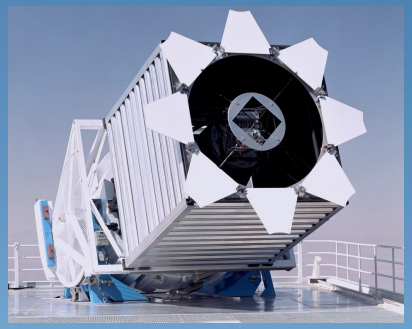
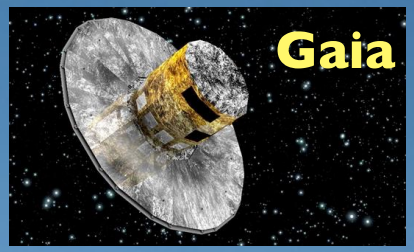
Week 9: Density estimation

Željko Ivezić and Mario Jurić, Department of Astronomy, UW

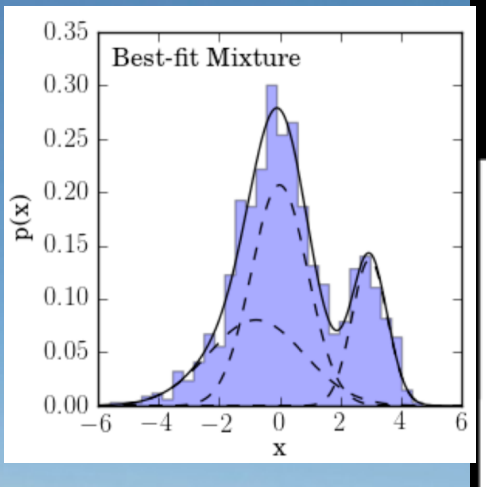
LSST

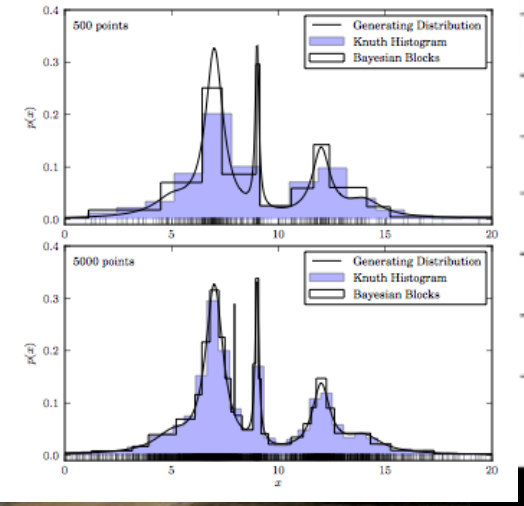


SDSS

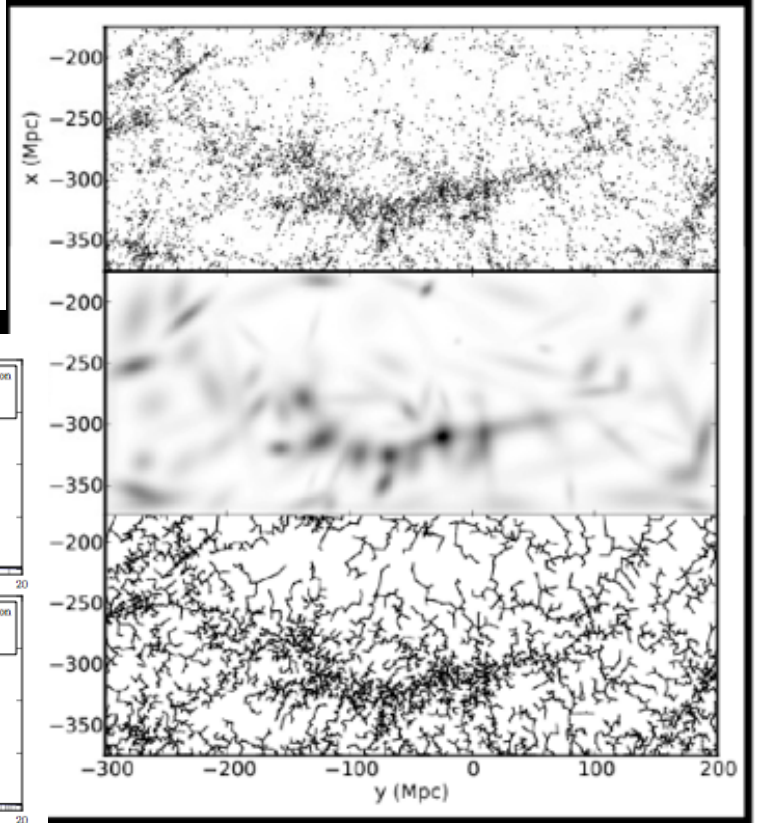


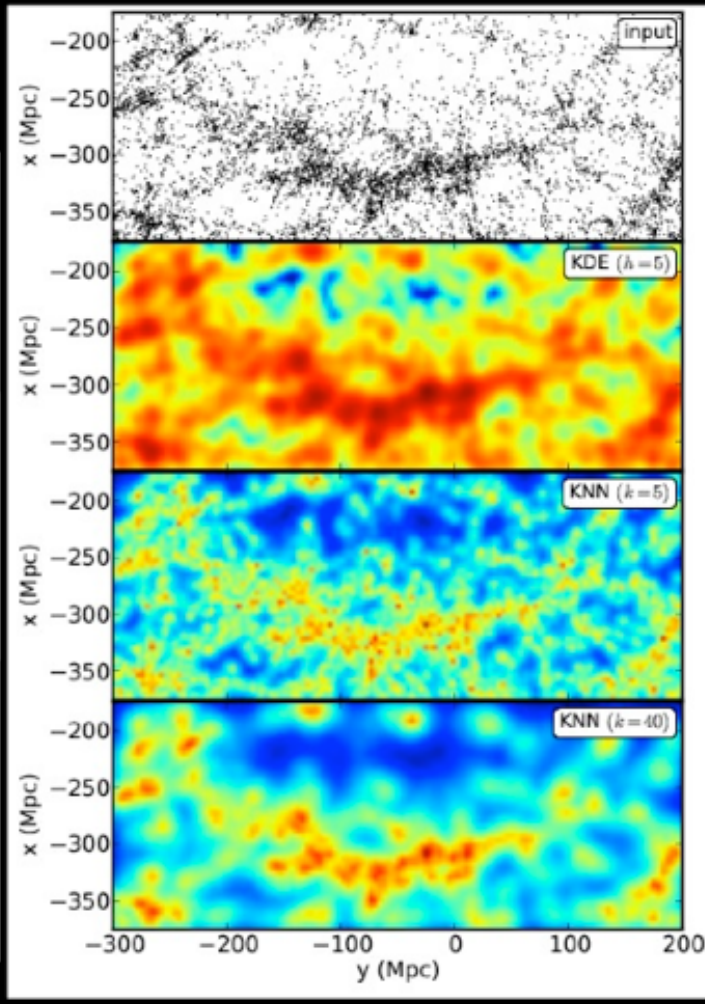
Density Estimation





Clustering and Density Estimation: SDSS Great Wall





Density Estimation

Very often we encounter problems where data are samples taken from a continuous distribution.

For example:

- The distribution of mass in the Galaxy can be considered as smooth and continuous at large scales; the stars we individually measure sample it.
  - Another way of looking at this is that the stars are drawn from an underlying PDF.
- The distribution of galaxies in the universe is similarly continuous at large scales
- We can also look at distributions in parameter spaces: e.g. the distribution of stars in (g-r, r) color-magnitude diagram.

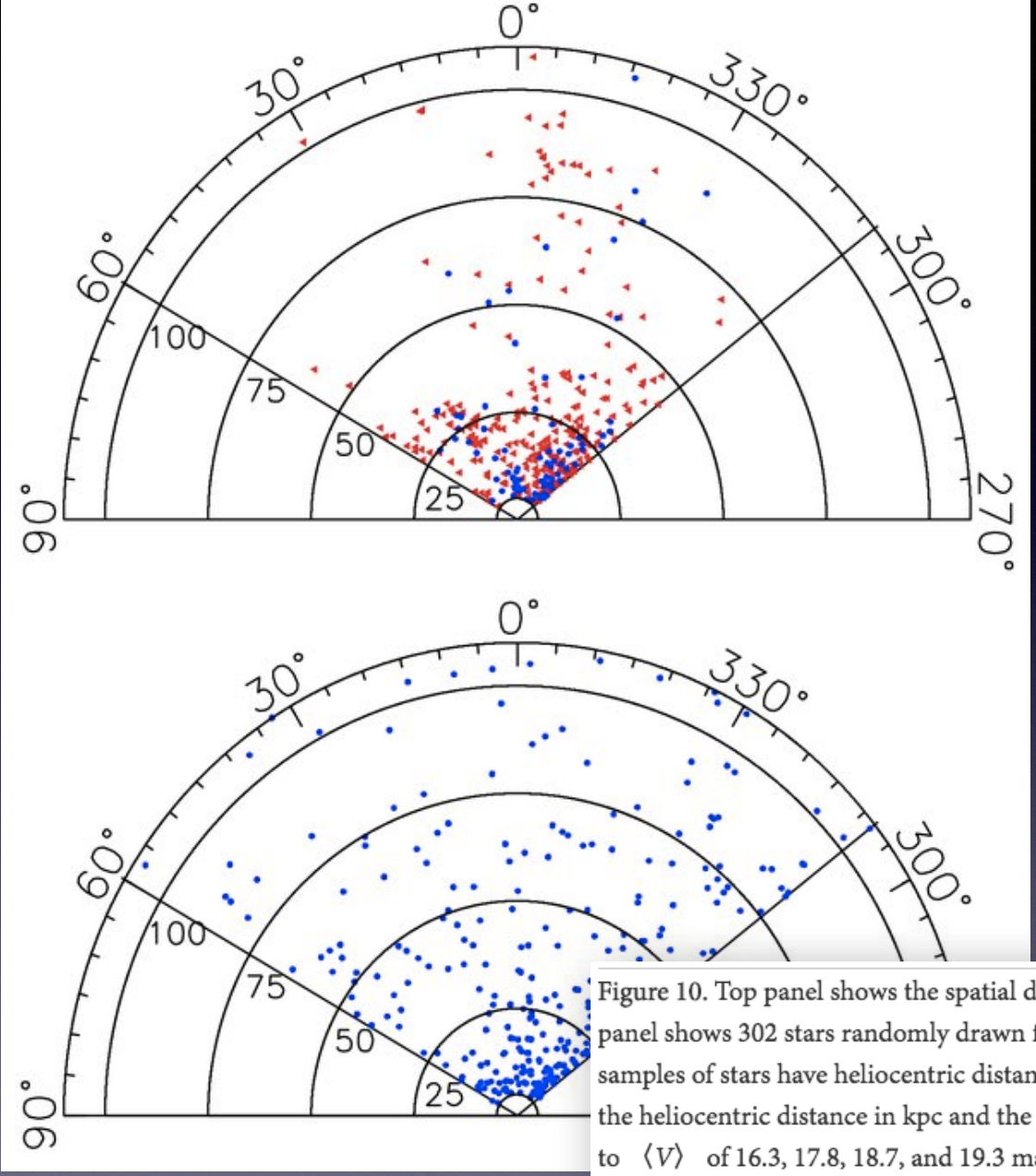


Figure 10. Top panel shows the spatial distribution of 366 RRab stars from stripe 82, while the bottom panel shows 302 stars randomly drawn from a smooth model distribution (Equation (16)). Both samples of stars have heliocentric distances between 5 and 120 kpc and $|\mathrm{decl.}| < 1.^{\circ}25$. The radial axis is the heliocentric distance in kpc and the angle is the equatorial right ascension. The circles correspond to $\langle V \rangle$ of 16.3, 17.8, 18.7, and 19.3 mag (corrected for ISM extinction). The RRab stars shown in the top panel are divided into Oosterhoff I (289 stars, red triangles) and Oosterhoff II (77 stars, blue dots) using the selection boundary shown in Figure 16.

Density Estimation

Very often we encounter problems where data are samples taken from a continuous distribution.

Given those samples, we'd like to infer the underlying smooth density (or the probability density function) from which they were drawn.

This is the problem of density estimation.

Very common in astronomy (and virtually all other fields).

Outline

Density estimation is the act of estimating a continuous density field from a discretely sampled set of points drawn from that density field.

- Non-parametric Density Estimation (today)
  - Histograms (Knuth's, Bayesian Blocks)
  - Kernel Density Estimation (KDE)
  - (Bayesian) nearest neighbors method
- Parametric Density Estimation (Thursday)
  - Gaussian Mixture Models
  - Extreme Deconvolution (XD)

What is a histogram?

- ---> Data modeled by a step function

Assuming that we have selected a bin size, Δ_b, the N values of x_i are sorted into M bins, with the count in each bin n_k, $k = 1, \ldots, M$. If we want to express the results as a properly normalized $f(x)$, with the values f_k in each bin, then it is customary to adopt

$$f_k = \frac{n_k}{\Delta_b N}. \tag{4.80}$$

The unit for f_k is the inverse of the unit for x_i.

Each estimate of f_k comes with some uncertainty. It is customary to assign "error bars" for each n_k equal to $\sqrt{n_k}$ and thus the uncertainty of f_k is

$$\sigma_k = \frac{\sqrt{n_k}}{\Delta_b N}. \tag{4.81}$$

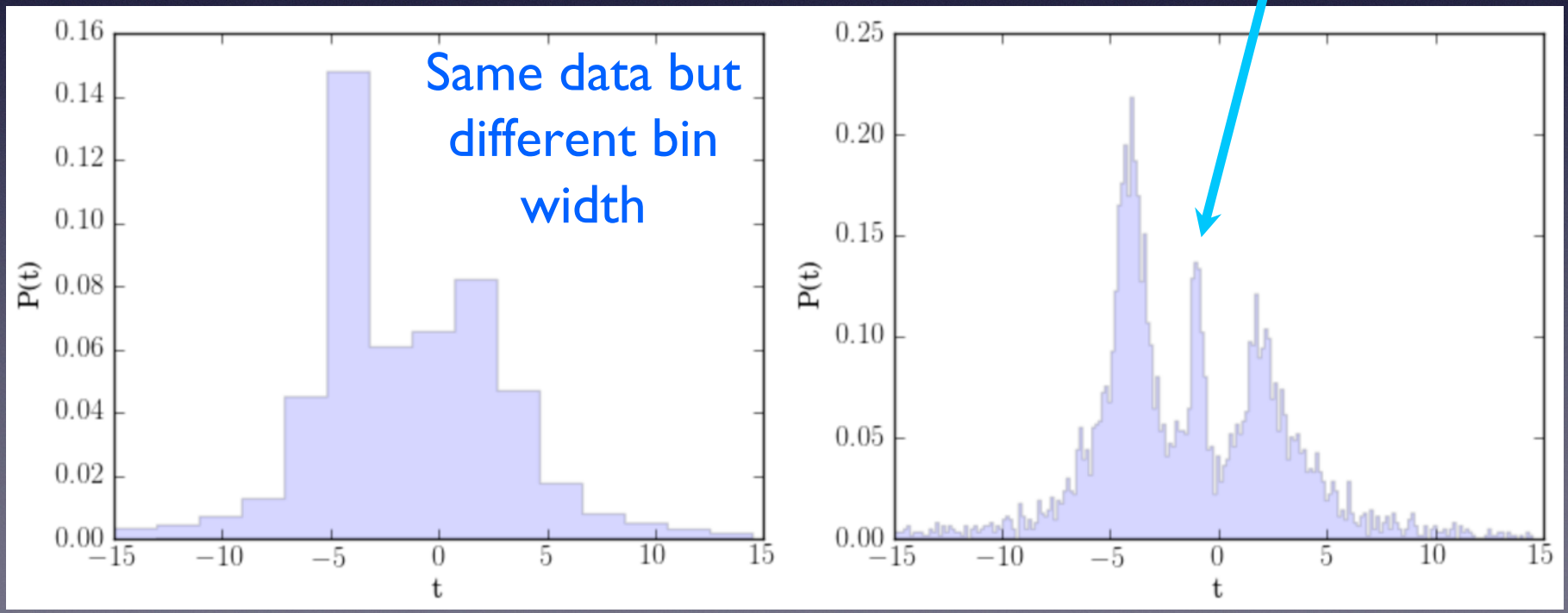
This practice assumes that n_k are scattered around the true values in each bin (μ) according to a Gaussian distribution, and that error bars enclose the 68% confidence range for the true value. However, when counts are low this assumption of Gaussianity breaks down and the Poisson distribution should be used instead. For example, according to the Gaussian distribution, negative values of μ have nonvanishing probability for small n_k (if $n_k = 1$, this probability is 16%). This is clearly wrong since in counting experiments, $\mu \geq 0$. Indeed, if $n_k \geq 1$, then even $\mu = 0$ is clearly ruled out. Note also that $n_k = 0$ does not necessarily imply that $\mu = 0$: even if $\mu = 1$, counts will be zero in $1/e \approx 37\%$ of cases. Another problem is that the range $n_k \pm \sigma_k$ does not correspond to the 68% confidence interval for true μ when n_k is small. These issues are important when fitting

What is a histogram?

- ---> Data modeled by a step function
- How do we determine/estimate/guess the bin width?
- Do all the bins have to have the same width?
- Do we really have to bin data to estimate model parameters?

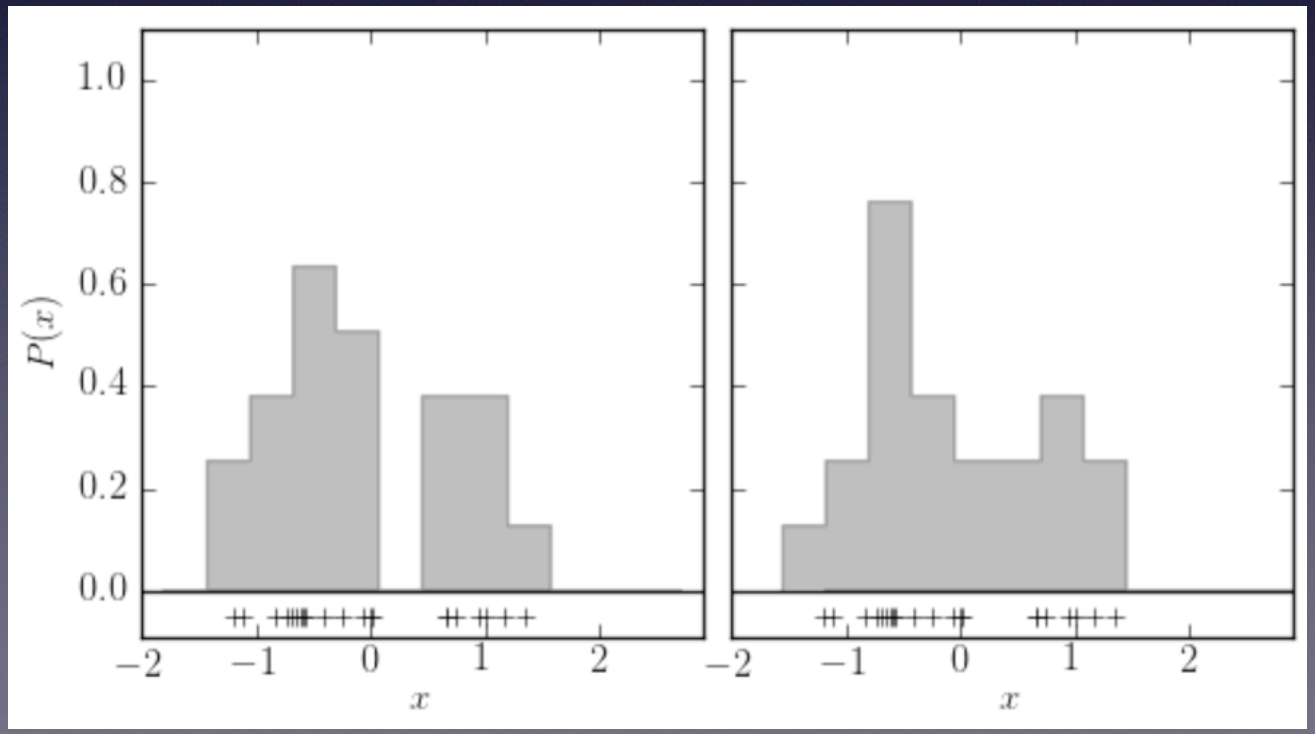
What is a histogram?

- ---> Data modeled by a step function
- How do we determine/estimate/guess the bin width?
- Do all the bins have to have the same width?
- Do we really have to bin data to estimate model parameters?



What is a histogram?

- ---> Data modeled by a step function
- How do we determine/estimate/guess the bin width?
- Do all the bins have to have the same width?
- Do we really have to bin data to estimate model parameters?



Despite the same bin width, a small offset in bin placement can give very different impressions about data behavior

Simple rules for estimating bin width: Scott's Rule and Freedman-Diaconis rule

Various proposed methods for choosing optimal bin width typically suggest a value proportional to some estimate of the distribution's scale, and decreasing with the sample size. The most popular choice is "Scott's rule" which prescribes a bin width

$$\Delta_b = \frac{3.5\sigma}{N^{1/3}}, \tag{4.78}$$

where σ is the sample standard deviation, and N is the sample size. This rule asymptotically minimizes the mean integrated square error (see eq. 4.14) and assumes that the underlying distribution is Gaussian; see [22]. An attempt to generalize this rule to non-Gaussian distributions is the Freedman–Diaconis rule,

$$\Delta_b = \frac{2(q_{75} - q_{25})}{N^{1/3}} = \frac{2.7\sigma_G}{N^{1/3}}, \tag{4.79}$$

which estimates the scale ("spread") of the distribution from its interquartile range (see [12]). In the case of a Gaussian distribution, Scott's bin width is 30% larger than the Freedman–Diaconis bin width. Some rules use the extremes of observed values to estimate the scale of the distribution, which is clearly inferior to using the interquartile range when outliers are present.

Although the Freedman–Diaconis rule attempts to account for non-Gaussian distributions, it is too simple to distinguish, for example, multimodal and unimodal distributions that have the same σ_G

Simple rules for estimating bin width: Scott's Rule and Freedman-Diaconis rule

The mean integrated square error (MISE), defined as

$$\mathrm{MISE} = \int_{-\infty}^{+\infty} [f(x) - h(x)]^2 \, dx, \tag{4.14}$$

is an often-used form of risk; it shows how "close" is our empirical estimate $f(x)$ to the true pdf $h(x)$. The MISE is based on a cost function given by the mean square error, also known as the L_2 norm. A cost function that minimizes absolute deviation

- Where do these (seemingly arbitrary) rules come from? The minimization of error (or, more generally, cost function).
  - I.e. "what does a *good fit* mean"?
- Scott's rule: minimizes the MSIE if the underlying distribution is Gaussian.
- Freedman-Diaconis rule: Minimize the difference in area between the empirical probability distribution (the histogram) and the theoretical (unknown) probability distribution.

Knuth's rule for estimating bin width

Knuth shows that the best piecewise constant model has the number of bins, M, which maximizes the following function (up to an additive constant, this is the logarithm of the posterior probability):

data

$$F(M|\{x_i\},I)) = N\,\log M + \log\left[\Gamma\left(\frac{M}{2}\right)\right] - M\log\left[\Gamma\left(\frac{1}{2}\right)\right] - \log\left[\Gamma\left(N+\frac{M}{2}\right)\right] + \sum_{k=1}^{M}\log\left[\Gamma\left(n_k+\frac{1}{2}\right)\right], \quad (5.107)$$

This is Bayesian model selection of M models

where Γ is the gamma function, and n_k is the number of measurements x_i, $i = 1,\ldots,N$, which are found in bin k, $k = 1,\ldots,M$. Although this expression is more involved than the "rules of thumb" listed in §4.8.1, it can be easily evaluated for an *arbitrary* data set.

Knuth derived eq. 5.107 using Bayesian model selection and treating the histogram as a piecewise constant model of the underlying density function. By assumption, the bin width is constant and the number of bins is the result of model selection. Given the number of bins, M, the model for the underlying pdf is

$$h(x) = \sum_{k=1}^{M} h_k \Pi(x|x_{k-1},x_k), \quad (5.108)$$

where the boxcar function $\Pi = 1$ if $x_{k-1} < x \le x_k$, and 0 otherwise. The M model parameters, h_k, $k = 1,\ldots,M$, are subject to normalization constraints, so that there are only $M-1$ free parameters. The uninformative prior distribution for $\{h_k\}$ is given by

$$p(\{h_k\}|M,I) = \frac{\Gamma(\frac{M}{2})}{\Gamma(\frac{1}{2})^M}\left[h_1\,h_2\ldots h_{M-1}\left(1-\sum_{k=1}^{M-1}h_k\right)\right]^{-1/2}, \quad (5.109)$$

which is known as the Jeffreys prior for the multinomial likelihood. The joint data likelihood is a multinomial distribution (see §3.3.3)

$$p(\{x_i\}|\{h_k\},M,I) \propto h_1^{n_1}\,h_2^{n_2}\ldots h_M^{n_M}. \quad (5.110)$$

Knuth's rule for estimating bin width

The posterior pdf for model parameters h_k is obtained by multiplying the prior and data likelihood. The posterior probability for the number of bins M is obtained by marginalizing the posterior pdf over all h_k. This marginalization includes a series of nested integrals over the $(M-1)$-dimensional parameter space, and yields eq. 5.107; details can be found in Knuth's paper.

Knuth also derived the posterior pdf for h_k, and summarized it by deriving its expectation value and variance. The expectation value is

Classic result →

$$h_k = \frac{n_k + \frac{1}{2}}{N + \frac{M}{2}}, \qquad (5.111)$$

h_k is not 0 even when n_k=0 !

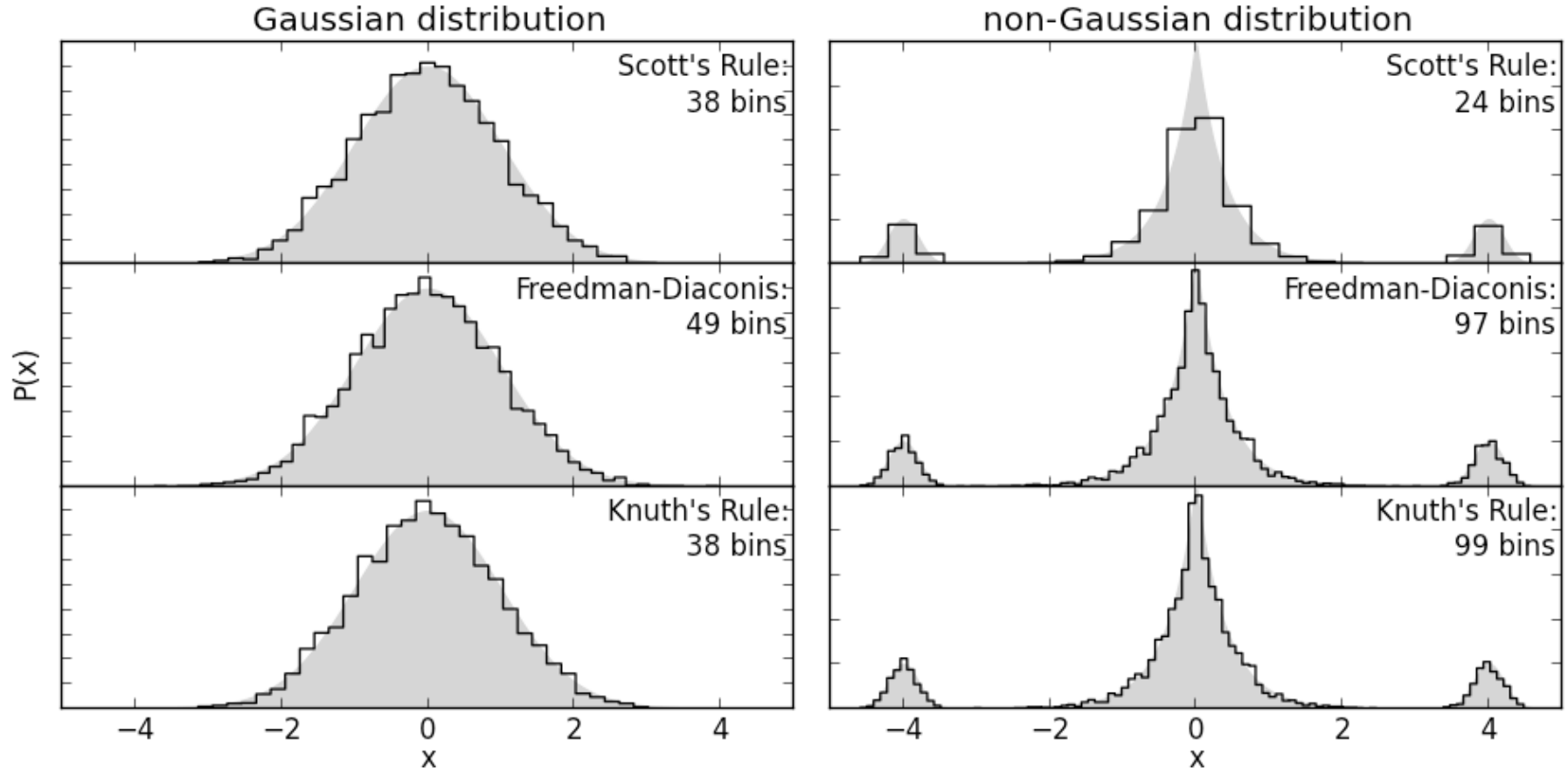
(the naive expectation is $h_k = n_k/N$)

The usefulness of Knuth's analysis and the result summarized by eq. 5.107 goes beyond finding the optimal bin size. The method is capable of recognizing substructure in data and, for example, it results in $M = 1$ when the data are consistent with a uniform distribution, and suggests more bins for a multimodal distribution than for a unimodal distribution even when both samples have the same size and σ_G (again, eq. 5.112 is an approximation valid only for unimodal centrally concentrated distributions; if in doubt, use eq. 5.107; for the latter, see the Python code used to generate figure 5.20).

Lastly, remember that Knuth's derivation assumed that the uncertainty of each x_i is negligible.

Comparing simple rules and Knuth's rule

- make this plot by running
  %run fig_hist_binsize.py



- the work horse code lives in
  $astroMLdir/astroML/density_estimation/hist_tools.py

Scargle's Bayesian Blocks algorithm

- Knuth's method assumes constant bin width!
- We can use the same Bayesian machinery to generalize Knuth's model to varying bin width (these are additional model parameters, just like the other ones)

In the Bayesian blocks formalism, the data are segmented into *blocks*, with the borders between two blocks being set by *changepoints*. Using a Bayesian analysis based on Poissonian statistics within each block, an objective function, called the log-likelihood fitness function, can be defined for each block:

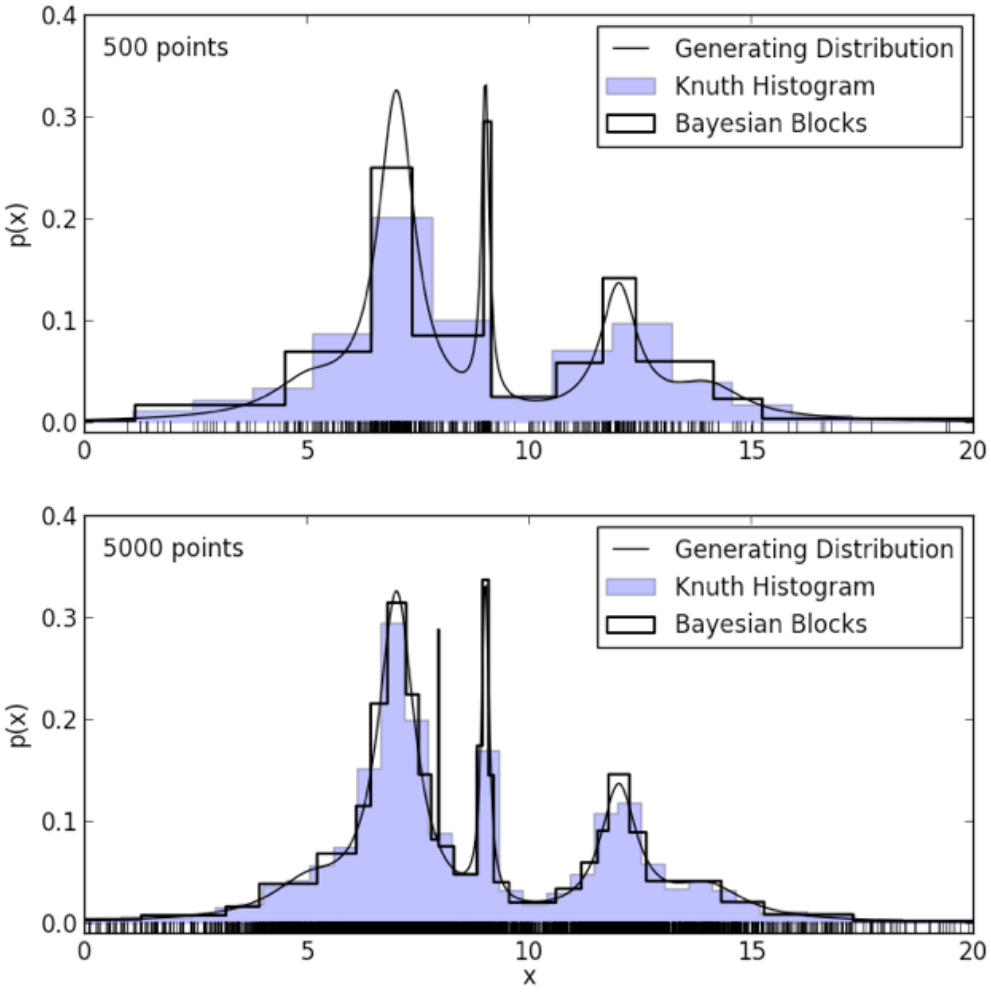
$$F(N_i, T_i) = N_i(\log N_i - \log T_i), \tag{5.113}$$

where N_i is the number of points in block i, and T_i is the width of block i (or the duration, in time-series analysis). Because of the additive nature of log-likelihoods, the fitness function for any set of blocks is simply the sum of the fitness functions for each individual block. This feature allows for the configuration space to be explored quickly using dynamic programming concepts: for more information see [31] or the Bayesian blocks implementation in AstroML.

[31] Scargle, J. D., J. P. Norris, B. Jackson, and J. Chiang (2012).
Studies in astronomical time series analysis. VI.
Bayesian block representations. ArXiv:astro-ph/1207.5578.

Comparing Knuth's rule and Bayesian Blocks

- make this plot by running
  %run fig_bayes_blocks.py

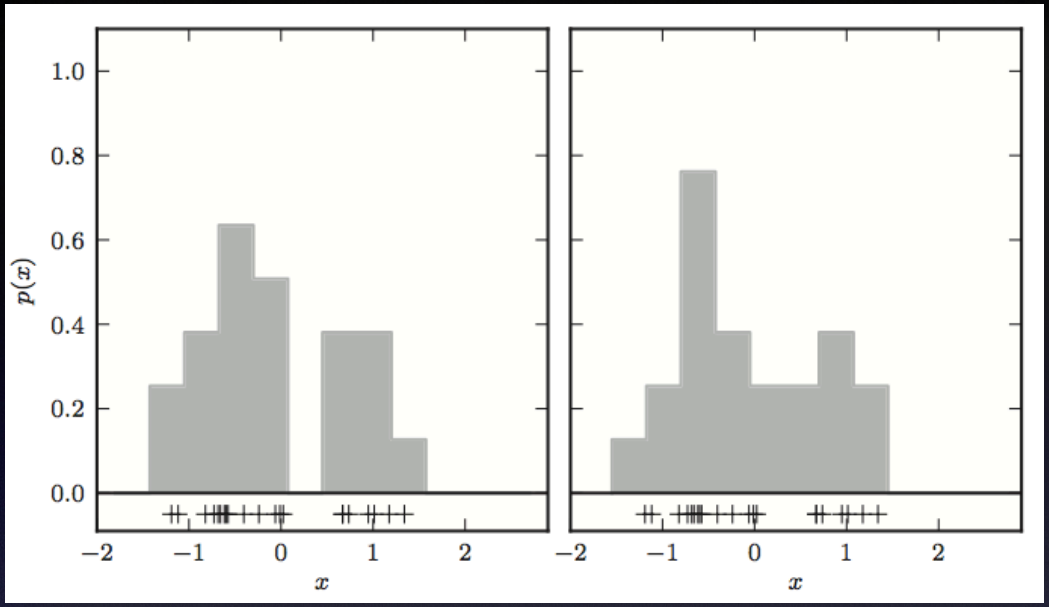


Note that Knuth's method does not find the narrow peak in the middle for the smaller dataset!

Bayesian Blocks method gives you the best step function that describes your data. It is excellent for low-count data and for time-series analysis!

But remember that data errors are not included!!!

Beyond Histograms: Kernel Density Estimation



Histograms are sensitive to (potentially arbitrary) choices made by the researcher.

E.g., the example of the left shows the same sample binned to a histogram with bins of the same width, but with bins offset by 0.25 in the x direction.

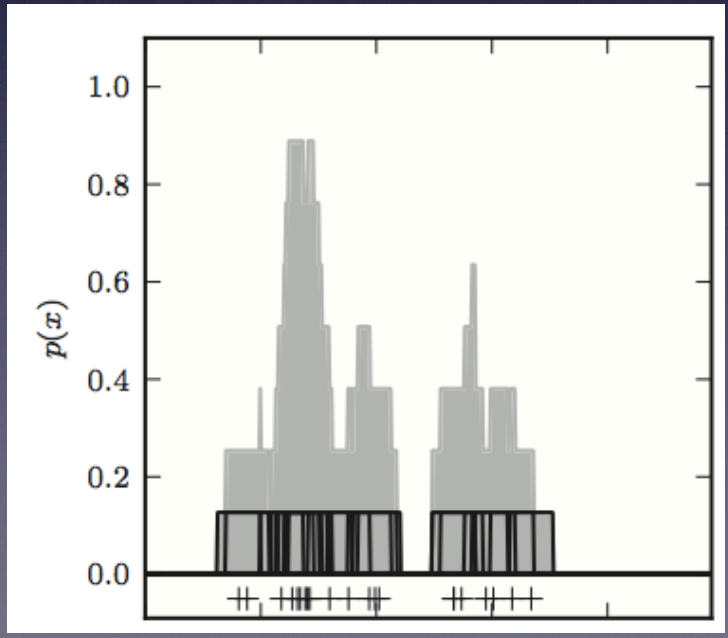
The binning to the left results in a distribution that appears bimodal; the binning to the right appears ~flat + a peak.

Can we do better?

The Big Idea: let's allow each point to have a bin of its own, and allow the bins to overlap.

I.e., replace each point with a "boxcar" function.

The resulting histogram preserves the bimodal structure of the original data much better.



the underlying pdf is

$$h(x) = \sum_{k=1}^{M} h_k \Pi(x|x_{k-1}, x_k),$$

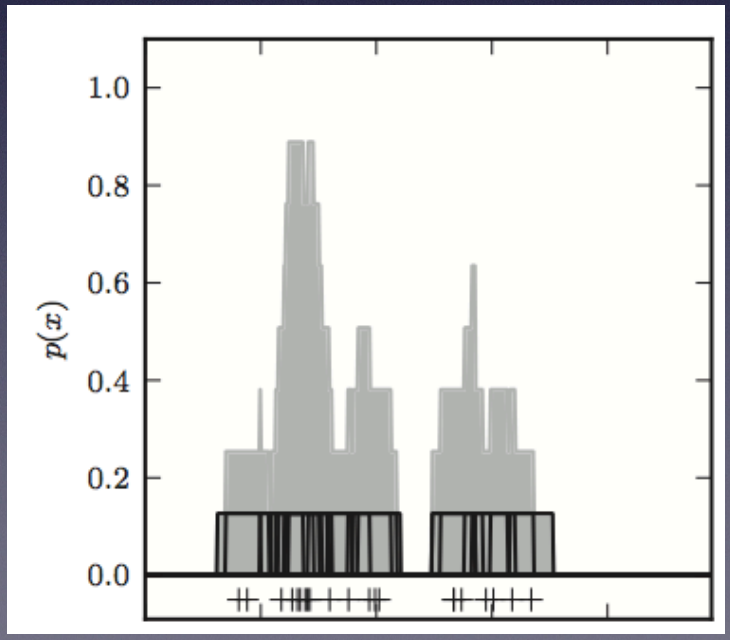
where the boxcar function $\Pi = 1$ if $x_{k-1} < x \leq x_k$, and 0 otherwise.

Can we do better?

The Big Idea: let's allow each point to have a bin of its own, and allow the bins to overlap.

I.e., replace each point with a "boxcar" function.

The resulting histogram preserves the bimodal structure of the original data much better.

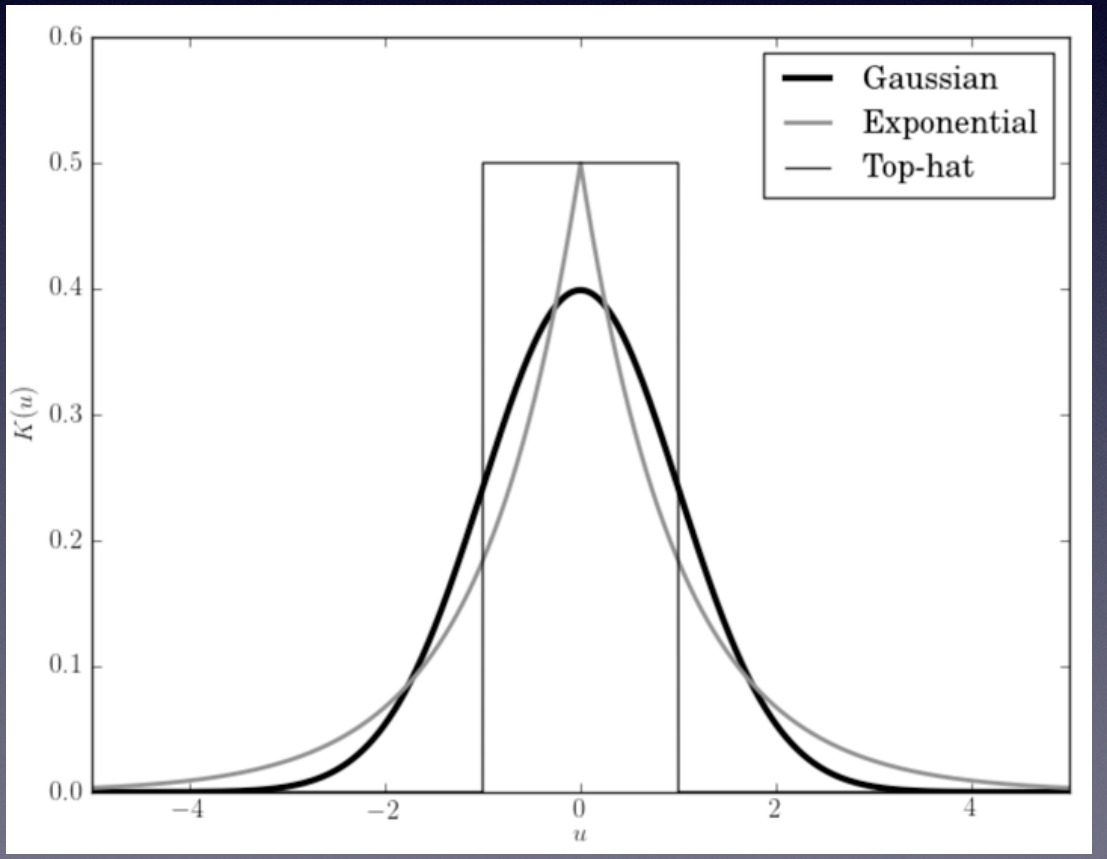


Kernel Density Estimates (KDE)

When computing histogram, we replace each object by the boxcar (top-hat) function. This function is called **kernel**.

But of course we can use any other function (well, non-negative, normalized to 1, zero-mean, finite variance).

<u>This is the main idea of the KDE</u>; it also works in high-D spaces.



Kernel Density Estimates (KDE)

Given a set of measurements $\{x_i\}$, the kernel density estimator (i.e., an estimator of the underlying pdf) at an arbitrary position x is defined as

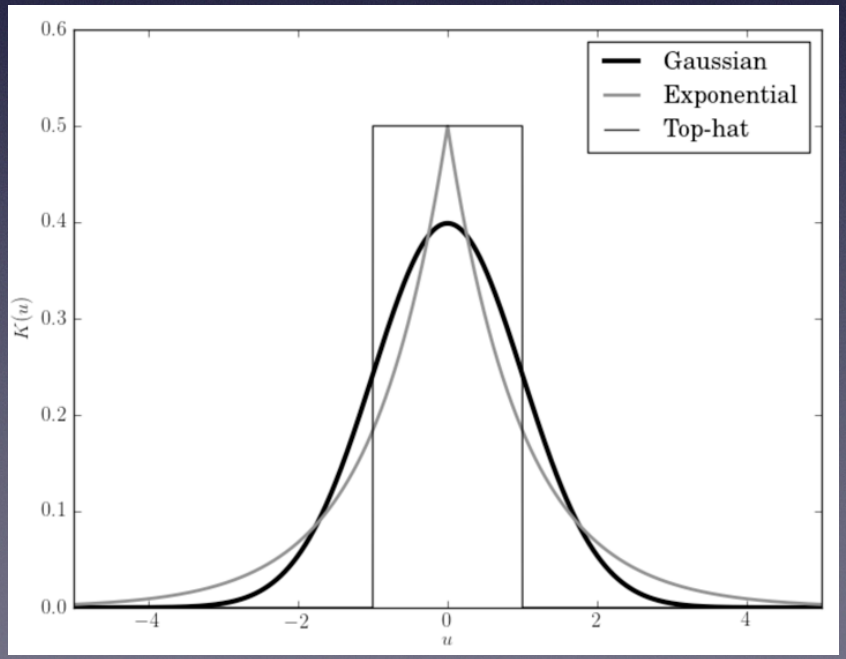
$$\widehat{f}_N(x) = \frac{1}{Nh^D} \sum_{i=1}^{N} K\left(\frac{d(x, x_i)}{h}\right), \qquad (6.1)$$

where $K(u)$ is the *kernel function* and h is known as the bandwidth (which defines the size of the kernel). The local density is estimated as a weighted mean of all points, where the weights are specified via $K(u)$ and typically decrease with distance $d(x, x_i)$.

This way, we get an estimator $f_N(x)$ of the underlying smooth PDF (or density distribution).

K: The ***kernel function***.

h: The ***bandwidth*** (defines the size of the kernel)



Kernel Density Estimates (KDE)

Given a set of measurements $\{x_i\}$, the kernel density estimator (i.e., an estimator of the underlying pdf) at an arbitrary position x is defined as

$$\hat{f}_N(x) = \frac{1}{Nh^D} \sum_{i=1}^{N} K\left(\frac{d(x, x_i)}{h}\right), \tag{6.1}$$

where $K(u)$ is the *kernel function* and h is known as the bandwidth (which defines the size of the kernel). The local density is estimated as a weighted mean of all points, where the weights are specified via $K(u)$ and typically decrease with distance $d(x, x_i)$.

Gaussian

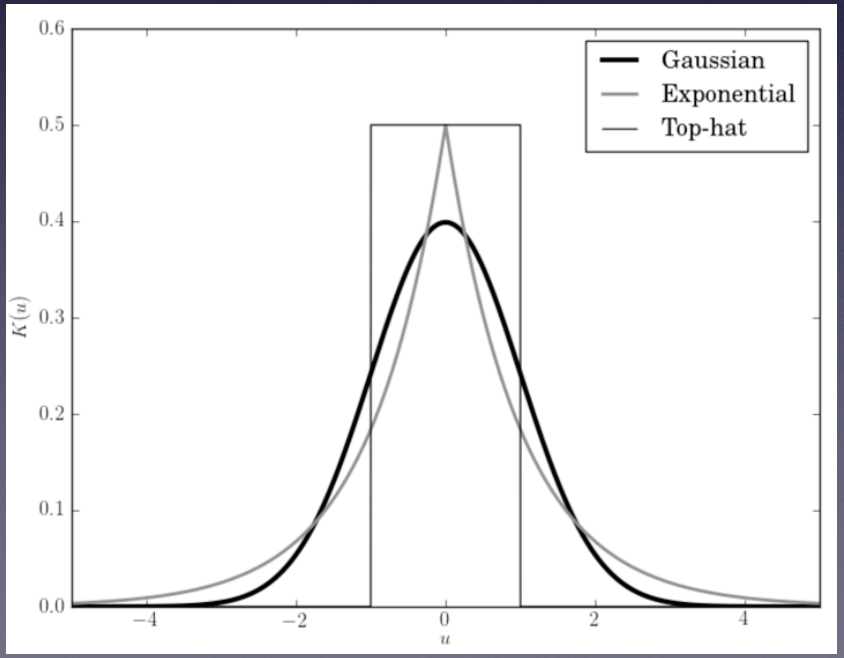
$$K(u) = \frac{1}{(2\pi)^{D/2}} e^{-u^2/2},$$

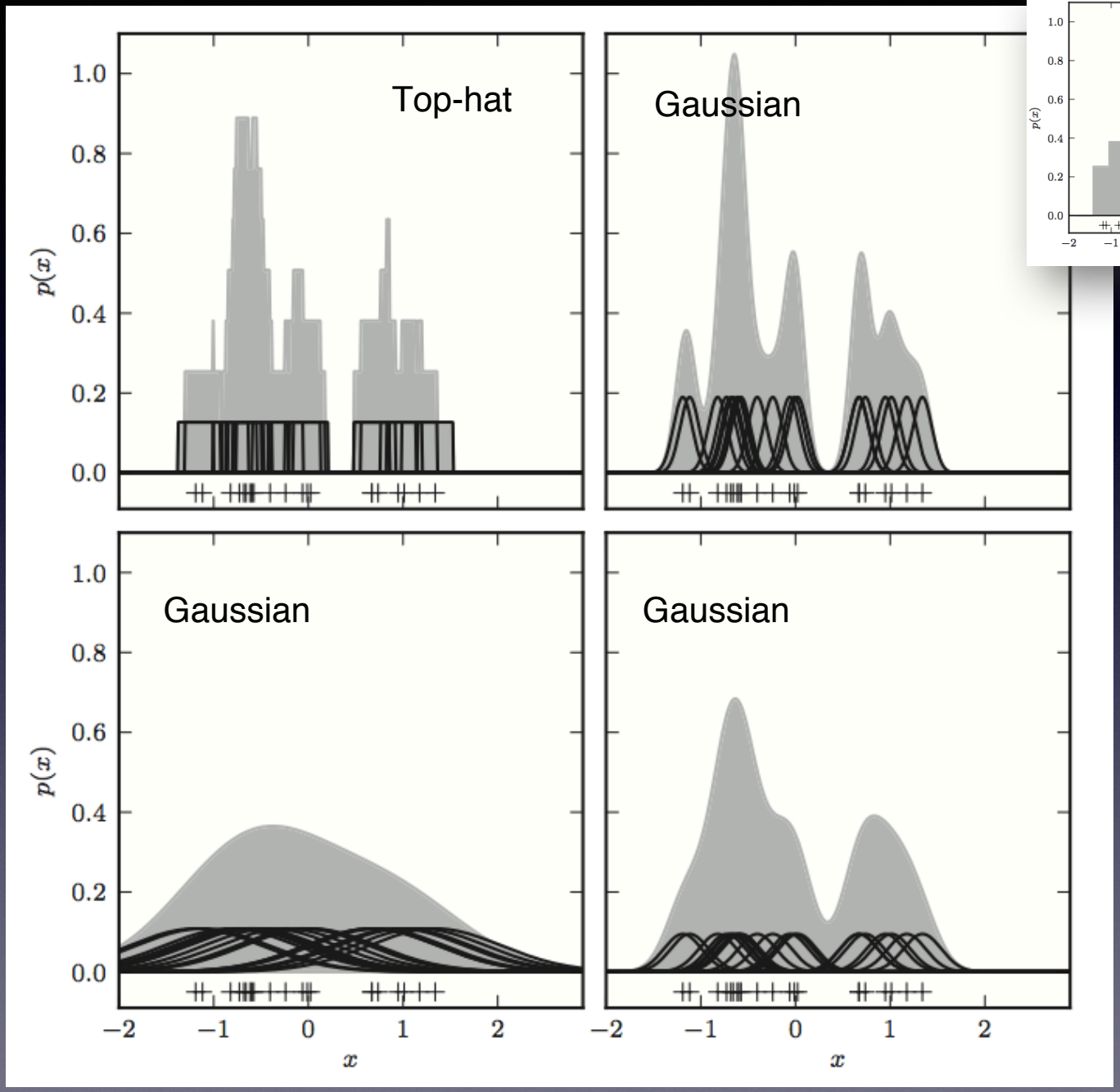
Top-hat

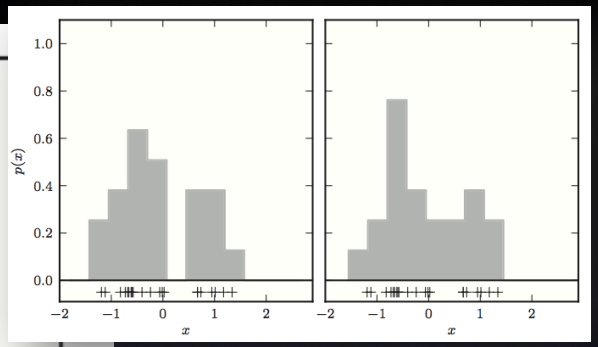
$$K(u) = \begin{cases} \frac{1}{V_D(1)} & \text{if } u \le 1, \\ 0 & \text{if } u > 1, \end{cases}$$

Exponential

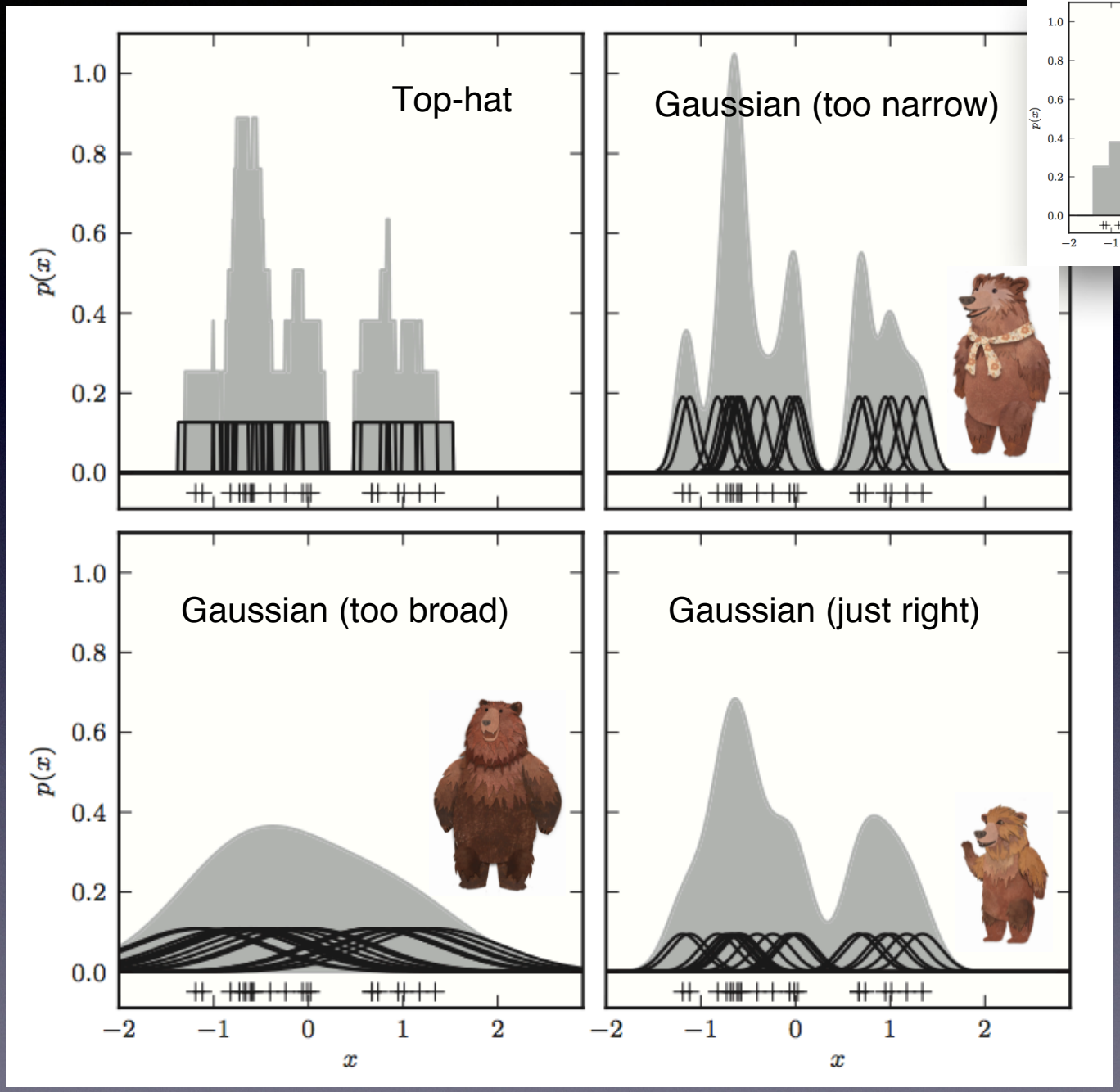
$$K(u) = \frac{1}{D!\, V_D(1)} e^{-|u|},$$

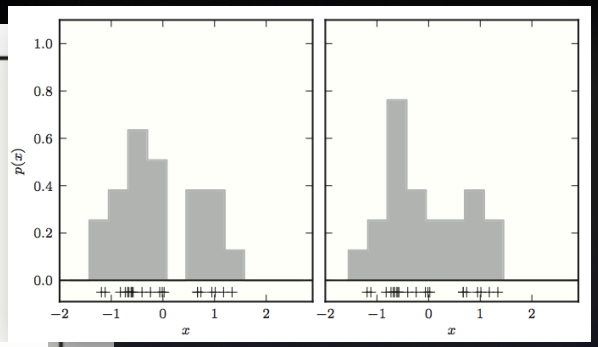






Bandwidth is a free parameter.

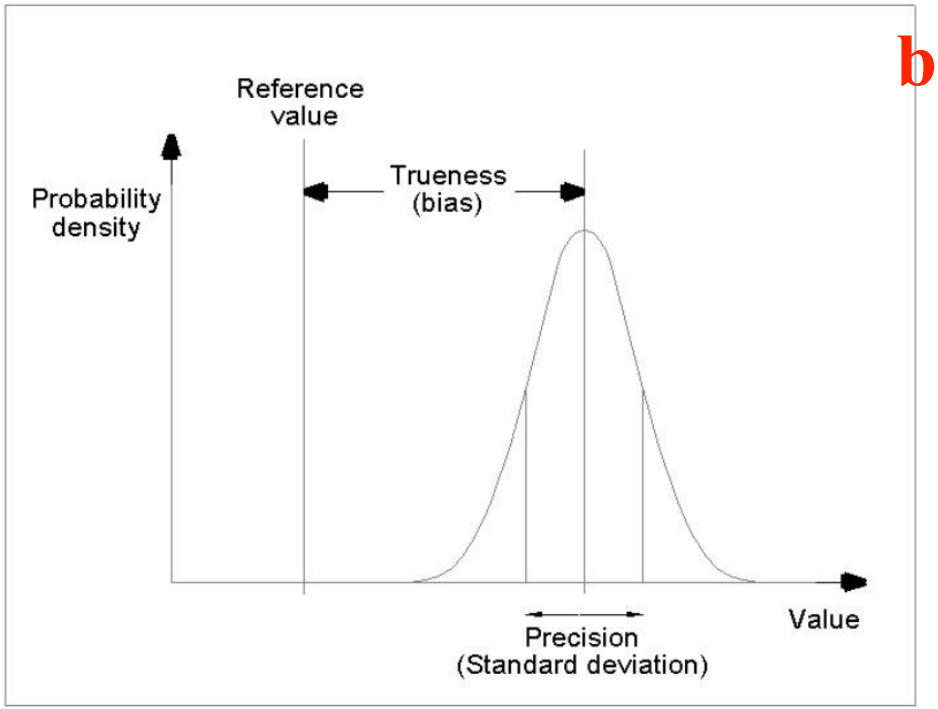


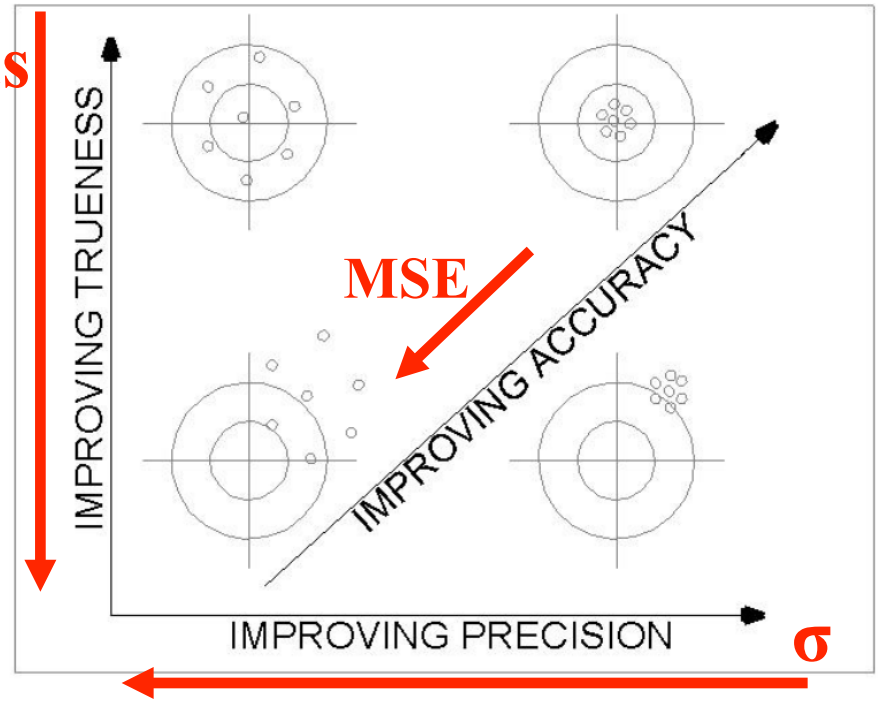


Bandwidth is a free parameter.

Refresher: Bias & Variance

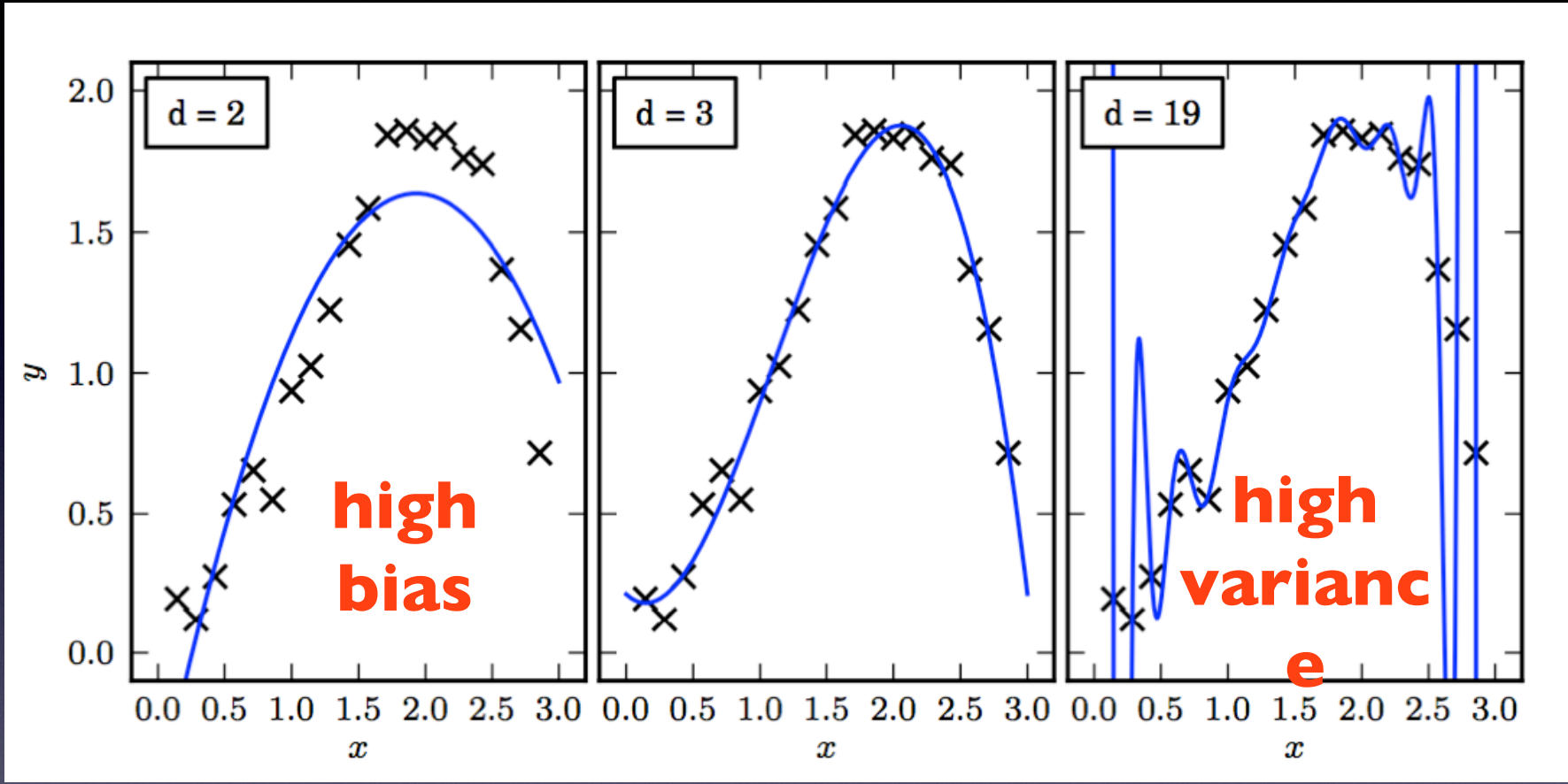
Let's assume that we have N measurements x_i, and that for each measurement we know the corresponding error distribution, that is, the expected distribution of x_i around the true value μ (which we want to estimate)





Mean Squared Error: $\mathrm{MSE} = V + \mathrm{bias}^2$ (V=variance=σ^2)

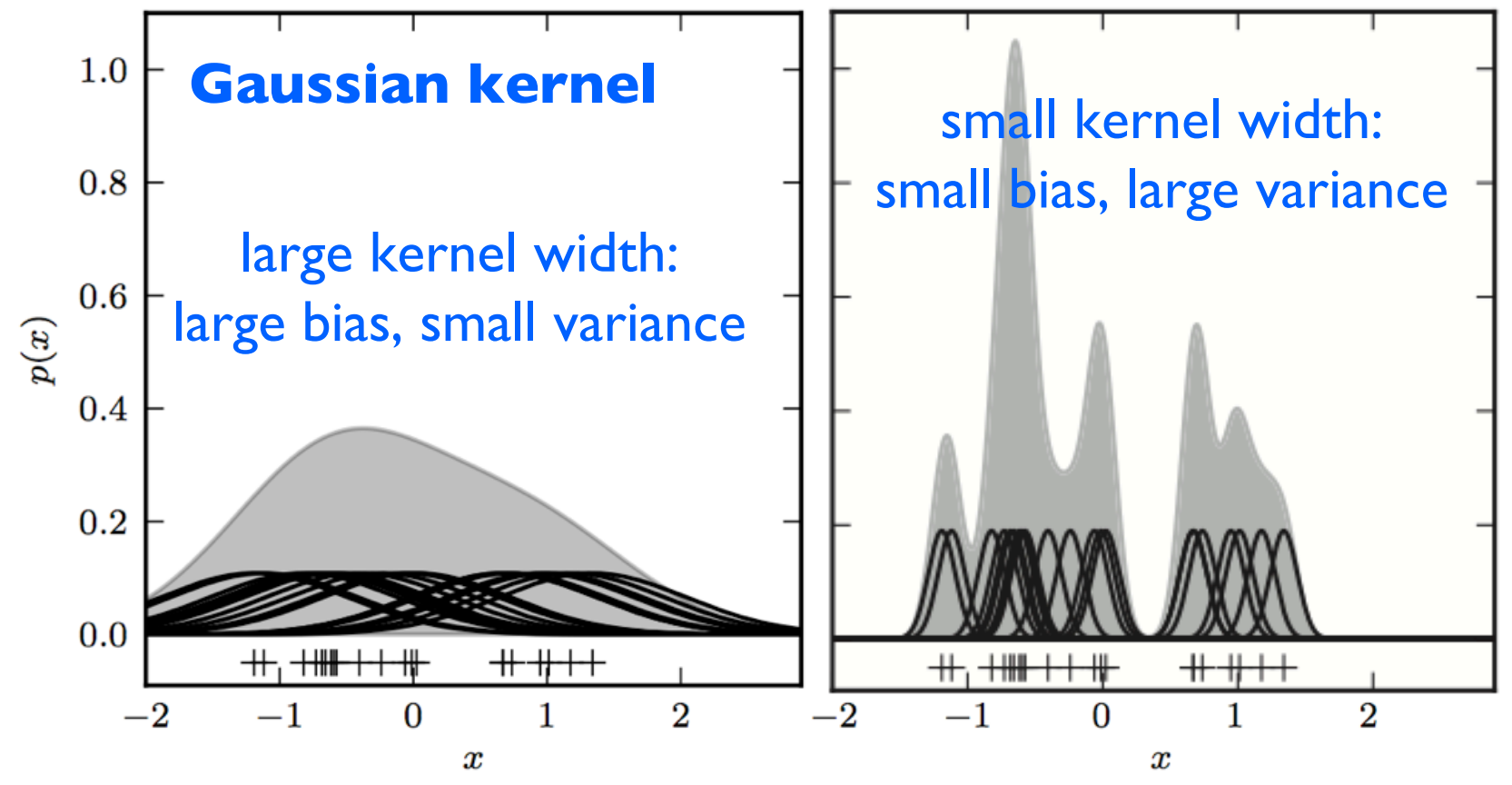
Intuitively: Bias vs. variance tradeoff



high bias: “can’t fit” data points

high variance: “oscillations” between data points

Kernel Density Estimates (KDE)



Kernel Density Estimates (KDE)

So what's the algorithm to choose the kernel width?
Depends on what you wish to optimize.

a) The likelihood of the estimated distribution generating a left-out point (using *cross-validation*):

Cross-validation can be used for any cost function (see §8.11); we just have to be able to evaluate the cost on *out-of-sample* data (i.e., points not in the training set). If we consider the likelihood cost for KDE, for which we have the leave-one-out *likelihood cross-validation*, then the cost is simply the sum over all points in the data set (i.e., $i = 1, \ldots, N$) of the log of the likelihood of the density, where the density, $\widehat{f}_{h,-i}(x_i)$, is estimated leaving out the ith data point. This can be written as

$$\mathrm{CV}_l(h) = \frac{1}{N} \sum_{i=1}^{N} \log \widehat{f}_{h,-i}(x_i), \tag{6.5}$$

We minimize the above as a function of bandwidth, to derive the optimal *h*.

Kernel Density Estimates (KDE)

So what's the algorithm to choose the kernel width?
Depends on what you wish to optimize.

b) Mean Integrated Square Error:

An alternative to likelihood cross-validation is to use the mean integrated square error (MISE), introduced in eq. 4.14, as the cost function. To determine the value of h that minimizes the MISE we can write

$$\int (\hat{f}_h - f)^2 = \int \hat{f}_h^2 - 2\int \hat{f}_h f + \int f^2. \qquad (6.6)$$

This motivates the L_2 *cross-validation* score:

$$\mathrm{CV}_{L_2}(h) = \int \hat{f}_h^2 - 2\frac{1}{N}\sum_{i=1}^{N} \hat{f}_{h,-i}(x_i) \qquad (6.8)$$

Kernel Density Estimates (KDE)

How do we choose the ***kernel***?

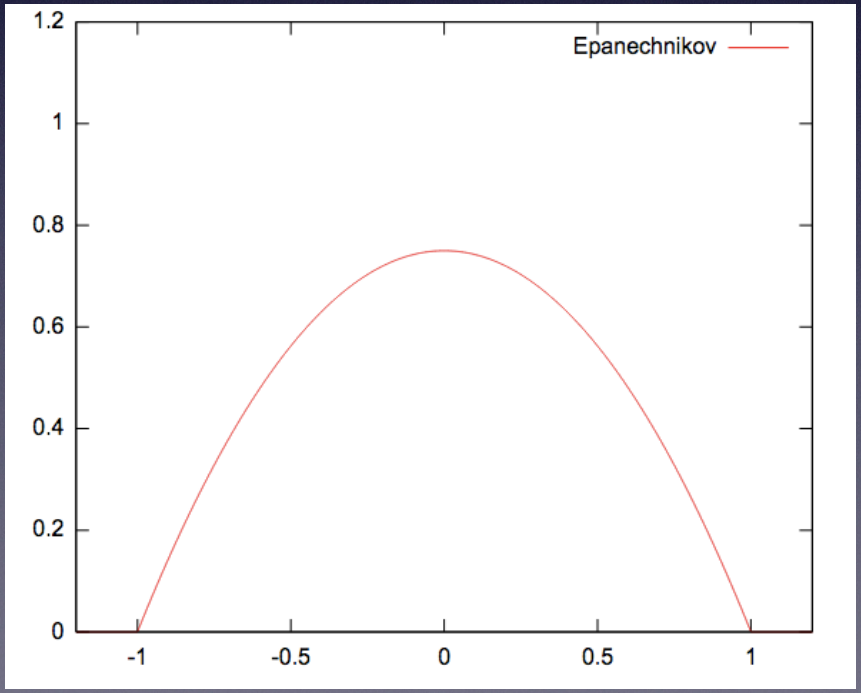
Ideally we would select a kernel that has h as small as possible. If h becomes too small we increase the variance of the density estimation. If h is too large then the variance decreases but at the expense of the bias in the derived density. The optimal kernel function, in terms of minimum variance, turns out to be

$$K(x) = \frac{3}{4}\left(1 - x^2\right) \tag{6.9}$$

for $|x| \leq 1$ and 0 otherwise; see [37]. This function is called the *Epanechnikov kernel.*

Formally, Epanechinkov kernel is the optimal choice if we're trying to minimize the MISE.

That said, the effectiveness of KDE is not very sensitive to the choice of a kernel!



Kernel	Efficiency
Epanechnikov	1.000
Biweight	0.994
Triangular	0.986
Normal	0.951
Uniform	0.930

That said, the effectiveness of KDE is not very sensitive to the choice of a kernel!

Given this, the Gaussian (Normal) kernel is a frequent practical choice: decent efficiency, yet continuous in all derivatives.

Efficiency is defined as $\sqrt{\int u^2 K(u)\,du \int K(u)^2\,du}.$

Kernel Density Estimates (KDE): Convergence

The optimal KDE bandwidth decreases at the rate $\mathcal{O}(N^{-1/5})$ (in a one-dimensional problem), and the error of the KDE using the optimal bandwidth converges at the rate $\mathcal{O}(N^{-4/5})$; it can be shown that histograms converge at a rate $\mathcal{O}(N^{-2/3})$; see [35]. KDE is, therefore, theoretically superior to the histogram as an estimator of the density. It can also be shown that there does not exist a density estimator that converges faster than $\mathcal{O}(N^{-4/5})$ (see Wass10).

astroML implementation is very easy to use:

AstroML contains an implementation of kernel density estimation in D dimensions using the above kernels:

```
import numpy as np
from astroML.density_estimation import KDE

X = np.random.normal(size=(1000, 2))  # 1000 points in 2 dims
kde = KDE('gaussian', h=0.1)  # select the gaussian kernel
kde.fit(X)  # fit the model to the data
dens = kde.eval(X)  # evaluate the model at the data
```

Kernel Density Estimates (KDE)

What to do when data have uncertainties?

Suppose now that the points (i.e., their coordinates) are measured with some error σ. We begin with the simple one-dimensional case with homoscedastic errors. Assume that the data is drawn from the true pdf $h(x)$, and the error is described by the distribution $g(x|\sigma)$. Then the observed distribution $f(x)$ is given by the convolution (see §3.44)

$$f(x) = (h \star g)(x) = \int_{-\infty}^{\infty} h(x')g(x - x')\,dx'. \tag{6.10}$$

This suggests that in order to obtain the underlying noise-free density $h(x)$, we can obtain an estimate $f(x)$ from the noisy data first, and then "deconvolve" the noise pdf. The nonparametric method of *deconvolution KDE* does precisely this; see [10, 38]. According to the convolution theorem, a convolution in real space corresponds to a product in Fourier space (see §10.2.2 for details). Because of this, deconvolution KDE can be computed using the following steps:

Kernel Density Estimates (KDE)

What to do when data have uncertainties?

Deconvolution KDE (Stefanski & Raymond 1990):

1. Find the kernel density estimate of the observed data, $f(x)$, and compute the Fourier transform $F(k)$.

2. Compute the Fourier transform $G(k)$ of the noise distribution $g(x)$.

3. From eq. 6.10 and the convolution theorem, the Fourier transform of the true distribution $h(x)$ is given by $H(k) = F(k)/G(k)$. The underlying noise-free pdf $h(x)$ can be computed via the inverse Fourier transform of $H(k)$.

Nearest Neighbor Density Estimation

(K-)NN Density Estimation

The Big Idea:

If we're trying to estimate the underlying probability density distribution out of which a point in a sample was drawn, the number of neighbors within some radius around it, or the distance to some Kth nearest neighbor, should allow us to estimate the local density.

Intuitively:

- The number of nearest neighbors in 100ft radius of an apartment in the U District is much larger than for a house in Cedar Falls, WA
- The distance to your 10^{th} nearest neighbor in the U District will be much smaller than to the 10^{th} neighbor in Cedar Falls, WA

K-th Nearest Neighbor

Another often used and simple density estimation technique is based on the distribution of nearest neighbors. For each point (e.g., a pixel location on the two-dimensional grid) we can find the distance to the Kth-nearest neighbor, d_K. In this method, originally proposed in an astronomical context by Dressler et al. [11], the implied point density at an arbitrary position x is estimated as

$$\widehat{f}_K(x) = \frac{K}{V_D(d_K)}, \tag{6.13}$$

Why this works: this is the solution for an estimator that assumes the underlying **density field is locally constant**.

Kth NN is able to resolve multiple scales!

K-th Nearest Neighbor

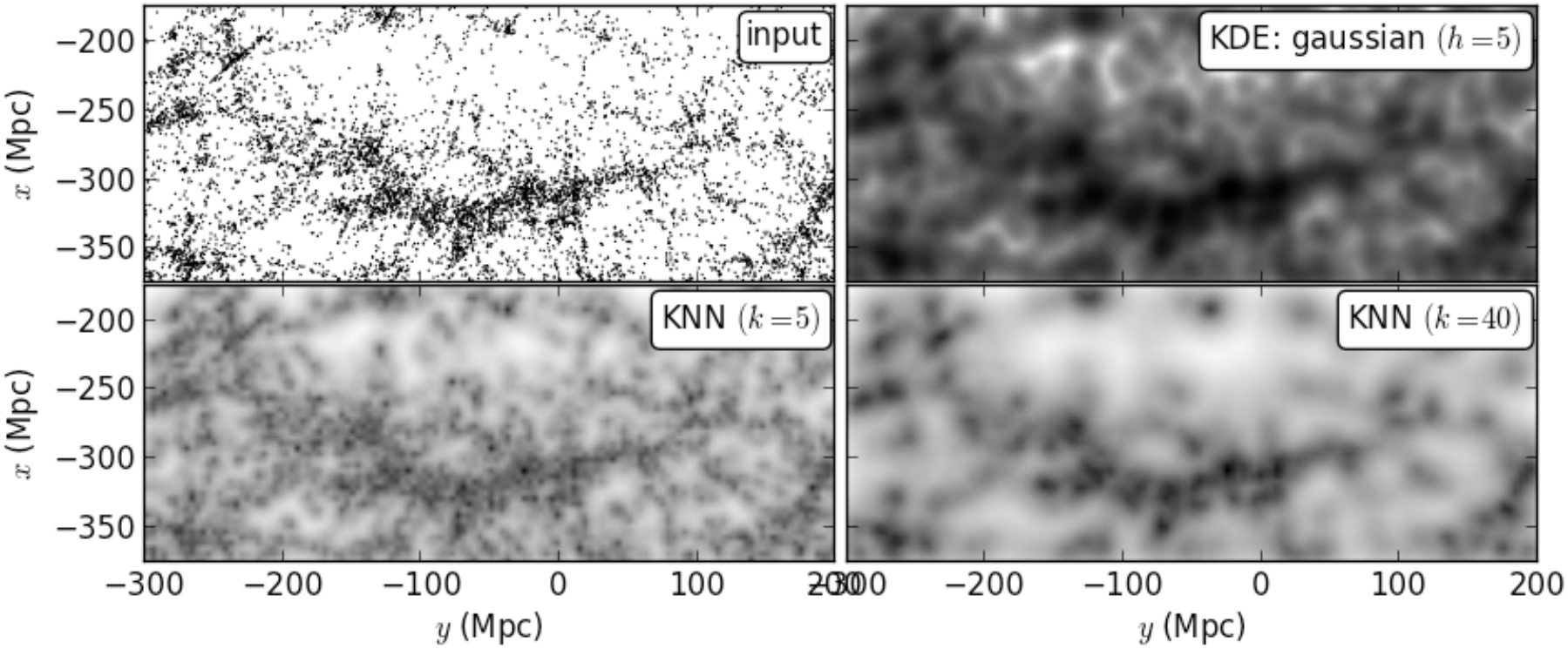
Another often used and simple density estimation technique is based on the distribution of nearest neighbors. For each point (e.g., a pixel location on the two-dimensional grid) we can find the distance to the Kth-nearest neighbor, d_K. In this method, originally proposed in an astronomical context by Dressler et al. [11], the implied point density at an arbitrary position x is estimated as

$$\widehat{f}_K(x) = \frac{K}{V_D(d_K)}, \tag{6.13}$$

In practice, the evaluation is even simpler: we use the estimator to the right, and compute the normalization constant C so that f_K sums up to 1 (or to the number of data points, if we're not estimating the PDF).

$$\widehat{f}_K(x) = \frac{C}{d_K^D},$$

K-th Nearest Neighbor: The Sloan Great Wall



Note the multi-scale performance: we resolve much more detail in areas where more samples have been drawn.

Trade-offs and free parameters: what K to choose? The fractional accuracy grows with K (error is proportional to sqrt(K)), but at the expense of resolution.

Improvement: K nearest neighbors

With Kth NN, we're throwing away information: where neighbor 1, 2, … , K-1 are tells us something about the local density field as well!

Example: If virtually all your neighbors are in a ring of some radius, and only a few are within it, you're clearly in a lower density region than the Kth NN method would predict.

Improvement: K nearest neighbors

We can do better:

This general method can be improved (the error in $\hat{f}$ can be decreased without a degradation in the spatial resolution, or alternatively the resolution can be increased without increasing the error in $\hat{f}$) by considering distances to *all* K nearest neighbors instead of only the distance to the Kth-nearest neighbor; see [18]. Given distances to all K neighbors, $d_i, i = 1, \ldots, K$,

$$\hat{f}_K(x) = \frac{C}{\sum_{i=1}^{K} d_i^D}. \tag{6.15}$$

This (simple!) estimator is derived using the same assumption of locally constant density as the Kth Nearest Neighbor, but is shown to be optimal.

I.e., it optimally uses all information that is available to estimate the local density.

Examples

Distribution of RR Lyrae in SDSS Stripe 82

(Sesar et al. 2009)

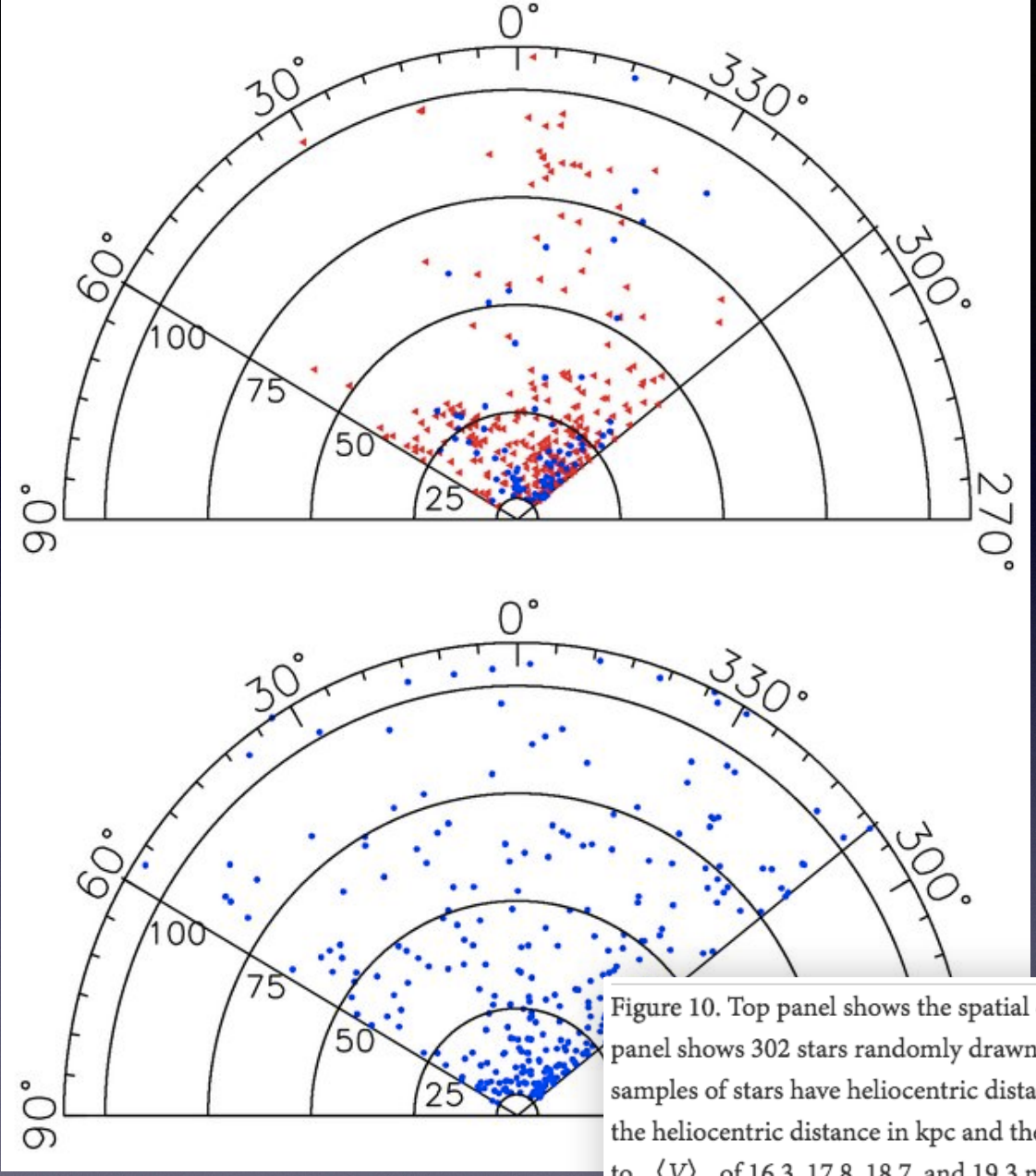


Figure 10. Top panel shows the spatial distribution of 366 RRab stars from stripe 82, while the bottom panel shows 302 stars randomly drawn from a smooth model distribution (Equation (16)). Both samples of stars have heliocentric distances between 5 and 120 kpc and $|\mathrm{decl.}| < 1.^{\circ}25$. The radial axis is the heliocentric distance in kpc and the angle is the equatorial right ascension. The circles correspond to $\langle V \rangle$ of 16.3, 17.8, 18.7, and 19.3 mag (corrected for ISM extinction). The RRab stars shown in the top panel are divided into Oosterhoff I (289 stars, red triangles) and Oosterhoff II (77 stars, blue dots) using the selection boundary shown in Figure 16.

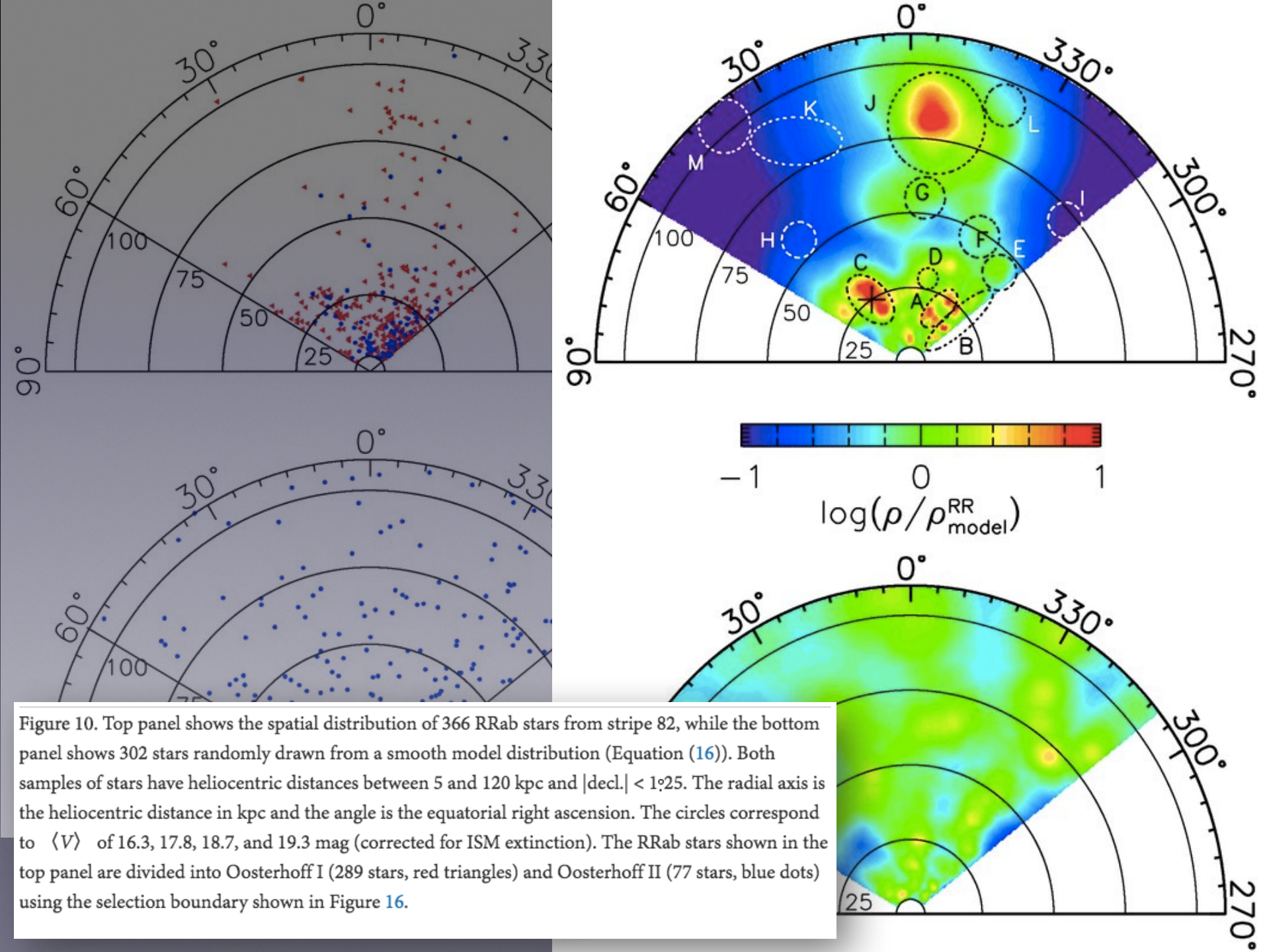
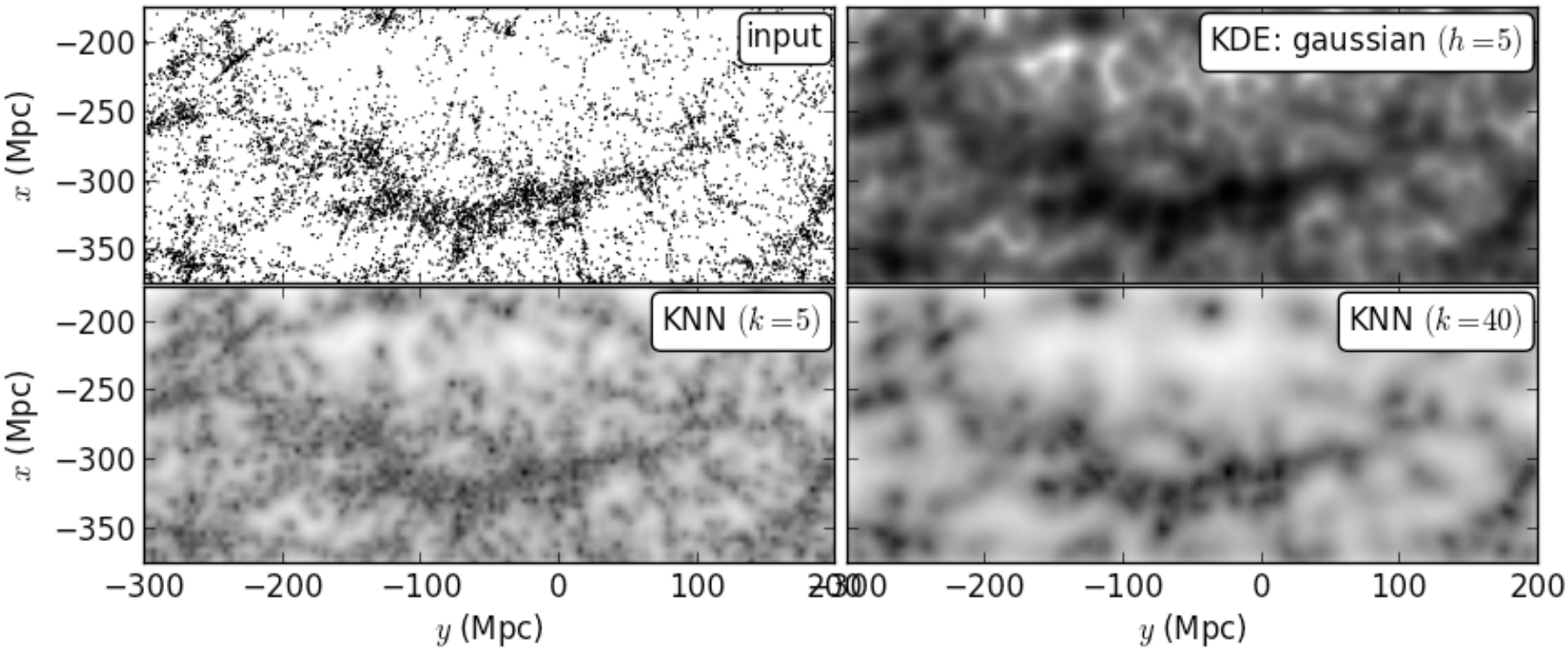


Figure 10. Top panel shows the spatial distribution of 366 RRab stars from stripe 82, while the bottom panel shows 302 stars randomly drawn from a smooth model distribution (Equation (16)). Both samples of stars have heliocentric distances between 5 and 120 kpc and |decl.| < 1.°25. The radial axis is the heliocentric distance in kpc and the angle is the equatorial right ascension. The circles correspond to $\langle V \rangle$ of 16.3, 17.8, 18.7, and 19.3 mag (corrected for ISM extinction). The RRab stars shown in the top panel are divided into Oosterhoff I (289 stars, red triangles) and Oosterhoff II (77 stars, blue dots) using the selection boundary shown in Figure 16.

Nearest Neighbors: Sloan Great Wall

- make this plot by running
  %run fig_great_wall.py

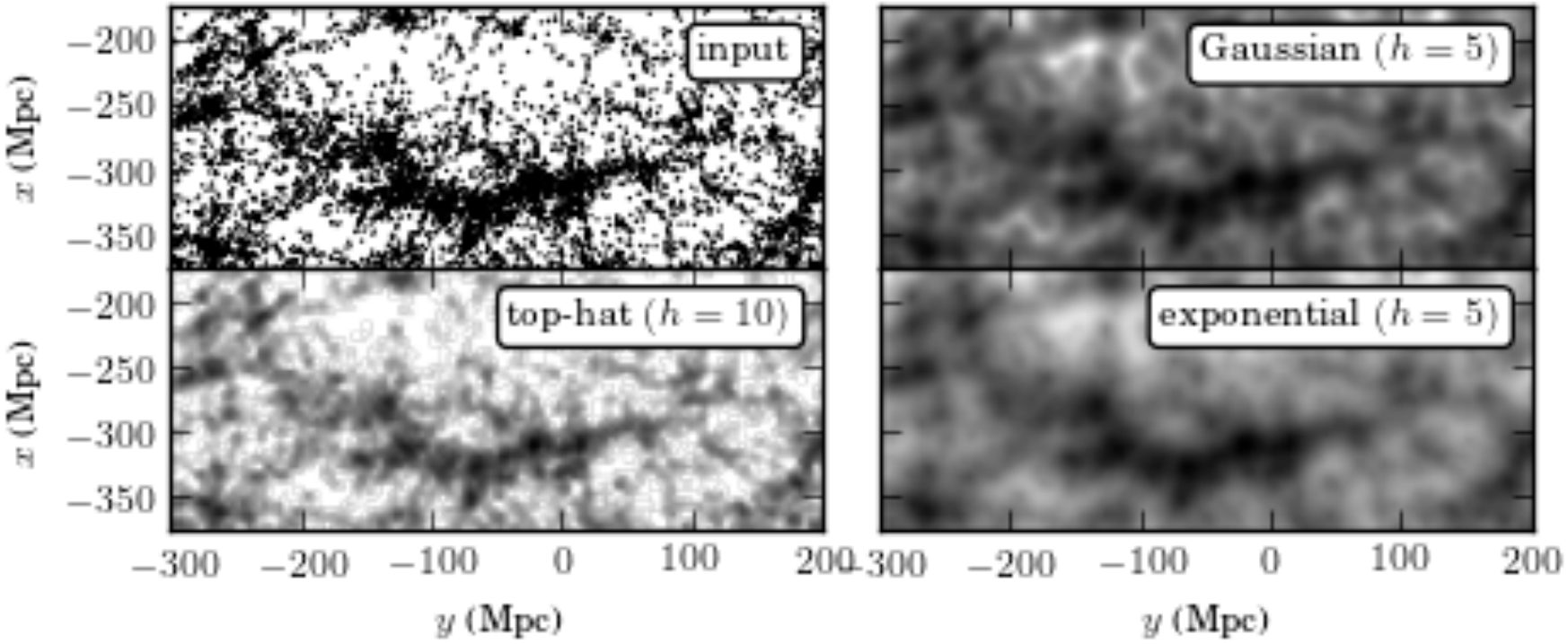


KNN here stands for "K nearest neighbors": a Bayesian method good for sparsely sampled data (Ivezić et al. 2005, AJ 129, 1096)

KDE in 2-D case: Sloan Great Wall

- you **could** make this plot by running
  %run fig_great_wall_KDE.py

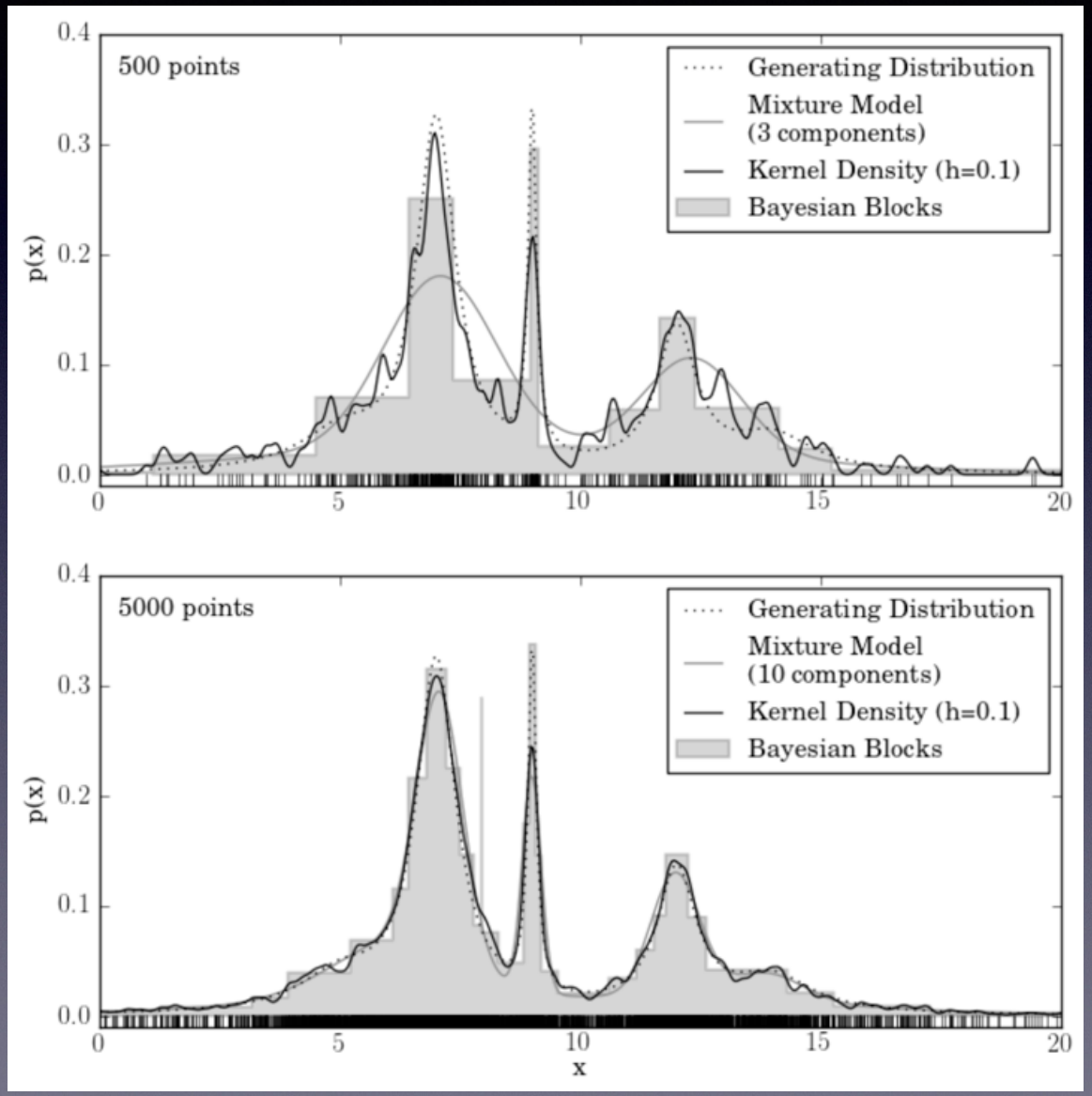
SDSS Great Wall



"SDSS Great Wall" is the largest known concentration of galaxies (Gott et al. 2005, ApJ 624, 463)

Comparing different methods:

- you **could** make this plot by running
  %run fig_GMM_density_estimation.py



Key point: for large datasets, all methods result in similar estimates

puzzle: note the spike at x~8 for the BB algorithm in the bottom panel