bivariate 0.4.1

Bivariate Probability Distributions

Abby Spurdle

May 24, 2019

Contains convenience functions for constructing, plotting and evaluating bivariate probability distributions, including their probability mass functions, probability density functions and cumulative distribution functions. Supports uniform (discrete and continuous), binomial, Poisson, categorical, normal, bimodal and Dirichlet (trivariate) distributions, and kernel smoothing and empirical cumulative distribution functions.

Pre-Intro

This package is based on function objects.

Some functions (constructors) return other functions (probability distributions), which can be evaluated.

This is equivalent but different to (a subset of) the d, p, q, r approach used in R's stats package.

Introduction

This package contains convenience functions for constructing, plotting and evaluating bivariate probability distributions, including their probability mass functions, probability density functions and cumulative distribution functions.

It supports the following parametric probability distributions:

1. **Discrete** bivariate **Uniform** distributions.
2. Bivariate **Binomial** distributions.
3. Bivariate **Poisson** distributions*.
4. Bivariate **Categorical** distributions*.
5. **Continuous** bivariate **Uniform** distributions.
6. Bivariate **Normal** distributions*.
7. Bivariate **Bimodal** distributions*.
8. Trivariate **Dirichlet** distributions*.

And it supports the following nonparametric probability distributions:

1. Bivariate **Kernel** density estimates*.
2. Bivariate **Empirical** cumulative distribution functions*.

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

In general, we can compute their probability mass function (PMF) or their probability density function (PDF), and their cumulative distribution function (CDF).

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

Note that discrete probability distributions convert their arguments to integers before evaluation.

The functions for plotting discrete distributions take a function object and can plot either a heat map or a 3d bar plot. Most of the functions for plotting continuous distributions take a function object and can plot either a contour plot or a 3d surface plot.

Refer to my barsurf package for information on how to customize plots, if required.

Note that my probhat package supports kernel smoothing, more generally.

Loading The Packages

I'm going to load (and attach) the intoo, bivariate and MASS packages.

```
> library (intoo)
> library (bivariate)
> library (MASS)
```

Note that the bivariate package imports the intoo, barsurf, mvtnorm and KernSmooth packages.

Discrete Bivariate Uniform Distributions

We can describe a bivariate uniform distribution as the product of two univariate uniform distributions.

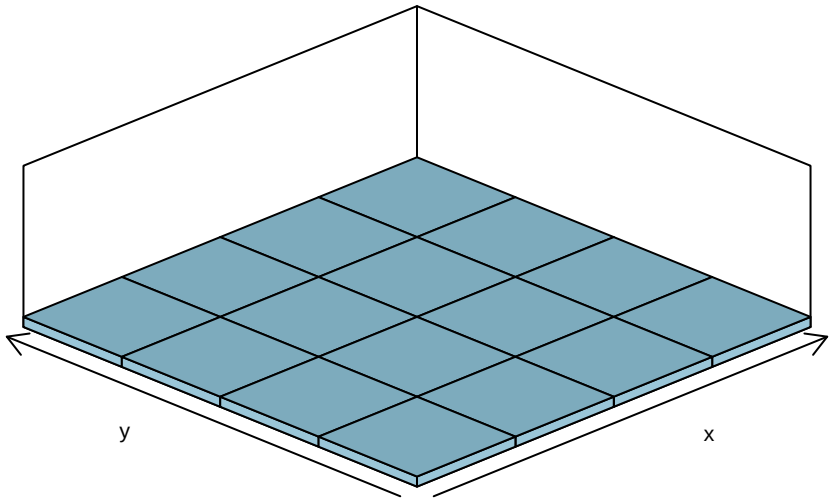
We can construct its probability mass function using the dubvpmf() function. It takes four arguments, the a and b values of X and the a and b values of Y.

```
> f = dubvpmf (1, 4, 1, 4)
```

We can print the function, however, I will give an example later.

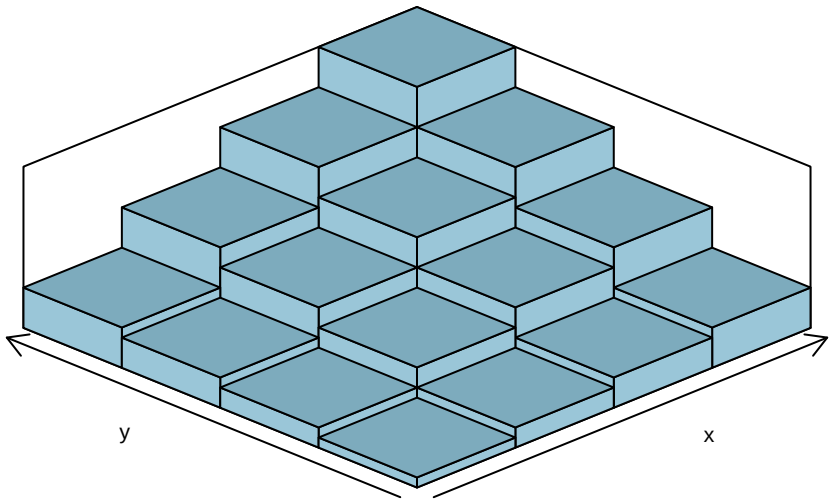
Also, we can plot it.

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



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

```
> F = dubvcdf (1, 4, 1, 4)
> plot (F, TRUE)
```



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

```
> f (1, 1)

[1] 0.0625
```

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

Bivariate Binomial Distributions

One way to describe 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.

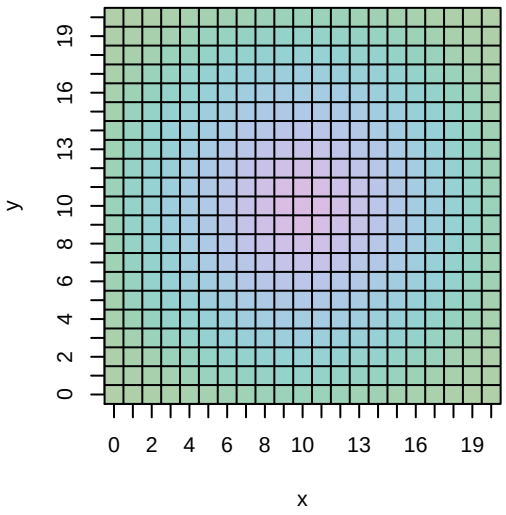
Like the bivariate uniform distribution, we can describe a bivariate binomial distribution as the product of two univariate binomial distributions with the same n parameter, so:

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

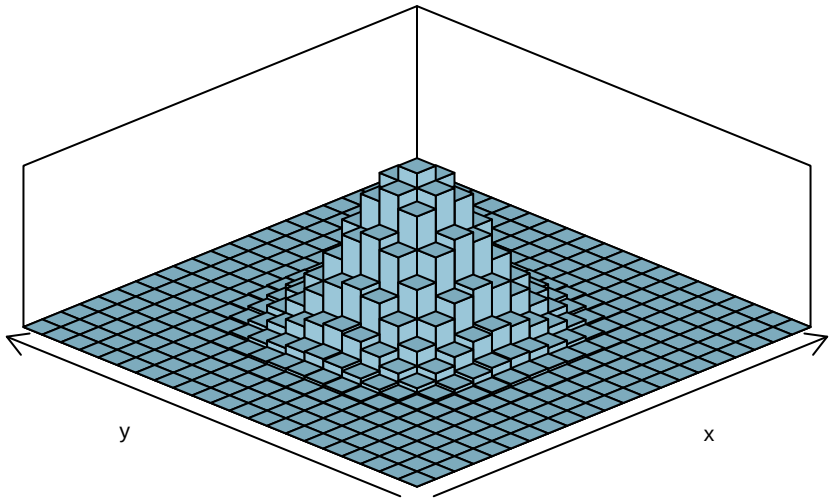
Where p_X is the probability of the first success and p_Y is the probability of the second success.

We can construct its probability mass function using the bnbvpmf() function. It takes three arguments, the probability of the first success, the probability of the second success and the number of trials.

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

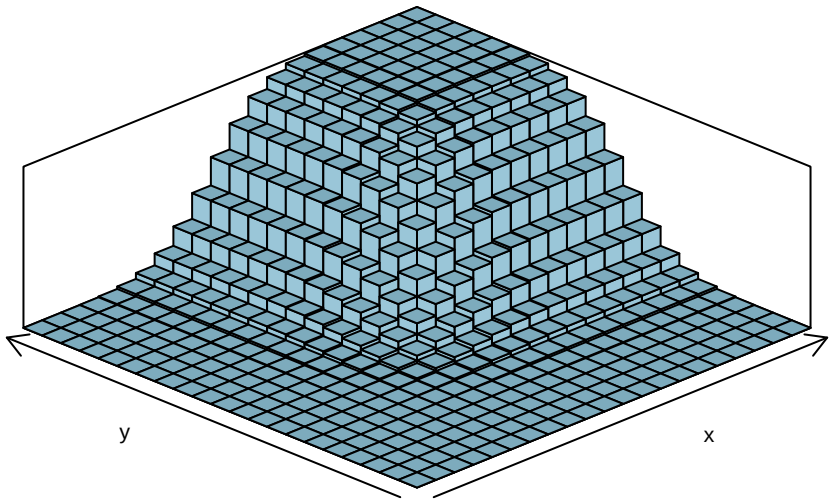



```
> plot (f, TRUE)
```



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

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



Note that I've put the probabilities first, however, it's customary for the number of trials to be first.

If n is omitted, it defaults to one, giving a Bernoulli distribution.

Bivariate Poisson Distributions*

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

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

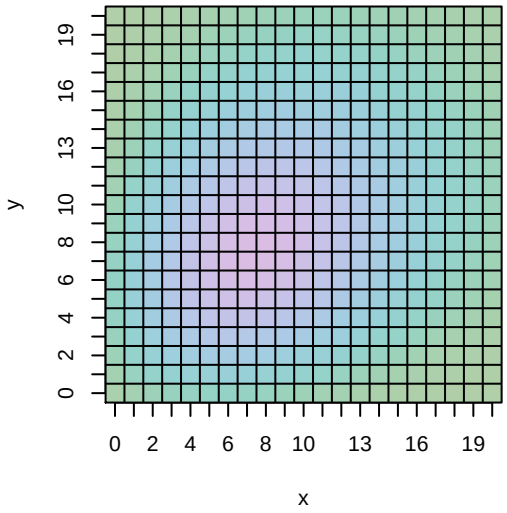
Where k is in 0 to $\min(x, y)$.

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

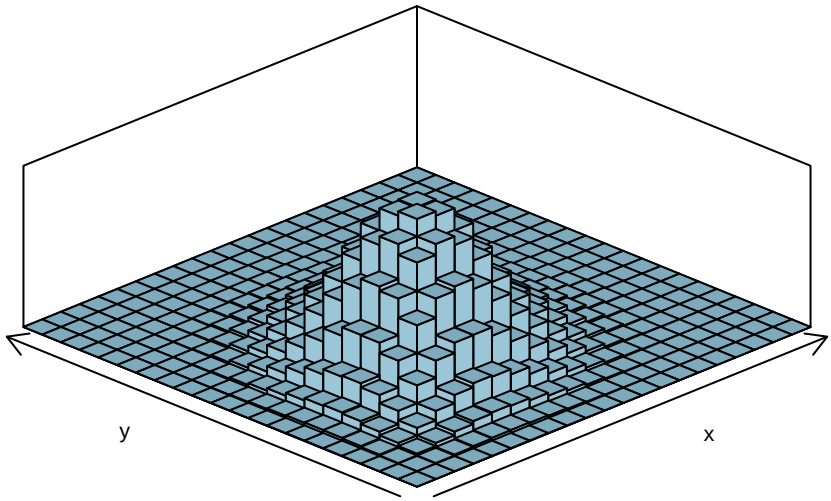
We can use the pbvpmf() or pbvpmf.2() functions.

The first takes three arguments, the three λ parameters. The second also takes three arguments, but the mean of X, the mean of Y and the covariance between X and Y.

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

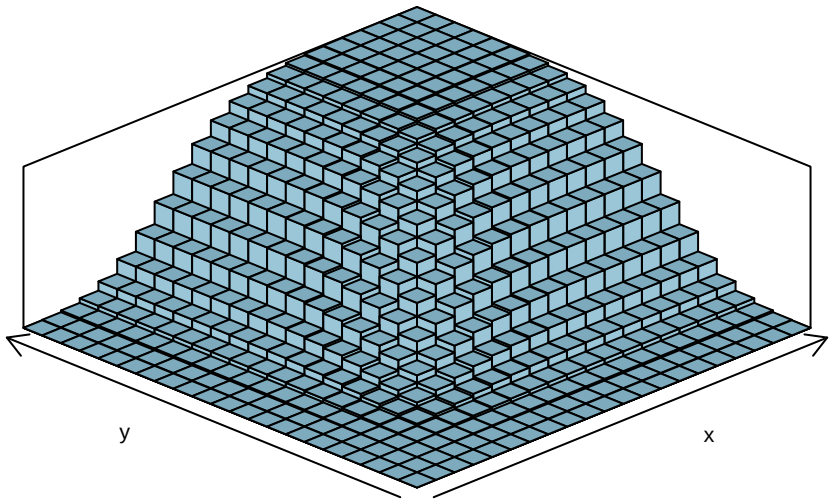



```
> plot (f, TRUE)
```



To construct a cumulative distribution function, we can use the pbvcdf() or pbvcdf.2() functions. They take the same arguments as pbvpmf() and pbvpmf.2().

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



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 $\text{cov}(X, Y)$.

Bivariate Categorical Distributions*

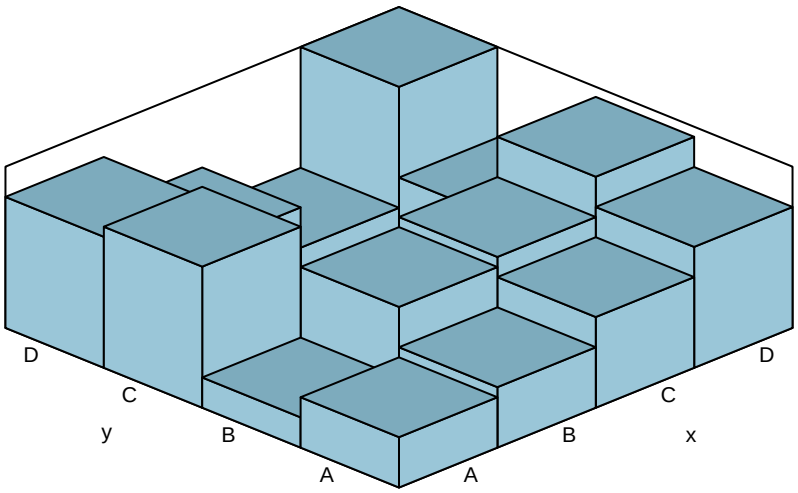
Bivariate categorical distributions are different to other probability distributions in this package. They are defined by a matrix of parameters (with either probabilities or frequencies), preferably with row and column names.

```
> z = matrix (sample (1:16), 4, 4) / 136
> rownames (z) = colnames (z) = c ("A", "B", "C", "D")
```

```
> z
```

```
           A          B           C          D
A 0.03676471 0.02205882 0.102941176 0.09558824
B 0.04411765 0.07352941 0.007352941 0.05882353
C 0.06617647 0.08088235 0.014705882 0.02941176
D 0.08823529 0.11029412 0.051470588 0.11764706
```

```
> f = cbvpmf (z)
> plot (f, TRUE)
```



Evaluation can use either integers or strings, and returns probabilities.

```
> f (1, 2)
```

```
[1] 0.02205882
```

```
> f ("A", "B")
```

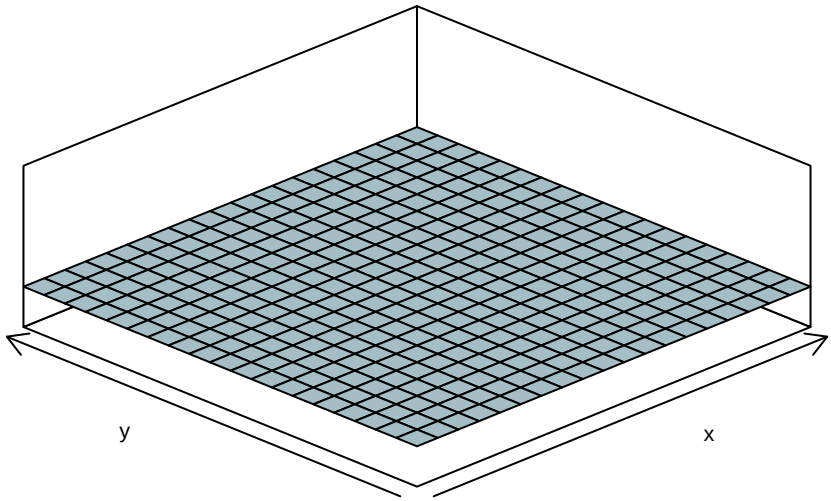
```
[1] 0.02205882
```

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.

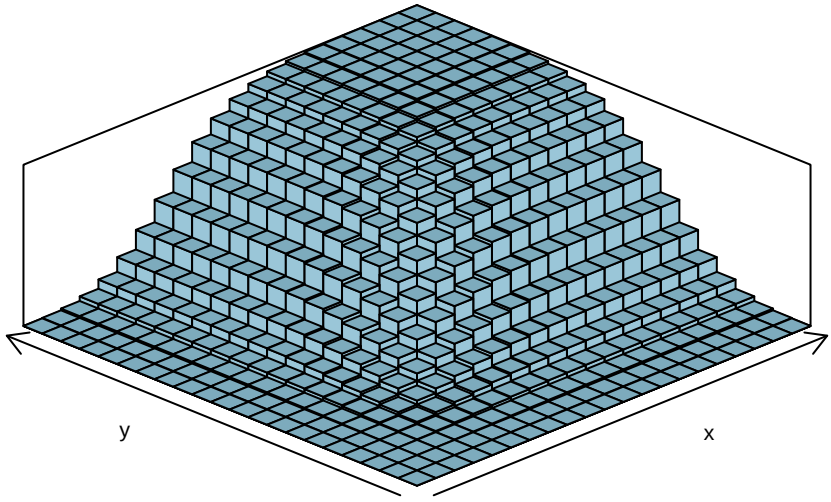
We can use the cubvpdf() function. It takes four arguments, the a and b values of X and the a and b values of Y.

```
> f = cubvpdf (0, 2, 0, 2)
> plot (f, TRUE)
```



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

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



Bivariate Normal Distributions*

We can construct a normal bivariate probability density function using the nbvpdf() or nbvpdf.2() functions, both of which use the dmvnorm() function from the mvtnorm package, and take five arguments.

The first takes the mean of X, the mean of Y, the standard deviation of X, the standard deviation of Y and their correlation. The second takes the mean of X, the mean of Y, the variance of X, the variance of Y and their covariance.

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

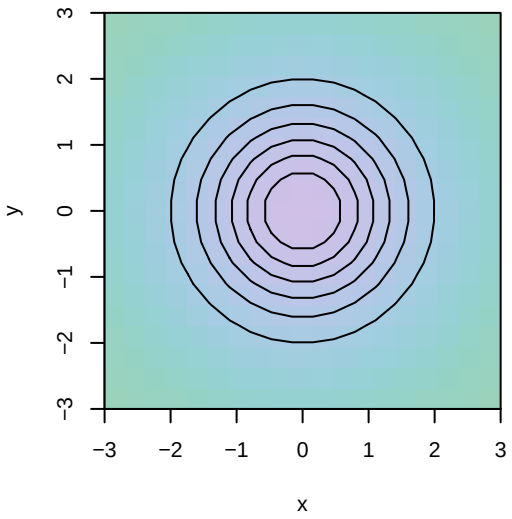
We can print the function.

```
> f
```

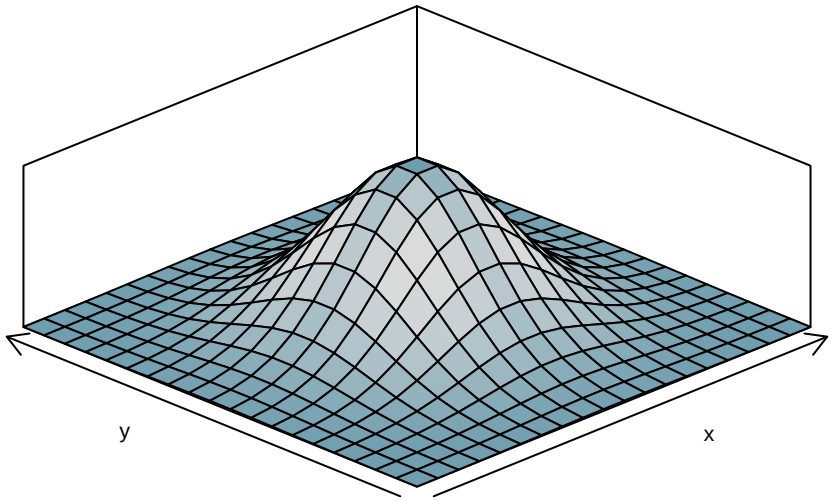
```
nbvpdf, 5
function (x, y)
{
    . = THAT()
    v = .val.numeric.args(x, y)
    .nbvpdf.eval(., v$x, v$y)
}
%$% vector.means, numeric, 2
[1] 0 0
%$% matrix.variances, matrix, 2 * 2
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

And plot it.

```
> plot (f)
```




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

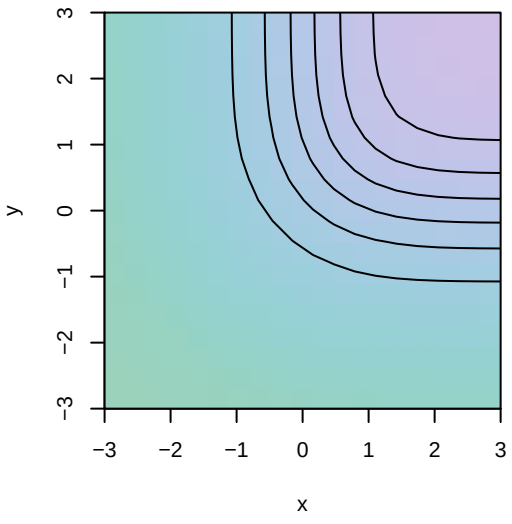


We can construct a cumulative distribution function using the nbvcdf() or nbvcdf.2() functions, both of which use the pmvnorm() function from the mvtnorm package, and take the same arguments as nbvpdf() and nbvpdf.2().

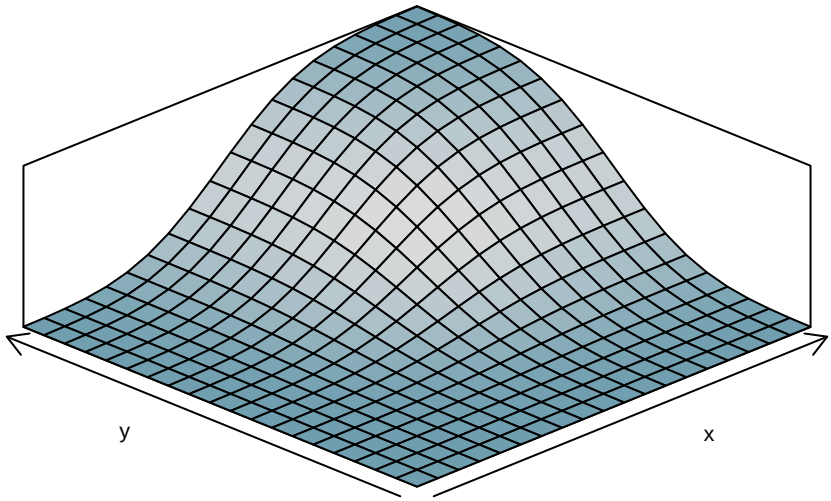
```
> F = nbvcdf (0, 0, 1, 1, 0)
```

Once we have constructed our object we can plot it.

```
> plot (F)
```




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



Note that there's an appendix later that compares normal distributions with different parameters.

Bivariate Bimodal Distributions*

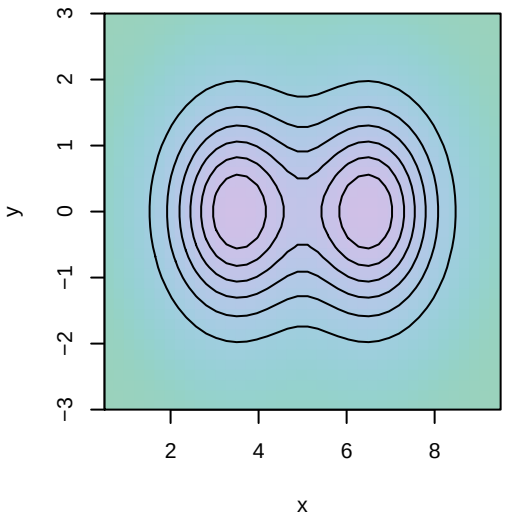
Here, a bimodal distribution isn't an officially recognized distribution, however, it's still of interest.

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

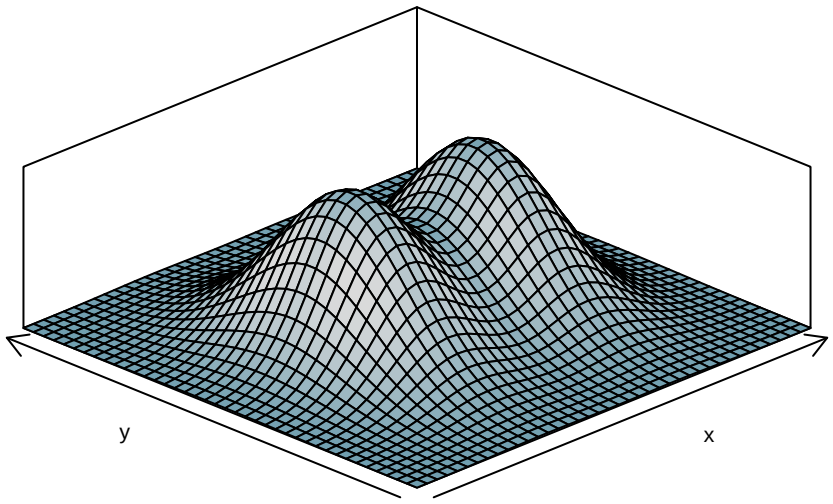
We can construct such a probability density function using the bmbvpdf() or bmbvpdf.2() functions, both of which take eight arguments.

The first function takes the mean of X, the mean of Y, the standard deviation of X and the standard deviation of Y for the first component distribution, and the mean of X, the mean of Y, the standard deviation of X and the standard deviation of Y for the second component distribution. The second function is the same except we use variances instead of standard deviations.

```
> f = bmbvpdf (3.5, 0, 1, 1, 6.5, 0, 1, 1)
> plot (f, npoints=40)
```

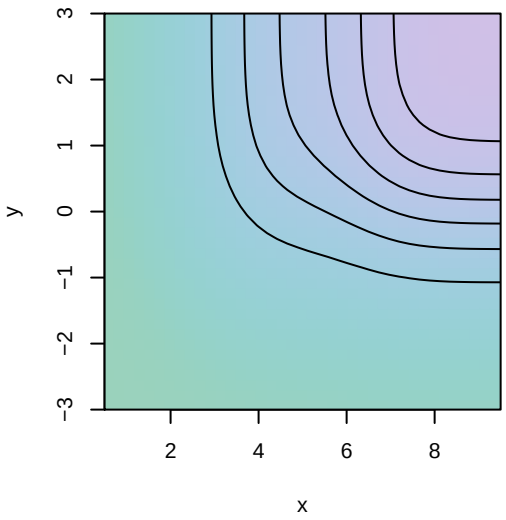



```
> plot (f, TRUE, npoints=40)
```

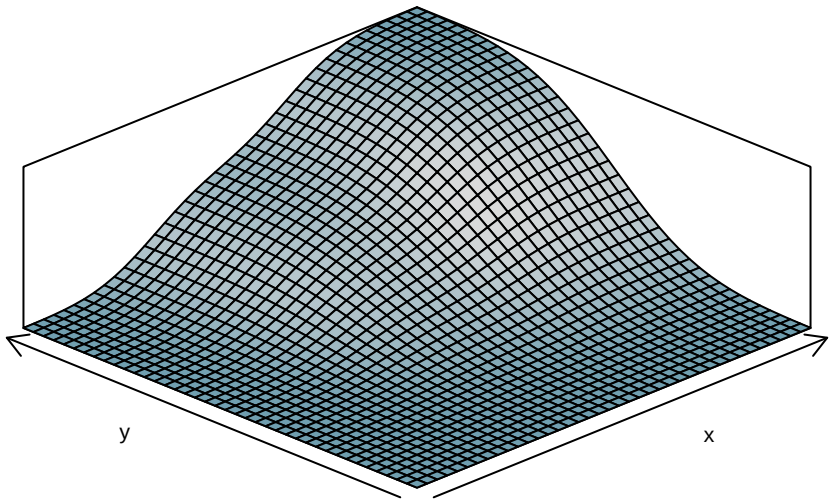


We can construct a cumulative distribution function using the bmbvcdf() or bmbvcdf.2() functions. They take the same arguments as bmbvpdf() and bmbvpdf.2().

```
> F = bmbvcdf (3.5, 0, 1, 1, 6.5, 0, 1, 1)
> plot (F, npoints=40)
```




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



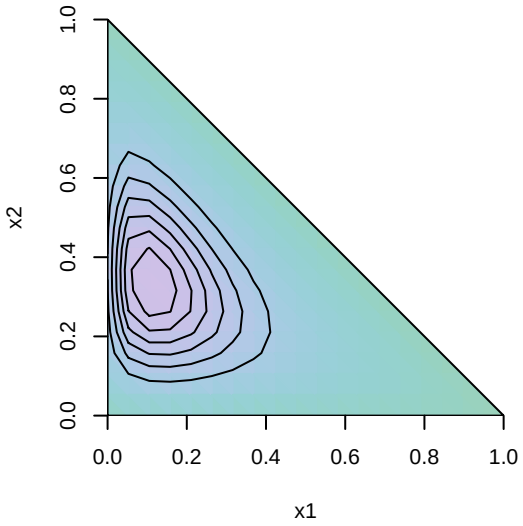
Note that this method is similar kernel smoothing, discussed later.

Trivariate Dirichlet Distributions*

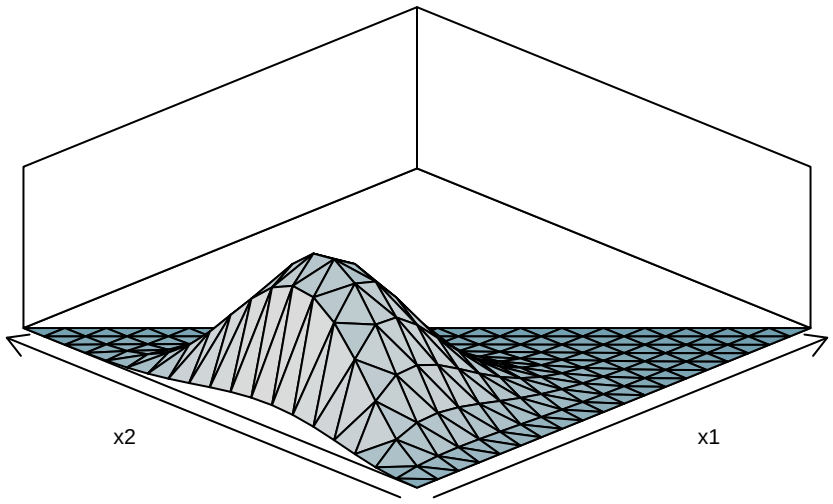
Dirichlet distributions with three variables are similar to other probability distributions with two variables.

We can construct one, using the dtvpdf() function, which takes three α parameters.

```
> f = dtvpdf (2, 4, 6)
> plot (f)
```

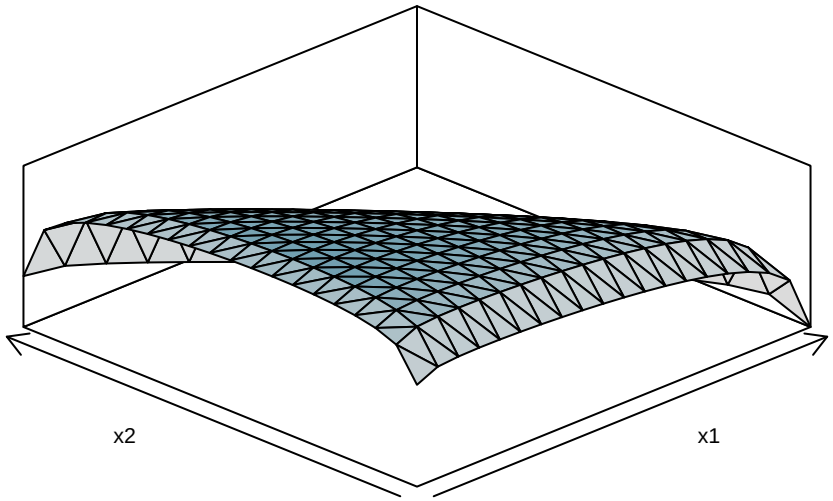



```
> plot (f, TRUE)
```



Or using the log density.

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



Note that Dirichlet distributes can take a large variety of shapes.

I've provided some more examples in an appendix.

Bivariate Kernel Density Estimates*

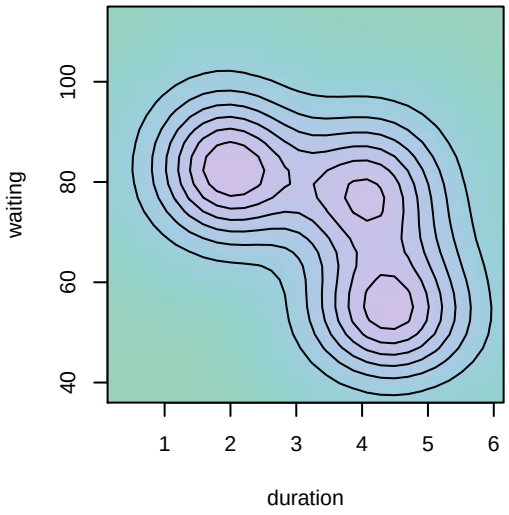
We can construct bivariate kernel density estimates using the kbvpdf() function, which uses the bkde2D() function from KernSmooth. It has four arguments. The first two arguments are x and y vectors of data. The second two arguments are the bandwidths.

I've adapted the example from KernSmooth.

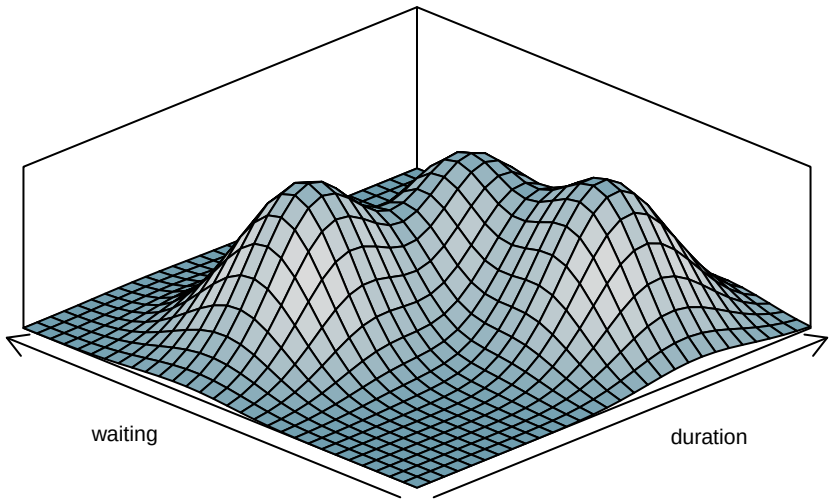
```
> data ("geyser")
> attach (geyser)

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

Again, once we have constructed our object we can plot it.

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




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



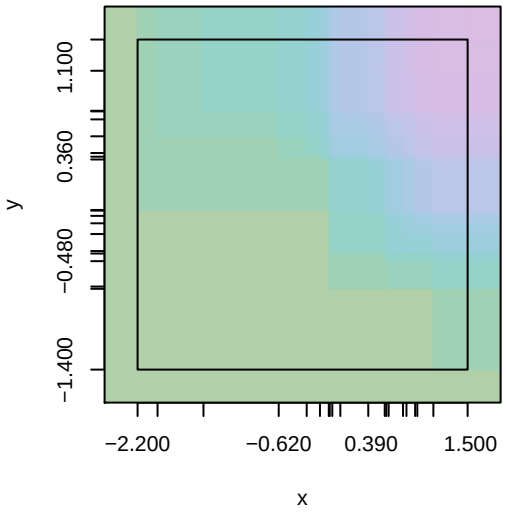

```
> detach (geyser)
```

Bivariate Empirical Cumulative Distribution Functions*

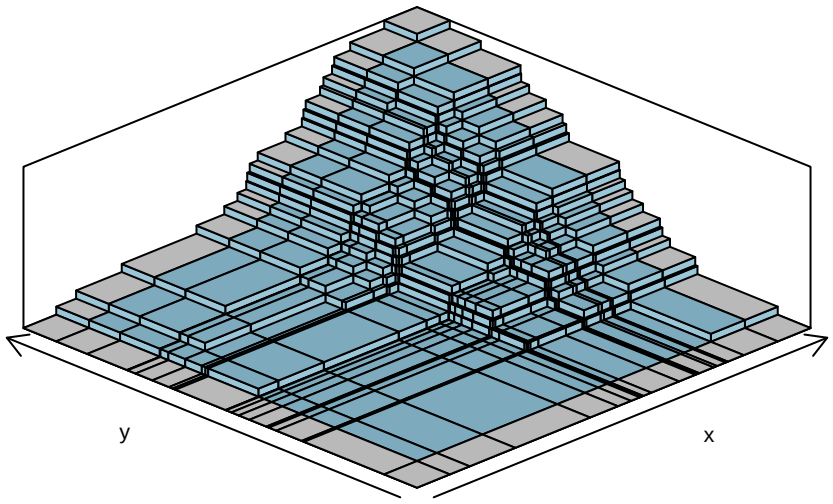
Bivariate ECDFs are constructed in a similar way to bivariate kernel density estimates, except they give us a CDF rather than a PDF.

```
> x = rnorm (20)
> y = rnorm (20)
```

```
> F = ebvcdf (x, y)
> plot (F)
```




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



Note that if the number of observations is small, the function is plotted as a step function, evaluated over the observations, with an extrapolated region. However, if the number of observations is large, the function is plotted as a (continuous) surface, evaluated over a regularly spaced grid.

References

R Packages

Spurdle, A. (2019). intoo: Object Oriented Extensions.

Spurdle, A. (2019). barsurf: Bar, Surface and Related Plots.

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. (2019). probhat: Generalized Kernel Smoothing.

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

Journal Articles

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

Appendix A:
Comparing Normal Distributions

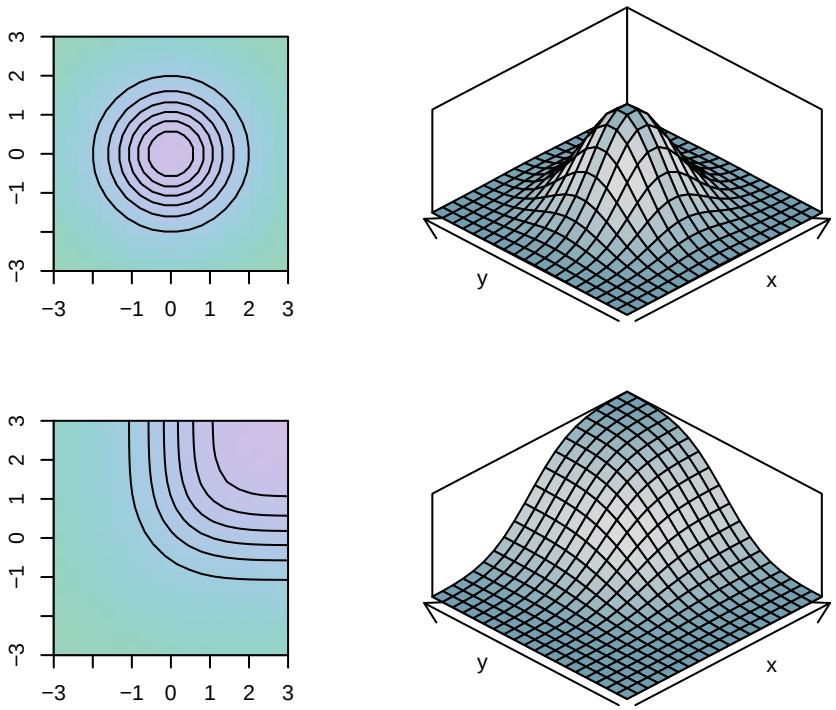
We can compare different normal distributions using different correlation or covariance parameters.

First, let's consider the bivariate distributions from the earlier section with zero correlation.

```
> f1 = nbvpdf (0, 0, 1, 1, 0)
> f1 %$% matrix.variances
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

```
> plot (f1, all=TRUE)
```

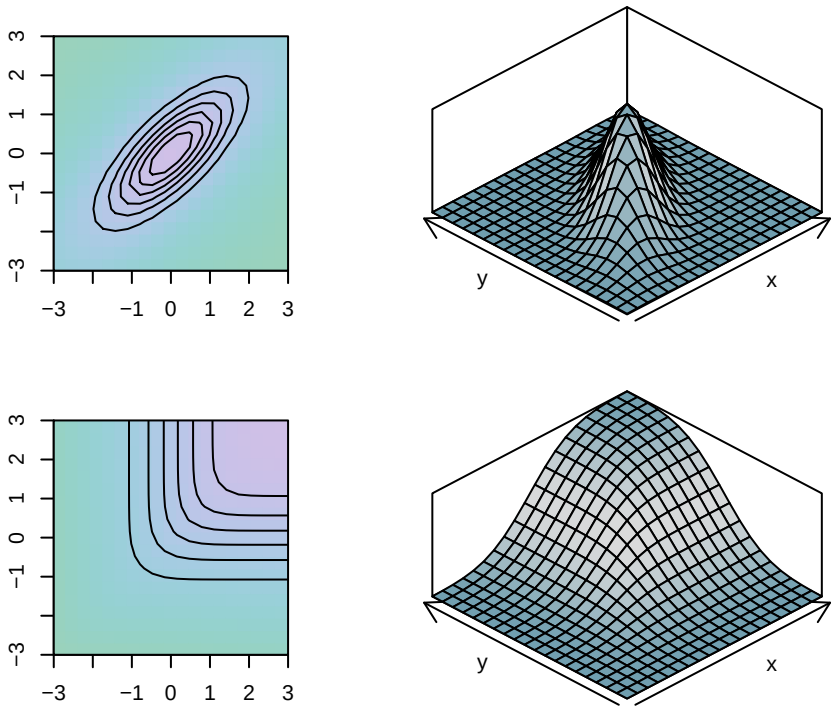


Second, let's consider bivariate distributions with positive correlation.

```
> f2 = nbvpdf (0, 0, 1, 1, 0.75)
> f2 %$% matrix.variances
```

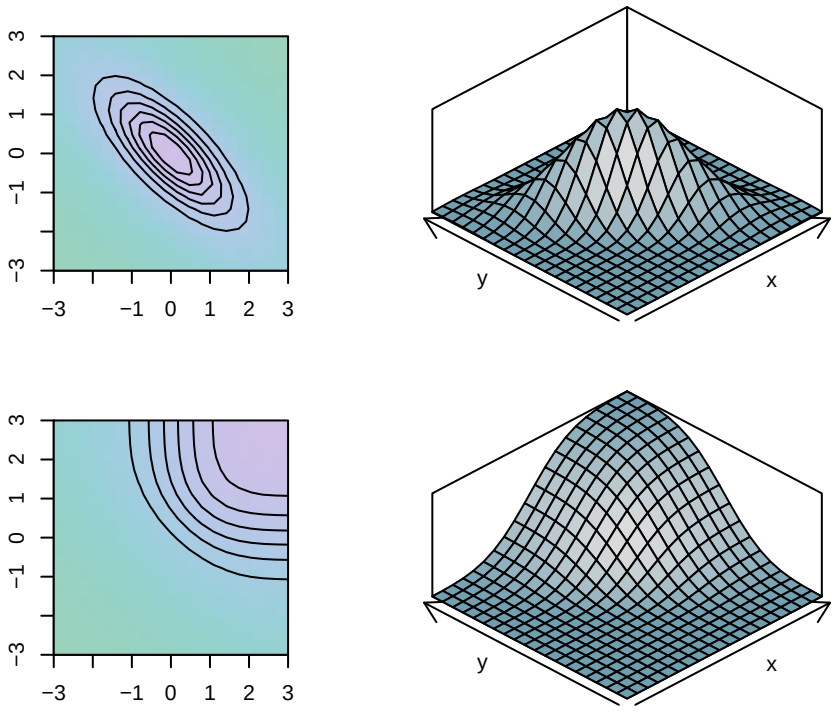
```
     [,1] [,2]
[1,] 1.00 0.75
[2,] 0.75 1.00
```

```
> plot (f2, all=TRUE)
```



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

```
> f3 = nbvpdf (0, 0, 1, 1, -0.75)
> f3 %$% matrix.variances

      [,1]  [,2]
[1,]  1.00 -0.75
[2,] -0.75  1.00

> plot (f3, all=TRUE)
```

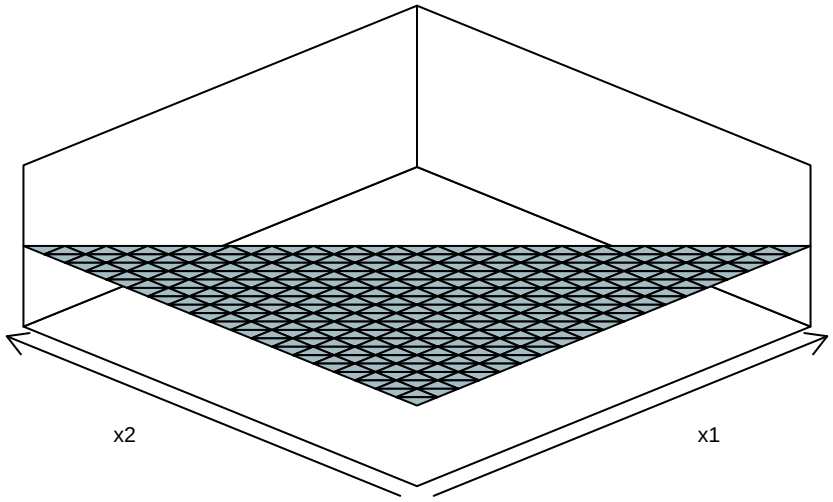


Note that currently, the all=TRUE option requires the PMF or PDF rather than the CDF.

Appendix B:
Comparing Dirichlet Distributions

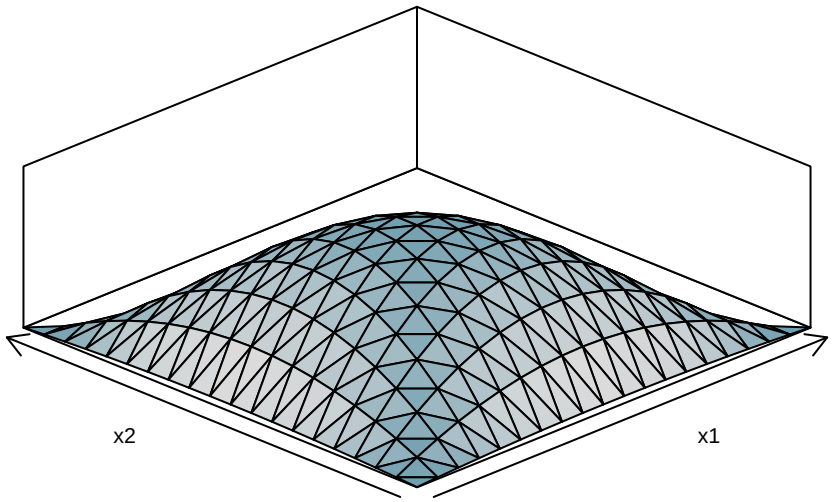
$\boldsymbol{\alpha} = (1, 1, 1)$

```
> plot (dtvpdf (1, 1, 1), TRUE)
```



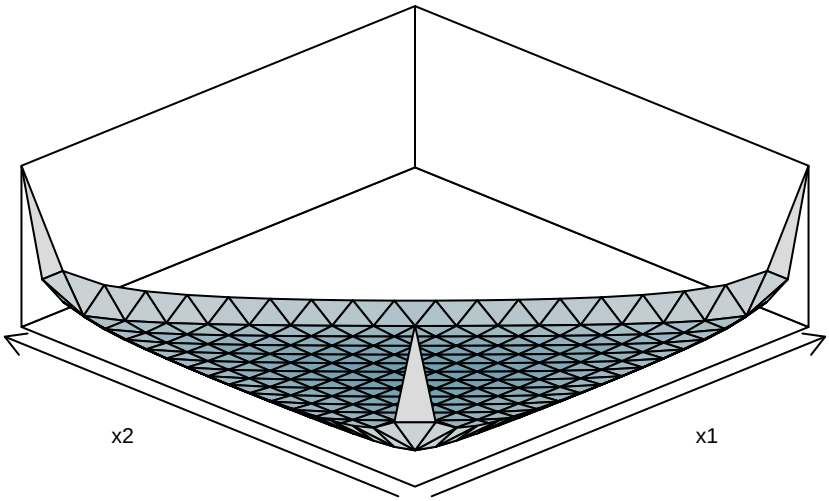
$\boldsymbol{\alpha} = (2, 2, 2)$

```
> plot (dtvpdf (2, 2, 2), TRUE)
```



$\boldsymbol{\alpha} = (0.5, 0.5, 0.5)$

```
> plot (dtvpdf (0.5, 0.5, 0.5), TRUE)
```


x2
x1