

Package: astronomR (via r-universe)

June 7, 2026

Type Package

Title Cosmic Insights: Statistical Frameworks for Astronomers

Version 0.2.0

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 and 'Saha' equation solutions, and 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 testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev libssl-dev libx11-dev

Repository <https://samrit2442.r-universe.dev>

Date/Publication 2026-06-07 17:18:37 UTC

RemoteUrl <https://github.com/samrit2442/astronomr>

RemoteRef HEAD

RemoteSha 11bc24f1adc59cc469f9529443219241ac387d7f

Contents

constant_value	2
constants_df	3
deg_to_dms	4
deg_to_hms	4
deg2rad	5
dms_to_deg	6
drake_equation	6
get_gaia_data	8
hms_to_deg	9
km_to_Mpc	10
Mpc_to_km	10
photon_energy_density_fn_T	11
photon_energy_density_fn_z	11
photon_number_density_fn_T	12
photon_number_density_fn_z	12
rad2deg	13
Saha_Xe	14
soln_saha	14
Index	15

constant_value	<i>Retrieve the Value of a Physical Constant</i>
----------------	--

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: <ul style="list-style-type: none"> "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")
```

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.

Examples

```
constants_df
```

deg_to_dms	<i>Convert Decimal Degrees to Degrees, Minutes, and Seconds (DMS) Format</i>
------------	--

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: <ul style="list-style-type: none"> "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	<i>Convert Decimal Degrees to Hours, Minutes, and Seconds (HMS) Format</i>
------------	--

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: <ul style="list-style-type: none"> "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)
```

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))
```

`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

<code>d</code>	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 <code>m</code> and <code>s</code> should be omitted.
<code>m</code>	Numeric. The minutes component. Must be less than 60. Optional when <code>d</code> is a string.
<code>s</code>	Numeric. The seconds component. Must be less than 60. Optional when <code>d</code> is a string.
<code>digit</code>	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
```

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)
```

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")
```

`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
```

```
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)

Index

[constant_value](#), 2
[constants_df](#), 3

[deg2rad](#), 5
[deg_to_dms](#), 4
[deg_to_hms](#), 4
[dms_to_deg](#), 6
[drake_equation](#), 6

[get_gaia_data](#), 8

[hms_to_deg](#), 9

[km_to_Mpc](#), 10

[Mpc_to_km](#), 10

[photon_energy_density_fn_T](#), 11
[photon_energy_density_fn_z](#), 11
[photon_number_density_fn_T](#), 12
[photon_number_density_fn_z](#), 12

[rad2deg](#), 13

[Saha_Xe](#), 14
[soln_saha](#), 14