

Package ‘swephR’

February 18, 2019

Type Package

Title High Precision Swiss Ephemeris

Version 0.1.5

Description The Swiss Ephemeris is a high precision ephemeris based upon the DE431 ephemerides from NASA's JPL. It covers the time range 13201 BC to AD 17191. This package uses the semi-analytic theory by Steve Moshier. For faster and more accurate calculations, the compressed Swiss Ephemeris data is available in the 'swephRdata' package. To access this data package, run 'install.packages("swephRdata", repos = "https://rstub.github.io/drat/", type = "source")'. The size of the 'swephRdata' package is approximately 115 MB. The user can also use the original JPL DE431 data.

License AGPL | file LICENSE

Imports Rcpp (>= 0.12.18)

LinkingTo Rcpp

RoxygenNote 6.1.1

Suggests testthat, swephRdata, knitr, rmarkdown

Encoding UTF-8

URL <https://github.com/rstub/swephR/>, <http://www.astro.com/swisseph/>

BugReports <https://github.com/rstub/swephR/issues/>

Additional_repositories <https://rstub.github.io/drat>

VignetteBuilder knitr

NeedsCompilation yes

Author Ralf Stubner [aut, cre],
Victor Reijs [aut],
Authors and copyright holder of the Swiss Ephemeris [aut, cph] (see
LICENSE for details)

Maintainer Ralf Stubner <ralf.stubner@gmail.com>

Repository CRAN

Date/Publication 2019-02-18 15:00:03 UTC

R topics documented:

swephR-package	2
SE	3
Section1	3
Section16	4
Section2	5
Section3	6
Section4	6
Section6	7
Section7	12
Section8	13
Section9	14

Index	16
--------------	-----------

swephR-package	<i>swephR: High Precision Swiss Ephemeris</i>
----------------	-----------------------------------------------

Description

The Swiss Ephemeris is a high precision ephemeris based upon the DE431 ephemerides from NASA's JPL. It covers the time range 13201 BC to AD 17191. This package uses the semi-analytic theory by Steve Moshier. For faster and more accurate calculations, the compressed Swiss Ephemeris data is available in the 'swephRdata' package. To access this data package, run 'install.packages("swephRdata", repos = "https://rstub.github.io/drat/", type = "source")'. The size of the 'swephRdata' package is approximately 115 MB. The user can also use the original JPL DE431 data.

Author(s)

Maintainer: Ralf Stubner <ralf.stubner@gmail.com>

Authors:

- Victor Reijs
- Authors and copyright holder of the Swiss Ephemeris (see LICENSE for details) [copyright holder]

See Also

Useful links:

- <https://github.com/rstub/swephR/>
- <http://www.astro.com/swisseph/>
- Report bugs at <https://github.com/rstub/swephR/issues/>

SE	<i>Constants used in swephR</i>
----	---------------------------------

Description

- name of variable
- value of the variable

Usage

```
data(SE)
```

Format

A data frame with 217 rows and 2 variables

Section1	<i>Section 1: The Ephemeris file related functions</i>
----------	--------------------------------------------------------

Description

Several initialization functions

Usage

```
swe_set_ephe_path(path)
```

```
swe_close()
```

```
swe_set_jpl_file(fname)
```

```
swe_version()
```

Arguments

path	Directory for the sefstars.txt, swe_deltat.txt and jpl files
------	--------------------------------------------------------------

fname	JPL ephemeris name as string (JPL ephemeris file, e.g. de431.eph)
-------	-------------------------------------------------------------------

Details

swe_set_ephe_path() This is the first function that should be called before any other function of the Swiss Ephemeris. Even if you don't want to set an ephemeris path and use the Moshier ephemeris, it is nevertheless recommended to call `swe_set_ephe_path(NULL)`, because this function makes important initializations. If you don't do that, the Swiss Ephemeris may work, but the results may be not 100% consistent.

swe_close() At the end of your computations this function releases most resources (open files and allocated memory) used by Swiss Ephemeris.

swe_set_jpl_file() Set name of JPL ephemeris file.

swe_version() The function provides the version number of the Swiss Ephemeris software.

Value

`swe_version` returns Swiss Ephemeris software version as string

See Also

http://www.astro.com/swisseph/swephprg.htm?lang=g#_Toc505244831

Examples

```
## Not run: swe_set_ephe_path("c:\\sweph\\ephe")
swe_close()
swe_set_jpl_file("de431.eph")
swe_version()
```

Description

Useful functions

Usage

```
swe_day_of_week(jd)
```

Arguments

jd Julian day number as numeric vector (day)

Details

swe_day_of_week() Determine day of week from Julian day number.

Value

`swe_day_of_week` returns the day of week as integer vector (0 Monday .. 6 Sunday)

See Also

http://www.astro.com/swiseph/swephprg.htm#_Toc505244906

Examples

```
swe_day_of_week(1234.567)
```

 Section2

Section 2: Computing positions

Description

Computing positions of planets, asteroids, lunar nodes and apogees using Swiss Ephemeris.

Usage

```
swe_calc_ut(jd_ut, ipl, iflag)
```

```
swe_calc(jd_et, ipl, iflag)
```

Arguments

jd_ut	UT Julian day number as double (day)
ipl	Body/planet as integer (SE\$SUN=0, SE\$Moon=1, ... SE\$PLUTO=9)
iflag	Computation flag as integer, many options possible (section 2.3)
jd_et	ET Julian day number as double (day)

Details

swe_calc_ut() It compute positions using UT.

swe_calc() It compute positions using ET.

Value

swe_calc_ut returns a list with named entries: return status flag as integer, xx information on planet position, and serr error message as string.

swe_calc returns a list with named entries: return status flag as integer, xx updated star name as string and serr error message as string.

See Also

http://www.astro.com/swiseph/swephprg.htm#_Toc505244837

Examples

```
data(SE)
swe_calc_ut(2458346.82639, SE$MOON, SE$FLG_MOSEPH)
swe_calc(2458346.82639, SE$MOON, SE$FLG_MOSEPH)
```

Section3

Section 3: Find a planetary or asteroid name

Description

Find a planetary or asteroid name.

Usage

```
swe_get_planet_name(ipl)
```

Arguments

ipl Body/planet as integer (SE\$SUN=0, SE\$Moon=1, ... SE\$PLUTO=9)

Details

swe_get_planet_name() Convert object number into object name.

Value

swe_get_planet_name returns the object's name as string

See Also

http://www.astro.com/swisseph/swephprg.htm#_Toc505244843

Examples

```
data(SE)
swe_get_planet_name(SE$MOON)
```

Section4

Section 4: Fixed stars functions

Description

The following functions are used to calculate positions of fixed stars.

Usage

```
swe_fixstar2_mag(starname)
```

```
swe_fixstar2(starname, jd_et, iflag)
```

```
swe_fixstar2_ut(starname, jd_ut, iflag)
```

Arguments

starname	Star name as string ("" for no star)
jd_et	ET Julian day number as double (day)
iflag	Calculation flag as integer, many options possible (section 2.3)
jd_ut	UT Julian day number (day)

Details

swe_fixstar2_mag() Calculate visible magnitude (Vmag) of star.

swe_fixstar2() Compute information of star using ET.

swe_fixstar2_ut() Compute information of star using UT

Value

swe_fixstar2_mag returns a list with named entries: return status flag as integer, starname updated star name as string, mag magnitude of star as double, and serr for error message as string.

swe_fixstar2 returns a list with named entries: return status flag as integer, starname updated star name as string, xx star phenomena as numeric vector, and serr error message as string.

swe_fixstar2_ut returns a list with named entries: return status flag as integer, starname updated star name as string, xx star information as numeric vector, and serr for error message as string.

See Also

http://www.astro.com/swisseph/swephprg.htm#_Toc505244844

Examples

```
data(SE)
swe_fixstar2_mag("sirius")
swe_set_topo(0,50,10)
swe_fixstar2("sirius",1234567,SE$FLG_TOPOCTR+SE$FLG_MOSEPH+SE$FLG_EQUATORIAL)
swe_fixstar2_ut("sirius",1234567,SE$FLG_TOPOCTR+SE$FLG_MOSEPH+SE$FLG_EQUATORIAL)
```

Description

Functions for: determining eclipse and occultation calculations, computing the times of rising, setting and meridian transits for all planets, asteroids, the moon and the fixed stars; computing phase, phase angle, elongation, apparent diameter, apparent magnitude for the Sun, the Moon, all planets and asteroids; and determining heliacal phenomenon after a given start date

Usage

```

swe_sol_eclipse_when_loc(jd_start, ephe_flag, geopos, backward)

swe_lun_eclipse_when_loc(jd_start, ephe_flag, geopos, backward)

swe_lun_eclipse_how(jd_ut, ephe_flag, geopos)

swe_lun_eclipse_when(jd_start, ephe_flag, ifltype, backward)

swe_rise_trans_true_hor(jd_ut, ipl, starname, ephe_flag, rsmi, geopos,
    atpress, attemp, horhgt)

swe_pheno_ut(jd_ut, ipl, ephe_flag)

swe_pheno(jd_et, ipl, ephe_flag)

swe_azalt(jd_ut, coord_flag, geopos, atpress, attemp, xin)

swe_azalt_rev(jd_ut, coord_flag, geopos, xin)

swe_refrac_extended(InAlt, height, atpress, attemp, lapse_rate, calc_flag)

swe_heliacal_ut(jd_utstart, dgeo, datm, dobs, objectname, event_type,
    helflag)

swe_vis_limit_mag(jd_ut, dgeo, datm, dobs, objectname, helflag)

swe_heliacal_pheno_ut(jd_ut, dgeo, datm, dobs, objectname, event_type,
    helflag)

swe_topo_arcus_visionis(jd_ut, dgeo, datm, dobs, helflag, mag, Azi0, Alt0,
    AziS, AziM, AltM)

swe_heliacal_angle(jd_ut, dgeo, datm, dobs, helflag, mag, Azi0, AziS, AziM,
    AltM)

```

Arguments

jd_start	Julian day number as double (UT)
ephe_flag	Ephemeris flag as integer (SE\$FLG_JPLEPH=1, SE\$FLG_SWIEPH=2 or SE\$FLG_MOSEPH=4)
geopos	position as numeric vector (longitude, latitude, height)
backward	backwards search as boolean (TRUE)
jd_ut	UT Julian day number as double (day)
ifltype	eclipse type as integer (e.g.: SE\$ECL_CENTRAL=1, SE\$ECL_NONCENTRAL=2, SE\$ECL_TOTAL=4,
ipl	Body/planet as integer (SE\$SUN=0, SE\$MOON=1, ... SE\$PLUTO=9)
starname	Star name as string ("" for no star)

rsmi	Event flag as integer (e.g.: SE\$CALC_RISE=1, SE\$CALC_SET=2, SE\$CALC_MTRANSIT=4, SE\$CALC_
atpress	Atmospheric pressure as double (hPa)
attemp	Atmospheric temperature as double (Celsius)
horhgt	Horizon apparent altitude as double (deg)
jd_et	ET Julian day number as double (day)
coord_flag	Coordinate flag as integer (reference system (SE\$ECL2HOR=0 or SE\$EQU2HOR=1))
xin	Position of body as numeric vector (either ecliptical or equatorial coordinates, depending on coord_flag)
InAlt	object's apparent/topocentric altitude as double (depending on calc_flag) (deg)
height	observer's height as double (m)
lapse_rate	lapse rate as double (K/m)
calc_flag	Calculation flag as integer (refraction direction (SE\$TRUE_TO_APP=0 or SE\$APP_TO_TRUE=1))
jd_utstart	UT Julian day number as double (day)
dgeo	Geographic position as numeric vector (longitude, latitude, height)
datm	Atmospheric conditions as numeric vector (pressure, temperature, relative humidity, visibility)
dobs	Observer description as numeric vector
objectname	Name of fixed star or planet as string
event_type	Event type as integer
helflag	Calculation flag (incl. ephe_flag values) as integer
mag	Object's visible magnitude (Vmag) as double (-)
Azi0	Object's azimuth as double (deg)
Alt0	Object's altitude as double (deg)
AziS	Sun's azimuth as double (deg)
AziM	Moon's azimuth as double (deg)
AltM	Moon's altitude as double (deg)

Details

swe_sol_eclipse_when_loc() Find the next solar eclipse for a given geographic position.

swe_lun_eclipse_when_loc() Find the next lunar eclipse for a given geographic position.

swe_lun_eclipse_how() Compute the attributes of a lunar eclipse for a given time.

swe_lun_eclipse_when() Find the next lunar eclipse on earth.

swe_rise_trans_true_hor() Compute the times of rising, setting and meridian transits for planets, asteroids, the moon, and the fixed stars for a local horizon that has an altitude.

swe_pheno_ut() Compute phase, phase angle, elongation, apparent diameter, apparent magnitude for the Sun, the Moon, all planets and asteroids (UT)

swe_pheno() Compute phase, phase angle, elongation, apparent diameter, apparent magnitude for the Sun, the Moon, all planets and asteroids (ET).

swe_azalt() Compute the horizontal coordinates (azimuth and altitude) of a planet or a star from either ecliptical or equatorial coordinates.

swe_azalt_rev() compute either ecliptical or equatorial coordinates from azimuth and true altitude. If only an apparent altitude is given, the true altitude has to be computed first with e.g. the function `swe_refrac_extended()`.

swe_refrac_extended() Calculate either the topocentric altitude from the apparent altitude or the apparent altitude from the topocentric altitude. It allows correct calculation of refraction for heights above sea > 0 , where the ideal horizon and planets that are visible may have a negative altitude.

swe_heliacal_ut() Compute the Julian day of the next heliacal phenomenon after a given UT start date. It works between geographic latitudes 60 South and 60 North.

swe_vis_limit_mag() Determine the limiting visual magnitude in dark skies. If the visual magnitude `mag` of an object is known for a given date (e. g. from a call of function `swe_pheno_ut()`), and if magnitude is smaller than the value returned by `swe_vis_limit_mag()`, then it is visible.

swe_heliacal_pheno_ut() Provide data that are relevant for the calculation of heliacal risings and settings. This function does not provide data of heliacal risings and settings itself, just some additional data mostly used for test purposes. To calculate heliacal risings and settings, use the function `swe_heliacal_ut()`.

swe_topo_arcus_visionis() Compute topocentric arcus visionis.

swe_heliacal_angle() Compute heliacal angle.

Value

`swe_sol_eclipse_when_loc` returns a list with named entries: `return` status flag as integer, `tret` for eclipse timing moments as numeric vector, `attr` phenomena during eclipse as numeric vector and `serr` error message as string

`swe_lun_eclipse_when_loc` returns a list with named entries: `return` status flag as integer, `tret` for eclipse timing moments, `attr` phenomena during eclipse and `serr` error warning as string

`swe_lun_eclipse_how` returns a list with named entries: `return` status flag as integer, `attr` phenomena during eclipse as numeric vector and `serr` error message as string

`swe_lun_eclipse_when` returns a list with named entries: `return` status flag as integer, `tret` for eclipse timing moments as numeric vector and `serr` error warning as string

`swe_rise_trans_true_hor` returns a list with named entries: `return` status flag as integer, `tret` for azimuth/altitude info as numeric vector and `serr` error message as string

`swe_pheno_ut` returns a list with named entries: `return` status flag as integer, `attr` for phenomenon information as numeric vector and `serr` error warning as string

`swe_pheno` returns a list with named entries: `return` status flag as integer, `attr` for phenomenon information as numeric vector and `serr` error message as string

swe_azalt returns a list with named entries: xaz for azi/alt info as numeric vector.

swe_azalt_rev returns a list with named entries: xaz for celestial info as numeric vector.

swe_refrac_extended returns a list with named entries: return status flag as integer, dret refraction results as numeric vector (TopoAlt, AppAlt, refraction)

swe_heliacal_ut returns a list with named entries return status flag as integer, dret heliacal results as numeric vector, and serr error message as string.

swe_vis_limit_mag returns a list with named entries: return status flag as integer, dret limiting magnitude as double and serr error message as string

swe_heliacal_pheno_ut returns a list with named entries: return status flag as integer darr for heliacal details as numeric vector and serr error message as string

swe_topo_arcus_visionis returns a list with named entries: return status flag as integer, darr heliacal details as numeric vector and serr error message as string

swe_heliacal_angle returns a list with named entries: return status flag as integer, dret heliacal angle as numeric vector and serr error message as string

See Also

http://www.astro.com/swisseph/swephprg.htm#_Toc505244853

Examples

```
data(SE)
swe_sol_eclipse_when_loc(1234567, SE$FLG_MOSEPH, c(0, 50, 10), FALSE)
swe_lun_eclipse_when_loc(1234567, SE$FLG_MOSEPH, c(0, 50, 10), FALSE)
swe_lun_eclipse_how(1234580.19960447, SE$FLG_MOSEPH, c(0, 50, 10))
swe_lun_eclipse_when(1234567, SE$FLG_MOSEPH, SE$ECL_CENTRAL, FALSE)
swe_rise_trans_true_hor(1234567.5, SE$SUN, "", SE$FLG_MOSEPH, 0, c(0, 50, 10), 1013.25, 15, 0)
swe_pheno_ut(1234567, 1, SE$FLG_MOSEPH)
swe_pheno(1234567, 1, SE$FLG_MOSEPH)
swe_azalt(1234567, SE$EQU2HOR, c(0, 50, 10), 15, 1013.25, c(186, 22))
swe_azalt_rev(1234567, SE$ECL2HOR, c(0, 50, 10), c(123, 2))
swe_refrac_extended(2, 0, 1013.25, 15, -0.065, SE$TRUE_TO_APP)
swe_heliacal_ut(1234567, c(0, 50, 10), c(1013.25, 15, 50, 0.25), c(25, 1, 1, 1, 5, 0.8), "sirius",
  SE$HELIALACAL_RISING, SE$HELFLAG_HIGH_PRECISION+SE$FLG_MOSEPH)
swe_vis_limit_mag(1234567.5, c(0, 50, 10), c(1013.25, 15, 20, 0.25), c(25, 1, 1, 1, 5, 0.8), 'sirius',
  SE$HELFLAG_HIGH_PRECISION+SE$FLG_MOSEPH)
swe_heliacal_pheno_ut(1234567.5, c(0, 50, 10), c(1013.25, 15, 20, 0.25), c(25, 1, 1, 1, 5, 0.8), 'sirius',
  SE$HELIALACAL_RISING, SE$HELFLAG_HIGH_PRECISION+SE$FLG_MOSEPH)
swe_topo_arcus_visionis(1234567.5, c(0, 50, 10), c(1013.25, 15, 20, 0.25), c(25, 1, 1, 1, 5, 0.8),
  SE$HELFLAG_HIGH_PRECISION+SE$HELFLAG_OPTICAL_PARAMS, -1, 124, 2, 120, 0, -45)
swe_heliacal_angle(1234567.5, c(0, 50, 10), c(1013.25, 15, 20, 0.25), c(25, 1, 1, 1, 5, 0.8),
  SE$HELFLAG_HIGH_PRECISION+SE$HELFLAG_OPTICAL_PARAMS, -1, 124, 120, 0, -45)
```

Description

Functions related to calendar and time conversions.

Usage

```
swe_julday(year, month, day, hour, gregflag)
```

```
swe_date_conversion(year, month, day, hour, cal)
```

```
swe_revjul(jd, gregflag)
```

Arguments

year	Astronomical year as integer
month	Month as integer
day	Day as integer
hour	Hour as double
gregflag	Calendar type as integer (SE\$JUL_CAL=0 or SE\$GREG_CAL=1)
cal	Calendar type "g" [Gregorian] or "j" [Julian] as char
jd	Julian day number as double

Details

swe_julday() Convert calendar dates to the astronomical time scale which measures time in Julian day number.

swe_date_conversion() Convert calendar dates to the astronomical time scale which measures time in Julian day number and checks if the calendar date is legal.

swe_revjul() Compute year, month, day and hour from a Julian day number.

Value

swe_date_conversion returns a list with named entries: return status flag as integer, jd Julian day number as double

swe_revjul returns a list with named entries: year year as integer, month month as integer, day day as integer and hour hour as double.

See Also

http://www.astro.com/swiseph/swephprg.htm#_Toc505244873

Examples

```

data(SE)
swe_julday(2000,1,1,12,SE$GREG_CAL)
swe_date_conversion(2000,1,1,12,"g")
swe_revjul(2452500,SE$GREG_CAL)

```

Section8

Section 8: Delta T-related functions

Description

Functions related to DeltaT and tidal acceleration

Usage

```

swe_deltat_ex(jd_ut, ephe_flag)

swe_deltat(jd_ut)

swe_set_tid_acc(t_acc)

swe_get_tid_acc()

swe_set_delta_t_userdef(delta_t)

```

Arguments

jd_ut	Julian day number (UT) as numeric vector (day)
ephe_flag	ephemeris flag as integer (SE\$FLG_JPLEPH=1, SE\$FLG_SWIEPH=2 or SE\$FLG_MOSEPH=4) (section 2.3.2)
t_acc	Tidal acceleration as double (arcsec/century ²)
delta_t	DeltaT (day)

Details

swe_deltat_ex() Determine DeltaT from Julian day number for a specific ephemeris.

swe_deltat() Determine DeltaT from Julian day number for a used ephemeris. This function is only safe if:

- your software consistently uses the same ephemeris flag
- if software consistently uses the same ephemeris files (with SE\$FLG_SWIEPH and SE\$FLG_MOSEPH)
- if swe_set_ephe_path() is first called (with SE\$FLG_SWIEPH) and swe_set_jpl_file() (with SE\$FLG_JPLEPH)

swe_set_tid_acc() Set the tidal acceleration.

swe_get_tid_acc() Get the present configured tidal acceleration.

swe_set_delta_t_userdef() Allows the user to set a fixed DeltaT value that will be returned by swe_deltat() or swe_deltat_ex().

Value

swe_deltat_ex returns a list with named entries: deltat for DeltaT as double (day) and serr for error message as string.

swe_deltat returns the DeltaT as double (day)

swe_get_tid_acc returns the tidal acceleration as double (arcsec/century²)

See Also

http://www.astro.com/swissep/sweprg.htm#_Toc505244878

Examples

```
data(SE)
swe_deltat_ex(1234.567, SE$FLG_MOSEPH)
swe_deltat(1234.567)
swe_set_tid_acc(1.23)
swe_get_tid_acc()
swe_set_delta_t_userdef(0.23)
```

Description

Function for topocentric planet positions

Usage

```
swe_set_topo(longitude, lat, height)
```

Arguments

longitude	Geographic longitude as double (deg)
lat	Geographic latitude as double (deg)
height	Height as double (m)

Details

swe_set_topo Set the topocentric location of the observer.

See Also

http://www.astro.com/swisseph/swephprg.htm#_Toc505244884

Examples

```
swe_set_topo(0,50,10)
```

Index

*Topic **datasets**

SE, 3

SE, 3

Section1, 3

Section16, 4

Section2, 5

Section3, 6

Section4, 6

Section6, 7

Section7, 12

Section8, 13

Section9, 14

swe_azalt (Section6), 7

swe_azalt_rev (Section6), 7

swe_calc (Section2), 5

swe_calc_ut (Section2), 5

swe_close (Section1), 3

swe_date_conversion (Section7), 12

swe_day_of_week (Section16), 4

swe_deltat (Section8), 13

swe_deltat_ex (Section8), 13

swe_fixstar2 (Section4), 6

swe_fixstar2_mag (Section4), 6

swe_fixstar2_ut (Section4), 6

swe_get_planet_name (Section3), 6

swe_get_tid_acc (Section8), 13

swe_heliacal_angle (Section6), 7

swe_heliacal_pheno_ut (Section6), 7

swe_heliacal_ut (Section6), 7

swe_julday (Section7), 12

swe_lun_eclipse_how (Section6), 7

swe_lun_eclipse_when (Section6), 7

swe_lun_eclipse_when_loc (Section6), 7

swe_pheno (Section6), 7

swe_pheno_ut (Section6), 7

swe_refrac_extended (Section6), 7

swe_revjul (Section7), 12

swe_rise_trans_true_hor (Section6), 7

swe_set_delta_t_userdef (Section8), 13

swe_set_ephe_path (Section1), 3

swe_set_jpl_file (Section1), 3

swe_set_tid_acc (Section8), 13

swe_set_topo (Section9), 14

swe_sol_eclipse_when_loc (Section6), 7

swe_topo_arcus_visionis (Section6), 7

swe_version (Section1), 3

swe_vis_limit_mag (Section6), 7

swephR (swephR-package), 2

swephR-package, 2