10.1 Generate $n = 100$ standard lognormal data points via the R commands

```r
set.seed(1001)
n = 100; x = rlnorm(n)
```

Sturges’ rule then gives a bandwidth of $h = 3.8528$:

```r
R = diff(range(x))
h.Sturges = R/(1 + log2(n)); print(h.Sturges)
[1] 3.852815
nclass.Sturges(x)   # same as: ceiling(R/h.Sturges)
[1] 8
```

The default histogram in R ‘pretties’ this up to $h = 5$:

```r
hist( x, breaks='Sturges', prob=T, main='' )
```

which is only slightly informative, visually: is $f(x)$ a simple exponential, or unimodal with a right skew as with lognormal or gamma? (The breaks are clearly at $x = 0, 5, 10, 15, 20, 25, 30$.)
10.1 (cont’d)

Applying Doane’s correction requires the sample skewness. Write this as a function:

```r
skew <- function(x) {
  xbar <- mean(x)
  m3 <- mean((x - xbar)^3)
  m2 <- mean((x - xbar)^2)
  return(m3/m2^1.5)
}
```

Now build the Doane constant:

```r
absb1 = abs( skew(x) )
se = sqrt( 6 * (n - 2)/((n + 1) * (n + 3)) )
Ke = log2( 1 + absb1/se )
nclass.Doane = ceiling( nclass.Sturges(x) + Ke ); print(nclass.Doane)
[1] 13
h.Doane = R/nclass.Doane; print(h.Doane)
[1] 2.265412
```

As expected, the bandwidth drops, to \( h = 2.2654 \), and we have more class intervals (13, vs. 8) with which to work. Enter these directly in R:

```r
breaks.Doane = min(x) + h.Doane*seq(0,nclass.Doane)
round(breaks.Doane, 2)
[1]  0.08  2.35  4.61  6.88  9.14 11.41 13.67 15.94 18.20 20.47
[11] 22.74 25.00 27.27 29.53
hist( x, breaks=breaks.Doane, prob=T, main='' )
```

This is more informative visually, but still doesn't convey strong information on the nature of the density near \( x = 0 \).

(continues)
10.1 (cont’d)

The deciles of the standard lognormal, \( d_q (q = 1, \ldots, 9) \), are found via

```r
decile.true = qlnorm( seq(.1,.9,.1) )
round( decile.true, 3 )
```

\[
[1] 0.278 0.431 0.592 0.776 1.000 1.288 1.689 2.320 3.602
\]

while \( f(d_q) \) is

```r
f.true10 = dlnorm( decile.true )
round( f.true10, 3 )
```

\[
[1] 0.632 0.650 0.587 0.498 0.399 0.300 0.206 0.121 0.049
\]

Recall the break points for the (pretty) Sturges histogram

```r
hist( x, breaks='Sturges', plot=F )$breaks
```

\[
[1] 0 5 10 15 20 25 30
\]

and for the Doane histogram:

```r
round(breaks.Doane, 2)
```

\[
[1] 0.08 2.35 4.61 6.88 9.14 11.41 13.67 15.94 18.20 20.47
[11] 22.74 25.00 27.27 29.53
\]

Notice that for the (pretty) Sturges breaks, all of the true deciles lie below the first break point; i.e., \( d_1 < d_2 < \cdots < d_9 = 3.602 < 5 \). Thus the Sturges histogram will estimate every value of \( f(d_q) \) as the same number: the height of the first class interval’s bar, which is

```r
hist( x, breaks='Sturges', plot=F )$density[1]
```

\[
[1] 0.18
\]

Similarly, for the Doane breaks all true deciles but the last, \( d_9 = 3.602 \), lie below the first class interval’s break at \( x = 2.42 \). Thus the Doane histogram will estimate every value of \( f(d_q), q = 1, \ldots, 8 \), as the same number: the height of the first class interval’s bar at

```r
hist( x, breaks=breaks.Doane, plot=F )$density[1]
```

\[
[1] 0.3487224
\]

except the final value of \( f(d_9) \), which is estimated as the height of the second class interval’s bar:

```r
hist( x, breaks=breaks.Doane, plot=F )$density[2]
```

\[
[1] 0.03972786
\]

Thus, as mentioned above, these histograms provide poor estimates of the true density near \( x = 0 \).

10.7. Load the `precip` dataset via

```r
data( precip )
```

and use the `ash` package:

```r
require( ash )
```

(continues)
10.7. (cont’d)

Begin with Scott’s normal reference rule for the bandwidth, \( h \),

\[
R = \text{diff( range(precip) )}
\]
\[
n = \text{length( precip )}
\]
\[
h = 3.49 * \text{sd( precip )}/n^{(1/3)} \quad \text{#bin width from Scott's rule}
\]
\[
totbins = \text{ceiling}(R*20/h) \quad \text{#total number of bins}
\]

then build the bins via the \texttt{bins1()} function and the ASH object via the \texttt{ash1()} function in \texttt{ash}:

\[
xinterval = c(0,80)
\]
\[
xbins = \text{bin1( precip, ab=xinterval, nbin=totbins )}
\]
\[
fASH = \text{ash1( bins=xbins, m=10, kopt=c(2,2) )} \quad \text{#biweight kernel}
\]
\[
\text{plot( fASH, type='l', ylab=expression(f[ASH]), xlab='Precipitation', xlim=xinterval, ylim=c(0,0.04) )}
\]

For comparison, overlay the (pretty) Scott histogram:

\[
\text{par( new=T )}
\]
\[
\text{hist( precip, prob=T, breaks='scott', main='', xlab='', ylab='', xlim=xinterval, ylim=c(0,0.04) )}
\]

(plot follows →)
10.7. (cont’d)

10.8. Load the buffalo dataset from the gss package

```r
library( gss )
data( buffalo )
```

For kernel density estimates (KDEs) in the R `density()` function, two recommended bandwidth algorithms are the default (`bw='nrd0'`, Silverman’s rule), and the Sheather-Jones option (`bw='SJ'`). Implement these with the Gaussian kernel via the sample code

```r
buffalo.dens0 = density( buffalo, bw='nrd0', kernel='gaussian')
buffalo.dens0SJ = density( buffalo, bw='SJ', kernel='gaussian')
```

Employing instead the biweight kernel gives

```r
buffalo.densBiw = density( buffalo, bw='nrd0', kernel='biweight')
buffalo.densBiwSJ = density( buffalo, bw='SJ', kernel='biweight')
```

Plot these. First the two Gaussian-based KDEs:

```r
par( mfrow= c(1,2) )
plot( buffalo.dens0, xlab='Snowfall', xlim=c(0,150), ylim=c(0,0.018) )
plot( buffalo.dens0SJ, xlab='Snowfall', xlim=c(0,150), ylim=c(0,0.018) )
```

(plot follows →)
10.8. (cont’d)

Next the two biweight-based KDEs:

```r
par( mfrow= c(1,2) )
plot( buffalo.densBiw, xlab='Snowfall', xlim=c(0,150),
     ylim=c(0,0.018) )
plot( buffalo.densBiwSJ, xlab='Snowfall', xlim=c(0,150),
     ylim=c(0,0.018) )
```

(Cont’d)
10.8. (cont’d)

Little distinguishability is evidenced among the plots. (Moving to more extreme choices for the bandwidth would give clear differences, although the resulting KDEs may not be very useful.)

10.9. One can generate the normal mixture data as follows, for a sample of size $n = 100$:

```r
n = 100
p = 1/2
set.seed(1009)
means = sample( c(0,3), size=n, replace=T, prob=c(p,1-p) )
x = rnorm( n, mean=means, sd=1 )
```

Work exclusively with the default Gaussian kernel in the \texttt{R density()} function. For bandwidth selection, Equation (10.13) is Scott’s normal reference rule (\texttt{bw='nrd'} in \texttt{density()}), and (10.14) is Silverman’s rule (\texttt{bw='nrd0'}). Implement these with the Gaussian kernel via the sample code

```r
xmix.dens0 = density( x, bw='nrd0', kernel='gaussian')
xmix.dens0Scot = density( x, bw='nrd', kernel='gaussian')
xmix.dens0SJ = density( x, bw='SJ', kernel='gaussian')
```

Now plot. Include (first plot) the true mixture p.d.f.:

```r
par( mfrow= c(2,2) )
curve( p*dnorm(x,0,1) + (1-p)*dnorm(x,3,1), from=-3, to=7, ylab='True f(x)', ylim=c(0,0.22) )
abline ( h=0, col='gray' )
```

```r
plot( xmix.dens0, xlim=c(-4,7), ylim=c(0,0.22), main='Gaussian/nrd0' )
plot( xmix.dens0Scot, xlim=c(-4,7), ylim=c(0,0.22), main='Gaussian/Scott' )
plot( xmix.dens0SJ, xlim=c(-4,7), ylim=c(0,0.22), main='Gaussian/SJ' )
```

In the plot, which follows, we see the default Silverman rule (\texttt{bw='nrd0'}) seems to mimic the true p.d.f. closest, although Scott’s rule (\texttt{bw='nrd'}) is not much different.

(Plot follows →)
10.9. (cont’d)

Comparison plots for KDEs of normal mixture:

**Gaussian/nrd0**

![Graph 1](image1)

**Gaussian/Scott**

![Graph 2](image2)

**Gaussian/SJ**

![Graph 3](image3)
10.A. Suppose bivariate data on (log) bromide concentrations in industrial workers are observed as pairs \((x_i, y_i)\), where \(x\) = worker’s serum log-concentration at the beginning of the work-week and \(y\) is the same worker’s measurement at the end of the week. Selected values are:

<table>
<thead>
<tr>
<th>(i)</th>
<th>(x_i)</th>
<th>(y_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3364722</td>
<td>1.1939225</td>
</tr>
<tr>
<td>2</td>
<td>1.0986123</td>
<td>1.0986123</td>
</tr>
<tr>
<td>3</td>
<td>0.8329091</td>
<td>0.9932518</td>
</tr>
<tr>
<td>64</td>
<td>-1.5141277</td>
<td>-1.2378744</td>
</tr>
</tbody>
</table>

Access the data online: [http://math.arizona.edu/~piegorsch/675/ToraasonA2.csv](http://math.arizona.edu/~piegorsch/675/ToraasonA2.csv). Construct a bivariate kernel density estimate (KDE) for these data, using the normal reference bandwidths. Provide both a contour plot and a surface (‘perspective’) plot of the estimated bivariate density.

Answer: Sample R code for acquiring the data and calling the 2D KDE routine is

```r
data10A = read.csv( file.choose() )
attach( data10A )
library(MASS)

c( bandwidth.nrd(x), bandwidth.nrd(y) )  #gives Normal ref. bandwidths
f2hat = kde2d(x, y)                      #default b.w. is Normal reference

The contour plot follows via

contour( f2hat )
```
10.A. (cont’d)
An elongated (high correlation) relationship appears, with two clusters/modes. One can also try color-coded contours:

\[ \text{filled.contour( f2hat , color.palette=terrain.colors )} \]

A surface plot gives similar visualization:

\[ \text{persp( f2hat, phi = 30, theta = 75, d = 5, xlab = "x", ylab="y" )} \]

(plot follows →)
10.A. (cont’d)
Surface plot of 2D KDE: