# Package 'rancovr'

February 15, 2022

**Type** Package

**Title** Cluster detection in R with RAndom Neighbourhood COVeRing

**Version** 0.1.0

**Author** Massimo Cavallaro

**Maintainer** Massimo Cavallaro <mcavallaro@warwick.ac.uk>

**Description**

rancovr detects statio-temporal clusters against a baseline of sparse Poisson point events using the random neighbourhood covering algorithm.

**URL** https://github.com/mcavallaro/rancovr

**License** MIT License

**Encoding** UTF-8

**LazyData** True

**Imports** Matrix, optimx, truncnorm, sf, jsonlite

**Suggested** spatstat, KernSmooth, tsne

**RoxygenNote** 7.1.2

## R topics documented:

---

`Clean` *Clean*

---

## Description

Delete all '*.RData' files with given basename. Can be used to clean the saved observation and baseline matrices. Use with caution.

## Usage

```
Clean(basename)
```

## Arguments

| | |
|---|---|
| `basename` | A `character` string. |

## Value

None

## Examples

```
Clean("33.0_observation_matrix_tmp")
Clean("observation_matrix_tmp")
```

---

`cmle` *Estimate the parameters for the seasonal model given the aggregated vector of event times.*

---

## Description

Optimize the function `neg.log.like` recursively calling `clme1` and `cmle2`.

## Usage

```
cmle(data, n.cycles = 10, start = NULL, save.on.dir = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | A `numeric` vector of event times. |
| `n.cycles` | `integer`. Number of times the conditional optimizers are called.# |
| `start` | `numeric`. Starting parameters (not yet implemented). |
| `save.on.dir` | `logical`. If TRUE, will save the inferred parameter in 'timefactor_parameters.Rdata'. |

## Value

`numeric`. A 2D 2x4 matrix with the estimated parameters.

## Examples

```
cmle(data)
```

---

`cmle1` *Optimize function* `neg.log.like` *conditioned on having the last parameter fixed.*

---

## Description

Optimize function `neg.log.like` conditioned on having the last parameter fixed.

## Usage

```
cmle1(data, cpar, iterations = 10000)
```

## Arguments

| | |
|---|---|
| `data` | A `numeric` vector of event times. |
| `cpar` | A `numeric`. |
| `iteration` | This is the maximum number of interations allowed for the function `optimx`. |

## Value

2D 4x2 matrix.

## Examples

```
cmle1(data, 1, iterations=20)
```

---

`cmle2` *Optimize function* `neg.log.like` *conditioned on having the first three parameters fixed.*

---

## Description

Optimize function `neg.log.like` conditioned on having the first three parameters fixed.

## Usage

```
cmle2(data, cpar)
```

## Arguments

| | |
|---|---|
| `data` | A `numeric` vector of event times. |
| `cpar` | `numeric`. |

## Value

`numeric`.

## Examples

```
cmle2(data, c(1,1,1))
```

---

`compute` *Compute exceedance probabality in a cylinder.*

---

## Description

A cylinder is defined by the circle coordinated (say, x,y, and radius) and lower and upper height limits (aay, t.low and t.upp, respectively). For a given cylinder, this function computes the number of observed events (`n_cases`) in the cylinder according to `observation.matrix`, the expected number `mu` of events according the Poisson point model (with intensity defined in `baseline.matrix`), and the probability that . The function returns `c(n_cases,mu,p.val)`.

## Usage

```
compute(cylinder, observation.matrix, baseline.matrix, postcode.locations)
```

### Arguments

| | |
|---|---|
| cylinder | |
| observation.matrix | A sparseMatrix object enconding the events. |
| baseline.matrix | A Matrix object enconding the baseline. |
| postcode.locations | A data.frame that maps the rows of observation.matrix to geographical coordinates. |

### Value

A numeric vector of dimension 3.

### Examples

```
exceedance=compute(c(x,y,rho,t.low,t.upp), observation.matrix, baseline.matrix, postcode.locations)
```

---

## compute.from.tab.baseline

*postcode.in.england*

---

### Description

Check if postcode is in England.

Check if postcode is in England.

### Usage

```
compute.from.tab.baseline(
  cylinder,
  observation.matrix,
  tab.baseline,
  postcode.locations
)

compute.from.tab.baseline(
  cylinder,
  observation.matrix,
  tab.baseline,
  postcode.locations
)
```

### Arguments

| | |
|---|---|
| x | integer. number of cylinder samples. |
| postcode.field | numeric. |

## CreateBaselineMatrix *Create and save a baseline matrix*

### Description

Compute and save on disk a 2D dense matrix representing the spatial component of the baseline.

### Usage

```
CreateBaselineMatrix(
  case.df,
  save.on.dir = FALSE,
  date.time.field = "week",
  postcode.field = "postcode"
)
```

### Arguments

| | |
|---|---|
| case.df | A data.frame of events. |
| save.on.dir | logical. If TRUE then the vector is saved in 'baseline_matrix.Rdata' file. |
| date.time.field | A character string. |
| postcode.field | A character string. |

### Value

A Matrix.

### Examples

```
baseline.matrix = CreateBaselineMatrix(case.df)
baseline.matrix = CreateBaselineMatrix(case.df, date.time.field = 'SAMPLE_DT_numeric', postcode.field = 'Patient
```

## CreateCylinders *Create cylinders.*

### Description

Create cylinders to cover events encoded in observation.matrix and evaluate their exceedances with respect to baseline. If observation.matrix and baseline have the same matrix dimension, the exceedances are computed calling compute. Otherwise the function this function assumes that the baseline is an expand.grid data.frame and exceedancies are computed with compute.from.tab.baseline.

### Usage

```
CreateCylinders(
  observation.matrix,
  baseline,
  week.range,
  n.cylinders = 1000,
  p.val.threshold = 0.05,
  size_factor = 1
)
```

### Arguments

| | |
|---|---|
| observation.matrix | A 2D Matrix or sparseMatrix object. |
| baseline | A 2D matrix or a Nx4 expand.grid data frame. |
| week.range | A numeric vector to set the lower and upper limit to the heigh of the cylinders. |
| n.cylinders | An integer, total number of drawn cylinders. |
| p.val.threshold | A numeric. If the probability of observed exceedance is < p.val.threshold, flag the cylinder as anomalous. |
| size_factor | A numeric multiplier to increase or reduce the cylinder heights and radia. |

### Examples

```
CreateCylinders(observation.matrix, baseline.matrix, week.range = c(0,99), n.cylinders = 10000)
CreateCylinders(observation.matrix, baseline.tab, week.range = c(0,99), n.cylinders = 100)
```

---

## CreateCylinders.delay *n.cylinders=10000 takes around 3 hours for the whole dataset, to end up with 300 non-empty cylinders observation.matrix and baseline.matrix have dimension This function scans the matrices -1 in observation.matrix index means that we are excluding from week NA*

---

### Description

n.cylinders=10000 takes around 3 hours for the whole dataset, to end up with 300 non-empty cylinders observation.matrix and baseline.matrix have dimension This function scans the matrices -1 in observation.matrix index means that we are excluding from week NA

### Usage

```
CreateCylinders.delay(
  observation.matrix.typed,
  baseline.matrix.typed,
  observation.matrix.untyped,
  baseline.matrix.untyped,
```

```
  emmtype,
  week.range,
  n.cylinders = 1000,
  p.val.threshold = 0.05,
  coord.df = postcode2coord,
  size_factor = 1
)
```

## Arguments

| | |
|---|---|
| observation.matrix.untyped | A 2D matrix. |
| baseline.matrix.untyped | A 2dD matrix. |
| week.range | A numeric vector to set the lower and upper limit to the heigh of the cylinders. |
| n.cylinders | An integer, total number of drawn cylinders. |
| p.val.threshold | A numeric. If the probability of observed exceedance is < p.val.threshold, flag the cylinder as anomalous. |
| size_factor | A numeric multiplier to increase or reduce the cylinder heights and radia. |

---

CreateObservationMatrices *CreateObservationMatrices*

---

## Description

Compute 2D Sparse matrices enconding observations recorded in case.df. Rows are locations (indexed by postcodes) and columns are time steps. If a character vector of length N is passed as argument types, then N matrices are created only for the events with matching case.df$types. The matrices are saved on disk as '*observation_matrix.Rdata' files.

## Usage

```
CreateObservationMatrices(
  case.df,
  types = NULL,
  date.time.field = "week",
  postcode.field = "postcode"
)
```

## Arguments

| | |
|---|---|
| case.df | A data.frame of events. |
| types | NULL or a character vector. |
| date.time.field | A character string. |
| postcode.field | A character string. |

## Value

None

## Examples

```
CreateObservationMatrices(case.df)
CreateObservationMatrices(case.df, types=c("1.0", "33.0"), date.time.field = "SAMPLE_DT_numeric", postcode.field
```

---

`EmmtypeFactor.delay_` *At a given week, a fraction lambda_untyped \approx 0.6 of all cases are not typed. the baselines e.g. are as follows: - for the emmtype 33.0: (1-lambda_untyped) * lambda_33.0 * lambda_t * lambda_geo - for the entyped: lambda_untyped * lambda_t * lambda_geo*

---

## Description

At a given week, a fraction lambda_untyped \approx 0.6 of all cases are not typed. the baselines e.g. are as follows: - for the emmtype 33.0: (1-lambda_untyped) * lambda_33.0 * lambda_t * lambda_geo - for the entyped: lambda_untyped * lambda_t * lambda_geo

## Usage

```
EmmtypeFactor.delay_(case.file, starting.week, n.weeks)
```

---

`f_radia_and_heights` *Find optimal radia*

---

## Description

This function compute the optimal cylinder radia, such that the corresponding cylinders contain one event in average for the chosen N heights (`heights`) and a given `baseline.matrix`. It returns an Nx2 `Matrix` whose first column contains the heights and and the second column contains the corresponding radia.

## Usage

```
f_radia_and_heights(baseline.matrix, heigths = 1:100)
```

## Arguments

| | |
|---|---|
| `baseline.matrix` | A `Matrix` enconding the baseline. |
| `heigths` | `integer`. |

## Value

A Matrix.

## Examples

```
radia_and_heights = f_radia_and_heights(baseline.matrix, 1:10)
```

---

f_radia_and_heights_ *Find optimal radia*

---

## Description

This function compute the optimal cylinder radia, such that the corresponding cylinders contain one event in average for the chosen N heights (heights) and a given tabulated baseline. It returns an Nx2 Matrix whose first column contains the heights and and the second column contains the corresponding radia.

## Usage

```
f_radia_and_heights_(baseline.tab, heights = 1:100)
```

## Arguments

| | |
|---|---|
| baseline.matrix | An expand.grid tab enconding the baseline. |
| heigths | integer. |

## Value

A Matrix.

## Examples

```
radia_and_heights = f_radia_and_heights(baseline.matrix, 1:10)
```

## GB.region.boundaries *GB boundaries.*

### Description

A dataset containing the boundaries of the Euro constituencies of all Great Britain, Coordinate reference systems (CRS) EPSG code 4326, which uses units of longitude and latitude on the World Geodetic System 1984 (WGS84) ellipsoid. This information is license under the Open Government Licence v3.0, see https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/.

### Usage

```
GB.region.boundaries
```

### Format

Simple feature collection.

### Source

https://osdatahub.os.uk/

## lambda *Lambda Seasonal model for the intensity function,* $\lambda(t) = a + b + b\sin(c\,2\pi/365 + x2\,\pi/365)$.

### Description

Lambda

Seasonal model for the intensity function, $\lambda(t) = a + b + b\sin(c\,2\pi/365 + x2\,\pi/365)$.

### Usage

```
lambda(x, params)
```

### Arguments

| | |
|---|---|
| x | numeric vector of time points. |
| params | numeric vector of parameters of length = 4. |

### Value

a numeric vector of the same length as x.

### Examples

```
lambda(c(1,2,3,4,5,6), c(1,1,1,1) )
```

`make_circle` *centers: the data frame of centers with ID radius: radius measured in kilometer*

## Description

centers: the data frame of centers with ID radius: radius measured in kilometer

## Usage

```
make_circle(x0, y0, radius, n.points = 100)
```

`neg.log.like` *Negative log-likelihood*

## Description

$l(\lambda) = \log L(\lambda) = -\int_0^T \lambda(x)dx + \sum_{i=1}^n \log \lambda(x_i)$

## Usage

```
neg.log.like(params, data)
```

## Arguments

| | |
|---|---|
| `params` | A `numeric` vector of parameters of length = 4. |
| `data` | A `numeric` vector of event times. |

## Value

`numeric`.

## Examples

```
neg.log.like(c(1,1,1,1),c(1,2,3,4,5,6))
```

`plotCircle` *centers: the data frame of centers with ID radius: radius measured in kilometer*

## Description

centers: the data frame of centers with ID radius: radius measured in kilometer

## Usage

```
plotCircle(cylinders, n.points = 100, add = T, ...)
```

## postcode.data *UK postcode location and population.*

### Description

A dataset containing the population and the centroid coordinates of all UK postcodes.

### Usage

```
postcode.data
```

### Format

A data frame with 1048575 rows and 7 variables:

**postcode**

**Total**

**latitude**

**longitude**

### Source

https://www.ons.gov.uk

## postcode.in.england *postcode.in.england*

### Description

Check if postcode is in England.

### Usage

```
postcode.in.england(x, postcode.field = "postcode")
```

### Arguments

| | |
|---|---|
| x | integer. number of cylinder samples. |
| postcode.field | numeric. |

### Value

A (data.frame).

## postcode.to.location.and.population *postcode.to.location.and.population*

### Description

Find geo coordinates and population from postcode data frame.

### Usage

```
postcode.to.location.and.population(x, postcodes, postcode.field = "postcode")
```

### Arguments

| | |
|---|---|
| x | integer. number of cylinder samples. |
| postcodes | numeric. |
| postcode.field | numeric. |

### Value

A (data.frame).

## postcode.to.location2 *postcode.to.location2*

### Description

postcode.to.location2

### Usage

```
postcode.to.location2(x, postcode.field = "postcode")
```

### Arguments

| | |
|---|---|
| x | integer. number of cylinder samples. |
| postcode.field | numeric. |

### Value

A (data.frame).

## postcode.to.location3 *postcode.to.location3*

### Description

postcode.to.location3

### Usage

```
postcode.to.location3(x)
```

### Arguments

| | |
|---|---|
| x | integer. number of cylinder samples. |

### Value

A (data.frame).

## postcode.to.region *postcode.to.region*

### Description

postcode.to.region

### Usage

```
postcode.to.region(x, Area2Region_list)
```

### Arguments

| | |
|---|---|
| x | integer. number of cylinder samples. |
| Area2Region_list | numeric. |

### Value

A (data.frame)

PostcodeMap — *Map postcodes to coordinates Returns and save is 'postcode2coord.Rdata' a data frame that maps the postcodes included in* `rownames(matrix)` *to geographical coordinates.* `matrix` *can be a* `Matrix` *or a* `sparseMatrix` *object storing baseline or observation data. Requires the all postcode data tabulated in a* `data.frame` *called* `postcode.data` *available in the workspace.*

## Description

Map postcodes to coordinates Returns and save is 'postcode2coord.Rdata' a data frame that maps the postcodes included in `rownames(matrix)` to geographical coordinates. `matrix` can be a `Matrix` or a `sparseMatrix` object storing baseline or observation data. Requires the all postcode data tabulated in a `data.frame` called `postcode.data` available in the workspace.

## Usage

```
PostcodeMap(matrix, postcode.field = "postcode")
```

## Arguments

| | |
|---|---|
| `matrix` | A `Matrix` or `sparseMatrix`. |

## Value

A `data.frame`.

## Examples

```
postcode2coord = PostcodeMap(observation.matrix)
```

predict.cmle — *Lambda Seasonal model for the intensity function,* $\lambda(t) = a + b + b\sin(c\,2\pi/365 + x2\,\pi/365)$.

## Description

Lambda

Seasonal model for the intensity function, $\lambda(t) = a + b + b\sin(c\,2\pi/365 + x2\,\pi/365)$.

## Usage

```
## S3 method for class 'cmle'
predict(x, params)
```

### Arguments

| | |
|---|---|
| x | numeric vector of time points. |
| params | numeric vector of parameters of length = 4. |

### Value

a numeric vector of the same length as x.

### Examples

```
predict.cmle(c(1,2,3,4,5,6), c(1,1,1,1) )
```

---

## rcylinder *Draw random cylinder coordinates*

---

### Description

Find the coordinates (centers and height limits) of cylinders that contain events defined in observation.matrix, with radia and heights given in radia_and_heights.

### Usage

```
rcylinder(
  n.cylinders,
  observation.matrix,
  time.range,
  radia_and_heights,
  postcode2coord
)
```

### Arguments

| | |
|---|---|
| n.cylinders | An integer; the number of cylinders to draw. |
| observation.matrix | A sparseMatrix object enconding the events. |
| time.range | An integer vector. |
| radia_and_heights | A Matrix. |
| postcode2coord | A data.frame that maps the rows of observation.matrix to geographical coordinates. |

### Value

A (data.frame).

### Examples

```
cylinders=rcylinder(10, observation.matrix, time.range, radia_and_heights, postcode2coord)
```

## tabulated.baseline *tabulated.baseline*

### Description

tabulate the baseline intensity function.

### Usage

```
tabulated.baseline(case.df, date.time.field = "week")
```

### Arguments

| | |
|---|---|
| case.df | A data.frame of events. |
| date.time.field | A character string. |

## tabulated.baseline2 *tabulated.baseline*

### Description

tabulate the baseline intensity function.

### Usage

```
tabulated.baseline2(case.df, date.time.field = "week")
```

### Arguments

| | |
|---|---|
| x | integer. number of cylinder samples. |
| postcode.field | numeric. |

`TimeFactor` *Find temporal component of the baseline*

## Description

Returns a vector representing the temporal component of the baseline. This is obtained by calling `cmle` and `predict.cmle`, which fit a seasonal trend model to aggregated data and return its best estimate, respectively.

## Usage

```
TimeFactor(
  case.df,
  save.on.dir = TRUE,
  get.from.dir = FALSE,
  date.time.field = "week",
  start = NULL,
  n.iterations = 20
)
```

## Arguments

| | |
|---|---|
| `case.df` | A `data.frame` containing the events. |
| `save.on.dir` | `logical`. If TRUE then the vector is saved in 'timefactor.Rdata' file. |
| `get.from.dir` | `logical`. If TRUE then the vector is obtained from the 'timefactor.Rdata' file. |
| `date.time.field` | A `character` string. |
| `start` | `numeric`. Starting parameters (not yet implemented). |
| `n.iterations` | An `integer`. |

## Value

A `numeric` vector.

## Examples

```
time.factor = TimeFactor(case.df)
```

## UpdateBaselineMatrix *Update baseline matrix*

### Description

Create a new baseline matrix to match the new observation matrix. This function is very similar to UpgradeBaselineMatrix does not perform fit if time.factor parameters were saved, only create a baseline matrix to match the new observation matrix.

### Usage

```
UpdateBaselineMatrix(
  previous.baseline.matrix,
  recent.case.df,
  save.on.dir = FALSE,
  date.time.field = "week",
  postcode.field = "postcode"
)
```

### Arguments

| | |
|---|---|
| save.on.dir | logical. If TRUE then the vector is saved in 'baseline_matrix.Rdata' file. |
| date.time.field | A character string. |
| postcode.field | A character string. |

### Value

A 2D matrix

## UpdateObservationMatrices *Upgrade observation matrix*

### Description

Create a new observation matrix and overwrite the old one

### Usage

```
UpdateObservationMatrices(
  case.df.old,
  case.df.new,
  types = NULL,
  date.time.field = "week",
  postcode.field = "postcode"
)
```

### Arguments

| | |
|---|---|
| types | NULL or a character vector. |
| date.time.field | A character string. |
| postcode.field | A character string. |

---

UpdateTimeFactor *Update Time factor*

---

### Description

Returns a vector representing the temporal component of the baseline.

### Usage

```
UpdateTimeFactor(
  case.df,
  save.on.dir = TRUE,
  get.from.dir = FALSE,
  date.time.field = "week",
  parameters = NULL,
  n.iterations = 20
)
```

### Arguments

| | |
|---|---|
| case.df | A data.frame containing the events. |
| save.on.dir | logical. If TRUE then the vector is saved in 'timefactor.Rdata' file. |
| get.from.dir | logical. If TRUE then the vector is obtained from the 'timefactor.Rdata' file. |
| date.time.field | A character string. |
| n.iterations | An integer. |

### Value

A vector

### Examples

```
time.factor = TimeFactor(case.df)
```

`UpgradeBaselineMatrix` *Upgrade baseline matrix*

## Description

Create a new baseline matrix to match the new observation matrix and makes new fit for the temporal factor.

## Usage

```
UpgradeBaselineMatrix(
  previous.baseline.matrix,
  case.df.old,
  case.df.new,
  save.on.dir = FALSE,
  date.time.field = "week",
  postcode.field = "postcode",
  n.iterations = 10
)
```

## Arguments

| | |
|---|---|
| `save.on.dir` | `logical`. If TRUE then the vector is saved in 'baseline_matrix.Rdata' file. |
| `date.time.field` | A `character` string. |
| `postcode.field` | A `character` string. |

## Value

A 2D matrix

# Index