Package 'gtfsio'

October 12, 2024

Type Package

Title Read and Write General Transit Feed Specification (GTFS) Files

Version 1.2.0

Description Tools for the development of packages related to General Transit Feed Specification (GTFS) files. Establishes a standard for representing GTFS feeds using R data types. Provides fast and flexible functions to read and write GTFS feeds while sticking to this standard. Defines a basic 'gtfs' class which is meant to be extended by packages that depend on it. And offers utility functions that support checking the structure of GTFS objects.

License MIT + file LICENSE

URL https://r-transit.github.io/gtfsio/, https://github.com/r-transit/gtfsio

BugReports https://github.com/r-transit/gtfsio/issues

Imports data.table, fs, utils, zip, jsonlite

Suggests knitr, rmarkdown, tinytest

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

Collate 'gtfsio_error.R' 'assert_gtfs.R' 'assert_inputs.R' 'checks.R' 'data.R' 'export_gtfs.R' 'get_gtfs_standards.R' 'gtfs_methods.R' 'gtfs_subset.R' 'gtfsio.R' 'import_gtfs.R' 'new_gtfs.R' 'utils.R'

LazyData true

Depends R (>= 3.1.0)

NeedsCompilation no

Author Daniel Herszenhut [aut, cre] (<https://orcid.org/0000-0001-8066-1105>), Flavio Poletti [aut], Mark Padgham [aut], Rafael H. M. Pereira [rev] (<https://orcid.org/0000-0003-2125-7465>),

Tom Buckley [rev],
Ipea - Institute for Applied Economic Research [cph, fnd]

Maintainer Daniel Herszenhut <dhersz@gmail.com>

Repository CRAN

Date/Publication 2024-10-11 23:30:02 UTC

Contents

`assert_gtfs` *GTFS object validator*

Description

Asserts that a GTFS object is valid. Valid objects are those in which:

- Every element is named.
- Every element inherits from `data.frames`.

The exception to the second rule are objects that contain an element named `"."`. In such case, this element is actually composed by a named list of elements who inherit from `data.frames`.

Usage

```
assert_gtfs(x)
```

Arguments

x	A GTFS object.

Value

The same GTFS object passed to `x`.

See Also

Other constructors: new_gtfs()

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

gtfs <- import_gtfs(gtfs_path)
gtfs <- assert_gtfs(gtfs)
```

---

check_field_class *Check the classes of fields in a GTFS object element*

Description

Checks the classes of fields, represented by columns, inside a GTFS object element.

Usage

```
check_field_class(x, file, fields, classes)

assert_field_class(x, file, fields, classes)
```

Arguments

x	A GTFS object.
file	A string. The element, that represents a GTFS text file, whose fields' classes should be checked.
fields	A character vector. The fields to have their classes checked.
classes	A character vector, with the same length of fields. The classes that each field must inherit from.

Value

check_field_class returns TRUE if the check is successful, and FALSE otherwise.
assert_field_class returns x invisibly if the check is successful, and throws an error otherwise.

See Also

Other checking functions: check_field_exists(), check_file_exists()

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

check_field_class(
  gtfs,
  "calendar",
  fields = c("monday", "tuesday"),
  classes = rep("integer", 2)
)

check_field_class(
  gtfs,
  "calendar",
  fields = c("monday", "tuesday"),
  classes = c("integer", "character")
)
```

---

check_field_exists *Check the existence of fields in a GTFS object element*

Description

Checks the existence of fields, represented by columns, inside a GTFS object element.

Usage

```
check_field_exists(x, file, fields)

assert_field_exists(x, file, fields)
```

Arguments

x	A GTFS object.
file	A string. The element, that represents a GTFS text file, where fields should be searched.
fields	A character vector. The fields to check the existence of.

Value

`check_field_exists` returns `TRUE` if the check is successful, and `FALSE` otherwise.
`assert_field_exists` returns `x` invisibly if the check is successful, and throws an error otherwise.

See Also

Other checking functions: `check_field_class()`, `check_file_exists()`

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

check_field_exists(gtfs, "calendar", c("monday", "tuesday"))

check_field_exists(gtfs, "calendar", c("monday", "oi"))
```

---

check_file_exists *Check the existence of text files in a GTFS object*

Description

Checks the existence of elements inside a GTFS object that represent specific GTFS text files.

Usage

```
check_file_exists(x, files)

assert_file_exists(x, files)
```

Arguments

x	A GTFS object.
files	A character vector. The files to check the existence of.

Value

`check_file_exists` returns `TRUE` if the check is successful, and `FALSE` otherwise.
`assert_file_exists` returns `x` invisibly if the check is successful, and throws an error otherwise.

See Also

Other checking functions: `check_field_class()`, `check_field_exists()`

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

check_file_exists(gtfs, c("calendar", "agency"))

check_file_exists(gtfs, c("calendar", "oi"))
```

export_gtfs *Export GTFS objects*

Description

Writes GTFS objects to disk as GTFS transit feeds. The object must be formatted according to the standards for reading and writing GTFS transit feeds, as specified in gtfs_reference (i.e. data types are not checked). If present, does not write auxiliary tables held in a sub-list named ".".

Usage

```
export_gtfs(
  gtfs,
  path,
  files = NULL,
  standard_only = FALSE,
  compression_level = 9,
  as_dir = FALSE,
  overwrite = TRUE,
  quiet = TRUE
)
```

Arguments

gtfs	A GTFS object.
path	A string. Where the resulting .zip file must be written to.
files	A character vector. The name of the elements to be written to the feed.
standard_only	A logical. Whether only standard files and fields should be written (defaults to TRUE, which drops extra files and fields).
compression_level	A numeric, between 1 and 9. The higher the value, the best the compression, which demands more processing time. Defaults to 9 (best compression).
as_dir	A logical. Whether the feed should be exported as a directory, instead of a .zip file. Defaults to FALSE.
overwrite	A logical. Whether to overwrite an existing .zip file (defaults to TRUE).
quiet	A logical. Whether to hide log messages and progress bars (defaults to TRUE).

Value

Invisibly returns the same GTFS object passed to gtfs.

See Also

gtfs_reference

Other io functions: import_gtfs()

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

gtfs <- import_gtfs(gtfs_path)

tmpf <- tempfile(pattern = "gtfs", fileext = ".zip")

export_gtfs(gtfs, tmpf)
zip::zip_list(tmpf)$filename

export_gtfs(gtfs, tmpf, files = c("shapes", "trips"))
zip::zip_list(tmpf)$filename
```

---

get_gtfs_standards *Generate GTFS standards*

Description

The dataset gtfs_reference now contains the standard specifications. This function is deprecated and no longer used in `import_gtfs()` or `export_gtfs()`.

Usage

```
get_gtfs_standards()
```

Details

Generates a list specifying the standards to be used when reading and writing GTFS feeds with R. Each list element (also a list) represents a distinct GTFS table, and describes:

- whether the table is required, optional or conditionally required;
- the fields that compose the table, including which R data type is best suited to represent it, whether the field is required, optional or conditionally required, and which values it can assume (most relevant to GTFS `ENUM`s.

Note: the standards list is based on the specification as revised in May 9th, 2022.

Value

A named list, in which each element represents the R equivalent of each GTFS table standard.

Details

GTFS standards were derived from GTFS Schedule Reference. The R data types chosen to represent each GTFS data type are described below:

- Color = `character`

- Currency amount = `numeric`
- Currency code = `character`
- Date = `integer`
- Email = `character`
- ENUM = `integer`
- ID = `character`
- Integer = `integer`
- Language code = `character`
- Latitude = `numeric`
- Longitude = `numeric`
- Float = `numeric`
- Phone number = `character`
- Text = `character`
- Time = `character`
- Timezone = `character`
- URL = `character`

See Also

gtfs_reference

Examples

```
## Not run:
  gtfs_standards <- get_gtfs_standards()

## End(Not run)
```

`gtfs_reference` *GTFS reference*

Description

The data from the official GTFS specification document parsed to a list. Revision date: `2024-08-16`.

Usage

```
gtfs_reference
```

Format

A list with data for every GTFS file. Each named list element (also a list) has specifications for one GTFS file in the following structure:

- `File_Name`: file name including file extension (txt or geojson)
- `File_Presence`: Presence condition applied to the file
- `file`: file name without file extension
- `file_ext`: file extension
- `fields`: data.frame with parsed field specification (columns: `Field_Name`, `Type`, `Presence`, `Description`, `gtfsio_type`)
- `primary_key`: primary key as vector
- `field_types`: named vector on how GTFS types (values) should be read in gtfsio (names). Values are the same as in `fields`.

Details

GTFS Types are converted to R types in gtfsio according to the following list:

- Array = `geojson_array`
- Color = `character`
- Currency amount = `numeric`
- Currency code = `character`
- Date = `integer`
- Email = `character`
- Enum = `character, integer`
- Float = `numeric`
- ID = `character`
- Integer = `integer`
- Language code = `character`
- Latitude = `numeric`
- Longitude = `numeric`
- Non-negative float = `numeric`
- Non-negative integer = `integer`
- Non-null integer = `integer`
- Non-zero integer = `integer`
- Object = `geojson_object`
- Phone number = `character`
- Positive float = `numeric`
- Positive integer = `integer`
- String = `character`

- Text = character
- Text or URL or Email or Phone number = character
- Time = character
- Timezone = character
- URL = character
- Unique ID = character

Source

https://github.com/google/transit/blob/master/gtfs/spec/en/reference.md

---

`import_gtfs` *Import GTFS transit feeds*

Description

Imports GTFS transit feeds from either a local `.zip` file or an URL. Columns are parsed according to the standards for reading and writing GTFS feeds specified in `gtfs_reference`.

Usage

```
import_gtfs(
  path,
  files = NULL,
  fields = NULL,
  extra_spec = NULL,
  skip = NULL,
  quiet = TRUE,
  encoding = "unknown"
)
```

Arguments

`path`	A string. The path to a GTFS `.zip` file.
`files`	A character vector. The text files to be read from the GTFS, without the `.txt` extension. If `NULL` (the default), all existing text files are read.
`fields`	A named list. The fields to be read from each text file, in the format `list(file1 = c("field1", "field2"))`. If `NULL` (the default), all fields from the files specified in `files` are read. If a file is specified in `files` but not in `fields`, all fields from that file will be read (i.e. you may specify in `fields` only files whose fields you want to subset).
`extra_spec`	A named list. Custom specification used when reading undocumented fields, in the format `list(file1 = c(field1 = "type1", field2 = "type2"))`. If `NULL` (the default), all undocumented fields are read as character. Similarly, if an undocumented field is not specified in `extra_spec`, it is read as character (i.e. you may specify in `extra_spec` only the fields that you want to read as a different type). Only supports the `character`, `integer` and `numeric` types.

skip	A character vector. Text files that should not be read from the GTFS, without the .txt extension. If NULL (the default), no files are skipped. Cannot be used if files is set.
quiet	A logical. Whether to hide log messages and progress bars (defaults to TRUE).
encoding	A string. Passed to fread, defaults to "unknown". Other possible options are "UTF-8" and "Latin-1". Please note that this is not used to re-encode the input, but to enable handling encoded strings in their native encoding.

Value

A GTFS object: a named list of data frames, each one corresponding to a distinct text file from the given GTFS feed.

See Also

gtfs_reference

Other io functions: export_gtfs()

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

# read all files and columns
gtfs <- import_gtfs(gtfs_path)
names(gtfs)
names(gtfs$trips)

# read all columns from selected files
gtfs <- import_gtfs(gtfs_path, files = c("trips", "stops"))
names(gtfs)
names(gtfs$trips)

# read specific columns from selected files
gtfs <- import_gtfs(
  gtfs_path,
  files = c("trips", "stops"),
  fields = list(
    trips = c("route_id", "trip_id"),
    stops = c("stop_id", "stop_lat", "stop_lon")
  )
)
```

---

new_gtfs *GTFS object constructor*

Description

Creates a GTFS object. Mostly useful for package authors who may want to either create `gtfs` objects in their packages or create subclasses of the main `gtfs` class. The usage of this function assumes some knowledge on `gtfs` objects, thus inputs are not extensively checked. See `assert_gtfs` for more thorough checks.

Usage

```
new_gtfs(x, subclass = character(), ...)
```

Arguments

`x`	A GTFS-like object (either a GTFS object or a named list). Each element must represent a distinct GTFS text file.
`subclass`	A character vector. Subclasses to be assigned to the `gtfs` object.
`...`	Name-value pairs. Additional attributes.

Value

A GTFS object: a named list of data frames, each one corresponding to a distinct GTFS text file, with `gtfs` and `list` classes.

See Also

Other constructors: `assert_gtfs()`

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

tmpdir <- tempfile(pattern = "new_gtfs_example")
zip::unzip(gtfs_path, exdir = tmpdir)

agency <- data.table::fread(file.path(tmpdir, "agency.txt"))
stops <- data.table::fread(file.path(tmpdir, "stops.txt"))
routes <- data.table::fread(file.path(tmpdir, "routes.txt"))
trips <- data.table::fread(file.path(tmpdir, "trips.txt"))
stop_times <- data.table::fread(file.path(tmpdir, "stop_times.txt"))
calendar <- data.table::fread(file.path(tmpdir, "calendar.txt"))

txt_files <- list(
  agency = agency,
  stops = stops,
  routes = routes,
  trips = trips,
  stop_times = stop_times,
  calendar = calendar
)

gtfs <- new_gtfs(txt_files)
```

```
class(gtfs)
names(gtfs)
```

`print.gtfs` *Print a GTFS object*

Description

Prints a GTFS object suppressing the `class` attribute.

Usage

```
## S3 method for class 'gtfs'
print(x, ...)
```

Arguments

`x`	A GTFS object.
`...`	Optional arguments ultimately passed to `format`.

Value

The GTFS object that was printed, invisibly.

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

# subset 'gtfs' for a smaller output
gtfs <- gtfs[c("routes", "trips")]

print(gtfs)
```

summary.gtfs *Print summary of a GTFS object*

Description

Print summary of a GTFS object

Usage

```
## S3 method for class 'gtfs'
summary(object, ...)
```

Arguments

object	A GTFS object.
...	Ignored.

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

summary(gtfs)
```

[.gtfs *Subset a GTFS object*

Description

Subsetting a GTFS object using [preserves the object class.

Usage

```
## S3 method for class 'gtfs'
x[value]
```

Arguments

x	A GTFS object.
value	Either a numeric or a character vector. Designates the elements to be returned.

Value

A GTFS object.

Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

gtfs <- import_gtfs(gtfs_path)
names(gtfs)
class(gtfs)

small_gtfs <- gtfs[1:5]
names(small_gtfs)
class(small_gtfs)

small_gtfs <- gtfs[c("shapes", "trips")]
names(small_gtfs)
class(small_gtfs)
```

Index