

# Package ‘repurrrsive’

July 23, 2025

**Title** Examples of Recursive Lists and Nested or Split Data Frames

**Version** 1.1.0

**Description** Recursive lists in the form of R objects, 'JSON', and 'XML', for use in teaching and examples. Examples include color palettes, Game of Thrones characters, 'GitHub' users and repositories, music collections, and entities from the Star Wars universe. Data from the 'gapminder' package is also included, as a simple data frame and in nested and split forms.

**License** CC0

**URL** <https://jennybc.github.io/repurrrsive/>,  
<https://github.com/jennybc/repurrrsive>

**BugReports** <https://github.com/jennybc/repurrrsive/issues>

**Depends** R (>= 2.10)

**Imports** tibble, utils

**Suggests** jsonlite, testthat (>= 3.0.0), xml2

**Config/Needs/website** dplyr, purrr, tidyr

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.1

**NeedsCompilation** no

**Author** Jennifer Bryan [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-6983-2759>>),  
Charlotte Wickham [ctb],  
Posit PBC [cph, fnd]

**Maintainer** Jennifer Bryan <jenny@rstudio.com>

**Repository** CRAN

**Date/Publication** 2022-12-17 20:50:02 UTC

Contents

|                            |           |
|----------------------------|-----------|
| discog . . . . .           | 2         |
| discog_json . . . . .      | 3         |
| gap_simple . . . . .       | 3         |
| gh_repos . . . . .         | 4         |
| gh_users . . . . .         | 5         |
| gh_users_json . . . . .    | 5         |
| gmaps_cities . . . . .     | 6         |
| got_chars . . . . .        | 7         |
| got_chars_json . . . . .   | 8         |
| sw_people . . . . .        | 8         |
| wesanderson . . . . .      | 10        |
| wesanderson_json . . . . . | 10        |
| <b>Index</b>               | <b>12</b> |

---

|        |                                          |
|--------|------------------------------------------|
| discog | <i>Sharla Gelfand's music collection</i> |
|--------|------------------------------------------|

---

Description

A music collection, as represented in a recursive list returned from the Discogs API.

Usage

discog

Format

A unnamed list with 155 components, each representing an item in Sharla's music collection.

Source

- Data retrieved on 2019-07-15 from <https://www.discogs.com>
- Original blog post by Sharla Gelfand <https://sharla.party/post/discog-purrr/>

See Also

Other Discogs data and functions: [discog\\_json\(\)](#)

Examples

```
length(discog)

str(discog, max.level = 2, list.len = 2)

vapply(discog[1:6], `[`, c("basic_information", "title"), FUN.VALUE = "")
```

---

|             |                                     |
|-------------|-------------------------------------|
| discog_json | <i>Path to Discogs data as JSON</i> |
|-------------|-------------------------------------|

---

**Description**

Path to Discogs data as JSON

**Usage**

```
discog_json()
```

**Value**

Local path to JSON file containing Discogs data

**See Also**

Other Discogs data and functions: [discog](#)

**Examples**

```
discog_json()
if (require("jsonlite")) {
  d <- fromJSON(discog_json(), simplifyVector = FALSE)
  identical(discog, d)
}
```

---

|            |                                              |
|------------|----------------------------------------------|
| gap_simple | <i>Gapminder data frame in various forms</i> |
|------------|----------------------------------------------|

---

**Description**

The main data frame from the gapminder package in three forms:

1. gap\_simple, same as gapminder::gapminder
2. gap\_nested, nested by country and continent
3. gap\_split, split by country

**Usage**

```
gap_simple
```

```
gap_nested
```

```
gap_split
```

**Format**

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1704 rows and 6 columns.

**Examples**

```
gap_simple
gap_nested

str(gap_split, max.level = 1, list.len = 10)
str(gap_split[[1]])
```

---

gh\_repos

*GitHub repos*


---

**Description**

Info on GitHub repos, retrieved from the GitHub API.

**Usage**

```
gh_repos
```

**Format**

A unnamed list with 6 components, each itself a list of 30 repos for a specific GitHub user. Each repo's component is a list of length >60, containing information such as name, owner (a list), fork status, and creation date.

**Source**

<https://developer.github.com/v3/repos/#list-user-repositories>

**See Also**

Other GitHub data and functions: [gh\\_users\\_json\(\)](#), [gh\\_users](#)

**Examples**

```
str(gh_repos, max.level = 1)
str(gh_repos[[1]], max.level = 1)
str(gh_repos[[1]][[1]])

str(lapply(gh_repos[[1]][1:3], `[`, c("full_name", "created_at")))
```

---

`gh_users`*GitHub users*

---

**Description**

Info on GitHub users, retrieved from the GitHub API.

**Usage**

```
gh_users
```

**Format**

A unnamed list with 6 components, each representing a GitHub user. Each user's component is a list of length 30, containing information such as username, GitHub id, and join date.

**Source**

<https://developer.github.com/v3/users/#get-a-single-user>

**See Also**

Other GitHub data and functions: [gh\\_repos](#), [gh\\_users\\_json\(\)](#)

**Examples**

```
str(gh_users, max.level = 1)
str(gh_users[[1]])

str(lapply(gh_users, `[`, c("login", "name")))
```

---

`gh_users_json`*Paths to GitHub data as JSON and XML*

---

**Description**

Paths to GitHub data as JSON and XML

**Usage**

```
gh_users_json()

gh_repos_json()

gh_users_xml()

gh_repos_xml()
```

**Value**

Local path to JSON or XML file containing GitHub data

**See Also**

Other GitHub data and functions: [gh\\_repos](#), [gh\\_users](#)

**Examples**

```
gh_users_json()
if (require("jsonlite")) {
  ghuj <- fromJSON(gh_users_json(), simplifyDataFrame = FALSE)
  identical(gh_users, ghuj)
}
gh_repos_json()
if (require("jsonlite")) {
  ghrj <- fromJSON(gh_repos_json(), simplifyDataFrame = FALSE)
  identical(gh_repos, ghrj)
}
gh_users_xml()
if (require("xml2")) {
  xml <- read_xml(gh_users_xml())
  xml
}
gh_repos_xml()
if (require("xml2")) {
  xml <- read_xml(gh_repos_xml())
  xml
}
```

---

gmaps\_cities

*Geocoded cities from Google Maps*


---

**Description**

This tibble contains the results of geocoding five cities ("Houston", "Washington", "New York", "Chicago", "Arlington") using the Google Maps API on 2022-06-08. Two cities, Washington and Arlington, were deliberately picked for their ambiguity: Washington could refer to the city or the state, and Arlington could mean the one in Virginia or the one in Texas.

**Usage**

```
gmaps_cities
```

**Format**

A tibble with 5 rows and two columns. `city` gives the original search term and `json` gives the returned JSON converted to a list.

**Source**

<https://developers.google.com/maps/documentation/geocoding>

**Examples**

gmaps\_cities

---

got\_chars

*Game of Thrones POV characters*

---

**Description**

Info on the point-of-view (POV) characters from the first five books in the Song of Ice and Fire series by George R. R. Martin. Retrieved from An API Of Ice And Fire.

**Usage**

got\_chars

**Format**

A unnamed list with 30 components, each representing a POV character. Each character's component is a named list of length 18, containing information such as name, aliases, and house allegiances.

**Source**

<https://anapioficeandfire.com>

**See Also**

Other Game of Thrones data and functions: [got\\_chars\\_json\(\)](#)

**Examples**

```
str(got_chars, max.level = 1, list.len = 10)
str(got_chars[[1]])
str(lapply(got_chars, `[`, c("name", "culture")))
```

---

`got_chars_json`*Paths to Game of Thrones data as JSON and XML*

---

**Description**

Paths to Game of Thrones data as JSON and XML

**Usage**

```
got_chars_json()
```

```
got_chars_xml()
```

**Value**

Local path to JSON or XML file containing Game of Thrones data

**See Also**

Other Game of Thrones data and functions: [got\\_chars](#)

**Examples**

```
got_chars_json()
if (require("jsonlite")) {
  gotcj <- fromJSON(got_chars_json(), simplifyDataFrame = FALSE)
  identical(got_chars, gotcj)
}
got_chars_xml()
if (require("xml2")) {
  xml <- read_xml(got_chars_xml())
  xml
}
```

---

`sw_people`*Entities from the Star Wars Universe*

---

**Description**

Data retrieved from the swapi API on the Star Wars Universe.

**Usage**

```
sw_people  
  
sw_films  
  
sw_planets  
  
sw_species  
  
sw_vehicles  
  
sw_starships
```

**Format**

Unnamed lists with varying number of components.

**Details**

- sw\_people List of individual people or characters within the Star Wars universe.
- sw\_starships List of transport crafts with hyperdrive capability.
- sw\_vehicles List of transport crafts without hyperdrive capability.
- sw\_films List of Star Wars films.
- sw\_species List of types of people or characters within the Star Wars Universe.
- sw\_planets List of large masses, planets or planetoids in the Star Wars Universe, at the time of 0 ABY.

**Source**

Data originally obtained from <http://swapi.co/> using the rwars package: <https://github.com/Ironholds/rwars>. The Star Wars API appears to have moved to <https://pipedream.com/apps/swapi> since that time.

**Examples**

```
# sw_people  
str(sw_people, max.level = 1)  
str(sw_people[[1]])  
sapply(sw_people, `[`, "name")  
  
# sw_films  
str(sw_films, max.level = 1)  
str(sw_films[[1]])  
sapply(sw_films, `[`, "title")
```

---

`wesanderson`*Color palettes from Wes Anderson movies*

---

**Description**

A list of color palettes inspired by Wes Anderson movies, taken from the from [wesanderson](#) package.

**Usage**

```
wesanderson
```

**Format**

A named list with 15 components, each containing a color palette from a specific movie. Each palette consists of 4 or 5 hexadecimal color values.

**Source**

<https://cran.r-project.org/package=wesanderson>

<http://wesandersonpalettes.tumblr.com>

**See Also**

Other wesanderson data and functions: [wesanderson\\_json\(\)](#)

**Examples**

```
str(wesanderson)
```

---

`wesanderson_json`*Path to wesanderson JSON and XML*

---

**Description**

Path to wesanderson JSON and XML

**Usage**

```
wesanderson_json()
```

```
wesanderson_xml()
```

**Value**

Local path to JSON or XML file containing Wes Anderson color palettes

**See Also**

Other wesanderson data and functions: [wesanderson](#)

**Examples**

```
wesanderson_json()
if (require("jsonlite")) {
  jsonlite::fromJSON(wesanderson_json())
}
wesanderson_xml()
if (require("xml2")) {
  xml2::read_xml(wesanderson_xml())
}
```

# Index

## \* Discogs data and functions

discog, [2](#)  
discog\_json, [3](#)

## \* Game of Thrones data and functions

got\_chars, [7](#)  
got\_chars\_json, [8](#)

## \* GitHub data and functions

gh\_repos, [4](#)  
gh\_users, [5](#)  
gh\_users\_json, [5](#)

## \* datasets

discog, [2](#)  
gap\_simple, [3](#)  
gh\_repos, [4](#)  
gh\_users, [5](#)  
gmaps\_cities, [6](#)  
got\_chars, [7](#)  
sw\_people, [8](#)  
wesanderson, [10](#)

## \* wesanderson data and functions

wesanderson, [10](#)  
wesanderson\_json, [10](#)

discog, [2](#), [3](#)  
discog\_json, [2](#), [3](#)

gap\_nested(gap\_simple), [3](#)  
gap\_simple, [3](#)  
gap\_split(gap\_simple), [3](#)  
gh\_repos, [4](#), [5](#), [6](#)  
gh\_repos\_json(gh\_users\_json), [5](#)  
gh\_repos\_xml(gh\_users\_json), [5](#)  
gh\_users, [4](#), [5](#), [6](#)  
gh\_users\_json, [4](#), [5](#), [5](#)  
gh\_users\_xml(gh\_users\_json), [5](#)  
gmaps\_cities, [6](#)  
got\_chars, [7](#), [8](#)  
got\_chars\_json, [7](#), [8](#)  
got\_chars\_xml(got\_chars\_json), [8](#)

sw\_films(sw\_people), [8](#)  
sw\_people, [8](#)  
sw\_planets(sw\_people), [8](#)  
sw\_species(sw\_people), [8](#)  
sw\_starships(sw\_people), [8](#)  
sw\_vehicles(sw\_people), [8](#)

wesanderson, [10](#), [10](#), [11](#)  
wesanderson\_json, [10](#), [10](#)  
wesanderson\_xml(wesanderson\_json), [10](#)