quica Documentation

Release 0.2.5

Federico Bianchi

Nov 09, 2020

CONTENTS:

CHAPTER

ONE

QUICK INTER CODER AGREEMENT IN PYTHON

Quica (Quick Inter Coder Agreement in Python) is a tool to run inter coder agreement pipelines in an easy and effective way. Multiple measures are run and results are collected in a single table than can be easily exported in Latex. quica supports binary or multiple coders.

Quick Inter Coder Agreement in Python

- Free software: MIT license
- Documentation: https://quica.readthedocs.io.

1.1 Installation

```
pip install -U quica
```

1.2 Jump start Tutorial

Name	Link
Different possible usages of QUICA	

1.3 Get Quick Agreement

If you already have a python dataframe you can run Quica with few liens of code! Let's assume you have two coders; we will create a pandas dataframe just to show how to use the library. As for now, we support only integer values and we still have not included weighting.

```
from quica.quica import Quica
import pandas as pd

coder_1 = [0, 1, 0, 1, 0, 1]
coder_3 = [0, 1, 0, 1, 0, 0]

dataframe = pd.DataFrame({"coder1" : coder_1,
              "coder3" : coder_3})

quica = Quica(dataframe=dataframe)
print(quica.get_results())
```

This is the expected output:

```
Out[1]:
             score
names
krippendorff  0.685714
fleiss        0.666667
scotts        0.657143
raw           0.833333
mace          0.426531
cohen         0.666667
```

It was pretty easy to get all the scores, right? What if we do not have a pandas dataframe? what if we want to directly get the latex table to put into the paper? worry not, my friend: it's easier done than said!

```
from quica.measures.irr import *
from quica.dataset.dataset import IRRDataset
from quica.quica import Quica

coder_1 = [0, 1, 0, 1, 0, 1]
coder_3 = [0, 1, 0, 1, 0, 0]

disagreeing_coders = [coder_1, coder_3]
disagreeing_dataset = IRRDataset(disagreeing_coders)

quica = Quica(disagreeing_dataset)

print(quica.get_results())
print(quica.get_latex())
```

you should get this in output, note that the latex table requires the booktabs package:

```
Out[1]:
             score
names
krippendorff  0.685714
fleiss        0.666667
scotts        0.657143
raw           0.833333
mace          0.426531
cohen         0.666667

Out[2]:

\begin{tabular}{lr}
```

(continues on next page)

(continued from previous page)

```
\toprule
{} &     score \\
names        &           \\
\midrule
krippendorff &  0.685714 \\
fleiss       &  0.666667 \\
scotts       &  0.657143 \\
raw          &  0.833333 \\
mace         &  0.426531 \\
cohen        &  0.666667 \\
\bottomrule
\end{tabular}
```

1.4 Features

```
from quica.measures.irr import *
from quica.dataset.dataset import IRRDataset
from quica.quica import Quica

coder_1 = [0, 1, 0, 1, 0, 1]
coder_2 = [0, 1, 0, 1, 0, 1]
coder_3 = [0, 1, 0, 1, 0, 0]

agreeing_coders = [coder_1, coder_2]
agreeing_dataset = IRRDataset(agreeing_coders)

disagreeing_coders = [coder_1, coder_3]
disagreeing_dataset = IRRDataset(disagreeing_coders)

kri = Krippendorff()
cohen = CohensK()

assert kri.compute_irr(agreeing_dataset) == 1
assert kri.compute_irr(agreeing_dataset) == 1
assert cohen.compute_irr(disagreeing_dataset) < 1
assert cohen.compute_irr(disagreeing_dataset) < 1
```

1.5 Supported Algorithms

- **MACE (Multi-Annotator Competence Estimation)**
  - Hovy, D., Berg-Kirkpatrick, T., Vaswani, A., & Hovy, E. (2013, June). Learning whom to trust with MACE. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 1120-1130).
  - We define the inter coder agreeement as the average competence of the users.
- Krippendorff's Alpha
- Cohens' K
- Fleiss' K
- Scotts' PI

- Raw Agreement: Standard Accuracy

1.6 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template. Thanks to Pietro Lesci and Dirk Hovy for their implementation of MACE.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install quica, run this command in your terminal:

```
$ pip install quica
```

This is the preferred method to install quica, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for quica can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/vinid/quica
```

Or download the tarball:

```
$ curl -OJL https://github.com/vinid/quica/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use quica in a project:

```
import quica
```

CHAPTER

FOUR

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/vinid/quica/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

quica could always use more documentation, whether as part of the official quica docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/vinid/quica/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *quica* for local development.

1. Fork the *quica* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/quica.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv quica
$ cd quica/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 quica tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/vinid/quica/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ pytest tests.test_quica
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Federico Bianchi <f.bianchi@unibocconi.it>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

HISTORY

6.1 0.1.0 (2020-11-08)

- New API to get the output
- Fixed test cases
- Extended documentation on the README file

6.2 0.1.0 (2020-11-05)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search