

Package ‘isocubes’

May 8, 2026

Type Package

Title Voxel Data Visualization with Isometric Cubes

Version 1.0.0

Maintainer Mike Cheng <mikefc@coolbutuseless.com>

Description A voxel is a representation of a value on a regular, three-dimensional grid; it is the 3D equivalent of a 2D pixel. Voxel data can be visualised with this package using fixed viewpoint isometric cubes for each data point. This package also provides sample voxel data and tools for transforming the data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports grid, methods

Suggests png, ambient, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

URL <https://github.com/coolbutuseless/isocubes>

BugReports <https://github.com/coolbutuseless/isocubes/issues>

VignetteBuilder knitr

Depends R (>= 4.1.0), colorfast (>= 1.0.1)

LazyData true

LinkingTo colorfast

NeedsCompilation yes

Author Mike Cheng [aut, cre, cph]

Repository CRAN

Date/Publication 2025-09-17 11:00:02 UTC

Contents

calc_heightmap_coords	2
calc_visibility	4
coord_align	5
coord_rotate	6
coord_translate	6
gen_isosurface	7
gen_prism	8
gen_sphere	9
isoaxesGrob	9
isocubesGrob	11
isolinesGrob	12
isopointsGrob	13
obj_letter	14
obj_organic	15
obj_test	15
rand_palette	16

Index	17
--------------	-----------

calc_heightmap_coords *Calculate voxel coordinates from a matrix where values in the matrix indicate height above the ground*

Description

Calculate voxel coordinates from a matrix where values in the matrix indicate height above the ground

Usage

```
calc_heightmap_coords(
  mat,
  fill = NULL,
  scale = 1,
  flipx = FALSE,
  flipy = TRUE,
  ground = "xy",
  solid = TRUE,
  check_visibility = FALSE,
  verbose = FALSE
)
```

Arguments

mat	integer matrix. The matrix will be interpreted as cubes flat on the page, with the value in the matrix interpreted as the height above the page.
fill	matrix of colours the same dimensions as the mat argument. Default: NULL. If fill is not NULL, then a fill column will be included in the final returned coordinates.
scale	scale factor for values in matrix. Default = 1
flipx, flipy	Should the matrix be flipped in the horizontal/vertical directions (respectively)? Default: flipx = FALSE, flipy = TRUE. Note: flipy defaults to TRUE as matrices are indexed from the top-down, but the isometric coordinate space is increasing from the bottom up. Flipping the matrix vertically is usually what you want.
ground	Orientation of the ground plane. Default: 'xy'. Possible values 'xz', 'xy'
solid	Should the heightmap be made 'solid' i.e. without holes? default: TRUE. This can be an expensive operation in terms of both memory and CPU, but should be OK for simple examples. Set to FALSE if things take too long. This operation works by extruding cubes down from the top of the height map to the floor to ensure gaps do not appear when the slope is too great.
check_visibility	Should non-visible cubes be removed? Default: FALSE. If you plan on rotating or manipulating the returned coordinates then this should definitely be FALSE. If TRUE, then non-visible voxels will be entirely removed from the returned coordinates i.e. they will be missing if you change the rendering viewpoint from the default.
verbose	Be verbose? default: FALSE

Value

data.frame of voxel coordinates

Examples

```
# Plot the standard volcano
mat <- volcano

# normalise height
mat <- mat - min(mat)

# Assign a distinct colour for each height value
val <- as.vector(mat)
val <- round(255 * val / max(val))
fill <- matrix("", nrow=nrow(mat), ncol=ncol(mat))
fill[] <- terrain.colors(256)[val + 1L]

# Calculate coordinates of heightmap, render as isocubes
coords <- calc_heightmap_coords(mat, fill = fill, scale = 0.3)
head(coords)
isocubesGrob(coords, size = 2, y = 0) |>
```

```
grid::grid.draw()
```

calc_visibility	<i>Calculate indices of visible voxels when rendered from the specified view.</i>
-----------------	---

Description

Returned value is depth-sorted in back-to-front rendering order

Usage

```
calc_visibility(
  coords,
  xyplane = "flat",
  handedness = "right",
  value = "index",
  verbosity = 0,
  ...
)
```

Arguments

coords	data.frame of x,y,z coordinates for the cubes (integer coordinates)
xyplane	How is the xyplane oriented with respect to the unit isometric cube?. "left", "right", "flat" (or "top"). Default: "flat".
handedness	How is the z-axis positioned with respect to the xy-plane? I.e. is this a right-handed or left-handed coordinate system? Default: "right"
value	type of value to return. Default: 'index'. Valid values are 'index' and 'full'. If 'index', then returns an integer vector of which rows to render in back-to-front ordering. 'full' returns more information in a data.frame
verbosity	Verbosity level. Default: 0
...	other values passed to gpar() to set the graphical parameters e.g. lwd and col for the linewidth and colour of the outline stroke for each cube face.

Value

if value argument is 'index' then integer vector of visible vertices in back-to-front draw ordering. For value = 'full' return a data.frame with more complete information.

Examples

```
obj_sphere <- gen_sphere()
nrow(obj_sphere)
calc_visibility(obj_sphere) |>
  length()

calc_visibility(obj_sphere, value = 'full') |>
  head()
```

coord_align	<i>Align the object with the given coordinates</i>
-------------	--

Description

Align the object with the given coordinates

Usage

```
coord_align(coords, loc = c(0, 0, 0), x = "mean", y = "mean", z = "mean")
```

Arguments

coords	data.frame with 'x', 'y' and 'z' coordinates
loc	location to align to. Default: c(0, 0, 0)
x, y, z	how to align the x coordinates to the given location. Default: 'mean'. Valid values 'min', 'mean', 'max', 'identity', 'median'

Value

data.frame of transformed coordinates

Examples

```
gen_sphere() |>
  coord_align(z = 'max', y = 'min') |>
  isocubesGrob(size = 3) |>
  grid::grid.draw()
```

coord_rotate	<i>Rotate object around a coordinate axis</i>
--------------	---

Description

Rotate object around a coordinate axis

Usage

```
coord_rotate(coords, theta, axis = "z")
```

Arguments

coords	data.frame with 'x', 'y' and 'z' coordinates
theta	angle in radians.
axis	axis to rotate around. Default: 'z'. Valid values: 'x', 'y', 'z'

Value

data.frame of transformed coordinates

Examples

```
obj_letter |>  
  coord_rotate(pi/2, 'y') |>  
  isocubesGrob() |>  
  grid::grid.draw()
```

coord_translate	<i>Translate object</i>
-----------------	-------------------------

Description

Translate object

Usage

```
coord_translate(coords, x = 0, y = 0, z = 0)
```

Arguments

coords	data.frame with 'x', 'y' and 'z' coordinates
x, y, z	amount to translate along each axis. Default: 0

Value

data.frame of transformed coordinates

Examples

```
gen_sphere() |>
  coord_translate(x = 20, z = 40) |>
  isocubesGrob(size = 2) |>
  grid::grid.draw()
```

gen_isosurface	<i>Generate voxel coordinates defined by an implicit function</i>
----------------	---

Description

Generate voxel coordinates defined by an implicit function

Usage

```
gen_isosurface(
  f,
  upper = 0,
  lower = -Inf,
  scale = 1,
  nx = 51,
  ny = nx,
  nz = nx
)
```

Arguments

f	function of the form $f(x, y, z)$ which returns a numeric value
lower, upper	When the supplied function is evaluated, the lower and upper limits define the range of values which will be considered to be inside the object. The default $[-\text{Inf}, 0]$ means that any value less than or equal to zero is inside, and all positive values are outside.
scale	extra scaling factor applied to coordinates before calling function
nx, ny, nz	the dimensions of the volume within which the function will be evaluated

Value

data.frame of coordinates

Examples

```

#-----
# Create a sphere of radius 10
#-----
coords <- gen_isosurface(
  f = function(x, y, z) {x^2 + y^2 + z^2},
  upper = 10^2
)

coords |>
  isocubesGrob() |>
  grid::grid.draw()

#-----
# Create a complex shape
#-----
grid::grid.newpage()
f <- function(x, y, z) {
  (x-2)^2 * (x+2)^2 +
  (y-2)^2 * (y+2)^2 +
  (z-2)^2 * (z+2)^2 +
  3 * (x^2 * y^2 + x^2 * z^2 + y^2 * z^2) +
  6 * x * y * z -
  10 * (x^2 + y^2 + z^2) + 22
}

gen_isosurface(
  f = f,
  scale = 0.1,
  nx = 70
) |>
  isocubesGrob(size = 2) |>
  grid::grid.draw()

```

gen_prism

Generate a rectangular prism centered at the origin

Description

To simplify implementation, only odd side lengths are generated.

Usage

```
gen_prism(x = 5, y = x, z = x)
```

Arguments

x, y, z prism dimensions. Default 5x5x5. Note that due to the quantization of coordinates to integer values for rendering, this function rounds up even dimensions to the next odd number.

Value

data.frame of voxel coordinates

Examples

```
gen_prism(3, 5, 7) |>  
  isocubesGrob() |>  
  grid::grid.draw()
```

gen_sphere	<i>Generate voxel coordinates for a sphere centered at the origin</i>
------------	---

Description

Generate voxel coordinates for a sphere centered at the origin

Usage

```
gen_sphere(r = 10)
```

Arguments

r radius. Default: 10

Value

data.frame of voxel coordinates

Examples

```
gen_sphere(1) |>  
  isocubesGrob() |>  
  grid::grid.draw()
```

isoaxesGrob	<i>Create a grob representing the specified axis orientation.</i>
-------------	---

Description

The x, y and z axes are drawn in red, green and blue respectively.

Usage

```

isoaxesGrob(
  size = 5,
  x = 0.5,
  y = 0.5,
  default.units = "mm",
  xyplane = "flat",
  handedness = "right",
  labels = TRUE,
  verbosity = 0,
  ...
)

```

Arguments

<code>size</code>	length of each axis in <code>default.units</code>
<code>x, y</code>	the origin of the isometric coordinate system. If these values are given as vanilla floating point values, they will be interpreted as 'npc' units, otherwise a valid grid unit object must be supplied. By default the origin is the middle of the graphics device i.e. $(x, y) = (0.5, 0.5)$
<code>default.units</code>	Default unit for size of a cube is 'mm'
<code>xyplane</code>	How is the xyplane oriented with respect to the unit isometric cube?. "left", "right", "flat" (or "top"). Default: "flat".
<code>handedness</code>	How is the z-axis positioned with respect to the xy-plane? I.e. is this a right-handed or left-handed coordinate system? Default: "right"
<code>labels</code>	Include axis labels? Default: TRUE
<code>verbosity</code>	Verbosity level. Default: 0
<code>...</code>	other values passed to <code>gpar()</code> to set the graphical parameters e.g. <code>lwd</code> and <code>col</code> for the linewidth and colour of the outline stroke for each cube face.

Value

grid grob object

Examples

```

isoaxesGrob() |>
  grid::grid.draw()

```

isocubesGrob	<i>Create a grob of isocubes representing the voxels at the given coordinates</i>
--------------	---

Description

Create a grob of isocubes representing the voxels at the given coordinates

Usage

```
isocubesGrob(
  coords,
  fill = NULL,
  fill_left = NULL,
  fill_right = NULL,
  intensity = c(1, 0.3, 0.7),
  size = 5,
  x = 0.5,
  y = 0.5,
  col = "black",
  default.units = "mm",
  xyplane = "flat",
  handedness = "right",
  verbosity = 0,
  ...
)
```

Arguments

<code>coords</code>	data.frame of x,y,z coordinates for the cubes (integer coordinates)
<code>fill</code>	fill colour for the top face of cube. Default: NULL will attempt to use the 'fill' colour in the coords data.frame, otherwise 'grey50'
<code>fill_left, fill_right</code>	fill colours for left and right faces of cube.
<code>intensity</code>	c(1, 0.3, 0.6) Intensity shading for fill for the top, left and right faces respectively. Note: this setting has no effect on the shading of the left face if <code>fill_left</code> has been set explicitly by the user; same for the right face.
<code>size</code>	dimensions of cube i.e. the length of the vertical edge of the cube. Default: 5mm
<code>x, y</code>	the origin of the isometric coordinate system. If these values are given as vanilla floating point values, they will be interpreted as 'npc' units, otherwise a valid grid unit object must be supplied. By default the origin is the middle of the graphics device i.e. $(x, y) = (0.5, 0.5)$
<code>col</code>	Stroke colour for outline of cube faces. Default: black. If NA then no outlines will be drawn. If negative, then outline colour will be the same as the face colour.

default.units	Default unit for size of a cube is 'mm'
xyplane	How is the xyplane oriented with respect to the unit isometric cube?. "left", "right", "flat" (or "top"). Default: "flat".
handedness	How is the z-axis positioned with respect to the xy-plane? I.e. is this a right-handed or left-handed coordinate system? Default: "right"
verbosity	Verbosity level. Default: 0
...	other values passed to gpar() to set the graphical parameters e.g. lwd and col for the linewidth and colour of the outline stroke for each cube face.

Value

grid grob object

Examples

```
obj_sphere <- gen_sphere(r = 10)
fill <- rainbow(nrow(obj_sphere))
isocubesGrob(obj_sphere, fill = fill, size = 2) |>
  grid::grid.draw()

# The 'obj_organic' data.frame includes a 'fill' column which will be
# used by default
grid::grid.newpage()
isocubesGrob(obj_organic, size = 2) |>
  grid::grid.draw()
```

isolinesGrob

Create grob representing isometric grid of lines

Description

Create grob representing isometric grid of lines

Usage

```
isolinesGrob(
  N = 50,
  size = 5,
  x = 0.5,
  y = 0.5,
  col = "black",
  default.units = "mm",
  verbosity = 0,
  ...
)
```

Arguments

N	extents
size	dimensions of cube i.e. the length of the vertical edge of the cube. Default: 5mm
x, y	the origin of the isometric coordinate system. If these values are given as vanilla floating point values, they will be interpreted as 'npc' units, otherwise a valid grid unit object must be supplied. By default the origin is the middle of the graphics device i.e. $(x, y) = (0.5, 0.5)$
col	Stroke colour for outline of cube faces. Default: black. If NA then no outlines will be drawn. If negative, then outline colour will be the same as the face colour.
default.units	Default unit for size of a cube is 'mm'
verbosity	Verbosity level. Default: 0
...	other values passed to <code>gpar()</code> to set the graphical parameters e.g. <code>lwd</code> and <code>col</code> for the linewidth and colour of the outline stroke for each cube face.

Value

isometric line grid

Examples

```
isolinesGrob() |>
  grid::grid.draw()
```

isopointsGrob	<i>Create grob representing isometric grid of points</i>
---------------	--

Description

Create grob representing isometric grid of points

Usage

```
isopointsGrob(
  N = 50,
  size = 5,
  x = 0.5,
  y = 0.5,
  col = "black",
  pch = ".",
  default.units = "mm",
  verbosity = 0,
  ...
)
```

Arguments

N	extents
size	dimensions of cube i.e. the length of the vertical edge of the cube. Default: 5mm
x, y	the origin of the isometric coordinate system. If these values are given as vanilla floating point values, they will be interpreted as 'npc' units, otherwise a valid grid unit object must be supplied. By default the origin is the middle of the graphics device i.e. $(0.5, 0.5)$
col	Stroke colour for outline of cube faces. Default: black. If NA then no outlines will be drawn. If negative, then outline colour will be the same as the face colour.
pch	plotting character. default '.'
default.units	Default unit for size of a cube is 'mm'
verbosity	Verbosity level. Default: 0
...	other values passed to gpar() to set the graphical parameters e.g. lwd and col for the linewidth and colour of the outline stroke for each cube face.

Value

isometric point grid

Examples

```
isopointsGrob(pch = '+') |>
  grid::grid.draw()
```

obj_letter

Voxel coordinates for the letter R

Description

Voxel coordinates for the letter R

Usage

```
obj_letter
```

Format

An object of class `data.frame` with 68 rows and 3 columns.

See Also

Other datasets: [obj_organic](#), [obj_test](#)

Examples

```
head(obj_letter)
isocubesGrob(obj_letter, size = 5, y = 0.05) |>
  grid::grid.draw()
```

obj_organic	<i>Voxel coordinates for an organic shape</i>
-------------	---

Description

Voxel coordinates for an organic shape

Usage

```
obj_organic
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 14292 rows and 4 columns.

See Also

Other datasets: [obj_letter](#), [obj_test](#)

Examples

```
head(obj_organic)
cubes <- isocubesGrob(obj_organic, size = 2) |>
  grid::grid.draw()
```

obj_test	<i>Voxel coordinates for a test object useful for debugging orientation and visibility checks</i>
----------	---

Description

Voxel coordinates for a test object useful for debugging orientation and visibility checks

Usage

```
obj_test
```

Format

An object of class `data.frame` with 16 rows and 4 columns.

See Also

Other datasets: [obj_letter](#), [obj_organic](#)

Examples

```
head(obj_test)
isocubesGrob(obj_test, size = 5, y = 0.05) |>
  grid::grid.draw()
```

rand_palette	<i>Generate a random colour palette</i>
--------------	---

Description

Generate a random colour palette

Usage

```
rand_palette(N = 256, seed = NULL)
```

Arguments

N	number of colors
seed	integer seed. Default: NULL

Value

character vector of colors

Examples

```
rand_palette(N = 20)
```

Index

* datasets

- obj_letter, [14](#)
- obj_organic, [15](#)
- obj_test, [15](#)

- calc_heightmap_coords, [2](#)
- calc_visibility, [4](#)
- coord_align, [5](#)
- coord_rotate, [6](#)
- coord_translate, [6](#)

- gen_isosurface, [7](#)
- gen_prism, [8](#)
- gen_sphere, [9](#)

- isoaxesGrob, [9](#)
- isocubesGrob, [11](#)
- isolinesGrob, [12](#)
- isopointsGrob, [13](#)

- obj_letter, [14](#), [15](#), [16](#)
- obj_organic, [14](#), [15](#), [16](#)
- obj_test, [14](#), [15](#), [15](#)

- rand_palette, [16](#)