7 Approximating distributions

Posterior distributions can be quite complicated. In this chapter I will first look at how we can approximate a distribution using a Taylor expansion. I will then introduce the method of kernel density estimation, which is used to estimate a distribution given samples drawn from it. A well-known example of this is histograms. This technique will be indispensable when we come to using Monte Carlo sampling methods in the next two chapters.

7.1 The quadratic approximation

7.1.1 One dimension

In the real world there are relatively few problems for which either the full posterior or a marginalization thereof is a standard distribution. Posterior PDFs are sometimes quite complicated and difficult to summarize, as we shall see in chapter 9. Nonetheless, if the PDF is dominated by a single mode, and that mode is quite "peaky", then it is often sufficient to approximate the posterior around that peak. The more informative the data, the greater the amount of probability concentrated around the peak, and the better such an approximation will be. Here we look at a convenient approximation for such cases.

To do this, I use the logarithm of the posterior PDF

$$\Phi = \ln P(\theta \,|\, D) \tag{7.1}$$

where, for now, θ is a scalar. As the logarithm is a strictly monotonic function of its argument, turning points in the PDF correspond to turning points in its logarithm. Whereas, for informative data, the probability density will span many order of magnitudes, its logarithm will vary more slowly with θ. This suggest that a reasonable approximation for Φ can be achieved by expanding it around its mode $\hat{\theta}$ with a second-order Taylor expansion:

$$\begin{aligned}
\Phi &\simeq \Phi(\hat{\theta}) + (\theta - \hat{\theta}) \frac{d\Phi}{d\theta}\Big|_{\hat{\theta}} + \frac{1}{2}(\theta - \hat{\theta})^2 \frac{d^2\Phi}{d\theta^2}\Big|_{\hat{\theta}} \\
&\text{where} \quad \frac{d\Phi}{d\theta}\Big|_{\hat{\theta}} = 0 \quad \text{so} \\
\Phi - \Phi(\hat{\theta}) &\simeq \frac{1}{2}(\theta - \hat{\theta})^2 \frac{d^2\Phi}{d\theta^2}\Big|_{\hat{\theta}} . \qquad (7.2)
\end{aligned}$$

Taking the exponential of this and using equation 7.1 we get

$$P(\theta \,|\, D) \;\simeq\; A \exp\left(\frac{1}{2}(\theta - \hat{\theta})^2 \frac{d^2\Phi}{d\theta^2}\bigg|_{\hat{\theta}}\right) \tag{7.3}$$

which is a Gaussian with mean $\hat{\theta}$ and variance

$$\sigma^2 \;=\; \left(-\frac{d^2\Phi}{d\theta^2}\bigg|_{\hat{\theta}}\right)^{-1} \tag{7.4}$$

as the term $A = P(\hat{\theta}\,|\,D)$ is independent of θ. The second derivative is negative at a mode (maximum), ensuring that the variance is positive.

The quadratic approximation approximates a general PDF as a Gaussian, which is convenient because this has nice properties. For example, if we have a posterior which we cannot (or do not want to) normalize, then we cannot calculate its standard deviation to use as an uncertainty measure. But with the quadratic approximation we can calculate this relatively easily. Whether this is a good approximation naturally depends on how accurate the second-order Taylor expansion is in the case at hand.

Example: exponential distribution

Suppose we have a likelihood function

$$P(x \,|\, \theta) \;=\; \frac{1}{\theta}\exp(-x/\theta) \tag{7.5}$$

for $x \geq 0$ and $\theta \geq 0$, and a prior $P(\theta)$ which is uniform for $\theta \geq 0$ and zero otherwise. We measure N data points $\{x_i\}$. What is the quadratic approximation of the posterior PDF for θ?

Assuming the data are measured independently, the likelihood of the data is the product of the individual likelihoods. The posterior is proportional to the product of this with the prior, so the posterior is

$$P(\theta \,|\, \{x_i\}) \;\propto\; \frac{1}{\theta^N}\exp(-N\bar{x}/\theta) \tag{7.6}$$

where $\bar{x}$ is the mean of the data and the (missing) normalization constant is independent of the model parameter θ. Taking the natural logarithm and differentiating twice with respect to θ we get

$$\Phi \;=\; -N\ln\theta - \frac{N\bar{x}}{\theta} + \text{constant} \tag{7.7}$$

$$\frac{d\Phi}{d\theta} \;=\; -\frac{N}{\theta} + \frac{N\bar{x}}{\theta^2} \tag{7.8}$$

$$\frac{d^2\Phi}{d\theta^2} \;=\; \frac{N}{\theta^2} - \frac{2N\bar{x}}{\theta^3}. \tag{7.9}$$

From this we can see that the maximum (zero first derivative) is $\hat{\theta} = \bar{x}$, and the second derivative at this value is $-N/\bar{x}^2$. There is only one maximum. Hence we can approximate this posterior around its maximum as a Gaussian with mean $\bar{x}$ and variance $\bar{x}^2/N$.

7.1.2 Two dimensions

The quadratic approximation, like the Taylor expansion, generalizes to higher dimensions. Suppose we have the two-dimensional posterior $P(a, b \,|\, D)$. The second-order Taylor expansion at the maximum $(\hat{a}, \hat{b})$ of the log posterior (so the first derivatives are zero) is

$$\Phi \;\simeq\; \Phi(\hat{a}, \hat{b}) \;+\; \frac{1}{2}\left[(a-\hat{a})^2\frac{\partial^2\Phi}{\partial a^2} \;+\; (b-\hat{b})^2\frac{\partial^2\Phi}{\partial b^2} \;+\; 2(a-\hat{a})(b-\hat{b})\frac{\partial^2\Phi}{\partial a \partial b}\right] \tag{7.10}$$

where all the derivatives are evaluated at the maximum. The quantity in the square brackets can be written as a matrix multiplication

$$R \;=\; \begin{pmatrix} a-\hat{a} & b-\hat{b} \end{pmatrix} \begin{pmatrix} v_{aa} & v_{ab} \\ v_{ab} & v_{bb} \end{pmatrix} \begin{pmatrix} a-\hat{a} \\ b-\hat{b} \end{pmatrix} \tag{7.11}$$

where

$$v_{aa} \;=\; \frac{\partial^2\Phi}{\partial a^2}, \quad v_{bb} \;=\; \frac{\partial^2\Phi}{\partial b^2}, \quad v_{ab} \;=\; \frac{\partial^2\Phi}{\partial a \partial b}. \tag{7.12}$$

Using the same logic as with one parameter, the approximation to the posterior around its maximum can be written as

$$P(a, b \,|\, D) \;\simeq\; A \exp\left(\frac{1}{2}R\right) \tag{7.13}$$

which, given the definition of R, is the equation for a two-dimensional Gaussian in (a, b) with mean $(\hat{a}, \hat{b})$ and covariance matrix

$$\begin{aligned} \mathrm{Cov}(a, b) \;\equiv\; \begin{pmatrix} \sigma_a^2 & \sigma_{ab}^2 \\ \sigma_{ab}^2 & \sigma_b^2 \end{pmatrix} &= -\begin{pmatrix} v_{aa} & v_{ab} \\ v_{ab} & v_{bb} \end{pmatrix}^{-1} \\ &= \frac{1}{v_{aa}v_{bb} - v_{ab}^2}\begin{pmatrix} -v_{bb} & v_{ab} \\ v_{ab} & -v_{aa} \end{pmatrix}. \end{aligned} \tag{7.14}$$

From this we can read off the values of the variances σ_a^2 and σ_b^2 of the two parameters, and their covariance σ_{ab}^2.

7.1.3 Higher dimensions

The quadratic approximation can be extended to higher dimensions. Consider the J-dimensional parameter vector $\boldsymbol{\theta}$ with maximum $\hat{\boldsymbol{\theta}}$, i.e. $\boldsymbol{\nabla}\Phi(\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}) = 0$. Then

$$\Phi(\boldsymbol{\theta}) \;\simeq\; \Phi(\hat{\boldsymbol{\theta}}) + \frac{1}{2}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^{\mathsf{T}}[\boldsymbol{\nabla\nabla}\Phi(\hat{\boldsymbol{\theta}})](\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}). \tag{7.15}$$

The posterior is a multivariate Gaussian with covariance matrix equal to the negative of the inverse of the matrix of second derivatives of the logarithm of the posterior. This is quite a mouthful and is more easily expressed as

$$\mathrm{Cov}(\boldsymbol{\theta})_{\hat{\boldsymbol{\theta}}} \;=\; -[\boldsymbol{\nabla\nabla}\Phi(\hat{\boldsymbol{\theta}})]^{-1}. \tag{7.16}$$

The J-dimensional square matrix $\boldsymbol{\nabla\nabla}\Phi(\hat{\boldsymbol{\theta}})$ is called the *Hessian matrix*, the elements of which are $\partial^2\Phi/\partial\theta_i\partial\theta_j$ evaluated at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. This is a symmetric matrix (as the covariance is symmetric).[1] We see that the width of the distribution – the uncertainty in the parameters – is determined by the second derivatives of the logarithm of the posterior, which are a measure of its curvature.

7.2 Density estimation

Up until this point we have been able to plot arbitrary PDFs simply by evaluating them on a pre-defined grid. But as we shall see in the next chapter this it not always practicable. So in preparation for the following two chapters consider the following situation: given a sample of N data points $\{x_i\}$ drawn from an unknown PDF, how can we find (or rather estimate) that PDF?

One approach is to adopt a parametrized model for the PDF, for example a Gaussian or Poisson distribution. We saw in section 4.4 how we could then maximize the product of probability densities with respect to the model parameters in order to find the best fitting parameters.

Unfortunately, in many situations we will not be able to find a parametrized model that approximates the data sufficiently well. We then need to take a non-parametric approach. Suppose the data are univariate. If they are also discrete, and we have many samples at each value, then we could just stack up the data into a histogram. If the data are real, each value is unique: our data set is essentially a string of delta functions. We can still construct a histogram by averaging over narrow bins of the data, although there are issues with this, as we shall see below. A histogram is in fact a simple case of a more general method called *density estimation*, which we shall investigate below.

Sometimes we do have an equation for the density distribution. In section 6.3 this distribution was a bivariate posterior, and we plotted it and calculated summary statistics simply by evaluating it on a grid. But if the distribution has more dimensions or a complex variation, then evaluation on a grid will be computationally expensive and may not capture all the variations. In that case it is preferable to draw samples from the distribution and then use density estimation to approximate the distribution. We will see how to do this sampling in chapter 8 using Monte Carlo methods.

I use several R packages below. `density` and `persp` are all in the base R packages. `image.plot` is in the package `fields`. Venables & Ripley (2002) introduce the package `MASS`. This contains the R methods `truehist` and `kde2d` as well as the data set `geyser`, all of which I will use below. My example R scripts are also based on scripts in Venables & Ripley.

7.2.1 Histograms

A histogram estimates the density by aggregating data points into bins of pre-specified width and location. These bins are adjacent, equal-width (usually), uniform functions over

[1] This is a symmetric matrix only if the order of differentiation does not matter, for which the functions must be continuous functions. But they must be continuous for the Taylor expansion to be valid.

the variable x. Examples are shown in figure 7.1. Let the bin width be λ. The number of points that fall into each bin is summed up and this is taken as the frequency (counts) of data over that bin; call this c_i for bin i. We can normalize the histogram by dividing the counts in each bin by the area under the histogram (which is $\lambda \sum c_i$) so that the histogram now integrates to one. Each bin then reports the density – the fraction of objects per unit x, which is $f_i = c_i/(\lambda \sum c_i)$ – under the assumption that the density is constant over the bin. Both representations are of course discontinuous at the bin boundaries. We can appreciate the difference between counts and density by considering a uniform distribution. If we half the bin width, the counts c_i will half, but the densities f_i will stay constant. The density will still integrate to one, because although there are now twice as many bins, they are half as wide.

Histograms are quick and easy, and useful when we have a lot of data such that we have well-populated bins. However, they can be sensitive to the placement and width of the bins, as we see in the following example.

The data set `geyser` in the R package `MASS` contains data on the eruptions of the Old Faithful geyser in Yellowstone National Park in the USA. It records two variables: the eruption duration and the time until the next eruption (both in minutes). We might be interested in finding out whether there is a relationship between these two variables, or whether the eruptions fall into groups in one or both of the variables. Here we look at the duration variable and use histograms to find its distribution. The following R code investigates two things. It first plots histograms with different bin sizes, figure 7.1. We see that increasing the number of bins increases the amount of detail, but taken to an extreme we would end up with just one or zero objects per bin, which doesn't tell us anything beyond the original data. This is another example of the bias-variance trade-off discussed in section 4.8. Broad bins have small variance because they average over a lot of points, but a large bias because they are not a very accurate representation of the density at all positions across of the bin. Narrow bins are the opposite: the bias is small, but the variance is large because small changes in bin size or position result in big changes in the histogram.

The second part of the code below plots histograms all with the same bin width, but with different bin centers, shifting them by a fifth of the bin centre each time. The result is shown in figure 7.2. We see how sensitive the results are to this shift. We are unlikely to have any information telling us which set of bin centres is the most appropriate.

Let us now see how we can generalize the idea of histograms.

R file: `histograms.R`

```
##### Investigate histograms
##### Modified from the example given by Venables & Ripley (2002)

library(MASS) # for truehist and geyser data
attach(geyser)
# Look at sensitivity to bin width
pdf("histograms1.pdf", 12, 8)
par(mfrow=c(2,3), mar=c(3.5,3.5,1.5,0.5), oma=c(0.5,0.5,0.5,0.5),
    mgp=c(2.2,0.8,0), cex=1.0)
for(h in c(2, 1, 1/2, 1/4, 1/8, 1/16)) {
 truehist(duration, h=h, x0=0.0, xlim=c(0,6), ymax=0.8, ylab="density",
```

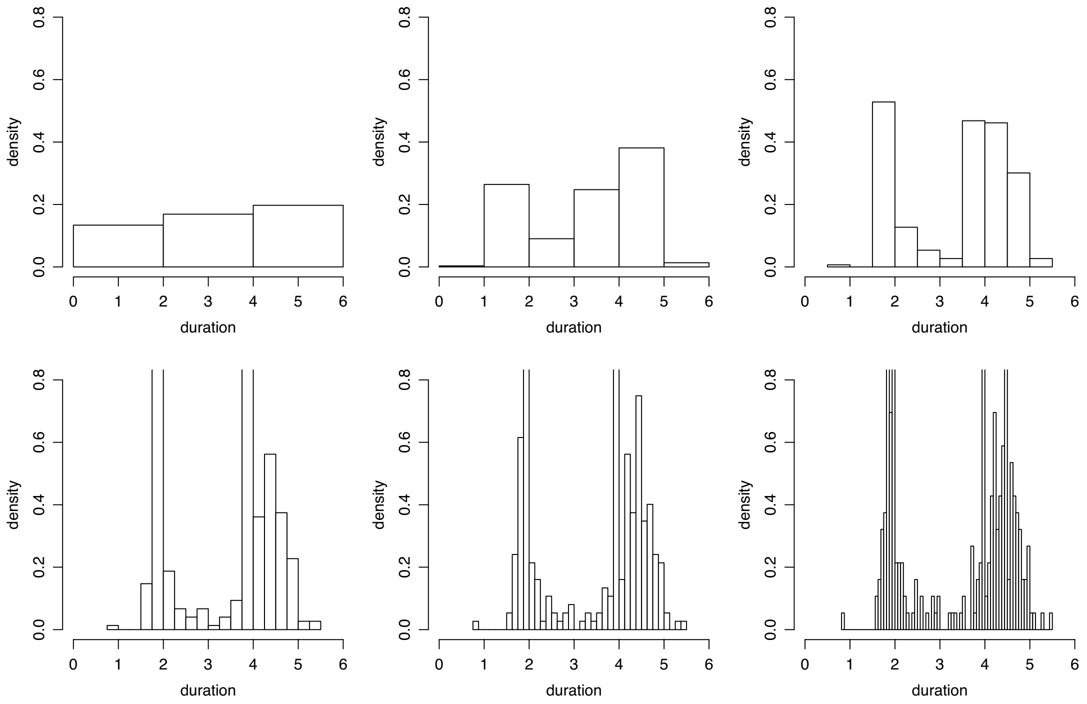

Fig. 7.1 Six histograms of the same data (geyser duration) with bin sizes ranging from 2 to $1/16$ in steps of factors of $1/2$. In the bottom row the highest peaks extend off the scale, the highest being for the narrowest bins (up to around 3.0 at a duration of about 4).

```
          col="white")
}
dev.off()
# Look at sensitivity to placing of bin centres
pdf("histograms2.pdf", 12, 8)
par(mfrow=c(2,3), mar=c(3.5,3.5,1.5,0.5), oma=c(0.5,0.5,0.5,0.5),
    mgp=c(2.2,0.8,0), cex=1.0)
binwidth <- 0.5
for(x0 in binwidth*(0:5)/5) {
  truehist(duration, h=binwidth, x0=x0, xlim=c(0,6), ymax=0.8,
           ylab="density", col="white")
}
dev.off()
detach(geyser)
```

7.2.2 Kernel density estimation

A histogram models the density by assuming it to be constant over some finite range of the variable, namely the bin width. Every point is assigned to a single bin, and the density

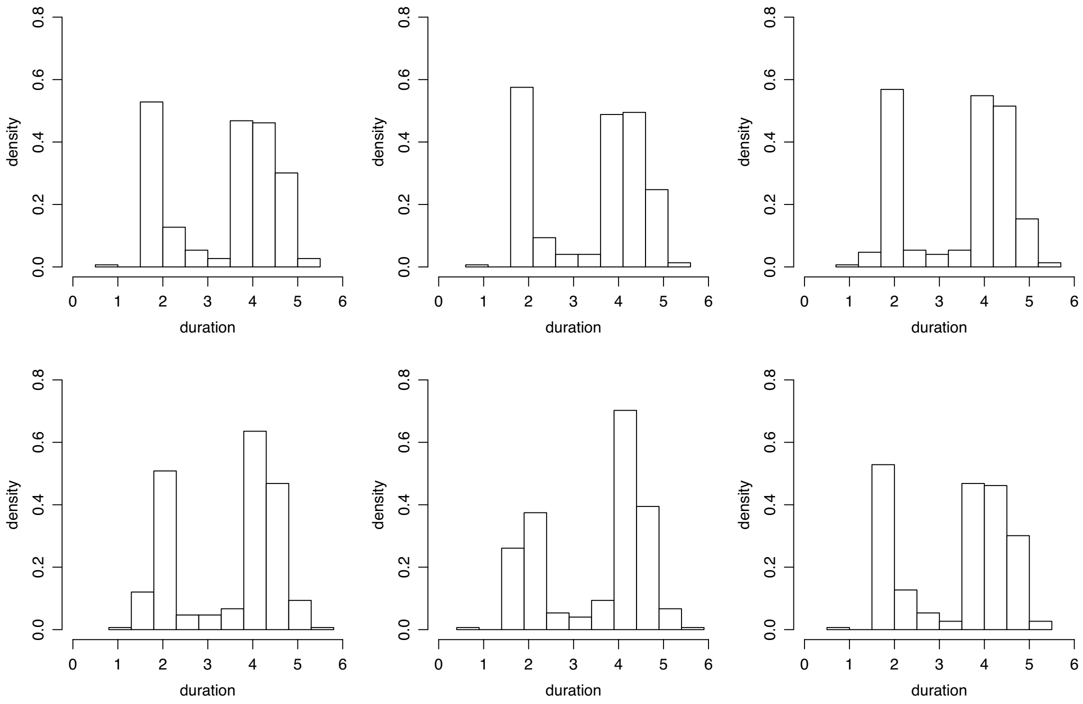

Fig. 7.2 Histograms of the same data (geyser duration) with the same bin size ($h = 1/2$), but bin centers offset by $1/5$ of the bin width each time.

is the average number of points in the bin. We could do something more sophisticated than assuming a constant density. This generalization leads us to the idea of *kernel density estimation*. In this context a kernel is a weighted function of data. The weights are usually a function of the distance from the point at which the kernel is being evaluated.

Given a one-dimensional data set $\{x_i\}$ of size N, we can estimate the density at an arbitrary point x as being

$$f(x) = \frac{1}{N\lambda}\sum_{i=1}^{N} K(u_i) \quad \text{where} \quad u_i = \frac{x - x_i}{\lambda}. \tag{7.17}$$

K is the *kernel*, which must be normalized ($\int K(u)du = 1$), and as it is being used for density estimation it must also be non-negative. The bandwidth λ characterizes the length scale over which the kernel operates. We may set this ourselves, or we can estimate it from the data (we'll see one way of doing this later in this section). It need not be fixed. $N\lambda$ is a normalization constant which ensures $f(x)$ is a density. We are free to choose any appropriate kernel. A simple choice is the rectangular kernel

$$K(u) = \begin{cases} 1 & \text{if } |u| < 1/2 \\ 0 & \text{otherwise.} \end{cases} \tag{7.18}$$

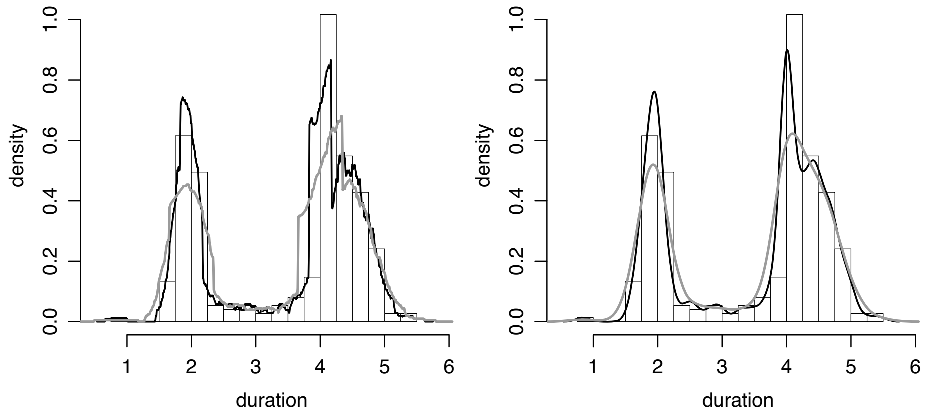

Fig. 7.3 Kernel density estimation (using `density` in R) on the geyser duration data set using a rectangular kernel (left) and a Gaussian kernel (right) with bandwidth parameter 0.1 (in black) and 0.2 (in grey). A histogram of the same data with bin width 0.25 is overplotted in both cases for comparison.

This is similar to the histogram in the sense that the density at point x is determined just by the number of points that fall within the bin, i.e. within $\pm\lambda/2$ of x. But unlike the histogram this kernel can be computed at any point x, and not just at a pre-defined set of bin centres separated by the bin width. We can think of moving a box of width λ along the x-axis and computing the density continuously. This is known as a *moving average*. Whereas the densities in each histogram bin are independent of their neighbouring bins (they share no data), here they are not. The resulting density estimate is shown for two different choices of λ in the left panel of figure 7.3.

As with the histogram, we see that the rectangular kernel does not give a very smooth variation of the density. There are other kernels which generally give better estimates of the density. Here I mention just two.

The first is a *Gaussian kernel*. We put down a Gaussian at x (i.e. with mean x) and make a weighted sum of all the data $\{x_i\}$, where the weights are given by the value of the Gaussian function. The standard deviation of the Gaussian is the bandwidth λ. The kernel is

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}u^2\right) \tag{7.19}$$

with the density at any point x again given by equation 7.17. An example is shown in the right panel of figure 7.3. It tends to produce smoother density estimates than the rectangular kernel, for a given bandwidth.

Another commonly used kernel is the *k-nearest neighbours kernel*, whereby k is a fixed positive integer. At a specified point x we identify the k nearest neighbours among the $\{x_i\}$ and compute the length $\Delta(x)$ that they span: that is, $\Delta(x)$ is the distance between the most distant neighbour above x and most distant one below x. The kernel size therefore varies with x, in order to include a fixed number of neighbours. The kernel is just $K(u) = k$, and it follows from equation 7.17 that the density function is

$$f(x) = \frac{k}{N\Delta(x)} . \tag{7.20}$$

The smoothness of the density curve produced by a kernel density estimate is determined by the size of the kernel, or by the number of nearest neighbours. The larger the kernel (or the smaller the number of nearest neighbours) the smoother the curve will be. This may smooth out "real" features. If the kernel is too small (or we have too few neighbours), the variations in density may just be artefacts (noise) from having too few data in the density estimate at each point.

Note an important difference between the two types of kernel. The rectangular and Gaussian kernels have a fixed bandwidth, and so include a variable number of points that depends on the local density of the data. The nearest neighbour kernel instead includes a fixed number of points, but has a variable bandwidth. If we have large spans of the variable where there are no data, the latter kernel will always give a (non-zero) density estimate, but this may be inaccurate because it relies on distant data points. In regions of high density, the nearest neighbour kernel may be more precise than a fixed-width kernel, because the former will shrink and so will follow density variations more closely.

I will introduce a couple of other kernels in section 12.4 when we look at using them for regression.

The following R code produces the plots in figure 7.3. The density estimation is performed by the function `density`. The parameter `n` in this function is the number of points at which the density is estimated. These points are spread uniformly over the range of the data (although the range can be specified using `from` and `to`). Provided `n` is sufficiently high, its exact choice will have no impact on the appearance of the density estimate. I encourage you to experiment with changing the kernel, specified by `kernel`, and the bandwidth, specified by `bw`. Note that the `bw` parameter may not give a size for the bandwidth which you expect. This is because the functions internal to `density` may scale this by an amount that depends on the choice of kernel. Read the documentation to find out more.

R file: `kde.R`

```
##### Investigate kernel density estimation in 1D
##### Modified from the example given by Venables & Ripley (2002)

library(MASS) # for truehist and geyser data
attach(geyser)
pdf("kde.pdf", 8,4)
par(mfrow=c(1,2), mar=c(3.5,3.0,0.5,0.5), oma=c(0.5,0.5,0.5,0.5),
    mgp=c(2.2,0.8,0), cex=1.0)
truehist(duration, h=0.25, xlim=c(0.5, 6), ymax=1.1, ylab="density",
         col="white", lwd=0.5)
lines(density(duration, kernel="rectangular", bw=0.1, n=2^10), lwd=1.5)
lines(density(duration, kernel="rectangular", bw=0.2, n=2^10), lwd=2,
      col="grey60")
truehist(duration, h=0.25, xlim=c(0.5, 6), ymax=1.1, ylab="density",
         col="white", lwd=0.5)
```

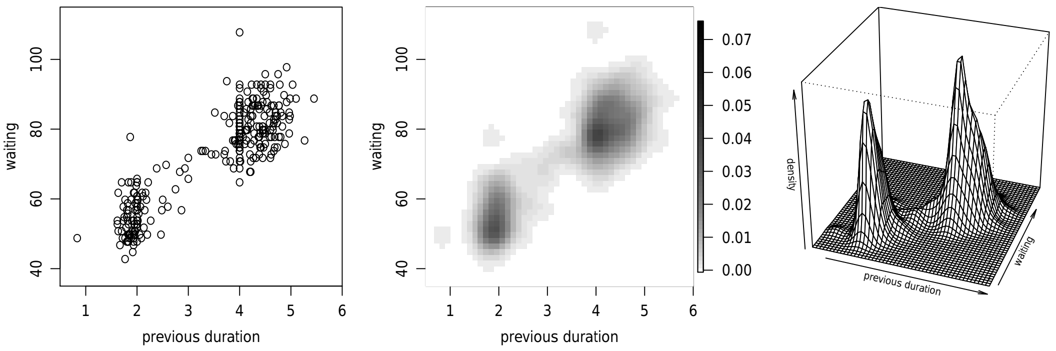

Fig. 7.4 Two-dimensional kernel density estimation with a Gaussian kernel applied to the geyser data. Left: original data. Centre and right: kernel density estimation using a bandwidth of 0.75 for “previous duration” and 5 for “waiting”. The central panel shows the kernel density estimation as a density map, the right panel as a three-dimensional perspective mesh.

```
lines(density(duration, kernel="gaussian",    bw=0.1, n=2^10), lwd=1.5)
lines(density(duration, kernel="gaussian",    bw=0.2, n=2^10), lwd=2,
      col="grey60")
dev.off()
detach(geyser)
```

You may be interested in looking at what the function `density` actually produces. If you do

```
attach(geyser)
mydense <- density(duration, kernel="gaussian", bw=0.2, n=2^10)
```

then `attributes(mydense)` lists all the things computed and stored by `density`. This produces the following.

```
  $names
  [1] "x"  "y"  "bw"  "n"  "call"  "data.name"  "has.na"
  $class
  [1] "density"
```

The attribute `x` is the vector of `n` points at which the density was estimated, to give the values `y`. You can retrieve these values with `mydense$x`. Note that density estimation does not produce an analytic function, but rather evaluations of the estimator at a pre-defined set of points. The function `lines` in the above code just connects the `x` and `y` values in order to plot an apparently continuous function.

The idea of kernel density estimation extends to higher dimensions. The following code is an example in two dimensions applied to the geyser data. It produces figure 7.4.

R file: `kde2d.R`

```
##### Investigate kernel density estimation in 2D
##### Modified from the example given by Venables & Ripley (2002)
```

```
library(fields) # for image.plot
library(MASS)
library(RColorBrewer)
mypalette <- colorRampPalette(brewer.pal(9, "Greys"), space="rgb",
                              interpolate="linear", bias=2)
mycols <- mypalette(64)
geyser2 <- data.frame(as.data.frame(geyser)[-1, ],
                      pduration=geyser$duration[-299])
attach(geyser2)
pdf("kde2d.pdf", 12, 4)
par(mfrow=c(1,3), mar=c(3.5,3.5,0.5,1), oma=0.5*c(1,1,1,1),
    mgp=c(2.2,0.8,0), cex=1.0)
plot(pduration, waiting, xlim=c(0.5, 6), ylim=c(35, 115), xaxs="i",
     yaxs="i", xlab="previous duration", ylab="waiting")
f1 <- kde2d(pduration, waiting, n=50, h=c(0.75, 10),
            lims=c(0.5, 6, 35, 115))
image.plot(f1, zlim=c(0, 0.075), xlim=c(0.5, 6), ylim=c(35, 115), xaxs="i",
           yaxs="i", xlab="previous duration", ylab="waiting", col=mycols)
persp(f1, phi=30, theta=20, d=5, xlim=c(0.5, 6), ylim=c(35, 115), xaxs="i",
      yaxs="i", xlab="previous duration", ylab="waiting", zlab="density")
dev.off()
detach(geyser2)
```

Good kernel density estimation requires that we adopt a suitable bandwidth. There are various ways to achieve this. Experimenting with various values and inspecting the results can be quite effective. Judgement will anyway be required, no matter what theoretical scheme is used. A widely used scheme is to compute the *mean integrated squared error* (also called the L^2 *risk function*), defined as

$$\text{MISE} = E\left[\int (\hat{f}(x;\lambda) - f(x))^2\,dx\right] \tag{7.21}$$

where $\hat{f}(x;\lambda)$ is our kernel density estimate of the true, unknown function $f(x)$. The expectation value in the above equation is taken with respect to the sample of data, i.e. it represents what we would get if we had infinite data. Minimizing the MISE with respect to λ gives us a good choice of bandwidth. The problem is that the expression involves the very function we do not know, $f(x)$. But we can work around this by first expanding the square to write it as

$$\text{MISE} = E\left[\int \hat{f}^2(x;\lambda)\,dx\right] - 2E\left[\int \hat{f}(x;\lambda)f(x)\,dx\right] + E\left[\int f^2(x)\,dx\right]. \tag{7.22}$$

The first term we can estimate directly from the data and our kernel density estimator. The last term is independent of λ, so can be ignored. The middle term may be estimated by *leave-one-out cross-validation*: given a set of N data points, we select one point x_i, then use all the other points to estimate the density (with our estimator) at that one point. Label this estimated density as $\hat{f}_{-i}(x_i;\lambda)$. Essentially we are using all the other data points to approximate $f(x)$ for the sake of computing $\hat{f}_{-i}(x_i,\lambda)$. We then repeat this for all N points and average. To find the optimal bandwidth we therefore minimize the following metric with respect to λ

$$\int \hat{f}^2(x;\lambda)\,dx - \frac{2}{N}\sum_{i=1}^{N} \hat{f}_{-i}(x_i;\lambda). \tag{7.23}$$

This is sometimes called the *estimated risk*.

The R function `density` implements various ways of estimating the bandwidth of a kernel from the data. Details on these can be obtained with `help("bw.nrd")`. An example is the function `bw.nrd`. It can be used to find the bandwidth by typing `bw.nrd(geyser$duration)` which gives the value 0.389. We can instead do this directly within the call to `density` by setting `bw="nrd"`.

We will use kernels for density estimation in chapter 9, and we will encounter kernels again in section 12.4 for doing local regression.