Package 'telraamStats'

May 27, 2024

Title Retrieval and Visualization of Mobility Data from 'Telraam' Sensors

Version 1.1.2

Description Streamline the processing of 'Telraam' data, sourced from open data mobility sensors. These tools range from data retrieval (without the need for API knowledge) to data visualization, including data preprocessing.

License CC BY 4.0

BugReports https://github.com/KetsiaGuichard/telraamStats/issues/

Imports config, dplyr, ggplot2, httr, jsonlite, lubridate, paletteer, purrr, reshape2, rlang, scales, tidyr, yaml

Encoding UTF-8

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://annuaire.agistaterre.org/telraamStats/

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Lilou Augeray [aut],
Ulysse Caromel [aut],
Ketsia Guichard [aut, cre] (<https://orcid.org/0009-0002-2280-7465>),
Pascal Irz [aut] (<https://orcid.org/0000-0002-2066-8935>),
Mijin Park [aut],
Tanguy Richard [aut],
Agis-ta-terre [cph, fnd]

Maintainer Ketsia Guichard <ketsia.guichard@univ-rennes.fr>

Repository CRAN

Date/Publication 2024-05-27 17:40:02 UTC

R topics documented:

`create_config` *Create config file if needed.*

Description

If you want to specify the IDs of your own sensors, this function create a local configuration template file in the `\inst` directory, to edit with specific information. By default, the function doesn't create the file in the project directory but in a temp directory. If you want to have a permanent configuration, please use .

Usage

```
create_config(
  segments = list(`segment-01` = "9000000000", `segment-02` = "9000000000"),
  create_directory = FALSE,
  overwrite = FALSE
)
```

Arguments

`segments`	Named List of segments ("name1" = "9000000000", ...). Default to the example version.
`create_directory`	Boolean: Does the file need to be created in the project directory? Default to FALSE.
`overwrite`	Boolean: if the file exist, should it be overwriten? Default to FALSE.

Details

If you use the temporary options, please fill directly the name and number of your sensors in the "segments" argument.

Value

Boolean: TRUE if the file is created, FALSE overwise (config already exists for example).

Examples

```
create_config(create_directory=FALSE)
list_of_segments = list("Burel"= "9000002156", "Vitre" = "9000001844")
create_config(segments = list_of_segments,
  create_directory = FALSE,
  overwrite = TRUE) # the file already exists
```

---

get_api_state *Check API state*

Description

Return the state of the 'Telraam' API. Determine if updates can be made.

Usage

```
get_api_state(key = get_telraam_token())
```

Arguments

key	the API key (set by the set_telraam_token function - default -, or directly filled).

Value

Boolean: TRUE if the API responds well, FALSE otherwise.

Examples

```
my_token <- 'ThisIsNotAValidToken'
get_api_state(my_token)
```

get_telraam_token *Get the current authentication token for the 'Telraam' API*

Description

Get the current authentication token for the 'Telraam' API

Usage

```
get_telraam_token()
```

Value

Token currently used, set by `set_telraam_token()`

Examples

```
my_token <- "MyTelraamToken"
set_telraam_token(my_token)
get_telraam_token()
```

gg_availability *Availability and quality of sensors during a period through a heatmap.*

Description

Higher is the uptime average, higher is the quality of data. A null uptime means that the sensor wasn't available during this period.

Usage

```
gg_availability(enriched_data, date_range = NULL)
```

Arguments

`enriched_data`	enriched data.frame containing all the data for all your sensors
`date_range`	Date vector. Example: c('2021-01-01','2022-01-01'). Full period if NULL.

Value

Graph showing availability and quality of sensors over the selected date range.

Examples

```
gg_availability(traffic)
```

gg_car_speed_histogram *Histogram of car speed over a period, for a segment or a subset of segment.*

Description

Histogram of car speed over a period, for a segment or a subset of segment.

Usage

```
gg_car_speed_histogram(
  enriched_data,
  date_range = NULL,
  segments = NULL,
  weekday = NULL,
  hours = NULL,
  aggregated_by = NULL
)
```

Arguments

enriched_data	enriched data.frame containing all the data for all your sensors
date_range	Date vector. Example: c('2021-01-01','2022-01-01'). Full period if NULL (default).
segments	Character vector. Selected road segment, all if NULL (default).
weekday	Character vector. Weekday choosen. Default to the all week.
hours	Integer vector. Hours choosen, default to the all day.
aggregated_by	Character. Enables comparison with other segments or weekdays. Options are : 'segment_name', 'weekday', NULL (no comparison, default).

Value

Graph showing histogram of car speed over a period.

Examples

```
library(dplyr)
subset_traffic <- traffic %>% filter(day < '2022-02-01', hour > 9)
gg_car_speed_histogram(subset_traffic)
gg_car_speed_histogram(subset_traffic,
  aggregated_by = 'segment_name')
gg_car_speed_histogram(subset_traffic,
  weekday = c('monday','sunday'),
  segments = 'RteVitre-06',
  hours = 17:20,
  aggregated_by = "weekday")
```

gg_car_speed_v85 *Average of v85 car speed per hour over a period, for a segment or a subset of segment.*

Description

v85 is the estimated car speed limit in km/h that 85% of all cars respect. 15% of drivers drive faster than this v85 indicator.

Usage

```
gg_car_speed_v85(
  enriched_data,
  date_range = NULL,
  segments = NULL,
  weekday = NULL,
  hours = NULL,
  aggregated_by = NULL
)
```

Arguments

enriched_data	enriched data.frame containing all the data for all your sensors
date_range	Date vector. Example: c('2021-01-01','2022-01-01'). Full period if NULL (default).
segments	Character vector. Selected road segment, all if NULL (default).
weekday	Character vector. Weekday choosen. Default to the all week.
hours	Integer vector. Hours choosen, default to the all day.
aggregated_by	Character. Enables comparison with other segments or weekdays. Options are : 'segment_name', 'weekday', NULL (no comparison, default).

Value

Graph showing the average of v85 speed per hour.

Examples

```
library(dplyr)
subset_traffic <- traffic %>% filter(day < '2022-02-01', hour > 9)
gg_car_speed_histogram(subset_traffic[0:100,])
gg_car_speed_histogram(subset_traffic, aggregated_by = 'segment_name')
gg_car_speed_histogram(subset_traffic,
  weekday = c('monday','sunday'),
  segments = 'RteVitre-06',
  hours = 17:20,
  aggregated_by = "weekday")
```

gg_traffic_avg *Average of traffic during a week.*

Description

A short description... Average of traffic during a week, over a period for a segment or a subset of segment, for a transportation mode or more, for a direction or both.

Usage

```
gg_traffic_avg(
  enriched_data,
  date_range = NULL,
  segments = NULL,
  modes = c("heavy", "car"),
  direction = "both",
  weekday = NULL,
  aggregated_by = "weekday"
)
```

Arguments

enriched_data	enriched data.frame containing all the data for all your sensors
date_range	Date vector. example: c('2021-01-01','2022-01-01'). Full period if NULL (default).
segments	Character vector. Selected road segment, all if NULL (default).
modes	Character vector. Different modes of transportation aggregated (heavy, car, bike, pedestrian) . Default: heavy & car
direction	Character. Direction of the traffic (lft, rgt, both). Default to both.
weekday	Character vector. Weekday choosen. Default to the all week.
aggregated_by	Character. Options are: 'segment_name', 'weekday', 'direction', 'mode'. Default: 'weekday'.

Value

Graph showing weekly average evolution of traffic (for specified parameters) during the specified period.

Examples

```
gg_traffic_avg(traffic)
gg_traffic_avg(traffic,
  date_range = c('2022-07-01','2022-09-01'),
  segment = 'RteVitre-06',
  mode = 'car',
  direction = 'rgt',
```

```
  weekday = c('monday','friday')
  )
```

`gg_traffic_evolution` *Evolution of traffic and smoothed traffic.*

Description

Evolution of traffic (global, per mode ou per direction), smoothed traffic during a period.

Usage

```
gg_traffic_evolution(
  enriched_data,
  date_range = NULL,
  segments = NULL,
  modes = c("heavy", "car"),
  direction = "both",
  smoothed = TRUE,
  agg_day = TRUE
)
```

Arguments

`enriched_data`	enriched data.frame containing all the data for all your sensors
`date_range`	Date vector. Example: c('2021-01-01','2022-01-01'). Full period if NULL (default).
`segments`	Character vector. Selected road segment by its name, all if NULL (default).
`modes`	Character vector. Different modes of transportation aggregated (heavy, car, bike, pedestrian) . Default: heavy & car
`direction`	Character. Direction of the traffic (lft, rgt, both). Default to both.
`smoothed`	Boolean. Should the smoothed traffic be plotted ? Default: True
`agg_day`	Boolean. Should the data be aggregated per day ? Default : True

Value

Graph showing the evolution of traffic (for specified parameters) during the specified period.

Examples

```
gg_traffic_evolution(traffic)
gg_traffic_evolution(traffic,
  date_range = c('2022-01-01','2022-03-01'),
  segment = 'RteVitre-06',
  mode = c('car','pedestrian'),
  direction = 'lft',
  smoothed = FALSE,
  agg_day = FALSE)
```

import_sensor *Imports data associated with a list of sensors*

Description

Imports data associated with a given list of sensor names from .RData files contained in a data directory. The main purpose of this function is to load the data saved with write update data.

Usage

```
import_sensor(list_sensor)
```

Arguments

list_sensor	A character vector specifying the names of sensors to import data for.

Value

A dataframe containing the imported data.

Examples

```
## Not run:  # This example requires a valid API key
period <- as.Date(c('2022-01-01', '2022-12-31'))
write_update_data('RteVitre-06', period[1], period[2])
write_update_data('ParisArcEnCiel-05', period[1], period[2])
import_sensor(c('RteVitre-06', 'ParisArcEnCiel-05'))

## End(Not run)
```

retrieve_sensor *Retrieves data associated with a sensor from the Telraam API*

Description

Retrieves data associated with a sensor from the Telraam API. The data is retrieved for a specified time period between `start_date` and `end_date` (inclusive).

Usage

```
retrieve_sensor(segment_name, start_date, end_date, key = get_telraam_token())
```

Arguments

segment_name	Character. Name of the segment, as specified in config.
start_date	Date. Start date "aaaa-mm-jj", must be of the date type.
end_date	Date. End date "aaaa-mm-jj", must be of the date type.
key	the API key (set by the `set_telraam_token()` function)

Value

Dataframe from Telraam API, enriched with `enrich_traffic()` function.

Examples

```
## Not run:  # This function requires a valid API key
period <- as.Date(c('2022-01-01', '2022-12-31'))
retrieve_sensor('RteVitre-06', period[1], period[2])

## End(Not run)
```

`set_telraam_token` *Saves an authentication token for the 'Telraam' API.*

Description

If you want to get this token after this instruction, please use `get_telraam_token()`.

Usage

```
set_telraam_token(token)
```

Arguments

`token`	a `string` with the token

Value

Boolean: TRUE if the token is correctly set

Examples

```
my_token <- "MyTelraamToken"
set_telraam_token(my_token)
get_telraam_token()
```

`traffic` *Traffic data of Telraam sensors in Châteaubourg (FR)*

Description

Telraam sensors continously monitor a street form a citizen window. They count heavy vehicules, cars, twok-wheelers and pedestrians, every hour. Châteaubourg is one of the city in France with the highest density of sensors This dataframe is a subset of sensors data in Châteaubourg for 2022. Additional properties are not present natively in the Telraam API but are added by the package.

Usage

```
traffic
```

Format

`traffic`**:**

A data frame with 16,729 rows and 22 columns:

instance_id Sensor number. Equals -1 if the API request was made for a road segment and not for a camera.

segment_id Road segment Telraam ID. Equals -1 if the API request was made for a camera and not for a road segment.

segment_name Additional property - Segment name specified in configuration file.

segment_fullname Additional property - Concatenation of the segment_id and the segment name specified in configuration.

date date and UTC time of the reporting interval (beginning of the interval).

day Additional property - Day of the reporting interval.

hour Additional property - Hour of the reporting interval.

weekday Additional property - Weekday of the reporting interval.

holiday Additional property - boolean, indicates whether this entry is during a French public holiday.

vacation Indicates whether this entry is during a French vacation period, and if true, the vacation period name.

interval can be "hourly" or "daily" for hourly or daily aggregate data.

uptime between 0 and 1, represents the portion of the reporting interval that was actively spent counting the traffic

uptime_quality Additional property - boolean, indicates whether this entry has an uptime greather or equal than 0.

heavy, heavy_lft, heavy_rgt number of heavy vehicles, total and in both directions.

car, car_lft, car_rgt number of cars, total and in both directions.

bike, bike_lft, bike_rgt number of two-wheelers, total and in both directions.

pedestrian, pedestrian_lft, pedestrian_rgt number of pedestrians, total and in both directions.

direction 1, internal consistency value for Telraam.

car_speed_hist_0to70plus, car_speed_hist_0to120plus the estimated car speed distribution in 10 km/h bins from 0 to 70+ km/h or 120+ km/h (in percentage of the total 100%).

timezone name of the Time zone where the segment can be found.

v85 estimated car speed limit in km/h that 85% of all cars respect

Source

https://telraam-api.net/

`write_update_data` *Write or update the sensor data in the data folder*

Description

Writes or updates the sensor data in the data folder. It retrieves the data for the specified sensor between `start_date` and `end_date` (inclusive) using the `retrieve_sensor` function, and then converts certain columns to character strings before writing the data to a RData file in the `data` folder (if `create_directory = TRUE`), to a temporary folder otherwise.

Usage

```
write_update_data(segment_name, start_date, end_date, create_directory = FALSE)
```

Arguments

`segment_name`	Character. Name of the segment, as specified in config.
`start_date`	Date. Start date "aaaa-mm-jj"
`end_date`	Date. End date "aaaa-mm-jj"
`create_directory`	Boolean: Does the file need to be created in the project directory? Default to FALSE.

Value

Boolean: TRUE if the data is well saved/written, FALSE otherwise (no data for example)

Examples

```
## Not run:  # This function requires a valid API key
period <- as.Date(c('2022-01-01', '2022-12-31'))
write_update_data('RteVitre-06', period[1], period[2])

## End(Not run)
```

Index