

# Package ‘rcosmo’

February 14, 2020

**URL** <https://github.com/frycast/rcosmo>

**BugReports** <https://github.com/frycast/rcosmo/issues>

**Title** Cosmic Microwave Background Data Analysis

**Version** 1.1.2

**Description** Handling and Analysing Spherical,  
HEALPix and Cosmic Microwave Background data on a HEALPix grid.

**Depends** R (>= 3.4.0)

**License** GPL-3 | file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** FITSio (>= 2.1-0), Rcpp (>= 0.12.11), mmap (>= 0.6-17), tibble  
(>= 1.4.2), rgl (>= 0.99.16), cli (>= 1.0.0), entropy (>=  
1.2.1), geoR (>= 1.7-5.2.1), nnls (>= 1.4)

**Suggests** knitr, rmarkdown, testthat, R.rsp, gsl

**LinkingTo** Rcpp

**RoxygenNote** 6.1.1

**NeedsCompilation** yes

**Author** Daniel Fryer [aut, cre] (<<https://orcid.org/0000-0001-6032-0522>>),  
Andriy Olenko [aut] (<<https://orcid.org/0000-0002-0917-7000>>),  
Ming Li [aut] (<<https://orcid.org/0000-0002-1218-2804>>),  
Yuguang Wang [aut]

**Maintainer** Daniel Fryer <d.fryer@latrobe.edu.au>

**Repository** CRAN

**Date/Publication** 2020-02-14 10:50:02 UTC

## R topics documented:

ancestor . . . . .	4
areCompatibleCMBDFs . . . . .	4

as.CMBDataFrame	5
assumedConvex	6
assumedUniquePix	7
baseNeighbours	8
cbind.CMBDataFrame	8
chi2CMB	9
children	10
CMBDat	10
CMBDataFrame	11
CMBWindow	13
coords	14
coords.CMBDataFrame	14
coords.CMBWindow	15
coords.data.frame	16
coords.HPDataFrame	17
corrCMB	18
covCMB	20
covmodelCMB	21
covPwSp	23
displayPixelBoundaries	24
displayPixels	25
downloadCMBMap	26
downloadCMBPS	27
entropyCMB	29
exprob	30
extrCMB	31
fmf	32
fRen	33
geo2sph	34
geoAngle	34
geoArea	35
geoArea.CMBDataFrame	36
geoArea.CMBWindow	36
geoArea.HPDataFrame	37
geoDist	38
header	39
healpixCentered	39
HPDataFrame	40
ibp2p	43
is.CMBDat	43
is.CMBDataFrame	44
is.CMBWindow	45
is.HPDataFrame	45
linesCMB	46
maxDist	47
maxWindowDist	48
minDist	49
neighbours	50

nest2ring	51
nestSearch	51
nside	53
nside.CMBDataFrame	53
nside.HPDataFrame	54
numeric2col	54
ordering	55
ordering.CMBDataFrame	55
ordering.HPDataFrame	56
p2bp	57
p2ibp	57
parent	58
pix	58
pix.CMBDataFrame	59
pix.HPDataFrame	59
pix2coords	60
pixelArea	61
pixelWindow	61
plot.CMBCorrelation	62
plot.CMBCovariance	63
plot.CMBDataFrame	64
plot.CMBWindow	65
plot.HPDataFrame	66
plot.variogram	67
plotAngDis	68
plotcovmodelCMB	69
plotvariogram	70
practicalRangeCMB	71
print.CMBDataFrame	72
print.HPDataFrame	72
pwSpCorr	73
qqnormWin	74
qqplotWin	75
qstat	76
rbind.CMBDataFrame	77
rcosmo	78
resolution	79
ring2nest	79
sampleCMB	80
siblings	81
sphericalHarmonics	81
summary.CMBDataFrame	82
summary.CMBWindow	83
summary.HPDataFrame	84
triangulate	84
variofitCMB	85
variogramCMB	87
window	89

window.CMBDat . . . . .	89
window.CMBDataFrame . . . . .	90
window.data.frame . . . . .	93
window.HPDataFrame . . . . .	95
winType . . . . .	96

<b>Index</b>	<b>98</b>
--------------	-----------

---

ancestor	<i>Return index of kth ancestor pixel</i>
----------	---

---

### Description

Gives the pixel at resolution  $j - k$  that contains  $p$ , where  $p$  is specified at resolution  $j$  (notice it does not depend on  $j$ ).

### Usage

```
ancestor(p, k)
```

### Arguments

<code>p</code>	A pixel index specified in nested order.
<code>k</code>	A generation of an ancestor pixel.

### Examples

```
ancestor(4,2)
ancestor(17,2)
```

---

areCompatibleCMBDFs	<i>Check compatibleness of CMBDataFrames</i>
---------------------	--

---

### Description

Compare attributes to decide if two CMBDataFrames are compatible

### Usage

```
areCompatibleCMBDFs(cmbdf1, cmbdf2, compare.pix = FALSE)
```

**Arguments**

cmbdf1	a <a href="#">CMBDataFrame</a>
cmbdf2	a <a href="#">CMBDataFrame</a>
compare.pix	A boolean. If TRUE then cmbdf1 and cmbdf2 must share the same pixel indices to be considered compatible

**Details**

If the CMBDataFrames do not have compatible attributes then a message is printed indicating the attributes that do not match. To suppress this use the [suppressMessages](#) function

**Examples**

```
a <- CMBDataFrame(nside = 2, ordering = "ring", coords = "cartesian")
b <- CMBDataFrame(nside = 1, ordering = "nested", coords = "spherical")
areCompatibleCMBDFs(a,b)

suppressMessages(areCompatibleCMBDFs(a,b))
```

---

as.CMBDataFrame      *Convert objects to CMBDataFrame*

---

**Description**

Safely converts a [data.frame](#) to a CMBDataFrame. The rows of the data.frame are assumed to be in the HEALPix order given by ordering, and at the HEALPix resolution given by nside. Coordinates, if present, are assumed to correspond to HEALPix pixel centers. The coordinates must be named either x,y,z (cartesian) or theta, phi (spherical colatitude and longitude respectively). If df is a HPDataFrame then it is possible that df has attribute healpixCentered = TRUE, in which case as.CMBDataFrame will perform HEALPix centering of coordinates.

**Usage**

```
as.CMBDataFrame(df, ordering, nside, spix, drop.coords = FALSE)
```

**Arguments**

df	Any data.frame or HPDataFrame whose rows are in HEALPix order
ordering	character string that specifies the ordering scheme ("ring" or "nested")
nside	an integer $2^k$ that specifies the Nside (resolution) HEALPix parameter
spix	an integer vector that specifies the HEALPix pixel index corresponding to each row of df. If spix is left blank and df is a data.frame, then df is assumed to contain data for every pixel at resolution parameter nside (the full sky). In other words, in this case, the number of rows of df must be equal to $12 * nside^2$ . However, if spix is left blank and df is a CMBDataFrame, then spix is set equal to pix(df)

`drop.coords` A logical. If `df` is a `HPDataFrame` then it is possible that `df` has attribute `healpixCentered = TRUE`, in which case `as.CMBDataFrame` will perform HEALPix centering of coordinates. If `drop.coords = TRUE` then this will be done by dropping existing coordinates entirely (quicker) to rely only on HEALPix pixel indices. Otherwise if `drop.coords = FALSE` this will be done by replacing existing coordinates with locations of HEALPix pixel centers.

## Value

A `CMBDataFrame`

## Examples

```
## Example 1: Create df with no coords, then create CMBDataFrames cmbdf and
## df2 with spherical coords
```

```
df <- data.frame(I=rnorm(12))
df
```

```
cmbdf <- as.CMBDataFrame(df,ordering= "ring", nside=1)
summary(cmbdf)
pix(cmbdf)
coords(cmbdf)
```

```
df2 <- coords(cmbdf, new.coords = "spherical")
df2
```

```
## Example 2: Create CMBDataFrames for first 10 Healpix centers
```

```
df <- data.frame(I=rnorm(10))
df
cmbdf <- as.CMBDataFrame(df,ordering= "ring", nside=2, spix=1:10)
summary(cmbdf)
pix(cmbdf)
```

---

`assumedConvex`

*Check if a [CMBWindow](#) is assumed convex.*

---

## Description

Initially any [CMBWindow](#) is not assumed convex. The `assumedConvex` attribute can be change for any [CMBWindow](#).

## Usage

```
assumedConvex(win, assume.convex)
```

**Arguments**

`win` a CMBWindow object  
`assume.convex` optionally change the assumedConvex attribute to TRUE or FALSE

**Examples**

```
win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
assumedConvex(win1)
win2 <- assumedConvex(win1, assume.convex = TRUE)
assumedConvex(win2)
assumedConvex(win1) <- TRUE
assumedConvex(win1)
```

---

assumedUniquePix	<i>Check if object was assumed to have unique HEALPix indices</i>
------------------	---

---

**Description**

The function checks object's attribute assumedUniquePix. The attribute is True if the object was assumed to have rows that correspond to unique HEALPix pixel indices.

**Usage**

```
assumedUniquePix(obj)
```

**Arguments**

`obj` Any object.

**Value**

A boolean. This is TRUE if `obj` is a [CMBDataFrame](#) or a [HPDataFrame](#) whose rows were assumed to correspond to unique HEALPix pixel indices.

**Examples**

```
hp1 <- HPDataFrame(I=rnorm(5), nside = 1, spix = c(1,1,2,2,3))
pix(hp1)
coords(hp1, new.coords = "cartesian")
assumedUniquePix(hp1)

sky <- CMBDataFrame(nside = 32, coords = "cartesian", ordering = "nested")
sky.s <- CMBDataFrame(sky, sample.size = 100)
hpdf <- HPDataFrame(sky.s, auto.spix = TRUE)
assumedUniquePix(hpdf)
```

---

baseNeighbours	<i>Return neighbours of base pixels</i>
----------------	---

---

### Description

The base-resolution comprises twelve pixels. `baseNeighbours` returns a map from the base pixel index `bp` to the vector of base pixels that are neighbours of `bp`, in counterclockwise order of direction: S,SE,E,NE,N,NW,W,SW. The presence of -1 indicates that the corresponding direction is empty.

### Usage

```
baseNeighbours(bp)
```

### Arguments

<code>bp</code>	The base pixel index
-----------------	----------------------

### Examples

```
## Return neighbours of base pixel 1
baseNeighbours(1)

## There is no base pixel 14, so baseNeighbours returns NULL
baseNeighbours(14)
```

---

cbind.CMBDataFrame	<i>cbind for CMBDataFrames</i>
--------------------	--------------------------------

---

### Description

Add a new column or columns (vector, matrix or data.frame) to a `CMBDataFrame`. Note that method dispatch occurs on the first argument. So, the `CMBDataFrame` must be the first argument

### Usage

```
## S3 method for class 'CMBDataFrame'
cbind(..., deparse.level = 1)
```

### Arguments

<code>...</code>	(generalized) vectors or matrices. Columns to bind.
<code>deparse.level</code>	Integer controlling the construction of labels in the case of non-matrix-like arguments.



## Details

See the documentation for [cbind](#)

## Examples

```
cmbdf <- CMBDataFrame(nside = 1, ordering = "nested", coords = "spherical")
cmbdf2 <- cbind(cmbdf, myData = rep(1, 12))
cmbdf2
```

---

chi2CMB	<i>Chi-squared statistic for two <a href="#">CMBWindows</a></i>
---------	---

---

## Description

This function returns the empirical chi-squared statistic for intensities observations from two [CMBWindows](#) of the specified [CMBDataFrame](#). The functions employs the function [chi2.empirical](#) and uses histogram counts of intensities for computations.

## Usage

```
chi2CMB(cmbdf, win1, win2, intensities = "I")
```

## Arguments

cmbdf	A <a href="#">CMBDataFrame</a> .
win1	A <a href="#">CMBWindow</a>
win2	A <a href="#">CMBWindow</a>
intensities	A <a href="#">CMBDataFrame</a> column with measured values.

## Value

Estimated Chi-squared statistic for observations in two [CMBWindows](#). For small sample sizes and many zero counts Chi-squared statistic is inefficient.

## References

[chi2.empirical](#)

**Examples**

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 1000)
#
# win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
# win2 <- CMBWindow(theta = c(pi/2,pi,pi/2), phi = c(0,0,pi/2))
# plot(win1)
# plot(win2,col="green")
#
# chi2CMB(cmbdf, win1, win2)
```

---

children	<i>Return children of a pixel</i>
----------	-----------------------------------

---

**Description**

Gives four pixels at resolution  $j + 1$  that are contained in  $p$ , where  $p$  is a pixel specified at resolution  $j$  (notice it does not depend on  $j$ ).

**Usage**

```
children(p)
```

**Arguments**

`p` A pixel index specified in nested order.

**Examples**

```
children(11)
```

---

CMBDat	<i>CMBDat class</i>
--------	---------------------

---

**Description**

The function CMBDat creates objects of class CMBDat. These are lists containing header information and other metadata as well as an element called data, whose columns may include, for example, the intensity (I), polarisation (Q, U), PMASK and TMASK. It also may contain an `mmap` object that points to the CMB map data table in the FITS file.

**Usage**

```
CMBDat(filename, mmap = FALSE, spix)
```

**Arguments**

filename	The path to the fits file.
mmap	A boolean indicating whether to use memory mapping.
spix	The sample pixels (rows) to read from the FITS file binary data table (optional)

**Value**

A list containing header information and other metadata as well as an element called data where: If `mmap = FALSE` then a `data.frame` is included, named `data`, whose columns may include, for example, the intensity (I), polarisation (Q, U), PMASK and TMASK. If `mmap = TRUE` then a [mmap](#) object is returned that points to the CMB map data table in the FITS file.

**Examples**

```
## Ensure you have a FITS file with the correct path
## before running the example:
## download a FITS file and use real data
# downloadCMBMap()
# cmbdat <- CMBDat("CMB_map_smica1024.fits", mmap = TRUE)
# class(cmbdat)
# str(cmbdat)

## View metadata
# cmbdat$header1
# cmbdat$header2
# cmbdat$resoln
# cmbdat$method
# cmbdat$coordsys
# cmbdat$nside
# cmbdat$hdr
```

---

CMBDataFrame

*CMBDataFrame class*


---

**Description**

The function `CMBDataFrame` creates objects of class `CMBDataFrame`. These are a special type of [data.frame](#) that carry metadata about, e.g., the HEALPix ordering scheme, coordinate system, and `nside` parameter.

**Usage**

```
CMBDataFrame(CMBData, coords, win, include.polar = FALSE,
             include.masks = FALSE, spix, sample.size, nside, ordering, I, ...)
```

**Arguments**

CMBData	Can be a string location of FITS file, another CMBDataFrame, a CMBData object, a <a href="#">HPDataFrame</a> or unspecified.
coords	Can be "spherical," "cartesian", or unspecified (HEALPix only).
win	optional <a href="#">CMBWindow</a> object that specifies a spherical polygon within which to subset the full sky CMB data.
include.polar	TRUE if polarisation data is required, otherwise FALSE.
include.masks	TRUE if TMASK and PMASK are required, otherwise FALSE.
spix	Optional vector of sample pixel indices, or a path to a file containing comma delimited sample pixel indices. The ordering scheme is given by ordering. If ordering is unspecified then CMBData must be either a CMBDataFrame or a FITS file and the ordering scheme is then assumed to match that of CMBData.
sample.size	If a positive integer is given, a simple random sample of size equal to sample.size will be taken from CMBData. If spix is specified then sample.size must be unspecified.
nside	Optionally specify the nside parameter manually $nside=2^k$ (usually 1024 or 2048).
ordering	Specifies the desired HEALPix ordering scheme ("ring" or "nested") for the output CMBDataFrame. If ordering is unspecified then the ordering scheme will be taken from the CMBData object. If ordering information cannot be found the default will be "nested". This parameter also specifies the ordering scheme of spix.
I	A vector of intensities to be included if CMBData is unspecified. Note that $\text{length}(I)$ must equal $12 * nside^2$ if either spix or sample.size are unspecified.
...	Optional names data columns of length $\text{nrow}(CMBData)$ to add to the CMBDataFrame.

**Value**

A CMBDataFrame whose row.names attribute contains HEALPix indices.

**Examples**

```
## Method 1: Read the data while constructing the CMBDataFrame
## download a FITS file and use real data
# downloadCMBMap()
# df <- CMBDataFrame("CMB_map_smica1024.fits")
df <- CMBDataFrame(nside = 16, I = rnorm(12 * 16 ^ 2),
                  ordering = "nested")
```

```

# Specify a sample size for a random sample
df.sample <- CMBDataFrame(df, sample.size = 80)
plot(df.sample)

# Specify a vector of pixel indices using spix
df.subset <- CMBDataFrame(df, spix = c(2,4,6))

# Take a look at the summary
summary(df)

# Access HEALPix pixel indices using pix function
# (these are stored in the row.names attribute)
pix(df.subset)

```

---

CMBWindow

*CMBWindow class.*


---

## Description

The function `CMBWindow` creates objects of class `CMBWindow`. It is either a polygon or a disc type.

## Usage

```
CMBWindow(..., r, set.minus = FALSE, assume.convex = FALSE)
```

## Arguments

<code>...</code>	these arguments are compulsory and must be labelled either <code>x</code> , <code>y</code> , <code>z</code> (cartesian) or <code>theta</code> , <code>phi</code> (spherical, colatitude and longitude respectively). Alternatively, a single data.frame may be passed in with columns labelled <code>x</code> , <code>y</code> , <code>z</code> or <code>theta</code> , <code>phi</code> .
<code>r</code>	if a disc type window is required then this specifies the radius of the disc
<code>set.minus</code>	when TRUE the window will be the unit sphere minus the window specified
<code>assume.convex</code>	when TRUE the window is assumed to be convex resulting in a faster computation time when the window is used with functions such as <code>subWindow</code> . This argument is irrelevant when the window is not a polygon

## Details

If `r` is unspecified then the rows of `...` correspond to counter-clockwise ordered vertices defining a spherical polygon lying entirely within one open hemisphere on the unit sphere. Counter-clockwise is understood from the perspective outside the sphere, facing the hemisphere that contains the polygon, looking toward the origin. Note that there must be at least 3 rows (vertices) to define a polygon (we exclude bygones). On the other hand, if `r` is specified then `...` must specify just one row, and this row is taken to be the center of a disc of radius `r`

**Examples**

```
win <- CMBWindow(theta = c(pi/2,pi/2,pi/3, pi/3), phi = c(0,pi/3,pi/3,0))
plot(win)

## Create a disc type window
win1<- CMBWindow(x=0,y=3/5,z=4/5,r=0.8, set.minus =TRUE)
plot(win1)

## Apply a disc type window to CMBDataFrame
cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian", ordering = "nested")
window(cmbdf) <- CMBWindow(x=0,y=3/5,z=4/5,r=0.8, set.minus =TRUE)
plot(cmbdf)
```

---

coords	<i>Coordinate conversion generic</i>
--------	--------------------------------------

---

**Description**

Detailed descriptions and and examples can be found in documentation for specific coords functions [coords.CMBDataFrame](#), [coords.CMBWindow](#), [coords.HPDataFrame](#), [coords.data.frame](#)

**Usage**

```
coords(x, ...)
```

**Arguments**

x	An object.
...	Unused arguments.

**See Also**

[coords.CMBDataFrame](#) [coords.CMBWindow](#) [coords.HPDataFrame](#) [coords.data.frame](#)

---

coords.CMBDataFrame	<i>Coordinate system from a CMBDataFrame</i>
---------------------	--

---

**Description**

If new.coords is unspecified then this function returns the coordinate system used in the CMB-DataFrame cmbdf. The coordinate system is either "cartesian" or "spherical". If a new coordinate system is specified, using e.g. new.coords = "spherical", then this function instead returns a new CMBDataFrame whose coordinates are of the specified type. The original CMBDataFrame, cmbdf, is unaffected. If you would like to change cmbdf without creating a new variable, then use [coords<- .CMBDataFrame](#) (see examples below).

**Usage**

```
## S3 method for class 'CMBDataFrame'
coords(x, new.coords, ...)
```

**Arguments**

x	A CMBDataFrame, cmbdf.
new.coords	Specifies the new coordinate system ("spherical" or "cartesian") if a change of coordinate system is desired.
...	Unused arguments.

**Value**

If new.coords is unspecified, then the name of the coordinate system of cmbdf is returned. Otherwise a new CMBDataFrame is returned equivalent to cmbdf but having the desired change of coordinates

**Examples**

```
## Create df with no coords, then create df2 with cartesian coords
df <- CMBDataFrame(nside = 16)
df
coords(df)
df2 <- coords(df, new.coords = "cartesian")
coords(df2)

## Change the coords of df directly (to spherical)
coords(df) <- "spherical"
coords(df)
```

---

coords.CMBWindow      *Coordinate system from a CMBWindow*

---

**Description**

This function returns the coordinate system used in a [CMBWindow](#). The coordinate system is either "cartesian" or "spherical"

**Usage**

```
## S3 method for class 'CMBWindow'
coords(x, new.coords, ...)
```

**Arguments**

x	a CMBWindow, win.
new.coords	specifies the new coordinate system ("spherical" or "cartesian") if a change of coordinate system is desired
...	Unused arguments.

**Details**

If a new coordinate system is specified, using e.g. new.coords = "spherical", the coordinate system of the CMBWindow will be converted

**Value**

If new.coords is unspecified, then the name of the coordinate system of win is returned. Otherwise a new CMBWindow is returned equivalent to win but having the desired change of coordinates

**Examples**

```
## Create win with sperical coords, then change it to win1 with cartesian coords
win <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
coords(win)
win1 <- coords(win, new.coords = "cartesian")
coords(win1)

## Change back to spherical coordinates

coords(win1) <- "spherical"
coords(win1)
```

---

coords.data.frame      *Create a new data.frame with a given coordinate system*

---

**Description**

This does not affect the original object unless new coordinate system is directly assigned.

**Usage**

```
## S3 method for class 'data.frame'
coords(x, new.coords, ...)
```

**Arguments**

x	a data.frame with columns labelled x, y, z (for cartesian) or theta, phi (for spherical colatitude and longitude respectively)
new.coords	specifies the new coordinate system ("spherical" or "cartesian").
...	Unused arguments.



**Value**

A new data.frame whose coordinates are as specified by new.coords

**Examples**

```
## Create df with no coords, then create df2 with spherical coords
df <- data.frame(x = c(1,0,0), y = c(0,1,0), z = c(0,0,1))
df

df2 <- coords(df, new.coords = "spherical")
df2

## The function coords does not affect the original object.
## To change the coords assign a new value ("spherical or "cartesian")

coords(df, new.coords = "spherical")
df
coords(df) <- "spherical"
df
```

---

coords.HPDataFrame      *Coordinate system from a HPDataFrame*

---

**Description**

Add or change coordinates in a [HPDataFrame](#). This does not affect the argument object hpdf. Instead it returns a new [HPDataFrame](#) with the desired coordinates. To change hpdf directly see [coords<- .HPDataFrame](#).

**Usage**

```
## S3 method for class 'HPDataFrame'
coords(x, new.coords, healpixCentered = FALSE, ...)
```

**Arguments**

x	a HPDataFrame, hpdf.
new.coords	specifies the new coordinate system ("spherical" or "cartesian")
healpixCentered	boolean. If TRUE then columns x,y,z or theta, phi will be ignored and removed if present. This forces the coordinates to be found from HEALPix pixel indices only. Then the HEALPixCentered attribute of hpdf will be set to TRUE.
...	Unused arguments.

**Details**

If columns exist labelled x,y,z (cartesian) or theta, phi (colatitude and longitude respectively), then these will be treated as the coordinates of hpdf and converted accordingly. If columns x,y,z or theta,phi are not present then the healpix pixel indices as given by pix(hpdf) are used for assigning coordinates.

**Value**

A `HPDataFrame` with columns x,y,z (cartesian) or theta, phi (colatitude and longitude respectively)

**Examples**

```
df <- HPDataFrame(I = rep(0,12), nside = 1)
coords(df, new.coords = "cartesian")
# Notice that df is unchanged
df

# Instead, change df directly
coords(df) <- "spherical"

## specify cartesian coordinates then convert to spherical
hp1 <- HPDataFrame(x = c(1,0,0), y = c(0,1,0), z = c(0,0,1),
                  nside = 1, auto.spix = TRUE)
hp1 <- coords(hp1, new.coords = "spherical")

## Instead, ignore/drop existing coordinates and use HEALPix only
hp2 <- HPDataFrame(x = c(1,0,0), y = c(0,1,0), z = c(0,0,1),
                  nside = 1, auto.spix = TRUE)
hp2 <- coords(hp1, new.coords = "spherical", healpixCentered = TRUE)
```

---

 corrCMB

*Sample correlation function*


---

**Description**

This function provides an empirical correlation function for data in a `CMBDataFrame` or `data.frame`. It assumes that data are from a stationary spherical random field and the correlation depends only on a geodesic distance between locations. Output is a binned correlation.

**Usage**

```
corrCMB(cmbdf, num.bins = 10, sample.size, max.dist = pi, breaks,
        equiareal = TRUE, calc.max.dist = FALSE)
```

**Arguments**

<code>cmbdf</code>	is a <a href="#">CMBDataFrame</a> or <a href="#">data.frame</a>
<code>num.bins</code>	specifies the number of bins
<code>sample.size</code>	optionally specify the size of a simple random sample to take before calculating correlation. This may be useful if the full correlation computation is too slow.
<code>max.dist</code>	an optional number between 0 and pi specifying the maximum geodesic distance to use for calculating correlation. Only used if breaks are unspecified.
<code>breaks</code>	optionally specify the breaks manually using a vector giving the break points between cells. This vector has length <code>num.bins</code> since the last break point is taken as <code>max.dist</code> . If <code>equiareal = TRUE</code> then these breaks should be $\cos(r_i)$ where $r_i$ are radii. If <code>equiareal = FALSE</code> then these breaks should be $r_i$ .
<code>equiareal</code>	if TRUE then the bins have equal spherical area. If false then the bins have equal annular widths. Default is TRUE.
<code>calc.max.dist</code>	if TRUE then the <code>max.dist</code> will be calculated from the locations in <code>cmbdf</code> . Otherwise either <code>max.dist</code> must be specified or <code>max.dist</code> will default to pi.

**Value**

#' An object of the class `CMBCorrelation` that is a modification of [variog](#) from the package **geoR** with variogram values replaced by correlation.

The attribute "breaks" contains the break points used to create bins. The result has `num.bins + 1` values since the first value at distance 0 is not counted as a bin.

**u** a vector with distances.

**v** a vector with estimated correlation values at distances given in **u**.

**n** number of pairs in each bin

**sd** standard deviation of the values in each bin

**bins.lim** limits defining the interval spanned by each bin

**ind.bin** a logical vector indicating whether the number of pairs in each bin is greater or equal to the value in the argument `pairs.min`

**var.mark** variance of the data

**beta.ols** parameters of the mean part of the model fitted by ordinary least squares

**output.type** echoes the option argument

**max.dist** maximum distance between pairs allowed in the correlation calculations

**n.data** number of data

**direction** direction for which the correlation was computed

**call** the function call

**References**

**geoR** package, [variog](#), [variogramCMB](#), [covCMB](#)

## Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 100000)
# corcmb <- corrCMB(cmbdf, max.dist = 0.03, num.bins = 10, sample.size=1000)
# corcmb
```

---

 covCMB

*Sample covariance function*


---

## Description

This function provides an empirical covariance function for data in a [CMBDataFrame](#) or `data.frame`. It assumes that data are from a stationary spherical random field and the covariance depends only on a geodesic distance between locations. Output is a binned covariance.

## Usage

```
covCMB(cmbdf, num.bins = 10, sample.size, max.dist = pi, breaks,
       equiareal = TRUE, calc.max.dist = FALSE)
```

## Arguments

<code>cmbdf</code>	is a <a href="#">CMBDataFrame</a> or <code>data.frame</code>
<code>num.bins</code>	specifies the number of bins
<code>sample.size</code>	optionally specify the size of a simple random sample to take before calculating covariance. This may be useful if the full covariance computation is too slow.
<code>max.dist</code>	an optional number between 0 and $\pi$ specifying the maximum geodesic distance to use for calculating covariance. Only used if <code>breaks</code> are unspecified.
<code>breaks</code>	optionally specify the breaks manually using a vector giving the break points between cells. This vector has length <code>num.bins</code> since the last break point is taken as <code>max.dist</code> . If <code>equiareal = TRUE</code> then these breaks should be $\cos(r_i)$ where $r_i$ are radii. If <code>equiareal = FALSE</code> then these breaks should be $r_i$ .
<code>equiareal</code>	if <code>TRUE</code> then the bins have equal spherical area. If <code>false</code> then the bins have equal annular widths. Default is <code>TRUE</code> .
<code>calc.max.dist</code>	if <code>TRUE</code> then the <code>max.dist</code> will be calculated from the locations in <code>cmbdf</code> . Otherwise either <code>max.dist</code> must be specified or <code>max.dist</code> will default to $\pi$ .

**Value**

An object of the class `CMB`Covariance that is a modification of `variog` from the package `geoR` with variogram values replaced by covariances.

The attribute "breaks" contains the break points used to create bins. The result has `num.bins + 1` values since the first value, the sample variance, is not counted as a bin.

**u** a vector with distances.

**v** a vector with estimated covariance values at distances given in `u`.

**n** number of pairs in each bin

**sd** standard deviation of the values in each bin

**bins.lim** limits defining the interval spanned by each bin

**ind.bin** a logical vector indicating whether the number of pairs in each bin is greater or equal to the value in the argument `pairs.min`

**var.mark** variance of the data

**beta.ols** parameters of the mean part of the model fitted by ordinary least squares

**output.type** echoes the option argument

**max.dist** maximum distance between pairs allowed in the covariance calculations

**n.data** number of data

**direction** direction for which the covariance was computed

**call** the function call

**References**

`geoR` package, `variog`, `variogramCMB`, `corrCMB`

**Examples**

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 100000)
# Cov <- covCMB(cmbdf, max.dist = 0.03, num.bins = 10)
# Cov
```

**Description**

This function computes the covariances given the separation distance of locations. Options for different covariance functions on spheres are available. The function extends the function `cov.spatial` for additional covariance models on spheres.

**Usage**

```
covmodelCMB(obj, cov.model = "matern",
  cov.pars = stop("no cov.pars argument provided"), kappa = 0.5)
```

**Arguments**

obj	Vector of distances between pairs of spatial locations.
cov.model	A type of the correlation function. Available choices are: "matern", "exponential", "spherical", "powered.exponential", "cauchy", "gencauchy", "pure.nugget", "askey", "c2wendland", "c4wendland", "sinpower", "multiquadric". Default is "matern"
cov.pars	A vector with two covariance parameters. The first parameter corresponds to the variance $\sigma^2$ . The second parameter corresponds to the range $\phi$ of the correlation function.
kappa	A smoothness parameter of the correlation function.

**Details**

The function returns the value of the covariance  $C(h)$  at the distance  $h$ . The covariance function has the form

$$C(h) = \sigma^2 * \rho(h/\phi).$$

The parameters of the covariance are positive numbers  $\sigma^2$ ,  $\phi$  and  $\kappa$ .

Expressions for the correlation functions which are not included in the package **geoR**:

**askey**

$$\rho(h/\phi) = (1 - h/\phi)^{\kappa}, \text{ if } h < \phi;$$

$$0, \text{ otherwise.}$$

**c2wendland**

$$\rho(h/\phi) = (1 + \kappa * h/\phi) * (1 - h/\phi)^{\kappa}, \text{ if } h < \phi;$$

$$0, \text{ otherwise.}$$

**c4wendland**

$$\rho(h/\phi) = (1 + \kappa * h/\phi + (\kappa^2 - 1) * (h/\phi)^2 / 3) * (1 - h/\phi)^{\kappa}, \text{ if } h < \phi;$$

$$0, \text{ otherwise.}$$

**sinpower**

$$\rho(h/\phi) = 1 - (\sin(h/(2\phi)))^{\kappa}$$

**multiquadric**

$$C(h) = (1 - \phi)^{(2*\kappa)} / (1 + \phi^2 - 2 * \phi * \cos(h))^{\kappa}, 0 < \phi < 1$$

Additional information can be found in the section Details in [cov.spatial](#).

**Value**

Values of a covariance function for the given distances.

**References**

**geoR** package, [cov.spatial](#)

T. Gneiting. Strictly and non-strictly positive definite functions on spheres. *Bernoulli* 19 (2013), no. 4, 1327-1349.

**Examples**

```
## Compute Askey variogram at x = pi/4

1 - covmodelCMB(pi/4, cov.pars = c(1, pi), kappa = 3, cov.model = "askey" )

## Plot of the Askey covariance function

v1.f <- function(x, ...) {covmodelCMB(x, ...)}
curve( v1.f(x, cov.pars = c(1, 0.03), kappa = 3, cov.model = "askey"),
       from = 0, to = 0.1, xlab = "distance", ylab = expression(cov(h)), lty = 2,
       main = "covariance")
```

---

 covPwSp

*Covariance estimate via power spectra*


---

**Description**

This function provides a covariance estimate using the values of the estimated power spectra.

**Usage**

```
covPwSp(PowerSpectra, Ns)
```

**Arguments**

PowerSpectra	A data frame which first column lists values of multipole moments and the second column gives the corresponding values of CMB power spectra.
Ns	A number of points in which the covariance estimate is computed on the interval [-1,1]

**Value**

A data frame which first column is 1-d grid starting at  $-1+1/Ns$  and finishing at 1 with the step  $2/Ns$ . The second column is the values of estimated covariances on this grid.

## References

Formula (2.1) in Baran A., Terdik G. Power spectrum estimation of spherical random fields based on covariances. *Annales Mathematicae et Informaticae* 44 (2015) pp. 15–22.

Power Spectra data are from Planck Legacy Archive <http://pla.esac.esa.int/pla/#cosmology>

## Examples

```
## Download the power spectrum first
# N <- 20000
# COM_PowerSpectra <- downloadCMBPS(link=1)
#
# Cov_est <- covPwSp(COM_PowerSpectra[,1:2], N)
# plot(Cov_est, type="l")

## Plot the covariance estimate as a function of angular distances
# plot(acos(Cov_est[,1]), Cov_est[,2], type="l",
#       xlab="angular distance", ylab="Estimated Covariance")
```

---

displayPixelBoundaries

*Plot HEALPix pixel boundaries*

---

## Description

Plot the HEALPix pixel boundaries at nside

## Usage

```
displayPixelBoundaries(nside, eps = pi/90, col = "gray", lwd = 1,
  ordering, incl.labels = 1:(12 * nside^2), nums.col = col,
  nums.size = 1, font = 2, depth_test = "always", ...)
```

## Arguments

nside	the HEALPix nside parameter (integer number $2^k$ )
eps	controls the smoothness of the plot, smaller eps implies more samples
col	the colour of plotted boundary lines
lwd	the thickness of the plotted boundary lines
ordering	optionally specify an ordering scheme from which to plot HEALPix pixel numbers. Can be either "ring" or "nested"
incl.labels	If ordering is specified then this parameter sets the pixel indices that will be displayed (default is all indices at nside)
nums.col	specifies the colour of pixel numbers if ordering is specified



nums.size	specifies the size of pixel numbers if ordering is specified
font	A numeric font number from 1 to 5, used if ordering is specified
depth_test	The depth test to be applied. This controls how resistant the plotted object is to being obscured. See <a href="#">rgl.material</a>
...	arguments passed to <code>rgl::plot3d</code>

**Value**

Produces a plot of the HEALPix pixel boundaries.

**Examples**

```
displayPixelBoundaries(1, eps = pi/90, col = "red")
displayPixelBoundaries(2, eps = pi/90, col = "green")
```

---

displayPixels	<i>Display the pixels and grandchildren</i>
---------------	---

---

**Description**

Display the pixels `spix` at resolution `j` by colouring in the grandchildren of `spix` at resolution `plot.j`

**Usage**

```
displayPixels(boundary.j, j, plot.j = 5, spix, boundary.col = "gray",
             boundary.lwd = 1, incl.labels = 1:(12 * 4^boundary.j),
             col = "blue", size = 3)
```

**Arguments**

<code>boundary.j</code>	The resolution to display boundaries at. If this is missing then boundaries will not be plotted.
<code>j</code>	The resolution that <code>spix</code> are specified at.
<code>plot.j</code>	The resolution to plot grandchildren at
<code>spix</code>	Integer vector. The pixel indices to display. These must be in nested order.
<code>boundary.col</code>	The boundary colour.
<code>boundary.lwd</code>	The boundary line width.
<code>incl.labels</code>	Integer vector of pixel indices to label at resolution <code>j</code> .
<code>col</code>	The colour to make the grandchildren.
<code>size</code>	The size to make the grandchildren.

**Examples**

```

## Example 1

## Plot base pixels 1,2,3 by colouring their grandchildren at resolution
## 5 (by default). No pixel boundaries.
displayPixels(j=0, spix=c(1,2,3))

## Plot base pixels 1,2,3 display and their boundaries (boundary.j=0)
displayPixels(0,0, spix=c(1,2,3))

## Plot base pixels 1,2,3 by colouring their grandchildren at resolution 2
displayPixels(0,0, plot.j = 2, spix=c(1,2,3))

## Example 2

demoNeighbours <- function(p,j) {
  neighbours(p, j)
  displayPixels(boundary.j = j, j = j, plot.j = 5,
               spix = neighbours(p, j),
               boundary.col = "gray",
               boundary.lwd = 1,
               incl.labels = neighbours(p, j),
               col = "blue",
               size = 3)
  rcosmo::displayPixelBoundaries(inside = 1, col = "blue", lwd = 3)
}

demoNeighbours(1,2)

```

---

downloadCMBMap

*Download CMB Maps from Planck Public Data Release.*


---

**Description**

The function downloadCMBMap downloads CMB maps from [http://irsa.ipac.caltech.edu/data/Planck/release\\_2/all-sky-maps/matrix\\_cmb.html](http://irsa.ipac.caltech.edu/data/Planck/release_2/all-sky-maps/matrix_cmb.html).

**Usage**

```
downloadCMBMap(foreground = "smica", nside = 1024, destfile)
```

**Arguments**

foreground	A string naming the foreground separation method pipeline. Please choose one of "COMMANDER", "NILC", "SEVEM" or "SMICA" (not case sensitive).
nside	An integer. The nside parameter (resolution) required. The available options are 1024 or 2048.

`destfile` An optional character string with the path and file name for the downloaded file to be saved. Defaults to the working directory. Tilde-expansion is performed.

### Details

CMB maps have been produced by the COMMANDER, NILC, SEVEM, and SMICA pipelines, respectively.

For each pipeline, the intensity maps are provided at  $N_{\text{side}} = 2048$ , at 5 arcmin resolution, and the polarization maps are provided at  $N_{\text{side}} = 1024$  at 10 arcmin resolution.

### Value

CMB Map FITS File (Flexible Image Transport System). The FITS file can be loaded into a `CMBDataFrame` using the `CMBDataFrame` function (see examples).

### References

Planck Public Data Release 2 Maps [http://irsa.ipac.caltech.edu/data/Planck/release\\_2/all-sky-maps/matrix\\_cmb.html](http://irsa.ipac.caltech.edu/data/Planck/release_2/all-sky-maps/matrix_cmb.html)

Other fits maps can also be downloaded using the general command [download.file](#).

### Examples

```
## Download SMICA with \code{nside = 1024}
## and save in working directory
## as "CMB_map_smica1024.fits"
# downloadCMBMap(foreground = "smica", nside = 1024)
## Load the downloaded map into a CMBDataFrame
# sky <- CMBDataFrame("CMB_map_smica1024.fits")

## Download SMICA with Nside=2048 and save in the working directory
## as "CMB_map_smica2048.fits"
# downloadCMBMap(foreground = "smica", nside = 2048)

## Download COMMANDER with Nside=1024 and save in a specified folder,
## for example,
# dest <- "CMB_map_commander1024.fits"
# downloadCMBMap(foreground = "commander", nside = 1024, destfile = dest)
```

---

downloadCMBPS

*Download CMB Power Spectra from Planck Legacy Archive.*

---

### Description

The function `downloadCMBPS` downloads CMB power spectra components from <http://pla.esac.esa.int/pla/#cosmology>.

**Usage**

```
downloadCMBPS(link = 1, destfile, save = TRUE)
```

**Arguments**

link	The link code (an integer from 1 to 6) for the URL to download the file. See code details in this help file.
destfile	A character string with the file name for the downloaded file to be saved. Tilde-expansion is performed.
save	A boolean indicating whether to save or not (since the downloaded data is returned anyway).

**Details**

link = 1: Best-fit LCDM CMB power spectra from the baseline Planck TT, TE, EE+lowE+lensing ( $2 \leq \ell \leq 2508$ ).

link = 2: Baseline high- $\ell$  Planck TT power spectra ( $2 \leq \ell \leq 2508$ ).

link = 3: Baseline high- $\ell$  Planck EE power spectra ( $2 \leq \ell \leq 1996$ ).

link = 4: Baseline high- $\ell$  Planck TE power spectra ( $2 \leq \ell \leq 1996$ ).

link = 5: Low- $\ell$  Planck EB power spectra ( $2 \leq \ell \leq 29$ ).

link = 6: Low- $\ell$  Planck BB power spectra ( $2 \leq \ell \leq 29$ ).

**Value**

The Data Frame with CMB Power Spectra and, if `save = TRUE` a txt file is saved in `destfile`

**References**

Planck Legacy Archive <http://pla.esac.esa.int/pla/#cosmology>

**Examples**

```
## Download the Low-ell Planck BB power spectra (2 <= ell <= 29) and
## save it to C:/PW.txt
# downloadCMBPS(link=6, destfile="C:/PW.txt")

## Download the Best-fit LCDM CMB power spectra
## and plot it
# CMBPS <- downloadCMBPS(link=1, save = FALSE)
# plot(CMBPS$L,CMBPS$TT, type="o",col="red",cex=0.3,
#       main="CMB Angular Power Spectra",xlab=expression(l),
#       ylab=expression(paste(D[l],"(",mu,K^2,")")))

```

---

entropyCMB	<i>CMB Entropy</i>
------------	--------------------

---

## Description

This function returns an estimated entropy for the specified [CMBDataFrame](#) column intensities and [CMBWindow](#) region. The function employs the function [entropy](#) and uses histogram counts of intensities for computations. All arguments of the standard [entropy](#) can be used.

## Usage

```
entropyCMB(cmbdf, win, intensities = "I", method)
```

## Arguments

cmbdf	A <a href="#">CMBDataFrame</a> .
win	A <a href="#">CMBWindow</a>
intensities	A <a href="#">CMBDataFrame</a> column with measured values.
method	A method to estimate entropy, see <a href="#">entropy</a>

## Value

Estimated Shannon entropy for observations in [CMBWindow](#)

## References

[entropy](#)

## Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 10000)
# win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
# entropyCMB(cmbdf, win1)
```

---

exprob	<i>Threshold exceedance probability</i>
--------	---

---

## Description

This function returns an estimated exceedance probability for the specified [CMBDataFrame](#) column intensities, threshold level *alpha* and [CMBWindow](#) region.

## Usage

```
exprob(cmbdf, win, alpha, intensities = "I")
```

## Arguments

<code>cmbdf</code>	A <a href="#">CMBDataFrame</a> .
<code>win</code>	A <a href="#">CMBWindow</a>
<code>alpha</code>	A numeric threshold level.
<code>intensities</code>	The name of the column in <code>cmbdf</code> that contains the measured values.

## Value

Estimated threshold exceedance probability

## Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 1000)

# intensities <- "I"
# alpha <- mean(cmbdf[,intensities, drop = TRUE])
# alpha

# win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
# exprob(cmbdf, win1, alpha)
```

---

extrCMB	<i>Extreme values</i>
---------	-----------------------

---

## Description

This function returns  $n$  largest extreme values for the specified `CMBDataFrame` column intensities and `CMBWindow` region.

## Usage

```
extrCMB(cmbdf, win, n, intensities = "I")
```

## Arguments

<code>cmbdf</code>	A <code>CMBDataFrame</code> .
<code>win</code>	A <code>CMBWindow</code>
<code>n</code>	An integer value.
<code>intensities</code>	A <code>CMBDataFrame</code> column with measured values.

## Value

A `CMBDataFrame` with  $n$  largest extreme values

## Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 1000)
#
# win1 <- CMBWindow(theta = c(pi/2,pi,pi/2), phi = c(0,0,pi/2))
# extrCMB(cmbdf, win1,5)
#
## Plotting the window and 5 top extreme values
# plot(win1)
# plot(extrCMB(cmbdf, win1,5), col = "blue", size = 4, add = TRUE)
```

---

fmf

*First Minkowski functional*

---

### Description

This function returns an area of the spherical region where measured values are above of the specified threshold level *alpha*.

### Usage

```
fmf(cmbdf, alpha, intensities = "I")
```

### Arguments

`cmbdf`            A [CMBDataFrame](#).  
`alpha`            A numeric threshold level.  
`intensities`     A [CMBDataFrame](#) column with measured values.

### Value

The area of the exceedance region

### References

Leonenko N., Olenko A. (2014) Sojourn measures of Student and Fisher-Snedecor random fields. *Bernoulli*, 20:1454-1483.

### Examples

```
n <- 64
cmbdf <- CMBDataFrame(nside=n, I = rnorm(12*n^2),
                      coords = "cartesian",
                      ordering = "nested")

fmf(cmbdf, 0, 4)
fmf(cmbdf, 2, 4)

win <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
cmbdf.win <- window(cmbdf, new.window = win)
fmf(cmbdf.win, 0, 4)
```



---

fRen	<i>Sample Renyi function</i>
------	------------------------------

---

### Description

This function computes values of the sample Renyi function. Returns the estimated values of  $T(q)$  for  $q$  taking values on a grid. For large data sets could be rather time consuming.

### Usage

```
fRen(cmbdf, q.min = 1.01, q.max = 10, N = 20,
     k.box = log2(inside(cmbdf)) - 3, intensities = "I")
```

### Arguments

cmbdf	A <a href="#">CMBDataFrame</a> .
q.min	Left endpoint of the interval to compute the Renyi function. The default value is 1.01,
q.max	Right endpoint of the interval to compute the Renyi function. The default value is 10
N	Number of points to compute the Renyi function. The default value is 20.
k.box	A dyadic decomposition level in computing the Renyi function, see the references in Details. The default value is $\log_2(\text{inside}(\text{cmbdf})) - 3$
intensities	A <a href="#">CMBDataFrame</a> column with measured values

### Value

Data frame which first column is the sampling grid  $\text{seq}(q.\text{min}, q.\text{max}, \text{length.out} = N)$  of  $q$  values. Another column consists of values of the sample Renyi function  $T(q)$  computed on the grid using the  $k.\text{box}$ th level dyadic decomposition of the unit ball.

### References

- (1) Leonenko, N., and Shieh, N. 2013. Rényi function for multifractal random fields. *Fractals* 21, Article No. 1350009.
- (2) <http://mathworld.wolfram.com/RenyiEntropy.html>

### Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
#
# cmbdf <- CMBDataFrame("CMB_map_smica1024.fits")
# win <- CMBWindow(theta = c(pi/4, pi/2, pi/2), phi = c(0, 0, pi/2))
# cmbdf <- window(cmbdf, new.window = win)
```

```
# Tq <- fRen(cmbdf)
#
# plot(Tq[,1], Tq[,2], ylab =expression(D[q]), xlab = "q",
# main = "Sample Renyi function", pch = 20, col = "blue")
```

---

geo2sph

*Convert geographic to spherical coordinates*

---

### Description

Convert latitude (lat) and longitude (lon) to spherical coordinates (theta, phi) with theta in  $[0, \pi]$  and phi in  $[0, 2 * \pi]$ . All values lat, lon, theta, phi are assumed to be in radians.

### Usage

```
geo2sph(...)
```

### Arguments

... A data.frame with columns lat and lon, or named vectors of lat and lon.

### Value

A data.frame with columns theta and phi.

### Examples

```
geo <- data.frame( lat = c(0, pi/3, pi/2), lon = c(0, pi/3, pi))
geo
sph <- geo2sph(geo)
sph
```

---

geoAngle

*Angle between two spherical directions*

---

### Description

Get an angle between two directions defined by arcs on the unit sphere

### Usage

```
geoAngle(p1)
```

**Arguments**

`p1` A `data.frame` with rows specifying numeric points located on the unit sphere. It should have columns labelled `x,y,z` for Cartesian or `theta, phi` for spherical colatitude and longitude respectively.

**Value**

Let  $p1[1,]$ ,  $p1[2,]$ , and  $p1[3,]$  denote the rows of `p1`. Then the returned object is an angle in radians between two arcs determined by # the pairs # of spherical points ( $p1[1,]$ ,  $p1[2,]$ ) and ( $p1[2,]$ ,  $p1[3,]$ ) respectively.

**Examples**

```
p1 <- data.frame(diag(3))
p1
colnames(p1) <- c("x", "y", "z")
geoAngle(p1)

geo <- data.frame( lat = c(30, 0, 20), lon = c(30, 60, 10))*(pi/180)
geo
p2 <- geo2sph(geo)
p2
geoAngle(p2)
```

---

 geoArea

*geoArea generic*


---

**Description**

Detailed descriptions and examples can be found in documentation for specific `geoArea` functions [geoArea.CMBDataFrame](#), [geoArea.HPDataFrame](#), [geoArea.CMBWindow](#)

**Usage**

```
geoArea(x)
```

**Arguments**

`x` An object.

**See Also**

[geoArea.CMBDataFrame](#) [geoArea.HPDataFrame](#) [geoArea.CMBWindow](#)

---

geoArea.CMBDataFrame    *Geodesic area covered by a CMBDataFrame*

---

### Description

Gives the surface on the unit sphere that is encompassed by all pixels in cmbdf

### Usage

```
## S3 method for class 'CMBDataFrame'
geoArea(x)
```

### Arguments

x                    a CMBDataFrame.

### Value

The sum of the areas of all pixels (rows) in x.

### Examples

```
## At low resolution, a few data points can
## occupy a large pixel area, e.g.:

cmbdf <- CMBDataFrame(nside = 1, spix = c(1,2,3))
pix(cmbdf)

## The total number of Healpix points at nside=1 equals 12. As cmbdf has 3 Healpix
## it occupies pi = 1/4*(surface area of unit sphere)

geoArea(cmbdf)
plot(cmbdf, size = 5, hp.boundaries = 1)
```

---

geoArea.CMBWindow    *Geodesic area of a CMBWindow*

---

### Description

Geodesic area of a CMBWindow

### Usage

```
## S3 method for class 'CMBWindow'
geoArea(x)
```

**Arguments**

x                    A CMBWindow.

**Value**

The spherical area inside the CMBWindow x.

**Examples**

```
## A window that covers 1/8 of the unit sphere is constructed and its area is
## pi/2 = 1/8*(surface area of unit sphere)

win <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
geoArea(win)
```

---

geoArea.HPDataFrame    *Geodesic area covered by a [HPDataFrame](#)*

---

**Description**

Gives the surface on the unit sphere that is encompassed by all pixels in x.

**Usage**

```
## S3 method for class 'HPDataFrame'
geoArea(x)
```

**Arguments**

x                    A HPDataFrame.

**Value**

The sum of the areas of all pixels (rows) in x.

**Examples**

```
## Generate random I for HPDataFrame
hp1 <- HPDataFrame(I=rnorm(5), nside = 1, spix = c(1,1,2,2,3))
pix(hp1)

## The total number of Healpix points at nside=1 equals 12. As hp1 has five
## I values at 3 Healpix points, then the occupied area is
## pi = 1/4*(surface area of unit sphere)

geoArea(hp1)
```

```
plot(hp1, size = 5, hp.boundaries = 1)
```

---

geoDist	<i>Geodesic distance on the unit sphere</i>
---------	---

---

### Description

Get geodesic distance between points on the unit sphere

### Usage

```
geoDist(p1, p2, include.names = FALSE)
```

### Arguments

p1	A <code>data.frame</code> with rows specifying numeric points located on the unit sphere. It should have columns labelled x,y,z for Cartesian or theta, phi for spherical colatitude and longitude respectively.
p2	Same as p1.
include.names	Boolean. If TRUE then the row and column names of the returned matrix will be taken from the points in p1 and p2 (see examples below).

### Value

Let  $n$  denote the number of rows of p1 and let  $m$  denote the number of rows of p2. Then the returned object is an  $n$  by  $m$  matrix whose entry in position  $ij$  is the geodesic distance from the  $i$ th row of p1 to the  $j$ th row of p2.

### Examples

```
p1 <- data.frame(diag(3))
colnames(p1) <- c("x", "y", "z")
p1
p2 <- data.frame(x=c(1,0), y=c(0,3/5), z=c(0,4/5))
p2
geoDist(p1, p2, include.names = FALSE)
```

---

header	<i>Get the FITS headers from a <a href="#">CMBDataFrame</a></i>
--------	---

---

**Description**

Get the FITS headers from a [CMBDataFrame](#)

**Usage**

```
header(cmbdf)
```

**Arguments**

cmbdf            a [CMBDataFrame](#).

**Value**

The FITS headers belonging to the FITS file from which cmbdf data was imported

**Examples**

```
## First download the map
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# df.sample <- CMBDataFrame(df, sample.size = 10000)
# header(df.sample)
```

---

healpixCentered	<i>Check if object is assumed to have <b>HEALPix</b> centered coordinates</i>
-----------------	---

---

**Description**

The function checks object's attribute healpixCentered. The attribute is TRUE if the object was assumed to have rows that correspond to unique HEALPix pixel indices.

**Usage**

```
healpixCentered(obj)
```

**Arguments**

obj            Any object.

**Value**

A boolean. This is TRUE if obj is a [CMBDataFrame](#) or a [HPDataFrame](#) whose coordinates were assumed to correspond to HEALPix pixel center locations.

**Examples**

```

hp1 <- HPDataFrame(I=rnorm(5), nside = 1, spix = c(1,1,2,2,3))
pix(hp1)
coords(hp1, new.coords = "cartesian")
healpixCentered(hp1)

sky <- CMBDataFrame(nside = 32, coords = "cartesian", ordering = "nested")
sky.s <- CMBDataFrame(sky, sample.size = 100)
hpdf <- HPDataFrame(sky.s, auto.spix = TRUE)
healpixCentered(hpdf)

```

---

HPDataFrame

*HPDataFrame class*


---

**Description**

HPDataFrames are a type of data.frame designed to carry data located on the unit sphere. Each row of a HPDataFrame is associated with a HEALPix pixel index. The HPDataFrame also holds an attribute called nside which stores the HEALPix Nside parameter (i.e., the resolution of the HEALPix grid that is being used). Unlike `CMBDataFrame`, HPDataFrames may have repeated pixel indices. They are made this way so that multiple data points falling within a given pixel can be stored in different rows of any given HPDataFrame.

**Usage**

```

HPDataFrame(..., nside, ordering = "nested", auto.spix = FALSE, spix,
  assumedUniquePix = FALSE, delete.duplicates = FALSE,
  save.dots = FALSE, save.duplicate.indices = FALSE)

```

**Arguments**

...	Data. Can be named vectors or a data.frame. May include columns (x,y,z) or (theta, phi) representing Cartesian or spherical coordinates of points on the unit sphere.
nside	Integer number $2^k$ , the nside parameter, i.e, resolution. If nside is unspecified, then the an attempt is made to use columns x,y and z from the provided data, as Cartesian coordinates, to calculate an nside that is sufficient to ensure all points belong to unique pixels.
ordering	The HEALPix ordering scheme ("ring" or "nested").
auto.spix	Boolean. If TRUE then spix will be found from the coordinates provided in the data. That is, each row of data will be assigned the pixel index of its closest HEALPix pixel center. There must be columns x,y,z for cartesian or theta, phi for spherical colatitude and longitude respectively. If auto.spix = FALSE then nside must be specified.



<code>spix</code>	A vector of HEALPix pixel indices indicating the pixel locations of the data. Note that <code>spix</code> is ignored if <code>auto.spix = TRUE</code> .
<code>assumedUniquePix</code>	A boolean. Sets the <code>assumedUniquePix</code> attribute of the HPDataFrame. This attribute indicates whether or not the rows of a HPDataFrame can be assumed to belong to unique pixels.
<code>delete.duplicates</code>	Boolean. If TRUE then rows corresponding to duplicate pixel indices will be dropped from the returned HPDataFrame, and <code>assumedUniquePix</code> will be set to TRUE.
<code>save.dots</code>	A logical. If TRUE then the dot product of each observation with the nearest child HEALPix pixel center will be stored as a column called "distance" in the returned HPDataFrame, provided that <code>auto.spix = TRUE</code> . Note that a 'child' pixel is any one of the four pixels contained in the current pixel, in the nested scheme, at the next highest resolution. See <a href="#">children</a> .
<code>save.duplicate.indices</code>	A logical. If TRUE and <code>delete.duplicates</code> is also TRUE, then the row indices of duplicated pixels will be retained as an attribute called "duplicates". Note that row index refers to the row position of the duplicated pixel in the original HPDataFrame, and not the actual pixel index itself.

## Details

HPDataFrame with `auto.spix = TRUE` can be used to transform any spherical data (not necessarily CMB) to the Healpix representation, see Example 3 below.

## Examples

```
##Example 1.

hp1 <- HPDataFrame(I=rnorm(5), nside = 1, spix = c(1,1,2,2,3))
pix(hp1)
coords(hp1, new.coords = "cartesian")
class(hp1)
assumedUniquePix(hp1)

##Example 2.

# Where nside is not specified
sky <- CMBDataFrame(nside = 32, coords = "cartesian", ordering = "nested")
sky.s <- CMBDataFrame(sky, sample.size = 100)
hpdf <- HPDataFrame(sky.s, auto.spix = TRUE)
class(hpdf)
assumedUniquePix(hpdf)

# ## Example 3.
# ## Create a HPDataFrame with NON-UNIQUE pixel indices
#
```

```

# ## With earth data.
# ## Download World Cities Database from
# ## https://simplemaps.com/static/data/world-cities/basic/simplemaps_worldcities_basicv1.4.zip
# ## unpack the file worldcities.csv
#
# worldcities <- read.csv("worldcities.csv")
#
# ## Prepare a data frame with cities' coordinates
# sph <- geo2sph(data.frame(lon = pi/180*worldcities$lng,
#                           lat = pi/180*worldcities$lat))
# df <- data.frame(phi = sph$phi,
#                  theta = sph$theta,
#                  I = rep(1,nrow(sph)))
#
# ## Create and plot the corresponding HPDataFrame with
# ## pixel indices that are not necessarily unique
# ## by choosing your desired resolution (nside)
# hp <- HPDataFrame(usdf, auto.spix = TRUE, nside = 1024)
# plot(hp, size = 3, col = "darkgreen", back.col = "white")
# ## Add some pixels to visualise the sphere
# plot(CMBDataFrame(nside = 64), add = TRUE, col = "gray")

# ## Example 4.
# ## Create a HPDataFrame with UNIQUE pixel indices.
#
# ## With earth data.
# ## Download World Cities Database from
# ## https://simplemaps.com/static/data/world-cities/basic/simplemaps_worldcities_basicv1.4.zip
# ## unpack the file worldcities.csv
#
# worldcities <- read.csv("worldcities.csv")
# uscities <- worldcities[worldcities$country == "United States",]
#
# ## Prepare a data frame with cities' coordinates
# sph <- geo2sph(data.frame(lon = pi/180*uscities$lng,
#                           lat = pi/180*uscities$lat))
# usdf <- data.frame(phi = sph$phi,
#                   theta = sph$theta,
#                   I = rep(1,nrow(sph)))
#
# ## Select k cities with unique coordinates. The
# ## coordinates must be unique otherwise the
# ## automatically chosen separating nside
# ## will be infinite.
# k <- 1000
# usdf <- usdf[sample(nrow(usdf), k), ]
# plot(usdf$phi, usdf$theta)
# usdf[duplicated(usdf), ]
# usdf <- usdf[!duplicated(usdf), ]
# usdf[duplicated(usdf), ]
# usdf <- coords(usdf, new.coords = "cartesian")
#
# ## Create and plot the corresponding HPDataFrame . To make

```

```

# ## sure the pixels are unique, do not select a resolution
# ## resolution (nside), since it will be chosen automatically.
# ushp <- HPDataFrame(usdf, auto.spix = TRUE)
# nside(ushp)
# assumedUniquePix(ushp)
# plot(ushp, size = 3, col = "darkgreen", back.col = "white")
# ## Add some pixels to visualise the sphere
# plot(CMBDataFrame(nside = 64), add = TRUE, col = "gray")

```

---

ibp2p

*Computes pixel's index using its subindex within base resolution*


---

### Description

Find the pixel index  $p$  of a given pixel with index  $ibp$  in base pixel  $bp$ .

### Usage

```
ibp2p(ibp, bp, j)
```

### Arguments

<code>ibp</code>	The pixel index within base pixel $bp$ , at resolution $j$ , in nested order.
<code>bp</code>	The base pixel index
<code>j</code>	The resolution parameter $nside = 2^j$

### Examples

```

ibp2p(1, 1, 2)
ibp2p(1, 2, 2)

```

---

is.CMBDat

*Check if an object is of class CMBDat*


---

### Description

Check if an object is of class CMBDat

### Usage

```
is.CMBDat(cmbdf)
```

**Arguments**

cmbdf            Any R object

**Value**

TRUE if cmbdf is a CMBDat object, otherwise FALSE

**Examples**

```
## First download the map
# downloadCMBMap(foreground = "smica", nside = 1024)
# cmbdat <- CMBDat("CMB_map_smica1024.fits", mmap = TRUE)
# class(cmbdat)
# is.CMBDat(cmbdat)
```

---

is.CMBDataFrame            *Check if an object is of class CMBDataFrame*

---

**Description**

Check if an object is of class CMBDataFrame

**Usage**

```
is.CMBDataFrame(cmbdf)
```

**Arguments**

cmbdf            Any R object

**Value**

TRUE if cmbdf is a CMBDataFrame, otherwise FALSE

**Examples**

```
df <- CMBDataFrame(nside = 16)
is.CMBDataFrame(df)
df2 <- coords(df, new.coords = "cartesian")
is.CMBDataFrame(df2)
```

---

is.CMBWindow                    *Check if an object is a CMBWindow*

---

**Description**

Check if an object is a CMBWindow

**Usage**

```
is.CMBWindow(win)
```

**Arguments**

win                    any object

**Value**

TRUE or FALSE depending if win is a CMBWindow

**Examples**

```
win <- CMBWindow(x=0,y=3/5,z=4/5,r=0.8, set.minus = TRUE)
is.CMBWindow(win)
```

---

is.HPDataFrame                    *Check if an object is of class [HPDataFrame](#)*

---

**Description**

Check if an object is of class [HPDataFrame](#)

**Usage**

```
is.HPDataFrame(hpdf)
```

**Arguments**

hpdf                    Any R object

**Value**

TRUE if hpdf is a HPDataFrame, otherwise FALSE

## Examples

```
df <- CMBDataFrame(nside = 16)
is.HPDataFrame(df)

df <- HPDataFrame(I = rep(0,12), nside = 1)
is.HPDataFrame(df)
```

---

linesCMB

*Adds lines of fitted variograms to variogram plots*

---

## Description

This function adds a line with the variogram model fitted by the function `variofitCMB` to a current variogram plot. The function modifies `lines.variomodel.variofit` from the package **geoR** for additional covariance models on spheres.

## Usage

```
linesCMB(x, max.dist, scaled = FALSE, ...)
```

## Arguments

<code>x</code>	An object of the class <code>variofit</code> containing information about the fitted model obtained as an output of the function <code>variofitCMB</code> .
<code>max.dist</code>	A maximum distance to draw the variogram line. The default is <code>x\$max.dist</code> .
<code>scaled</code>	logical. If TRUE the sill in the plot is 1.
<code>...</code>	other plotting parameters passed to <code>curve</code>

## Details

The function adds a line with fitted variogram model to a plot. It is used to compare empirical variograms against fitted models returned by `variofitCMB`.

Available models are: "matern", "exponential", "spherical", "powered.exponential", "cauchy", "gen-cauchy", "pure.nugget", "askey", "c2wendland", "c4wendland", "sinpower", "multiquadric".

## Value

A line with a fitted variogram model is added to a plot.

## References

**geoR** package, `lines.variomodel.variofit`, `covmodelCMB`, `variofitCMB`

## Examples

```
## Plot the fitted Matern variogram versus its empirical variogram
#
# df <- CMBDataFrame("../CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 10000)
# varcmb <- variogramCMB(cmbdf, max.dist = 0.1, num.bins = 30)
# varcmb
# ols <- variofitCMB(varcmb, fix.nug=FALSE, wei="equal", cov.model= "matern")
# plot(varcmb)
# lines(ols, lty=2)
#
## Plot the fitted Askey variogram versus its empirical variogram
#
# ols <- variofitCMB(vario1, ini.cov.pars = c(1, 0.03), fix.nug = TRUE,
#   kappa = 3, wei = "equal", cov.model = "askey")
# plot(varcmb, main = ols$cov.model)
# linesCMB(ols, lty = 2)
```

---

maxDist

*Get the maximum geodesic distance between points*

---

## Description

Get the maximum geodesic distance either between all points in a data.frame pairwise, or between all points in a data.frame and one target point.

## Usage

```
maxDist(df, point)
```

## Arguments

df	A data.frame with columns x,y,z for cartesian or theta, phi for spherical colatitude and longitude respectively. The rows must correspond to points on the unit sphere. If this is a <a href="#">HPDataFrame</a> or <a href="#">CMBDataFrame</a> and coordinate columns are missing, then coordinates will be assigned based on HEALPix pixel indices.
point	An optional target point on the unit sphere in cartesian coordinates, in which case all distances are calculated between point and the points in df.

## Value

If point is specified: the longest geodesic distance from point to the points specified by the rows of df. If point is not specified: the longest geodesic distance pairwise between points in df.

**Examples**

```

## Using a CMBDataFrame with HEALPix coordinates only
cmbdf <- CMBDataFrame(nside = 1, spix = c(1,5,12), ordering = "ring")
plot(cmbdf, hp.boundaries = 1, col = "blue", size = 5)
p <- c(0,0,1)
maxDist(cmbdf, p) # no need to have coordinates

## Using a HPDataFrame with HEALPix coordinates only
hp <- HPDataFrame(nside = 1, I = rep(0,3), spix = c(1,5,12) )
maxDist(hp, p) # notice no need to have coordinates

## Using a data.frame with cartesian coordinates
coords(hp) <- "cartesian"
df <- data.frame(x = hp$x, y = hp$y, z = hp$z)
maxDist(df, p)

## Using a data.frame with spherical coordinates
coords(hp) <- "spherical"
df <- data.frame(theta = hp$theta, phi = hp$phi)
maxDist(df, p)

## max distance between points in cmdf
maxDist(cmbdf)

```

---

maxWindowDist

*Get the maximum distance between all points in a [CMBWindow](#)*


---

**Description**

Get the maximum distance between all points in a [CMBWindow](#)

**Usage**

```
maxWindowDist(x)
```

**Arguments**

x                    A CMBWindow object.

**Value**

The maximum distance between window's points.



## Examples

```
## win is a equilateral spherical triangle which sides are pi/2
win <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
maxWindowDist(win)
```

---

minDist

*Get the minimum geodesic distance between points*

---

## Description

Get the minimum geodesic distance either between all points in a data.frame pairwise, or between all points in a data.frame and one target point.

## Usage

```
minDist(df, point)
```

## Arguments

df	A data.frame with columns x,y,z for cartesian or theta, phi for spherical colatitude and longitude respectively. The rows must correspond to points on the unit sphere. If this is a <a href="#">HPDataFrame</a> or <a href="#">CMBDataFrame</a> and coordinate columns are missing, then coordinates will be assigned based on HEALPix pixel indices.
point	An optional target point on the unit sphere in cartesian coordinates, in which case all distances are calculated between point and the points in df.

## Value

If point is specified: the shortest distance from point to the points specified by the rows of df. If point is not specified: the shortest distance pairwise between points in df.

## Examples

```
## Using a CMBDataFrame with HEALPix coordinates only
cmbdf <- CMBDataFrame(nside = 1, spix = c(1,5,12), ordering = "ring")
plot(cmbdf, hp.boundaries = 1, col = "blue", size = 5)
p <- c(0,0,1)
minDist(cmbdf, p) # no need to have coordinates

## Using a HPDataFrame with HEALPix coordinates only
hp <- HPDataFrame(nside = 1, I = rep(0,3), spix = c(1,5,12) )
minDist(hp, p) # notice no need to have coordinates

## Using a data.frame with cartesian coordinates
coords(hp) <- "cartesian"
```

```
df <- data.frame(x = hp$x, y = hp$y, z = hp$z)
minDist(df, p)

## Using a data.frame with spherical coordinates
coords(hp) <- "spherical"
df <- data.frame(theta = hp$theta, phi = hp$phi)
minDist(df, p)

## min distance between points in cmdf
minDist(cmbdf)
```

---

neighbours	<i>Return neighbouring pixels</i>
------------	-----------------------------------

---

### Description

Return the neighbouring pixels to a given pixel  $p$  that is specified at resolution  $j$ , in the nested order.

### Usage

```
neighbours(p, j)
```

### Arguments

$p$	Pixel index $p$ at resolution $j$ .
$j$	The resolution parameter with $n_{\text{side}} = 2^j$ .

### Examples

```
## Return the neighbouring pixels for base pixel 1
neighbours(1, 0)

## Plot the neighbouring pixels for base pixel 1
demoNeighbours <- function(p,j) {
  neighbours(p, j)
  displayPixels(boundary.j = j, j = j, plot.j = j+3,
               spix = neighbours(p, j),
               boundary.col = "gray",
               boundary.lwd = 1,
               incl.labels = neighbours(p, j),
               col = "blue",
               size = 3)
  rcosmo::displayPixelBoundaries(nside = 1, col = "blue", lwd = 3)
}

demoNeighbours(1,2)
demoNeighbours(1,0)
```

---

nest2ring	<i>Convert nest to ring ordering</i>
-----------	--------------------------------------

---

**Description**

Convert from "nested" to "ring" ordering

nest2ring computes the HEALPix pixel index in the "ring" ordering scheme from the pixel index in the "nested" ordering scheme.

**Usage**

```
nest2ring(nside, pix)
```

**Arguments**

nside	is the HEALPix nside parameter.
pix	is the set or subset of pixel indices at nside. If pix is left blank then all pixels are converted.

**Value**

the output is the corresponding set of pixel in the ring ordering scheme.

**Examples**

```
# compute HEALPix indices in the ring ordering scheme
nside <- 8
pix <-c(1,2,23)
nest2ring(nside,pix)
```

---

nestSearch	<i>Finds the closest pixel center to a point</i>
------------	--

---

**Description**

Finds the closest HEALPix pixel center to a given target point, specified in Cartesian coordinates, using an efficient nested search algorithm. HEALPix indices are all assumed to be in the "nested" ordering scheme.

**Usage**

```
nestSearch(target, nside, index.only = FALSE, save.dots = FALSE)
```

**Arguments**

target	A data.frame, matrix or vector of Cartesian (x,y,z) coordinates for the target point. If a data.frame is used then spherical coordinates can be specified with row names theta and phi.
nside	An integer, the target resolution at which the resulting pixels are returned.
index.only	A boolean indicating whether to return only the pixel index (TRUE), or cartesian coordinates as well (FALSE).
save.dots	A logical. A If TRUE then the dot product of each observation with the nearest child HEALPix pixel center will be returned as an attribute called "dot". Note that a 'child' pixel is any one of the four pixels contained in the current pixel in the nested scheme, at the next highest resolution. See <a href="#">children</a> .

**Value**

if index.only = TRUE then the output will be a HEALPix index. If index.only FALSE then the output is the list containing the HEALPix index and Cartesian coordinate vector of the HEALPix point closest to target at resolution nside.

**Examples**

```
## Find the closest HEALPix pixel center at resolution j=2 for
## the point (0.6,0.8,0)

point <- c(0.6,0.8,0)
j <- 2
cpoint <- nestSearch(point, nside = 2^j)

## Plot the closest pixel center in blue and the point (0.6,0.8,0) in red
displayPixels(j, j, plot.j=j, spix=c(cpoint$pix),
              size=5, incl.labels =FALSE)
rgl::plot3d(point[1], point[2], point[3],
            col="red", size = 5, add = TRUE)

## Repeat the above for 4 points in a data.frame
points <- data.frame(x = c(1,0,0,0.6),
                    y = c(0,1,0,0.8),
                    z = c(0,0,1,0))

points
j <- 2
cpoints <- nestSearch(points, nside = 2^j)

## Plot the closest pixel center in blue and the point (0.6,0.8,0) in red
displayPixels(j, j, plot.j=j, spix=c(cpoints$pix),
              size=5, incl.labels =FALSE)
rgl::plot3d(points[,1], points[,2], points[,3],
            col="red", size = 5, add = TRUE)
```

---

nside	<i>nside generic</i>
-------	----------------------

---

**Description**

Detailed descriptions and examples can be found in documentation for specific nside functions [nside.CMBDataFrame](#), [nside.HPDataFrame](#)

**Usage**

```
nside(x)
```

**Arguments**

x                    An object.

**See Also**

[nside.CMBDataFrame](#) [nside.HPDataFrame](#)

---

nside.CMBDataFrame	<i>HEALPix Nside parameter from a CMBDataFrame</i>
--------------------	--

---

**Description**

This function returns the HEALPix Nside parameter of a CMBDataFrame

**Usage**

```
## S3 method for class 'CMBDataFrame'
nside(x)
```

**Arguments**

x                    A [CMBDataFrame](#).

**Value**

The HEALPix Nside parameter.

**Examples**

```
df <- CMBDataFrame(nside = 16)
nside(df)
```

---

nside.HPDataFrame	<i>HEALPix Nside parameter from a <a href="#">HPDataFrame</a></i>
-------------------	---

---

**Description**

This function returns the HEALPix Nside parameter of a [HPDataFrame](#)

**Usage**

```
## S3 method for class 'HPDataFrame'
nside(x)
```

**Arguments**

x                   A [HPDataFrame](#).

**Value**

The HEALPix Nside parameter.

**Examples**

```
df <- HPDataFrame(I = rep(0,12), nside = 1)
nside(df)
```

---

numeric2col	<i>numeric2col</i>
-------------	--------------------

---

**Description**

Map numeric values to a colour map

**Usage**

```
numeric2col(num, colmap = grDevices::terrain.colors(100),
  breaks.length = length(colmap))
```

**Arguments**

num                   A numeric vector. The numbers which will be mapped to colours.  
colmap                A colour map. See [palette](#).  
breaks.length        A single integer. Controls the number of breaks in the discretisation of num.

**Examples**

```
ns <- 16
sky <- CMBDataFrame(I = rnorm(12*ns^2), nside = ns)
plot(sky, col = numeric2col(sky$I))
```

---

ordering	<i>ordering generic</i>
----------	-------------------------

---

**Description**

Detailed descriptions and examples can be found in documentation for specific ordering functions [ordering.CMBDataFrame](#), [ordering.HPDataFrame](#)

**Usage**

```
ordering(x, ...)
```

**Arguments**

x	An object.
...	Extra arguments.

**See Also**

[ordering.CMBDataFrame](#) [ordering.HPDataFrame](#)

---

ordering.CMBDataFrame	<i>HEALPix ordering scheme from a CMBDataFrame</i>
-----------------------	--

---

**Description**

This function returns the HEALPix ordering scheme from a CMBDataFrame. The ordering scheme is either "ring" or "nested".

**Usage**

```
## S3 method for class 'CMBDataFrame'
ordering(x, new.ordering, ...)
```

**Arguments**

x	A <a href="#">CMBDataFrame</a> .
new.ordering	Specifies the new ordering ("ring" or "nest") if a change of ordering scheme is desired.
...	Unused arguments.

**Details**

If a new ordering is specified, using e.g. `new.ordering = "ring"`, the ordering scheme of the CMB-DataFrame will be converted.

**Value**

The name of the HEALPix ordering scheme that is used in the CMBDataFrame `x`.

**Examples**

```
df <- CMBDataFrame(nside = 1, ordering = "nested")
ordering(df)
df1 <- ordering(df, new.ordering = "ring")
ordering(df1)
```

---

ordering.HPDataFrame *HEALPix ordering scheme from a HPDataFrame*

---

**Description**

This function returns the HEALPix ordering scheme from a HPDataFrame. The ordering scheme is either "ring" or "nested". If a new ordering is specified, using e.g. `new.ordering = "ring"`, the ordering scheme of the HPDataFrame will be converted.

**Usage**

```
## S3 method for class 'HPDataFrame'
ordering(x, new.ordering, ...)
```

**Arguments**

<code>x</code>	a <a href="#">HPDataFrame</a> .
<code>new.ordering</code>	Specifies the new ordering ("ring" or "nest") if a change of ordering scheme is desired.
<code>...</code>	Unused arguments.

**Value**

The name of the HEALPix ordering scheme that is used in the HPDataFrame `x`, or a new HP-DataFrame with the desired `new.ordering`



**Examples**

```
df <- HPDataFrame(I = rep(0,12), nside = 1, ordering = "nested")
ordering(df)
df1 <- ordering(df, new.ordering = "ring")
ordering(df1)
```

---

p2bp

*Return base pixel to which pixel belongs*


---

**Description**

The base pixel to which pixel  $p$  belongs at resolution  $j$

**Usage**

```
p2bp(p, j)
```

**Arguments**

$p$                     The pixel index at resolution  $j$ , in nested order.  
 $j$                     The resolution parameter  $nside = 2^j$

**Examples**

```
p2bp(5, 0)
p2bp(5, 1)
```

---

p2ibp

*Return pixel index within its base pixel*


---

**Description**

Convert a pixel index  $p$  to its index within the base pixel to which  $p$  belongs

**Usage**

```
p2ibp(p, j)
```

**Arguments**

$p$                     The pixel index at resolution  $j$ , in nested order.  
 $j$                     The resolution parameter  $nside = 2^j$

**Examples**

```
p2ibp(6, 0)
p2ibp(6, 1)
```

---

parent	<i>Return index of parent pixel</i>
--------	-------------------------------------

---

**Description**

Gives the pixel at resolution  $j - 1$  that contains  $p$ , where  $p$  is specified at resolution  $j$  (notice it does not depend on  $j$ ).

**Usage**

```
parent(p)
```

**Arguments**

p	A pixel index specified in nested order.
---	--

**Examples**

```
parent(4)
parent(5)
```

---

pix	<i>pix generic</i>
-----	--------------------

---

**Description**

Detailed descriptions and examples can be found in documentation for specific pix functions [pix.CMBDataFrame](#), [pix.HPDataFrame](#)

**Usage**

```
pix(x, ...)
```

**Arguments**

x	An object.
...	Extra arguments.

**See Also**

[pix.CMBDataFrame](#) [pix.HPDataFrame](#)

---

pix.CMBDataFrame      *HEALPix pixel indices from CMBDataFrame*

---

### Description

If new.pix is unspecified then this function returns the vector of HEALPix pixel indices from a CMBDataFrame. If new.pix is specified then this function returns a new CMBDataFrame with the same number of rows as cmbdf, but with pix attribute new.pix. Thus, new.pix must have length equal to nrow(cmbdf).

### Usage

```
## S3 method for class 'CMBDataFrame'
pix(x, new.pix, ...)
```

### Arguments

x	A CMBDataFrame.
new.pix	Optional vector of pixel indices with length equal to nrow(x).
...	Unused arguments.

### Value

The vector of HEALPix pixel indices or, if new.pix is specified, a new CMBDataFrame.

### Examples

```
## First download the map
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits", sample.size = 800000)
# pix(df)

df <- CMBDataFrame(nside = 16, sample.size = 10, ordering = "nested")
pix(df)
```

---

pix.HPDataFrame      *HEALPix pixel indices from HPDataFrame*

---

### Description

If new.pix is unspecified then this function returns the vector of HEALPix pixel indices from a HPDataFrame. If new.pix is specified then this function returns a new HPDataFrame with the same number of rows as x, but with pix attribute new.pix. Thus, new.pix must have length equal to nrow(x).

**Usage**

```
## S3 method for class 'HPDataFrame'
pix(x, new.pix, ...)
```

**Arguments**

x	a <a href="#">HPDataFrame</a> .
new.pix	optional vector of pixel indices with length equal to nrow(x)
...	Unused arguments.

**Value**

The vector of HEALPix pixel indices (integers) or, if new.pix is specified, a new HPDataFrame.

**Examples**

```
df <- HPDataFrame(I = rep(0,12), nside = 1)
pix(df)
```

---

pix2coords

*Convert pixel indices to cartesian/spherical coordinates*

---

**Description**

Convert HEALPix pixel indices to cartesian or spherical coordinates

**Usage**

```
pix2coords(nside, coords = "cartesian", ordering = "nested", spix)
```

**Arguments**

nside	the nside parameter (integer number $2^k$ )
coords	'cartesian' or 'spherical' coordinates
ordering	'ring' or 'nested' ordering
spix	optional integer or vector of sample pixel indices

**Value**

a data.frame with columns 'x', 'y', 'z' (cartesian) or 'theta', 'phi' (spherical)

**Examples**

```
pix2coords(nside=1, spix=c(2,5))
pix2coords(nside=1, coords = "spherical", spix=c(2,5))
```

---

pixelArea	<i>Area of a HEALPix pixel</i>
-----------	--------------------------------

---

**Description**

Get the area of a single HEALPix pixel

**Usage**

```
pixelArea(nsideObject)
```

**Arguments**

nsideObject     [CMBDataFrame](#), a [HPDataFrame](#), or an integer giving the nside parameter.

**Value**

the area of a single HEALPix pixel at the nside resolution of nsideObject

**Examples**

```
## First download the map
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# pixelArea(df)

df1 <- CMBDataFrame(nside = 64,
                    coords = "cartesian",
                    ordering = "nested")

pixelArea(df1)
```

---

pixelWindow	<i>Find high resolution pixels falling in a lower resolution window</i>
-------------	---

---

**Description**

Find all pixels in a higher resolution that fall within the specified pixel area at a lower resolution. All pixels are assumed to be in nested ordering.

**Usage**

```
pixelWindow(j1, j2, pix.j1)
```

**Arguments**

j1	An integer. The lower resolution, with $j1 \leq j2$ .
j2	An integer. The upper resolution.
pix.j1	An integer. The pixel index at resolution j1 within which all pixels from resolution j2 will be returned. pix.j1 can also be a vector of non-zero pixel indices.

**Value**

All pixels in resolution j2 that fall within the pixel pix.j1 specified at resolution j1

**Examples**

```
pixelWindow(3, 3, 2)
pixelWindow(3, 4, 2)
pixelWindow(3, 5, 2)
```

---

plot.CMBCorrelation    *Plot sample CMBCorrelation*

---

**Description**

Plots sample (empirical) correlation function. Uses [plot.variogram](#) from **geoR** package.

**Usage**

```
## S3 method for class 'CMBCorrelation'
plot(x, ...)
```

**Arguments**

x	An object of class CMBCorrelation.
...	Extra arguments as in <a href="#">plot.variogram</a> passed to plot.default.

**Value**

Produces a plot with the sample correlation function.

**References**

**geoR** package, [corrCMB](#), [variog](#), [plot.variogram](#)

## Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 100000)
# corcmb <- corrCMB(cmbdf, max.dist = 0.03, num.bins = 10, sample.size=1000)
# plot(corcmb)
```

---

plot.CMBCovariance      *Plot sample CMBCovariance*

---

## Description

Plots sample (empirical) covariance function. Uses [plot.variogram](#) from **geoR** package.

## Usage

```
## S3 method for class 'CMBCovariance'
plot(x, ...)
```

## Arguments

x                    An object of class CMBCovariance.  
...                    Extra arguments as in [plot.variogram](#) passed to plot.default.

## Value

Produces a plot with the sample covariance function.

#' @references **geoR** package, [covCMB](#), [variog](#), [plot.variogram](#)

## Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 100000)
# Cov <- covCMB(cmbdf, max.dist = 0.03, num.bins = 10)
# plot(Cov)
```

---

plot.CMBDataFrame      *Plot CMB Data*

---

### Description

This function produces a plot from a [CMBDataFrame](#).

### Usage

```
## S3 method for class 'CMBDataFrame'
plot(x, intensities = "I", add = FALSE,
     sample.size, type = "p", size = 1, box = FALSE, axes = FALSE,
     aspect = FALSE, col, back.col = "black", labels, hp.boundaries = 0,
     hpb.col = "gray", depth_test = "less", lab_depth_test = "always",
     ...)
```

### Arguments

x	A <a href="#">CMBDataFrame</a> .
intensities	The name of a column that specifies CMB intensities. This is only used if col is unspecified.
add	If TRUE then this plot will be added to any existing plot. Note that if back.col (see below) is specified then a new plot window will be opened and add = TRUE will have no effect.
sample.size	Optionally specifies the size of a simple random sample to take before plotting. This can make the plot less computationally intensive.
type	A single character indicating the type of item to plot. Supported types are: 'p' for points, 's' for spheres, 'l' for lines, 'h' for line segments from z = 0, and 'n' for nothing.
size	The size of plotted points.
box	Whether to draw a box.
axes	Whether to draw axes.
aspect	Either a logical indicating whether to adjust the aspect ratio, or a new ratio.
col	Specify the colour(s) of the plotted points.
back.col	Optionally specifies the background colour of the plot. This argument is passed to rgl::bg3d.
labels	Optionally specify a vector of labels to plot, such as words or vertex indices. If this is specified then rgl::text3d is used instead of rgl::plot3d. Then length(labels) must equal nrow(x).
hp.boundaries	Integer. If greater than 0 then HEALPix pixel boundaries at nside = hp.boundaries will be added to the plot.
hpb.col	Colour for the hp.boundaries.



depth\_test      The depth test to be applied to the plotted points. This controls how resistant the plotted object is to being obscured. See [rgl.material](#)

lab\_depth\_test   The [rgl](#) depth test to be applied to the labels and pixel boundaries if present. See [rgl.material](#)

...              Arguments passed to `rgl::plot3d`.

**Value**

A plot of the CMB data

**Examples**

```
## First download the map
# downloadCMBMap(foreground = "smica", nside = 1024)
# sky <- CMBDataFrame("CMB_map_smica1024.fits")
# plot(sky, sample.size = 800000)
```

---

plot.CMBWindow      *Visualise a CMBWindow*

---

**Description**

Visualise a [CMBWindow](#)

**Usage**

```
## S3 method for class 'CMBWindow'
plot(x, add = TRUE, type = "l", col = "red",
     size = 2, box = FALSE, axes = FALSE, aspect = FALSE, back.col,
     depth_test = "always", ...)
```

**Arguments**

x                    A CMBWindow.

add                  if TRUE then this plot will be added to any existing plot. Note that if `back.col` (see below) is specified then a new plot window will be opened and `add = TRUE` will have no effect

type                a single character indicating the type of item to plot. Supported types are: 'p' for points, 's' for spheres, 'l' for lines, 'h' for line segments from  $z = 0$ , and 'n' for nothing.

col                  specify the colour(s) of the plotted points

size                the size of plotted points

box                 whether to draw a box

axes                whether to draw axes

aspect	either a logical indicating whether to adjust the aspect ratio, or a new ratio.
back.col	specifies the background colour of the plot. This argument is passed to <code>rgl::bg3d</code> .
depth_test	The depth test to be applied. This controls how resistant the plotted object is to being obscured. See <a href="#">rgl.material</a>
...	arguments passed to <code>rgl::plot3d</code>

### Examples

```
win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
win2 <- CMBWindow(theta = c(2*pi/3,3*pi/4,3*pi/4, 2*pi/3), phi = c(pi/4,pi/4,pi/3,pi/3))
plot(win1)
plot(win2)
```

---

plot.HPDataFrame	<i>Plot HPDataFrame</i>
------------------	-------------------------

---

### Description

This function produces a plot from a [HPDataFrame](#). If columns `x,y,z` (cartesian) or `theta,phi` (colatitude and longitude respectively) are present in `x`, then these will be used as coordinates for plotting. Otherwise, the HEALPix indices as in `pix(x)` will be used. If HEALPix indices are used and multiple rows correspond to a single pixel index, then beware that values may be obfuscated in the plot, and all locations are pixel centers.

### Usage

```
## S3 method for class 'HPDataFrame'
plot(x, intensities = "I", add = FALSE,
     sample.size, type = "p", size = 1, box = FALSE, axes = FALSE,
     aspect = FALSE, col = "blue", back.col = "black", labels,
     hp.boundaries = 0, hpb.col = "gray", depth_test = "less",
     lab_depth_test = "always", ...)
```

### Arguments

<code>x</code>	A <code>HPDataFrame</code> .
<code>intensities</code>	The column name for the data in <code>x</code> that is to be treated as intensities for plotting.
<code>add</code>	if <code>TRUE</code> then this plot will be added to any existing plot. Note that if <code>back.col</code> (see below) is specified then a new plot window will be opened and <code>add = TRUE</code> will have no effect
<code>sample.size</code>	optionally specifies the size of a simple random sample to take before plotting. This can make the plot less computationally intensive

type	a single character indicating the type of item to plot. Supported types are: 'p' for points, 's' for spheres, 'l' for lines, 'h' for line segments from $z = 0$ , and 'n' for nothing.
size	the size of plotted points
box	whether to draw a box
axes	whether to draw axes
aspect	either a logical indicating whether to adjust the aspect ratio, or a new ratio.
col	specify the colour(s) of the plotted points
back.col	optionally specifies the background colour of the plot. This argument is passed to <code>rgl::bg3d</code> .
labels	optionally specify a vector of labels to plot, such as words or vertex indices. If this is specified then <code>rgl::text3d</code> is used instead of <code>rgl::plot3d</code> . Then <code>length(labels)</code> must equal <code>nrow(x)</code>
hp.boundaries	integer. If greater than 0 then HEALPix pixel boundaries at <code>nside = hp.boundaries</code> will be added to the plot
hpb.col	colour for the <code>hp.boundaries</code>
depth_test	The depth test to be applied to the plotted points. This controls how resistant the plotted object is to being obscured. This controls how resistant the plotted
lab_depth_test	The <code>rgl</code> depth test to be applied to the labels and pixel boundaries if present. See <a href="#">rgl.material</a>
...	arguments passed to <code>rgl::plot3d</code>

**Value**

A plot of the data locations according to coordinate columns or HEALPix index

**Examples**

```
hpdf <- HPDataFrame(I = rep(0,12), nside = 1)
plot(hpdf, size = 5, col = "yellow", back.col = "black",
      hp.boundaries = 1)
```

---

plot.variogram	<i>Plot sample variogram</i>
----------------	------------------------------

---

**Description**

Plots sample (empirical) variogram. Uses [plot.variogram](#) from **geoR** package.

**Arguments**

x	An object of class <code>variogram</code> .
...	Extra arguments as in <a href="#">plot.variogram</a> passed to <code>plot.default</code> .

**Value**

Produces a plot with the sample variogram.

**References**

**geoR** package, [variogramCMB](#), [variog](#), [plot.variogram](#)

**Examples**

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 100000)
# varcmb <- variogramCMB(cmbdf, max.dist = 0.1, num.bins = 30, sample.size=1000)
# plot(varcmb)
```

---

plotAngDis

*Plot angular scatterplots and means*

---

**Description**

For specified measurements from [CMBDataFrame](#) this function produces scatterplots and binned means versus theta and phi angles.

**Usage**

```
plotAngDis(cmbdf, intensities = "I")
```

**Arguments**

**cmbdf**            A [CMBDataFrame](#) object.

**intensities**    The name of a column of cmbdf, containing measured values.

**Value**

2x2 plot. The first row shows scatterplots. The second row gives barplots of the corresponding means computed over bins. The first column corresponds to the values of theta and the second one is for psi.

**Examples**

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# df.sample <- CMBDataFrame(df, sample.size = 80000)
# win <- CMBWindow(theta = c(pi/4,pi/2,pi/2,pi/4), phi = c(0,0,pi/2,pi/2))
# cmbdf.win <- window(df.sample, new.window = win)
#
# intensities <- "I"
# plotAngDis(cmbdf.win, intensities)
```

---

plotcovmodelCMB

*Plot theoretical CMB Covariance*


---

**Description**

Plots theoretical covariance functions from the list defined in [covmodelCMB](#)

**Usage**

```
plotcovmodelCMB(cov.model = "matern", sigmasq = 1, phi = pi,
  kappa = 0.5, from = 0, to = pi, ...)
```

**Arguments**

cov.model	A type of the correlation function. Available choices are: "matern", "exponential", "spherical", "powered.exponential", "cauchy", "gencauchy", "pure.nugget", "askey", "c2wendland", "c4wendland", "sinpower", "multiquadric". The default is "matern"
sigmasq	The variance parameter as documented in <a href="#">covmodelCMB</a> . The default is 1.
phi	The range parameter as documented in <a href="#">covmodelCMB</a> . The default is pi.
kappa	A smoothness parameter of the correlation function. The default is 0.5.
from	A lower range of the plotting region. The default is lb=0
to	An upper range of the plotting region. The default is ub= pi.
...	optional plotting parameters.

**Value**

Produces a plot with the theoretical covariance function.

**References**

[covmodelCMB](#)

**Examples**

```
plotcovmodelCMB("matern", sigmasq = 5)
plotcovmodelCMB("askey", phi = pi/4, to = pi/2, kappa = 4)
```

---

plotvariogram

*Plot theoretical variogram*


---

**Description**

Plots theoretical variogram functions from the list defined in [covmodelCMB](#)

**Usage**

```
plotvariogram(cov.model = "matern", sigmasq = 1, phi = pi,
              kappa = 0.5, from = 0, to = pi, ...)
```

**Arguments**

cov.model	A type of the variogram function. Available choices are: "matern", "exponential", "spherical", "powered.exponential", "cauchy", "gencauchy", "pure.nugget", "askey", "c2wendland", "c4wendland", "sinepower", "multiquadric". The default is "matern"
sigmasq	The variance parameter as documented in <a href="#">covmodelCMB</a> . The default is 1.
phi	The range parameter as documented in <a href="#">covmodelCMB</a> . The default is pi.
kappa	A smoothness parameter of the variogram function. The default is 0.5.
from	A lower range of the plotting region. The default is lb = 0
to	An upper range of the plotting region. The default is ub = pi.
...	optional plotting parameters.

**Value**

Produces a plot with the theoretical variogram.

**References**

[covmodelCMB](#)

**Examples**

```
plotvariogram("matern", sigmasq = 5)
plotvariogram("askey", phi = pi/4, to = pi/2, kappa = 4)
```

---

practicalRangeCMB      *Practical range for covariance function*


---

### Description

This function computes the practical range for covariance functions on spheres. The function extends [practicalRange](#) from the package **geoR** to additional covariance models on spheres.

### Usage

```
practicalRangeCMB(cov.model, phi, kappa = 0.5, correlation = 0.05, ...)
```

### Arguments

cov.model	A type of the correlation function. Available choices are: "matern", "exponential", "spherical", "powered.exponential", "cauchy", "gencauchy", "pure.nugget", "askey", "c2wendland", "c4wendland", "sinpower", "multiquadric".
phi	The range parameter as documented in <a href="#">covmodelCMB</a>
kappa	A smoothness parameter of the correlation function.
correlation	A correlation threshold (default is 0.05)
...	other optimisation parameters

### Details

The practical(effective) range for a covariance function is the distance at which a covariance function first time reaches the specified value `correlation`. For covariance functions on spheres the practical range does not exceed  $\pi$ , the distance beyond which a covariance function is not defined. For the covariance functions "spherical", "askey", "c2wendland", "c4wendland" their practical ranges are equal to lengths of their support.

### Value

Value of the practical range for the covariance function specified in [covmodelCMB](#)

### References

**geoR** package, [practicalRange](#), [covmodelCMB](#)

### Examples

```
practicalRangeCMB(cov.model = "sinpower", phi = 0.1, kappa = 0.5)
practicalRangeCMB(cov.model = "askey", phi = 0.1, kappa = 0.5)
```

print.CMBDataFrame     *Print CMBDataFrame*

---

### Description

This function neatly prints the contents of a CMBDataFrame.

### Usage

```
## S3 method for class 'CMBDataFrame'  
print(x, ...)
```

### Arguments

x                    A [CMBDataFrame](#).  
...                   arguments passed to [print.tbl\\_df](#)

### Value

Prints contents of the CMB data frame to the console.

### Examples

```
## First download the map  
# downloadCMBMap(foreground = "smica", nside = 1024)  
# df <- CMBDataFrame("CMB_map_smica1024.fits")  
# print(df)
```

---

print.HPDataFrame     *Print a [HPDataFrame](#)*

---

### Description

This function neatly prints the contents of a HPDataFrame.

### Usage

```
## S3 method for class 'HPDataFrame'  
print(x, ...)
```

### Arguments

x                    A [HPDataFrame](#).  
...                   arguments passed to [print.tbl\\_df](#)



**Value**

Prints contents of the HPDataFrame to the console.

**Examples**

```
df <- HPDataFrame(I = rep(0,12), nside = 1, ordering = "nested")
print(df)
df
```

---

pwSpCorr

*Power spectra estimate via correlation*

---

**Description**

This function provides an angular power spectra estimate using the values of the sample correlations. The approach is based on Lawson-Hanson algorithm for non-negative least squares.

**Usage**

```
pwSpCorr(corcmb, lmax = 20 * length(corcmb$u))
```

**Arguments**

corcmb            An object of the class CMBCorrelation.  
lmax              A number of angular power spectra components to estimate

**Value**

A data frame which first column is 1-d grid of l values from 0 to lmax. The second column is estimated angular power spectra components on this grid.

**References**

Formula (2.1) in Baran A., Terdik G. Power spectrum estimation of spherical random fields based on covariances. *Annales Mathematicae et Informaticae* 44 (2015) pp. 15–22.

**Examples**

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# Corrf <- corrCMB(df, max.dist = 0.1, num.bins = 30,
# sample.size=10000)
# pw <- pwSpCorr(Corrf)
```

---

qqnormWin	<i>Normal QQ plot for CMBWindow</i>
-----------	-------------------------------------

---

### Description

This function is a modification of standard [qqnorm](#) functions to work with [CMBWindow](#) regions.

### Usage

```
qqnormWin(cmbdf, win, intensities = "I")
```

### Arguments

cmbdf	A <a href="#">CMBDataFrame</a> .
win	A <a href="#">CMBWindow</a>
intensities	A <a href="#">CMBDataFrame</a> column with measured values.

### Details

[qqnormWin](#) returns a normal QQ plot of for the specified [CMBDataFrame](#) column [intensities](#) and [CMBWindow](#) region. The function automatically adds a line of a “theoretical” normal quantile-quantile plot.

### Value

A list with quantile components x and y and a normal QQ plot with QQ line

### References

[qqnorm](#), [qqplot](#), [qqplotWin](#)

### Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 1000)

# win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
# qqnormWin(cmbdf, win1)
```

---

`qqplotWin`*Quantile-Quantile plots for CMBWindows*

---

### Description

This function is a modification of standard `qqplot` functions to work with `CMBWindow` regions.

### Usage

```
qqplotWin(cmbdf, win1, win2, intensities = "I")
```

### Arguments

<code>cmbdf</code>	A <code>CMBDataFrame</code> .
<code>win1</code>	A <code>CMBWindow</code>
<code>win2</code>	A <code>CMBWindow</code>
<code>intensities</code>	A <code>CMBDataFrame</code> column with measured values.

### Details

`qqplotWin` produces a QQ plot of quantiles of observations in two `CMBWindows` against each other for the specified `CMBDataFrame` column intensities. The function automatically adds a diagonal line.

### Value

A list with quantile components x and y and a QQ plot with a diagonal line

### References

[qqnormWin](#), [qqnorm](#), [qqplot](#)

### Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 10000)

# win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
# win2 <- CMBWindow(theta = c(2*pi/3,3*pi/4,3*pi/4, 2*pi/3),
#                   phi = c(pi/4,pi/4,pi/3,pi/3))

# qqplotWin(cmbdf, win1, win2)
```

---

qstat	<i>q-statistic</i>
-------	--------------------

---

### Description

This function returns an estimated q-statistic for the specified column intensities in a [CMBDataFrame](#) and the list of [CMBWindows](#).

### Usage

```
qstat(cmbdf, listwin, intensities = "I")
```

### Arguments

cmbdf	A <a href="#">CMBDataFrame</a> .
listwin	A list of <a href="#">CMBWindows</a>
intensities	A <a href="#">CMBDataFrame</a> column with measured values.

### Details

The q-statistics is used to measure spatial stratified heterogeneity and takes values in [0, 1]. It gives the percent of the variance of intensities explained by the stratification. 0 corresponds to no spatial stratified heterogeneity, 1 to perfect spatial stratified heterogeneity.

### Value

Estimated q-statistics for observations in a list of [CMBWindows](#)

### References

Wang, J.F, Zhang, T.L, Fu, B.J. (2016). A measure of spatial stratified heterogeneity. *Ecological Indicators*. 67: 250–256.

### Examples

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 1000)
# win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
# win2 <- CMBWindow(theta = c(pi/2,pi,pi/2), phi = c(0,0,pi/2))
#
# lw <- list(win1, win2)
# qstat(cmbdf, lw)
```

---

rbind.CMBDataFrame     [rbind](#) for CMBDataFrames

---

## Description

Add a new row or rows to a [CMBDataFrame](#). All arguments passed to `...` must be [CMBDataFrames](#). If the [CMBDataFrame](#) arguments have overlapping pixel indices then all but one of the non-unique rows will be deleted unless `unsafe = TRUE`. If `unsafe = TRUE` then a [HPDataFrame](#) will be returned instead of a [CMBDataFrame](#).

## Usage

```
## S3 method for class 'CMBDataFrame'  
rbind(..., deparse.level = 1, unsafe = FALSE)
```

## Arguments

`...`                    A number of [CMBDataFrames](#)

`deparse.level`        See documentation for [rbind.data.frame](#).

`unsafe`                A boolean. If the [CMBDataFrame](#) arguments have overlapping pixel indices then all but one of the non-unique rows will be deleted unless `unsafe = TRUE`. If `unsafe = TRUE` then a [HPDataFrame](#) will be returned instead of a [CMBDataFrame](#).

## See Also

See the documentation for [rbind](#)

## Examples

```
df <- CMBDataFrame(nside = 1, I = 1:12)  
  
df.123 <- CMBDataFrame(df, spix = c(1,2,3))  
df.123  
df.234 <- CMBDataFrame(df, spix = c(2,3,4))  
df.234  
  
df.1234 <- rbind(df.123, df.234)  
df.1234  
class(df.1234) # A CMBDataFrame  
pix(df.1234)  
  
df.123234 <- rbind(df.123, df.234, unsafe = TRUE)  
df.123234  
class(df.123234) # A HPDataFrame  
pix(df.123234)
```

## Description

Handling and Analysing Spherical, Healpix and Cosmic Microwave Background data on a HEALPix grid.

## Details

The package `rcosmo` offers various tools for

- Downloading and transforming Cosmic Microwave Background radiation (CMB) and spherical data
- Working with Hierarchical Equal Area isoLatitude Pixelation of a sphere (Healpix)
- Spherical geometry
- Statistical analysis of CMB and spherical data
- Visualisation of Healpix data

Most of `rcosmo` features were developed for CMB, but it can also be used for other spherical data. It contains tools for transforming spherical data in cartesian and geographic coordinates to the Healpix representation.

## Update

Current updates are available through URL: <https://github.com/frycast/rcosmo>

## BugReports

<https://github.com/frycast/rcosmo/issues>

## Author(s)

Daniel Fryer <[d.fryer@latrobe.edu.au](mailto:d.fryer@latrobe.edu.au)>, Andriy Olenko <[a.olenko@latrobe.edu.au](mailto:a.olenko@latrobe.edu.au)>, Ming Li <[Ming.Li@latrobe.edu.au](mailto:Ming.Li@latrobe.edu.au)>, Yuguang Wang.

---

resolution	<i>Get the arcmin resolution from a <a href="#">CMBDataFrame</a></i>
------------	--

---

**Description**

Get the arcmin resolution from a [CMBDataFrame](#)

**Usage**

```
resolution(cmbdf)
```

**Arguments**

cmbdf            a [CMBDataFrame](#).

**Value**

The arcmin resolution as specified by the FITS file where the data was sourced

**Examples**

```
## First download the map
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# resolution(df)
```

---

ring2nest	<i>Convert ring to nest ordering.</i>
-----------	---------------------------------------

---

**Description**

ring2nest converts HEALPix pixel indices in the 'ring' ordering scheme to HEALPix pixel indices in the 'nested' ordering scheme.

**Usage**

```
ring2nest(nside, pix)
```

**Arguments**

nside            is the HEALPix nside parameter (integer number  $2^k$ )  
pix                is a vector of HEALPix pixel indices, in the 'ring' ordering scheme.

**Value**

the output is a vector of HEALPix pixel indices in the 'nested' ordering scheme.

**Examples**

```
## Convert (1,2,23) from ring to nest at nside = 8
nside <- 8
pix <-c(1,2,23)
ring2nest(nside,pix)
```

---

sampleCMB

*Take a simple random sample from a [CMBDataFrame](#)*

---

**Description**

This function returns a [CMBDataFrame](#) which size equals to `sample.size`, whose rows comprise a simple random sample of the rows from the input `CMBDataFrame`.

**Usage**

```
sampleCMB(cmbdf, sample.size)
```

**Arguments**

`cmbdf`            a [CMBDataFrame](#).  
`sample.size`      the desired sample size.

**Value**

A [CMBDataFrame](#) which size equals to `sample.size`, whose rows comprise a simple random sample of the rows from the input `CMBDataFrame`.

**Examples**

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# plot(sampleCMB(df, sample.size = 800000))

df <- CMBDataFrame(nside = 16, I = rnorm(12 * 16 ^ 2), ordering = "nested")
df.sample <- sampleCMB(df, sample.size = 100)
df.sample
```



---

siblings	<i>Return siblings of pixel</i>
----------	---------------------------------

---

**Description**

The siblings of pixel  $p$  are defined as the children of the parent of  $p$ . Note this is resolution independent.

**Usage**

```
siblings(p)
```

**Arguments**

$p$	Pixel index in nested order.
-----	------------------------------

**Examples**

```
siblings(11)
siblings(12)
```

---

sphericalHarmonics	<i>Compute spherical harmonic values at given points on the sphere.</i>
--------------------	---

---

**Description**

The function sphericalHarmonics computes the spherical harmonic values for the given 3D Cartesian coordinates.

**Usage**

```
sphericalHarmonics(L, m, xyz)
```

**Arguments**

$L$	The degree of spherical harmonic ( $L=0,1,2,\dots$ )
$m$	The order number of the degree- $L$ spherical harmonic ( $m=-L,-L+1,\dots,L-1,L$ )
$xyz$	Dataframe for given points in 3D cartesian coordinates

**Value**

values of spherical harmonics

## References

See [https://en.wikipedia.org/wiki/Table\\_of\\_spherical\\_harmonics](https://en.wikipedia.org/wiki/Table_of_spherical_harmonics)

It uses equation (7) in Hesse, K., Sloan, I. H., & Womersley, R. S. (2010). Numerical integration on the sphere. In Handbook of Geomathematics (pp. 1185-1219). Springer Berlin Heidelberg, but instead of the order  $k=1, \dots, 2L+1$  in the book we use  $m=k-L-1$ .

## Examples

```
## Calculate spherical harmonic value at
## the point (0,1,0) with L=5, m=2
point<-data.frame(x=0,y=1,z=0)
sphericalHarmonics(5,2,point)

## Calculate spherical harmonic values at
## the point (1,0,0), (0,1,0), (0,0,1) with L=5, m=2
points<-data.frame(diag(3))
sphericalHarmonics(5,2,points)
```

---

summary.CMBDataFrame    *Summarise a* [CMBDataFrame](#)

---

## Description

This function produces a summary from a CMBDataFrame.

## Usage

```
## S3 method for class 'CMBDataFrame'
summary(object, intensities = "I", ...)
```

## Arguments

object	A <a href="#">CMBDataFrame</a> .
intensities	the name of a column specifying CMB intensities (or potentially another numeric quantity of interest)
...	Unused arguments.

## Value

A summary includes window's type and area, total area covered by observations, and main statistics for intensity values

**Examples**

```
## First download the map
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# df.sample <- CMBDataFrame(df, sample.size = 800000)
# summary(df.sample)

ns <- 16
df <- CMBDataFrame(I = rnorm(12*ns^2), nside = ns,
                   ordering = "nested")

win1 <- CMBWindow(x=0,y=3/5,z=4/5,r=0.8)
df.sample1 <- window(df, new.window = win1)
summary(df)
```

---

summary.CMBWindow      *Summarise a CMBWindow*

---

**Description**

This function produces a summary from a CMBWindow

**Usage**

```
## S3 method for class 'CMBWindow'
summary(object, ...)
```

**Arguments**

object            A CMBWindow.  
...                Unused arguments.

**Value**

A summary includes window's type and area

**Examples**

```
win <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
summary(win)

win1 <- CMBWindow(x=0,y=3/5,z=4/5,r=0.8, set.minus = TRUE)
summary(win1)
```

---

summary.HPDataFrame     *Summarise a [HPDataFrame](#)*

---

### Description

This function produces a summary from a HPDataFrame.

### Usage

```
## S3 method for class 'HPDataFrame'
summary(object, intensities = "I", ...)
```

### Arguments

object	A HPDataFrame.
intensities	the name of a column specifying intensities (or potentially another numeric quantity of interest)
...	Unused arguments.

### Value

A summary includes window's type and area, total area covered by observations, and main statistics for intensity values

### Examples

```
ns <- 2
hpdf <- HPDataFrame(I = rnorm(12*ns^2), nside = 2,
                    ordering = "nested")
win <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
hpdf.win <- window(hpdf, new.window = win)
summary(hpdf.win)
```

---

triangulate     *Triangulate a polygonal [CMBWindow](#)*

---

### Description

Triangulate a polygonal [CMBWindow](#)

### Usage

```
triangulate(win)
```

**Arguments**

win                    a CMBWindow object

**Value**

a list of CMBWindow polygons or minus.polygons, each having 3 vertices and representing a triangle. If winType of win does not include "minus" then these triangles have pairwise disjoint interiors and their union is equal to the original polygon, win. Otherwise, if winType of win does include "minus" the triangles are the same as for the non-minus type above, but have "minus" types.

**Examples**

```
## Example 1

win <- CMBWindow(theta = c(2*pi/3,3*pi/4,3*pi/4, 2*pi/3),
                  phi = c(pi/4,pi/4,pi/3,pi/3))

win
plot(win)
win1 <- triangulate(win)
win1
summary(win1[[1]])
plot(win1[[1]], add= FALSE, col="green")
plot(win1[[2]], col="blue")

## Example 2: triangulation minus-type polygon

win <- CMBWindow(theta = c(pi/5,pi/3,pi/4, pi/3, pi/5),
                  phi = c(pi/5,pi/5, pi/4 ,pi/3,pi/3), set.minus =TRUE)

win
plot(win)
summary(win)
win1 <- triangulate(win)
win1
plot(win1[[1]], add= FALSE, col="green")
plot(win1[[2]], col="blue")
plot(win1[[3]], col="yellow")
summary(win1[[1]])
summary(win1[[2]])
summary(win1[[3]])
```

**Description**

This function estimates variogram parameters by fitting a parametric model from [covmodelCMB](#) to a sample variogram. The function extends [variofit](#) from the package **geoR** to additional covariance models on spheres.

**Usage**

```
variofitCMB(vario, ini.cov.pars, cov.model, fix.nugget = FALSE,
  nugget = 0, fix.kappa = TRUE, kappa = 0.5, simul.number = NULL,
  max.dist = vario$max.dist, weights, minimisation.function,
  limits = geoR::pars.limits(), messages, ...)
```

**Arguments**

vario	An object of the class <code>variogram</code> obtained as an output of the function <code>variogramCMB</code> .
ini.cov.pars	A vector with initial values for the variogram parameters. The first parameter corresponds to the variance $\sigma^2$ . The second parameter corresponds to the range $\phi$ of the correlation function.
cov.model	A type of the variogram function. Available choices are: "matern", "exponential", "spherical", "powered.exponential", "cauchy", "gencauchy", "pure.nugget", "askey", "c2wendland", "c4wendland", "sinpower", "multiquadric". The default is "matern"
fix.nugget	logical. Indicates whether the nugget variance should be regarded as fixed or be estimated. The default is FALSE.
nugget	A value for the nugget parameter. Regarded as a fixed values if <code>fix.nugget = TRUE</code> or as a initial value for the minimization algorithm if <code>fix.nugget = FALSE</code> . The default is zero.
fix.kappa	logical. Indicates whether the parameter $\kappa$ should be regarded as fixed or be estimated. The default is TRUE.
kappa	A value for the smoothness parameter. Regarded as a fixed values if <code>fix.kappa = TRUE</code> or as a initial value for the minimization algorithm if <code>fix.kappa = FALSE</code> . Required not in all covariance models, see <code>covmodelCMB</code> . The default is 0.5.
simul.number	number of simulation. Used if <code>vario</code> has empirical variograms for more than one data-set (simulations). The default is NULL
max.dist	A maximum distance to fit a variogram model. The default is <code>x\$max.dist</code> .
weights	Weights used in the loss function in the minimization algorithm.
minimisation.function	Minimization function ("optim", "nlm", "nls") to estimate the parameters.
limits	Lower and upper limits for the model parameters used in the numerical minimization by <code>minimisation.function = "optim"</code> .
messages	logical. Indicates whether or not status messages are printed on the screen.
...	other minimisation parameters

**Details**

The parameter values of a variogram function from `covmodelCMB` are found by numerical optimization using one of the functions: `optim`, `nlm` and `nls`.

The function extends `variofit` from the package `geoR` to additional variogram models on spheres. Available models are: "matern", "exponential", "spherical", "powered.exponential", "cauchy", "gencauchy", "pure.nugget", "askey", "c2wendland", "c4wendland", "sinpower", "multiquadric".

Additionally it rescales an empirical variogram to the range  $[0,1]$  before numerical optimisation and then transforms all obtained results to the original scale. If `ini.cov.pars` are not provided then the  $5 \times 5$  grid (`seq(0,max(vario$v),l=5),seq(0,vario$max.dist,l=5)`) of initial values of  $\sigma^2$  and  $\phi$  is used.

### Value

An object of the class `variomodel` and `variofit`, see [variofit](#)

### References

**geoR** package, [variofit](#), [covmodelCMB](#)

### Examples

```
#
# df <- CMBDataFrame("../CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 10000)
# varcmb <- variogramCMB(cmbdf, max.dist = 0.1, num.bins = 30)
# varcmb
#
# ols <- variofitCMB(varcmb, fix.nug=FALSE, wei="equal", cov.model= "matern")
# plot(varcmb)
# lines(ols, lty=2)
# str(ols)
#
# ols <- variofitCMB(varcmb, fix.nug = TRUE, kappa = 3, wei = "equal",
# cov.model = "askey")
# plot(varcmb, main = ols$cov.model)
# linesCMB(ols, lty = 2)
# str(ols)
```

---

variogramCMB

*Sample variogram*

---

### Description

This function provides an empirical variogram for data in a [CMBDataFrame](#) or `data.frame`. It assumes that data are from a stationary spherical random field and the covariance depends only on a geodesic distance between locations. Output is a binned variogram.

### Usage

```
variogramCMB(cmbdf, num.bins = 10, sample.size, max.dist = pi, breaks,
  equiareal = TRUE, calc.max.dist = FALSE)
```

**Arguments**

<code>cmbdf</code>	is a <code>CMBDataFrame</code> or <code>data.frame</code>
<code>num.bins</code>	specifies the number of bins
<code>sample.size</code>	optionally specify the size of a simple random sample to take before calculating variogram. This may be useful if the full covariance computation is too slow.
<code>max.dist</code>	an optional number between 0 and pi specifying the maximum geodesic distance to use for calculating covariance. Only used if breaks are unspecified.
<code>breaks</code>	optionally specify the breaks manually using a vector giving the break points between cells. This vector has length <code>num.bins</code> since the last break point is taken as <code>max.dist</code> . If <code>equiareal = TRUE</code> then these breaks should be $\cos(r_i)$ where $r_i$ are radii. If <code>equiareal = FALSE</code> then these breaks should be $r_i$ .
<code>equiareal</code>	if TRUE then the bins have equal spherical area. If false then the bins have equal annular widths. Default is TRUE.
<code>calc.max.dist</code>	if TRUE then the <code>max.dist</code> will be calculated from the locations in <code>cmbdf</code> . Otherwise either <code>max.dist</code> must be specified or <code>max.dist</code> will default to pi.

**Value**

An object of class `variog` specified in the package `geoR`.

The attribute "breaks" contains the break points used to create bins. The result has `num.bins + 1` values since the first value at distance 0 is not counted as a bin.

**u** a vector with distances.

**v** a vector with estimated variogram values at distances given in u.

**n** number of pairs in each bin

**sd** standard deviation of the values in each bin

**bins.lim** limits defining the interval spanned by each bin

**ind.bin** a logical vector indicating whether the number of pairs in each bin is greater or equal to the value in the argument `pairs.min`

**var.mark** variance of the data

**beta.ols** parameters of the mean part of the model fitted by ordinary least squares

**output.type** echoes the option argument

**max.dist** maximum distance between pairs allowed in the variogram calculations

**n.data** number of data

**direction** direction for which the variogram was computed

**call** the function call

**References**

`geoR` package, `variog`, `covCMB`, `corrCMB`



**Examples**

```
## Download the map first
# downloadCMBMap(foreground = "smica", nside = 1024)
# df <- CMBDataFrame("CMB_map_smica1024.fits")
# cmbdf <- sampleCMB(df, sample.size = 100000)
# varcmb <- variogramCMB(cmbdf, max.dist = 0.1, num.bins = 30, sample.size=100)
# varcmb
```

---

window	<i>window generic</i>
--------	-----------------------

---

**Description**

Detailed descriptions and examples can be found in documentation for specific window functions [window.CMBDataFrame](#), [window.HPDataFrame](#), [window.data.frame](#)

**Usage**

```
window(x, ...)
```

**Arguments**

x	An object.
...	Extra arguments.

**See Also**

[window.CMBDataFrame](#) [window.HPDataFrame](#) [window.data.frame](#)

---

window.CMBDat	<i>Get a sub window from a CMBDat object</i>
---------------	--

---

**Description**

This function returns a data.frame containing the data in x restricted to the CMBWindow `new.window`

**Usage**

```
## S3 method for class 'CMBDat'
window(x, new.window, intersect = TRUE, ...)
```

**Arguments**

<code>x</code>	a <code>CMBDat</code> object.
<code>new.window</code>	A single <code>CMBWindow</code> object or a list of them.
<code>intersect</code>	A boolean that determines the behaviour when <code>new.window</code> is a list containing BOTH regular type and "minus" type windows together (see details).
<code>...</code>	Unused arguments.

**Details**

Windows that are tagged with `set.minus` (see `CMBWindow`) are treated differently from other windows.

If the argument is a list of `CMBWindows`, then interiors of all windows whose `winType` does not include "minus" are united (let  $A$  be their union) and exteriors of all windows whose `winType` does include "minus" are intersected, (let  $B$  be their intersection). Then, provided that `intersect = TRUE` (the default), the returned data.frame will be the points of `cmbdat$data` in the the intersection of  $A$  and  $B$ . Otherwise, if `intersect = FALSE`, the returned data.frame consists of the points of `x$data` in the union of  $A$  and  $B$ .

Note that if  $A$  (resp.  $B$ ) is empty then the returned data.frame will be the points of  $x$  in  $B$  (resp.  $A$ ).

**Value**

A `CMBDataFrame` containing the data in `x` restricted to the `CMBWindow` `new.window`

**Examples**

```
win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))

## Ensure you have a FITS file with correct path
## before uncommenting and running the rest of the example:
# cmbdat <- CMBDat("CMB_map_smica1024.fits", mmap = TRUE)
# class(cmbdat)
# cmbdat.win <- window(cmbdat, new.window = win1)
# class(cmbdat.win)
```

---

`window.CMBDataFrame`    *Get a sub window from `CMBDataFrame`*

---

**Description**

When `new.window` or `in.pixels` is unspecified this function returns the `CMBWindow` attribute of a `CMBDataFrame`. The return value is `NULL` if the window is full sky. When `new.window` is specified this function instead returns a new `CMBDataFrame` whose `CMBWindow` attribute is `new.window`

**Usage**

```
## S3 method for class 'CMBDataFrame'
window(x, new.window, intersect = TRUE, in.pixels,
       in.pixels.res = 0, ...)
```

**Arguments**

<code>x</code>	A <a href="#">CMBDataFrame</a> .
<code>new.window</code>	Optionally specify a new window in which case a new <a href="#">CMBDataFrame</a> is returned whose <code>CMBWindow</code> is <code>new.window</code> . <code>new.window</code> may also be a list (see details section and examples).
<code>intersect</code>	A boolean that determines the behaviour when <code>new.window</code> is a list containing BOTH regular type and "minus" type windows together (see details).
<code>in.pixels</code>	A vector of pixels at resolution <code>in.pixels.res</code> whose union contains the window(s) <code>new.window</code> entirely, or if <code>new.window</code> is unspecified then this whole pixel is returned.
<code>in.pixels.res</code>	An integer. Resolution (i.e., $j$ such that $n_{side} = 2^j$ ) at which the <code>in.pixels</code> parameter is specified
<code>...</code>	Unused arguments.

**Details**

Windows that are tagged with `set.minus` (see [CMBWindow](#)) are treated differently from other windows.

If the argument `new.window` is a list of [CMBWindows](#), then interiors of all windows whose `winType` does not include "minus" are united (let  $A$  be their union) and exteriors of all windows whose `winType` does include "minus" are intersected, (let  $B$  be their intersection). Then, provided that `intersect = TRUE` (the default), the returned [CMBDataFrame](#) will be the points of `cmbdf` in the the intersection of  $A$  and  $B$ . Otherwise, if `intersect = FALSE`, the returned [CMBDataFrame](#) consists of the points of `x` in the union of  $A$  and  $B$ .

Note that if  $A$  (resp.  $B$ ) is empty then the returned [CMBDataFrame](#) will be the points of `x` in  $B$  (resp.  $A$ ).

**Value**

The window attribute of `x` or, if `new.window/in.pixels` is specified, a new [CMBDataFrame](#).

**Examples**

```
## Example 1: Create a new CMBDataFrame with a window

cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian",
                    ordering = "nested")
win <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
cmbdf.win <- window(cmbdf, new.window = win)
```

```

plot(cmbdf.win)
window(cmbdf.win)

## Example 2: Change the window of an existing CMBDataFrame

cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian", ordering = "nested")
window(cmbdf) <- win2 <- CMBWindow(theta = c(pi/6,pi/3,pi/3, pi/6),
                                   phi = c(0,0,pi/6,pi/6))

plot(cmbdf)

## Example 3: union of windows

## Create 2 windows
win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
win2 <- CMBWindow(theta = c(2*pi/3,3*pi/4,3*pi/4, 2*pi/3),
                 phi = c(pi/4,pi/4,pi/3,pi/3))

plot(win1)
plot(win2)

## Create CMBDataFrame with points in the union of win1 and win2

cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian", ordering = "nested")
cmbdf.win <- window(cmbdf, new.window = list(win1, win2), intersect = FALSE)
plot(cmbdf.win)

## Example 4: intersection of windows

## Create 2 windows
win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
win2 <- CMBWindow(theta = c(pi/4,pi/3,pi/3, pi/4),
                 phi = c(pi/4,pi/4,pi/3,pi/3))

plot(win1)
plot(win2)

## Create CMBDataFrame with points in the intersection of win1 and win2

cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian", ordering = "nested")
cmbdf.win1 <- window(cmbdf, new.window = win1)
cmbdf.win12 <- window(cmbdf.win1, new.window = win2)
plot(cmbdf.win12)
plot(win1)
plot(win2)

## Example 5: intersection of windows with "minus" type

## Create 2 windows with "minus" type
win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2),
                 set.minus =TRUE)
win2 <- CMBWindow(theta = c(pi/4,pi/3,pi/3, pi/4),
                 phi = c(pi/4,pi/4,pi/3,pi/3),
                 set.minus =TRUE)

plot(win1)

```

```

plot(win2)

## Create CMBDataFrame with points in the intersection of win1 and win2

cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian", ordering = "nested")
cmbdf.win <- window(cmbdf, new.window = list(win1, win2))
plot(cmbdf.win)

## Example 6: intersection of windows with different types

##Create 2 windows, one with "minus" type

win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
win2 <- CMBWindow(theta = c(pi/4,pi/3,pi/3, pi/4),
                  phi = c(pi/4,pi/4,pi/3,pi/3),
                  set.minus =TRUE)

plot(win1)
plot(win2)

## Create CMBDataFrame with points in the intersection of win1 and win2

cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian", ordering = "nested")
cmbdf.win <- window(cmbdf, new.window = list(win1, win2), intersect = TRUE)
plot(cmbdf.win)

## Example 7: union of windows with different types

win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2), set.minus =TRUE)
win2 <- CMBWindow(theta = c(pi/4,pi/3,pi/3, pi/4), phi = c(pi/4,pi/4,pi/3,pi/3))
plot(win1)
plot(win2)

## Create CMBDataFrame with points in the union of win1 and win2

cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian", ordering = "nested")
cmbdf.win <- window(cmbdf, new.window = list(win1, win2), intersect = FALSE)
plot(cmbdf.win)

```

---

window.data.frame

*Get a sub window from a data.frame*


---

## Description

This function returns a data.frame containing the data in `x` restricted to the `CMBWindow` `new.window`

**Usage**

```
## S3 method for class 'data.frame'
window(x, new.window, intersect = TRUE, ...)
```

**Arguments**

<code>x</code>	A data.frame. Must have columns labelled x,y,z specifying cartesian coordinates, or columns labelled theta, phi specifying colatitude and longitude respectively.
<code>new.window</code>	A single <a href="#">CMBWindow</a> object or a list of them.
<code>intersect</code>	A boolean that determines the behaviour when <code>new.window</code> is a list containing BOTH regular type and "minus" type windows together (see details).
<code>...</code>	Unused arguments.

**Details**

Windows that are tagged with `set.minus` (see [CMBWindow](#)) are treated differently from other windows.

If the argument is a list of [CMBWindows](#), then interiors of all windows whose `winType` does not include "minus" are united (let  $A$  be their union) and exteriors of all windows whose `winType` does include "minus" are intersected, (let  $B$  be their intersection). Then, provided that `intersect = TRUE` (the default), the returned data.frame will be the points of  $x$  in the intersection of  $A$  and  $B$ . Otherwise, if `intersect = FALSE`, the returned data.frame consists of the points of  $x$  in the union of  $A$  and  $B$ .

Note that if  $A$  (resp.  $B$ ) is empty then the returned data.frame will be the points of  $x$  in  $B$  (resp.  $A$ ).

**Value**

A data.frame containing the data in `x` restricted to the [CMBWindow](#) `new.window`

**Examples**

```
win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))

cmbdf <- CMBDataFrame(nside = 4)
df2 <- coords(cmbdf, new.coords = "cartesian")
df <- as.data.frame(df2[,1:3])
df
df.win <- window(df, new.window = win1)
df.win
```

---

window.HPDataFrame      *Get a sub window from a [HPDataFrame](#)*

---

### Description

This function returns a HPDataFrame containing the data in hpdf restricted to the CMBWindow new.window. If the HPDataFrame has columns x,y,z or theta, phi then these will be used to determine locations with priority over the HEALPix indices in pix(hpdf) unless healpixCentered = TRUE is given. Note that if healpixCentered = TRUE then columns x,y,z or theta, phi will be discarded and replaced with pixel center locations.

### Usage

```
## S3 method for class 'HPDataFrame'
window(x, new.window, intersect = TRUE,
       healpixCentered = FALSE, ...)
```

### Arguments

x	A <a href="#">HPDataFrame</a> .
new.window	Optional. A single <a href="#">CMBWindow</a> object or a list of them.
intersect	A boolean that determines the behaviour when new.window is a list containing BOTH regular type and "minus" type windows together (see details).
healpixCentered	A boolean. If the HPDataFrame has columns x,y,z or theta, phi then these will be used to determine locations with priority over the HEALPix indices in pix(x) unless healpixCentered = TRUE is given. Note that if healpixCentered = TRUE then columns x,y,z or theta, phi will be discarded and replaced with pixel center locations.
...	Unused arguments.

### Details

Windows that are tagged with set.minus (see [CMBWindow](#)) are treated differently from other windows.

If the argument is a list of CMBWindows, then interiors of all windows whose winType does not include "minus" are united (let  $A$  be their union) and exteriors of all windows whose winType does include "minus" are intersected, (let  $B$  be their intersection). Then, provided that intersect = TRUE (the default), the returned data.frame will be the points of df in the the intersection of  $A$  and  $B$ . Otherwise, if intersect = FALSE, the returned data.frame consists of the points of df in the union of  $A$  and  $B$ .

Note that if  $A$  (resp.  $B$ ) is empty then the returned data.frame will be the points of df in  $B$  (resp.  $A$ ).

**Value**

A HPDataFrame containing the data in `x` restricted to the CMBWindow `new.window`. Or, if `new.window` is unspecified, then the `window` attribute of `x` is returned instead (and may be NULL).

**Examples**

```
ns <- 16
hpdf <- HPDataFrame(nside = ns, I = 1:(12*ns^2))
hpdf

win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
plot(hpdf); plot(win1)

hpdf.win <- window(hpdf, new.window = win1)
plot(hpdf.win, col = "yellow", size = 4, add = TRUE)
attributes(hpdf.win)
window(hpdf.win)
hpdf.win
```

---

winType

*Get/change winType*


---

**Description**

Get/change the `winType` (polygon or disk) of a `CMBWindow`. If `new.type` is missing then the `winType` of `win` is returned. Otherwise, a new window is returned with `winType` equal to `new.type`. If you want to change the `winType` of `win` directly, then use `winType<-`, see the examples below.

**Usage**

```
winType(win, new.type)
```

**Arguments**

<code>win</code>	a <code>CMBWindow</code> object or a list of such
<code>new.type</code>	optionally specify a new type. Use this to change between "polygon" and "minus.polygon" or to change between "disc" and "minus.disc"

**Value**

If `new.type` is missing then the `winType` of `win` is returned. Otherwise a new window is returned with `winType` equal to `new.type`



**Examples**

```
win <- CMBWindow(theta = c(pi/2,pi/2,pi/3, pi/3), phi = c(0,pi/3,pi/3,0))
winType(win)
```

```
win1 <- CMBWindow(x=0,y=3/5,z=4/5,r=0.8)
winType(win1)
cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian",
                      ordering = "nested")
cmbdf.win1 <- window(cmbdf, new.window = win1)
plot(cmbdf.win1)
```

```
winType(win1) <- "minus.disc"
winType(win1)
cmbdf <- CMBDataFrame(nside = 64, coords = "cartesian",
                      ordering = "nested")
cmbdf.win1 <- window(cmbdf, new.window = win1)
plot(cmbdf.win1)
```

# Index

ancestor, 4  
areCompatibleCMBDFs, 4  
as.CMBDataFrame, 5  
assumedConvex, 6  
assumedUniquePix, 7  
  
baseNeighbours, 8  
  
cbind, 8, 9  
cbind.CMBDataFrame, 8  
chi2.empirical, 9  
chi2CMB, 9  
children, 10, 41, 52  
CMBDat, 10, 89, 90  
CMBDataFrame, 5, 7–9, 11, 14, 18–20, 27,  
29–33, 36, 39, 40, 47, 49, 53, 55, 59,  
61, 64, 68, 72, 74–77, 79, 80, 82, 87,  
88, 90, 91  
CMBWindow, 6, 9, 12, 13, 15, 29–31, 36, 48, 65,  
74–76, 83, 84, 90, 91, 94–96  
coords, 14  
coords.CMBDataFrame, 14, 14  
coords.CMBWindow, 14, 15  
coords.data.frame, 14, 16  
coords.HPDataFrame, 14, 17  
corrCMB, 18, 21, 62, 88  
cov.spatial, 21–23  
covCMB, 19, 20, 63, 88  
covmodelCMB, 21, 46, 69–71, 85–87  
covPwSp, 23  
curve, 46  
  
data.frame, 5, 11, 19, 20, 35, 38, 88  
displayPixelBoundaries, 24  
displayPixels, 25  
download.file, 27  
downloadCMBMap, 26  
downloadCMBPS, 27  
  
entropy, 29  
  
entropyCMB, 29  
exprob, 30  
extrCMB, 31  
  
fmf, 32  
fRen, 33  
  
geo2sph, 34  
geoAngle, 34  
geoArea, 35  
geoArea.CMBDataFrame, 35, 36  
geoArea.CMBWindow, 35, 36  
geoArea.HPDataFrame, 35, 37  
geoDist, 38  
  
header, 39  
healpixCentered, 39  
HPDataFrame, 7, 12, 17, 18, 37, 39, 40, 45, 47,  
49, 54, 56, 59–61, 66, 72, 77, 84, 95  
  
ibp2p, 43  
is.CMBDat, 43  
is.CMBDataFrame, 44  
is.CMBWindow, 45  
is.HPDataFrame, 45  
  
lines.variomodel.variofit, 46  
linesCMB, 46  
  
maxDist, 47  
maxWindowDist, 48  
minDist, 49  
mmap, 10, 11  
  
neighbours, 50  
nest2ring, 51  
nestSearch, 51  
nlm, 86  
nls, 86  
nside, 53  
nside.CMBDataFrame, 53, 53

nside.HPDataFrame, 53, 54  
numeric2col, 54

optim, 86  
ordering, 55  
ordering.CMBDataFrame, 55, 55  
ordering.HPDataFrame, 55, 56

p2bp, 57  
p2ibp, 57  
palette, 54  
parent, 58  
pix, 58  
pix.CMBDataFrame, 58, 59  
pix.HPDataFrame, 58, 59  
pix2coords, 60  
pixelArea, 61  
pixelWindow, 61  
plot.CMBCorrelation, 62  
plot.CMBCovariance, 63  
plot.CMBDataFrame, 64  
plot.CMBWindow, 65  
plot.HPDataFrame, 66  
plot.variogram, 62, 63, 67, 67, 68  
plotAngDis, 68  
plotcovmodelCMB, 69  
plotvariogram, 70  
practicalRange, 71  
practicalRangeCMB, 71  
print.CMBDataFrame, 72  
print.HPDataFrame, 72  
print.tbl\_df, 72  
pwSpCorr, 73

qqnorm, 74, 75  
qqnormWin, 74, 74, 75  
qqplot, 74, 75  
qqplotWin, 74, 75, 75  
qstat, 76

rbind, 77  
rbind.CMBDataFrame, 77  
rbind.data.frame, 77  
rcosmo, 78, 78  
rcosmo-package (rcosmo), 78  
resolution, 79  
rgl, 65, 67  
rgl.material, 25, 65–67  
ring2nest, 79

sampleCMB, 80  
siblings, 81  
sphericalHarmonics, 81  
subWindow, 13  
summary.CMBDataFrame, 82  
summary.CMBWindow, 83  
summary.HPDataFrame, 84  
suppressMessages, 5

triangulate, 84

variofit, 85–87  
variofitCMB, 46, 85  
variog, 19, 21, 62, 63, 68, 88  
variogramCMB, 19, 21, 68, 86, 87

window, 89  
window.CMBDat, 89  
window.CMBDataFrame, 89, 90  
window.data.frame, 89, 93  
window.HPDataFrame, 89, 95  
winType, 96