Bivariate Probability Distributions

Abby Spurdle

February 27, 2020

Convenience functions for constructing, plotting and evaluating bivariate probability distributions, including their probability mass/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.

Introduction

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

It supports the following parametric probability distributions:

- Discrete bivariate **Uniform** distributions (PMF and CDF).
- Bivariate **Binomial** distributions (PMF and CDF).
- Bivariate **Poisson** distributions (PMF and CDF).
- Bivariate **Categorical** distributions (PMF).
- Continuous bivariate **Uniform** distributions (PDF and CDF).
- Bivariate **Normal** distributions (PDF and CDF).
- Bivariate **Bimodal** distributions (PDF and CDF).
- Trivariate **Dirichlet** distributions (PDF).

And it supports the following nonparametric probability distributions:

- Bivariate **Kernel** density estimates (PDF).
- Bivariate **Empirical** cumulative distribution functions (CDF).

It's debatable whether categorical distributions are parametric or not. In this package, categorical distributions are constructed from (small) matrices of parameters, and hence have been classified as parametric. However, they could be constructed from (large) vectors of data (with many categories), which would make them more nonparametric-like.

Some of these distributions (color-coded in gold, or brown) are equivalent to the product of their marginal distributions. Others (color-coded in blue) may be equivalent to the product of their marginal distributions in some cases, but are not in generality.
(i.e. A bivariate normal distribution can be represented as the product of two univariate

normal distributions if it has no correlation).

Note that there's more than one way of formulating bivariate binomial, Poisson and bimodial distributions. I've used the simplest approaches that I could for the binomial and bimodal distributions, with the Poisson distribution adapted from Karlis and Ntzoufras (2003).

Also note that:

- The help files provide more information about specific functions.
  This vignette is designed to give an overview, references, (some) theoretical background and better examples. Also, I hope that it makes the subject of multivariate probability more intuitive and appealing.
- This package uses a system of self-referencing function objects.
  This allows us to plot and evaluate functions, without specifying their parameters, each time.
- This vignette contains non-visible R code to change the color theme.
- The functions for evaluating discrete probability distributions, coerce their arguments to integers. If you try to evaluate discrete probability distributions with non-integer arguments, you may get unexpected results.

Preliminary Code (And Required Packages)

I will load (and attach) the intoo, barsurf, bivariate and MASS packages:

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

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

I will set the rendering style for viewing PDF documents, electronically:

```
> set.bs.options (rendering.style="e")
```

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

Discrete Bivariate Uniform Distributions

We can describe a bivariate uniform distribution as the product of two univariate uniform distributions, so:

$$\begin{aligned}\mathbb{P}(X = x, Y = y) &= f_{X,Y}(x, y; a_X, a_Y, b_X, b_Y) \\ &= f_X(x; a_X, b_X) f_Y(y; a_Y, b_Y) \qquad \text{(here, x and y are ignored)} \\ &= \left[\frac{1}{b_X - a_X + 1}\right]\left[\frac{1}{b_Y - a_Y + 1}\right]\end{aligned}$$

Where a_X and a_Y are integers giving the lower bounds of X and Y, and b_X and b_Y are integers giving the upper bounds of X and Y.
(And assuming that x is in the interval $[a_X, b_X]$ and y is in the interval $[a_Y, b_Y]$).

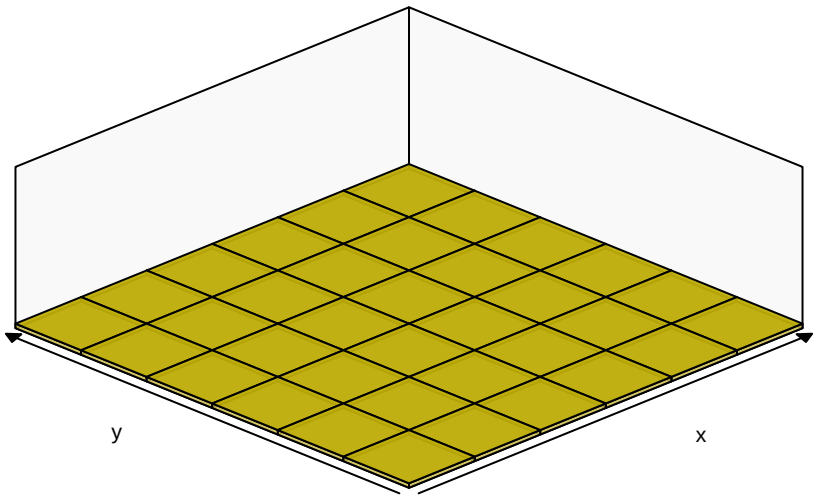
We can construct its probability mass function using the **dubvpmf** function, and its cumulative distribution function using the **dubvcdf** function.
Both constructors take four arguments, the lower bounds of X and Y, and the upper bounds of X and Y.

Here's an example, where both X and Y, can take values between one and six:

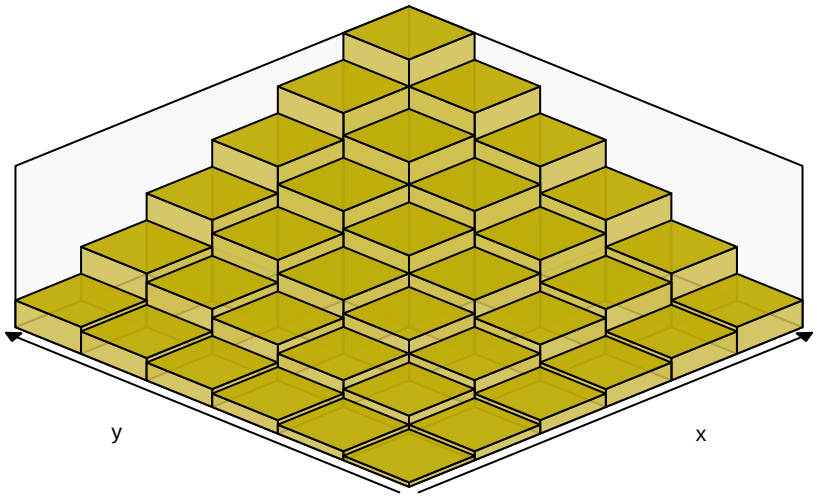
```
> f <- dubvpmf (1, 1, 6, 6)
> F <- dubvcdf (1, 1, 6, 6)
```

And we can plot the functions:

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




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



Reiterating, this package uses a system of self-referencing function objects, which allows us to plot and evaluate functions, without specifying their parameters, each time.

In both cases, we can evaluate the functions for x and y:

```
> f (2, 4)

[1] 0.02777778

> F (2, 4)

[1] 0.2222222
```

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

Note that we can compute these values from univariate distributions:
(Again, assuming that x and y are within the supported region).

```
> d.unif.pmf.eval <- function (x, a, b)    #x ignored
        1 / (b - a + 1)
> d.unif.cdf.eval <- function (x, a, b)    #x used
        (x - a + 1) / (b - a + 1)

> d.unif.pmf.eval (2, 1, 6) * d.unif.pmf.eval (4, 1, 6)

[1] 0.02777778

> d.unif.cdf.eval (2, 1, 6) * d.unif.cdf.eval (4, 1, 6)

[1] 0.2222222
```

This only applies to the probability distributions color-coded in gold.

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 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.

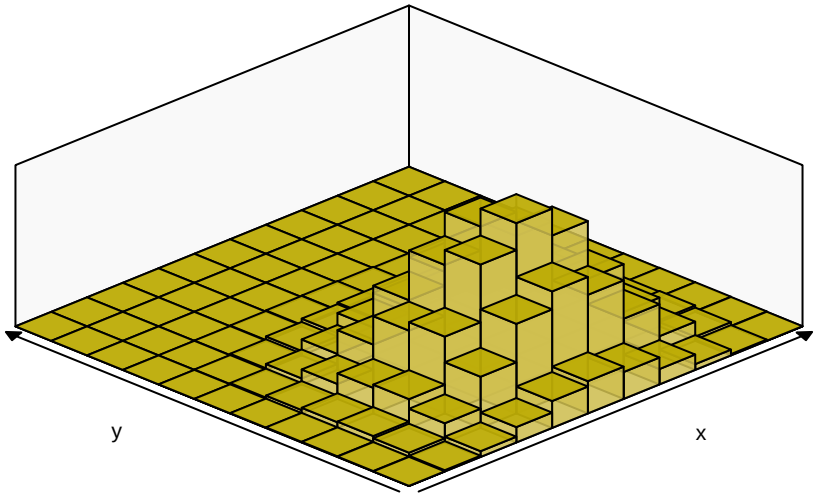
We can construct its probability mass function using the **bnbvpmf** function, and its cumulative distribution function using the **bnbvcdf** function.
It takes three arguments, the probability of the first success, the probability of the second success and the number of trials.

Here's an example where the probability of the first success is 0.5, the probability of the second success is 0.25, and there's ten trials:

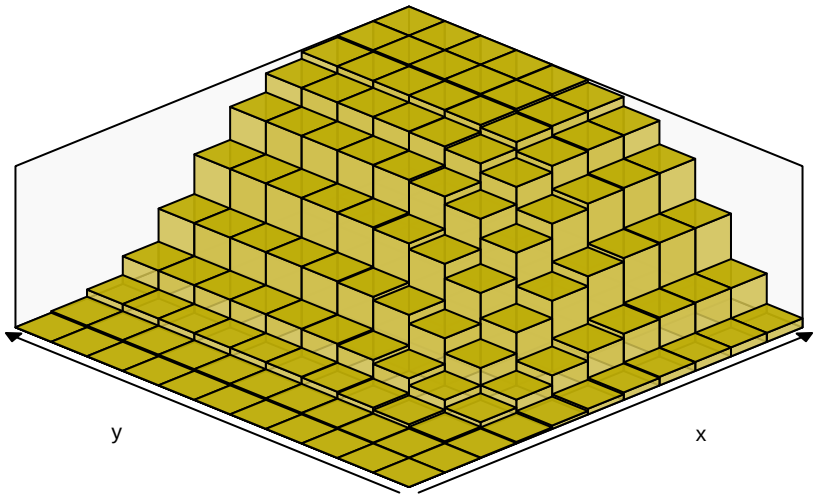
```
> f <- bnbvpmf (0.5, 0.25, 10)
> F <- bnbvcdf (0.5, 0.25, 10)
```

And again we can plot the functions:

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




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



Note that I've put the probabilities ("p") first, however, it's customary for the number of trials ("n") to go first.

If n is omitted, it defaults to one, giving a bivariate 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:
λ_1, λ_2 and λ_1 are positive real numbers.
k is an integer in the sequence 0 to $\min(x, y)$.

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

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

Contrary to the two previous probability distributions, this probability distribution is not the product of two marginal distributions.

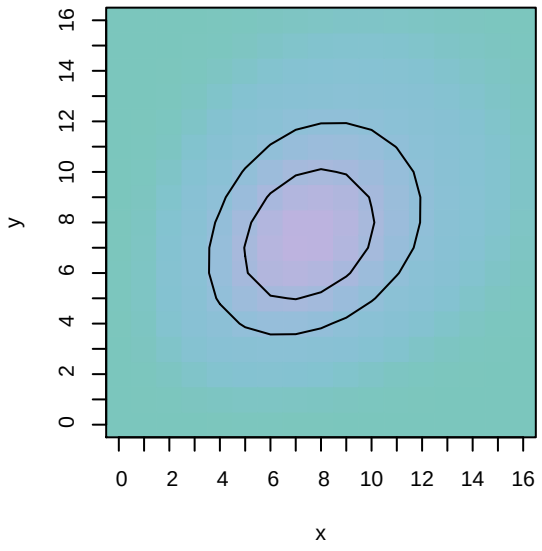
We can construct its probability mass function using the **pbvpmf** or **pbvpmf.2** functions, and its cumulative distribution function using the **pbvcdf** or **pbvcdf.2** functions.
All functions take three arguments, the first versions (with no suffix) take the three λ parameters, and the second versions (with a suffix) take the expected value of X, the expected value of Y and the covariance between X and Y.

Here's an example where the expected value of both X and Y is eight, and the covariance is two:

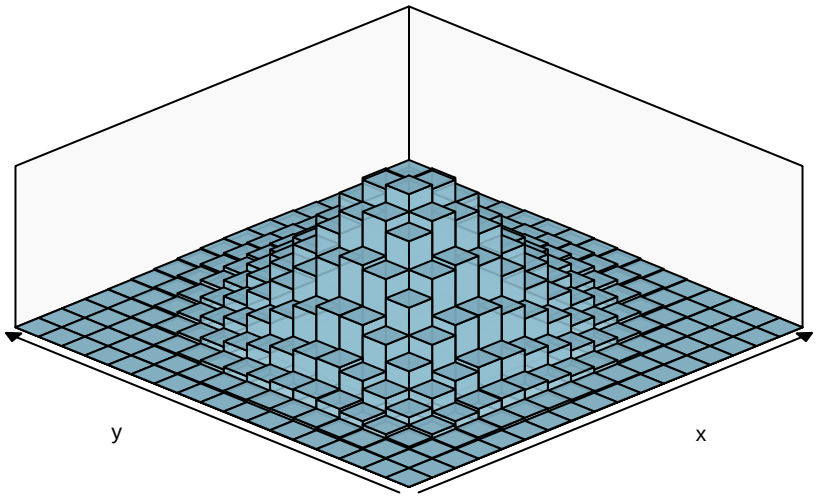
```
> f <- pbvpmf.2 (8, 8, 2)
> F <- pbvcdf.2 (8, 8, 2)
```

And plots, in 2D and 3D:

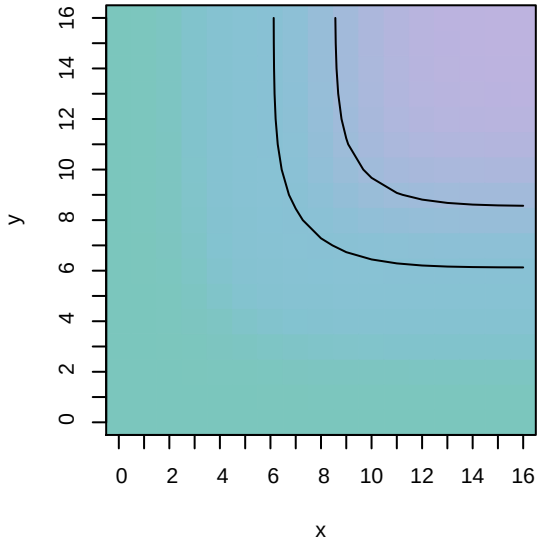
```
> plot (f)
```



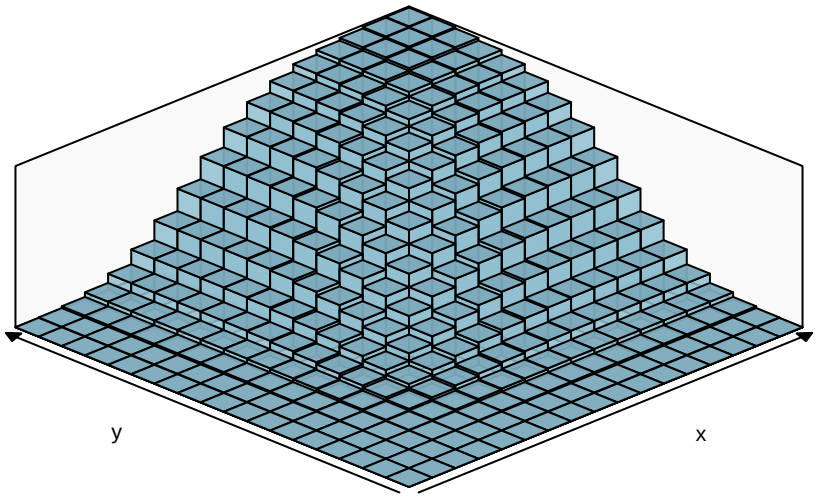
```
> plot (f, TRUE)
```




```
> plot (F)
```



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



Bivariate Categorical Distributions

We can construct a probability mass function for a categorical distribution, using the **cbvpmf** function.
It takes a single argument, a matrix of probabilities (or frequencies), preferably with row and column names.

Note that increasing rows correspond to increasing x values and increasing columns correspond to increasing y values.

Here's a example using a matrix of random frequencies:

```
> h <- sample (1:24)
> h <- matrix (h, 6, 4)
> rownames (h) <- LETTERS [1:6]
> colnames (h) <- letters [1:4]
```

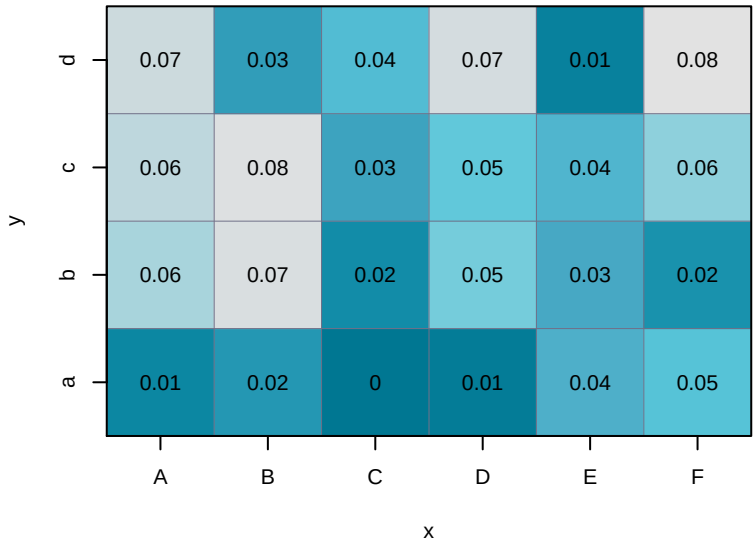
```
> h
```

```
   a  b  c  d
A  4 18 19 20
B  7 22 23  8
C  1  5  9 13
D  2 16 15 21
E 11 10 12  3
F 14  6 17 24
```

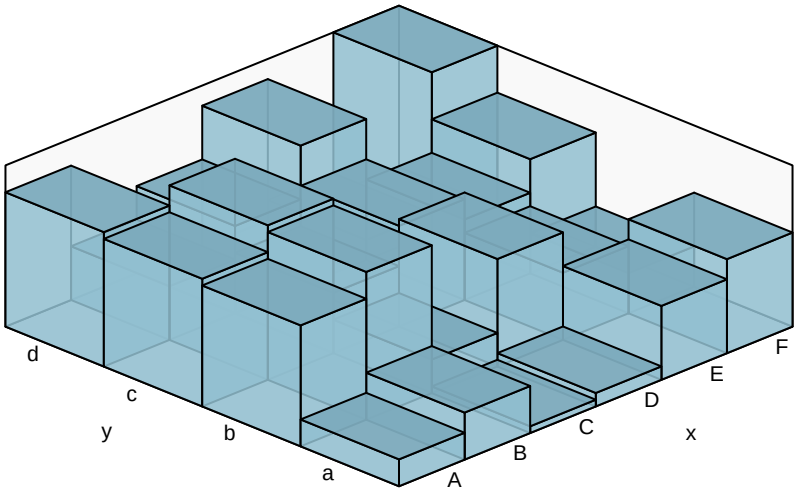
```
> f <- cbvpmf (h)
```

And plots in 2D and 3D:

```
> plot (f)
```




```
> plot (f, TRUE,
       arrows=FALSE)
```



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

```
> f (2, 4)
[1] 0.02666667
> f ("B", "d")
[1] 0.02666667
```

Continuous Bivariate Uniform Distributions

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

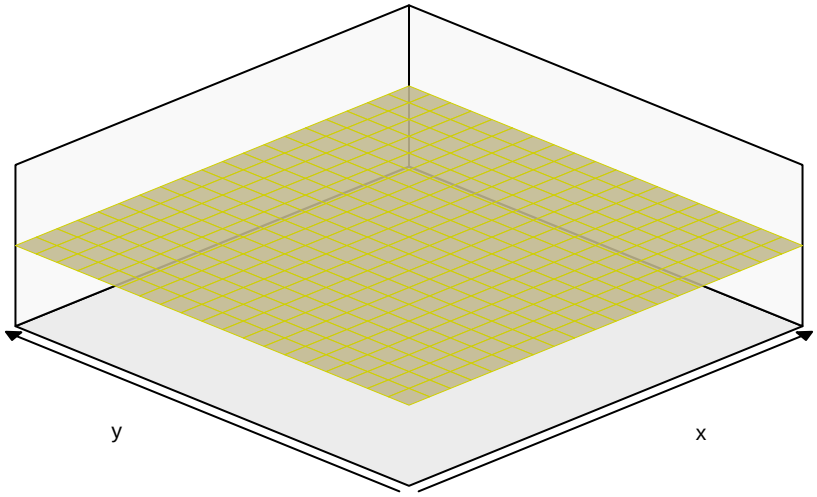
We can construct its probability density function using the **cubvpdf** function, and its cumulative distribution function using the **cubvcdf** function.
Both take four arguments, the lower bounds of X and Y, and the upper bounds of X and Y.

Here's an example, where both X and Y, can take values between zero and two:

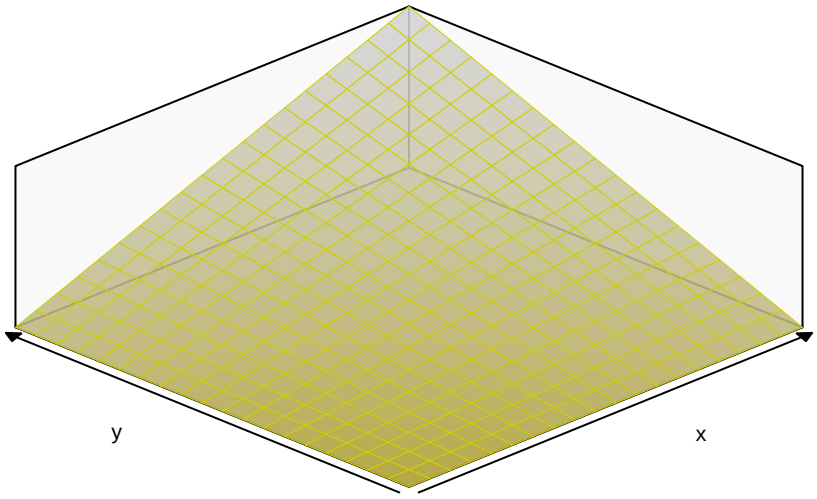
```
> f <- cubvpdf (0, 0, 2, 2)
> F <- cubvcdf (0, 0, 2, 2)
```

And plots of the functions:

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



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



Note that the density is 0.25 over the supported region, however, its vertical positioning may (incorrectly) suggest a higher value.

Bivariate Normal Distributions

This package uses the mvtnorm package to evaluate bivariate normal distributions.
Please refer to that package for technical details.

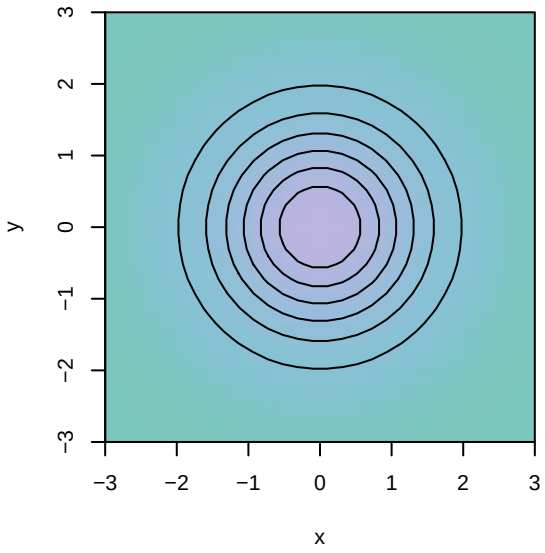
We can construct a probability density function for the bivariate normal distribution using the **nbvpdf** or **nbvpdf.2** functions, and its cumulative distribution function using the **nbvcdf** or **nbvcdf.2** functions.
All functions take five parameters. The first functions (with no suffix) take the means of X and Y, the standard deviations of X and Y, and their correlation. The second functions (with the suffix) take the means of X and Y, the variances of X and Y, and their covariance.

Here's an example with zero means, standard deviations of one, and no correlation:
(Essentially, a bivariate generalization of the "standard normal distribution").

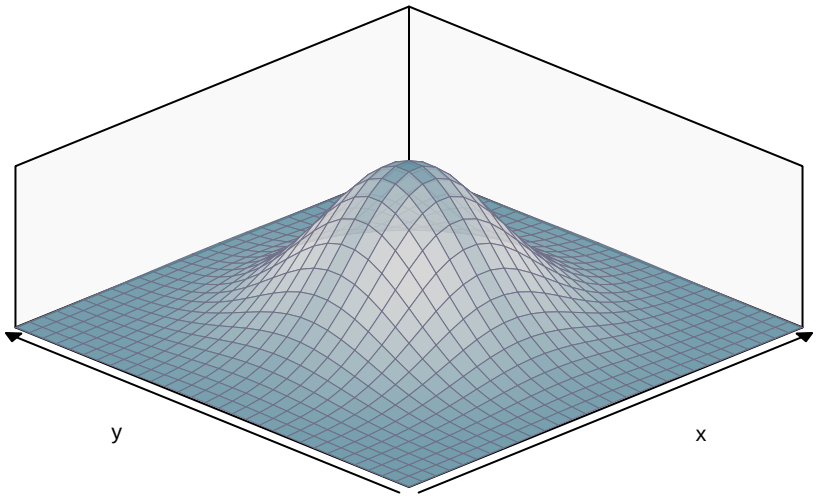
```
> f <- nbvpdf (0, 0, 1, 1, 0)
> F <- nbvcdf (0, 0, 1, 1, 0)
```

And plots in 2D and 3D:

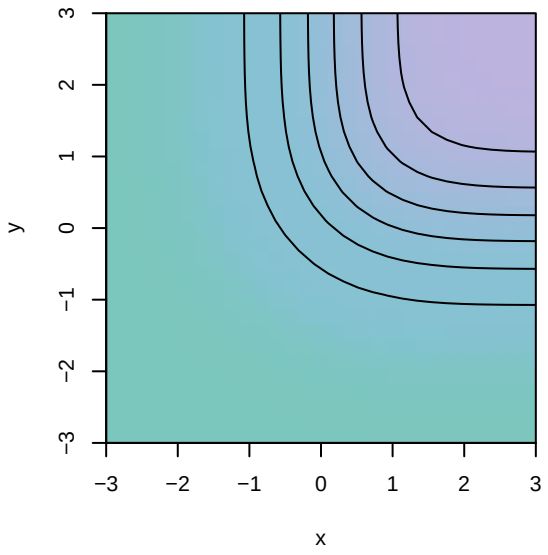
```
> plot (f)
```



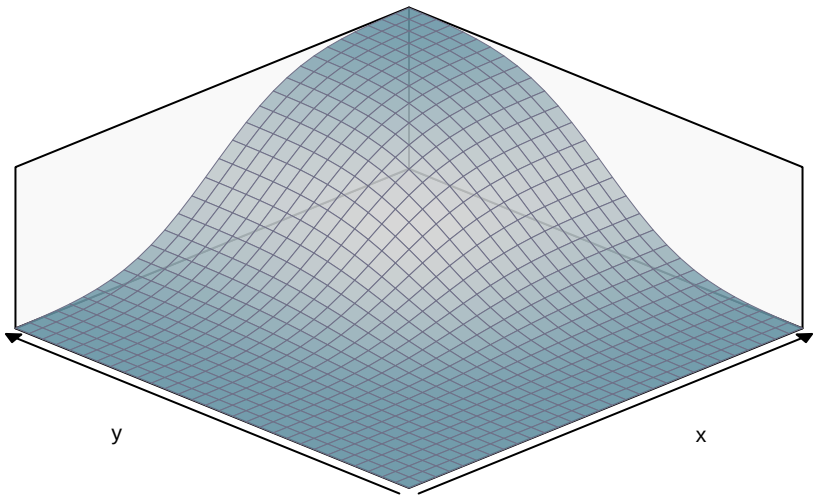

```
> plot (f, TRUE)
```



```
> plot (F)
```




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



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

Bivariate Bimodal Distributions

It's possible to construct a bivariate bimodal probability density function by taking two bivariate normal probability density functions, then adding their densities together, and then dividing by two.

We can construct such a probability density function using the **bmbvpdf** or **bmbvpdf.2** functions, and its cumulative distribution function using **bmbvcdf** or **bmbvcdf.2** functions. All functions take eight arguments, and follow the same principles as the normal distributions, discussed in the previous section. The first four arguments are the means and standard deviations (or variances) of the first component distribution. The last four ar-

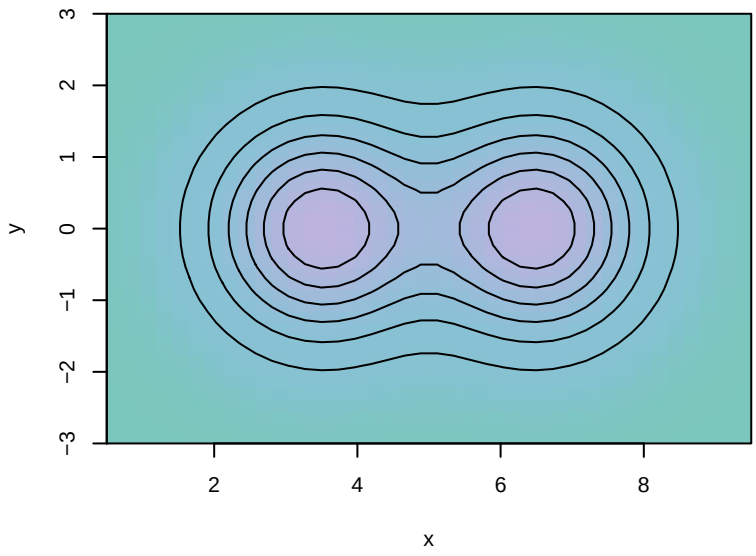
guments are the means and standard deviations (or variances) of the second component distribution.

Here's a an example, where the component means of X are 3.5 and 6.5, the component means of Y are zero, and all standard deviations are one:

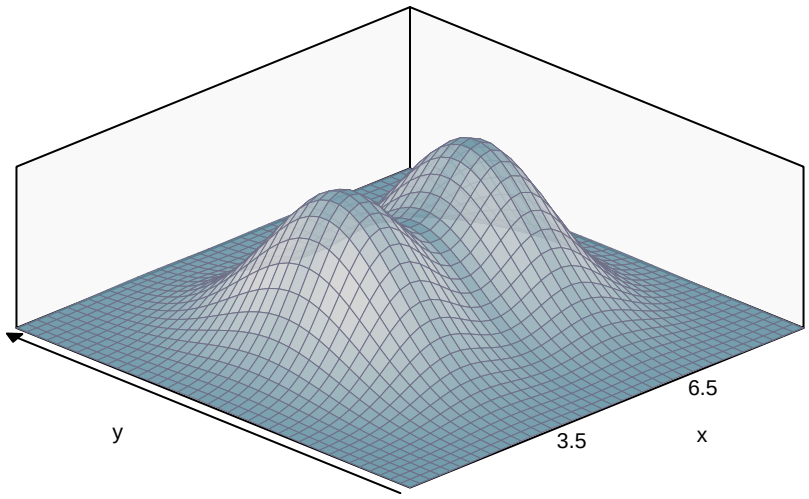
```
> f <- bmbvpdf (
       3.5, 0, 1, 1,    #first component distribution
       6.5, 0, 1, 1)    #second component distribution
> F <- bmbvcdf (
       3.5, 0, 1, 1,    #first component distribution
       6.5, 0, 1, 1)    #second component distribution
```

And plots in 2D and 3D:

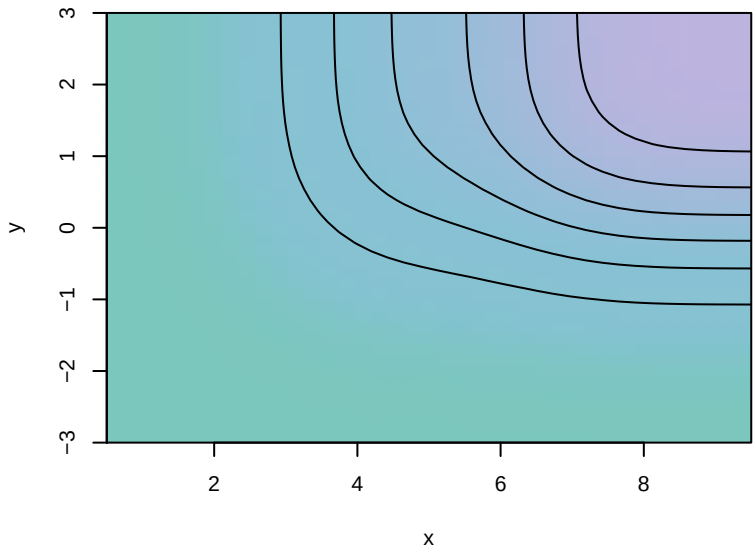
```
> plot (f)
```



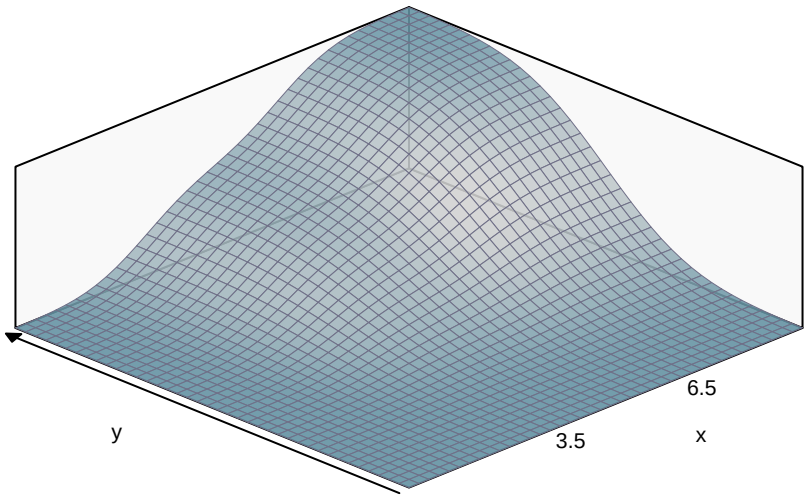
```
> plot (f, TRUE,
       arrows = c (FALSE, TRUE), xat = c (3.5, 6.5) )
```




```
> plot (F)
```



```
> plot (F, TRUE,
       arrows = c (FALSE, TRUE), xat = c (3.5, 6.5) )
```



Note that this method (of adding normal distributions) is similar kernel smoothing, discussed later.

Trivariate Dirichlet Distributions

Dirichlet distributions with three variables are similar to other probability distributions with two variables.
(Because it's possible to compute the third variable from the first two variables).

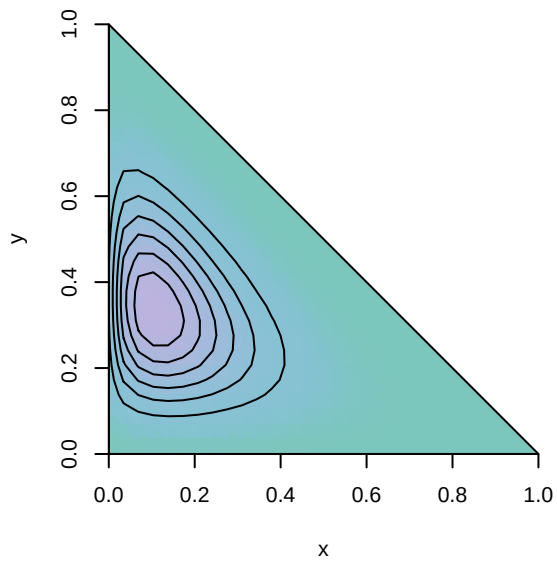
We can construct their probability density function using the **dtvpdf** function.
It takes three α parameters.

Here's an example with α parameters of two, four and six:

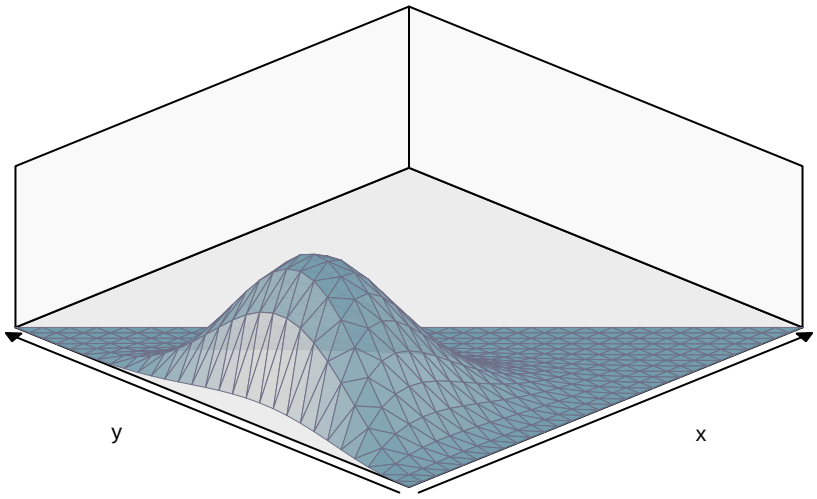
```
> f <- dtvpdf (2, 4, 6)
```

And plots in 2D and 3D:

```
> plot (f)
```

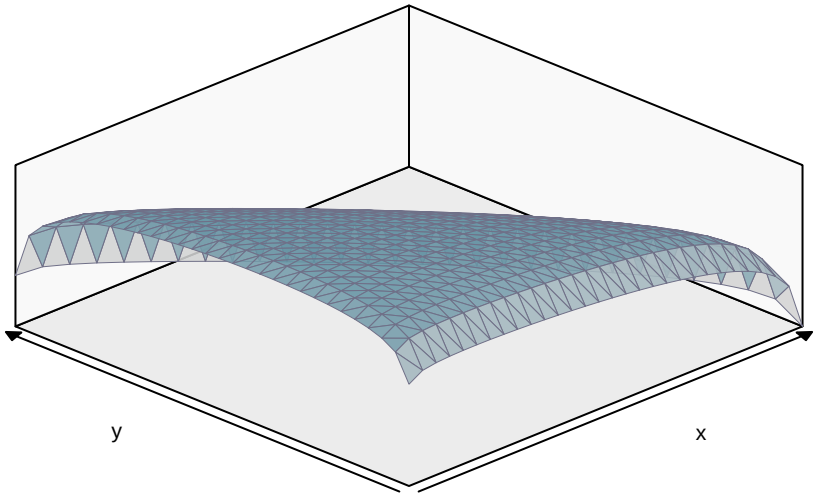



```
> plot (f, TRUE)
```



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 later.

Bivariate Kernel Density Estimates

This package uses the KernSmooth package to produce bivariate kernel density estimates. Please refer to that package for technical details.

We can construct a probably distribution representing bivariate kernel density estimates using the **kbvpdf** function.
It takes four arguments, two equal length vectors of data, and two bandwidth parameters.

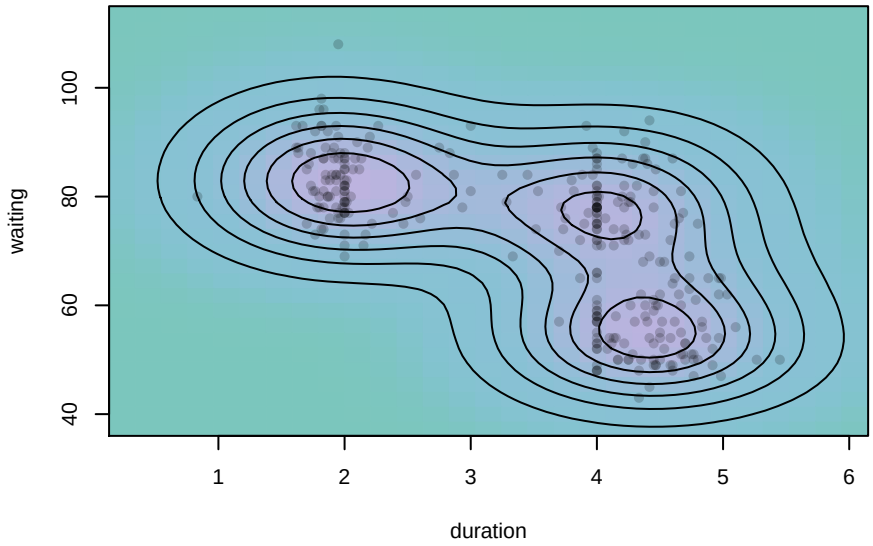
I've adapted this example from KernSmooth:
(Which has a bandwidth parameter of 0.7 for duration, and 7 for waiting).

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

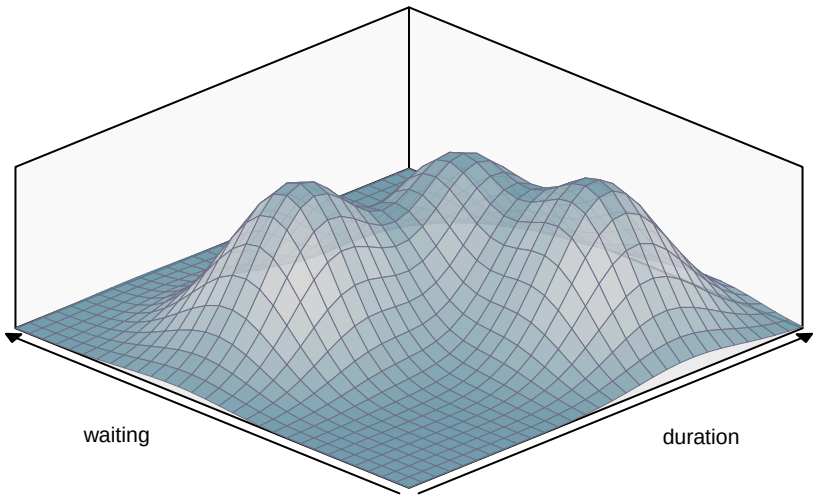
```
> fh <- kbvpdf (duration, waiting, 0.7, 7)
```

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

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




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




```
> detach (geyser)
```

Unlike other probability distributions in this package, you can't evaluate the function, fh.

Note that the probhat package provides more tools for kernel smoothing.

Bivariate Empirical Cumulative Distribution Functions

Like bivariate kernel density estimates, bivariate ECDFs are computed from vectors of data.

The resulting probability distribution is a step function, however in general, they represent the distribution of continuous random variables.

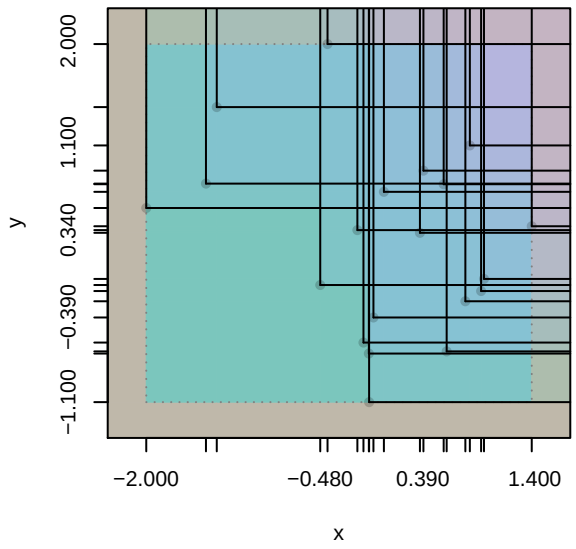
We can construct it using the **ebvcdf** function.
It takes two arguments, two equal length vectors of observations.

Here's an example, using some random data:
(Which should have low correlation).

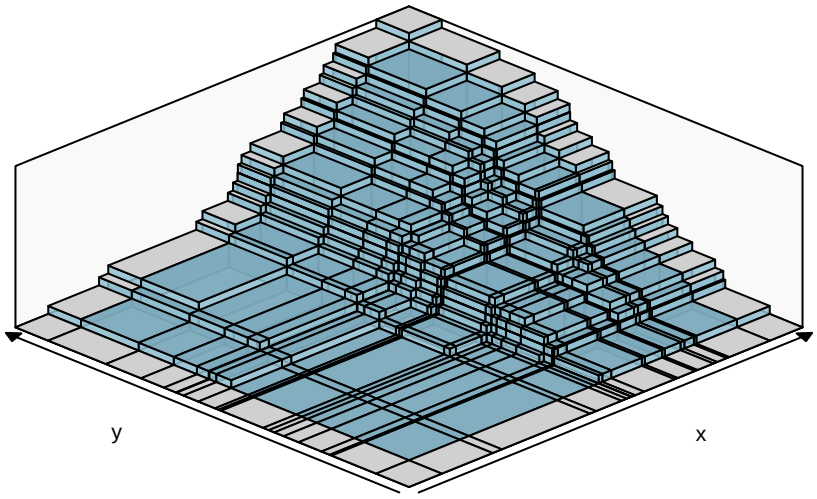
```
> x <- rnorm (20)
> y <- rnorm (20)

> Fh <- ebvcdf (x, y)
```

And plots in 2D and 3D:

```
> plot (Fh,
        xyrel="f")
```



```
> plot (Fh, TRUE)
```



Note that by default, if the number of observations is less than or equal to forty, the function is plotted as a (discrete) step function, with an extrapolated region outside the range of observed values. However, if the number of observations is larger than forty, 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. (2020). barsurf: Heatmap-Related Plots and Smooth Multiband Color Interpolation.

Genz, A., Bretz, F., Miwa, T., Mi, X. & Hothorn, T. (2020). mvtnorm: Multivariate Normal and t Distributions

Wand, M. (2019). KernSmooth: Functions for Kernel Smoothing Supporting Wand & Jones (1995).

Spurdle, A. (2019). probhat: Multivariate Generalized Kernel Smoothing and Related Statistical Methods.

Ripley, B. (2019). 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

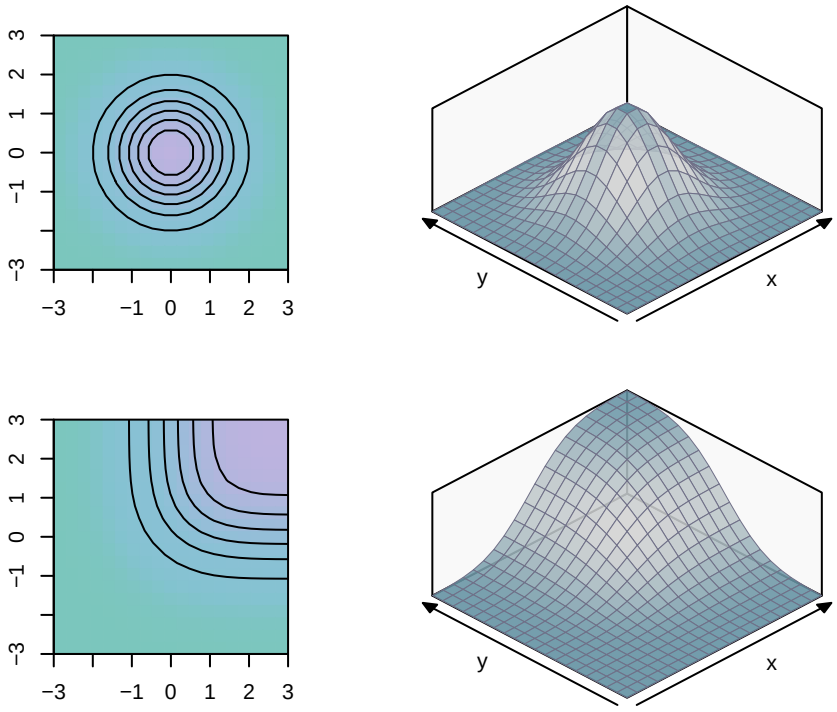
We can compare different normal distributions using different correlation or covariance parameters.

First, let's consider the bivariate distribution from the earlier section with zero correlation:

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

```
  X Y
X 1 0
Y 0 1
```

```
> plot (f1, all=TRUE, n=20)
```

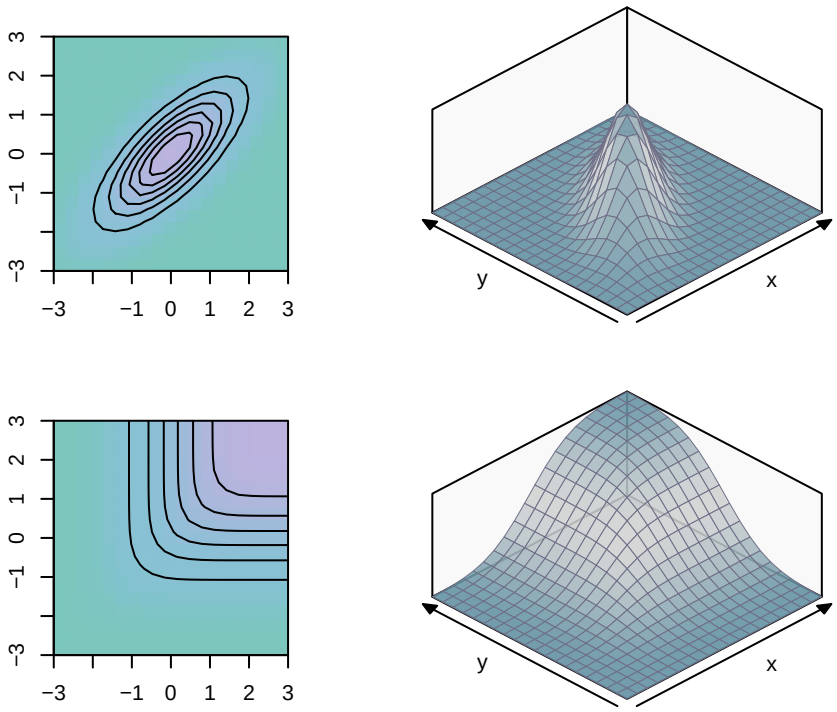


Second, let's consider bivariate distributions with positive correlation:
(Note that if both the standard deviations are one, then the correlation will equal the covariance).

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

```
     X    Y
X 1.00 0.75
Y 0.75 1.00
```

```
> plot (f2, all=TRUE, n=20)
```

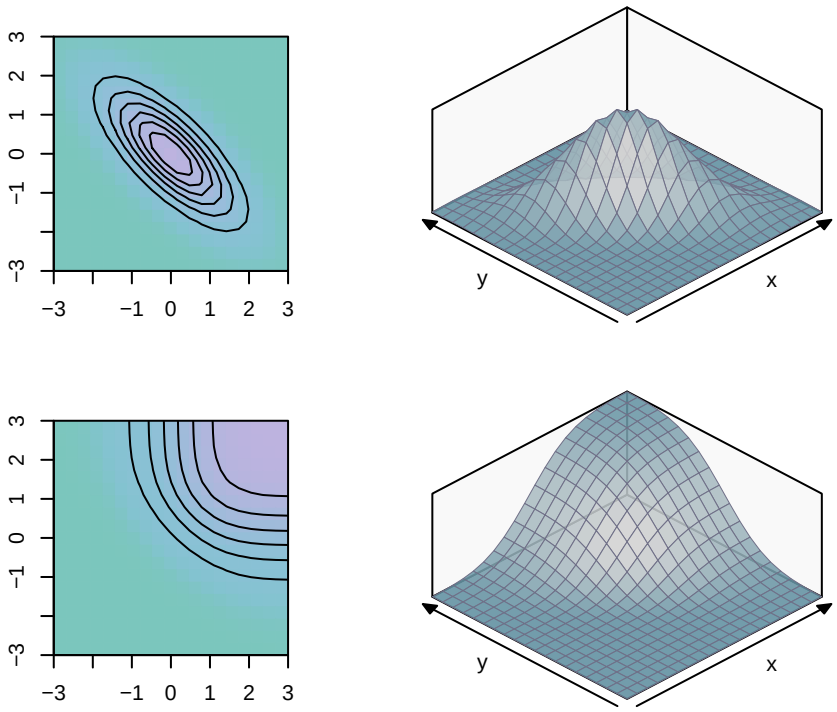


Third, let's consider bivariate distributions with negative correlation:

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

```
      X     Y
X  1.00 -0.75
Y -0.75  1.00
```

```
> plot (f3, all=TRUE, n=20)
```

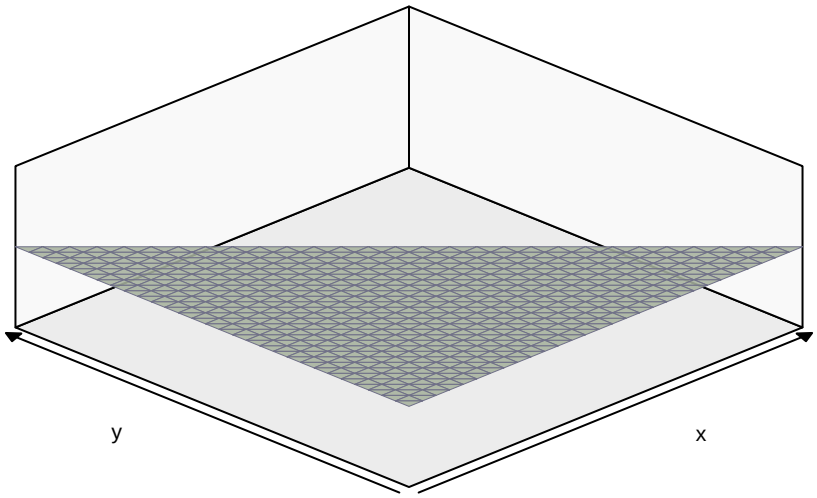


Note that currently, the all=TRUE option requires the PMF or PDF rather than the CDF.

Appendix B:
Comparing Dirichlet Distributions

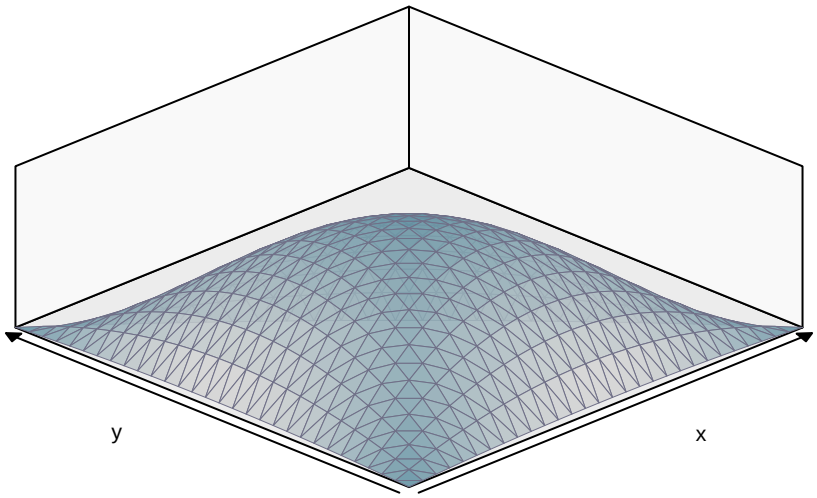
$\boldsymbol{\alpha} = (1, 1, 1)$

```
> 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)
```