Spatial analyses involving binning require that every bin have the same area, but this is impossible using a rectangular grid laid over the Earth or over any projection of the Earth. Discrete global grids use hexagons, triangles, and diamonds to overcome this issue, overlaying the Earth with equally-sized bins. This package provides utilities for working with discrete global grids, along with utilities to aid in plotting such data.
System Requirements: C++11
Repository: CRAN
Date/Publication: 2020-04-29 17:50:02 UTC

### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dgcellstogrid</td>
<td>3</td>
</tr>
<tr>
<td>dgconstruct</td>
<td>4</td>
</tr>
<tr>
<td>dgearthgrid</td>
<td>5</td>
</tr>
<tr>
<td>dgGEO_to_GEO</td>
<td>6</td>
</tr>
<tr>
<td>dgGEO_to_PLANE</td>
<td>7</td>
</tr>
<tr>
<td>dgGEO_to_PROJTRI</td>
<td>8</td>
</tr>
<tr>
<td>dgGEO_to_Q2DD</td>
<td>8</td>
</tr>
<tr>
<td>dgGEO_to_Q2DI</td>
<td>9</td>
</tr>
<tr>
<td>dgGEO_to_SEQNUM</td>
<td>10</td>
</tr>
<tr>
<td>dgetres</td>
<td>11</td>
</tr>
<tr>
<td>dginfo</td>
<td>11</td>
</tr>
<tr>
<td>dgmmaxcell</td>
<td>12</td>
</tr>
<tr>
<td>dgPROJTRI_to_GEO</td>
<td>13</td>
</tr>
<tr>
<td>dgPROJTRI_to_PLANE</td>
<td>13</td>
</tr>
<tr>
<td>dgPROJTRI_to_PROJTRI</td>
<td>14</td>
</tr>
<tr>
<td>dgPROJTRI_to_Q2DD</td>
<td>15</td>
</tr>
<tr>
<td>dgPROJTRI_to_Q2DI</td>
<td>16</td>
</tr>
<tr>
<td>dgPROJTRI_to_SEQNUM</td>
<td>16</td>
</tr>
<tr>
<td>dgQ2DD_to_GEO</td>
<td>17</td>
</tr>
<tr>
<td>dgQ2DD_to_PLANE</td>
<td>18</td>
</tr>
<tr>
<td>dgQ2DD_to_PROJTRI</td>
<td>19</td>
</tr>
<tr>
<td>dgQ2DD_to_Q2DD</td>
<td>19</td>
</tr>
<tr>
<td>dgQ2DD_to_Q2DI</td>
<td>20</td>
</tr>
<tr>
<td>dgQ2DD_to_SEQNUM</td>
<td>21</td>
</tr>
<tr>
<td>dgQ2DI_to_GEO</td>
<td>22</td>
</tr>
<tr>
<td>dgQ2DI_to_PLANE</td>
<td>22</td>
</tr>
<tr>
<td>dgQ2DI_to_PROJTRI</td>
<td>23</td>
</tr>
<tr>
<td>dgQ2DI_to_Q2DD</td>
<td>24</td>
</tr>
<tr>
<td>dgQ2DI_to_Q2DI</td>
<td>25</td>
</tr>
<tr>
<td>dgQ2DI_to_SEQNUM</td>
<td>25</td>
</tr>
<tr>
<td>dgquakes</td>
<td>26</td>
</tr>
<tr>
<td>dgrectgrid</td>
<td>27</td>
</tr>
<tr>
<td>dgsavegrid</td>
<td>28</td>
</tr>
<tr>
<td>dgSEQNUM_to_GEO</td>
<td>28</td>
</tr>
<tr>
<td>dgSEQNUM_to_PLANE</td>
<td>29</td>
</tr>
<tr>
<td>dgSEQNUM_to_PROJTRI</td>
<td>30</td>
</tr>
<tr>
<td>dgSEQNUM_to_Q2DD</td>
<td>30</td>
</tr>
<tr>
<td>dgSEQNUM_to_Q2DI</td>
<td>31</td>
</tr>
<tr>
<td>dgSEQNUM_to_SEQNUM</td>
<td>32</td>
</tr>
<tr>
<td>dgetres</td>
<td>32</td>
</tr>
<tr>
<td>dgshptogrid</td>
<td>33</td>
</tr>
</tbody>
</table>
**dgcellstogrid**

Return boundary coordinates for specified cells

**Description**

Returns the coordinates constituting the boundary of a specified set of cells. Duplicates are eliminated to reduce processing and storage requirements.

**Usage**

```r
dgcellstogrid(dggs, cells, frame = TRUE, wrapcells = TRUE, savegrid = NA)
```

**Arguments**

- `dggs`: A `dggs` object from `dgconstruct()`
- `cells`: The cells to get the boundaries of
- `frame`: If TRUE, return a data frame suitable for ggplot plotting. If FALSE, return an OGR poly object
- `wrapcells`: Cells which cross -180/180 degrees can present difficulties for plotting. Setting this TRUE will result in cells with components in both hemispheres to be mapped entirely to positive degrees (the Eastern hemisphere). As a result, such cells will have components in the range [180,360). Only used when `frame=TRUE`.
- `savegrid`: If savegrid is set to a file path, then a shapefile containing the grid is written to that path and the filename is returned. No other manipulations are done. Default: NA (do not save grid, return it)

**Value**

Returns a data frame or OGR poly object, as specified by `frame`. If `!is.na(savegrid)`, returns a filename.
Examples

```r
library(dggridR)
data(dgquakes)

# Construct a grid with cells about ~1000 miles wide
dggs <- dgconstruct(spacing=1000,metric=FALSE)
dgquakes$cell <- dgtransform(dggs,dgquakes$lat,dgquakes$lon)

# Get grid cells for the earthquakes identified
grid <- dgcellstogrid(dggs, dgquakes$cell, frame=TRUE)
```

---

**dgconstruct**

Construct a discrete global grid system (dggs) object

**Description**

Construct a discrete global grid system (dggs) object

**Usage**

```r
dgconstruct(
  projection = "ISEA",
  aperture = 3,
  topology = "HEXAGON",
  res = NA,
  precision = 7,
  area = NA,
  spacing = NA,
  cls = NA,
  resround = "nearest",
  metric = TRUE,
  show_info = TRUE,
  azimuth_deg = 0,
  pole_lat_deg = 58.28252559,
  pole_lon_deg = 11.25
)
```

**Arguments**

- **projection**
  - Type of grid to use. Options are: ISEA and FULLER. Default: ISEA3H
- **aperture**
  - How finely subsequent resolution levels divide the grid. Options are: 3, 4. Not all options work with all projections and topologies. Default: 3
- **topology**
  - Shape of cell. Options are: HEXAGON, DIAMOND, TRIANGLE. Default: HEXAGON
dgearthgrid

res Resolution. Must be in the range [0,30]. Larger values represent finer resolutions. Appropriate resolutions can be found with dg_closest_res_to_area(), dg_closest_res_to_spacing(), and dg_closest_res_to_cls(). Default is 9, which corresponds to a cell area of ~2600 sq km and a cell spacing of ~50 km. Only one of res, area, length, or cls should be used.

precision Round output to this number of decimal places. Must be in the range [0,30]. Default: 7.

area The desired area of the grid’s cells. Only one of res, area, length, or cls should be used.

spacing The desired spacing between the center of adjacent cells. Only one of res, area, length, or cls should be used.

cls The desired CLS of the cells. Only one of res, area, length, or cls should be used.

resround What direction to search in. Must be nearest, up, or down.

metric Whether input and output should be in metric (TRUE) or imperial (FALSE)

show_info Print the area, spacing, and CLS of the chosen resolution.

azimuth_deg Rotation in degrees of grid about its pole, value in [0,360]. Default=0.

pole_lat_deg Latitude in degrees of the pole, value in [-90,90]. Default=58.28252559.

pole_lon_deg Longitude in degrees of the pole, value in [-180,180]. Default=11.25.

Value

Returns a dggs object which can be passed to other dggridR functions

Examples

library(dggridR)
dggs <- dgconstruct(res=20)
dggs <- dgconstruct(area=5,metric=FALSE)

dgearthgrid

Return the coordinates constituting the boundary of cells for the entire Earth

Description

Note: If you have a high-resolution grid this may take a looooonng time to execute.

Usage

dgearthgrid(dggs, frame = TRUE, wrapcells = TRUE, savegrid = NA)
Arguments

dggs A dggs object from dgconstruct()
frame If TRUE, return a data frame suitable for ggplot plotting. If FALSE, return an OGR poly object
wrapcells Cells which cross -180/180 degrees can present difficulties for plotting. Setting this TRUE will result in cells with components in both hemispheres to be mapped entirely to positive degrees (the Eastern hemisphere). As a result, such cells will have components in the range [180,360). Only used when frame=TRUE.
savegrid If savegrid is set to a file path, then a shapefile containing the grid is written to that path and the filename is returned. No other manipulations are done. Default: NA (do not save grid, return it)

Value

Returns a data frame or OGR poly object, as specified by frame. If !is.na(savegrid), returns a filename.

Examples

library(dgridR)
dggs <- dgconstruct(res=20)
res <- dg_closest_res_to_spacing(dggs, spacing=1000, round=’down’, metric=FALSE)
dggs <- dgsetres(dggs, res)
gridfilename <- dgearthgrid(dggs, savegrid=’temp.shp’) #Save directly to a file

---

dgGEO_to_GEO Convert from GEO to GEO

Description

Uses a discrete global grid system to convert between GEO and GEO (see vignette for details)

Usage

dgGEO_to_GEO(dggs, in_lon_deg, in_lat_deg)

Arguments

dggs A dggs object from dgconstruct()
in_lon_deg Vector of longitude, in degrees
in_lat_deg Vector of latitude, in degrees

Value

Returns a dggs object which can be passed to other dgridR functions
dgGEO_to_PLANE

Examples

## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)

dgGEO_to_GEO(dggs, in_lon_deg, in_lat_deg)

## End(Not run)

dgGEO_to_PLANE

Convert from GEO to PLANE

Description

Uses a discrete global grid system to convert between GEO and PLANE (see vignette for details)

Usage

dgGEO_to_PLANE(dggs, in_lon_deg, in_lat_deg)

Arguments

dggs A dggs object from dgconstruct()
in_lon_deg Vector of longitude, in degrees
in_lat_deg Vector of latitude, in degrees

Value

Returns a dggs object which can be passed to other dggridR functions

Examples

## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)

dgGEO_to_PLANE(dggs, in_lon_deg, in_lat_deg)

## End(Not run)
### dgGEO_to_PROJTRI

*Convert from GEO to PROJTRI*

**Description**

Uses a discrete global grid system to convert between GEO and PROJTRI (see vignette for details)

**Usage**

\[
dgGEO_to_PROJTRI(dggs, \text{in\_lon\_deg}, \text{in\_lat\_deg})
\]

**Arguments**

- **dggs**
  
  A dggs object from dgconstruct()

- **in\_lon\_deg**
  
  Vector of longitude, in degrees

- **in\_lat\_deg**
  
  Vector of latitude, in degrees

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgGEO_to_PROJTRI(dggs, in_lon_deg, in_lat_deg)
## End(Not run)
```

### dgGEO_to_Q2DD

*Convert from GEO to Q2DD*

**Description**

Uses a discrete global grid system to convert between GEO and Q2DD (see vignette for details)

**Usage**

\[
dgGEO_to_Q2DD(dggs, \text{in\_lon\_deg}, \text{in\_lat\_deg})
\]
**dgGEO_to_Q2DI**

**Arguments**

- **dggs** A dggs object from dgconstruct()
- **in_lon_deg** Vector of longitude, in degrees
- **in_lat_deg** Vector of latitude, in degrees

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgGEO_to_Q2DD(dggs, in_lon_deg, in_lat_deg)
## End(Not run)
```

---

**Description**

Uses a discrete global grid system to convert between GEO and Q2DI (see vignette for details)

**Usage**

```r
dgGEO_to_Q2DI(dggs, in_lon_deg, in_lat_deg)
```

**Arguments**

- **dggs** A dggs object from dgconstruct()
- **in_lon_deg** Vector of longitude, in degrees
- **in_lat_deg** Vector of latitude, in degrees

**Value**

Returns a dggs object which can be passed to other dggridR functions
dgGEO_to_SEQNUM

Convert from GEO to SEQNUM

Description

Uses a discrete global grid system to convert between GEO and SEQNUM (see vignette for details)

Usage

dgGEO_to_SEQNUM(dggs, in_lon_deg, in_lat_deg)

Arguments

dggs  A dggs object from dgconstruct()
in_lon_deg  Vector of longitude, in degrees
in_lat_deg  Vector of latitude, in degrees

Value

Returns a dggs object which can be passed to other dggridR functions

Examples

## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgGEO_to_SEQNUM(dggs, in_lon_deg, in_lat_deg)
## End(Not run)
**dggetres**  
*Get table of grid resolution information*

**Description**  
Gets a grid’s resolution and cell property info as a data frame.

**Usage**  
dggetres(dggs)

**Arguments**  
dggs  
A dggs object from dgconstruct()

**Value**  
A data frame containing the resolution levels, number of cells, area of those cells, intercell spacing, and characteristic length scale of the cells. All values are in kilometres.

**Examples**  
library(dggridR)  
dggs <- dgconstruct(res=20)  
dggetres(dggs)

---

**dginfo**  
*Print a buncha info about a dggs object to the screen*

**Description**  
dggs objects have many settings. This returns all of them, along with info about the grid being specified.

**Usage**  
dginfo(dggs)

**Arguments**  
dggs  
A dggs object from dgconstruct()

**Value**  
No return. All info is printed to the screen.
Examples

library(dggridR)
dggs <- dgconstruct(res=20)
dginfo(dggs)

dgmaxcell

Get largest cell id for a dggs

Description

Cells are labeled 1-N. This function returns N. This is useful if you want to choose cells from the 
dggs randomly.

Usage

dgmaxcell(dggs, res = NA)

Arguments

dggs A dggs object from dgconstruct()
res If NA, use the resolution specified by the dggs. Otherwise, override the resolu-
tion.

Value

The maximum cell id.

Examples

#Choose a set of cells randomly distributed over the Earth
library(dggridR)
dggs <- dgconstruct(spacing=1000, metric=FALSE, resround='down')
N <- 100 #Number of cells
maxcell <- dgmaxcell(dggs) #Get maximum cell id
cells <- sample(1:maxcell, N, replace=FALSE) #Choose random cells
grid <- dgcellstogrid(dggs,cells,frame=TRUE,wrapcells=TRUE) #Get grid
**dgPROJTRI_to_GEO**

*Convert from PROJTRI to GEO*

**Description**

Uses a discrete global grid system to convert between PROJTRI and GEO (see vignette for details)

**Usage**

\[
dgPROJTRI_to_GEO(dggs, in_tnum, in_tx, in_ty)
\]

**Arguments**

- **dggs**: A dggs object from dgconstruct()
- **in_tnum**: Vector of triangle numbers
- **in_tx**: Vector of triangle x values
- **in_ty**: Vector of triangle y values

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

## Not run:

```r
library(dggridR)
dggs <- dgconstruct(res=20)
dgPROJTRI_to_GEO(dggs, in_tnum, in_tx, in_ty)
```

## End(Not run)

---

**dgPROJTRI_to_PLANE**

*Convert from PROJTRI to PLANE*

**Description**

Uses a discrete global grid system to convert between PROJTRI and PLANE (see vignette for details)

**Usage**

\[
dgPROJTRI_to_PLANE(dggs, in_tnum, in_tx, in_ty)
\]
**dgPROJTRI_to_PROJTRI**

Convert from PROJTRI to PROJTRI

**Description**

Uses a discrete global grid system to convert between PROJTRI and PROJTRI (see vignette for details)

**Usage**

`dgPROJTRI_to_PROJTRI(dggs, in_tnum, in_tx, in_ty)`

**Arguments**

- **dggs**  
  A dggs object from dgconstruct()
- **in_tnum**  
  Vector of triangle numbers
- **in_tx**  
  Vector of triangle x values
- **in_ty**  
  Vector of triangle y values

**Value**

Returns a dggs object which can be passed to other dggridR functions

---

## Not run:

```r
library(dggridR)
dggs <- dgconstruct(res=20)
dgPROJTRI_to_PLANE(dggs, in_tnum, in_tx, in_ty)
```

## End(Not run)
dgPROJTRI_to_Q2DD

Examples

## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)

dgPROJTRI_to_PROJTRI(dggs, in_tnum, in_tx, in_ty)
## End(Not run)

---

dgPROJTRI_to_Q2DD  
*Convert from PROJTRI to Q2DD*

Description

Uses a discrete global grid system to convert between PROJTRI and Q2DD (see vignette for details)

Usage

dgPROJTRI_to_Q2DD(dggs, in_tnum, in_tx, in_ty)

Arguments

- `dggs`  
  A dggs object from dgconstruct()
- `in_tnum`  
  Vector of triangle numbers
- `in_tx`  
  Vector of triangle x values
- `in_ty`  
  Vector of triangle y values

Value

Returns a dggs object which can be passed to other dggridR functions

Examples

## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)

dgPROJTRI_to_Q2DD(dggs, in_tnum, in_tx, in_ty)
## End(Not run)
dgPROJTRI_to_Q2DI  
*Convert from PROJTRI to Q2DI*

**Description**

Uses a discrete global grid system to convert between PROJTRI and Q2DI (see vignette for details)

**Usage**

```r
dgPROJTRI_to_Q2DI(dggs, in_tnum, in_tx, in_ty)
```

**Arguments**

- `dggs`  
  A dggs object from dgconstruct()
- `in_tnum`  
  Vector of triangle numbers
- `in_tx`  
  Vector of triangle x values
- `in_ty`  
  Vector of triangle y values

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgPROJTRI_to_Q2DI(dggs, in_tnum, in_tx, in_ty)
## End(Not run)
```

dgPROJTRI_to_SEQNUM  
*Convert from PROJTRI to SEQNUM*

**Description**

Uses a discrete global grid system to convert between PROJTRI and SEQNUM (see vignette for details)

**Usage**

```r
dgPROJTRI_to_SEQNUM(dggs, in_tnum, in_tx, in_ty)
```
**dgQ2DD_to_GEO**

**Arguments**

- **dggs** A dggs object from dgconstruct()
- **in_tnum** Vector of triangle numbers
- **in_tx** Vector of triangle x values
- **in_ty** Vector of triangle y values

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgPROJTRI_to_SEQNUM(dggs, in_tnum, in_tx, in_ty)
## End(Not run)
```

---

**dgQ2DD_to_GEO** Convert from Q2DD to GEO

**Description**

Uses a discrete global grid system to convert between Q2DD and GEO (see vignette for details)

**Usage**

`dgQ2DD_to_GEO(dggs, in_quad, in_qx, in_qy)`

**Arguments**

- **dggs** A dggs object from dgconstruct()
- **in_quad** Vector of quad numbers
- **in_qx** Vector of quadrant x values
- **in_qy** Vector of quadrant y values

**Value**

Returns a dggs object which can be passed to other dggridR functions
Examples

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DD_to_GEO(dggs, in_quad, in_qx, in_qy)
## End(Not run)
```

## Convert from Q2DD to PLANE

### Description
Uses a discrete global grid system to convert between Q2DD and PLANE (see vignette for details)

### Usage
```
dgQ2DD_to_PLANE(dggs, in_quad, in_qx, in_qy)
```

### Arguments
- `dggs`: A dggs object from `dgconstruct()`
- `in_quad`: Vector of quad numbers
- `in_qx`: Vector of quadrant x values
- `in_qy`: Vector of quadrant y values

### Value
Returns a dggs object which can be passed to other dggridR functions

### Examples
```
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DD_to_PLANE(dggs, in_quad, in_qx, in_qy)
## End(Not run)
```
**dgQ2DD_to_PROJTRI**  
*Convert from Q2DD to PROJTRI*

**Description**
Uses a discrete global grid system to convert between Q2DD and PROJTRI (see vignette for details)

**Usage**
```
dgQ2DD_to_PROJTRI(dggs, in_quad, in_qx, in_qy)
```

**Arguments**
- `dggs` A dggs object from dgconstruct()
- `in_quad` Vector of quad numbers
- `in_qx` Vector of quadrant x values
- `in_qy` Vector of quadrant y values

**Value**
Returns a dggs object which can be passed to other dggridR functions

**Examples**
```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DD_to_PROJTRI(dggs, in_quad, in_qx, in_qy)
## End(Not run)
```

---

**dgQ2DD_to_Q2DD**  
*Convert from Q2DD to Q2DD*

**Description**
Uses a discrete global grid system to convert between Q2DD and Q2DD (see vignette for details)

**Usage**
```
dgQ2DD_to_Q2DD(dggs, in_quad, in_qx, in_qy)
```
**Arguments**

- `dggs`: A dggs object from dgconstruct()
- `in_quad`: Vector of quad numbers
- `in_qx`: Vector of quadrant x values
- `in_qy`: Vector of quadrant y values

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DD_to_Q2DI(dggs, in_quad, in_qx, in_qy)
## End(Not run)
```

---

**Description**

Uses a discrete global grid system to convert between Q2DD and Q2DI (see vignette for details)

**Usage**

`dgQ2DD_to_Q2DI(dggs, in_quad, in_qx, in_qy)`
**dgQ2DD_to_SEQNUM**  

Convert from Q2DD to SEQNUM

**Description**

Uses a discrete global grid system to convert between Q2DD and SEQNUM (see vignette for details)

**Usage**

```r
dgQ2DD_to_SEQNUM(dggs, in_quad, in_qx, in_qy)
```

**Arguments**

- `dggs`: A dggs object from dgconstruct()
- `in_quad`: Vector of quad numbers
- `in_qx`: Vector of quadrant x values
- `in_qy`: Vector of quadrant y values

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DD_to_Q2DI(dggs, in_quad, in_qx, in_qy)
## End(Not run)
```
dgQ2DI_to_GEO  

Convert from Q2DI to GEO

Description

Uses a discrete global grid system to convert between Q2DI and GEO (see vignette for details)

Usage

dgQ2DI_to_GEO(dggs, in_quad, in_i, in_j)

Arguments

dggs A dggs object from dgconstruct()
in_quad Vector of quad numbers
in_i Vector of quadrant i values
in_j Vector of quadrant j values

Value

Returns a dggs object which can be passed to other dggridR functions

Examples

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DI_to_GEO(dggs, in_quad, in_i, in_j)
## End(Not run)
```

dgQ2DI_to_PLANE  

Convert from Q2DI to PLANE

Description

Uses a discrete global grid system to convert between Q2DI and PLANE (see vignette for details)

Usage

dgQ2DI_to_PLANE(dggs, in_quad, in_i, in_j)
dgQ2DI_to_PROJTRI

Arguments

dggs  A dggs object from dgconstruct()
in_quad  Vector of quad numbers
in_i  Vector of quadrant i values
in_j  Vector of quadrant j values

Value

Returns a dggs object which can be passed to other dggridR functions

Examples

## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DI_to_PLANE(dggs, in_quad, in_i, in_j)
## End(Not run)

dgQ2DI_to_PROJTRI  Convert from Q2DI to PROJTRI

Description

Uses a discrete global grid system to convert between Q2DI and PROJTRI (see vignette for details)

Usage

dgQ2DI_to_PROJTRI(dggs, in_quad, in_i, in_j)

Arguments

dggs  A dggs object from dgconstruct()
in_quad  Vector of quad numbers
in_i  Vector of quadrant i values
in_j  Vector of quadrant j values

Value

Returns a dggs object which can be passed to other dggridR functions
dgQ2DI_to_Q2DD

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DI_to_Q2DD(dggs, in_quad, in_i, in_j)
## End(Not run)
```

---

**dgQ2DI_to_Q2DD**

*Convert from Q2DI to Q2DD*

**Description**

Uses a discrete global grid system to convert between Q2DI and Q2DD (see vignette for details)

**Usage**

```r
dgQ2DI_to_Q2DD(dggs, in_quad, in_i, in_j)
```

**Arguments**

- `dggs` A dggs object from `dgconstruct()`
- `in_quad` Vector of quad numbers
- `in_i` Vector of quadrant i values
- `in_j` Vector of quadrant j values

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DI_to_Q2DD(dggs, in_quad, in_i, in_j)
## End(Not run)
```
**dgQ2DI_to_Q2DI**

*Convert from Q2DI to Q2DI*

---

**Description**

Uses a discrete global grid system to convert between Q2DI and Q2DI (see vignette for details)

**Usage**

```r
dgQ2DI_to_Q2DI(dggs, in_quad, in_i, in_j)
```

**Arguments**

- `dggs`: A dggs object from dgconstruct()
- `in_quad`: Vector of quad numbers
- `in_i`: Vector of quadrant i values
- `in_j`: Vector of quadrant j values

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DI_to_Q2DI(dggs, in_quad, in_i, in_j)
## End(Not run)
```

---

**dgQ2DI_to_SEQNUM**

*Convert from Q2DI to SEQNUM*

---

**Description**

Uses a discrete global grid system to convert between Q2DI and SEQNUM (see vignette for details)

**Usage**

```r
dgQ2DI_to_SEQNUM(dggs, in_quad, in_i, in_j)
```

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DI_to_SEQNUM(dggs, in_quad, in_i, in_j)
## End(Not run)
```
Arguments

- `dggs`: A dggs object from dgconstruct()
- `in_quad`: Vector of quad numbers
- `in_i`: Vector of quadrant i values
- `in_j`: Vector of quadrant j values

Value

Returns a dggs object which can be passed to other dggridR functions

Examples

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgQ2DI_to_SEQNUM(dggs, in_quad, in_i, in_j)
## End(Not run)
```

---

dgquakes  

All earthquakes with magnitude >=3.0 earthquakes for 2015

Description

A data frame with 19914 observations on the following 4 variables.

- `lat`: Latitude of the epicenter. Example: -7.0711
- `lon`: Longitude of the epicenter. Example: -173.5178
- `mag`: Magnitude of the quake. Example: 3.2

Usage

```r
data(dgquakes)
```

Format

- data frame

Source

The USGS Earthquake Hazards Program (http://earthquake.usgs.gov/earthquakes/).
**dgrectgrid**

Return the coordinates constituting the boundary of cells within a specified region

---

**Description**

Note: This may generate odd results for very large rectangles, because putting rectangles on spheres is weird... as you should know, if you’re using this package.

**Usage**

```r
dgrectgrid(
  dggs,
  minlat = -1,
  minlon = -1,
  maxlat = -1,
  maxlon = -1,
  cellsize = 0.1,
  frame = TRUE,
  wrapcells = TRUE,
  savegrid = NA
)
```

**Arguments**

- **dggs**: A dggs object from dgconstruct()
- **minlat**: Minimum latitude of region of interest
- **minlon**: Minimum longitude of region of interest
- **maxlat**: Maximum latitude of region of interest
- **maxlon**: Maximum longitude of region of interest
- **cellsize**: Distance, in degrees, between the sample points used to generate the grid. Small values yield long generation times while large values may omit cells.
- **frame**: If TRUE, return a data frame suitable for ggplot plotting. If FALSE, return an OGR poly object
- **wrapcells**: Cells which cross -180/180 degrees can present difficulties for plotting. Setting this TRUE will result in cells with components in both hemispheres to be mapped entirely to positive degrees (the Eastern hemisphere). As a result, such cells will have components in the range [180,360). Only used when frame=TRUE.
- **savegrid**: If savegrid is set to a file path, then a shapefile containing the grid is written to that path and the filename is returned. No other manipulations are done. Default: NA (do not save grid, return it)

**Value**

Returns a data frame or OGR poly object, as specified by frame. If !is.na(savegrid), returns a filename.
Examples

```r
library(dggridR)
dggs <- dgconstruct(spacing=100, metric=FALSE, resround='down')

# Get grid cells for the conterminous United States
grid <- dgrectgrid(dggs,
                   minlat=24.7433195, minlon=-124.7844079,
                   maxlat=49.3457868, maxlon=-66.9513812, frame=TRUE)
```

---

```r
dgsavegrid  
Saves a generated grid to a shapefile

Description

Saves a generated grid to a shapefile

Usage

dgsavegrid(grid, shpfname)

Arguments

grid   
Grid to be saved

shpfname   
File to save the grid to

Value

The filename the grid was saved to

---

```r
dgSEQNUM_to_GEO  
Convert from SEQNUM to GEO

Description

Uses a discrete global grid system to convert between SEQNUM and GEO (see vignette for details)

Usage

dgSEQNUM_to_GEO(dggs, in_seqnum)

Arguments

dggs   
A dggs object from dgconstruct()

in_seqnum   
Globally unique number identifying the surface polygon
Value

Returns a dggs object which can be passed to other dggridR functions

Examples

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgSEQNUM_to_GEO(dggs, in_seqnum)
## End(Not run)
```

---

**Description**

Uses a discrete global grid system to convert between SEQNUM and PLANE (see vignette for details)

**Usage**

```r
dgSEQNUM_to_PLANE(dggs, in_seqnum)
```

**Arguments**

- `dggs` A dggs object from dgconstruct()
- `in_seqnum` Globally unique number identifying the surface polygon

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgSEQNUM_to_PLANE(dggs, in_seqnum)
## End(Not run)
```
dgSEQNUM_to_PROJTRI  

Convert from SEQNUM to PROJTRI

Description

Uses a discrete global grid system to convert between SEQNUM and PROJTRI (see vignette for details)

Usage

dgSEQNUM_to_PROJTRI(dggs, in_seqnum)

Arguments

dggs  A dggs object from dgconstruct()
in_seqnum  Globally unique number identifying the surface polygon

Value

Returns a dggs object which can be passed to other dggridR functions

Examples

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgSEQNUM_to_PROJTRI(dggs, in_seqnum)
## End(Not run)
```

dgSEQNUM_to_Q2DD  

Convert from SEQNUM to Q2DD

Description

Uses a discrete global grid system to convert between SEQNUM and Q2DD (see vignette for details)

Usage

dgSEQNUM_to_Q2DD(dggs, in_seqnum)
**dgSEQNUM_to_Q2DI**

**Arguments**

- `dggs`: A dggs object from dgconstruct()
- `in_seqnum`: Globally unique number identifying the surface polygon

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgSEQNUM_to_Q2DI(dggs, in_seqnum)
## End(Not run)
```

---

**Description**

Uses a discrete global grid system to convert between SEQNUM and Q2DI (see vignette for details)

**Usage**

`dgSEQNUM_to_Q2DI(dggs, in_seqnum)`

**Arguments**

- `dggs`: A dggs object from dgconstruct()
- `in_seqnum`: Globally unique number identifying the surface polygon

**Value**

Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgSEQNUM_to_Q2DI(dggs, in_seqnum)
## End(Not run)
```
**dgSEQNUM_to_SEQNUM**  
*Convert from SEQNUM to SEQNUM*

**Description**  
Uses a discrete global grid system to convert between SEQNUM and SEQNUM (see vignette for details)

**Usage**  
dgSEQNUM_to_SEQNUM(dggs, in_seqnum)

**Arguments**  
- **dggs**: A dggs object from dgconstruct()
- **in_seqnum**: Globally unique number identifying the surface polygon

**Value**  
Returns a dggs object which can be passed to other dggridR functions

**Examples**

```r
## Not run:
library(dggridR)
dggs <- dgconstruct(res=20)
dgSEQNUM_to_SEQNUM(dggs, in_seqnum)
## End(Not run)
```

---

**dgsetres**  
*Set the resolution of a dggs object*

**Description**  
Set the resolution of a dggs object

**Usage**  
dgsetres(dggs, res)


**Arguments**

- **dggs**: A dggs object from `dgconstruct()`.
- **res**: Resolution. Must be in the range $[0,30]$. Larger values represent finer resolutions. Appropriate resolutions can be found with `dg_closest_res_to_area()`, `dg_closest_res_to_spacing()`, and `dg_closest_res_to_cls()`. Default is 9, which corresponds to a cell area of ~2600 sq km and a cell spacing of ~50 km. Default: 9.

**Value**

Returns a dggs object which can be passed to other dggridR functions.

**Examples**

```r
library(dggridR)
dggs <- dgconstruct(res=20)
dggs <- dgsetres(dggs,10)
```

---

**dgshptogrid**  
*Return boundary coordinates for cells intersecting a shapefile*

**Description**

Returns the coordinates constituting the boundary of a set of cells which intersect or are contained by a polygon (or polygons) specified in a shapefile. Note that grid cells are also generated for holes in the shapefile's polygon(s).

Note that coordinates in the shapefile must be rounded to check polygon intersections. Currently this round preserves eight decimal digits of precision.

The eighth decimal place is worth up to 1.1 mm of precision: this is good for charting the motions of tectonic plates and the movements of volcanoes. Permanent, corrected, constantly-running GPS base stations might be able to achieve this level of accuracy.

In other words: you should be just fine with this level of precision.

**Usage**

```r
dgshptogrid(
  dggs,
  shpfname,
  cellsize = 0.1,
  frame = TRUE,
  wrapcells = TRUE,
  savegrid = NA
)
```
Arguments

dggs  A dggs object from dgconstruct()
shpname  File name of the shapefile. Filename should end with `.shp`
cells  Distance, in degrees, between the sample points used to generate the grid. Small values yield long generation times while large values may omit cells.
frame  If TRUE, return a data frame suitable for ggplot plotting. If FALSE, return an OGR poly object
wrapcells  Cells which cross -180/180 degrees can present difficulties for plotting. Setting this TRUE will result in cells with components in both hemispheres to be mapped entirely to positive degrees (the Eastern hemisphere). As a result, such cells will have components in the range [180,360). Only used when frame=TRUE.
savegrid  If savegrid is set to a file path, then a shapefile containing the grid is written to that path and the filename is returned. No other manipulations are done. Default: NA (do not save grid, return it)

Value

Returns a data frame or OGR poly object, as specified by frame. If !is.na(savegrid), returns a filename.

Examples

library(dggridR)

dggs <- dgconstruct(spacing=25, metric=FALSE, resround='nearest')
south_africa_grid <- dgshptogrid(dggs,dg_shpfname_south_africa())

dgtransform  (DEPRECATED) Converts lat-long pairs into discrete global grid cell numbers

Description

A discrete global grid maps lat-long points to particular cells. These cells are uniquely numbered, for a given resolution, from 1 to some maximum number. Cell numbers may be reused from one resolution to the next. THIS FUNCTION IS DEPRECATED.

Usage

dgtransform(dggs, lat, lon)

Arguments

dggs  A dggs object from dgconstruct().
lat  A vector of latitudes. Same length at the longitudes
lon  A vector of longitudes. Same length as the latitudes.
**dgverify**

**Value**

A vector of the same length as latitudes and longitudes containing the cell id numbers of the points’
cells in the discrete grid.

**Examples**

```r
library(dggridR)
data(dgquakes)

#Construct a grid with cells about ~1000 miles wide
dggs <- dgconstruct(spacing=1000,metric=FALSE)
dgquakes$cell <- dgtransform(dggs,dgquakes$lat,dgquakes$lon)
```

**Description**

Verify that a dggs object has appropriate values

**Usage**

`dgverify(dggs)`

**Arguments**

- `dggs` The dggs object to be verified

**Value**

The function has no return value. A stop signal is raised if the object is misspecified

**Examples**

```r
library(dggridR)
dggs <- dgconstruct(res=20)
dgverify(dggs)
```
**dg_closest_res**  
*Determine an appropriate grid resolution based on input data.*

**Description**

This is a generic function that is used to determine an appropriate resolution given an area, cell spacing, or correlated length scale. It does so by extracting the appropriate length/area column and searching it for a value close to the input.

**Usage**

```r
dg_closest_res(
  dggs,  
  col,  
  val,  
  round = "nearest",  
  show_info = TRUE,  
  metric = TRUE
)
```

**Arguments**

- **dggs**: A dggs object from dgconstruct()
- **col**: Column in which to search for a close value. Should be: area_km, spacing_km, or cls_km.
- **val**: The value to search for
- **round**: What direction to search in. Must be nearest, up, or down.
- **show_info**: Print the area, spacing, and CLS of the chosen resolution.
- **metric**: Whether input and output should be in metric (TRUE) or imperial (FALSE)

**Value**

A number representing the grid resolution

**Examples**

```r
library(dggridR)
dggs <- dgconstruct(res=20)
res <- dg_closest_res(dggs,'area_km',1)
dggs <- dgsetres(dggs,res)
```
**dg_closest_res_to_area**

*Determine resolution based on desired area*

**Description**

Determine an appropriate grid resolution based on a desired cell area.

**Usage**

```
dg_closest_res_to_area(
  dggs,
  area,
  round = "nearest",
  show_info = TRUE,
  metric = TRUE
)
```

**Arguments**

- **dggs**: A dggs object from dgconstruct()
- **area**: The desired area of the grid’s cells
- **round**: What direction to search in. Must be nearest, up, or down.
- **show_info**: Print the area, spacing, and CLS of the chosen resolution.
- **metric**: Whether input and output should be in metric (TRUE) or imperial (FALSE)

**Value**

A number representing the grid resolution

**Examples**

```
library(dggridR)
dggs <- dgconstruct(res=20)
res <- dg_closest_res_to_area(dggs,1)
dggs <- dgsetres(dggs,res)
```
**dg_closest_res_to_cls**  
Determine an appropriate grid resolution based on a desired characteristic length scale of the cells.

**Description**

The characteristic length scale (CLS) is the diameter of a spherical cap of the same area as a cell of the specified resolution.

**Usage**

```r
dg_closest_res_to_cls(
  dggs,
  cls,
  round = "nearest",
  show_info = TRUE,
  metric = TRUE
)
```

**Arguments**

- `dggs`: A dggs object from dgconstruct()
- `cls`: The desired CLS of the cells.
- `round`: What direction to search in. Must be nearest, up, or down.
- `show_info`: Print the area, spacing, and CLS of the chosen resolution.
- `metric`: Whether input and output should be in metric (TRUE) or imperial (FALSE)

**Value**

A number representing the grid resolution

**Examples**

```r
library(dggridR)
dggs <- dgconstruct(res=20)
res <- dg_closest_res_to_cls(dggs,1)
dggs <- dgsetres(dggs,res)
```
**dg_closest_res_to_spacing**

_Determine grid resolution from desired spacing._

**Description**

Determine an appropriate grid resolution based on a desired spacing between the center of adjacent cells.

**Usage**

```r
dg_closest_res_to_spacing(
    dggs,  # A dggs object from dgconstruct()
    spacing,  # The desired spacing between the center of adjacent cells
    round = "nearest",  # What direction to search in. Must be nearest, up, or down.
    show_info = TRUE,  # Print the area, spacing, and CLS of the chosen resolution.
    metric = TRUE  # Whether input and output should be in metric (TRUE) or imperial (FALSE)
)
```

**Arguments**

- `dggs`: A dggs object from dgconstruct()
- `spacing`: The desired spacing between the center of adjacent cells
- `round`: What direction to search in. Must be nearest, up, or down.
- `show_info`: Print the area, spacing, and CLS of the chosen resolution.
- `metric`: Whether input and output should be in metric (TRUE) or imperial (FALSE)

**Value**

A number representing the grid resolution

**Examples**

```r
library(dggridR)
dggs <- dgconstruct(res=20)
res <- dg_closest_res_to_spacing(dggs,1)
dggs <- dgsetres(dggs,res)
```
**dg_env**

*Control global aspects of the dggridR package*

**Description**

This environment is used to control global features of the dggridR package. At the moment the only option is 'dg_debug' which, when set to TRUE provides extensive outputs useful for tracking down bugs.

**Usage**

```r
dg_env
```

**Format**

An object of class `environment` of length 1.

---

**dg_process_polydata**

*Load a KML file*

**Description**

Convert data from internal dggrid functions into something useful: an sp object or a data frame

**Usage**

```r
dg_process_polydata(polydata, frame, wrapcells)
```

**Arguments**

- `polydata` Polygons generated by dggrid. These will be converted.
- `frame` If TRUE, return a data frame suitable for ggplot plotting. If FALSE, return an `SpatialPolygons`
- `wrapcells` Cells which cross -180/180 degrees can present difficulties for plotting. Setting this TRUE will result in cells with components in both hemispheres to be mapped entirely to positive degrees (the Eastern hemisphere). As a result, such cells will have components in the range [180,360). Only used when frame=TRUE.

**Value**

Returns a data frame or OGR poly object, as specified by `frame`
dg_shpfname_south_africa

National border of South Africa

Description
This variable points to a shapefile containing the national border of South Africa

Usage
dg_shpfname_south_africa()

Value
A filename of a shapefile containing the national border of South Africa
Index

- *Topic* datasets
  
  - dg_env, 40
  - dgquakes, 26
  
  - dg_closest_res, 36
  - dg_closest_res_to_area, 37
  - dg_closest_res_to_clsl, 38
  - dg_closest_res_to_spacing, 39
  - dg_env, 40
  
  - dg_process_polydata, 40
  - dg_shpfname_south_africa, 41
  - dgcellstogrid, 3
  - dgconstruct, 4
  - dgearthgrid, 5
  - dgGEO_to_GEO, 6
  - dgGEO_to_PLANE, 7
  - dgGEO_to_ProjTri, 8
  - dgGEO_to_Q2DD, 8
  - dgGEO_to_Q2DI, 9
  - dgGEO_to_SEQNUM, 10
  - dggetres, 11
  - dginfo, 11
  - dgmaxcell, 12
  - dgProjTri_to_GEO, 13
  - dgProjTri_to_PLANE, 13
  - dgProjTri_to_ProjTri, 14
  - dgProjTri_to_Q2DD, 15
  - dgProjTri_to_Q2DI, 16
  - dgProjTri_to_SEQNUM, 16
  - dgQ2DD_to_GEO, 17
  - dgQ2DD_to_PLANE, 18
  - dgQ2DD_to_ProjTri, 19
  - dgQ2DD_to_Q2DD, 19
  - dgQ2DD_to_Q2DI, 20
  - dgQ2DD_to_SEQNUM, 21
  - dgQ2DI_to_GEO, 22
  - dgQ2DI_to_PLANE, 22
  - dgQ2DI_to_ProjTri, 23
  - dgQ2DI_to_Q2DD, 24
  - dgQ2DI_to_Q2DI, 25
  - dgq2DI_to_SEQNUM, 25
  - dgquakes, 26
  - dgrectgrid, 27
  - dgsavegrid, 28
  - dgSEQNUM_to_GEO, 28
  - dgSEQNUM_to_PLANE, 29
  - dgSEQNUM_to_ProjTri, 30
  - dgSEQNUM_to_Q2DD, 30
  - dgSEQNUM_to_Q2DI, 31
  - dgSEQNUM_to_SEQNUM, 32
  - dgsetres, 32
  - dgshptogrid, 33
  - dgtransform, 34
  - dgverify, 35