Bivariate Probability Distributions

Abby Spurdle

September 18, 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.

Introduction

This package contains alternatives to persp() for plotting bivariate 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*.
   (Kernel density and distribution function estimation).

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

The functions for constructing distributions take some parameters and return functions which can be evaluated for x and y, except for kernel distributions.

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

I'm planning to add other probability distributions in the near future.

Loading The Package

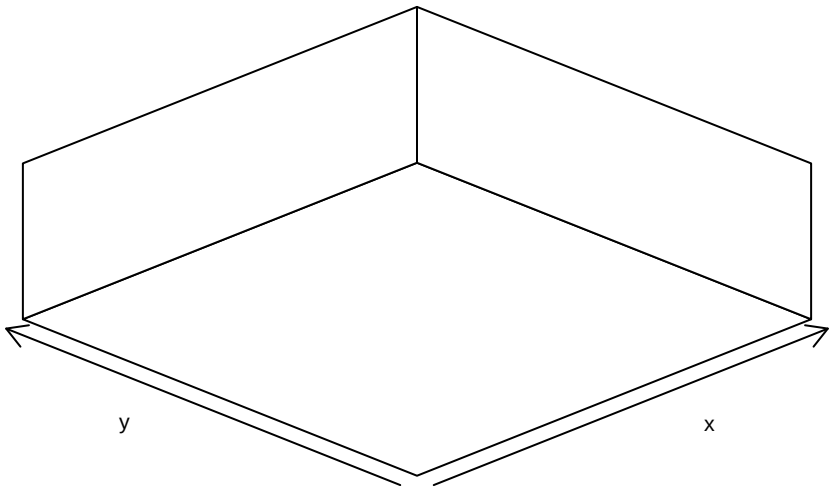
First we need to load the bivariate package.

```
> library (bivariate)
```

Coordinate System

This package uses a nonstandard coordinate system. It's designed to make it simpler to view cumulative distribution functions.

```
> plot3d.empty ()
```



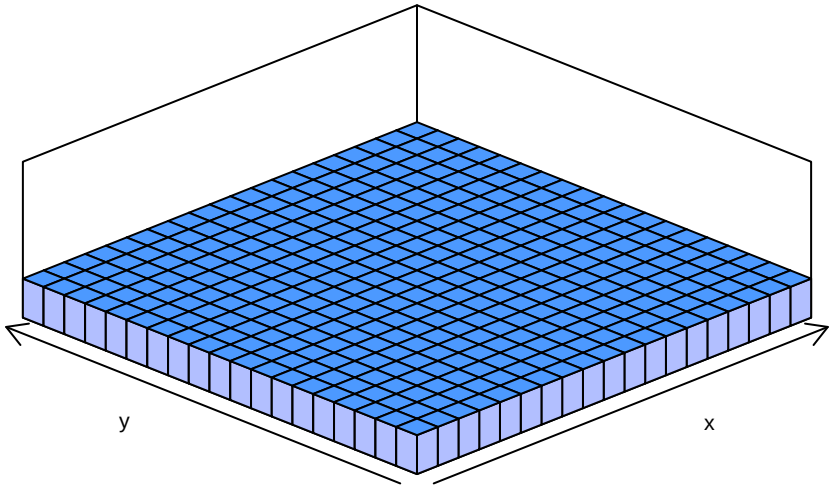
Note the directions of x and y.

Discrete Bivariate Uniform Distributions

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

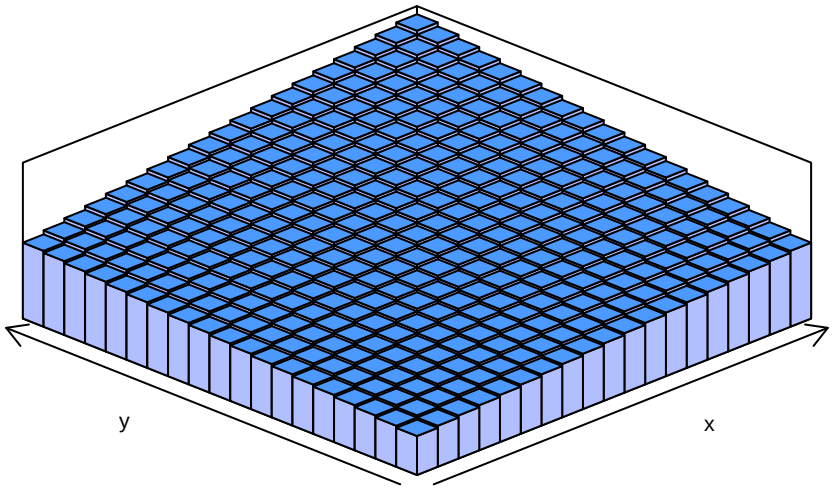
We can use the dubvpmf() function. It takes four arguments. The a and b values of X and the a and b values of Y.

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



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

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



In both cases, we can print the resulting function or evaluate it for x and y.

```
> f

function (x, y)
{
    .dubvpmf.eval(x, y)
}
attr(,"class")
[1] "dubvpmf"
attr(,"n")
[1] 2 2
attr(,"a")
```

```
[1] 0 0
attr(,"b")
[1] 1 1
```

```
> f (0.5, 0.5)
```

```
[1] 0.25
```

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

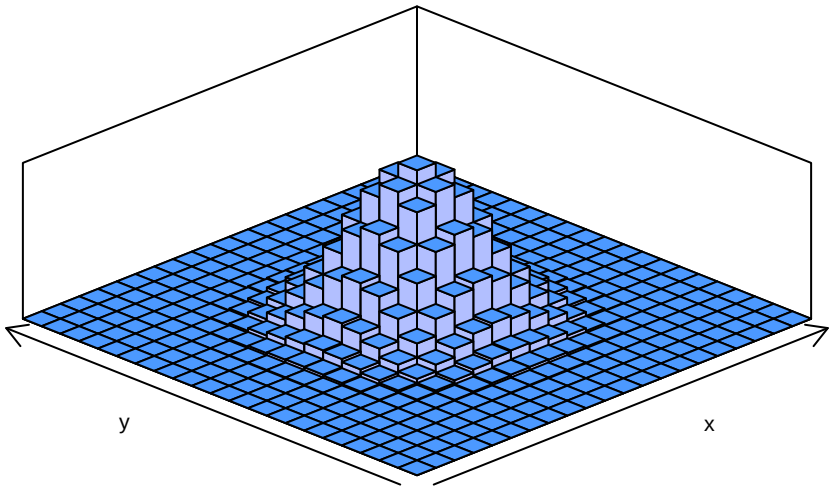
Bivariate Binomial Distributions

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.

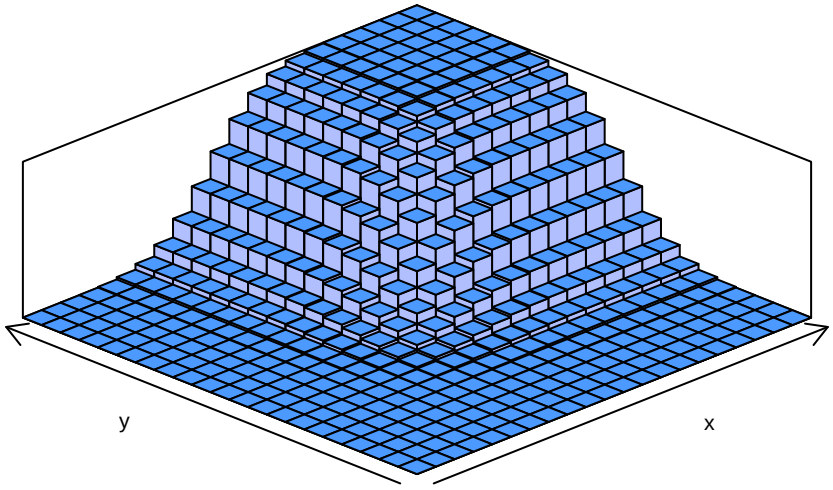
We can use the bnbvpmf() function. It takes three arguments. The number of trials and the probability of each success.

```
> 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*

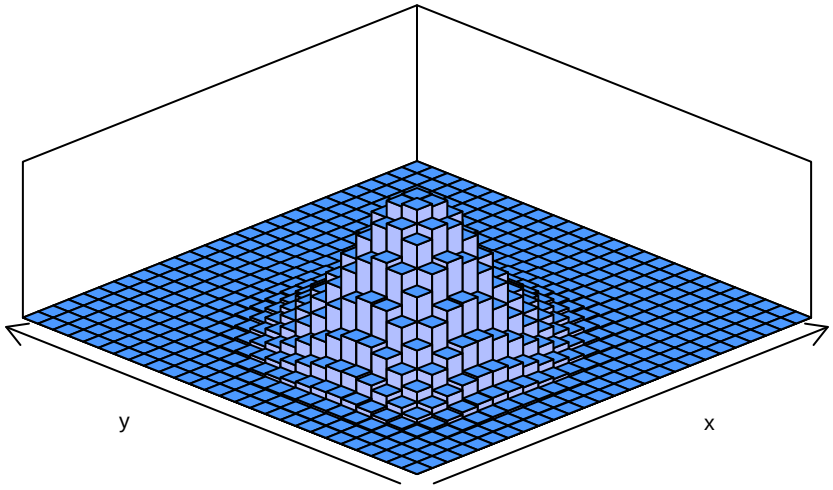
Based on (Karlis and Ntzoufras, 2003) we can define the bivariate Poisson probability mass function as:

$$\mathbb{P}(X = x, Y = y) = \mathrm{f}_{X,Y}(x, y) = e^{-(\lambda_1+\lambda_2+\lambda_3)} \frac{\lambda_1^x}{x!} \frac{\lambda_2^y}{y!} \sum_{k=0}^{min(x,y)} \binom{x}{k} \binom{y}{k} k! \Big(\frac{\lambda_3}{\lambda_1 \lambda_2}\Big)^k$$

Where:
$\mathbb{E}(X) = \mathrm{var}(X) = \lambda_1 + \lambda_3$
$\mathbb{E}(Y) = \mathrm{var}(Y) = \lambda_2 + \lambda_3$
$\mathrm{cov}(X, Y) = \lambda_3$

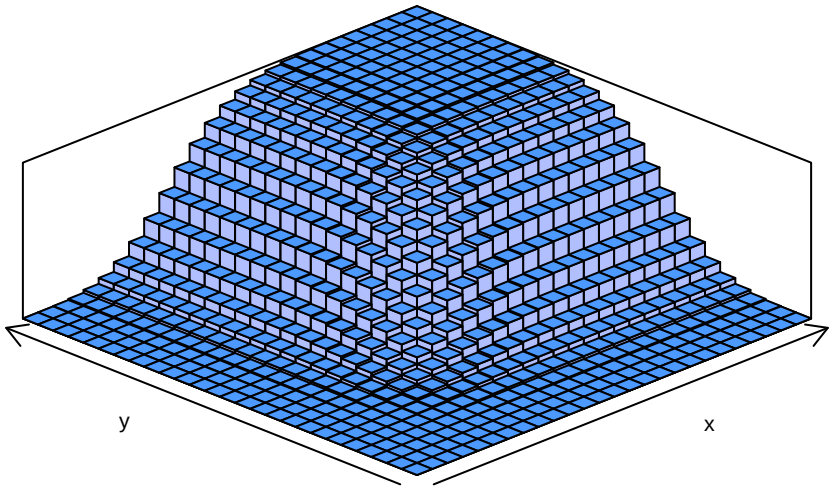
We can use the pbvpmf() function. It takes three arguments. The mean of X, the mean of Y and the covariance between X and Y.

```
> f = pbvpmf (10, 10, 2)
> plot (f)
```



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

```
> F = pbvcdf (10, 10, 2)
> plot (F)
```

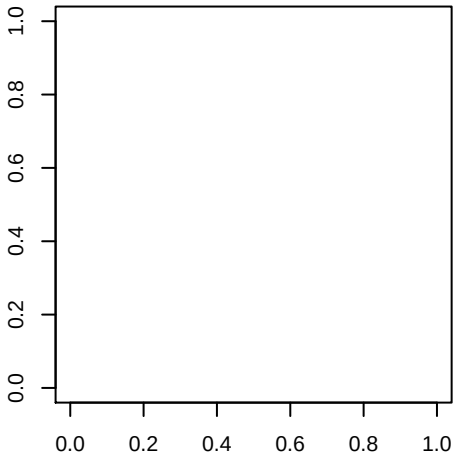


Continuous Bivariate Uniform Distributions

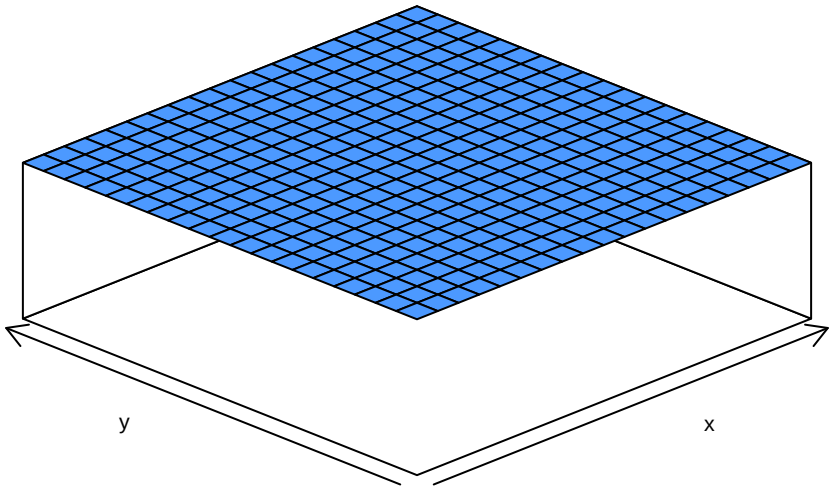
Continuous bivariate uniform distributions are similiar to discrete bivariate uniform distributions. However, we have a probability density function instead of a probability mass function.

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)
```

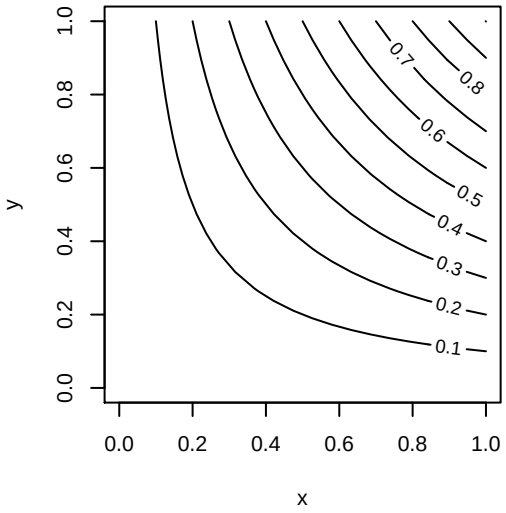



```
> plot (f, TRUE)
```

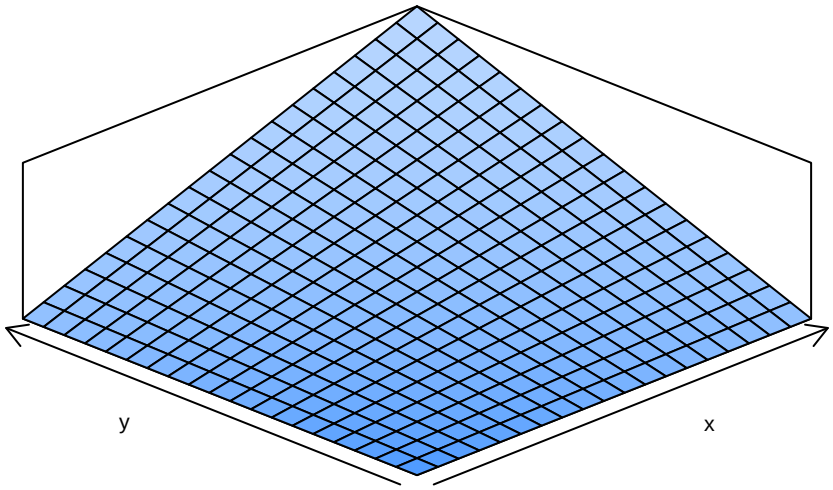


To construct a cumulative distribution function, we can use the cubvcdf() function. It takes the same arguments as cubvpdf().

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




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



Bivariate Normal Distributions*

(Genz, Bretz, Miwa, Mi, Leisch, Scheipl and Hothorn, 2018) provide the mvtnorm package, which is 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.

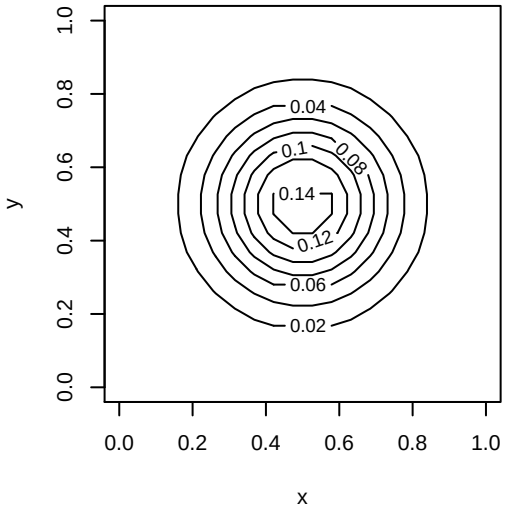
```
> f = nbvpdf (0, 0, 1, 1, 0)
```

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

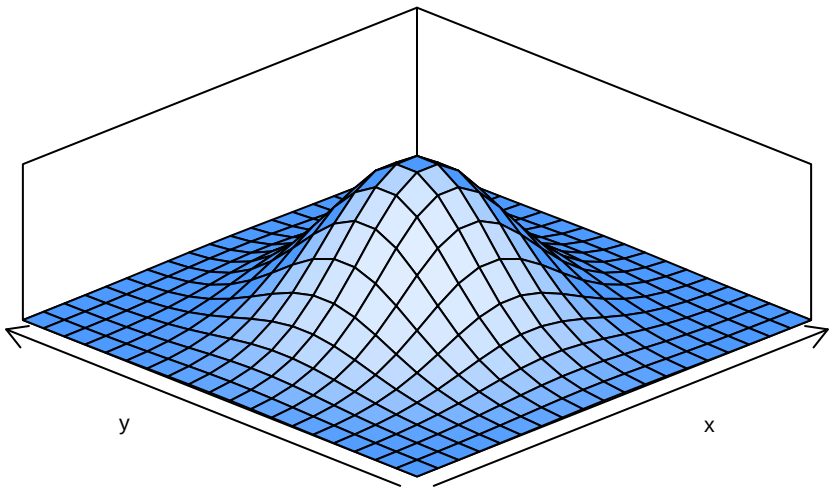
```
> #f = nbvpdf (mean.x, mean,y, sd.x ^ 2, sd.y ^ 2, sd.x * sd.y * cor.xy)
```

Once we have created 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().

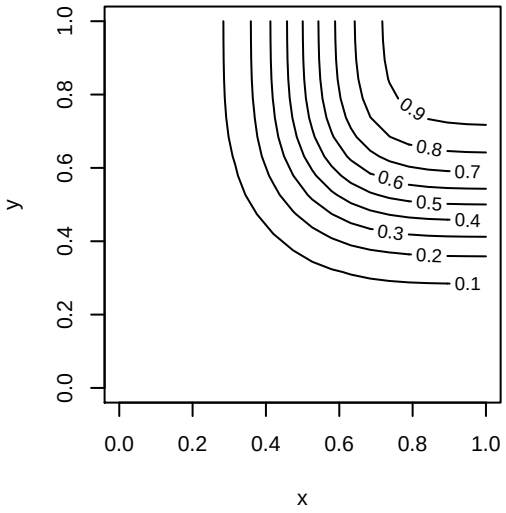
```
> F = nbvcdf (0, 0, 1, 1, 0)
```

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

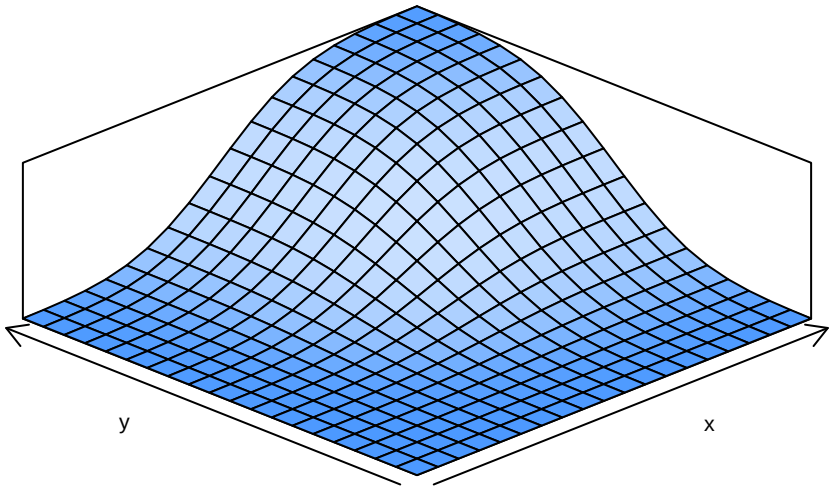
```
> #F = nbvcdf (mean.x, mean,y, sd.x ^ 2, sd.y ^ 2, sd.x * sd.y * cor.xy)
```

Once we have created our object we can plot it.

```
> plot (F)
```




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

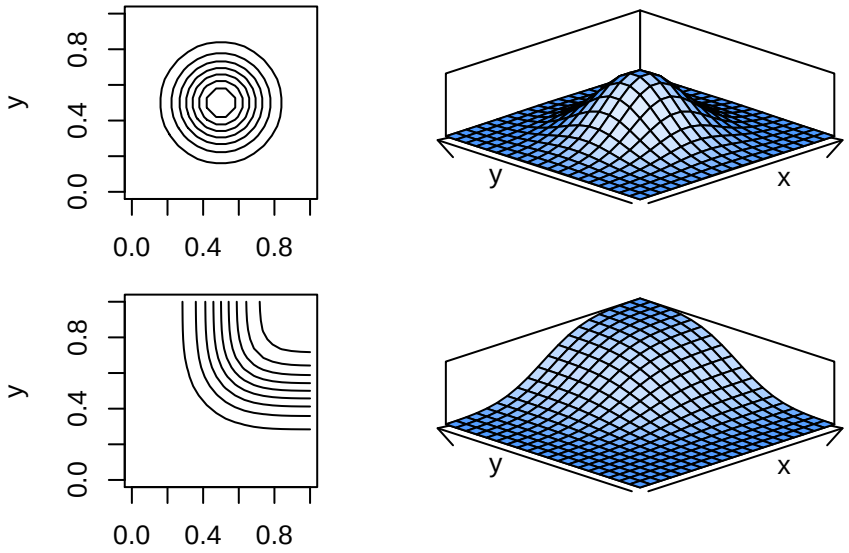


Comparing Normal Distributions

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.

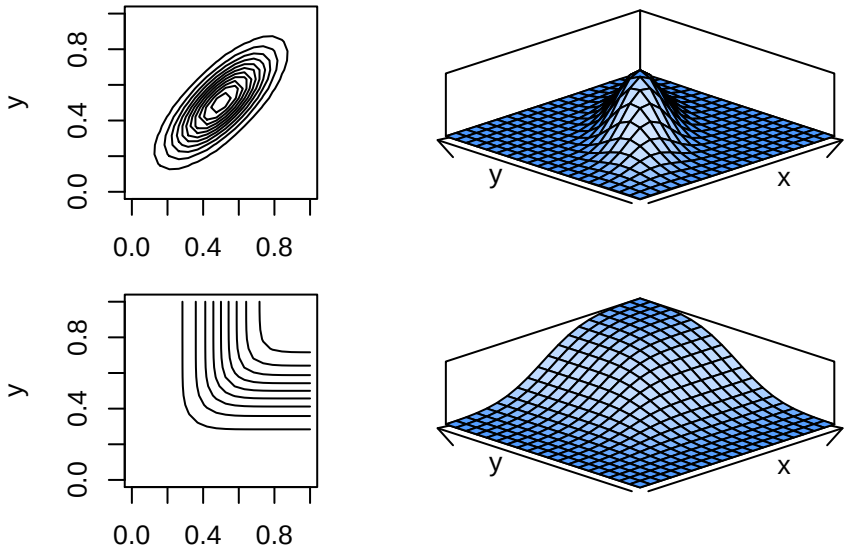
```
> plot (f, all=TRUE)
```



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

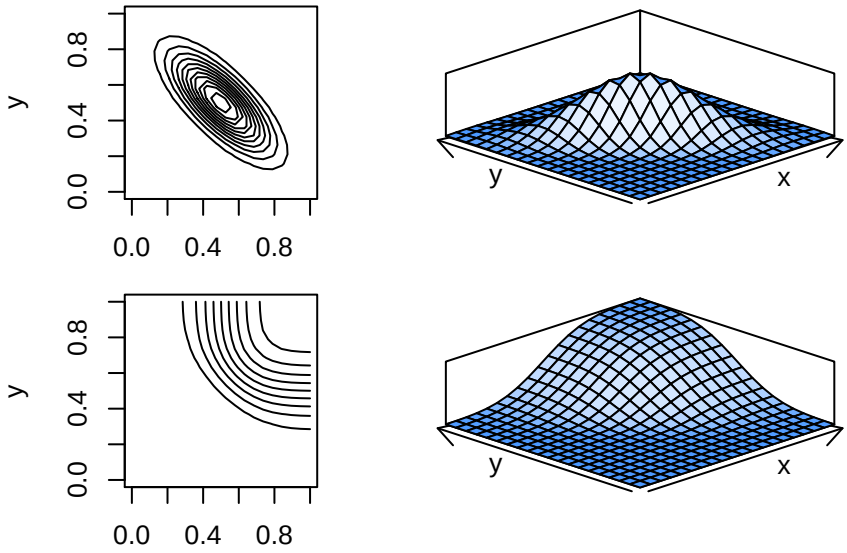
Second, let's consider bivariate distributions with positive correlation/covariance.

```
> 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)
```

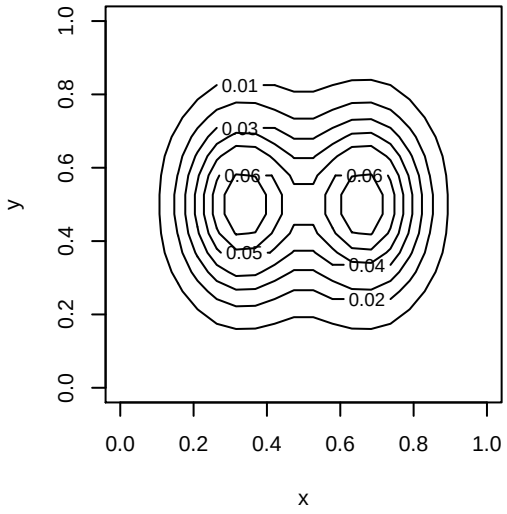


Bivariate Bimodal Distributions*

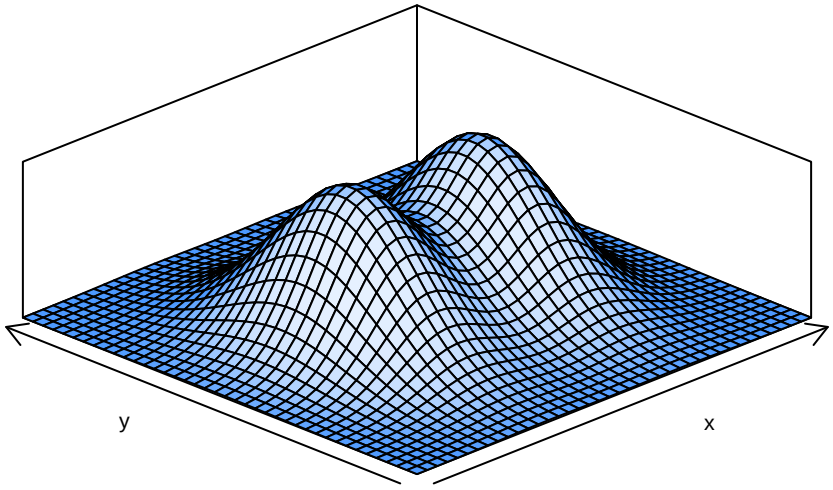
One way to construct a bivariate bimodal probability density function is to construct two bivariate normal probability density functions and then add them together and divide by two.

We can construct such a probability density function using the bmbvpdf() function. It takes eight 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.

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



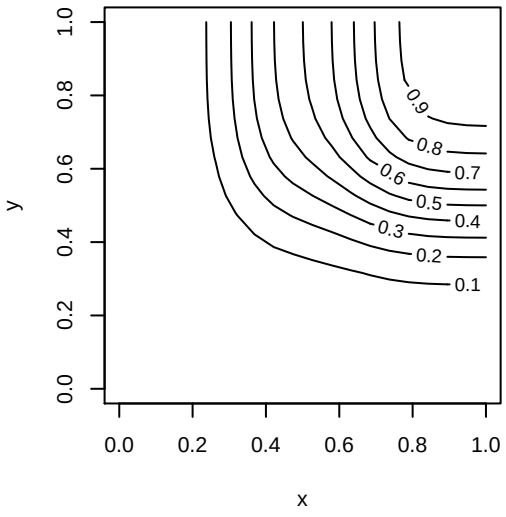

```
> plot (f, TRUE, 40)
```



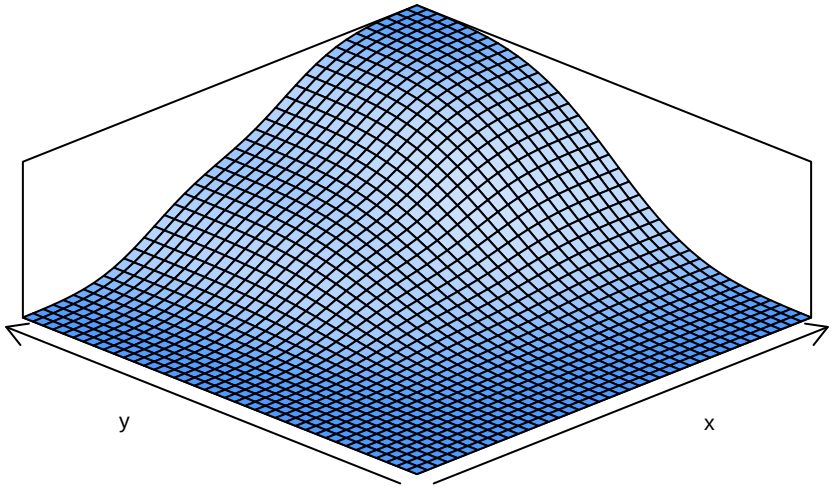
Note that it may be useful to include a third argument to increase the resolution of the plot.

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, 40)
```

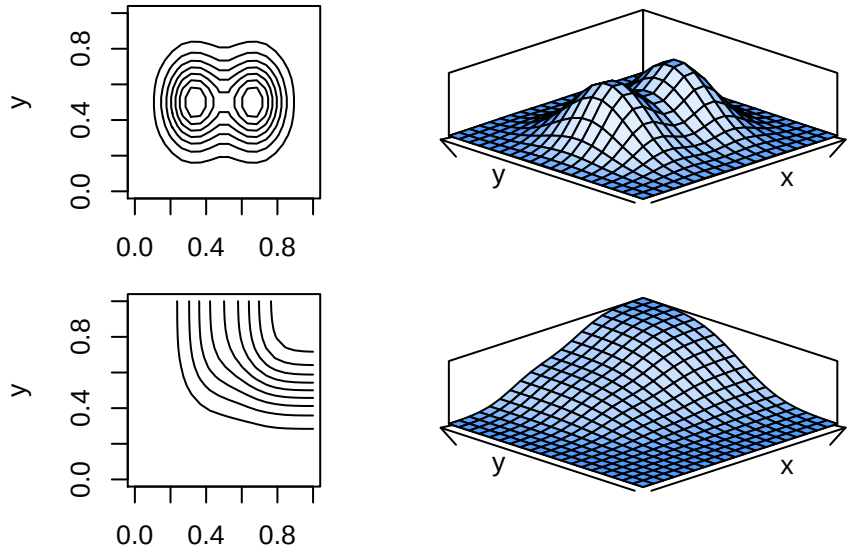


Comparing Bimodal Distributions

We can compare different bimodal distributions using different covariance structures.

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

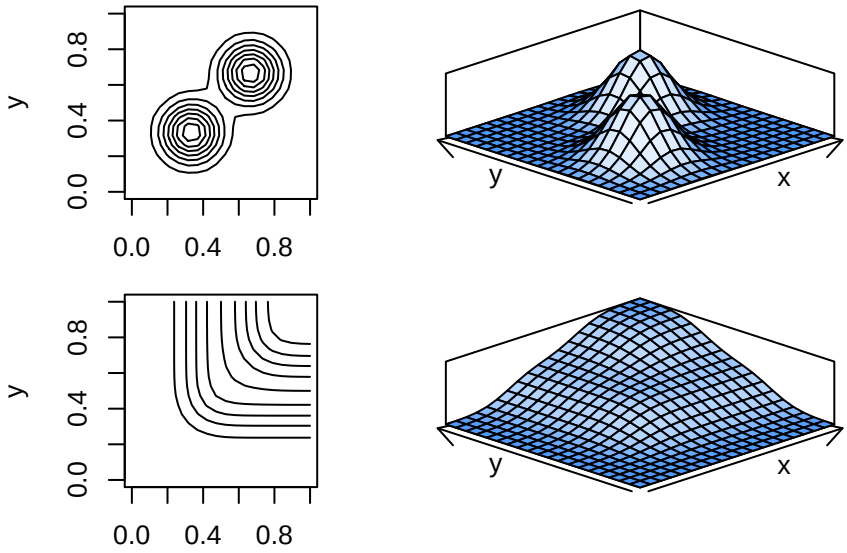
```
> plot (f, all=TRUE)
```



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

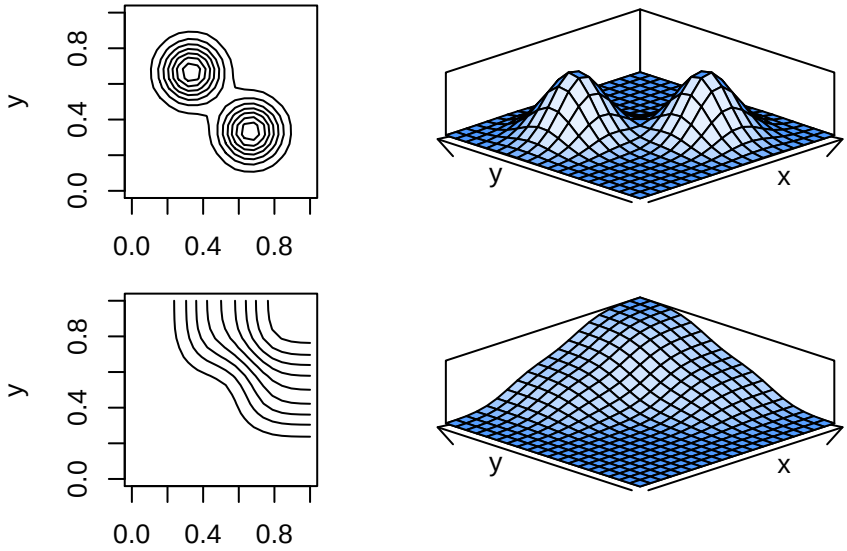
Second, let's consider bimodal distributions with a positive covariance structure.

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

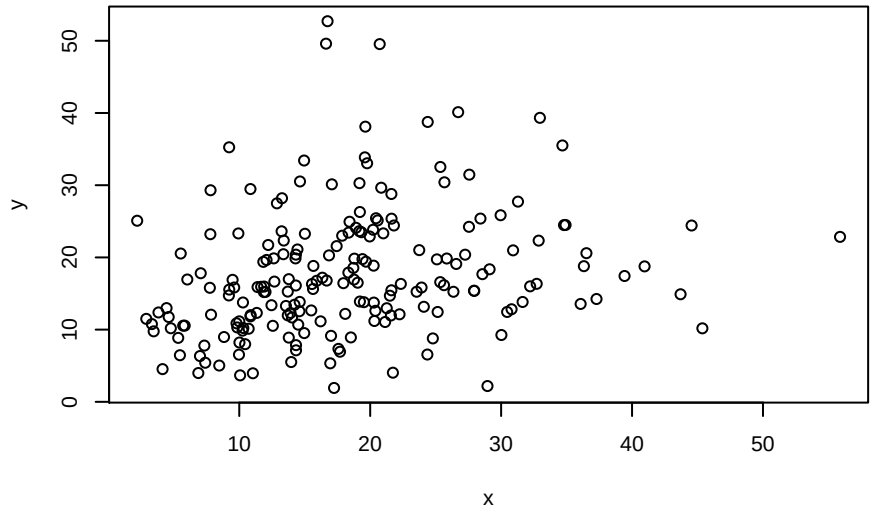


Sample Data

The following section requires data rather than just parameters.

Lets create a normal like positively correlated positively skewed sample.

```
> xy = generate.bivariate.sample (200)
> x = xy [,1]
> y = xy [,2]
> plot (x, y)
```

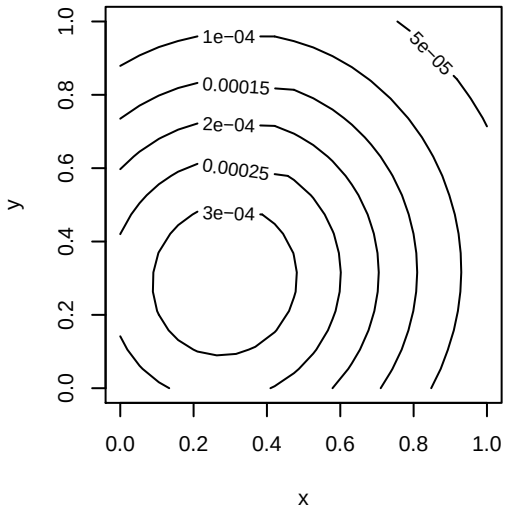


Bivariate Kernel Distributions*

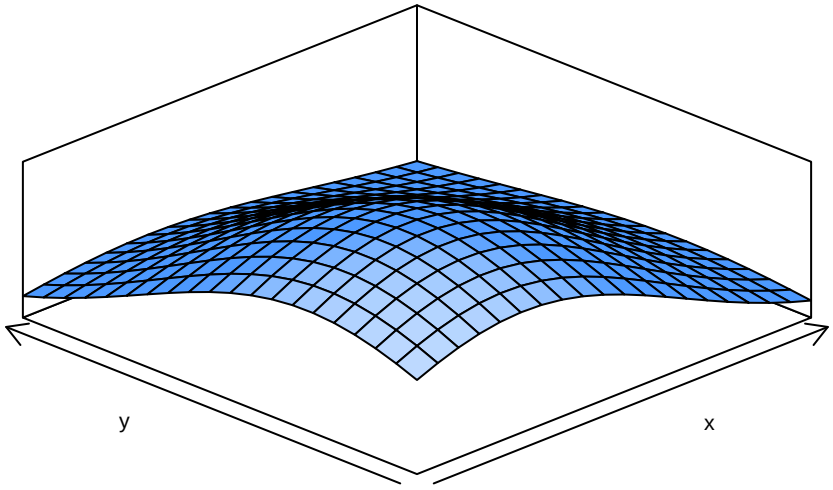
(Wand, 2015) provides the KernSmooth package, which is used by this package.

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

```
> f = kbvpdf (x, y, 20, 20)
> plot (f)
```

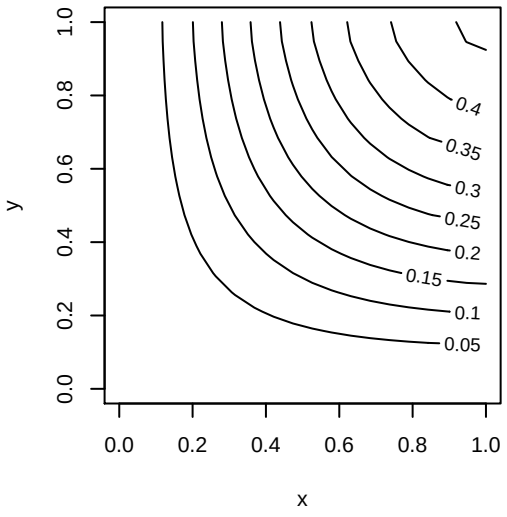



```
> plot (f, TRUE)
```

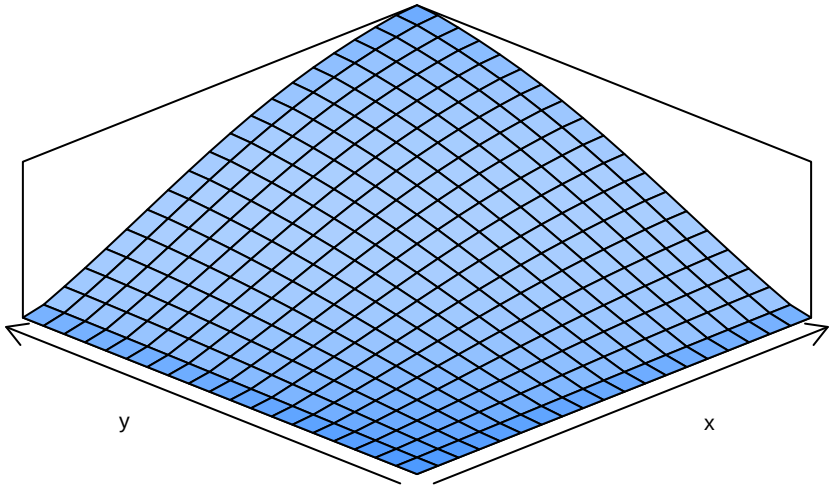


We can use the kbvcdf() function to produce bivariate kernel distribution function estimation.

```
> F = kbvcdf (x, y, 20, 20)
> plot (F)
```




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



References

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

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

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