

# Package ‘antarcticR’

July 1, 2017

**Title** Visualisation tools for Antarctica, including some clustering methods

**Version** 0.0.0.900

**Author** Luke Batten [aut, cre]

**Maintainer** Luke Batten <lukeb@physics.org>

**Description** This package mainly contains functions to plot longitude-latitude points onto the Antarctic continent. antarcticR can convert CSV files into data frames for plotting, or into Haversine distance matrices for clustering. The results can be combined and visualised on the bottom of a globe, or using the BEDMAP2 data.

**Depends** R (>= 3.3.3)

**License** None currently

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1.9000

**Honourable mentions** John Russell (base list spreadsheet producer)

## R topics documented:

clusterResult . . . . .	2
csvToDF . . . . .	2
csvToHaversineMat . . . . .	3
drawAntarctica . . . . .	3
drawBedmap . . . . .	4
genCartesianMat . . . . .	4
genHaversineMat . . . . .	5
longLatToSimpleBEDMAP . . . . .	5
plotAntarctica . . . . .	6
<b>Index</b>	<b>7</b>

---

clusterResult	<i>A function to use some clustering methods from the dbscan package</i>
---------------	--

---

**Description**

A function to use some clustering methods from the dbscan package

**Usage**

```
clusterResult(haversineMatrix, eps = 2e+05, minPts, eps_cl)
```

---

csvToDF	<i>Turn a longitude, latitude csv file into a dataframe</i>
---------	---

---

**Description**

Generate a dataframe from a longitude-latitude csv file

**Usage**

```
csvToDF(csvFile)
```

**Arguments**

csvFile	Your csv file
---------	---------------

**Value**

A dataframe

**Examples**

```
df <- csvToHaversineMat("myData.csv")
```

---

csvToHaversineMat	<i>A function to generate a Haversine matrix from a csv file</i>
-------------------	--

---

**Description**

Generate a distance matrix of great-circle distances from a csv file with longitude and latitude distances

**Usage**

```
csvToHaversineMat(csvFile)
```

**Arguments**

csvFile	Your csv file
---------	---------------

**Value**

A haversine distance matrix

**Examples**

```
mat <- csvToHaversineMat("myData.csv")
```

---

drawAntarctica	<i>Set up the drawing of a map of Antarctica</i>
----------------	--

---

**Description**

Set up the drawing of a map of Antarctica

**Usage**

```
drawAntarctica()
```

**Examples**

```
world3 <- drawAntarctica()  
world3
```

---

drawBedmap

*Set up the drawing of a map of Antarctica BEDMAP2*


---

### Description

Set up the drawing of a map of Antarctica BEDMAP2

### Usage

```
drawBedmap(BEDMAP_GRAD = "thickness", reduceResolutionBy = 5)
```

### Examples

```
world3 <- drawAntarctica()
world3
```

---

genCartesianMat

*A function to generate a Cartesian matrix from a dataframe*


---

### Description

Generate a distance matrix of x-y-z distances from a dataframe with longitude and latitude points

### Usage

```
genCartesianMat(df)
```

### Arguments

df                      Your data frame

### Value

A Cartesian distance matrix

### Examples

```
points <- read.csv("dividedEvents1.csv",header=T, sep=",")
df.points <- as.matrix(points)
antFrame = data.frame(df.points)
print("Computing distance matrix...")
d <- genCartesianMatrix(antFrame)
```

---

genHaversineMat	<i>A function to generate a Haversine matrix from a dataframe</i>
-----------------	---

---

**Description**

Generate a distance matrix of great-circle distances from a dataframe with longitude and latitude distances

**Usage**

```
genHaversineMat(df)
```

**Arguments**

df	Your data frame
----	-----------------

**Value**

A haversine distance matrix

**Examples**

```
points <- read.csv("dividedEvents1.csv",header=T, sep=",")
df.points <- as.matrix(points)
antFrame = data.frame(df.points)
print("Computing distance matrix...")
require(geosphere)
d <- genHaversineMat(antFrame)
```

---

longLatToSimpleBEDMAP	<i>A function to convert from lon/lat to the BEDMAP grid</i>
-----------------------	--

---

**Description**

A function to convert from lon/lat to the BEDMAP grid

**Usage**

```
longLatToSimpleBEDMAP(longLatDataFrame)
```

**Value**

bedmapFrame

---

plotAntarctica	<i>Plot points on the antarctic map</i>
----------------	---

---

**Description**

Plot points on the antarctic map

**Usage**

```
plotAntarctica(antMap, df, clusterPlot = FALSE, selfClusterPlot = FALSE,  
  pointSize = 2, shapes = TRUE, newSetPlot = 0, BEDMAP = FALSE,  
  BEDMAP_GRAD = "thickness", reduceResolutionBy = 5)
```

**Arguments**

antMap	your map made from drawAntarctica
df	Your lon/lat data frame

**Examples**

```
world4 <- plotAntarctica(map, dataframe)  
world4
```

# Index

- \*Topic **Antarctica**,
  - clusterResult, [2](#)
  - drawAntarctica, [3](#)
  - drawBedmap, [4](#)
  - longLatToSimpleBEDMAP, [5](#)
  - plotAntarctica, [6](#)
- \*Topic **BEDMAP**
  - longLatToSimpleBEDMAP, [5](#)
- \*Topic **Cartesian**,
  - genCartesianMat, [4](#)
- \*Topic **Haversine**,
  - csvToHaversineMat, [3](#)
  - genHaversineMat, [5](#)
- \*Topic **csv**,
  - csvToHaversineMat, [3](#)
- \*Topic **dataframe**,
  - genCartesianMat, [4](#)
  - genHaversineMat, [5](#)
- \*Topic **dataframe**
  - csvToDF, [2](#)
- \*Topic **distance**
  - csvToHaversineMat, [3](#)
  - genCartesianMat, [4](#)
  - genHaversineMat, [5](#)
- \*Topic **draw**,
  - clusterResult, [2](#)
  - drawAntarctica, [3](#)
  - drawBedmap, [4](#)
  - longLatToSimpleBEDMAP, [5](#)
  - plotAntarctica, [6](#)
- \*Topic **matrix**,
  - csvToHaversineMat, [3](#)
  - genCartesianMat, [4](#)
  - genHaversineMat, [5](#)
- \*Topic **plot**,
  - longLatToSimpleBEDMAP, [5](#)
- \*Topic **plotx**
  - plotAntarctica, [6](#)
- \*Topic **plot**
  - clusterResult, [2](#)
  - drawAntarctica, [3](#)
  - drawBedmap, [4](#)
  - clusterResult, [2](#)
  - csvToDF, [2](#)
  - csvToHaversineMat, [3](#)
  - drawAntarctica, [3](#)
  - drawBedmap, [4](#)
  - genCartesianMat, [4](#)
  - genHaversineMat, [5](#)
  - longLatToSimpleBEDMAP, [5](#)
  - plotAntarctica, [6](#)