

Package: AirExposure (via r-universe)

October 9, 2026

Title Air Pollution Exposure Assessment Using Mobility Data

Version 0.0.0.9000

Description Provides tools to estimate personal exposure to air pollutants based on individual mobility patterns. The package integrates route information obtained from the TomTom Routing API with gridded air pollution data to estimate exposure along travel trajectories or under static exposure assumptions.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Imports dplyr, rlang, htmltools, http, jsonlite, leaflet, lubridate, sp, sf

Suggests knitr, rmarkdown, testthat (>= 3.0.0), webmockr

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://flortames.github.io/AirExposure/>

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libpng-dev libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository <https://flortames.r-universe.dev>

Date/Publication 2026-10-09 12:50:14 UTC

RemoteUrl <https://github.com/flortames/airexposure>

RemoteRef HEAD

RemoteSha b254e68bcf24c91e2eddb4f286b3f5758319a861

Contents

alternative_trajectories	2
function_hours	4
hourly_grid	4

make_grid	5
map_colors	6
points_to_line	7
temporary_grid_search	8
total_exposure	9
traditional_model	11
trajectories_tomtom	12
Index	14

alternative_trajectories
<i>Compare alternative travel routes using pollutant concentrations</i>

Description

Retrieves alternative routes between an origin and a destination using the TomTom Routing API and estimates the pollutant exposure associated with each route by intersecting the trajectories with hourly pollutant concentration grids.

Usage

```
alternative_trajectories(  
  origin,  
  dest,  
  mode,  
  dir,  
  key,  
  output,  
  hours = NULL,  
  gridID,  
  shapeValue,  
  units,  
  pollutant  
)
```

Arguments

origin	Character string containing the origin coordinates in the format "latitude,longitude".
dest	Character string containing the destination coordinates in the format "latitude,longitude".
mode	Character. Travel mode accepted by the TomTom Routing API (e.g., "car", "truck", "pedestrian").
dir	Character. Directory containing the hourly pollutant grid shapefiles.
key	Character. TomTom API key.
output	Character. Output type. Either "df" to return a data frame or "plot" to return an interactive leaflet map.

hours	Character. Departure date and time in the format "YYYY-mm-dd HH:MM:SS".
gridID	Character. Name of the grid identifier field.
shapeValue	Character. Name of the pollutant concentration field in the grid.
units	Character. Units of the pollutant concentration (e.g., "µg/m ³ ").
pollutant	Character. Pollutant name used for map labels and legends.

Details

The function identifies the fastest, shortest, least polluted, most polluted, lowest exposure, and highest exposure routes. Results can be returned either as a data frame or as an interactive map.

This function combines route information obtained from the TomTom Routing API with hourly pollutant concentration grids. If the trip spans multiple hourly grids, pollutant concentrations are averaged using `temporary_grid_search()`. Exposure is estimated as the product of the average pollutant concentration along each route and the travel time.

A valid TomTom API key and an active internet connection are required.

Value

If `output = "df"`, a data frame summarizing the selected alternative routes and their associated travel metrics, pollutant concentrations and exposure estimates.

If `output = "plot"`, an interactive leaflet map displaying the pollutant grid together with the selected alternative routes.

Examples

```
## Not run:
routes <- alternative_trajectories(
  origin = "-31.4201,-64.1888",
  dest = "-31.4300,-64.2000",
  mode = "car",
  hours = "2019-08-01 08:00:00",
  key = "YOUR_API_KEY",
  dir = "path/to/grids",
  gridID = "ID",
  shapeValue = "PM25",
  pollutant = "PM2.5",
  units = "µg/m3",
  output = "df"
)

## End(Not run)
```

function_hours	<i>Convert minutes to HH:MM format</i>
----------------	--

Description

Converts a duration expressed in minutes into a character string in "HH:MM" format.

Usage

```
function_hours(minutes)
```

Arguments

minutes	Numeric. Duration expressed in minutes.
---------	---

Value

A character string representing the duration in "HH:MM" format.

Examples

```
function_hours(125)
# Returns "02:05"
```

hourly_grid	<i>Retrieve the Hourly Grid File for a Given Date and Time</i>
-------------	--

Description

Searches a directory containing hourly pollutant grid shapefiles and returns the filename corresponding to the requested date and hour.

Usage

```
hourly_grid(hour, time_format, dir)
```

Arguments

hour	Character. Date and time used to identify the grid file.
time_format	Character. Format used to parse hour.
dir	Character. Directory containing the hourly grid shapefiles.

Details

Hourly grid shapefiles are expected to follow the naming convention "YYYY-mm-dd_HHMM.shp".

Value

A character string containing the name of the matching shapefile.

Examples

```
## Not run:
hourly_grid(
  hour = "2019-08-01 08:00:00",
  time_format = "%Y-%m-%d %H:%M:%S",
  dir = "path/to/grids"
)

## End(Not run)
```

make_grid

*Create example hourly pollutant grids***Description**

Creates a regular spatial grid over a user-defined bounding box and exports 24 hourly shapefiles. Each grid cell is assigned a random value, making this function useful for generating example datasets and testing workflows.

Usage

```
make_grid(ymin, ymax, xmin, xmax, pixelSize, dir, date, values = NULL)
```

Arguments

ymin	Numeric. Minimum latitude of the grid extent.
ymax	Numeric. Maximum latitude of the grid extent.
xmin	Numeric. Minimum longitude of the grid extent.
xmax	Numeric. Maximum longitude of the grid extent.
pixelSize	Numeric. Grid cell size in meters.
dir	Character. Directory where the shapefiles will be written.
date	Character. Date used in the output filenames ("YYYY-mm-dd").
values	Optional. Currently unused.

Details

The function creates one shapefile for each hour of the day. Pollutant values are generated randomly and are intended only for examples and testing.

Value

This function is called for its side effect of writing shapefiles to disk. No value is returned.

Examples

```
## Not run:
make_grid(
  ymin = -31.6,
  ymax = -31.3,
  xmin = -64.3,
  xmax = -64.0,
  pixelSize = 1000,
  dir = "example_grids/",
  date = "2019-08-01"
)

## End(Not run)
```

map_colors

Assign AQI categories and colors to a pollutant grid

Description

Classifies pollutant concentrations according to the U.S. Environmental Protection Agency (EPA) Air Quality Index (AQI) breakpoints and assigns a color to each grid cell for visualization.

Usage

```
map_colors(grid, pollutant)
```

Arguments

grid	An sf object containing a pollutant concentration field named value.
pollutant	Character. Pollutant name. Supported pollutants are "PM2.5", "PM10", "CO", "SO2", "NO2" and "O3".

Details

Two new columns are added to the input grid:

category AQI category assigned according to the pollutant concentration.

color Hexadecimal color associated with the AQI category.

AQI breakpoints follow the U.S. EPA Air Quality Index.

Value

The input sf object with two additional columns: category and color.

Examples

```
## Not run:
# grid must be an sf object containing
# a pollutant concentration column named "value"

colored_grid <- map_colors(
  grid = pollutant_grid,
  pollutant = "PM2.5"
)

## End(Not run)
```

points_to_line	<i>Convert Points to LINESTRING Geometries</i>
----------------	--

Description

Converts a set of ordered point coordinates into one or more LINESTRING geometries. If a grouping variable is provided, one LINESTRING is created for each group.

Usage

```
points_to_line(data, long, lat, id_field = NULL, sort_field = NULL)
```

Arguments

data	A data frame containing point coordinates.
long	Character. Name of the longitude column.
lat	Character. Name of the latitude column.
id_field	Optional character. Name of the column used to group points into different LINESTRING geometries.
sort_field	Optional character. Name of the column used to order points before creating the LINESTRING geometries.

Value

If `id_field` is `NULL`, returns an `sfc_LINESTRING` object with CRS EPSG:4326.

If `id_field` is provided, returns an `sf` object containing one LINESTRING for each group and a column with the corresponding group identifier.

Examples

```
## Not run:
# trajectory must contain point coordinates
# and an identifier for each route

line <- points_to_line(
  data = trajectory,
  long = "long",
  lat = "lat",
  id_field = "alternative",
  sort_field = "ID"
)

## End(Not run)
```

temporary_grid_search *Retrieve Pollutant Grid(s) for a Given Time Period*

Description

Retrieves the pollutant grid corresponding to a given date and time. When a time interval spanning multiple hours is provided, the function averages the pollutant values across all hourly grids and returns a single grid.

Usage

```
temporary_grid_search(
  start_hour,
  end_hour = NULL,
  dir,
  time_format,
  gridID,
  shapeValue
)
```

Arguments

start_hour	Character. Start date and time.
end_hour	Optional character. End date and time. If NULL, only the grid corresponding to start_hour is returned.
dir	Character. Directory containing the hourly grid shapefiles.
time_format	Character. Format used to parse start_hour and end_hour.
gridID	Character. Name of the field containing the grid cell identifier.
shapeValue	Character. Name of the field containing the pollutant concentration values.

Details

If start_hour and end_hour span multiple hours, the function retrieves every corresponding hourly grid, calculates the mean pollutant concentration for each grid cell, and returns the resulting averaged grid as an sf object.

The returned object is always transformed to the WGS84 geographic coordinate reference system (EPSG:4326).

Value

An sf object containing the pollutant grid corresponding to the requested hour or the average pollutant grid over the requested time interval.

Examples

```
## Not run:
temporary_grid_search(
  start_hour = "2019-08-01 08:00:00",
  end_hour = "2019-08-01 10:00:00",
  dir = "path/to/grids",
  time_format = "%Y-%m-%d %H:%M:%S",
  gridID = "ID",
  shapeValue = "PM25"
)

## End(Not run)
```

total_exposure	<i>Estimate total daily exposure to an air pollutant</i>
----------------	--

Description

Estimates an individual's total daily exposure to an air pollutant by combining exposure during trips, exposure at destinations, and exposure at the home location. Travel routes are obtained from the TomTom Routing API, while pollutant concentrations are extracted from hourly grid shapefiles.

Usage

```
total_exposure(
  travel_list,
  mode,
  dir,
  key,
  selection,
  output_exp,
  departure_time_home,
  activity_minutes,
```

```

    pollutant,
    shapeValue,
    gridID,
    units
  )

```

Arguments

travel_list	Data frame containing the sequence of locations visited during the day. It must contain latitude and longitude columns. The first location is assumed to be the home location and the final trip always returns to this point.
mode	Character vector indicating the travel mode for each trip (for example "car", "pedestrian" or "bicycle").
dir	Character. Working directory containing the pollutant grids and temporary files.
key	Character. TomTom API key.
selection	Integer or character vector indicating which alternative route is selected for each trip.
output_exp	Character. Output type. One of "df", "plot" or "polyline".
departure_time_home	Character. Departure date and time from the home location.
activity_minutes	Numeric vector indicating the duration (minutes) of the activity performed at each destination.
pollutant	Character. Pollutant name used for map classification (for example "PM2.5").
shapeValue	Character. Name of the pollutant concentration field in the grid shapefiles.
gridID	Character. Name of the grid identifier field.
units	Character. Pollutant concentration units displayed in the map.

Details

The function performs the following steps:

- Retrieves alternative routes between consecutive locations using the TomTom Routing API.
- Selects one route for each trip according to selection.
- Estimates exposure during travel using the selected route.
- Estimates exposure while remaining at each destination.
- Estimates exposure at the home location before the first trip.
- Calculates total daily exposure.
- Optionally returns an interactive map showing routes and pollutant concentrations.

This function relies on the helper functions `alternative_trajectories()`, `temporary_grid_search()`, `points_to_line()`, `map_colors()`, and `function_hours()`.

Value

Depending on output_exp:

"df" A data frame summarizing exposure for each trip and the total daily exposure.

"plot" An interactive leaflet map showing the selected routes, visited locations and pollutant concentrations.

"polyline" An sf object containing the route polylines.

Examples

```
## Not run:
exposure <- total_exposure(
  travel_list = travel_list,
  mode = c("car", "pedestrian"),
  dir = data_dir,
  key = api_key,
  selection = c(1, 1),
  output_exp = "plot",
  departure_time_home = "2019-08-01 08:00:00",
  activity_minutes = c(480),
  pollutant = "PM2.5",
  shapeValue = "value",
  gridID = "ID",
  units = expression(mu * g/m^3)
)

## End(Not run)
```

traditional_model	<i>Estimate Daily Exposure Using the Traditional Model</i>
-------------------	--

Description

Estimates daily exposure assuming that an individual remains at the origin location throughout the entire day. Exposure is calculated by intersecting the origin point with the daily average pollutant grid and multiplying the resulting concentration by 24 hours.

Usage

```
traditional_model(origin_point, date, dir, gridID, shapeValue)
```

Arguments

origin_point	A data frame containing the coordinates of the origin location. It must include the columns longitude and latitude.
date	Character. Date of interest in the format "YYYY-mm-dd".

dir	Character. Directory containing the hourly pollutant grid shapefiles.
gridID	Character. Name of the field containing the grid cell identifier.
shapeValue	Character. Name of the field containing the pollutant concentration values.

Details

The function retrieves all hourly pollutant grids for the requested day, computes the daily average concentration using `temporary_grid_search()`, identifies the grid cell containing the origin location, and estimates daily exposure assuming continuous residence at that location.

Value

A numeric value representing the estimated daily exposure.

Examples

```
## Not run:
traditional_model(
  origin_point = home_location,
  date = "2019-08-01",
  dir = "path/to/grids",
  gridID = "ID",
  shapeValue = "PM25"
)

## End(Not run)
```

trajectories_tomtom	<i>Retrieve Alternative Routes from the TomTom Routing API</i>
---------------------	--

Description

Retrieves up to five alternative routes between an origin and destination using the TomTom Routing API. The function returns the route geometry as point coordinates together with travel time, distance, and traffic-related metrics for each alternative route.

Usage

```
trajectories_tomtom(origin, dest, mode, hour_trajectory, key)
```

Arguments

origin	Character string containing the origin coordinates in the format "latitude,longitude".
dest	Character string containing the destination coordinates in the format "latitude,longitude".
mode	Character. Travel mode accepted by the TomTom Routing API (e.g., "car", "truck", "pedestrian", "bicycle").

hour_trajectory	Character. Departure date and time in the format "YYYY-mm-dd HH:MM:SS".
key	Character. Valid TomTom Routing API key.

Details

This function requires a valid TomTom Routing API key and an active internet connection. The output includes one row for each point along each alternative route returned by the API.

Value

A data frame with one row per route point and the following variables:

ID Sequential point identifier within each route.
long Longitude of the route point.
lat Latitude of the route point.
departureTime Departure time returned by the API.
arrivalTime Arrival time returned by the API.
lengthInKM Route length in kilometers.
trafficLengthInKM Length affected by traffic in kilometers.
travelMode Travel mode used for routing.
trafficDelayInMinutes Estimated traffic delay in minutes.
travelTimeInMinutes Estimated travel time in minutes.
liveTrafficIncidentsTravelTimeInMinutes Travel time considering live traffic incidents.
historicTrafficTravelTimeInMinutes Travel time estimated from historical traffic conditions.
noTrafficTravelTimeInMinutes Estimated travel time assuming no traffic.
alternative Alternative route identifier.

Examples

```
## Not run:
routes <- trajectories_tomtom(
  origin = "-31.4201,-64.1888",
  dest = "-31.4300,-64.2000",
  mode = "car",
  hour_trajectory = "2019-08-01 08:00:00",
  key = "YOUR_API_KEY"
)

## End(Not run)
```

Index

`alternative_trajectories`, [2](#)

`function_hours`, [4](#)

`hourly_grid`, [4](#)

`make_grid`, [5](#)

`map_colors`, [6](#)

`points_to_line`, [7](#)

`temporary_grid_search`, [8](#)

`total_exposure`, [9](#)

`traditional_model`, [11](#)

`trajectories_tomtom`, [12](#)