Package 'gtfs2emis'

July 22, 2025

Type Package

Title Estimating Public Transport Emissions from General Transit Feed Specification (GTFS) Data

Version 0.1.1

Description
A bottom up model to estimate the emission levels of public transport systems based on General Transit Feed Specification (GTFS) data. The package requires two main inputs: i) Public transport data in the GTFS standard format; and ii) Some basic information on fleet characteristics such as fleet age, technology, fuel and Euro stage. As it stands, the package estimates several pollutants at high spatial and temporal resolutions. Pollution levels can be calculated for specific transport routes, trips, time of the day or for the transport system as a whole. The output with emission estimates can be extracted in different formats, supporting analysis on how emission levels vary across space, time and by fleet characteristics. A full description of the methods used in the 'gtfs2emis' model is presented in Vieira, J. P. B.; Pereira, R. H. M.; Andrade, P. R. (2022) <doi:10.31219/osf.io/8m2cy>.

URL https://ipeagit.github.io/gtfs2emis/ ,
https://github.com/ipeaGIT/gtfs2emis

BugReports https://github.com/ipeaGIT/gtfs2emis/issues

License MIT + file LICENSE

Depends R (>= 3.6)

Imports checkmate, data.table, furrr, future, gtfs2gps, methods, sf (>= 0.9-0), sfheaders, terra, units

Suggests gtfstools, ggplot2, knitr, lwgeom, progressr, rmarkdown, testthat (>= 2.1.0)

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

LazyData true

NeedsCompilation no

Author Joao Bazzo [aut, cre] (ORCID: <https://orcid.org/0000-0003-4536-5006>),
Rafael H. M. Pereira [aut] (ORCID:
<https://orcid.org/0000-0003-2125-7465>),
Pedro R. Andrade [aut] (ORCID: <https://orcid.org/0000-0001-8675-4046>),
Sergio Ibarra-Espinosa [ctb] (ORCID:
<https://orcid.org/0000-0002-3162-1905>),
Ipea - Institute for Applied Economic Research [cph, fnd]

Maintainer Joao Bazzo <joao.bazzo@gmail.com>

Repository CRAN

Date/Publication 2024-12-02 15:40:12 UTC

Contents

`ef_brazil_cetesb` *Running exhaust emissions factors for buses from Brazil (CETESB)*

Description

Returns a vector or `data.table` of emission factors for buses based on estimates from the Environment Company of Sao Paulo, Brazil (CETESB) 2019. Emission factor estimates are expressed in units 'g/km'.

Usage

```
ef_brazil_cetesb(pollutant, veh_type, model_year, as_list = TRUE)
```

Arguments

`pollutant`	character. Pollutants "CH4", "CO2", "PM10", "N2O", "NOx", "NO2", "NO", "RCHO", "ETOH" "KML" (Vehicle Kilometers Traveled), "FC" (Fuel Consumption), "gD/KWH" (grams of Diesel per kWh), "gCO2/KWH" (grams of CO2 per per kWh), "CO", "HC" (Total Hydrocarbon), "NMHC" (Non-Methane Hydrocarbon), "FS"(Fuel Sales) and "NH3".
`veh_type`	character. Vehicle categories by fuel: "BUS_URBAN_D", "BUS_MICRO_D", "BUS_COACH_D" and "BUS_ARTIC_D".
`model_year`	numeric. Vehicle model year. Supports `model_year` from 1960 to 2020.
`as_list`	logical. If `TRUE` (default), the function returns the output in a `list` format. If `FALSE`, the output is returned in a `data.table` format.

Details

The new convention for vehicles names are translated from CETESB report:

vehicle	description
BUS_URBAN_D	Urban Bus Diesel (5perc bio-diesel)
BUS_MICRO_D	Micro Urban Bus Diesel (5perc bio-diesel)
BUS_COACH_D	Coach (inter-state) Bus Diesel (5perc bio-diesel)
BUS_ARTIC_D	Articulated Urban Bus Diesel (5perc bio-diesel)

The percentage varies of biofuels varies by law.

These emission factors are not exactly the same as the report of CETESB.

1. In this emission factors, there is also NO and NO2 based on split by published in the EMEP/EEA air pollutant emission inventory guidebook.
2. Also, the emission factors were extended till 50 years of use, repeating the oldest value.

Value

data.table. Emission factors in units 'g/km' by model_year.

See Also

Other Emission factor model: `ef_europe_emep()`, `ef_scaled_euro()`, `ef_usa_emfac()`, `ef_usa_moves()`, `emi_europe_emep_wear()`

Examples

```
df <- ef_brazil_cetesb(
          pollutant = c("CO","PM10","CO2","CH4","NOx"),
          veh_type = "BUS_URBAN_D",
          model_year = 2015,
          as_list = TRUE)
```

ef_brazil_cetesb_db — *Emission factors from Environment Company of Sao Paulo, Brazil (CETESB)*

Description

units 'g/km'; Emission factors for buses based on estimates from the Environment Company of Sao Paulo, Brazil (CETESB) 2017, and obtained from vein package. The R script used to organize the CETESB database can be found in the repository <https://github.com/ipeaGIT/gtfs2emis/blob/master/data-raw/ef_brazil_cetesb_db.R>.

Usage

```
ef_brazil_cetesb_db
```

Format

A data.table:

pollutant character; Pollutants: "CH4", "CO2", "PM10", "N2O", "KML", "FC" (Fuel Consumption), "gD/KWH", "gCO2/KWH", "CO", "HC" (Total Hydrocarbon), "NMHC" (Non-Methane Hydrocarbon), "NOx", "NO2", "NO", "RCHO", "ETOH", "FS"(Fuel Sales) and "NH3"

veh_type character; Vehicle categories by fuel:"BUS_URBAN_D", "BUS_MICRO_D", "BUS_COACH_D" and "BUS_ARTIC_D".

model_year numeric; Filter the emission factor to start from a specific base year.

as_list logical; Returns emission factors as a list, instead of data.table format.

See Also

Other emission factor data: ef_europe_emep_db, ef_usa_emfac_db, ef_usa_moves_db

ef_europe_emep — *Speed-dependent emission factor from the European Environment Agency (EMEP/EEA) model*

Description

Returns a list or data.table of emission factors for buses based on EMEP/EEA air pollutant emission inventory guidebooks. The function uses four emission factor databases published by EMEP/EEA, considering the editions of 2019, 2016, 2013 and 2007. Estimates are expressed in units 'g/km'. See more in @details.

Usage

```
ef_europe_emep(
  speed,
  veh_type,
  euro,
  pollutant,
  fuel = "D",
  tech = "SCR",
  slope = 0,
  load = 0.5,
  fcorr = 1,
  as_list = TRUE
)
```

Arguments

speed	units; Speed in 'km/h'.
veh_type	character; Bus type, classified in "Ubus Midi <=15 t","Ubus Std 15 - 18 t","Ubus Artic >18 t", "Coaches Std <=18 t","Coaches Artic >18 t".
euro	character; Euro period of vehicle, classified in "Conventional", "I", "II", "III", "IV", "V", "VI", and "EEV".
pollutant	character; Pollutant, classified in "FC","CO2","CO","NOx","VOC","PM10","EC","CH4","NH3","N2O". "FC" means Fuel Consumption.
fuel	character; Fuel type, classified in "D" (Diesel),"DHD" (Diesel Hybrid ~ Diesel), "DHE" (Diesel Hybrid ~ Electricity), "CNG" (Compressed Natural Gas), "BD" (Biodiesel).
tech	character; After treatment technology, classified in "SCR" (Selective Catalytic Reduction), "EGR" (Exhaust Gas Recirculation), and "DPF+SCR" (Diesel Particulate Filter + SCR, for Euro VI). Default is "SCR" for "IV" and "V". There are no available after treatment technology associated with euro standards "Conventional", "I", "II" and "III".
slope	numeric; Slope gradient, classified in -0.06, -0.04, -0.02, 0.00, 0.02, 0.04 and 0.06. Negative gradients means downhills and positive uphills. Default is 0.0.
load	numeric; Load ratio, classified in 0.0, 0.5 and 1.0. Default is 0.5.
fcorr	numeric; Correction based on fuel composition. The length must be one per each euro standards. Default is 1.0.
as_list	logical; Returns emission factors as a list, instead of data.table format. Default is TRUE.

Details

The new convention for vehicles names are translated from the EMEP/EEA report:

vehicle category	description
Ubus Midi <=15 t	Urban Bus Midi size, Gross Vehicle Weight (GVW) <= 15 tons

Ubus Std 15 - 18 t	Urban Bus Standard size, GVW between 15 - 18 tons
Ubus Artic >18 t	Urban Bus Articulated size, GVW >= 18 tons
Coaches Std <=18 t	Coach (inter-state) Standard size, GVW <= 18 tons
Coaches Artic >18 t	Coach (inter-state) Articulated size, GVW > 18 tons

When the information of vehicle technology does not match the existing database, the function display a message mentioning the returned technology. User can either select an existing data for the combining variables (`euro`, `tech`, `veh_type` and `pollutant`), or accept the assumed change in vehicle technology.

In order to cover more pollutants, vehicle technologies, and fuel consumption data, the function uses four emission factor databases published by EMEP/EEA, considering the editions of 2019, 2016, 2013 and 2007.

The R scripts used to download and pre-process 4 EMEP/EEA editions (2019, 2016, 2013 and 2007) can be accessed in the 'gtfs2emis' GitHub repository at <https://github.com/ipeaGIT/gtfs2emis/blob/master/data-raw/ef_europe_emep_db.R>

EMEP/EEA data and reports can be accessed in the following links:

- 2019 edition https://www.eea.europa.eu/themes/air/air-pollution-sources-1/emep-eea-air-pollutant-
- 2016 edition https://www.eea.europa.eu/publications/emep-eea-guidebook-2016/,
- 2013 edition https://www.eea.europa.eu/publications/emep-eea-guidebook-2013/, and
- 2007 edition https://www.eea.europa.eu/publications/EMEPCORINAIR5/.

Value

List. emission factors in units 'g/km' (list or a data.table).

See Also

Other Emission factor model: `ef_brazil_cetesb()`, `ef_scaled_euro()`, `ef_usa_emfac()`, `ef_usa_moves()`, `emi_europe_emep_wear()`

Examples

```
ef_europe_emep( speed = units::set_units(1:100,"km/h"),
                veh_type = c("Ubus Midi <=15 t","Ubus Std 15 - 18 t","Ubus Artic >18 t"),
                euro = c("IV","V"),
                fuel = "D",
                pollutant = c("CO","PM10","CH4","NOx"),
                as_list = FALSE)
```

`ef_europe_emep_db` *Emission factors from European Environment Agency — EMEP/EEA*

Description

Hot exhaust emission factors are speed dependent functions and are expressed in g/km. It varies by fuel, vehicle segment, euro standard, pollutant, and after treatment technology. These variables are consolidated in different EF equations, given by:

Usage

```
ef_europe_emep_db
```

Format

A data.table with 6431 rows and 22 variables:

Category Buses.

Fuel Fuel type, classified in "D" (Diesel),"DHD" (Diesel Hybrid ~ Diesel), "DHE" (Diesel Hybrid ~ Electricity), "CNG" (Compressed Natural Gas), "BD" (Biodiesel).

Segment character; Bus type, classified in "Ubus Midi <=15 t","Ubus Std 15 - 18 t","Ubus Artic >18 t", "Coaches Std <=18 t","Coaches Artic >18 t".

Euro character; Euro period of vehicle, classified in "Conventional", "I", "II", "III", "IV", "V", "VI", and "EEV".

Technology character; After treatment technology, classified in "SCR" (Selective Catalytic Reduction), "EGR" (Exhaust Gas Recirculation), and "DPF+SCR" (Diesel Particulate Filter + SCR, for Euro VI). Default is "SCR" for "IV" and "V". There are no available after treatment technology associated with euro standards "Conventional", "I", "II" and "III".

Pol character; Pollutant, classified in "FC","CO2","CO","NOx","VOC","PM10","EC","CH4","NH3","N2O". "FC" means Fuel Consumption.

Vmin Minimum speed for emission factor estimation, in km/h.

Vmax Maximum speed for emission factor estimation, in km/h.

Alpha, Beta, Gamma, Delta, Epsilon, Zita, Hta, Thita Constant parameters.

RF Reduction Factor; In percentage (%) units.

k Constant factor.

Details

EF = EF(Alpha, Beta, Gamma, Delta, Epsilon, Zita, Hta, RF, Speed, Function_ID, k, fcorr),

where Alpha, Beta, Gamma, Delta, Epsilon, Zeta, Eta are constant parameters; RF is the Reduction Factor, Speed in the average speed, Function_ID is the equation (function of on the year of the inventory and the pollutant); k is a constant value, and fcorr is the fuel correction factor.

The emissions factors are derived from the EMEP/EEA air pollutant emission inventory guidebook (formerly called the EMEP CORINAIR emission inventory guidebook). The document provides guidance on estimating emissions from both anthropogenic and natural emission sources.

The package presents a combination of emission factors from EMEP/EEA guidelines of 2007, 2013, 2016, and 2019, aiming to cover a greater number of pollutants and vehicle segments. The script used to process the raw EMEP/EEA databases can be found in the repository <https://github.com/ipeaGIT/gtfs2emis/blob/master/data-raw/ef_europe_emep_db.R>.

Source

More information can be found at https://www.eea.europa.eu//publications/emep-eea-guidebook-2019, https://www.eea.europa.eu/publications/emep-eea-guidebook-2016/, https://www.eea.europa.eu/publications/emep-eea-guidebook-2013/, and https://www.eea.europa.eu/publications/EMEPCORINAIR5/.

See Also

Other emission factor data: ef_brazil_cetesb_db, ef_usa_emfac_db, ef_usa_moves_db

---

ef_scaled_euro *Scale local emission factors in order to make emission estimates a function of speed.*

Description

Scale emission factors to account for vehicle speed based on values from the emission factor model by the European Environment Agency (EMEP/EEA). Emission factor estimates are expressed in units 'g/km'.

Usage

```
ef_scaled_euro(
  ef_local,
  speed,
  veh_type,
  euro,
  pollutant,
  fuel = "D",
  tech = "SCR",
  SDC = 19,
  slope = 0,
  load = 0.5,
  fcorr = 1
)
```

Arguments

ef_local	data.frame or a list containing the emission factors data.frame. Local emission factors, in units 'g/km'.
speed	units. Speed in 'km/h'.
veh_type	character. Bus type, classified as "Ubus Midi <=15 t", "Ubus Std 15 - 18 t", "Ubus Artic >18 t", "Coaches Std <=18 t", or "Coaches Artic >18 t".
euro	character. Euro period of vehicle, classified in "Conventional", "I", "II", "III", "IV", "V", "VI", and "EEV".
pollutant	character. Pollutant: "FC", "CO2", "CO", "NOx", "VOC", "PM10", "EC", "CH4", "NH3", "N2O", "FC" (fuel consumption).
fuel	character. Fuel type, classified in "D" (Diesel), "DHD" (Diesel Hybrid ~ Diesel), "DHE" (Diesel Hybrid ~ Electricity), "CNG" (Compressed Natural Gas), "BD" (Biodiesel). Default is "D".
tech	character. After treatment technology, classified in "SCR" (Selective Catalytic Reduction), "EGR" (Exhaust Gas Recirculation), and "DPF+SCR" (Diesel Particulate Filter + SCR, for Euro VI). Default is "SCR" for "IV" and "V".
SDC	numeric. Average speed of urban driving condition in 'km/h'. Default is 19 km/h, which is the average speed adopted in EMEP/EEA report.
slope	numeric. Slope gradient, categorized in -0.06, -0.04, -0.02, 0.00, 0.02, 0.04 and 0.06. Negative gradients means downhills and positive uphills. Default is 0.0.
load	numeric. Passenger load ratio, classified in 0.0, 0.5 and 1.0. Default is 0.5.
fcorr	numeric. Correction based on fuel composition. The length must be one per each euro standards. Default is 1.0.

Details

The scaled emission factor is related to speed by the expression

EF_scaled (V) = EF_local * (EF(V) / EF(SDC)),

where EF_scaled(V) is the scaled emission factors for each street link, EF_local is the local emission factor, EF(V) and EF(SDC) are the EMEP/EEA emission factor the speed of V and the average urban driving speed 'SDC', respectively.

Please note that the function reads the vector arguments in the same order as informed by the user. For instance, if the pollutant input is `c("CO","PM10")` input in the local emission factor function, the order needs to be the same for the pollutant in the `ef_scaled_euro` function.

In the case of vehicle type, which generally changes according to the emission factor source, the input argument in the `ef_scaled_euro` needs to be consistent with the order adopted in the local emission factor function.

For example, if the vector of local vehicle type is `c("BUS_URBAN_D","BUS_MICRO_D")`, the related vector for EMEP/EEA model needs to be `c("Ubus Std 15 - 18 t","Ubus Midi <=15 t")`. The same approach applies for other input arguments. See more in the examples.

Value

list. Emission factors in units 'g/km'.

See Also

Other Emission factor model: ef_brazil_cetesb(), ef_europe_emep(), ef_usa_emfac(), ef_usa_moves(), emi_europe_emep_wear()

Examples

```
temp_ef_br <- ef_brazil_cetesb(
                    pollutant = c("CO","PM10","CO2","CH4","NOx"),
                    veh_type = c("BUS_URBAN_D","BUS_MICRO_D"),
                    model_year = c(2015,2015),
                    as_list = TRUE
                    )

temp_ef_scaled <- ef_scaled_euro(
                    ef_local = temp_ef_br,
                    speed = units::set_units(1:100,"km/h"),
                    veh_type = c("Ubus Std 15 - 18 t","Ubus Midi <=15 t"),
                    euro = c("IV","IV"),
                    fuel = c("D","D"),
                    tech = c("SCR","SCR"),
                    pollutant = c("CO","PM10","CO2","CH4","NOx")
                    )
```

---

ef_usa_emfac *Running exhaust emissions factors for buses from United States (EMFAC2017 model)*

Description

Returns a vector or data.frame of emission factors for buses based on the California EMission Factor model (EMFAC2017). The model considers emission factors (EF) of urban buses in California (United States), considering different pollutants, years of reference, model year, fuel, speed ranges, type of regions, model version, and type of season. The gtfs2emis package currently supports EF only for "Statewide" region type, and "Annual" season. Specific data of these variables can be download at <https://arb.ca.gov/emfac/emissions-inventory>.

Usage

```
ef_usa_emfac(
  pollutant,
  reference_year = 2020,
  fuel = "D",
  model_year,
  speed,
  as_list = TRUE
)
```

Arguments

pollutant	character. Pollutants: "CH4" (Methane), "CO" (Carbon Monoxide), "CO2" (Carbon Dioxide), "N2O" (Nitrous Oxide), "NOx" (Oxides of Nitrogen), "PM10" (Primary Exhaust PM10 - Total), "PM25" (Primary Exhaust PM2.5 - Total), "SOX" (Oxides of Sulfur), "TOG" (Total Organic Gases), "ROG" (Reactive Organic Gases).
reference_year	numeric. Year of reference, in which the emissions inventory is estimated. Default is 2020. Values between 2015 - 2022.
fuel	character. Type of fuel: 'D' (Diesel),'G' (Gasoline), 'CNG' (Compressed Natural Gas). Default is 'D'.
model_year	Numeric; Model year of vehicle.
speed	Units. Speed in 'km/h'; Emission factor are returned in speed intervals: "5-10", "10-15", "15-20", "20-25", "25-30", "30-35", "35-40", "40-45", "45-50", "50-55", "55-60", "60-65", "65-70", "70-75", "75-80", "80-85", "85-90", ">90" mph (miles/h).
as_list	logical. If TRUE (default), the function returns the output in a list format. If FALSE, the output is returned in a data.table format.

Value

List or data.table. Emission factors in units 'g/km' by speed and model_year.

Source

https://arb.ca.gov/emfac/

See Also

Other Emission factor model: ef_brazil_cetesb(), ef_europe_emep(), ef_scaled_euro(), ef_usa_moves(), emi_europe_emep_wear()

Examples

```
df <- ef_usa_emfac(
        pollutant = c("CO","PM10"),
        reference_year = 2019,
        model_year = 2015,
        speed = units::set_units(10:100,"km/h"),
        fuel = "D",
        as_list = TRUE
        )
```

`ef_usa_emfac_db` *Emission factors from California Air Resources Board (EMFAC Model)*

Description

Running exhaust emissions factors from EMFAC2017 model. The model generates emission factors (EF) of urban buses in California (United States), considering different pollutants, years of reference, model year, fuel, speed ranges, type of regions, model version, and type of season. Currently, the package supports EFs only for "Statewide" region type, and "Annual" season. Specific data of other regions and seasons can be download at <https://arb.ca.gov/emfac/emissions-inventory>.

Usage

```
ef_usa_emfac_db
```

Format

A data.table with 79198 rows and 8 variables:

pol Character; Pollutants: CH4(Methane), CO(Carbon Monoxide), CO2(Carbon Dioxide), N2O(Nitrous Oxide), NOx(Oxides of Nitrogen), PM10(Primary Exhaust PM10 - Total), PM25(Primary Exhaust PM2.5 - Total), SOX(Oxides of Sulfur), TOG(Total Organic Gases), ROG (Reactive Organic Gases)

reference_year Numeric; Year of reference between 2010 - 2020

fuel character; Type of fuel: 'D' (Diesel),'G' (Gasoline),'CNG' (Compressed Natural Gas).

model_year Model year.

speed Units; Speed in 'km/h'; Emission factor are returned in speed intervals such as "5-10", "10-15", "15-20", "20-25", "25-30", "30-35", "35-40", "40-45", "45-50" "50-55", "55-60", "60-65", "65-70", "70-75", "75-80", "80-85", "85-90", ">90" mph (miles/h)

Details

The function returns the data in a data.frame format. The R script used to process the raw EMFAC database can be found in the repository <https://github.com/ipeaGIT/gtfs2emis/blob/master/data-raw/ef_usa_emfac_db.R>.

Source

https://arb.ca.gov/emfac/emissions-inventory

See Also

Other emission factor data: `ef_brazil_cetesb_db`, `ef_europe_emep_db`, `ef_usa_moves_db`

`ef_usa_moves` *Running exhaust emissions factors for buses from United States (MOVES3 model)*

Description

Returns a vector or data.frame of emission factors for urban buses based on values from the MOVES3 Model. Emission factor estimates are expressed in units 'g/km'.

Usage

```
ef_usa_moves(
  pollutant,
  model_year,
  reference_year = 2020,
  speed,
  fuel = "D",
  as_list = TRUE
)
```

Arguments

`pollutant`	character. Pollutants: "CH4" (Methane), "CO" (Carbon Monoxide), "CO2" (Carbon Dioxide), "EC" (Energy Consumption), "HONO" (Nitrous Acid), "N2O" (Nitrous Oxide), "NH3" (Ammonia), "NH4" (Ammonium), "NO" (Nitrogen Oxide), "NO2" (Nitrogen Dioxide), "NO3" (Nitrate), "NOx" (Oxides of Nitrogen), "PM10" (Primary Exhaust PM10 - Total), "PM25" (Primary Exhaust PM2.5 - Total), "SO2" (Sulfur Dioxide), "THC" (Total Gaseous Hydrocarbons), "TOG" (Total Organic Gases) and "VOC" (Volatile Organic Compounds)
`model_year`	numeric. Model year of vehicle.
`reference_year`	numeric. Year of reference, in which the emissions inventory is estimated. Default is 2020. Values between 2015 - 2022.
`speed`	units. Speed in 'km/h'. Emission factor are returned in speed intervals: "0-2.5", "2.5-7.5", "7.5-12.5", "12.5-17.5", "17.5-22.5", "22.5-27.5", "27.5-32.5", "32.5-37.5", "37.5-42.5", "42.5-47.5", "47.5-52.5", "52.5-57.5", "57.5-62.5", "62.5-67.5", "67.5-72.5", ">72.5" mph (miles/h).
`fuel`	character. Type of fuel: 'D' (Diesel),'G' (Gasoline), 'CNG' (Compressed Natural Gas). Default is 'D'.
`as_list`	logical. If `TRUE` (default), the function returns the output in a `list` format. If `FALSE`, the output is returned in a `data.table` format.

Details

Users can view the pre-processed database in `data(ef_usa_moves_db)` function.

Value

List. Emission factors in units 'g/km' by speed and model_year.

See Also

Other Emission factor model: `ef_brazil_cetesb()`, `ef_europe_emep()`, `ef_scaled_euro()`, `ef_usa_emfac()`, `emi_europe_emep_wear()`

Examples

```
df <- ef_usa_moves(
         pollutant = c("CO","PM10"),
         model_year = 2015,
         speed = units::set_units(10:100,"km/h"),
         reference_year = 2016,
         fuel = "D",
         as_list = TRUE
        )
```

`ef_usa_moves_db` *Emission factors from MOtor Vehicle Emission Simulator (MOVES) Data.frame of emission factors for buses based on values from the* R*hrefhttps://www.epa.gov/movesMOVES3 Model. Estimates expressed in units 'g/km'.*

Description

Emission factors from MOtor Vehicle Emission Simulator (MOVES)

Data.frame of emission factors for buses based on values from the MOVES3 Model. Estimates expressed in units 'g/km'.

Usage

```
ef_usa_moves_db
```

Format

A data.table:

pollutant character; Pollutants: CH4 (Methane), CO (Carbon Monoxide), CO2 (Carbon Dioxide), EC (Energy Consumption), HONO (Nitrous Acid), N2O (Nitrous Oxide), NH3 (Ammonia), NH4 (Ammonium), NO (Nitrogen Oxide), NO2 (Nitrogen Dioxide), NO3 (Nitrate), NOx (Oxides of Nitrogen), PM10 (Primary Exhaust PM10 - Total), PM25 (Primary Exhaust PM2.5 - Total), SO2 (Sulfur Dioxide), THC (Total Gaseous Hydrocarbons), TOG (Total Organic Gases) and VOC (Volatile Organic Compounds)

fuel_type character; Type of fuel: 'D' (Diesel),'G' (Gasoline),'CNG' (Compressed Natural Gas).

reference_year Numeric; Calendar Year between 2015 - 2022. Year in which the emissions inventory is estimated.

model_year numeric; Model year of vehicle.

lower_speed_interval units 'km/h'; Represents the lower value of the speed intervals; The speed intervals are " - 2.5", "2.5 - 7.5", "7.5 - 12.5" , "12.5 - 17.5", "17.5 - 22.5", "22.5 - 27.5","27.5 - 32.5","32.5 - 37.5" ,"37.5 - 42.5","42.5 - 47.5","47.5 - 52.5","52.5 - 57.5", "57.5 - 62.5" , "62.5 - 67.5", "67.5 - 72.5", and ">72.5" mph (miles/h).

upper_speed_interval units in km/h; Represents the upper value of the speed intervals. The speed intervals are analogous to `lower_speed_interval` above.

source_type character; Type of vehicle, which currently has only "Transit Bus".

id_speed integer;it caracterizes the types of vehicle speeds.

Source

https://www.epa.gov/moves

See Also

Other emission factor data: `ef_brazil_cetesb_db`, `ef_europe_emep_db`, `ef_usa_emfac_db`

---

emission_model *Emission model*

Description

Estimate hot-exhaust emissions of public transport systems. This function must be used together with `transport_model`.

Usage

```
emission_model(
  tp_model,
  ef_model,
  fleet_data,
  pollutant,
  reference_year = 2020,
  process = "hot_exhaust",
  heightfile = NULL,
  parallel = TRUE,
  ncores = NULL,
  output_path = NULL,
  continue = FALSE,
  quiet = TRUE
)
```

Arguments

tp_model	sf_linestring object or a character path the to sf_linestring objects. The tp_model is the output from transport_model, or the path in which the output files from the transport_model are saved.
ef_model	character. A string indicating the emission factor model to be used. Options include ef_usa_moves, ef_usa_emfac,ef_europe_emep, ,ef_brazil_cetesb, and ef_brazil_scaled_euro (scale ef_brazil_cetesb() based on ef_scaled_euro()).
fleet_data	data.frame. A data.frame with information the fleet characteristics. The required columns depend on the ef_model selection. See @examples for input.
pollutant	character. Vector with one or more pollutants to be estimated. Example: c("CO", "CO2", "PM10", "NOx"). See the documentation to check which pollutants are available for each emission factor model (ef_usa_moves, ef_usa_emfac, ef_europe_emep, or ef_brazil_cetesb).
reference_year	numeric. Year of reference considered to calculate the emissions inventory. Defaults to 2020. This argument is only required when the ef_model parameter is ef_usa_moves or ef_usa_emfac.
process	character; Emission process, classified in "hot_exhaust" (Default), and wear processes (identified as "tyre","brake" and/or "road" wear). Note that wear processes are only available when the ef_europe_emep is selected in the @param ef_model. Details on wear emissions are presented in emi_europe_emep_wear.
heightfile	character or raster data. The raster file with height data, or its filepath, used to estimate emissions considering the effect of street slope. This argument is used only when ef_brazil_scaled_euro or ef_europe_emep are selected. Default is NULL. Details are provided in slope_class_europe_emep.
parallel	logical. Decides whether the function should run in parallel. Defaults is TRUE.
ncores	integer. Number of cores to be used in parallel execution. This argument is ignored if parallel is FALSE. Default (NULL) selects the total number of available cores minus one.
output_path	character. File path where the function output is exported. If NULL (Default), the function returns the output to user.
continue	logical. Argument that can be used only with output_path When TRUE, it skips processing the shape identifiers that were already saved into files. It is useful to continue processing a GTFS file that was stopped for some reason. Default value is FALSE.
quiet	Logical; Display messages from the emissions or emission factor functions. Default is 'TRUE'.

Details

The fleet_data must be a data.frame organized according to the desired ef_model. The required columns is organized as follows (see @examples for real data usage).

- veh_type: character; Bus type, classified according to the @param ef_model . For ef_emep_europe, use "Ubus Midi <=15 t","Ubus Std 15 - 18 t", "Ubus Artic >18 t", "Coaches Std <=18 t" or "Coaches Artic >18 t"; For ef_usa_moves or ef_usa_emfac, use "BUS_URBAN_D";

For ef_brazil_cetesb, use "BUS_URBAN_D", "BUS_MICRO_D", "BUS_COACH_D" or "BUS_ARTIC_D".

- type_name_eu: character; Bus type, used only for @param ef_model ef_scaled_euro are selected. The classes can be "Ubus Midi <=15 t","Ubus Std 15 - 18 t", "Ubus Artic >18 t", "Coaches Std <=18 t" or "Coaches Artic >18 t".
- reference_year: character; Base year of the emission factor model input. Required only when ef_usa_moves or ef_usa_emfac are selected.
- tech: character; After treatment technology. This is required only when emep_europe is selected. Check ?ef_emep_europe for details.
- euro: character; Euro period of vehicle, classified in "Conventional", "I", "II", "III", "IV", "V", "VI", and "EEV". This is required only when ef_emep_europe is selected. Check ef_europe_emep for details.
- fuel: character; Required when ef_usa_moves, ef_usa_emfac and ef_europe_emep are selected.
- fleet_composition: Numeric. Scaled composition of fleet. In most cases, the user might not know which vehicles run on each specific routes. The composition is used to attribute a probability of a specific vehicle to circulate in the line. The probability sums one. Required for all emission factors selection. Users can check the gtfs2emis fleet data vignette, for more examples.

Based on the input height data, the function returns the slope class between two consecutive bus stop positions of a LineString Simple Feature (transport model object). The slope is given by the ratio between the height difference and network distance from two consecutive public transport stops. The function classifies the slope into one of the seven categories available on the European Environmental Agency (EEA) database, which is -0.06, -0.04,-0.02, 0.00, 0.02, 0.04, and 0.06.

Value

A list with emissions estimates or NULL with output files saved locally at output_path.

See Also

Other Core function: transport_model()

Examples

```
 if (requireNamespace("gtfstools", quietly=TRUE)) {

# read GTFS
gtfs_file <- system.file("extdata/bra_cur_gtfs.zip", package = "gtfs2emis")
gtfs <- gtfstools::read_gtfs(gtfs_file)

# keep a single trip_id to speed up this example
gtfs_small <- gtfstools::filter_by_trip_id(gtfs, trip_id ="4451136")

# run transport model
tp_model <- transport_model(gtfs_data = gtfs_small,
                            min_speed = 2,
                            max_speed = 80,
```

```
                            new_speed = 20,
                            spatial_resolution = 100,
                            parallel = FALSE)

# Example using Brazilian emission model and fleet
fleet_data_ef_cetesb <- data.frame(veh_type = "BUS_URBAN_D",
                                   model_year = 2010:2019,
                                   fuel = "D",
                                   fleet_composition = rep(0.1,10)
                                   )

emi_cetesb <- progressr::with_progress(emission_model(
                tp_model = tp_model,
                ef_model = "ef_brazil_cetesb",
                fleet_data = fleet_data_ef_cetesb,
                pollutant = c("CO","PM10","CO2","CH4","NOx")
                ))

# Example using European emission model and fleet
fleet_data_ef_europe <- data.frame(  veh_type = c("Ubus Midi <=15 t",
                                                  "Ubus Std 15 - 18 t",
                                                  "Ubus Artic >18 t")
                                   , euro = c("III","IV","V")
                                   , fuel = rep("D",3)
                                   , tech = c("-","SCR","SCR")
                                   , fleet_composition = c(0.4,0.5,0.1))

emi_emep <- progressr::with_progress(emission_model(tp_model = tp_model
                           , ef_model = "ef_europe_emep"
                           , fleet_data = fleet_data_ef_europe
                           , pollutant = c("PM10","NOx")))
emi_emep_wear <- progressr::with_progress(emission_model(tp_model = tp_model
                           , ef_model = "ef_europe_emep"
                           , fleet_data = fleet_data_ef_europe
                           , pollutant = "PM10"
                           , process = c("tyre","road","brake")))
raster_cur <- system.file("extdata/bra_cur-srtm.tif", package = "gtfs2emis")
emi_emep_slope <- progressr::with_progress(emission_model(tp_model = tp_model
                           , ef_model = "ef_europe_emep"
                           , fleet_data = fleet_data_ef_europe
                           , heightfile = raster_cur
                           , pollutant = c("PM10","NOx")))

# Example using US EMFAC emission model and fleet
fleet_data_ef_moves <- data.frame(  veh_type = "BUS_URBAN_D"
                                  , model_year = 2010:2019
                                  , fuel = "D"
                                  , reference_year = 2020
                                  , fleet_composition = rep(0.1,10))

fleet_data_ef_emfac <- data.frame(  veh_type =  "BUS_URBAN_D"
                                  , model_year = 2010:2019
                                  , fuel = "D"
```

```
                                    , reference_year = 2020
                                    , fleet_composition = rep(0.1,10))

# Example using US MOVES emission model and fleet
emi_moves <- emission_model(tp_model = tp_model
                            , ef_model = "ef_usa_moves"
                            , fleet_data = fleet_data_ef_moves
                            , pollutant = c("CO","PM10","CO2","CH4","NOx")
                            , reference_year = 2020)

emi_emfac <- emission_model(tp_model = tp_model
                            , ef_model = "ef_usa_emfac"
                            , fleet_data = fleet_data_ef_emfac
                            , pollutant = c("CO","PM10","CO2","CH4","NOx")
                            , reference_year = 2020)
}
```

`emis_grid` *Spatial aggregation of emission estimates into a grid*

Description

Aggregate emissions proportionally in an sf polygon grid, by performing an intersection operation between emissions data in `sf linestring` format and the input grid cells. User can also aggregate the emissions in the grid by time of the day.

Usage

```
emis_grid(
  emi_list,
  grid,
  time_resolution = "day",
  quiet = TRUE,
  aggregate = FALSE
)
```

Arguments

`emi_list`	list. A list containing the data of emissions 'emi' ("data.frame" class) and the transport model 'tp_model' ("sf" "data.frame" classes).
`grid`	Sf polygon. Grid cell data to allocate emissions.
`time_resolution`	character. Time resolution in which the emissions is aggregated. Options are 'hour', 'minute', or 'day (Default).
`quiet`	logical. User can print the total emissions before and after the intersection operation in order to check if the gridded emissions were estimated correctly. Default is 'TRUE'.
`aggregate`	logical. Aggregate emissions by pollutant. Default is `FALSE`.

Value

An "sf" "data.frame" object with emissions estimates per grid cell.

See Also

Other emission analysis: emis_summary(), emis_to_dt()

Examples

```
if (requireNamespace("gtfstools", quietly=TRUE)) {
library(sf)

# read GTFS
gtfs_file <- system.file("extdata/bra_cur_gtfs.zip", package = "gtfs2emis")
gtfs <- gtfstools::read_gtfs(gtfs_file)

# keep a single trip_id to speed up this example
gtfs_small <- gtfstools::filter_by_trip_id(gtfs, trip_id ="4451136")

# run transport model
tp_model <- transport_model(gtfs_data = gtfs_small,
                            spatial_resolution = 100,
                            parallel = FALSE)

# Fleet data, using Brazilian emission model and fleet
fleet_data_ef_cetesb <- data.frame(veh_type = "BUS_URBAN_D",
                                   model_year = 2010:2019,
                                   fuel = "D",
                                   fleet_composition = rep(0.1,10)
                                   )
# Emission model
emi_list <- emission_model(
                tp_model = tp_model,
                ef_model = "ef_brazil_cetesb",
                fleet_data = fleet_data_ef_cetesb,
                pollutant = c("CO","PM10","CO2","CH4","NOx")
                )

# create spatial grid
grid <- sf::st_make_grid(
  x = sf::st_make_valid(emi_list$tp_model)
  , cellsize = 0.25 / 200
  , crs= 4326
  , what = "polygons"
  , square = FALSE
  )

emi_grid <- emis_grid( emi_list,grid,'day')

plot(grid)
plot(emi_grid["PM10_2010"],add = TRUE)
plot(st_geometry(emi_list$tp_model), add = TRUE,col = "black")
```

```
}
```

`emis_summary` *Summarize emissions estimates*

Description

Summarize emissions estimates, aggregating emissions by pollutant, time of the day, vehicle.

Usage

```
emis_summary(
  emi_list,
  by = "pollutant",
  veh_vars = "veh_type",
  segment_vars = NULL,
  process_vars = "process"
)
```

Arguments

`emi_list`	list. Emission or emission factor list.
`by`	character. Emissions can be aggregated by 'time', 'vehicle', or simply 'pollutant' (Default).
`veh_vars`	character. data.frame names of 'emi_list' attributed to vehicle characteristics. Default is 'veh_type'.
`segment_vars`	character. data.frame names of 'emi_list' object attributed to the road segments. Default is NULL.
`process_vars`	character. data.frame names of 'emi_list' object attributed to the emission processes. Default is 'process'.

Value

`data.table` with pollutants units ('g') aggregated by pollutant, time, or vehicle type.

See Also

Other emission analysis: `emis_grid()`, `emis_to_dt()`

Examples

```
if (requireNamespace("gtfstools", quietly=TRUE)) {

# read GTFS
gtfs_file <- system.file("extdata/irl_dub_gtfs.zip", package = "gtfs2emis")
gtfs <- gtfstools::read_gtfs(gtfs_file)

# Keep a single trip
gtfs <- gtfstools::filter_by_trip_id(gtfs
                                     , trip_id = c('238.2.60-118-b12-1.59.I'
                                                   ,"7081.2.60-X27-b12-1.106.I"))
# Transport model
tp_model <- transport_model(gtfs_data = gtfs,
                            spatial_resolution = 100,
                            parallel = FALSE)

# fleet data
fleet_df <- read.csv(system.file("extdata/irl_dub_fleet.txt"
                                 , package = "gtfs2emis"))
# emission model
emi_list <- emission_model(tp_model = tp_model
                           , ef_model = "ef_europe_emep"
                           , fleet_data = fleet_df
                           , pollutant = c("CO2","PM10"))

# Aggregate total emissions by 'pollutant'
emis_summary(emi_list)

# by vehicle type
emis_summary(emi_list, by = "vehicle")

emis_summary(emi_list
             , by = "vehicle"
             , veh_vars = c("euro"))

emis_summary(emi_list
             , by = "vehicle"
             , veh_vars = c("fuel"))

emis_summary(emi_list
             , by = "vehicle"
             , veh_vars = c("veh_type","euro","tech","fuel"))

# by time of the day
emis_summary(emi_list
             , by = "time"
             , segment_vars = "slope")
}
```

emis_to_dt *Convert emission estimates from list to data.table format*

Description

Read emission estimates generated by the `emission_model` or from emission factor functions (e.g. `ef_brazil_cetesb`) and convert them into a `data.table` format.

Usage

```
emis_to_dt(
  emi_list,
  emi_vars = "emi",
  veh_vars = "veh_type",
  pol_vars = "pollutant",
  process_vars = "process",
  segment_vars = NULL
)
```

Arguments

`emi_list`	list. A list of emission estimates
`emi_vars`	character. data.frame names of 'emi_list' object attributed to emissions or emission factors. Default is 'emi'.
`veh_vars`	character. data.frame names of 'emi_list' object attributed to vehicle characteristics. Default is 'veh_type'.
`pol_vars`	character. data.frame names of 'emi_list' object attributed to pollutants. Default is 'pollutant'.
`process_vars`	character. data.frame names of 'emi_list' object attributed to the emission processes. Default is 'process'.
`segment_vars`	character. data.frame names of 'emi_list' object attributed to the road segments. Default is NULL.

Value

data.table.

See Also

Other emission analysis: `emis_grid()`, `emis_summary()`

Examples

```
 if (requireNamespace("gtfstools", quietly=TRUE)) {

# read GTFS
gtfs_file <- system.file("extdata/bra_cur_gtfs.zip", package = "gtfs2emis")
gtfs <- gtfstools::read_gtfs(gtfs_file)

# keep a single trip_id to speed up this example
gtfs_small <- gtfstools::filter_by_trip_id(gtfs, trip_id ="4451136")

# run transport model
tp_model <- transport_model(gtfs_data = gtfs_small,
                            min_speed = 2,
                            max_speed = 80,
                            new_speed = 20,
                            spatial_resolution = 100,
                            parallel = FALSE)

# Example using Brazilian emission model and fleet
fleet_data_ef_cetesb <- data.frame(veh_type = "BUS_URBAN_D",
                                   model_year = 2010:2019,
                                   fuel = "D",
                                   fleet_composition = rep(0.1,10)
                                   )

emi_list <- emission_model(
                tp_model = tp_model,
                ef_model = "ef_brazil_cetesb",
                fleet_data = fleet_data_ef_cetesb,
                pollutant = c("CO","PM10","CO2","CH4","NOx")
                )

# convert emission list to data.table
dt <- emis_to_dt(emi_list)
}
```

`emi_europe_emep_wear` *Emissions from road vehicle tyre, brake, and surface wear from the European Environment Agency (EMEP/EEA) model*

Description

Returns a list or data.table of emissions for urban buses based on Tier 2 of EMEP/EEA air pollutant emission inventory guidebooks (2019). The function concerns the emissions of particulate matter (PM), encompassing black carbon (BC) (1), which arises from distinct sources, namely, road vehicle tire and brake wear (NFR code 1.A.3.b.vi), and road surface wear (NFR code 1.A.3.b.vii). It is important to note that PM emissions exhaust from vehicle exhaust are excluded. The focus is on primary particles, which refer to those that are directly emitted due to surface wear, rather than those generated from the resuspension of previously deposited material. See more in @details.

Usage

```
emi_europe_emep_wear(
  dist,
  speed,
  pollutant,
  veh_type,
  fleet_composition,
  load = 0.5,
  process = "tyre",
  as_list = TRUE
)
```

Arguments

dist	units; Length of each link in 'km'.
speed	units; Speed in 'km/h'.
pollutant	character; Pollutant, classified in "TSP"(Total Suspended Particles), "PM10","PM2.5", "PM1.0","PM0.1". Please note that emissions factors for "PM1.0" and "PM0.1" are not available for road surface wear process.
veh_type	character; Bus type, classified in "Ubus Midi <=15 t", "Ubus Std 15 - 18 t", "Ubus Artic >18 t", "Coaches Std >18 t", or "Coaches Artic >18 t".
fleet_composition	vector; Fleet composition, which is a distribution of fleet based on frequency. If there is only one, 'fleet_composition' is 1.0.
load	numeric; Load ratio, classified in 0.0, 0.5 and 1.0. Default is 0.5.
process	character; Emission process sources, classified in "tyre","brake" and/or "road".
as_list	logical; Returns emission factors as a list, instead of data.table format. Default is TRUE.

Details

The following equation is employed to evaluate emissions originating from tyre and brake wear

TE(i) = dist x EF_tsp(j) x mf_s(i) x sc(speed),

where:

- TE(i) = total emissions of pollutant i (g),
- dist = distance driven by each vehicle (km),
- EF_tsp(j) = TSP mass emission factor for vehicles of category j (g/km),
- mf_s(i) = mass fraction of TSP that can be attributed to particle size class i,
- sc(speed) = correction factor for a mean vehicle travelling at a given speed (-)

Tyre

In the case of heavy-duty vehicles, the emission factor needs the incorporation of vehicle size, as determined by the number of axles, and load. These parameters are introduced into the equation as follows:

EF_tsp_tyre_hdv = 0.5 x N_axle x LCF_tyre x EF_tsp_tyre_pc

where

- EF_tsp_tyre_hdv = TSP emission factor for tyre wear from heavy-duty vehicles (g/km),
- N_axle = number of vehicle axles (-),
- LCF_t = a load correction factor for tyre wear (-),
- EF_tsp_tyre_pc = TSP emission factor for tyre wear from passenger car vehicles (g/km).

and LCF_tyre = 1.41 + (1.38 x LF),

where:

- LF = load factor (-), ranging from 0 for an empty bus to 1 for a fully laden one.

The function considers the following look-up table for number of vehicle axes:

vehicle class (j)	number of axes
Ubus Midi <=15 t	2
Ubus Std 15 - 18 t	2
Ubus Artic >18 t	3
Coaches Std <=18 t	2
Coaches Artic >18 t	3

The size distribution of tyre wear particles are given by:

particle size class (i)	mass fraction of TSP
TSP	1.000
PM10	0.600
PM2.5	0.420
PM1.0	0.060
PM0.1	0.048

Finally, the speed correction is:

sc_tyre(speed) = 1.39, when V < 40 km/h; sc_tyre(speed) = -0.00974 x speed + 1.78, when 40 <= speed <= 90 km/h; sc_tyre(speed) = 0.902, when speed > 90 km/h.

Brake

The heavy-duty vehicle emission factor is derived by modifying the passenger car emission factor to conform to experimental data obtained from heavy-duty vehicles.

EF_tsp_brake_hdv = 1.956 x LCF_brake x EF_tsp_brake_pc

where:

- EF_tsp_brake_hdv = heavy-duty vehicle emission factor for TSP,
- LCF_brake = load correction factor for brake wear,
- EF_tsp_brake_pc = passenger car emission factor for TSP,

and LCF_brake = 1 + (0.79 x LF),

where:

- LF = load factor (-), ranging from 0 for an empty bus to 1 for a fully laden one.

The size distribution of brake wear particles are given by:

particle size class (i)	mass fraction of TSP
TSP	1.000
PM10	0.980
PM2.5	0.390
PM1.0	0.100
PM0.1	0.080

Finally, the speed correction is:

sc_brake(speed) = 1.67, when V < 40 km/h; sc_brake(speed) = -0.0270 x speed + 2.75, when 40 <= speed <= 95 km/h; sc_brake(speed) = 0.185, when speed > 95 km/h.

Road Wear

Emissions are calculated according to the equation:

TE(i) = dist x EF_tsp_road(j) x mf_road

where:

- TE = total emissions of pollutant i (g),
- dist = total distance driven by vehicles in category j (km),
- EF_tsp_road = TSP mass emission factor from road wear for vehicles j (0.0760 g/km),
- mf_road = mass fraction of TSP that can be attributed to particle size class i (-).

The following table shows the size distribution of road surface wear particles

particle size class (i)	mass fraction of TSP
TSP	1.00
PM10	0.50
PM2.5	0.27

References

#' EMEP/EEA data and reports can be accessed in the following links:

- 2019 edition https://www.eea.europa.eu/publications/emep-eea-guidebook-2019/part-b-sectoral-guida 1-energy/1-a-combustion/1-a-3-b-vi/view.

Value

List. emission in units 'g' (list or a data.table).

See Also

Other Emission factor model: ef_brazil_cetesb(), ef_europe_emep(), ef_scaled_euro(), ef_usa_emfac(), ef_usa_moves()

Examples

```
emi_europe_emep_wear(dist = units::set_units(1,"km"),
                     speed =  units::set_units(30,"km/h"),
                     pollutant = c("PM10","TSP","PM2.5"),
                     veh_type = c("Ubus Std 15 - 18 t","Ubus Artic >18 t"),
                     fleet_composition = c(0.5,0.5),
                     load = 0.5,
                     process = c("brake","tyre","road"),
                     as_list = TRUE)
```

---

slope_class_europe_emep

Add slope class into the transport model (LineString Simple Feature object)

Description

Based on the input height data, the function returns the slope class between two consecutive bus stop positions of a LineString Simple Feature (transport model object). The slope is given by the ratio between the height difference and network distance from two consecutive public transport stops. The function classifies the slope into one of the seven categories available on the European Environmental Agency (EEA) database, which is -0.06, -0.04,-0.02, 0.00, 0.02, 0.04, and 0.06. The classifications is described in @details .

Usage

```
slope_class_europe_emep(tp_model, heightfile, keep = FALSE)
```

Arguments

tp_model	LineString Simple Feature; transport model output.
heightfile	character or raster data; The raster file with height data, or its filepath.
keep	A logical. Whether the columns related height and slope to the consecutive bus stops should be kept or dropped (defaults to FALSE, which keeps only the slope classification).

Value

The transport model with slope information.

Slopes classification:

| **slope interval** | **slope class** | | slope <= -0.070 | -0.06 | | slope > -0.070 & slope <= -0.050 | -0.06 | | slope > -0.050 & slope <= -0.030 | -0.04 | | slope > -0.030 & slope <= -0.010 | -0.02 | | slope > -0.010 & slope <= +0.010 | +0.00 | | slope > +0.010 & slope <= +0.030 | +0.02 | | slope > +0.030 & slope <= +0.050 | +0.04 | | slope > +0.050 & slope <= +0.070 | +0.06 | | slope > +0.070 | -0.06 |

Examples

```
 if (requireNamespace("gtfstools", quietly=TRUE)) {
gtfs_file <- system.file("extdata/bra_cur_gtfs.zip", package = "gtfs2emis")
gtfs <- gtfstools::read_gtfs(gtfs_file)

# keep a single trip_id to speed up this example
gtfs_small <- gtfstools::filter_by_trip_id(gtfs, trip_id ="4451136")

# run transport model
tp_model <- transport_model(gtfs_data = gtfs_small,
                            min_speed = 2,
                            max_speed = 80,
                            new_speed = 20,
                            spatial_resolution = 100,
                            parallel = FALSE)

# read raster file
raster_cur <- system.file("extdata/bra_cur-srtm.tif", package = "gtfs2emis")

tp_model_slope <- slope_class_europe_emep(tp_model,raster_cur)
}
```

---

transport_model *Transport model*

Description

This function converts a public transport data set in GTFS format into a GPS-like table with the space-time positions and speeds of public transport vehicles. The function also allow users to set the spatial resolution of the output and to adjust the speed of public transport vehicles given a `min_speed` and `max_speed` range.

Usage

```
transport_model(
  gtfs_data,
  min_speed = 2,
  max_speed = 80,
  new_speed = NULL,
  parallel = TRUE,
```

```
  ncores = NULL,
  spatial_resolution = 100,
  output_path = NULL,
  continue = FALSE
)
```

Arguments

gtfs_data	A path to a GTFS file or a GTFS data organized as a list of data.tables created with gtfstools::read_gtfs().
min_speed	numeric (in km/h) or a speed units value. Minimum speed to be considered as valid. Values below minimum speed will be updated according to the new_speed parameter, which can affect the arrival and departure times of vehicles at transit stops. Defaults to 2 km/h.
max_speed	numeric (in km/h) or a speed units value. Maximum speed to be considered as valid. Values above maximum speed will be updated according to the new_speed parameter, which can affect the arrival and departure times of vehicles at transit stops. Defaults to 80 km/h.
new_speed	numeric (in km/h) or a speed units value. Speed value used to replace the speeds that fall outside the min_speed and max_speed range or which are missing from the GTFS input. When new_speed = NULL (the default), the function uses the average speed of the entire GTFS data feed.
parallel	logical. Decides whether the function should run in parallel. Defaults is TRUE.
ncores	integer. Number of cores to be used in parallel execution. This argument is ignored if parallel is FALSE. Default (NULL) selects the total number of available cores minus one.
spatial_resolution	The spatial resolution in meters. Defaults to 100. The function only creates points in order to guarantee that the minimum distance between two consecutive points will be at most the spatial_resolution value. If a given GTFS shape_id has two consecutive points with a distance smaller than the spatial resolution, the algorithm will not remove such points.
output_path	character. A directory path. If NULL (Default), the function returns the output. If the user passes a valid passed, the output will be saved in the output_path dir. Note that that the output of each public transport shape_id is saved separately in different files. Setting an output_path is recommended when working with large public transport system because the output of the function can be significantly large.
continue	logical. Argument that can be used only with output_path When TRUE, it skips processing the shape identifiers that were already saved into files. It is useful to continue processing a GTFS file that was stopped for some reason. Default value is FALSE.

Value

A data.table sf_linestring object or NULL.

See Also

Other Core function: emission_model()

Examples

```
if (requireNamespace("gtfstools", quietly=TRUE)) {

# read GTFS
gtfs_file <- system.file("extdata/bra_cur_gtfs.zip", package = "gtfs2emis")
gtfs <- gtfstools::read_gtfs(gtfs_file)

# keep a single trip_id to speed up this example
gtfs_small <- gtfstools::filter_by_trip_id(gtfs, trip_id ="4451136")

# run transport model
tp_model <- transport_model(gtfs_data = gtfs_small,
                            min_speed = 2,
                            max_speed = 80,
                            new_speed = 20,
                            spatial_resolution = 100,
                            parallel = FALSE)
  }
```

Index