

# Package ‘spatialrisk’

July 29, 2019

**Type** Package

**Title** Calculating Concentration Risk under Solvency II

**Version** 0.6.2

**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** classInt, dplyr, ggplot2, Rcpp, RcppProgress, sf, shiny, tmap, viridis

**Depends** R (>= 3.3)

**RoxygenNote** 6.1.1

**Suggests** knitr, rmarkdown, testthat

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2019-07-29 14:00:02 UTC

## R topics documented:

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

nl_gemeente . . . . .	9
nl_postcode1 . . . . .	10
nl_postcode2 . . . . .	11
nl_postcode3 . . . . .	11
nl_postcode4 . . . . .	12
nl_provincie . . . . .	13
points_in_circle . . . . .	13
world_countries . . . . .	14

---

choropleth\_ggplot2 *Map object of class sf using ggplot2*

---

### Description

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

### Usage

```
choropleth_ggplot2(sf_object, value = output, n = 7, dig.lab = 2,
  legend_title = "Class", option = "D", direction = 1)
```

### Arguments

<code>sf_object</code>	object of class <code>sf</code>
<code>value</code>	column to shade the polygons
<code>n</code>	number of clusters (default is 7)
<code>dig.lab</code>	number of digits in legend (default is 2)
<code>legend_title</code>	title of legend
<code>option</code>	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").
<code>direction</code>	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.

### Value

ggplot map

### Author(s)

Martin Haringa

### Examples

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

---

choropleth\_sf      *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

```
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

### Examples

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

## End(Not run)
```

---

choropleth\_tmap      *Map object of class sf using tmap*

---

### 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

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

### Value

ggplot map

### Author(s)

Martin Haringa

### Examples

```
test <- choropleth_sf(nl_provincie, insurance, sum(amount, na.rm = TRUE))
choropleth_tmap(test)
choropleth_tmap(test, id_name = "areaname", mode = "view")
```

---

concentration	<i>Concentration risk</i>
---------------	---------------------------

---

### Description

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

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

### Details

The data.frame `sub` should include at least columns for longitude and latitude.

The data.frame `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**

Commission Delegated Regulation (EU) (2015). Solvency II Delegated Act 2015/35. Official Journal of the European Union, 58:124.

**Examples**

```
df <- data.frame(location = c("p1", "p2"), lon = c(6.561561, 6.561398), lat = c(53.21369, 53.21369),
  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**

Sinnott, R.W, 1984. Virtues of the Haversine. *Sky and Telescope* 68(2): 159.

**Examples**

```
haversine(53.24007, 6.520386, 53.24054, 6.520386)
```

---

insurance

*Sum insured per postal code in the Netherlands*

---

**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 30,000 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



---

`nl_corop`*Object of class sf for COROP regions in the Netherlands*

---

**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 3 variables:

**corop\_nr** corop number

**areaname** corop name

**geometry** geometry object of COROP region

**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 Coördinatiecommissie Regionaal Onderzoeksprogramma, literally the Coordination Commission Regional Research Programme.

**Author(s)**

Martin Haringa

---

`nl_gemeente`*Object of class sf for gemeentes (municipalities) in the Netherlands*

---

**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 4 variables:

**id** id of gemeente

**code** code of gemeente

**areaname** name of gemeente

**geometry** geometry object of gemeente

**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 2 variables:

**areaname** 1-digit postal code

**geometry** geometry object of postal code

**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\_postcode2                      *Object of class sf for 2-digit postcode regions in the Netherlands*

---

**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 2 variables:

**areaname** 2-digit postal code

**geometry** geometry object of postal code

**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\_postcode3                      *Object of class sf for 3-digit postcode regions in the Netherlands*

---

**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 2 variables:

**areaname** 3-digit postal code

**geometry** geometry object of postal code

**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	<i>Object of class sf for 4-digit postcode regions in the Netherlands</i>
--------------	---

---

**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 2 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

**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\_provincie      *Object of class sf for provinces (provinces) in the Netherlands*

---

### Description

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

### Usage

```
nl_provincie
```

### Format

A simple feature object with 12 rows and 2 variables:

**areaname** province name

**geometry** geometry object of province

### Author(s)

Martin Haringa

---

points\_in\_circle      *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)
```

---

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