Package 'vein'

June 11, 2018

Type Package

Title Vehicular Emissions Inventories

Version 0.3.35

Description Elaboration of vehicular emissions inventories, consisting in four stages, pre-processing activity data, preparing emissions factors, estimating the emissions and post-processing of emissions in maps and databases.

License MIT + file LICENSE

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

BugReports https://github.com/atmoschem/vein/issues/

LazyData no

Depends R (>= 2.10)

Imports sf, sp, data.table, graphics, stats, units, methods, eixport

Suggests knitr, rmarkdown, testthat, covr, lwgeom, cptcity

RoxygenNote 6.0.1

NeedsCompilation no

Author Sergio Ibarra-Espinosa [aut, cre] (<https://orcid.org/0000-0002-3162-1905>)

Maintainer Sergio Ibarra-Espinosa <sergio.ibarra@usp.br>

Repository CRAN

Date/Publication 2018-06-11 13:43:46 UTC

R topics documented:

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

Description

This function calculates ADT based on hourly traffic data. The input traffic data is usually for morning rush hours.

Usage

```
adt(pc, lcv, hgv, bus, mc, p_pc, p_lcv, p_hgv, p_bus, p_mc, expanded = FALSE)
```

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.
expanded	boolean argument for returning numeric vector or "Vehicles"

Value

numeric vector of total volume of traffic per link, or data-frames of expanded traffic

Examples

```
{
data(net)
data(pc_profile)
p1 <- pc_profile[, 1]
adt1 <- adt(pc = net$ldv*0.75,
            lcv = net$ldv*0.1,
            hgv = net$hdv,
            bus = 0,
            mc = net$ldv*0.15,
            p_pc = p1,
            p_lcv = p1,
            p_hgv = p1,
            p_bus = p1,
```

```
            p_mc = p1)
head(adt1)
plot(adt1)
adt2 <- adt(pc = net$ldv*0.75,
            lcv = net$ldv*0.1,
            hgv = net$hdv,
            bus = net$hdv,
            mc = net$ldv*0.15,
            p_pc = p1,
            p_lcv = p1,
            p_hgv = p1,
            p_bus = p1*0, # when zero, must be the same size
            p_mc = p1,
            TRUE)
head(adt2)
plot(adt2) # Class Vehicles
}
```

---

age_hdv *Returns amount of vehicles at each age*

Description

Returns amount of vehicles at each age

Usage

```
age_hdv(x, name = "veh", a = 0.2, b = 17, agemin = 1, agemax = 50,
  k = 1, bystreet = F, net, message = TRUE)
```

Arguments

x	numerical vector of vehicles with length equal to lines features of raod network
name	of vehicle assigned to columns of dataframe
a	parameter of survival equation
b	parameter of survival equation
agemin	age of newest vehicles for that category
agemax	age of oldest vehicles for that category
k	multiplication factor
bystreet	when TRUE it is expecting that 'a' and 'b' are numeric vectors with length equal to x
net	SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
message	message with average age and total numer of vehicles

Value

dataframe of age distrubution of vehicles

Examples

```
{
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)
}
```

`age_ldv` *Returns amount of vehicles at each age*

Description

Returns amount of vehicles at each age

Usage

```
age_ldv(x, name = "veh", a = 1.698, b = -0.2, agemin = 1, agemax = 50,
  k = 1, bystreet = F, net, message = TRUE)
```

Arguments

`x`	numerical vector of vehicles
`name`	word of vehicle assigned to columns of dataframe
`a`	parameter of survival equation
`b`	parameter of survival equation
`agemin`	age of newest vehicles for that category
`agemax`	age of oldest vehicles for that category
`k`	multiplication factor
`bystreet`	when TRUE it is expecting that 'a' and 'b' are numeric vectors with length equal to x
`net`	SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
`message`	message with average age and total numer of vehicles

Value

dataframe of age distrubution of vehicles

Examples

```
{
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)
}
```

age_moto *Returns amount of vehicles at each age*

Description

Returns amount of vehicles at each age

Usage

```
age_moto(x, name = "veh", a = 0.2, b = 17, agemin = 1, agemax = 50,
  k = 1, bystreet = F, net, message = TRUE)
```

Arguments

x	numerical vector of vehicles
name	of vehicle assigned to columns of dataframe
a	parameter of survival equation
b	parameter of survival equation
agemin	age of newest vehicles for that category
agemax	age of oldest vehicles for that category
k	multiplication factor
bystreet	when TRUE it is expecting that 'a' and 'b' are numeric vectors with length equal to x
net	SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
message	message with average age and total numer of vehicles

Value

dataframe of age distrubution of vehicles

Examples

```
{
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)
}
```

`ef_evap` *Evaporative emission factor*

Description

A lookup table with tier 2 evaporative emission factors from EMEP/EEA emisison guidelines

Usage

```
ef_evap(ef, v, cc, dt, ca, k = 1, show = FALSE)
```

Arguments

`ef`	Name of evaporative emission factor as *eshotc*: mean hot-soak with carburator, *eswarmc*: mean cold and warm-soak with carburator, eshotfi: mean hot-soak with fuel injection, *erhotc*: mean hot running losses with carburator, *erwarmc* mean cold and warm running losses, *erhotfi* mean hot running losses with fuel injection
`v`	Type of vehicles, "PC", "Motorcycles", "Motorcycles_2S" and "Moped"
`cc`	Size of engine in cc. PC "<=1400", "1400_2000" and "2000" Motorcycles_2S: "<=50". Motorcyces: ">50", "<250", "250_750" and ">750"
`dt`	Average daily temperature variation: "-5_10", "0_15", "10_25" and "20_35"
`ca`	Size of canister: "no" meaning no canister, "small", "medium" and "large"
`k`	multiplication factor
`show`	when TRUE shows row of table with respective emission factor.

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/proceed the argument show must by T

References

Mellios G and Ntziachristos 2016. Gasoline evaporation. In: EEA, EMEP. EEA air pollutant emission inventory guidebook-2009. European Environment Agency, Copenhagen, 2009

Examples

```
## Not run:
# Do not run
ef_evap(ef = "erhotc",v = "PC", cc = "<=1400", dt = "0_15", ca = "no",
show = TRUE)

## End(Not run)
```

`ef_hdv_scaled` *Scaling constant with speed emission factors of Heavy Duty Vehicles*

Description

This function creates a list of scaled functions of emission factors. A scaled emission factor which at a speed of the dricing 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

```
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"
`g`	Gross weight of each category: "<=18", ">18", "<=15", ">15 & <=18", "<=7.5", ">7.5 & <=12", ">12 & <=14", ">14 & <=20", ">20 & <=26", ">26 & <=28", ">28 & <=32", ">32", ">20 & <=28", ">28 & <=34", ">34 & <=40", ">40 & <=50" or ">50 & <=60"
`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

Examples

```
{
# Do not run
data(fe2015)
co1 <- fe2015[fe2015$Pollutant=="CO",]
lef <- ef_hdv_scaled(co1, co1$LT, v = "Trucks", t = "RT",
```

```
g = "<=7.5", eu = co1$Euro_HDV, gr = 0, l = 0.5, p = "CO")
length(lef)
plot(x = 0:150, y = lef[[36]](0:150), col = "red", type = "b", ylab = "[g/km]",
pch = 16, xlab = "[km/h]",
main = "Variation of emissions with speed of oldest vehicle")
plot(x = 0:150, y = lef[[1]](0:150), col = "blue", type = "b", ylab = "[g/km]",
pch = 16, xlab = "[km/h]",
main = "Variation of emissions with speed of newest vehicle")
}
```

`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 guidelines 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)
```

Arguments

v	Category vehicle: "Coach", "Trucks" or "Ubus"
t	Sub-category of of vehicle: "3Axes", "Artic", "Midi", "RT, "Std" and "TT"
g	Gross weight of each category: "<=18", ">18", "<=15", ">15 & <=18", "<=7.5", ">7.5 & <=12", ">12 & <=14", ">14 & <=20", ">20 & <=26", ">26 & <=28", ">28 & <=32", ">32", ">20 & <=28", ">28 & <=34", ">34 & <=40", ">40 & <=50" or ">50 & <=60"
eu	Euro emission standard: "PRE", "I", "II", "III", "IV" and "V"
x	Numeric; if pollutant is "SO2", it is sulphur in fuel in ppm, if is "Pb", Lead in fuel in ppm.
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	Character; pollutant: "CO", "FC", "NOx", "HC", "PM", "NMHC", "CH4", "CO2", "SO2" or "Pb". Only when p is "SO2" pr "Pb" x is needed. Also polycyclic aromatic hydrocarbons (PAHs) and persistent organi pollutants (POPs).
k	Multiplication factor
show.equation	Option to see or not the equation parameters

Value

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

Note

Pollutants: "CO", "NOx", "HC", "PM", "CH4", "NMHC", "CO2", "SO2", "Pb".

PAH and POP: "indeno(1,2,3-cd)pyrene", "benzo(k)fluoranthene", "benzo(b)fluoranthene", "benzo(ghi)perylene", "fluoranthene", "benzo(a)pyrene", "pyrene", "perylene", "anthanthrene", "benzo(b)fluorene", "benzo(e)pyrene", "triphenylene", "benzo(j)fluoranthene", "dibenzo(a,j)anthacene", "dibenzo(a,l)pyrene", "3,6-dimethylphenanthrene", "benzo(a)anthracene", "acenaphthylene", "acenapthene", "fluorene", "chrysene", "phenanthrene", "napthalene", "anthracene", "coronene", "dibenzo(ah)anthracene".

Dioxins and furans: PCDD, PCDF and PCB expressed as (g equivalent toxicity / km).

Metals: "As", "Cd", "Cr", "Cu", "Hg", "Ni", "Pb", "Se", "Zn" (g/km). **NMHC**:

ALKANES: "ethane", "propane", "butane", "isobutane", "pentane", "isopentane", "hexane", "heptane", "octane", "TWO_methylhexane", "nonane", "TWO_methylheptane", "THREE_methylhexane", "decane", "THREE_methylheptane", "alcanes_C10_C12", "alkanes_C13".

CYCLOALKANES: "cycloalcanes".

ALKENES: "ethylene", "propylene", "propadiene", "ONE_butene", "isobutene", "TWO_butene", "ONE_3_butadiene", "ONE_pentene", "TWO_pentene", "ONE_hexene", "dimethylhexene".

ALKYNES:"ONE_butine", "propine", "acetylene".

ALDEHYDES: "formaldehyde", "acetaldehyde", "acrolein", "benzaldehyde", "crotonaldehyde", "methacrolein", "butyraldehyde", "isobutanaldehyde", "propionaldehyde", "hexanal", "i_valeraldehyde", "valeraldehyde", "o_tolualdehyde", "m_tolualdehyde", "p_tolualdehyde".

KETONES: "acetone", "methylethlketone".

AROMATICS: "toluene", "ethylbenzene", "m_p_xylene", "o_xylene", "ONE_2_3_trimethylbenzene", "ONE_2_4_trimethylbenzene", "ONE_3_5_trimethylbenzene", "styrene", "benzene", "C9", "C10", "C13".

Examples

```
{
# Quick view
pol <- c("CO", "NOx", "HC", "NMHC", "CH4", "FC", "PM", "CO2", "SO2")
f <- sapply(1:length(pol), function(i){
print(pol[i])
ef_hdv_speed(v = "Trucks",t = "RT", g = "<=7.5", e = "II", gr = 0,
l = 0.5, p = pol[i], x = 10)(30)
})
f
# PAH POP
ef_hdv_speed(v = "Trucks",t = "RT", g = "<=7.5", e = "II", gr = 0,
l = 0.5, p = "napthalene", x = 10)(30)
ef_hdv_speed(v = "Trucks",t = "RT", g = "<=7.5", e = "II", gr = 0,
l = 0.5, p = "fluoranthene", x = 10)(30)

# Dioxins and Furans
ef_hdv_speed(v = "Trucks",t = "RT", g = "<=7.5", e = "II", gr = 0,
l = 0.5, p = "PCB", x = 10)(30)

# NMHC
ef_hdv_speed(v = "Trucks",t = "RT", g = "<=7.5", e = "II", gr = 0,
```

```
l = 0.5, p = "heptane", x = 10)(30)

V <- 0:130
ef1 <- ef_hdv_speed(v = "Trucks",t = "RT", g = "<=7.5", e = "II", gr = 0,
l = 0.5, p = "HC")
plot(1:130, ef1(1:130), pch = 16, type = "b")
euro <- c(rep("V", 5), rep("IV", 5), rep("III", 5), rep("II", 5),
          rep("I", 5), rep("PRE", 15))
lef <- lapply(1:30, function(i) {
ef_hdv_speed(v = "Trucks", t = "RT", g = ">32", gr = 0,
eu = euro[i], l = 0.5, p = "NOx",
show.equation = FALSE)(25) })
efs <- EmissionFactors(unlist(lef)) #returns 'units'
plot(efs, xlab = "age")
lines(efs, type = "l")
}
```

`ef_ldv_cold` *Cold-Start Emissions factors for Light Duty Vehicles*

Description

This function returns speed functions 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

Usage

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

Arguments

v	Category vehicle: "LDV"
ta	Ambient temperature. Monthly men can be used
cc	Size of engine in cc: "<=1400", "1400_2000" or ">2000"
f	Type of fuel: "G", "D" or "LPG"
eu	Euro standard: "PRE", "I", "II", "III", "IV", "V", "VI" or "VIc"
p	Pollutant: "CO", "FC", "NOx", "HC" or "PM"
k	Multiplication factor
show.equation	Option to see or not the equation parameters

Value

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

Examples

```
{
ef1 <- ef_ldv_cold(ta = 15, cc = "<=1400", f ="G", eu = "I", p = "CO")
ef1(10)
}
```

`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

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

Arguments

df	Dataframe with local emission factor
v	Category vehicle: "LDV"
ta	ambient temperature. Montly average van be used
cc	Size of engine in cc: <=1400", "1400_2000" and ">2000"
f	Type of fuel: "G" or "D"
eu	character vector of euro standards: "PRE", "I", "II", "III", "IV", "V", "VI" or "VIc".
p	Pollutant: "CO", "FC", "NOx", "HC" or "PM"

Value

A list of cold start 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

Examples

```
{
# 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)
}
```

`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

```
ef_ldv_scaled(df, dfcol, SDC = 34.12, v, t = "4S", cc, f, eu, 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: "PC", "LCV", "Motorcycle" or "Moped
`t`	Sub-category of 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"
`cc`	Size of engine in cc: PC: "<=1400", ">1400", "1400_2000", ">2000", "<=800", "<=2000". Motorcycle: ">=50" (for "2S"), "<=250", "250_750", ">=750". Moped: "<=50". LCV : "<3.5" for gross weight.
`f`	Type of fuel: "G", "D", "LPG" or "FH" (Full Hybrid: starts by electric motor)
`eu`	Euro standard: "PRE", "I", "II", "III", "III+DPF", "IV", "V", "VI", "VIc"
`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

```
{
data(fe2015)
co1 <- fe2015[fe2015$Pollutant=="CO", ]
lef <- ef_ldv_scaled(dfcol = co1$PC_G, v = "PC", t = "4S", cc = "<=1400", f = "G",
eu = co1$Euro_LDV, p = "CO")
length(lef)
lef[[1]](40) # First element of the lit of speed functions at 40 km/h
lef[[36]](50) # 36th element of the lit of speed functions at 50 km/h
plot(x = 0:150, y = lef[[36]](0:150), col = "red", type = "b", ylab = "[g/km]",
pch = 16, xlab = "[km/h]",
main = "Variation of emissions with speed of oldest vehicle")
plot(x = 0:150, y = lef[[1]](0:150), col = "blue", type = "b", ylab = "[g/km]",
pch = 16, xlab = "[km/h]",
main = "Variation of emissions with speed of newest vehicle")
}
```

---

ef_ldv_speed *Emissions factors for Light Duty Vehicles and Motorcycles*

Description

ef_ldv_speed returns speed dependent 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

```
ef_ldv_speed(v, t = "4S", cc, f, eu, p, x, k = 1, show.equation = FALSE)
```

Arguments

v	Character; category vehicle: "PC", "LCV", "Motorcycle" or "Moped
t	Character; sub-category of 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"
cc	Character; size of engine in cc: PC: "<=1400", ">1400", "1400_2000", ">2000", "<=800", "<=2000". Motorcycle: ">=50" (for "2S"), "<=250", "250_750", ">=750". Moped: "<=50". LCV : "<3.5" for gross weight.

f	Character; type of fuel: "G", "D", "LPG" or "FH" (Full Hybrid: starts by electric motor)
eu	Character; euro standard: "PRE", "I", "II", "III", "III+DPF", "IV", "V", "VI" or "VIc"
p	Character; pollutant: "CO", "FC", "NOx", "HC", "PM", "NMHC", "CH4", "CO2", "SO2" or "Pb". Only when p is "SO2" pr "Pb" x is needed. Also polycyclic aromatic hydrocarbons (PAHs) and persistent organi pollutants (POPs).
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
show.equation	Logical; option to see or not the equation parameters

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.

Pollutants: "CO", "NOx", "HC", "PM", "CH4", "NMHC", "CO2", "SO2", "Pb", "FC".

PAH and POP: "indeno(1,2,3-cd)pyrene", "benzo(k)fluoranthene", "benzo(b)fluoranthene", "benzo(ghi)perylene", "fluoranthene", "benzo(a)pyrene", "pyrene", "perylene", "anthanthrene", "benzo(b)fluorene", "benzo(e)pyrene", "triphenylene", "benzo(j)fluoranthene", "dibenzo(a,j)anthacene", "dibenzo(a,l)pyrene", "3,6-dimethylphenanthrene", "benzo(a)anthracene", "acenaphthylene", "acenapthene", "fluorene", "chrysene", "phenanthrene", "napthalene", "anthracene", "coronene", "dibenzo(ah)anthracene" (g/km).

Dioxins and furans: "PCDD", "PCDF" and "PCB" expressed as (g equivalent toxicity / km).

Metals: "As", "Cd", "Cr", "Cu", "Hg", "Ni", "Pb", "Se", "Zn" (g/km).

NMHC:

ALKANES: "ethane", "propane", "butane", "isobutane", "pentane", "isopentane", "hexane", "heptane", "octane", "TWO_methylhexane", "nonane", "TWO_methylheptane", "THREE_methylhexane", "decane", "THREE_methylheptane", "alcanes_C10_C12", "alkanes_C13".

CYCLOALKANES: "cycloalcanes".

ALKENES: "ethylene", "propylene", "propadiene", "ONE_butene", "isobutene", "TWO_butene", "ONE_3_butadiene", "ONE_pentene", "TWO_pentene", "ONE_hexene", "dimethylhexene".

ALKYNES:"ONE_butine", "propine", "acetylene".

ALDEHYDES: "formaldehyde", "acetaldehyde", "acrolein", "benzaldehyde", "crotonaldehyde", "methacrolein", "butyraldehyde", "isobutanaldehyde", "propionaldehyde", "hexanal", "i_valeraldehyde", "valeraldehyde", "o_tolualdehyde", "m_tolualdehyde", "p_tolualdehyde".

KETONES: "acetone", "methylethlketone".

AROMATICS: "toluene", "ethylbenzene", "m_p_xylene", "o_xylene", "ONE_2_3_trimethylbenzene", "ONE_2_4_trimethylbenzene", "ONE_3_5_trimethylbenzene", "styrene", "benzene", "C9", "C10", "C13".

Examples

```
{
# Do 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]")

# Quick view
pol <- c("CO", "NOx", "HC", "NMHC", "CH4", "FC", "PM", "CO2", "Pb", "SO2")
f <- sapply(1:length(pol), function(i){
ef_ldv_speed("PC", "4S", "<=1400", "G", "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)
ef_ldv_speed(v = "PC",t = "4S", cc = "<=1400", f = "G", eu = "PRE",
p = "napthalene")(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:
euro <- c(rep("V", 5), rep("IV", 5), rep("III", 5), rep("II", 5),
          rep("I", 5), rep("PRE", 15))
speed <- 25
lef <- lapply(1:40, function(i) {
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")

# 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(1:40, 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
eurom <- c(rep("III", 5), rep("II", 5), rep("I", 5), rep("PRE", 25))
lef <- lapply(1:30, function(i) {
ef_ldv_speed(v = "Motorcycle", t = "4S", cc = "<=250", f = "G",
eu = eurom[i], p = "CO",
show.equation = FALSE)(25) })
efs <- EmissionFactors(unlist(lef)) #returns 'units'
plot(efs, xlab = "age")
lines(efs, type = "l")
}
```

`ef_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, cc, f, eu, p, S, k = 1, show.equation = TRUE)
```

Arguments

v	Category vehicle: "PC", "LCV", "LDV", "Motorcycle", "Trucks", "HDV", "HDV-A", "BUS" or "Coach".
t	Type: "Cold", "Hot", "<50", ">=50", ">3.5", "7.5_12", "12_18", "28_34", ">34" and "ALL".

cc	"Urban", "Rural", "Highway" and "ALL".
f	Type of fuel: "G", "D" or "LPG"
eu	Euro standard: "PRE", "I", "II", "III", "IV", "V", "VI", "VIc", "2S", 4S" and "ALL"
p	Pollutant: "N2O", "NH3"
S	Sulphur (ppm). Number.
k	Multiplication factor
show.equation	Option to see or not the equation parameters

Value

an emission factor function which depends on the accumulated mileage

Examples

```
{
# Do not run
efe10 <- ef_nitro(v = "PC", t = "Hot", cc = "Urban", f = "G",
eu = "III", p = "NH3", S = 10,
show.equation = FALSE)
efe50 <- ef_nitro(v = "PC", t = "Hot", cc = "Urban", f = "G",
eu = "III", p = "NH3", S = 50,
show.equation = TRUE)
efe10(10)
efe50(10)
}
```

---

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

```
ef_wear(wear, type, pol = "TSP", speed, load = 0.5, axle = 2)
```

Arguments

wear	Character; type of wear: "tyre", "break" and "road"
type	Character; type of vehicle: "2W", "PC", "LCV", 'HDV"
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

Value

emission factors grams/km

References

Ntziachristos and Boulter 2016. Automobile tyre and break wear and road abrasion. In: EEA, EMEP. EEA air pollutant emission inventory guidebook-2009. European Environment Agency, Copenhagen, 2016

Examples

```
## Not run:
data(net)
data(pc_profile)
pc_week <- temp_fact(net$ldv+net$hdv, pc_profile)
df <- netspeed(pc_week, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
ef <- ef_wear(wear = "tyre", type = "PC", pol = "PM10", speed = df)

## End(Not run)
```

---

emis *Estimation of emissions*

Description

`emis` estimates vehicular emissions as the product of the vehicles on a road, length of the road, emission factor avaliated at the respective speed. $E = VEH * LENGTH * EF(speed)$

Usage

```
emis(veh, lkm, ef, speed = 34, agemax = ifelse(is.data.frame(veh),
  ncol(veh), ncol(veh[[1]])), profile, hour = nrow(profile),
  day = ncol(profile), array = T)
```

Arguments

`veh`	"Vehicles" data-frame or list of "Vehicles" data-frame. Each data-frame as number of columns matching the age distribution of that ype of vehicle. The number of rows is equal to the number of streets link
`lkm`	Length of each link
`ef`	List of functions of emission factors
`speed`	Speed data-frame with number of columns as hours
`agemax`	Age of oldest vehicles for that category
`profile`	Dataframe or Matrix with nrows equal to 24 and ncol 7 day of the week
`hour`	Number of considered hours in estimation. Default value us number of rows of profile

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 columns x hours x days)

Value

emission estimation g/h

Note

If the user apply 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. Hour and day will be deprecate because they can be infered 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)
veh <- data.frame(PC_G = PC_G)
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)
lef <- EmissionFactorsList(fe2015[fe2015$Pollutant=="CO", "PC_G"])
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed,
             profile = 1)
# Estimation for 168 hour and local factors
pcw <- temp_fact(net$ldv+net$hdv, pc_profile)
speed <- netspeed(pcw, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
lef <- EmissionFactorsList(fe2015[fe2015$Pollutant=="CO", "PC_G"])
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed,
             profile = profiles$PC_JUNE_2014)
summary(E_CO)
# Estimation for 168 hour and COPERT factors
pcw <- temp_fact(net$ldv+net$hdv, pc_profile)
speed <- netspeed(pcw, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
euro <- as.character(fe2015[fe2015$Pollutant=="CO", "Euro_LDV"])
lef <- lapply(1:length(euro), function(i) {
ef_ldv_speed(v = "PC", t = "4S", cc = "<=1400", f = "G", p = "CO",
             eu= euro[i], show.equation = FALSE)
})
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed,
             profile = profiles$PC_JUNE_2014)
```

```
# Estimation for 168 hour and scaled factors
pcw <- temp_fact(net$ldv+net$hdv, pc_profile)
speed <- netspeed(pcw, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
pckm <- fkm[[1]](1:24); 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)
length(lef) != ncol(pc1)
#emis change length of 'ef' to match ncol of 'veh'
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed,
             profile = profiles$PC_JUNE_2014)
class(E_CO)
lpc <- list(pc1, pc1)
E_COv2 <- emis(veh = lpc,lkm = net$lkm, ef = lef, speed = speed,
               hour = 2, day = 1)
# Entering wrong results
pc1[ , ncol(pc1) + 1] <- pc1$PC_1
dim(pc1)
length(lef)
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed,
             profile = profiles$PC_JUNE_2014)
E_COv2 <- emis(veh = lpc,lkm = net$lkm, ef = lef, speed = speed,
               hour = 2, day = 1)

## End(Not run)
```

`EmissionFactors` *Construction function for class "EmissionFactors"*

Description

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

Usage

```
EmissionFactors(x, ...)

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

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

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

Arguments

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

Value

Objects of class "EmissionFactors" or "units"

Examples

```
{
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)
}
```

EmissionFactorsList *Construction function for class "EmissionFactorsList"*

Description

EmissionFactorsList returns a tranformed 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

```
{
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)
ef1
}
```

Emissions *Construction function for class "Emissions"*

Description

Emissions returns a tranformed 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

```
Emissions(x, ...)

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

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

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

Arguments

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

Value

Objects of class "Emissions" or "units"

Examples

```
## Not run:
data(net)
data(pc_profile)
data(fe2015)
data(fkm)
PC_G <- c(33491,22340,24818,31808,46458,28574,24856,28972,37818,49050,87923,
          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$lkm, alpha = 1,
isList = T)
pckm <- fkm[[1]](1:24); pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[12:24])
#vehicles newer than pre-euro
co1 <- fe2015[fe2015$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",  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, hour = 24, day = 7, array = T)
dim(E_CO) # streets x vehicle categories x hours x days
class(E_CO[ , , 1, 1])
df <- Emissions(E_CO[ , , 1, 1]) # Firt hour x First day
class(df)
summary(df)
head(df)
plot(df)

## End(Not run)
```

EmissionsArray *Construction function for class "EmissionsArray"*

Description

EmissionsArray returns a tranformed object with class "EmissionsArray" with 4 dimensios.

Usage

```
EmissionsArray(x, ...)

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

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

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

Arguments

x	Object with class "data.frame", "matrix" or "numeric"
...	ignored
object	object with class "EmissionsArray'

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)
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,
isList = T)
pckm <- fkm[[1]](1:24); 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$PC_G[25:nrow(co1)])
lef <- ef_ldv_scaled(co1, cod, v = "PC", cc = "<=1400",
                     f = "G",p = "CO", eu=co1$Euro_LDV)
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed, agemax = 41,
             profile = pc_profile, hour = 24, day = 7, array = T)
class(E_CO)
summary(E_CO)
E_CO
plot(E_CO)
lpc <- list(pc1, pc1)
E_COv2 <- emis(veh = lpc,lkm = net$lkm, ef = lef, speed = speed, agemax = 41,
             profile = pc_profile, hour = 2, day = 1)

## End(Not run)
```

`EmissionsList` *Construction function for class "EmissionsList"*

Description

`EmissionsList` returns a tranformed object with class "EmissionsList".

Usage

```
EmissionsList(x, ...)

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

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

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

Arguments

`x`	object with class "EmissionList"
`...`	ignored
`object`	object with class "EmissionList"

Value

Objects of class "EmissionsList" and numeric elements as "units"

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)
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$lkm, alpha = 1,
isList = T)
pckm <- fkm[[1]](1:24); pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[12:24])
#vehicles newer than pre-euro
co1 <- fe2015[fe2015$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", 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, hour = 24, day = 7, array = F)
class(E_CO)

## End(Not run)
```

`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 avaliated at the respective speed.The estimation considers beta parameter, the fraction of mileage driven

Usage

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

Arguments

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

lkm	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	Datraframe 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
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 columns x hours x days)

Value

EmissionsArray g/h

Note

Actually dcold is not necessary, it would be enough to multiply an existing cold-start distribution with the daily profile, but it was added because it is important to clarify both, the data and the concepts. Hour and day will be deprecate because they can be infered from the profile matrix.

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$lkm, alpha = 1)
pckm <- fkm[[1]](1:24); pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[12:24])
#vehicles newer than pre-euro
co1 <- fe2015[fe2015$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", cc = "<=1400",
                     f = "G",p = "CO", eu=co1$Euro_LDV)
# Mohtly average temperature 18 Celcius degrees
lefec <- ef_ldv_cold_list(df = co1, ta = 18, cc = "<=1400", f = "G",
                          eu = co1$Euro_LDV, p = "CO" )
lefec <- c(lefec,lefec[length(lefec)], lefec[length(lefec)],
           lefec[length(lefec)], lefec[length(lefec)],
           lefec[length(lefec)])
length(lefec) == ncol(pc1)
#emis change length of 'ef' to match ncol of 'veh'
class(lefec)
PC_CO_COLD <- emis_cold(veh = pc1, lkm = net$lkm, ef = lef, efcold = lefec,
beta = pcf, speed = speed, profile = pc_profile)
class(PC_CO_COLD)
plot(PC_CO_COLD)
lpc <- list(pc1, pc1)
PC_CO_COLDv2 <- emis_cold(veh = pc1, lkm = net$lkm, ef = lef, efcold = lefec,
beta = pcf, speed = speed, profile = pc_profile, hour = 2,
day = 1)
class(PC_CO_COLDv2)
plot(PC_CO_COLDv2)

## 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/emep-eea-air-pollutant-emission-inventory-guidebook This function subset an internal database of emission factors with each argument

Usage

```
emis_det(po, cc, eu, km)
```

Arguments

`po`	Pollutant
`cc`	Size of engine in cc
`eu`	Euro standard: "PRE", "I", "II", "III", "III", "IV", "V", "VI"
`km`	mileage in km

Value

It returns a numeric vector without "units"

Examples

```
## Not run:
data(fkm)
pckm <- fkm[[1]](1:24); pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])

## End(Not run)
```

`emis_evap` *Estimation of evaporative emissions*

Description

`emis_evap` performs the estimation of evaporative emissions from EMEP/EEA emisison guidelines with Tier 2.

Usage

```
emis_evap(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 lsit 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 celcius degrees
`nd3`	Number of days with temperature between 10 and 25 celcius degrees
`nd2`	Number of days with temperature between 0 and 15 celcius degrees
`nd1`	Number of days with temperature between -5 and 10 celcius degrees
`hs_nd4`	average daily hot-soak evaporative emissions for days with temperature between 20 and 35 celcius degrees
`hs_nd3`	average daily hot-soak evaporative emissions for days with temperature between 10 and 25 celcius degrees
`hs_nd2`	average daily hot-soak evaporative emissions for days with temperature between 0 and 15 celcius degrees
`hs_nd1`	average daily hot-soak evaporative emissions for days with temperature between -5 and 10 celcius degrees

rl_nd4	average daily running losses evaporative emissions for days with temperature between 20 and 35 celcius degrees
rl_nd3	average daily running losses evaporative emissions for days with temperature between 10 and 25 celcius degrees
rl_nd2	average daily running losses evaporative emissions for days with temperature between 0 and 15 celcius degrees
rl_nd1	average daily running losses evaporative emissions for days with temperature between -5 and 10 celcius degrees
d_nd4	average daily diurnal evaporative emissions for days with temperature between 20 and 35 celcius degrees
d_nd3	average daily diurnal evaporative emissions for days with temperature between 10 and 25 celcius degrees
d_nd2	average daily diurnal evaporative emissions for days with temperature between 0 and 15 celcius degrees
d_nd1	average daily diurnal evaporative emissions for days with temperature between -5 and 10 celcius degrees

Value

dataframe of emission estimation in grams/days

References

Mellios G and Ntziachristos 2016. Gasoline evaporation. In: EEA, EMEP. EEA air pollutant emission inventory guidebook-2009. European Environment Agency, Copenhagen, 2009

Examples

```
{
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,
          84771,55864,36306,21079,20138,17439, 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")
ef1 <- ef_evap(ef = "erhotc",v = "PC", cc = "<=1400", dt = "0_15", ca = "no")
dfe <- emis_evap(veh = pc1,
                 name = "PC",
                 size = "<=1400",
                 fuel = "G",
                 aged = 1:ncol(pc1),
                 nd4 = 10,
                 nd3 = 4,
                 nd2 = 2,
                 nd1 = 1,
                 hs_nd4 = ef1*1:ncol(pc1),
                 hs_nd3 = ef1*1:ncol(pc1),
                 hs_nd2 = ef1*1:ncol(pc1),
```

```
                  hs_nd1 = ef1*1:ncol(pc1),
                  d_nd4 = ef1*1:ncol(pc1),
                  d_nd3 = ef1*1:ncol(pc1),
                  d_nd2 = ef1*1:ncol(pc1),
                  d_nd1 = ef1*1:ncol(pc1),
                  rl_nd4 = ef1*1:ncol(pc1),
                  rl_nd3 = ef1*1:ncol(pc1),
                  rl_nd2 = ef1*1:ncol(pc1),
                  rl_nd1 = ef1*1:ncol(pc1))
lpc <- list(pc1, pc1, pc1, pc1,
            pc1, pc1, pc1, pc1)
dfe <- emis_evap(veh = lpc,
                  name = "PC",
                  size = "<=1400",
                  fuel = "G",
                  aged = 1:ncol(pc1),
                  nd4 = 10,
                  nd3 = 4,
                  nd2 = 2,
                  nd1 = 1,
                  hs_nd4 = ef1*1:ncol(pc1),
                  hs_nd3 = ef1*1:ncol(pc1),
                  hs_nd2 = ef1*1:ncol(pc1),
                  hs_nd1 = ef1*1:ncol(pc1),
                  d_nd4 = ef1*1:ncol(pc1),
                  d_nd3 = ef1*1:ncol(pc1),
                  d_nd2 = ef1*1:ncol(pc1),
                  d_nd1 = ef1*1:ncol(pc1),
                  rl_nd4 = ef1*1:ncol(pc1),
                  rl_nd3 = ef1*1:ncol(pc1),
                  rl_nd2 = ef1*1:ncol(pc1),
                  rl_nd1 = ef1*1:ncol(pc1))
}
```

`emis_grid` *Allocate emissions into a grid*

Description

`emis_grid` allocates emissions proportionally to each grid cell. The process is performed by intersection between geometries and the grid. It means that requires "sr" according with your location for the projection. It is assumed that spobj is a spatial*DataFrame or an "sf" with the pollutants in data. This function return an object class "sf".

Usage

```
emis_grid(spobj, g, sr, type = "lines")
```

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" or "points".

Note

When spobj is a 'Spatial' object (class of sp), they are converted into 'sf'. Also, The aggregation of data ise done with data.table functions.

Examples

```
{
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)
plot(netg["hdv"], axes = 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.

Usage

```
emis_merge(pol = "CO", what = "STREETS.rds", streets = T, net,
  path = "emi", crs)
```

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_CO_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.
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

Value

'Spatial feature' of lines or a dataframe of emissions

Examples

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


## End(Not run)
```

`emis_paved` *Estimation of resuspension emissions from paved roads*

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 anual aggregation should consider moisture factor. It depends on Average Daily Traffic (ADT)

Usage

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

Arguments

veh	Numeric vector with length of elements equals to number of streets It is an array with dimenssions number of streets x hours of day x days of week
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
sL3	Silt loading (g/m2) for roads with ADT > 5000 and <= 1000
sL4	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

Value

emission estimation g/h

References

EPA, 2016. Emission factor documentation for AP-42. Section 13.2.1, Paved Roads. https://www3.epa.gov/ttn/chief/ap42/ch

Examples

```
{
# Do not run
veh <- array(pnorm(q=c(1:100), mean=500, sd = 100),
             c(100,24,7))
W <- veh*1e+05
lkm <-  rnorm(n = 100, mean = 10, sd = 1)
sL1 <- 0.6
emi  <- emis_paved(veh = veh, lkm = lkm, k = 0.65,
                        sL1 = sL1, sL2 = sL1/4, sL3 = sL1/16, sL4 = sL1/32,
                        W = W)
class(emi)
head(emi)
}
```

---

emis_post *Post emissions*

Description

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

Usage

```
emis_post(arra, veh, size, fuel, pollutant, by = "veh", net)
```

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`	Type of vehicle
`size`	Size or weight
`fuel`	Fuel
`pollutant`	Pollutant

by	Type of output, "veh" for total vehicular category , "streets_narrow" or "streets_wide". "streets_wide" 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_wide 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'

Note

This function depends on EmissionsArray objests which currently has 4 dimensions. However, a future version of VEIN will produce EmissionsArray with 3 dimensiones and his fungeorge soros drugsction also will change. This change will be made in order to not produce inconsistencies with previous versions, therefore, if the user count with an EmissionsArry 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)
lef <- EmissionFactorsList(fe2015[fe2015$Pollutant=="CO", "PC_G"])
E_CO <- emis(veh = pc1,lkm = net$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_wide", 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 <- fkm[[1]](1:24); 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$PC_G[25:nrow(co1)])
lef <- ef_ldv_scaled(dfcol = cod, v = "PC",  cc = "<=1400",
                     f = "G",p = "CO", eu=co1$Euro_LDV)
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed, agemax = 41,
             profile = pc_profile)
# arguments required: arra, pollutant ad by
E_CO_STREETS <- emis_post(arra = E_CO, pollutant = "CO", by = "streets_wide")
summary(E_CO_STREETS)
# arguments required: arra, veh, size, fuel, pollutant ad by
E_CO_DF <- emis_post(arra = E_CO,  veh = "PC", size = "<1400", fuel = "G",
pollutant = "CO", by = "veh")
head(E_CO_DF)
# recreating 24 profile
lpc <-list(pc1*0.2, pc1*0.1, pc1*0.1, pc1*0.2, pc1*0.5, pc1*0.8,
           pc1, pc1*1.1, pc1,
           pc1*0.8, pc1*0.5, pc1*0.5,
           pc1*0.5, pc1*0.5, pc1*0.5, pc1*0.8,
           pc1, pc1*1.1, pc1,
           pc1*0.8, pc1*0.5, pc1*0.3, pc1*0.2, pc1*0.1)
E_COv2 <- emis(veh = lpc,  lkm = net$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",
pollutant = "CO", by = "veh")
head(E_CO_DFv2)

## End(Not run)
```

`emis_wear` *Emission estimation from tyre, break and road surface wear*

Description

`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

Ntziachristos and Boulter 2016. Automobile tyre and break wear and road abrasion. In: EEA, EMEP. EEA air pollutant emission inventory guidebook-2009. European Environment Agency, Copenhagen, 2016

Examples

```
{
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
}
```

---

emis_wrf *Generates emissions dataframe to generate WRF-Chem inputs*

Description

emis_wrf returns a dataframes with columns lat, long, id, pollutants, local time and GMT time. This dataframe has the proper format to be used with WRF assimilation system: "ASimilation System 4 WRF (AS4WRF Vera-Vala et al (2016))

Usage

```
emis_wrf(sdf, nr = 1, dmyhm, tz, crs = 4326, islist, utc)
```

Arguments

sdf	Gridded emissions, which can be a SpatialPolygonsDataFrame, or a list of SpatialPolygonsDataFrame, or a sf object of "POLYGON". The user must enter a list with 36 SpatialPolygonsDataFrame with emissions for the mechanism CBMZ.
nr	Number of repetitions of the emissions period
dmyhm	String indicating Day Month Year Hour and Minute in the format "d-m-Y H:M" e.g.: "01-05-2014 00:00" It represents the time of the first hour of emissions in Local Time
tz	Time zone as required in for function as.POSIXct
crs	Coordinate reference system, e.g: "+init=epsg:4326". Used to transform the coordinates of the output
islist	logical value to indicate if sdf is a list or not
utc	ignored.

Value

data-frame of gridded emissions (mass)/h. Remember convert to mol.

Note

The reference of the emissions assimilation system is Vara-Vela, A., Andrade, M. F., Kumar, P., Ynoue, R. Y., and Munoz, A. G.: Impact of vehicular emissions on the formation of fine particles in the Sao Paulo Metropolitan Area: a numerical study with the WRF-Chem model, Atmos. Chem. Phys., 16, 777-797, doi:10.5194/acp-16-777-2016, 2016. A good website with timezones is http://www.timezoneconverter.com/cgi-bin/tzc The crs is the same as used by sp package It returns a dataframe with id,, long, lat, pollutants, time_lt, time_utc and day-UTC-hour (dutch) The pollutants for the CBMZ are: e_so2, e_no, e_ald, e_hcho, e_ora2, e_nh3 e_hc3, e_hc5, e_hc8, e_eth, e_co, e_ol2, e_olt, e_oli, e_tol, e_xyl, e_ket e_csl, e_iso, e_no2, e_ch3oh, e_c2h5oh, e_pm25i, e_pm25j, e_so4i, e_so4j e_no3i, e_no3j, e_orgi, e_orgj, e_eci, e_ecj, e_so4c, e_no3c, e_orgc, e_ecc

See Also

emis_post emis

Examples

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

## End(Not run)
```

`Evaporative` *Construction function for class "Evaporative"*

Description

`Evaporative` returns a tranformed object with class "Evaporative" and units g/day. This class represents the daily emissions presented by Mellios G and Ntziachristos (2016) Gasoline evaporation, Tier 2. Eventually it will be incorporated the techniques of Tier 3.

Usage

```
Evaporative(x, ...)

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

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

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

Arguments

`x`	Object with class "numeric"
`...`	ignored
`object`	Object with class "Evaporative"

Value

Objects of class "Evaporative" or "units"

Examples

```
{
ef1 <- ef_evap(ef = "erhotc",v = "PC", cc = "<=1400", dt = "0_15", ca = "no")
ef1
}
```

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"

Proconve_HDV Proconve emission standard: "PP", "P1", "P2", "P3", "P4", "P5", "P7"

Euro_HDV Euro emission standard equivalence: "PRE", "I", "II", "III", "V"

Promot Promot emission standard: "PP", "M1", "M2", "M3"

Euro_moto Euro emission standard equivalence: "PRE", "I", "II", "III"

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

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

Source

http://veicular.cetesb.sp.gov.br/relatorios-e-publicacoes/

`fkm` *List of functions of mileage in km fro Brazilian fleet*

Description

Functions from CETESB: Antonio de Castro Bruni and Marcelo Pereira Bales. 2013. Curvas de intensidade de uso por tipo de veiculo automotor da frota da cidade de Sao 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

http://veicular.cetesb.sp.gov.br/relatorios-e-publicacoes/

`fuel_corr` *Correction due Fuel effects*

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 nota degradation function.

Usage

```
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 celcius degrees kg/m3), pah ((Back end distillation in Celcius 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.

Examples

```
{
f <- fuel_corr(euro = "I")
names(f)
}
```

GriddedEmissionsArray *Construction function for class "GriddedEmissionsArray"*

Description

GriddedEmissionsArray returns a tranformed object with class "EmissionsArray" with 4 dimensios.

Usage

```
GriddedEmissionsArray(x, ..., cols, rows, times = ncol(x), rotate = FALSE)

## 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	Logical to rotate TRUE or not FALSE the array
object	object with class "EmissionsArray'

Value

Objects of class "GriddedEmissionsArray"

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)
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$lkm, alpha = 1)
pckm <- fkm[[1]](1:24); pckma <- cumsum(pckm)
cod1 <- emis_det(po = "CO", cc = 1000, eu = "III", km = pckma[1:11])
cod2 <- emis_det(po = "CO", cc = 1000, eu = "I", km = pckma[12:24])
#vehicles newer than pre-euro
co1 <- fe2015[fe2015$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$Euro_LDV)
E_CO <- emis(veh = pc1,lkm = net$lkm, ef = lef, speed = speed, agemax = 41,
             profile = pc_profile, hour = 24, day = 7, array = T)
class(E_CO)
E_CO_STREETS <- emis_post(arra = E_CO, pollutant = "CO", by = "streets_wide")
net@data <- cbind(net@data, E_CO_STREETS)
head(net@data)
g <- make_grid(net, 1/102.47/2, 1/102.47/2) #500m in degrees
net@data <- net@data[,- c(1:9)]
names(net)
E_CO_g <- emis_grid(spobj = net, g = g, sr= 31983)
head(E_CO_g) #class sf
library(mapview)
mapview(E_CO_g, zcol= "V1", legend = T, col.regions = cptcity::cptcity(1))
gr <- GriddedEmissionsArray(E_CO_g, rows = 19, cols = 23, times = 168, T)
plot(gr)

# For some cptcity color gradients:
devtools::install_github("ibarraespinosa/cptcity")
plot(gr, col = cptcity::cptcity(1))

## End(Not run)
```

---

hot_soak *Estimation of average running hot-soak evaporative emissions*

Description

hot_soak estimates of evaporative emissions from EMEP/EEA emisison guidelines

Usage

```
hot_soak(x, carb, p, eshotc, eswarmc, eshotfi)
```

Arguments

x	Mean number of trips per vehicle per day
carb	fraction of gasoline vehicles with carburator or fuel return system

p	Fraction of trips finished with hot engine
eshotc	average daily hot-soak evaporative factor for vehicles with carburator or fuel return system
eswarmc	average daily cold-warm-soak evaporative factor for vehicles with carburator or fuel return system
eshotfi	average daily hot-soak evaporative factor for vehicles with fuel injection and returnless fuel systems

Value

numeric vector of emission estimation in grams

References

Mellios G and Ntziachristos 2016. Gasoline evaporation. In: EEA, EMEP. EEA air pollutant emission inventory guidebook-2009. European Environment Agency, Copenhagen, 2009

Examples

```
{
# Do not run
ev <- hot_soak(x = 1:10, carb = 0, p = 1, eshot = 1, eswarmc =1,
eshotfi = 1)
}
```

---

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 outtput 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

Note

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

Examples

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

## End(Not run)
```

`inventory` *Inventory function.*

Description

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

Usage

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

Arguments

name	Character, one word indicating the name of the main directory for running vein. It is better to write the pull path to the new directory.
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 tthere are no vehiles 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.
scripts	Logical value for aggregate or no 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.

Value

Structure of directories and scripts for automating compilation of vehicular emissions inventory. The structure can be used with other 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.

daily: it is for storing the profiles saved as .csv files

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 intance, 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 includes 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 generates objects of class "Vehicles". The user can rename these scripts.

Examples

```
## Not run:
## Not run:
name = file.path(tempdir(), "YourCity")
inventory(name = name, show.dir = TRUE, show.scripts = TRUE)
source(paste0(name, "/main.R"))
## End(**Not run**)

## End(Not run)
```

---

make_grid *Creates rectangular grid for emission allocation*

Description

`make_grid` creates a SpatialGridDataFrame. The spatial reference is taken from the spatial object.

Usage

```
make_grid(spobj, width, height = width, polygon, crs = 4326, ...)
```

Arguments

spobj	A spatial object of class sp or sf or a Character. When it is a character, it is assumed that it is a path to wrfinput file to create a grid class 'sf' based on this file. This is done by running eixport::wrf_grid.
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
...	ignored

Value

A grid of polygons class 'sf'

Examples

```
{
data(net)
grid <- make_grid(net, width = 0.5/102.47) #500 mts
plot(grid, axes = TRUE) #class sf
wrf <- paste(system.file("extdata", package = "eixport"),
"/wrfinput_d02", sep="")
gwrf  <- make_grid(wrf)
plot(gwrf, axes = TRUE)

}
```

---

my_age *Returns amount of vehicles at each age*

Description

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

Usage

```
my_age(x, y, name = "veh", k = 1, net, message = TRUE)
```

Arguments

x	Numeric; vehicles by street (or spatial feature).
y	Numeric; vehicles by age of use
name	of vehicle assigned to columns of dataframe.
k	multiplication factor.
net	SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
message	message with average age and total numer of vehicles.

Value

dataframe of age distrubution of vehicles.

Examples

```
{
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)
}
```

---

net *Road network of the west part of Sao Paulo city*

Description

This dataset is a SpatialLineDataFrame of sp package with roads from a traffic simulations made by CET Sao Paulo, Brazil

Usage

```
data(net)
```

Format

A data frame with 1796 rows and 1 variables:

ldv Light Duty Vehicles (1/h)

hdv Heavy Duty Vehicles (1/h)

lkm Length of the link (km)

ps Peak Speed (km/h)

ffs Free Flow Speed (km/h)

tstreet Type of street

lanes Number of lanes per link

capacity Capacity of vehicles in each link (1/h)

tmin Time for travelling each link (min)

Source

http://www.cetsp.com.br/

`netspeed` *Calculate speeds of traffic network*

Description

`netspeed` Creates a dataframe of speeds fir diferent 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, distance = "km", time = "h", isList)
```

Arguments

`q`	Data-frame of traffic flow to each hour (veh/h)
`ps`	Peak speed (km/h)
`ffs`	Free flow speed (km/h)
`cap`	Capacity of link (veh/h)
`lkm`	Distance of link (km)
`alpha`	Parameter of BPR curves
`beta`	Parameter of BPR curves
`net`	SpatialLinesDataFrame or Spatial Feature of "LINESTRING"
`scheme`	Logical to create a Speed data-frame with 24 hours and a default profile. It needs ffs and ps:

00:00-06:00	ffs
06:00-07:00	average between ffs and ps
07:00-10:00	ps
10:00-17:00	average between ffs and ps
17:00-20:00	ps
20:00-22:00	average between ffs and ps
22:00-00:00	ffs

distance	Deprecated. Character specifying the units for distance. Default is "km"
time	Deprecated. Character specifying the units for time Default is "h".
isList	Deprecated

Value

dataframe speeds with units or sf.

Examples

```
{
data(net)
data(pc_profile)
pc_week <- temp_fact(net$ldv+net$hdv, pc_profile)
df <- netspeed(pc_week, net$ps, net$ffs, net$capacity, net$lkm, alpha = 1)
class(df)
plot(df) #plot of the average speed at each hour, +- sd
df <- netspeed(ps = net$ps, ffs = net$ffs, scheme = TRUE)
class(df)
plot(df) #plot of the average speed at each hour, +- sd
dfsf <- netspeed(ps = net$ps, ffs = net$ffs, scheme = TRUE, net = net)
class(dfsf)
head(dfsf)
plot(dfsf) #plot of the average speed at each hour, +- sd
}
```

---

pc_cold *Profile of Vehicle start patterns*

Description

This dataset is a dataframe with percetage 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

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

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

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

Description

This dataset is n 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, MC, 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

MC_JUNE_2013 168 hours

MC_JUNE_2014 168 hours

HGV_JUNE_2012 168 hours

HGV_JUNE_2013 168 hours

HGV_JUNE_2014 168 hours

PC_JANUARY_2012 168 hours

PC_JANUARY_2013 168 hours

PC_JANUARY_2014 168 hours

LCV_JANUARY_2012 168 hours

LCV_JANUARY_2013 168 hours

LCV_JANUARY_2014 168 hours

MC_JANUARY_2012 168 hours

MC_JANUARY_2014 168 hours

HGV_JANUARY_2012 168 hours

HGV_JANUARY_2013 168 hours

HGV_JANUARY_2014 168 hours

---

running_losses *Estimation of average running losses evaporative emissions*

Description

`running_losses` estimates evaporative emissions from EMEP/EEA emisison guidelines

Usage

```
running_losses(x, carb, p, erhotc, erwarmc, erhotfi)
```

Arguments

x	Mean number of trips per vehicle per day
carb	fraction of gasoline vehicles with carburator or fuel return system
p	Fraction of trips finished with hot engine
erhotc	average daily running losses evaporative factor for vehicles with carburator or fuel return system
erwarmc	average daily cold and warm running losses evaporative factor for vehicles with carburator or fuel return system
erhotfi	average daily hot running losses evaporative factor for vehicles with fuel injection and returnless fuel systems

Value

numeric vector of emission estimation in grams

References

Mellios G and Ntziachristos 2016. Gasoline evaporation. In: EEA, EMEP. EEA air pollutant emission inventory guidebook-2009. European Environment Agency, Copenhagen, 2009

Examples

```
{
# Do not run
ev <- running_losses(x = 1:10, carb = 0, p = 1, erhot = 1, erwarmc =1,
erhotfi = 1)
summary(ev)
}
```

---

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, spec = "bcom", veh, fuel, eu, show = FALSE, list = FALSE, dx)
```

Arguments

x	Emissions estimation
spec	speciation: The speciations are: "bcom", tyre" (or "tire"), "brake", "road", "iag", "nox" and "nmhc". 'iag' now includes a speciation for use of industrial and building paintings. "bcom" stands for black carbon and organic matter. "pmiag" speciates PM2.5 and requires only argument x of PM2.5 emissions in g/h/km^2 as gridded emissions (flux). It also accepts one of the following pollutants: 'e_eth', 'e_hc3', 'e_hc5', 'e_hc8', 'e_ol2', 'e_olt', 'e_oli', 'e_iso', 'e_tol', 'e_xyl', 'e_c2h5oh', 'e_hcho', 'e_ch3oh', 'e_ket', "e_so4i", "e_so4j", "e_no3i", "e_no3j", "e_pm2.5i", "e_pm2.5j", "e_orgi", "e_orgj", "e_eci", "e_ecj". Also "h2o"
veh	Type of vehicle: When spec is "bcom" or "nox" veh can be "PC", "LCV", HDV" or "Motorcycle". When spec is "iag" veh can take two values depending: when the speciation is for vehicles veh accepts "veh", eu "Evaporative", "Liquid" or "Exhaust" and fuel "G", "E" or "D", when the speciation is for painting, veh is "paint" fuel or eu can be "industrial" or "building" when spec is "nmhc", veh can be "LDV" with fuel "G" or "D" and eu "PRE", "I", "II", "III", "IV", "V", or "VI". when spec is "nmhc", veh can be "HDV" with fuel "D" and eu "PRE", "I", "II", "III", "IV", "V", or "VI". when spec is "nmhc" and fuel is "LPG", veh and eu must be "ALL"
fuel	Fuel. When spec is "bcom" fuel can be "G" or "D". When spec is "iag" fuel can be "G", "E" or "D". When spec is "nox" fuel can be "G", "D", "LPG", "E85" or "CNG". Not required for "tyre", "brake" or "road". When spec is "nmhc" fuel can be G, D or LPG.
eu	Euro emission standard: "PRE", "ECE_1501", "ECE_1502", "ECE_1503", "I", "II", "III", "IV", "V", "III-CDFP","IV-CDFP","V-CDFP", "III-ADFP", "IV-ADFP","V-ADFP" and "OPEN_LOOP". When spec is "iag" accept the values "Exhaust" "Evaporative" and "Liquid". When spec is "nox" eu can be "PRE", "I", "II", "III", "IV", "V", "VI", "VIc", "III-DPF" or "III+CRT". Not required for "tyre", "brake" or "road"
show	when TRUE shows row of table with respective speciation
list	when TRUE returns a list with number of elements of the list as the number species of pollutants
dx	Integer, used when spec = "pmiag". It is the spatial disntace

Value

dataframe of speciation in grams or mols

Note

when spec = "iag": veh is only "veh", fuel is "G" (blended with 25% ethanol), "D" (blended with 5% of biodiesel) or "E" (Ethanol 100%). eu is "Evaporative", "Liquid" or "Exhaust",

emissions of "pmiag" speciate pm2.5 into e_so4i, e_so4j, e_no3i, e_no3j, e_mp2.5i, e_mp2.5j, e_orgi, e_orgj, e_eci, e_ecj and h2o. Reference: Rafee, S.: Estudo numerico do impacto das emissoes veiculares e fixas da cidade de Manaus nas concentracoes de poluentes atmosfericos da regiao amazonica, Master thesis, Londrina: Universidade Tecnologica Federal do Parana, 2015.

References

"bcom": Ntziachristos and Zamaras. 2016. Passneger cars, light commercial trucks, heavy-duty vehicles including buses and motor cycles. In: EEA, EMEP. EEA air pollutant emission inventory guidebook-2009. European Environment Agency, Copenhagen, 2016

"tyre", "brake" and "road": Ntziachristos and Boulter 2016. Automobile tyre and brake wear and road abrasion. In: EEA, EMEP. EEA air pollutant emission inventory guidebook-2009. European Environment Agency, Copenhagen, 2016

"iag": Ibarra-Espinosa S. Air pollution modeling in Sao Paulo using bottom-up vehicular emissions inventories. 2017. PhD thesis. Instituto de Astronomia, Geofisica e Ciencias Atmosfericas, Universidade de Sao Paulo, Sao Paulo, page 88. Speciate EPA: https://cfpub.epa.gov/speciate/. : K. Sexton, H. Westberg, "Ambient hydrocarbon and ozone measurements downwind of a large automotive painting plant" Environ. Sci. Tchnol. 14:329 (1980).P.A. Scheff, R.A. Schauer, James J., Kleeman, Mike J., Cass, Glen R., Characterization and Control of Organic Compounds Emitted from Air Pollution Sources, Final Report, Contract 93-329, prepared for California Air Resources Board Research Division, Sacramento, CA, April 1998. 2004 NPRI National Databases as of April 25, 2006, http://www.ec.gc.ca/pdb/npri/npri_dat_rep_e.cfm. Memorandum Proposed procedures for preparing composite speciation profiles using Environment Canada s National Pollutant Release Inventory (NPRI) for stationary sources, prepared by Ying Hsu and Randy Strait of E.H. Pechan Associates, Inc. for David Niemi, Marc Deslauriers, and Lisa Graham of Environment Canada, September 26, 2006.

Examples

```
{
# Do not run
pm <- rnorm(n = 100, mean = 400, sd = 2)
df <- speciate(pm, veh = "PC", fuel = "G", eu = "I")
dfa <- speciate(pm, spec = "e_eth", veh = "veh", fuel = "G", eu = "Exhaust")
dfb <- speciate(pm, spec = "e_tol", veh = "veh", fuel = "G", eu = "Exhaust")
dfc <- speciate(pm, spec = "e_so4i")
}
```

Speed *Construction function for class "Speed"*

Description

Speed returns a tranformed object with class "Speed" and units km/h. This functions includes two arguments, distance and time. Therefore, it is posibel to change the units of the speed to "m" to "s" for example. This function returns a dataframe with units for speed. When this function is applied to numeric vectors it add class "units".

Usage

```
Speed(x, ...)

## S3 method for class 'Speed'
```

```
print(x, ...)

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

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

Arguments

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

Value

Constructor for class "Speed" or "units"

See Also

units

Examples

```
{
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)
}
```

---

temp_fact *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)
```

Arguments

q	Numeric; traffic data per each link
pro	Numeric; expansion factors data-frames
net	SpatialLinesDataFrame or Spatial Feature of "LINESTRING"

Value

data-frames of expanded traffic or sf.

Examples

```
{
# 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)
}
```

Vehicles *Construction function for class "Vehicles"*

Description

Vehicles returns a tranformed object with class "Vehicles" and units 1/h. 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". The function emis_paved needs veh to be an array, therefore in this case, veh must be an array in the total fleet at each street and dimensions total fleet, hours and days

Usage

```
Vehicles(x, ...)

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

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

## S3 method for class 'Vehicles'
plot(x, ..., message = TRUE)
```

Arguments

x	Object with class "Vehicles"
...	ignored
object	Object with class "Vehicles"
message	message with average age

Value

Objects of class "Vehicles" or "units"

Examples

```
{
lt <- rnorm(100, 300, 10)
class(lt)
vlt <- Vehicles(lt)
class(vlt)
plot(vlt)
LT_B5 <- age_hdv(x = lt,name = "LT_B5")
print(LT_B5)
summary(LT_B5)
plot(LT_B5)
}
```

---

vein *vein: a package for elaborating vehicular emissions inventories*

Description

This package provides functions to arrange traffic data, prepare emission factors, estimate emissions and process emissions

Details

1) Inventory

It is recommended to start with the function `inventory` which produces a set of directories and scripts to run vein.

2) Traffic data

The user must count with traffic data at each street at least for one hour. The format of the data must be spatial, either "SpatialLinesDataFrame" or an object class of "sf". Then the user must use any ob tje age functions: `age_ldv`, `age_hdv`, `age_moto` or `my_age`. The outputs of these functions can be saved in directory 'veh' with the extension .rds.

3) Emission factors

The user must chosse a type of emission factor: from Copert with the `ef_ldv_speed` or `ef_hdv_speed`, from local sources as one constant emission factors by age of use of vehicles with `EmissionFactorsList` or as a merge between both with `ef_ldv_scaled` or `ef_hdv_scaled`.

4) Estimating emissions

Once all information is obtained, the user can estimate the emissions with `emis`, `emis_cold` or other.

5) Processing the emissions

The function for processing the emissions ins `emis_post`.

---

vkm *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 vehiles and then extrapolates the vehicles

Usage

```
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

```
{
# 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)
class(vkms2)
dim(vkms2)
}
```

Index