R vs Python for Data Science and Visualization

Originally published at https://www.iflexion.com/blog/r-vs-python-data-science

Published: November 23, 2020 | By Martin Anderson

Python and R are frequently pitted against each other as if one or the other might eventually become the Betamax[1] of programming languages for data science analysis.

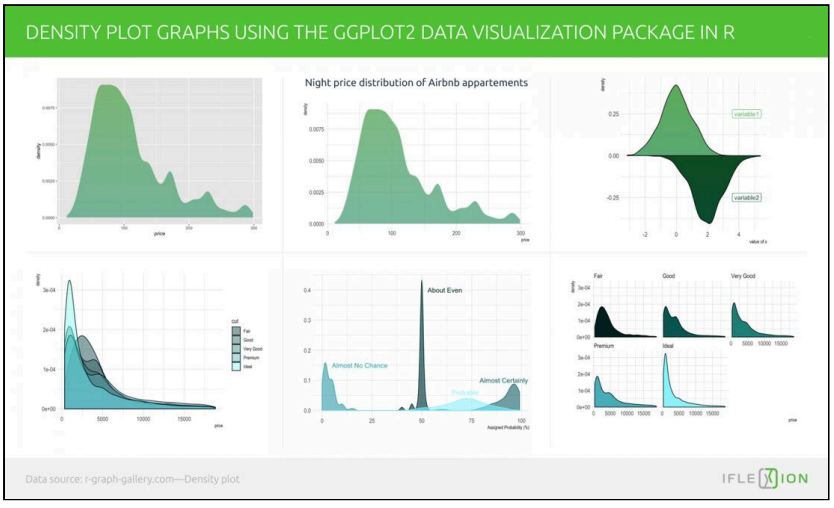
If economy of scale and the growth of market share were the only criteria, we could award Python as the winner without further consideration. However, comparing R and Python in this way may be a false equivalency that's compounded by the impact of the GPU-accelerated machine learning revolution of the last ten years on both and by the evolving confluence between data science and machine learning. Neither framing the languages' adherents as 'rivals' nor the arguable unique strengths of each account for the interoperability that can be achieved between them.

In this article we'll consider what Python and R currently have to offer for enterprise deep neural networks and data science consulting, and how emerging trends may affect their uptake.

R for Data Science

R evolved in the mid-1990s from S[2], a statistical programming language popular in the 1980s, as statistical analysis became a driving feature of the new business computing revolution.

Among its core strengths, R provides a very high-quality and configurable graphing output even on a base install.



R is an extensible language, with more than 20,000 available user-contributed extensions[3]. Areas covered include finance, genetics, econometrics, medical imaging, machine learning, psychometrics and social sciences, among many others[4]. Packages are archived and distributed from the Comprehensive R Archive Network (CRAN)[5].

In 2010, the release of the RStudio integrated development environment (IDE) gave CLI-based R a more user-friendly but powerful interface, while the curated Tidyverse collection of R packages for data science became a standard environment for selecting and using the most powerful and popular adjuncts to R.

Python for Data Science

Python is an interpreted and dynamically-typed language developed from the ABC language in the late 1980s. Like R, Python is highly extensible, with its popularity driven by the extraordinary range of additional functionality from over 137,000 available libraries[6].

Python's ability to create native and cohesive frameworks has driven it to the heart of the machine learning revolution over the last 15 years. Pivotal AI libraries such as TensorFlow, Pandas, Scikit-learn, and NumPy are either written in Python or support it as a priority over other languages, such as C.

In the same period, a confluence was forming between machine learning and data science, as the power of high-scale deep neural networks transformed a culture of 'local' and limited analysis into the prospect of developing an industrialized AI-driven panopticon of statistical insight.

Thus, over the last decade, Python's strengths as an analysis platform have met up with its vanguard position in machine learning development, recently pushing the language to the first position (57%) on GitHub's survey of machine learning languages, with rival R lagging in eighth place[7].

Besides those mentioned above, popular Python data science libraries include PyTorch and Keras, as well as staples such as Matplotlib, Seaborn, Pydot (for the C-written GraphViz data visualization library), SciPy, and the Beautiful Soup HTML parser for web-scraping.

The Python Package Index (PyPi) is the central library repository and equivalent to R's CRAN. It's a crowded place, with a wider scope than data science, and therefore PyForest offers a filter for the automatic import of PyPi data science libraries.

R or Python for data science?

Interoperability

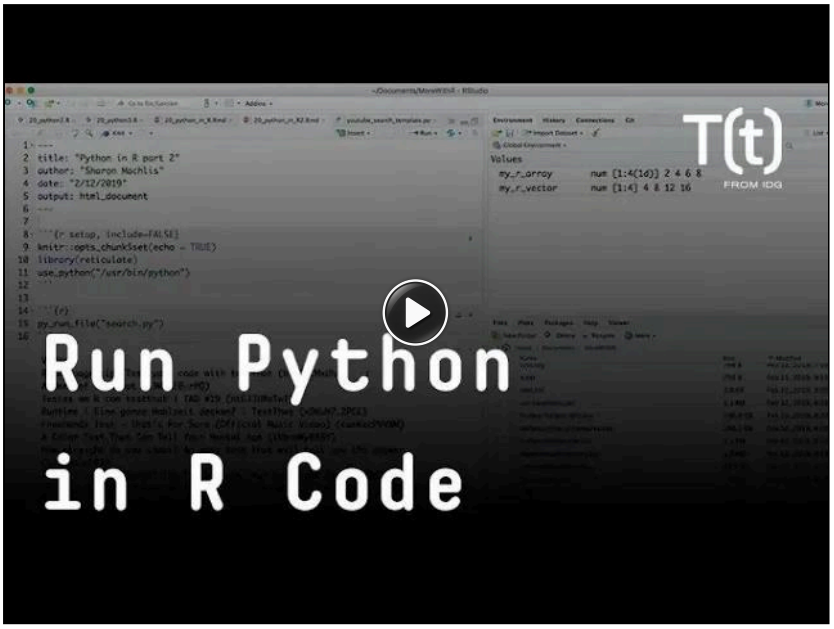
Running R from Python

The rpy2 repository provides full-featured access to the entire R functionality from within Python, translating R's objects into Python functions, with transparent conversion to Pandas and NumPy data structures.

Running Python from R

Reticulate is the most popular method to access Python functionality from within R. Reticulate inserts a Python session directly within an R session, allowing calls to Python by various methods:

- Importing Python modules and gaining direct access to its functions.
- Using the R Markdown language engine as a bi-directional interchange between R and Python.
- Sourcing Python scripts with seamless object/function translation, allowing the same functionality as calling an R script.



Additionally, the RTorch repository provides a wrapper to PyTorch, effectively combining all the capabilities of each language.

Machine Learning and Statistical Libraries Across R and Python

Both R and Python offer a comprehensive array of importable or native libraries covering use cases for data science analysis, from the most popular functions to the most arcane. Since interoperability between the two languages is well-facilitated (see above), there is no appreciable gap in functionality. Some libraries, such as XGBoost, are equally available to either language.

In general, Python relies on the machine learning library Scikit-learn for the majority of data science-related functionality, while CRAN is the equivalent external source for R. Since R was created with statistical analysis in mind, it features a slightly higher range of related native functions.

Graphing

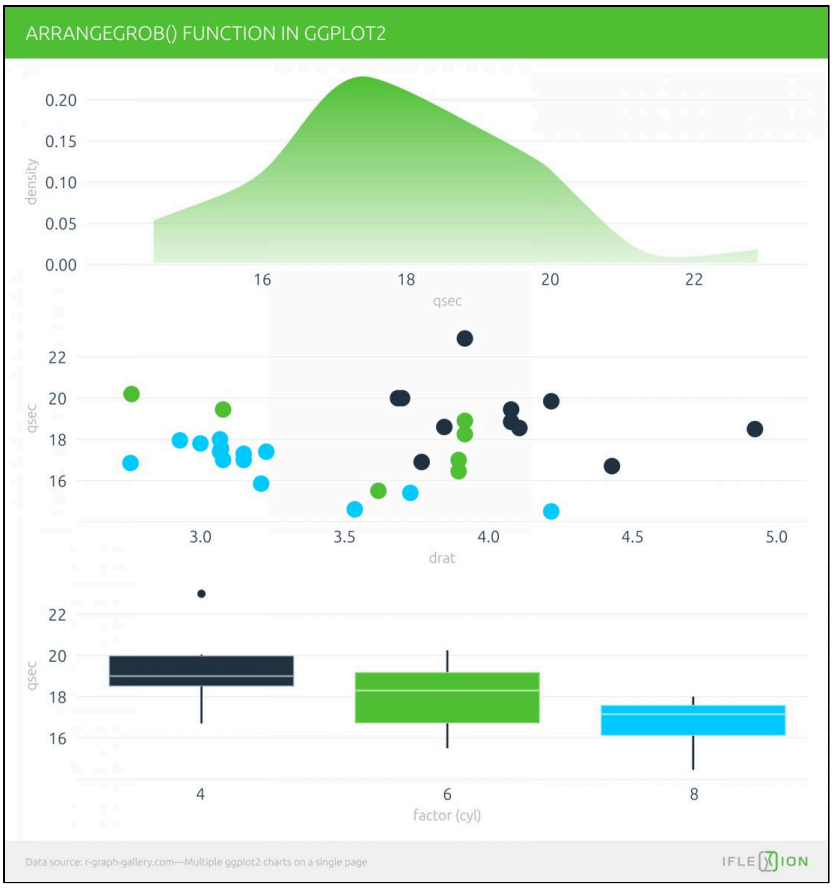
Graphing and Visual Data Exploration in R

Though R's native Graphics package provides 100 functions for generating histograms, scatterplots, and boxplots, the far more sophisticated capabilities of the ggplot2[40] package have arguably been the driver for uptake.

Ggplot2 provides a layered and logical sequential approach to data visualization development, with aesthetic components chosen and configured after essential elements such as axes and data positioning are established.

At the time of writing there are 80 registered extensions for ggplot2[41], offering functionality as diverse as spellchecking, theme additions, HTML embedding, facility for survival curves and alluvial diagrams, time series visualizations, partial rasterizations and visualization of IP addresses and networks.

In addition to a versatile range of styles and configurations, graphs can be nested for single implementation into a document or interface:



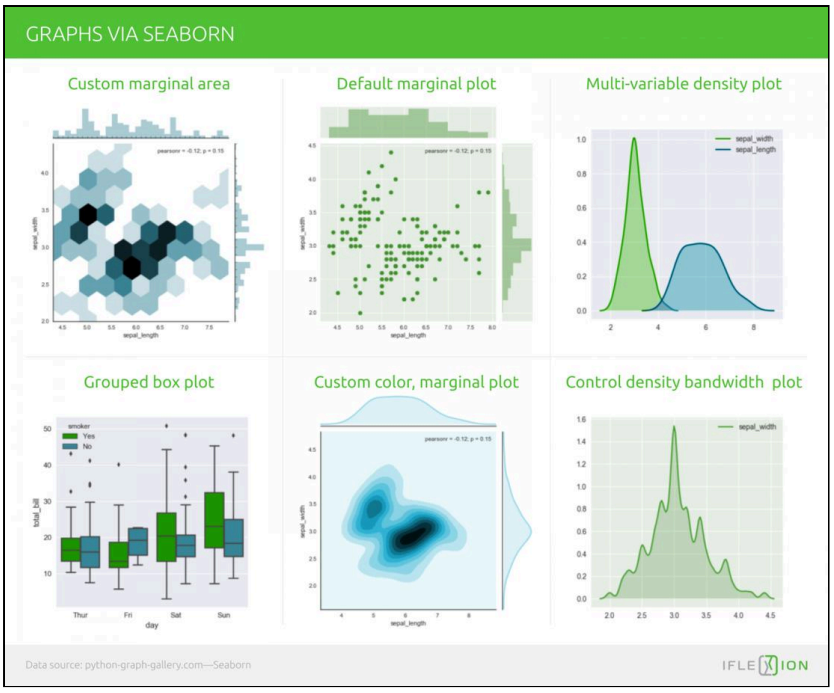
The gganimate library can also compile sequential or time data into elegant animations, created as GIF files or passed to the FFMPEG renderer for video output.

Graphing and Visual Data Exploration in Python

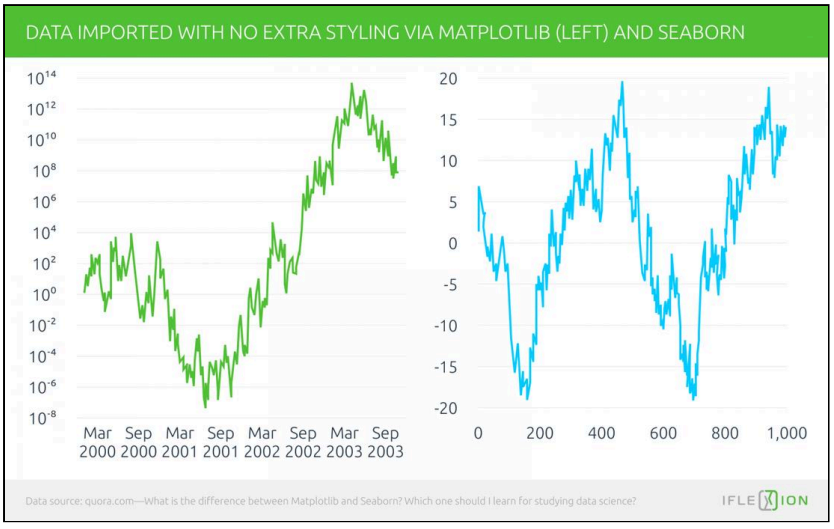
In keeping with its generalized scope as an all-purpose programming language, Python defers to external libraries such as Matplotlib to provide an equivalent graphing functionality to that of R.

Matplotlib is an open-source Python derivation of the proprietary MATLAB programming environment. Similar to R's ggplot2 extensions repository (see above), it is greatly enhanced by a comprehensive range of third-party mapping toolkits[42].

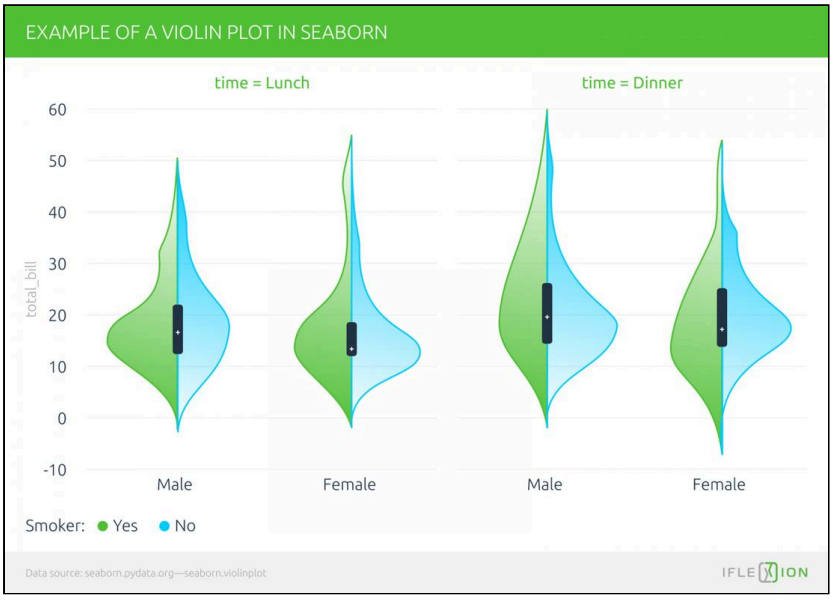
Additional functionality available via these libraries includes the superimposition of data on map projections; facility for interactive data cursors; support for cross-platform visualization applications; enhanced table support (compared to Matplotlib's native tables); support for annotated DNA and astronomy data maps; ridge maps; and non-contiguous graph axes.



However, arguably the most important additions are the ggplot library, which ports the entire graphics-driven philosophy and utility of R's ggplot2 to native Python, and Seaborn, which not only facilitates Trellis graphs (see below), but provides a super-layer of statistical graphics capability to Matplotlib via a high-level API that's easier to use and more up-to-date.

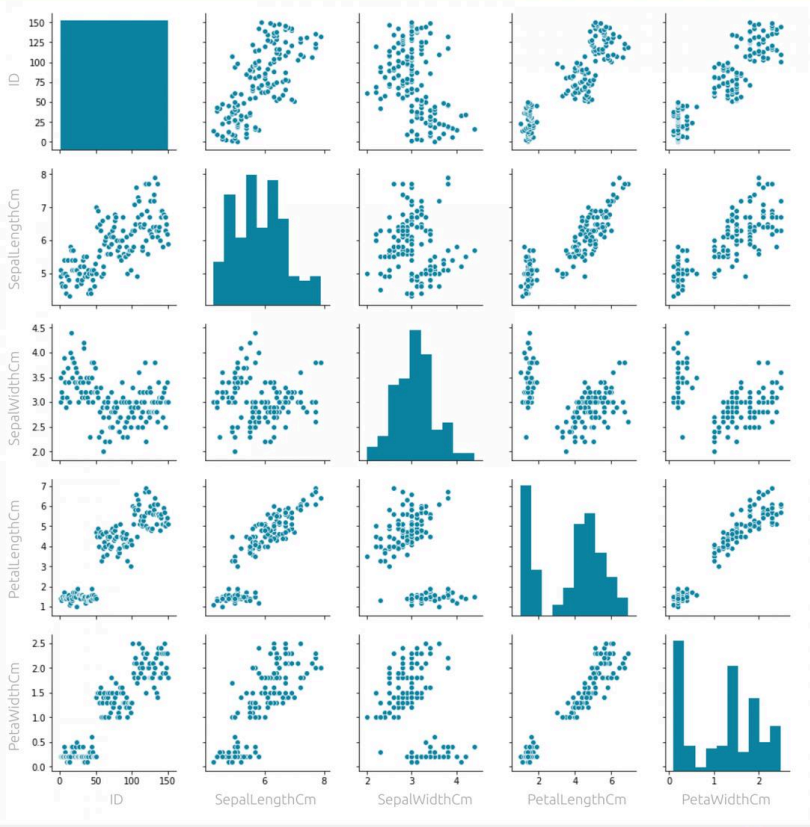


Seaborn offers greater control over color use in graphs, essential in differentiating data streams, as well as offering additional plot types, such as the Violin plot:



Furthermore, Seaborn's pairplot() function, among others, enables at-a-glance exploration of data prior to more sophisticated choices for graph output, and with less code than is necessary in R[43]:

FACILE INITIAL DATA EXPLORATION WITH SEABORN

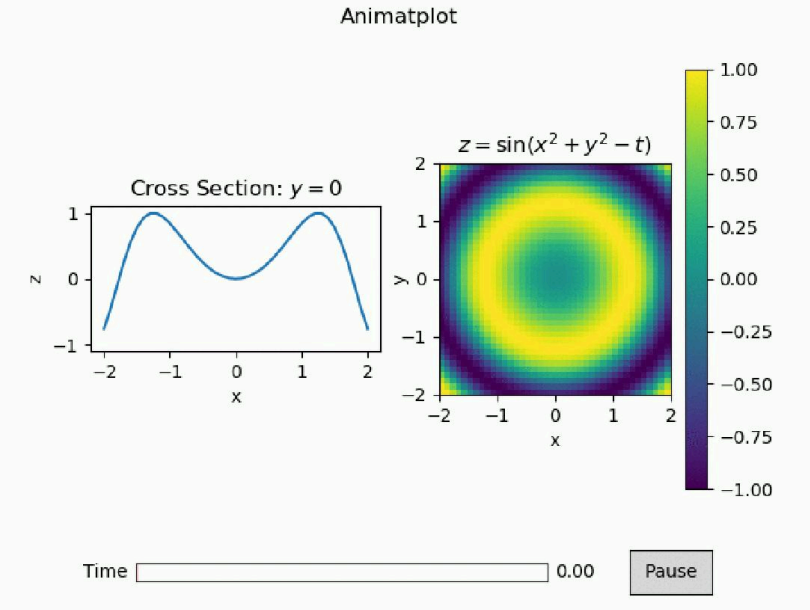


Data source: inwt-statistics.com—Data Visualization in R vs. Python, 2019

IFLE ION

Animation is also catered for via the Animatplot library, an extension for Matplotlib, as well as being natively supported in Matplotlib itself[44].

ANIMATED GRAPHS VIA ANIMATPLOT

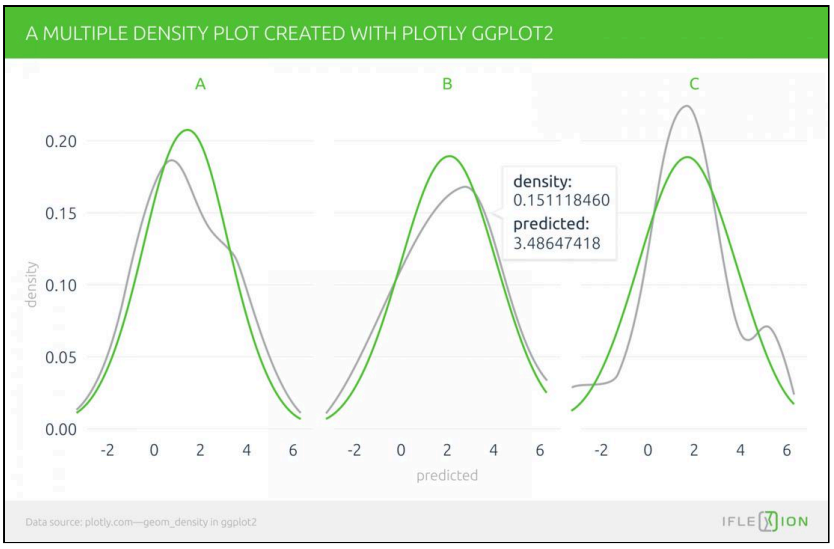


Data source: animatplot.readthedocs.io—Animatplot

IFLE ION

Plotly for R and Python

Canadian software company Plotly has developed various data visualization libraries for Python and R, including the Dash Python and R (DashR) frameworks for the creation of interactive dashboards and graphics.

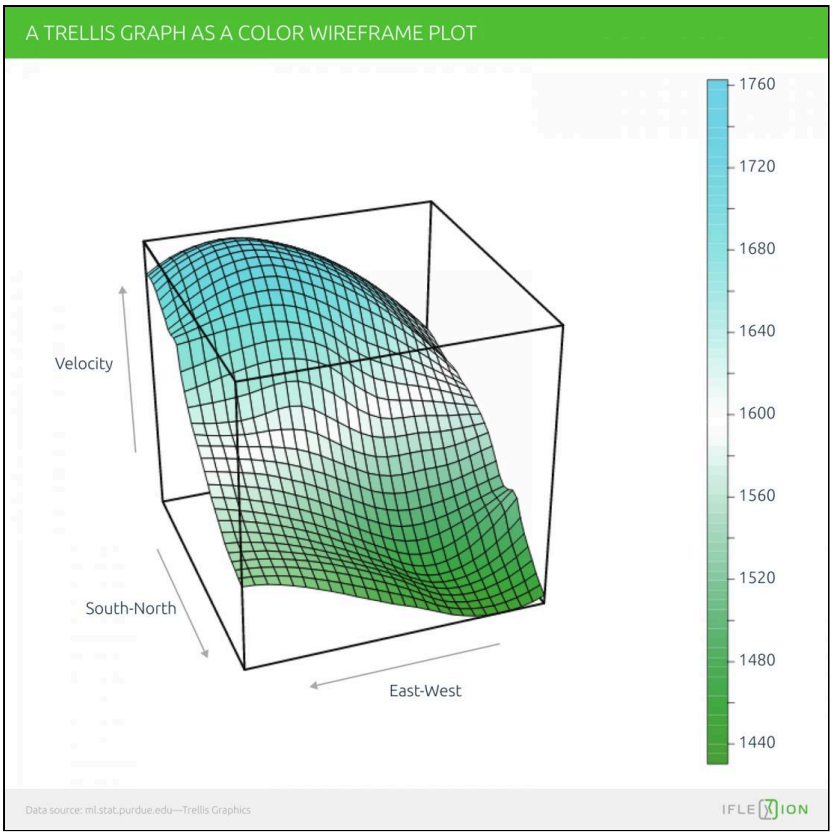


Implementations of Plotly are derived from the plotly.js JavaScript repository, which can output still or animated raster graphics in addition to resolution-independent vector-based SVG files, allowing for more sophisticated and interactive applications, such as PDF-style dynamic documents:

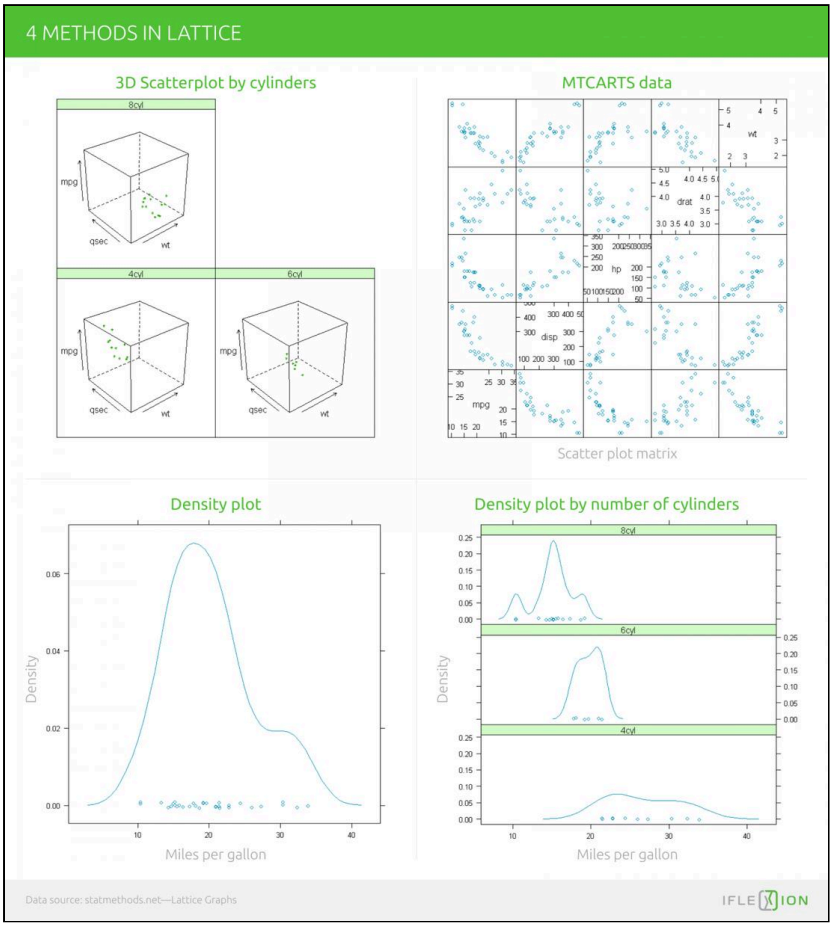
In addition to Python and R implementations, Plotly also supports the Julia programming language.

Trellis Graphs for R and Python

Trellis graphs[45] visualize the relationship between any number of variables, represented as a rectangular array of plots (histograms, box, or scatter).



R's Lattice library is a high-level visualization resource specializing in multivariate data and enabling the creation of highly stylized Trellis graphs, including impactful 3D wireframe graphs.



Available methods include bar chart, boxplot, 3D scatterplot, 3D contour plot, histogram, kernel density plot, 3D level plot, scatterplot matrix, parallel coordinates plot, strip plot, dotplot, scatterplot and 3D wireframe graph.

Besides interoperable builds where Python accesses R's Lattice library transparently (probably the easiest method), there are various ways to generate Trellis graphs in Python, including the LatticeDrawing repository or PyPi's PythonLattice; accessing R's RPlot via Pandas[46]; using Plotly's

subplot capabilities[47]; and using the FacetGrid class in Seaborn[48].

Accessibility

R's tight focus on local data science analysis means that all tooling and all help resources will be in some way pertinent to the task, whereas the signal-to-noise ratio of the Python ecostructure can be an initial obstacle when developing dependencies in a new framework.

However, it's generally acknowledged that R has a steeper initial learning curve than Python, and is less widely applicable to other environments.

Python's object-oriented approach is less specious than R and its learning curve shallower. Additionally, knowledge of other programming languages with a similar paradigm will speed the progress of a Python initiate substantially, but may be less useful when learning R.

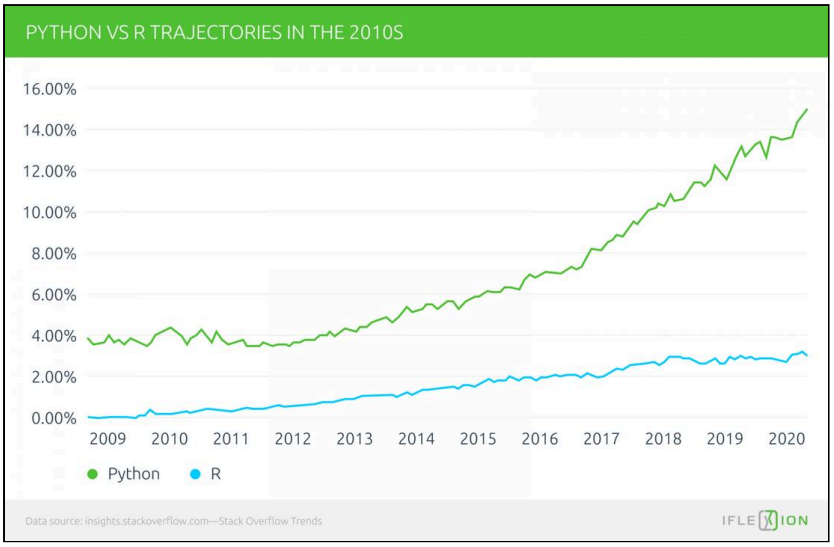
In terms of code complexity, there is very little difference between R and Python[49].

Diffusion and Take-Up

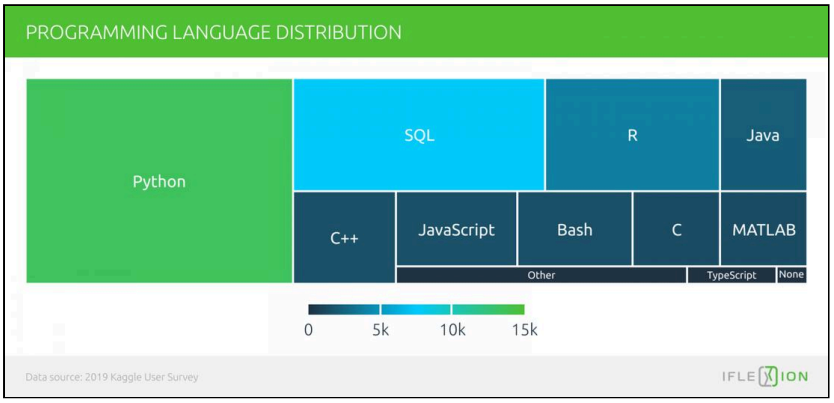
We saw earlier that Python is the most popular machine learning language. According to the Stack Overflow 2020 Developer Survey[50], Python maintains its 2019 position as the fourth most popular general programming language among professional developers at a 41% share, while R drops one place to the 17th position with a 5.5% share.

However, it should be considered that R is a dedicated data science analysis language and Python a heterogeneous programming language with a wider scope, particularly in machine learning — a later consideration in the development of R.

Nonetheless the ascending trajectories of the two languages over ten years is a broad indicator of talent availability and market share in favor of Python:



A recent Kaggle survey indicates[51] that Python comfortably dominates R in terms of adoption for deep learning, covering the majority of work in computer vision and natural language processing (NLP), with a wider diffusion and uptake in the data science community.



R vs Python: Key Points for Comparison

Since, as we have seen, there is no notable shortfall in functionality for data science analysis between Python and R, there are only a limited number of convincing reasons to choose R:

- It has a shallower initial learning curve for pure data science projects.
- The Shiny R package facilitates the easy creation of interactive web applications and dashboards, and is a superior and more mature product than Python's Dash equivalent.
- R uses more native functions, as opposed to Python's classes and secondary libraries.
- R offers a large ecostructure focused on data science and a larger number of libraries than Python for more marginal use cases.

- RStudio is arguably the most mature and complete IDE for data science analysis.
- R has a clearer and more usable versioning system than Python, with much less risk of technical debt[52] or the need to support multiple installations or instances.
- Any pre-existing data science talent on your team may already be familiar with it.
- It is suitable in cases where the ultimate scope and breadth of the project are known and affordable turnkey Python-based solutions are not going to be needed later when the scope develops or the data scales up significantly.
- R has a good market share and loyal entrenchment in specific sectors of the scientific community, with comprehensive and focused help resources.

Conversely, there are stronger indicators for choosing Python over R:

- Clear market saturation makes talent for Python software development more affordable[53] and available than for R.
- Python is a general modular programming language, meaning that infrastructure and deployments can take place in the same popular development environment as the core code.
- Package management, including dependency management, with Pip is superior as opposed to Packrat and other R package managers[54].
- Developmentof high-level C/C++ interfaces is easier[55].
- Python has demonstrated that it has a sufficient community willing to maintain or exceed feature parity with R, while the converse is likely to remain logistically difficult for R in the future.
- A high likelihood that a company's technical teams already have some grounding in Python, even for purposes unrelated to data science.
- Its performance in typical machine learning and general analytical tasks is faster[56,57].
- Core Python-facing libraries such as Matplotlib, Scikit-learn, Keras, SciPy, NumPy and Pandas have grown massively in well-funded industry support and community impetus in the age of GPU-driven machine learning, assuring future support and updates and fewer brittle or precarious interdependencies in future projects.
- There is a wider variety of mature and well-supported IDEs[58] , including RStudio itself since 2018[59].

Conclusion

We began this article by considering whether Python or R might become an outmoded language in data science, or whether the two languages will maintain their current equilibrium. With feature parity between Python and R now a moot point, it can be argued that R will remain on the back foot, and is indeed becoming a 'legacy' approach when considered against the ascendancy of Python in data science analysis, and that R's specialization in this field is no longer a compelling reason to prefer it.

R's specialization in data science analysis is no longer a compelling reason to prefer it over Python.