To R or to Python? Is this the question?

German Data Science Days, München
March 07, 2024
Content

- General Overview on R and Python
- Classical criteria
- Data Science criteria
- Educational aspects
- Conclusion/Open Questions
- programming language and environment
- initiated by Ross Ihaka and Robert Gentleman
- inspired by S programming language (“different implementation of S”) and Scheme
- GNU-project, multi-platform
- open-source environment widely used for statistical computing and graphics
- integrated suite of software facilities for data manipulation, calculation and graphical display
- “environment” = fully planned and coherent system
- for computationally intensive tasks, C, C++, and Fortran code can be linked and called at run-time
- R has its own LaTeX-like documentation format
- R distribution comes with 14 packages, more than 20k available from CRAN

https://www.r-project.org/about.html
- designed by Guido van Rossum
- high-level general purpose programming language
- multi-paradigm:
  - object-oriented
  - procedural (imperative)
  - functional
  - structured
  - reflective
- emphasising code readability
- multi-platform, open-source
- more than 500 k projects on Python Package Index

https://www.python.org/about/
https://pypi.org/
Popularity

- PYPL popularity index of programming languages
  - https://pypl.github.io/PYPL.html (raw data from Google Trends)
- TIOBE index
  - https://www.tiobe.com/tiobe-index/
- Stackoverflow
  - https://survey.stackoverflow.co/2023/#work-employment
# Syntax and Readability

## Python example
```python
def calculate_average(numbers):
    total = sum(numbers)
    count = len(numbers)
    average = total / count
    return average

data = [25, 30, 35, 40, 45]
result = calculate_average(data)
print("The average is:", result)
```

## R example
```r
# R example
calculate_average <- function(numbers) {
    total <- sum(numbers)
    count <- length(numbers)
    average <- total / count
    return(average)
}
data <- c(25, 30, 35, 40, 45)
result <- calculate_average(data)
print(paste("The average is:", result))
```

- Python uses indentation to define code blocks, while R uses curly braces `{}`.
- Python has a more straightforward and readable syntax, with an emphasis on code readability and simplicity.
- Python uses the `.` notation for method and attribute access, and the `[ ]` notation for indexing and slicing.
- Traditional R coders use the `<-` operator for assignment, while Python uses the `=` operator.
- R uses the `function_name <- function(arguments) { ... }` syntax for function definition, while Python uses `def function_name(arguments):`.
- R often uses the `print()` function to display output, while Python uses the `print` statement.
- R uses the `$` operator to access elements within a data frame, while Python uses the `.` notation for object attributes or the `[ ]` notation for dictionary and list elements.
Speed

- Membership testing on an unsorted vector of integers

https://towardsdatascience.com/r-vs-python-vs-julia-90456a2bcbab

```r
# 0.93 seconds
in_search <- function(vec, x) x %in% vec
```

```r
# 2.68 seconds
vec_search <- function(vec, x) any(x == vec)
```

```r
# 13.33 seconds
foreach_search <- function(vec, x) {
  for (v in vec)
    if (v == x)
      return (TRUE)
  FALSE
}
```

```r
# 21.94 seconds
for_search <- function(vec, x) {
  for (i in 1:length(vec))
    if (vec[i] == x)
      return(TRUE)
  FALSE
}
```

For evaluating different implementations in R, Python, and Julia, a dataset with 1.000.000 unique integers ranging from 1 to 2.000.000 was generated and 1.000 searches with all integers from 1 to 1.000 were performed. The probability of a search being successful is ~50%, so half the times the algorithm will scan the complete vector to conclude that the search was unsuccessful.

In the remaining cases, the algorithm should require \((n+1)/2\) evaluations (on average) to find the element, with \(n\) being the length of the vector.
# Python example using plotnine
from plotnine import *
from plotnine.data import mpg

# Create a facet plot using facet_wrap
    ggplot(mpg, aes(x='displ', y='hwy', colour = 'cyl')) +
    geom_point() +
    facet_wrap(~class, ncol=2)
Data Science Workflow with Python

Important Resources
- Anaconda Distribution: https://www.anaconda.com/download/
- Python Documentation: https://docs.python.org/
- Python Standard Library: https://docs.python.org/3/library

Join the Python for Data Science Automation Course
Data Science Workflow with R

Click the links for Documentation

Import
- read: import CSV, Excel, databases, SQL
- tidy: tidy data

Transform
- scale: standardize, normalize, log transform
- factor: categorical data
- transform: apply transformations

Model
- predict: model evaluation
- validate: model validation

Communicate
- R Markdown: create reports, slides, and documents
- Shiny: create interactive web applications

Important Resources
- R for Data Science: http://r4ds.had.co.nz/
- Big Book of R: https://www.r4ds.com/
- Rmarkdown: Book and CrowdBook
- Here are links: https://www.365datascience.com/docs/
- tidyverse package: https://tidyverse.org/
- Connecting to Databases: https://db.rstudio.com/
- R reproducible environments: https://reproducible.rstudio.com/

CS = Cheat Sheet
Data Science with Python – Special Topics

Time Series Forecasting
- ssktime - Sklearn extension for Time Series
- statsmodels - Time Series Analysis
- GluonTS - MXNet/Gluon Deep Learning for Time Series

Time Series Features
- TSFresh - Time Series Feature Engineering
- tslearn - Time Series Features
- Pandas Time Series
- Arrow - Human-Friendly Time

EDA
- pandas-profiling, SweetViz, lux

Web
- beautifulsoup - Extract data from HTML
- requests-html - HTML Parsing
- scrapy - Web crawling

MS Office & PDF
- XlsteWriter - Create Excel Workbooks
- pyexcel - Read/Write Excel
- xhwings - Call python from Excel
- python-docx - Word Documents
- python-pdfs - PowerPoint Documents
- edfreader - Text extraction from PDF
- textcat - Extract text from any document
- PyPDF2 - Create PDF documents
- pisa - Google Sheets

Text Analysis & NLP
- NLTK - Text Tokenization & Modeling
- spaCy - NLP using Cython for Speed
- fuzzywuzzy - Fuzzy String Matching

Recommendation Systems
- Annoy - Approximate Nearest Neighbors
- LightFM - Popular recommendation algo's.

Apps & APIs
- FastAPI - Web framework for building APIs in Python
- Flask - Web Development
- Dash & Streamlit - DS Web Frameworks

MLOps
- Pycaret MLFlow Integration
- MLFlow - Machine Learning Lifecycle, Tracking, Deployment
- MetaFlow - Scalable AWS Jobs for Data Scientists

Cloud
- boto3 (AWS) - AWS Python SDK
- Google Cloud - GCP Python SDK
- Azure - Azure Python SDK

ETL & Automations
- Airflow - Workflow Scheduling & Monitoring
- Luigi - batch job tool, scheduling, monitoring
- Ansible - Deployment Automation
- JobLib - Run python jobs

Machine Learning
- Scikit-Learn - ML in Python
- H2O - Scalable & AutoML
- TPOT - TPOT Automated ML Tool
- PyCaret - PyCaret Low Code ML
- Dask ML - Scalable ML with Dask
- ML Packages: XGBoost, LightGBM, CatBoost

Feature Engineering
- Sklearn Data Transformations
- sklearn-pandas - Sklearn Extension for Pandas
- featuretools - Automated Feature Engineering
- category_encoders - Categorical Encoding
- imbalanced-learn - Resampling for Imbalanced
- fancyimpute - Extended imputation strategies

Deep Learning
- TensorFlow & Keras
- PyTorch
- MXNet, Gluon, & GluonTS
- OpenAI Gym - Reinforcement Learning

Image & Comp Vision
- OpenCV - Open Source Computer Vision
- Scikit Image - Image Processing
- Pillow - Python imaging Library

Speed & Scale
- datatable - C++ Speed Up
- Dask (CS) - Parallel Pandas & Scikit Learn
- RAPID (CS) - GPU Accelerated Pandas
- PySpark - Spark Clusters
- Optuna - PySpark Extension for Humans

Coming from R?
- RtoPandas Conversion
- rdata & rdatafile - dplyr/tidyr ports
- datatable - data.table port
- dplyr - ggplot2 port
Data Science with R – Special Topics
Industry Adoption

Consider your specific data science specialization when choosing between R and Python:

- For general-purpose data science, machine learning, and AI applications, Python's versatility makes it a compelling choice.

- If your focus is primarily on traditional statistical analysis, hypothesis testing, and specialized graphical techniques, R offers a rich environment for these tasks.

- In academia and research, R remains a staple for statistical research and data analysis, while Python is increasingly gaining ground, especially in machine learning research.

- In industry, Python's widespread adoption and comprehensive ecosystem give it a competitive edge, especially in sectors requiring scalable data solutions and AI integration.

https://iabac.org/blog/r-vs-python-for-data-science-a-friendly-comparison

Educational Aspects

- Common Learning Goals:
  - Proficiency in Programming Languages
    - Students are expected to gain proficiency in programming languages commonly used in data science, such as Python, R, SQL, C/C++, and Java
    - Python is particularly emphasized due to its popularity, extensive library support, and ease of use for data science tasks
    - R is highlighted as an open-source language specifically designed for data science, focusing on statistical computing, machine learning, data manipulation, and visualization
    - C/C++ and Java are sometimes mentioned for their roles in high-performance applications, machine learning, statistical analysis, and data visualization
  - Application of Programming Languages
    - Students are expected to apply programming languages for tasks such as data manipulation, statistical analysis, machine learning, data visualization, and building machine learning models
    - Proficiency in using Python and R for machine-learning models and dealing with large datasets is emphasized.
  - Understanding of Data Science Tools and Libraries:
    - Mastery of data science tools and libraries associated with programming languages, such as Pandas, NumPy, Matplotlib in Python, and machine learning libraries like PyTorch and TensorFlow written in C/C++
  - Practical Machine Learning:
    - Acquisition of practical machine learning skills is a key component of the program.
  - Database Systems and Data Preparation:
  - Data Visualization:
    - Proficiency in data visualization using programming languages like Python and R is a crucial learning goal
  - Software Development Skills:
    - Build and automate data pipelines and analysis frameworks
Integrating R and Python

Distribution of the R and Python programming languages across Constructor University’s Master of Data Engineering Technologies program courses.
Integrating R and Python

Distribution of the R and Python programming languages across Constructor University’s Master of Data Science for Society and Business program courses.
Integrating R and Python

Distribution of the R and Python programming languages across the University of British Columbia’s Master of Data Science program courses.

https://ubc-mds.github.io/2020-02-03-teach-python-and-r/
Pedagogical challenges of teaching R and Python concurrently

• Mixed proficiencies of previous R & Python programming skills
  • in students
  • in instructors
• Newcomers have to learn both data science concepts and tools
• Dual (triple) task interference
• Memory decay during breaks in practice
• Standard environment for both or optimised environment for each
• Relevance of fundamental programming language specifics
• How to teach underlying paradigms through/in instead of/in addition to language specifics
Some considerations

- achieve proficiency in diverse programming languages
- diverse backgrounds and expectations
- data-centric vs. model-centric vs. output-centric
- role of software development
- role of automation
- causality
- underlying paradigms and philosophies
- further differentiation in jobs and roles
- qualification and learning standards
Conclusion

Python vs R:
- both languages are efficient for Data science
- provide rich tool kit for data analysis pipeline

- R
  - excels at data visualization
  - has a scientific orientation
  - is more focused on handling data in a statistical perspective
  - provides a large ecosystem for data science and communication

- Python
  - focuses on practical side of software implementation.
  - automation
  - deep learning
  - production or deployment