Density Estimation

Su2016 - Dr. Junvie Pailden

August 1, 2016

Contents

Univariate Density Estimation

Density estimation is a collection of methods for constructing an estimate of a probability density, as a function of an observed sample of data.

We will cover the following nonparametric methods of density estimation:

- histograms
- frequency polygons
- average shifted histogram
- kernel density estimators

Histograms

In elementary data analysis projects we are faced with tricky questions such as

- how to determine the best number of bins
- how to determine the boundaries and width of class intervals
- how to handle unequal class interval widths

These decisions are made automatically in many software packages but some produce undesirable results.

With R software, the user has control over several options in constructing histograms.

Issues with Histograms

The histogram is a piecewise constant approximation of the density function.

Because data is contaminated by noise, the estimator that presents to much detail is not necessary better.

The choice of bin width for a histogram is a choice of smoothing parameter.

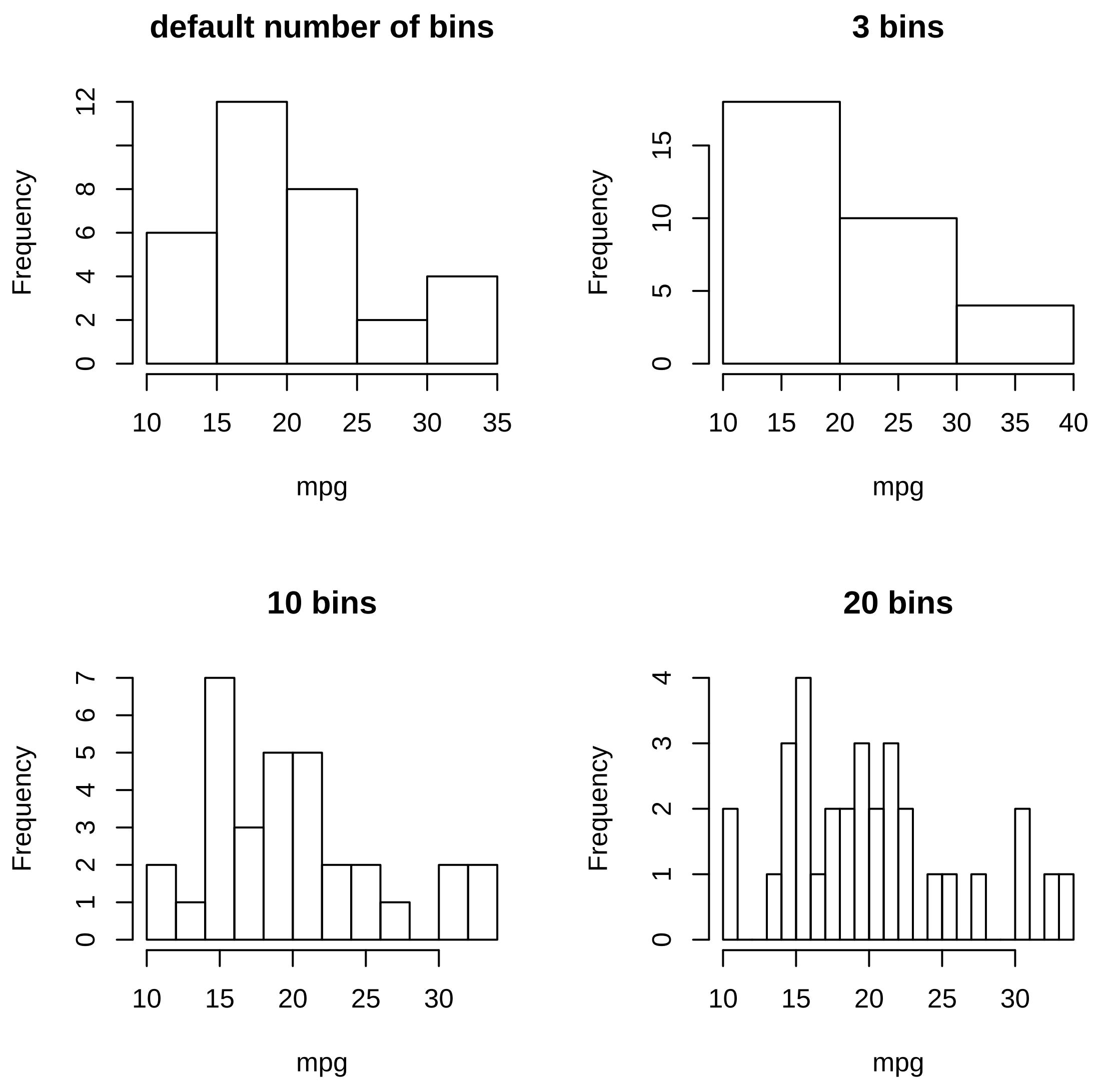
A narrow bin width may undersmooth the data, presenting too much detail, while wider bin width may oversmooth the data, obscuring important featers.

Several rules are commonly applied that suggest an optimal important features.

```r
mpg <-  mtcars$mpg
length(mpg)
```

```
## [1] 32
```

```r
par(mfrow = c(2,2))
hist(mpg, main = "default number of bins")
hist(mpg, breaks=3, main = "3 bins")
hist(mpg, breaks=10, main = "10 bins")
hist(mpg, breaks=20, main = "20 bins")
```



```r
par(mfcol = c(1,1))
```

Suppose that a random $X_1, \ldots, X_n$ is observed. Given class intervals of equal width h, the histogram density estimate based on a sample size n is

$$\hat{f}(x) = \frac{v_k}{nh}, \quad t_k \leq x < t_{k+1},$$

where v_k is the number of sample points in the class interval $[t_k, t_{k+1})$.

If the bin width $h = 1$, then the density estiamte $\hat{f}(x)$ the relative frequency of class containing the point x.

Sturges' Rule

The "Sturges' Rule" is default rule (in `{hist} in R) in determining the width of class intervals.`

`Sturges rule is based on the implicit assumption that the sampled population is normally distributed.`

`We choose a family of discrete distributions that converge in distribution to normal as the number of classes (and sample size` n `) tend to infinity.`

`The binomial distribution is the most obvious candidate in this case.  Consider sample sizes` $n = 2^k, k = 1, 2, \ldots.$ `For` k `large (`n `large),`

$$Binomial(k, 1/2) \rightarrow Normal\left(\mu = \frac{n}{2}, \sigma^2 = \frac{n}{4}\right)$$

`Here` $k = \log_2 n$ `and we have` $k+1$ `bins with expected class frequency`

$$\binom{\log_2 n}{j}, \quad j = 0, 1, \ldots, k.$$

`The width of class intervals is given by`

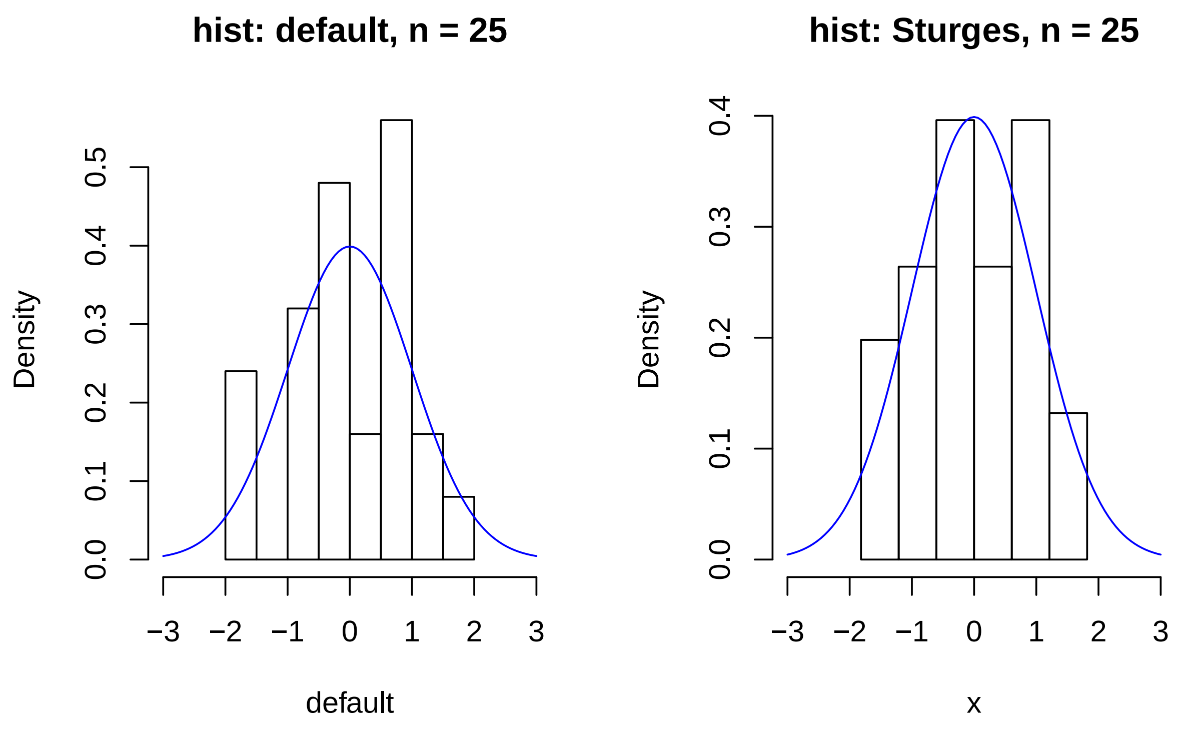
$$h = \frac{R}{1 + \log_2 n}$$

`where` R `is the sample range.`

> `The Sturges' rule is designed for data sampled from symmetric, unimodal populations, but is not a good choice for skewed distributiosn or distributions with more than one mode.  For large samples, Sturges' rule tends to oversmooth.`

```r
set.seed(12345)
n <- 25
x <- rnorm(n)
# default in R
par(mfcol = c(1, 2))
h.default <- hist(x, prob = T, xlab = "default",
    main = "hist: default, n = 25",xlim = c(-3,3))
curve(dnorm(x), add = TRUE, col = "blue")

# calc breaks according to Sturges' Rule
nclass <- ceiling(1 + log2(n))
cwidth <- diff(range(x) / nclass)
```

```r
breaks <- min(x) + cwidth * 0:nclass

h.sturges <- hist(x, breaks = breaks, prob = T,
    main = "hist: Sturges, n = 25",xlim = c(-3,3))
curve(dnorm(x), add = TRUE, col = "blue")
```



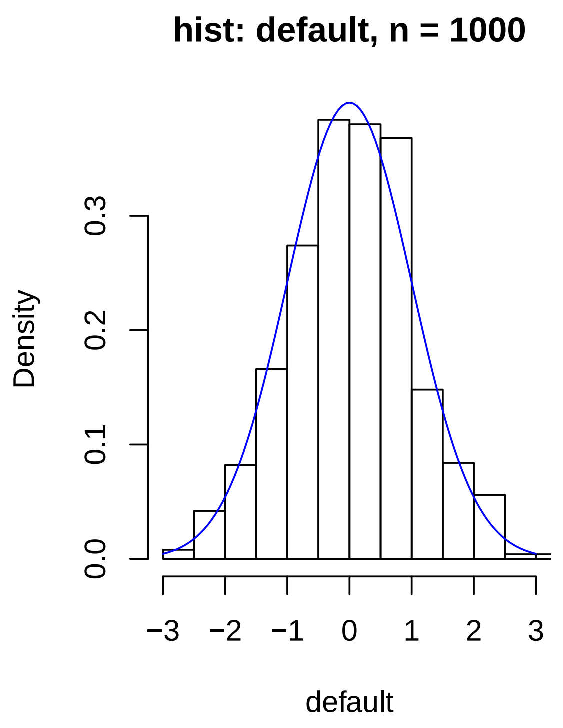
```r
# default endpoints, counts, density
list(breaks = h.default$breaks,
     counts = h.default$counts,
         density = h.default$density)
```

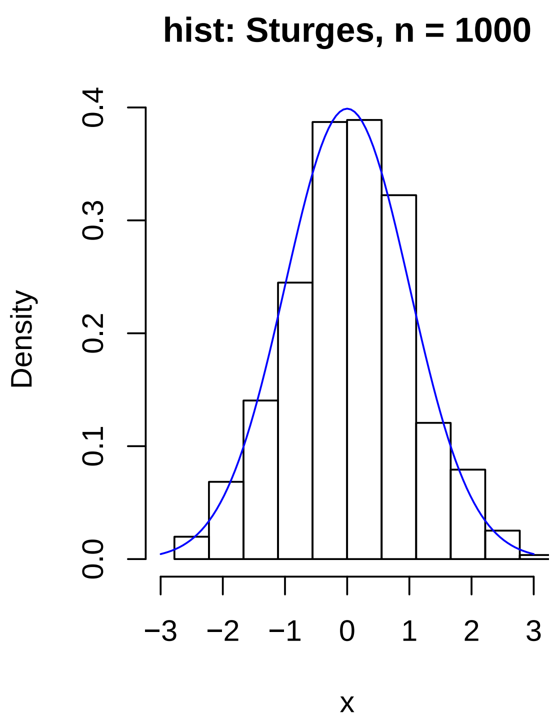
```
## $breaks
## [1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0
##
## $counts
## [1] 3 0 4 6 2 7 2 1
##
## $density
## [1] 0.24 0.00 0.32 0.48 0.16 0.56 0.16 0.08
```

```r
# sturges' endpoints, counts, density
list(breaks = round(h.sturges$breaks,1),
         counts = h.sturges$counts,
         density = round(h.sturges$density,2))
```

```
## $breaks
## [1] -1.8 -1.2 -0.6  0.0  0.6  1.2  1.8
##
## $counts
## [1] 3 4 6 4 6 2
##
## $density
## [1] 0.20 0.26 0.40 0.26 0.40 0.13
```

```
# large sample size
set.seed(12345)
n <- 1000
x <- rnorm(n)
# default in R

h.default <- hist(x, prob = T, xlab = "default",
    main = "hist: default, n = 1000",xlim = c(-3,3))
curve(dnorm(x), add = TRUE, col = "blue")

# calc breaks according to Sturges' Rule
nclass <- ceiling(1 + log2(n))
cwidth <- diff(range(x) / nclass)
breaks <- min(x) + cwidth * 0:nclass

h.sturges <- hist(x, breaks = breaks, prob = T,
    main = "hist: Sturges, n = 1000",xlim = c(-3,3))
curve(dnorm(x), add = TRUE, col = "blue")
```





Scott's Normal Reference Rule

The Mean Square Error (MSE) is a criterion often used for comparing smoothing parameters. The MSE of a density estimator $\hat{f}(x)$ at x is

$$MSE(\hat{f}(x)) = E(\hat{f}(x) - f(x))^2 = Var(\hat{f}(x)) + bias^2(\hat{f}(x)).$$

The MSE, however, measures pointwise error (at a particular value of x).

A global criterion is the integrated square error (ISE), which is the L_2 norm

$$ISE(\hat{f}(x)) = \int (\hat{f}(x) - f(x))^2 dx.$$

It is simple to consider the statistic, mean integrated square error (MISE),

$$\begin{aligned} MISE = E[ISE] &= E\left[\int (\hat{f}(x) - f(x))^2 dx\right] \\ &= \int E[(\hat{f}(x) - f(x))^2] dx \quad \text{(by Fubini's Theorem)} \\ &= \int MSE(\hat{f}(x)) := IMSE \end{aligned}$$

Under some regularity conditions on f, Scott (1979) shows that

$$MISE = \frac{1}{nh} + \frac{h^2}{12}\int f'(x)^2 dx + O\left(\frac{1}{n} + h^3\right)$$

where f' is the first derivative of the true density f and the terms inside $O(\cdot)$ is bounded as $n \to \infty$.

The optimal bin width under Scott's rule is

$$h_n^* = \left(\frac{6n}{\int f'(x)^2 dx}\right)^{1/3},$$

with asymptotic MISE

$$AMISE^* = \left(\frac{9}{16}\int f'(x) dx\right)^{1/3} n^{-2/3}.$$

For normal distributions with variance σ^2, $h_n^* = 2(3^{1/3})\pi^{1/6}\sigma n^{-1/3}$. SInce σ is unknown , we subsitute σ by the sample standard deviation $\hat{\sigma}$.

Thus, the Scott's Normal Reference Rule is

$$\hat{h} = 3.49\hat{\sigma}n^{-1/3}.$$

Consider a density histogram estimate of the data on the eruptions of the Old Faithful Geyser and use Scott's Normal Reference Rule to estimate the bin width.

```r
library(MASS)
#for geyser and truehist
waiting <- geyser$waiting
summary(waiting)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   43.00   59.00   76.00   72.31   83.00  108.00
```

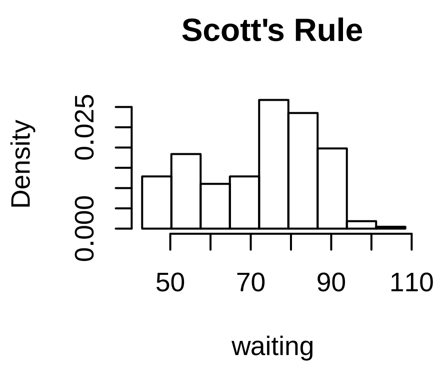
```r
n <- length(waiting)
# rounding the constant in Scott's rule
# and using sample standard deviation to estimate sigma
h <- 3.5 * sd(waiting) * n^(-1/3)


# number of classes is determined by the range and h
m <- min(waiting)
M <- max(waiting)
nclass <- ceiling((M - m) / h)
data.frame(sample_size = n, width = h, classes = nclass)
```

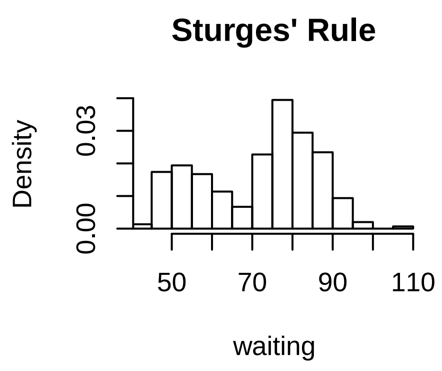
```
##   sample_size    width classes
## 1         299 7.270373       9
```

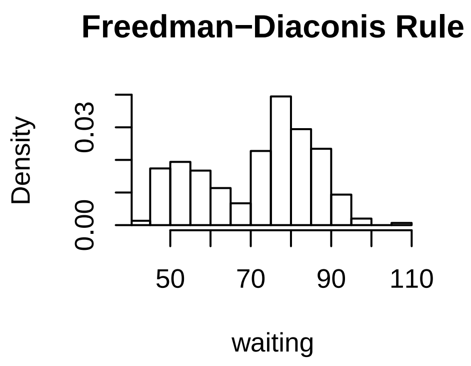
```r
breaks <- m + h * 0:nclass
round(breaks,3)
```

```
##  [1]  43.000  50.270  57.541  64.811  72.081  79.352  86.622  93.893
##  [9] 101.163 108.433
```

```r
par(mfrow = c(2,2))
hist(waiting, breaks = breaks,
        xlim = c(43, 108), prob = TRUE,
        main = "Scott's Rule")
hist(waiting, breaks = "Sturges", prob=TRUE,
        xlim = c(43, 108),
        main = "Sturges' Rule")
hist(waiting, breaks = "FD", prob=TRUE,
        xlim = c(43, 108),
        main = "Freedman-Diaconis Rule")
```







Freedman-Diaconis Rule

Scott's normal reference rule is a member of a class of rules that select the optimal bin width according to a formula $\hat{h} = Tn^{-1/3}$, where T is a statistic.

The Freedman-Diaconis (FD) rule is another member of this class. For the FD rule, the statistic T is twice the sample interquartile range.

$$\hat{h} = 2(IQR)n^{-1/3},$$

where IQR denotes the sample interquartile range.

The IQR is less sensitive than the sample standard deviation to outliers in the data.

The number of classes is the sample range divided by the bin width.

```r
# number of classes is determined by the range and h
m <- min(waiting)
h <- 2*IQR(waiting)*n^(-1/3)
nclass <- ceiling(diff(range(waiting) / h))
data.frame(sample_size = n, width = h, classes = nclass)
```

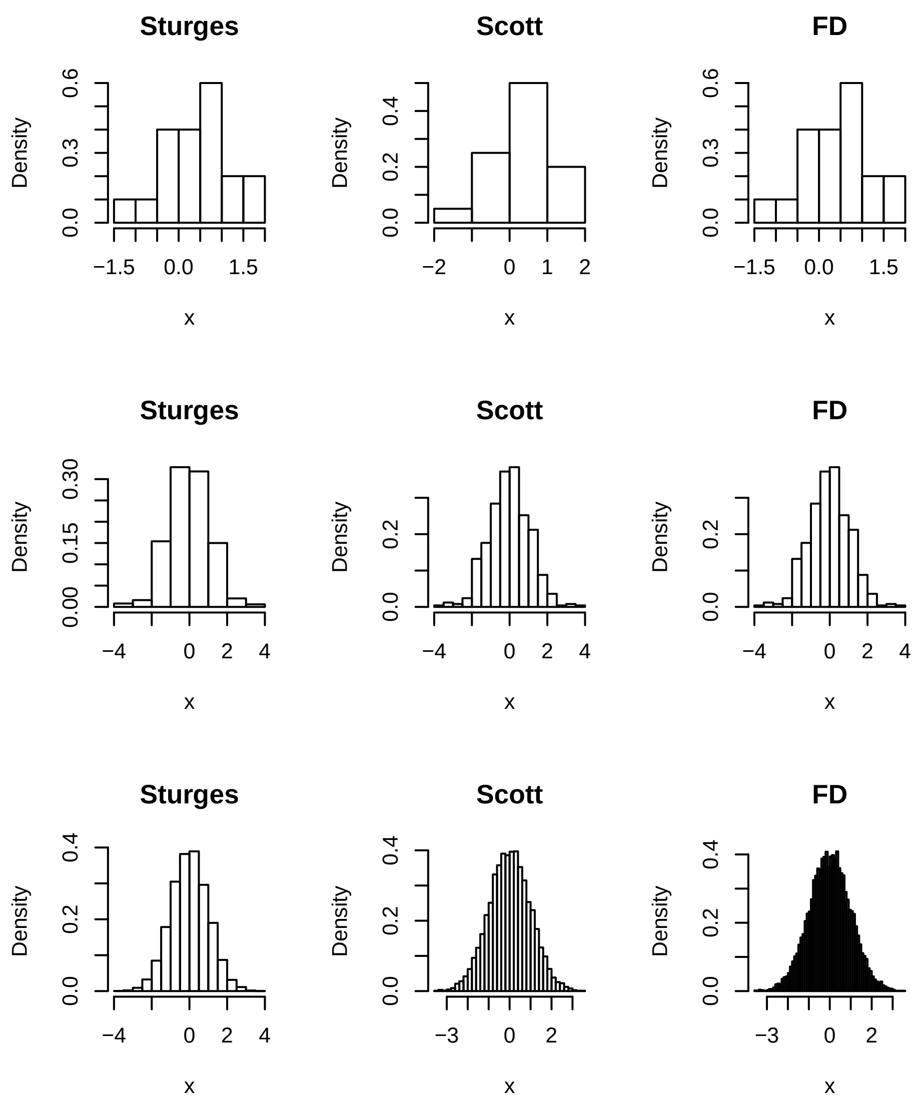
```
##   sample_size    width classes
## 1         299 7.178232      10
```

```
breaks <- m + h * 0:nclass
round(breaks,3)
```

```
## [1]  43.000  50.178  57.356  64.535  71.713  78.891  86.069  93.248
## [9] 100.426 107.604 114.782
```

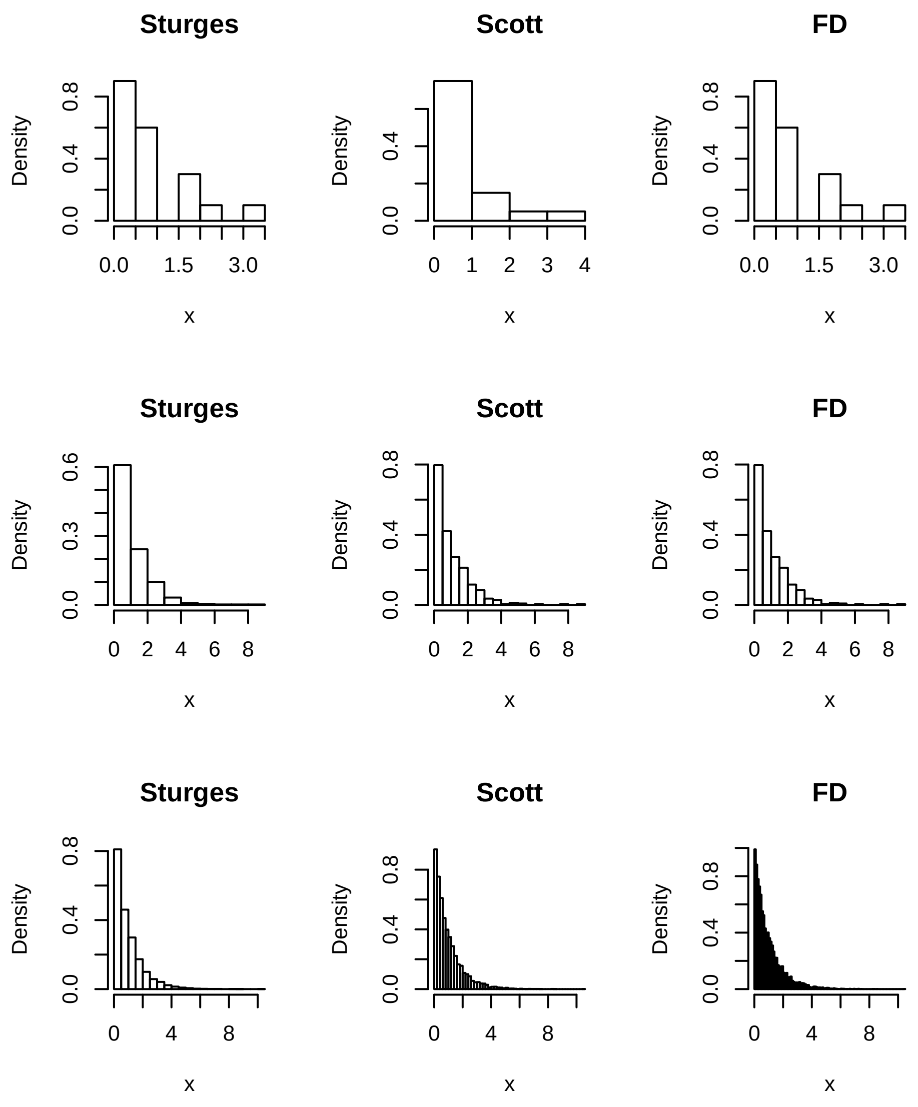
Comparison of Different Rules, Normal Data

```
set.seed(17)
par(mfrow = c(3,3))
n <- c(20,500,10000)
for( i in 1:3){
x <- rnorm(n[i])
hist(x, breaks = "Sturges", prob = TRUE, main = "Sturges")
hist(x, breaks = "Scott", prob = TRUE, main = "Scott")
hist(x, breaks = "FD", prob = TRUE, main = "FD")
}
```



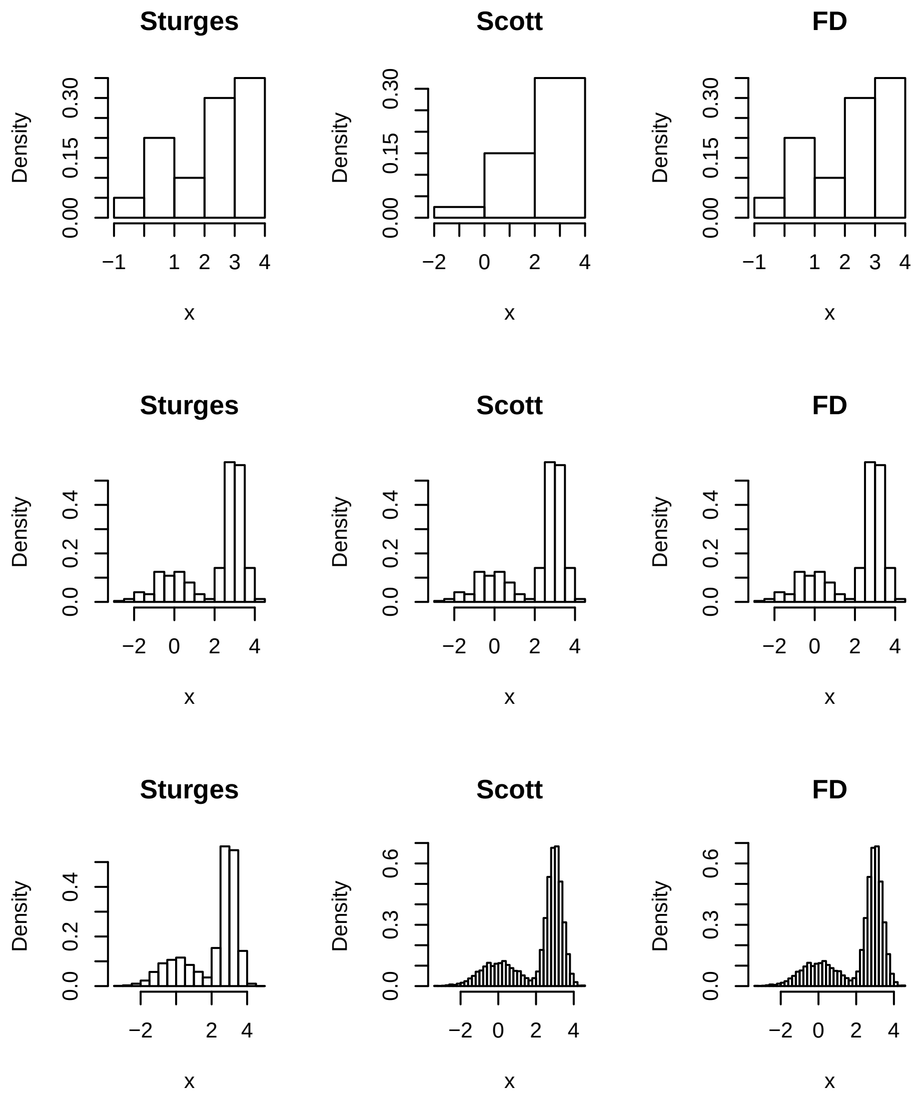
Comparison of Different Rules, Exponential Data

```r
set.seed(17)
par(mfrow = c(3,3))
n <- c(20,500,10000)
for( i in 1:3){
x <- rexp(n[i])
hist(x, breaks = "Sturges", prob = TRUE, main = "Sturges")
hist(x, breaks = "Scott", prob = TRUE, main = "Scott")
hist(x, breaks = "FD", prob = TRUE, main = "FD")
}
```



Comparison of Different Rules, Normal Mixture Data

```r
set.seed(17)
par(mfrow = c(3,3))
n <- c(20,500,10000)
for( i in 1:3){
x <- ifelse(runif(n[i])<0.3, rnorm(n[i],0,1),
        rnorm(n[i],3,0.4))
hist(x, breaks = "Sturges", prob = TRUE, main = "Sturges")
hist(x, breaks = "Scott", prob = TRUE, main = "Scott")
hist(x, breaks = "FD", prob = TRUE, main = "FD")
}
```



Kernel Density Estimation

Kernel density estimation generalizes the idea of a histogram density estimate.

If a histogram with bin h is constructed from a sample $X_1, \dots, X_n$, then a density estimate for a point x within the range of the data is

$$\hat{f}(x) = \frac{1}{2hn} \times k,$$

where k is the number of sample points in the interval $(x-h, x+h)$.

This estimator can be written

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

where $w(t) = \frac{1}{2}I(|t| < 1)$ is a weight function.

The density estimator $\hat{f}(x)$ with $w(t) = \frac{1}{2}I(|t| < 1)$ is called the *naive density estimator*.

This weight function has the property that $\int_{-1}^{1} w(t)dt = 1$, and $w(t) \geq 0$, so $w(t)$ is a probability density supported on the interval $[-1, 1]$.

Kernel density estimation replaces the weight function $w(t)$ in the naive estimator with a function $K(\cdot)$ called a kernel function, such that

$$\int_{-\infty}^{\infty} K(t)dt = 1.$$

In probability density estimation, $K(\cdot)$ is usually a symmetric probability density function.

Rectangular Kernel Weight Function

The weight function $w(t) = \frac{1}{2}I(|t| < 1)$ is called the rectangular kernel.

The rectangular kernel is a symmetric probability density centered at the origin, and

$$\frac{1}{nh} w\left(\frac{x - X_i}{h}\right),$$

corresponds to a rectangle of area $1/n$ centered at X_i.

The density estimate is the sum of rectangles located within h units from x.

Symmetric Kernel Density Estimators

Suppose that $K(\cdot)$ a symmetric probability density centered at the origin, and define

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

Then $\hat{f}$ is a probability density function.

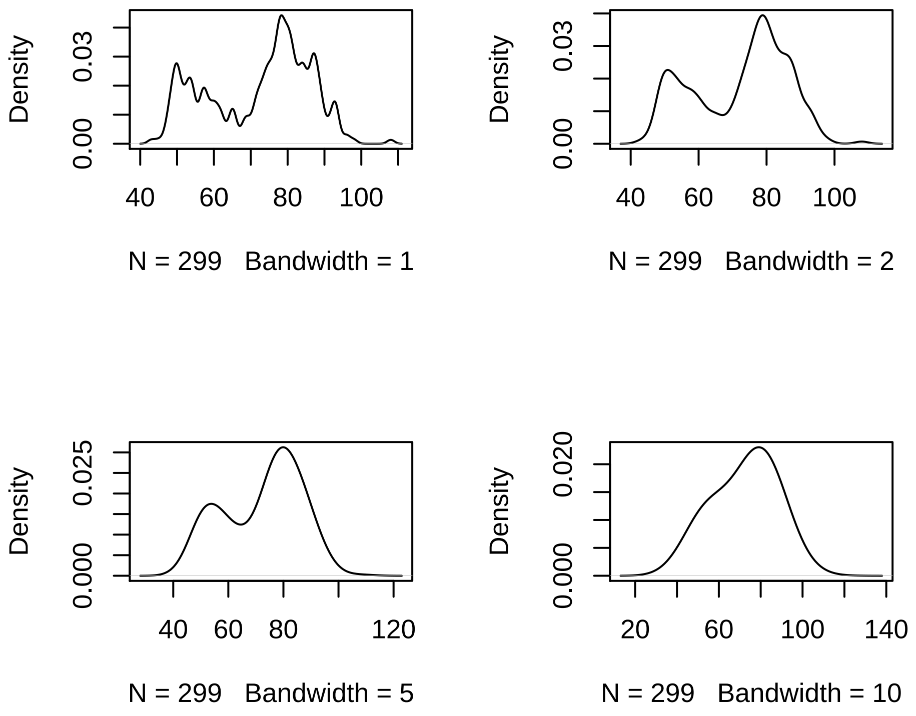
Note that certain continuity and differentiability properties of $K(\cdot)$ also hold for $\hat{f}_K(x)$.

If $K(x)$ is a probability density, then $\hat{f}_K(x)$ is continuous at x if $K(x)$ is continuous at x, and $\hat{f}_K(x)$ has an r^{th} order derivative at x if $K^{(r)}(x)$ exists.

The **bin width (bandwith or window width)** h is a smoothing parameter; small values of h reveal local features of the density while large values of h produce a smoother density estimate.

Old Faithful Data, Gausian Kernel Density Estimate

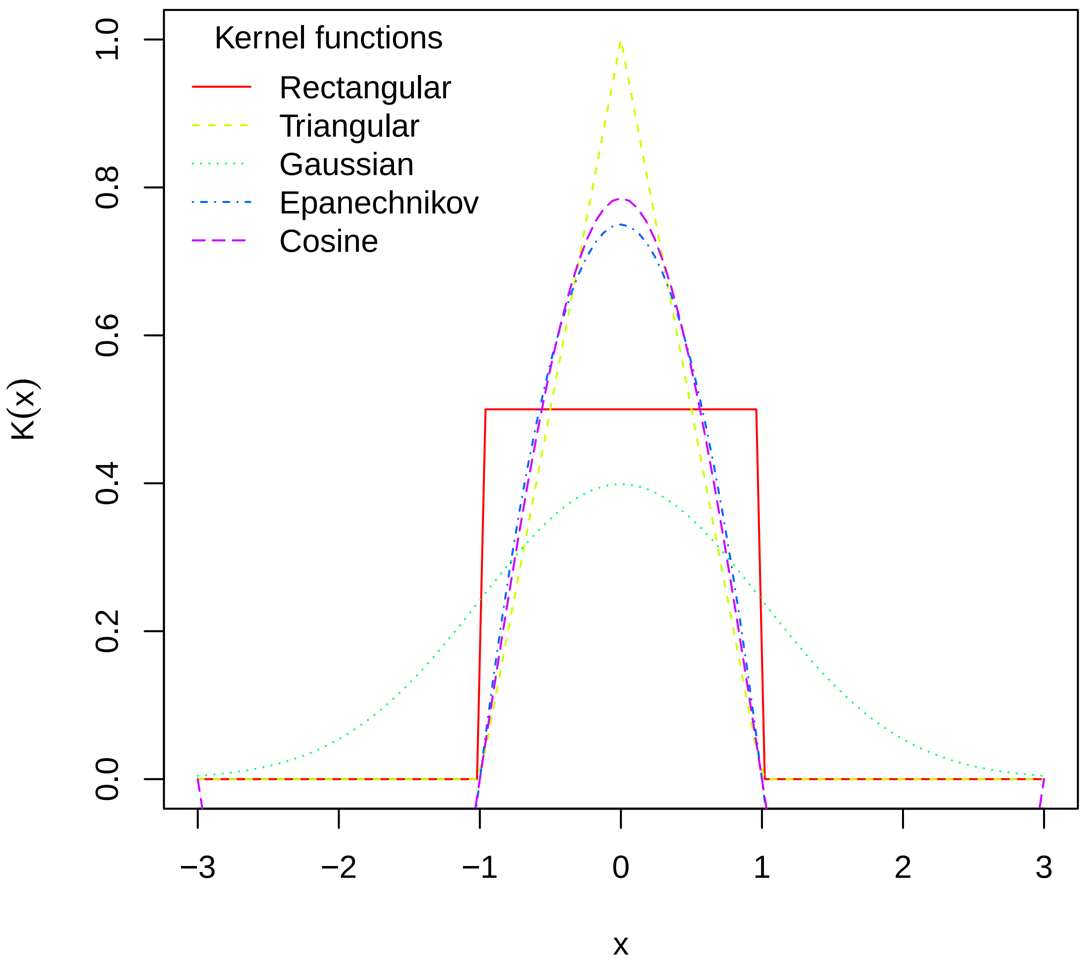
```r
par(mfrow = c(2, 2))
waiting <- geyser$waiting
plot(density(waiting, bw = 1), main = "")
plot(density(waiting, bw = 2), main = "")
plot(density(waiting, bw = 5), main = "")
plot(density(waiting, bw = 10), main = "")
```



Choices of Kernel Functions

Kernel	$K(t)$	Support	σ_K^2	Efficiency
Gaussian	$\frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}t^2)$	R	1	1.0513
Epanechnikov	$\frac{3}{4}(1-t^2)$	$abs(t) < 1$	1/5	1
Rectangular	$\frac{1}{2}$	$abs(t) < 1$	1/3	1.0758
Triangular	$1 - abs(t)$	$abs(t) < 1$	1/6	1.0143
Biweight	$\frac{15}{16}(1-t^2)^2$	$abs(t) < 1$	1/7	1.0061
Cosine	$\frac{\pi}{4}\cos(\frac{\pi}{2}t)$	R	$1 - 8/\pi^2$	1.0005

```r
# kernel functions
rec <- function(x) (abs(x) < 1) * 0.5
tri <- function(x) (abs(x) < 1) * (1 - abs(x))
gauss <- function(x) 1/sqrt(2*pi) * exp(-(x^2)/2)
epanech <- function(x) (3/4)*(1-x^2)
cosine <- function(x) (pi/4)*cos(x*pi/2)

# colors
cc <- rainbow(5)

# plot
curve(rec(x), xlim = c(-3,3), ylim = c(0,1),
  lty = 1, col = cc[1],
    ylab = expression(K(x)))
curve(tri(x), add = TRUE, lty = 2, col = cc[2])
curve(gauss(x), add = TRUE, lty = 3, col = cc[3])
curve(epanech(x), add = TRUE, lty = 4, col = cc[4])
curve(cosine(x), add = TRUE, lty = 5, col = cc[5])
legend("topleft", legend = c("Rectangular", "Triangular",
    "Gaussian", "Epanechnikov", "Cosine"), lty = 1:5,
    title = "Kernel functions", col = cc,
    bty = "n")
```



Gaussian Kernel Function

For a Gaussian Kernel, the bandwidth h (when the true distribution is normal) that optimizes the IMSE is

$$h = (4/3)^{1/5} \sigma n^{-1/5} = 1.06 \sigma n^{-1/5}.$$

If the true density is not unimodal, the Gaussian Kernel will tend to oversmooth. Alternatively, one can use a more robust estimate of dispersion and use

$$h = 0.9 \min(S, IQR/1.34) n^{-1/5}.$$

Gaussian Kernel Function Illustration

Consider the data set $x = 0, 1, 1.1, 1.5, 1.9, 2.8, 2.9, 3.5$. Let us construct a density estimator based on the Gaussian Kernel function on x.

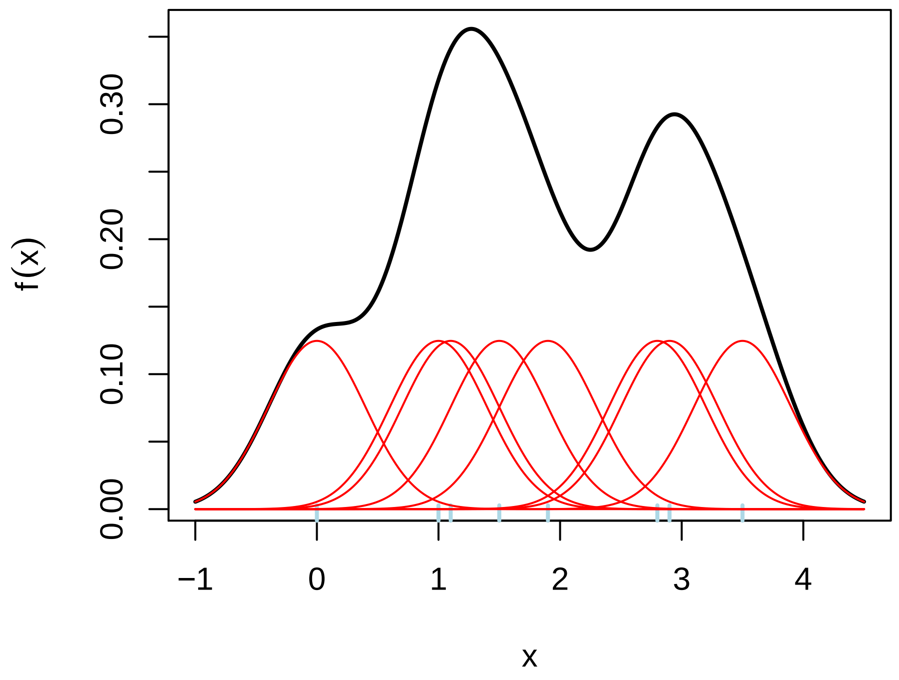
```
x <- c(0, 1, 1.1, 1.5, 1.9, 2.8, 2.9, 3.5)
n <- length(x)
xgrid <- seq(from = min(x) - 1, to = max(x) + 1, by = 0.01)
# bandwidth
h <- 0.4
gauss <- function(x) 1/sqrt(2*pi) * exp(-(x^2)/2)
# kernels on the values of xgrid
bumps <- sapply(x, function(a) gauss((xgrid - a)/h)/(n * h))
dim(bumps)
```

```
## [1] 551   8
```

```
# xgrid  and its kernel values
cbind(xgrid[201:204], round(bumps[201:204,],3))
```

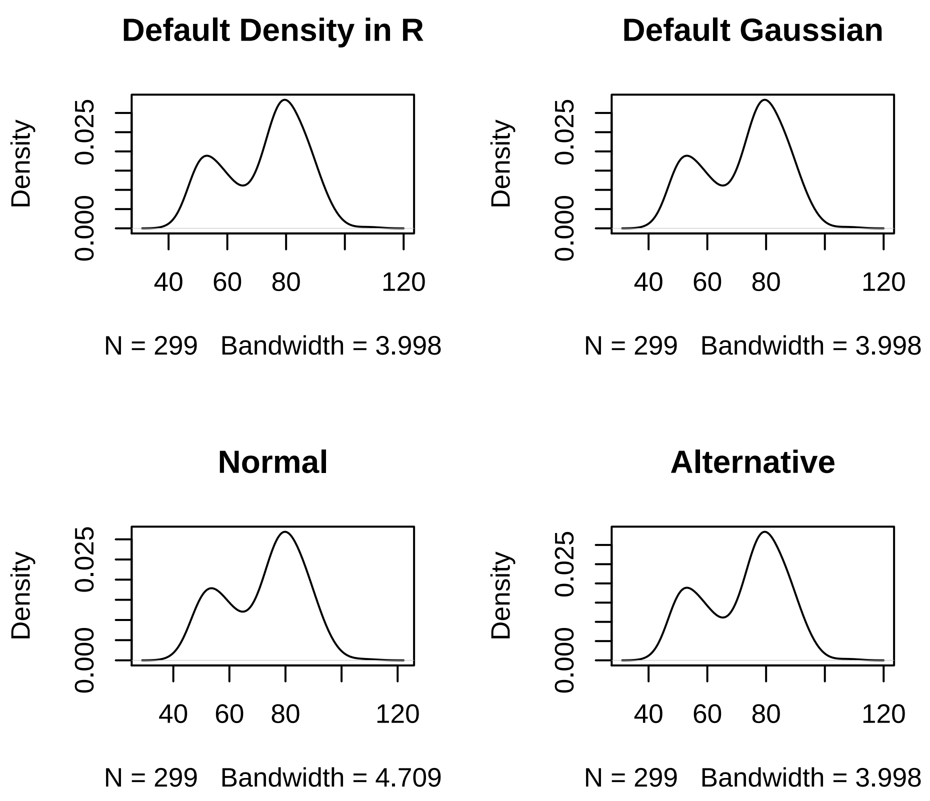
```
##      [,1]  [,2]  [,3]  [,4]  [,5]  [,6] [,7] [,8] [,9]
## [1,] 1.00 0.005 0.125 0.121 0.057 0.010    0    0    0
## [2,] 1.01 0.005 0.125 0.122 0.059 0.010    0    0    0
## [3,] 1.02 0.005 0.125 0.122 0.061 0.011    0    0    0
## [4,] 1.03 0.005 0.124 0.123 0.063 0.012    0    0    0
```

```
plot(xgrid, rowSums(bumps), ylab = expression(hat(f)(x)),
    type = "l", xlab = "x", lwd = 2)
rug(x, lwd = 2,col = "light blue")
out <- apply(bumps, 2, function(b)
lines(xgrid, b, col="red"))
```



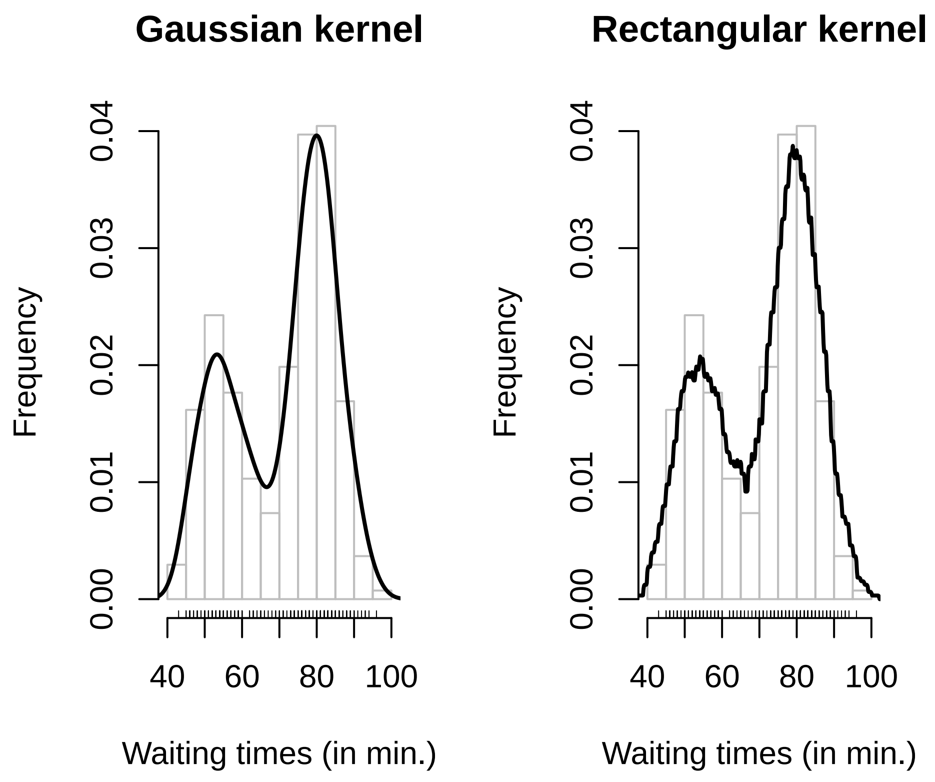
Old Faithful Data, Gausian Kernel Density Estimate

```
par(mfrow = c(2, 2))
library(MASS)
waiting <- geyser$waiting
n <- length(waiting)
h1 <- 1.06 * sd(waiting) * n^(-1/5)
h2 <- .9 * min(c(IQR(waiting)/1.34,
    sd(waiting))) * n^(-1/5)
plot(density(waiting), main = "Default Density in R")
plot(density(waiting, kernel = "gaussian"),
        main = "Default Gaussian")
plot(density(waiting, bw = h1), main = "Normal")
plot(density(waiting, bw = h2), main = "Alternative")
```



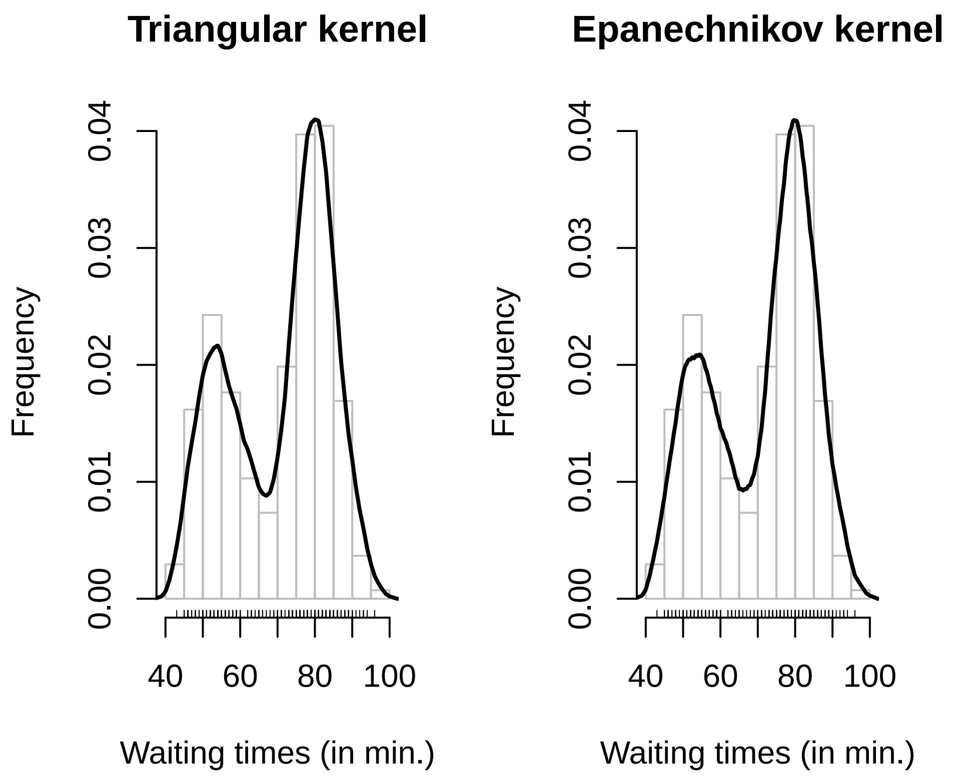
Old Faithful Data, Gaussian and Rectangular Kernel Functions

```r
x <- faithful$waiting
layout(matrix(1:2, ncol = 2))
hist(x, xlab = "Waiting times (in min.)",
    ylab = "Frequency", probability = TRUE,
    main = "Gaussian kernel", border = "gray")
y1 <- density(x, width = 12)
lines(y1, lwd = 2)
rug(x)
hist(x, xlab = "Waiting times (in min.)",
    ylab = "Frequency", probability = TRUE,
    main = "Rectangular kernel", border = "gray")
y2 <- density(x, width = 12, kernel = "rectangular")
lines(y2, lwd = 2)
rug(x)
```



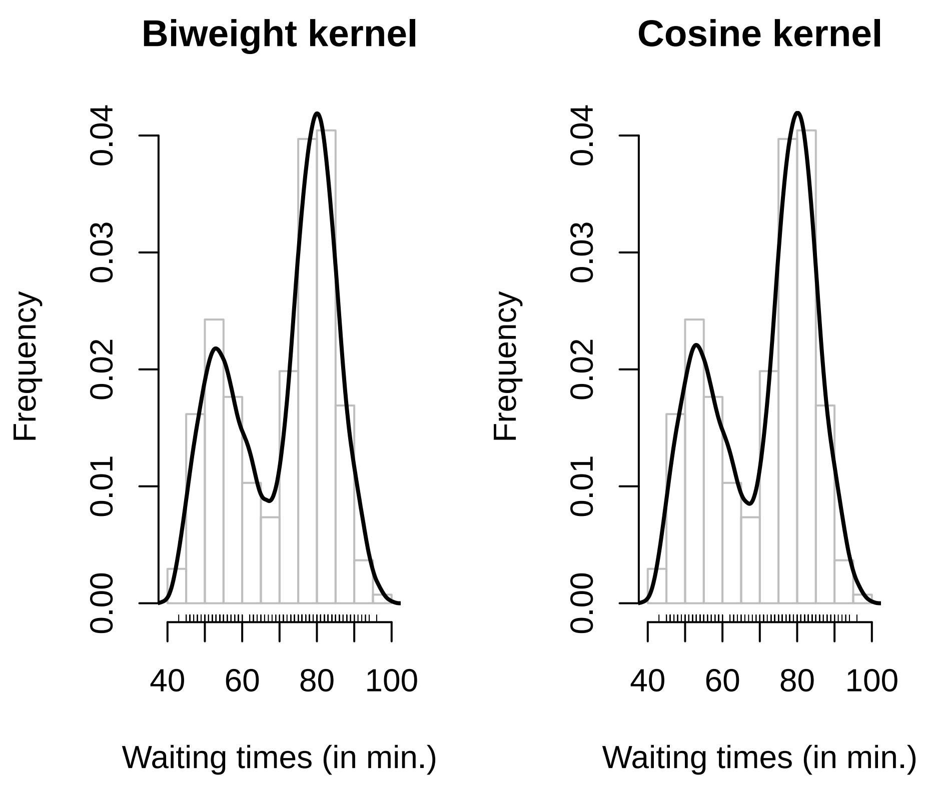
Old Faithful Data, Triangular and Epanechnikov Kernel Functions

```r
x <- faithful$waiting
layout(matrix(1:2, ncol = 2))
hist(x, xlab = "Waiting times (in min.)",
    ylab = "Frequency", probability = TRUE,
    main = "Triangular kernel", border = "gray")
y3 <- density(x, width = 12, kernel = "triangular")
lines(y3, lwd = 2)
rug(x)
hist(x, xlab = "Waiting times (in min.)",
    ylab = "Frequency", probability = TRUE,
    main = "Epanechnikov kernel", border = "gray")
y4 <- density(x, width = 12, kernel = "epanechnikov")
lines(y4, lwd = 2)
rug(x)
```



Old Faithful Data, Biweight and Cosine Kernel Functions

```
x <- faithful$waiting
layout(matrix(1:2, ncol = 2))
hist(x, xlab = "Waiting times (in min.)",
    ylab = "Frequency", probability = TRUE,
    main = "Biweight kernel", border = "gray")
y5 <- density(x, width = 12, kernel = "biweight")
lines(y5, lwd = 2)
rug(x)
hist(x, xlab = "Waiting times (in min.)",
    ylab = "Frequency", probability = TRUE,
    main = "Cosine kernel", border = "gray")
y6 <- density(x, width = 12, kernel = "cosine")
lines(y6, lwd = 2)
rug(x)
```



Kernel Density Estimation using ggplot2 package

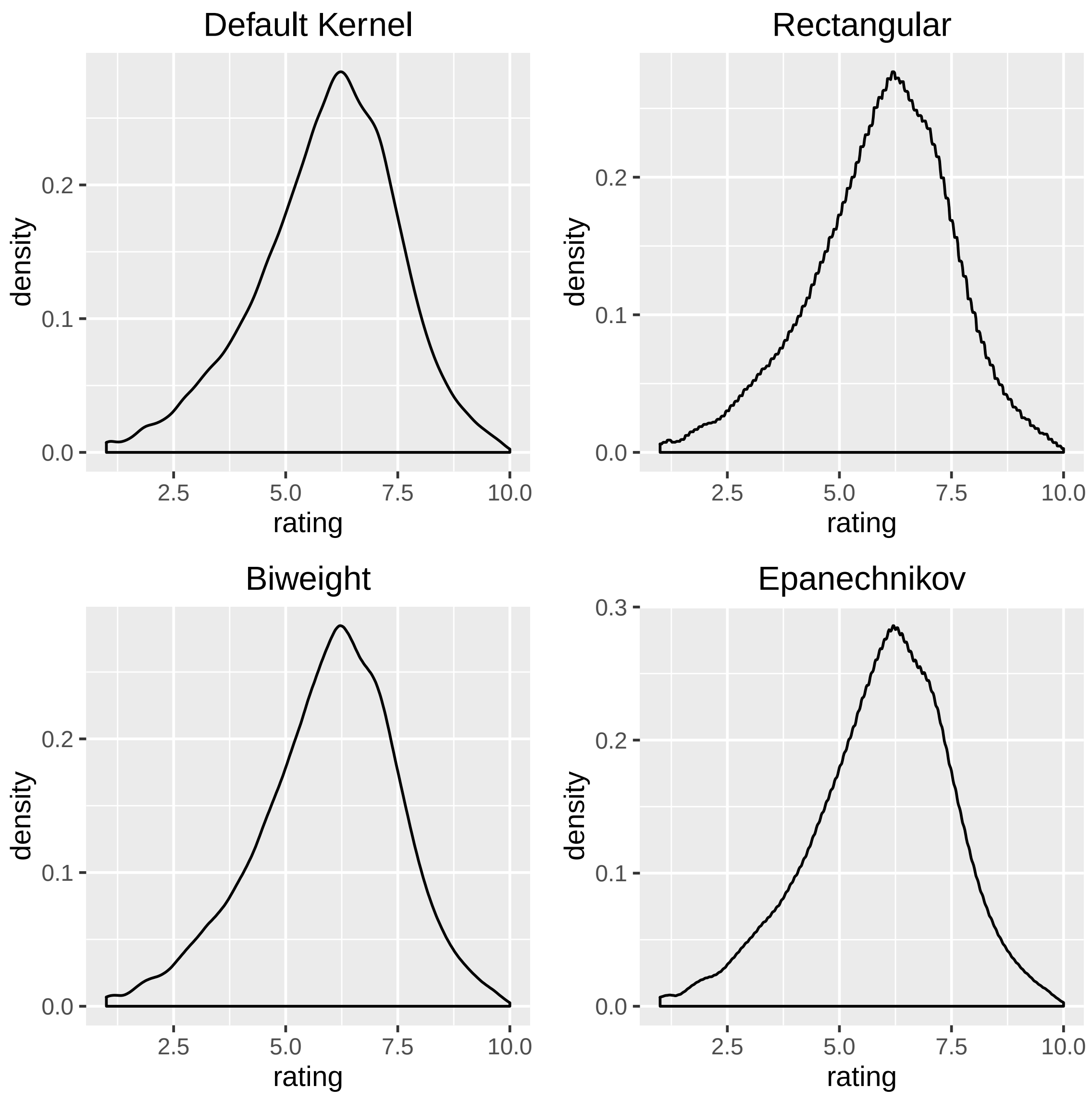
For this example we will be using the movies data (with 28819 rows and 24 variables) in ggplot2. This data contains movie information and user ratings from IMDB.com.

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following object is masked _by_ '.GlobalEnv':
##
##     mpg
```

```
library(gridExtra)
library(ggplot2movies)
m <- ggplot(movies, aes(x = rating))
g1 <- m + geom_density() + ggtitle("Default Kernel")
g2 <- m + geom_density(kernel = "rectangular") +
    ggtitle("Rectangular")
g3 <- m + geom_density(kernel = "biweight") +
    ggtitle("Biweight")
```

```r
g4 <- m + geom_density(kernel = "epanechnikov") +
    ggtitle("Epanechnikov")
grid.arrange(g1, g2, g3, g4, nrow = 2)
```



Multivariate Density Estimation

Suppose $\mathbf{X} = (X_1, \ldots, X_d)$ is a random vector in $\mathbb{R}^d$, and $K(\mathbf{X}) : \mathbb{R}^d \rightarrow \mathbb{R}$ is a kernel function, such that $K(X)$ is a density function on $\mathbb{R}^d$.

Let the $n \times d$ matrix (x_{ij}) be an observed sample from the distribution of X .

The smoothing parameter is a d-dimensional vector $h = (h_1, \ldots, h_d)$.

The product multivariate kernel density of $f(\mathbf{X})$ is

$$\hat{f}(\mathbf{X}) = \frac{1}{nh_1 \cdots h_d} \sum_{i=1}^{n} \prod_{j=1}^{d} K\left(\frac{X_i - x_{ij}}{h_j}\right).$$

Scott's bandwidth multivariate normal reference rule for d-dimensional data

$$\hat{h}_i = \hat{\sigma}_i n^{-1/(d+4)}.$$

Multivariate KDE : Bivariate Normal Example

This example plots the density estimate for a bivariate normal location mixture using kde2d in MASS package.

The mixture has three components with different mean vectors and identical variance $\Sigma = I_2$. The mean vectors are

$$\mu_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \ \mu_2 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}, \ \mu_3 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

and the mixing probabilities are $p = (0.2, 0.3, 0.5)$.

There is an R function called {kde2d} (Two-Dimensional Kernel Density Estimation), available in the MASS library package.

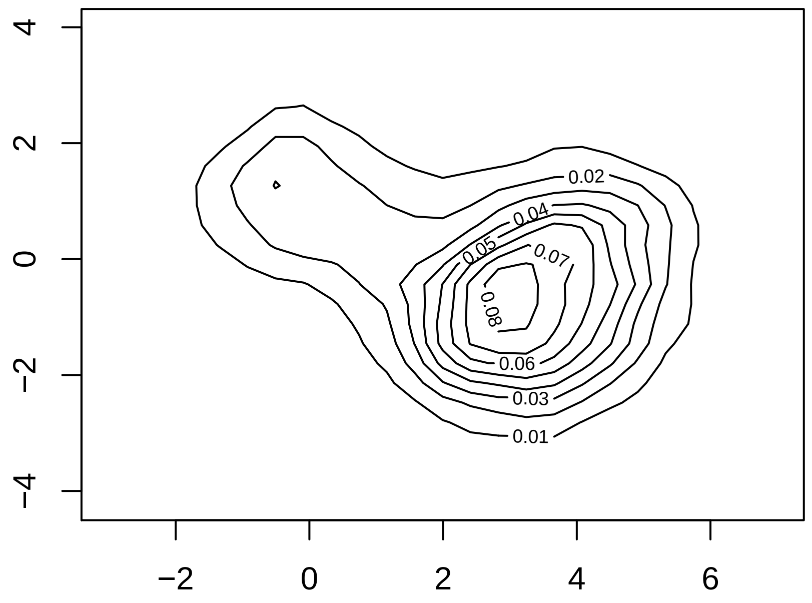
The function's description says it is a *Two-dimensional kernel density estimation with an axis-aligned bivariate normal kernel, evaluated on a square grid* - see help(kde2d).

bandwidth.nrd is another function in the MASS package for choosing the bandwidth of a Gaussian kernel density estimator.

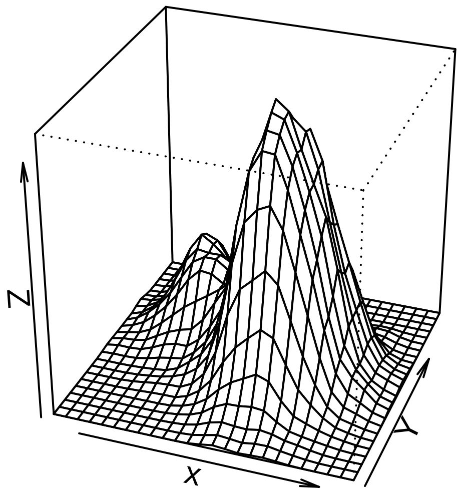
```r
library(MASS)  #for mvrnorm and kde2d
#generate the normal mixture data
n <- 2000
p <- c(.2, .3, .5)
mu <- matrix(c(0, 1, 4, 0, 3, -1), 3, 2, byrow=T)
Sigma <- diag(2)
i <- sample(1:3, replace = TRUE, prob = p, size = n)
k <- table(i)
x1 <- mvrnorm(k[1], mu = mu[1,], Sigma)
x2 <- mvrnorm(k[2], mu = mu[2,], Sigma)
x3 <- mvrnorm(k[3], mu = mu[3,], Sigma)
X <-  rbind(x1, x2, x3)   #the mixture data
x <- X[,1]  ; y <- X[,2]
# computes the bandwith for x and y
data.frame(h_x = bandwidth.nrd(x), h_y = bandwidth.nrd(y))
```

```
##        h_x      h_y
## 1 1.532696 1.176651
```

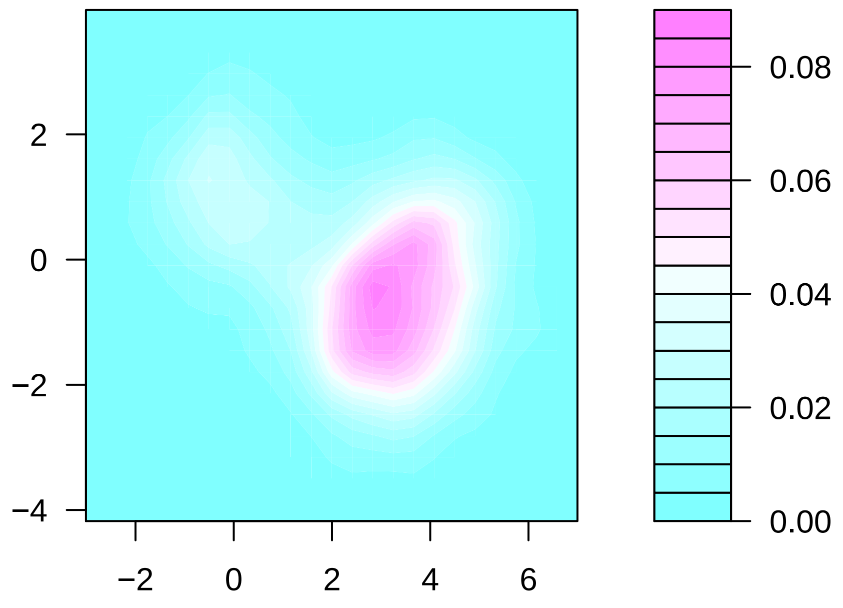
```r
# accepting the default normal reference bandwidth
fhat <- kde2d(x, y)
contour(fhat)
```



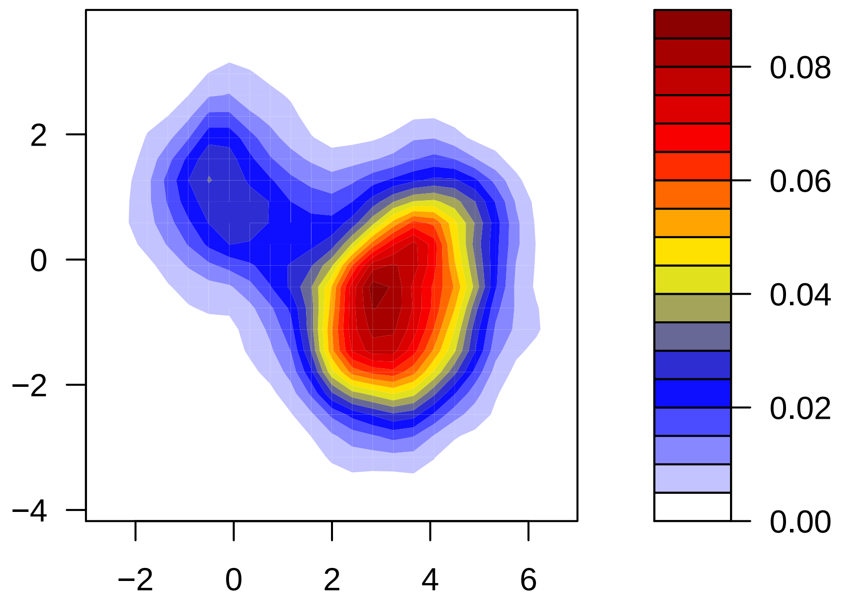
```r
persp(fhat, phi = 30, theta = 20, d = 5, xlab = "x")
```



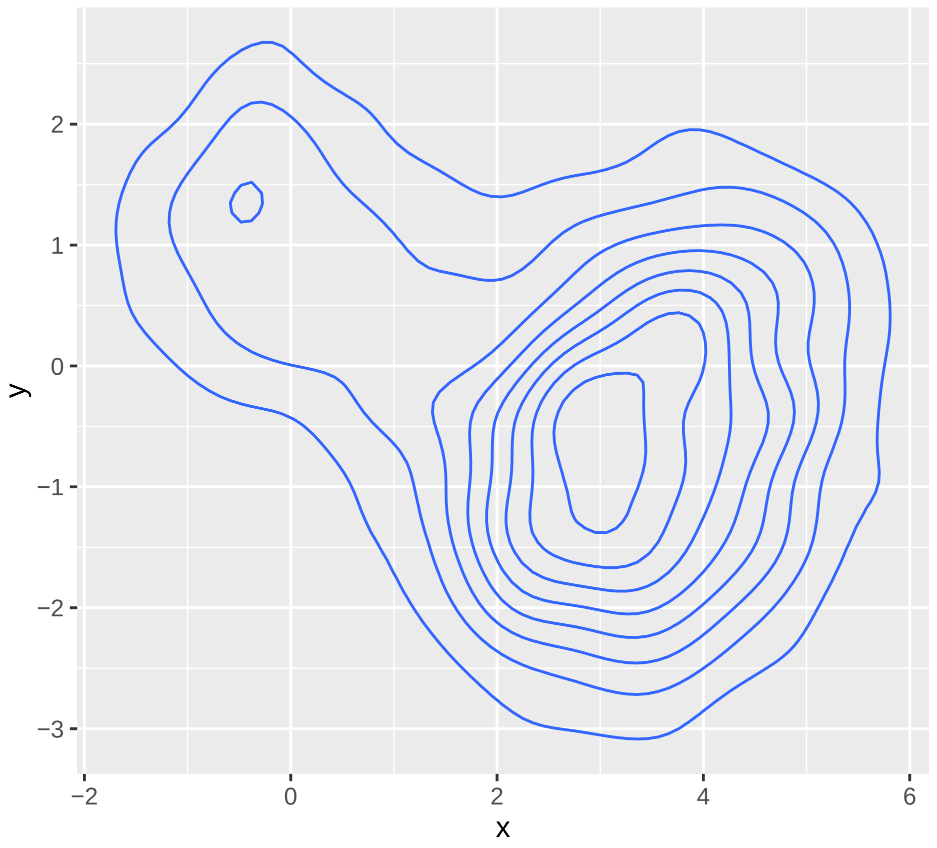
```
filled.contour(fhat)
```



```
filled.contour(fhat,color.palette=
    colorRampPalette(c('white','blue',
    'yellow','red','darkred')))
```



```
# Contour Plots using ggplot package in R

df <- data.frame(x,y)
ggplot(df,aes(x=x,y=y))+geom_density2d()
```


2
1
0
−1
−2
−3
y
−2
0
2
4
6
x