

Package ‘bggAnalytics’

October 12, 2022

Type Package

Title BoardGameGeek's Board Game Data Analysis Tools

Description Tools for analysing board game data. Mainly focused on providing an interface for BoardGameGeek's XML API2 through R6 class system objects. More details about the BoardGameGeek's API can be obtained here <https://boardgamegeek.com/wiki/page/BGG_XML_API2>.

Version 0.2.1

Author Jakub Bujnowicz [aut, cre]

Maintainer Jakub Bujnowicz <bujnowiczgithub@gmail.com>

URL <https://github.com/JakubBujnowicz/bggAnalytics>

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

Language en-GB

Imports checkmate, pryr, stringr, utils, xml2

Depends R (>= 4.0.0), data.table, R6

Collate 'class_utils.R' 'bggAPI.R' 'bggAnalytics-package.R'
'bggCollection.R' 'bggGames.R' 'bggSearch.R' 'bgg_tools.R'
'custom_fetches.R' 'data.R' 'h_index.R' 'html_utils.R'
'params_utils.R' 'postprocessing.R' 'squeeze.R' 'utils.R'
'zzz.R'

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2022-08-31 12:10:02 UTC

R topics documented:

bggAPI	2
bggCollection	4
bggGames	5
bggSearch	6
bgg_gameurl	8
bgg_merge	8
bgg_namestyle	9
bgg_topgames	10
bgg_variables	10
h_index	11
squeeze	12
Index	13

bggAPI

Interface for BoardGameGeek's XML API2

Description

This is a class that works as a main interface to BoardGameGeek's API. All other bgg* classes inherit from bggAPI. Furthermore, there is no initialization method for bggAPI as it serves just as a super class for other classes.

Active bindings

- `ids` Numeric vector of positive integers. Contains IDs of all BoardGameGeek items that were requested and are present in data. The vector is sorted by default.
- `data` Data.table with currently fetched variables from the object's XML. It has a row count equal to the length of `ids`. Note that the copy is returned, so no modification of data is possible outside of class methods.
- `xml` An XML nodeset obtained through the BoardGameGeek's API. It's length is equal to the length of `ids`.
- `api_url` A character vector of one or more strings with URLs used to fetch XMLs.
- `params` A list with object parameters.
- `timestamp` The date (with time) of the object creation and, what follows, the last moment that the data was certainly up to date.

Methods

Public methods:

- `bggAPI$fetch()`
- `bggAPI$expand()`
- `bggAPI$switch_namestyle()`
- `bggAPI$clone()`

Method `fetch()`: Fetches variables with given `variable_names` from the object's xml. Returns them as a list. This is a main method of getting non-scalar variables (as they are hard to fit into a `data.table`).

Usage:

```
bggAPI$fetch(variable_names = NULL, compress = FALSE)
```

Arguments:

`variable_names` a character vector with names of variables to fetch.

`compress` a logical value, decides whether the fetched variables should be compress into a scalar form (if possible).

Method `expand()`: Expands the data table by given `variable_names` by reference. For the list of available variables for every object check the [bgg_variables](#) dataset.

Usage:

```
bggAPI$expand(variable_names = NULL)
```

Arguments:

`variable_names` a character vector with names of variables to add to data.

Method `switch_namestyle()`: Switches between two styles of naming the variables: 'classic' and 'pretty'. The former is a default value and uses code names concordant with BoardGameGeek's naming convention in the XMLs. The latter is more intuitive and uses UpperCamelCase.

Usage:

```
bggAPI$switch_namestyle(to)
```

Arguments:

`to` a single string, either 'classic' or 'pretty'.

Returns: Returns TRUE or FALSE invisibly depending on whether the names have been switched.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
bggAPI$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

[BoardGameGeek XML API2](#)

 bggCollection

API for user collections

Description

Access the data of a given user's collection. See [bggAPI](#) for more details on inherited slots and methods.

Super class

`bggAnalytics::bggAPI -> bggCollection`

Active bindings

`username` A single string, name of a user whose collection should be fetched.

Methods

Public methods:

- `bggCollection$new()`
- `bggCollection$print()`
- `bggCollection$clone()`

Method `new()`: Object initialization.

Usage:

```
bggCollection$new(username = NULL, params = NULL)
```

Arguments:

`username` a single string with a BoardGameGeek name of a user whose collection is to be fetched.

`params` a list of object parameters. If not all the parameters are included in the list, default values are used (NULL instead of the list is possible for all the default parameters).

Following parameters are allowed for the `bggGames` class with default values in parentheses:

- `pretty_names` - (FALSE) a boolean value, should the object should use pretty names,
- `stats` - (TRUE) a boolean value, should the ranking and rating stats be included for every item. Note that some variables require that `stats` is TRUE.
- `brief` - (FALSE) a boolean value, should the results be abbreviated.
- `own`, `rated`, `played`, `comment`, `trade`, `want`, `wishlist` - (NULL) a boolean value, FALSE excludes items with a given status while TRUE includes only them. NULL returns items regardless of the status.
- `wishlistpriority` - (NULL) a positive integer between 1 and 5, returns only items with a given wishlist priority. NULL returns items regardless of the priority.
- `minrating`, `rating` - (NULL) a positive integer between 1 and 10, returns only items with a given minimum rating (`minrating`) or maximum rating (`rating`). NULL returns items regardless of the rating.

Method print(): Print object information.

Usage:

```
bggCollection$print()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
bggCollection$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

References

[BoardGameGeek XML API2](#)

bggGames

API for Games and Things

Description

This class provides an interface for games, expansions, accessories and other things listed on BoardGameGeek. The official documentation describes ‘things’ as every physical, tangible item. See [bggAPI](#) for more details on inherited slots and methods.

Details

Although this class is named ‘bggGames’, it inherits its functionality from the BoardGameGeek XML API2 ‘Things’ (see References). The name is motivated by the fact that the whole BoardGameGeek’s site as well as this package is mainly focused on board games.

Super class

[bggAnalytics::bggAPI](#) -> bggGames

Methods

Public methods:

- [bggGames\\$new\(\)](#)
- [bggGames\\$print\(\)](#)
- [bggGames\\$clone\(\)](#)

Method new(): Object initialization.

Usage:

```
bggGames$new(ids, chunk_size = 500, params = NULL)
```

Arguments:

ids a numeric vector of positive integers, IDs of games/things to include in the object.

`chunk_size` a positive integer, the maximum length of a chunk that ids are split into. All chunks connect to BoardGameGeek's API separately, so lowering this number increases computation time. On the other hand if a chunk is too long, URL might be too long to fetch.

`params` a list of object parameters. If not all the parameters are included in the list, default values are used (NULL instead of the list is possible for all the default parameters).

Following parameters are allowed for the `bggGames` class with default values in parentheses:

- `pretty_names` - (FALSE) a boolean value, should the object should use pretty names,
- `stats` - (TRUE) a boolean value, should the ranking and rating stats be included for every item. Note that some variables require that `stats` is TRUE.

Method `print()`: Print object information.

Usage:

```
bggGames$print()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
bggGames$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

[BoardGameGeek XML API2](#)

bggSearch

API for BoardGameGeek search engine

Description

Search for items on the BoardGameGeek with a given query. See [bggAPI](#) for more details on inherited slots and methods.

Details

Note that the result is trimmed to items with unique IDs. Due to XML API2 'Search' returning multiple items for a single ID with different types, variable 'type' might be not accurately represented.

Super class

[bggAnalytics::bggAPI](#) -> `bggSearch`

Active bindings

`query` A single string with the wanted query.

Methods

Public methods:

- [bggSearch\\$new\(\)](#)
- [bggSearch\\$print\(\)](#)
- [bggSearch\\$clone\(\)](#)

Method new(): Object initialization.

Usage:

```
bggSearch$new(query, params = NULL)
```

Arguments:

query a single string, query used to perform the search.

params a list of object parameters. If not all the parameters are included in the list, default values are used (NULL instead of the list is possible for all the default parameters).

Following parameters are allowed for the `bggGames` class with default values in parentheses:

- *pretty_names* - (FALSE) a boolean value, should the object should use pretty names,
- *type* - (NULL) a single string, type of things to look for. Possible values: 'rpgitem', 'videogame', 'boardgame', 'boardgameaccessory', 'boardgameexpansion'. NULL uses all possible values.
- *exact* - (FALSE) a boolean value, should the results be restricted to items that match the query exactly.

Method print(): Print object information.

Usage:

```
bggSearch$print()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
bggSearch$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

References

[BoardGameGeek XML API2](#)

`bgg_gameurl`*Get BoardGameGeek URLs for games with given IDs*

Description

This function is a simple wrapper that returns URLs to given games by using their IDs.

Usage

```
bgg_gameurl(ids)
```

Arguments

`ids` a numeric vector of positive integers.

Value

A character vector of the same length as `x`, contains URLs.

Author(s)

Jakub Bujnowicz <bujnowiczgithub@gmail.com>

Examples

```
bgg_gameurl(1:10)
```

`bgg_merge`*Merge data from two bggAPI objects*

Description

This allows for merging data from two bggAPI objects. Namestyle is inherited from `x` and columns from `y$data` are added to `x$data` (unless they are already there).

Usage

```
bgg_merge(x, y, ...)
```

Arguments

`x, y` objects that inherit from bggAPI class.
`...` other arguments passed to [merge](#).

Value

A data.table with variables from both x and y.

Examples

```
sr <- bggSearch$new("Terraforming Mars")
gm <- bggGames$new(sr$ids)

gm$expand(c("name", "type", "rank"))

bgg_merge(sr, gm)
```

bgg_namestyle *Detect the namestyle of the data table*

Description

Detects whether the dt data.table was created by a bggAPI object with 'pretty' or 'classic' names. Ends with the error if one of them cannot be unequivocally determined. This can be useful when working on a modified table that is no longer directly connected to any bggAPI object.

Usage

```
bgg_namestyle(dt)
```

Arguments

dt a data.table from data slot of a bggAPI object.

Value

A single string.

Author(s)

Jakub Bujnowicz <bujnowiczgithub@gmail.com>

Examples

```
gm <- bggGames$new(ids = 167791)
bgg_namestyle(gm$data)

gm$switch_namestyle("pretty")
bgg_namestyle(gm$data)

# Breaks
# bgg_namestyle(iris)
```

bgg_topgames	<i>Get IDs of top rated games on BoardGameGeek</i>
--------------	--

Description

This function scraps BoardGameGeek website for IDs of games with given places in the games ranking.

Usage

```
bgg_topgames(places = 1:100)
```

Arguments

places a numeric vector of positive integers.

Value

Numeric vector of IDs.

Author(s)

Jakub Bujnowicz <bujnowiczgithub@gmail.com>

Examples

```
bgg_topgames()

x <- 1:5 * 25 + 5
ids <- bgg_topgames(sample(x))
gm <- bggGames$new(ids)
gm$expand(c("name", "rank"))
gm
```

bgg_variables	<i>All variables that are available for fetching through bggAPI objects</i>
---------------	---

Description

Contains names and specification of variables that can be used in fetch and expand methods of classes that inherit from bggAPI.

Usage

```
bgg_variables
```

Format

A data.table with the following columns:

Class a character vector, names of class that is able to fetch given variable.

PrettyName, ClassicName a character vector, names of variables in the two available styles, "pretty" and "classic".

Scalar a logical vector, whether the variable is scalar, i.e. does the length of a fetched variable vector is equal to the length of object's ids field. Every scalar variable may be used in the expand methods.

Stats a logical vector, whether the object needs the parameter stats = TRUE to fetch this variable.

Compression a character vector, names of functions that are used to compress variables to scalar variables when using fetch(., compress = TRUE) or expand for non-scalar variables. "NULL" means that a variable is non-compressible.

Details

A variable can be used by the object's extend method if it's Scalar value is TRUE or Compression is not equal to "NULL".

 h_index

H-Index

Description

Calculate H-Index from the number of game plays. H-Index measures the experience of a player based on reported play counts. It rises only when one plays many different games multiple times. It tries to distinguish players who play a few games really often and these who try every game once and leave it on the shelf from those who have the broad collection of high-count plays.

Usage

```
h_index(num_plays)
```

Arguments

num_plays a numeric vector of non-negative integers.

Value

A single non-negative integer.

References

[H-Index in Wikipedia BoardGameGeek thread about H-Index](#)

Examples

```
h_index(0)
h_index(c(0, 0, 1, 2, 1))
h_index(c(2, 2, 5, 100))
h_index(c(2, 3, 5, 100))
```

squeeze

Squeeze integers into a single string

Description

This acts similar to `toString` function, but it tries to make the string as short as possible by squeezing sequences of integers into boundary values only. Please see the examples section. `unsqueeze` reverses this operation.

Usage

```
squeeze(integers)

unsqueeze(strings, strict = FALSE)
```

Arguments

<code>integers</code>	a numeric vector of integers.
<code>strings</code>	a character vector of strings, preferably outputs of <code>squeeze</code> .
<code>strict</code>	a logical value, decides whether the output should be strictly a list. If <code>FALSE</code> and the <code>strings</code> is a single string, the function returns an atomic vector instead.

Value

`squeeze` returns a vector of characters, `unsqueeze` returns a list of numerics or a numeric vector.

Author(s)

Jakub Bujnowicz <bujnowiczgithub@gmail.com>

Examples

```
integers <- c(1, 3:5, NA, 27:38, 10:13, 9:11, 6)
squeeze(integers)

unsqueeze(squeeze(integers))
setdiff(na.omit(integers), unsqueeze(squeeze(integers)))
```

Index

* datasets

- bgg_variables, [10](#)

- bgg_gameurl, [8](#)
- bgg_merge, [8](#)
- bgg_namestyle, [9](#)
- bgg_topgames, [10](#)
- bgg_variables, [3](#), [10](#)
- bggAnalytics::bggAPI, [4–6](#)
- bggAPI, [2](#), [4–6](#)
- bggCollection, [4](#)
- bggGames, [5](#)
- bggSearch, [6](#)

- h_index, [11](#)

- merge, [8](#)

- squeeze, [12](#)

- toString, [12](#)

- unsqueeze (squeeze), [12](#)