Package 'astronomR'

October 25, 2026

Type Package

Title Cosmic Insights: Statistical Frameworks for Astronomers

Version 0.4.0

Description A comprehensive toolkit for astronomical and cosmological computations. Provides functions for angular coordinate conversions (degrees, hours-minutes-seconds, degrees-minutes-seconds, and radians), Julian Days and the conversion of equatorial (right ascension, declination) to horizontal (altitude, azimuth) coordinates, access to fundamental physical constants ('CODATA' 2018), queries to the Gaia Catalogue of Nearby Stars through the Gaia Archive TAP (Table Access Protocol) service, a 'Lambda-CDM' cosmology model with comoving, luminosity and angular-diameter distances, cosmic time and the age of the universe, early-universe thermal physics including photon density, 'Saha' equation solutions, and a full thermal-cosmology module covering the Hubble rate in the radiation-dominated era, effective relativistic degrees of freedom, entropy density, equilibrium yields, the Boltzmann relic-abundance ('pebble') equation for WIMP freeze-out, the freeze-out temperature solver, and the Peebles equation for hydrogen recombination. Also includes the Drake equation for estimating the number of communicating extraterrestrial civilisations in the Milky Way.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

URL https://github.com/samrit2442/astronomR

BugReports https://github.com/samrit2442/astronomR/issues

Imports httr, jsonlite, stats, stringr (>= 1.5.0), tibble

Depends R (>= 4.1.0)

Suggests deSolve, testthat (>= 3.2.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Samrit Pramanik [aut, cre] (ORCID:
<https://orcid.org/0000-0003-2055-9007>),
Kazi Abu Rousan [aut] (ORCID: <https://orcid.org/0000-0002-7906-2030>)

Maintainer Samrit Pramanik <samrit.2442@gmail.com>

Repository CRAN

Date/Publication 2026-10-25 14:30:29 UTC

Contents

astronomR-package . . . 3
boltzmann_pebble_rhs . . . 4
constants_df . . . 6
constant_value . . . 7
cosmic_time . . . 8
cosmological_distances . . . 9
cosmology_model . . . 11
deg_to_dms . . . 13
deg_to_hms . . . 14
deg_to_rad . . . 15
dms_to_deg . . . 16
drake_equation . . . 17
entropy_density . . . 19
equilibrium_number_density . . . 20
equilibrium_yield . . . 21
freeze_out_xf . . . 22
get_gaia_data . . . 23
g_star_eff . . . 24
hms_to_deg . . . 26
hubble_radiation . . . 27
hubble_ratio . . . 28
julian_day . . . 29
km_to_Mpc . . . 30
Mpc_to_km . . . 31
peebles_rhs . . . 32
photon_energy_density_fn_T . . . 34
photon_energy_density_fn_z . . . 35
photon_number_density_fn_T . . . 36
photon_number_density_fn_z . . . 37
radec_to_altaz . . . 37
rad_to_deg . . . 40
redshift_to_scale_factor . . . 40
Saha_Xe . . . 41
solve_relic_abundance . . . 42

Index **44**

astronomR-package *astronomR: Cosmic Insights: Statistical Frameworks for Astronomers*

Description

A comprehensive toolkit for astronomical and cosmological computations. Provides functions for angular coordinate conversions (degrees, hours-minutes-seconds, degrees-minutes-seconds, and radians), Julian Days and the conversion of equatorial (right ascension, declination) to horizontal (altitude, azimuth) coordinates, access to fundamental physical constants (CODATA 2018), queries to the Gaia Catalogue of Nearby Stars through the Gaia Archive TAP (Table Access Protocol) service, a Lambda-CDM cosmology model with comoving, luminosity and angular-diameter distances, cosmic time and the age of the universe, early-universe thermal physics including photon density, 'Saha' equation solutions, and a full thermal-cosmology module covering the Hubble rate in the radiation-dominated era, effective relativistic degrees of freedom, entropy density, equilibrium yields, the Boltzmann relic-abundance ('pebble') equation for WIMP freeze-out, the freeze-out temperature solver, and the Peebles equation for hydrogen recombination. Also includes the Drake equation for estimating the number of communicating extraterrestrial civilisations in the Milky Way.

Angle conversions

`deg_to_dms`, `dms_to_deg`, `deg_to_hms` and `hms_to_deg` convert between decimal degrees and degrees-minutes-seconds or hours-minutes-seconds; `deg_to_rad` and `rad_to_deg` convert between degrees and radians.

Coordinates

`julian_day` gives the Julian Day of a date-time, and `radec_to_altaz` converts right ascension and declination to altitude and azimuth for an observer at a given place and time.

Physical constants

`constants_df` is a table of physical constants and `constant_value` looks a constant up by name.

Gaia archive

`get_gaia_data` queries the Gaia archive TAP service (an internet connection is required).

Cosmology

`cosmology_model` builds a cosmological model that the other functions take as their `cosmo` argument: `age_of_universe`, `cosmic_time`, `comoving_distance`, `luminosity_distance`, `angular_diameter_distance`, `transverse_comoving_distance`, `radius_of_curvature`, `hubble_distance` and `hubble_ratio`. Distances default to Mpc and times to Gyr. `redshift_to_scale_factor` and `scale_factor_to_redshift` convert between redshift and scale factor; `km_to_Mpc` and `Mpc_to_km` convert distances.

Early-universe thermal physics

Photon energy and number densities as functions of temperature or redshift (photon_energy_density_fn_T, photon_energy_density_fn_z, photon_number_density_fn_T, photon_number_density_fn_z); hydrogen ionization from the Saha equation (Saha_Xe, soln_saha); the radiation-era Hubble rate and thermal bath (hubble_radiation, g_star_eff, entropy_density, equilibrium_number_density, equilibrium_yield); thermal relic freeze-out (boltzmann_pebble_rhs, solve_relic_abundance, freeze_out_xf); and hydrogen recombination with the Peebles equation (peebles_rhs).

Drake equation

drake_equation estimates the number of communicating civilizations in the Milky Way.

Author(s)

Maintainer: Samrit Pramanik <samrit.2442@gmail.com> (ORCID)

Authors:

- Samrit Pramanik <samrit.2442@gmail.com> (ORCID)
- Kazi Abu Rousan <kaziaburousan1999@gmail.com> (ORCID)

See Also

Useful links:

- https://github.com/samrit2442/astronomR
- Report bugs at https://github.com/samrit2442/astronomR/issues

---

boltzmann_pebble_rhs *Boltzmann Relic-Abundance (Pebble) Equation dY/dx*

Description

Returns dY/dx, the right-hand side of the Boltzmann equation governing the evolution of the co-moving yield $Y = n/s$ of a thermally produced relic species:

$$\frac{dY}{dx} = -\frac{\langle \sigma v \rangle \, s}{H \, x} \left(Y^2 - Y_{\mathrm{eq}}^2 \right)$$

Usage

```
boltzmann_pebble_rhs(
  x,
  Y,
  m_GeV,
  sigmav_GeV2,
  g_dof = 2,
  g_star_val = NULL,
  g_star_S_val = NULL
)
```

Arguments

x	Numeric. Dimensionless inverse temperature $x = m/T$.
Y	Numeric. Current yield $Y = n/s$.
m_GeV	Numeric. Particle mass in GeV.
sigmav_GeV2	Numeric. Thermally averaged annihilation cross section $\langle\sigma v\rangle$ in GeV^{-2}, non-negative. A typical WIMP value is $\approx 2.2 \times 10^{-9}$ GeV^{-2}.
g_dof	Integer. Internal degrees of freedom. Default 2.
g_star_val	Numeric. g_* at $T = m/x$. If NULL, computed with g_star_eff(m_GeV / x).
g_star_S_val	Numeric. g_{*S} at $T = m/x$. If NULL, computed with g_star_eff(m_GeV / x, type = "entropy").

Details

This equation is commonly called the **"pebble equation"** in thermal dark-matter / freeze-out literature because the relic density drops away from equilibrium like a stone and then plateaus.

The variable $x \equiv m/T$ is used as the time variable so that the equation is well-behaved through freeze-out ($x \sim$ 20–30). H is evaluated with g_* and both s and Y_{eq} with g_{*S}.

Value

Numeric. dY/dx (dimensionless).

References

Kolb & Turner (1990), Eq. (5.42). Gondolo & Gelmini, Nucl. Phys. B 360 (1991) 145.

See Also

solve_relic_abundance, freeze_out_xf, equilibrium_yield, g_star_eff

Examples

```
sigmav <- 2.2e-9  # GeV^-2 (typical WIMP value)
Y_eq   <- equilibrium_yield(x = 20, m_GeV = 100)
boltzmann_pebble_rhs(x = 20, Y = Y_eq, m_GeV = 100,
                     sigmav_GeV2 = sigmav)
# Above equilibrium the yield is driven back down
boltzmann_pebble_rhs(x = 20, Y = 10 * Y_eq, m_GeV = 100,
                     sigmav_GeV2 = sigmav)
```

`constants_df` *Fundamental Physical Constants in SI and Natural Units*

Description

A table of 19 fundamental physical constants with their CODATA 2018 recommended values in SI units and in natural units.

Usage

```
constants_df
```

Format

A tibble with 19 rows and 6 columns:

name Character. Full name of the constant, e.g. `"Planck Constant"`.

symbol Character. Short symbol, e.g. `"h"`.

value_SI Numeric. CODATA 2018 value in SI units.

unit_SI Character. SI unit of `value_SI`; `"-"` means dimensionless.

value_Natural Numeric. Value in natural units with $\hbar = c = k_B = \epsilon_0 = 1$ (Heaviside-Lorentz) and energies in eV; see Details. `NA` for the Avogadro constant, the Faraday constant and the molar gas constant, which are defined per mole and have no natural-unit value.

unit_Natural Character. Unit of `value_Natural`: `"-"` (dimensionless), `"eV"` or `"eV^(-2)"`; `NA` where `value_Natural` is `NA`.

Details

The SI values are the CODATA 2018 values, published by NIST in May 2019. The speed of light c has been exact since 1983. Since the revision of the SI on 20 May 2019, the Planck constant h, the elementary charge e, the Boltzmann constant k and the Avogadro constant N_A are exact as well, and so are the constants derived from these five alone: the molar gas constant $R = N_A k$, the Faraday constant $F = N_A e$, the Stefan-Boltzmann constant $\sigma = 2\pi^5 k^4/(15h^3c^2)$, the magnetic flux quantum $\Phi_0 = h/(2e)$, the conductance quantum $G_0 = 2e^2/h$ and the electron volt (e joules). The magnetic constant μ_0 is now measured and is no longer exactly $4\pi \times 10^{-7}$ N A^-2.

The natural-unit values use $\hbar = c = k_B = \epsilon_0 = 1$ (the Heaviside-Lorentz convention), with energies in electron volts. In this system $h = 2\pi$ and the elementary charge is $e = \sqrt{4\pi\alpha}$, about 0.303, so that $\Phi_0 = \pi/e$, $G_0 = 4\alpha$ and $\sigma = \pi^2/60$. Masses are given as rest energies mc^2 in eV, and the Rydberg constant as the energy $\hbar c R_\infty$ in eV (the Rydberg energy hcR_∞, about 13.6 eV, is 2π times larger). The gravitational constant becomes $G = 1/M_{\mathrm{Pl}}^2$ in eV^-2, where M_{Pl} is the Planck mass.

Value

A tibble; see the Format section.

Source

CODATA 2018 recommended values of the fundamental physical constants: E. Tiesinga, P. J. Mohr, D. B. Newell and B. N. Taylor (2021), Rev. Mod. Phys. 93, 025010. https://physics.nist.gov/cuu/Constants/

See Also

constant_value to look up a single constant.

Examples

```
constants_df

# The mole-based constants have no natural-unit value
constants_df[is.na(constants_df$value_Natural), c("name", "value_SI")]
```

---

constant_value *Retrieve the Value of a Physical Constant*

Description

Looks up a physical constant in constants_df by name or symbol and returns its value and unit in SI or natural units.

Usage

```
constant_value(constant_name, unit = c("SI", "Natural"))
```

Arguments

constant_name	Character string. Name, symbol or part of the name of the constant, e.g. "Planck Constant", "h" or "planck". Case and leading or trailing white space are ignored.
unit	Character. Unit system of the result: "SI" (default) or "Natural".

Details

Matching ignores case and leading or trailing white space, and is tried in three steps: an exact match on the full name (e.g. "Boltzmann Constant"), then an exact match on the symbol (e.g. "k"), then a partial match on the name (e.g. "planck"). The first step that finds anything decides the result: a single match is returned, and several matches are an error that lists them. The partial match looks for constant_name as a fixed string, not a regular expression, so characters such as "." , "+" or "(" have no special meaning. The spelling "Boltzman" used for two constants before astronomR 0.4.0 is still accepted.

Natural units are $\hbar = c = k_B = \epsilon_0 = 1$ with energies in eV (see constants_df). The Avogadro constant, the Faraday constant and the molar gas constant have no natural-unit value: for these, unit = "Natural" gives NA for both value and unit, with a warning.

Value

A list with elements `name` (character, full name of the constant), `value` (numeric) and `unit` (character, unit of `value`; `"-"` if dimensionless). `value` and `unit` are `NA`, with a warning, when the constant has no value in the requested unit system.

See Also

`constants_df` for the full table.

Examples

```
constant_value("speed of light")
constant_value("k")                          # by symbol
constant_value("planck", unit = "Natural")   # h = 2 pi
constant_value("electron mass", unit = "Natural")

# Several partial matches are an error that lists them
try(constant_value("electron"))

# Mole-based constants have no natural-unit value (NA, with a warning)
constant_value("Avogadro", unit = "Natural")
```

`cosmic_time` *Cosmic Time and Age of the Universe*

Description

`cosmic_time()` gives the age of the universe at redshift `z`: the proper time since the big bang ($a = 0$) for a model created by `cosmology_model`,

$$t(z) = \frac{1}{H_0} \int_0^a \frac{a'\, da'}{\sqrt{\Omega_r + \Omega_m a' + \Omega_k a'^2 + \Omega_\Lambda a'^4}}, \qquad a = \frac{1}{1+z}.$$

`age_of_universe()` is the age today, `cosmic_time(cosmo, 0)`.

Usage

```
cosmic_time(cosmo, z, unit = c("Gyr", "year"))

age_of_universe(cosmo, unit = c("Gyr", "year"))
```

Arguments

`cosmo`	A cosmological model created by `cosmology_model`.
`z`	Numeric vector of finite redshifts, each greater than -1. `NA` values give `NA`.
`unit`	Unit of the result: `"Gyr"` (default, 10^9 Julian years) or `"year"` (Julian years).

Details

Negative redshifts (`-1 < z < 0`) give times in the future. The integral is evaluated numerically with `integrate` (relative tolerance `1e-10`). The Hubble time $1/H_0$ is converted to years with the IAU megaparsec (see `Mpc_to_km`) and the Julian year of 365.25 days.

A pure de Sitter model (`Omega_m = Omega_r = Omega_k = 0`) has no big bang, so its age is infinite and `Inf` is returned. An error is raised if $E(z)^2 \leq 0$ anywhere between the big bang and `z`: for example a bouncing model that never had $a = 0$, or a future redshift that a recollapsing model never reaches.

Value

Times since the big bang in `unit`: gigayears (Gyr) by default, or years. `cosmic_time()` returns an unnamed numeric vector with one element per redshift; `age_of_universe()` returns a single number.

References

Hogg, D. W. (1999). Distance measures in cosmology. https://arxiv.org/abs/astro-ph/9905116

See Also

`cosmology_model`, `hubble_ratio`, `redshift_to_scale_factor`, `comoving_distance`

Examples

```
cosmo <- cosmology_model()
age_of_universe(cosmo)
age_of_universe(cosmo, unit = "year")
cosmic_time(cosmo, c(1100, 10, 1, 0)) # Gyr after the big bang
cosmic_time(cosmo, -0.5)              # when the universe is twice as big
```

`cosmological_distances` *Cosmological Distances*

Description

Distance measures of Hogg (1999) for a model created by `cosmology_model`, vectorized over the redshift `z`.

- `hubble_distance()`: the Hubble distance

$$D_H = \frac{c}{H_0}.$$

- `comoving_distance()`: the line-of-sight comoving distance

$$D_C(z) = D_H \int_0^z \frac{dz'}{E(z')},$$

  with $E(z)$ from `hubble_ratio`.
- `transverse_comoving_distance()`: the transverse comoving distance D_M. It equals D_C in a flat model; in an open model ($\Omega_k > 0$)

$$D_M = \frac{D_H}{\sqrt{\Omega_k}} \sinh\left(\sqrt{\Omega_k}\frac{D_C}{D_H}\right)$$

  and in a closed model ($\Omega_k < 0$)

$$D_M = \frac{D_H}{\sqrt{|\Omega_k|}} \sin\left(\sqrt{|\Omega_k|}\frac{D_C}{D_H}\right).$$

- `luminosity_distance()`: $D_L = (1+z)D_M$.
- `angular_diameter_distance()`: $D_A = D_M/(1+z)$.
- `radius_of_curvature()`: the curvature radius today, $R_0 = D_H/\sqrt{|\Omega_k|}$, which is `Inf` for a flat model.

Usage

```
hubble_distance(cosmo, unit = c("Mpc", "km"))

comoving_distance(cosmo, z, unit = c("Mpc", "km"))

transverse_comoving_distance(cosmo, z, unit = c("Mpc", "km"))

luminosity_distance(cosmo, z, unit = c("Mpc", "km"))

angular_diameter_distance(cosmo, z, unit = c("Mpc", "km"))

radius_of_curvature(cosmo, unit = c("Mpc", "km"))
```

Arguments

`cosmo`	A cosmological model created by `cosmology_model`.
`unit`	Unit of the result: `"Mpc"` (default) or `"km"`.
`z`	Numeric vector of finite, non-negative redshifts. `NA` values give `NA`; `z = 0` gives 0.

Details

The integral for D_C is evaluated numerically with `integrate` (relative tolerance `1e-10`). An error is raised if $E(z)^2 \leq 0$ anywhere between 0 and the largest requested redshift, e.g. beyond the bounce of a bouncing model. In a closed model D_M is not monotonic: it falls back to zero at the antipode, where $\sqrt{|\Omega_k|}D_C/D_H = \pi$, and is negative beyond it.

Value

A single number for `hubble_distance()` and `radius_of_curvature()`; otherwise an unnamed numeric vector with one element per redshift. Distances are in `unit`: megaparsecs (Mpc) by default, or kilometers (km).

References

Hogg, D. W. (1999). Distance measures in cosmology. https://arxiv.org/abs/astro-ph/9905116

See Also

cosmology_model, hubble_ratio, cosmic_time, km_to_Mpc

Examples

```
cosmo <- cosmology_model()
z <- c(0.1, 0.5, 1, 2)
hubble_distance(cosmo)
comoving_distance(cosmo, z)
transverse_comoving_distance(cosmo, z) # equal to D_C: the model is flat
luminosity_distance(cosmo, z)
angular_diameter_distance(cosmo, z)
luminosity_distance(cosmo, 1, unit = "km")
radius_of_curvature(cosmo) # Inf for a flat model

open_model <- cosmology_model(Omega_m = 0.3, Omega_Lambda = 0.6,
                              Omega_k = 0.1)
radius_of_curvature(open_model)
transverse_comoving_distance(open_model, z)
```

---

cosmology_model *Define a Cosmological Model*

Description

Creates a homogeneous Friedmann-Lemaitre-Robertson-Walker (FLRW) model of the universe with matter, radiation, spatial curvature and a cosmological constant, whose expansion rate relative to today is

$$E(z) = \frac{H(z)}{H_0} = \sqrt{\Omega_r(1+z)^4 + \Omega_m(1+z)^3 + \Omega_k(1+z)^2 + \Omega_\Lambda}$$

with $H_0 = 100\,h$ km/s/Mpc. The model is the `cosmo` argument of hubble_ratio, cosmic_time and the distance functions such as comoving_distance. The defaults are the Planck 2015 flat Lambda-CDM parameters (Planck Collaboration 2016, Table 4, last column), without radiation.

Usage

```
cosmology_model(
  h = 0.6774,
  Omega_m = 0.3089,
  Omega_Lambda = 0.6911,
  Omega_r = 0,
  Omega_k = 0,
  T_cmb = 2.7255
)

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

Arguments

h	Dimensionless Hubble parameter, H0 / (100 km/s/Mpc): for example 0.6774 for a Hubble constant of 67.74 km/s/Mpc. Must satisfy 0 < h < 2.
Omega_m	Matter density parameter today (non-negative).
Omega_Lambda	Cosmological-constant (dark-energy) density parameter today. May be negative.
Omega_r	Radiation density parameter today (non-negative).
Omega_k	Curvature density parameter today: 0 for a flat, positive for an open and negative for a closed universe.
T_cmb	CMB temperature today in kelvin (non-negative). Stored for reference only; it is not used in any calculation.
x	An object of class astronomR_cosmology.
...	Ignored.

Details

The curvature type follows the sign of Omega_k: "flat" (Omega_k = 0), "open" (Omega_k > 0) or "closed" (Omega_k < 0). Omega_k is not derived from the other parameters: for a consistent model choose it so that Omega_m + Omega_Lambda + Omega_r + Omega_k = 1, which makes $E(0) = 1$. A warning is given when this sum differs from 1 by more than 0.001.

T_cmb is stored and printed for reference only. It does not set Omega_r, which must be given explicitly (CMB photons alone give $\Omega_\gamma h^2 \approx 2.47 \times 10^{-5}$ for T_cmb = 2.7255).

Value

cosmology_model() returns a list of class astronomR_cosmology with elements h, H0 (the Hubble constant in km/s/Mpc), Omega_m, Omega_Lambda, Omega_r, Omega_k, T_cmb (in K) and curvature ("flat", "open" or "closed"). The print() method prints a summary of the model and returns x invisibly.

References

Planck Collaboration (2016). Planck 2015 results. XIII. Cosmological parameters. Astronomy and Astrophysics, 594, A13. https://arxiv.org/abs/1502.01589

Fixsen, D. J. (2009). The temperature of the cosmic microwave background. The Astrophysical Journal, 707, 916. https://arxiv.org/abs/0911.1955

See Also

hubble_ratio, comoving_distance, luminosity_distance, cosmic_time, age_of_universe

Examples

```
# Planck 2015 flat Lambda-CDM (the defaults)
cosmo <- cosmology_model()
cosmo

# the same model with radiation
cosmology_model(h = 0.6774, Omega_m = 0.3089, Omega_Lambda = 0.6911,
                Omega_r = 8.4e-5)

# an open model: Omega_k = 1 - Omega_m - Omega_Lambda
open_model <- cosmology_model(h = 0.7, Omega_m = 0.3, Omega_Lambda = 0.6,
                              Omega_k = 0.1)
open_model$curvature
```

---

deg_to_dms *Convert Decimal Degrees to Degrees, Minutes, and Seconds (DMS) Format*

Description

Converts a decimal degree angle to the Degrees-Minutes-Seconds (DMS) format commonly used in astronomy for expressing declination and other angles.

Usage

```
deg_to_dms(deg, type = c("cat", "mat"), digit = 5)
```

Arguments

deg	Numeric vector of angles in decimal degrees. All values must be between -90 and 90 (the declination range). NA values give NA in the output.
type	Character string specifying the output format. Options are: • "cat" (default): Prints the angles in DMS notation, such as "+45\u00B030'15.12\"", separated by spaces and followed by a newline. • "mat": Returns a matrix with columns for sign, degrees, minutes, and seconds.
digit	Whole number from 0 to 10: the number of decimal places of the seconds. Default is 5.

Details

The angle is rounded once, to the nearest 10^{-digit} arcsecond, and then split exactly into degrees, minutes and seconds, so the minutes and seconds are always below 60: `deg_to_dms(10.1)` gives 10 degrees, 6 minutes and 0 seconds. Seconds are written in fixed notation without trailing zeros. The sign is `"-"` only when the angle is negative and still non-zero after rounding, so `deg_to_dms(-1e-9)` prints `+0` degrees.

Value

When `type = "cat"`, prints the DMS string and returns `NULL` invisibly. When `type = "mat"`, returns a character matrix with one row per angle and columns `SIGN` (`"+"` or `"-"`), `DEG` (degrees, 0 to 90), `MIN` (arcminutes, 0 to 59) and `SEC` (arcseconds, below 60). The row for an `NA` angle is all `NA`.

See Also

`dms_to_deg` for the reverse conversion, `deg_to_hms` for right ascension, and `deg_to_rad`.

Examples

```
deg_to_dms(45.5042)
deg_to_dms(c(10.1, -0.5))
deg_to_dms(-12.5, type = "mat")
deg_to_dms(c(10.25, 45.5), type = "mat")

# Paste the "mat" columns into strings to convert them back
x <- deg_to_dms(c(-0.5, 45.5042), type = "mat")
dms_to_deg(paste0(x[, "SIGN"], x[, "DEG"], "d", x[, "MIN"], "m",
                  x[, "SEC"], "s"))
```

---

deg_to_hms *Convert Decimal Degrees to Hours, Minutes, and Seconds (HMS) Format*

Description

Converts a decimal degree angle to Hours-Minutes-Seconds (HMS) format, which is used in astronomy to express Right Ascension (RA).

Usage

```
deg_to_hms(deg, type = c("cat", "mat"), digit = 5)
```

Arguments

`deg`	Numeric vector of angles in decimal degrees. Any finite value is accepted and wrapped into $[0, 360)$. `NA` values give `NA` in the output.
`type`	Character string specifying the output format. Options are:

- `"cat"` (default): Prints a formatted string (e.g., `3H0M0S`), separated by spaces for several angles and followed by a newline.
- `"mat"`: Returns a matrix with columns for hours, minutes, and seconds.

`digit` Whole number from 0 to 10: the number of decimal places of the seconds. Default is 5.

Details

Any finite angle is first wrapped into $[0, 360)$ degrees, so `-15` gives 23 hours and `375` gives 1 hour (one hour is 15 degrees). The angle is then rounded once, to the nearest 10^{-digit} second of time, and split exactly, so the minutes and seconds are always below 60 and the hours run from 0 to 23 (an angle that rounds up to 24 hours is shown as 0 hours). Seconds are written in fixed notation without trailing zeros.

Value

When `type = "cat"`, prints the HMS string and returns `NULL` invisibly. When `type = "mat"`, returns a numeric matrix with one row per angle and columns `HRS` (hours, 0 to 23), `MIN` (minutes of time, 0 to 59) and `SEC` (seconds of time, below 60). The row for an `NA` angle is all `NA`.

See Also

`hms_to_deg` for the reverse conversion, `deg_to_dms` for declination, and `deg_to_rad`.

Examples

```
deg_to_hms(deg = 45)
deg_to_hms(deg = 45, type = "mat")
deg_to_hms(deg = 177.74208, digit = 3)
deg_to_hms(c(-15, 20, 375))

# The "mat" columns convert straight back
x <- deg_to_hms(177.74208, type = "mat")
hms_to_deg(x[, "HRS"], x[, "MIN"], x[, "SEC"])
```

---

deg_to_rad *Convert Degrees to Radians*

Description

Converts angles from degrees to radians. Supports vectorized input.

Usage

```
deg_to_rad(deg)
```

Arguments

`deg` Numeric. The angle(s) in degrees to convert to radians. `NA` values give `NA`.

Value

Numeric. The corresponding angle(s) in radians, $deg \times \pi/180$.

See Also

`rad_to_deg` for the reverse conversion; `deg_to_dms` and `deg_to_hms` for sexagesimal output.

Examples

```
deg_to_rad(180)
deg_to_rad(c(0, 90, 180))
```

`dms_to_deg` *Convert Degrees, Minutes, and Seconds (DMS) to Decimal Degrees*

Description

Converts an angle expressed in Degrees-Minutes-Seconds (DMS) format to decimal degrees. Accepts either separate numeric arguments or a single formatted string.

Usage

```
dms_to_deg(d, m = 0, s = 0, digit = 5)
```

Arguments

`d`	Numeric or character vector. The degrees component, whose sign is the sign of the angle, or DMS strings such as `"12\u00B034'56\""` (see Details), in which case `m` and `s` must not be supplied.
`m`	Numeric vector. The minutes component, from 0 to less than 60. Default is `0`.
`s`	Numeric vector. The seconds component, from 0 to less than 60. Default is `0`.
`digit`	Whole number from 0 to 10: the number of decimal places to round the result to. Default is 5.

Details

With numeric input, `d`, `m` and `s` are recycled to a common length and the result is $\pm(|d| + m/60 + s/3600)$, with the sign taken from `d`. For a negative angle smaller than one degree, use R's negative zero: `dms_to_deg(-0, 30)` is `-0.5`, while `dms_to_deg(0, 30)` is `0.5`.

A character `d` is parsed element by element. Each string holds an optional sign (`"+"`, `"-"` or the Unicode minus sign U+2212), the degrees, and optionally the minutes and then the seconds. The degrees may be followed by the degree sign `"\u00B0"`, `"d"`, `"D"` or `":"`; the minutes by `"'"`, the Unicode prime U+2032, `"m"`, `"M"` or `":"`; and the seconds by `"\""`, the Unicode double prime U+2033, `"s"` or `"S"`. Spaces alone may also separate the fields. Only the last field may have a decimal part. Accepted strings include `"12\u00B034'56\""`, `"-0\u00B030'00\""`,

`"+12\u00B034'56.7\""`, `"12d34m56s"`, `"12:34:56"`, `"12 34 56"`, `"12\u00B034'"` and `"12.5\u00B0"`. A string that cannot be parsed gives `NA` with a warning; `NA` gives `NA` silently.

For both kinds of input the minutes and seconds must be in $[0, 60)$ and the angle in $[-90, 90]$ degrees (the declination range, as in `deg_to_dms`); other values are an error.

Value

Numeric vector of angles in decimal degrees (between -90 and 90), rounded to `digit` decimal places.

See Also

`deg_to_dms` for the reverse conversion, `hms_to_deg` for right ascension, and `deg_to_rad`.

Examples

```
dms_to_deg(d = 12, m = 34, s = 56)
dms_to_deg(d = "12\u00B034'56\"", digit = 3)

# The sign comes from 'd'; use -0 for angles between -1 and 0 degrees
dms_to_deg(c(-12, -0, 0), m = 30)

# Several string formats, and an NA
dms_to_deg(c("12d34m56s", "12:34:56", "-0\u00B030'", "12.5\u00B0", NA))
```

---

drake_equation *Estimate the Number of Communicating Civilizations via the Drake Equation*

Description

Computes **N**, the estimated number of active, technologically advanced civilizations capable of interstellar communication in the Milky Way, using the Drake equation:

$$N = R_* \times f_p \times n_e \times f_l \times f_i \times f_c \times L$$

Usage

```
drake_equation(
  R_star = 1.5,
  fp = 1,
  ne = 0.4,
  fl = 1,
  fi = 1,
  fc = 0.1,
  L = 1000
)
```

Arguments

`R_star`	Numeric. Average rate of star formation per year in the Milky Way (stars yr^{-1}). Default: `1.5`.
`fp`	Numeric. Fraction of stars that have planetary systems. Must be in $[0, 1]$. Default: `1.0` (current exoplanet surveys suggest nearly all Sun-like stars host planets).
`ne`	Numeric. Mean number of planets per planetary system that could potentially support life (Earth-like / habitable-zone planets). Default: `0.4`.
`fl`	Numeric. Fraction of suitable planets on which life actually appears. Must be in $[0, 1]$. Default: `1.0`.
`fi`	Numeric. Fraction of life-bearing planets on which intelligent life evolves. Must be in $[0, 1]$. Default: `1.0`.
`fc`	Numeric. Fraction of intelligent civilizations that develop detectable communication technology. Must be in $[0, 1]$. Default: `0.1`.
`L`	Numeric. Mean longevity (years) of a communicating civilization. Default: `1000`.

Details

Default values reflect widely cited contemporary estimates (see References). All fraction parameters must lie in $[0, 1]$.

Value

A named list with the following elements:

`N` Numeric. Estimated number of communicating civilizations.

`inputs` Named numeric vector of all seven input parameters.

`equation` Character. Human-readable form of the equation with values substituted, with embedded line breaks; print it with `cat()`.

`interpretation` Character. A qualitative interpretation of `N` spanning six tiers from *effectively zero* ($N < 0.001$) to *galaxy teeming with life* ($N \geq 10\,000$).

References

Drake, F. D. (1961). Discussion at the first SETI conference, Green Bank, West Virginia. *NRAO*.

Vakoch, D. A., & Dowd, M. F. (Eds.) (2015). *The Drake Equation: Estimating the Prevalence of Extraterrestrial Life through the Ages*. Cambridge University Press.

Examples

```
# Using all defaults (optimistic estimate)
drake_equation()

# Pessimistic "Rare Earth" scenario
drake_equation(R_star = 1.5, fp = 1.0, ne = 0.4,
               fl = 0.01, fi = 0.01, fc = 0.01, L = 304)
```

```
# Custom values
result <- drake_equation(R_star = 3, fp = 0.9, ne = 0.5,
                         fl = 0.5, fi = 0.1, fc = 0.1, L = 10000)
result$N

# cat() renders the equation string with real line breaks
cat(result$equation, "\n", sep = "")
result$interpretation
```

`entropy_density` *Entropy Density of the Thermal Bath*

Description

Returns the entropy density $s(T)$ of the cosmological radiation bath:

$$s(T) = \frac{2\pi^2}{45}\, g_{*S}(T)\, T^3$$

where g_{*S} is the effective number of entropy degrees of freedom.

Usage

```
entropy_density(T_GeV, g_star_S = NULL)
```

Arguments

`T_GeV`	Numeric. Temperature in GeV.
`g_star_S`	Numeric. Effective entropy degrees of freedom $g_{*S}(T)$. If `NULL` (default), computed with `g_star_eff(T_GeV, type = "entropy")`.

Value

Numeric. Entropy density s in GeV^3.

References

Kolb & Turner (1990), Eq. (3.65).

See Also

`g_star_eff`, `equilibrium_yield`

Examples

```
entropy_density(0.1)
entropy_density(1, g_star_S = 106.75)
```

`equilibrium_number_density` *Equilibrium Number Density of a Massive Particle Species*

Description

Computes the equilibrium number density $n^{\mathrm{eq}}(T)$ for a non-relativistic species in the Maxwell-Boltzmann limit:

$$n^{\mathrm{eq}}(T) = g \left(\frac{mT}{2\pi}\right)^{3/2} e^{-m/T}$$

Usage

```
equilibrium_number_density(T_GeV, m_GeV, g_dof = 2)
```

Arguments

`T_GeV`	Numeric. Temperature in GeV.
`m_GeV`	Numeric. Particle mass in GeV.
`g_dof`	Integer. Internal degrees of freedom. Default 2.

Value

Numeric. Equilibrium number density in GeV^3.

References

Kolb & Turner (1990), Eq. (5.25).

See Also

`equilibrium_yield`

Examples

```
equilibrium_number_density(T_GeV = 0.01, m_GeV = 0.1, g_dof = 2)
```

equilibrium_yield *Equilibrium Yield Y_eq as a Function of x = m/T*

Description

The yield $Y \equiv n/s$ is the comoving number density (number per comoving entropy). In equilibrium:

$$Y^{\rm eq}(x) = \frac{45\, g}{4\pi^4\, g_{*S}}\, x^2\, K_2(x)$$

where $x = m/T$ and K_2 is the modified Bessel function of the second kind of order 2.

Usage

```
equilibrium_yield(x, m_GeV, g_dof = 2, g_star_S = NULL)
```

Arguments

x	Numeric. Dimensionless parameter $x = m/T$.
m_GeV	Numeric. Particle mass in GeV.
g_dof	Integer. Internal degrees of freedom. Default 2.
g_star_S	Numeric. Entropy degrees of freedom. If NULL (default), computed with g_star_eff(m_GeV / x, type = "entropy").

Value

Numeric. Dimensionless equilibrium yield $Y^{\rm eq}$.

References

Kolb & Turner (1990), Eq. (5.34). Gondolo & Gelmini, Nucl. Phys. B 360 (1991) 145.

See Also

equilibrium_number_density, entropy_density, boltzmann_pebble_rhs

Examples

```
equilibrium_yield(x = 3, m_GeV = 100, g_dof = 2)
equilibrium_yield(x = 25, m_GeV = 100)
```

`freeze_out_xf` *Approximate Freeze-Out Parameter x_f = m / T_f*

Description

Estimates the freeze-out inverse temperature $x_f = m/T_f$ using the iterative approximation from the condition $\Gamma \sim H$ (Kolb & Turner 1990):

$$x_f = \ln\left(c(c+2)\,\sqrt{\frac{45}{8}}\,\frac{g\,m\,m_{\rm Pl}\,\langle\sigma v\rangle}{2\pi^3\,\sqrt{g_*}\,x_f^{1/2}} \right)$$

Usage

```
freeze_out_xf(
  m_GeV,
  sigmav_GeV2,
  g_dof = 2,
  c_coeff = 0.5,
  g_star = 106.75,
  tol = 1e-06,
  max_iter = 100
)
```

Arguments

`m_GeV`	Numeric. Particle mass in GeV.
`sigmav_GeV2`	Numeric. $\langle\sigma v\rangle$ in GeV^{-2}.
`g_dof`	Integer. Internal degrees of freedom. Default 2.
`c_coeff`	Numeric. Order-unity coefficient ($c \approx 0.5$ for s-wave). Default 0.5.
`g_star`	Numeric. g_* at freeze-out. Default 106.75.
`tol`	Numeric. Convergence tolerance. Default 1e-6.
`max_iter`	Integer. Maximum iterations. Default 100.

Details

Here $m_{\rm Pl} = G^{-1/2} = \sqrt{8\pi}\,M_{\rm Pl} \approx 1.2207 \times 10^{19}$ GeV is the full (non-reduced) Planck mass, for which the coefficient $\sqrt{45/8}/(2\pi^3)$ is defined; $M_{\rm Pl} = 2.435 \times 10^{18}$ GeV is the reduced Planck mass used by `hubble_radiation`.

The equation is solved by fixed-point iteration, starting from the logarithm of the argument without the $x_f^{-1/2}$ factor. The approximation describes non-relativistic freeze-out and needs a log argument well above e ($x_f \gg 1$); an error is raised if the argument would be ≤ 1, which happens for very small masses or cross sections. Typical WIMP values lie in the range $x_f \approx 20$–30: with the defaults, $m = 100$ GeV and $\langle\sigma v\rangle = 2.2 \times 10^{-9}$ GeV^{-2} give $x_f \approx 22.38$, i.e. $T_f \approx 4.47$ GeV.

Value

Numeric vector of the dimensionless freeze-out parameter x_f, recycled to the common length of `m_GeV`, `sigmav_GeV2`, `g_dof`, `c_coeff` and `g_star`. A warning is given if the iteration has not converged after `max_iter` steps.

References

Kolb & Turner (1990), Eq. (5.44). Jungman, Kamionkowski & Griest, Phys. Rep. 267 (1996) 195.

See Also

`solve_relic_abundance`, `boltzmann_pebble_rhs`

Examples

```
sigmav <- 2.2e-9  # GeV^-2
x_f <- freeze_out_xf(m_GeV = 100, sigmav_GeV2 = sigmav)
cat("x_f =", x_f, "  T_f =", 100 / x_f, "GeV\n")
freeze_out_xf(m_GeV = c(10, 100, 1000), sigmav_GeV2 = sigmav)
```

`get_gaia_data` *Query Gaia Archive Data*

Description

Queries the Gaia Archive TAP service for stars in the Gaia Catalogue of Nearby Stars (GCNS), the census of objects within 100 pc of the Sun built from Gaia Early Data Release 3 (EDR3) and served as the table `external.gaiaedr3_gcns_main_1`. Returns the requested columns for the rows that match a filter condition.

Usage

```
get_gaia_data(vars, condition, timeout = 60)
```

Arguments

`vars`	A single character string giving the columns to retrieve, separated by commas (e.g., `"source_id, ra, dec"`).
`condition`	A single character string giving the ADQL `WHERE` clause used to filter rows (e.g., `"parallax > 10"`).
`timeout`	A single number, at least `0.001`: the maximum time, in seconds, to wait for the whole request, including the time the archive spends running the query. Default is `60`.

Details

This function sends the synchronous ADQL query `SELECT vars FROM external.gaiaedr3_gcns_main_1 WHERE condition` to the Gaia Archive TAP service at https://gea.esac.esa.int/tap-server/tap/sync. An internet connection is required.

The function fails gracefully: problems on the remote side (no connection, a timeout, an HTTP error status such as a rejected query, or a malformed response) never raise an error or a warning. Instead a message explains what went wrong, including the HTTP status and the archive's error text when there is one, and `NULL` is returned invisibly, so check the result with `is.null()` before using it. Invalid arguments are still errors.

The columns of the result are named by splitting `vars` at its commas. If the archive returns fewer columns than this gives names, for example because an entry of `vars` is a function call with several arguments, the result is also `NULL` with a message.

`timeout` caps how long the function can block. Queries that return many rows can need more than the default 60 seconds; increase `timeout` for them. Opening the connection is also limited, separately, by the curl package, and a larger `timeout` does not extend that limit.

Value

A data frame with one row per matching star and one column per entry of `vars`, named as written there. Values are in the units of the GCNS table (e.g., `ra` and `dec` in degrees, `parallax` in mas).

If the query runs but no rows match, a data frame with zero rows and the same column names (all numeric) is returned, with a message.

If the Gaia Archive cannot be reached, does not answer in time, returns an HTTP error status, or sends a response that is not valid JSON or does not hold a table of results with a column for every entry of `vars`, a message gives the reason and `NULL` is returned invisibly.

See Also

`deg_to_hms` and `deg_to_dms` to express the returned `ra` and `dec` in sexagesimal form.

Examples

```
vars <- "source_id, ra, dec, phot_bp_mean_mag, phot_rp_mean_mag, parallax"
condition <- "parallax > 40"
result <- get_gaia_data(vars, condition)
head(result)
```

---

g_star_eff *Effective Relativistic Degrees of Freedom g_*(T)*

Description

Returns a step-function approximation of the Standard Model effective number of relativistic degrees of freedom as a function of temperature: $g_*(T)$, which sets the radiation energy density $\rho = (\pi^2/30)\, g_* T^4$, or $g_{*S}(T)$, which sets the entropy density $s = (2\pi^2/45)\, g_{*S} T^3$.

Usage

```
g_star_eff(T_GeV, type = c("energy", "entropy"))
```

Arguments

`T_GeV`	Numeric vector of positive temperatures in GeV, without missing values.
`type`	Character. `"energy"` (default) for g_*, the degrees of freedom in the energy density, or `"entropy"` for g_{*S}, those in the entropy density.

Details

Each relativistic species i with g_i internal degrees of freedom and temperature T_i contributes

$$g_* = \sum_{\text{bosons}} g_i \left(\frac{T_i}{T}\right)^4 + \frac{7}{8} \sum_{\text{fermions}} g_i \left(\frac{T_i}{T}\right)^4$$

and likewise for g_{*S} with the third power of T_i/T. This function uses the instantaneous-decoupling step approximation: a species counts fully while T is above its mass threshold and not at all below it. The bins are half-open, $T_{\text{low}} < T \leq T_{\text{high}}$, with T in GeV:

T (GeV)	g_*	g_*S	removed relative to the row above
above 173	106.75	106.75	none (all SM particles)
125 to 173	96.25	96.25	top quark (173 GeV)
80.4 to 125	95.25	95.25	Higgs boson (125 GeV)
4.18 to 80.4	86.25	86.25	W and Z bosons (both at 80.4 GeV)
1.777 to 4.18	75.75	75.75	bottom quark (4.18 GeV)
1.27 to 1.777	72.25	72.25	tau lepton (1.777 GeV)
0.15 to 1.27	61.75	61.75	charm quark (1.27 GeV)
0.1396 to 0.15	17.25	17.25	QCD transition (150 MeV)
0.1057 to 0.1396	14.25	14.25	pions (139.6 MeV)
5.11e-4 to 0.1057	10.75	10.75	muon (105.7 MeV)
up to 5.11e-4	3.36	3.91	electron and positron (511 keV)

At the QCD transition the u, d and s quarks and the gluons are replaced by the three pions. The last row counts photons and three neutrino species that decoupled before e+e- annihilation and are colder by $(4/11)^{1/3}$: $g_* = 2 + \frac{7}{8} \cdot 6 \cdot (4/11)^{4/3} = 3.36$ and $g_{*S} = 2 + \frac{7}{8} \cdot 6 \cdot \frac{4}{11} = 3.91$. At higher temperatures all species share one temperature and $g_{*S} = g_*$. The step values agree with the instantaneous-decoupling values of Husdal (2016); the true $g_*(T)$ is smooth across each threshold, most visibly around the QCD transition.

Value

Numeric vector, the same length as `T_GeV`: the dimensionless g_* or g_{*S}.

References

Husdal, L. (2016). On Effective Degrees of Freedom in the Early Universe. Galaxies, 4, 78. arXiv:1609.04979.

Kolb, E. W. & Turner, M. S. (1990). *The Early Universe*. Addison-Wesley, Chapter 3.

See Also

hubble_radiation, entropy_density, equilibrium_yield, boltzmann_pebble_rhs

Examples

```
g_star_eff(100)                    # 95.25: below the Higgs, above W/Z
g_star_eff(1e-3)                   # 10.75: photons, e+-, 3 neutrinos
g_star_eff(1e-4)                   # 3.36: after e+e- annihilation
g_star_eff(1e-4, type = "entropy")  # 3.91
g_star_eff(c(500, 1, 0.01))        # 106.75 61.75 10.75
```

---

hms_to_deg *Convert Hours, Minutes, and Seconds (HMS) to Decimal Degrees*

Description

Converts an angle expressed in Hours-Minutes-Seconds (HMS) format to decimal degrees. Accepts either separate numeric arguments or a single formatted string.

Usage

```
hms_to_deg(h, m = 0, s = 0, digit = 5)
```

Arguments

h	Numeric or character vector. The hours component, or HMS strings such as "03h15m30s" (see Details), in which case m and s must not be supplied.
m	Numeric vector. The minutes component, from 0 to less than 60. Default is 0.
s	Numeric vector. The seconds component, from 0 to less than 60. Default is 0.
digit	Whole number from 0 to 10: the number of decimal places to round the result to. Default is 5.

Details

With numeric input, h, m and s are recycled to a common length and the result is $15(h + m/60 + s/3600)$ degrees. h must be in $[0, 24]$, m and s in $[0, 60)$, and the total may not exceed 24 hours.

A character h is parsed element by element, ignoring case: the hours, then optionally the minutes and then the seconds, as in "03h15m30s", "3h 15m 30s", "3H15M30S", "03:15:30", "3 15 30", "3h15m30.5s", "3h15m" or "3h". Only the last field may have a decimal part, and a leading "+" is allowed. The parsed values must lie in the same ranges as numeric input (so a negative value is an error). A string that cannot be parsed gives NA with a warning; NA gives NA silently.

Value

Numeric vector of angles in decimal degrees (between 0 and 360), rounded to digit decimal places.

See Also

deg_to_hms for the reverse conversion, dms_to_deg for declination, and deg_to_rad.

Examples

```
hms_to_deg(h = 3, m = 15, s = 30)
hms_to_deg(h = 3, m = 15, s = 30.123)
hms_to_deg(h = "03h15m30s")
hms_to_deg(c("3h 15m 30s", "03:15:30", "3h15m", "3h", NA))
```

---

hubble_radiation *Hubble Rate in the Radiation-Dominated Era*

Description

Computes the Hubble expansion rate $H(T)$ during the radiation-dominated epoch using the Friedmann equation:

$$H(T) = \sqrt{\frac{\pi^2}{90}\, g_*(T)}\; \frac{T^2}{M_{\rm Pl}}$$

where $M_{\rm Pl} = 2.435 \times 10^{18}$ GeV is the reduced Planck mass and $g_*(T)$ is the effective number of relativistic degrees of freedom.

Usage

```
hubble_radiation(T_GeV, g_star = 106.75)
```

Arguments

T_GeV	Numeric. Plasma temperature in GeV.
g_star	Numeric. Effective relativistic degrees of freedom $g_*(T)$. Defaults to 106.75 (Standard Model value above the electroweak scale); see g_star_eff for other temperatures.

Value

Numeric. Hubble rate H in GeV (natural units).

References

Kolb, E. W. & Turner, M. S. (1990). *The Early Universe*. Addison-Wesley, Eq. (3.46).

See Also

g_star_eff, boltzmann_pebble_rhs

Examples

```
# H at T = 100 GeV with SM degrees of freedom
hubble_radiation(100)

# H at T = 1 MeV (neutrino decoupling era), g* ~ 10.75
hubble_radiation(1e-3, g_star = 10.75)
```

`hubble_ratio` *Dimensionless Hubble Parameter E(z)*

Description

Computes the expansion rate relative to today for a model created by `cosmology_model`, that is $E(z) = H(z)/H_0$ with

$$E(z) = \sqrt{\Omega_r(1+z)^4 + \Omega_m(1+z)^3 + \Omega_k(1+z)^2 + \Omega_\Lambda}.$$

An error is raised where $E(z)^2 \leq 0$, i.e. where the model has no real expansion rate.

Usage

```
hubble_ratio(cosmo, z)
```

Arguments

`cosmo`	A cosmological model created by `cosmology_model`.
`z`	Numeric vector of finite redshifts, each greater than -1 (negative values lie in the future). `NA` values give `NA`.

Value

Numeric vector of the same length as `z` (no names): the dimensionless ratio $H(z)/H_0$. Multiply by `cosmo$H0` to get $H(z)$ in km/s/Mpc.

References

Hogg, D. W. (1999). Distance measures in cosmology. https://arxiv.org/abs/astro-ph/9905116

See Also

`cosmology_model`, `hubble_distance`, `comoving_distance`, `cosmic_time`

Examples

```
cosmo <- cosmology_model()
hubble_ratio(cosmo, c(0, 0.5, 1, 2))
cosmo$H0 * hubble_ratio(cosmo, 1) # H(z = 1) in km/s/Mpc
```

`julian_day` *Julian Day of a Date-Time*

Description

Converts date-times to Julian Days (JD): the continuous count of days, with the fraction of the day, since noon Universal Time on 1 January 4713 BC of the proleptic Julian calendar.

Usage

```
julian_day(datetime)
```

Arguments

`datetime`	A `POSIXct`, `POSIXlt` or `Date` vector, or a character vector. Character strings are always read as UTC and must be a year-first date, optionally followed by a time: `"2000-01-01"`, `"2000-01-01 12:00"`, `"2000-01-01 12:00:00.5"`, with `"/"` instead of `"-"` or `"T"` instead of the space also accepted, and an optional trailing `"Z"`. Each element is read on its own, so dates and date-times can be mixed. Anything else, including time-zone offsets such as `"+05:30"` and day-first dates, is an error; for local clock times build a `POSIXct` with the right `tz` yourself. `Date` values refer to midnight UTC.

Details

The Julian Day is computed from the instant itself,

$$JD = \frac{t}{86400} + 2440587.5,$$

where t is the number of seconds since 1970-01-01 00:00:00 UTC, i.e. `as.numeric(as.POSIXct(datetime))`. The result therefore does not depend on the time zone a date-time is displayed in, and it includes seconds and fractions of a second.

Calendar dates are read on the proleptic Gregorian calendar, as everywhere in R, so for dates before 1582-10-15 the result differs from the Julian-calendar convention used by Meeus (1998). Leap seconds and the difference UT1 - UTC (always less than 0.9 s) are ignored: UTC is treated as Universal Time.

Value

A numeric vector of Julian Days (in days), the same length as `datetime`. Missing date-times (including an all-`NA` logical vector) give `NA`.

References

Meeus, J. (1998). *Astronomical Algorithms*, 2nd edition. Willmann-Bell, Richmond, Virginia. Chapter 7.

See Also

radec_to_altaz, which uses the Julian Day to obtain the sidereal time.

Examples

```
# The J2000.0 epoch
julian_day(as.POSIXct("2000-01-01 12:00:00", tz = "UTC"))

# Meeus (1998) Example 7.a: 1957 October 4.81 UT
print(julian_day("1957-10-04 19:26:24"), digits = 10)

# The same instant displayed in two time zones has one Julian Day
noon_new_york <- as.POSIXct("2024-07-04 12:00:00", tz = "America/New_York")
print(julian_day(noon_new_york), digits = 10)
print(julian_day("2024-07-04 16:00:00"), digits = 10)

# Dates refer to midnight UTC
print(julian_day(as.Date("2024-01-01")), digits = 10)
```

---

km_to_Mpc *Convert Kilometers to Megaparsecs*

Description

Converts distances from kilometers (km) to megaparsecs (Mpc) using the exact IAU definitions of the astronomical unit, 1 au = 149597870.7 km (IAU 2012 Resolution B2), and of the parsec, 1 pc = $648000/\pi$ au (IAU 2015 Resolution B2):

$$1\ \mathrm{Mpc} = 10^6 \times \frac{648000}{\pi} \times 149597870.7\ \mathrm{km} = 3.0856775814913673 \times 10^{19}\ \mathrm{km}.$$

Usage

```
km_to_Mpc(km)
```

Arguments

km	Numeric vector of distances in kilometers.

Value

Numeric vector of the same length as km: the distances in megaparsecs (Mpc).

See Also

Mpc_to_km for the inverse conversion; hubble_distance and the other distance functions, which return Mpc by default.

Examples

```
km_to_Mpc(3.0856775814913673e19) # 1 Mpc
km_to_Mpc(c(1e19, 1e20, 1e21))
```

`Mpc_to_km` *Convert Megaparsecs to Kilometers*

Description

Converts distances from megaparsecs (Mpc) to kilometers (km) using the exact IAU definitions of the astronomical unit, 1 au = 149597870.7 km (IAU 2012 Resolution B2), and of the parsec, 1 pc = $648000/\pi$ au (IAU 2015 Resolution B2):

$$1\ \mathrm{Mpc} = 10^6 \times \frac{648000}{\pi} \times 149597870.7\ \mathrm{km} = 3.0856775814913673 \times 10^{19}\ \mathrm{km}.$$

Usage

```
Mpc_to_km(mpc)
```

Arguments

`mpc`	Numeric vector of distances in megaparsecs.

Value

Numeric vector of the same length as `mpc`: the distances in kilometers (km).

See Also

`km_to_Mpc` for the inverse conversion.

Examples

```
Mpc_to_km(1) # 3.0856775814913673e19 km
Mpc_to_km(c(0.5, 1, 4425.6))
```

peebles_rhs *Peebles Equation for Hydrogen Recombination*

Description

Computes the derivative of the free-electron fraction during cosmological hydrogen recombination from the Peebles (1968) equation, with the Peebles C-factor in the form used by RECFAST (Seager, Sasselov & Scott 1999).

Usage

```
peebles_rhs(
  lna,
  xe,
  H,
  n_H,
  alpha_B,
  beta_B,
  lambda_alpha = 1.21567e-07,
  Lambda_2s1s = 8.22458,
  neutral_floor = 1e-30,
  T0 = 2.7255
)
```

Arguments

lna	Natural logarithm of the scale factor, `log(a) = -log(1 + z)`: a single finite number. It sets the radiation temperature $T = T_0 e^{-\ln a}$.
xe	Free-electron fraction per hydrogen nucleus, $0 \le x_e \le 1$.
H	Hubble expansion rate at `lna`, in s^{-1}.
n_H	Total hydrogen-nuclei number density at `lna`, in m^{-3}.
alpha_B	Case-B recombination coefficient at temperature T, in $\mathrm{m}^3\ \mathrm{s}^{-1}$.
beta_B	Photoionization rate from the $n = 2$ state at temperature T, in s^{-1}.
lambda_alpha	Lyman-alpha wavelength in metres; it also sets the $n = 2$ excitation energy $E_{21} = hc/\lambda_\alpha$. Default is 121.567 nm (`121.567e-9` m).
Lambda_2s1s	Hydrogen $2s \to 1s$ two-photon decay rate in s^{-1}. The standard value is approximately 8.22458 s^{-1}.
neutral_floor	Small lower limit for neutral-hydrogen density, used only to avoid division by zero at $x_e = 1$. Default `1e-30`.
T0	Radiation (CMB) temperature today in K. Default 2.7255 (Fixsen 2009).

Details

The Peebles equation for the evolution of the free-electron fraction is

$$\frac{dx_e}{d\ln a} = \frac{C}{H}\left[\beta_B(1-x_e)\,e^{-E_{21}/(k_B T)} - n_H \alpha_B x_e^2\right]$$

where $T = T_0 e^{-\ln a} = T_0(1+z)$ and $E_{21} = hc/\lambda_\alpha \approx 10.2$ eV. The Boltzmann factor gives the population of the $n=2$ level, from which β_B photoionizes, relative to the ground state.

The Peebles C-factor

$$C = \frac{\Lambda_{2s1s} + \lambda_\alpha^{\rm esc}}{\Lambda_{2s1s} + \lambda_\alpha^{\rm esc} + \beta_B}$$

accounts for the fact that a hydrogen atom formed in an excited state can be photoionized again before reaching the stable ground state via the two-photon or Lyman-alpha escape channels.

The Lyman-alpha escape rate is

$$\lambda_\alpha^{\rm esc} = \frac{8\pi H}{\lambda_\alpha^3\, n_{1s}}$$

where $n_{1s} \approx n_H(1-x_e)$ is the ground-state hydrogen density.

The matter temperature is assumed equal to the radiation temperature, which holds while Compton scattering keeps them coupled ($z \gtrsim 200$); `alpha_B` and `beta_B` should be evaluated at that T. All inputs are single numbers in SI units (metres, seconds, Kelvin). Note that H here is in s^{-1}, whereas `hubble_radiation` returns it in GeV.

The example uses inputs at $z = 1100$ ($T = 3001$ K) for a flat cosmology with $H_0 = 67.66$ km s^{-1} Mpc^{-1}, $\Omega_m = 0.3111$, $\Omega_r = 9.14\times10^{-5}$, $\Omega_b h^2 = 0.02242$ and helium mass fraction $Y_p = 0.2454$ (Planck 2018): $H = H_0\sqrt{\Omega_m(1+z)^3 + \Omega_r(1+z)^4}$, $n_H = (1-Y_p)\rho_{b,0}(1+z)^3/m_H$, α_B from the case-B fit of Pequignot, Petitjean & Boisson (1991) times the RECFAST fudge factor 1.14, and $\beta_B = \alpha_B(2\pi m_e k_B T/h^2)^{3/2} e^{-E_I/(4k_B T)}$ with $E_I = 13.598$ eV.

Value

A named numeric vector containing:

`dxe_dlna` Derivative of x_e with respect to $\ln a$ (dimensionless).

`C` Peebles C-factor (dimensionless probability that an excited atom reaches the ground state before being re-ionized).

`lambda_alpha_escape` Effective Lyman-alpha escape rate in s^{-1}.

References

Peebles, P. J. E. (1968). Recombination of the Primeval Plasma. Astrophysical Journal, 153, 1.

Seager, S., Sasselov, D. D. & Scott, D. (1999). A New Calculation of the Recombination Epoch. Astrophysical Journal Letters, 523, L1 (RECFAST).

Pequignot, D., Petitjean, P. & Boisson, C. (1991). Total and effective radiative recombination coefficients. Astronomy and Astrophysics, 251, 680.

See Also

`Saha_Xe` for the equilibrium (Saha) ionization fraction.

Examples

```
# Physically consistent inputs at z = 1100 (see Details)
z <- 1100
peebles_rhs(
  lna     = -log(1 + z),
  xe      = 0.5,
  H       = 5.14e-14,  # s^-1
  n_H     = 2.53e8,    # m^-3
  alpha_B = 7.62e-19,  # m^3 s^-1
  beta_B  = 590        # s^-1
)
```

`photon_energy_density_fn_T`

Photon Energy Density as a Function of Temperature

Description

Calculates the energy density of blackbody photons at temperature T, using

$$\rho_\gamma = \frac{\pi^2}{15} T^4$$

in natural units ($\hbar = c = k_B = 1$).

Usage

```
photon_energy_density_fn_T(temp, unit = "eV")
```

Arguments

`temp`	Numeric vector of non-negative temperatures, in eV or Kelvin.
`unit`	Character. The unit of the temperature input. Either `"eV"` (default) or `"K"`.

Details

A temperature given in Kelvin is first converted to eV with the exact Boltzmann constant $k_B = 8.617333 \times 10^{-5}$ eV/K. To obtain an energy density per volume, divide the result by $(\hbar c)^3$, with $\hbar c = 1.973270 \times 10^{-5}$ eV cm: for the CMB today (2.7255 K) this gives about 0.2606 eV cm^{-3}.

Value

Numeric vector, the same length as `temp`. The photon energy density in natural units, eV^4 (with T in eV).

See Also

`photon_energy_density_fn_z`, `photon_number_density_fn_T`

Examples

```
photon_energy_density_fn_T(1, "eV")
photon_energy_density_fn_T(300, "K")
# CMB today in eV cm^-3
photon_energy_density_fn_T(2.7255, "K") / 1.973270e-5^3
```

---

photon_energy_density_fn_z

Photon Energy Density as a Function of Redshift

Description

Calculates the energy density of the cosmic microwave background photons at redshift z, with temperature $T = T_0(1 + z)$ and $T_0 = 2.7255$ K (Fixsen 2009), using $\rho_\gamma = \pi^2 T^4/15$.

Usage

```
photon_energy_density_fn_z(z)
```

Arguments

z	Numeric vector of redshifts, $z > -1$.

Value

Numeric vector, the same length as z. The photon energy density in natural units, eV^4; see photon_energy_density_fn_T for the conversion to eV cm^{-3}.

References

Fixsen, D. J. (2009). The Temperature of the Cosmic Microwave Background. Astrophysical Journal, 707, 916.

See Also

photon_energy_density_fn_T, photon_number_density_fn_z

Examples

```
photon_energy_density_fn_z(1300)
photon_energy_density_fn_z(c(0, 1100))
```

`photon_number_density_fn_T` *Photon Number Density as a Function of Temperature*

Description

Calculates the number density of blackbody photons at temperature T, using

$$n_\gamma = \frac{2\zeta(3)}{\pi^2} T^3$$

in natural units ($\hbar = c = k_B = 1$).

Usage

```
photon_number_density_fn_T(temp, unit = "eV")
```

Arguments

`temp`	Numeric vector of non-negative temperatures, in eV or Kelvin.
`unit`	Character. The unit of the temperature input. Either `"eV"` (default) or `"K"`.

Details

A temperature given in Kelvin is first converted to eV with the exact Boltzmann constant $k_B = 8.617333 \times 10^{-5}$ eV/K. To obtain a number per volume, divide the result by $(\hbar c)^3$, with $\hbar c = 1.973270 \times 10^{-5}$ eV cm: for the CMB today (2.7255 K) this gives about 410.7 photons cm^{-3}.

Value

Numeric vector, the same length as `temp`. The photon number density in natural units, eV^3 (with T in eV).

See Also

`photon_number_density_fn_z`, `photon_energy_density_fn_T`

Examples

```
photon_number_density_fn_T(1, "eV")
photon_number_density_fn_T(300, "K")
# CMB today in photons per cm^3
photon_number_density_fn_T(2.7255, "K") / 1.973270e-5^3
```

photon_number_density_fn_z *Photon Number Density as a Function of Redshift*

Description

Calculates the number density of the cosmic microwave background photons at redshift z, with temperature $T = T_0(1 + z)$ and $T_0 = 2.7255$ K (Fixsen 2009), using $n_\gamma = 2\zeta(3)T^3/\pi^2$.

Usage

```
photon_number_density_fn_z(z)
```

Arguments

z	Numeric vector of redshifts, $z > -1$.

Value

Numeric vector, the same length as z. The photon number density in natural units, eV^3; see photon_number_density_fn_T for the conversion to cm^{-3}.

References

Fixsen, D. J. (2009). The Temperature of the Cosmic Microwave Background. Astrophysical Journal, 707, 916.

See Also

photon_number_density_fn_T, photon_energy_density_fn_z

Examples

```
photon_number_density_fn_z(1300)
photon_number_density_fn_z(c(0, 1100))
```

radec_to_altaz *Convert Right Ascension and Declination to Altitude and Azimuth*

Description

Converts equatorial coordinates (right ascension and declination) to horizontal coordinates (altitude and azimuth) for an observer at a given place on Earth and a given instant.

Usage

```
radec_to_altaz(ra, dec, lat, lon, datetime)
```

Arguments

`ra`	Numeric vector. Apparent right ascension in hours, in $[0, 24)$.
`dec`	Numeric vector. Apparent declination in degrees, in $[-90, 90]$.
`lat`	Numeric vector. Geographic latitude of the observer in degrees, in $[-90, 90]$ (north positive).
`lon`	Numeric vector. Geographic longitude of the observer in degrees, positive east of Greenwich and negative west of it. Any finite value is accepted (280 is the same as -80).
`datetime`	Instant(s) of the observation: anything accepted by `julian_day`, i.e. a `POSIXct`, `POSIXlt`, `Date` or character vector (character strings are read as UTC).

Details

The Greenwich mean sidereal time, in hours, is

$$GMST = 18.697374558 + 24.06570982441908(JD - 2451545.0),$$

which is equation 12.4 of Meeus (1998) without its very small quadratic and cubic terms, with JD from `julian_day`. For longitude λ and right ascension α the local sidereal time is $LST = GMST + \lambda/15$ and the hour angle is $H = 15(LST - \alpha)$ degrees. With declination δ and latitude ϕ, the altitude h and azimuth A are

$$\sin h = \sin\delta \sin\phi + \cos\delta \cos\phi \cos H,$$

$$A = \operatorname{atan2}(-\cos\delta \sin H, \sin\delta \cos\phi - \cos\delta \sin\phi \cos H),$$

where the sine of the altitude is clamped to $[-1, 1]$ to absorb rounding errors and the azimuth is reduced to $[0, 360)$.

Conventions and limitations:

- Longitude is positive east of Greenwich and negative west of it.
- Azimuth is measured from North through East: North is 0, East 90, South 180 and West 270 degrees. Meeus (1998) measures azimuth from South, so add 180 degrees to his values to compare.
- `ra` and `dec` must be apparent coordinates, referred to the true equator and equinox of the date. No precession, nutation or aberration is applied. Catalog (J2000.0) positions are fine for rough work, but precession alone shifts them by roughly 0.01 degrees for every year away from 2000.
- The altitude is geometric: atmospheric refraction, which raises objects near the horizon by about 0.5 degrees, is not applied.
- The sidereal time is mean, not apparent (the equation of the equinoxes, at most about 1.2 s of time, is ignored), and `datetime` is taken as Universal Time (UT1 - UTC is ignored).

For Meeus (1998) Example 13.b (Venus seen from the US Naval Observatory in Washington on 1987 April 10 at 19:21:00 UT) the function returns an altitude of 15.1243 and an azimuth of 248.0343 degrees, within 0.0007 degrees of the published 15.1249 and 248.0337 (68.0337 measured from South). The difference comes mainly from the mean sidereal time: given the apparent

sidereal time and the longitude (5h08m15.7s) that Meeus uses, the results agree with his to better than 0.0001 degrees.

All arguments are vectorized: each must have length 1 or a common length n, and the result has n rows. A missing value in any argument gives `NA` in the corresponding row.

Value

A data frame with one row per observation and the columns

`altitude` Altitude above the horizon in degrees, in $[-90, 90]$.

`azimuth` Azimuth in degrees, in $[0, 360)$, measured from North through East.

References

Meeus, J. (1998). *Astronomical Algorithms*, 2nd edition. Willmann-Bell, Richmond, Virginia. Chapters 12 and 13.

See Also

`julian_day`; `dms_to_deg` for a sexagesimal dec, and `hms_to_deg` for a sexagesimal right ascension. `hms_to_deg()` returns degrees, so pass `hms_to_deg(h, m, s) / 15` as `ra`.

Examples

```
# The globular cluster M13 (RA 16h41.7m, Dec +36d28m) seen from
# Birmingham, UK
radec_to_altaz(ra = 16.695, dec = 36.466667, lat = 52.5, lon = -1.9166667,
               datetime = as.POSIXct("1998-08-10 23:10:00", tz = "UTC"))

# Meeus (1998) Example 13.b: Venus seen from Washington, D.C.
radec_to_altaz(ra = 23 + 9 / 60 + 16.641 / 3600,
               dec = -(6 + 43 / 60 + 11.61 / 3600),
               lat = 38 + 55 / 60 + 17 / 3600,
               lon = -(77 + 3 / 60 + 56 / 3600),
               datetime = "1987-04-10 19:21:00")

# Track of Rigel seen from Rochester, NY, every three hours
times <- seq(as.POSIXct("2024-10-02 00:00:00", tz = "UTC"),
             by = "3 hours", length.out = 9)
track <- radec_to_altaz(ra = 78.634467 / 15, dec = -8.20164,
                        lat = 43.1566, lon = -77.6088, datetime = times)
data.frame(time = times, track)
```

rad_to_deg *Convert Radians to Degrees*

Description

Converts angles from radians to degrees. Supports vectorized input.

Usage

```
rad_to_deg(rad)
```

Arguments

rad	Numeric. The angle(s) in radians to convert to degrees. NA values give NA.

Value

Numeric. The corresponding angle(s) in degrees, $rad \times 180/\pi$.

See Also

deg_to_rad for the reverse conversion; deg_to_dms and deg_to_hms for sexagesimal output.

Examples

```
rad_to_deg(pi)
rad_to_deg(c(0, pi / 2, pi))
```

redshift_to_scale_factor *Convert Between Redshift and Scale Factor*

Description

redshift_to_scale_factor() returns the scale factor

$$a = \frac{1}{1+z}$$

($a = 1$ today) and scale_factor_to_redshift() the inverse, $z = 1/a - 1$.

Usage

```
redshift_to_scale_factor(z)

scale_factor_to_redshift(a)
```

Arguments

z	Numeric vector of redshifts, each greater than -1. NA values give NA.
a	Numeric vector of positive scale factors. NA values give NA.

Value

Numeric vector of the same length as the input (no names): dimensionless scale factors from `redshift_to_scale_factor()`, dimensionless redshifts from `scale_factor_to_redshift()`.

See Also

`cosmic_time`, `hubble_ratio`

Examples

```
redshift_to_scale_factor(c(0, 1, 3, 1100))
scale_factor_to_redshift(c(1, 0.5, 0.25))
```

Saha_Xe *Hydrogen Ionization Fraction from the Saha Equation*

Description

`Saha_Xe()` calculates the equilibrium ionization fraction X_e of hydrogen at redshift z from the Saha equation. `soln_saha()` returns $X_e - 0.5$, whose root (for example via `uniroot`) is the redshift at which hydrogen is half ionized.

Usage

```
Saha_Xe(z)

soln_saha(z)
```

Arguments

z	Numeric vector of redshifts, $z > -1$.

Details

In natural units the Saha equation for hydrogen reads

$$\frac{X_e^2}{1 - X_e} = \frac{1}{n_H} \left(\frac{m_e T}{2\pi} \right)^{3/2} e^{-E_I / T}$$

with $T = T_0(1 + z)$, $T_0 = 2.7255$ K, $E_I = 13.598434$ eV (NIST), $m_e = 0.51099895$ MeV and the hydrogen density $n_H = 0.76\, \eta\, n_\gamma(T)$, where the baryon-to-photon ratio is $\eta = 6 \times 10^{-10}$.

Writing the right-hand side as $1/f$, the physical root of $fX_e^2 + X_e - 1 = 0$ is evaluated in the cancellation-free form

$$X_e = \frac{2}{1 + \sqrt{1 + 4f}}$$

so X_e stays in $[0, 1]$: it tends to 1 at high redshift and is exactly 0 once f overflows (below $z \approx 80$).

Saha equilibrium describes the start of recombination but predicts it to finish too early; see `peebles_rhs` for the out-of-equilibrium Peebles equation.

Value

For `Saha_Xe()`, a numeric vector the same length as `z`: the dimensionless ionization fraction X_e, between 0 and 1. For `soln_saha()`, the numeric vector $X_e - 0.5$.

See Also

`peebles_rhs`, `photon_number_density_fn_z`

Examples

```
Saha_Xe(1300)
Saha_Xe(c(1100, 1300, 1500))
soln_saha(1350)
# Redshift at which hydrogen is half ionized in Saha equilibrium
uniroot(soln_saha, c(1000, 2000))$root
```

---

solve_relic_abundance *Solve the Boltzmann Relic-Abundance (Pebble) Equation*

Description

Numerically integrates the thermal relic Boltzmann equation (`boltzmann_pebble_rhs`) from $x_{\rm ini}$ to $x_{\rm fin}$ using **deSolve**, starting in equilibrium, $Y(x_{\rm ini}) = Y_{\rm eq}(x_{\rm ini})$. Returns the comoving yield $Y(x)$ and the final relic density parameter Ωh^2.

Usage

```
solve_relic_abundance(
  m_GeV,
  sigmav_GeV2,
  g_dof = 2,
  x_ini = 5,
  x_fin = 1000,
  n_steps = 2000
)
```

Arguments

`m_GeV`	Numeric. Particle mass in GeV (a single positive number).
`sigmav_GeV2`	Numeric. $\langle\sigma v\rangle$ in GeV^{-2} (a single positive number).
`g_dof`	Integer. Internal DOF. Default 2.
`x_ini`	Numeric. Initial $x = m/T$, where the yield starts at its equilibrium value; must be positive. Default 5.
`x_fin`	Numeric. Final $x = m/T$; must be greater than `x_ini`. Default 1000.
`n_steps`	Integer. Number of output grid points, at least 2. Default 2000.

Details

The relic abundance today is:

$$\Omega_\chi h^2 \approx 2.755 \times 10^8 \, \frac{m}{\mathrm{GeV}} \, Y_\infty$$

where Y_∞ is the yield at x_{fin}.

Requires the **deSolve** package. The equation is solved for $W = \log Y$ with the `"lsoda"` method (`rtol = 1e-8`, `atol = 1e-10`), using `g_star_eff` for g_* and g_{*S}. An error is raised if the integration stops before `x_fin`.

Value

A list with:

`x` Numeric vector of x values.

`Y` Numeric vector of yield $Y(x)$.

`Y_eq` Numeric vector of equilibrium yield $Y^{\mathrm{eq}}(x)$.

`Omega_h2` Numeric. Relic density $\Omega_\chi h^2$.

References

Kolb & Turner (1990), Chapter 5.

See Also

`boltzmann_pebble_rhs`, `freeze_out_xf`, `equilibrium_yield`

Examples

```
sigmav <- 2.2e-9  # GeV^-2
result <- solve_relic_abundance(m_GeV = 100, sigmav_GeV2 = sigmav)
cat("Omega h^2 =", result$Omega_h2, "\n")
```

Index

∗ **datasets**
constants_df, 6

age_of_universe, *3*, *13*
age_of_universe *(*cosmic_time*)*, 8
angular_diameter_distance, *3*
angular_diameter_distance *(*cosmological_distances*)*, 9
astronomR *(*astronomR-package*)*, 3
astronomR-package, 3

boltzmann_pebble_rhs, *4*, 4, *21*, *23*, *26, 27*, *42, 43*

comoving_distance, *3*, *9*, *11*, *13*, *28*
comoving_distance *(*cosmological_distances*)*, 9
constant_value, *3*, *7*, 7
constants_df, *3*, 6, *7, 8*
cosmic_time, *3*, 8, *11*, *13*, *28*, *41*
cosmological_distances, 9
cosmology_model, *3*, *8–11*, 11, *28*

deg_to_dms, *3*, 13, *15–17*, *24*, *40*
deg_to_hms, *3*, *14*, 14, *16*, *24*, *27*, *40*
deg_to_rad, *3*, *14, 15*, 15, *17*, *27*, *40*
dms_to_deg, *3*, *14*, 16, *27*, *39*
drake_equation, *4*, 17

entropy_density, *4*, 19, *21*, *26*
equilibrium_number_density, *4*, 20, *21*
equilibrium_yield, *4, 5*, *19, 20*, 21, *26*, *43*

freeze_out_xf, *4, 5*, 22, *43*

g_star_eff, *4, 5*, *19*, *21*, 24, *27*, *43*
get_gaia_data, *3*, 23

hms_to_deg, *3*, *15*, *17*, 26, *39*
hubble_distance, *3*, *28*, *30*
hubble_distance *(*cosmological_distances*)*, 9
hubble_radiation, *4*, *22*, *26*, 27, *33*
hubble_ratio, *3*, *9–11*, *13*, 28, *41*

integrate, *9, 10*

julian_day, *3*, 29, *38, 39*

km_to_Mpc, *3*, *11*, 30, *31*

luminosity_distance, *3*, *13*
luminosity_distance *(*cosmological_distances*)*, 9

Mpc_to_km, *3*, *9*, *30*, 31

peebles_rhs, *4*, 32, *42*
photon_energy_density_fn_T, *4*, 34, *35, 36*
photon_energy_density_fn_z, *4*, *34*, 35, *37*
photon_number_density_fn_T, *4*, *34*, 36, *37*
photon_number_density_fn_z, *4*, *35, 36*, 37, *42*
print.astronomR_cosmology *(*cosmology_model*)*, 11

rad_to_deg, *3*, *16*, 40
radec_to_altaz, *3*, *30*, 37
radius_of_curvature, *3*
radius_of_curvature *(*cosmological_distances*)*, 9
redshift_to_scale_factor, *3*, *9*, 40

Saha_Xe, *4*, *33*, 41
scale_factor_to_redshift, *3*
scale_factor_to_redshift *(*redshift_to_scale_factor*)*, 40
soln_saha, *4*
soln_saha *(*Saha_Xe*)*, 41
solve_relic_abundance, *4, 5*, *23*, 42

transverse_comoving_distance, *3*

transverse_comoving_distance
(cosmological_distances), 9

uniroot, *41*