Probability Density Estimation, Smoothed Mean Estimators

Mark M. Fredrickson (mfredric@umich.edu)

Computational Methods in Statistics and Data Science (Stats 406)

Probability Density Estimation

Setting

Suppose we have IID data with density f:

$$X_i \sim F, \quad F(x) = \int_{-\infty}^{x} f(t)\, dt$$

Goal: Estimate f.

We've seen the ECDF:

$$\widehat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} I(X_i \leq x)$$

The **empirical density function** is then a discrete probability mass function

$$\widehat{f}_n(x) = \frac{1}{n} \sum_{i=1}^{n} I(X_i = x)$$

Smoothing

If we think X is a continuous random variable, the empirical density function seems unsatisfactory.

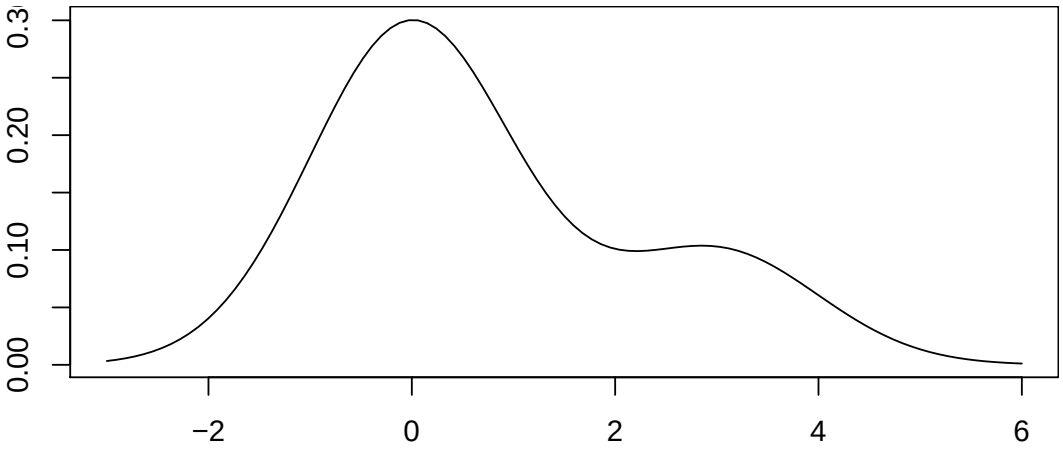
We will try to **smooth** the density function in some way so that $\widehat{f}_n(x) \neq 0$ for x not observed in the sample.

Key idea: if there are many observations X_i close to x, this implies $f(x)$ high (and vice-versa).

We will start with the **univariate case**. Later we'll consider **bivariate extensions**. After that, we'll see smoothing for **conditional distributions**.

Simulated Data: Mixture of $N(0,1)$ anf $N(3,1)$

```
> x <- rnorm(1000, mean = 3 * rbinom(100, size = 1, 0.25))
```



Histograms

A **histogram estimator** creates bins k bins

$$[t_j, t_{j+1}), \quad j = 1, \ldots, k$$

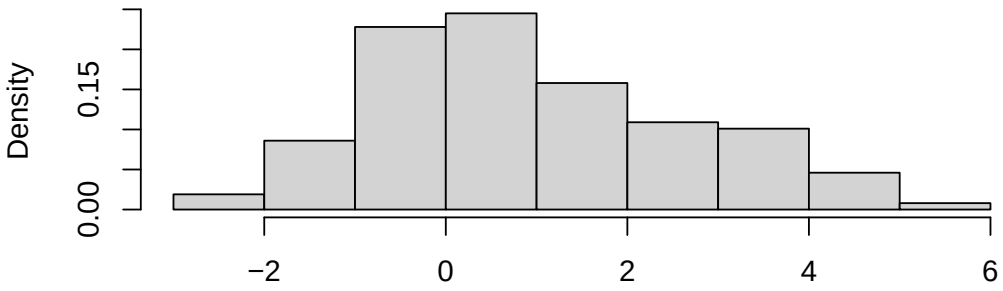
and counts then number of observations X_i that fall in each bin:

$$V_j = \sum_{i=1}^{n} I(X_i \in [t_j, t_{j+1}))$$

Then an estimator of the density at point x is:

$$\widehat{f}(x) = \frac{1}{n} \sum_{j=1}^{k} \frac{V_j}{t_{j+1} - t_j} I(x \in [t_j, t_{j+1}))$$

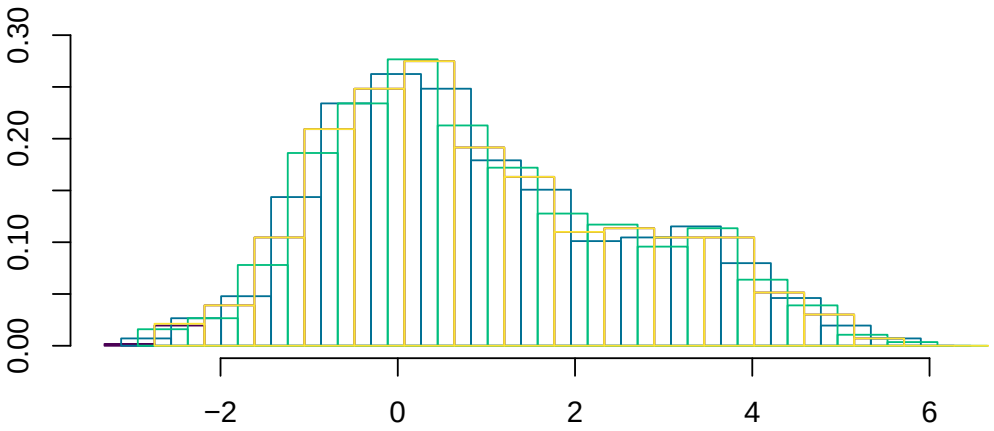
Histogram, further notes



- $\widehat{f}(x)$ is **piece-wise constant** (i.e., the same for all x in the same bin)
- We usually make all bins have the same width $h = t_{j+1} - t_j$
- There is a tradeoff between smoothness (large h) and precision (small h).
- There are several options for setting the bin width and locations (see Rizzo, chapter 10).

Multiple Histograms

If all bins have width h, there is only one "parameter" for the histogram: **where the bins are centered**.



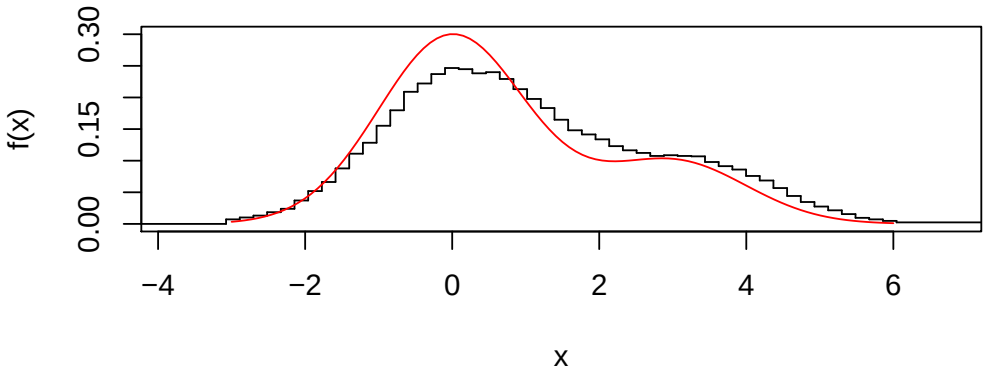
Average Shifted Histogram

Rather than estimate f with a single histogram, we can **average over m histograms**:

$$f_{\text{ASH}}(x) = \frac{1}{m} \sum_{j=1}^{m} \widehat{f}_j(x)$$

where $\widehat{f}_j$ is a histogram with bin size h start at $t^{(j)} = t_0 + h/m$.

```
> library(ash)
> ash_x <- ash1(bin1(x), 5, kopt = c(0,0))
```


f(x)
0.00
0.15
0.30
−4
−2
0
2
4
6
x

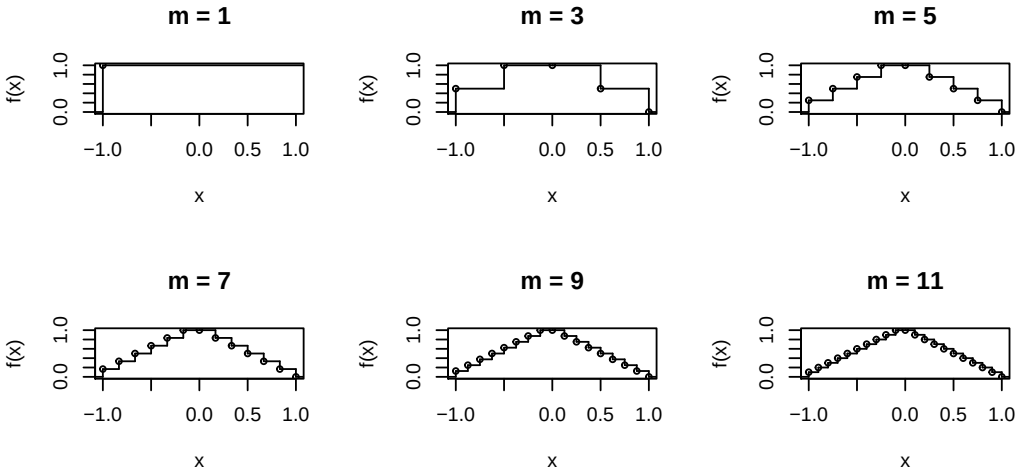
Taking $m \to \infty$

As we **increase** m (number of histograms in ASH) the approximation will get **more smooth**. What happens when $m \to \infty$?

Suppose all of the data is within $[-1, 1]$ and we want to estimate $f(0)$.

We will use histograms with **bin width one** and we will **discard any bins that exceed [-1,1]**. As we slide the histograms across, what proportion line up one point?

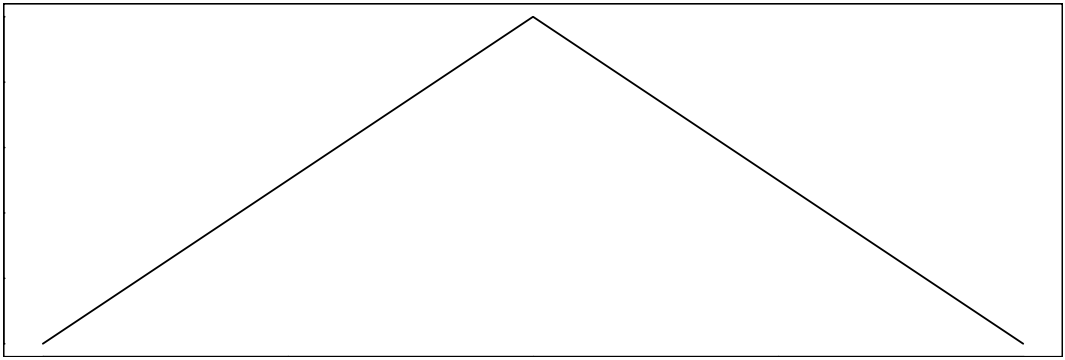
Another way of saying this is, if we keep averaging at a single point as $m \to \infty$, what proportion of neighbors within $+1/-1$ are included?


m = 1
m = 3
m = 5
m = 7
m = 9
m = 11
f(x)
x
0.0
1.0
−1.0
0.0
0.5
1.0

Triangular Distribution

As $m \to \infty$, the ASH becomes

$$\widehat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} I(X_i \in [x-h, x+h]) \left(1 - \frac{|x - X_i|}{h}\right)$$



Density objects in R

R has a built in function to perform **density estimation**:

```
> fx <- density(x, kernel = "triangular")
```

The object is list of two vectors: x values from the sample and $\hat{f}(x)$ estimates

```
> fx$x[1:5]
```

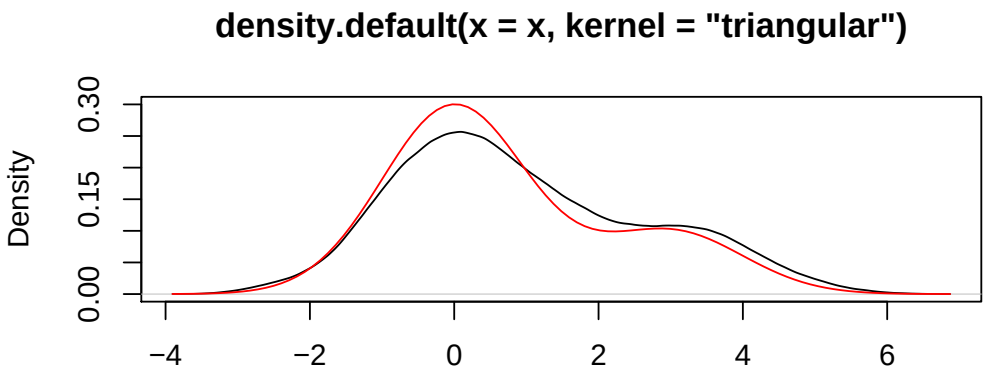
```
[1] -3.905 -3.884 -3.863 -3.842 -3.820
```

```
> fx$y[1:5]
```

```
[1] 9.544e-19 2.537e-18 8.760e-18 1.487e-17 1.626e-17
```

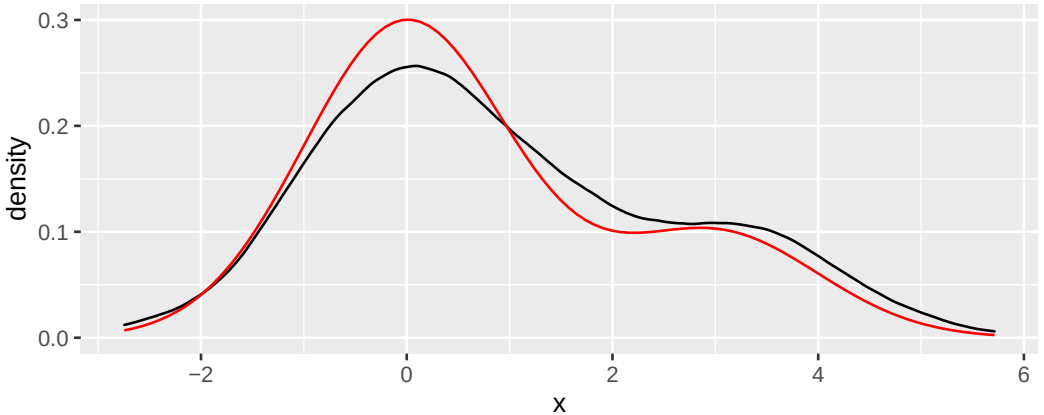
Plotting in Base R

```
> plot(fx, ylim = c(0, 0.3))
> truef <- function(x) { 0.75 * dnorm(x) + 0.25 * dnorm(x, mean = 3) }
> curve(truef(x), add = TRUE, col = "red")
```



Plotting in ggplot

```
> ggplot(data.frame(x), aes(x = x)) + geom_density(kernel = 'triangular') +
+   stat_function(fun = truef, col = 'red')
```



Kernel Density Estimators

In general, we can write:

$$\widehat{f}_{K,h}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

where

- h is the "bandwidth" of the estimator
- $K(t)$ is a probability density symmetric about 0 (the "kernel")

Optimal bandwidth depends on the true f. Heuristics discussed in Rizzo.

Some common kernels (with support)

- Gaussian $K(t) = (1/\sqrt{2\pi})\exp(-t^2/2)$ $(t \in (-\infty, \infty))$
- Epanenchnikov: $K(t) = \frac{3}{4}(1 - t^2)$ $(t \in [-1, 1])$
- Rectangular: $K(t) = 1/2$ $(t \in [-1, 1])$
- Triangular: $K(t) = 1 - |t|$ $(t \in [-1, 1])$
- Biweight: $K(t) = (15/16)(1 - t^2)^2$ $(t \in [-1, 1])$
- Cosine: $K(t) = (\pi/4)\cos(t\pi/2)$ $(t \in (-\infty, \infty))$

Properties of the kernel usually transfer to $\widehat{f}$. E.g., if K is smooth, then $f_{K,h}$ is also smooth.

Multivariate Density Smoothing

Suppose we have $X = (X_1, \ldots, X_d)$ from a **multivariate density**. We can create multivariate kernels two ways:

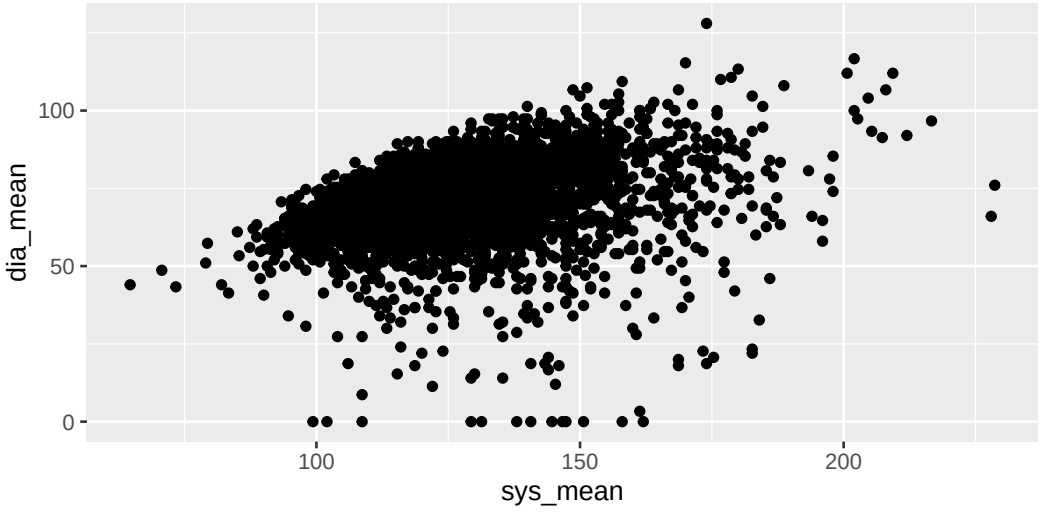
- Use multivariate kernels (e.g., the multivariate standard Normal distribution):

$$\widehat{f}(x_1, \ldots, x_d) = \frac{1}{nh_1h_2 \ldots h_d} \sum_{i=1}^{n} K\left(\frac{x_1 - X_{i1}}{h_1}, \frac{x_2 - X_{i2}}{h_2}, \ldots, \frac{x_d - X_{id}}{h_d}\right)$$

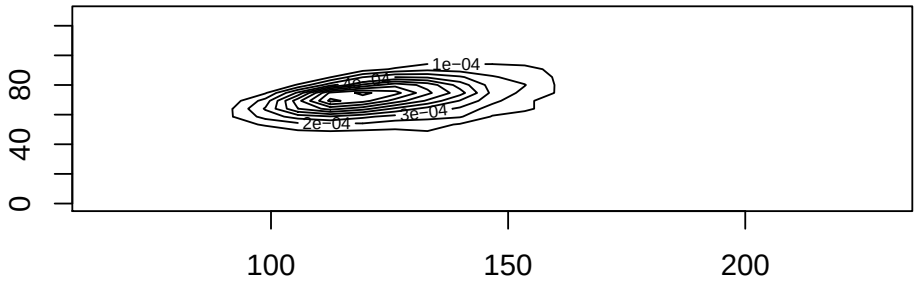
- Create a kernel from the product of unidimensional kernels ("independent joint distribution")

$$\widehat{f}(x_1, \ldots, x_d) = \frac{1}{nh_1h_2 \ldots h_d} \sum_{i=1}^{n} \prod_{j=1}^{d} K\left(\frac{x_j - X_{ij}}{h_j}\right)$$

```
> ggplot(nhanes, aes(x = sys_mean, y = dia_mean)) + geom_point()
```



```
> library(MASS)
> contour(kde2d(nhanes$sys_mean, nhanes$dia_mean))
```



Mixture distributions

Recall our definition of a **mixture distribution**

$$F(x) = \sum_{i=1}^{n} w_i F_i(x)$$

where

- Each F_i is a **valid CDF**.
- All the w_i **sum to 1**.

If each CDF has a **PDF/PMF** f_j, then

$$f(x) = \frac{d}{dx} \sum_{j=1}^{m} w_i F_i(x) = \sum_{j=1}^{m} w_i f_i(x)$$

Location-scale mixtures

Suppose we have a **mixture** where the F_j come from a single **location-scale family**:

$$F_j(x) = G\left(\frac{x - \mu_j}{\sigma}\right)$$

A slight extension to a result from the last homework gives us the density:

$$f_j(x) = \frac{g\left(\frac{x-\mu_j}{\sigma}\right)}{\sigma}$$

The mixture density is then

$$f(x) = \sum_{i=1}^{n} w_i \frac{g\left(\frac{x-\mu_j}{\sigma}\right)}{\sigma}$$

Density estimation as a mixture

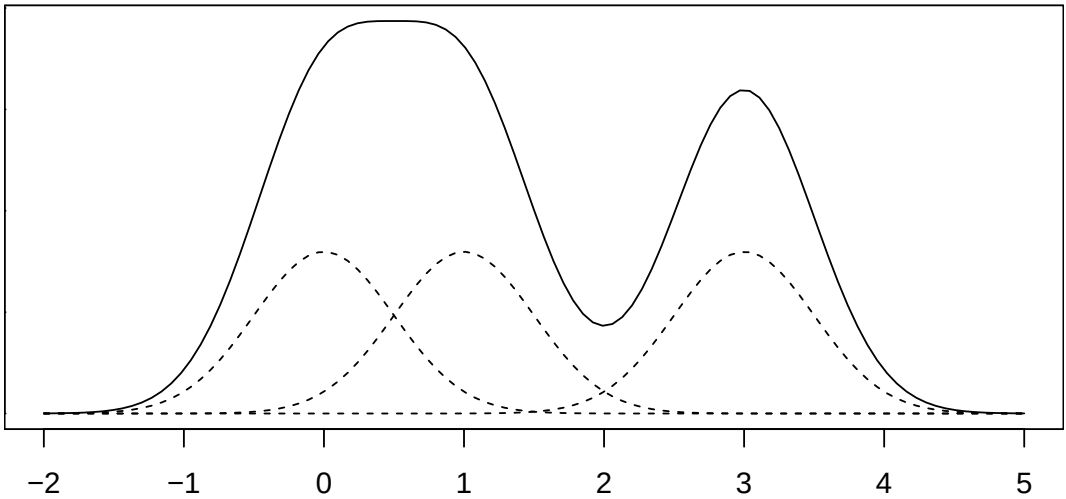
Now we can see that **kernel density estimator** is fitting a **mixture distribution**:

$$\widehat{f}_{K,h}(x) = \sum_{i=1}^{n} \frac{1}{n} \frac{K\left(\frac{x-X_i}{h}\right)}{h}$$

where

- K is the **location-scale density**,
- X_i is the **mean** μ_j
- h is the **scale/sd** σ.
- All **weights are 1/n**.

Gaussian kernel, $h = 1/2$, $X = (0, 1, 3)$



Density estimation summary

- Goal: estimate a density function from a sample
- Technique: for any point x, average using close by observations
- Need to pick a kernel and a bandwidth. Can be done with cross validation and a loss function.
- Uses:
  - Display
  - Smoothed boostrapping
  - Using densities as outcomes in regressions

Smoothed Means

Scenario

Suppose we have bivariate data (Y_i, x_i).

We will consider the Y_i to be random in some way, but the x_i values may be fixed.

We've seen examples of this in the HW, previous examples, etc.

Goal: Understand the relationship between x_i and Y_i (perhaps for prediction, data reduction, inference).

Mean functions

We've seen models of the form:

$$Y = \mu(x) + \epsilon$$

where we assume $E(\epsilon) = 0$.

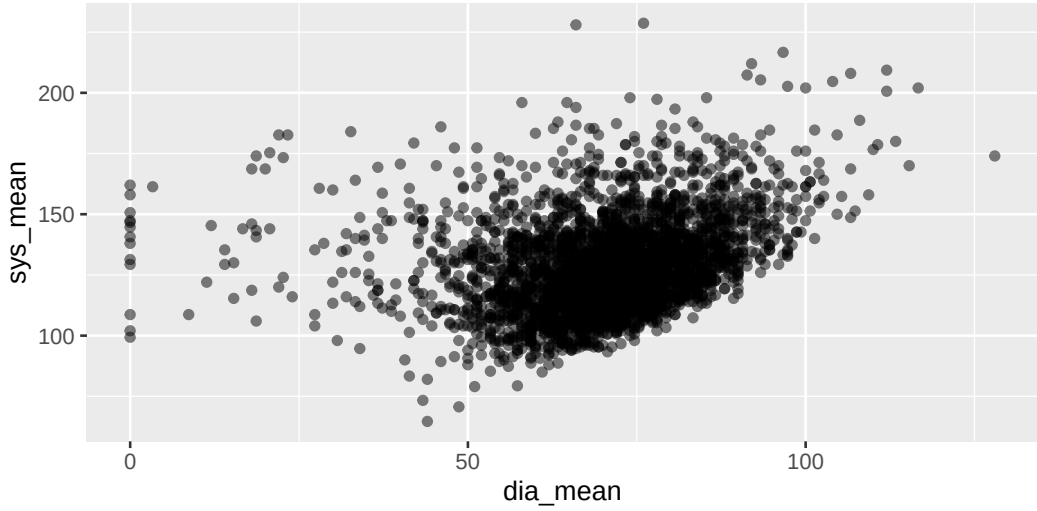
For example,

$$Y = \beta_0 + \beta_1 x + \epsilon$$

Notice because $E(\epsilon) = 0$, we have

$$E(Y|x) = \mu(x)$$

Can we use smoothing to estimate a good mean function?


200
150
100
sys_mean
0
50
100
dia_mean

Kernel Smoothing

As with density estimation, we want to let **near by values of Y_i** influence our estimate of $E(Y|x)$.

The Nardaraya-Watson estimator using a kernel function K:

$$\hat{\mu}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}$$

From density estimation to smoothed means

Consider the estimators for the **joint distribution of** (Y, X) and the **marginal distribution of** X:

$$\hat{f}(x, y) = \frac{1}{nh^2} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) K\left(\frac{Y_i - y}{h}\right)$$

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)$$

By the relationship that $f(y \mid x) = f(x, y)/f(x)$, we can plugin in our estimators to get

$$\hat{f}(y \mid x) = \frac{\frac{1}{h} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) K\left(\frac{Y_i - y}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}$$

Estimating $\mathsf{E}(Y \mid x)$

We want to estimate $\mathsf{E}(Y \mid x) = \int y\, f(y \mid x)\, dy$, so let's plug in our estimate $\hat{f}(y \mid x)$:

$$\begin{aligned}\int y\, \hat{f}(y \mid x)\, dy &= \int y\, \frac{\frac{1}{h}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) K\left(\frac{Y_i - y}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}\, dy \\ &= \frac{\frac{1}{h}\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) \int yK\left(\frac{Y_i - y}{h}\right)\, dy}{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}\end{aligned}$$

We're done if we can show

$$\frac{1}{h}\int y\, K\left(\frac{Y_i - y}{h}\right)\, dy = Y_i$$

Final step

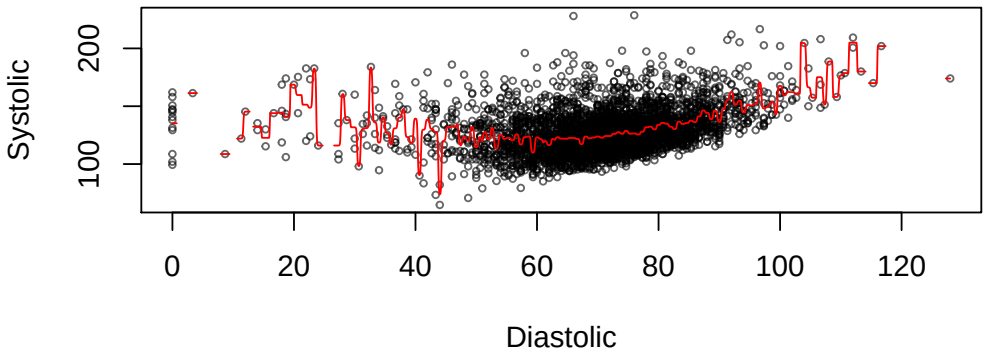
Recall that K **is a density function** for some random variable V, **symmetric about 0**. Let's do a little **change of variables**.

$$v = \frac{y - Y_i}{h} \Rightarrow y = Y_i + hv, dy = hdv$$

$$\frac{1}{h}\int (Y_i + hv)K(-v)h\,dv = \int (Y_i + hv)K(v)\,dv = Y_i + h\mathrm{E}\,(V) = Y_i$$

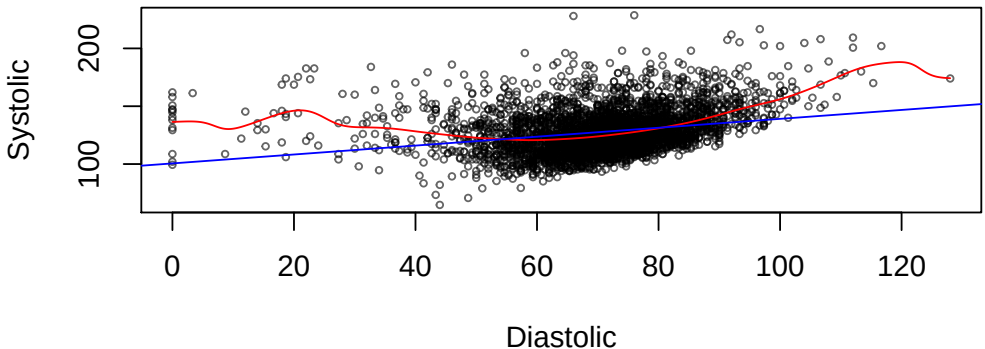
Narrow bandwidth

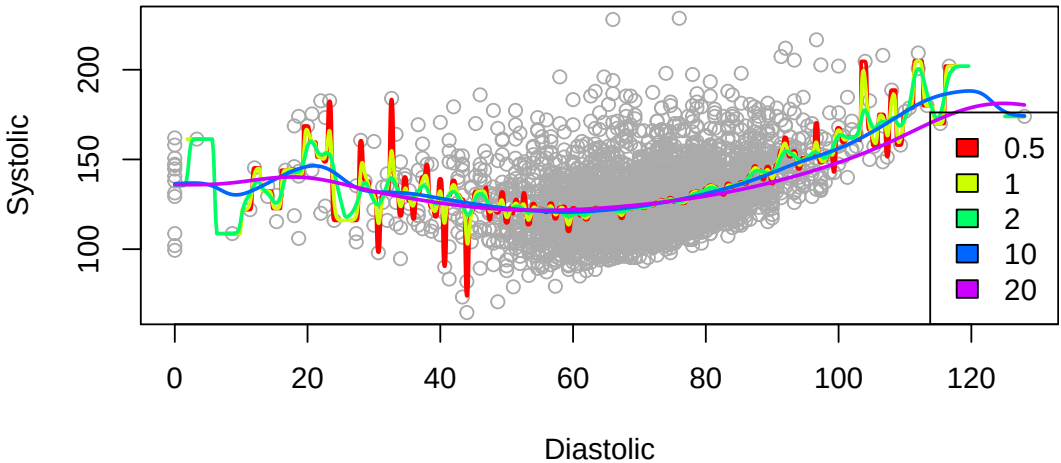
```
> ## default bandwidth is 0.5
> bp_smooth <- ksmooth(nhanes$dia_mean,
+                      nhanes$sys_mean,
+                      kernel = "normal")
> str(bp_smooth)

List of 2
 $ x: num [1:3567] 0 0.0359 0.0718 0.1077 0.1436 ...
 $ y: num [1:3567] 135 135 135 135 135 ...
```


Systolic
100
200
0
20
40
60
80
100
120
Diastolic

Better bandwidth

```
> bp_smooth_2 <- ksmooth(nhanes$dia_mean,
+                        nhanes$sys_mean,
+                        kernel = "normal", bandwidth = 10)
> bp_linea <- lm(sys_mean ~ dia_mean, data = nhanes)
```


Systolic
100
200
0
20
40
60
80
100
120
Diastolic


Systolic
100
150
200
0
20
40
60
80
100
120
Diastolic
0.5
1
2
10
20

Picking a bandwidth using CV

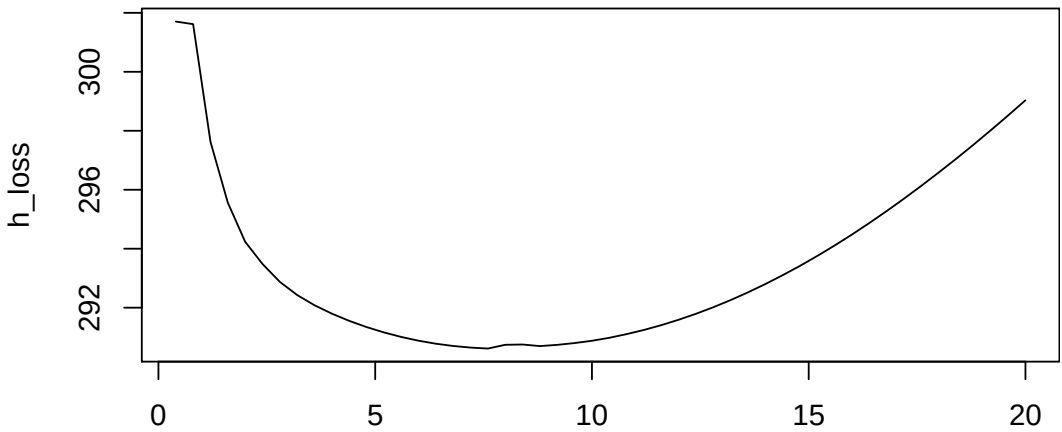
Recall that **cross validation** splits the data into (many) training and test sets and estimates **prediction loss** (costs paid for estimating new values).

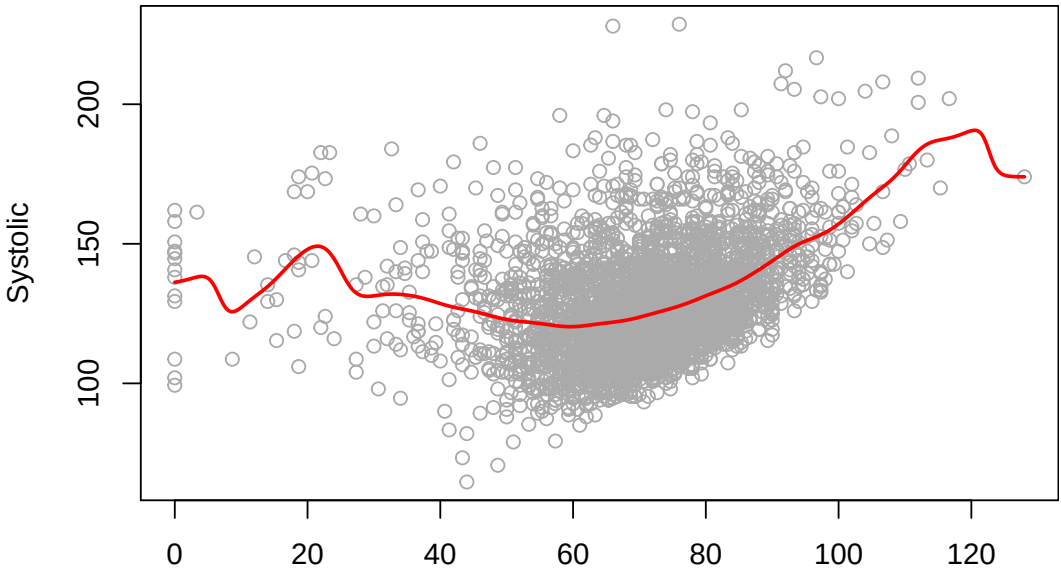
Let's pick bandwidth using **leave one out cross validation** with **squared error loss**.

```
> loss_one <- function(i, h) {
+     yhat_i <- ksmooth(bandwidth = h, kernel = "normal",
+                       x = nhanes$dia_mean[-i],
+                       y = nhanes$sys_mean[-i],
+                       x.points = nhanes$dia_mean[i])
+     (yhat_i$y - nhanes$sys_mean[i])^2 ## loss
+ }
> loss_LOOCV <- function(h) { mean(sapply(1:nrow(nhanes), loss_one, h = h)
```

```
> hs <- seq(0.4, 20, length.out = 50)
> h_loss <- sapply(hs, loss_LOOCV)
> (best <- hs[which.min(h_loss)])
```

```
[1] 7.6
```




Systolic
100
150
200
0
20
40
60
80
100
120

Comparing smoothed mean functions

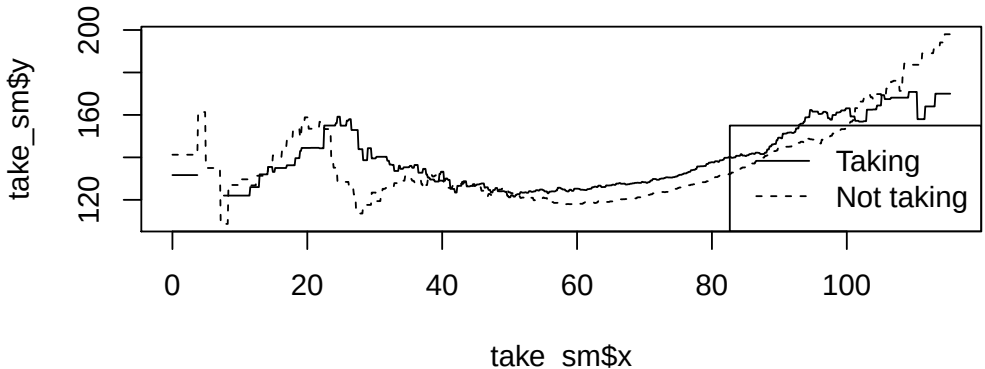
Question: Is the mean function for aspirin takers the same as the mean function for non-apsirin takers?

A little setup to make sure we compute the smooothers at the same point:

```
> range_d1 <- range(nhanes$dia_mean[nhanes$taking_aspirin])
> range_d2 <- range(nhanes$dia_mean[!nhanes$taking_aspirin])
> bounds <- c(max(range_d1[1], range_d2[1]),
+             min(range_d1[2], range_d2[2]))
```

Implementing the smoothers

```
> compute_smoother <- function(y, x, z) {
+     # we'll use the optimal bandwidth from before,
+     # but this could be computed within groups
+     ksmooth(x[z], y[z], bandwidth = 7.6,
+             n.points = 1000, range = bounds)
+ }

> take_sm <- with(nhanes,
+                 compute_smoother(sys_mean, dia_mean, taking_aspirin))
> nott_sm <- with(nhanes,
+                 compute_smoother(sys_mean, dia_mean, !taking_aspirin))
```

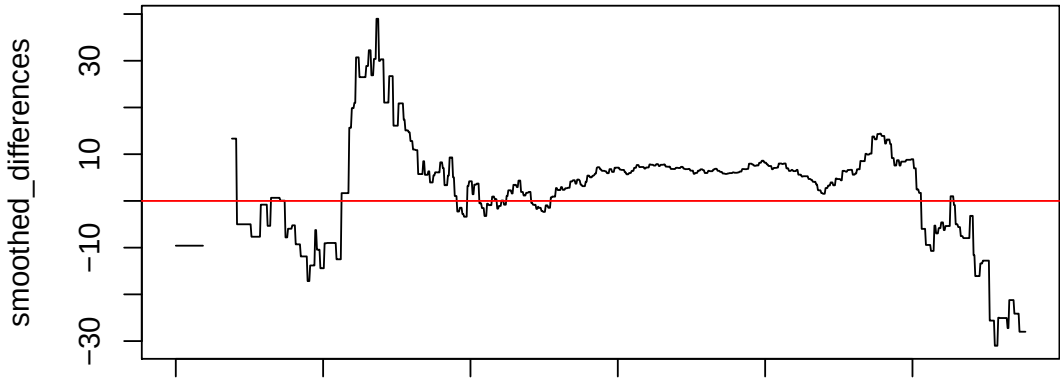

take_sm$y
120
160
200
0
20
40
60
80
100
take_sm$x
Taking
Not taking

Difference Function

We probably want to know if

$$\mu_1(x) > \mu_0(x) \Rightarrow \mu_1(x) - \mu_0(x) > 0$$

```
> smoothed_differences <- take_sm$y - nott_sm$y
```

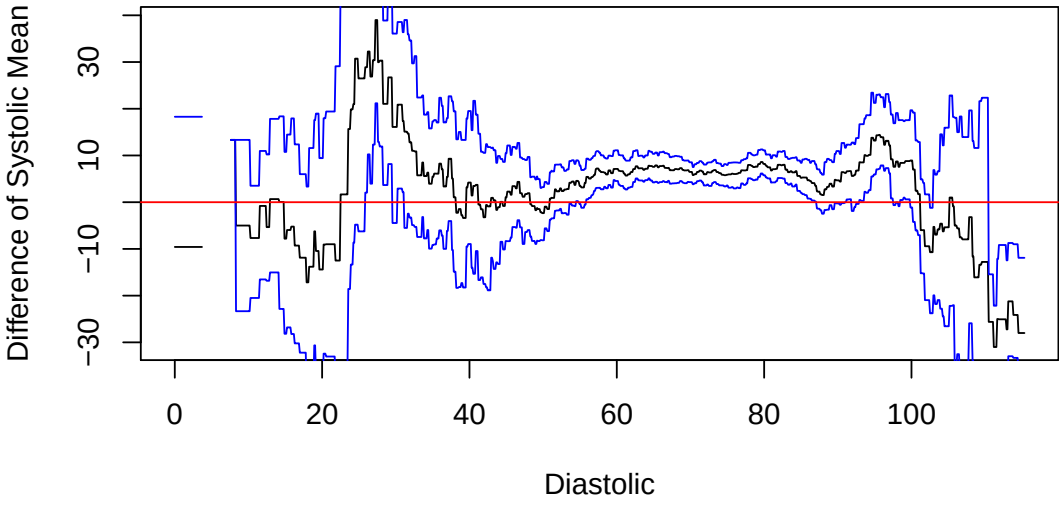


Confidence intervals for differences

```
> mu_diff <- function(data, index) {
+   xstar <- data[index, ]
+
+   s1 <- with(xstar,
+              compute_smoother(sys_mean, dia_mean, taking_aspirin))
+
+   s2 <- with(xstar,
+              compute_smoother(sys_mean, dia_mean, !taking_aspirin))
+
+   return(s1$y - s2$y)
+ }
```

```
> mu_boot <- boot(nhanes,
+   mu_diff,
+   strata = nhanes$taking_aspirin, R = 100)
> point_ci <- apply(mu_boot$t, 2,
+                   quantile, probs = c(0.025, 0.975), na.rm = TRUE)
```

95% confidence CIs for difference of mean fns



Smoothed Mean Function Summary

- Trying to estimate the function μ in

$$Y = \mu(x) + \epsilon, \mathsf{E}(\epsilon = 0) \iff \mathsf{E}(Y \mid x) = \mu(x)$$

- Used **smoothed joint and marginal densities** to create a **smoothed conditional density** $\hat{f}(y \mid x)$.
- Used the smooth conditional to estimate $\hat{\mu}(x) = \int y \hat{f}(y \mid x)\, dy$ (Nadaraya-Watson)
- Saw that NW is example of **linear smoother** (we'll see more later), bias and variance depend on h and K
- Added uncertainty using bootstrap (could have assumed constant variance as well)