Package ‘vein’

April 29, 2023

Type Package
Title Vehicular Emissions Inventories
Version 1.0.2
Date 2023-04-28


Before using VEIN you need to know the vehicular composition of your study area, in other words, the combination of type of vehicles, size and fuel of the fleet. Then, it is recommended to start with the project to download a template to create a structure of directories and scripts.

License MIT + file LICENSE

URL https://github.com/atmoschem/vein

BugReports https://github.com/atmoschem/vein/issues
LazyData no
Depends R (>= 3.5.0)
Imports sf (>= 1.0.1), data.table, units, graphics, stats, dotCall64, cptcity, grDevices
Suggests knitr, rmarkdown, testthat
RoxygenNote 7.2.3
Encoding UTF-8

NeedsCompilation yes

Config/testthat/parallel true

VignetteBuilder knitr

Author Sergio Ibarra-Espinosa [aut, cre]
(Daniel Schuch [ctb] (<https://orcid.org/0000-0001-5977-4519>),
Joao Bazzo [ctb] (<https://orcid.org/0000-0002-7371-1116>),
Mario Gavidia-Calderón [ctb] (<https://orcid.org/0000-0003-4536-5006>))
Maintainer  Sergio Ibarra-Espinosa <zergioibarra@gmail.com>
Repository  CRAN
Date/Publication  2023-04-29 19:20:02 UTC

R topics documented:

- addscale ................................................. 4
- add_lkm ................................................. 5
- add_miles .............................................. 6
- add_polid ............................................. 6
- adt ..................................................... 7
- age ...................................................... 9
- age_hdv ............................................... 10
- age_ldv ............................................... 12
- age_moto .............................................. 13
- aw ..................................................... 15
- celsius ............................................... 16
- check_nt .............................................. 17
- cold_mileage ........................................ 17
- colplot ............................................... 18
- decoder ............................................... 20
- dmonth ............................................... 21
- ef_cetesb ............................................. 21
- ef_china .............................................. 25
- ef_china_det ......................................... 29
- ef_china_h ........................................... 30
- ef_china_hu .......................................... 31
- ef_china_long ........................................ 32
- ef_china_s ........................................... 33
- ef_china_speed ....................................... 34
- ef_china_te ........................................... 35
- ef_china_th ........................................... 36
- ef_eea ............................................... 37
- ef_emfac ............................................. 38
- ef_evap ............................................... 39
- ef_fun ................................................ 41
- ef_hdv_scaled ........................................ 42
- ef_hdv_speed .......................................... 43
- ef_im .................................................. 45
- ef_ldv_cold .......................................... 46
- ef_ldv_cold_list ..................................... 48
- ef_ldv_scaled ........................................ 49
- ef_ldv_speed .......................................... 51
- ef_local .............................................. 55
- ef_nitro ............................................. 56
- ef_wear .............................................. 58
- ef_whe ............................................... 59
R topics documented:

emis .............................................................. 60
EmissionFactors .................................................. 64
EmissionFactorsList ............................................. 66
Emissions .......................................................... 67
EmissionsArray .................................................... 69
emis_chem ......................................................... 70
emis_chem2 ......................................................... 71
emis_china ......................................................... 73
emis_cold ......................................................... 75
demis_cold_td ..................................................... 77
demis_det ........................................................ 80
demis_dist ........................................................ 82
demis_emfac ....................................................... 83
demis_evap ......................................................... 84
demis_evap2 ....................................................... 85
demis_grid ......................................................... 88
demis_hot_td ....................................................... 90
demis_long ......................................................... 95
demis_merge ....................................................... 96
demis_order ....................................................... 98
demis_paved ....................................................... 100
demis_post ......................................................... 101
demis_to_streets ................................................ 104
demis_wear ....................................................... 105
fe2015 ............................................................ 106
fkm ............................................................... 107
fuel_corr .......................................................... 108
get_project ....................................................... 109
GriddedEmissionsArray ......................................... 110
grid_emis ......................................................... 112
invcop ............................................................ 114
inventory .......................................................... 115
long_to_wide ..................................................... 116
make_grid ........................................................ 117
movess_ef ......................................................... 118
movess_rpd ....................................................... 119
movess_rpdy ..................................................... 121
movess_rpdy_meta .............................................. 122
movess_rpdy_ssf .............................................. 123
movess_rpsy_meta .............................................. 125
movess_rpsy_ssf .............................................. 126
movess_speed .................................................... 127
my_age ............................................................ 128
net ................................................................. 129
netspeed ........................................................... 130
pc_cold ........................................................... 131
pc_profile ......................................................... 132
pollutants .......................................................... 132
addscale

function to add a scale to a image plot

Description

method to plot a scale in image plot.

Usage

```r
addscale(
  z,
  zlim = range(z, na.rm = TRUE),
  col = grDevices::heat.colors(12),
  breaks = pretty(zlim),
  horiz = TRUE,
  ylim = NULL,
  xlim = NULL,
  ...
)
```

Arguments

- `z`: matrix or vector
- `zlim`: z limit
- `col`: color
- `breaks`: interval for the tickmarks
- `horiz`: TRUE (default) to a horizontal scale
- `ylim`: y limit
- `xlim`: x limit
- `...`: other arguments to plot
add_lkm

Examples

## Not run:
mat <- matrix(1:100, ncol = 10, byrow = F)
cor <- grDevices::heat.colors(100)
image(mat, axe = FALSE, main = "numbers from 1 to 100", col = cor)
axis(2)
addscale(mat, col = cor)

## End(Not run)

---

add_lkm

Construction function to add unit km

Description

add_lkm just add unit 'km' to different R objects

Usage

add_lkm(x)

Arguments

x

Object with class "data.frame", "matrix", "numeric" or "integer"

Value

Objects of class "data.frame" or "units"

See Also

Other Add distance unitts: add_miles()

Examples

## Not run:
a <- add_lkm(rnorm(100)*10)
plot(a)
b <- add_lkm(matrix(rnorm(100)*10, ncol = 10))
print(head(b))

## End(Not run)
add_miles

Construction function to add unit miles

Description
add_miles just add 'miles' to different R objects

Usage
add_miles(x)

Arguments

x Object with class "data.frame", "matrix", "numeric" or "integer"

Value
Objects of class "data.frame" or "units"

See Also
Other Add distance units: add_lkm()

Examples
## Not run:
a <- add_miles(rnorm(100)*10)
plot(a)
b <- add_miles(matrix(rnorm(100)*10, ncol = 10))
print(head(b))
## End(Not run)

add_polid

Add polygon id to lines road network

Description
Sometimes you need to add polygon id into your streets road network. add_polid add add_polid id into your road network cropping your network by.

For instance, you have open street maps road network the you have the polygon of your regions. This function adds the id of your polygon as a new column in the streets network.

Usage
add_polid(polyg, street, by)
Arguments

polyg  sf object POLYGON or sp
street  streets road network class sf or sp
by     Character indicating the column with the id in polyg

See Also

emis_to_streets

Examples

## Not run:
data(net)
nets <- sf::st_as_sf(net)
bb <- sf::st_as_sf(sf::st_as_sfc(sf::st_bbox(nets)))
bb$id <- "a"
a <- add_polid(polyg = bb, street = nets, by = "id")
## End(Not run)

adt  Average daily traffic (ADT) from hourly traffic data.

Description

adt calculates ADT based on hourly traffic data.

Usage

adt(
    pc,
    lcv,
    hgv,
    bus,
    mc,
    p_pc,
    p_lcv,
    p_hgv,
    p_bus,
    p_mc,
    feq_pc = 1,
    feq_lcv = 1.5,
    feq_hgv = 2,
    feq_bus = 2,
    feq_mc = 0.5
)
Arguments

pc    numeric vector for passenger cars
lcv   numeric vector for light commercial vehicles
hgv   numeric vector for heavy good vehicles or trucks
bus   numeric vector for bus
mc    numeric vector for motorcycles
p_pc  data-frame profile for passenger cars, 24 hours only.
p_lcv data-frame profile for light commercial vehicles, 24 hours only.
p_hgv data-frame profile for heavy good vehicles or trucks, 24 hours only.
p_bus data-frame profile for bus, 24 hours only.
p_mc  data-frame profile for motorcycles, 24 hours only.
feq_pc Numeric, factor equivalence
feq_lcv Numeric, factor equivalence
feq_hgv Numeric, factor equivalence
feq_bus Numeric, factor equivalence
feq_mc Numeric, factor equivalence

Value
	numeric vector of total volume of traffic per link as ADT

Examples

## Not run:
data(net)
data(pc_profile)
p1 <- pc_profile[1]
adt1 <- adt(pc = net$ldv*0.75,
            lcv = net$ldv*0.1,
            hgv = net$hdv,
            bus = net$hdv*0.1,
            mc = net$ldv*0.15,
            p_pc = p1,
            p_lcv = p1,
            p_hgv = p1,
            p_bus = p1,
            p_mc = p1)
head(adt1)

## End(Not run)
Applies a survival rate to numeric new vehicles

Description

age returns survived vehicles

Usage

age(x, type = "weibull", a = 14.46, b = 4.79, agemax, verbose = FALSE)

Arguments

x Numeric; numerical vector of sales or registrations for each year
type Character; any of "gompertz", "double_logistic", "weibull" and "weibull2"
a Numeric; parameter of survival equation
b Numeric; parameter of survival equation
agemax Integer; age of oldest vehicles for that category
verbose Logical; message with average age and total number of vehicles regions or streets.

Value
dataframe of age distribution of vehicles

Note

The functions age* produce distribution of the circulating fleet by age of use. The order of using these functions is:

1. If you know the distribution of the vehicles by age of use, use: my_age
2. If you know the sales of vehicles, or the registry of new vehicles, use age to apply a survival function.
3. If you know the theoretical shape of the circulating fleet and you can use age_ldv, age_hdv or age_moto. For instance, you don't know the sales or registry of vehicles, but somehow you know the shape of this curve.
4. You can use/merge/transform/dapt any of these functions.


double_logistic: 1/(1 + \exp(a*(\text{time} + b))) + 1/(1 + \exp(a*(\text{time} - b)))\), defaults PC: b = 21, a = 0.19, LCV: b = 15.3, a = 0.17, HGV: b = 17, a = 0.1, BUS: b = 19.1, a = 0.16 MCT (2006). de Gases de Efeito Estufa-Emissoes de Gases de Efeito Estufa por Fontes Moveis, no Setor Energético. Ministerio da Ciencia e Tecnologia.
weibull: \( \exp(-\text{time}/a)^b \), defaults PC: \( b = 4.79, a = 14.46 \), Taxi: \( b = +\infty, a = 5 \), Government and business: \( b = 5.33, a = 13.11 \) Non-operating vehicles: \( b = 5.08, a = 11.53 \) Bus: \( b = +\infty, a = 9 \), non-transit bus: \( b = +\infty, a = 5.5 \) Heavy HGV: \( b = 5.58, a = 12.8 \), Medium HGV: \( b = 5.58, a = 10.09 \), Light HGV: \( b = 5.58, a = 8.02 \) Hao, H., Wang, H., Ouyang, M., & Cheng, F. (2011). Vehicle survival patterns in China. Science China Technological Sciences, 54(3), 625-629.


See Also

Other age: \code{age_hdv()}, \code{age_ldv()}, \code{age_moto()}

Examples

```r
## Not run:
vehLIA <- rep(1, 25)
PV_Minia <- age(x = vehLIA)
PV_Minib <- age(x = vehLIA, type = "weibull2", b = 11, a = 26)
PV_Minic <- age(x = vehLIA, type = "double_logistic", b = 21, a = 0.19)
PV_Minid <- age(x = vehLIA, type = "gompertz", b = -0.137, a = 1.798)
dff <- data.frame(PV_Minia, PV_Minib, PV_Minic, PV_Minid)
colplot(dff)
## End(Not run)
```

---

**age_hdv**

Returns amount of vehicles at each age

Description

age_hdv returns amount of vehicles at each age

Usage

```r
age_hdv(
  x,
  name = "age",
  a = 0.2,
  b = 17,
  agemin = 1,
  agemax = 50,
  k = 1,
  bystreet = F,
  net,
  verbose = FALSE,
  namerows,
)```


Arguments

x Numeric; numerical vector of vehicles with length equal to lines features of road network
name Character; of vehicle assigned to columns of dataframe
a Numeric; parameter of survival equation
b Numeric; parameter of survival equation
agemin Integer; age of newest vehicles for that category
agemax Integer; age of oldest vehicles for that category
k Numeric; multiplication factor. If its length is > 1, it must match the length of x
bystreet Logical; when TRUE it is expecting that ’a’ and ’b’ are numeric vectors with length equal to x
net SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
verbose Logical; message with average age and total number of vehicles
namerows Any vector to be change row.names. For instance, name of regions or streets.
time Character to be the time units as denominator, eg "1/h"

Value
dataframe of age distribution of vehicles at each street

Note

The functions age* produce distribution of the circulating fleet by age of use. The order of using these functions is:
1. If you know the distribution of the vehicles by age of use, use: my_age
2. If you know the sales of vehicles, or the registry of new vehicles, use age to apply a survival function. 3. If you know the theoretical shape of the circulating fleet and you can use age_ldv, age_hdv or age_moto. For instance, you don’t know the sales or registry of vehicles, but somehow you know the shape of this curve. 4. You can use/merge/transform/adapt any of these functions.

See Also

Other age: age_ldv(), age_moto(), age()

Examples

## Not run:
data(net)
LT_B5 <- age_hdv(x = net$hdv, name = "LT_B5")
plot(LT_B5)
LT_B5 <- age_hdv(x = net$hdv, name = "LT_B5", net = net)
plot(LT_B5)

## End(Not run)
age_ldv

Returns amount of vehicles at each age

Description

age_ldv returns amount of vehicles at each age

Usage

age_ldv(
  x,
  name = "age",
  a = 1.698,
  b = -0.2,
  agemin = 1,
  agemax = 50,
  k = 1,
  bystreet = F,
  net,
  verbose = FALSE,
  namerows,
  time
)

Arguments

x Numeric; numerical vector of vehicles with length equal to lines features of road network
name Character; of vehicle assigned to columns of dataframe
a Numeric; parameter of survival equation
b Numeric; parameter of survival equation
agemin Integer; age of newest vehicles for that category
agemax Integer; age of oldest vehicles for that category
k Numeric; multiplication factor. If its length is > 1, it must match the length of x
bystreet Logical; when TRUE it is expecting that 'a' and 'b' are numeric vectors with length equal to x
net SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
verbose Logical; message with average age and total number of vehicles
namerows Any vector to be change row.names. For instance, name of regions or streets.
time Character to be the time units as denominator, e.g. "1/h"

Value
dataframe of age distribution of vehicles
Note

The functions age* produce distribution of the circulating fleet by age of use. The order of using these functions is:

1. If you know the distribution of the vehicles by age of use, use: my_age
2. If you know the sales of vehicles, or the registry of new vehicles, use age to apply a survival function.
3. If you know the theoretical shape of the circulating fleet and you can use age_ldv, age_hdv or age_moto. For instance, you dont know the sales or registry of vehicles, but somehow you know the shape of this curve.
4. You can use/merge/transform/adapt any of these functions.

It consists in a Gompertz equation with default parameters from 1 national emissions inventory for greenhouse gases in Brazil, MCT 2006

See Also

Other age: age_hdv(), age_moto(), age()

Examples

```r
## Not run:
data(net)
PC_E25_1400 <- age_ldv(x = net$ldv, name = "PC_E25_1400")
plot(PC_E25_1400)
PC_E25_1400 <- age_ldv(x = net$ldv, name = "PC_E25_1400", net = net)
plot(PC_E25_1400)
## End(Not run)
```

---

**age_moto**

*Returns amount of vehicles at each age*

Description

age_moto returns amount of vehicles at each age

Usage

```r
age_moto(
  x,
  name = "age",
  a = 0.2,
  b = 17,
  agemin = 1,
  agemax = 50,
  k = 1,
  bystreet = FALSE,
  net,
  verbose = FALSE,
  namerows,
)```
Arguments

- **x**
  - Numeric; numerical vector of vehicles with length equal to lines features of road network
- **name**
  - Character; name of vehicle to be assigned to columns of dataframe
- **a**
  - Numeric; parameter of survival equation
- **b**
  - Numeric; parameter of survival equation
- **agemin**
  - Integer; age of newest vehicles for that category
- **agemax**
  - Integer; age of oldest vehicles for that category
- **k**
  - Numeric; multiplication factor. If its length is > 1, it must match the length of x
- **bystreet**
  - Logical; when TRUE it is expecting that ‘a’ and ‘b’ are numeric vectors with length equal to x
- **net**
  - SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
- **verbose**
  - Logical; message with average age and total number of vehicles
- **namerows**
  - Any vector to be change row.names. For instance, name of regions or streets.
- **time**
  - Character to be the time units as denominator, eg "1/h"

Value

dataframe of age distribution of vehicles

Note

The functions age* produce distribution of the circulating fleet by age of use. The order of using these functions is:
1. If you know the distribution of the vehicles by age of use, use: `my_age`
2. If you know the sales of vehicles, or the registry of new vehicles, use `age` to apply a survival function.
3. If you know the theoretical shape of the circulating fleet and you can use `age_ldv`, `age_hdv` or `age_moto`.
   For instance, you don’t know the sales or registry of vehicles, but somehow you know the shape of this curve.
4. You can use/merge/transform/adapt any of these functions.

See Also

Other age: `age_hdv()`, `age_ldv()`, `age()`

Examples

```r
## Not run:
data(net)
MOTO_E25_500 <- age_moto(x = net$ldv, name = "M_E25_500", k = 0.4)
plot(MOTO_E25_500)
MOTO_E25_500 <- age_moto(x = net$ldv, name = "M_E25_500", k = 0.4, net = net)
plot(MOTO_E25_500)

## End(Not run)
```
**aw**  
*Average Weight for hourly traffic data.*

### Description

*aw* average weight form traffic.

### Usage

```r
aw(
  pc,
  lcv,
  hgv,
  bus,
  mc,
  p_pc,
  p_lcv,
  p_hgv,
  p_bus,
  p_mc,
  w_pc = 1,
  w_lcv = 3.5,
  w_hgv = 20,
  w_bus = 20,
  w_mc = 0.5,
  net
)
```

### Arguments

- **pc**: numeric vector for passenger cars
- **lcv**: numeric vector for light commercial vehicles
- **hgv**: numeric vector for heavy good vehicles or trucks
- **bus**: numeric vector for bus
- **mc**: numeric vector for motorcycles
- **p_pc**: data-frame profile for passenger cars, 24 hours only.
- **p_lcv**: data-frame profile for light commercial vehicles, 24 hours only.
- **p_hgv**: data-frame profile for heavy good vehicles or trucks, 24 hours only.
- **p_bus**: data-frame profile for bus, 24 hours only.
- **p_mc**: data-frame profile for motorcycles, 24 hours only.
- **w_pc**: Numeric, factor equivalence
- **w_lcv**: Numeric, factor equivalence
- **w_hgv**: Numeric, factor equivalence
celsius

Construction function for Celsius temperature

description

celsius just add unit celsius to different R objects

Usage

celsius(x)

Arguments

x Object with class "data.frame", "matrix", "numeric" or "integer"

Value

Objects of class "data.frame" or "units"
check_nt

Examples
{
  a <- celsius(rnorm(100)*10)
  plot(a)
  b <- celsius(matrix(rnorm(100)*10, ncol = 10))
  print(head(b))
}

check_nt (Check the max number of threads)

Description
get_threads check the number of threads in this machine

Usage
check_nt()

Value
Integer with the max number of threads

Examples
{
  check_nt()
}

cold_mileage (Fraction of mileage driven with a cold engine or catalizer below normal temperature)

Description
This function depends length of trip and on ambient temperature. From the guidelines EMEP/EEA
air pollutant emission inventory guidebook http://www.eea.europa.eu/themes/air/emep-eea-air-pollutant-
emission-inventory-guidebook

Usage
cold_mileage(ltrip, ta)
Arguments

- **ltrip**: Numeric; Length of trip. It must be in ‘units’ km.
- **ta**: Numeric or data.frame; average monthly temperature Celsius. If it is a data.frame, it is convenient that each column is each month.

Note

This function is set so that values varies between 0 and 1.

Examples

```r
## Not run:
lkm <- units::set_units(1:10, km)
ta <- celsius(matrix(0:9, ncol = 12, nrow = 10))
a <- cold_mileage(lkm, ta)
colplot(a)
## End(Not run)
```

---

**colplot**  
*Function to plot columns of data.frames*

Description

**colplot** plots columns of data.frame

Usage

```r
colplot(
  df,
  cols = names(df),
  xlab = "",
  ylab = "",
  xlim = c(1, nrow(df)),
  ylim = range(unlist(df[[cols]]), na.rm = TRUE),
  main = NULL,
  theme = "black",
  col = cptcity::cpt(pal = cptcity::find_cpt("pastel")[4], n = length(names(df))),
  type = "b",
  lwd = 2,
  pch = 1:ncol(df),
  familyfont = "",
  ...
)
```
Arguments

- **df** : data.frame.
- **cols** : Character, columns of data.frame.
- **xlab** : a label for the x axis, defaults to a description of x.
- **ylab** : a label for the y axis, defaults to a description of y.
- **xlim** : x limits
- **ylim** : y limits
- **main** : Character, a main title for the plot, see also **title**.
- **theme** : Character; "black", "dark", "clean", "ink"
- **col** : The colors for lines and points. Multiple colors can be specified so that each point can be given its own color. If there are fewer colors than points they are recycled in the standard fashion. Default are cpitcity colour palette "ks18_pastels"
- **type** : 1-character string giving the type of plot desired. The following values are possible, for details, see plot: "p" for points, "l" for lines, "b" for both points and lines, "c" for empty points joined by lines, "o" for overplotted points and lines, "s" and "S" for stair steps and "h" for histogram-like vertical lines. Finally, "n" does not produce any points or lines.
- **lwd** : a vector of line widths, see **par**.
- **pch** : plotting ‘character’, i.e., symbol to use. This can either be a single character or an integer code for one of a set of graphics symbols. The full set of S symbols is available with pch = 0:18, see the examples below. (NB: R uses circles instead of the octagons used in S.). Value pch = "." (equivalently pch = 46) is handled specially. It is a rectangle of side 0.01 inch (scaled by cex). In addition, if cex = 1 (the default), each side is at least one pixel (1/72 inch on the pdf, postscript and xfig devices). For other text symbols, cex = 1 corresponds to the default fontsize of the device, often specified by an argument pointsize. For pch in 0:25 the default size is about 75 the character height (see par("cin")).
- **familyfont** : "Character" to specify font, default is "", options "serif", "sans", "mono" or more according device

Value

- a nice plot

Note

This plot shows values > 0 by default. To plot all values, use all_values = TRUE

See Also

- **par**

Other helpers: **dmonth()**, **to_latex()**, **wide_to_long()**
Examples

```r
## Not run:
a <- ef_cetesb("CO", c("PC_G", "PC_FE", "PC_FG", "PC_E"), agemax = 20)
colplot(df = a, ylab = "CO [g/km]", theme = "dark", type = "b")
colplot(df = a, ylab = "CO [g/km]", theme = "dark", pch = NULL, type = "b")
colplot(df = a, ylab = "CO [g/km]", theme = "clean", type = "b")
colplot(df = a, ylab = "CO [g/km]", theme = "clean", pch = NULL, type = "b")
#colplot(df = a, cols = "PC_FG", main = "EF", ylab = "CO [g/km]"
#colplot(df = a, ylab = "CO [g/km]", theme = "clean")

## End(Not run)
```

decoder  

Description data.frame for MOVES

Description

A data.frame descriptors to use MOVES functions

Usage

```r
data(decoder)
```

Format

A data frame with 69 rows and 4 columns:

- **CategoryField**: dayID, sourceTypID, roadTypeID, pollutantID and procesID
- **pollutantID**: Associated number
- **Description**: Associated description
- **V4**: pollutants

Source

US/EPA MOVES
**dmonth**

| dmonth | Number of days of the month |

**Description**

`ef_ldv_speed` return the number of days of the month

**Usage**

```r
dmonth(year, month)
```

**Arguments**

- `year` Numeric
- `month` Numeric

**Value**

days of the month

**See Also**

Other helpers: `colplot()`, `to_latex()`, `wide_to_long()`

**Examples**

```r
## Not run:
dmonth(2022, 1)
## End(Not run)
```

---

**ef_cetesb**

Emissions factors for Environment Company of Sao Paulo, Brazil (CETESB)

**Description**

`ef_cetesb` returns a vector or data.frame of Brazilian emission factors.
Usage

ef_cetesb(
  p,
  veh,
  year = 2017,
  agemax = 40,
  scale = "default",
  sppm,
  full = FALSE,
  efinput,
  verbose = FALSE,
  csv
)

Arguments

  p  Character;
      Pollutants: "CO", "HC", "NMHC", "CH4", "NOx", "CO2", "RCHO" (aldehydes + formaldehyde), "ETOH", "PM", "N2O", "KML", "FC", "NO2", "NO", "NH3", "gD/KWH", "gCO2/KWH", "RCHO_0km" (aldehydes + formaldehyde), "PM25RES", "PM10RES", "CO_0km", "HC_0km", "NMHC_0km", "NOx_0km", "NO2_0km", "NO_0km", "RCHO_0km" and "ETOH_0km", "FS" (fuel sales) (g/km). If scale = "tunnel" is used, there is also "ALD" for aldehydes and "HCHO" for formaldehydes. Evaporative emissions at average temperature ranges: "D_20_35", "S_20_35", "R_20_35", "D_10_25", "S_10_25", "R_10_25", "D_0_15", "S_0_15" and "R_0_15" where D means diurnal (g/day), S hot/warm soak (g/trip) and R hot/warm running losses (g/trip). The deteriorated emission factors are calculated inside this function.


  year  Numeric; Filter the emission factor to start from a specific base year. If project is 'constant' values above 2017 and below 1980 will be repeated

  agemax  Integer; age of oldest vehicles for that category

  scale  Character; values "default", "tunnel" or "tunnel2018". If "tunnel", emission factors are scaled to represent EF measurements in tunnels in Sao Paulo

  sppm  Numeric, sulfur (sulphur) in ppm in fuel.

  full  Logical; To return a data.frame instead or a vector adding Age, Year, Brazilian emissions standards and its euro equivalents.

  efinput  data.frame with efinput structure of sysdata cetesb. Allow apply deterioration for future emission factors

  verbose  Logical; To show more information

  csv  String with the path to download the ef in a .csv file. For instance, ef.csv
Value

A vector of Emission Factor or a data.frame

Note

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

<table>
<thead>
<tr>
<th>veh</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC_G</td>
<td>Passenger Car Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>PC_E</td>
<td>Passenger Car Ethanol (hydrous ethanol)</td>
</tr>
<tr>
<td>PC_FG</td>
<td>Passenger Car Flex Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>PC_FE</td>
<td>Passenger Car Flex Ethanol (hydrous ethanol)</td>
</tr>
<tr>
<td>LCV_G</td>
<td>Light Commercial Vehicle Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>LCV_E</td>
<td>Light Commercial Vehicle Ethanol (hydrous ethanol)</td>
</tr>
<tr>
<td>LCV_FG</td>
<td>Light Commercial Vehicle Flex Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>LCV_FE</td>
<td>Light Commercial Vehicle Flex Ethanol (hydrous ethanol)</td>
</tr>
<tr>
<td>LCV_D</td>
<td>Light Commercial Vehicle Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>TRUCKS_SL_D</td>
<td>Trucks Semi Light Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>TRUCKS_L_D</td>
<td>Trucks Light Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>TRUCKS_M_D</td>
<td>Trucks Medium Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>TRUCKS_SH_D</td>
<td>Trucks Semi Heavy Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>TRUCKS_H_D</td>
<td>Trucks Heavy Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>BUS_URBAN_D</td>
<td>Urban Bus Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>BUS_MICRO_D</td>
<td>Micro Urban Bus Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>BUS_COACH_D</td>
<td>Coach (inter-state) Bus Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>BUS_ARTIC_D</td>
<td>Articulated Urban Bus Diesel (5perc bio-diesel)</td>
</tr>
<tr>
<td>MC_150_G</td>
<td>Motorcycle engine less than 150cc Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>MC_150_500_G</td>
<td>Motorcycle engine 150-500cc Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>MC_500_G</td>
<td>Motorcycle greater than 500cc Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>MC_150_FG</td>
<td>Flex Motorcycle engine less than 150cc Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>MC_150_500_FG</td>
<td>Flex Motorcycle engine 150-500cc Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>MC_500_FG</td>
<td>Flex Motorcycle greater than 500cc Gasohol (Gasoline + 27perc of anhydrous ethanol)</td>
</tr>
<tr>
<td>MC_150_FE</td>
<td>Flex Motorcycle engine less than 150cc Ethanol (hydrous ethanol)</td>
</tr>
<tr>
<td>MC_150_500_FE</td>
<td>Flex Motorcycle engine 150-500cc Ethanol (hydrous ethanol)</td>
</tr>
<tr>
<td>MC_500_FE</td>
<td>Flex Motorcycle greater than 500cc Ethanol (hydrous ethanol)</td>
</tr>
<tr>
<td>PC_ELEC</td>
<td>Passenger Car Electric</td>
</tr>
<tr>
<td>LCV_ELEC</td>
<td>Light Commercial Vehicle Electric</td>
</tr>
</tbody>
</table>

The percentage varies of biofuels varies by law.

This 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.

3) CNG emission factors were expanded to other pollutants by comparison of US.EPA-AP42 emission factor: Section 1.4 Natural Gas Combustion.
In the previous versions I used the letter 'd' for deteriorated. I removed the letter 'd' internally to not break older code.

If by mistake, the user inputs one of veh names from the old convention, they are internally changed to the new convention: "SLT", "LT", "MT", "SHT", "HT", "UB", "SUB", "COACH", "ARTIC", "M_G_150", "M_G_150_500", "M_G_500", "M_FG_150", "M_FG_150_500", "M_FG_500", "M_FE_150", "M_FE_150_500", "M_FE_500", PC_ELEC, LCV_ELEC, TRUCKS_ELEC, BUS_ELEC, MC_150_ELEC, MC_150_500_ELEC, MC_500_ELEC

If pollutant is "SO2", it needs sppm. It is designed when veh has length 1, if it has length 2 or more, it will show a warning

Emission factor for vehicles older than the reported by CETESB were filled with las highest EF


Currently, 2020, there are not any system for recovery of fuel vapors in Brazil. Hence, the FS takes into account the vapour that comes from the fuel tank inside the car and released into the atmosphere when injecting new fuel. There are discussions about increasing implementing stage I and II and/or ORVR these days. The ef FS is calculated by transforming g FC/km into (L/KM)*g/L with g/L 1.14 for gasoline and 0.37 for ethanol (CETESB, 2016). The density considered is 0.75425 for gasoline and 0.809 for ethanol (t/m^3)

CETESB emission factors did not cover evaporative emissions from motorcycles, which occur. Therefore, in the absence of better data, it was assumed the same ratio from passenger cars.


If scale is used with tunnel, the references are:


Emission factors for resuspension applies only with top-down approach as a experimental feature. Units are g/(streets*veh)/day. These values were derived form a bottom-up resuspension emissions from metropolitan area of Sao Paulo 2018, assuming 50000 streets

NH3 from EEA Tier 2
References

Examples

{
a <- ef_cetesb(p = "CO", veh = "PC_G")
a <- ef_cetesb(p = "NOx", veh = "TRUCKS_M_D")
a <- ef_cetesb("R_10_25", "PC_G")
a <- ef_cetesb("CO", c("PC_G", "PC_FE"))
ef_cetesb(p = "CO", veh = "PC_G", year = 1970, agemax = 40)
ef_cetesb(p = "CO", veh = "TRUCKS_L_D", year = 2018)
ef_cetesb(p = "CO", veh = "SLT", year = 2018) # olds names
a <- ef_cetesb(p = "NMHC", veh = c("PC_G", "PC_FG", "PC_FE", "PC_E"), year = 2018, agemax = 20)
colplot(a, main = "NMHC EF", ylab = "[g/km]", xlab = "Years of use")
ef_cetesb(p = "PM25RES", veh = "PC_ELEC", year = 1970, agemax = 40)
ef_cetesb(p = "PM25RES", veh = "BUS_ELEC", year = 1970, agemax = 40)
}

---

ef_china

Emissions factors from Chinese emissions guidelines

Description

ef_china returns emission factors as vector or data.frames. The emission factors comes from the chinese emission guidelines (v3) from the Chinese Ministry of Ecology and Environment http://www.mee.gov.cn/gkml/hbb/bgth/201407/W020140708387895271474.pdf

Usage

ef_china(
  v = "PV",
  t = "Small",
  f = "G",
  standard,
  p,
  k = 1,
  ta = celsius(15),
  humidity = 0.5,
  altitude = 1000,
  speed = Speed(30),
  baseyear_det = 2016,
  sulphur = 50,
  load_factor = 0.5,
  details = FALSE,
  correction_only = FALSE
)
Arguments

v Character; category vehicle: "PV" for Passenger Vehicles or 'Trucks'


f Character; fuel: "G", "D", "CNG", "ALL"

standard Character or data.frame; "PRE", "I", "II", "III", "IV", "V". When it is a data.frame, it each row is a different region and ta, humidity, altitud, speed, sulphur and load_factor lengths have the same as the number of rows.

p Character; pollutant: "CO", "NOx", "HC", "PM", "Evaporative_driving" or "Evaporative_parking"

k Numeric; multiplication factor

ra Numeric; temperature of ambient in celcius degrees. When standard is a data.frame, the length must be equal to the number of rows of standard.

humidity Numeric; relative humidity. When standard is a data.frame, the length must be equal to the number of rows of standard.

altitude Numeric; altitude in meters. When standard is a data.frame, the length must be equal to the number of rows of standard.

speed Numeric; altitude in km/h When standard is a data.frame, the length must be equal to the number of rows of standard.

baseyear_det Integer; any of 2014, 2015, 2016, 2017, 2018

sulphur Numeric; sulphur in ppm. When standard is a data.frame, the length must be equal to the number of rows of standard.

load_factor Numeric; When standard is a data.frame, the length must be equal to the number of rows of standard.

details Logical; When TRUE, it shows a description of the vehicle in chinese and english. Only when length standard is 1.

correction_only Logical; When TRUE, return only correction factors.

Value

An emission factor

Note

Combination of vehicles:

<table>
<thead>
<tr>
<th>v</th>
<th>t</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>PV</td>
<td>Mini</td>
<td>G</td>
</tr>
<tr>
<td>PV</td>
<td>Bus</td>
<td>D</td>
</tr>
<tr>
<td>PV</td>
<td>Mini</td>
<td>CNG</td>
</tr>
<tr>
<td>PV</td>
<td>Bus</td>
<td>CNG</td>
</tr>
<tr>
<td>PV</td>
<td>Mini</td>
<td>G</td>
</tr>
</tbody>
</table>
standard VI is assumed as V

See Also

ef_ldv_speed, emis_hot_td

Other China: ef_china_det(), ef_china_hu(), ef_china_h(), ef_china_long(), ef_china_speed(), ef_china_s(), ef_china_te(), ef_china_th(), emis_china(), emis_long()

Examples

```
## Not run:
# when standard is 'character'
# Checking
df_st <- rev(c(as.character(as.roman(5:1)), "PRE"))
ef_china(t = "Mini", f = "G", standard = df_st, p = "CO")
ef_china(t = "Mini", f = "G", standard = df_st, p = "HC")
ef_china(t = "Mini", f = "G", standard = df_st, p = "NOx")
ef_china(t = "Mini", f = "G", standard = df_st, p = "PM2.5")
ef_china(t = "Mini", f = "G", standard = df_st, p = "PM10")
```
ef_china(t = "Small", f = "G", standard = df_st, p = "CO")
ef_china(t = "Small", f = "G", standard = df_st, p = "HC")
ef_china(t = "Small", f = "G", standard = df_st, p = "NOx")
ef_china(t = "Small", f = "G", standard = df_st, p = "PM2.5")
ef_china(t = "Small", f = "G", standard = df_st, p = "PM10")

ef_china(t = "Mini",
standard = c("PRE"),
p = "CO",
k = 1,
ta = celsius(15),
humidity = 0.5,
altitude = 1000,
speed = Speed(30),
baseyear_det = 2014,
sulphur = 50,
load_factor = 0.5,
details = FALSE)

ef_china(standard = c("PRE", "I"), p = "CO", correction_only = TRUE)

# when standard is 'data.frame'

df_st <- matrix(c("V", "IV", "III", "III", "II", "I", "PRE"), nrow = 2, ncol = 7, byrow = TRUE)
df_st <- as.data.frame(df_st)
a <- ef_china(standard = df_st,
p = "PM10",
  ta = rep(celsius(15), 2),
  altitude = rep(1000, 2),
  speed = rep(Speed(30), 2),
  sulphur = rep(50, 2))
dim(a)
dim(df_st)

ef_china(standard = df_st, p = "PM2.5", ta = rep(celsius(20), 2),
  altitude = rep(1501, 2), speed = rep(Speed(29), 2), sulphur = rep(50, 2))
a

# when standard, temperature and humidity are data.frames
# assuming 10 regions

df_st <- matrix(c("V", "IV", "III", "III", "II", "I", "PRE"), nrow = 10, ncol = 7, byrow = TRUE)
df_st <- as.data.frame(df_st)
df_t <- matrix(21:30, nrow = 10, ncol = 12, byrow = TRUE)
df_t <- as.data.frame(df_t)
for(i in 1:12) df_t[, i] <- celsius(df_t[, i])

df_h <- matrix(seq(0.4, 0.5, 0.05), nrow = 10, ncol = 12, byrow = TRUE)
df_h <- as.data.frame(df_h)
a <- ef_china(standard = df_st, p = "CO", ta = df_t, humidity = df_h,
  altitude = rep(1501, 10), speed = rep(Speed(29), 10), sulphur = rep(50, 10))
a

a <- ef_china(standard = df_st, p = "PM2.5", ta = df_t, humidity = df_h,
  altitude = rep(1501, 10), speed = rep(Speed(29), 10), sulphur = rep(50, 10))

ef_china_det

a <- ef_china(standard = df_st, p = "PM10", ta = df_t, humidity = df_h, altitude = rep(1501, 10), speed = rep(Speed(29), 10), sulphur = rep(50, 10))
a
dim(a)
## End(Not run)

---

**Description**

Correction of Chinese emission factors by deterioration

**Usage**

```r
ef_china_det(v = "PV", t = "Small", f = "G", standard, yeardet = 2015, p)
```

**Arguments**

- **v**: Character; category vehicle: "PV" for Passenger Vehicles or "Trucks"
- **f**: Character; fuel: "G", "D", "CNG", "ALL"
- **standard**: Character vector; "PRE", "I", "II", "III", "IV", "V".
- **yeardet**: Integer; any of 2014, 2015, 2016, 2017, 2018
- **p**: Character; pollutant: "CO", "NOx", "HC", "PM", "Evaporative_driving" or "Evaporative_parking"

**Value**

long data.frame

**See Also**

Other China: ef_china_hu(), ef_china_h(), ef_china_long(), ef_china_speed(), ef_china_s(), ef_china_te(), ef_china_th(), ef_china(), emis_china(), emis_long()

**Examples**

```r
{  
  ef_china_det(standard = "I", p = "CO")  
  ef_china_det(standard = c("I", "III"),  
               p = "CO",  
               f = "D")  
}
```
Description

Correction of Chinese emission

Usage

```r
ef_china_h(h, v = "PV", t = "Small", f = "G", p)
```

Arguments

- `h`: numeric altitude
- `v`: Character; category vehicle: "PV" for Passenger Vehicles or "Trucks"
- `f`: Character; fuel: "G", "D", "CNG"
- `p`: Character; pollutant: "CO", "NOx", "HC", "PM", "Evaporative_driving" or "Evaporative_parking"

Value

long data.frame

See Also

Other China: `ef_china_det()`, `ef_china_hu()`, `ef_china_long()`, `ef_china_speed()`, `ef_china_s()`, `ef_china_te()`, `ef_china_th()`, `ef_china()`, `emis_china()`, `emis_long()`

Examples

```r
{  
  ef_china_h(h = 1600, p = "CO")
}
```
**ef_china_hu**  
*Correction of Chinese emission factors by humidity*

**Description**  
Correction of Chinese emission

**Usage**

```r
ef_china_hu(hu, v = "PV", t = "Small", f = "G", standard, p)
```

**Arguments**

- `hu`: numeric humidity
- `v`: Character; category vehicle: "PV" for Passenger Vehicles or 'Trucks'
- `f`: Character; fuel: "G", "D", "CNG"
- `standard`: Character vector; "PRE", "I", "II", "III", "IV", "V".
- `p`: Character; pollutant: "CO", "NOx","HC", "PM", "Evaporative_driving" or "Evaporative_parking"

**Value**

long data.frame

**See Also**

Other China: `ef_china_det()`, `ef_china_h()`, `ef_china_long()`, `ef_china_speed()`, `ef_china_s()`, `ef_china_te()`, `ef_china_th()`, `ef_china()`, `emis_china()`, `emis_long()`

**Examples**

```r
{  
  ef_china_hu(hu = 60, standard = "I", p = "CO")
}
```
Description

Chinese emission factors in long format
Correction of Chinese emission

Usage

```
ef_china_long(v = "PV", t = "Small", f = "G", standard, p)
```

Arguments

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>Character; category vehicle: &quot;PV&quot; for Passenger Vehicles or 'Trucks'</td>
</tr>
<tr>
<td>f</td>
<td>Character; fuel: &quot;G&quot;, &quot;D&quot;, &quot;CNG&quot;, &quot;ALL&quot;</td>
</tr>
<tr>
<td>standard</td>
<td>Character vector; &quot;PRE&quot;, &quot;I&quot;, &quot;II&quot;, &quot;III&quot;, &quot;IV&quot;, &quot;V&quot;.</td>
</tr>
<tr>
<td>p</td>
<td>Character; pollutant: &quot;CO&quot;, &quot;NOx&quot;,&quot;HC&quot;, &quot;PM&quot;, &quot;Evaporative_driving&quot; or &quot;Evaporative_parking&quot;</td>
</tr>
</tbody>
</table>

Value

long data.frame

See Also

Other China: `ef_china_det()`, `ef_china_hu()`, `ef_china_h()`, `ef_china_speed()`, `ef_china_s()`, `ef_china_te()`, `ef_china_th()`, `ef_china()`, `emis_china()`, `emis_long()`

Other China: `ef_china_det()`, `ef_china_hu()`, `ef_china_h()`, `ef_china_speed()`, `ef_china_s()`, `ef_china_te()`, `ef_china_th()`, `ef_china()`, `emis_china()`, `emis_long()`

Examples

```{r}
## Not run:
# Do not run

## End(Not run)
```

```{r}
```
Correction of Chinese emission factors by sulfur

Description
Correction of Chinese emission

Usage
ef_china_s(s, f = "G", standard, p)

Arguments
s Numeric sulfur content in ppm
f Character; fuel: "G", "D", "CNG", "ALL"
standard Character vector; "PRE", "I", "II", "III", "IV", "V".
p Character; pollutant: "CO", "NOx", "HC", "PM", "Evaporative_driving" or "Evaporative_parking"

Value
long data.frame

See Also
Other China: ef_china_det(), ef_china_hu(), ef_china_h(), ef_china_long(), ef_china_speed(), ef_china_te(), ef_china_th(), ef_china(), emis_china(), emis_long()

Examples
{
  ef_china_s(s = 1000, standard = "I", p = "CO")
}

ef_china_long(standard = "I", p = "CO")
}
Correction of Chinese emission factors by speed

**Description**

Correction of Chinese emission factors by speed

**Usage**

```r
ef_china_speed(speed, f = "G", standard, p)
```

**Arguments**

- `speed`: numeric speed km/h
- `f`: Character; fuel: "G", "D", "CNG"
- `standard`: Character vector; "PRE", "I", "II", "III", "IV", "V".
- `p`: Character; pollutant: "CO", "NOx", "HC", "PM", "Evaporative_driving" or "Evaporative_parking"

**Value**

long data.frame

**See Also**

Other China: `ef_china_det()`, `ef_china_hu()`, `ef_china_h()`, `ef_china_long()`, `ef_china_s()`, `ef_china_te()`, `ef_china_th()`, `ef_china()`, `emis_china()`, `emis_long()`

**Examples**

```r
{
  data(net)
  head(ef_china_speed(speed = net$ps, standard = "I", p = "CO"))
  head(ef_china_speed(speed = net$ps, standard = c("II", "I"), p = "NOx"))
}
```
**Description**

Correction of Chinese emission

**Usage**

```r
ef_china_te(te, v = "PV", t = "Small", f = "G", p)
```

**Arguments**

- `te` numeric temperature in celsius
- `v` Character; category vehicle: "PV" for Passenger Vehicles or 'Trucks'
- `f` Character; fuel: "G", "D", "CNG"
- `p` Character; pollutant: "CO", "NOx","HC", "PM", "Evaporative_driving" or "Evaporative_parking"

**Value**

long data.frame

**See Also**

Other China: `ef_china_det()`, `ef_china_hu()`, `ef_china_h()`, `ef_china_long()`, `ef_china_speed()`, `ef_china_s()`, `ef_china_th()`, `ef_china()`, `emis_china()`, `emis_long()`

**Examples**

```r
{  
data(net)  
  head(ef_china_te(te = net$ps, p = "CO"))  
  head(ef_china_te(te = net$ps, p = "NOx"))  
}
```
**Description**

Correction of Chinese emission

**Usage**

```r
ef_china_th(hu, te, v = "PV", t = "Small", f = "G", p)
```

**Arguments**

- **hu** numeric humidity
- **te** numeric temperature in celsius
- **v** Character; category vehicle: "PV" for Passenger Vehicles or 'Trucks'
- **f** Character; fuel: "G", "D", "CNG"
- **p** Character; pollutant: "CO", "NOx", "HC", "PM", "Evaporative_driving" or "Evaporative_parking"

**Value**

long data.frame

**See Also**

Other China: ef_china_det(), ef_china_hu(), ef_china_h(), ef_china_long(), ef_china_speed(), ef_china_s(), ef_china_te(), ef_china(), emis_china(), emis_long()

**Examples**

```r
{  
  ef_china_th(hu = 60, te = 25, p = "CO")  
}
```
Description

`ef_cetesb` returns a vector or data.frame of Brazilian emission factors.

Usage

```r
ef_eea(
  category,
  fuel,
  segment,
  euro,
  tech,
  pol,
  mode,
  slope,
  load,
  speed,
  fcorr = rep(1, 8)
)
```

Arguments

- **category**: String: "PC" (Passenger Cars), "LCV" (Light Commercial Vehicles), "TRUCKS" (Heavy Duty Trucks), "BUS" (Buses) or "MC" (Motorcycles or L-Category as in EEA 2019).
- **segment**: String for type of vehicle (try different, the function will show values).
- **euro**: String; euro standard: "PRE", "IMPROVED CONVENTIONAL", "OPEN LOOP", "ECE 15/00-01", "ECE 15/02", "ECE 15/03", "ECE 15/04", "I", "II", "III", "IV", "V", "VI A/B/C", "VI D", "VI D-TEMP", "VI D/E", "EEV".
- **tech**: String; technology: "DPF", "DPF With S/W Update", "DPF+SCR" "EGR", "GDI", "GDI+GPF", "LNT+DPF", "PFI", "SCR".
- **pol**: String; "CO", "NOx", "NMHC" (VOC), "PM" (PM Exhaust), "EC", "CH4", "NH3", "N2O"
- **mode**: String; "Urban Peak", "Urban Off Peak", "Rural", "Highway", NA.
- **slope**: Numeric; 0.00, -0.06, -0.04, -0.02, 0.02, 0.04, 0.06, or NA
- **load**: Numeric; 0.0, 0.05, 1.0 or NA
- **speed**: Numeric; optional numeric in km/h.
- **fcorr**: Numeric; Correction by fuel properties by euro technology. See `fuel_corr`. The order from first to last is "PRE", "I", "II", "III", "IV", "V", "VI", "or other VI. Default is 1
Value

Return a function depending of speed or numeric (g/km)

Examples

{
  # ef_eea(category = "I DON'T KNOW")
  ef_eea(category = "PC",
  fuel = "G",
  segment = "Small",
  euro = "I",
  tech = NA,
  pol = "CO",
  mode = NA,
  slope = 0,
  load = 0)(10)
}

---

**ef_emfac**  
*Emission Factors from EMFAC emission factors*

Description

*ef_emfac* reads path to ef EMFAC. You must download the emission factors from EMFAC website.

Usage

```r
ef_emfac(efpath)
```

Arguments

- **efpath**: Character path to EMFAC ef (g/miles)

Value

data.table with emission estimation in long format

Examples

```r
## Not run:
# do not run
```
**Description**

`ef_evap` is a lookup table with tier 2 evaporative emission factors from EMEP/EEA emission guidelines.

**Usage**

```r
ef_evap(
  ef,
  v,
  cc,
  dt,
  ca,
  pollutant = "NMHC",
  k = 1,
  ltrip,
  kmday,
  show = FALSE,
  verbose = FALSE
)
```

**Arguments**

- **v**: Type of vehicles, "PC", "Motorcycle", "Motorcycle_2S" and "Moped".
- **dt**: Character or Numeric: Average monthly temperature variation: "-5_10", "0_15", "10_25" and "20_35". This argument can vector with several elements. dt can also be data.frame, but it is recommended that the number of columns are each month. So that dt varies in each row and each column.
- **ca**: Size of canister: "no" meaning no canister, "small", "medium" and "large".
- **pollutant**: Character indicating any of the covered pollutants: "NMHC", "ethane", "propane", "i-butane", "n-butane", "i-pentane", "n-pentane", "2-methylpentane", "3-methylpentane", "n-hexane", "n-heptane", "propene", "trans-2-butene", "isobutene", "cis-2-butene", "1,3-butadiene", "trans-2-pentene", "cis-2-pentene", "isoprene", "propyne", "acetylene", "benzene", "toluene", "ethylbenzene", "m-xylene", "o-xylene", "1,2,4-trimethylbenzene" and "1,3,5-trimethylbenzene". Default is "NMHC"
k  multiplication factor
ltrip Numeric; Length of trip. Experimental feature to counter g/trip and g/proced (assuming proced similar to trip) in g/km.
kmday Numeric; average daily mileage. Experimental option to convert g/day in g/km. It is an information more solid than to know the average number of trips per day.
show when TRUE shows row of table with respective emission factor.
verbose Logical; To show more information

Value

emission factors in g/trip or g/proced. The object has class (g) but it order to know it is g/trip or g/proced the argument show must by T

Note

Diurnal loses occur with daily temperature variations. Running loses occur during vehicles use. Hot soak emission occur following vehicles use.

References


Examples

```r
# Not run:
# Do not run
a <- ef_evap(ef = "eshotc", v = "PC", cc = "<=1400", dt = "0_15", ca = "no",
pollutant = "cis-2-pentene")
a <- ef_evap(ef = "ed", v = "PC", cc = "<=1400", dt = "0_15", ca = "no",
show = TRUE)
a <- ef_evap(ef = c("erhotc", "erhotc"), v = "PC", cc = "<=1400",
dt = "0_15", ca = "no",
show = TRUE)
a <- ef_evap(ef = c("erhotc", "erhotc"), v = "PC", cc = "<=1400",
dt = "0_15", ca = "no",
show = FALSE)
a <- ef_evap(ef = "eshotc", v = "PC", cc = "<=1400", dt = "0_15", ca = "no",
show = TRUE)
eff_evap(ef = "erhotc", v = "PC", cc = "<=1400", dt = "0_15", ca = "no",
show = TRUE)
temps <- 10:20
a <- ef_evap(ef = "erhotc", v = "PC", cc = "<=1400", dt = temps, ca = "no",
show = TRUE)
dt <- matrix(rep(1:24,5), ncol = 12) # 12 months
dt <- celsius(dt)
a <- ef_evap(ef = "erhotc", v = "PC", cc = "<=1400",
dt = dt, ca = "no")
lkm <- units::set_units(10, km)
a <- ef_evap(ef = "erhotc", v = "PC", cc = "<=1400", ltrip = lkm,
```
Description

ef_fun returns amount of vehicles at each age

Usage

ef_fun(
  ef,
  type = "logistic",
  x = 1:length(ef),
  x0 = mean(ef),
  k = 1/4,
  L = max(ef)
)

Arguments

  ef  Numeric; numeric vector of emission factors.
  type Character; "logistic" by default so far.
  x   Numeric; vector for ages of use.
  x0  Numeric; the x-value of the sigmoid’s midpoint,
  k   Numeric; the steepness of the curve.
  L   Integer; the curve’s maximum value.

Value

dataframe of age distribution of vehicles at each street.

References

  https://en.wikipedia.org/wiki/Logistic_function

Examples

  ## Not run:
  data(fe2015)
  CO <- ef_cetesb(p = "CO", veh = "PC_G")
  ef_logit <- ef_fun(ef = CO, x0 = 27, k = 0.4, L = 33)
  df <- data.frame(CO, ef_logit)
  colplot(df)
  ## End(Not run)
Description

`ef_hdv_scaled` creates a list of scaled functions of emission factors. A scaled emission factor which at a speed of the driving cycle (SDC) gives a desired value. This function needs a dataframe with local emission factors with a columns with the name "Euro_HDV" indicating the Euro equivalence standard, assuming that there are available local emission factors for several consecutive years.

Usage

```r
ef_hdv_scaled(df, dfcol, SDC = 34.12, v, t, g, eu, gr = 0, l = 0.5, p)
```

Arguments

- `df`: deprecated
- `dfcol`: Column of the dataframe with the local emission factors eg `df$dfcol`
- `SDC`: Speed of the driving cycle
- `v`: Category vehicle: "Coach", "Trucks" or "Ubus"
- `t`: Sub-category of of vehicle: "3Axes", "Artic", "Midi", "RT", "Std" and "TT"
- `eu`: Euro emission standard: "PRE", "I", "II", "III", "IV" and "V"
- `gr`: Gradient or slope of road: -0.06, -0.04, -0.02, 0.00, 0.02, 0.04 or 0.06
- `l`: Load of the vehicle: 0.0, 0.5 or 1.0
- `p`: Pollutant: "CO", "FC", "NOx" or "HC"

Value

A list of scaled emission factors g/km

Note

The length of the list should be equal to the name of the age categories of a specific type of vehicle.
ef_hdv_speed

Examples

{  
  # Do not run  
  CO <- ef_cetesb(p = "CO", veh = "TRUCKS_SL_D", full = TRUE)  
  lef <- ef_hdv_scaled(dfcol = CO$CO,  
                        v = "Trucks",  
                        t = "RT",  
                        g = "<=7.5",  
                        eu = CO$Euro_EqHDV,  
                        gr = 0,  
                        l = 0.5,  
                        p = "CO")  

  length(lef)  
  ages <- c(1, 10, 20, 30, 40)  
  EmissionFactors(do.call("cbind",  
                    lapply(ages, function(i) {  
                      data.frame(i = lef[[i]](1:100))  
                    })))-> df  
  names(df) <- ages  
  colplot(df)  
}

ef_hdv_speed         Emissions factors for Heavy Duty Vehicles based on average speed

Description

This function returns speed dependent emission factors. The emission factors comes from the guide-
lines EMEP/EEA air pollutant emission inventory guidebook http://www.eea.europa.eu/themes/air/emep-
eea-air-pollutant-emission-inventory-guidebook

Usage

ef_hdv_speed(  
  v,  
  t,  
  g,  
  eu,  
  x,  
  gr = 0,  
  l = 0.5,  
  p,  
  k = 1,  
  show.equation = FALSE,  
  speed,  
  fcorr = rep(1, 8)  
)
Arguments

v
  Category vehicle: "Coach", "Trucks" or "Ubus"

t
  Sub-category of vehicle: "3Axes", "Artic", "Midi", "RT", "Std" and "TT"

g

eu

x
  Numeric; if pollutant is "SO2", it is sulfur in fuel in ppm, if is "Pb", Lead in fuel in ppm.

g
  Gradient or slope of road: -0.06, -0.04, -0.02, 0.00, 0.02, 0.04 or 0.06

l
  Load of the vehicle: 0.0, 0.5 or 1.0

p
  Character; pollutant: "CO", "FC", "NOx", "NO", "NO2", "HC", "PM", "NMHC", "CH4", "CO2", "SO2" or "Pb". Only when p is "SO2" pr "Pb" x is needed. See notes.

k
  Multiplication factor

show.equation
  Option to see or not the equation parameters

speed
  Numeric; Speed to return Number of emission factor and not a function. It needs units in km/h

fcorr
  Numeric; Correction by fuel properties by euro technology. See fuel_corr. The order from first to last is "PRE", "I", "II", "III", "IV", "V", VI, "Vlc". Default is 1

Value

an emission factor function which depends of the average speed V g/km

Note

Pollutants (g/km): "CO", "NOx", "HC", "PM", "CH4", "NMHC", "CO2", "SO2", "Pb".

Black Carbon and Organic Matter (g/km): "BC", "OM"

PAH and POP (g/km): See speciate Dioxins and furans (g equivalent toxicity / km): See speciate

Metals (g/km): See speciate

Active Surface (cm²/km) See speciate

Total Number of particles (N/km): See speciate

The available standards for Active Surface or number of particles are: Euro II and III Euro II and III + CRDPF Euro II and III + SCR Euro IV + CRDPF Euro V + SCR

The categories Pre Euro and Euro I were assigned with the factors of Euro II and Euro III The categories euro IV and euro V were assigned with euro III + SCR

Fuel consumption for heavy VI comes from V
### Description

`ef_im` calculate the theoretical emission factors of vehicles. The approach is different from including deterioration factors (`emis_det`) but similar, because they represent how much emits a vehicle with a normal deterioration, but that it will pass the Inspection and Maintenance program.
Usage

```r
ef_im(ef, tc, amileage, max_amileage, max_ef, verbose = TRUE)
```

Arguments

- **ef**: Numeric; emission factors of vehicles with 0 mileage (new vehicles).
- **tc**: Numeric; rate of growth of emissions by year of use.
- **amileage**: Numeric; Accumulated mileage by age of use.
- **max_amileage**: Numeric; Max accumulated mileage. This means that after this value, mileage is constant.
- **max_ef**: Numeric; Max ef. This means that after this value, ef is constant.
- **verbose**: Logical; if you want detailed description.

Value

An emission factor of a deteriorated vehicle under normal conditions which would be approved in an inspection and maintenance program.

Examples

```r
## Not run:
# Do not run
# Passenger Cars PC
data(fkm)
# cumulative mileage from 1 to 50 years of use, 40:50
mil <- cumsum(fkm$KM_PC_E25(1:10))
ef_im(ef = seq(0.1, 2, 0.2), seq(0.1, 1, 0.1), mil)
## End(Not run)
```

---

**ef_ldv_cold**

*Cold-Start Emissions factors for Light Duty Vehicles*

Description

**ef_ldv_cold** returns speed functions or data.frames which depends on ambient temperature average speed. The emission factors comes from the guidelines EMEP/EEA air pollutant emission inventory guidebook [http://www.eea.europa.eu/themes/air/emep-eea-air-pollutant-emission-inventory-guidebook](http://www.eea.europa.eu/themes/air/emep-eea-air-pollutant-emission-inventory-guidebook)*
Usage

```r
ef_ldv_cold(
  v = "LDV",
  ta,
  cc,
  f,
  eu,
  p,
  k = 1,
  show.equation = FALSE,
  speed,
  fcorr = rep(1, 8)
)
```

Arguments

- **v** Character; Category vehicle: "LDV"
- **ta** Numeric vector or data.frame; Ambient temperature. Monthly mean can be used. When `ta` is a data.frame, one option is that the number of rows should be the number of rows of your `Vehicles` data.frame. This is convenient for top-down approach when each simple feature can be a polygon, with a monthly average temperature for each simple feature. In this case, the number of columns can be the 12 months.
- **cc** Character; Size of engine in cc: "<=1400", "1400_2000" or">2000"
- **f** Character; Type of fuel: "G", "D" or "LPG"
- **eu** Character or data.frame of Characters; Euro standard: "PRE", "I", "II", "III", "IV", "V", "VI" or "VIc". When `eu` is a data.frame and `ta` is also a data.frame both has to have the same number of rows. For instance, When you want that each simple feature or region has a different emission standard.
- **p** Character; Pollutant: "CO", "FC", "NOx", "HC" or "PM"
- **k** Numeric; Multiplication factor
- **show.equation** Option to see or not the equation parameters
- **speed** Numeric; Speed to return Number of emission factor and not a function.
- **fcorr** Numeric; Correction by fuel properties by euro technology. See `fuel_corr`. The order from first to last is "PRE", "I", "II", "III", "IV", "V", VI, "VIc". Default is 1

Value

an emission factor function which depends of the average speed V and ambient temperature. g/km

See Also

`fuel_corr`
Examples

```r
## Not run:
ef1 <- ef_ldv_cold(ta = 15, cc = "<=1400", f = "G", eu = "PRE", p = "CO",
show.equation = TRUE)
ef1(10)
speed <- Speed(10)
ef_ldv_cold(ta = 15, cc = "<=1400", f = "G", eu = "PRE", p = "CO", speed = speed)
# lets create a matrix of ef cold at different speeds and temperatures
te <- -50:50
lf <- sapply(1:length(te), function(i){
  ef_ldv_cold(ta = te[i], cc = "<=1400", f = "G", eu = "I", p = "CO", speed = Speed(0:120))
})
filled.contour(lf, col = cptcity::lucky())
euros <- c("V", "V", "IV", "III", "II", "I", "PRE", "PRE")
lf <- ef_ldv_cold(ta = 10, cc = "<=1400", f = "G", eu = euros, p = "CO", speed = Speed(0))
lf <- ef_ldv_cold(ta = 10, cc = "<=1400", f = "G", eu = euros, p = "CO", speed = Speed(0:120))
dt <- matrix(rep(2:25, 5), ncol = 12) # 12 months
ef_ldv_cold(ta = dt, cc = "<=1400", f = "G", eu = euros, p = "CO", speed = Speed(0))
ef_ldv_cold(ta = dt, cc = "<=1400", f = "G", eu = euros, p = "CO", speed = Speed(34))
euros2 <- c("V", "V", "IV", "IV", "IV", "III", "III")
dfe <- rbind(euros, euros2)
lf <- ef_ldv_cold(ta = 10, cc = "<=1400", f = "G", eu = dfe, p = "CO", speed = Speed(0))

# Fuel corrections
corr <- c(0.5, 1, 1, 0.9, 0.9, 0.9, 0.9)
ef1 <- ef_ldv_cold(ta = 15, cc = "<=1400", f = "G", eu = "PRE", p = "CO",
show.equation = TRUE, fcorr = corr)
ef_ldv_cold(ta = 10, cc = "<=1400", f = "G", eu = dfe, p = "CO", speed = Speed(0),
  fcorr = corr)
## End(Not run)
```

---

### ef_ldv_cold_list

**List of cold start emission factors of Light Duty Vehicles**

**Description**

This function creates a list of functions of cold start emission factors considering different euro emission standard to the elements of the list.

**Usage**

```r
ef_ldv_cold_list(df, v = "LDV", ta, cc, f, eu, p)
```

**Arguments**

- **df**: Dataframe with local emission factor
- **v**: Category vehicle: "LDV"
**ef_ldv_scaled**

Scaling constant with speed emission factors of Light Duty Vehicles

**Description**

This function creates a list of scaled functions of emission factors. A scaled emission factor which at a speed of the driving cycle (SDC) gives a desired value.

**Usage**

```r
ef_ldv_scaled(df, dfcol, SDC = 34.12, v, t = "4S", cc =<1400"", "1400_2000" and ">2000"

Type of fuel: "G" or "D"  
Character vector of euro standards: "PRE", "I", "II", "III", "IV", "V", "VI" or 
"VIc".

Pollutant: "CO", "FC", "NOx", "HC" or "PM"

A list of cold start emission factors g/km

The length of the list should be equal to the name of the age categories of a specific type of vehicle

### Examples

```r
## Not run:
# Do not run
df <- data.frame(age1 = c(1,1),
                 age2 = c(2,2))

eu = c("I", "PRE")

l <- ef_ldv_cold(t = 17, cc = "<=1400", f = "G",
                 eu = "I", p = "CO")

l_cold <- ef_ldv_cold_list(df, t = 17, cc = "<=1400", f = "G",
                           eu = eu, p = "CO")

length(l_cold)

## End(Not run)
```
Arguments

- `df`: deprecated
- `dfcol`: Column of the dataframe with the local emission factors eg `df$dfcol`
- `SDC`: Speed of the driving cycle
- `v`: Category vehicle: "PC", "LCV", "Motorcycle" or "Moped"
- `t`: Sub-category of vehicle: PC: "ECE_1501", "ECE_1502", "ECE_1503", "ECE_1504" , "IMPROVED_CONVENTIONAL", "OPEN_LOOP", "ALL", "2S" or "4S". LCV: "4S", Motorcycle: "2S" or "4S". Moped: "2S" or "4S"
- `f`: Type of fuel: "G", "D", "LPG" or "FH" (Full Hybrid: starts by electric motor)
- `p`: Pollutant: "CO", "FC", "NOx", "HC" or "PM". If your pollutant `dfcol` is based on fuel, use "FC", if it is based on "HC", use "HC".

Details

This function calls "ef_ldv_speed" and calculate the specific k value, dividing the local emission factor by the respective speed emissions factor at the speed representative of the local emission factor, e.g. If the local emission factors were tested with the FTP-75 test procedure, SDC = 34.12 km/h.

Value

A list of scaled emission factors g/km

Note

The length of the list should be equal to the name of the age categories of a specific type of vehicle. Thanks to Glauber Camponogara for the help.

See Also

- `ef_ldv_seed`

Examples

```r
{  CO <- ef_cetesb(p = "CO", veh = "PC_FG", full = TRUE)  lef <- ef_ldv_scaled(dfcol = CO$CO,  v = "PC",  t = "4S",  cc = "<=1400",  f = "G",  eu = CO$EqEuro_PC,  p = "CO")}
```
ef_ldv_speed

\[
\text{length}(\text{lef})
\]
\[
\text{ages} \leftarrow c(1, 10, 20, 30, 40)
\]
\[
\text{EmissionFactors}(\text{do.call}(\text{"cbind"},
\text{lapply}(\text{ages}, \text{function}(i) \{
\text{data.frame}(i = \text{lef}[i](1:100))
\text{\)}})) \rightarrow \text{df}
\]
\[
\text{names}(\text{df}) \leftarrow \text{ages}
\]
\[
\text{colplot(\text{df})}
\]

---

**ef_ldv_speed**

*Emissions factors for Light Duty Vehicles and Motorcycles*

**Description**

*ef_ldv_speed* returns speed dependent emission factors, data.frames or list of emission factors. The emission factors comes from the guidelines EMEP/EEA air pollutant emission inventory guidebook http://www.eea.europa.eu/themes/air/emep-eea-air-pollutant-emission-inventory-guidebook

**Usage**

\[
\text{ef_ldv_speed}(\text{v},
\text{t} = \text{"4S"},
\text{cc},
\text{f},
\text{eu},
\text{p},
\text{x},
\text{k} = 1,
\text{speed},
\text{show.equation} = \text{FALSE},
\text{fcorr} = \text{rep}(1, 8)
\)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>v</td>
<td>Character; category vehicle: &quot;PC&quot;, &quot;LCV&quot;, &quot;Motorcycle&quot; or &quot;Moped&quot;</td>
</tr>
<tr>
<td>t</td>
<td>Character; sub-category of vehicle: PC: &quot;ECE_1501&quot;, &quot;ECE_1502&quot;, &quot;ECE_1503&quot;, &quot;ECE_1504&quot;, &quot;IMPROVED_CONVENTIONAL&quot;, &quot;OPEN_LOOP&quot;, &quot;ALL&quot;, &quot;2S&quot; or &quot;4S&quot;. LCV: &quot;4S&quot;, Motorcycle: &quot;2S&quot; or &quot;4S&quot;. Moped: &quot;2S&quot; or &quot;4S&quot;</td>
</tr>
<tr>
<td>f</td>
<td>Character; type of fuel: &quot;G&quot;, &quot;D&quot;, &quot;LPG&quot; or &quot;FH&quot; (Gasoline Full Hybrid). Full hybrid vehicles cannot be charged from the grid and recharge; only its own engine may recharge its batteries.</td>
</tr>
</tbody>
</table>
ef_ldv_speed

eu
   Character or data.frame of characters; euro standard: "PRE", "I", "II", "III", "III+DPF", "IV", "V", "VI" or "VIc". When the pollutant is active surface or number of particles, eu can also be "III+DISI"

p
   Character; pollutant: "CO", "FC", "NOx", "NO", "NO2", "HC", "PM", "NMHC", "CH4", "CO2", "SO2" or "Pb". Only when p is "SO2" pr "Pb" x is needed. Also polycyclic aromatic hydrocarbons (PAHs), persistent organi pollutants (POPs), and Number of particles and Active Surface.

x
   Numeric; if pollutant is "SO2", it is sulphur in fuel in ppm, if is "Pb", Lead in fuel in ppm.

k
   Numeric; multiplication factor

speed
   Numeric; Speed to return Number of emission factor and not a function.

show.equation
   Logical; option to see or not the equation parameters.

fcorr
   Numeric; Correction by fuel properties by euro technology. See fuel_corr.

   The order from first to last is "PRE", "I", "II", "III", "IV", "V", VI, "VIc". Default is 1

Details

   The argument of this functions have several options which results in different combinations that returns emission factors. If a combination of any option is wrong it will return an empty value. Therefore, it is important ti know the combinations.

Value

   An emission factor function which depends of the average speed V g/km

Note

   t = "ALL" and cc == "ALL" works for several pollutants because emission fators are the same. Some exceptions are with NOx and FC because size of engine.

   Hybrid cars: the only cover "PC" and according to EMEP/EEA air pollutant emission inventory guidebook 2016 (Ntziachristos and Samaras, 2016) only for euro IV. When new literature is available, I will update these factors.

   Pollutants (g/km): "CO", "NOx", "HC", "PM", "CH4", "NMHC", "CO2", "SO2", "Pb", "FC".

   Black Carbon and Organic Matter (g/km): "BC", "OM"

   PAH and POP (g/km): speciate Dioxins and furans(g equivalent toxicity / km): speciate Metals (g/km): speciate

   NMHC (g/km): speciate

   Active Surface (cm2/km): speciate"AS_urban", "AS_rural", "AS_highway"

   Total Number of particles (N/km): speciate "N_urban", "N_rural", "N_highway", "N_50nm_urban", "N_50_100nm_rural", "N_100_1000nm_highway".

   The available standards for Active Surface or number of particles are Euro I, II, III, III+DPF dor diesle and III+DISI for gasoline. Pre euro vehicles has the value of Euro I and euro IV, V, VI and VIc the value of euro III.
ef_ldv_speed

See Also

fuel_corrmis ef_ldv_cold

Examples

## Not run:
# Passenger Cars PC
# Emission factor function
V <- 0:150
ef1 <- ef_ldv_speed(v = "PC",t = "4S", cc = "<=1400", f = "G", eu = "PRE", p = "CO")
efs <- EmissionFactors(ef1(1:150))
plot(Speed(1:150), efs, xlab = "speed[km/h]", type = "b", pch = 16, col = "blue")

# Quick view
pol <- c("CO", "NOx", "HC", "NMHC", "CH4", "FC", "PM", "CO2", "SO2", "1-butyne", "propyne")
f <- sapply(1:length(pol), function(i){
  ef_ldv_speed("PC", "4S", "<=1400", "G", "PRE", pol[i], x = 10)(30)
})
f

# PM Characteristics
pol <- c("AS_urban", "AS_rural", "AS_highway", "N_urban", "N_rural", "N_highway", "N_50nm_urban", "N_50_100nm_rural", "N_100_1000nm_highway")
f <- sapply(1:length(pol), function(i){
  ef_ldv_speed("PC", "4S", "<=1400", "D", "PRE", pol[i], x = 10)(30)
})
f

# PAH POP
ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G", eu = "PRE", p = "indeno(1,2,3-cd)pyrene")(10)

# Dioxins and Furans
ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G", eu = "PRE", p = "PCB")(10)

# NMHC
ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G", eu = "PRE", p = "hexane")(10)

# List of Copert emission factors for 40 years fleet of Passenger Cars.
# Assuming a euro distribution of euro V, IV, III, II, and I of 5 years each and the rest 15 as PRE euro:
speed <- 25
lef <- lapply(1:40, function(i) {
  ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G", eu = euro[i], p = "CO")
})
ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G",
eu = euro[i], p = "CO", show.equation = FALSE)(25))

# to check the emission factor with a plot

efs <- EmissionFactors(unlist(lef)) # returns 'units'

plot(efs, xlab = "age")
lines(efs, type = "l")

euros <- c("VI", "V", "IV", "III", "II")

ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G",
eu = euros, p = "CO")

a <- ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G",
eu = euros, p = "CO", speed = Speed(0:120))

head(a)

filled.contour(as.matrix(a)[1:10, 1:length(euros)],
col = cptcity::cpt(n = 18))
filled.contour(as.matrix(a)[110:120, 1:length(euros)],
col = cptcity::cpt(n = 16))
filled.contour(as.matrix(a)[1, 1:length(euros)],
col = cptcity::cpt("mpl_viridis", n = 21))
filled.contour(as.matrix(a)[1, 1:length(euros)],
col = cptcity::cpt("mpl_magma", n = 21))
persp(as.matrix(a)[1, 1:length(euros)], phi = 0, theta = 0)
persp(as.matrix(a)[1, 1:length(euros)], phi = 25, theta = 45)
persp(as.matrix(a)[1, 1:length(euros)], phi = 0, theta = 90)
persp(as.matrix(a)[1, 1:length(euros)], phi = 25, theta = 90+45)
persp(as.matrix(a)[1, 1:length(euros)], phi = 0, theta = 180)

new_euro <- c("VI", "VI", "V", "V", "V")
euro <- c("V", "V", "IV", "III", "II")
old_euro <- c("III", "II", "I", "PRE", "PRE")

meuros <- rbind(new_euro, euro, old_euro)

aa <- ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G",
eu = meuros, p = "CO", speed = Speed(0:11))

# Light Commercial Vehicles

V <- 0:150
ef1 <- ef_ldv_speed(v = "LCV", t = "4S", cc = "<3.5", f = "G",
eu = "PRE", p = "CO")

efs <- EmissionFactors(ef1[1:150])

plot(Speed(1:150), efs, xlab = "speed[km/h]")

lef <- lapply(1:5, function(i) {
  ef_ldv_speed(v = "LCV", t = "4S", cc = "<3.5", f = "G",
eu = euro[i], p = "CO", show.equation = FALSE)(25))

# to check the emission factor with a plot

efs <- EmissionFactors(unlist(lef)) # returns 'units'

plot(efs, xlab = "age")
lines(efs, type = "l")

# Motorcycles

V <- 0:150
ef1 <- ef_ldv_speed(v = "Motorcycle", t = "4S", cc = "<250", f = "G",
eu = "PRE", p = "CO", show.equation = TRUE)

efs <- EmissionFactors(ef1[1:150])

plot(Speed(1:150), efs, xlab = "speed[km/h]")

# euro for motorcycles


lef <- lapply(1:30, function(i) {

...
ef_local

Description

ef_local process an data.frame delivered by the user, but adding similar funcionality and arguments as ef_cetesb, which are classification, filtering and projections

Usage

ef_local(
  p,
  veh,
  year = 2017,
  agemax = 40,
  ef,
  full = FALSE,
  project = "constant",
  verbose = TRUE
)

Arguments

p Character; pollutant delivered by the user. the name of the column of the data.frame must be Pollutant.

veh Character; Vehicle categories available in the data.frame provided by the user

year Numeric; Filter the emission factor to start from a specific base year. If project is 'constant' values above 2017 and below 1980 will be repeated

agemax Integer; age of oldest vehicles for that category

ef data.frame, for local the emission factors. The names of the ef must be ‘Age’ ‘Year’ ‘Pollutant’ and all the vehicle categories...
ef_nitro

full
Logical; To return a data.frame instead or a vector adding Age, Year, Brazilian
emissions standards and its euro equivalents.

project
Character showing the method for projecting emission factors in future. Cur-
rently the only value is "constant"

verbose
Logical; To show more information

Details
returns a vector or data.frame of Brazilian emission factors.

Value
A vector of Emission Factor or a data.frame

Note
The names of the ef must be ‘Age’ ‘Year’ ‘Pollutant’ and all the vehicle categories...

See Also
ef_cetesb

Examples

## Not run:
#do not run

## End(Not run)

---

def nitro

Emissions factors of N2O and NH3

Description

ef_nitro returns emission factors as a functions of accumulated mileage. The emission factors
comes from the guidelines EMEP/EEA air pollutant emission inventory guidebook http://www.eea.europa.eu/themes/air/emep-
eea-air-pollutant-emission-inventory-guidebook

Usage

ef_nitro(
  v,
  t = "Hot",
  cond = "Urban",
  cc,
  f,
  eu,
  p = "NH3",
)
Arguments

t Type: "Cold" or "Hot"
cond "Urban", "Rural", "Highway"
f Type of fuel: "G", "D" or "LPG"
eu Euro standard: "PRE", "I", "II", "III", "IV", "V", "VI", "VIIc"
p Pollutant: "N2O", "NH3"
S Sulphur (ppm). Number.
cumileage Numeric; Accumulated mileage to return number of emission factor and not a function.
k Multiplication factor
show.equation Option to see or not the equation parameters
fcorr Numeric; Correction by by euro technology.

Value

an emission factor function which depends on the accumulated mileage, or an EmissionFactor

Note

if length of eu is bigger than 1, cumileage can have values of length 1 or length equal to length of eu

Examples

## Not run:

```R
efe10 <- ef_nitro(v = "PC", t = "Hot", cond = "Urban", f = "G", cc = "<=1400", eu = "III", p = "NH3", S = 10,
show.equation = FALSE)
efe50 <- ef_nitro(v = "PC", t = "Hot", cond = "Urban", f = "G", cc = "<=1400", eu = "III", p = "NH3", S = 50,
show.equation = TRUE)
efe10(10)
efe50(10)
efe10 <- ef_nitro(v = "PC", t = "Hot", cond = "Urban", f = "G", cc = "<=1400",
```
### ef_wear

**Emissions factors from tyre, break and road surface wear**

#### Description

`ef_wear` estimates wear emissions. The sources are tyres, breaks and road surface.

#### Usage

```r
def ef_wear(
    wear,        # Character; type of wear: "tyre" (or "tire"), "break" (or "brake") and "road"
    type,        # Character; type of vehicle: "2W", "MC", "Motorcycle", "PC", "LCV", "HDV", "BUS", "TRUCKS"
    pol = "TSP", # Character; pollutant: "TSP", "PM10", "PM2.5", "PM1" and "PM0.1"
    speed,       # Data.frame of speeds
    load = 0.5,  # Load of the HDV
    axle = 2,    # Number of axle of the HDV
    road = "urban", # Type of road "urban", "rural", "motorway". Only applies when type is "E6DV" or "BEV"
    verbose = FALSE # Logical to show more information. Only applies when type is "E6DV" or "BEV"
)
```

#### Arguments

- `wear`: Character; type of wear: "tyre" (or "tire"), "break" (or "brake") and "road"
- `type`: Character; type of vehicle: "2W", "MC", "Motorcycle", "PC", "LCV", "HDV", "BUS", "TRUCKS"
- `pol`: Character; pollutant: "TSP", "PM10", "PM2.5", "PM1" and "PM0.1"
- `speed`: Data.frame of speeds
- `load`: Load of the HDV
- `axle`: Number of axle of the HDV
- `road`: Type of road "urban", "rural", "motorway". Only applies when type is "E6DV" or "BEV"
- `verbose`: Logical to show more information. Only applies when type is "E6DV" or "BEV"

#### Value

emission factors grams/km
References


Examples

```r
{
  data(net)
  data(pc_profile)
  pc_week <- temp_fact(net$ldv+net$hdv, pc_profile)
  df <- netspeed(pc_week, net$ps, net$fff, net$capacity, net$lkm, alpha = 1)
  ef <- ef_wear(wear = "tyre", type = "PC", pol = "PM10", speed = df)
  ef_wear(wear = "tyre",
          type = c("E6DV"),
          pol = "PM10",
          verbose = TRUE)
  ef_wear(wear = "tyre",
          type = c("E6DV"),
          pol = "PM10",
          verbose = FALSE)
}
```

---

**ef_whe**

*Emission factor that incorporates the effect of high emitters*

Description

**ef_whe** return weighted emission factors of vehicles considering that one part of the fleet has a normal deterioration and another has a deteriorated fleet that would be rejected in a inspection and maintenance program but it is still in circulation. This emission factor might be applicable in cities without a inspection and maintenance program and with Weighted emission factors considering that part of the fleet are high emitters.

Usage

```r
ef_whe(efhe, phe, ef)
```
Arguments

- efhe: Numeric; Emission factors of high emitters vehicles. This vehicles would be rejected in a inspection and maintenance program.
- phe: Numeric; Percentage of high emitters.
- ef: Numeric; Emission factors deteriorated vehicles under normal conditions. These vehicles would be approved in a inspection and maintenance program.

Value

An emission factor by annual mileage.

Examples

```r
# Do not run
# Let’s say high emitter is 5 times the normal ef.
co_efhe <- ef_cetesb(p = "COd", "PC_G") * 5
# Let’s say that the perfil of high emitters increases linearly
# till 30 years and after that percentage is constant
perc <- c(seq(0.01, 0.3, 0.01), rep(0.3, 10))
# Now, lets use our ef with normal deterioration
co_ef_normal <- ef_cetesb(p = "COd", "PC_G")
efd <- ef_whe(efhe = co_efhe,
              phe = perc,
              ef = co_ef_normal)
# now, we can plot the three ef
colplot(data.frame(co_efhe, co_ef_normal, efd))
```

emis

Estimation of emissions

Description

emis estimates vehicular emissions as the product of the vehicles on a road, length of the road, emission factor evaluated at the respective speed. \( E = \text{VEH} \times \text{LENGTH} \times \text{EF(speed)} \)

Usage

```r
emis(
  veh,
  lkm,
  ef,
  speed,
  agemax = ifelse(is.data.frame(veh), ncol(veh), ncol(veh[[1]])),
  profile,
  simplify = FALSE,
)```
fortran = FALSE,
hour = nrow(profile),
day = ncol(profile),
verbose = FALSE,
nt = ifelse(check_nt() == 1, 1, check_nt()/2)
)

Arguments

veh  "Vehicles" data-frame or list of "Vehicles" data-frame. Each data-frame as number of columns matching the age distribution of that type of vehicle. The number of rows is equal to the number of streets link. If this is a list, the length of the list is the vehicles for each hour.

lkm  Length of each link in km

ef   List of functions of emission factors

speed Speed data-frame with number of columns as hours. The default value is 34km/h

agemax Age of oldest vehicles for that category

profile Dataframe or Matrix with nrows equal to 24 and ncol 7 day of the week

simplify Logical; to determine if EmissionsArray should les dimensions, being streets, vehicle categories and hours or default (streets, vehicle categories, hours and days). Default is FALSE to avoid break old code, but the recommendation is that new estimations use this parameter as TRUE

fortran Logical; to try the fortran calculation when speed is not used. I will add fortran for EmissionFactorsList soon.

hour  Number of considered hours in estimation. Default value is number of rows of argument profile

day Number of considered days in estimation

verbose Logical; To show more information

nt    Integer; Number of threads wich must be lower than max available. See check_nt. Only when fortran = TRUE

Value

If the user applies a top-down approach, the resulting units will be according its own data. For instance, if the vehicles are veh/day, the units of the emissions implicitly will be g/day.

Note

Hour and day will be deprecated because they can be inferred from the profile matrix.

Examples

## Not run:
# Do not run
data(net)
data(pc_profile)
data(profiles)
data(Fe2015)
data(fkm)

PC_G <- c(33491, 22340, 24818, 31808, 46458, 28574, 24856, 28972, 37818, 49050, 87923, 133833, 138441, 142682, 171029, 151048, 115228, 98664, 126444, 101027, 84771, 55864, 36306, 21079, 20138, 17439, 7854, 2215, 656, 1262, 476, 512, 1181, 4991, 3711, 5653, 7039, 5839, 4257, 3824, 3068)

c1 <- my_age(x = net$ldv, y = PC_G, name = "PC")

# Estimation for morning rush hour and local emission factors and speed
speed <- data.frame(S8 = net$ps)
lef <- EmissionFactorsList(ef_cetesb("CO", "PC_G", agemax = ncol(pc1)))
system.time(E_CO <- emis(veh = pc1, lkm = net$lkm, ef = lef, speed = speed))
system.time(E_CO_2 <- emis(veh = pc1, lkm = net$lkm, ef = lef, speed = speed, simplify = TRUE))
identical(E_CO, E_CO_2)

# Estimation for morning rush hour and local emission factors without speed
lef <- ef_cetesb("CO", "PC_G", agemax = ncol(pc1))
system.time(E_CO <- emis(veh = pc1, lkm = net$lkm, ef = lef))
system.time(E_CO_2 <- emis(veh = pc1, lkm = net$lkm, ef = lef, fortran = TRUE))
identical(E_CO, E_CO_2)

# Estimation for 168 hour and local factors and speed
pcw <- temp_fact(net$ldv + net$hdv, pc_profile)
speed <- netspeed(pcw, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
lef <- EmissionFactorsList(ef_cetesb("CO", "PC_G", agemax = ncol(pc1)))
system.time(
E_CO <- emis(
  veh = pc1,
  lkm = net$lkm,
  ef = lef,
  speed = speed,
  profile = profiles$PC_JUNE_2014
)
)
system.time(
E_CO_2 <- emis(
  veh = pc1,
  lkm = net$lkm,
  ef = lef,
  speed = speed,
  profile = profiles$PC_JUNE_2014,
  simplify = TRUE
)
)

# Estimation for 168 hour and local factors and without speed
lef <- ef_cetesb("CO", "PC_G", agemax = ncol(pc1))
system.time(
E_CO <- emis(
  veh = pc1,
lkm = net$lkm,
    ef = lef,
    profile = profiles$PC_JUNE_2014
  )
)
sum(E_CO)
system.time(
  E_CO_2 <- emis(
    veh = pc1,
    lkm = net$lkm,
    ef = lef,
    profile = profiles$PC_JUNE_2014,
    fortran = TRUE
  )
)
sum(E_CO)
system.time(
  E_CO_3 <- emis(
    veh = pc1,
    lkm = net$lkm,
    ef = lef,
    profile = profiles$PC_JUNE_2014,
    simplify = TRUE
  )
)
sum(E_CO)
system.time(
  E_CO_4 <- emis(
    veh = pc1,
    lkm = net$lkm,
    ef = lef,
    profile = profiles$PC_JUNE_2014,
    simplify = TRUE,
    fortran = TRUE
  )
)
sum(E_CO)
identical(round(E_CO, 2), round(E_CO_2, 2))
identical(round(E_CO_3, 2), round(E_CO_4, 2))
identical(round(E_CO_3[, , 1], 2), round(E_CO_4[, , 1], 2))
dim(E_CO_3)
dim(E_CO_4)
# but
a <- unlist(lapply(1:41, function(i) {
  unlist(lapply(1:168, function(j) {
    identical(E_CO_3[, i, j], E_CO_4[, i, j])
  })
}))
unique(a)

# Estimation with list of vehicles
lpc <- list(pc1, pc1)
lef <- EmissionFactorsList(ef_cetesb("CO", "PC_G", agemax = ncol(pc1)))
E_COv2 <- emis(veh = lpc, lkm = net$lkm, ef = lef, speed = speed)

# top down
veh <- age_ldv(x = net$ldv[1:4], name = "PC_E25_1400", agemax = 4)
mil <- fkm$KM_PC_E25(1:4)
ef <- ef_cetesb("COd", "PC_G")[1:4]
emis(veh, units::set_units(mil, "km"), ef)

# group online
bus1 <- age_hdv(30, agemax = 4)
veh <- bus1
lkm <- units::set_units(400, "km")
speed <- 40
efco <- ef_cetesb("COd", "UB", agemax = 4)
lef <- ef_hdv_scaled(
  dfcol = as.numeric(efco),
  v = "Ubus",
  t = "Std",
  g = ">15 & <=18",
  eu = rep("IV", 4),
  gr = 0,
  l = 0.5,
  p = "CO"
)
for (i in 1:length(lef)) print(lef[[i]](10))
(a <- emis(veh = bus1, lkm = lkm, ef = efco, verbose = TRUE))
(b <- emis(veh = bus1, lkm = lkm, ef = efco, verbose = TRUE, fortran = TRUE))

## End(Not run)

---

**EmissionFactors**

*Construction function for class "EmissionFactors"*

**Description**

`EmissionFactors` returns a transformed object with class "EmissionFactors" and units g/km.

**Usage**

```r
EmissionFactors(x, mass = "g", dist = "km", ...)
```

## S3 method for class 'EmissionFactors'
print(x, ...)

## S3 method for class 'EmissionFactors'
summary(object, ...)

## S3 method for class 'EmissionFactors'
plot(
```
Arguments

x Object with class "data.frame", "matrix" or "numeric"

mass Character to be the time units as numerator, default "g" for grams

dist String indicating the units of the resulting distance in speed.

... ignored

object object with class "EmissionFactors"

pal Palette of colors available or the number of the position

rev Logical; to internally revert order of rgb color vectors.

fig1 par parameters for fig, par.

fig2 par parameters for fig, par.

fig3 par parameters for fig, par.

mai1 par parameters for mai, par.

mai2 par parameters for mai, par.

mai3 par parameters for mai, par.

bias positive number. Higher values give more widely spaced colors at the high end.

Value

Objects of class "EmissionFactors" or "units"

Examples

```r
## Not run:
data(fe2015)
names(fe2015)
class(fe2015)
df <- fe2015[fe2015$Pollutant=="CO", c(ncol(fe2015)-1, ncol(fe2015))]
ef1 <- EmissionFactors(df)
class(ef1)
summary(ef1)
plot(ef1)
print(ef1)

## End(Not run)
```
**Description**

EmissionFactorsList returns a transformed object with class "EmissionsFactorsList".

**Usage**

EmissionFactorsList(x, ...)

## S3 method for class 'EmissionFactorsList'
print(x, ..., default = FALSE)

## S3 method for class 'EmissionFactorsList'
summary(object, ...)

## S3 method for class 'EmissionFactorsList'
plot(x, ...)

**Arguments**

- **x**: Object with class "list"
- **...**: ignored
- **default**: Logical value. When TRUE prints default list, when FALSE prints messages with description of list
- **object**: Object with class "EmissionFactorsList"

**Value**

Objects of class "EmissionFactorsList"

**Examples**

## Not run:
data(fe2015)
names(fe2015)
class(fe2015)
df <- fe2015[fe2015$Pollutant=="CO", c(ncol(fe2015)-1,ncol(fe2015))]
ef1 <- EmissionFactorsList(df)
class(ef1)
length(ef1)
length(ef1[[1]])
summary(ef1)

## End(Not run)
Description

`Emissions` returns a transformed object with class "Emissions". The type of objects supported are of classes "matrix", "data.frame" and "numeric". If the class of the object is "matrix" this function returns a dataframe.

Usage

```r
Emissions(x, mass = "g", time, ...)

## S3 method for class 'Emissions'
print(x, ...)

## S3 method for class 'Emissions'
summary(object, ...)

## S3 method for class 'Emissions'
plot(
  x,
  pal = "colo_angelafaye_Coloured_sky_in",
  rev = FALSE,
  fig1 = c(0, 0.8, 0, 0.8),
  fig2 = c(0, 0.8, 0.55, 1),
  fig3 = c(0.7, 1, 0, 0.8),
  mai1 = c(0.2, 0.82, 0.82, 0.42),
  mai2 = c(1.3, 0.82, 0.82, 0.42),
  mai3 = c(0.7, 0.72, 0.82, 0.42),
  main = NULL,
  bias = 1.5,
  ...)
```

Arguments

- `x`: Object with class "data.frame", "matrix" or "numeric"
- `mass`: Character to be the time units as numerator, default "g" for grams
- `time`: Character to be the time units as denominator, eg "h"
- `...`: ignored
- `object`: object with class "Emissions"
- `pal`: Palette of colors available or the number of the position
- `rev`: Logical; to internally revert order of rgb color vectors.
- `fig1`: par parameters for fig, `par`.
fig2  par parameters for fig, \texttt{par}.

fig3  par parameters for fig, \texttt{par}.

mai1  par parameters for mai, \texttt{par}.

mai2  par parameters for mai, \texttt{par}.

mai3  par parameters for mai, \texttt{par}.

main  title of plot

bias  positive number. Higher values give more widely spaced colors at the high end.

Value

Objects of class "Emissions" or "units"

Examples

## Not run:
data(net)
data(pc_profile)
data(fe2015)
data(fkm)
PC_G <- c(33491, 22340, 24818, 31808, 46458, 28574, 28872, 37818, 49050, 87923,
133833, 138441, 142682, 171029, 151048, 115228, 98664, 126444, 101027,
84771, 55864, 36306, 21079, 20138, 17439, 7854, 2215, 656, 1262, 476, 512,
1181, 4991, 3711, 5653, 7039, 5839, 4257, 3824, 3068)
veh <- data.frame(PC_G = PC_G)
pc1 <- my_age(x = net$ldv, y = PC_G, name = "PC")
pcw <- temp_fact(net$ldv+net$hdv, pc_profile)
speed <- netspeed(pcw, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
pckm <- units::as_units(fkm[1](1:24), "km"); pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[12:24])
#vehicles newer than pre-euro
c1 <- fe2015[fe2015$Pollutant=="CO", ] #24 obs!!!
cod <- c(cod1[PC_G[1:24]]+c(cod1,cod2), cod1[PC_G[25:nrow(c1)]])
lef <- ef_ldv_scaled(co1, cod, v = "PC", cc = "<=1400",
f = "G", p = "CO", eu=co1$Euro_LDV)
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed, agemax = 41,
profile = pc_profile)
dim(E_CO) # streets x vehicle categories x hours x days
class(E_CO)
plot(E_CO)
####
Emissions(1)
Emissions(1, time = "h")

## End(Not run)
EmissionsArray

Construction function for class "EmissionsArray"

Description

EmissionsArray returns a transformed object with class "EmissionsArray" with 4 dimensions.

Usage

EmissionsArray(x, ...)

## S3 method for class 'EmissionsArray'
print(x, ...)

## S3 method for class 'EmissionsArray'
summary(object, ...)

## S3 method for class 'EmissionsArray'
plot(x, main = "average emissions", ...)

Arguments

x Object with class "data.frame", "matrix" or "numeric"
...
object object with class "EmissionsArray"
main Title for plot

Value

Objects of class "EmissionsArray"

Note

Future version of this function will return an Array of 3 dimensions.

Examples

## Not run:
data(net)
data(pc_profile)
data(fe2015)
data(fkm)
PC_G <- c(33491,22340,24818,31808,46458,28574,24856,28972,37818,49050,87923,
          133833,138441,142682,171029,151048,115228,98664,126444,101027,
          84771,55864,36306,21079,20138,17439,7854,2215,656,1262,476,512,
          1181,4991,3711,5653,7039,5839,4257,3824,3068)
veh <- data.frame(PC_G = PC_G)
emis_chem

Aggregate emissions by lumped groups in chemical mechanism

Description

emis_chem aggregates emissions by chemical mechanism and convert grams to mol. This function reads all hydrocarbons and respective criteria pollutants specified in ef_ldv_speed and ef_hdv_speed.

Usage

emis_chem(dfe, mechanism, colby, long = FALSE)

Arguments

dfe  data.frame with column ‘emissions’ in grams and ‘pollutant’ in long format. It is supposed that each line is the pollution of some region. Then the ‘coldby’ argument is for include the name of the region.


colby Character indicating column name for aggregating extra column. For instance, region or province.

long Logical. Do you want data in long format?
Value
data.frame with lumped groups by chemical mechanism. It transform emissions in grams to mol.

Note
This feature is experimental and the mapping of pollutants and lumped species may change in future. This function is converting the intial data.frame input into data.table. To have a comprehensive speciation is necessary enter with a data.frame with column 'emission' in long format including another column named 'pollutant' with species of NMHC, CO, NO, NO2, NH3, SO2, PM2.5 and coarse PM10.

Groups derived from gases has units 'mol' and from aerosols 'g'. The aerosol units for WRF-Chem are ug/m^2/s while for CMAQ and CAMx are g/s. So, leaving the units just in g, allow to make further change while providing flexibility for several models. TODO: Enter with wide data.frame, with each line as a each street, each column for pollutant.

See Also
ef_ldv_speed ef_hdv_speed speciate ef_evap

Examples
## Not run:
# CO
df <- data.frame(emission = Emissions(1:10))
df$pollutant = "CO"
emis_chem(dfe = df, "CBMZ_MOSAIC")
# hexanal
df$pollutant = "hexanal"
emis_chem(df, "CBMZ_MOSAIC")
# propadiene and NO2
df2 <- df1 <- df
df1$pollutant = "propadiene"
df2$pollutant = "NO2"
(dfe <- rbind(df1, df2))
emis_chem(dfe, "CBMZ_MOSAIC")
dfe$region <- rep(letters[1:2], 10)
emis_chem(dfe, "CBMZ_MOSAIC", "region")
emis_chem(dfe, "CBMZ_MOSAIC", "region", TRUE)

## End(Not run)
Usage

emis_chem2(df, mech, nx, na.rm = FALSE)

Arguments

- **df**: data.frame with emissions including columns "id" and "pol".
- **nx**: Character, colnames for emissions data, for instance "V1", "V2"...
- **na.rm**: Logical, to remove lines with NA from group.

Value

data.frame with lumped groups by chemical mechanism.

Note

- **CB05**: "ALD" "ALDX" "ETH" "HC3" "HC5" "HC8" "HCHO" "KET" "OL2" "OLI" "OLT" "TOL" "XYL"
- **CB05opt2**: "ALD2" "ALDX" "BENZENE" "ETH" "ETHA" "FORM" "IOLE" "OLE" "PAR" "TOL" "XYL"
- **RADM2**: "ALD" "ETH" "HC3" "HC5" "HC8" "HCHO" "KET" "MACR" "OL2" "OLI" "OLT" "TOL" "XYL"
- **RACM2**: "ACD" "ACE" "ACT" "ALD" "BALD" "BENZENE" "CCHO" "ETHENE" "HCHO" "IPRD" "MACR" "MEK" "OLE1" "OLE2" "RCHO"
- **CB4**: "ALD2" "ETH" "FORM" "OLE" "PAR" "TOL" "XYL"
- **S99**: "ACET" "ALK1" "ALK2" "ALK3" "ALK4" "ALK5" "AR01NBZ" "AR02" "BALD" "BENZENE" "CCHO" "ETHENE" "HCHO" "IPRD" "MACR" "MEK" "OLE1" "OLE2" "RCHO"
- **CS7**: "ALK3" "ALK4" "AR01" "AR02" "CCHO" "ETHE" "HCHO" "IPRD" "NROG" "OLE1" "OLE2" "PRD2" "RCHO"
- **S7**: "ACET" "ACYE" "ALK1" "ALK2" "ALK3" "ALK4" "ALK5" "AR01" "AR02" "BALD" "BENZ" "CCHO" "ETHE" "HCHO" "IPRD" "MACR" "MEK" "OLE1" "OLE2" "RCHO"
- **S7T**: "13BDE" "ACET" "ACRO" "ACYE" "ALK1" "ALK2" "ALK3" "ALK4" "ALK5" "AR01" "AR02" "B124" "BALD" "BENZ" "CCHO" "ETHE" "HCHO" "IPRD" "MACR" "MEK" "OLE1" "OLE2" "RCHO"
- **S11**: "ACET" "ACYL" "ALK1" "ALK2" "ALK3" "ALK4" "ALK5" "AR01" "AR02" "BALD" "BENZ" "CCHO" "ETHE" "HCHO" "IPRD" "MACR" "MEK" "OLE1" "OLE2" "RCHO"
- **S11D**: "ACET" "ACRO" "ACYL" "ALLENE" "BALD" "BENZ" "BUTDE13" "BUTENE1" "C2BENZ" "C2BUTE" "C2PENT" "C4RCHO1" "CCHO" "CROTALD" "ETACTYL" "ETHANE" "ETHENE" "HCHO" "HEXENE1" "ISOBUTEN" "M2C3" "M2C4" "M2C6" "M2C7" "M3C6" "M3C7" "MACR" "MEACTYL" "MEK" "MXYLENE" "NC1" "NC4" "NC5" "NC6" "NC7"
emis_china

"NC8" "NC9" "OLE2" "OTH2" "OTH4" "OTH5" "OXYLENE" "PENTEN1" "PROPALD" "PROPANE" "PROPENE" "PXYLENE" "RCHO" "STYRENE" "TMB123" "TMB124" "TMB135" "TOLUENE"

- **S16C**: "ACET" "ACETL" "ACRO" "ACYLS" "ALK3" "ALK4" "ALK5" "BALD" "BENZ" "BUT13" "BZ123" "BZ124" "BZ135" "C2BEN" "ETCHO" "ETHAN" "ETHEN" "HCHO" "MACR" "MECHO" "MEK" "MXYL" "NC4" "OLE1" "OLE2" "OLE3" "OLE4" "OLEA1" "OTH1" "OTH3" "OTH4" "OXYL" "PROP" "PROPE" "PXYL" "RCHO" "STYRS" "TOLU"

- **S18B**: "ACET" "ACETL" "ACRO" "ACYLS" "ALK3" "ALK4" "ALK5" "BALD" "BENZ" "BUT13" "BZ123" "BZ124" "BZ135" "C2BEN" "ETCHO" "ETHAN" "ETHEN" "HCHO" "MACR" "MECHO" "MEK" "MXYL" "NC4" "OLE1" "OLE2" "OLE3" "OLE4" "OLEA1" "OTH1" "OTH3" "OTH4" "OXYL" "PROP" "PROPE" "PXYL" "RCHO" "STYRS" "TOLU"

References


See Also

speciate

Examples

```r
{
  id <- 1:2
df <- data.frame(V1 = 1:2, V2 = 1:2)
dx <- speciate(x = df,
                spec = "nmhc",
                fuel = "E25",
                veh = "LDV",
                eu = "Exhaust")
dx$id <- rep(id, length(unique(dx$pol)))
names(dx)
vocE25EX <- emis_chem2(df = dx,
                        mech = "CB05",
                        nx = c("V1", "V2"))
}
```

emis_china  Estimation with Chinese factors

Description

Emissions estimates
Usage

emis_china(
  x,
  lkm,
  tfs,
  v = "PV",
  t = "Small",
  f = "G",
  standard,
  s,
  speed,
  te,
  hu,
  h,
  yeardet = 2016,
  p,
  verbose = TRUE,
  array = FALSE
)

Arguments

x          Vehicles data.frame
lkm        Length of each link in km
tfs        temporal factor
v          Character; category vehicle: "PV" for Passenger Vehicles or 'Trucks'
f          Character; fuel: "G", "D", "CNG", "ALL"
standard   Character vector; "PRE", "I", "II", "III", "IV", "V".
s          Sulhur in ppm
speed      Speed (length nrow x)
te         Temperature (length tfs)
hu         Humidity (length tfs)
h          Altitude (length nrow x)
yeardet    Year, default 2016
p          Character; pollutant: "CO", "NOx", "HC", "PM", "Evaporative_driving" or "Evaporative_parking"
verbose    Logical to show more info
array      Logical to return EmissionsArray or not

Value

long data.frame
emis_cold

See Also

Other China: ef_china_det(), ef_china_hu(), ef_china_h(), ef_china_long(), ef_china_speed(), ef_china_s(), ef_china_te(), ef_china_th(), ef_china(), emis_long()

Examples

{
  ef_china_h(h = 1600, p = "CO")
}

emis_cold

Estimation of cold start emissions hourly for the of the week

Description

emis_cold emissions are estimated as the product of the vehicles on a road, length of the road, emission factor evaluated at the respective speed. The estimation considers the beta parameter, the fraction of mileage driven

Usage

emis_cold(
  veh,
  lkm,
  ef,
  ef_cold,
  beta,
  speed = 34,
  agemax = if (!inherits(x = veh, what = "list")) {
    ncol(veh)
  } else {
    ncol(veh[[1]])
  },
  profile,
  simplify = FALSE,
  hour = nrow(profile),
  day = ncol(profile),
  array = TRUE,
  verbose = FALSE
)

Arguments

veh "Vehicles" data-frame or list of "Vehicles" data-frame. Each data-frame as number of columns matching the age distribution of that type of vehicle. The number of rows is equal to the number of streets link
emis_cold

1km Length of each link
ef List of functions of emission factors of vehicular categories
efcold List of functions of cold start emission factors of vehicular categories
beta Dataframe with the hourly cold-start distribution to each day of the period.
Number of rows are hours and columns are days
speed Speed data-frame with number of columns as hours
agemax Age of oldest vehicles for that category
profile Numerical or dataframe with nrows equal to 24 and ncol 7 day of the week
simplify Logical; to determine if EmissionsArray should les dimensions, being streets, vehicle categories and hours or default (streets, vehicle categories, hours and days). Default is FALSE to avoid break old code, but the recommendation is that new estimations use this parameter as TRUE
hour Number of considered hours in estimation
day Number of considered days in estimation
array Deprecated! emis_cold returns only arrays. When TRUE and veh is not a list, expects a profile as a dataframe producing an array with dimensions (streets x columns x hours x days)
verbose Logical; To show more information

Value

EmissionsArray g/h

Examples

### Not run:
# Do not run
data(net)
data(pc_profile)
data(fe2015)
data(fkm)
data(pc_cold)
pcf <- as.data.frame(cbind(pc_cold, pc_cold, pc_cold, pc_cold, pc_cold, pc_cold, pc_cold))
PC_G <- c(33491, 22340, 24818, 31808, 46458, 28574, 24856, 28972, 37818, 49050, 87923,
  133833, 138441, 142682, 171029, 151048, 115228, 98664, 126444, 101027,
  84771, 55864, 36306, 21079, 20138, 17439, 7854, 2215, 656, 1262, 476, 512,
  1181, 4991, 3711, 5653, 7039, 5839, 4257, 3824, 3068)
veh <- data.frame(PC_G = PC_G)
pc1 <- my_age(x = net$ldv, y = PC_G, name = "PC")
pcw <- temp_fact(net$ldv + net$hdv, pc_profile)
speed <- netspeed(pcw, net$ps, net$fFs, net$capacity, net$lkm, alpha = 1)
pckm <- units::set_units(fkm[[1]][:24], "km"); pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[12:24])
#vehicles newer than pre-euro
col <- fe2015[fe2015$Pollutant=="CO", ] #24 obs!!!
emis_cold_td

Estimation of cold start emissions with top-down approach

Description

emis_cold_td estimates cold start emissions with a top-down approach. This is, annual or monthly emissions or region. Specifically, the emissions are estimated for each row of the simple feature (row of the spatial feature).

In general was designed so that each simple feature is a region with different average monthly temperature. This function, as other in this package, adapts to the class of the input data, providing flexibility to the user.

Usage

emis_cold_td(
eh, lkm, ef, efcold, beta, pro_month, params, verbose = FALSE, fortran = FALSE, nt = ifelse(check_nt() == 1, 1, check_nt()/2)
)

Arguments
veh "Vehicles" data-frame or spatial feature, where columns are the age distribution of that vehicle, and rows each simple feature or region. The number of rows is equal to the number of streets link.
lkm Numeric; mileage by the age of use of each vehicle.
ef Numeric; emission factor with
efcold Data.frame. When it is a data.frame, each column is for each type of vehicle by age of use, rows are each simple feature. When you have emission factors for each month, the order should a data.frame in a long format, as returned by ef_ldv_cold.
beta Data.frame with the fraction of cold starts. The rows are the fraction for each spatial feature or subregion, the columns are the age of use of vehicle.
pro_month Numeric; monthly profile to distribute annual mileage in each month.
params List of parameters; Add columns with information to returning data.frame
verbose Logical; To show more information
fortran Logical; to try the fortran calculation.
nt Integer; Number of threads which must be lower than max available. See check_nt. Only when fortran = TRUE

Value
Emissions data.frame

See Also
ef_ldv_cold

Examples
## Not run:
veh <- age_ldv(1:10, agemax = 8)
euros <- c("V", "V", "IV", "III", "II", "I", "PRE", "PRE")
dt <- matrix(rep(2:25, 5), ncol = 12, nrow = 10) # 12 months, 10 rows
row.names(dt) <- paste0("Simple_Feature.", 1:10)
efc <- ef_lv_cold(ta = dt, cc = "<=1400", f = "G", eu = euros, p = "CO", speed = Speed(34))
efh <- ef_lv_speed(
  v = "PC", t = "4S", cc = "<=1400", f = "G",
  eu = euros, p = "CO", speed = Speed(runif(nrow(veh), 15, 40)))
lkm <- units::as_units(18:11, "km") * 1000
cold_lkm <- cold_mileage(ltrip = units::as_units(20, "km"), ta = celsius(dt))
names(cold_lkm) <- paste0("Month.", 1:12)
veh_month <- c(rep(8, 1), rep(10, 5), 9, rep(10, 5))

system.time(
  a <- emis_cold_td(
    veh = veh,
    lkm = lkm,
    ef = efh[1, ],
    efcold = efc[1:10, ],
    beta = cold_lkm[, 1],
    verbose = TRUE
  )
)

system.time(
  a2 <- emis_cold_td(
    veh = veh,
    lkm = lkm,
    ef = efh[1, ],
    efcold = efc[1:10, ],
    beta = cold_lkm[, 1],
    verbose = TRUE,
    fortran = TRUE
  )
)  # emistd2coldf.f95
a$emissions <- round(a$emissions, 8)
a2$emissions <- round(a2$emissions, 8)
identical(a, a2)

# Adding parameters
emis_cold_td(
  veh = veh,
  lkm = lkm,
  ef = efh[1, ],
  efcold = efc[1:10, ],
  beta = cold_lkm[, 1],
  verbose = TRUE,
  params = list(  
    paste0("data.", 1:10),
    "moredata"
  )
)

system.time(
  aa <- emis_cold_td(
    veh = veh,
    lkm = lkm,
    ef = efh,
    params = list(  
      paste0("data.", 1:10),
      "moredata"
    )
  )
)
emis_det = efcold = efc,
beta = cold_lkm,
pro_month = veh_month,
verbose = TRUE
)
)

system.time(
  aa2 <- emis_cold_td(
    veh = veh,
    lkm = lkm,
    ef = efh,
    efcold = efc,
    beta = cold_lkm,
    pro_month = veh_month,
    verbose = TRUE,
    fortran = TRUE
  )
) # emistd5coldf.f95

aa$emissions <- round(aa$emissions, 8)

aa2$emissions <- round(aa2$emissions, 8)

identical(aa, aa2)

## End(Not run)

emis_det  Determine deterioration factors for urban conditions

Description

emis_det returns deterioration factors. The emission factors comes from the guidelines for developing emission factors of the EMEP/EEA air pollutant emission inventory guidebook http://www.eea.europa.eu/themes/air/emepeea-air-pollutant-emission-inventory-guidebook This function subset an internal database of emission factors with each argument

Usage

emis_det(
  po,
  cc,
  eu,
  speed = Speed(18.9),
  km,
  verbose = FALSE,
  show.equation = FALSE
)
Arguments

- **po**: Character; Pollutant "CO", "NOx" or "HC"
- **cc**: Character; Size of engine in cc covering "<=1400", "1400_2000" or ">2000"
- **eu**: Character; Euro standard: "I", "II", "III", "IV", "V", "VI", "Vlc"
- **speed**: Numeric; Speed to return Number of emission factor and not a function. It needs units in km/h
- **km**: Numeric; accumulated mileage in km.
- **verbose**: Logical; To show more information
- **show.equation**: Option to see or not the equation parameters

Value

It returns a numeric vector representing the increase in emissions due to normal deterioring

Note

The deterioration factors functions are available for technologies euro "II", "III" and "IV". In order to cover all euro technologies, this function assumes that the deterioration function of "III" and "IV" applies for "V", "VI" and "Vlc". However, as these technologies are relative new, accumulated mileage is low and hence, deterioration factors small.

Examples

```r
## Not run:
data(fkm)
pckm <- fkm[[1]][1:24]; pckma <- cumsum(pckm)
km <- units::set_units(pckma[1:11], km)
# length eu = length km = 1
emis_det(po = "CO", cc = "<=1400", eu = "III", km = km[5], show.equation = TRUE)
# length eu = length km = 1, length speed > 1
emis_det(po = "CO", cc = "<=1400", eu = "III", km = km[5], speed = Speed(1:10))
# length km != length eu error
# (cod1 <- emis_det(po = "CO", cc = "<=1400", eu = c("III", "IV"), speed = Speed(30),
# km = km[4]))
# length eu = 1 length km > 1
emis_det(po = "CO", cc = "<=1400", eu = "III", km = km)
# length eu = 2, length km = 2 (if different length, error!)
(cod1 <- emis_det(po = "CO", cc = "<=1400", eu = c("III", "IV"), km = km[4:5]))
# length eu = 2, length km = 2, length speed > 1
(cod1 <- emis_det(po = "CO", cc = "<=1400", eu = c("III", "IV"), speed = Speed(0:130),
km = km[4:5]))
# length eu = 2, length km = 2, length speed > 1
(cod1 <- emis_det(po = "CO", cc = "<=1400", eu = euros, speed = Speed(1:100),
km = km[1:10]))
cod1 <- as.matrix(cod1[, 1:11])
filled.contour(cod1, col = cptcity::cpt(6277, n = 20))
filled.contour(cod1, col = cptcity::lucky(n = 19))
euro <- c(rep("V", 5), rep("IV", 5), "III")
```
emis_dist <- rbind(euro, euro)
(cod1 <- emis_det(po = "CO", cc = "<=1400", eu = euros, km = km))

## End(Not run)

emis_dist

Allocate emissions into spatial objects (street emis to grid)

Description

emis_dist allocates emissions proportionally to each feature. "Spatial" objects are converter to "sf" objects. Currently, 'LINESTRING' or 'MULTILINESTRING' supported. The emissions are distributed in each street.

Usage

emis_dist(gy, spobj, pro, osm, verbose = FALSE)

Arguments

gy Numeric; a unique total (top-down)
spobj A spatial data frame of class "sp" or "sf". When class is "sp" it is transformed to "sf".
pro Matrix or data-frame profiles, for instance, pc_profile.
osm Numeric; vector of length 5, for instance, c(5, 3, 2, 1, 1). The first element covers 'motorway' and 'motorway_link'. The second element covers 'trunk' and 'trunk_link'. The third element covers 'primary' and 'primary_link'. The fourth element covers 'secondary' and 'secondary_link'. The fifth element covers 'tertiary' and 'tertiary_link'.
verbose Logical; to show more info.

Note

When spobj is a 'Spatial' object (class of sp), they are converted into 'sf'.

Examples

## Not run:
data(net)
data(pc_profile)
po <- 1000
t1 <- emis_dist(gy = po, spobj = net)
head(t1)
sum(t1$gy)
#t1 <- emis_dist(gy = po, spobj = net, osm = c(5, 3, 2, 1, 1) )
t1 <- emis_dist(gy = po, spobj = net, pro = pc_profile)

## End(Not run)
emis_emfac

Emission calculation based on EMFAC emission factors

Description

emis_emfac estimates emissions based on an emission factors database from EMFAC. You must download the emission factors from EMFAC website.

Usage

emis_emfac(  
  ef,  
  veh,  
  lkm,  
  speed,  
  vehname,  
  pol = "CO_RUNEX",  
  modelyear = 2021:1982,  
  noyear = 2022,  
  hours = paste0("S", 1:24),  
  vkm = TRUE,  
  verbose = TRUE  
)

Arguments

- ef: Character path to EMFAC ef (g/miles)
- veh: Vehicles data.frame
- lkm: Distance per street-link in miles
- speed: Speed data.frame in miles/hour
- vehname: numeric vector for heavy good vehicles or trucks
- pol: character, "CO_RUNEX"
- modelyear: numeric vector, 2021:1982
- noyear: newest numeric year to take out from ef
- hours: Character, name of hours in speed, paste0("S", 1:24) data-frame profile for passenger cars, 24 hours only.
- vkm: logical, to return vkm
- verbose: logical, to show more information

Value

data.table with emission estimation in long format
emis_evap

Estimation of evaporative emissions

Description

emis_evap estimates evaporative emissions from EMEP/EEA emission guidelines.

Usage

emis_evap(
  veh,
  x,
  ed,
  hotfi,
  hotc,
  warmc,
  carb = 0,
  p,
  params,
  pro_month,
  verbose = FALSE
)

Arguments

veh Numeric or data.frame of Vehicles with units 'veh'.

x Numeric which can be either, daily mileage by age of use with units 'lkm', number of trips or number of proc. When it has units 'lkm', all the emission factors must be in 'g/km'. When ed is in g/day, x it is the number of days (without units). When hotfi, hotc or warmc are in g/trip, x it is the number of trips (without units). When hotfi, hotc or warmc are in g/proced, x it is the number of proced (without units).

ed average daily evaporative emissions. If x has units 'lkm', the units of ed must be 'g/km', other case, this are simply g/day (without units).

hotfi average hot running losses or soak evaporative factor for vehicles with fuel injection and returnless fuel systems. If x has units 'lkm', the units of ed must be 'g/km', other case, this is simply g/trip or g/proced

hotc average running losses or soak evaporative factor for vehicles with carburetor or fuel return system for vehicles with fuel injection and returnless fuel systems. If x has units 'lkm', the units of ed must be 'g/km',
emis_evap2

Description

emis_evap2 performs the estimation of evaporative emissions from EMEP/EEA emission guidelines with Tier 2.

Value

numeric vector of emission estimation in grams

Note

When veh is a "Vehicles" data.frame, emission factors are evaluated till the number of columns of veh. For instance, if the length of the emission factor is 20 but the number of columns of veh is 10, the 10 first emission factors are used.

References


See Also

ef_evap

Examples

## Not run:
(a <- Vehicles(1:10))
(lkm <- units::as_units(1:10, "km"))
(ef <- EmissionFactors(1:10))
(ev <- emis_evap(veh = a, x = lkm, hotfi = ef))
## End(Not run)
Usage

```r
emis_evap2(
  veh,
  name,
  size,
  fuel,
  aged,
  nd4,
  nd3,
  nd2,
  nd1,
  hs_nd4,
  hs_nd3,
  hs_nd2,
  hs_nd1,
  rl_nd4,
  rl_nd3,
  rl_nd2,
  rl_nd1,
  d_nd4,
  d_nd3,
  d_nd2,
  d_nd1
)
```

Arguments

- **veh**: Total number of vehicles by age of use. If is a list of 'Vehicles' data-frames, it will sum the columns of the eight element of the list representing the 8th hour. It was chosen this hour because it is morning rush hour but the user can adapt the data to this function.
- **name**: Character of type of vehicle.
- **size**: Character of size of vehicle.
- **fuel**: Character of fuel of vehicle.
- **aged**: Age distribution vector. E.g.: 1:40
- **nd4**: Number of days with temperature between 20 and 35 Celsius degrees.
- **nd3**: Number of days with temperature between 10 and 25 Celsius degrees.
- **nd2**: Number of days with temperature between 0 and 15 Celsius degrees.
- **nd1**: Number of days with temperature between -5 and 10 Celsius degrees.
- **hs_nd4**: Average daily hot-soak evaporative emissions for days with temperature between 20 and 35 Celsius degrees.
- **hs_nd3**: Average daily hot-soak evaporative emissions for days with temperature between 10 and 25 Celsius degrees.
- **hs_nd2**: Average daily hot-soak evaporative emissions for days with temperature between 0 and 15 Celsius degrees.
emis_evap2

hs_nd1 average daily hot-soak evaporative emissions for days with temperature between -5 and 10 Celsius degrees
r1_nd4 average daily running losses evaporative emissions for days with temperature between 20 and 35 Celsius degrees
r1_nd3 average daily running losses evaporative emissions for days with temperature between 10 and 25 Celsius degrees
r1_nd2 average daily running losses evaporative emissions for days with temperature between 0 and 15 Celsius degrees
r1_nd1 average daily running losses evaporative emissions for days with temperature between -5 and 10 Celsius degrees
d_nd4 average daily diurnal evaporative emissions for days with temperature between 20 and 35 Celsius degrees
d_nd3 average daily diurnal evaporative emissions for days with temperature between 10 and 25 Celsius degrees
d_nd2 average daily diurnal evaporative emissions for days with temperature between 0 and 15 Celsius degrees
d_nd1 average daily diurnal evaporative emissions for days with temperature between -5 and 10 Celsius degrees

Value
dataframe of emission estimation in grams/days

References

Examples
## Not run:
data(net)
PC_G <- c(33491,22340,24818,31808,46458,28574,24856,28972,37818,49050,87923,
          133833,138441,142682,171029,151048,115228,98664,126444,101027,
          84771,55864,36306,21079,28138,17439,7854,2215,656,1262,476,512,
          1181,4991,3711,5653,7839, 5839,4257,3824,3068)
veh <- data.frame(PC_G = PC_G)
pc1 <- my_age(x = net$ldv, y = PC_G, name = "PC")
ef1 <- ef_evap(ef = "erhotc", v = "PC", cc = "<=1400", dt = "0_15", ca = "no")
dfe <- emis_evap2(veh = pc1,
                name = "PC",
                size = "<=1400",
                fuel = "G",
                aged = 1:ncol(pc1),
                nd4 = 10,
                nd3 = 4,
                nd2 = 2,
                nd1 = 1,
emis_grid

Allocate emissions into a grid returning point emissions or flux

Description

tt


tt

It is
Usage

emis_grid(spobj = net, g, sr, type = "lines", FN = "sum", flux = TRUE, k = 1)

Arguments

spobj A spatial dataframe of class "sp" or "sf". When class is "sp" it is transformed to "sf".

g A grid with class "SpatialPolygonsDataFrame" or "sf".

sr Spatial reference e.g: 31983. It is required if spobj and g are not projected. Please, see http://spatialreference.org/.

type type of geometry: "lines", "points" or "polygons".

FN Character indicating the function. Default is "sum"

flux Logical, if TRUE, it return flux (mass / area / time (implicit)) in a polygon grid, if false, mass / time (implicit) as points, in a similar fashion as EDGAR provide data.

k Numeric to multiply emissions

Note

1) If flux = TRUE (default), emissions are flux = mass / area / time (implicit), as polygons. If flux = FALSE, emissions are mass / time (implicit), as points. Time units are not displayed because each use can have different time units for instance, year, month, hour second, etc.

2) Therefore, it is good practice to have time units in 'spobj'. This implies that spobj MUST include units!.

3) In order to check the sum of the emissions, you must calculate the grid-area in km^2 and multiply by each column of the resulting emissions grid, and then sum.

4) If FN = "sum", is mass conservative!.

Examples

## Not run:
data(net)
g <- make_grid(net, 1/102.47/2) #500m in degrees
names(net)
netsf <- sf::st_as_sf(net)
netg <- emis_grid(spobj = netsf[, c("ldv", "hdv")], g = g, sr= 31983)
plot(netg["ldv"],
     axes = TRUE,
     graticule = TRUE,
     bg = "black",
     lty = 0)
g <- sf::st_make_grid(net, 1/102.47/2, square = FALSE) #500m in degrees
g <- st_sf(i =1, geometry = g)
netg <- emis_grid(spobj = netsf[, c("ldv", "hdv")], g = g, sr= 31983)
plot(netg["ldv"],
     axes = TRUE,
     graticule = TRUE,
emis_hot_td

Estimation of hot exhaust emissions with a top-down approach

Description

emis_hot_td estimates cold start emissions with a top-down approach. This is, annual or monthly emissions or region. Specifically, the emissions are estimated for the row of the simple feature (row of the spatial feature).

In general was designed so that each simple feature is a region with different average monthly temperature. This function, as others in this package, adapts to the class of the input data, providing flexibility to the user.

Usage

emis_hot_td(
  veh,  # "Vehicles" data-frame or spatial feature, where columns are the age distribution of that vehicle, and rows each simple feature or region.
  lkm,  # Numeric; mileage by the age of use of each vehicle.
  ef,
  pro_month,
  params,
  verbose = FALSE,
  fortran = FALSE,
  nt = ifelse(check_nt() == 1, 1, check_nt()/2)
)

Arguments

veh  

lkm  

emis_grid(spobj = netsf[, c("ldv", "hdv")], g = g, sr= 31983, FN = "mean")
plot(netg["ldv"], axes = TRUE)
plot(netg["hdv"], axes = TRUE)
netg <- emis_grid(spobj = netsf[, c("ldv", "hdv")], g = g, sr= 31983, flux = FALSE)
plot(netg["ldv"],
  axes = TRUE,
  pch = 16,
  pal = cptcity::cpt(colorRampPalette= TRUE,
                     rev = TRUE),
  cex = 3)

## End(Not run)
ef  Numeric or data.frame; emission factors. When it is a data.frame number of rows can be for each region, or also, each region repeated along 12 months. For instance, if you have 10 regions the number of rows of ef can also be 120 (10 * 120). when you have emission factors that varies with month, see ef_china.

pro_month  Numeric or data.frame; monthly profile to distribute annual mileage in each month. When it is a data.frame, each region (row) can have a different monthly profile.

params  List of parameters; Add columns with information to returning data.frame

verbose  Logical; To show more information

fortran  Logical; to try the fortran calculation.

nt  Integer; Number of threads which must be lower than max available. See check_nt. Only when fortran = TRUE

Details
List to make easier to use this function.

1. ‘pro_month’ is data.frame AND rows of ‘ef’ and ‘veh’ are equal.
2. ‘pro_month’ is numeric AND rows of ‘ef’ and ‘veh’ are equal.
3. ‘pro_month’ is data.frame AND rows of ‘ef’ is 12X rows of ‘veh’.
4. ‘pro_month’ is numeric AND rows of ‘ef’ is 12X rows of ‘veh’.
5. ‘pro_month’ is data.frame AND class of ‘ef’ is ‘units’.
6. ‘pro_month’ is numeric AND class of ‘ef’ is ‘units’.
7. NO ‘pro_month’ AND class of ‘ef’ is ‘units’.
8. NO ‘pro_month’ AND ‘ef’ is data.frame.
9. ‘pro_month’ is numeric AND rows of ‘ef’ is 12 (monthly ‘ef’).

Value
Emissions data.frame

See Also

ef_ldv_speed ef_china

Examples

```r
## Not run:
# Do not run
euros <- c("V", "V", "IV", "III", "II", "I", "PRE", "PRE")
efh <- ef_ldv_speed(
  v = "PC", t = "4S", cc = "c=1400", f = "G",
  eu = euros, p = "CO", speed = Speed(34)
)
lkm <- units::as_units(c(20:13), "km") * 1000
veh <- age_ldv(1:10, agemax = 8)
system.time(
```
a <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = EmissionFactors(as.numeric(efh[, 1:8])),
    verbose = TRUE
)

system.time(
a2 <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = EmissionFactors(as.numeric(efh[, 1:8])),
    verbose = TRUE,
    fortran = TRUE
)
) # emistd7f.f95
identical(a, a2)

# adding columns
emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = EmissionFactors(as.numeric(efh[, 1:8])),
    verbose = TRUE,
    params = list(paste0("data\_", 1:10), "moredata")
)

# monthly profile (numeric) with numeric ef
veh\_month <- c(rep(8, 1), rep(10, 5), 9, rep(10, 5))

system.time(
aa <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = EmissionFactors(as.numeric(efh[, 1:8])),
    pro\_month = veh\_month,
    verbose = TRUE
)
)

system.time(
aa2 <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = EmissionFactors(as.numeric(efh[, 1:8])),
    pro\_month = veh\_month,
    verbose = TRUE,
    fortran = TRUE
)
) # emistd5f.f95

aa\$emissions <- round(aa\$emissions, 8)
aa2\$emissions <- round(aa2\$emissions, 8)
identical(aa, aa2)

# monthly profile (numeric) with data.frame ef
emis_hot_td

veh_month <- c(rep(8, 1), rep(10, 5), 9, rep(10, 5))
def <- matrix(EmissionFactors(as.numeric(efh[, 1:8])),
nrow = nrow(veh), ncol = ncol(veh), byrow = TRUE)
def <- EmissionFactors(def)

system.time(
  aa <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = def,
    pro_month = veh_month,
    verbose = TRUE
  )
)

system.time(
  aa2 <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = def,
    pro_month = veh_month,
    verbose = TRUE,
    fortran = TRUE
  )
)

# emistd1f.f95
aa$emissions <- round(aa$emissions, 8)
aa2$emissions <- round(aa2$emissions, 8)
identical(aa, aa2)

# monthly profile (data.frame)
dfm <- matrix(c(rep(8, 1), rep(10, 5), 9, rep(10, 5)),
  nrow = 10, ncol = 12,
  byrow = TRUE)

system.time(
  aa <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = EmissionFactors(as.numeric(efh[, 1:8])),
    pro_month = dfm,
    verbose = TRUE
  )
)

system.time(
  aa2 <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = EmissionFactors(as.numeric(efh[, 1:8])),
    pro_month = dfm,
    verbose = TRUE,
    fortran = TRUE
  )
)

# emistd6f.f95
aa$emissions <- round(aa$emissions, 2)
aa2$emissions <- round(aa2$emissions, 2)
identical(aa, aa2)

# Suppose that we have a EmissionsFactor data.frame with number of rows for each month
# number of rows are 10 regions
# number of columns are 12 months
tem <- runif(n = 6 * 10, min = -10, max = 35)
temp <- c(rev(tem[order(tem)]), tem[order(tem)])
plot(temp)
dftemp <- celsius(matrix(temp, ncol = 12))
dfef <- ef_evap(
  ef = c(rep("eshotfi", 8)),
  v = "PC",
  cc = "<=1400",
  dt = dftemp,
  show = F,
  ca = "small",
  ltrip = units::set_units(10, km),
  pollutant = "NMHC"
)
dim(dfef) # 120 rows and 9 columns, 8 ef (g/km) and 1 for month
system.time(
  aa <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = dfef,
    pro_month = veh_month,
    verbose = TRUE
  )
)

system.time(
  aa2 <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = dfef,
    pro_month = veh_month,
    verbose = TRUE,
    fortran = TRUE
  )
)
# emistd3f.f95
aa$emissions <- round(aa$emissions, 2)
aa2$emissions <- round(aa2$emissions, 2)
identical(aa, aa2)
plot(aggregate(aa$emissions, by = list(aa$month), sum)$x)

# Suppose that we have a EmissionsFactor data.frame with number of rows for each month
# monthly profile (data.frame)
system.time(
  aa <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = dfef,
    pro_month = dfm,
emis_long

verbose = TRUE
)
)
system.time(
  aa2 <- emis_hot_td(
    veh = veh,
    lkm = lkm,
    ef = ddef,
    pro_month = dfm,
    verbose = TRUE,
    fortran = TRUE
  )
)
) # emistd4f.f95
aa$emissions <- round(aa$emissions, 8)
aa2$emissions <- round(aa2$emissions, 8)
identical(aa, aa2)
plot(aggregate(aa$emissions, by = list(aa$month), sum)$x)

## End(Not run)

---

# Description

Estimation with long format

# Usage

emis_long(x, lkm, ef, tfs, speed, verbose = TRUE, array = FALSE)

# Arguments

x: Vehicles data.frame. x repeats down for each hour
lkm: Length of each link in km. lkm repeats down for each hour
ef: data.frame. ef repeats down for each hour
tfs: temporal factor
speed: Speed data.frame (nrow x)
verbose: Logical to show more info
array: Logical to return EmissionsArray or not

# Value

long data.frame
See Also

Other China: ef_china_det(), ef_china_hu(), ef_china_h(), ef_china_long(), ef_china_speed(), ef_china_s(), ef_china_te(), ef_china_th(), ef_china(), emis_china()

Examples

```r
def(data)  
  net <- net[1:100, ]  
data(pc_profile)  
x <- age_ldv(net$ldv)  
pc_week <- temp_fact(net$ldv+net$hdv, pc_profile[[1]])  
df <- netspeed(pc_week,  
  net$ps,  
  net$ffs,  
  net$capacity,  
  net$lkm,  
  alpha = 1)  

s <- do.call("rbind", lapply(1:ncol(df), function(i) {  
  as.data.frame(replicate(ncol(x), df[, i]))  
}))  

ef <- ef_wear(wear = "tyre",  
  type = "PC",  
  pol = "PM10",  
  speed = as.data.frame(s))  

e <- emis_long(x = x,  
  lkm = net$lkm,  
  ef = ef,  
  tfs = pc_profile[[1]],  
  speed = df)  

ea <- emis_long(x = x,  
  lkm = net$lkm,  
  ef = ef,  
  tfs = pc_profile[[1]],  
  speed = df,  
  array = TRUE)
```

emis_merge

Merge several emissions files returning data-frames or 'sf' of lines

Description

emis_merge reads rds files and returns a data-frame or an object of 'spatial feature' of streets, merging several files.
emis_merge

Usage

emis_merge(
  pol = "CO",
  what = "STREETS.rds",
  streets = T,
  net,
  FN = "sum",
  ignore,
  path = "emi",
  crs,
  under = "after",
  as_list = FALSE,
  k = 1,
  verbose = TRUE
)

Arguments

pol Character. Pollutant.
what Character. Word to search the emissions names, "STREETS", "DF" or whatever name. It is important to include the extension ".rds". For instance, If you have several files "XX_C0_STREETS.rds", what should be "STREETS.rds"
streets Logical. If true, emis_merge will read the street emissions created with emis_post by "streets_wide", returning an object with class 'sf'. If false, it will read the emissions data-frame and rbind them.
net 'Spatial feature' or 'SpatialLinesDataFrame' with the streets. It is expected that the number of rows is equal to the number of rows of street emissions. If not, the function will stop.
FN Character indicating the function. Default is "sum"
ignore Character; Which pollutants or other charavter would you like to remove?
path Character. Path where emissions are located
 CRS coordinate reference system in numeric format from http://spatialreference.org/ to transform/project spatial data using sf::st_transform
under "Character": "after" when you stored your pollutant x as 'X_' "before" when '_X' and "none" for merging directly the files.
as_list "Logical"; for returning the results as list or not.
k factor
verbose Logical to display more information or not. Default is TRUE

Value

'Spatial feature' of lines or a dataframe of emissions
Examples

```
## Not run:
# Do not run

## End(Not run)
```

---

**emis_order**

*Re-order the emission to match specific hours and days*

**Description**

Emissions are usually estimated for a year, 24 hours, or one week from Monday to Sunday (with 168 hours). This depends on the availability of traffic data. When an air quality simulation is going to be done, they cover specific periods of time. For instance, WRF Chem emissions files support periods of time, or two emissions sets for a representative day (0-12z 12-0z). Also, a WRF Chem simulation scan starts a Thursday at 00:00 UTC, covers 271 hours of simulations, but hour emissions are in local time and cover only 168 hours starting on Monday. This function tries to transform our emissions in local time to the desired UTC time, by recycling the local emissions.

**Usage**

```r
emis_order(
x,  
lt_emissions,  
start_utc_time, 
desired_length, 
tz_lt = Sys.timezone(), 
seconds = 0, 
k = 1, 
net, 
verbose = TRUE
)
```

**Arguments**

- **x**
  - one of the following:
    - Spatial object of class "Spatial". Columns are hourly emissions.
    - Spatial Object of class "sf". Columns are hourly emissions.
    - "data.frame", "matrix" or "Emissions".
  - In all cases, columns are hourly emissions.

- **lt_emissions**
  - Local time of the emissions at the first hour. It must be the **before** time of start_utc_time. For instance, if start_utc_time is 2020-02-02 00:00, and your emissions starts Monday at 00:00, your lt_emissions must be 2020-01-27 00:00. The argument tz_lt will detect your current local time zone and do the rest for you.
emis_order

start_utc_time
UTC time for the desired first hour. For instance, the first hour of the namelist.input for WRF.

desired_length
Integer; length to recycle or subset local emissions. For instance, the length of the WRF Chem simulations, states at namelist.input.

tz_lt
Character, Time zone of the local emissions. Default value is derived from Sys.timezone(), however, it accepts any other. If you enter a wrong tz, this function will show you a menu to choose one of the 697 time zones available.

seconds
Number of seconds to add

k
Numeric, factor.

net
SpatialLinesDataFrame or Spatial Feature of "LINESTRING".

verbose
Logical, to show more information, default is TRUE.

Value

sf or data.frame

See Also

GriddedEmissionsArray

Examples

## Not run:
#do not run
data(net)
data(pc_profile)
data(fe2015)
data(fkm)
PC_G <- c(33491, 22340, 24818, 31808, 46458, 28574, 24856, 28972, 37818, 49058, 87923,
133833, 138441, 142682, 171029, 151048, 115228, 98664, 126444, 101027,
84771, 55864, 36306, 21079, 20138, 17439, 7854, 2215, 656, 1262, 476, 512,
1181, 4991, 3711, 5653, 7039, 5839, 4257, 3824, 3068)
veh <- data.frame(PC_G = PC_G)
veh1 <- my_age(x = net$ldv, y = PC_G, name = "PC")
veh2 <- temp_fact(net$ldv+net$hdv, pc_profile)
speed <- netspeed(veh2, net$ps, net$dfs, net$capacity, net$lkm, alpha = 1)
pckm <- units::set_units(fkm[[1]](1:24), "km")
pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[[1:11]])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[[12:24]])
#vehicles newer than pre-euro
co1 <- fe2015[fe2015$Pollutant=="CO", ] #24 obs!!!
cod <- c(co1$PC_G[1:24]<c(cod1,cod2),co1$PC_G[25:nrow(co1)])
lef <- ef_ldv_scaled(co1, cod, v = "PC", t = "4S", cc = "<=1400",
f = "G", p = "CO", eu=co1$Euro_LDV)
E_CO <- emis(veh = pckma, lkm = net$lkm, ef = lef, speed = speed, agemax = 41,
profile = pc_profile, simplify = TRUE)
class(E_CO)
E_CO_STREETS <- emis_post(arra = E_CO, pollutant = "CO", by = "streets", net = net)
g <- make_grid(net, 1/102.47/2, 1/102.47/2) #500m in degrees
emis_paved

Description

emis_paved estimates vehicular emissions from paved roads. The vehicular emissions are estimated as the product of the vehicles on a road, length of the road, emission factor from AP42 13.2.1 Paved roads. It is assumed dry hours and annual aggregation should consider moisture factor. It depends on Average Daily Traffic (ADT).

Usage

emis_paved(
  veh,
  adt,
  lkm,
  k = 0.62,
  sL1 = 0.6,
  sL2 = 0.2,
  sL3 = 0.06,
  sL4 = 0.03,
  W,
  net = net
)

Arguments

veh Numeric vector with length of elements equals to number of streets It is an array with dimensions number of streets x hours of day x days of week
adt Numeric vector of with Average Daily Traffic (ADT)
lkm Length of each link
k K_PM30 = 3.23 (g/vkm), K_PM15 = 0.77 (g/vkm), K_PM10 = 0.62 (g/vkm) and K_PM2.5 = 0.15 (g/vkm).
sL1 Silt loading (g/m2) for roads with ADT <= 500
sL2 Silt loading (g/m2) for roads with ADT > 500 and <= 5000
Silt loading (g/m²) for roads with ADT > 5000 and <= 1000

Silt loading (g/m²) for roads with ADT > 10000

$W$  
array of dimensions of veh. It consists in the hourly averaged weight of traffic fleet in each road

net
SpatialLinesDataFrame or Spatial Feature of "LINESTRING"

**Value**

emission estimation g/h

**Note**

silt values can vary a lot. For comparison:

<table>
<thead>
<tr>
<th>ADT</th>
<th>US-EPA g/m²</th>
<th>CENMA (Chile) g/m²</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 500</td>
<td>0.6</td>
<td>2.4</td>
</tr>
<tr>
<td>500-5000</td>
<td>0.2</td>
<td>0.7</td>
</tr>
<tr>
<td>5000-1000</td>
<td>0.06</td>
<td>0.6</td>
</tr>
<tr>
<td>&gt;10000</td>
<td>0.03</td>
<td>0.3</td>
</tr>
</tbody>
</table>

**References**


CENMA Chile: Actualizacion de inventario de emisiones de contaminantes atmosfericos RM 2020
Universidad de Chile

**Examples**

```r
## Not run:
# Do not run
veh <- matrix(1000, nrow = 10, ncol = 10)
W <- veh*1.5
lkm <- 1:10
ADT <- 1000:1010
emi <- emis_paved(veh = veh, adt = ADT, lkm = lkm, k = 0.65, W = W)
class(emi)
head(emi)
## End(Not run)
```

---

emis_post

Post emissions
Description

emis_post simplify emissions estimated as total per type category of vehicle or by street. It reads EmissionsArray and Emissions classes. It can return a dataframe with hourly emissions at each street, or a database with emissions by vehicular category, hour, including size, fuel and other characteristics.

Usage

emis_post(arra, veh, size, fuel, pollutant, by = "veh", net, type_emi, k = 1)

Arguments

arra  Array of emissions 4d: streets x category of vehicles x hours x days or 3d: streets x category of vehicles x hours
veh  Character, type of vehicle
size  Character, size or weight
fuel  Character, fuel
pollutant  Pollutant
by  Type of output, "veh" for total vehicular category , "streets_narrow" or "streets". "streets" returns a dataframe with rows as number of streets and columns the hours as days*hours considered, e.g. 168 columns as the hours of a whole week and "streets repeats the row number of streets by hour and day of the week
net  SpatialLinesDataFrame or Spatial Feature of "LINESTRING". Only when by = 'streets_wide'
type_emi  Character, type of emissions(exhaust, evaporative, etc)
k  Numeric, factor

Note

This function depends on EmissionsArray objects which currently has 4 dimensions. However, a future version of VEIN will produce EmissionsArray with 3 dimensiones and his function also will change. This change will be made in order to not produce inconsistencies with previous versions, therefore, if the user count with an EmissionsArray with 4 dimension, it will be able to use this function.

Examples

## Not run:
# Do not run
data(net)
data(pc_profile)
data(fe2015)
data(fkm)
PC_G <- c(33491,22340,24818,31808,46458,28574,24856,28972,37818,49050,87923,
  133833,138441,142682,171029,151048,115228,98664,126444,101027,
  84771,55864,36306,21079,20138,17439,7854,2215,656,1262,476,512,
  1181, 4991, 3711, 5653, 7039, 5839, 4257,3824, 3068)
pc1 <- my_age(x = net$ldv, y = PC_G, name = "PC")
# Estimation for morning rush hour and local emission factors
speed <- data.frame(S8 = net$ps)
p1h <- matrix(1)
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed, 
profile = p1h)
E_CO_STREETS <- emis_post(arra = E_CO, pollutant = "CO", by = "streets_wide")
summary(E_CO_STREETS)
E_CO_STREETssf <- emis_post(arra = E_CO, pollutant = "CO", 
by = "streets", net = net)
summary(E_CO_STREETssf)
plot(E_CO_STREETssf, main = "CO emissions (g/h)")

arguments required: arra, veh, size, fuel, pollutant ad by
E_CO_DF <- emis_post(arra = E_CO, veh = "PC", size = "<1400", fuel = "G",
pollutant = "CO", by = "veh")
# Estimation 168 hours
pc1 <- my_age(x = net$ldv, y = PC_G, name = "PC")
pcw <- temp_fact(net$ldv+net$hdv, pc_profile)
speed <- netspeed(pcw, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
pckm <- units::set_units(fkm[[1]][1:24],"km"); pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[12:24])

vehicles newer than pre-euro
c1o <- fe2015[fe2015$Pollutant == "CO", ] #24 obs!!!
cod <- c(co1PC_G[1:24],co1PC_G[25:nrow(co1)])
lef <- ef_ldv_scaled(dfcol = cod, v = "PC", cc = "<1400", 
f = "G", p = "CO", eu=co1$Euro_LDV)
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed, agemax = 41, 
profile = pc_profile)
# arguments required: arra, pollutant ad by
E_CO_STREETS <- emis_post(arra = E_CO, pollutant = "CO", by = "streets")
summary(E_CO_STREETS)
# arguments required: arra, veh, size, fuel, pollutant ad by
E_CO_DF <- emis_post(arra = E_CO, veh = "PC", size = "<1400", fuel = "G",
pollutant = "CO", by = "veh")
head(E_CO_DF)
# recreating 24 profile
lpc <- list(pc1*0.2, pc1*0.1, pc1*0.1, pc1*0.2, pc1*0.5, pc1*0.8, 
pc1, pc1*1.1, pc1,
pc1*0.8, pc1*0.5, pc1*0.5,
pc1*0.5, pc1*0.5, pc1*0.5, pc1*0.8,
pc1, pc1*1.1, pc1,
pc1*0.8, pc1*0.5, pc1*0.3, pc1*0.2, pc1*0.1)
E_COv2 <- emis(veh = lpc, lkm = net$lkm, ef = lef, speed = speed[, 1:24],
agemax = 41, hour = 24, day = 1)
plot(E_COv2)
E_CO_DFv2 <- emis_post(arra = E_COv2, 
veh = "PC", size = "<1400",
fuel = "G",
type_emi = "Exhaust",
pollutant = "CO", by = "veh")
emis_to_streets

Emis to streets distribute top-down emissions into streets

Description

emis_to_streets allocates emissions proportionally to each feature. "Spatial" objects are converter to "sf" objects. Currently, 'LINESTRING' or 'MULTILINESTRING' supported. The emissions are distributed in each street.

Usage

emis_to_streets(streets, dfemis, by = "ID", stpro, verbose = TRUE)

Arguments

streets       sf object with geometry 'LINESTRING' or 'MULTILINESTRING'. Or SpatialLinesDataFrame
dfemis        data.frame with emissions
by            Character indicating the columns that must be present in both 'street' and 'dfemis'
stpro         data.frame with two columns, category of streets and value. The name of the first column must be "stpro" and the sf streets must also have a column with the nam "stpro" indicating the category of streets. The second column must have the name "VAL" indicating the associated values to each category of street
verbose       Logical; to show more info.

Note

When spobj is a 'Spatial' object (class of sp), they are converted into 'sf'.

See Also

add_polid

Examples

## Not run:
data(net)
stpro = data.frame(stpro = as.character(unique(net$tstreet)),
                   VAL = 1:9)
dnet <- net["ldv"]
dnet$stpro <- as.character(net$tstreet)
dnet$ID <- "A"
df2 <- data.frame(BC = 10, CO = 20, ID = "A")
emis_wear <- emis_to_streets(streets = dnet, dfemis = df2)
sum(ste$lvd)
sum(net$lvd)
sum(ste$BC)
sum(df2$BC)
ste2 <- emis_to_streets(streets = dnet, dfemis = df2, stpro = stpro)
sum(ste2$lvd)
sum(net$lvd)
sum(ste2$BC)
sum(df2$BC)

## End(Not run)

emis_wear  

emis_wear estimates wear emissions. The sources are tyres, breaks and road surface.

Usage

emis_wear(
  veh,
  lkm,
  ef,
  what = "tyre",
  speed,
  agemax = ncol(veh),
  profile,
  hour = nrow(profile),
  day = ncol(profile)
)

Arguments

veh Object of class "Vehicles"
lkm Length of the road in km.
ef list of emission factor functions class "EmissionFactorsList", length equals to hours.
what Character for indicating "tyre", "break" or "road"
speed Speed data-frame with number of columns as hours
agemax Age of oldest vehicles for that category
profile Numerical or dataframe with nrows equal to 24 and ncol 7 day of the week
hour Number of considered hours in estimation
day Number of considered days in estimation
Value
emission estimation g/h

References

Examples
```r
## Not run:
data(net)
data(pc_profile)
pc_week <- temp_fact(net$ldv[1:10] + net$hdv[1:10], pc_profile[, 1])
df <- netspeed(pc_week, net$ps[1:10], net$ffs[1:10],
(net$capacity[1:10], net$lkm[1:10], alpha = 1)
ef <- ef_wear(wear = "tyre", type = "PC", pol = "PM10", speed = df)
emi <- emis_wear(veh = age_ldv(net$ldv[1:10], name = "VEH"),
(lkm = net$lkm[1:10], ef = ef, speed = df, profile = pc_profile[, 1])
emi
## End(Not run)
```

fe2015 Emission factors from Environmental Agency of Sao Paulo CETESB

Description
A dataset containing emission factors from CETESB and its equivalency with EURO

Usage
data(fe2015)

Format
A data frame with 288 rows and 12 variables:

- **Age**  Age of use
- **Year**  Year of emission factor
- **Pollutant**  Pollutants included: "CH4", "CO", "CO2", "HC", "N2O", "NMHC", "NOx", and "PM"
- **Proconve_LDV**  Proconve emission standard: "PP", "L1", "L2", "L3", "L4", "L5", "L6"
- **t_Euro_LDV**  Euro emission standard equivalence: "PRE_ECE", "I", "II", "III", "IV", "V"
- **Euro_LDV**  Euro emission standard equivalence: "PRE_ECE", "I", "II", "III", "IV", "V"
**Euro_HDV** Euro emission standard equivalence: "PRE", "I", "II", "III", "V"

**PC_G** CETESB emission standard for Passenger Cars with Gasoline (g/km)

**LT** CETESB emission standard for Light Trucks with Diesel (g/km)

**Source**

CETESB

---

**fkm**

*List of functions of mileage in km for Brazilian fleet*

**Description**

Functions from CETESB: Antonio de Castro Bruni and Marcelo Pereira Bales. 2013. Curvas de intensidade de uso por tipo de veículo automotor da frota da cidade de São Paulo. This functions depends on the age of use of the vehicle.

**Usage**

`data(fkm)`

**Format**

A data frame with 288 rows and 12 variables:

- **KM_PC_E25** Mileage in km of Passenger Cars using Gasoline with 25% Ethanol
- **KM_PC_E100** Mileage in km of Passenger Cars using Ethanol 100%
- **KM_PC_FLEX** Mileage in km of Passenger Cars using Flex engines
- **KM_LCV_E25** Mileage in km of Light Commercial Vehicles using Gasoline with 25% Ethanol
- **KM_LCV_FLEX** Mileage in km of Light Commercial Vehicles using Flex
- **KM_PC_B5** Mileage in km of Passenger Cars using Diesel with 5% biodiesel
- **KM_TRUCKS_B5** Mileage in km of Trucks using Diesel with 5% biodiesel
- **KM_BUS_B5** Mileage in km of Bus using Diesel with 5% biodiesel
- **KM_LCV_B5** Mileage in km of Light Commercial Vehicles using Diesel with 5% biodiesel
- **KM_SBUS_B5** Mileage in km of Small Bus using Diesel with 5% biodiesel
- **KM_ATRUCKS_B5** Mileage in km of Articulated Trucks using Diesel with 5% biodiesel
- **KM_MOTO_E25** Mileage in km of Motorcycles using Gasoline with 25% Ethanol
- **KM_LDV_GNV** Mileage in km of Light Duty Vehicles using Natural Gas

**Source**

CETESB
**Description**

Take into account the effect of better fuels on vehicles with older technology. If the ratio is less than 1, return 1. It means that it is not a degradation function.

**Usage**

```r
define_fuel_corr(
euro,
g = c(e100 = 52, aro = 39, o2 = 0.4, e150 = 86, olefin = 10, s = 165),
d = c(den = 840, pah = 9, cn = 51, t95 = 350, s = 400)
)
```

**Arguments**

- `euro`  
  Character; Euro standards ("PRE", "I", "II", "III", "IV", "V", VI, "VIc")
- `g`  
  Numeric; vector with parameters of gasoline with the names: e100(vol. (sulphur, ppm))
- `d`  
  Numeric; vector with parameters for diesel with the names: den (density at 15 Celsius degrees kg/m3), pah ( (Back end distillation in Celsius degrees) and s (sulphur, ppm)

**Value**

A list with the correction of emission factors.

**Note**

This function cannot be used to account for deterioration, therefore, it is restricted to values between 0 and 1. Parameters for gasoline (g):

- O2 = Oxygenates in
- S = Sulphur content in ppm
- ARO = Aromatics content in
- OLEFIN = Olefins content in
- E100 = Mid range volatility in
- E150 = Tail-end volatility in

Parameters for diesel (d):

- DEN = Density at 15 C (kg/m3)
- S = Sulphur content in ppm
- PAH = Aromatics content in
- CN = Cetane number
- T95 = Back-end distillation in o C.
get_project

Examples

## Not run:

f <- fuel_corr(euro = "I")
names(f)

## End(Not run)

get_project
Download vein project

Description

get_project downloads a project for running vein. The projects are available on Github.com/atmoschem/vein/projects

Usage

get_project(directory, case = "brazil_bu_chem", url)

Arguments

directory Character; Path to an existing or a new directory to be created.

case Character; One of the following:

case Description EF
brazil or brazil_bu or braslis or brasil_bu Bottom-up CETESB
emislacovid Bottom-up March 2020 CETESB
brazil_bu_csvgz Bottom-up. Faster but heavier CETESB+tunnel
brazil_td_chem Top-down with chemical mechanisms CETESB
brazil_bu_chem Bottom-up chemical mechanisms CETESB+tunnel
brazil_bu_chem_streets Bottom-up chemical mechanisms for streets and MUNICH CETESB+tunnel
brazil_bu_chem_streets_im Bottom-up chemical mechanisms for streets and MUNICH+IM CETESB+tunnel
brazil_bu_chem_month Bottom-up chemical mechanisms CETESB+tunnel
brazil_bu_chem_months_im Bottom-up chemical mechanisms for streets and MUNICH+IM CETESB+tunnel
sebr_cb05co2 Top-down SP, MG and RJ CETESB+tunnel
amazon2014 Top-down Amazon CETESB+tunnel
curitiba Bottom-down +GTFS CETESB+tunnel
masp2020 Bottom-down CETESB+tunnel
ecuador_td Top-down EEA
ecuador_td_im Top-down EEA
ecuador_td_hot Top-down EEA
ecuador_td_hot_month Top-down US/EPA MOVES EEA
moves_bu Bottom-up EEA
manizales_bu Bottom-up chemical mechanisms EEA
sebr_cb05co2_im Top-down SP, MG and RJ IM EEA
eu_bu_chem Bottom-up chemical mechanisms EEA 2019
GriddedEmissionsArray

url String, with the URL to download VEIN project

Note
default case can be any of "brasil", "brazil", "brazil_bu", "brasil_bu", they are the same In any case, if you find any error, please, send a pull request in github.

In Sao Paulo the IM programs was functioning until 2011.

Examples

```# Not run:
# do not run
get_project("awesomecity")

# End(Not run)`
```

GriddedEmissionsArray Construction function for class "GriddedEmissionsArray"

Description

GriddedEmissionsArray returns a transformed object with class "EmissionsArray" with 4 dimensions.

Usage

GriddedEmissionsArray(x, ..., cols, rows, times = ncol(x), rotate = "default")

```# S3 method for class 'GriddedEmissionsArray'
print(x, ...)

# S3 method for class 'GriddedEmissionsArray'
summary(object, ...)

# S3 method for class 'GriddedEmissionsArray'
plot(x, ..., times = 1)```
Arguments

- **x**: Object with class "SpatialPolygonDataFrame", "sf", "data.frame" or "matrix"
- **...**: ignored
- **cols**: Number of columns
- **rows**: Number of rows
- **times**: Number of times
- **rotate**: Character, rotate array: "default", "left", "right", "cols", "rows", "both", "br", "colsbr", "rowsbr", "bothbr". br means starting a matrix byrow
- **object**: object with class "EmissionsArray"

Value

Objects of class "GriddedEmissionsArray"

Examples

```r
## Not run:
data(net)
data(pc_profile)
data(fe2015)
data(fkm)
PC_G <- c(33491, 22340, 24818, 31808, 46458, 28574, 24856, 28972, 37818, 49050, 87923,
          133833, 138441, 142682, 171029, 151048, 115228, 98664, 126444, 101027,
          84771, 55864, 36306, 21079, 20138, 17439, 7854, 2215, 656, 1262, 476, 512,
          1181, 4991, 3711, 5653, 7039, 5839, 4257, 3824, 3068)
veh <- data.frame(PC_G = PC_G)
pc1 <- my_age(x = net$ldv, y = PC_G, name = "PC")
pcw <- temp_fact(net$ldv+net$hdv, pc_profile)
speed <- netspeed(pcw, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
pckma <- units::set_units(fkm[[1]](1:24), "km")
pckma <- cumsum(pckma)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[12:24])
# vehicles newer than pre-euro
c01 <- fe2015[fe2015$Pollutant=="CO", , drop=F] #24 obs!!!
cod <- c(co1$PC_G[1:24], c(cod1, cod2), co1$PC_G[25:nrow(co1)])
lef <- ef_ldv_scaled(co1, cod, v = "PC", t = "4S", cc = "<1400",
f = "G", p = "CO", eu=co1$Euro_LDV)
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed, agemax = 41,
            profile = pc_profile, simplify = TRUE)
class(E_CO)
E_CO_STREETS <- emis_post(arra = E_CO, pollutant = "CO", by = "streets",
                         net = net, k = units::set_units(1, "1/h"))
g <- make_grid(net, 1/102.47/2, 1/102.47/2) #500m in degrees
E_CO_g <- emis_grid(spobj = E_CO_STREETS, g = g, sr= 31983)
plot(E_CO_g["V9"])
# check all
rots <- c("default", "left", "right",
          "cols", "rows", "both",
```
grid_emis <- par()
par(mfrow = c(2,5))
lg <- lapply(seq_along(rots), function(i){
  x <- GriddedEmissionsArray(E_CO_g, 
    rows = 19,
    cols = 23,
    times = 168,
    rotate = rots[i])
  plot(x, main = rots[i])
})
par(mfrow = c(1,1))
## End(Not run)

grid_emis

Allocate emissions gridded emissions into streets (grid to emis street)

Description

grid_emis is sort of the opposite of emis_grid. It allocates gridded emissions into streets. This function applies emis_dist into each grid cell using lapply. This function is in development and pull request are welcome.

Usage

grid_emis(spobj, g, top_down = FALSE, sr, pro, char, verbose = FALSE)

Arguments

spobj
A spatial data frame of class "sp" or "sf". When class is "sp" it is transformed to "sf".

g
A grid with class "SpatialPolygonsDataFrame" or "sf". This grid includes the total emissions with the column "emission". If the profile is going to be used, the column 'emission' must include the sum of the emissions for each profile. For instance, if profile covers the hourly emissions, the column 'emission' bust be the sum of the hourly emissions.

top_down
Logical; requires emissions named 'emissions' and allows to apply profile factors. If your data is hourly emissions or a spatial grid with several emissions at different hours, being each hour a column, it is better to use top_down = FALSE. In this way all the hourly emissions are considered, however, each hourly emissions has to have the name "V" and the number of the hour like "V1"

sr
Spatial reference e.g: 31983. It is required if spobj and g are not projected. Please, see http://spatialreference.org/.

pro
Numeric, Matrix or data-frame profiles, for instance, pc_profile.
grid_emis

char  Character, name of the first letter of hourly emissions. New variables in R start with the letter “V”, for your hourly emissions might start with the letter “h”. This option applies when top_down is FALSE. For instance, if your hourly emissions are: “h1”, “h2”, “h3”... ‘char’ can be “h”

verbose  Logical; to show more info.

Note

Your gridded emissions might have flux units (mass / area / time(implicit)) You must multiply your emissions with the area to return to the original units.

Examples

```r
## Not run:
data(net)
data(pc_profile)
data(fkm)
PC_G <- c(33491, 22340, 24818, 31808, 46458, 28574, 24856, 28972, 37818, 49050, 87923,
133833, 138441, 142682, 171029, 151048, 115228, 98664, 126444, 101027,
84771, 55864, 36306, 21079, 20138, 17439, 7854, 2215, 656, 1262, 476, 512,
1181, 4991, 3711, 5653, 7039, 5839, 4257, 3824, 3068)
pc1 <- my_age(x = net$ldv, y = PC_G, name = "PC")
# Estimation for morning rush hour and local emission factors
lef <- EmissionFactorsList(ef_cetesb("CO", "PC_G"))
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef,
              profile = 1, speed = Speed(1))
E_CO_STREETS <- emis_post(arra = E_CO, by = "streets", net = net)

g <- make_grid(net, 1/102.47/2) #500m in degrees

gCO <- emis_grid(spobj = E_CO_STREETS, g = g)
gCO$emission <- gCO$V1
area <- sf::st_area(gCO)
area <- units::set_units(area, "km^2") #Check units!
gCO$emission <- gCO$emission*area

#\dontrun{
#do not run
library(osmdata)
library(sf)
osm <- osmdata_sf(
  add_osm_feature(
    opq(bbox = st_bbox(gCO)),
    key = 'highway')$osm_lines[, c("highway")]
  st <- c("motorway", "motorway_link", "trunk", "trunk_link",
          "primary", "primary_link", "secondary", "secondary_link",
          "tertiary", "tertiary_link")
osm <- osm[osm$highway %in% st, ]
}
plot(osm, axes = T)
# top_down requires name 'emissions' into gCO
xnet <- grid_emis(osm, gCO, top_down = TRUE)
plot(xnet, axes = T)
```
## invcop

Helper function to copy and zip projects

### Description

invcop help to copy and zip projects

### Usage

invcop(in_name = getwd(),
        out_name,
        all = FALSE,
        main = TRUE,
        ef = TRUE,
        est = TRUE,
        network = TRUE,
        veh_rds = FALSE,
        veh_csv = TRUE,
        zip = TRUE)

### Arguments

- **in_name**: Character; Name of current project.
- **out_name**: Character; Name of output project.
- **all**: Logical; copy ALL (and for once) or not.
- **main**: Logical; copy or not.
- **ef**: Logical; copy or not.
- **est**: Logical; copy or not.
- **network**: Logical; copy or not.
- **veh_rds**: Logical; copy or not.
- **veh_csv**: Logical; copy or not.
- **zip**: Logical; zip or not.

### Value

emission estimation g/h
inventory

Note

This function was created to copy and zip project without the emis.

Examples

```r
## Not run:
# Do not run
## End(Not run)
```

```r
inventory
inventory function.
```

Description

`inventory` produces a structure of directories and scripts in order to run vein. It is required to know the vehicular composition of the fleet.

Usage

```r
inventory(
  name,
  vehcomp = c(PC = 1, LCV = 1, HGV = 1, BUS = 1, MC = 1),
  show.main = FALSE,
  scripts = TRUE,
  show.dir = FALSE,
  show.scripts = FALSE,
  clear = TRUE,
  rush.hour = FALSE,
  showWarnings = FALSE
)
```

Arguments

- `name`: Character, path to new main directory for running vein. NO BLANK SPACES
- `vehcomp`: Vehicular composition of the fleet. It is required a named numerical vector with the names "PC", "LCV", "HGV", "BUS" and "MC". In the case that there are no vehicles for one category of the composition, the name should be included with the number zero, for example, PC = 0. The maximum number allowed is 99 per category.
- `show.main`: Logical; Do you want to see the new main.R file?
- `scripts`: Logical: Do you want to generate or no R scripts?
- `show.dir`: Logical value for printing the created directories.
- `show.scripts`: Logical value for printing the created scripts.
- `clear`: Logical value for removing recursively the directory and create another one.
- `rush.hour`: Logical, to create a template for morning rush hour.
- `showWarnings`: Logical, showWarnings?
Value

Structure of directories and scripts for automating the compilation of vehicular emissions inventory. The structure can be used with another type of sources of emissions. The structure of the directories is: daily, ef, emi, est, images, network and veh. This structure is a suggestion and the user can use another. 'ef: it is for storing the emission factors data-frame, similar to data(fe2015) but including one column for each of the categories of the vehicular composition. For instance, if PC = 5, there should be 5 columns with emission factors in this file. If LCV = 5, another 5 columns should be present, and so on.

emi: Directory for saving the estimates. It is suggested to use .rds extension instead of .rda.

est: Directory with subdirectories matching the vehicular composition for storing the scripts named input.R.

images: Directory for saving images.

network: Directory for saving the road network with the required attributes. This file will include the vehicular flow per street to be used by age* functions.

veh: Directory for storing the distribution by age of use of each category of the vehicular composition. Those are data-frames with number of columns with the age distribution and number of rows as the number of streets. The class of these objects is "Vehicles". Future versions of vein will generate Vehicles objects with the explicit spatial component.

The name of the scripts and directories are based on the vehicular composition, however, there is included a file named main.R which is just an R script to estimate all the emissions. It is important to note that the user must add the emission factors for other pollutants. Also, this function creates the scripts input.R where the user must specify the inputs for the estimation of emissions of each category. Also, there is a file called traffic.R to generate objects of class "Vehicles". The user can rename these scripts.

Examples

```r
## Not run:
name = file.path(tempdir(), "YourCity")
inventory(name = name)

## End(Not run)
```

long_to_wide

Transform data.frame from long to wide format

Description

`long_to_wide` transform data.frame from long to wide format
Usage

long_to_wide(
  df,
  column_with_new_names = names(df)[1],
  column_with_data = "emission",
  column_fixed,
  net
)

Arguments

   df         data.frame with three column.
  column_with_new_names    Character, column that has new column names
 column_with_data         Character column with data
 column_fixed            Character, column that will remain fixed
    net                   To return a sf

Value

   wide data.frame.

See Also

emis_hot_td emis_cold_td wide_to_long

Examples

## Not run:
  df <- data.frame(pollutant = rep(c("CO", "propadiene", "NO2"), 10),
                   emission = vein::Emissions(1:30),
                   region = rep(letters[1:2], 15))
  df
  long_to_wide(df)
  long_to_wide(df, column_fixed = "region")
  ## End(Not run)

---

make_grid

Creates rectangular grid for emission allocation

Description

make_grid creates a sf grid of polygons. The spatial reference is taken from the spatial object.
Usage

make_grid(spobj, width, height = width, polygon, crs = 3857)

Arguments

spobj A spatial object of class sp or sf.
width Width of grid cell. It is recommended to use projected values.
height Height of grid cell.
polygon Deprecated! make_grid returns only sf grid of polygons.
crs coordinate reference system in numeric format from http://spatialreference.org/ to transform/project spatial data using sf::st_transform. The default value is 3857, Pseudo Mercator

Value

A grid of polygons class 'sf'

Examples

## Not run:
data(net)
grid <- make_grid(net, width = 0.5/102.47) #500 mts
plot(grid, axes = TRUE) #class sf
# make grid now returns warnings for crs with form +init...
#grid <- make_grid(net, width = 0.5/102.47) #500 mts

## End(Not run)

---

moves_ef

MOVES emission factors

Description

moves_ef reads and filter MOVES data.frame of emission factors.

Usage

moves_ef(
  ef,
  vehicles,
  source_type_id = 21,
  process_id = 1,
  fuel_type_id = 1,
  pollutant_id = 2,
  road_type_id = 5,
  speed_bin
)
Arguments

- `ef`: EmissionFactors data.frame
- `veh`: MOVES estimation of using rates per distance
- `lkm`: MOVES estimation of using rates per distance
- `source_type_id`: Number to identify type of vehicle as defined by MOVES.
- `process_id`: Number to identify emission process defined by MOVES.
- `fuel_type_id`: Number to identify type of fuel as defined by MOVES.
- `pollutant_id`: Number to identify type of pollutant as defined by MOVES.
- `road_type_id`: Number to identify type of road as defined by MOVES.
- `speed_bin`: Data.frame or vector of avgSpeedBinID as defined by MOVES.

Value

EmissionFactors data.frame

Note

‘decoder’ shows a decoder for MOVES to identify

Examples

```
{ 
  data(decoder) 
  decoder 
}
```

Description

`moves_rpd` estimates running exhaust emissions using MOVES emission factors.

Usage

```
moves_rpd( 
  veh, 
  lkm, 
  ef, 
  fuel_type, 
  speed_bin, 
  profile, 
  source_type_id = 21, 
  fuel_type_id = 1, 
  pollutant_id = 91, 
  road_type_id = 5, 
)```
process_id = 1,  
vehicle = NULL,  
vehicle_type = NULL,  
fuel_subtype = NULL,  
net,  
path_all,  
verbose = FALSE  
)

Arguments
veh  "Vehicles" data-frame or list of "Vehicles" data-frame. Each data-frame as number of columns matching the age distribution of that type of vehicle. The number of rows is equal to the number of streets link.

lk reimb  Length of each link in miles

ef  emission factors from EmissionRates_running exported from MOVES

fuel_type  Data.frame of fuelSubtypeID exported by MOVES.

speed_bin  Data.frame or vector of avgSpeedBinID as defined by MOVES.

profile  Data.frame or Matrix with n rows equal to 24 and ncol 7 day of the week

source_type_id  Number to identify type of vehicle as defined by MOVES.

fuel_type_id  Number to identify type of fuel as defined by MOVES.

pollutant_id  Number to identify type of pollutant as defined by MOVES.

road_type_id  Number to identify type of road as defined by MOVES.

process_id  Number to identify type of pollutant as defined by MOVES.

vehicle  Character, type of vehicle

vehicle_type  Character, subtype of vehicle

fuel_subtype  Character, subtype of vehicle

net  Road network class sf

path_all  Character to export whole estimation. It is not recommended since it is usually too heavy.

verbose  Logical; To show more information. Not implemented yet

Value

a list with emissions at each street and data.base aggregated by categories. See link{emis_post}

Note
‘decoder’ shows a decoder for MOVES

Examples
{
data(decoder)
decoder
}
Description

`moves_rpdy` estimates running exhaust emissions using MOVES emission factors.

Usage

```r
moves_rpdy(
  veh,
  lkm,
  ef,
  source_type_id = 21,
  fuel_type_id = 1,
  pollutant_id = 91,
  road_type_id = 5,
  fuel_type,
  speed_bin,
  profile,
  vehicle,
  vehicle_type,
  fuel_subtype,
  process_id,
  net,
  path_all,
  verbose = FALSE
)
```

Arguments

- **veh**: "Vehicles" data-frame or list of "Vehicles" data-frame. Each data-frame as number of columns matching the age distribution of that type of vehicle. The number of rows is equal to the number of streets link.
- **lkm**: Length of each link in miles
- **ef**: emission factors from EmissionRates_running exported from MOVES
- **source_type_id**: Number to identify type of vehicle as defined by MOVES.
- **fuel_type_id**: Number to identify type of fuel as defined by MOVES.
- **pollutant_id**: Number to identify type of pollutant as defined by MOVES.
- **road_type_id**: Number to identify type of road as defined by MOVES.
- **fuel_type**: Data.frame of fuelSubtypeID exported by MOVES.
- **speed_bin**: Data.frame or vector of avgSpeedBinID as defined by MOVES.
- **profile**: Data.frame or Matrix with nrows equal to 24 and ncol 7 day of the week
- **vehicle**: Character, type of vehicle
moves_rpdy_meta

vehicle_type  Character, subtype of vehicle
fuel_subtype   Character, subtype of vehicle
process_id     Character, processID
net            Road network class sf
path_all       Character to export whole estimation. It is not recommended since it is usually too heavy.
verbose        Logical; To show more information. Not implemented yet

Value

a list with emissions at each street and data.base aggregated by categories. See link{emis_post}

Note

‘decoder’ shows a decoder for MOVES

Examples

```
{ data(decoder)
  decoder
}
```

Description

`moves_rpdy_meta` estimates running exhaust emissions using MOVES emission factors.

Usage

```
moves_rpdy_meta(
  metadata,
  lkm,
  ef,
  fuel_type,
  speed_bin,
  profile,
  agemax = 31,
  net,
  simplify = TRUE,
  verbose = FALSE
)
```
moves_rpdy_sf

Arguments

- **metadata**: data.frame with the metadata for a vein project for MOVES.
- **lkm**: Length of each link in miles
- **ef**: emission factors from EmissionRates_running exported from MOVES
- **fuel_type**: Data.frame of fuelSubtypeID exported by MOVES.
- **speed_bin**: Data.frame or vector of avgSpeedBinID as defined by MOVES.
- **profile**: Data.frame or Matrix with nrows equal to 24 and ncol 7 day of the week
- **agemax**: Integer; max age for the fleet, assuming the same for all vehicles.
- **net**: Road network class sf
- **simplify**: Logical, to return the whole object or processed by streets and veh
- **verbose**: Logical; To show more information. Not implemented yet

Value

a list with emissions at each street and data.base aggregated by categories.

Note

The idea is the user enter with emissions factors by pollutant

Examples

```
{
  data(decoder)
  decoder
}
```

Description

**moves_rpdy_sf** estimates running exhaust emissions using MOVES emission factors.

Usage

```
moves_rpdy_sf(
  veh,
  lkm,
  ef,
  speed_bin,
  profile,
  source_type_id = 21,
  vehicle = NULL,
  vehicle_type = NULL,
```
fuel_subtype = NULL,
path_all,
verbose = FALSE
)

Arguments

veh "Vehicles" data-frame or list of "Vehicles" data-frame. Each data-frame as number of columns matching the age distribution of that type of vehicle. The number of rows is equal to the number of streets link.

lkm Length of each link in miles

ef emission factors from EmissionRates_running exported from MOVES filtered by sourceTypeID and fuelTypeID.

speed_bin Data.frame or vector of avgSpeedBinID as defined by MOVES.

profile numeric vector of normalized traffic for the morning rush hour

source_type_id Number to identify type of vehicle as defined by MOVES.

vehicle Character, type of vehicle

vehicle_type Character, subtype of vehicle

fuel_subtype Character, subtype of vehicle

path_all Character to export whole estimation. It is not recommended since it is usually too heavy.

verbose Logical; To show more information. Not implemented yet

Value

a list with emissions at each street and data.base aggregated by categories. See link{emis_post}

Note

‘decoder’ shows a decoder for MOVES

Examples

{
data(decoder)
decoder
}

moves_rpsy_sf
**moves_rpsy_meta**

**MOVES estimation of using rates per start by model year**

**Description**

`moves_rpsy_meta` estimates running exhaust emissions using MOVES emission factors.

**Usage**

```r
moves_rpsy_meta(
    metadata,
    lkm,
    ef,
    fuel_type,
    profile,
    agemax = 31,
    net,
    simplify = TRUE,
    verbose = FALSE,
    colk,
    colkt = F
)
```

**Arguments**

- `metadata`: data.frame with the metadata for a vein project for MOVES.
- `lkm`: Length of each link in miles
- `ef`: emission factors from `EmissionRates_running` exported from MOVES
- `fuel_type`: Data.frame of fuelSubtypeID exported by MOVES
- `profile`: Data.frame or Matrix with nrows equal to 24 and ncol 7 day of the week
- `agemax`: Integer; max age for the fleet, assuming the same for all vehicles.
- `net`: Road network class sf
- `simplify`: Logical, to return the whole object or processed by streets and veh
- `verbose`: Logical; To show more information. Not implemented yet
- `colk`: Character identifying a column in `metadata` to multiply the emission factor
- `colkt`: Logical, TRUE if `colk` is used

**Value**

a list with emissions at each street and data.base aggregated by categories.

**Note**

The idea is the user enter with emissions factors by pollutant
moves_rpsy_sf

**Examples**

```r
{
data(decoder)
decoder
}
```

**Description**

*move_rpsy_sf* estimates running exhaust emissions using MOVES emission factors.

**Usage**

```r
moves_rpsy_sf(
  veh,
  lkm,
  ef,
  profile,
  source_type_id = 21,
  vehicle = NULL,
  vehicle_type = NULL,
  fuel_subtype = NULL,
  net,
  path_all,
  verbose = FALSE
)
```

**Arguments**

- **veh** "Vehicles" data-frame or list of "Vehicles" data-frame. Each data-frame as number of columns matching the age distribution of that type of vehicle. The number of rows is equal to the number of streets link.
- **lkm** Length of each link in miles
- **ef** emission factors from EmissionRates_running exported from MOVES filtered by sourceTypeID and fuelTypeID.
- **profile** numeric vector of normalized traffic for the morning rush hour
- **source_type_id** Number to identify type of vehicle as defined by MOVES.
- **vehicle** Character, type of vehicle
- **vehicle_type** Character, subtype of vehicle
- **fuel_subtype** Character, subtype of vehicle
- **net** Road network class sf
- **path_all** Character to export whole estimation. It is not recommended since it is usually too heavy.
- **verbose** Logical; To show more information. Not implemented yet
Value

A list with emissions at each street and database aggregated by categories. See link{emis_post}

Note

‘decoder’ shows a decoder for MOVES

Examples

```{r}
{ 
data(decoder)
decoder
}
```

moves_speed

Return speed bins according to US/EPA MOVES model

Description

`speed_moves` return an object of average speed bins as defined by US EPA MOVES. The input must be speed as miles/h (mph)

Usage

`moves_speed(x, net)`

Arguments

- **x**: Object with class, "sf", "data.frame", "matrix" or "numeric" with speeds in miles/h (mph)
- **net**: optional spatial dataframe of class "sf". It is transformed to "sf".

Examples

```{r}
{ 
data(net)
net$mph <- units::set_units(net$ps, "miles/h")
net$speed_bins <- moves_speed(net$mph)
head(net)
moves_speed(net["ps"])
}
```
Description

`my_age` returns amount of vehicles at each age using a numeric vector.

Usage

```r
my_age(
  x,
  y,
  agemax,
  name = "vehicle",
  k = 1,
  pro_street,
  net,
  verbose = FALSE,
  namerows
)
```

Arguments

- **x**: Numeric; vehicles by street (or spatial feature).
- **y**: Numeric or data.frame; when `pro_street` is not available, `y` must be 'numeric', else, a 'data.frame'. The names of the columns of this data.frame must be the same as the elements of `pro_street` and each column must have a profile of age of use of vehicle. When 'y' is 'numeric' the vehicles has the same age distribution to all streets. When 'y' is a data.frame, the distribution by age of use varies the streets.
- **agemax**: Integer; age of oldest vehicles for that category
- **name**: Character; of vehicle assigned to columns of dataframe.
- **k**: Integer; multiplication factor. If its length is > 1, it must match the length of `x`
- **pro_street**: Character; each category of profile for each street. The length of this character vector must be equal to the length of 'x'. The names of the data.frame 'y' must have the same content of 'pro_street'
- **net**: SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
- **verbose**: Logical; message with average age and total number of vehicles.
- **namerows**: Any vector to be change row.names. For instance, the name of regions or streets.

Value

dataframe of age distribution of vehicles.
The functions age* produce distribution of the circulating fleet by age of use. The order of using these functions is:

1. If you know the distribution of the vehicles by age of use, use: `my_age`
2. If you know the sales of vehicles, or (the regis)*better) the registry of new vehicles, use `age` to apply a survival function.
3. If you know the theoretical shape of the circulating fleet and you can use `age_ldv`, `age_hdv` or `age_moto`. For instance, you dont know the sales or registry of vehicles, but somehow you know the shape of this curve. 4. You can use/merge/transform/adapt any of these functions.

Examples

```r
## Not run:
data(net)
dpc <- c(seq(1,20,3), 20:10)
PC_E25_1400 <- my_age(x = net$ldv, y = dpc, name = "PC_E25_1400")
class(PC_E25_1400)
plot(PC_E25_1400)
PC_E25_1400sf <- my_age(x = net$ldv, y = dpc, name = "PC_E25_1400", net = net)
class(PC_E25_1400sf)
plot(PC_E25_1400sf)
PC_E25_1400nsf <- sf::st_set_geometry(PC_E25_1400sf, NULL)
class(PC_E25_1400nsf)

yy <- data.frame(a = 1:5, b = 5:1) # perfiles por categoria de calle
pro_street <- c("a", "b", "a") # categorias de cada calle
x <- c(100,5000, 3) # vehiculos
my_age(x = x, y = yy, pro_street = pro_street)
## End(Not run)
```

---

### Description

This dataset is an sf class object with roads from a traffic simulation made by CET Sao Paulo, Brazil

#### Usage

```r
data(net)
```

#### Format

A Spatial.data.frame (sf) with 1796 rows and 1 variables:

- **ldv** Light Duty Vehicles (veh/h)
- **hdv** Heavy Duty Vehicles (veh/h)
- **lkm** Length of the link (km)
netspeed

Description

netspeed Creates a dataframe of speeds for different hours and each link based on morning rush traffic data

Usage

netspeed(
  q = 1,
  ps,
  ffs,
  cap,
  lkm,
  alpha = 0.15,
  beta = 4,
  net,
  scheme = FALSE,
  dist = "km"
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>Data-frame of traffic flow to each hour (veh/h)</td>
</tr>
<tr>
<td>ps</td>
<td>Peak speed (km/h)</td>
</tr>
<tr>
<td>ffs</td>
<td>Free flow speed (km/h)</td>
</tr>
<tr>
<td>cap</td>
<td>Capacity of link (veh/h)</td>
</tr>
<tr>
<td>lkm</td>
<td>Distance of link (km)</td>
</tr>
<tr>
<td>alpha</td>
<td>Parameter of BPR curves</td>
</tr>
<tr>
<td>beta</td>
<td>Parameter of BPR curves</td>
</tr>
<tr>
<td>net</td>
<td>SpatialLinesDataFrame or Spatial Feature of &quot;LINESTRING&quot;</td>
</tr>
<tr>
<td>scheme</td>
<td>Logical to create a Speed data-frame with 24 hours and a default profile. It needs ffs and ps:</td>
</tr>
<tr>
<td>dist</td>
<td>String indicating the units of the resulting distance in speed. Default is units from peak speed ‘ps’</td>
</tr>
</tbody>
</table>
Profile of Vehicle start patterns

Description

This dataset is a dataframe with percentage of hourly starts with a lapse of 6 hours with engine turned off. Data source is: Lents J., Davis N., Nikkila N., Osses M. 2004. Sao Paulo vehicle activity study. ISSRC. www.issrc.org

Usage

data(pc_cold)
Format

A data frame with 24 rows and 1 variables:

V1 24 hours profile vehicle starts for Monday

Description

This dataset is a dataframe with traffic activity normalized monday 08:00-09:00. This data is normalized at 08:00-09:00. It comes from data of toll stations near Sao Paulo City. The source is ARTESP (www.artesp.com.br)

Usage

data(pc_profile)

Format

A data frame with 24 rows and 7 variables:

V1 24 hours profile for Monday
V2 24 hours profile for Tuesday
V3 24 hours profile for Wednesday
V4 24 hours profile for Thursday
V5 24 hours profile for Friday
V6 24 hours profile for Saturday
V7 24 hours profile for Sunday

pollutants

Description

This dataset also includes MIR, MOIR and EBIR is Carter SAPRC07.xls https://www.engr.ucr.edu/~carter/SAPRC/

Usage

data(pollutants)
profiles

Format

A data frame with 148 rows and 10 variables:

- **n** Number for each pollutant, from 1 to 132

**group1** classification for pollutants including "NMHC", "PAH", "METALS", "PM", "criteria" and "PCDD"

**group2** A sub classification for pollutants including "alkenes", "alkynes", "aromatics", "alkanes", "PAH", "aldehydes", "ketones", "METALS", "PM_char", "criteria", "cycloalkanes", "NMHC", "PCDD", "PM10", "PM2.5"

**pollutant** 1 of the 132 pollutants covered

**CAS** CAS Registry Number

**g_mol** molar mass

**MIR** Maximum incremental Reactivity (gm O3 / gm VOC)

**MOIR** Reactivity (gm O3 / gm VOC)

**EBIR** Reactivity (gm O3 / gm VOC)

**notes** Inform some assumption for molar mass

profiles

Profile of traffic data 24 hours 7 n days of the week

Description

This dataset is a list of data-frames with traffic activity normalized monday 08:00-09:00. It comes from data of toll stations near Sao Paulo City. The source is ARTESP (www.artesp.com.br) for months January and June and years 2012, 2013 and 2014. The type of vehicles covered are PC, LCV, MC and HGV.

Usage

data(pc_profile)

Format

A list of data-frames with 24 rows and 7 variables:

- **PC_JUNE_2012** 168 hours
- **PC_JUNE_2013** 168 hours
- **PC_JUNE_2014** 168 hours
- **LCV_JUNE_2012** 168 hours
- **LCV_JUNE_2013** 168 hours
- **LCV_JUNE_2014** 168 hours
- **MC_JUNE_2012** 168 hours
remove_units

Description

remove_units Remove units from sf, data.frames, matrix or units.

Usage

remove_units(x)

Arguments

x Object with class "sf", "data.frame", "matrix" or "units"

Value

"sf", data.frame", "matrix" or numeric

Examples

## Not run:
ef1 <- ef_cetesb(p = "CO", c("PC_G", "PC_FE"))
class(ef1)
sapply(ef1, class)
a <- remove_units(ef1)

## End(Not run)
speciate

Speciation of emissions

Description

speciate separates emissions in different compounds. It covers black carbon and organic matter from particulate matter. Soon it will be added more speciations

Usage

speciate(
  x = 1,
  spec = "bcom",
  veh,
  fuel,
  eu,
  list = FALSE,
  pmpar,
  verbose = FALSE
)

Arguments

x  Emissions estimation
spec  The speciations are:
  • "bcom": Splits PM2.5 in black carbon and organic matter.
  • "tyre" or "tire": Splits PM in PM10, PM2.5, PM1 and PM0.1.
  • "brake": Splits PM in PM10, PM2.5, PM1 and PM0.1.
  • "road": Splits PM in PM10 and PM2.5.
  • "nox": Splits NOx in NO and NO2.
  • "nmhc": Splits NMHC in compounds, see ef_ldv_speed.
  • "pmiag", "pmneu", "pmneu2": Splits PM in groups, see note below.
veh  Type of vehicle:
  • "bcom": veh can be "PC", "LCV", HDV" or "Motorcycle".
  • "tyre" or "tire": not necessary.
  • "brake": not necessary.
  • "road": not necessary.
  • "nox": veh can be "PC", "LCV", HDV" or "Motorcycle".
  • "nmhc" see below
  • ""pmiag", "pmneu", "pmneu2": not necessary.
fuel  Fuel.
  • "bcom": "G" or "D".
  • "tyre" or "tire": not necessary.
• "brake": not necessary.
• "road": not necessary.
• "nox": "G", "D", "LPG", "E85" or "CNG".
• "nmhc": see below
• "pmiag", "pmneu", "pmneu2": not necessary.

Emission standard
• "bcom": "G" or "D".
• "tyre" or "tire": not necessary.
• "brake": not necessary.
• "road": not necessary.
• "nox": "G", "D", "LPG", "E85" or "CNG".
• "nmhc": see below
• "pmiag", "pmneu", "pmneu2": not necessary.

when TRUE returns a list with number of elements of the list as the number of species of pollutants

Numeric vector for PM speciation eg: c(e_so4i = 0.0077, e_so4j = 0.0623, e_no3i = 0.00247, e_no3j = 0.01053, e_pm25i = 0.1, e_pm25j = 0.3, e_orgi = 0.0304, e_orgj = 0.1296, e_eci = 0.056, e_ecj = 0.024, h2o = 0.277) These are default values. however, when this argument is present, new values are used.

Logical to show more information

dataframe of speciation in grams or mols

options for spec "nmhc":

<table>
<thead>
<tr>
<th>veh</th>
<th>fuel</th>
<th>eu</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDV</td>
<td>G</td>
<td>PRE</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>I</td>
</tr>
<tr>
<td>LDV</td>
<td>D</td>
<td>all</td>
</tr>
<tr>
<td>HDV</td>
<td>D</td>
<td>all</td>
</tr>
<tr>
<td>LDV</td>
<td>LPG</td>
<td>all</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>Evaporative</td>
</tr>
<tr>
<td>LDV</td>
<td>E25</td>
<td>Evaporative</td>
</tr>
<tr>
<td>LDV</td>
<td>E100</td>
<td>Evaporative</td>
</tr>
<tr>
<td>LDV</td>
<td>E25</td>
<td>Exhaust</td>
</tr>
<tr>
<td>LDV</td>
<td>E100</td>
<td>Exhaust</td>
</tr>
<tr>
<td>HDV</td>
<td>B5</td>
<td>Exhaust</td>
</tr>
<tr>
<td>LDV</td>
<td>E85</td>
<td>Exhaust</td>
</tr>
<tr>
<td>LDV</td>
<td>E85</td>
<td>Evaporative</td>
</tr>
<tr>
<td>LDV</td>
<td>CNG</td>
<td>Exhaust</td>
</tr>
<tr>
<td>ALL</td>
<td>E100</td>
<td>Liquid</td>
</tr>
<tr>
<td>ALL</td>
<td>G</td>
<td>Liquid</td>
</tr>
<tr>
<td></td>
<td>E25</td>
<td>Liquid</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>ALL</td>
<td>E25</td>
<td>OM</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>OM-001</td>
</tr>
<tr>
<td>LDV</td>
<td>D</td>
<td>OM-002</td>
</tr>
<tr>
<td>HDV</td>
<td>D</td>
<td>OM-003</td>
</tr>
<tr>
<td>MC</td>
<td>G</td>
<td>OM-004</td>
</tr>
<tr>
<td>ALL_LPG</td>
<td>G</td>
<td>OM-005</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>OM-001-001</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>OM-001-002</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>OM-001-003</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>OM-001-004</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>OM-001-005</td>
</tr>
<tr>
<td>LDV</td>
<td>G</td>
<td>OM-001-006</td>
</tr>
<tr>
<td>LDV</td>
<td>D</td>
<td>OM-002-001</td>
</tr>
<tr>
<td>LDV</td>
<td>D</td>
<td>OM-002-002</td>
</tr>
<tr>
<td>LDV</td>
<td>D</td>
<td>OM-002-003</td>
</tr>
<tr>
<td>LDV</td>
<td>D</td>
<td>OM-002-004</td>
</tr>
<tr>
<td>LDV</td>
<td>D</td>
<td>OM-002-005</td>
</tr>
<tr>
<td>LDV</td>
<td>D</td>
<td>OM-002-006</td>
</tr>
<tr>
<td>HDV</td>
<td>D</td>
<td>OM-003-001</td>
</tr>
<tr>
<td>HDV</td>
<td>D</td>
<td>OM-003-002</td>
</tr>
<tr>
<td>HDV</td>
<td>D</td>
<td>OM-003-003</td>
</tr>
<tr>
<td>HDV</td>
<td>D</td>
<td>OM-003-004</td>
</tr>
<tr>
<td>HDV</td>
<td>D</td>
<td>OM-003-005</td>
</tr>
<tr>
<td>HDV</td>
<td>D</td>
<td>OM-003-006</td>
</tr>
<tr>
<td>MC</td>
<td>G</td>
<td>OM-004-001</td>
</tr>
<tr>
<td>MC</td>
<td>G</td>
<td>OM-004-002</td>
</tr>
<tr>
<td>MC</td>
<td>G</td>
<td>OM-004-003</td>
</tr>
<tr>
<td>ALL_ALL</td>
<td>urban</td>
<td></td>
</tr>
<tr>
<td>ALL_ALL</td>
<td>highway</td>
<td></td>
</tr>
</tbody>
</table>

After eu = OM, all profiles are Chinese # the following specs will be removed soon

- "iag_racm": ethanol emissions added in hc3.
- "iag" or "iag_cb05": Splits NMHC by CB05 (WRF exb05_opt1) group.
- "petroiag_cb05": Splits NMHC by CB05 (WRF exb05_opt1) group.
- "iag_cb05v2": Splits NMHC by CB05 (WRF exb05_opt2) group.
- "neu_cb05": Splits NMHC by CB05 (WRF exb05_opt2) group alternative.
- "petroiag_cb05v2": Splits NMHC by CB05 (WRF exb05_opt2) group alternative.


Specs: "neu_cb05", "pmneu" and "pmneu2" provided by Daniel Schuch, from Northeastern University. "pm2023" provided by Iara da Silva; Leila D. Martins
Speciation with fuels "E25", "E100" and "B5" made by Prof. Leila Martins (UTFPR), represents BRAZILIAN fuel

pmag2 pass the mass only on j fraction

References


Examples

```r
## Not run:
# Do not run
pm <- rnorm(n = 100, mean = 400, sd = 2)
(df <- speciate(pm, veh = "PC", fuel = "G", eu = "I"))
(df <- speciate(pm, spec = "brake", veh = "PC", fuel = "G", eu = "I"))
(dfa <- speciate(pm, spec = "iag", veh = "veh", fuel = "G", eu = "Exhaust"))
(dfb <- speciate(pm, spec = "iag_cb05v2", veh = "veh", fuel = "G", eu = "Exhaust"))
(dfb <- speciate(pm, spec = "iag", veh = "veh", fuel = "G", eu = "Exhaust"))
pm <- units::set_units(pm, "g/km^2/h")
(dfb <- speciate(as.data.frame(pm), spec = "pmiaag", veh = "veh", fuel = "G", eu = "Exhaust"))
(dfb <- speciate(as.data.frame(pm), spec = "pmneu", veh = "veh", fuel = "G", eu = "Exhaust"))
(dfb <- speciate(as.data.frame(pm), spec = "pmneu2", veh = "veh", fuel = "G", eu = "Exhaust"))
# new
(pah <- speciate(spec = "pah", veh = "LDV", fuel = "G", eu = "I"))
(xs <- speciate(spec = "pccd", veh = "LDV", fuel = "G", eu = "I"))
(xs <- speciate(spec = "pmchar", veh = "LDV", fuel = "G", eu = "I"))
(xs <- speciate(spec = "metals", veh = "LDV", fuel = "G", eu = "all"))
## End(Not run)
```
Description

Speed returns a transformed object with class "Speed" and units km/h. This function includes two arguments, distance and time. Therefore, it is possible to change the units of the speed to "m" to "s" for example. This function returns a data.frame with units for speed. When this function is applied to numeric vectors it adds class "units".

Usage

```r
Speed(x, ..., dist = "km", time = "h")
```

## S3 method for class 'Speed'
```
print(x, ...)
```

## S3 method for class 'Speed'
```
summary(object, ...)
```

## S3 method for class 'Speed'
```
plot(
    x,
    pal = "mpl_inferno",
    rev = FALSE,
    fig1 = c(0, 0.8, 0, 0.8),
    fig2 = c(0, 0.8, 0.55, 1),
    fig3 = c(0.7, 1, 0, 0.8),
    mai1 = c(1, 0.82, 0.82, 0.42),
    mai2 = c(1.8, 0.82, 0.5, 0.42),
    mai3 = c(1, 1, 0.82, 0.2),
    bias = 1.5,
    ...)
```

Arguments

- `x` Object with class "data.frame", "matrix" or "numeric"
- `...` ignored Default is units is "km"
- `dist` String indicating the units of the resulting distance in speed.
- `time` Character to be the time units as denominator, default is "h"
- `object` Object with class "Speed"
- `pal` Palette of colors available or the number of the position
- `rev` Logical; to internally revert order of rgb color vectors.
- `fig1` par parameters for fig, `par`.  ```
fig2 \( \text{par parameters for fig, } \text{par.} \)  
fig3 \( \text{par parameters for fig, } \text{par.} \)  
mai1 \( \text{par parameters for mai, } \text{par.} \)  
mai2 \( \text{par parameters for mai, } \text{par.} \)  
mai3 \( \text{par parameters for mai, } \text{par.} \)  
bias \( \text{positive number. Higher values give more widely spaced colors at the high end.} \)

**Value**

Constructor for class "Speed" or "units"

**Note**

default time unit for speed is hour

**See Also**

units

**Examples**

```r
{  
data(net)  
data(pc_profile)  
speed <- Speed(net$ps)  
class(speed)  
plot(speed, type = "l")  
pc_week <- temp_fact(net$ldv+net$hdv, pc_profile)  
df <- netspeed(pc_week, net$ps, net$ffs, net$capacity, net$lkm)  
summary(df)  
plot(df)  
  
# changing to miles  
net$ps <- units::set_units(net$ps, "miles/h")  
net$ffs <- units::set_units(net$ffs, "miles/h")  
net$lkm <- units::set_units(net$lkm, "miles")  
df <- netspeed(pc_week, net$ps, net$ffs, net$capacity, net$lkm, dist = "miles")  
plot(df)  
}
```

---

**split_emis**  

*Split street emissions based on a grid*

**Description**

`split_emis` split street emissions into a grid.
Usage

\texttt{split_emis(net, distance, add_column, verbose = TRUE)}

Arguments

- \texttt{net} A spatial dataframe of class "sp" or "sf". When class is "sp" it is transformed to "sf" with emissions.
- \texttt{distance} Numeric distance or a grid with class "sf".
- \texttt{add_column} Character indicating name of column of distance. For instance, if distance is an sf object, and you want to add one extra column to the resulting object.
- \texttt{verbose} Logical, to show more information.

Examples

```r
## Not run:
data(net)
g <- make_grid(net, 1/102.47/2) #500m in degrees
names(net)
dim(net)
netsf <- sf::st_as_sf(net [, "ldv"]
x <- split_emis(net = netsf, distance = g)
dim(x)
g$A <- rep(letters, length = 20)[1:nrow(g)]
g$B <- rev(g$A)
netsf <- sf::st_as_sf(net [, c("ldv", "hdv")]
xx <- split_emis(netsf, g, add_column = c("A", "B"))
## End(Not run)
```

---

temp_fact

\textit{Expansion of hourly traffic data}

Description

temp_fact is a matrix multiplication between traffic and hourly expansion data-frames to obtain a data-frame of traffic at each link to every hour.

Usage

temp_fact(q, pro, net, time)

Arguments

- \texttt{q} Numeric; traffic data per each link
- \texttt{pro} Numeric; expansion factors data-frames
- \texttt{net} SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
- \texttt{time} Character to be the time units as denominator, eg "1/h"
temp_veh

Value

data-frames of expanded traffic or sf.

Examples

```r
## Not run:
# Do not run
data(net)
data(pc_profile)
pc_week <- temp_fact(net$ldv+net$hdv, pc_profile)
plot(pc_week)
pc_weeksf <- temp_fact(net$ldv+net$hdv, pc_profile, net = net)
plot(pc_weeksf)

## End(Not run)
```

temp_veh

Expanded Vehicles data.frame by hour

Description

temp_veh multiplies vehicles with temporal factor

Usage

temp_veh(x, tfs, array = FALSE)

Arguments

  x   Vehicles data.frame
  tfs  temporal factor
  array Logical, to return an array

Value

data.table

See Also

temp_fact
Examples

```r
## Not run:
data(net)
data(pc_profile)
x <- age_ldv(x = net$ldv)
dx <- temp_veh(x = x, tfs = pc_profile[[1]])
plot(Vehicles(as.data.frame(dx[, 1:50])))
dx2 <- temp_veh(x = x, tfs = pc_profile[[1]],
array = TRUE)
plot(EmissionsArray(dx2))
## End(Not run)
```

to_latex creates a .tex a table from a data.frame

description
to_latex reads a data.frame and generates a .tex table, aiming to replicate the method of tablegenerator.com

usage
to_latex(df, file, caption = "My table", label = "tab:df")

arguments
- df data.frame with three column.
- file Character, name of new .tex file
- caption Character caption of table
- label Character, label of table

value
a text file with extension .tex.

see also
vein_notes long_to_wide
Other helpers: colplot(), dmonth(), wide_to_long()
Examples

## Not run:

```r
def <- data.frame(pollutant = rep(c("CO", "propadiene", "NO2"), 10),
  emission = vein::Emissions(1:30),
  region = rep(letters[1:2], 15))

df
long_to_wide(df)
(df2 <- long_to_wide(df, column_fixed = "region"))
to_latex(df2)
to_latex(long_to_wide(df, column_fixed = "region"),
  file = paste0(tempfile(), ".tex"))

## End(Not run)
```

---

### Vehicles

*Construction function for class "Vehicles"*

#### Description

Vehicles returns a transformed object with class "Vehicles" and units 'veh'. The type of objects supported are of classes "matrix", "data.frame", "numeric" and "array". If the object is a matrix it is converted to `data.frame`. If the object is "numeric" it is converted to class "units".

#### Usage

```r
Vehicles(x, ..., time = NULL)

## S3 method for class 'Vehicles'
print(x, ...)

## S3 method for class 'Vehicles'
summary(object, ...)

## S3 method for class 'Vehicles'
plot(
  x,
  pal = "colo_lightningmccarl_into_the_night",
  rev = TRUE,
  bk = NULL,
  fig1 = c(0, 0.8, 0, 0.8),
  fig2 = c(0, 0.8, 0.55, 1),
  fig3 = c(0.7, 1, 0, 0.8),
  mai1 = c(1, 0.82, 0.82, 0.42),
  mai2 = c(1.8, 0.82, 0.5, 0.42),
  mai3 = c(1, 1, 0.82, 0.2),
  bias = 1.5, 
  ...
)
```

```r
```
Arguments

- **x**: Object with class "Vehicles"
- **...**: ignored
- **time**: Character to be the time units as denominator, eg "1/h"
- **object**: Object with class "Vehicles"
- **pal**: Palette of colors available or the number of the position
- **rev**: Logical; to internally revert order of rgb color vectors.
- **bk**: Break points in sorted order to indicate the intervals for assigning the colors.
- **fig1**: par parameters for fig, `par`
- **fig2**: par parameters for fig, `par`
- **fig3**: par parameters for fig, `par`
- **mai1**: par parameters for mai, `par`
- **mai2**: par parameters for mai, `par`
- **mai3**: par parameters for mai, `par`
- **bias**: positive number. Higher values give more widely spaced colors at the high end.

Value

Objects of class "Vehicles" or "units"

Examples

```r
## Not run:
lt <- rnorm(100, 300, 10)
class(lt)
vlt <- Vehicles(lt)
class(vlt)
plot(vlt)
LT_B5 <- age_hdv(x = lt, name = "LT_B5")
summary(LT_B5)
plot(LT_B5)
## End(Not run)
```

Description

`vein_notes` creates aa text file `.txt` for writting technical notes about this emissions inventory.
Usage

vein_notes(
  notes,
  file = "README",
  yourname = Sys.info()\["login"\],
  title = "Notes for this VEIN run",
  approach = "Top Down",
  traffic = "Your traffic information",
  composition = "Your traffic information",
  ef = "Your information about emission factors",
  cold_start = "Your information about cold starts",
  evaporative = "Your information about evaporative emission factors",
  standards = "Your information about standards",
  mileage = "Your information about mileage"
)

Arguments

notes Character; vector of notes.
file Character; Name of the file. The function will generate a file with an extension `.txt`.
yourname Character; Name of the inventor compiler.
title Character; Title of this file. For instance: "Vehicular Emissions Inventory of Region XX, Base year XX"
approach Character; vector of notes.
traffic Character; vector of notes.
composition Character; vector of notes.
ef Character; vector of notes.
cold_start Character; vector of notes.
evaporative Character; vector of notes.
standards Character; vector of notes.
mileage Character; vector of notes.

Value

Writes a text file.

Examples

## Not run:
# do not run
a <- "delete"
f <- vein_notes("notes", file = a)
file.remove(f)

## End(Not run)
**Estimation of VKM**

**Description**

`vkm` consists in the product of the number of vehicles and the distance driven by these vehicles in km. This function reads hourly vehicles and then extrapolates the vehicles.

**Usage**

```r
vkm(
  veh,  
  lkm,  
  profile,  
  hour = nrow(profile),  
  day = ncol(profile),  
  array = TRUE,  
  as_df = TRUE
)
```

**Arguments**

- `veh` Numeric vector with number of vehicles per street
- `lkm` Length of each link (km)
- `profile` Numerical or dataframe with nrows equal to 24 and ncol 7 day of the week
- `hour` Number of considered hours in estimation
- `day` Number of considered days in estimation
- `array` When FALSE produces a dataframe of the estimation. When TRUE expects a profile as a dataframe producing an array with dimensions (streets x hours x days)
- `as_df` Logical; when TRUE transform returning array in data.frame (streets x hour*days)

**Value**

emission estimation of vkm

**Examples**

```r
## Not run:
# Do not run
pc <- lkm <- abs(rnorm(10,1,1))*100
pro <- matrix(abs(rnorm(24*7,0.5,1)), ncol=7, nrow=24)
vkms <- vkm(veh = pc, lkm = lkm, profile = pro)
class(vkms)
dim(vkms)
vkms2 <- vkm(veh = pc, lkm = lkm, profile = pro, as_df = FALSE)
```
wide_to_long

Transform data.frame from wide to long format

Description

wide_to_long transform data.frame from wide to long format

Usage

wide_to_long(df, column_with_data = names(df), column_fixed, geometry)

Arguments

df data.frame with three column.
column_with_data Character column with data
column_fixed Character, column that will remain fixed
gometry To return a sf

Value

long data.frame.

See Also

emis_hot_td emis_cold_td long_to_wide
Other helpers: colplot(), dmonth(), to_latex()

Examples

## Not run:
data(net)
net <- sf::st_set_geometry(net, NULL)
af <- wide_to_long(df = net)
head(df)

## End(Not run)
Index

* Add distance units
  add_lkm, 5
  add_miles, 6

* China
  ef_china, 25
  ef_china_det, 29
  ef_china_h, 30
  ef_china_hu, 31
  ef_china_long, 32
  ef_china_s, 33
  ef_china_speed, 34
  ef_china_te, 35
  ef_china_th, 36
  emis_china, 73
  emis_long, 95

* age
  age, 9
  age_hdv, 10
  age_ldv, 12
  age_moto, 13

* cold
  cold_mileage, 17
  ef_ldv_cold, 46
  ef_ldv_cold_list, 48

* cumileage
  ef_nitro, 56

* datasets
  decoder, 20
  fe2015, 106
  fkm, 107
  net, 129
  pc_cold, 131
  pc_profile, 132
  pollutants, 132
  profiles, 133

* deterioration
  emis_det, 80

* ef_china
  ef_china, 25

* emission
  ef_cetesb, 21
  ef_china, 25
  ef_eea, 37
  ef_hdv_scaled, 42
  ef_hdv_speed, 43
  ef_im, 45
  ef_ldv_cold, 46
  ef_ldv_cold_list, 48
  ef_ldv_scaled, 49
  ef_ldv_speed, 51
  ef_local, 55
  ef_nitro, 56
  ef_whe, 59
  emis_det, 80

* emitters
  ef_whe, 59

* factors
  ef_cetesb, 21
  ef_china, 25
  ef_eea, 37
  ef_hdv_scaled, 42
  ef_hdv_speed, 43
  ef_im, 45
  ef_ldv_cold, 46
  ef_ldv_cold_list, 48
  ef_ldv_scaled, 49
  ef_ldv_speed, 51
  ef_local, 55
  ef_nitro, 56
  ef_whe, 59
  emis_det, 80

* helpers
  colplot, 18
  dmonth, 21
  to_latex, 143
  wide_to_long, 148

* high
  ef_whe, 59
* mileage
  cold_mileage, 17
  ef_im, 45
* speed
  ef_hdv_scaled, 42
  ef_hdv_speed, 43
  ef_ldv_scaled, 49
  ef_ldv_speed, 51
* start
  ef_ldv_cold_list, 48
* units
  remove_units, 134

add_lkm, 5, 6
add_miles, 5, 6
add_poloid, 6, 6, 104
addsacle, 4
adt, 7, 7
age, 9, 9, 11, 13, 14, 129
age_hdv, 9, 10, 11, 13, 14, 129
age_ldv, 9–12, 12, 13, 14, 129
age_moto, 9–11, 13, 13, 14, 129
aw, 15, 15
celsius, 16
check_nt, 17, 61, 78, 91
cold_mileage, 17
colplot, 18, 18, 21, 143, 148
decoder, 20
dmonth, 19, 21, 143, 148
ef_cetesb, 21, 21, 37, 55, 56
ef_china, 25, 25, 29–36, 75, 91, 96
ef_china_det, 27, 29, 30–36, 75, 96
ef_china_h, 27, 29, 30, 31–36, 75, 96
ef_china_hu, 27, 29, 30, 31, 32–36, 75, 96
ef_china_long, 27, 29–31, 32, 33–36, 75, 96
ef_china_s, 27, 29–32, 33, 34–36, 75, 96
ef_china_speed, 27, 29–33, 34, 35, 36, 75, 96
ef_china_te, 27, 29–34, 35, 36, 75, 96
ef_china_th, 27, 29–33, 36, 75, 96
ef_eea, 37
ef_emfac, 38, 38
ef_evap, 39, 39, 71, 85
ef_fun, 41, 41
ef_hdv_scaled, 42, 42
ef_hdv_speed, 43, 70, 71
ef_im, 45, 45
ef_ldv_cold, 45, 46, 46, 53, 78
ef_ldv_cold_list, 48
ef_ldv_scaled, 49
ef_ldv_speed, 21, 27, 51, 51, 70, 71, 91, 135
ef_local, 55, 55
ef_nitro, 56, 56
ef_wear, 58, 58
ef_whe, 59, 59
emis, 45, 51, 60, 60
emis_chem, 70, 70
emis_chem2, 71, 71
emis_china, 27, 29–36, 73, 96
emis_cold, 75, 76
emis_cold_td, 77, 77, 117, 148
emis_det, 45, 80, 80
emis_dist, 82, 82, 112
emis_emfac, 83, 83
emis_evap, 84, 84
emis_evap2, 85
emis_grid, 88, 88, 112
emis_hod_td, 27, 90, 90, 117, 148
emis_long, 27, 29–36, 75, 95
emis_merge, 95, 96, 97
emis_order, 98
emis_paved, 100
emis_post, 97, 101
emis_to_streets, 7, 104, 104
emis_wear, 105
EmissionFactors, 64
EmissionFactorsList, 66
Emissions, 67
EmissionsArray, 69
fe2015, 106
fkm, 107
fuel_corr, 37, 44, 45, 47, 52, 53, 108
get_project, 109, 109
grid_emis, 112, 112
GriddedEmissionsArray, 99, 110
invcop, 114
inventory, 115
long_to_wide, 116, 116, 143, 148
make_grid, 117, 118
moves_ef, 118, 118
moves_rpd, 119, 119
INDEX

moves_rpdy, 121, 121
moves_rpdy_meta, 122, 122
moves_rpdy_sf, 123, 123
moves_rpsy_meta, 125, 125
moves_rpsy_sf, 126, 126
moves_speed, 127
my_age, 9, 11, 13, 14, 128, 129
net, 129
netspeed, 130
par, 19, 65, 67, 68, 139, 140, 145
pc_cold, 131
pc_profile, 132
plot.EmissionFactors (EmissionFactors), 64
plot.EmissionFactorsList
   (EmissionFactorsList), 66
plot.Emissions (Emissions), 67
plot.EmissionsArray (EmissionsArray), 69
plot.GriddedEmissionsArray
   (GriddedEmissionsArray), 110
plot.Speed (Speed), 139
plot.Vehicles (Vehicles), 144
pollutants, 132
print.EmissionFactors
   (EmissionFactors), 64
print.EmissionFactorsList
   (EmissionFactorsList), 66
print.Emissions (Emissions), 67
print.EmissionsArray (EmissionsArray), 69
print.GriddedEmissionsArray
   (GriddedEmissionsArray), 110
print.Speed (Speed), 139
print.Vehicles (Vehicles), 144
profiles, 133
remove_units, 134, 134
speciate, 44, 45, 52, 71, 73, 135
Speed, 139
split_emis, 140, 140
summary.EmissionFactors
   (EmissionFactors), 64
summary.EmissionFactorsList
   (EmissionFactorsList), 66
summary.Emissions (Emissions), 67
summary.EmissionsArray
   (EmissionsArray), 69
summary.GriddedEmissionsArray
   (GriddedEmissionsArray), 110
summary.Speed (Speed), 139
summary.Vehicles (Vehicles), 144
temp_fact, 141, 142
temp_veh, 142, 142
title, 19
to_latex, 19, 21, 143, 143, 148
units, 140
Vehicles, 144
vein_notes, 143, 145, 145
vkm, 147
weekly (emis_order), 98
wide_to_long, 19, 21, 117, 143, 148, 148