gtfs2gps: Converting GTFS data to GPS format

Rafael H. M. Pereira, Pedro R. Andrade, Joao Bazzo

25 November 2019

Abstract

Package `gtfs2gps` has a set of functions to convert GTFS data to GPS format using `data.table`. It also has some functions to subset GTFS data in time and space and to convert both representations to simple feature format.

Introduction

Package `gtfs2gps` allows users to handle and converting GTFS data using `data.table` format. Before using the package, just install it from GitHub.

```
devtools::install_github("ipeaGIT/gtfs2gps")
```

Loading data

After loading the package, GTFS data can be read into R by using `read_gtfs()`. This function gets a zipped GTFS file and returns a list of `data.table` objects. The returning list contains the data of each GTFS file indexed according to their file names without extension.

```
library("gtfs2gps")
sp <- read_gtfs(system.file("extdata/saopaulo.zip", package="gtfs2gps"))
names(sp)
```

```
## [1] "agency"      "routes"      "stops"       "stop_times"  "shapes"
## [6] "trips"       "calendar"    "frequencies"
```

```
sp$trips
```

```
##      route_id service_id   trip_id    trip_headsign direction_id shape_id
##   1:  121G-10        USD 121G-10-0   Metrô Tucuruvi            0    52421
##   2:  148L-10        USD 148L-10-0             Lapa            0    52857
##   3:  148L-10        USD 148L-10-1  Cohab Antártica            1    52858
##   4:  1720-10        USD 1720-10-0       Cantareira            0    54502
##   5:  1720-10        USD 1720-10-1       Jd. Guancã            1    54503
##  ---
## 229:  N732-11        USD N732-11-0 Term. Jd. Jacira            0    51990
## 230:  N739-11        USD N739-11-0    Jd. Universal            0    51954
## 231:  N740-11        USD N740-11-0      Jd. Riviera            0    51939
## 232:  N838-11        USD N838-11-0  Cptm Leopoldina            0    52072
## 233:  N840-11        USD N840-11-0     Sta. Cecília            0    52135
```

Note that not all GTFS files are loaded into R. This function loads only the necessary data to spatially and temporally handle trips and stops, which are: agency.txt, calendar.txt, routes.txt, shapes.txt, stop_times.txt, stops.txt, trips.txt, and frequencies.txt (this last one is optional). If a given GTFS zipped file does not contain all the required files then `read_gtfs()` will stop with an error.

Simplifying Data

As GTFS data are usually very big, any exploratory analysis has to firstly subset it in order to speedup the process. There are some functions to filter a GTFS data:

filter_by_shape_id(): Filter shapes using given shape ids.

filter_valid_stop_times(): Return only stop times that have geospatial locations.

filter_week_days(): Remove weekend trips.

filter_single_trip(): Return only one trip per shape_id.

These functions subset all the relevant GTFS files in order to remove all the unnecessary rows, keeping the data consistent. The returning values of the four functions is a list of `data.table` objects, in the same way of the input data. For example, in the code below we filter only shape ids between 53000 and 53020.

```
library(magrittr)

object.size(sp) %>% format(units = "Kb")
```

```
## [1] "5419.4 Kb"
```

```
sp_small <- gtfs2gps::filter_by_shape_id(sp, 53000:53020)

object.size(sp_small) %>% format(units = "Kb")
```

```
## [1] "84 Kb"
```

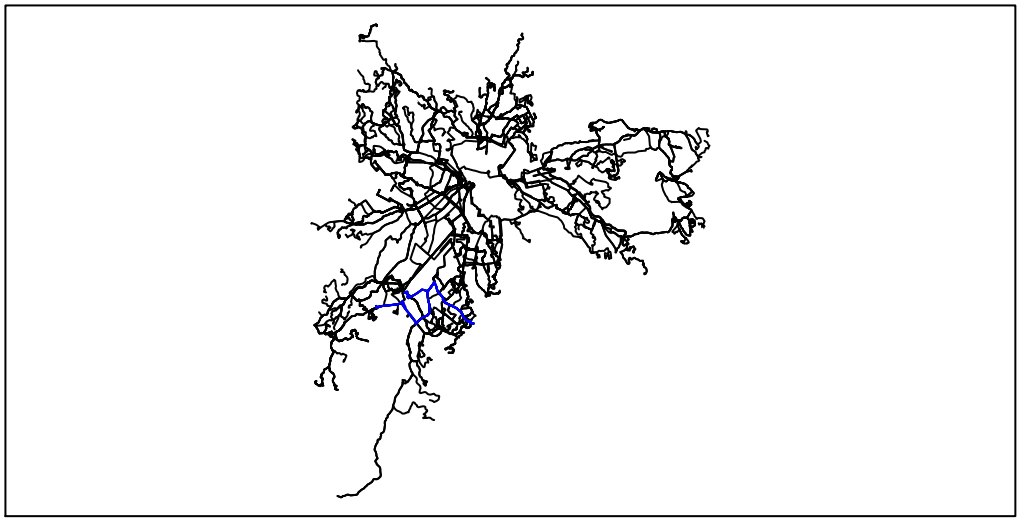
We can then convert both data to simple feature format and plot them.

```
sp_sf <- gtfs_shapes_as_sf(sp)
```

```
## Linking to GEOS 3.6.1, GDAL 2.2.3, PROJ 4.9.3
```

```
sp_small_sf <- gtfs_shapes_as_sf(sp_small)

plot(sf::st_geometry(sp_sf))
plot(sf::st_geometry(sp_small_sf), col = "blue", add = TRUE)
box()
```



After simplifying the data, it is also possible to save it as a new GTFS file using `write_gtfs()`, as shown below.

```
write_gtfs(sp_small, "sp_small.zip")
```

Converting to GPS format

To convert GTFS to GPS format, use `gtfs2gps()`. This function takes a GTFS zipped file as argument and returns a `data.table` where each row works as a GPS sample for a given trip in the GTFS file. See the example as follows.

```
  sp_gps <- gtfs2gps("sp_small.zip", progress = FALSE)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
  sp_gps
```

```
##            trip_id route_type  id shape_pt_lon shape_pt_lat departure_time
##      1: 5129-10-0          3   1    -46.63408    -23.68256       04:00:03
##      2: 5129-10-0          3   2    -46.63407    -23.68250       04:00:06
##      3: 5129-10-0          3   3    -46.63405    -23.68243       04:00:12
##      4: 5129-10-0          3   4    -46.63403    -23.68232       04:00:18
##      5: 5129-10-0          3   5    -46.63400    -23.68221       04:00:24
##     ---
## 486960: 5129-41-1          3 913    -46.63420    -23.68327       00:25:00
## 486961: 5129-41-1          3 914    -46.63419    -23.68316       00:25:03
```

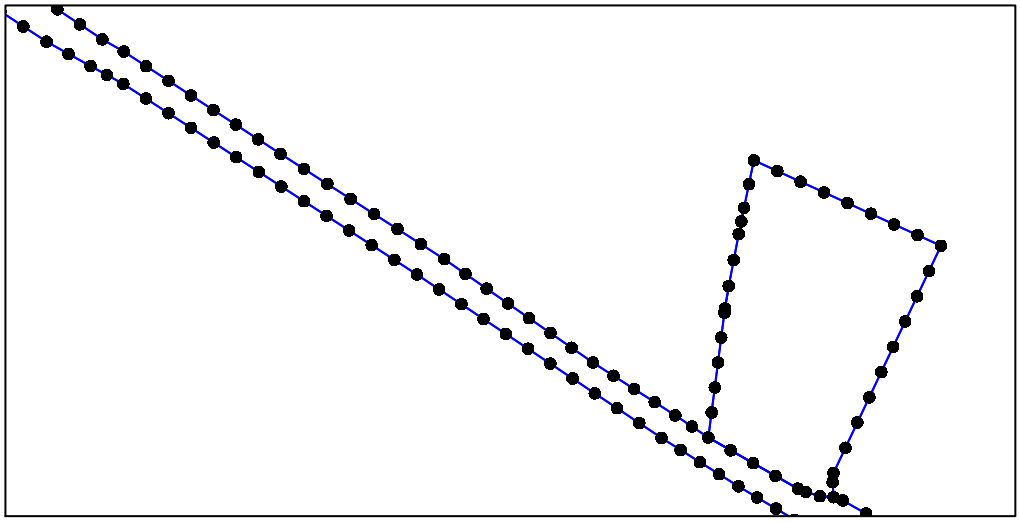
```
## 486962: 5129-41-1          3 915    -46.63417    -23.68304       00:25:07
## 486963: 5129-41-1          3 916    -46.63415    -23.68292       00:25:07
## 486964: 5129-41-1          3 917    -46.63415    -23.68290       00:25:10
##           stop_id stop_sequence      dist      cumdist     speed
##      1: 220013666             1  6.700226     6.700226  7.864121
##      2:        NA            NA  7.156881    13.857106  7.864121
##      3:        NA            NA 12.616563    26.473669  7.864121
##      4:        NA            NA 12.616563    39.090233  7.864121
##      5:        NA            NA 13.364733    52.454965  7.864121
##     ---
## 486960:        NA            NA 13.087084 11949.263407 14.576115
## 486961:        NA            NA 13.087084 11962.350491 14.576115
## 486962:        NA            NA 13.087084 11975.437575 14.576115
## 486963: 220013667            27  2.137067 11977.574642 15.445973
## 486964:        NA            NA 11.847948 11989.422591 15.445973
##             cumtime shape_id
##      1:    3.067197    53001
##      2:    6.343440    53001
##      3:   12.118991    53001
##      4:   17.894541    53001
##      5:   24.012585    53001
##     ---
## 486960: 3300.224698    53011
## 486961: 3303.456938    53011
## 486962: 3306.689178    53011
## 486963: 3307.187265    53011
## 486964: 3309.948672    53011
```

Finally, we can plot the points as shown below. As the points are created using a distance of $15m$, it is necessary to zoom in to visually separate them. The following figure shows the points in a bounding box from the first 60 points.

```
sp_gps_sf <- gps_as_sf(sp_gps)
sp_gps_small <- sp_gps[1:60, ]

sp_gps_small_sf <- gps_as_sf(sp_gps_small)

plot(sf::st_geometry(sp_gps_small_sf), pch = 20)
plot(sf::st_geometry(sp_small_sf), col = "blue", add = TRUE)
plot(sf::st_geometry(sp_gps_sf), add = TRUE, pch = 20)
box()
```



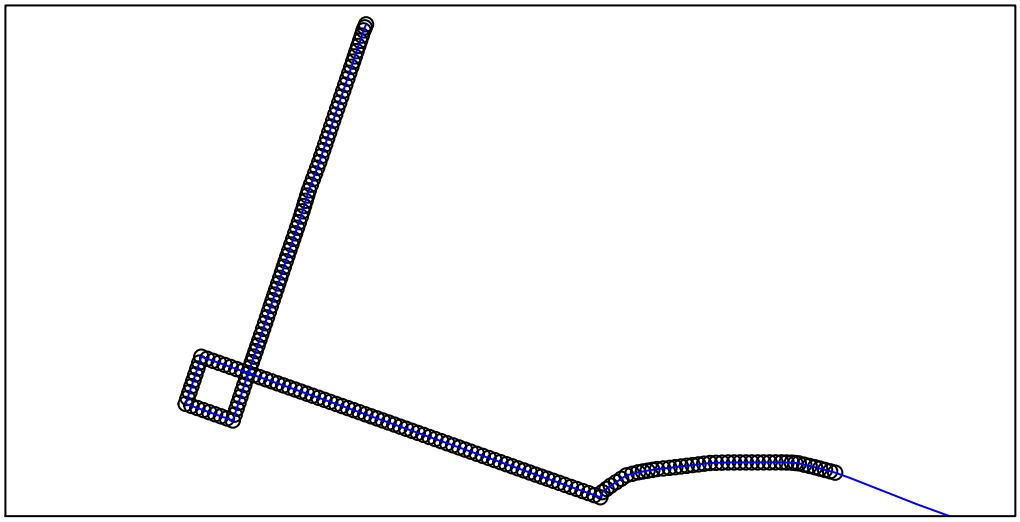
Function `gtfs2gps()` also works with GTFS data that do not have frequency.txt. It automatically recognises the format and then it uses only stop_times.txt to create the GPS data.

```
poa <- system.file("extdata/poa.zip", package="gtfs2gps")

poa_gps <- gtfs2gps(poa, progress = FALSE)
poa_gps_sf <- gps_as_sf(poa_gps)

poa_sf <- read_gtfs(poa) %>% gtfs_shapes_as_sf()

plot(sf::st_geometry(poa_gps_sf[1:200,]))
plot(sf::st_geometry(poa_sf), col = "blue", add = TRUE)
box()
```



Final remarks

If you have any suggestions or want to report an error, please visit the GitHub page of the package here.