barsurf 0.7.0

Contour Plots, 3D Plots, Vector Fields and Heatmaps

Abby Spurdle

January 20, 2021

Combined contour/heatmap plots, 3d bar/surface plots, 2d/3d triangular plots, isosurface plots and 2d/3d vector fields. By default, uses vector graphics, but it's possible to use raster graphics for regular heatmaps. Builds on the colorspace package (Zeileis, et al., 2020 <doi:10.18637/jss.v096.i01>), by supporting smooth multiband color interpolation, in sRGB, HSV and HCL color spaces.

Introduction

This package contains plotting functions for visualizing mathematical functions of two to three variables.

This includes:

- Combined contour-heatmap plots, for discretely-spaced data.
- Combined contour-heatmap plots, for continuously-spaced data.
- 3d bar and surface plots.
- Triangular contour and surface plots.
- Isosurface plots.
- 3d-based contour-heatmap plots, for three continuous variables.

Also, there is/are:

- A wrapper function, for matrix visualization.
- Plots of 2d and 3d vector fields.

The main plotting functions take matrices/arrays as their main arguments.

But there are also functional versions, which take a function as their main argument, along with xlim/ylim values.
(These call the main plotting functions, but compute the input matrices/arrays, along with some arguments).

This package uses the base graphics system, however, uses a different color system.

A system of "litmus" objects support smooth multiband color interpolation, primarily for heatmaps and surface plots, but may be used for other purposes.

Note that:

- By default, all plots use vector graphics.
  But there's an option to use raster graphics for regular heatmaps.
  (This applies to the **plot_dfield** and **plot_cfield** functions, discussed later).
- Currently, 3d plots use a diamond-like orthographic/dimetric projection, with a fixed viewing angle.
  Plots can't be rotated, however, most plots can be reversed along one or more axes.
  Reversing a 3d plot, in both the x and y axes, is equivalent to a 180°rotation.

Also, there are global options, which can change a variety of settings, including the reference arrows and default colors.

Refer to **set.bs.options**.

Preliminary Code (And Required Packages)

I will load (and attach) the barsurf and misc3d packages.

```
> library (barsurf)
> library (misc3d)
```

Note that the barsurf package imports the kubik and colorspace packages.

Also, the misc3d package needs to be installed and loaded, in order to plot isosurfaces.

```
> set.bs.options (rendering.style="pdf", theme="blue")
```

The "pdf" rendering style, uses finer lines.

And I'm setting the theme to blue.
(In principle, this is unnecessary because blue is the default).

Functions of Two Discrete Variables (And Discretely-Spaced Scalar Fields)

The functions **plot_dfield** and **plot_bar** can used to plot matrices representing discretely-spaced scalar fields.

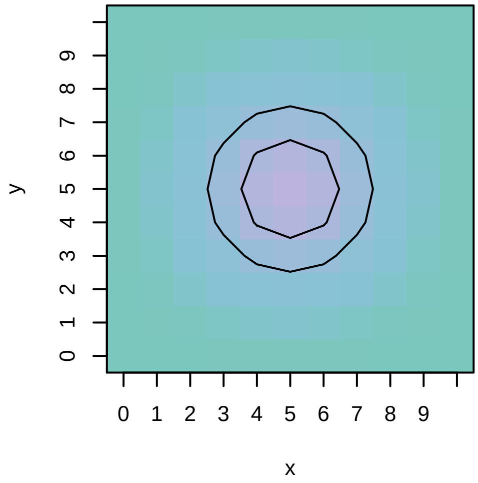
Also, the functions **plotf_dfield** and **plotf_bar** may be used, which take a function, along with xlim and ylim arguments.

Let's construct a simple matrix representing the product of two binomial distributions:

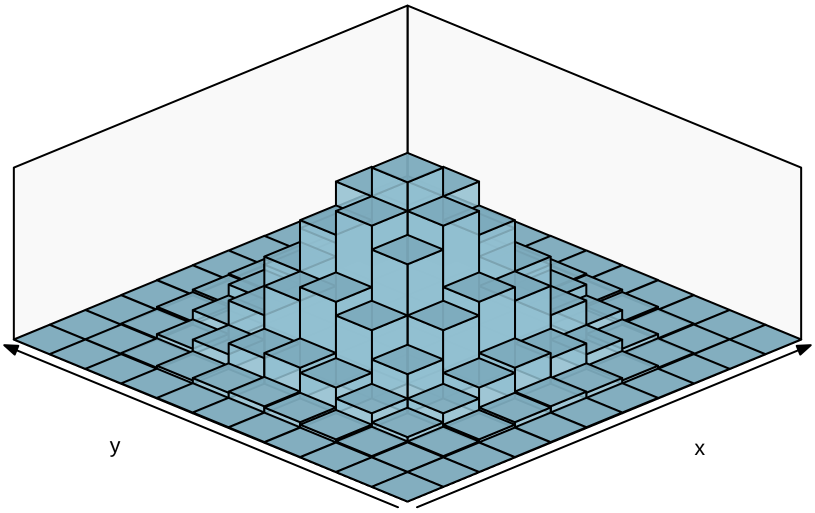
```
> n <- 10
> p <- 0.5

> x <- y <- 0:10
> f <- function (x, y, n, p)
      dbinom (x, n, p) * dbinom (y, n, p)
> fv <- outer (x, y, f, n, p)
```

Then plot in 2d and 3d:

```
> plot_dfield (x, y, fv)
```



```
> plot_bar (,,fv)
```



Note that it's possible to specify the third argument, using two commas, as in the above example.

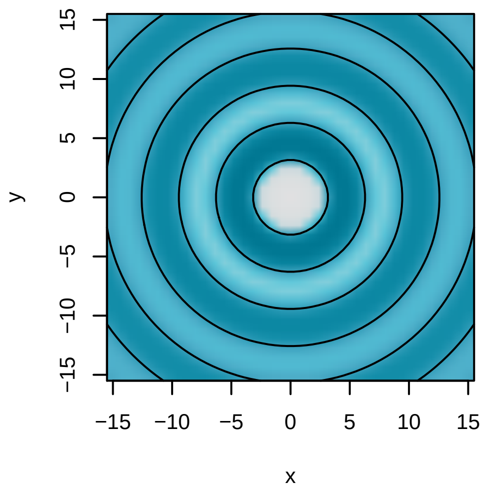
Functions of Two Continuous Variables (And Continuously-Spaced Scalar Fields)

The functions **plot_cfield** and **plot_surface** can used to plot matrices representing continuously-spaced scalar fields.

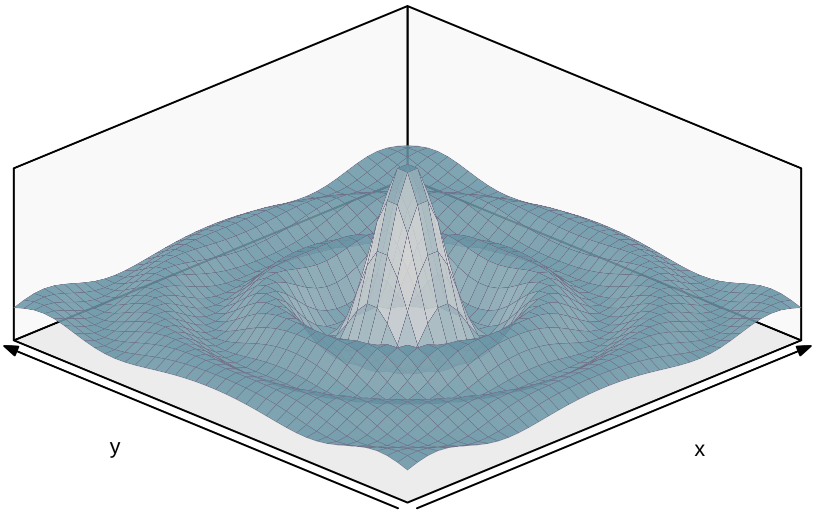
Also, the functions **plotf_cfield** and **plotf_surface** may be used, which take a function, along with xlim, ylim and n arguments.

Let's plot the rotated sinc function, adapted from the graphics::**persp** examples:

```
> plotf_cfield (rotated.sinc, c (-15.5, 15.5), fb=0, n=40,
      raster=TRUE, hcv=TRUE)
```



```
> plotf_surface (rotated.sinc, c (-15.5, 15.5), n=40)
```



In the contour plot, the contour values have been set to zero, and the high color variation option has been used.

Also note that the heatmap does not have a smooth appearance.

The simplest solution (for a smooth appearance) is to increase the n value.
However, there's an appendix later, which gives an example of plotting one heatmap on top of another, in the area of high curvature.
This should be more efficient.

Triangular Plots

The functions **plot_tricontour**, **plot_trisurface**, **plotf_tricontour** and **plotf_trisurface** can be used to produce triangular plots.

They're similar to the **plot_cfield**, **plot_surface**, **plotf_cfield** and **plotf_surface** functions.

In the matrix version, the matrix must be square, and only upper left part of the matrix (including the diagonal) is used.

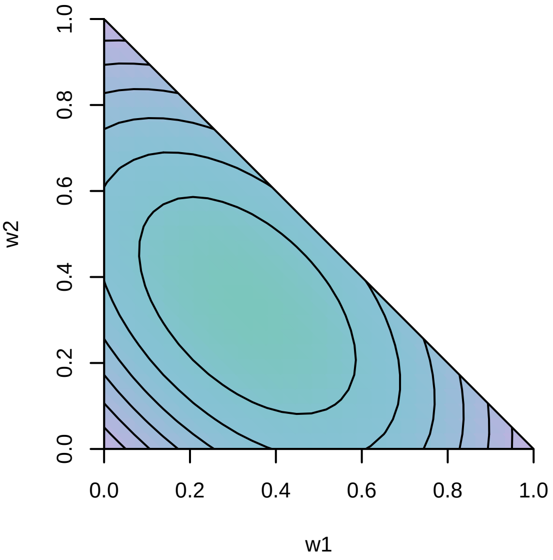
In functional versions, the function needs to be a function of two variables, which takes values between 0 and 1, where they (along with a third implicit variable) sum to one.

If you need to plot a function that doesn't have these properties, then you need to create a wrapper function.

Here's a simple example:

```
> f <- function (w1, w2, w3 = 1 - w1 - w2)
      (w1 - 1 / 3)^2 + (w2 - 1 / 3)^2 + (w3 - 1 / 3)^2
```

```
> plotf_tricontour (f, xlab="w1", ylab="w2")
```



Note that the x and y arguments are ignored.

Isosurface Plots
(For Functions/Equations of Three Variables)

Re-iterating, these functions require the misc3d package to be installed and loaded.

The **plot_isosurface** and **plotf_isosurface** functions can be used to plot 3d contour plots.

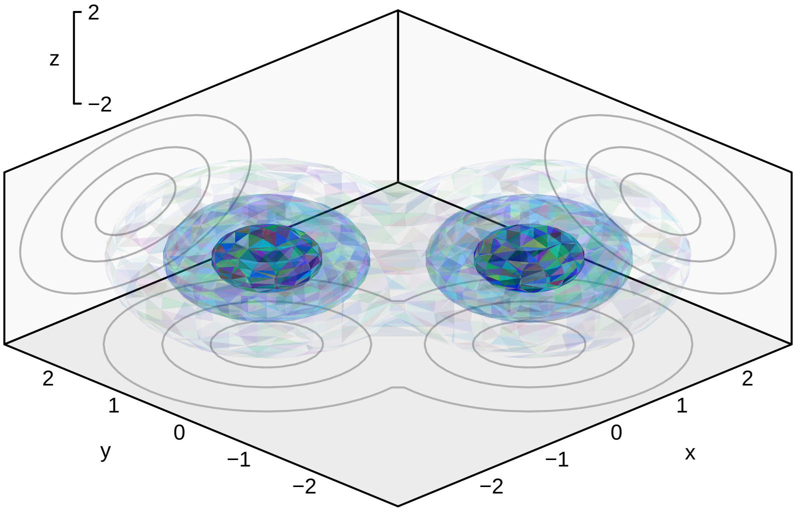
Unlike previous plotting functions, the fv argument is a three dimensional array rather than a matrix.
(And there's a z coordinate, separate from the function value).

The first two variables have the same interpretation as other 3d plots in this package, with the third variable giving the height. Except that x, y and z describe coordinates of the fv array, not the resulting isosurfaces.

Also, it's possible for the the x, y, z, fv and n arguments to be lists, one list element for each isosurface.

The following example plots the **bispherical.dist** function, with three isosurfaces.

```
> plotf_isosurface (bispherical.dist,
        c (-3, 3),, c (-2, 2), nsurfaces=3,
        ref.arrows=FALSE, pconstants = c (1, 1, 0) )
```



Note that my functions use misc3d::**computeContour3d** function to compute isosurfaces, which in turn, uses the "Marching Cubes" algorithm.

3D-Based Contour-Heatmap Plots

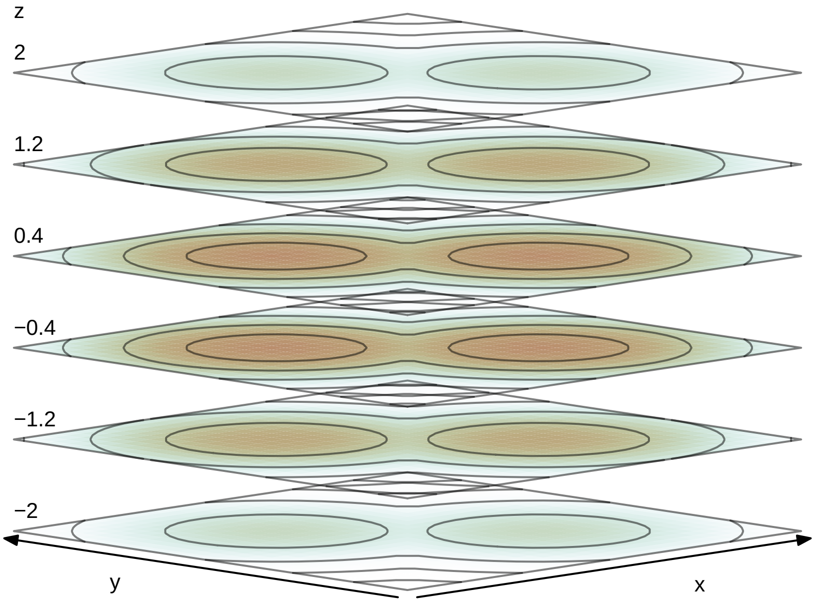
Here, 3d-based combined contour-heat plots contain a set of 2d slides (or slices).

The **plot_cfield3** and **plotf_cfield3** functions are similar to the **plot_cfield** and **plotf_cfield** functions.

Unlike previous plotting functions, the main argument, fv, is a is a list of two or more matrices.
One matrix for each slide.

The following example also plots the **bispherical.dist** function, but gives slides rather than isosurfaces:

```
> plotf_cfield3 (bispherical.dist, c (-3, 3),, c (-2, 2), emph="l")
```



2D Vector Fields

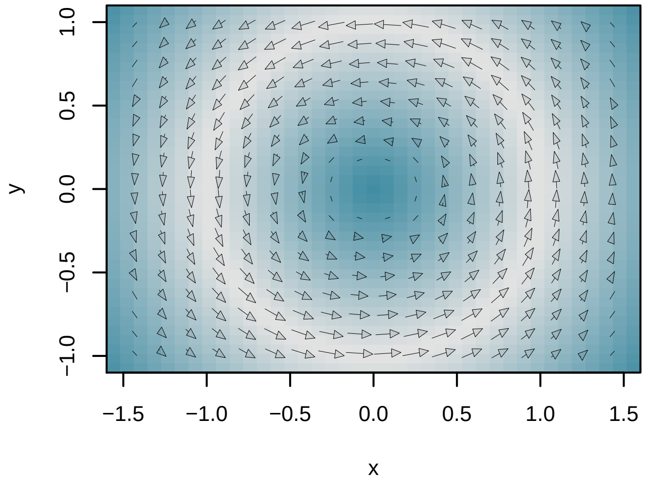
The **plot_vec** and **plotf_vec** functions can be used to produce plots of 2d vector fields.

They're similar to the **plot_cfield** and **plotf_cfield** functions.

In the matrix version, there's two input matrices, dx (for the x component) and dy (for the y component). In the functional version, the function needs to return a two-column matrix, with the first column being the x component and the second column being the y component.

Here's a simple example:

```
> plotf_vec (circular.field, c (-1.6, 1.6), c (-1.1, 1.1) )
```



3D Vector Fields

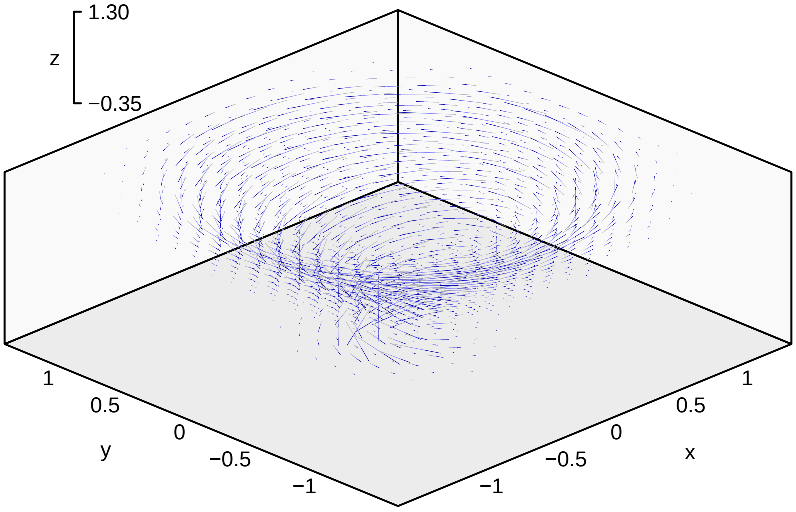
The **plot_vec3** and **plotf_vec3** functions can be used to produce plots of 3d vector fields.

They're similar to the **plot_vec** and **plotf_vec** functions, from the previous section.

In the array version, there's three input arrays, dx, dy and dy.
In the functional version, the function needs to return a three-column matrix.

Here's a simple example:

```
> plotf_vec3 (plughole.field, c (-1.5, 1.5),, c (-0.35, 1.3),
      ref.arrows=FALSE)
```



Plotting Colors (Overview)

This package uses the base graphics system, but largely, uses a different system for color.

Plotting colors, can be changed by:

- Changing the theme, in a function call.
- Changing the theme, via a global option.
- Changing specific global options, for particular plotting functions.
- Setting color-related arguments, when calling the plotting functions.

Many functions have a theme argument, which is an optional string that can be set to any of the supported themes.

Refer to the **set.bs.theme** and **set.bs.options** functions, for more information on the supported themes, and global options

This package uses a system of "litmus" objects, for heatmaps and surface plots.

A litmus object is a function that maps a numeric vector to a character vector of R color strings.
(Suitable for use with the base graphics system).

Also, there are multi-litmus objects that combine two or more litmus objects together, and litmus-fitting functions, that fit a litmus object to a vector of data.

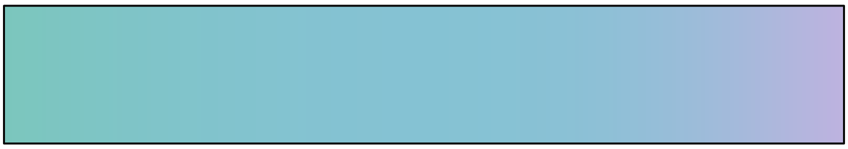
Plotting functions that use heatmaps, have colf and colff arguments.
The colf argument is for a litmus object (refer to Appendix A, for an example) and the colff argument is for a litmus-fitting function (refer to Appendix B, for an example).

There are a range of predefined litmus objects and litmus-fitting functions, but it's also possible to define your own.

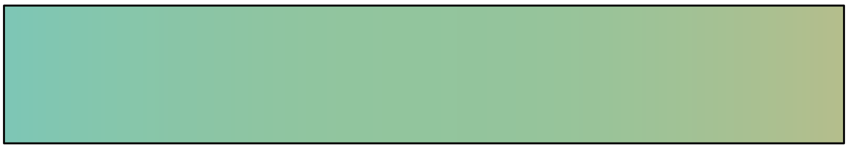
Predefined Litmus Objects and Litmus-Fitting Functions

This package has several predefined litmus objects:

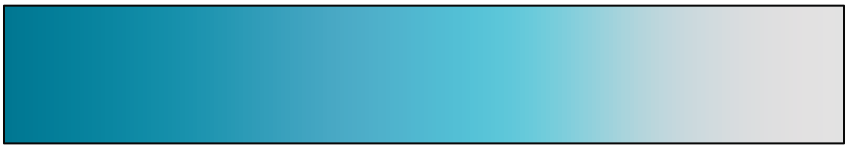
```
> plot (blue.litmus () )
```



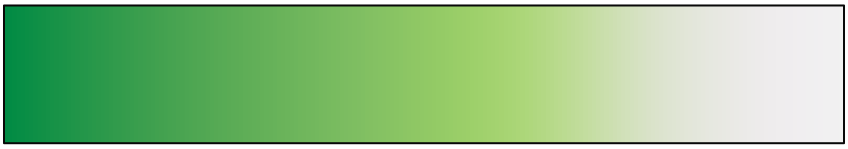
```
> plot (green.litmus () )
```



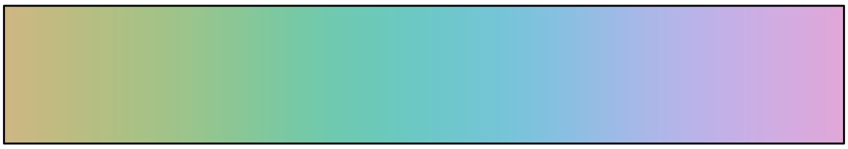
```
> plot (blue.litmus.hcv () )
```



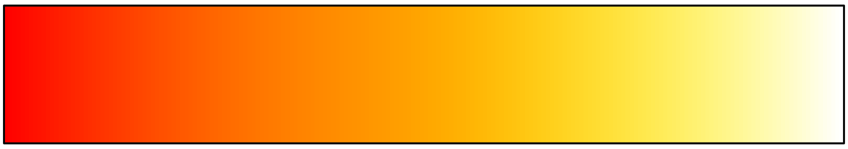
```
> plot (green.litmus.hcv () )
```



```
> plot (rainbow.litmus () )
```

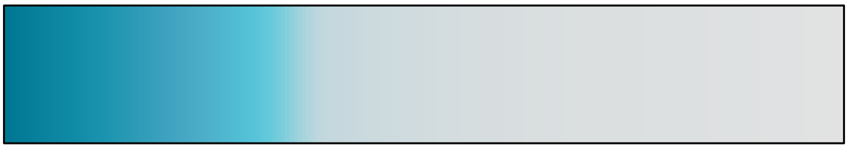


```
> plot (heat.litmus () )
```



Note that in general, it's easier to create litmus objects via litmus-fitting functions:

```
> u <- rnorm (30, 4, 1)^3
> plot (blue.litmus.fit.hcv (u) )
```



Note that the fitted litmus object doesn't necessarily appear smooth, however, when used with heatmaps, it should look better.

Also note that the rainbow litmus objects are adapted from the **rainbow_hcl** function from the colorspace package.

User-Defined Litmus Objects and Litmus-Fitting Functions

The **litmus** and **litmus.spline** functions can be used to create litmus objects.

Also, we can create a function to fit litmus objects by wrapping the **litmus.fit** function.

All three functions take a 3-column or 4-column matrix, representing a set of length-3 or length-4 color vectors, along with a string specifying the input color space.

The **litmus** function takes lower and upper limits.

The **litmus.spline** function takes a vector of knots.
And the **litmus.fit** function takes a vector of data.

A matrix of color vectors:

```
> .colvs <- cbind (c (c (100, 150, 200, 250) ), 35, 75)
> rownames (.colvs) <- c ("first color", "(2nd)", "(3rd)", "last color")
> colnames (.colvs) <- c ("H", "C", "L")
```

```
> .colvs
```

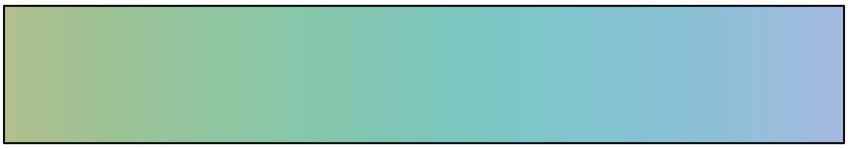
```
              H  C  L
first color 100 35 75
(2nd)       150 35 75
(3rd)       200 35 75
last color  250 35 75
```

Note that its not necessary to set row and column names.
(I've just done that to make the matrix easier to interpret).

A user defined function to create a litmus object:

```
> my.litmus <- function (a=0, b=1)
      litmus (a, b, .colvs, color.space="HCL")
```

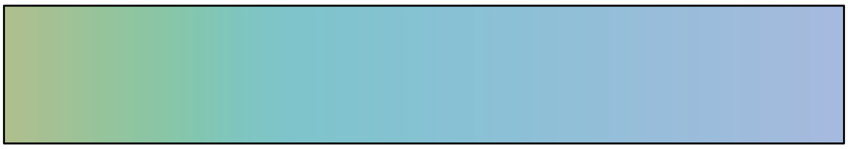
```
> plot (my.litmus () )
```



Or a litmus-fitting function:

```
> my.litmus.fit <- function (x, ...)
      litmus.fit (x, .colvs, color.space="HCL", ...)
```

```
> plot (my.litmus.fit (u) )
```



References

R Packages

Spurdle, A. (2020). kubik: Cubic Hermite Splines and Related Optimization Methods

Ihaka, R., Murrell, P., Hornik, K., Fisher, J. Stauffer, R., Wilke, C., McWhite, C., & Zeileis, A. (2020). colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes

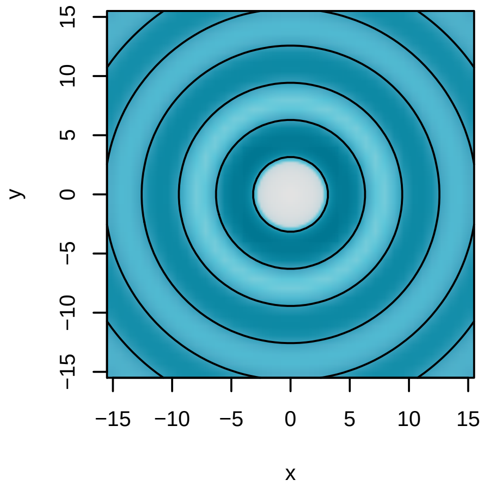
Feng, D., & Tierney, L. (2020) misc3d: Miscellaneous 3D Plots

Notes

The colorspace package contains further color-related references.

Appendix A:
Improved Contour Plot
(and an example of using the colf argument)

```
> x <- y <- seq (-15.5, 15.5, length.out=60)
> fv <- outer (x, y, rotated.sinc)
> colf <- blue.litmus.fit.hcv (fv)

> #larger heatmap
> plotf_cfield (rotated.sinc, range (x), n=40,
      contours=FALSE, raster=TRUE, colf=colf)
> #smaller heatmap
> plotf_cfield (rotated.sinc, c (-4, 4), n=40,
      add=TRUE, contours=FALSE, raster=TRUE, colf=colf)
> #contour lines
> plot_cfield (x, y, fv,
      add=TRUE, fb=0, heatmap=FALSE)
```



Note that the litmus object has to be constructed outside the plotting functions, because a common litmus object is required.
(If the litmus object was constructed within the plotting functions by setting the colff argument, then each heatmap would be on a different color scale).

Appendix B:
Hot and Cold Style Colors
(and an example of using the colff argument)

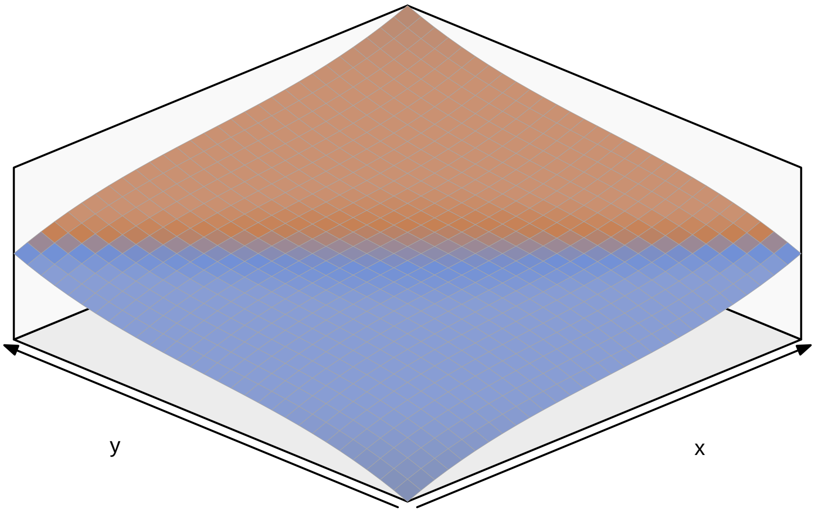
```
> f <- function (x, y) x + x^3 + y + y^3
```

A wrapper for the hot.and.cold.fit function:

```
> tempff <- function (x)
      hot.and.cold.fit (x, t = c (-0.2, 0.2) )
```

And setting the colff argument:

```
> plotf_surface (f, c (-1, 1),
      grid.color="grey65",
      gradient.shading=FALSE,
      colff=tempff)
```



The colors aren't limited to red and blue.

The following uses green and brown:

```
> tempff2 <- function (x)
      hot.and.cold.fit (x, t = c (-0.2, 0.2),
      hot.hue=100, cold.hue=60)
```