

# Package ‘HilbertCurve’

June 2, 2022

**Type** Package

**Title** Making 2D Hilbert Curve

**Version** 1.26.0

**Date** 2022-04-12

**Author** Zuguang Gu

**Depends** R (>= 3.6.0), grid

**Imports** methods, utils, HilbertVis, png, grDevices, circlize (>= 0.3.3), IRanges, GenomicRanges, polylabelr

**Suggests** knitr, testthat (>= 1.0.0), ComplexHeatmap (>= 1.99.0), markdown, RColorBrewer, RCurl, GetoptLong, rmarkdown

**VignetteBuilder** knitr

**Description** Hilbert curve is a type of space-filling curves that fold one dimensional axis into a two dimensional space, but with still preserves the locality. This package aims to provide an easy and flexible way to visualize data through Hilbert curve.

**biocViews** Software, Visualization, Sequencing, Coverage, GenomeAnnotation

**URL** <https://github.com/jokergoo/HilbertCurve>

**License** MIT + file LICENSE

**Collate** 00\_S4\_generic\_methods.R HilbertCurve.R hc\_polygon.R utils.R GenomicHilbertCurve.R zzz.R

**git\_url** <https://git.bioconductor.org/packages/HilbertCurve>

**git\_branch** RELEASE\_3\_15

**git\_last\_commit** 0b7de6e

**git\_last\_commit\_date** 2022-04-26

**Date/Publication** 2022-06-02

**Maintainer** Zuguang Gu <z.gu@dkfz.de>

**R topics documented:**

default_overlay . . . . .	2
GenomicHilbertCurve . . . . .	3
GenomicHilbertCurve-class . . . . .	5
hc_centered_text-HilbertCurve-method . . . . .	6
hc_layer-dispatch . . . . .	7
hc_layer-GenomicHilbertCurve-method . . . . .	7
hc_layer-HilbertCurve-method . . . . .	8
hc_level-HilbertCurve-method . . . . .	10
hc_map-GenomicHilbertCurve-method . . . . .	11
hc_normal_points-HilbertCurve-method . . . . .	12
hc_offset-HilbertCurve-method . . . . .	14
hc_png-HilbertCurve-method . . . . .	15
hc_points-dispatch . . . . .	16
hc_points-GenomicHilbertCurve-method . . . . .	16
hc_points-HilbertCurve-method . . . . .	17
hc_polygon-dispatch . . . . .	19
hc_polygon-GenomicHilbertCurve-method . . . . .	20
hc_polygon-HilbertCurve-method . . . . .	21
hc_rect-dispatch . . . . .	22
hc_rect-GenomicHilbertCurve-method . . . . .	22
hc_rect-HilbertCurve-method . . . . .	23
hc_segmented_points-HilbertCurve-method . . . . .	24
hc_segments-dispatch . . . . .	26
hc_segments-GenomicHilbertCurve-method . . . . .	26
hc_segments-HilbertCurve-method . . . . .	27
hc_text-dispatch . . . . .	28
hc_text-GenomicHilbertCurve-method . . . . .	28
hc_text-HilbertCurve-method . . . . .	29
HilbertCurve . . . . .	31
HilbertCurve-class . . . . .	33
is_white . . . . .	34
show-HilbertCurve-method . . . . .	35
unzoom-HilbertCurve-method . . . . .	35
zoom-HilbertCurve-method . . . . .	36
<b>Index</b>	<b>38</b>

---

default_overlay	<i>Default color overlay for adding new layers</i>
-----------------	--

---

**Description**

Default color overlay for adding new layers

**Usage**

```
default_overlay(r0, g0, b0, r, g, b, alpha = 1)
```

**Arguments**

<code>r0</code>	red channel for the layers that are already in the plot.
<code>g0</code>	green channel for the layers that are already in the plot.
<code>b0</code>	blue channel for the layers that are already in the plot.
<code>r</code>	red channel for the new layer
<code>g</code>	green channel for the new layer
<code>b</code>	blue channel for the new layer
<code>alpha</code>	alpha channel for the new layer

**Details**

The default overlay is (take red channel for example)  $r \cdot \alpha + r0 \cdot (1 - \alpha)$ .

**Value**

A list which contains overlaid RGB colors (values between 0 and 1).

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

Color overlay function is always used in [hc\\_layer, HilbertCurve-method](#) or [hc\\_layer, GenomicHilbertCurve-method](#).

**Examples**

```
# red (1, 0, 0) overlay to the grey (0.5, 0.5, 0.5) with 0.5 transparency
default_overlay(1, 0, 0, 0.5, 0.5, 0.5, 0.5)
```

---

GenomicHilbertCurve    *Initialize a Hilbert curve specifically for genomic data*

---

**Description**

Initialize a Hilbert curve specifically for genomic data

**Usage**

```
GenomicHilbertCurve(chr = paste0("chr", c(1:22, "X", "Y")), species = "hg19",
  background = NULL, ...)
```

**Arguments**

chr	a vector of chromosome names. Note it should have 'chr' prefix. This argument will be ignored when background is set.
species	abbreviation of species, e.g. 'hg19' or 'mm10'. <a href="#">read.chromInfo</a> is used to retrieve the chromosome information.
background	the background can be provided as a <a href="#">GRanges</a> object. Chromosomes should be unique across rows. Or more generally, the 'seqnames' should be different.
...	common arguments in <a href="#">HilbertCurve</a> can be used here.

**Details**

Multiple chromosomes can be visualized in a same Hilbert curve. All chromosomes are concatenated on after the other based on the order which is specified.

Since chromosomes will have irregular shapes on the curve, under 'pixel' mode, users can set border option in [hc\\_map](#), [GenomicHilbertCurve-method](#) to highlight borders of chromosomes to identify their locations on the curve.

**Value**

A [GenomicHilbertCurve-class](#) object

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
require(circlize)
require(GenomicRanges)
bed = generateRandomBed()
gr = GRanges(seqnames = bed[[1]], ranges = IRanges(bed[[2]], bed[[3]])
hc = GenomicHilbertCurve()
hc_points(hc, gr)

hc = GenomicHilbertCurve(chr = c("chr1", "chr2"))
hc_points(hc, gr)

bg = GRanges(seqnames = c("chr1", "chr2"),
  ranges = IRanges(c(1,10000000), c(10000000,20000000)))
hc = GenomicHilbertCurve(background = bg, level = 6)
hc_points(hc, gr, gp = gpar(fill = rand_color(length(gr))))
hc_map(hc, fill = NA, border = "grey", add = TRUE)
```

---

GenomicHilbertCurve-class

*The GenomicHilbertCurve class*

---

## Description

The GenomicHilbertCurve class

## Details

The [GenomicHilbertCurve-class](#) is inherited from the [HilbertCurve-class](#). Basically the structure of this class is almost the same as the [HilbertCurve-class](#) but with several additional slots added to facilitate visualizing genomic data.

## Methods

The [GenomicHilbertCurve-class](#) provides following methods:

- [GenomicHilbertCurve](#): constructor method;
- [hc\\_points, GenomicHilbertCurve-method](#): add points;
- [hc\\_segments, GenomicHilbertCurve-method](#): add lines;
- [hc\\_rect, GenomicHilbertCurve-method](#): add rectangles;
- [hc\\_polygon, GenomicHilbertCurve-method](#): add polygons;
- [hc\\_text, GenomicHilbertCurve-method](#): add text;
- [hc\\_layer, GenomicHilbertCurve-method](#): add layers under "pixel" mode;
- [hc\\_map, GenomicHilbertCurve-method](#): show the map of different categories on the curve. Works both for "normal" and "pixel" mode

The usage of above functions are almost same as those functions for the [HilbertCurve-class](#) except that the second argument which specifies the intervals should be a [GRanges](#) object.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

NULL

---

hc\_centered\_text-HilbertCurve-method

*Add text to the center of the block*

---

## Description

Add text to the center of the block

## Usage

```
## S4 method for signature 'HilbertCurve'  
hc_centered_text(object, ir = NULL, labels, x1 = NULL, x2 = NULL, gp = gpar(), ...)
```

## Arguments

object	A <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object that contains positions which correspond to text. The middle points of the intervals will be the positions of the text.
labels	text corresponding to intervals in <code>ir</code> .
x1	if start positions are not integers, they can be set by <code>x1</code> .
x2	if end positions are not integers, they can be set by <code>x2</code> .
gp	graphic parameters for text. It should be specified by <a href="#">gpar</a> .
...	pass to <a href="#">grid.text</a> . E.g. you can set text justification by <code>just</code> here.

## Details

If the interval is long enough that it represents as a block in the 2D space, the corresponding label is put approximately at center (or at least inside) of the block.

Please use [hc\\_text,HilbertCurve-method](#) directly.

## Value

NULL

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## See Also

It is used in [hc\\_map,GenomicHilbertCurve-method](#) to put chromosome names in the center of chromosomes.

### Examples

```
hc = HilbertCurve(1, 10)
hc_rect(hc, x1 = c(1, 3, 7), x2 = c(3, 7, 10), gp = gpar(fill = 2:5))
hc_centered_text(hc, x1 = 1, x2 = 3, labels = "A")
hc_centered_text(hc, x1 = 3, x2 = 7, labels = "B")
hc_centered_text(hc, x1 = 7, x2 = 10, labels = "C")
```

---

hc\_layer-dispatch      *Method dispatch page for hc\_layer*

---

### Description

Method dispatch page for hc\_layer.

### Dispatch

hc\_layer can be dispatched on following classes:

- [hc\\_layer, HilbertCurve-method, HilbertCurve-class](#) class method
- [hc\\_layer, GenomicHilbertCurve-method, GenomicHilbertCurve-class](#) class method

### Examples

```
# no example
NULL
```

---

hc\_layer-GenomicHilbertCurve-method  
*Add a new layer to the Hilbert curve*

---

### Description

Add a new layer to the Hilbert curve

### Usage

```
## S4 method for signature 'GenomicHilbertCurve'
hc_layer(object, gr, col = "red", border = NA,
  mean_mode = c("w0", "absolute", "weighted", "max_freq"), grid_line = 0,
  grid_line_col = "black", overlay = default_overlay)
```

## Arguments

object	a <a href="#">GenomicHilbertCurve-class</a> object
gr	a <a href="#">GRanges</a> object which contains the genomic regions to be mapped to the curve
col	a scalar or a vector of colors which correspond to regions in gr, pass to <a href="#">hc_layer,HilbertCurve-method</a>
border	a scalar or a vector of colors which correspond to the borders of regions. Set it to NA if borders are suppressed.
mean_mode	Under 'pixel' mode, each pixel represents a small window. This argument provides methods to summarize value for the small window if the input genomic regions can not completely overlap with the window, pass to <a href="#">hc_layer,HilbertCurve-method</a>
grid_line	whether add grid lines to show blocks of the Hilber curve, pass to <a href="#">hc_layer,HilbertCurve-method</a>
grid_line_col	color for the grid lines, pass to <a href="#">hc_layer,HilbertCurve-method</a>
overlay	a self-defined function which defines how to overlay new layer to the plot, pass to <a href="#">hc_layer,HilbertCurve-method</a>

## Details

It is basically a wrapper of [hc\\_layer,HilbertCurve-method](#).

## Value

Refer to [hc\\_layer,HilbertCurve-method](#)

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
require(circlize)
require(GenomicRanges)
bed = generateRandomBed()
gr = GRanges(seqnames = bed[[1]], ranges = IRanges(bed[[2]], bed[[3]]))
hc = GenomicHilbertCurve(mode = "pixel", level = 9)
hc_layer(hc, gr, col = rand_color(length(gr)))
```

---

hc\_layer-HilbertCurve-method

*Add a new layer to the Hilbert curve*

---

## Description

Add a new layer to the Hilbert curve



**Usage**

```
## S4 method for signature 'HilbertCurve'
hc_layer(object, ir = NULL, x1 = NULL, x2 = x1, col = "red", border = NA,
         mean_mode = c("w0", "absolute", "weighted", "max_freq"), grid_line = 0,
         grid_line_col = "black", overlay = default_overlay)
```

**Arguments**

object	A <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object which specifies the input intervals.
x1	if start positions are not integers, they can be set by x1.
x2	if end positions are not integers, they can be set by x2.
col	a scalar or a vector of colors which correspond to intervals in ir (or x1 and x2).
border	a scalar or a vector of colors for the borders of intervals. Set it to NA if borders are suppressed.
mean_mode	Under 'pixel' mode, each pixel represents a small window. This argument provides methods to summarize value for the small window if the input intervals can not completely overlap with the window. See explanation in <a href="#">hc_points</a> , <a href="#">HilbertCurve-method</a> .
grid_line	whether add grid lines to show blocks of the Hilber curve. It should be an integer number and there will be $2^{(\text{grid\_line}-1)}-1$ horizontal and vertical grid lines.
grid_line_col	color for the grid lines
overlay	a self-defined function which defines how to overlay new layer to the plot. By default it is <a href="#">default_overlay</a> . Let's assume the red channel for the layers which are already in the plot is $r_0$ , the red channel for the new layer is $r$ and the alpha channel is $\alpha$ , the overlaid color is calculated as $r*\alpha + r_0*(1-\alpha)$ . This self-defined function should accept 7 arguments which are: vectors of r, g, b channels which correspond to the layers that are already in the plot, and r, g, b, alpha channels which corresponds to the new layer. All the values passed into are between 0 to 1. The returned value for this function should be a list which contains r, g, b channels which correspond to the overlaid colors. Note that these 7 arguments only correspond to the pixels which are covered by the new layer.

**Details**

This function only works under 'pixel' mode.

Under "pixel" mode, color is the only graphic representation of values in the input intervals. To make a more precise and robust color mapping, users may consider [colorRamp2](#) to create a color mapping function.

If you want to add more than one layers to the curve, remember to set colors with transparency.

overly argument is useful for changing color themes for the overlapped areas, please refer to the vignette to see examples of how to swith color themes in easy ways.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
hc = HilbertCurve(1, 100, level = 9, mode = "pixel")

x = sort(sample(100, 20))
s = x[1:10*2 - 1]
e = x[1:10*2]
require(IRanges)
ir = IRanges(s, e)

hc_layer(hc, ir)

hc = HilbertCurve(1, 100, level = 9, mode = "pixel")
hc_layer(hc, ir, grid_line = 3)

hc = HilbertCurve(1, 100, level = 9, mode = "pixel")
hc_layer(hc, ir, border = "black")
```

---

hc\_level-HilbertCurve-method

*Level of the Hilbert curve*

---

**Description**

Level of the Hilbert curve

**Usage**

```
## S4 method for signature 'HilbertCurve'
hc_level(object)
```

**Arguments**

object            A [HilbertCurve-class](#) object.

**Value**

The level of the Hilbert curve.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
hc = HilbertCurve(1, 100)
hc_level(hc)

hc = HilbertCurve(1, 100, level = 5)
hc_level(hc)
```

---

hc\_map-GenomicHilbertCurve-method

*Draw a map which represents positions of different chromosomes on the curve*

---

**Description**

Draw a map which represents positions of different chromosomes on the curve

**Usage**

```
## S4 method for signature 'GenomicHilbertCurve'
hc_map(object, level = 7,
        fill = rand_color(length(background), transparency = 0.5), border = NA,
        labels = names(object@background), show_labels = TRUE, labels_gp = gpar(),
        add = FALSE, ...)
```

**Arguments**

object	a <a href="#">GenomicHilbertCurve-class</a> object
level	Since a map does not need to have high resolution, a value of around 7 would be enough. If add is set to TRUE, level will be enforced to have the same level in the current Hilbert curve.
fill	colors for different chromosomes, or more generally, for different 'seqnames'.
border	colors for the borders of chromosomes. Set it to NA if borders are suppressed.
labels	label for each chromosome, or more generally, for different 'sequences'
show_labels	whether show text labels
labels_gp	graphic settings for labels
add	whether add the map to the current curve or draw it in a new graphic device. Notice if add is set to TRUE, you should set fill with transparency so that it will not hide your original plot.
...	pass to <a href="#">GenomicHilbertCurve</a> . It is only used if you want the map to be plotted in a new graphic device.

## Details

When multiple genomic categories (e.g. chromosomes) are drawn into one single Hilbert curve, a map which shows the positions of categories on the curve is necessary to distinguish different genomic categories.

Under "pixel" mode, if the map is directly added to the Hilbert curve, no chromosome name is drawn. The chromosome names are only drawn if the map is plotted in a new graphic device or added to the Hilbert curve under "normal" mode.

Just be careful if you directly overlay the map to the curve that the color of the map does not affect the original plot too much.

## Value

A [GenomicHilbertCurve-class](#) object

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
require(circlize)
require(GenomicRanges)
bed = generateRandomBed(nr = 100)
gr = GRanges(seqnames = bed[[1]], ranges = IRanges(bed[[2]], bed[[3]]))
hc = GenomicHilbertCurve()
hc_points(hc, gr, gp = gpar(fill = rand_color(length(gr))))
# add it in the same graphic device
hc_map(hc, fill = rand_color(24, transparency = 0.5), add = TRUE)

# add the map only with borders
hc = GenomicHilbertCurve()
hc_points(hc, gr, gp = gpar(fill = rand_color(length(gr))))
hc_map(hc, fill = NA, border = "grey", add = TRUE)

# or open a new graphic device
hc_map(hc, fill = rand_color(24))
```

---

hc\_normal\_points-HilbertCurve-method

*Add points to the Hilbert curve*

---

## Description

Add points to the Hilbert curve

**Usage**

```
## S4 method for signature 'HilbertCurve'  
hc_normal_points(object, ir = NULL, x1 = NULL, x2 = x1, gp = gpar(),  
  pch = 1, size = unit(1, "char"))
```

**Arguments**

object	A <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object which specifies the input intervals.
x1	if start positions are not integers, they can be set by x1.
x2	if end positions are not integers, they can be set by x2.
size	size of the points. It should be a <a href="#">unit</a> object, pass to <a href="#">grid.points</a> .
pch	shape of points, pass to <a href="#">grid.points</a> .
gp	graphic parameters for points. It should be specified by <a href="#">gpar</a> .

**Details**

Points are added at the middle of the intervals in `ir` (or `x1` and `x2`), so there is only one point for each interval.

This function is used internally. Please use [hc\\_points, HilbertCurve-method](#) instead.

**Value**

A data frame which contains coordinates (in the 2D space) of points.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[hc\\_points, HilbertCurve-method](#)

**Examples**

```
# see documentation of hc_points  
NULL
```

---

hc\_offset-HilbertCurve-method  
*Adjust positions*

---

## Description

Adjust positions

## Usage

```
## S4 method for signature 'HilbertCurve'  
hc_offset(object, x)
```

## Arguments

object	A <a href="#">HilbertCurve-class</a> object.
x	positions.

## Details

Since internally positions are transformed to positive integers, if input positions are specified as negative values when initializing the Hilbert curve, a shift will be recorded internally and positions are transformed to positive value automatically.

## Value

A positive numeric value.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
hc = HilbertCurve(-100, 100)  
hc_offset(hc, c(-100, -50, 0, 50, 100))
```

---

hc\_png-HilbertCurve-method

*Save Hilbert curve as a PNG figure*

---

## Description

Save Hilbert curve as a PNG figure

## Usage

```
## S4 method for signature 'HilbertCurve'  
hc_png(object, file = "HilbertCurve.png")
```

## Arguments

object	A <a href="#">HilbertCurve-class</a> object.
file	file name. If the suffix of the file name is not .png, it will be added automatically no matter you like it or not.

## Details

A PNG figure with resolution of  $2^{\text{level}} \times 2^{\text{level}}$  is generated.

Only the body of the Hilbert curve will be written to PNG file.

This function only works under 'pixel' mode.

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
hc = HilbertCurve(1, 100, level = 9, mode = "pixel")  
  
x = sort(sample(100, 20))  
s = x[1:10*2 - 1]  
e = x[1:10*2]  
require(IRanges)  
ir = IRanges(s, e)  
  
hc_layer(hc, ir)  
hc_png(hc, file = "test.png")
```

---

hc\_points-dispatch      *Method dispatch page for hc\_points*

---

### Description

Method dispatch page for hc\_points.

### Dispatch

hc\_points can be dispatched on following classes:

- [hc\\_points, GenomicHilbertCurve-method, GenomicHilbertCurve-class](#) class method
- [hc\\_points, HilbertCurve-method, HilbertCurve-class](#) class method

### Examples

```
# no example
NULL
```

---

hc\_points-GenomicHilbertCurve-method  
*Add points to the Hilbert curve*

---

### Description

Add points to the Hilbert curve

### Usage

```
## S4 method for signature 'GenomicHilbertCurve'
hc_points(object, gr,
  np = max(c(2, 10 - hc_level(object))), size = unit(1, "char"),
  pch = 1, gp = gpar(), mean_mode = c("w0", "absolute", "weighted", "max_freq"),
  shape = "circle")
```

### Arguments

object	a <a href="#">GenomicHilbertCurve-class</a> object
gr	a <a href="#">GRanges</a> object which contains the genomic regions to be mapped to the curve
np	pass to <a href="#">hc_points, HilbertCurve-method</a>
size	size of points when np <= 1, pass to <a href="#">hc_points, HilbertCurve-method</a>
pch	shape of the points when np <= 1, pass to <a href="#">hc_points, HilbertCurve-method</a>
gp	graphic parameters of the points when np <= 1, pass to <a href="#">hc_points, HilbertCurve-method</a>
mean_mode	pass to <a href="#">hc_points, HilbertCurve-method</a>
shape	shape of the points when np >= 2, pass to <a href="#">hc_points, HilbertCurve-method</a>



**Details**

It is basically a wrapper of [hc\\_points,HilbertCurve-method](#).

**Value**

Refer to [hc\\_points,HilbertCurve-method](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
require(circlize)
require(GenomicRanges)
bed = generateRandomBed(nr = 100)
gr = GRanges(seqnames = bed[[1]], ranges = IRanges(bed[[2]], bed[[3]]))
hc = GenomicHilbertCurve()
hc_points(hc, gr, gp = gpar(fill = rand_color(length(gr))))
```

---

hc\_points-HilbertCurve-method

*Add points to the Hilbert curve*

---

**Description**

Add points to the Hilbert curve

**Usage**

```
## S4 method for signature 'HilbertCurve'
hc_points(object, ir = NULL, x1 = NULL, x2 = x1,
  np = max(c(2, 10 - hc_level(object))), size = unit(1, "char"),
  pch = 1, gp = gpar(), mean_mode = c("w0", "absolute", "weighted", "max_freq"),
  shape = "circle")
```

**Arguments**

object	A <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object which specifies the input intervals.
x1	if start positions are not integers, they can be set by x1.
x2	if end positions are not integers, they can be set by x2.
np	number of points (a circle or a square, ...) that are put in a segment. np controls the mode of how to add the points to the curve. See 'Details' section.
size	size of the points. It should be a <a href="#">unit</a> object. Only works if np <= 1
pch	shape of points, used for points if np <= 1.

gpar	graphic parameters for points. It should be specified by <code>gpar</code> .
mean_mode	when $np \geq 2$ , each segment on the curve is split into $np$ windows and each window actually represents a small interval in the axis. When overlapping input intervals to the windows on the curve and when the window can not completely cover the input intervals, some averaging method should be applied to get a more accurate estimation for the value in the window. Here the HilbertCurve package provides four modes: "w0", "weighted", "absolute" and "max_freq" which calculate the mean value in the window with respect to different scenarios. See 'Details' section and the vignette for more informative explanation.
shape	shape of points, used for points if $np \geq 2$ . Possible values are "circle", "square", "triangle", "hexagon", "star".

### Details

If  $np$  is set to 1 or NULL, points will be added in the middle for each interval in `ir` (or `x1`, `x2`).

If  $np$  is set to a value larger or equal to 2, every segment on the curve is split by  $np$  points (e.g. circles). In this case, each point actually represent a window on the curve and when the window is not fully covered by the input intervals, there are three different metrics to average the values in the window.

Following illustrates different settings for `mean_mode`:

100	80	60	values in ir
+++++	+++	++++	ir
=====			window (width = 16)
4	3	3	overlap

absolute:  $(100 + 80 + 60)/3$

weighted:  $(100*4 + 80*3 + 60*3)/(4 + 3 + 3)$

w0:  $(100*4 + 80*3 + 60*3 + 0*6)/16$

So which mode to use depends on specific scenario. If the background is not of interest, absolute and weighted modes may be proper and if the value also needs to be averaged with background, w0 is the proper choice. Section "Averaging models" in the vignette gives a more detailed explanation for this argument.

There is one more value for `mean_mode` which is `max_freq`. `max_freq` is mainly for discrete signals and in a segment, value with the highest frequency (or with the highest length) is selected for this segment

If  $np \geq 2$ , the value of  $np$  also controls the size of points.

Graphic parameters is always represented as numeric values (e.g. colors can be converted into numeric RGB values) and they will be averaged according to above rules.

Internally, it will depatch to `hc_normal_points, HilbertCurve-method` or `hc_segmented_points, HilbertCurve-method` depending on the value of  $np$ .

### Value

A data frame which contains coordinates (in the 2D space) of points.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
hc = HilbertCurve(1, 100, level = 4, reference = TRUE)

x = sort(sample(100, 20))
s = x[1:10*2 - 1]
e = x[1:10*2]
require(IRanges)
ir = IRanges(s, e)

hc_points(hc, ir)

hc = HilbertCurve(1, 100, level = 4, reference = TRUE)
hc_points(hc, x1 = c(1.5, 50.5), x2 = c(10.5, 60.5))

require(circlize)
value = runif(length(ir))
col_fun = colorRamp2(range(value), c("white", "red"))
hc = HilbertCurve(1, 100, level = 4, reference = TRUE)
hc_points(hc, ir, np = 3, shape = "star", gp = gpar(fill = col_fun(value)))

hc = HilbertCurve(1, 100, level = 4, reference = TRUE)
hc_points(hc, ir, np = 0)

hc = HilbertCurve(1, 100, level = 4, reference = TRUE)
hc_points(hc, np = 0, x1 = c(1.5, 50.5), x2 = c(10.5, 60.5))
hc_points(hc, np = 0, x1 = 70.5, gp = gpar(col = "red"))
```

---

hc\_polygon-dispatch    *Method dispatch page for hc\_polygon*

---

**Description**

Method dispatch page for hc\_polygon.

**Dispatch**

hc\_polygon can be dispatched on following classes:

- [hc\\_polygon,HilbertCurve-method,HilbertCurve-class](#) class method
- [hc\\_polygon,GenomicHilbertCurve-method,GenomicHilbertCurve-class](#) class method

**Examples**

```
# no example
NULL
```

---

hc\_polygon-GenomicHilbertCurve-method  
*Add text to Hilbert curve*

---

**Description**

Add text to Hilbert curve

**Usage**

```
## S4 method for signature 'GenomicHilbertCurve'
hc_polygon(object, gr, gp = gpar(),
           end_type = c("average", "expanding", "shrinking"))
```

**Arguments**

object	a <a href="#">GenomicHilbertCurve-class</a> object
gr	a <a href="#">GRanges</a> object which contains the genomic regions to be mapped to the curve
gp	pass to <a href="#">hc_polygon,HilbertCurve-method</a>
end_type	pass to <a href="#">hc_polygon,HilbertCurve-method</a>

**Details**

It is basically a wrapper of [hc\\_polygon,HilbertCurve-method](#).

**Value**

Refer to [hc\\_polygon,HilbertCurve-method](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
require(circlize)
require(GenomicRanges)
bed = generateRandomBed(nr = 20)
gr = GRanges(seqnames = bed[[1]], ranges = IRanges(bed[[2]], bed[[3]]))
hc = GenomicHilbertCurve()
hc_polygon(hc, gr)
```

---

hc\_polygon-HilbertCurve-method  
*Add polygons to Hilbert curve*

---

## Description

Add polygons to Hilbert curve

## Usage

```
## S4 method for signature 'HilbertCurve'  
hc_polygon(object, ir = NULL, x1 = NULL, x2 = NULL,  
           gp = gpar(), end_type = c("expanding", "average", "shrinking"))
```

## Arguments

object	a <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object which specifies the input intervals.
x1	if start positions are not integers, they can be set by x1.
x2	if end positions are not integers, they can be set by x2.
gp	graphic parameters. It should be specified by <a href="#">gpar</a> .
end_type	since two ends of a continuous interval do not necessarily completely overlap with the Hilbert curve segments, this argument controls how to determine the ends of the interval which will be presented on the curve. average: if the end covers more than half of the segment, the whole segment is included and if the end covers less than half of the segment, the segment is removed; expanding: segments are included as long as they are overlapped; shrinking: segments are removed if they are not completely covered.

## Details

Drawing polygons are quite visually similar as drawing rectangles. The major differences are: 1) for rectangles, colors for the ends of the interval can change if they are not completely covered by the Hilbert curve segments, and 2) polygons can have borders.

Normally polygons are used to mark areas in the Hilbert curve.

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
require(IRanges)
ir = IRanges(10, 40)

hc = HilbertCurve(0, 100, level = 4, reference = TRUE)
hc_segments(hc, ir)
hc_text(hc, x1 = 10:40, labels = 10:40)
hc_polygon(hc, ir, gp = gpar(fill = "#FF000080", col = 1))
```

---

hc\_rect-dispatch      *Method dispatch page for hc\_rect*

---

**Description**

Method dispatch page for hc\_rect.

**Dispatch**

hc\_rect can be dispatched on following classes:

- [hc\\_rect,HilbertCurve-method,HilbertCurve-class](#) class method
- [hc\\_rect,GenomicHilbertCurve-method,GenomicHilbertCurve-class](#) class method

**Examples**

```
# no example
NULL
```

---

hc\_rect-GenomicHilbertCurve-method  
*Add rectangles on Hilbert curve*

---

**Description**

Add rectangles on Hilbert curve

**Usage**

```
## S4 method for signature 'GenomicHilbertCurve'
hc_rect(object, gr, gp = gpar(fill = "red", col = "red"),
        mean_mode = c("w0", "absolute", "weighted", "max_freq"))
```

**Arguments**

object	a <a href="#">GenomicHilbertCurve-class</a> object
gr	a <a href="#">GRanges</a> object which contains the genomic regions to be mapped to the curve
gp	pass to <a href="#">hc_rect,HilbertCurve-method</a>
mean_mode	pass to <a href="#">hc_rect,HilbertCurve-method</a>

**Details**

It is basically a wrapper of [hc\\_rect,HilbertCurve-method](#).

**Value**

Refer to [hc\\_rect,HilbertCurve-method](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
require(circlize)
require(GenomicRanges)
bed = generateRandomBed(nr = 100)
gr = GRanges(seqnames = bed[[1]], ranges = IRanges(bed[[2]], bed[[3]]))
hc = GenomicHilbertCurve()
hc_rect(hc, gr, gp = gpar(fill = rand_color(length(gr))))
```

---

hc\_rect-HilbertCurve-method

*Add rectangles on Hilbert curve*

---

**Description**

Add rectangles on Hilbert curve

**Usage**

```
## S4 method for signature 'HilbertCurve'
hc_rect(object, ir = NULL, x1 = NULL, x2 = NULL,
         gp = gpar(fill = "red"),
         mean_mode = c("w0", "absolute", "weighted", "max_freq"))
```

**Arguments**

object	A <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object which specifies the input intervals.
x1	if start positions are not integers, they can be set by x1.
x2	if end positions are not integers, they can be set by x2.
gp	graphic parameters for rectangles. It should be specified by <a href="#">gpar</a> . Note you cannot set <code>linejoin</code> and <code>lineend</code> .
mean_mode	when a segment in the curve can not be overlapped with intervals in <code>ir</code> , how to calculate the mean values for this segment. See explanation in <a href="#">hc_points,HilbertCurve-method</a> .

**Details**

Rectangles are put if a segment in the Hilbert curve overlaps with the input intervals. You cannot set the width or height of the rectangles. It is always fixed (actually it is a square).

It can be thought as the low-resolution version of [hc\\_layer,HilbertCurve-method](#).

**Value**

A data frame which contains coordinates (in the 2D space) of rectangles.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
hc = HilbertCurve(1, 100, level = 4, reference = TRUE)

x = sort(sample(100, 20))
s = x[1:10*2 - 1]
e = x[1:10*2]
require(IRanges)
ir = IRanges(s, e)
hc_rect(hc, ir)
```

---

hc\_segmented\_points-HilbertCurve-method  
*Add points to the Hilbert curve*

---

**Description**

Add points to the Hilbert curve



**Usage**

```
## S4 method for signature 'HilbertCurve'  
hc_segmented_points(object, ir = NULL, x1 = NULL, x2 = NULL, gp = gpar(),  
  np = max(c(2, 10 - hc_level(object))),  
  mean_mode = c("w0", "absolute", "weighted", "max_freq"),  
  shape = "circle")
```

**Arguments**

object	A <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object which specifies the input intervals.
x1	if start positions are not integers, they can be set by x1.
x2	if end positions are not integers, they can be set by x2.
np	number of points (a circle or a square, ...) that are put in a segment.
gp	graphic parameters for points. It should be specified by <a href="#">gpar</a> . The size of the points can be set here because the size of points are determined by np argument.
mean_mode	when a segment in the curve overlaps with intervals in ir, how to calculate the mean values for this segment. See explanation in <a href="#">hc_points</a> .
shape	shape of points. Possible values are "circle", "square", "triangle", "hexagon", "star".

**Details**

Every segment on the curve is split by np points.

This function is used internally, please use [hc\\_points,HilbertCurve-method](#) directly.

**Value**

A data frame which contains coordinates (in the 2D space) of points.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# see documentation of hc_points  
NULL
```

---

hc\_segments-dispatch *Method dispatch page for hc\_segments*

---

### Description

Method dispatch page for hc\_segments.

### Dispatch

hc\_segments can be dispatched on following classes:

- [hc\\_segments, GenomicHilbertCurve-method, GenomicHilbertCurve-class](#) class method
- [hc\\_segments, HilbertCurve-method, HilbertCurve-class](#) class method

### Examples

```
# no example  
NULL
```

---

hc\_segments-GenomicHilbertCurve-method  
*Add line segments to Hilbert curve*

---

### Description

Add line segments to Hilbert curve

### Usage

```
## S4 method for signature 'GenomicHilbertCurve'  
hc_segments(object, gr, gp = gpar(lty = 1, lwd = 1, col = 1))
```

### Arguments

object	a <a href="#">GenomicHilbertCurve-class</a> object
gr	a <a href="#">GRanges</a> object which contains the genomic regions to be mapped to the curve
gp	pass to <a href="#">hc_segments, HilbertCurve-method</a>

### Details

It is basically a wrapper of [hc\\_segments, HilbertCurve-method](#).

**Value**

Refer to [hc\\_segments,HilbertCurve-method](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
require(circlize)
require(GenomicRanges)
bed = generateRandomBed(nr = 100)
gr = GRanges(seqnames = bed[[1]], ranges = IRanges(bed[[2]], bed[[3]])
hc = GenomicHilbertCurve()
hc_segments(hc, gr, gp = gpar(col = rand_color(length(gr))))
```

---

hc\_segments-HilbertCurve-method

*Add line segments to Hilbert curve*

---

**Description**

Add line segments to Hilbert curve

**Usage**

```
## S4 method for signature 'HilbertCurve'
hc_segments(object, ir = NULL, x1 = NULL, x2 = NULL,
            gp = gpar(lty = 1, lwd = 1, col = 1))
```

**Arguments**

object	A <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object which specifies the input intervals.
x1	if start positions are not integers, they can be set by x1.
x2	if end positions are not integers, they can be set by x2.
gp	graphic parameters for lines. It should be specified by <a href="#">gpar</a> . Note you cannot set <code>linejoin</code> and <code>lineend</code> .

**Value**

A data frame which contains coordinates (in the 2D space) of segments.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```

hc = HilbertCurve(1, 100, level = 4, reference = TRUE)

x = sort(sample(100, 20))
s = x[1:10*2 - 1]
e = x[1:10*2]
require(IRanges)
ir = IRanges(s, e)

hc_segments(hc, ir)

```

---

[hc\\_text-dispatch](#)
*Method dispatch page for hc\_text*


---

**Description**

Method dispatch page for hc\_text.

**Dispatch**

hc\_text can be dispatched on following classes:

- [hc\\_text,GenomicHilbertCurve-method, GenomicHilbertCurve-class](#) class method
- [hc\\_text,HilbertCurve-method, HilbertCurve-class](#) class method

**Examples**

```

# no example
NULL

```

---

[hc\\_text-GenomicHilbertCurve-method](#)
*Add text to Hilbert curve*


---

**Description**

Add text to Hilbert curve

**Usage**

```

## S4 method for signature 'GenomicHilbertCurve'
hc_text(object, gr, labels, gp = gpar(),
        centered_by = c("interval", "polygon"), ...)

```

**Arguments**

object	a <a href="#">GenomicHilbertCurve-class</a> object
gr	a <a href="#">GRanges</a> object which contains the genomic regions to be mapped to the curve
labels	pass to <a href="#">hc_text,HilbertCurve-method</a>
gp	pass to <a href="#">hc_text,HilbertCurve-method</a>
centered_by	how to define the "center" of the interval represented in Hilbert curve. Pass to <a href="#">hc_text,HilbertCurve-method</a> .
...	pass to <a href="#">hc_text,HilbertCurve-method</a>

**Details**

It is basically a wrapper of [hc\\_text,HilbertCurve-method](#).

**Value**

Refer to [hc\\_text,HilbertCurve-method](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
require(circlize)
require(GenomicRanges)
bed = generateRandomBed(nr = 20)
gr = GRanges(seqnames = bed[[1]], ranges = IRanges(bed[[2]], bed[[3]]))
hc = GenomicHilbertCurve()
hc_text(hc, gr, labels = sample(letters, nrow(bed), replace = TRUE))
```

---

hc\_text-HilbertCurve-method

*Add text to Hilbert curve*

---

**Description**

Add text to Hilbert curve

**Usage**

```
## S4 method for signature 'HilbertCurve'
hc_text(object, ir = NULL, labels, x1 = NULL, x2 = x1, gp = gpar(),
        centered_by = c("interval", "polygon"), ...)
```

**Arguments**

object	A <a href="#">HilbertCurve-class</a> object.
ir	an <a href="#">IRanges</a> object that contains positions which correspond to text. The middle point of the interval will be the position of the text.
labels	text corresponding to intervals in ir.
x1	if start positions are not integers, they can be set by x1.
x2	if end positions are not integers, they can be set by x2.
gp	graphic parameters for text. It should be specified by <a href="#">gpar</a> .
centered_by	how to define the "center" of the interval represented in Hilbert curve. See <a href="#">Details</a> section.
...	pass to <a href="#">grid.text</a> . E.g. you can set text justification by just here.

**Details**

If `centered_by == "interval"`, the text is added corresponding to the middle of each interval in `ir`, while if `centered_by == "polygon"`, the text is put in the visual center of the polygon of the interval in the Hilbert curve.

**Value**

A data frame which contains coordinates (in the 2D space) of text.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
hc = HilbertCurve(1, 100, level = 4, reference = TRUE)

x = sort(sample(100, 20))
s = x[1:10*2 - 1]
e = x[1:10*2]
require(IRanges)
ir = IRanges(s, e)

labels = sample(letters, length(ir), replace = TRUE)
hc_text(hc, ir, labels = labels)
```

---

HilbertCurve	<i>Initialize a Hilbert curve</i>
--------------	-----------------------------------

---

## Description

Initialize a Hilbert curve

## Usage

```
HilbertCurve(s, e, level = 4, mode = c("normal", "pixel"),
  reference = FALSE, reference_gp = gpar(lty = 3, col = "#999999"),
  arrow = TRUE, zoom = NULL, newpage = TRUE,
  background_col = "transparent", background_border = NA,
  title = NULL, title_gp = gpar(fontsize = 16),
  start_from = c("bottomleft", "topleft", "bottomright", "topright"),
  first_seg = c("horizontal", "vertical"), legend = list(),
  padding = unit(2, "mm"))
```

## Arguments

s	position that will be mapped as the start of the Hilbert curve. The value should be a single numeric value. If it is a vector, the minimum is used.
e	position that will be mapped as the end of the Hilbert curve. The value should be a single numeric value. If it is a vector, the maximum is used.
level	iteration level of the Hilbert curve. There will be $4^{\text{level}} - 1$ segments in the curve.
mode	"normal" mode is used for low level value and "pixel" mode is always used for high level value, so the "normal" mode is always for low-resolution visualization while "pixel" mode is used for high-resolution visualization. See 'details' for explanation.
reference	whether add reference lines on the plot. Only works under 'normal' mode. The reference line is only used for illustrating how the curve folds.
reference_gp	graphic settings for the reference lines. It should be specified by <a href="#">gpar</a> .
arrow	whether add arrows on the reference line. Only works under 'normal' mode.
zoom	Internally, position are stored as integer values. To better map the data to the Hilbert curve, the original positions are zoomed according to the range and the level of Hilbert curve. E.g. if the curve visualizes data ranging from 1 to 2 but level of the curve is set to 4, the positions will be zoomed by $\sim x2000$ so that values like 1.5, 1.555 can be mapped to the curve with more accuracy. You don't need to care the zooming thing, proper zooming factor is calculated automatically.
newpage	whether call <a href="#">grid.newpage</a> to draw on a new graphic device.
background_col	background color.

background_border	background border border.
title	title of the plot.
title_gp	graphic parameters for the title. It should be specified by <a href="#">gpar</a> .
start_from	which corner on the plot should the curve starts?
first_seg	the orientation of the first segment.
legend	a <a href="#">grob</a> object, a <a href="#">Legends-class</a> object, or a list of them.
padding	padding around the Hilbert curve.

## Details

This function initializes a Hilbert curve with level `level` which corresponds to the range between `s` and `e`.

Under 'normal' mode, there is a visible Hilbert curve which plays like a folded axis and different low-level graphics can be added afterwards according to the coordinates. It works nice if the level of the Hilbert curve is small (say less than 6).

When the level is high (e.g.  $> 10$ ), the whole 2D space will be almost completely filled by the curve and it is impossible to add or visualize e.g. points on the curve. In this case, the 'pixel' mode visualizes each tiny 'segment' as a pixel and maps values to colors. Internally, the whole plot is represented as an RGB matrix and every time a new layer is added to the plot, the RGB matrix will be updated according to the color overlay. When all the layers are added, normally a PNG figure is generated directly from the RGB matrix. So the Hilbert curve with level 11 will generate a PNG figure with 2048x2048 resolution. This is extremely useful for visualize genomic data. E.g. If we make a Hilbert curve for human chromosome 1 with level 11, then each pixel can represent 60bp ( $249250621/2048/2048$ ) which is of very high resolution.

Under 'pixel' mode, if the current device is an interactive device, every time a new layer is added, the image will be added to the interactive device as a rastered image. But still you can use [hc\\_png](#), [HilbertCurve-method](#) to export the plot as PNG file.

To make it short and clear, under "normal" mode, you can use following low-level graphic functions:

- [hc\\_points](#), [HilbertCurve-method](#)
- [hc\\_segments](#), [HilbertCurve-method](#)
- [hc\\_rect](#), [HilbertCurve-method](#)
- [hc\\_polygon](#), [HilbertCurve-method](#)
- [hc\\_text](#), [HilbertCurve-method](#)

And under "pixel" mode, you can use following functions:

- [hc\\_layer](#), [HilbertCurve-method](#)
- [hc\\_png](#), [HilbertCurve-method](#)
- [hc\\_polygon](#), [HilbertCurve-method](#)
- [hc\\_text](#), [HilbertCurve-method](#)

Notice, `s` and `e` are not necessarily to be integers, it can be any values (e.g. numeric or even negative values).



**Value**

A `HilbertCurve-class` object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
HilbertCurve(1, 100, reference = TRUE)
HilbertCurve(1, 100, level = 5, reference = TRUE)
HilbertCurve(1, 100, title = "title", reference = TRUE)
HilbertCurve(1, 100, start_from = "topleft", reference = TRUE)

# plot with one legend
require(ComplexHeatmap)
legend = Legend(labels = c("a", "b"), title = "foo",
  legend_gp = gpar(fill = c("red", "blue")))
hc = HilbertCurve(1, 100, title = "title", legend = legend)
hc_segments(hc, x1 = 20, x2 = 40)

# plot with more than one legend
require(circlize)
legend1 = Legend(labels = c("a", "b"), title = "foo",
  legend_gp = gpar(fill = c("red", "blue")))
col_fun = colorRamp2(c(-1, 0, 1), c("green", "white", "red"))
legend2 = Legend(col_fun = col_fun, title = "bar")
hc = HilbertCurve(1, 100, title = "title", legend = list(legend1, legend2))
hc_segments(hc, x1 = 20, x2 = 40)
```

---

HilbertCurve-class      *The HilbertCurve class*

---

**Description**

The `HilbertCurve` class

**Details**

Hilbert curve ([https://en.wikipedia.org/wiki/Hilbert\\_curve](https://en.wikipedia.org/wiki/Hilbert_curve)) is a type of space-filling curves that folds one-dimensional axis into a two-dimensional space, but still keeps the locality. It has advantages to visualize data with long axis with high resolution.

This package aims to provide an easy and flexible way to visualize data through Hilbert curve. The implementation and example figures are based on following sources:

- <http://mkweb.bcgsc.ca/hilbert/>
- <http://corte.si/posts/code/hilbert/portrait/index.html>
- <http://bioconductor.org/packages/devel/bioc/html/HilbertVis.html>

**Methods**

The `HilbertCurve`-class provides following methods:

- `HilbertCurve`: constructor method;
- `hc_points,HilbertCurve-method`: add points;
- `hc_segments,HilbertCurve-method`: add lines;
- `hc_rect,HilbertCurve-method`: add rectangles;
- `hc_polygon,HilbertCurve-method`: add polygons;
- `hc_text,HilbertCurve-method`: add text;
- `hc_layer,HilbertCurve-method`: add layers, works under "pixel" mode;
- `hc_png,HilbertCurve-method`: save plot as PNG format, works under "pixel" mode.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

The `GenomicHilbertCurve`-class inherits `HilbertCurve`-class and is designed specifically for handling genomic data.

**Examples**

NULL

---

is\_white

*Whether the color is white*

---

**Description**

Whether the color is white

**Usage**

```
is_white(r, g, b, maxColorValue = 1)
```

**Arguments**

r	Red channel.
g	Green channel.
b	Blue channel.
maxColorValue	1 or 255.

**Examples**

```
# There is no example
NULL
```

---

show-HilbertCurve-method

*Print the HilbertCurve object*

---

### Description

Print the HilbertCurve object

### Usage

```
## S4 method for signature 'HilbertCurve'  
show(object)
```

### Arguments

object            A [HilbertCurve-class](#) object.

### Value

No value is returned.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
HilbertCurve(1, 100)
```

---

unzoom-HilbertCurve-method

*Transform zoomed positions to their original values*

---

### Description

Transform zoomed positions to their original values

### Usage

```
## S4 method for signature 'HilbertCurve'  
unzoom(object, x)
```

### Arguments

object            A [HilbertCurve-class](#) object.  
x                  positions.

**Details**

This is a reverse function of [zoom,HilbertCurve-method](#).

The function is used internally.

**Value**

A numeric vector of original positions.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
hc = HilbertCurve(1, 2)
z = zoom(hc, 1.5)
unzoom(hc, z)
```

---

zoom-HilbertCurve-method

*Zoom original positions*

---

**Description**

Zoom original positions

**Usage**

```
## S4 method for signature 'HilbertCurve'
zoom(object, x)
```

**Arguments**

object	A <a href="#">HilbertCurve-class</a> object.
x	original positions.

**Details**

Internally, position are stored as integer values. To better map the data to the Hilbert curve, the original positions are zoomed according to the range and the level of Hilbert curve. E.g. if the curve visualizes data ranging from 1 to 2 but level of the curve is set to 4, the positions will be zoomed by  $\sim x2000$  so that values like 1.5, 1.555 can be mapped to the curve with more accuracy.

The function is used internally.

**Value**

A numeric vector which is zoomed positions.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
hc = HilbertCurve(1, 2)
zoom(hc, 1.5)
```

# Index

colorRamp2, 9

default\_overlay, 2, 9

GenomicHilbertCurve, 3, 5, 11

GenomicHilbertCurve-class, 5

gpar, 6, 13, 18, 21, 24, 25, 27, 30–32

GRanges, 4, 5, 8, 16, 20, 23, 26, 29

grid.newpage, 31

grid.points, 13

grid.text, 6, 30

grob, 32

hc\_centered\_text  
(hc\_centered\_text-HilbertCurve-method), 6

hc\_centered\_text, HilbertCurve-method  
(hc\_centered\_text-HilbertCurve-method), 6

hc\_centered\_text-HilbertCurve-method, 6

hc\_layer (hc\_layer-dispatch), 7

hc\_layer, GenomicHilbertCurve-method  
(hc\_layer-GenomicHilbertCurve-method), 7

hc\_layer, HilbertCurve-method  
(hc\_layer-HilbertCurve-method), 8

hc\_layer-dispatch, 7

hc\_layer-GenomicHilbertCurve-method, 7

hc\_layer-HilbertCurve-method, 8

hc\_level  
(hc\_level-HilbertCurve-method), 10

hc\_level, HilbertCurve-method  
(hc\_level-HilbertCurve-method), 10

hc\_level-HilbertCurve-method, 10

hc\_map  
(hc\_map-GenomicHilbertCurve-method), 11

hc\_map, GenomicHilbertCurve-method  
(hc\_map-GenomicHilbertCurve-method), 11

hc\_map-GenomicHilbertCurve-method, 11

hc\_normal\_points  
(hc\_normal\_points-HilbertCurve-method), 12

hc\_normal\_points, HilbertCurve-method  
(hc\_normal\_points-HilbertCurve-method), 12

hc\_normal\_points-HilbertCurve-method, 12

hc\_offset  
(hc\_offset-HilbertCurve-method), 14

hc\_offset, HilbertCurve-method  
(hc\_offset-HilbertCurve-method), 14

hc\_offset-HilbertCurve-method, 14

hc\_png (hc\_png-HilbertCurve-method), 15

hc\_png, HilbertCurve-method  
(hc\_png-HilbertCurve-method), 15

hc\_png-HilbertCurve-method, 15

hc\_points, 25

hc\_points (hc\_points-dispatch), 16

hc\_points, GenomicHilbertCurve-method  
(hc\_points-GenomicHilbertCurve-method), 16

hc\_points, HilbertCurve-method  
(hc\_points-HilbertCurve-method), 17

hc\_points-dispatch, 16

hc\_points-GenomicHilbertCurve-method, 16

hc\_points-HilbertCurve-method, 17

hc\_polygon (hc\_polygon-dispatch), 19

hc\_polygon, GenomicHilbertCurve-method  
(hc\_polygon-GenomicHilbertCurve-method), 19

- 20
- hc\_polygon, HilbertCurve-method  
(hc\_polygon-HilbertCurve-method),  
21
- hc\_polygon-dispatch, 19
- hc\_polygon-GenomicHilbertCurve-method,  
20
- hc\_polygon-HilbertCurve-method, 21
- hc\_rect (hc\_rect-dispatch), 22
- hc\_rect, GenomicHilbertCurve-method  
(hc\_rect-GenomicHilbertCurve-method),  
22
- hc\_rect, HilbertCurve-method  
(hc\_rect-HilbertCurve-method),  
23
- hc\_rect-dispatch, 22
- hc\_rect-GenomicHilbertCurve-method, 22
- hc\_rect-HilbertCurve-method, 23
- hc\_segmented\_points  
(hc\_segmented\_points-HilbertCurve-method),  
24
- hc\_segmented\_points, HilbertCurve-method  
(hc\_segmented\_points-HilbertCurve-method),  
24
- hc\_segmented\_points-HilbertCurve-method,  
24
- hc\_segments (hc\_segments-dispatch), 26
- hc\_segments, GenomicHilbertCurve-method  
(hc\_segments-GenomicHilbertCurve-method),  
26
- hc\_segments, HilbertCurve-method  
(hc\_segments-HilbertCurve-method),  
27
- hc\_segments-dispatch, 26
- hc\_segments-GenomicHilbertCurve-method,  
26
- hc\_segments-HilbertCurve-method, 27
- hc\_text (hc\_text-dispatch), 28
- hc\_text, GenomicHilbertCurve-method  
(hc\_text-GenomicHilbertCurve-method),  
28
- hc\_text, HilbertCurve-method  
(hc\_text-HilbertCurve-method),  
29
- hc\_text-dispatch, 28
- hc\_text-GenomicHilbertCurve-method, 28
- hc\_text-HilbertCurve-method, 29
- HilbertCurve, 4, 31, 34
- HilbertCurve-class, 33
- IRanges, 6, 9, 13, 17, 21, 24, 25, 27, 30
- is\_white, 34
- read.chromInfo, 4
- show (show-HilbertCurve-method), 35
- show, HilbertCurve-method  
(show-HilbertCurve-method), 35
- show-HilbertCurve-method, 35
- unit, 13, 17
- unzoom (unzoom-HilbertCurve-method), 35
- unzoom, HilbertCurve-method  
(unzoom-HilbertCurve-method),  
35
- unzoom-HilbertCurve-method, 35
- zoom (zoom-HilbertCurve-method), 36
- zoom, HilbertCurve-method  
(zoom-HilbertCurve-method), 36
- zoom-HilbertCurve-method, 36