

# Package ‘diseasemapping’

January 17, 2012

**Type** Package

**Title** Calculate observed and expected disease incidence counts from a case file and population data.

**Version** 0.6

**Date** 2012-01-17

**Depends** mgcv, sp

**Enhances** diseasemappingDevel, glmmBUGS, spdep, INLA

**Author** Patrick Brown <patrick.brown@utoronto.ca>

**Maintainer** Lutong Zhou <carly\_zhou@hotmail.com>

**Description** Some tools for calculating rates and Standardized Incidence Ratios from case and population data.

**License** GPL

**Repository** CRAN

**Date/Publication** 2012-01-17 19:37:49

## R topics documented:

|                                  |    |
|----------------------------------|----|
| diseasemapping-package . . . . . | 2  |
| cancerRates . . . . .            | 3  |
| casedata . . . . .               | 4  |
| computeArea . . . . .            | 5  |
| expandCensus . . . . .           | 5  |
| formatCases . . . . .            | 6  |
| formatPopulation . . . . .       | 7  |
| getBreaks . . . . .              | 8  |
| getRates . . . . .               | 9  |
| getSMR . . . . .                 | 10 |
| getStdRate . . . . .             | 11 |
| mergeBugsData . . . . .          | 12 |
| popdata . . . . .                | 13 |
| referencepop . . . . .           | 14 |

---

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(popdata)
data(casedata)

model = getRates(casedata, popdata, ~age*sex, breaks=seq(0, 90, by=10) )
ontario = getSMR(popdata,model, casedata,regionCode="CSDUID", regionCodeCases="CSD2006")
## Not run:
splot(ontario, 'SMR')

## End(Not run)

# 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)

data(ontarioResult)
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
```

---

|             |  |
|-------------|--|
| cancerRates | <i>Download cancer incidence rates from the International Agency for Research on Cancer (IARC)</i> |
|-------------|--|

---

### Description

Rates by age and sex group are retrieved 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

```
## Not run:  
# won't work if offline or if the iarc web site is down  
qcLungF=cancerRates(area="canada", year=2001:2002, site="lung", sex="F")  
data(popdata)  
qcLungExp = getSMR(popdata, qcLungF)  
spplot(qcLungExp, "logExpected")  
  
## End(Not run)
```

---

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) ...
```

---

|             |                                  |
|-------------|----------------------------------|
| computeArea | <i>Compute areas of polygons</i> |
|-------------|----------------------------------|

---

**Description**

Computes the areas of all the polygons of a spatialPolygonsDataFrame object

**Usage**

```
computeArea(sp)
```

**Arguments**

sp                    A spatialPolygonsDataFrame

**Value**

A list, each element containing the area of a polygon.

**Examples**

```
data(popdata)
theAreas = computeArea(popdata)
# histogram of areas of ontario CDS's
hist(unlist(theAreas))
```

---

|              |   |
|--------------|---|
| expandCensus | <i>Given population data at each census year and range of the years that the data spans, this function returns a new list of population data for each consecutive year assuming population is constant between two census years</i> |
|--------------|---|

---

**Description**

Use if census are not done at each year

**Usage**

```
expandCensus(popdata, year.range = range(as.integer(names(popdata))), years = as.integer(names(popdat
```

**Arguments**

|            |  |
|------------|--|
| popdata    | A list of population data at each census year  |
| year.range | The range of the years which the census covers |
| years      | The years that the census data are collected   |

**Author(s)**

Patrick Brown

---

|             |   |
|-------------|---|
| formatCases | <i>Format the disease case data set</i> |
|-------------|---|

---

**Description**

The formatCases function 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 be 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 summed 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 | <i>Format the population data set</i> |
|------------------|---------------------------------------|

---

**Description**

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

**Usage**

```

formatPopulation(popdata, aggregate.by = NULL, breaks = NULL, ...)
## S3 method for class 'data.frame'
formatPopulation(popdata, aggregate.by=NULL, breaks=NULL,...)
## S3 method for class 'list'
formatPopulation(popdata, aggregate.by=NULL, breaks = NULL, years=as.integer(names(popdata)), year.range)

## S3 method for class 'SpatialPolygonsDataFrame'
formatPopulation(popdata, ...)

```

**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).                                    |
| years        | the years vector   |
| aggregate    | if TRUE then aggregate the population data   |
| year.range   | the range of the year vector   |
| time         | the time variable, i.e years   |
| personYears  | convert populations to person-years  |
| S            | can choose both or either of each sex.   |
| ...          | 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 not offer a breaks value, the function will aggregate by "age" as default.

**Author(s)**

Patrick Brown

**Examples**

```
data(popdata)
head(popdata@data)
poptry <- formatPopulation(popdata, breaks = c(seq(0, 80, by=10), Inf))
head(poptry)
poptryagg <- formatPopulation(popdata, 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**

|                       |  |
|-----------------------|--|
| <code>colNames</code> | names from the population data set                           |
| <code>breaks</code>   | the age breaks, i.e <code>breaks = seq(0, 80, by= 10)</code> |

**Examples**

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

---

|          |   |
|----------|---|
| getRates | <i>Calculate the estimated coefficients of age and sex group from the glm model</i> |
|----------|---|

---

**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, ~age*sex,breaks=c(seq(0, 80, by=10), Inf))
therates
```

---

getSMR

*Calculate the standardized mortality/morbidity ratios*

---

**Description**

The getSMR function 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**

```
getSMR(popdata, model, casedata, regionCode , regionCodeCases , area = FALSE, area.scale = 1, ...)
## S3 method for class 'data.frame'
getSMR(popdata, model, casedata=NULL, regionCode = intersect(names(popdata), names(casedata))[1], regionCodeCases, area, area.scale, years)
## S3 method for class 'list'
getSMR(popdata, model, casedata = NULL, regionCode , regionCodeCases, area , area.scale , years = NULL, personYears, year.range)
## S3 method for class 'SpatialPolygonsDataFrame'
getSMR(popdata, ...)
```

**Arguments**

|                 |  |
|-----------------|--|
| popdata         | the name of population data set  |
| model           | rates, either fitted model (usually a glm object), or a vector of rates. |
| ...             | additional arguments   |
| 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            | if TRUE, calculate the expected number of cases per area in each region  |
| area.scale      | control the unit of area. e.g $10^6$ : per square kilometers             |
| years           | the vector of years  |
| personYears     | a logic variable. If TRUE, it compute offset adding person years         |
| year.range      | the range of the year vector   |

**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(casedata)
data(popdata)
therates = getRates(casedata, popdata, ~age*sex,breaks=c(seq(0, 80, by=10), Inf) )
thesmr = getSMR(popdata, therates, casedata,regionCode="CSDUID", regionCodeCases="CSD2006")
head(thesmr@data)
```

---

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**

|                                  |   |
|----------------------------------|---|
| <code>relativeRate</code>        | the relative cancer rate calculated by <code>glmmBUGS</code> of different sex and age group of people from ontario .  |
| <code>model</code>               | the estimated cancer rate calculated by <code>glm</code> model of different sex and age group of people from ontario. |
| <code>referencePopulation</code> | the standard Canadian population from year 1991.  |
| <code>scale</code>               | the unit of the population.   |

**Author(s)**

Lutong Zhou

**Examples**

```

data(referencepop)
data(casedata)
data(popdata)
# get the age sex group rate:
model = getRates(casedata, popdata, ~age*sex, S=c("M","F"))
# calculate the relative rate using glmmBUGS:
ontario = getSMR(popdata, model, casedata, regionCode="CSDUID", regionCodeCases="CSD2006")
newpop <- getStdRate(ontario$SMR, model, referencepop, scale=100000)

```

---

|               |  |
|---------------|--|
| mergeBugsData | <i>merge the result from bugs function</i> |
|---------------|--|

---

**Description**

merge the result from bugs function

**Usage**

```

mergeBugsData(x, bugsSummary, by.x = NULL, newcol = "mean", ...)
## S3 method for class 'SpatialPolygonsDataFrame'
mergeBugsData(x, bugsSummary, by.x=NULL, newcol="mean", ...)
## S3 method for class 'data.frame'
mergeBugsData(x, bugsSummary, by.x=NULL, newcol="mean", ...)

```

**Arguments**

|             |  |
|-------------|--|
| x           | spatial polygon object i.e population data set (popdata)                     |
| bugsSummary | posterior distribution result from summaryChain function                     |
| by.x        | the common term from the spatial polygon object and the bugs function result |
| newcol      | the summary statistic that to be merged back to the data frame               |
| ...         | additional arguments   |

**Author(s)**

Patrick Brown

**Examples**

```

#data(popdata)
#newdata = c("3560102"=2, "3560104"=3)
#popdatatry = mergeBugsData(popdata, newdata, by.x="CSDUID")

# if the data set is a spatial polygons data frame:
#popdatatry = mergeBugsData.SpatialPolygonsDataFrame(popdata, newdata, by.x="CSDUID")

```

```

# if the data set is a data frame
#popdatatry = mergeBugsData.data.frame(popdata, newdata, by.x="CSDUID")

## Not run:
library(glmBUGS)
data(popdata)
data(casedata)

therates = getRates(casedata, popdata, ~age*sex)
ontario = getSMR(popdata, therates, casedata)
ontario@data = ontario@data[,c("CSDUID", "observed", "logExpected")]
library(spdep)
popDataAdjMat = poly2nb(ontario, ontario[["CSDUID"]])
library(glmBUGS)
forBugs = glmBUGS(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)

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

## 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/census](http://www12.statcan.gc.ca/english/census)

**Examples**

```
data(popdata)
head(popdata@data)
## Not run:
library(sp)
plot(popdata)

## 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

**Details**

This is a data set contains 36 rows and 3 columns.

**Examples**

```
data(referencepop)
## maybe str(referencepop) ; plot(referencepop) ...
```

# Index

## \*Topic **datasets**

- [casedata](#), [4](#)
  - [popdata](#), [13](#)
  - [referencepop](#), [14](#)
  
- [cancerRates](#), [3](#)
- [casedata](#), [4](#)
- [computeArea](#), [5](#)
  
- [diseasemapping](#)
  - [\(diseasemapping-package\)](#), [2](#)
- [diseasemapping-package](#), [2](#)
  
- [expandCensus](#), [5](#)
  
- [formatCases](#), [6](#)
- [formatPopulation](#), [7](#)
  
- [getBreaks](#), [8](#)
- [getRates](#), [9](#)
- [getSMR](#), [10](#)
- [getStdRate](#), [11](#)
  
- [mergeBugsData](#), [12](#)
  
- [popdata](#), [13](#)
  
- [referencepop](#), [14](#)