bivariate 0.3.0

Bivariate Probability Distributions

Abby Spurdle

December 10, 2018

Contains convenience functions for constructing and plotting bivariate probability distributions (probability mass functions, probability density functions and cumulative distribution functions). Currently, supports uniform (discrete and continuous), binomial, Poisson, normal, bimodal and kernel distributions.

Pre-Intro

This package uses objects, which are also functions, which are also models, which are also probability distributions.

Some functions (constructors) return other functions (models), which can be evaluated. The resulting functions have attributes (which along with their arguments) determine their return values.

This is intermediate between a standard object oriented approach and a functional approach. And I believe that this is the best approach for implementing probability distributions, especially nonparametric probability distributions.

Introduction

This package imports and uses the barsurf package, which contains the plot3d.bar() and plot3d.surf() functions.

Currently, there are convenience functions for constructing and plotting:

1. Discrete bivariate uniform distributions.
2. Bivariate binomial distributions.
3. Bivariate Poisson distributions*.
4. Continuous bivariate uniform distributions.
5. Bivariate normal distributions*.
6. Bivariate bimodal distributions*.
7. Bivariate kernel distributions*.
   (Probability density functions and cumulative distribution functions resulting from kernel smoothing).

Some of these distributions are simply the product of their marginal distributions. Others, marked with an * are not necessarily so.

We can compute either their probability mass function (PMF) or their probability density function (PDF), and their cumulative distribution function (CDF). Note that I think we should recognize bivariate CDFs, which don't appear in probability and statistics literature frequently. Refer to Appendix 1 for information on how to compute bivariate probabilities.

The functions for constructing probability distributions take some parameters and return functions which can be evaluated for x and y, except for kernel distributions. Note that discrete probability distributions convert their arguments to integers before evaluation.

The functions for plotting discrete distributions take a function object and plot a 3d bar plot. The functions for plotting continuous distributions take a function object and can plot either a 2d contour plot or a 3d surface plot.

Refer to the barsurf package for information on how to change colors, if required.

Note that I'm considering alternative formulations of bivariate binomial distributions. And it's possible that I may support bivariate-like (but trivariate) Dirichlet distributions in the near future.

Important Notes

Here, PMFs and PDFs are equivalent to R's "d" functions. And CDFs are equivalent to R's "p" functions. R's "q" and "r" functions are not supported.

Loading The Packages

I'm going to load (and attach) the intoo, bivariate and moments packages:

```
> library (intoo)
> library (bivariate)
> library (moments)
```

Discrete Bivariate Uniform Distributions (and Core Functionality)

We can construct a discrete bivariate uniform probability mass function as the product of two discrete univariate uniform probability mass functions.

We can use the dubvpmf() function. It takes four arguments, the a and b values of X and the a and b values of Y.

```
> f = dubvpmf (1, 10, 1, 10)
```

We can print the object directly, however, I recommend using the object.info() function from the intoo package.

```
> object.info (f)

value, 1
function (x, y)
```

```
{
    x = as.integer(x)
    y = as.integer(y)
    stopifnot(length(x) == length(y))
    .dubvpmf.eval(x, y)
}
class, 1
[1] "dubvpmf"
%$%n, numeric, 2
[1] 10 10
%$%a, numeric, 2
[1] 1 1
%$%b, numeric, 2
[1] 10 10
```

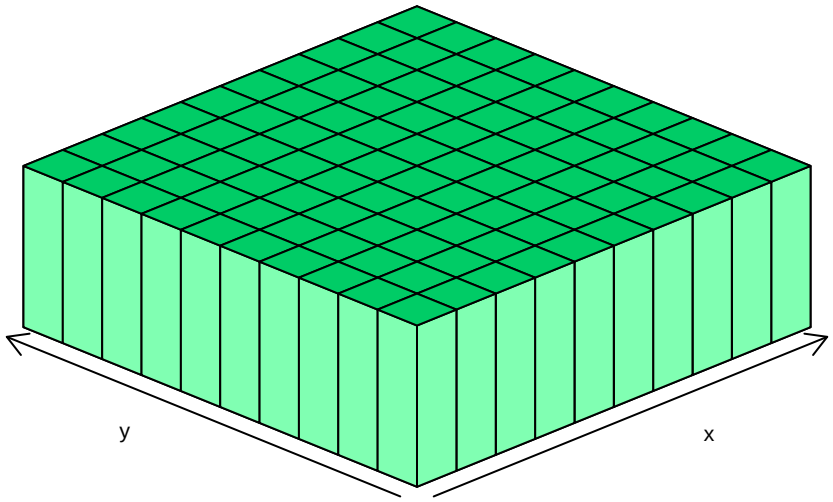
We can access a single attribute using the attribute operator, also from the intoo package, if required.

```
> f %$% n
```

```
[1] 10 10
```

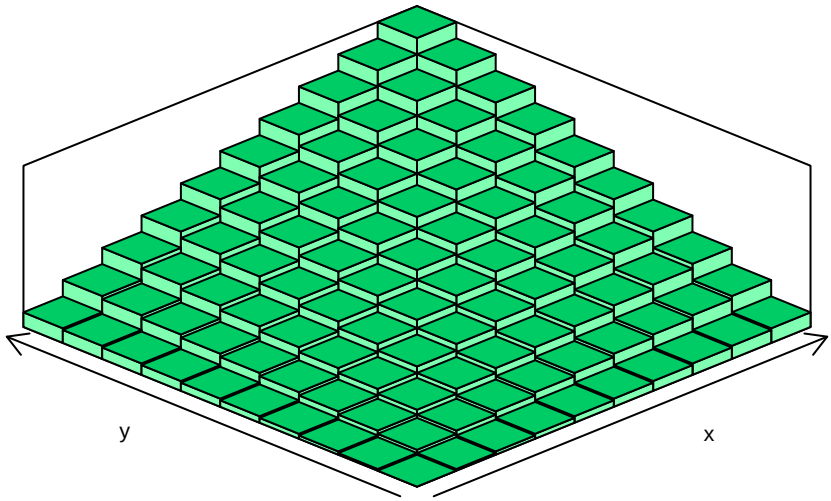
We can plot the probability mass function.

```
> plot (f)
```



To construct a cumulative distribution function, we can use the dubvcdf() function. It takes the same arguments as dubvpmf().

```
> F = dubvcdf (1, 10, 1, 10)
> plot (F)
```



In both cases, we can evaluate the function for x and y.

```
> f (1, 1)
```

```
[1] 0.01
```

The same applies to all the other probability distributions in this package, except for kernel distributions.

Bivariate Binomial Distributions

One way to define a bivariate binomial distribution is to say that we have n trials. In each trial there are two independent events, each with a particular probability of success. Like flipping two coins, n times.

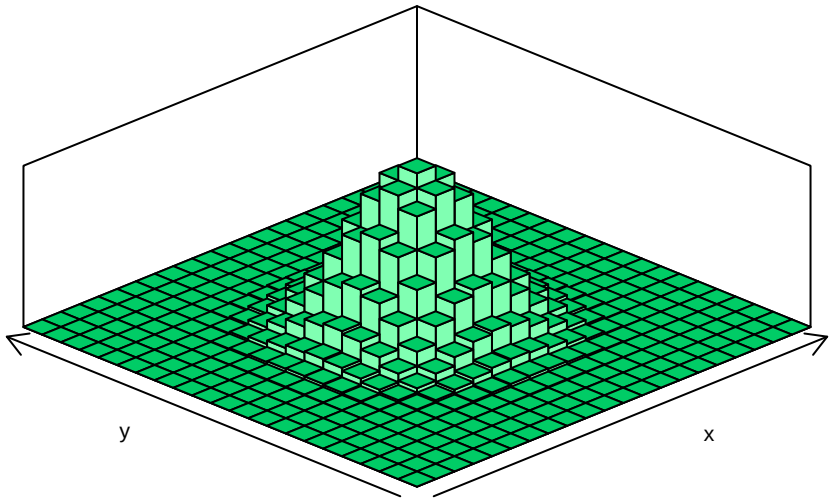
Like the discrete bivariate uniform probability mass function, we can construct a bivariate binomial probability mass function as the product of two univariate binomial probability mass functions, so:

$$\begin{aligned}\mathbb{P}(X = x, Y = y) = f_{X,Y}(x, y) &= f_X(x) f_Y(y) \\ &= \left[\binom{n}{x} p_X^x (1 - p_X)^{n-x}\right]\left[\binom{n}{y} p_Y^y (1 - p_Y)^{n-y}\right]\end{aligned}$$

Where p_X is the probability of the first success and p_Y is the probability of the second success.

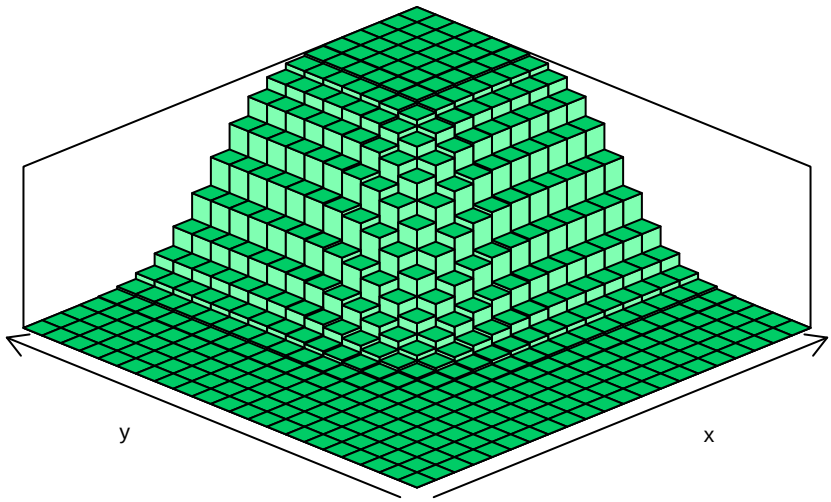
We can use the bnbvpmf() function. It takes three arguments, the number of trials and the probability of each success.

```
> f = bnbvpmf (20, 0.5, 0.5)
> plot (f)
```



To construct a cumulative distribution function, we can use the bnbvcdf() function. It takes the same arguments as bnbvpmf().

```
> F = bnbvcdf (20, 0.5, 0.5)
> plot (F)
```



Bivariate Poisson Distributions*

Based on Karlis and Ntzoufras (2003) we can define the bivariate Poisson probability mass function as:

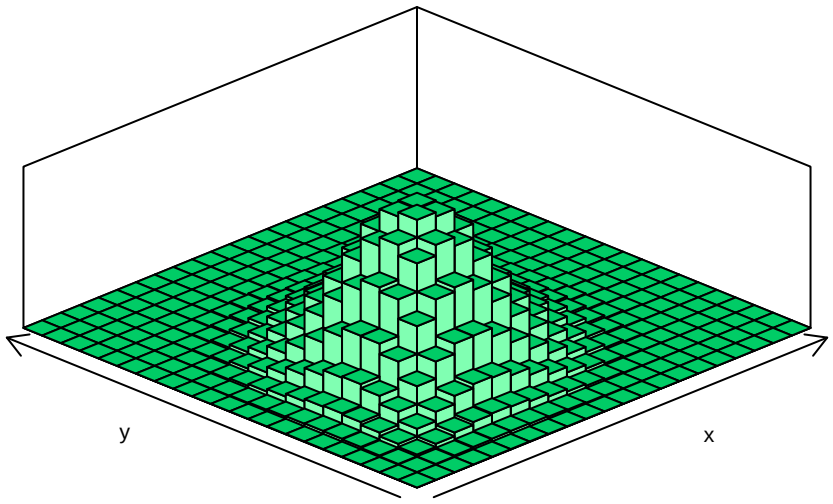
$$\mathbb{P}(X = x, Y = y) = f_{X,Y}(x, y) = e^{-(\lambda_1+\lambda_2+\lambda_3)} \frac{\lambda_1^x}{x!} \frac{\lambda_2^y}{y!} \sum_k \binom{x}{k} \binom{y}{k} k! \left(\frac{\lambda_3}{\lambda_1 \lambda_2} \right)^k$$

Where k is in 0 to $\min(x, y)$.

And where:
$\mathbb{E}(X) = \text{var}(X) = \lambda_1 + \lambda_3$
$\mathbb{E}(Y) = \text{var}(Y) = \lambda_2 + \lambda_3$
$\text{cov}(X, Y) = \lambda_3$

We can use the pbvpmf() function. It takes three arguments, the mean of X, the mean of Y and the covariance between X and Y.

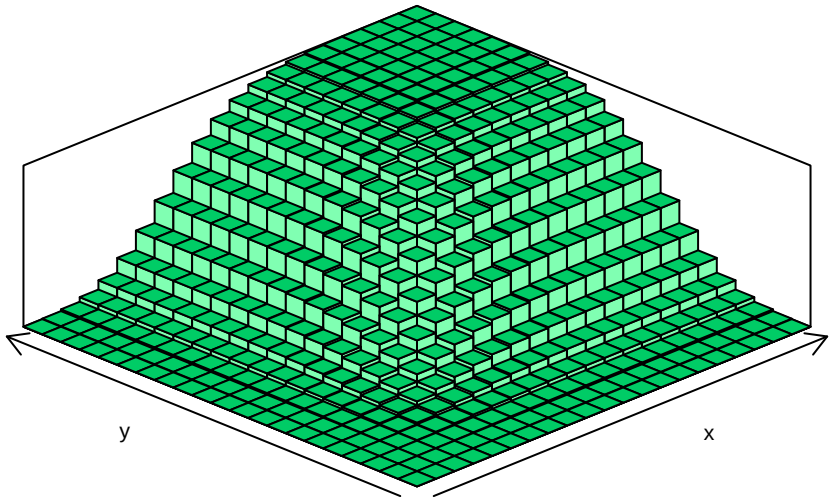
```
> f = pbvpmf (8, 8, 2)
> plot (f)
```



Note that λ_1 and λ_2 need to be greater than zero, which means that $\mathbb{E}(X)$ and $\mathbb{E}(Y)$ need to be greater than $\text{cov}(X, Y)$.

To construct a cumulative distribution function, we can use the pbvcdf() function. It takes the same arguments as pbvpmf().

```
> F = pbvcdf (8, 8, 2)
> plot (F)
```

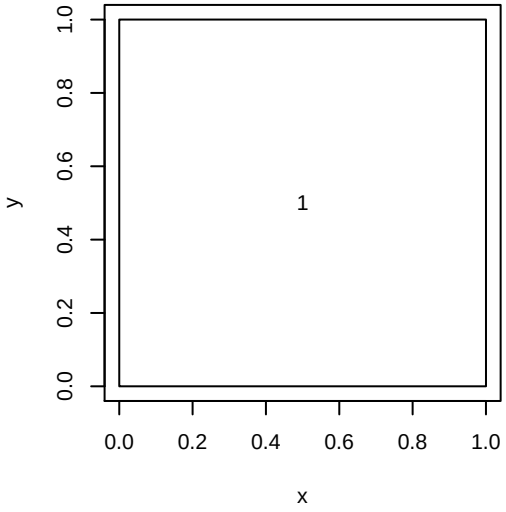


Continuous Bivariate Uniform Distributions

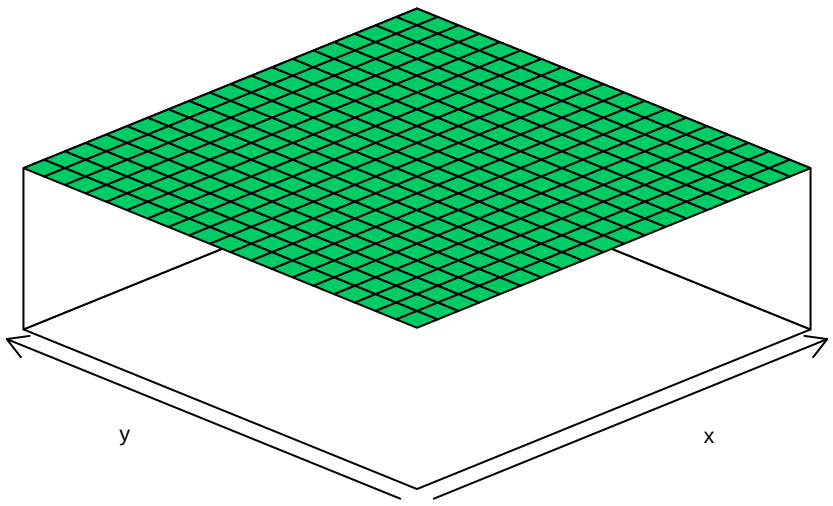
Continuous bivariate uniform distributions are similar to discrete bivariate uniform distributions. However, we have a probability density function instead of a probability mass function.

We can use the cubvpdf() function. It takes four arguments, the a and b values of X and the a and b values of Y.

```
> f = cubvpdf (0, 1, 0, 1)
> plot (f)
```

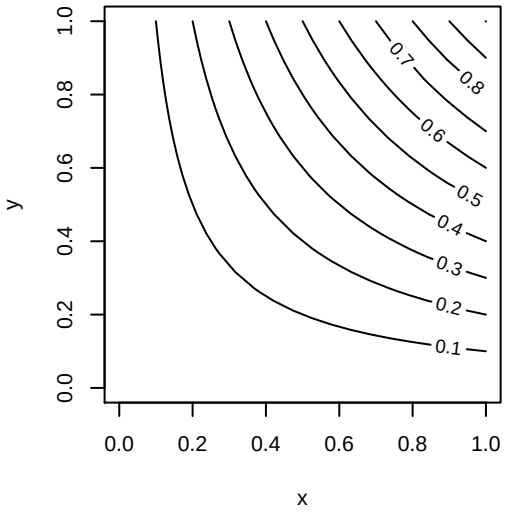



```
> plot (f, TRUE)
```

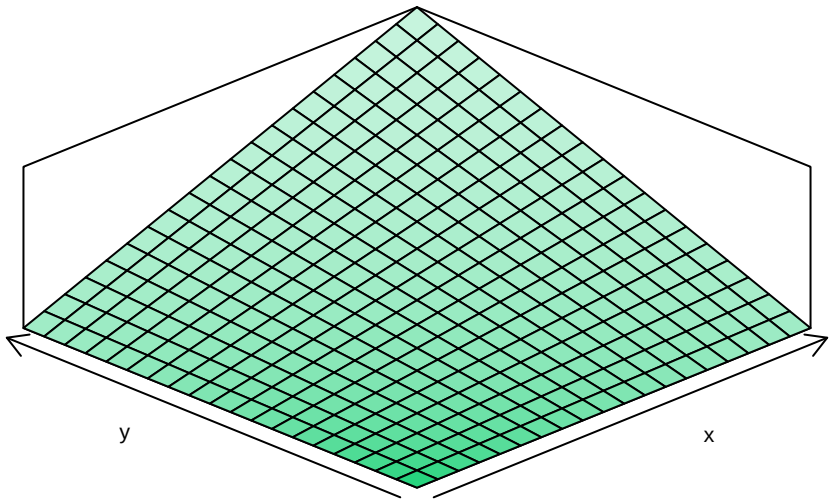


To construct a cumulative distribution function, we can use the cubvcdf() function. It takes the same arguments as cubvpdf().

```
> F = cubvcdf (0, 1, 0, 1)
> plot (F)
```




```
> plot (F, TRUE)
```



Bivariate Normal Distributions*

Hothorn, Bretz, Genz, Mi and Miwa (2018) provide the mvtnorm package, which is imported and used by this package.

We can construct a normal bivariate probability density function using the nbvpdf() function. It takes five arguments, the mean of X, the mean of Y, the variance of X, the variance of Y and the covariance of X and Y.

```
> f = nbvpdf (0, 0, 1, 1, 0)
```

Alternatively we could use the standard deviations and correlation, using something like:

```
> #f = nbvpdf (mean.x, mean,y, sd.x ^ 2, sd.y ^ 2, sd.x * sd.y * cor.xy)
```

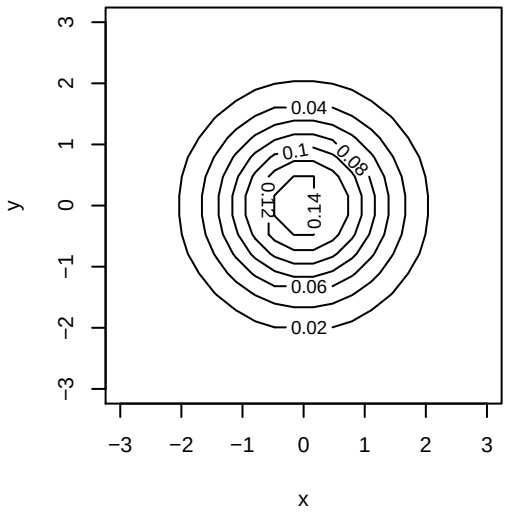
It may be useful to print our object, here.

```
> object.info (f)
```

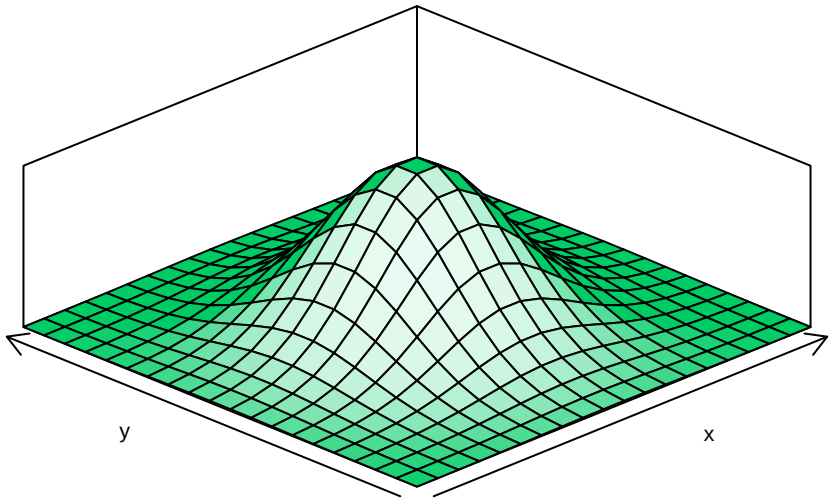
```
value, 1
function (x, y)
{
    x = as.numeric(x)
    y = as.numeric(y)
    stopifnot(length(x) == length(y))
    .nbvpdf.eval(x, y)
}
class, 1
[1] "nbvpdf"
%$%vector.means, numeric, 2
[1] 0 0
%$%matrix.variances, matrix, 2 * 2
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

Once we have constructed our object we can plot it.

```
> plot (f)
```




```
> plot (f, TRUE)
```



We can construct a cumulative distribution function using the nbvcdf() function. It takes the same arguments as nbvpdf().

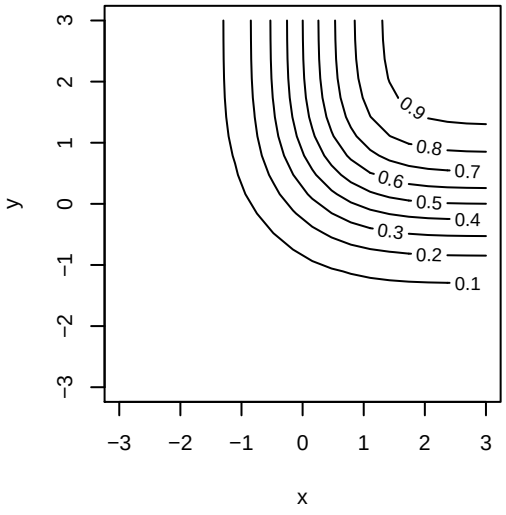
```
> F = nbvcdf (0, 0, 1, 1, 0)
```

Alternatively we could use the standard deviations and correlation, using something like:

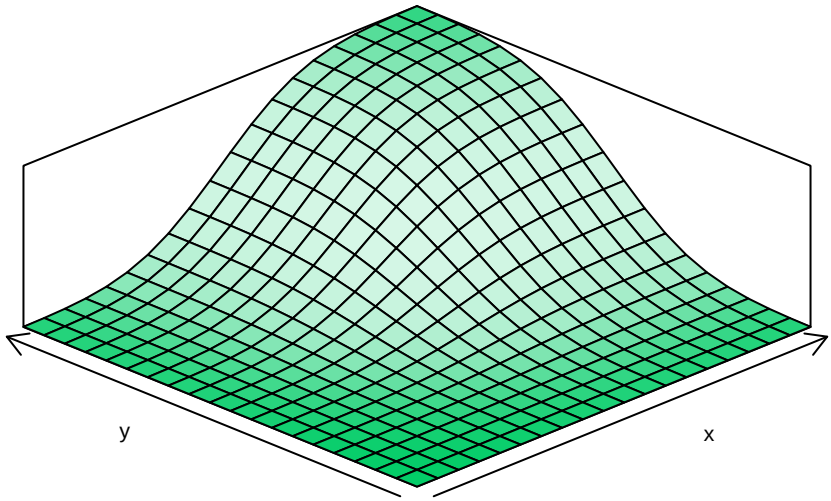
```
> #F = nbvcdf (mean.x, mean,y, sd.x ^ 2, sd.y ^ 2, sd.x * sd.y * cor.xy)
```

Once we have constructed our object we can plot it.

```
> plot (F)
```




```
> plot (F, TRUE)
```

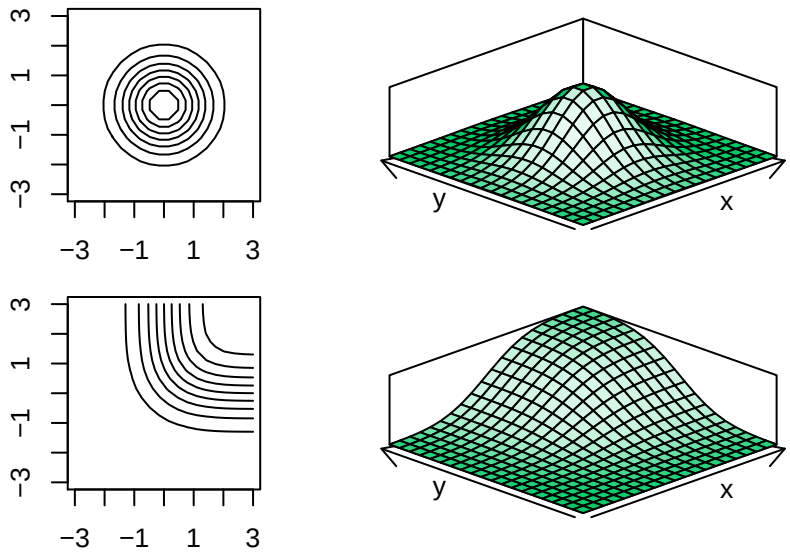


Comparing Normal Distributions

We can compare different normal distributions using different covariance parameters.

First, let's consider the bivariate distributions from the previous section with zero correlation/covariance.

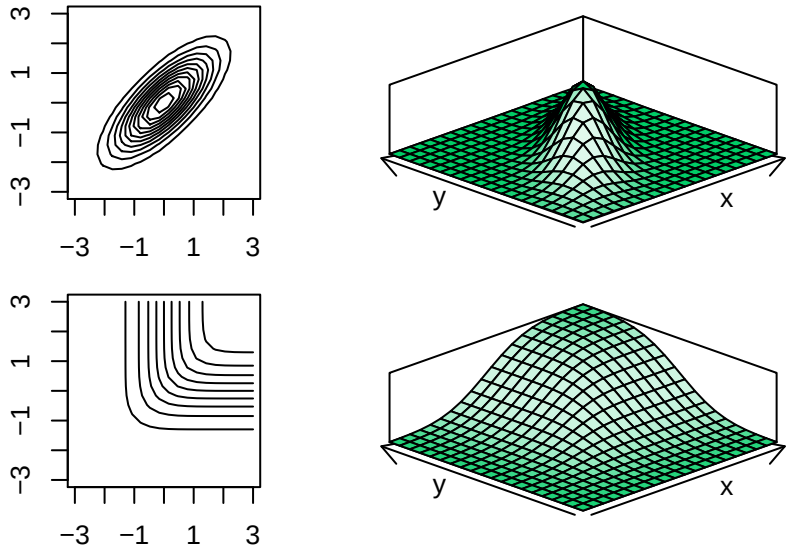
```
> plot (f, all=TRUE)
```



Note that this requires the probability density function rather than the cumulative distribution function.

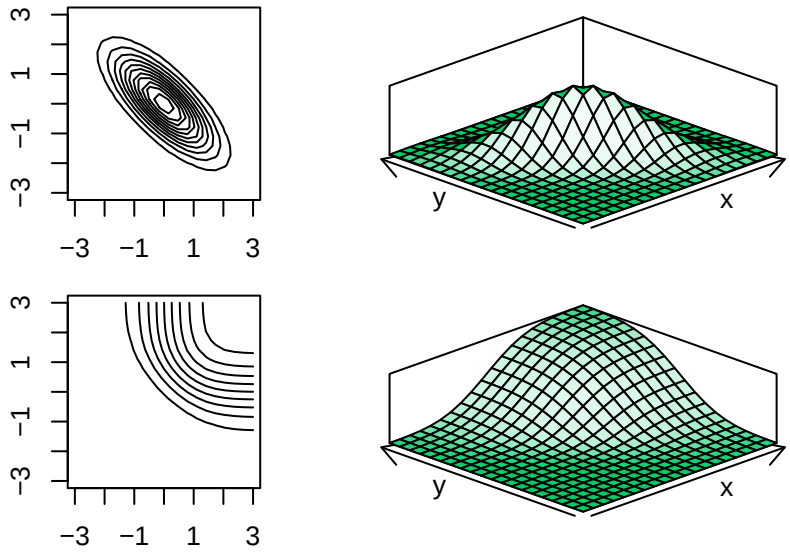
Second, let's consider bivariate distributions with positive correlation/covariance.

```
> f2 = nbvpdf (0, 0, 1, 1, 0.75)
> plot (f2, all=TRUE)
```



Third, let's consider bivariate distributions with negative correlation/covariance.

```
> f3 = nbvpdf (0, 0, 1, 1, -0.75)
> plot (f3, all=TRUE)
```



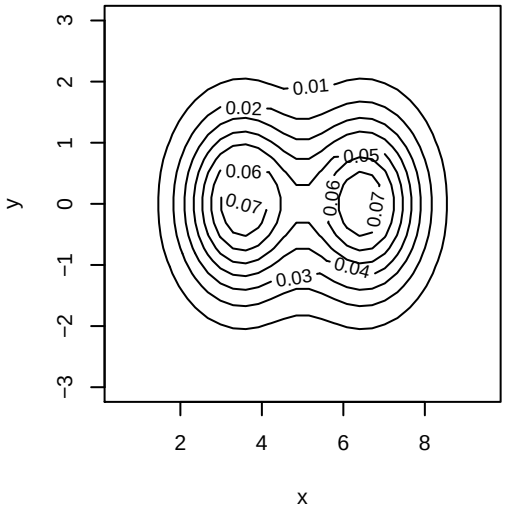
Bivariate Bimodal Distributions*

One way to construct a bivariate bimodal probability density function is to construct two bivariate normal probability density functions, then add them together and then divide by two. This can be generalized by allowing any two weights which sum to one.

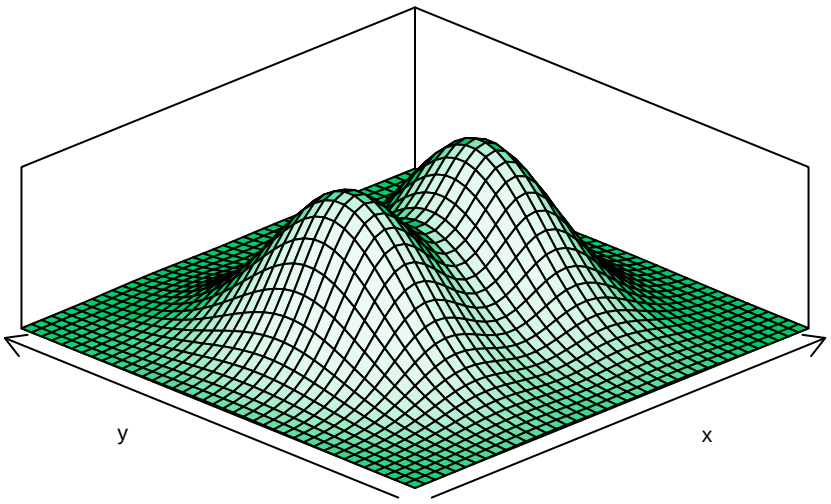
We can construct such a probability density function using the bmbvpdf() function. It takes eight (or nine) arguments. The mean of X, the mean of Y, the variance of X and the variance of Y for the first component distribution. And the mean of X, the mean of Y, the variance of X and the variance of Y for the second component distribution. Optionally, we can provide another argument for the weight of the first component distribution in the

interval (0, 1).

```
> f = bmbvpdf (3.5, 0, 1, 1, 6.5, 0, 1, 1)
> plot (f)
```

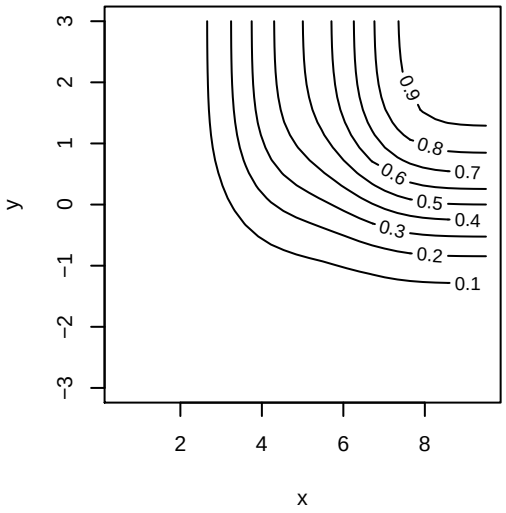



```
> plot (f, TRUE, npoints=40)
```

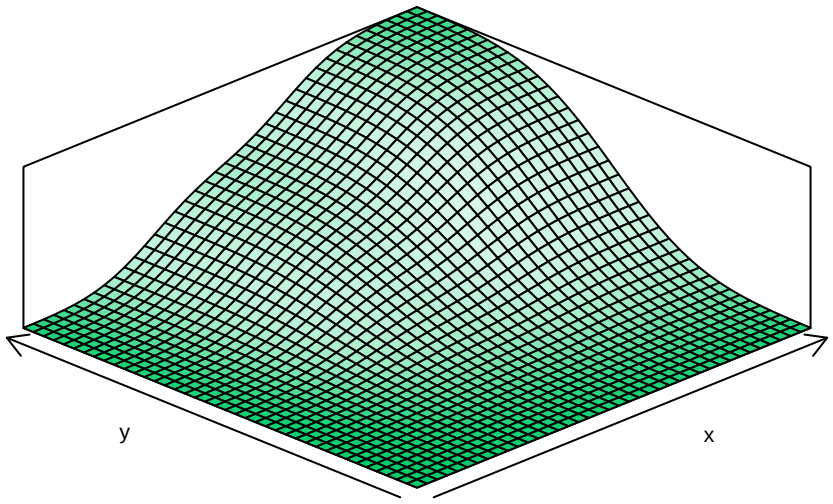


We can construct a cumulative distribution function using the bmbvcdf() function. It takes the same arguments as bmbvpdf().

```
> F = bmbvcdf (3.5, 0, 1, 1, 6.5, 0, 1, 1)
> plot (F)
```




```
> plot (F, TRUE, npoints=40)
```

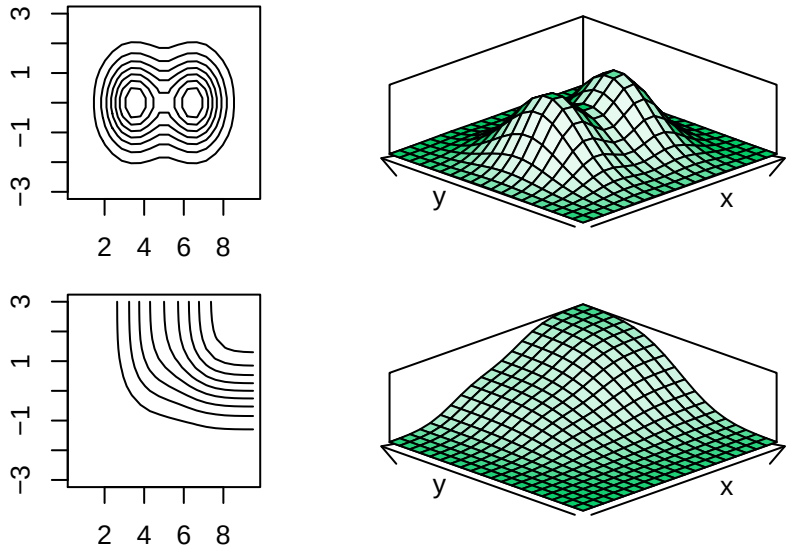


Comparing Bimodal Distributions

We can compare different bimodal distributions using different covariance structures.

First, let's consider the bimodal distributions from the previous section.

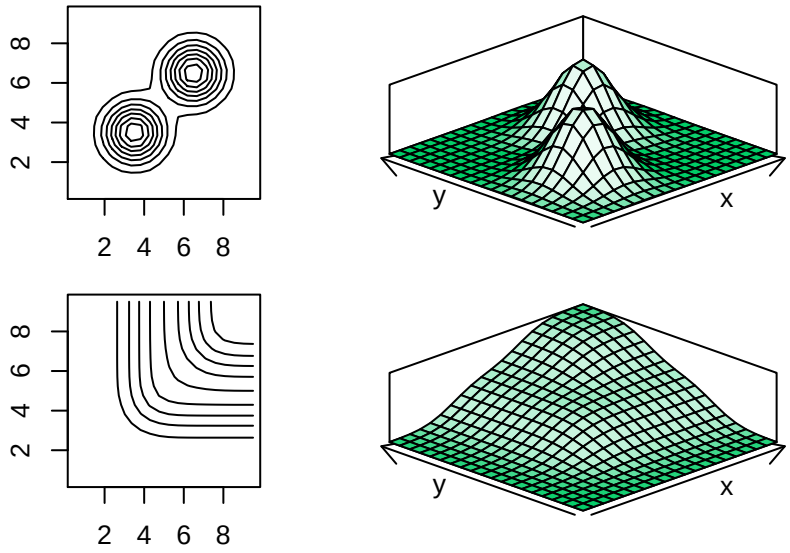
```
> plot (f, npoints=20, all=TRUE)
```



Note that this requires the probability density function rather than the cumulative distribution function.

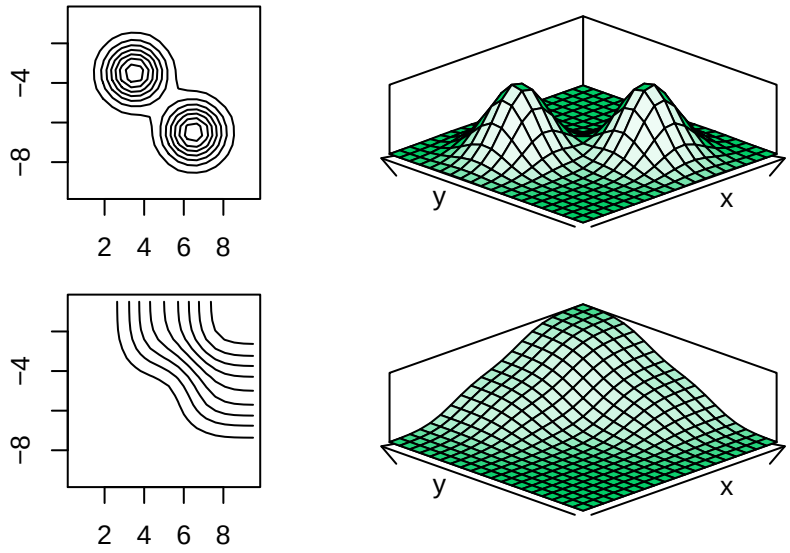
Second, let's consider bimodal distributions with a positive covariance structure.

```
> f2 = bmbvpdf (3.5, 3.5, 1, 1, 6.5, 6.5, 1, 1)
> plot (f2, npoints=20, all=TRUE)
```



Third, let's consider bimodal distributions with a negative covariance structure.

```
> f3 = bmbvpdf (3.5, -3.5, 1, 1, 6.5, -6.5, 1, 1)
> plot (f3, npoints=20, all=TRUE)
```



Bivariate Kernel Distributions*

Ripley and Wand (2015) provide the KernSmooth package.

This package imports and uses the bkde2D() function from that package, which has it's own example. Note that I'm using the same bandwidth parameters and geyser data (from the MASS package) as it's example. Refer to Appendix 2 for more information on the geyser data.

We can use the kbvpdf() function to produce bivariate kernel probability density functions. The first two arguments are x and y vectors of data. The second two arguments are the bandwidths.

```
> data ("geyser", package="MASS")
> attach (geyser)

> f = kbvpdf (duration, waiting, 0.7, 7)
```

Again, it may be useful to print our object, here.

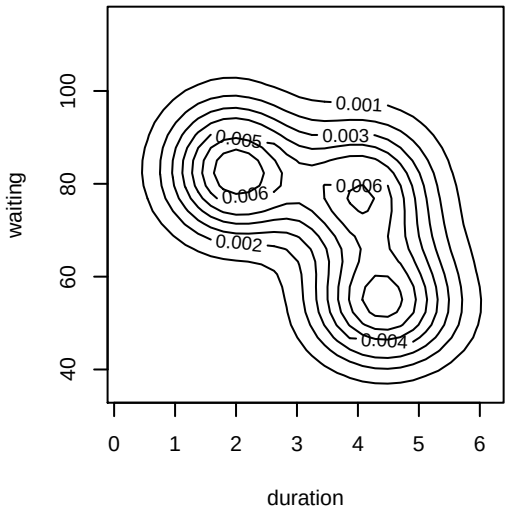
```
> object.info (f)

value, 1
function (x, y)
{
    NULL
}
class, 1
[1] "kbvpdf"
%$%bw, numeric, 2
[1] 0.7 7.0
%$%data, matrix, 299 * 2 (1)
         [,1] [,2]
[1,] 4.016667   80
[2,] 2.150000   71
[3,] 4.000000   57
[4,] 4.000000   80
```

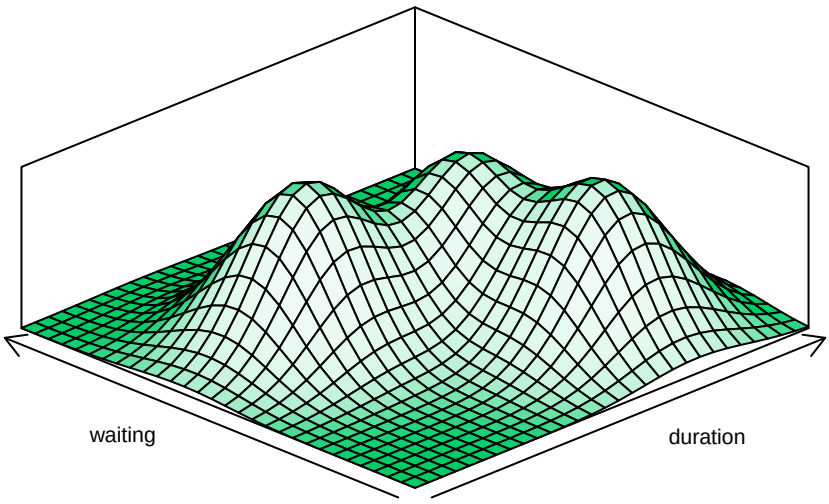
```
[5,] 4.000000   75
[6,] 2.000000   77
[7,] 4.383333   60
[8,] 4.283333   86
```

And again, once we have constructed our object we can plot it.

```
> plot (f, xlab="duration", ylab="waiting")
```

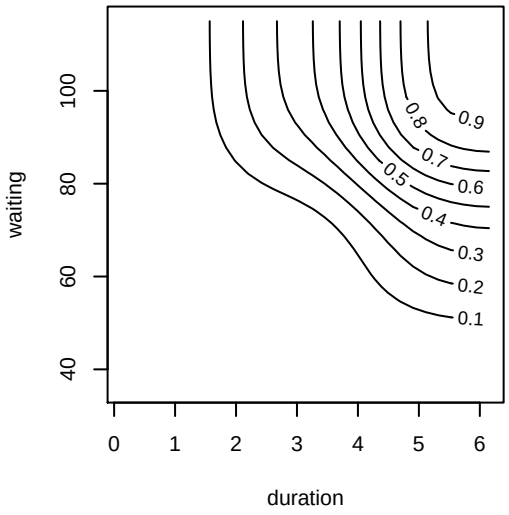



```
> plot (f, TRUE, xlab="duration", ylab="waiting")
```

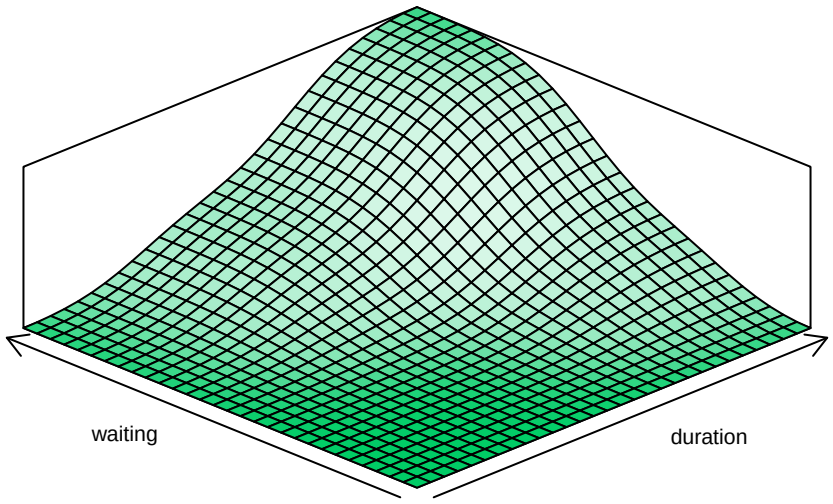


We can use the kbvcdf() function to produce bivariate kernel cumulative distribution functions.

```
> F = kbvcdf (duration, waiting, 0.7, 7)
> plot (F, xlab="duration", ylab="waiting")
```




```
> plot (F, TRUE, xlab="duration", ylab="waiting")
```



Reiterating, we can't evaluate these functions. Refer to the empirical package to evaluate a bivariate PDF or CDF resulting from bivariate kernel smoothing.

References

Spurdle, A. (2018). intoo: Object Oriented Intelligence.

Spurdle, A. (2018). barsurf: Minimal Bar Plots and Surface Plots in 3D.

Hothorn, T., Bretz, F., Genz, A., Mi, X. & Miwa, T. (2018). mvtnorm: Multivariate Normal and t Distributions.

Ripley, B. & Wand, M. (2015). KernSmooth: Functions for Kernel Smoothing Supporting Wand & Jones (1995).

Spurdle, A. (2018). empirical: Probability Distributions as Models of Data.

Komsta, L. & Novomestky, F. (2015). moments: Moments, cumulants, skewness, kurtosis and related tests.

Ripley, B. (2018). MASS: Support Functions and Datasets for Venables and Ripley's MASS.

Karlis, D. & Ntzoufras, I. (2003). Analysis of sports data by using bivariate Poisson models.

Appendix 1
Computing Bivariate Probabilities

We can compute the probability that two discrete random variables, X and Y, are between two pairs of values, (x_1 and x_2) and (y_1 and y_2), using the following expression:

$$\mathbb{P}(x_1 \leq X \leq x_2, y_1 \leq Y \leq y_2) = F(x_2, y_2) - [F(x_1 - 1, y_2) + F(x_2, y_1 - 1)] + F(x_1 - 1, y_1 - 1)$$

And we can compute the probability that two continuous random variables, X and Y, are between two pairs of values, (x_1 and x_2) and (y_1 and y_2), using the following expression:

$$\mathbb{P}(x_1 \leq X \leq x_2, y_1 \leq Y \leq y_2) = F(x_2, y_2) - [F(x_1, y_2) + F(x_2, y_1)] + F(x_1, y_1)$$

Note that these aren't necessarily the most efficient ways of doing it.

Appendix 2
geyser Data (from the MASS Package)

```
> str (geyser)

'data.frame':        299 obs. of  2 variables:
 $ waiting : num  80 71 57 80 75 77 60 86 77 56 ...
 $ duration: num  4.02 2.15 4 4 4 ...

> summary (geyser)

    waiting          duration
 Min.   : 43.00   Min.   :0.8333
 1st Qu.: 59.00   1st Qu.:2.0000
 Median : 76.00   Median :4.0000
 Mean   : 72.31   Mean   :3.4608
 3rd Qu.: 83.00   3rd Qu.:4.3833
 Max.   :108.00   Max.   :5.4500

> cov (geyser)

           waiting   duration
waiting  192.94110 -10.278355
duration -10.27836   1.317683

> skewness (geyser)

   waiting   duration
-0.3392250 -0.4530714

> plot (duration, waiting)
```