Package ‘starwarsdb’

November 3, 2020

Title  Relational Data from the 'Star Wars' API for Learning and Teaching

Version  0.1.2

Description  Provides data about the 'Star Wars' movie franchise in a set of relational tables or as a complete 'DuckDB' database. All data was collected from the open source 'Star Wars' API <https://swapi.dev/>.

License  MIT + file LICENSE

URL  https://github.com/gadenbuie/starwarsdb

BugReports  https://github.com/gadenbuie/starwarsdb/issues

Depends  R (>= 2.10)

Imports  DBI, duckdb, magrittr, tibble

Suggests  dbplyr, dm, dplyr, testthat

Encoding  UTF-8

LazyData  true

RoxygenNote  7.1.1

NeedsCompilation  no

Author  Garrick Aden-Buie [aut, cre] (<https://orcid.org/0000-0002-7111-0077>)

Maintainer  Garrick Aden-Buie <garrick@adenbuie.com>

Repository  CRAN

Date/Publication  2020-11-02 23:50:02 UTC

R topics documented:

films  ................................................................. 2
films_people  ......................................................... 3
films_planets  ......................................................... 3
films_vehicles  ......................................................... 4
people  ............................................................... 4
pilots  ............................................................... 5
Description
Films in the Star Wars movie franchise.

Usage
films

Format
A data frame with 6 rows and 6 variables:

- title: The title of this film.
- episode_id: The episode number of this film.
- opening_crawl: The opening crawl text at the beginning of this film.
- director: The director of this film.
- producer: The producer(s) of this film.
- release_date: The release date at original creator country.

References
https://swapi.dev/
**films_people**  
*People in Films*

**Description**
Links characters (people) to the films in which they appear.

**Usage**
```r
films_people
```

**Format**
A data frame with 162 rows and 2 variables:
- `title`  The title of the film.
- `character`  The name of the character who appeared in the film.

**References**
https://swapi.dev/

---

**films_planets**  
*Planets in Films*

**Description**
Links planets to the films in which they appear.

**Usage**
```r
films_planets
```

**Format**
A data frame with 33 rows and 2 variables:
- `title`  The title of the film.
- `planet`  The name of the planet that appeared in the film.

**References**
https://swapi.dev/
**films_vehicles**  
*Vehicles in Films*

**Description**  
Links vehicles to the films in which they appear.

**Usage**  
films_vehicles

**Format**  
A data frame with 104 rows and 2 variables:
- **title**: The title of the film.
- **vehicle**: The name of the vehicle that appeared in the film.

**References**
https://swapi.dev/

**people**  
*People*

**Description**  
Characters within the Star Wars universe.

**Usage**  
people

**Format**  
A data frame with 82 rows and 10 variables:
- **name**: The name of this person.
- **height**: The height of this person in meters.
- **mass**: The mass of this person in kilograms.
- **hair_color**: The hair color of this person.
- **skin_color**: The skin color of this person.
- **eye_color**: The eye color of this person.
- **birth_year**: The birth year of this person. BBY (Before the Battle of Yavin) or ABY (After the Battle of Yavin).
The biological sex of the character. One of male, female, hermaphroditic, or none.
gender  The gender role or gender identity of the character.
homeworld  The planet the character was born on.
species  The species of the character.

References

https://swapi.dev/

<table>
<thead>
<tr>
<th>pilots</th>
<th>Pilots</th>
</tr>
</thead>
</table>

Description

Links people to the vehicles they have piloted.

Usage

pilots

Format

A data frame with 43 rows and 2 variables:
pilot  The name of the person who piloted the vehicle.
vehicle  The name of the vehicle that was piloted.

References

https://swapi.dev/

<table>
<thead>
<tr>
<th>planets</th>
<th>Planets</th>
</tr>
</thead>
</table>

Description

Planets in the Star Wars universe.

Usage

planets
Format

A data frame with 59 rows and 9 variables:

name  The name of this planet.
rotation_period  The number of standard hours it takes for this planet to complete a single rotation on its axis.
orbital_period  The number of standard days it takes for this planet to complete a single orbit of its local star.
diameter  The diameter of this planet in kilometers.
climate  The climate of this planet. Comma-separated if diverse.
gravity  A number denoting the gravity of this planet. Where 1 is normal.
terrain  The terrain of this planet. Comma-separated if diverse.
surface_water  The percentage of the planet surface that is naturally occurring water or bodies of water.
population  The average population of sentient beings inhabiting this planet.

References

https://swapi.dev/

schema

Star Wars Data Schema

Description

Includes information about the schema of the tables that were sourced from SWAPI, the Star Wars API. Not all properties returned from the API are columns in the data in this package: some properties were refactored into separate tables. For example, I combined the starships/ and vehicles/ endpoint into a single table. Both API endpoints returned a "pilots" property, which is described in the schema as an array of people who piloted the vehicle. The information in this property has been extracted into a separate table called pilots in the starwarsdb package.

Usage

schema

Format

A data frame with 5 rows and 4 variables:

endpoint  The name of the SWAPI endpoint.
endpoint_title  The title of the SWAPI endpoint.
endpoint_description  The description of the SWAPI endpoint.
properties  The properties of the endpoint as a nested table containing the variable, the data type, a description and the format of the property.
**Species**

**References**

https://swapi.dev/

---

<table>
<thead>
<tr>
<th>species</th>
<th>Species</th>
</tr>
</thead>
</table>

**Description**

Species within the Star Wars universe.

**Usage**

species

**Format**

A data frame with 37 rows and 10 variables:

- name: The name of this species.
- classification: The classification of this species.
- designation: The designation of this species.
- average_height: The average height of this person in centimeters.
- skin_colors: A comma-separated string of common skin colors for this species, none if this species does not typically have skin.
- hair_colors: A comma-separated string of common hair colors for this species, none if this species does not typically have hair.
- eye_colors: A comma-separated string of common eye colors for this species, none if this species does not typically have eyes.
- average_lifespan: The average lifespan of this species in years.
- homeworld: The URL of a planet resource, a planet that this species originates from.
- language: The language commonly spoken by this species.

**References**

https://swapi.dev/
Description

Provides a connection to a DuckDB database of the Star Wars data. Alternatively, you can use `starwars_db()` to manually connect to the database using `DBI::dbConnect()` and `duckdb::duckdb()`.

Usage

```
starwars_connect(dbdir = "memory", ...)  
starwars_disconnect(con)  
starwars_db()  
```

Arguments

- `dbdir` Location for database files. Should be a path to an existing directory in the file system. With the default, all data is kept in RAM
- `...` Additional parameters passed to `DBI::dbConnect()`
- `con` A connection to the Star Wars database

Value

A connection to the Star Wars database, or the path to the database.

Functions

- `starwars_connect`: Connect to the DuckDB database
- `starwars_disconnect`: Disconnect from the DuckDB database
- `starwars_db`: Returns the path to the starwarsdb database

Examples

```
# Manually connect using {duckdb} and {DBI}
con <- DBI::dbConnect(  
  duckdb::duckdb(),  
  dbdir = starwars_db(),  
  read_only = TRUE  
)

if (requireNamespace("dplyr", quietly = TRUE)) {
  dplyr::tbl(con, "films")
}

# Disconnect from that database (shutdown is specific to duckdb)
```
Create a Star Wars Data Model Object

Description

Creates a \texttt{dm} object with the starwarsdb tables.

Usage

\begin{verbatim}
starwars_dm(configure_dm = TRUE, remote = FALSE)

starwars_dm_configure(dm)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{configure_dm} \hspace{1cm} If TRUE (default) the returned \texttt{dm} object is completely configured with all of the primary and foreign keys. Set to FALSE if you want to practice configuring the \texttt{dm} object yourself.
  \item \texttt{remote} \hspace{1cm} If TRUE, uses the internal DuckDB database rather than local tables, which is the default.
  \item \texttt{dm} \hspace{1cm} A \texttt{dm} object with the starwarsdb tables
\end{itemize}

Value

A \texttt{dm} object

Functions

- \texttt{starwars_dm_configure}: Configure the starwars \texttt{dm} object with primary and foreign keys and colors.

See Also

\texttt{dm::dm()}, \texttt{dm::dm_add_pk()}, \texttt{dm::dm_add_fk()}, \texttt{dm::dm_from_src()}

DBI::dbDisconnect(con, shutdown = TRUE)

# Or connect without worrying about connection details
con <- starwars_connect()

if (requireNamespace("dplyr", quietly = TRUE)) {
  dplyr::tbl(con, "films")
}

# Similarly, disconnect quickly without worrying about duckdb arguments
starwars_disconnect(con)
Examples

```r
# If the {dm} package is installed...
if (requireNamespace("dm", quietly = TRUE)) {
    # Create a full starwars (dm) object from local tables
    starwars_dm(remote = TRUE)

    # Create a base starwars (dm) object from remote tables without keys
    starwars_dm(configure_dm = FALSE, remote = TRUE)
}
```

### vehicles

**Starships or Vehicles**

#### Description

A Starship or vehicle in the Star Wars universe.

#### Usage

`vehicles`

#### Format

A data frame with 75 rows and 14 variables:

- **name**: The name of this vehicle. The common name, such as Sand Crawler.
- **type**: The type of the vehicle: starship or vehicle.
- **class**: The class of the vehicle, source from `starship.class` or `vehicle.class`.
- **model**: The model or official name of this vehicle. Such as All Terrain Attack Transport.
- **manufacturer**: The manufacturer of this vehicle. Comma seperated if more than one.
- **cost_in_credits**: The cost of this vehicle new, in galactic credits.
- **length**: The length of this vehicle in meters.
- **max_atmosphering_speed**: The maximum speed of this vehicle in atmosphere.
- **crew**: The number of personnel needed to run or pilot this vehicle.
- **passengers**: The number of non-essential people this vehicle can transport.
- **cargo_capacity**: The maximum number of kilograms that this vehicle can transport.
- **consumables**: The maximum length of time that this vehicle can provide consumables for its entire crew without having to resupply.
- **hyperdrive_rating**: The class of this starship’s hyperdrive.
- **MGLT**: The Maximum number of Megalights this starship can travel in a standard hour. A Megalight is a standard unit of distance and has never been defined before within the Star Wars universe. This figure is only really useful for measuring the difference in speed of starships. We can assume it is similar to AU, the distance between our Sun (Sol) and Earth.
vehicles

References

https://swapi.dev/
Index

* datasets
  films, 2
  films_people, 3
  films_planets, 3
  films_vehicles, 4
  people, 4
  pilots, 5
  planets, 5
  schema, 6
  species, 7
  vehicles, 10

DBI::dbConnect(), 8
dm::dm(), 9
dm::dm_add_fk(), 9
dm::dm_add_pk(), 9
dm::dm_from_src(), 9
duckdb::duckdb(), 8

films, 2
films_people, 3
films_planets, 3
films_vehicles, 4

people, 4
pilots, 5
planets, 5

schema, 6
species, 7
starwars_connect(starwars_db), 8
starwars_db, 8
starwars_disconnect(starwars_db), 8
starwars_dm, 9
starwars_dm_configure(starwars_dm), 9

vehicles, 10