Chapter 15

Estimating Distributions and Densities

We have spent a lot of time looking at how to estimate expectations (which is regression). We have also seen how to estimate variances, by turning it into a problem about expectations. We could extend the same methods to looking at higher moments — if you need to find the conditional skewness or kurtosis functions[1], you can tackle that in the same way as finding the conditional variance. But what if we want to look at the whole distribution?

You've already seen the parametric solution to the problem in earlier statistics courses: posit a parametric model for the density (Gaussian, Student's t, exponential, gamma, beta, Pareto, ...) and estimate the parameters. Maximum likelihood estimates are generally consistent and efficient for such problems. Chapter 14 reminded us of how this machinery can be extended to multivariate data. But suppose you don't have any particular parametric density family in mind, or want to check one — how could we estimate a probability distribution non-parametrically?

15.1 Histograms Revisited

For most of you, making a histogram was probably one of the first things you learned how to do in intro stats (if not before). This is a simple way of estimating a distribution: we split the sample space up into bins, count how many samples fall into each bin, and then divide the counts by the total number of samples. If we hold the bins fixed and take more and more data, then by the law of large numbers we anticipate that the relative frequency for each bin will converge on the bin's probability.

So far so good. But one of the things you learned in intro stats was also to work with probability *density* functions, not just probability *mass* functions. Where do we get pdfs? Well, one thing we could do is to take our histogram estimate, and then say

[1] When you find out what the kurtosis is good for, be sure to tell the world.

that the probability density is uniform within each bin. This gives us a piecewise-constant estimate of the density.

Unfortunately, this isn't going to work — isn't going to converge on the true pdf — unless we can shrink the bins of the histogram as we get more and more data. To see this, think about estimating the pdf when the data comes from any of the standard distributions, like an exponential or a Gaussian. We can approximate the true pdf $f(x)$ to arbitrary accuracy by a piecewise-constant density (indeed, that's what happens every time we plot it on our screens), but, for a fixed set of bins, we can only come so close to the true, continuous density.

This reminds us of our old friend the bias-variance trade-off, and rightly so. If we use a large number of very small bins, the minimum bias in our estimate of any density becomes small, but the variance in our estimates grows. (Why does variance increase?) To make some use of this insight, though, there are some things we need to establish first.

- Is learning the whole distribution non-parametrically even feasible?
- How can we measure error so deal with the bias-variance trade-off?

15.2 "The Fundamental Theorem of Statistics"

Let's deal with the first point first. In principle, something even dumber than shrinking histograms will work to learn the whole distribution. Suppose we have one-dimensional one-dimensional samples $x_1, x_2, \ldots x_n$ with a common cumulative distribution function F. The **empirical cumulative distribution function** on n samples, $\tilde{F}_n(a)$ is

$$\tilde{F}_n(a) \equiv \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{(-\infty,a]}(x_i) \tag{15.1}$$

In words, this is just the fraction of the samples which are $\leq a$. Then the **Glivenko-Cantelli theorem** says

$$\max_a |\tilde{F}_n(a) - F(a)| \to 0 \tag{15.2}$$

So the empirical CDF converges to the true CDF everywhere; the maximum gap between the two of them goes to zero. Pitman (1979) calls this the "fundamental theorem of statistics", because it says we can learn distributions just by collecting enough data.[2] The same kind of result also holds for higher-dimensional vectors.

[2]Note that for any one, fixed value of a, that $|\tilde{F}_n(a) - F(a)| \to 0$ is just an application of the law of large numbers. The extra work Glivenko and Cantelli did was to show that this held for *infinitely many* values of a at once, so that even if we focus on the biggest gap between the estimate and the truth, that still shrinks with n. We won't go into the details, but here's the basic idea. Fix an $\epsilon > 0$; first show that there is some *finite* set of points on the line, call them $b_1, \ldots b_q$, such that $|\tilde{F}_n(a) - \tilde{F}_n(b_i)| < \epsilon|$ and $|F(a) - F(b_i)| < \epsilon$ for *some* b_i. Next, show that, for large enough n, $|F(b_i) - \tilde{F}_n(b_i)| < \epsilon$ for all the b_i. (This follows from the law of large numbers and the fact that q is finite.) Finally, use the triangle inequality to conclude that, for large enough n, $|\tilde{F}_n(a) - F(a)| < 3\epsilon$. Since ϵ can be made arbitrarily small, the Glivenko-Cantelli theorem follows. (Yes, there are some details I'm glossing over.) This general strategy — combining pointwise

If the Glivenko-Cantelli theorem is so great, why aren't we just content with the empirical CDF? Sometimes we are, but it inconveniently doesn't give us a probability *density*. Suppose that $x_1, x_2, \ldots x_n$ are sorted into increasing order. What probability does the empirical CDF put on the interval (x_i, x_{i+1})? Clearly, zero. (Whereas the interval $[x_i, x_{i+1}]$ gets probability $2/n$.) This *could* be right, but we have centuries of experience now with probability distributions, and this tells us that *pretty often* we can expect to find some new samples between our old ones. So we'd like to get a *non-zero* density between our observations.

Using a uniform distribution within each bin of a histogram doesn't have this issue, but it does leave us with the problem of picking where the bins go and how many of them we should use. Of course, there's nothing magic about keeping the bin size the same and letting the number of points in the bins vary; we could equally well pick bins so they had equal counts.[3] So what should we do?

15.3 Error for Density Estimates

Our first step is to get clear on what we mean by a "good" density estimate. There are three leading ideas:

1. $\int (f(x) - \hat{f}(x))^2 dx$ should be small: the squared deviation from the true density should be small, averaging evenly over all space.

2. $\int |f(x) - \hat{f}(x)| dx$ should be small: minimize the average *absolute*, rather than squared, deviation.

3. $\int f(x) \log \frac{f(x)}{\hat{f}(x)} dx$ should be small: the average log-likelihood ratio should be kept low.

Option (1) is reminiscent of the MSE criterion we've used in regression. Option (2) looks at what's called the L_1 or **total variation** distance between the true and the estimated density. It has the nice property that $\frac{1}{2}\int |f(x) - \hat{f}(x)| dx$ is exactly the maximum error in our estimate of the probability of *any* set. Unfortunately it's a bit tricky to work with, so we'll skip it here. (But see Devroye and Lugosi (2001)). Finally, minimizing the log-likelihood ratio is intimately connected to maximizing

convergence theorems with approximation arguments — forms the core of what's called **empirical process theory**, which underlies the consistency of basically all the non-parametric procedures we've seen. If this line of thought is at all intriguing, the closest thing to a gentle introduction is Pollard (1989).

[3] A specific idea for how to do this is sometimes called a $k-d$ tree. We have d random variables and want a joint density for all of them. Fix an ordering of the variables Start with the first variable, and find the thresholds which divide it into k parts with equal counts. (Usually but not always $k = 2$.) Then sub-divide each part into k equal-count parts on the *second* variable, then sub-divide each of those on the third variable, etc. After splitting on the d^{th} variable, go back to splitting on the first, until no further splits are possible. With n data points, it takes about $\log_k n$ splits before coming down to individual data points. Each of these will occupy a cell of some volume. Estimate the density on that cell as one over that volume. Of course it's not strictly necessary to keep refining all the way down to single points.

the likelihood. We will come back to this (§15.6), but, like most texts on density estimation, we will give more attention to minimizing (1), because it's mathematically tractable.

Notice that

$$\int (f(x)-\hat{f}(x))^2 dx = \int f^2(x)dx - 2\int \hat{f}(x)f(x)dx + \int \hat{f}^2(x)dx \tag{15.3}$$

The first term on the right hand side doesn't depend on the estimate $\hat{f}(x)$ at all, so we can ignore it for purposes of optimization. The third one only involves $\hat{f}$, and is just an integral, which we can do numerically. That leaves the middle term, which involves both the true and the estimated density; we can approximate it by

$$-\frac{2}{n}\sum_{i=1}^{n}\hat{f}(x_i) \tag{15.4}$$

The reason we can do this is that, by the Glivenko-Cantelli theorem, integrals over the true density are approximately equal to sums over the empirical distribution.

So our final error measure is

$$-\frac{2}{n}\sum_{i=1}^{n}\hat{f}(x_i) + \int \hat{f}^2(x)dx \tag{15.5}$$

In fact, this error measure does *not* depend on having one-dimension data; we can use it in any number of dimensions.[4] For purposes of cross-validation (you knew that was coming, right?), we can estimate $\hat{f}$ on the training set, and then restrict the sum to points in the testing set.

15.3.1 Error Analysis for Histogram Density Estimates

We now have the tools to do most of the analysis of histogram density estimation. (We'll do it in one dimension for simplicity.) Choose our favorite location x, which lies in a bin whose boundaries are x_0 and $x_0 + h$. We want to estimate the density at x, and this is

$$\hat{f}_n(x) = \frac{1}{h}\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{(x_0,x_0+h]}(x_i) \tag{15.6}$$

Let's call the sum, the number of points in the bin, b. It's a random quantity, $B \sim \text{Binomial}(n,p)$, where p is the true probability of falling into the bin, $p = F(x_0+h) - F(x_0)$. The mean of B is np, and the variance is $np(1-p)$, so

$$\mathbf{E}\left[\hat{f}_n(x)\right] = \frac{1}{nh}\mathbf{E}[B] \tag{15.7}$$

$$= \frac{n[F(x_0+h)-F(x_0)]}{nh} \tag{15.8}$$

$$= \frac{F(x_0+h)-F(x_0)}{h} \tag{15.9}$$

[4] Admittedly, in high-dimensional spaces, doing the final integral can become numerically challenging.

and the variance is

$$\begin{aligned}
\mathrm{Var}\left[\hat{f}_n(x)\right] &= \frac{1}{n^2 h^2}\mathrm{Var}[B] && (15.10)\\
&= \frac{n[F(x_0+h)-F(x_0)][1-F(x_0+h)+F(x_0)]}{n^2 h^2} && (15.11)\\
&= \mathbf{E}\left[\hat{f}_n(x)\right]\frac{1-F(x_0+h)+F(x_0)}{nh} && (15.12)
\end{aligned}$$

If we let $h \to 0$ as $n \to \infty$, then

$$\mathbf{E}\left[\hat{f}_h(x)\right] \to \lim_{h\to 0}\frac{F(x_0+h)-F(x_0)}{h} = f(x_0) \tag{15.13}$$

since the pdf is the derivative of the CDF. But since x is between x_0 and x_0+h, $f(x_0) \to f(x)$. So if we use smaller and smaller bins as we get more data, the histogram density estimate is unbiased. We'd also like its variance to shrink as the same grows. Since $1-F(x_0+h)+F(x_0) \to 1$ as $h \to 0$, to get the variance to go away we need $nh \to \infty$.

To put this together, then, our first conclusion is that histogram density estimates will be consistent when $h \to 0$ but $nh \to \infty$ as $n \to \infty$. The bin-width h needs to shrink, but slower than n^{-1}.

At what rate should it shrink? Small h gives us low bias but (as you can verify from the algebra above) high variance, so we want to find the trade-off between the two. One can calculate the bias at x from our formula for $\mathbf{E}\left[\hat{f}_h(x)\right]$ through a somewhat lengthy calculus exercise, analogous to what we did for kernel smoothing in Chapter 4[5]; the upshot is that the integrated squared bias is

$$\int \left(f(x)-\mathbf{E}\left[\hat{f}_h(x)\right]\right)^2 dx = \frac{h^2}{12}\int \left(f'(x)\right)^2 dx + o(h^2) \tag{15.14}$$

We already got the variance at x, and when we integrate that over x we find

$$\int \mathrm{Var}\left[\hat{f}_h(x)\right] dx = \frac{1}{nh} + o(n^{-1}) \tag{15.15}$$

So the total integrated squared error is

$$\mathrm{ISE} = \frac{h^2}{12}\int \left(f'(x)\right)^2 dx + \frac{1}{nh} + o(h^2) + o(n^{-1}) \tag{15.16}$$

Differentiating this with respect to h and setting it equal to zero, we get

$$\frac{h_{\mathrm{opt}}}{6}\int \left(f'(x)\right)^2 dx = \frac{1}{nh_{\mathrm{opt}}^2} \tag{15.17}$$

[5] You need to use the intermediate value theorem multiple times; see for instance Wasserman (2006, sec. 6.8).

$$h_{\text{opt}} = \left(\frac{6}{\int \left(f'(x)\right)^2 dx} \right)^{1/3} n^{-1/3} = O(n^{-1/3}) \tag{15.18}$$

So we need narrow bins if the density changes rapidly ($\int \left(f'(x)\right)^2 dx$ is large), and wide bins if the density is relatively flat. No matter how rough the density, the bin width should shrink like $O(n^{-1/3})$. Plugging that rate back into the equation for the ISE, we see that it is $O(n^{-2/3})$.

It turns out that if we pick h by cross-validation, then we attain this optimal rate in the large-sample limit. By contrast, if we *knew* the correct parametric form and just had to estimate the parameters, we'd typically get an error decay of $O(n^{-1})$. This is substantially faster than histograms, so it would be nice if we could make up some of the gap, without having to rely on parametric assumptions.

15.4 Kernel Density Estimates

It turns out that one can improve the convergence rate, as well as getting smoother estimates, but using kernels. The **kernel density estimate** is

$$\widehat{f}_h(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x - x_i}{h} \right) \tag{15.19}$$

where K is a kernel function such as we encountered when looking at kernel regression. (The factor of $1/h$ inside the sum is so that $\widehat{f}_h$ will integrate to 1; we could have included it in both the numerator and denominator of the kernel regression formulae, but then it would've just canceled out.) As before, h is the **bandwdith** of the kernel. We've seen typical kernels in things like the Gaussian. One advantage of using them is that they give us a smooth density everywhere, unlike histograms, and in fact we can even use them to estimate the derivatives of the density, should that be necessary.[6]

15.4.1 Analysis of Kernel Density Estimates

How do we know that kernels will in fact work? Well, let's look at the mean and variance of the kernel density estimate at a particular point x, and use Taylor's theorem

[6]The advantage of histograms is that they're computationally and mathematically simpler.

on the density.

$$\begin{aligned}
\mathbf{E}\left[\widehat{f}_h(x)\right] &= \frac{1}{n}\sum_{i=1}^{n}\mathbf{E}\left[\frac{1}{h}K\left(\frac{x-X_i}{h}\right)\right] && (15.20)\\
&= \mathbf{E}\left[\frac{1}{h}K\left(\frac{x-X}{h}\right)\right] && (15.21)\\
&= \int \frac{1}{h}K\left(\frac{x-t}{h}\right)f(t)dt && (15.22)\\
&= \int K(u)f(x-hu)du && (15.23)\\
&= \int K(u)\left[f(x)-huf'(x)+\frac{h^2u^2}{2}f''(x)+o(h^2)\right]du && (15.24)\\
&= f(x)+\frac{h^2f''(x)}{2}\int K(u)u^2du+o(h^2) && (15.25)\\
& && (15.26)
\end{aligned}$$

because, by definition, $\int K(u)du = 1$ and $\int uK(u)du = 0$. If we call $\int K(u)u^2du = \sigma_K^2$, then the bias of the kernel density estimate is

$$\mathbf{E}\left[\widehat{f}_h(x)\right] - f(x) = \frac{h^2\sigma_K^2 f''(x)}{2} + o(h^2) \quad (15.27)$$

So the bias will go to zero if the bandwidth h shrinks to zero. What about the variance? Use Taylor's theorem again:

$$\begin{aligned}
\mathrm{Var}\left[\widehat{f}_h(x)\right] &= \frac{1}{n}\mathrm{Var}\left[\frac{1}{h}K\left(\frac{x-X}{h}\right)\right] && (15.28)\\
&= \frac{1}{n}\left[\mathbf{E}\left[\frac{1}{h^2}K^2\left(\frac{x-X}{h}\right)\right] - \left(\mathbf{E}\left[\frac{1}{h}K\left(\frac{x-X}{h}\right)\right]\right)^2\right] && (15.29)\\
&= \frac{1}{n}\left[\int \frac{1}{h^2}K^2\left(\frac{x-t}{h}\right)dt - \left[f(x)+O(h^2)\right]^2\right] && (15.30)\\
&= \frac{1}{n}\left[\int \frac{1}{h}K^2(u)f(x-hu)du - f^2(x)+O(h^2)\right] && (15.31)\\
&= \frac{1}{n}\left[\int \frac{1}{h}K^2(u)\left(f(x)-huf'(x)\right)du - f^2(x)+O(h)\right] && (15.32)\\
&= \frac{f(x)}{hn}\int K^2(u)du + O(1/n) && (15.33)
\end{aligned}$$

This will go to zero if $nh \to \infty$ as $n \to \infty$. So the conclusion is the same as for histograms: h has to go to zero, but slower than $1/n$.

Since the expected squared error at x is the bias squared plus the variance,

$$\frac{h^4\sigma_K^4(f''(x))^2}{4} + \frac{f(x)}{hn}\int K^2(u)du + \text{small} \quad (15.34)$$

the expected integrated squared error is

$$\text{ISE} \approx \frac{h^4 \sigma_K^4}{4} \int (f''(x))^2 dx + \frac{\int K^2(u) du}{nh} \tag{15.35}$$

Differentiating with respect to h for the optimal bandwidth h_{opt}, we find

$$h_{\text{opt}}^3 \sigma_K^4 \int (f''(x))^2 dx = \frac{\int K^2(u) du}{n h_{\text{opt}}^2} \tag{15.36}$$

$$h_{\text{opt}} = \left(\frac{\int K^2(u) du}{\sigma_K^4 \int (f''(x))^2 dx} \right)^{1/5} n^{-1/5} = O(n^{-1/5}) \tag{15.37}$$

That is, the best bandwidth goes to zero like one over the fifth root of the number of sample points. Plugging this into Eq. 15.35, the best ISE $= O(n^{-4/5})$. This is better than the $O(n^{-2/3})$ rate of histograms, but still includes a penalty for having to figure out what kind of distribution we're dealing with. Remarkably enough, using cross-validation to pick the bandwidth gives near-optimal results.[7]

As an alternative to cross-validation, or at least a starting point, one can use Eq. 15.37 to show that the optimal bandwidth for using a Gaussian kernel to estimate a Gaussian distribution is $1.06\sigma n^{-1/5}$, with σ being the standard deviation of the Gaussian. This is sometimes called the **Gaussian reference rule** or the **rule-of-thumb** bandwidth. When you call `density` in R, this is basically what it does.

Yet another technique is the **plug-in method**. Eq. 15.37 calculates the optimal bandwidth from the second derivative of the true density. This doesn't help if we don't know the density, but it becomes useful if we have an initial density estimate which isn't too bad. In the plug-in method, we start with an initial bandwidth (say from the Gaussian reference rule) and use it to get a preliminary estimate of the density. Taking that crude estimate and "plugging it in" to Eq. 15.37 gives us a new bandwidth, and we re-do the kernel estimate with that new bandwidth. Iterating this a few times is optional but not uncommon.

15.4.2 Joint Density Estimates

The discussion and analysis so far has been focused on estimating the distribution of a one-dimensional variable. Just as kernel regression can be done with multiple input variables (§4.3), we can make kernel density estimates of joint distributions. We simply need a kernel for the vector:

$$\widehat{f}(\vec{x}) = \frac{1}{n} \sum_{i=1}^{n} K(\vec{x} - \vec{x_i}) \tag{15.38}$$

[7] Substituting Eq. 15.37 into Eq. 15.35 gives a squared error of $1.25 n^{-4/5} \sigma_K^{4/5} \left(\int (f''(x))^2 dx \right)^{1/5} \left(\int K^2(u) du \right)^{4/5}$. The only two parts of this which depend on the kernel are σ_K and $\int K^2(u) du$. This is the source of the (correct) folklore that the choice of kernel is less important than the choice of bandwidth.

One could use any multivariate distribution as the kernel (provided it is centered and has finite covariance). Typically, however, just as in smoothing, one uses a product kernel, i.e., a product of one-dimensional kernels,

$$K(\vec{x} - \vec{x_i}) = K_1(x^1 - x_i^1)K_2(x^2 - x_i^2)\ldots K_d(x^d - x_i^d)\,, \tag{15.39}$$

Doing this requires a bandwidth for each coordinate, so the over-all form of the joint PDF estimate is

$$\widehat{f}(\vec{x}) = \frac{1}{n\prod_{j=1}^{d} h_j}\sum_{i=1}^{n}\prod_{j=1}^{d} K_j\left(\frac{x^j - x_i^j}{h_j}\right) \tag{15.40}$$

Going through a similar analysis for d-dimensional data shows that the ISE goes to zero like $O(n^{-4/(4+d)})$, and again, if we use cross-validation to pick the bandwidths, asymptotically we attain this rate. Unfortunately, if d is large, this rate becomes very slow — for instance, if $d = 24$, the rate is $O(n^{-1/7})$. There is simply no universally good way to figure out high-dimensional distributions from scratch; either we make strong parametric assumptions, which could be badly wrong, or we accept a potentially very slow convergence.

15.4.3 Categorical and Ordered Variables

Estimating probability mass functions with discrete variables can be straightforward: there are only a finite number of values, and so one just counts how often they occur and takes the relative frequency. If one has a discrete variable X and a continuous variable Y and one wants a joint distribution, one could just get a separate density for Y for each value of x, and tabulate the probabilities for x.

In principle, this will work, but it can be practically awkward if the number of levels for the discrete variable is large compared to the number of samples. Moreover, for the joint distribution problem, it has us estimating completely separate distributions for Y for every x, without any sharing of information between them. It would seem more plausible to smooth those distributions towards each others. To do this, we need kernels for discrete variables.

Several sets of such kernels have been proposed. The most straightforward, however, are the following. If X is a categorical, unordered variable with c possible values, then, for $0 \leq h < 1$,

$$K(x_1, x_2) = \begin{cases} 1-h & x_1 = x_2 \\ h/c & x \neq x_i \end{cases} \tag{15.41}$$

is a valid kernel. For an ordered x,

$$K(x_1, x_2) = \binom{c}{|x_1 - x_2|} h^{|x_1 - x_2|}(1-h)^{c-|x_1-x_2|} \tag{15.42}$$

where $|x_1 - x_2|$ should be understood as just how many levels apart x_1 and x_2 are. As $h \to 0$, both of these become indicators, and return us to simple relative frequency counting. Both of these are implemented in `np`.

15.4.4 Practicalities

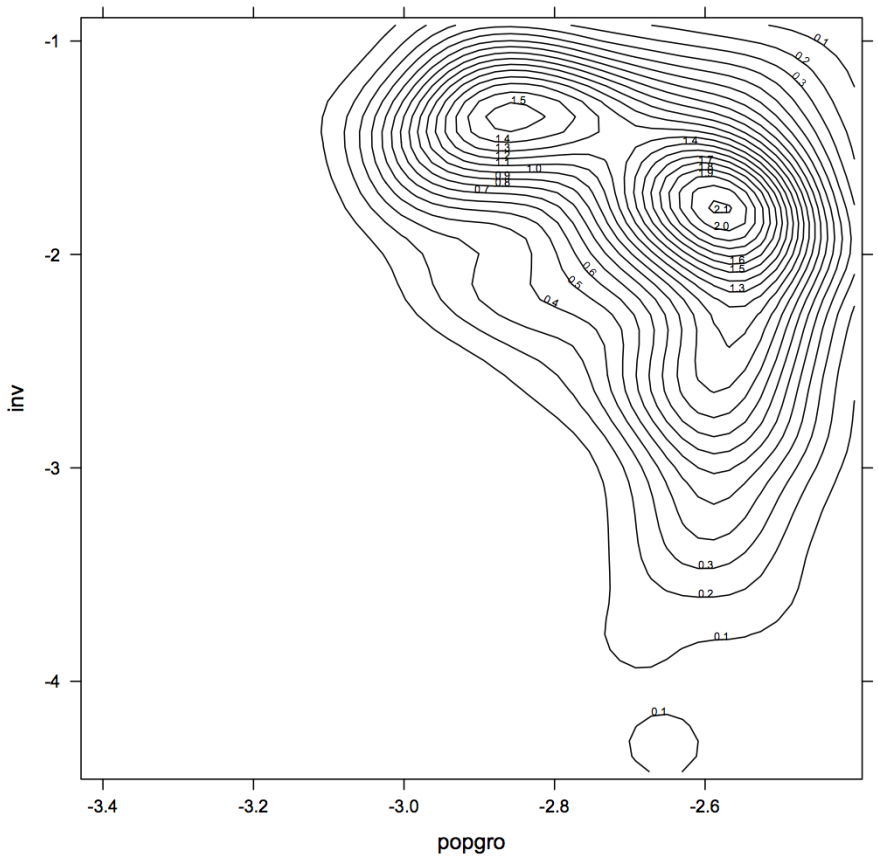
The standard R function `density` implements one-dimensional kernel density estimation, defaulting to Gaussian kernels with the rule-of-thumb bandwidth. There are some options for doing cleverer bandwidth selection, including a plug-in rule. (See the help file.)

For more sophisticated methods, and especially for more dimensions, you'll need to use other packages. The `np` package estimates joint densities using the `npudens` function. (The `u` is for "unconditional".) This has the same sort of automatic bandwidth selection as `npreg`, using cross-validation. Other packages which do kernel density estimation include `KernSmooth` and `sm`.

15.4.5 Kernel Density Estimation in R: An Economic Example

The data set `oecdpanel`, in the `np` library, contains information about much the same sort of variables at the Penn World Tables data you worked with in the homework, over much the same countries and years, but with some of the variables pre-transformed, with identifying country information removed, and slightly different data sources. See `help(oecdpanel)` for details.

Here's an example of using `npudens` with variables from the `oecdpanel` data set [[you saw in the homework]]. We'll look at the joint density of `popgro` (the logarithm of the population growth rate) and `inv` (the logarithm of the investment rate). Figure 15.1 illustrates how to call the command, and a useful trick where we get `np`'s plotting function to do our calculations for us, but then pass the results to a different graphics routine. (See `help(npplot)`.) The distribution we get has two big modes, one at a comparatively low population growth rate (≈ -2.9 — remember this is logged so it's not actually a shrinking population) and high investment (≈ -1.5), and the other at a lower rate of investment (≈ -2) and higher population grow (≈ -2.6). There is a third, much smaller mode at high population growth (≈ -2.7) and very low investment (≈ -4).




```
library(np)
data(oecdpanel)
popinv <- npudens(~popgro+inv, data=oecdpanel)
fhat <- plot(popinv,plot.behavior="data")
fhat <- fhat$d1
library(lattice)
contourplot(fhat$dens~fhat$eval$Var1*fhat$eval$Var2,cuts=20,
  xlab="popgro",ylab="inv",labels=list(cex=0.5))
```

Figure 15.1: Gaussian kernel estimate of the joint distribution of logged population growth rate (`popgro`) and investment rate (`inv`). Notice that `npudens` takes a formula, but that there is no dependent variable on the left-hand side of the $\sim$. With objects produced by the `np` library, one can give the plotting function the argument `plot.behavior` — the default is `plot`, but if it's set to `data` (as here), it calculates all the information needed to plot and returns a separate set of objects, which can be plotted in other functions. (The value `plot-data` does both.) See `help(npplot)` for more.

15.5 Conditional Density Estimation

In addition to estimating marginal and joint densities, we will often want to get conditional densities. The most straightforward way to get the density of Y given X, $f_{Y|X}(y|x)$, is

$$\widehat{f}_{Y|X}(y|x) = \frac{\widehat{f}_{X,Y}(x,y)}{\widehat{f}_X(x)} \tag{15.43}$$

i.e., to estimate the joint and marginal densities and divide one by the other.

To be concrete, let's suppose that we are using a product kernel to estimate the joint density, and that the marginal density is consistent with it:

$$\widehat{f}_{X,Y}(x,y) = \frac{1}{nh_X h_Y}\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_X}\right) K_Y\left(\frac{y-y_i}{h_Y}\right) \tag{15.44}$$

$$\widehat{f}_X(x) = \frac{1}{nh_X}\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_X}\right) \tag{15.45}$$

Thus we need to pick two bandwidths, h_X and h_Y, one for each variable.

This might seem like a solved problem — we just use cross-validation to find h_X and h_Y so as to minimize the integrated squared error for $\widehat{f}_{X,Y}$, and then plug in to Equation 15.43. However, this is a bit hasty, because the optimal bandwidths for the *joint* density are not necessarily the optimal bandwidths for the *conditional* density. An extreme but easy to understand example is when Y is actually independent of X. Since the density of Y given X is just the density of Y, we'd be best off just ignoring X by taking $h_X = \infty$. (In practice, we'd just use a very big bandwidth.) But if we want to find the joint density, we would not want to smooth X away completely like this.

The appropriate integrated squared error measure for the conditional density is

$$\int dx f_X(x) \int dy \left(f_{Y|X}(y|x) - \widehat{f}_{Y|X}(y|x)\right)^2 \tag{15.46}$$

and this is what we want to minimize by picking h_X and h_Y. The cross-validation goes as usual.

One nice, and quite remarkable, property of cross-validation for conditional density estimation is that it can detect and exploit conditional independence. Say that $X = (U, V)$, and that Y is independent of U given V — symbolically, $Y \perp\!\!\!\perp U|V$. Then $f_{Y|U,V}(y|u,v) = f_{y|v}(y|v)$, and we should just ignore U in our estimation of the conditional density. It turns out that when cross-validation is used to pick bandwidths for conditional density estimation, $\widehat{h_U} \to \infty$ when $Y \perp\!\!\!\perp U|V$, but not otherwise (Hall *et al.*, 2004). In other words, cross-validation will automatically detect which variables are irrelevant, and smooth them away.

15.5.1 Practicalities and a Second Example

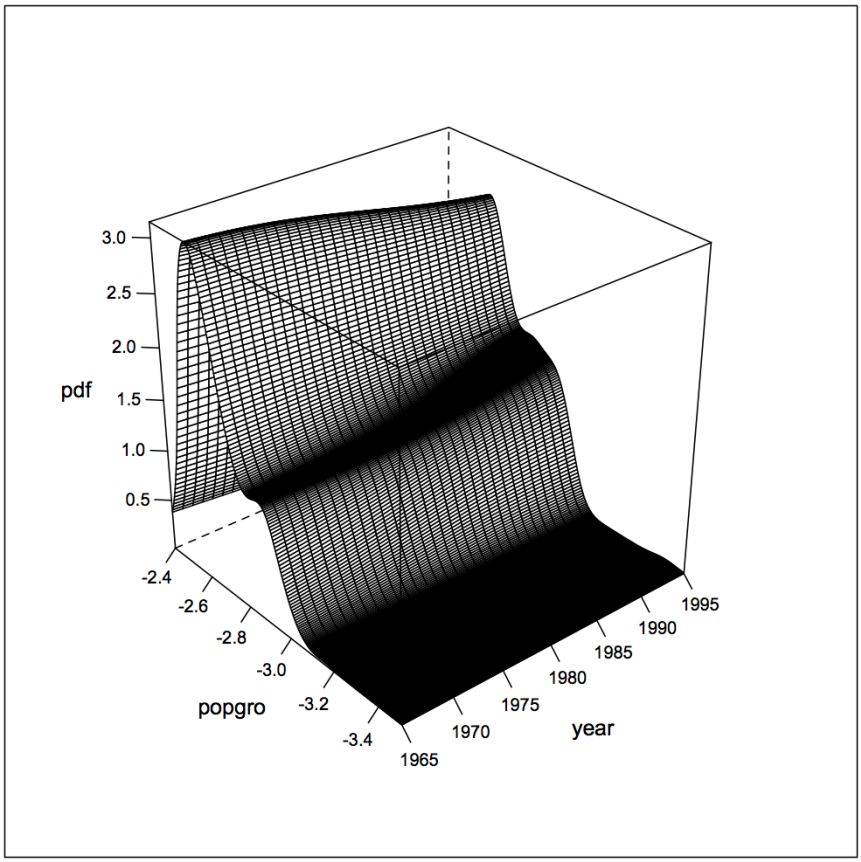
The `np` package implements kernel conditional density estimation through the function `npcdens`. The syntax is pretty much exactly like that of `npreg`, and indeed we can think of estimating the conditional density as a sort of regression, where the dependent variable is actually a distribution.

To give a concrete example, let's look at how the distribution of countries' population growth rates has changed over time, using the `oecdpanel` data (Figure 15.2). The selected bandwidth for `year` is 10, while that for `popgro` is 0.048. (Note that `year` is being treated as a continuous variable.)

You can see from the figure that the mode for population growth rates is towards the high end of observed values, but the mode is shrinking and becoming less pronounced over time. The distribution in fact begins as clearly bimodal, but the smaller mode at the lower growth rate turns into a continuous "shoulder". Overall, Figure 15.2 shows a trend for population growth rates to shrink over time, and for the distribution of growth rates to become less dispersed.

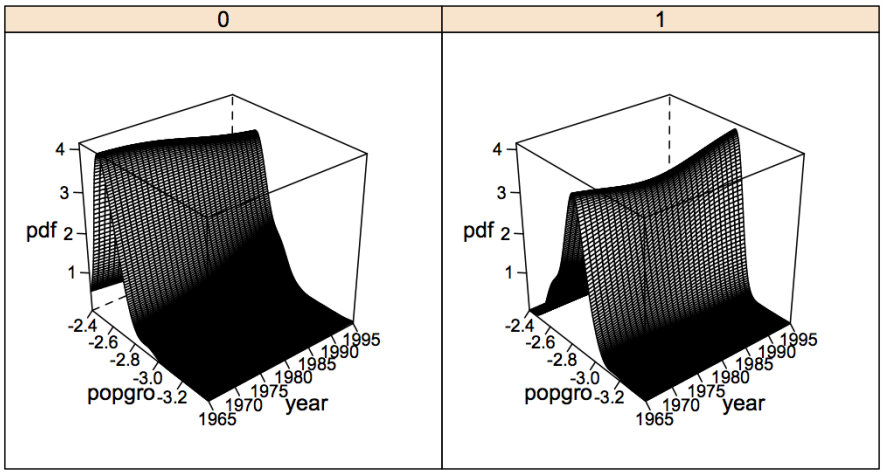
Let's expand on this point. One of the variables in `oecdpanel` is `oecd`, which is 1 for countries which are members of the Organization for Economic Cooperation and Development, and 0 otherwise. The OECD countries are basically the "developed" ones (stable capitalist democracies). We can include OECD membership as a conditioning variable for population growth (we need to use a categorical-variable kernel), and look at the combined effect of time and development (Figure 15.3).

What the figure shows is that OECD and non-OECD countries both have unimodal distributions of growth rates. The mode for the OECD countries has become sharper, but the value has decreased. The mode for non-OECD countries has also decreased, while the distribution has become more spread out, mostly by having more probability of lower growth rates. (These trends have continued since 1995.) In words, despite the widespread contrary impression, population growth has actually been slowing for decades in both rich and poor countries.




```
pop.cdens <- npcdens(popgro ~ year,data=oecdpanel)
plotting.grid <- expand.grid(year=seq(from=1965,to=1995,by=1),
  popgro=seq(from=-3.5,to=-2.4,length.out=300))
fhat <- predict(pop.cdens,newdata=plotting.grid)
wireframe(fhat~plotting.grid$year*plotting.grid$popgro,
  scales=list(arrows=FALSE),xlab="year",ylab="popgro",zlab="pdf")
```

Figure 15.2: Conditional density of logarithmic population growth rates as a function of time.




```
pop.cdens.o <- npcdens(popgro~year+factor(oecd),data=oecdpanel)
oecd.grid <- expand.grid(year=seq(from=1965,to=1995,by=1),
  popgro=seq(from=-3.4,to=-2.4,length.out=300),
  oecd=unique(oecdpanel$oecd))
fhat <- predict(pop.cdens.o,newdata=oecd.grid)
wireframe(fhat~oecd.grid$year*oecd.grid$popgro|oecd.grid$oecd,
  scales=list(arrows=FALSE),xlab="year",ylab="popgro",zlab="pdf")
```

Figure 15.3: Conditional density of population growth rates given year and OECD membership. The left panel is countries not in the OECD, the right is ones which are.

15.6 More on the Expected Log-Likelihood Ratio

I want to say just a bit more about the expected log-likelihood ratio $\int f(x) \log \frac{f(x)}{\widehat{f}(x)} dx$. More formally, this is called the **Kullback-Leibler divergence** or **relative entropy** of $\widehat{f}$ from f, and is also written $D(f||\widehat{f})$. Let's expand the log ratio:

$$D(f||\widehat{f}) = -\int f(x) \log \widehat{f}(x) dx + \int f(x) \log f(x) dx \tag{15.47}$$

The second term does not involve the density estimate, so it's irrelevant for purposes of optimizing over $\widehat{f}$. (In fact, we're just subtracting off the entropy of the true density.) Just as with the squared error, we could try approximating the integral with a sum:

$$\int f(x) \log \widehat{f}(x) dx \approx \frac{1}{n} \sum_{i=1}^{n} \log \widehat{f}(x_i) \tag{15.48}$$

which is just the log-likelihood per observation. Since we know and like maximum likelihood methods, why not just use this?

Well, let's think about what's going to happen if we plug in the kernel density estimate:

$$\frac{1}{n} \sum_{i=1}^{n} \log \left(\frac{1}{nh} \sum_{j=1}^{n} K\left(\frac{x_j - x_i}{h} \right) \right) = -\log nh + \frac{1}{n} \sum_{i=1}^{n} \log \left(\sum_{j=1}^{n} K\left(\frac{x_j - x_i}{h} \right) \right) \tag{15.49}$$

If we take h to be very small, $K(\frac{x_j - x_i}{h}) \approx 0$ unless $x_j = x_i$, so the over-all likelihood becomes

$$\approx -\log nh + \log K(0) \tag{15.50}$$

which goes to $+\infty$ as $h \to 0$. So if we want to maximize the likelihood of a kernel density estimate, we *always* want to make the bandwidth as small as possible. In fact, the limit is to say that the density is

$$\tilde{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \delta(x - x_i) \tag{15.51}$$

where δ is the Dirac delta function.[8] Of course this is just what we'd get if we took the empirical CDF "raw".

What's gone wrong here? Why is maximum likelihood failing us? Well, it's doing exactly what we asked it to: to find the distribution where the observed sample is as probable as possible. Giving any probability to *un*-observed values can only come

[8] Recall that the delta function is defined by how it integrates with other functions: $\int \delta(x) f(x) dx = f(0)$. You can imagine $\delta(x)$ as zero everywhere except at the origin, where it has an infinitely tall, infinitely narrow spike, the area under the spike being one. If you are suspicious that this is really a valid function, you're right; strictly speaking it's just a linear operator on actual functions. We can however approximate it as the limit of well-behaved functions. For instance, take $\delta_h(x) = 1/h$ when $x \in [-h/2, h/2]$ with $\delta_h(x) = 0$ elsewhere, and let h go to zero. This is, of course, where we came in.

at the expense of the probability of observed values, so Eq. 15.51 really is the unrestricted maximum likelihood estimate of the distribution. Anything else imposes some restrictions or constraints which don't, strictly speaking, come from the data. However, those restrictions are what let us generalize to new data, rather than just memorizing the training sample.

One way out of this is to use the *cross-validated* log-likelihood to pick a bandwidth, i.e., to restrict the sum in Eq. 15.48 to running over the testing set only. This way, very small bandwidths don't get an unfair advantage for concentrating around the training set. (If the test points are in fact all very close to the training points, then small bandwidths get a *fair* advantage.) This is in fact the default procedure in the `np` package, through the `bwmethod` option (`"cv.ml"` vs. `"cv.ls"`).

15.7 Simulating from Density Estimates

15.7.1 Simulating from Kernel Density Estimates

There are times when one wants to draw a random sample from the estimated distribution. This is easy with kernel density estimates, because each kernel is itself a probability density, generally a very tractable one. The general pattern goes as follows. Suppose the kernel is Gaussian, that we have scalar observations $x_1, x_2, \ldots x_n$, and the selected bandwidth is h. Then we pick an integer `i` uniformly at random from 1 to n, and invoke `rnorm(1,x[i],h)`.[9] Using a different kernel, we'd just need to use the random number generator function for the corresponding distribution.

We can see that this works, i.e., that it gives the right distribution, with just a little math. A kernel $K(x, x_i h)$ with bandwidth h and center x_i is a probability density function. The probability the density estimate gives to any set A is just an integral:

$$\begin{aligned}
\widehat{F}(A) &= \int_A \widehat{f}(x) dx && (15.52)\\
&= \int_A \frac{1}{n}\sum_{i=1}^{n} K(x, x_i, h) dx && (15.53)\\
&= \frac{1}{n}\sum_{i=1}^{n} \int_A K(x, x_i, h) dx && (15.54)\\
&= \frac{1}{n}\sum_{i=1}^{n} C(A, x_i, h) && (15.55)
\end{aligned}$$

introducing C to stand for the probability distribution corresponding to the kernel. The simulation procedure works if the probability that the simulated value $\tilde{X}$ falls into A matches this. To generate $\tilde{X}$, we first pick a random data point, which really

[9] In fact, if we want to draw a sample of size q, `rnorm(q,sample(x,q,replace=TRUE),h)` will work in R — it's important though that sampling be done *with* replacement.

means picking a random integer J, uniformly from 1 to n. Then

$$\begin{aligned}
\Pr\left(\tilde{X} \in A\right) &= \mathbf{E}\left[\mathbf{1}_A(\tilde{X})\right] && (15.56)\\
&= \mathbf{E}\left[\mathbf{E}\left[\mathbf{1}_A(\tilde{X})|J\right]\right] && (15.57)\\
&= \mathbf{E}\left[C(A, x_J, h)\right] && (15.58)\\
&= \frac{1}{n}\sum_{i=1}^{n} C(A, x_i, h) && (15.59)
\end{aligned}$$

The first step uses the fact that a probability is the expectation of an indicator function; the second uses the law of total expectation; the last steps us the definitions of C and J, and the distribution of J.

15.7.1.1 Sampling from a Kernel Joint Density Estimate

The procedure given above works with only trivial modification for sampling from a joint, multivariate distribution. If we're using a product kernel, we pick a random data point, and then sample each coordinate independently. The argument for correctness actually goes exactly as before.

15.7.1.2 Sampling from Kernel Conditional Density Estimates

Sampling from a conditional density estimate with product kernels is again straightforward. The one trick is that one needs to do a *weighted* sample of data points. To see why, look at the conditional distribution (not density) function:

$$\begin{aligned}
&\widehat{F}(Y \in A|X = x) && (15.60)\\
&= \int_A \widehat{f}_{Y|X}(y|x)dy\\
&= \int_A \frac{\frac{1}{nh_X h_Y}\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_X}\right)K_Y\left(\frac{y-y_i}{h_Y}\right)}{\widehat{f}_X(x)}dy && (15.61)\\
&= \frac{1}{nh_X h_Y \widehat{f}_X(x)}\int_A \sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_X}\right)K_Y\left(\frac{y-y_i}{h_Y}\right)dy && (15.62)\\
&= \frac{1}{nh_X h_Y \widehat{f}_X(x)}\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_X}\right)\int_A K_Y\left(\frac{y-y_i}{h_Y}\right)dy && (15.63)\\
&= \frac{1}{nh_X \widehat{f}_X(x)}\sum_{i=1}^{n} K_X\left(\frac{x-x_i}{h_X}\right)C_Y(A, y_i, h_Y) && (15.64)
\end{aligned}$$

If we select the data point i with a weight proportional to $K_X\left(\frac{x-x_i}{h_X}\right)$, and then generate $\tilde{Y}$ from the K_Y distribution centered at y_i, then, $\tilde{Y}$ will follow the appropriate probability density function.

15.7.2 Sampling from Histogram Estimates

Sampling from a histogram estimate is also simple, but in a sense goes in the opposite order from kernel simulation. We first randomly pick a bin by drawing from a multinomial distribution, with weights proportional to the bin counts. Once we have a bin, we draw from a uniform distribution over its range.

15.7.3 Examples of Simulating from Kernel Density Estimates

To make all this more concrete, let's continue working with the `oecdpanel` data. Section 15.4.5 shows the joint pdf estimate for the variables `popgro` and `inv` in that data set. These are the logarithms of the population growth rate and investment rate. Undoing the logarithms and taking the density,

```
popinv2 <- npudens(~exp(popgro)+exp(inv),data=oecdpanel)
```

gives Figure 15.4.

Let's abbreviate the actual (not logged) population growth rate as X and the actual (not logged) investment rate as Y in what follows.

Since this is a joint distribution, it implies a certain expected value for Y/X, the ratio of investment rate to population growth rate[10]. Extracting this by direct calculation from `popinv2` would not be easy; we'd need to do the integral

$$\int_{x=0}^{1}\int_{y=0}^{1}\frac{y}{x}\widehat{f}_{X,Y}(x,y)dydx \tag{15.65}$$

To find $\mathbf{E}[Y/X]$ by simulation, however, we just need to generate samples from the joint distribution, say $(\tilde{X}_1,\tilde{Y}_1),(\tilde{X}_2,\tilde{Y}_2),\ldots(\tilde{X}_T,\tilde{Y}_T)$, and average:

$$\frac{1}{T}\sum_{i=1}^{T}\frac{\tilde{Y}_i}{\tilde{X}_i} = \tilde{g}_T \stackrel{T\to\infty}{\longrightarrow} \mathbf{E}\left[\frac{Y}{X}\right] \tag{15.66}$$

where the convergence happens because that's the law of large numbers. If the number of simulation points T is big, then $\tilde{g}_T \approx \mathbf{E}[Y/X]$. How big do we need to make T? Use the central limit theorem:

$$\tilde{g}_T \rightsquigarrow \mathcal{N}(\mathbf{E}[Y/X], \mathrm{Var}[\tilde{g}_1]/\sqrt{T}) \tag{15.67}$$

How do we find the variance $\mathrm{Var}[\tilde{g}_1]$? We approximate it by simulating.

Code Example 31 is a function which draws a sample from the fitted kernel density estimate. First let's check that it works, by giving it something easy to do, namely reproducing the means, which we *can* work out:

```
> mean(exp(oecdpanel$popgro))
[1] 0.06930789
```

[10]Economically, we might want to know this because it would tell us about how quickly the capital stock per person grows.

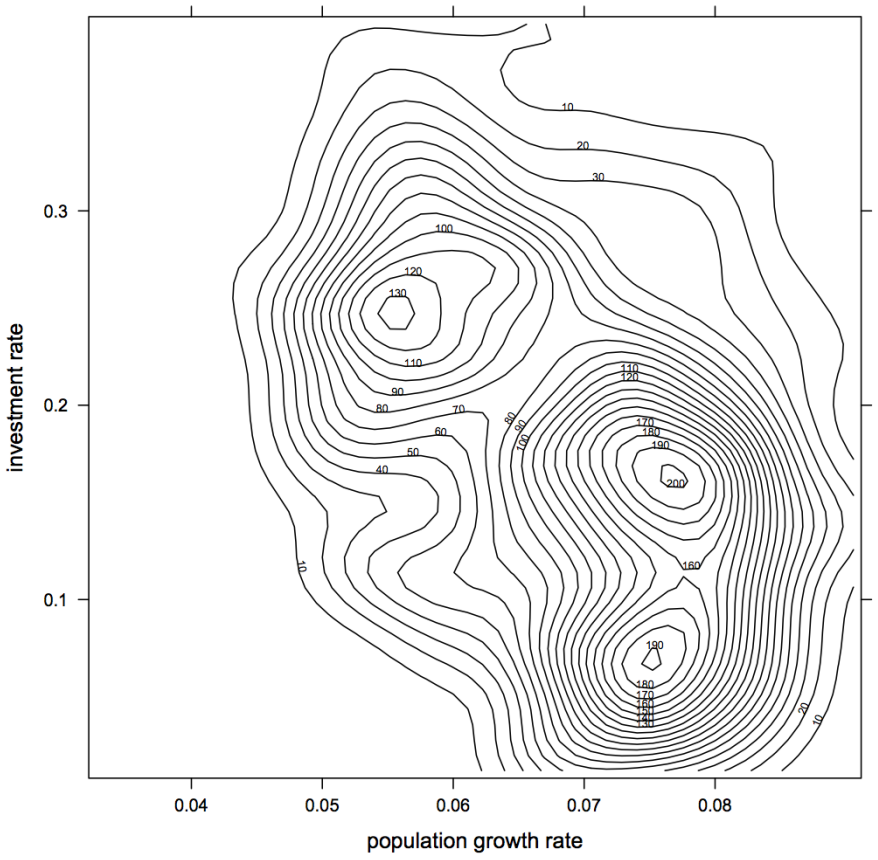


Figure 15.4: Gaussian kernel density estimate for the un-logged population growth rate and investment rate.

```
rpopinv <- function(n) {
  n.train <- length(popinv2$dens)
  ndim <- popinv2$ndim
  points <- sample(1:n.train,size=n,replace=TRUE)
  z <- matrix(0,nrow=n,ncol=ndim)
  for (i in 1:ndim) {
    coordinates <- popinv2$eval[points,i]
    z[,i] <- rnorm(n,coordinates,popinv2$bw[i])
  }
  colnames(z) <- c("pop.growth.rate","invest.rate")
  return(z)
}
```

Code Example 31: Sampling from the fitted kernel density estimate `popinv2`. Can you see how to modify it to sample from other bivariate density estimates produced by `npudens`? From higher-dimensional distributions? Can you replace the `for` loop with less iterative code?

```
> mean(exp(oecdpanel$inv))
[1] 0.1716247
> colMeans(rpopinv(200))
pop.growth.rate     invest.rate
     0.06865678      0.17623612
```

This is pretty satisfactory for only 200 samples, so the simulator seems to be working. Now we just use it:

```
> z <- rpopinv(2000)
> mean(z[,"invest.rate"]/z[,"pop.growth.rate"])
[1] 2.597916
> sd(z[,"invest.rate"]/z[,"pop.growth.rate"])/sqrt(2000)
[1] 0.0348991
```

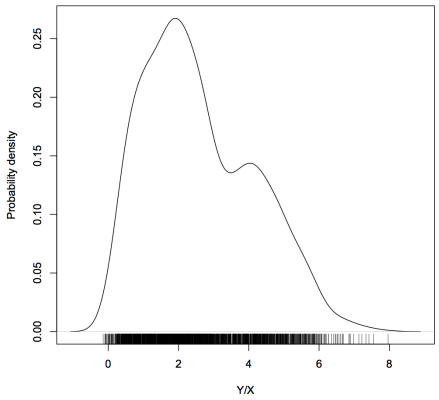
So this tells us that $\mathbf{E}[Y/X] \approx 2.59$, with a standard error of ± 0.035.

Suppose we want not the mean of Y/X but the median?

```
> median(z[,"invest.rate"]/z[,"pop.growth.rate"])
[1] 2.31548
```

Getting the whole distribution of Y/X is not much harder (Figure 15.5). Of course complicated things like distributions converge more slowly than simple things like means or medians, so we want might want to use more than 2000 simulated values for the distribution. Alternately, we could repeat the simulation many times, and look at how much variation there is from one realization to the next (Figure 15.6).

Of course, if we are going to do multiple simulations, we could just average them together. Say that $\tilde{g}_T^{(1)}$, $\tilde{g}_T^{(2)}$, $\ldots$ $\tilde{g}_T^{(s)}$ are estimates of our statistic of interest from s




```
YoverX <- z[,"invest.rate"]/z[,"pop.growth.rate"]
plot(density(YoverX),xlab="Y/X",ylab="Probability density",main="")
rug(YoverX,side=1)
```

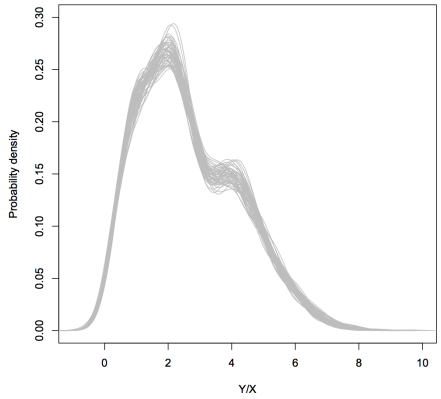
Figure 15.5: Distribution of Y/X implied by the joint density estimate `popinv2`.

independent realizations of the model, each of size T. We can just combine them into one grand average:

$$\tilde{g}_{s,T} = \frac{1}{s}\sum_{i=1}^{s} \tilde{g}_T^{(1)} \tag{15.68}$$

As an average of IID quantities, the variance of $\tilde{g}_{s,T}$ is $1/s$ times the variance of $\tilde{g}_T^{(1)}$.

By this point, we are getting the sampling distribution of the density of a nonlinear transformation of the variables in our model, with no more effort than calculating a mean.




```
plot(0,xlab="Y/X",ylab="Probability density",type="n",xlim=c(-1,10),
     ylim=c(0,0.3))
one.plot <- function() {
  zprime <- rpopinv(2000)
  YoverXprime <- zprime[,"invest.rate"]/zprime[,"pop.growth.rate"]
  density.prime <- density(YoverXprime)
  lines(density.prime,col="grey")
}
invisible(replicate(50,one.plot()))
```

Figure 15.6: Showing the sampling variability in the distribution of Y/X by "overplotting". Each line is a distribution from an estimated sample of size 2000, as in Figure 15.5; here 50 of them are plotted on top of each other. The thickness of the bands indicates how much variation there is from simulation to simulation at any given value of Y/X. (Setting the `type` of the initial `plot` to `n`, for "null", creates the plotting window, axes, legends, etc., but doesn't actually plot anything.)

15.8 Exercises

To think through, not to hand in.

1. Reproduce Figure 15.4?
2. Qualitatively, is this compatible with Figure 15.1?
3. How could we use `popinv2` to calculate a joint density for `popgro` and `inv` (not `exp(popgro)` and `exp(inv)`)?
4. Should the density `popinv2` implies for those variables be the same as what we'd get from directly estimating their density with kernels?