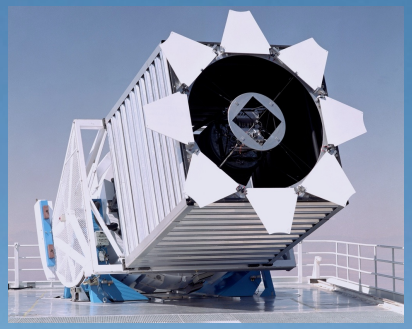
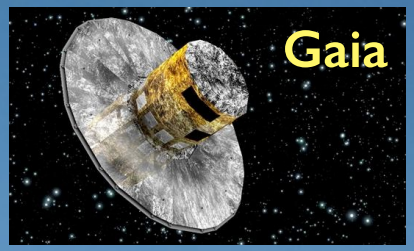
Lecture 1: Introduction and warm-up with astroML

Željko Ivezić, University of Washington

LSST



SDSS



Lectures:

1) Introduction and warm-up with astroML

2) Density Estimation and
   Dimensionality Reduction
   (and perhaps a bit of Regression)

**The main goal: to make you love
astroML (and Jake!)**

Outline

- Motivation
  – ever increasing data volume and complexity
  – sophisticated analysis, need for reproducability
  – open-source approach (also) improves efficiency
- astroML
  - what is it?
  - how to get it?
- Examples of astroML use:
  - 4-D visualization
  - robust parameters for a 2-dimensional gaussian
  - Bayesian parameter estimation for uniform distr.
  - correcting for truncation bias (C minus method)
  - matched filter analysis of a low-SNR signal
  - (an example of wavelet analysis)

Motivation

- **Ever increasing data volume and complexity**
  - SDSS is ~30 TB; LSST will be one SDSS per night, or a total of >100 PB of data (20 billion objects); of course, also Gaia and many other surveys
  - who and how will do the required data analysis?

- **Sophisticated analysis, need for reproducability**
  - with the increasing data complexity, analysis becomes more complex, too; what do we do in case of disagreement?

- **Open-source approach improves efficiency**
  - we are not data starved any more!
  - the bottleneck for new results is in human resources (as in "grad students and postdocs") and analysis tools
  - nobody has an unlimited budget; we are a small field and we should collaborate and share!

Data analysis challenges in the era of Big Data

1) Large data volume

2) Large number of objects

3) Highly multi-dimensional space

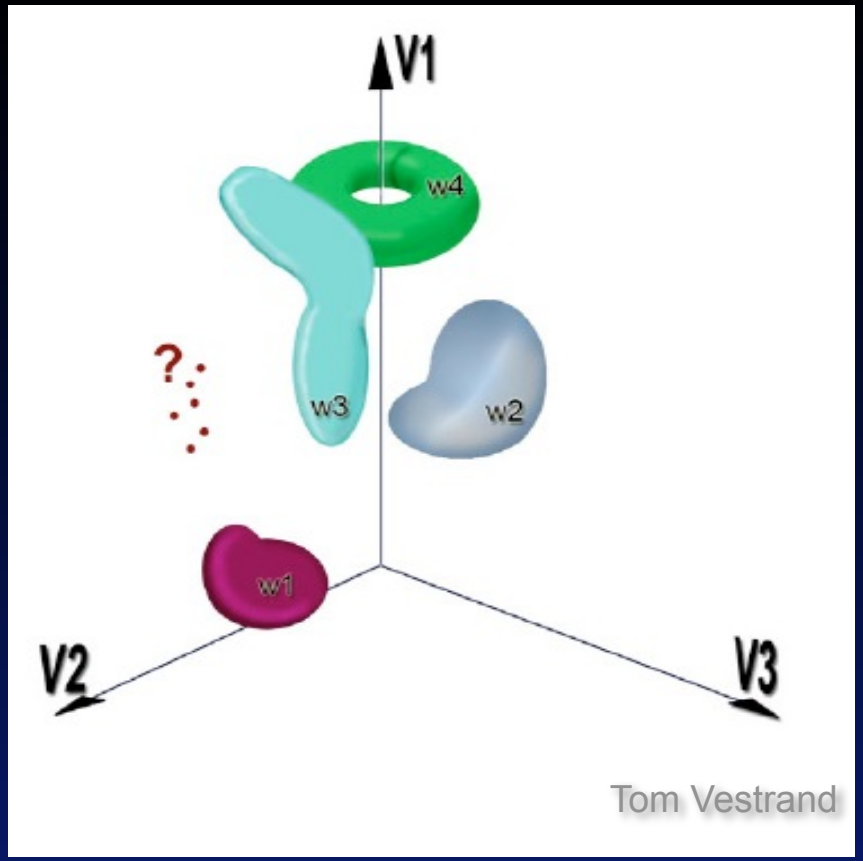
4) Unknown statistical distributions

5) Time-series data

6) Truncated, censored and missing data

7) Unreliable quantities (e.g. unknown systematics and random errors)

- Characterize the known (clustering)
- Assign the new (classification)
- Discover the unknown (outlier detection)



Benefits of very large data sets:

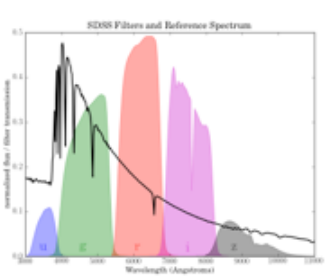
- best statistical analysis of "typical" events
- automated search for "rare" events

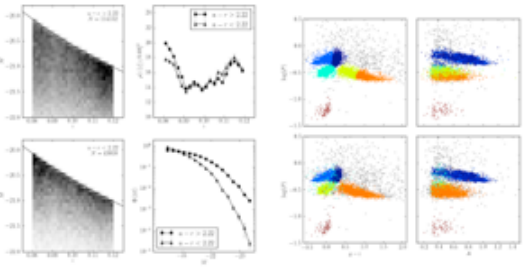
How do we efficiently train astronomy students to use sophisticated methods from statistics, data mining and machine learning?

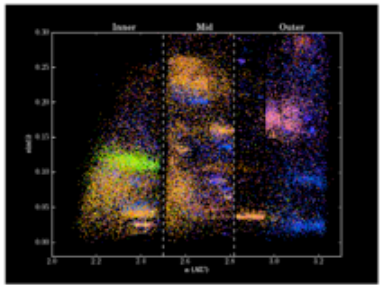
astroML | Home | User Guide | Book Figures | Examples Plots

Google™ Custom Search

AstroML: Machine Learning and Data Mining for Astronomy







News

October 2012: astroML 0.1 has been released! Get the source on Github

Our Introduction to astroML paper received the CIDU 2012 best paper award.

Links

astroML Mailing List

GitHub Issue Tracker

Videos

Scipy 2012 (15 minute talk)

Citing

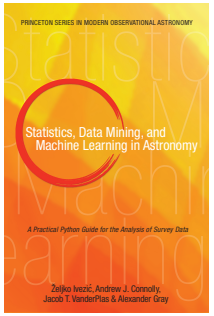
If you use the software, please consider citing astroML.

AstroML is a Python module for machine learning and data mining built on numpy, scipy, scikit-learn, and matplotlib, and distributed under the 3-clause BSD license. It contains a growing library of statistical and machine learning routines for analyzing astronomical data in python, loaders for several open astronomical datasets, and a large suite of examples of analyzing and visualizing astronomical datasets.

Downloads

- Released Versions: Python Package Index
- Bleeding-edge Source: github

The goal of astroML is to provide a community repository for fast Python implementations of common tools and routines used for statistical data analysis in astronomy and astrophysics, to provide a uniform and easy-to-use interface to freely available astronomical datasets. We hope this package will be useful to researchers and students of astronomy. The astroML project was started in 2012 to accompany the book **Statistics, Data Mining, and Machine Learning in Astronomy** by Zeljko Ivezic, Andrew Connolly, Jacob VanderPlas, and Alex Gray, to be published in late 2013. The table of contents is available here: **`here(pdf)`**.



User Guide

1. Introduction

- 1.1. Philosophy

Lead developer: Jacob Vanderplas

Open source! www.astroML.org

astroML Home User Guide Book Figures Examples Plots

Google™ Custom Search

AstroML: Machine Learning and Data Mining for Astronomy

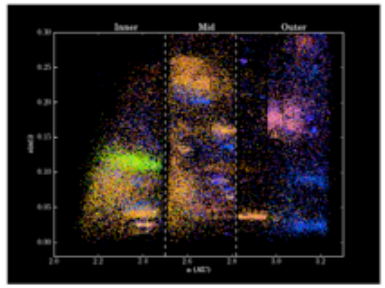
News

October 2012: astroML 0.1

The Vision:

- Maintain a coherent & well-written set of examples of data processing in python *using real data.*
- Offer a standard repository of high-performance python code for astronomy.
- Complement the effort of the AstroPy team.
- Let the cream rise to the top: useful code – once battle-tested – can be moved upstream and become useful in other fields:
  - Ball Tree & two-point statistics (scikit-learn 0.10)
  - Minimum Spanning Tree (scipy 0.12)
  - binned_statistics (scipy 0.12)

Big Goal: community involvement! astroML should be much more than simply a textbook software package. It should become a community repository.



Downloads

- Released Versions: Python Package Index
- Bleeding-edge Source: github

on implementations of common tools
hysics, to provide a uniform and easy-
s package will be useful to researchers
o accompany the book **Statistics,**
Andrew Connolly, Jacob VanderPlas,
vailable here: **here(pdf)**.

Open source!
www.astroML.org

1. Introduction
   - 1.1. Philosophy

Textbook Figures

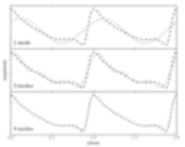
This section makes available the source code used to generate every figure in the book *Statistics, Data Mining, and Machine Learning in Astronomy*. Many of the figures are fairly self-explanatory, though some will be less so without the book as a reference. The table of contents of the book can be seen **here(pdf)**.

Figure Contents

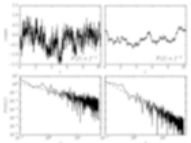
Each chapter links to a page with thumbnails of the figures from the chapter.

Chapter 10: Time Series Analysis

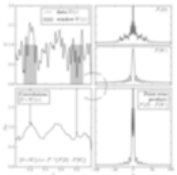
This chapter covers the analysis of both periodic and non-periodic time series, for both regularly and irregularly spaced data.



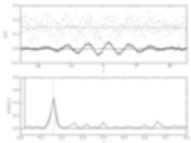
Fourier Reconstruction of RR-Lyrae Templates



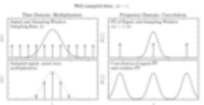
Generating Power-law Light Curves



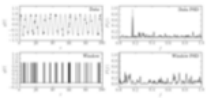
Plot a Diagram explaining a Convolution



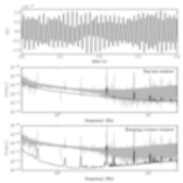
Fast Fourier Transform Example



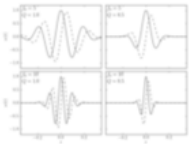
The effect of Sampling



The effect of Sampling



Plot the power spectrum of the LIGO big dog event



Examples of Wavelets

AstroML: let's start!

1) Do we all have it installed?

Go to:
http://www.astroml.org/user_guide/installation.html

2) Did we all test the installation?

Run:
http://www.astroml.org/book_figures/chapter1/fig_SSPP_metallicity.html

3) Did we all download data files?

Run:
http://www.astro.washington.edu/users/ivezic/astroML/AstroHackWeek2014/fetchAllData.py

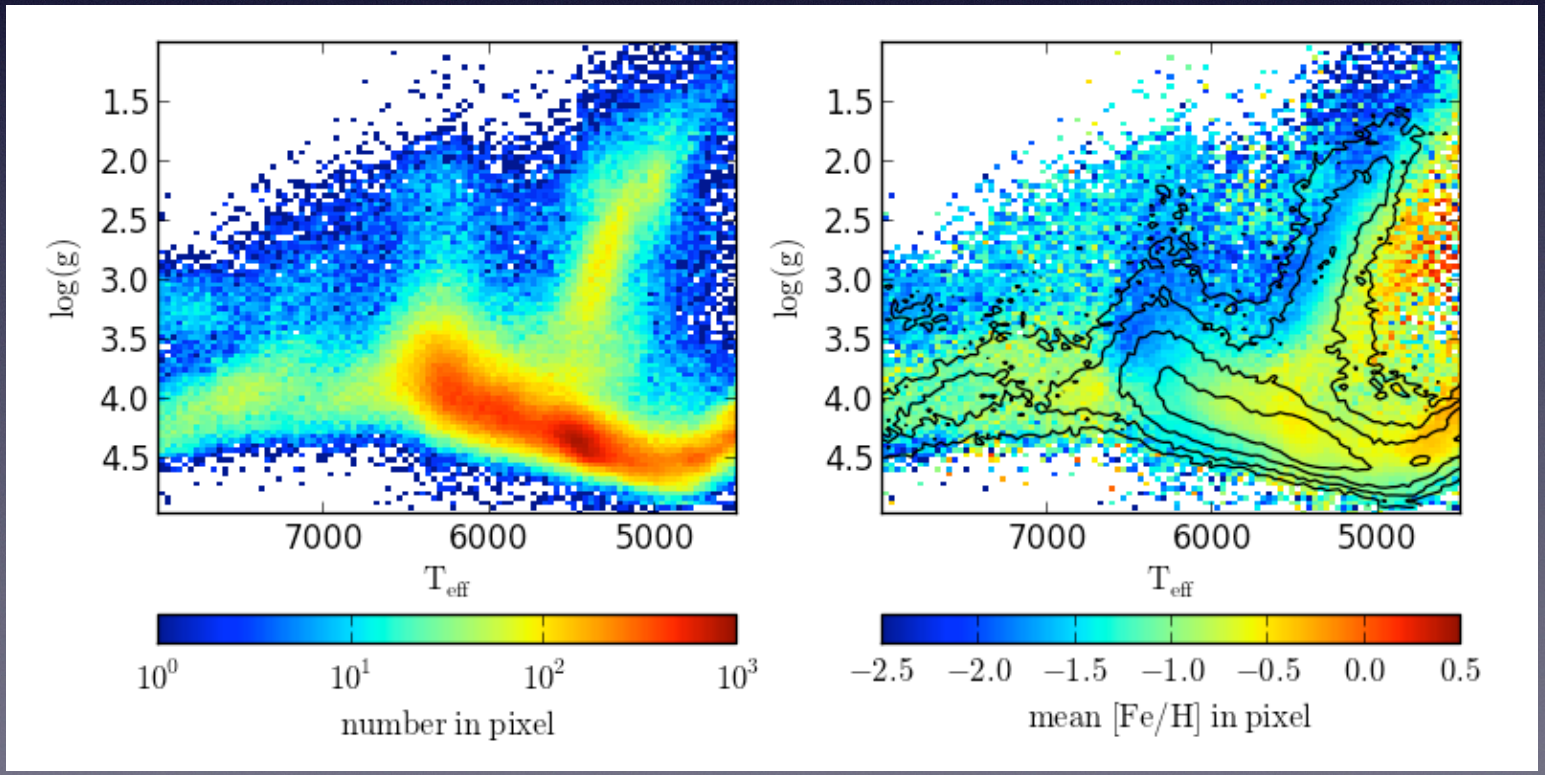
Or copy directory:
http://www.astro.washington.edu/users/ivezic/astroML/AstroHackWeek2014/astroML_data

4) Did we all download *.py files?

Download:
http://www.astro.washington.edu/users/ivezic/astroML/AstroHackWeek2014/pythonAll.tar.gz

- Examples of astroML use:
  - a Hess diagram coded by a third quantity
  - 4-dimensional visualization
  - robust 2D gaussian parameters
  - Bayesian parameter estimation for uniform distr.
  - correcting for truncation bias (C minus method)
  - matched filter analysis of a low-SNR signal
  - (an example of wavelet analysis)

- **A Hess diagram coded by a third quantity**
- Hess diagram is astronomical term for pixelated color-magnitude diagram, where each pixel is coded to display the number of objects in it (also know as “two-dimensional histogram”)
- Of course, the pixels don’t have to be coded by the number of objects in it - we can use instead any statistic



- **A Hess diagram coded by a third quantity**
- Hess diagram is astronomical term for pixelated color-magnitude diagram, where each pixel is coded to display the number of objects in it (also know as “two-dimensional histogram”)
- Of course, the pixels don’t have to be coded by the number of objects in it - we can use instead any statistic

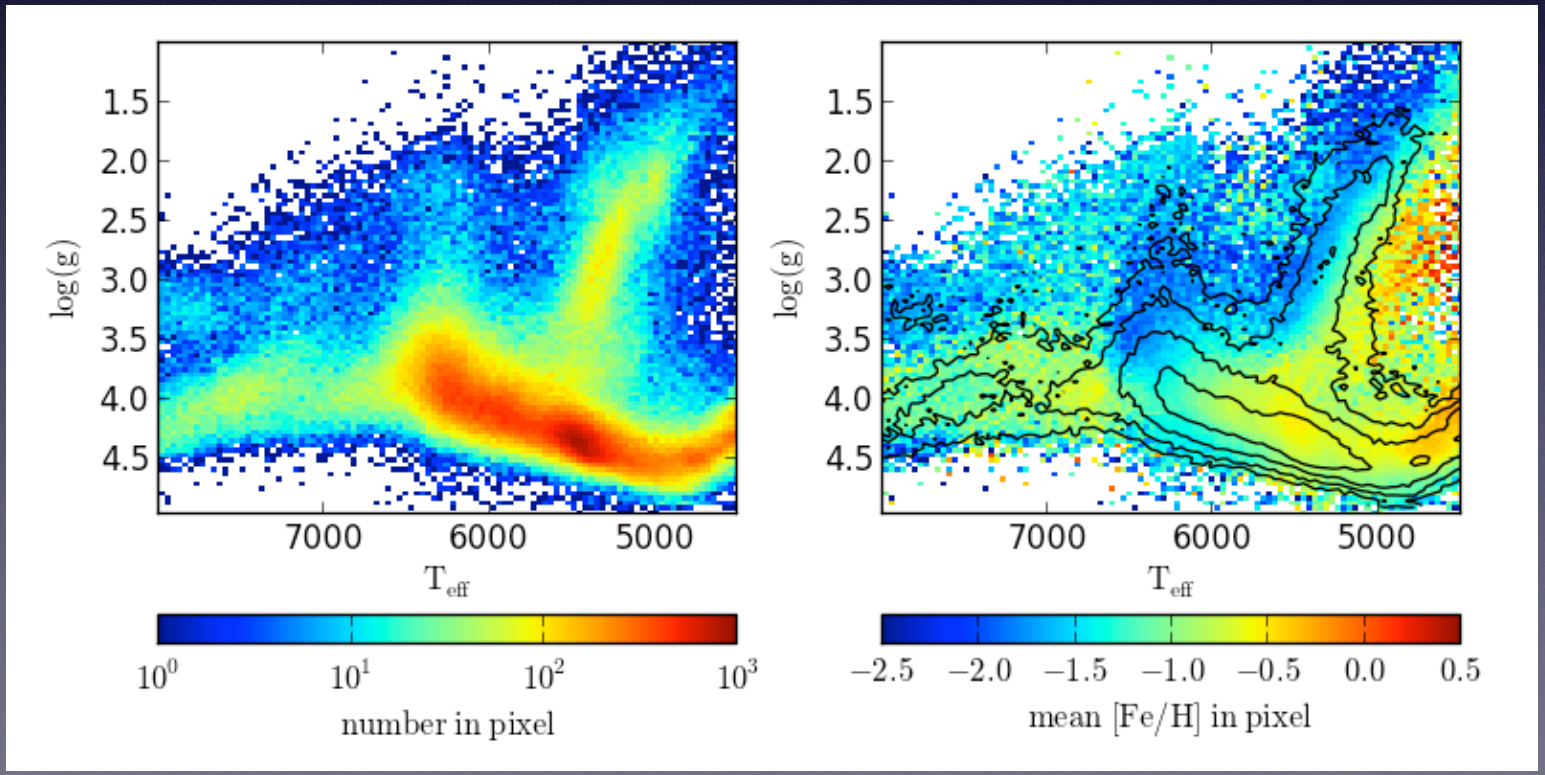
```
#------------------------------------------------------------
# Plot the results using the binned_statistic function
from astroML.stats import binned_statistic_2d
N, xedges, yedges = binned_statistic_2d(Teff, logg, FeH,
                                        'count', bins=100)

FeH_mean, xedges, yedges = binned_statistic_2d(Teff, logg, FeH,
                                               'mean', bins=100)
```

- $astroMLdir/astroML/stats/_binned_statistic.py

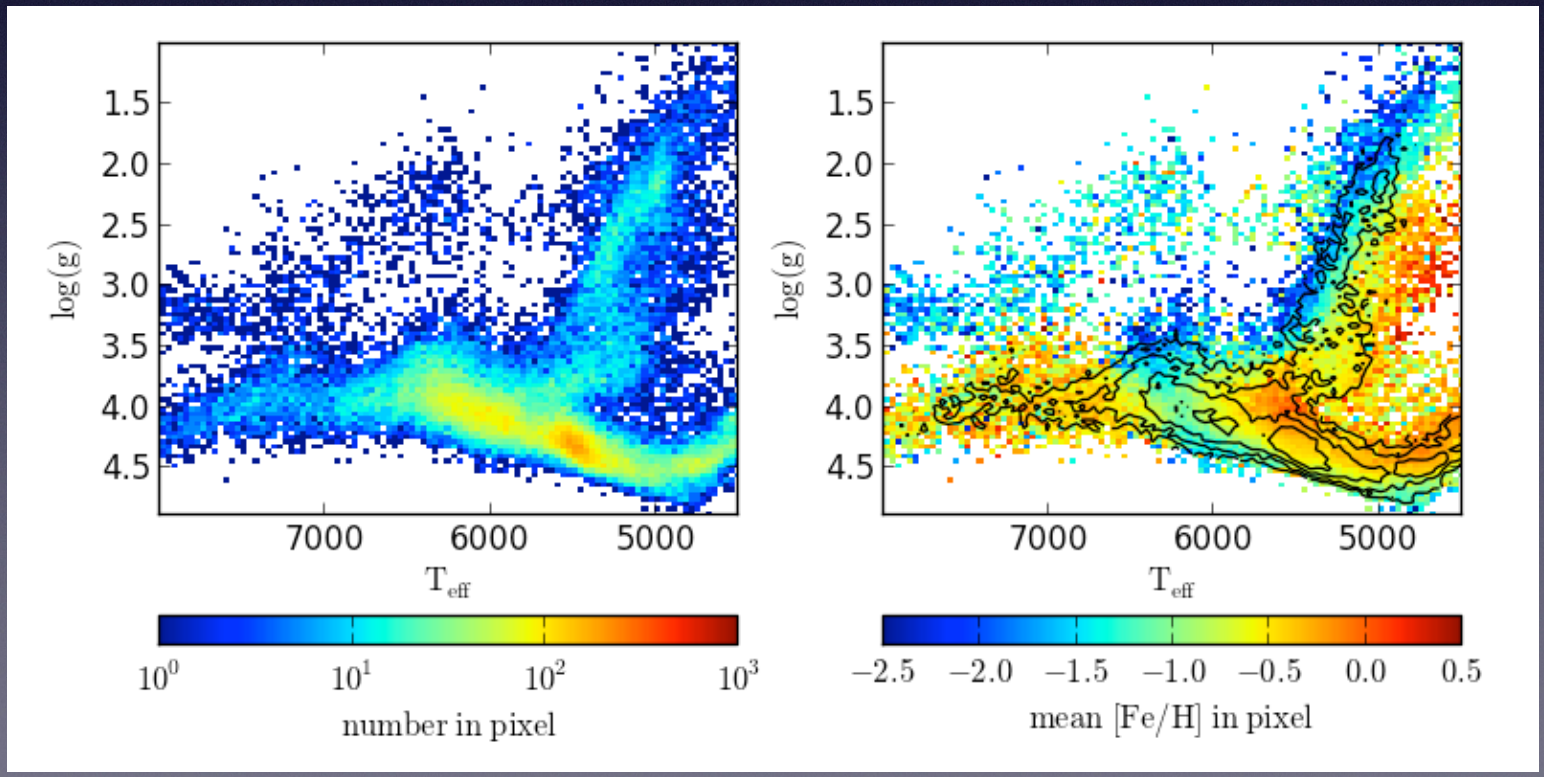
- $astroMLdir/astroML/stats/_binned_statistic.py

```
def binned_statistic(x, values, statistic='mean',
                     bins=10, range=None):
    """
    Compute a binned statistic for a set of data.

    This is a generalization of a histogram function.  A histogram divides
    the space into bins, and returns the count of the number of points in
    each bin.  This function allows the computation of the sum, mean, median,
    or other statistic of the values within each bin.

    Parameters
    ----------
    x : array_like
        A sequence of values to be binned.
    values : array_like
        The values on which the statistic will be computed.  This must be
        the same shape as x.
    statistic : string or callable, optional
        The statistic to compute (default is 'mean').
        The following statistics are available:

          * 'mean' : compute the mean of values for points within each bin.
            Empty bins will be represented by NaN.
          * 'median' : compute the median of values for points within each
            bin. Empty bins will be represented by NaN.
          * 'count' : compute the count of points within each bin.  This is
            identical to an unweighted histogram.  `values` array is not
            referenced.
          * 'sum' : compute the sum of values for points within each bin.
            This is identical to a weighted histogram.
          * function : a user-defined function which takes a 1D array of
            values, and outputs a single numerical statistic. This function
            will be called on the values in each bin.  Empty bins will be
            represented by function([]), or NaN if this returns an error.

    bins : int or sequence of scalars, optional
        If `bins` is an int, it defines the number of equal-width
```

- **A Hess diagram coded by a third quantity**
- make this plot by running
  %run plot_SDSS_SSPP.py
- now, let's look at the code and change something, for example, let's only select stars with 15 < rpsf < 16
  find line 23: data = data[(rpsf > 15) & (rpsf < 19)]
  and change 19 to 16, save and then rerun...

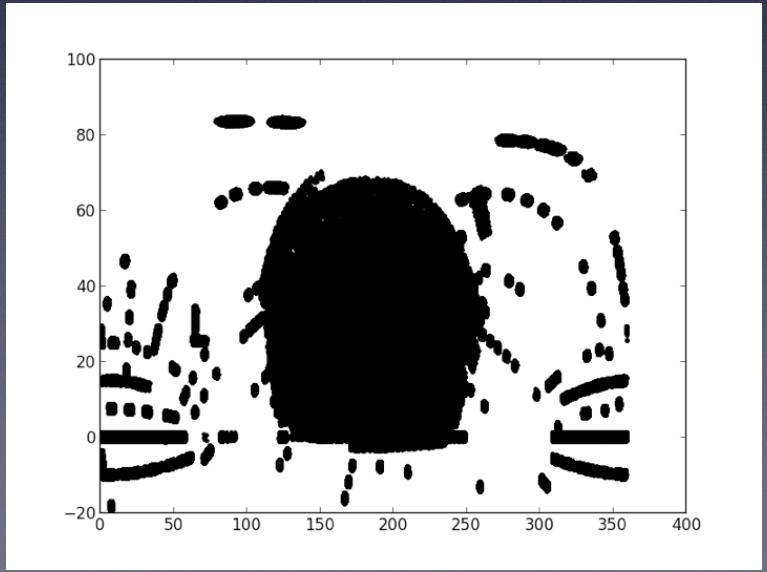


- **A Hess diagram coded by a third quantity**
- make this plot by running
  %run plot_SDSS_SSPP.py
- now, let's look at the code and change something, for example, let's only select stars with 15 < rpsf < 16



- **A Hess diagram coded by a third quantity**
- how did we get these (SDSS spectroscopic) data?
  data = fetch_sdss_sspp()
- this code lives in $astroMLdir/astroML/datasets/
  sdss_sspp.py fitsdata = download_with_progress_bar(DATA_URL)
- if you didn't have data file locally, it would have fetched it
- there are many other parameters, e.g.

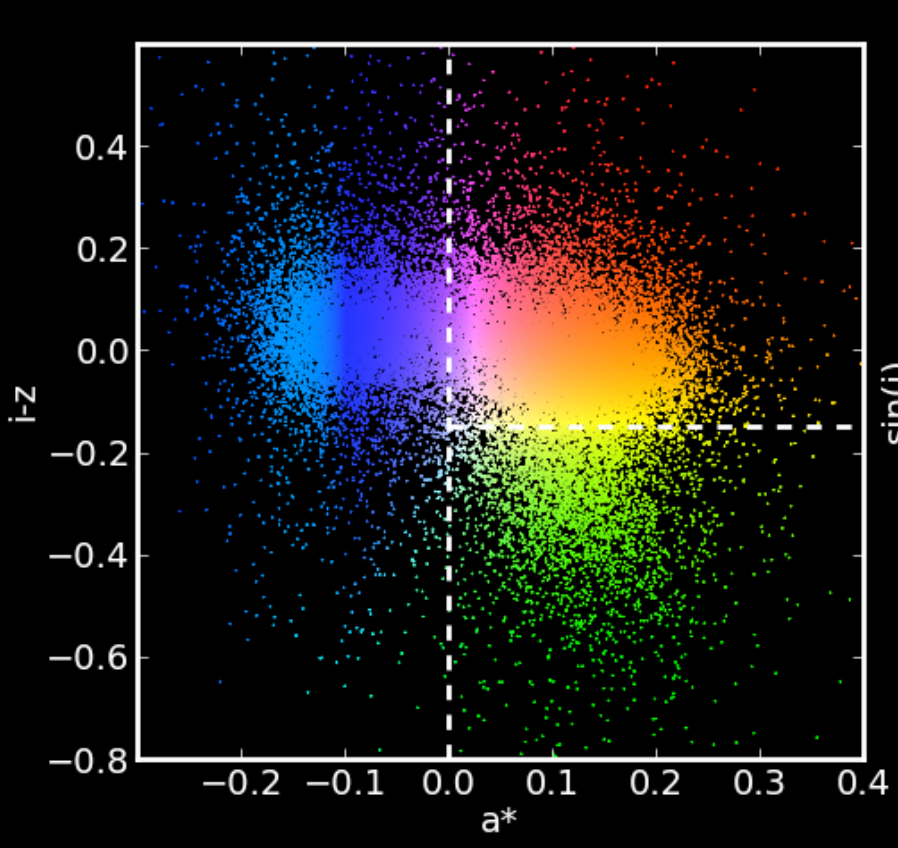
```
ra = data['ra']
dec = data['dec']
ax = plt.axes()
ax.plot(ra, dec, '.k')
plt.show()
```

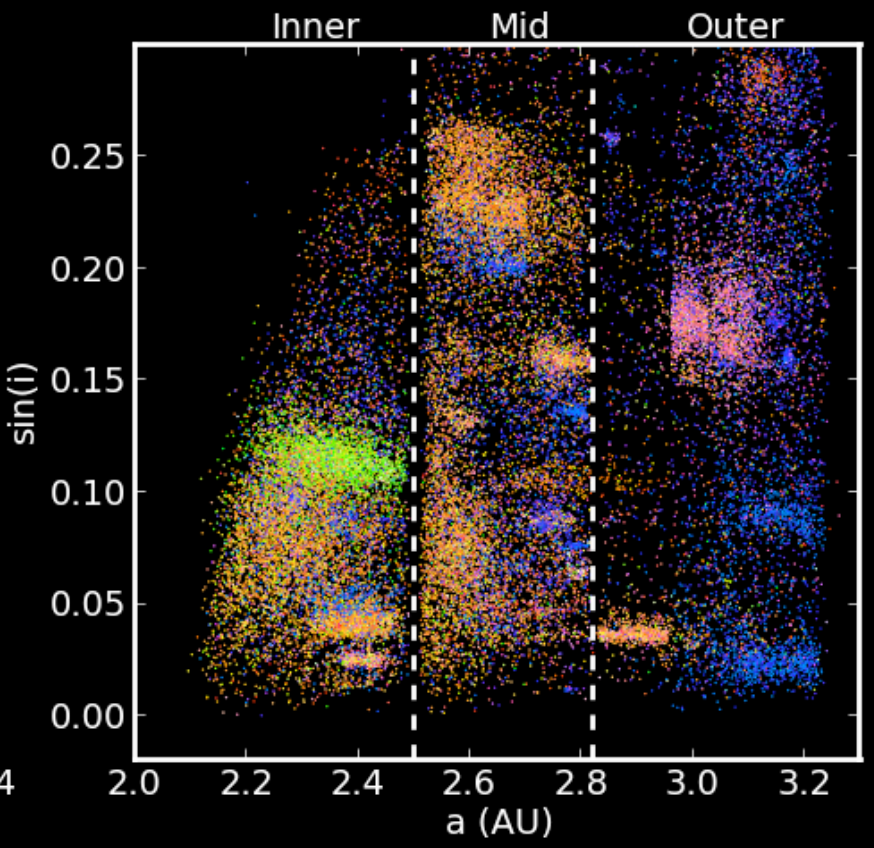


Ugly plot!

You can make this plot from scratch in <1 hour!

Visualization of 4-dimensional correlations





Our goal here is to discuss in detail a few more complex examples.

- **Robust parameters for a 2-dimensional gaussian**
- in one dimensional case, when we have outliers, we can still often robustly estimate location parameter by using the median instead of the mean (this works even in case of the Cauchy distribution!)
- what do we do in 2-dimensional case?

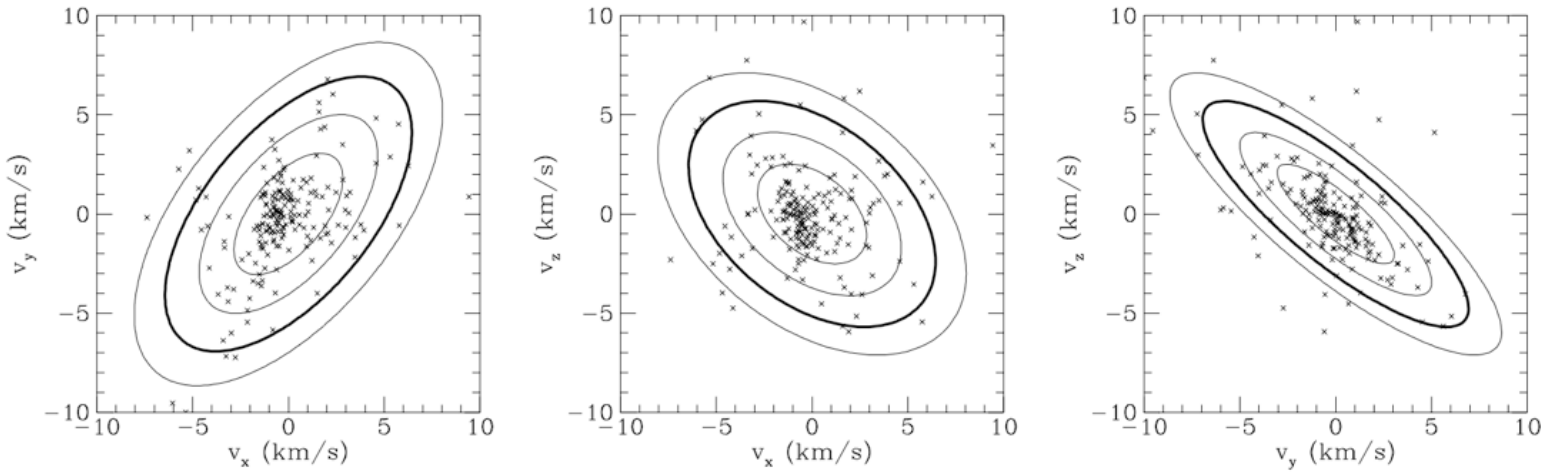


Fig. 16. Contour diagrams delineating the (projected) confidence regions corresponding to the mean covariance matrix of the space velocities for all candidate members with available radial velocities (cf. Fig. 8b). The confidence regions indicate the expected distributions of residual velocities in the absence of intrinsic dispersion. The contours correspond to the 68.3%, 95.4%, 99.73%, and 99.99% confidence levels. The thick line, at 99.73%, is the $c = 14.16$ contour (see Eq. 16). The crosses correspond to the residuals between the observed space motions, and the mean (centre of mass) cluster motion.

Perryman et al. (1998, A&A 331, 81)

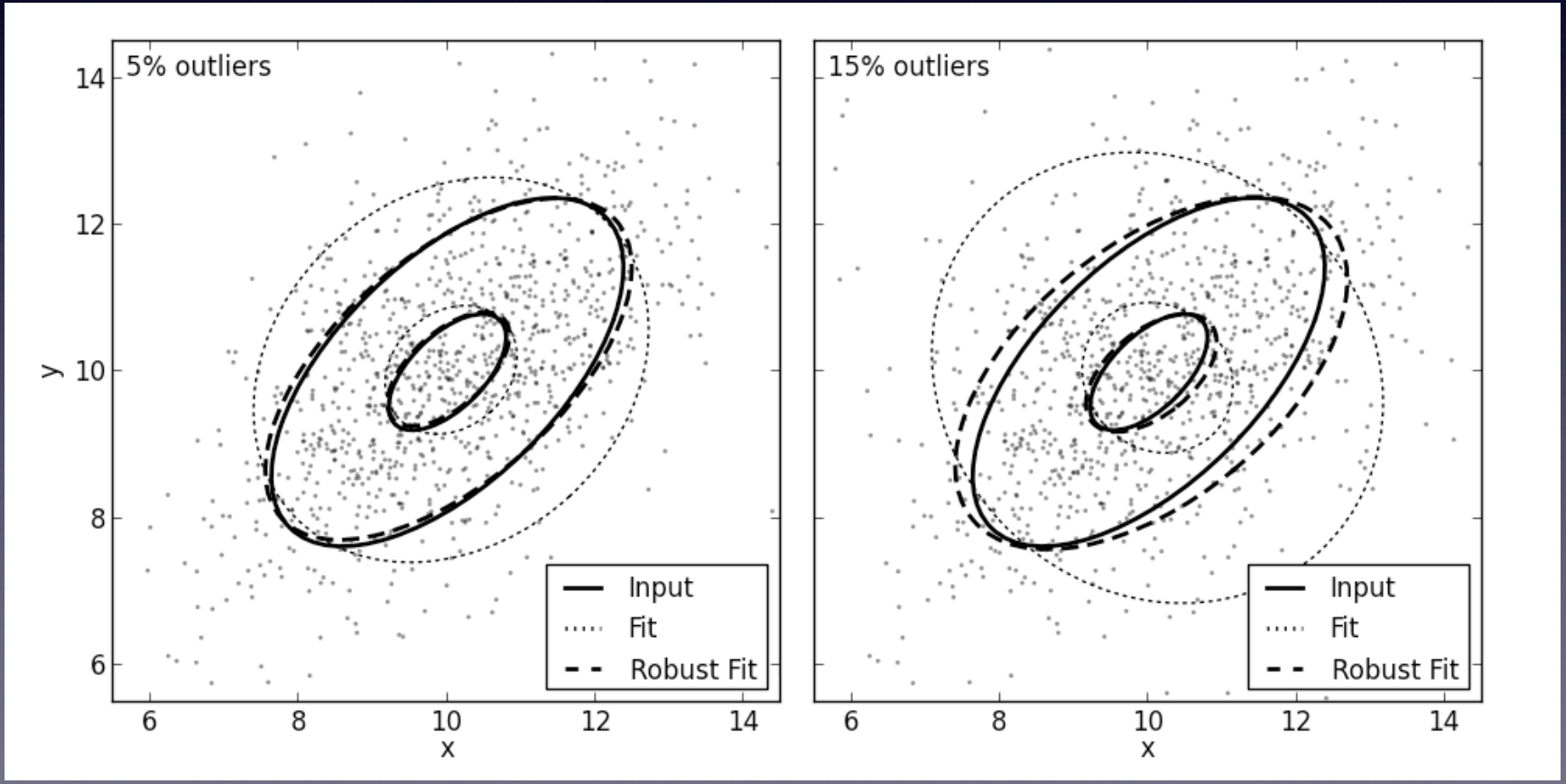
- **Robust parameters for a 2-dimensional gaussian**
- in one dimensional case, when we have outliers, we can still often robustly estimate location parameter by using the median instead of the mean (this works even in case of the Cauchy distribution!)
- what do we do in 2-dimensional case?
- similarly to using the median (q50) instead of the mean, we can use s_G = 0.7413 (q75 - q25) instead of std. deviation (the price we pay for robustness is about 25% larger error for the median than for the mean, and about 50% larger error for s_G than for st.dev; for nearly-gaussian distributions).

$$\tan(2\alpha) = 2\,\rho\,\frac{\sigma_x \sigma_y}{\sigma_x^2 - \sigma_y^2}$$

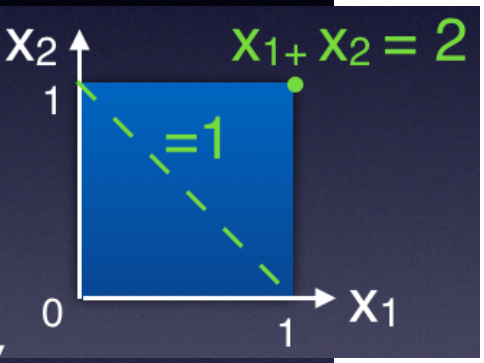
$$\rho = \frac{V_u - V_w}{V_u + V_w},$$

u and w are non-covariant linear combinations of x and y

- **Robust parameters for a 2-dimensional gaussian**
- what do we do in 2-dimensional case?
- make this plot by running
  %run fig_robust_pca.py
- as easy as fit_bivariate_normal(x, y, robust=True)



- **Bayesian parameter estimation for uniform distribution**
- CLT:



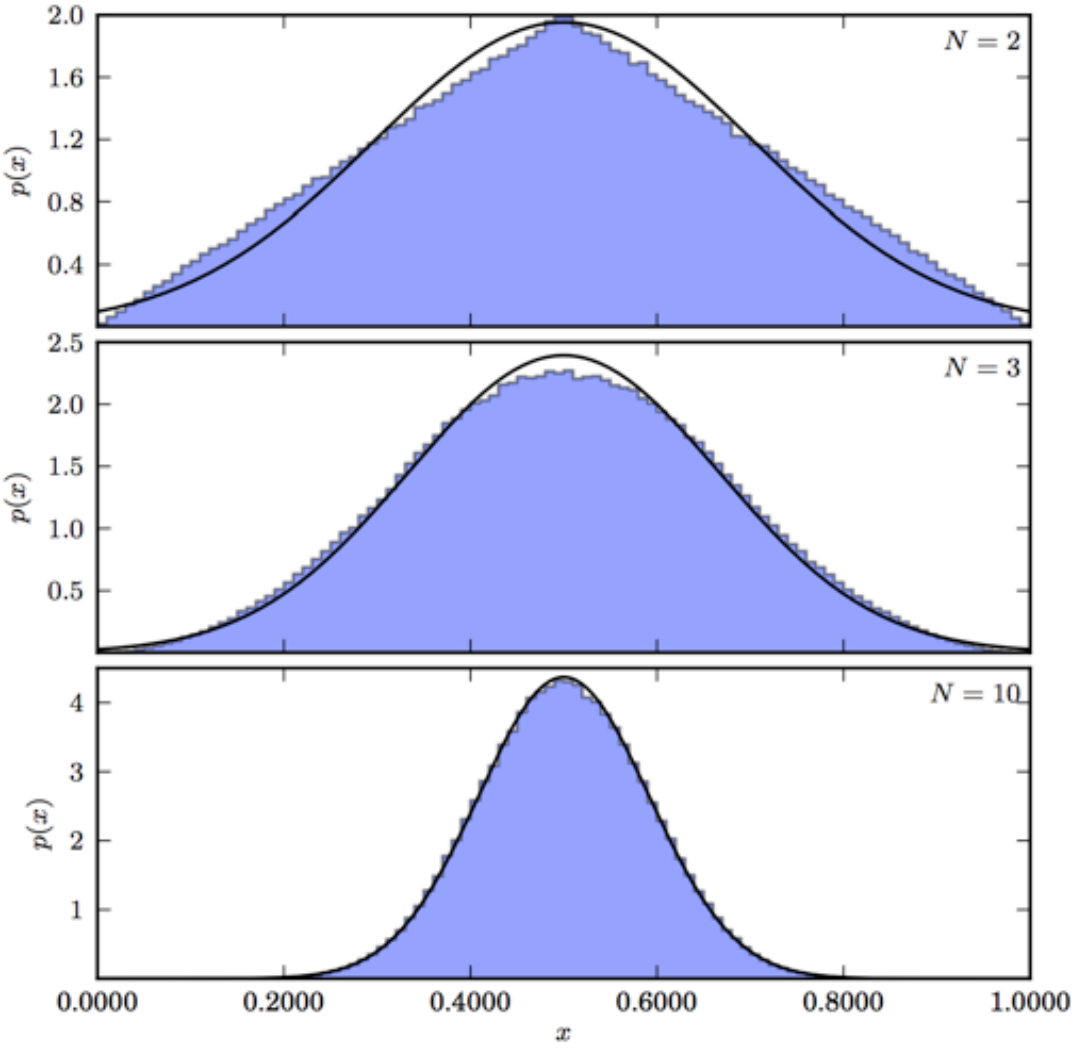


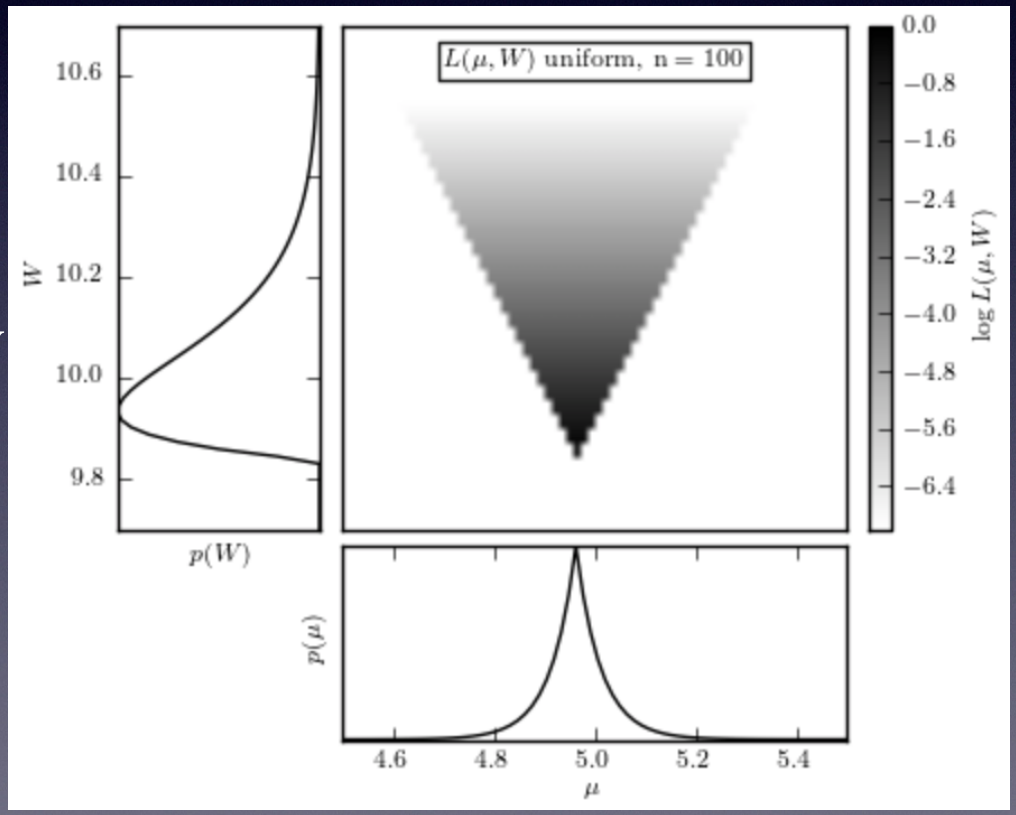
Figure 3.20.: An illustration of the central limit theorem. The histogram in each panel shows the distribution of the mean value of N random variables drawn from the $(0, 1)$ range (a uniform distribution with $\mu = 0.5$ and $W = 1$; see eq. 3.39). The distribution for $N = 2$ has a triangular shape and as N increases it becomes increasingly similar to a Gaussian, in agreement with the central limit theorem. The predicted normal distribution with $\mu = 0.5$ and $\sigma = 1/\sqrt{12N}$ is shown by the line. Already for $N = 10$, the "observed" distribution is essentially the same as the predicted distribution.

- **Bayesian parameter estimation for uniform distribution**
- CLT: for distributions with finite variance, the mean is an unbiased estimator of the location parameter with uncertainty that decreases as 1/sqrt(N)
- CLT is of course correct in case of uniform distribution. However, the mean is NOT the best (most efficient, that is, with the smallest uncertainty, to be statistically precise) estimator of the location parameter in this case.
- Due to the absence of tails, the mean of the two extreme values (min and max) is a much more efficient estimator than the sample mean, with an uncertainty that decreases as 1/N.

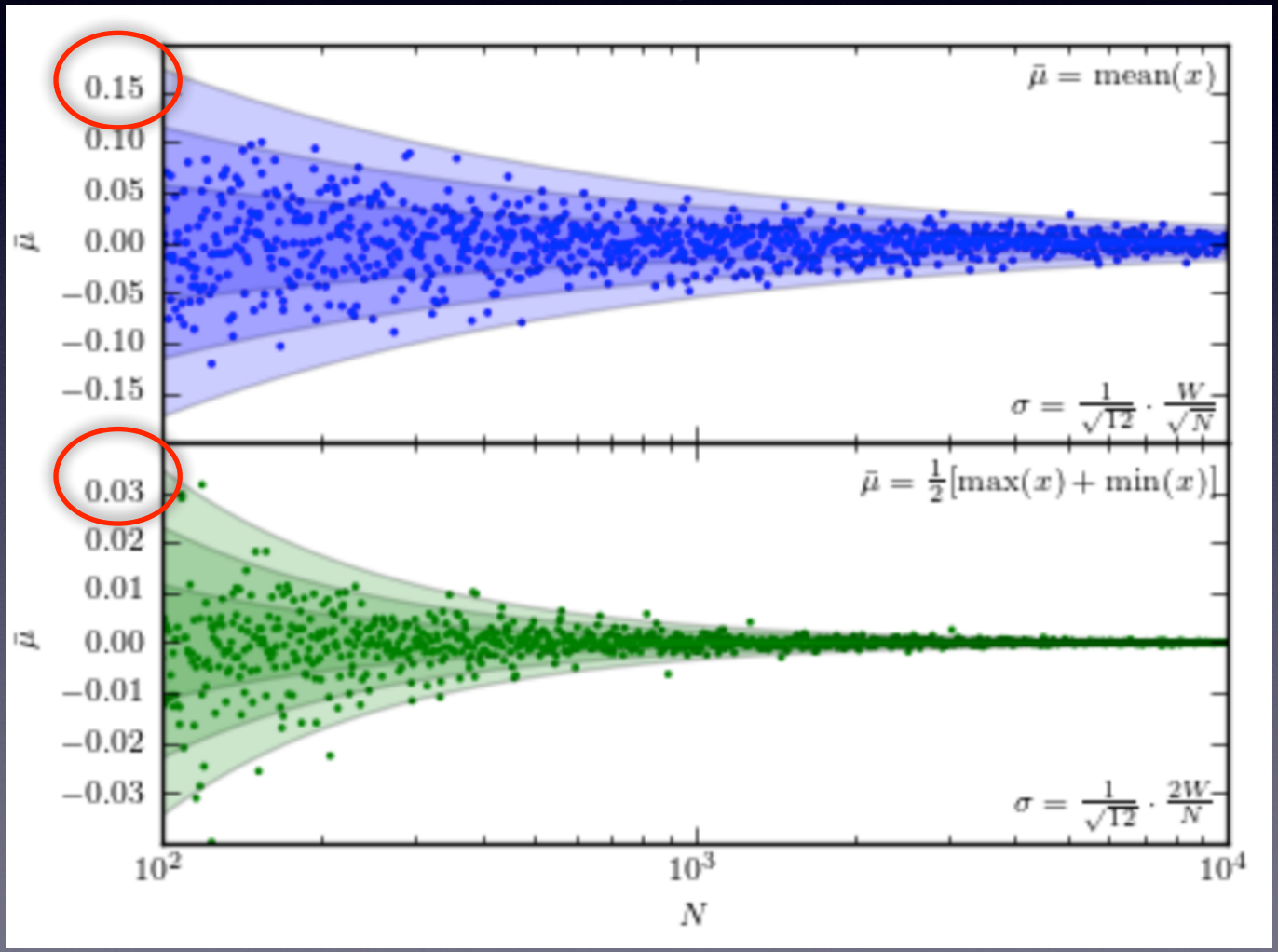
$$p(x_i|\mu, W) = \frac{1}{W} \text{ for } |x_i - \mu| \leq \frac{W}{2}$$

Task: given measurements x_i, i=1…N, drawn from the Uniform distribution, find the best estimate of μ, let's call it μ^0, and its uncertainty, σ_μ

In this case, using the mean value results in much larger uncertainty than proper Bayesian treatment!

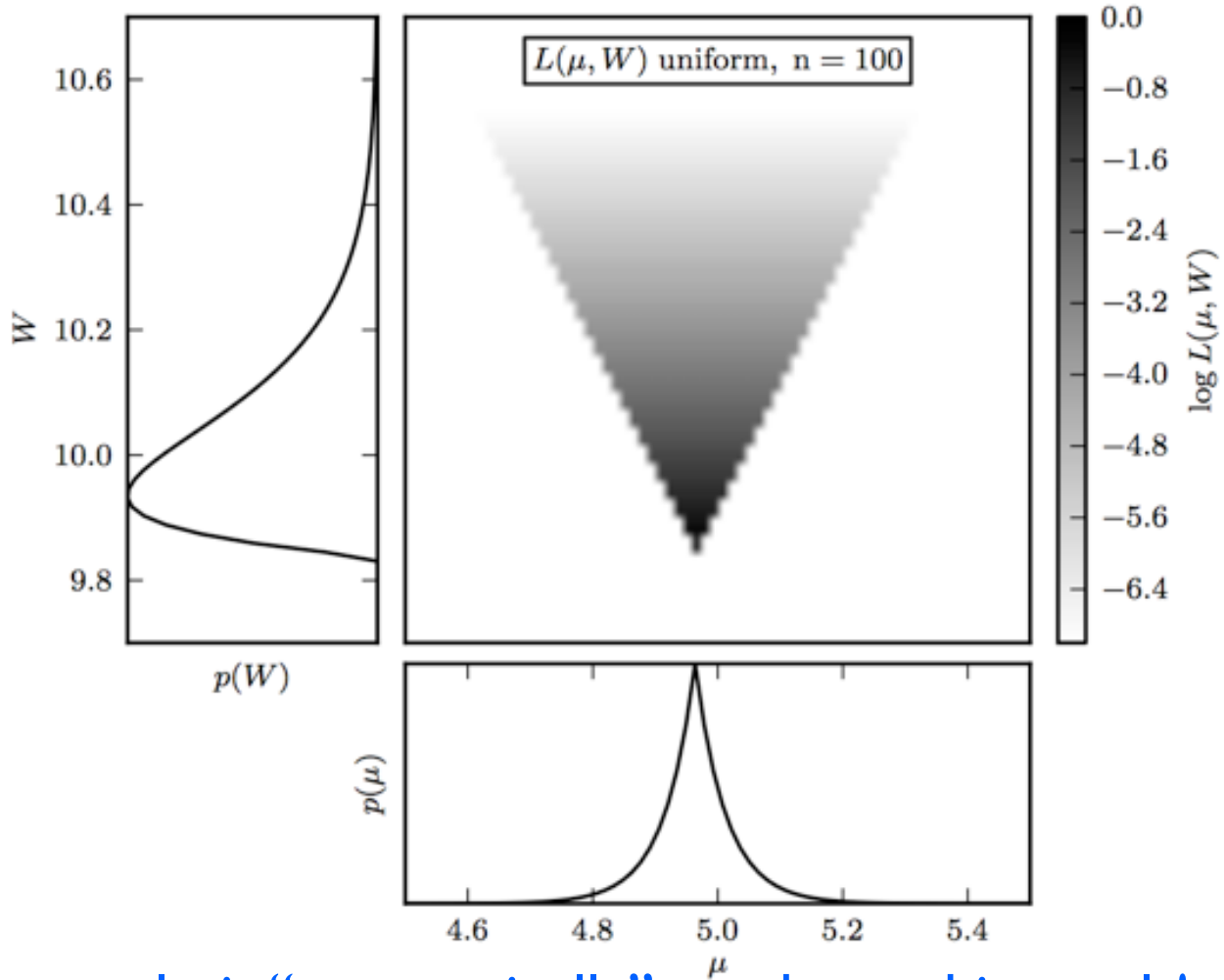


- **Bayesian parameter estimation for uniform distribution**
- The mean of the two extreme values “beats” the sample mean, with an uncertainty that decreases as 1/N.
- To demonstrate this, let’s run fig_uniform_mean.py



- **Bayesian parameter estimation for uniform distribution**

This is http://www.astroml.org/book_figures/chapter5/fig_likelihood_uniform.html



Bayesian analysis "automatically" produces this result! (next page)

Figure 5.12.: An illustration of the logarithm of the posterior probability distribution $L(\mu, W)$ (see eq. 5.77) for $N = 100$, $\mu = 5$, and $W = 10$. The maximum of L is renormalized to 0, and color coded on a scale from -5 to 0, as shown in the legend. The bottom panel shows the marginal posterior for μ (see eq. 5.79), and the left panel shows the marginal posterior for W (see eq. 5.80).

- **Bayesian parameter estimation for uniform distribution**

Given the uniform distribution described by eq. 3.39, the likelihood of observing x_i is given by

$$p(x_i|\mu, W) = \frac{1}{W} \text{ for } |x_i - \mu| \leq \frac{W}{2}, \tag{5.76}$$

and 0 otherwise (i.e., x_i can be at most $W/2$ away from μ). We assume that both μ and W are unknown and need to be estimated from data. We shall assume a uniform prior for μ between $-\Delta$ and Δ, where Δ greatly exceeds the plausible range for $|\mu + W|$, and a scale-invariant prior for W. The likelihood of observing the observed data set $\{x_i\}$ is

$$p(\mu, W|\{x_i\}, I) \propto \frac{1}{W^{N+1}} \prod_{i=1}^{N} g(\mu, W|x_i), \tag{5.77}$$

where

$$g(\mu, W|x_i) = 1 \text{ for } W \geq 2|x_i - \mu|, \tag{5.78}$$

and 0 otherwise. In the (μ, W) plane, the region allowed by x_i (where $p(\mu, W|\{x_i\}, I) > 0$) is the wedge defined by $W \geq 2(x_i - \mu)$ and $W \geq 2(\mu - x_i)$. For the whole data set, the allowed region becomes a wedge defined by $W \geq 2(x_{\rm max} - \mu)$ and $W \geq 2(\mu - x_{\rm min})$, where $x_{\rm min}$ and $x_{\rm max}$ are the minimum and maximum values in the data set $\{x_i\}$ (see figure 5.12). The minimum allowed W is $W_{\rm min} = (x_{\rm max} - x_{\rm min})$, and the posterior is symmetric with respect to $\tilde{\mu} = (x_{\rm min} + x_{\rm max})/2$. The highest value of the posterior probability is at the point $(\mu = \tilde{\mu}, W = W_{\rm min})$. Within the allowed range, the posterior is proportional to $1/W^{N+1}$ without a dependence on μ, and is 0 otherwise. The logarithm of the posterior probability $p(\mu, W|\{x_i\}, I)$ based on $N = 100$ values of x_i, generated using $\mu = 0$, $W = 1$ (for this particular realization, $x_{\rm min} = 0.047$ and $x_{\rm max} = 9.884$), is shown in figure 5.12. We see that in this case, the likelihood contours are not well described by ellipses!

It is straightforward to show that the marginal posterior pdf for μ is

$$p(\mu) = \int_0^\infty p(\mu, W|\{x_i\}, I)\, dW \propto \frac{1}{(|\mu - \tilde{\mu}| + W_{\rm min}/2)^N}, \tag{5.79}$$

$$p(x_i|\mu, W) = \frac{1}{W} \text{ for } |x_i - \mu| \leq \frac{W}{2}$$

Task: given measurements x_i, i=1…N, drawn from the Uniform distribution, find the best estimate of μ, let's call it μ^0, and its uncertainty, σ_μ

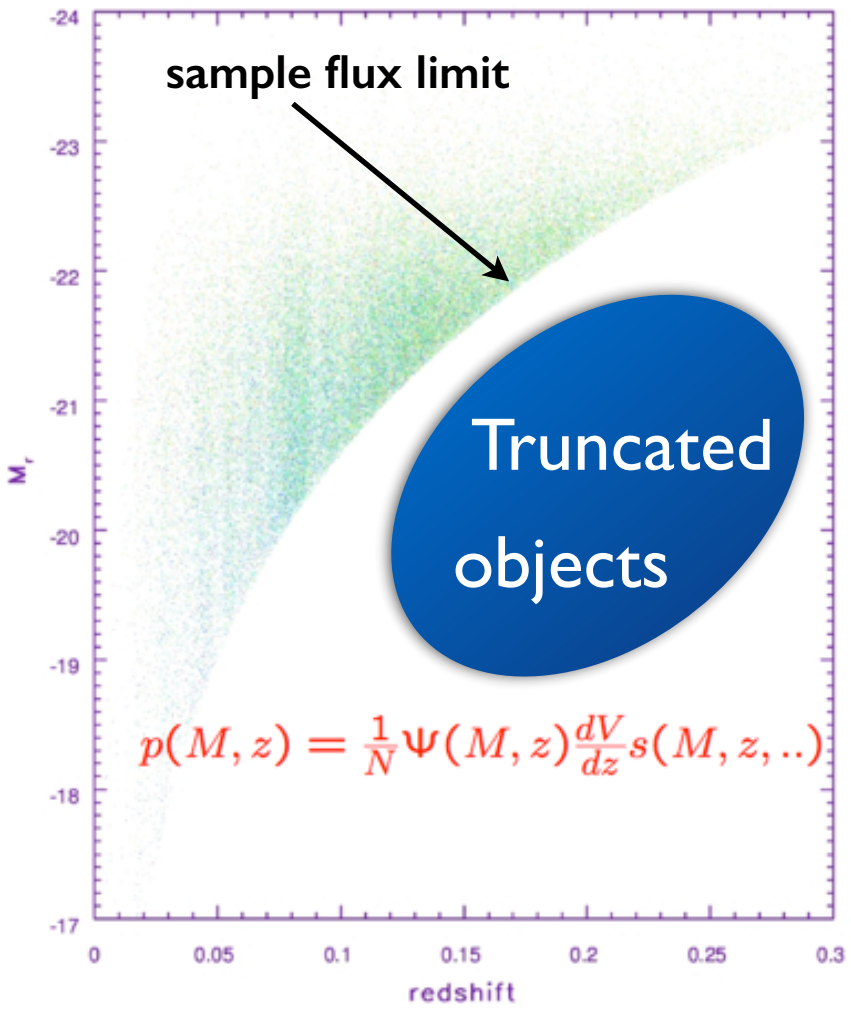
Given that the Uniform distribution satisfies the CLT assumptions (in particular, it has finite σ), the mean will behave as the CLT says. However, the mean is not the most efficient estimator of the location parameter in this case (nor the CLT ever claimed that…). Use instead

$$\tilde{\mu} = \frac{\min(x_i) + \max(x_i)}{2}$$

Or Bayes in other cases!

- **Correcting for truncation bias (C minus method)**
- Important when estimating luminosity functions

M vs z. for SDSS galaxies:



Luminosity Function

- Luminosity Function is the distribution in the luminosity–position plane; how many galaxies per unit interval in luminosity and unit volume (or redshift): $\Psi(M,z)$
- Imagine a tiny area with the widths ΔMr and Δz centered at some Mr and z in the plot to the left: count the number of galaxies, divide by $\Delta Mr \Delta z$, correct for the fraction of sky covered by your survey, and for the selection probability (a function of Mr, z, and possibly many other parameters): this gives you $\Psi(M,z)$.

Correcting for truncation bias (C minus method)

- Luminosity Function is the distribution in the luminosity–position plane; how many galaxies per unit interval in luminosity and unit volume: $\Psi(M, z)$

- Often, this is a separable function: $\Psi(M, z) = \Phi(M)\, n(z)$, where $\Phi(M)$ is the absolute magnitude (i.e. luminosity) distribution, and $n(z)$ is the number volume density.

- Luminosity is a product of flux and distance squared (ignore cosmological effects for simplicity): $L = 4\pi D^2 F$

- The samples are usually *flux-limited* (meaning: all sources brighter than some flux limit are detected) – the minimum detectable luminosity depends on distance: $L > 4\pi D^2 F_{min}$, or for absolute magnitude $M < M_{max}(D)$ (c.f. the first plot)

- Lynden-Bell (1971, MNRAS 155, 95); a non-parametric method that works for separable LFs, $\Psi(L, z) = \Phi(L)n(z)$

- practically all non-parametric methods can be reduced to the C^- method (Petrosian 1992)

- parametric methods are usually based on maximizing likelihood (e.g. Marshall 1985)

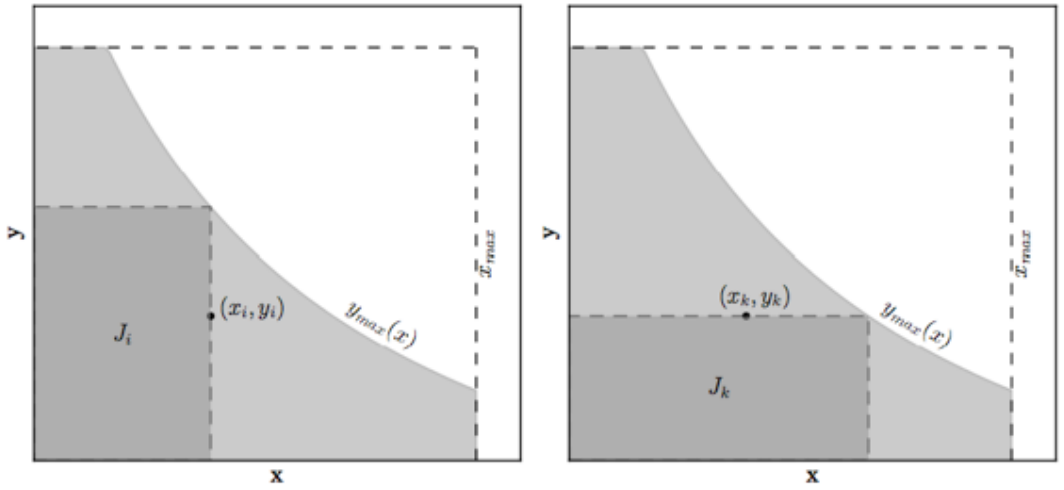
astroML implementation of the C^- method

- C^- method is applicable when the distributions along the two coordinates x and y are uncorrelated, that is, when we can assume that the bivariate distribution $n(x, y)$ is separable

$$n(x, y) = \Psi(x)\, \rho(y). \quad (1)$$

- Define a *comparable* or *associated* set for each object i such that $J_i = \{j : x_j < x_i, y_j < y_{max}(x_i)\}$; this is the largest x-limited and y-limited data subset for object i, with N_i elements (see the left panel).

- Sort the set J_i by y_j; this gives us the rank R_j for each object (ranging from 1 to N_i)



- **Correcting for truncation bias (C minus method)**
- make this plot by running
  %run fig_lyndenbell_toy.py

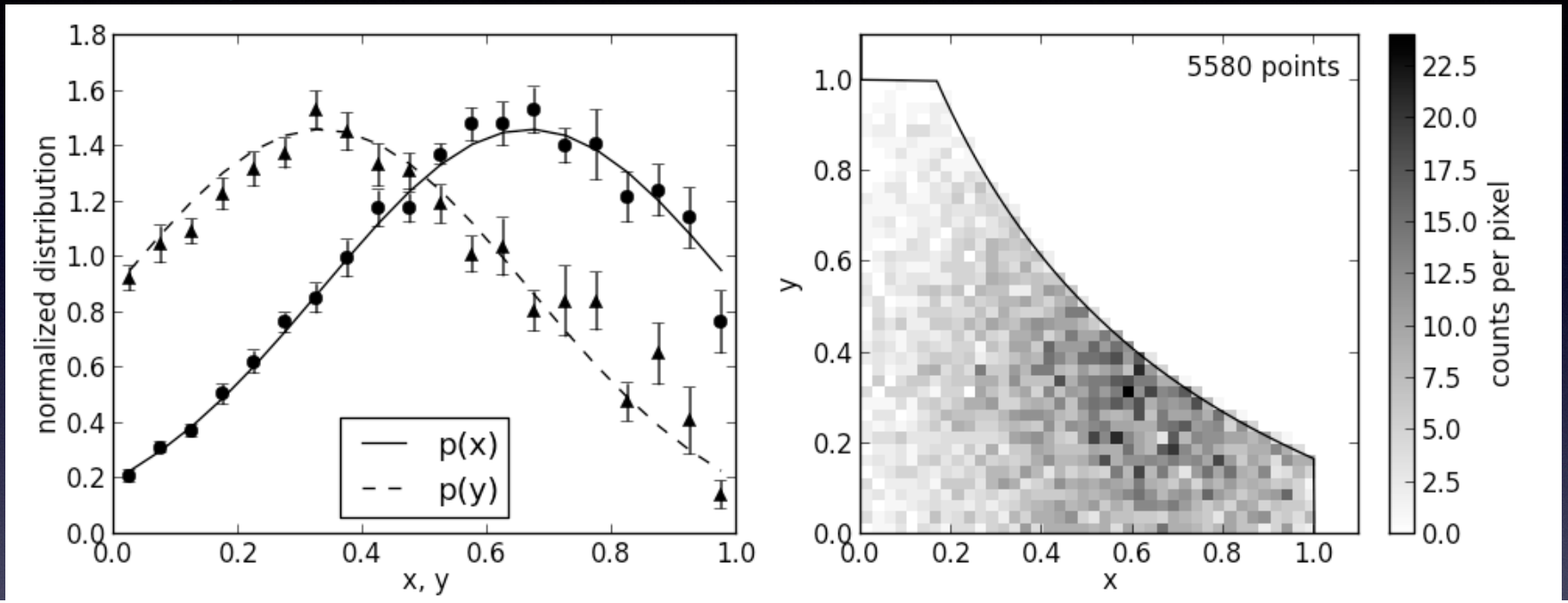


Figure 4.9.: An example of using Lynden-Bell's C^- method to estimate a bivariate distribution from a truncated sample. The lines in the left panel show the true one-dimensional distributions of x and y (truncated Gaussian distributions). The two-dimensional distribution is assumed to be separable; see eq. 4.85. A realization of the distribution is shown in the right panel, with a truncation given by the solid line. The points in the left panel are computed from the truncated data set using the C^- method, with error bars from 20 bootstrap resamples.

the work horse code lives in $astroMLdir/astroML/lumfunc.py

More info in http://www.astro.washington.edu/users/ivezic/Teaching/Astr511/lec5.pdf

- **Matched filter analysis of a low-SNR signal**
- a case of many-parameter model (need MCMC)

10.4.1. Searching for a burst signal

Consider a model where the signal is stationary, $y(t) = b_0 + \epsilon$, and at some unknown time, T, it suddenly increases, followed by a decay to the original level b_0 over some unknown time period. Let us describe such a burst by

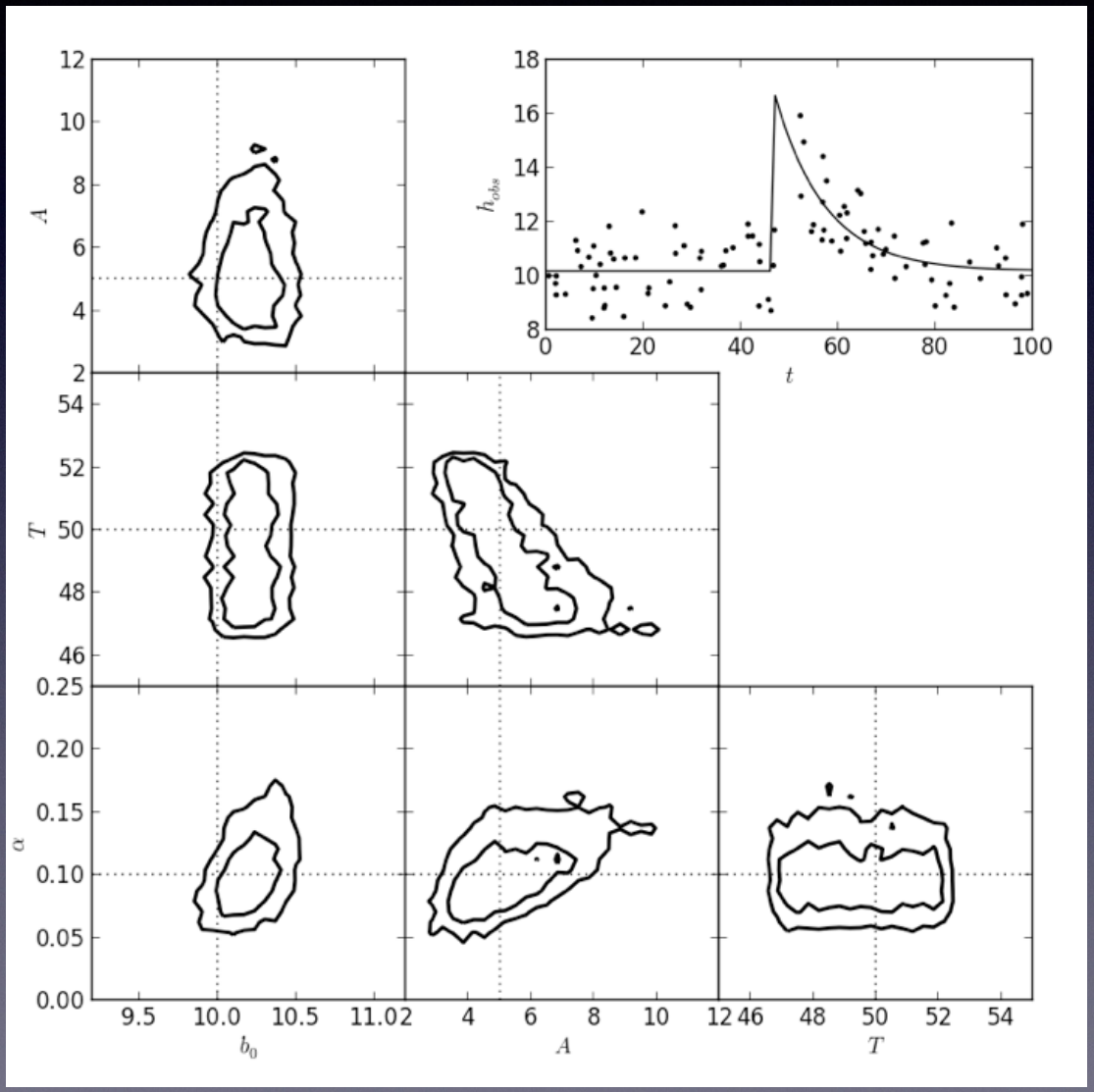
$$y_B(t|T, A, \boldsymbol{\theta}) = A\, g_B(t - T|\boldsymbol{\theta}), \tag{10.85}$$

where the function g_B describes the shape of the burst signal ($g_B(t < 0) = 0$). This function is specified by a vector of parameters $\boldsymbol{\theta}$ and can be analytic, tabulated in the form of a template, or treated in a nonparametric form. Typically, MCMC methods are used to estimate model parameters.

For illustration, we consider here a case with $g_B(t|\alpha) = \exp(-\alpha t)$. Figure 10.25 shows the simulated data and projections of posterior pdf for the four model parameters (b_0, T, A, and α). Other models for the burst shape can be readily analyzed using the same code with minor modifications.

Alternatively, the burst signal could be treated in the case of arrival time data, using the approach outlined in §10.3.5. Here, the rate function is not periodic, and can be obtained as $r(t) = (\Delta t)^{-1} y(t)$, where $y(t)$ is the sum of the stationary signal and the burst model (eq. 10.85).

- **Matched filter analysis of a low-SNR signal**
- make this plot by running (note “pickling” of results):
  %run fig_matchedfilt_burst.py



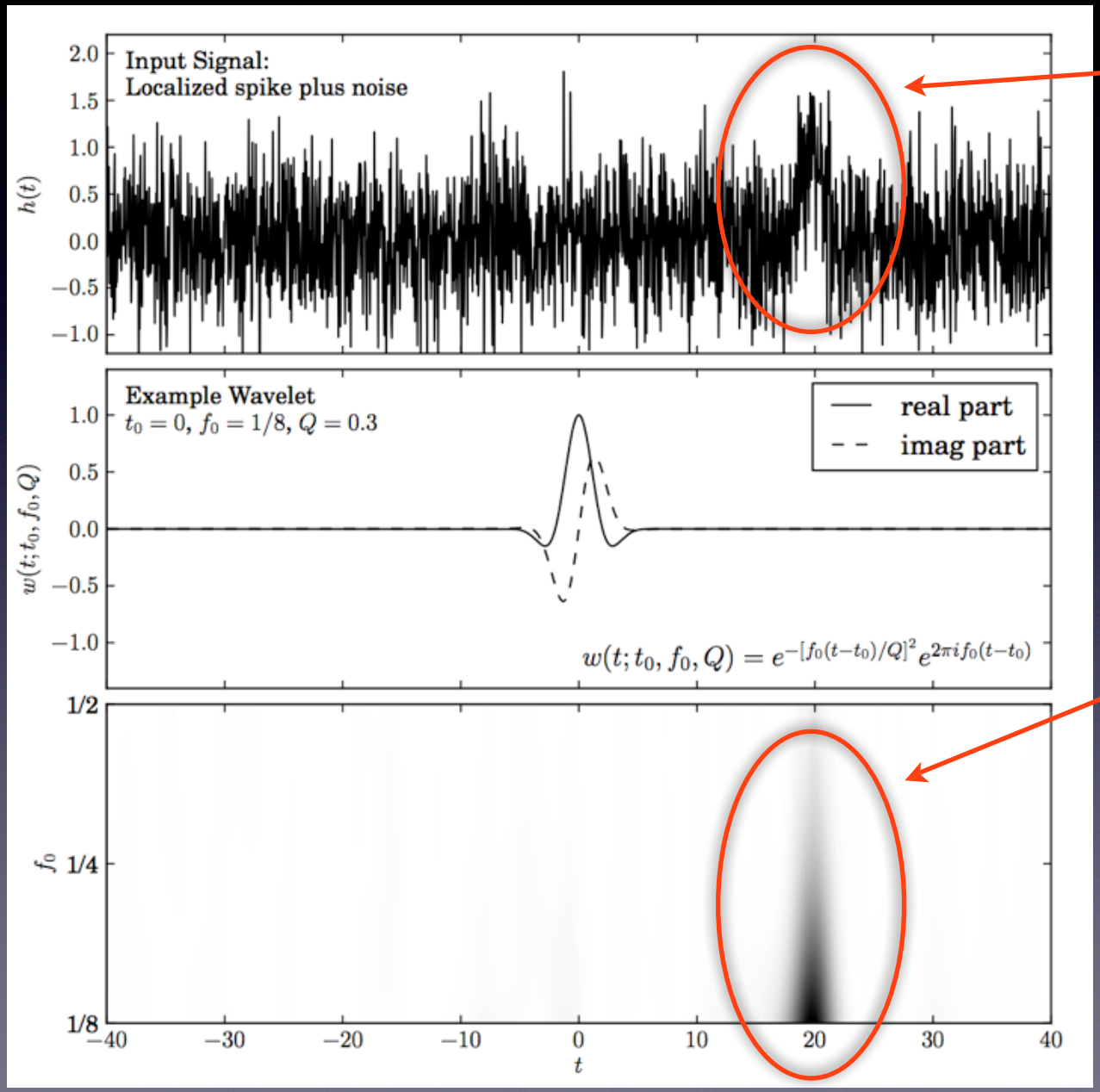
It calls pyMC module
Easy to change model
It tells you about covariances

Change something!

- **Find a low-SNR burst (wavelet analysis)**
- What if we don't know the signal shape?
- And we know that the signal is not periodic.
- The discrete wavelet transform (DWT) can be used to analyze the power spectrum of a time series as a function of time. While similar analysis could be performed using the Fourier transform evaluated in short sliding windows, the DWT is superior. If a time series contains a localized event in time and frequency, DWT may be used to discover the event and characterize its power spectrum. **PyWavelets** is a toolkit with wavelet analysis implemented in Python.

$$w(t|t_0, f_0, Q) = A \exp[i2\pi f_0(t - t_0)] \exp[-f_0^2(t - t_0)^2/Q^2]$$

- make this plot by running:
  %run fig_line_wavelet_PSD.py



Signal

Found it!