Package 'REXoplanets'

September 11, 2025

Title Creates Interface with NASA 'Exoplanets Archive API'

Version 0.1.2

Description Provides a user-friendly interface to NASA 'Exoplanets Archive API' <https://exoplanetarchive.ipac.caltech.edu/>, enabling retrieval and analysis of exoplanetary and stellar data. Includes functions for querying, filtering, summarizing, and computing derived parameters from the 'Exoplanets' catalog.

License MIT + file LICENSE

Depends R (>= 4.1)

Imports dplyr, checkmate, purrr, httr2, readr, utils, ggplot2, jsonlite, logger

Suggests testthat (>= 3.0.0), roxygen2, lintr, devtools, spelling, shiny, shinyjs, bslib, shinytest2, bsicons, htmltools

URL https://JKolomanski.github.io/REXoplanets/

https://github.com/JKolomanski/REXoplanets

BugReports https://github.com/JKolomanski/REXoplanets/issues

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

LazyData true

Language en-US

NeedsCompilation no

Author Jakub Kołomański [cre, aut], Mateusz Kołomański [aut]

Maintainer Jakub Kołomański <xqubula@gmail.com>

Repository CRAN

Date/Publication 2025-09-11 09:10:02 UTC

Contents

`app` *Function responsible for initializing and running the REXoplanets shiny application.*

Description

Function responsible for initializing and running the REXoplanets shiny application.

Usage

```
app(..., run = TRUE)
```

Arguments

`...`	Additional arguments passed to `shiny::runApp()`.
`run`	If true, runs the application. If false, returns an app object.

Value

App object.

`calculate_esi` *Calculate the Earth Similarity Index (ESI)*

Description

Calculate the Earth Similarity Index (ESI)

Usage

```
calculate_esi(..., radius_w = 0.57, flux_w = 0.7)
```

Arguments

`...`	Any number of numeric parameters representing the planet's characteristics. Parameter names should have corresponding weight names ending in `_w` (e.g., `mass`, `mass_w`).
`radius_w`	Numeric. Weight for radius. Default is 0.57.
`flux_w`	Numeric. Weight for stellar flux. Default is 0.7.

Details

The function calculates the ESI for any parameter or number of parameters. By default, it uses the weights of 0.57 for radius and 0.7 for stellar flux.

ESI (Earth Similarity Index) is a characterization of how similar a planetary-mass object or natural satellite is to Earth. It is designed to be a scale from zero to one, with Earth having a value of 1.

Value

Numeric. Earth Similarity Index (ESI).

References

Schulze-Makuch, D., Méndez, A., Fairén, A. G., von Paris, P., Turse, C., Boyer, G., Davila, A. F., Resendes de Sousa António, M., Irwin, L. N., and Catling, D. (2011) A Two-Tiered Approach to Assess the Habitability of Exoplanets. Astrobiology 11(10): 1041-1052.

Examples

```
# ESI for radius and flux, returns 1 (Earth is perfectly similar to itself)
calculate_esi(radius = 1, flux = 1)
# Mars approximation using radius = 0.532 and flux = 0.43, returns approximately 0.754
calculate_esi(radius = 0.532, flux = 0.43)
# Custom 3-parameter ESI (e.g. radius, flux, temperature)
calculate_esi(radius = 1.1, flux = 1.2, temp = 288,
              radius_w = 0.5, flux_w = 0.3, temp_w = 0.2)
```

calculate_star_habitable_zone

Calculate star's habitable zone

Description

Calculates star's habitable zone inner and outer radius, based on star's luminosity.

Usage

```
calculate_star_habitable_zone(st_lum, log_lum = TRUE)
```

Arguments

st_lum	A numeric Stellar luminosity value (log10(L/Lsun) or linear).
log_lum	A logical value. If TRUE assumes st_lum is logarithmic. Defaults to TRUE.

Value

A numeric vector of 2 elements where first is habitable zone inner radius and second is the outer radius.

Examples

```
calculate_star_habitable_zone(0) # habitable zone for sun, with logarithmic units
calculate_star_habitable_zone(1, log_lum = FALSE) # habitable zone for sun, with linear units
```

calculate_stellar_flux

Calculate stellar flux value

Description

Calculate stellar flux value

Usage

```
calculate_stellar_flux(st_lum, pl_orbsmax, log_lum = TRUE, unit = "relative")
```

Arguments

st_lum	Numeric. Stellar luminosity (log10(L/Lsun) or linear).
pl_orbsmax	Numeric. Orbital distance in AU.
log_lum	Logical. If TRUE, assumes st_lum is in log10(L / Lsun). Defaults to TRUE.
unit	Character. Either "relative" (default) or "wm2" to convert to W/m².

Details

This function calculates the stellar flux based on provided values. It assumes luminosity is either logarithmic (log10 of L/Lsun) or linear, and optionally converts flux to absolute units (W/m²) if requested.

Stellar flux is the amount of energy from a star that reaches a given area per unit time.

Value

Numeric. Stellar flux (relative or in W/m²).

Examples

```
# Solar-type star, Earth-like orbit
calculate_stellar_flux(st_lum = 0, pl_orbsmax = 1)
# Linear luminosity input (not log), 5x Sun at 2 AU
calculate_stellar_flux(st_lum = 5, pl_orbsmax = 2, log_lum = FALSE)
# Output in absolute units (W/m²), Earth-like conditions
calculate_stellar_flux(st_lum = 0, pl_orbsmax = 1, unit = "wm2")
```

---

classify_planet_type *Classify Planet Type*

Description

This function returns a compact four-character code representing the classification of an exoplanet based on its mass, equilibrium temperature, orbital eccentricity, and density.

The classification code is composed of four parts:

1. **Mass class:**
   - `M`: Mercury-like planets (< 0.22 Earth masses)
   - `E`: Earth-like planets (0.22–2.2 Earth masses)
   - `S`: Super-Earths (2.2–22 Earth masses)
   - `N`: Neptune-like planets (22–127 Earth masses)
   - `J`: Jupiter-like giants (127–4450 Earth masses)
   - `D`: Degenerate-matter/brown dwarf-like objects (>= 4450 Earth masses)
2. **Temperature class:**
   - `F`: Frozen (T < 250 K)
   - `W`: Temperate/water zone (250–450 K)
   - `G`: Gaseous (450–1000 K)
   - `R`: Roasters (>= 1000 K)
3. **Eccentricity:**
   - First decimal digit of orbital eccentricity. For example, 0.26 → `3` → appended as `3`.
4. **Density-based surface/composition class:**

- g: Gas-dominated (< 0.25 g/cm³)
- w: Water/ice-rich (0.25–2 g/cm³)
- t: Terrestrial/rocky (2–6 g/cm³)
- i: Iron-rich (6–13 g/cm³)
- s: Super-dense (>= 13 g/cm³)

Usage

```
classify_planet_type(pl_bmasse, pl_eqt, pl_orbeccen, pl_dens)
```

Arguments

pl_bmasse	Numeric. Planetary mass in Earth masses. Must be > 0.
pl_eqt	Numeric. Planetary equilibrium temperature in Kelvin. Must be > 0.
pl_orbeccen	Numeric. Orbital eccentricity. Must be > 0.
pl_dens	Numeric. Planetary density in g/cm³. Must be > 0.

Value

A character string containing a 4-character planet classification code.

Examples

```
classify_planet_type(1.0, 288, 0.0167, 5.5)   # Earth-like: "EW0t"
classify_planet_type(318, 1300, 0.05, 1.3)    # Hot Jupiter: "JR3w"
classify_planet_type(0.1, 180, 0.2, 0.1)      # Cold, light, low-density: "MF2g"
```

---

classify_star_spectral_type

Classify spectral type of a star

Description

The function takes in effective stellar temperature (in K), and return letter for the star's spectral type based on Morgan–Keenan (MK) system.

Usage

```
classify_star_spectral_type(st_teff)
```

Arguments

st_teff	Numeric. effective stellar temperature (in K)

Value

Character. spectral type classification with a letter:

- M: 2,500 - 3,500
- K: 3,500 - 5,000
- G: 5,000 - 6,000
- F: 6,000 - 7,500
- A: 7,500 - 10,000
- B: 10,000 - 28,000
- O: 28,000 - 50,000

Examples

```
classify_star_spectral_type(5778)
```

---

closest_50_exoplanets *closest_50_exoplanets*

Description

A sample dataset of 50 exoplanets closest to earth, taken from table `Planetary Systems Composite Parameters`, including their names, discovery methods, and other relevant information.

Usage

```
closest_50_exoplanets
```

Format

A data frame with 683 columns and 50 rows.

Source

https://exoplanetarchive.ipac.caltech.edu/

exoplanets_col_labels *exoplanets_col_labels A collection of Exoplanets archive columns names and their respective labels / comments. Currently only supports* ps *and* pscomppars *tables*

Description

exoplanets_col_labels A collection of Exoplanets archive columns names and their respective labels / comments. Currently only supports ps and pscomppars tables

Usage

```
exoplanets_col_labels
```

Format

named vectors nested in a list.

Source

https://exoplanetarchive.ipac.caltech.edu/docs/API_PS_columns.html#columns

Examples

```
exoplanets_col_labels[["ps"]][["Planet Name"]]
```

fetch_table *Fetch Exoplanets table*

Description

Fetch Exoplanets table

Usage

```
fetch_table(
  table,
  query_string = NULL,
  pretty_colnames = FALSE,
  format = "csv"
)
```

Arguments

table	A string specifying the table to query. Must be one of: • ps - Planetary Systems table • pscomppars – Planetary Systems Composite Parameters table • stellarhosts – Stellar Hosts table • keplernames – Kepler Confirmed Names
query_string	Optional ADQL WHERE clause as a string, e.g., pl_bmasse > 1 AND st_teff < 6000.
pretty_colnames	Optional bool value. If TRUE replaces database column names with their labels / descriptions. Defaults to FALSE. Currently only tables ps and pscomppars are supported.
format	Optional char value specifying output format. Can be either "csv" for data frame, or "json" for a named list.

Details

Fetches data from an exoplanets TAP (Table Access Protocol) service and returns it as a data frame or a named list. You can optionally specify WHERE ADQL clause to filter rows based on conditions.

Value

A data frame or named list containing fetched data.

Examples

```
# All entries from Stellar Hosts table
fetch_table("stellarhosts")
# Entries from Planetary Systems table where planetary mass > 3 times the earth mass
fetch_table("ps", query_string = "pl_bmasse > 3")
# Planets orbiting Teegarden's Star with radius > 1 Earth radius
fetch_table("pscomppars", query_string = "hostname = 'Teegarden''s Star' and pl_rade > 1")
```

---

module_planet_details *Module Displaying Planet Details*

Description

Module Displaying Planet Details

Usage

```
planet_details_ui(id)

planet_details_server(id, planet_info)
```

Arguments

id	A unique identifier for the module.
planet_info	A reactive expression returning a data frame with information about the planet.

Details

This module provides a dynamically refreshing table component for displaying basic information about a planet.

Value

A Shiny UI object.

Functions

- planet_details_ui(): UI function for the planet details module.
- planet_details_server(): server function for the planet details module.

---

module_search *Search Module*

Description

Search Module

Usage

```
search_ui(id, label = "Search...", multiple = FALSE, random = FALSE)

search_server(id, choices, start_random = FALSE)
```

Arguments

id	A unique identifier for the module.
label	The label text for the select input.
multiple	Whether to allow multiple selections (default: FALSE).
random	Whether to show a randomize button (default: FALSE).
choices	A reactive expression that returns a vector of choices for the select input.
start_random	Whether to select a random value when choices are updated.

Details

This module provides a reusable search component with a select input and optional randomize button. It can be used to create searchable drop-downs with support for multiple selection and random choice selection. The module handles updating choices dynamically and provides a reactive value for the selected item(s).

Value

A Shiny UI object.

A reactive expression containing the selected value(s).

Functions

- `search_ui()`: UI function for the search module.
- `search_server()`: Server function for the search module.

`module_star_systems` *Star Systems Module*

Description

Star Systems Module

Usage

```
star_systems_ui(id)

star_systems_server(id, data)
```

Arguments

`id`	A unique identifier for the module.
`data`	A reactive expression that returns a data frame containing star system data.

Details

This module provides functionality for exploring star systems and their planets. Provides a handy UI for selecting a star system. The star is visualized alongside its planets and their orbits. The display provides basic information about the system itself, as well as particular planets. A legend is available for the plot.

Value

A Shiny UI object.

A Shiny server module.

Functions

- `star_systems_ui()`: UI function for the module.
- `star_systems_server()`: Server function for the module.

module_system_info *Module Displaying Star System Info with Value Boxes (Two Rows)*

Description

Module Displaying Star System Info with Value Boxes (Two Rows)

Usage

```
system_info_ui(id)

system_info_server(id, system_info)
```

Arguments

id	A unique identifier for the module.
system_info	A reactive expression returning a data frame with information about the system.

Value

A Shiny UI object.

Functions

- system_info_ui(): UI function for the system info module using value boxes.
- system_info_server(): Server function for the system info module using value boxes.

module_system_plot_settings *System Plot Settings Module*

Description

System Plot Settings Module

Usage

```
system_plot_settings_ui(id)

system_plot_settings_server(id)
```

Arguments

id	A unique identifier for the module.

Details

This module provides a series of controls to customize the visible elements of a star system map

Value

A Shiny UI object.

A reactive list object containing bool value whether to show the habitable zone and plot legend.

Functions

- `system_plot_settings_ui()`: UI function for the plot settings module.
- `system_plot_settings_server()`: function for the search module.

`module_visualize_star_system`

Module Visualizing Star Systems

Description

Module Visualizing Star Systems

Usage

```
visualize_star_system_ui(id)

visualize_star_system_server(id, plot_data, show_hz, show_legend)
```

Arguments

`id`	A unique identifier for the module.
`plot_data`	A reactive expression returning a data frame with information about the system.
`show_hz`	A reactive boolean expression determining whether to show habitable zone.
`show_legend`	A reactive boolean expression determining whether to show plot legend.

Details

This module provides a dynamically refreshing plot component for visualizing star systems based on data provided

Value

A Shiny UI object.

Functions

- `visualize_star_system_ui()`: UI function for the system mapping module.
- `visualize_star_system_server()`: server function for the system mapping module.

`plot_star_system` *Plot a Stylized Star System*

Description

Creates a stylized polar plot of a planetary system, displaying planets in circular orbits around a central star. Planet size is scaled by radius, orbit position is randomized for aesthetics, planet color is mapped by density. The star's color is optionally based on spectral type. Optionally, star's habitable zone visualization can be overlayed.

Usage

```
plot_star_system(
  planet_data,
  spectral_type = " ",
  habitable_zone = c(0, 0),
  show_legend = FALSE
)
```

Arguments

`planet_data`	A data frame containing planetary system data. Must include: • `pl_orbsmax`: semi-major axis (orbital distance), • `pl_rade`: planetary radius (in Earth radii), • `pl_dens`: planetary density (g/cm³).
`spectral_type`	Optional character string indicating the star's spectral type. Accepted values: `O`, `B`, `A`, `F`, `G`, `K`, `M`.
`habitable_zone`	Optional numeric vector containing 2 values: Inner and outer habitable zone edges in AU.
`show_legend`	Optional bool value, whether to show plot legend.

Details

The central star is positioned at the origin with planets arranged in orbits of increasing radius. Orbit lines are shown in gray for clarity.

Value

A ggplot2 object representing the planetary system visualization.

Examples

```
# Plot system GJ 682 (with hostid == "2.101289")
data = closest_50_exoplanets |>
  subset(hostid == 2.101289)
spectral_type = classify_star_spectral_type(data$st_teff[1])
```

```
plot_star_system(data,
                 spectral_type,
                 habitable_zone = calculate_star_habitable_zone(data$st_lum[1]))
```

`scatterplot_esi` *Generate a Scatterplot of the Earth Similarity Index (ESI)*

Description

Creates a log-log scatterplot of planetary radius versus stellar flux, colored by the Earth Similarity Index (ESI).

Usage

```
scatterplot_esi(data, plot_limits = c(0.1, 10))
```

Arguments

`data`	A data frame containing exoplanet data. Must include the columns: • `pl_insol`: incident stellar flux (in Earth flux units), • `pl_rade`: planetary radius (in Earth radii), • `esi`: Earth Similarity Index (numeric).
`plot_limits`	A numeric vector of length 2 specifying the lower and upper bounds. Default is `c(0.1, 10)`.

Details

Dashed lines at (1,1) indicate Earth's reference values for stellar flux and radius.

Value

A ggplot2 object representing the scatterplot.

Examples

```
closest_50_exoplanets |>
  dplyr::mutate(esi = calculate_esi(radius = pl_rade, flux = pl_insol)) |>
  scatterplot_esi()
```

summarize_star_occurrences *Summarize star occurrences*

Description

Summarize star occurrences

Usage

```
summarize_star_occurrences(data)
```

Arguments

data	A data frame with KOI data. Must contain kepoi_name column.

Details

The function takes in KOI data frame and summarizes the number of planets that appear in a given dataset for each star.

Value

A data frame containing: - Star column with star ID. - Count column with number of planets.

trim_ps_table *Trim planetary systems table*

Description

Trim planetary systems table

Usage

```
trim_ps_table(data)
```

Arguments

data	A data frame with planetary systems data.

Details

The function takes in a planetary systems data frame, and trims it to include only 20 most important columns.

Value

A data frame containing:

- `objectid` - Object ID
- `pl_name` – Planet name
- `hostname` – Host star name
- `sy_dist` – Distance to the system (parsecs)
- `pl_rade` – Planetary radius (Earth radii)
- `pl_bmasse` – Planetary mass (Earth masses)
- `pl_orbper` – Orbital period (days)
- `pl_orbsmax` – Semi-major axis (AU)
- `pl_orbeccen` – Orbital eccentricity
- `pl_insol` – Incident stellar flux (Earth units)
- `st_teff` – Stellar effective temperature (K)
- `st_rad` – Stellar radius (Solar radii)
- `st_mass` – Stellar mass (Solar masses)
- `st_lum` – Stellar luminosity (log10 L/Lsun)
- `pl_eqt` – Planetary equilibrium temperature (K)
- `pl_dens` – Planetary density
- `discoverymethod` – Discovery method
- `disc_year` – Year of discovery
- `sy_snum` – Number of stars in system
- `sy_pnum` – Number of planets in system

Examples

```
trim_ps_table(closest_50_exoplanets)
```

Index