Package 'barsurf'

May 4, 2020

Title Heatmap-Related Plots and Smooth Multiband Color Interpolation

Version 0.5.0

Date 2020-05-04

License GPL (>= 2)

Maintainer Abby Spurdle <spurdle.a@gmail.com>

Author Abby Spurdle

URL https://sites.google.com/site/spurdlea/r

Description Supports combined contour-heat plots and 3D bar/surface plots, for plotting scalar fields, either discretely-spaced or continuously-spaced. Also, supports matrix visualization (per se), isosurfaces (for scalar fields over three variables), triangular plots and vector fields. All plots use static vector graphics (suitable for Sweave documents), but high resolution heatmaps can produce smooth raster-like visual effects. Contains a flexible system for smooth multiband color interpolation in RGB, HSV and HCL color spaces.

Depends methods

Imports kubik, colorspace

Suggests intoo, vectools, misc3d, Matrix

NeedsCompilation no

Repository CRAN

Date/Publication 2020-05-04 05:20:27 UTC

R topics documented:

`01_scalar_fields_discretely-spaced`

Discretely-Spaced Scalar Fields

Description

Plots of functions of two variables (which can be mapped to discretely-spaced scalar fields), and matrix visualization (per se).

Usage

```
#x-y-fv versions
#(main argument is a matrix)
plot_dfield (x, y, fv, fb, ...,
    grid.lines, contours=TRUE, heatmap=TRUE,
    bin.labels=FALSE, contour.labels=FALSE,
    main, xlab="x", ylab="y", xat, yat, xlabs, ylabs,
    xyrel = test.xyrel (x, y, fv), transpose=FALSE,
    add=FALSE, axes=TRUE, reverse=FALSE,
    ncontours=2, clabs, blabs,
    grid.color, contour.color="#000000",
    color.function, color.fit, colors, hcv=FALSE)

plot_bar (x, y, fv, ...,
    main, xlab="x", ylab="y", xat, yat, xlabs, ylabs,
    zlim,
    axes=TRUE, arrows=TRUE,
    color.function, colors)

plot_matrix (x, y, fv, fb, ...,
    grid.lines, contours=FALSE, heatmap=TRUE,
    bin.labels=FALSE, contour.labels=FALSE,
    main, xlab="col", ylab="row", xat, yat, xlabs, ylabs,
    xyrel = test.xyrel (x, y, fv), transpose=TRUE,
    add=FALSE, axes=TRUE, reverse = c (FALSE, transpose),
    ncontours=2, clabs, blabs,
    grid.color, contour.color="#000000",
    color.function, color.fit, colors, hcv=TRUE)
```

```
#functional versions
#(which call the x-y-fv versions above)
plotf_dfield (f, xlim, ylim=xlim, ...)
plotf_bar (f, xlim, ylim=xlim, zlim, ...)
```

Arguments

x, y	Optional sorted numeric vectors of x and y midpoints or breakpoints, refer to details.
fv	A numeric matrix (representing a discretely-spaced scalar field), which may include NAs.
fb	Optional numeric vector of function values (or "levels"), for contours.
grid.lines	Logical, include grid lines. Defaults to true, if is.matrix is true and both the number of rows and columns don't exceed 20.
contours	Logical, include contour lines.
heatmap	Logical, include heatmap.
bin.labels	Logical, include bin labels.
contour.labels	Logical, include contour labels.
main, xlab, ylab	Optional strings, main/axes titles.
xat, yat	Optional numeric vectors, the x and y axes tick points.
xlabs, ylabs	Optional character vectors, the x and y axes tick labels. In 3D plots, ignored unless axes is true and arrows is false.
xyrel	Single character, either "f", "s" or "m". "f" produces a plot with a fixed aspect ratio of one, "s" produces a square plot, and "m" a maximized plot.
transpose	Logical, if true transpose fv. Note that x, y, etc have the same interpretation, regardless of whether the matrix is tranposed or not. (i.e. x values always apply to values on the horizontal axis).
add	Logical, if true add contours/heatmap to an existing plot.
axes	Logical vector of length one or two, if true plot reference arrows or axis ticks with labels.
arrows	Logical vector of length one or two, (subject to axes, above) if true, plot reference arrows, if false, use axis ticks with labels.
reverse	Logical vector of one or two, to reverse the x and y axes. Note that this argument could change.
ncontours	Integer, the number of contour lines. Ignored if fb supplied.
clabs	Optional character vector of contour labels.
blabs	Optional character matrix of bin labels. Defaults to the values of fv.

grid.color	String, giving the grid line color.
contour.color	Character vector of length one or with the same length as the number of contours, giving the contour line colors.
color.function	Optional color function, such as a barface object (for plot_bar) or an opaque litmus object (for the other plots), refer to details.
color.fit	Optional color fitting function, such as a litmus.fit wrapper, refer to details.
colors	Optional character matrix of R colors, for the heatmap bins. For plot_bar, this may also be a list with two matrices, one for tops and one for sides.
hcv	Logical, use the high color variation option.
f	A numeric-valued function of two variables (x and y), suitable for use with base::outer.
xlim, ylim	Length-2 numeric vectors, the x and y ranges. Currently, these need to be ascending.
zlim	Length length-2 numeric vector, the vertical range, corresponding to the value of the function or matrix.
...	In the x-y-fv versions ignored.

Details

These functions produce combined contour-heat plots and 3d bar plots, for discretely-spaced scalar fields.

This can be used to plot numeric matrices (here, labelled as fv), and functions that can be mapped to numeric matrices.

In these plots, each heatmap bin represents one element from fv.

Increasing rows in fv correspond to increasing x values and increasing columns in fv correspond to increasing y values. If an x vector is supplied its length needs to equal the number of rows (exact-match case) or the number of rows plus one (plus-one case). Likewise, if a y vector is supplied its length needs to equal the number of columns or the number of columns plus one. In the exact-match cases, the vectors give the midpoints of bins, and in the plus-one cases, vectors give the breakpoints of bins, including the outermost coordinates.

By default, the plot_matrix function calls the plot_dfield function with fv transposed, such that increasing rows in fv correspond to increasing y values and increasing columns in fv correspond to increasing x values. Also by default, it reverses the y axis, such that increasing y values (increasing rows) go from top to bottom. This means that the resulting plots have the same orientation as standard matrices, in text form.

Currently, bar plots use a diamond-based projection with a fixed viewing angle, such that the origin is at the bottom center.

With the exception of plot_bar, if none of color.function, color.fit and colors are supplied, then:
(1) A color.fit function is determined by global options.
(2) The color.fit function is used to compute a color.function from fv.
(3) The color.function is used to compute a color matrix from fv.

The color function can be any function that maps a numeric vector/matrix to a character vector/matrix of R colors, however, I recommend using litmus objects. The color fit function can

be any function that maps a numeric vector to a valid color function, however, I recommend using a litmus.fit wrapper.

In plot_bar, the color function is a function that maps a logical (true or false) to an R color.

If there are missing values, the bars are omitted.
(So, the plot_bar color function is never called with missing values).

Note that:
(1) In the plot_* functions (not the plotf_* functions) the only argument that's required is fv.
(2) There's no guarantee that default contour lines will be suitable.
(3) Only opaque color functions should be used in heatmaps, that is, color functions that produce opaque colors with no transparency. However, bar plots may use semitransparent colors.

References

Refer to the vignette for an overview, references and better examples.

See Also

Continuously-spaced versions:
plot_cfield

Plots of scalar fields over three variables:
plot_contour_3d, plot_cfield_3d

Other functions:
barface, litmus, litmus.fit

test.xyrel

Examples

```
fv <- matrix (sample (1:100), 10, 10)

plot_dfield (,,fv)
plot_bar (,,fv)
plot_matrix (,,fv)
```

02_scalar_fields_continuously-spaced

Continuously-Spaced Scalar Fields

Description

Plots of functions of two variables (which can be mapped to continuously-spaced scalar fields), for both rectangular and triangular areas.

Usage

```
#x-y-fv versions
#(main argument is a matrix)
plot_cfield (x, y, fv, fb, ...,
    contours=TRUE, heatmap=TRUE, contour.labels=FALSE,
    main, xlab="x", ylab="y", xat, yat, xlabs, ylabs,
    xyrel = test.xyrel (x, y, fv),
    add=FALSE, axes=TRUE, reverse=FALSE,
    ncontours=6, clabs,
    contour.color="#000000",
    color.function, color.fit, hcv=FALSE)

plot_surface (x, y, fv, ...,
    grid.lines=TRUE,
    main, xlab="x", ylab="y", xat, yat, xlabs, ylabs,
    zlim, axes=TRUE, arrows=TRUE,
    grid.color, color.function, color.fit)

plot_tricontour (x, y, fv, fb, ...,
    contours=TRUE, heatmap=TRUE, contour.labels=FALSE,
    main, xlab="x", ylab="y",
    xyrel="s",
    axes=TRUE,
    ncontours=6, clabs,
    contour.color="#000000",
    color.function, color.fit, hcv=FALSE)

plot_trisurface (x, y, fv, ...,
    grid.lines=TRUE,
    main, xlab="x", ylab="y",
    zlim, axes=TRUE, arrows=TRUE,
    grid.color, color.function, color.fit)

#functional versions
#(which call the x-y-fv versions above)
plotf_cfield (f, xlim, ylim=xlim, ..., n=30)
plotf_surface (f, xlim, ylim=xlim, zlim, ..., n=30)
plotf_tricontour (f, ..., n=30)
plotf_trisurface (f, ..., n=30)
```

Arguments

x, y	Optional sorted numeric vectors of x and y coordinates, refer to details. In the triangular case, ignored.
fv	A numeric matrix (representing a continuously-spaced scalar field), which may not include NAs. In the triangular case, fv needs to be a square matrix, but only the top-left triangle is used, including the diagonal.

fb	Optional numeric vector of function values (or "levels"), for contours.
grid.lines	Logical, include grid lines.
contours	Logical, include contour lines.
heatmap	Logical, include heatmap.
contour.labels	Logical, include contour labels.
main, xlab, ylab	Optional strings, main/axes titles.
xat, yat	Optional numeric vectors, the x and y axes tick points.
xlabs, ylabs	Optional character vectors, the x and y axes tick labels. In 3D plots, ignored unless axes is true and arrows is false.
xyrel	Single character, either "f", "s" or "m". "f" produces a plot with a fixed aspect ratio of one, "s" produces a square plot, and "m" a maximized plot.
add	Logical, if true add contours/heatmap to an existing plot.
axes	Logical vector of length one or two, if true plot reference arrows or axis ticks with labels.
arrows	Logical vector of length one or two, (subject to axes, above) if true, plot reference arrows, if false, use axis ticks with labels.
reverse	Logical vector of one or two, to reverse the x and y axes. Note that this argument could change.
ncontours	Integer, the number of contour lines. Ignored if fb supplied.
clabs	Optional character vector of contour labels.
grid.color	String, giving the grid line color.
contour.color	Character vector of length one or with the same length as the number of contours, giving the contour line colors.
color.function	Optional color function, such as an arbitrary litmus object (for plot_surface) or an opaque litmus object (for the other plots), refer to details.
color.fit	Optional color fitting function, such as a litmus.fit wrapper, refer to details.
hcv	Logical, use the high color variation option, refer to details.
f	In the rectangular case, a numeric-valued function of two variables (x and y), suitable for use with base::outer. In the triangular case, a similar function, but of three variables (w1, w2, w3) or (x, y and z) etc, with values between zero and one, which in general, sum to one.
xlim, ylim	Length-2 numeric vectors, the x and y ranges. Currently, these need to be ascending.
zlim	Length-2 numeric vector, the vertical range, corresponding to the value of the function or matrix.
n	An integer vector of length one or two, giving the number of grid points.
...	In the x-y-fv versions, ignored.

Details

These functions produce combined contour-heat plots and 3d surface plots, for rectangular (and triangular) continuously-spaced scalar fields.

This can be used to plot numeric matrices (here, labelled as fv), and functions that can be mapped to numeric matrices.

In contrast to the discretely-spaced case, each heatmap bin represents one area between four (or three) fv points.

Increasing rows in fv correspond to increasing x values and increasing columns in fv correspond to increasing y values. If the x vector is supplied its length needs to equal the number of rows. Likewise, if the y vector is supplied its length needs to equal the number of columns.

Currently, surface plots use a diamond-based projection with a fixed viewing angle, such that the origin is at the bottom center.

If none of color.function and color.fit are supplied, then:
(1) A color.fit function is determined by global options.
(2) The color.fit function is used to compute a color.function from interpolated bin values (in heatmaps) or the magnitudes of their gradients (for surface plots).
(3) The color.function is used to compute a color matrix from the interpolated bin values or magnitudes.

The color function can be any function that maps a numeric vector/matrix to a character vector/matrix of R colors, however, I recommend using litmus objects. The color fit function can be any function that maps a numeric vector to a valid color function, however, I recommend using a litmus.fit wrapper.

Note that:
(1) In the plot_* functions (not the plotf_* functions) the only argument that's required is fv.
(2) There's no guarantee that default contour lines will be suitable.
(3) If using plot_surface or plot_trisurface to plot near-constant values, set zlim, for a better result.
(4) In publication graphics, you may want to increase the resolution of the heatmap.
(5) Only opaque color functions should be used in heatmaps, that is, color functions that produce opaque colors with no transparency. However, surface plots may use semitransparent colors.

Expanding on point (3), functions that are constant in theory, may product non-constant (but near-constant) values when computed via floating point arithmetic.

References

Refer to the vignette for an overview, references and better examples.

See Also

Discretely-spaced versions:
plot_dfield

Plots of scalar fields over three variables:
plot_contour_3d, plot_cfield_3d

Other functions:
litmus, litmus.fit

test.xyrel

Examples

```
x <- y <- seq (-15.5, 15.5, length.out=40)
fv <- outer (x, y, rotated.sinc)

plot_cfield (,,fv)
plot_surface (,,fv)
```

`03_plot_contour_3d` *Contour Plots in 3D*

Description

Contour plots in 3d (for functions of three variables), with one or more isosurfaces.

Usage

```
#x-y-fv version
#(main argument is an array, or a list of arrays)
plot_contour_3d (x, y, z, fv, fb, ...,
    wire.frame=FALSE,
    main, xlab="x", ylab="y", xat, yat, xlabs, ylabs,
    xlim, ylim, zlim,
    axes=TRUE, arrows=TRUE,
    ncontours=2, wire.frame.color="#808080", iso.colors)

#functional version
#(which calls the x-y-fv version above)
plotf_contour_3d (f, xlim, ylim=xlim, zlim=xlim, ...,
    n=20, maximal=FALSE,
    base.contours=FALSE, rear.contours=FALSE,
    pconstants)

#functional version
#(which calls plotf_contour_3d above)
nested_isosurfaces (f, xlim, ylim=xlim, zlim=xlim, ...,
    nfirst=30, nlast=15)
```

Arguments

`x, y, z`	Sorted numeric vectors of x, y and z coordinates, or lists of such vectors. If lists, then their lengths need to equal the number of isosurfaces. (If fv is an array, then x, y and z are optional).
`fv`	A 3-dimensional numeric array, or a list of such arrays. If fv is a list, then its length needs to equal the number of isosurfaces.

`fb`	Numeric vector of function values (or "levels"), for contour/isosurface values. This is optional, however in general, you need to specify this for good plots. Note that if isosurfaces are nested, the first isosurface (for the first fb value) is assumed to be inside the second isosurface (for the second fb value), the second inside the third, and so on. In general, if you need to plot functions with minimal (low-valued) focal points, fb values should be increasing, and if need to plot functions with maximal (high-valued) focal points, fb values should be decreasing.
`wire.frame`	Logical, include the wire.frame lines.
`main, xlab, ylab`	Strings, main/axes titles.
`xat, yat`	Optional numeric vectors, the x and y axes tick points.
`xlabs, ylabs`	Optional character vectors, the x and y axes tick labels. In 3D plots, ignored unless axes is true and arrows is false.
`xlim, ylim`	Length-2 numeric vectors, the x and y ranges. Note that the package has only been tested with ascending values.
`zlim`	Length-2 numeric vector, the z (vertical) range.
`axes`	Logical vector of length one or two, if true plot reference arrows or axis ticks with labels.
`arrows`	Logical vector of length one or two, (subject to axes, above) if true, plot reference arrows, if false, use axis ticks with labels.
`ncontours`	Integer, the number of contours/isosurfaces, ignored if fb supplied.
`wire.frame.color`	Optional string, giving the wire frame color.
`iso.colors`	Character vector (with the same length as the number of isosurfaces), giving the (initial) isosurface colors. Optional for up to three isosurfaces.
`f`	A numeric-valued function of three variables (x, y, and z).
`n`	An integer vector of length one or three (giving the number of grid points), or list of such vectors (giving the number of grid points for each isosurface). If a list, its length needs to equal the number of isosurfaces.
`nfirst, nlast`	Integer vectors of length one or three (giving the number of grid points), for the first and last isosurfaces.
`maximal`	Logical, if true assume maximal (high-valued) focal points, that is, any focal points within the plot have higher function/array values than their surrounding values. This determines whether the default fb values are ascending or descending. And is ignored if the fb vector is supplied.
`base.contours`	Logical, include contour lines on the base panel of the plot.
`rear.contours`	Logical, include contour lines on the rear panels of the plot.
`pconstants`	Optional length-3 numeric vector of panel constants. Defaults to the midpoints of xlim, ylim and zlim. Refer to details and examples.
`...`	In the x-y-fv version, ignored.

Details

These functions require the misc3d package to be installed, and be on the search path.

These functions are similar to plot_cfield and plot_surface.
(So, please refer to those functions for background information).

In plot_contours_3d, fv is a 3-dimensional array, or a list of such arrays.

x, y and z can be vectors, or lists of vectors.

Increasing along the first array dimension corresponds to increasing x values. Increasing along the second array dimension corresponds to increasing y values. And increasing along the third array dimension corresponds to increasing z values.

x and y (the first two variables) have the same interpretation as they do in plot_surface. z (the third variable) gives the vertical position. Except that x, y and z describe coordinates of the fv array, not the resulting isosurfaces.

If x, y, z, fv or n are lists, then their lengths need to equal the number of isosurfaces, and each list element applies to one isosurface.

Unlike other plots in this package, the isosurfaces don't use color interpolation, however, a small amount of random color variation is added.

The plotf_contour_3d function calls plot_contour_3d, but computes the x, y, z and fv values.

Optionally, 2D contour lines may added to the base panel and rear panels.

If necessary, a vector of panel constants (pconstants) is used.

A matrix is computed by evaluating the function, f, while holding the third variable constant (at the third panel constant). This matrix is used to compute the base contours. Likewise, the first variable is held constant (at the first panel constant) for the right rear panel, and the second variable is held constant (at the second panel constant) for the left rear panel.

The nested_isosurfaces function calls the plotf_contour_3d function, but computes the grid size for each isosurface.

Unlike the other two functions, it needs at least two isosurfaces, and assumes that the first isosurface(s) is/are inside the second, and that the second isosurface(s) is/are inside the third, and so on.

By default, the resolution of fv is highest for the first isosurface, and each fv array has a progressively lower resolution.

Resulting plots should result in smaller file sizes, and render more quickly.

References

Refer to the vignette for an overview, references and better examples.

See Also

Plots of scalar fields over two variables:
plot_dfield, plot_cfield

Other plots of scalar fields over three variables:
plot_cfield_3d

Other functions:
litmus, litmus.fit

Examples

```
library (misc3d)

plotf_contour_3d (bispherical.dist, c (-3, 3),, c (-2, 2), fb = c (0.5, 1, 1.75),
    base.contours=TRUE)

#panel contours condition on x=1, y=1, z=0
nested_isosurfaces (bispherical.dist, c (-3, 3),, c (-2, 2), fb = c (0.5, 1, 1.75),
    base.contours=TRUE, rear.contours=TRUE,
    arrows=FALSE,
    pconstants = c (1, 1, 0) )
```

04_plot_cfield_3d *Contour-Heat Plots, 3D-Based*

Description

Plots of continuously-spaced scalar fields, of three variables.

Usage

```
#x-y-fv version
#(main argument is a list of matrices)
plot_cfield_3d (x, y, z, fv, fb, ...,
    contours=TRUE, heatmap=TRUE,
    main, xlab="x", ylab="y",
    axes=TRUE, reverse.z=FALSE,
    ncontours=6, emph="n", color.function, color.fit)

#functional version
#(which calls the x-y-fv version, above)
plotf_cfield_3d (f, xlim, ylim=xlim, zlim=xlim, ..., nslides=6, n=30, z)
```

Arguments

x, y	Optional sorted numeric vectors of x and y coordinates, refer to details.
z	Optional numeric vector of two or more z coordinates, refer to details.
fv	A list of two or more numeric matrices (representing a continuously-spaced scalar fields), one for each z value.
fb	Optional numeric vector of function values (or "levels"), for contours.
contours	Logical, include contour lines.
heatmap	Logical, include heatmap.
main, xlab, ylab	Strings, main/axes titles.
axes	Logical vector of length one or two, plot reference arrows.

reverse.z	Logical, if true reverse the z axis.
ncontours	Integer, the number of contour lines. Ignored if fb supplied.
emph	What to emphasize, either "n" (for nothing), "b" for both (low and high regions), "l" for low regions or "h" for high regions, or "B", "L" or "H" for the same but with a stronger effect.
color.function	Optional color function, such as a litmus object, refer to details.
color.fit	Optional color fitting function, such as a litmus.fit wrapper, refer to details.
f	A numeric-valued function of three variables (x, y, and z), suitable for use with base::outer, called with four arguments, x, y, f, and z [i].
xlim, ylim	Length-2 numeric vectors, the x and y ranges. Currently, these need to be ascending.
zlim	Length-2 numeric vector, the z (vertical) range, ignored is z supplied.
nslides	Integer, giving the number of slides, ignored if z supplied.
n	An integer vector of length one or two, giving the number of grid points.
...	In the x-y-fv version, ignored.

Details

These functions are similar to plot_cfield, plot_contour_3d, plot_surface and plotf_contour_3d. (So, please refer to those functions for background information).

These functions produce 3d-based combined contour-heat plots.

Plots contain a set of of two or more 2d slides (or slices), which can be used to plot functions of three variables, x, y and z.

In plot_cfield_3d, fv is a list of matrices, rather than a single matrix.

Each matrix represents one slide.

In general, six or seven slides produces a good result. More slides may be used, but with care.

x and y (the first two variables) have the same interpretation as they do in plot_surface. z (the third variable) gives the vertical position of the slides.

Note that:
(1) In plot_cfield_3d the only argument that is required is fv.
(2) In some cases, the plotting device/window should be opened before calling these functions, and the plot should be higher than it is wide.
(3) There's no guarantee that default contour lines will be suitable.
(4) Unlike other heatmaps in this package, these functions may use semitransparent colors.

Expanding on point (4), some PDF viewers may produce visual artifacts, such as grid lines.

In general, these artifacts are minimal.

If you want to remove them, possible options are to change your PDF viewer's settings or save the plot in a raster format, such as PNG.

References

Refer to the vignette for an overview, references and better examples.

See Also

Plots of scalar fields over two variables:
plot_dfield, plot_cfield

Other plots of scalar fields over three variables:
plot_contour_3d

Other functions:
litmus, litmus.fit

Examples

```
plotf_cfield_3d (bispherical.dist, c (-3, 3),, c (-2, 2), emph="l")
```

`05_vector_fields` *Vector Fields*

Description

Plots of vector fields.

Usage

```
#x-y-fx-fy version
#(main arguments are matrices)
plot_vecfield (x, y, fx, fy, ...,
    vectors=TRUE, heatmap=TRUE, all=FALSE,
    main, xlab="x", ylab="y", xat, yat, xlabs, ylabs,
    xyrel = test.xyrel (x, y, fv),
    add=FALSE, axes=TRUE, reverse=FALSE,
    arrowh.length=1.75, arrowh.width = 0.75 * arrowh.length,
    arrow.color="#000000", fill.color="#08080810",
    color.function, color.fit, hcv=FALSE)

#functional version
#(which calls the x-y-fx-fy version above)
plotf_vecfield (f, xlim, ylim=xlim, ..., nv=20, nh=40)
```

Arguments

`x, y`	Optional sorted numeric vectors of x and y coordinates, refer to details.
`fx, fy`	Numeric matrices (representing x and y components of a vector field), which may include NAs.
`vectors`	Logical, include vector arrows.
`heatmap`	Logical, include heatmap.
`all`	Logical, plot all vector arrows. Otherwise (the default), exclude the outermost rows and columns.

main, xlab, ylab	Optional strings, main/axes titles.
xat, yat	Optional numeric vectors, the x and y axes tick points.
xlabs, ylabs	Optional character vectors, the x and y axes tick labels.
xyrel	Single character, either "f", "s" or "m". "f" produces a plot with a fixed aspect ratio of one, "s" produces a square plot, and "m" a maximized plot.
add	Logical, if true add contours/heatmap to an existing plot.
axes	Logical vector of length one or two, if true plot axis ticks with labels.
reverse	Logical vector of one or two, to reverse the x and y axes. Note that this argument could change.
arrowh.length	Arrow head length, in mm.
arrowh.width	Arrow head width, in mm.
arrow.color	Arrow line color.
fill.color	Arrow fill color.
color.function	Optional color function, such as an opaque litmus object, refer to details.
color.fit	Optional color fitting function, such as a litmus.fit wrapper, refer to details.
hcv	Logical, use the high color variation option, refer to details.
f	A function of two variables (x and y), which returns a list of two numeric vectors, one value for the x component and one for the y component.
xlim, ylim	Length-two numeric vectors, the x and y ranges. Currently, these need to be ascending.
nv, nh	Integer vectors of length one or two, giving the number of grid points, where nv is for the vector arrows and nh is for the heatmap.
...	In the x-y-fx-fy version, ignored.

Details

Refer to plot_cfield for background information.

The plot_vecfield function is the same, except that:
(1) There are vector arrows rather than contour lines.
(2) fv is replaced by two matrices, for the x and y components.
(3) The color of the heatmap is determined by the magnitude of the vectors, similar to plot_surface.
(4) Missing values are allowed.

The plotf_vecfield function calls plot_vecfield twice, and by default, uses a higher resolution for the heatmap than the arrows.

If you want to plot a vector field with a subset of vector arrows then I recommend you plot the vector field with vectors=FALSE, then plot another vector field on top of it with add=TRUE, heatmap=FALSE, and set parts of fx or fy to NA. A similar approach can be used to plot vector fields with arrows that have different sizes or colors.

Note that:
(1) In the plot_vecfield the only arguments that are required are fx and fy.
(2) In publication graphics, you may want to increase the resolution of the heatmap.
(3) Only opaque color functions should be used, that is, color functions that produce opaque colors with no transparency.

References

Refer to the vignette for an overview, references and better examples.

See Also

plot_cfield

litmus, litmus.fit

test.xyrel

Examples

```
plotf_vecfield (concentric.field, c (-1.5, 1.5), c (-1.5, 1.5) )
```

06_color_themes *Color Themes*

Description

Set global options and color themes.

Usage

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

Arguments

theme	String, either "heat", "gold", "blue", "green" or "purple".
rendering.style	Single character, either "r", "p" or "e", refer to details.
...	Ignored.

Details

The set.bs.options and set.bs.theme functions set global options which determine line width and default colors.

The heat theme is designed for high impact, whereas the blue and green themes are designed for (higher) perceptual uniformity.

Currently, parts of the heat, gold and purple themes are the same as the blue theme.

By default, with the "r" rendering style (for standard R graphic devices), all lines have a line width of one and are black, with the exception of grid lines, isosurface lines and some contour lines. The "p" rendering style (for black and white printed documents) is the same as "r", except that all lines are black, with the exception of some contour lines. The "e" rendering style (for PDF and other documents, intended to be viewed electronically) is the same as "r", except that vector arrows, isosurfaces and grids use narrower lines.

Examples

```
set.bs.theme ("heat")
```

`07_barface_objects` *Barface Objects*

Description

Color functions for 3d bar plots.

Usage

```
barface (coltv, colfv, ..., color.space="sRGB")

heat.barface ()
gold.barface ()
blue.barface ()
green.barface ()
purple.barface ()
```

Arguments

`coltv, colfv`	Length-3 or length-4 numeric vectors, representing colors.
`color.space`	String, a color space, refer to the details section for litmus objects.
`...`	Ignored.

Details

A barface object is a function that maps a logical vector to a character vector, representing R colors. (Noting that these functions return barface objects, so you call these functions, and then if necessary you can evaluate the resulting function).

The coltv defines a color, for true, which plot_bar uses for the tops of bars. The colfv defines a color, for false, which plot_bar uses for the sides of bars. It's possible to omit the colfv argument, in which case, the resulting color will be similar to the top color, but lighter.

The other functions (e.g. heat.barface) are wrappers, that create barface objects with particular colors.

Value

All these functions return barface objects.

Refer to details.

References

Refer to the vignette for an overview, references and better examples.

See Also

litmus

Examples

```
colf <- barface (c (1, 0, 0), c (0, 0, 1) )
plot (colf)

#evaluate
colf (c (TRUE, FALSE) )
```

08_litmus_objects *Litmus Objects*

Description

Color functions for heatmaps and surface plots.

Usage

```
########################################
#general cases
########################################
#equally spaced knots
litmus (a=0, b=1, colvs, ...,
    color.space="sRGB", na.color="#FFFFFF")

#arbitrary knots
litmus.spline (cx, colvs, ...,
    color.space="sRGB", na.color="#FFFFFF")

########################################
#wrappers, with predefined colors
########################################
#opaque
#high impact
heat.litmus (a=0, b=1, ..., reverse=FALSE)

#opaque
#interpolate over hue
blue.litmus (a=0, b=1, ..., reverse=FALSE)
green.litmus (a=0, b=1, ..., reverse=FALSE)

#opaque
#high color variation
#dark-color -> light-color -> white
blue.litmus.hcv (a=0, b=1, ..., reverse=FALSE)
```

```
green.litmus.hcv (a=0, b=1, ..., reverse=FALSE)

#opaque
#interpolate over lum
blue.litmus.flow (a=0, b=1, ..., reverse=FALSE)
green.litmus.flow (a=0, b=1, ..., reverse=FALSE)

#opaque
#adapted from colorspace::rainbow_hcl
rainbow.litmus (a=0, b=1, ..., c=42.5, l=75, start=65, end=315)
rainbow.litmus.2 (a=0, b=1, ..., c=50, l=70, start=0, end=360)

#semi-transparent
glass.rainbow (a=0, b=1, alpha=0.3, ..., c=42.5, l=62.5, start=42.5, end=260)
```

Arguments

a	Numeric, the lower limit (first knot).
b	Numeric, the upper limit (last knot).
cx	A numeric vector of knots (including the outermost values), which should be unique and ascending.
colvs	A 3-column or 4-column numeric matrix, where each row is one color vector, and the optional fourth column is alpha values.
color.space	A string giving the color space, refer to details.
na.color	A single string representing an R color.
reverse	Logical, reverse the order of the colors.
c, l, start, end	Same as colorspace::rainbow_hcl.
alpha	A numeric vector giving the alpha component. If it has two or more values, then each alpha value is assigned to each color.
...	Ignored.

Details

A litmus object maps a numeric vector to a character vector, representing R colors.
(Noting that these functions return litmus objects, so you call these functions, and then if necessary you can evaluate the resulting function).

Color vectors are mapped from the input color space into sRGB color space.

And a set of cubic Hermite splines interpolates over each component.

Input color spaces include "XYZ", "RGB", "LAB", "polarLAB", "HSV", "HLS", "LUV" and "polarLUV" (from the colorspace package), in addition to "HCL" (which is the same as polarLUV, except that the color components are in the reverse order.). Input color vectors may have an alpha component, in which case, the mapping preserves it, such that the resulting sRGB colors will have the same alpha values.

There are two constructors, one for equally spaced knots and one for arbitrary knots.

The other functions (e.g. heat.litmus) are wrappers, that create litmus objects with particular colors.

Note that in theory, the interpolation is smooth, however, a smooth appearance (or a not so smooth appearance) is dependent on the choice of knots and colors. Similar consecutive colors tend to produce smoother looking results. In general, interpolating over hue, with constant chroma and luminescence produces the smoothest results, however, there are many situations where it's desirable for one area (within a plot) to appear brighter than others.

Note that if a litmus object is evaluated with x values outside the knots, then the function will return the first or last color.

Also note that it may be easier to construct litmus objects using the litmus.fit function, or one of it's wrapper functions.

Value

All functions return litmus objects.

Refer to details.

References

Refer to the vignette for an overview, references and better examples.

See Also

mlitmus, litmus.fit

Examples

```
#bad example
colvs <- matrix (c (
    0, 0, 0,
    0.75, 0, 0.25,
    0.25, 0, 0.75,
    1, 1, 1),, 3, byrow=TRUE)
colf <- litmus (,,colvs)
plot (colf)

#better example
rainbow.litmus.3 <- function (a=0, b=1)
{   colvs <- cbind (c (110, 170, 230, 290), 42.5, 75)
    litmus (a, b, colvs, color.space="HCL")
}
colf <- rainbow.litmus.3 ()
plot (colf)

#evaluate
colf (c (0, 0.33, 0.67, 1) )
```

09_multi-litmus_objects
Multi-Litmus Objects

Description

Color functions that combine multiple litmus objects.

Usage

```
mlitmus (..., default.color="#D0D0D0", na.color=default.color)

hot.and.cold (a=-1, b=1, xb=0)
```

Arguments

...	One or more litmus objects.
default.color	String, color that is returned for x values outside the litmus objects' knots.
na.color	String, color that is returned for NA x values.
a	Numeric, the lower limit (first knot).
b	Numeric, the upper limit (last knot).
xb	Numeric, giving the breakpoint between "hot" and cold".

Details

This function creates a color function similar to a litmus object, containing one or more litmus objects.
(Noting that these functions return mlitmus objects, so you call these functions, and then if necessary you can evaluate the resulting function).

The color function works out which litmus object to use for each input value.

An example is using one set of colors for positive values and another set of colors for negative values.

Note that it's possible for litmus objects to overlap.
This may be changed, so this feature should not be used inside packages.

Value

An mlitmus object.

References

Refer to the vignette for an overview, references and better examples.

See Also

litmus

Examples

```
colf <- mlitmus (blue.litmus (-1, 0, reverse=TRUE), green.litmus (0, 1) )

#evaluate
colf (c (-1, 0, 1) )
```

`10_supporting_methods` *Supporting Methods*

Description

Print and plot methods for barface, litmus and mlitmus objects.

Usage

```
## S3 method for class 'barface'
print(x, ...)
## S3 method for class 'litmus'
print(x, ...)
## S3 method for class 'mlitmus'
print(x, ...)

## S3 method for class 'barface'
plot(x, ...)
## S3 method for class 'litmus'
plot(x, n=200, ...)
## S3 method for class 'mlitmus'
plot(x, n=200, ...)
```

Arguments

`x`	A barface, litmus or mlitmus object.
`n`	Integer, number of strips.
`...`	Ignored.

Details

The plot method calls colorspace::swatchplot, with a vector of colors.

Examples

```
plot (heat.barface () )
```

11_other_supporting_functions_1
Test Axis Relationship

Description

Estimate the (plotting) relationship between the x and y axes.

Usage

```
test.xyrel (x, y, fv)
```

Arguments

x, y	Optional numeric vectors.
fv	A numeric matrix.

Details

This function is designed to work with the xyrel argument in the main plotting functions.

It returns "f" (for a fixed aspect ratio of one) or "m" (for maximized).

If the ratio between the x-size and the y-size is between 0.1 and 10, it will return "f".
Otherwise, it returns "m".

If both x any y are missing, the "size" refers to the dimensions of the matrix.
(So, returns "f" for up to ten times more columns than rows, or vice versa).

If both x and y are supplied, the "size" refers to xlim and ylim, computed from the ranges of x and y.

If one is supplied but the other is not, then "m" is returned.

Value

Refer to details.

Examples

```
fv <- matrix (1:20, 2, 10)
test.xyrel (,,fv)
```

12_other_supporting_functions_2

Matrix Margins

Description

Reverse the bottom and top margins.

Usage

```
matrix.margins ()
```

Details

The function changes the par settings for the margins, for subsequent plots.

It reverses the bottom and top margins, based on the assumption that the user wants the x-axis ticks/labels on the top, and a possible main title on the bottom.

Value

A named list giving the original par settings for the margins.

Examples

```
fv <- matrix (sample (1:24), 4, 6)

p0 <- matrix.margins ()
plot_matrix (,,fv)
par (p0)
```

13_other_supporting_functions_3

Utility Functions for Color Conversion

Description

Convert a color vector from sRGB space to HSV/HCL space, or vice versa.

Usage

```
rgb2hsv (colv)
hsv2rgb (colv)

rgb2hcl (colv)
hcl2rgb (colv, correction=FALSE)
```

Arguments

colv	A length-3 numeric vector of sRGB, HSV or HCL values.
correction	Logical, correct the sRGB values, if they're outside the interval [0, 1].

Details

These functions are wrappers for functions in the colorspace package.

Note that these functions support single length-3 vectors only, however, other functions in this package support length-4 vectors with an alpha component.

Value

A length-3 vector.

Examples

```
rgb2hcl (c (0, 0, 1) )
```

14_sample_functions *Sample Functions*

Description

Functions to produce sample scalar and vector fields.

Usage

```
#scalar-valued (theoretically)
rotated.sinc (x, y)
bispherical.dist (x, y, z)

#vector-valued (theoretically)
concentric.field (x, y)
```

Arguments

x, y, z	Numeric vectors, where the functions are evaluated.

Details

The rotated.sinc function was adapted from the graphics::persp examples.

The bispherical.dist function gives the smaller of distances from two points at:
(-1, 1, 0)
(1, -1, 0)

The concentric.field functions generates a vector field with circular flow, and highest magnitude at r=1, where r is the distance from the origin.

Value

The (theoretically) scalar-valued functions return a numeric vector.

The (theoretically) vector-valued functions return a list containing two numeric vectors.

Examples

```
rotated.sinc (0, 0)
bispherical.dist (0, 0, 0)
concentric.field (c (0, 1), 0)
```

15_fitting_litmus_objects_to_data

Fit Litmus Objects to Data

Description

Functions to fit litmus objects to vectors of data.

Usage

```
########################################
#general cases
########################################
litmus.fit (x, colvs, ...,
    color.space="sRGB", reverse=FALSE, equalize=0.85, na.color="#FFFFFF")

########################################
#wrappers, with predefined colors
########################################
#opaque
#high impact
heat.litmus.fit (x, ..., reverse=FALSE, equalize=0.85)
hot.and.cold.fit (x, xb=0)

#opaque
#interpolate over hue
blue.litmus.fit (x, ..., reverse=FALSE, equalize=0.85)
green.litmus.fit (x, ..., reverse=FALSE, equalize=0.85)

#opaque
#dark-color -> light-color -> white
blue.litmus.fit.hcv (x, ..., reverse=FALSE, equalize=0.85)
green.litmus.fit.hcv (x, ..., reverse=FALSE, equalize=0.85)

#opaque
#interpolate over chroma and lum
```

```
blue.litmus.fit.flow (x, ..., reverse=FALSE, equalize=0.85)
green.litmus.fit.flow (x, ..., reverse=FALSE, equalize=0.85)

#semitransparent
#interpolate over chroma and lum
heat.litmus.fit.lum (x, ..., reverse=FALSE, equalize=0.85)
gold.litmus.fit.lum (x, ..., reverse=FALSE, equalize=0.85)
blue.litmus.fit.lum (x, ..., reverse=FALSE, equalize=0.85)
green.litmus.fit.lum (x, ..., reverse=FALSE, equalize=0.85)
purple.litmus.fit.lum (x, ..., reverse=FALSE, equalize=0.85)

#opaque
#adapted from colorspace::rainbow_hcl
rainbow.litmus.fit (x, ..., c=42.5, l=75, start=65, end=315, equalize=0.85)
rainbow.litmus.fit.2 (x, ..., c=50, l=70, start=0, end=360, equalize=0.85)

#semitransparent
glass.rainbow.fit (x, alpha=0.3, ..., c=42.5, l=62.5, start=42.5, end=260,
    equalize=0.85)
```

Arguments

x	A numeric vector.
xb	Numeric, giving the breakpoint between "hot" and cold".
colvs	A 3-column or 4-column numeric matrix, where each row is one color vector, and the optional fourth column is alpha values.
color.space	A string giving the color space, refer to the details section for litmus objects.
reverse	Logical, reverse the order of the colors.
equalize	Numeric, between zero and one, refer to details.
na.color	A single string representing an R color.
c, l, start, end	Same as colorspace::rainbow_hcl.
alpha	A numeric vector giving the alpha component. If it has two or more values, then each alpha value is assigned to each knot.
...	Ignored.

Details

Refer to the litmus function for background information.

The litmus.fit function constructs a litmus object.

Given n colors, it computes a length-n vector of knots computed from a vector of data.

If equalize is zero, the knots are equally spaced from the lowest x value to the highest. If equalize is one, then knots are selected, such that there's an approximately equal number of points between each pair of knots. And equalization values between zero and one result in an intermediate effect.

Note that high equalize values (higher than the default) may cause color interpolation to appear less smooth.

In general, it's easiest to wrap the litmus.fit function inside another function, which defines the color space and the colors.

This package defines a range of wrapper functions, for the heat, blue, green and purple color themes.

Value

All functions return litmus objects, except the hot.and.cold function which returns a mlitmus object.

References

Refer to the vignette for an overview, references and better examples.

See Also

litmus

Examples

```
rainbow.litmus.fit.3 <- function (x)
{   colvs <- cbind (c (110, 170, 230, 290), 42.5, 75)
    litmus.fit (x, colvs, color.space="HCL")
}

x <- rnorm (200)
colf <- rainbow.litmus.fit.3 (x)

#evaluate
colf (min (x) )
colf (mean (x) )
colf (max (x) )
```

16_deprecated *Deprecated Functions*

Description

Deprecated functions, please do not use.

Usage

```
use.theme (...)
plot2d.contour (...)
plot3d.bar (...)
plot3d.surface (...)
litmus.rainbow.fit (...)
```

Arguments

. . . .

Index