Pymicra:

A Python tool for Micrometeorological Analyses

Tomás L. Chor[1] tomaschor@ucla.edu
Nelson L. Dias[2] nldias@ufpr.br

[1]University of California, Los Angeles, Department of Atmospheric and Oceanic Sciences

[2]Federal University of Paraná, Department of Environmental Engineering

Summary

Pymicra is a Python package that is designed to make micrometeorological data processing significantly more efficient.

It was written with the goal of becoming a community-driven software that is a full-blown micrometeorology package, not an just eddy-covariance one. It was designed to make research more efficient and to make sharing code easier and it was motivated by many examples of similar initiatives that thrived: Obspy for seismologists [3], Astropy for astrophysicists [2], Sunpy for solar physicists [1], Metpy for weather scientists [4], etc.
Here are a few characteristics that make Pymicra a good candidate to be community-driven:

- Open-source
- Code written with Pymicra is very readable and intuitive
- Visualization of data is extremely easy
- Easy to develop upon (because of well-documented Python code)
- Based on well-known mature Python packages

Pymicra's philosophy is not to "reinvent the wheel". There are already several open-source packages that process, organize and manipulate data very efficiently, so the goal was simply to put them together is a way specifically designed to the micrometeorological community.

Quality control example

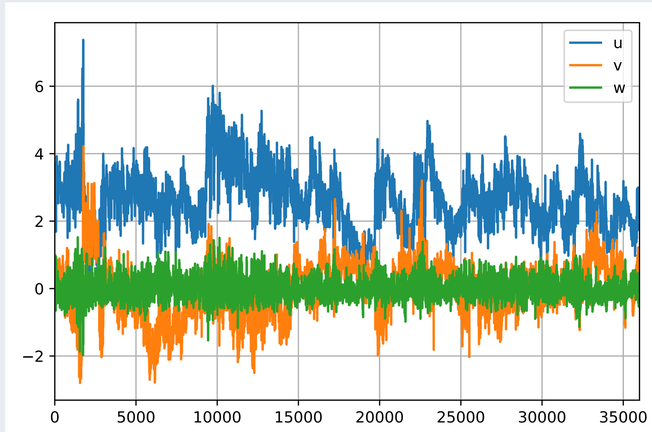
```
In [1]: fnames = sorted(glob('mydata/*.out'))
        # Prints reports on screen and writes further info to file
        pymicra.util.qc_replace(fnames, fconfig,
            file_lines=36000,
            lower_limits=dict(theta_v=10, mrho_h2o=0, mrho_co2=0),
            upper_limits=dict(theta_v=45),
            spikes_test=True,
            max_replacement_count=360,
            chunk_size=1200,
            outdir='out1',
            replaced_report='rrep.txt')

        fnames2 = sorted(glob('out1/*.out'))
        # Prints reports on screen and writes further info to file
        pymicra.util.qc_discard(fnames2, fconfig,
            std_limits = dict(u=0.03, v=0.03, w=0.01, theta_v=0.02),
            dif_limits = dict(u=4.0, v=4.0, w=1.0, theta_v=2.0),
            chunk_size=1200,
            outdir='out2',
            summary_file='discard_summary.csv',
            full_report='frep.txt')
```

Pre-processing and calculation of fluxes example

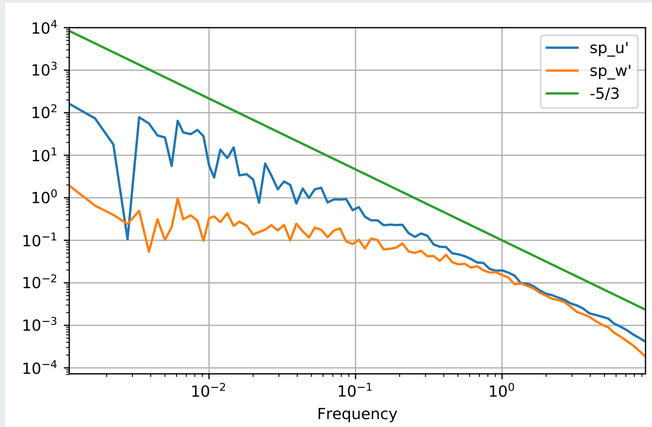
```
In [2]: fconfig = pymicra.fileConfig('tij_pr.config')
        sconfig = pymicra.siteConfig('tij_pr.site')
        fname = 'out2/20110224-1340.out'

        data, units = pymicra.timeSeries(fname, fconfig,
                                         parse_dates=False)
        # Prints reports showing which calculations are being done
        data = pymicra.micro.preProcess(data, units, solutes=['co2'])
        fulldata = data.detrend(units=units, ignore=['p'],
                                join_data=True)
        # Prints reports showing which calculations are being done
        results = pymicra.micro.eddyCovariance(fulldata, units,
                    site_config=sconfig, wpl=True, solutes=['co2'])
```

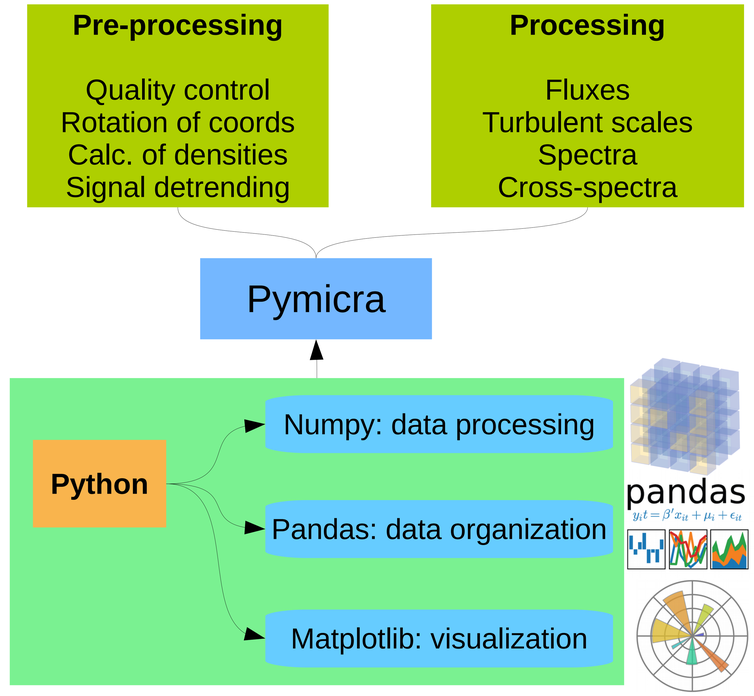
Visualization example

```
In [3]: data[['u', 'v', 'w']].plot(grid=True)
        plt.show()
```




```
In [4]: uw_spectra = pymicra.spectra(fulldata[["u'", "w'"]],
                                      frequency=20,
                                      anti_aliasing=True)
        axis = uw_spectra.binned(bins_number=100).plot(loglog=True,
                                      grid=True)
        axis.plot(uw_spectra.index, 1e-1*uw_spectra.index**(-5/3),
                                      label='-5/3')
        axis.legend(); plt.show()
```





Why Pymicra instead of writing my own code?

There are many advantages in migrating to a community package such as Pymicra aims to be, both community-wise and for the individual:

- Package becomes very reliable (more people are constantly checking for bugs and improving)
- The amount of code to process the data is significantly smaller (thus faster to write)
- Sharing your code and understanding other people's is easier
- Code adaptability (no need to keep writing *ad hoc* code for each dataset)
- Flexibility (you can easily use Pymicra to do basic processing and link it to other tools for more specific things)

Pymicra's docs ↓

How to contribute

The code is hosted on Github and any person can fork, develop functionality into it and request a merge back.
Bug reports, suggestions and documentation improvements are also very welcome via Github issues or via email.
Scan the QR code for more information on contributing.

[1] T. S. e. a. Community. SunPypython for solar physics. *Computational Science & Discovery*, 8(1):014009, 2015.
[2] T. et al. Astropy: A community python package for astronomy. *Astronomy & Astrophysics*, 558:A33, 2013.
[3] L. Krischer, T. Megies, R. Barsch, M. Beyreuther, T. Lecocq, C. Caudron, and J. Wassermann. ObsPy: a bridge for seismology into the scientific python ecosystem. *Computational Science & Discovery*, 8(1):014003, 2015.
[4] R. May, S. Arms, P. Marsh, E. Bruning, and J. Leeman. Metpy: A Python package for meteorological data, 2008 - 2017. URL https://github.com/Unidata/MetPy.