Statistical Tools in Python

A NASA AISR Project

Tom Loredo, Alanna Connors, Travis Oliphant

Cornell/Eureka Scientific/BYU
Motivation

Many advanced methods are conceptually simple but computationally difficult.

Competing methods of very different levels of sophistication are often similar from end-user’s perspective.

Principal obstacle to understanding/use is the art of statistical computing.

Eliminate this obstacle!
Main Features

- A deep and broad tool set
  - Tools tailored to astronomer-specific needs:
    Poisson processes, truncated power-law distributions, spherical distributions, upper limits, hierarchical methods . . .
  - Multiple methods in each problem class, esp. frequentist/Bayes

- Use of a modern VHL computer language: Python
  - Single implementation facilitates depth/breadth
  - Python’s VHL features speed development
  - Python’s simplicity allows easy access

- Outreach
A Bit About Python

- Very simple syntax—resembles “pseudo code”
- Use interactively, or via scripts/modules
- Object oriented—high level interfaces, modularity
- A general purpose language with rich standard library
- Sophisticated and fast numerical capability via NumPy/Numeric
- Easily extendible/embeddable with C/C++/Fortran
- Open source, cross-platform, active & growing user community
Scientific Computing With Python

- Numeric package (efficient array numerics)
  - Developed by LLNL/MIT scientists & programmers
  - Inspired by Matlab/IDL/Fortran90
  - Successor in development by NASA/STScI (numarray)

- SciPy package
  - High level interfaces to large, popular libraries:
    special functions, linear algebra, FFTs, DSP, quadrature, ODE solvers, optimizers, stats
  - Inline C via weave package

- Plotting
  - Interfaces to very many popular libraries (but no std.)
  - New package in development by NASA/STScI (Chaco)
NASA Support of Python

- PyRAF — Data analysis environment (STScI)
- numarray — Successor to Numeric (STScI)
- Chaco — Cross-platform, publication-quality 2-D plotting (STScI)
- Statistical Tools (Cornell/Eureka/BYU)
Simple Example: The Rayleigh Statistic

Search for periodic signals in arrival time series, \( \{t_i\} \).

\[
R(\omega) = \frac{1}{N} \left[ \left( \sum_i \sin \omega t_i \right)^2 + \left( \sum_i \cos \omega t_i \right)^2 \right]
\]

Frequentist approach: Maximize over \( \omega \); \( R_{\text{max}} \sim \chi^2_2 \)

Bayesian approach: log-sinusoid rate model

\[
r(t) \propto \exp[\kappa \cos(\omega t + \phi)]
\]

Likelihood for frequency \( \mathcal{L}(\omega) \propto e^{\kappa R(\omega)} \)
Sample Source Code

Python source code

def Rayleigh (data, w):
    wd = w*data
    return (sum(sin(wd))**2 +
            sum(cos(wd))**2)/len(data)

C source code

double Rayleigh (int n, double *data, double w) {
    double S, C, wt;
    int i;
    S = 0.;
    C = 0.;
    for (i=0; i<n; i++) {
        wt = w*data[i];
        S += sin(wt);
        C += cos(wt);
    }
    return (S*S + C*C)/n;
}
Proposed Components

Methods for data from sampled functions, \( d_i = f(t_i) + e_i \)

- Basic statistics (build on SciPy)
- Errors-in-variables models (EVM)
- Detection/measurement of periodic signals: Schuster periodogram, Lomb-Scargle, Bretthorst algorithm, piecewise-constant modeling
- Nonperiodic time series analysis (QPOs, \( 1/f \) noise): ARMA models, long-memory processes
- Robust estimation/outlier detection
Methods for discrete data:

- Counting processes (confidence/credible regions, backgrounds)
- Periodic point processes: Rayleigh statistic, $Z^2_N$, log-Fourier models, G-L method
- Inhomogeneous point processes: Bayes blocks, Poisson wavelets
- Population analyses: Survival analysis (ASURV), point process $+$ noise
- Nonparametric methods: PiCA KDE algorithm, mixture models
Parametric Inference Engine

- Constrained optimization
- Automated evaluation on grids, with refinement
- Projection (optimization on parameter subset)
- Simple parameter transformations
- Hessian calculation
- Multidimensional integration
- Tools for creating & analyzing Markov chains
We are very open to advice/suggestions/requests!