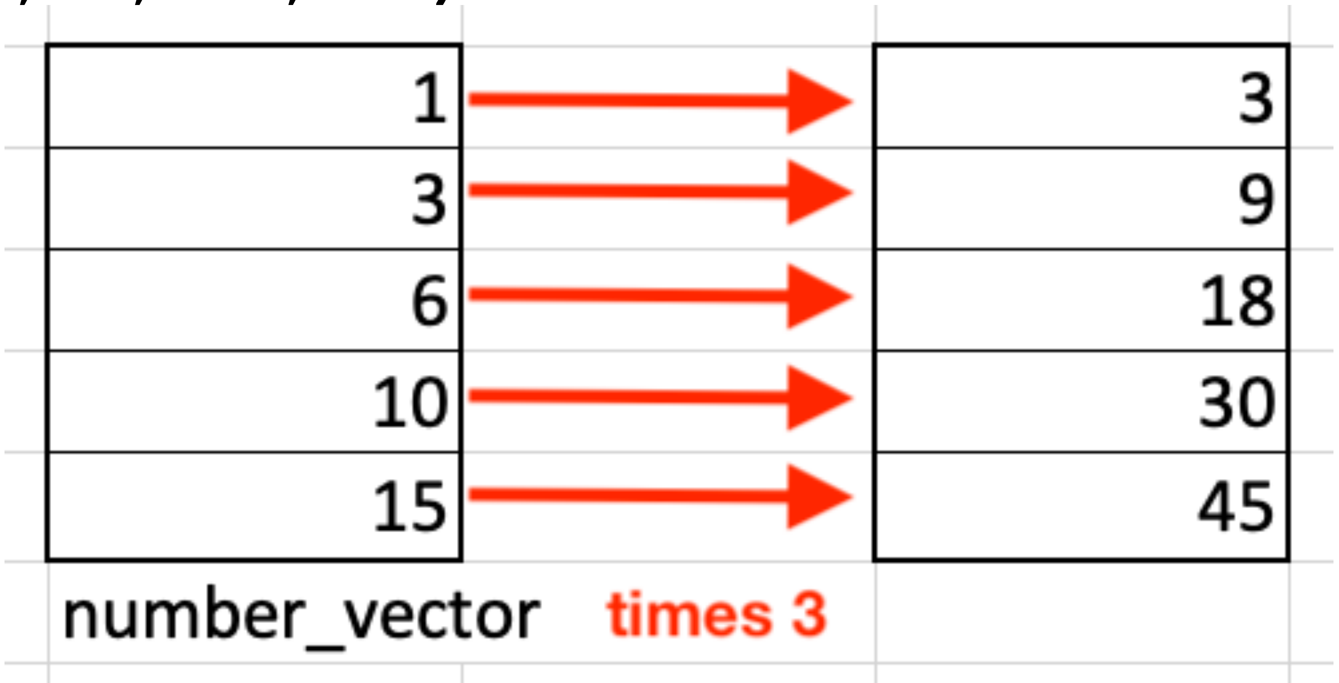
- **Vectorized computing** is a programming paradigm used by R and Python Pandas.
- When operations (math or function) are performed on vectors, they generally are performed on **all items** in the vectors without having to iterate through each item in the vector.
- When an operation is done involving two vectors, the operation is carried out sequentially on pairwise items in the two vectors.
- The **result** is generally a **vector** with the same length.

Operations on vectors

- Many functions work equally well for a single item or a multi-item vector (since they are both vectors):

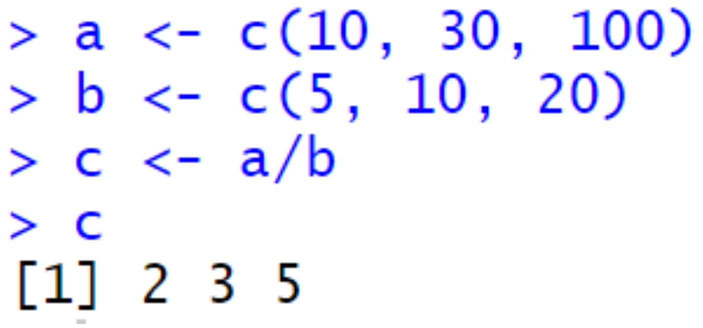
```
number <- 2
sqrt(number)

number_vector <- c(1, 3, 6, 10, 15)
sqrt(number_vector)
number_vector * 3
```

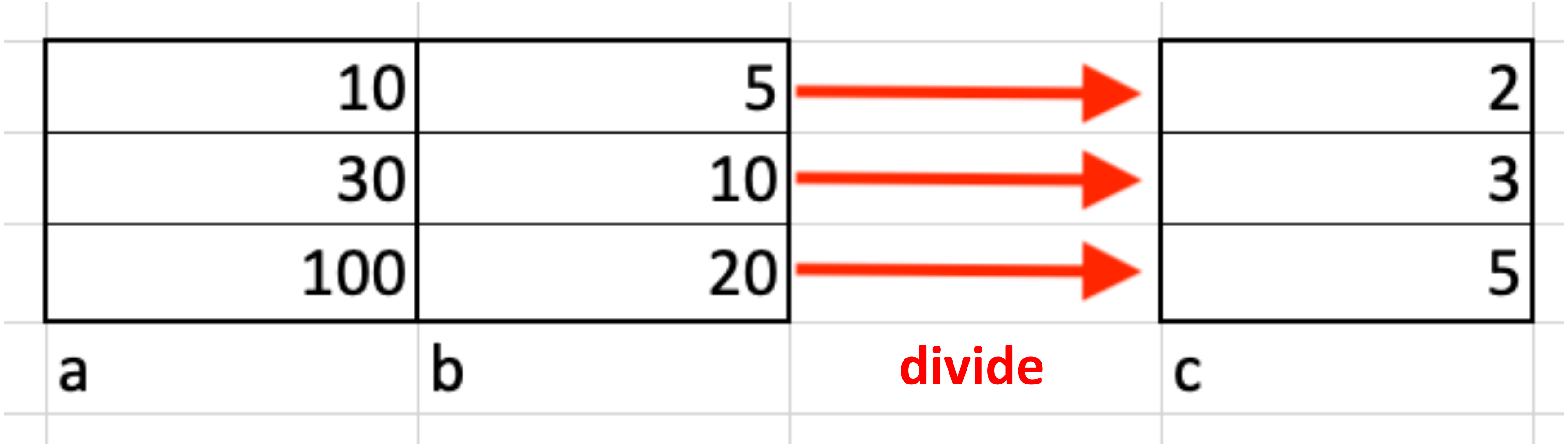


Operations on vectors

- Example of **two-vector** operation:



```
> a <- c(10, 30, 100)
> b <- c(5, 10, 20)
> c <- a/b
> c
[1] 2 3 5
```



Using packages

DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

Loading packages

- Some functions that are not built-in are part of packages that must be loaded.
- Example: loading by command
- Example: loading by GUI
- Example: including **`library()`** in a script

Installing packages

- A package may need to be installed the first time you use it.
- Installing causes download from CRAN
- If using Anaconda, installing often not necessary

- Example: command line
- Example: GUI

Dependencies

- Some packages need code from other packages (dependencies) to operate
- Installing a package with dependencies may also result in the dependences being installed as well.
- Some large umbrella packages (like **`tidyverse`**) may take a long time to install