

Gaussian Mixture Model

Introduction

- For a recent review , see M. Kuhn and E. Feigelson arXiv:1711.11101

Likelihood of a datum x_i for a Gaussian mixture model is given by:

$$p(x_i|\theta) = \sum_{j=1}^M \alpha_j \mathcal{N}(\mu_j, \sigma_j) \quad \sum_{j=1}^M \alpha_j = 1$$

Log likelihood given by : $\ln L = \sum_{i=1}^N \ln \left[\sum_{j=1}^M \alpha_j \mathcal{N}(\mu_j, \sigma_j) \right]$

GMM (Contd)

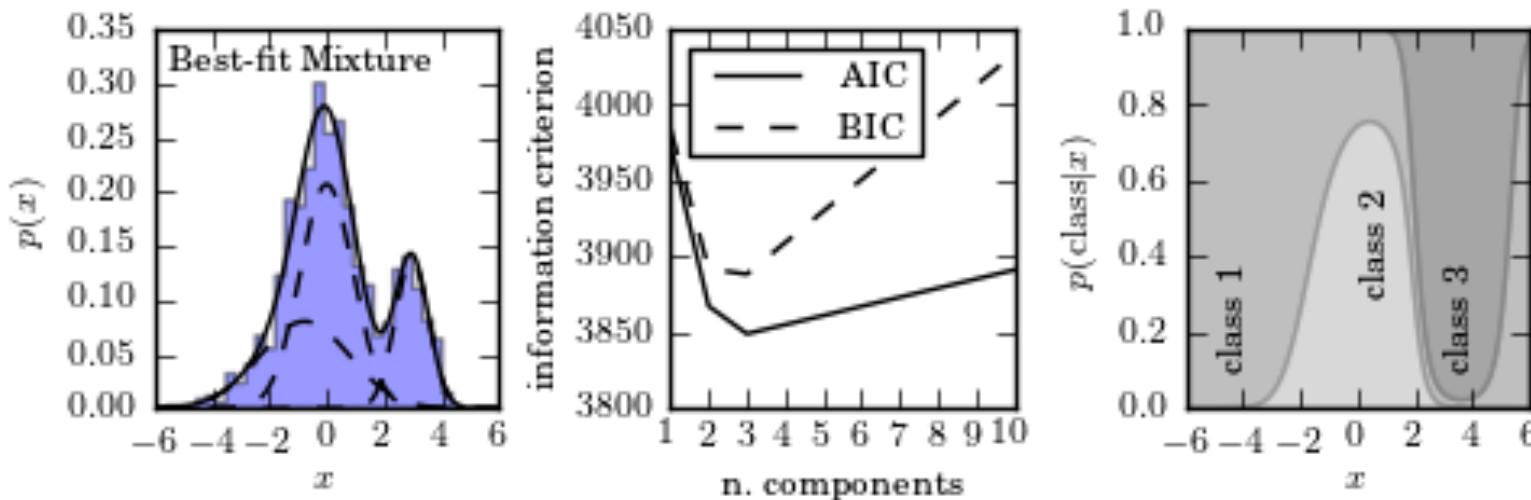
- Maximization of $\ln L$ can be trivially done using Expectation-Maximization Algorithm (Dempster, Laird, Rubin 1977)
- The optimum number of components can be computed using model comparison techniques.
- Generalization of GMM to account for errors is known as ``Extreme Deconvolution'' in astrophysics literature ([arXiv:0905.2979](#)). For an application of this see [arXiv:2206.02751](#)
- GMM can be generalized to other mixture models, but you cannot use E-M algorithm for maximization (also other variants , eg. Dirichlet mixture models) Routines available in R to do mixture models of t-distributions etc (see [arXiv:1910.08968](#))

GMM in Python

```
>>> import numpy as np
>>> from sklearn.mixture import GaussianMixture (GMM deprecated
after v0.17)
>>> X=np.random.normal(size=(100,1))
>>> model=GaussianMixture(2)
>>> model.fit(X)
    GaussianMixture(covariance_type='diag', init_params='wmc',
min_covar=0.001,
    n_components=2, n_init=1, n_iter=100, params='wmc',
random_state=None, thresh=None, tol=0.001, verbose=0)
>>> model.means_
array([[-0.44802848], [ 0.53396065]])
```

More detailed documentation in sklearn

Example from AstroML (Fig. 4.2)



Example of a one-dimensional Gaussian mixture model with three components. The left panel shows a histogram of the data, along with the best-fit model for a mixture with three components. The center panel shows the model selection criteria AIC (see Section 4.3) and BIC (see Section 5.4) as a function of the number of components. Both are minimized for a three-component model. The right panel shows the probability that a given point is drawn from each class as a function of its position. For a given x value, the vertical extent of each region is proportional to that probability. Note that extreme values are most likely to belong to class 1.

Example from Astro Literature

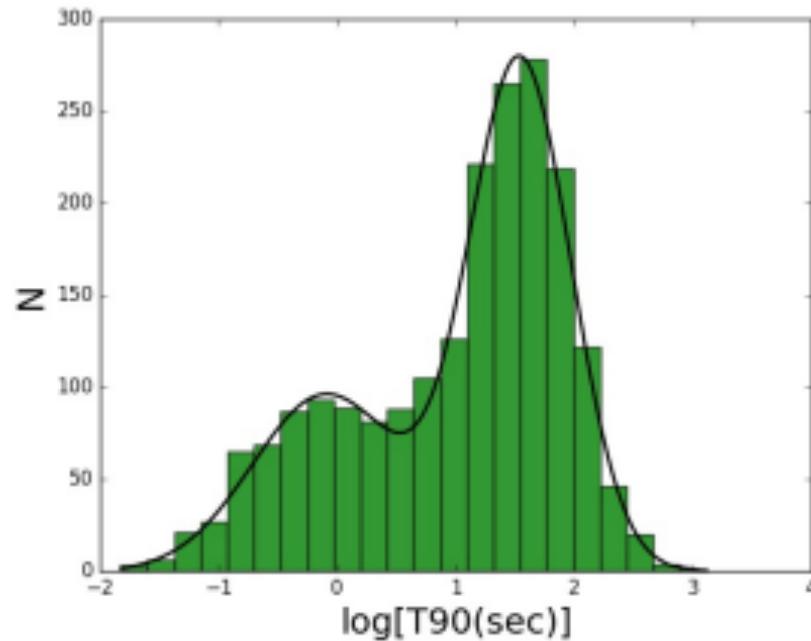


Fig. 1 A fit for the 2-component model for BATSE GRBs.
Details of the fits can be found in Table 1.

Soham Kulkarni & SD (arXiv:1612.08235)

Introduction

Data Mining

- Set of techniques for analyzing and describing structured data, for example finding patterns in large datasets.

Examples : Density estimation, unsupervised classification, clustering, principal component analysis, locally linear embedding.

Often interchanged with knowledge discovery. Traditionally deals with unsupervised learning techniques (referred to by statisticians as exploratory data analysis)

Machine Learning

- Umbrella term for a set of techniques for interpreting data by comparing them to models for data behavior.

Examples : regression methods, supervised classification methods, maximum likelihood estimators, Bayesian methods.

Also known as inference techniques, data-based statistical inferences or fitting. Emphasis has been on prediction of one variable based on the other variable, also known as supervised learning.

One-stop link for course content

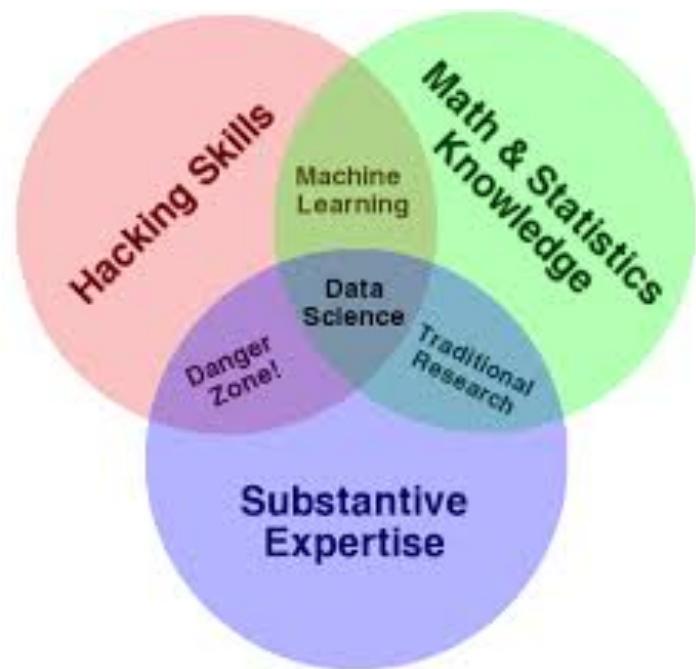
- <http://www.astroml.org>

- Current course website

https://piazza.com/iit_hyderabad/spring2023/ep4130ph6130/info

- Google classroom to be used for uploading assignments rmv2er7

What is Data Science?



Cross-disciplinary set of skills

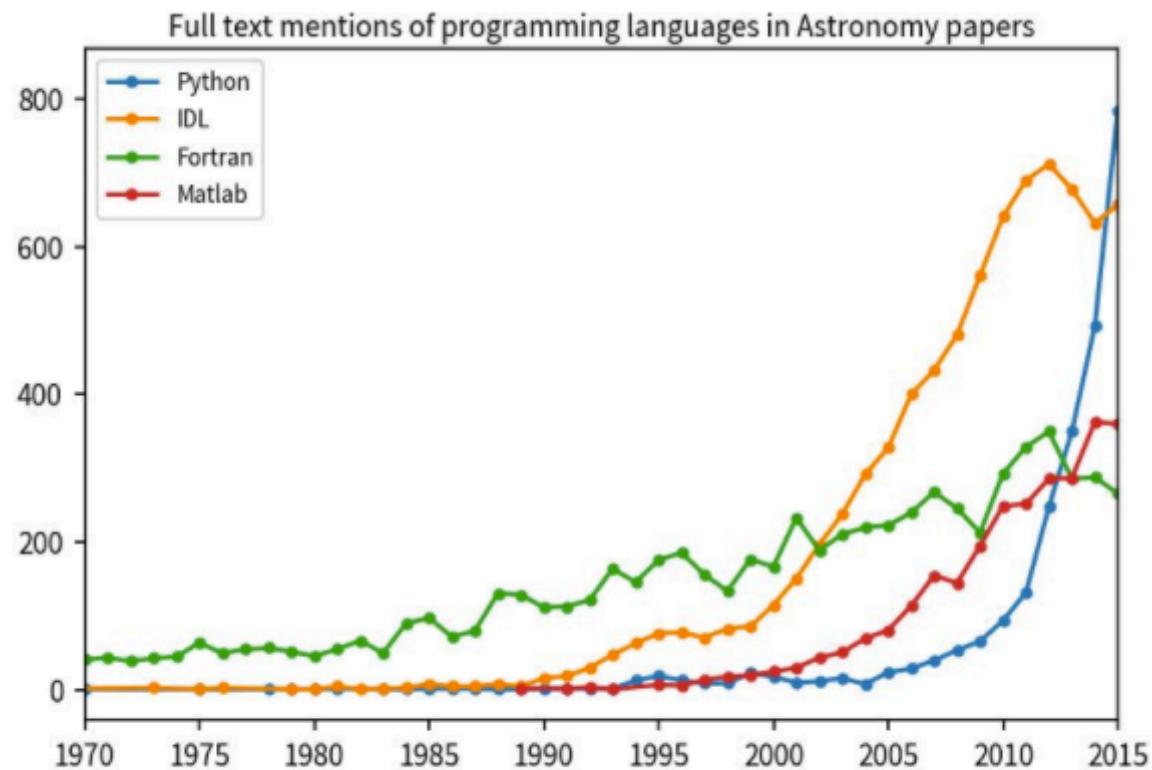
- Statistician
- Computer Scientist
- Domain expertise

Credit:drewconway.com

Jayant Narlikar to C.R. Rao

You may ask “What can a hard headed statistician offer to a starry eyed astronomer?”. The answer is “Plenty”. One normally associates statistics with large numbers, and astronomy is full of large numbers...I have every reason to believe that increased interaction between statistics and astronomy will be to the benefit of both subjects.

Trend of Programming languages in Astro literature



From <https://twitter.com/astrofrog/status/787007261877166080>
does not include R and C (as difficult to parse texts)

Statistical Data Analysis tasks in Astronomy

Photometric Redshifts (Regression)

Source Classification

Dimensionality Reduction/Visualization

Clustering

N-point statistics

Period Finding

Transient and Outlier Detection

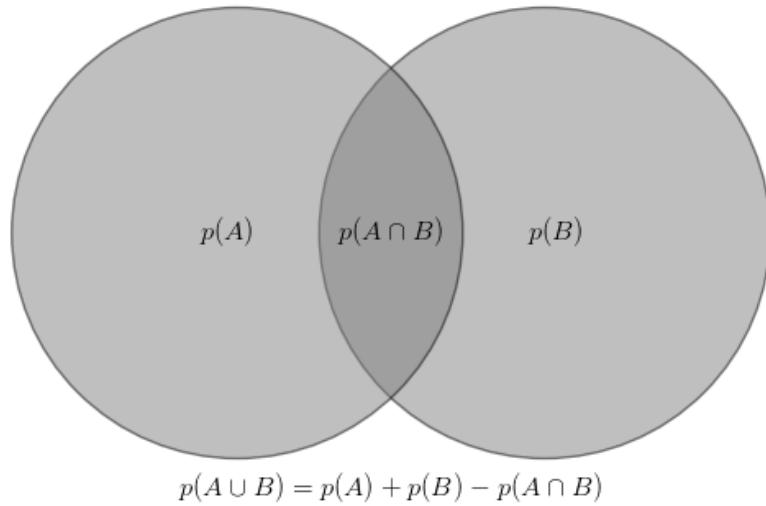
Density Estimation

Matched Filtering

Source Extraction

Cross-Matching

Probability Axioms



$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Probability for both A and B to happen = $P(A \cap B)$

$$P(A \cap B) = P(A/B)P(B) = P(B/A)P(A)$$

$P(A)$ is a probability if it satisfies three axioms:

$$P(A) > 0$$

Sum $P(A) = 1$ (for all possible outcomes)

For disjoint events A_1, A_2, \dots

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$$

Bayes Theorem

$P(A/B)$ is conditional probability of event A given that (on conditioned on) B has occurred

If events B_i , $i=1,\dots,N$ are disjoint and union is set of all possible outcomes then

$$P(A) = \sum_{i=1}^N P(A \cap B_i) = \sum_{i=1}^N P(A|B_i)P(B_i)$$

This is called “[Law of Total Probability](#)”

Conditional probabilities also satisfy law of total probability. Assuming an event C_i is not mutually exclusive with A or any of the B_i then

$$P(A|B) = \sum_i P(A|B \cap C_i)P(C_i|B)$$

Probability rules were derived from two different sets of axioms by Cox and Kolmogorov (Jaynes)

If events C_i , $i=1,\dots,N$ are disjoint and union is set of all possible outcomes then

$$P(A \cap B) = \sum_i P(A \cap B \cap C_i)$$

$$P(A \cap B) = \sum_i P(A|B \cap C_i)P(B \cap C_i)$$

$$P(A \cap B) = \sum_i P(A|B \cap C_i)P(C_i|B)P(B)$$

$$P(A|B) = \frac{P(A\cap B)}{P(B)}$$

$$P(A|B)=\frac{\sum_i P(A|B\cap C_i)P(C_i|B)P(B)}{P(B)}$$

$$P(A|B)=\sum_i P(A|B\cap C_i)P(C_i|B)$$

Random Variables

- A random variables is a variable whose value results from the measurement of a quantity subject to stochastic variations.
- Independent identically distributed random variables are drawn from the same distribution and independent.

Two random variables x and y are *independent* if and only if

$$P(x,y) = P(x) P(y)$$

Conditional Probability and Bayes Rule

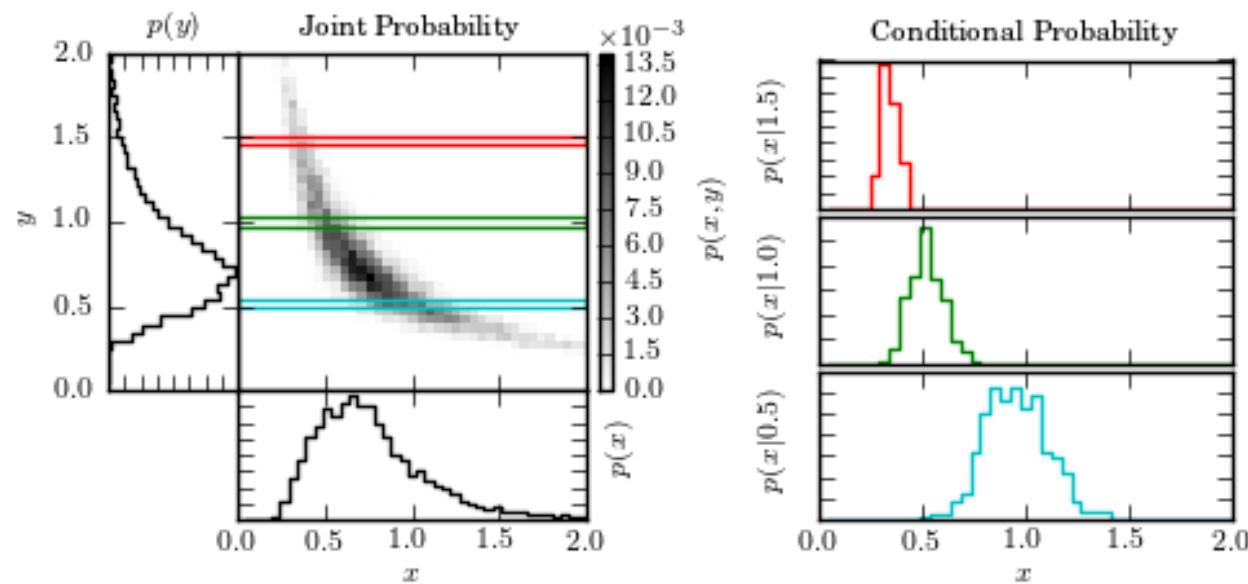
If two continuous random variables are not independent, it follows that

$$p(x,y) = p(x|y)p(y) = p(y|x) p(x)$$

Marginal Probability function defined as

$$p(x) = \int p(x,y)dy \quad \text{By combining above two equations we get}$$

$$p(x) = \int p(x|y)p(y)dy$$



$P(x|y=y_0)$ are one-dimensional “slices” through the two-dimensional Image $p(x,y)$ at given values of y_0 divided by the value of marginal distribution $p(y)$

$$\int P(x) dx = 1$$

Code to reproduce this available from astroml.org website

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x|y)p(y)}{\int p(x|y)p(y)dy}$$

For a discrete random variable y_j with M possible values the above integral becomes a sum:

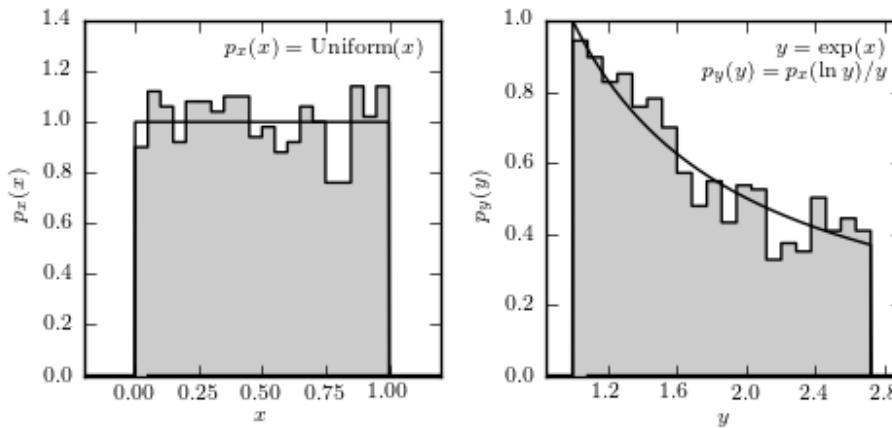
$$p(y_j|x) = \frac{p(x|y_j)p(y_j)}{\sum_{j=1}^M p(x|y_j)p(y_j)}$$

Homework : Read about Monty Hall problem

Transformations of Random Variables

Any function of a random variable x $y = \phi(x)$ is a random variable. We can calculate $p(y)$ from $p(x)$ as follows :

$$p(y) = p[\Phi^{-1}(y)] \left| \frac{d\Phi^{-1}(y)}{dy} \right|$$



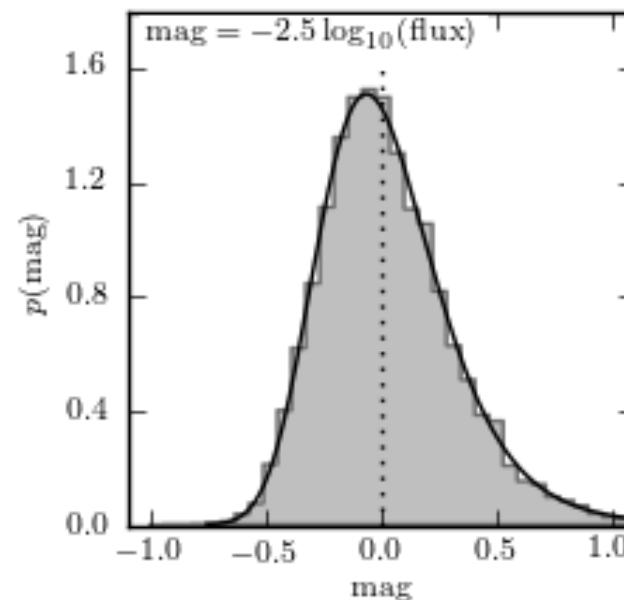
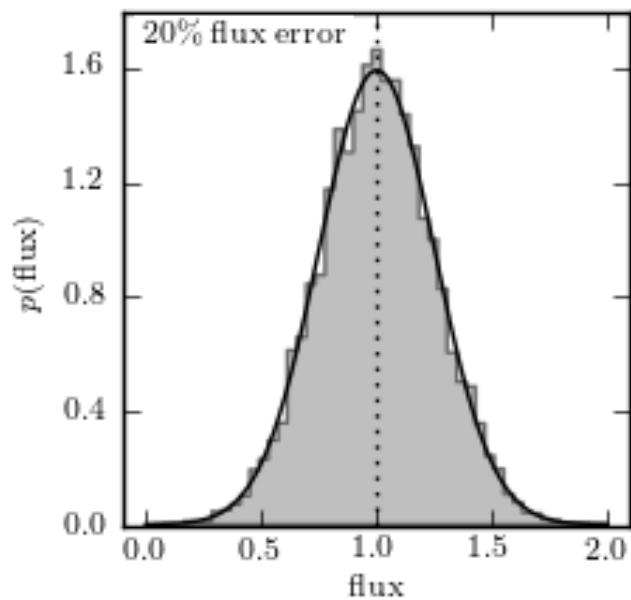
Cumulative statistics such as medians do not change their order under monotonic transformations

If uncertainty in x at a given value of x_0 is given by σ_x then we can use Taylor series expansion to Estimate the uncertainty in y at $y_0 = \phi(x_0)$

$$\sigma_y = \left| \frac{d\Phi(x)}{dx} \right|_0 \sigma_x$$

Sometimes this can lead to misleading results for non-linear transformations
for example (in Astronomy) magnitude = $-2.5\log(\text{Flux})$

Example :



Error Propagation (Without Covariances)

Consider $G = G(x_1, x_2, \dots, x_n)$ with uncertainties $\sigma_1, \sigma_2, \dots, \sigma_n$

$$\sigma_G^2 = \sum_{i=1}^N \left(\frac{\partial G}{\partial x_i} \right)^2 \sigma_{x_i}^2$$

Iff the errors in x_1, x_2, \dots are uncorrelated

Eg. $\Delta m = m_1 - m_2$
 $\sigma_{\Delta m}^2 = \sigma_{m1}^2 + \sigma_{m2}^2$

Error Propagation (with Covariances)

$$\sigma_G^2 = \sum_{i=1}^N \left(\frac{\partial G}{\partial x_i} \right)^2 \sigma_{x_i}^2 + 2\sigma_{x_1 x_2}^2 \frac{\partial G}{\partial x_1} \frac{\partial G}{\partial x_2} + \dots$$

where

$$\sigma_{x_1 x_2}^2 = \frac{\sum_{i=1}^N [(x_{1i} - \bar{x}_1)][(x_{2i} - \bar{x}_2)]}{N}$$

Ref. Bevington's book

Descriptive Statistics

An arbitrary distribution $h(x)$ is characterized by its location parameters, scale or width parameters and “shape” parameters.

When they are based on the distribution $h(x)$ they are called *population* statistics. If they are based upon a finite-sized dataset, they are called *sample* statistics.

Arithmetic mean based upon expectation value

$$\mu = E(x) = \int_{-\infty}^{+\infty} x h(x)$$

Variance

$$V = \int_{-\infty}^{+\infty} (x - \mu)^2 h(x) dx$$

Standard Deviation

$$\sigma = \sqrt{V}$$

Skewness

$$\Sigma = \int_{-\infty}^{+\infty} \left(\frac{x - \mu}{\sigma} \right)^3 h(x) dx$$

Kurtosis

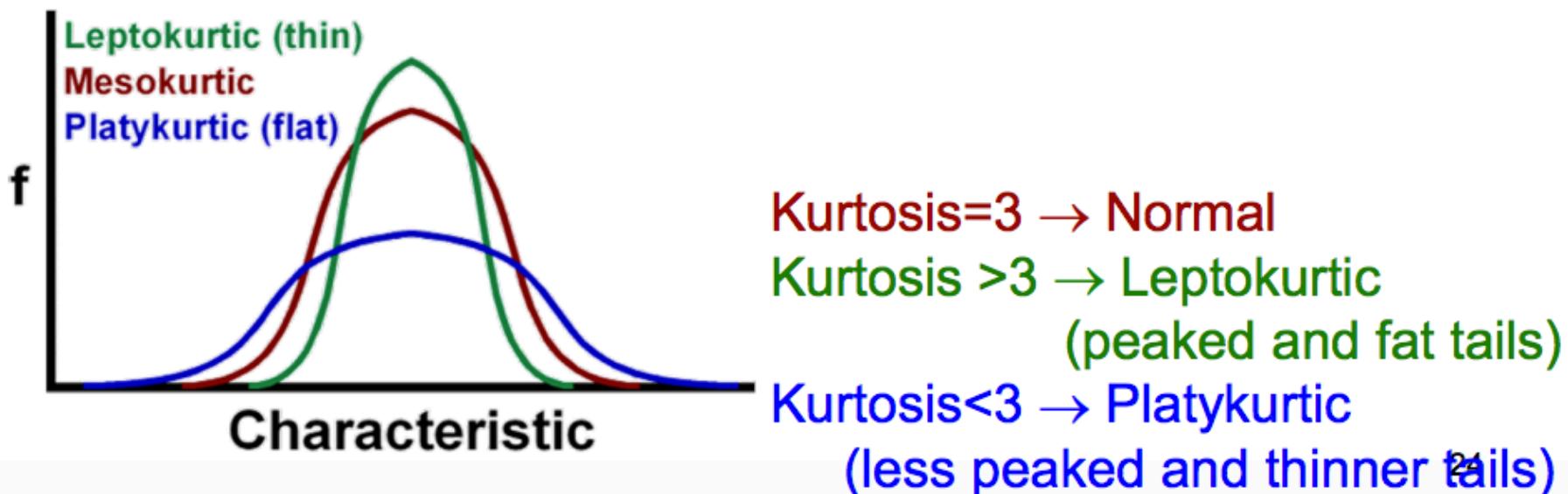
$$K = \int_{-\infty}^{+\infty} \left(\frac{x - \mu}{\sigma} \right)^4 h(x) dx - 3$$

Variance, skewness., kurtosis related to k^{th} central moment of a distribution ($k=2,3,4$)

Kurtosis

- Measure of the peakedness of the pdf. Describes the shape of the r.v.

$$Kurtosis = \frac{E(X - \mu)^4}{\sigma^4} = \frac{\mu_4}{\mu_2^2}$$



Moments From Generating Function

Ref: arXiv:0712.3028

Generating Function allows you to calculate the moments of a distribution

$$Z(k) = \langle \exp(ikx) \rangle = \int \exp(ikx) P(x) dx$$

This can be written as an infinite series by expanding the exponential giving :

$$Z(k) = \sum_{n=0}^{\infty} \frac{(ik)^n}{n!} \hat{\mu}_n$$

$$\hat{\mu}_n = (-i^n) \frac{d^n}{dk^n} Z(k) |_{k=0}$$

Absolute Deviation about d

$$\delta = \int_{-\infty}^{+\infty} |x - d| h(x) dx$$

Absolute deviation about the mean ($d=\text{mean}(x)$) is called mean deviation

Mode (or most probable value in case of unimodal functions) x_m

$$\left(\frac{dh(x)}{dx} \right)_{x_m} = 0$$

P % quantiles (or p percentiles)

$$\frac{p}{100} = \int_{-\infty}^{q_p} h(x) dx$$

All the moments are built into NumPy and SciPy. Useful functions are

```
numpy.median, numpy.mean, numpy.var  
numpy.percentile, numpy.std, scipy.stats.skew,  
scipy.stats.kurtosis,  
scipy.stats.mode
```

```
import numpy as np  
x = np.random.random(100)  
q25,q50,q75 = np.percentile(x,[25,50,75])
```

Difference between third and first quartile is called interquartile range

A useful relation between mode, median and mean valid for mildly non-gaussian distributions

$$\text{Mode} = 3(\text{median}) - 2(\text{mean})$$

Data-Based Estimates of Descriptive Statistics

If the above quantities are derived from data, they are called sample statistics (instead of population Statistics).

Assume we have N given measurements x_i for $i=1, \dots, N$ abbreviated as $\{x_i\}$

For a sample of N measurements

$$\int_{-\infty}^{+\infty} g(x)h(x)dx \equiv (1/N) \sum_{i=1}^N g(x_i)$$

Sample arithmetic mean and standard deviation given by :

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$$

If the samples have an error σ_i , mean and error in mean are given by:

$$\bar{x} = \frac{\sum_{i=1}^N x_i / \sigma_i^2}{\sum_{i=1}^N 1 / \sigma_i^2}$$
$$\sigma_{\bar{x}}^2 = \frac{1}{\sum_{i=1}^N (1 / \sigma_i^2)}$$

Sample Standard deviation is calculated as follows:

$$s = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N - 1}}$$

N-1 is used so that variance is unbiased

Uncertainty in the standard mean is given by :

(if errors in each data point are equal)

$$\sigma_{\bar{x}} = \frac{s}{\sqrt{N}}$$

$$\sigma_s = \frac{s}{\sqrt{2(N - 1)}}$$

For real data with outliers, calculation of s from data samples can lead to wrong estimates

Median Absolute Deviation (wikipedia)

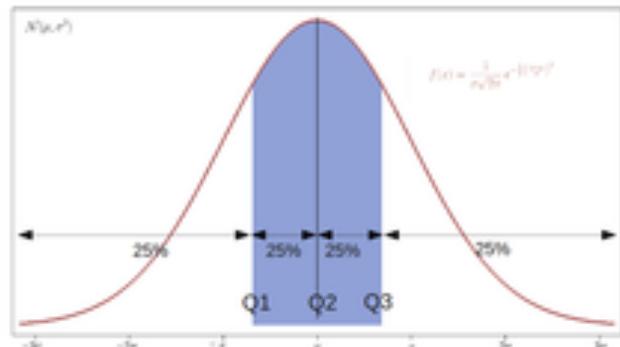
$$MAD = median(|x_i - median(x)|)$$

For a normal distribution, $\sigma = 1.482 \text{ MAD}$ where σ is the std deviation of Gaussian distribution

Alternately, you can use a rank based estimate of the standard deviation.
Inter-quartile range ($q_{75} - q_{25}$) is a more robust estimator of the scale parameter than the standard deviation.

Even in absence of outliers for some distributions which do not have finite variance such as the Cauchy distribution, the median and interquartile range are the best choices for estimating the location and scale parameters.

For a Gaussian, $\sigma_G = 0.7413(q_{75} - q_{25})$ (σ_G can be computed with astroML library)



Quantiles

Source : wikipedia

Specialized quantiles [\[edit\]](#)

Some q -quantiles have special names:[\[citation needed\]](#)

- The only 2-quantile is called the [median](#)
- The 3-quantiles are called [tertiles](#) or [terciles](#) → T
- The 4-quantiles are called [quartiles](#) → Q; the difference between upper and lower quartiles is also called the [interquartile range](#), [midspread](#) or [middle fifty](#) → IQR = $Q_3 - Q_1$
- The 5-quantiles are called [quintiles](#) → QU
- The 6-quantiles are called [sextiles](#) → S
- The 7-quantiles are called [septiles](#)
- The 8-quantiles are called [octiles](#) → O

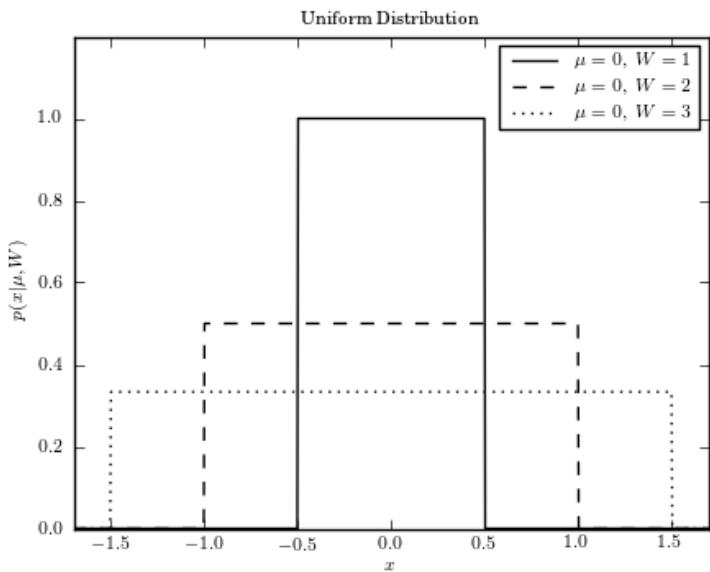
Some Definitions

- Probability Density Function : The function $h(x)$ quantifies the probability that a value lies between x and $x+dx$ equal to $h(x) dx$ and is called the probability density function (pdf).
- Probability Mass function : When x is discrete, this is called probability mass function
- Cumulative Distribution Function (cdf) $H(x)$

$$H(x) = \int_{-\infty}^x h(x')dx'$$

Examples of Distribution Functions

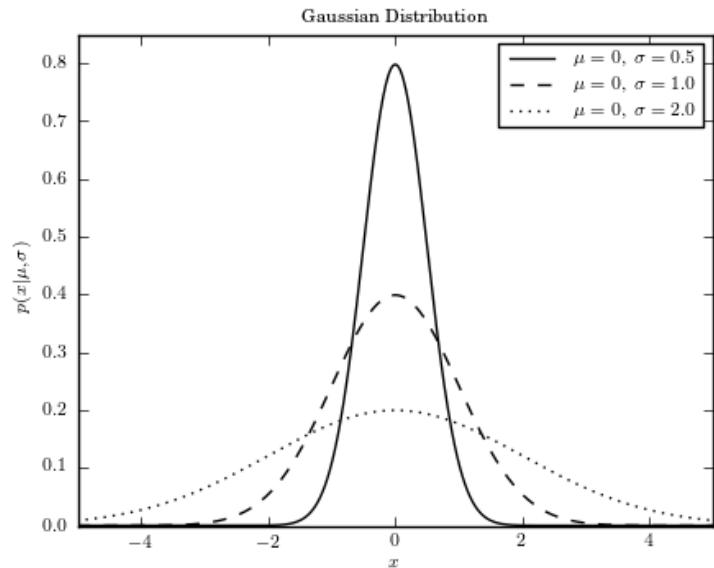
Uniform Distribution



```
from scipy import stats
dist = stats.uniform(0,2)
# left edge at 0 and width=2
r = dist.rvs(10)
# 10 random draws
P = dist.pdf(1)
#PDF evaluated at x=1
Look at scipy.stats page for
more information
```

$$P(x|\mu, W) = \frac{1}{W} \text{ for } |x - \mu| \leq W/2$$

Gaussian Distribution



```
dist = stats.norm(0,1)
#mean=0, stddev=1
r = dist.rvs(10)
p = dist.pdf(0)
```

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$

Also called Normal distribution

$$\mathcal{N}(\mu, \sigma)$$

Convolution of two Gaussians is also a Gaussian function

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(x')g(x - x')dx' = \int_{-\infty}^{+\infty} f(x - x')g(x')dx'$$

Q: Consider convolution of two normal distributions : $\mathcal{N}(\mu_0, \sigma_0)$ and $\mathcal{N}(b, \sigma_e)$

What is the mean and std. deviation of the resulting Gaussian?

Cumulative distribution function of a Gaussian distribution is given by :

$$P(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^x \exp\left(\frac{-(x - \mu)^2}{2\sigma^2}\right) = \frac{1}{2}(1 \pm \text{erf}\left(\frac{|x - \mu|}{\sqrt{2}\sigma}\right))$$

where erf = Gauss error function

Gauss error function is available in `scipy.special`

```
from scipy.special import erf  
erf(1) = 0.842
```

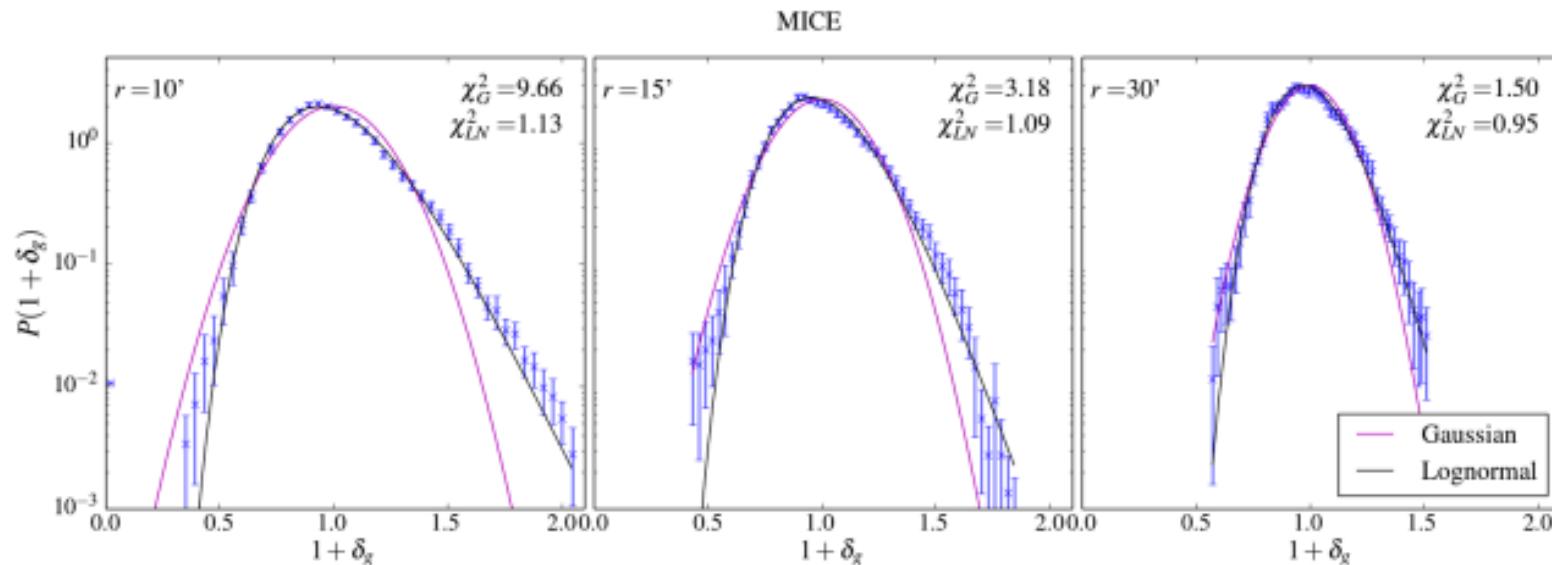
$$\int_a^b p(x|\mu, \sigma) dx = P(b|\mu, \sigma) - P(a|\mu, \sigma)$$

For $a = \mu - M\sigma$ and $b = \mu + M\sigma$, the above integral = $\text{erf}(M/\sqrt{2})$

$M=1,2,3$, give values of 0.68, 0.954, 0.997 respectively for $\text{erf}(M/\sqrt{2})$

If x follows a Gaussian distribution, $\exp(x)$ follows a log-normal distribution.

LogNormal Distribution



Number distribution of galaxies as a function of density contrast

[arxiv:1605.02036](https://arxiv.org/abs/1605.02036)

Lognormal Distribution Examples

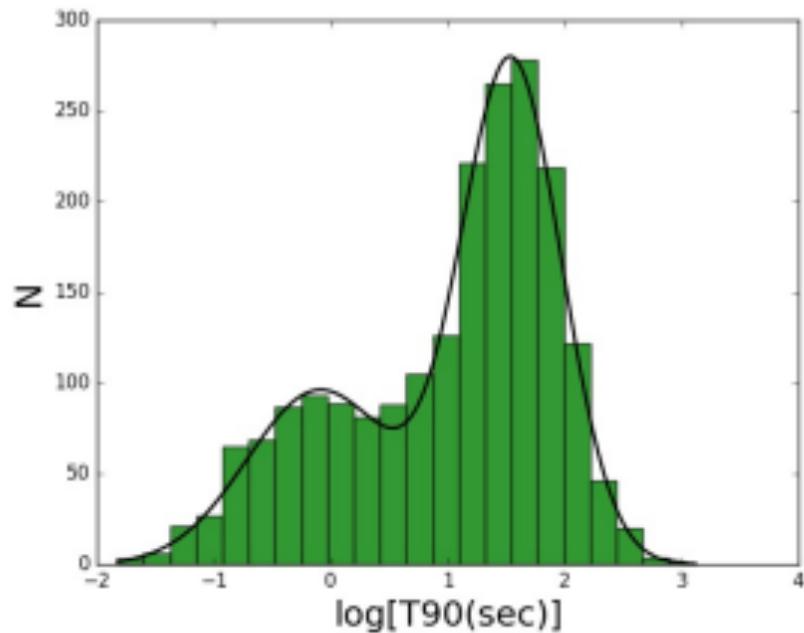


Fig. 1 A fit for the 2-component model for BATSE GRBs.

Gamma-Ray Burst Duration ([Soham Kulkarni arXiv:1612.08235](#))

How to Gaussianize a distribution

Perform a Box-Cox (1964) transformation on the data (arXiv:1508.00931)

$$y_\lambda(a) = \begin{cases} \frac{a^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log a & \text{if } \lambda = 0 \end{cases}$$

$$\bar{y}_\lambda = \sum_{i=1}^N \frac{y_\lambda(a_i)}{n}$$

Maximum likelihood estimate of the variance of the transformed data is given by

$$s_{\lambda}^2 = \sum_{i=1}^N \frac{(y_{\lambda}(a_i) - \bar{y}_{\lambda})^2}{n}$$

We choose λ such that we maximize the log likelihood function

$$I(\lambda) = -\frac{n}{2} \log(2\pi) - \frac{n}{2} - \frac{n}{2} \log s_{\lambda}^2 + (\lambda - 1) \sum_{i=1}^N \log(a_i)$$

y_{λ} (a) will be an exact normal distribution if $\lambda=0$ or $1/\lambda$ is an even integer

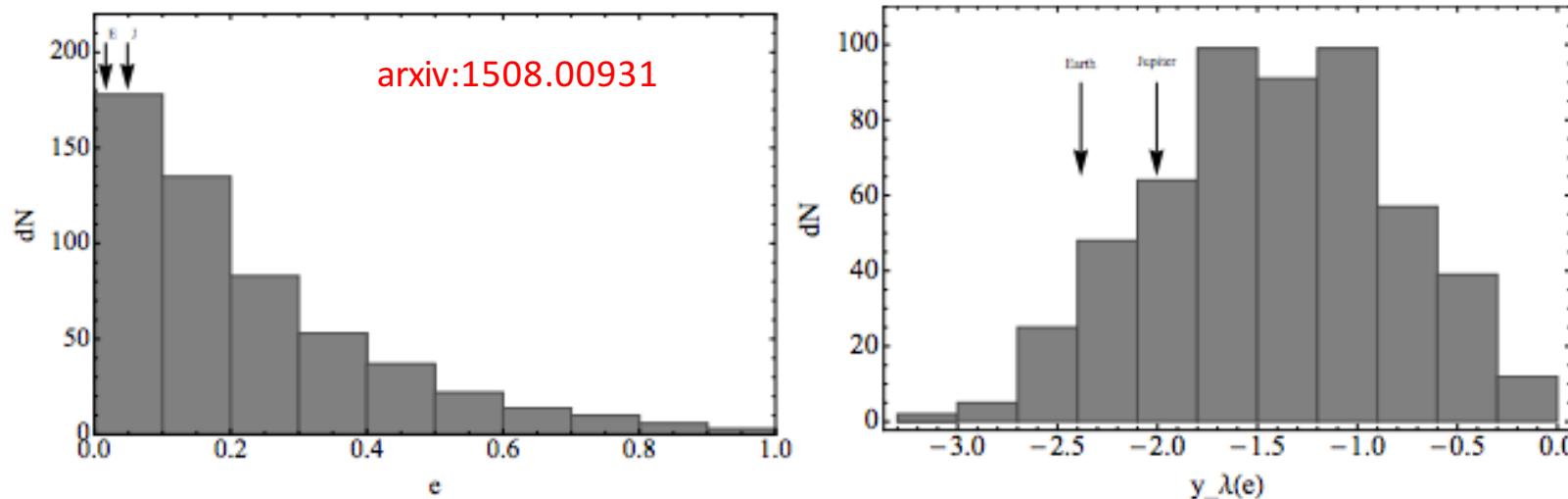


FIG. 1.— Left: Eccentricity distribution for all observed exoplanets with a measured orbital eccentricity. Right: Box-Cox transformed distribution of exoplanet eccentricities. The total number of exoplanets is 539.

Eccentricity distribution of exoplanets before and after Box-Cox transformation

Box-Cox Transformation in Python (see also stackexchange for examples)

scipy.stats.boxcox

scipy.stats.boxcox(*x*, *lmbda=None*, *alpha=None*)

[\[source\]](#)

Return a positive dataset transformed by a Box-Cox power transformation.

Parameters: *x* : ndarray

Input array. Should be 1-dimensional.

lmbda : {None, scalar}, optional

If *lmbda* is not None, do the transformation for that value.

If *lmbda* is None, find the lambda that maximizes the log-likelihood function and return it as the second output argument.

alpha : {None, float}, optional

If *alpha* is not None, return the `100 * (1-alpha)%` confidence interval for *lmbda* as the third output argument. Must be between 0.0 and 1.0.

Returns: *boxcox* : ndarray

Box-Cox power transformed array.

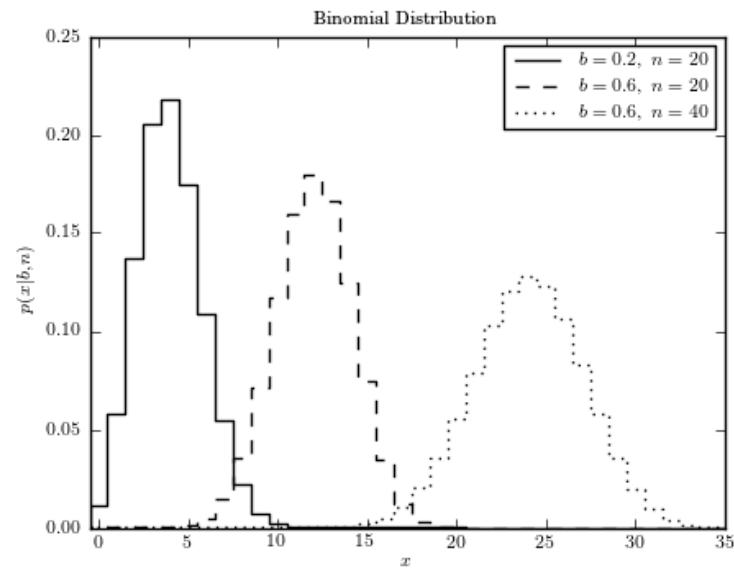
maxlog : float, optional

If the *lmbda* parameter is None, the second returned argument is the lambda that maximizes the log-likelihood function.

(min_ci, max_ci) : tuple of float, optional

If *lmbda* parameter is None and *alpha* is not None, this returned tuple of floats represents the minimum and maximum confidence limits given *alpha*.

Binomial Distribution



```
from scipy import stats  
dist=stats.binom(20,0.7)  
r= dist.rvs(10)  
P = dist.pmf(8) # prob.  
evaluated at k=8
```

Binomial distribution describes the distribution of a variable that can only take discrete values
If probability of success is b , distribution of a discrete variable k that measures how many times Success occurs in N trials is given by [Probability Mass Function](#)

$$p(k|b, N) = \frac{N!}{k!(N-k)!} b^k (1-b)^{N-k}$$

$N=1$ is called Bernoulli distribution

Mean of a binomial distribution is given by

$$\bar{k} = Nb$$

Standard deviation is given by :

$$\sigma_k = [Nb(1-b)]^{1/2}$$

Binomial distribution can be generalized to a Multinomial distribution in case a variable has more than two discrete values.

Poisson Distribution

Poisson distribution special case of the binomial distribution describing the distribution of a discrete variable, when the number of trials (N) goes to infinity and probability of success ($p=k/N$) stays fixed.

Distribution of number of success k is controlled by $\mu = k N$ and is given by

$$P(k|\mu) = \frac{\mu^k \exp(-\mu)}{k!}$$

Poisson distribution is ubiquitous describes independent point process:
photon noise, radioactive decay, galaxy distribution for very few galaxies, point sources

Mean (or expectation value) = μ

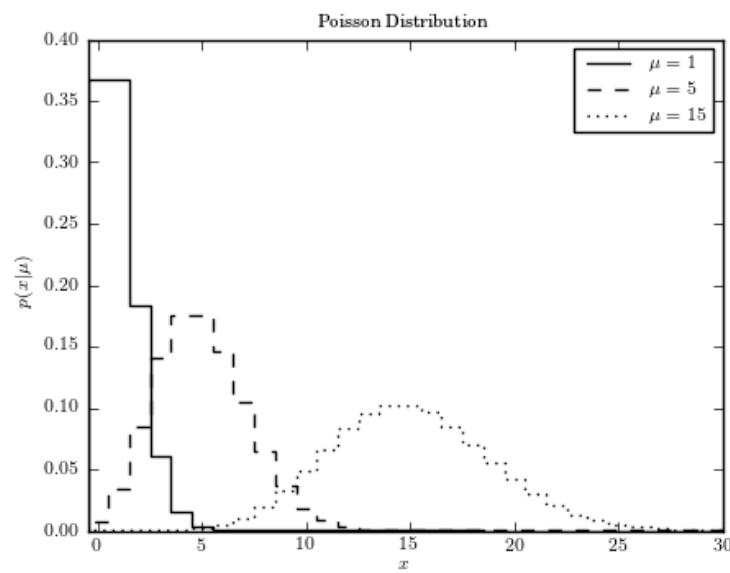
Standard deviation = $\sqrt{\mu}$

As μ increases the Poisson distribution becomes more and more similar to Gaussian distribution

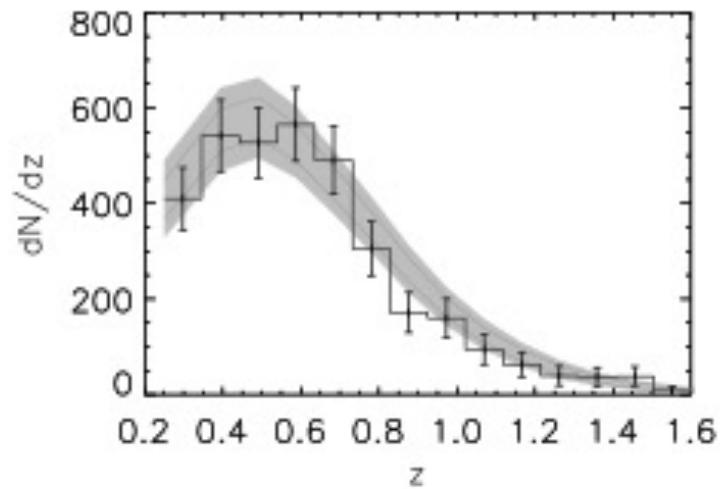
As μ increases the Poisson distribution becomes more and more similar to a Gaussian distribution given by $\mathcal{N}(\mu, \sqrt{\mu})$

Difference between Mean and Median does not become 0 but becomes $1/6$

```
from scipy import stats
dist = stats.poisson(5)
r = dist.rvs(10)
p = dist.pmf(3)
```

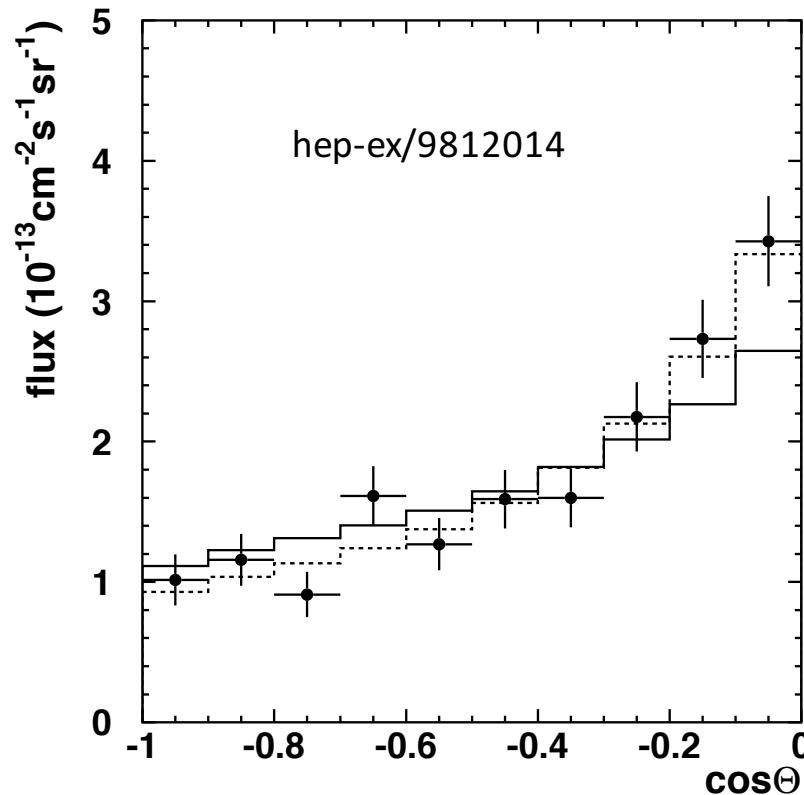


Examples of Poisson Distribution



Galaxy clusters discovered with SPT
as a function of redshift

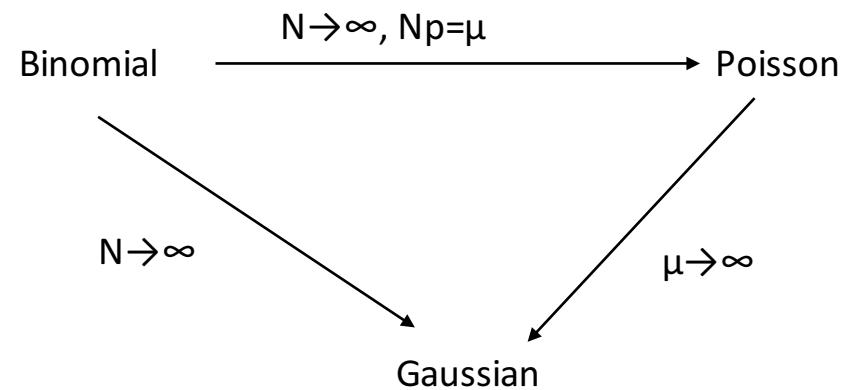
arXiv:1603.06522



Flux of upward muons in Super-K as a function
of zenith angle (1998)

hep-ex/9812014

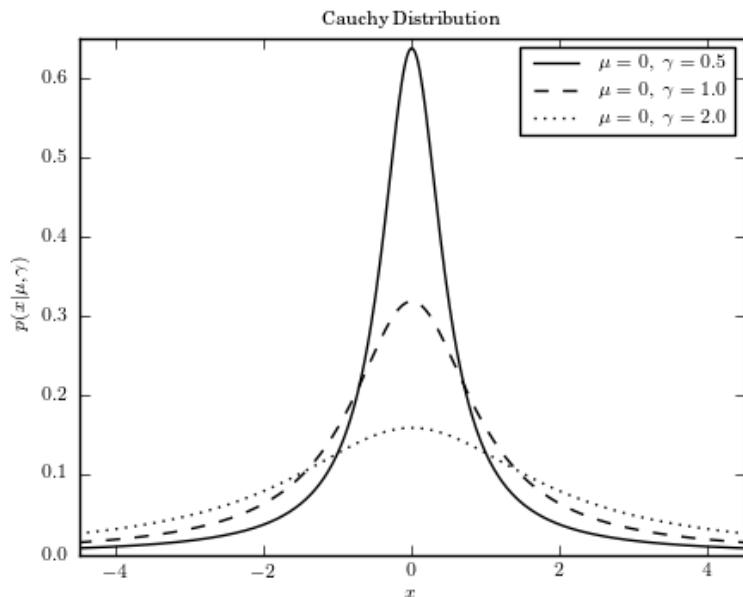
Relation between Poisson, Gaussian, and Binomial Distribution



Source: L. Lyons Data Analysis for Physical Science Students

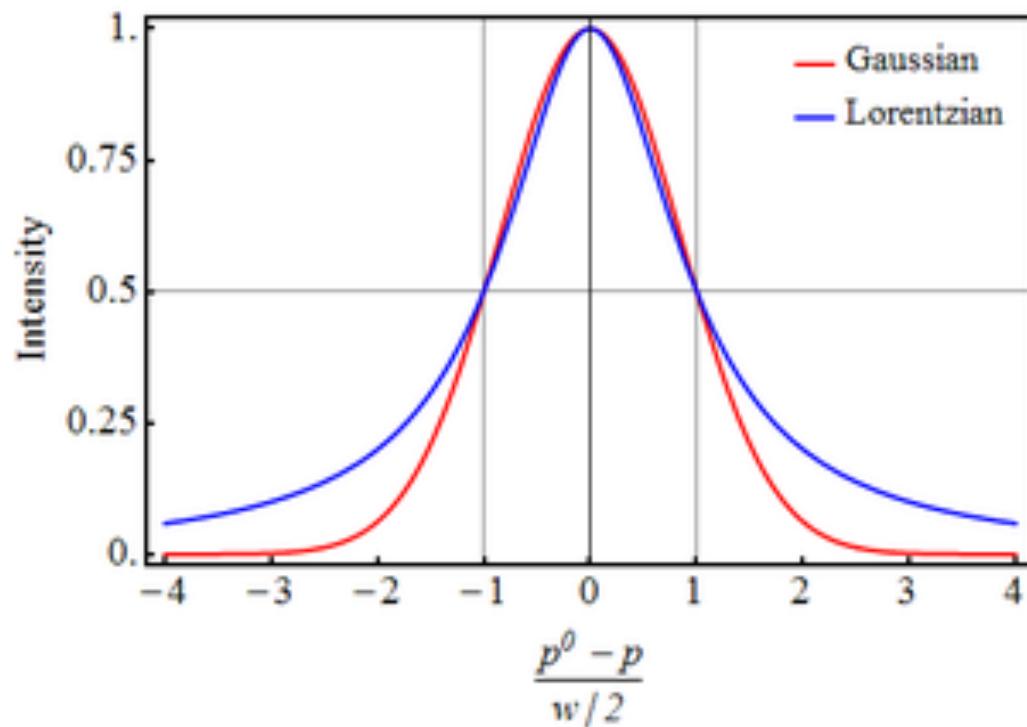
Cauchy (Lorentzian) Distribution

$$p(x|\mu, \gamma) = \frac{1}{\pi\gamma} \left(\frac{\gamma^2}{\gamma^2 + (x - \mu)^2} \right)$$



Cauchy distribution described by location
Parameters μ and scale parameter γ

Exercise : Redo the above plots with $\mu=1$ and $\gamma=4$



Lorentzian line shape function

```
from scipy import stats  
dist = stats.cauchy(0,1)  
r = dist.rvs(10)  
P = dist.pdf(3) # pdf evaluated at x=3
```

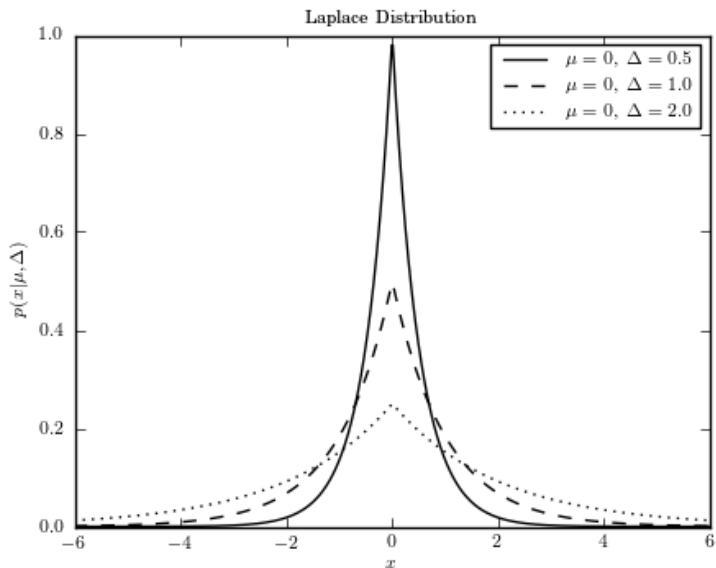
Ratio of two independent standard normal variables $z = (x-\mu)/\sigma$ with z drawn from a Normal Distribution with $\mu=0$ and $\sigma=1$ follows a Cauchy distribution with $\mu=0$ and $\gamma=1$

However, ratio of two random variables drawn from two different Gaussian distributions is more complicated (*follows the Hinkley distribution*)

Mean, variance, and higher order moments **undefined for** Cauchy distribution.

Exponential (Laplace) Distribution

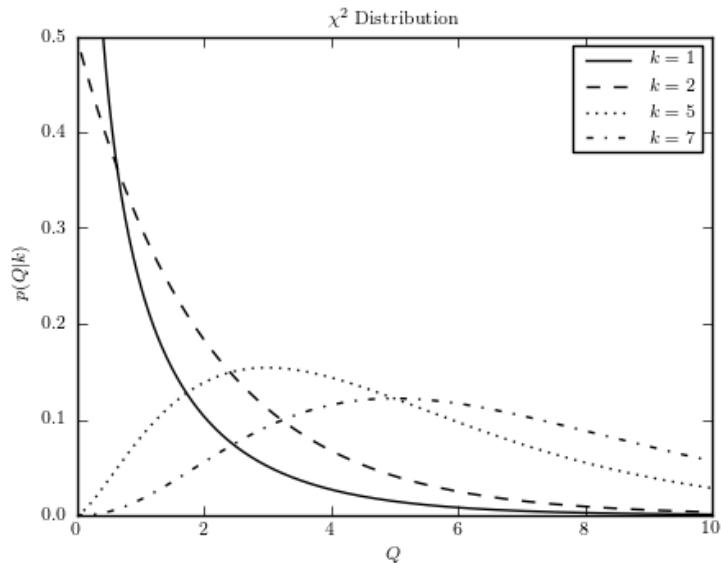
$$p(x|\mu, \Delta) = \frac{1}{2\Delta} \exp\left(-\frac{|x - \mu|}{\Delta}\right)$$



```
from scipy import stats  
dist = stats.laplace(0,0.5)  
r = dist.rvs(10)  
P = dist.pdf(3)
```

Describes time between two successive events which occur continuously and independently at constant Rate

Chi-Square Distribution



```
from scipy import stats  
dist = stats.chi2(5) #k=5  
r = dist.rvs(10) #10 random draws  
P = dist.pdf(3) #evaluated at x=1
```

If $\{x_i\}$ are drawn from a Gaussian distribution and if we define

$$z_i = (x_i - \mu)/\sigma$$

$$Q = \sum_{i=1}^N z_i^2 \quad \text{follows a } \chi^2 \text{ distribution with } k=N \text{ degrees of freedom}$$

$$p(Q|k) \equiv \chi^2(Q|k) = \frac{1}{2^{k/2}\Gamma(k/2)} Q^{k/2-1} \exp(-Q/2)$$

Γ is incomplete Gamma function

EP 4130/PH6130

Second Week

Chi-Square Distribution

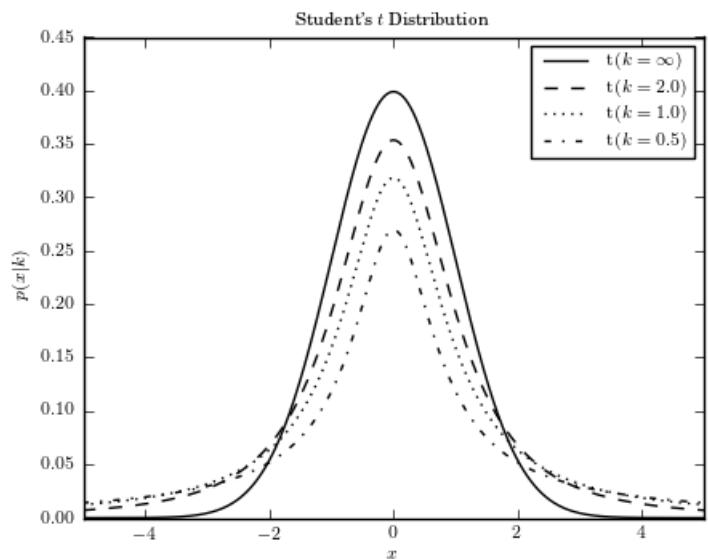
$$p(Q|k) \equiv \chi^2(Q|k) = \frac{1}{2^{k/2}\Gamma(k/2)} Q^{k/2-1} \exp(-Q/2)$$

Sometimes χ^2 distribution per degree of freedom is defined and is given by :

$$\chi_{dof}^2 \equiv \chi^2(Q/k|k)$$

Mean value of χ^2_{dof} is equal to 1. As k increases χ^2_{dof} tends to $\mathcal{N}(1, \sqrt{2/k})$

Student's t distribution



$$p(x|k) = \frac{\Gamma(k + 1/2)}{\sqrt{\pi k} \Gamma(k/2)} \left(1 + \frac{x^2}{k}\right)^{-\frac{k+1}{2}}$$

```
from scipy import stats
dist=stats.t(5) #k=5
r= dist.rvs(10) # 10 random draws
P = dist.pdf(4) #pdf evaluated at x=4
```

- For $k=1$, this distribution is a Cauchy distribution with $\mu=0$ and $\gamma=1$
- Mean, Mode and Median=0 for $k>1$ and undefined for $k=1$

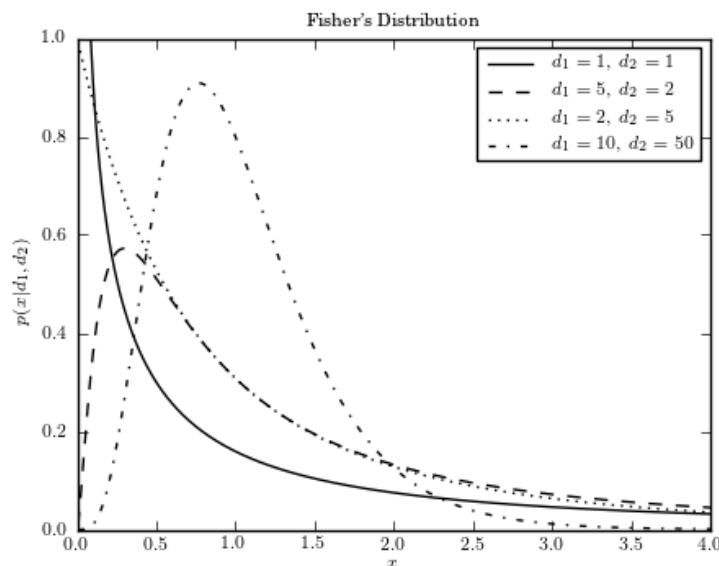
Given a sample of N measurements $\{x_i\}$ drawn from a Gaussian distribution $\mathcal{N}(\mu, \sigma)$

Define
$$t = \frac{\bar{x} - \mu}{s/\sqrt{N}}$$

where \bar{x} and s are the sample mean and sample variance,
follows Student's t-distribution with $k=N-1$ degrees of freedom

- Student's t distribution based on **data** based estimates of mean and std. deviation, whereas Cauchy distribution based on true mean and standard deviation.
- t-distribution stays the same for samples drawn from a Gaussian distribution with different values of mean and standard deviation. as long as number of samples stays the same.
- Ratio of a standard normal variable and one drawn from a χ^2 distribution follows Student's t-distribution.
- t-distribution is used when comparing means of two samples

Fisher's F distribution



Also known as Snedecor distribution, variance ratio Distribution, F-distribution

```
from scipy import stats  
dist=stats.f(2,3) #d1=2,d2=3  
r= dist.rvs(10) # 10 random draws  
P = dist.pdf(1) #pdf evaluated at x=1
```

$$p(x|d_1, d_2) = C \left(1 + \frac{d_1}{d_2}x\right)^{-\frac{d_1+d_2}{2}} x^{\frac{d_1}{2}-1}$$

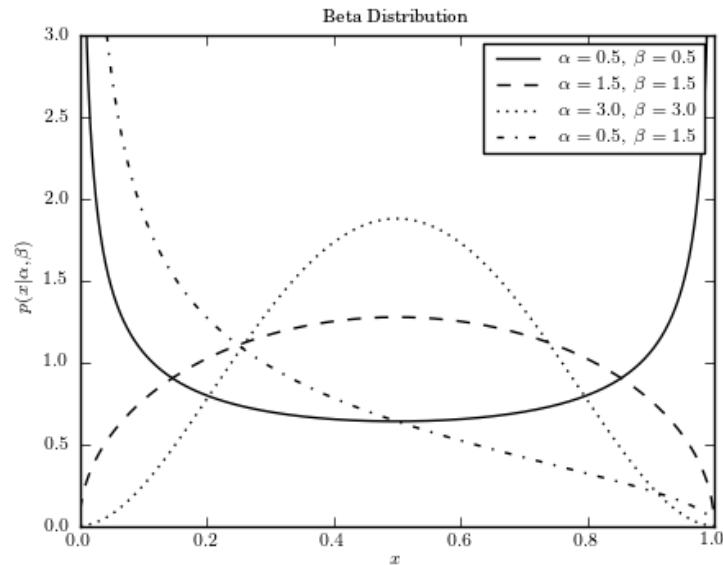
where C is given by

$$C = \frac{1}{B(d_1/2, d_2/2)} \left(\frac{d_1}{d_2}\right)^{d_1/2}$$

B is the Beta function

- Fisher F distribution describes the ratio of two independent χ^2 variables with d_1 and d_2 degrees of freedom.
- If x_1 and x_2 are two independent random variables drawn from the Cauchy distribution with Location parameter μ , then the ratio $|x_1 - \mu|/|x_2 - \mu|$ follows Fisher's F distribution with $d_1 = d_2 = 2$.

Beta Distribution

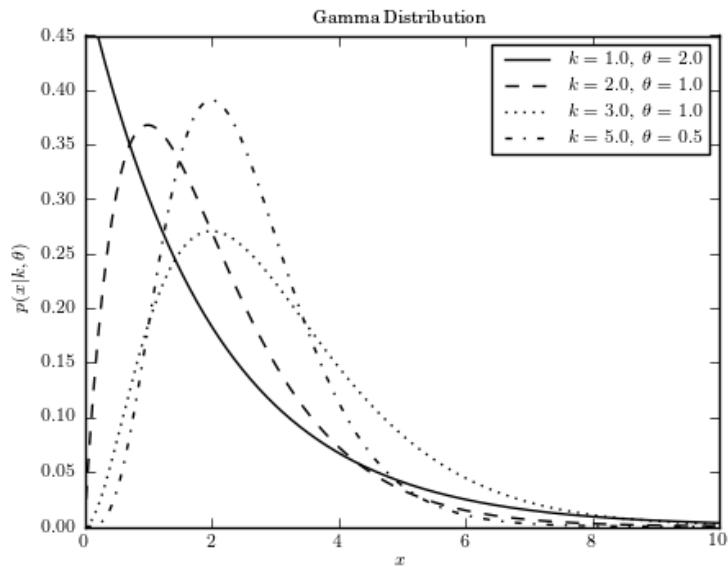


```
from scipy import stats  
dist=stats.beta(0.5,1.5) #alpha=0.5, beta=1.5  
r= dist.rvs(10) # 10 random draws  
P = dist.pdf(0.6) #pdf evaluated at x=0.6
```

$$p(x|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1} (1-x)^{\beta-1}$$

- Mean value of beta distribution is $\alpha/(\alpha+\beta)$
- Beta distribution is conjugate prior of the binomial distribution

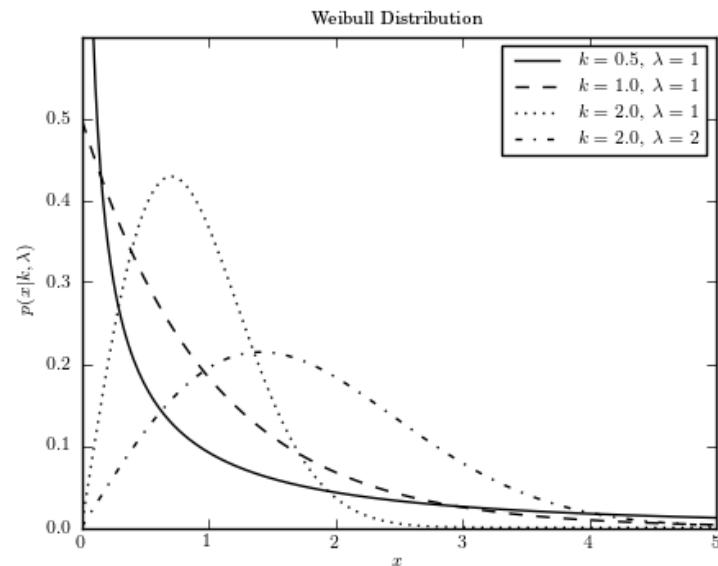
Gamma Distribution



```
from scipy import stats  
dist=stats.gamma(1,0,2) #k=1, loc=0, theta=2  
r= dist.rvs(10) # 10 random draws  
P = dist.pdf(1) #pdf evaluated at x=1
```

- Gamma Function is a conjugate prior to several distributions such as the Laplace distribution and the Poisson distribution

Weibull Distribution



```
from scipy import stats  
dist=stats.dweibull(1,0,2) #k=1, loc=0,lambda=2  
r= dist.rvs(10) # 10 random draws  
P = dist.pdf(1) #pdf evaluated at x=1
```

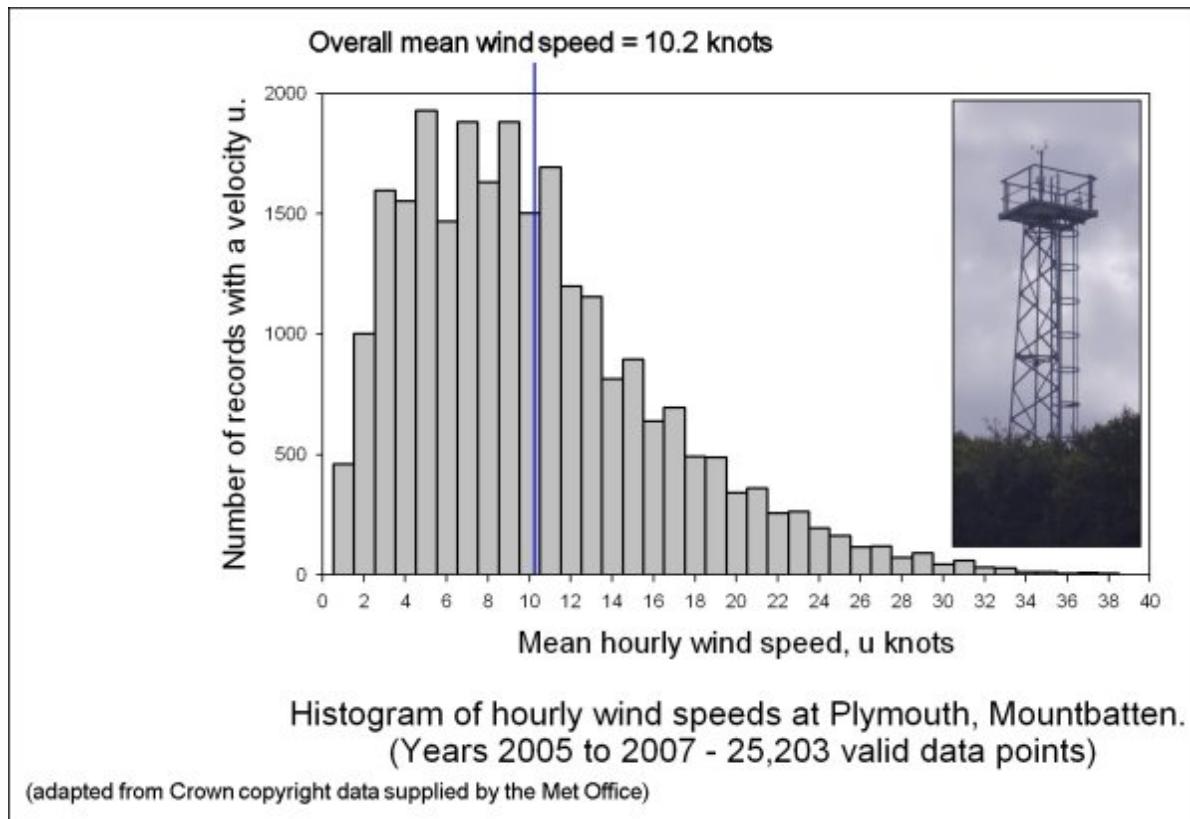
$$p(x|k, \lambda) = \frac{k}{\lambda} \left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$$

Weibull distribution describes good description of random failure process with variable rate, wind speeds, distribution of extreme values, and size distribution of particles., overvoltage in an Electrical system.

eg:

- If x is the time to failure for a device with a failure rate proportional to time t^m , x follows the Weibull distribution with $k=m+1$
- Wind speed follows Weibull distribution with $k \sim 2$

Example of Weibull Distribution



1 knot = 0.52 m/sec

Typo in astroML book

- Typo in Weibull distribution pdf in AstroML book. What is plotted is positive part of double Weibull distribution (`dweibull`). PDF of Weibull distribution (defined in class) is given by twice this value. Alternatively, use `stats.weibull_min`

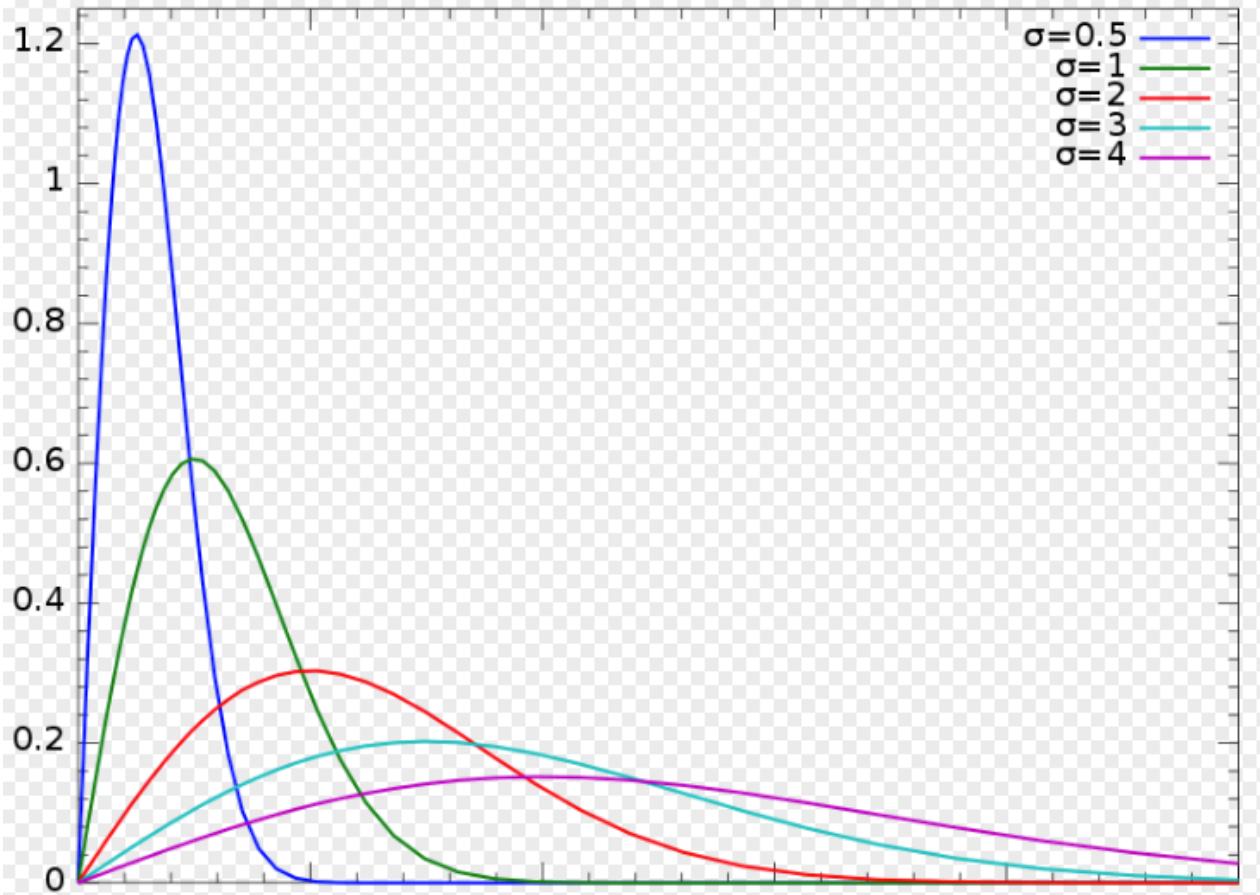
Rayleigh Distribution (not discussed in astroML)

$$f(x, \sigma) = \frac{x}{\sigma^2} e^{-x^2/2\sigma^2} \quad x \geq 0$$

Cumulative distribution function is given by $F(x, \sigma) = 1 - e^{-x^2/2\sigma^2}$

Equal to Chi-square distribution for two degrees of freedom

Observed when overall magnitude of a vector is related to its directional Components. eg. Wind velocity in two dimensions



Examples

```
>>> from scipy.stats import rayleigh  
>>> import matplotlib.pyplot as plt  
>>> fig, ax = plt.subplots(1, 1)
```

>>>

Example of Rayleigh Distribution in astrophysics

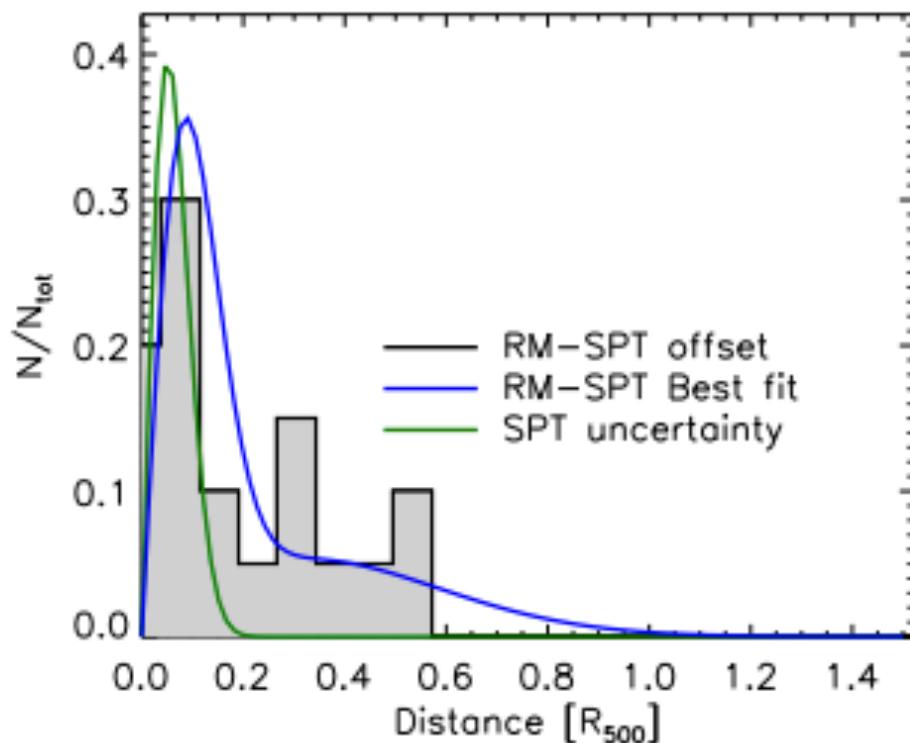


Figure 7. Solid histogram shows the measured fraction of SPT-SZ+RM clusters as a function of the optical-SZE positional offset in units of R_{500} . The green curve shows the SPT-SZ positional uncertainty, and the blue curves shows the best fitting SZE-optical positional offset model.

$$P(x) = 2\pi x \left(\frac{\rho_0}{2\pi\sigma_0^2} e^{-\frac{x^2}{2\sigma_0^2}} + \frac{1-\rho_0}{2\pi\sigma_1^2} e^{-\frac{x^2}{2\sigma_1^2}} \right) \quad (12)$$

Mixture of two Rayleigh distributions

arXiv:1506.07814

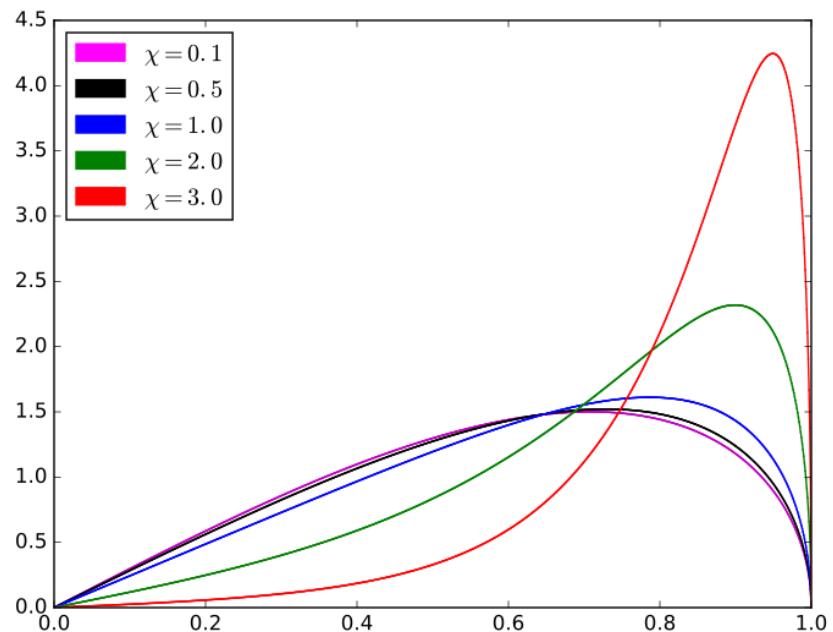
Other distributions in scipy can be found at

<https://docs.scipy.org/doc/scipy-0.18.1/reference/stats.html>

Other potential distributions among these : Rice, Pareto, Power-Law,

Argus Distribution

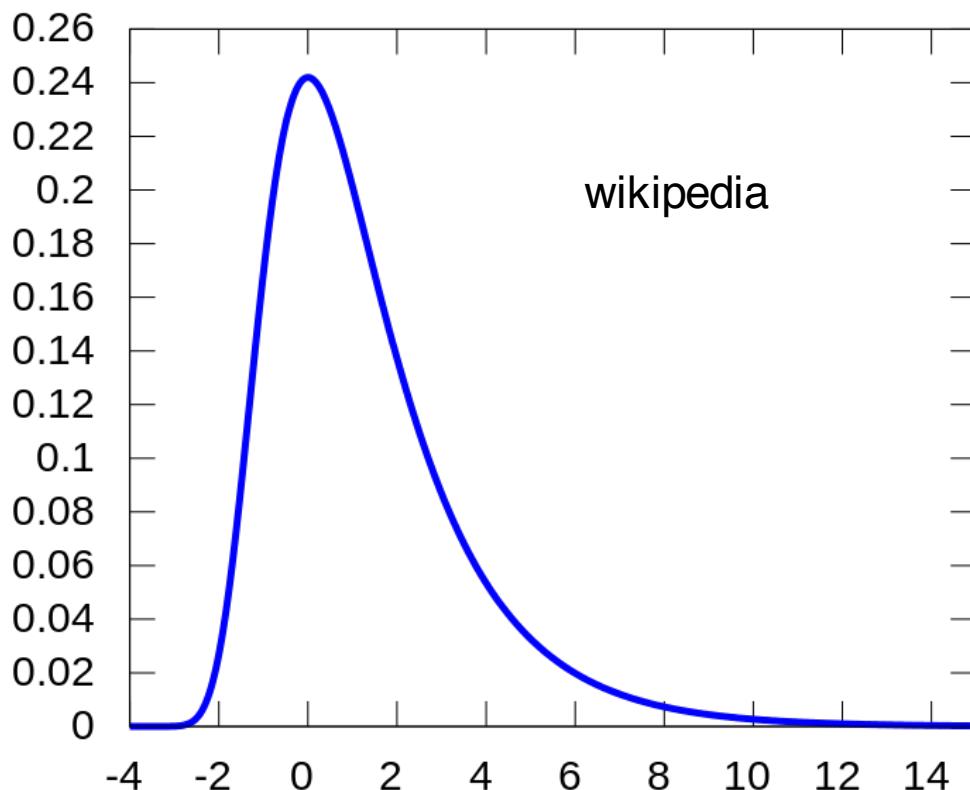
Probability distribution named after the particle physics experiment ARGUS is the reconstructed invariant mass of a decayed particle candidate in a continuum background.



Look up `scipy.stats.argus`

wikipedia

Landau Distribution



Not in stats.scipy

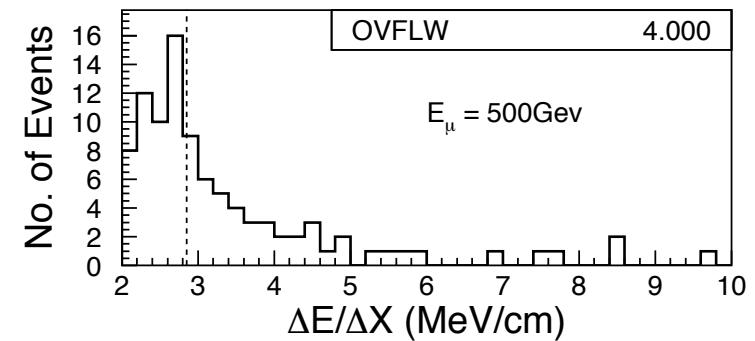
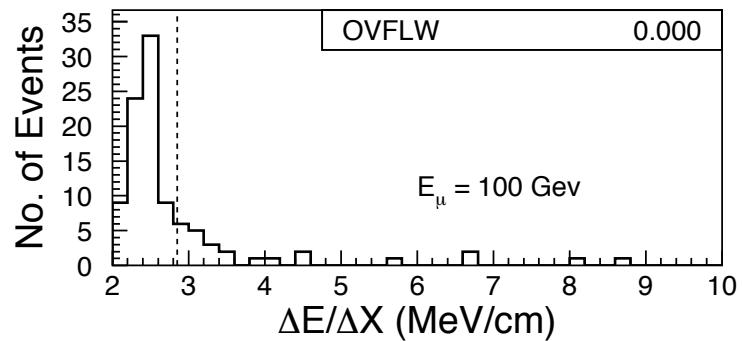
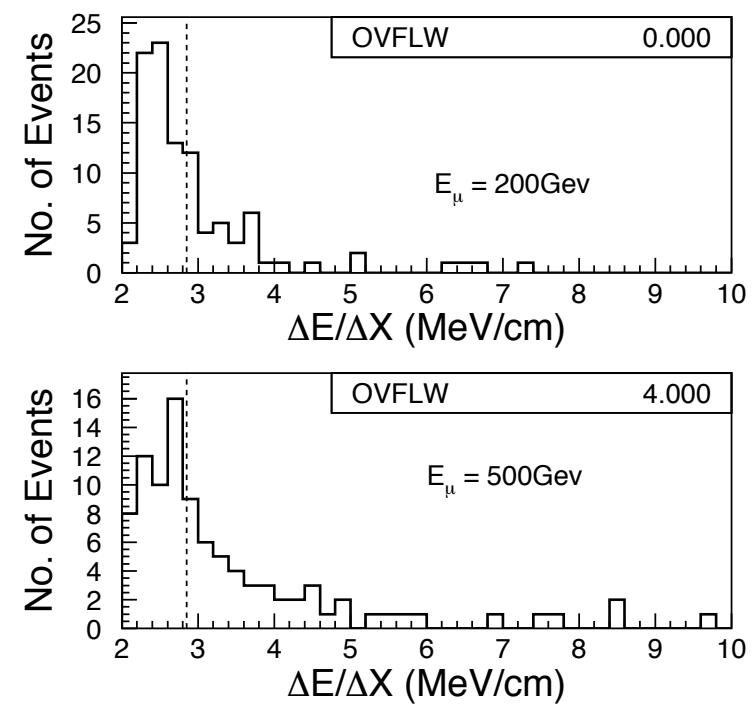
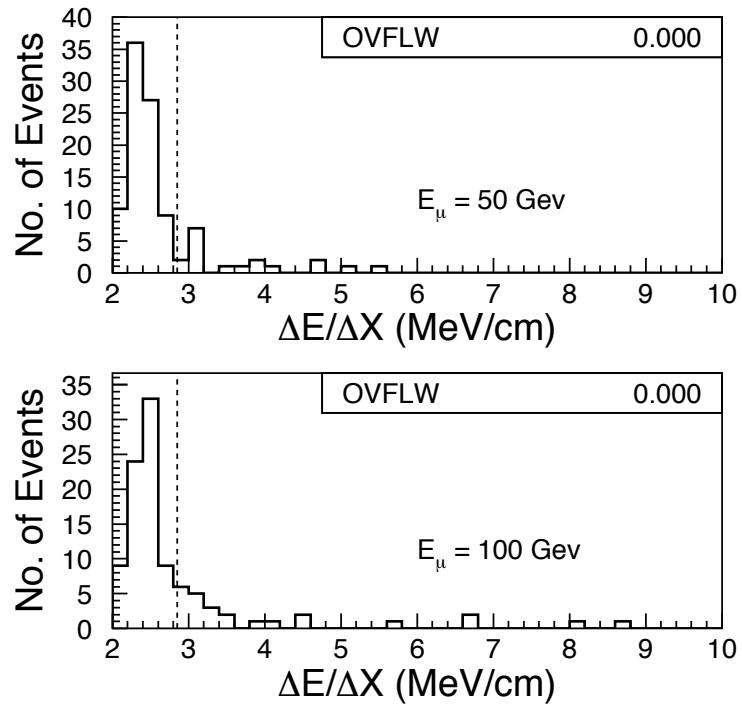
No analytic formulae for pdf

Mean and variance are undefined because of very long tail

Python function (pyLandau) can be downloaded from the web

Scipy.stats.moyal provides a good approximation to Landau distribution.

Example of Landau Distribution

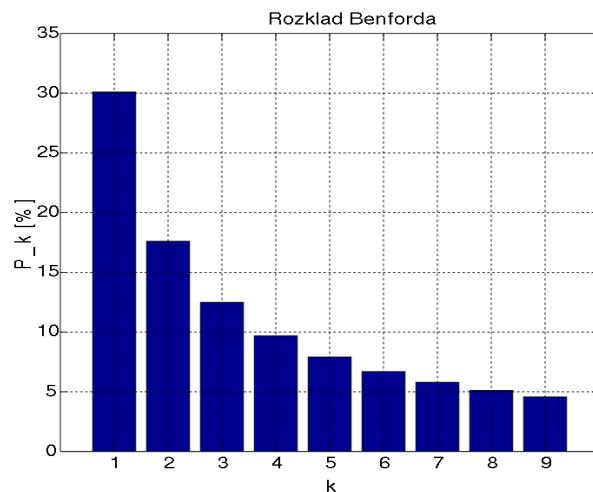


Simulations of Muon Energy loss in water for various mono-energetic muon samples (SD thesis)

Benford Distribution

- Benford's law is a law about frequency of leading digits
- A set of numbers is said to satisfy Benford's law if the leading digit occurs with probability given by

$$P(d) = \log_{10}(d + 1) - \log_{10}(d) = \log_{10}\left(\frac{d + 1}{d}\right) = \log_{10}\left(1 + \frac{1}{d}\right)$$



Examples of Benford Law

- Surface Areas of Rivers
- Physical Constants
- Molecular Weights
- Street Addresses of people listed in American Men of Science
- Heights of Tallest structures in world
- Telephone Bills etc (see wikipedia)

Has been Used to detect bank & election fraud and election rigging

For more details, please read

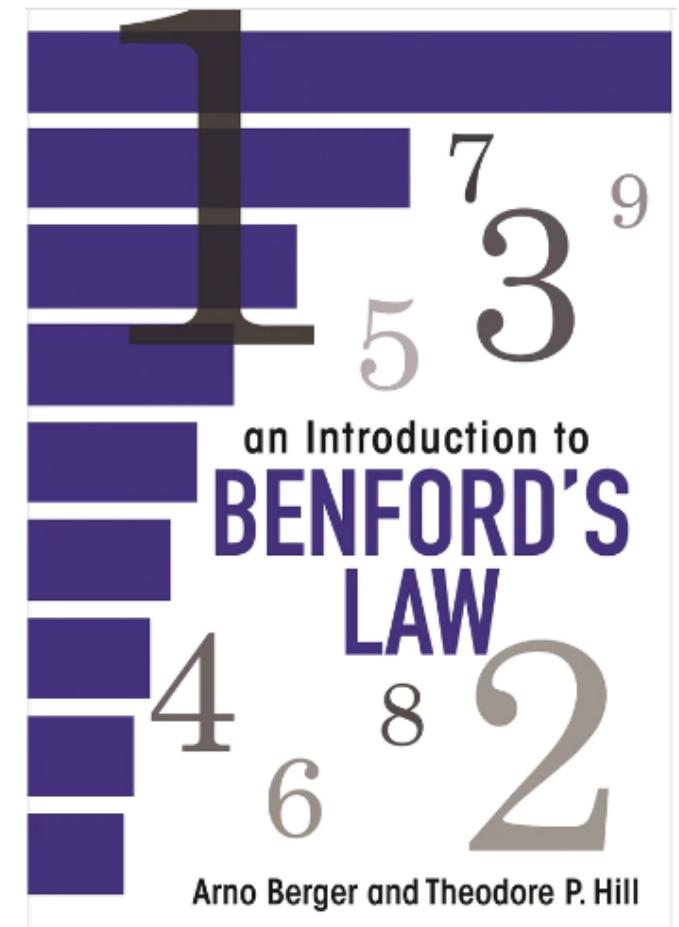
arxiv:2008.12271 and

arXiv:1709.09823 by A. Dantuluri & SD

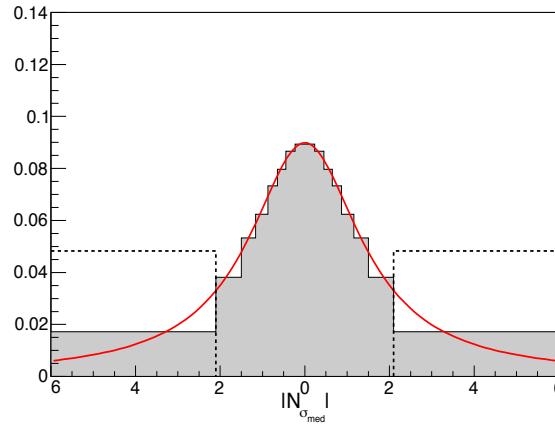
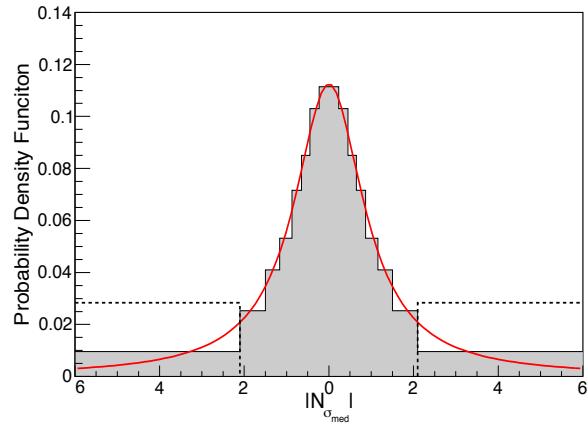
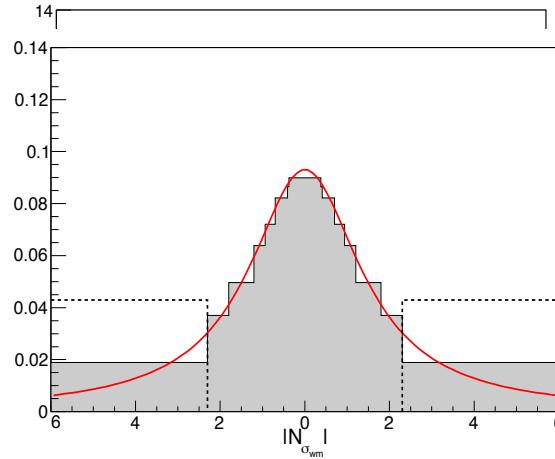
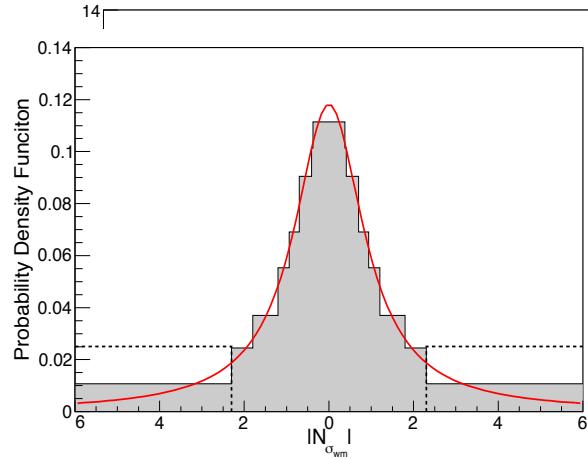
arXiv:2207.09696 by P. Mamidipaka & SD

Also a whole book on Benford's law

HW : do google search on COVID-19 & Benford's law



Primordial Lithium-7 Measurements



tom) row uses the weighted mean (median) of the 66 measurements as the central estimate. Fig. 3. left (right) Galropy probability density distributions. That top left (right) plot corresponds to positive (negative) N_σ represent a value that is greater (less) than the central estimate.

Crandall , Houston
and Ratra
arXiv:1409.7332

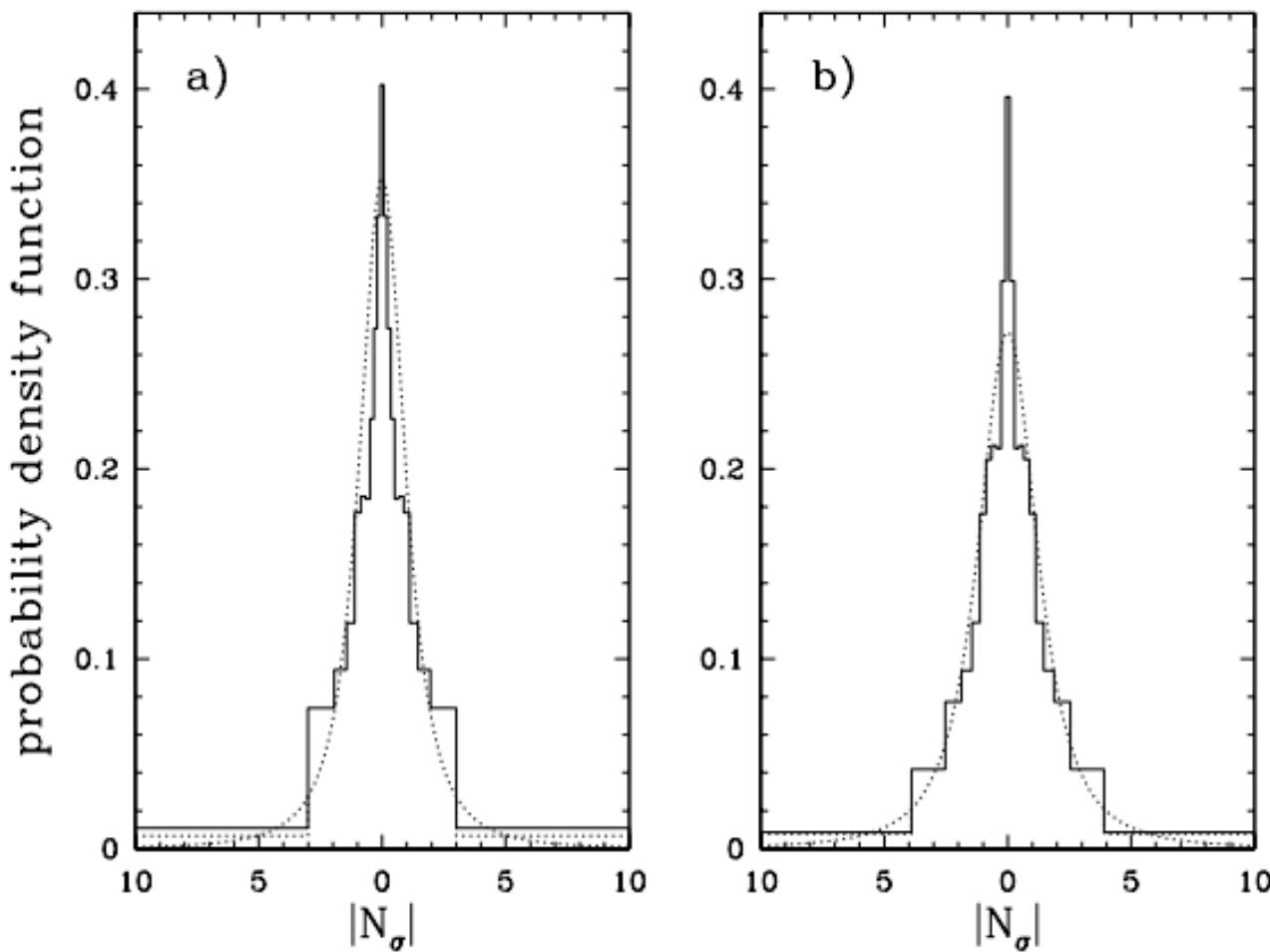


Fig. 4.— Binned data (solid lines) and best-fit $n = 2$ Student's t probability distribution functions (dotted lines) for $H_0 = 71 \text{ km s}^{-1} \text{ Mpc}^{-1}$ estimated by the WMAP collaboration, all normalized to unit area. See Fig. 2 caption for more details. Left panel *a)* shows a distribution with scale

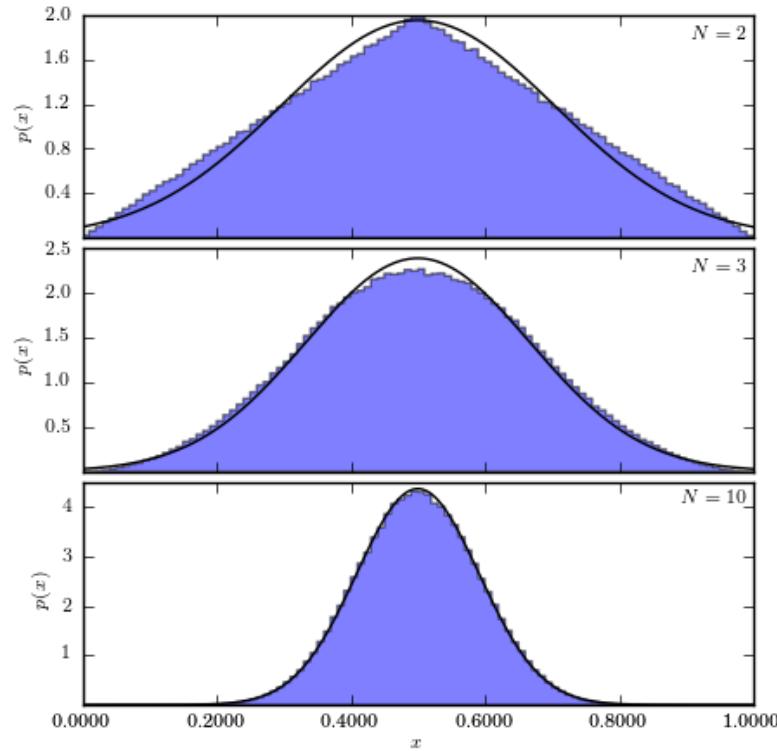
Error Distributions for Hubble
Constant measurements
Fit to students-t distribution

astro-ph/0308099

Central Limit Theorem

- Given an arbitrary distribution $h(x)$ characterized by its mean μ and standard deviation σ , mean of N values x_i drawn from that distribution will approximately follow a Gaussian distribution $\mathcal{N}(\mu, \sigma/\sqrt{N})$ with the accuracy increasing with increasing N .
- $h(x)$ must have a standard deviation and therefore its tails must fall off faster than $1/x^2$ for large x
- Central limit theorem can be derived using method of characteristic functions (arXiv:0712.3028) or repeated convolutions (Gregory 2005).
- Central limit theorem does not apply to Cauchy distributions, as it does not have a well-defined mean and standard deviation.
- Sample Mean converges to distribution mean as sample size increases. This is called weak law of Large Numbers.

Illustration of Central Limit Theorem



Histogram of N random variables (from uniform distribution) drawn from 0 to 1

Bivariate Gaussian Distributions

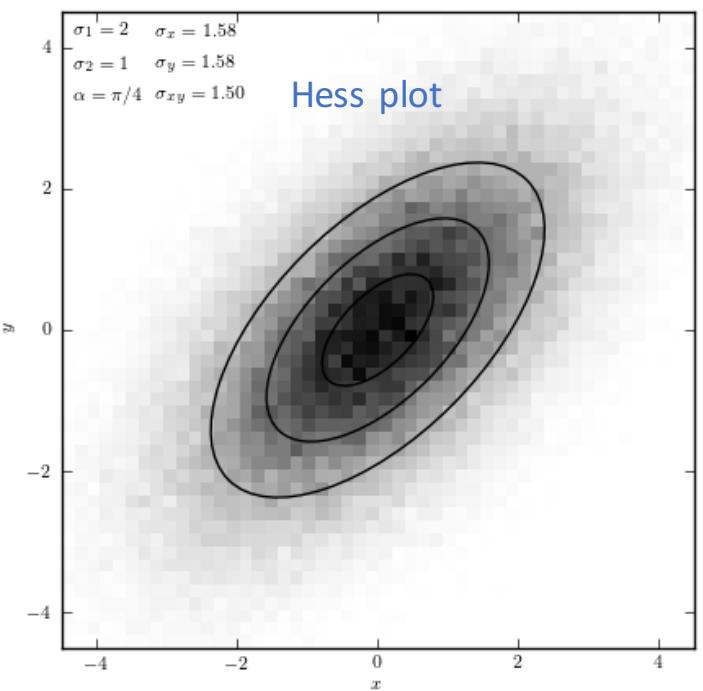
$$p(x, y | \mu_x, \mu_y, \sigma_x, \sigma_y, \sigma_{xy}) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{z^2}{2(1-\rho^2)}\right)$$

where
$$z^2 = \frac{(x - \mu_x)^2}{\sigma_x^2} + \frac{(y - \mu_y)^2}{\sigma_y^2} - 2\rho \frac{(x - \mu_x)(y - \mu_y)}{\sigma_x\sigma_y}$$

$$\rho = \frac{\sigma_{xy}}{\sigma_x\sigma_y}$$

ρ is called population correlation coefficient.
For correlated variables $\rho \sim 1$. For
uncorrelated variables $\rho = 0$

Data from Bivariate Gaussian Distributions



Ellipses centered on (μ_x, μ_y)
Angle between X-axis and ellipse major axis
is given by :

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

Correlation between x and y can be eliminated by Principal axis transformation. Correlation between two variables disappears once we rotate the vectors to P1 and P2

$$P_1 = (x - \mu_x) \cos \alpha + (y - \mu_y) \sin \alpha$$

$$P_2 = -(x - \mu_x) \sin \alpha + (y - \mu_y) \cos \alpha$$

Multivariate Gaussian Distributions

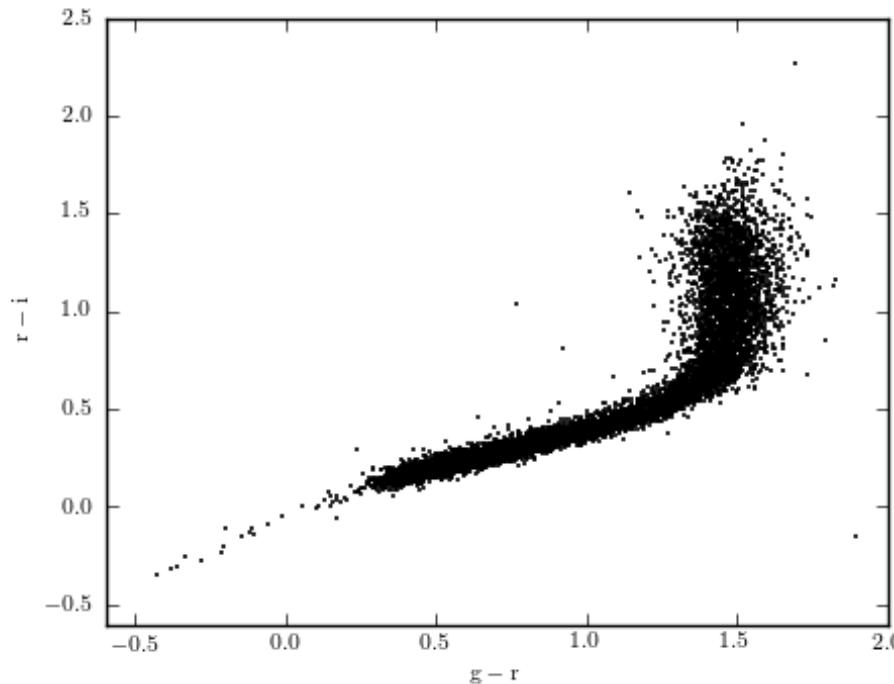
```
import numpy as np  
mu=[1,2]  
Cov = [[1,0.2],[0.2,0.3]]  
np.random.multivariate_normal(mu,cov)
```

$$p(x|I) = \frac{1}{(2\pi)^{M/2} \sqrt{\det(C)}} \exp\left(-\frac{1}{2}x^T H x\right)$$

where x is vector. C is covariance matrix. H is inverse of covariance matrix also called Hessian

$$C_{kj} = \int_{-\infty}^{+\infty} x^k x^j p(x|I) d^M x \quad x^T H x = \sum_{k=1}^M \sum_{j=1}^M H_{kj} x^k x^j$$

Data visualization techniques



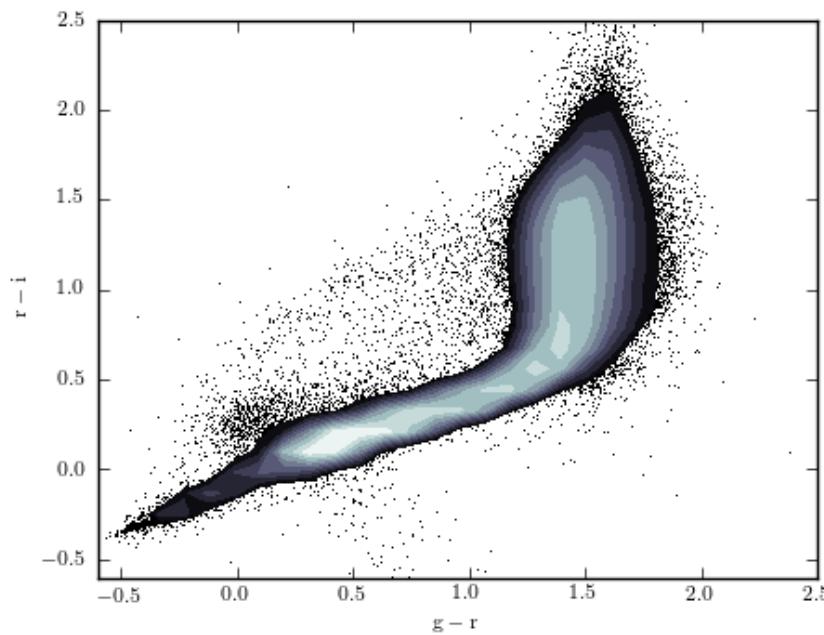
Simple Scatter plots

For more details, see *The Visual Display of Quantitative Information* by Tufte

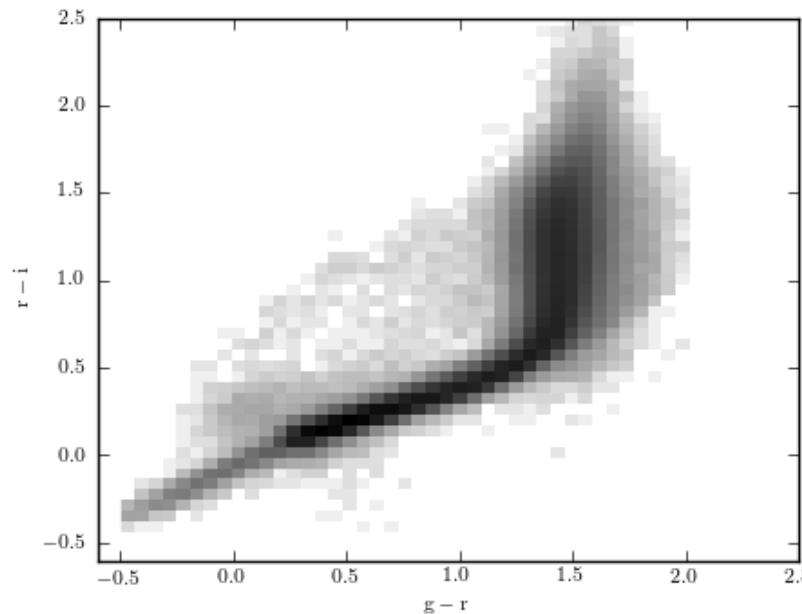
Data visualization techniques

If density of two points too high, no longer practical to use ordinary ‘scatter plots’. Need to switch to Contour plots. However this loses information in case of only a few points.

→ Use contours for high density region and individual points in low density region (used extensively in SDSS)

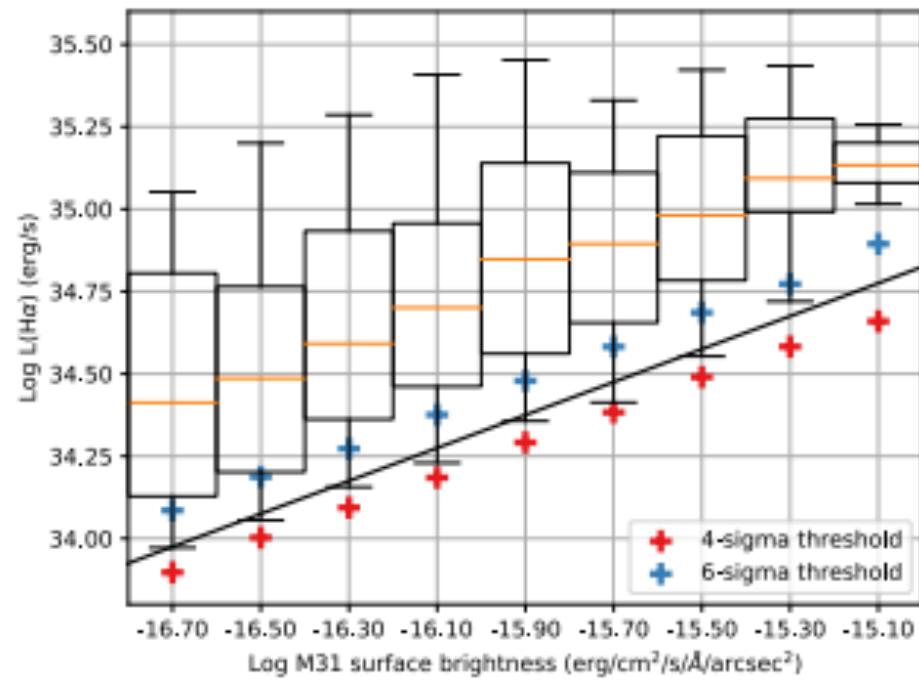
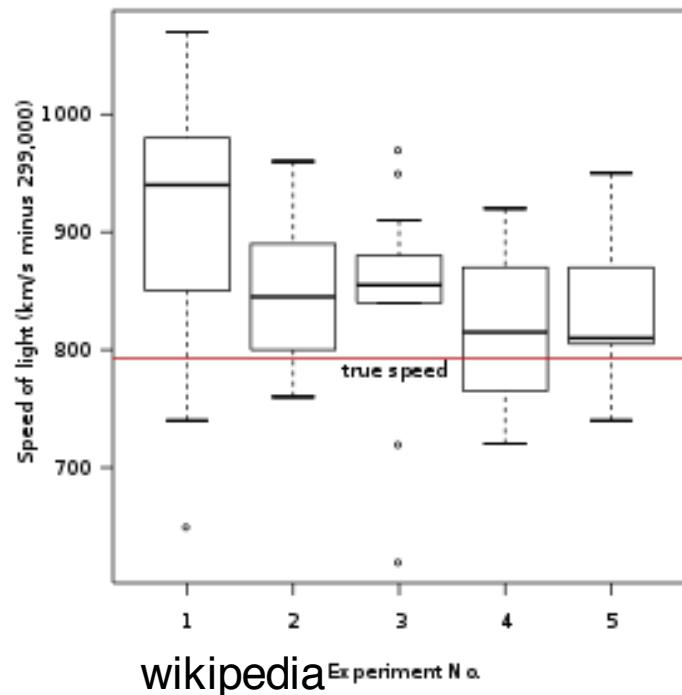


Alternatively pixelize the plotted diagram and display the counts of points in each pixel.
Known as Hess-diagram



For higher dimensions, see section 1.6.2 of astroML book or source code of Figure 1.11 and 1.12

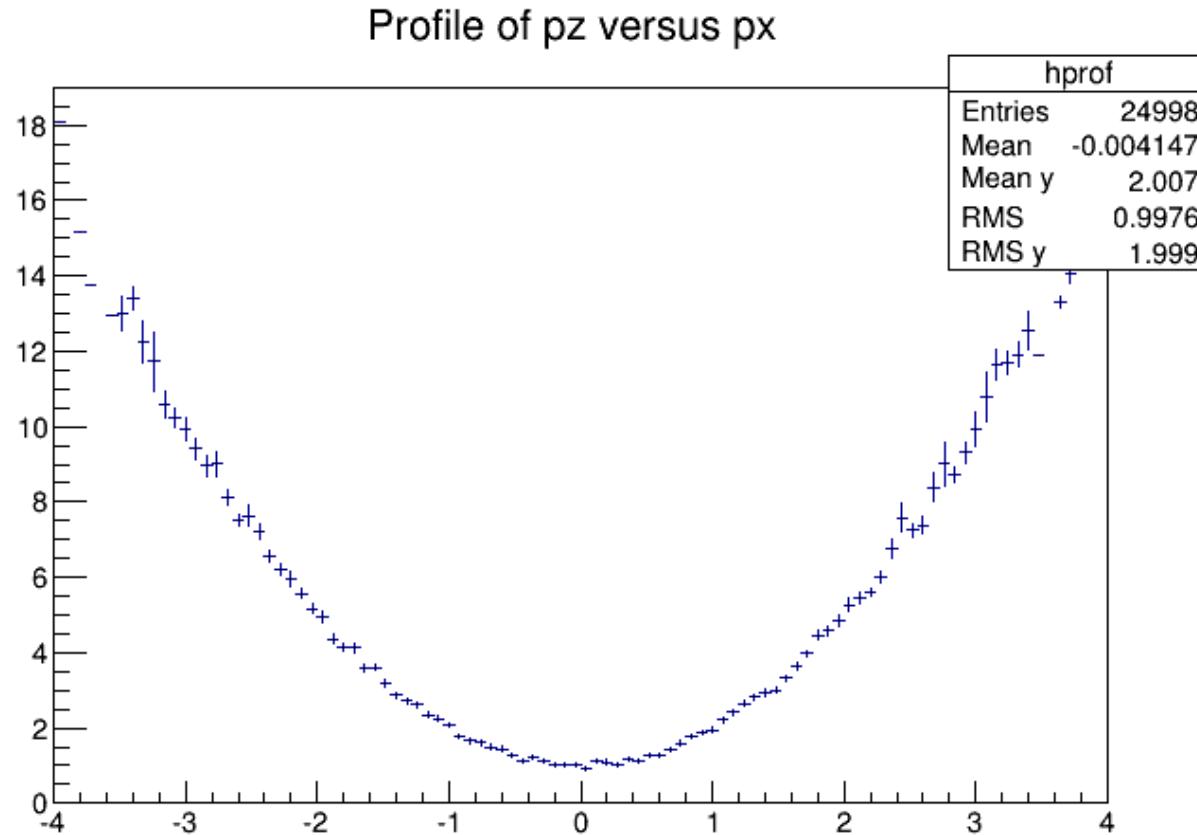
Box and Whisker Plots



arxiv:1707.01366

Uses quartiles to represent data. In high energy physics a variant of this called profile histogram is used.

Profile Histograms (used in High energy Physics)



A profile histogram example

Not inbuilt in python. Only inbuilt in ROOT programming language. However see
<https://stackoverflow.com/questions/23709403/plotting-profile-histograms-in-python>

How to test if two datasets are correlated?

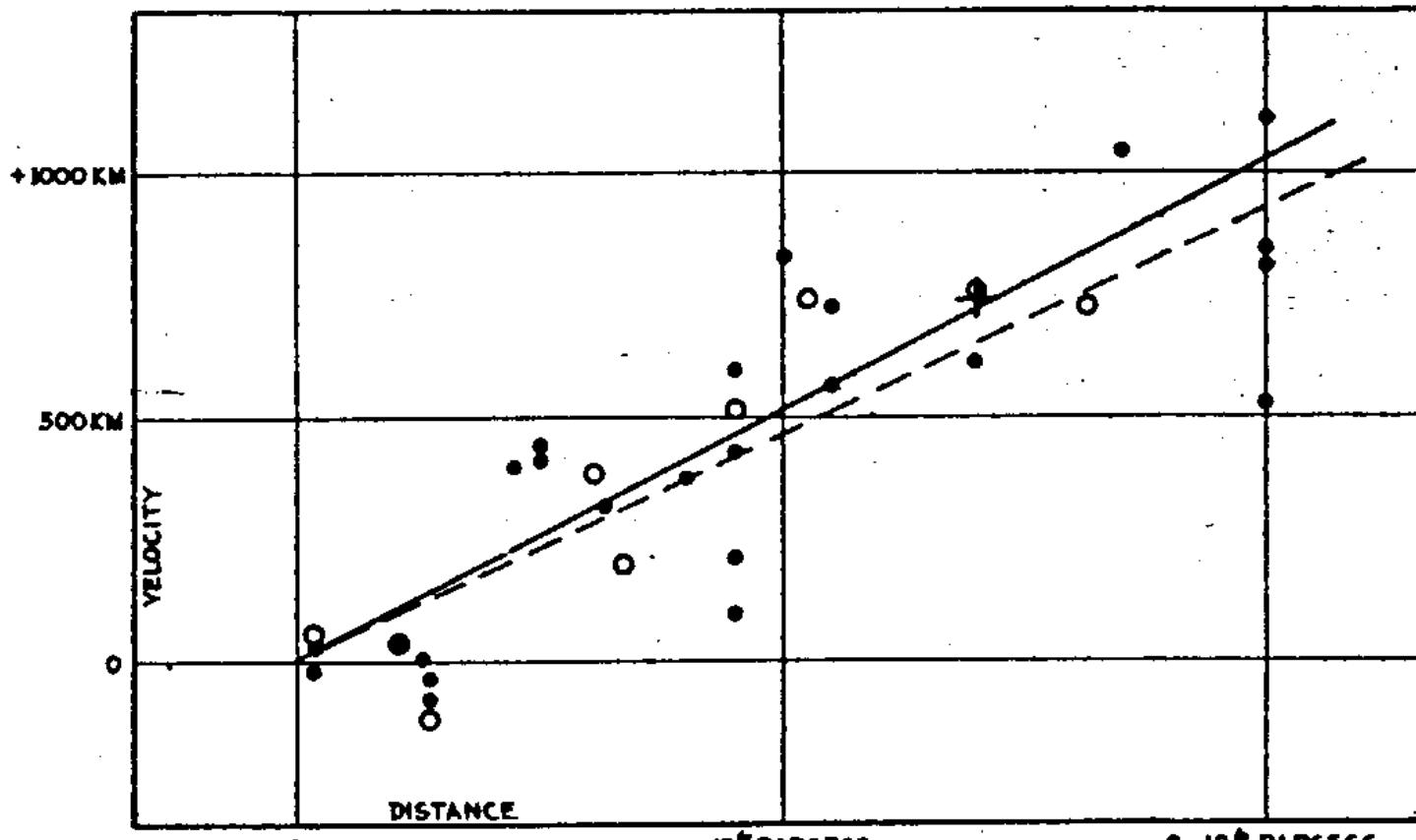
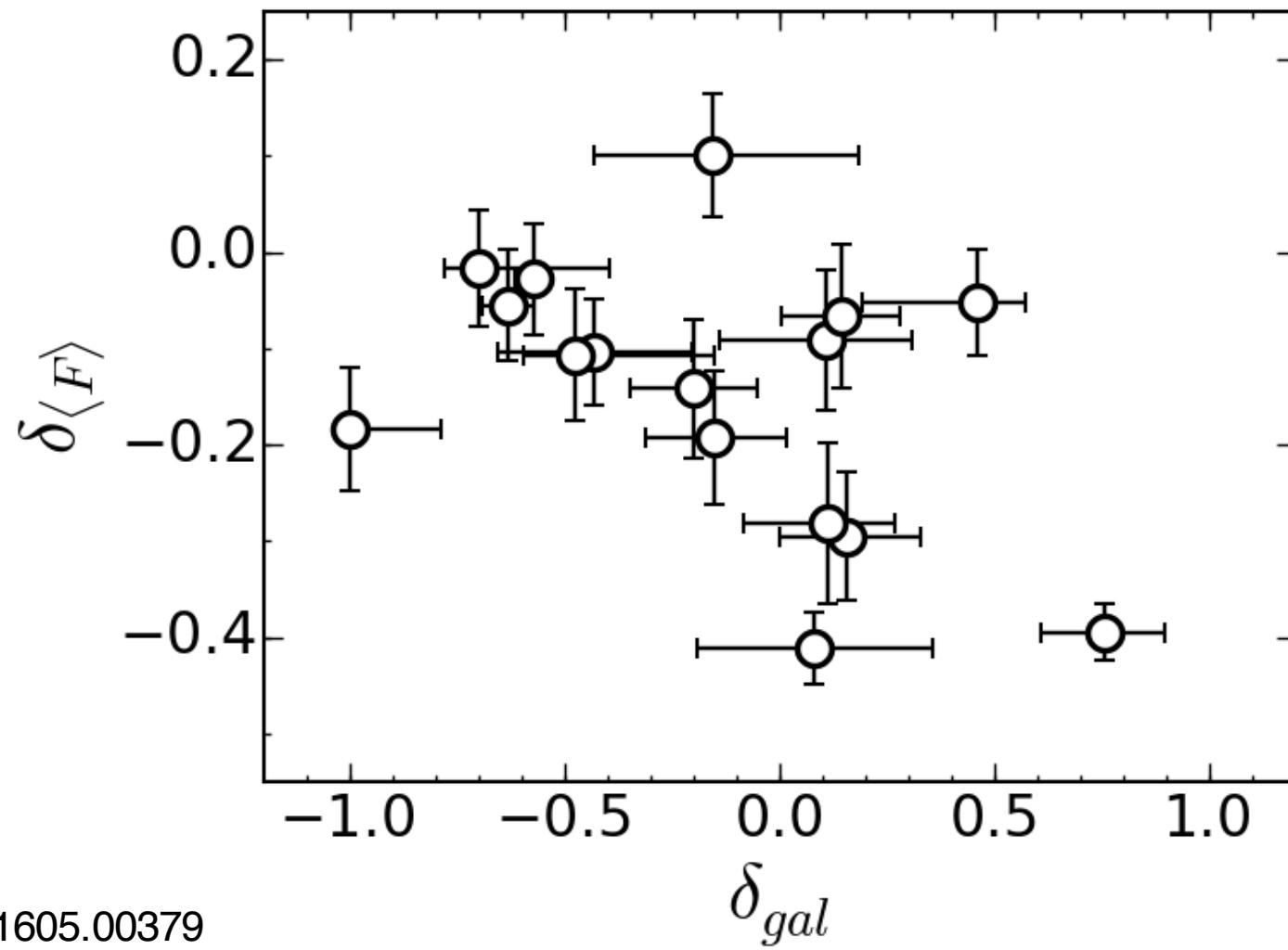


FIGURE 1

Hubble 1929 data
showing
Expansion of
universe



Correlation Coefficients

Q_t : How to quantitatively assess if two datasets {x_i} and {y_i} of size N are correlated

Introduce three such terms : Pearson's sample correlation coeff. , Spearman rank correlation coeff., and Kendall tau

```
from scipy import stats
x,y = np.random.random((2,100)) # two random arrays
corr_coeff,p_value = stats.pearsonr(x,y)
rho,p_value = stats.spearmanr(x,y)
tau,p_value = stats.kendalltau(x,y)
```

P –value indicates the probability of an uncorrelated system producing datasets that have a correlation coefficient at least as extreme as the one computed from these datasets

Pearson Correlation Coefficient

$$r = \frac{\sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^N (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^N (y_i - \bar{y})^2}}$$

$-1 \leq r \leq 1$ For uncorrelated datasets $r = 0$

If the pairs (x_i, y_i) are drawn from two uncorrelated univariate Gaussian distributions (i.e. the population correlation coefficient $\rho=0$), distribution of r follows Students t-distribution with $k=N-2$ degrees of freedom and t given by

$$t = r \sqrt{\frac{N-2}{1-r^2}}$$

Example:

N=10 r=0.72

Probability of getting a value as large as the observed value of r which is 0.72 (by chance from a random Fluctuation) is 1%

Or one-sided 99% confidence level for Student's t-distribution with k=8 DOF is t=2.93

```
>>> import numpy as np  
>>> x=0.72*np.sqrt(8/(1-0.72*0.72))  
>>> x  
2.9345009253812004  
>>> from scipy.stats import t  
>>> t.cdf(2.934,8)
```

Cumulative Distribution function of t-distribution

0.99253

- For bivariate Gaussian distribution with a non-vanishing population correlation coefficient ρ , the **Fisher transformation (or Fisher z-transformation)** can be used to estimate the confidence interval for ρ from the measured value of r . The distribution of F is given by:

$$F(r) = \frac{1}{2} \ln \left(\frac{1+r}{1-r} \right)$$

approximately follows a Gaussian distribution with mean $\mu = F(\rho)$ and a standard deviation given by $\sigma_F = (N-3)^{-1/2}$

Advantages/Disadvantages of Pearson Correlation Coefficient

- r does not take into account errors in x_i and y_i
- r is very sensitive to Gaussian outliers

Spearman Rank Correlation Coefficient

Spearman Correlation Coefficient (r_s) based on the concept of **ranks** (sort the data in ascending order and the index i of a value x_i in the sorted data is its rank R_i^x)

$$r_s = \frac{\sum_{i=1}^N (R_i^X - \bar{R}^X)(R_i^Y - \bar{R}^Y)}{\sqrt{\sum_{i=1}^N (R_i^X - \bar{R}^X)^2} \sqrt{\sum_{i=1}^N (R_i^Y - \bar{R}^Y)^2}}$$

Same as Pearson correlation coefficients for the ranks. Distribution of Spearman correlation coeff. (for null hypothesis) is same as Pearson correlation coefficient

Alternate definition:
(Lupton 93)

$$r_s = 1 - \frac{6}{N(N^2 - 1)} \sum_{i=1}^N (R_i^X - R_i^Y)^2$$

Kendall Tau Correlation Coefficient

- Rank the two datasets. Count the number of *concordant* pairs, defined by $(x_j - x_k)(y_j - y_k) > 0$ and *discordant* pairs defined by $(x_j - x_k)(y_j - y_k) < 0$

For a perfect correlation (anti-correlation) all possible $N(N-1)/2$ pairs will be concordant (discordant).

Kendall's tau is defined as :

$$\tau = 2 \frac{N_c - N_d}{N(N-1)} \quad -1 \leq \tau \leq 1$$

- When $N > 10$ distribution of Kendall tau in case of no-correlation (null hypothesis) can be approximated as a Gaussian distribution with $\mu=0$ and width given by

$$\sigma_\tau = \left[\frac{2(2N+5)}{9N(N-1)} \right]^{1/2}$$

- True distributions of Spearman and Pearson correlation coefficient harder to represent analytically in case of a true correlation for a general case
- For a bi-variate Gaussian distribution with true correlation coefficient ρ expectation value for Kendall's tau is given by ([arXiv:1011.2009](#))

$$\bar{\tau} = \frac{2}{\rho} \sin^{-1}(\rho)$$

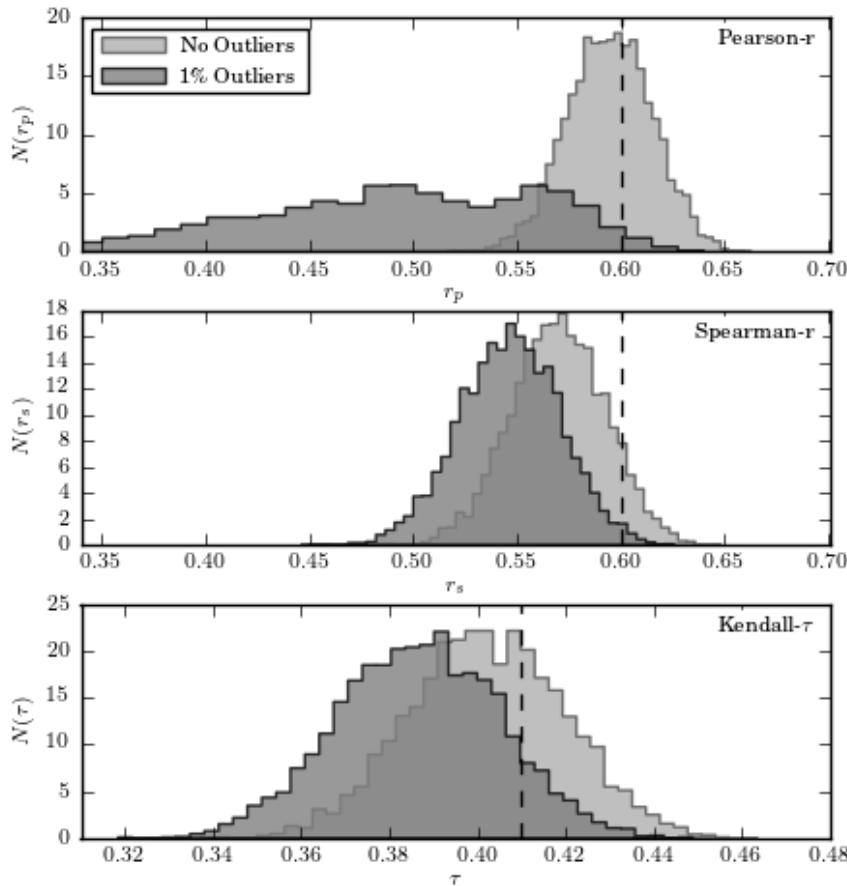
.

- Kendall's co-relation coefficient approaches normality faster than Spearman's correlation coefficient
- Usually, a bootstrap method is used to place confidence estimates on the measured values

Advantages of Kendall tau over Spearman r correlation coeff.

- Kendall tau correlation coefficient approaches normality faster than Spearman correlation coefficient
- Kendall tau offers a more unbiased estimate of population value while Spearman correlation coeff. does not. (Lupton 1993)
- Efficiency of Kendall tau relative to Pearson correlation coefficient for a bi-variate Gaussian greater than 90% and much greater than non-Gaussian distributions.

Distribution of correlation coefficient



2000 bootstrap resamples of 1000 datapoints drawn from a bivariate Gaussian with $\rho=0.6$ without and with (1%) outliers

Pearson correlation coefficient not robust against outliers.

Spearman and Kendall correlation coefficient have variance which is robust to outliers

Random Number Generation

Distributions in `scipy.stats.distributions` have a method called `rvs` which generates pseudo-random sample from the distribution.

Also `numpy.random` implements samplers for a number of distributions (look up `numpy.random` documentation)
(behind the scenes, `scipy` calls `numpy`)

Select 5 random integers between 5 and 10:

```
>>> import numpy as np  
>>> np.random.random_integers(0,10,5)  
 array([10,  4,  0,  8,  2])
```

Numerical simulations of measurement processes are called Monte Carlo simulations and the resulting samples are called Monte Carlo or mock samples.

See discussion in Numerical Recipes on how to generate random number distributions

Parameter Estimation to bivariate Gaussian distribution

- For 1-D Gaussian mean and standard deviation can be estimated from sample mean and sample variance . This can be extended to 2-D
- Correlation coefficient (ρ) can be estimated using Pearson's sample correlation coefficient. Principal axes can be found with a estimated using

$$\tan(2\alpha) = 2 \frac{s_x s_y}{s_x^2 - s_y^2} r$$

However this doesn't work in case of outliers.

σ_x and σ_y can estimated from interquartile range (or MAD).

Robust estimate of ρ

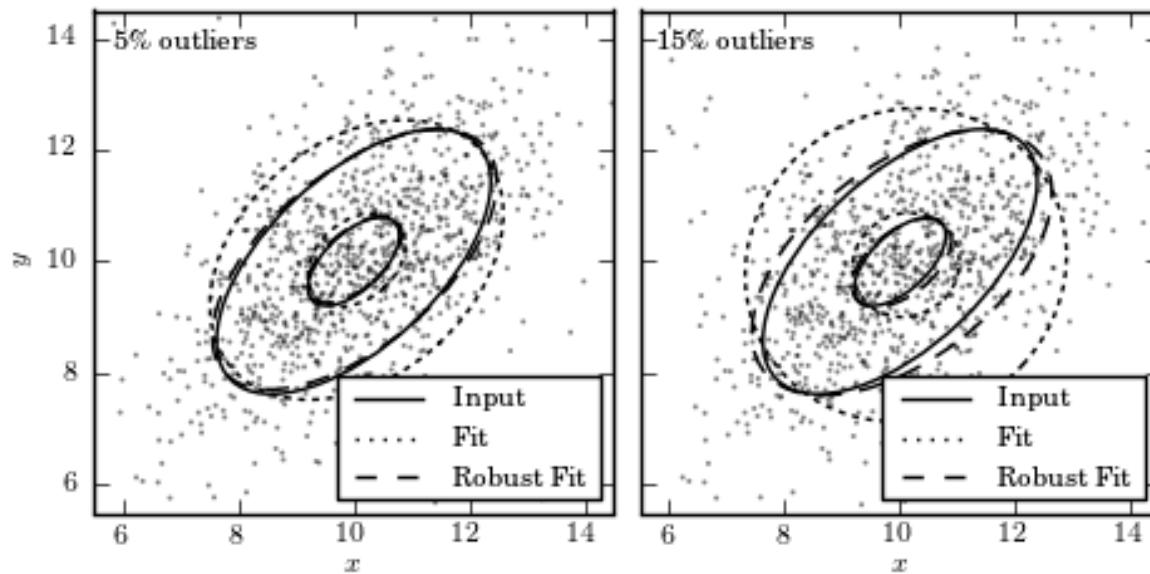
$$\rho = \frac{V_u - V_w}{V_u + V_w}$$

$$u = \sqrt{2} \left(\frac{x}{\sigma_x} + \frac{y}{\sigma_y} \right) \quad v = \sqrt{2} \left(\frac{x}{\sigma_x} - \frac{y}{\sigma_y} \right)$$

Ref : Shevlyakov and Smirnov, Austrian Journal of Statistics, 40, 147-156 (2011)

Python Code for robust estimate of bivariate dist params

```
From astroML.stats import fit_bivariate_normal  
(mu_r,sigma1_r,sigma2_r,alpha_r) = fit_bivariate_normal(x,y,robust=True)
```



An example of computing the components of a bivariate Gaussian using a sample with 1000 data values (points), with two levels of contamination. The core of the distribution is a bivariate Gaussian with

$(\mu_x, \mu_y, \sigma_1, \sigma_2, \alpha) = (10, 10, 2, 1, 45^\circ)$

The “contaminating” subsample contributes 5% (left) and 15% (right) of points centered on the same (μ_x, μ_y) , and with $\sigma_1 = \sigma_2 = 5$. Ellipses show the 1- and 3-sigma contours. The solid lines correspond to the input distribution. The thin dotted lines show the nonrobust estimate, and the dashed lines show the robust estimate of the best-fit distribution parameters (see Section 3.5.3 for details).

EP 4130/PH6130

Second and Third Week

Frequentist Parameter Estimation

Numerical Recipes 1992 edition Chapter 15

Bevington

arXiv: 0712.3028 by L. Verde

AstroML treatment of the above subject will be discussed later using more “formal” notations

- Estimate best fit parameters of a model (function) given data and associated errors.
- Error estimates on the parameters.
- Statistical Measure of Goodness of fit.

Maximum likelihood Estimation

- Maximize the probability of data given the parameters or the *likelihood* of the data given the parameters . This form of parameter estimation is called *maximum likelihood estimation*

Consider data points y_i with measurement errors σ_i which is independent and has a Gaussian distribution around the true model (or the function)

$$P \propto \prod_{i=1}^N \exp \left[-\frac{1}{2} \left(\frac{y_i - y(x_i, \theta)}{\sigma_i} \right)^2 \right]$$

To get maximum likelihood estimate $dP/d\theta = 0$

Least-Squares Minimization

$$\chi^2 = \sum_{i=1}^N \left(\frac{y_i - y(x_i, \theta)}{\sigma_i} \right)^2$$

Best-fit parameters are obtained by minimizing chi-square

For models that are linear in theta, probability distribution of χ^2 is chi-square PDF with $v = N - M$ degrees of freedom, where N is the number of data points and M is the number of free parameters.

A rule of thumb : $\chi^2 \sim v$ for a good fit (sometimes called “chi by eye”)

$\langle \chi^2 \rangle = v$ and its standard deviation = $\text{sqrt}(v)$

Reduced chi-square = χ^2 / v

Chi-Square G.O.F (Numerical Recipes Notation)

Probability that observed χ^2 for a correct model is less than a particular value $\hat{\chi}^2$
Is given by the cdf of the chi-square probability distribution

$$\mathcal{P}(\chi^2 < \hat{\chi}^2, \nu) = \mathcal{P}(\nu/2, \hat{\chi}^2/2) = \Gamma(\nu/2, \hat{\chi}^2/2)$$

where Γ is the incomplete gamma function. (Refer to Numerical Recipes Chap 6 & 15 for notation)

$$Q \equiv 1 - \mathcal{P}(\nu/2, \hat{\chi}^2/2)$$

is the probability that the observed χ^2 should exceed a particular value $\hat{\chi}^2$ by chance

Q is a quantitative measure of goodness of fit of a model is sometimes called chi-square GOF.

This is called p-value

Chi-Square Probability and Goodness of Fit

Chi-square probability or Chi-square likelihood for chi-square=Q and DOF by k is given by

$$p(Q|k) \equiv \chi^2(Q|k) = \frac{1}{2^{k/2}\Gamma(k/2)} Q^{k/2-1} \exp(-Q/2)$$

p-value or chi-square goodness of fit

$$p = \int_{\chi^2}^{\infty} p(Q|k)dQ = 1 - \frac{\gamma(\nu/2, \chi^2/2)}{\Gamma(\nu/2)}$$

Probability of finding χ^2_{min} as high as the one we got or higher if hypothesis is correct and is called p-value

Chi-square c.d.f given in terms of $P(\nu/2, \chi^2/2)$ (regularized Gamma function)

$$P(a, x) = \frac{1}{\Gamma(a)} \int_0^x e^{-t} t^{a-1} dt$$

Chi-Square Probability and GOF in Python

Chi-square likelihood in Python

```
stats.chi2(v).pdf(x2)
```

How to calculate p-value from chi-square (χ^2) and Degrees of freedom (v) in python

```
p-value = 1-stats.chi2(v).cdf(x2)
```

OR

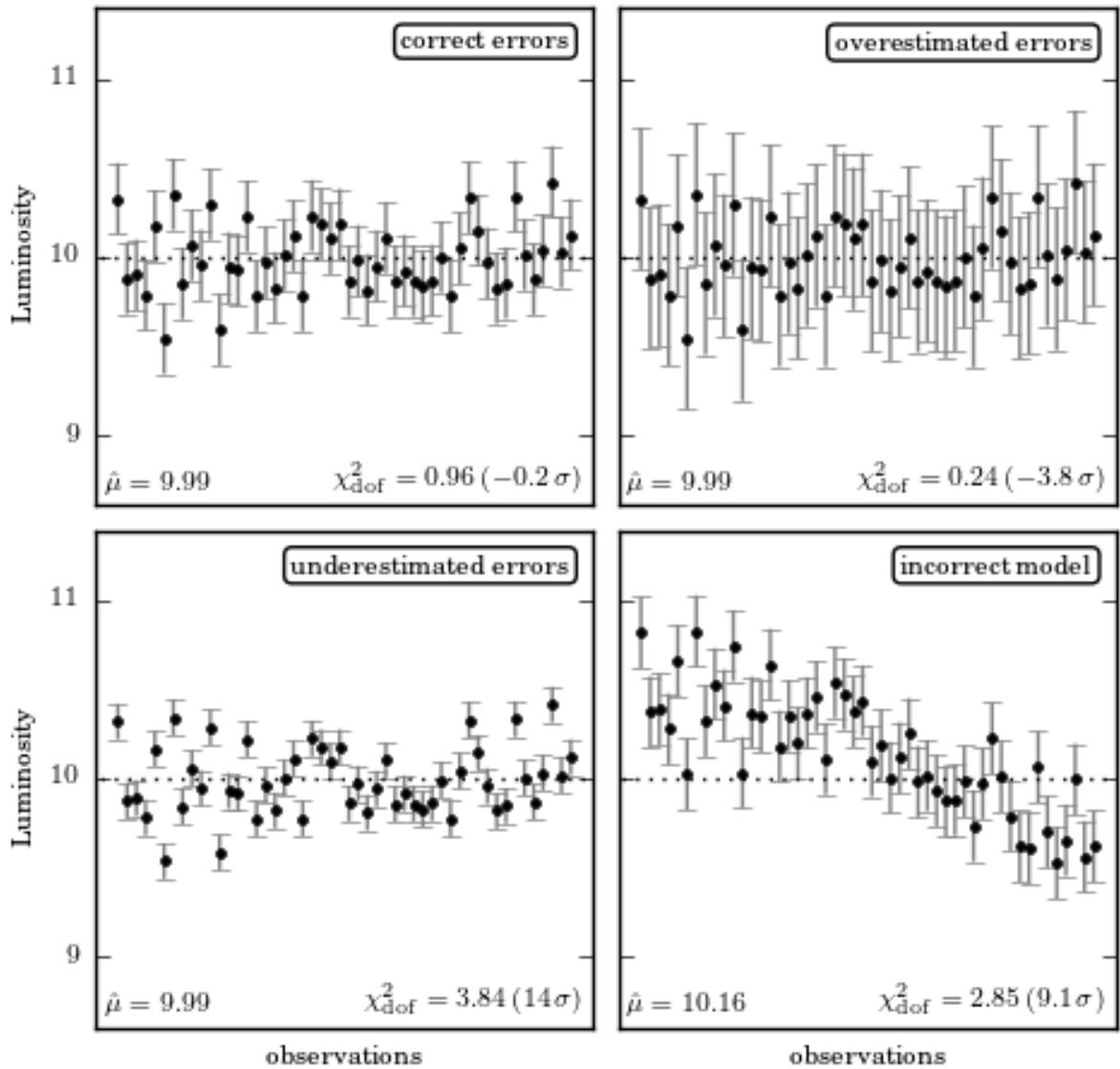
```
stats.chi2(v).sf(x2)
```

If p-value is a very small

- Model is wrong and can be rejected.
- Errors are really smaller than stated.
- Measurement errors are not Gaussianly distributed.

If p-value is a very large or close to 1

- Errors maybe overestimated
- Data are correlated and correlations are ignored in the fit
- Distribution is more compact than a Gaussian distributed. But this is almost never the case.



Astroml figure 4.1

No of sigmas = (reduced χ^2 - 1)/(Error in Reduced χ^2)

Error in Reduced χ^2 = $1/\sqrt{2(N-1)}$

Confidence Intervals

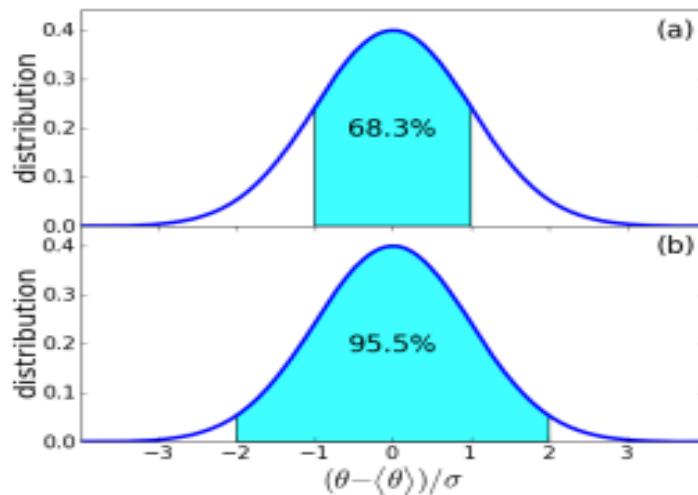
A *confidence region* (or *confidence interval*) is a region of M-dimensional space that contains a certain percentage (fraction) of the total probability distribution.

For eg. 99% confidence region implies that there is 99 % chance that the true parameter value falls within this region around the measured value. (Note that Bayesians use a different definition)

Usually in Physics/Astrophysics literature what is plotted in papers is 68.3%, 95.4% and 99.7% Confidence level.

These are obtained by contours of constant $\Delta\chi^2$ around the minimum χ^2
There is formal mathematical relation between $\Delta\chi^2$, confidence intervals and standard errors.

Confidence Intervals (single parameter)



arXiv: 1009.2755

$$\text{prob}(\theta_- \leq \hat{\theta} \leq \theta_+) = \int_{\theta_-}^{\theta_+} d\theta \text{prob}(\theta) = C,$$

Figure 5: Confidence intervals for the Gaussian distribution of mean $\langle \theta \rangle$ and standard deviation σ . If we draw N values of θ from a Gaussian distribution, 68.3% of the values will be inside the interval $[\langle \theta \rangle - \sigma, \langle \theta \rangle + \sigma]$ as shown in panel (a), whereas 95.5% of the values will be inside the interval $[\langle \theta \rangle - 2\sigma, \langle \theta \rangle + 2\sigma]$ as shown in panel (b).

C.I. (single parameter, asymmetric distribution)

- Prev. equation does not uniquely define a C.I. in case of an asymmetric distribution.
1. Symmetric interval: θ_- and θ_+ are symmetric around the parameter estimate, i.e., $\hat{\theta} - \theta_- = \theta_+ - \hat{\theta}$.
 2. Shortest interval: $\theta_+ - \theta_-$ is smallest for all intervals that satisfy Eq. (21).
 3. Central interval: The probabilities above and below the interval are equal, i.e., $\int_{-\infty}^{\theta_-} d\theta \text{prob}(\theta) = \int_{\theta_+}^{\infty} d\theta \text{prob}(\theta) = (1 - C)/2$.

Example of asymmetric likelihood

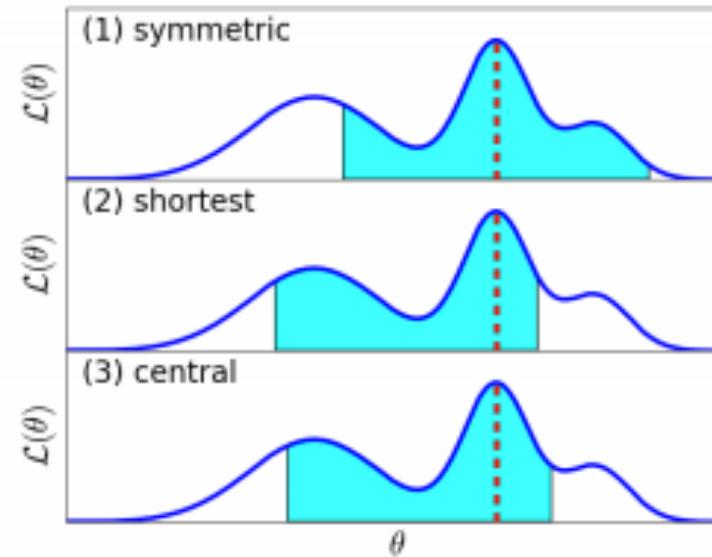


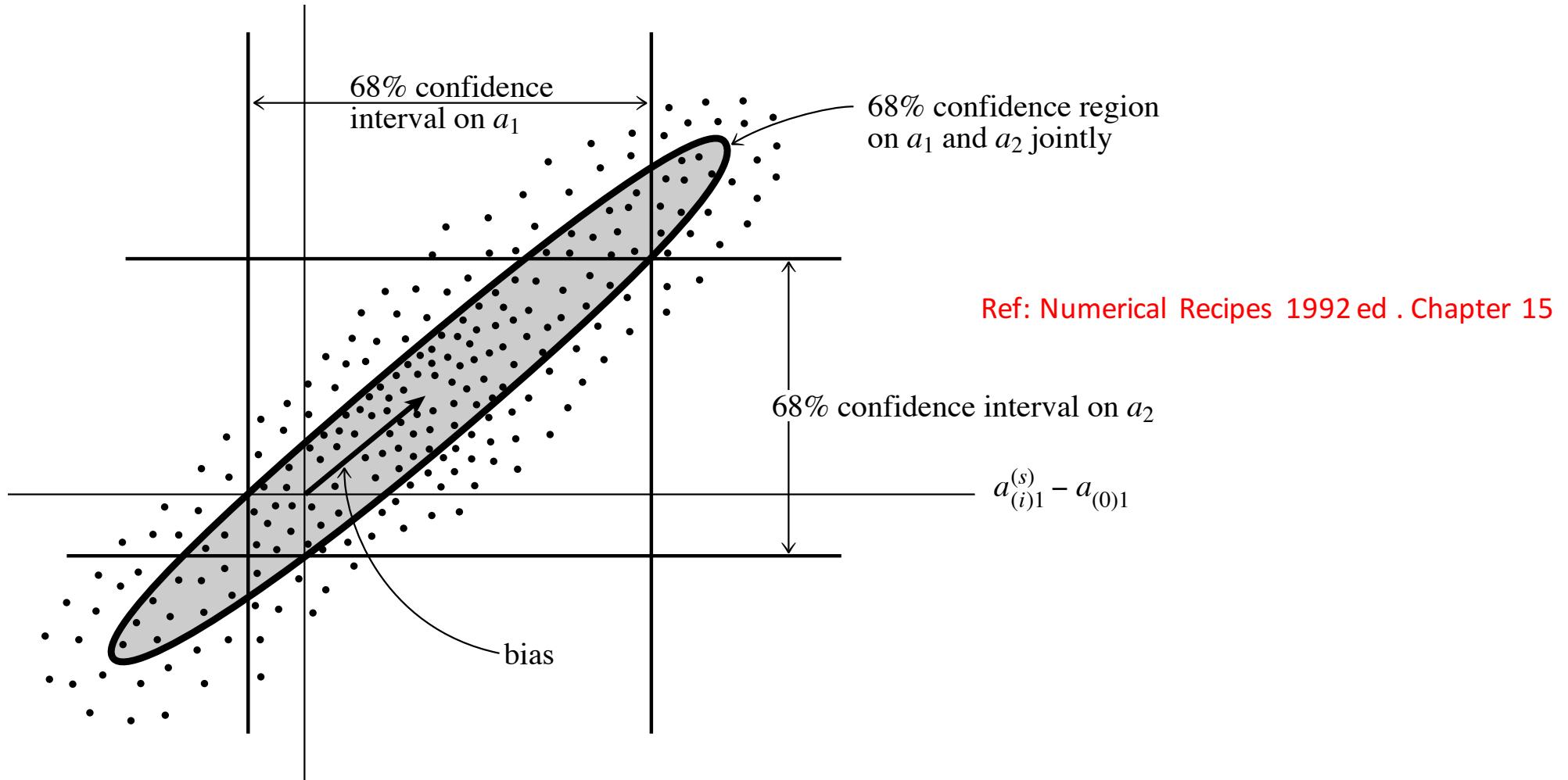
Figure 6: Different types of 68.3% confidence intervals for a multimodal likelihood function. The vertical dashed red line indicates the maximum-likelihood estimate $\hat{\theta}$. The panels are numbered according to the definitions in the main text.

Confidence Intervals in > 1 dimension

First, consider the case where the central-limit theorem indeed ensures that some (multidimensional) likelihood function is approximately Gaussian at its maximum position. Such (multivariate) Gaussians, with P -dimensional mean vector $\vec{\mu}$ and $P \times P$ covariance matrix Σ ,

$$\text{prob}(\vec{x}|\vec{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^P \det \Sigma}} \exp \left[-\frac{1}{2} (\vec{x} - \vec{\mu})^T \cdot \Sigma^{-1} \cdot (\vec{x} - \vec{\mu}) \right], \quad (22)$$

One sigma contour of a 2-d gaussian marks a 39.4% confidence region.



Cookbook for Confidence Intervals

Numerical Recipes (1992) Sect 15.6

- Assume μ is the number of free parameters for which you want to plot the joint confidence interval and $p\%$ is the confidence limit desired .
- $\Delta\chi^2$ is distributed as a chi-square distribution with μ degrees of freedom. $\Delta\chi^2 = \chi^2 - \chi^2_{\min}$
Calculate $\Delta\chi^2$ such that chi-square probability for μ free parameters is less than p .
 $P(\chi^2 < \Delta\chi^2, \mu) = p\%$

p	ν					
	1	2	3	4	5	6
68.3%	1.00	2.30	3.53	4.72	5.89	7.04
90%	2.71	4.61	6.25	7.78	9.24	10.6
95.4%	4.00	6.17	8.02	9.70	11.3	12.8
99%	6.63	9.21	11.3	13.3	15.1	16.8
99.73%	9.00	11.8	14.2	16.3	18.2	20.1
99.99%	15.1	18.4	21.1	23.5	25.7	27.8

$\Delta\chi^2$ tables as a function of number of free parameters (p) and confidence level (%)

Numerical recipes Sect 15.6

Delta chi-square Intervals in python

```
>>> stats.chi2(2).cdf(2.3)  
0.68336323062094673
```

Inverse of cdf in scipy is ppf

```
>>> stats.chi2(2).ppf(0.6833)  
2.2996006508750853
```

Ref: Numerical Recipes 1992 ed . Chapter 15

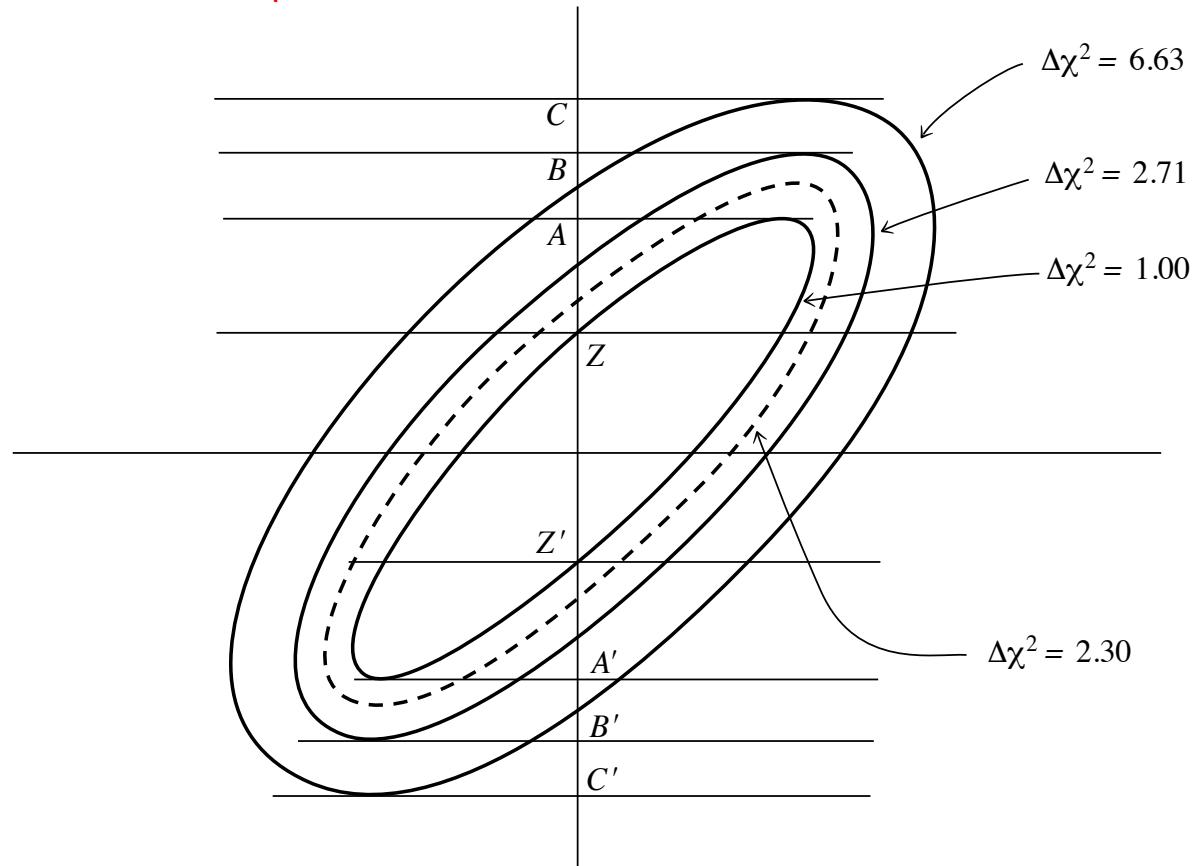


Figure 15.6.4. Confidence region ellipses corresponding to values of chi-square larger than the fitted minimum. The solid curves, with $\Delta\chi^2 = 1.00, 2.71, 6.63$ project onto one-dimensional intervals AA' , BB' , CC' . These intervals — not the ellipses themselves — contain 68.3%, 90%, and 99% of normally distributed data. The ellipse that contains 68.3% of normally distributed data is shown dashed, and has $\Delta\chi^2 = 2.30$. For additional numerical values, see accompanying table.

How to calculate projected 1-D errors

Numerical Recipes (1992) Sect 15.6

For one variable

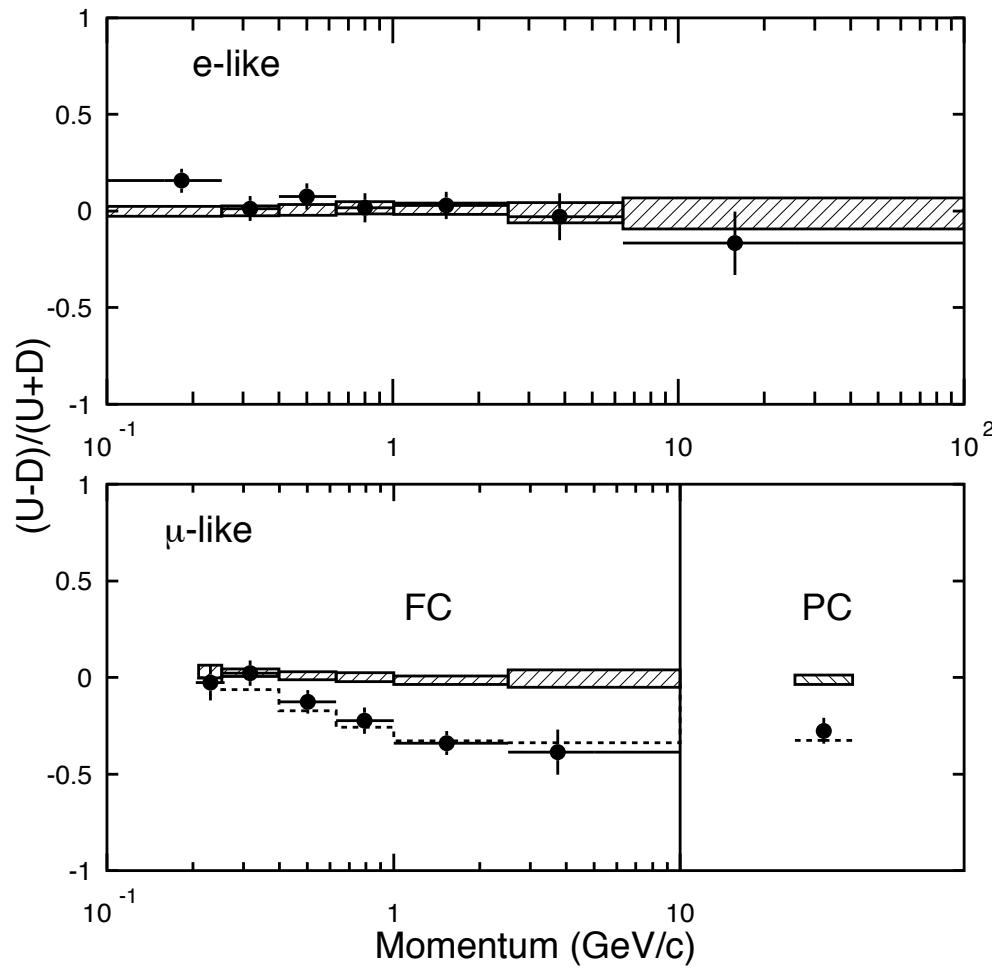
$$\delta a_1 = \pm \sqrt{\Delta \chi^2} \sqrt{C_{11}}$$

This gives relation between formal standard error $\sigma_{11} = \text{sqrt}(C_{11})$ and confidence interval (δa_1)

[C] = covariance matrix = Inverse of Fisher Information matrix or Inverse of curvature matrix
(to be defined later)

68% confidence interval	\longleftrightarrow	1σ
95% confidence interval	\longleftrightarrow	2σ
99% confidence interval	\longleftrightarrow	3σ

Note that this is valid only for 1 dimensions



hep-ex/9807003

$$\chi^2 = \sum_{\cos\Theta,p} (N_{DATA} - N_{MC})^2 / \sigma^2 + \sum_j \epsilon_j^2 / \sigma_j^2,$$

Best-fit $\chi^2 = 65.2$ for 67 DOF

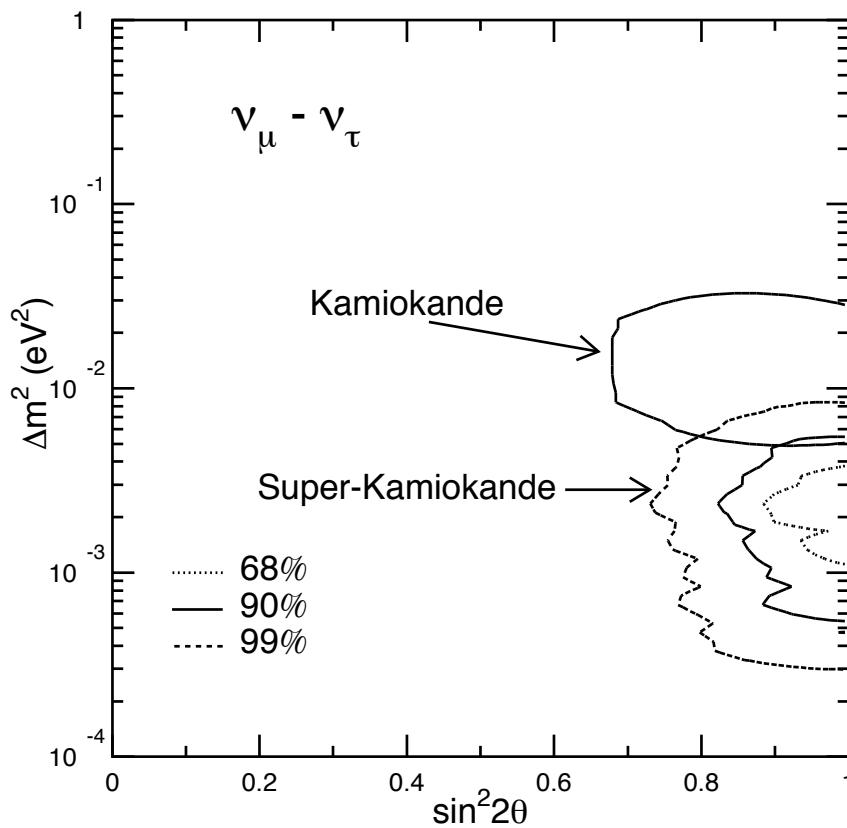


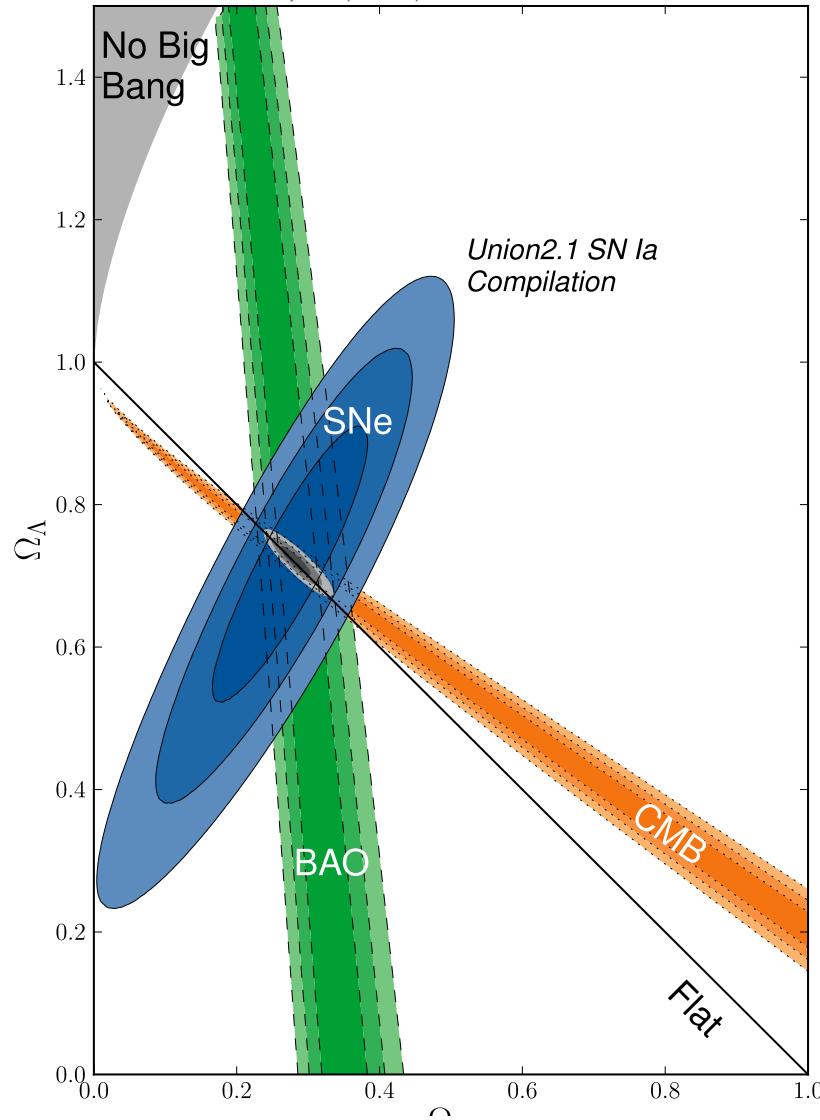
FIG. 2. The 68%, 90% and 99% confidence intervals are shown for $\sin^2 2\theta$ and Δm^2 for $\nu_\mu \leftrightarrow \nu_\tau$ two-neutrino oscillations based on 33.0 kiloton-years of Super-Kamiokande data. The 90% confidence interval obtained by the Kamiokande experiment is also shown.

hep-ex/9807003

$\Delta\chi^2$ intervals of 2.6, 5.0, and 9.6
Correspond to 68%, 90% and 99%
confidence intervals

2015 Physics Nobel Prize

Supernova Cosmology Project
Suzuki, et al., Ap.J. (2011)



arXiv:1105.3470

68.3%, 95.4% and 99.7% confidence levels

$$\chi^2_{\text{stat}} = \sum_{\text{SNe}} \frac{[\mu_B(\alpha, \beta, \delta, M_B) - \mu(z; \Omega_m, \Omega_w, w)]^2}{\sigma_{\text{lc}}^2 + \sigma_{\text{ext}}^2 + \sigma_{\text{sample}}^2}.$$

2011 Physics Nobel Prize

Generalization of chi-square for Poisson data

- In case of bins with very low counts (or Poisson data) for comparing models with data we use :

$$C = 2 \sum_{i=1}^N [f(x_i, \theta) - y_i + y_i \ln(y_i/f(x_i, \theta))]$$

Cash ApJ 229, 939 (1979)
Baker and Cousins, NIM 221, 437 (1984)

In astrophysical literature ONLY, C is called Cash statistics or C-stat. Based on C , one can calculate goodness of fit of a model in the same way as for a Gaussian distribution.

Also this can be used for model comparison as ΔC between two models of parameters also has a χ^2 distribution with DOF = no of free parameters

Derivation of Cash statistics

- Obtained from likelihood ratio test. Ref. Baker and Cousins, NIM 221, 437 (1984)

Define $\lambda = \frac{L_p(y, n)}{L_p(n, n)}$ where n = observed data and y= expected model

where $L_p(y, n) = \prod_i \exp(-y_i) y_i^n / n!$

$$L_p(n, n) = \prod_i \exp(-n_i) n_i^{n_i} / n_i!$$

$\chi^2 = -2 \ln \lambda$ asymptotically obeys chi-square distribution (Wilk's theorem)

and evaluates to the definition of Cash statistics on previous slide

THE ASTROPHYSICAL JOURNAL, 228:939-947, 1979 March 15

© 1979. The American Astronomical Society. All rights reserved. Printed in U.S.A.

PARAMETER ESTIMATION IN ASTRONOMY THROUGH APPLICATION OF THE LIKELIHOOD RATIO

WEBSTER CASH

Space Sciences Laboratory, Department of Physics, University of California, Berkeley

Received 1977 August 22; accepted 1978 August 24

ABSTRACT

Many problems in the experimental estimation of parameters for models can be solved through use of the likelihood ratio test. Applications of the likelihood ratio, with particular attention to photon counting experiments, are discussed. The procedures presented solve a greater range of problems than those currently in use, yet are no more difficult to apply. The procedures are proved analytically, and examples from current problems in astronomy are discussed.

Subject heading: functions: numerical methods

Recent exposition on Cash statistic

2.1. The method of maximum likelihood and the C statistic

The N Poisson data points D_i are assumed to be measurements from a parent model that describes the properties of the source. Models can be either fully specified with no free parameters, or more commonly featuring a number of free parameters. The likelihood of the data with the model is

$$\mathcal{L} = \prod_{i=1}^N \frac{e^{-S_i} S_i^{D_i}}{D_i!} \quad (1)$$

arXiv:1912.05444

where D_i is an integer number of counts (the i -th data point) and S_i the mean value of the model for that data point. It is convenient to calculate the logarithm of the likelihood,

$$\ln \mathcal{L} = \sum_{i=1}^N (-S_i + D_i \ln S_i - \ln D_i!)$$

and then define the *Cash* or C statistic as

$$C = -2 \ln \mathcal{L} - B = 2 \sum_{i=1}^N (S_i - D_i + D_i \ln(D_i/S_i)) = \sum_{i=1}^N C_i \quad (2)$$

where

$$B = 2 \sum_{i=1}^N (D_i - D_i \ln D_i + \ln D_i!)$$

and

$$C_i = 2 (S_i - D_i + D_i \ln(D_i/S_i)).$$

Relation between Cash statistic and χ^2

it can be shown that the C statistic is approximately equal to the χ^2 statistic:

arXiv:1912.05444

$$\begin{aligned} C \simeq 2 \sum_{i=1}^N & \left(S_i - D_i - D_i \left(-\frac{d}{D_i} - \frac{1}{2} \left(\frac{d}{D_i} \right)^2 \right) \right) = \\ & 2 \sum_{i=1}^N \left(S_i - D_i + d + \frac{d^2}{2D_i} \right) = \sum_{i=1}^N \frac{(D_i - S_i)^2}{D_i} = \sum_{i=1}^N \frac{(D_i - S_i)^2}{S_i} \frac{S_i}{D_i}, \quad (3) \end{aligned}$$

where, by definition, $S_i - D_i + d = 0$. Since $D_i \sim \text{Poiss}(S_i)$, an estimate of the deviation d is given by the standard deviation of the Poisson distribution, i. e., $|d| \simeq \sqrt{S_i}$, further approximated as $|d| \simeq \sqrt{D_i}$, as also suggested by [9]. Using this approximation, each term in Equation 3 differs from a χ^2 distribution by a factor

$$\frac{S_i}{D_i} = \left(1 - \frac{d}{D_i} \right) \simeq \left(1 \pm \frac{1}{\sqrt{D_i}} \right)$$

Example of usage of Cash statistics

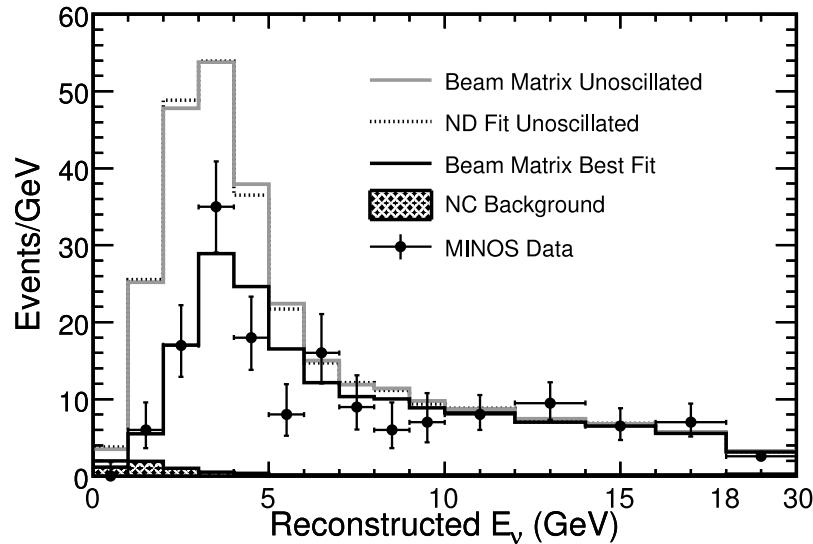


FIG. 3: Comparison of the Far Detector spectrum with predictions for no oscillations for both analysis methods and for oscillations with the best-fit parameters from the Beam Matrix extrapolation method. The estimated NC background is also shown. The last energy bin contains events between 18-30 GeV.

$$\chi^2 = \sum_{nbins} (2(e_i - o_i) + 2o_i \ln(o_i/e_i)) + \sum_{n_{sys}} \frac{\Delta s_j^2}{\sigma_{s_j}^2}$$

hep-ex/0607088

Disconnect in usage of nomenclature

The jargon “Cash statistics” is used only in Astrophysics and NOT in particle physics

energy and the observed number of events in each bin is compared to the expected number of events for this oscillation hypothesis. The best fit parameters are those which minimize $\chi^2 = -2 \ln \lambda$ where λ is the likelihood ratio:

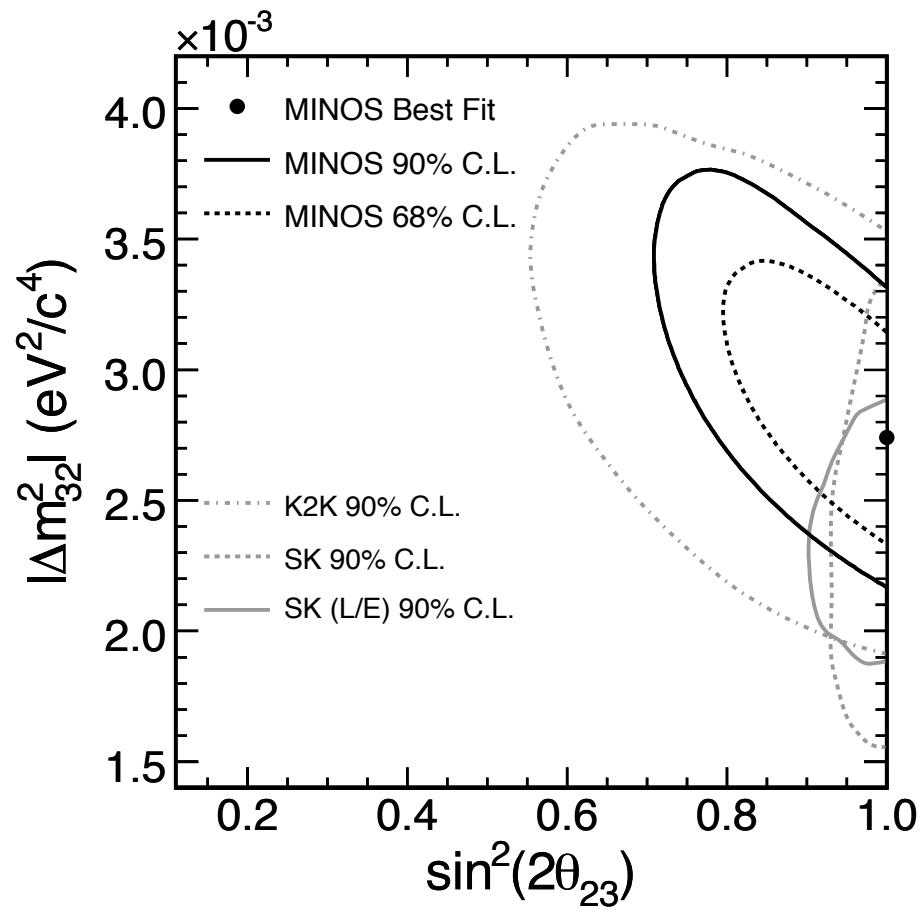
$$\chi^2 = \sum_{nbins} (2(o_i - e_i) + 2o_i \ln(o_i/e_i)) + \sum_{nsys} \frac{\Delta s_j^2}{\sigma_{s_j}^2} \quad (2)$$

where o_i and e_i are the observed and expected numbers of events in bin i , and the $\Delta s_j^2/\sigma_{s_j}^2$ are the penalty terms for nuisance parameters associated with the systematic uncertainties. The expected number of events depends

certainties of $B(m)$ are small due to its being drawn from an area that is 25 times the area of the cluster. That is, the uncertainties of the cluster LF are dominated by the cluster field. We use the Cash (1979) statistic with a maximum likelihood estimator C_{stat} to estimate the parameters in the fitting.

$$C_{\text{stat}} = 2 \sum_j \left(M(m_j) - N(m_j) + N(m_j) \ln \left(\frac{N(m_j)}{M(m_j)} \right) \right), \quad (5)$$

where j runs over all the magnitude bins in the fit and N is the observed magnitude distribution for the cluster. Using the estimator C_{stat} allows us to estimate the goodness of fit (GOF) for the data following the Poisson distribution in the same way as a χ^2 -distribution. The GOF of the LF fitting is defined by the ratio of the best-fit C_{stat} to the degrees of freedom (d.o.f) (i.e., $\text{GOF} = C_{\text{stat}}/\text{d.o.f}$) and has a corresponding probability to exceed that provides information about tension between the best fit model and the data.



hep-ex/0607088 (based on
minimization of Cash statistics)

χ^2 including covariances

$$\begin{aligned} \mathbf{Y} &= \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, & \xrightarrow{\hspace{10em}} \quad \mathbf{Y} &= \mathbf{A} \mathbf{X} \\ \mathbf{A} &= \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_N \end{bmatrix}, & & \\ \mathbf{C} &= \begin{bmatrix} \sigma_{y1}^2 & 0 & \cdots & 0 \\ 0 & \sigma_{y2}^2 & \cdots & 0 \\ & \ddots & & \\ 0 & 0 & \cdots & \sigma_{yN}^2 \end{bmatrix} & \chi^2 &= \sum_{i=1}^N \frac{[y_i - f(x_i)]^2}{\sigma_{yi}^2} \equiv [\mathbf{Y} - \mathbf{A} \mathbf{X}]^\top \mathbf{C}^{-1} [\mathbf{Y} - \mathbf{A} \mathbf{X}] \end{aligned}$$

(Definition of C above includes no-correlations in errors and hence off-diagonal elements are 0. Also A above is only true for a straight line fit of the form $y = Ax + c$)

More details in arXiv:1008.4686

MLE in case of Truncated/Censored Data

- Truncation : when $S(x) = 0$ for $x > x_{\max}$ or $x < x_{\min}$
- Censoring : Measurement of an existing source only resulted in *upper* limits

MLE in case of truncation becomes :

$$p(x_i|\mu, \sigma, x_{min}, x_{max}) = C(\mu, x_{min}, x_{max}) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(x - \mu)^2}{2\sigma^2}\right)$$

where the normalization constant C is given by:

$$C(\mu, \sigma, x_{min}, x_{max}) = (P(x_{max}|\mu, \sigma) - P(x_{min}|\mu, \sigma))^{-1}$$

Chi-square for errors in X-variable

- There is no uniform method to deal with this. Multiple methods used (BCES, ODR, etc)
Some references:

[physics/051182](#) by Guido D'agostini

[astro-ph/9605002](#) by Akritas and Bershady

[arXiv:0705.2774](#) Also other techniques such as Orthogonal Distance Regression

Simplest recipe as follows: If σ_x and σ_y denote the error in X (independent) and Y (dependent) variables, then total error σ_t is given by

$$\sigma_t = \sqrt{\sigma_y^2 + \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2}$$

σ_t can be plugged in the expression for χ^2 in the denominator

Classical Statistical Inference (astroML)

- **Point Estimation** : Best estimate for a model parameter based on available data
- **Confidence Estimation** : How confident should we be in our point estimate
- **Hypothesis Testing** : Are data consistent with a given hypothesis or model?

Two types of statistical paradigms to address the statistical inference questions : classical or *Frequentist* paradigm and the *Bayesian* paradigm.

Likelihood Function

- Starting point of both MLE and Bayesian estimation is *likelihood* of the data. The data likelihood represents a quantitative description of our measuring process also called P(DIM)

$$\mathcal{L} \equiv p(\{x_i\}|M(\theta)) = \prod_{i=1}^N p(x_i|M(\theta))$$

where M stands for the model and θ is model parameter vector

- Although the interpretation of \mathcal{L} is the probability of the data given the model, \mathcal{L} is not a true properly normalized PDF. Likelihood of a single data point is a pdf. But the product of such functions is no longer normalized to 1.
- Likelihood is a function of both the data and the model.

Best-fit Model parameters which maximize P(DIM) are called point estimates.

Frequentist	Bayesian
Probabilities refer to relative frequencies of events	Refer to degree of subjective belief, not limiting frequency
Parameters are fixed unknown constants. Probability statements about parameters are meaningless	Probability statements can be made about things other than data including the model parameters and models themselves.
Frequentists consider model parameters to be fixed and data to be random	Bayesians consider data to be fixed and model parameters to be random
Confidence intervals should have well-defined long run frequency properties. 95% c.i. should bracket the true value of the parameter with a limiting frequency of at least 95%.	Inferences about a parameter are made by producing its probability. <i>Distribution quantifies amount of uncertainty of our knowledge about the parameter.</i> These are called credible intervals.

Bayesian data analysis often computationally intensive compared to frequentist analysis

Maximum Likelihood Estimators

- They are *consistent* estimators; they converge to the true parameter values as the number of data points increases.
- They are *asymptotically normal* estimators. The distribution of the parameter estimate, as the number of data points increase to infinity approaches a normal distribution centered on MLE with a certain spread.
- They asymptotically achieve the theoretical minimum possible variance called **Cramer-Rao bound**

(they achieve best possible error given the data at hand)

MLE Confidence Intervals

- We first compute the error matrix or Fisher information matrix.

$$F_{jk} = \frac{-\partial^2 \ln L}{\partial \theta_j \partial \theta_k}$$

- Inverse of the Fisher information matrix gives a lower bound on the variance of any unbiased estimator of θ . This is called **Cramer-Rao** bound.

$$\sigma_{jk} = \sqrt{[F^{-1}]_{jk}}$$

- Diagonal elements of σ correspond to marginal parameters for θ . If non-diagonal elements are 0, inferred values of parameters are uncorrelated.

Jackknife, bootstrap, etc.

To estimate a parameter we have various tools such as maximum likelihood, least squares, etc.

Usually one also needs to know the variance (or the full sampling distribution) of the estimator – this can be more difficult.

Often use asymptotic properties, e.g., sampling distribution of ML estimators becomes Gaussian in large sample limit; std. dev. from curvature of log-likelihood at maximum.

The jackknife and bootstrap are examples of “resampling” methods used to estimate the sampling distribution of statistics.

In HEP we often do this implicitly by using Toy MC to determine sampling properties of statistics (e.g., Brazil plot for 1σ , 2σ bands of limits).



Glenn Cowan
Lecture notes

The Bootstrap (Efron, 1979)

Idea is to produce a set of “bootstrapped” data samples of same size as the original (real) one by sampling from some distribution that approximates the true (unknown) one.

By evaluating a statistic (such as an estimator for a parameter θ) with the bootstrapped-samples, properties of its sampling distribution (often its variance) can be estimated.

If the data consist of n events, one way to produce the bootstrapped samples is to randomly select from the original sample n events *with replacement* (the non-parametric bootstrap).

That is, some events might get used multiple times, others might not get used at all.

In other cases could generate the bootstrapped samples from a parametric MC model, using parameter values estimated from real data in the MC (parametric bootstrap).

Glenn Cowan lecture notes

The Bootstrap (cont.)

Call the data sample $\mathbf{x} = (x_1, \dots, x_n)$, observed data are \mathbf{x}_{obs} ,
and the bootstrapped samples are $\mathbf{x}_1^*, \mathbf{x}_2^*, \dots$

Idea is to use the distribution of

$$\hat{\theta}(\mathbf{x}^*) - \hat{\theta}(\mathbf{x}_{\text{obs}})$$

as an approximation for the distribution of

$$\hat{\theta}(\mathbf{x}) - \theta$$

In the first quantity everything is known from the observed data plus bootstrapped samples, so we can use its distribution to estimate bias, variance, etc. of the estimator $\hat{\theta}$.

Glenn Cowan lecture notes

Bootstrap Resampling

- Bootstrap and Jackknife methods rely on resampling of the dataset $\{x_i\}$ (instead of generating them from probability distributions).
- We do not know $h(x)$ [parent distribution of data] and the best we can do is computations relying on various estimates of $h(x)$, based upon the data (called $f(x)$).

Bootstrapping is based upon the approximation

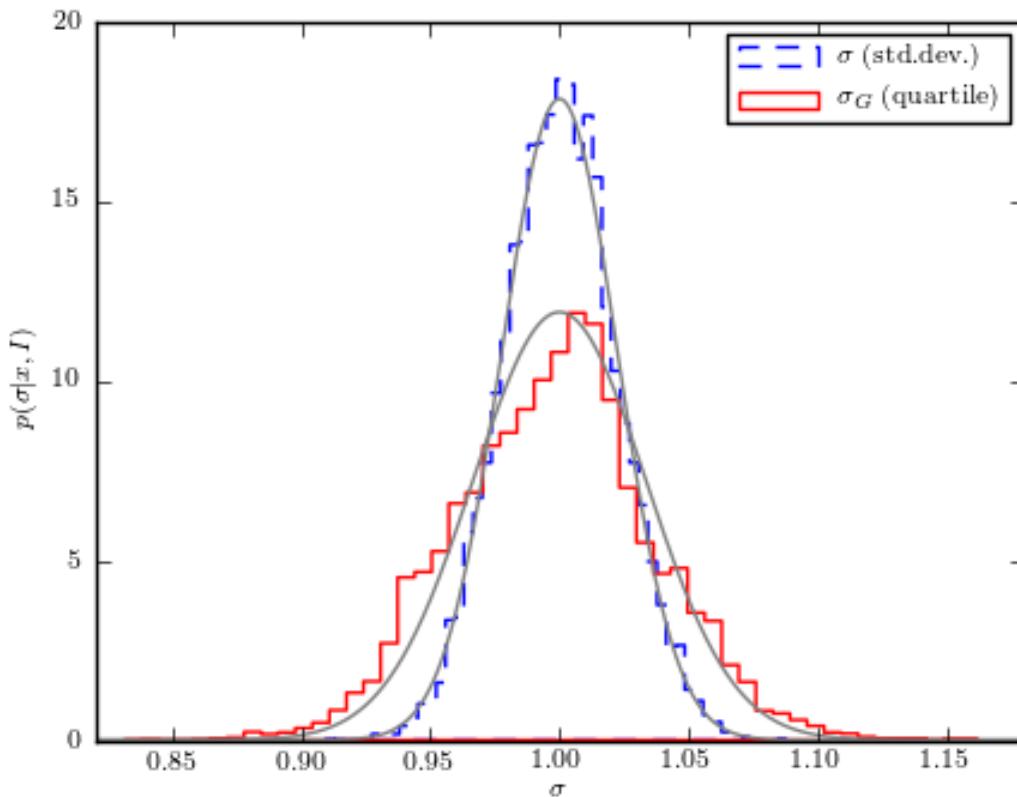
$$f(x) = \frac{1}{N} \sum_{i=1}^N \delta(x - x_i)$$

$f(x)$ maximizes the probability of obtaining $h(x)$ and is the MLE of $h(x)$. Therefore, $f(x)$ can be used as a proxy for $h(x)$.

Basic idea of bootstrap is to draw new samples from the measured dataset itself. Drawing these ``bootstrap'' resamples can be done with replacement , i.e. same data point can occur multiple times in our data sample. (Non-parametric bootstrap).

For more details see Efron (1979); Lupton (1993); G.J. Babu lecture notes in PSU school

Example of Bootstrap Resampling



The bootstrap uncertainty estimates for the sample standard deviation σ (dashed line; see eq. 3.32) and σ_G (solid line; see eq. 3.36). The sample consists of $N = 1000$ values drawn from a Gaussian distribution with $\mu = 0$ and $\sigma = 1$. The bootstrap estimates are based on 10,000 samples. The thin lines show Gaussians with the widths determined as $s/\sqrt{2(N - 1)}$ (eq. 3.35) for σ and $1.06s/\sqrt{N}$ (eq. 3.37) for σ_G .

$$\sigma_G = 0.7413(q_{75} - q_{25})$$

Formulae for
Standard error for σ and σ_G provided
In astroML book

The Jackknife

Invented by Quenouille (1949) and Tukey (1958).

Glenn Cowan lecture notes

Suppose data sample consists of n events: $\mathbf{x} = (x_1, \dots, x_n)$.

We have an estimator $\hat{\theta}(\mathbf{x})$ for a parameter θ .

Idea is to produce pseudo data samples $\mathbf{x}_{-i} = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$ by leaving out the i th event.

See, e.g., Notes on Jackknife and Bootstrap by G. J. Babu:

[www.iiap.res.in/astrostat/School10/LecFiles/
JBabu_JackknifeBootstrap_notes.pdf](http://www.iiap.res.in/astrostat/School10/LecFiles/JBabu_JackknifeBootstrap_notes.pdf)

Assume α is the jackknife estimate of the sample (with one data point removed) and α_N is the value estimated from the full dataset.

Bias-corrected Jackknife estimate of α is given by:

$$\alpha_J = \alpha_N + \Delta\alpha \quad (1)$$

$$\Delta\alpha = (N - 1)(\alpha_N - \frac{1}{N} \sum_{i=1}^N \alpha_i^*)$$

Standard error for a jackknife estimator is given by:

$$\sigma_\alpha = \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^N [N\alpha_N - (N-1)\alpha_i^*]^2} \quad (2)$$

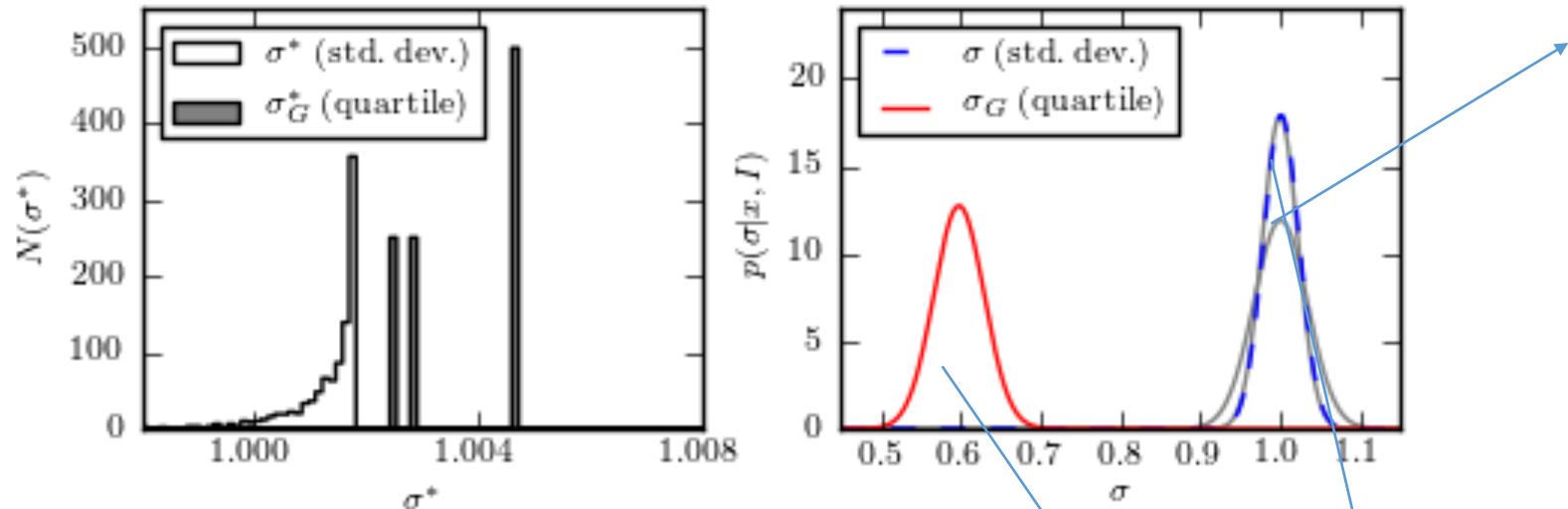
- Confidence Limits for α can be computed using Student's t-distribution with t defined as :

$$t = (\bar{\alpha} - \alpha_J)/\sigma_{\bar{\alpha}}$$
 with $N-1$ degrees of freedom.
- Jackknife standard error is more reliable than jackknife bias correction.
- Jackknife bias correction completely fails for any robust statistics (median, quantiles, rank-based statistics).

Qt: Should one use bootstrap or jackknife?

For smooth statistics they produce similar results. Bootstrap better for calculating confidence intervals

In Machine learning , cross-validation and aggregating are closely related to jackknife and bootstrap



The jackknife uncertainty estimates for the width of a Gaussian distribution. This example uses the same data as figure 4.3. The upper panel shows a histogram of the widths determined using the sample standard deviation, and using the interquartile range. The lower panel shows the corrected jackknife estimates (eqs. 4.33 and 4.35) for the two methods. The gray lines show the theoretical results, given by eq. 3.35 for σ and eq. 3.37 for σ_G . The result for σ matches the theoretical result almost exactly, but note the failure of the jackknife to correctly estimate σ_G (see the text for a discussion of this result).

Expected distribution for σ_G

Computed using (1) and (2)

Pdf from jackknife mean and sigma for sigma

Pdf from jackknife mean and sigma for sigma)G

Jackknife & Bootstrap Functions in AstroML

```
from astroML.resample import jackknife  
  
[mean, stddev, raw_distribution] =  
jackknife(data, user_statistic, kwargs, return_raw_distribution, pass_indices)  
  
Example: mu1, sigma_mu1, mu1_raw =  
jackknife(data, np.std, kwargs=dict(axis=1, ddof=1), return_raw_distrbution=True)  
  
astroML.resample.bootstrap(data, n_bootstraps, user_statistic, kwargs, pass_indices, random_state)  
  
Example: mu2_bootstrap = bootstrap(data, n, sigmaG, kwargs=dict(axis=1))  
  
Lookup astroML.resample.jackknife and astroML.resample.bootstrap for full documentation
```

Linear Algebra Solution to Straight line fit

As an example of a more realistic data-driven analysis, let's consider a simple three-parameter linear model which fits a straight-line to data with unknown errors. The parameters will be the the y-intercept α , the slope β , and the (unknown) normal scatter σ about the line.

For data $D = \{x_i, y_i\}$, the model is

$$\hat{y}(x_i|\alpha, \beta) = \alpha + \beta x_i,$$

and the likelihood is the product of the Gaussian distribution for each point:

$$\mathcal{L}(D|\alpha, \beta, \sigma) = (2\pi\sigma^2)^{-N/2} \prod_{i=1}^N \exp \left[\frac{-(y_i - \hat{y}(x_i|\alpha, \beta))^2}{2\sigma^2} \right].$$

arXiv:1411.5018

Jake Van Der Plas

Frequentism and Bayesianism :
A Python Driven Primer

algebra. If we define the *parameter vector*, $\theta = [\alpha \ \beta]^T$; the *response vector*, $Y = [y_1 \ y_2 \ y_3 \ \cdots \ y_N]^T$; and the *design matrix*,

$$X = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_1 & x_2 & x_3 & \cdots & x_N \end{bmatrix}^T,$$

it can be shown that the maximum likelihood solution is

$$\hat{\theta} = (X^T X)^{-1} (X^T Y).$$

The confidence interval around this value is an ellipse in parameter space defined by the following matrix:

$$\Sigma_{\hat{\theta}} \equiv \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha\beta} \\ \sigma_{\alpha\beta} & \sigma_\beta^2 \end{bmatrix} = \sigma^2 (M^T M)^{-1}.$$

Here σ is our unknown error term; it can be estimated based on the variance of the residuals about the fit. The off-diagonal elements of $\Sigma_{\hat{\theta}}$ are the correlated uncertainty between the estimates. In code, the computation looks like this:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy import ndimage
np.random.seed(42)
theta_true = (25, 0.5)
xdata = 100 * np.random.random(20)
ydata = theta_true[0] + theta_true[1] * xdata
ydata = np.random.normal(ydata, 10)
# Compute the frequentist version
X = np.vstack([np.ones_like(xdata), xdata]).T
theta_freq = np.linalg.solve(np.dot(X.T, X),
                            np.dot(X.T, ydata))
y_model = np.dot(X, theta_freq)
sigma_y = np.std(ydata - y_model)
Sigma_freq = sigma_y ** 2 *
np.linalg.inv(np.dot(X.T, X))
```

Code by Jake Van der Plas

Full source code available at <https://tinyurl.com/y9hhded4>

```
def get_principal(Sigma):
    # See Ivezic, Connolly, VanderPlas, and Gray, section 3.5.2
    sigma_x2 = Sigma[0, 0]
    sigma_y2 = Sigma[1, 1]
    sigma_xy = Sigma[0, 1]

    alpha = 0.5 * np.arctan2(2 * sigma_xy, sigma_x2 - sigma_y2)
    tmp1 = 0.5 * (sigma_x2 + sigma_y2)
    tmp2 = np.sqrt(0.25 * (sigma_x2 - sigma_y2) ** 2 + sigma_xy ** 2)

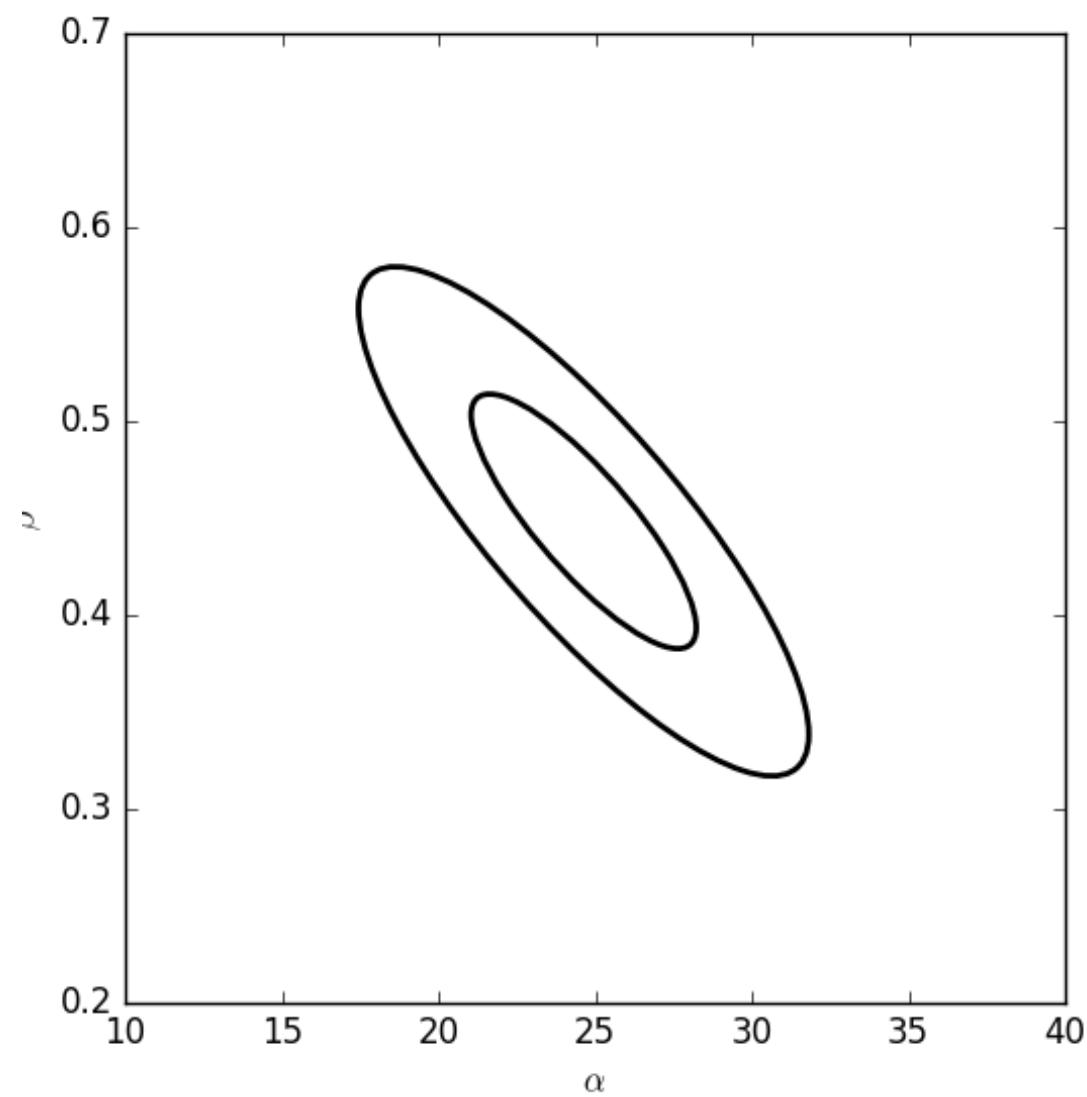
    return np.sqrt(tmp1 + tmp2), np.sqrt(tmp1 - tmp2), alpha
```

```
print("Frequentist Version")
print(" theta = {0}".format(theta_freq))
print(" sigma_y = {0}".format(sigma_y))
print(" Sigma = {0}".format(Sigma_freq))

from matplotlib.patches import Ellipse
fig, ax = plt.subplots(figsize=(6, 6))
sigma1, sigma2, alpha = get_principal(Sigma_freq)
for nsigma in [1, 2]:
    ax.add_patch(Ellipse(theta_freq,
                          2 * nsigma * sigma1, 2 *
nsigma * sigma2,
                          angle=np.degrees(alpha),
                          lw=2, ec='k', fc='none'))

ax.plot([0, 0], [0, 0], '-k', lw=2)
ax.set_xlim(10, 40)
ax.set_ylim(0.2, 0.7)
ax.set_xlabel(r'$\alpha$')
ax.set_ylabel(r'$\beta$')

plt.show()
```



Model Comparison

Week 4+5

Introduction to Model Comparison

Qt : Given a dataset:

- 1) how do we decide which among (two or more) models fit the data best
- 2) How do we quantify the significance of goodness of fit of the best model compared to disfavored model?

Multiple methods to address the above questions :

- Frequentist model comparison tests (based on difference in chi-square or log-likelihood ratio)
- Bayesian methods (**to be discussed later**)
- Information criterion based methods

Frequentist Model Comparison

<http://jakevdp.github.io/blog/2015/08/07/frequentism-and-bayesianism-5-model-selection/>

arXiv:1607.03549 by Louis Lyons (Sect 13)

Astro-ph/0701113

arXiv:1901.07726 Kerscher & Weller

arXiv:1607.03845 (SD)

A simple application to test if measurements of Newton's Constant show sinusoidal dependence with time as proposed by some authors.

arXiv: 1706.01202 (Shalini Ganguly + SD)

Statistical Significance of spectral lag transition in GRB160625B

arXiv:1906.05726 Aditi Krishak, Aisha Dantuluri, SD model comparison of annual modulation in DM experiments

arXiv:2003.10127 Aditi Krishak, SD model comparison for modified gravity in Eot-wash

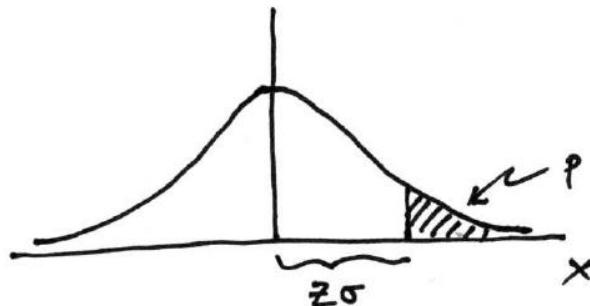
Frequentist Model Comparison Summary

- Calculate best fit χ^2 for each model
- **Calculate χ^2 for each model and the one with larger value of χ^2 GOF (or χ^2 pdf) is the preferred value.**
- Use the fact that if the two models are nested $\Delta\chi^2$ between the two models has a χ^2 distribution with DOF = difference in number of free parameters between model 2 and model 1.
- Consider the model with fewer free parameters as the null hypothesis calculate the p-value that simply by chance we would see the more complicated model.
- p-value = 1-chi-square CDF for DOF=difference in no of free parameters and $\chi^2 = \Delta\chi^2$ (between the two models)

In Python use `p-value = 1-stats.chi2(v).cdf($\Delta\chi^2$)`

Significance from p -value

Often define significance Z as the number of standard deviations that a Gaussian variable would fluctuate in one direction to give the same p -value.



Glenn Cowan lecture
Notes

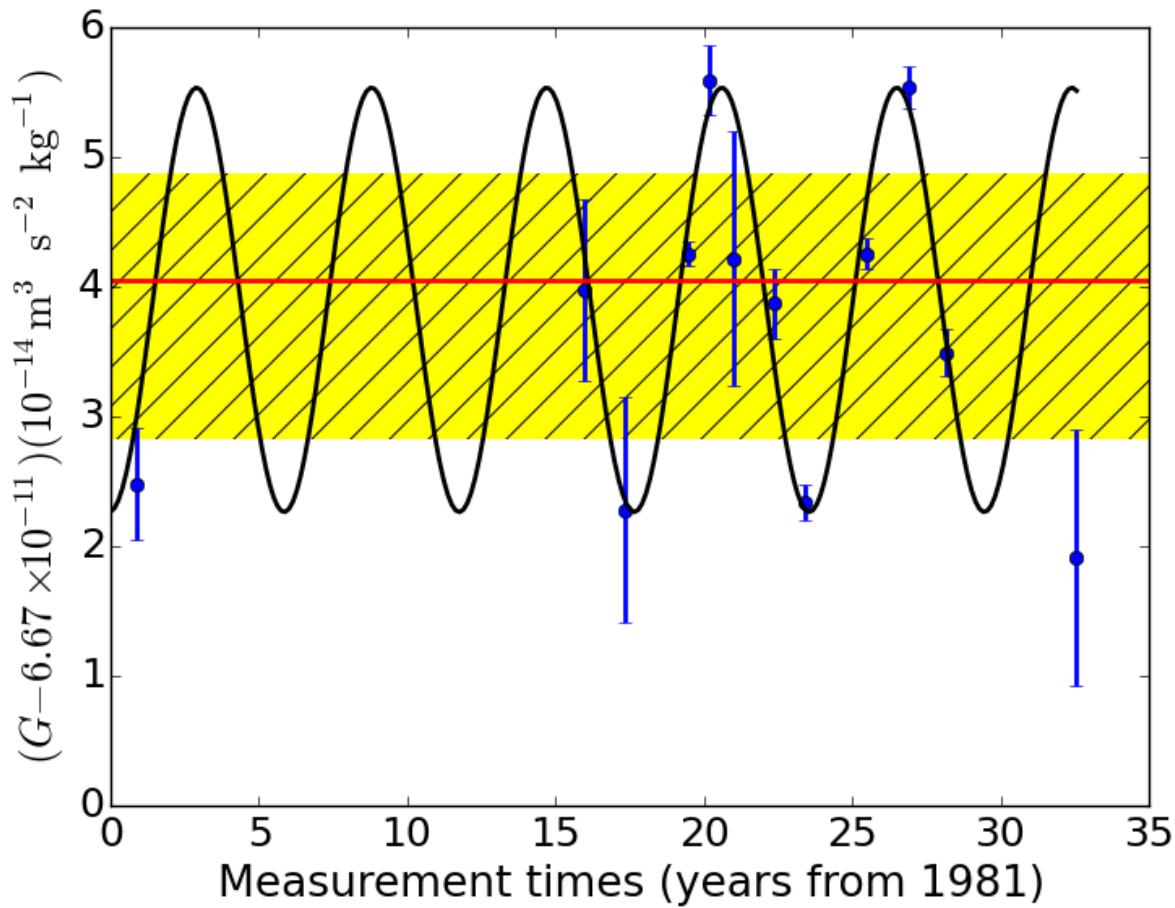
Significance also known as Z-score

$$p = \int_Z^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = 1 - \Phi(Z) \quad \text{TMath::Prob}$$

$$Z = \Phi^{-1}(1 - p) \quad \text{TMath::NormQuantile}$$

In Python : Use `stats.norm.isf(p-value)`

Model Comparison Tests of G variation



arXiv:1607.03845

Model Comparison Tests of G variation

$$y_i = A \sin[\phi_0 + 2\pi(t_i/P)] + \mu_G,$$

- H1. Data is consistent with a constant offset + measured uncertainties
- H2. Same as H1, but an additional unknown systematic offset.
- H3. Data is described by above equation (showing sinusoidal variation)
- H4. Same as H3, but an additional unknown systematic offset.

Hypothesis	μ	σ_{sys}	A	$P(yrs)$	ϕ_0	DOF	χ^2/DOF	$P(\chi^2,\nu)$
H1	6.766×10^{-11}	-	-	-	-	11	28.04	6.8×10^{-60}
H2	6.674×10^{-11}	10^{-14}	-	-	-	10	1.27	0.059
H3	6.674×10^{-11}	-	1.64×10^{-14}	5.9	-0.07	8	2.93	0.0011
H4	6.571×10^{-11}	10^{-12}	1.9×10^{-14}	7.57	0.0011	7	1.71	0.032

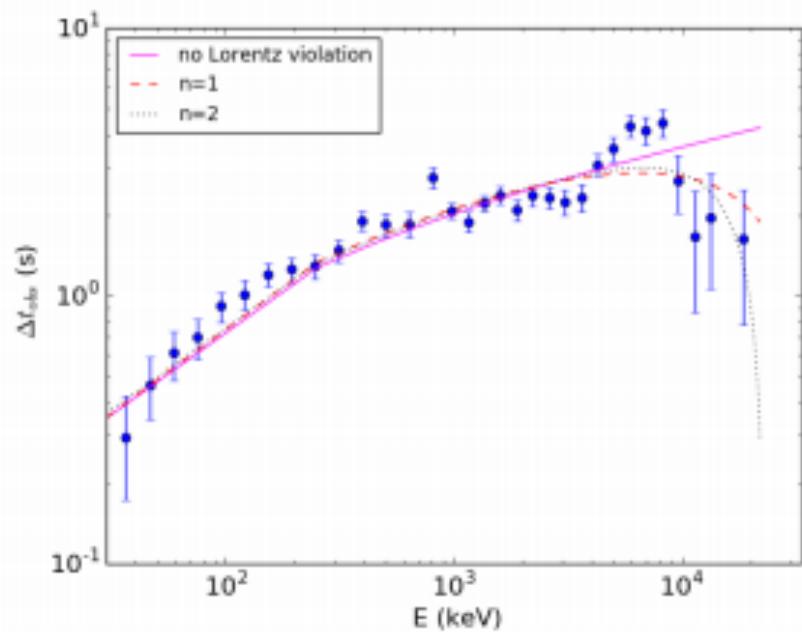


Figure 1 : Summary of the best fit LIV models for $n = 1$ and $n = 2$ along with no Lorentz violation superposed on top of the spectral lag data from GRB 160625B. We note that one data point at $(E, \Delta t) = (15708 \text{ keV}, -0.223 \text{ sec})$ has been omitted for brevity. All the spectral lag data points have been obtained from Table 1 of W17.

	No LIV ^a	(n=1) ^b	(n=2) ^c
Frequentist			
DOF	35	34	34
χ^2/DOF	2.6	2.37	2.23
$\chi^2\text{GOF}$	2.2×10^{-7}	3.7×10^{-6}	1.5×10^{-5}
$p\text{-value}$		0.0014	9.2×10^{-5}
significance		3.05σ	3.74σ
$\Delta \text{ AIC}$		8.2	12.9
$\Delta \text{ BIC}$		6.9	11.7

^a No Lorentz Invariance

^b Lorentz Invariance up to linear ($n=1$) order

^c Lorentz Invariance up to quadratic ($n=2$) order

Information Criterion based tests

Akaike Information Criterion : based on Kullbeck-Leibler information entropy

Bayesian Information Criterion : based on Bayesian evidence

arXiv:1207.5875 Shi, Huang and Lu astro-ph/0701113 Liddle

$$\text{BIC} \equiv -2 \ln \mathcal{L}_{\max} + k \ln N ,$$

$$\text{AIC} \equiv -2 \ln \mathcal{L}_{\max} + 2k ,$$

$$\text{AIC}_c = \text{AIC} + \frac{2k(k+1)}{N - k - 1} .$$

N = no of data points

K = no of free parameters

Strength of Evidence Tests for AIC/BIC

For the AIC, Burnham & Anderson (2003) featured the following “strength of evidence” in the form of $\Delta\text{AIC} = \text{AIC}_i - \text{AIC}_{min}$:

ΔAIC	Level of Empirical Support For Model i
0 – 2	Substantial
4 – 7	Considerably Less
> 10	Essentially None

arXiv:1207. 5875
by Shi et al

For the BIC, Robert & Adrian (1995) featured the following “strength of evidence”, where $\Delta\text{BIC} = \text{BIC}_i - \text{BIC}_{min}$:

ΔBIC	Evidence against Model i
0 – 2	Not Worth More Than A Bare Mention
2 – 6	Positive
6 – 10	Strong
> 10	Very Strong

Other Advanced Model-Comparison Tests

- Deviance Information Criterion
- Takeuchi Information Criterion
- WAIC
- F-test
- Likelihood ratio tests
- Cross-validation
- Bayesian Methods : Bayes Factor, Posterior Odds Ratio

For more advanced details see the following note by Louis Lyons

<https://www-cdf.fnal.gov/physics/statistics/notes/H0H1.pdf>

For more examples of model comparison in Cosmology/Astrophysics, see

Liddle : astro-ph/0701113

Liddle, Mukherjee, Parkinson : astro-ph/0608184

Kerscher and Weller arXiv:1901.07726

Comparison of Distributions

Week 4

Some quotes from astrophysicists

- “Understanding data better is always an unsolved problem in Astrophysics”
Bill Press, astro-ph/9604126
- “We identify the probability of the data given the parameters as the *likelihood* of the parameters given the data. This identification is entirely based on intuition. It has no formal mathematical basis in and of itself; as we already remarked, statistics is *not* a branch of mathematics.”
Bill Press, Numerical Recipes 1992 edition, Chapter 15 on maximum likelihood estimation.
- You may ask “What can a hard headed statistician offer to a starry eyed astronomer?”. The answer is “Plenty”. One normally associates statistics with large numbers, and astronomy is full of large numbers...I have every reason to believe that increased interaction between statistics and astronomy will be to the benefit of both subjects.
Jayant Narlikar to C.R. Rao

Introduction

- Two samples are drawn from the same distribution or whether two sets of measurements imply a difference in measured quantity.
- A sample is consistent as been drawn from a known distribution.
- Two sets of measurements with different measurement errors are drawn have the same mean.

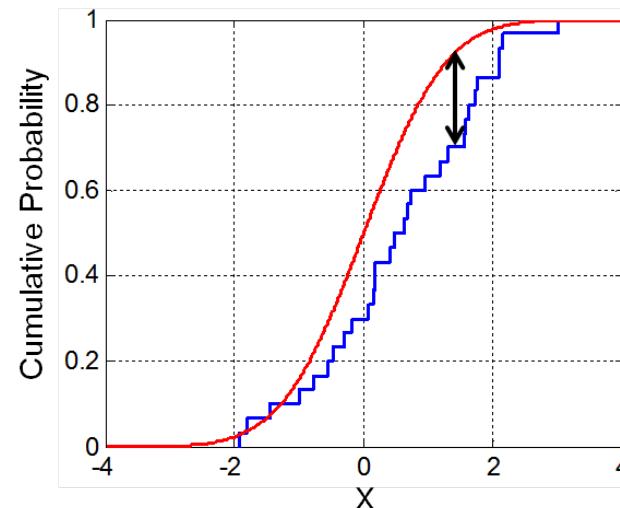
Use data to compute an approximate statistic and compare the data-based value to its expected distribution. The expected distribution is evaluated by assuming that the *null hypothesis is true*.

We now discuss various non-parametric methods for comparing distributions.

Kolmogorov-Smirnov (K-S) tests

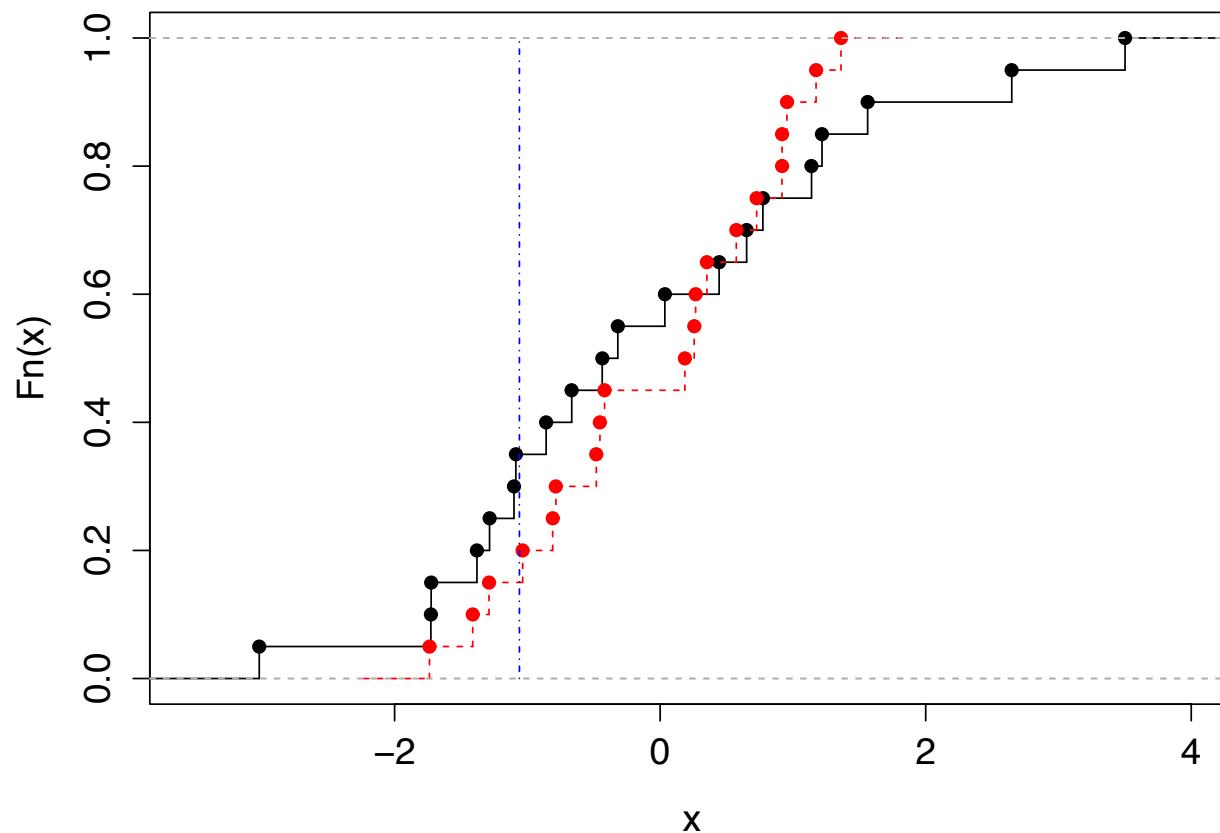
- K-S test compares cumulative distribution function of two samples.
- Consider two samples $\{x_{1i}\}$ where $i=1, \dots, N_1$ and $\{x_{2i}\}$ where $i=1, \dots, N_2$
Sort the sample and divide the rank of x_i by the sample size to get $F(x_i)$
 $F(x)$ is a step function that increases by $1/N$ at each data point $0 \leq F(x) \leq 1$.
- K-S test is based on the maximum distance of the cumulative distribution function $F_1(x_1)$ and $F_2(x_2)$.

$$D = \max |F_1(x_1) - F_2(x_2)|$$



Source: wikipedia

Example of 2-sample KS test



Jessica Cisewski notes at 2016 PSU astrostatistics summer school

Null Hypothesis of KS-test

Qt : How often would the value D computed from the data arise by chance if the two samples were drawn from the *same* distribution (null hypothesis in this case).

The probability of obtaining by chance a value D larger than the observed value is given by the function

$$Q_{KS}(\lambda) = 2 \sum_{k=1}^{\infty} (-1)^{k-1} e^{-2k^2 \lambda^2}$$

$$\lambda = (0.12 + \sqrt{n_e} + \frac{0.11}{\sqrt{n_e}})D$$

$$n_e = \frac{N_1 N_2}{N_1 + N_2}$$

One-Sample K-S test

- Qt : Is the measured $f(x)$ consistent with a known reference distribution function $h(x)$

This is called “one-sample” K-S test and is special case of “two-sample” K-S test discussed previously. In this case $N_2=\infty$

A small value of Q_{ks} indicates that it is unlikely at given confidence level α that the data summarized by $f(x)$ are drawn from $h(x)$.

K-S test is sensitive to the location, scale and shape of the underlying distribution. It is insensitive to reparametrization of the data points.

K-S tests for histograms can be found in arXiv:0804.0380

K-S test in Python

- Lookup `kstest`, `ks_2samp`, `ksone` in `scipy.stats`

```
import numpy as np
from scipy import stats
vals = np.random.normal(loc=0,scale=1,size=1000)
stats.kstest(vals,"norm")
```

0.025, 0.529

D-value = 0.025

p-value = 0.529

Critic of usage of K-S test in astrophysics

- See

<https://asaip.psu.edu/Articles/beware-the-kolmogorov-smirnov-test>

Other non-parametric tests:

Anderson-Darling test

Cramer-Von Mises Test

Watson Test

Gini Coefficient (based on cumulative distribution function) : measures the deviation of a given cumulative distribution function from that expected for a normal distribution.

$$G = 1 - 2 \int_{x_{min}}^{x_{max}} F(x) dx$$

Examples of KS test usage in astrophysics

- Too many examples (homework assignment for you all)

Kuiper test

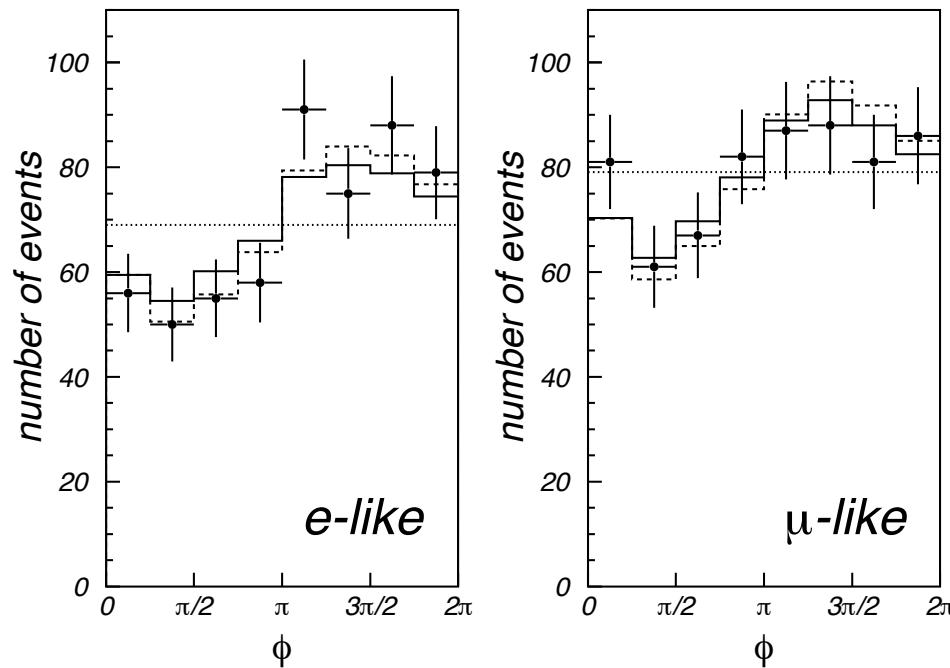
- Simple variant of K-S test to treat distributions defined on a circle. It is based on the statistic

$$D^* = \max\{F_1(x_1) - F_2(x_2)\} + \max\{F_2(x_1) - F_1(x_2)\}$$

For distributions on a circle D^* is invariant to location of origin. Good test for comparing longitude distribution of two astronomical samples.

Null hypothesis probability calculated in the same way as K-S test.

Example of Kuiper Test usage



astro-ph/9901139

East-West asymmetry of atmospheric neutrino flux in Super-Kamiokande

A quantitative comparison between the data and Monte Carlo was performed using a Kuiper test [8,9]. This test calculates a binning and starting-point free probability that observed data are the result of an assumed distribution. The Kuiper statistic V is defined as:

$$V = \max_{0 < \phi < 2\pi} [S_N(\phi) - P(\phi)] + \max_{0 < \phi < 2\pi} [P(\phi) - S_N(\phi)],$$

where ϕ is the azimuthal angle, $S_N(\phi)$ is a cumulative probability function from data and $P(\phi)$ is the one from Monte Carlo. The significance is obtained from the statistic $V^* = V(\sqrt{n} + 0.155 + 0.24/\sqrt{n})$, and defined as

$$Prob = 2 \sum_{j=1}^{\infty} (4j^2 V^{*2} - 1) \exp(-2j^2 V^{*2}),$$

where n is number of events.

From this test, the probability that the azimuthal distribution of the data originated from a flat parent distribution was 0.0008% (20%) for e -like (μ -like) events. The azimuthal distribution of e -like events is inconsistent with a flat distribution at more than 99% CL. Also, the probabilities that the data match the Monte Carlo in shape with the flux of Ref. [3] were 42% for e -like events and 92% for μ -like events. For a Monte Carlo with neutrino

Anderson-Darling Test for Gaussianity.

Qt: Is the *shape* of measured $f(x)$ consistent with a Gaussian distribution?

Aim is to predict distribution of test statistic when null hypothesis is true.

$$A^2 = -N - \frac{1}{N} \sum_{i=1}^N [(2i-1) \ln(F_i) + (2N-2i+1) \ln(1-F_i)]$$

Where F_i is the i^{th} value of the cumulative distribution function of z_i defined as :

$$z_i = \frac{x_i - \mu}{\sigma}$$

Anderson-Darling test in SciPy

scipy.stats.anderson

```
scipy.stats.anderson(x, dist='norm')\[source\]
```

Anderson-Darling test for data coming from a particular distribution

The Anderson-Darling test is a modification of the Kolmogorov-Smirnov test [kstest](#) for the null hypothesis that a sample is drawn from a population that follows a particular distribution. For the Anderson-Darling test, the critical values depend on which distribution is being tested against. This function works for normal, exponential, logistic, or Gumbel (Extreme Value Type I) distributions.

Parameters: `x : array_like`
array of sample data
`dist : {'norm', 'expon', 'logistic', 'gumbel', 'gumbel_l', 'gumbel_r', 'extreme1'}`, optional the type of distribution to test against. The default is 'norm' and 'extreme1', 'gumbel_l' and 'gumbel' are synonyms.

Returns: `statistic : float`
The Anderson-Darling test statistic
`critical_values : list`
The critical values for this distribution
`significance_level : list`
The significance levels for the corresponding critical values in percents. The function returns critical values for a differing set of significance levels depending on the distribution that is being tested against.

This is also coded in statsmodels

Lookup

`statsmodels.stats.diagnostic.normal_ad`

Anderson-Darling Test for Gaussianity

```
From scipy import stats  
X = np.random.normal(0,1,size=1000)  
A,crit,sig = stats.anderson(X,'norm')
```

A-D test in `scipy.stats` can also be used to test for exponential, logistic, Gumbel distribution
Note that there is also a k-sample -sample A-D test called `anderson_ksamp` in `scipy`

Shapiro-Wilk test.

$$W = \frac{\left(\sum_{i=1}^N a_i R_i \right)^2}{\sum_{i=1}^N (x_i - \bar{x})^2}$$

R_i is the rank of the sample and a_i is the expected values of the order statistics for random variables sampled from the normal distribution

Lookup `stats.shapiro` in `scipy`. A value of W close to 1 indicates that distribution is entirely gaussian

Outliers in Gaussian Distributions

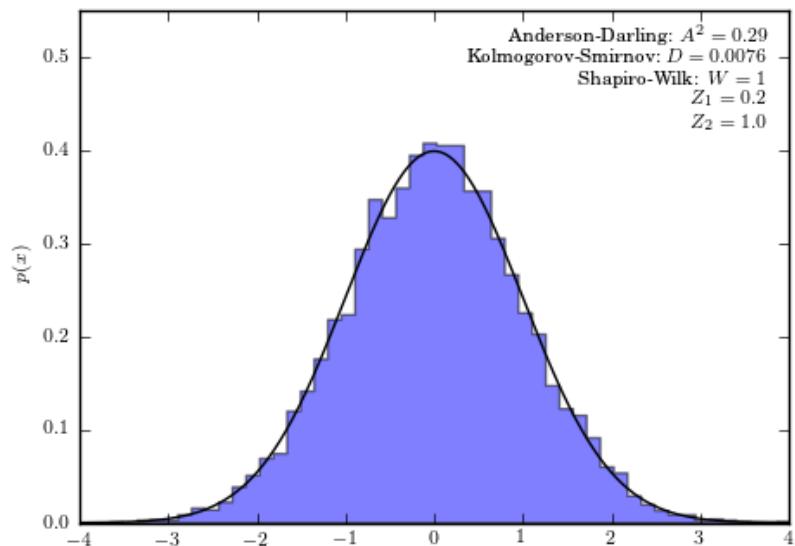
- Compare sample standard deviation and σ_G
- For a Gaussian distribution when $N > 100$, s/σ_G follows a Gaussian distribution with $\mu \sim 1$ and $\sigma = 0.92/\sqrt{N}$
- Difference between mean and median drawn from a Gaussian distribution follows a Gaussian distribution with $\mu \sim 0$ and $\sigma \sim 0.76/\sqrt{N}$

Define $Z_1 = 1.3 |\mu - q_{50}| \sqrt{N} / s$ and $Z_2 = 1.1 \left| \frac{s}{\sigma_G} - 1 \right| \sqrt{N}$

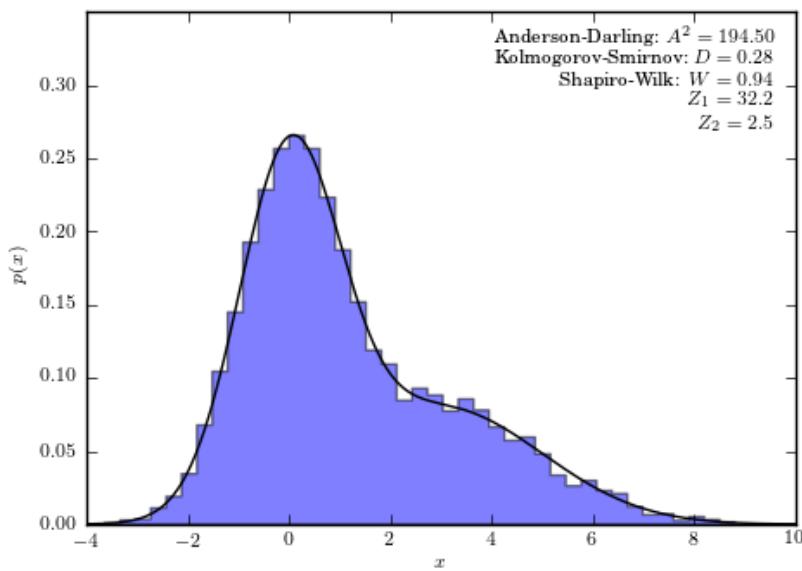
obey Gaussian distribution with mean = Z_1 and std. deviation = Z_2

Cheat-Sheet of Tests for Gaussianity

- For data which depart from Gaussian distributions :
 - A-D test ($A^2 > 1$)
 - K-S test ($D > 1/\sqrt{N}$)
 - Shapiro-Wilk's test ($W < 1$)
 - $Z_1, Z_2 > (\text{many times}) \sigma$
- For an example from astrophysics (dynamical state of galaxy groups) see arXiv:0908.0938



Normal Distribution



Combination of two normal distributions

Astroml figure 4.7

Modeling Non-Gaussianity

- In case our empirical distributions fail tests for Gaussianity, but if there is no strong motivation for choosing a specific alternative distribution.
- Model Non-Gaussian distribution by Gram-Charlier series

$$h(x) = \mathcal{N}(\mu, \sigma) \sum_{k=0}^{\infty} a_k H_k(z)$$

where

$$z = \frac{x - \mu}{\sigma}$$

$H_k(z)$ are Hermite polynomials, which form an orthonormal set

and

$$a_k = \frac{1}{k!} \int_{-\infty}^{\infty} f(x) H_k(x) dx$$

RC Boston 1971, IEEE

Mann-Whitney Test/U Test/Wilcoxon rank sum tests

- Used to check if two datasets are drawn from distributions with different location parameters (or different means). (In case the two datasets are drawn from Gaussian distributions, you can use t-test).
- Calculation of U is obtained from the ranks of the two samples after concatenating them (will skip details of calculation. consult wikipedia)

```
>>> import numpy as np
>>> from scipy import stats
>>> x,y=np.random.normal(0,1,size=(2,1000))
>>> stats.mannwhitneyu(x,y)          (See also stats.ranksums)
MannwhitneyuResult(statistic=488708.0, p-value=0.19094553275490711)
```

U-test expected result is close to the expected $N_1 N_2 / 2$ if they are drawn from the same distribution with Same mean. See <https://data.library.virginia.edu/the-wilcoxon-rank-sum-test/> for an illustrative example

Wilcoxon signed-rank test

Special case of comparing the means of two data sets is when the data sets have the same size ($N_1 = N_2 = N$) and data points are paired. Non-parametric test used to compare means of two distributions is the Wilcoxon signed-rank test. This test cannot be used when data points are not equal

- Calculate $y_i = x_{1i} - x_{2i}$ and exclude all values with $y_i=0$
- Order the new sample by $|y_i|$ resulting in the rank R_i for each pair
- Each pair is assigned $\Phi_i=1$ if $x_{1i} > x_{2i}$ and 0 otherwise.
- Calculate W_+ as follows

$$W_+ = \sum_i^m \Phi_i R_i$$

All ranks with $y_i > 0$ are summed

Similarly calculate W_- and Wilcoxon signed rank test (T) is smaller of the two.

p-value of Wilcoxon signed-rank test

For value of $n > 20$, behaviour of T can be approximated by Gaussian distribution $\mathcal{N}(\mu, \sigma)$

where :

$$z = \frac{T - \mu}{\sigma}$$

$$\mu = \frac{N(2N + 1)}{2}$$

$$\sigma = N \sqrt{\frac{2N + 1}{12}}$$

Wilcoxon signed-rank test in scipy

```
>>> import numpy as np  
>>> from scipy import stats  
>>> x,y = np.random.normal(0,1,size=(2,1000))  
>>> T,p= stats.wilcoxon(x,y)  
>>> T,p # T is the value of smaller value of W+ and W-  
(243583.0, 0.46552103910817078)
```

Difference between Wilcoxon signed-rank test and Wilcoxon rank-sum test

19

12

I was wondering what the theoretical difference is between the Wilcoxon Rank-Sum Test and the Wilcoxon Signed-Rank Test using paired observations. I know that the Wilcoxon Rank-Sum Test allows for different amount of observations in two different samples, whereas the Signed-Rank test for paired samples does not allow that, however they both seem to test the same in my opinion. Can someone give me some more background / theoretical information when one should use the Wilcoxon Rank-Sum Test and when one should use the Wilcoxon Signed-Rank Test using paired observations?

You should use the signed rank test when the data are *paired*.

You'll find many definitions of pairing, but at heart the criterion is something that makes pairs of values at least somewhat positively dependent, while unpaired values are not dependent. Often the dependence-pairing occurs because they're observations on the same unit (repeated measures), but it doesn't have to be on the same unit, just in some way tending to be associated (while measuring the same kind of thing), to be considered as 'paired'.

You should use the rank-sum test when the data are *not* paired.

That's basically all there is to it.

More details available in <http://tinyurl.com/j37sts5>

Parametric Methods for comparing distributions (t-test)

- If two datasets are drawn from Gaussian distributions with different means, but same sigma and sigma is known , you can calculate p-value of the difference in means using error propagation.
- If we do not know sigma, calculate $\Delta = \bar{x_1} - \bar{x_2}$ and $M_s = \frac{\Delta}{s_\Delta}$ where

$$s_\Delta = \sqrt{s_1^2/N_1 + s_2^2/N_2} \quad \text{follows Student's t-distribution. DOF} = N_1+N_2-2$$

For large samples, this asymptotes to the case of distributions with known sigma (or Gaussian distribution)

If the two datasets are drawn from two underlying distributions with different variances, then appropriate test is called Welch's t-test.

t-test in Python

- Different variants of t-test in Python called `ttest_rel`, `ttest_ind`, and `ttest_1samp`
See documentation for details.

```
>>> import numpy as np
>>> from scipy import stats
>>> x, y = np.random.normal(size=(2,1000))
>>> t,p = stats.ttest_ind(x,y)
>>> t,p
(0.93039549523650522, 0.35227874770046763)
```

Comparison of Gaussian variances using F-test

- F test is used to compare variances of two unspecified distributions. The null hypothesis is that variances of the two samples are equal and the statistics is based on the ratio of the sample variances

$F = s_1^2/s_2^2$ follows Fisher F distribution with $d_1 = N_1 - 1$ and $d_2 = N_2 - 1$

```
>>> import numpy as np  
>>> from scipy import stats  
>>> x, y = np.random.normal(size=(2, 10000))  
>>> F, p = stats.f_oneway(x, y)
```

- p
- 0.85157563615482457
- F
- 0.035010462229159989

Efficacy of these tests

In astrophysics a comparative study of these tests for detecting non-gaussianity was done in
<https://arxiv.org/abs/0908.0938>

Choosing histogram bin width

- Three different rules for choosing bin width:

- Scott Rule
- Freedman rule
- Knuth Rule

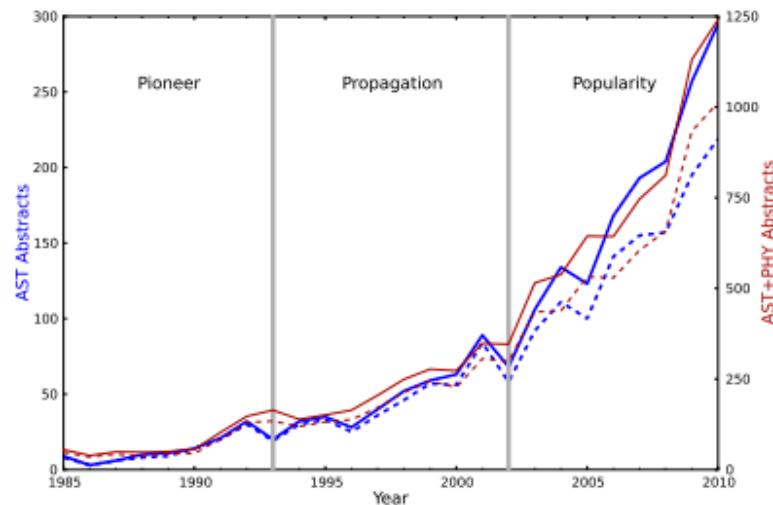
```
from astroML.plotting import hist  
hist(x,bins='freedman') # can also choose knuth or  
scott
```

Bayesian Analysis

Weeks 5+6

Introduction to Bayesian Analysis

- This is a huge subject with potential for year-long course.



Tom Loredo arxiv:1208.3036

Fig. 1 Simple bibliometrics measuring the growing use of Bayesian methods in astronomy and physics, based on queries of the NASA ADS database in October 2011. Thick (blue) curves (against the left axis) are from queries of the astronomy database; thin (red) curves (against the right axis) are from joint queries of the astronomy and physics databases. For each case the dashed lower curve indicates the number of papers each year that include “Bayes” or “Bayesian” in the title or abstract. The upper curve is based on the same query, but also counting papers that use characteristically Bayesian terminology in the abstract (e.g., the phrase “posterior distribution” or the acronym “MCMC”); it is meant to capture Bayesian usage in areas where the methods are well-established, with the “Bayesian” appellation no longer deemed necessary or notable.

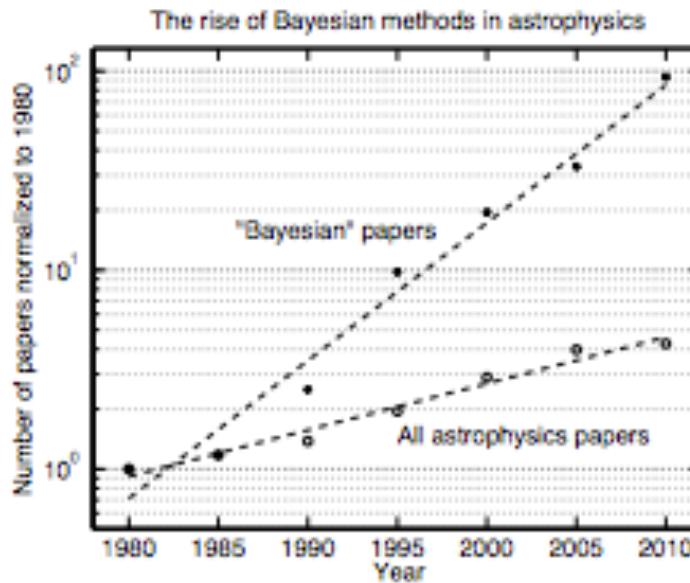
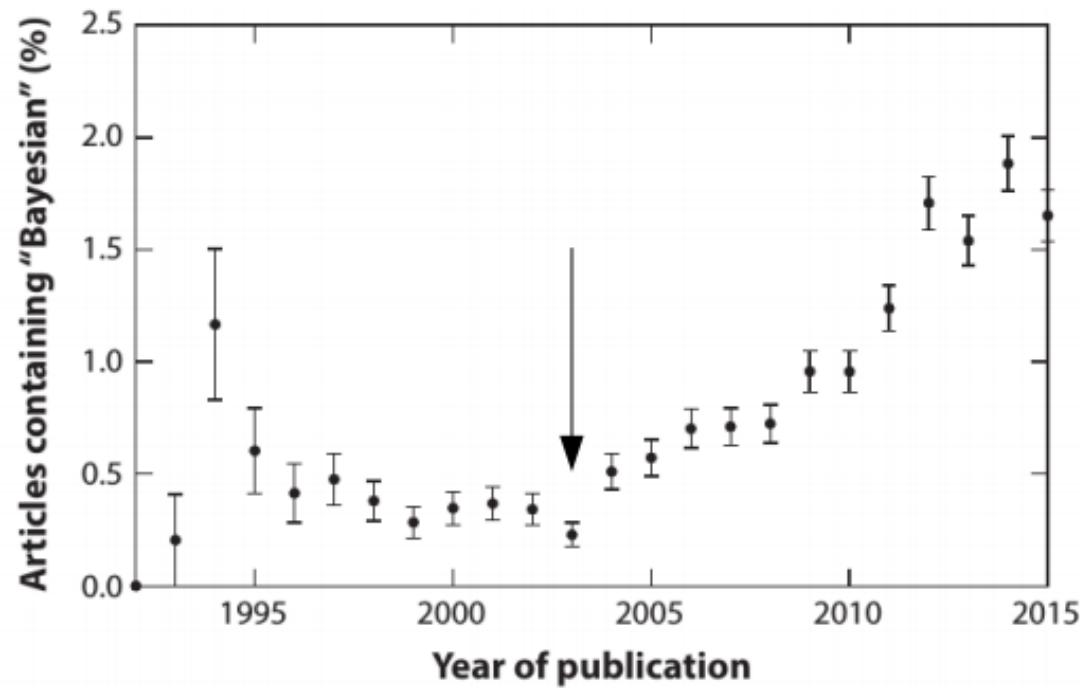


Fig. 3 Number of articles in astronomy and cosmology with “Bayesian” in the title, as a function of publication year (upper data points) and total number of articles (lower data points) as a function of publication year. Numbers are normalized to 1980 levels for each data series. The number of Bayesian papers doubles every 4.3 years, while the total number of papers doubles “only” every 12.6 years. At the present rate, by 2060 all papers on the archive will be Bayesian. (source: NASA/ADS).

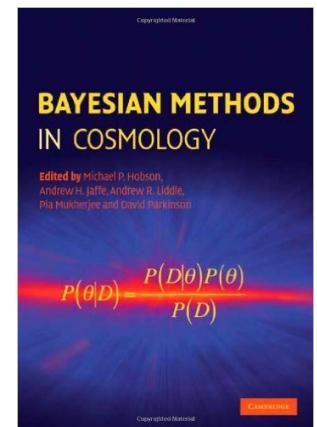
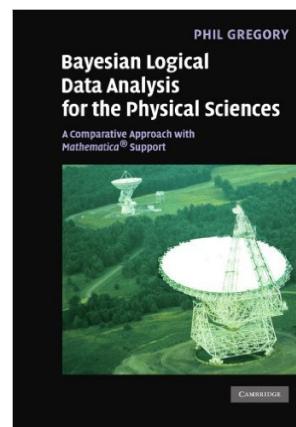
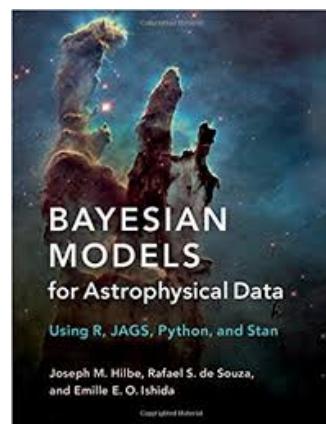
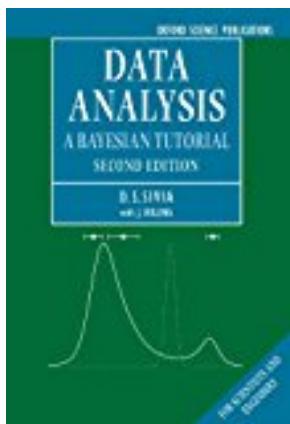
Bayesian Literature in Astronomy



Sanjib Sharma seminar @ IITH Jan 2020

Books on Bayesian Analysis (Physicist Pov)

- Data Analysis – A Bayesian tutorial by D.S. Sivia and John Skilling
- Bayesian Logical Data Analysis for Physical Sciences
- Online book by E.T. Jaynes (Probability Theory, Logic of Science)
- Bayesian Methods in Cosmology by Michael Hobson et al



Arxiv resources on Bayesian Analysis

- <https://arxiv.org/abs/0803.4089> Roberto Trotta
- <https://arxiv.org/abs/1701.01467> Trotta (Bayesian methods in Cosmology)
- <https://arxiv.org/abs/1301.1273> Louis Lyons (particle physics)
- <https://arxiv.org/abs/1302.1721> Andrew Liddle
- <https://arxiv.org/abs/0911.3105> Licia Verde (Statistical Methods in Cosmology)
- <https://arxiv.org/abs/1208.3036> Tom Loredo (Bayesian Astrostatistics)
- <https://arxiv.org/abs/1012.3589> Glenn Cowan (particle physics)
- <https://arxiv.org/abs/1411.5018> J Vanderplas (Practical Introduction) (Easy to read)
- <https://arxiv.org/abs/1809.02293> Eric Thrane & Colm Talbot(for GW astronomy)
- <https://arxiv.org/abs/2302.04703> Practical guide for Bayesian Inference in Astronomy

Also lectures on Bayesian statistics in PSU summer school
Jake Van der Plas blog article

Introduction to Bayesian Statistics

- First introduced by Rev. Thomas Bayes (1702-1761) who wrote an article on how to combine an initial belief with new data to arrive at an improved belief
- Probability statements can be made for model parameters and models themselves.
- Both Classical (and frequentist) statistics use data likelihood. In frequentist statistics, likelihood function is used to find model parameters with the highest data likelihood. But likelihood function cannot be interpreted as PDF for the model parameters.
- Bayesian statistics extends the concept of data likelihood function by adding extra (or *prior*) information to the analysis and assigning pdfs to all model parameters and the models themselves.

Limitations of Max. Likelihood Estimates

Imagine you arrive at a bus stop (with no knowledge of the bus schedule) and the next bus arrives t minutes later.

Q: What is the mean time τ between two successive buses , if the buses keep a regular schedule?

Ans: Wait time is distributed uniformly in the interval $0 \leq t \leq \tau$ and average wait time= $\tau/2$,
or $\tau = 2t$

Max Likelihood Estimates

Probability that you shall wait t minutes (likelihood of the data) is given by the uniform distribution

$$p(t|\tau) = \begin{cases} 1/\tau & \text{if } 0 \leq t \leq \tau \\ 0 & \text{otherwise} \end{cases}$$

Because there is only a single data point, the data likelihood is simple equal to this probability. Maximum of $p(t|\tau)$ occurs for $\tau = t$ (or when τ is smallest)

Expectation value of τ or its median value diverges

Problem is solved when we introduce priors.

Essence of Bayesian idea

$$p(M|D) = \frac{p(D|M)p(M)}{p(D)}$$

Bayes theorem quantifies the rule for combining initial belief with new data to arrive at improved belief and “improved belief” is the product of “initial belief” and the probability that “initial belief” generated the observed data.

$$p(M, \theta | D, I) = \frac{p(D|M, \theta, I)p(M, \theta | I)}{p(D|I)}$$

where model M includes k model parameters θ_p . (note that M and θ are used interchangeably)

- $P(M, \theta | D, I)$ is the *posterior* pdf for model M and some prior information I.
- $P(D | M, \theta)$ is the *likelihood* of the data given some model M and some fixed value of parameters θ
- $P(M, \theta | I)$ is the a priori joint probability for model M and its parameters θ in the absence of any of the data used to compute the likelihood and is often simply called the *prior*

Prior can be expanded as follows $p(M, \theta | I) = p(\theta | M, I)p(M | I)$

Usually for parameter estimation problems, we only need to specify $P(\theta | M, I)$

- $P(D|I)$ is *probability of the data* or the prior predictive probability for D.

Provides normalization for the posterior pdf and is not explicitly computed when estimating model parameters.

Instead $P(M, \theta|D, I)$ for a given model is renormalized so that its integral over all model parameters is unity.

Integral of the prior $P(\theta|M, I)$ over all parameters should also be equal to unity.

For parameter estimation, calculation of posterior pdf done with an arbitrary normalization

One exception is model comparison where correct normalization of numerator in Bayes's formulae is important.

Why is Bayesian Interpretation Controversial

- $P(M, \theta | D, I)$ is not a probability in the strict sense (compared to the likelihood in frequentist statistics) . We accept that $P(M, \theta | D, I)$ corresponds to the state of our knowledge (or degree of belief) about a model and its parameters given data D and prior information I . This introduces the notion of probability for models and model parameters.

Bus-Stop Example

- Assume prior is proportion to $1/\tau$. Posterior probability density function is given by :

$$p(\tau|t, I) = \begin{cases} t/\tau^2 & \text{if } \tau \geq t \\ 0 & \text{otherwise} \end{cases}$$

Note that the extra t is added so that integral of posterior is unity.

$$\int_t^\infty p(\tau|t, I)d\tau = \int_t^\infty \frac{C}{\tau^2}d\tau = 1 \quad \longrightarrow \quad C = t$$

Median value of τ for $p(\tau|t, I)$ is given by $2t$ (in agreement with expectation)

Also $p\%$ quantiles are equal to $(1 - t/\tau_p)$

For more on the bus-stop example see

<http://jakevdp.github.io/blog/2018/09/13/waiting-time-paradox/>

Summary of Bayesian Statistical Inference

Notation : Replace $M(\theta)$ with M whenever the absence of explicit dependence on θ is not confusing.

- Construction of the Data likelihood $P(D | M)$
- Choice of the prior incorporating previous information that might exist, but is not used in computing the likelihood. $P(\theta | M, I)$
- Determination of posterior pdf using Bayes theorem. $P(M | D)$
This step is computationally intensive for multi-dimensional posterior problems. Also $P(D | I)$ is not explicitly stated because $P(M | D)$ can be properly normalized by renormalizing the product $P(D | M) P(M)$
- Best model parameters are found by maximizing $P(M | D)$ which yield maximum *aposteriori estimate*. This *point estimate* is the natural analog to maximum likelihood estimation (MLE) from classical statistics.

- Uncertainty in parameter estimates is obtained via *credible regions* (Bayesian counterpart to frequentist confidence region). Obtained by integrating the posterior. Various numerical methods can be used to simulate samples from the posterior.
- Hypothesis testing as needed to make conclusions about model or parameter estimates.

Bayesian approach can be thought of as formalizing the process of continually refining our state of the knowledge about the world, beginning with no data and then updating that by multiplying by the likelihood once the data is observed to obtain the posterior.

Posterior estimates and Marginalization

Alternately calculate the Bayesian posterior means as follows :

$$\bar{\theta} = \int \theta p(\theta|D) d\theta$$

For a vector of parameters θ (i.e. one particular value) is obtained from $P(M, \theta|D, I)$ by the process of *marginalization* , i.e. by integrating $P(M, \theta|D, I)$ over all the other unknown parameters and then renormalization

$$\int p(\theta|D) d\theta = 1$$

Although there is no natural way to marginalize over nuisance parameters
in frequentist statistic,
However , profile likelihood is used (particle physics literature)

$$\mathcal{L}(\theta_1) \equiv \max_{\theta_2, \dots, \theta_N} \mathcal{L}(\theta),$$

Marginalization of Parameters

Consider posterior pdf $P(M, \theta_1, \theta_2, \dots, \theta_k)$. Suppose we are only interested in θ_1 . In order to obtain the posterior pdf for θ_1 , we integrate over all the uninteresting (or nuisance) parameters. This integration procedure is called the process of marginalization and the resulting pdf is called *marginal posterior pdf*.

$$p(M, \theta_1 | D) = \int p(M, \theta_1, \theta_2, \dots, \theta_k | D) d\theta_2 \dots d\theta_k$$

Convergence of Posterior with More data

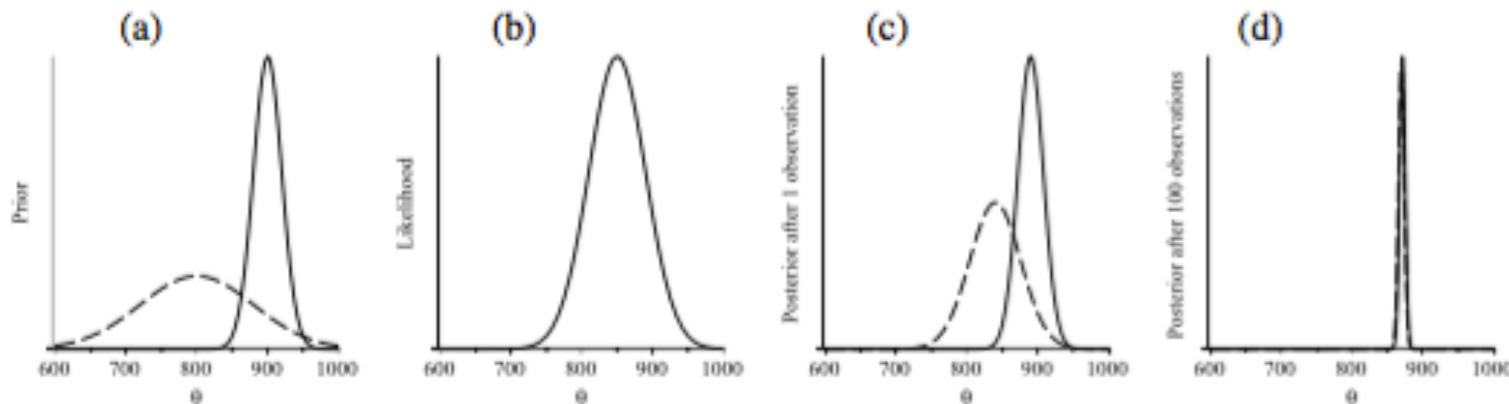


Fig. 4 Converging views in Bayesian inference. Two scientists having different prior beliefs $p(\theta)$ about the value of a quantity θ (panel (a), the two curves representing two different priors) observe one datum with likelihood $\mathcal{L}(\theta)$ (panel (b)), after which their posteriors $p(\theta|d)$ (panel (c), obtained via Bayes Theorem, Eq. (4)) represent their updated states of knowledge on the parameter. This posterior then becomes the prior for the next observation. After observing 100 data points, the two posteriors have become essentially indistinguishable (d).

Bayesian Priors

- Prior incorporates all other knowledge that might exist, but is used when computing the likelihood. Data may chronologically proceed the information in the prior

Examples of priors include knowledge from prior measurements of the same type as data at hand or different measurements that constrain the same quantity whose posterior pdf we are trying to constrain with new data. We may know the mass of an elementary particle from previous measurements with uncertainty.. Priors that incorporate information based on other measurements are called *informative priors*

Priors Assigned by Formal Rules

- When no information is available except for the data been analyzed then such priors are called non-informative priors.

Consider a *flat prior* $p(\theta) \propto \text{constant}$ since $\int p(\theta|I)d\theta = \infty \rightarrow$ this is not a pdf

This is considered an improper prior. Not a problem as long as the resulting posterior is a well-defined pdf.

Principle of indifference : A set of mutually exclusive possibilities need to be assigned equal probabilities. For a fair die prior = 1/6

Principle of consistency: Prior for a location parameter should not change with translation of coordinate system and yields a flat prior. Prior for scale parameter should not depend on choice of units eg if scale parameter is σ and rescaling by a constant factor gives $p(\sigma|I)d\sigma = p(a\sigma|I)d(a\sigma)$

Solution to this is $P(\sigma|I) \propto \sigma^{-1}$ (Jeffreys prior)

Principle of Maximum Entropy

- Entropy measures the information content of a pdf. Entropy for a pdf defined by N discrete values is given by :

$$S = - \sum_{i=1}^N p_i \ln(p_i)$$

This is also called Shannon's entropy. This can be generalized to a continuous distribution (Sivia 2006)

$$S = - \int_{-\infty}^{\infty} p(x) \ln \left(\frac{p(x)}{m(x)} \right) dx$$

where the ``Lebesgue measure'' $m(x)$ ensures that entropy is invariant under change of variables. (Sivia 06)

Principle of Maximum Entropy (contd)

- By maximizing entropy over a suitable set of pdfs, we find the distribution that is least informative (given the constraints)
- Power of the principle comes from a straightforward ability to add additional information about the prior distribution, such as the mean value and the variance.

Example : Consider a 6-faced die where we need to assign 6 prior probabilities. From principle of indifference prior probability is $1/6$. But suppose we know the mean value (for a large number of rolls) then we need to adjust the prior probabilities to be consistent with this information.

$$\sum_{i=1}^6 ip_i = \mu$$

$$\sum_{i=1}^6 p_i = 1$$

Maximize

$$Q = -\sum_{i=1}^6 p_i \ln \left(\frac{p_i}{m_i} \right) + \lambda_0 (1 - \sum_{i=1}^6 p_i) + \lambda_1 (\mu - \sum_{i=1}^6 ip_i)$$

- where m_i is the values of p_i when no additional information is known (in this case 1/6). Solution is given by

$$p_i = m_i \exp(-1 - \lambda_0) \exp(i\lambda_1)$$

λ_0 and λ_1 can be obtained from the constraint equations.

Generalizes to Poisson distribution if no of discrete events is infinite with mean μ

For the continuous case maximum entropy solution for the prior is given by :

$$p(\theta|\mu) = \frac{1}{\mu} \exp\left(-\frac{\theta}{\mu}\right)$$

obtained by assuming a flat distribution on $m(\theta)$ (when no constraint on mean value) and assuming we know the expectation value of

$$\mu = \int \theta p(\theta) d\theta$$

Kullback-Leibler Divergence

$$KL = \sum_i p_i \ln \left(\frac{p_i}{m_i} \right) \text{ and analogously for continuous case}$$

KL divergence is sometimes called KL distance between the two pdfs.
But it is NOT the metric distance since value is not the same when $p(x)$ and $m(x)$ are switched.

In Bayesian statistics KL divergence can be used to measure the information gain
When moving the prior distribution to posterior distribution. KL divergence also
used in information theory.

Conjugate Priors.

- If posterior probability has the same functional form as likelihood for a given prior, these priors are called conjugate priors. and represent a convenient way for generalizing computations.

Likelihood	Conjugate Prior
Gaussian	Gaussian
Binomial	Beta distribution
Poisson	Gamma Distribution
Multinomial	Dirichlet priors

Bayesian Parameter Uncertainty Quantification

To obtain Bayesian *credible region* estimate, we find a and b such that

$$\int_{-\infty}^a f(\theta) d\theta = \int_b^\infty f(\theta) d\theta = \alpha/2$$

The probability that the true values of θ is in the interval (a,b) is equal to $1-\alpha$
In analogy with classical confidence interval and (a,b) is called a $1-\alpha$
Posterior interval

Difference between Confidence/Credible Regions

- Bayesianism : Probabilistic statement about *model parameters* given a *fixed credible region*.

Given our observed data, there is a 95% probability that the true value of θ lies within the credible region.

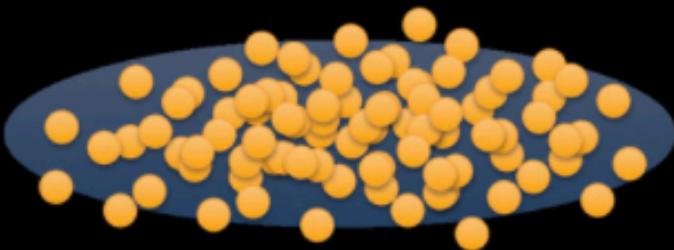
- Frequentism : Probabilistic statement about a recipe *for generating confidence intervals* given a *fixed model parameter*.

If our experiment is repeated many times, in 95% of these cases, the computed confidence interval will contain the true θ

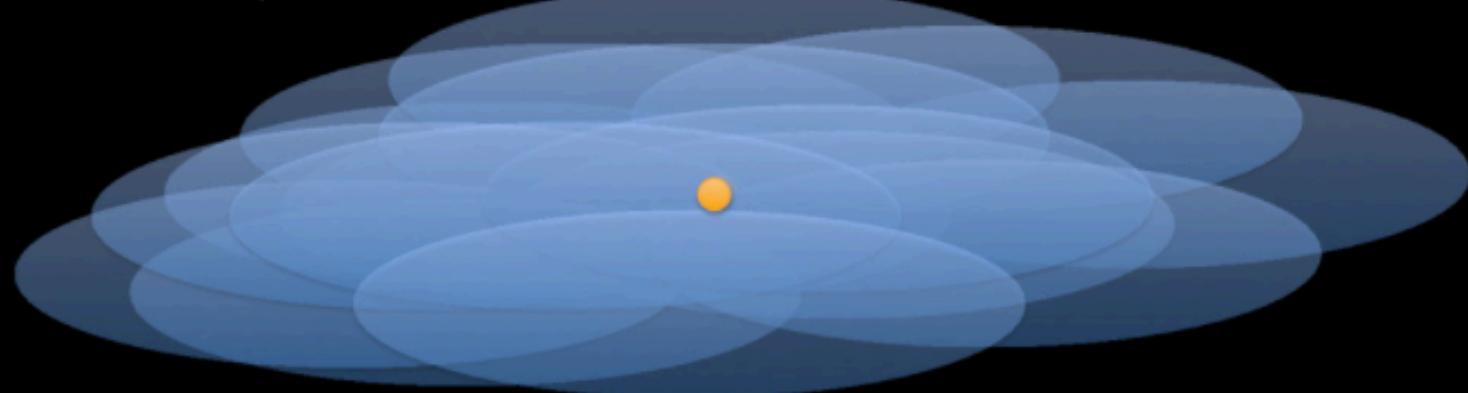
Confidence vs. Credibility

- = Parameter
- = Interval

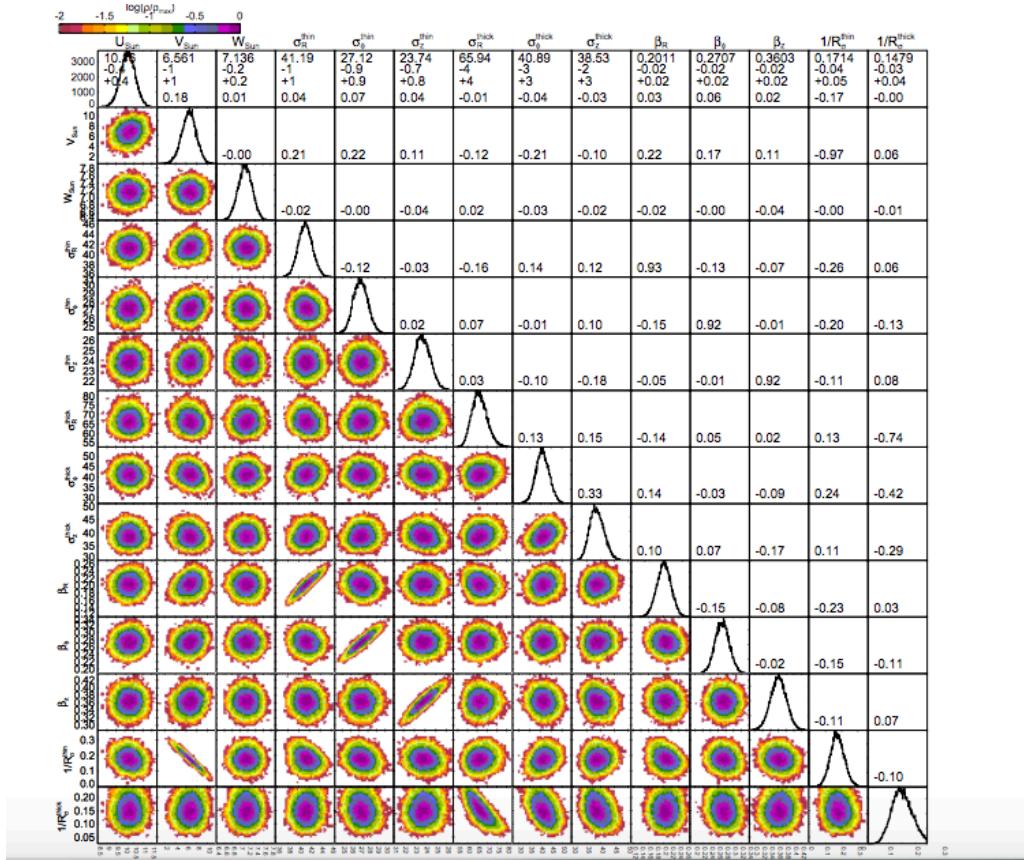
Bayesian Credible Region:



Frequentist Confidence Interval:



Examples of Marginalized Posteriors



arXiv:1405.7435

Bayesian Model Selection

- To find out which of two models say M_1 and M_2 are supported by data we compare their posterior probabilities via the *odds ratio* in favor of model M_2 over model M_1 (sometimes also called posterior odds)

$$O_{21} = \frac{P(M_2|D)}{P(M_1|D)}$$

Where the posterior probability that the model M is correct given data D is given by

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

where

$$E(M) \equiv p(D|M) = \int p(D|M, \theta)p(\theta|M)d\theta$$

is called the **evidence** or **marginal likelihood** or **Bayesian evidence**. It quantifies the Probability that the data D would be observed *if* the model M were the correct model

Odds Ratio becomes :

$$O_{21} = \frac{E(M_2)p(M_2)}{E(M_1)p(M_1)} = B_{21} \frac{p(M_2)}{p(M_1)}$$

where

$$B_{21} = \frac{\int p(D|M_2, \theta_2)p(\theta_2|M_2)d\theta_2}{\int p(D|M_1, \theta_1)p(\theta_1|M_1)d\theta_1}$$
 Bayes Factor

Interpretation of Odds Ratio

- Jeffreys Scale used as a qualitative guide to decide between model 2 and model 1

K	dHart	bits	Strength of evidence	$K = \text{Bayes Factor}$
$< 10^0$	< 0		negative (supports M_2)	
$10^0 \text{ to } 10^{1/2}$	0 to 5	0 to 1.6	barely worth mentioning	
$10^{1/2} \text{ to } 10^1$	5 to 10	1.6 to 3.3	substantial	
$10^1 \text{ to } 10^{3/2}$	10 to 15	3.3 to 5.0	strong	
$10^{3/2} \text{ to } 10^2$	15 to 20	5.0 to 6.6	very strong	
$> 10^2$	> 20	> 6.6	decisive	

wikipedia (note that model M_1 is in the numerator in wikipedia definition. of Bayes factor)

Open Questions/Limitations of Bayesian Analysis

- Criticism of Bayesian Model selection by Efsthathiou (for cosmology) ([arXiv:0802.3185](#)) and Bob Cousins (particle physics) ([arXiv: 0807.1330](#)) , where not enough care is given to selection of models and priors and posterior model probabilities maybe function of unjustified assumptions.
- How to deal with Lindley's paradox (where Bayesian and frequentist hypothesis testing lead to contradictory answers). ([See arXiv:1310.3791](#))
- How to assess the completeness of a set of known models? Is there a principled way of constructing an *absolute* scale for model performance in a Bayesian context? (see [arxiv:1511.02363 for some research on this](#))
- Is Bayesian model averaging (model independent estimation of parameters) useful ?
- Is there such a thing as a correct prior?
- Many posteriors used in astronomical literature are improper [arXiv:1712.03549](#)
- One should also consider fluctuations in Bayesian evidence [arXiv:2102.09547](#)

Introduction to Markov Chain Monte-Carlo

Week 6+

Markov Chain Monte-Carlo

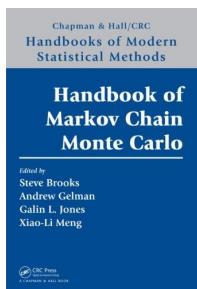
- 3264 papers in astro-ph (in arXiv) which have MCMC in abstract ! First used in 1994 in Astrophysics but widespread after 2002
- Most widely used sampler is Emcee (also recently check out Zeus)
<http://dan.iel.fm/emcee/current/>

For R-based MCMC resources see PSU astrostats summer school notes by M. Haran

MCMC Resources

- See the literature on Bayesian analysis (mentioned in Lecture 6 notes)
- We shall follow Trotta's Jan 2017 review on Bayesian([arXiv:1701.01467](https://arxiv.org/abs/1701.01467))
- Another recent review by Hogg et al ([arXiv:1710.06068](https://arxiv.org/abs/1710.06068))
- Another review (including history) by Sanjib Sharma ([arxiv:1706.01629](https://arxiv.org/abs/1706.01629))
- <https://jellis18.github.io/post/2018-01-02-mcmc-part1/>
- <https://arxiv.org/abs/1909.12313> A conceptual introduction to MCMC methods
- For more advanced stuff :

<https://www.coursera.org/learn/statistical-mechanics>



<http://www.mcmchandbook.net/HandbookTableofContents.html>

History of MCMC

- Brand new Monte Carlo coined by Metropolis & Ulam (1949) while discussing a stochastic method making use of random numbers
- Sampling methods developed in 1953 by Metropolis for calculating problems in statistical mechanics involving canonical ensemble. Probability density for the system being at E for density of states $g(E)$ in thermal equilibrium is given by

$$P(E) = \frac{e^{-\beta E} g(E)}{Z} \quad Z = \int e^{-\beta E} g(E) dE$$

- Hastings realized in 1970 mathematically the above equation same as Bayes theorem with
 - P(E) ~ Posterior
 - Z ~ Evidence
 - $e^{-\beta E}$ ~ Likelihood
 - $g(E)$ ~ Prior

History of MCMC (Contd)

- Hastings generalized Metropolis algorithm and derived the essential condition that the acceptance ratio must satisfy in order to sample target distribution (*known as Metropolis Hastings algorithm*)
- Gelfand and Smith (1990) showed that Gibbs Sampling and Metropolis-Hastings based sampling methods can used to solve a wide class of Bayesian problems
- First applied in astrophysics in 1994. then 2 papers in 1998 and one in 2000
- MCMC package CosmoMC for solving cosmological parameter estimation problems (2002) (*Lewis and Bridle*)
- Release of emcee in 2013.

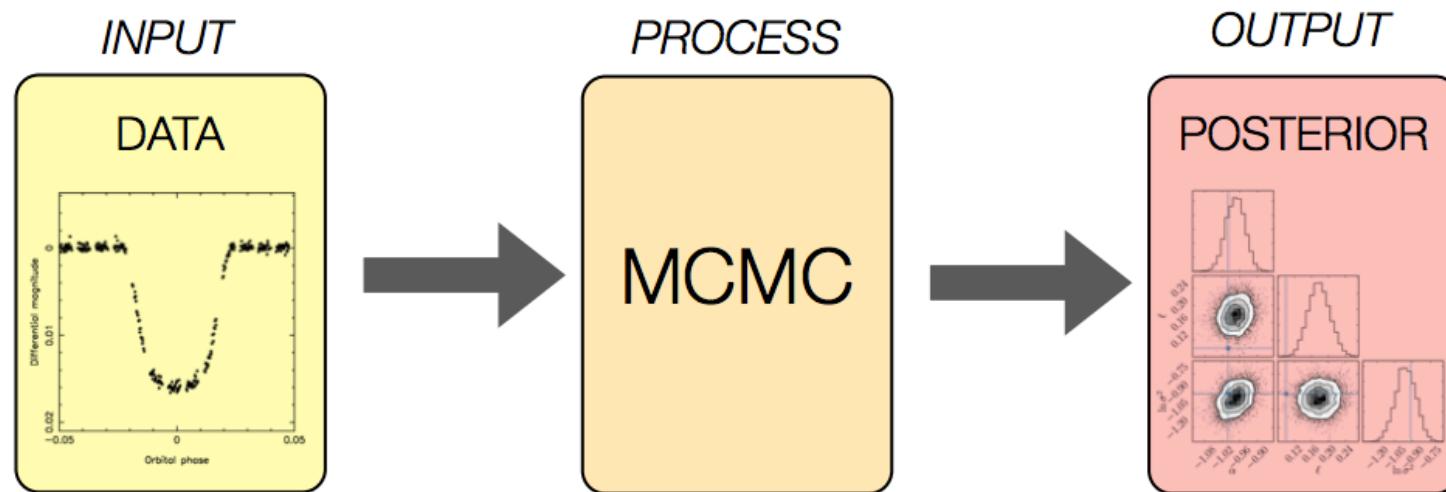
Basic idea behind MCMC

- Let's say we have a probability distribution (possibly in a very high dimension) and we want to sample from or find the expected value of it and doing so analytically is impossible.
- We can do this through a Monte Carlo method known as Markov Chain Monte Carlo:
 - The basic idea is that we perform a ``random walk'' through the probability distribution in question favoring values with higher probabilities.
 - i.e. once we have a starting point, we randomly pick a nearby point and evaluate its probability. If that probability is higher than the point we started on, we move there. Otherwise we simply stay put or move to that point with some probability.
 - The amazing thing is that if we repeat this enough under the right conditions we will hit every point in the space with a frequency proportional to its probability.

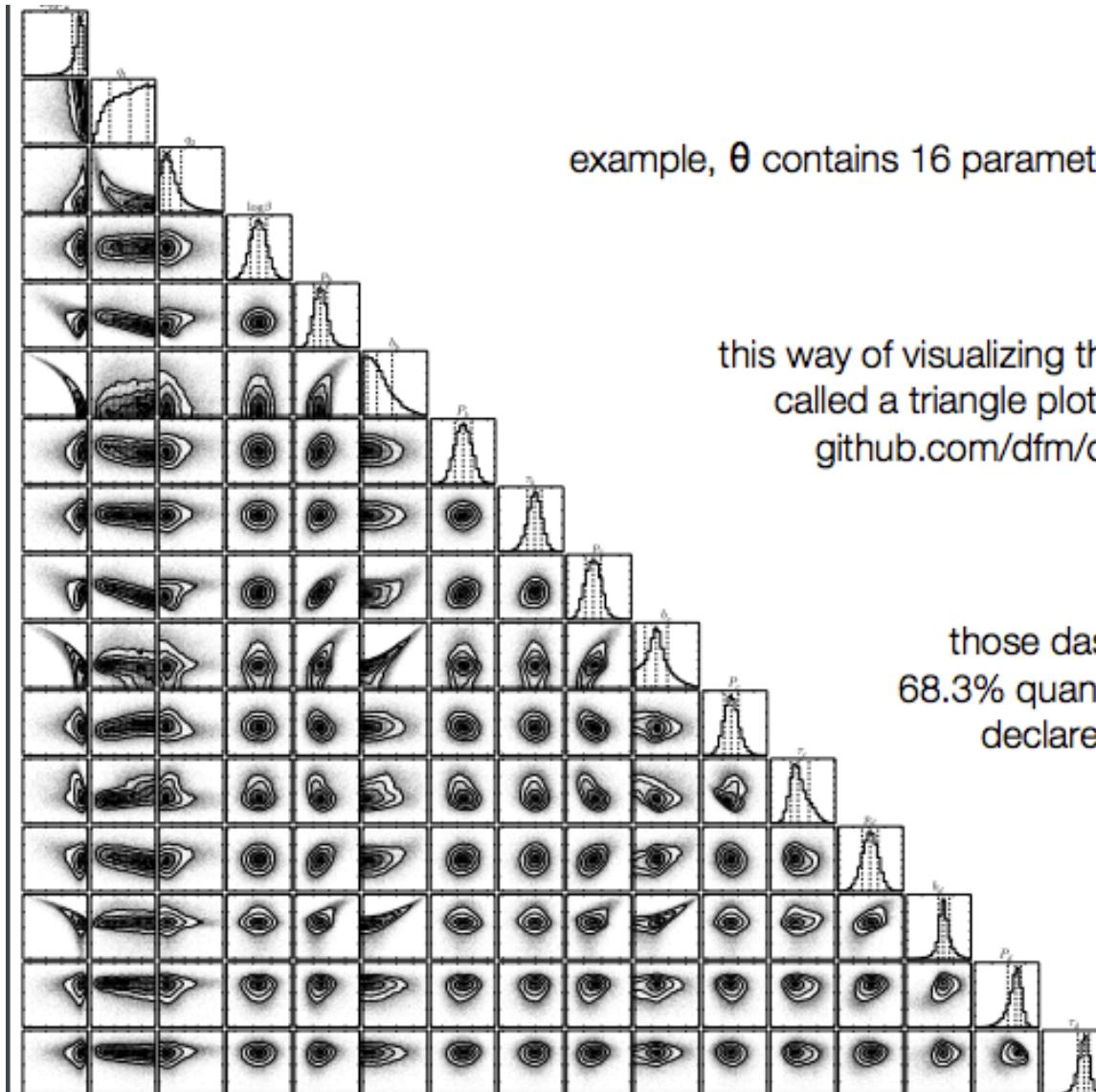
General Theory

- Purpose of a Markov Chain Monte Carlo algorithm is to construct a sequence of points ("samples") in parameter space (called "chain"). Density of samples is proportional to the posterior pdf, which allows us to construct a map of the posterior distribution.
- A Markov chain is a sequence of random variables $\{X^0, X^1, \dots, X^{M-1}\}$ such that the probability of $(t+1)$ -th element depends only on the value of the t -th element. They can be shown to converge to a stationary state , where successive elements of the chain are *samples from target distribution* (mostly the posterior).

What is the product of MCMC



and really by this I mean a set of samples from the posterior

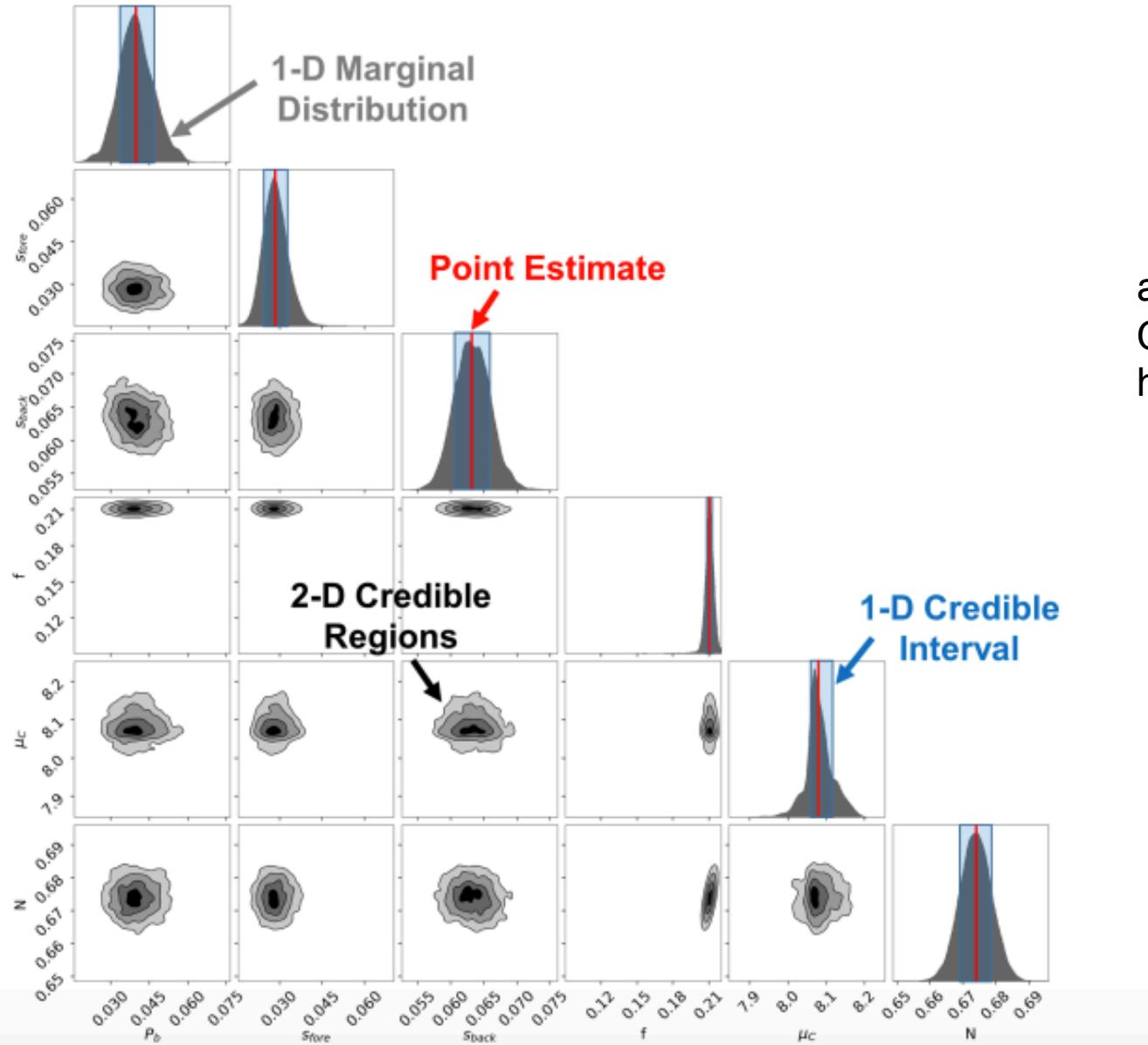


example, θ contains 16 parameters

this way of visualizing the posteriors is
called a triangle plot, check out
github.com/dfm/corner.py

those dashed lines are the
68.3% quantiles, which we often
declare as $\theta_1=2.0\pm0.1$

Kipping et al. (2016)



arXiv:1909.12313
Corner (or triangle) plot showing how posteriors are used in MCMC

- Generation of elements of the chain is *probabilistic* in nature and is described by a *transition probability* $T(\theta^{(t)}, \theta^{(t+1)})$, giving the probability of moving from point $\theta^{(t)}$ to $\theta^{(t+1)}$ in parameter space. Sufficient condition to obtain a Markov chain is that the transition probability satisfy the *detailed balanced conditions*.

$$\frac{T(\theta^{(t)}, \theta^{(t+1)})}{T(\theta^{(t+1)}, \theta^{(t)})} = \frac{p(\theta^{(t+1)}|d)}{p(\theta^{(t)}|d)}$$

Ratio of transition probabilities is **inversely proportional** to the ratio of the posterior probabilities at the two points.

Posterior mean given by

$$E[\theta] = \int P(\theta|d)\theta d\theta \approx \frac{1}{M} \sum_{t=0}^{M-1} \theta^{(t)}$$

$$E[f(\theta)] \approx \frac{1}{M} \sum_{t=0}^{M-1} f(\theta^{(t)})$$

Calculation of Marginalized Posterior

- 1-dimensional marginalized probability for the j^{th} element of θ , θ_j obtained by integrating out all other parameters from the posterior.

$$P(\theta_1|d) = \int P(\theta|d)d\theta_2 \dots d\theta_n$$

This can be easily calculated from the Markov Chain. Since elements of the Markov Chains are samples from the full posterior $P(\theta|d)$, their density reflects the value of the full posterior

Divide the range of θ_j in a series of bins and count the number of samples falling in each bin (ignoring the values for other coordinates)

Same with 2-d posterior.

Calculation of Credible Intervals

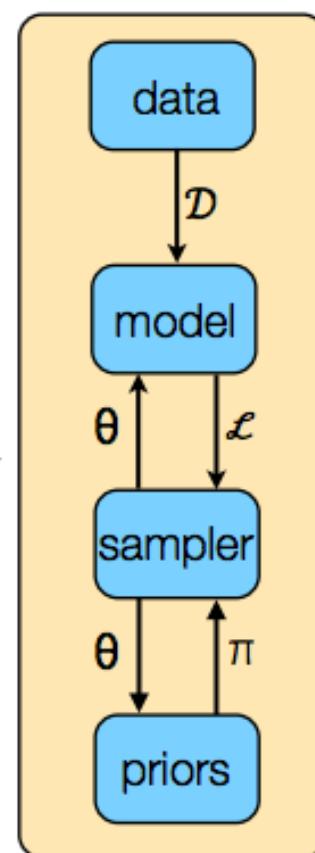
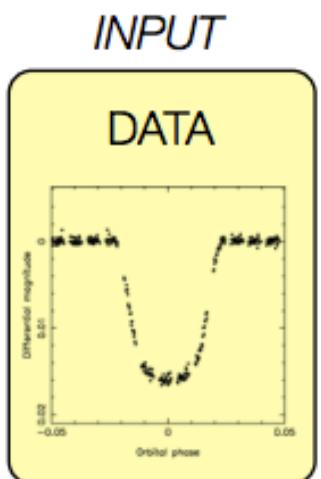
- A 1D 2-tail symmetric α % credible region is given by the interval within which $\alpha\%$ of the samples are contained, obtained in such a way that a fraction $(1-\alpha)/2$ of the samples lie outside the interval on either side.
(This is called symmetric credible interval)
- In case of an upper (lower) limit, we report the value of the quantity below (above) which $\alpha\%$ of the samples are to be found.

Role of MCMC Sampler

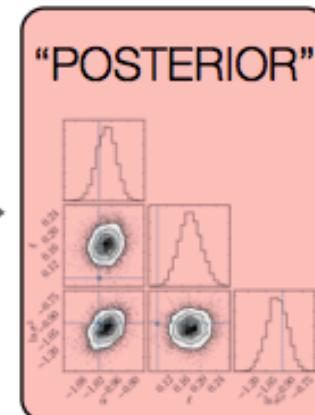
the sampler “guesses” different θ vectors, calculates the posterior probability of that guess, and then makes small jumps

PROCESS

actually the point of the sampler is to make intelligent guesses with high posterior probabilities



OUTPUT



MCMC

Metropolis-Hastings Algorithm

1. Start from a random point θ_0 having associated posterior probability $p_0 \equiv p(\theta^{(0)} | d)$
2. Propose a candidate θ^c from the proposal distribution $q(\theta_0, \theta^c)$. The proposal distribution might be for example a Gaussian of fixed σ centered around the current point. For the Metropolis algorithm, distribution q satisfies the symmetry condition $q(x,y) = q(y,x)$
3. Calculate the posterior at the candidate point $p_c = p(\theta^c | d)$. Accept the candidate point with probability

$$\alpha = \min \left(\frac{p_c q(\theta^{(c)}, \theta^{(o)})}{p_o q(\theta^{(o)}, \theta^{(c)})}, 1 \right) \Rightarrow \alpha = \min \left(\frac{p_c}{p_o}, 1 \right) \text{ for Metropolis algo.}$$

4. Generate a uniform random number u from the uniform distribution $[0,1)$ and accept the candidate sample if $u < \alpha$ and reject it otherwise.
5. If the candidate point is accepted, add it to the chain and move there. Otherwise stay at the old point (which is counted twice in the chain). Go back to 2.
 - if $P_c > P_o$, the candidate is always accepted
 - Only the un-normalized posterior is required (since normalization constant drops out of the ratio)
 - Metropolis algorithm satisfies the detailed balance condition.
 - Affine invariant ensemble samplers evolve a series of ``walkers'' to generate a move with greatly reduced correlation length. This is used in emcee
 - Other sampling methods such as Gibbs sampling used.

M-H Algorithm described in 2017 Hogg review

The M-H MCMC algorithm requires two inputs. The first is a handle to the function $f(\theta)$ that is the function to be sampled, such that the algorithm can evaluate $f(\theta)$ for any value of the parameters θ . In data-analysis contexts, this function would be the prior $p(\theta)$ times the likelihood $p(D | \theta)$ evaluated at the observed data D . The second input is a handle to a proposal pdf function $q(\theta' | \theta)$ that can deliver samples, such that the algorithm can draw a new position θ' in the parameter space given an “old” position θ . This second function must meet a symmetry requirement (detailed balance) we discuss further below. It permits us to random-walk around the parameter space in a fair way.

The algorithm¹⁴ is the following: We have generated some set of samples, the most recent of which is θ_k . To generate the next sample θ_{k+1} do the following:

- Draw a proposal θ' from the proposal pdf $q(\theta' | \theta_k)$.
- Draw a random number $0 < r < 1$ from the uniform distribution.
- If $f(\theta')/f(\theta_k) > r$ then $\theta_{k+1} \leftarrow \theta'$; otherwise $\theta_{k+1} \leftarrow \theta_k$.

That is, at each step, either a new proposed position in the parameter space gets accepted into the list of samples or else the previous sample in the parameter space *gets repeated*. The algorithm can be iterated a large number K of times to produce K samples.

Why does this algorithm work? The answer is not absolutely trivial¹⁵, but there are two components to the argument: The first is that the Markov process delivers

arXiv:1802.07683

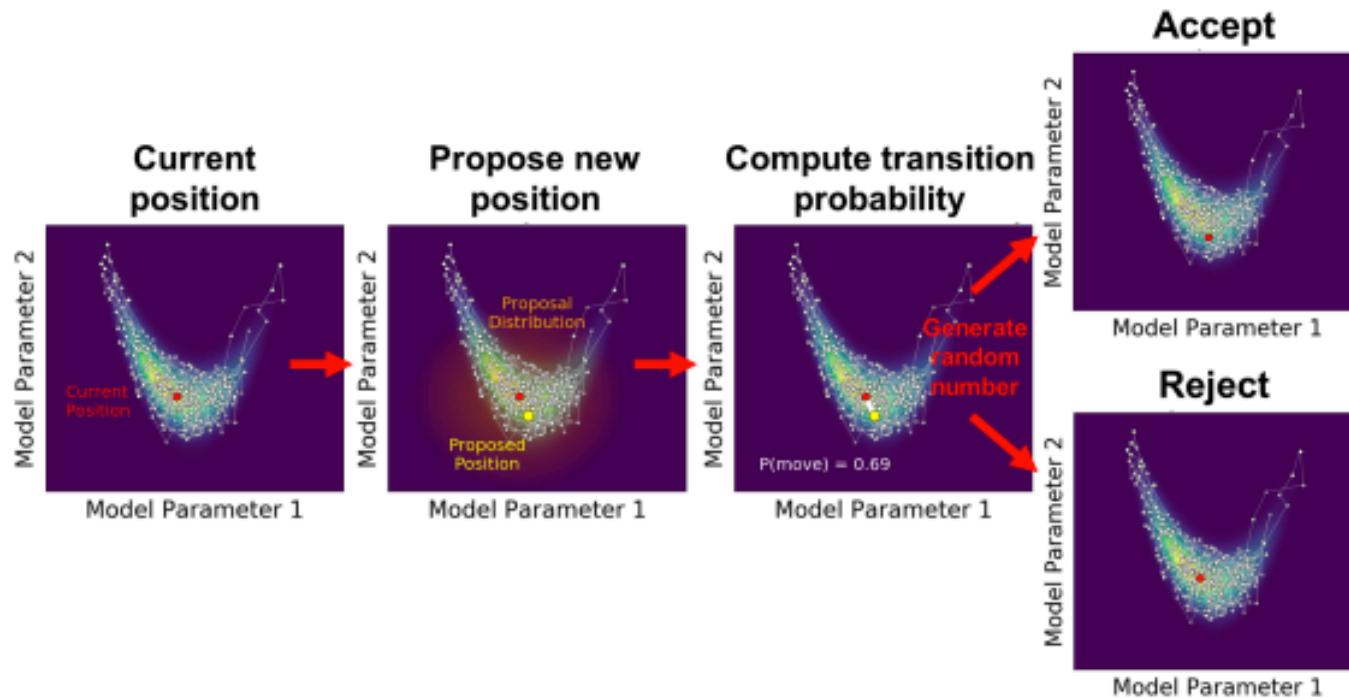
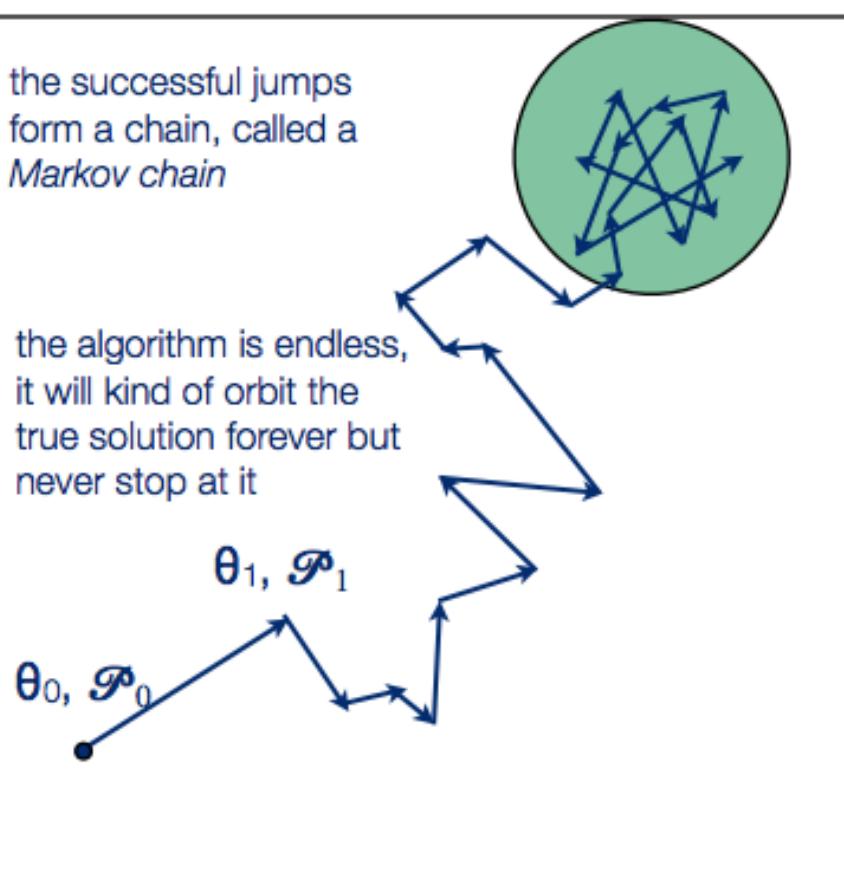


Figure 9: A schematic illustration of the Metropolis-Hastings algorithm. At a given iteration i , we have generated a chain of samples $\{\Theta_1 \rightarrow \dots \rightarrow \Theta_i\}$ (white) up to the current position Θ_i (red) whose behavior follows the underlying posterior $\mathcal{P}(\Theta)$ (viridis color map). We then propose a new position Θ'_{i+1} (yellow) from the proposal distribution (orange shaded region). We then compute the transition probability $T(\Theta'_{i+1}|\Theta_i)$ (white) based on the posterior $\mathcal{Q}(\Theta)$ and proposal $\mathcal{Q}(\Theta'|\Theta)$ densities. We then generate a random number u_{i+1} uniformly from 0 to 1. If $u_{i+1} \leq T(\Theta'_{i+1}|\Theta_i)$, we accept the move and make our next position in the chain $\Theta_{i+1} = \Theta'_{i+1}$. If we reject the move, then $\Theta_{i+1} = \Theta_i$. See §6.1 for additional details.

b



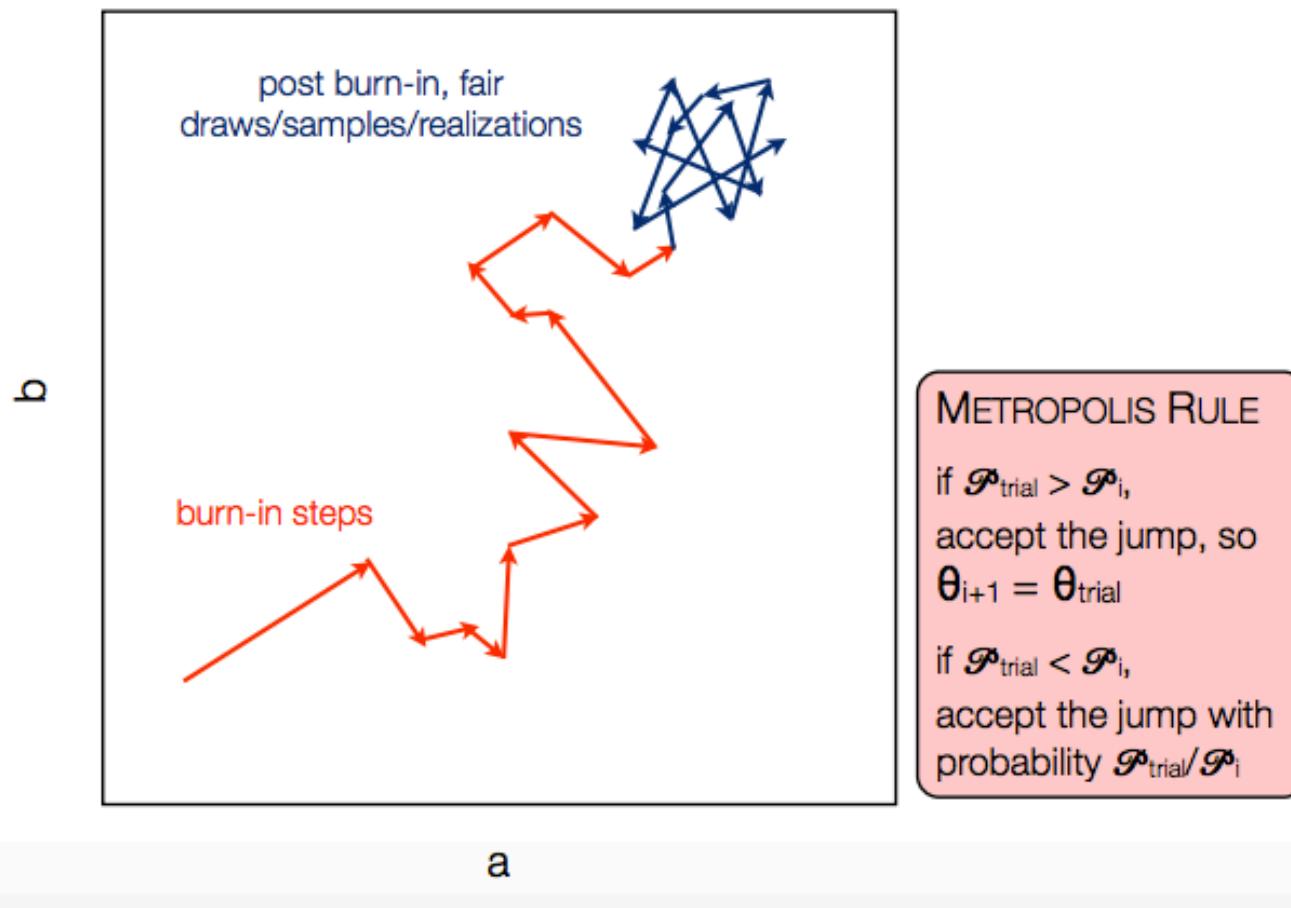
a

high \mathcal{P} region

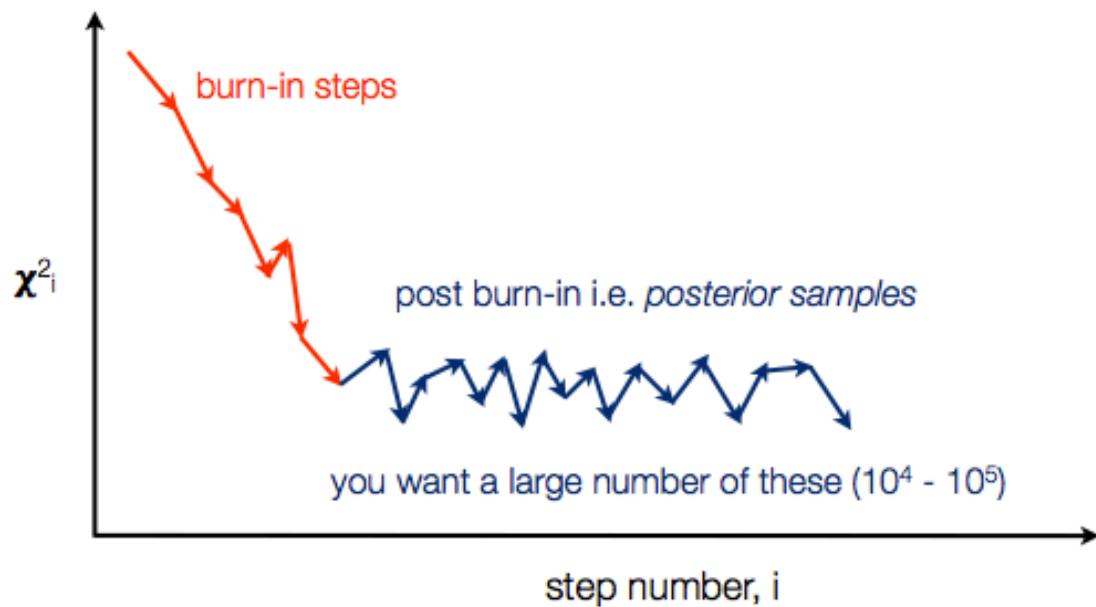
METROPOLIS RULE

if $\mathcal{P}_{\text{trial}} > \mathcal{P}_i$,
accept the jump, so
 $\theta_{i+1} = \theta_{\text{trial}}$
if $\mathcal{P}_{\text{trial}} < \mathcal{P}_i$,
accept the jump with probability $\mathcal{P}_{\text{trial}}/\mathcal{P}_i$

5. keep jumping!
6. after you've done many steps, remove burn-in steps



burn-in point can be spotted by eye...



Practical and Numerical Issues

- Initial Samples from the Chain must be discarded, since the Markov process is not yet sampling from the equilibrium distribution (so called *burn-in period*)
- Assessment of chain convergence, which aims at establishing whether the Monte Carlo estimate is sufficiently accurate. Some tools include Gelman and Rubin criterion and Raftery/Lewis statistic.
- Make sure it does not get trapped in local minima
- Successive samples in a chain are correlated. Calculate the auto-correlation, which is a good measure of the number of steps required before chain has ``forgotten'' its previous state.

Successive samples in a chain are correlated. Calculate the auto-correlation, which is a good measure of the number of steps required before chain has ``forgotten'' its previous state.

Define $\hat{\gamma} = \frac{\sum_{i=0}^{M-k} (\theta_i - \bar{\theta})(\theta_{i+k} - \bar{\theta})}{\sum_{i=0}^{M-k} (\theta_i - \bar{\theta})^2}$

where k is called lag and \bar{x} the sample mean and a plot of $\hat{\gamma}$ versus lag k Is called the auto-correlation function and the value of the lag after which it drops to 0 provides an estimate of the thinning factor K required to obtain independent samples from the chain.

- For more information on how to check convergence of MCMC, check out
<http://astrostatistics.psu.edu/RLectures/diagnosticsMCMC16.pdf>
- Subroutine for calculating Gelman-Rubin Convergence Criterion in Emcee
<http://joergdietrich.github.io/emcee-convergence.html>

Summary of all MCMC and Nested sampling routines in python3

<http://mattpitkin.github.io/samplers-demo/pages/samplers-samplers-everywhere/>

Urge all of you to bookmark this

Tools for Calculating Bayesian Evidence

- <https://dynesty.readthedocs.io/en/latest/dynamic.html>
based on Nested Sampling algorithm (check out wikipedia)
Details in <https://arxiv.org/abs/2101.09675>

<https://dynesty.readthedocs.io/en/latest/dynamic.html>
(see also arXiv:1907.07199 Krishak & SD)

Youtube lectures on nested sampling (at AstroHackWeek 2015@NYU)
<https://www.youtube.com/watch?v=5bO2JwsJY1M>

Also check out <https://arxiv.org/abs/1904.02180>

Density Estimation/ Exploratory Data Analysis

Week 7+

Introduction

- Given a sample of N points in D-dimensional space :
 - Density Estimation
 - Cluster Finding
 - Statistical Description of the observed structure
- To infer a pdf from a sample of data is called density estimation.
- Does it display structure. Finding concentrations of multi-variate points. Called ``clustering'' (could be spatial, could be in terms of properties, eg clusters in color space)
- Unsupervised clustering refers to case when no prior information on number and properties of clusters in the data.

Non-Parametric Density Estimation

- Weeks 1-3 : We considered underlying density of data using parametric models of probability density functions.
- Weeks 4-6 : Estimation of parameters from frequentist and Bayesian perspectives.

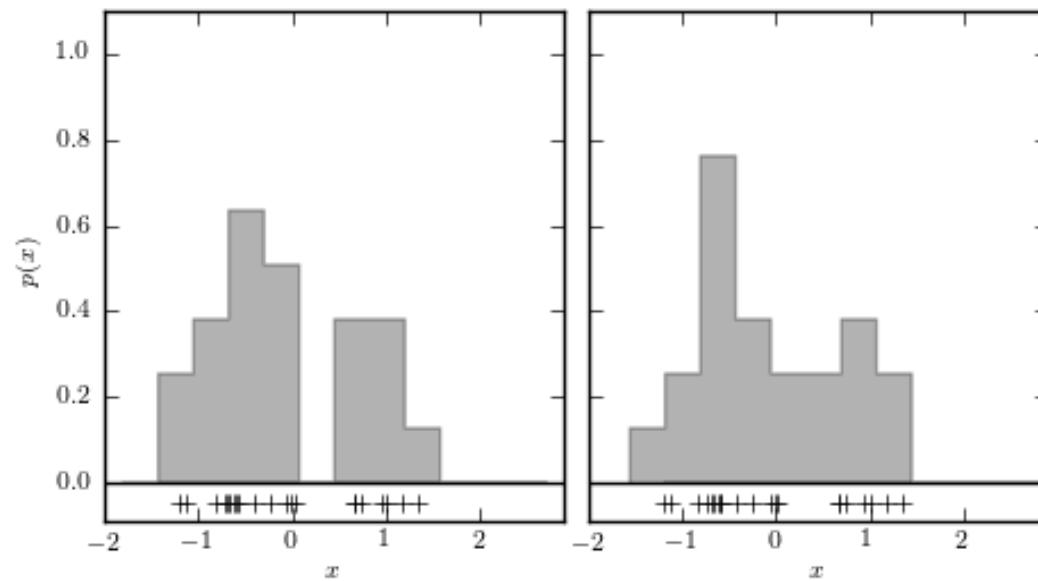
How to estimate density *non-parametrically* without specifying a specific functional form.

Non parametric methods are meant to capture every aspect of the density's shape.

Simplest method is **Kernel Density Estimation**

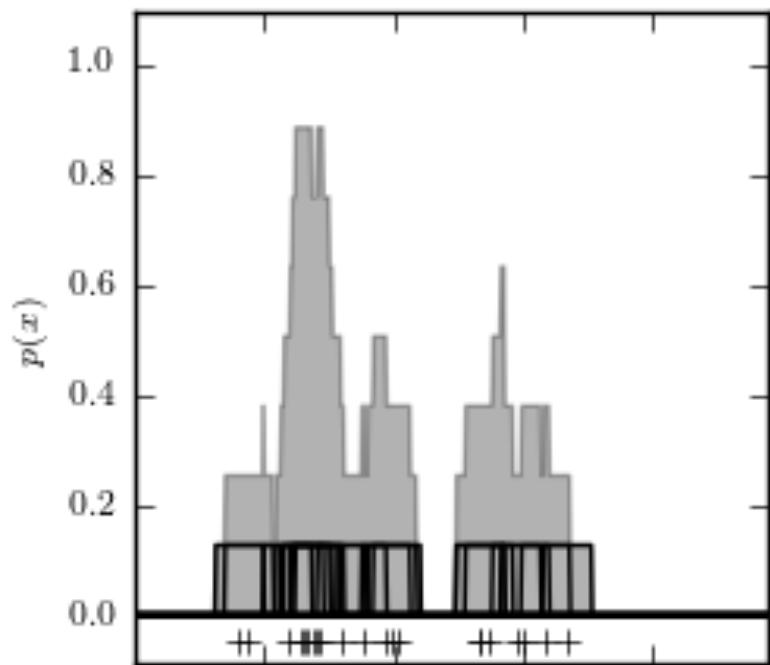
Kernel Density Estimation

Problems with histogramming data



Same data with bin centers offset by 0.25 (but same bin width)

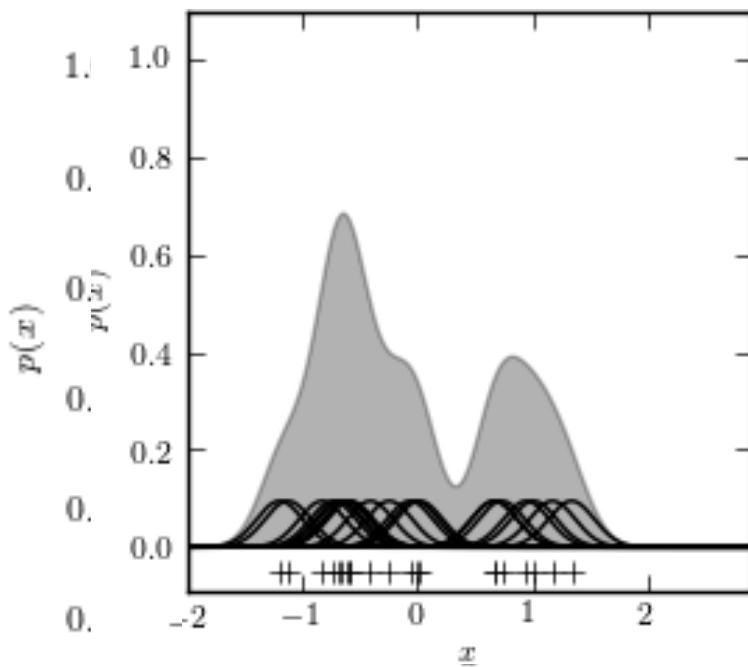
- Allow each point to have its own bin rather than arranging the bins in a regular grid and allow the bins to overlap.
→ Each point is replaced by a box of unit height and some predefined width.



Data drive the bin positioning and does a much better job of controlling the bimodal character of underlying distribution

Simplest example of Kernel Density Estimation (KDE). The Kernel is a top-hat distribution centered on each individual point. KDE is a much better estimator of density than an ordinary histogram.

However the rectangular kernel is not smooth and shows spikes. So use Gaussian Kernels.



Given a set of measurements $\{x_i\}$, the kernel density estimator at an arbitrary position x is defined as

$$\hat{f}_n(x) = \frac{1}{Nh^D} \sum_{i=1}^N K\left(\frac{d(x, x_i)}{h}\right)$$

where $K(u)$ is known as the *kernel function*

and h is known as the bandwidth (which defines the size of the kernel).

The local density is a weighted mean of all the points, where the weights are specified via $K(u)$.

Alternately KDE can be viewed as replacing each point with a cloud described by $K(u)$

Properties of Kernel Function.

- $K(u)$ is smooth and positive at all points ($K(u) \geq 0$)

- $K(u)$ is normalized to unity $\int K(u)du = 1$

- Mean of $K(u) = 0$ $\int uK(u)du = 0$

- Variance of $K(u) > 0$ $\sigma_k^2 = \int u^2K(u)du > 0$

Examples of Kernel Functions

Gaussian Kernel

$$K(u) = \frac{1}{(2\pi)^{D/2}} e^{-u^2/2}$$

where D is the number of dimensions of the parameter space and $u = d(x, x_i)/h$

Top-Hat (box) kernel

$$K(u) = \begin{cases} \frac{1}{V_D(1)} & \text{if } u \leq 1 \\ 0 & \text{if } u > 1 \end{cases}$$

Exponential Kernel

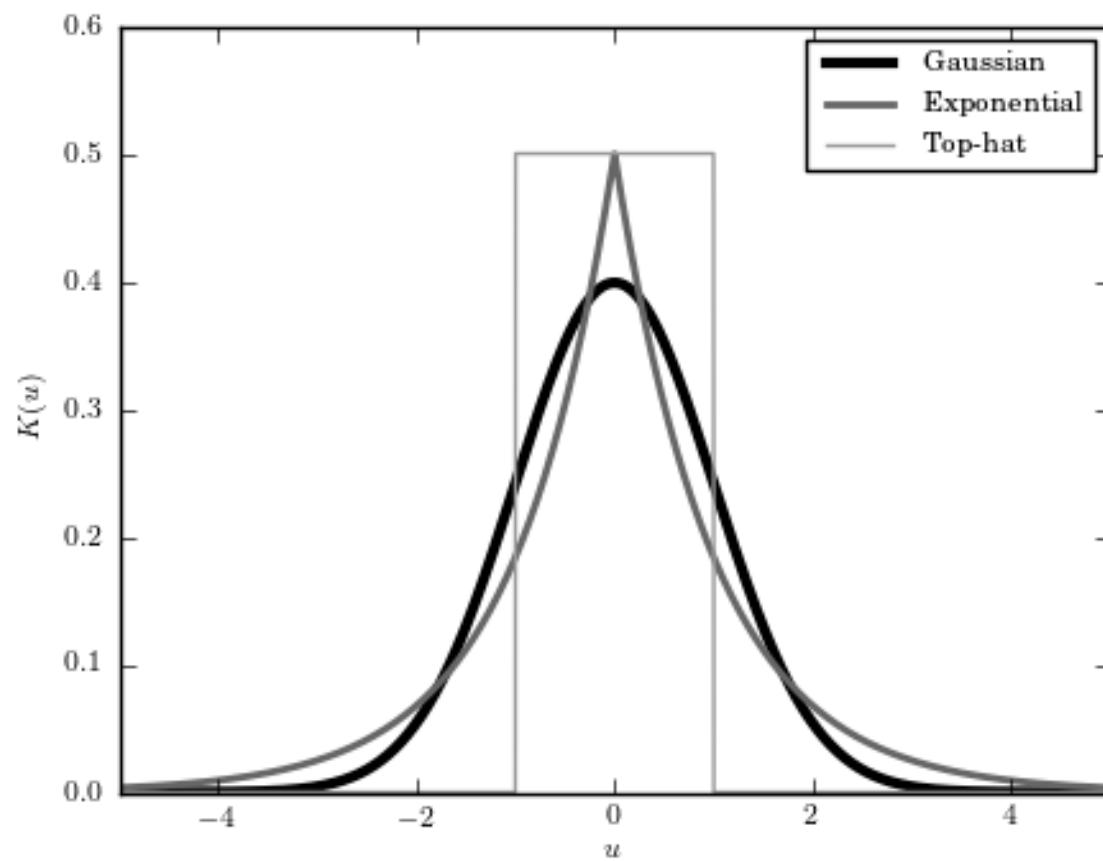
$$K(u) = \frac{1}{D!V_D(1)} e^{-|u|}$$

where $V_D(r)$ is the volume of the D-dimensional hypersphere of radius r

Optimal Kernel in terms of minimum variance is called *Epanechnikov kernel*

$K(x) = 0.75(1-x^2)$ for $|x| \leq 1$ and 0 otherwise.

Comparison of Kernels



Selecting the KDE bandwidth using cross-validation

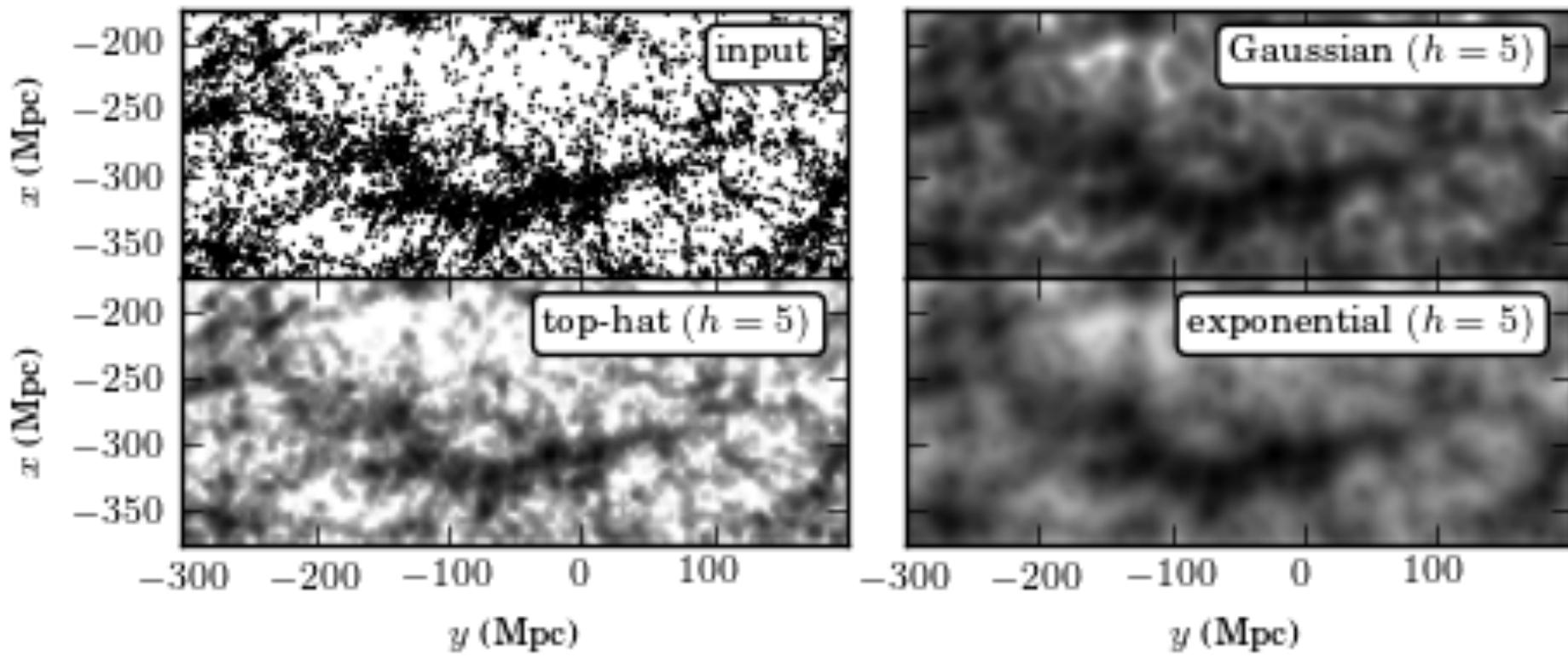
- Evaluate the cost on *out-of-sample* data. (i.e. points not in the training set)

$$CV_l(h) = \frac{1}{N} \sum_{i=1}^N \log \hat{f}_{h,-i}(x_i)$$

by maximizing $CV_l(h)$ as a function of bandwidth, we can optimize for the width of the kernel h

- Alternately, you can minimize mean-integrated square error

$$CV_{L_2}(h) = \int \hat{f}_h^2 - \frac{2}{N} \sum_{i=1}^N \hat{f}_{h,-i}(x_i)$$



Kernel density estimation for galaxies within the SDSS “Great Wall.” The top-left panel shows points that are galaxies, projected by their spatial locations (right ascension and distance determined from redshift measurement) onto the equatorial plane (declination ~ 0 degrees). The remaining panels show estimates of the density of these points using kernel density estimation with a Gaussian kernel (upper right), a top-hat kernel (lower left), and an exponential kernel (lower right). Compare also to figure 6.4.

KDE in sklearn

```
import numpy as np
from sklearn.neighbors import KernelDensity
X = np.random.normal(size=(1000,2)) # 1000 pts in 2 dim.
kde = KernelDensity('gaussian', h=0.1)
kde.fit(X)
dens = kde.eval(X)
```

kde in astroML deprecated.

Brute force algorithms to compute KDE scale as $O(N^2)$. To speed these algorithms
Tricks are used such as dual-tree fast Gauss transforms .

Nearest Neighbor density estimation

- Another density based estimation technique is based on nearest neighbors.

$$\hat{f}_k(x) = \frac{C}{\sum_{i=1}^K d_i^D}$$

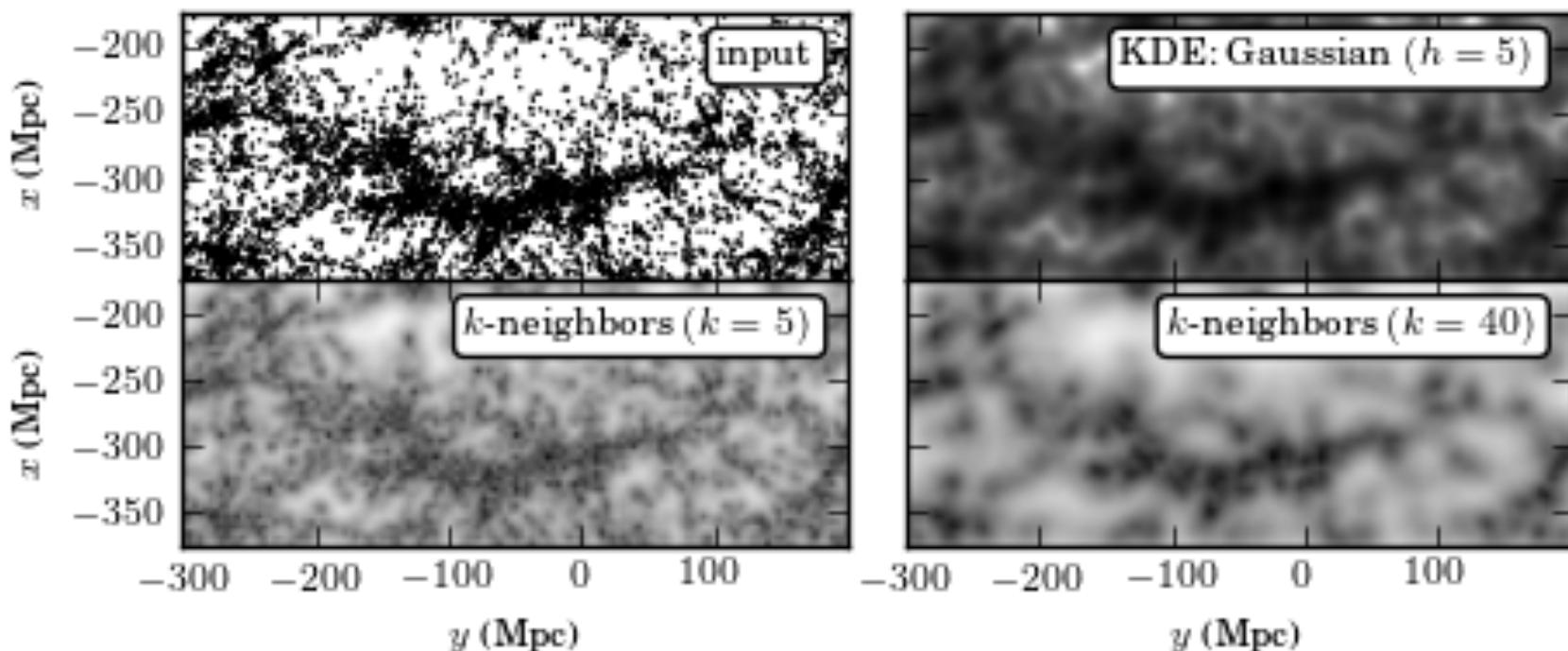
where d_i is the distance to i^{th} nearest neighbor

$$C = \frac{K(K+1)}{2V_D(1)} \quad V_D(x) = \frac{2x^D \pi^{D/2}}{D\Gamma(D/2)}$$

V_d is d-dimensional hypervolume

N.N. density estimation in Python

```
import numpy as np
from astroML.density_estimation import KNeighborsDensity
X = np.random.normal(size=(1000,2)) # 1000 points
knd = KNeighborsDensity("bayesian",10) # no of neighbours
to use
knd.fit(X) fit the model to data
dens = knd.eval(X) # evaluate the model at the data
```



Density estimation for galaxies within the SDSS “Great Wall.” The upper-left panel shows points that are galaxies, projected by their spatial locations onto the equatorial plane (declination ~ 0 degrees). The remaining panels show estimates of the density of these points using kernel density estimation (with a Gaussian kernel with width 5Mpc), a K-nearest-neighbor estimator (eq. 6.15) optimized for a small-scale structure (with $K = 5$), and a K-nearest-neighbor estimator optimized for a large-scale structure (with $K = 40$).

Unsupervised Clustering

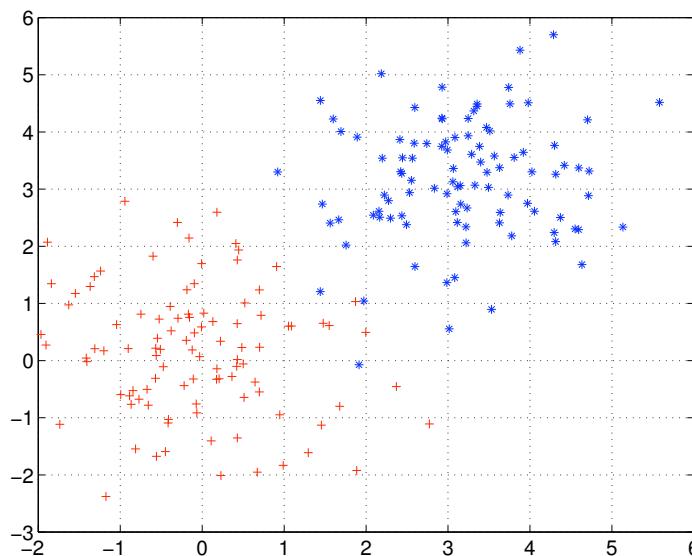
- Given multivariate point dataset, does it display any structure or concentration of points .
- Alternatively, when a density estimate is available, we can search for “overdensities”
- Seek a partitioning or segmentation of data into smaller parts according
- Unsupervised means no prior information about number and properties of clusters.

http://www.iiap.res.in/astrostat/School10/LecFiles/TKrishnan_ClusterAnal_presentn.pdf

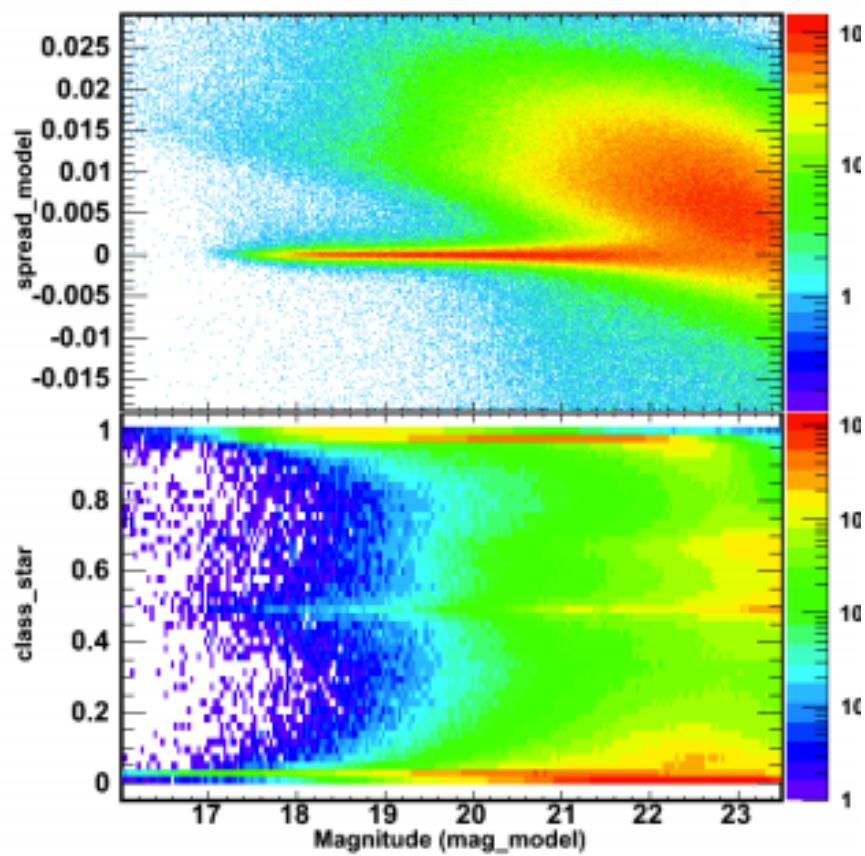
Lectures by Jia Li (PSU) and T. Krishnan (Strand Life Sciences)

Clustering

- A basic tool in data mining/pattern recognition:
 - Divide a set of data into groups.
 - Samples in one cluster are close and clusters are far apart.

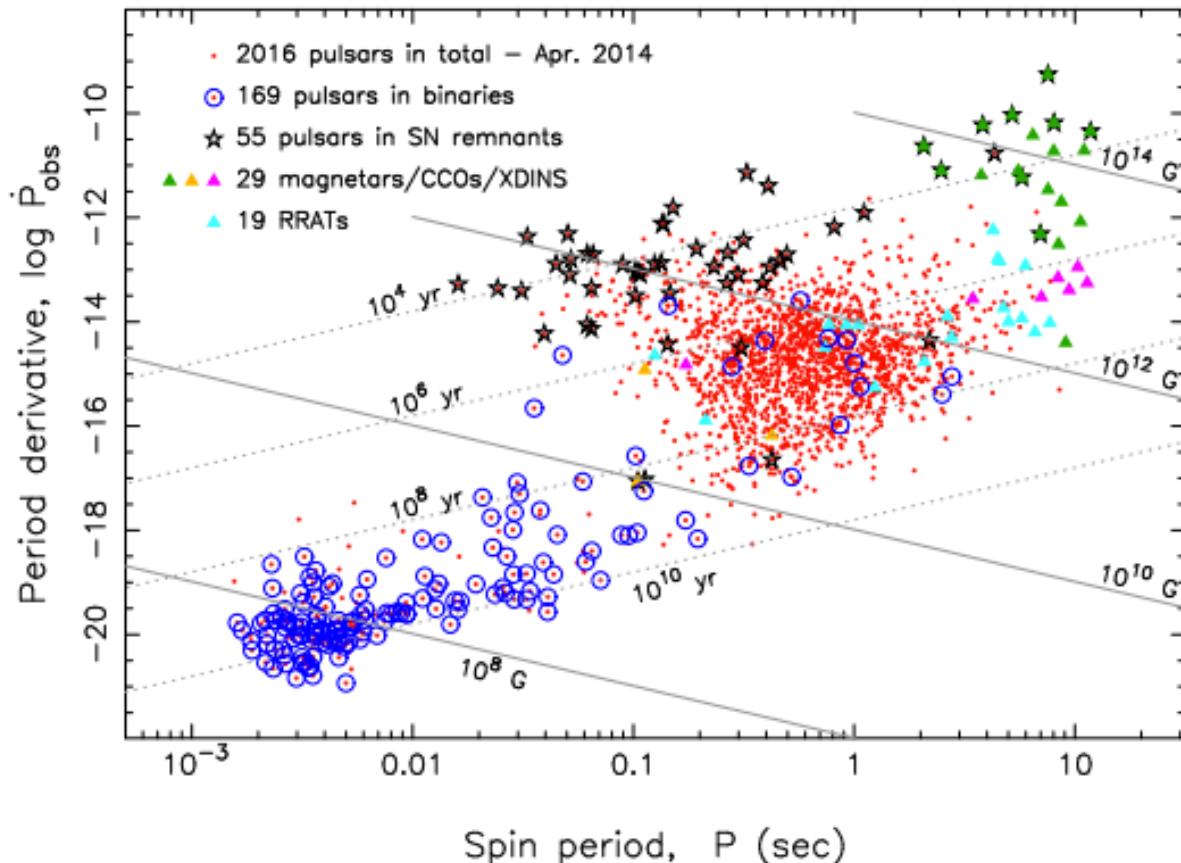


- Motivations:
 - Discover classes of data in an unsupervised way (unsupervised learning).
 - Efficient representation of data: fast retrieval, data complexity reduction.
 - Various engineering purposes: tightly linked with pattern recognition.



arXiv:1204.1210 S.Desai et al
Blanco Cosmology Survey

Pulsar P-Pdot diagram



arXiv:1602.07738

Fig. 3. $P-\dot{P}$ diagram. Red dots indicate known radio pulsars as of April 2014. Blue circles represent binaries. Stars represent associated supernova remnants. Magnetars are represented by green triangles (see §5). The XDINS and CCOs (see §4,5) are pink and yellow triangles, respectively. RRATs (§9) are in cyan. Solid grey lines are of constant magnetic field (Eq. 1) and dotted lines are of constant characteristic age (Eq. 2). From Tauris et al. (2014).

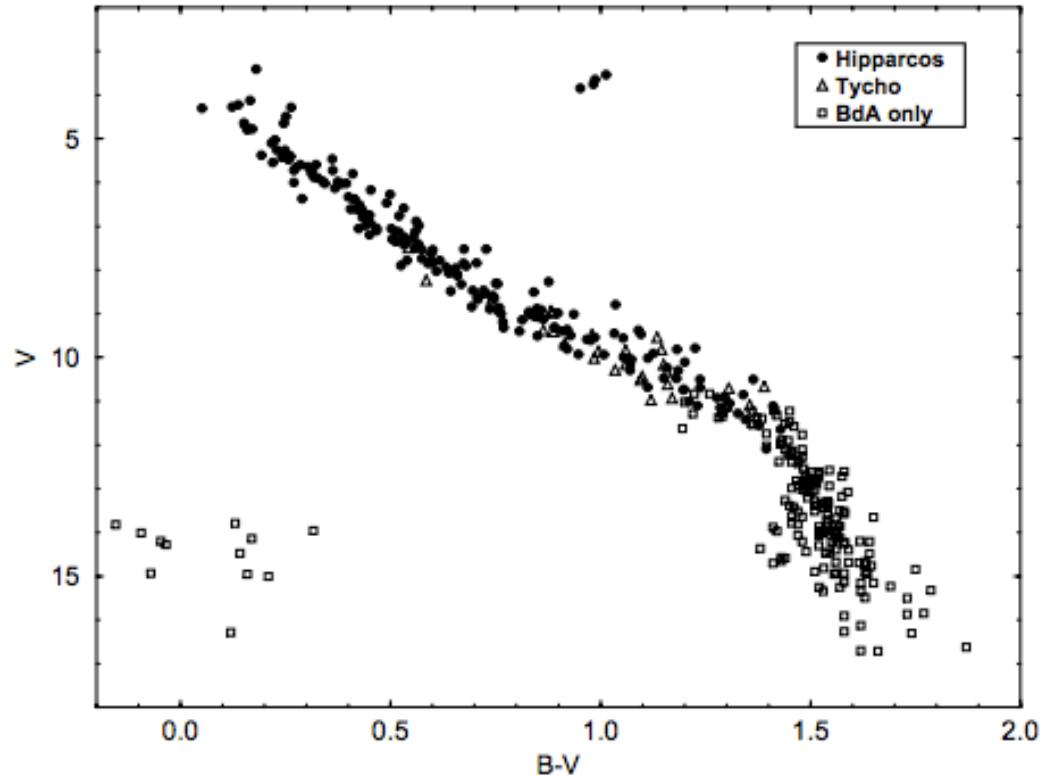


Fig. 2. Hyades stars from the ‘Base des Amas’ (BDA). Stars contained in the Hipparcos Catalogue are displayed as filled circles (190 stars). Stars not contained in the Hipparcos Catalogue, but appearing in the Tycho Catalogue, are displayed as open triangles (27 stars). The remaining 174 stars contained only in the BDA are displayed as open squares.

Approaches to Clustering

- Represent samples by feature vectors.
- Define a distance measure to assess the closeness between data.
- “Closeness” can be measured in many ways.

Define distance based on various norms.

For stars with measured parallax, the multivariate “distance” between stars is the spatial Euclidean distance. For a galaxy redshift survey, however, the multivariate “distance” depends on the Hubble constant which scales velocity to spatial distance. For many astronomical datasets, the variables have incompatible units and no prior known relationship. The result of clustering will depend on the arbitrary choice of variable scaling.

Approaches to Clustering

- Clustering: grouping of similar objects (unsupervised learning)
- Approaches

Prototype methods:

- * K-means (for vectors)
- * K-center (for vectors)
- * D2 clustering (for bags of weighted vectors)

Statistical modeling

- * Mixture modeling by the EM algorithm
- * Modal clustering
- Pairwise distance based partition:
 - * Spectral graph partitioning
 - * Dendrogram clustering (agglomerative): single linkage (friends of friends algorithm), complete linkage, etc.

K-Means Clustering

- K-means seeks a partitioning of the points into K disjoint sets C_k with each subset containing N_k points such that the following sum of square objective function is minimized.

$$\sum_{k=1}^K \sum_{i \in C_K} \|x - \mu_k\|^2 \quad \mu_k = \frac{1}{N_k} \sum_{i \in C_K} x_i$$

- Choose an initial guess for the centroid μ_k of each of the K clusters. We then assign each point to the cluster that it is closest to according to

$$C(x_i) = \arg \min_k \|x_i - \mu_k\|$$

- Update the centroid of each cluster by recomputing μ_k according to the new assignments. Continue until no new assignments.

- Run the above procedure multiple times with different starting values for the centroid and the result with lowest sum of squares error is used

```
import numpy as np
from sklearn.cluster import kMeans
X = np.random.normal(size=(1000,2))
clf = kMeans(n_clusters=3)
clf.fit(X)
centers = clf.clusters_centers_ location of the clusters
labels = clf.predict(X) labels for each of the points
```

Alternate objective function for K-Means Clustering

- Instead of minimizing sum of square errors, one can minimize the maximum radius of a cluster

$$\min_k \max_{x_i \in C_k} \|x_i - \mu_k\|$$

How to choose cluster-centers

Gonzalez algorithm:

Starting with no clusters we progressively add one cluster at a time (by arbitrarily selecting a point within the dataset to be the center of the cluster)

We then find the point x_i which maximizes the distance from centers of existing clusters and set that as the next cluster center. The procedure is repeated until we achieve K clusters. At this stage each point in the dataset is assigned the label of its nearest cluster center.

<https://scicomp.stackexchange.com/questions/19016/gonzalez-algorithm>

Example

- Training set: $\{1.2, 5.6, 3.7, 0.6, 0.1, 2.6\}$.
- Apply k-means algorithm with 2 centroids, $\{z_1, z_2\}$.
- Initialization: randomly pick $z_1 = 2, z_2 = 5$.

fixed	update
2	$\{1.2, 0.6, 0.1, 2.6\}$
5	$\{5.6, 3.7\}$
$\{1.2, 0.6, 0.1, 2.6\}$	1.125
$\{5.6, 3.7\}$	4.65
1.125	$\{1.2, 0.6, 0.1, 2.6\}$
4.65	$\{5.6, 3.7\}$

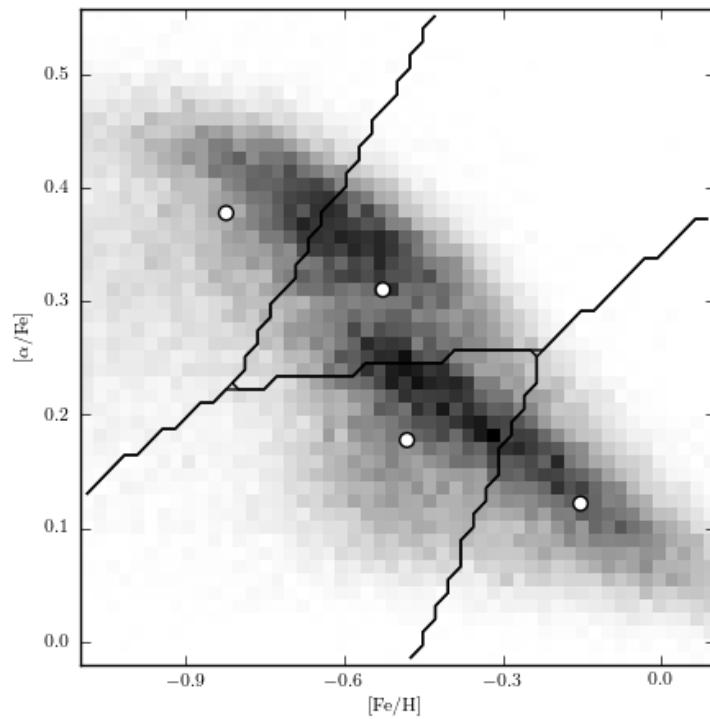
The two prototypes are: $z_1 = 1.125, z_2 = 4.65$. The objective function is $L(\mathcal{Z}, A) = 5.3125$.

- Initialization: randomly pick $z_1 = 0.8$, $z_2 = 3.8$.

fixed	update
0.8	$\{1.2, 0.6, 0.1\}$
3.8	$\{5.6, 3.7, 2.6\}$
$\{1.2, 0.6, 0.1\}$	0.633
$\{5.6, 3.7, 2.6\}$	3.967
0.633	$\{1.2, 0.6, 0.1\}$
3.967	$\{5.6, 3.7, 2.6\}$

The two prototypes are: $z_1 = 0.633$, $z_2 = 3.967$. The objective function is $L(\mathcal{Z}, A) = 5.2133$.

- Starting from different initial values, the k-means algorithm converges to different local optimum.
- It can be shown that $\{z_1 = 0.633, z_2 = 3.967\}$ is the global optimal solution.



The K-means analysis of the stellar metallicity data used in figure 6.6. Note how the background distribution “pulls” the cluster centers away from the locus where one would place them by eye. This is why more sophisticated models like GMM are often better in practice.

Clustering using Non-Parametric density Estimation

Another way to find arbitrary shaped clusters is to define the clusters in terms of modes or peaks of the non-parametric density estimate, associating each data point with its closest peak.

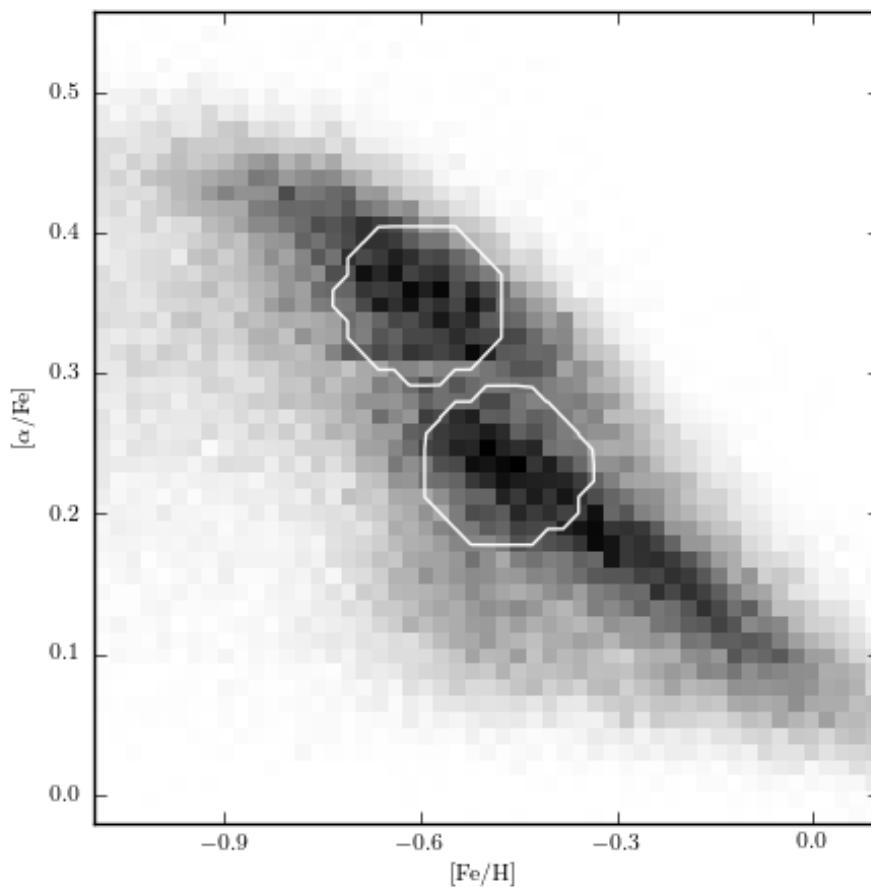
This so-called *mean-shift* algorithm is a technique to find local modes (bumps) in the density estimation of the data.

The concept behind mean shift is that we move the data points in the direction of the log of the gradient of the density of data, until they finally converge to each other at the peaks of the bumps. Number of modes is found implicitly by this method.

Mean-Shift algorithm in Python

```
import numpy
from sklearn.cluster import MeanShift
X= np.random.normal(size=(1000,2))
ms = MeanShift(bandwidth=1.0) # if no bandwidth is specified, it
will be learned from the data
```

For more details look at documentation in sklearn



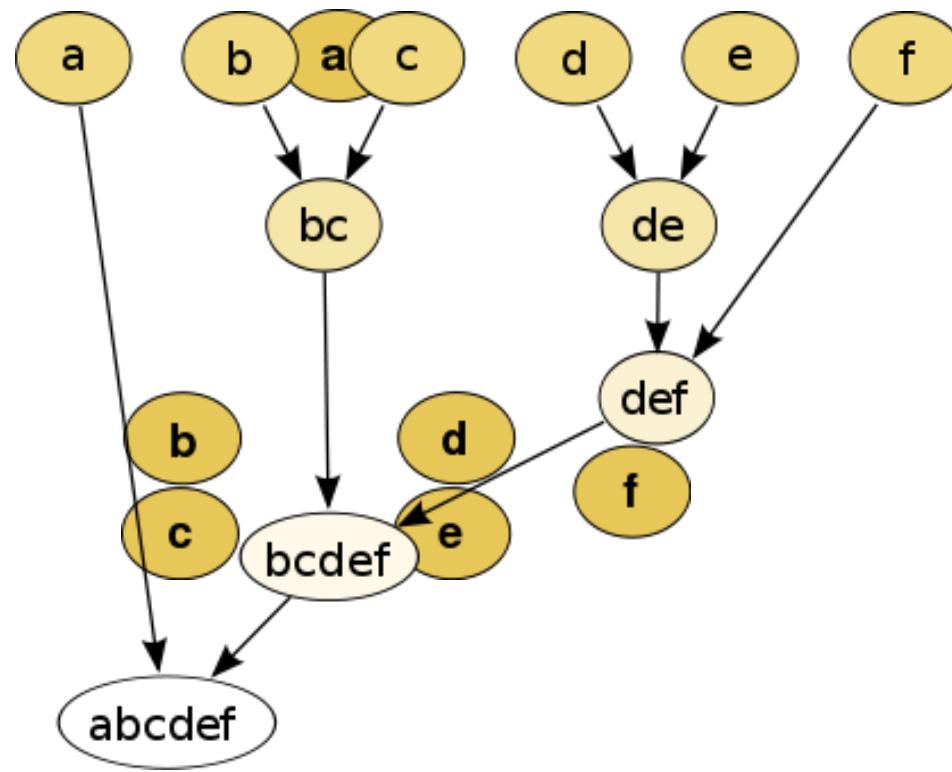
Mean-shift clustering on the metallicity data used in figures 6.6 and 6.13. The method finds two clusters associated with local maxima of the distribution (interior of the circles). Points outside the circles have been determined to lie in the background. The mean shift does not attempt to model correlation in the clusters: that is, the resulting clusters are axis aligned.

Hierarchical Clustering

- Hierarchical clustering relaxes the need to specify the number of clusters K by finding the clusters at all scales. Two approaches:
 - Top- Down (*Divisive*) Procedure
 - Bottom-up (*Agglomerative*) Procedure

Agglomerative Clustering

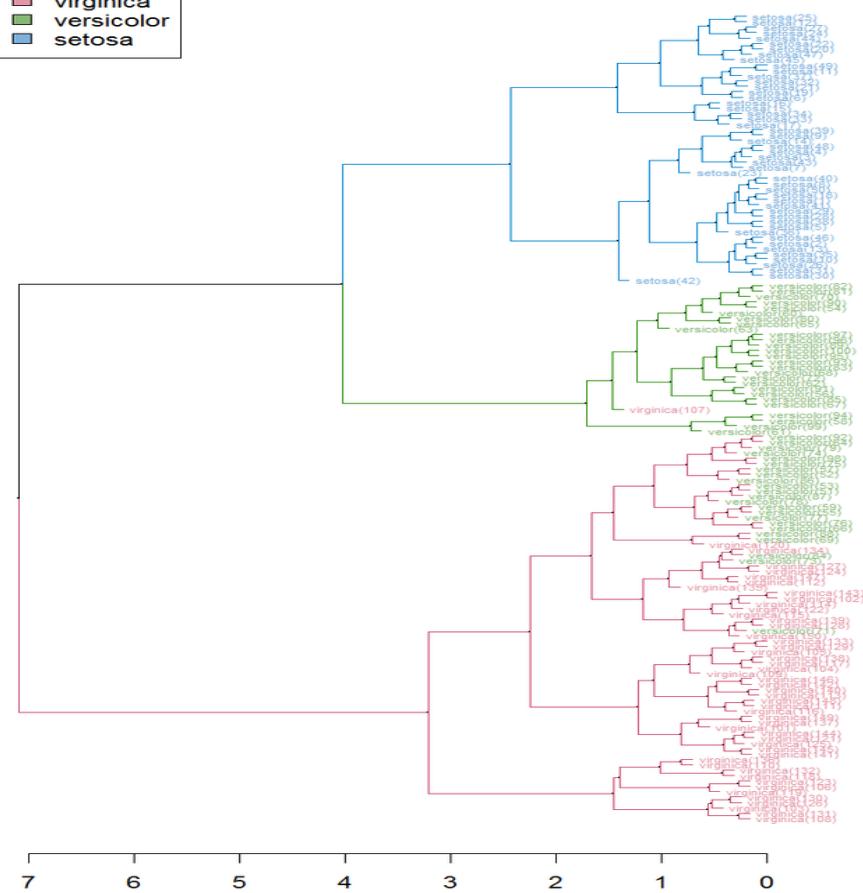
- Generate clusters in a hierarchical way.
- Let the data set be $A = \{x_1, \dots, x_n\}$.
- Start with n clusters, each containing one data point.
- Merge the two clusters with minimum pairwise distance.
- Update between-cluster distance.
- Iterate the merging procedure.
- The clustering procedure can be visualized by a tree structure called *dendrogram*.
- Definition for between-cluster distance?
 - For clusters containing only one data point, the between-cluster distance is the between-object distance.
 - For clusters containing multiple data points, the between-cluster distance is an agglomerative version of the between-object distances.
 - Examples: minimum or maximum between-objects distances for objects in the two clusters.
 - The agglomerative between-cluster distance can often be computed recursively.

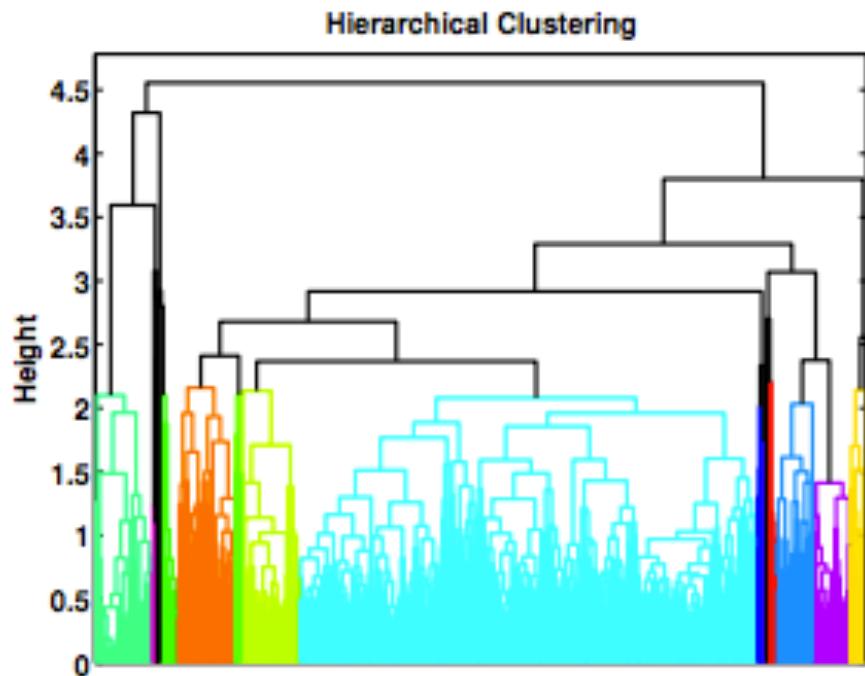


Source : wikipedia

**Clustered Iris data set
(the labels give the true flower species)**

■ virginica
■ versicolor
■ setosa





Heights usually indicate the order in which Clusters are joined or distance between the clusters

FIG. 2. Dendrogram showing hierarchical clustering of 1000 Omicron triggers from O1 Data provided by Livingston observatory. The transient morphology changes progressively from left to right

Histograms of triggers from LIGO O1 data

arXiv:1609.07259 (by N. Mukund et al)

At each step of the clustering process, nearest set of clusters are merged .
Multiple options for defining the distance between the two clusters (C_k) and ($C_{k'}$) is used

$$d_{min}(C_k, C_{k'}) = \min_{x \in C_k, x' \in C'_{k'}} ||x - x'|| \text{ (Minimum spanning tree)}$$

$$d_{avg}(C_k, C'_{k'}) = \frac{1}{N_k N'_{k'}} \sum_{x \in C_k} \sum_{x' \in C'_{k'}} ||x - x'||$$

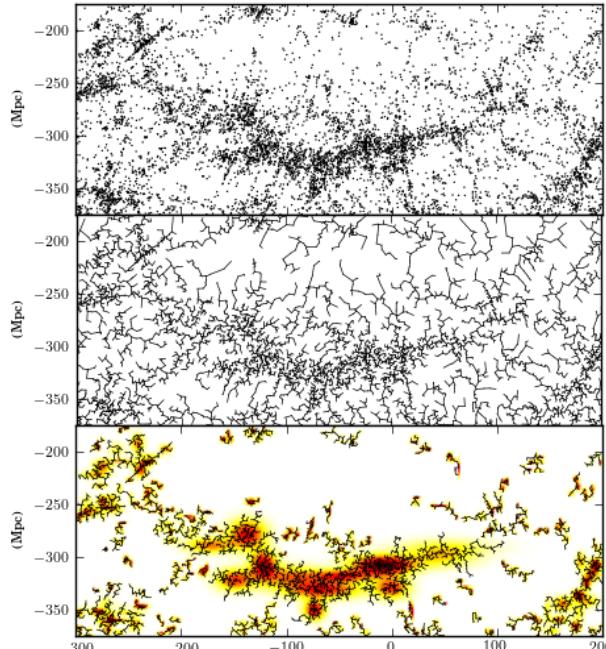
$$d_{cen}(C_k, C'_{k'}) = ||\mu_k - \mu'_{k'}||$$

More details about agglomerative clustering in scikit-learn in
[`sklearn.cluster.agglomerativeclustering`](#)

Using the distance d_{\min} results in a hierarchical clustering known as **minimum spanning tree** will produce clusters with extended chains of points.

Single-linkage hierarchical clustering (also called friends of friends algorithm in Astronomy) refers to the distance between two clusters is determined by a single element pair, namely those two elements (one in each cluster) that are closest to each other (d_{\min})

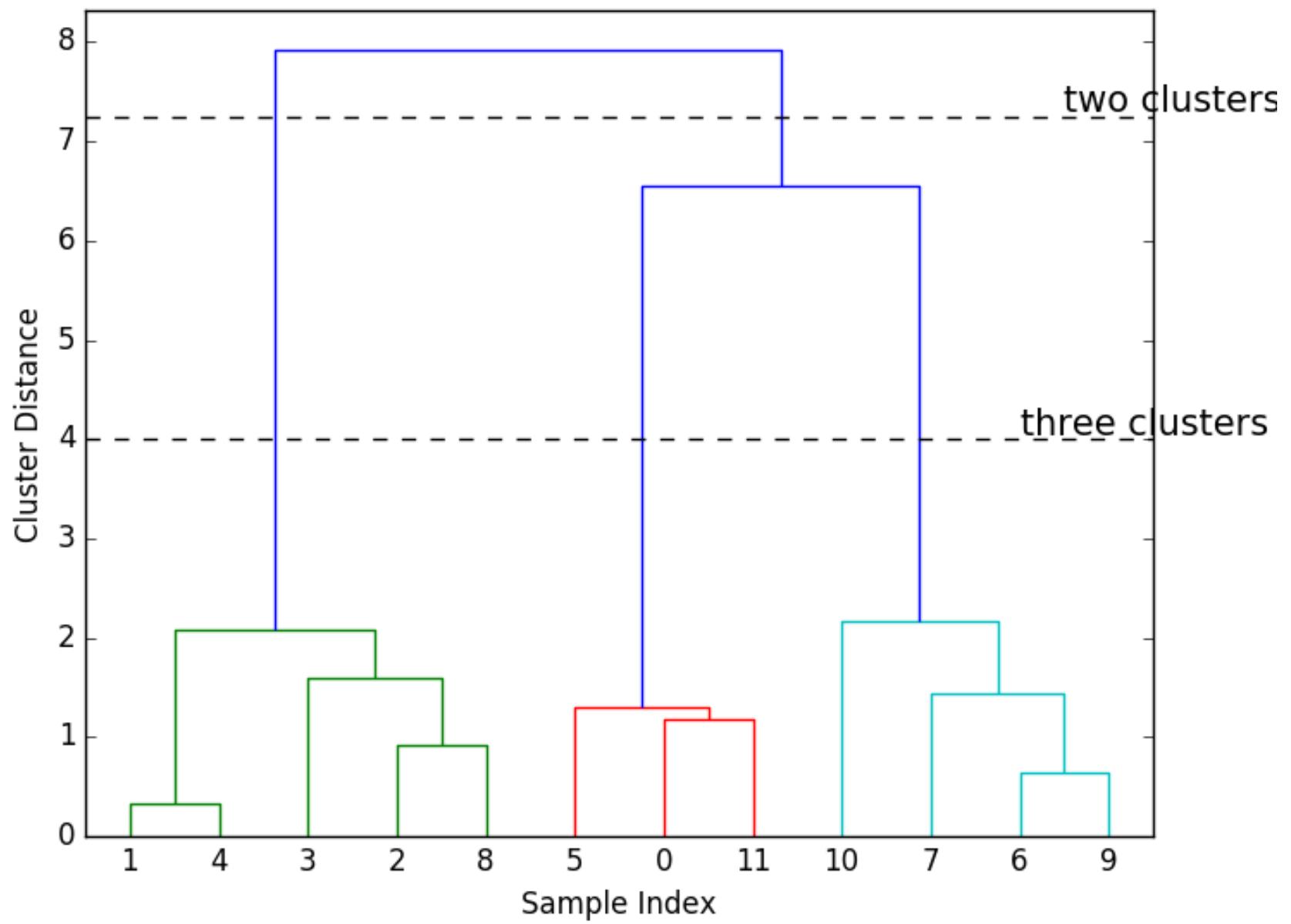
2-D dendrogram



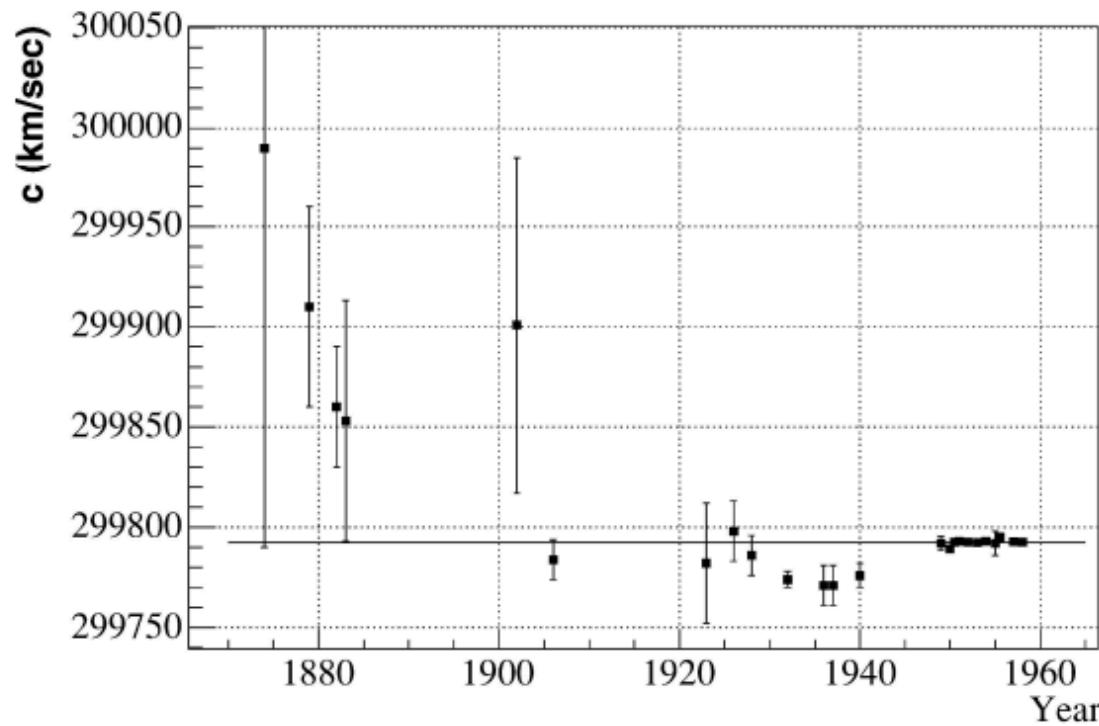
The three panels of this figure show a hierarchical clustering of a subset of galaxies from the Sloan Digital Sky Survey (SDSS). This region is known as the "SDSS Great Wall", and contains an extended cluster of several thousand galaxies approximately 300Mpc (about 1 billion light years) from earth. The top panel shows the positions of over 8,000 galaxies projected to a 2D plane with Earth at the point (0, 0). The middle panel shows a dendrogram representation of a Euclidean Minimum Spanning Tree (MST) over the galaxy locations. By eliminating edges of a MST which are greater than a given length, we can measure the amount of clustering at that scale: this is one version of a class of models known as Hierarchical Clustering. The bottom panel shows the results of this clustering approach for an edge cutoff of 3.5Mpc, along with a Gaussian Mixture Model fit to the distribution within each cluster.

```
from sklearn import datasets
from scipy.cluster.hierarchy import dendrogram,ward
from matplotlib.pylab import plt
X,y = datasets.make_blobs(random_state=0,n_samples=12) #Generate
isotropic Gaussian blobs for clustering
# apply ward clustering to the data Array X
# scipy ward function returns an array that specifies distances
bridge when performing
# agglomerative clustering
linkage_array=ward(X)
# plot the dendrogram for the linkage array containing the
distances between clusters
plt.figure()
dendrogram(linkage_array)
ax=plt.gca()
bounds=ax.get_xbound()
ax.plot(bounds,[7.25, 7.25], '--',c='k')
ax.plot(bounds,[4, 4], '--',c='k')
ax.text(bounds[1]-10,7.4, 'two clusters',
va='center',fontdict={'size':15})
ax.text(bounds[1]-15,4.15, 'three clusters',
va='center',fontdict={'size':15})
plt.xlabel('Sample Index')
plt.ylabel('Cluster Distance')
```

Dendograms in Python



Motivation for Blind Analysis



Klein JR, Roodman A. 2005.
Annu. Rev. Nucl. Part. Sci. 55:141–63

Surely you are joking Mr. Feynman (1985)

In his speech *Cargo Cult Science* (58), Richard Feynman warns that

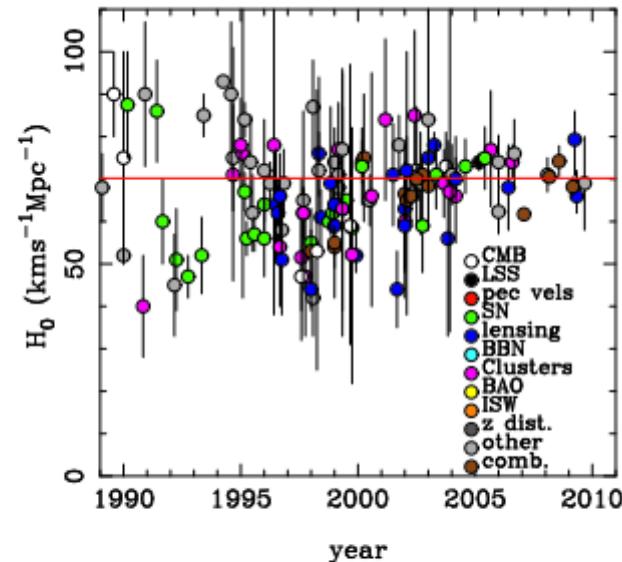
It's a thing that scientists are ashamed of—this history—because it's apparent that people did things like this: When they got a number that was too high above Millikan's, they thought something must be wrong—and they would look for and find a reason why something might be wrong. When they got a number closer to Millikan's value they didn't look so hard...

The first principle is that you must not fool yourself—and you are the easiest person to fool.

Blind Analysis done in LIGO (unlike in astronomy) usually done
In high energy physics experiments.

Look at the observed signal events ONLY after the
background has been tuned and fixed.

Confirmation Bias and Blind Analysis



arXiv:1112.3108

Figure 4. Individual published values of the Hubble constant, H_0 as a function of year. We show one sigma error bars on the points, and the point colour (shown in the legend) denotes the technique used to make the measurement (see Section 2.2 for more details).

Check: J. Klein and A. Roodman “Blind Analysis in Nuclear and Particle Physics”
Annual Review of Nuclear and Particle Science, vol. 55, Issue 1, pp.141-163 (2005)

Correlation Functions

Week 8+

Introduction

- An extension to the idea of density estimation and cluster identification is to characterize how far (and on what scales) the distribution of points differs from a random distribution.

For this purpose correlation functions and auto-correlation functions have been introduced.

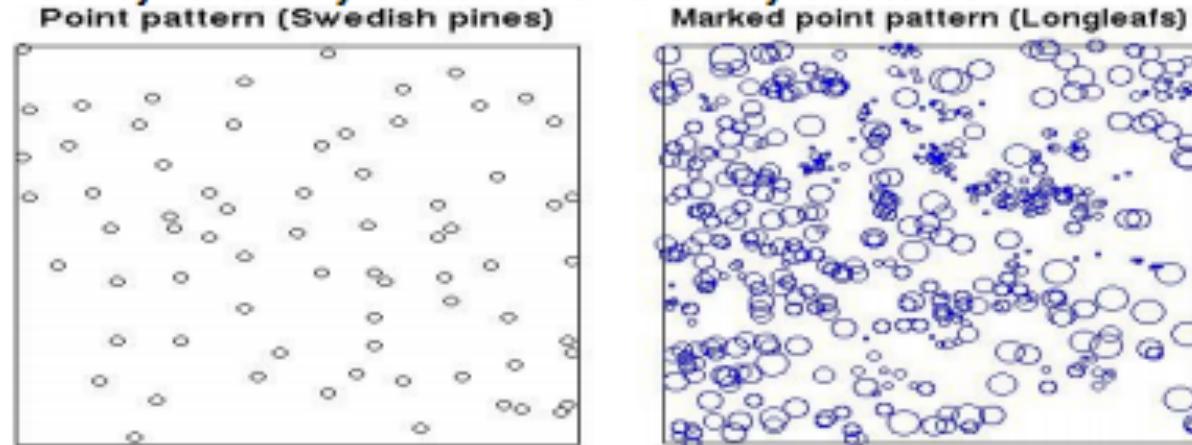
Main goal in astrophysics is to characterize the fluctuations in the densities of galaxies and quasars as a function of luminosity, galaxy type and age of universe. Test it with cosmological models

Spatial Point Process Data examples

Locations of pine saplings in a Swedish forest.

Location, diameter of longleaf pines (*marked point process*).

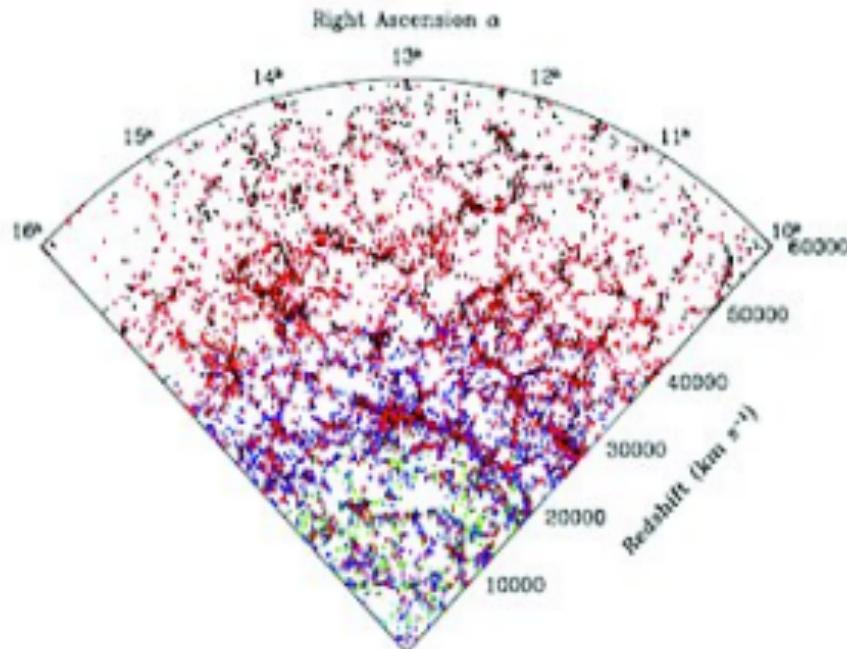
Are they randomly scattered or are they clustered?



(from Baddeley and Turner R package, 2006)

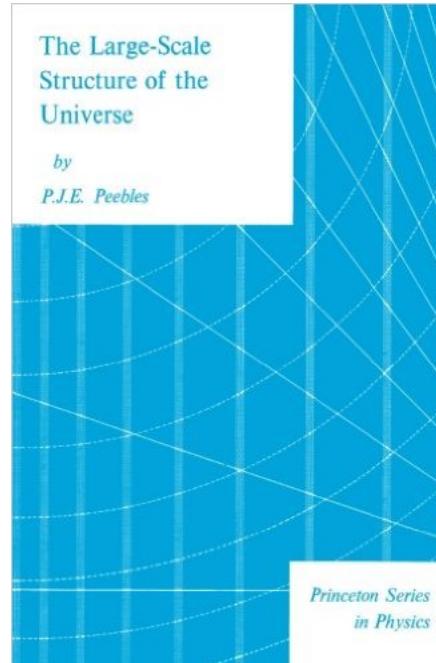
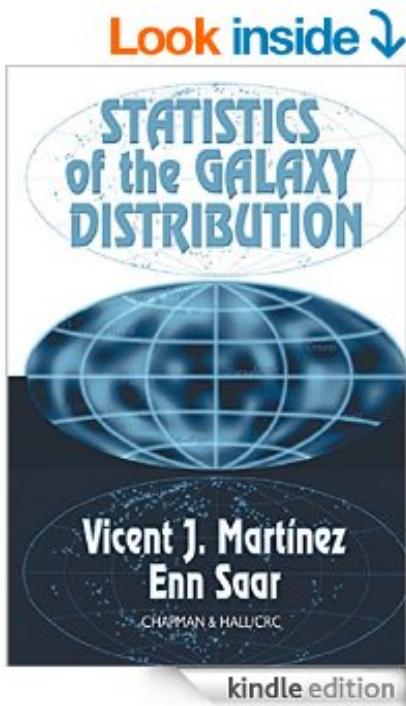
The galaxy distribution: 3D spatial point process

2d location in sky, 1d from redshift as a surrogate for distance.



(Tegmark et al. 2004, The three-dimensional power spectrum of galaxies from the Sloan Digital Sky Survey,
Astrophys. J. 606, 702-740)

References



- astro-ph/0503603 Power Spectrum Estimation I : The Basics
astro-ph/0406086 Scaling Laws in Distribution of Galaxies (*)
arXiv:0712.3928 by L. Verde
Dragan Huterer lecture notes at [Michigan LSS](#) school (2020)

Definition of Two-Point Correlation Function

- Excess probability (compared to a Poisson distribution) of finding a pair of points in two volume elements dV_1 and dV_2 separated by a distance r is given by :

$$dP_{12} = \rho^2 dV_1 dV_2 (1 + \xi(r_{12}))$$

$\xi(r)$ is called *two-point correlation function*

It describes the excess probability of finding a pair of points, as a function of separation compared to a random distribution.

Positive, negative, or zero amplitudes in $\xi(r)$ correspond to distributions which are correlated, anti-correlated or random.

For a homogenous and isotropic universe $\xi(r)$ depends only on $|r|$ and is equal to $\xi(|r|)$.

Conditional probability that a galaxy within volume element dV at a distance r given that

there is a galaxy at the origin of r is given by:

$$dP = n[1 + \xi(r)]dV$$

It measures the clustering in excess or in defect compared with a random Poisson point distribution for which $\xi(r) = 0$

In statistical mechanics, correlation function used is $g(r) = 1 + \xi(r)$ (called **radial Distribution function**) and also called **pair correlation function** in statistics

Integral of the Two-point correlation function is called **Ripley K-Function**

Relation to Power Spectrum

- Two-point correlation function is the Fourier Transform of the power spectrum.

$$P(k) = \frac{4\pi}{k} \int_0^\infty \sin(kr) \xi(r) r dr$$

$$\xi(r) = \frac{1}{2\pi^2} \int dk \ k^2 P(k) \frac{\sin(kr)}{kr}$$

where k is the wavenumber and is related to scale of wavelength of fluctuation given by $\lambda = 2\pi/k$

Correlation function can be used to define density fluctuation of sources given by:

$$\xi(r) = \left\langle \frac{\delta\rho(x)}{\bar{\rho}} \frac{\delta\rho(x+r)}{\bar{\rho}} \right\rangle$$

where

$$\frac{\delta\rho(x)}{\bar{\rho}} = \frac{\rho - \bar{\rho}}{\bar{\rho}}$$

is the density contrast , relative to the mean value $\bar{\rho}$ at position x

- In studies of galaxy distributions $\xi(r)$ is parameterized in terms of a power law

$$\xi(r) = \left(\frac{r}{r_0}\right)^{-\gamma}$$

where r_0 is the clustering length scale and γ the power-law exponent.

For local universe, $r_0 = 6$ Mpc and $\gamma \sim 1.38$

For cluster $r_0 \sim 20 h^{-1}$ Mpc

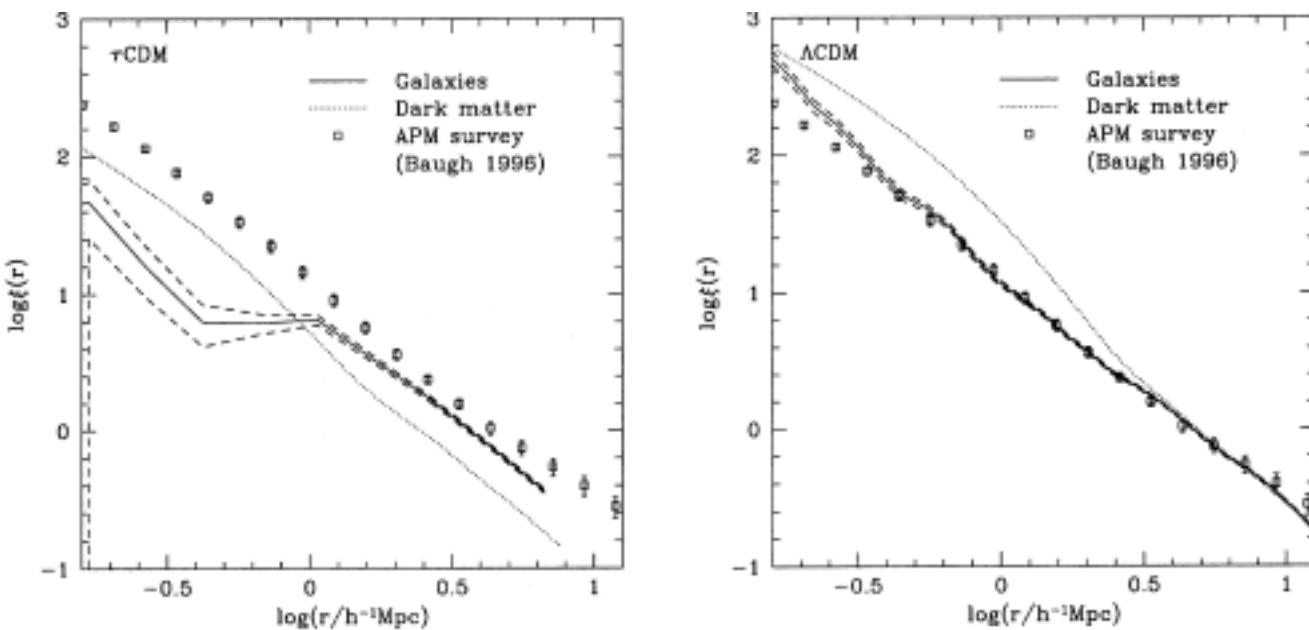
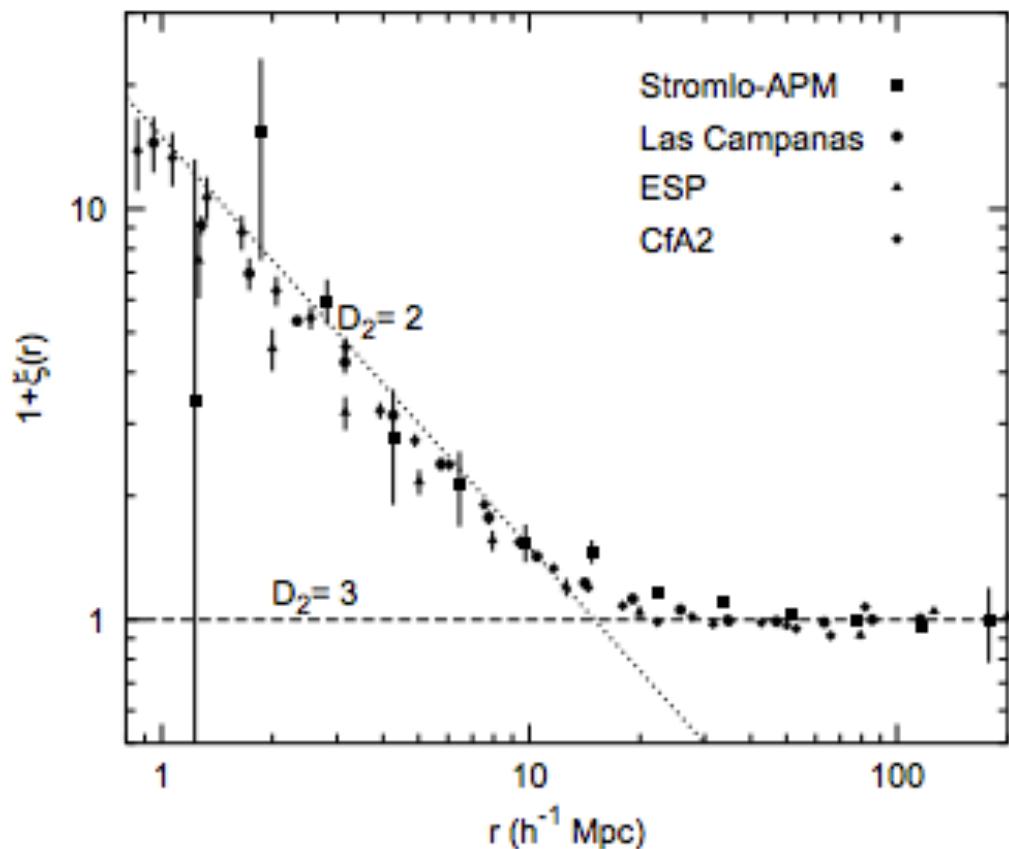


Figure 5. The left hand panel shows the two-point correlation function of galaxies brighter than $M_B - 5 \log h = -19.5$ in a τCDM model as a solid line. The dashed lines to either side indicate the Poisson sampling errors. This is compared to the observed APM real-space correlation function (points with error bars) and to the mass correlation functions in the N -body simulations (dotted line). The right hand panel shows the equivalent plot for a ΛCDM model.



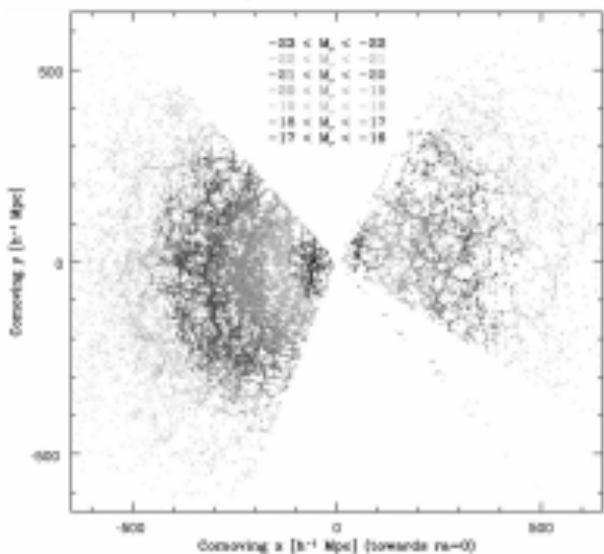
$$1 + \xi(r) \propto r^{D_2 - 3}$$

[astro-ph/0406086](https://arxiv.org/abs/astro-ph/0406086)

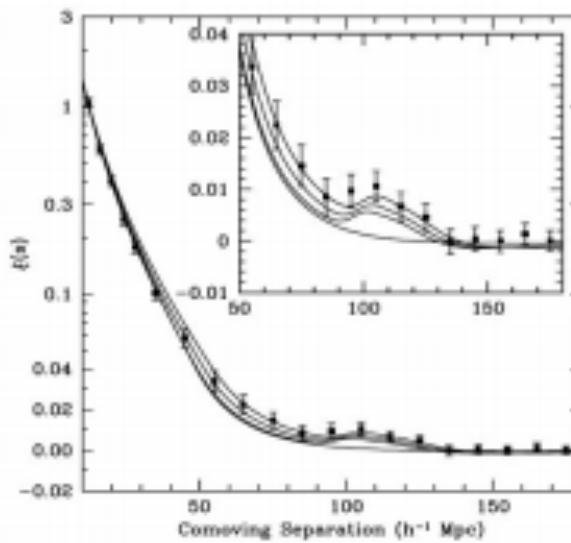
FIG. 10 The correlation function $1 + \xi(r)$ for different samples calculated with different estimators. We can see that the small scale fractal regime is followed by a gradual transition to homogeneity.

Example: Galaxy clustering (Sloan Digital Sky Survey)

Galaxy distribution



Two-point correlation function



Distribution of 67,676 galaxies in two slices of the sky showing strong anisotropic clustering (Tegmark et al. 2004).

Bottom: Two-point correlation function showing the faint feature around 100 megaparsec scales revealing cosmological
Baryonic Acoustic Oscillations (Eisenstein et al. 2005).
³⁰

Angular Two-Point Correlation Function

- Angular Two-Point Correlation function $w(\theta)$ can be defined as the conditional probability of finding a galaxy within solid angle $d\Omega$ lying at an angular distance θ from a given galaxy is given by :

$$dP = N[1 + w(\theta)]d\Omega$$

where N is the mean number density of galaxies per unit area in projected space

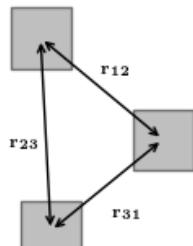
For galaxies

$$w(\theta) = \left(\frac{\theta}{\theta_0} \right)^\delta \quad \text{where } \delta = 1 - \gamma$$

Three-Point Correlation Function

- 2-point correlation function concept can be extended to higher orders by considering configurations of points comprising of triplets, quadruplets, etc

Probability of finding three points in volume elements dV_1, dV_2, dV_3 defined by a triangle with sides r_{12}, r_{13} , and r_{23} given by:



(b) 3 point

Three-Point Correlation function

$$dP_{123} = \rho^3 dV_1 dV_2 dV_3 (1 + \xi(r_{12}) + \xi(r_{23}) + \xi(r_{13}) + \zeta(r_{12}, r_{23}, r_{13}))$$

Computing the Two-Point Correlation Function

- 2-point correlation function can be estimated by calculating the excess or deficit of pairs of points within a distance r and $r+dr$, compared to a random distribution. Random distribution are selected with the same selection function as the data (i.e. within the same volume and within identical masked regions).

$$\hat{\xi}(r) = \frac{DD(r)}{RR(r)} - 1$$

where $DD(r)$ = no of pairs of data points

$RR(r)$ = no of pairs of random points (generated with density higher times than that of the data)

$DR(r)$ = no of data-random pairs

Landy-Szalay estimator

- Edges effects due to interaction between the distribution of sources and irregular survey geometry bias estimates of the correlation function. So therefore this is calculated using Landy-Szalay estimator.

$$\hat{\xi}(r) = \frac{DD(r) - 2DR(r) + RR(r)}{RR(r)} - 1$$

Landy and Szalay ApJ 412, 64 (1993)

L-S Estimator for Higher-Order Estimators

$$\hat{\xi}(r) = \frac{DDD(r) - 3DDR(r) + DRR(r) - RRR(r)}{RRR(r)}$$

(valid for an equilateral triangle)

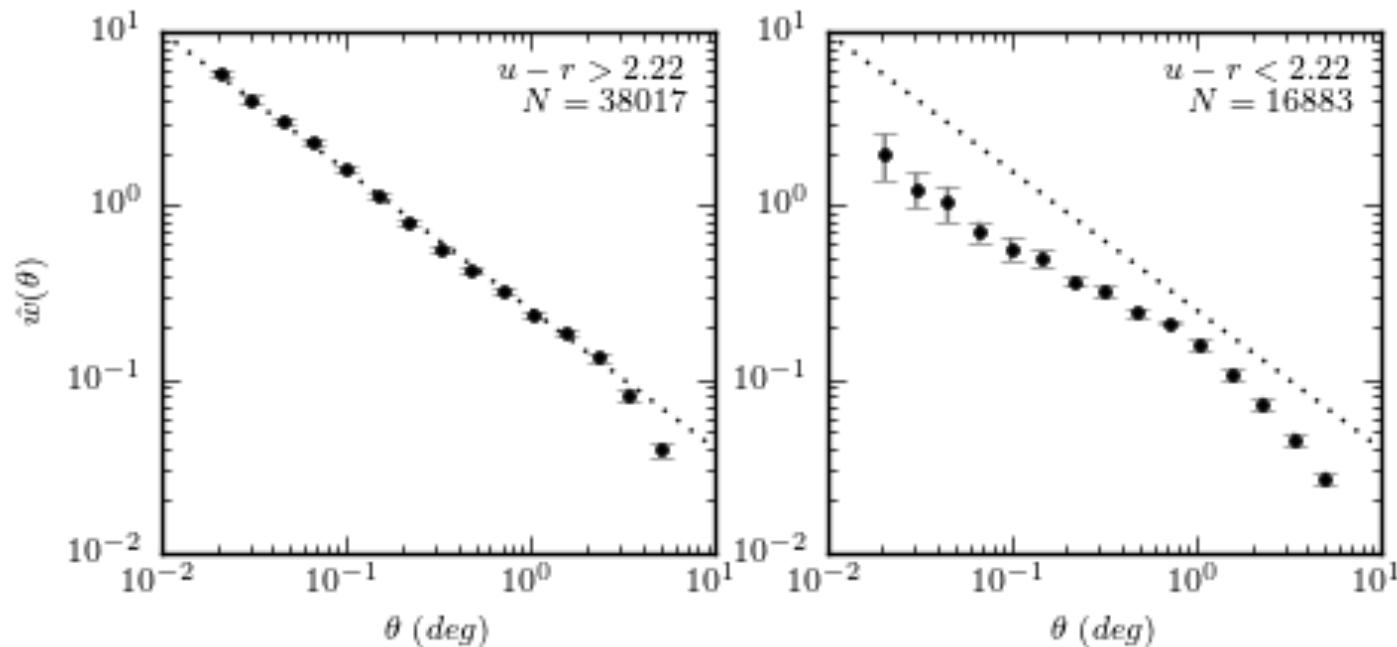
where DDD(r) is the number of data triplets and DDR(r), DRR(r) and RRR(r) are the associated configurations for the data-data-random, data-random-random, and random-random-random triplets.

Python Implementation

```
import numpy as np
from astroML.correlation import two_point_angular
RA = 40*np.random.random(1000)
DEC = 10*np.random.random(1000)
bins = np.linspace(0.1,10,11) # evaluate in 10 bins with
these edges
corr = two_point_angular (RA,DEC, bins, method='landy-
szalay')
```

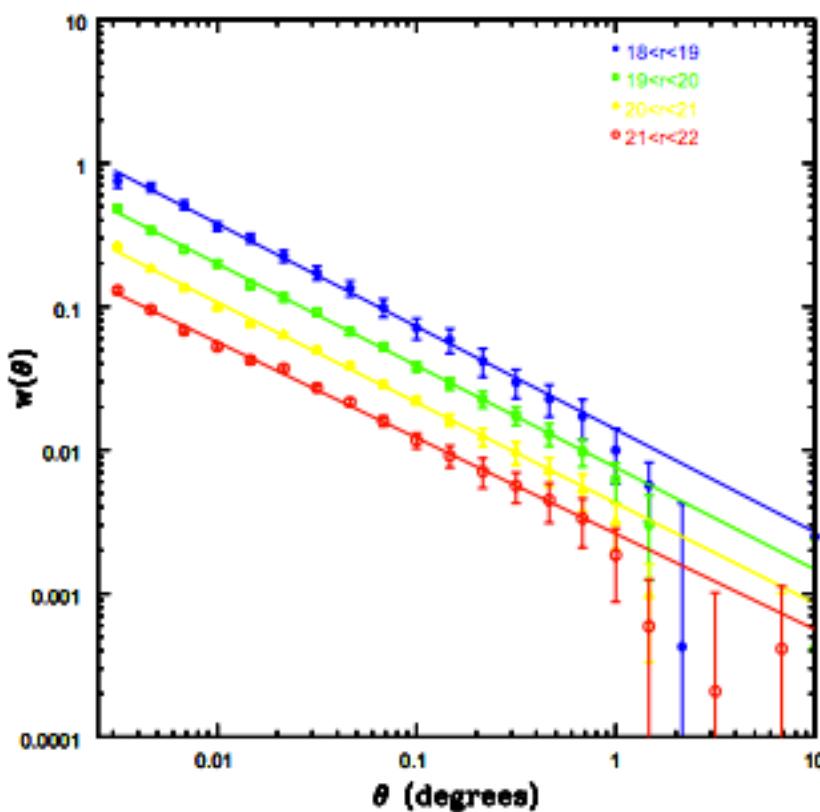
(only other option supported is 'standard')

$350 \text{ Mpc} < d < 8.4 \text{ Gpc}$



The two-point correlation function of SDSS spectroscopic galaxies in the range $0.08 < z < 0.12$, with $m < 17.7$. This is the same sample for which the luminosity function is computed in figure 4.10. Errors are estimated using ten bootstrap samples. Dotted lines are added to guide the eye and correspond to a power law proportional to $\theta^{-0.8}$. Note that the red galaxies (left panel) are clustered more strongly than the blue galaxies (right panel).

AstroML figure 6.14



astro-ph/0406086

FIG. 9 The angular correlation function from the SDSS as a function of magnitude from Connolly *et al.* (2002). The correlation function is determined for the magnitude intervals $18 < r^* < 19$, $19 < r^* < 20$, $20 < r^* < 21$ and $21 < r^* < 22$. The fits to these data, over angular scales of $1'$ to $30'$, are shown by the solid lines.

KdTrees and all that

Week 9+

Reference : Ivezic et al Chapter 2

References

- Ivezic et al, Chapter 2

http://andrewd.ces.clemson.edu/courses/cpsc805/references/nearest_search.pdf

<https://jakevdp.github.io/blog/2013/04/29/benchmarking-nearest-neighbor-searches-in-python/>

(Benchmarking of several python codes for KdTree)

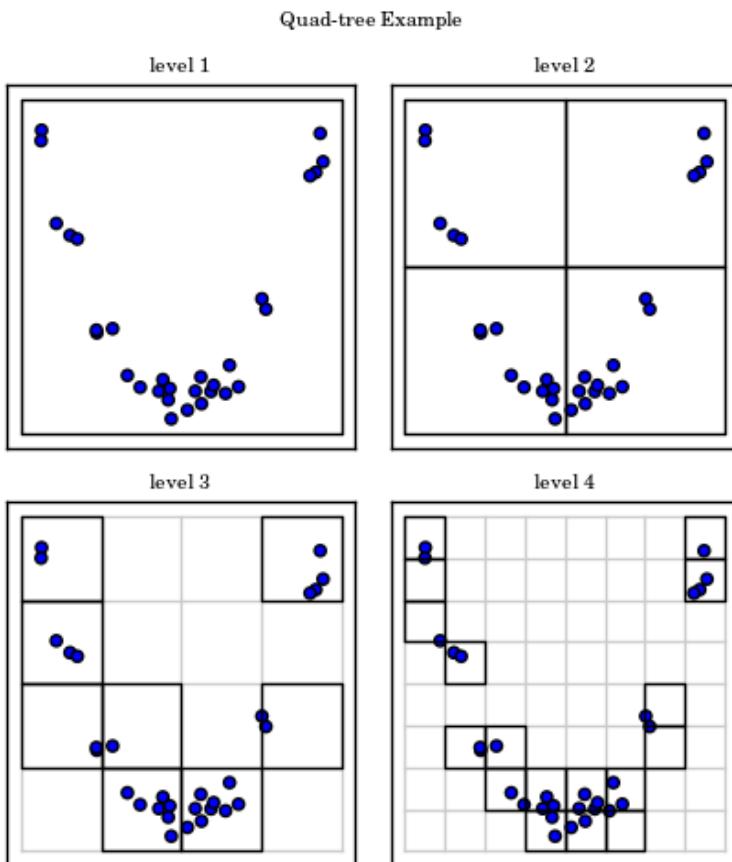
Introduction

- Question : How to accelerate nearest neighbor searches?

Problem : Brute Force search algorithms don't scale well with N $O(N^2)$.

Vectorizing (ala numpy, matlab) speeds up by a factor of 100, but with increased usage of memory.

Quad Trees and Oct Trees



Quad and *Oct* trees work in 2 and 3 dimensions respectively.

A quad tree is a simple data structure in which each tree node has exactly four children, representing its four quadrants. Each node is defined by its four numbers: left, right, top, and bottom quadrants.

By grouping points in this manner, one can progressively constrain the distance between a test point and a group of Points in a tree using the bounding box of each group of points to provide a lower bound on the distance between the query point and any point in the group.

If lower bound > best candidate nearest neighbor distance it can prune the group from the search

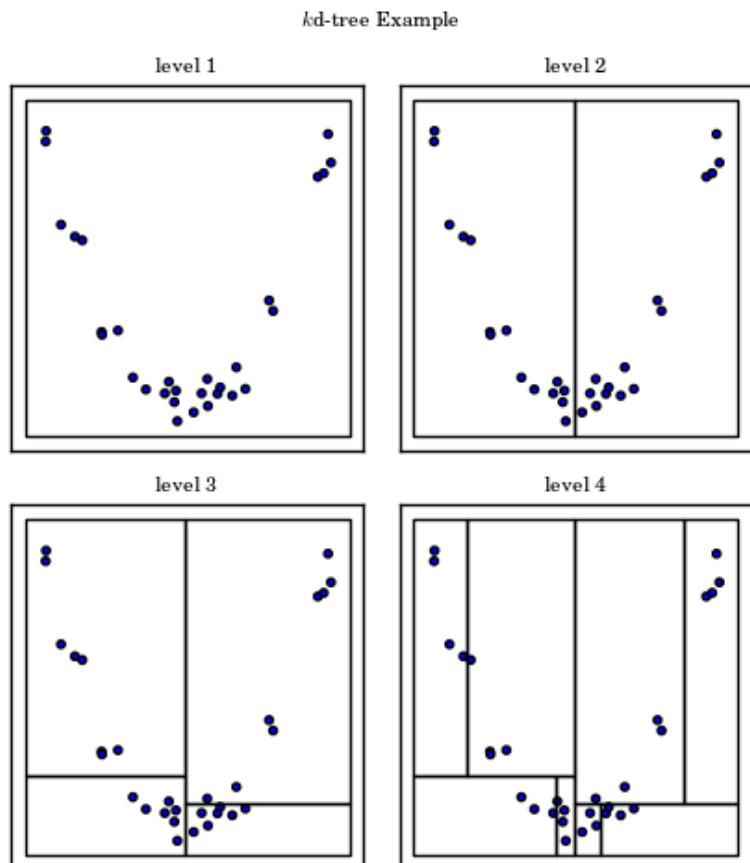
Speed up in Quad Tree

- Cost of nearest-neighbour search reduces to $O(\log N)$ for a single-query point.
- Building up the tree takes $O(N \log N)$ time to construct.

Motivation for Kd Trees

- Generalizing quad-tree and oct-tree to higher dimensions would imply that in D-dimensions, we would build a tree with 2^D children per node.
- Unfortunately size of tree would blow up for large D. For D=10, each node would require 2^{10} children (1024). To further divide each dimension into four units would require 10^6 nodes. This is called “**Curse of Dimensionality**”
- For D=100, would require 10^{15} petabytes of storage (Total volume of internet worldwide traffic in 2010)

Introduction to Kd Trees



Kd tree is a k -dimensional generalization of quad tree.

It is implemented as a *binary* tree. Each node has two children.

Top node of a k d tree is a D -dimensional hyperrectangle which contains the entire dataset.

To create the subnodes, volume is split into two regions along a simple dimension and this procedure is repeated recursively until the lowest node contains a specified number of points.

Kd Trees in Python

- KdTree available in scipy as well as sklearn.
Fast Kdtree available in scipy is shown below

```
import numpy as np
from scipy.spatial import KDTree
np.random.seed(0)
X=np.random.random((100000,3))
kdt = cKDTree(X) #build the Kdtree
kdt.query(X, k=2) #query for two neighbours
```

The above query takes 949 ms . OTOH a brute force search takes 23 minutes

Problems with Kd Trees

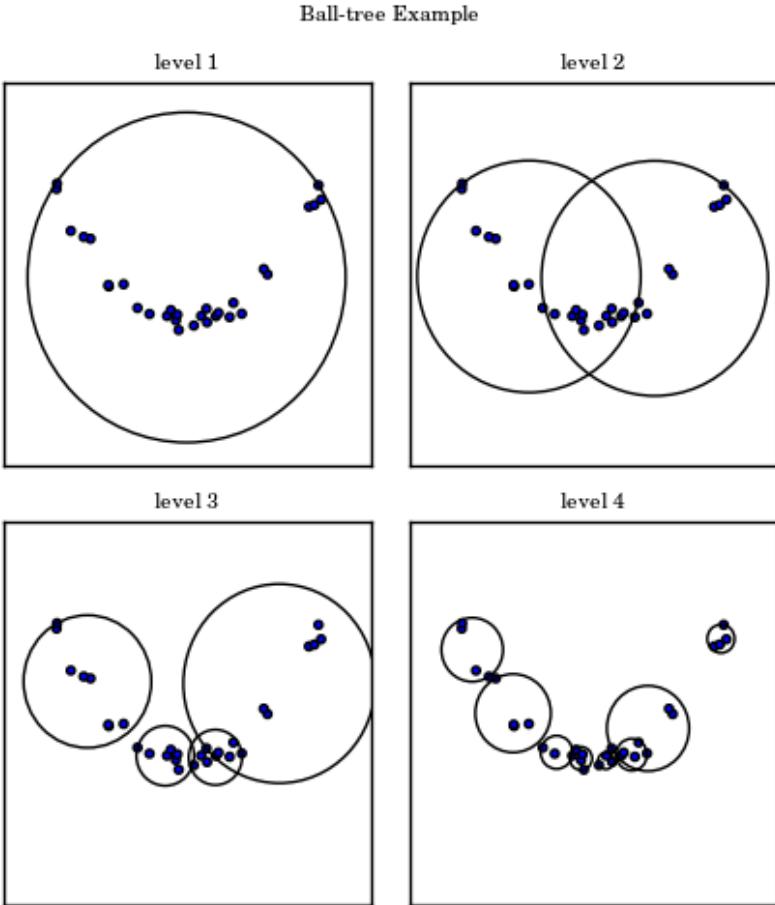
- Kdtree relies on rectilinear splitting of the data space, it is also subject to the curse of dimensionality
- Because the Kd-Tree splits along a single dimension in each level, one must go D levels deep before each dimension has been split.
- For $D=100$, we must create $2^{100} \sim 10^{30}$ nodes in order to split each dimension once.
- For N points in D dimensions, a kd tree will loose efficiency when $D \gg \log_2 N$

Ball Trees

- If x_1 is far from x_2 and x_2 is near x_3 , then x_1 is also far from x_3

Consequence of the Triangle inequality (which also applies to Euclidean distances)

$$D(x_1, x_2) + D(x_2, x_3) \geq D(x_1, x_3)$$



- Instead of building rectlinear nodes in D dimensions, ball-tree construction builds hyper-spherical nodes.
 - Each node defined by a centroid c_i and a radius r_i , such that $D(y, c_i) \leq r_i$ for every point contained in the node. With this Construction, given a point x outside the node One can show that for any point y in the node
- $$[D(x, c_i) - r_i] \leq D(x, y) \leq [D(x, c_i) + r_i]$$

Ball Trees in Python

```
import numpy as np
from sklearn.neighbors import BallTree
np.random.seed(0)
X=np.random.random((1000,3))
bt = BallTree(X) # build the balltree
bt.query(X, k=2)
```

Useful links to scipy/scikit learn documentation

Ball Tree

<http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.BallTree.html>

KDTree (scipy)

<https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.spatial.KDTree.query.html>
(Note that scipy has both KDTree and cKDTree)

KDTree (sklearn)

<http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KDTree.html>

Matching of Astrophysical Catalogs

```
radec_mags = NP.dstack((magnitudes['ra'],magnitudes['dec']))[0]
kdtree_object = spatial.cKDTree(radec_twomass)
distances, indexes =
kdtree_object.query(radec_mags,distance_upper_bound=match_radius/3600.)
nomatch = NP.isinf(distances)
indexes = indexes[~nomatch]
n_match = NP.sum(~nomatch)
```

Dimensionality Reduction

Week 9+

References

Principal Component Analysis

arXiv:1404.1100 (a tutorial on PCA)

arXiv:2211.14683

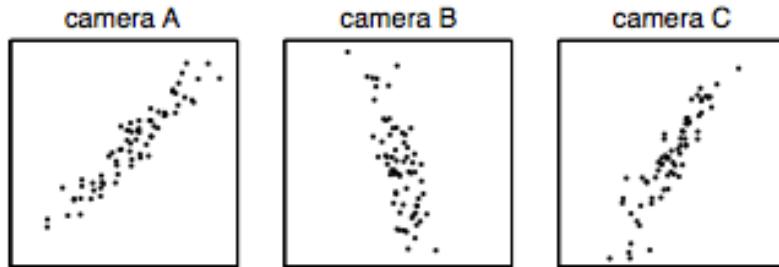
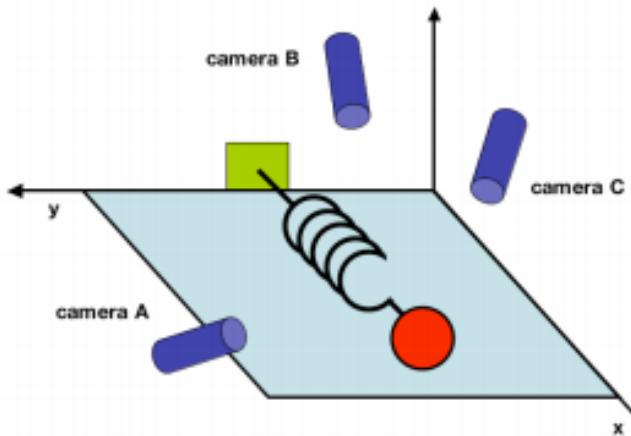


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

Question : How do we get from this data to a simple equation of x ?

Also how to account for noise?

Goal of PCA

- Identify the most meaningful basis to re-express a dataset

For the spring example explicit goal of PCA is to determine that “the dynamics is along the X-axis” or to determine that the unit vector along x-axis is the most important dimension

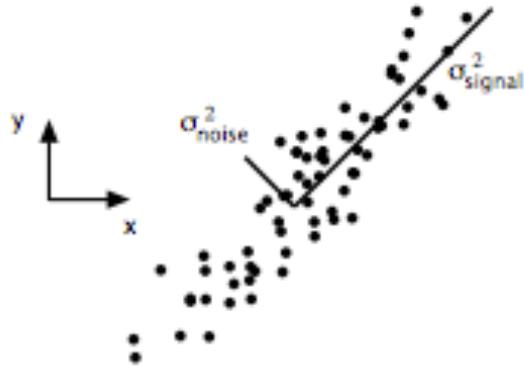


FIG. 2 Simulated data of (x, y) for camera A. The signal and noise variances σ_{signal}^2 and σ_{noise}^2 are graphically represented by the two lines subtending the cloud of data. Note that the largest direction of variance does not lie along the basis of the recording (x_A, y_A) but rather along the best-fit line.

Directions with largest variance of interest in our measurement space contain the dynamics of interest.



FIG. 3 A spectrum of possible redundancies in data from the two separate measurements r_1 and r_2 . The two measurements on the left are uncorrelated because one can not predict one from the other. Conversely, the two measurements on the right are highly correlated indicating highly redundant measurements.

Recording solely one response would express the data more concisely and reduce the number of sensor recordings.

Limitations of Principal Component Analysis

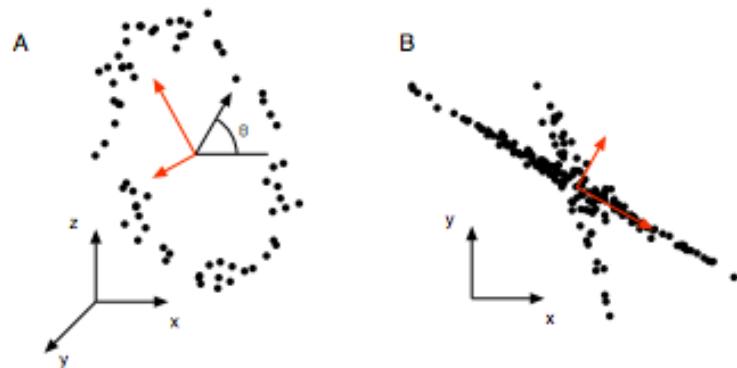
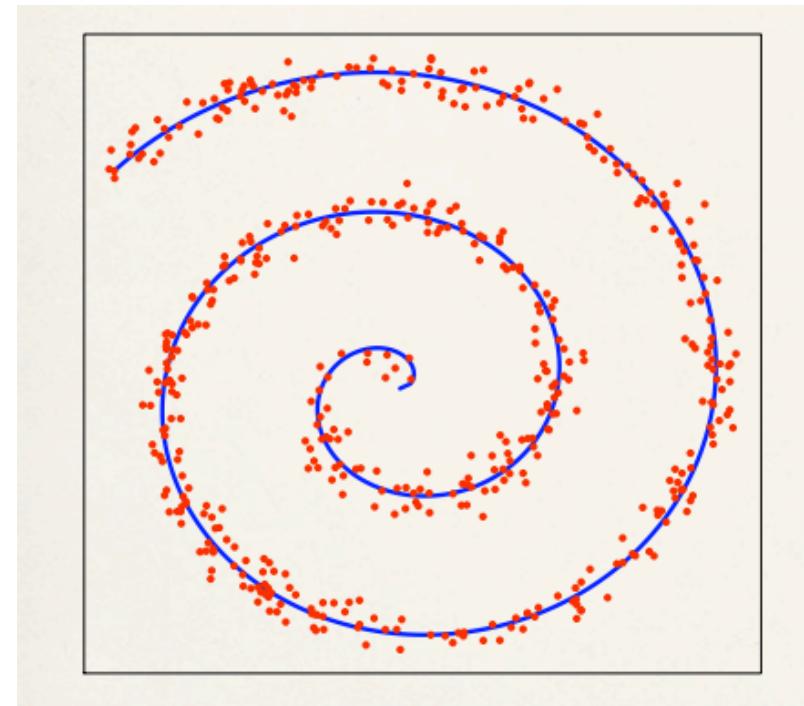


FIG. 6 Example of when PCA fails (red lines). (a) Tracking a person on a ferris wheel (black dots). All dynamics can be described by the phase of the wheel θ , a non-linear combination of the naive basis. (b) In this example data set, non-Gaussian distributed data and non-orthogonal axes causes PCA to fail. The axes with the largest variance do not correspond to the appropriate answer.



Derivation of Principal Component Analysis

Consider a set of data $\{x_i\}$ consisting of N observations, with each observation containing K features.

Subtract the mean of each feature in x_i and call this $N \times K$ matrix as X .

Covariance matrix of the centered data C_x is given by $C_x = \frac{1}{N-1} X^T X$

(where $N - 1$ term comes from fact that we are working with the sample covariance matrix)

Non-zero diagonal elements exists because there exist correlations between measured features.

Goal of PCA is identify a projection of $\{x_i\}$ that is aligned with the directions of maximal variance.

This projection can be written as $Y = XR$ and its covariance can be written as

$$C_Y = R^T X^T X R = R^T C_X R$$

with C_X the covariance of X as defined above.

First principal component, r_1 of R is defined as the projection with the maximal variance subject to the following constraint:

$$r_1^T r_1 = 1 \quad \text{Define a cost function} \quad \phi(r_1, \lambda)$$

$$\phi(r_1, \lambda) = r_1^T C_X r_1 - \lambda_1(r_1^T r_1 - 1)$$

Setting the derivative of $\Phi(r_1, \lambda)$ to 0 gives

$$C_X r_1 - \lambda_1 r_1 = 0$$

λ_1 is therefore the root of the equation $\det(C_X - \lambda_1 I) = 0$ and is an eigenvalue of the covariance matrix.

Variance for the first principal component is maximized when

$$\lambda_1 = r_1^T C_X r_1 \quad \text{is the largest eigenvalue of the covariance matrix}$$

The second (and further) principal components can be derived in an analogous manner by applying additional constraints to the cost function that the principal components are uncorrelated ($r_2^T C_X r_1 = 0$)

The second (and further) principal components can be derived in an analogous manner by applying additional constraints to the cost function that the principal components are uncorrelated ($r_2^T C_X r_1 = 0$)

Columns of R are then the eigenvectors or principal components and the diagonal values of C_Y define the amount of variance contained within each component.

With $C_X = R C_Y R^T$ and ordering the eigenvectors by their eigenvalues we can define the principal components of X

PCA through Singular Value Decomposition

$$X_1 = U_1 \Sigma_1 V_1^T$$

$$X_2 = U_2 \Sigma_2 V_2^T$$

Scaled SVD can be written as

$$U \Sigma V^T = \frac{1}{\sqrt{N-1}} X$$

Columns of U are left singular vectors

columns of V are right singular vectors and

Σ is a square, diagonal matrix of shape $[R \times R]$, where $R = \min(N, K)$ is the rank of the matrix X

U is $[N \times R]$ matrix $U^T U = 1$

V is $[R \times K]$ matrix $V^T V = 1$

Covariance matrix becomes

$$C_X = \left[\frac{1}{\sqrt{N-1}} X \right]^T \left[\frac{1}{\sqrt{N-1}} X \right]$$

$$\begin{aligned} C_X &= V \Sigma U^T U \Sigma V^T \\ &= V \Sigma^2 V^T \end{aligned}$$

Comparing with $C_X = R C_Y R^T$ we see that the right singular vectors V correspond to the principal components R , and the diagonal matrix of eigenvalues C_Y is equivalent to the square of the singular values.

$$\Sigma^2 = C_Y$$

SVD and Eigenvectors in Python

Singular value decomposition in Python can be accomplished using `svd` and `eigh` for computing the symmetric eigenvalue decomposition

```
import numpy as np
X = np.random.random ((100,3))
CX = np.dot(X.T,X)
U,Sdiag,VT = np.linalg.svd(X,full_matrices=false)
CYdiag, R = np.linalg.eigh(CX)
```

full_matrices : bool, optional

If True (default), u and v have the shapes (M, M) and (N, N) , respectively. Otherwise, the shapes are (M, K) and (K, N) , respectively, where $K = \min(M, N)$.

Follows the SVD convention two slides back and for both C_Y and Σ , only diagonal elements are returned.

svd puts the largest singular values first
eigh puts the smallest eigenvalue first

`np.allclose(Cydiag, Sdiag[::-1] **2)` # returns true if all elements are equal within a tolerance

True

`VT[::-1].T / R`

`array([[-1., 1., 1.],
 [-1., 1., 1.],
 [-1., 1., 1.]])`

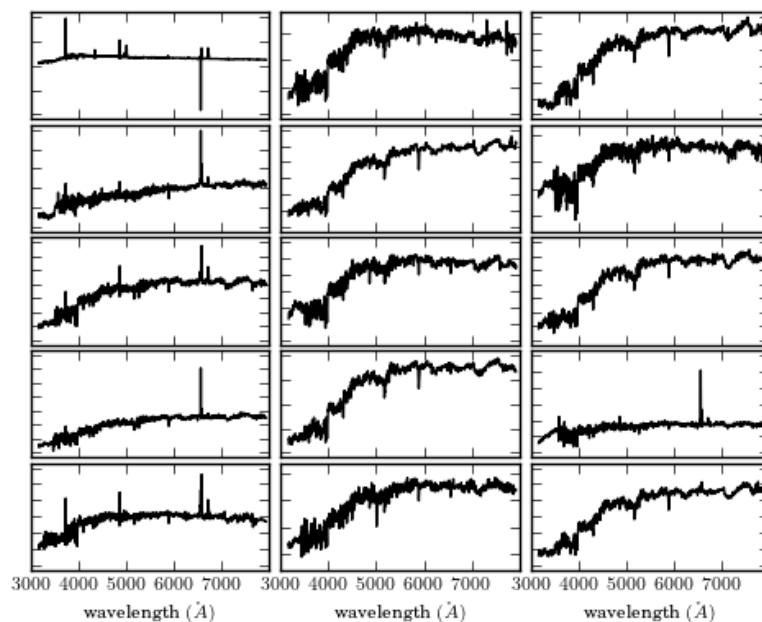
The eigenvectors of Cx and the right singular vectors of X agree up to a sign as expected.

PCA in Python

```
import numpy as np
from sklearn.decomposition import PCA
X=np.random.normal(size=(100,3))
R= np.random.random((3,10)) # projection matrix
X = np.dot(X,R) # X is 10 dimension with 5 intrinsic dimensions
pca= PCA(n_components=4) # no of components can be optionally set
pca.fit(X)
comp=pca.transform(X) # compute the sub-space projection of X
mean = pca.mean._ # length 10 mean of the data
components= pca.components_ # 4 x 10 matrix of components
var = pca.explained_variance_ # length 4 array of eigenvalues
```

For larger datasets RandomizedPCA is useful

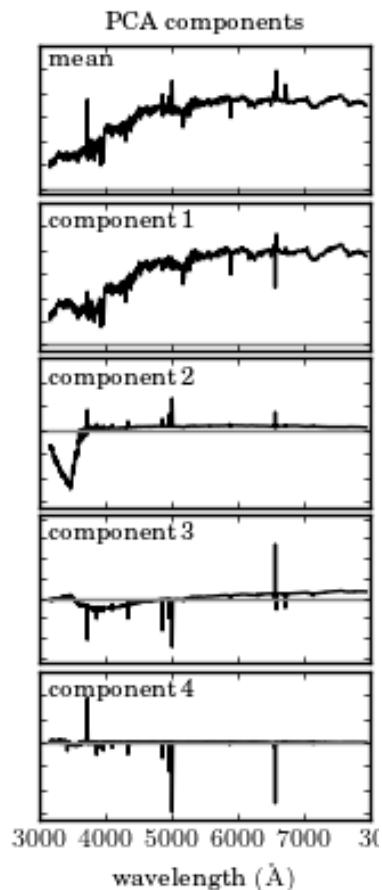
Dataset Used for Illustration of PCA



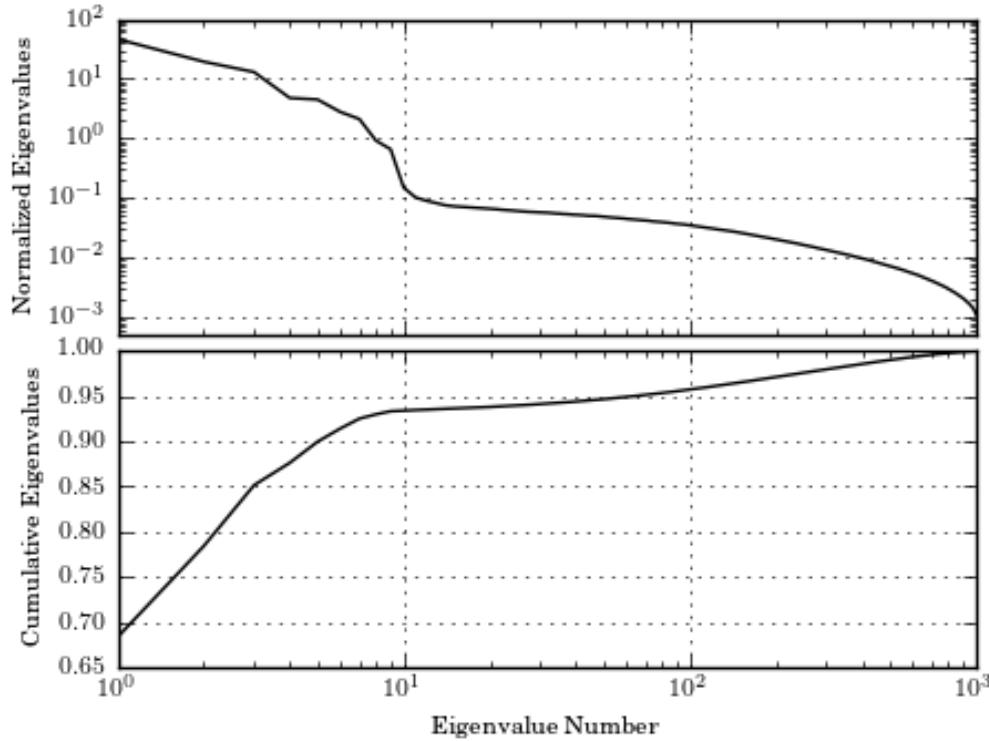
SDSS spectrum covering the Wavelength range from 3200 to 7800 Å in 1000 wavelength bins

A sample of 15 galaxy spectra selected from the SDSS spectroscopic data set (see Section 1.5.5). These spectra span a range of galaxy types, from star-forming to passive galaxies. Each spectrum has been shifted to its rest frame and covers the wavelength interval 3000-8000 Angstroms. The specific fluxes, $F_\lambda(\lambda)$, on the ordinate axes have an arbitrary scaling.

Data needs to be pre-processed before applying PCA (subtract the mean and divide by the variance)



Mean spectra and first four eigen-spectra
(normalized to unity)



First 10 eigenvectors responsible for 94% of the variance in the sample

Choosing first 10 components (out of 1000) allows a compression by a factor of 100

Also known as scree plots

The eigenvalues for the PCA decomposition of the SDSS spectra described in Section 7.3.2. The top panel shows the decrease in eigenvalue as a function of the number of eigenvectors, with a break in the distribution at ten eigenvectors. The lower panel shows the cumulative sum of eigenvalues normalized to unity. 94% of the variance in the SDSS spectra can be captured using the first ten eigenvectors.

Reconstruction of Spectrum from eigenspectra

$$x_i(k) = \mu(k) + \sum_j^R \theta_{ij} e_j(k)$$

i represents the number of the input spectra

j represents the eigen-spectrum number and k is the wavelength

$\mu(k)$ is the mean spectrum and θ_{ij} are the linear expansion coefficients derived from

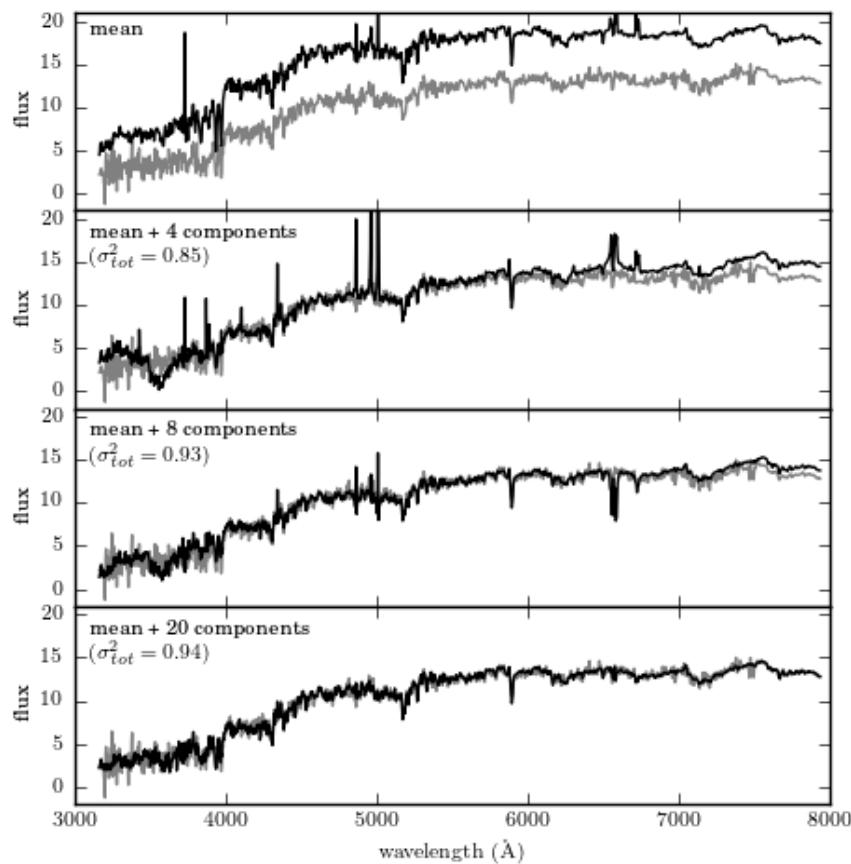
$$\theta_{ij} = \sum_k e_j(k)(x_j(k) - \mu(k))$$

Truncation of Eigenspectra

If the summation is over all eigenvectors the input vectors is fully described without any loss of information. Truncating the expansion to a value $r (< R)$ we get

$$x_i(k) = \sum_i^{r < R} \theta_i e_i(k)$$

will exclude those eigencomponents with smaller eigenvalues. These components reflect the noise within the dataset



Orthogonal components correlate strongly with specific physical properties (they relate to star formation and composition of the stellar types within a galaxy spectrum)

The reconstruction of a particular spectrum from its eigenvectors. The input spectrum is shown in gray, and the partial reconstruction for progressively more terms is shown in black. The top panel shows only the mean of the set of spectra. By the time 20 PCA components are added, the reconstruction is very close to the input, as indicated by the expected total variance of 94%.

Choosing the Level of Truncation

Qt : While reconstructing a data set from a linear combination of eigenvectors , how many components to keep?

- Total variance contained in the first r eigenvectors.

$$\frac{\sum_{i=1}^{i=r} \sigma_i}{\sum_{i=1}^{i=R} \sigma_i} < \alpha$$

Typical values for α range from 0.70 to 0.85

Choosing the Level of Truncation

- Shape of the scree plot : Sharp change in the gradient of the eigenvalue (knee) could be used to define the cutoff value. Knee is defined as

$$\Sigma_r^2 - \Sigma_{r+1}^2$$

- Plot logarithm of eigenvalue against number of eigenvectors (LEV diagram)
(If noise decays geometrically, variation in eigenvalues should drop as a linear function)

PCA with missing data

More details in AstroML book or Connolly & Szalay AJ 117, 142 (1999)

Nonnegative Matrix Factorization

One problem with PCA is that eigenvectors are defined relative to the mean data vector. Thus, principal components can be negative or positive.

Nonnegative matrix factorization (NMF) applies an additional constraint on the components that comprise the data matrix X

$$X = WY$$

where both all elements in W and Y are nonnegative.

NMF

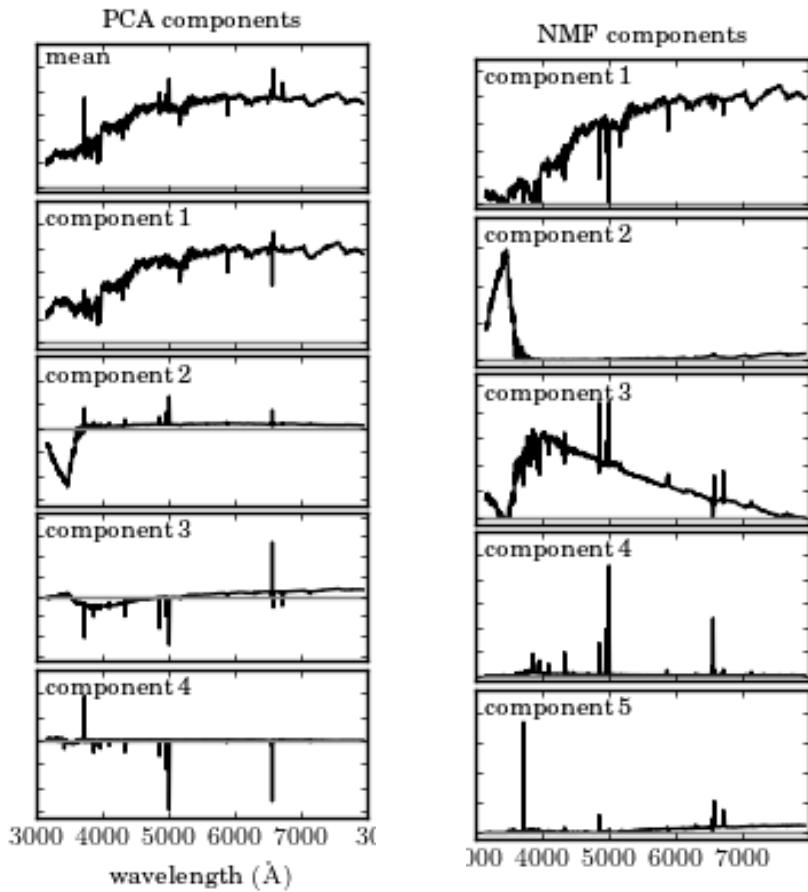
By minimizing the reconstruction error $\| (X - WY) \|^2$ one can show that nonnegative bases can be derived using a simple update rule

$$W_{ki} = W_{ki} \frac{[XY^T]_{ki}}{[WYY^T]_{ki}}$$

$$Y_{in} = Y_{in} \frac{[W^T X]_{in}}{[W^T W Y]_{in}}$$

n, k, i denote the wavelength, spectrum and template indices

Comparison of PCA and NMF



NMF does not require mean subtraction

- NMF components are broadly consistent with PCA but with different ordering of the basis functions.
- For the case of NMF, assumption is that each spectrum can be represented by a smaller no of non-negative components than the underlying dimensionality of the data.
- Number of components must be defined prior to the generation of the NMF and can be derived through cross-validation techniques

NMF in sklearn

```
import numpy as np
from sklearn.decomposition import NMF
X = np.random.random((100,3)) # 100 points in 3 dimensions, all
positive
nmf = NMF(n_components=3) # setting n_components is optional
nmf.fit(X)
Proj= nmf.transform(X) # project to 3 dimensions
Comp = nmf.components_ # 3 x 10 array of components
err = nmf.reconstruction_err_ # how well 3 components capture data
```

For more information check out sklearn documentation on
Non-negative matrix factorization

Additional applications of NMF

- Computer Vision
 - Identifying/Classifying Images
 - Object Tracking
 - Generally Reducing Feature Space of Images
- Text mining
- Speech denoising with stationary noise (Separate into speed parts and variable noise parts)

Source : <https://www.youtube.com/watch?v=UQGEB3Q5-fQ> by Steven Ford

NMF is computationally less expensive than PCA and can be extended to data with error bars. It is also suited to datasets with low error bars.

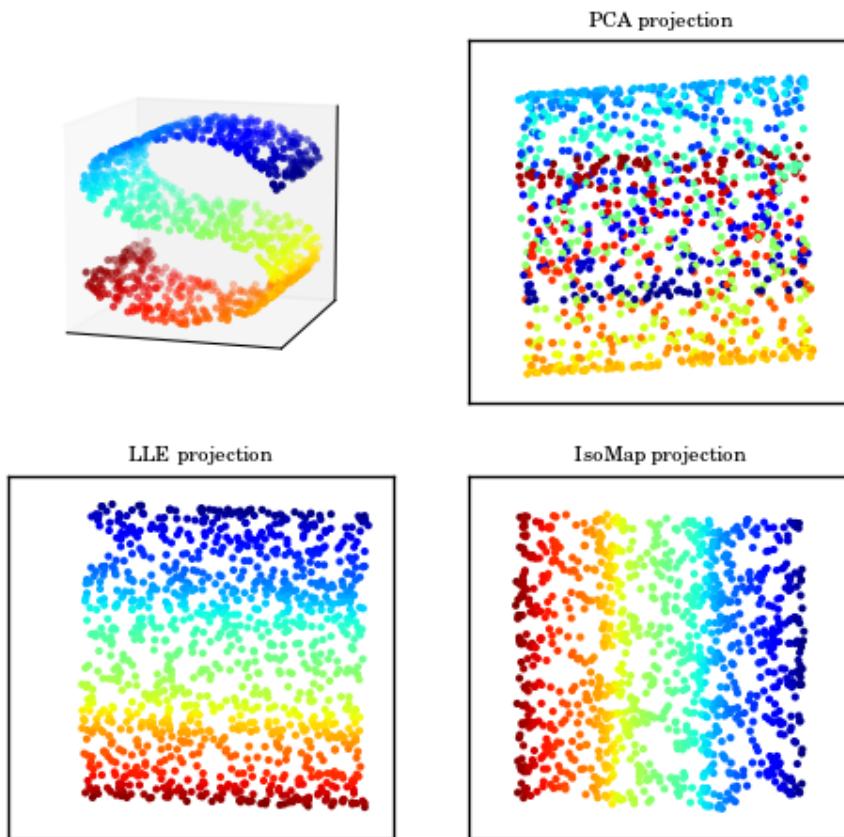
References on NMF

https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

Manifold Learning

- One limitation with PCA, NMF etc are that they are linear dimensionality techniques. Real-world datasets have non-linear features, which are hard to capture with a simple linear basis.
(In astrophysics emission line galaxies and quasars require upto ~30 linear components to characterize. These emission lines are non-linear features of the spectra and non-linear methods are required to project that info on to fewer dimensions. Jvdp and Connolly found that two non-linear components sufficient to recover galaxy spectra, whereas dozens of components required in linear projection)
- Manifold Learning Techniques comprise a set of techniques which accomplish this non-linear dimensionality reduction.

Comparison of PCA and Manifold Learning



Locally Linear Embedding

- Locally Linear Embedding is an unsupervised learning algorithm, which attempts to embed high-dimensional data in lower-dimensional space, while preserving geometry of local neighborhoods of each point. These local neighborhoods are determined by relation of each point to k nearest neighbors.
- Two step algorithm:
 - For each point a set of weights is derived which best reconstruct the point from its k nearest neighbors. These weights encode the local geometry of the local neighborhood.
 - With these weights fixed a new lower-dimensional data is found which maintains the neighborhood relation described by these weights.

Math Behind LLE First Step

Let X be an $N \times K$ matrix representing N points in K dimensions. We seek an $N \times N$ weight matrix W which minimizes the reconstruction error

$$\mathcal{E}_1(W) = |X - WX|^2$$

subject to certain constraints on W .

Rewrite this equation in component form :

$$\mathcal{E}_1(W) = \sum_{i=1}^N |x_i - \sum_{j=1}^N W_{ij}x_j|^2$$

First Step of LLE

Qt : Geometrical interpretation of minimizing previous equation:

What we are doing is finding a linear combination of points in the dataset which best reconstructs each point from the others.

Essentially finding the hyperplane that best describes the local surface at each point within the data set.

Each row of the weight matrix gives a set of weights for the corresponding point.

- To prevent the trivial solution ($W=I$) posit that the diagonal elements are zero.
- The key insight of LLE is to constrain ALL $W_{ij} = 0$ except when point j is one of the k nearest neighbors of point i

Out of N^2 entries in W , only Nk are non-zero. The matrix W becomes very sparse for $k \ll N$. The rows of W encode the *local* properties of the data set: how much each point relates to its nearest neighbours.

Second Step of LLE

Second step is similar to the first step, but instead seeks an $N \times d$ matrix Y , where $d < D$ is the dimension of the embedded manifold. Y is found by minimizing :

$$\mathcal{E}_2(Y) = |Y - WY|^2$$

where W is kept fixed (and obtained from step 1)

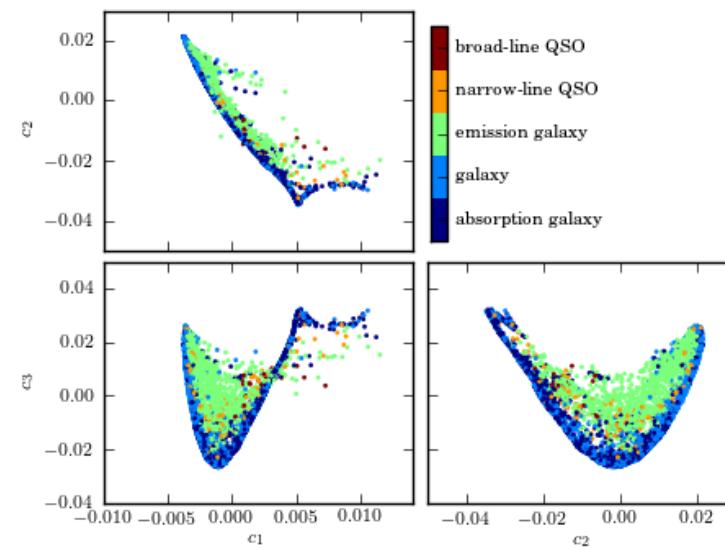
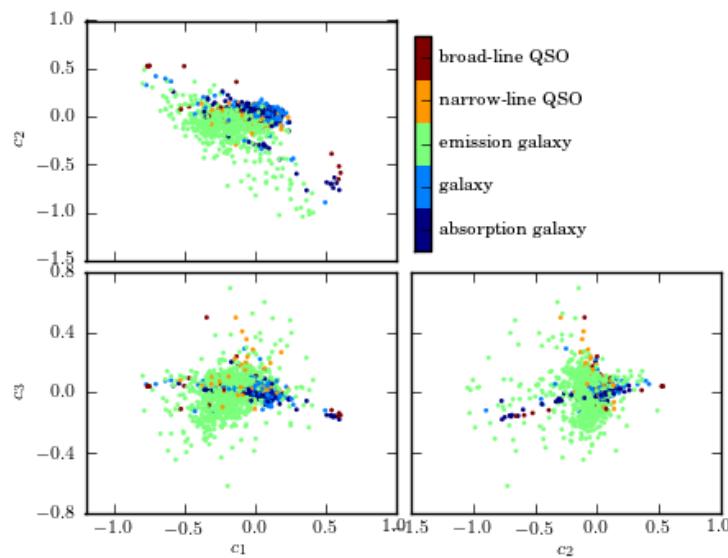
This is similar to the corresponding equation in first step. Because of the Symmetry and constraints on W , local neighborhoods in low-dimensional embedding will reflect the properties of the local neighborhoods in X

LLE in Python

Solutions to the two minimization equations are found using linear-algebra techniques (ARPACK etc, found in `scipy.sparse.linalg.eigsh`)

```
import numpy as np
from sklearn.manifold import LocallyLinearEmbedding
X = np.random.normal(size=(1000,2)) # 1000 pts in 10 dim
R = np.random.random((2,10)) # 2-D linear manifold in 10-D space
X = np.dot(X,R)
k = 5 # no of neighbours used in the fit
n = 2 # no of dimensions used in th efit
lle = LocallyLinearEmbedding(k,n)
lle.fit(X)
Proj=lle.transofrmation(X) # 100 x 2 projection of data.
```

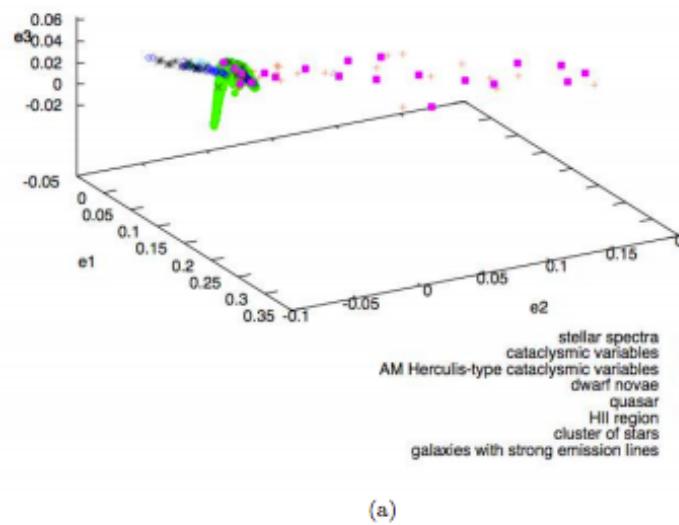
Comparison of LLE and PCA



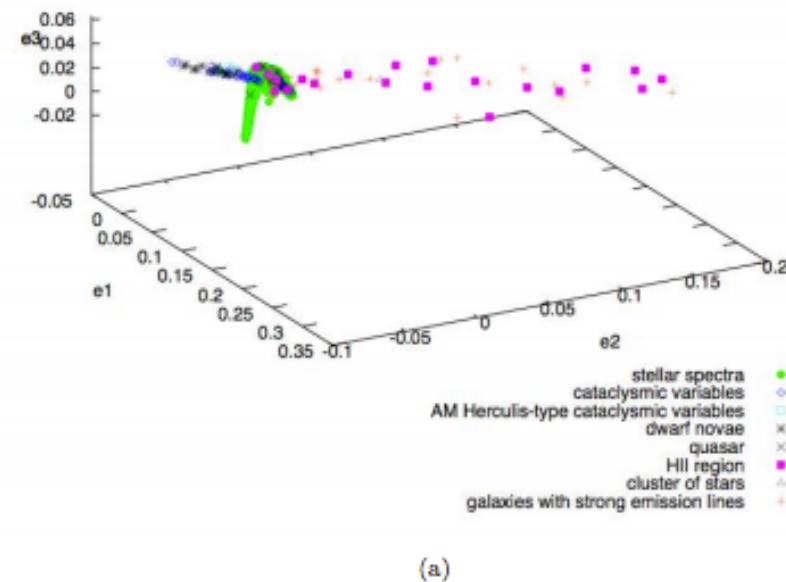
A comparison of the classification of quiescent galaxies and sources with strong line emission using LLE and PCA. The top panel shows the segregation of galaxy types as a function of the first three PCA components. The lower panel shows the segregation using the first three LLE dimensions. The preservation of locality in LLE enables nonlinear features within a spectrum (e.g., variation in the width of an emission line) to be captured with fewer components. This results in better segregation of spectral types with fewer dimensions.

Application of LLE to Astrophysical data

Applied to galaxy spectra, stellar spectra and photometric light curves.



(a)



(a)

arXiv: 1110.4646 Classification of SDSS stellar spectra

IsoMap (Isometric Mapping)

- Isomap based on multi-dimensional scaling (MDS) framework . Method to reconstruct a dataset from a matrix of pairwise distances.

If one has a dataset represented by an $N \times K$ matrix X , then one can trivially compute $N \times N$ distance matrix D_x such that $[D_x]_{ij}$ contains the distance between points i and j .

Classical MDS reverses this operation.

Given a distance matrix D_x , MDS discovers a new dataset Y which minimizes the error

Given a distance matrix D_x , MDS discovers a new dataset Y which minimizes the error

$$\mathcal{E}_{XY} = |\tau(D_x) - \tau(D_y)|^2$$

where τ is given by

$$\tau(D) = \frac{HSH}{2}$$

Where S is the matrix of square distances $S_{ij} = D_{ij}^2$ and H is the centering matrix given by :

$$H_{ij} = \delta_{ij} - 1/N$$

With this choice of τ it can be shown that the optimal embedding Y is identical to the top D eigenvectors of the matrix $\tau(Dx)$

The key insight of IsoMap is that we can use this metric MDS framework to derive a nonlinear embedding by constructing a suitable proxy for the distance matrix Dx

IsoMap recovers the nonlinear structure by approximating geodesic curves which lie within the embedded manifold and computing the distance between each point in the dataset along these geodesic curves, which lie within this Embedded manifold and computing the distance between point in the dataset along these geodesic curves.

IsoMap Algorithm (wikipedia)

Algorithm [edit]

A very high-level description of **Isomap** algorithm is given below.

- Determine the neighbors of each point.
 - All points in some fixed radius.
 - K nearest neighbors.
- Construct a neighborhood graph.
 - Each point is connected to other if it is a K nearest neighbor.
 - Edge length equal to Euclidean distance.
- Compute shortest path between two nodes.
 - Dijkstra's algorithm
 - Floyd–Warshall algorithm
- Compute lower-dimensional embedding.
 - Multidimensional scaling

Isomap in Python

```
import numpy as np
from sklearn.manifold import Isomap
X = np.random.normal(size=(1000,2)) # 1000 pts in 2
dimensions
R = np.random.random((2,10)) # projection matrix
X = np.dot(X,R) # X is now a 2-D manifold in 10 D space
k = 5 # no of neighbours used in the fit
n = 2 # number of dimensions used in the fit
iso = Isomap(k,n)
iso.fit(X)
proj = iso.transform(X) # 1000 x 2 projection of data
```

Another example of Manifold learning is Laplacian Eigenmap

Weakness of Manifold Learning

- Not suited to fitting data plagued by noise or gaps
- Nonlinear projection obtained using these techniques depend on the set of nearest neighbours used for each point. No solid recommendation in literature for choosing optimal set of neighbors for a given embedding.
- Choice of output dimensionality is a free parameter. (Usually $d=1$ or $d=2$ or $d=3$ is chosen), as it leads to a projection which is easy to visualize.
- Very sensitive to outliers. A single outlier can short-circuit the manifold
- Manifold learning methods do not provide a set of basis functions. Projection derived from these methods cannot be used to compare data in same way as in PCA

Independent Component Analysis

- ICA is popular in the biomedical signal processing community to solve the “cocktail party problem” (Multiple microphones located in a room containing N people. Each microphone picks up a linear combination of the N voices. Goal of ICA is to isolate the individual signals)

Each galaxy spectra can be considered a linear combination of input signals from individual stars and HII regions.

- ICA is not used for reducing dimensionality.

Example of ICA



Math behind ICA

$$x_1(k) = a_{11}s_1(k) + a_{12}s_2(k) + a_{13}s_3(k) + \dots$$

$$x_2(k) = a_{21}s_1(k) + a_{22}s_2(k) + a_{23}s_3(k) + \dots$$

$$x_3(k) = a_{31}s_1(k) + a_{32}s_2(k) + a_{33}s_3(k) + \dots$$

$s_i(k)$ are the individual stellar spectra and a_{ij} the mixing amplitudes. In matrix format this can be written as $X=AS$, where

X = input spectra matrix

S = stellar spectra matrix

Equivalent to solving the weight matrix W , such that

$$S=WX$$

Principle behind ICA

- Input signal $s_i(k)$ should be statistically independent. Two random variables are considered statistically independent if their joint probability distribution can be fully described by their marginalized probabilities.

$$f(x^p, y^q) = f(x^p)f(y^q)$$

where p,q are arbitrary higher-order moments of the probability distributions.

(For the case of PCA, p=q=1 and the statement of independence simplifies to the weaker condition of uncorrelated data)

In most implementations of ICA algorithms, the requirement for statistical independence is expressed in terms of non-gaussianity of probability distributions
-> identify unmixing matrix which maximizes non-gaussianity of the distributions

ICA in Python

Scikit-learn implementation of ICA is based on the FastICA algorithm.

```
import numpy as np
from sklearn.decomposition import FastICA
X = np.random.normal(size=(100,2)) # 100 pt in 2 dimensions
R=np.random.random((2,5)) # mixing matrix
X = np.dot(X,R) # X is now 2D data in 5-d space
ica=FastICA(2)
ica.fit(X)
proj = ica.transform(X) # 100 x 2 projection of data
comp=ica.components_ # 2 x 5 matrix of indep. components(unmixing
matrix)
Sources = ica.sources_ #100 x 2 matrix of sources (original
sources)
```

A comparison of the decomposition of SDSS spectra using PCA (left panel - see Section 7.3.1), ICA (middle panel - see Section 7.6) and NMF (right panel - see Section 7.4). The rank of the component increases from top to bottom. For the ICA and PCA the first component is the mean spectrum (NMF does not require mean subtraction). All of these techniques isolate a common set of spectral features (identifying features associated with the continuum and line emission). The ordering of the spectral components is technique dependent.

