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.

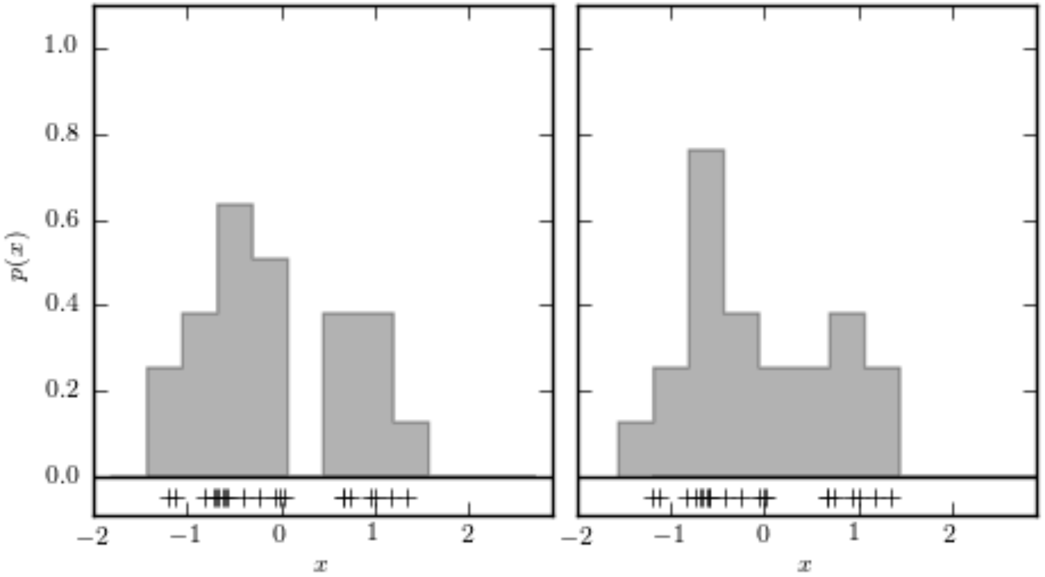
How to estimate density *non-parametrically* without specifying a specific functional form.

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

Simplest method is **Kernel Density Estimation**

Kernel Density Estimation

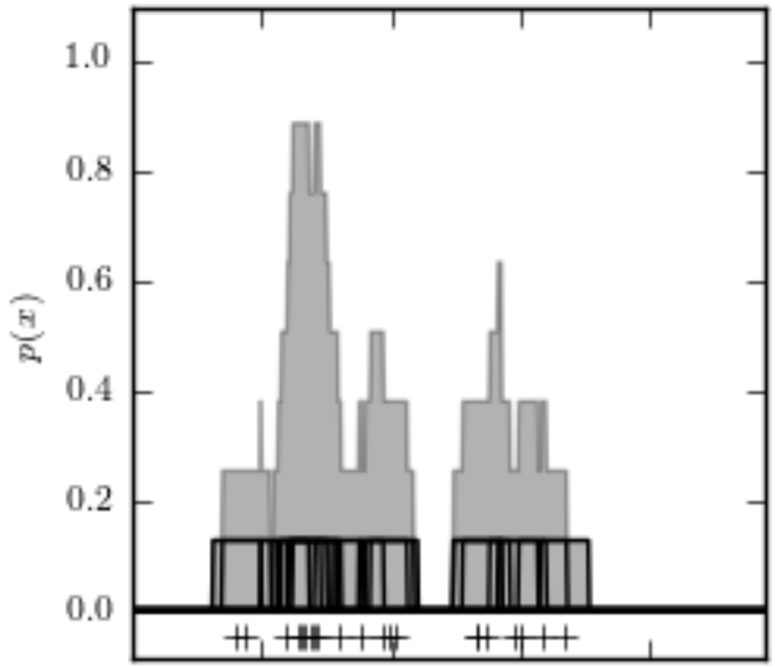
Problems with histograming data



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.

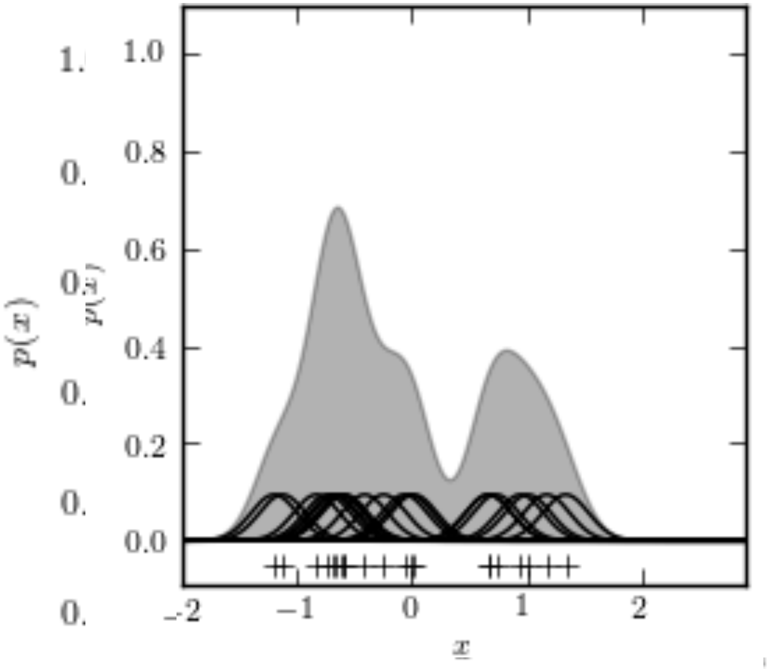
→ Each point is replaced by a box of unit height and some predefined width.



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 {xi}, 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) >= 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 $\sigma_k^2 = \int u^2 K(u)du > 0$

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

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

Exponential Kernel

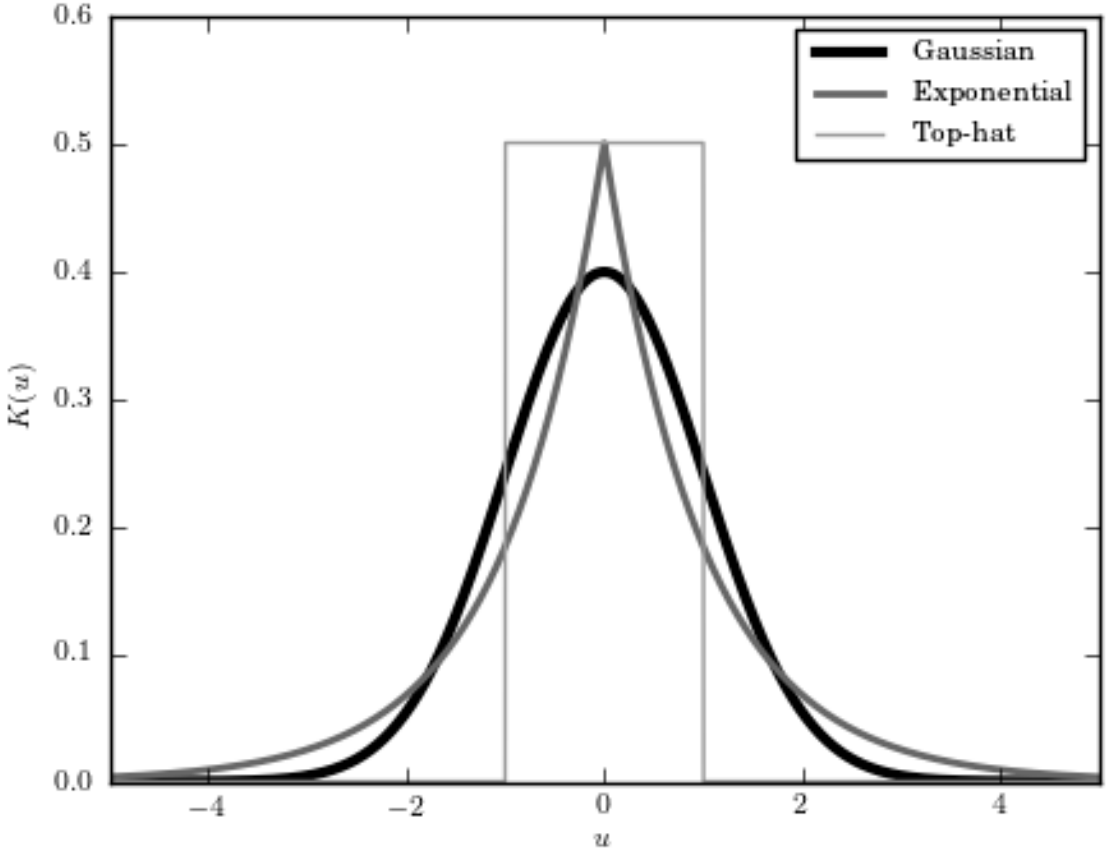
$$K(u) = \frac{1}{D!V_D(1)} e^{-|u|}$$

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 lxl <=1 and 0 otherwise.

Comparison of Kernels



Selecting the KDE bandwidth using cross-validation

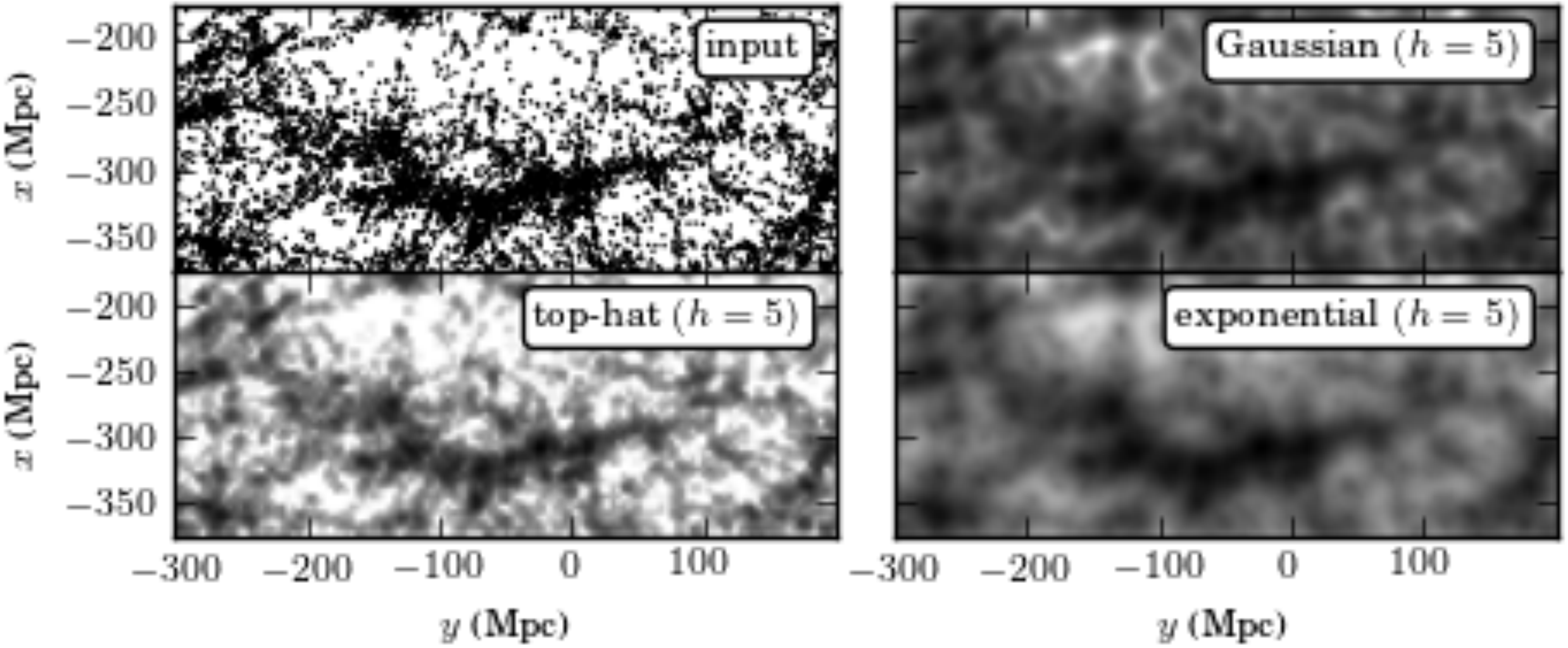
- Evaluate the cost on *out-of-sample* data. (i.e. points not in the training set)

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

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.neighbours 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.

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

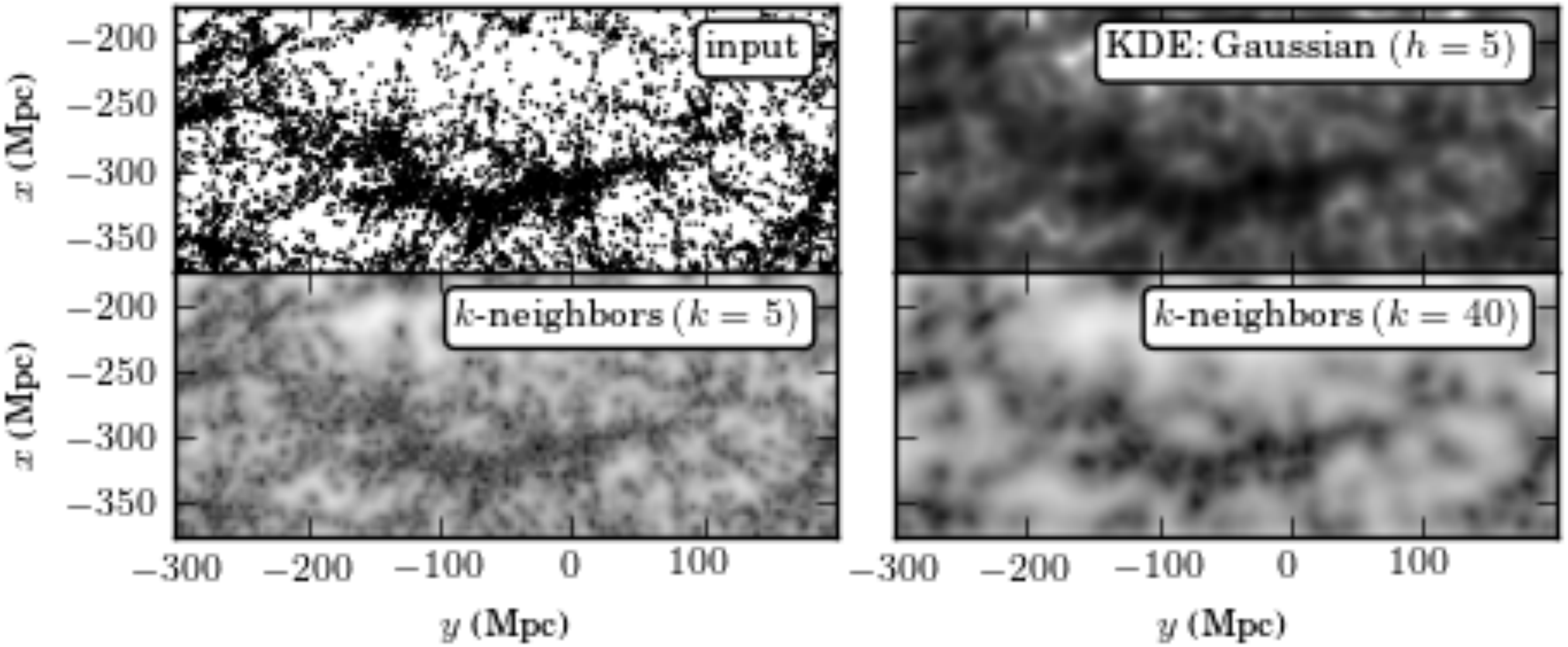
where d_i is the distance to ith nearest neighbor

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

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

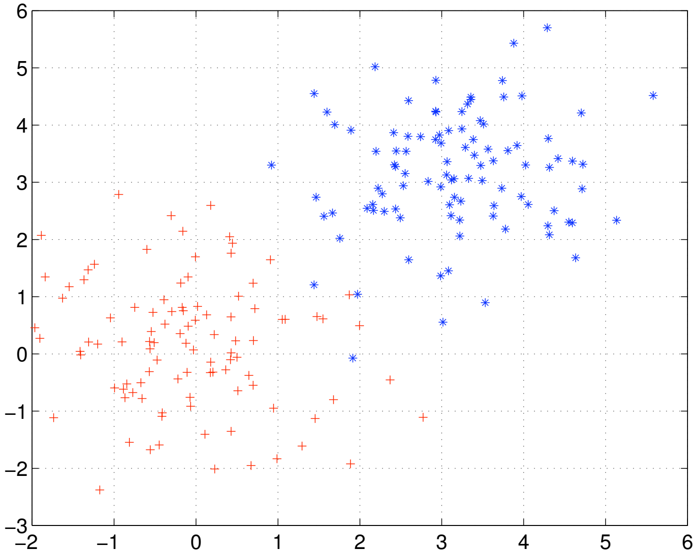
- Given multivariate point dataset, does it display any structure or concentration of points .
- Alternatively, when a density estimate is available, we can search for "overdensities"
- Seek a partitioning or segmentation of data into smaller parts according
- Unsupervised means no prior information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

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

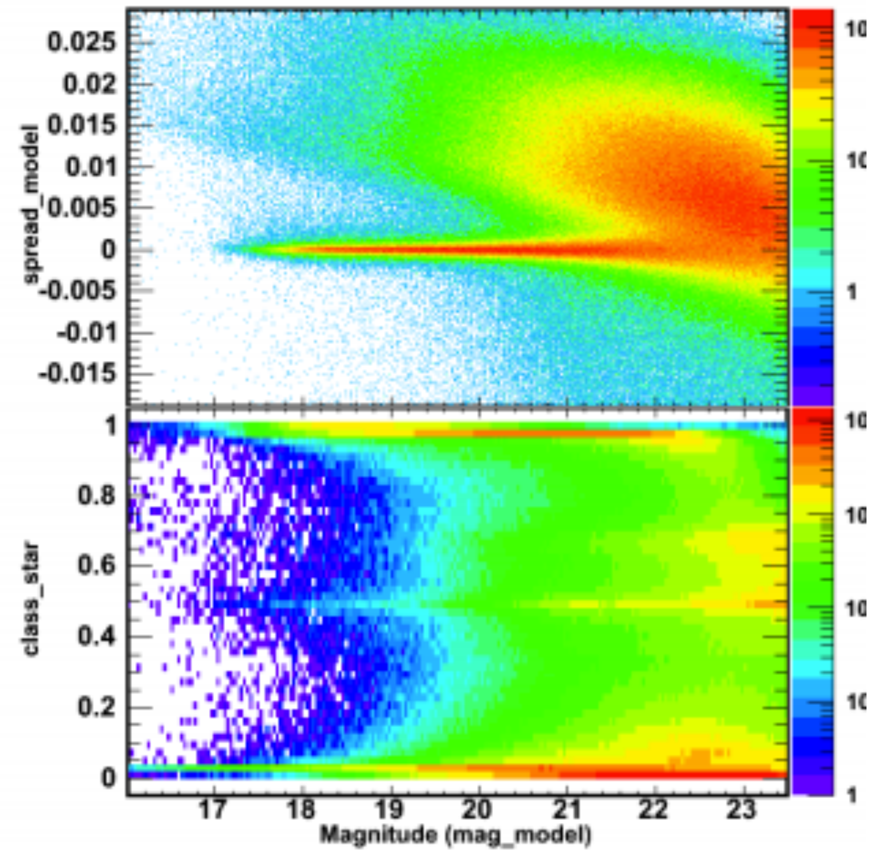
Lectures by Jia Li (PSU) and T. Krishnan (Strand Life Sciences)

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

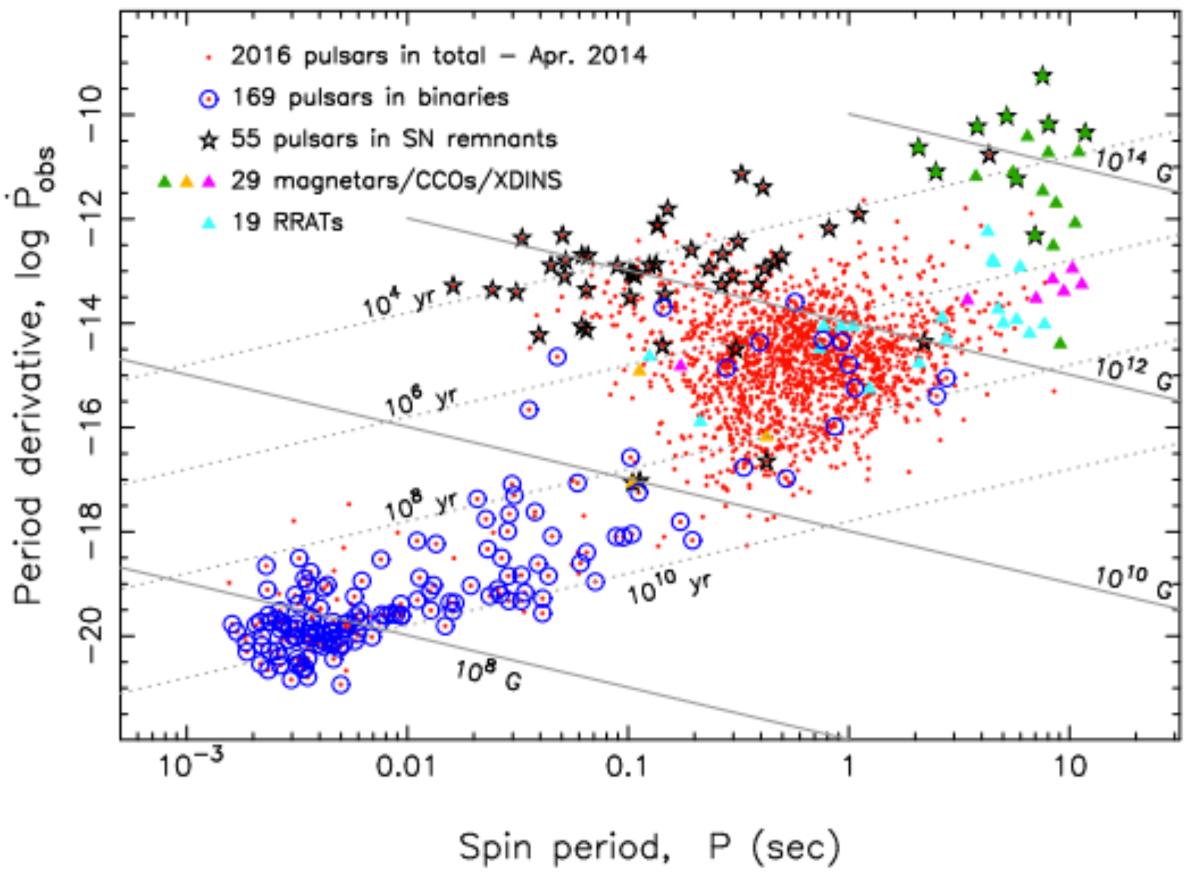


Fig. 3. P-$\dot{P}$ diagram. Red dots indicate known radio pulsars as of April 2014. Blue circles represent binaries. Stars represent associated supernova remnants. Magnetars are represented by green triangles (see §5). The XDINS and CCOs (see §4,5) are pink and yellow triangles, respectively. RRATs (§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).

arXiv:1602.07738

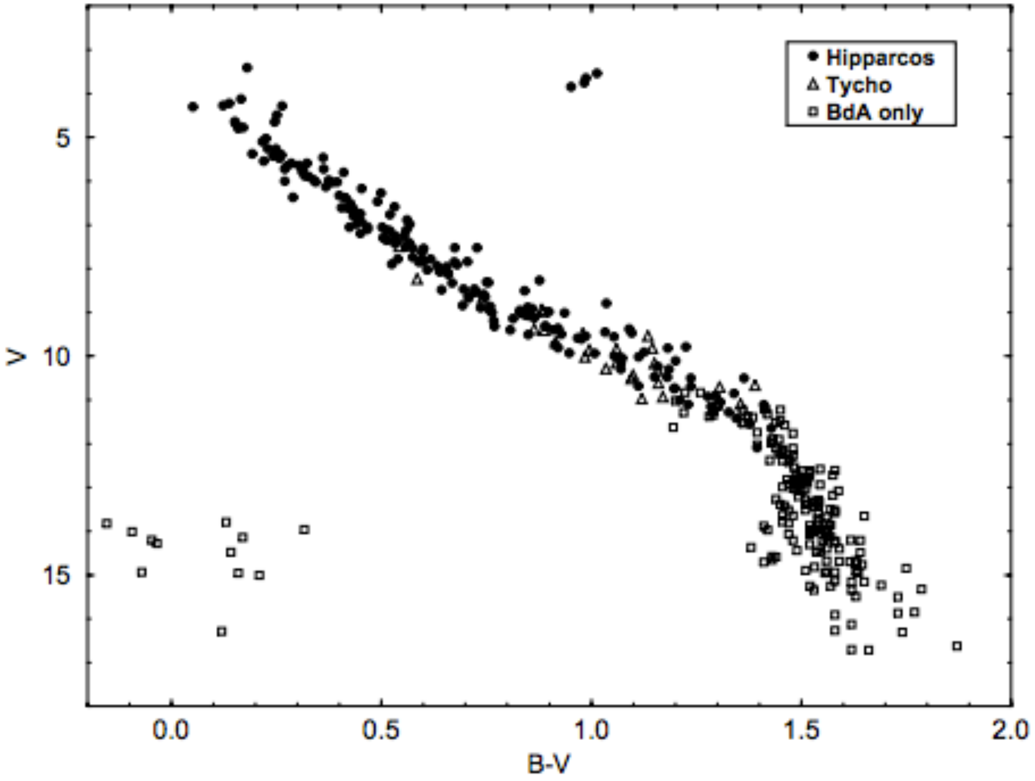


Fig. 2. Hyades stars from the 'Base des Amas' (BDA). Stars contained in the Hipparcos Catalogue are displayed as filled circles (190 stars). Stars not contained in the Hipparcos Catalogue, but appearing in the Tycho Catalogue, are displayed as open triangles (27 stars). The remaining 174 stars contained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

Perryman at el, A & A 331, 81 (1998)

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 depends 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.

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

- 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

$$C(x_i) = \arg\ \mathrm{min}_{\mathrm{k}} ||\mathrm{x}_{\mathrm{i}} - \mu_{\mathrm{k}}||$$

- 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
```

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 $L(\mathcal{Z}, A) = 5.3125$.

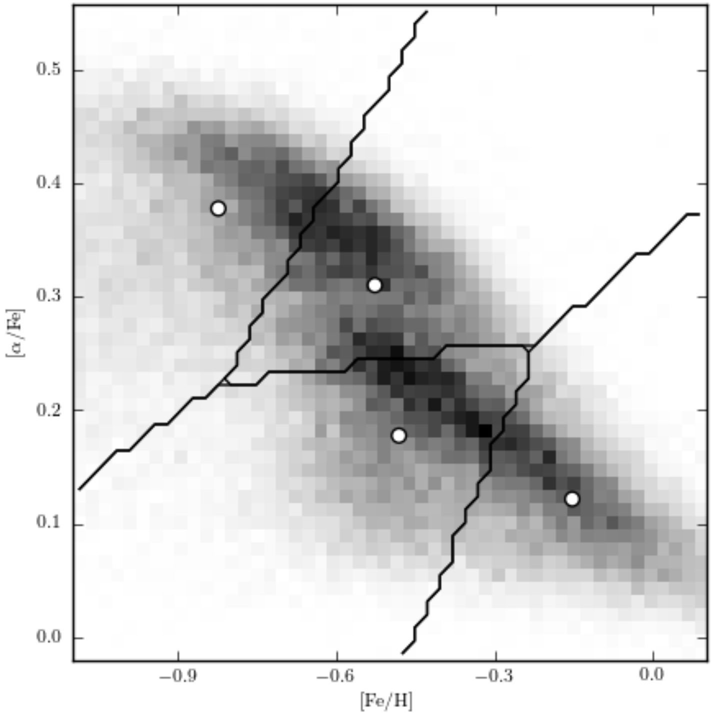
- Initialization: randomly pick $z_1 = 0.8$, $z_2 = 3.8$.

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.

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

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.

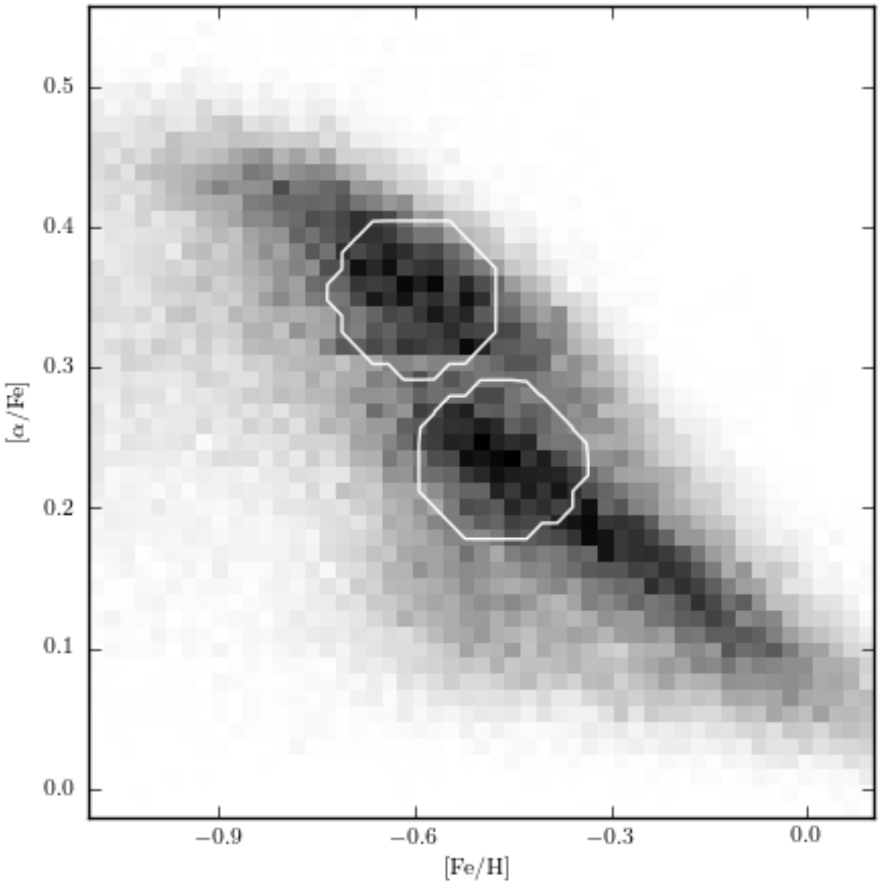
This so-called *mean-shift* algorithm is a technique to find local modes (bumps) in the density estimation of the data.

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



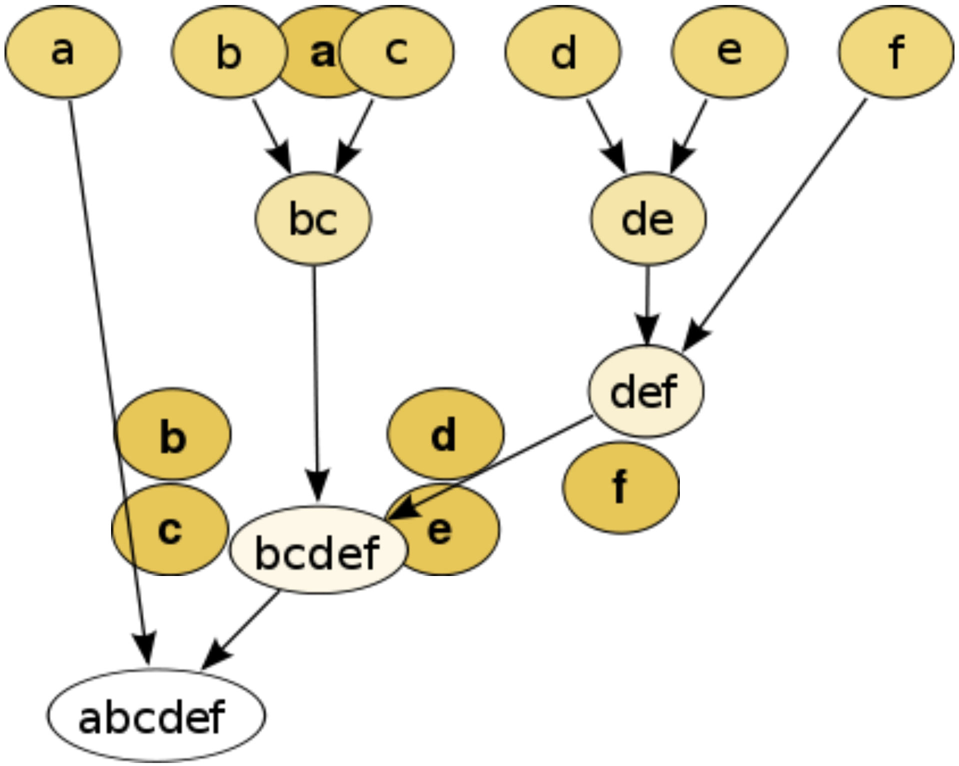
Mean-shift clustering on the metallicity datas et used in figures 6.6 and 6.13. The method finds two clusters associated with local maxima of the distribution (interior of the circles). Points outside the circles have been determined to lie in the background. The mean shift does not attempt to model 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

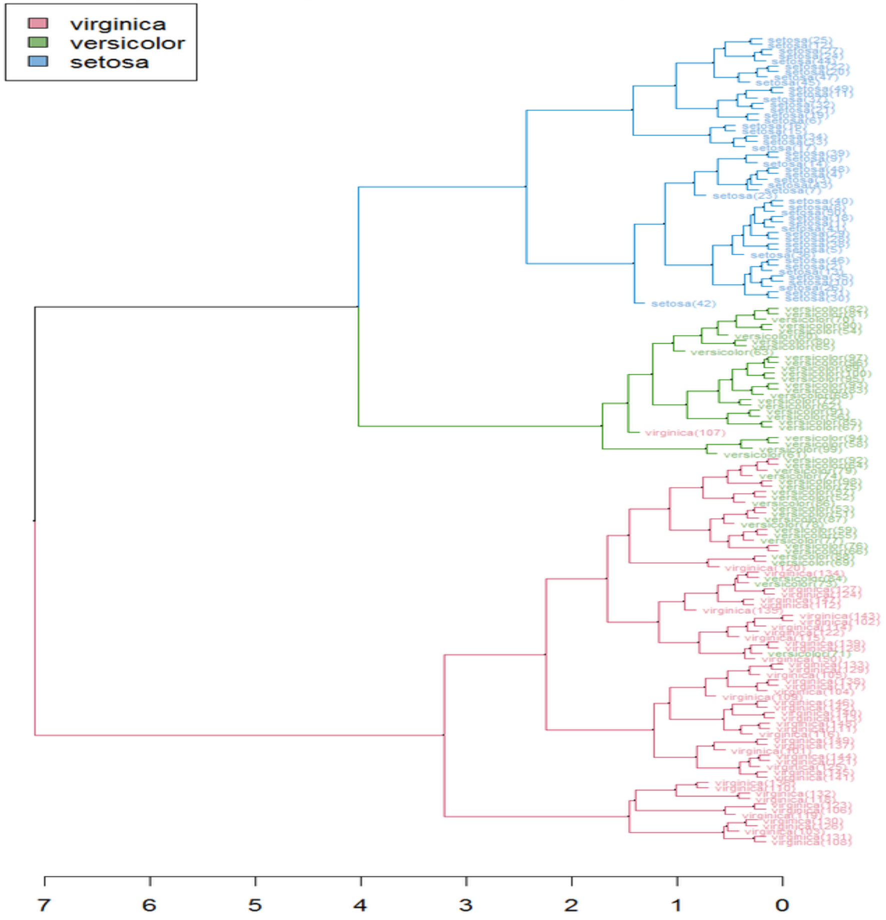
Agglomerative Clustering

- Generate clusters in a hierarchical way.
- Let the data set be $A = \{x_1, ..., x_n\}$.
- 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)



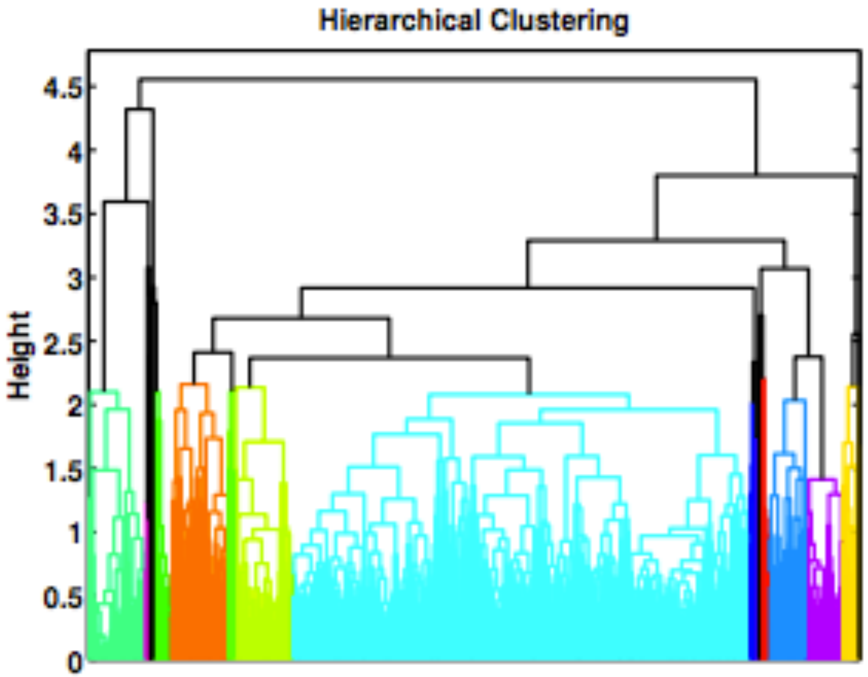


FIG. 2. Dendrogram showing hierarchical clustering of 1000 Omicron triggers from O1 Data provided by Livingston observatory. The transient morphology changes progressively from left to right

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

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

$$d_{min}(C_k, C_{k'}) = \min_{x \in C_k x' \in C'_k} ||x - x'||$$

(Minimum spanning tree)

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

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

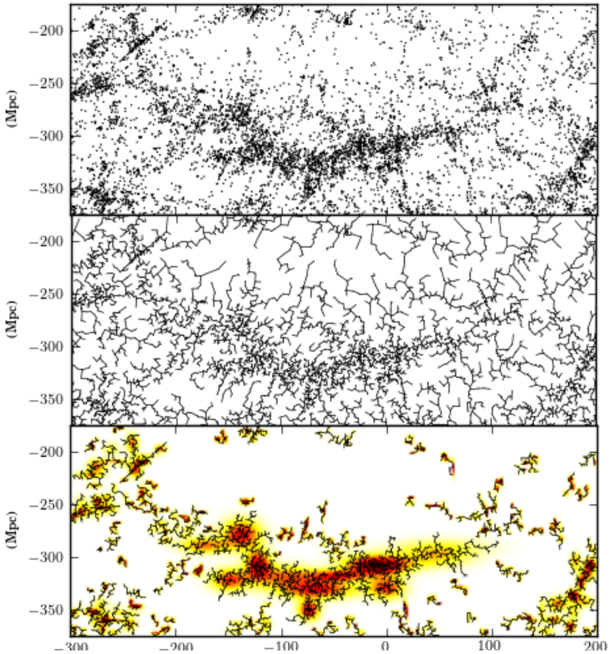
More details about agglomerative clustering in scikit-learn in

`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.

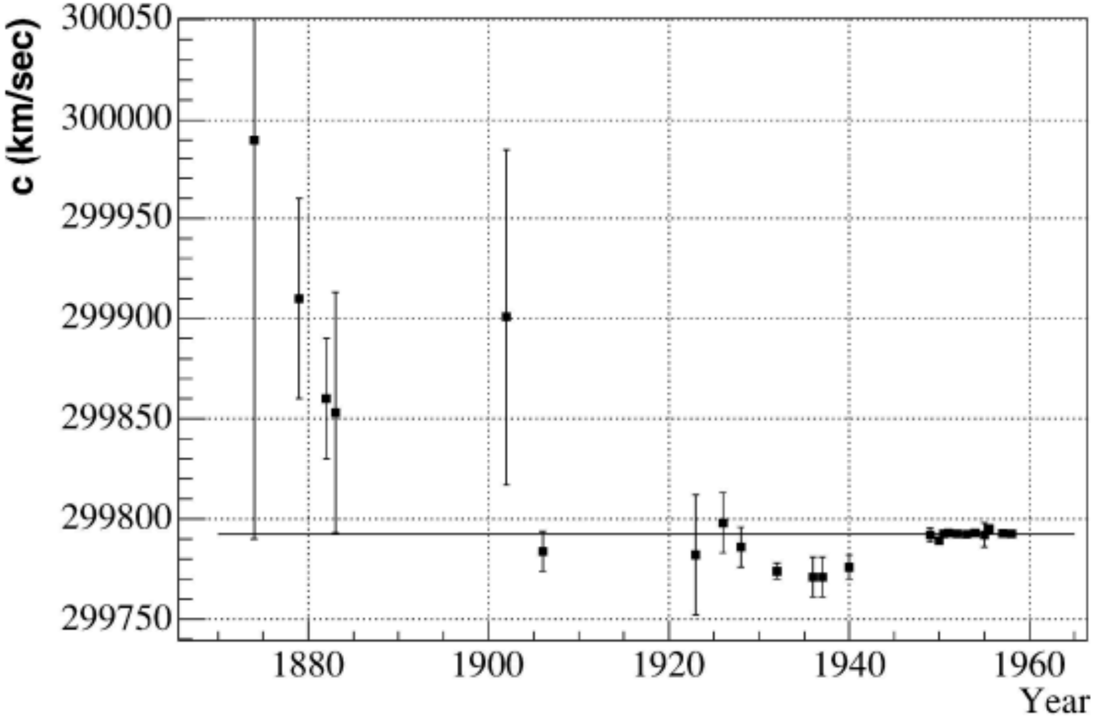
Single-linkage hierarchical clustering (also called friends of friends algorithm in Astronomy) refers to the distance between two clusters is determined by a single element pair, namely those two elements (one in each cluster) that are closest to each other (d_{min})

2-D dendogram



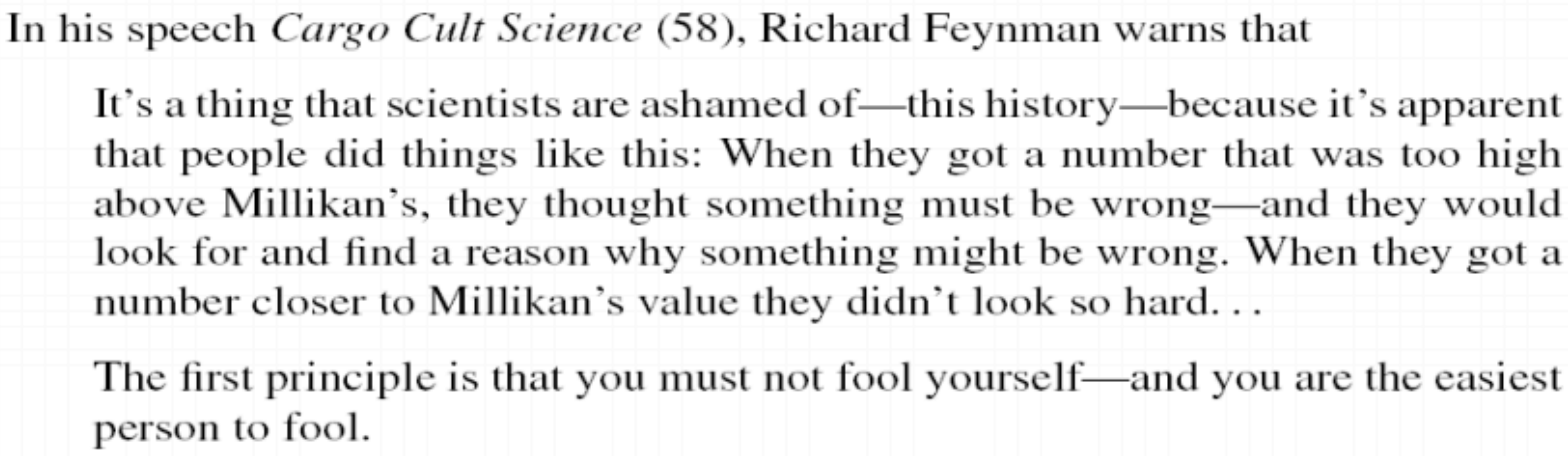
The three panels of this figure show a hierarchical clustering of a subset of galaxies from the Sloan Digital Sky Survey (SDSS). This region is known as the “SDSS Great Wall”, and contains an extended cluster of several thousand galaxies approximately 300Mpc (about 1 billion light years) from earth. 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.

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

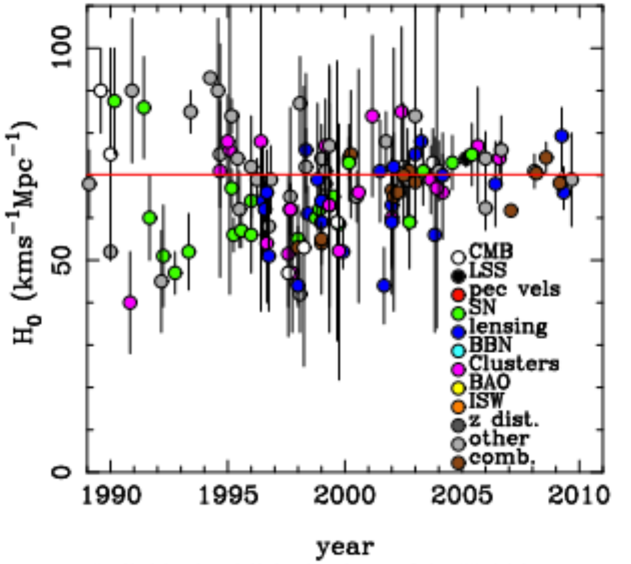


Figure 4. Individual published values of the Hubble constant, H_0 as a function of year. We show one sigma error bars on the points, and the point colour (shown in the legend) denotes the technique used to make the measurement (see Section 2.2 for more details).

arXiv:1112.3108

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)