Package 'gtfsrealtime'

June 12, 2026

Title Read GTFS-Realtime Files into Data Frames

Version 0.2.1

Description
GTFS-realtime is a format transit agencies use to provide current vehicle positions, predicted arrival times, and service alerts. This package provides efficient functions to read this format into data frames. It can be used to retrieve current data or to process archived data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Config/rextendr/version 0.4.2

SystemRequirements Cargo (Rust's package manager), rustc >= 1.88.0, xz

Depends R (>= 4.2)

Imports cli, sf

Suggests knitr, rmarkdown, ggplot2, data.table, purrr, gifski, lubridate, dplyr, tidyr, testthat (>= 3.3.0), jsonlite, tidyselect, tibble, stringr

BugReports https://github.com/mattwigway/gtfsrealtime-r/issues

VignetteBuilder knitr

URL http://projects.indicatrix.org/gtfsrealtime-r/

Config/testthat/edition 3

NeedsCompilation yes

Author Matthew Bhagat-Conway [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1210-2982>), Matthew Palm [ctb, cph] (ORCID: <https://orcid.org/0000-0002-8800-2777>), The GTFS-realtime authors and the authors of bundled Rust crates [ctb, cph] (See inst/AUTHORS.md for information on bundled dependencies and their licensing and copyright status.)

Maintainer Matthew Bhagat-Conway <mwbc@unc.edu>

Repository CRAN

Date/Publication 2026-06-12 19:30:08 UTC

Contents

`read_gtfsrt_alerts` *Read GTFS-realtime alerts*

Description

This function reads GTFS realtime alerts. Alerts are hierarchical: a single alert can have multiple applicability periods, multiple affected entities, and translations to multiple languages. This function flattens all of that to a tabular format, with one row for every combination of applicability, entity, and language. All rows from a single alert can be identified through a common `id`. This is true even if there are duplicate IDs in the feed; they will be deduplicated by adding _duplicated_1, _duplicated_2, etc.

Usage

```
read_gtfsrt_alerts(filename, timezone, label_values = TRUE)
```

Arguments

`filename`	filename to read. Can be uncompressed or compressed with gzip or bzip2, or can be a ZIP file containing multiple feeds (e.g., all alerts for a whole day). Can also be an http:// or https:// URL (ZIP files are not supported when reading from a URL, but gzip and bzip2 are).
`timezone`	timezone of feed, in Olson format. Times in GTFS-realtime are stored as Unix time in UTC; this option will convert to local times. If you want to read times in UTC, specify "Etc/UTC".
`label_values`	should enum types in GTFS-realtime (i.e. categorical variables) be converted to factors with their English labels. If false, they will be left as numeric codes. Default true.

Details

Alerts are intended to capture widespread or long-term disruptions or changes. Trip updates (see `read_gtfsrt_trip_updates()`) are better suited to providing information about day-to-day delays and cancellations to specific trips.

Typically, GTFS-realtime feeds will contain only a single type of entity, but if there are multiple types of entities in a single feed, this function will read only the alerts.

Value

A data frame with one row for every combination of alert, applicability period, affected entity, and language. One alert potentially becomes many rows. The data frame has the following columns. Many of these descriptions come verbatim or nearly so from the GTFS-realtime specification. Path refers to where in the original GTFS-realtime Alert data structure each column comes from.

- `id`: GTFS-realtime entity ID. Since one alert can become multiple rows (for example, for different languages, or different informed entities), the ID can be used to identify rows that came from the same Alert.

  IDs are required by the specification to be unique within a GTFS-realtime file, but sometimes are not. If there are non-unique IDs in the feed, they will be made unique when data are loaded by appending `_duplicated_1`, `_duplicated_2`, and so on and a warning will be issued, which guarantees that all rows from a single file have unique IDs. When working with archived data, there will quite likely be duplicated IDs between files archived at different times (path: `id` property of `FeedEntity` containing this `Alert`).
- `start`: Time when the alert should first be shown. If missing, the alert should be shown as long as it appears in the feed. Converted to local time based on the `timezone` argument. One alert may have multiple start and end times, in which case it will be presented in multiple rows. (path: `active_period.start`)
- `end`: Time when the alert should last be shown. If missing, the alert should be shown as long as it appears in the feed. (path: `active_period.end`).
- The next few colums represent the informed entity of the alert (i.e. the agency, route, etc. for which the alert should be shown). If there are multiple informed entities, the alert will be duplicated for each one.
  - `agency_id`: The `agency_id` from the GTFS static feed that this alert refers to. (path: `informed_entity.agency_id`)
  - `route_id`: The `route_id`from the GTFS that this alert refers to. If `direction_id`is provided, `route_id` must also be provided. (path: `informed_entity.route_id`)
  - `route_type`: The `route_type` from the static GTFS that this alert refers to. (path: `informed_entity.route_type`)
  - `direction_id`: The `direction_id`from the static GTFS feed `trips.txt` file, used to select all trips in one direction for a route, specified by route_id. If `direction_id` is provided, route_id must also be provided. Caution: this field is still experimental, and subject to change. It may be formally adopted in the future. (path: `informed_entity.direction_id`).
  - `trip_trip_id`: The `trip_id` from the GTFS feed that this selector refers to. For non frequency-based trips (trips not defined in GTFS `frequencies.txt`), this field is enough to uniquely identify the trip. For frequency-based trips defined in GTFS `frequencies.txt`, `trip_id`, `start_time`, and `start_date` are all required. For scheduled-based trips (trips not defined in GTFS frequencies.txt), trip_id can only be omitted if the trip can be uniquely identified by a combination of `route_id`, `direction_id`, `start_time`, and `start_date`, and all those fields are provided. (path: `informed_entity.trip.trip_id`)
  - `trip_route_id`: The route_id from the GTFS that this trip refers to. If `trip_id` is omitted, `route_id`, `direction_id`, `start_time`, and `schedule_relationship=SCHEDULED` must all be set to identify a trip instance. `trip_route_id` should not be used to specify a route-wide alert that affects all trips for a route, and generally will not be - `route_id` should be used instead. (path: `informed_entity.trip.route_id`)

- trip_direction_id: The direction_idfrom the GTFS feed trips.txt file, indicating the direction of travel for trips this selector refers to. If trip_id is omitted, direction_id must be provided. Caution: this field is still experimental, and subject to change. It may be formally adopted in the future. (path: informed_entity.trip.route_id)
- trip_start_time: The initially scheduled start time of this trip instance. When the trip_id corresponds to a non-frequency-based trip, this field should either be omitted or be equal to the value in the GTFS feed. When the trip_id correponds to a frequency-based trip defined in GTFS frequencies.txt, start_time is required and must be specified for trip updates and vehicle positions. If the trip corresponds to exact_times=1 GTFS record, then start_time must be some multiple (including zero) of headway_secs later than frequencies.txt start_time for the corresponding time period. If the trip corresponds to exact_times=0, then its start_time may be arbitrary, and is initially expected to be the first departure of the trip. Once established, the start_time of this frequency-based exact_times=0 trip should be considered immutable, even if the first departure time changes – that time change may instead be reflected in a StopTimeUpdate in a TripUpdate. Format and semantics of the field is same as that of GTFS/frequencies.txt/start_time, e.g., 11:15:35 for 11:15:35 AM or 25:15:35 for 1:15:35 AM the day after the service day. (path: informed_entity.trip.start_time)
- trip_start_date: The start date of this trip instance in YYYYMMDD format. For scheduled trips (trips not defined in GTFS frequencies.txt), this field must be provided to disambiguate trips that are so late as to collide with a scheduled trip on a next day. For example, for a train that departs 8:00 and 20:00 every day, and is 12 hours late, there would be two distinct trips on the same time. This field can be provided but is not mandatory for schedules in which such collisions are impossible - for example, a service running on hourly schedule where a vehicle that is one hour late is not considered to be related to schedule anymore. This field is required for frequency-based trips defined in GTFS frequencies.txt. (path: informed_entity.trip.start_date)
- trip_schedule_relationship: The relation between this trip and the static schedule. This is not supposed to be used with alerts, but is provided for completeness. (path: informed_entity.trip.schedule_relationship). Possible values:
  - SCHEDULED: Trip that is running in accordance with its GTFS schedule, or is close enough to the scheduled trip to be associated with it.
  - ADDED: This value has been deprecated as the behavior was unspecified. Use DUPLICATED for an extra trip that is the same as a scheduled trip except the start date or time, or NEW for an extra trip that is unrelated to an existing trip.
  - UNSCHEDULED: A trip that is running with no schedule associated to it (GTFS frequencies.txt exact_times=0). Trips with trip_schedule_relationship=UNSCHEDULED must also set all stop_schedule_relationship=UNSCHEDULED.
  - CANCELED: A trip that existed in the schedule but was removed.
  - REPLACEMENT: A trip that replaces an existing trip in the schedule. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
  - DUPLICATED: An extra trip that was added in addition to a running schedule, for example, to replace a broken vehicle or to respond to sudden passenger load. Used with trip_id, start_date, and start_time to copy an existing trip from static GTFS but start at a different service date and/or time. Duplicating a trip is allowed if the service related to the original trip in (CSV) GTFS (in calendar.txt or calendar_dates.txt) is operating within the next 30 days. The trip to be duplicated is identified via trip_id.

This enumeration does not modify the existing trip referenced by `trip_id` - if a producer wants to cancel the original trip, it must publish a separate TripUpdate with the value of `CANCELED` or `DELETED`. If a producer wants to replace the original trip, a value of `REPLACEMENT` should be used instead.
Trips defined in GTFS `frequencies.txt` with `exact_times` that is empty or equal to 0 cannot be duplicated.
Existing producers and consumers that were using the ADDED enumeration to represent duplicated trips must follow the migration guide to transition to the `DUPLICATED` enumeration. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.

    - `DELETED`: A trip that existed in the schedule but was removed and must not be shown to users. `DELETED` should be used instead of `CANCELED` to indicate that a transit provider would like to entirely remove information about the corresponding trip from consuming applications, so the trip is not shown as cancelled to riders, e.g. a trip that is entirely being replaced by another trip. This designation becomes particularly important if several trips are cancelled and replaced with substitute service. If consumers were to show explicit information about the cancellations it would distract from the more important real-time predictions. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
    - `NEW`: An extra trip unrelated to any existing trips, for example, to respond to sudden passenger load. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
  - `trip_modification_id`: Linkage to any modifications done to this trip (shape changes, removal or addition of stops). Reading trip modifications themselves is not yet supported (see #21). If you have a feed with trip modifications, please comment on that issue so we are aware they exist in the wild. (path: `informed_entity.trip.modification_id`)
  - `stop_id`: The `stop_id` from the GTFS feed that this alert refers to.
- `cause`: The cause of the disruption. (path: `cause`) Possible values:
  - `UNKNOWN_CAUSE`
  - `OTHER_CAUSE`
  - `TECHNICAL_PROBLEM`
  - `STRIKE`
  - `DEMONSTRATION`
  - `ACCIDENT`
  - `HOLIDAY`
  - `WEATHER`
  - `MAINTENANCE`
  - `CONSTRUCTION`
  - `POLICE_ACTIVITY`
  - ‘MEDICAL_EMERGENCY“
- `effect`: The effect of the disruption (path: `effect`). Possible values:
  - `NO_SERVICE`
  - `REDUCED_SERVICE`
  - `SIGNIFICANT_DELAYS`
  - `DETOUR`

  - ADDITIONAL_SERVICE
  - MODIFIED_SERVICE
  - OTHER_EFFECT
  - UNKNOWN_EFFECT
  - STOP_MOVED
  - NO_EFFECT
  - ACCESSIBILITY_ISSUE
- language: The free-text fields in alerts can be presented in multiple languages. If they are, rows associated with the alert will be duplicated for each language; this column will contain the language identifier (e.g. "EN", "ES"), and the remaining fields will contain the translated alerts in that language. Not every feed will have multiple languages (or lange flags at all), and it is also possible that some fields are translated into a particular language and others are left NA.
- cause_detail: Description of the cause of the alert that allows for agency-specific language; more specific than the cause. Caution: this field is still experimental, and subject to change. It may be formally adopted in the future.(path: cause_detail)
- effect_detail: Description of the effect of the alert that allows for agency-specific language; more specific than the Effect. Caution: this field is still experimental, and subject to change. It may be formally adopted in the future. (path: effect_detail)
- url: The URL which provides additional information about the alert. May differ for different languages. (path: url).
- header_text: Header (i.e. title) for the alert (path: header)
- description_text: Description for the alert. The information in the description should add to the information of the header. (path: description)
- tts_header_text: Text containing the alert's header to be used for text-to-speech implementations. This field is the text-to-speech version of header_text. It should contain the same information as header_text but formatted such that it can read as text-to-speech (for example, abbreviations removed, numbers spelled out, etc.) (path: tts_header_text)
- tts_description_text: Text containing a description for the alert to be used for text-to-speech implementations. This field is the text-to-speech version of description_text. It should contain the same information as description_textbut formatted such that it can be read as text-to-speech (for example, abbreviations removed, numbers spelled out, etc.) (path: tts_description_text)
- severity_level: Severity of the alert. (path: severity_level). Possible values:
  - UNKNOWN_SEVERITY
  - INFO
  - WARNING
  - SEVERE
- file_timestamp: Timestamp of the GTFS-realtime file itself (i.e. when the file was generated, not when the updates were generated)
- file_index: When reading a ZIP file, a one-based index of which file each observation came from Note that it is in the the order the files appeared in the ZIP file, which may not be chronological.

Examples

```
# This will read an alerts feed included with gtfsrealtime. Replace with
# the path to your own file if desired.
file = system.file("nyc-service-alerts.pb.bz2", package = "gtfsrealtime")

# Need to specify timezone so timestamps will be in local time.
read_gtfsrt_alerts(file, "America/New_York")
```

`read_gtfsrt_positions` *Read GTFS-realtime vehicle positions into a data frame.*

Description

Reads vehicle position data from a GTFS-realtime feed into a data.frame with one row per position update. GTFS-realtime timestamps are represented in Unix time; a timezone must be specified to convert these to time-zone-aware R date-time objects (i.e. POSIXct).

Usage

```
read_gtfsrt_positions(filename, timezone, as_sf = FALSE, label_values = TRUE)
```

Arguments

`filename`	filename to read. Can be uncompressed or compressed with gzip or bzip2, or can be a ZIP file containing multiple feeds (e.g., all positions for a whole day). Can also be an http:// or https:// URL (ZIP files are not supported when reading from a URL, but gzip and bzip2 are).
`timezone`	timezone of feed, in Olson format. Times in GTFS-realtime are stored as Unix time in UTC; this option will convert to local times. If you want to read times in UTC, specify "Etc/UTC"
`as_sf`	return an sf (spatial) object rather than a data frame.
`label_values`	should enum types in GTFS-realtime (i.e. categorical variables) be converted to factors with their English labels. If false, they will be left as numeric codes? Default TRUE

Details

Typically, GTFS-realtime feeds will contain only a single type of entity, but if there are multiple types of entities in a single feed, this function will read only the vehicle positions. Each vehicle position will become a single row in the data frame.

Value

Data frame containing vehicle position data. The data frame will have the following columns. Note that most of these columns can contain NAs (and in most feeds, many will be entirely NA). GTFS-realtime is a hierarchical format that is converted to a flat format for use in R; the paths refer to where each column comes from within the GTFS-realtime VehiclePosition data structure. Each column is reported below with its definition, many of which come verbatim from the GTFS-realtime specification

- `id`: GTFS-realtime entity ID. These are required by the specification to be unique within a GTFS-realtime file, but sometimes are not. If there are non-unique IDs in the feed, they will be made unique when data are loaded by appending `_duplicate_1`, `_duplicate_2`, and so on and a warning will be issued, which guarantees that all rows from a single file have unique IDs. When working with archived data, there will quite likely be duplicated IDs between files archived at different times (path: `id` property of `FeedEntity` containing this `VehiclePosition`).
- `latitude`: reported latitude of vehicle (path: `position.latitude`).
- `longitude`: reported latitude of vehicle (path: `position.latitude`).
- `bearing`: current bearing (compass heading) of vehicle, in degrees (path: `position.bearing`).
- `odometer`: the distance the vehicle is traveled, according to the specification should be in meters (path: `position.odometer`).
- `speed`: current speed of the vehicle, in meters per second (path: `position.speed`).
- `trip_id`: Trip ID the vehicle is currently serving (path: `trip.trip_id`)
- `route_id`: Route ID the vehicle is currently serving (path `trip.route_id`)
- `direction_id`: GTFS direction ID (0 or 1) of the trip the vehicle is currently serving (path: `trip.direction_id`)
- `start_time`: the time this trip started, needed to differentiate trips in frequency based GTFS. (path: `trip.start_time`)
- `start_date`: the time this trip started, needed to differentiate trips in frequency based GTFS or when the service day is otherwise not clear. (path: `trips.start_date`)
- `schedule_relationship`: How is this trip related to the schedule in the static GTFS? (path: `trip.schedule_relationship`). Possible values:
  - `SCHEDULED`: Trip that is running in accordance with its GTFS schedule, or is close enough to the scheduled trip to be associated with it.
  - `ADDED`: This value has been deprecated as the behavior was unspecified. Use `DUPLICATED` for an extra trip that is the same as a scheduled trip except the start date or time, or `NEW` for an extra trip that is unrelated to an existing trip.
  - `UNSCHEDULED`: A trip that is running with no schedule associated to it (GTFS `frequencies.txt` `exact_times=0`). Trips with `trip_schedule_relationship=UNSCHEDULED` must also set all `stop_schedule_relationship=UNSCHEDULED`.
  - `CANCELED`: A trip that existed in the schedule but was removed.
  - `REPLACEMENT`: A trip that replaces an existing trip in the schedule. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.

  - DUPLICATED: An extra trip that was added in addition to a running schedule, for example, to replace a broken vehicle or to respond to sudden passenger load. Used with trip_id, start_date, and start_time to copy an existing trip from static GTFS but start at a different service date and/or time. Duplicating a trip is allowed if the service related to the original trip in (CSV) GTFS (in calendar.txt or calendar_dates.txt) is operating within the next 30 days. The trip to be duplicated is identified via trip_id. This enumeration does not modify the existing trip referenced by trip_id - if a producer wants to cancel the original trip, it must publish a separate TripUpdate with the value of CANCELED or DELETED. If a producer wants to replace the original trip, a value of REPLACEMENT should be used instead.
    Trips defined in GTFS frequencies.txt with exact_times that is empty or equal to 0 cannot be duplicated.
    Existing producers and consumers that were using the ADDED enumeration to represent duplicated trips must follow the migration guide to transition to the DUPLICATED enumeration. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
  - DELETED: A trip that existed in the schedule but was removed and must not be shown to users. DELETED should be used instead of CANCELED to indicate that a transit provider would like to entirely remove information about the corresponding trip from consuming applications, so the trip is not shown as cancelled to riders, e.g. a trip that is entirely being replaced by another trip. This designation becomes particularly important if several trips are cancelled and replaced with substitute service. If consumers were to show explicit information about the cancellations it would distract from the more important real-time predictions. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
  - NEW: An extra trip unrelated to any existing trips, for example, to respond to sudden passenger load. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.

- stop_id: Stop ID in static GTFS the vehicle is at/approaching. See current_status. (path: stop_id)
- current_stop_sequence: Stop sequence in static GTFS the vehicle is at/approaching (used to disambiguate trips that serve the same stop twice). See current_status. (path: current_stop_sequence)
- current_status: Current status of the vehicle in relation to the stop identified by stop_id/current_stop_sequence. (path: current_status). Potential values:
  - "INCOMING_AT": The vehicle is approaching this stop.
  - "STOPPED_AT": The vehicle is stopped at this stop.
  - "IN_TRANSIT_TO": The GTFS-realtime specification says this means "The vehicle has departed and is in transit to the next stop." which would suggest that it has *departed* the stop identified by stop_id. However, the value itself suggests the vehicle has departed the *previous* stop. Auditing this across a number of GTFS feeds is planned.
- timestamp: Moment at which the vehicle's position was measured. In the GTFS-realtime data this is recorded as second since January 1, 1970, UTC (i.e. Unix time), but when read by this function it will be converted to local time in the timezone specified by the function call. (path: timestamp).
- congestion_level: Congestion level the vehicle is experiencing. Possible values are UNKNOWN_CONGESTION_LEVEL, RUNNING_SMOOTHLY, STOP_AND_GO, CONGES-

TION, SEVERE_CONGESTION, with SEVERE_CONGESTION flagged as meaning "People leaving their cars."

- occupancy_status: How full the vehicle is. (path: occupancy_status). Possible values (with official descriptions from the GTFS-realtime spec).:
  - "EMPTY": The vehicle or carriage is considered empty by most measures, and has few or no passengers onboard, but is still accepting passengers.
  - "MANY_SEATS_AVAILABLE": The vehicle or carriage has a large number of seats available. The amount of free seats out of the total seats available to be considered large enough to fall into this category is determined at the discretion of the producer.
  - "FEW_SEATS_AVAILABLE": The vehicle or carriage has a relatively small number of seats available. The amount of free seats out of the total seats available to be considered small enough to fall into this category is determined at the discretion of the feed producer.
  - "STANDING_ROOM_ONLY": The vehicle or carriage can currently accommodate only standing passengers.
  - "CRUSHED_STANDING_ROOM_ONLY": The vehicle or carriage can currently accommodate only standing passengers and has limited space for them.
  - "FULL": The vehicle or carriage is considered full by most measures, but may still be allowing passengers to board.
  - "NOT_ACCEPTING_PASSENGERS": The vehicle or carriage is not accepting passengers, but usually accepts passengers for boarding.
  - "NO_DATA_AVAILABLE": The vehicle or carriage doesn't have any occupancy data available at that time.
  - "NOT_BOARDABLE": The vehicle or carriage is not boardable and never accepts passengers. Useful for special vehicles or carriages (engine, maintenance carriage, etc...).
- occupancy_percentage: A percentage value indicating the degree of passenger occupancy in the vehicle. The values are represented as an integer without decimals. 0 means 0% and 100 means 100%. The value 100 should represent the total maximum occupancy the vehicle was designed for, including both seated and standing capacity, and current operating regulations allow. The value may exceed 100 if there are more passengers than the maximum designed capacity. The precision of occupancy_percentage should be low enough that individual passengers cannot be tracked boarding or alighting the vehicle. If multi_carriage_status is populated with per-carriage occupancy_percentage, then this field should describe the entire vehicle with all carriages accepting passengers considered. This field is still experimental, and subject to change. It may be formally adopted in the future. (path: occupancy_percentage)
- vehicle_id: Internal system identification of the vehicle. Should be unique per vehicle, and is used for tracking the vehicle as it proceeds through the system. This id should not be made visible to the end-user; for that purpose use the label field. (path: vehicle.id)
- vehicle_label: User visible label, i.e., something that must be shown to the passenger to help identify the correct vehicle. (path: vehicle.label)
- vehicle_license_plate: The license plate of the vehicle. (path: vehicle.license_plate)
- vehicle_wheelchair_accessible: Whether the vehicle is wheelchair accessible. If provided, can overwrite the wheelchair_accessible value from the static GTFS. (path: vehicle.wheelchair_accessible) Possible values:
  - NO_VALUE The trip doesn't have information about wheelchair accessibility. This is the default behavior. If the static GTFS contains a wheelchair_accessible value, it won't be overwritten.

  - UNKNOWN The trip has no accessibility value present. This value will overwrite the value from the GTFS.
  - WHEELCHAIR_ACCESSIBLE The trip is wheelchair accessible. This value will overwrite the value from the GTFS.
  - WHEELCHAIR_INACCESSIBLEThe trip is not wheelchair accessible. This value will overwrite the value from the GTFS.

- `file_timestamp`: Timestamp of the GTFS-realtime file itself (i.e. when the file was generated, not when the updates were generated)
- `file_index`: When reading a ZIP file, a one-based index of which file each observation came from Note that it is in the the order the files appeared in the ZIP file, which may not be chronological.

Examples

```
# This will read a positions feed included with gtfsrealtime. Replace with
# the path to your own file if desired.
file = system.file("nyc-vehicle-positions.pb.bz2", package = "gtfsrealtime")

# Need to specify timezone so timestamps will be in local time.
read_gtfsrt_positions(file, "America/New_York")
```

`read_gtfsrt_trip_updates`

Read GTFS-realtime trip updates

Description

This reads GTFS-realtime trip updates into a data frame, and flattens the hierarchical structure.

Usage

```
read_gtfsrt_trip_updates(filename, timezone, label_values = TRUE)
```

Arguments

`filename`	filename to read. Can be uncompressed or compressed with gzip or bzip2, or can be a ZIP file containing multiple feeds (e.g., all trip updates for a whole day). Can also be an http:// or https:// URL (ZIP files are not supported when reading from a URL, but gzip and bzip2 are).
`timezone`	timezone of feed, in Olson format. Times in GTFS-realtime are stored as Unix time in UTC; this option will convert to local times. If you want to read times in UTC, specify "Etc/UTC"
`label_values`	should enum types in GTFS-realtime (i.e. categorical variables) be converted to factors with their English labels. If false, they will be left as numeric codes. Default true.

Details

Trip updates can contain many stop time updates for different stops along a trip. These will be flattened to one row per stop time update, with fields of the trip update that apply to all stop time updates duplicated. You can figure out which rows came from a single trip update by looking at for common values in the `id` column. Rows which share an `id` are guaranteed to have come from a single trip update, *even if* there are duplicate IDs in the GTFS-realtime data. If there are duplicate IDs, they will be deduplicated by appending _duplicated_1, _duplicated_2, etc., and a warning will be issued. If a trip update contains no stop time updates, it will result in a single row in the output.

Value

a data frame with a row for each stop time update (and a minimum of one row for each trip update when there are no stop time updates. The data frame has the following columns, with descriptions based on or copied directly from the GTFS-realtime specification. The path of each column identifies where in the GTFS-realtime TripUpdate object the field's value is taken from.

Note that

- `id`: GTFS-realtime entity ID. Since one trip update can become multiple rows (for example, for different languages, or different informed entities), the ID can be used to identify rows that came from the same trip update.

  IDs are required by the specification to be unique within a GTFS-realtime file, but sometimes are not. If there are non-unique IDs in the feed, they will be made unique when data are loaded by appending `_duplicated_1`, `_duplicated_2`, and so on and a warning will be issued, which guarantees that all rows from a single file have unique IDs. When working with archived data, there will quite likely be duplicated IDs between files archived at different times (path: `id` property of `FeedEntity` containing this `TripUpdate`).
- `trip_id`: The `trip_id` from the static GTFS feed that this selector refers to. For non frequency-based trips (trips not defined in GTFS `frequencies.txt`), this field is enough to uniquely identify the trip. For frequency-based trips defined in GTFS `frequencies.txt`, `trip_id`, `start_time`, and `start_date` are all required. For scheduled-based trips (trips not defined in GTFS frequencies.txt), `trip_id` can only be omitted if the trip can be uniquely identified by a combination of `route_id`, `direction_id`, `start_time`, and `start_date`, and all those fields are provided. When `trip_schedule_relationship` is `NEW`, it must be specified with a unique value not defined in the GTFS static. When `trip_schedule_relationship` is `REPLACEMENT`, the trip_id identifies the trip from static GTFS to be replaced. When `trip_schedule_relationship` is `DUPLICATED` within a TripUpdate, the `trip_id` identifies the trip from static GTFS to be duplicated. (path: `trip.trip_id`)
- `route_id`: The `route_id` from the static GTFS that this selector refers to. If `trip_id` is omitted, `route_id`, `direction_id`, `start_time`, and `trip_schedule_relationship=SCHEDULED` must all be set to identify a trip instance. When `trip_schedule_relationship` is `NEW`, `route_id` must be specified for route which the new trip belongs to. (path: `trip.route_id`)
- `direction_id`: The direction_id from the GTFS feed trips.txt file, indicating the direction of travel for trips this selector refers to. If trip_id is omitted, direction_id must be provided.

  Caution: this field is still experimental, and subject to change. It may be formally adopted in the future. (path: `trip.direction_id`)
- `start_time`:The initially scheduled start time of this trip instance. When the `trip_id` corresponds to a non-frequency-based trip, this field should either be omitted or be equal to the

value in the GTFS feed. When the trip_id correponds to a frequency-based trip defined in GTFS `frequencies.txt`, start_time is required and must be specified for trip updates and vehicle positions. If the trip corresponds to exact_times=1 GTFS record, then start_time must be some multiple (including zero) of headway_secs later than `frequencies.txt` `start_time` for the corresponding time period. If the trip corresponds to exact_times=0, then its start_time may be arbitrary, and is initially expected to be the first departure of the trip. Once established, the `start_time` of this frequency-based exact_times=0 trip should be considered immutable, even if the first departure time changes – that time change may instead be reflected in a StopTimeUpdate in a TripUpdate. Format and semantics of the field is same as that of GTFS/frequencies.txt/start_time, e.g., `11:15:35` for 11:15:35 AM or `25:15:35` for 1:15:35 AM the day after the service day. (path: `trip.start_time`)

  - `start_date`: The start date of this trip instance in YYYYMMDD format. For scheduled trips (trips not defined in GTFS frequencies.txt), this field must be provided to disambiguate trips that are so late as to collide with a scheduled trip on a next day. For example, for a train that departs 8:00 and 20:00 every day, and is 12 hours late, there would be two distinct trips on the same time. This field can be provided but is not mandatory for schedules in which such collisions are impossible - for example, a service running on hourly schedule where a vehicle that is one hour late is not considered to be related to schedule anymore. This field is required for frequency-based trips defined in GTFS `frequencies.txt`. (path: `trip.start_date`)

- `trip_schedule_relationship`: How this trip update is related to the trip in the scheduled GTFS. (path: `trip.schedule_relationship`). Possible values:
  - `SCHEDULED`: Trip that is running in accordance with its GTFS schedule, or is close enough to the scheduled trip to be associated with it.
  - `ADDED`: This value has been deprecated as the behavior was unspecified. Use `DUPLICATED` for an extra trip that is the same as a scheduled trip except the start date or time, or `NEW` for an extra trip that is unrelated to an existing trip.
  - `UNSCHEDULED`: A trip that is running with no schedule associated to it (GTFS `frequencies.txt` `exact_times=0`). Trips with `trip_schedule_relationship=UNSCHEDULED` must also set all `stop_schedule_relationship=UNSCHEDULED`.
  - `CANCELED`: A trip that existed in the schedule but was removed.
  - `REPLACEMENT`: A trip that replaces an existing trip in the schedule. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
  - `DUPLICATED`: An extra trip that was added in addition to a running schedule, for example, to replace a broken vehicle or to respond to sudden passenger load. Used with `trip_id`, `start_date`, and `start_time` to copy an existing trip from static GTFS but start at a different service date and/or time. Duplicating a trip is allowed if the service related to the original trip in (CSV) GTFS (in calendar.txt or calendar_dates.txt) is operating within the next 30 days. The trip to be duplicated is identified via `trip_id`. This enumeration does not modify the existing trip referenced by `trip_id` - if a producer wants to cancel the original trip, it must publish a separate TripUpdate with the value of `CANCELED` or `DELETED`. If a producer wants to replace the original trip, a value of `REPLACEMENT` should be used instead.
    Trips defined in GTFS `frequencies.txt` with `exact_times` that is empty or equal to 0 cannot be duplicated.
    Existing producers and consumers that were using the ADDED enumeration to represent duplicated trips must follow the migration guide to transition to the `DUPLICATED` enumer-

ation. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
  - DELETED: A trip that existed in the schedule but was removed and must not be shown to users. DELETED should be used instead of CANCELED to indicate that a transit provider would like to entirely remove information about the corresponding trip from consuming applications, so the trip is not shown as cancelled to riders, e.g. a trip that is entirely being replaced by another trip. This designation becomes particularly important if several trips are cancelled and replaced with substitute service. If consumers were to show explicit information about the cancellations it would distract from the more important real-time predictions. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
  - NEW: An extra trip unrelated to any existing trips, for example, to respond to sudden passenger load. NOTE: This field is still experimental, and subject to change. It may be formally adopted in the future.
- modifications_id: Linkage to any modifications done to this trip (shape changes, removal or addition of stops). Reading trip modifications themselves is not yet supported (see #21). If you have a feed with trip modifications, please comment on that issue so we are aware they exist in the wild. (path: trip.modification_id)
- vehicle_id: Internal system identification of the vehicle. Should be unique per vehicle, and is used for tracking the vehicle as it proceeds through the system. This id should not be made visible to the end-user; for that purpose use the label field. (path: vehicle.id)
- vehicle_label: User visible label, i.e., something that must be shown to the passenger to help identify the correct vehicle. (path: vehicle.label)
- vehicle_license_plate: The license plate of the vehicle. (path: vehicle.license_plate)
- vehicle_wheelchair_accessible: Whether the vehicle is wheelchair accessible. If provided, can overwrite the wheelchair_accessible value from the static GTFS. (path: vehicle.wheelchair_accessible) Possible values:
  - NO_VALUE The trip doesn't have information about wheelchair accessibility. This is the default behavior. If the static GTFS contains a wheelchair_accessible value, it won't be overwritten.
  - UNKNOWN The trip has no accessibility value present. This value will overwrite the value from the GTFS.
  - WHEELCHAIR_ACCESSIBLE The trip is wheelchair accessible. This value will overwrite the value from the GTFS.
  - WHEELCHAIR_INACCESSIBLEThe trip is not wheelchair accessible. This value will overwrite the value from the GTFS.
- stop_sequence: Must be the same as in stop_times.txt in the corresponding GTFS feed. Either stop_sequence or stop_id must be provided - both fields cannot be empty. stop_sequence is required for trips that visit the same stop_id more than once (e.g., a loop) to disambiguate which stop the prediction is for. Required if trip_schedule_relationship is NEW or REPLACEMENT, and the value must be increasing along the trip. (path: stop_time_update.stop_sequence
- stop_id: Must be the same as in stops.txt in the corresponding GTFS feed. Required if trip_schedule_relationship is NEW or REPLACEMENT. (path: stop_time_update.stop_id)
- The next few fields contain information about the arrival time at the stop. If stop_schedule_relationship is empty or SCHEDULED either arrival or departure must be provided - both fields cannot be

empty. arrival and departure may both be empty when stop_schedule_relationship is SKIPPED Required if trip_schedule_relationship is NEW or REPLACEMENT.

  - arrival_delay: Delay (in seconds) can be positive (meaning that the vehicle is late) or negative (meaning that the vehicle is ahead of schedule). Delay of 0 means that the vehicle is exactly on time. Forbidden if stop_schedule_relationship is NO_DATA. Required otherwise if arrival_time is not given. (path: stop_time_update.arrival.delay)
  - arrival_time: Estimated or actual date and time of arrival at the stop, in local time (specified by timezone argument). Forbidden if stop_schedule_relationship is NO_DATA. Required otherwise if arrival_delay is not given. (path: stop_time_update.arrival.time)
  - arrival_scheduled_time: Scheduled date/time of arrival in local time. Optional if trip_schedule_relationship is NEW, REPLACEMENT or DUPLICATED, forbidden otherwise. (path: stop_time_update.arrival.scheduled_time)
  - arrival_uncertainty: If arrival_uncertainty is omitted, it is interpreted as unknown. To specify a completely certain prediction, set its uncertainty to 0. Forbidden if stop_schedule_relationship is NO_DATA. Units are not documented in the GTFS-realtime specification, but are likely seconds. (path: stop_time_update.arrival.uncertainty)

- The next few fields contain information about the departure time at the stop. If stop_schedule_relationship is empty or SCHEDULED either arrival or departure must be provided - both fields cannot be empty. arrival and departure may both be empty when stop_schedule_relationship is SKIPPED Required if trip_schedule_relationship is NEW or REPLACEMENT.

  - departure_delay: Delay (in seconds) can be positive (meaning that the vehicle is late) or negative (meaning that the vehicle is ahead of schedule). Delay of 0 means that the vehicle is exactly on time. Forbidden if stop_schedule_relationship is NO_DATA. Required otherwise if departure_time is not given. (path: stop_time_update.departure.delay)
  - departure_time: Estimated or actual date and time of arrival at the stop, in local time (specified by timezone argument). Forbidden if stop_schedule_relationship is NO_DATA. Required otherwise if departure_delay is not given. (path: stop_time_update.departure.time)
  - departure_scheduled_time: Scheduled date/time of arrival in local time. Optional if trip_schedule_relationship is NEW, REPLACEMENT or DUPLICATED, forbidden otherwise. (path: stop_time_update.departure.scheduled_time)
  - departure_uncertainty: If departure_uncertainty is omitted, it is interpreted as unknown. To specify a completely certain prediction, set its uncertainty to 0. Forbidden if stop_schedule_relationship is NO_DATA. Units are not documented in the GTFS-realtime specification, but are likely seconds. (path: stop_time_update.departure.uncertainty)

- departure_occupancy_status: How full the vehicle is expected to be upon departure. (path: stop_time_update.departure_occupancy_status). Possible values (with official descriptions from the GTFS-realtime spec).:

  - EMPTY: The vehicle or carriage is considered empty by most measures, and has few or no passengers onboard, but is still accepting passengers.
  - MANY_SEATS_AVAILABLE: The vehicle or carriage has a large number of seats available. The amount of free seats out of the total seats available to be considered large enough to fall into this category is determined at the discretion of the producer.
  - FEW_SEATS_AVAILABLE: The vehicle or carriage has a relatively small number of seats available. The amount of free seats out of the total seats available to be considered small enough to fall into this category is determined at the discretion of the feed producer.

  - STANDING_ROOM_ONLY: The vehicle or carriage can currently accommodate only standing passengers.
  - CRUSHED_STANDING_ROOM_ONLY: The vehicle or carriage can currently accommodate only standing passengers and has limited space for them.
  - FULL: The vehicle or carriage is considered full by most measures, but may still be allowing passengers to board.
  - NOT_ACCEPTING_PASSENGERS: The vehicle or carriage is not accepting passengers, but usually accepts passengers for boarding.
  - NO_DATA_AVAILABLE: The vehicle or carriage doesn't have any occupancy data available at that time.
  - NOT_BOARDABLE: The vehicle or carriage is not boardable and never accepts passengers. Useful for special vehicles or carriages (engine, maintenance carriage, etc...).

- stop_schedule_relationship: How this particular stop time relates to the underlying schedule, not to be confused with trip_schedule_relationship which describes how the trip overall relates to the schedule. (path: stop_time_update.schedule_relationship. Possible values:
  - SCHEDULED: The vehicle is proceeding in accordance with its static schedule of stops, although not necessarily according to the times of the schedule. This is the default behavior. At least one of arrival and departure must be provided. Frequency-based trips (GTFS frequencies.txt with exact_times = 0) should not have a SCHEDULED value and should use UNSCHEDULED instead.
  - SKIPPED: The stop is skipped, i.e., the vehicle will not stop at this stop. Arrival and departure are optional. When set SKIPPED is not propagated to subsequent stops in the same trip (i.e., the vehicle will stop at subsequent stops in the trip unless those stops also have a stop_time_update with schedule_relationship: SKIPPED). Delay from a previous stop in the trip does propagate over the SKIPPED stop. In other words, if a stop_time_update with an arrival or departure prediction is not set for a stop after the SKIPPED stop, the prediction upstream of the SKIPPED stop will be propagated to the stop after the SKIPPED stop and subsequent stops in the trip until a stop_time_update for a subsequent stop is provided.
  - NO_DATA: No real-time data is given for this stop. It indicates that there is no realtime timing information available. When set NO_DATA is propagated through subsequent stops so this is the recommended way of specifying from which stop you do not have realtime timing information. When NO_DATA is set, arrival or departure must not be supplied, unless trip_schedule_relationship is NEW or REPLACEMENT, in such case only the scheduled time, but not predictions, must be supplied. When trip_schedule_relationship is NEW or REPLACEMENT, arrival_scheduled_time and departure_scheduled_timemust still be given with scheduled times, as the StopTimeUpdate defines the stop list of the trip. In this case it indicates that the schedule is unrelated to the static GTFS, but real-time prediction is not available yet.
  - UNSCHEDULED: The vehicle is operating a frequency-based trip (GTFS frequencies.txt with exact_times = 0). This value should not be used for trips that are not defined in GTFS frequencies.txt, or trips in GTFS frequencies.txt with exact_times = 1. Trips containing stop_time_updates with stop_schedule_relationship: UNSCHEDULED must also set the trip_schedule_relationship: UNSCHEDULED

    Caution: this field is still experimental, and subject to change. It may be formally adopted in

- `file_timestamp`: Timestamp of the GTFS-realtime file itself (i.e. when the file was generated, not when the updates were generated)
- `file_index`: When reading a ZIP file, a one-based index of which file each observation came from Note that it is in the the order the files appeared in the ZIP file, which may not be chronological.

Examples

```
# This will read an updates feed included with gtfsrealtime. Replace with
# the path to your own file if desired.
file = system.file("nyc-trip-updates.pb.bz2", package = "gtfsrealtime")

# Need to specify timezone so timestamps will be in local time.
read_gtfsrt_trip_updates(file, "America/New_York")
```

Index