

Lecture 8: Kernel Density Estimator

D. Jason Koskinen
koskinen@nbi.ku.dk

Advanced Methods in Applied Statistics
Feb - Apr 2022

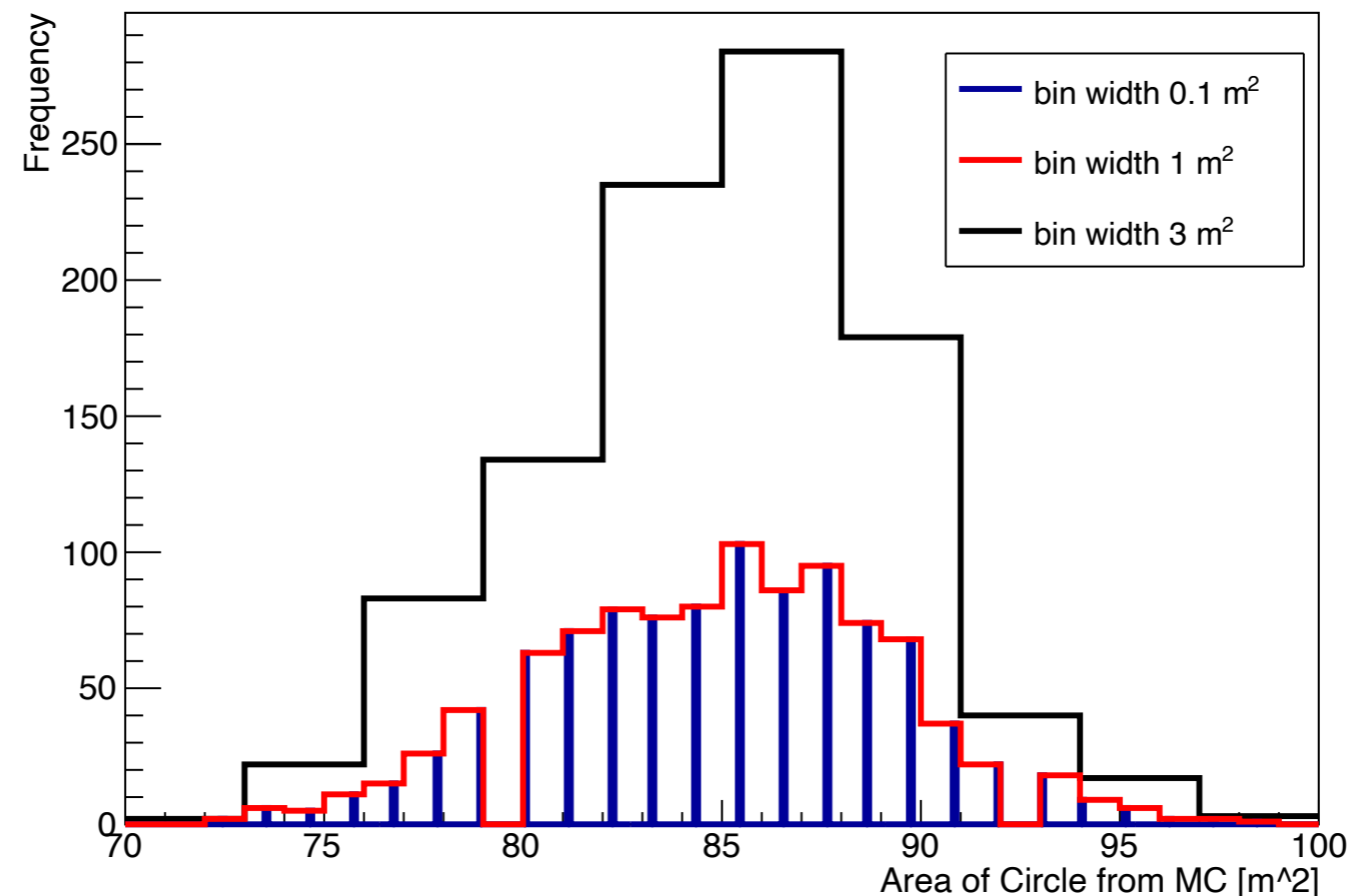
Overview

- Much of what we have covered has been parameter estimation, but using analytic or defined density expressions
- Today we cover density estimates from the data itself
- The methods are regularly employed on finite data samples that need smoothing or require non-parametric methods to get a probability distribution (function)
- Last few slides of this lecture contain extended literature for further reading

Histogram

- The histogram is one of the most simple forms of a data-driven non-parametric density estimator
- But, the only two histogram parameters (bin width and bin position(s)) are arbitrary

Circle Area Histogram



*From Lecture 1

Histograms

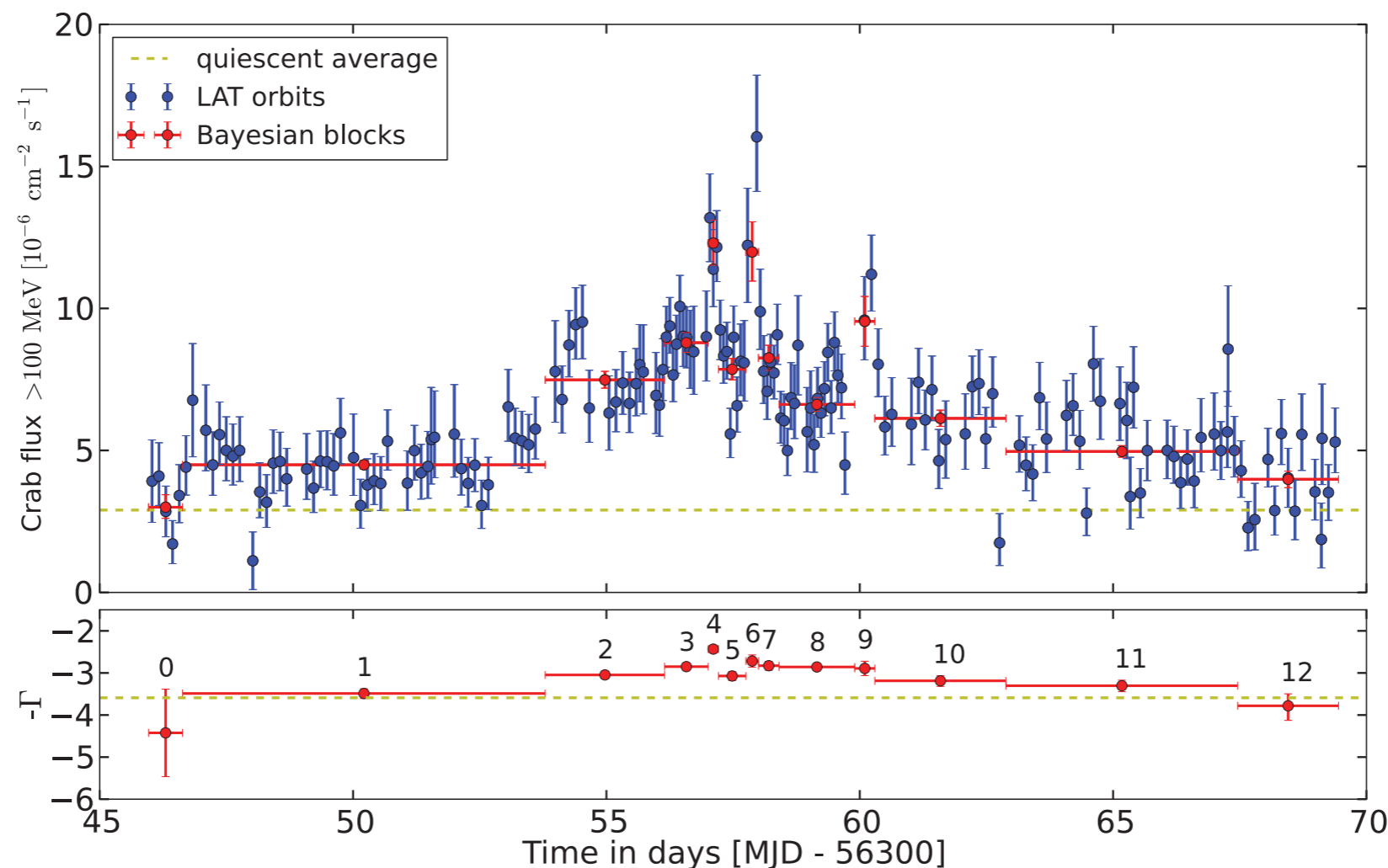
- The histogram is one of the most simple forms of a data-driven non-parametric density estimator
- But, the only two histogram parameters (bin width and bin position(s)) are arbitrary
 - Histograms are not smooth, but if the underlying true PDF is smooth then it would be good that our density estimator/function is smooth
 - More dimensions requires more data in order to have a multi-dimensional histogram which can match the true PDF
- We can avoid some of these issues with density estimates by using something more sensible

Wish List

- For density estimates what do we want?
 - non-parametric, i.e. no explicit requirement for the form of the PDF
 - (Easily) extendable to higher dimensions
 - Use data to get local point-wise density estimates which can be combined to get an overall density estimate
 - Smooth
 - At least smoother than a 'jagged' histogram
 - Preserves real probabilities, i.e. any transformation has to give PDFs which integrate to 1 and don't ever go negative
- The answer... Kernel Density Estimation (KDE)
 - Sometimes it is "Estimator" too for KDE

Bayesian Blocks

- An alternative to constant bins for histograms is to use Bayesian Blocks developed by J.D. Scargle
 - Bayesian Blocks are very useful for determining time varying changes
 - Covers many, but not all, wish list items



$$\text{Crab Flux} = \frac{dN}{dE} \propto E^{-\Gamma}$$

Basics of Data Driven Estimation

- Fixed regions in parameter space are expected to have approximately equal probabilities
 - The smaller the region(s) the more supported our assumption that probability is constant
 - The more data in each region the more accurate the density estimate
- We will keep the region fixed and find some compromise; large enough to collect some data, but small enough that our probability assumption is reasonable
 - For more thorough treatment of the original idea see the articles by Parzen and Rosenblatt in the last slides of this lecture

Example in 1 dimension

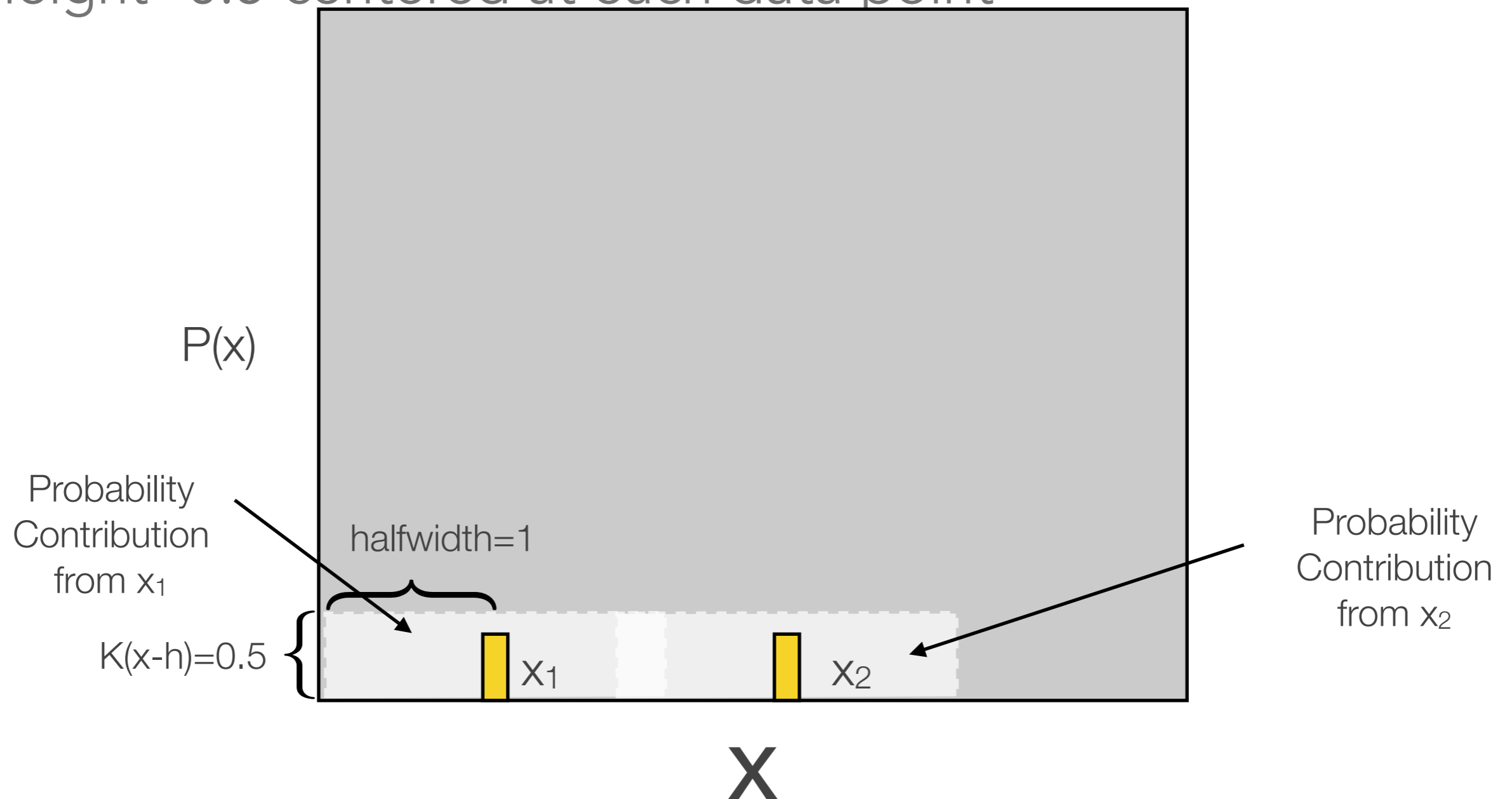
- Start with a uniform probability 'box' around data points:

$$K(u) = \begin{cases} 0.5, & \text{for } |u| \leq 1 \\ 0, & \text{for } |u| > 1 \end{cases}$$

- Notice that this kernel $K(u)$ is normalized. The **kernel is always normalized!!** The total width is $2*1$, so the 'height' must be $1/(2*1)$
- For a 1D set of data (x) , then $u=(x_i - x_n)/L$, where L is a distance around a data point that has a non-zero probability
 - The value of h is set by the analyzer
 - Here, the $L=1$

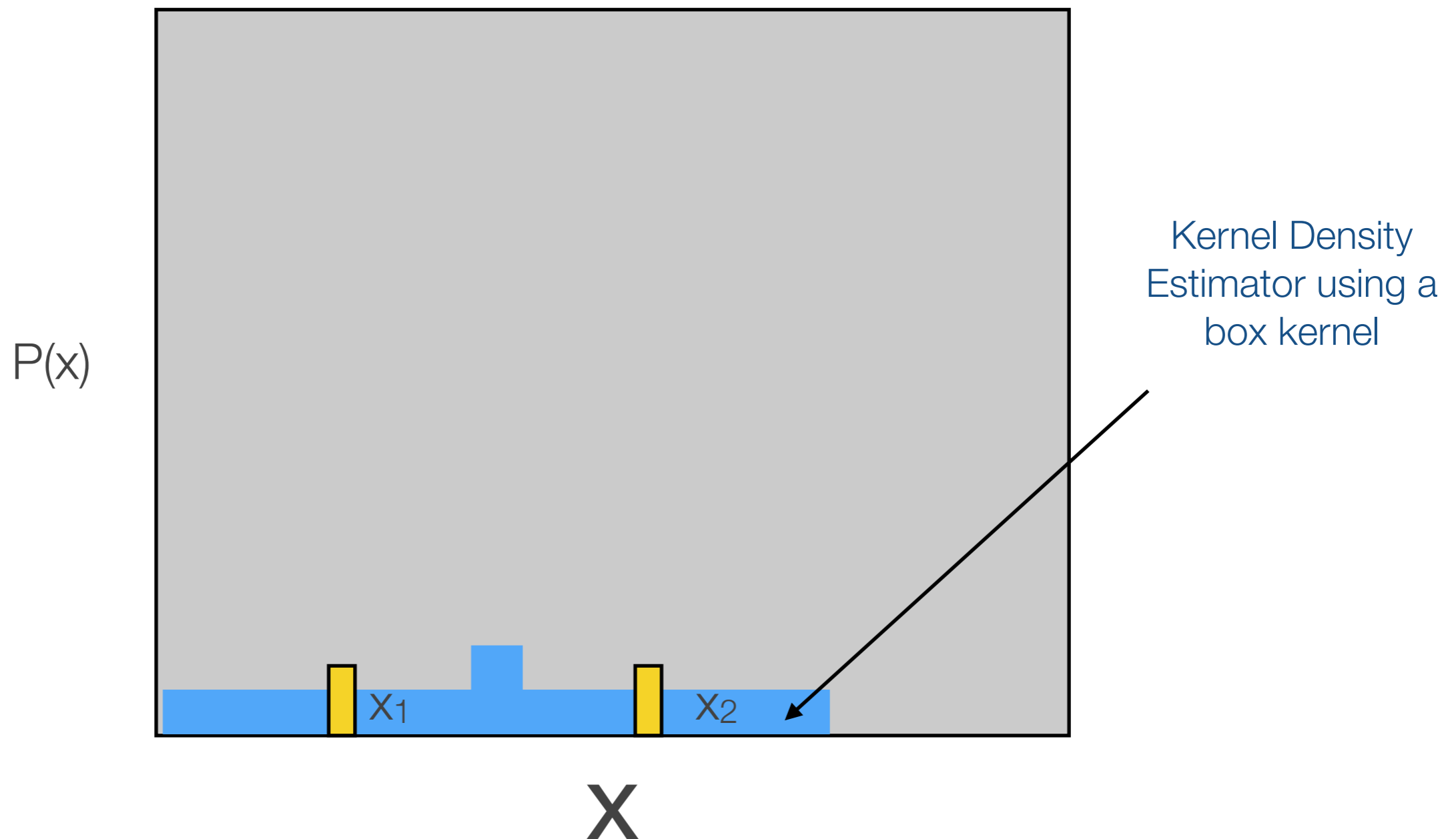
KDE by pieces

- For 2 data points (x_1 and x_2) we can produce a data driven probability density $P(x)$ using a box of total width=2 and height=0.5 centered at each data point



Final KDE

- Combine the contributions, normalize the new function to 1, and we have generated a data driven PDF.



Hyper-Cube

- Our extended 'region' definition is a D-dimensional hyper-cube with lengths equal to $2h$ on each side
- We include all points within the hyper-cube volume via some weighting scheme. This is known as the kernel (K) which is for this KDE:

$$K(\vec{x}_i) = \begin{cases} 0.5, & \vec{x}_i \text{ in region } \mathfrak{R}_d \\ 0, & \vec{x}_i \text{ outside region } \mathfrak{R}_d \end{cases}$$

for some \mathfrak{R} centered at point \vec{x}_d

- Sometimes you will see the kernel as $K(u)$ where u is the 'distance' from x_i to x_d

Kernel Characteristics

- The integrated kernel **always** equals 1

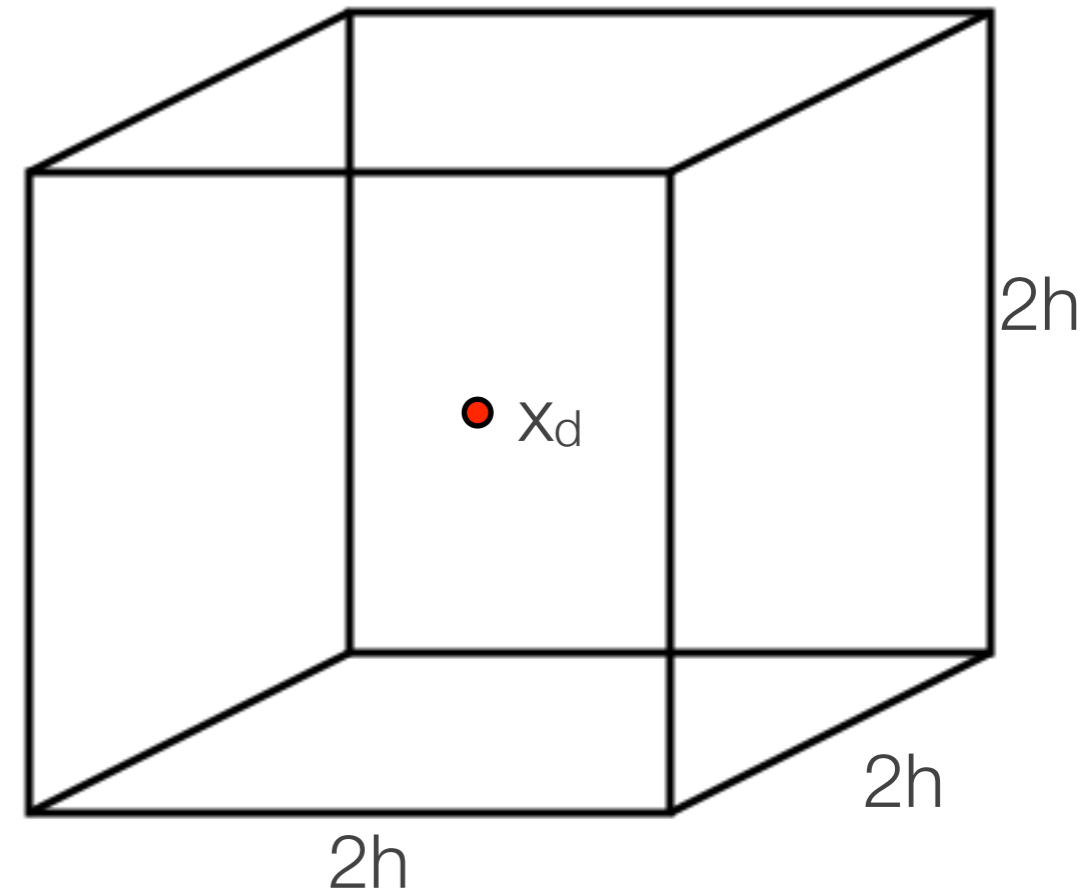
$$\int_{-\infty}^{+\infty} K(u) du = 1$$

- If 'u' is multidimensional, then so is the integration. But, the integral is **always** 1.
- Even if the kernel is not transformed to be 1D, e.g. $K(x,y,z)$ instead of $K(u)$, the integral of the kernel is always 1.

$$\iiint K(x, y, z) dx dy dz = 1 \quad \text{3D in cartesian coordinate x, y, and z}$$
$$\int \cdots \int K(s_1, \dots, s_k) ds_1 \dots ds_k = 1 \quad \text{k-dimensions represented by } \vec{s}$$

Visual Region

- Everything within the hypercube is included with an appropriate value (also known as a weight)
- For the density estimator we need:
 - To normalize by the total number of events in the sample (N)
 - The kernel is always normalized, i.e. integral or sum equals 1
- The PDF estimator (P_{KDE}) is now constructed from the individual data points
- The illustration is the estimation at a single point $x_d \in x$



$$P_{KDE}(\vec{x}) = \frac{1}{N} \sum_{n=1}^N K\left(\frac{\vec{x} - \vec{x}_n}{h}\right)$$

Exercise 1

- Use a fixed length hyper-cube KDE in 1D
- Using the following data [1,2,5,6,12,15,16,16,22,22,22,23] for the finite data sample and $h=1.5$
 - Normalize the kernel!!!!!!!!!!
 - If the total width is $2h$ then kernel height is $1/(2h)$, i.e. $K(u)=1/(2h)$ or $K(u)=0$
- Because the length is $2h$, to be in the hyper-cube each data point x_i only needs to be h in each dimension from x_d
- Calculate $P_{\text{KDE}}(x=6)$, $P_{\text{KDE}}(x=10.1)$, $P_{\text{KDE}}(x=20.499)$, and $P_{\text{KDE}}(x=20.501)$

Code this by-hand, i.e. no external packages

Exercise 1 Example

- Calculate the $P_{KDE}(x=6)$ by taking all 12 data points and seeing if they are within $\pm h$ of $x=6$, i.e. in the range 4.5 to 7.5.
- We include the data point at $x=6$ in the KDE

$$\begin{aligned} P_{KDE}(x = 6) &= \frac{1}{12} \sum_{n=1}^N K\left(\frac{6 - \vec{x}_n}{1.5}\right) \\ &= \frac{1}{12} \left[K\left(\frac{6 - 1}{1.5}\right) + K\left(\frac{6 - 2}{1.5}\right) + K\left(\frac{6 - 5}{1.5}\right) + K\left(\frac{6 - 6}{1.5}\right) + K\left(\frac{6 - 12}{1.5}\right) + \dots + K\left(\frac{6 - 23}{1.5}\right) \right] \\ &= \frac{1}{12} \left[0 + 0 + \frac{1}{3} + \frac{1}{3} + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 \right] \end{aligned}$$

KDE Comments

- The function $K()$ is known as the kernel, and h is the bandwidth
- Larger bandwidths mean more smoothing, but it can remove real features
- Smaller bandwidths will approach the true PDF better, but need lots of data points otherwise they are 'bumpy'
- The fixed window KDE is similar to a histogram, but has better support for local densities

Hyper-Cube

- We could use the hyper-cube kernel to construct a probability density, but there are a few drawbacks to a hyper-cube kernel
 - We have discrete jumps in density and limited smoothness
 - Nearby points in x have some sharp differences in probability density, e.g. $P_{\text{KDE}}(x=20.499)=0$ but $P_{\text{KDE}}(x=20.501)=0.08333$
 - All data have equal weighting and contribution regardless of distance to the estimation point
- So let's switch to a different kernel with weights that decrease smoothly as a function of distance from the estimation point

Gaussian Kernel

- The generic KDE expression remains similar, e.g.

$$P_{KDE}(\vec{x}) = \frac{1}{N} \sum_{n=1}^N K\left(\frac{\vec{x} - \vec{x}_n}{h}\right)$$

- The kernel is now (and there is no 'h' because we're switching to a gaussian kernel):

$$K(\vec{x}, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{|\vec{x} - \vec{x}_n|^2}{2\sigma^2}}$$

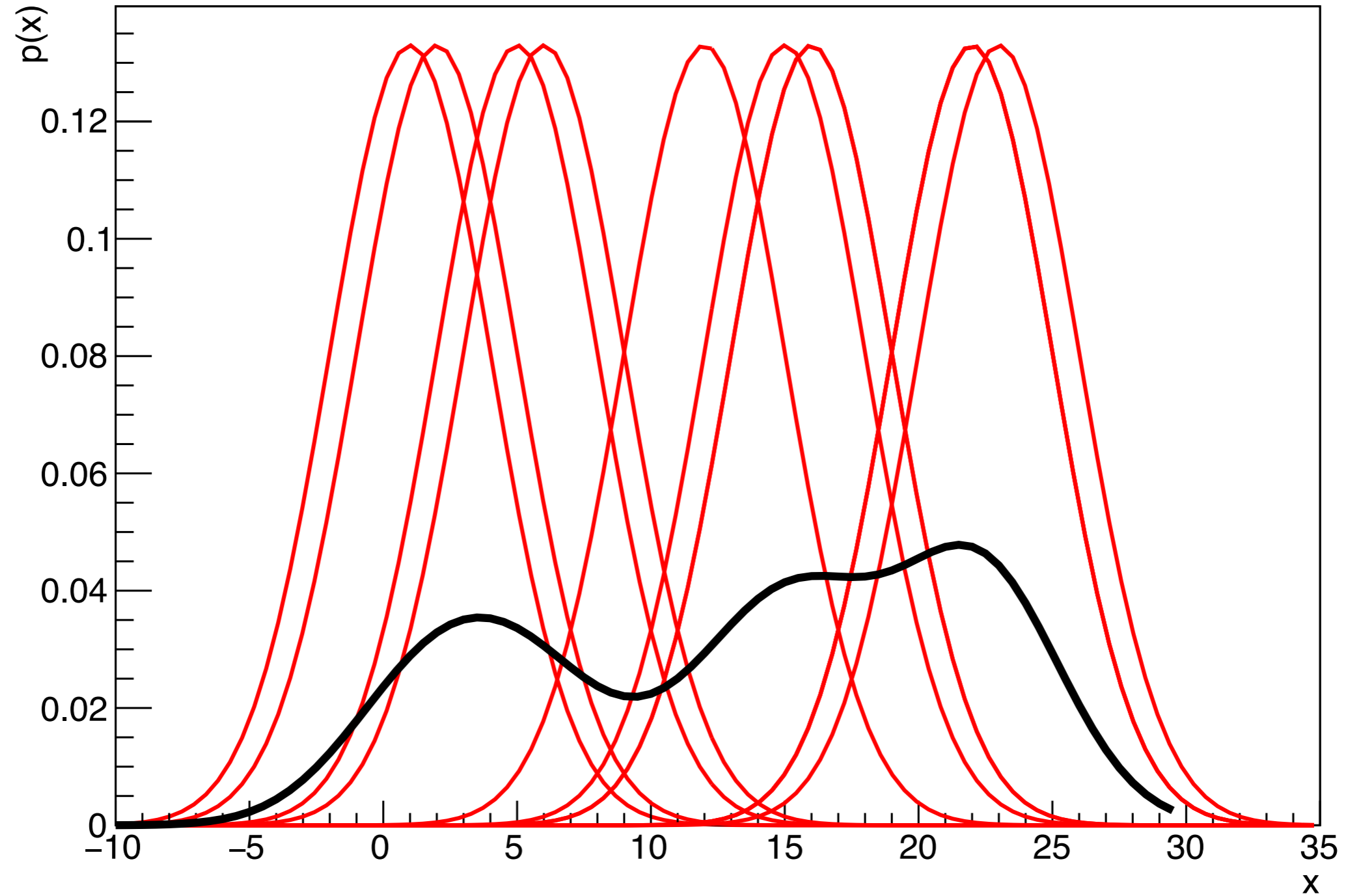
- The kernel at each data point now contributes a non-zero probability from $[-\infty, +\infty]$ smoothly with decreasing contributions as a function of distance
 - Each data point and corresponding kernel integrate to 1 over the whole parameter space

Exercise 2

- Redo exercise 1 using the new Gaussian kernel
- For the gaussian width use $\sigma=3$
- Calculate the KDE two ways:
 - Writing software where you code the gaussian function
 - Using an external software package
- Plot the density estimate $P_{\text{KDE}}(x)$ over the following range of $-10 < x < 35$
- [Optional] If you have time, plot the individual kernel contributions too

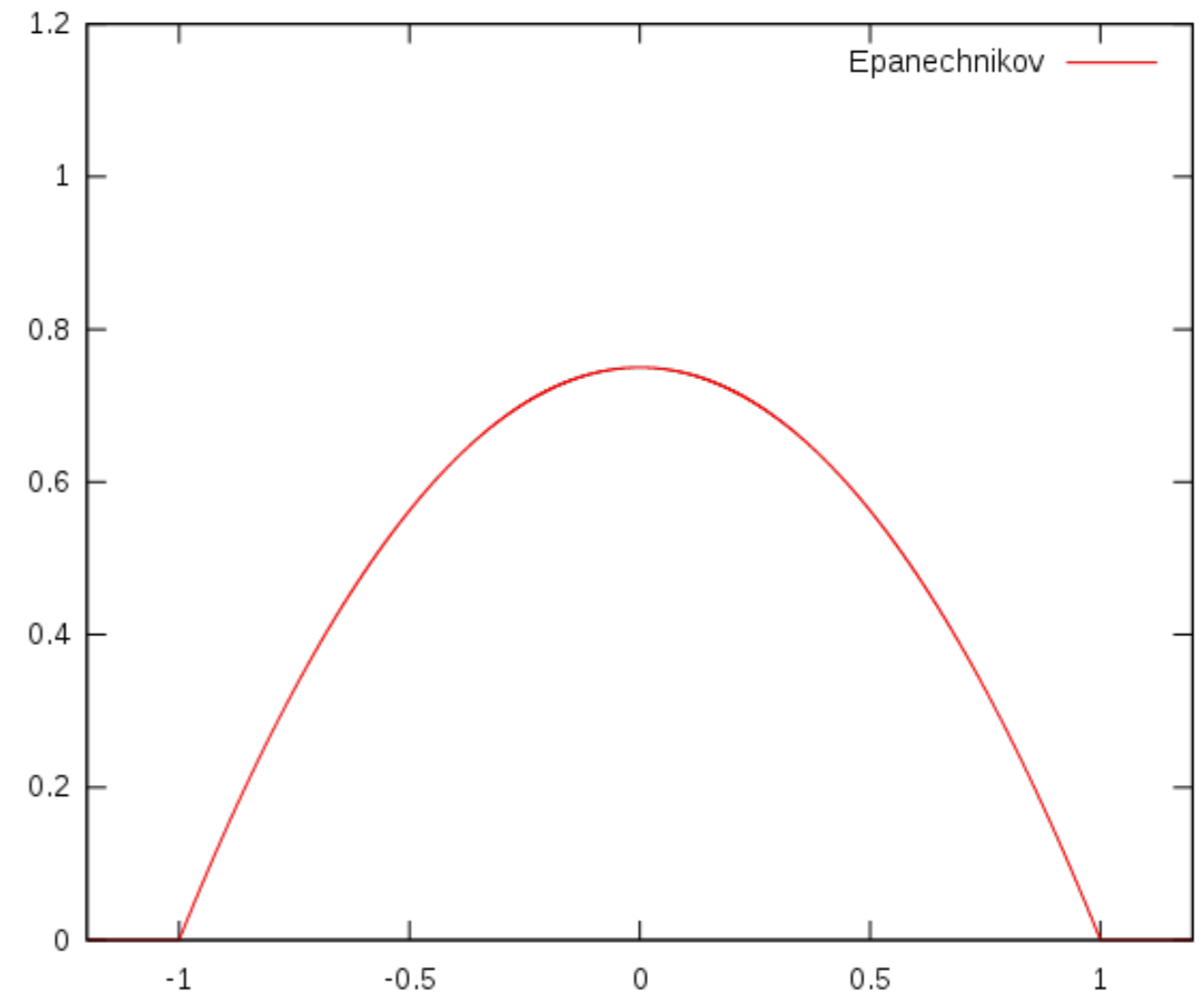
Exercise 2 KDE plot

Gaussian Kernels ($\sigma=3.00$)



Compact Kernel

- The gaussian kernel contributes across the whole space (infinite support), but sometimes we want compact support, i.e. zero outside of a specific range
 - Maybe some parameters are constrained to be non-negative
 - Maybe, we know the physical system has either boundaries or effective cut-offs
- A common compact support kernel is the Epanechnikov kernel



$$K(u) = \begin{cases} \frac{3}{4}(1 - u^2) & \text{for } |u| \leq 1 \\ 0 & \text{for } |u| > 1 \end{cases}$$

*[https://en.wikipedia.org/wiki/Kernel_\(statistics\)](https://en.wikipedia.org/wiki/Kernel_(statistics))

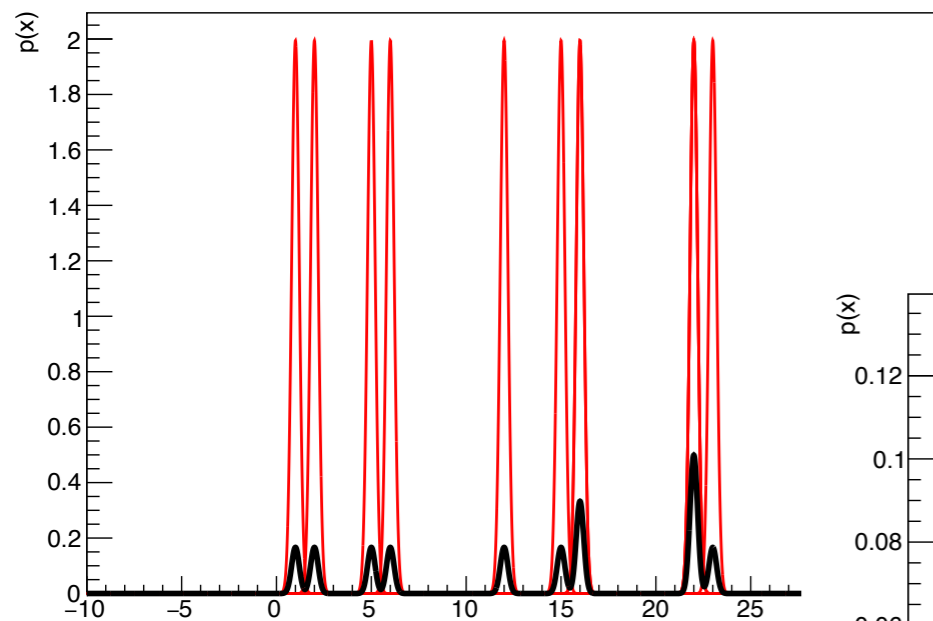
Exercise 3

- Redo exercise 2 using the Epanechnikov kernel with a bandwidth that you choose
- In a nicely formatted table compare Calculate $P_{\text{KDE}}(x=6)$, $P_{\text{KDE}}(x=10.1)$, $P_{\text{KDE}}(x=20.499)$, and $P_{\text{KDE}}(x=20.501)$ between the 3 different kernels; Parzen-Rosenblatt, gaussian, and Epanechnikov
 - Use either your by-hand(s) version or external package

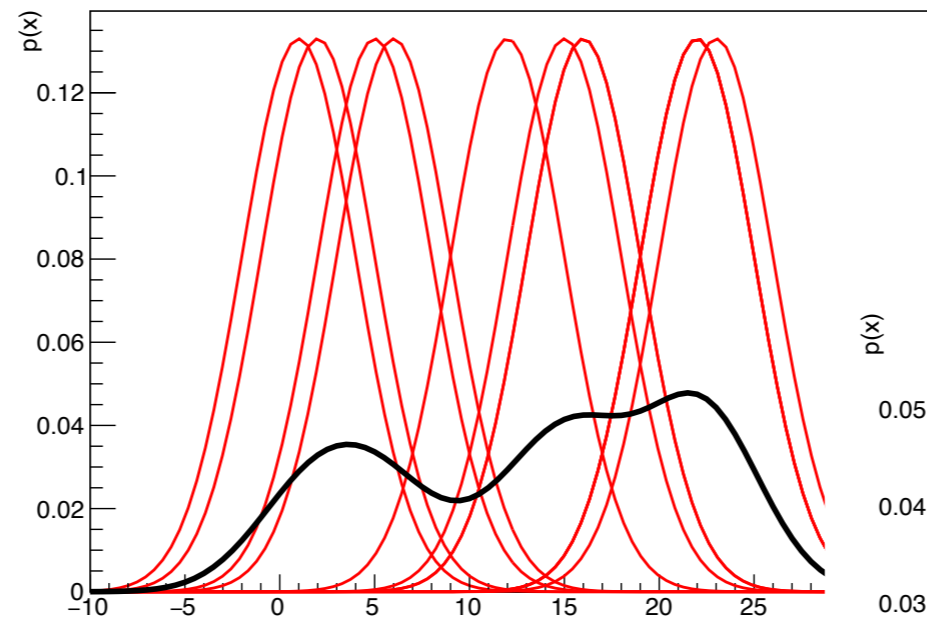
Kernel Bandwidth

- Every KDE is, unfortunately, strongly influenced by the kernel bandwidth, which is a user defined free parameter

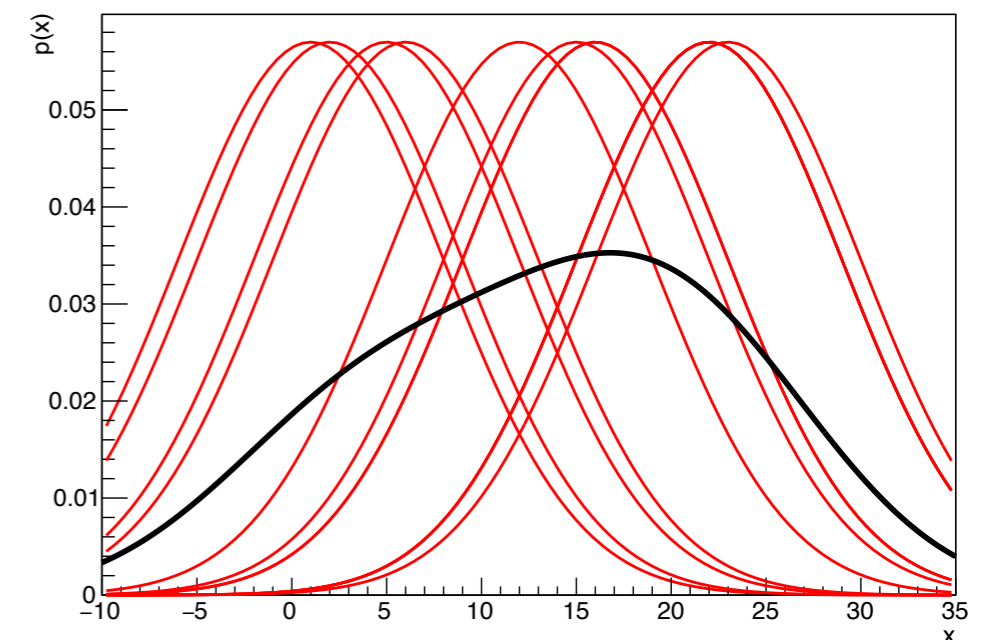
Gaussian Kernels ($\sigma=0.20$)



Gaussian Kernels ($\sigma=3.00$)



Gaussian Kernels ($\sigma=7.00$)



Bandwidth Selection

- An analytic approach to bandwidth selection is to choose a bandwidth which minimizes the mean integrated square error (MISE)

$$MISE(h) = E \left[\int (P_{KDE}(\vec{x}) - P(\vec{x}))^2 dx \right]$$

- But analytically this requires some known form of the underlying distribution
- Assuming that the underlying distribution is gaussian, the optimal bandwidth is

$$h \approx 1.06 \hat{\sigma} N^{-1/5}$$

$\hat{\sigma}$ standard deviation from data
 N number of data points

Bandwidth Selection Non-parametric

- Instead of using a known function we can use subsets of the data as cross-validation of the kernel bandwidth with the integrated square error (ISE)

$$\begin{aligned} ISE(\hat{f}_h) &= \int (\hat{f}_h(y) - f(y))^2 dy \\ &= \int (\hat{f}_h(y))^2 dy - 2 \int \hat{f}_h(y) f(y) dy + \int f(y)^2 dy \end{aligned}$$

- This can be shown to converge to a least squares cross-validation (LSCV) expression as

$$LSCV(h) = \int (\hat{f}_h(y))^2 dy - \frac{2}{N} \sum_{i=1}^n \hat{f}_{-i}(y_i)$$

- The expression $\hat{f}_{-i}(y_i)$ is the kernel estimator from the data omitting the data point y_i which is also known as the “leave-one-out” density estimator

*<https://projecteuclid.org/euclid.ss/1113832723>

KDE in 2D, and more

- While the previous examples and work were for 1-dimension, the kernels work just fine in additional dimensions
 - No escaping the curse of dimensionality :-)
 - Similar to all other multi-dimensional problems, anything beyond 3D is difficult to visualize
- Kernel bandwidths do not have to be the same in each dimension
 - Either specify the bandwidth in each dimension, or
 - Transform the parameter space(s) to be uniform for a given bandwidth

Exercise 4

- There are many online tutorials covering different 2D density estimation problems in R, python, MatLab, etc.
 - Eruption of "Old Faithful" geyser
 - Rendering of text and numerals
 - Spread of diseases
 - Geographical population densities

Exercise 5

- Use the 500 pseudo-experiment bootstrap from Lecture “Parameter Estimation and Confidence Intervals” exercise 1b to produce a 2D KDE
 - Because the LLH method gives precise contours, you can compare the contours from the KDE

The End

More KDE comments

- The kernel is symmetric about each data point
 - Makes sense, because the region near the data point should have a similar probability for a narrow (enough) bandwidth
 - Kernel symmetry is not technically a requirement, but in practice symmetry is often desirable because then the average of the kernel is centered on the data point
- The kernel density estimator PDF is often used for Monte Carlo sampling
 - E.g. N-body simulations (galaxy formation, astrophysical large scale structure, disease propagation in an ecosystem, etc.) can take months to generate 200 data points across 3 dimensions or parameters. Real data is much, much larger. In order to use our N-body PDF, we can sample from a smoothed PDF from a KDE.
- Because KDEs require 'subjective' input, clearly state the kernel, bandwidth, and any optimization from an analysis

Further Info

- Fixed kernel width window, Parzen-Rosenblatt window
 - Parzen (<http://www.jstor.org/stable/2237880>)
 - Rosenblatt (<http://projecteuclid.org/euclid.aoms/1177728190>)
- Nice list of various kernels at [https://en.wikipedia.org/wiki/Kernel_\(statistics\)](https://en.wikipedia.org/wiki/Kernel_(statistics))
- Very nice article on kernel bandwidth selection review <https://projecteuclid.org/euclid.ss/1113832723>
- Collection of other cross-validation techniques <https://cran.r-project.org/web/packages/kedd/vignettes/kedd.pdf>

Further Info (cont.)

- Variable bandwidth kernels
 - Z. I. Botev, et al., Kernel Density Estimation via Diffusion, *Annals of Statistics*, 38 5, 2916-2957 (2010).
 - I. S. Abramson, On bandwidth variation in kernel estimates—A square root law, *Annals of Statistics*, 10 4, 1217-1223 (1982).
- Any other suggestions?