Package ‘skyscapeR’

October 14, 2022

Type Package

Title Data Analysis and Visualization for Skyscape Archaeology

Version 1.0.0

Date 2021-10-29

Description Data reduction, visualization and statistical analysis of measurements of orientation of archaeological structures, following Silva (2020) <doi:10.1016/j.jas.2020.105138>.

Depends R (>= 4.1), swephR

Imports png, oce, utils, stats, graphics, grDevices, MESS,
RColorBrewer, numDeriv, parallel, foreach, doParallel,
rootSolve, httr, plotrix, zoo

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Silva Fabio [aut, cre],
Reijs Victor [ctb]

Maintainer Silva Fabio <fsilva@bournemouth.ac.uk>

Repository CRAN

Date/Publication 2021-10-29 22:40:02 UTC

R topics documented:

antizenith *Declination of the anti-zenith sun for a given location*

Description

This function returns the declination of the sun when it is at the anti-zenith for a given location with corrected average parallax. If this phenomena does not occur at given location (i.e. if location is outside the tropical band) the function returns a *NULL* value.

Usage

```
antizenith(loc, parallax = 0.00224, altitude = 0)
```

Arguments

`loc`	This can be either the latitude of the location, or a *skyscapeR.horizon* object.
`parallax`	(Optional) Average parallax value for the sun. Defaults to 0.00224.
`altitude`	(Optional) Altitude of the sun. Defaults to 0 degrees.

See Also

`jS`, `dS`, `eq`, `zenith`, `spatial.equinox`, `parallax.corr`

Examples

```
# Anti-zenith sun declination for Mexico City:
antizenith(19.419)

# There is no anti-zenith sun phenomena in London:
antizenith(51.507)
```

az.pdf *Convert discrete azimuth measurements into probability distributions*

Description

Convert discrete azimuth measurements into probability distributions

Usage

```
az.pdf(
  pdf = "normal",
  az,
  unc,
  name,
  verbose = T,
  .cutoff = 1e-04,
  .res = 0.01
)
```

Arguments

pdf	(Optional) String describing the probability distribution to be used. At the moment only *normal* and *uniform* are supported. Default is *normal*
az	An array of azimuths
unc	Azimuth uncertainties as either an array of the same length as *az* or a single value to be applied to all measurements
name	(Optional) An array of names to identify each measurement
verbose	(Optional) Boolean to control whether or not to display text. Default is TRUE.
.cutoff	(Optional) Value of probability distribution(s) at which point it will be cutoff to save on memory. Default is 1e-4
.res	(Optional) Azimuth resolution with which to output probability distribution(s). Default is 0.01 degrees.

References

Silva, F (2020) A probabilistic framework and significance test for the analysis of structural orientations in skyscape archaeology *Journal of Archaeological Science* 118, 105138. <doi:10.1016/j.jas.2020.105138>

Examples

```
test <- az.pdf(az=c(87,93,90,110), unc=3)
plot(test)
```

az2dec *Calculates declination from azimuth and altitude measurements*

Description

This function calculates the declination corresponding to an orientation , i.e. an azimuth. The altitude can either be given or, alternatively, if a *skyscapeR.horizon* object is provided, the corresponding horizon altitude will be automatically retrieved. This function is a wrapper for function swe_azalt_rev of package *swephR*.

Usage

```
az2dec(
  az,
  loc,
  alt,
  refraction = skyscapeR.env$refraction,
  atm = skyscapeR.env$atm,
  temp = skyscapeR.env$temp
)
```

Arguments

`az` Azimuth(s) for which to calculate declination(s). See examples below.

`loc` Location, can be either a *skyscapeR.horizon* object or, alternatively, an array of latitude values.

`alt` (Optional) Altitude of orientation. If left empty and a *skyscapeR.horizon* is provided then this is will automatically retrieved from the horizon data via hor2alt

`refraction` (Optional) Whether atmospheric refraction is to be taken into account. If not given the value set by skyscapeR.vars will be used instead.

`atm` (Optional) Atmospheric pressure for refraction calculation. If not given the value set by skyscapeR.vars will be used instead.

`temp` (Optional) Atmospheric temperature for refraction calculation. If not given the value set by skyscapeR.vars will be used instead.

See Also

swe_azalt_rev, hor2alt

Examples

```
dec <- az2dec(az=92, loc=c(35,-8), alt=2)

# flat horizon with 2 degrees of altitude
hor <- createHor(az=c(0,360), alt=c(2,2), loc=c(35,-8,25))
dec <- az2dec(92, loc=hor)
```

```
# Can also be used for an array of azimuths:
decs <- az2dec(az=c(87,92,110), loc=hor)
```

BC.AD *Converts year number (epoch) to calendar year*

Description

Converts year number (epoch) to calendar year

Usage

```
BC.AD(year)
```

Arguments

`year` year number

Examples

```
BC.AD(100)
BC.AD(0)
BC.AD(-1)
BC.AD(-99)
BC.AD(-100)
```

bernoulli.trial *Execute a (series of) Bernoulli trial(s)*

Description

This function allows one to calculate the probability of having r structures out of n, orientated towards a target with probability p.

Usage

```
bernoulli.trial(n, p, r, type = "tail")
```

Arguments

`n` Total number of structures

`p` Probability of target (e.g. ratio of azimuths)

`r` Number of structures orientated towards target (hits)

`type` (Optional) Type of probability to output. Possibilities are: (a) *single* in which case the result of a single Bernoulli trial is reported; or (b) *tail* in which case it calculates Bernoulli trials for all hit values between 1 and (r-1) and then outputs 1 minus the calculated probability. The latter is effectively a p-value. Default is *tail*

References

Ruggles, C (1999) *Astronomy in Prehistoric Britain and Ireland*. Yale University Press.

Examples

```
# probability of having at least 10 out of 30 structures
# aligned to targets covering 20% of the horizon
bernoulli.trial(30, 0.2, 10)
```

`body.position` *Computes position of Solar System bodies in equatorial coordinates*

Description

This function calculates the geocentric or topocentric declination and right ascension of solar system bodies at a given time. It is a wrapper for function `swe_calc_ut` of package *swephR*.

Usage

```
body.position(
  obj = "sun",
  time,
  timezone,
  calendar,
  dec,
  loc = NULL,
  refraction,
  atm,
  temp,
  verbose = T
)
```

Arguments

`obj` (Optional) String containing name of the solar system body of interest. Can be any of the planets (inc. Pluto), the Moon, the Sun or the Ecliptic. Defaults to 'sun'.

`time` Either a string containing the date and time in the format "YYYY/MM/DD HH:MM:SS" (see `timestring`), or a numeric containing the julian date (see `time2jd`).

`timezone` (Optional) Timezone of input either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London"). See timezones for details. Only needed if *time* is a string. #' If not given the value set by `skyscapeR.vars` will be used instead.

`calendar` (Optional) Calendar used in parameter *time*. G for gregorian and J for julian. Only needed if *time* is a string. If not given the value set by `skyscapeR.vars` will be used instead.

`dec`	(Optional) Output declination: *geo* for the geocentric, or *topo* for the topocentric frame of reference. If not given the value set by `skyscapeR.vars` will be used instead.
`loc`	(Optional) Location, only needed if output is in topocentric declination.
`refraction`	(Optional) Whether atmospheric refraction is to be taken into account. If not given the value set by `skyscapeR.vars` will be used instead.
`atm`	(Optional) Atmospheric pressure for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.
`temp`	(Optional) Atmospheric temperature for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.
`verbose`	(Optional) Boolean to control whether or not to display text. Default is TRUE.

See Also

`swe_calc_ut`, `timestring`, `time2jd`

Examples

```
# Position of the sun at noon GMT on Christmas day 2018:
body.position('sun', '2018/12/25 12:00:00', timezone='GMT')

# Declination of the moon at same time
body.position('moon', '2018/12/25 12:00:00', timezone='GMT')$equatorial$Dec
```

coordtrans — *Coordinate-transform azimuth prob distributions into declination prob distributions*

Description

Coordinate-transform azimuth prob distributions into declination prob distributions

Usage

```
coordtrans(pdf, hor, refraction, atm, temp, verbose = T, .res = 0.1)
```

Arguments

`pdf`	A *skyscapeR.pdf* object created with `az.pdf`
`hor`	A *skyscapeR.horizon* object created with `createHor` or `downloadHWT`
`refraction`	(Optional) Whether atmospheric refraction is to be taken into account. If not given the value set by `skyscapeR.vars` will be used instead.
`atm`	(Optional) Atmospheric pressure for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.
`temp`	(Optional) Atmospheric temperature for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.

`verbose`	(Optional) Boolean to control whether or not to display text. Default is TRUE.
`.res`	(Optional) Declination resolution with which to output probability distribution(s). Default is 0.1 degrees.

References

Silva, F (2020) A probabilistic framework and significance test for the analysis of structural orientations in skyscape archaeology *Journal of Archaeological Science* 118, 105138. <doi:10.1016/j.jas.2020.105138>

Examples

```
Az <- az.pdf(az=c(87,93,90,110), unc=3)
hor <- createHor(az=c(0,360), alt=c(0,0), loc=c(35,-8,25)) # flat horizon with 0 degrees of altitude
Dec <- coordtrans(Az, hor)
plot(Dec)
```

`createHor` *Create* skyscapeR.horizon *object from Az/Alt data*

Description

This function creates a *skyscapeR.horizon* object from measurements of azimuth and altitude.

Usage

```
createHor(az, alt, alt.unc = 0.5, loc, name = "", smooth = F, .scale = 1000)
```

Arguments

`az`	Array of azimuth values
`alt`	Array of altitude values.
`alt.unc`	(Optional) Either a single value or an array of altitude uncertainty.
`loc`	Location, a vector containing the latitude, longitude and elevation of the location, in this order.
`name`	Name of site.
`smooth`	Boolean to control whether to smooth horizon profile using rolling mean. Defaults to FALSE
`.scale`	Rolling mean window for smoothing. See `createHWT`

See Also

`plot.skyscapeR.horizon`

`createHWT`, `downloadHWT`

Examples

```
# Create a skyscapeR.horizon from 5 measurements:
az <- c(0,90,180,270,360)
alt <- c(0,5,5,0,0)
hor <- createHor(az, alt, 0.1, c(40.1,-8), 'Test')
plot(hor)
```

---

createHWT *Create and download horizon data from* HeyWhatsThat

Description

This function send a data request to *HeyWhatsThat*, for the creation of a horizon profile for a give Lat/Lon and elevation. It then downloads the data and saves it as a *skyscapeR.horizon* object.

Usage

```
createHWT(lat, lon, elevation = 1.6, name, src = "skyscapeR", verbose = T)
```

Arguments

lat	The latitude of the location.
lon	The longitude of the location.
elevation	(Optional) The elevation of the observer above ground level in meters. Default is 1.6 meters (eye level).
name	(Optional) Name for horizon.
src	(Optional) Request source ID for *HeyWhatsThat*. Default is 'skyscapeR'. Only change this if you have been given a source ID by the creator of *HeyWhatsThat*.
verbose	(Optional) Boolean switch to control output. Default is *TRUE*.

References

HeyWhatsThat.com

See Also

downloadHWT

Examples

```
## Not run:
# Create and retrieve horizon data for the London Mithraeum:
hor <- createHWT(lat=ten(51,30,45), lon=ten(0,5,26.1), name='London Mithraeum')

## End(Not run)
```

curvigram *(Defunct) Computes declination curvigram*

Description

This function no longer works. Please use `spd` of `density` instead.

Usage

```
curvigram()
```

See Also

`spd`

downloadHWT *Download horizon data from* HeyWhatsThat

Description

This function downloads previously created horizon data from *HeyWhatsThat*, given its ID, and saves it as a *skyscapeR.horizon* object.

Usage

```
downloadHWT(HWTID)
```

Arguments

`HWTID`	This is the 8 character ID attributed by *HeyWhatsThat.com*

References

HeyWhatsThat.com

See Also

`createHWT`

Examples

```
## Not run:
# Retrieve horizon data for \href{https://www.heywhatsthat.com/?view=HIFVTBGK}{Liverpool Cathedral}:
hor <- downloadHWT('HIFVTBGK')

## End(Not run)
```

dS *Declination of December Solstice for a given year*

Description

This function calculates the declination of the sun at December Solstice for a given year, based upon obliquity estimation and corrected average parallax.

Usage

```
dS(
  year = skyscapeR.env$cur.year,
  loc = FALSE,
  parallax = 0.00224,
  altitude = 0,
  verbose = TRUE
)
```

Arguments

`year` Year for which to calculate the declination. Defaults to present year as given by *Sys.Date*.

`loc` (Optional) This can be either the latitude of the location, or a *skyscapeR.horizon* object. If missing or *FALSE*, function will output geocentric declination.

`parallax` (Optional) Average parallax value for the sun. Defaults to 0.00224.

`altitude` (Optional) Altitude of the sun. Defaults to 0 degrees.

`verbose` (Optional) Boolean to control output of warnings and messages. Defaults to TRUE.

See Also

`obliquity`, `jS`, `eq`, `zenith`, `antizenith`, `spatial.equinox`, `parallax.corr`

Examples

```
# December Solstice geocentric declination for year 4001 BC:
dS(-4000)

# Topocentric declination for same year and latitude of 50 degrees N:
dS(-4000, loc=50)
```

EFM *Equinoctial Full Moons*

Description

This function calculates the date, rise/set times, azimuth and declination for sun and moon on the days of the Spring Full Moon (SFM) and Autumn Full Moon (AFM), for a given year and location.

Usage

```
EFM(
  season = "spring",
  rise = T,
  year,
  loc,
  min.phase = 0.99,
  refraction,
  atm,
  temp,
  timezone,
  calendar
)
```

Arguments

`season` (Optional) Either 'spring' or 'autumn'. Default is 'spring.

`rise` (Optional) Boolean to choose whether to calculate Equinoctial Full Moon rises or sets. Defaults to *TRUE*.

`year` Epoch(s) for which to do calculations. Can be either a single value (the year), two values (range of years), or a vector of years.

`loc` This can be either a *skyscapeR.horizon* object, or a vector with the latitude, longitude and elevation of the site, in this order.

`min.phase` (Optional) Minimum lunar phase (between 0 and 1) for which a moon is considered to be full. Defaults to 0.99.

`refraction` (Optional) Boolean for whether or not atmospheric refraction should be taken into account. Defaults to *TRUE*.

`atm` (Optional) Atmospheric pressure (in mbar). Only needed if *refraction* is set to *TRUE*. Default is 1013.25 mbar.

`temp` (Optional) Atmospheric temperature (in Celsius). Only needed if *refraction* is set to *TRUE*. Default is 15 degrees.

`timezone` (Optional) Timezone for output of rising and setting time either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London"). See timezones for details. Defaults to system timezone.

`calendar` (Optional) Calendar used to output dates. G for gregorian and J for julian. Defaults to *Gregorian*.

Examples

```
# Spring Full Moon from a location in Portugal in the year 2018
EFM(year=2018, loc=c(35,-8,100))

# Autumn Full Moons in the last three years
## Not run:
EFM(season='autumn', year=c(2019,2021), loc=c(35,-8,100))

## End(Not run)
```

eq *Declination of the sun at the Equinox*

Description

This function calculates the declination of the sun at the astronomical equinox with corrected average parallax.

Usage

```
eq(loc = FALSE, parallax = 0.00224, altitude = 0, verbose = TRUE)
```

Arguments

`loc` (Optional) This can be either the latitude of the location, or a *skyscapeR.horizon* object. If missing or *FALSE*, function will output geocentric declination.

`parallax` (Optional) Average parallax value for the sun. Defaults to 0.00224.

`altitude` (Optional) Altitude of the sun. Defaults to 0 degrees.

`verbose` (Optional) Boolean to control output of warnings and messages. Defaults to TRUE.

See Also

`obliquity`, `jS`, `eq`, `zenith`, `antizenith`, `spatial.equinox`, `parallax.corr`

Examples

```
# Equinoctial geocentric declination:
eq()

# Topocentric declination for same year and latitude of 50 degrees N:
eq(loc=50)
```

exportHor — *Exports a* skyscapeR.horizon *object into* Stellarium *format*

Description

This function exports any *skyscapeR.horizon* object into the landscape format of *Stellarium*, ready to be imported.

Usage

```
exportHor(hor, name, author = "skyscapeR", description, ground_col, hor_col)
```

Arguments

`hor`	Horizon data in *skyscapeR.horizon* format.
`name`	Horizon name to be displayed in *Stellarium*, if different from one in *skyscapeR.horizon* object.
`author`	(Optional) Author, to be included in *landscape.ini* file.
`description`	(Optional) Description, to be included in *landscape.ini* file.
`ground_col`	Color of ground. Defaults to *Stellarium*'s default.
`hor_col`	Color of horizon line. Defaults to *Stellarium*'s default.

References

Stellarium: a free open source planetarium

See Also

`createHor`, `downloadHWT`, `plot.skyscapeR.horizon`

Examples

```
# Downloads horizon data from HeyWhatsThat and exports it into Stellarium:
## Not run:
hor <- downloadHWT('HIFVTBGK')
exportHor(hor, name='Test', description='Test horizon export to Stellarium')

## End(Not run)
```

findTargets *Find celestial targets within declination and time ranges*

Description

Find celestial targets within declination and time ranges

Usage

```
findTargets(
  decrange,
  timerange,
  max.mag = 2.5,
  loc = FALSE,
  calendar = skyscapeR.env$calendar
)
```

Arguments

decrange	Range of declination to consider.
timerange	Temporal range to consider
max.mag	(Optional) Maximum magnitude of stars to consider. Defaults to 2.5
loc	Location, either a *skyscapeR.object* or a vector containing the latitude, longitude and elevation of location, in this order. Defaults to FALSE, thus checking only geocentric declination.
calendar	(Optional) Calendar used in parameter *time*. G for gregorian and J for julian. If not given the value set by skyscapeR.vars will be used instead.

Examples

```
## Not run:
findTargets(c(-25,-17.5), c(-2500,-1750))

# if a location is given then the zenith and anti-zenith sun will also be looked at:
findTargets(c(3,12), c(-2500,-1750), loc=c(8.6, 7.3, 200))

# if a horizon profile is given then the spatial equinox will also be looked at:
hor <- downloadHWT('J657KVEV')
findTargets(c(-7,2), c(-2500,-1750), loc=hor)

## End(Not run)
```

hor2alt *Retrieves horizon altitude for a given azimuth from a given horizon profile*

Description

This function retrieves the horizon altitude for a given azimuth from a previously created *skyscapeR.horizon* object via spline interpolation.

Usage

```
hor2alt(hor, az, return.unc = F)
```

Arguments

`hor` A *skyscapeR.horizon* object from which to retrieve horizon altitude.

`az` Array of azimuth(s) for which to retrieve horizon altitude(s).

`return.unc` (Optional) Boolean switch control where to output altitude uncertainty. Default is *FALSE*.

See Also

createHor, downloadHWT

Examples

```
hor <- downloadHWT('HIFVTBGK')
hor2alt(hor, 90)
```

hpdi *Returns the high-density region of a probability distribution*

Description

Returns the high-density region of a probability distribution

Usage

```
hpdi(x, mass = 0.954)
```

Arguments

`x` A *skyscapeR.spd* or *skyscapeR.pdf* object.

`mass` (Optional) Probability mass of the region. Default is 0.954.

jd2time *Converts Julian date and time (in any timezone) to julian date*

Description

Converts Julian date and time (in any timezone) to julian date

Usage

```
jd2time(jd, timezone, calendar, verbose = F)
```

Arguments

`jd`	Julian date in numeric format
`timezone`	(Optional) Desired timezone for output either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London"). See `timezones` for details. Default is system timezone.
`calendar`	(Optional) Calendar used in parameter *time*. G for gregorian and J for julian. Only needed if *time* is a string. Defaults to *Gregorian*.
`verbose`	(Optional) Controls whether messages should be displayed. Default is *FALSE*.

See Also

`swe_julday`, `as.POSIXlt`, `timezones`

Examples

```
jd <- time2jd('2018/12/25 12:00:00', 'GMT') # Julian date at noon GMT on Christmas day 2018
jd2time(jd, 'CET') # converts julian date to Central European timezone
```

jS *Declination of June Solstice for a given year*

Description

This function calculates the declination of the sun at June Solstice for a given year, based upon obliquity estimation and corrected average parallax.

Usage

```
jS(
  year = skyscapeR.env$cur.year,
  loc = FALSE,
  parallax = 0.00224,
  altitude = 0,
  verbose = TRUE
)
```

Arguments

`year`	Year for which to calculate the declination. Defaults to present year as given by *Sys.Date*.
`loc`	(Optional) This can be either the latitude of the location, or a *skyscapeR.horizon* object. If missing or *FALSE*, function will output geocentric declination.
`parallax`	(Optional) Average parallax value for the sun. Defaults to 0.00224.
`altitude`	(Optional) Altitude of the sun. Defaults to 0 degrees.
`verbose`	(Optional) Boolean to control output of warnings and messages. Defaults to TRUE.

See Also

`obliquity`, `dS`, `eq`, `zenith`, `antizenith`, `spatial.equinox`, `parallax.corr`

Examples

```
# June Solstice geocentric declination for year 4001 BC:
jS(-4000)

# Topocentric declination for same year and latitude of 50 degres N:
jS(-4000, loc=50)
```

`long.date` *Converts day-month in 'MM-DD' format to a more readable format*

Description

Converts day-month in ’MM-DD’ format to a more readable format

Usage

```
long.date(date)
```

Arguments

`date`	date in ’MM-DD’ format

Examples

```
long.date('01-01')
long.date('08-23')
```

mag.dec *Estimates magnetic declination (difference between true and magnetic north) based on IGRF 12th gen model*

Description

This function estimates the magnetic declination at a given location and moment in time, using the *12th generation International Geomagnetic Reference Field (IGRF)* model. This function is a wrapper for function `magneticField` of package *oce*.

Usage

```
mag.dec(loc, date)
```

Arguments

`loc`	Location, can be either a *skyscapeR.horizon* object or, alternatively, a latitude.
`date`	Date for which to calculate magnetic declination in the format: 'YYYY/MM/DD'

See Also

`magneticField`

Examples

```
# Magnetic Declination for London on April 1st 2016:
loc <- c( 51.5074, -0.1278 )
mag.dec( loc, "2016/04/01" )
```

moonphase *Computes the phase of the moon*

Description

This function calculates the moon phase, in percentage of full. It is a wrapper for function `swe_pheno_ut` of package *swephR*.

Usage

```
moonphase(time, timezone, calendar)
```

Arguments

time — Either a string containing the date and time in the format "YYYY-MM-DD HH:MM:SS" (see timestring), or a numeric containing the julian date (see time2jd).

timezone — (Optional) Timezone of input either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London"). See timezones for details. Only needed if *time* is a string. Defaults to system timezone.

calendar — (Optional) Calendar used in parameter *time*. G for gregorian and J for julian. Only needed if *time* is a string. Defaults to Gregorian.

See Also

swe_pheno_ut

Examples

```
# Moonphase at noon GMT on Christmas day 2018:
moonphase('2018/12/25 12:00:00', 'GMT')
```

nMjLX — *Declination of northern major Lunar Extreme for a given year*

Description

This function calculates the declination of the northern major Lunar Extreme for a given year, based upon obliquity estimation and corrected average parallax.

Usage

```
nMjLX(
  year = skyscapeR.env$cur.year,
  loc = FALSE,
  parallax = 0.952,
  altitude = 0,
  verbose = TRUE
)
```

Arguments

year — Year for which to calculate the declination. Defaults to present year as given by *Sys.Date*.

loc — (Optional) This can be either the latitude of the location, or a *skyscapeR.horizon* object. If missing or *FALSE*, function will output geocentric declination.

parallax — (Optional) Average parallax value for the moon Defaults to 0.952.

altitude — (Optional) Altitude of the sun. Defaults to 0 degrees.

verbose — (Optional) Boolean to control output of warnings and messages. Defaults to TRUE.

See Also

nmnLX, smnLX, sMjLX, parallax.corr

Examples

```
# Northern major Lunar Extreme geocentric declination for year 2501 BC:
nMjLX(-2500)

# Topocentric declination for same year and latitude of 50 degrees N:
nMjLX(-2500, loc=50)
```

---

nmnLX *Declination of northern minor Lunar Extreme for a given year*

Description

This function calculates the declination of the northern minor Lunar Extreme for a given year, based upon obliquity estimation and corrected average parallax.

Usage

```
nmnLX(
  year = skyscapeR.env$cur.year,
  loc = FALSE,
  parallax = 0.952,
  altitude = 0,
  verbose = TRUE
)
```

Arguments

year	Year for which to calculate the declination. Defaults to present year as given by *Sys.Date*.
loc	(Optional) This can be either the latitude of the location, or a *skyscapeR.horizon* object. If missing or *FALSE*, function will output geocentric declination.
parallax	(Optional) Average parallax value for the moon Defaults to 0.952.
altitude	(Optional) Altitude of the sun. Defaults to 0 degrees.
verbose	(Optional) Boolean to control output of warnings and messages. Defaults to TRUE.

See Also

smnLX, nMjLX, sMjLX, parallax.corr

Examples

```
# Northern minor Lunar Extreme geocentric declination for year 2501 BC:
nmnLX(-2500)

# Topocentric declination for same year and latitude of 50 degrees N:
nmnLX(-2500, loc=50)
```

---

obliquity — *Computes obliquity of the ecliptic*

Description

This function calculates the obliquity for a given epoch. It is a wrapper for function `swe_calc_ut` of package *swephR*.

Usage

```
obliquity(year = skyscapeR.env$cur.year)
```

Arguments

`year`	Year for which to calculate the obliquity. Defaults to present year as given by *Sys.Date*

References

Laskar, J. et al. (2004), A long-term numerical solution for the insolation quantities of the Earth, *Astron. Astroph.*, 428, 261-285, doi:10.1051/0004-6361:20041335.

See Also

`swe_calc_ut`

Examples

```
#' # Obliquity for year 3999 BC:
obliquity(-4000)
```

orbit *Calculate visible path of celestial object at given location*

Description

This function calculates the visible path of a celestial object from any location on earth. It outputs a *skyscapeR.orbit* object, which includes AZ and ALT information.

Usage

```
orbit(dec, loc, res = 0.25, refraction, atm, temp)
```

Arguments

`dec` Declination of object.

`loc` Location, either a *skyscapeR.object* or a vector containing the latitude and longitude of location, in this order.

`res` The resolution (in degrees of RA) with which to calculate the path.

`refraction` (Optional) Whether atmospheric refraction is to be taken into account. If not given the value set by `skyscapeR.vars` will be used instead.

`atm` (Optional) Atmospheric pressure for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.

`temp` (Optional) Atmospheric temperature for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.

Examples

```
# Visible path of sun on june solstice on year 3999 BC from London:
sun.dec <- jS(-4000)
london.lat <- 51.5074 #N
london.lon <- -0.1278 #W
loc <- c( london.lat, london.lon, 0 )
path <- orbit(sun.dec, loc)
plot(path$az, path$alt, ylim=c(0,90), type='l', xlab='AZ', ylab='ALT', col='red', lwd=2)
```

parallax.corr *Corrected parallax for a given location and object altitude*

Description

Given the average parallax, this function corrects this value for a given latitude of the observer and for the altitude of the celestial object.

Usage

```
parallax.corr(parallax, loc, altitude = 0)
```

Arguments

`parallax` Average parallax to correct (e.g. 0.00224 for the Sun, or 0.952 for the Moon)

`loc` This can be either the latitude of the location, or a *skyscapeR.horizon* object.

`altitude` (Optional) Altitude of the celestial object.. Defaults to 0 degrees.

Examples

```
# Parallax correction for the moon, as seen from latitude 50ºN and at 0º altitude
parallax.corr(0.952, 50, 0)
```

`plot.skyscapeR.horizon` *Plot horizon data*

Description

This function creates a plot of horizon data.

Usage

```
## S3 method for class 'skyscapeR.horizon'
plot(
  x,
  show.az = F,
  xlim,
  ylim,
  obj,
  refraction = F,
  col.ground = "#fdae61",
  ...
)
```

Arguments

`x` Object of *skyscapeR.horizon* format.

`show.az` (Optional) Boolean that controls whether to display azimuth values or cardinal directions. Defaults to FALSE.

`xlim` (Optional) Azimuth rage for plotting.

`ylim` (Optional) Altitude rage for plotting.

`obj` (Optional) A *skyscapeR.object* object created with `sky.objects` for displaying the paths of celestial objects.

refraction	(Optional) Boolean switch controlling whether to take refraction into account when displaying the paths of celestial objects.
col.ground	(Optional) Color of the ground. Defaults to *#fdae61*.
...	Additional arguments to be passed to *plot*.

See Also

downloadHWT, sky.objects

Examples

```
# Plot a horizon retrieved from HeyWhatsThat:
hor <- downloadHWT('HIFVTBGK')
plot(hor)

# Add the paths of the solstices and equinoxes sun in the year 1999 BC:
tt <- sky.objects('solar extremes', epoch=-2000, col='blue')
plot(hor, obj=tt)
```

plot.skyscapeR.pdf *Plot orientation probability distributions*

Description

Plot orientation probability distributions

Usage

```
## S3 method for class 'skyscapeR.pdf'
plot(
  x,
  index,
  hdr = 0.954,
  show.az = T,
  xlim,
  col = MESS::col.alpha("blue", 0.5),
  ...
)
```

Arguments

x	A *skyscapeR.pdf* object created with either az.pdf or coordtrans
index	(Optional) A value to indicate which distribution to plot, when only one is desired.
hdr	(Optional) High density region to highlight. Defaults to 0.954
show.az	(Optional) Whether to show the azimuth and transformation curve (horizon) when displaying declination distributions. Defaults to TRUE.

xlim (Optional) Range of x-axis to plot.

col (Optional) Color of high density region to highlight. Defaults to blue with 0.5 alpha.

... Additional arguments to be passed to *plot*.

---

plot.skyscapeR.sigTest *Plot significance test of orientations*

Description

Plot significance test of orientations

Usage

```
## S3 method for class 'skyscapeR.sigTest'
plot(
  x,
  xlim,
  title = NULL,
  show.pval = T,
  show.local = F,
  pal = brewer.pal(5, ”PRGn”)[c(1, 5)],
  ...
)
```

Arguments

x A *skyscapeR.sigTest* object created with randomTest

xlim (Optional) Range of x-axis to plot.

title (Optional) Title to add to the plot.

show.pval (Optional) Boolean to control whether to print the global p-value. Default is TRUE.

show.local (Optional) Boolean to control whether to show local regions of significance. Default is FALSE

pal (Optional) Color palette for local regions of significance. Default is brewer.pal(5, ’PRGn’)

... Additional arguments to be passed to *plot*.

plot.skyscapeR.spd *Plot orientation summed probability density*

Description

Plot orientation summed probability density

Usage

```
## S3 method for class 'skyscapeR.spd'
plot(x, xlim, ylim, title = NULL, col = "blue", shading = T, ...)
```

Arguments

`x`	A *skyscapeR.spd* object created with `spd`
`xlim`	(Optional) Range of x-axis to plot.
`ylim`	(Optional) Range of y-axis to plot.
`title`	(Optional) Title to add to the plot.
`col`	(Optional) Color of summed probability density. Defaults to blue with 0.5 alpha.
`shading`	(Optional) Boolean to control whether to color the entire distribution. Defaults to TRUE
`...`	Additional arguments to be passed to *plot*.

plot.skyscapeR.starphases *Plot stellar phase and seasonality*

Description

This function creates a plot of stellar seasonality and phases/events.

Usage

```
## S3 method for class 'skyscapeR.starphases'
plot(x, ...)
```

Arguments

`x`	Object of *skyscapeR.starphases* format.
`...`	Additional arguments to be passed to *plot*.

See Also

`star.phases`

Examples

```
# Plot the seasonality of Aldebaran for 3999 BCE:
## Not run:
ss <- star.phases('Aldebaran',-4000, c(35,-8, 200))
plot(ss)

## End(Not run)
```

`plotAzimuth` *Polar plot of orientations (azimuths)*

Description

Polar plot of orientations (azimuths)

Usage

```
plotAzimuth(az, col = "blue", lwd = 1.5, lty = 1, obj, show.obj.labels = T)
```

Arguments

`az` (Optional) Array of azimuths. Can be omitted is *obj* is given.

`col` (Optional) Single color or color palette to use for plotting measurements.

`lwd` (Optional) Line width to plot measurements. Defaults to 1.

`lty` (Optional) Line type to plot measurements. Defaults to 1.

`obj` (Optional) A *skyscapeR.object* object created with `sky.objects` for displaying the azimuths of celestial objects. Note that this assumes a single location and a flat horizon of zero degrees.

`show.obj.labels` (Optional) Boolean to control whether to display celestial objects names. Defaults to TRUE.

Examples

```
# Plot some azimuth data:
az <- c(120, 100, 93, 97, 88, 115, 112, 67)
plotAzimuth(az)

# To visualize this data against the common solar and lunar targets:
tt <- sky.objects('solar extremes', epoch=-2000, loc=c(35,-8), col='red')
plotAzimuth(az, obj=tt)

# To display only celestial objects
plotAzimuth(obj=tt)
```

plotBars *Bar plot of orientations (declination)*

Description

Bar plot of orientations (declination)

Usage

```
plotBars(
  val,
  unc,
  names,
  unit = "Declination",
  col = "blue",
  shade = TRUE,
  mark = TRUE,
  sort = FALSE,
  xrange,
  yrange,
  obj,
  show.obj.label = TRUE
)
```

Arguments

`val` Array of declination or azimuth values.

`unc` Single value or array of measurement uncertainty

`names` (Optional) Array of names of measurements in *val*

`unit` (Optional). Either ’Declination’ or ’Azimuth’. Defaults to ’Declination’.

`col` (Optional) Color to plot measurements in. Defaults to *blue*.

`shade` (Optional) Boolean to control whether to shade a polygon of measurements. Defaults to *TRUE*

`mark` (Optional) Boolean to control whether to mark the declination value. Defaults to *TRUE*

`sort` (Optional) Boolean to control whether to sort the measurements by their declination value. Defaults to *FALSE*

`xrange` (Optional) Array of limits for x-axis.

`yrange` (Optional) Array of limits for y-axis.

`obj` (Optional) A *skyscapeR.object* object created with `sky.objects` for displaying the declination values of celestial objects.

`show.obj.label` (Optional) Boolean to control whether to label the celestial objects in the polar plot. Defaults to *TRUE*.

See Also

sky.objects

Examples

```
# Plot some declination data:
decs <- c(10, 12, -5, 4)
plotBars(decs, unc=5)

# To visualize this data against the common solar and lunar targets:
tt <- sky.objects(c('solar extremes','lunar extremes'), epoch=-2000, lty=c(2,3))
plotBars(decs, unc=5, obj=tt)
```

print.skyscapeR.sigTest *Prints significance test results*

Description

This function prints the results of randomTest.

Usage

```
## S3 method for class 'skyscapeR.sigTest'
print(x, ...)
```

Arguments

x	Object of *skyscapeR.sigTest* format.
...	Additional arguments to be passed to *print*.

See Also

randomTest

pval2stars *Converts p-value into symbol*

Description

Converts p-value into symbol

Usage

```
pval2stars(p.value)
```

Arguments

`p.value`	p-value

`randomTest` *Significance test against the null hypothesis of random orientation*

Description

Significance test against the null hypothesis of random orientation

Usage

```
randomTest(
  pdf,
  nsims = 1000,
  conf = 0.95,
  tails = 2,
  normalise = F,
  ncores = parallel::detectCores() - 1,
  save.sim = F,
  verbose = T
)
```

Arguments

`pdf`	A *skyscapeR.pdf* object created with either `az.pdf` or `coordtrans`
`nsims`	(Optional) Boolean switch controlling whether to normalize the SPD. Default is FALSE
`conf`	(Optional) Array of values (min and max) for SPD if different from range of *pdf*
`tails`	(Optional) Whether to calculate 1-tailed p-value (greater than) or 2-tailed p-value (smaller than or greater than). Default is 2.
`normalise`	(Optional) Boolean to control whether to normalize SPDs. Default is FALSE
`ncores`	(Optional) Number of CPU cores to use. Default is the number of available cores minus 1.
`save.sim`	(Optional) Boolean to control whether to save the output of each random simulation. For testing/advanced use only. Default is FALSE
`verbose`	(Optional) Boolean to control whether or not to display text. Default is TRUE.

References

Silva, F (2020) A probabilistic framework and significance test for the analysis of structural orientations in skyscape archaeology *Journal of Archaeological Science* 118, 105138. <doi:10.1016/j.jas.2020.105138>

Examples

```
## Not run:
# significance test for azimuth
Az <- az.pdf(az=c(87,93,90,110), unc=3)
st1 <- randomTest(Az, nsims=1000)
plot(st1)

# significance test for declination
hor <- createHor(az=c(0,360), alt=c(0,0), loc=c(35,-8,25)) # flat horizon with 0 degrees of altitude
Dec <- coordtrans(Az, hor)
st2 <- randomTest(Dec, nsims=1000)
plot(st2)

## End(Not run)
```

`reduct.compass` *Data reduction for compass measurements*

Description

This function calculates the true azimuth of a structure measured with a compass.

Usage

```
reduct.compass(loc, mag.az, date, magdec, alt, name, ID, HWT.ID)
```

Arguments

`loc` Location, either a *skyscapeR.object* or a vector containing the latitude, longitude and elevation of location, in this order.

`mag.az` Array of magnetic azimuth measurements.

`date` (Optional) Date of measurements as a string in the format: 'YYYY/MM/DD'. Only necessary if *magdec* is not given.

`magdec` (Optional) Magnetic declination, if known.

`alt` (Optional) Altitude, necessary for automatic declination calculation. If missing and *loc* is a *skyscapeR.horizon* object then the altitude will be automatically read from the horizon profile.

`name` (Optional) Names or labels to identify each measurement.

`ID` (Optional) IDs or codes to identify each measurement.

`HWT.ID` (Optional) HeyWhatsThat IDs relating to a previously generated horizon profile for measurement.

See Also

`mag.dec`, `az2dec`, `hor2alt`

Examples

```
loc <- c(35,-7, 100)
mag.az <- c(89.5, 105, 109.5)
data <- reduct.compass(loc, mag.az, "2016/04/02")

# Declination will be automatically calculated if the altitude is also given:
data <- reduct.compass(loc, mag.az, "2016/04/02", alt=c(1,2,0))

# Alternatively, the altitude can be automatically retrieved from a horizon profile:
hor <- downloadHWT('HIFVTBGK')
data <- reduct.compass(hor, mag.az, "2016/04/02")
```

reduct.theodolite *Data reduction for theodolite measurements using the sun-sight method*

Description

This function calculates the true azimuth of a structure measured with a theodolite using the sun-sight technique.

Usage

```
reduct.theodolite(
  loc,
  az,
  date,
  time,
  tz,
  az.sun = 0,
  limb,
  alt,
  name,
  ID,
  HWT.ID
)
```

Arguments

`loc` Location, either a *skyscapeR.object* or a vector containing the latitude, longitude and elevation of location, in this order.

`az` Array of azimuths. Use `ten` to convert to decimal point format if necessary.

`date` Date of measurements as a string in the format: 'YYYY/MM/DD'

`time` Time of sun-sight measurement in the format: 'HH:MM:SS'

`tz` Timezone of input wither as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London").

`az.sun` (Optional) Measured azimuth of the sun. Defaults to zero.

`limb` (Optional) Measured limb of the sun. Options are *left*, *right*. If missing the center of the sun will be used for calculations.

`alt` (Optional) Altitude, necessary for automatic declination calculation. If missing and *loc* is a *skyscapeR.horizon* object then the altitude will be automatically read from the horizon profile.

`name` (Optional) Names or labels to identify each measurement.

`ID` (Optional) IDs or codes to identify each measurement.

`HWT.ID` (Optional) HeyWhatsThat IDs relating to a previously generated horizon profile for measurement.

References

Ruggles, C.L.N. (1999). *Astronomy in Prehistoric Britain and Ireland.* Yale University Press.

See Also

sunAz, ten

Examples

```
lat <- ten(35,50,37.8)
lon <- ten(14,34,6.4)
elev <- 100
az <- c( ten(298,24,10), ten(302,20,40))
az.sun <- ten(327,29,50)
date <- "2016/02/20"
time <- "11:07:17"

data <- reduct.theodolite(c(lat,lon,elev), az, date , time, tz= "Europe/Malta", az.sun)

# Declination will be automatically calculated if the altitude is also given:
data <- reduct.theodolite(c(lat,lon,elev), az, date , time, tz= "Europe/Malta", az.sun, alt=c(2,5))

# Alternatively, the altitude can be automatically retrieved from a horizon profile:
hor <- downloadHWT('HIFVTBGK')
data <- reduct.theodolite(hor, az, date, time, tz= "Europe/Malta", az.sun)
```

riseset *Computes the rising and setting azimuth, declination and time of a Solar System object for a given location and day*

Description

Computes the rising and setting azimuth, declination and time of a Solar System object for a given location and day

Usage

```
riseset(
  obj = "sun",
  date,
  jd,
  alt = 0,
  loc,
  calendar,
  timezone,
  dec,
  refraction,
  atm,
  temp,
  verbose = T
)
```

Arguments

`obj`	(Optional) String containing name of the solar system body of interest. Can be any of the planets (inc. Pluto), the Moon, the Sun or the Ecliptic. Defaults to 'sun'.
`date`	Either a string containing the date in the format "YYYY/MM/DD" (see `timestring`), or a numeric containing the julian date (see `time2jd`). Can also be a single year ("YYYY") or a month ("YYYY/MM") to calculate risings and settings for every day in the year or month, respectively. Not necessary if *jd* is given.
`jd`	(Optional) A numeric containing the julian date (see `time2jd`) for which to calculate rising and settings. Only needed if *date* is not given.
`alt`	(Optional) The altitude of the horizon to consider. Defaults to zero degrees.
`loc`	Location, either a *skyscapeR.object* or a vector containing the latitude, longitude and elevation of location, in this order.
`calendar`	(Optional) Calendar used in parameter *date*. G for gregorian and J for julian. If not given the value set by `skyscapeR.vars` will be used instead.
`timezone`	(Optional) Timezone for output of rising and setting time either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London"). See timezones for details. If not given the value set by `skyscapeR.vars` will be used instead.
`dec`	(Optional) Output declination: *geo* for the geocentric, or *topo* for the topocentric frame of reference. If not given the value set by `skyscapeR.vars` will be used instead.
`refraction`	(Optional) Whether atmospheric refraction is to be taken into account. If not given the value set by `skyscapeR.vars` will be used instead.
`atm`	(Optional) Atmospheric pressure for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.
`temp`	(Optional) Atmospheric temperature for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.#'
`verbose`	(Optional) Boolean to control whether or not to display text. Default is TRUE.

See Also

swe_calc_ut, swe_azalt

Examples

```
# Rising and setting of the sun on june solstice 2018, from the location of London
riseset('sun', '2018/06/21', loc=c(51.5, 0.11, 100))

# Rising ans setting of the moon on june solstice 2018, using a horizon profile
hor <- downloadHWT('HIFVTBGK') # Liverpool cathedral
riseset('moon', '2018/06/21', loc=hor)

# Rising and setting of the sun throughout February 1999, from the location of London
riseset('sun', '1999/02', loc=c(51.5, 0.11, 100))

## Not run:
# Rising and setting of the sun throughout 3001 BC, from the location of London
riseset('sun', -3000, loc=c(51.5, 0.11, 100))

## End(Not run)
```

RugglesCKR *Cork and Kerry Stone Row Data*

Description

Data from C.L.N. Ruggles' fieldwork on the Stone Rows of Cork and Kerry.

Usage

```
data(RugglesCKR)
```

Format

A data frame with 41 rows and 5 variables:

Ref Site Ref

NE.SW String indicating whether they are towards the NE or SW

Az.Hill Azimuth of hill towards which the Stone Rows are pointing

Alt.Hill Altitude of hill towards which the Stone Rows are pointing

Dec.Hill Declination of hill towards which the Stone Rows are pointing

References

Ruggles, C.L.N. (1999). *Astronomy in Prehistoric Britain and Ireland.* Yale University Press.

Examples

```
data(RugglesCKR)
kde <- density(RugglesCKR$Dec.Hill, 2)
plot(kde)
```

RugglesRSC *Recumbent Stone Circle Data*

Description

Declination data from C.L.N. Ruggles' fieldwork on the Scottish Recumbent Stone Circles.

Usage

```
data(RugglesRSC)
```

Format

A data frame with 37 rows and 2 variables:

Dec Declination

ID Site ID

References

Ruggles, C.L.N. (1999). *Astronomy in Prehistoric Britain and Ireland.* Yale University Press.

Examples

```
data(RugglesRSC)
kde <- density(RugglesRSC$Dec, 2)
plot(kde)
```

---

sigTest *(Defunct) Perform a null hypothesis significance test of a given curvigram*

Description

This function no longer works. Please use `randomTest` instead.

Usage

```
sigTest()
```

See Also

`randomTest`

sky.objects — *Creates a* skyscapeR.object *for plotting of celestial objects at given epoch*

Description

This function creates an object containing all the necessary information to plot celestial objects/events unto the many plotting functions of *skyscapeR* package.

Usage

```
sky.objects(names, epoch, loc = FALSE, col = "red", lty = 1, lwd = 1)
```

Arguments

names	The name(s) of the celestial object(s) or event(s) of interest. These can be one of the following soli-lunar events: *jS*, *dS*, *eq*, *nmnLX*, *nMjLX*, *smnLX*, *sMjLX*, or the name of any star in the database. As shorthand, the names *sun* and *moon* can be used to represent all the above solar and lunar events, respectively. Alternatively, custom declination values can also be used.
epoch	The year or year range (as an array) one is interested in.
loc	(Optional) This can be either a vector with the latitude and longitude of the location, or a *skyscapeR.horizon* object.
col	(Optional) The color for plotting, and differentiating these objects. Defaults to red for all objects.
lty	(Optional) Line type (see par) used for differentiation. Only activated for single year epochs.
lwd	(Optional) Line width (see par) used for differentiation. Only activated for single year epochs.

Examples

```
# Create a object with solar targets for epoch range 4000-2000 BC:
tt <- sky.objects('solar extremes', c(-4000,-2000))

# Create an object with a few stars for same epoch:
tt <- sky.objects(c('Sirius', 'Betelgeuse', 'Antares'), c(-4000,-2000),
col=c('white', 'red', 'orange'))

# Create an object with solstices and a custom declination value:
tt <- sky.objects(c('December Solstice','June Solstice', -13), c(-4000,-2000))
```

sky.sketch *Create a simplistic sketch of the sky at a given moment in time*

Description

Create a simplistic sketch of the sky at a given moment in time

Usage

```
sky.sketch(
  time,
  timezone,
  calendar,
  xrange = c(30, 150),
  yrange = c(-45, 45),
  sun = T,
  moon = T,
  planets = F,
  exagerate = T,
  max.mag = 6,
  loc,
  atm,
  temp
)
```

Arguments

time	Either a string containing the date and time in the format "YYYY-MM-DD HH:MM:SS" (see timestring), or a numeric containing the julian date (see time2jd).
timezone	(Optional) Timezone of input either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London"). See timezones for details. Only needed if *time* is a string. If not given the value set by skyscapeR.vars will be used instead.
calendar	(Optional) Calendar used in parameter *time*. G for gregorian and J for julian. Only needed if *time* is a string. If not given the value set by skyscapeR.vars will be used instead.
xrange	Range of azimuths to display, preferably no larger than 120 degrees.
yrange	Range of altitudes to display, preferably no larger than 120 degrees.
sun	(Optional) Boolean on whether the sun should be displayed. Defaults to *TRUE*.
moon	(Optional) Boolean on whether the moon should be displayed. Defaults to *TRUE*.
planets	(Optional) Boolean on whether the visible planets should be displayed. Defaults to *FALSE*.

exagerate (Optional) Boolean on whether the size of the sun, moon and planets should be exaggerated, which can be useful when attempting wider viewing angles. Defaults to *TRUE*.

max.mag (Optional) Maximum magnitude of stars to consider. Default is 6.

loc Location, either a *skyscapeR.object* or a vector containing the latitude and longitude of location, in this order.

atm (Optional) Atmospheric pressure for refraction calculation. If not given the value set by skyscapeR.vars will be used instead.

temp (Optional) Atmospheric temperature for refraction calculation. If not given the value set by skyscapeR.vars will be used instead.

Examples

```
sky.sketch(time='2019/01/10 18:51', loc=c(35,-8,100))
```

skyscapeR.vars *See and change the global variables used by skyscapeR*

Description

See and change the global variables used by skyscapeR

Usage

```
skyscapeR.vars(timezone, calendar, refraction, atm, temp, dec)
```

Arguments

timezone Timezone of input either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London"). See timezones for details. Default is the system timezone

calendar Calendar used in parameter *time*. G for gregorian and J for julian. Defaults to *Gregorian*.

refraction Whether atmospheric refraction is to be taken into account. Default is TRUE.

atm Atmospheric pressure for refraction calculation. Default is 1013.25 mbar.

temp Atmospheric temperature for refraction calculation. Default is 15 degrees.

dec Output declination: *geo* for the geocentric, or *topo* for the topocentric frame of reference. Defaults to topocentric.

Examples

```
# Julian date at noon GMT on Christmas day 2018
time2jd('2018-12-25 12:00:00', 'GMT')
```

sMjLX *Declination of southern major Lunar Extreme for a given year*

Description

This function calculates the declination of the southern major Lunar Extreme for a given year, based upon obliquity estimation and corrected average parallax.

Usage

```
sMjLX(
  year = skyscapeR.env$cur.year,
  loc = FALSE,
  parallax = 0.952,
  altitude = 0,
  verbose = TRUE
)
```

Arguments

year	Year for which to calculate the declination. Defaults to present year as given by *Sys.Date*.
loc	(Optional) This can be either the latitude of the location, or a *skyscapeR.horizon* object. If missing or *FALSE*, function will output geocentric declination.
parallax	(Optional) Average parallax value for the moon Defaults to 0.952.
altitude	(Optional) Altitude of the sun. Defaults to 0 degrees.
verbose	(Optional) Boolean to control output of warnings and messages. Defaults to TRUE.

See Also

nmnLX, smnLX, nMjLX, parallax.corr

Examples

```
# Southern major Lunar Extreme geocentric declination for year 2501 BC:
sMjLX(-2500)

# Topocentric declination for same year and latitude of 50 degrees N:
sMjLX(-2500, loc=50)
```

smnLX *Declination of southern minor Lunar Extreme for a given year*

Description

This function calculates the declination of the southern minor Lunar Extreme for a given year, based upon obliquity estimation and corrected average parallax.

Usage

```
smnLX(
  year = skyscapeR.env$cur.year,
  loc = FALSE,
  parallax = 0.952,
  altitude = 0,
  verbose = TRUE
)
```

Arguments

year	Year for which to calculate the declination. Defaults to present year as given by *Sys.Date*.
loc	(Optional) This can be either the latitude of the location, or a *skyscapeR.horizon* object. If missing or *FALSE*, function will output geocentric declination.
parallax	(Optional) Average parallax value for the moon Defaults to 0.952.
altitude	(Optional) Altitude of the sun. Defaults to 0 degrees.
verbose	(Optional) Boolean to control output of warnings and messages. Defaults to TRUE.

See Also

nmnLX, nMjLX, sMjLX, parallax.corr

Examples

```
# Southern minor Lunar Extreme geocentric declination for year 2501 BC:
smnLX(-2500)

# Topocentric declination for same year and latitude of 50 degrees N:
smnLX(-2500, loc=50)
```

solar.date *Solar Date*

Description

Returns the calendar date when the sun has the same declination as the input declination.

Usage

```
solar.date(dec, year, calendar, verbose = T)
```

Arguments

`dec`	Single value or array of declination values.
`year`	Year for which to do calculations.
`calendar`	(Optional) Calendar used for output. G for gregorian and J for julian. Defaults to Gregorian.
`verbose`	(Optional) Boolean to control whether or not to display text. Default is TRUE.

Examples

```
solar.date(-23, 2018)
solar.date(-12, 1200, calendar='G')
solar.date(-12, 1200, calendar='J')
solar.date(14, -2000)
```

spatial.equinox *Declination of the spatial equinox for a given location*

Description

Declination of the spatial equinox for a given location

Usage

```
spatial.equinox(hor, parallax = 0.00224)
```

Arguments

`hor`	This should be a *skyscapeR.horizon* object.
`parallax`	(Optional) Average parallax value for the sun. Defaults to 0.00224.

See Also

`jS`, `dS`, `eq`, `zenith`, `antizenith`, `parallax.corr`

Examples

```
## Not run:
hor <- createHWT(34.174051531543405, 110.81299818694872, name='Xipo, Lingbao')
spatial.equinox(hor)

## End(Not run)
```

---

spd *Summed probability density (SPD)*

Description

Summed probability density (SPD)

Usage

```
spd(pdf, normalise = F, xrange, .cutoff = 1e-05, .res = 0.01)
```

Arguments

`pdf`	A *skyscapeR.pdf* object created with either `az.pdf` or `coordtrans`
`normalise`	(Optional) Boolean to control whether to normalize the SPD. Default is FALSE
`xrange`	(Optional) Array of values (min and max) for SPD if different from range of *pdf*
`.cutoff`	(Optional) Value of SPD at which point it will be cutoff to save on memory. Default is 1e-5
`.res`	(Optional) Resolution with which to output SPD. Default is 0.01 degrees.

References

Silva, F (2020) A probabilistic framework and significance test for the analysis of structural orientations in skyscape archaeology *Journal of Archaeological Science* 118, 105138. <doi:10.1016/j.jas.2020.105138>

Examples

```
# SPD of azimuths
Az <- az.pdf(az=c(87,93,90,110), unc=3)
s1 <- spd(Az)
plot(s1)

# SPD of declinations
hor <- createHor(az=c(0,360), alt=c(0,0), loc=c(35,-8,25)) # flat horizon with 0 degrees of altitude
Dec <- coordtrans(Az, hor)
s2 <- spd(Dec)
plot(s2)
```

star *Create* skyscapeR.star *object*

Description

This function retrieves information for a given star and saves it in the *skyscapeR.star* format ready to be used by other skyscapeR package function.

Usage

```
star(string, year = skyscapeR.env$cur.year)
```

Arguments

`string`	This can be either the traditional name for the star or its Bayer designation.
`year`	Year for which to calculate the coordinates. Defaults to current year.

See Also

`swe_fixstar2_ut`, `swe_fixstar2_mag`

Examples

```
# Retrieve data for Aldebaran:
Aldeb <- star('Aldebaran')

# Retrieve data for Aldebaran on 2999 BC:
ss <- star('Aldebaran', -3000)
```

star.phases *Calculate the seasons and phase type of a star*

Description

This function calculates the seasons (Rising, Setting, etc.) and phase types (Arising and Lying Hidden, Curtailed Passage) of a star for a given location and epoch. This functions uses the *arcus visionis* approximation of Purrington (1988) and the atmospheric extinction approximation of Schaefer (1989). For the nomenclature used, and description of star phase types, see Brady (2015).

Usage

```
star.phases(
  star,
  year,
  loc,
  alt.hor = 0,
  k = 0.2,
  limit = 6,
  alt.rs = 10,
  res = 1/24/6,
  refraction,
  atm,
  temp
)
```

Arguments

`star` Either the star name or a *skyscapeR.star* object.

`year` The year of interest. Must be in the *swephR* range of 13201 cal BC to 17191 AD

`loc` Location, either a *skyscapeR.object* or a vector containing the latitude and longitude of location, in this order.

`alt.hor` (Optional) The altitude of the horizon to consider. Defaults to zero degrees.

`k` (Optional) Extinction coefficient (see Schaefer 1989). Defaults to 0.2, corresponding to a poor night on mountain top or best night at a dry sea level site.

`limit` (Optional) The maximum magnitude of a star that can be visible with the naked eye. Defaults to 6.

`alt.rs` (Optional) The maximum altitude of a star's first or last visibility for it to still be considered to be as rising or setting. Defaults to ten degrees.

`res` (Optional) Resolution of calculation. The smaller this figure the slower the computation. Defaults to 1/24/6, i.e. every 10 minutes.

`refraction` (Optional) Whether atmospheric refraction is to be taken into account. If not given the value set by `skyscapeR.vars` will be used instead.

`atm` (Optional) Atmospheric pressure for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.

`temp` (Optional) Atmospheric temperature for refraction calculation. If not given the value set by `skyscapeR.vars` will be used instead.

References

Purrington, Robert D. (1988) Heliacal Rising and Setting: Quantitative Aspects, *Journal for the History of Astronomy (Archaeoastronomy Supplement 12)* 19, S72-S84. Available online at [SAO/NASA ADS Astronomy Abstract Service](http://adsabs.harvard.edu/abs/1988JHAS...19...72P)

Brady, Bernadette (2015) Star Phases: the Naked-eye Astronomy of the Old Kingdom Pyramid Texts. In F Silva and N Campion (eds) *Skyscapes: The Role and Importance of the Sky in Archaeology*. Oxford: Oxbow Books, pp. 76-86.

See Also

plot.skyscapeR.starphases

Examples

```
## Not run:
ss1 <- star.phases('Aldebaran',-4000, c(35,-8,200))

# One can then look at the star's phase type:
ss1$metadata$type

# Date range of seasons:
ss1$metadata$seasons

# Date range of phase-type events:
ss1$metadata$events

# And plot them:
plot(ss1)

# You can play with the parameters and see how predictions change:
ss1 <- star.phases('Aldebaran',-4000, c(35,-8,200), alt.hor=2, alt.rs=5)
plot(ss1)

## End(Not run)
```

sunAz *Returns the azimuth of the sun at a given time from a specific location*

Description

This function returns the azimuth of the sun at a given time and location, useful for data reduction of theodolite measurements using the sun-sight technique (reduct.theodolite).

Usage

```
sunAz(loc, time, timezone, limb, alt = F)
```

Arguments

`loc` Location, either a *skyscapeR.object* or a vector containing the latitude, longitude and elevation of location, in this order.

`time` String containing the date and time in the following format: "YYYY-MM-DD HH:MM:SS"

`timezone` Timezone of input either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London").

`limb` (Optional) Measured limb of the sun. Options are *left*, *right*. If missing the center of the sun will be output.

alt	(Optional) Boolean that triggers output of altitude of the sun at exact time. Default is FALSE.

See Also

reduct.theodolite

Examples

```
sunAz(c(52,-3,100), '2017-10-04 12:32:14', 'Europe/London')
```

ten *Converts degree measurements in deg-min-sec (º ’ ") format into decimal-point degree format.*

Description

Converts degree measurements in deg-min-sec (º ’ ") format into decimal-point degree format.

Usage

```
ten(dd, mm = 0, ss = 0)
```

Arguments

dd	Degree
mm	(Optional) Arcminutes
ss	(Optional) Arcseconds

Examples

```
deg <- ten(24, 52, 16)
```

time2jd *Converts date and time (in any timezone) to Julian date*

Description

Converts date and time (in any timezone) to Julian date

Usage

```
time2jd(time, timezone, calendar, verbose = F)
```

Arguments

`time` String containing the date and time in the format "YYYY/MM/DD HH:MM:SS". BCE dates should use negative sign. Use `timestring` if needed.

`timezone` (Optional) Timezone of input either as a known acronym (e.g. "GMT", "CET") or a string with continent followed by country capital (e.g. "Europe/London"). See `timezones` for details. Default is the system timezone

`calendar` (Optional) Calendar used in parameter *time*. G for gregorian and J for julian. Defaults to *Gregorian*.

`verbose` (Optional) Controls whether messages should be displayed. Default is *FALSE*.

See Also

`swe_julday`, `as.POSIXlt`, `timezones`, `timestring`

Examples

```
# Julian date at noon GMT on Christmas day 2018
time2jd('2018/12/25 12:00:00', 'GMT')
```

`timestring` *Converts date and time numeric values to a single string*

Description

Converts date and time numeric values to a single string

Usage

```
timestring(year, month, day, hour = 12, minute = 0, second = 0)
```

Arguments

`year`	Year
`month`	Month
`day`	Day
`hour`	Hour
`minute`	Minute
`second`	Second

Examples

```
timestring(2018, 12, 25, 2, 34)
```

zenith *Declination of the zenith sun for a given location*

Description

This function returns the declination of the sun when it is at the zenith for a given location with corrected average parallax. If this phenomena does not occur at given location (i.e. if location is outside the tropical band) the function returns a *NULL* value.

Usage

```
zenith(loc, parallax = 0.00224, altitude = 0)
```

Arguments

`loc`	This can be either the latitude of the location, or a *skyscapeR.horizon* object.
`parallax`	(Optional) Average parallax value for the sun. Defaults to 0.00224.
`altitude`	(Optional) Altitude of the sun. Defaults to 0 degrees.

See Also

`jS`, `dS`, `eq`, `antizenith`, `spatial.equinox`, `parallax.corr`

Examples

```
# Zenith sun declination for Mexico City:
zenith(19.419)

# There is no zenith sun phenomenon in London:
zenith(51.507)
```

Index