

# Package ‘rcosmo’

August 13, 2018

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

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

**Title** R Cosmic Microwave Background Data Analysis

**Version** 0.0.0.9000

**Description** Handling and statistical analysis of Cosmic Microwave Background data on a HEALPix grid.

**Depends** R (>= 3.3.1)

**License** MIT + 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)

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

**LinkingTo** Rcpp

**RoxygenNote** 6.0.1

**VignetteBuilder** R.rsp

**Author** Daniel Fryer [aut, cre],  
Andriy Olenko [aut],  
Yuguang Wang [aut],  
Ming Li [aut]

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

**Archs** x64

## R topics documented:

areCompatibleCMBDFs . . . . .	3
as.CMBDataFrame . . . . .	3
assumedConvex . . . . .	4
assumedConvex<- . . . . .	4
cbind.CMBDataFrame . . . . .	5
CMBDataFrame . . . . .	5
CMBReadFITS . . . . .	6
CMBWindow . . . . .	7
coords.CMBDataFrame . . . . .	8

coords.CMBWindow . . . . .	9
coords.data.frame . . . . .	9
coords.HPDataFrame . . . . .	10
coords<-.CMBDataFrame . . . . .	11
coords<-.CMBWindow . . . . .	11
coords<-.HPDataFrame . . . . .	11
covCMB . . . . .	12
geoArea.CMBDataFrame . . . . .	12
geoArea.CMBWindow . . . . .	13
geoDist . . . . .	13
header . . . . .	14
HPDataFrame . . . . .	14
is.CMBDat . . . . .	15
is.CMBDataFrame . . . . .	15
is.CMBWindow . . . . .	16
JacobiRecursive . . . . .	16
maxDist.CMBDataFrame . . . . .	17
maxDist.CMBWindow . . . . .	17
minDist . . . . .	17
nest2ring . . . . .	18
nestSearch . . . . .	18
nestSearch_step . . . . .	19
nside.CMBDataFrame . . . . .	20
nside.HPDataFrame . . . . .	20
ordering.CMBDataFrame . . . . .	21
ordering.HPDataFrame . . . . .	21
ordering<-.CMBDataFrame . . . . .	22
ordering<-.HPDataFrame . . . . .	22
pix.CMBDataFrame . . . . .	23
pix.HPDataFrame . . . . .	23
pix2coords . . . . .	24
pix<-.CMBDataFrame . . . . .	24
pixelArea . . . . .	25
pixelWindow . . . . .	25
plot.CMBDataFrame . . . . .	26
plot.CMBWindow . . . . .	27
plot.HPDataFrame . . . . .	27
plotHPBoundaries . . . . .	29
print.CMBDataFrame . . . . .	29
print.summary.CMBDataFrame . . . . .	30
print.summary.CMBWindow . . . . .	30
rbind.CMBDataFrame . . . . .	31
rcosmo . . . . .	31
resolution . . . . .	32
ring2nest . . . . .	32
sampleCMB . . . . .	33
SphericalHarmonics . . . . .	33
subWindow . . . . .	34
summary.CMBDataFrame . . . . .	35
summary.CMBWindow . . . . .	35
triangulate . . . . .	36
window . . . . .	36

*areCompatibleCMBDFs* 3

window<- . . . . . 37

winType . . . . . 37

winType<- . . . . . 38

**Index** 39

---

*areCompatibleCMBDFs*      *areCompatibleCMBDFs*

---

## Description

Compare attributes to decide if two CMBDataFrames are compatible

## Usage

```
areCompatibleCMBDFs(cmbdf1, cmbdf2)
```

## Arguments

cmbdf1              a [CMBDataFrame](#)

cmbdf2              a [CMBDataFrame](#)

## 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")
a <- CMBDataFrame(nside = 1, ordering = "nested", coords = "spherical")
areCompatibleCMBDFs(a,b)

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

---

*as.CMBDataFrame*              *as.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 checked to correspond to HEALPix pixel centers. The coordinates must be named either x,y,z (cartesian) or theta, phi (spherical colatitude and longitude respectively).

## Usage

```
as.CMBDataFrame(df, ordering, nside, spix)
```

**Arguments**

df	Any data.frame whose rows are in HEALPix order
ordering	character string that specifies the ordering scheme ("ring" or "nested")
nside	an integer that specifies the Nside (resolution) HEALPix parameter
spix	a 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). However, if spix is left blank and df is a CMBDataFrame, then spix is set equal to pix(df)

**Value**

A CMBDataFrame

---

assumedConvex	<i>Check if a <a href="#">CMBWindow</a> is assumed convex</i>
---------------	---

---

**Description**

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

**Usage**

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

**Arguments**

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

---

assumedConvex<-	<i>Change the <a href="#">assumedConvex</a> boolean of a <a href="#">CMBWindow</a></i>
-----------------	--

---

**Description**

Change the [assumedConvex](#) boolean of a [CMBWindow](#)

**Usage**

```
assumedConvex(win, ...) <- value
```

---

cbind.CMBDataFrame      [cbind](#) for CMBDataFrames

---

## 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)
```

## Details

See the documentation for [cbind](#)

## Examples

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

---

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 CMBDat object, 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.

<code>spix</code>	Optional vector of sample pixel indices, or a path to a file containing comma delimited sample pixel indices. The ordering scheme is given by <code>ordering</code> . If <code>ordering</code> is unspecified then <code>CMBData</code> must be either a <code>CMBDataFrame</code> or a FITS file and the ordering scheme is then assumed to match that of <code>CMBData</code> .
<code>sample.size</code>	If a positive integer is given, a simple random sample of size equal to <code>sample.size</code> will be taken from <code>CMBData</code> . If <code>spix</code> is specified then <code>sample.size</code> must be unspecified.
<code>nside</code>	Optionally specify the <code>nside</code> parameter manually (usually 1024 or 2048).
<code>ordering</code>	Specifies the desired HEALPix ordering scheme ("ring" or "nested") for the output <code>CMBDataFrame</code> . If <code>ordering</code> is unspecified then the ordering scheme will be taken from the <code>CMBData</code> object, which must then be either a <code>CMBDataFrame</code> or a path to a FITS file. This parameter also specifies the ordering scheme of <code>spix</code> .
<code>I</code>	A vector of intensities to be included if <code>CMBData</code> is unspecified. Note that <code>length(I)</code> must equal $12 * nside^2$ if either <code>spix</code> or <code>sample.size</code> are unspecified.
<code>...</code>	Optional names data columns of length <code>nrow(CMBData)</code> to add to the <code>CMBDataFrame</code> .

### Value

A `CMBDataFrame` whose `row.names` attribute contains HEALPix indices.

### Examples

```
## Method 1: Read the data while constructing the CMBDataFrame
df <- CMBDataFrame("CMB_map_smica1024.fits")

# Specify a sample size for a random sample
df.sample <- CMBDataFrame(df, sample.size = 800000)
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)
```

---

CMBReadFITS

*Read CMB data from a FITS file.*


---

### Description

CMBReadFITS is adapted from the [readFITS](#) function in package [FITSio](#). CMBReadFITS is in development stage and will only work with 'CMB\_map\_smica1024.fits'. When it works, CMBReadFITS is much faster than [readFITS](#). However, [readFITS](#) is more general and so is more likely to work.

**Usage**

```
CMBReadFITS(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**

```
dat <- CMBReadFITS("CMB_map_smica1024.fits")

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

---

CMBWindow

---

CMBWindow

---

**Description**

Create a CMBWindow: Either a polygon or a disc type

**Usage**

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

**Arguments**

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

## Details

If  $r$  is unspecified then the rows of  $\dots$  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 bygons). On the other hand, if  $r$  is specified then  $\dots$  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(0,pi/2,pi/2), phi = c(0,0,pi/2))
```

---

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

---

## Description

This function returns the coordinate system used in a CMBDataFrame. The coordinate system is either "cartesian" or "spherical"

## Usage

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

## Arguments

cmbdf	a CMBDataFrame.
new.coords	specifies the new coordinate system ("spherical" or "cartesian") if a change of coordinate system is desired.

## Details

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

## 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

```
df <- CMBDataFrame("CMB_map_smica1024.fits", sample.size = 800000)
coords(df)
coords(df, new.coords = "cartesian")
```



---

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

---

### 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(win, new.coords)
```

### Arguments

new.coords	specifies the new coordinate system ("spherical" or "cartesian") if a change of coordinate system is desired
cmbdf	a CMBWindow.

### 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

```
df <- CMBDataFrame("CMB_map_smica1024.fits", sample.size = 800000)
coords(df)
coords(df, new.coords = "cartesian")
```

---

coords.data.frame	<i>Change the coordinate system of a data.frame</i>
-------------------	---

---

### Description

Change the coordinate system of a data.frame

### Usage

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

**Arguments**

df	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").

**Value**

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

---

coords.HPDataFrame	<i>Coordinate system from a <a href="#">HPDataFrame</a></i>
--------------------	---

---

**Description**

Add or change coordinates in a [HPDataFrame](#)

**Usage**

```
## S3 method for class 'HPDataFrame'
coords(hpdf, new.coords, healpix.only = FALSE)
```

**Arguments**

hpdf	a HPDataFrame.
new.coords	specifies the new coordinate system ("spherical" or "cartesian")
healpix.only	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

**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")

## 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
```

```
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", healpix.only = TRUE)
```

---

*coords<-.CMBDataFrame*    *Assign new [coords](#) system to CMBDataFrame*

---

## Description

Assign new [coords](#) system to [CMBDataFrame](#)

## Usage

```
## S3 replacement method for class 'CMBDataFrame'
coords(cmbdf, ...) <- value
```

---

*coords<-.CMBWindow*    *Assign new coordinate system to CMBWindow*

---

## Description

Assign new coordinate system to [CMBWindow](#)

## Usage

```
## S3 replacement method for class 'CMBWindow'
coords(win, ...) <- value
```

---

*coords<-.HPDataFrame*    *Assign new [coords](#) system to HPDataFrame*

---

## Description

Assign new [coords](#) system to [HPDataFrame](#)

## Usage

```
## S3 replacement method for class 'HPDataFrame'
coords(hpdf, ...) <- value
```

---

covCMB	<i>Covariance for CMB</i>
--------	---------------------------

---

### Description

This function provides an empirical covariance estimate for data in a CMBDataFrame or data.frame. It places data into bins.

### Usage

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

### Arguments

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

### Value

An object of class CMBCovariance consisting of a [data.frame](#) containing sample covariance values, bin centers, and number n of data point pairs whose distance falls in the corresponding bin. The first row of this data.frame corresponds to the sample variance. The attribute "breaks" contains the break points used. The returned [data.frame](#) has num.bins + 1 rows since the first row, the sample variance, is not counted as a bin.

---

geoArea.CMBDataFrame	<i>Geodesic area covered by a <a href="#">CMBDataFrame</a></i>
----------------------	--

---

### Description

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

### Usage

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

**Arguments**

cmbdf                      a CMBDataFrame

**Value**

the sum of the areas of all pixels (rows) in cmbdf

---

geoArea.CMBWindow              *Get the geodesic area of a [CMBWindow](#)*

---

**Description**

Get the geodesic area of a [CMBWindow](#)

**Usage**

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

**Arguments**

win                          a CMBWindow

**Value**

The spherical area inside win

---

geoDist                          *Geodesic distance on the unit sphere*

---

**Description**

Geodesic distance on the unit sphere

**Usage**

```
geoDist(p1, p2)
```

**Arguments**

p1                          a 3 element vector on the unit sphere given in Cartesian coordinates (x,y,z), or a 2 element vector (theta, phi) giving spherical coordinates, can also be a [data.frame](#) or matrix with rows specifying vectors

p2                          a vector on the unit sphere given in Cartesian coordinates (x,y,z) or a named vector (theta, phi) giving spherical coordinates, can also be a [data.frame](#) or matrix with rows specifying vectors

**Value**

The geodesic distance between p1 and p2

---

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

---

HPDataFrame	<i>HPDataFrame class</i>
-------------	--------------------------

---

**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 [HPDataFrames](#), 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)
```

**Arguments**

<code>...</code>	data, can be named vectors or a <code>data.frame</code>
<code>nside</code>	integer, the nside parameter, i.e, resolution
<code>ordering</code>	the HEALPix ordering scheme ("ring" or "nested")
<code>auto.spix</code>	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 <code>x,y,z</code> for cartesian or <code>theta, phi</code> for spherical colatitude and longitude respectively
<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>

---

is.CMBDat	<i>Check if an object is of class CMBDat</i>
-----------	--

---

**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

---

is.CMBDataFrame	<i>Check if an object is of class CMBDataFrame</i>
-----------------	--

---

**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

---

<code>is.CMBWindow</code>	<i>Check if an object is a CMBWindow</i>
---------------------------	--

---

**Description**

Check if an object is a CMBWindow

**Usage**

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

**Arguments**

<code>win</code>	any object
------------------	------------

**Value**

TRUE or FALSE depending if win is a CMBWindow

---

<code>JacobiRecursive</code>	<i>Calculate Jacobi polynomial values of degree <math>L</math> at given point <math>T</math> in <math>[-1,1]</math>.</i>
------------------------------	--

---

**Description**

Calculate Jacobi polynomial values of degree  $L$  at given point  $T$  in  $[-1,1]$ .

**Usage**

```
JacobiRecursive(a, b, L, T)
```

**Arguments**

<code>L</code>	The degree of Jacobi polynomial
<code>T</code>	Given point in $[-1,1]$ .
<code>(a, b)</code>	The parameters of Jacobi polynomial

**Value**

Jacobi polynomial values

**Source**

<http://dlmf.nist.gov/18.9>

**Examples**

```
JacobiRecursive(0,0,5,0)
JacobiRecursive(1,2,4,0.5)
```



---

maxDist.CMBDataFrame	<i>Get the maximum distance between all points in a <a href="#">CMBDataFrame</a></i>
----------------------	--

---

**Description**

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

**Usage**

```
## S3 method for class 'CMBDataFrame'
maxDist(cmbdf)
```

**Arguments**

cmbdf	a CMBDataFrame object
-------	-----------------------

---

maxDist.CMBWindow	<i>Get the maximum distance between all points in a <a href="#">CMBWindow</a></i>
-------------------	---

---

**Description**

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

**Usage**

```
## S3 method for class 'CMBWindow'
maxDist(win)
```

**Arguments**

win	a CMBWindow object
-----	--------------------

---

minDist	<i>minDist</i>
---------	----------------

---

**Usage**

```
minDist(cmbdf, point)
```

**Arguments**

cmbdf	a data.frame or <a href="#">CMBDataFrame</a>
point	a point on the unit sphere in cartesian coordinates

**Value**

the shortest distance from point to cmbdf

---

nest2ring	<i>nest2ring</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.

---

nestSearch	<i>Nested Search</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, j = 0:log2(nside),
  demo.plot = FALSE)
```

### Arguments

target	is a vector of Cartesian coordinates for the target point on $S^2$
nside	is the nside for which the HEALPix points are searched
demo.plot	If TRUE then a plot will be produced with target pixel in yellow and closest pixel at each step in red

### 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.

**Examples**

```
# Find the pix index and Cartesian coordinates of the HEALPix point
# at nside closest to the target point c(0,0,1)
h <- nestSearch(c(0,0,1), nside=1024)
cat("Closest HEALPix point to (0,0,1) at nside = 1024 is (",h$xyz,")")
```

nestSearch\_step

*nestSearch\_step***Description**

Search for the closest HEALPix pixel to a target point, where the search is restricted to within HEALPix pixel, `pix.j1`, at resolution `j1`. The returned value is a HEALPix pixel (and, optionally, the cartesian coordinates of its center) at resolution `j2`, where `j2 > j1`. All pixels are assumed to be in nested ordering scheme.

**Usage**

```
nestSearch_step(target, j1 = j2, j2, pix.j1 = 0, demo.plot = FALSE)
```

**Arguments**

<code>target</code>	is the target point on $S^2$ in spherical coordinates.
<code>j1</code>	is the lower resolution, with <code>j1 &lt; j2</code> .
<code>j2</code>	is the upper resolution.
<code>pix.j1</code>	is the initial pix index at resolution <code>j1</code> , i.e., the <code>j1</code> -level pixel to search in. If <code>pix.j1 = 0</code> then all pixels will be searched (slow).
<code>demo.plot</code>	If TRUE then a plot will be produced with target pixel in yellow and closest pixel in red

**Details**

`j1` and `j2` are HEALPix resolution parameters, i.e.,  $n_{side} = 2^j$ .

`nestSearch_step(target, j2, j1, pix.j1)` searches within the subregion `pix.j1`, where `pix.j1` is a HEALPix pixel index at resolution `j1`. The return value is the HEALPix point closest to `target`, at resolution `j2`.

Setting `pix.j1 = 0` (the default) searches for the HEALPix point closest to `target` at resolution `j2`, among all HEALPix points at resolution `j1`.

**Value**

A list containing the Cartesian coordinates, `xyz`, and the HEALPix pixel index, `pix`, of the closest HEALPix pixel center to the target point, `target`, at resolution `j2`

**Examples**

```
# search for the HEALPix pixel center closest to North pole
# (0,0,1) at level 3
nestSearch_step(target = c(0,0,1), j2 = 3, j1 = -1, demo.plot = TRUE )
```

---

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(cmbdf)
```

**Arguments**

cmbdf                    a CMB Data Frame.

**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(hpdf)
```

**Arguments**

hpdf                    a [HPDataFrame](#).

**Value**

The HEALPix Nside parameter

**Examples**

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

---

ordering.CMBDataFrame *HEALPix ordering scheme from a CMBDataFrame*

---

### 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(cmbdf, new.ordering)
```

### Arguments

cmbdf	a CMB Data Frame.
new.ordering	specifies the new ordering ("ring" or "nest") if a change of ordering scheme is desired.

### Details

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

### Value

The name of the HEALPix ordering scheme that is used in the CMBDataFrame cmbdf

### Examples

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

---

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(hpdf, new.ordering)
```

**Arguments**

hpdf                    a [HPDataFrame](#).

new.ordering        specifies the new ordering ("ring" or "nest") if a change of ordering scheme is desired.

**Value**

The name of the HEALPix ordering scheme that is used in the HPDataFrame hpdf, or a new hpdf with the desired new.ordering

**Examples**

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

## Plot using coordinates
hp1 <- HPDataFrame(x = c(1,0,0),
                   y = c(0,1,0),
                   z = c(0,0,1),
                   nside = 1,
                   auto.spix = TRUE)
plot(hp, size = 5, hp.boundaries = 1)
```

---

```
ordering<-.CMBDataFrame
```

*Assign new ordering scheme to CMBDataFrame*

---

**Description**

Assign new ordering scheme to CMBDataFrame

**Usage**

```
## S3 replacement method for class 'CMBDataFrame'
ordering(cmbdf, ...) <- value
```

---

```
ordering<-.HPDataFrame
```

*Assign new ordering scheme to HPDataFrame*

---

**Description**

Assign new ordering scheme to HPDataFrame

**Usage**

```
## S3 replacement method for class 'HPDataFrame'
ordering(hpdf, ...) <- value
```

---

pix.CMBDataFrame	<i>HEALPix pixel indices from</i> <a href="#">CMBDataFrame</a>
------------------	--

---

**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 pixel indices new.pix

**Usage**

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

**Arguments**

cmbdf	a CMBDataFrame.
new.pix	optional vector of pixel indices

**Value**

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

**Examples**

```
df <- CMBDataFrame("CMB_map_smica1024.fits", sample.size = 800000)
pix(df)
```

---

pix.HPDataFrame	<i>HEALPix pixel indices from</i> <a href="#">HPDataFrame</a>
-----------------	---

---

**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 pixel indices new.pix

**Usage**

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

**Arguments**

hpdf	a <a href="#">HPDataFrame</a> .
new.pix	optional vector of pixel indices

**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)
```

---

<code>pix2coords</code>	<i>pix2coords</i>
-------------------------	-------------------

---

**Description**

convert HEALPix pixel indices to cartesian or spherical coordinates

**Usage**

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

**Arguments**

<code>nside</code>	the <code>nside</code> parameter
<code>coords</code>	'cartesian' or 'spherical' coordinates
<code>ordering</code>	'ring' or 'nested' ordering
<code>spix</code>	optional integer or vector of sample pixel indices

**Value**

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

---

<code>pix&lt;-.CMBDataFrame</code>	<i>Assign new pixel indices to a CMBDataFrame</i>
------------------------------------	---

---

**Description**

Assign new pixel indices to a `CMBDataFrame`

**Usage**

```
## S3 replacement method for class 'CMBDataFrame'
pix(cmbdf, ...) <- value
```



---

pixelArea	<i>pixelArea</i>
-----------	------------------

---

**Description**

Get the area of a single HEALPix pixel

**Usage**

```
pixelArea(cmbdf)
```

**Arguments**

cmbdf            a [CMBDataFrame](#)

**Value**

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

---

pixelWindow	<i>Pixel window</i>
-------------	---------------------

---

**Description**

All pixels are assumed to be in nested ordering

**Usage**

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

**Arguments**

j1	is the lower resolution, with $j1 < j2$
j2	the upper resolution
pix.j1	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

---

plot.CMBDataFrame	<i>Plot CMB Data</i>
-------------------	----------------------

---

## Description

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

## Usage

```
## S3 method for class 'CMBDataFrame'
plot(cmbdf, intensities = "I", add = FALSE,
     sample.size, type = "p", size = 1, box = FALSE, axes = FALSE,
     aspect = FALSE, col, back.col = "black", labels, ...)
```

## Arguments

cmbdf	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 <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(cmbdf)</code>
...	arguments passed to <code>rgl::plot3d</code>

## Value

A plot of the CMB data

## Examples

```
filename <- "CMB_map_smica1024.fits"
sky <- CMBDataFrame(filename)
plot(sky, sample.size = 800000)
```

---

plot.CMBWindow	visualise a CMBWindow
----------------	-----------------------

---

## Description

visualise a CMBWindow

## Usage

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

## Arguments

win	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 rgl::bg3d.
...	arguments passed to rgl::plot3d
eps	the geodesic distance between consecutive points to draw on the window boundary

---

plot.HPDataFrame	Plot HPDataFrame
------------------	------------------

---

## Description

This function produces a plot from a HPDataFrame. If columns x,y,z (cartesian) or theta,phi (co-latitude and longitude respectively) are present in hpdf, then these will be used as coordinates for plotting. Otherwise, the HEALPix indices as in pix(hpdf) 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(hpdf, 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", ...)
```

**Arguments**

hpdf	a HPDataFrame.
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(hpdf)
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
...	arguments passed to rgl::plot3d

**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)
```

---

plotHPBoundaries	<i>plotHPBoundaries</i>
------------------	-------------------------

---

**Description**

plot the HEALPix pixel boundaries at nside

**Usage**

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

**Arguments**

nside	the HEALPix nside parameter
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
...	arguments passed to <code>rgl::plot3d</code>

**Value**

produces a plot

---

print.CMBDataFrame	<i>Print CMB Data</i>
--------------------	-----------------------

---

**Description**

This function neatly prints the contents of a CMB Data Frame.

**Usage**

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

**Arguments**

cmbdf	a CMB Data Frame.
...	arguments passed to <a href="#">print.tbl_df</a>

**Value**

Prints contents of the CMB data frame to the console.

**Examples**

```
df <- CMBDataFrame("CMB_map_smica1024.fits", sample.size = 800000)
print(df)
df
```

---

```
print.summary.CMBDataFrame
```

*Print a summary of a CMBDataFrame*

---

**Description**

Print a summary of a CMBDataFrame

**Usage**

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

**Arguments**

x a summary.CMBDataFrame object, i.e., the output of [summary.CMBDataFrame](#)

---

```
print.summary.CMBWindow
```

*Print a summary of a CMBWindow*

---

**Description**

Print a summary of a CMBWindow

**Usage**

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

**Arguments**

x a summary.CMBWindow object, i.e., the output of [summary.CMBWindow](#)

---

rbind.CMBDataFrame	Like <a href="#">rbind</a> for CMBDataFrames
--------------------	--

---

**Description**

Add a new row or rows to a [CMBDataFrame](#). All arguments passed to ... must be CMBDataFrames.

**Usage**

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

**Arguments**

unsafe	defaults to FALSE. If unsafe = TRUE then overlapping pixel coordinates will not throw an error (faster). See the documentation for <a href="#">rbind</a>
--------	---

---

rcosmo	<i>rcosmo - This Documentation is a place holder.</i>
--------	---

---

**Description**

To be completed

**Section1**

To be completed

**Section2**

To be completed

**Section 3**

To be completed

**Dependencies**

To be completed

**Author(s)**

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

---

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

---

ring2nest	<i>Ring to Nest.</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.  
 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**

```
# compute HEALPix indices in the ring order of the set pix given in the nest order at nside
nside <- 8
pix <- c(1,2,23)
ring2nest(nside,pix)
```



---

sampleCMB	<i>Take a simple random sample from a CMBDataFrame</i>
-----------	--

---

**Description**

This function returns a CMBDataFrame with size sample.size, whose rows comprise a simple random sample of the rows from the input CMBDataFrame.

**Usage**

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

**Arguments**

cmbdf	a CMB Data Frame.
sample.size	the desired sample size.

**Value**

A CMBDataFrame with size sample.size, whose rows comprise a simple random sample of the rows from the input CMBDataFrame.

**Examples**

```
df <- CMBDataFrame("CMB_map_smica1024.fits")
plot(sampleCMB(df, sample.size = 800000))
```

---

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

---

**Description**

The function SphericalHarmonics computes the spherical harmonic values on the given 3D Cartesian coordinates.

**Usage**

```
SphericalHarmonics(L, m, xyz)
```

**Arguments**

L	The degree of spherical harmonic
m	The order number of the degree-L spherical harmonic
xyz	Given points in 3D cartesian coordinates

**Value**

The spherical harmonic values

## References

Hesse, K., Sloan, I. H., & Womersley, R. S. (2010). Numerical integration on the sphere. In Handbook of Geomathematics (pp. 1185-1219). Springer Berlin Heidelberg.

## Examples

```
SphericalHarmonics(5,2,c(0,1,0))
SphericalHarmonics(5,2,diag(3))
```

---

subWindow	<i>subWindow</i>
-----------	------------------

---

## Description

Restricts a [CMBDataFrame](#), [CMBDat](#) object, or [data.frame](#) to a [CMBWindow](#) region. A single CMBWindow or a list of CMBWindows can be passed to the win argument.

## Usage

```
subWindow(cmbdf, win, intersect = TRUE, in.pixels, in.pixels.res = 0)
```

## Arguments

cmbdf	a <a href="#">CMBDataFrame</a> , a <a href="#">data.frame</a> , or <a href="#">CMBDat</a> object. If this is a <a href="#">data.frame</a> then it must have columns labelled x,y,z specifying cartesian coordinates, or columns labelled theta, phi specifying colatitude and longitude respectively.
win	a <a href="#">CMBWindow</a> or a list of CMBWindows
intersect	a boolean that determines the behaviour when win is a list (see details).
in.pixels	a vector of pixels at resolution in.pixels.res whose union contains the window(s) win entirely. This will only be used if cmbdf is a <a href="#">CMBDataFrame</a>
in.pixels.res	a resolution (i.e., $j$ such that $n_{\text{side}} = 2^j$ ) at which the in.pixels parameter is specified

## Details

Windows that are tagged with `set.minus` (see [CMBWindow](#)) are treated differently from other windows: Let  $A$  be the union of the interiors of all windows whose `winType` does not include "minus", and let  $B$  be the intersection of the exteriors of all the windows whose `winType` does include "minus". Then, provided that `intersect = TRUE` (the default), the returned [CMBDataFrame](#) will be the intersection of the points in `cmbdf` with  $A$  and  $B$ . Otherwise, if `intersect = FALSE`, the returned [CMBDataFrame](#) will be the intersection of the points in `cmbdf` with the union of  $A$  and  $B$ . Note that if  $A$  (resp.  $B$ ) is empty then the returned [CMBDataFrame](#) will be the intersection of  $B$  (resp.  $A$ ) with `cmbdf`.

## Value

a [CMBDataFrame](#), or just a [data.frame](#), which is restricted to the region of the sky specified by win

---

summary.CMBDataFrame	Summarise a <a href="#">CMBDataFrame</a>
----------------------	--

---

**Description**

This function produces a summary from a CMBDataFrame.

**Usage**

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

**Arguments**

cmbdf	a CMBDataFrame.
intensities	the name of a column specifying CMB intensities (or potentially another numeric quantity of interest)

**Value**

A summary

---

summary.CMBWindow	Summarise a <a href="#">CMBWindow</a>
-------------------	---------------------------------------

---

**Description**

This function produces a summary from a CMBWindow

**Usage**

```
## S3 method for class 'CMBWindow'  
summary(win)
```

**Arguments**

cmbdf	a CMBWindow
-------	-------------

**Value**

A summary

---

triangulate	<i>Triangulate a polygonal <a href="#">CMBWindow</a></i>
-------------	--

---

**Description**

Triangulate a polygonal [CMBWindow](#)

**Usage**

```
triangulate(win)
```

**Arguments**

win	a <a href="#">CMBWindow</a> object
-----	------------------------------------

**Value**

a list of [CMBWindow](#) polygons or minus.polygons, each having 3 vertices and representing a triangle. These triangles have pairwise disjoint interiors and their union is equal to the original polygon, win.

---

window	<i>Window attribute of <a href="#">CMBDataFrame</a></i>
--------	---

---

**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**

```
window(cmbdf, new.window, intersect = TRUE, in.pixels, in.pixels.res = 0)
```

**Arguments**

cmbdf	a <a href="#">CMBDataFrame</a> .
new.window	optionally specify a new window in which case a new <a href="#">CMBDataFrame</a> is returned whose <a href="#">CMBWindow</a> is new.window. new.window may also be a list (see details section).
intersect	a boolean that determines the behaviour when win is a list (see details).
in.pixels	a vector of pixels at resolution in.pixels.res whose union contains the window(s) win entirely, or if new.window is unspecified then this whole pixel is returned
in.pixels.res	a resolution (i.e., $j$ such that $n_{\text{side}} = 2^j$ ) at which the in.pixels parameter is specified

**Details**

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

**Value**

The window attribute of `cmbdf` or, if `new.window/in.pixels` is specified, a new `CMBDataFrame`.

**Examples**

```
cmbdf <- CMBDataFrame(nside = 16, coords = "cartesian", ordering = "nested")

## Create a new CMBDataFrame with a window
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)

## Change the window of an existing CMBDataFrame
window(cmbdf) <- CMBWindow(theta = rep(0.1, 10),
                           phi = seq(0, 18*pi/10, length.out = 10))
plot(cmbdf)
```

---

window<-	Assign a new <a href="#">CMBWindow</a> to a <a href="#">CMBDataFrame</a>
----------	--

---

**Description**

Assign a new [CMBWindow](#) to a [CMBDataFrame](#)

**Usage**

```
window(cmbdf, ...) <- value
```

---

winType	Get the type (polygon or disk) of a <a href="#">CMBWindow</a>
---------	---

---

**Description**

Get the type (polygon or disk) of a [CMBWindow](#)

**Usage**

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

**Arguments**

win	a <a href="#">CMBWindow</a> object or a list of such
new.type	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

---

winType<-	<i>Assign new winType to a CMBWindow</i>
-----------	--

---

**Description**

Assign new winType to a CMBWindow

**Usage**

```
winType(win, ...) <- value
```

# Index

## \*Topic **Jacobi,Orthogonal**

JacobiRecursive, [16](#)

## \*Topic **harmonic**

SphericalHarmonics, [33](#)

## \*Topic **polynomials.**

JacobiRecursive, [16](#)

## \*Topic **spherical**

SphericalHarmonics, [33](#)

areCompatibleCMBDFs, [3](#)

as.CMBDataFrame, [3](#)

assumedConvex, [4](#), [4](#)

assumedConvex<-, [4](#)

cbind, [5](#)

cbind.CMBDataFrame, [5](#)

CMBDataFrame, [3](#), [5](#), [5](#), [11](#), [12](#), [14](#), [17](#), [23](#), [25](#),  
[26](#), [31](#), [32](#), [34–37](#)

CMBReadFITS, [6](#)

CMBWindow, [4](#), [5](#), [7](#), [9](#), [13](#), [17](#), [27](#), [30](#), [34–38](#)

coords, [11](#)

coords.CMBDataFrame, [8](#)

coords.CMBWindow, [9](#)

coords.data.frame, [9](#)

coords.HPDataFrame, [10](#)

coords<- .CMBDataFrame, [11](#)

coords<- .CMBWindow, [11](#)

coords<- .HPDataFrame, [11](#)

covCMB, [12](#)

data.frame, [3](#), [5](#), [12](#), [13](#), [34](#)

geoArea.CMBDataFrame, [12](#)

geoArea.CMBWindow, [13](#)

geoDist, [13](#)

header, [14](#)

HPDataFrame, [10](#), [11](#), [14](#), [20](#), [22](#), [23](#), [27](#)

HPDataFrames, [14](#)

is.CMBDat, [15](#)

is.CMBDataFrame, [15](#)

is.CMBWindow, [16](#)

JacobiRecursive, [16](#)

maxDist.CMBDataFrame, [17](#)

maxDist.CMBWindow, [17](#)

minDist, [17](#)

mmap, [7](#)

nest2ring, [18](#)

nestSearch, [18](#)

nestSearch\_step, [19](#)

nside.CMBDataFrame, [20](#)

nside.HPDataFrame, [20](#)

ordering.CMBDataFrame, [21](#)

ordering.HPDataFrame, [21](#)

ordering<- .CMBDataFrame, [22](#)

ordering<- .HPDataFrame, [22](#)

pix.CMBDataFrame, [23](#)

pix.HPDataFrame, [23](#)

pix2coords, [24](#)

pix<- .CMBDataFrame, [24](#)

pixelArea, [25](#)

pixelWindow, [25](#)

plot.CMBDataFrame, [26](#)

plot.CMBWindow, [27](#)

plot.HPDataFrame, [27](#)

plotHPBoundaries, [29](#)

print.CMBDataFrame, [29](#)

print.summary.CMBDataFrame, [30](#)

print.summary.CMBWindow, [30](#)

print.tbl\_df, [29](#)

rbind, [31](#)

rbind.CMBDataFrame, [31](#)

Rcosmo(rcosmo), [31](#)

rCosmo(rcosmo), [31](#)

rcosmo, [31](#)

rcosmo-package(rcosmo), [31](#)

readFITS, [6](#)

resolution, [32](#)

ring2nest, [32](#)

sampleCMB, [33](#)

SphericalHarmonics, [33](#)

subWindow, [7](#), [34](#), [37](#)

summary.CMBDataFrame, [30](#), [35](#)

`summary.CMBWindow`, [30](#), [35](#)

`suppressMessages`, [3](#)

`triangulate`, [36](#)

`window`, [36](#)

`window<=`, [37](#)

`winType`, [37](#), [38](#)

`winType<=`, [38](#)