PyAstronomy

Programmer's Club
30 April 2015

Jason Neal

Can be found at
http://www.hs.uni-hamburg.de/DE/Ins/Per/Czesla/PyA/PyA/index.html#

Overview:

- What is it?
- Packages available
- My usage
- Examples

What is it?

- Collection of astronomy related packages for python.
- Not developed for a specific field
- Contains many of the “IDL Astronomy User’s Library” (AstroLib)

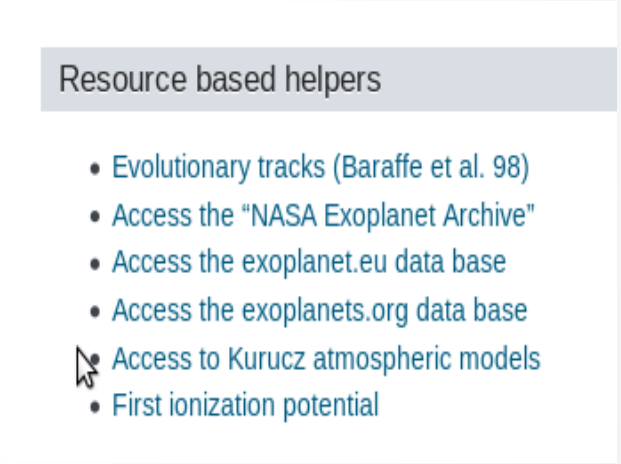
Python AstroLib

- General data analysis
  - Smoothing data, SNR estimates, Folding time series
- Spectral analysis
  - Cross-correlation, Doppler shifting spectrum, Broadening mechanisms, Barycentric velocity correction
- Coordinates time, and observation tools
  - Lunar phase, Sun position, date conversion, airmass, observatory location

- Transit and planet routines
  - Transit duration, visibility, planet phases
- Statistics
  - F-test
- Miscellaneous
  - Magnitude conversions, S-index and RHK, Stellar ages, effective temperature vs stellar colour.
- Resource based helpers
  - Access to - Atmospheric models, exoplanets.org, NASA Exoplanet Archive, First ionization potential.

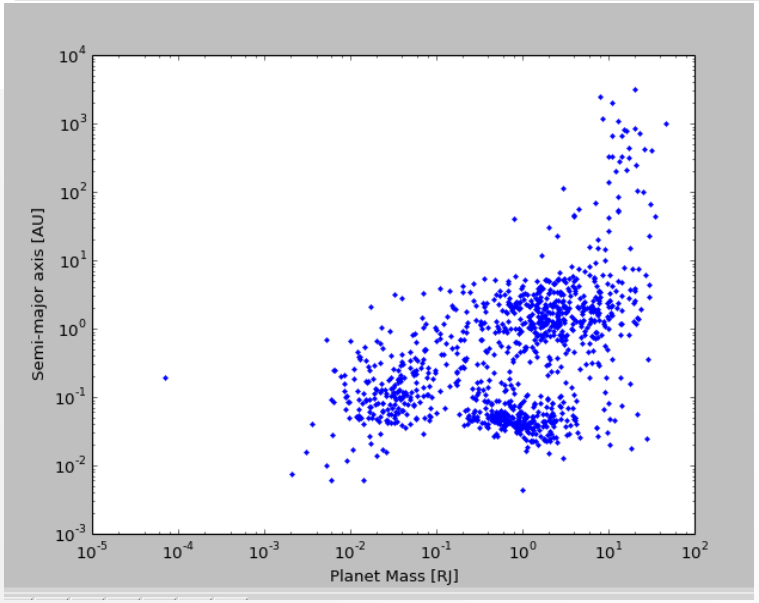
Access the exoplanet.eu database

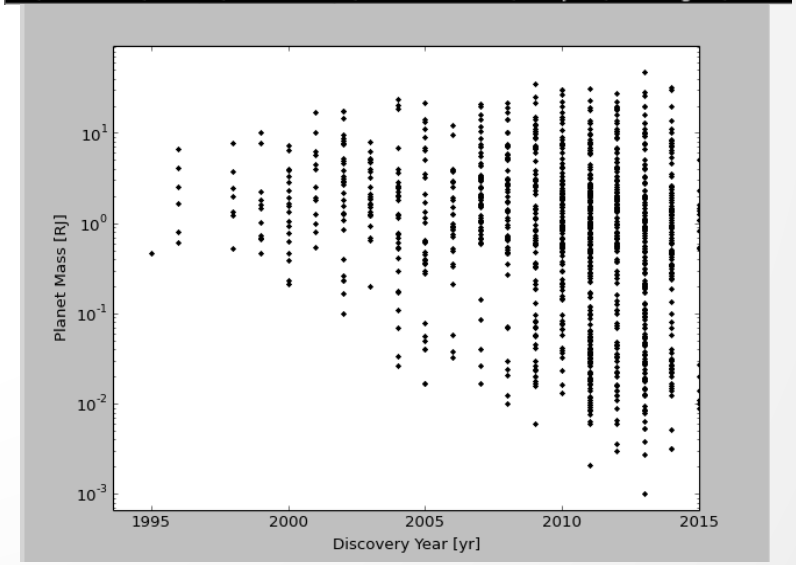
- Easy access of exoplanet database.
  - Don't have to try play with large tables
- Downloads database to your computer and updates regularly.
- Con - didn't bring the errors/uncertainties



```
>>> # See what information is available
... cols = eu.availableColumns()
-------------------------------------------------
 Column Name                    Description   Unit
-------------------------------------------------
      plName                  Name of planet
      plMass                  Mass of planet     MJ
    plRadius                Radius of planet     RJ
      period                  Orbital period      d
         sma                 Semi-major axis     AU
eccentricity           Orbital eccentricity
 inclination            Orbital inclination    deg
 angDistance               Angular Distance  arcsec
   pubStatus              Publication status
  discovered               Year of discovery     yr
     updated            Date of data update
       omega          Argument of Periastron    deg
       tperi             Epoch of Periastron      d
     detType                  Detection type
   molecules      List of detected molecules
      stName                    Name of star
          ra         Right ascension (J2000)    hms
         dec             Declination (J2000)    dms
        mag_v      V magnitude of a host star    mag
        mag_i      I magnitude of a host star    mag
        mag_j      J magnitude of a host star    mag
        mag_h      H magnitude of a host star    mag
        mag_k      K magnitude of a host star    mag
         dist          Distance to host star     pc
           mh       Metallicity of host star    dex
      stMass                    Stellar mass  solar
    stRadius                  Radius of star  solar
         SpT      Spectral type of host star
       stAge                     Stellar age     Ga
      stTeff   Stellar effective temperature      K
-------------------------------------------------
>>> print cols
['plName', 'plMass', 'plRadius', 'period', 'sma', 'eccentric
 'angDistance', 'pubStatus', 'discovered', 'updated', 'omega
', 'molecules', 'stName', 'ra', 'dec', 'mag_v', 'mag_i', 'ma
k', 'dist', 'mh', 'stMass', 'stRadius', 'SpT', 'stAge', 'st
```

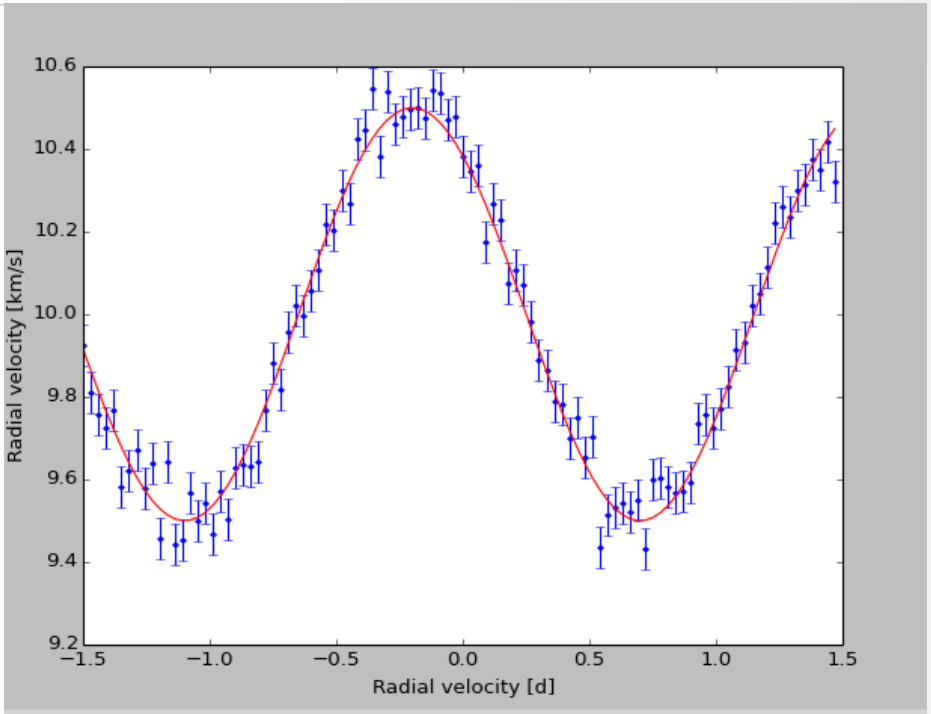
```python
from PyAstronomy import pyasl
import matplotlib.pylab as plt

eu = pyasl.ExoplanetEU()

# See what information is available
cols = eu.availableColumns()
print cols

print
# Get all data and plot planet Mass vs.
# semi-major axis in log-log plot
dat = eu.getAllData()
plt.xlabel("Planet Mass [RJ]")
plt.ylabel("Semi-major axis [AU]")
plt.loglog(dat.plMass, dat.sma, 'b.')
plt.show()
```





Fitting Function - RV Fitting

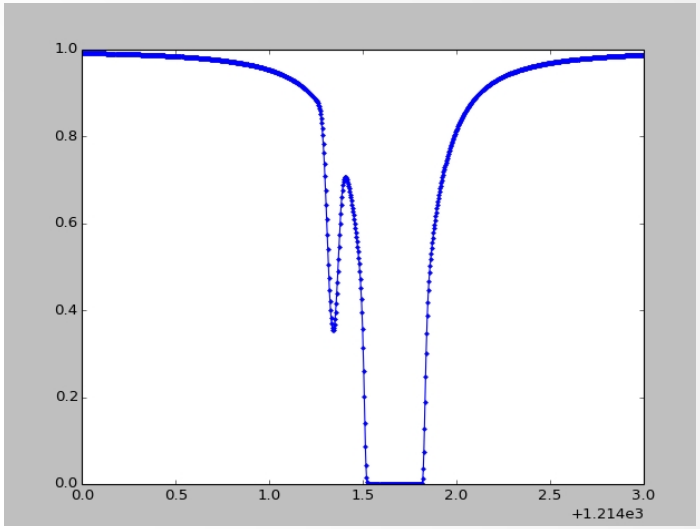
```
# Import some unrelated modules
from numpy import arange, random, ones
import matplotlib.pylab as mpl
# ... and now the radVel module
from PyAstronomy.modelSuite import radVel as rv

# Create Radial Velocity object
r = rv.SinRadVel()
# Set parameters
r.assignValue({"P":1.8, "T0":0.25, "K": 0.5, "rv0": 10.0})
# Choose some time axis and calculate model
time = arange(100)/100.0 * 3.0 - 1.5
y = r.evaluate(time)

# Create some faked data by adding noise
rvData = y + random.normal(0.0, 0.05, y.size)

# Randomize starting parameters for fit
for p, v in r.parameters().iteritems():
  r[p] = v + (random.random() - 0.5) * v
# Show starting values
print "Starting values for fit:"
r.parameterSummary()

# Thaw all parameters
r.thaw(r.parameters().keys())
# Start the fit
r.fit(time, rvData, yerr=ones(y.size)*0.05)

# Show fit results
print "Fitted values:"
r.parameterSummary()

# Let's see what happened...
mpl.ylabel("Radial velocity [km/s]")
mpl.xlabel("Radial velocity [d]")
mpl.errorbar(time, rvData, yerr=ones(y.size)*0.05, fmt='b.')
mpl.plot(time, y, 'r-')
mpl.show()
```



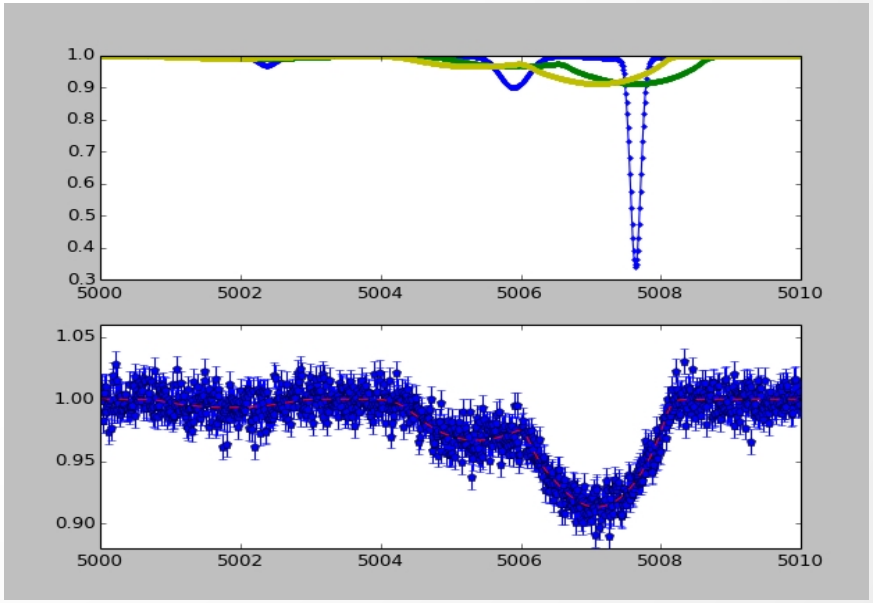
PyA model suite

- Transit modeling
  - Transit light-curves
  - Rossiter McLaughlin curves
- Circular orbit—radial-velocity shift
  - Example code - Fit RV curve
- The Kepler-ellipse model
  - Example: Evaluating and fitting model using MCMC
- Line list based Gaussian spectral model
  - Example: Evaluation and fitting
- Voigt profile with astronomical parameterization
  - Example
- Hydrogen Lyman-alpha line-profile
  - Example
- Rotational broadening profile
  - Example of usage

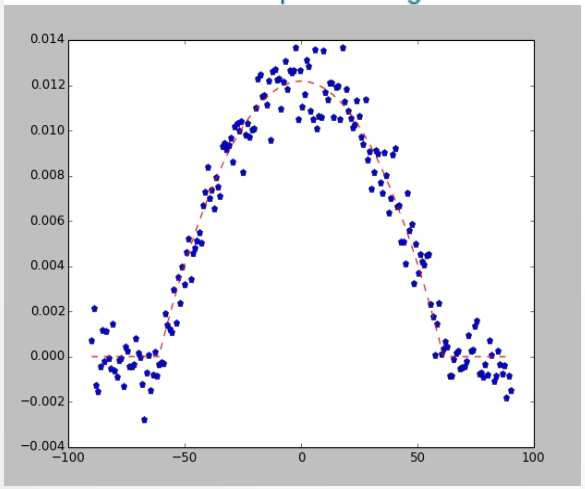
Lyman alpha line profile



Gaussian spectral model



Rotational broadening profile

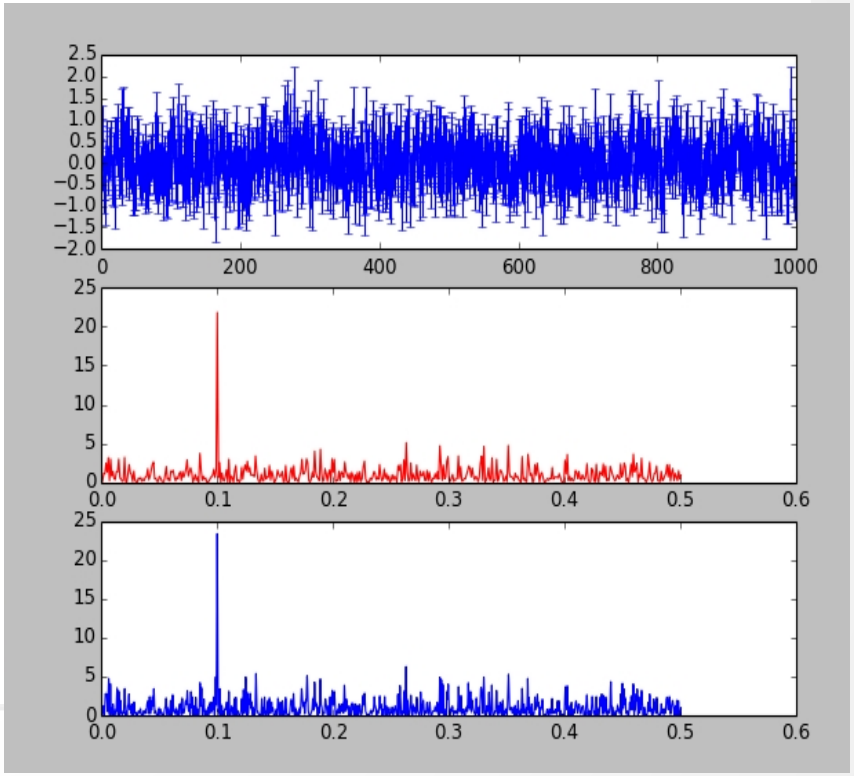


Harmonic timing analysis using periodograms

The search for harmonic signals in the presence of noise is the fundamental problem of timing analysis. This package provides a collection of periodogram implementations that conceptually rely on the methods of Fourier analysis.

Contents:

- Base classes
  - The *TimeSeries* base class
  - The PeriodBase Base class
- Periodograms
  - The Fourier Transform
  - The Lomb-Scargle-Periodogram (*fast*)
  - The generalized Lomb-Periodogram
  - References
- Examples
  - Fourier spectrum of evenly-sampled Poisson data
  - Error-weighted (generalized) Lomb periodogram

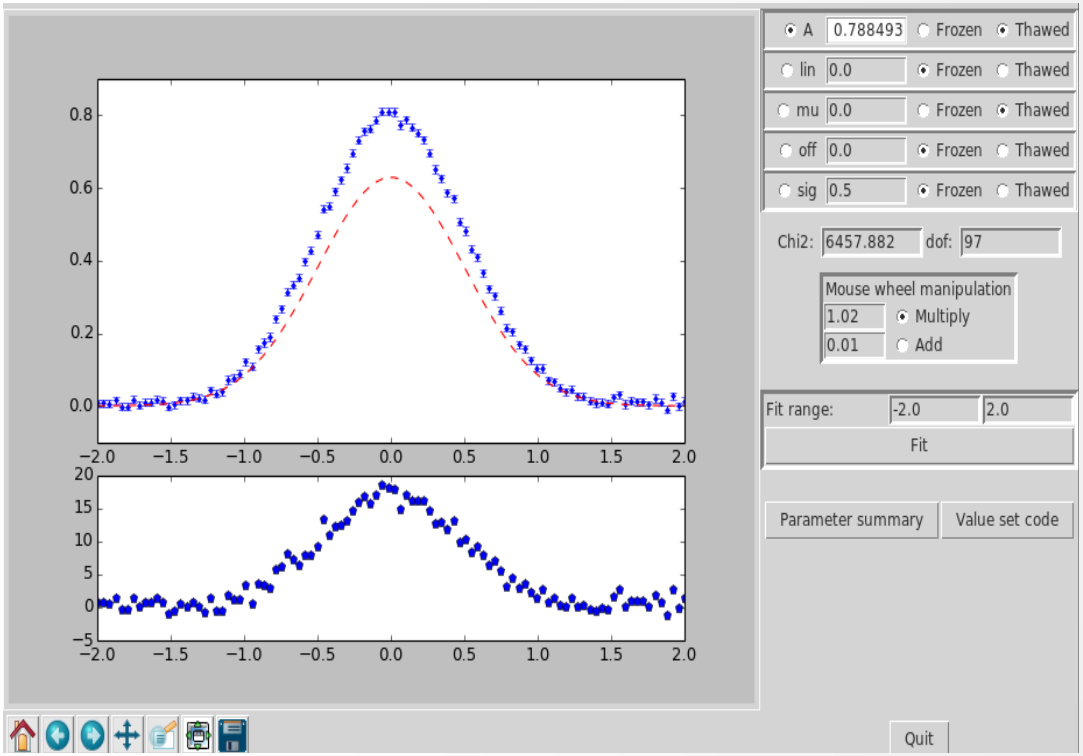


GUI packages:

Example—Manipulating a Gaussian

This example demonstrates the basic usage of the model explorer.

```
import numpy as np
from PyAstronomy import pyaGui
from PyAstronomy import funcFit as fuf

# Create a Gaussian fitting instance
# and set some parameters
gg = fuf.GaussFit1d()
gg["A"] = 1.0
gg["sig"] = 0.5

# Let A and mu be free during a fit
gg.thaw(["A", "mu"])

# Create some artificial data
x = np.linspace(-2., 2., 100)
yerr = np.ones(len(x))*0.01
y = gg.evaluate(x) + np.random.normal(0.,0.01, len(x))

# In order to use the interactive explorer, you
# need a class, which plots the model. The default
# class for this purpose is "FFModelPlotFit", which
# needs the x and y values. Optionally, you can specify
# errors via `yerr`. Depending on the setting for
# "withResiduals", the residuals will be shown or not.
mp = pyaGui.FFModelPlotFit(x, y, yerr=yerr, withResiduals=True)

# Use the function ffmodelExplorer (note the lowercase letters)
# to create an instance of the FFModelExplorer class, which
# needs to be given the model (gg in this case) and
# the plotter (and fitter), which we created above.
#
g = pyaGui.ffmodelExplorer(gg, mp)
g.show()
```



Conclusion/Comments:

- Mix of functions suitable to wide range of astronomy tasks.
- Not a complete tool set.
- Has good example code and documentation
- Take a look there may be something for you.