WHEN SHOULD I USE PYTHON OR R?

Python and R are commonly used, versatile programming languages for data science and analytics. Unlike commercial tools such as SAS and SPSS, both languages are open-source, free for anyone to download. However, both have different strengths and weaknesses meaning that the language you use will depend on your specific use case. Ultimately, both languages can accomplish nearly any data science task to various extents, from data manipulation and automation to ad-hoc analysis and exploring datasets. Users may leverage both languages for different purposes, e.g., conducting early stage data analysis and exploration in R, then switching to Python when it's time to build a production API. If you are new to learning programming languages, it is best to stay focused on learning how to break down and think through a problem, rather than trying to become a R or Python expert right out of the gate.

R is an open source programming language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering) and graphical techniques, and is highly extensible.

Python is an open source programming language often used for web development, software development, mathematics, and system scripting.

Remember, the choice is not "R vs Python"

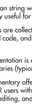
There will be times where either Python or R will serve the needs of your project. There will be times where your company may lean more heavily on one language versus the other. At the end of the day, the true competitive skill is knowing how to break down and solve the given business problem, regardless of the language. From there, you'll choose the programming language that makes sense for what you're trying to accomplish and/or satisfies what your company requires you to use. Knowing what problems you want to solve and what tasks you need to accomplish will set you up with a mindset to succeed as a Data Analyst.

In many ways, this is like knowing how to drive a car. While some vehicles will have different features and can accommodate different needs (e.g., more passengers, Bluetooth, heated seats, tow package), your ultimate goal in learning how to drive is not to learn the specifics of any one car – it's to know how to use any available car to get from point A to point B.

Discover more about the:

Usability | Ease of Learning | Ecosystem
Advantages | Disadvantages | Examples
Use Cases | Popular Libraries | Support

Usability



- People with no software engineering experience often find R easier to learn than Python.
- Statistical models can be written with only a few lines of code - very efficient!
- Functionalities can be written in several different ways - R is not known for consistent syntax.
- Users can easily process complex functions in R. All types of statistical tests and models are readily available and easily executed through a variety of libraries.

- People with software engineering experience often find Python easier to learn than R.
- Simple syntax makes coding and debugging easy.
- Indentation of code affects meaning.
- Any piece of functionality is always written the same way - consistent syntax.
- Python is flexible when you need to create something brand new. Developers can also use it for scripting websites or other applications.

Ease of Learning

- It is easier to develop baseline competence in R, but the intricacies of advanced functionalities make it more difficult to develop expertise broadly across R's libraries.
- Because syntax and functionality will vary from library-to-library, users need to be well versed in researching R skills on-line (e.g., Google, Stack Overflow) to learn.

- Python's focus on readability and simplicity makes its learning curve relatively linear and smooth.
- While the ability to research various functionalities online is important, the consistent syntax of Python makes learning new packages and modules a straightforward task.

Ecosystem

- R has a rich ecosystem of cutting-edge interface packages available to communicate between open-source languages.
- R users can string workflows together, which is especially useful for data analysis.
- Packages are collections of R functions, data, and compiled code, and can be installed in R with one line.
- R documentation is often readily available for most libraries (typically includes example code).
- Complementary offerings, such as "RStudio", provide R users with an easy-to-use interface for writing, editing, and deploying code.

- Python has a robust ecosystem, the programming syntax is simple, and commands mimic the English language.
- Python code is syntactically clear, elegant, easy to type, and easy to interpret.
- Python is great for building data science pipelines and machine learning products integrated with web frameworks at scale. However, be mindful of dependencies when installing Python libraries.
- The Python Package Index (PyPI) and Anaconda are repositories of Python software with all libraries. Users can contribute to these repositories, but it's a bit complicated in practice to do so.

Advantages

- R has many functionalities for data analysis.
- R is great for statistical analysis, which are often cited in academic journals.
- While built around a command line, many R users work inside of RStudio. This environment includes a data editor, debugging support, and a window to hold graphics.

- As a general-purpose programming language, Python is useful beyond just data analysis.
- Python is popular for its code readability, speed, and diverse functionalities.
- Python is great for mathematical computation and learning how algorithms work.
- Python has a high ease of deployment and reproducibility.

Disadvantages

- R was developed by statisticians, not coders, meaning that it was not designed to make coding easier.
- R must be learned library-by-library – it has no common syntax.
- Researching packages for use in R can be time consuming.
- There are many dependencies between R libraries.
- Poorly written R code runs slowly.
- R is not as popular as Python for deep learning and NLP - though this may change over time.

- Python has fewer purpose-specific libraries for data science than R.
- Working in Python requires rigorous testing, as errors show up in runtime.
- Visualizations in Python are more convoluted and not as eye-pleasing or informative as in R.

DIGGING DEEPER

Examples


```
#load libraries
library(datasets)
library(dplyr)

#load the iris dataset
data(iris)
# print iris dataset summary
summary(iris)
# convert names to lower case
names(iris) <- tolower(names(iris))

# filter the data for species virginica
virginica <- filter(iris, species == "virginica")
head(virginica) # this displays the first six rows

# select the specified columns
selected <- select(iris, sepal.length, sepal.
width, petal.length)
# display the top 3 selected columns
head(selected, 3)

# create histogram of values for sepal length
hist(iris$sepal.length,
     col='steelblue',
     main='Histogram',
     xlab='Length',
     ylab='Frequency')
```

```
# import statements
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# load the iris dataset
df = pd.read_csv('iris.csv')
# display data summary
df.info()

# filter the data for species virginica
df = df[df['species'] == 'virginica']
# display the top 5 columns
df.head()

# select the specified columns
selected = df[['sepal_length', 'sepal_width',
'petal_length']]
# display the top 3 selected columns
selected.head()

# create histogram of values for sepal length
sns.set_theme(style = 'darkgrid')
sns.histplot(data = df, x = 'sepal_length')
plt.show()
```

Use Cases

R is frequently used when the data analysis tasks require standalone computing or analysis on individual servers.

For exploratory work, R is easier for beginners. Statistical models can be written with a few lines of code.

R is great for data analysis because of its huge number of packages, readily usable tests, and the advantage of using formulas.

R can handle basic data analysis without needing to install packages. Big datasets require the use of packages such as data.table and dplyr.

RStudio is the most popular R IDE, available in two formats: RStudio Desktop, which runs locally as a regular desktop application and RStudio Server, accessed via web browser, running on a remote Linux server.

Python is frequently used when the data analysis tasks need to be integrated with web apps or if statistics code needs to be incorporated into a production database.

As a full-fledged programming language, Python is a good tool to implement algorithms for use in production.

Python requires users to install packages for data analysis. In recent years, packages have improved greatly.

NumPy and pandas, among others, are popular packages for data analysis.

Developers have created many Python IDEs that drastically reduce the overhead of organizing code, output, and notes files. Jupyter Notebooks and Spyder are popular, as is Jupyter Lab, the data science IDE for Python.

Popular Libraries and Packages

- dplyr, tidyr and data.table to easily manipulate data
- stringr to manipulate strings
- zoo to work with regular and irregular time series
- ggplot2 to visualize data
- caret for machine learning

- pandas to easily manipulate data
- SciPy and NumPy for scientific computing
- Scikit-learn for machine learning
- Matplotlib and seaborn to make graphics
- statsmodels to explore data, estimate statistical models, and perform statistical tests and unit tests

Support and Communities

- R projects
- Intro to R
- Getting started with R
- Code Academy
- DataCamp Slack Community
- Stack Overflow
- Reddit rstats
- Rdocumentation
- R-help
- ROpenSci
- Jumping Rivers list of local R User Groups

- Intro to Python
- Getting started with Python
- Learn Python Basics
- Code Academy
- DataCamp Slack Community
- Stack Overflow
- Reddit Python
- PyLadies
- pydata
- pystatmodels
- numpy-discussion and sci-py-user

To learn more about WGU programs and courses, particularly Data Analytics, that use these programming languages, visit wgu.edu/online-it-degrees.html