Package 'eixport'

June 25, 2020

Title Export Emissions to Atmospheric Models

Version 0.4.7

Date 2020-06-24

Description Emissions are the mass of pollutants released into the atmosphere. Air quality models need emissions data, with spatial and temporal distribution, to represent air pollutant concentrations. This package, eixport, creates inputs for the air quality models 'WRF-Chem' Grell et al (2005) <doi:10.1016/j.atmosenv.2005.04.027>, 'BRAMS-SPM' Freitas et al (2005) <doi:10.1016/j.atmosenv.2005.07.017> and 'RLINE' Snyder et al (2013) <doi:10.1016/j.atmosenv.2013.05.074>. See the eixport website (<https://atmoschem.github.io/eixport/>) for more information, documentations and examples. More details in Ibarra-Espinosa et al (2018) <doi.org/10.21105/joss.00607>.

License MIT + file LICENSE

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

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

Depends R (>= 3.5.0)

Imports sf, ncdf4, raster, sp, methods, cptcity, utils, tidyr, silicate (>= 0.3), sfheaders (>= 0.2.1)

Encoding UTF-8

LazyData no

RoxygenNote 7.0.2

Suggests testthat, covr, lwgeom

NeedsCompilation no

Author Sergio Ibarra-Espinosa [aut, cre] (<https://orcid.org/0000-0002-3162-1905>), Daniel Schuch [ctb] (<https://orcid.org/0000-0001-5977-4519>), Edmilson Freitas [ctb] (<https://orcid.org/0000-0001-8783-2747>)

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

Repository CRAN

Date/Publication 2020-06-25 05:40:02 UTC

R topics documented:

`emisco` *Emissions from VEIN demo*

Description

Emissions with units for R-LINE

Usage

```
data(emisco)
```

Format

A `sf` object of type LINESTRING with 288 rows and 15 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)

V8 Emissions (g/s)

xmin Initial x coordinates

xmax Ending x coordinates

ymin Initial y coordinates

ymax Ending y coordinates

geometry geometry column of the sf object data(emisco)

Source

https://github.com/atmoschem/vein

`emis_opt` *List of WRF emission species*

Description

Emission package definitions from WRF 4.0.1, for use in wrf_create function.

Usage

```
data(emis_opt)
```

Format

A list of emision variables names, same number as emis_opt in namelist.

Note

look to the number of aerosol of the emis_opt in WRF domumentation / code.

Author(s)

Daniel Schuch

Source

https://github.com/wrf-model/WRF

See Also

`wrf_create`

Examples

```
data(emis_opt)
names(emis_opt)
emis_opt[["eradm"]]
```

---

gCO *Gridded emissions from VEIN demo*

Description

Emissions in g/h for morning rush hour.

Usage

```
data(gCO)
```

Format

A sf object of lines with 437 rows and 2 variables:

V9 Emissions of CO (g/h) for 08:00-09:00

geometry geometry data(gCO)

Source

https://github.com/atmoschem/vein

---

get_edgar *Download datasets of EDGAR emissions*

Description

The Emissions Database for Global Atmospheric Research (EDGAR) is a project from the Joint Research Centre. They provide provides global past and present day anthropogenic emissions of greenhouse gases and air pollutants by country and on spatial grid. get_edgar provide functions to download any of the EDGAR datasets.

Usage

```
get_edgar(
  dataset = "v432_AP",
  pol,
  sector,
  year,
  destpath = tempdir(),
  txt = TRUE,
  return_url = TRUE,
  copyright = TRUE
)
```

Arguments

dataset	Character; name of the datasets: "v432", "v432_AP", "v432_VOC_spec", "v4tox2", "htap_v2_2"
pol	Character; one of the pollutants shown on note.
sector	Character; one of the sectors shown on note.
year	Integer; on of the years shown on note.
destpath	Character: Path to create the directory for downloads datasets
txt	Logical; if TRUE, download data as .txt and untis t/year, if FALSE units are ug/m2/s
return_url	Logical; return url?
copyright	Logical; to show copyright information.

Value

Downloads data

Note

I recommend 2 ways:

1. include 'sector' and dont include 'pol', which download all pollutants as default

get_edgar(dataset = "v432_AP", destpath = tempdir(), sector = c("TRO", "TOTALS"), year = 2012)

2. include 'pol' and dont include 'sector', which download all sectors as default

get_edgar(dataset = "v432_AP", destpath = tempdir(), pol = c("CO", "NOx"), year = 2012)

dataset	pollutant
v432	CH4, N2O, CO2_excl_short-cycle_org_C, CO2_org_short-cycle_C
v432_AP	BC, CO, NH3, NMVOC, NOx, OC, PM10, PM2.5_bio, PM2.5_fossil, SO2
v432_VOC_spec	voc1, voc2, voc3, voc3, voc5, voc6, voc7, voc8, bvoc9, voc10. voc11, voc12, voc13, voc14, voc15, voc16,
v4tox2	HG, HG_D, HG_G, HG_P
htap_v2_2	BC, CO, NH3, NMVOC, NOx, OCm, PM10, PM2.5, SO2

voc11 only 2008

References

- **v432**: Muntean, M., Guizzardi, D., Schaaf, E., Crippa, M., Solazzo, E., Olivier, J.G.J., Vignati, E. Fossil CO2 emissions of all world countries - 2018 Report, EUR 29433 EN, Publications Office of the European Union, Luxembourg, 2018, ISBN 978-92-79-97240-9, doi:10.2760/30158, JRC113738.
- **v432_AP**:Crippa, M., Guizzardi, D., Muntean, M., Schaaf, E., Dentener, F., van Aardenne, J. A., Monni, S., Doering, U., Olivier, J. G. J., Pagliari, V., and Janssens-Maenhout, G.: Gridded emissions of air pollutants for the period 1970–2012 within EDGAR v4.3.2, Earth Syst. Sci. Data, 10, 1987–2013, https://doi.org/10.5194/essd-10-1987-2018, 2018.
- **v432_VOC_spec**: Huang, G., Brook, R., Crippa, M., Janssens-Maenhout, G., Schieberle, C., Dore, C., Guizzardi, D., Muntean, M., Schaaf, E., and Friedrich, R.: Speciation of anthropogenic emissions of non-methane volatile organic compounds: a global gridded data set for 1970–2012, Atmos. Chem. Phys., 17, 7683-7701, https://doi.org/10.5194/acp-17-7683-2017, 2017.
- **v4tox2**: Muntean, M., Janssens-Maenhout, G., Song, S., Giang, A., Selin, N. E., Zhong, H., ... & Schaaf, E. (2018). Evaluating EDGARv4. tox2 speciated mercury emissions ex-post scenarios and their impacts on modelled global and regional wet deposition patterns. Atmospheric Environment, 184, 56-68.
- **htap_v2_2**: Janssens-Maenhout, G., Crippa, M., Guizzardi, D., Dentener, F., Muntean, M., Pouliot, G., Keating, T., Zhang, Q., Kurokawa, J., Wankmüller, R., Denier van der Gon, H., Kuenen, J. J. P., Klimont, Z., Frost, G., Darras, S., Koffi, B., and Li, M.: HTAP_v2.2: a mosaic of regional and global emission grid maps for 2008 and 2010 to study hemispheric transport of air pollution, Atmos. Chem. Phys., 15, 11411–11432, https://doi.org/10.5194/acp-15-11411-2015, 2015.

MNM is MNN for NOx v432_AP

Examples

```
## Not run:
# Download all pollutants for sector TRO
get_edgar(dataset = "v432_AP", destpath = tempdir(),
sector = c("TOTALS"),
year = 2012)
# Download all sectors for pollutant CO
get_edgar(dataset = "v432_AP", destpath = tempdir(),
pol = c("CO"),
year = 2012)

## End(Not run)
```

`Lights` *Spatial distribution example*

Description

Spatial distribution for vehicular emissions based on an image of persistent lights of the Defense Meteorological Satellite Program (DMSP) for 5 Brazilian states (Sao Paulo, Rio de Janeiro, Mato Grosso, and Santa Catarina e Parana).

Usage

```
data(Lights)
```

Format

A matrix of spatial distribution

Details

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

Author(s)

Daniel Schuch

Source

https://ngdc.noaa.gov/eog/dmsp/downloadV4composites.html

See Also

to_wrf

Examples

```
## Not run:

dir.create(file.path(tempdir(), "EMISS"))
wrf_create(wrfinput_dir = system.file("extdata", package = "eixport"),
          wrfchemi_dir = file.path(tempdir(), "EMISS"),
          frames_per_auxinput5 = 24)

# get the name of created file
files <- list.files(path = file.path(tempdir(), "EMISS"),
                    pattern = "wrfchemi",
                    full.names = TRUE)

data(Lights)

perfil <- c(0.010760058, 0.005280596, 0.002883553, 0.002666932,
           0.005781312, 0.018412838, 0.051900411, 0.077834636,
           0.067919758, 0.060831614, 0.055852868, 0.052468599,
           0.050938043, 0.051921718, 0.052756244, 0.052820165,
           0.058388406, 0.072855890, 0.075267137, 0.063246412,
           0.042713523, 0.029108975, 0.022091855, 0.015298458)

plot(perfil, ty = "l", col= "purple", xlab = "Hour", main = "Time profile",
    ylab = "Weight", axes = FALSE, xlim = c(0, 24))
axis(2)
axis(1, at = c(0, 6, 12, 18, 24),
    labels = c("00:00","06:00","12:00","18:00","00:00"))
```

```
to_wrf(Lights, files[1], total = 1521983, profile = perfil, name = "E_CO")

## End(Not run)
```

`rawprofile` *Raw profile*

Description

Raw profile

Usage

```
data(rawprofile)
```

Format

A matrix with 1 column and 168 rows
data(rawprofile)

`sfx_explode` *splits line by vertex*

Description

`sfx_explode` splits line by vertex

Usage

```
sfx_explode(x, ...)
```

Arguments

`x`	spatial lines.
`...`	ignored

Value

spatial lines

Author(s)

Michael Summer githubdot/hypertidy/silicate/issues/102

Examples

```
{
# Do not run
data(emisco)
emisco <- emisco[1:100, "V8"]
dfco <- sfx_explode(emisco)
etm <- to_munich(sdf = dfco)
names(etm)
class(etm)
head(etm$Emissions)
head(etm$Street)
write.table(x = etm$Emissions, file = paste0(tempfile(), "_Emissions.txt"),
row.names = FALSE, sep = " ", quote = FALSE)
write.table(x = etm$Street, file = paste0(tempfile(), "_Street.txt"),
row.names = FALSE, sep = " ", quote = FALSE)
}
```

---

st_explode *Split line by vertex (experimental)*

Description

st_explode split a lines data.frame into each vertex. It to mimic the function explode from qgis, that the reason for the name st_explode

Usage

```
st_explode(net)
```

Arguments

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

Note

All variables are transformed into numeric.

Thanks Michael Summer (mdsummer) for the function sfx_explode!

Examples

```
## Not run:
# do not run
library(vein)
data(net)
net2 <- st_explode(net)
dim(net)
dim(net2)

## End(Not run)
```

`to_as4wrf` *Generates emissions dataframe to generate WRF-Chem inputs*

Description

`to_as4wrf` 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: "Another Asimilation System 4 WRF (AAS4WRF)" as published by Vera-Vala et al (2016)

Usage

```
to_as4wrf(sdf, nr = 1, dmyhm, tz, crs = 4326, islist)
```

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. When there are no emissions available, the SpatialPolygonsDataFrame must contain 0.
`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=crs:4326". Used to transform the coordinates of the output
`islist`	logical value to indicate if sdf is a list or not

Value

data-frame of gridded emissions g/h

Note

The user must produce a text file with the data-frame resulting of this function. Then, use this file with the NCL script AAS4WRF.ncl

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

wrf_create to_wrf

Examples

```
{
data(gCO)
df <- to_as4wrf(sdf = gCO, dmyhm = "29-04-2018 00:00",
                tz = "America/Sao_Paulo")
head(df)
df2 <- to_as4wrf(sdf = list(co = gCO, pm = gCO), dmyhm = "29-04-2018 00:00",
                tz = "America/Sao_Paulo")
head(df2)
}
```

---

to_brams_spm *Inputs for BRAMS-SPM*

Description

Create inputs for BRAMS-SPM. The inputs consist of a data-frame or a list of data-frames with daily emissions (mol/day), lat, long. Also, including a functions describing the hourly profile.

Usage

```
to_brams_spm(sdf, epsg = 4326)
```

Arguments

sdf	Grid emissions, which can be a SpatialPolygonsDataFrame or polygon grid class sf' including the hourly emissions in mol/h for 24 hours. The object can also be a list of objects SpatialPolygonsDataFrame or Spatial Features polygon grid class 'sf'.
epsg	Coordinate reference system, e.g: "4326". Used to transform the coordinates of the output.

Value

data-frame of daily gridded emissions, lat, long and a message with function.

Note

When the input is class 'Spatial', they are converted to 'sf'. If the input is a data-frame, the output is a data-frame. If the input is a list, the output is a list.

Author(s)

Sergio Ibarra and Edmilson Freitas

References

SPM BRAMS: FREITAS, E. MARTINS, L., SILVA, P. and ANDRADE, M. A simple photochemical module implemented in rams for tropospheric ozone concentration forecast in the metropolitan area of são paulo, brazil: Coupling and validation. Atmospheric Environment, Elsevier, n. 39, p. 6352–6361, 2005.

Examples

```
## Not run:
data(gCO)
df1 <- to_brams_spm(sdf = gCO, epsg = 4326)
head(df1)
df2 <- to_brams_spm(sdf = list(co = gCO, pm = gCO), epsg = 4326)
lapply(df2, head)

## End(Not run)
```

---

to_munich *Export emissions to Model of Urban Network of Intersecting Canyons and Highways (MUNICH)*

Description

to_munich Export spatial emissions objects according the format required by MUNICH. This function was designed to read street emissions from VEIN by it can be used to read any other.

Usage

```
to_munich(sdf, idbrin, typo, width, height, crs = 4326)
```

Arguments

sdf	Street Emissions object class 'sf' LINESTRING or "SpatialLinesdataFrame". The columns are the emissions.
idbrin	Integer; id.
typo	Integer; id2.
width	Integer; width.
height	Integer; heigth.
crs	Numeric; Coordenade Reference System with default value of 4326.

Value

A list with a data frame with columns "i", "idbrin", "typo", "xa", "ya", "xb", "yb" and the pollutants; and another data.frame with "i", "length" (m), "width" (with value 0) and "height" (with value 0). Width and height must be obtained by the user.

Note

The user must ensure that the spatial object has one line feature per vertex and lines with more than one vertex must be previously splitted.

References

Kim, Y., Wu, Y., Seigneur, C., and Roustan, Y.: Multi-scale modeling of urban air pollution: development and application of a Street-in-Grid model (v1.0) by coupling MUNICH (v1.0) and Polair3D (v1.8.1), Geosci. Model Dev., 11, 611-629, https://doi.org/10.5194/gmd-11-611-2018, 2018.

Examples

```
{
# Do not run
data(emisco)
dfco <- emisco[1:1000,"V8"]
etm <- to_munich(sdf = dfco)
names(etm)
class(etm)
head(etm$Emissions)
head(etm$Street)
write.table(x = etm$Emissions, file = paste0(tempfile(), "_Emissions.txt"),
row.names = FALSE, sep = " ", quote = FALSE)
write.table(x = etm$Street, file = paste0(tempfile(), "_Street.txt"),
row.names = FALSE, sep = " ", quote = FALSE)
}
```

---

to_rline *Export emissions to other formats*

Description

Export emissions object according to format of file 'Sources.txt' of the model R-LINE

Usage

```
to_rline(
  X_b,
  Y_b,
  Z_b,
  X_e,
  Y_e,
  Z_e,
  dCL,
  sigmaz0,
  lanes,
  Emis,
  Hw1,
```

```
  dw1,
  Hw2,
  dw2,
  Depth,
  Wtop,
  Wbottom,
  experimental = FALSE
)
```

Arguments

X_b	initial x coordinates.
Y_b	initial y coordinates.
Z_b	initial meters above sea level (m).
X_e	final x coordinates.
Y_e	final y coordinates.
Z_e	final meters above sea level (m).
dCL	offset distance for each source relative to the centerline.
sigmaz0	vertical dispersion (m).
lanes	number of lanes at each street.
Emis	Column with the emissions whose unit must be g/ms.
Hw1	Height of the barrier 1 (m).
dw1	Distance to barrier 1 (m).
Hw2	height of the barrier 2 (m).
dw2	Distance to barrier 2 (m).
Depth	Depth of the depression. USed for depressed roadway (m).
Wtop	width of the opening at the top of the depression (m).
Wbottom	width of the roadway at the bottom of the depression (m).
experimental	Boolean argument to denote the use of the experimental features (TRUE) or not (FALSE).

Value

Data frame with format for R-LINE model.

Note

Michelle G. Snyder, Akula Venkatram, David K. Heist, Steven G. Perry, William B. Petersen, Vlad Isakov, RLINE: A line source dispersion model for near-surface releases, In Atmospheric Environment, Volume 77, 2013, Pages 748-756, ISSN 1352-2310, https://doi.org/10.1016/j.atmosenv.2013.05.074.

Examples

```
## Not run:
# Do not run
data(emisco)
Source <- to_rline(X_b = emisco$xmin,
                Y_b = emisco$ymin,
                Z_b =0,
                X_e = emisco$xmin,
                Y_e = emisco$ymin,
                Z_e =0,
                dCL = 0,
                Emis = emisco$V8,
                sigmaz0 = 2,
                lanes = emisco$lanes)
head(Source)
write.table(x = Source, file = paste0(tempdir(), "/Sources.txt"),
row.names = FALSE, sep = " ", quote = FALSE)

## End(Not run)
```

---

to_wrf *Combine total/spatial/temporal/split and write emission to file*

Description

Function to expand, split and write emissions. The input is expanded into time by profile and split between variables with diferent weights.

Usage

```
to_wrf(
  POL,
  file = file.choose(),
  name = NA,
  total = NA,
  norm = F,
  profile = 1,
  weights = 1,
  verbose = T
)
```

Arguments

POL	matrix or array of emissions of spatial weights
file	emission file name
name	species to be write
total	total of emited species (modifier)

norm	if the spatial weights need to be normalized (modifier)
profile	temporal profile to expand the emissions (modifier)
weights	weight of each species (modifier)
verbose	display additional information

Note

length(profile) must be the number of times in the emission file (value of frames_per_auxinput5 if wrf_create() was used to create this file).

total is an additional way to calculate or correct the total emissions

sum(profile) = 1 and sum(weights) = 1 to conserve mass

names and weights must have the same length

Author(s)

Daniel Schuch

See Also

wrf_create, wrf_get,wrf_profile and wrf_plot

Examples

```
## Not run:
dir.create(file.path(tempdir(), "EMISS"))
wrf_create(wrfinput_dir = system.file("extdata", package = "eixport"),
          wrfchemi_dir = file.path(tempdir(), "EMISS"),
          frames_per_auxinput5 = 24)

# get the name of created file
files <- list.files(path = file.path(tempdir(), "EMISS"),
                    pattern = "wrfchemi",
                    full.names = TRUE)

data(Lights)

perfil <- c(0.010760058, 0.005280596, 0.002883553, 0.002666932,
            0.005781312, 0.018412838, 0.051900411, 0.077834636,
            0.067919758, 0.060831614, 0.055852868, 0.052468599,
            0.050938043, 0.051921718, 0.052756244, 0.052820165,
            0.058388406, 0.072855890, 0.075267137, 0.063246412,
            0.042713523, 0.029108975, 0.022091855, 0.015298458)

plot(perfil, ty = "l", col= "purple", xlab = "Hour", main = "Time profile",
    ylab = "Weight", axes = FALSE, xlim = c(0, 24))
axis(2)
axis(1, at = c(0, 6, 12, 18, 24),
    labels = c("00:00","06:00","12:00","18:00","00:00"))

to_wrf(Lights, files[1], total = 1521983, profile = perfil, name = "E_CO")
```

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

---

wrf_add *Function to add values for variables on emission files*

Description

Add values to a variable in a netCDF file, the main use is to combine different emissions like top-down emission (EmissV emissions) and inventary emission (sush as EDGAR, GAINS, RETRO, etc).

Usage

```
wrf_add(file = file.choose(), name = NA, POL)
```

Arguments

file	name of file interactively (default) or specified
name	name of the variable (any variable)
POL	variable to be written

Note

this function might be deprecated in future

Author(s)

Daniel Schuch

Examples

```
{
# create the folder and emission file
dir.create(file.path(tempdir(), "EMISS"))
wrf_create(wrfinput_dir = system.file("extdata", package = "eixport"),
          wrfchemi_dir = file.path(tempdir(), "EMISS"))

# get the name of created file
files <- list.files(path = file.path(tempdir(), "EMISS"),
                    pattern = "wrfchemi",
                    full.names = TRUE)

# open, put some numbers and write
CO <- wrf_get(file = files[1], name = "E_CO")
CO[] = rnorm(length(CO),mean = 5,sd = 1)
wrf_put(file = files[1], name = "E_CO", POL = CO)
# open, put some different numbers and write
CO[] = rnorm(length(CO),mean = 10,sd = 1)
```

```
wrf_add(file = files[1], name = "E_CO", POL = CO)
}
```

`wrf_create` *Create emission files for the WRF-Chem model*

Description

Create WRF-chem emission files using information from the WRF initial condictions (wrfinput) file(s). The wrfinput file of the corresponding domain is read from the current folder or from the wrfinput_dir.

There are two emission styles available: the 12 hour pair of emissions (that will be recycled by the model) using io_style_emissions = 1 and the date_hour format using io_style_emissions = 2 (default), see notes for more detail.

The initial time is the original (wrfinput file) adjusted by the day_offset argument, this argument can be useful for split the emissions into several files or for a restarted simulation. The emissions are recorded at the interval of 60 minutes (or the auxinput5_interval_m argument) for 1 time (or frames_per_auxinput5 argument times).

The variables created on output file is based on emis_opt data or a character vector contains the species, any change in variables need to be followed by a change in the n_aero for the correspondent number of aerosol species in the emission file (the n_aero last variables).

Title argument will be written on global attribute TITLE, from the version 4.0 the model checks if the TITLE version contains "V4.", this can be disabled setting 'force_use_old_data = .true.' on WRF namelist.input.

Usage

```
wrf_create(
  wrfinput_dir = getwd(),
  wrfchemi_dir = wrfinput_dir,
  domains = 1,
  frames_per_auxinput5 = 1,
  auxinput5_interval_m = 60,
  day_offset = 0,
  io_style_emissions = 2,
  kemit = 1,
  variables = "ecbmz_mosaic",
  n_aero = 14,
  COMPRESS = NA,
  force_ncdf4 = FALSE,
  title = "Anthropogenic emissions for WRF V4.0",
  separator = "default",
  verbose = FALSE
)
```

Arguments

wrfinput_dir	input folder with the wrfinput file(s)
wrfchemi_dir	output folder
domains	domain / domains to be process
frames_per_auxinput5	value from wrf &time_control namelist.input, number of times (frames) in a single emission file
auxinput5_interval_m	value from wrf &time_control namelist.input, interval in minutes between different times (frames) see Details
day_offset	number of days (can be a fraction) see Details
io_style_emissions	from wrf &chem namelist.input see Details
kemit	from wrf &chem namelist.input, number of vertical levels of the emission file
variables	emission species, can be used data(emis_opt)
n_aero	number of aerosol species
COMPRESS	integer between 1 (least compr) and 9 (most compr) or NA for no compression
force_ncdf4	force NetCDF4 format
title	TITLE attribute for the NetCDF
separator	filename alternative separator when io_style_emission=1
verbose	print file info

Note

Using io_style_emissions = 1, the wrfchemi_00z will be generated with day_offset = 0 and wrfchemi_12z with day_offset = 0.5 (frames_per_auxinput5 and auxinput5_interval_m will have no effect).

Windows users may need to rename the emission files or change in namelist the defoult filename before run wrf.exe with these emission files.

The separator argument can be useful for write in NTSF format discs on linux systems, for 'default' the separator is ':' for linux-like systems and '%3A' for windowns.

Author(s)

Daniel Schuch

See Also

to_wrf and emis_opt

Examples

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

# emissions for a 1 day forecast for domains 1 and 2

dir.create(file.path(tempdir(), "EMISS"))

# emissions on date_hour style
wrf_create(wrfinput_dir         = system.file("extdata", package = "eixport"),
           wrfchemi_dir         = file.path(tempdir(), "EMISS"),
           domains              = 1:2,
           frames_per_auxinput5 = 25,
           auxinput5_interval_m = 60,
           verbose              = TRUE)

# emissions on 00z / 12z style, create the 00z
wrf_create(wrfinput_dir         = system.file("extdata", package = "eixport"),
           wrfchemi_dir         = file.path(tempdir(), "EMISS"),
           domains              = 1:2,
           io_style_emissions   = 1,
           day_offset           = 0,
           verbose              = TRUE,
           )
# emissions on 00z / 12z style, create the 12z
wrf_create(wrfinput_dir         = system.file("extdata", package = "eixport"),
           wrfchemi_dir         = file.path(tempdir(), "EMISS"),
           domains              = 1:2,
           io_style_emissions   = 1,
           day_offset           = 0.5,
           verbose              = TRUE)

## End(Not run)
```

---

wrf_get *Function to read variables of emission files*

Description

Read a variable

Usage

```
wrf_get(
  file = file.choose(),
  name = NA,
  as_raster = FALSE,
  raster_crs = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs",
  raster_lev = 1,
```

```
  verbose = F
)
```

Arguments

file	name of file interactively (default) or specified
name	name of the variable (any variable) or time to return a POSIXlt object from model
as_raster	return a raster instead of an array
raster_crs	crs to use if as_raster is TRUE
raster_lev	level for rasters from a 4D variable
verbose	display additional information

Format

array or raster object

Author(s)

Daniel Schuch

See Also

wrf_plot and wrf_put

Examples

```
{

# create the folder and emission file
dir.create(file.path(tempdir(), "EMISS"))
wrf_create(wrfinput_dir = system.file("extdata", package = "eixport"),
         wrfchemi_dir = file.path(tempdir(), "EMISS"))

# get the name of created file
files <- list.files(path = file.path(tempdir(), "EMISS"),
                    pattern = "wrfchemi",
                    full.names = TRUE)

# open, put some numbers and write
CO <- wrf_get(file = files[1], name = "E_CO")
CO[] = rnorm(length(CO))
wrf_put(file = files[1], name = "E_CO", POL = CO)
COr <- wrf_get(file = files[1], name = "E_CO", as_raster = TRUE)

}
```

wrf_grid *Creates grid from wrf file*

Description

Return a Spatial Feature multipolygon or matrix

Usage

```
wrf_grid(filewrf, type = "wrfinput", matrix = FALSE, as_raster = FALSE)
```

Arguments

filewrf	wrf file
type	Type of wrf file: "wrfinput" or "geo". When type is "geo", lat long comes from mass grid, XLONG_M and XLAT_M
matrix	if the output is matrix or polygon (sf)
as_raster	logical, to return a raster

Note

The default crs is 4326 (see http://spatialreference.org/ref/epsg/)

Examples

```
{
# Do not run
wrf <- paste(system.file("extdata", package = "eixport"),
                         "/wrfinput_d02", sep="")
gwrf  <- wrf_grid(wrf)
plot(gwrf, axes = TRUE)
}
```

wrf_meta *Returns metadata (attributes) of wrf file in a data.frame*

Description

wrf_meta returns the attributes of a wrf NetCDF file in a data.frame. Therefore, there is no need to use ncdump -h "wrf_file"

Usage

```
wrf_meta(file = file.choose())
```

Arguments

file	Character; name of file interactively (default) or specified

Examples

```
{
# Do not run
}
```

`wrf_plot` *Simple plot from wrf emission file*

Description

Create a quick plot from wrf emission file

Usage

```
wrf_plot(
  file = file.choose(),
  name = NA,
  time = 1,
  nivel = 1,
  barra = T,
  lbarra = 0.2,
  col = cptcity::cpt(n = 13),
  verbose = T,
  ...
)
```

Arguments

file	emission file name
name	pollutant name
time	time from emission file
nivel	level from the emission file
barra	barblot if TRUE
lbarra	length of barplot
col	color vector
verbose	if TRUE print some information
...	Arguments to be passed to plot methods

Note

If the file contains levels (kemit>1), and one frame (auxinput5_interval_m = 1) time with control the level which will be ploted

In case of an error related to plot.new() margins lbarra must be adjusted

Author(s)

Daniel Schuch

See Also

Lights, to_wrf and wrf_create

Examples

```
{

dir.create(file.path(tempdir(), "EMISS"))
wrf_create(wrfinput_dir = system.file("extdata", package = "eixport"),
          wrfchemi_dir = file.path(tempdir(), "EMISS"))

# get the name of created file
files <- list.files(path = file.path(tempdir(), "EMISS"),
                    pattern = "wrfchemi",
                    full.names = TRUE)

# load end write some data in this emission file
data(Lights)
to_wrf(Lights, files[1], total = 1521983, name = "E_CO")

wrf_plot(files[1], "E_CO")
}
```

---

wrf_profile *Create a spatial profile from a wrf emission file and a data frame with*

Description

returns a traffic intensity profile (based on wrf file Times) and a traffic intensity data frame

Usage

```
wrf_profile(x, file, adjust = 0, verbose = T)
```

Arguments

x	data.frame of intenticy of traffic by hours (rows) and weekdays (columns)
file	emission file name
adjust	numer of hours to advance (positive value) or delay (negative value)
verbose	display additional information

Format

a numeric vector

Note

It might be deprecatedin future release

Author(s)

Daniel Schuch

See Also

wrf_create and to_wrf

Examples

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

# Profile based on Sao Paulo tunnel experiments
data(rawprofile)
rawprofile  <- matrix(rawprofile, nrow = 24, byrow = TRUE)
rawprofile <- as.data.frame(rawprofile)
names(rawprofile) <- c("Sunday","Monday","Tuesday","Wednesday","Thursday",
                       "Friday","Saturday")
row.names(rawprofile) <- c("00:00","01:00","02:00","03:00","04:00","05:00",
                           "06:00","07:00","08:00","09:00","10:00","11:00",
                           "12:00","13:00","14:00","15:00","16:00","17:00",
                           "18:00","19:00","20:00","21:00","22:00","23:00")

print(rawprofile)

# create the folder and emission file
dir.create(file.path(tempdir(), "EMISS"))
wrf_create(wrfinput_dir = system.file("extdata", package = "eixport"),
          wrfchemi_dir = file.path(tempdir(), "EMISS"),
          frames_per_auxinput5 = 24)

files <- list.files(path = file.path(tempdir(), "EMISS"),
                    pattern = "wrfchemi",
                    full.names = TRUE)
```

```
profile <- wrf_profile(rawprofile,files[1])

plot(profile, ty="l", lty = 2, axe = FALSE,
     main = "Traffic Intensity for Sao Paulo", xlab = "hour")
axis(2)
axis(1, at = 0.5 + c(0, 6, 12, 18, 24),
    labels = c("00:00","06:00","12:00","18:00", "00:00"))

## End(Not run)
```

---

wrf_put *Function to write variables in emission files*

Description

Extract variable

Usage

```
wrf_put(file = file.choose(), name = NA, POL, mult = NA, verbose = F)
```

Arguments

file	Character; name of file interactively (default) or specified
name	Character; name of the variable (any variable)
POL	Numeric; emissions input or string/POSIXlt time
mult	Numeric; multiplier. If the length is more than 1, it multiplies POL for each value of mult. It can be used if you want to add an hourly profile to your emissions.
verbose	display additional information

Author(s)

Daniel Schuch and Sergio Ibarra

See Also

wrf_plot and wrf_get

Examples

```
{
# create the folder and emission file
dir.create(file.path(tempdir(), "EMISS"))
wrf_create(wrfinput_dir = system.file("extdata", package = "eixport"),
          wrfchemi_dir = file.path(tempdir(), "EMISS"))

# get the name of created file
```

```
files <- list.files(path = file.path(tempdir(), "EMISS"),
                    pattern = "wrfchemi",
                    full.names = TRUE)

# open, put some numbers and write
CO <- wrf_get(file = files[1], name = "E_CO")
CO[] = rnorm(length(CO))
wrf_put(file = files[1], name = "E_CO", POL = CO)
}
```

Index