Package 'HilbertVisGUI'

May 15, 2024

Version 1.63.0

Date 2013-10-08

Title HilbertVisGUI

Author Simon Anders <sanders@fs.tum.de>

Maintainer Simon Anders <sanders@fs.tum.de>

Depends R (>= 2.6.0), HilbertVis (>= 1.1.6)

SystemRequirements gtkmm-2.4, GNU make

Suggests lattice, IRanges

Description An interactive tool to visualize long vectors of
integer data by means of Hilbert curves

License GPL (>= 3)

URL http://www.ebi.ac.uk/~anders/hilbert

biocViews Visualization

git_url https://git.bioconductor.org/packages/HilbertVisGUI

git_branch devel

git_last_commit 817bff0

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-05-15

Contents

---

dotsapply   *List apply for '...' arguments*

Description

A kludge to get around a certain problem in using `lapply` with '...' ellipsis function arguments.

Usage

```
dotsapply( fun, ... )
```

Arguments

fun   A function that takes one argument

...   Arguments to which the function should be applied

Details

"dotsapply(fun,...)" gives the same result as `lapply( list(...), fun)`. However, the construction with "`list`", when used for variables enumerated explicitely, will result in their duplication.

Assume, for example, that you have three very large vectors a, b, and c, whose lengths you wish to know. If your write "`lapply( list(a,b,c),length )`", R will duplicate all three vectors in memory when constructing the list, which results in unnecessary use of memory. The alternative "`dotsapply( length, a, b, c )`" avoids this. Of course, you could also write "`list( length(a), length(b), length(c) )`", which neither causes duplication.

This last possibility is, however, not an option, if, instead of "`a,b,c`", you have ellipsed function arguments, i.e. literally "`...`". In this special case, `dotsapply` comes in handy, and as this case arose in [hilbertDisplay](), I implemented `dotsapply` as a kludge and export it from the package just in case somebody ahs use for it.

Value

A list of function values

Author(s)

Simon Anders, EMBL-EBI, `<sanders@fs.tum.de>`

---

hilbertDisplay *Hilbert curve visualization*

Description

Display one or several long integer vectors in an interactive fashion by means of the Hilbert curve.
See the vignette for a full explanation.

Usage

```
hilbertDisplay(
      ...,
      palettePos = colorRampPalette( c( "white", "red" ) )( 300 ),
    paletteNeg = colorRampPalette( c( "white", "blue" ) )( length(palettePos) ),
      maxPaletteValue = NULL,
      naColor = "gray",
      plotFunc = simpleLinPlot,
      names=NULL,
      sameScale=TRUE,
      pow2=FALSE,
      portrait=TRUE,
      fullLengths = NULL )
```

Arguments

...	The data vectors to be visualized. This may be integer or real vectors, either as ordinary R vectors or as `Rle` vectors (as defined in the `IRanges` packages). Care is taken within `hilbertDisplay` that these vectors do not get duplicated, so that you can pass very large vectors.
palettePos	The color palette used to visualize positive valued bins in the data. This must be either be a character vector of valid color names. If no palette is supplied, a ramp from white to red is used.
paletteNeg	The color palette for bins with negative values. Both palettes must have the same number of colours.
maxPaletteValue	
	The absolute bin value that should correspond to the end of the two palettes. (The beginning of the palettes always represents 0.) Within the graphical user interface, buttons are provided to adjust this interactively. If no value is provided, the median of the absolute values of maxima and minima of the data vectors is used.
naColor	The color to be used for bins which contain NAs or correspond to data outside the limits of the data vector. Pass a color name or a triple of RGB values. By default, "gray" is used.
plotFunc	An R function that is called if you use the "Linear plot" function offer by HilbertCurveDisplay's GUI. If you enable this function and then click on a pixel

in the display, the function supplied as plotFunc is called. If you do not supply
this parameter, the function [simpleLinPlot](part of this package) is used. If
you supply your own function, it must accept two parameters: data and info.
data will be the currently displayed data vector. Be careful that your function
does not duplicate it (check with [tracemem](#), if in doubt) in order to avoid per-
formance problems when dealing with large data. The second argument, info,
is a list, supplying the following fields, all of which, except for the last one, are
single integers: info$binLo, info$bin, and info$binHi are the lower, mid-
dle, and upper coordinate (i.e., vector index) of the bin represented by the pixel
onto which the user has clicked. info$dispLo and info$dispHi are the lowest
and highest index of the part of the vector currently displayed. info$seqIdx
is the index of the currently displayed vector (i.e., its position in the '...' argu-
ment) and ionfo$seqName is its name. All indices are one-based. Your function
should plot a region of interest around data[info$bin], or do some other use-
ful operation. Any return value is ignored. For a very simple example, see the
body of [simpleLinPlot](#).

names   The names of the sequences. A character vector. If not supplied, the expressions
               used in the '...' argument, as reported by [substitute](#) are used.

sameScale   Setting this argument to TRUE pads all but the largest vector with NAs such that
               all vectors have the same length. (The padding is done "virtually", i.e. no du-
               pliocation in memory occurs.) The purpose of this is to make sure that the bin
               size (i.e. the number of values depicted by one pixel) stays constant, when using
               the DisplayHilbertViewer GUI's "Prev" and "Next" buttons, which switch the
               display through the supplied data vectors.

pow2   Setting thsi argument to TRUE pads all vectors virtually with NAs such that their
               length becomes a power of 2. The purpose of this becomes apparent if you
               zoom in so much that several pixels correspond to the same data vector element.
               Then, without this options, the values take on strange fractal forms, while they
               are square in case of a power-of-2 length.

portrait   Setting this option to FALSE changes the GUI layout such that the controls appear
               to the right of the curve display ("landscape layout") as opposed to the usual case
               of them appearing below ("portrait layout"). This is useful for small screens as
               the GUI window may be two tall to fit on the screen in portrait mode.

fullLengths   This option allows you to manually control the padding of vectors with NAs if
               you do not like the result of the same.scale or pow2 option. Supply an integer
               vector with as many values as their are vectors in the '...' argument, specifying
               the length including padding for each data vector. Passing numbers smaller than
               the length of the data.vector results in only the beginning of the vector being
               displayed.

Value

Returns an invisible NULL.

Author(s)

Simon Anders, EMBL-EBI, <sanders@fs.tum.de>

See Also

simpleLinPlot

Examples

```
random <- c( as.integer( runif(100000)*30 ) )
ramp <- c( as.integer( 0:19999/100 ) )
try( hilbertDisplay( random, ramp ) )
```

---

hilbertDisplayThreeChannel

Show up to three data vectors in the Hilbert curve display GUI.

Description

This is a variant of hilbertDisplay that takes up to three long data vector but does not display them in different panels (allowing to switch with the "Previous" and "Next" buttons but instead put them in one picture, using the three colour channels to overlay them. This is look for correlations in spatial distribution of the different data vectors.

Usage

```
hilbertDisplayThreeChannel(
    dataRed,
    dataGreen = 0,
    dataBlue = 0,
    naColor = col2rgb("gray"),
    fullLength = max(length(dataRed), length(dataGreen), length(dataBlue)),
    portrait = FALSE)
```

Arguments

dataRed   The data to be displayed in the red colour channel, to be provided as a numeric
               vector with values between 0 (black) and 1 (bright red). You may either pass an
               ordinary R vector or an `Rle` vector as defined in the `IRanges` package.

dataGreen   As `dataRed` but for the green channel.

dataBlue   As `dataRed` but for the blue channel.

naColor   The color to use to indicate NA.

fullLength   The vector length that should correspond to the full display. By default, the
               length of the longest data vector. It may make sense to round up to the next
               power of two.

portrait   Whether to arrange the buttons to the right of the display. Useful on small
               screens.

Value

Invisible NULL.

Note

This function is a bit less mature than the function [hilbertDisplay](). Especially, the GUI functions "Linear plot", "Previous", and "Next" are displayed but defunct.

Author(s)

Simon Anders

See Also

[hilbertDisplay]()

---

simpleLinPlot	*Default plot callback function for hilbertDisplay*

Description

Plots a small part of the supplied vector.

Usage

```
simpleLinPlot( data, info )
```

Arguments

data	The data vector, a vector of integers.
info	A list with fields describing the context, as described in the help page for [hilbertDisplay]().

Details

This is a very simple function that is called by default by [hilbertDisplay]() when the user chooses the "Linear plot" option and clicks on a pixel in the display. It displays a piece of 2000 values of the data vector, centered around info$bin. You will often want to replace this function by one tailored to your needs.

Value

Null.

Author(s)

Simon Anders, EMBL-EBI, <sanders@fs.tum.de>

See Also

[hilbertDisplay](#)

Index