

Package ‘moonsun’

February 14, 2012

Type Package

Title Basic astronomical calculations with R

Version 0.1.2

Date 2010-09-08

Author Lukasz Komsta

Maintainer Lukasz Komsta <luke@novum.am.lublin.pl>

Description A collection of basic astronomical routines for R based on
“Practical astronomy with your calculator” by Peter Duffet-Smith.

License GPL-2

Repository CRAN

Date/Publication 2010-09-23 06:14:27

R topics documented:

moonsun-package	2
angle	2
as.ecc	3
as.gmt	4
bright	5
constel	6
ecc	7
format.dms	8
format.jd	8
format.time	9
gmt	9
jd	10
moon	11
planet	12
planets	13

plot.apos	14
print.eqc	14
rst	15
starcats	16
sun	17
track	18

Index	21
--------------	-----------

moonsun-package	<i>Basic astronomical routines with R</i>
-----------------	---

Description

A collection of basic astronomical routines for R - coordinates conversion, ephemerides, rise and set computing, prediction of angles between objects and so on.

Details

Package: moonsun
 Type: Package
 Version: 0.1.1
 Date: 2008-09-03
 License: GPL2

Author(s)

Maintainer: Lukasz Komsta <luke@novum.am.lublin.pl>

References

Duffet-Smith, P. Practical Astronomy with your calculator, Third Edition. Cambridge University Press, 2004.

angle	<i>Angular distance between the places in the sky</i>
-------	---

Description

The function computes angle between two or more objects on the sky.

Usage

```
angle(x, y = NULL)
```

Arguments

x	an object of class "apos"
y	optional second object of class "apos"

Value

If y==NULL, the function returns value of class (angle,dist), containing the angles (in degrees) between all positions in the given object (for example distances between all planets for one day).

If y is given (y should contain the same row number than x) and the return value is a vector of distances between the subsequent corresponding rows (for example distance between Moon and Sun for some days ahead).

Author(s)

Lukasz Komsta

Examples

```
options(latitude=51,longitude=22)
data(bright)
angle(bright)
angle(planets())
j=jd(length=100)
plot(angle(mercury(j),venus(j))) # angle between Venus and Mercury for next 100 days
```

as.ecc

Convert between different coordinate systems

Description

The function converts data between equatorial, ecliptic and horizontal coordinates.

Usage

```
as.ecc(x)
as.eqc(x,time=lst(),phi=getOption("latitude"))
as.hoc(x,time=lst(),phi=getOption("latitude"))
```

Arguments

x	An object of class eqc,ecc or hoc.
time	Local Sidereal Time - the LST at the moment by default.
phi	Latitude of the observer - taken from options by default.

Value

An converted object of desired class.

Author(s)

Lukasz Komsta

Examples

```
options(latitude=51,longitude=22)
data(bright)
plot(as.hoc(bright))
plot(as.hoc(bright,time=lst(hour=0)))
plot(as.ecc(bright))
plot(as.hoc(planets()))
```

as.gmt

Converting between several time standards

Description

These functions are used for converting between astronomical and sidereal times.

Usage

```
as.gmt(x, jday = jd(), lambda = getOption("longitude"), ...)
as.gst(x, jday = jd(), lambda = getOption("longitude"), ...)
as.lst(x, lambda = getOption("longitude"), ...)
as.lt(x, ...)
```

Arguments

x	an object of class gmt (Greenwich Mean Time), gst (Greenwich Sidereal Time), lst (Local Sidereal Time) or lt (Local Time)
jday	Julian Day Number (default for today)
lambda	Longitude of observer (default taken from options)
...	Additional arguments

Value

A converted object of appropriate class.

Author(s)

Lukasz Komsta

Examples

```
l=lt(length=10)
as.gst(l)
as.lst(l)
as.gmt(l)
as.lt(as.gst(as.gmt(l)))
```

bright

Brigheest stars coordinates

Description

The equatorial coordinates of 23 brightest stars.

Usage

```
data(bright)
```

Format

A data frame of class `eqc` with 23 observations on the following 2 variables.

ra right ascension

d declination

Source

The position data was taken from English Wikipedia.

Examples

```
data(bright)
options(latitude=51, longitude=22)
plot(bright)
plot(as.hoc(bright))
as.lt(rst(bright))
angle(bright)
```

constel	<i>Assign constellation abbreviations to equatorial coordinates</i>
---------	---

Description

This function finds the abbreviations of constellations for given coordinates in equatorial system.

Usage

```
constel(x)
```

Arguments

x an object of class eqc

Details

To be added.

Value

A character vector with constellation abbreviations

Author(s)

Lukasz Komsta

References

<http://vizier.u-strasbg.fr/viz-bin/VizieR?-source=6042>

Examples

```
options(latitude=51,longitude=22)
data(bright)
constel(bright)
constel(planets())
```

`ecc`*Create objects containing coordinates*

Description

These functions are simple way to create objects containing a set of horizontal (hoc), ecliptic (ecc) and equatorial (eqc) coordinates.

Usage

```
ecc(lat, long, names = NULL)
eqc(ra, d, names = NULL)
hoc(az, alt, names = NULL)
```

Arguments

<code>lat</code>	ecliptic latitude
<code>long</code>	ecliptic longitude
<code>ra</code>	right ascension
<code>d</code>	declination
<code>az</code>	azimuth
<code>alt</code>	altititude
<code>names</code>	names of objects

Details

All the arguments to these functions are vectors of the same length, containing corresponding coordinates and names. These are collected into a dataframe of appropriate class - eqc, ecc or hoc.

Value

An object of class eqc/ecc/hoc, apos, data.frame.

Author(s)

Lukasz Komsta

Examples

```
a = ecc(1:360, rep(0, 360), 1:360)
a
as.eqc(a)
plot(as.eqc(a))
```

format.dms

Format an angle for printing

Description

Function for pretty formatting (degrees, minutes, seconds) for angular class of data.

Usage

```
## S3 method for class 'dms'  
format(x, ...)
```

Arguments

x	object to format
...	additional arguments

Value

String with formatted data.

Author(s)

Lukasz Komsta

format.jd

Format Julian Day Number for pretty printing

Description

Convert Julian Day Number back to date and create string with formatted output.

Usage

```
## S3 method for class 'jd'  
format(x, ...)
```

Arguments

x	object to format
...	additional arguments

Author(s)

Lukasz Komsta

format.time	<i>Format time-related data for pretty printing</i>
-------------	---

Description

Format data expressed as hours from midnight to hours, minutes and seconds.

Usage

```
## S3 method for class 'time'
format(x, ...)
```

Arguments

x	object to format
...	additional arguments

Author(s)

Lukasz Komsta

gmt	<i>Create sequences of a time</i>
-----	-----------------------------------

Description

The functions for creating a time (or time sequences) measured in Greenwich Mean Time (gmt), Greenwich Sidereal Time (gst), Local Sidereal Time (lst), Local Time (lt).

Usage

```
gmt(hour = NULL, minute = 0, second = 0, epoch = Sys.time(), length = 1, by = 1)
gst(jday = jd(), hour = NULL, minute = 0, second = 0, epoch = Sys.time(), length = 1, by = 1)
lst(..., lambda = getOption("longitude"))
lt(hour = NULL, minute = 0, second = 0, epoch = Sys.time(), length = 1, by = 1)
```

Arguments

jday	Julian Day Number
hour	hour
minute	minute
second	second
epoch	epoch
length	length

by step in sequence
 . . . additional arguments
 lambda longitude of the observer, default taken from options

Details

The functions `gmt()` and `lt()` are simple time series generators. By default they take current time expressed as local or GMT.

The functions `lst()` and `gst()` compute sidereal times for given Julian Day Number and time. Default is for now.

Value

A vector containing times expressed as hours from 00h 00m 00s, of class "time".

Author(s)

Lukasz Komsta

Examples

```
lt()
gmt()
gst()
lst()
options(latitude=51,longitude=22)
lst(jd(2008,01,01),hour=12) # Local Sidereal Time, 1st January 2008 1200 UTC
lst(length=10) # 10 hours ahead sequence from now
```

jd	<i>Julian Day Number</i>
----	--------------------------

Description

Compute the Julian Day Number for a given date, optionally generating a sequence.

Usage

```
jd(year = NULL, month = NULL, day = NULL, epoch = Sys.time(), length = 1, by = 1)
```

Arguments

year year
 month month
 day day
 epoch epoch (number of seconds since 1st January 1970 0000 UTC)
 length length of sequence
 by step of sequence

Details

If any of the year, month or day parameters is given (and thus nonzero) the date is taken from these parameters. If not, the epoch parameter is considered (default taken from system timer).

Value

A vector of Julian Day Numbers.

Author(s)

Lukasz Komsta

Examples

```
jd()  
jd(1978,10,16)  
jd(length=10)
```

moon

Equatorial Coordinates of Moon

Description

The function computes equatorial coordinates of Moon.

Usage

```
moon(jday = jd() + gmtime()/24)
```

Arguments

jday Julian Day Number.

Value

An object of class eqc, apos, data.frame containing computed coordinates. See planet() for details.

Note

The daily motion of the Moon is significant, and therefore default behavior of the function is to add a day fraction to the Julian Day Number, depending on current hour.

The algorithm used here is fairly simple and the expected accuracy is within 12 arc minutes of expected coordinates.

Author(s)

Lukasz Komsta

Examples

```
moon()
moon(jd(length=30))
as.ecc(moon())
```

planet

Compute coordinates of a planet in solar system.

Description

This function computes equatorial coordinates for inner or outer planet for given Julian Day Number.

Usage

```
planet(jday = jd(), name = "", inner = FALSE, tp, ep, oo, e, a, i, om, th, mag)
```

Arguments

jday	Julian Day Number, default today
name	name of a planet (appended to dates in result)
inner	TRUE if it is inner, FALSE if outer planet
tp	period of a planet (tropical years)
ep	longitude at epoch 1990 January 0.00 (degrees)
oo	longitude of the perihelion (degrees)
e	eccentricity of the orbit
a	semi-major axis of the orbit (AU)
i	inclination of the orbit (degrees)
om	longitude of the ascending node (degrees)
th	angular diameter at 1 AU (arcsecs)
mag	visual magnitude at 1 AU

Details

The algorithm used here is fairly simple, it does not consider the Kepler equation, nor gravitational influences from other planets. See `sun()` for details.

This function is not called by user unless calculating a position for planetoid or modified data. The `planets()` function calls it with appropriate parameters automatically.

Value

An object of class `eqc`, containing position and other data for requested days, see `planets()` for details.

Author(s)

Lukasz Komsta

Examples

```
planets()
```

planets	<i>Coordinates of all planets for given day</i>
---------	---

Description

Compute equatorial coordinates for all planets (and also Moon and Sun if needed) by one function call.

Usage

```
planets(jday = jd(), show.sun = TRUE, show.moon = TRUE)
```

Arguments

jday	Julian Day Number (default today)
show.sun	should the Sun position be computed?
show.moon	should the Moon position be computed?

Details

The function calls the planet(), sun() and moon() function for each object.

Value

An object of class eqc, containing all computed coordinates.

Author(s)

Lukasz Komsta

Examples

```
planets()  
rst(planets())
```

plot.apos

Coordinates plots

Description

Plot positions of objects in the sky

Usage

```
## S3 method for class 'apos'
plot(x, label = TRUE, grid = TRUE, type = "n", ...)
```

Arguments

x	an object inherited from class "apos" ("eqc", "ecc" or "hoc")
label	should labels be plotted?
grid	shoulf grid be plotted?
type	type passed to plot(), default "n" (when labels are TRUE)
...	additional parameters passed to plot()

Author(s)

Lukasz Komsta

Examples

```
options(latitude=51,longitude=22)
par(mfrow=c(2,2))
data(bright)
plot(bright)
plot(as.ecc(bright))
plot(as.hoc(bright))
plot(as.lt(rst(bright)))
```

print.eqc

Format astronomical data for pretty printing

Description

Internal functions for pretty printing.

Usage

```

## S3 method for class 'eqc'
print(x, ...)
## S3 method for class 'jd'
print(x, ...)
## S3 method for class 'time'
print(x, ...)

```

Arguments

x	object to be formatted
...	another arguments

Author(s)

Lukasz Komsta

 rst

Rise, Transit and Set of specific coordinate points

Description

Compute Time of Rise, Transit and Set, and also azimuths of Rise and Set, for given positions, expressed in LOCAL SIDEREAL TIME.

Usage

```

rst(x, phi = getOption("latitude"))
sun.rst(jday = jd(), phi = getOption("latitude"))
moon.rst(jday = jd(), phi = getOption("latitude"))

## S3 method for class 'rst'
plot(x, annotate = TRUE, ...)

```

Arguments

x	an object of class eqc
jday	Julian Day Number
phi	observer's latitude (default taken from options)
annotate	should the plot be annotated (set FALSE for large periods)
...	additional arguments passed to plot()

Details

The computed time is expressed as LOCAL SIDEREAL TIME (thus, longitude is not needed). If you want to convert it to local time, use `as.lt()`.

The `rst()` function does not consider any motion of the object, so it shows some inaccuracy for Sun and very significant inaccuracy (even an hour) for the Moon. `sun.rst()` and `moon.rst()` function are designed for calculating the better rise, transit and set times of Sun and Moon by stepwise approximation.

Value

An object of class "rst". If all values are -Inf, the object never rises above horizon. If rise and set are Inf, the object is always above horizon and only transit time is computed.

Author(s)

Lukasz Komsta

Examples

```
options(latitude=51,longitude=22)
data(bright)
rst(bright)
as.lt(rst(bright))
as.lt(rst(planets()))
as.lt(moon.rst(jd(length=30)))
```

starcat

Data extracted from Yale bright star catalogue

Description

The data frame contains the equatorial coordinates at epoch 2000.0 of 9110 bright stars extracted from Yale bright star catalogue.

Usage

```
data(starcat)
```

Format

A data frame with 9110 rows on the following 4 columns.

resRA Right Ascension of each star at epoch 2000.0

rrr Declination of each star at epoch 2000.0

Vmag Visual magnitude for each star.

SAO SAO index for each star.

Source

<ftp://cdsarc.u-strasbg.fr/pub/cats/V/50/catalog.gz>

References

Hoffleit D., Warren Jr W.H. The Bright Star Catalogue, 5th Revised Ed. (Preliminary Version) 1991

Examples

```
data(starcats)
```

sun

Equatorial coordinates for celestial objects (ephemerides)

Description

These functions compute equatorial coordinates of celestial objects at given day, their phase, position of the limb, distance from earth and the magnitude.

Usage

```
sun(jday = jd())  
mercury(jday = jd())  
venus(jday = jd())  
mars(jday = jd())  
jupiter(jday = jd())  
saturn(jday = jd())  
uranus(jday = jd())  
neptune(jday = jd())  
pluto(jday = jd())
```

Arguments

jday Julian Day number

Details

The algorithms used here are fairly simple and not with top-accuracy.

Sun is assumed to be always on ecliptic and no eccentric anomaly is considered. The accuracy should be within 10s of right ascension and few minutes of declination.

Planets position are calculated without solving the Kepler Equation and considering perturbations, so the accuracy is similar.

Value

An object of class "eqc, apos, data.frame", containing a row for each day, and following columns:

ra	Right Ascension
d	Declination
phase	Percentage of bright area visible from Earth
angle	Angle between the limb and north-south equatorial axis
dist	Distance from Earth in AUs
size	Size in arcsecs
mag	Magnitude

Author(s)

Lukasz Komsta

Examples

```
options(latitude=51.25,longitude=22.5) # Lublin, Poland
j=jd(length=30) # Next 30 days
sun(j) # Equatorial position
as.hoc(sun(j),j) # Horizontal position at current time
```

track

Plot track of planets

Description

To plot tracks of planets during to dates.

Usage

```
track(ephem, mag = 7, edge = 0.2, cex.star = 1,xlab = "Right Ascension",
      ylab = expression(paste("Declination", degree)), col.track = "red",
      interval.lab = 15, lwd.track = 2, grid = TRUE, bright.lab = TRUE,
      bright, starcat,...)
```

Arguments

ephem	Equatorial coordinates of celestial objects as generated. see planet for more information.
mag	Magnitude of stars to be plot on the background.
edge	Numeric, the edge extention of the plots from each side of the plotting margin.
cex.star	Numeric, indicating the point size of the stars to be plotted on the background.
xlab	label for the x axis.

ylab	label for the y axis.
col.track	color of the planet's track .
interval.lab	Time interval for the label.
lwd.track	Line width of the planet's track.
grid	Whether the grid of coordinate's should be drawn.
bright.lab	Whether to draw the labels of bright stars on the plot.
bright	Data frame for bright stars
starcats	Star catalogue used in plotting.
...	Other specified methods to draw the plot.

Details

User may employ this function when she/he wants to see the tracks of certain planet during time 1 to time 2. The input data should be generated by `planet`, or `planet` related functions such as `mercury`, `mars`, `saturn`. User may define her/his own celestial objects and plot the tracks using this function.

Value

NULL

Author(s)

Jinlong Zhang <jinlongzhang01@gmail.com>

References

<http://www.clearskyinstitute.com/xephem/>

<http://www.alcyone.de/>

Examples

```
# Beijing
options(longitude = 116.433, latitude = 39.874)

### plot the background of sky chart
data(bright)
data(starcats)

ephem.mercury <- mercury(jd(2010,1,1,length = 365))
track(ephem.mercury, mag = 4, interval.lab = 30, bright.lab = TRUE, starcats = starcats, bright = bright)

ephem.mercury <- mercury(jd(2010,7,30,length = 60))
track(ephem.mercury, mag = 7, interval.lab = 10, cex.star = 2, starcats = starcats, bright = bright)

ephem.venus <- venus(jd(2011,4,30,length = 100))
track(ephem.venus, col.track = "blue", lwd = 3, mag = 5, starcats = starcats, bright = bright)
```

```
ephem.mars <- mars(jd(2010,8,30,length = 100))
track(ephem.mars, cex.star = 1.5, grid = FALSE, starcat = starcat, bright = bright)

ephem.jupiter <- jupiter(jd(2010,9,22,length = 100))
track(ephem.jupiter, cex.star = 1.5, col.track = "black" , starcat = starcat, bright = bright)

ephem.saturn <- saturn(jd(2011,9,22,length = 100))
track(ephem.saturn, starcat = starcat, bright = bright)

ephem.uranus <- uranus(jd(2008,8,1,length = 100))
track(ephem.uranus, starcat = starcat, bright = bright)
```

Index

*Topic **datasets**

bright, 5
starcats, 16

*Topic **hplot**

plot.apos, 14

*Topic **manip**

as.ecc, 3
as.gmt, 4
constel, 6
ecc, 7
format.dms, 8
format.jd, 8
format.time, 9

*Topic **math**

angle, 2
gmt, 9
jd, 10
moon, 11
planet, 12
planets, 13
rst, 15
sun, 17

*Topic **package**

moonsun-package, 2

*Topic **print**

print.eqc, 14

*Topic **track**

track, 18

angle, 2
as.ecc, 3
as.eqc (as.ecc), 3
as.gmt, 4
as.gst (as.gmt), 4
as.hoc (as.ecc), 3
as.lst (as.gmt), 4
as.lt (as.gmt), 4

bright, 5

constel, 6

ecc, 7
eqc (ecc), 7

format.dms, 8
format.jd, 8
format.time, 9

gmt, 9
gst (gmt), 9

hoc (ecc), 7

jd, 10
jupiter (sun), 17

lst (gmt), 9
lt (gmt), 9

mars, 19
mars (sun), 17
mercury, 19
mercury (sun), 17
moon, 11
moon.rst (rst), 15
moonsun (moonsun-package), 2
moonsun-package, 2

neptune (sun), 17

planet, 12, 18, 19
planets, 13
plot.apos, 14
plot.rst (rst), 15
pluto (sun), 17
print.eqc, 14
print.jd (print.eqc), 14
print.time (print.eqc), 14

rst, 15

saturn, [19](#)
saturn (sun), [17](#)
starcats, [16](#)
sun, [17](#)
sun.rst (rst), [15](#)

track, [18](#)

uranus (sun), [17](#)

venus (sun), [17](#)