Package ‘astrolibR’

February 19, 2015

Type Package

Title Astronomy Users Library

Version 0.1

Date 2014-07-30

Author Arnab Chakraborty and Eric D. Feigelson

Maintainer Eric D. Feigelson <edf@astro.psu.edu>

Description Several dozen low-level utilities and codes from the Interactive Data Language (IDL) Astronomy Users Library (http://idlastro.gsfc.nasa.gov) are implemented in R. They treat: time, coordinate and proper motion transformations; terrestrial precession and nutation, atmospheric refraction and aberration, barycentric corrections, and related effects; utilities for astrometry, photometry, and spectroscopy; and utilities for planetary, stellar, Galactic, and extragalactic science.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2014-08-09 00:14:43

R topics documented:

`astrolibR-package` *astrolibR: Astronomy Users Library*

Description

Several dozen low-level utilities and codes from the Interactive Data Language (IDL) Astronomy Users Library (http://idlastro.gsfc.nasa.gov) are implemented in R. They treat: time, coordinate and proper motion transformations; terrestrial precession and nutation, atmospheric refraction and aberration, barycentric corrections, and related effects; utilities for astrometry, photometry, and spectroscopy; and utilities for planetary, stellar, Galactic, and extragalactic science.

Details

Package:	astrolibR
Type:	Package
Version:	0.1
Date:	2014-07-30
License:	GPL

`adstring` *Return RA and Dec as character string(s) in sexigesimal format*

Description

Return RA and Dec as character string(s) in sexigesimal format

Usage

```
adstring(ra_dec, dec, precision, truncate = FALSE)
```

Arguments

ra_dec
: either a 2-element vector giving Right Ascension and declination, or a scalar giving Right Ascension, in decimal degrees

dec
: a scalr giving declination or ra_dec is also a scalar, in decimal degrees (optional)

precision
: Integer scalar giving number of digits after the decimal of declination. The R.A. precision is automatically 1 digit more. (optional, default = 1)

truncate
: if set, then the last dispayed digit in the output is truncated in precision rather than rounded (optional, default = FALSE)

Details

Common calling sequences are: result <- adstring(ra_dec, [precision=ndigit, truncate = truncate]) or result <- adstring(ra, dec, [precision=ndigit]) or result <- adstring(dec).

The TRUNCATE=TRUE option is useful if adstring() is used to form an official IAU name (see http://vizier.u-strasbg.fr/Dic/iau-spec.htx) with coordinate specification. The IAU name will typically be created with a call like: strcompress(adstring(ra,dec,0,truncate=TRUE))

Value

result
: character string(s) contining HR, MIN, SEC, DEC, MIN, SEC formatted at (2I3, F5.(p+1), 2I3, F4.p) where p is the PRECISION parameter. If only a single scalar is supplied, it is converted to a sexigesimal string with format (2I3, F5.1).

Author(s)

Written W. Landsman June 1988

R adaptation by Arnab Chakraborty (June 2013)

See Also

radec sixty

Examples

```
adstring(30.42, -1.23, 1)  # ' 2  1 40.80 -01 13 48.0'
adstring(30.42, -1.23, 0)  # ' 2  1 40.8 -01 13 48'
adstring(0.4, -0.6, 1)     # ' 0  1 36.00 -00 36  0.0"
adstring(230.42711, -49.5922, 0)  # '15 21 42.5 -49 35 32'
adstring(230.42711, -49.5922, 4)  # '15 21 42.50640 -49 35 31.9200'
```

airtovac — *Convert air wavelengths to vacuum wavelengths*

Description

Convert air wavelengths to vacuum wavelengths

Usage

```
airtovac(wave_air)
```

Arguments

`wave_air`	wavelength in Angstroms, scalar or vector

Details

Wavelengths are corrected for the index of refraction of air under standard conditions. Wavelength values below 2000 A will not be altered. Take care within 1 A of 2000 A. Uses relation of Ciddor (1996).

Value

`wave_vac`	Vacuum wavelength in Angstroms, same number of elements as wave_air

Author(s)

Written W. Landsman November 1991

R adaptation by Arnab Chakraborty June 2013

References

Ciddor 1996, Applied Optics 62, 958 http://adsabs.harvard.edu/abs/1996ApOpt..35.1566C

See Also

vactoair

Examples

```
airtovac(4861.363) #  H beta line wavelength in air
```

`aitoff` *Convert longitude, latitude to X,Y using an AITOFF projection*

Description

Convert longitude, latitude to X,Y using an AITOFF projection

Usage

```
aitoff(l,b)
```

Arguments

`l`	longitude, scalar or vector, in degrees
`b`	latitude, scalar or vector (same length as l), in degrees

Details

This procedure can be used to create an all-sky map in Galactic coordinates with an equal-area Aitoff projection. Output map coordinates are zero longitude centered. See AIPS memo No. 46, page 4, for details of the algorithm. This version of AITOFF assumes the projection is centered at b=0 degrees.

Several polar equal-area map projections are provided by the CRAN package *mapproj*.

Value

`x`	x coordinate in range -180 to +180, same length as l.
`yx`	y coordinate in range -90 to +90, same length as l.

Author(s)

Written W.B. Landsman STX December 1989

R adaptation by Arnab Chakraborty June 2013

References

Additional Non-linear Coordinates in AIPS, Eric W. Greisen, AIPS Memo 46, 1993 ftp://ftp.aoc.nrao.edu/pub/software/aips/TEXT/PUBL/AIPSMEMO46.PS.

Examples

```
aitoff(227.23,-8.890)  #  celestial location of Sirius in Galactic coordinates
```

`altaz2hadec` *Convert horizon (Alt-Az) coordinates to hour angle and declination*

Description

Convert horizon (Alt-Az) coordinates to hour angle and declination

Usage

```
altaz2hadec(alt,az,lat)
```

Arguments

`alt`	local apparent altitude, in degrees, scalar or vector
`az`	local apparent altitude, in degrees, scalar or vector, measured east of north
`lat`	local geodetic latitude, in degrees, scalar or vector

Details

For inputs, if you have measured azimuth west-of-south (like the book MEEUS does), convert it to east of north via: az = (az + 180) mod 360 For outputs, the hour angle is the time that right ascension of 0 hours crosses the local meridian.

Value

`ha`	local apparent hour angle, in degrees
`dec`	local apparent declination, in degrees

Author(s)

Written by Chris O'Dell Univ. of Wisconsin-Madison May 2002

R adaptation by Arnab Chakraborty June 2013

See Also

`hadec2altaz`

Examples

```
altaz2hadec(59.0861,133.3081,41.3)
```

baryvel — *Calculates heliocentric and barycentric velocity components of Earth*

Description

Calculates heliocentric and barycentric velocity components of Earth

Usage

```
baryvel(dje,deq)
```

Arguments

`dje`	Julian ephemeris date (scalar)
`deq`	epoch of mean equinox of dvelh and dvelb (scalar). If deq=0, then deq is assumed to be equal to dje.

Details

The 3-vectors DVELH and DVELB are given in a right-handed coordinate system with the +X axis toward the Vernal Equinox, and +Z axis toward the celestial pole.

The projected velocity towards the celestial object can be computed from: v = dvelb[1]*cos(dec)*cos(ra) + dvelb[2]*cos(dec)*sin(ra) + dvelb[3]*sin(dec)

The algorithm here is taken from FORTRAN program of Stumpff (1980). Stumpf claimed an accuracy of 42 cm/s for the velocity. A comparison with the JPL FORTRAN planetary ephemeris program PLEPH found agreement to within about 65 cm/s between 1986 and 1994. The option in IDL astrolib's baryvel.pro to use the full JPL ephemeris is not implemented in R.

Value

`dvelh`	heliocentric velocity components (3 element vector) in km/s
`dvelb`	barycentric velocity components (3 element vector) in km/s

Author(s)

Jeff Valenti, U.C. Berkeley Translated BARVEL.FOR to IDL
R adaptation by Arnab Chakraborty June 2013

References

Two Self-Consistent FORTRAN Subroutines for the Computation of the Earth's Motion, P. Stumpff, Astronomy & Astrophysics Supplement, 41, 1, 1980.

Examples

```
baryvel(2456469.5, 0)
```

`bprecess` *Precess celestial positions from J2000.0 (FK5) to B1950.0 (FK4)*

Description

Precess celestial positions from J2000.0 (FK5) to B1950.0 (FK4)

Usage

```
bprecess(ra,dec,mu_radec,parallax,rad_vel,epoch)
```

Arguments

`ra`	J2000 right ascension, in degrees (scalar or N-vector)
`dec`	J2000 declination, in degrees (scalar or N-vector)
`mu_radec`	2xN element vector containing proper motion in seconds or arc per tropical century in right ascension and declination (optional)
`parallax`	parallax in seconds of arc, scalar or N-vector (optional)
`rad_vel`	radial velocity in km/s, scalaror N-vector (optional)
`epoch`	epoch of original observation, default 2000.0 (optional)

Details

Calculates the mean place of a celestial object at B1950.0 on the FK4 system from the mean place at J2000.0 on the FK5 system. The epoch input is only used if the mu_radec input is not set.

The algorithm is taken from the Explanatory Supplement to the Astronomical Almanac 1992, page 186. Also see Aoki et al (1983).

BPRECESS distinguishes between the following two cases: (1) The proper motion is known and non-zero; and (2) the proper motion is unknown or known to be exactly zero (i.e. extragalactic radio sources). In the latter case, the reverse of the algorithm in Appendix 2 of Aoki et al. (1983) is used to ensure that the output proper motion is exactly zero. Better precision can be achieved in this case by inputting the EPOCH of the original observations.

The error in using the IDL procedure PRECESS for converting between B1950 and J1950 can be up to 12", mainly in right ascension. If better accuracy than this is needed then BPRECESS should be used. An unsystematic comparison of BPRECESS with the IPAC precession routine (http://nedwww.ipac.caltech.edu/forms/calculator.html) always gives differences less than 0.15".

Value

`ra_1950`	B1950 right ascension, in degrees (scalar or N-vector)
`dec_1950`	B1950 declination, in degrees (scalar or N-vector)

Author(s)

Written, W. Landsman October, 1992

R adaptation by Arnab Chakraborty June 2013

References

The Explanatory Supplement to the Astronomical Almanac, U.S. Naval Observatory, http://aa.usno.navy.mil/publications/doc

Aoki, S., Soma, M., Kinoshita, H. & Inoue, K., Conversion matrix of epoch B 1950.0 FK 4-based positions of stars to epoch J 2000.0 positions in accordance with the new IAU resolutions, Astronomy & Astrophysics 128, 263-267, 1983.

Examples

```
# The star HD 119288 has
# RA(2000) = 13h 42m 12.740s Dec(2000) = 8d 23' 17.69''
#   Mu(RA) = -.0257 s/yr Mu(Dec) = -.090 ''/yr

mu_radec=100*c(-15*0.257, -0.090)
ra = ten(13, 42, 12.740)*15.
dec = ten(8,23,17.69)
bprecess(ra, dec, mu_radec)
```

---

calz_unred — *Deredden a galaxy spectrum using the Calzetti et al. (2000) recipe*

Description

Deredden a galaxy spectrum using the Calzetti et al. (2000) recipe

Usage

```
calz_unred(wave,flux,ebv,R_V)
```

Arguments

wave	wavelength in Angstroms, scalar or N-vector
flux	calibrated flux vector, scalar or N-vector
ebv	color excess E(B-V), scalar
R_V	ratio of total to selected extinction, defauly=4.05 (optional)

Details

Calzetti et al. (2000) developed a recipe for dereddening the spectra of galaxies where massive stars dominate the radiation output, valid between 0.12 to 2.2 microns. Reddening values are extrapolated between 0.12 and 0.0912 microns. Calzetti et al. (2000) estimate R_V = 4.05 +/- 0.80 from optical-IR observations of four starburst galaxies.

If a negative E(B-V) is supplied, then fluxes will be reddened rather than deredenned. Note that the supplied color excess should be that derived for the stellar continuum, *ebv*(stars), which is related to the reddening derived from the gas, *ebv*(gas), via the Balmer decrement by *ebv(stars) = 0.44*ebv(gas)*.

Output funred values will be zero outside the wavelength domain 0.0912 to 2.2 microns.

Value

`funred`	unreddened flux, scalar or N-vector.

Author(s)

Written W. Landsman Raytheon ITSS December, 2000

R adaptation by Arnab Chakraborty June 2013

References

Calzetti, D., Armus, L., Bohlin, R. C., Kinney, A. L., Koorneef, J. & Storchi-Bergmann, T., The dust content and opacity of actively star-forming galaxies, Astrophysical Journal, 533, 682-695, 2000. http://adsabs.harvard.edu/abs/2000ApJ...533..682C

See Also

`ccm_unred`

Examples

```
w <- 1200 + seq(50, 2000, by=50)  # wavelength vector
f <- rep(1, length(w))   # flat initial spectrum
calz_unred(w, f, ebv=0.1)
```

`ccm_unred` *Deredden a flux vector using the Cardelli et al. (1989) parameterization*

Description

Deredden a flux vector using the Cardelli et al. (1989) parameterization

Usage

```
ccm_unred(wave,flux,ebv,R_V)
```

Arguments

`wave`	wavelength in Angstroms, scalar or N-vector
`flux`	calibrated flux vector, scalar or N-vector
`ebv`	color excess E(B-V), scalar
`R_V`	ratio of total to selected extinction, default=3.1 (optional)

Details

The reddening curve is that of Cardelli, Clayton, and Mathis (1989), including the update for the near-UV given by O'Donnell (1994). Parameterization is valid from the IR to the far-UV (3.5 microns to 0.1 microns). Curve is extrapolated between 912 and 1000 A as suggested by Longo et al. (1989).

R_V specifies the ratio of total selective extinction R(V) = A(V) / E(B - V). If not specified, then R_V = 3.1 Extreme values of R(V) range from 2.75 to 5.3.

Many sightlines with peculiar ultraviolet interstellar extinction can be represented with a CCM curve, if the proper value of R(V) is supplied.

Users might wish to consider using the alternate procedure FM_UNRED which uses the extinction curve of Fitzpatrick (1999).

The CCM curve shows good agreement with the Savage and Mathis (1979) ultraviolet curve shortward of 1400 A, but is probably preferable between 1200 and 1400 A.

Valencic et al. (2004) revise the ultraviolet CCM curve (3.3 – 8.0 um-1). But since their revised curve does not connect smoothly with longer and shorter wavelengths, it is not included here.

Value

`funred`	unreddened flux, scalar or N-vector.

Author(s)

Written W. Landsman Hughes/STX January, 1992

R adaptation by Arnab Chakraborty June 2013

References

Cardelli, J. A., Clayton, G. C., Mathis, J. S., The relationship between infrared, optional, and ultraviolet extinction, Astrophysical Journal, 345, 245-256, 1989. http://adsabs.harvard.edu/abs/1989ApJ...345..245C

Fitzpatrick, E. D., Correcting the effects of interstellar extinction, Publ. Astron. Soc. Pacific, 111, 63-75, 1999. http://adsabs.harvard.edu/abs/1999PASP..111...63F

Longo, R., Stalio, R., Polidan, R. S., Rossi, L., Intrinsic ultraviolet (912-3200 A) energy distribution of OB stars, Astrophysical Journal 339, 474-487, 1989. http://adsabs.harvard.edu/abs/1989ApJ...339..474L

O'Donnell, J. E., R-nu-dependent optical and near-ultraviolet extinction, Astrophysical Journal, 422, 158-163, 1994. http://adsabs.harvard.edu/abs/1994ApJ...422..158O

Savage, B. D. & Mathis, J. S., Observed properties of interstellar dust, Ann. Rev. Astron. Astrophys. 17, 73-111, 1979. http://www.annualreviews.org/doi/pdf/10.1146/annurev.aa.17.090179.000445

Valencic, L. A., Clayton, G., C., Gordon, K. D., Ultraviolet extinction properties in the Milky Way, Astrophysical Journal., 616, 912-924, 2004. http://adsabs.harvard.edu/abs/2004ApJ...616..912V

See Also

calz_unred

Examples

```
w <- 1200 + seq(50, 2000, by=50)  # wavelength vector
f <- rep(1, length(w))   # flat initial spectrum
ccm_unred(w, f, ebv=0.1)
```

---

cirrange — *Force an angle into the range 0 <= ang < 360*

Description

Force an angle into the range 0 <= ang < 360

Usage

```
cirrange(ang, radians=FALSE)
```

Arguments

`ang`	angle, scalar or vector
`radians`	indicates that angle is specified in radians rather than decimal degrees (optional, default = FALSE)

Details

The input angle is transformed into the range 0 to +360 degrees (or 0 to 2*pi for radians). This function is used by several other **astrolib** functions, and is rarely used directly by the user.

Value

`ang`	transformed angle

Author(s)

Written by Michael R. Greason, Hughes STX, 10 February 1994

R adaptation by Arnab Chakraborty June 2013

Examples

```
new_ang <- cirrange(-40.)   # returns 320.
```

---

cosmo_param — *Derive full set of cosmological density parameters from a partial set*

Description

Derive full set of cosmological density parameters from a partial set

Usage

```
cosmo_param(omega_m, omega_lambda, omega_k, q0)
```

Arguments

`omega_m`	normalized matter energy density, non-negative scalar
`omega_lambda`	normalized cosmological constant, scalar
`omega_k`	normalized curvature parameter, scalar (= 0 for a flat universe
`q0`	deceleration parameter, scalar (= -R*(R")/(R')^2 = 0.5*omega_m - omega_lambda)

Details

This procedure is called by *lumdist* and *galage* to allow the user a choice in defining any two of four cosmological density parameters.

Given any two of the four input parameters – (1) the normalized matter density omega_m (2) the normalized cosmological constant, omega_lambda (3) the normalized curvature term, Omega_k and (4) the deceleration parameter q0 – this program will derive the remaining two. Here "normalized" means divided by the closure density so that omega_m + omega_lambda + omega_k = 1. For a more precise definition see Carroll et al (1992).

If fewer than two parameters are defined, this procedure sets default values of omega_k=0 (flat space), omega_lambda = 0.7, omega_m = 0.3 and q0 = -0.55

If more than two parameters are defined upon input (overspecification), then the first two defined parameters in the ordered list omega_m, omega_lambda, omega_k, q0 are used to define the cosmology.

Value

`omega_m`	normalized matter energy density, non-negative scalar
`omega_lambda`	normalized cosmological constant, scalar
`omega_k`	normalized curvature parameter, scalar (= 0 for a flat universe
`q0`	deceleration parameter, scalar (= -R*(R")/(R')^2 = 0.5*omega_m - omega_lambda)

Author(s)

Written by W. Landsman Raytheon ITSS April 2000

R adaptation by Arnab Chakraborty (June 2013)

References

Carroll, S. M., Press, W. H. & Turner, E. L., The cosmological constant, Ann. Rev. Astro. Astrophys, 30, 499, 1992. http://www.annualreviews.org/doi/pdf/10.1146/annurev.aa.30.090192.002435

Examples

```
cosmo_param(0.3, NULL, 1.0, NULL)  # returns omega_lambda=0.7 and q0=-1.05
```

---

co_aberration — *Calculate changes to right ascension and declination due to astronomical aberration*

Description

Calculate changes to right ascension and declination due to astronomical aberration

Usage

```
co_aberration(jd, ra, dec, eps)
```

Arguments

`jd`	Julian Date [scalar or vector]
`ra`	right ascension, in degrees (scalar or N-vector)
`dec`	declination, in degrees (scalar or N-vector)
`eps`	true obliquity of the ecliptic, in radians (optional),

Details

Algorithm described in Meeus (1991), Chap 23. Accuracy is much better than 1 arcsecond.

Value

`d_ra`	corrections to ra due to aberration (must then be added to ra to get corrected values).
`d_dec`	corrections to dec due to aberration (must then be added to dec to get corrected values).
`eps`	true obliquity of the ecliptic (in radians).

Author(s)

Original IDL program by Chris O'Dell, U. of Wisconsin (June 2002)

R adaptation by Arnab Chakraborty (June 2013)

References

Meeus, J., Astronomical Algorithms, Willmann-Bell, 1991

See Also

co_refract

Examples

```
co_aberration(2456469.5,253.215,-32.449)
```

---

co_nutate — *Calculate changes in right ascension and declination due to nutation of the Earth's rotation*

Description

Calculate changes in right ascension and declination due to nutation of the Earth's rotation

Usage

```
co_nutate(jd, ra, dec)
```

Arguments

jd	Julian Date [scalar or vector]
ra	right ascension, in degrees (scalar or N-vector)
dec	declination, in degrees (scalar or N-vector)

Details

Calculates necessary changes to right ascension and declination due to the nutation of the Earth's rotation axis, as described in Meeus (1991), Chap 23. Uses formulae from the Astronomical Almanac (1984) and does the calculations in equatorial rectangular coordinates to avoid singularities at the celestial poles.

Value

d_ra	corrections to ra due to aberration, in arcseconds (must then be added to ra to get corrected values).
d_dec	corrections to dec due to aberration, in arcseconds (must then be added to dec to get corrected values).
eps	true obliquity of the ecliptic, in radians
d_psi	nutation of the longitude of the ecliptic
d_eps	nutation in the obliquity of the ecliptic

Author(s)

Written Chris O'Dell (2002)

R adaptation by Arnab Chakraborty (June 2013)

References

Meeus, J., Astronomical Algorithms, Willmann-Bell, 1991

See Also

nutate

Examples

```
co_nutate(2456469.5,253.215,-32.449)
```

---

co_refract — *Calculate correction to altitude due to atmospheric refraction*

Description

Calculate correction to altitude due to atmospheric refraction

Usage

```
co_refract(a, altitude, pressure, temperature, to_observed, epsilon)
```

Arguments

a	observed (apparent) altitude, in degrees (scalar or vector)
altitude	height of the observing location, in meters. This is only used to determine an approximate temperature and pressure, if these are not specified separately. (default=0, i.e. sea level)
pressure	pressure at the observing location, in millibars (default = 1010)
temperature	temperature at the observation location, in Kelvin (default = 283)

`to_observed`	Flag to calculate from Apparent->Observed altitude, using the iterative technique (default = omitted gives a single estimate of the refraction correction)
`epsilon`	iteration accuracy (default = 0.25 arcseconds)

Details

Because the index of refraction of air is not precisely 1.0, the atmosphere bends all incoming light, making a star or other celestial object appear at a slightly different altitude (or elevation) than it really is. It is important to understand the following definitions:

Observed Altitude: The altitude that a star is SEEN to BE, with a telescope. This is where it appears in the sky. This is always GREATER than the apparent altitude.

Apparent Altitude: The altitude that a star would be at, if *there were no atmosphere* (sometimes called "true" altitude). This is usually calculated from an object's celestial coordinates. Apparent altitude is always LOWER than the observed altitude.

Thus, for example, the Sun's apparent altitude when you see it right on the horizon is actually -34 arcminutes.

This program uses couple simple formulae to estimate the effect for most optical and radio wavelengths. Typically, you know your observed altitude (from an observation), and want the apparent altitude. To go the other way, this program uses an iterative approach.

WAVELENGTH DEPENDENCE: This correction is 0 at zenith, about 1 arcminute at 45 degrees, and 34 arcminutes at the horizon FOR OPTICAL WAVELENGTHS. The correction is NON-NEGLIGIBLE at all wavelengths, but is not very easily calculable. These formulae assume a wavelength of 550 nm, and will be accurate to about 4 arcseconds for all visible wavelengths, for elevations of 10 degrees and higher. Amazingly, they are also ACCURATE FOR RADIO FREQUENCIES LESS THAN ~ 100 GHz.

It is important to understand that these formulae really can't do better than about 30 arcseconds of accuracy very close to the horizon, as variable atmospheric effects become very important.

Value

`aout`	observed (apparent) altitude, in degrees

Author(s)

Written Chris O'Dell

R adaptation by Arnab Chakraborty (June 2013)

References

Meeus, J., Astronomical Algorithms, Chapter 15.

Explanatory Supplement to the Astronomical Almanac, 1992.

Methods of Experimental Physics, Vol 12 Part B, Astrophysics, Radio Telescopes, Chapter 2.5, "Refraction Effects in the Neutral Atmosphere", by R.K. Crane.

See Also

`co_refract_forward` `co_aberration`

Examples

```
co_refract(0.5)
```

`co_refract_forward` *Calculate the true altitude of a celestial object from an observed altitude.*

Description

Calculate the true altitude of a celestial object from an observed altitude. This function is called by *co_refract*, the function to correct altitude due to atmospheric refraction.

Usage

```
co_refract_forward(a, p = 1010, t = 283)
```

Arguments

`a`	The observed ('apparent') altitude, in degrees
`p`	Atmospheric pressure, in millibars (default=1010)
`t`	Ground temperature, in degrees Celsius (default=0)

Details

See description for *co_refract*.

Value

`R`	Correction to apparent altitude, subtracted from a, in degrees

Note

The IDL documentation may be incorrect: the input variable t appears to be in degrees Kelvin. The value t=0 is converted to 283, but other values should be in Kelvin.

Author(s)

Chris O'Dell 2002

Arnab Chakraborty R version 2013

See Also

`co_refract`

Examples

```
co_refract_forward (30)
```

ct2lst *Convert from Local Civil Time to Local Mean Sidereal Time*

Description

Convert from Local Civil Time to Local Mean Sidereal Time

Usage

```
ct2lst(lng, tz, tme, day, mon, year)
```

Arguments

`lng`	The longitude in degrees (east of Greenwich) of the place for which the local sidereal time is desired, scalar. Greenwich mean sidereal time (GMST) can be found by setting lng = 0.
`tz`	The time zone of the site in hours, positive East of the Greenwich meridian (ahead of GMT). Use this parameter to easily account for Daylight Savings time (e.g. -4=EDT, -5 = EST/CDT). This scalar parameter is not needed (and ignored) if Julian date is supplied. Note that the sign of TZ was changed in July 2008 to match the standard definition.
`tme`	If more than three parameters are specified, then this is the time of day of the specified date in decimal hours. If exactly three parameters are specified, then this is the Julian date of time in question, scalar or vector
`day`	The day of the month (1-31),integer scalar or vector
`mon`	The month, in numerical format (1-12), integer scalar or vector
`year`	The 4 digit year (e.g. 2008), integer scalar or vector

Details

The Julian date of the day and time is question is used to determine the number of days to have passed since 0 Jan 2000. This is used in conjunction with the GST of that date to extrapolate to the current GST; this is then used to get the LST. See Meeus (1991), p.84, equation 11-4. The Web site `http://tycho.usno.navy.mil/sidereal.html` contains more information on sidereal time, as well as an interactive calculator.

Value

`lst`	Local Sidereal Tme for the date/time specified in hours

Author(s)

Adapted from the FORTRAN program GETSD by Michael R. Greason, STX, 1988

R adaptation by Arnab Chakraborty (June 2013)

References

Meeus, J., Astronomical Algorithms, Willmann-Bell, 1991

See Also

jdcnv

Examples

```
lng_Balt <- 76.72   # Baltimore MD
ct2lst(lng_Balt, -4, 15.3, 30, 07, 2008)
```

---

daycnv *Convert Julian dates to Gregorian calendar dates*

Description

Convert Julian dates to Gregorian calendar dates

Usage

```
daycnv(xjd)
```

Arguments

xjd	Julian Date [scalar or vector]

Details

If the input xjd is a vector, then the outputs (yr,mn,day, and hr) will be vectors of the same length.

Uses the algorithm of Fliegel and Van Flandern (1968) as reported in the "Explanatory Supplement to the Astronomical Almanac" (1992), p. 604 Works for all Gregorian calendar dates with XJD > 0, i.e., dates after -4713 November 23. Be sure that the Julian date is specified as double precision to maintain accuracy at the fractional hour level.

Other conversions and manipulations of Julian dates are provided by the CRAN packages *chron* and *dates*.

Value

yr	year (integer)
mn	month (integer)
day	day (integer)
hr	hours and fractional hours (real)

Author(s)

Converted to IDL from Yeoman's Comet Ephemeris Generator, B. Pfarr, STX, 1988

R adaptation by Arnab Chakraborty (June 2013)

Examples

```
daycnv(2440000.0)  # 1200 on May 23 1968
```

---

deredd — *Deredden stellar Stromgren parameters given for a value of E(b-y)*

Description

Deredden stellar Stromgren parameters given for a value of E(b-y)

Usage

```
deredd(eby,by,m1,c1,ub)
```

Arguments

`eby`	color index E(b-y), E(b-y) = 0.73*E(B-V), scalar
`by`	b-y color (observed)
`m1`	Stromgren line blanketing parameter (observed)
`c1`	Stromgren Balmer discontinuity parameter (observed)
`ub`	u-b color (observed)

Details

This function is used by *uvbybeta* where more information is provided.

Value

`by0`	b-y color (dereddened)
`m0`	Line blanketing index (dereddened)
`c0`	Balmer discontinuity parameter (dereddened)
`ub0`	u-b color (dereddened)

Author(s)

Adapted from FORTRAN routine DEREDD by T.T. Moon R adaptation by Arnab Chakraborty (June 2013)

Examples

```
deredd(0.4,0.2,1.0,1.0,0.1)
```

dtdz — *Integrand for cosmic age vs. redshift in standard cosmology*

Description

Used by *galage*

Usage

```
dtdz(z,lambda0,q0)
```

Arguments

z	measured redshift, scalar or vector
lambda0	cosmological constant normalized to the closure density
q0	cosmological deceleration parameter

Details

See *galage*.

Author(s)

R adaptation by Arnab Chakraborty (June 2013)

See Also

galage

Examples

```
dtdz(1.5, 0,0.5)
```

eci2geo — *Convert Earth-centered inertial coordinates to geographic spherical coordinates*

Description

Convert Earth-centered inertial coordinates to geographic spherical coordinates

Usage

```
eci2geo(eci_xyz, jdtim)
```

Arguments

eci_xyz	ECI [X,Y,Z] coordinates (in km), can be an array [3,n] of n such coordinates.
jdtim	Julian Day time, double precision. Can be a 1-D array of n such times.

Details

Converts from ECI (Earth-Centered Inertial) (X,Y,Z) rectangular coordinates to geographic spherical coordinates (latitude, longitude, altitude). JD time is also needed as input.

ECI coordinates are in km from Earth center. Geographic coordinates are in degrees/degrees/km. Geographic coordinates assume the Earth is a perfect sphere, with radius equal to its equatorial radius.

gcoord can be further transformed into geodetic coordinates (using **astrolib** *geo2geodetic*) or into geomagnetic coordinates (using *geo2mag*)

Value

lat, lon, alt	3-element array of geographic [latitude,longitude,altitude], or an array [3,n] of n such coordinates, double precision

Author(s)

Written by Pascal Saint-Hilaire (Saint-Hilaire@astro.phys.ethz.ch) on 2001/05/13

R adaptation by Arnab Chakraborty (June 2013)

See Also

ct2lst

Examples

```
eci2geo(c(6378.137+600, 0, 0), 2452343.38982663)  # result 0.0000000  232.27096 600.00000
```

---

eq2hor — *Convert celestial (ra-dec) coords to local horizon coords (alt-az)*

Description

Convert celestial (ra-dec) coords to local horizon coords (alt-az)

Usage

```
eq2hor(ra, dec, jd, lat, lon, ws, obsname, b1950, precess_, nutate_, refract_,
       aberration_, altitude, ...)
```

Arguments

ra	Right Ascension of object (J2000), in degrees (FK5), scalar or vector
dec	declination of object (J2000), in degrees (FK5), scalar or vector
jd	Julian Date, scalar or vector
lat	north geodetic latitude of location, in degrees (default = 43.0783)
lon	east longitude of location, in degrees. Specify west longitude with a negative sign. (default = -89.865)
ws	set to TRUE for azimuth measured westward from south, rather than East of North. (default = FALSE)
obsname	set this to a valid observatory name to be used by the astrolib OBSERVATORY procedure, which will return the latitude and longitude to be used by this program. (default = NULL)
b1950	set to TRUE if ra and dec are specified in B1950 FK4 coordinates, instead of J2000 FK5. (default = FALSE)
precess_	set to 1 to force precession, 0 for no precession correction. (default = 1)
nutate_	set to 1 to force nutation, 0 for no nutation. (default = 1)
refract_	set to 1 to force refraction correction, 0 for no correction. (default = 1)
aberration_	set to 1 to force aberration correction, 0 for no correction. (default = 1)
altitude	altitude of the observing location, in meters. (default=0)
...	may include setting temperature and pressure explicitly used by astrolib co_refract to calculate the refraction effect of the atmosphere.

Details

Calculates horizon (alt,az) coordinates from equatorial (ra,dec) coords. It is typically accurate to about 1 arcsecond or better (checked against the publicly available XEPHEM software). It performs precession, nutation, aberration, and refraction corrections. Array inputs are permitted (except lat, lon, and altitude).

If RA and DEC are arrays, then alt and az will also be arrays. If RA and DEC are arrays, JD may be a scalar OR an array of the same dimensionality.

The code has the following steps: (a) Apply refraction correction to find apparent Alt. (b) Calculate Local Mean Sidereal Time. (c) Calculate Local Apparent Sidereal Time. (d) Spherical trigonometry to find apparent hour angle, declination. (e) Calculate Right Ascension from hour angle and local sidereal time. (f) Nutation Correction to Ra-Dec. (g) Aberration correction to Ra-Dec. (h) Precess Ra-Dec to current equinox.

The following corrections are not made: (i) Deflection of Light by the sun due to General Relativity (typically milliarcseconds, but can be arseconds within one degree of the sun). (j) Effect of annual parallax (typically < 1 arcsecond). (k) Improved refraction correction with wavelength dependence and integration through the atmosphere). (l) Topocentric parallax correction accounting for elevation of the observatory. (m) Proper motion. (n) Difference between UTC and UT1 in determining LAST. (o) Polar motion. (p) Improved connection to Julian Date Calculator.

Value

`alt`	altitude, in degrees
`az`	azimuth angle measured EAST from NORTH (but see input ws above), in degrees
`ha`	hour angle (optional), in degrees

Author(s)

Written Chris O'Dell, Univ. of Wisconsin-Madison

R adaptation by Arnab Chakraborty (June 2013)

See Also

nutate precess sunpos co_nutate co_aberration co_refract altaz2hadec hadec2altaz

Examples

```
# Find the position of the open cluster NGC 2264 at the Effelsburg Radio
# Telescope in Germany, on June 11, 2023, at local time 22:00 (METDST).
# The inputs will then be:
#   Julian Date = 2460107.250
#   Latitude = 50d 31m 36s
#   Longitude = 06h 51m 18s
#   Altitude = 369 meters
#   RA (J2000) = 06h 40m 58.2s
#   Dec(J2000) = 09d 53m 44.0s

eq2hor(ten(6,40,58.2)*15., ten(9,53,44), 2460107.250,
       lat=ten(50,31,36), lon=ten(6,51,18), altitude=369.0,
       pres=980.0, temp=283.0)

# Output expected:
# Az = 17 42 25.6
# El = +16 28 22.8
# Hour Angle = +05 04 27.6
```

---

eqpole — *Convert Right Ascension and declination to X,Y using an equal-area polar projection*

Description

Convert Right Ascension and declination to X,Y using an equal-area polar projection

Usage

```
eqpole(l, b, southpole)
```

Arguments

`l`	longitude, in degrees, scalar or vector
`b`	latitude, in degrees, same number of elements as longitude
`southpole`	Set to TRUE if plot is centered on the south pole instead of the north pole (optional, default = FALSE)

Details

The output X and Y coordinates are scaled to be between -90 and +90 to go from equator to pole to equator. Output map points can be centered on the north pole or south pole.

Several polar equal-area map projections are provided by the CRAN package *mapproj*.

Value

`X`	X coordinate, ranging from -90 to +90, same number of elements as longitude, in degrees
`Y`	Y coordinate, ranging from -90 to +90, same number of elements as longitude, in degrees

Author(s)

Written by J. Bloch LANL, SST-9 1.1 5/16/91

R adaptation by Arnab Chakraborty (June 2013)

Examples

```
eqpole(100, -20.)
```

`euler` *Transform between Galactic, celestial, and ecliptic coordinates*

Description

Transform between Galactic, celestial, and ecliptic coordinates

Usage

```
euler(ai, bi, select, fk4, radian)
```

Arguments

ai	longitude, in degrees unless radian=TRUE is set, scalar or vector
bi	latitude, degrees unless radian=TRUE is set, scalar or vector
select	integer (1-6) specifying type of coordinate transformation: 1 = RA-Dec (J2000) to Galactic; 2 = Galactic to RA-Dec; 3 = RA-Dec (J2000) to Ecliptic; 4 = Ecliptic to RA-Dec; 5 = Ecliptic to Galactic; 6 = Galactic to Ecliptic
fk4	set to TRUE if input and output celestial and equatorial coordinates are given in equinox B1950 (default = FALSE)
radian	set to TRUE in all input and output angles are in radians rather than degrees (default = FALSE)

Details

J2000 coordinate conversions are based on sec. 1.2 of Perryman (1997).

Related functions are provided in CRAN packages *moonsun* and *sphereplot*.

Value

ao	longitude, in degrees
bo	latitude, in degrees

Author(s)

Written by W. Landsman, February 1987; adapted from Fortran by Daryl Yentis NRL

R adaptation by Arnab Chakraborty (June 2013)

References

Perryman, M. (editor) 1997, The Hipparcos and Tycho Catalogues, Vol. 1, ESA SP-1200. http://www.rssd.esa.int/SA/HIPPARCOS/docs/vol1_all.pdf

Examples

```
# Input: RA and dec of Cyg X-1
# Output: Galactic long and lat  = (71.334990, 3.0668335)

euler(299.590315, 35.201604, 1)
```

`flux2mag` *Convert from flux (ergs/s/cm^2/A) to astronomical magnitudes*

Description

Convert from flux (ergs/s/cm^2/A) to astronomical magnitudes

Usage

```
flux2mag(flux, zero_pt=21.10, ABwave)
```

Arguments

`flux`	flux, in erg cm-2 s-1 A-1, scalar or vector
`zero_pt`	zero point level of the magnitude (default = 21.1)
`ABwave`	wavelength for conversion to Oke AB magnitudes, in Angstroms (optional)

Details

The default zero point of 21.1 mag is from Code et al. (1976). It is ignored of the ABwave parameter is supplied.

If ABwave is not supplied, the routine returns magnitudes given by the expression

`mag <- -2.5*log10(flux) - zero_pt`.

If ABwave is supplied, then the routine returns AB magnitudes from Oke and Gunn (1983) according to `abmag <- -2.5*log10(flux) - 5*log10(ABwave) - 2.406`.

Use *mag2flux* for conversions in the opposite direction.

Value

`mag`	magnitude, scalar or vector

Author(s)

Converted to IDL from Yeoman's Comet Ephemeris Generator, B. Pfarr, STX, 1988

R adaptation by Arnab Chakraborty (June 2013)

References

Oke, J. B. and Gunn, J. E., 1983, Secondary standard stars for absolute spectrophotometry, Astrophysical Journal, 266, 713-717

Examples

```
flux2mag(3e-17)   #  returns 20.21

ytext <- expression(Flux~~ (erg/s~cm^2~A))
plot(seq(2000,5000,length=100), flux2mag(seq(3e-17,3e-16,length=100)), pch=20)
```

`fm_unred` *Deredden a flux vector using the Fitzpatrick (1999) parameterization*

Description

Deredden a flux vector using the Fitzpatrick (1999) parameterization

Usage

```
fm_unred(wave, flux, ebv, r_v = 3.1, avglmc=FALSE, lmc2=FALSE, x0=NULL,
gamma=NULL, c4=NULL, c3=NULL, c2=NULL, c1=NULL)
```

Arguments

`wave`	wavelength in Angstroms, scalar or N-vector
`flux`	calibrated flux vector, scalar or N-vector
`ebv`	color excess E(B-V), scalar
`r_v`	ratio of total to selected extinction, (optional, default=3.1
`avglmc`	if set to TRUE, then the default fit parameters (c1,c2,c3,c4,gamma,x0) are set to the average values determined for reddening in the general Large Magellanic Cloud (LMC) field by Misselt et al. (1999), (optional)
`lmc2`	if set to TRUE, then the fit parameters are set to the values determined for the LMC2 field (including 30 Dor) by Misselt et al. Note that neither /AVGLMC or /LMC2 will alter the default value of R_V which is poorly known for the LMC.
`x0`	Centroid of 2200 A bump in microns (optional, default = 4.596)
`gamma`	Width of 2200 A bump in microns (optional, default = 0.99)
`c4`	FUV curvature (optional, default = 0.41)
`c3`	Strength of the 2200 A bump (optional, default = 3.23)
`c2`	Slope of the linear UV extinction component (optional, default = -0.824 + 4.717/R)
`c1`	Intercept of the linear UV extinction component (optional, default = 2.030 - 3.007*c2)

Details

The Galactic extinction curve is given by Fitzpatrick & Massa (FM, 1999). Parameterization is valid from the infrared to the far-ultraviolet (3.5 microns to 0.1 microns). The ultraviolet extinction curve is extrapolated down to 912 Angstroms. Many sightlines with peculiar ultraviolet interstellar extinction can be represented with the FM curve, if the proper value of R(V) is supplied.

When the inputs omit extinction curve parameters (x0, gamma, c4, c3, c2, and c1) then the default extinction curve is adopted from Clayton et al. (2003).

The parameter *R_V* specifies the ratio of total to selective extinction, `R(V) = A(V) / E(B - V)`. Extreme values of R(V) range from 2.3 to 5.3, and the default is 3.1.

If a negative *ebv* is supplied, then fluxes will be reddened rather than dereddened.

The following comparisons between the FM curve and that of Cardelli, et al. (CCM, 1989) have been made (see function *ccm_unred*):
a) In the ultraviolet, the FM and CCM curves are similar for R < 4.0, but diverge for larger R
b) In the optical region, the FM more closely matches the monochromatic extinction, especially near the R band

Value

`extcurve`	`E(wavelength - V)/E(B-V)` extinction curve, interpolated onto the input wavelength vector

Author(s)

Written W. Landsman Raytheon STX October, 1998

R adaptation by Arnab Chakraborty June 2013

References

Cardelli, J.A., Clayton, G. C., Mathis, J. S., 1989, The relationship between infrared, optical, and ultraviolet extinction, Astrophys. J. 345, 245-256. http://adsabs.harvard.edu/abs/1989ApJ...345..245C

Clayton, G. C., Wolff, M. J., Sofia, U. J., Gordon, K. D., Misselt, K. A., 2003, Dust grain size distributions from MRN to MEM, Astrophys. J., 588, 871-880. http://adsabs.harvard.edu/abs/2003ApJ...588..871C

Fitzpatrick, E. L., 1999, Correcting the effects of interstellar extinction, Publ. Astron. Soc. Pacific, 111, 63-75. http://adsabs.harvard.edu/abs/1999PASP..111...63F

Misselt, K. A., Clayton, G. C., Gordon, K. D., 1999, A reanalysis of the ultraviolet extinction from interstellar dust in the Large Magellanic Cloud, Astrophys. J. 515, 128-139. http://adsabs.harvard.edu/abs/1999ApJ...515..128M

See Also

`polyidl` `spline`

Examples

```
w <- 1200 + seq(50, 2000, by=50)  # wavelength vector
f <- rep(1, length(w))   # flat initial spectrum
fm_unred(w, f, ebv=0.1)
```

---

galage — *Determine the age of a galaxy given its redshift and a formation redshift*

Description

Determine the age of a galaxy given its redshift and a formation redshift

Usage

```
galage(z, zform, h0=70.0, omega_m, lambda0, k,q0, silent=FALSE)
```

Arguments

`z`	measured redshift, scalar or vector
`zform`	redshift of galaxy formation, scalar or vector
`h0`	Hubble constant, in km/s/Mpc (default = 70)
`omega_m`	cosmological matter density normalized to the closure density (default = 0.3)
`lambda0`	cosmological constant normalized to the closure density (default = 0.7)
`k`	cosmological curvature constant (default = 0.0 for a flat Universe)
`q0`	cosmological deceleration parameter (default = -0.55)
`silent`	if =TRUE, adopted cosmological parameters are not displayed (default=FALSE)

Details

For a given formation time *zform* and a measured *z*, this procedure integrates dt/dz from *zform* to *z* using the analytic formula of Gardner (1998, eqn. 7). The integration is implemented by the R function *integrate*.

The value of assumed formation redshift, *zform*, must exceed the measured redshift *z*. To determine the age of the universe at a given redshift, set *zform* to a large number (e.g. 1000).

Value

`age`	age of galaxy, scalar or vector, in years

Author(s)

Written by W. Landsman Raytheon ITSS April 2000

R adaptation by Arnab Chakraborty (June 2013)

References

Gardner, J. P., 1998, An extendable galaxy number count model, Publ. Astron. Soc. Pacific, 11, 291-305. http://adsabs.harvard.edu/abs/1998PASP..110..291G

See Also

integrate cosmo_param dtdz

Examples

```
galage(1.5, 25., omega_m=0.3, lambda0=0.0)   #  result: 3.35 Gyr
```

---

gal_uvw — *Calculate the Galactic space velocity (U,V,W) of a star*

Description

Calculate the Galactic space velocity (U,V,W) of a star

Usage

```
gal_uvw(distance, lsr=F, ra, dec, pmra, pmdec, vrad, plx)
```

Arguments

`distance`	distance, scalar or vector, in parsecs
`lsr`	if =TRUE, then output velocities are corrected for solar motion
`ra`	Right Ascension, scalar or vector, in decimal degrees
`dec`	declination, scalar or vector, in decimal degrees
`pmra`	proper motion in R.A., scalar or vector, in milliarcseconds/yr
`pmdec`	proper motion in declination, scalar or vector, in milliarcseconds/yr
`vrad`	radial velocity, scalar or vector, in km/s
`plx`	parallax, scalar or vector, in milliarcseconds

Details

Calculates the Galactic space velocity U, V, W of star given its coordinates, proper motion, distance (or parallax), and radial velocity. ; The calculation follows the general outline of Johnson & Soderblom (1987) except that U is positive outward toward the Galactic anticenter, and the J2000 transformation matrix to Galactic coordinates is taken from the introduction to the Hipparcos catalog.

The *distance* parameter should be zero if parallax *plx* is provided. If the user has proper motion in R.A. given in seconds of time/yr, mu_alpha, then it should first be converted to seconds of time/yr using

```
pmra = 15*mu_alpha*cos(dec)
```

.

If *lsr*=TRUE, then output velocities are corrected to the local standard of rest (LSR) assuming a solar motion (U,V,W)_Sun = (-8.5, 13.38, 6.49) km/s, as given by Coskunoglu et al. 2011). Note that the value of the solar motion through the LSR remains poorly determined.

Value

U	velocity positive toward the Galactic anticenter, in km/s
V	velocity positive in the direction of Galactic rotation, in km/s
W	velocity positive toward the North Galactic Pole, in km/s

Author(s)

Written by W. Landsman, December 2000

R adaptation by Arnab Chakraborty (June 2013)

References

Coskunoglu, B., Ak, S., Bilir, S., et al. 2011, Local stellar kinematics from the RAVE data. I. Local standard of rest, Mon. Not. Royal Astron. Soc., 412, 1237-1245. http://adsabs.harvard.edu/abs/2011MNRAS.412.1237C

Johnson, D. R. H. and Soderblom, D. R., 1987, Calculating galactic space velocities and their uncertainties, with an application to the Ursa Major group, Astron. J., 93, 864-867. http://adsabs.harvard.edu/abs/1987AJ.....93..864J

Examples

```
#  Compute (U,V,W) for the halo star HD 6755 using measurments by the Hipparcos satellite
#  Result: u=141.2  v = -491.7  w = 93.9        ;km/s

gal_uvw(139, lsr=TRUE, ten(1,9,42.3)*15., ten(61,32,49.5), 628.42, 76.65, -321.4)
```

---

gcirc — *Computes rigorous great circle arc distances between points on the celestial sphere*

Description

Computes rigorous great circle arc distances between points on the celestial sphere

Usage

```
gcirc(u,ra1,dc1,ra2,dc2)
```

Arguments

u	indicator integer describing units of inputs and outputs: 0: radians 1: Right Ascension in decimal hours, declination in decimal degrees, angular distance in arc seconds 2: Right Ascension and declination in decimal degrees, angular distance in arc seconds
ra1	Right Ascension or longitude of point 1
dc1	declination or latitude of point 1
ra2	Right Ascension or longitude of point 2
dc2	declination or latitude of point 2

Details

Input position can be in radians, sexigesimal (R.A., Dec), or decimal degrees. The procedure uses the Haversine forumla http://en.wikipedia.org/wiki/Great-circle_distance. The haversine formula can give rounding errors for antipodal points.

If (ra1,dc1) are scalars and (ra2,dc2) are vectors, then *dis* is a vector giving the distance of each element of (ra2,dc2) to (ra1,dc1). Similarly, if (ra1,dc1) are vectors and (ra2,dc2) are scalars, then *dis* is a vector giving the distance of each element of (ra1,dc1) to (ra2,dc2). If both (ra1,dc1) and (ra2,dc2) are vectors then *dis* is a vector giving the distance of each element of (ra1,dc1) to the corresponding element of (ra2,dc2). If the input vectors are not the same length, then excess elements of the longer ones will be ignored.

The **astrolib** function *sphdist* provides an alternate method of computing a spherical distance.

Value

dis	angular distance on the sky between points 1 and 2

Author(s)

Written in Fortran by R. Hill, SASC Technologies, January 1986

R adaptation by Arnab Chakraborty June 2013

See Also

sphdist

Examples

```
gcirc(2, 100., -35., 180., +35)
```

geo2eci — *Convert geographic spherical coordinates to Earth-centered inertial coordinates*

Description

Convert geographic spherical coordinates to Earth-centered inertial coordinates

Usage

```
geo2eci(incoord, jdtim)
```

Arguments

`incoord`	a 3-element vector of geographic coordinates [latitude, longitude, altitude], or an array [3,n] of n such coordinates, in degrees/degrees/km
`jdtim`	Julian Day time, in days, scalar or vector

Details

Converts from geographic spherical coordinates [latitude, longitude, altitude] to ECI (Earth-Centered Inertial) [X,Y,Z] rectangular coordinates. Geographic coordinates assume the Earth is a perfect sphere, with radius equal to its equatorial radius.

Value

`(x, y, z)`	a 3-element vector of ECI coordinates, or an array [3,n] of coordinates, in km from Earth center

Author(s)

Written by Pascal Saint-Hilaire (ETH) on 2002/05/14

R adaptation by Arnab Chakraborty June 2013

See Also

`ct2lst`

Examples

```
# Obtain the ECI coordinates of the intersection of the equator and Greenwich's meridian
# on 2002/03/09 21:21:21.021
# Returns: -3902.9606 5044.5548  0.0000
geo2eci(c(0,0,0), 2452343.38982663)
```

geo2geodetic — *Convert from geographic/planetographic to geodetic coordinates*

Description

Convert from geographic/planetographic to geodetic coordinates

Usage

```
geo2geodetic(gcoord, planet, equatorial_radius, polar_radius)
```

Arguments

- `gcoord` — a 3-element vector of geographic coordinates [latitude, longitude, altitude], or an array [3,n] of n such coordinates
- `planet` — keyword or integer specifying planet (see details, default = earth)
- `equatorial_radius` — equatorial radius of chosen planet, in km. If not set, the *planet* value is used.
- `polar_radius` — polar radius of chosen planet, in km. If not set, the *planet* value is used.

Details

Converts from geographic (latitude, longitude, altitude) to geodetic (latitude, longitude, altitude). In geographic coordinates, the Earth is assumed a perfect sphere with a radius equal to its equatorial radius. The geodetic (or ellipsoidal) coordinate system takes into account the Earth's oblateness. The method is from Keeper and Nievergelt (1998) with planetary constants from Allen (2000).

Geographic and geodetic longitudes are identical. Geodetic latitude is the angle between local zenith and the equatorial plane. Geographic and geodetic altitudes are both the closest distance between the satellite and the ground.

The *planet* input is either as an integer (1-9) or one of the (case-independent) strings 'mercury','venus','earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune', or 'pluto'.

The *equitorial_radius* and *polar_radius* inputs allow the transformation for any ellipsoid.

Whereas the conversion from geodetic to geographic coordinates is given by an exact, analytical formula, the conversion from geographic to geodetic is not. Approximative iterations (as used here) exist, but tend to become less accurate with increasing eccentricity and altitude. The formula used in this routine should give correct results within six digits for all spatial locations, for an ellipsoid (planet) with an eccentricity similar to or less than Earth's. More accurate results can be obtained via calculus, needing a non-determined amount of iterations.

Author(s)

Written by Pascal Saint-Hilaire (ETH) and Robert L. Marcialis (LPL), 2002

R adaptation by Arnab Chakraborty June 2013

References

Allen (ed.) 2000, "Astrophysical Quantities", 4th ed.

Keeler, S. P. and Nievergelt, Y., 1998, Computing geodetic coordinates", SIAM Rev., 40, 300-309.

See Also

ct2lst

Examples

```
# Locate the geographic North pole for Earth, altitude 0., in geodetic coordinates
# Returns: 90.000000       0.0000000       21.385000
geo2geodetic(c(90.0,0.0,0.0) )

# As above, but for the case of Mars
# Returns:  90.000000       0.0000000       18.235500
geo2geodetic(c(90.0,0.0,0.0), 'mars')
```

`geodetic2geo` *Convert from geodetic (or planetodetic) to geographic coordinates*

Description

Convert from geodetic (or planetodetic) to geographic coordinates

Usage

```
geodetic2geo(ecoord, planet, equatorial_radius, polar_radius)
```

Arguments

ecoord	a 3-element array of geodetic [latitude,longitude,altitude], or an array [3,n] of n such coordinates.
planet	keyword or integer specifying planet (see details, default = earth)
equatorial_radius	equatorial radius of chosen planet, in km. If not set, the *planet* value is used.
polar_radius	polar radius of chosen planet, in km. If not set, the *planet* value is used.

Details

Converts from geodetic (latitude, longitude, altitude) to geographic (latitude, longitude, altitude) coordinates. In geographic coordinates, the Earth is assumed a perfect sphere with a radius equal to its equatorial radius. The geodetic (or ellipsoidal) coordinate system takes into account the Earth's oblateness. The method is from Keeper and Nievergelt (1998) with planetary constants from Allen (2000).

Geographic and geodetic longitudes are identical. Geodetic latitude is the angle between local zenith and the equatorial plane. Geographic and geodetic altitudes are both the closest distance between the satellite and the ground.

The *planet* input is either as an integer (1-9) or one of the (case-independent) strings 'mercury','venus','earth', 'mars', 'jupiter', 'saturn', 'uranus', 'neptune', or 'pluto'.

The *equitorial_radius* and *polar_radius* inputs allow the transformation for any ellipsoid.

Whereas the conversion from geodetic to geographic coordinates is given by an exact, analytical formula, the conversion from geographic to geodetic is not. Approximative iterations (as used here) exist, but tend to become less accurate with increasing eccentricity and altitude. The formula used in this routine should give correct results within six digits for all spatial locations, for an ellipsoid (planet) with an eccentricity similar to or less than Earth's. More accurate results can be obtained via calculus, needing a non-determined amount of iterations.

Value

`gcoord` a 3-element vector of geographic coordinates [latitude, longitude, altitude], or an array [3,n] of n such coordinates

Author(s)

Written by Pascal Saint-Hilaire (ETH) on 2002

R adaptation by Arnab Chakraborty June 2013

Examples

```
# Convert North Pole, zero altitude, to geographic coordinates
# Results: 90.000000       0.0000000      -21.385000
geodetic2geo(c(90,0,0))

# Same calculation but for Mars
# Results: 90.000000       0.0000000      -18.235500
geodetic2geo(c(90,0,0),'mars')
```

`glactc` *Convert between celestial and Galactic (or Supergalactic) coordinates*

Description

Convert between celestial and Galactic (or Supergalactic) coordinates

Usage

```
glactc(ra, dec, year, gl, gb, j, degree=FALSE, fk4 = FALSE, supergalactic = FALSE)
```

Arguments

- `ra` Right Ascension (j=1) or Galactic longitude (j=2), in decimal hours or degrees, scalar or vector
- `dec` Declination (j=1) or Galactic latitude (j=2), in degrees, scalar or vector
- `year` equinox of ra and dec, scalar
- `gl` Galactic longitude or Right Ascension, in degrees, scalar or vector
- `gb` Galactic latitude or Declination, in degrees, scalar or vector
- `j` integer indicator, direction of conversion
  1: ra,dec –> gl,gb
  2: gl,gb –> ra,dec
- `degree` if set, then the RA parameter (both input and output) is given in degrees rather than hours (default=FALSE)
- `fk4` if set, then the celestial (RA, Dec) coordinates are assumed to be input/output in the FK4 system. By default, coordinates are assumed to be in the FK5 system. (default=FALSE)
- `supergalactic` if set, the function returns SuperGalactic coordinates (see details). (default=FALSE)

Details

If *j=1*, this function converts proper motion in equatorial coordinates (ra,dec) to proper motion in Galactic coordinates (gl, gb) or Supergalactic Coordinates (sgl,sgb). If *j=2*, the conversion is reversed from Galactic/Supergalactic coordinates to equatorial coordinates. The calculation includes precession on the coordinates.

For B1950 coordinates, set *fk4=TRUE* and *year=1950*.

If *supergalactic=TRUE* is set, Supergalactic coordinates are defined by de Vaucouleurs et al. (1976) to account for the local supercluster. The North pole in Supergalactic coordinates has Galactic coordinates l = 47.47, b = 6.32, and the origin is at Galactic coordinates l = 137.37, b = 0.00.

Value

- `ra` Galactic longitude (j=1) or Right Ascension (j=2), in decimal hours or degrees, scalar or vector
- `dec` Galactic latitude (j=1) or Declination (j=2), in degrees, scalar or vector

Author(s)

FORTRAN subroutine by T. A. Nagy, 1978. Conversion to IDL, R. S. Hill, STX, 1987.

R adaptation by Arnab Chakraborty June 2013

See Also

`precess` `jprecess` `bprecess`

Examples

```
# Find the Galactic coordinates of Altair (RA (J2000): 19 50 47 Dec (J2000): 08 52 06)
# Result: gl = 47.74, gb = -8.91

glactc(ten(19,50,47), ten(8,52,6), 2000, gl, gb, 1)
```

---

glactc_pm — *Convert between celestial and Galactic (or Supergalactic) proper motion and coordinates*

Description

Convert between celestial and Galactic (or Supergalactic) proper motion and coordinates

Usage

```
glactc_pm(ra, dec, mu_ra, mu_dec, year, gl, gb, mu_gl, mu_gb, j, degree=FALSE,
fk4 = FALSE, supergalactic = FALSE, mustar=FALSE)
```

Arguments

`ra`	Right Ascension, in decimal hours (or decimal degrees if *degree* is set), scalar or vector
`dec`	declination, in decimal degrees, scalar or vector
`mu_ra`	Right Ascension proper motion, in any proper motion unit (angle/time), scalar or vector
`mu_dec`	declination proper motion, in any proper motion unit (angle/time), scalar or vector
`year`	equinox of ra and dec, scalar
`gl`	Galactic longitude, decimal degrees, scalar or vector
`gb`	Galactic latitude, decimal degrees, scalar or vector
`mu_gl`	Galactic longitude proper motion, in any proper motion unit (angle/time), scalar or vector
`mu_gb`	Galactic latitude proper motion, in any proper motion unit (angle/time), scalar or vector
`j`	integer indicator, direction of conversion 1: ra,dec,mu_ra,mu_dec –> gl,gb,mu_gl,mu_gb 2: gl,gb,mu_gl,mu_gb –> ra,dec,mu_ra,mu_dec
`degree`	if set, then the RA parameter (both input and output) is given in degrees rather than hours (default=FALSE)
`fk4`	if set, then the celestial (RA, Dec) coordinates are assumed to be input/output in the FK4 system. By default, coordinates are assumed to be in the FK5 system. (default=FALSE)
`supergalactic`	if set, the function returns SuperGalactic coordinates (see details). (default=FALSE)
`mustar`	see details (default=FALSE)

Details

If *j=1*, this function converts proper motion in equatorial coordinates (ra,dec) to proper motion in Galactic coordinates (gl, gb) or Supergalactic Coordinates (sgl,sgb). If *j=2*, the conversion is reversed from Galactic/Supergalactic coordinates to equatorial coordinates. The calculation includes precession on the coordinates, but does not take care of precession of the proper motions which is usually a very small effect.

For B1950 coordinates, set *fk4=TRUE* and *year=1950*.

If *supergalactic=TRUE* is set, Supergalactic coordinates are defined by de Vaucouleurs et al. (1976) to account for the local supercluster. The North pole in Supergalactic coordinates has Galactic coordinates l = 47.47, b = 6.32, and the origin is at Galactic coordinates l = 137.37, b = 0.00.

If *mustar=TRUE* is set, the input and output of mu_ra and mu_dec are the projections of mu in the ra or dec direction rather than the d(ra)/dt or d(mu)/dt. So mu_ra becomes mu_ra*cos(dec) and mu_gl becomes mu_gl*cos(gb).

Value

`ra`	Right Ascension, in decimal hours (or decimal degrees if *degree* is set), scalar or vector
`dec`	declination, in decimal degrees, scalar or vector
`mu_ra`	Right Ascension proper motion, in any proper motion unit (angle/time), scalar or vector
`mu_dec`	declination proper motion, in any proper motion unit (angle/time), scalar or vector
`year`	equinox of ra and dec, scalar
`gl`	Galactic longitude, decimal degrees, scalar or vector
`gb`	Galactic latitude, decimal degrees, scalar or vector
`mu_gl`	Galactic longitude proper motion, in any proper motion unit (angle/time), scalar or vector
`mu_gb`	Galactic latitude proper motion, in any proper motion unit (angle/time), scalar or vector

Author(s)

Written by Ed Shaya (Univ Maryland))2009.

R adaptation by Arnab Chakraborty June 2013

See Also

precess bprecess jprecess

Examples

```
# Find the SuperGalactic proper motion of M33 given its
#  equatorial proper motion mu* =(-29.2, 45.2) microas/yr.
#  Here the (*) indicates ra component is actual mu_ra*cos(dec)
```

```
# (Position: RA (J2000): 01 33 50.9, Dec (J2000): 30 39 35.8)
# Result: SGL = 328.46732 deg, SGB = -0.089896901 deg
# mu_sgl = 35.02 microarcsecond/yr, mu_sgb = 38.09 microarcsecond/yr.

glactc_pm(ten(1,33,50.9), ten(30,39,35.8), -29.2, 45.2, 2000,
     gl, gb, mu_gl, mu_gb, 1)
```

---

hadec2altaz   *Convert Hour Angle and Declination to Horizon (alt-az) coordinates*

Description

Convert Hour Angle and Declination to Horizon (alt-az) coordinates

Usage

```
hadec2altaz( ha, dec, lat, ws=F)
```

Arguments

ha	local apparent hour angle, in degrees, scalar or vector
dec	local apparent declination, in degrees, scalar or vector
lat	local latitude, in degrees, scalar or vector
ws	if FALSE, the output azimuth is measured East from North. If TRUE, the output azimuth is measured West from South. (default=FALSE)

Details

This function is intended mainly to be used by function *eq2hor*. It correctly treats the singularities at the North and South Celestial Poles.

Similar functions, *elev* and *azimuth*, are provided in the CRAN package *astroFns*.

Value

alt	local apparent altitude, in degrees
az	local apparent azimuth, in degrees

Author(s)

Written by Chris O'Dell (Univ. Wisconsin), 2002

R adaptation by Arnab Chakraborty June 2013

See Also

altaz2hadec eq2hor

Examples

```
# What were the apparent altitude and azimuth of the sun when it transited
# the local meridian at Pine Bluff Observatory (Lat=+43.07833 degrees) on
# April 21, 2002?   An object transits the local meridian at 0 hour angle.
# Assume this will happen at roughly 1 PM local time (18:00 UTC).
# Result: Altitude alt = 58.90,  Azimuth  az = 180.0

jd <- jdcnv(2002, 4, 21, 18.) # get rough Julian date to determine Sun declination
sun_pos <- sunpos(jd)
hadec2altaz(0., sun_pos$dec, 43.078333)
```

---

helio — *Compute (low-precision) heliocentric coordinates for the planets*

Description

Compute (low-precision) heliocentric coordinates for the planets

Usage

```
helio(jd, list1, radian=FALSE)
```

Arguments

`jd`	Julian date, scalar or vector
`list1`	List of planets array. May be a single number. 1 = merc, 2 = venus, ... 9 = pluto
`radian`	If =TRUE, then the output longitude and latitude are given in radians. If =FALSE, the output are in degrees. (default=FALSE)

Details

The mean orbital elements for epoch J2000 are used. These are derived from a 250 yr least squares fit of the DE 200 planetary ephemeris to a Keplerian orbit where each element is allowed to vary linearly with time. For dates between 1800 and 2050, this solution fits the terrestrial planet orbits to ~25" or better, but achieves only ~600" precision for Saturn.

These output arrays are dimensioned Nplanet x Ndate, where Nplanet is the number of elements of *list1*, and Ndate is the number of elements of J*jd*.

Use *planet_coords* (which calls *helio*) to get celestial (RA, Dec) coordinates of the planets

Value

`hrad`	array of heliocentric radii, in Astronomical Units
`hlong`	array of heliocentric (ecliptic) longitudes, in degrees or radians
`hlat`	array of heliocentric latitudes, in degrees or radians

Author(s)

R. Sterner 1986 and W. Landsman 2000

R adaptation by Arnab Chakraborty June 2013

See Also

cirrange

Examples

```
# (1) Find the current heliocentric positions of all the planets

jd_today <- 2456877.5
helio(jd_today,seq(1,9))

# (2) Find heliocentric position of Mars on August 23, 2000
# Result: hrad = 1.6407 AU hlong = 124.3197 hlat = 1.7853
# For comparison, the JPL ephemeris gives hrad = 1.6407 AU hlong = 124.2985 hlat = 1.7845
helio(2451779.5,4)
```

---

helio_jd *Convert geocentric (reduced) Julian date to heliocentric Julian date*

Description

Convert geocentric (reduced) Julian date to heliocentric Julian date

Usage

```
helio_jd(date, ra, dec, B1950=FALSE, time_diff=FALSE)
```

Arguments

date	reduced Julian date (= JD - 2400000), scalar or vector
ra,dec	scalars giving right ascension and declination, in degrees
B1950	If =FALSE, equinox is J2000, if =TRUE, equinox is B1950 (default = FALSE)
time_diff	If =TRUE, then the function returns the time difference (heliocentric JD - geocentric JD), in seconds (default=FALSE)

Details

This procedure correct for the extra light travel time between the Earth and the Sun. An online calculator for this quantity is available at http://www.physics.sfasu.edu/astro/javascript/hjd.html. Users requiring more precise calculations and documentation should look at the IDL code available at http://astroutils.astronomy.ohio-state.edu/time/. The algorithm here is from Henden and Kaitchuck (1982, p.114).

Value

jdhelio	heliocentric reduced Julian date (but see *time_diff*)

Author(s)

Written by W. Landsman STX 1989

R adaptation by Arnab Chakraborty June 2013

References

Henden, A. A. and Kaitchuck, R. H., 1982, "Astronomical Photometry", Van Nostrand Reinhold

See Also

bpresess xyz

Examples

```
# What is the heliocentric Julian date of an observation of V402 Cygni
# (J2000: RA = 20 9 7.8, Dec = 37 09 07) taken June 15, 1973 at 11:40 UT?
# Result: hjd = 41848.9881

jd <- juldate(c(1973,6,15,11,40))   #  Get geocentric Julian date
hjd <- helio_jd(jd, ten(20,9,7.8)*15., ten(37,9,7))
```

---

helio_rv — *Calculate the heliocentric radial velocity of a spectroscopic binary*

Description

Calculate the heliocentric radial velocity of a spectroscopic binary

Usage

```
helio_rv(hjd, t, p, v0, k, e, omega, maxiter=100)
```

Arguments

hjd	time of observation
t	time of periastron passage (maximum positive velocity for circular orbits), same time system as hjd
p	orbital period, same time system as hjd
v0	systemic velocity
k	velocity semi-amplitude, same units as v0
e	orbital eccentricity (default=0.0)
omega	longitude of periastron, in degrees (default=0.0, but must be specified for eccentric orbits
maxiter	maximum number of iterations to achieve 1e-8 convergence

Details

The user should ensure consistency with time and velocity systems being used (e.g. days and km/s). Generally, users should reduce large time values by subtracting a large constant offset, which may improve numerical accuracy.

Value

`status`	Iterations needed for convergence
`rv`	predicted heliocentric radial velocity for the date(s) specified by *hjd*, same units as v0

Author(s)

Written by Pierre Maxted ()CUOBS) 1994

R adaptation by Arnab Chakraborty June 2013

Examples

```
# What was the heliocentric radial velocity of the primary component of HU Tau
# at 1730 UT 25 Oct 1994?
# Result: -63.66 km/s

jd <- juldate(c(94,10,25,17,30))   #  obtain Geocentric julian date
hjd <- helio_jd(jd, ten(04,38,16)*15., ten(20,41,05)) #  convert to HJD
helio_rv(hjd, 46487.5303, 2.0563056, -6.0, 59.3)

# Plot two cycles of an eccentric orbit, e=0.6, omega=45 for both
# components of a binary star

phase <- seq(0.0,2.0,length=100)   #  generate 100 phase points
plot(phase, helio_rv(phase, 0, 1, 0, 100, 0.6, 45)$rv, ylim=c(-100,150), pch=20)
lines(phase, helio_rv(phase, 0, 1, 0, 50, 0.6, 45+180)$rv)
```

---

hor2eq — *Converts local horizon coordinates (alt-az) to equatorial coordinates(ra-dec)*

Description

Converts local horizon coordinates (alt-az) to equatorial coordinates(ra-dec)

Usage

```
hor2eq(alt, az, jd,  lat=43.0783, lon= -89.865, ws=FALSE,
            b1950 = FALSE, precess_=TRUE, nutate_=TRUE,
            refract_ = TRUE, aberration_ = TRUE, altitude=0)
```

Arguments

alt	local apparent altitude, in degrees, scalar or vector
az	local apparent altitude, in degrees, scalar or vector, measured east of north unless *ws=TRUE*
jd	Julian Date, in days, scalar or vector
lat	local geodetic latitude of observer, in degrees, scalar or vector (default=43.0783)
lon	east longitude of observer, in degrees; specify west longitude with a negative sign (default=-89.865)
ws	if =TRUE, azimuth is measured westward from south, rather than eastward of north
b1950	if =TRUE, Right Ascension and declination are specified in B1950/FK4, rather than J2000/FK5 coordinates (default=FALSE)
precess_	if =TRUE, precession is applied (default=TRUE)
nutate_	if =TRUE, nutation is applied (default=TRUE)
refract_	if =TRUE, refraction correction is applied (default=TRUE)
aberration_	if =TRUE, aberration correction is applied (default=TRUE)
altitude	altitude of the observing location, in meters (default=0)

Details

This function calculates equatorial (ra,dec) coordinates from horizon (alt,az) coords. It is typically accurate to about 1 arcsecond or better, performing precession, nutation, aberration, and refraction corrections. Inputs can be vectors except for the observer's latitude, longitude and altitude. *ra*, *dec*, *alt* and *az* must be vectors of the same length, but *jd* may be a scalar or a vector of the same length.

Steps in the calculation:
Precess Ra-Dec to current equinox
Nutation Correction to Ra-Dec
Aberration correction to Ra-Dec
Calculate Local Mean Sidereal Time
Calculate Local Apparent Sidereal Time
Calculate Hour Angle
Apply spherical trigonometry to find Apparent Alt-Az
Apply refraction correction to find observed Alt

The user can add specification for *temperature* and *pressure* used by function *co_refract* to calculate the refraction effect of the atmosphere. See *co_refract* for more details.

Value

ra	Right Ascension of object (J2000/FK5), in degrees, scalar or vector
dec	declination of object (J2000/FK5), in degrees, scalar or vector
ha	hour angle, in degrees

Author(s)

Written by Chris O'Dell Univ. of Wisconsin-Madison

R adaptation by Arnab Chakraborty June 2013

See Also

altaz2hadec co_nutate co_refract ct2lst precess

Examples

```
#   You are at Kitt Peak National Observatory, looking at a star at azimuth
#   angle 264d 55m 06s and elevation 37d 54m 41s (in the visible).  Today is
#   Dec 25, 2041 and the local time is 10 PM precisely.  What is the ra and dec
#   (J2000) of the star you're looking at?   The temperature here is about 0
#   Celsius, and the pressure is 781 millibars.    The Julian date for this
#   time is 2466879.7083333.
#   Result: ra=00h13m14.s  dec=+15d11'0.3"   ha=+03h3m30.1s
#   The star is Algenib

hor2eq(ten(37,54,41), ten(264,55,06), 2466879.7083333, lat=+31.9633, lon=-111.6)

# The program produces this output (because the VERBOSE keyword was set):
# Latitude = +31 57 48.0   Longitude = *** 36  0.0   longitude prints weirdly b/c of negative
# input to ADSTRING!!
# Julian Date =  2466879.708333
# Az, El =  17 39 40.4  +37 54 41.0   (Observer Coords)
# Az, El =  17 39 40.4  +37 53 39.6   (Apparent Coords)
# LMST = +03 53 54.1
# LAST = +03 53 53.6
# Hour Angle = +03 38 30.1  (hh:mm:ss)
# Ra, Dec:  00 15 23.5  +15 25  1.9   (Apparent Coords)
# Ra, Dec:  00 15 24.2  +15 25  0.1   (J2041.9841)
# Ra, Dec:  00 13 14.1  +15 11  0.3   (J2000)
# The star is therefore Algenib!  Compare the derived Ra, Dec with what XEPHEM got:
# Ra, Dec:      00 13 14.2  +15 11  1.0   (J2000)
```

---

imf *Compute an N-component power-law logarithmic stellar initial mass function*

Description

Compute an N-component power-law logarithmic stellar initial mass function

Usage

```
imf(mass, expon, mass_range)
```

Arguments

`mass`	mass in units of solar masses, scalar or vector
`expon`	power law exponent, usually negative, scalar or vector
`mass_range`	vector containing the mass upper and lower limits of the IMF and masses where the IMF exponent changes, in solar masses. The number of values in mass_range should be one more than in *expon* and should be monotonically increasing.

Details

The mass spectrum f(m) gives the number of stars per unit mass interval is related to psi(m) by m*f(m) = psi(m). The 'initial' mass function (IMF) refers to the mass spectrum before stellar evolution has reduced the number of higer mass stars. ' For background, see Scalo (1986).

The *imf* function first calculates the constants to multiply the power-law components such that the IMF is continuous at the intermediate masses, and that the total mass integral is one solar mass. That is, the normalization condition is that the integral of psi(m) between the upper and lower mass limit is unity. The IMF is then calculated for the supplied masses.

The number of values in *expon* equals the number of different power-law components in the IMF. A Saltpeter (1955) IMF has a scalar value of expon = -1.35.

Value

`psi`	mass function, number of stars per unit logarithmic mass interval evaluated for supplied masses

Author(s)

Written W. Landsman 1989

R adaptation by Arnab Chakraborty June 2013

References

Salpeter, E. D., 1955, The luminosity function and stellar evolution, Astrophys. J. 121, 161 http://adsabs.harvard.edu/abs/1955ApJ...121..161S

Scalo, J., 1986, The stellar initial mass function, Fund. of Cosmic Physics, 11, 1-278 http://adsabs.harvard.edu/abs/1986FCPh...11....1S

Examples

```
# Calculate the number of stars per unit mass interval at 3 Msun
# for a Salpeter (expon = -1.35) IMF, with a mass range from
# 0.1 MSun to 110 Msun.

imf(3, -1.35, c(0.1, 110) ) / 3

# Lequeux et al. (1981) describes an IMF with an
# exponent of -0.6 between 0.007 Msun and 1.8 Msun, and an
# exponent of -1.7 between 1.8 Msun and 110 Msun.  Plot
```

```
# the mass spectrum f(m)

m = seq(0.01,0.1,length=110)  # construct a mass vector
expon = c(-0.6, -1.7)         # exponent vector
mass_range = c(0.007, 1.8, 110)    # mass range
plot (log10(m), log10(imf(m, expon, mass_range) / m), pch=20)
```

`intdiv` *Integer divide*

Description

Coerces dividend and divisor into integer prior to divide

Usage

```
intdiv(dividend, divisor)
```

Arguments

`dividend`	scalar or vector
`divisor`	scalar or vector

Author(s)

Arnab Chakraborty 2013

Examples

```
intdiv(5.6,-3.1)
intdiv(6.6,-3.1)
```

`ismeuv` *Compute the continuum interstellar extreme ultraviolet (EUV) optical depth*

Description

Compute the continuum interstellar extreme ultraviolet (EUV) optical depth

Usage

```
ismeuv(wave, hcol, heicol=0.1*hcol, heiicol=0*hcol, fano=F)
```

Arguments

wave	vector of wavelength values, in Angstroms
hcol	scalar specifying interstellar hydrogen column density, in atoms cm-2
heicol	scalar specifying neutral helium column density, in atoms cm-2 (default = 0.1*hcol)
heiicol	scalar specifying ionized helium column density, in atoms cm-2 (default = 0.0)
fano	If =TRUE, then the 4 strongest auto-ionizing resonances of He I are included (default = FALSE)

Details

The EUV optical depth is computed from the photoionization of hydrogen and helium. The useful range for *wave* is 40 - 912 A; at shorter wavelengths, metal opacity should be considered, and at longer wavelengths there is no photoionization.To obtain the attenuation of an input spectrum, multiply by exp(-tau).

This function only computes continuum opacities, and for example, the He ionization edges at 504 A and 228 A are blurred by converging line absorptions (Dupuis et al. 1995). The more complete program *ismtau.pro* at http://hea-www.harvard.edu/PINTofALE/pro/ extends this work to shorter wavelengths and includes metal and molecular hydrogen opacities.

Typical values for *hcol* range from 1E17 to 1E20. For *fano*=TRUE, the shape of th auto-ionizing resonances of He I is given by a Fano profile (Rumph et al. 1994). If these resonances are included, then the input wavelength vector should have a fine (>~0.01 A) grid between 190 A and 210 A, since the resonances are very narrow.

Value

tau	vector giving resulting optical depth for each element of *wave*

Author(s)

Written by W. Landsman 1994

R adaptation by Arnab Chakraborty June 2013

References

Dupuis, J., Vennes, S., Bowyer, S., Pradhan, A. K. and Thejll, P., 1995, Hot White Dwarfs in the Local Interstellar Medium: Hydrogen and Helium Interstellar Column Densities and Stellar Effective Temperatures from Extreme-Ultraviolet Explorer Spectroscopy, Astrophys. J. 455, 574 http://adsabs.harvard.edu/abs/1995ApJ...455..574D

Rumph, T., Bowyer, S. and Vennes, S. 1994, Interstellar medium continuum, autoionization, and line absorption in the extreme ultraviolet, Astron. J. 107, 2108-2114 http://adsabs.harvard.edu/abs/1994AJ....107.2108R

Examples

```
# One has a model EUV spectrum with wavelength, w (in Angstroms) and
# flux,f .  Plot the model flux after attenuation by 1e18 cm-2 of HI,
# with N(HeI)/N(HI) = N(HeII)/N(HI) = 0.05

hcol = 1e18
w = seq(100,900,length=801)
ismeuv(w, hcol)

# f = rep(1,length=8*20)
# plot(w, f*exp(-ismeuv(w, hcol, .05*hcol, .05*hcol)), pch=20)

#  Plot the cross-section of HeI from 180 A to 220 A for 1e18 cm-2
#  of HeI, showing the auto-ionizing resonances.   This is
#  Figure 1 in Rumph et al. (1994)

# w = 180 + seq(0,40,length=40000)        # create a fine wavelength grid
# plot(w, ismeuv(w, 0, 1e18, fano=TRUE), pch=20)
```

---

jdcnv *Convert Gregorian dates to Julian days*

Description

Convert Gregorian dates to Julian days

Usage

```
jdcnv(yr, mn, day, hr)
```

Arguments

yr	year, integer scalar or vector
mn	month, integer (1-12) scalar or vector
day	day, integer 1-31) scalar or vector
hr	hours and fractions of hours of universal time (U.T.), scalar or vector

Value

julian	Julian date

Author(s)

Converted to IDL from Yeoman's Comet Ephemeris Generator, B. Pfarr, STX, 1988

R adaptation by Arnab Chakraborty (June 2013)

Examples

```
# To find the Julian Date for 1978 January 1, 0h (U.T.)
# Result: julian = 2443509.5

jdcnv(1978, 1, 1, 0.)
```

---

jprecess *Precess celestial positions from B1950.0 (FK4) to J2000.0 (FK5) with proper motion*

Description

Precess celestial positions from B1950.0 (FK4) to J2000.0 (FK5) with proper motion

Usage

```
jprecess(ra,dec,mu_radec,parallax,rad_vel,epoch=1950)
```

Arguments

`ra`	B1950 right ascension, in degrees (scalar or N-vector)
`dec`	B1950 declination, in degrees (scalar or N-vector)
`mu_radec`	2xN element vector containing proper motion in seconds or arc per tropical century in right ascension and declination (optional)
`parallax`	parallax in seconds of arc, scalar or N-vector (optional)
`rad_vel`	radial velocity in km/s, scalaror N-vector (optional)
`epoch`	epoch of original observation, (default = 1950) (optional)

Details

Calculates the mean place of a celestial object at J2000.0 on the FK5 system from the mean place at B1950.0 on the FK4 system. Use the function *bprecess* for precession in the other direction from J2000 to B1950.

The algorithm is taken from the Explanatory Supplement to the Astronomical Almanac 1992, page 186. Also see Aoki et al (1983).

BPRECESS distinguishes between the following two cases: (1) The proper motion is known and non-zero; and (2) the proper motion is unknown or known to be exactly zero (i.e. extragalactic radio sources). In the latter case, the reverse of the algorithm in Appendix 2 of Aoki et al. (1983) is used to ensure that the output proper motion is exactly zero. Better precision can be achieved in this case by inputting the EPOCH of the original observations.

Value

`ra_2000`	J2000 right ascension, in degrees (scalar or N-vector)
`dec_2000`	J2000 declination, in degrees (scalar or N-vector)

Author(s)

Written, W. Landsman October, 1992

R adaptation by Arnab Chakraborty June 2013

References

The Explanatory Supplement to the Astronomical Almanac, U.S. Naval Observatory, http://aa.usno.navy.mil/publications/doc

Aoki, S., Soma, M., Kinoshita, H. & Inoue, K., Conversion matrix of epoch B 1950.0 FK 4-based positions of stars to epoch J 2000.0 positions in accordance with the new IAU resolutions, Astronomy & Astrophysics 128, 263-267, 1983.

Examples

```
#  The SAO catalogue gives the B1950 position and proper motion for the
#  star HD 119288.   Find the J2000 position.
#  RA(1950) = 13h 39m 44.526s      Dec(1950) = 8d 38' 28.63''
#  Mu(RA) = -.0259 s/yr      Mu(Dec) = -.093 ''/yr
#  Result: 13h 42m 12.740s    +08d 23' 17.69"

mu_radec = c(100*(-15)*0.0259, 100*(-0.093))
ra = ten(13,39,44.526)*15
dec = ten(8,38,28.63)
result = jprecess(ra, dec, mu_radec = mu_radec)
result
adstring(result$ra2000, result$dec2000,2)
```

---

juldate — *Convert from calendar to Reduced Julian Date*

Description

Convert from calendar to Reduced Julian Date

Usage

```
juldate(date)
```

Arguments

`date`	3 to 6-element vector containing year,month (1-12),day, and optionally hour, minute, and second. These are all values of Universal Time. Year should be supplied with all digits. Years B.C should be entered as negative numbers (and note that Year 0 did not exist). If hour, minute or seconds are not supplied, they will default to 0.

Details

Julian Day Number is a count of days elapsed since Greenwich mean noon on 1 January 4713 B.C. The Julian Date (JD) is the Julian day number followed by the fraction of the day elapsed since the preceding noon. The output of this function is the Reduced Julian Date

```
RJD = JD - 2400000.0
```

The function *helio_jd* can be used after *juldate* if a heliocentric Julian date is required. The function *daycnv* converts dates in the opposite direction from Julian dates to Gragorian calendar dates.

The algorithm is obtained from Sky and Telescope April 1981

Value

jd	Reduced Julian Date, scalar

Author(s)

Adapted from IUE RDAF (S. Parsons) 8-31-87

R adaptation by Arnab Chakraborty (June 2013)

See Also

helio_jd daycnv

Examples

```
#  The date of 25-DEC-2006 06:25 UT
#  Result:   JD = 54094.7673611

juldate(c(2006, 12, 25, 6, 25))
juldate(c(2006, 12, 25.2673611))
```

---

ldist *Integrand for luminosity distance calculation*

Description

Integrand for luminosity distance calculation

Usage

```
ldist(z, q0, lambda0)
```

Arguments

`z`	redshift, positive scalar or vector
`q0`	deceleration parameter, scalar
`lambda0`	cosmological constant normalized to the closure density

Details

Used in place of the Mattig formula when the cosmological constant is non-zero.

Value

Value of the partial integral

Examples

```
ldist(3.0, 0.5, 0.7)
```

`lsf_rotate` *Create a 1-d convolution kernel to broaden a spectrum from a rotating star*

Description

Create a 1-d convolution kernel to broaden a spectrum from a rotating star

Usage

```
lsf_rotate(deltav, vsini, epsilon=0.6)
```

Arguments

`deltav`	step increment in the output rotation kernel, scalar, in km/s
`vsini`	rotational velocity projected along the line of sight, scalar, in km/s
`epsilon`	limb-darkening coefficient, scalar (default = 0.6)

Details

This function can be used to derive the broadening effect, or line spread function (LSF), due to stellar rotation on a synthetic stellar spectrum. It assumes constant limb darkening across the disk. To actually compute the broadening. the spectrum should be convolved with the rotational LSF using a function like *kernapply* or *filter*.

The number of points in the output *lsf* will be always be odd (the kernel is symmetric) and equal to either ceil(2*Vsini/deltav) or ceil(2*Vsini/deltav) +1 (whichever number is odd).

The limb darkening coefficient *epsilon* = 0.6 is typical for photospheric lines. The specific intensity I at any angle theta from the specific intensity Icen at the center of the disk is given by

```
I = Icen*(1-epsilon*(1-cos(theta))
```

.

The algorithm is adapted from rotin3.f in the SYNSPEC software of Hubeny & Lanz http://nova.astro.umd.edu/. Also see Eq. 17.12 in Gray (1992).

Value

lsf	convolution kernel vector for the specified rotational velocity

Author(s)

Written by W. Landsman 2001

R adaptation by Arnab Chakraborty June 2013

References

Gray, D., 1992, "The Observation and Analysis of Stellar Photospheres"

Examples

```
# Plot the LSF for a star rotating at 90 km/s in both velocity space and
# for a central wavelength of 4300 A.    Compute the LSF every 3 km/s

lsf = lsf_rotate(3,90)      # LSF will contain 61 pts
```

---

lumdist *Calculate luminosity distance (in Mpc) of an object given its redshift*

Description

Calculate luminosity distance (in Mpc) of an object given its redshift

Usage

```
lumdist(z, h0=70, k, lambda0, omega_m, q0)
```

Arguments

z	redshift, positive scalar or vector
h0	Hubble expansion parameter, in km/s/Mpc (default = 70.0)
k	curvature constant normalized to the closure density (default = 0.0 corresponding to a flat universe)
omega_m	matter density normalized to the closure density (default = 0.3)
lambda0	cosmological constant normalized to the closure density (default = 0.7)
q0	deceleration parameter, scalar (default = 0.55)

Details

The luminosity distance in the Friedmann-Robertson-Walker model is taken from Carroll et al. (1992, p.511). It uses a closed form (Mattig equation) to compute the distance when the cosmological constant is zero, and otherwise computes the partial integral using the R function *integrate*. The integration can fail to converge at high redshift for closed universes with non-zero lambda.

No more than two of the four parameters (k, omega_M, lambda0, q0) should be specified. None of them need be specified if the default values are adopted.

Value

`lumdist` The result of the function is the luminosity distance (in Mpc) for each input value of z

Author(s)

Written W. Landsman Raytheon ITSS 2000

R adaptation by Arnab Chakraborty June 2013

References

Carroll, S. M., Press, W. H. and Turner, E. L., 1992, The cosmological constant, Ann. Rev. Astron. Astrophys., 30, 499-542

Examples

```
# Plot the distance of a galaxy in Mpc as a function of redshift out
#  to z = 5.0, assuming the default cosmology (Omega_m=0.3, Lambda = 0.7,
#  H0 = 70 km/s/Mpc)

z <- seq(0,5,length=51)
plot(z, lumdist(z), type='l', lwd=2, xlab='z', ylab='Distance (Mpc)')

# Now overplot the relation for zero cosmological constant and
# Omega_m=0.3

lines(z,lumdist(z, lambda=0, omega_m=0.3), lty=2, lwd=2)
```

`mag2flux` *Convert from astronomical magnitudes to flux (ergs/s/cm^2/A)*

Description

Convert from astronomical magnitudes to flux (ergs/s/cm^2/A)

Usage

```
mag2flux(mag, zero_pt=21.10, ABwave=F)
```

Arguments

`mag`	magnitude, scalar or vector
`zero_pt`	zero point level of the magnitude (default = 21.1)
`ABwave`	wavelength for conversion to Oke AB magnitudes, in Angstroms (optional)

Details

The default zero point of 21.1 mag is from Code et al. (1976). It is ignored of the ABwave parameter is supplied.

If ABwave is not supplied, the routine returns magnitudes given by the expression

`mag <- -2.5*log10(flux) - zero_pt`. If ABwave is supplied, then the routine returns AB magnitudes from Oke and Gunn (1983) according to

`abmag <- -2.5*log10(flux) - 5*log10(ABwave) - 2.406`.

Use *mag2flux* for conversions in the opposite direction.

Value

`flux`	flux, in erg cm-2 s-1 A-1, scalar or vector

Author(s)

Converted to IDL from Yeoman's Comet Ephemeris Generator, B. Pfarr, STX, 1988

R adaptation by Arnab Chakraborty (June 2013)

References

Oke, J. B. and Gunn, J. E., 1983, Secondary standard stars for absolute spectrophotometry, Astrophysical Journal, 266, 713-717

Examples

```
mag2flux(19.8)

ytext <- expression(Flux~~ (erg/s~cm^2~A))
plot(seq(2000,5000,length=100), flux2mag(seq(15, 20,length=100)), pch=20)
```

month_cnv *Convert between a month name and the equivalent number*

Description

Convert between a month name and the equivalent number

Usage

```
month_cnv(monthinput, up=FALSE, short=FALSE)
```

Arguments

monthinput	either a string ('January', 'Jan', 'Decem', etc.) or a number from 1 to 12, scalar or array.
up	if =TRUE and if a string is being returned, it will be in all uppercase letters. If = FALSE, all lowercase letters are used (default = FALSE)
short	if =TRUE and if a string is being returned, only the first three letters are returned (default = FALSE)

Details

For example, this function converts from 'January' to 1, or vice-versa. String output can be in the form "january", "JANUARY", "jan", "JAN".

Value

If the input is a string, the output is the matching month number.If an input string is not a valid month name, -1 is returned.
If the input is a number, the output is the matching month name. See details for formats.

Author(s)

Written by: Joel Wm. Parker, SwRI, 1998

R adaptation by Arnab Chakraborty June 2013

Examples

```
month_cnv('Apr')
month_cnv(4,short=TRUE,up=TRUE)
```

`moonpos` *Compute the Right Ascension and Declination of the Moon at specified Julian date(s)*

Description

Compute the Right Ascension and Declination of the Moon at specified Julian date(s)

Usage

```
moonpos(jd, radian=F)
```

Arguments

`jd`	Julian ephemeris date, scalar or vector
`radian`	if =TRUE, then all output variables are given in radians rather than degrees (default=FALSE)

Details

The method is derived from the Chapront ELP2000/82 Lunar Theory (Chapront-Touze and Chapront, 1983), as described by Jean Meeus (1998, Chpt. 47). Meeus quotes an approximate accuracy of 10" in longitude and 4" in latitude, but he does not give the time range for this accuracy. Comparison of this procedure with the example in “Astronomical Algorithms” reveals a very small discrepancy (~1 km) in the distance computation, but no difference in the position calculation.

Value

`ra`	apparent right ascension of the moon, referred to the true equator of the specified date(s), in degrees
`dec`	declination of the moon, in degrees
`dis`	Earth-moon distance between the center of the Earth and the center of the Moon, in km
`geolong`	apparent longitude of the moon referred to the ecliptic of the specified date(s), in degrees
`geolat`	apparent longitude of the moon referred to the ecliptic of the specified date(s), in degrees

Author(s)

Written W. Landsman

R adaptation by Arnab Chakraborty June 2013

References

Chaprint-Touze, M, and Chapront, J. 1983, The lunar ephemeris ELP 2000, Astron. Astrophys. 124, 50-62.

Meeus, J., 1998, “Astronomical Algorithms”, 2nd ed., Willmann-Bell

See Also

cirrange nutate

Examples

```
# Find the position of the moon on April 12, 1992
# Result: 08 58 45.23  +13 46  6.1
# This is within 1" from the position given in the Astronomical Almanac

jd = jdcnv(1992,4,12,0)    # get Julian date
pos = moonpos(jd)     # get RA and Dec of moon
adstring(pos$ra,pos$dec,1)


# Plot the Earth-moon distance for every day at 0 TD in July, 1996

jd = jdcnv(1996,7,1,0)  # get Julian date of July 1
pos = moonpos(jd+rep(0,31))  # Moon position at all 31 days
plot(seq(1,31), pos$dis, xlab="July 1996 (day)", ylab="Lunar distance (km)")
```

---

mphase — *Calculate the illuminated fraction of the Moon at given Julian date(s)*

Description

Calculate the illuminated fraction of the Moon at given Julian date(s)

Usage

```
mphase(jd)
```

Arguments

jd	Julian date, scalar or vector

Details

The output k = 0 indicates a new moon, while k = 1 indicates a full moon.

The algorithm is from Meeus (1991, Chpt. 46) The functions *sunpos* and *moonpos* are used to get positions of the Sun and the Moon (and the Moon distance). The selenocentric elongation of the Earth from the Sun (phase angle) is then computed, and used to determine the illuminated fraction.

Value

`k` illuminated fraction of Moon's disk (0.0 < k < 1.0), same number of elements as jd.

Author(s)

Written W. Landsman Hughes STX 1996

R adaptation by Arnab Chakraborty June 2013

References

Meeus, J., 1991, "Astronomical Algorithms", Willmann-Bell, Richmond

See Also

sunpos moonpos

Examples

```
mphase(2456877.5)

#  Plot the illuminated fraction of the moon for every day in July
#  1996 at 0 TD (Greenwich noon).
#
# jd = jdcnv(1996, 7, 1, 0)         # Get Julian date of July 1
# phases = mphase(jd:(jd+31))       # Moon phase for all 31 days
# plot(1:31,phases)                 # Plot phase vs. July day number
```

---

nutate *Calculate the nutation in longitude and obliquity for a given Julian date*

Description

Calculate the nutation in longitude and obliquity for a given Julian date

Usage

```
nutate(jd)
```

Arguments

`jd` Julian ephemeris date, scalar or vector

Details

This function uses the formula in Meuss (1998, Chpt. 22) which is based on the 1980 IAU Theory of Nutation and includes all terms larger than 0.0003".

Value

nut_long	nutation in longitude, same number of elements as jd
nut_obliq	nutation in latitude, same number of elements as jd

Author(s)

Written, W. Landsman 1992

R adaptation by Arnab Chakraborty June 2013

References

Meeus, J., 1998, "Astronomical Algorithms", 2nd ed.

See Also

cirrange polyidl

Examples

```
# Find the nutation in longitude and obliquity 1987 on Apr 10 at Oh.
# Result: nut_long = -3.788    nut_obliq = 9.443
# This is example 22.a from Meeus

jul = jdcnv(1987,4,10,0)
nutate(jul)

# Plot the large-scale variation of the nutation in longitude
# during the 20th century. This plot will reveal the dominant 18.6 year
# period, but a finer grid is needed to display the shorter periods in
# the nutation.


yr = 1900 + seq(0,100)    # establish sequence of years
jul = jdcnv(yr,1,1,0)          # find Julian date of first day of year
out = nutate(jul)               # compute nutation
plot(yr, out$nut_long, lty=1, lwd=2, xlab='Year', ylab='Nutation longitude (degrees)')
```

planck *Calculate the Planck function in units of ergs/cm2/s/A*

Description

Calculate the Planck function in units of ergs/cm2/s/A

Usage

```
planck (wave,temp)
```

Arguments

wave	wavelength(s) at which the Planck function is to be evaluated, in Angstroms, scalar or vectxor
temp	temperature of the Planck function, in degree K, scalar

Details

In this function, the wavelength data are converted to centimeters, and the Planck function is calculated for each wavelength point. See Allen (1973, sec 44) for more information.

If a star with a blackbody spectrum has a radius R and distance d, then the flux at Earth in erg/cm^2/s/A will be bbflux*R^2/d^2.

Value

bbflux	blackbody flux (i.e. pi*Intensity) at the specified wavelengths, in erg/cm^2/s/A

Author(s)

Adapted from the IUE RDAF 1989

R adaptation by Arnab Chakraborty June 2013

References

Allen, C. W., "Astrophysical Quantities", Athlone Press, 3rd ed. http://adsabs.harvard.edu/abs/1973asqu.book.....A

Examples

```
#   Calculate the blackbody flux at 30,000 K every 100 Angstroms between 2000A and 4000A

wave = 2000 + seq(0,2000,by=100)
plot(wave, planck(wave,30000), lty=1, lwd=2)
```

`planet_coords` *Calculate low precision Right Ascension and declination for the planets given a date*

Description

Calculate low precision Right Ascension and declination for the planets given a date

Usage

```
planet_coords(date, planet=planet, jd = FALSE)
```

Arguments

`date`	If *jd=FALSE*, then *date* is a 3-6 element vector containing year,month (1-12), day, and optionally hour, minute, & second. If *jd=TRUE*, then *date* is a vector of Julian dates.
`planet`	scalar string giving name of a planet, e.g. 'venus' (default = planet that computes coordinates for all planets except Earth)
`jd`	If =TRUE, then the date parameter should be supplied as one or more Julian dates (default = FALSE)

Details

For low precision, this routine uses function *helio* to get the heliocentric ecliptic coordinates of the planets at the given date, then converts these to geocentric ecliptic coordinates following Meeus (1991, p.209). These are then converted to Right Ascension and declination using the function *euler*. The function returns astrometric coordinates, i.e. no correction for aberration. The accuracy between the years 1800 and 2050 is better than 1 arcminute for the terrestial planets, but reaches 10 arcminutes for Saturn. Before 1850 or after 2050 the accuracy can get much worse.

The high precision option available in the IDL procedure based on JPL planetary ephemerides is not current available in the R **astrolib** package. The *helio* function is based on the two-body problem and neglects interactions between the planets. This is why the worst results are for Saturn.

Value

`ra`	Right Ascension of planet(s), J2000 degrees
`dec`	declination of planet(s), J2000 degrees

Author(s)

Written P.Plait & W. Landsman 2000

R adaptation by Arnab Chakraborty June 2013

References

Meeus, J. 1991, "Astronomical Algorithms"

See Also

helio euler juldate

Examples

```
# Find the RA, Dec of Venus on 1992 Dec 20
# Result: RA = 21 05  2.66  Dec = -18 51 45.7

planet_coords(c(1992,12,20))    # compute for all planets
adstring(ra[2],dec[2],1)        # Venus is second planet
# This position is 37" from the full DE406 ephemeris position of
# RA = 21 05  5.24        -18 51 43.1

# Plot the declination of Mars for every day in the year 2001

jd = jdcnv(2001,1,1,0)      # get Julian date of midnight on Jan 1
out = planet_coords(jd+seq(0,365), planet='mars')
plot(jd+seq(0,365), out$dec, pch=20, xlab='Day of 2001', ylab='Declination of Mars (degrees)')
```

---

polyidl *Calculate polynomial*

Description

Calculate polynomial following IDL's poly.pro function

Usage

```
polyidl(x,cc)
```

Arguments

x	scalar, vector or array
cc	vector of polynomial coefficients for polynomial of degree length(cc)-1

Value

This function returns the quantity
cc[1] + cc[2]*x + cc[3]*x^2 + cc[4]*x^3 + ...

Author(s)

Eric Feigelson July 2014

References

See http://www.exelisvis.com/docs/POLY.html

Examples

```
polyidl(2:4, 3:5)  # returns 31,60,99
```

---

posang *Compute position angle of source 2 relative to source 1*

Description

Computes position angle of source 2 relative to source 1

Usage

```
posang(u,ra1,dc1,ra2,dc2)
```

Arguments

u	binary indicator describing units of inputs and output: 0 = radians; 1= RAx in decimal hours, DCx in decimal degrees, ANGLE in degrees
ra1	Right Ascension of point 1
dc1	declination of point 1
ra2	Right Ascension of point 2
dc2	declination of point 2

Details

Computes the rigorous position angle of source 2 (with given RA, Dec) using source 1 (with given RA, Dec) as the center based on the "four-parts formula" from spherical trigonometry (Smart 1977, p.12)

If *(ra1,dc1)* and *(ra2,dc2)* are vectors, then *angle* is a vector giving the position angle between each element of *(ra2,dc2)* and *(ra1,dc1)*. Similarly, if *(ra1,dc1)* are vectors and *(ra2,dc2)* are scalars, then *angle* is a vector giving the position angle of each element of *(ra1,dc1)* and *(ra2,dc2)*. If both *(ra1,dc1)* and *(ra2,dc2)* are vectors, then *angle* is a vector giving the position angle between each element of *(ra1,dc1)* and the corresponding element of *(ra2,dc2)*. If then vectors are not the same length, then excess elements of the longer one will be ignored.

Note that *posang* is not commutative: the position angle between A and B is theta, then the position angle between B and A is 180+theta

Value

angle	angle of the great circle containing (ra2, dc2) from the meridian containing (ra1, dc1), in the sense north through east rotating about (ra1, dc1). See *u* above for units.

Author(s)

Modified from GCIRC, R. S. Hill, RSTX, 1 Apr. 1998

R adaptation by Arnab Chakraborty June 2013

References

Smart, W. M., 1977, "Textbook on Spherical Astronomy", Cambridge Univ. Press (originally published 1931) http://adsabs.harvard.edu/abs/1977tsa..book.....S

Examples

```
#  For the star 56 Per, the Hipparcos catalog gives a position of
#  RA = 66.15593384, Dec = 33.94988843 for component A, and
#  RA = 66.15646079, Dec =  33.96100069 for component B.
#  What is the position angle of B relative to A?
#  Result:  21.4 degrees

ra1 = 66.15593384/15. ;  dc1 = 33.95988843
ra2 = 66.15646079/15. ;  dc2 = 33.96100069
posang(1,ra1,dc1,ra2,dc2)
```

---

precess — *Precess coordinates from EQUINOX1 to EQUINOX2*

Description

Precess coordinates from EQUINOX1 to EQUINOX2

Usage

```
precess(ra, dec, equinox1, equinox2, fk4=F, radian=F)
```

Arguments

ra	Right Ascension, in degrees, scalar or vector
dec	declination, in degrees, scalar or vector
equinox1	original equinox of coordinates, scalar
equinox2	equinox of precessed coordinates
fk4	if =TRUE, the FK4 (B1950.0) system will be used; otherwise FK5 (J2000.0) will be used (default = FALSE)
radian	if =TRUE, the input and output RA and DEC vectors are in radians rather than degrees (default = FALSE)

Details

The algorithm of this function is obtained from Taff (1983, p.24) for FK4. FK5 constants are obtained from "Astronomical Almanac Explanatory Supplement (1992), page 104, Table 3.211.1.

The accuracy of precession decreases for declination values near 90 degrees. PRECESS should not be used more than 2.5 centuries from 2000 on the FK5 system (1950.0 on the FK4 system).

The default (RA,DEC) system is FK5 based on epoch J2000.0, but FK4 based on B1950.0 is available via the /FK4 keyword. Use BPRECESS and JPRECESS to convert between FK4 and FK5 systems

Value

ra	precessed Right Ascension, in degrees, scalar or vector
dec	precessed declination, in degrees, scalar or vector

Author(s)

Written, Wayne Landsman, STI Corporation 1986

R adaptation by Arnab Chakraborty June 2013

References

Taff, L. G., 1983, "Computational Spherical Astronomy", Krieger Publ.

See Also

premat ten

Examples

```
#  The Pole Star has J2000.0 coordinates (2h, 31m, 46.3s,
#  89d 15' 50.6"); compute its coordinates at J1985.0
#  Result: 2h 16m 22.73s, 89d 11' 47.3"

precess(ten(2,31,46.3)*15, ten(89,15,50.6), 2000, 1985)


# Precess the B1950 coordinates of Eps Ind (RA = 21h 59m,33.053s,
# DEC = (-56d, 59', 33.053") to equinox B1975.

ra = ten(21, 59, 33.053)*15
dec = ten(-56, 59, 33.053)
precess(ra, dec , 1950, 1975, fk4=TRUE)
```

precess_xyz *Precess equatorial geocentric rectangular coordinates*

Description

Precess equatorial geocentric rectangular coordinates

Usage

```
precess_xyz(x, y, z, equinox1, equinox2)
```

Arguments

x	heliocentric rectangular coordinate, scalar or vector
y	heliocentric rectangular coordinate, scalar or vector
z	heliocentric rectangular coordinate, scalar or vector
equinox1	equinox of input coordinates, scalar
equinox2	equinox of output coordinates, scalar

Details

The equatorial geocentric rectangular coordinates are converted to (RA,Dec), precessed in the normal way, and then converted back to (x,y,z) using unit vectors.

The input (x,y,z) coordinates are changed upon return.

Value

x	precessed heliocentric rectangular coordinate, scalar or vector
y	precessed heliocentric rectangular coordinate, scalar or vector
z	precessed heliocentric rectangular coordinate, scalar or vector

Author(s)

P. Plait ACC 1999

Arnab Chakraborty R adaptation 2013

Examples

```
precess(1.0, 1.0, 1.0, 2000, 2050)
```

premat *Return the precession matrix needed to go from EQUINOX1 to EQUINOX2*

Description

Return the precession matrix needed to go from EQUINOX1 to EQUINOX2

Usage

```
premat(equinox1, equinox2, fk4=F)
```

Arguments

`equinox1`	original equinox of coordinates, scalar
`equinox2`	equinox of precessed coordinates
`fk4`	if =TRUE, the FK4 (B1950.0) system will be used; otherwise FK5 (J2000.0) will be used (default = FALSE)

Details

This matrix is used by the functions *precess* and *baryvel* to precess astronomical coordinates. The algorithm of this function is obtained from Taff (1983, p.24) for FK4. FK5 constants are obtained from "Astronomical Almanac Explanatory Supplement (1992), page 104, Table 3.211.1.

Value

`matrix`	3 x 3 precession matrix, used to precess equatorial rectangular coordinates

Author(s)

Written, Wayne Landsman, HSTX Corporation, 1994

R adaptation by Arnab Chakraborty June 2013

References

Taff, L. G., 1983, "Computational Spherical Astronomy", Krieger Publ.

See Also

baryvel precess

Examples

```
#  Return the precession matrix from 1950.0 to 1975.0 in the FK4 system

premat(1950.0, 1975.0, fk4=TRUE)
```

`radec` *Convert Right Ascension and declination from decimal to sexigesimal units*

Description

Convert Right Ascension and declination from decimal to sexigesimal units

Usage

```
radec(ra, dec, hours=F)
```

Arguments

`ra`	Right Ascension, in degrees, scalar or vector
`dec`	declination, in degrees, scalar or vector
`hours`	if =TRUE, then the input right ascension should be specified in decimal hours instead of degrees (default = FALSE)

Details

The conversion is to sexigesimal hours for RA, and sexigesimal degrees for declination.

Value

A *list* with components:

`ihr`	Right Ascension hours, integer scalar or vector
`imin`	Right Ascension minutes, integer scalar or vector
`xsec`	Right Ascension seconds, real scalar or vector
`ideg`	declination degrees, integer scalar or vector
`imn`	declination degrees, integer scalar or vector
`xsc`	declination degrees, real scalar or vector

Author(s)

Written by B. Pfarr, STX, 1987

R adaptation by Arnab Chakraborty June 2013

Examples

```
radec(10.592, -82.663)
```

`rhotheta` *Calculate the separation and position angle of a binary star*

Description

Calculate the separation and position angle of a binary star

Usage

```
rhotheta(p, t, e, a, i, omega, omega2, t2)
```

Arguments

`p`	period (scalar, year)
`t`	time of periastron passage (scalar, year)
`e`	orbit eccentricity (scalar between 0 and 1)
`a`	semi-major axis (scalar, arc second)
`i`	scalar, inclination
`omega`	node (scalar, degree)
`omega2`	longitude of periastron (scalar, degree)
`t2`	epoch of observation (scalar, year)

Details

This function will return the separation rho and position angle theta of a visual binary star derived from its orbital elements. The algorithm is from Meuss (1992; also 1998).

Value

An R list with two scalar elements: \

`rho`	separation (arcsec)
`theta`	position angle measured east of north (degree)

In case of errors, rho and theta are returned as -1.

Author(s)

Written, Sebastian Kohl, 2012.

R adaptation by Arnab Chakraborty June 2013

References

Meeus J., 1992, Astronomische Algorithmen, Barth. Meeus, J., 1998, “Astronomical Algorithms”, 2nd ed.

Examples

```
#  Find the position of Eta Coronae Borealis at the epoch 1980.0
#  Result:   rho=     0.411014    theta=       318.42307
rhotheta(41.623, 1934.008, 0.2763, 0.907, 59.025, 23.717, 219.907, 1980.0)
```

---

sixty *Convert a decimal number to sexigesimal*

Description

Convert a decimal number to sexigesimal

Usage

```
sixty(scalar, trailsign = F)
```

Arguments

`scalar`	decimal quantity, scalar
`trailsign`	if =TRUE, then the function returns a negative sign to the first element, even if it is zero. If = FALSE, then the function returns a negative sign in the first nonzero element. (default = FALSE)

Details

Reverse of the function *ten*.

Value

`result`	real vector of three elements, sexigesimal equivalent of input decimal quantity

Author(s)

Written by R. S. Hill, STX, 1987

R adaptation by Arnab Chakraborty June 2013

Examples

```
sixty(136.127)
sixty(-0.345)  #  returns (0.0,-20.0,42.0)
sixty(-0.345, trailsign=TRUE)  #  returns (-0.0,20.0,42.0)
```

sphdist *Distance on a sphere*

Description

Angular distance between two points on a sphere, specified by longitude and latitude

Usage

```
sphdist(long1, lat1, long2, lat2, degrees = FALSE)
```

Arguments

long1	Longitude of the first point
lat1	Latitude of the first point
long2	Longitude of the second point
lat2	Latitude of the second point
degrees	Flag denoting whether input angles are in degrees or radians

Details

The distance is computed after conversion from spherical to rectangular coordinates.

Value

dis	Angle, in degrees or radians

Author(s)

Arnab Chakraborty R adaptation 2013

See Also

gcirc

Examples

```
sphdist(2, 100, -35, 180, +35)
```

`sunpos` *Compute the Right Ascension and Declination of the Sun at specified Julian date(s)*

Description

Compute the Right Ascension and Declination of the Sun at specified Julian date(s)

Usage

```
sunpos(jd, radian=F)
```

Arguments

`jd`	Julian ephemeris date, scalar or vector
`radian`	if =TRUE, then all output variables are given in radians rather than degrees (default=FALSE)

Details

This function uses a truncated version of Newcomb's Sun http://en.wikipedia.org/wiki/Newcomb's_Tables_of_the_Sun. The returned RA and Dec are in the given date's equinox.

Patrick Wallace (Rutherford Appleton Laboratory, UK) has tested the accuracy of a C adaptation of the IDL *sunpos.pro* code and found the following results. From 1900-2100 *sunpos* gave 7.3 arcsec maximum error, 2.6 arcsec RMS. Over the shorter interval 1950-2050 the figures were 6.4 arcsec max, 2.2 arcsec RMS.

Value

`ra`	apparent right ascension of the Sun, referred to the true equator of the specified date(s), in degrees
`dec`	declination of the Sun, in degrees
`elong`	ecliptic longitude of the Sun, in degrees
`obliquity`	obliquity of the ecliptic, in degrees

Author(s)

FORTRAN routine by B. Emerson (RGO); IDL version by Michael R. Greason, STX, 1988

R adaptation by Arnab Chakraborty June 2013

See Also

`cirrange` `nutate` `polyidl` `ten`

Examples

```
# Find the apparent RA and Dec of the Sun on May 1, 1982
# Result:  02 31 32.61  +14 54 34.9
# The Astronomical Almanac gives 02 31 32.58 +14 54 34.9,
# so the error in sunpos for this case is < 0.5".

jd = jdcnv(1982, 5, 1,0)      # Find Julian date jd = 2445090.5
out = sunpos(jd)


# Plot the apparent declination of the Sun for every day in 1997

jd = jdcnv(1997,1,1,0)  # Julian date on Jan 1, 1997
days = seq(0,365)
plot(days, sunpos(jd+days)$dec, type='b', pch=20, lwd=2)
```

`ten` *Convert a sexigesimal number or string to decimal*

Description

Convert a sexigesimal number or string to decimal

Usage

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

Arguments

`dd`	degrees (0-360) or hour (0-24), integer, scalar or string giving sexigesimal quantity separated by spaces or colons; e.g.. "10 23 34" or "-3:23:45.2".
`mm`	minutes, integer (0-60), scalar (default = 0)
`ss`	seconds, integer (0-60), scalar (default = 0)

Details

The output is a real number

`= dd + mm/60. + ss/3600`

. Inverse of the *sixty* function. The function *tenv* can be used when dealing with a vector of sexigesimal quantities.

Value

decimal equivalent of input sexigesimal quantity, real, scalar

Author(s)

Written W. Landsman Raytheon ITSS 2000

R adaptation by Arnab Chakraborty June 2013

See Also

sixty

Examples

```
ten(12,0,0)   # gives 12
ten("12:00:00")   # gives 12
ten(0,-23,34)  # gives -0.39277778
ten("-0:23:34")   # gives -0.39277778
```

---

uvbybeta *Derive dereddened colors, metallicity, and Teff from Stromgren colors*

Description

Derive dereddened colors, metallicity, and Teff from Stromgren colors

Usage

```
uvbybeta(xby, xm1, xc1, xhbeta, xn, eby_in, name)
```

Arguments

xby	Stromgren b-y color, in real magnitudes, scalar or vector
xm1	Stromgren line-blanketing parameter, real, scalar or vector
xc1	Stromgren Balmer discontinuity parameter, real, scalar or vector
xhbeta	H-beta line strength index. See details for use.
xn	Spectral class integer indicator (1-8), scalar or vector. See details for class assignments.
eby_in	E(b-y) color, in real magnitudes, scalar. If not supplied, then E(b-y) will be estimated from the Stromgren colors.
name	string giving name(s) of star(s), scalar or vector. Used only when writing to disk for identification purposes.

Details

Method and code adapted from FORTRAN routine of same name published by T.T. Moon, Communications of University of London Observatory, No. 78 (1985).

Set input *xhbeta* to 0 if it is not known, and the function *ubvybeta* will estimate a value based on xby, xm1,and xc1. H-beta is not used for stars in group 8.

The indicator *n* gives approximate spectral class assignments as follows:

```
n=1 B0 - A0, classes III - V, 2.59 < Hbeta < 2.88,-0.20 <   c0  < 1.00
n=2 B0 - A0, class   Ia     , 2.52 < Hbeta < 2.59,-0.15 <   c0  < 0.40
n=3 B0 - A0, class   Ib     , 2.56 < Hbeta < 2.61,-0.10 <   c0  < 0.50
n=4 B0 - A0, class   II     , 2.58 < Hbeta < 2.63,-0.10 <   c0  < 0.10
n=5 A0 - A3, classes III - V, 2.87 < Hbeta < 2.93,-0.01 < (b-y)o< 0.06
n=6 A3 - F0, classes III - V, 2.72 < Hbeta < 2.88, 0.05 < (b-y)o< 0.22
n=7 F1 - G2, classes III - V, 2.60 < Hbeta < 2.72, 0.22 < (b-y)o< 0.39
n=8 G2 - M2, classes  IV - V, 0.20 < m0    < 0.76, 0.39 < (b-y)o< 1.00
```

Value

A data frame with the following columns:

`name`	string giving name(s) of star(s), scalar or vector.
`group`	derived n, approximate spectral class
`by`	Stromgren b-y color, in real magnitudes, scalar or vector
`hbeta.status`	Flag: 0 = H-beta value is input; 1 = H-beta value is estimated
`by0`	dereddened color index, E(b-y)_0
`m0`	dereddened magnitude, m_0
`c0`	dereddened Stromgren Balmer discontinuity parameter, c_0
`eby`	color excess, E(b-y)
`mv`	dereddened visual magnitude, M_V
`radius`	estimated stellar radius, in R_solar
`delm0.status`	Metallicity flag: 1 if n=1; 2 if n=3; otherwise 0
`delm0`	estimated metallicity index, m0_ZAMS - m_0
`teff`	estimated stellar effective temperature, T_eff
`warn`	warnings from code

Author(s)

W. Landsman IDL coding 1988

R adaptation by Arnab Chakraborty June 2013

See Also

`deredd`

Examples

```
#  Suppose 5 stars have the following Stromgren parameters.
#  Determine their stellar parameters.
#  Result: E(b-y) = 0.050    0.414   0.283  0.023  -0.025
#          Teff =   13060    14030   18420  7250    5760
#          M_V =    -0.27    -6.91   -5.94  2.23    3.94
#          radius=  2.71     73.51    39.84 2.02    1.53

by = c(-0.001 ,0.403, 0.244, 0.216, 0.394)
m1 = c(0.105, -0.074, -0.053, 0.167, 0.186)
c1 = c(0.647, 0.215, 0.051, 0.785, 0.362)
hbeta = c(2.75, 2.552, 2.568, 2.743, 0)
nn = c(1,2,3,7,8)

out = uvbybeta(by, m1, c1, hbeta, nn)
c(out$by0, out$teff, out$mv, out$radius)
```

`vactoair` *Convert vacuum wavelengths to air wavelengths*

Description

Convert vacuum wavelengths to air wavelengths

Usage

```
vactoair(wave_vac)
```

Arguments

`wave_vac`	vacuum wavelength, in Angstroms, scalar or vector

Details

Corrects for the index of refraction of air under standard conditions. Wavelength values below 2000 A will not be altered. Method from Ciddor (1996). Accurate to about 10 m/s.

Value

`wave_air`	air wavelength, in Angstroms, scalar -r vector

Author(s)

Written by D. Lindler 1982

R adaptation by Arnab Chakraborty June 2013

References

Ciddor, P. E. 1996, Refractive index of air: New equations for the visible and near infrared, Applied Optics, 35, 1566. http://adsabs.harvard.edu/abs/1996ApOpt..35.1566C

See Also

airtovac

Examples

```
vactoair(2000.000)   # yields air wavelength 1999.353 Angstroms
```

---

xyz *Calculate geocentric X,Y, and Z and velocity coordinates of the Sun*

Description

Calculate geocentric X,Y, and Z and velocity coordinates of the Sun

Usage

```
xyz(date, equinox)
```

Arguments

`date`	reduced julian date (=JD - 2400000), scalar or vector
`equinox`	equinox of output (default = 1950)

Details

Calculates geocentric X,Y, and Z vectors and velocity coordinates (dx, dy and dz) of the Sun. The positive X axis is directed towards the equinox, the y-axis is directed towards the point on the equator at right ascension 6h, and the z axis is directed toward the north pole of the equator. Typical position accuracy is <1e-4 AU (15000 km).

The Earth-Sun distance is given by sqrt(x^2 + y^2 + z^2) for the given date. Note that velocities in the Astronomical Almanac are for Earth/Moon barycenter (a very minor offset); see AA 1999 page E3.

Value

`x,y,z`	geocentric rectangular coordinates, in Astronomical Units, scalar or vector
`xvel,yvel,zvel`	velocity vectors corresponding to X, Y and Z

Author(s)

Written W. Landsman Raytheon ITSS 1989 and 2000

R adaptation by Arnab Chakraborty June 2013

References

Original algorithm from Almanac for Computers, Doggett et al. USNO 1978 Adapted from the book Astronomical Photometry by A. Henden

Examples

```
#   What were the rectangular coordinates of the Sun on
#  Jan 22, 1999 0h UT (= JD 2451200.5) in J2000 coords?
#  NOTE: Astronomical Almanac (AA) is in TDT, so add 64 seconds to UT to convert.

xyz(51200.5+64./86400, equinox=2000)

#   Compare to Astronomical Almanac (1999 page C20)
#              X  (AU)        Y  (AU)     Z (AU)
#  XYZ:      0.51456871   -0.76963263  -0.33376880
#  AA:       0.51453130   -0.7697110   -0.3337152
#  abs(err): 0.00003739    0.00007839   0.00005360
#  abs(err)
#      (km):   5609          11759         8040
```

---

ydn2md *Convert from year and day number of year to month and day of month*

Description

Convert from year and day number of year to month and day of month

Usage

```
ydn2md(yr, dy)
```

Arguments

yr	4-digit year (like 1988), integer, scalar
dy	day number in year (like 310), integer, scalar or vector

Details

Conversion in the opposite direction is given by function *ymd2dn*.

On error, the function returns

```
m = d = -1
```

.

Value

m	month number, integer (1-12)
d	day of month, integer (1-31)

Author(s)

Adapted from Johns Hopkins University/Applied Physics Laboratory

R adaptation by Arnab Chakraborty (June 2013)

See Also

ymd2dn

Examples

```
#  Find the month/day of days 155 and 255 in the year 2001
#  Result: m=c(6,9)  d=c(4,12), or June 4 and September 12

ydn2md(2001, c(155,255))
```

---

ymd2dn *Convert from year, month, day to day number of year*

Description

Convert from year, month, day to day number of year

Usage

```
ymd2dn(yr, m, d)
```

Arguments

yr	4-digit year (like 1988), integer, scalar or vector
m	month number, integer (1-12), scalar or vector
d	day of month, integer (1-31), scalar or vector

Details

On error, the function returns

```
m = d = -1
```

.

Conversion in the opposite direction is given by function *ydn2md*.

Value

dy	day number in year, integer (1-365), scalar or vector

Author(s)

Written by R. Sterner, JHU/APL 1985

Adapted to IDL by W. Landsman 1998

R adaptation by Arnab Chakraborty (June 2013)

Examples

```
#  Find the days of the year for June 4 and September 12 in the year 2001
#  Result: days 155 and 255

ymd2dn(2001, c(6,9), c(4,12))
```

---

zang *Determine the angular size of an object as a function of redshift*

Description

Determine the angular size of an object as a function of redshift

Usage

```
zang(dl, z, h0, k, lambda0, omega_m, q0)
```

Arguments

`dl`	linear size of the object, in kpc, scalar or vector
`z`	redshift, scalar or vector
`h0`	Hubble expansion parameter, in km/s/Mpc (default = 70.0)
`k`	curvature constant normalized to the closure density (default = 0.0 corresponding to a flat universe)
`lambda0`	cosmological constant normalized to the closure density (default = 0.7)
`omega_m`	matter density normalized to the closure density (default = 0.3)
`q0`	deceleration parameter, scalar corresponding to `-R*(R'')/(R')^2` (default = -0.55)

Details

This function requires an input size in kpc and returns an angular size in arc seconds.

Default cosmology has a Hubble constant of 70 km/s/Mpc, Omega (matter)=0.3 and a normalized cosmological constant Lambda = 0.7. However these values can be changed by the user. Note that

```
Omega_m + lambda0 + k = 1
```

and

```
q0 = 0.5*omega_m - lambda0
```

.

Value

`angsiz`	angular size of the object at the given redshift, in arc seconds, scalar or vector

Author(s)

Written J. Hill STX 1988

R adaptation by Arnab Chakraborty June 2013

See Also

`lumdist`

Examples

```
# What would be the angular size of galaxy of diameter 50 kpc at a redshift
#      of 1.5 in an open universe with Lambda = 0 and Omega (matter) = 0.3.
#      Assume the default Hubble constant value of 70 km/s/Mpc.
# Result: 6.58 arc seconds

zang(50, 1.5, lambda = 0, omega_m = 0.3)

# Plot the angular size of a 50 kpc diameter galaxy as a function of
#      redshift for the default cosmology (Lambda = 0.7, Omega_m=0.3) up to
#      z = 0.5

# zseq = seq(0.01,0.5,length=50)
# plot(zseq, zang(50.0,zseq),xlab='z',ylab='Angular Size (arcsec)')
```

Index