Findings

Bhagat-Conway, Matthew, and Matthew Palm. 2026. "{gtfsrealtime}: A Package for Working with GTFS-Realtime Transit Data in R." *Findings*, August 31. https://doi.org/10.32866/001c.168472.

TRANSPORT FINDINGS

{gtfsrealtime}: a Package for Working with GTFS-Realtime Transit Data in R

Matthew Bhagat-Conway, PhD[1], Matthew Palm, PhD[1]

[1] City and Regional Planning, University of North Carolina at Chapel Hill

Keywords: gtfs, gtfsrealtime, transit, r

https://doi.org/10.32866/001c.168472

Findings

Many transit agencies publish real-time data on vehicle positions, arrival predictions, and service alerts in GTFS-realtime format. While these data provide a wealth of information, to date they have been underutilized in research. In part, this is due to a lack of tooling for reading the bespoke binary format into commonly-used research platforms. This article introduces the `{gtfsrealtime}` R package which allows reading GTFS-realtime into the R statistical programming environment.

1. Questions

The General Transit Feed Specification (GTFS) is one of the great successes of open data standards. GTFS is a standardized format for communicating transit schedule data (Voulgaris and Begwani 2023). While GTFS was originally intended to support customer-facing trip planning apps (McHugh 2013), it has seen extensive use in research (Ho 2021). Many open source tools are available for analyzing GTFS data (e.g., Herszenhut et al. 2022; Pereira et al. 2021; Reynolds et al. 2025; Kocsis and Varga 2025), many using the R statistical programming language (R Core Team 2026).

Originally, GTFS only provided data on scheduled arrival times. In 2011, GTFS-realtime was released, allowing agencies to provide real-time information (Rychev 2011). To date, most analytical applications have used the original GTFS specification (retroactively renamed GTFS-static), with some exceptions (e.g., Liu et al. 2023; Javanmard et al. 2025); a Scopus search for GTFS-realtime returns only 17 articles.[1] One factor is the lack of tooling to read GTFS-realtime's bespoke binary format. This article introduces the `{gtfsrealtime}` R package, which reads GTFS-realtime feeds into standard R data frames. It answers in the affirmative the question of whether R can be efficiently used to analyze large GTFS-realtime feeds.

1 The exact search used was `( gtfs realtime ) OR ( gtfsrealtime ) OR ( gtfs-realtime ) OR ( gtfs AND realtime )`

2. Methods

GTFS-realtime feeds come in three types: vehicle positions, trip updates, and service alerts. The `{gtfsrealtime}` package exposes three functions for reading these: `read_gtfsrt_positions`, `read_gtfsrt_trip_updates`, and `read_gtfsrt_alerts`, which are detailed below. There are several experimental extensions to GTFS-realtime, which are not currently supported.

`{gtfsrealtime}` can be installed through the Comprehensive R Archive Network (CRAN):

```
install.packages("gtfsrealtime")
```

The current version is 0.2.1. `{gtfsrealtime}` source code and development versions are available from https://github.com/mattwigway/gtfsrealtime-r.

For performance, `{gtfsrealtime}` has components written in Rust (Rust Team 2025); CRAN provides binary packages for the most recent version of R on Windows and macOS (if you experience installation issues, ensure you are using the latest version of R).

Once installed, it can be loaded (along with a few other packages we will be using):

```
library(gtfsrealtime)
library(ggplot2) # for plotting
library(dplyr) # for general data manipulation
library(data.table) # for high-performance data filtering
library(sf) # for spatial data analysis
```

The following sections demonstrate the main functions of the package, using example data shipped with the package. Note that the example vehicle positions are from a different time period than the other datasets and cannot be joined with them.

`read_gtfsrt_positions`

Read vehicle positions with `read_gtfsrt_positions`. The feed used below comes from the New York City bus network. This file is included with `{gtfsrealtime}` and is compressed with `bzip2`. `{gtfsrealtime}` can read uncompressed files and files compressed with `zip`, `gzip`, or `bzip2`, from the local machine or a URL. To use with your own feed, replace `VEHICLE_POSITIONS_FILE` with your own file or URL.

GTFS-realtime does not include timezone information. We specify a timezone so that times are automatically converted to local time. Timezones are specified in standardized TZ database format (generally `Continent/City`; see list here). `{sf}` is an R package for geographic data (Pebesma 2018; Pebesma and Bivand 2023); specifying `as_sf=TRUE` will convert the data to `{sf}` format so it can be mapped.

Table 1. Key columns in a vehicle position feed

vehicle_id	timestamp	trip_id	route_id	stop_id
MTA NYCT_9771	2026-01-21 18:58:29	MV_A6-...M5_527	M4	400041
MTA NYCT_8440	2026-01-21 18:58:24	JA_A6-...17_334	Q17	501341
MTA NYCT_9770	2026-01-21 18:58:32	GH_A6-...38_640	BX28	101927
MTA NYCT_7112	2026-01-21 18:58:05	EN_A6-...15_142	B15	301136
MTA NYCT_8443	2026-01-21 18:58:32	JA_A6-...17_317	Q17	501369
MTA NYCT_9775	2026-01-21 18:58:06	MV_A6-...M4_454	M4	400645

```
VEHICLE_POSITIONS_FILE = system.file("nyc-vehicle-positions.pb.bz2",
    package = "gtfsrealtime")
positions = read_gtfsrt_positions(
    VEHICLE_POSITIONS_FILE,
    "America/New_York",
    as_sf = TRUE
)
```

This produces a data frame of the positions file. A few rows are shown in Table 1 (only the most relevant columns are shown; all are described in the documentation).

Since we read the data using `as_sf=TRUE`, we can also map the locations of all vehicles, colored by how full they are (Figure 1). One could use other packages such as `ggspatial` (Dunnington 2025) to add a basemap, scale bar, etc.; this is left to the reader for brevity.

```
positions |>
    ggplot(aes(color = occupancy_status)) +
    geom_sf(size = 0.5) +
    theme_void()
```

read_gtfsrt_trip_updates

`{gtfsrealtime}` likewise includes an example trip updates feed from New York MTA buses, which we can read with `read_gtfsrt_trip_updates`. Modify `TRIP_UPDATES_FILE` to work with your own file or URL.

```
TRIP_UPDATES_FILE = system.file("nyc-trip-updates.pb.bz2",
    package = "gtfsrealtime")
updates = read_gtfsrt_trip_updates(
    TRIP_UPDATES_FILE,
    "America/New_York"
)
```

GTFS-realtime trip updates are hierarchical; one trip update can contain information about the trip as a whole as well as updates for multiple stops along that trip. `read_gtfsrt_trip_updates()` flattens that structure into a data frame. As a result, the same `id` may appear in multiple rows when the feed contains stop-level updates for multiple stops. In Table 2, we inspect all rows

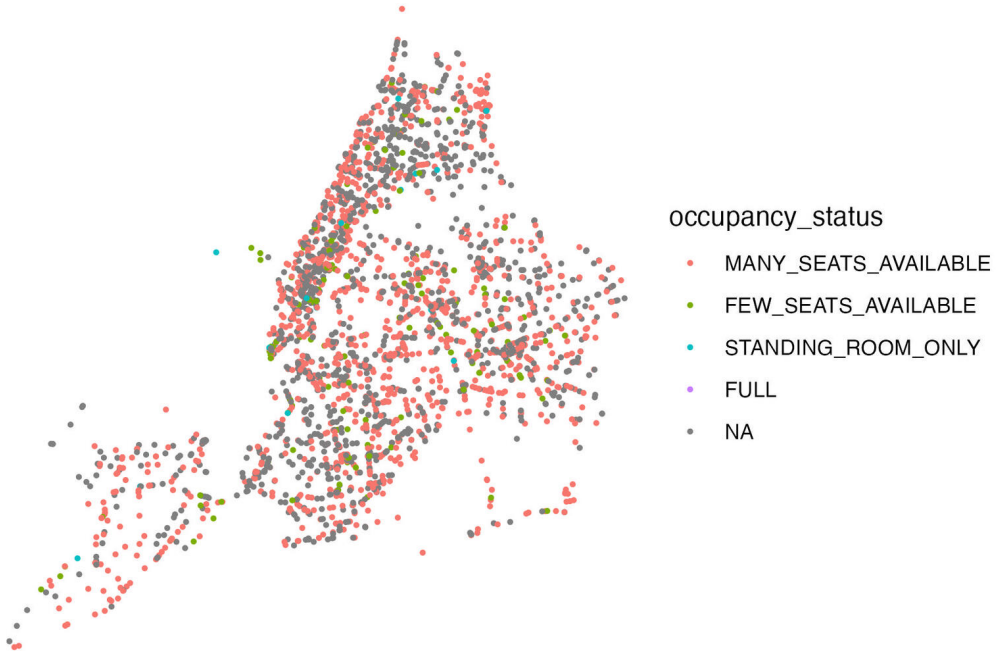

Figure 1. Bus positions in NYC

Table 2. Example arrival time predictions for a single trip.

id	trip_id	route_id	stop_id	arrival_time
MV_A6-...96_826	MV_A6-...96_826	M96	401933	2026-01-28 17:13:20
MV_A6-...96_826	MV_A6-...96_826	M96	401935	2026-01-28 17:14:15
MV_A6-...96_826	MV_A6-...96_826	M96	401936	2026-01-28 17:15:58
MV_A6-...96_826	MV_A6-...96_826	M96	401937	2026-01-28 17:17:42
MV_A6-...96_826	MV_A6-...96_826	M96	404087	2026-01-28 17:21:41
MV_A6-...96_826	MV_A6-...96_826	M96	401939	2026-01-28 17:24:30
MV_A6-...96_826	MV_A6-...96_826	M96	401941	2026-01-28 17:27:43
MV_A6-...96_826	MV_A6-...96_826	M96	401942	2026-01-28 17:30:05
MV_A6-...96_826	MV_A6-...96_826	M96	401943	2026-01-28 17:32:03
MV_A6-...96_826	MV_A6-...96_826	M96	903003	2026-01-28 17:33:38

associated with one `id` (and show the most relevant columns). If a trip update has no stop time updates, it will appear as a single row with all the stop-specific fields NA.

When reading this example feed, `{gtfsrealtime}` warns that some GTFS-realtime entity IDs are duplicated, which is problematic since these are used to identify the rows belonging to a single trip update. In these cases, the package appends suffixes such as `_duplicated_1` so that each trip update can be represented with a unique `id` within a feed (when reading a ZIP file containing multiple feeds, `id`s are only unique within each of the contained feeds, which can be disambiguated with the `file_index` field). This

Table 3. Example alerts data

id	start	end	route_id	stop_id	header_text
MTA NY...:22245	2025-01-03 00:00:35	NA	X38	NA	X28 and X38 stops on Sur...closed in both directions
MTA NY...:22245	2025-01-03 00:00:35	NA	X28	NA	X28 and X38 stops on Sur...closed in both directions

deduplication applies only to the `id` field. Other fields, for example `trip_id`, are not modified so they will still correctly join with GTFS-static. Deduplication also happens with alerts and vehicle position `id`s.

`read_gtfsrt_alerts`

```
ALERTS_FILE = system.file("nyc-service-alerts.pb.bz2",
    package = "gtfsrealtime")

alerts = read_gtfsrt_alerts(
    ALERTS_FILE,
    "America/New_York"
)
```

Like trip updates, alerts are hierarchical; one alert can include multiple time periods, "informed entities" (such as routes or stops), and languages. `read_gtfsrt_alerts()` flattens this structure, repeating the alert for every combination of time period, entity, and language. Therefore, the same alert `id` may appear in multiple rows.

In Table 3, we select the first alert and display the route or stop fields and the alert text. This alert appears as two rows because it affects two routes.

Archived data

Archived data is often useful for analytical applications; `{gtfsrealtime}` also supports reading directly from ZIP files containing multiple GTFS-realtime feeds (for example, all vehicle positions over the course of a day). Examples of working with archived data, including vehicle position animations and stringline charts, are in the `{gtfsrealtime}` documentation.

3. Findings

This article has introduced the `{gtfsrealtime}` R package, an open-source (MIT-licensed) package for reading GTFS-realtime data into R data frames. As far as we know, this is the only actively-maintained package for reading GTFS-realtime into R. It reads the hierarchical GTFS-realtime format into a tabular format, which is conducive to efficient analysis in R. It aims to be as user-friendly as possible, handling duplicate IDs, adding factor columns rather than numeric codes for categorical variables, and processing

timestamps into R date-time objects. It is fast, reading a full day of New York City vehicle positions (n=2.8 million) in 15 seconds on an M1 Macbook Pro—fast enough to readily analyze months or years of data.

We are aware of only one other actively-maintained option for reading GTFS-realtime data into R. GTFS-realtime is based on the Protocol Buffers format, and as such the `{RProtoBuf}` package (Eddelbuettel et al. 2016), a general-purpose package for reading any Protocol Buffers-based format, can read it. `{RProtoBuf}` actually reads GTFS-realtime somewhat faster than `{gtfsrealtime}` (the same full day of vehicle positions takes only 3 seconds to read). However, it returns data in a hierarchical format that is not conducive to efficient analysis in R. Converting to the tabular format necessary for further analysis takes an additional 24 minutes using `vapply` and `lapply`.[2] `RProtoBuf` also does not provide GTFS-realtime specific features such as data compression, ID deduplication, or time zone parsing. Another package, `{gtfsway}`, has not been maintained in many years and is formally archived (Cooley, n.d.). There are of course also GTFS-realtime bindings in many other languages, such as Python.

There is a wealth of applications of real-time transit data—for example, accessibility analysis (e.g., Wessel and Farber 2019), operations (e.g., Aemmer et al. 2022), or passenger information evaluation (e.g, Newmark 2024). We hope that the `{gtfsrealtime}` package will support these types of analysis and more, and will underly development of additional standardized packages for common analyses using realtime data.

Submitted: June 13, 2026 AEST. Accepted: August 19, 2026 AEST. Published: September 01, 2026 AEST.

2 Full details of this test are in the Performance article available in the `{gtfsrealtime}` documentation.

REFERENCES

Aemmer, Zack, Andisheh Ranjbari, and Don MacKenzie. 2022. "Measurement and Classification of Transit Delays Using GTFS-RT Data." *Public Transport* (Heidelberg, Netherlands) 14 (2): 263–85. https://doi.org/10.1007/s12469-022-00291-7.

Cooley, David. n.d. "Gtfsway." Accessed August 8, 2026. https://github.com/SymbolixAU/gtfsway.

Dunnington, Dewey. 2025. *Ggspatial: Spatial Data Framework for Ggplot2*. https://paleolimbot.github.io/ggspatial/.

Eddelbuettel, Dirk, Murray Stokely, and Jeroen Ooms. 2016. "RProtoBuf: Efficient Cross-Language Data Serialization in R." *Journal of Statistical Software* 71 (July): 1–24. https://doi.org/10.18637/jss.v071.i02.

Herszenhut, Daniel, Rafael H. M. Pereira, Pedro R. Andrade, and João Bazzo. 2022. "Gtfstools: General Transit Feed Specification (GTFS) Editing and Analysing Tools." *Zenodo*, May. https://doi.org/10.5281/zenodo.6577028.

Ho, Stanley. 2021. *General Transit Feed Specification (GTFS) Data in Transportation Research: A Review of Applications and Methods*. https://hdl.handle.net/20.500.14721/25263.

Javanmard, Reyhane, Luyu Liu, Jed A. Long, and Jinhyung Lee. 2025. "Using Realtime GTFS to Generate Easy-to-Use Transit Accessibility Measures under Travel Time Uncertainty." *Travel Behaviour and Society* 41 (October): 101054. https://doi.org/10.1016/j.tbs.2025.101054.

Kocsis, Gergely, and Imre Varga. 2025. "Gtfs2net: Extraction of General Transit Feed Specification Data Sets to Abstract Networks and Their Analysis." *Big Data* 13 (1): 30–41. https://doi.org/10.1089/big.2022.0269.

Liu, Luyu, Adam Porr, and Harvey Miller. 2023. "Realizable Accessibility: Evaluating the Reliability of Public Transit Accessibility Using High-Resolution Real-Time Data." *Journal of Geographical Systems* (Heidelberg, Netherlands) 25 (3): 429–51. https://doi.org/10.1007/s10109-022-00382-w.

McHugh, Bibiana. 2013. "Pioneering Open Data Standards: The GTFS Story." In *Beyond Transparency: Open Data and the Future of Civic Innovation*, edited by Brett Goldstein and Lauren Dyson. Code for America Press.

Newmark, Gregory L. 2024. *Assessing GTFS Accuracy*. Mineta Transportation Institute. https://doi.org/10.31979/mti.2024.2017.

Pebesma, Edzer. 2018. "Simple Features for R: Standardized Support for Spatial Vector Data." *The R Journal* 10 (1): 439–46. https://doi.org/10.32614/RJ-2018-009.

Pebesma, Edzer, and Roger Bivand. 2023. *Spatial Data Science: With Applications in R*. Chapman and Hall/CRC. https://doi.org/10.1201/9780429459016.

Pereira, Rafael H. M., Marcus Saraiva, Daniel Herszenhut, Carlos Kaue Vieira Braga, and Matthew Wigginton Conway. 2021. "R5r: Rapid Realistic Routing on Multimodal Transport Networks with R5 in R." *Findings*, 10. https://doi.org/10.32866/001c.21262.

R Core Team. 2026. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. https://doi.org/10.32614/R.manuals.

Reynolds, James, Graham Currie, and Yanda Qu. 2025. "Social Needs and Gaps in Transit Service: New GTFS Tools & Updates to 2021 Data." *Australasian Transport Research Forum* (Auckland, NZ).

Rust Team. 2025. *The Rust Programming Language, Version 1.88*.

Rychev, Vladimir. 2011. "Introducing GTFS-Realtime to Exchange Realtime Transit Updates." Google Lat Long. https://maps.googleblog.com/2011/08/introducing-gtfs-realtime-to-exchange.html.

Voulgaris, Carole Turley, and Charuvi Begwani. 2023. "Predictors of Early Adoption of the General Transit Feed Specification." *Findings*, January. https://doi.org/10.32866/001c.57722.

Wessel, Nate, and Steven Farber. 2019. "On the Accuracy of Schedule-Based GTFS for Measuring Accessibility." *Journal of Transport and Land Use* 12 (1). https://doi.org/10.5198/jtlu.2019.1502.