Package 'lunar'

June 13, 2022

Type Package

Title Calculate Lunar Phase & Distance, Seasons and Related Environmental Factors

Author Emmanuel Lazaridis [aut, cre]

Maintainer Emmanuel Lazaridis <emmanuel@strategicarrow.com>

Depends R (>= 2.10.0)

Description Provides functions to calculate lunar and other related environmental covariates.

License MIT + file LICENSE

Encoding UTF-8

LazyLoad no

Version 0.2-1

Date 2022-06-13

RoxygenNote 7.2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-13 15:40:02 UTC

R topics documented:

`lunar-package` *The lunar Package*

Description

Lunar Phase & Distance, Seasons and Other Environmental Factors.

Provides functions to calculate the phase of the moon, its distance from the earth, the season and possibly other environmental factors, based on date and location.

Details

Package:	lunar
Type:	Package
Version:	0.2-01
Date:	2022-06-13
Depends:	R (>= 2.10.0)
Encoding:	UTF-8
License:	MIT
LazyLoad:	no

Provides functions to calculate lunar and other environmental covariates.

This package is used by the author to calculate covariates for studies of lunar effect on health and healthcare. It is based on an astronomical calculation that considers the moon's phase and its distance from the earth to calculate theoretical relative amount of moonlight on any given night. The package is not based on luminance data. It does not consider local cloud cover, for example, so it only provides maximum possible moonlight, not actual moonlight.

References forthcoming.

Author(s)

Emmanuel Lazaridis

`lunar.4phases` *Lunar Phase Categories (4)*

Description

Return 4 category labels for lunar phases.

Usage

```
lunar.4phases
```

Format

An object of class `character` of length 4.

Details

These are category names corresponding to phases of the moon. Moon phase category names may be returned in the output of the `lunar.phase` function if its `name` option is set to `TRUE`.

See Also

`lunar.phase`

Examples

```
print(lunar.4phases)
```

`lunar.8phases` *Lunar Phase Categories (8)*

Description

Return 8 category labels for lunar phases.

Usage

```
lunar.8phases
```

Format

An object of class `character` of length 8.

Details

These are category names corresponding to phases of the moon. Moon phase category names may be returned in the output of the `lunar.phase` function if its `name` option is set to `TRUE`.

See Also

`lunar.phase`

Examples

```
print(lunar.8phases)
```

lunar.distance *Lunar Distance*

Description

Returns the distance of the moon from the earth on specified dates.

Usage

```
lunar.distance(x, shift = 0, ..., name = FALSE, strict = FALSE)
```

Arguments

x	A vector of Date values.
shift	The number of hours by which to shift the distance calculation. By default distance is calculated at 12 noon UT.
...	Other optional arguments are ignored.
name	Optional parameter indicating whether the return is a factor variable consisting of a lunar distance label, or the lunar distance in earth radii. By default lunar phase is returned in earth radii.
strict	Optional parameter indicating whether the return should employ strict definitions for distance labels, that is, with apogee and perigee within 5 definition breaks the distance categories evenly into 20 The 'average' category is the same in both definitions.

Details

Distance to the moon is returned in units of earth radii, or as a 5-level factor variable referring to the moon's perigee (at about 56 earth radii) and apogee (at about 63.8 earth radii).

Adapted from Stephen R. Schmitt: Lunar Phase Computation: https://web.archive.org/web/20140716104947/http://mysite.verizon.net/res148h4j/zenosamples/zs_lunarphasecalc.html, Last accessed: 1 September 2014.

See Also

lunar.distances

Examples

```
lunar.distance(as.Date("2004-03-24"))
```

lunar.distance.mean *Average Lunar Distance*

Description

Returns the average lunar distance around specified dates.

Usage

```
lunar.distance.mean(
  x,
  towards = -6,
  ...,
  by = c("date", "hours", "day", "night")
)
```

Arguments

x	A vector of Date values.
towards	The directed window size from x in days. By default the window looks back 7 days including x.
...	Other optional arguments are ignored.
by	The exposure interval and integration basis. The default is to represent a day's distance by the distance at 12 noon UT. The other options integrate midrange distances over hours.

See Also

lunar.distance

Examples

```
lunar.distance.mean(as.Date("2004-03-24"))
```

lunar.distances *Lunar Distance Categories*

Description

Return category labels for lunar distances.

Usage

```
lunar.distances
```

Format

An object of class `character` of length 5.

Details

These are category names corresponding to distances from the center of the Earth to the center of its moon. Distance category names are used in the output of the `lunar.distance` function if its `name` option is set to `TRUE`.

The perigee occurs at roughly 56 Earth radii, and the apogee at about 62.8 Earth radii. These categories are not determined according to any common standard. They may have different precise definitions for their bounds in different analyses.

See Also

`lunar.distance`

Examples

```
print(lunar.distances)
```

---

lunar.illumination *Lunar Illumination*

Description

Returns the proportion of lunar illumination on specified dates.

Usage

```
lunar.illumination(x, shift = 0)
```

Arguments

`x`	A vector of `Date` values.
`shift`	The number of hours by which to shift the calculation of lunar phase. By default lunar phase is calculated at 12 noon UT.

Details

Adapted from function `moon.illumination` in from the R4MFCL project (not an R package), which was developed by the Secretariat of the Pacific Community (SPC). The R4MFCL project was led by Simon Hoyle, and also includes code by Shelton Harley, Nick Davies, and Adam Langley of the SPC, and Pierre Kleiber of the US National Marine Fisheries Service. Pierre Kleiber is the author of the `moon.illumination` function.

Code from project R4MFCL is distributed under the MIT License:

https://opensource.org/licenses/mit-license.php

Here is a link to the R4MFCL project:

https://code.google.com/archive/p/r4mfcl/

See Also

lunar.illumination.mean

https://stackoverflow.com/questions/71757462/calculate-lunar-illumination-using-the-lunar-package-i

Examples

```
lunar.illumination(as.Date("2004-03-24"))
```

`lunar.illumination.mean`

Average Lunar Illumination

Description

Returns the average lunar illumination around specified dates.

Usage

```
lunar.illumination.mean(
  x,
  towards = -6,
  ...,
  by = c("date", "hours", "day", "night")
)
```

Arguments

x	A vector of Date values.
towards	The directed window size from x in days. By default the window looks back 7 days including x.
...	Other optional arguments are ignored.
by	The exposure interval and integration basis. The default is to represent a day's illumination by the illumination at 12 noon UT. The other options integrate midrange illuminations over hours. Options 'day' and 'night' are not currently implemented, but will be used to limit exposure intervals. The use of an unimplemented option in a function call will result in a NULL value being returned.

See Also

lunar.illumination

Examples

```
lunar.illumination.mean(as.Date("2004-03-24"))
```

`lunar.metric.mean` *Average Lunar Metrics*

Description

Returns an average measurement around specified dates.

Usage

```
lunar.metric.mean(
  x,
  towards = -6,
  ...,
  by = c("date", "hours", "day", "night"),
  type = c("illumination", "distance")
)
```

Arguments

`x`	A vector of `Date` values.
`towards`	The directed window size from `x` in days. By default the window looks back 7 days including `x`.
`...`	Other optional arguments are ignored.
`by`	The exposure interval and integration basis. The default is to represent a day's illumination by the illumination at 12 noon UT. The other options integrate midrange illuminations over hours. Options 'day' and 'night' are not currently implemented, but will be used to limit exposure intervals. The use of an unimplemented option in a function call will result in a `NULL` value being returned.
`type`	Whether illumination or distance metrics are to be returned. The use of an unimplemented option in a function call will result in a `NULL` value being returned.

Details

This in an internal support function that integrates a lunar measurement over time using step sizes of days or hours.

See Also

`lunar.illumination`

`lunar.distance`

Examples

```
## Not run:
lunar.metric.mean(as.Date("2004-03-24"), type="illumination")

## End(Not run)
```

`lunar.phase` *Lunar Phase*

Description

Returns the lunar phase on specified dates.

Usage

```
lunar.phase(x, shift = 0, ..., name = FALSE)
```

Arguments

`x`	A vector of `Date` values.
`shift`	The number of hours by which to shift the calculation of lunar phase. By default lunar phase is calculated at 12 noon UT.
`...`	Other optional arguments are ignored.
`name`	Optional parameter indicating whether the return is a factor variable By default lunar phase is returned in radians. If assigned the value 8, it returns a factor variable with 8 phase levels. If `TRUE` or any value other than 0 or 8, it returns a factor variable with 4 phase labels.

Details

Adapted from function `moon.illumination` in from the R4MFCL project (not an R package), which was developed by the Secretariat of the Pacific Community (SPC). The R4MFCL project was led by Simon Hoyle, and also includes code by Shelton Harley, Nick Davies, and Adam Langley of the SPC, and Pierre Kleiber of the US National Marine Fisheries Service. Pierre Kleiber is the author of the `moon.illumination` function.

Code from project R4MFCL is distributed under the MIT License:

https://opensource.org/licenses/mit-license.php

Here is a link to the R4MFCL project:

https://code.google.com/archive/p/r4mfcl/

The R4MFCL code was modified as follows:

- Changed function name from `moonphase` to `lunar.phase`.
- Changed input date to be in `Date` format (as opposed to `POSIXct`).
- Removed reliance on other R4MFCL functions.
- Changed name of primary input from `ptime` to `x`.
- Added optional `shift` term (in hours) relative to 12h UT.
- Added optional `name` term to control whether phase names as opposed to radians should be returned.
- Changed the documentation.

Where radians are returned:

- 0 refers to the new moon
- $\pi/2$ refers to the first quarter
- π refers to the full moon
- $3\pi/2$ refers to the last quarter

Adapted originally from Stephen R. Schmitt: Sky & Telescope, Astronomical Computing, April 1994 and https://web.archive.org/web/20140716104947/http://mysite.verizon.net/res148h4j/zenosamples/zs_lunarphasecalc.html, which references Jean Meeus, Astronomical Algorithms. Willmann-Bell, Inc. (1991) 429p.

See Also

lunar.4phases

lunar.8phases

Examples

```
lunar.phase(as.Date("2013-05-06"))
```

---

terrestrial.season *Terrestrial Season*

Description

Returns the season on specified dates.

Usage

```
terrestrial.season(
  x,
  cutoffs = c(80, 172, 263, 354),
  southern.hemisphere = FALSE
)
```

Arguments

x	A vector of Date values.
cutoffs	A vector of numbers corresponding to days of the year when season labels change.
southern.hemisphere	The season labels follow a northern hemisphere order unless this option is set to TRUE.

Details

The definitions for non-leap years are as follows (dates are inclusive):

Winter: 21 December through 21 March

Spring: 22 March through 21 June

Summer: 22 June through 20 September

Autumn: 21 September through 20 December

In leap years spring comes a day early! Thanks to Mehis Rohtla for finding an error in the 0.1-04 version code.

See Also

terrestrial.seasons

Examples

```
terrestrial.season(as.Date("2017-03-21"))
terrestrial.season(as.Date("2017-03-22"))
terrestrial.season(as.Date("2017-12-20"), southern.hemisphere = TRUE)
terrestrial.season(as.Date("2017-12-21"), southern.hemisphere = TRUE)
```

---

terrestrial.seasons *Terrestrial Season Categories*

Description

Return category labels for the seasons on Earth.

Usage

```
terrestrial.seasons
```

Format

An object of class `character` of length 4.

Details

These are category names corresponding to the seasons of the planet Earth.

Examples

```
print(terrestrial.seasons)
```

Index