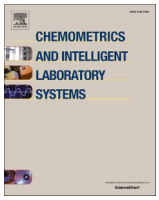
Chemometrics and Intelligent Laboratory Systems 241 (2023) 104941

Contents lists available at ScienceDirect

Chemometrics and Intelligent Laboratory Systems

journal homepage: www.elsevier.com/locate/chemometrics



An updated tutorial on reproducible PyPI applications for advancing chemometrics and boosting learner motivation

Yoshiyasu Takefuji

Faculty of Data Science, Musashino University, 3-3-3 Ariake Koto-ku, Tokyo 135-8181, Japan

ARTICLE INFO

Keywords:
Python package index (PyPI)
Software reproducibility
Open-source
Multiple operating systems

ABSTRACT

Context: Programming, particularly with the widely-used scripting language Python, is a powerful tool for expressing ideas in science and technology. PyPI, or the Python Package Index, is a management tool for Python packages. Despite the availability of many tutorials on PyPI, including on its official website, many users have encountered difficulties due to outdated information and issues with the twine upload library.
Objectives: This paper demonstrates the use of a gas chromatography program to prototype the PyPI application, with the goal of maximizing its dissemination and validating its reproducibility in science and engineering worldwide.
Methods: The peer-reviewed Python program on gas chromatography is converted to the PyPI application. Third-party evaluations motivate learners to boost programming skills. The more the downloads from third-party evaluations, the greater the incentive for learners.
Results: The gas chromatography program is successfully converted to agci PyPI application. Learners can be excited to check their progress on external downloads to motivate them to learn and program. This tutorial shows how to verify software reproducibility of the PyPI application via Code Ocean.
Conclusion: This paper demonstrated the real gas chromatography program to the PyPI application and showed how to verify software reproducibility for maximizing dissemination skills and boosting the motivation of learners for advancing chemometrics research.

1. Introduction

The data analysis approach for gas chromatography is designed to be flexible through programming. In today's technological age, it is becoming increasingly important for chemists to fully utilize the computational power of computers. Python is a scripting language most commonly used in all areas of science and technology. Python Package Index or PyPI is a management tool for maximally disseminating created Python programs in the world. PyPI allows the PyPI applications to run on Windows, MacOS, and Linux operating systems as long as Python is installed on the system. Learners and instructors must know how to debut a PyPI application for maximum dissemination of software.

This paper presents an example of converting a gas chromatography Python program into a PyPI application and verifying its reproducibility using Code Ocean. PyPI enables users worldwide to easily download and utilize such applications. These steps are essential for advancing chemometrics research, yet there is a lack of tutorials on the subject.

Teachers and instructors are always interested in tools that can motivate their learners. PyPI provides programmers and learners with greater motivation and incentives through third-party evaluations. The number of downloads serves as an indicator of the popularity of the developed software, and as this number increases, so does the motivation of the programmer. Our observations show that when a learner's PyPI application experiences a rapid increase in downloads from third parties, their motivation increases dramatically in proportion to the number of downloads. PyPI is a valuable tool for increasing learners' motivation and incentive to learn Python programming.

There are many tutorials on PyPI over the Internet such as the official PyPI site. However, even the official PyPI site has not updated the tutorial and many uploading users have failed due to the twine upload library from users to the PyPI site. The latest twine has changed the authenticity of uploading file. This paper's contribution is significant for PyPI users to solve their uploading problems. PyPI allows scientists and engineers to maximize dissemination of their software applications to the world for vitalization of industries and the advance of science. Tools such as PyInstaller can automatically convert Python code into a PyPI application. However, it is still necessary to prepare several important files, such as setup.py and README.md, and to use twine to upload

E-mail address: takefuji@keio.jp.

https://doi.org/10.1016/j.chemolab.2023.104941
Received 30 June 2023; Received in revised form 20 August 2023; Accepted 22 August 2023
Available online 25 August 2023

important files to the PyPI website where Twine uploading scheme was changed recently. This paper presents the fastest way for instructors and learners to debut their Python programs to the world as PyPI applications.

We define usefulness as the degree to which application's functionality can help a given user achieve his or her goals. There is no challenge to study the software usefulness. In existing studies, software has been playing a key role in science and technology, but learner's motivation on software programming has been neglected.

Traditional studies on software usefulness are based on questionnaire survey [1–3] or conducting interviews with experienced assessors [4]. Hanrahan et al. reported that the usefulness of an app will largely depend on the purpose of that app relative to one's practice setting [5].

With the rapid progress of open-source software, it is possible to evaluate the software usefulness using packaging information. It is common to assume that the usefulness of software to professionals is proportional to its visibility, such as the number of downloads. In other words, third parties believe that the more useful a PyPI application is, the more downloads it will receive. As the number of downloads increases, learners receive stronger incentives.

Basic knowledge of Python programming is highly needed in science and technology. However, the tutorial on PyPI applications is not offered in many refereed journals. The purpose of this paper is to provide a tutorial on the PyPI applications. Many refereed journals emphasize software reproducibility [6–11], but they have no updated tutorials on PyPI applications and software reproducibility in their journals. The updated tutorial on PyPI and software reproducibility via Code Ocean will make a significant contribution to science and engineering.

According to the 2022 TIOBE Index, the number one programming language is Python, followed by C and Java [12]. It is essential for scientists and engineers in general to understand and deploy PyPI packaging and its software reproducibility verification.

Using an actual Python program (Asymmetric GC integration.py) [13] published in the chemical journal, the program will be transformed into the renamed PyPI application (agci.py) in order to illustrate exactly how to debut a PyPI application. PyPI environment allows the program to run on Windows, MacOS, and Linux operating systems without being aware of operating systems as long as Python is installed on the system.

The key to analyzing data in this program with agci.py, based on the pedagogical framework, is to understand that the normal distribution of phase velocity in gas chromatography is typically Gaussian. This is because the transfer of individual molecules from the stationary phase to the mobile phase within the GC column is random. The "agci.py" model currently calculates the areas of two peaks, those of acetone and cyclohexane. To enable users to separate more components in the initial mixture, the system allows them to include additional sets of initial values. To determine the mole fraction of the distillate, two acetone/cyclohexane solutions are used in datasets.

QSAR is a powerful tool for understanding how molecular structure affects biological activity [15, 16, 17]. It has been used to develop models that can predict the activity of new compounds, but these models need to be interpretable to be useful. Shoombuatong et al. reviewed the concepts of QSAR modeling and discussed the key issues that influence interpretability in computational chemistry and physics.

Reproducibility is essential in science, and the reproducibility of peer-reviewed articles must be rigorously assessed. This paper provides the source code to the reader through PyPI and uses Code Ocean to objectively guarantee reproducibility. The author not only introduces these resources, but also provides actual examples of Gas Chromatography programs and detailed explanations of the surrounding commands to ensure complete reproducibility.

This paper will also show you step-by-step how to validate software reproducibility of the converted Python application via Code Ocean. By providing the specific example, this paper can fill in the missing gaps in the current teaching environment and advanced software development environment for advancing practice skills in teaching.

PyPI is a valuable tool for maximizing the dissemination of chemometrics programs and boosting learner motivation. It is important for chemometrics researchers to know how to convert their Python programs into PyPI applications and verify their reproducibility using Code Ocean.

2. Methods

2.1. PyPI packaging

The instructions were created for a Linux terminal. PyPI needs three files such as README.md, setup.py and agci.py respectively. The changes from the original program (Asymmetric Gas Chromatography integration.py [13]) are to allow input of sample csv files, to create a new main() function, and to embed a function to save the result image as result.png. The renamed program is called agci.py [14]. Shaded seven lines in the source code in Fig. 1 show newly added lines to the original program.

It is recommended that filename should not contain spaces since it is difficult to know how many spaces there are between strings of filename. To indicate that a filename contains spaces, we should use underscores (_) instead of spaces.

This conversion from Asymmetric GC integration.py to agci.py does not require any knowledge of chemistry. It took less than 5 min for this conversion.

README.md file can be easily built with GitHub site creating a new repository with a README file option.

The following setup.py template file can be used for a PyPI application. Shaded ten lines in Fig. 2 should be modified for your application:

The following is necessary steps to debut a PyPI application.

1 Go to https://PyPI.org/account/register for creating a new account
2 To debut a PyPI application by uploading files to PyPI site, we need twine library.

$ pip install twine.

The structure of directory (src) and files (README.md, setup.py, agci.py) is as follows:

$ tree.

```
├── README.md
├── setup.py
└── src
    ├── agci.py
```

However, before run the command, you need to create the authentication file (.pypirc) in your home directory. The file.pypirc is with three lines. The first two lines are fixed which you don't have to modify. The last line should be modified with your unique PyPI API token. Go to you PyPI account and click Account settings. The API token can be generated by clicking "Add 2FA with authentication application".

```
[pypi]
username = __token__
password = <your PyPI token>
```

Once.pypirc file is ready, run the following four commands to upload two files to PyPI site for a new PyPI debut. twine(4.0.2) was used.

```
$ python setup.py install
$ python setup.py sdist bdist_wheel
$ rm dist/*.egg
$ twine upload dist/*
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import leastsq
from scipy.integrate import quad
import sys
data_set = pd.read_csv(sys.argv[1]).to_numpy()
initials = [
    [4.5, 13, 1],
    [2.5, 19, 1]
]
n_value = len(initials)
def gaussian(x,a,b,c):
    return a*np.exp((-(x-b)**2.0)/c**2.0)
def GaussSum(x, p, n):
    return sum(gaussian(x, p[3*k], p[3*k+1], p[3*k+2]) for k in range(n))
def residuals(p, y, x, n):
    return y - GaussSum(x,p,n)
cnsts =  leastsq(
            residuals,
            initials,
            args=(
                data_set[:,1],          # y data
                data_set[:,0],          # x data
                n_value                 # n value
            )
        )[0]
areas = dict()
for i in range(n_value):
    areas[i] = quad(
        gaussian,
        data_set[0,0],      # lower integration bound
        data_set[-1,0],     # upper integration bound
        args=(
            cnsts[3*i],
            cnsts[3*i+1],
            cnsts[3*i+2]
        )
    )[0]
x = np.linspace(data_set[0,0],data_set[-1,0],200)
fig, ax = plt.subplots(dpi = 100)
ax.tick_params(direction = 'in', pad = 15)
ax.set_xlabel('time / s', labelpad = 20, fontsize = 15)
ax.set_ylabel('Intensity / a.u.', labelpad = 20, fontsize = 15)
ax.plot(data_set[:,0], data_set[:,1], 'ko')
ax.plot(x,GaussSum(x,cnsts, n_value))
for i in range(n_value):
    ax.plot(
        x,
        gaussian(
            x,
            cnsts[3*i],
            cnsts[3*i+1],
            cnsts[3*i+2]
        )
    )
ledger = ['Data', 'Resultant']
for i in range(n_value):
    ledger.append(
        str(round(cnsts[3*i+1], 2)) +
        '$e^{(x-' + str(round(cnsts[3*i], 2)) +
        ')^2 / ' + str(round(cnsts[3*i + 2], 2)) +
        '^2}$' + '\n Area = ' + str(round(areas[i], 3))
    )
ax.legend(ledger)
plt.tight_layout()
def main():
  plt.savefig('result.png')
  plt.show()
if __name__ == "__main__":
  main()
```

Fig. 1. Python program.

If your submission is successful, URL will be shown in the terminal. If you want to update the PyPI program, delete all files under sdist/and build/directory respectively by the following command:

```
$ rm -rf dist/* build/*
```

2.2. Reproducibility via Code Ocean

The following steps are needed for validation of software reproducibility.

1. Go to https://codeocean.com/signup for creating a new account.
2. Click “Add Capsule” button and select “Create New”
3 Environment file will be popped and fill all in the Environment file.

Select Python3.8.5 with four Python libraries such as numpy (1.22.3), pandas(1.4.2), matplotlib(3.5.1), scipy(1.8.0) and agci(0.0.1) using pip3.

4 Click “Start with Sample Files” button to generate run file. run file is modified as follows.

```
#!/usr/bin/env bash
set -ex
cp sample1.csv ../results
cd ../results
agci sample1.csv
```

5. Click “Submit for publication” button and “Pre submit” form will be popped. Click all squares and click “Go to metadata”.
6. Fill out the metadata form and click the blue colored “Submit” button in Pre submit form.
7. Within a few days, Code Ocean will send you a message of “ACCEPTED FOR PUBLICATION”.

3. Discussions

This paper presented how to debut a PyPI application using a real example. Using a simple Python program published in the Journal of Chemical Education, the original program was converted to the PyPI application with several lines changed and renamed agci.py with setup.py and README.md files newly added.

The modified original Python program is able to run on Windows, MacOS, and Linux operating systems without being aware of operating systems as long as Python is installed on the system.

In the original program, modifications are needed to run on individual operating systems. In other words, the PyPI environment allows the PyPI application to maximize worldwide software dissemination.

Verification of software reproducibility was also demonstrated in a detailed procedure via Code Ocean. By providing the specific example, this paper can fill in the missing gaps in chemistry and current chemical education for advancing chemical technology and education.

According to PePy: https://pepy.tech/project/agci, the AGCI application has been downloaded 3117 times worldwide. The number of

```
import setuptools
with open("README.md", "r", encoding="utf-8") as fh:
    long_description = fh.read()
setuptools.setup(
    name="agci",
    version="0.0.1",
    author="Author_name",
    author_email="email_address",
    description="how to debut a PyPI for chemistry",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://github.com/author/agci",
    project_urls={
        "Bug Tracker": "https://github.com/author/agci",
    },
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    package_dir={"": "src"},
    py_modules=['agci'],
    packages=setuptools.find_packages(where="src"),
    python_requires=">=3.7",
    entry_points = {
        'console_scripts': [
            'agci = agci:main'
        ]
    },
)
```

Fig. 2. Setup.py.

users indicates the applicability, the usefulness, the reproducibility and the proposed claim were justified. The more the downloads, the stronger the incentive for programmers or learners. External evaluation, such as the number of downloads, is essential to incentivize learners and programmers. In the current engineering & science teaching, PyPI has never been discussed from learner' motivation tool. Therefore, the contribution of this paper is important for teaching in science and technology. This paper demonstrated how to convert the gas chromatography Python program to the agci PyPI application and to verify it via Code Ocean for advancing chemometric research.

Please note that the agci.py program, which was converted from the original Python program (Asymmetric GC integration.py), can only handle a single set of initial values with one peak in the chromatogram. This is for educational purposes only.

Reproducibility was fulfilled by providing the source code to the reader through PyPI and uses Code Ocean to objectively guarantee reproducibility. The author not only introduced these resources, but also provides actual examples of Gas Chromatography programs and detailed explanations of the surrounding commands to ensure complete reproducibility.

Conclusions

The outdated tutorials on PyPI, including the official website, have left PyPI users unable to upload packages due to the new authentication requirements of the twine library. The tutorial on PyPI in this paper has been updated to no longer cause PyPI users to fail to upload PyPI packages. This paper provides an example of how to convert an original Python program into a PyPI application. This paper shows that PyPI is a new tool that motivates learners through third-party evaluations and the number of downloads. This is because the learner can observe the progress and transition of the download in chronological order. The more downloads, the stronger the incentive for learners. Finally, the reproducibility verification of software via Code Ocean was presented.

Limitation

The agci.py program is a pedagogical tool for understanding the Gaussian distribution of phase velocity in gas chromatography. This distribution arises from the random transfer of individual molecules from the stationary phase to the mobile phase within the GC column. As a result, the program can only handle a single set of initial values with one peak in the chromatogram. Two acetone/cyclohexane solutions are used in datasets to determine the mole fraction of the distillate.

The methods presented in this paper may someday stop working due to version upgrades, as the original PyPI manual did. This is because software distribution over the Internet is a daily process, and the methods presented in this paper may not be updated to reflect changes in the software distribution landscape. The date of execution is confirmed as August 1, 2023.

Author statement

YT completed this research and wrote the Python program and this article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] Leandro Flores da Silva, Edson Oliveira, Evaluating usefulness, ease of use and usability of an UML-based software product line tool, in: *Proceedings of the 34th Brazilian Symposium on Software Engineering* (*SBES '20*), Association for Computing Machinery, New York, NY, USA, 2020, pp. 798–807, https://doi.org/10.1145/3422392.3422402.

[2] Robert W. Root, Steve Draper, Questionnaires as a software evaluation tool, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '83), Association for Computing Machinery, New York, NY, USA, 1983, pp. 83–87, https://doi.org/10.1145/800045.801586.

[3] C. Ruman, J. Gillette, Distance learning software usefulness and usability: user-centered issues in practical deployment, in: Ed at a Distance Journal, vol. 15, 2001, p. 3, n3.

[4] E. Bouwers, A. van Deursen, J. Visser, Evaluating Usefulness of Software Metrics: an Industrial Experience Report, 2013, pp. 921–930, https://doi.org/10.1109/ICSE.2013.6606641, 35th International Conference on Software Engineering (ICSE), 2013.

[5] C. Hanrahan, T.D. Aungst, S. Cole, Evaluating Mobile Medical Applications, 2014 ashp publications, https://www.ashp.org/-/media/store%20files/mobile-medical-apps.pdf.

[6] J.M. Perkel, Challenge to scientists: does your ten-year-old code still run? Nature 584 (7822) (2020) 656–658, https://doi.org/10.1038/d41586-020-02462-7. PMID: 32839567.

[7] Julia Koehler Leman, et al., Bonneau Ensuring scientific reproducibility in bio-macromolecular modeling via extensive, automated benchmarks, Nat. Commun. 12 (2021) 6947, https://doi.org/10.1038/s41467-021-27222-7.

[8] But is the code (re)useable? Nat Comput Sci 1 (2021) 449, https://doi.org/10.1038/s43588-021-00109-9.

[9] Supporting computational reproducibility through code review, Nat. Human Behav. 5 (2021) 965–966, https://doi.org/10.1038/s41562-021-01190-w.

[10] J.F. Pimentel, L. Murta, V. Braganholo, Juliana Freire Understanding and improving the quality and reproducibility of Jupyter notebooks, Empir. Software Eng. 26 (2021) 65, https://doi.org/10.1007/s10664-021-09961-9.

[11] Mukherjee, Suchita, Almanza, Abigail, & Rubio-González, Cindy. Fixing dependency errors for Python build reproducibility. https://doi.org/10.1145/3460319.3464797 Retrieved from https://par.nsf.gov/biblio/10284919.

[12] https://www.tiobe.com/tiobe-index/.

[13] Michael Green, Xiaobo Chen, Data functionalization for gas chromatography in Python, J. Chem. Educ. 97 (4) (2020) 1172–1175, https://doi.org/10.1021/acs.jchemed.9b00818.

[14] Takefuji, Y. (2022) Agci for a reproducible PyPI application [Source Code]. https://doi.org/10.24433/CO.8712645.v1.

[15] W. Shoombuatong, et al., Towards the revival of interpretable QSAR models, in: K. Roy (Ed.), Advances in QSAR Modeling. Challenges and Advances in Computational Chemistry and Physics, vol. 24, Springer, Cham, 2017, https://doi.org/10.1007/978-3-319-56850-8_1.

[16] W. Shoombuatong, N. Schaduangrat, C. Nantasenamat, Towards understanding aromatase inhibitory activity via QSAR modeling, EXCLI J, 17 (2018) 688–708, https://doi.org/10.17179/excli2018-1417.

[17] W. Shoombuatong, P. Prathipati, V. Prachayasittikul, N. Schaduangrat, A.A. Malik, R. Pratiwi, S. Wanwimolruk, J.E.S. Wikberg, M.P. Gleeson, O. Spjuth, C. Nantasenamat, Towards predicting the cytochrome P450 modulation: from QSAR to proteochemometric modeling, Curr. Drug Metabol. 18 (6) (2017) 540–555, https://doi.org/10.2174/1389200218666170320121932.