Package 'EmissV'

June 19, 2018

Title Vehicular Emissions by Top-Down Methods

Version 0.664.7

Maintainer Daniel Schuch <schuch@usp.br>

Description Creates emissions for use in air quality models. Vehicular emissions are estimated by a top-down approach, total emissions are calculated using the statistical description of the fleet of vehicles, the emission is spatially distributed according to satellite images or openstreetmap data <https://www.openstreetmap.org> and then distributed temporarily (Vara-Vela et al., 2016, <doi:10.5194/acp-16-777-2016>).

Depends R (>= 3.4)

Imports ncdf4, units(>= 0.5-1), raster, sp, sf, methods, data.table,lwgeom

Suggests testthat,covr,osmar,rgdal

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

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

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

NeedsCompilation no

Author Daniel Schuch [aut, cre] (<https://orcid.org/0000-0001-5977-4519>), Sergio Ibarra-Espinosa [aut] (<https://orcid.org/0000-0002-3162-1905>)

Repository CRAN

Date/Publication 2018-06-19 04:48:13 UTC

R topics documented:

`areaSource` *Distribution of emissions by area*

Description

Calculate the spatial distribution by a raster kasked by shape/model grid information.

Usage

```
areaSource(s, r, grid = NA, name = "", as_frac = F, verbose = T)
```

Arguments

`s`	input shape object
`r`	input raster object
`grid`	grid with the output format
`name`	area name
`as_frac`	return a fraction instead of a raster
`verbose`	display additional data

Format

a raster

Details

About the DMSP and example data https://en.wikipedia.org/wiki/Defense_Meteorological_Satellite_Program

Source

Data avaliable http://www.ospo.noaa.gov/Operations/DMSP/index.html

Examples

```
shape  <- raster::shapefile(paste(system.file("extdata", package = "EmissV"),
                            "/BR.shp",sep=""),verbose = FALSE)
shape  <- shape[22,1] # subset for Sao Paulo - BR
raster <- raster::raster(paste(system.file("extdata", package = "EmissV"),
                         "/dmsp.tiff",sep=""))
grid  <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d02",sep=""))
SP     <- areaSource(shape,raster,grid,name = "SPMA")

sp::spplot(SP,scales = list(draw=TRUE),ylab="Lat",xlab="Lon",
           main=list(label="Spatial Distribution by Lights for Sao Paulo - Brazil"),
           col.regions = c("#031638","#001E48","#002756","#003062",
                           "#003A6E","#004579","#005084","#005C8E",
                           "#006897","#0074A1","#0081AA","#008FB3",
                           "#009EBD","#00AFC8","#00C2D6","#00E3F0"))
```

`emission` *Emissions in the format for atmospheric models*

Description

Combine area sources and total emissions to model output

Usage

```
emission(total, pol, area, grid, inventory = NULL, mm = 1, aerosol = F,
  plot = F, verbose = T)
```

Arguments

`total`	list of total emission
`pol`	pollutant name
`area`	list of area sources or matrix with a spatial distribution
`grid`	grid information
`inventory`	a inventory raster from read
`mm`	pollutant molar mass
`aerosol`	TRUE for aerosols and FALSE (defoult) for gazes
`plot`	TRUE for plot the final emissions
`verbose`	display additional information

Format

matrix of emission

Note

if Inventory is provided, the firsts tree arguments are not be used by the funciton.

Is a good practice use the set_units(fe,your_unity), where fe is your emission factory and your_unity is usually g/km on your emission factory

the list of area must be in the same order as defined in vehicles and total emission.

just WRF-Chem is suported by now

See Also

totalEmission and areaSource

Examples

```
veiculos <- vehicles(example = TRUE)

EmissionFactors <- emissionFactor(example = TRUE)

TOTAL  <- totalEmission(veiculos,EmissionFactors,pol = c("CO"),verbose = TRUE)

grid   <- gridInfo(paste0(system.file("extdata", package = "EmissV"),"/wrfinput_d01"))
shape  <- raster::shapefile(paste0(system.file("extdata", package = "EmissV"),"/BR.shp"))
raster <- raster::raster(paste0(system.file("extdata", package = "EmissV"),"/dmsp.tiff"))

SP     <- areaSource(shape[22,1],raster,grid,name = "SP")
RJ     <- areaSource(shape[17,1],raster,grid,name = "RJ")

e_CO   <- emission(TOTAL,"CO",list(SP = SP, RJ = RJ),grid,mm=28)
```

---

emissionFactor *Tool to set-up emission factors*

Description

Return a data frame with vehicle information. Types argument defines the diary use:

Usage

```
emissionFactor(ef, poluttant = names(ef), vnames = NA, unit = "g/km",
  example = F, verbose = T)
```

Arguments

ef	list with emission factors
poluttant	poluttant names
vnames	name of each vehicle categoy (optional)

unit	tring with unit from unit package, for default is "g/km"
example	TRUE to diaplay a simple example
verbose	display additional information

Format

data frame

See Also

areaSource and totalEmission

Examples

```
EF <- emissionFactor(example = TRUE)

# or the code for the same result
EF <- emissionFactor(ef = list(CO = c(1.75,10.04,0.39,0.45,0.77,1.48,1.61,0.75),
                               PM = c(0.0013,0.0,0.0010,0.0612,0.1052,0.1693,0.0,0.0)),
                     vnames = c("Light Duty Vehicles Gasohol","Light Duty Vehicles Ethanol",
                            "Light Duty Vehicles Flex","Diesel Trucks","Diesel Urban Busses",
                                  "Diesel Intercity Busses","Gasohol Motorcycles",
                                  "Flex Motorcycles"))
```

---

gridInfo *Read grid information from a NetCDF file*

Description

Return a list containing information of a regular grid / domain

Usage

```
gridInfo(file = file.choose(), z = F, verbose = T)
```

Arguments

file	file name/path to a wrfinput of wrfchemi file
z	TRUE for read wrfinput vertical coordinades
verbose	display additional information

Note

just WRF-Chem is suported by now

Examples

```
grid_d1 <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d01",sep=""))

grid_d2 <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d02",sep=""))
grid_d3 <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d03",sep=""))
names(grid_d1)
# for plot the shapes
library(sp)
shape  <- raster::shapefile(paste0(system.file("extdata", package = "EmissV"),"/BR.shp"))
plot(shape,xlim = c(-55,-40),ylim = c(-30,-15), main="3 nested domains")
axis(1); axis(2); box(); grid()
lines(grid_d1$Box, col = "red")
text(grid_d1$xlim[2],grid_d1$Ylim[1],"d1",pos=4, offset = 0.5)
lines(grid_d2$Box, col = "red")
text(grid_d2$xlim[2],grid_d2$Ylim[1],"d2",pos=4, offset = 0.5)
lines(grid_d3$Box, col = "red")
text(grid_d3$xlim[1],grid_d3$Ylim[2],"d3",pos=2, offset = 0.0)
```

---

lineSource *Distribution of emissions by streets*

Description

Create a distribution from sp spatial lines data frame or spatial lines.

Use "wrfinput" for a gridInfo from an output from real.exe and "geo" for a output from geog.exe

Usage

```
lineSource(s, grid, as_raster = F, verbose = T, type = "wrfinput")
```

Arguments

s	SpatialLinesDataFrame of SpatialLines object
grid	grid object with the grid information
as_raster	output format, TRUE for raster, FALSE for matrix
verbose	display additional information
type	"wrfinput" or "geo" for grid type

Source

OpenstreetMap data avaliable https://www.openstreetmap.org/ and https://download.geofabrik.de/

See Also

gridInfo and rasterSource

Examples

```
roads <- osmar::get_osm(osmar::complete_file(),
                        source = osmar::osmsource_file(paste(system.file("extdata",
                        package="EmissV"),"/streets.osm.xz",sep="")))
road_lines <- osmar::as_sp(roads,what = "lines")
roads <- sf::st_as_sf(road_lines)

d3    <- gridInfo(paste0(system.file("extdata", package = "EmissV"),"/wrfinput_d03"))

roadLength <- lineSource(roads,d3,as_raster=TRUE)
sp::spplot(roadLength, scales = list(draw=TRUE), ylab="Lat", xlab="Lon",main="Length of roads",
           sp.layout=list("sp.lines", road_lines))
```

`perfil` *Temporal profile for veicular emissions*

Description

set of houtly profiles that represent the mean activity for each day of the week. These profiles comes from traffic counts of toll stations located in São Paulo city, for summer and winters of 2012, 2013 and 2014.

Usage

```
data(perfil)
```

Format

A list of data frames by hour and weekday.

Note

The profile is normalized by days (but is balanced for a complete week) so diary emission x profile = hourly emission.

Examples

```
# load the data
data(perfil)

# function to simple view
plot.perfil <- function(per = perfil$LDV, text="", color = "#0000FFBB"){
  plot(per[,1],ty = "l", ylim = range(per),axe = FALSE,
       xlab = "hour",ylab = "Intensity",main = text,col=color)
  for(i in 2:7){
```

```
    lines(per[,i],col = color)
  }
  for(i in 1:7){
    points(per[,i],col = "black", pch = 20)
  }
  axis(1,at=0.5+c(0,6,12,18,24),labels = c("00:00","06:00","12:00","18:00","00:00"))
  axis(2)
  box()
}

# view all profiles in perfil data
for(i in 1:length(names(perfil))){
  print(paste("profile",i,names(perfil)[i]))
  plot.perfil(perfil[[i]],names(perfil)[i])
}
```

`plumeRise` *Calculate plume rise height.*

Description

Calculate the maximum height of rise based on Brigs (1975), the height is calculated using different formulations depending on stability and wind conditions.

Usage

```
plumeRise(df, imax = 10, ermax = 1/100, Hmax = T, verbose = T)
```

Arguments

`df`	data.frame with micrometeorological and emission data
`imax`	maximum number of iteractions
`ermax`	maximum error
`Hmax`	use weil limit for plume rise, see details
`verbose`	display additional information

Format

data.frame with the input, rise (m) and effective higt (m)

Details

The input data.frame must contains the folloging colluns:

- z: height of the emission (m)

- r: source raius (m)

- Ve: emission velocity (m/s)

- Te: emission temperature (K)

- ws: wind speed (m/s)

- Temp: ambient temperature (K)

- h: height of the Atmospheric Boundary Layer-ABL (m)

- L: Monin-Obuhkov Lench (m)

- dtdz: lapse ration of potential temperature, used only for stable ABL (K/m)

- Ustar: atriction velocity, used only for neutral ABL (m/s)

- Wstar: scale of convectie velocity, used only for convective ABL (m/s)

Addcitionaly some combination of ws, Wstar and Ustar can produce inacurate results, Weil (1979) propose a geometric limit of 0.62 * (h - Hs) for the rise value.

References

The plume rise formulas are from Brigs (1975):"Brigs, G. A. Plume rise predictions, Lectures on Air Pollution and Environmental Impact Analyses. Amer. Meteor. Soc. p. 59-111, 1975." and Arya 1999: "Arya, S.P., 1999, Air Pollution Meteorology and Dispersion, Oxford University Press, New York, 310 p."

The limits are from Weil (1979): "WEIL, J.C. Assessmet of plume rise and dispersion models using LIDAR data, PPSP-MP-24. Prepared by Environmental Center, Martin Marietta Corporation, for Maryland Department of Natural Resources. 1979."

The example is data from a chimney of the Candiota thermoelectric powerplant from Arabage et al (2006) "Arabage, M. C.; Degrazia, G. A.; Moraes O. L. Simulação euleriana da dispersão local da pluma de poluente atmosférico de Candiota-RS. Revista Brasileira de Meteorologia, v.21, n.2, p. 153-160, 2006."

Examples

```
candiota <- matrix(c(150,1,20,420,3.11,273.15 + 3.16,200,-34.86,3.11,0.33,
                     150,1,20,420,3.81,273.15 + 4.69,300,-34.83,3.81,0.40,
                     150,1,20,420,3.23,273.15 + 5.53,400,-24.43,3.23,0.48,
                     150,1,20,420,3.47,273.15 + 6.41,500,-15.15,3.48,0.52,
                     150,1,20,420,3.37,273.15 + 6.35,600, -8.85,3.37,2.30,
                     150,1,20,420,3.69,273.15 + 5.93,800,-10.08,3.69,2.80,
                     150,1,20,420,3.59,273.15 + 6.08,800, -7.23,3.49,1.57,
                     150,1,20,420,4.14,273.15 + 6.53,900,-28.12,4.14,0.97),
                     ncol = 10, byrow = TRUE)
candiota <- data.frame(candiota)
names(candiota) <- c("z","r","Ve","Te","ws","Temp","h","L","Ustar","Wstar")
row.names(candiota) <- c("08:00","09:00",paste(10:15,":00",sep=""))
candiota <- plumeRise(candiota,Hmax = TRUE)
```

```
print(candiota)
```

`pointSource` *Emissions from point sources*

Description

Transform a set of points into a grinded output

Usage

```
pointSource(emissions, grid, verbose = T)
```

Arguments

`emissions`	list of points
`grid`	grid object with the grid information
`verbose`	display additional information

Value

a raster

See Also

`gridInfo` and `rasterSource`

Examples

```
d1 <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d01",sep=""))

p = data.frame(lat      = c(-22,-22,-23.5),
               lon      = c(-46,-48,-47  ),
               z        = c(0  ,  0,  0  ),
               emission = c(666,444,111  ) )

p_emissions <- pointSource(emissions = p, grid = d1)

sp::spplot(p_emissions,scales = list(draw=TRUE), ylab="Lat", xlab="Lon",
           main = "3 point sources for domain d1")
```

rasterSource *Distribution of emissions by a georeferenced image*

Description

Calculate the spatial distribution by a raster

Usage

```
rasterSource(r, grid, nlevels = "all", verbose = T)
```

Arguments

r	input raster object
grid	grid object with the grid information
nlevels	number of vertical levels off the emission array
verbose	display additional information

Details

About the DMSP and example data https://en.wikipedia.org/wiki/Defense_Meteorological_Satellite_Program

Value

Returns a matrix

Source

Exemple data is a low resolution cutting from image of persistent lights of the Defense Meteorological Satellite Program (DMSP) https://pt.wikipedia.org/wiki/Defense_Meteorological_Satellite_Program

Data avaliable http://www.ospo.noaa.gov/Operations/DMSP/index.html

See Also

gridInfo and lineSource

Examples

```
grid  <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d01",sep=""))
x    <- raster::raster(paste(system.file("extdata", package = "EmissV"),"/dmsp.tiff",sep=""))
test  <- rasterSource(x,grid)

image(test,axe = F, main = "Spatial distribution by Persistent Nocturnal Lights from DMSP")
```

read *Read NetCDF data from global inventaries*

Description

Read data from global inventoris

Usage

```
read(file, version = "EDGAR 4.3.1 v2", as_raster = T, verbose = T)
```

Arguments

file	file name or names (variables are summed)
version	inventorie information
as_raster	return a raster (defoult) or matrix (with units)
verbose	display additional information

Value

Matrix or raster

Source

read abbout EDGAR at http://edgar.jrc.ec.europa.eu

See Also

rasterSource and gridInfo

Examples

```
d1    <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d01",sep=""))
d2    <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d02",sep=""))
print("download and untar EDGAR data from:")
print("http://edgar.jrc.ec.europa.eu/gallery.php?release=v431_v2&substance=NOx&sector=TRO")
nox    <- read("v431_v2_REFERENCE_NOx_2010_10_TRO.0.1x0.1.nc")
sp::spplot(nox, scales = list(draw=TRUE), xlab="Lat", ylab="Lon",main="NOx emissions from EDGAR")
nox_d1 <- rasterSource(nox,d1)
nox_d2 <- rasterSource(nox,d2)
image(nox_d1, main = "NOx emissions from transport from EDGAR 3.4.1 for d1")
image(nox_d2, main = "NOx emissions from transport from EDGAR 3.4.1 for d2")
```

`streetDist` *Distribution by OpenStreetnMap street*

Description

Distribute emissions by streets of OpenStreetMap

Usage

```
streetDist(emission = 1, dist = c(1, 0, 0, 0, 0), grid = NULL,
  osm = NULL, epsg = 31983)
```

Arguments

`emission`	numeric of emissions
`dist`	numeric vector with length 5. The order represents motorway, trunk, primary, secondary and tertiary
`grid`	grid of polygons class sf
`osm`	streets of OpenStreetMaps class sf
`epsg`	spatial code for projecting spatial data

Value

grid of polygon

Examples

```
## Not run:
# Do not run
library(sf)
# Download OSM streets
streets <- st_read("path")
streets <- streets[streets$highway != "residential", ]
# Grid
grid  <- gridInfo(paste(system.file("extdata", package = "EmissV"),"/wrfinput_d02",sep=""))
names(grid)
d3 <- data.frame(x = as.numeric(grid$Lon),
                 y = as.numeric(grid$Lat))
d3 <- st_as_sf(d3, coords = c("x","y"))
st_crs(d3) <- st_crs(4326)
library(vein)
g <- st_transform(st_as_sf(vein::make_grid(as(st_transform(d3, 31983),
                 "Spatial"),
               grid$DX*1000, grid$DX*1000, T)), 4326)
streets$id <- NULL
per <- c(1, 0, 0, 0, 0)
teste <- streetDist(emission = 1000000, dist = per, grid = g,
                    osm = streets, epsg = 31983)
```

```
## End(Not run)
```

`totalEmission` *Calculate total emissions*

Description

Caculate the total emission with:

Emission(pollutant) = sum(Vehicles(n) * Km_day_use(n) * Emission_Factor(n,pollutant))

where n is the type of the veicle

Usage

```
totalEmission(v, ef, pol, verbose = T)
```

Arguments

`v`	dataframe with the vehicle data
`ef`	emission factor
`pol`	pollutant name in ef
`verbose`	display additional information

Format

Return a list with the daily total emission by interest area (cityes, states, countries, etc).

Note

the units (set_units("value",unit) where the recomended unit is g/d) must be used to make the ef data.frame

See Also

`rasterSource`, `lineSource` and `emission`

Examples

```
veic <- vehicles(example = TRUE)

EmissionFactors <- emissionFactor(example = TRUE)

TOTAL <- totalEmission(veic,EmissionFactors,pol = c("CO","PM"))
```

totalVOC *Calculate Total VOCs emissions*

Description

Calculate Volatile Organic Compounds (COVs) emited by the process of exhaust (through the exhaust pipe), liquid (carter and evaporative) and vapor (fuel transfer operations).

Avaliable COVs are: eth, hc3, hc5, hc8, ol2, olt, oli, iso, tol, xyl, ket, ch3oh and ald

Usage

```
totalVOC(v, ef, pol, verbose = T)
```

Arguments

v	data frame with the vehicle data
ef	emission factors
pol	pollutant name in ef
verbose	display additional information

Format

Return a list with the daily total emission by territory.

Note

The same ef can be used to totalEmission and voc

See Also

totalEmission and vehicles

Examples

```
veic <- vehicles(example = TRUE)

COV = c("eth","hc3","hc5","hc8","ol2","olt","oli","iso","tol","xyl","ket","ch3oh","ald")
EF_COV <- as.data.frame(matrix(NA,ncol = 9,nrow = 8,byrow = TRUE),
                        row.names = row.names(veic))
names(EF_COV) <-  c("vap_g","vap_e","vap_d",
                    "liq_g","liq_e","liq_d",
                    "exa_g","exa_e","exa_d")

EF_COV["vap_g"]  <- c(0.230,0.00,0.120,0.00,0.00,0.00,0.00,0.00,0.00)
EF_COV["vap_e"]  <- c(0.000,0.25,0.120,0.00,0.00,0.00,0.00,0.00,0.00)
EF_COV["vap_d"]  <- c(0.000,0.00,0.000,0.00,0.00,0.00,0.00,0.00,0.00)
EF_COV["liq_g"]  <- c(2.000,0.00,0.875,0.00,0.00,0.00,1.20,0.00)
EF_COV["liq_e"]  <- c(0.000,1.50,0.875,0.00,0.00,0.00,0.00,1.20)
```

```
EF_COV["liq_d"]  <- c(0.000,0.00,0.000,0.00,0.00,0.00,0.00,0.00)
EF_COV["exa_g"]  <- c(0.425,0.00,0.217,0.00,0.00,0.00,1.08,0.00)
EF_COV["exa_e"]  <- c(0.000,1.30,0.217,0.00,0.00,0.00,0.00,1.08)
EF_COV["exa_d"]  <- c(0.000,0.00,0.000,2.05,0.00,0.00,0.00,0.00)

print(EF_COV)

COV_total <- totalVOC(veic,EF_COV,pol = COV[10])
```

`vehicles` *Tool to set-up vehicle data table*

Description

Return a data frame with 4 columns (vehicle category, type, fuel and avarage kilometers driven) and an aditional column with the number of vehicles for each interest area (cityes, states, countries, etc).

Average daily kilometres driven are defined by vehicle type:

- LDV (Light duty Vehicles) 41 km / day

- TRUCKS (Trucks) 110 km / day

- BUS (Busses) 165 km / day

- MOTO (motorcycles and other vehicles) 140 km / day

The number of vehicles are defined by the distribution of vehicles by vehicle classs and the total number of vehicles by area.

Usage

```
vehicles(total_v, area_name = names(total_v), distribution, type,
  category = NA, fuel = NA, vnames = NA, example = F, verbose = T)
```

Arguments

`total_v`	total of vehicles by area (area length)
`area_name`	area names (area length)
`distribution`	distribution of vehicles by vehicle class
`type`	type of vehicle by vehicle class (distribution length)
`category`	category name (distribution length / NA)
`fuel`	fuel type by vehicle class (distribution length / NA)
`vnames`	name of each vehicle class (distribution length / NA)
`example`	a simple example
`verbose`	display additional information

Note

total_v and area_name must have the same length.

distribution, type, category (if used), fuel (if used) and vnames (if used) must have the same length.

See Also

areaSource and totalEmission

Examples

```
veiculos <- vehicles(example = TRUE)

# or the code bellow for the same result
# DETRAN 2016 data for total number of vehicles for 5 Brazilian states (Sao Paulo,
# Rio de Janeiro, Minas Gerais, Parana and Santa Catarina)
# vahicle distribution of Sao Paulo

veiculos <- vehicles(total_v = c(27332101, 6377484, 10277988, 7140439, 4772160),
                     area_name = c("SP", "RJ", "MG", "PR", "SC"),
                     distribution = c( 0.4253, 0.0320, 0.3602, 0.0260,
                                       0.0290, 0.0008, 0.1181, 0.0086),
                     category =  c("LDV_E25","LDV_E100","LDV_F","TRUCKS_B5",
                                   "CBUS_B5","MBUS_B5","MOTO_E25","MOTO_F"),
                     type = c("LDV", "LDV", "LDV","TRUCKS",
                              "BUS","BUS","MOTO", "MOTO"),
                     fuel = c("E25", "E100", "FLEX","B5",
                              "B5","B5","E25", "FLEX"),
                   vnames = c("Light duty Vehicles Gasohol","Light Duty Vehicles Ethanol",
                          "Light Duty Vehicles Flex","Diesel trucks","Diesel urban busses",
                                "Diesel intercity busses","Gasohol motorcycles",
                                "Flex motorcycles"))
```

Index