

Overview of R and Python

Alex Emmons, PhD (BTEP)

2025-08-21

**Bioinformatics Training
and Education Program**

Learning Objectives

1. Learn about popular programming languages in bioinformatics
2. Compare advantages and disadvantages of Python and R
3. Discuss what you will need to learn to use these languages
4. Discuss learning resources

Choosing a programming language

**Bioinformatics Training
and Education Program**

What is a programming language?

A programming language is a formal language that specifies a set of instructions for a computer to perform specific tasks. It is used to write software programs and applications, and to control and manipulate computer systems.—[GeeksforGeeks](#)

**Bioinformatics Training
and Education Program**

What is a programming language?

Key features of programming languages include:

Syntax

data types

Variables

Operators

Control Structures

Libraries

Paradigms (programming styles / philosophies) — [GeeksforGeeks](#)

Examples include C++, C#, Perl, Java, Ruby, Python, Julia, and R.

More on paradigms, [here](#).

**Bioinformatics Training
and Education Program**

Why learn programming?

Do all molecular scientists need to learn a programming language?

- Absolutely not.

BUT

- We are in a big data era, and learning to code can be extremely beneficial, especially if you do not have access to bioinformatics analysts to analyze the data for you or expensive **licensed software**.

**Bioinformatics Training
and Education Program**

Which programming language should I learn?

1. Bash

- Most of bioinformatics can be done by understanding specific software applications and running those applications in a pipeline, usually using some form of bash scripting. Bash as a scripting language is fairly important for processing biological data, **though arguably, not a formal programming language.**

2. Python or R

- Depending on your goals, you may lean toward one programming language over another. For example:
 - Interested in statistics and data visualization? R may be for you.
 - Interested in software development and machine learning? A more general language like Python may be a better fit.

**Bioinformatics Training
and Education Program**

Check out **this video!**

What is R?

- released in 1993
- a computational language and environment for statistical computing and graphics.
 - complex statistical functions easily accessible
 - easy to get started, but more difficult to learn
- Key features:
 - open-source
 - extensible (Packages on CRAN (> 22,000 packages), Github, Bioconductor)
 - Tidyverse for data wrangling and visualization
 - wide community
 - Maintained by a network of collaborators – The R Core Team

Check out more on [The R Project for Statistical Computing website](#).

**Bioinformatics Training
and Education Program**

What is Python?

- developed as early as 1991
- high-level, popular, general-purpose programming language that has a readable and easy to learn syntax
- Key features:
 - easy to read
 - easy to learn
 - interpreted
 - multi-platform
 - wide community (**highly popular**)
 - open source libraries (> 500,000)
- Two major versions (**python2** and **python3**)
- Not as easy to just start analyzing data

What is Python?

Check out more at <https://www.python.org/>.
Also, check out [this introduction for biologists](#).

**Bioinformatics Training
and Education Program**

Advantages of R and Python

R Programming

- Tidyverse
- Data Visualization (Base R and ggplot2)
 - additional packages that enhance these, especially for -omics data
- More packages for data science / bioinformatics
 - Bioconductor
- Report generation
 - R Markdown
 - Quarto
- more popular among scientists and academics (i.e., non-programmers)

Python

- More consistent syntax (generally a right way to do something)
- Large data manipulation (generally more efficient)
 - shines in machine learning (**scikit-learn**)
- Report Generation
 - Jupyter Notebook
- More popular among software developers and across multiple domains

**Bioinformatics Training
and Education Program**

	Python	R
General	Python is a general-purpose programming language for data analysis and scientific computing.	R is a functional programming environment and language for statistical computing and graphics.
Objective	Data Science, Web Development, Embedded Systems	Data Science & Statistical Modeling
IDE	iPython, Pycharm, Jupyter Notebook, Spyder	Rstudio, R GUI, R KWARD
Data Collection	Supports CSV files, SQL, JSON , and webscraping with BeautifulSoup .	Can also import csv files with built-in readr library. R's library RCurl provides a simple way to make API requests, similar to Python's requests package.
Data Analysis	Organize dataframes with Pandas filtering, sorting. Python takes a more streamlined approach for data science projects.	Complex data visualization tools make the exploratory data analysis (EDA) process much more complex than Python.
Essential Packages & Libraries	Numpy, Pandas, matplotlib, scipy, scikit-learn, TensorFlow	caret, stringr, ggplot2, knitr, tidyverse, markdown, shiny, forcats, haven
Database Handling Capacity	Can easily handle large data because there are less constraints for memory usage	R computes everything in memory, so its capabilities are limited by RAM size. A major downfall of R is the inability to handle massive amounts of data
Data Visualization	Despite the capabilities of data visualization tools like Matplotlib and Seaborn , Python fails to measure up to data visualization features of R.	Developed by and for statisticians, R has complex data visualization features.
Syntax	The 'zen of python' is that there's a proper way to write code.	R doesn't have this set of rules. Also indexing starts at 1, which can be considered unconventional for general programmers.
Learning Curve	Simple and readable code structure makes it easier for beginners to learn. It also allows for object-oriented programming. It also offers a wide range of data structures that you wouldn't expect from a general-purpose language.	R's functional syntax isn't easy for beginners, but not too challenging for those well versed in programming. It also offers a few data structures, but fails to handle large amounts of data.

Bioinformatics Training and Education Program

Image from Toward Data Science, *Python vs R: The Basics*, author Sidney Kung

What do you need to know to
learn R or Python?

**Bioinformatics Training
and Education Program**

Installation

If you intend to use through Biowulf, no installation necessary.

R:

- Use these [instructions](#) from Posit.

Python:

- You can download directly from <https://www.python.org/downloads/>.

**Bioinformatics Training
and Education Program**

How do we execute our code?

With both R and Python, code is executed

- interactively line by line from the command line
- interactively in an IDE (Recommended)
- as a script submitted from the command line or in an IDE

For python, to get started from the command line:

```
1 python
2 quit()
```

For R, to get started from the command line:

```
1 R
2 q()
```

**Bioinformatics Training
and Education Program**

What is an IDE?

An IDE is an integrated development environment. IDEs generally include features such as:

- Console
- File access
- Environment / variable view
- Data view
- Plotting window
- History
- Autocomplete
- Debugging
- Markdown

IDEs make coding easier. They increase productivity and facilitate project management.

**Bioinformatics Training
and Education Program**

IDEs for R and Python

R

- RStudio
- VS Code*
 - R
 - Python
- Positron*

Python

- JupyterLab / Jupyter Notebook*
 - Can be used with C++, Julia, GNU octave, R, Ruby, and Scheme
- Spyder
- PyCharm
- iPython

**Bioinformatics Training
and Education Program**

Elements of programming with python or R

- libraries / modules
- syntax
- variables
- functions
- data types (dictionaries and tuples in python)
- loops and conditionals

Libraries

R Packages can be found at:

- CRAN
 - METACRAN– to search for packages
- Bioconductor
- Github

Python

- Python Package Index (PyPI)

**Bioinformatics Training
and Education Program**

Bioconductor

- A repository for R packages related to biological data analysis, primarily bioinformatics and computational biology.
- a great place to search for -omics packages and pipelines.
- Released every 6 months and work with a specific version of R.
 - included packages are “mutually compatible, traceable, and guaranteed to function for the associated version of R”
 - Package types: Software, annotation, experimental data, workflows

Bioinformatics related python packages

- Biopython
- Bioconda
 - Conda, as a package management and environment management system was created for python but now can be used for any language.
- scverse

R Syntax

- more functional
 - built around functions (`function()`)
- Case sensitive
- white space insensitive (rules for line continuation)
- `<=` or `=` assignment operators
- `#` used for comments
- keywords or words with special meaning (`?reserved`)
 - for example, `if`, `else`, `repeat`, `while`, `function`, `for`, `in`, `next`, and `break` are used for control-flow statements and declaring user-defined functions.
- statement grouping with `{}`
- indexing starts with 1; `-` removes values
- Getting help with `help()` or `?` (e.g., `?print`)
- Paths use `/`; `\` is an escape

Python Syntax

- more object oriented (`.` is an operator and should not be used to name variables, `object.method()`)
- `=` assignment operator
- `#` used for comments
- 33 reserved words `help("keywords")`
- lists use brackets `[]`, dictionaries use `{}`
- indentation is important (4 spaces) – defines blocks of code
- indexing starts with 0; `-` for negative indexing
- Getting help with `help()` (e.g., `help(print)`)
- Paths use `/`; `\` is an escape

Compare the code

A syntax comparison from Dataquest:

<https://www.dataquest.io/blog/python-vs-r/>.

 R code can be run using python with the `rpy2` library. Python code can be executed through R using the `reticulate` package.

**Bioinformatics Training
and Education Program**

Variables

Essentially named storage that can be manipulated.

Rules for R variables:

1. Avoid spaces or special characters EXCEPT '_' and '.'
2. No numbers or underscores at the beginning of an object name.
3. Avoid common names with special meanings (See ?Reserved) or assigned to existing functions (These will auto complete).
4. Case sensitive

Rules for Python variables:

1. Contains alpha-numeric characters and underscores
2. Must start with a letter or the underscore character
3. cannot start with a number
4. Case sensitive

**Bioinformatics Training
and Education Program**

Functions

Used to perform specific tasks.

R:

```
1 product <- function(a,b){  
2   c<- a*b  
3   c  
4 }  
5 product(5,7)
```

```
[1] 35
```

Python:

```
1 def product(a,b):  
2   c = a*b  
3   return c  
4  
5 print(product(5,7))
```

```
35
```

Code example from <https://www.r-bloggers.com/2017/05/r-vs-python-different-similarities-and-similar-differences/>

**Bioinformatics Training
and Education Program**

Data Types

R:

Data types: integer, double (numeric), character, and logical. (Also, complex and raw)

Data structures: vectors, lists, data frames, matrices, factors.

```
1 x <- c(1,2,3)
2 typeof(x)
3 ## [1] "double"
4 class(x)
5 ## [1] "numeric"
6 is.vector(x)
7 ## [1] TRUE
```

Python:

Data types: Integers, Floats, Complex, Strings, booleans (TRUE, FALSE)

Data structures: arrays, **tuples**, **lists**, **dictionaries**, Pandas data frames.

```
1 import numpy as np
2 x = [1,2,3]
3 x = np.array(x)
4 print(type(x))
```

```
<class 'numpy.ndarray'>
```

**Bioinformatics Training
and Education Program**

Mutable / Immutable Objects

```
1 # Initial assignment
2 original_list = [1, 2, 3]
3 # Assigning a new variable to the same list
4 new_list = original_list
5
6 # Modifying the new list
7 new_list.append(4)
8
9 # Both lists show the modification
10 print("Original list:", original_list) # Output: [1, 2, 3, 4]
```

Original list: [1, 2, 3, 4]

```
1 print("New list:", new_list) # Output: [1, 2, 3, 4]
```

New list: [1, 2, 3, 4]

**Bioinformatics Training
and Education Program**

Loops and conditionals

Loops – used to iterate over a sequence

R:

```
1 fruit <- c('apples','bananas','cantaloupe')
2
3 for(i in fruit) {
4   print(i)
5 }
```

```
[1] "apples"
[1] "bananas"
[1] "cantaloupe"
```

Python:

```
1 fruit=['apples', 'bananas', 'cantaloupe'] #
2
3 for i in fruit:
4   print(i)
```

```
apples
bananas
cantaloupe
```

Conditionals – code is executed based on conditions

R:

```
1 x<-3
2 y<-5
3
4 if(x<y){
5   print(paste(x, 'is less than', y))
6 } else{
7   print(paste(x, 'is not less than', y))
8 }
```

```
[1] "3 is less than 5"
```

Python:

```
1 x=3
2 y=5
3
4 if x<y:
5   print(x, 'is less than', y)
6 else:
7   print(x, 'is not less than', y)
```

```
3 is less than 5
```

**Bioinformatics Training
and Education Program**

Resources to learn

**Bioinformatics Training
and Education Program**

BTEP and Others

- Check the [NIH Bioinformatics Calendar](#) for upcoming events including courses or lessons on python and R.
- Past BTEP courses
 - [Class documentation](#)
 - [Video Archive](#)
- [NIH library](#)
- [NIAID Bioinformatics Resources](#)

**Bioinformatics Training
and Education Program**

Coursera

- Coursera – programming, AI, genomics, bioinformatics, and data science
Click [here](#) for license information.

Books and other resources:

- See [this list](#) for introductory R material.
- [A Primer for Computational Biology](#), Shawn T. O’Neil
- [An Introduction to R and Python for Data Analysis : A Side-By-Side Approach](#) – requires VPN

**Bioinformatics Training
and Education Program**

Other Interesting Resources

- Unix – <https://explainshell.com/>
- R statistics – <https://swirlstats.com/>
- R data science – <https://education.rstudio.com/learn/beginner/>
- Python – <https://www.python.org/about/gettingstarted/>
- Python – <https://www.learnpython.org/>
- Python – <https://realpython.com/>
- Glittr – <https://glittr.org>
- CHIRP – NIH Community Chatbot– <https://chirp.od.nih.gov>

Sources

1. https://www.datacamp.com/blog/python-vs-r-for-data-science-whats-the-difference#gs.JrY_3bk
2. https://shiring.github.io/r_vs_python/2017/01/22/R_vs_Py_post
3. <https://realpython.com/python-ides-code-editors-guide/>
4. https://medium.com/@hamza_33678/programming-for-bioinformatics-r-vs-python-52969a1f7a49#:~:text=While%20both%20R%20and%20Python,in%20keeping%20RAM%20consumption%20lo
5. <https://towardsdatascience.com/python-vs-r-the-basics-d754c45c1596>
6. <https://www.dataquest.io/blog/python-vs-r/>
7. Learning Python for Data Science: What to Learn and Why, Cindy Sheffield, NIH Library

**Bioinformatics Training
and Education Program**

Bioinformatics Training and Education Program