Medicon Medical Sciences
Volume 8 Issue 6 June 2025
Article Type: Research Article
ISSN: 2972-2721

SiPy - Bringing Python and R to the End-User in a Plugin-Extensible System

Nicholas TF Tan[1], Mathialagan Mugundhan[1], Tiantong Liu[2], Rick YH Tan[1], Alexander Y Tang[1], Bryan JH Sim[1], Jensen ZH Tan[1] and Maurice HT Ling[1,3]*

[1]*School of Applied Science, Temasek Polytechnic, Singapore*

[2]*Independent Researcher*

[3]*HOHY PTE LTD, Singapore*

***Corresponding Author:** Maurice HT Ling, School of Applied Science, Temasek Polytechnic, Singapore; HOHY PTE LTD, Singapore.

Received: May 12, 2025; **Published:** May 31, 2025

DOI: 10.55162/MCMS.08.295

Abstract

Data/statistical literacy is required for informed participation and better decision making in a literate society, and learning a data analysis tool may enhance the learning of statistical concepts. R and Python are well-known platforms for data analysis but are also difficult to learn even though they may be synergistic. Here, we present SiPy (Statistics in Python) as a data analysis tool built using Python and integrates analysis from R, and extensible using plug-ins. We will describe the architecture of the current version of SiPy, SiPy 0.6.0 codenamed "Otoro-Chutoro Continuum", with a listing of the 60 available analytical functions across 8 function classes. This is followed by an elaboration of SiPy plugin system where end-users can add functions as plugins, before ending with a description of SiPy's scripting system catering to modular scripting.

Keywords: Data analysis software; Statistical analysis software; Python; R; Plugin; Scripting

Introduction

Statistical education and literacy is deemed as an important part of education curriculum in all stages of life [1] - from high school [2] to undergraduates [3] to the population at large [4], as statistical literacy is an important component of literate societies [5]. Data and statistical literacy (which appears to be rather synonymous in modern day context [6-8]) is required for informed participation and better decision making [5, 9-11]. Moreover, statistics is an integral part of many sciences [12]. Despite its importance, many students persistently find statistical courses to be difficult [13-16]. Learning to use a suitable statistical software is important in statistical curriculum [17] but is also infamous for its steep learning curve [18, 19]. Yet, learning a data analysis or statistical tool may enhance the learning of statistical concepts [20, 21].

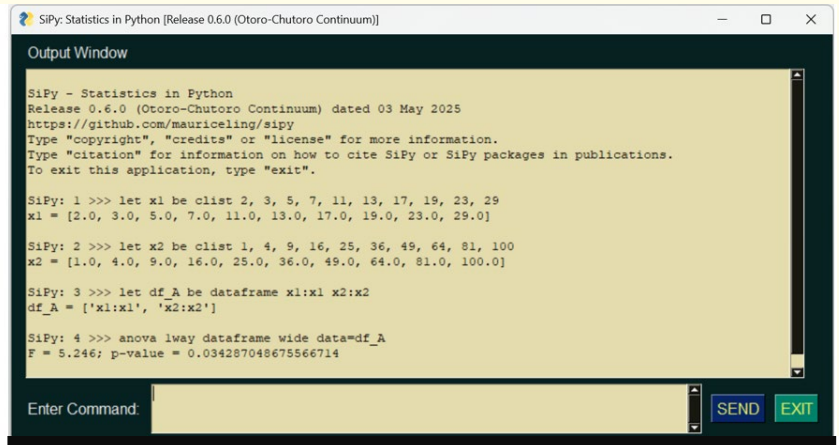
R (URL 1) is a very well-known open sourced and freely available statistical software in biosciences [22] but is also known to be difficult [23, 24]. Hence, tools such as Jamovi [25] (URL 2) aims to simplify the use of R through better graphical user interface. Python (URL 3), an open sourced and freely available general programming language, has also gained prominence in statistical and data analytic community [26]. This leads to comparison between Python and R for data analysis [27]. However, Python and R may be synergistic [28-30].

Building on a previous work [31]; in this article, we present SiPy - a data analysis tool built using Python and integrates analysis from R, and extensible using plug-ins. SiPy is the acronym for "Statistics with Python" as it is fundamentally built on top of well-known Python data analytics libraries, pandas [32] (URL 4) and scipy [33] (URL 5), via fitter (URL 6) and pingouin (URL 7) libraries. The current version of SiPy is version 0.6.0, codenamed "Otoro-Chutoro Continuum". In the following sections, we will describe the architecture of SiPy 0.6.0, with a listing of the available analytical functions; followed by how end-users can add functions into SiPy via its plugin system, before ending with a description of SiPy's scripting system catering to modular and stackable analyses.

Architecture of SiPy

SiPy 0.6.0 consists of both command-line user interface (CLI) and a basic graphical user interface linking to three major components (Figure 1): (a) libsipy which contains all of the analytical functions in SiPy, (b) SiPy Plugin Manager which manages and executes user- or community-implemented plugins, and (c) a scripting system which enables users to execute modular scripts. SiPy Plugin Manager (file = sipy_pm.py) is responsible for finding, loading, executing, and unloading of plugins; which are assumed to be in sipy_plugins folder (see section on Extension using Plugins for more information on plugins). Scripting system reads a script file of SiPy commands and executes each command sequentially (see section on SiPy Scripting System for more information on scripting).

(A)



(B)

```
SiPy - Statistics in Python
Release 0.6.0 (Otoro-Chutoro Continuum) dated 03 May 2025
https://github.com/mauriceling/sipy
Type "copyright", "credits" or "license" for more information.
Type "citation" for information on how to cite SiPy or SiPy packages in publications.
To exit this application, type "exit".

SiPy: 1 >>> let x1 be clist 2, 3, 5, 7, 11, 13, 17, 19, 23, 29
x1 = [2.0, 3.0, 5.0, 7.0, 11.0, 13.0, 17.0, 19.0, 23.0, 29.0]

SiPy: 2 >>> let x2 be clist 1, 4, 9, 16, 25, 36, 49, 64, 81, 100
x2 = [1.0, 4.0, 9.0, 16.0, 25.0, 36.0, 49.0, 64.0, 81.0, 100.0]

SiPy: 3 >>> let df_A be dataframe x1:x1 x2:x2
df_A = ['x1:x1', 'x2:x2']

SiPy: 4 >>> anova 1way dataframe wide data=df_A
F = 5.246; p-value = 0.034287048675566714

SiPy: 5 >>>
```

Figure 1: Basic Graphical User Interface (GUI) and Command-Line Interface (CLI) of SiPy 0.6.0. Panel A shows the basic GUI. Panel B shows the CLI.

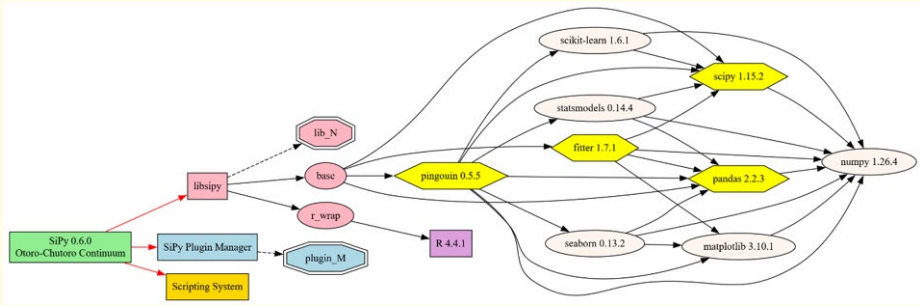


Figure 2: Architecture and Dependencies of SiPy 0.6.0.

LibSiPy (acronym for "library sipy" or "sipy library) is a collection of modules for data analytics. The purpose of bundling the analytical modules into a distinct library is for potential future library reuse [34, 35]. Hence, libsipy can have many modules but only two modules (base, and r_wrap) are currently implemented in SiPy 0.6.0. The base module contains Python-based statistical tests, and is based on four fundamental libraries; namely, (i) pandas [32] (URL 4) version 2.2.3, (ii) SciPy [33] (URL 5) version 1.15.2, (iii) fitter (URL 6) version 1.7.1, and (iv) pingouin [36] (URL 7) version 0.5.5. The other five supporting libraries are (i) numpy [37] (URL 8) version 1.26.4, (ii) seaborn [38] (URL 9) version 0.13.3, (iii) matplotlib [39] (URL 10) version 3.10.1, (iv) statsmodels [40] (URL 11) version 0.14.4, and (v) scikit-learn [41] (URL 12) version 1.6.1. The basic data structures used in SiPy 0.6.0 are panda's [32] data series and data frames which are based on numpy [37]. Pingouin [36] provides a high-level interface to commonly used statistical tests found in pandas [32], statsmodels [40], scikit-learn [41], and SciPy [33]; such as, t-tests and ANOVA; hence, forms the foundation of Python-based statistical tests in SiPy. When a high-level interface is not provided by Pingouin [36], the functions in SciPy [33] are used directly. Fitter, on the other hand, provides a method to fit probability distributions to data.

The r_wrap module acts as an interface between a portable version of R (R version 4.4.1), and follows a standardized operation for all R functions - Firstly, it writes out the passed pandas data frame into a comma-delimited file with a file name in the format of data_<random number>.csv. Secondly, a R script containing the required R codes and uses the comma-delimited data file is generated and saved as script_<random number>.R. Lastly, the R script is then executed using Rscript.exe found in the portable version of R and the output is captured and returned, after deleting both the comma-delimited data and R script file.

Analytical Functions in SiPy Version 0.6.0

SiPy 0.6.0 consists of 60 analytical functions across 8 class, as listed in Table 1. Of these 60 functions, the 22 types of regression are mainly from R, and using R packages; such as, MASS [42], and glmnet [43]. It is widely known that R (a statistical language) has a vast collection of statistical packages as compared to Python (a general programming language), and we formalized a method to bring R to Python.

Function Class	*List of Functions*
1. Descriptive Statistics	(i) Arithmetic Mean, (ii) Geometric Mean, (iii) Harmonic Mean, (iv) Kurtosis, (v) Standard Error, (vi) Skew, (vii) Standard Deviation, and (viii) Variance.
2. ANOVA	(i) 1-way ANOVA, and (ii) repeated measures ANOVA.
3. Effect Size	(i) Area Under the Curve, (ii) Common Language Effect Size, (iii) Cohen's D, (iv) Eta Square, (v) Hedges g, (vi) Odds Ratio, and (vii) Pearson's Correlation.
4. Correlation	(i) Biweight Midcorrelation, (ii) Distance Correlation, (iii) Kendall's tau-B Correlation, (iv) Pearson's Correlation, (v) Percentage Bend Correlation, (vi) Skipped Correlation, and (vii) Spearman's Correlation.
5. Normality Tests	(i) Kurtosis, (ii) Jarque-Bera, (iii) Shapiro-Wilk, and (iv) Skew.
6. Regression	(i) Complementary Log-Log Regression, (ii) Decision Tree Regression, (iii) Elastic Net Regression, (iv) Gamma Regression, (v) Gradient Boosting Regression, (vi) Hurdle Regression, (vii) Inverse Gaussian Regression, (viii) Lasso Regression, (ix) Linear Regression, (x) Logistic Regression, (xi) Multinomial Log-Linear Regression, (xii) Negative Binomial Regression, (xiii) Poisson Regression, (xiv) Proportional Odds Logistic Regression, (xv) Probit Regression, (xvi) Quasi-Binomial Regression, (xvii) Quasi-Poisson Regression, (xviii) Random Forest Regression, (xix) Support Vector Machine Regression, (xx) Support Vector Regression, (xxi) Tweedie Regression, and (xxii) Zero Inflated Count Regression.
7. t-tests	(i) 1-sample t-test, (ii) 2-samples / independent samples t-test assuming equal variance, (iii) 2-samples / independent samples t-test assuming unequal variance, (iv) Mann-Whitney U-Test (also known as Wilconox Rank-Sum Test), (v) Paired / dependent samples t-test, (vi) Two One-Sided Test, and (vii) Wilconox Signed-Rank Test.
8. Variance Tests	(i) Bartlett's Test, (ii) Fligner-Killeen's Test, and (iii) Levene's Test.

Table 1: List of Analytical Functions by Class.

Extension using Plugins

A SiPy plugin is a Python file that can be discovered, dynamically loaded, executed, and results returning back to SiPy. SiPy Plugin System is the engine that enables plugin discovery, loading, and execution; and is based on the framework proposed by Ritchey and Parker [44], which had been used in Ritchey and Perry [45]. Plugin discovery occurs at SiPy startup, and scans the sipy_plugins folder for any file with "py" extension but excluding base_plugin.py and sample_plugin.py.

The file, sample_plugin.py, provides the template codes for plugin development (see Code 1). Each plugin is a Python class, which inherits the BasePlugin class from sipy_plugins.base_plugin, and has four mandatory methods; namely, (i) initialization method, (ii) purpose method, (iii) usage method, and (iv) execute method. The initialization method of a plugin performs the initialization and requires 3 parameters: (i) the name of the plugin, (ii) version of the plugin, and (iii) author of the plugin. The purpose method takes no parameters and returns a short description to illustrate the purpose of the plugin to the user. The usage method is a longer version of purpose method, and is mainly used to illustrate the functions and usage of the plugin; hence, acts as a user manual. Finally, the execute method takes keyword dictionary (kwargs) as parameter, executes the operation according to the keyword arguments, and returns the results.

Six non-mandatory methods can also be defined in BasePlugin class and executed in the following order; namely, (i) initialize (which performs any initialization operations), (ii) setup (which is called by initialize method to perform any set up operations), (iii) pre_execute (which performs any execution after initialize method and prior to the calling of execute method), (iv) post_execute (which per-

forms any execution after the calling of execute method), (v) finalize (which performs any finalization operations after post_execute method), and (vi) cleanup (which is called by finalize method to perform any clean up operations). All of these six non-mandatory methods do not take any parameters.

Code 1. Sample Plugin

```
from sipy_plugins.base_plugin import BasePlugin
class SamplePlugin(BasePlugin):
    def __init__(self, name="Sample Plugin", version="1.0",
                 author="Maurice Ling"):
        super().__init__(name, version, author)
    def execute(self, kwargs):
        print("Executing SamplePlugin")
        return(f"Keyword Arguments: {kwargs}")
    def purpose(self):
        return """This is a sample plugin"""
    def usage(self):
        return self.purpose() + """
Usage: pg sample"""
```

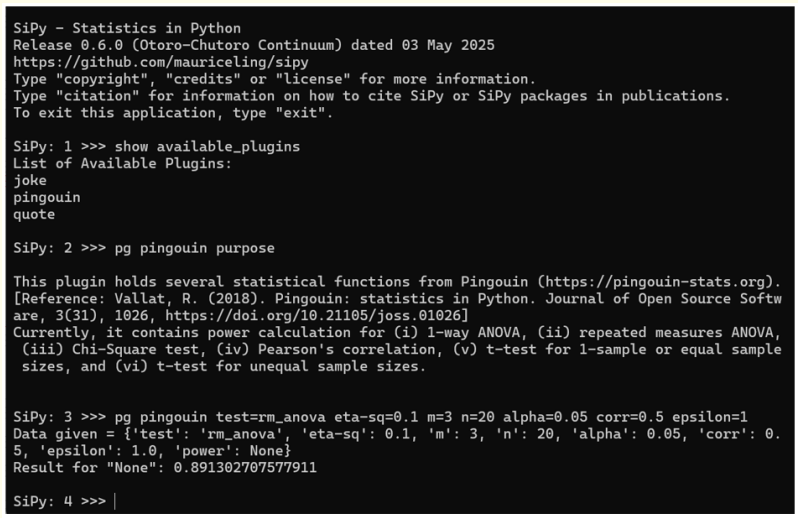


Figure 3: Showing Available Plugins, Followed by Power Calculation using Pingouin Plugin.

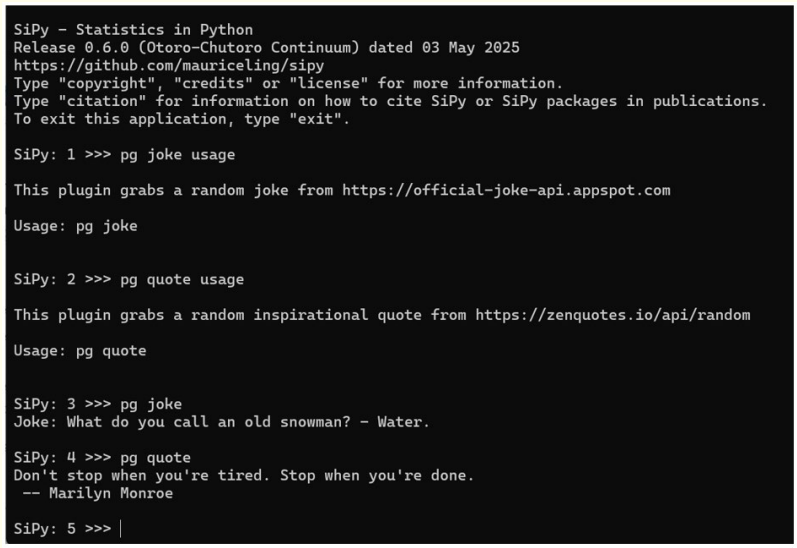


Figure 4: Example of a Joke and Quote using Plugins.

Three plugins (pingouin, joke, and quote) are implemented in SiPy 0.6.0 to demonstrate the versatility of SiPy plugin system. Pingouin plugin (Figure 3) uses pingouin [36] to perform power calculations, demonstrating that plugins can use any of the Python Standard Library (URL 15) or any third-party Python libraries installed. Joke and quote plugins use request library found within Python Standard Library to grab a joke or quote from internet; thus, demonstrating that SiPy plugins can have internet access.

SiPy Scripting System

A series of SiPy commands / instructions can be written in a text document and executed as a script. This can be useful for both running a series of commands that can be a long time (such as simulations or processing large datasets) [46], or for documenting analytical steps. Snippets of commands, representing an individual processing task, can also be reused and aggregated larger scripts [47]. SiPy caters to these needs by providing a means to execute a script file from operating system command line interface.

At a very basic level, a script file is nothing more than a sequential series of SiPy commands / instructions written into a text file. This forms a standalone script. However, it is common to have common data used for multiple analytical scripts. Hence, instead of needing to replicate the data declaration across multiple scripts, various analytical scripts can use the same data declaration. Therefore, it makes sense to have the data declaration in a file (assuming this file is data_values.sipy), and the analytical script (assuming this file is regression.sipy) performs regression analyses on the data in data_values.sipy. To enable this, regression.sipy file should include data_values.sipy. This is made possible using the @include operator, which will insert the script file into and in place of the @include statement.

Furthermore, @include statement can be chained. For example, there are 4 scripts (in example_scripts folder, URL 13): (i) script_01.py, which is standalone; (ii) script_02.py, which is standalone; (iii) script_03.py, which includes script_01.py; and (iv) script_04.py, which includes script_02.py and script_03.py. The end result is script_04.py including script_01.py, script_02.py, and script_03.py at the respective positions depending on @include statement as shown in Figure 5.

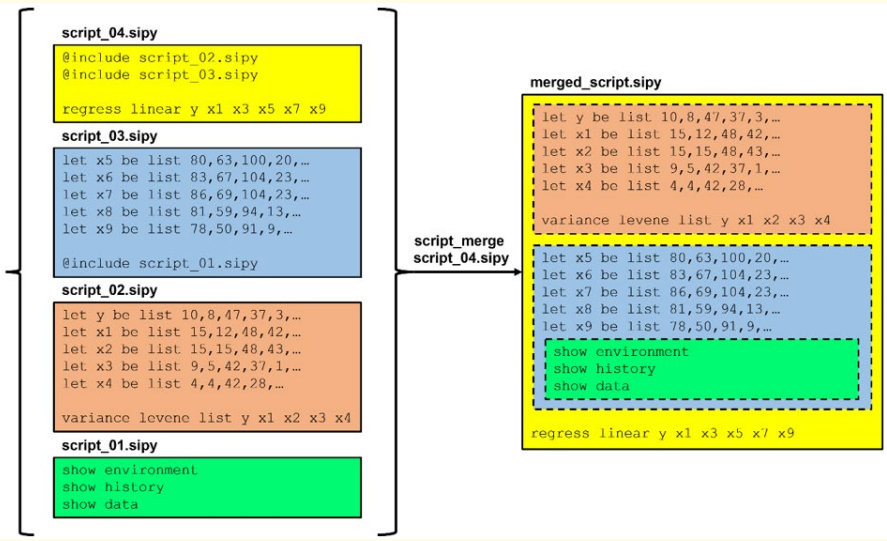


Figure 5: Merging of Scripts.

There are 2 main operations for scripts; namely, (i) script_merge to construct a monolithic standalone script from scripts chained using @include statements, and (ii) script_execute to execute the script. Both of these operations are performed at command line. For example, to merge the scripts with script_04.py as the entry point, the command at Anaconda prompt will be python sipy.py script_merge example_scripts\script_04.sipy and the merge script will be displayed. This can be saved into a file, merged_script.sipy, using redirection operator ">"; such as, python sipy.py script_merge example_scripts\script_04.sipy > merged_script.sipy. To execute script_04.py, the command at Anaconda prompt will be python sipy.py script_execute example_scripts\script_04.sipy.

Similarly, test scripts (in URL 14) are structured in the same format where there data declarations in data_values.sipy are used in all the tests scripts; such as, anova.sipy, regression.sipy, and ttest.sipy. Furthermore, an all-test.sipy file includes all the test scripts so that all tests can be executed by a command at Anaconda prompt python sipy.py script_execute test_scripts\all-test.sipy.

Therefore, modular analytical scripts can be made stackable using SiPy scripting system. This improves code sharing and reuse [48], which has the potential to improve reproducibility [49].

Future Work

Post version 0.6.0, future development on SiPy can extend in three directions. Firstly, SiPy needs a simple method of installation for end-users as SiPy 0.6.0 requires forking of the SiPy repository onto Anaconda distribution of Python with a suitable virtual environment. Secondly, the GUI of SiPy is a rudimentary GUI, much like the standard RGui in all R installations. Advancement in GUI can take reference from Jamovi [25]. Finally, more R and Python data analytics functions can be added to SiPy. On top of adding more R and Python data analytics functions, command-line analytical tools may be connected using a standard interface.

Conclusion

In this article. we present SiPy (Statistics in Python) version 0.6.0 (codenamed "Otoro-Chutoro Continuum") as a scripting and plugin-extendable data analysis tool built using Python and integrates analysis from R.

Supplementary Materials

Source codes SiPy can be found at https://github.com/mauriceling/sipy while documentation can be found at https://github.com/mauriceling/sipy/wiki. The release page for SiPy 0.6.0 (codenamed as Otoro-Chutoro Continuum) can be found at https://bit.ly/SiPy-0-6-0.

Acknowledgement

ML will like to acknowledge the time and resources given by Temasek Polytechnic for this project up to 31 August 2024; moreover, ML wishes to note that article represents his personal view and not that of Temasek Polytechnic.

List of URLs

URL 1 = https://www.r-project.org

URL 2 = https://www.jamovi.org

URL 3 = https://www.python.org

URL 4 = https://pandas.pydata.org/

URL 5 = https://scipy.org/

URL 6 = https://fitter.readthedocs.io/en/latest/

URL 7 = https://pingouin-stats.org/build/html/index.html

URL 8 = https://numpy.org/

URL 9 = https://seaborn.pydata.org/

URL 10 = https://matplotlib.org/

URL 11 = https://www.statsmodels.org/stable/index.html

URL 12 = https://scikit-learn.org/stable/

URL 13 = https://github.com/mauriceling/sipy/tree/main/example_scripts

URL 14 = https://github.com/mauriceling/sipy/tree/main/test_scripts

URL 15 = https://docs.python.org/3/library/index.html

Conflict of Interest

The authors declare no conflict of interest.

References

1. Watson J and Smith C. "Statistics Education at a Time of Global Disruption and Crises: A Growing Challenge for the Curriculum, Classroom and Beyond". Curriculum Perspectives 42.2 (2022): 171-179.
2. Kurnia AB, Lowrie T and Patahuddin SM. "The Development of High School Students' Statistical Literacy Across Grade Level". Mathematics Education Research Journal 36.S1 (2024): 7-35.

3. Woltenberg LN. "Cultivating Statistical Literacy Among Health Professions Students: a Curricular Model". Medical Science Educator 31.2 (2021): 417-422.
4. Gal I. "Adults' statistical literacy: Meanings, components, responsibilities". International statistical review 70.1 (2002): 1-25.
5. Hidayati NA, Waluya SB and Rochmad Wardono. "Statistics Literacy: What, Why and How?". Journal of Physics: Conference Series 1613.1 (2020): 012080.
6. Gould R. "Data Literacy is Statistical Literacy". Statistics Education Research Journal 16.1 (2017): 22-25.
7. Sabbati G. "Statistical and Data Literacy, A Practitioner's View for Policy-Making: How to Provide Independent, Objective and Authoritative Data and Information for Policy-Making". Statistical Journal of the IAOS 38.2 (2022): 463-469.
8. Umbach G. "Statistical and Data Literacy in Policy-Making". Statistical Journal of the IAOS 38.2 (2022): 445-452.
9. Radermacher WJ. "Literacy in Statistics for the Public Discourse". Statistical Journal of the IAOS 37.3 (2021): 747-752.
10. Witte V, Schwering A and Frischemeier D. "Strengthening Data Literacy in K-12 Education: A Scoping Review". Education Sciences 15.1 (2024): 25.
11. Yuniawatika Y. "Statistical Literacy and Its Urgency for Students". (Atlantis press) (2018): 170-173.
12. Kim KD, Chua SC and Ling MH. "Science/Education Portraits VII: Statistical Methods Used in 1081 Papers Published in Year 2020 Across 12 Life Science Journals Under BioMed Central". Acta Scientific Nutritional Health 4.3 (2021): 06-12.
13. Silva PN and Sarnecka BW. "What do Your Students Struggle with? A Survey of Statistics Instructors". Journal of Statistics and Data Science Education (2025).
14. Al-Qadri AH., et al. "The Prevalence of the Academic Learning Difficulties: An Observation Tool". Heliyon 7.10 (2021): e08164.
15. Peiró-Signes Á., et al. "Anxiety towards Statistics and Its Relationship with Students' Attitudes and Learning Approach". Behavioral Sciences (Basel, Switzerland) 11.3 (2021): 32.
16. Sutter CC, Givvin KB and Hulleman CS. "Concerns and Challenges in Introductory Statistics and Correlates with Motivation and Interest". The Journal of Experimental Education 92.4 (2024): 662-691.
17. Méndez-Romero RA., et al. "Rethinking the Teaching of University Statistics: Challenges and Opportunities Learned from the Colombia-UK Dialogue". Mathematics 11.1 (2022): 52.
18. Fernandes C and Ponte JPD. "Using Statistical Software in Basic Education: Difficulties and Affordances". (Unpublished) (2014): 35.
19. Ashour L. "A Review of User-Friendly Freely-Available Statistical Analysis Software for Medical Researchers and Biostatisticians". Research in Statistics 2.1 (2024): 2322630.
20. Zavez K and Harel O. "Teaching Statistical Concepts Using Computing Tools: A Review of the Literature". Journal of Statistics and Data Science Education (2025): 1-12.
21. Suhermi and Widjajanti DB. "What are the Roles of Technology in Improving Student Statistical Literacy?". Journal of Physics: Conference Series 1581.1 (2020): 012067.
22. Eglen SJ. "A Quick Guide to Teaching R Programming to Computational Biology Students". PLoS Computational Biology 5.8 (2009): e1000482.
23. Daswito R, Besral B and Ilmaskal R. "Analysis Using R Software: A Big Opportunity for Epidemiology and Public Health Data Analysis". Journal of Health Sciences and Epidemiology 1.1 (2023): 1-5.
24. Auker LA and Barthelmess EL. "Teaching R in the Undergraduate Ecology Classroom: Approaches, Lessons Learned, and Recommendations". Ecosphere 11.4 (2020): e03060.
25. ŞahiN M and Aybek E. "Jamovi: An Easy to Use Statistical Software for the Social Scientists". International Journal of Assessment Tools in Education 6.4 (2020): 670-692.
26. Ozgur C., et al. "MatLab vs. Python vs. R". Journal of Data Science 15.3 (2021): 355-372.
27. Hill C., et al. "Comparing Programming Languages for Data Analytics: Accuracy of Estimation in Python and R". WIREs Data Mining and Knowledge Discovery 14.3 (2024): e1531.

28. Deshmukh SS. "Data Analytics: Power of Integrating Python in R". Journal of Scientific and Engineering Research 10.5 (2023): 367-371.
29. Zhang H., et al. "easySCF: A Tool for Enhancing Interoperability Between R and Python for Efficient Single-Cell Data Analysis". Bioinformatics 40.12 (2024): btae710.
30. Varela LC. "Integrating R and Python into an Applied Econometrics Course". Proceedings of the IASE 2024 Roundtable Conference - Fostering Learning of Statistics and Data Science (International Association for Statistics Education, Toronto, Canada) (2024).
31. Chew JS and Ling MH. "TAPPS Release 1: Plugin-Extensible Platform for Technical Analysis and Applied Statistics". Advances in Computer Science: an International Journal 5.1 (2016): 132-141.
32. McKinney W. "Data Structures for Statistical Computing in Python". Proceedings of the 9th Python in Science Conference (Austin, Texas, USA) 445 (2010): 51-56.
33. Virtanen P., et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". Nature Methods 17.3 (2020): 261-272.
34. Fowler GS, Korn DG and Vo K-P. "Principles for Writing Reusable Libraries". ACM SIGSOFT Software Engineering Notes 20.SI (1995): 150-159.
35. Xu B., et al. "Why Reinventing the Wheels? An Empirical Study on Library Reuse and Re-Implementation". Empirical Software Engineering 25 (2020): 755-789.
36. Vallat R. "Pingouin: Statistics in Python". Journal of Open Source Software 3.31 (2018): 1026.
37. Harris CR., et al. "Array Programming with NumPy". Nature 585.7825 (2020): 357-362.
38. Waskom M. "Seaborn: Statistical Data Visualization". Journal of Open Source Software 6.60 (2021): 3021.
39. Hunter JD. "Matplotlib: A 2D Graphics Environment". Computing in Science & Engineering 9.3 (2007): 90-95.
40. Seabold S and Perktold J. "Statsmodels: Econometric and Statistical Modeling with Python". Proceedings of the 9th Python in Science Conference (2010).
41. Pedregosa F., et al. "Scikit-learn: Machine learning in Python". Journal of Machine Learning Research 12 (2011): 2825-2830.
42. Venables WN and Ripley BD. Modern Applied Statistics with S (Springer, New York), Fourth (2002).
43. Friedman J, Tibshirani R and Hastie T. "Regularization Paths for Generalized Linear Models via Coordinate Descent". Journal of Statistical Software 33.1 (2010): 1-22.
44. Ritchey RP and Parker TW. "Simple Plugin Methodology in Python". (Computational and Information Sciences Directorate, US Army Research Laboratory, USA, Adelphi, MD, USA), ARL-CR-0743 (2014).
45. Ritchey RP and Perry R. "Machine Learning Toolkit for System Log File Reduction and Detection of Malicious Behavior". IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (IEEE, Vancouver, BC, Canada) (2021).
46. Cingolani P, Sladek R and Blanchette M. "BigDataScript: A Scripting Language for Data Pipelines". Bioinformatics 31.1 (2015): 10-16.
47. Mooers BHM and Brown ME. "Templates for Writing PyMOL Scripts". Protein Science 30.1 (2021): 262-269.
48. Cadwallader L., et al. "Advancing Code Sharing in the Computational Biology Community". PLoS computational biology 18.6 (2022): e1010193.
49. Cadwallader L and Hrynaszkiewicz I. "A Survey of Researchers' Code Sharing and Code Reuse Practices, and Assessment of Interactive Notebook Prototypes". Peer J 10 (2022): e13933.

Volume 8 Issue 6 June 2025