Technical Guide

Helping you to get the most out of the
Apteco Marketing Suite™

Python, R, and Apteco
D080T1X001 - May 2018

Revision Tracking Sheet

This manual may be revised periodically to incorporate new or updated information.

Listed below is the revision date of each page (if applicable):

PageRevision

All pages May 2018

Contents

Chapter 1.Introduction

This document is the product of some preliminary research undertaken prior to starting development work on the FastStats Python integration. The aim was to gain a better understanding of Python and R as programming languages, their current position and usage in the Data Science world, and the implications for Apteco in offering integrations with these languages. This material was originally delivered as a *Scientia* presentation at Apteco on 1st December 2017, entitled *"There's a Snake in the Graph!".*

Chapter 2.Meet the languages

Introduction to Python and R

2.1 Python

Python was released in 1991, and was inspired by the programming languages C, Modula-3, and in particular, ABC. Just as any good biography of a person starts with their parents, so too it can be instructive to know about the inventor of a programming language. Python was created by Guido van Rossum, a Dutchman. He has worked for various software and tech companies, including Google and, at the time of writing, Dropbox. This says something about the culture around Python – open source, with a kind of modern, cool, tech-y vibe. Another clue to Python's culture is its name; this came from *Monty Python's Flying Circus*, with no particular rationale other than that Guido van Rossum was a fan of the show and in an "irreverent mood" when he first created Python.

The development of Python is now overseen by the Python Software Foundation, with Guido van Rossum as "Benevolent Dictator for Life" (BDFL). Dropbox allow him 50% of his time to work on Python.[i]

2.2 What's it like?

One of the first things people are likely to learn about Python is that it uses "significant whitespace"; whereas other languages use curly brackets or keywords to delimit code blocks, Python uses whitespace indentation. While other languages have conventions about indentation to make code more readable, these aren't compulsory. In Python, however, changing indentation will change the meaning of the code, or cause syntax errors. This means that code has readability baked into it.[ii]

In 1999, van Rossum submitted a funding proposal to DARPA called "Computer Programming for Everybody", in which he further defined his goals for Python:[iii]

- an easy and intuitive language just as powerful as major competitors
- open source, so anyone can contribute to its development
- code that is as understandable as plain English
- suitability for everyday tasks, allowing for short development times

This philosophy is well-illustrated by the "Zen of Python", the guiding principles of the design of Python, which are listed in PEP 20 and included as an Easter Egg in the Python interpreter, where they can be displayed by entering the statement **import this**.[iv]

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
```

```
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

While this might seem like a fastidious chasing of ideological purity, the line "practicality beats purity" demonstrates that is not quite the case. Instead, it comes from a very good understanding of people and their interaction with computers in the coding process. Specifically, that "code is read much more often than it's written" (either by the original programmer in the future, or by other programmers – especially in the context of open-source software).[v] Moreover, that programmer time is often more expensive than CPU time and so compromises in code speed and efficiency are often worthwhile for the sake of human readability. For segments of code that are time-sensitive or need to be efficient, implementations in C++ can be written, as is the case for many Python packages.[vi]

Python is an interpreted, dynamically typed, multi-paradigm language.[vii] Being interpreted (that is, it doesn't need compiling before running) allows for quick development and testing. "Multi-paradigm" includes object-oriented, and a lot of Python packages are object-oriented. This is great for building programmes in a modular way, enabling them to be built up and extended easily. In a similar vein, it has a very small core of built-in functions (76 in Python 2.7, 68 in Python 3.6), with a large standard library offering more functionality.

Some standard library packages and other commonly-used packages:

Package	Use
random	generating random numbers
os	interacting with operating system
datetime	working with dates and times
sqlite3	creating, editing and reading SQLite databases
django	developing web applications
NumPy	scientific computing, especially with arrays
SciPy	computing for science, mathematical and engineering
PyGame	making games

2.3 Example code

Here is an example of a Python script – a simple number-guessing game:[viii]

```
import random

guesses_made = 0

name = raw_input('Hello! What is your name?\n')

number = random.randint(1, 20)
print 'Well, {0}, I am thinking of a number between 1 and 20.'.format(
    name)

while guesses_made < 6:

    guess = int(raw_input('Take a guess: '))

    guesses_made += 1

    if guess < number:
        print 'Your guess is too low.'

    if guess > number:
        print 'Your guess is too high.'

    if guess == number:
        break

if guess == number:
    print 'Good job, {0}! You guessed my number in {1}
guesses!'.format(
        name, guesses_made)
else:
    print 'Nope. The number I was thinking of was {0}'.format(number)
```

Some examples of things written in Python:

Name	Details
YouTube	video sharing website
The Onion	satirical news website
The Washington Post	news website (using the **django** web framework)
Eve Online	video game set in space
Paint Shop Pro	image editing software package
Google	(significant number of applications at Google use Python)
Civilisation IV	turn-based simulation game

2.4 R

R was released in 1995, so is a similar age to Python. It was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. Ihaka and Gentleman were both statistics professors, and again, this is illustrative of its culture and usage; it is used a lot in statistical fields and in academia.

It was initially created as an implementation of the S language, and its name originated as a joke on this (R being adjacent in the alphabet), as well as the initial letter of both creators' first names. It is now developed by the R Development Core Team, of member of which is John Chambers, who had developed S at Bell Labs.

Chapter 3.The showdown: R vs Python

Comparison of Python and R

3.1 Key questions

We will use four key questions to compare Python and R:

- What is it good for?
- What tasks would you choose it for?
- Who would choose it?
- What are its drawbacks?

We will be comparing the two languages in both a general sense, as well as in the context of the data science field.[ix]

3.2 What is it good for?

3.2.1 R

- Data handling: this is what R was built for!
- Building data models: can do quite sophisticated things with a couple of lines of code.
- Heavy calculations: very good at churning through lots of data with ease, though there is a caveat to this, which we'll return to later.
- Excellent visualisations: lots of packages for producing publication-quality visualisations.

Map showing number of Facebook friendships between pairs of cities. Producing in R using about 150 lines of code with no external dependencies.

- Can pre-process with anything else: relatively easy to get data from most sources into R.

3.2.2 Python

- Anything and everything! Being a general programming language, it could be used for almost anything you might want to.
- Particularly strong in fields of maths, science and engineering: by making use of the **NumPy** and **SciPy** packages you could use it for almost any task you would use MatLab for.

3.3 What tasks would you choose it for?

3.3.1 R

- Standalone computing/analysis: R is very powerful and is a good choice especially if you're working with your data in isolation.
- Exploratory work: doing command-line data analysis to explore a data set.
- Producing great visualisations: there are plenty of mature packages for producing very high-quality visualisations.

3.3.2 Python

- Pre-processing data: a good choice for ETL (extract, load, transform).
- Integrating with other parts of production flow: because it is a general programming language, it is very flexible and could be used for:
    - Production algorithms
    - Data analysis to be integrated with web apps
- Data handling: while it wasn't built specifically for this, there are great data science packages available, such as **numpy**, **scipy**, **pandas**, **statsmodels**, **matplotlib**, **seaborn**.
- Deep learning/Machine learning: a real strength of Python's, with the popular **scikit-learn** package.

3.4 Who would choose it?

3.4.1 R

- People working in or close to statistics, research or data science.
- People in the academic world: this is where it emerged from, and it is still strong here.
- "Expert" in statistics: it is the go-to language for lots of statistical things, so statisticians may well have experience of it.

- People who already know R: an obvious point, but people are likely to choose to use a language they already know, if they can.

3.4.2 Python

- People with an engineering or programming background.
- Non-expert developers or programmers moving into data science.
- People who already know Python: again, where at all possible, people are likely to avoid having to learn a new language when one they know already will suffice.

3.5 What are its drawbacks?

3.5.1 R

- Steep learning curve: R is comparatively harder to learn.
- Not a programming language: a potentially controversial point, but as R expert John Cook points out R *"is an interactive environment for doing statistics,"* not really a programming language. As he suggests, *"I find it more helpful to think of R as having a programming language than being a programming language."*
- Memory limit: this is the caveat to its heavy calculation power mentioned above. If data exceeds the computer's memory, R gets stuck. This is a well-known issue, and there are packages to deal with this, but it's still something of an inherent architectural flaw.

3.5.2 Python

- Not as strong on data handling and visualisations as R: but has improved a lot in this area in the last few years, with packages like **pandas** for data analysis and **seaborn** for visualisations.
- Perhaps not yet completely mature (with respect to data science). For example, **scikit-learn** seeming to throw CPU power at a problem, rather choosing a more appropriate and precise method for it. (Overall, however, it seems to be quickly "coming of age".)

3.6 Summary comparison

R: Data analysis & Visualisations

- "By statisticians, for statisticians": built specifically for data handling

Python: General programming & Machine learning

High-level general programming

Chapter 4.Python and R in the wild

Use of Python and R in the Data Science world

4.1 Why use either?

How did Python, which many people know as a “beginner” programming language, and R, which originated in academia end up being used in the commercial world?

As the amount of data has grown in the commercial world, companies realised they needed to make use of it. So they looked around to see what people were available to do this.

"As enterprises struggle to put data to work, they’re also struggling to find qualified data scientists. More often than not, however, such data scientists may already work for them and likely have some familiarity with Python." [x]

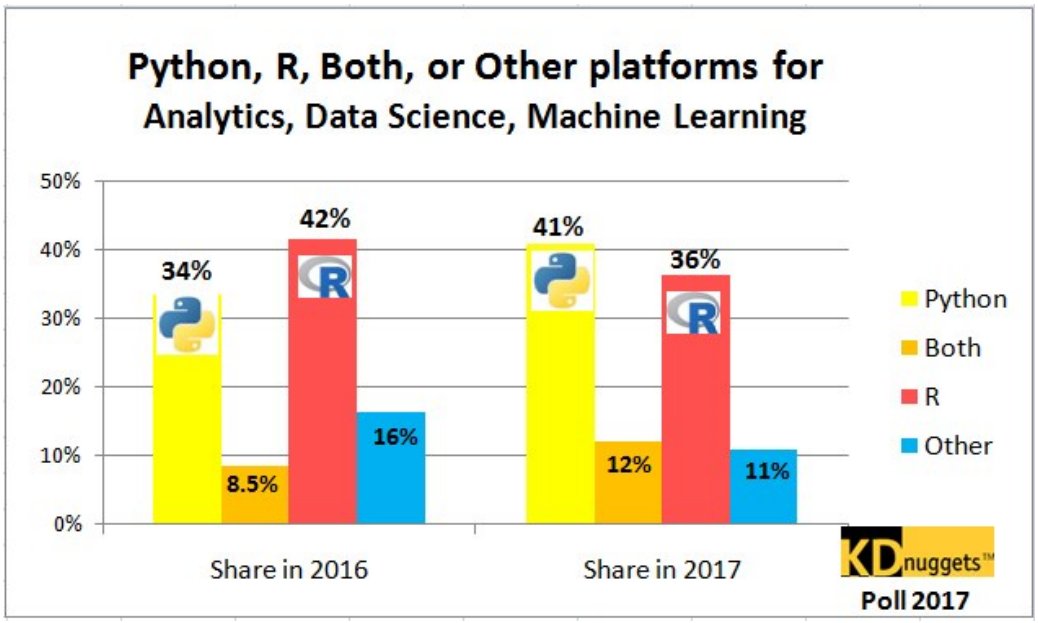
It may be that companies have science graduates hired as analysts, who used Python in their degrees, and so are comfortable using it to do their analysis work.

“I think the number one value to businesses [in using R] is access to talent,” says Smith. “So many businesses now are doing much more with data, especially with the big data revolution and doing much more with analytics. And because they’re hiring people coming out of [university]. They know R already.” [xi]

Similarly, if a company is hiring talented people to work with their data, it could be that this talent simply comes with knowledge of R.

4.2 Some statistics

The graph below, based on a KDnuggets poll, shows that from 2016 to 2017, Python overtook R in having the largest share usage as a platform in the data world. [xii]



KDnuggets graph showing share of platforms in data world in 2016 and 2017

The following diagram, based on the same poll, shows a general overall net shift towards using Python more, as well as R and Python between them taking usage away from other platforms. [xiii]

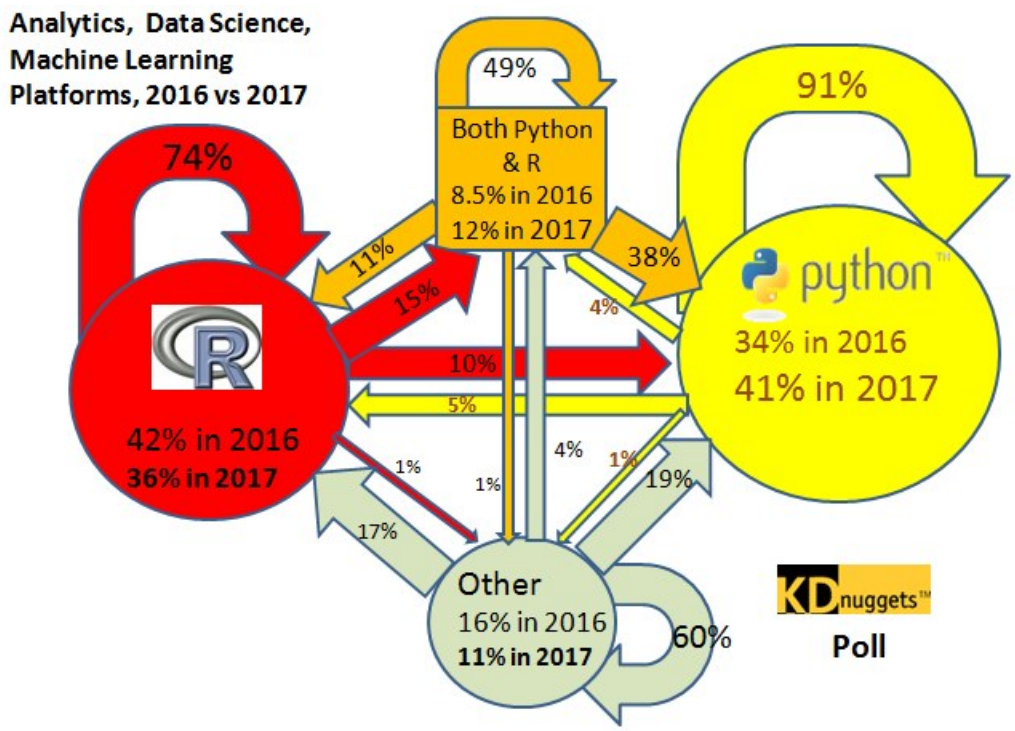


Diagram showing changing usage of different platforms in data world from 2016 to 2017

Survey data from Kaggle.com (data science website).[xiv]

What tools are used at work?

	Python	R
All industries	71.5%	63.1%
CRM/marketing	76.3%	59.2%

So it seems that the CRM/marketing industry has fairly typical Python/R usage compared across all industries.

What language would you recommend new data scientists learn first?

	Python	R
All users	63.1%	24.0%
Python users	89.2%	1.6%
R users	63.3%	19.4%
Users of both R & Python	62.5%	29.7%

Users of one language tend to recommend that language, with Python users having a very strong preference for Python. But users of both languages, and all users taken together, would recommend Python over R in a 2:1 ratio.

It is worth noting the exact wording of the question – it could just be that users of R acknowledge that Python is an easier language to learn, while nevertheless maintaining

a view, for example, that R is worth working towards later, and maybe even better in the long run.

4.3 Why not both?

From the data in the previous section, it is clear that there are many users of each language, and many who use both. So it seems foolish to try and force a binary choice when both have great assets, some of which the other lacks.

4.3.1 Example production flow

Consider the following straightforward data production flow:

Python's versatility and broad functionality as a general-purpose language makes it a strong choice for the pre-processing and aggregation, and equally for creating data products from the analysis; R's strength lies in its data analysis power and visualisation production.

Applying this to our imagined production flow:

4.3.2 Other considerations

A helpful way to think of this is with R as "muscle fibre" and Python as "connective tissue".[xv] This view chimes with common practice internal to Python in having C-implementations for code that needs to run efficiently and [pure] Python code between these components acting as "code glue".

Using R and Python together is made easier by packages/libraries that support this; the **rPython** library in R and the **RPy2** package in Python. Moreover, the creator of the popular Python data analysis package **pandas** has developed a standard data frame format for Python and R, called **Feather.**[xvi]

It is interesting to note that in my research, I came across no articles suggesting movement from Python to R. Many articles were asking the "neutral" question of which language they should learn or use, but there were also some asking the question of "what do I still have to do in R that I can't do in Python?".

Chapter 5.New tools for the workbench?

Implications for Apteco

5.1 Apteco Marketing Suite

Having looked at Python and R in the data world generally, we'll bring the Apteco Marketing Suite into the picture and consider how this fits in.

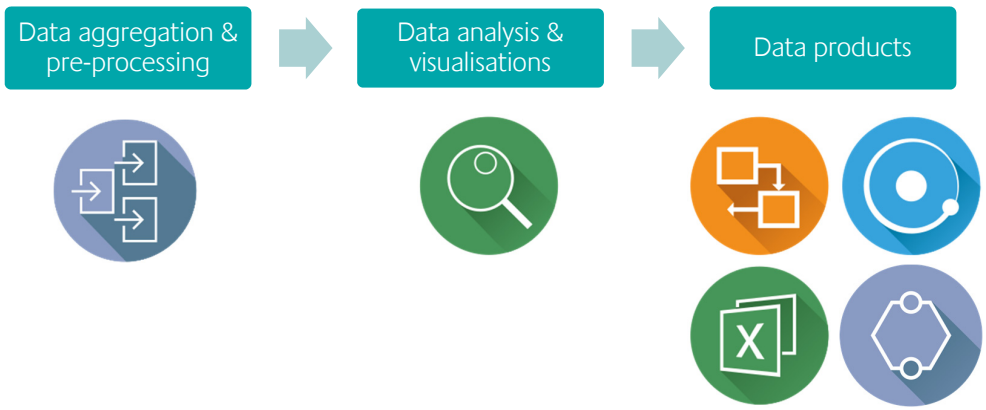
5.1.1 Example production flow revisited

Returning to our imagined production flow:

The different components of the Apteco Marketing Suite largely cover these different stages well:

- Data aggregation & pre-processing: *Designer* (and filtering and aggregation within *FastStats*)
- Data analysis & visualisations: *FastStats*
- Data products: *PeopleStage*, *Orbit*, *Excelsior*, *API*

Applying this to our diagram:



5.1.2 Current support for R in FastStats

There is already some integration with R in FastStats:

- The **RCall** and **RScript** functions in Expressions (under "Modelling Functions")

- **RCall** calls a function in R and enables you to pass a vector of numbers to this.
- **RScript** enables you to run a script in R (with some restrictions), including passing numbers to be substituted into variables

- Using R for modelling:
  - You can export data from FastStats into R to create a model, which is saved as a PMML file. This can be read back into FastStats to apply the model as an expression.
  - You can create RMarkdown files using this data (both the data fed to the model, and the model itself) to generate HTML pages with embedded R code.

5.2 Remaining uses for R or Python

Given that the Apteco Marketing Suite covers most parts of the data production flow quite well, the question to ask is: *What might people still want to use R or Python for?*

5.2.1 Both

- Command-line data analysis: while the drag-and-drop interface in FastStats is very user-friendly, there may be situations where doing analysis in the command line is easier or more efficient, and there may be people who are more accustomed to working in this way or prefer it.
- Web integrations: while Orbit is enabling increasingly sophisticated sharing of insights, the way to access data that allows maximum flexibility of usage (for example, in building web output) is through the API, using a programming language.

5.2.2 R

- Visualisations: making use of the visualisations packages available in R – there will always be more out there that isn't implemented in FastStats.
- Geographical applications: similarly, there is more functionality available in R packages than through Geo in FastStats.
- *Are there desired data analysis techniques we don't offer?*

5.2.3 Python

- Machine learning: an area where Python is strong, and something which people in industry are increasingly aware of and looking to make use of.
- *What do customers want to do with their data outside of FastStats?*
- *What do customers want to do with results from FastStats?*

5.3 Summary

Python is on the up in the data world, and doesn't seem to have peaked yet. It seems likely that Python will continue to grow in popularity against R, as more people with knowledge of programming, and specifically Python, enter the job market. But R is very well-established and will still be around for a while yet.

It seems highly likely that there are potential (or even current) customers out there, for whom a Python integration would be attractive; either as a bridge to tease them off their current system into using the Apteco Marketing Suite, or to enable even better integration of FastStats into their workflow.

Appendix A:Python integration outline

A.1 Purpose and form

The planned Python integration will aim to facilitate using data and results from FastStats in Python. It is currently possible to access certain types of results in FastStats via the API, but this requires some (not insignificant) data wrangling, especially for more complex things like cubes.

The Python integration will take the form of a Python package that provides a much more usable interface for accessing results from the API, including doing the leg-work of transforming the results into a more “Pythonic” form. This will enable users to focus on using the data and results for their desired purpose, instead of expending time on issues like format and structure.

It is hoped that a second version of the API will be developed at the same time, which the Python package will use. A corresponding library for R with matching functionality will also be developed.

A.2 Supported functions

This is subject to change, but a tentative list of the functions that the Python package will support is given in the following table:

Category	Function
Cube	getCube()
Selection	getSelectionCount()
	getSelectionRunDate()
	getSelectionResolveTable()
Expression	getExpressionText()
Segmentation	getSegmentationTimeReport()
	getSegmentationMigration()
	getSegmentationJourney()
	getSegmentationRetention()
Profile	getProfileSummary()
	getProfileValues()
	scoreRecords()
Data grid	getDataGrid()
API access	login()

Appendix B:Miscellaneous research

B.1 Customer research

Key points from the blog post[xvii] *"Python modules for data science"* by Stefan Seltmann at Apteco DE partner b.telligent:

- R is powerful for data analysis and modelling, but it's frustrating when you hit the memory limit while number crunching.
- There are Python-based alternatives to R, though as often is the case in an open source environment, it can be difficult at first to find your way in a new ecosystem of different tools.
- Motivation for developers and their backgrounds different from that of SPSSS/SAS users: clear orientation towards machine learning.
    - Clear technical orientation: can manipulate data at low-level.
    - High acceptance of black-box methods (for **scikit**): producing better performance justifies using an obscure method, even if this can't be demonstrated in terms of the interplay of the dependent variables.
    - Brute force gets to the goal: just throw all the computing power possible, e.g. with parallelisation, at the problem and set the CPUs glowing!
- Sometimes jumps straight to over-powerful more complex methods, when a simpler one would be more appropriate.
- Even if the initial impression can be confusing, there is huge potential in this technology.
- [Brief overview of the following packages: **Numpy**, **SciPy**, **pandas**, **scikit-learn**, **statsmodel**, **IPython**, **matplotlib**]
- Hype with machine learning in Python started parallel to the "data science" buzzword, but the technologies aren't brand new.
- These technologies have arrived at a resilient state of development for "professional" user (NumPy & pandas are used in the financial world) – user needs some experience to deal with not very conclusive error messages.
- Installation can be tricky, but there are distributions like **Anaconda** which come bundled with the above packages.

B.2 Tableau Python integration

Tableau has a Python integration, TabPy[xviii], which enables the user to run a Python script from within Tableau. This can be used, for example, to utilise machine learning packages available in Python and apply them to existing Tableau data. These results are returned within Tableau and so can then be used for further analysis.

B.3 Sources for Python and R comparison

The following tables list the articles used as sources for the Python and R comparison in Chapter 3. Each table includes the date the article was written, its title, a link to it, followed by key quotes.

May 15	DataCamp: Choosing R or Python for Data Analysis? An Infographic
https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis	
"My current strategy is to leverage the best of both worlds - do early data analysis in R, then switch to Python when it's time to get serious, be a team player, and ship some real code and data products." "I use R to conduct statistical tests, graph data, and inspect large data sets. If I actually have to write an algorithm, I prefer Python..." "I'd rather do math in a general-purpose language than try to do general-purpose programming in a math language."	

Apr 17	InfoWorld: Python vs. R: The battle for data scientist mind share
https://www.infoworld.com/article/3187550/data-science/python-vs-r-the-battle-for-data-scientist-mind-share.html	
"Why not make the best of both worlds, as many data scientists already do? The first stage of data aggregation can be accomplished with Python. Then the data is fed into R, which applies the well-tested, optimized statistical analysis routines built into the language."	

Oct 17	Betanews: Python vs R: Which programming language is better for data science?
https://betanews.com/2017/10/09/python-vs-r-data-science-programming-language/	
“So, should you use Python for data science? Python is a powerful and versatile tool that allows you to do more in less time. R, meanwhile, is a specialized tool, designed specifically for data analysis. In a market where diversifying is increasingly becoming key to development, adding Python to your repertoire, whether it’s your first language of choice or your second, can only be a good thing -- as one of the hottest tools in tech right now, not doing so could leave you in the dust. The great thing about Python is the fact that it’s versatile and easy to pick up, meaning that you can incorporate it into your own workflow, making it work alongside the tools you already use or might use later down the line -- yes, including R.” In the comments: • “For “data science” tasks Python might be better; for “statistics” problems I prefer Stata.” • “I use R-studio for python, best of both worlds. R is very powerful, but could they have picked a more horrible implementation. R language is tedious.”	

Sep 17	Analytics Vidhya: Python vs. R (vs. SAS) – which tool should I learn?
https://www.analyticsvidhya.com/blog/2017/09/sas-vs-vs-python-tool-learn/	
“Strategically, corporate setups that require more hands-on assistance and training choose SAS as an option.”	

“Researchers and statisticians choose R as an alternative because it helps in heavy calculations. As they say, R was meant to get the job done and not to ease your computer.” “Python has been the obvious choice for startups today due to its lightweight nature and growing community. It is the best choice for deep learning as well.”

Nov 13	Readwrite: Python Displacing R As The Programming Language For Data Science
http://readwrite.com/2013/11/25/python-displacing-r-as-the-programming-language-for-data-science/	
"As enterprises struggle to put data to work, they’re also struggling to find qualified data scientists. More often than not, however, such data scientists may already work for them and likely have some familiarity with Python." "Python provides employees with different backgrounds—notably engineers, mathematicians and analysts—a common, easy-to-understand language that can be used to prototype new functionality for the company."	

Sep 17	Quartz: If you want to upgrade your data analysis skills, which programming language should you learn?
https://qz.com/1063071/the-great-r-versus-python-for-data-science-debate/	
"In a nutshell, he says, Python is better for data manipulation and repeated tasks, while R is good for ad hoc analysis and exploring datasets."	

Jul 17	DataScience.com: R vs. Python: What language is Best for Building Data Models?
https://www.datascience.com/blog/r-vs-python-for-data-models-data-science	
"This extensive community has a distinct advantage because its members develop programs for the language that are a lot more diverse than those for R with respect to functionality. Most notably, Python’s suite of specialized deep learning and other machine learning libraries includes popular tools like scikit-learn, Keras, and TensorFlow, which enable data scientists to develop sophisticated data models that plug directly into a production system." "Python’s popularity is growing with the rise of deep learning and other machine learning techniques, but R will likely remain a favorite for complex statistical models."	

May 15	KDnuggets: R vs Python for Data Science: The Winner is ...
https://www.kdnuggets.com/2015/05/r-vs-python-data-science.html	
"R is mainly used when the data analysis task requires standalone computing or analysis on individual servers. It’s great for exploratory work, and it's handy for almost any type of data analysis because of the huge number of packages and readily usable tests that often provide you with the necessary tools to get up and running quickly. R can even be part of a big data solution." "You can use Python when your data analysis tasks need to be integrated with web apps or if statistics code needs to be incorporated into a production database. Being	

a fully fledged programming language, it's a great tool to implement algorithms for production use."

2017	Kaggle: The State of Data Science & Machine Learning 2017
https://www.kaggle.com/surveys/2017	
"Python was the most commonly used data analysis tool across employed data scientists overall, but more Statisticians are still loyal to R." "Everyone data scientist has an opinions on what language you should learn first. As it turns out, people who solely use Python or R feel like they made the right choice. But if you ask people that use both R and Python, they are twice as likely to recommend Python."	

May 14	FastCompany: Why The R Programming Language Is Good For Business
https://www.fastcompany.com/3030063/why-the-r-programming-language-is-good-for-business	
"I think the number one value to businesses [in using R] is access to talent," says Smith. "So many businesses now are doing much more with data, especially with the big data revolution and doing much more with analytics. And because they're hiring people coming out of school. They know R already."	

For customer service and technical support visit:

www.apteco.com/support

T: +44 (0)1926 407 595 (Support Desk)

Note: If you have purchased the Apteco Marketing Suite™ via one of our partners then they are your first line of support.

Apteco GmbH

Schaumainkai 87
60596 Frankfurt am Main
Germany
T: +49 (0) 69 25 66 97 0 – 0
support@apteco.de
www.apteco.de

Apteco Australia Pty Ltd

Level 2
99 Macquarie Street
Sydney NSW 2000
Australia
T: +61 (0) 2 8355 2524

www.apteco.com.au

Apteco Benelux

Stationsplein 45, Unit 4.004
3013 AK, Rotterdam
The Netherlands
T: +31 (0) 10 80 80 875
Email: support@apteco.nl
www.apteco.nl

Head Office

Apteco UK

Tink-a-Tank House
21 Jury Street
Warwick
CV34 4EH, UK

T: +44 (0) 1926 407565
E: support@apteco.com
W: www.apteco.com

Apteco

INSIGHT INTO ACTION

[i] Sources: Guido van Rossum's personal home page: https://gvanrossum.github.io/, Foreword for "Programming Python" (1st ed.), viewable at: https://www.python.org/doc/essays/foreword/
[ii] Discussion of this here: https://unspecified.wordpress.com/2011/10/18/why-pythons-whitespace-rule-is-right/
[iii] The text is available at: https://www.python.org/doc/essays/everybody/
[iv] https://www.python.org/dev/peps/pep-0020/ ('PEP' stands for Python Enhancement Proposal and is a design document providing information to the Python community or describing a new feature for Python)
[v] PEP 8 – style guide for Python code: https://www.python.org/dev/peps/pep-0008/
[vi] *Python for Data Analysis*, Wes McKinney (2nd edition), p3
[vii] https://docs.python.org/faq/general.html
[viii] https://wiki.python.org/moin/SimplePrograms
[ix] Since this chapter was a synthesis and summary of lots of different sources, these are listed as a collection, along with key quotes, in Appendix 5.3B.3
[x] Readwrite.com: *Python Displacing R As The Programming Language For Data Science:* http://readwrite.com/2013/11/25/python-displacing-r-as-the-programming-language-for-data-science/
[xi] FastCompany: *Why The R Programming Language Is Good For Business:* https://www.fastcompany.com/3030063/why-the-r-programming-language-is-good-for-business
[xii] KDnuggets: *Python overtakes R, becomes the leader in Data Science, Machine Learning platforms:* https://www.kdnuggets.com/2017/08/python-overtakes-r-leader-analytics-data-science.html
[xiii] [See xii]
[xiv] Kaggle: *The State of Data Science & Machine Learning 2017:* https://www.kaggle.com/surveys/2017
[xv] University of Miami – Center for Computational Science: *Should I learn Python or R or both?:* http://ccs.miami.edu/should-i-learn-python-or-r-or-both
[xvi] https://arrow.apache.org/docs/python/ipc.html#feather-format
[xvii] http://www.btelligent.com/en/blog/python-modules-for-data-science/
[xviii] Tableau blog post announcing the beta release, with a short overview: https://www.tableau.com/about/blog/2016/11/leverage-power-python-tableau-tabpy-62077