Package 'rtrek'

July 23, 2025

Title Data Analysis Relating to Star Trek

Version 0.5.2

Description Provides datasets related to the Star Trek fictional universe and functions for working with the data.
The package also provides access to real world datasets based on the televised series and other related licensed media productions.
It interfaces with the Star Trek API (STAPI) (<http://stapi.co/>), Memory Alpha (<https://memory-alpha.fandom.com/wiki/Portal:Main>), and Memory Beta (<https://memory-beta.fandom.com/wiki/Main_Page>) to retrieve data, metadata and other information relating to Star Trek.
It also contains several local datasets covering a variety of topics.
The package also provides functions for working with data from other Star Trek-related R data packages containing larger datasets not stored in 'rtrek'.

License MIT + file LICENSE

URL https://github.com/leonawicz/rtrek

BugReports https://github.com/leonawicz/rtrek/issues

Depends R (>= 4.1)

Imports downloader, dplyr, ggplot2, jpeg, jsonlite, memoise, png, purrr, rvest, tibble, tidyr, xml2

Suggests ggrepel, gridExtra, knitr, leaflet, leaflet.extras, lubridate, rmarkdown, showtext, sysfonts, testthat, trekfont

ByteCompile true

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Matthew Leonawicz [aut, cre] (ORCID: <https://orcid.org/0000-0001-9452-2771>)

Maintainer Matthew Leonawicz <rpkgs@pm.me>

Repository CRAN

Date/Publication 2025-06-19 23:40:01 UTC

Contents

`ma_article` *Read Memory Alpha article*

Description

Read Memory Alpha article content and metadata.

Usage

```
ma_article(
  url,
  content_format = c("xml", "character"),
  content_nodes = c("h1", "h2", "h3", "h4", "h5", "h6", "p", "b", "ul"),
  browse = FALSE
)
```

Arguments

`url`	character, article URL. Expects package-style short URL. See examples.
`content_format`	character, the format of the article main text, `"xml"` or `"character"`.
`content_nodes`	character, which top-level nodes in the article main text to retain.
`browse`	logical, also open `url` in browser.

Details

Article content is returned in a nested, tidy data frame.

Value

a nested data frame

Examples

```
ma_article("Azetbur")
```

`ma_image` *Memory Alpha images*

Description

Download a Memory Alpha image and return a ggplot object.

Usage

```
ma_image(url, file, keep = FALSE)
```

Arguments

`url`	character, the short URL of the image, for example as returned by `memory_alpha()`. Must be JPG or PNG. See example.
`file`	character, output file name. Optional. See details.
`keep`	logical, if `FALSE` (default) then `file` is only temporary.

Details

By default the downloaded file is not retained (`keep = FALSE`). The filename is derived from `url` if `file` is not provided. Whether or not the output file is kept, a ggplot object of the image is returned.

Value

a ggplot object

Examples

```
## Not run: ma_image("File:Gowron_attempts_to_recruit_Worf.jpg")
```

`ma_search` *Memory Alpha site search*

Description

Perform a Memory Alpha site search.

Usage

```
ma_search(text, browse = FALSE)
```

Arguments

`text`	character, search query.
`browse`	logical, open search results page in browser.

Details

This function returns a data frame containing the title, truncated text preview, and relative URL for the first page of search results. It does not recursively collate search results through subsequent pages of results. There could be an unexpectedly high number of pages of results depending on the search query. Since the general nature of this search feature seems relatively casual anyway, it aims only to provide a first page preview.

Value

a data frame

Examples

```
ma_search("Worf")
```

mb_article *Read Memory Beta article*

Description

Read Memory Beta article content and metadata.

Usage

```
mb_article(
  url,
  content_format = c("xml", "character"),
  content_nodes = c("h1", "h2", "h3", "h4", "h5", "h6", "p", "b", "ul"),
  browse = FALSE
)
```

Arguments

`url`	character, article URL. Expects package-style short URL. See examples.
`content_format`	character, the format of the article main text, `"xml"` or `"character"`.
`content_nodes`	character, which top-level nodes in the article main text to retain.
`browse`	logical, also open `url` in browser.

Details

Article content is returned in a nested, tidy data frame.

Value

a nested data frame

Examples

```
mb_article("Azetbur")
```

mb_image *Memory Beta images*

Description

Download a Memory Beta image and return a ggplot object.

Usage

```
mb_image(url, file, keep = FALSE)
```

Arguments

`url`	character, the short URL of the image, for example as returned by `memory_beta()`. Must be JPG or PNG. See example.
`file`	character, output file name. Optional. See details.
`keep`	logical, if `FALSE` (default) then `file` is only temporary.

Details

By default the downloaded file is not retained (`keep = FALSE`). The filename is derived from `url` if `file` is not provided. Whether or not the output file is kept, a ggplot object of the image is returned.

Value

a ggplot object

Examples

```
## Not run: mb_image("File:DataBlaze.jpg")
```

`mb_search` *Memory Beta site search*

Description

Perform a Memory Beta site search.

Usage

```
mb_search(text, browse = FALSE)
```

Arguments

`text`	character, search query.
`browse`	logical, open search results page in browser.

Details

This function returns a data frame containing the title, truncated text preview, and relative URL for the first page of search results. It does not recursively collate search results through subsequent pages of results. There could be an unexpectedly high number of pages of results depending on the search query. Since the general nature of this search feature seems relatively casual anyway, it aims only to provide a first page preview.

Value

a data frame

Examples

```
mb_search("Worf")
```

`mb_timeline` *Memory Beta timeline*

Description

Access curated data frames containing Star Trek timeline data.

Usage

```
mb_timeline(x, html = FALSE)
```

Arguments

`x`	numeric or character, description of the desired timeline window. See details.
`html`	logical, set to `TRUE` to return the `details` text column of the `Events` data frame as basic HTML character strings.

Details

The timeline data is from the Memory Beta Chronology.

`x` can be a numeric vector of years, e.g. `x = 2361:2364`. This should only be used if you know (or can safely assume) a year exists as a page on Memory Beta. Check there first if unsure. `x` may otherwise be scalar character. This can be a specific decade in the form, e.g., `"2370s"`. If a decade, it must fall in the range from `"1900s"` through `"2490s"`. The decade option pulls back data from the decade page entry, or if individual year pages exist within the given decade, it will pull the data for each existing year.

Special values: For the more distant past or future, use the character options `x = "past"` or `x = "future"`. `x = "main"` will pull from the main part of the timeline, 1900 - 2499. `x = "complete"` combines past, main, and future in order.

The distant past and future have few entries, and thus few pages. However, both of these last two options, `"main"` and `complete`, must download a large number of pages. For this reason, `rtrek` employs anti-DOS measures to prevent an unwitting user from making too many requests too quickly from Memory Beta. The function would otherwise be far faster. However, to be a friendly neighbor in the cosmos, `rtrek` enforces a minimum one-second wait between timeline requests. This can lead to downloading the full timeline to take ten minutes or so even if you have a fast connection; most of the time it takes is spent waiting patiently.

Also, like other functions that work with Memory Alpha and Memory Beta data, `mb_timeline()` wraps around internal functions that are sensibly memoized. This means that if you make the same call twice in your R session, you won't have to wait at all, because the result is cached in memory. The call will appear to run instantaneously the second time around, but that's because nothing is happening other than returning the cached result from the initial call.

Value

a list of two data frames

Examples

```
mb_timeline(2360)
## Not run:
mb_timeline("2360s")
mb_timeline("past")
mb_timeline("future")
mb_timeline("main")
mb_timeline("complete")
mb_timeline("complete", html = TRUE)

## End(Not run)
```

`memory_alpha` *Memory Alpha API*

Description

Access Star Trek content from Memory Alpha.

Usage

```
memory_alpha(endpoint)
```

Arguments

`endpoint`	character, See details.

Details

The content returned is always a data frame. The structure changes slightly depending on the nature of the endpoint, but results from different endpoints can be merged easily.

Value

a data frame

Portals

At the highest level, passing `endpoint = "portals"` returns a data frame listing the available Memory Alpha portals supported by `rtrek`. A column of relative URLs is also included for reference, but can be ignored.

Portal Categories

In all other cases, the endpoint string must begin with one of the valid portal IDs. Passing only the ID returns a data frame with IDs and relative URLs associated with the available categories in the specific portal. There are two additional columns, `group` and `subgroup`, that may provide additional grouping context for the entry IDs in larger tables. As with the relative URLs, you do not have to make explicit use of these variables.

Selecting a specific category within a portal is done by appending the portal ID in `endpoint` with the category ID, separated by a forward slash. You can append nested subcategory IDs with forward slashes, provided the subcategories exist.

Articles

When the endpoint is neither a top-level portal or one of a portal's categories (or subcategories, if available), it is an article. An article is a terminal node, meaning you cannot nest further. An article will be any entry whose URL does not begin with `Category:`. In this case, the content returned is still a data frame for consistency, but differs substantially from the results of non-terminal endpoints.

Memory Alpha is not a database containing convenient tables. Articles comprise the bulk of what Memory Alpha has to offer. They are not completely unstructured text, but are loosely structured. Some assumptions are made and `memory_alpha()` returns a data frame containing article text and links. It is up to the user what to do with this information, e.g., performing text analyses.

Additional Notes

The `url` column included in results for context uses relative paths to save space. The full URLs all begin the same. To visit a URL directly, prepend it with `https://memory-alpha.fandom.com/wiki/`.

Also note that once you know the relative URL for an article, e.g., `"Worf"`, you do not need to traverse through one of the portals using an `endpoint` string to retrieve its content. You can instead use `ma_article("Worf")`.

`memory_alpha()` provides an overview perspective on how content available at Memory Alpha is organized and can be searched for through a variety of hierarchical layouts. And in some cases this structure that can be obtained in table form can be useful as data or metadata in itself. Alternatively, `ma_article()` is focused exclusively on pulling back content from known articles.

See Also

`ma_article()`, `memory_beta()`

Examples

```
memory_alpha("portals") # show available portals

memory_alpha("people") # show portal categories for People portal
memory_alpha("people/Klingons") # show people in Klingons subcategory
memory_alpha("people/Klingons/Worf") # return terminal article content
```

`memory_beta` *Memory Beta API*

Description

Access Star Trek content from Memory Alpha.

Usage

```
memory_beta(endpoint)
```

Arguments

`endpoint`	character, See details.

Details

The content returned is always a data frame. The structure changes slightly depending on the nature of the endpoint, but results from different endpoints can be merged easily.

Value

a data frame

Portals

At the highest level, passing `endpoint = "portals"` returns a data frame listing the available Memory Beta portals supported by `rtrek`. A column of relative URLs is also included for reference, but can be ignored. Compared to Memory Alpha, Memory Beta does not technically offer "portals", but for consistency in `rtrek`, several high level categories on Memory Beta are treated as portal options. See `memory_alpha()` for comparison.

Portal Categories

In all other cases, the endpoint string must begin with one of the valid portal IDs. Passing only the ID returns a data frame with IDs and relative URLs associated with the available categories in the specific portal. Unlike `memory_alpha()`, there are no `group` or `subgroup` columns. Memory Beta offers a more consistent reliance on the simple hierarchy of categories and articles.

Selecting a specific category within a portal is done by appending the portal ID in `endpoint` with the category ID, separated by a forward slash. You can append nested subcategory IDs with forward slashes, provided the subcategories exist.

Articles

When the endpoint is neither a top-level portal or one of a portal's categories (or subcategories, if available), it is an article. An article is a terminal node, meaning you cannot nest further. An article will be any entry whose URL does not begin with `Category:`. In this case, the content returned is still a data frame for consistency, but differs substantially from the results of non-terminal endpoints.

Memory Beta is not a database containing convenient tables. Articles comprise the bulk of what Memory Beta has to offer. They are not completely unstructured text, but are loosely structured. Some assumptions are made and `memory_beta()` returns a data frame containing article text and links. It is up to the user what to do with this information, e.g., performing text analyses.

Additional Notes

The `url` column included in results for context uses relative paths to save space. The full URLs all begin the same. To visit a URL directly, prepend it with `https://memory-beta.fandom.com/wiki/`.

Also note that once you know the relative URL for an article, e.g., `"Worf"`, you do not need to traverse through one of the portals using an `endpoint` string to retrieve its content. You can instead use `mb_article("Worf")`.

`memory_beta()` provides an overview perspective on how content available at Memory Beta is organized and can be searched for through a variety of hierarchical layouts. And in some cases this structure that can be obtained in table form can be useful as data or metadata in itself. Alternatively, `mb_article()` is focused exclusively on pulling back content from known articles.

See Also

`mb_article()`, `memory_alpha()`

Examples

```
memory_beta("portals") # show available portals
endpoint <- "characters/Characters by races and cultures/Klingonoids/Klingons"

x <- memory_beta(endpoint)
x <- x[grep("Worf", x$Klingons), ]
x
memory_beta(paste0(endpoint, "/Worf")) # return terminal article content
```

---

rtrek *rtrek: Data Analysis Relating to Star Trek*

Description

Provides datasets related to the Star Trek fictional universe and functions for working with the data. The package also provides access to real world datasets based on the televised series and other related licensed media productions. It interfaces with the Star Trek API (STAPI) (http://stapi.co/), Memory Alpha (https://memory-alpha.fandom.com/wiki/Portal:Main), and

Memory Beta (https://memory-beta.fandom.com/wiki/Main_Page) to retrieve data, metadata and other information relating to Star Trek. It also contains several local datasets covering a variety of topics. The package also provides functions for working with data from other Star Trek-related R data packages containing larger datasets not stored in 'rtrek'.

Author(s)

Maintainer: Matthew Leonawicz <rpkgs@pm.me> (ORCID)

See Also

Useful links:

- https://github.com/leonawicz/rtrek
- Report bugs at https://github.com/leonawicz/rtrek/issues

`stapi` *Retrieve Star Trek data from STAPI*

Description

Retrieve Star Trek data from the Star Trek API (STAPI).

Usage

```
stapi(id, page = 1, uid = NULL, page_count = FALSE)
```

Arguments

`id`	character, name of STAPI entity. See details.
`page`	integer vector, defaults to first page.
`uid`	`NULL` for search mode, character for extraction mode. See details.
`page_count`	logical, set to `TRUE` to do a preliminary check of the total number a pages of results available for a potential entity search. This will only have the impact of searching the first page.

Details

See `stapiEntities()` for all the currently available API entities. These are the IDs for dataset collections or categories passed to `id`.

The universal ID `uid` can be supplied to retrieve a more specific subset of data. By default, `uid = NULL` and `stapi()` operates in search mode. As part of a stepwise process, you can first use search mode. Then if the resulting data frame includes a `uid` column, you can make a second call to the function providing a specific `uid`. This puts `stapi()` into extraction mode and will return satellite data associated with the unique entry from the original general sweep of the entity `id`.

`rtrek` employs anti-DOS measures. It will not perform an API call to STAPI more than once per second. To be an even better neighbor, you can increase this wait time using `options()`, e.g.

`options(rtrek_antidos = 10)` to increase the minimum time between API calls to ten seconds. Values less than one are ignored (defaulting back to one second) and a warning will be thrown when making any API call if this is the case.

Currently STAPI contains primarily real world data such as episode air dates, movie metadata, or production company information. Fictional world data is secondary and more limited.

Value

a data frame in search mode, a list in extraction mode, and nothing is returned in page count check mode but the result is printed to the console.

Examples

```
library(dplyr)
stapi("character", page_count = TRUE) # check first
stapi("character", page = 2) |> select(1:2)
Q <- stapi("character", uid = "CHMA0000025118")
Q$episodes |> select(uid, title, stardateFrom, stardateTo)
```

`stapiEntities` *Star Trek API entities.*

Description

A data frame with 40 rows and 4 columns listing the available STAPI entity IDs that can be passed to `stapi()`, along with additional metadata regarding the content returned form an API call to each entity. This data frame helps you see what you will obtain from API calls beforehand. Every entity search returns a tibble data frame, with varying numbers of columns and different names depending on the entity content. There is also one nested column containing the column names of the data frame returned for each entity. This can be inspected directly for specific entities or `stapiEntities` can be unnested with a function like `tidyr::unnest()`.

Usage

```
stapiEntities
```

Format

A data frame

See Also

`stapi()`

stBooks *Star Trek novel metadata.*

Description

A data frame with 783 rows and 11 columns containing metadata on Star Trek novels and other books taken directly from original books. The data frame contains most of the novels but is not comprehensive and may be out of date temporarily whenever new novels are published. It is largely complete through the end of 2017, though some older entries are still missing.

Usage

```
stBooks
```

Format

A data frame

Details

`stBooks`: There may be some irregularities or erroneous entries based on the imperfect methods use to compile the metadata, but it is overall an accurate dataset.

The `nchap` column is largely accurate, but imperfect. Some entries suggest a book has an unusual number of chapters, but the parser is not perfect at determining what constitutes a chapter. However, many of the books with unusually high numbers of chapters are not erroneous but rather indicate a reference book, omnibus or anthology, as opposed to a standard novel.

See Also

`stSeries()`, `st_books_wiki()`

stGeo *Raster grid location data for stellar cartographic map tile sets.*

Description

A data frame of with 18 rows and 4 columns. This data frame has an ID column for map tile set, a column of location names, and columns of respective column and row number of each location per map tile set.

Usage

```
stGeo
```

Format

A data frame

stLogos *Star Trek logos metadata.*

Description

A data frame with 236 rows and 3 columns containing Star Trek logo metadata: category, description and URL. Logo artwork credited to Kris Trigwell. The logo images are served by st-minutiae.com for personal and fair use.

Usage

```
stLogos
```

Format

A data frame

See Also

`st_logo()`

stSeries *Star Trek series.*

Description

A data frame with 35 rows and 3 columns containing names and abbreviations of Star Trek series and anthologies. There are so many because the table pertains to written works, which is inclusive of the more limited televised series.

Usage

```
stSeries
```

Format

A data frame

Details

Some entries listed as series can be interpreted as miniseries, but that distinction is not made here. The official line between the two is not always clear and can also change as more novels are released.

Anthologies are listed as such, rather than as series. Reference manuals have a distinct entry. The Miscellaneous category can be considered synonymous with All-Series/Crossover, abbreviated elsewhere as simply `ST` for Star Trek in general, rather than as `MISC`.

See Also

st_books_wiki()

---

stSpecies *Species names and avatars, linked primarily from Memory Alpha.*

Description

A data frame with 9 rows and 2 columns.

Usage

```
stSpecies
```

Format

A data frame

---

stTiles *Available Star Trek map tile sets.*

Description

A data frame with 2 row and 8 columns.

Usage

```
stTiles
```

Format

A data frame

st_books_wiki *Go to Wikipedia entry for a specific book series*

Description

This function opens a browser tab to the main Wikipedia entry for all Star Trek novels. For a more complete set of Star Trek series, miniseries and anthology names and acronyms, see the stSeries and stBooks datasets.

Usage

```
st_books_wiki()
```

Value

opens a browser tab, nothing is returned.

See Also

stBooks stSeries

Examples

```
## Not run: st_books_wiki()
```

st_datasets *Available datasets*

Description

List the available datasets in the rtrek package.

Usage

```
st_datasets()
```

Value

a character vector.

Examples

```
st_datasets()
```

st_font *Preview Star Trek fonts*

Description

This function produces a plot showing a preview of a Star Trek font from the `trekfont` package. It will return a message if any of `trekfont`, `showtext` or `ggplot2` are not installed. If `family` is missing, it will return a vector of all available font families.

Usage

```
st_font(family, size = 11)
```

Arguments

`family`	character, font family.
`size`	numeric, font size passed to ggplot.

Details

In RStudio on Windows the font may not show in the RStudio graphics device. Try using the regular R GUI.

Value

a character vector, or a plot side effect. See details.

Examples

```
if(all(c("trekfont", "showtext", "ggplot2") %in% installed.packages())){
  st_font()
}
## Not run: st_font("Federation") # should be run in an interactive session
```

st_logo *Star Trek logos*

Description

Download an image of a Star Trek logo and return a ggplot object.

Usage

```
st_logo(url, file, keep = FALSE)
```

Arguments

`url`	character, the url of the image, must be one from the dataset `stLogos()`. See example.
`file`	character, output file name. Optional. See details.
`keep`	logical, if `FALSE` (default) then `file` is only temporary.

Details

By default the downloaded file is not retained (`keep = FALSE`). The filename is derived from `url` if `file` is not provided. These files are all .gif. Whether or not the output file is kept, a ggplot object of the image is returned. For more information on attribution, see `stLogos()`.

Value

a ggplot object

See Also

`stLogos()`

Examples

```
## Not run: st_logo(stLogos$url[1])
```

`st_tiles` *Return the URL associated with a tile set*

Description

This function returns the URL associated with a tile set matching `id`.

Usage

```
st_tiles(id)
```

Arguments

`id`	character, name of map tile set ID. See stTiles.

Details

Tile set data are stored in the stTiles dataset. See for available IDs.

Value

a character string.

See Also

stTiles, st_tiles_data()

Examples

```
st_tiles("galaxy1")
```

---

st_tiles_data *Ancillary location data for map tiles*

Description

Obtain a table of ancillary data associated with various locations of interest, given a specific map tile set ID.

Usage

```
st_tiles_data(id)
```

Arguments

id	character, name of a map tile set.

Details

This function returns a small example data frame of location-specific data along with grid cell coordinates that are specific to the requested map tile set ID.

Value

a data frame

See Also

stTiles, st_tiles()

Examples

```
st_tiles_data("galaxy2")
```

`st_transcripts` *Import transcripts*

Description

Download a curated data frame based on episode and movie transcripts containing metadata and variables for analysis of scenes, character presence, dialog, sentiment, etc.

Usage

```
st_transcripts(type = c("clean", "raw"))
```

Arguments

`type`	character, `"clean"` for curated nested data frame or `"raw"` for unprocessed text. See details.

Details

The data frame contains metadata associated with each transcript, one row per episode. It also contains a list column. By default (`type = "clean"`), this is a nested data frame of preprocessed text split into several variables including the speaking character, line spoken, scene descriptions, etc. For the raw text version, the list column contains vectors of unprocessed plain text.

Metadata includes the format (episode or movie), series, season, overall episode number, title, production order and original airdate if available and applicable. The two columns `url` and `url2` show where source material can be browsed online, though not in a useful format for data analysis. The first set is used if possible because it contains more complete, higher quality data. When necessary, the derived data is based on text from the alternate source.

The dataset is nicely curated, but imperfect. There are text-parsing edge cases that are difficult to handle generally. The quality varies substantially across series. Datasets assembled based on original transcripts are more informative, but not universally available. Other episodes are based on transcripts derived from closed captioning, in which case more fields will contain `NA` values.

This function downloads and returns a sizable tibble data frame. Each version is about 13-15 MB compressed. The returned tibble contains 726 rows (716 episodes and 10 movies), but each row has nested data.

Value

a tibble data frame

Examples

```
## Not run: stTranscripts <- st_transcripts()
```

theme_rtrek *ggplot2 themes*

Description

A collection of ggplot2 themes.

Usage

```
theme_rtrek(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_rtrek_dark(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)
```

Arguments

base_size	base font size.
base_family	base font family.
base_line_size	base size for line elements.
base_rect_size	base size for rect elements.

tile_coords *Simple CRS coordinates*

Description

Convert `(column, row)` numbers to `(x, y)` coordinates for a given tile set.

Usage

```
tile_coords(data, id)
```

Arguments

data	a data frame containing columns named `col` and `row`. These contain column-row number pairs defining matrix cells in tile set `id`. See details.
id	character, name of map tile set ID. See stTiles.

Details

This function converts column and row indices for an available map tile set matrix to coordinates that can be used in a Leaflet map. See stTiles for available tile sets.

`data` cannot contain columns named `x` or `y`, which are reserved for the column-appended output data frame.

Each tile set has a simple/non-geographical coordinate reference system (CRS). Respective coordinates are based on the dimensions of the source image used to generate each tile set. The same column and row pair will yield different map coordinates for different tile sets. Typical for matrices, columns are numbered increasing from left to right and rows increasing from top to bottom. The output of `tile_coords()` is a typical Cartesian coordinate system, increasing from left to right and bottom to top.

Value

a data frame.

Examples

```
d <- data.frame(row = c(0, 3222, 6445), col = c(0, 4000, 8000))
tile_coords(d, "galaxy1")
```

---

tlBooks *Star Trek novel-based timeline.*

Description

A data frame with 2122 rows and 14 columns containing Star Trek timeline data. This dataset is novel-driven, meaning that the timeline entries (rows) provide a chronologically ordered list of licensed Star Trek novels.

Usage

```
tlBooks
```

Format

A data frame

Details

Specifically, this curated dataset includes data derived from historical timeline information in the appendix of the Star Trek reference manual, Voyages of the Imagination, which provides information on the large collection of licensed Star Trek literature. The authors note that the original timeline includes "novels, short stories, eBooks, novelizations, Simon & Schuster Audio original audio books, Minstrel Books young adult books, and classic novels from Bantam and Ballantine Books, published through October 2006."

While this data is very informative, it is clearly many years out of date. It is also necessarily speculative. Settings are determined based in part on what is interpreted to be the intention of a given author for a given production. Nevertheless, it still represents possibly the highest quality representation of the chronological ordering of Star Trek fiction that combines episodes and movies with written works. The concurrent timeline of Star Trek TV episodes and movies are interleaved with the novels and other written fiction for fuller context resulting in a much richer timeline. See the `tlEvents` dataset for an event-driven timeline.

See Also

`tlEvents()` `tlFootnotes()`

`tlEvents` *Star Trek event-based timeline.*

Description

A data frame with 1241 rows and 6 columns containing Star Trek timeline data. This dataset is event-driven, meaning that the timeline entries (rows) provide chronologically ordered historical events from the Star Trek universe. See the `tlBooks` dataset for an novel-driven timeline.

Usage

```
tlEvents
```

Format

A data frame

Details

As with `tlBooks`, this timeline is quite out of date. In fact it is at least somewhat more out of date than `tlBooks`. This timeline is also more problematic than the other, and less relevant moving forward. Its updating essentially ceased as the other began.

However, it is included because unlike `tlBooks`, which is a timeline of production titles, this timeline dataset is event-driven. While it may now be erroneous in places even independent from being out of date, it is useful for its informative textual entries referencing historically significant events in Star Trek lore.

See Also

`tlBooks()` `tlFootnotes()`

`tlFootnotes` *Star Trek timeline footnotes.*

Description

A data frame with 605 rows and 3 columns containing footnotes associated by ID with various entries in package timeline datasets, `tlBooks` and `tlEvents`.

Usage

```
tlFootnotes
```

Format

A data frame

See Also

`tlBooks()` `tlEvents()`

Index