Package 'astronomR'

July 28, 2026

Type Package

Title Cosmic Insights: Statistical Frameworks for Astronomers

Version 0.3.0

Description A comprehensive toolkit for astronomical and cosmological computations. Provides functions for angular coordinate conversions (degrees, hours-minutes-seconds, degrees-minutes-seconds, and radians), access to fundamental physical constants, queries to the Gaia Archive TAP (Table Access Protocol) service, cosmological distance calculations, 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 dplyr, httr, jsonlite, pracma, readr, stringr, tibble, tidyr

Depends R (>= 4.1.0)

Suggests deSolve, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

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 2026-07-28 09:10:09 UTC

Contents

`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}. 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 from g_star_eff.
g_star_S_val	Numeric. g_{*S} at $T = m/x$. If NULL, set equal to g_star_val.

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).

Value

Numeric. dY/dx.

References

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

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)
```

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

Description

A dataset containing commonly used physical constants in both SI units and natural units. The dataset includes the constant's name, symbol, value in SI units, SI unit, value in natural units, and natural unit representation.

Usage

```
constants_df
```

Format

A tibble with 19 rows and 6 columns:

name Character. Name of the physical constant.

symbol Character. Symbol representing the constant.

value_SI Numeric. The value of the constant in SI units.

unit_SI Character. The SI unit for the constant.

value_Natural Numeric. The value of the constant in natural units.

unit_Natural Character. The unit of the constant in natural units.

Value

A `tibble` (data frame) with 19 rows and 6 columns:

name Character. Full name of the physical constant.

symbol Character. Symbol representing the constant.

value_SI Numeric. Value of the constant in SI units.

unit_SI Character. The SI unit string for the constant.

value_Natural Numeric. Value of the constant in natural units.

unit_Natural Character. The natural unit string for the constant.

Examples

```
constants_df
```

constant_value *Retrieve the Value of a Physical Constant*

Description

Retrieves the value and unit of a specified physical constant from the `constants_df` dataset. You can search by the constant's name and choose whether to retrieve the value in SI units or natural units.

Usage

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

Arguments

`constant_name`	Character. The name (or part of the name) of the constant to search for. Case-insensitive and partial matches are allowed.
`unit`	Character. The unit system to retrieve the constant in. Options are: • `"SI"`: Retrieves the value in SI units (default). • `"Natural"`: Retrieves the value in natural units.

Value

A list containing:

- `name`: The full name of the constant.
- `value`: The numeric value of the constant.
- `unit`: The unit string for the selected unit system.

Examples

```
constant_value("speed of light", unit = "SI")
constant_value("planck", unit = "SI")
constant_value("electron mass", unit = "Natural")
```

deg2rad *Convert Degrees to Radians*

Description

Converts angles from degrees to radians. Supports vectorized input.

Usage

```
deg2rad(deg)
```

Arguments

deg	Numeric. The angle(s) in degrees to convert to radians.

Value

Numeric. The corresponding angle(s) in radians.

Examples

```
deg2rad(180)
deg2rad(c(0, 90, 180))
```

---

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 = "cat", digit = 5)
```

Arguments

deg	Numeric vector of angles in decimal degrees. All values must be between -90 and 90.
type	Character string specifying the output format. Options are: • "cat" (default): Prints the angle as a formatted string in DMS notation. • "mat": Returns a matrix with columns for sign, degrees, minutes, and seconds.
digit	Integer specifying the number of digits to round the seconds to. Default is 5.

Value

When type = "cat", prints the DMS string and returns NULL invisibly. When type = "mat", returns a character matrix with columns SIGN, DEG, MIN, and SEC.

Examples

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

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 = "cat", digit = 5)
```

Arguments

deg	Numeric vector of angles in decimal degrees. Values can range from 0 to 360.
type	Character string specifying the output format. Options are: • "cat" (default): Prints a formatted string (e.g., 3H0M0S). • "mat": Returns a matrix with columns for hours, minutes, and seconds.
digit	Integer specifying the number of decimal places to round seconds to. Default is 5.

Value

When type = "cat", prints the HMS string and returns NULL invisibly. When type = "mat", returns a numeric matrix with columns HRS, MIN, and SEC.

Examples

```
deg_to_hms(deg = 45)
deg_to_hms(deg = 45, type = "mat")
deg_to_hms(deg = 177.74208, digit = 3)
```

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, s, digit = 5)
```

Arguments

d	Numeric or character. The degrees component. If a single character string is provided (e.g., "12\u00B034'56\""), the degrees, minutes, and seconds are parsed automatically, and m and s should be omitted.
m	Numeric. The minutes component. Must be less than 60. Optional when d is a string.
s	Numeric. The seconds component. Must be less than 60. Optional when d is a string.
digit	Integer. Number of decimal places to round the result to. Default is 5.

Value

Numeric. The angle in decimal degrees.

Examples

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

---

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:

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

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

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.

`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
```

`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 from `g_star_eff`.

Value

Numeric. Entropy density s in GeV^3.

References

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

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).

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:

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

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 from g_star_eff.

Value

Numeric. Equilibrium yield Y^{eq}.

References

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

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 $\Gamma \sim H$:

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

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

Typical WIMP values lie in the range $x_f \approx 20$–30.

Value

Numeric. Freeze-out parameter x_f.

References

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

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")
```

---

get_gaia_data *Query Gaia Archive Data*

Description

Queries the Gaia Archive TAP service to retrieve stellar data based on specified variables and filter conditions. Uses the Gaia Early Data Release 3 (EDR3) catalog.

Usage

```
get_gaia_data(vars, condition)
```

Arguments

`vars`	A character string specifying the variables (columns) to retrieve, separated by commas (e.g., `"source_id, ra, dec"`).
`condition`	A character string specifying the SQL WHERE clause used to filter rows (e.g., `"parallax > 10"`).

Details

This function sends a synchronous ADQL query to the Gaia Archive TAP service at https://gea.esac.esa.int/tap-server/tap/sync. An internet connection is required.

Value

A data frame containing the requested columns for all rows matching `condition`.

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 $g_*(T)$, the effective number of relativistic degrees of freedom as a function of temperature, based on Standard Model particle content.

Usage

```
g_star_eff(T_GeV)
```

Arguments

T_GeV	Numeric. Temperature in GeV.

Value

Numeric. Effective relativistic degrees of freedom g_*.

References

Husdal (2016), arXiv:1609.04979.

Examples

```
g_star_eff(100)   # ~ 86.25 (above QCD transition)
g_star_eff(1e-3)  # ~ 10.75 (neutrino era)
g_star_eff(1e-4)  # ~ 3.91  (after e+e- annihilation)
```

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, s, digit = 5)
```

Arguments

h	Numeric or character. The hours component. If a single character string is provided (e.g., "03h15m30s"), the hours, minutes, and seconds are parsed automatically, and m and s should be omitted.
m	Numeric. The minutes component. Optional when h is a string.
s	Numeric. The seconds component. Optional when h is a string.
digit	Integer. Number of decimal places to round the result to. Default is 5.

Value

Numeric. The angle in decimal degrees.

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")
```

---

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).

Value

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

References

Kolb & Turner, *The Early Universe* (1990), Eq. (3.46).

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)
```

`km_to_Mpc` *Convert Kilometers to Megaparsecs*

Description

This function converts a distance value from kilometers (km) to megaparsecs (Mpc). The conversion factor is based on 1 parsec being equivalent to 3.262 light-years, and 1 light-year being approximately 9.461×10^{12} kilometers.

Usage

```
km_to_Mpc(km)
```

Arguments

`km`	A numeric value representing the distance in kilometers.

Value

A numeric value representing the equivalent distance in megaparsecs (Mpc).

Examples

```
km_to_Mpc(3.086e19) # Converts 3.086e19 km (approx. 1 Mpc) to megaparsecs
```

`Mpc_to_km` *Convert Megaparsecs to Kilometers*

Description

This function converts a distance value from megaparsecs (Mpc) to kilometers (km). The conversion factor is based on 1 parsec being equivalent to 3.262 light-years, and 1 light-year being approximately 9.461×10^{12} kilometers.

Usage

```
Mpc_to_km(mpc)
```

Arguments

mpc	A numeric value representing the distance in megaparsecs.

Value

A numeric value representing the equivalent distance in kilometers (km).

Examples

```
Mpc_to_km(1) # Converts 1 Mpc to kilometers
```

---

peebles_rhs *Peebles Equation for Hydrogen Recombination*

Description

Computes the derivative of the free-electron fraction during cosmological hydrogen recombination using the Peebles C-factor.

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
)
```

Arguments

lna	Natural logarithm of the scale factor, `log(a)`.
xe	Free-electron fraction per hydrogen nucleus, $0 \leq x_e \leq 1$.
H	Hubble expansion rate in s^{-1}.
n_H	Total hydrogen-nuclei number density in m^{-3}.
alpha_B	Case-B recombination coefficient in $\mathrm{m}^3\ \mathrm{s}^{-1}$.
beta_B	Photoionization coefficient from the $n = 2$ state in s^{-1}.
lambda_alpha	Lyman-alpha wavelength in metres. Default is 121.567 nm (`121.567e-9` m).
Lambda_2s1s	Hydrogen $2s \rightarrow 1s$ 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`.

Details

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

$$\frac{dx_e}{d\ln a} = \frac{C}{H}\left[\beta_B(1-x_e) - n_H \alpha_B x_e^2\right].$$

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.

All quantities are in SI units (metres, seconds). Note that H here is in s^{-1}, not GeV as in the rest of this file.

Value

A named numeric vector containing:

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

`C` Peebles C-factor (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.

Examples

```
# Evaluate at z ~ 1100 (recombination epoch)
# Approximate inputs at z = 1100:
H_rec      <- 3.3e-15   # s^-1  (Hubble rate at recombination)
n_H_rec    <- 400e6     # m^-3  (hydrogen number density)
alpha_B    <- 2.6e-19   # m^3 s^-1  (case-B recombination coeff)
beta_B     <- 4e-15     # s^-1  (photoionization rate)
peebles_rhs(
  lna          = log(1 / 1101),
  xe           = 0.5,
  H            = H_rec,
  n_H          = n_H_rec,
  alpha_B      = alpha_B,
  beta_B       = beta_B
)
```

photon_energy_density_fn_T

Photon Energy Density as a Function of Temperature

Description

Calculates the photon energy density for a given temperature T, using $\rho_\gamma = \frac{\pi^2}{15} T^4$.

Usage

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

Arguments

temp	Numeric. Temperature in either eV or Kelvin.
unit	Character. The unit of the temperature input. Either "eV" (default) or "K".

Value

Numeric. The photon energy density in natural units.

Examples

```
photon_energy_density_fn_T(1, "eV")
photon_energy_density_fn_T(300, "K")
```

photon_energy_density_fn_z

Photon Energy Density as a Function of Redshift

Description

Calculates the photon energy density at a given cosmological redshift z by first converting to temperature in eV.

Usage

```
photon_energy_density_fn_z(z)
```

Arguments

z	Numeric. The redshift value.

Value

Numeric. The photon energy density in natural units.

Examples

```
photon_energy_density_fn_z(1300)
```

`photon_number_density_fn_T`
Photon Number Density as a Function of Temperature

Description

Calculates the photon number density for a given temperature T, using $n_\gamma = \frac{2\zeta(3)}{\pi^2} T^3$.

Usage

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

Arguments

`temp`	Numeric. Temperature in either eV or Kelvin.
`unit`	Character. The unit of the temperature input. Either `"eV"` (default) or `"K"`.

Value

Numeric. The photon number density in natural units.

Examples

```
photon_number_density_fn_T(1, "eV")
photon_number_density_fn_T(300, "K")
```

`photon_number_density_fn_z`
Photon Number Density as a Function of Redshift

Description

Calculates the photon number density at a given cosmological redshift z by first converting to temperature in eV.

Usage

```
photon_number_density_fn_z(z)
```

Arguments

`z`	Numeric. The redshift value.

Value

Numeric. The photon number density in natural units.

Examples

```
photon_number_density_fn_z(1300)
```

---

rad2deg *Convert Radians to Degrees*

Description

Converts angles from radians to degrees. Supports vectorized input.

Usage

```
rad2deg(rad)
```

Arguments

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

Value

Numeric. The corresponding angle(s) in degrees.

Examples

```
rad2deg(pi)
rad2deg(c(0, pi / 2, pi))
```

Saha_Xe *Ionization Fraction Using the Saha Equation*

Description

Calculates the ionization fraction X_e as a function of redshift z using the Saha equation for hydrogen recombination.

Usage

```
Saha_Xe(z)
```

Arguments

z	Numeric. The redshift value.

Value

Numeric. The ionization fraction X_e (between 0 and 1).

Examples

```
Saha_Xe(1300)
Saha_Xe(1100)
```

`soln_saha` *Deviation of Ionization Fraction from 0.5*

Description

Returns $X_e(z) - 0.5$, useful for finding the redshift at which the ionization fraction equals 0.5 (e.g., via root-finding).

Usage

```
soln_saha(z)
```

Arguments

`z`	Numeric. The redshift value.

Value

Numeric. $X_e - 0.5$.

Examples

```
soln_saha(1350)
```

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

Description

Numerically integrates the thermal relic Boltzmann equation from $x_{\rm ini}$ to $x_{\rm fin}$ using **deSolve**. 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.
`sigmav_GeV2`	Numeric. $\langle \sigma v \rangle$ in GeV^{-2}.
`g_dof`	Integer. Internal DOF. Default 2.
`x_ini`	Numeric. Initial $x = m/T$. Default 1.
`x_fin`	Numeric. Final $x = m/T$. Default 1000.
`n_steps`	Integer. Number of output grid points. Default 2000.

Details

The relic abundance today is:

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

Requires the **deSolve** package.

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.

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