Package 'hhh4contacts'

June 30, 2017

Version 0.13.0

Date 2017-06-29

License GPL-2

Title Age-Structured Spatio-Temporal Models for Infectious Disease Counts

Description Meyer and Held (2017) <doi:10.1093/biostatistics/kxw051> present an age-structured spatio-temporal model for infectious disease counts. The approach is illustrated in a case study on norovirus gastroenteritis in Berlin, 2011-2015, by age group, city district and week, using additional contact data from the POLYMOD survey. This package contains the data and code to reproduce the results from the paper, see 'demo(``hhh4contacts'')'.

Depends R (>= 3.2.0), grDevices, graphics, methods, stats, utils, surveillance (>= 1.14.0)

Suggests MASS, lattice, gridExtra, scales, numDeriv

LazyData yes

RoxygenNote 6.0.1

NeedsCompilation no

Author Sebastian Meyer [aut, cre], Leonhard Held [ctb, ths]

Maintainer Sebastian Meyer <seb.meyer@fau.de>

Repository CRAN

Date/Publication 2017-06-30 17:28:14 UTC

R topics documented:

`adaptP` *Adapt a Transition Matrix to a Specific Stationary Distribution*

Description

Experimental Metropolis-Hastings algorithm, which tries to adjust a transition matrix such that its stationary distribution becomes approximately equal to a prespecified probability vector.

Usage

```
adaptP(P, target, niter = 1e+06)
```

Arguments

`P`	a transition matrix, i.e., a square matrix where all rows sum to 1.
`target`	the stationary probability vector to approximate.
`niter`	the number of iterations of the MCMC algorithm

Value

the adjusted transition matrix.

Author(s)

Leonhard Held

See Also

`C2pop` for an alternative method.

Examples

```
## a row-normalized contact matrix
C <- matrix(c(0.8, 0.1, 0.1,
              0.2, 0.6, 0.2,
              0.1, 0.2, 0.7), byrow=TRUE, ncol=3, nrow=3)
stationary(C)
## population fractions define the target distribution
popfracs <- c(0.4, 0.3, 0.3)
## adapt 'C' to the given population fractions
Cpop <- adaptP(C, popfracs, niter = 50000)
stationary(Cpop)
## this method increases the diagonal values of 'C'
round(C, 3)
round(Cpop, 3)
round(Cpop/C, 3)
```

---

addGroups2WFUN *Group-Dependent Parametric Weights*

Description

This function takes a specification of parametric weights and returns a modified version with group-dependent parameters. Only single-parameter functions are currently supported.

Usage

```
addGroups2WFUN(WFUN, groups, initial = rep.int(WFUN$initial, nlevels(groups)))
```

Arguments

WFUN	a list specification of parametric weights, e.g., as returned by the constructor functions W_powerlaw and W_np.
groups	a vector of length nUnits determining to which group each unit belongs to. The supplied vector is converted to a factor using as.factor.
initial	(named) vector of initial parameters.

Value

a list specifying group-dependent parametric weights for hhh4.

Author(s)

Sebastian Meyer

Examples

```
data("measlesWeserEms")
WPLgroups <- addGroups2WFUN(
  W_powerlaw(maxlag = 5, normalize = FALSE, log = FALSE),
  groups = factor(sample(2, ncol(measlesWeserEms), replace = TRUE)))
```

---

aggregateC *Aggregate a Contact Matrix*

Description

The (age) groups of a contact matrix can be joined together by the `grouping` argument, which first sums over contact groups (columns) and then averages over the corresponding participant groups (rows), optionally using weights such as the age distribution of the study participants.

Usage

```
aggregateC(C, grouping, ..., weights = NULL)
```

Arguments

`C`	a square numeric contact matrix such as `contactmatrix_POLYMOD`.
`grouping`	specification of how to aggregate groups (a named list or an integer vector of group sizes) using `aggregateC` with the `"agedistri"` attribute of the contact matrix as weights. If `NULL`, the original 5-year intervals are returned. The default setting produces the six age groups of Meyer and Held (2017).
`...`	specification of how to aggregate groups (alternative to using a named list as the `grouping` argument).
`weights`	a named numeric vector containing the weights for the rows of `C`, typically the age distribution of the participants. The names are matched against `rownames(C)`. A value of `NULL` is interpreted as uniform weights.

Author(s)

Sebastian Meyer

aggregateCountsArray *Aggregate an Array of Counts wrt One Dimension (Stratum)*

Description

Aggregate an Array of Counts wrt One Dimension (Stratum)

Usage

```
aggregateCountsArray(counts, dim, grouping, ..., sort = TRUE)
```

Arguments

counts	an (integer) array of counts with dimnames, e.g., counts or pop2011.
dim	the dimension index of the stratum defining groups.
grouping, ...	how the groups should be built.
sort	logical indicating if the resulting array should be ordered by the grouping levels in the dim dimension.

Value

an array with similar dimensions as the input counts except for the dim dimension, which will be smaller due to the aggregation as specified by the grouping argument.

Author(s)

Sebastian Meyer

Examples

```
## works for matrices
aggregateCountsArray(pop2011, dim = 2, grouping = c(2,1,3,2,4))
aggregateCountsArray(pop2011, dim = 1, grouping = list(
    "a" = c("chwi","span","zehl"),
    "b" = c("neuk","scho")
))
## and of course for arrays
str(aggregateCountsArray(counts, dim = 3, grouping = c(1, 3, 4)))
```

C2pop *Adapt a Contact Matrix to Population Fractions*

Description

Experimental function, which tries to adjust a given contact matrix such that the stationary distribution of its row-normalized version (i.e., the transition matrix) becomes approximately equal to a prespecified probability vector.

Usage

```
C2pop(C, target, eps = 0.001, iter.max = 100)
```

Arguments

C	a square numeric (contact) matrix.
target	the stationary probability vector to approximate.
eps	the tolerated mean absolute difference between the target probabilities and the stationary distribution of the adapted, normalized contact matrix.
iter.max	maximum number of iterations (guard against infinite loop).

Value

the adapted, normalized contact matrix.

Author(s)

Leonhard Held (original) and Sebastian Meyer (this implementation)

See Also

adaptP for an alternative method.

Examples

```
GROUPING <- c(1, 2, 2, 4, 4, 2)
C <- contactmatrix(grouping = GROUPING)
popBErbyg <- aggregateCountsArray(pop2011, dim = 2, grouping = GROUPING)
popfracs <- prop.table(colSums(popBErbyg))
## adapt 'C' to the given population fractions
Cpop <- C2pop(C, popfracs)
## compare the stationary distributions
compstat <- cbind(before = stationary(C/rowSums(C)), popBE = popfracs,
                  after = stationary(Cpop))
matplot(compstat, type="b", lty=1, ylim=c(0, max(compstat)),
        xlab="age group", ylab="population fraction")
## compare the normalized contact matrices
print(plotC(C/rowSums(C), main="original", at=seq(0,0.6,length.out=17)),
```

```
      split=c(1,1,2,1), more=TRUE)
print(plotC(Cpop, main="adapted", at=seq(0,0.6,length.out=17)),
      split=c(2,1,2,1), more=FALSE)
```

`contactmatrix` *POLYMOD Contact Matrices for Germany*

Description

The function `contactmatrix` retrieves various social contact matrices for Germany from the POLYMOD survey (Mossong et al., 2008). Such a matrix contains the average numbers of reported contacts by participant age group. The original age groups (5-year intervals) can be joined together by the `grouping` argument, which first sums over contact groups (columns) and then averages over the corresponding participant groups (rows) using the corresponding age distribution as weights.

Usage

```
contactmatrix(which = c("corrected", "mossong", "reciprocal"),
  type = c("all", "physical"), grouping = c(1, 2, 2, 4, 4, 2),
  normalize = FALSE)

contactmatrix_mossong

contactmatrix_mossong_physical

contactmatrix_POLYMOD

contactmatrix_POLYMOD_physical

contactmatrix_wallinga

contactmatrix_wallinga_physical
```

Arguments

`which`	character string indicating which contact matrix to return. `"mossong"` uses the average numbers of reported contacts as published in Table S5 of Mossong et al. (2008), available as `contactmatrix_mossong` or `contactmatrix_mossong_physical`. `"corrected"` (from `contactmatrix_POLYMOD` or `contactmatrix_POLYMOD_physical`) fixes an error in these numbers related to the age group 70+ (see the Examples) and is the default. If `which="reciprocal"` (corresponding to `contactmatrix_wallinga` or `contactmatrix_wallinga_physical` as used by Meyer and Held, 2017), the returned social contact matrix fulfils reciprocity of contacts with respect to the age distribution of Berlin, `pop2011`, via the method of Wallinga et al. (2006).
`type`	a character string to select the type of contacts to use: either `"all"` contacts, i.e., count both physical and pure conversational contacts, or only `"physical"` contacts.

`grouping` specification of how to aggregate groups (a named list or an integer vector of group sizes) using `aggregateC` with the `"agedistri"` attribute of the contact matrix as weights. If `NULL`, the original 5-year intervals are returned. The default setting produces the six age groups of Meyer and Held (2017).

`normalize` a logical indicating whether to normalize the matrix such that each row sums to 1.

Format

The dataset `contactmatrix_POLYMOD` and its variants are all square numeric matrices with 15 rows (participants) and 15 columns (contacts), labelled with the corresponding age groups. There is an attribute `"agedistri"`, a named numeric vector of length 15, which for the "_mossong_" and "_POLYMOD_" variants gives the age distribution of the German POLYMOD sample, and for the `_wallinga_` variants gives the age distribution of Berlin, i.e., `prop.table(colSums(pop2011))`.

Value

a square numeric matrix containing the average numbers of contact persons recorded per day per survey participant in Germany, potentially averaged over multiple *row* (participant) age groups and aggregated over the corresponding *column* (contact) age groups.

Author(s)

Sebastian Meyer

Source

`contactmatrix_mossong` and `contactmatrix_mossong_physical` are taken from the Supporting Information in Mossong et al. (2008): the matrices from Table S5 (8.2), and the attached age distribution from Table S2 (3.2).

The corrected versions `contactmatrix_POLYMOD` and `contactmatrix_POLYMOD_physical` were constructed from the raw POLYMOD data available at https://www.researchgate.net/publication/232701632_POLYMOD_contact_survey_for_researchers. The reciprocal contact matrices `contactmatrix_wallinga` and `contactmatrix_wallinga_physical` were estimated from these raw data via the method of Wallinga et al. (2006).

References

Meyer S and Held L (2017): Incorporating social contact data in spatio-temporal models for infectious disease spread. *Biostatistics*, **18** (2), 338-351. doi: 10.1093/biostatistics/kxw051

Mossong et al. (2008): Social contacts and mixing patterns relevant to the spread of infectious diseases. *PLoS Medicine*, **5** (3), e74. doi: 10.1371/journal.pmed.0050074

Wallinga J, Teunis P and Kretzschmar M (2006): Using data on social contacts to estimate age-specific transmission parameters for respiratory-spread infectious agents. *American Journal of Epidemiology*, **164** (10), 936-944. doi: 10.1093/aje/kwj317

Examples

```
## contact matrix reported in Mossong et al (2008, Table S5)
(C_original <- contactmatrix(which = "mossong", grouping = NULL))
## this simply returns the dataset 'contactmatrix_mossong'
stopifnot(identical(C_original, contactmatrix_mossong))

## with corrected numbers for the 70+ age group (the default)
C_corrected <- contactmatrix(which = "corrected", grouping = NULL)
## this simply returns the dataset 'contactmatrix_POLYMOD'
stopifnot(identical(C_corrected, contactmatrix_POLYMOD))

## check for differences
C_original == round(C_corrected, 2)
## compare entries of last row and last column
round(rbind(original = C_original[15,], corrected = C_corrected[15,]), 2)
round(cbind(original = C_original[,15], corrected = C_corrected[,15]), 2)

## contact matrix estimated to be reciprocal on the population level
C_reciprocal <- contactmatrix(which = "reciprocal", grouping = NULL)
## this simply returns the dataset 'contactmatrix_wallinga'
## (without its "overdisp" attribute)
stopifnot(all.equal(C_reciprocal, contactmatrix_wallinga, check.attributes=FALSE))

## check reciprocity
agedistriBE <- attr(C_reciprocal, "agedistri")
stopifnot(identical(agedistriBE, prop.table(colSums(pop2011))))
stopifnot(isSymmetric(C_reciprocal * agedistriBE, check.attributes=FALSE))

## visually compare raw to reciprocal contact matrix
if (require("gridExtra"))
    grid.arrange(plotC(C_corrected, main = "raw"),
                 plotC(C_reciprocal, main = "reciprocal"),
                 nrow = 1)

## select physical contacts and aggregate into 5 age groups
contactmatrix(type = "physical", grouping = c(1, 2, 7, 3, 2))

## the default 6 age groups, normalized to a transition matrix
contactmatrix(normalize = TRUE)

## reciprocity also holds for this grouping
(C6 <- contactmatrix(which = "reciprocal"))
stopifnot(isSymmetric(C6 * attr(C6, "agedistri"), check.attributes=FALSE))
```

---

dssAggregate *Compute the DSS on Aggregated Predictions and Observations*

Description

The expectation and variance of aggregated predictions is just a sum if the predictions are (conditionally) independent. This function computes the DSS for a matrix of observations and a matrix of predictions where the columns are to be summed according to a given factor.

Usage

```
dssAggregate(observed, pred, psi, groups)
```

Arguments

observed	a numeric matrix of observed counts.
pred	a numeric matrix of predicted counts.
psi	a numeric vector or matrix of overdispersion parameters such that `pred * (1 + pred/exp(psi))` is the prediction's variance. Alternatively, `psi = NULL` indicated Poisson predictions.
groups	a factor variable of length `ncol(observed)` indicating which columns should be aggregated.

Value

a matrix of DSS values

---

expandC *Expand the Contact Matrix over Regions*

Description

This is simply the Kronecker product of the contact matrix `C` with a matrix of ones of dimension `n` x `n`.

Usage

```
expandC(C, n)
```

Arguments

C	a contactmatrix.
n	the size of the secondary dimension to expand to.

Value

a square matrix with `n*ncol(C)` rows and columns.

Examples

```
expandC(contactmatrix(), 2)
```

fitC — *Estimate the Power of the Contact Matrix in a "hhh4" Model*

Description

The profile log-likelihood of the log(power) parameter of the contact matrix (see powerC) is maximized using optim. The hhh4 fit for the optimal power value is returned with an additional element logpower which holds information on the result of the optimization.

Usage

```
fitC(object, C, normalize = TRUE, truncate = TRUE, optim.args = list(),
  ...)
```

Arguments

object	a model fit of class "hhh4".
C	the contact matrix to use.
normalize, truncate	see powerC.
optim.args	a list to modify the default optimization parameters.
...	additional arguments for each run of update.hhh4.

Value

an object of class "fitC", which is an "hhh4" object with an additional element logpower.

Author(s)

Sebastian Meyer

noroBE — *Create "sts" Objects from the Berlin Norovirus Data*

Description

The function noroBE() creates an "sts" object based on the array of norovirus surveillance counts, the map of Berlin's city district, and the pop2011 data stored in the package. This is the data analysed by Meyer and Held (2017).

Usage

```
noroBE(by = c("districts", "agegroups", "all", "none"), agegroups = c(1, 2,
  2, 4, 4, 2), timeRange = c("2011-w27", "2015-w26"), flatten = FALSE)

counts

map
```

Arguments

by	character string determining the stratification, i.e., which units the resulting "sts" object should represent: **"districts":** aggregates counts and pop2011 over the age groups and stores the matrix of adjacency orders from the map in the neighbourhood slot. The latter is obtained via nbOrder(poly2adjmat(map), maxlag = 5). **"agegroups":** aggregates counts and pop2011 over the districts and stores the contactmatrix() in the neighbourhood slot, potentially also combining some age groups via the agegroups argument. **"all":** retains both dimensions, either as a list of spatial "sts" objects per age group, or in a single "sts" object (see flatten below). **"none":** creates the overall (univariate) time series of rowSums(counts).
agegroups	how the age groups in counts (and pop2011) should be aggregated. Will be used as the grouping argument in aggregateCountsArray and contactmatrix. The default setting uses the six age groups of Meyer and Held (2017).
timeRange	character vector of length two determining the time range of the "sts" object to generate. The two strings are matched against dimnames(counts)[[1]], which ranges from "2011-w01" until "2016-w30". The default value extracts four seasons (years) starting at "2011-w27".
flatten	logical indicating whether for by = "all" a single "sts" object should be returned where the observation unit is the interaction of district and age group ("flattened" counts array, see as.data.frame.array). By default (flatten = FALSE), a list of district-based "sts" objects is returned, one for each age group.

Format

counts: an integer-valued array of norovirus surveillance counts with labelled dimensions of size 290 ("week") x 12 ("district") x 15 ("agegroup").

map: a "SpatialPolygonsDataFrame" of length 12 with row.names(map) matching colnames(counts), representing Berlin's city districts in longlat coordinates (WGS84). The data slot contains the full "NAME"s of the city districts as well as their "POPULATION", i.e., rowSums(pop2011).

The function noroBE() returns an "sts" object generated from these data (and pop2011).

Author(s)

Sebastian Meyer

Source

counts: based on norovirus surveillance counts retrieved from the SurvStat@RKI 2.0 online service (https://survstat.rki.de) of Germany's public health institute, the Robert Koch Institute, as of 2016-09-08.

map: based on a KML file of Berlin's 97 local centres ("Ortsteile") downloaded from the Berlin Open Data repository at https://daten.berlin.de/datensaetze/geometrien-der-ortsteile-von-berlin-jul as of 2014-11-12, published by *Amt fuer Statistik Berlin-Brandenburg* (Statistical Office of Berlin-Brandenburg) under the 'CC BY 3.0 DE' license (https://creativecommons.org/licenses/by/3.0/de/). The map included here is a gUnaryUnion of these local centres by city district.

References

Meyer S and Held L (2017): Incorporating social contact data in spatio-temporal models for infectious disease spread. *Biostatistics*, **18** (2), 338-351. doi: 10.1093/biostatistics/kxw051

Examples

```
## the raw data
str(counts)
summary(map)

## district-specific time series
noroBEr <- noroBE(by = "districts")
plot(noroBEr)

## age group-specific time series
noroBEg <- noroBE(by = "agegroups")
plot(noroBEg)

## list of spatio-temporal surveillance counts, one for each age group
noroBErbyg <- noroBE(by = "all", flatten = FALSE)
plot(noroBErbyg[[1L]], par.list = list(oma=c(0,0,2,0)))
title(main = names(noroBErbyg)[1], outer = TRUE, line = -1)

## flattened "sts" object (the 'neighbourhood' only reflects spatial info)
noroBEall <- noroBE(by = "all", flatten = TRUE)
dev.new(width = 16, height = 7)
plot(noroBEall, par.list = list(
    xaxt = "n", mar = c(1,4,1,1), mfrow = c(ncol(noroBEg), ncol(noroBEr))
))
```

---

plotC *Generate an Image of a Contact Matrix*

Description

Generate an Image of a Contact Matrix

Usage

```
plotC(C, grouping = NULL, xlab = "age group of contact",
  ylab = "age group of participant", at = 15,
  col.regions = rev(heat.colors(length(at) - 1)), ..., contour = FALSE)
```

Arguments

C	a square numeric matrix.
grouping	numeric vector of sizes of aggregated groups, e.g., grouping = c(1,3), to draw separation lines after the first and the forth subgroup. This is ignored if contour = TRUE.
xlab, ylab	axis labels.
at	numeric vector of break points of the color levels, or a single integer specifying the number of cuts (which defaults to 15 as in levelplot).
col.regions	vector of color levels.
...	further arguments passed to levelplot or filled.contour (if contour = TRUE).
contour	logical indicating if a filled.contour should be drawn instead of a levelplot (the default).

Examples

```
## contour plot
plotC(contactmatrix_POLYMOD, contour = TRUE)

## level plots illustrating aggregation of age groups
if (require("gridExtra")) {
    grid.arrange(plotC(contactmatrix_POLYMOD, grouping = c(1,2,2,4,4,2)),
                 plotC(contactmatrix(grouping = c(1,2,2,4,4,2))),
                 nrow = 1)
}
```

---

plotHHH4_fitted_groups

Plot Mean Components of a hhh4 *Fit by Group*

Description

Fitted mean components for age-structured, areal time series hhh4 models can be aggregated over districts or age groups.

Usage

```
plotHHH4_fitted_groups(x, groups, total = FALSE, decompose = NULL, ...)
```

Arguments

x	an object of class "hhh4".
groups	a factor of grouping the units in the model, i.e., it must be of length x$nUnit. There will be one plot for each factor level.
total	a logical indicating if the group-wise mean components should be subsequently summed up over all groups for an overall plot.
decompose, ...	see plotHHH4_fitted.

Value

see plotHHH4_fitted.

---

plotHHH4_maps_groups *Plot Mean Components of a* hhh4 *Fit by District Averaged Over Time*

Description

This is a wrapper for plotHHH4_maps with prior aggregation over different (age) groups.

Usage

```
plotHHH4_maps_groups(x, map, districts, ...)
```

Arguments

x	an object of class "hhh4".
map	an object inheriting from "SpatialPolygons".
districts	a factor of length x$nUnit with as many levels as there are districts and names according to row.names(map).
...	arguments passed to plotHHH4_maps.

Value

see plotHHH4_maps

plotHHH4_season_groups

Plot Seasonality of a hhh4 *Fit by Group*

Description

A plot method for models with group-specific seasonality terms that are not handled correctly by plotHHH4_season.

Usage

```
plotHHH4_season_groups(x, component = "end", seasonStart = 1,
  conf.level = 0.95, conf.B = 999, col = 1:6, xlab = "time",
  ylab = "multiplicative effect", ..., refline.args = list(),
  yearline.args = list(), legend.args = list())
```

Arguments

x	an object of class "hhh4".
component	character string indicating from which component seasonality terms should be extracted.
seasonStart	an integer defining the epochInYear that starts a new season (by default the first).
conf.level, conf.B	a confidence level for the pointwise confidence intervals around the group-specific seasonal effects. The confidence intervals are based on quantiles of conf.B samples from the asymptotic multivariate normal distribution of the maximum likelihood estimate. Alternatively, if conf.level = NA, the individual samples are drawn instead of the confidence lines. Set conf.level = NULL to disable confidence intervals.
col	a vector of group-specific colors, recycled as necessary and passed to matplot.
xlab, ylab, ...	arguments passed to matplot.
refline.args	a list of arguments for abline to change the style of the horizontal reference line at 1. This line is omitted if refline.args is not a list.
yearline.args	a list of arguments for abline to change the style of the line marking the end of the year at x$stsObj@freq if seasonStart is not 1. This line is omitted if yearline.args is not a list.
legend.args	a list of arguments for legend modifying the internal defaults. If legend.args is not a list, the legend is omitted.

Value

a matrix of the plotted point estimates of the multiplicative seasonal effect by group.

pop2011 *Berlin and German Population by Age Group, 2011*

Description

Population numbers from Berlin are available in the city district x age group (5-year intervals) matrix `pop2011`. The corresponding age distribution for whole Germany is stored in the vector `popDE`.

Usage

```
## Berlin population by city district and age group, 2011
pop2011

## German population by age group, 2011
popDE
```

Format

pop2011: a named, integer-valued 12 (city districts) x 15 (age groups) matrix.

popDE: a named integer vector of length 15 (age groups).

Author(s)

Sebastian Meyer

Source

pop2011: numbers extracted from https://www.statistik-berlin-brandenburg.de/webapi/opendatabase?id=BevBBBE as of 2011-12-31 (before census), published by *Amt fuer Statistik Berlin-Brandenburg* (Statistical Office of Berlin-Brandenburg) under the 'CC BY 3.0 DE' license (https://creativecommons.org/licenses/by/3.0/de/).

popDE: numbers extracted from https://www-genesis.destatis.de/genesis/online/link/tabellen/12411-0005 as of 2010-12-31, published by *Statistisches Bundesamt* (Destatis, Federal Statistical Office of Germany) under the 'Data licence Germany - attribution - Version 2.0' (https://www.govdata.de/dl-de/by-2-0).

powerC *Exponentiate a Matrix via Eigendecomposition*

Description

Based on a (contact) matrix C, the function make_powerC generates a function with a single argument power that returns the input matrix raised to that power. Matrix exponentiation is thereby defined via the eigendecomposition of C as $C^{power} := E\Lambda^{power}E^{-1}$.

Usage

```
make_powerC(C, normalize = FALSE, truncate = FALSE)
```

Arguments

C	a square numeric matrix.
normalize	a logical indicating if C should be normalized in advance such that all rows sum to 1 (becomes a transition matrix).
truncate	a logical indicating whether to force entries in the resulting matrix to be non-negative (by truncation at 0).

Value

a function of the power that returns the exponentiated matrix.

Examples

```
Cnorm <- contactmatrix(normalize = TRUE)
powerC <- make_powerC(Cnorm)
powerC(1)
powerC(0)
powers <- c(0, 0.5, 1, 2)
Cp <- lapply(powers, powerC)
if (require("gridExtra"))
    grid.arrange(
        grobs = mapply(plotC, C = Cp, main = paste("power =", powers),
                       SIMPLIFY = FALSE),
        nrow = 2, ncol = 2)

## truncation to enforce non-negative entries
powerC(0.2)  # some entries become negative for small powers
powerC0 <- make_powerC(Cnorm, truncate = TRUE)
powerC0(0.2)
```

stationary *Stationary Distribution of a Transition Matrix*

Description

This auxiliary function determines the stationary distribution from a transition matrix.

Usage

```
stationary(P)
```

Arguments

P	a transition matrix, i.e., a square matrix where all rows sum to 1.

Value

the stationary probability vector.

Author(s)

Leonhard Held

Examples

```
Cgrouped_norm <- contactmatrix(normalize = TRUE)
Cgrouped_norm
(p <- stationary(Cgrouped_norm))
(Cpowered <- make_powerC(Cgrouped_norm)(1e6))
stopifnot(all.equal(Cpowered[1,], p))
```

stratum *Extract Strata*

Description

Methods to extract strata information from an object. Here we only define a method for class "sts".

Usage

```
stratum(x, ...)

## S4 method for signature 'sts'
stratum(x, which = NULL, ...)
```

Arguments

x	an object of class "sts".
...	further arguments passed to methods.
which	an integer (strata dimension) or NULL (to get the plain colnames, the default).

Value

a character vector of strata names of length ncol(x).

Methods (by class)

- sts: Extract the names of the units, i.e., the colnames, from a multivariate "sts" object. If the units result from the interaction of multiple strata separated by dots, e.g., "region.group", the function can also extract the names corresponding to a specific strata dimension, e.g., which = 2 to get the group names.

Examples

```
noroBEall <- noroBE(by = "all", flatten = TRUE)
stratum(noroBEall)  # just colnames(noroBEall)
stratum(noroBEall, which = 2)  # the age groups
```

---

stsplothook *Hook functions for* stsplot_time1

Description

Hook functions can be passed to stsplot_time1, which are evaluated after all the plotting has been done, and with the hook function environment set to the evaluation environment of stsplot_time1 such that local variables can be accessed. They are not intended to be called directly.

Usage

```
stsplothook_highlight(christmas = FALSE, epochInYear = NULL, col = 2,
  lwd = 2)
```

Arguments

christmas	logical indicating if Christmas should be highlighted.
epochInYear	integer vector of epochs to highlight.
col, lwd	graphical parameters for the highlighting lines.

Author(s)

Sebastian Meyer

Examples

```
plot(noroBE("agegroups"), hookFunc = stsplothook_highlight(epochInYear=51))
```

Index