VIGNETTE

`rcosmo` : Cosmic Microwave Background in R.

Author: Daniel Vidali Fryer

July 9, 2018

Contents

All computations and graphics presented here were produced using the package `rcosmo`*, in the R programming language and software environment, version 3.4.3. Editing and package development was conducted with Rstudio version 1.1.414. Particular use was made of Hadley Wickham's books on advanced R programming [19] and package development [20].* `rcosmo` *owes its functionality to various dependencies, including C++ integration through* `Rcpp` *[6], efficient and abstract memory mapping through* `mmap` *[17], OpenGL 3D vector graphics library integration through* `rgl` *[1], development workflow enhancement through* `devtools` *[21], and some assistance handling FITS data from* `FITSio` *[8]. This package was created in collaboration with Dr Andriy Olenko, Dr Ming Lee and Dr Yu Guang Wang.*

1 Data description

In this section we give a brief description of the kind of data that `rcosmo` is designed to deal with. The purpose of `rcosmo` is to conduct efficient information processing, visualisation, manipulation and statistical analysis tasks, with Cosmic Microwave Background (CMB) data. Specifically, the data should be located on $\mathbb{S}_2$, the unit sphere, and structured according to the Hierarchical Equal Area isoLatitude Pixelation (HEALPix). As is the case with most modern astronomical data, this is usually stored in the Flexible Image Transport System (FITS) format. Section 4 gives a detailed introduction to HEALPix. For even further details on HEALPix, see [7]; and, for a clear derivation of HEALPix, see [16].

The term "CMB data" refers to a broad range of location tagged quantities describing properties of the CMB. For example, the Infrared Science Archive (IRSA) by Caltech's Infrared Processing and Analysis Center (IPAC) hosts curated CMB products from the North American Space Agency (NASA) at the following link:

http://irsa.ipac.caltech.edu/data/Planck/release_2/

To arrive at CMB maps, the products of the Planck mission data (in the range of frequencies from 30 to 857 GHz) are separated from foreground noise using one of the four detailed methods named COMMANDER, NILC, SEVEM and SMICA[1]. These CMB maps are provided at either low resolution ($N_{\text{side}} = 1024$, i.e., 10 arcmin resolution), or high resolution ($N_{\text{side}} = 2048$, i.e., 5 arcmin resolution). The maps include temperature intensity and polarisation data, as well as common masks for identifying regions where the reconstructed CMB is untrusted. Also provided are "foreground only" maps, having the CMB subtracted, containing only informative foreground data and noise[2].

Our focus will mostly be on CMB temperature intensity data. In Planck CMB products, this data is stored as 4-byte floating point binary numbers in big-endian byte-order [10]. The units are given in K_{cmb}, defined as the unit in which a black body spectrum at 2.725 Kelvin (K) is flat with respect to the frequency [9]. We consider this data to be essentially mean-centered about the average temperature of 2.725 K. The fluctuations about this mean are limited to at most 10^{-3} K [12].

[1] https://wiki.cosmos.esa.int/planckpla2015/index.php/Astrophysical_component_separation#CMB_and_foreground_separation

[2] https://wiki.cosmos.esa.int/planckpla2015/index.php/CMB_and_astrophysical_component_maps

2 Highlights and advantages

Prior to the development of the `rcosmo` software package, there existed no efficient packaged means of importing CMB maps into the R programming environment, nor of conducting the necessary conversions between HEALPix schemes and elementary coordinate systems in R. In the Python and MATLAB programming languages, this niche has been served efficiently by the HEALPy [15] and MEALPix [14] toolkits, respectively. One existing R package, `FITSio`, an interface to the C programming language `CFITSIO` library, was capable of importing a vector of approximately 12 million intensity samples from a FITS file in roughly 40 minutes of run time on a modern laptop[3]. By specialising the `FITSio` algorithm to work with CMB data, and by introducing some additional improvements, the author was able to reduce the necessary run time on the same laptop from 40 minutes to under 4 minutes with the `rcosmo` function `CMBReadFITS` (see Section 5), representing an order of magnitude decrease in computation time.

The approach taken by `CMBReadFITS` is to read all data from the FITS file into R memory as a `data.frame`. Unfortunately, this approach is typically no longer tenable when faced with importing more than a few hundred megabytes of data. This is because it is not usually possible to obtain sufficiently large blocks of continguous memory from the operating system [4]. A far superior approach is to use *memory mapping* [17], whereby a C-style pointer is maintained at a particular byte-offset to the target binary file (e.g., the FITS file), so that data can be read into memory on command from this offset. Jeffrey Ryan's R package, `mmap`, provides a highly optimised interface to the underlying operating system calls that afford this functionality [17]. Furthermore, `mmap` supplies the abstractions and conversions necessary for simplified data access and manipulation within R. By integrating `mmap` into the `rcosmo` function `CMBReadFITS`, we were able to eliminate the need to read a full sky map into R, and thereupon bypass this major obstacle to analysing CMB data in R at any resolution. For example, using memory mapping, `rcosmo` can read a simple random sample of size 1 million rows, having coordinate, intensity, polarisation and masking data, from a FITS file into R memory in under 2 seconds. This represents a decrease in computation time of over 3 orders of magnitude compared with the `FITSio`-based approach of reading all 12 million pixels into memory before sampling.

[3]Laptop specifications: Microsoft Surface Laptop with 7th Gen Intel Core m3 (i5) processor; 8GB LPDDR3 SDRAM (1866MHz)

Aside from importing HEALPix data and providing the necessary coordinate scheme conversions, `rcosmo` includes a growing number of geometrical tools, statistical methods, visualisation apparatuses, and processes for subsetting or combining data. These accoutrements are provided under a layer of high-level abstraction (rooted in R's S3 class system) that aims to lower the skills barrier between applied statistics and CMB astronomy.

3 Installation

There are various standardisations and minor adjustments that need to be considered before `rcosmo` is hosted on the popular Comprehensive R Archive Network (CRAN). Meanwhile, availability is provided through GitHub, at

https://github.com/vidalilama/rcosmo

For a simple installation, the package `devtools` should be installed:

```
install.packages("devtools")
```

For efficiency during heavy computations, `rcosmo` directly utilises the C++ programming language through the package `Rcpp` . Hence, if using Microsoft Windows, an up-to-date installation of RTools is necessary to build `rcosmo` . At the time of writing, this could be found on CRAN at the following link:

https://cran.r-project.org/bin/windows/Rtools/

Following the above, we complete the installation of `rcosmo` :

```
devtools::github_install("VidaliLama/rcosmo")
```

Once the process is complete, load the package as usual:

```
library(rcosmo)
```

The package is now ready for use.

4 Introduction to HEALPix

Present generation Cosmic Microwave Background experiments produce data with up to 5 arcminutes resolution on the sphere. For a full-sky map, this amounts to approximately 50 million pixels, each describing distinct location, intensity, polarisation and other attributes. The statistical analysis of such massive datasets, and associated discretisation of functions on the sphere, can involve prohibitive computational complexity in the absence of an appropriate data structure. The Hierarchical Equal Area isoLatitude Pixelation (HEALPix) is a mathematical structure designed to meet this demand using a self-similar refinable mesh; and is currently the most widely used pixelation for storing and analysing CMB data [7].

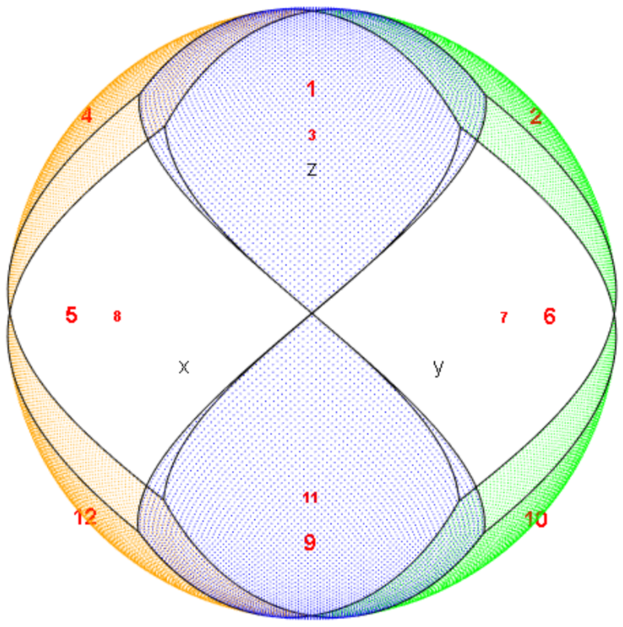


Figure 1: HEALPix base pixel boundary visualisation. The 12 base pixels are labelled 1 to 12.

HEALPix divides the sphere into 12 equiareal *base* pixels. To visualise these with `rcosmo` , we can execute the following code, whose output is given in Figure 1). Many aspects of the code used to generate the images in this section will not be introduced properly until later in this Chapter, but we include the code here for its relevance. Note that all images are screenshots of the interactive 3D vector graphics that `rcosmo` produces through `OpenGL` . Reproducing the original script will allow the reader to rotate and scale their perspective. It is also straightforward, using `knitr` [22] and `RMarkdown` [23], to embed these interactive graphics into a HTML document, though that is outside the scope of this thesis.

```
# Generate a CMB Data Frame at some low resolution (nside = 64)
cmbdf <- CMBDataFrame(nside = 64, ordering = "nested")

# Take CMB Window subsets in the pixels that we intend to colour
w1 <- window(cmbdf, in.pixels = c(1,9))
w2 <- window(cmbdf, in.pixels = c(2,10))
w3 <- window(cmbdf, in.pixels = c(4,12))

# Colour CMB Window subsets with plot.CMBDataFrame generic
plot(w1, col = "blue", back.col = "white")
plot(w2, col = "green", add = TRUE)
plot(w3, col = "orange", add = TRUE)

# Plot the base pixel boundaries (nside = 1) and add labels
plotHPBoundaries(nside = 1, ordering = "nested",
                 incl.labels = 1:12, col = "red")
```

Note that, while all HEALPix pixels are 4-sided, their edges are not geodesics, i.e., they are not spherical quadrillaterals [5]. Each of the 12 base pixels can be further subdivided into 4 equiareal 4-sided pixels. We can demonstrate this with the following visualisation (see output in Figure 2).

```
# Generate higher resolution CMB Data Frame (nside = 256)
w21 <- window(CMBDataFrame(nside = 256, ordering = "nested",
                           coords = "cartesian"), in.pixels = 1)

# Slightly decrease pixel distances to the origin.
# This will make boundaries more visible in the final plot.
w21[,c("x","y","z")] <- w21[,c("x","y","z")]*0.99

# Colour base pixel 1.
# Colour is stronger due to higher resolution in w21.
plot(w21, col = "light blue", back.col = "white",
     add = TRUE, size = 1.2)

# Plot pixel boundaries at nside = 2
plotHPBoundaries(nside = 2, col = "black", lwd = 1,
     ordering = "nested", incl.labels = c(1,2,3,4))
```

```
# Colour some of the base pixels
plot(window(cmbdf,in.pixels = 2), col = "green", add = TRUE)
plot(window(cmbdf,in.pixels = 4), col = "purple", add = TRUE)
plot(window(cmbdf,in.pixels = 5), col = "orange", add = TRUE)
plot(window(cmbdf,in.pixels = 6), col = "red", add = TRUE)
```

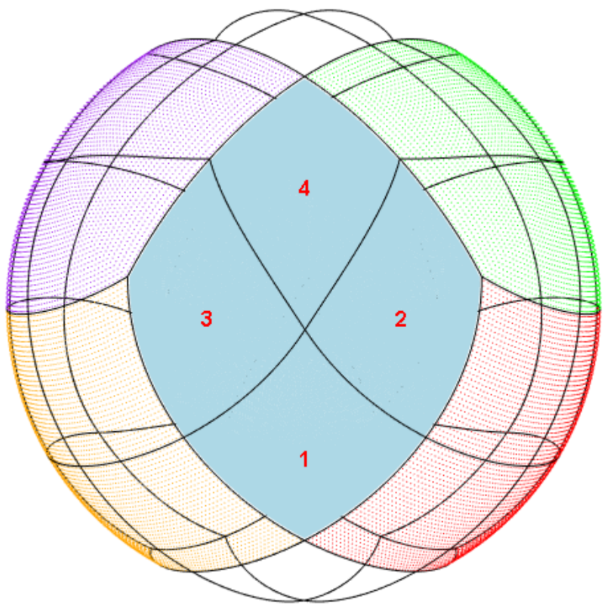


Figure 2: HEALPix nested ordering visualisation at $N_{\text{side}} = 2$,. Pixels 1, 2, 3 and 4 (labelled) all fall within base pixel 1 (coloured solid).

This process of subdivision is repeated until a desired resolution is achieved. An immediate advantage of equiareal pixel sizes is that simple random sampling is not regionally dependent [7]. That is, a simple random sample of pixel indices translates to an approximately uniform sample of locations on the sphere. Thus, in `rcosmo`, uniform sampling is uncomplicated:

```
sample.size <- 1000
sample.index <- sample(1:nrow(cmbdf), size = sample.size)
sample.cmbdf <- cmbdf2[sample.index, ]
```

The above is, basically, that which is executed by the built-in `rcosmo` function:

```
cmb.sample <- sampleCMB(cmbdf2, sample.size = 1000)
```

At the required resolution, the number of edge segments per base pixel edge is referred to as the N_{side} parameter, and clearly satisfies $N_{\text{pix}} = 12N_{\text{side}}^2$, where we use N_{pix} to denote the total number of pixels.

HEALPix offers a choice of either *nested* or *ring* numbering schemes. The nested ordering scheme was demonstrated in Figure 2. Nested ordering produces a hierarchical data structure, facilitating highly efficient nearest neighbour searches (discussed in Subsection **??** and Figure **??**). Another key advantage to the HEALPix structure is that all pixel center locations lie on $4N_{\text{side}} - 1$ rings of constant latitude. This feature facilitates fast discrete spherical harmonic transforms, since the associated Legendre functions need only be evaluated once per isolatitude pixel ring (see Equation (**??**)). In the ring ordering scheme, pixel numbers are assigned in a spiral, from north to south, with consecutive pixels sharing the same isolatitude rings. We can visualise this by producing a line plot of a CMB Data Frame in ring order (see result in Figure 3):

```
cmbdf <- CMBDataFrame(nside = 8, ordering = "ring")
plot(cmbdf, type = 'l', col = "black", back.col = "white")
tolabel <- c(1,100:107,768)
plot(cmbdf[tolabel,], labels = tolabel, col = "red", add = TRUE)
```

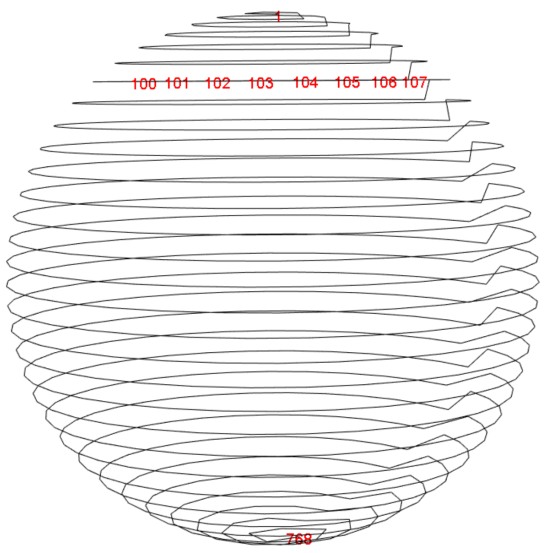


Figure 3: HEALPix ring ordering scheme visualisation. The black line traces in order through pixel centers from 1 to $N_{\text{pix}} = 768$. The locations of pixels 100 to 107 are also labelled.

5 Importing maps

In this section, we demonstrate importing CMB data in the typical case that the data is present as a full sky map in a FITS file. Such maps can be sourced from the NASA/IPAC Infrared Science Archive, hosted by Caltech at the following link:

http://irsa.ipac.caltech.edu/data/Planck/release_2/

To reach the maps of interest from the above webpage, navigate first to "all-sky-maps" and then to "Maps of the Cosmic Microwave Background". The map used in this example is a FITS file of size roughly 200 megabytes, named

`COM_CMB_IQU-smica_1024_R2.02_full.fits` .

This map has resolution $N_{\text{side}} = 1024$, so it contains $N_{\text{pix}} = 12 \times 1024^2 = 12582912$ pixels, each having its own intensity I, polarisation Q, U, temperature mask value $T_{\text{mask}} \in \{0, 1\}$ and polarisation mask value $P_{\text{mask}} \in \{0, 1\}$. In this example, we employ memory mapping, being faster and more robust than loading the whole dataset into R memory. If the target file is located in the current working directory, then we can proceed as follows to create an object of class `mmap` , named `map` :

```
filename <- "COM_CMB_IQU-smica-field-Int_1024_R2.02_full.fits"
map <- CMBReadFITS(filename, mmap = TRUE)
```

Data can immediately be accessed from `map` , using the generic square bracket operator, as if `map` were a `data.frame` . In fact, the returned object after applying the square bracket operator *is* a `data.frame` . For example, the following reads the first 3 rows of CMB data from file, corresponding to HEALPix pixel indices 1, 2 and 3, and returns them as a `data.frame` :

```
map[1:3,]
```

```
              I             Q             U  PMASK  TMASK
1 -9.201023e-05  6.470930e-08 -6.570615e-07      0      0
2 -8.041522e-05 -9.187644e-09 -6.943168e-07      0      0
3 -8.985696e-05  7.361150e-08 -6.845866e-07      0      0
```

The statistics and geometry tools in `rcosmo` are designed to work with objects that exist in R memory. So, in order to analyse the CMB data, we must commit a relevant subset of the full map to R memory. When `map` is passed to the `window` function, HEALPix ordering information is used to import data only from the region of $\mathbb{S}_2$ specified by the `window` function parameter `new.window`. For example, if we wish to import the north polar cap of spherical radius $1/10$, we proceed as follows:

```
win <- CMBWindow(x = 0, y = 0, z = 1, r = 0.1)
cap <- window(map, new.window = win)
```

See Section 6 for further details on the `window` function. The object `map` also contains CMB metadata which it receives from the header of the target FITS file. In the background, `window` uses this information from `map` to build a valid `CMBDataFrame` object. In the above code, we stored this `CMBDataFrame` object in the variable `cap`. To get an understanding of this object, we can produce a summary and an interactive 3D plot (see plot in Figure 4):

```
summary(cap)
```

```
=====================  CMBDataFrame Object  ===================
Number of CMBWindows:  1
+--------------------------+
|                          |
|   Window type: disc      |
|   Window area: 0.0314    |
|                          |
+--------------------------+

METHOD  = 'smica    '           / Separation method
Total area covered by all pixels:  0.03140666
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Intensity quartiles
     Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-5.88e-04 -1.08e-04 -2.71e-05 -3.26e-05  4.61e-05  4.31e-04
===============================================================
```

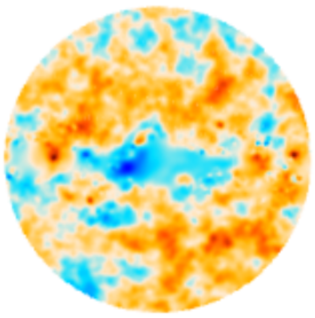


Figure 4: CMB north polar cap with spherical radius 1/10.

The other way to create a `CMBDataFrame` object from an object of class `mmap` is to use the `CMBDataFrame` function. We can import either a simple random sample, a specific set of indices, or the full sky map:

```
# Import a uniform sample of 1,000,000 pixels
sky.sample <- CMBDataFrame(map, sample.size = 100000)

# Import a specific set of indices
indices <- c(1:100, 1111, 1112, 1113, 12000000:12000100)
sky.subset <- CMBDataFrame(map, spix = indices)

# Import the full sky map.
# Warning: The following may overload your memory limits
fullsky <- CMBDataFrame(map)
```

The objects `sky.sample`, `sky.subset`, and `fullsky` are all now of class `CMBDataFrame`. So, we can use these as we would any other `CMBDataFrame` object in the coming sections. If we wish to assign spherical or cartesian coordinates to these objects, we can do so using the `coords` function:

```
coords(sky.sample) <- "cartesian"
```

6 Subsetting and combining

We have already seen, in Section 5, a single use case of the `window` function. In this section, we go into more depth on `window` and the associated `CMBWindow` class. Class `CMBWindow` is designed to carry geometrical information describing the interior or exterior of either a closed polygon, or a closed disc, on $\mathbb{S}_2$. The polygons can be non-convex, though `CMBWindow` carries a boolean attribute `assumedConvex` that should be set to `TRUE`, if the polygon is known in advance to be convex, in order to decrease computation times. We can create a `CMBWindow` object using the `CMBWindow` function. For a polygon, we specify the vertices in a counterclockwise order (understood from outside the sphere looking in towards the origin) using either spherical or cartesian coordinates. For a disc, we give its radius r, along with the location on $\mathbb{S}_2$ of its center point.

```
# Specify a polygon using spherical coordinates
polygon <- CMBWindow(phi = c(0, pi/4, pi/4, pi/5),
                     theta = c(pi/2, pi/2, pi/4, pi/2 - pi/20))

# Specify a disc using spherical coordinates
disc <- CMBWindow(theta = pi/2, phi = -pi/8, r = 0.2)
```

To inspect these `CMBWindow` objects using interactive 3D graphics, we can pass them to the generic plot function. Below, we also plot the data `sky.sample` from Section 5 as a visual aid. See the resulting plot in Figure 5.

```
plot(sky.sample, back.col = "white")
plot(disc); plot(polygon)
summary(polygon); summary(disc)
```

```
    +---------------------------+   +---------------------------+
    |                           |   |                           |
    |   Window type: polygon    |   |   Window type: disc       |
    |   Window area: 0.299      |   |   Window area: 0.1252     |
    |                           |   |                           |
    +---------------------------+   +---------------------------+
```

Note, for a `CMBWindow` polygon to be well defined, the entire polygon must lie within any one hemisphere of $\mathbb{S}_2$. To obtain `CMBWindow` objects that occupy more than one hemisphere, we can specify a polygon or disc exterior using the `set.minus = TRUE` parameter:

```
# Exterior of a disc of radius 0.5
d.exterior <- CMBWindow(theta = pi/2, phi = 0, r = 0.5,
                        set.minus = TRUE)
```

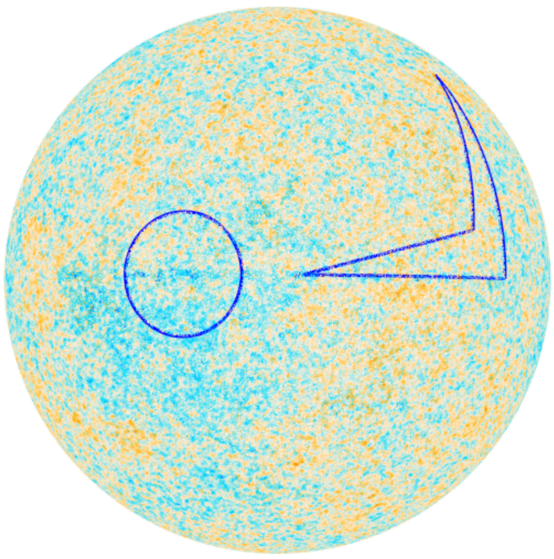


Figure 5: Boundary visualisation of polygon and disc CMBWindow objects, plotted against 10^5 CMB intensities sampled uniformly from the sky.

To specify more complicated regions, we can combine multiple `CMBWindow` objects into a `list` :

```
# List containing d.exterior and the interior of a disc of r = 1
wins <- list(d.exterior, CMBWindow(theta = pi/2, phi = 0, r = 1))
```

By passing `CMBWindow` objects to our `window` function, we can extract data from a `CMBDataFrame` or `mmap` object. Below, using our `list` called `wins` , we extract data from the `mmap` object called `map` , created in Section 5. The output `CMBDataFrame` is named `sky.annulus` . See the plot in Figure 6.

```
sky.annulus <- window(map, new.window = wins)
plot(sky.annulus, back.col = "white")
plot(wins[[1]], lwd = 5); plot(wins[[2]], lwd = 5)
```

In the case above, the default behaviour of `window` is to return the annulus that is the intersection of the 2 windows in `wins` . There is an option to alter this default behaviour by passing the parameter value `intersect = FALSE` to the `window` function. For more information on the behaviour of `window` and the `intersect` parameter, consult the `rcosmo` documentation.[4]

[4] `rcosmo` documentation is available here: `https://github.com/VidaliLama/rcosmo`

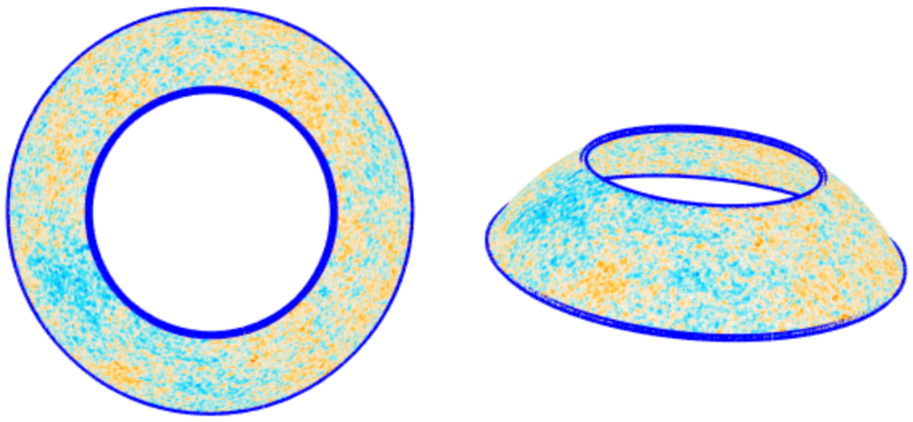


Figure 6: CMB intensity data extracted from an `mmap` object by the `window` function. Includes window boundary plot in blue. Two perspectives shown.

The `rbind` and `cbind` generics that work with the `data.frame` class have been customised in `rcosmo` to preserve the validity of `CMBDataFrame` objects. In particular, any `CMBWindow` objects belonging to a `CMBDataFrame` are preserved in the output of `rbind`. For more information on `rbind` and `cbind`, see the `rcosmo` documentation.

7 Statistical methods

In this section, we demonstrate how to produce empirical covariance estimates, approximate angular spectra, and simulate random fields on a HEALPix grid using `rcosmo`. The `rcosmo` functions that we cover here are created to work optimally with `CMBDataFrame` objects. Previously, in Sections 6 and 5, we demonstrated a variety of ways to create objects of class `CMBDataFrame`. After a `CMBDataFrame` has been created, it can act as a `data.frame` for the purpose of calculating basic statistics using built-in R functions, or using other R packages. Thus, we include a short tutorial on calculating basic statistics such as the mean, variance, and first Minkowski functional with a `CMBDataFrame`.

7.1 Calculating basic statistics

In this subsection, we demonstrate calculation of some common statistics, using `CMBDataFrame` objects together with built-in R functions. We will calculate the mean, variance, and first Minkowski functional of the `CMBDataFrame` we created in Section 5, called `sky.sample`. This also allows us to demonstrate a use case for the temperate `TMASK`, present in `fullsky`, for masking pixels that correspond to regions of the sky with high background interference.

Since `CMBDataFrame` objects inherit structure from class `data.frame`, we can treat `fullsky` as if it is a `data.frame`. The CMB intensities are stored by default in a column labelled `I`, which can be accessed using the `$` operator.

```r
mean(fullsky$I)
var(fullsky$I)
```

```
> mean(fullsky$I)
[1] 2.272742e-13
> var(fullsky$I)
[1] 1.037107e-08
```

Another column, `TMASK`, contains values 0 or 1. Rows containing 0 correspond to pixels that contain high background noise, e.g., radiation emanating from nearby bodies in the milky way. We can apply the mask easily, discarding pixels that have high noise. The resulting plot is given in Figure 7.

```r
# Apply the temperature mask
```

```r
sky.masked <- fullsky[as.logical(fullsky$TMASK),]
# Visualise the temperature mask
plot(fullsky, col = fullsky$TMASK + 1,
   sample.size = 100000)

# Calculate var and mean on masked sky
mean(sky.masked$I)
var(sky.masked$I)
```

```
> mean(sky.masked$I)
[1] 9.377827e-07
> var(sky.masked$I)
[1] 1.022269e-08
```

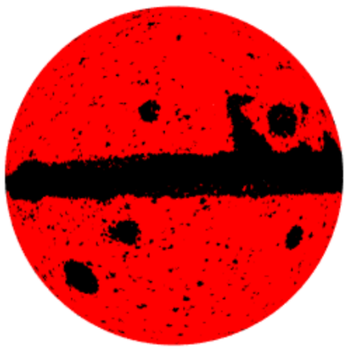


Figure 7: Visualisation of temperature mask `TMASK` . Black pixels are masked and red pixels are unmasked. The mask is concentrated around the equator, in plane with the milky way galaxy.

Let $\alpha \in \mathbb{R}$ be some constant. Given a random field $\{X_t\}_{t \in H_N}$ where H_N denotes the set of HEALPix pixel center locations at resolution $N = N_{\text{side}}$, the first Minkowski functional $M(X_t)$ is defined as

$$M(X_t) = \text{Area}\{t \in H_N : X_t > \alpha\} = A_N \sum_{t \in H_N} \mathbb{1}_{\{X_t > \alpha\}},$$

where A_N is the area of a single HEALPix pixel at resolution N_{side}, and is given by

$$A_N = \frac{4\pi}{N_{\text{pix}}} = \frac{\pi}{3N_{\text{side}}^2}.$$

Thus we can calculate the first Minkowski functional with $\alpha = 0$ as follows, using the `rcosmo` built in function `pixelArea` to simplify our code:

```
# Set alpha and find the pixel area at resolution nside
alpha <- 0; A <- pixelArea(sky.sample)

# Calculate the first Minkowski functional
fmf <- A*sum(sky$I > alpha); fmf
```

```
> fmf <- A*sum(sky$I > alpha); fmf
[1] 6.302987
```

From the output we see that the CMB temperature exceeds 0 for approximately half of the sky, since `fmf` $\approx \text{Area}(\mathbb{S}_2)/2$. This is expected, since `I` is mean-centered.

7.2 Covariance function: `covCMB`

In this subsection, we explain some of the theory behind `rcosmo`'s covariance estimation function `covCMB`. We then give a short tutorial on basic usage of the current version of `covCMB`.

Future versions of `covCMB` will allow anisotropic covariance function estimation. However, in the current version of `rcosmo`, the data are assumed to be samples from an isotropic random field. Put simply, this means that empirical covariance function estimation in `rcosmo` works much like a histogram. In the simplest case, the parameters we must specify are the number of bins, `num.bins`, in which to divide the data, and the maximum distance of interest, `max.dist`. By default, `covCMB` then produces a grid, `breaks`, of radii that specify the boundaries of intervals $R_i,\ i \in \{1, \dots,$ `num.bins` $\}$ that correspond to equiareal spherical annuli. We also define a zeroth interval, $R_0 = \{0\}$, corresponding to a radius of 0 (variance). Then, letting r_i denote the mid-point of interval R_i, and defining the set $R_i(t) := \{s \in H_N : d(s,t) \in R_i\}$, where $H_N \subseteq \mathbb{S}_2$ denotes the set of pixel centers on the HEALPix grid at a given resolution $N = N_{\text{side}}$, a covariance estimate $\hat{C}(r_i)$ is produced using the estimator:

$$\hat{C}(r_i) = \frac{1}{|H_N||R_i(t)|} \sum_{t \in H_N} \sum_{s \in R_i(t)} K_t K_s,$$

where $K_t, K_s \in \mathbb{R}$ are the mean-centered intensities, sampled within the HEALPix pixels with centers t and s respectively. Note that to define the spherical annuli as equiareal, with spherical area A, we ensure that the end-points $b_1^{(i)}, b_2^{(i)}$ of the interval $R_i,\ i > 0$, with $b_1^{(i)} < b_2^{(i)}$, satisfy

$$\cos(b_1^{(i)}) - \cos(b_2^{(i)}) = A.$$

For CMB empirical covariance estimation we should make the following consideration: Any given CMB photon received at a detector today has been travelling for around 14 billion years; and, the point from which it was emitted is now about 46 billion light years away [3]. So, with `num.bins` $= 1000$, and maximum distance of `max.dist` $= 0.03$ radians, the first bin in empirical covariance function covers a disc of radius roughly 1.4 million light years at the source of radiation.

Below, we use the `CMBDataFrame` called `sky.sample`, from Section 5, to demonstrate covariance function estimation with `covCMB`.

```
ecov <- covCMB(sky.sample, max.dist = 0.03, num.bins = 200)
```

Executing `head(ecov, n = 3)` in the R console gives the following:

```
         dist          cov      n
1 0.000000000 1.034981e-08 100000
2 0.001060621 9.511345e-09   2696
3 0.002560565 9.112465e-09   2278
```

The above are the first 3 of 200 rows from `ecov`. Each row corresponds to a bin with center given by `dist`, and estimated covariance `cov`. The column `n` counts the number of data point pairs in each bin. The first row, having distance 0, represents variance. Hence, we produce the Figure 8 correlation plot as follows.

```
plot(ecov$dist, ecov$cov/sky$cov[1], main = "Empirical CMB
   Correlation", xlab = "Spherical distance", ylab =
   "Correlation")
```

7.3 Random field simulation

The theory described in this subsection will be the basis of the `rcosmo` function `simulateRF`, which is under development and will be included in a coming update. This function will allow fast generation of simulated isotropic Gaussian random fields on the HEALPix grid.

Any mean-square continuous L_2 random field $\{X_t\}_{\in \mathbb{S}_2}$, $t = (\theta, \phi)$, has an expansion in terms of spherical harmonics $Y_\ell^m(t)$:

$$X_t = \sum_{\ell=0}^{\infty} \sum_{m=-\ell}^{\ell} Z_\ell^m Y_\ell^m(t), \tag{7.1}$$

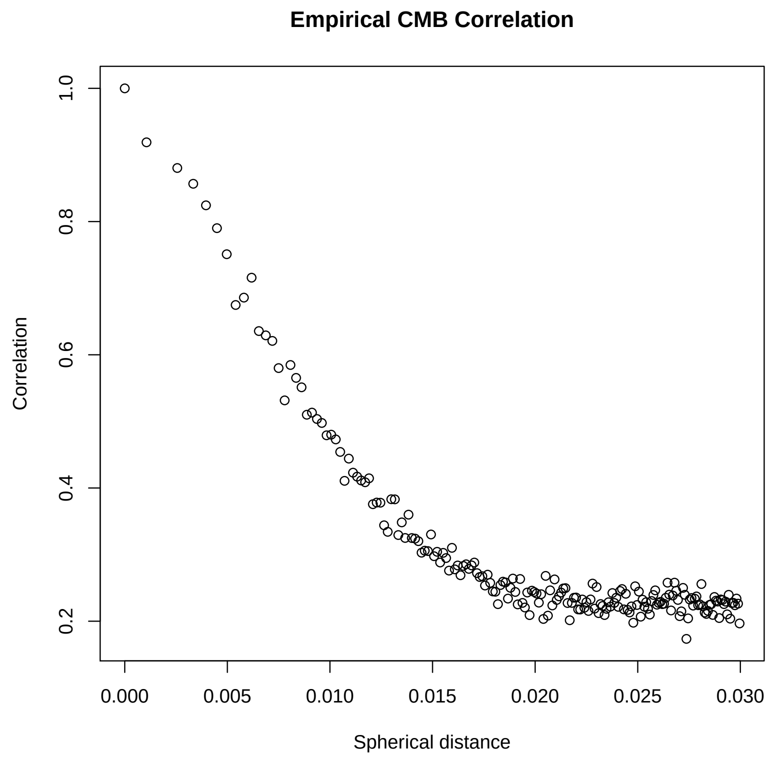


Figure 8: Plot of CMB correlation estimates using 200 bins on the interval $[0, 0.03)$.

where the coefficients Z_ℓ^m are complex mean-zero random variables satisying [2]

$$Z_\ell^m = (-1)^m \overline{Z_\ell^{-m}}, \tag{7.2}$$

From (**??**) and (**??**) we can see that

$$Y_\ell^m(\theta,\phi) = (-1)^m \overline{Y_\ell^{-m}(\theta,\phi)}. \tag{7.3}$$

Expansion (7.1) admits the following simplification

$$\begin{aligned} X_t &= Z_0^0 Y_0^0(\,t\,) + \sum_{\ell=1}^{\infty}\sum_{m=1}^{\ell}[Z_\ell^m Y_\ell^m(\,t\,) + Z_\ell^{-m} Y_\ell^{-m}(\,t\,)] \\ &= \frac{Z_0^0}{2\sqrt{\pi}} + \sum_{\ell=1}^{\infty}\sum_{m=1}^{\ell}\text{Re}(Z_\ell^m Y_\ell^m(\,t\,)), \end{aligned} \tag{7.4}$$

since $Y_0^0(\,t\,) \equiv (2\sqrt{\pi})^{-1}$ and since (7.3) and (7.2) give us

$$\begin{aligned} Z_\ell^m Y_\ell^m(\,t\,) + Z_\ell^{-m} Y_\ell^{-m}(\,t\,) &= Z_\ell^m Y_\ell^m(\,t\,) + (-1)^{2m}\overline{Z_\ell^m}\;\overline{Y_\ell^{-m}(\,t\,)} \\ &= Z_\ell^m Y_\ell^m(\,t\,) + \overline{Z_\ell^m Y_\ell^{-m}(\,t\,)}. \end{aligned}$$

Now, from (7.4) and (**??**) we have

$$X_t = \frac{Z_0^0}{2\sqrt{\pi}} + \sum_{\ell=1}^{\infty}\sum_{m=1}^{\ell} Z_\ell^m \lambda_\ell^m(\cos\theta)\cos(m\phi)$$

So, using (**??**) we get

$$X_t = \frac{Z_0^0}{2\sqrt{\pi}} + \sum_{\ell=1}^{\infty}\sum_{m=1}^{\ell} Z_\ell^m \sqrt{\frac{2\ell+1}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}} P_\ell^m(\cos\theta)\cos(m\phi). \tag{7.5}$$

If the Z_ℓ^m have mean zero and are iid, then this implies that $\{X_t\}_{t\in\mathbb{S}_2}$ is isotropic and Gaussian [18, Section 2, p.2]. Thus, using a finite number of terms from the above formula, with appropriate distribution for the coefficients Z_ℓ^m, we can simulate an isotropic Gaussian random field on $\mathbb{S}_2$.

7.1 Example. We choose the following angular power spectrum

$$f_\ell = \frac{2}{(\ell(\ell+1)+4)^2}.$$

This is the same function chosen for the simulation in [2]. We then generate independent normal random variables $Z_\ell^m \sim N(0, f_\ell)$ for each $\ell \in \{1, \dots, 100\}$ and all $m \in \{1, \dots, \ell\}$. Then, we can calculate the first 100 terms of (7.5) for $t = (\theta, \phi)$ on a HEALPix grid from a `CMBDataFrame`. The resulting plot is given in Figure 9.

```
# Using library gsl to calculate legendre functions
library(gsl)
library(rcosmo)

# The maximum value of l
Nl <- 200

# The angular power spectrum
fl <- 2/((1:Nl)*((1:Nl)+1)+4)^2
# Independent normal random variables
ZA <- sapply(1:Nl, function(x){rnorm(x,0,fl[x])})
# Data frame with theta and phi HEALPix centers.
df <- CMBDataFrame(nside = 32, coords = "spherical",
                   ordering = "ring")
theta <- df$theta
phi <- df$phi

# Calculate the coefficient not depending on theta and phi
```

```
for (l in 1:Nl){ for (m in 1:l) {
   fact <- prod(l + seq(m,-m+1)) # factorial(l+m)/factorial(l-m)
   coef <- (2*l+1)/(4*pi)/fact
   ZA[[l]][m] <- ZA[[l]][m]*sqrt(coef)
}}
# Vectorise calculation of legendre funcs with package gsl
result <- vector(mode = "numeric", length = nrow(df))
for (l in 1:Nl){ for (m in 1:l) {
   Plm <- gsl::legendre_Plm(l, m, cos(theta))
   result <- result + ZA[[l]][m]*Plm*cos(m*phi)
}}
# Store result in CMBDataFrame intensity column
df$I <- result; plot(df, size = 3)
```

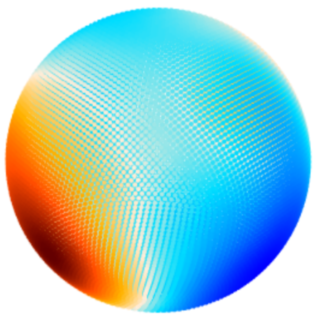


Figure 9: Plot of a random field simulated on the HEALPix grid with $N_{\text{side}} = 32$.

7.4 Angular spectrum estimation

The theory described in this subsection will be the basis of the `rcosmo` function `specCMB`, which is under development and will be included in a coming update. This function will allow fast estimation of angular spectra given a single realisation of a an isotropic Gaussian random field, sampled on a HEALPix grid.

Let $\{X_t\}_{t\in\mathbb{S}_2}$ be a mean-square continuous and isotropic Gaussian random field on the unit sphere $\mathbb{S}_2$. Then, the covariance function $C(t,s)$ depends only on the geodesic distance $\gamma \in [0,\pi)$ between t and s. We can thus find an appropriate function B such that $C(t,s) = B(\cos\gamma)$. It is known [18, Section 3.2] that

$$B(\cos\gamma) = \sum_{\ell=0}^{\infty} f_\ell \frac{2\ell+1}{4\pi} P_\ell(\cos\gamma). \tag{7.6}$$

for expansion coefficients $f_\ell \in \mathbb{R}^+$, known as the *angular power spectrum* of $\{X_t\}_{t\in\mathbb{S}_2}$, which are shown in [2] and [18] to satisfy $f_\ell = \mathrm{Var}(Z_\ell^m)$, where the Z_ℓ^m are as in expansion (7.1). We could find the empirical variance directly with, e.g.,

$$\hat{f}_\ell = \frac{1}{2\ell+1} \sum_{m=-\ell}^{\ell} (Z_\ell^m)^2,$$

but this estimator faces reduced precision for small ℓ since $|m| \leq \ell$. Alternatively, we can proceed to define $\tau = \cos(\gamma)$ and, using (7.6) [2], notice that

$$\int_{-1}^{1} C(\tau)P_\ell(\tau)\,dt = \sum_{\ell'=0}^{\infty} f_{\ell'} \frac{2\ell'+1}{4\pi} \int_{-1}^{1} P_{\ell'}(\tau)P_\ell(\tau)\,d\tau.$$

Now, using (**??**) where $\delta_{\ell\ell'} = 0$ for $\ell \neq \ell'$ and $\delta_{\ell\ell'} = 1$ for $\ell = \ell'$, this becomes

$$\int_{-1}^{1} C(\tau)P_\ell(\tau)\,dt = f_\ell \frac{2\ell+1}{4\pi} \int_{-1}^{1} [P_\ell(\tau)]^2\,d\tau = \frac{f_\ell}{2\pi}.$$

We now have a practical formula for estimating the angular power spectrum f_ℓ:

$$f_\ell = 2\pi \int_{-1}^{1} C(\tau)P_\ell(\tau)\,dt.$$

Remark. There is a uniting framework between the spectral decomposition of a random field on $\mathbb{S}_2$ and the Euclidean counterpart (**??**). They fall under the more general spectral theory on groups known as the *representation theory of compact groups*, resting on a result known as the Peter-Weyl theorem [11, Section 2.5].

8 Geometrical tools

There are a number of functions in `rcosmo` for handling common spherical geometry calculations. We discuss these in this section. Where possible, we aim to take advantage of HEALPix properties to simplify computation. Perhaps the most basic and important geometrical quantities of interest are spherical angles, and spherical area. The latter can be given in terms of the former, using a special case of the Gauss-Bonnet theorem:

8.1 Proposition ([13]). *A spherical polygon with n interior angles α_i satisfies*

$$\sum_{i=1}^{n} \alpha_i + (n-2)\pi = A,$$

where A is the spherical area of the polygon.

In the interest of finding the angles, α_i, we should keep in mind that the edges of spherical polygons are geosdesic line segments. Thus, each edge E_i is representable as the intersection of $\mathbb{S}_2$ with some plane P_i through the origin. If p_i, a_i are any two points on E_i, then P_i is uniquely defined by the cross product $\nu = p_i \times a_i$. If p_i and a_i are chosen to be vertices, and the polygon is counterclockwise oriented, then ν points from E_i towards the polygon interior. Suppose we have a third vertex, q_i. Then, we can define an interior spherical angle $\alpha_i = \sphericalangle p_i a_i q_i$ at a_i, and this spherical angle must be less than π if $\nu \cdot q_i$ is positive. In terms of the scalar triple product, this is

$$(p_i \times a_i) \cdot q_i > 0. \tag{8.1}$$

If the angle less than π, then it can be calculated as

$$\alpha_i = \arccos \langle \nu, u \rangle, \tag{8.2}$$

where $u := a_i \times q_i$. In other words, α_i is equal to the angle between the normal vectors that define its two edge planes. If α_i is greater than π, i.e., if (8.1) is false, then α_i is simply equal to 2π minus the RHS of (8.2). This is how spherical angles are calculated in `rcosmo` . For example, the function `geoArea` takes a `CMBWindow` object and returns its area using Proposition 8.1.

```
# Specify a polygon using spherical coordinates
polygon <- CMBWindow(phi = c(0, pi/4, pi/4, pi/5),
                     theta = c(pi/2, pi/2, pi/4, pi/2 - pi/20))
geoArea(polygon)
```

```
> geoArea(polygon)
[1] 0.1262341
```

This geometrical area is useful, but there is one important consideration: The geometrical area will never exactly match the total sample pixel area, since the borders of HEALPix pixels are not great circle segments. So, no given circle or spherical polygon can coincide perfectly with a set of HEALPix pixel boundaries. We can find the exact area occupied by all pixels in a `CMBDataFrame` using the same generic function `geoArea` :

```
sky <- CMBDataFrame(nside = 32)
win <- window(sky, polygon)
geoArea(win)
```

```
> geoArea(win)
[1] 0.1431715
```

When `geoArea` is applied to a `CMBDataFrame` , such as `win` , it acts differently to when it is applied to a `CMBWindow` , such as `polygon` . Notice that the area occupied by the pixels at resolution $N_{\text{side}} = 32$ is quite larger than the result we found previously using the analytic approach. If we increase the resolution, this difference will decrease. However, for matters of accuracy, it may be important to decide on the most appropriate area calculation for a given use case.

9 Consistency testing

As development on `rcosmo` continues, new functions will be added and existing functions will be extended, simplified, or removed entirely. It is reasonable to expect that multiple developers may be contributing to this process. Inevitably, inconsistencies will be introduced into the package, leading to one of the following undesireable outcomes during normal execution:

- A function will crash unexpectedly, producing an error message.
- A function will not crash, but will return a logically inconsistent result, without producing a warning or error message.
- A function, having been removed, will no longer run at all (though the documentation has not been updated).

Unit testing is an approach taken by software developers to reduce the chances of such malfunctions, or *bugs*, and to mitigate the risks associated with their occurrence. In `rcosmo` , unit testing is achieved with assistance from `testthat` , an R package included in Hadley Wickham's `devtools` [21].

The intention is to provide `testthat` with small example procedures, describing the most critical `rcosmo` functionality. These example procedures are associated with logically consistent outputs. Having stored a large collection of such test procedures, an `rcosmo` developer can easily execute all tests by running the following command, whose current output is provided in Figure 10.

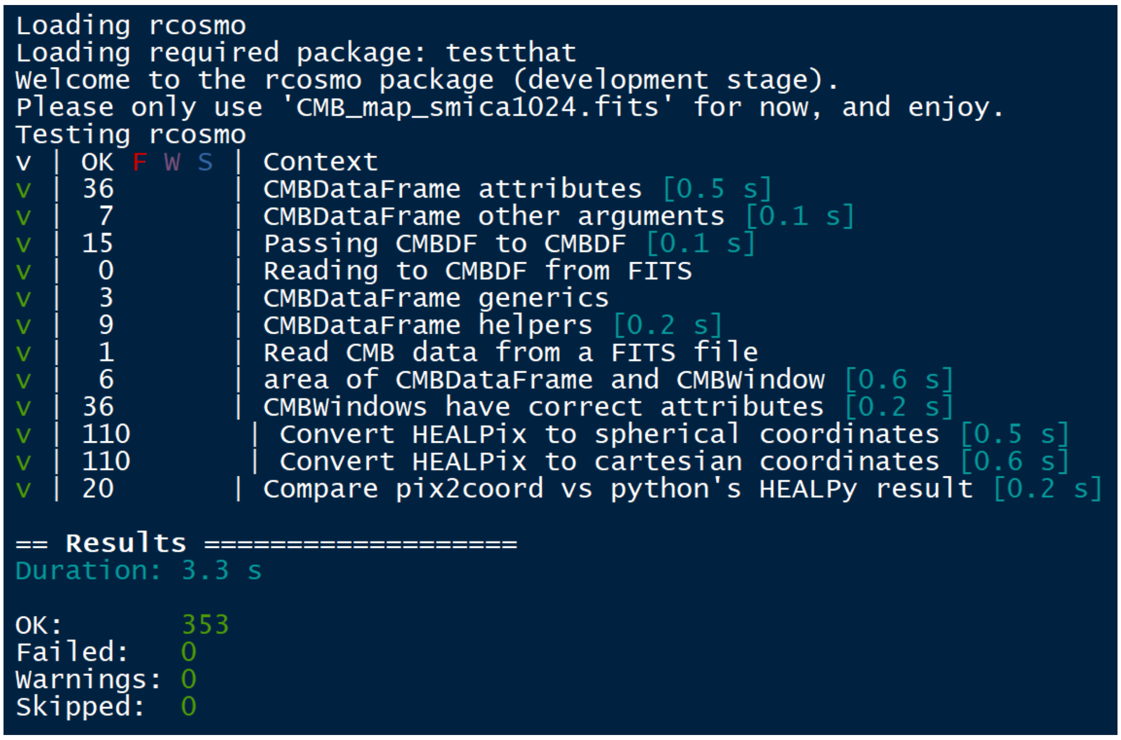
```
devtools::test("rcosmo")
```

In Figure 10, we can see that 353 individual unit tests have been executed. Each of these unit tests were created specifically for `rcosmo` . For example, by including the following code in our `testthat` procedures, we have tested that our conversions from the HEALPix ring ordering scheme to cartesian coordinates are in agreement, at $N_{\text{side}} = 2, 4, 8, 16, 32, 64$, with the results produced by the `python` CMB data analysis package `HEALPy` [15]:

```
# Name the testing procedures for reference
testthat::test_that("cartesian 'ring' data agrees with HEALpy
    result", {

  # Define nside parameter values
  ns <- 2^c(1,2,3,4,5)

  # Create vector of filenames containing valid HEALPy data
  filenames <- paste0("testdata/verify_py/cartesian_ring_ns",
```


```
Loading rcosmo
Loading required package: testthat
Welcome to the rcosmo package (development stage).
Please only use 'CMB_map_smica1024.fits' for now, and enjoy.
Testing rcosmo
v | OK F W S | Context
v | 36       | CMBDataFrame attributes [0.5 s]
v |  7       | CMBDataFrame other arguments [0.1 s]
v | 15       | Passing CMBDF to CMBDF [0.1 s]
v |  0       | Reading to CMBDF from FITS
v |  3       | CMBDataFrame generics
v |  9       | CMBDataFrame helpers [0.2 s]
v |  1       | Read CMB data from a FITS file
v |  6       | area of CMBDataFrame and CMBWindow [0.6 s]
v | 36       | CMBWindows have correct attributes [0.2 s]
v | 110        | Convert HEALPix to spherical coordinates [0.5 s]
v | 110        | Convert HEALPix to cartesian coordinates [0.6 s]
v | 20       | Compare pix2coord vs python's HEALPy result [0.2 s]

== Results ===================
Duration: 3.3 s

OK:       353
Failed:   0
Warnings: 0
Skipped:  0
```

Figure 10: Output for `rcosmo` unit testing process.

```
                  formatC(ns, width=2, flag="0"), ".rds")

  # Loop over all nside test cases
  for (i in 1:length(ns))
  {

    # Run the tests, expecting equal to python data
    eval(bquote(testthat::expect_equal( readRDS(filenames[.(i)]),
                     pix2coords_internal(nside = ns[.(i)],
                                         nested = FALSE,
                                         cartesian = TRUE)[,1:3] )))
  }
})
```

References

[1] D. Adler, D. Murdoch and W. Zucchini (2014) rgl: 3D visualization device system (OpenGL). R package version 0.93.1098.

[2] A. Baran and G. Terdik (2015) Power spectrum estimation of spherical random fields based on covariances. *Annales Matematicae at Informaticae*, 44:15–22.

[3] K. Bolejko (2011) The effect of inhomogeneities on the distance to the last scattering surface and the accuracy of the cmb analysis. *Journal of Cosmology and Astroparticle Physics*, 2011(2):25.
URL `http://stacks.iop.org/1475-7516/2011/i=02/a=025`

[4] P. Burns (2011) The R inferno.
URL `http://www.burns-stat.com`

[5] M. R. Calabretta and B. F. Roukema (2007) Mapping on the healpix grid. *Monthly Notices of the Royal Astronomical Society*, 381(2):865–872.

[6] D. Eddelbuettel, R. François, J. Allaire, K. Ushey, Q. Kou, N. Russel, J. Chambers and D. Bates (2011) Rcpp: Seamless R and C++ integration. R package version 0.12.17.

[7] K. M. Gorski, E. Hivon, A. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke and M. Bartelmann (2005) Healpix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2):759.

[8] A. Harris (2016) Package FITSio. CRAN. Version 2.1-0.

[9] G. Hurier, M. Douspis, N. Aghanim, E. Pointecouteau, M. Diego, J and F. Macias-Perez, J (2015) Cosmological constraints from the observed angular cross-power spectrum between Sunyaev-Zeldovich and x-ray surveys. *Astronomy and Astrophysics*, 576(90).

[10] International Astronomical Union (2008) Definition of the Flexible Image Transport System (FITS), 3.0 edition.
URL `http://fits.gsfc.nasa.gov/iaufwg`

[11] D. Marinucci and G. Peccati (2011) Random fields on the sphere: Representation, limit theorems and cosmological applications. Cambridge University Press.

[12] P. Newman (2018). North American Space Agency. Accessed June 2018. URL `https://asd.gsfc.nasa.gov/archive/arcade/cmb_intensity.html`

[13] B. O'neill (2006) Elementary differential geometry. Academic press.

[14] Penn State Gravitational-Wave Astronomy Group (2016) MEALPix. MATLAB package version 3.0.

[15] C. Rosset and A. Zonca (2018) HEALPy. Python library version 1.

[16] B. F. Roukema and B. Lew (2004) A solution to the isolatitude, equi-area, hierarchical pixel-coordinate system. *arXiv preprint astro-ph/0409533*.

[17] J. A. Ryan (2018) mmap: R interface to POSIX mmap and Window's MapViewOfFile. R package Version 0.6-15.

[18] G. Terdik et al. (2015) Angular spectra for non-gaussian isotropic fields. *Brazilian Journal of Probability and Statistics*, 29(4):833–865.

[19] H. Wickham (2014) Advanced R. CRC Press.

[20] H. Wickham (2015) R packages: Organize, test, document, and share your code. O'Reilly Media, Inc.

[21] H. Wickham and W. Chang (2018) Devtools: Tools to make developing R packages easier. R package version 1.13.5.

[22] Y. Xie (2018) knitr. CRAN. R package version 1.20.

[23] Y. Xie (2018) rmarkdown. CRAN. R package version 1.10.