Espirales
Revista multidisciplinaria de investigación científica

Analysis of the use of the Python programming language for statistical calculations.

Análisis del uso del lenguaje de programación Python para cálculos estadísticos

Santiago Israel Logroño Naranjo*
Néstor Augusto Estrada Brito*
Vanessa Alexandra Vásconez Núñez*
Evelin Marisol Rosero Ordóñez*

Received: July 13, 2021
Approved: September11 , 2021

Cite this:

Logroño, S., Estrada, N., Vásconez, V., Rosero, E. (2022). Analysis of the use of the Python programming language for statistical calculations. *Espirales. Revista Multidisciplinaria de investigación científica*, *6*(41), 1-13

Abstract

This article presents a review of the use of Python for data analysis. It discusses the main specialized libraries and the advantages of using Python for data analysis. Also, the main phases of data analysis are explained in general and in an example with the programming language. It is concluded that Python presents multiple advantages in its use and that in spite of being an open source software, it allows professional and sophisticated work in data management and statistical calculations.

Keyword: Python, Programming Language, Statistics, Data.

* Master's Politécnica de Chimborazo (ESPOCH), Riobamba, Ecuador. israel.logronio@espoch.edu.ec https://orcid.org/0000-0002-1205-3017
* Master's Degree in Communication Technologies, Systems and Networks, Escuela Superior Politécnica de Chimborazo (ESPOCH), Riobamba, Ecuador. nestor.estrada@espoch.edu.ec https://orcid.org/0000-0002-4100-7351
* Master's Degree in Computer and Network Engineering, Universidad Nacional de Chimborazo, Riobamba, Ecuador. vavasconez@unach.edu.ec, https://orcid.org/0000-0002-6336-5598
* Master's Degree in Management Information Systems, Escuela Superior Politécnica de Chimborazo (ESPOCH), Riobamba, Ecuador. marisol.rosero@espoch.edu.ec https://orcid.org/0000-0001-9024-7725

Resumen

Este artículo presenta una revisión del uso de Python para el análisis de datos. Se analizan las principales librerías especializadas y cuáles son las ventajas del uso de Python en el tema. Asimismo, se explican las fases principales del análisis de datos de manera general y en un ejemplo con el lenguaje de programación. Se concluye que Python presenta múltiples ventajas en su uso y que a pesar de ser un software de código abierto permite el trabajo profesional y sofisticado en el manejo de datos y cálculos estadísticos.

Palabras clave: Python, lenguaje de programación, Estadistica, Data.

Introduction

Currently, statistical graphs are the most widely used method to represent data analysis. The visual display of such data allows more information to be provided in a clear manner. There are many programs and tools used for statistical analysis that have evolved from mathematical routines and subprograms to high-level professional programs and tools.

In the same context, the quality of the data corresponds to the quality of the tool and the expertise of the analyst. One such language is Python, which, in addition to being free, allows advanced data processing and is compatible with sophisticated models.

Python is one of the most widely used programming languages for software development. Its language construction, as well as its object-oriented approach, are intended to help programmers write clear and logical code for small and large-scale projects. In this sense, Python is a language that can be used to develop software for scientific applications, statistics, networking, desktop applications, games, web applications, among others. (Hudson Pérez et al., 2013, p. 112)

With this background, and before analyzing the important aspects of the Python programming language and its use for statistical calculations and graphical representation of data, a brief theoretical framework is presented.

According to Giroux, (2021), there are several Python libraries focused on the topic that can be used in case the datasets are minimal (not too large) or if you do not rely on imports from other libraries. Some of the most widely used and popular packages are: NumPy, Pandas, Seaborn and Matplotlib.

NumPy: Refers to a library used to work with matrices, it is effective for use with both unidimensional and multidimensional arrays. This library consists of several routines used for statistical analysis and also to store data in Nd-arrays for storage. (Pérez-García et al., 2021, p. 224)

Matplotlib: It is a library for data visualization. The union of all SciPy, NumPy and Panda with Matplotlib contributes with a good performance. Matplotlib is a graph plotting data package for Python along with its NumPy numerical extension that works on all platforms. (Bødker et al., 2022)

Pandas: These are types of libraries used for numerical computation based on NumPy. Pandas works well in handling named data of one type with sets of objects and two types of data with Data Frame objects. Through the conversion of data into a tabular form, pandas allow data to be easily read and more structured. (Takefuji, 2021, p.102)

Seaborn: is a Python data visualization package based on matplotlib that is mainly connected to Pandas data structures. Visualization is the main component of Seaborn that aids in data understanding and exploration. (Silvestri et al., 2022)

Sklearn/Scikit-learn: are the most important libraries for machine learning in Python. It includes many crucial tools for clustering, classification, dimension reduction and regression.

SciPy: is a library used for scientific calculations that is based on NumPy. SciPy provides one more added functionality than NumPy, which includes scipy stats for statistical analysis. (Silvestri et al., 2022)

Thus, data visualization is an important point in data analysis, because it ensures that it is more interactive and understandable by displaying or plotting the data in graphical form. It is worth noting that Pandas is an open source Python package that handles three types of data structures: data frames, panels and series that solves the need to visualize and analyze data. Boelens & Tchelepi, (2021). Similarly, statistical data analysis when using Python facilitates the task as its programming language has many advantages over other programming languages. It has important features such as being a high-level language, easy to understand and can also be used by any user or programmer (Sahoo et al., 2009).

Materials and methods

Descriptive statistics involves describing and summarizing data. There are two main approaches it uses:

Visual approach to illustrate data with histograms, tables, charts, diagrams and many other graphs.

The quantitative approach gives descriptions and summaries of the data numerically.

According to Zolotov et al., (2021)descriptive statistics is applied to one or several variables or data sets. When a variable is summarized and described, univariate analysis is performed. When searching for statistical relationships between pairs of variables,

bivariate analysis is performed, finally, multivariate analysis is joined with a different variable at the same time.

The types of measures used in descriptive statistics: Central tendency shows the centers of data. Measures that are important such as median, mean, mode. Correlation illustrates the relationship between pairs of variables in the data set. Important measures are: the covariance coefficient and also the correlation. Variability illustrates the dispersion of the data. Useful measures are: standard deviation and variance. (Schwarz et al., 2021)

In statistics, the population is a set of all items or elements in which one is interested. They are usually large, which makes them inefficient and unsuitable for analysis and data collection. By selecting and examining a subset that represents the sample population, it helps to draw some conclusions about a given population. The subset of the population is called a sample, which should always maintain the main population characteristics of the statistic to be satisfied, which will allow the use of the sample to draw conclusions about a given population. Zurheide et al., (2021). The null hypotheses, according to which the intercepts and gradients of the population are individually equal to 0 are tested by using test statistics that are provided by the coefficient estimate that is divided by the error of the norm. The comparison of test statistics is performed with the distribution of n - 2 sample size - regression coefficients and number of degrees of freedom. The 95% confidence interval for each individual coefficient in the population is normally evaluated as follows coefficient ± (tn-2 × the standard error), with the tn-2 points being 5% for a t distributed with n - 2 degrees of freedom. (Iweka et al., 2021)

The P value that is for the urea coefficient (0.005) provides great results on the null hypothesis, indicating that the population coefficient is not 0 because there is relationship between age and urea. The urea coefficient is the regression gradient line and its hypothesis test is equal to the correlation coefficient of the population test as indicated above. The constant P-value 0.054 gives inadequate results to show that the population coefficient is not equal to 0. Although the intercept is not substantial, it is important to ensure that it remains in the equation. The present conditions where a straight line passes between the origin are seen as appropriate data, which, in this situation, an important regression analysis could be done to ensure that it eliminates the present constant (de Rosa & Papa, 2021, p. 112)

For Gunaratne & Garibay, (2021) data are the most significant unit in any study. The supply of data is used as inputs in the analysis depending on the requirements of the analysis. The term known as experimental unit is the form of organization used in data collection (such as, for example, the population of people or an individual). It was possible to obtain and identify certain population variables (e.g., height, age, and wage weight). Regardless of whether the data were categorical or numerical. Organization or processing of the data collected for analysis must be performed. For example, organizing the data into columns and rows in the data structured in table format for further analysis, typically through the use of statistical software or spreadsheet

For (Matsuda et al., 2021) data cleaning is performed after data processing and organization. It involves searching for duplicates, errors, and inconsistencies in the data, and subsequently eliminating them. The data cleaning process includes tasks such as sorting data, matching records, identifying outliers, identifying inaccurate data, maintaining data quality, and spell-checking textual data. As a result, it helps to avoid unexpected results in delivering high quality data, which is important for a robust and successful outcome.

Once the data sets are cleaned and free of errors, the analysis can proceed. Then, different modes of techniques are applied such as, preliminary analysis of the data by understanding the messages contained in the data obtained, and expressive statistics - finding the mode, median, mean and others for Meng et al., (2021) data visualization is one of the techniques used, in which the data are illustrated in a graphical format to obtain additional observation regarding the information in the data.

Mathematical models or formulas (called algorithms), were added to the data in order to discover relationships within variables; such as, the use of causality/correlation.

Results

Python is easy to master and learn. Most people can learn it, even those with less programming knowledge. By using a popular language, there are many possibilities to find a solution to problems that may arise. Watcharasupat et al., (2022) Writing code in Python is easy, which enhances development. Also, Python can be accessed by design, making it one of the fastest languages in terms of development speed. In addition, reading Python code is intuitive, making it easy to maintain. Python's syntax is concise and clear. The layout of the language is fairly close to English and readable, making it easy to interpret.

Fewer lines of code are needed for Python to get results compared to other languages such as Java or C. This simplicity of Python helps a lot when reading written code or another developer's code. Code review is much faster and easier when there is less line code to review, and the reading is more like English. There is less updating that is involved each time code changes hands, the calculation is done quickly as far as each function is concerned. With code that makes it easy to understand and navigate, the user may be able to minimize the amount of work it takes to extend and maintain their code base. Likewise, Python provides tried and tested versatility. Likewise, Python is utilized by some assertive projects around the web like, Reddit, EVE Online, YouTube to reliably serve the base to their users.

In terms of statistical analysis in Python, the path explained above is followed, but through a practical example presented below: data collection in Python was done through the creation of behavioral experiments or electronic surveys with flexibility in how they presented audio/visual stimuli (e.g., text, shapes, sounds, images, movies, animations), recording simple timing measurements (e.g., stimulus durations or onsets),

and collecting behavioral responses (e.g., reaction times, initiation of button presses). PsychoPy is a Python package that allowed researchers to perform a wide range of psychology or neuroscience experiments. Aspects of their experiments are customized using PsychoPy's graphical user interface. Whenever they chose to use Python code to write behavioral experiments, they were advised to start by identifying online tutorials and also edit other people's code for their experimental template design (Terven & Córdova-Esparza, 2021, p. 108)

Python had important tools that helped to organize and clean data, helped to iteratively move, create and copy folders. The Os module was useful for working with the fundamental operating computer system (e.g., Linux, Windows, Mac), which was important when using large amounts of data that were not saved in the Excel spreadsheet. The Pandas package was an intuitive and extended tool that allowed researchers to incorporate all types of data as Excel-style comma-separated spreadsheets. Data could be saved in Pandas and Data Frames, which made it easy to perform operations on all labeled column and row data, such as reshaping and merging data sets or scoring questionnaire data.

Data analysis: Python performed large types of statistical calculations for data analysis. When using Pandas, quick pairwise Pearson correlations were performed between data over columns in case observations were put in rows. There were more statistical packages, which included Pymer4, statsmodels, scipy.stats module in SciPy.

Data visualization: After applying it, the matplotlib.pyplot module in the Matplotlib package it was possible to create graph types. Seaborn was an important and consistent package that was based on Matplotlib that created great statistical plots.

Histograms are especially important when there are a large number of significant values in the data set. They divide the values in the data set that were ordered into bins. Most often the bins have the same width, although this is not the case. The values in the upper and lower intervals are called edges of the cube. The data representation of a bar chart groups the different results in columns along the x-axis. The y-axis is used to illustrate data distributions by representing the percentage of occurrence or numerical count in each individual column. The matplotlib.pyplot.hist()- is used to draw a histogram in Python.

The frequency of the individual values corresponds in each bin. These are the numbers of elements in the data set with bin edge values. The bins include the values equal to the lower edges, but exclude the values equal to the edges that were upper. The bin on the right is closed because it includes all the edges. When the data set is split with bin edges 0, 5, 10, and 15, all three bins were presented:

> The leftmost bin contained values greater than or equal to 0 and also less than 5. The second tray below has values greater than or equal to 5 and also less than 10.

The bin number three on the right has values greater than or equal to 10 but less than or equal to 15.

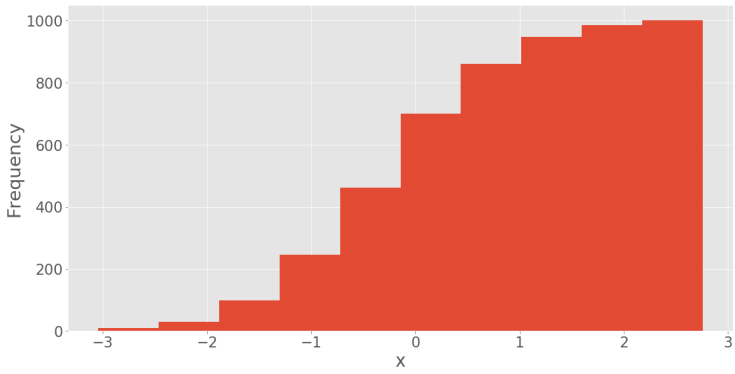
The np.histogram() function was the appropriate way to obtain data for histograms:
It took the array that had the data and the number of bins and NumPy returns two arrays:

The frequency in the histogram corresponds to each individual bin.
The edges of the bin contain the edges of the bin.
What was calculated by the histogram (), was shown graphically by hist()

```
fig, ax = plt.subplots()
ax.hist(x, bin_edges, cumulative=False)
ax.set_xlabel('x')
ax.set_ylabel('Frequency')
plt.show()
```

These code performances are in the sample below:

Figure 1. Code performance



The histogram above shows the cumulative values. The frequency of the first tray on the left is the number of items in the tray. The frequency of the second tray is the sum total of the number of items in the second and first tray. The next boxes followed a similar pattern. Finally, the last frequency box on the right was the total number of items available in the data set, which in this case was 1000.

Graphs represent data labels of fewer numbers that also give frequencies that are relative. They work with unordered labels like data that are nominal. Each slice corresponds to a single label data set and has an area equal to the frequency that is relative and connected to that same label.

The data associated with three labels were defined as follows:

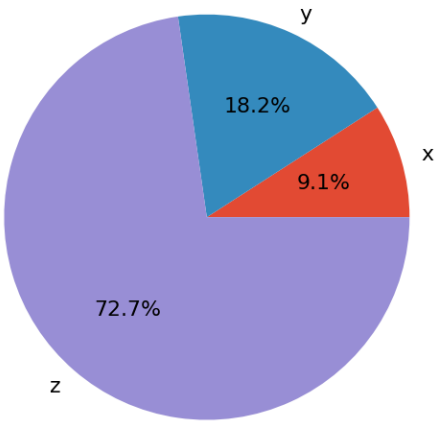
```
>>> x, y, z = 312, 543, 236
```

The pie chart was created with foot ():

```
fig, ax = plt.subplots()
ax.pie((x, y, z), labels=('x', 'y', 'z'), autopct='%1.1f%%')
plt.show()
```

The first argument of foot () is the data, and the second is the sequence of matching labels. The auto pct defines the format of the relative frequencies in the figure below.

Figure 2. Relative size value



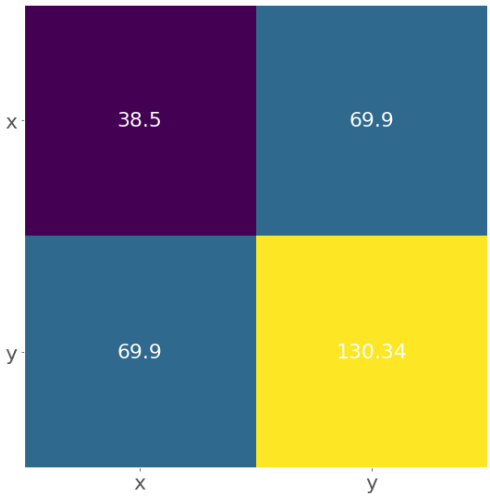
The graph shows the smaller part as X, y is the mean, and z is the larger part of the graph. The percentages show each value in relative size compared to its total sum.

It was used to illustrate the visual matrix and the colors representing the matrix numbers. In addition, heat maps were very important to illustrate the correlation and covariance matrices. An. Imshow() was used to create a heat map for a covariance matrix below:

```
(Gehlenborg & Wong, 2012).
matrix = np. covariance (x, y). rounding(decimals=2)
fig, ax = plt.subplots()
ax.smshow (matrix)
ax.grid (False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('x', 'y'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('x', 'y'))
ax.set_ylim(1.6, -0.6)
for t in range(2):
for the s in range(2):
```

The heat map below consists of the x and y labels, and also the covariance matrix numbers.

Figure 3. Heat map



The field in yellow represents the largest element of the 130.34 matrices; the purple field corresponded to 38.5 smaller elements, while the blue squares between purple and yellow were associated with the value of 69.9.

Conclusions

A discussion of the steps for data analysis with Python, such as data collection, data cleaning, data analysis, and exploratory data analysis is presented. The Python programming language was used for the main implementation. By using different visualization and analysis methods, numerous results were obtained. Explanations of how one variable affects other variables is shown in the analysis and different graphs, pie charts and heat maps have been plotted with the use of many attributes in the data set and conclusions of the data sets were made in a simple manner. Quantities that summarize and describe the datasets and how their calculations are performed in Python have been identified.

It is also concluded that Python as a programming language and its application in the presentation of statistical calculations and graphs provides numerous libraries for this purpose, is versatile, of high quality and its learning curve is easy and fast and therefore, it is a useful and effective tool.

References

Bødker, M. S., Wilkinson, C. J., Mauro, J. C., & Smedskjaer, M. M. (2022). StatMechGlass: Python based software for composition-structure prediction in oxide glasses using statistical mechanics. *SoftwareX*, *17*, 100-913. https://doi.org/10.1016/j.softx.2021.100913.

Boelens, A. M. P., & Tchelepi, H. A. (2021). QuantImPy: Minkowski functionals and functions with Python. *SoftwareX*, *16*, 100-109. https://doi.org/10.1016/j.softx.2021.100823.

de Rosa, G. H., & Papa, J. P. (2021). OPFython: A Python implementation for Optimum-Path Forest[Formula presented]. *Software Impacts*,*9* (July), 100-113. https://doi.org/10.1016/j.simpa.2021.100113

Giroux, B. (2021). ttcrpy: A Python package for traveltime computation and raytracing. *SoftwareX*, *16*, 100-134. https://doi.org/10.1016/j.softx.2021.100834.

Gunaratne, C., & Garibay, I. (2021). NL4Py: Agent-based modeling in Python with parallelizable NetLogo workspaces. *SoftwareX*,*16* , 100801. https://doi.org/10.1016/j.softx.2021.100801. https://doi.org/10.1016/j.softx.2021.100801

Hudson Pérez, M. C., Förster Marín, C. E., Rojas-Barahona, C. A., Valenzuela Hasenohr, M. F., Valdés, P. R., & Ferré,

A. R. (2013). Comparison of the effectiveness of two strategies teaching methodology in the development of the reading comprehension in the first school year. *Perfiles Educativos*,*35* (140), 100-118. https://doi.org/10.1016/s0185-2698(13)71824-5.

Iweka, S. C., Owuama, K. C., Chukwuneke, J. L., & Falowo, O. A. (2021). Optimization of biogas yield from anaerobic co-digestion of corn-chaff and cow dung digestate: RSM and python approach. *Heliyon*,*7* (11), 255-260. https://doi.org/10.1016/j.heliyon.2021.e08255.

Matsuda, F., Maeda, K., Taniguchi, T., Kondo, Y., Yatabe, F., Okahashi, N., & Shimizu, H. (2021). mfapy: An open-source Python package for 13C-based metabolic flux analysis. *Metabolic Engineering Communications*, *13*(July), 177-188. https://doi.org/10.1016/j.mec.2021.e00177.

Meng, S., Fu, Z., Qin, J., Ma, X., Li, Y., Hao, L., Liu, Y., Sun, K., & Chen, D. (2021). magcoilcalc: A Python package for modeling and optimization of axisymmetric magnet coils generating uniform magnetic field for noble gas spin-polarizers. *SoftwareX*, *16*, 100-105. https://doi.org/10.1016/j.softx.2021.100805.

Pérez-García, F., Sparks, R., & Ourselin, S. (2021). TorchIO: A Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. *Computer Methods and*

Programs in Biomedicine,*208* , 200-230. https://doi.org/10.1016/j.cmpb.2021.106236. https://doi.org/10.1016/j.cmpb.2021.106236.

Schwarz, S., Uerlich, S. A., & Monti, A. (2021). pycity_scheduling-A Python framework for the development and assessment of optimisation-based power scheduling algorithms for multi-energy systems in city districts. *SoftwareX*, *16*, 100-104. https://doi.org/10.1016/j.softx.2021.100839.

Silvestri, L. G., Stanek, L. J., Dharuman, G., Choi, Y., & Murillo, M. S. (2022). Sarkas: A fast pure-python molecular dynamics suite for plasma physics. *Computer Physics Communications*, *272*, 1100-1140. https://doi.org/10.1016/j.cpc.2021.108245.

Takefuji, Y. (2021). SCORECOVID: A Python Package Index for scoring the individual policies against COVID-19. *Healthcare Analytics*,*1* (August), 100-105. https://doi.org/10.1016/j.health.2021.100005

Terven, J. R., & Córdova-Esparza, D. M. (2021). KinZ an Azure Kinect toolkit for Python and Matlab. *Science of Computer Programming*,*211* , 102-109. https://doi.org/10.1016/j.scico.2021.102702. https://doi.org/10.1016/j.scico.2021.102702

Watcharasupat, K. N., Lee, J., & Lerch, A. (2022). Latte: Cross-framework Python

package for evaluation of latent-based generative models. *Software Impacts*, 100-222. https://doi.org/10.1016/j.simpa.2022.100222.

Zolotov, O., Romanovskaya, Y., & Knyazeva, M. (2021). pyFIRI - A free and open source Python software package of the non-auroral Earth's lower ionosphere. *SoftwareX*, *16*, 100-105. https://doi.org/10.1016/j.softx.2021.100885. https://doi.org/10.1016/j.softx.2021.100885

Zurheide, F. T., Hermann, E., & Lampesberger, H. (2021). PyBNBowTie: Python library for Bow-Tie Analysis based on Bayesian Networks. *Procedia Computer Science*,*180* (2019), 344-351. https://doi.org/10.1016/j.procs.2021.01.172.