Package ‘rcosmo’

September 3, 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 6.1.0

VignetteBuilder R.rsp

R topics documented:

`areCompatibleCMBDFs` *Check compatibleness of CMBDataFrames*

Description

Compare attributes to decide if two CMBDataFrames are compatible

Usage

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

Arguments

`cmbdf1`	a `CMBDataFrame`
`cmbdf2`	a `CMBDataFrame`
`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 dataframes to CMBDataFrames*

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).

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 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)`

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)
```

`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
```

`CMBDat` *CMBDat class.*

Description

The function `CMBReadFITS` 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.

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

```
cmbdat <- CMBReadFITS("CMB_map_smica1024.fits", mmap = TRUE)
class(cmbdat)
str(cmbdat)

# View metadata
dat$header1
dat$header2
dat$resoln
dat$method
dat$coordsys
dat$nside
dat$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 `CMBDat` object, or unspecified.
`coords`	Can be "spherical," "cartesian", or unspecified (HEALPix only).
`win`	optional `CMBWindow` 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, which must then be either a CMB-DataFrame or a path to a FITS file. This parameter also specifies the ordering scheme of `spix`.
`I`	A vector of intensities to be included if `CMBData` is unspecified. Note that `length(I)` must equal $12 * nside^2$ if either spix or sample.size are unspecified.
`...`	Optional names data columns of length nrow(CMBData) to add to the CMB-DataFrame.

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.subset)
```

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)

Details

The function CMBReadFITS 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.

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

```
cmbdat <- CMBReadFITS("CMB_map_smica1024.fits", mmap = TRUE)
class(cmbdat)
str(cmbdat)

# View metadata
dat$header1
dat$header2
dat$resoln
```

```
dat$method
dat$coordsys
dat$nside
dat$hdr
```

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

`...`	these arguments are compulsory and must be labelled either x, y, z (cartesian) or theta, phi (spherical, colatitude and longitude respectively). Alternatively, a single data.frame 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 `subWindow`. 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 exlude bygons). 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.CMBDataFrame *Coordinate system from a CMBDataFrame*

Description

If `new.coords` is unspecified then this function returns the coordinate system used in the CMBDataFrame `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(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.

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(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

```
## 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(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`

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

`covCMB` *Sample covariance for CMB*

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 `CMBDataFrame` or `data.frame`
`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.

`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.
`N`	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

```
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")
```

---

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.

Usage

```
downloadCMBmap(link = 1, destfile)
```

Arguments

link	A character string naming the URL of a resource to be downloaded.
destfile	A character string with the file name for the downloaded file to be saved. 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 Nside = 2048, at 5 arcmin resolution, and the polarization maps are provided at Nside = 1024 at 10 arcmin resolution.

`link = 1`: CMB Maps produced by Commander with Nside=1024;

`link = 2`: CMB Maps produced by NILC with Nside=1024;

`link = 3`: CMB Maps produced by SEVEM with Nside=1024;

`link = 4`: CMB Maps produced by SMICA with Nside=1024;

`link = 5`: CMB Maps produced by Commander with Nside=2048;

`link = 6`: CMB Maps produced by NILC with Nside=2048;

`link = 7`: CMB Maps produced by SEVEM with Nside=2048;

`link = 8`: CMB Maps produced by SMICA with Nside=2048;

Value

CMB Map Fits File

References

Planck Public Data Release 2 Maps 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 Commander with Nside=1024 and save in the default folder
## as "../rcosmo/CMB_map_commander1024.fits"
downloadCMBmap(link=1)
## Download SMICA with Nside=2048 and save in the default folder
## as "../rcosmo/CMB_map_smica2048.fits"
downloadCMBmap(link=8)
## Download SMICA with Nside=1024 and save in the specified folder,
## fpr example, "C:/CMB_map_smica1024.fits"
downloadCMBmap(link=8, destfile="C:/CMB_map_smica1024.fits")
```

`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)
```

Arguments

`link`	The URL to download the file
`destfile`	A character string with the file name for the downloaded file to be saved. Tilde-expansion is performed.

Details

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

`link = 2`: Baseline high-ell Planck TT power spectra (2 <= ell <= 2508).

`link = 3`: Baseline high-ell Planck EE power spectra (2 <= ell <= 1996).

`link = 4`: Baseline high-ell Planck TE power spectra (2 <= ell <= 1996).

`link = 5`: Low-ell Planck EB power spectra (2 <= ell <= 29).

`link = 6`: Low-ell Planck BB power spectra (2 <= ell <= 29).

Value

The Data Frame with CMB Power Spectra and a txt file 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 to the working directory
## and plot it
CMBPS<- downloadCMBPS(link=1)
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,")")))
```

`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, var)
```

Arguments

`cmbdf`	a CMB Data Frame.
`var`	an index of CMBDataFrame column with measured values
$alpha$	a threshold level

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)
```

`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(cmbdf)
```

Arguments

`cmbdf`	a CMBDataFrame

Value

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

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 Helpix
## 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(win)
```

Arguments

win	a CMBWindow

Value

Tthe spherical area inside win

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 hpdf

Usage

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

Arguments

hpdf	a HPDataFrame

Value

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

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 Helpix points, then the occupied area is
## pi = 1/4*(surface area of unit sphere)

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

`geoDist` *Geodesic distance on the unit sphere*

Description

Get geodesic distance between points on the unit sphere

Usage

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

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.
`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 ith row of `p1` to the jth 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` *Get the FITS headers from a* `CMBDataFrame`

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

```
df <- CMBDataFrame("CMB_map_smica1024.fits")
df.sample <- CMBDataFrame(df, sample.size = 10000)
header(df.sample)
```

`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)
```

Arguments

`...`	data, can be named vectors or a data.frame
`nside`	integer number 2^k, the nside parameter, i.e, resolution
`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
`spix`	a vector of HEALPix pixel indices indicating the pixel locations of the data. Note that `spix` is ignored if `auto.spix = TRUE`

Examples

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

`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

```
cmbdat <- CMBReadFITS("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)
```

jacobiPol *Calculate Jacobi polynomial values*

Description

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

Usage

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

Arguments

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

Value

Jacobi polynomial values

Source

http://dlmf.nist.gov/18.9

Examples

```
jacobiPol(0,0,5,0)
jacobiPol(2,-5,2,-1)
jacobiPol(1,2,4,0.5)
```

`maxDist.CMBDataFrame` *Get the maximum distance between all points in a* `CMBDataFrame`

Description

Get the maximum distance between all points in a `CMBDataFrame`

Usage

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

Arguments

`cmbdf`	a CMBDataFrame object

Value

maximum distance between all points

Examples

```
## For CMBDataFrame with all Healpix ponts included it must be pi
cmbdf <- CMBDataFrame(nside = 4)
pix(cmbdf)
maxDist(cmbdf)

## Example for CMBDataFrame with Healpix only ponts 1 and 2 included

cmbdf <- CMBDataFrame(nside = 4, spix =c(1,2))
pix(cmbdf)
maxDist(cmbdf)
```

`maxDist.CMBWindow` *Get the maximum distance between all points in a* `CMBWindow`

Description

Get the maximum distance between all points in a `CMBWindow`

Usage

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

Arguments

`win`	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))
maxDist(win)
```

`minDist` *Get the minimum distance between points*

Description

Get the minimum distance between a point and points in a data frame

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 `HPDataFrame` or `CMBDataFrame` and coordinate columns are missing, then coordinates will be assigned based on HEALPix pixel indices.
`point`	A point on the unit sphere in cartesian coordinates.

Value

the shortest distance from `point` to the points specified by the rows of `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)
```

`nest2ring` *Convert nest to ring ordering*

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 *Nested Search*

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 an integer number 2^k 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

`target`	is the target point on S^2 in spherical coordinates.
`j1`	is the lower resolution, with j1 < j2.
`j2`	is the upper resolution.
`pix.j1`	is the initial pix index at resolution j1, i.e., the j1-level pixel to search in. If `pix.j1 = 0` then all pixels will be searched (slow).
`demo.plot`	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., $nside = 2^j$.

`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` *HEALPix Nside parameter from a CMBDataFrame*

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 *HEALPix Nside parameter from a* HPDataFrame

---

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 CMB-DataFrame 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)
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(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

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

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(cmbdf, new.pix)
```

Arguments

`cmbdf`	a CMBDataFrame.
`new.pix`	optional vector of pixel indices with length equal to `nrow(cmbdf)`

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 *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 `hpdf`, but with pix attribute `new.pix`. Thus, `new.pix` must have length equal to `nrow(hpdf)`.

Usage

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

Arguments

`hpdf`	a HPDataFrame.
`new.pix`	optional vector of pixel indices with length equal to `nrow(hpdf)`

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 *Area of a HEALPix pixel*

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

Examples

```
df <- CMBDataFrame("CMB_map_smica1024.fits")
pixelArea(df)

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

pixelWindow *Find high resolution pixels falling in a lower resolution window*

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 =< 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.CMBDataFrame` *Plot CMB Data*

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, hp.boundaries = 0,
  hpb.col = "gray", ...)
```

Arguments

`cmbdf`	A `CMBDataFrame`.
`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(cmbdf)`.
`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 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

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 *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` *Plot HEALPix pixel boundaries*

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 (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
`...`	arguments passed to `rgl::plot3d`

Value

produces a plot of the HEALPix pixel boundaries

Examples

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

`print.CMBDataFrame` *Print CMB Data*

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 `print.tbl_df`

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.HPDataFrame` *Print a* `HPDataFrame`

Description

This function neatly prints the contents of a HPDataFrame.

Usage

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

Arguments

`hpdf`	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
```

---

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
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)
```

`rcosmo` *rcosmo - This Documentation is a place holder.*

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` *Get the arcmin resolution from a `CMBDataFrame`*

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

```
df <- CMBDataFrame("CMB_map_smica1024.fits")
resolution(df)
```

`ring2nest` *Convert ring to nest ordering.*

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

```
# 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` *Take a simple random sample from a CMBDataFrame*

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` *Compute spherical harmonic values at given points on the sphere.*

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,...)
`m`	The order number of the degree-L spherical harmonic (m=-L,-L+1,...,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

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,...,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(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 includes window's type and area, total area covered by observations, and main statistcs for intensity values

Examples

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

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

summary.CMBWindow *Summarise a* CMBWindow

Description

This function produces a summary from a CMBWindow

Usage

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

Arguments

`cmbdf`	a CMBWindow

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)
```

`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: triangilation 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]])
```

`window.CMBDat` *Get a sub window from a `CMBDat` object*

Description

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

Usage

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

Arguments

`cmbdat`	a `CMBDat` object.
`new.window`	A single `CMBWindow` object or a list of them.
`intersect`	A boolean that determines the behaviour when `win` is a list containing BOTH regular type and "minus" type windows together (see details).

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 interious 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$data` in the the intersection of A and B. Otherwise, if `intersect = FALSE`, the returned data.frame consists of the points of `cmbdat$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 `cmbdat` in B (resp. A).

Value

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

Examples

```
win1 <- CMBWindow(theta = c(0,pi/2,pi/2), phi = c(0,0,pi/2))
cmbdat <- CMBReadFITS("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(cmbdf, new.window, intersect = TRUE,
  in.pixels, in.pixels.res = 0)
```

Arguments

cmbdf	a CMBDataFrame.
new.window	optionally specify a new window in which case a new CMBDataFrame is returned whose CMBWindow is new.window. new.window may also be a list (see details section and examples).
intersect	A boolean that determines the behaviour when win is a list containing BOTH regular type and "minus" type windows together (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	An integer. Resolution (i.e., j such that nside = 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.

If the argument is a list of CMBWindows, then interious 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 cmbdf in the union of A and B.

Note that if A (resp. B) is empty then the returned CMBDataFrame will be the points of cmbdf in B (resp. A).

Value

The window attribute of cmbdf 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 = TRUE)
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 `df` restricted to the CMBWindow `new.window`

Usage

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

Arguments

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

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 data.frame containing the data in `df` 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 `healpix.only = TRUE` is given. Note that if `healpix.only = 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(hpdf, new.window, intersect = TRUE,
  healpix.only = FALSE)
```

Arguments

`hpdf`	A HPDataFrame.
`new.window`	A single `CMBWindow` object or a list of them.
`intersect`	A boolean that determines the behaviour when `win` is a list containing BOTH regular type and "minus" type windows together (see details).
`healpix.only`	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(hpdf)` unless `healpix.only = TRUE` is given. Note that if `healpix.only = TRUE` then columns x,y,z or theta, phi will be discarded and replaced with pixel center locations.

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 `hpdf` restricted to the CMBWindow `new.window`

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)
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<-`.

Usage

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

Arguments

`win`	a `CMBWindow` 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`

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