Package 'rcosmo'

June 24, 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), tibble, rgl, cli (>= 1.0.0)

Suggests knitr, rmarkdown, testthat

LinkingTo Rcpp

RoxygenNote 6.0.1

VignetteBuilder knitr

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

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

Archs x64

R topics documented:

`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

Usage

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

Arguments

df	Any data.frame with a column labelled "I" for intensities
coords	specifies the coordinate system to be "spherical", "cartesian" or unspecified (HEALPix only). If "spherical" then df must have columns named "theta" and "phi" (colatitude and longitude respectively). If "cartesian" then df must have columns named "x", "y", and "z"
ordering	specifies the ordering scheme ("ring" or "nested")
nside	specifies the Nside parameter

Value

A CMBDataFrame

---

assumedConvex *Check if a* CMBWindow *is assumed convex*

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<- *Change the* assumedConvex *boolean of a* CMBWindow

---

Description

Change the assumedConvex boolean of a CMBWindow

Usage

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

car2sph *car2sph*

Usage

```
car2sph(df)
```

Arguments

df	a data.frame with columns labelled x, y and z

Value

a data.frame with columns theta and phi for colatitude and longitude in ranges $[0, pi]$ and $[0, 2pi]$ respectively

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, 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 (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 data frame whose columns contain the pixel center coordinates theta, phi (colatitude in range $[0, pi]$ and longitude in range $[0, 2pi)$ respectively) or (x,y,z), CMB intensities (I), and optionally polarisation (Q,U) and masks (TMASK, PMASK). The row.names attribute of the resulting CMB Data Frame 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 = "CMB_map_smica1024.fits", sample.size)
```

Arguments

filename	The path to the fits file.

Value

A list containing a matrix named data whose columns may include the intensity (I), polarisation (Q, U), PMASKa and TMASK. Other list elements correspond to header information and other metadata.

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 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(0,pi/2,pi/2), phi = c(0,0,pi/2))
```

coords.CMBDataFrame *Coordinate system from a CMBDataFrame*

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 *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

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

---

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
```

covCMB *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.

covCMBold *(old/unworking) Covariance for CMB*

Description

This function computes the covariance for CMB. It does not place data into bins, but instead generates points on a circle of radius r and then finds the closest HEALPix point to each, using nestSearch.

Usage

```
covCMBold(df = CMBDataFrame(CMBData = "CMB_map_smica1024.fits", coords =
  "HEALPix", ordering = "nested"), rmin = 10^(-6), rmax = pi - 10^(-6),
  Nr = 10, Nside = 1024, N_x_vec = 10)
```

Arguments

rmin, rmax	are the minimum and maximum of radii.
Nr	is the number of radii in between rmin and rmax for which covariance of CMB is evaluated.
Nside	is the Nside for which the HEALPix points are used.
N_x_vec	is the number of points x for each radius, the number of points y for each is 2^(ceil(log2((sqrt(N_x_vec))))) which is equivalent to sqrt(N_x_vec).

Value

the output is the data frame of radius r and the covariance Tcov.

Examples

```
# compute the covariance of CMB at Nside = 1024 and radii between 10^(-6) and pi-10^(-6) with 10 radii
covCMB(rmin = 10^(-6), rmax = pi-10^(-6), Nr = 10, Nside = 1024, N_x_vec = 10)
```

---

covCMB_internal1 *covCMB_internal1*

Usage

```
covCMB_internal1(cmbdf, breaks)
```

---

covCMB_internal2 *covCMB_internal2 This function acknowledges that there is no need to transform with acos. The breaks are cos(r_i) where r_i is the radius. The bins will have equal area provided that cos(r_i) - cos(r_i+1) is fixed. Alternatively, the bins will have equal annular width if r_i+1 - r_i is fixed, but cos(r_i) must be passed in, regardless of which metric is used to fix distance beforehand. We must note that: For r in (0,pi), cos is a strictly decreasing function, e.g. cos(0) > cos(max.dist)*

Usage

```
covCMB_internal2(cmbdf, cos_breaks)
```

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

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

Tthe 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 *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

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

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

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

Description

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

Usage

```
JacobiRecursive(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

```
JacobiRecursive(0,0,5,0)
JacobiRecursive(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

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

maxDist_internal *maxDist_internal*

Usage

```
maxDist_internal(cmbdf)
```

Arguments

cmbdf	a data.frame or CMBDataFrame

Value

the maximum distance between any of the points in cmbdf

minDist *minDist*

Usage

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

Arguments

cmbdf	a data.frame or CMBDataFrame
point	a point on the unit sphere in cartesian coordinates

Value

the shortest distance from point to cmbdf

nest2ring *nest2ring*

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.

nest2ringR *Nest to Ring.*

Description

nest2ringR converts HEALPix pixel indices in the 'ring' ordering scheme to HEALPix pixel indices in the 'nest' ordering scheme.

Usage

```
nest2ringR(nSide, Pix)
```

Arguments

nSide	is the HEALPix Nside parameter.
Pix	is a vector or matrix of HEALPix pixel indices, in the 'nest' ordering scheme.

Value

the output is a vector or matrix of HEALPix pixel indices in the 'ring' 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)
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 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

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 *HEALPix Nside parameter from a CMBDataFrame*

Description

This function returns the HEALPix Nside parameter of a CMBDataFrame

Usage

```
nside(cmbdf)
```

Arguments

cmbdf	a CMB Data Frame.

Value

The HEALPix Nside parameter

Examples

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

`ordering` *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

```
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("CMB_map_smica1024.fits", sample.size = 800000)
ordering(df)
ordering(df, new.ordering = "ring")
```

`ordering<-` *Assign new ordering scheme to CMBDataFrame*

Description

Assign new ordering scheme to CMBDataFrame

Usage

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

pix *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 pixel indices new.pix

Usage

```
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)
```

pix2ang *Pix to Angle*

Description

pix2ang computes the Cartesian cooridinates of the HEALPix points at Pix index at nSide in order ord.

Usage

```
pix2ang(nSide, order_ring, Pix)
```

Arguments

nSide	is the Nside of the HEALPix points.
order_ring	TRUE if the HEALPix order is in the ring order; FALSE if in the nested order.
Pix	is the set of Pix index.

Value

the output is a 2 by length(Pix) matrix of spherical coordinates (theta,phi)' of HEALPix points at nSide at Pix index in the order ord.

Examples

```
# if nargs() < 3, Pix = 1:nPix, nPix = 12*nSide^2. That is, taking all
   all HEALPix in the ring order.
Nside <- 8
pix2ang(Nside,TRUE)

# comput the HEALPix points in spherical coordinates at Nside in the nest order at pix indices from Pix
Nside <- 8
Pix <- c(1,2,23)
pix2ang(Nside,FALSE,Pix)
```

`pix2coords` *pix2coords*

Description

convert HEALPix pixel indices to cartesian or spherical coordinates

Usage

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

Arguments

`nside`	the nside parameter
`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)

`pix2coords_internal` *pix2coords_internal*

Description

Converts HEALPix pixel scheme to spherical or Cartesian coordinates.

Usage

```
pix2coords_internal(nside = 0L, nested = TRUE, spix = NULL,
  cartesian = FALSE)
```

Arguments

nside	The number of cuts to a HEALPix base resolution pixel.
nested	Set to TRUE for NESTED ordering scheme and FALSE for RING.
spix	Optional integer or vector of sample pixel indices.
cartesian	Set to FALSE to output spherical coordinates or else TRUE for cartesian.

Details

This is a place holder

Value

A matrix with columns theta and phi (in that order), or x, y, z (if cartesian = TRUE). Theta (in [0,pi]) is the colatitude in radians measured from the North Pole and phi (in [0, 2*pi]) is the longitude in radians measured Eastward. The remaining 3 columns returned are i, j, and p which represent the HEALPix ring index, pixel-in-ring index, and pixel index respectively.

---

pix2vec *pix2vec*

Description

Compute the Cartesian cooridinates of the HEALPix points at indices spix at Nside in the specified ordering scheme.

Usage

```
pix2vec(Nside = 16, order = "ring", spix)
```

Arguments

Nside	is the Nside of the HEALPix points.
order	specifies the ordering scheme used for the input pixels spix. If order = "nested" then the input pixels are converted to ring ordering. If order = "ring" then no conversion is done.
spix	is a vector of one or more pixel indices whose ordering scheme is given by order and whose Nside parameter is given by Nside. If spix = 0 then all pixel indices at Nside will be used.

Value

Output is a data.frame with length(spix) rows, or $12 * Nside^2$ rows if spix = 0, and 3 columns of Cartesian coordinates (x,y,z) of HEALPix points at Nside in ring ordering scheme.

Examples

```
# Taking all HEALPix points at Nside = 8, using ring order.
pix2vec(Nside = 8, order = "ring")

# Compute the HEALPix points in Cartesian coordinates at Nside = 8,
# in nested ordering scheme, at the indices from spix
Pix <- c(1,2,23)
pix2vec(Nside = 8, order = "nested", spix = Pix)
```

`pix<-` *Assign new pixel indices to a CMBDataFrame*

Description

Assign new pixel indices to a CMBDataFrame

Usage

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

`pixelArea` *pixelArea*

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 *Pixel window*

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 *Plot CMB Data*

Description

This function produces a plot from a CMB Data Frame.

Usage

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

Arguments

cmbdf	a CMB Data Frame with either spherical or cartesian coordinates.
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)
...	arguments passed to rgl::plot3d

Value

A plot of the CMB data

Examples

```
df <- CMBDataFrame("CMB_map_smica1024.fits")
plot(df, 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

---

plotHPBoundaries *plotHPBoundaries*

Description

plot the HEALPix pixel boundaries at nside

Usage

```
plotHPBoundaries(nside, eps = pi/90, col = "black", 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 rgl::plot3d

Value

produces a plot

pointInConvexPolygon *pointInConvexPolygon*

Usage

```
pointInConvexPolygon(df, win)
```

Arguments

df	a data.frame with columns x, y, z for cartesian coordinates. The rows represent points on the surface of a unit sphere
win	a data.frame with columns x, y, z for cartesian coordinates. The rows represent clockwise oriented vertices of a convex spherical polygon that lies entirely within one open hemisphere of the unit sphere.

Value

a logical vector indicated which rows of df lie within the spherical convex polygon determined by win

pointInConvexPolygonHP *pointInConvexPolygonHP*

Usage

```
pointInConvexPolygonHP(nside, nested, win, spix = NULL)
```

Arguments

nside	the nside parameter at which to find pixels
nested	Set to TRUE for NESTED ordering scheme and FALSE for RING
win	a data.frame with columns x, y, z for cartesian coordinates The rows represent clockwise oriented vertices of a convex spherical polygon that lies entirely within one open hemisphere of the unit sphere
spix	Optional integer or vector of sample pixel indices. If spix is unspecified then all pixels at nside are used

Value

a logical vector indicated which pixels in spix lie within the spherical convex polygon determined by win

pointInDisc *pointInDisc*

Usage

```
pointInDisc(df, win)
```

Arguments

df	a data.frame with columns x, y, z for cartesian coordinates. The rows represent points on the surface of a unit sphere
win	a data.frame with columns x, y, z for the cartesian coordinates of a point on the unit sphere, representing a disc center, and column r for the radius or that disc.

Value

a logical vector indicated which rows of df lie within the spherical disc determined by win

pointInDiscHP *pointInDiscHP*

Usage

```
pointInDiscHP(nside, nested, win, spix = NULL)
```

Arguments

nside	the nside parameter at which to find pixels
nested	Set to TRUE for NESTED ordering scheme and FALSE for RING
win	a data.frame with columns x, y, z for the cartesian coordinates of a point on the unit sphere, representing a disc center, and column r for the radius or that disc
spix	Optional integer or vector of sample pixel indices. If spix is unspecified then all pixels at nside are used

Value

a logical vector indicated which pixels in spix lie within the spherical disc determined by win

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.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* rbind *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 rbind

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

Description

It slices, it dices, it handles:

RequestLogs

We don't study readers enough, and we finally have the tools to do that. WMUtils contains several functions centred on the RequestLogs. hive_query allows you to query the unsampled logs, while sampled_logs allows you to retrieve the 1:1000 sampled ones. For both data types, log_strptime turns the timestamp format used into POSIXlt timestamps.

MySQL

If you study editors, our MySQL databases are where all the data lives. `mysql_query` allows you to query a single database on `analytics-store.eqiad.wmnet`, while `global_query` allows you to run over multiple databases. Either way, `mw_strptime` turns the timestamp format used in our DB into POSIXlt timestamps. And once you're done processing, use `mysql_write` to stream the results up to the databases again. Need to update previously written rows? No problem! `mysql_delete` is the function for you.

Geodata

Thanks to a nice python library, pygeoip, we have an easy API to access geographic data associated with IP addresses. `geo_country` retrieves country codes, `geo_city` retrieves cities, and `geo_tz` tzdata-compatible timezones.

User agents

Our user-agent parsing, which use's tobie's ua-parser, is in Python. It's also now in R thanks to `ua_parse`. If you run into incorrectly identified user agents, poke Oliver, since he's a maintainer on the ua-parser repository.

Namespace matching

`namespace_match` allows you convert namespace numbers to localised names, or vice versa, handling the presence of namespaces in reader or editor data. The dataset is also made available as `namespace_names`, or rebuildable via `namespace_match_generator`.

Python integration

Both geodata retrieval and user agent parsing are dependent on Python libraries, so this also contains a R-to-Python-to-R connector, `rpy`. This allows you to pipe R objects into Python, run an arbitrary Python script over them, and pipe the results back into R, using TSVs, .txts or JSON blobs as the intermediary.

Dependencies

Everything has dependencies; WMUtils is weird in that its primary dependencies are Python modules. Specifically, it needs tobie's ua-parser and pygeoip.

Author(s)

Daniel Fryer <daniel-fryer@live.com.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

---

ring2nest *Ring to Nest.*

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 *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))
```

---

sph2car *sph2car*

Usage

```
sph2car(df)
```

Arguments

df	a data.frame with columns labelled theta and phi for colatitude and longitude respectively

Value

a data.frame with columns x, y, z (cartesian coordinates)

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

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` *Restrict a* `CMBDataFrame` *to a* `CMBWindow`

Description

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 CMBDataFrame
`win`	a CMBWindow 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
`in.pixels.res`	a 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: 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 which is restricted to the region of the sky specified by `win`

summary.CMBDataFrame *Summarise a* CMBDataFrame

Description

This function produces a summary from a CMBDataFrame.

Usage

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

Arguments

cmbdf	a CMBDataFrame.

Value

A summary

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

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. These triangles have pairwise disjoint interiors and their union is equal to the original polygon, win.

window *Window attribute of* 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

```
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).
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 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: Let A be the union of the interiors of all windows in the list, `new.window`, 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

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, 9*2*pi/10, length.out = 10))
plot(cmbdf)
```

---

window<- *Assign a new CMBWindow to a CMBDataFrame*

Description

Assign a new CMBWindow to a CMBDataFrame

Usage

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

---

winType *Get the type (polygon or disk) of a CMBWindow*

Description

Get the type (polygon or disk) of a CMBWindow

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

---

winType<- *Assign new* winType *to a* CMBWindow

Description

Assign new winType to a CMBWindow

Usage

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

Index