Package 'diseasemapping'

March 10, 2021

Type Package

Title Modelling Spatial Variation in Disease Risk for Areal Data

Version 1.5.0

Date 2021-03-09

Depends R (>= 3.5.0)

Imports stats, utils, sp, methods

Suggests spdep, mgcv, geostatsp, mapmisc (>= 1.0), knitr

Enhances INLA, XML

Additional_repositories https://inla.r-inla-download.org/R/testing

Author Patrick Brown <`patrick.brown@utoronto.ca`>, Lutong Zhou

Maintainer Patrick Brown <`patrick.brown@utoronto.ca`>

Description Formatting of population and case data, calculation of Standardized
Incidence Ratios, and fitting the BYM model using 'INLA'. For de-
tails see Brown (2015) <doi:10.18637/jss.v063.i12>.

License GPL

VignetteBuilder knitr

Repository R-Forge

Repository/R-Forge/Project diseasemapping

Repository/R-Forge/Revision 2626

Repository/R-Forge/DateTimeStamp 2021-03-10 17:42:52

Date/Publication 2021-03-10 17:42:52

R topics documented:

`diseasemapping-package` *Disease Mapping*

Description

Functions for calculating observed and expected counts by region, and manipulating posterior samples from Bayesian models produced by glmmBUGS.

Author(s)

Patrick Brown

Examples

```
# creating SMR's
data('kentucky')

if(FALSE) {
# must have an internet connection to do the following
larynxRates= cancerRates("USA", year=1998:2002,site="Larynx")
# get rid of under 10's
larynxRates = larynxRates[-grep("_(0|5)$",names(larynxRates))]
dput(larynxRates)
} else {
larynxRates = structure(c(0, 0, 0, 0, 1e-06, 6e-06, 2.3e-05, 4.5e-05, 9.9e-05,
0.000163, 0.000243, 0.000299, 0.000343, 0.000308, 0.000291, 0.000217,
0,0, 0, 0, 1e-06, 1e-06, 3e-06, 8e-06, 1.3e-05, 2.3e-05, 3.5e-05,
5.8e-05, 6.8e-05, 7.5e-05, 5.5e-05, 4.1e-05, 3e-05), .Names = c("M_10",
"M_15", "M_20", "M_25", "M_30", "M_35", "M_40", "M_45", "M_50",
"M_55", "M_60", "M_65", "M_70", "M_75", "M_80", "M_85", "F_0", "F_10",
"F_15", "F_20", "F_25", "F_30", "F_35", "F_40", "F_45", "F_50",
"F_55", "F_60", "F_65", "F_70", "F_75", "F_80", "F_85"))

}

kentucky2 = getSMR(kentucky, larynxRates, larynx,
regionCode="County")

if(require('mapmisc', quietly=TRUE)) {
mycol = colourScale(kentucky2$SMR, breaks=9,
dec=-log10(0.5), style='equal', transform='sqrt')
plot(kentucky2, col=mycol$plot)
legendBreaks('topleft', mycol)
}

if( require("spdep", quietly=TRUE) & require("INLA", quietly=TRUE)) {
```

```
kBYM = bym(observed ~ offset(logExpected) + poverty, kentucky2,
 priorCI = list(sdSpatial=c(0.1, 5), sdIndep=c(0.1, 5)),
 control.mode=list(theta=c(3.52, 3.35),restart=TRUE))

kBYM$par$summary

}

# an example of a spatial model with glmmBUGS

## Not run:
# run the model
library('spdep')
popDataAdjMat = poly2nb(ontario,row.names=as.character(ontario[["CSDUID"]]))

library('glmmBUGS')
forBugs = glmmBUGS(formula=observed + logExpected ~ 1,
  effects="CSDUID", family="poisson", spatial=popDataAdjMat,
  data=ontario@data)
startingValues = forBugs$startingValues
source("getInits.R")
library('R2WinBUGS')
ontarioResult = bugs(forBugs$ragged, getInits, parameters.to.save = names(getInits()),
    model.file="model.bug", n.chain=3, n.iter=100, n.burnin=10, n.thin=2,
      program="winbugs", debug=TRUE)

ontarioParams = restoreParams(ontarioResult, forBugs$ragged)
ontarioSummary = summaryChain(ontarioParams)

# merge results back in to popdata
ontario = mergeBugsData(ontario, ontarioSummary)

## End(Not run)


# running the same thing with INLA
## Not run:
library('INLA')
# get rid of regions with no neighbours
ontario2 = ontario[! as.character(ontario$CSDUID)
c("3510005" ,"3501007", "3537001", "3551031", "3560065", "3560062"),]
popDataAdjMat2 = poly2nb(ontario2,
row.names=as.character(ontario2[["CSDUID"]]))
nb2INLA("nb.graph",popDataAdjMat2)

ontario2$CSDUID = as.character(ontario2$CSDUID)

prior.iid=prior.besag=c(.42,.00015)
formula.bym = observed ~  f(CSDUID,
    model = "bym", graph = "nb.graph", values=CSDUID ,
    param = c(prior.iid, prior.besag))

result1.bym = inla(formula.bym,family="poisson",data=ontario2@data,
offset=ontario2@data$logExpected,
    verbose=FALSE, keep = TRUE,
    control.predictor=list(compute=TRUE))


tomerge = result1.bym$summary.random$CSDUID
rownames(tomerge) = tomerge$ID

ontario2@data =   cbind(ontario2@data,
tomerge[as.character(ontario2$CSDUID),] )

require('mapmisc')
mycol = colourScale(ontario2$mean, breaks=9,
dec=1, style='equal', transform='sqrt')
plot(ontario2, col=mycol$plot)
legendBreaks('topleft', mycol)


## End(Not run)
```

bym-methods *Fit the BYM model*

Description

Uses inla to fit a Besag, York and Mollie disease mapping model

Usage

```
## S4 method for signature 'formula,ANY,ANY,missing'
bym(formula,data,adjMat,region.id,...)
## S4 method for signature 'formula,ANY,missing,missing'
bym(formula,data,adjMat,region.id,...)
## S4 method for signature 'formula,SpatialPolygonsDataFrame,NULL,character'
bym(formula, data, adjMat, region.id, ...)
## S4 method for signature 'formula,SpatialPolygonsDataFrame,missing,character'
bym(formula, data, adjMat, region.id, ...)
## S4 method for signature 'formula,SpatialPolygonsDataFrame,nb,character'
bym(formula,data,adjMat,region.id,...)
## S4 method for signature 'formula,data.frame,nb,character'
bym(
formula,data,adjMat,region.id,
priorCI=list(sdSpatial=c(0.01,2),sdIndep=c(0.01,2)),
family="poisson",formula.fitted=formula,...)
```

Arguments

`formula`	model formula, defaults to intercept-only model suitable for output from `getSMR` if `data` is a `SpatialPolygonsDataFrame`.
`data`	The observations and covariates for the model, can be output from `getSMR`.
`adjMat`	An object of class `nb` containing the adjacency matrix. If not supplied it will be computed from `data`, but is required if `data` is a `SpatialPolygonDataFrame`

`region.id`	Variable in `data` giving identifiers for the spatial regions. If not supplied, row numbers will be used.
`priorCI`	named list of vectors specifying priors, see Details
`family`	distribution of the observations, defaults to `poisson`
`formula.fitted`	formula to use to compute the fitted values, defaults to the model formula but may, for example, exclude individual-level covariates.
`...`	Additional arguments passed to `inla`in the `INLA`package, such as `control.inla`

Details

The Besag, York and Mollie model for Poisson distributed case counts is:

$$Y_i \sim Poisson(O_i \lambda_i)$$

$$\log(\mu_i) = X_i \beta + U_i$$
$$U_i \sim BYM(\sigma_1^2, \sigma_2^2)$$

- Y_i is the response variable for region i, on the left side of the `formula` argument.
- O_i is the 'baseline' expected count, which is specified in `formula` on the log scale with $\log(O_i)$ an `offset` variable.
- X_i are covariates, on the right side of `formula`
- U_i is a spatial random effect, with a spatially structured variance parameter σ_1^2 and a spatially independent variance σ_2^2.

The `priorCI` argument can be a `list` containing elements named `sdSpatial` and `sdIndep`, each being a vector of length 2 with 2.5pct and 97.5pct quantiles for the prior distributions of the standard deviations σ_1 and σ_2 respectively. Gamma prior distributions for the precision parameters $1/\sigma_1^2$ and $1/\sigma_2^2$ yielding quantiles specified for the standard deviations are computed, and used with the `model="bym"` option to `f`.

The other possible format for `priorCI` is to have elements named `sd` and `propSpatial`, which specifies `model="bym2"` should be used with penalized complexity priors. The `sd` element gives a prior for the marginal standard deviation $\sigma_0 = \sqrt{\sigma_1^2 + \sigma_2^2}$. This prior is approximately exponential, and `priorCI$sd = c(1,0.01)` specifies a prior probability $pr(\sigma_0 > 1) = 0.01$. The `propSpatial` element gives the prior for the ratio $\phi = \sigma_1/\sigma_0$. Having `priorCI$propSpatial = c(0.5,0.9)` implies $pr(\phi < 0.5) = 0.9$.

Value

A list containing

`inla`	results from the call to `inla`. Two additional elements are added: `marginals.bym` for the marginal distributions of the spatial random effects, and `marginals.fitted.bym` for the marginals of the fitted values.
`data`	A `data.frame` or `SpatialPolygonsDataFrame` containing posterior means and quantiles of the spatial random effect and fitted values.
`parameters`	Prior and posterior distributions of the two covariance parameters, and a table summary with posterior quantiles of all model parameters.

Author(s)

Patrick Brown

See Also

https://www.r-inla.org, inla glgm, getSMR

Examples

```
data('kentucky')

# must have an internet connection to do the following
## Not run:
larynxRates= cancerRates("USA", year=1998:2002,site="Larynx")
dput(larynxRates)

## End(Not run)

larynxRates = structure(c(0, 0, 0, 0, 0, 0, 1e-06, 6e-06, 2.3e-05, 4.5e-05,
9.9e-05, 0.000163, 0.000243, 0.000299, 0.000343, 0.000308, 0.000291,
0.000217, 0, 0, 0, 0, 0, 1e-06, 1e-06, 3e-06, 8e-06, 1.3e-05,
2.3e-05, 3.5e-05, 5.8e-05, 6.8e-05, 7.5e-05, 5.5e-05, 4.1e-05,
3e-05), .Names = c("M_0", "M_5", "M_10", "M_15", "M_20", "M_25",
"M_30", "M_35", "M_40", "M_45", "M_50", "M_55", "M_60", "M_65",
"M_70", "M_75", "M_80", "M_85", "F_0", "F_5", "F_10", "F_15",
"F_20", "F_25", "F_30", "F_35", "F_40", "F_45", "F_50", "F_55",
"F_60", "F_65", "F_70", "F_75", "F_80", "F_85"),
site = "Larynx", area = "USA, SEER", year = "1998-2002")

# get rid of under 10's
larynxRates = larynxRates[-grep("_(0|5)$",names(larynxRates))]

kentucky = getSMR(kentucky, larynxRates, larynx, regionCode="County")

if( require("spdep", quietly=TRUE)) {

kBYM = bym(observed ~ offset(logExpected) + poverty, kentucky,
 priorCI = list(sdSpatial=c(0.1, 5), sdIndep=c(0.1, 5)),
 control.mode=list(theta=c(3.52, 3.35),restart=TRUE))

kBYM$par$summary

if(requireNamespace('geostatsp', quietly=TRUE))
kBYM$data$exc1 = geostatsp::excProb(
kBYM$inla$marginals.fitted.bym, log(1.2)
)
}  else {
kBYM = list()
}




if(require('mapmisc', quietly=TRUE) & length(kBYM$data$fitted.exp)){

thecol = colourScale(kBYM$data$fitted.exp,
breaks=5, dec=1, opacity = 0.7)

map.new(kBYM$data)
## Not run:
kmap = openmap(kBYM$data)
plot(kmap,add=TRUE)

## End(Not run)

plot(kBYM$data, col=thecol$plot,add=TRUE)
legendBreaks("topleft", thecol)

}
```

---

cancerRates — *Download cancer incidence rates from the International Agency for Research on Cancer (IARC)*

Description

Rates by age and sex group are retreived from http://ci5.iarc.fr/CI5plus/ci5plus.htm

Usage

```
cancerRates(area = "canada", year=2000, sex=c("M", "F"), site="Lung")
```

Arguments

`area`	Region to retrieve rates from,
`year`	year or sequence of years to retrieve data from, within the period 1978 to 2002
`site`	a vector of cancer sites, see details
`sex`	`"M"` or `"F"` for male or female rates only, `c("M","F")` (the default) for both sexes.

Details

`area` must be one of Canada, Norway, Latvia, Lithuania, Iceland, Finland, Estonia, Denmark, "Czech Republic", "Costa Rica", USA, Iowa, "New Mexico" or the Canadian provinces of Newfoundland, Prince Edward Island, Nova Scotia, Ontario, Manitoba, Saskatchewan, Alberta, and British Columbia. Alternately an integer specifying a registry code from http://ci5.iarc.fr.

`site` must be one or more of All Sites, Oral Cavity and Pharynx, Oesophagus. Stomach, Colon, Rectum and Anus, Liver, Gallbladder, Pancreas, Larynx, Lung, Bone, Melanoma of skin, Prostate **(Males only)**, Testis **(Males only)**, Breast **(Females only)**, Cervix uteri **(Females only)**, Corpus uteri **(Females only)**, Ovary and other uterine adnexa **(Females only)**, Kidney, Bladder, Eye, Brain and Central Nervous System, Thyroid, Non-Hodgkin Lymphoma, Hodgkin Lymphoma, Multiple myeloma, Leukaemia.

Value

vector of cancer rates, by age and sex group

Examples

```
# won't work if offline or if the iarc web site is down

if(interactive() | Sys.info()['user'] =='patrick') {
  qcLungF=cancerRates(area="canada",
   year=2001:2002, site="lung", sex="F")
} else {
qcLungF = structure(c(0, 5e-06, 0, 0, 5e-06, 1e-05, 0, 3.4e-05, 9.6e-05,
0.000211, 0.000559, 0.001289, 0.002003, 0.002508, 0.002728, 0.003189,
0.002792, 0.001905), .Names = c("F_0", "F_5", "F_10", "F_15",
"F_20", "F_25", "F_30", "F_35", "F_40", "F_45", "F_50", "F_55",
"F_60", "F_65", "F_70", "F_75", "F_80", "F_85"), site = "Lung",
area = "Canada", year = "2001-2002")
}
qcLungF

data('popdata')

qcLungExp = getSMR(popdata, qcLungF)

names(qcLungExp)

if(require('mapmisc', quietly=TRUE)) {

mycol = colourScale(qcLungExp$expected,
breaks=12, dec=0, style='quantile')
plot(popdata[1:400,])
plot(qcLungExp, col=mycol$plot, border='#00000040',add=TRUE)
legendBreaks('topright', mycol)
}
```

`casedata` *Data set contains the number of cases information*

Description

Cases of Hepatitis Z in Ontario.

Usage

```
data(casedata)
```

Format

data frame

Details

This dataset refers to cases of Hepatitis Z in Ontario for the years 1916 to 1918, giving the number of cases in each census subdivision by age, sex and year. For reasons of privacy, any counts between 1 and 5 have been changed to 1.

Examples

```
data(casedata)
head(casedata)
table(casedata$cases)
tapply(casedata$cases, casedata$age, sum)

## maybe str(casedata) ; plot(casedata) ...
```

`formatCases` *Format the disease case data set*

Description

The formatCases funtion formats the case data set. Changes other formats of age and sex group to three columns: age, ageNumeric and sex.

Usage

```
formatCases(casedata, ageBreaks = NULL, years = NULL, aggregate.by = NULL)
```

Arguments

`casedata`	disease cases data set, usually a data.frame which contains age and sex and number of cases.
`ageBreaks`	results from `getBreaks` function.
`years`	if it contains multiple years, define which years will be included in.
`aggregate.by`	if want to view the data set from a macro way, could aggregate the data set by age or sex or other variables.

Details

After using formatCases function, the age columns will change to a "character" column contains ages in `cut` format, i.e [50,55), denotes age 50. The cut breaks can be found from the breaks of the population data set or defined by user. The original "age" column will changed to "ageNumeric" columns as factors. The sex column will have two levels "M" and "F" as factors. If "aggregate.by" is not NULL, the number of cases will be sum up by the groups defined in `aggregate.by` argument.

Value

formatCases function will return a data frame.

Author(s)

Patrick Brown

Examples

```
data('casedata')
data('popdata')
head(casedata)
caseformat <- formatCases(casedata, ageBreaks = getBreaks(names(popdata@data)))
head(caseformat)
caseformatagg <- formatCases(casedata, ageBreaks = getBreaks(names(popdata@data)),
  aggregate.by=c("age", "sex"))
head(caseformatagg)
```

---

formatPopulation-methods
Format a population data set

Description

The formatCases funtion formats the population data set. Reshape the population data set to "long" format, add in 4 columns : GROUP, POPULATION, sex and age.

Usage

```
## S4 method for signature 'data.frame'
formatPopulation(
popdata, aggregate.by = NULL, breaks = NULL, ...
)
## S4 method for signature 'SpatialPolygonsDataFrame'
formatPopulation(
popdata, aggregate.by = NULL, breaks = NULL, ...
)
## S4 method for signature 'list'
formatPopulation(
popdata, aggregate.by = NULL, breaks = NULL,
years=as.integer(names(popdata)), year.range=NULL,
 time="YEAR",
        personYears=TRUE,...
)
```

Arguments

popdata	population data set. It can be a data frame, list, database connection, or spatial polygon data frame
aggregate.by	if want to view the data set from a macro way, could aggregate the data set by age or sex or other variables
breaks	age breaks the user want to use. i.e breaks = c(10, 20, 30 ,40, 60, Inf).
time	the time variable, i.e years
personYears	convert populations to person-years
years	a vector with the year of each dataset
year.range	two dimensional vector with first and last year
...	additional arguments

Details

After using the `formatPopulation` function, it will return the population data set in the same class as the original data set. i.e if a spatial polygon data frame has been put into the `formatPopulation` function, it will return a spatial polygon data frame. If `aggregate.by` is not NULL, the number of cases will be sum up by the groups define in aggregate.by. The "Group" column contains information of sex and age groups,in the format of M.55, denotes male, year 55. The "POPULATION" column is a numeric column, denotes the size of population for the particular age and sex group. The "age" column will be a "character" column contains ages in a cut format. i.e [50,55), denotes age 50. The cut breaks will get from the breaks of population data set or define by user. The sex column will have two levels "M" and "F" as factors.

Note

If `breaks` is not specified, the function will aggregate by "age" as default.

Author(s)

Patrick Brown

Examples

```
data('kentucky')
head(kentucky@data)
poptry <- formatPopulation(kentucky, breaks = c(seq(0, 80, by=10), Inf))
head(poptry)
poptryagg <- formatPopulation(kentucky, breaks = c(seq(0, 80, by=10), Inf),
  aggregate.by=c("sex", "age"))
head(poptryagg)
```

`getBreaks` *Age Breaks*

Description

An internal function to return a list contains age breaks, ages in the population data set, sex in the population data set, and age sex groups will be used in the `formatPopulation` function.

Usage

```
getBreaks(colNames, breaks = NULL)
```

Arguments

`colNames`	names from the population data set
`breaks`	the age breaks, i.e breaks =seq(0, 80, by= 10)

Examples

```
data('kentucky')
ageBreaks = getBreaks(names(kentucky), breaks=c(seq(0, 80, by=10), Inf))
ageBreaks
```

getRates — *Calculate the estimated coefficients of age and sex group from the glm model*

Description

The getRates function calculates the estimated coefficient of the age and sex group from the case and population data set. It fits a glm model with Poisson distribution by default.

Usage

```
getRates(casedata, popdata, formula, family = 'poisson', minimumAge = 0,
    maximumAge = 100, S = c("M", "F"), years = NULL, year.range = NULL,
    case.years = grep("^year$", names(casedata), ignore.case = TRUE,
        value = TRUE), fit.numeric=NULL, breaks = NULL)
```

Arguments

`casedata`	A data frame of case data, with columns corresponding to variables in `formula`. Assumed to be one row per case, unless a column called `y` or `cases` or `count` is included, in which case this column gives the number of cases per row.
`popdata`	population data set
`formula`	the glm model you want to fit. ie. `~age*sex`
`family`	the distribution to fit the model
`minimumAge`	the lower boundary of the age, default is 0
`maximumAge`	the higher boundary of the age, default is 100
`S`	vector of sexes to include in the analysis. Defaults to both "M" and "F"
`years`	a vector of census years
`year.range`	study period: a vector of two elements, starting dates and ending dates
`case.years`	variable name in the case data which contains time
`fit.numeric`	the variables which needed to be changed from factor to numeric
`breaks`	the age breaks

Details

It fits a glm model with Poisson or binomial distribution over case and population data sets. If there is no data set in some age and sex group, an NA will show there.

Value

A summary of the glm model contains set of estimated coefficients for different age and sex groups.

Author(s)

Patrick Brown

Examples

```
data('casedata')
data('popdata')
therates = getRates(casedata, popdata, ~sex*age,
breaks=c(seq(0, 80, by=10), Inf))
therates
```

`getSMR-methods` *Calculate the standardized mortality/morbidity ratios*

Description

Calculates the rate of observe value over expected value. It will also merge back the observed value, expected value and the ratio back to the population data set.

Usage

```
## S4 method for signature 'SpatialPolygonsDataFrame,ANY,ANY,ANY,ANY'
getSMR(
popdata, model, casedata, regionCode , regionCodeCases ,
area.scale = 1,  sex=c('m','f'),...
)
## S4 method for signature 'list,ANY,ANY,ANY,ANY'
getSMR(
popdata, model, casedata, regionCode , regionCodeCases ,
area.scale=1, sex=c('m','f'), ...
)
## S4 method for signature 'data.frame,ANY,missing,missing,missing'
getSMR(
popdata, model, casedata, regionCode , regionCodeCases ,
area.scale = 1,  sex=c('m','f'),...
)

## S4 method for signature 'data.frame,ANY,data.frame,missing,missing'
getSMR(
popdata, model, casedata, regionCode , regionCodeCases ,
area.scale = 1,  sex=c('m','f'),...
)
## S4 method for signature 'data.frame,ANY,data.frame,character,missing'
getSMR(
popdata, model, casedata, regionCode , regionCodeCases ,
area.scale = 1,  sex=c('m','f'),...
)
## S4 method for signature 'data.frame,ANY,missing,character,missing'
getSMR(
popdata, model, casedata, regionCode , regionCodeCases ,
area.scale = 1,  sex=c('m','f'),...
)

## S4 method for signature 'data.frame,ANY,data.frame,character,character'
```

```
getSMR(
popdata, model, casedata, regionCode , regionCodeCases ,
area.scale = 1,  sex=c('m','f'),...
)
```

Arguments

`popdata`	the name of population data set
`model`	rates, either fitted model (usually a `glm` object), or a vector of rates.
`casedata`	the name of case data set
`regionCode`	the name of district area column in population data set
`regionCodeCases`	the name of district area column in case data set
`area.scale`	scale the unit of area. e.g $10^6$: if your spatial coordinates are metres and you want intensity in cases per km2
`sex`	possible subsetting of cases and population, set `sex='f'` for females only.
`...`	additional arguments. When `popdata` is a `list`, arguments can be `personYears` (logical, convert rates to person years), `years` (a vector with the year of each dataset), or `year.range` (two dimensional vector with first and last year)

Details

If `model` is numeric, it's assumed to be a vector of rates, with the names of the elements corresponding to columns of the population data set. Names do not need to match exactly (can have M in one set of names, male in another for instance).

Otherwise, `model` is passed to the `predict` function.

Value

Returns a new population data set contains expected number of cases, observed number of cases and SMR. It has the same format as the population data set which put into the function.

Examples

```
data(kentucky)

kentucky2 = getSMR(kentucky, larynxRates, larynx,
regionCode="County")

data.frame(kentucky2)[1:10,grep("^F|^M", names(kentucky2), invert=TRUE)]
```

getStdRate — *Calculate the standardized rate*

Description

A function to calculate the standard rate according to the Canadian standard population data set from year 1991.

Usage

```
getStdRate(relativeRate, model, referencePopulation, scale = 1e+05)
```

Arguments

`relativeRate`	the relative cancer rate calculated by glmmBUGS of different sex and age group of people from ontario .
`model`	Model to standardize to, either `glm` model output or a vector of rates by age and sex group
`referencePopulation`	population to standardize to
`scale`	compute the expected rate per 'scale' number of people.

Author(s)

Lutong Zhou

Examples

```
data(kentucky)

kentucky2 = getSMR(kentucky, larynxRates, larynx,
regionCode="County")

data(referencepop)
newpop <- getStdRate(kentucky2$SMR, larynxRates, referencepop, scale=100000)

newpop[1:10]
```

inla.models — *Valid models in INLA*

Description

calls the function of the same name in INLA

Usage

```
inla.models()
```

Value

a list

See Also

https://www.r-inla.org

`kentucky` *Larynx cancer cases and population in Kentucky*

Description

Data set contains the information of population, by age, sex, and census subdivision.

Usage

```
data('kentucky')
```

Format

A `SpatialPolygonsDataFrame` of Kentucky boundaries and populations, case numbers for each county, and a vector of cancer rates by age and sex group.

Details

`larynx` is a `data.frame` of cancer case counts by county, obtained from http://www.cancer-rates.info and are for a single deliberately unspecified year.

`kentucky` contains country boundaries and populations.

`kentuckyTract` contains census tract boundaries and populations.

Examples

```
data('kentucky')

head(larynx)
10^5*larynxRates[paste(c("M","F"), 50, sep="_")]


kentucky2 = getSMR(kentucky, larynxRates, larynx,
regionCode="County")

names(kentucky2)
length(kentucky2)

data('kentuckyTract')
length(kentuckyTract)

if(require('mapmisc', quietly=TRUE)) {
mycol = colourScale(kentucky2$SMR,
breaks=10, dec=-log10(0.5), style='quantile')
map.new(kentucky2)
plot(kentucky2, col=mycol$plot, border='#00000040',add=TRUE)
legendBreaks('topright', mycol)
} else {
plot(kentucky2)
}

breaks = c(0,1,seq(2, ceiling(max(kentucky2$SMR,na.rm=TRUE)),by=2))
thecol = terrain.colors(length(breaks)-1)

plot(kentucky2, col = thecol[cut(kentucky2$SMR,
breaks,include.lowest=TRUE)] )

legend("topleft", pch=15, pt.cex=2.5, adj=c(0,15),
  legend=rev(breaks), col=c(NA, rev(thecol)))

## Not run:
# the data were created with
larynxRates= cancerRates("USA", year=1998:2002,site="Larynx")

load(url("http://biogeo.ucdavis.edu/data/gadm2/R/USA_adm2.RData"))
kentucky = gadm[gadm$NAME_1 =="Kentucky",]

# population data
download.file(
"http://www.census.gov/popest/data/counties/asrh/2011/files/CC-EST2011-ALLDATA-21.csv",
destfile =
"/store/patrick/spatialData/C-EST2011-ALLDATA-21.csv")
# file layout
download.file(
"http://www.census.gov/popest/data/counties/asrh/2011/files/CC-EST2011-ALLDATA.pdf",
destfile = "/store/patrick/spatialData/kentuckyPopFormat.pdf")

kpop = read.table(
"/store/patrick/spatialData/C-EST2011-ALLDATA-21.csv",
header=TRUE,as.is=TRUE,sep=",")
kpop = kpop[kpop$YEAR==1 & kpop$AGEGRP != 0, ]
names(kpop) = gsub("^TOT_","", names(kpop))
names(kpop) = gsub("(EM)?ALE$","", names(kpop))

kpop$age = (kpop$AGEGRP-1)*5
kpop$County =  gsub(" County$", "", kpop$CTYNAME)
kpop = kpop[,c("County","age","M","F")]
kpop2 = reshape(kpop,direction="wide", idvar="County",
v.names=c("M","F"), timevar="age")
rownames(kpop2) = kpop2$County


# poverty
download.file(
paste(
"http://www.ers.usda.gov/ReportExport.aspx?reportPath=/State_Fact_Sheets/",
"PovertyReport&stat_year=2011&stat_type=0&fips_st=21&",
"exportType=EXCEL&exportName=PovertyReport",
sep=""),
destfile="/store/patrick/spatialData/poverty.xls")
library('gdata')
kpov = read.xls("/store/patrick/spatialData/poverty.xls",
header=TRUE,skip=3)
```

```
kpov = kpov[!is.na(kpov$Percent),c("FIPS.", "Name","Percent")]
rownames(kpov) = kpov$Name
kpop2$poverty = kpov[rownames(kpop2), "Percent"]

# merge population and spatial data
kdata = kpop2[kentucky$NAME_2,]
rownames(kdata) = NULL
kentucky = SpatialPolygonsDataFrame(
polygons(kentucky),
data=kdata,match.ID=FALSE)

larynx <- structure(
list(County = c("Hickman", "Caldwell", "Anderson",
"Marion", "Wayne", "Lincoln", "Livingston", "Montgomery", "Adair",
"Henderson", "Knox", "Martin", "Monroe", "Wolfe", "Breathitt",
"Fleming", "Woodford", "Garrard", "Bracken", "Barren", "Lawrence",
"Logan", "Clark", "Scott", "Madison", "Oldham", "Clay", "Russell",
"Shelby", "Letcher", "Campbell", "Graves", "Johnson", "Metcalfe",
"Pulaski", "Bullitt", "Knott", "Boyd", "Ohio", "Bath", "Butler",
"Todd", "Mercer", "Green", "Greenup", "Larue", "Calloway", "Webster",
"Morgan", "Pendleton", "Mason", "Hardin", "Lewis", "McCreary",
"Spencer", "Union", "Marshall", "Jessamine", "Henry", "Trigg",
"Pike", "Nelson", "Jefferson", "Floyd", "Bourbon", "McCracken",
"Boone", "Kenton", "Grayson", "Taylor", "Hopkins", "Boyle", "Meade",
"Fayette", "Daviess", "Harlan", "Warren", "Christian", "Magoffin",
"Carter", "Hart", "Lee", "Elliott", "Edmonson", "Crittenden",
"Leslie", "Laurel", "Cumberland", "Menifee", "Fulton", "Carlisle",
"McLean", "Owsley", "Carroll", "Estill", "Harrison", "Owen",
"Breckinridge", "Nicholas", "Bell", "Trimble", "Allen", "Rowan",
"Simpson", "Perry", "Powell", "Rockcastle", "Hancock", "Robertson",
"Franklin", "Washington", "Casey", "Clinton", "Lyon", "Muhlenberg",
"Ballard", "Gallatin", "Whitley", "Grant", "Jackson", "Breathitt",
"Nicholas", "Bracken", "Todd", "Magoffin", "Pendleton", "Metcalfe",
"Webster", "Leslie", "Henry", "Union", "Adair", "Casey", "Pike",
"Jessamine", "Nelson", "Garrard", "Pulaski", "Meade", "Harlan",
"Floyd", "Carter", "Shelby", "Barren", "Franklin", "Boyd", "Jefferson",
"Fayette", "Hopkins", "Kenton", "Warren", "Bullitt", "Knox",
"Butler", "Bourbon", "Elliott", "Johnson", "Estill", "Boone",
"Boyle", "Breckinridge", "Bell", "Crittenden", "Cumberland",
"Daviess", "Edmonson", "Calloway", "Caldwell", "Anderson", "Ballard",
"Bath", "Allen", "Graves", "Clinton", "Fleming", "Fulton", "Gallatin",
"Grayson", "Hardin", "Lincoln", "Green", "Greenup", "Hancock",
"Grant", "Harrison", "Laurel", "Larue", "Henderson", "Hickman",
"Lewis", "Letcher", "Hart", "Lawrence", "Lee", "Jackson", "Ohio",
"Taylor", "Owen", "Monroe", "Madison", "Trigg", "Spencer", "Washington",
"Trimble", "Scott", "Simpson", "Mason", "Marion", "Lyon", "Logan",
"McCracken", "Mercer", "Menifee", "McCreary", "Marshall", "Martin",
"McLean", "Woodford", "Wolfe", "Knott", "Montgomery", "Perry",
"Powell", "Christian", "Clark", "Campbell", "Carlisle", "Carroll",
"Robertson", "Rockcastle", "Rowan", "Russell", "Wayne", "Whitley",
"Oldham", "Muhlenberg", "Owsley", "Livingston", "Morgan", "Clay"
),
Cases = c(2, 3, 3, 3, 4, 3, 1, 3, 3, 5, 3, 2, 2, 1, 2, 2,
2, 2, 1, 4, 2, 3, 4, 3, 7, 4, 2, 2, 3, 2, 6, 3, 1, 1, 5, 4, 1,
3, 1, 1, 1, 1, 1, 1, 3, 1, 2, 1, 1, 1, 1, 4, 1, 1, 1, 1, 2, 2,
1, 1, 3, 1, 28, 2, 1, 2, 3, 5, 1, 1, 2, 1, 1, 6, 2, 1, 2, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 2, 2, 1, 3, 1, 1, 1,
1, 1, 1, 1, 1, 11, 3, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
sex = c("M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", "M",
"M", "M", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F",
"F", "F", "F", "F", "F")),
.Names = c("County", "Cases", "sex"),
row.names = 1:240, class = "data.frame")
larynx$age=NA

save(kentucky, larynx, larynxRates,
file="~/workspace/diseasemapping/pkg/diseasemapping/data/kentucky.RData",
compress='xz')

## End(Not run)
```

`nbToInlaGraph` *Write a graph file for INLA*

Description

Writes a graph file from an adjacency matrix suitable for use with INLA.

Usage

```
nbToInlaGraph(adjMat, graphFile="graph.dat")
```

Arguments

`adjMat`	An object of class nb containing the adjacency matrix.
`graphFile`	name of file to save adjacency matrix to.

Details

This function correctly handles regions which have zero neighbours.

Value

A vector of names and indices

Author(s)

Patrick Brown

See Also

poly2nb, nb2INLA

Examples

```
data('kentucky')
# remove all the neighbours Ballard county
kSub = kentucky[-c(2,20,79),]

if( require("spdep", quietly=TRUE)) {


adjMat = poly2nb(
kSub,
row.names=kSub$County,
queen=FALSE
)

nFile = tempfile()
nbRes = nbToInlaGraph(adjMat, nFile)

# Ballard is region 3
nbRes['Ballard']
# note that Ballard has no neighbours
adjMat[[3]]

cat(readLines(nFile, n=5), sep='\n')

## Not run:
# there will be a warning about zero neighbours
junk = bym(poverty ~ 1, data=kSub, family='gaussian')

## End(Not run)

}
```

popdata *Ontario 2006 population by census subdivision*

Description

Data set contains the information of population, by age, sex, and census subdivision.

Usage

```
data(popdata)
```

Format

A SpatialPolygonsDataFrame object, which needs the sp package for full functionality.

Details

This data is from the 2006 Census of canada offering by Statistics Canada web site, `www12.statcan.gc.ca/english/ce`

Examples

```
data(popdata)
head(popdata@data)
## Not run:
library(sp)
spplot(popdata, zcol='F.50_54', breaks=9, col=rainbow(8))

## End(Not run)
## Not run:

library('raster')
library('sp')
bfile = tempfile(fileext='.zip')
download.file(
paste('http://www12.statcan.gc.ca/census-recensement/',
'2011/geo/bound-limit/files-fichiers/gcsd000a06a_e.zip',
sep=''),
bfile)
unzip(bfile, exdir=tempdir())
sfile = grep('shp$',unzip(bfile, list=TRUE)$Name, value=TRUE)
popdata = shapefile(file.path(tempdir(),sfile))
popdata$PRNAME = iconv(popdata$PRNAME, 'UTF-8', 'latin1')
popdata = popdata[grep("^Ont", popdata$PRNAME),]
popdataS= rgeos::gSimplify(popdata, 0.01, topologyPreserve=TRUE)
popdata = SpatialPolygonsDataFrame(popdataS, popdata@data)
projection(popdata) = CRS('+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0')

pfile = tempfile(fileext='zip')

download.file(
paste('https://www12.statcan.gc.ca/census-recensement/',
'2011/dp-pd/prof/details/download-telecharger/comprehensive/',
'comp_download.cfm?CTLG=92-591-XE&FMT=CSV301&Lang=E&Tab=1&',
'Geo1=PR&Code1=01&Geo2=PR&Code2=01&Data=Count&SearchText=&',
'SearchType=Begins&SearchPR=01&B1=All&Custom=&TABID=1', sep=''),
pfile, method='curl')
unzip(pfile, exdir=tempdir())
ofile = grep('ONT', unzip(pfile, list=TRUE)$Name,value=TRUE)
opop = read.table(file.path(tempdir(),ofile),header=F,skip=3,
sep=',', nrows=163210,stringsAsFactors=FALSE)
opop= opop[grep("^([[:digit:]]|to| )+ years( and over)?$", opop[,7]),]
opop = opop[,c(1,4,7,11,13)]
colnames(opop) = c('id','name','var','M','F')
opop[,'var'] = gsub(" to ", "_", opop[,'var'])
opop[,'var'] = gsub(" years( and over)?", "", opop[,'var'])
opop[,'var'] = gsub("[[:space:]]", "", opop[,'var'])
opop2 = reshape(opop, direction='wide',
idvar=c('id','name'),
timevar='var', v.names=c('M','F'))


popdata = sp::merge(popdata, opop2, by.x='CSDUID', by.y='id')
popdata=popdata[,c('CSDUID', grep("^(M|F)", names(popdata), value=TRUE))]



save(popdata, file=
'/home/patrick/workspace/diseasemapping/pkg/diseasemapping/data/popdata.RData',
compress='xz')

## End(Not run)
```

`referencepop` *Standard Canadian population data set from year 1991.*

Description

A data set contains population and age sex group from year 1991.

Usage

```
data(referencepop)
```

Format

Data frame with columns POPULATION, sex, and age for the Canada 1991 population.

Details

data frame with rows representing age-sex groups, first column giving proportion of Canada 1991 population in that group, and subsequent columns giving sex and start of age range for each group

Examples

```
data(referencepop)
head(referencepop)
sum(referencepop$POPULATION)
```

Index