Package ‘spatialrisk’

March 21, 2020

Type Package
Title Calculating Concentration Risk under Solvency II
Version 0.6.6
Author Martin Haringa
Maintainer Martin Haringa <mtharinga@gmail.com>
Description Methods for determining spatial risk, in particular calculating the maximum value of insured fire risk policies of all buildings that are partly or fully located within circle of a radius of 200m.
License GPL (>= 2)
LazyData true
LinkingTo Rcpp, RcppProgress
Imports automap, classInt, dplyr, fs, ggplot2, gstat, lubridate, methods, mgcv, Rcpp, RcppProgress, sf, sp, tmap, viridis, vroom
Depends R (>= 3.3)
RoxygenNote 7.1.0
Suggests knitr, rmarkdown, testthat
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-03-21 08:40:02 UTC

R topics documented:

choropleth .................................................. 2
choropleth_ggplot2 ........................................ 3
choropleth_sf ............................................... 4
choropleth_tmap ........................................... 5
concentration ............................................. 6
europe_countries ......................................... 7
Groningen ................................................... 8
haversine ................................................... 8
insurance .................................................. 9
Create choropleth map

Takes an object produced by `points_to_polygon()`, and creates the corresponding choropleth map.

```r
color_fill <- function(s, value = "output", id_name = "areaname", mode = "plot", n = 7, legend_title = "Clustering", palette = "viridis")
```

Arguments

- `sf_object`: object of class sf
- `value`: column name to shade the polygons
- `id_name`: column name of ids to plot
- `mode`: choose between static ("plot" is default) and interactive map ("view")
- `n`: number of clusters (default is 7)
- `legend_title`: title of legend
- `palette`: palette name or a vector of colors. Use a "-" as prefix to reverse the palette. The default palette is "viridis".
Value
tmap

Author(s)
Martin Haringa

Examples

\begin{verbatim}
  test <- points_to_polygon(nl_provincie, insurance, sum(amount, na.rm = TRUE))
  choropleth(test)
  choropleth(test, id_name = "areaname", mode = "view")
\end{verbatim}

\section*{choropleth_ggplot2}

\textit{Map object of class sf using ggplot2}

\section*{Description}
Takes an object produced by \texttt{choropleth.sf()}, and creates the corresponding choropleth map.

\section*{Usage}

\begin{verbatim}
choropleth_ggplot2(  
  sf_object,  
  value = output,  
  n = 7,  
  dig.lab = 2,  
  legend_title = "Class",  
  option = "D",  
  direction = 1
)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \texttt{sfobject} \hspace{1cm} object of class sf
  \item \texttt{value} \hspace{1cm} column to shade the polygons
  \item \texttt{n} \hspace{1cm} number of clusters (default is 7)
  \item \texttt{dig.lab} \hspace{1cm} number of digits in legend (default is 2)
  \item \texttt{legend_title} \hspace{1cm} title of legend
  \item \texttt{option} \hspace{1cm} a character string indicating the colormap option to use. Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C"), "viridis" (or "D", the default option) and "cividis" (or "E").
  \item \texttt{direction} \hspace{1cm} Sets the order of colors in the scale. If 1, the default, colors are ordered from darkest to lightest. If -1, the order of colors is reversed.
\end{itemize}
Value

ggplot map

Author(s)

Martin Haringa

Examples

test <- points_to_polygon(nl_postcode2, insurance, sum(amount, na.rm = TRUE))
choropleth_ggplot2(test)

---

choropleth_sf

Aggregate attributes of coordinates to area level (deprecated function; use 'points_to_polygon' instead)

Description

A data.frame containing coordinates (in terms of longitude and latitude) is joined to the polygon level. Then arithmetic operations on the attributes of the coordinates are applied to obtain aggregated values for each polygon.

Usage

choropleth_sf(sf_map, df, oper, crs = 4326, outside_print = FALSE)

Arguments

sf_map object of class sf
df data.frame containing coordinates (column names should be 'lon' and 'lat')
oper an arithmetic operation on the polygon level
crs coordinate reference system: integer with the EPSG code, or character with proj4string
outside_print print points that are not within a polygon (default is FALSE).

Value

an object of class sf

Author(s)

Martin Haringa
choropleth_tmap

Map object of class sf using tmap (deprecated function; use 'choropleth' instead)

Description

Takes an object produced by choropleth_sf(), and creates the corresponding choropleth map.

Usage

choropleth_tmap(
  sf_object,
  value = "output",
  id_name = "areaname",
  mode = "plot",
  n = 7,
  legend_title = "Clustering",
  palette = "viridis"
)

Arguments

  sf_object    object of class sf
  value        column name to shade the polygons
  id_name      column name of ids to plot
  mode         choose between static (‘plot’ is default) and interactive map (‘view’)
  n            number of clusters (default is 7)
  legend_title title of legend
  palette      palette name or a vector of colors. See tmaptools::palette_explorer() for the named palettes. Use a "-" as prefix to reverse the palette. The default palette is "viridis".

Value

tmap

Author(s)

Martin Haringa
The sum of all observations within a radius from center point(s). In particular, it can be used to determine concentration risk in the context of the EU insurance regulation framework (Solvency II). The function offers an effective approach to calculate the 'standard formula' under Solvency II. The 'standard formula' under Solvency II asks companies to report their largest fire concentration in respect of the fire peril within a radius of 200m. This is the maximum gross sum insured of the set of buildings fully or partly located within this radius (Commission Delegated Regulation (EU), 2015, Article 132).

Usage

concentration(
  sub,
  full,
  value,
  lon_sub = lon,
  lat_sub = lat,
  lon_full = lon,
  lat_full = lat,
  radius = 200,
  display_progress = TRUE
)

Arguments

sub data.frame of locations to calculate concentration risk for (target points).
full data.frame to find the locations within radius \( r \) from locations in sub (reference locations).
value Column with value in full.
lon_sub Column in sub with longitude (lon is default).
lat_sub Column in sub with latitude (lat is default).
lon_full Column in full with longitude in full (lon is default).
lat_full Column in full with latitude in full (lat is default).
radius Radius (in meters) (default is 200m).
display_progress Show progress bar (TRUE/FALSE).

Details

The data.frame \( \text{sub} \) should include at least columns for longitude and latitude.
The data.frame \( \text{full} \) should include at least columns for longitude, latitude and value of interest to summarize.
Value

A data.frame equal to data.frame sub including an extra column concentration.

Author(s)

Martin Haringa

References


Examples

df <- data.frame(location = c("p1", "p2"), lon = c(6.561561, 6.561398), lat = c(53.21369, 53.21326))
concentration(df, Groningen, value = amount, radius = 100)

europe_countries

Object of class sf for countries of Europe

Description

An object of class sf (simple feature) for countries of Europe

Usage

europe_countries

Format

A simple feature object with 51 rows and 29 variables.

Details

The epsg (SRID) is set to 102013 (Europe Albers Equal Area Conic).

Author(s)

Martin Haringa
Groningen

Coordinates of houses in Groningen

Description

A dataset of postal codes and the corresponding spatial locations in terms of a latitude and a longitude.

Usage

Groningen

Format

A data frame with 56200 rows and 8 variables:

- **street** Name of street
- **number** Number of house
- **letter** Letter of house
- **suffix** Suffix to number of house
- **postal_code** Postal code of house
- **city** The name of the city
- **lon** Longitude (in degrees)
- **lat** Latitude (in degrees)
- **amount** Random value

Source

The BAG is the Dutch registry for Buildings and addresses (Basisregistratie adressen en gebouwen).

haversine

Haversine great circle distance

Description

The shortest distance between two points (i.e., the 'great-circle-distance' or 'as the crow flies'), according to the 'haversine method'. This method assumes a spherical earth, ignoring ellipsoidal effects. Note that this version is implemented in C++. A quick benchmark to the version of geosphere showed it to be a non-insignificant speed enhancement. The algorithm converges in one-twentieth of the original time.

Usage

haversine(lat_from, lon_from, lat_to, lon_to, r = 6378137)
Arguments

lat_from  Latitude of point.
lon_from  Longitude of point.
lat_to    Latitude of point.
lon_to    Longitude of point.
r         Radius of the earth; default = 6378137m

Details

The Haversine ('half-versed-sine') formula was published by R.W. Sinnott in 1984, although it has been known for much longer.

Value

Vector of distances in the same unit as r (default in meters).

Author(s)

Martin Haringa

References


Examples

haversine(53.24007, 6.520386, 53.24054, 6.520386)

<table>
<thead>
<tr>
<th>insurance</th>
<th>Sum insured per postal code in the Netherlands</th>
</tr>
</thead>
</table>

Description

A dataset of postal codes with their sum insured, population and the corresponding spatial locations in terms of a latitude and a longitude.

Usage

insurance
Format

A data frame with 29,990 rows and 5 variables:

- **postcode**: 6-digit postal code
- **population_pc4**: Population per 4-digit postal code
- **amount**: Sum insured
- **lon**: Longitude (in degrees) of the corresponding 6-digit postal code
- **lat**: Latitude (in degrees) of the corresponding 6-digit postal code

interpolate_krige

Ordinary kriging

Description

Interpolation and smoothing on the sphere by means of ordinary kriging.

Usage

```r
interpolate_krige(
    observations,
    targets,
    value,
    lon_obs = lon,
    lat_obs = lat,
    lon_targets = lon,
    lat_targets = lat
)
```

Arguments

- **observations**: data.frame of observations.
- **targets**: data.frame of locations to calculate the interpolated and smoothed values for (target points).
- **value**: Column with values in observations.
- **lon_obs**: Column in observations with longitude (lon is default).
- **lat_obs**: Column in observations with latitude (lat is default).
- **lon_targets**: Column in targets with longitude (lon is default).
- **lat_targets**: Column in targets with latitude (lat is default).
Details

observations should include at least columns for longitude and latitude.

targets should include at least columns for longitude, latitude and value of interest to interpolate and smooth.

Kriging can be considered as linear regression with spatially correlated residuals. Kriging is most appropriate when it is known there is a spatially correlated distance or directional bias in the data. It is often used in soil science and geology.

See splines on the sphere for interpolation and smoothing on the sphere by means of splines.

Value

Object equal to object targets including extra columns for the predicted value and the variance.

Author(s)

Martin Haringa

References

gstat::krige

Examples

```r
## Not run:
target <- sf::st_drop_geometry(nl_postcode3)
obs <- insurance %>% dplyr::sample_n(1000)
pop_df <- interpolate_krige(obs, target, population_pc4)
pop_sf <- left_join(nl_postcode3, pop_df)
choropleth(pop_sf, value = "population_pc4_pred", n = 13)
choropleth(pop_sf, value = "population_pc4_var", n = 13)
## End(Not run)
```

interpolate_spline  Spline interpolation and smoothing on the sphere.
Usage

```r
interpolate_spline(
  observations,
  targets,
  value,
  lon_obs = lon,
  lat_obs = lat,
  lon_targets = lon,
  lat_targets = lat,
  k = 50
)
```

Arguments

- `observations`: data.frame of observations.
- `targets`: data.frame of locations to calculate the interpolated and smoothed values for (target points).
- `value`: Column with values in `observations`.
- `lon_obs`: Column in `observations` with longitude (`lon` is default).
- `lat_obs`: Column in `observations` with latitude (`lat` is default).
- `lon_targets`: Column in `targets` with longitude (`lon` is default).
- `lat_targets`: Column in `targets` with latitude (`lat` is default).
- `k`: (default 50) is the basis dimension. For small data sets reduce `k` manually rather than using default.

Details

- `observations` should include at least columns for longitude and latitude.
- `targets` should include at least columns for longitude, latitude and value of interest to interpolate and smooth.

A smooth of the general type discussed in Duchon (1977) is used: the sphere is embedded in a 3D Euclidean space, but smoothing employs a penalty based on second derivatives (so that locally as the smoothing parameter tends to zero we recover a “normal” thin plate spline on the tangent space). This is an unpublished suggestion of Jean Duchon.

See ordinary kriging for interpolation and smoothing on the sphere by means of kriging.

Value

Object equal to object `targets` including an extra column with predicted values.

Author(s)

Martin Haringa

References

- Splines on the sphere
knmi_historic_data

## Examples
```r
## Not run:
target <- sf::st_drop_geometry(nl_postcode3)
obs <- dplyr::sample_n(insurance, 1000)
pop_df <- interpolate_spline(obs, target, population_pc4, k = 20)
pop_sf <- left_join(nl_postcode3, pop_df)
choropleth(pop_sf, value = "population_pc4_pred", n = 13)
## End(Not run)
```

knmi_historic_data

Retrieve historic weather data for the Netherlands

### Description
This function retrieves historic weather data collected by the official KNMI weather stations. See spatialrisk::knmi_stations for a list of the official KNMI weather stations.

### Usage

```r
knmi_historic_data(startyear, endyear)
```

### Arguments

- `startyear`: start year for historic weather data.
- `endyear`: end year for historic weather data.

### Format
The returned data frame contains the following columns:
- `station`: ID of measurement station;
- `date`: Date;
- `FH`: Hourly mean wind speed (in 0.1 m/s);
- `FX`: Maximum wind gust (in 0.1 m/s) during the hourly division;
- `T`: Temperature (in 0.1 degrees Celsius) at 1.50 m at the time of observation;
- `DR`: Precipitation duration (in 0.1 hour) during the hourly division;
- `RH`: Hourly precipitation amount (in 0.1 mm) (-1 for <0.05 mm);
- `city`: City where the measurement station is located;
- `lon`: Longitude of station (crs = 4326);
- `lat`: Latitude of station (crs = 4326).

### Value
Data frame containing weather data and meta data for weather station locations.
Author(s)

Martin Haringa

Examples

```r
## Not run:
knni_historic_data(2015, 2019)
## End(Not run)
```

---

**knmi_stations**

<table>
<thead>
<tr>
<th>knmi_stations</th>
<th>KNMI stations</th>
</tr>
</thead>
</table>

### Description

A data frame containing the IDs and meta-data on the official KNMI weather stations.

### Usage

```r
knmi_stations
```

### Format

A data frame with 50 rows and 7 variables:

- **station**: ID of the station (209-391)
- **city**: City where the station is located
- **lon**: Longitude of station (crs = 4326)
- **lat**: Latitude of the station (crs = 4326)
- **altitude**: Altitude of the station (in meters)
- **X**: X coordinate of the station (crs = 32631)
- **Y**: Y coordinate of the station (crs = 32631)

### Author(s)

Martin Haringa
Description

An object of class sf (simple feature) for COROP regions in the Netherlands.

Usage

nl_corop

Format

A simple feature object with 40 rows and 5 variables:

- corop_nr  corop number
- areaname  corop name
- geometry  geometry object of COROP region
- lon       longitude of the corop centroid
- lat       latitude of the corop centroid

Details

A COROP region is a regional area within the Netherlands. These regions are used for analytical purposes by, among others, Statistics Netherlands. The Dutch abbreviation stands for Coordinatiecommissie Regionaal Onderzoeksprogramma, literally the Coordination Commission Regional Research Programme.

Author(s)

Martin Haringa

Description

An object of class sf (simple feature) for gemeentes (English: municipalities) in the Netherlands in the year 2018.

Usage

nl_gemeente
Format

A simple feature object with 380 rows and 6 variables:

- id  id of gemeente
- code code of gemeente
- areaname name of gemeente
- geometry geometry object of gemeente
- lon longitude of the gemeente centroid
- lat latitude of the gemeente centroid

Author(s)

Martin Haringa

---

nl_postcode1  Object of class sf for 1-digit postcode regions in the Netherlands

Description

An object of class sf (simple feature) for 1-digit postcode (English: postal code) regions in the Netherlands.

Usage

nl_postcode1

Format

A simple feature object with 9 rows and 4 variables:

- areaname 1-digit postal code
- geometry geometry object of postal code
- lon longitude of the 1-digit postal code centroid
- lat latitude of the 1-digit postal code centroid

Details

Postal codes in the Netherlands, known as postcodes, are alphanumeric, consisting of four digits followed by two uppercase letters. The first two digits indicate a city and a region, the second two digits and the two letters indicate a range of house numbers, usually on the same street.

Author(s)

Martin Haringa
Description

An object of class sf (simple feature) for 2-digit postcode (English: postal code) regions in the Netherlands.

Usage

nl_postcode2

Format

A simple feature object with 90 rows and 4 variables:

- areaname 2-digit postal code
- geometry geometry object of postal code
- lon longitude of the 2-digit postal code centroid
- lat latitude of the 2-digit postal code centroid

Details

Postal codes in the Netherlands, known as postcodes, are alphanumeric, consisting of four digits followed by two uppercase letters. The first two digits indicate a city and a region, the second two digits and the two letters indicate a range of house numbers, usually on the same street.

Author(s)

Martin Haringa

Description

An object of class sf (simple feature) for 3-digit postcode (English: postal code) regions in the Netherlands.

Usage

nl_postcode3
Format

A simple feature object with 799 rows and 3 variables:

- **areaname**: 3-digit postal code
- **geometry**: geometry object of postal code
- **lon**: longitude of the 3-digit postal code centroid
- **lat**: latitude of the 3-digit postal code centroid

Details

Postal codes in the Netherlands, known as postcodes, are alphanumeric, consisting of four digits followed by two uppercase letters. The first two digits indicate a city and a region, the second two digits and the two letters indicate a range of house numbers, usually on the same street.

Author(s)

Martin Haringa

---

**nl_postcode4**

*Object of class sf for 4-digit postcode regions in the Netherlands*

Description

An object of class sf (simple feature) for 4-digit postcode (English: postal code) regions in the Netherlands.

Usage

`nl_postcode4`

Format

A simple feature object with 4053 rows and 7 variables:

- **pc4**: 4-digit postal code
- **areaname**: name of corresponding 4-digit postal code
- **city**: name of city
- **biggest_20cities**: pc4 is in one of the following twenty (biggest) cities in the Netherlands: Amsterdam, Rotterdam, ’s-Gravenhage, Utrecht, Eindhoven, Tilburg, Groningen, Almere, Breda, Nijmegen, Enschede, Apeldoorn, Haarlem, Amersfoort, Arnhem, ’s-Hertogenbosch, Zoetermeer, Zwolle, Maastricht, Leiden.
- **geometry**: geometry object of postal code
- **lon**: longitude of the 4-digit postal code centroid
- **lat**: latitude of the 4-digit postal code centroid
Postal codes in the Netherlands, known as postcodes, are alphanumeric, consisting of four digits followed by two uppercase letters. The first two digits indicate a city and a region, the second two digits and the two letters indicate a range of house numbers, usually on the same street.

Author(s)

Martin Haringa

Description

An object of class sf (simple feature) for provincies (provinces) in the Netherlands.

Usage

nl_provincie

Format

A simple feature object with 12 rows and 4 variables:

- areaname: province name
- geometry: geometry object of province
- lon: longitude of the province centroid
- lat: latitude of the province centroid

Author(s)

Martin Haringa
points_in_circle  

Description

The observations within radius from the center point.

Usage

points_in_circle(
  data,
  lon_center,
  lat_center,
  lon = lon,
  lat = lat,
  radius = 200
)

Arguments

data A data.frame.
lon_center Longitude of center point.
lat_center Latitude of center point.
lon Name of column in data with longitudes (lon is default).
lat Name of column in data with latitudes (lat is default).
radius Radius (in meters) (default is 200m).

Value

A data.frame of coordinates within radius around (lon_center, lat_center). The column distance_m gives the distance from the center point (in meters).

Author(s)

Martin Haringa

Examples

points_in_circle(Groningen, lon_center = 6.571561, lat_center = 53.21326, radius = 50)
points_to_polygon  Aggregate attributes of coordinates to area level

Description

A data.frame containing coordinates (in terms of longitude and latitude) is joined to the polygon level. Then arithmetic operations on the attributes of the coordinates are applied to obtain aggregated values for each polygon.

Usage

points_to_polygon(sf_map, df, oper, crs = 4326, outside_print = FALSE)

Arguments

sf_map object of class sf
df data.frame containing coordinates (column names should be 'lon' and 'lat')
oper an arithmetic operation on the polygon level
crs coordinate reference system: integer with the EPSG code, or character with proj4string
outside_print print points that are not within a polygon (default is FALSE).

Value

an object of class sf

Author(s)

Martin Haringa

Examples

points_to_polygon(nl_postcode2, insurance, sum(amount, na.rm = TRUE))
## Not run:
shp_read <- sf::st_read(~path/to/file.shp)
points_to_polygon(shp_read, insurance, sum(amount, na.rm = TRUE))
## End(Not run)
world_countries

Object of class sf for countries of the entire world

Description
An object of class sf (simple feature) for countries of the entire world.

Usage
world_countries

Format
A simple feature object with 234 rows and 29 variables.

Author(s)
Martin Haringa
Index

*Topic datasets
  - europe_countries, 7
  - Groningen, 8
  - insurance, 9
  - knmi_stations, 14
  - nl_corop, 15
  - nl_gemeente, 15
  - nl_postcode1, 16
  - nl_postcode2, 17
  - nl_postcode3, 17
  - nl_postcode4, 18
  - nl_provincie, 19
  - world_countries, 22

  choropleth, 2
  choropleth_ggplot2, 3
  choropleth_sf, 4
  choropleth_tmap, 5
  concentration, 6

  europe_countries, 7

  Groningen, 8
  gstat::krige, 11

  haversine, 8

  insurance, 9
  interpolate_krige, 10
  interpolate_spline, 11

  knmi_historic_data, 13
  knmi_stations, 14

  nl_corop, 15
  nl_gemeente, 15
  nl_postcode1, 16
  nl_postcode2, 17
  nl_postcode3, 17
  nl_postcode4, 18
  nl_provincie, 19

  ordinary kriging, 12
  points_in_circle, 20
  points_to_polygon, 21
  Splines on the sphere, 12
  splines on the sphere, 11
  world_countries, 22

23