Package 'tidytransit'

April 16, 2019

Type Package

Title Read, Validate, Analyze, and Map Files in the General Transit Feed Specification

Version 0.4.0

Description Read General Transit Feed Specification (GTFS) zipfiles into a list of R dataframes. Perform validation of the data structure against the specification. Analyze the headways and frequencies at routes and stops. Create maps and perform spatial analysis on the routes and stops. Please see the GTFS documentation here for more detail: <http://gtfs.org/>.

License GPL

LazyData TRUE

Depends R (>= 3.2.5)

Imports dplyr, zip, tibble, readr, data.table, httr, htmltools, magrittr, stringr, assertthat, scales, here, rlang, sf, lubridate, hms, tidyr, tools, digest

Suggests testthat, knitr, rmarkdown, ggplot2

RoxygenNote 6.1.1

URL https://github.com/r-transit/tidytransit

BugReports https://github.com/r-transit/tidytransit

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Tom Buckley [aut, cre],
Flavio Poletti [aut],
Danton Noriega-Goodwin [aut],
Mark Padgham [aut],
Angela Li [ctb],
Elaine McVey [ctb],
Charles Hans Thompson [ctb],
Michael Sumner [ctb],
Patrick Hausmann [ctb],

Bob Rudis [ctb],
Kearey Smith [ctb],
Dave Vautin [ctb],
Kyle Walker [ctb]

Maintainer Tom Buckley <tom@tbuckl.com>

Repository CRAN

Date/Publication 2019-04-16 16:13:09 UTC

R topics documented:

`feedlist` *Dataframe of source GTFS data from Transitfeeds*

Description

A dataset containing a list of URLs for GTFS feeds

Usage

```
feedlist
```

Format

A data frame with 911 rows and 10 variables:

id the id of the feed on transitfeeds.com
t title of the feed
loc_id location id
loc_pid location placeid of the feed on transitfeeds.com
loc_t the title of the location
loc_n the shortname fo the location
loc_lat the location latitude
loc_lng the location longitude
url_d GTFS feed url
url_i the metadata url for the feed

Source

http://www.transitfeeds.com/

---

filter_stops *Get a set of stops for a given set of service ids and route ids*

Description

Get a set of stops for a given set of service ids and route ids

Usage

```
filter_stops(gtfs_obj, service_ids, route_ids)
```

Arguments

gtfs_obj	as read by read_gtfs()
service_ids	the service for which to get stops
route_ids	the route_ids for which to get stops

Value

stops for a given service

Examples

```
local_gtfs_path <- system.file("extdata", "google_transit_nyc_subway.zip", package = "tidytransit")
nyc <- read_gtfs(local_gtfs_path,local=TRUE)
select_service_id <- filter(nyc$calendar, monday==1) %>% pull(service_id)
select_route_id <- sample_n(nyc$routes, 1) %>% pull(route_id)
filtered_stops_df <- filter_stops(nyc, select_service_id, select_route_id)
```

get_feedlist *Get list of all available feeds from transitfeeds API*

Description

Get list of all available feeds from transitfeeds API

Usage

```
get_feedlist()
```

Value

a data frame with the gtfs feeds on transitfeeds.

See Also

feedlist_df

Examples

```
feedlist_df <- get_feedlist()
```

get_route_frequency *Get Route Frequency*

Description

Note that some GTFS feeds contain a frequency data frame already. Consider using this instead, as it will be more accurate than what tidytransit calculates.

Usage

```
get_route_frequency(gtfs_obj, start_hour = 6, end_hour = 22,
  quiet = FALSE, service_ids = c(), dow = c(1, 1, 1, 1, 1, 0, 0))
```

Arguments

gtfs_obj	a list of gtfs dataframes as read by the trread package.
start_hour	(optional) an integer, default 6 (6 am)
end_hour	(optional) an integer, default 22 (10 pm)
quiet	default FALSE. whether to echo process messages
service_ids	(optional) a string from the calendar dataframe identifying a particular service schedule.
dow	(optional) an integer vector with days of week. monday=1. default: c(1,1,1,1,1,0,0)

Details

should take:

Value

a gtfs_obj with a dataframe of routes with variables (gtfs_obj$.$routes_frequency) for headway/frequency for a route within a given time frame

Examples

```
data(gtfs_obj)
gtfs_obj <- get_route_frequency(gtfs_obj)
x <- order(gtfs_obj$.$routes_frequency$median_headways)
head(gtfs_obj$.$routes_frequency[x,])
```

---

get_route_geometry *Make Routes into Simple Features Lines*

Description

Make Routes into Simple Features Lines

Usage

```
get_route_geometry(gtfs_obj, route_ids = NULL, service_ids = NULL)
```

Arguments

gtfs_obj	tidytransit gtfs object
route_ids	select routes to convert to simple features
service_ids	select service_ids to convert to simple features

Value

an sf dataframe for gtfs routes with a multilinestring column

Examples

```
data(gtfs_obj)
routes_sf <- get_route_geometry(gtfs_obj)
plot(routes_sf[1,])
```

`get_stop_frequency` *Get Stop Frequency*

Description

Note that some GTFS feeds contain a frequency data frame already. Consider using this instead, as it will be more accurate than what tidytransit calculates.

Usage

```
get_stop_frequency(gtfs_obj, start_hour = 6, end_hour = 22,
  service_ids = c(), dow = c(1, 1, 1, 1, 1, 0, 0), by_route = TRUE,
  wide = FALSE)
```

Arguments

`gtfs_obj`	a list of gtfs dataframes as read by read_gtfs().
`start_hour`	(optional) an integer indicating the start hour (default 7)
`end_hour`	(optional) an integer indicating the end hour (default 20)
`service_ids`	(optional) a set of service_ids from the calendar dataframe identifying a particular service id
`dow`	(optional) integer vector indicating which days of week to calculate for. default is weekday, e.g. c(1,1,1,1,1,0,0)
`by_route`	default TRUE, if FALSE then calculate headway for any line coming through the stop in the same direction on the same schedule.
`wide`	(optional) if true, then return a wide rather than tidy data frame

Value

a gtfs_obj with a dataframe of stops (gtfs_obj$.$stops_frequency) with a "Trips" variable representing the count trips taken through each stop for a route within a given time frame

Examples

```
data(gtfs_obj)
gtfs_obj <- get_stop_frequency(gtfs_obj)
x <- order(gtfs_obj$.$stops_frequency$headway)
head(gtfs_obj$.$stops_frequency_df[x,])
```

get_stop_geometry *Make Stops into Simple Features Points*

Description

Make Stops into Simple Features Points

Usage

```
get_stop_geometry(stops)
```

Arguments

stops	a gtfs$stops dataframe

Value

an sf dataframe for gtfs routes with a point column

Examples

```
data(gtfs_obj)
some_stops <- gtfs_obj$stops[sample(nrow(gtfs_obj$stops), 40),]
some_stops_sf <- get_stop_geometry(some_stops)
plot(some_stops_sf)
```

gtfs_as_sf *Add Simple Features for Stops and Routes to GTFS Object*

Description

Add Simple Features for Stops and Routes to GTFS Object

Usage

```
gtfs_as_sf(gtfs_obj, quiet = TRUE)
```

Arguments

gtfs_obj	a standard tidytransit gtfs object
quiet	boolean whether to print status messages

Value

gtfs_obj a tidytransit gtfs object with a bunch of simple features tables

gtfs_obj *Example GTFS data*

Description

Data obtained from http://data.trilliumtransit.com/gtfs/duke-nc-us/duke-nc-us.zip.

Usage

```
gtfs_obj
```

Format

An object of class `gtfs` of length 22.

See Also

read_gtfs

import_gtfs *This function is deprecated. Please use read_gtfs*

Description

This function reads GTFS text files from a local or remote zip file. It also validates the files against the GTFS specification by file, requirement status, and column name The data are returned as a list of dataframes and a validation object, which contains details on whether all required files were found, and which required and optional columns are present.

Usage

```
import_gtfs(path, local = FALSE, quiet = FALSE)
```

Arguments

`path`	Character. url link to zip file OR path to local zip file. if to local path, then option `local` must be set to TRUE.
`local`	Boolean. If the paths are searching locally or not. Default is FALSE (that is, urls).
`quiet`	Boolean. Whether to see file download progress and files extract. FALSE by default.

Value

Dataframes of GTFS data.

Examples

```
library(dplyr)
u1 <- "https://github.com/r-transit/tidytransit/raw/master/inst/extdata/sample-feed-fixed.zip"
sample_gtfs <- import_gtfs(u1)
attach(sample_gtfs)
#list routes by the number of stops they have
routes %>% inner_join(trips, by="route_id") %>%
  inner_join(stop_times) %>%
    inner_join(stops, by="stop_id") %>%
      group_by(route_long_name) %>%
        summarise(stop_count=n_distinct(stop_id)) %>%
          arrange(desc(stop_count))
```

`plot.gtfs` *Plot GTFS object routes and their frequencies*

Description

Plot GTFS object routes and their frequencies

Usage

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

Arguments

`x`	a gtfs_obj as read by read_gtfs()
`...`	further specifications

Examples

```
local_gtfs_path <- system.file("extdata",
                               "google_transit_nyc_subway.zip",
                               package = "tidytransit")
nyc <- read_gtfs(local_gtfs_path,
                 local=TRUE)
plot(nyc)
```

read_gtfs *Get and validate dataframes of General Transit Feed Specification (GTFS) data.*

Description

This function reads GTFS text files from a local or remote zip file. It also validates the files against the GTFS specification by file, requirement status, and column name The data are returned as a list of dataframes and a validation object, which contains details on whether all required files were found, and which required and optional columns are present.

Usage

```
read_gtfs(path, local = FALSE, quiet = TRUE, geometry = FALSE,
  frequency = FALSE)
```

Arguments

path	Character. url link to zip file OR path to local zip file. if to local path, then option `local` must be set to TRUE.
local	Boolean. If the paths are searching locally or not. Default is FALSE (that is, urls).
quiet	Boolean. Whether to see file download progress and files extract. FALSE by default.
geometry	Boolean. Whether to add simple feature dataframes of routes and stops to the gtfs object
frequency	Boolean. Whether to add frequency/headway calculations to the gtfs object

Value

A GTFS object. That is, a list of dataframes of GTFS data.

Examples

```
library(dplyr)
u1 <- "https://github.com/r-transit/tidytransit/raw/master/inst/extdata/sample-feed-fixed.zip"
sample_gtfs <- read_gtfs(u1)
attach(sample_gtfs)
#list routes by the number of stops they have
routes %>% inner_join(trips, by="route_id") %>%
  inner_join(stop_times) %>%
    inner_join(stops, by="stop_id") %>%
      group_by(route_long_name) %>%
        summarise(stop_count=n_distinct(stop_id)) %>%
          arrange(desc(stop_count))
```

routes_df_as_sf *This function is deprecated. Please use get_route_geometry Make Routes into Simple Features Lines*

Description

This function is deprecated. Please use get_route_geometry Make Routes into Simple Features Lines

Usage

```
routes_df_as_sf(gtfs_obj, route_ids = NULL, service_ids = NULL)
```

Arguments

gtfs_obj	gtfs object
route_ids	select routes to convert to simple features
service_ids	select service_ids to convert to simple features

Value

an sf dataframe for gtfs routes with a multilinestring column

Examples

```
data(gtfs_obj)
routes_sf <- get_route_geometry(gtfs_obj)
plot(routes_sf[1,])
```

route_type_names *Dataframe of route type id's and the names of the types (e.g. "Cable Car")*

Description

Dataframe of route type id's and the names of the types (e.g. "Cable Car")

Usage

```
route_type_names
```

Format

A data frame with 122 rows and 2 variables:

id the id of route type

name name of the gtfs route type

Source

https://gist.github.com/derhuerst/b0243339e22c310bee2386388151e11e

---

`set_api_key` *Set TransitFeeds API key for recall*

Description

Set TransitFeeds API key for recall

Usage

```
set_api_key()
```

`set_date_service_table` *Returns all possible date/service_id combinations as a data frame*

Description

Use it to summarise service. For example, get a count of the number of services for a date. See example.

Usage

```
set_date_service_table(gtfs_obj)
```

Arguments

`gtfs_obj`	a gtfs_object as read by read_gtfs

Value

a date_service data frame

Examples

```
library(dplyr)
local_gtfs_path <- system.file("extdata", "google_transit_nyc_subway.zip", package = "tidytransit")
nyc <- read_gtfs(local_gtfs_path, local=TRUE) %>% set_date_service_table()
nyc_services_by_date <- nyc$.$date_service_table
# count the number of services running on each date
nyc_services_by_date %>% group_by(date) %>% count()
```

set_hms_times *Add hms::hms columns to feed*

Description

Adds columns to stop_times (arrival_time_hms, departure_time_hms) and frequencies (start_time_hms, end_time_hms) with times converted with hms::hms().

Usage

```
set_hms_times(gtfs_obj)
```

Arguments

gtfs_obj	a gtfs object in which hms times should be set, the modified gtfs_obj is returned

Value

gtfs_obj with added hms times columns for stop_times and frequencies

set_trippattern *Add trip pattern data frame to the gtfs object*

Description

Add trip pattern data frame to the gtfs object

Usage

```
set_trippattern(gtfs_obj, id_prefix = "t_", hash_length = 7,
  hash_algo = "md5")
```

Arguments

gtfs_obj	gtfs feed
id_prefix	all ids start with this string
hash_length	length the hash should be cut to with substr(). Use -1 if the full hash should be used
hash_algo	hashing algorithm used by digest

Value

gtfs_obj

stops_df_as_sf *This function is deprecated. Please use get_stop_geometry Make Stops into Simple Features Points*

Description

This function is deprecated. Please use get_stop_geometry Make Stops into Simple Features Points

Usage

```
stops_df_as_sf(stops)
```

Arguments

stops	a gtfs$stops dataframe

Value

an sf dataframe for gtfs routes with a point column

Examples

```
data(gtfs_obj)
some_stops <- gtfs_obj$stops[sample(nrow(gtfs_obj$stops), 40),]
some_stops_sf <- get_stop_geometry(some_stops)
plot(some_stops_sf)
```

Index