Aug 31 · 12 min read

Which Languages Should You Learn For Data Science?

Data Science is an exciting field to work in, combining advanced statistical and quantitative skills with real-world programming ability. There are many potential programming languages that the aspiring data scientist might consider specializing in.

While there is no correct answer, there are several things to take into consideration. Your success as a data scientist will depend on many

points, including:

Specificity

When it comes to advanced data science, you will only get so far reinventing the wheel each time. Learn to master the various packages and modules offered in your chosen language. The extent to which this is possible depends on what domain-specific packages are available to you in the first place!

Generality

A top data scientist will have good all-round programming skills as well as the ability to crunch numbers. Much of the day-to-day work in data science revolves around sourcing and processing raw data or 'data cleaning'. For this, no amount of fancy machine learning packages are going to help.

Productivity

In the often fast-paced world of commercial data science, there is much to be said for getting the job done quickly. However, this is what enables technical debt to creep in—and only with sensible practices can this be minimized.

Performance

In some cases it is vital to optimize the performance of your code, especially when dealing with large volumes of mission-critical data. Compiled languages are typically much faster than interpreted ones; likewise statically typed languages are considerably more fail-proof than dynamically typed. The obvious trade-off is against productivity.

To some extent, these can be seen as a pair of axes (Generality-Specificity, Performance-Productivity). Each of the languages below fall somewhere on these spectra.

With these core principles in mind, let's take a look at some of the more popular languages used in data science. What follows is a combination of research and personal experience of myself, friends and colleagues —but it is by no means definitive! In approximately order of popularity, here goes:

R

What you need to know

Released in 1995 as a direct descendant of the older S programming language, R has since gone from strength to strength. Written in C, Fortran and itself, the project is currently supported by the R Foundation for Statistical Computing.

License

Free!

Pros

- Excellent range of high-quality, domain specific and open source packages. R has a package for almost every quantitative and statistical application imaginable. This includes neural networks, non-linear regression, phylogenetics, advanced plotting and many, many others.
- The base installation comes with very comprehensive, in-built statistical functions and methods. R also handles matrix algebra particularly well.
- Data visualization is a key strength with the use of libraries such as ggplot2.

Cons

- Performance. There's no two ways about it, R is not a quick language.
- Domain specificity. R is fantastic for statistics and data science purposes. But less so for general purpose programming.
- Quirks. R has a few unusual features that might catch out programmers experienced with other languages. For instance: indexing from 1, using multiple assignment operators, unconventional data structures.

Verdict—"brilliant at what it's designed for"

R is a powerful language that excels at a huge variety of statistical and data visualization applications, and being open source allows for a very active community of contributors. Its recent growth in popularity is a testament to how effective it is at what it does.

Python

What you need to know

Guido van Rossum introduced Python back in 1991. It has since become an extremely popular general purpose language, and is widely used within the data science community. The major versions are currently 3.6 and 2.7.

License

Free!

Pros

- Python is a very popular, mainstream general purpose programming language. It has an extensive range of purpose-built modules and community support. Many online services provide a Python API.
- Python is an easy language to learn. The low barrier to entry makes it an ideal first language for those new to programming.
- Packages such as pandas, scikit-learn and Tensorflow make Python a solid option for advanced machine learning applications.

Cons

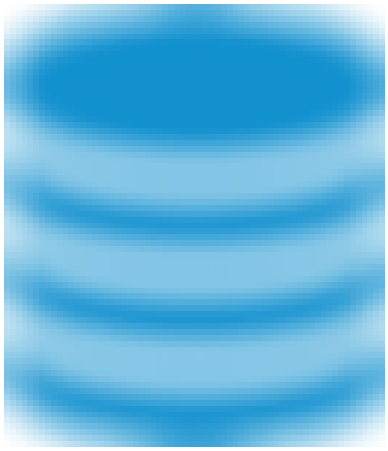
- Type safety: Python is a dynamically typed language, which means you must show due care. Type errors (such as passing a String as an argument to a method which expects an Integer) are to be expected from time-to-time.
- For specific statistical and data analysis purposes, R's vast range of packages gives it a slight edge over Python. For general purpose languages, there are faster and safer alternatives to Python.

Verdict—"excellent all-rounder"

Python is a very good choice of language for data science, and not just at entry-level. Much of the data science process revolves around the ETL process (extraction-transformation-loading). This makes Python's generality ideally suited. Libraries such as Google's Tensorflow make Python a very exciting language to work in for machine learning.

SQL

What you need to know



SQL ('Structured Query Language') defines, manages and queries relational databases. The language appeared by 1974 and has since undergone many implementations, but the core principles remain the same.

License

Varies—some implementations are free, others proprietary

Pros

- Very efficient at querying, updating and manipulating relational databases.
- Declarative syntax makes SQL an often very readable language . There's no ambiguity about what `SELECT name FROM users WHERE age > 18` is supposed to do!
- SQL is very used across a range of applications, making it a very useful language to be familiar with. Modules such as SQLAlchemy make integrating SQL with other languages straightforward.

Cons

- SQL's analytical capabilities are rather limited—beyond aggregating and summing, counting and averaging data, your options are limited.
- For programmers coming from an imperative background, SQL's declarative syntax can present a learning curve.

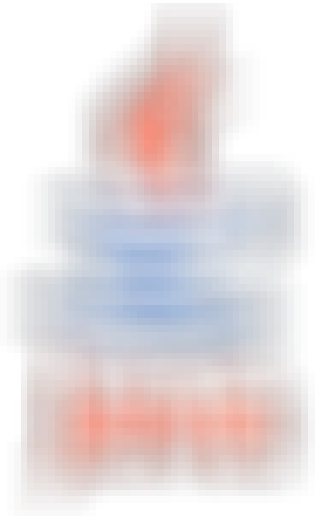
- There are many different implementations of SQL such as PostgreSQL, SQLite, MariaDB . They are all different enough to make inter-operability something of a headache.

Verdict—"timeless and efficient"

SQL is more useful as a data processing language than as an advanced analytical tool. Yet so much of the data science process hinges upon ETL, and SQL's longevity and efficiency are proof that it is a very useful language for the modern data scientist to know.

Java

What you need to know



Java is an extremely popular, general purpose language which runs on the (JVM) Java Virtual Machine. It's an abstract computing system that enables seamless portability between platforms. Currently supported by Oracle Corporation.

License

Version 8—Free! Legacy versions, proprietary.

Pros

- Ubiquity . Many modern systems and applications are built upon a Java back-end. The ability to integrate data science methods

directly into the existing codebase is a powerful one to have.

- Strongly typed. Java is no-nonsense when it comes to ensuring type safety. For mission-critical big data applications, this is invaluable.
- Java is a high-performance, general purpose, compiled language . This makes it suitable for writing efficient ETL production code and computationally intensive machine learning algorithms.

Cons

- For ad-hoc analyses and more dedicated statistical applications, Java's verbosity makes it an unlikely first choice. Dynamically typed scripting languages such as R and Python lend themselves to much greater productivity.
- Compared to domain-specific languages like R, there aren't a great number of libraries available for advanced statistical methods in Java.

Verdict—"a serious contender for data science"

There is a lot to be said for learning Java as a first choice data science language. Many companies will appreciate the ability to seamlessly integrate data science production code directly into their existing codebase, and you will find Java's performance and and type safety are real advantages. However, you'll be without the range of stats-specific packages available to other languages. That said, definitely one to consider—especially if you already know one of R and/or Python.

Scala

What you need to know

Developed by Martin Odersky and released in 2004, Scala is a language which runs on the JVM. It is a multi-paradigm language, enabling both object-oriented and functional approaches. Cluster computing framework Apache Spark is written in Scala.

License

Free!

Pros

- Scala + Spark = High performance cluster computing. Scala is an ideal choice of language for those working with high-volume data sets.
- Multi-paradigmatic: Scala programmers can have the best of both worlds. Both object-oriented and functional programming paradigms available to them.
- Scala is compiled to Java bytecode and runs on a JVM. This allows inter-operability with the Java language itself, making Scala a very powerful general purpose language, while also being well-suited for data science.

Cons

- Scala is not a straightforward language to get up and running with if you're just starting out. Your best bet is to download sbt and set up an IDE such as Eclipse or IntelliJ with a specific Scala plug-in.

- The syntax and type system are often described as complex. This makes for a steep learning curve for those coming from dynamic languages such as Python.

Verdict—"perfect, for suitably big data"

When it comes to using cluster computing to work with Big Data, then Scala + Spark are fantastic solutions. If you have experience with Java and other statically typed languages, you'll appreciate these features of Scala too. Yet if your application doesn't deal with the volumes of data that justify the added complexity of Scala, you will likely find your productivity being much higher using other languages such as R or Python.

Julia

What you need to know

Released just over 5 years ago, Julia has made an impression in the world of numerical computing. Its profile was raised thanks to early adoption by several major organizations including many in the finance industry.

License

Free!

Pros

- Julia is a JIT ('just-in-time') compiled language, which lets it offer good performance. It also offers the simplicity, dynamic-typing and scripting capabilities of an interpreted language like Python.
- Julia was purpose-designed for numerical analysis. It is capable of general purpose programming as well.
- Readability. Many users of the language cite this as a key advantage

Cons

- Maturity. As a new language, some Julia users have experienced instability when using packages. But the core language itself is reportedly stable enough for production use.
- Limited packages are another consequence of the language's youthfulness and small development community. Unlike long-established R and Python, Julia doesn't have the choice of packages (yet).

Verdict—"one for the future"

The main issue with Julia is one that cannot be blamed for. As a recently developed language, it isn't as mature or production-ready as its main alternatives Python and R. But, if you are willing to be patient, there's every reason to pay close attention as the language evolves in the coming years.

MATLAB

What you need to know

MATLAB is an established numerical computing language used throughout academia and industry. It is developed and licensed by MathWorks, a company established in 1984 to commercialize the software.

License

Proprietary—pricing varies depending on your use case

Pros

- Designed for numerical computing. MATLAB is well-suited for quantitative applications with sophisticated mathematical requirements such as signal processing, Fourier transforms, matrix algebra and image processing.
- Data Visualization. MATLAB has some great inbuilt plotting capabilities.
- MATLAB is often taught as part of many undergraduate courses in quantitative subjects such as Physics, Engineering and Applied Mathematics. As a consequence, it is widely used within these fields.

Cons

- Proprietary licence. Depending on your use-case (academic, personal or enterprise) you may have to fork out for a pricey licence. There are free alternatives available such as Octave. This is something you should give real consideration to.

- MATLAB isn't an obvious choice for general-purpose programming.

Verdict—"best for mathematically intensive applications"

MATLAB's widespread use in a range of quantitative and numerical fields throughout industry and academia makes it a serious option for data science. The clear use-case would be when your application or day-to-day role requires intensive, advanced mathematical functionality; indeed, MATLAB was specifically designed for this.

Other Languages

There are other mainstream languages that may or may not be of interest to data scientists. This section provides a quick overview… with plenty of room for debate of course!

C++

C++ is not a common choice for data science, although it has lightning fast performance and widespread mainstream popularity. The simple reason may be a question of productivity versus performance.

As one Quora user puts it:

> "If you're writing code to do some ad-hoc analysis that will probably only be run one time, would you rather spend 30 minutes writing a program that will run in 10 seconds, or 10 minutes writing a program that will run in 1 minute?"

The dude's got a point. Yet for serious production-level performance, C++ would be an excellent choice for implementing machine learning algorithms optimized at a low-level.

Verdict—"not for day-to-day work, but if performance is critical…"

JavaScript

With the rise of Node.js in recent years, JavaScript has become more and more a serious server-side language. However, its use in data science and machine learning domains has been limited to date

(although checkout brain.js and synaptic.js!). It suffers from the following disadvantages:

- Late to the game (Node.js is only 8 years old!), meaning…
- Few relevant data science libraries and modules are available. This means no real mainstream interest or momentum
- Performance-wise, Node.js is quick. But JavaScript as a language is not without its critics.

Node's strengths are in asynchronous I/O, its widespread use and the existence of languages which compile to JavaScript. So it's conceivable that a useful framework for data science and realtime ETL processing could come together. The key question is whether this would offer anything different to what already exists.

Verdict—"there is much to do before JavaScript can be taken as a serious data science language"

Perl

Perl is known as a 'Swiss-army knife of programming languages', due to its versatility as a general-purpose scripting language. It shares a lot in common with Python, being a dynamically typed scripting language. But, it has not seen anything like the popularity Python has in the field of data science.

This is a little surprising, given its use in quantitative fields such as bioinformatics. Perl has several key disadvantages when it comes to data science. It isn't stand-out fast, and its syntax is famously unfriendly. There hasn't been the same drive towards developing data science specific libraries. And in any field, momentum is key.

Verdict—"a useful general purpose scripting language, yet it offers no real advantages for your data science CV"

Ruby

Ruby is another general purpose, dynamically typed interpreted language. Yet it also hasn't seen the same adoption for data science as has Python.

This might seem surprising, but is likely a result of Python's dominance in academia, and a positive feedback effect . The more people use Python, the more modules and frameworks are developed, and the more people will turn to Python. The SciRuby project exists to bring scientific computing functionality, such as matrix algebra, to Ruby. But for the time being, Python still leads the way.

Verdict—"not an obvious choice yet for data science, but won't harm the CV"

Conclusion

Well, there you have it—a quickfire guide to which languages to consider for data science. The key here is to understand your usage requirements in terms of generality vs specificity, as well as your personal preferred development style of performance vs productivity.

I use R, Python and SQL on a regular basis, as my current role largely focuses on developing existing data pipeline and ETL processes. These languages give the right balance of generality and productivity to do the job, with the option of using R's more advanced statistics packages when needed.

However—you may already have some experience with Java. Or you may want to use Scala for big data. Or, perhaps you're keen to get involved with the Julia project.

Maybe you learned MATLAB at university, or want to give SciRuby a chance? Perhaps you have an altogether different suggestion! If so, please leave a reply below—I look forward to hearing from you!

Thanks for reading!