Package ‘DDM’

October 12, 2022

Type Package

Title Death Registration Coverage Estimation

Version 1.0-0

Date 2017-05-29

Author Tim Riffe, Everton Lima, Bernardo Queiroz

Maintainer Tim Riffe <riffe@demogr.mpg.de>

Description A set of three two-census methods to the estimate the degree of death registration coverage for a population. Implemented methods include the Generalized Growth Balance method (GGB), the Synthetic Extinct Generation method (SEG), and a hybrid of the two, GGB-SEG. Each method offers automatic estimation, but users may also specify exact parameters or use a graphical interface to guess parameters in the traditional way if desired.

Depends R (>= 2.15)

License GPL-2

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-05-29 14:55:29 UTC

R topics documented:

`addcod` *append a* `$cod` *column if missing*

Description

Only handles the case of missing `$cod` splitting variable for data of a single year/region. This is not super robust. If you have many regions or whatever then do it yourself. This function was just written to make `ggb()` robust to the case of a user specifying data that don't have any territorial or other subgroups, aside from sex.

Usage

```
addcod(X)
```

Arguments

`X`	a `data.frame`, possibly but not necessarily with column `$sex`.

Value

X with a new column, `$cod` appended.

`adjustages` *adjust the range of ages used by* `ggbChooseAges()`

Description

a utility function called by `ggbChooseAges()`. After clicking a point, this function readjusts the age range

Usage

```
adjustages(a, age, agesfit)
```

Arguments

`a`	an age specified by the user, as returned by `guessage()`
`age`	ages present in dataset
`agesfit`	the former age range used for calculating the coverage coefficient

Value

the adjusted set of ages used for calculating the coverage coefficient

---

`assignNoteCode` *a cheap way to choose which column to assign a* `NoteCode` *to*

Description

One property of the LexisDB scripts that might be useful for downstream checks is the ability to trace which functions have modified a given data object. These can go into NoteCode slots. This function writes code to the first unoccupied `NoteCode` column. If all three `NoteCode` columns are occupied, it concatenates the end of the third column. This way we preserve a full history. Unfortunately it gets split between columns. Oh well. Good for eyeballing. This function written for the sake of modularity. Function copied from Human Mortality Database collection directly as-is.

Usage

```
assignNoteCode(X, code)
```

Arguments

`X`	the HMD data object that presumably has three `NoteCode` columns
`code`	character string to assign to the column, typically the name of the function operating on `X`.

`avgDeaths` *if necessary divide deaths by intercensal interval*

Description

ideally `deaths` is the average annual deaths in the intercensal period, but it is also common to give it as the sum. If this was the case, set `deaths.summed` to `TRUE` and we take care of it.

Usage

```
avgDeaths(codi, deaths.summed = FALSE)
```

Arguments

`codi`	the standard object as described in e.g. `ggb()`.
`deaths.summed`	logical. If `TRUE` then `deaths` was specified as the sum over the intercensal period. Otherwise it was the mean.

Value

codi a new column, `deathsAvg` will be appended.

`BrasilFemales` *Example data for Brasil females by federal states, years 1991 to 2000*

Description

A dataset containing 486 rows and 7 variables: Population counts for 1991 and 2000 in abridged ages 0, 1, 5, ... 75, with an open age of 80. Deaths are given as the average death count per age group over the intercensal period. In total there are 53 states in this dataset.

Usage

```
BrasilFemales
```

Format

A data frame with 53940 rows and 10 variables:

cod integer an id number for each state

pop1 integer the census population count in 1991

pop2 integer the census population count in 2000

deaths numeric average deaths between censuses

year1 integer 1991

year2 integer 2000

age integer lower age bound for each age group

sex character, 'f'

Source

data downloaded from DATASUS `http://www.datasus.gov.br`

`BrasilMales` *Example data for Brasil males by federal states, years 1980 to 1991*

Description

A dataset containing 486 rows and 7 variables: Population counts for 1980 and 1991 in abridged ages 0, 1, 5, ... 75, with an open age of 80. Deaths are given as the average death count per age group over the intercensal period. In total there are 53 states in this dataset.

Usage

```
BrasilMales
```

Format

A data frame with 53940 rows and 10 variables:

cod integer an id number for each state

pop1 integer the census population count in 1991

pop2 integer the census population count in 2000

deaths numeric average deaths between censuses

year1 integer 1991

year2 integer 2000

age integer lower age bound for each age group

sex character, 'm'

Source

data downloaded from DATASUS `http://www.datasus.gov.br`

`cdmltw` *Coale-Demeny model life table: West*

Description

function from now-deprecated `demogR` package. Originally written by Ken Wachter, modified by James Jones, and again by the current maintainer, Tim Riffe. Only minor edits to margin naming in the current version.

Usage

```
cdmltw(sex = "F")
```

Arguments

`sex`	`"F"` or `"M"`

Value

Tons of lifetable output in matrices. Age in columns, lifetable levels in rows.

`ddm` *run all three deaths registration coverage estimation methods*

Description

Estimate the generalized growth balance method, and the two Bennett-Horiuchi methods of estimating death registration coverage. This requires two censuses and an estimate of the deaths in each 5-year age group between censuses. This might be the arithmetic average of deaths in each age class, or simply the average of deaths around the time of the two censuses. All methods use some stable population assumptions.

Usage

```
ddm(X, minA = 15, maxA = 75, minAges = 8, exact.ages = NULL,
  eOpen = NULL, deaths.summed = FALSE)
```

Arguments

`X`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, `$age`, `$sex`, and `$cod` (if there are more than 1 region/sex/intercensal period).
`minA`	the lowest age to be included in search
`maxA`	the highest age to be included in search (the lower bound thereof)
`minAges`	the minimum number of adjacent ages to be used in estimating
`exact.ages`	optional. A user-specified vector of exact ages to use for coverage estimation

`eOpen`	optional. A user-specified value for remaining life-expectancy in the open age group.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Details

All methods require some specification about which age range to base results on. If not given, an optimal age range will be estimated automatically, and this information is returned to the user. To identify an age-range in the visually, see `plot.ggb()`, when working with a single year/sex/region of data. The automatic age-range determination feature of this function tries to implement an intuitive way of picking ages that follows the advice typically given for doing so visually. We minimize the square of the average squared residual between the fitted line and right term.If you want coverage estimates for a variety of partitions (intercensal periods/regions/by sex), then stack them, and use a variable called `$cod` with unique values for each data partition. If data is partitioned using the variable `$cod`, then the age range automatically determined might not be the same for each partition. If user-specified, (using a vector of `exact.ages`) the age ranges will be the same for all partitions. If you want to specify particular age ranges for each data partition, then you'll need to loop it somehow.

All three methods require time points of the two censuses. Census dates can be given in a variety of ways: 1) (preferred) using `Date` classes, and column names `$date1` and `$date2` (or an unambiguous character string of the date, like, `"1981-05-13"`) or 2) by giving column names `"day1","month1","year1","day2","month2","year2"` containing respective integers. If only `year1` and `year2` are given, then we assume January 1 dates. If year and month are given, then we assume dates on the first of the month. Different values of `$cod` could indicate sexes, regions, intercensal periods, etc. The `$deaths` column should refer to the average annual deaths for each age class in the intercensal period. Sometimes one uses the arithmetic average of recorded deaths in each age, or simply the average of the deaths around the time of census 1 and census 2.

The synthetic extinct generation methods require an estimate of remaining life expectancy in the open age group of the data provided. This is produced using a standard reference to the Coale-Demeny West model life tables. That is a place where things can be improved.

Value

data.frame with columns `$cod`, `$ggb`, `$bh1`, `$bh2`, `$lower`, and `$upper`.

References

Bennett Neil G, Shiro Horiuchi. Estimating the completeness of death registration in a closed population. Population Index. 1981; 1:207-221.

Hill K. Estimating census and death registration completeness. Asian and Pacific Population Forum. 1987; 1:1-13.

Hill K, You D, Choi Y. Death distribution methods for estimating adult mortality: sensitivity analysis with simulated data errors. Demographic Research. 2009; 21:235-254.

Brass, William, 1975. Methods for Estimating Fertility and Mortality from Limited and Defective Data, Carolina Population Center, Laboratory for Population Studies, University of North Carolina, Chapel Hill.

Preston, S. H., Coale, A. J., Trussel, J. & Maxine, W. Estimating the completeness of reporting of adult deaths in populations that are approximately stable. Population Studies, 1980; v.4: 179-202

Examples

```
# The Mozambique data
res <- ddm(Moz)
head(res)
# The Brasil data
BM <- ddm(BrasilMales)
BF <- ddm(BrasilFemales)
head(BM)
head(BF)
```

ddmplot *get a quick overview of the different estimates produced*

Description

produce a dot plot, where each x position is a unique value of $cod, and points indicate the GGB, SEG, GGB-SEG, and harmonic mean of these. Feed this function the output of `ddm()`.

Usage

```
ddmplot(X, ...)
```

Arguments

`X`	output of `ddm()`.
`...`	other arguments passed to `plot()`

Value

called for its graphical device side-effects.

Examples

```
# just a rough sketch of the results!
res <- ddm(Moz)
ddmplot(res)
```

`detectAgeInterval` *Detect the age interval for some demographic data*

Description

Since death distribution methods are primarily used in adult ages, it's OK to chop off the irregular infant and child age intervals (0,1], (1,5]. Further, if high ages are in different intervals this might also be a non-issue. In principal, the user should set `MinAge` and `MaxAge` to the same values used in the death distribution methods. Here we have some defaults that should almost always return the result `5` for standard abridged data, or `1` for single age data. Really there are not any other common age-specifications, but it is best to identify these and be explicit about them. We return a warning and `NA` if more than one age interval is used. It is assumed that ages refer to the lower bounds of age intervals, as is the standard in demography.

Usage

```
detectAgeInterval(Dat, MinAge = 5, MaxAge = 70, ageColumn = "Age")
```

Arguments

`Dat`	a `data.frame` containing a column called `Age`, or codeage.
`MinAge`	integer ignore ages below this age.
`MaxAge`	integer ignore ages above this age.
`ageColumn`	character string giving the name of the Age column `"Age"` assumed.

Value

integer the age interval. `NA` if this is not unique.

`detectSex` *Detect the sex for some demographic data*

Description

The column name can be `"sex"` or `"Sex"` and nothing else. If coded with integer, the number 1 is recognized as male and numbers, 0, 2, or 6 are assumed to be female. Any other integer will throw an error. If character, if the first letter is `"f"`, then we assume female, and if the first letter is `"m"` we assume male. Case does not matter. Anything else will throw an error. This function allows for just a little flexibility.

Usage

```
detectSex(Dat, sexColumn = "Sex")
```

Arguments

`Dat`	a `data.frame` containing a column called `Sex`, or codesex.
`sexColumn`	character string giving the name of the Sex column `"Sex"` assumed.

Value

either `"f"` or `"m"`

`eOpenCD` *estimate remaining life expectancy in the open age group*

Description

This calculation is based on an indirect method to reference the Coale-Demeny West model life table. First one makes a pseudo life table deaths column using some stable pop assumptions (different in SEG vs GGB-SEG). Then take the ratio of the sum of ages 10-39 to 40-59. These ratios have been worked out for each model life table level, so we can pick the level based on the ratio we produce from the data. From there, we just pick out the remaining life expectancy that corresponds to the top age in our data, which for now hopefully is not higher than 95. The model life tables do not go higher than 95 for now, but that's well beyond the range for this method family. If your data go beyond 85 or so, then just group down to 85, say, and estimate using that instead of keeping a high open age. Called by `segMakeColumns()` and `ggbsegMakeColumns()`, and not intended for direct user interface, because you need to produce the `$deathsLT` column. You can skip calling this function by specifying `eOpen` in the top call to `seg()` or `ggbseg()`.

Usage

```
eOpenCD(codiaugmented)
```

Arguments

`codiaugmented`	the standard codi object being passed through the chain, but having been preprocessed in the course of `segMakeColumns()` or `ggbsegMakeColumns()`

Value

numeric an estimate of remaining life expectancy in the open age group

---

`fakeDates` *assume Jan 1 if no month or day given*

Description

We still require two year columns, `year1` and `year2`, at a minimum. If this function is called, and if month and day columns are missing we add these columns, with values of 1. If date columns are given, then these must be either in an unambiguous character format (`"YYYY-MM-DD"`, e.g. `"2016-05-30"` is unambiguous). Date columns will override the presence of other year, month, day columns.

Usage

```
fakeDates(X)
```

Arguments

`X`	a `data.frame` with at least columns `year1` and `year2`.

Value

X the same `data.frame`, possibly with columns for year, month, or day added.

`ggb` *estimate death registration coverage using the GGB method*

Description

Given two censuses and an average annual number of deaths in each age class between censuses, we can use stable population assumptions to estimate the degree of underregistration of deaths. The method is based on finding a best-fitting linear relationship between two modeled parameters (right term and left term), but the fit, and resulting coverage estimate, depend on exactly which age range is taken. This function either finds a nice age range for you automatically, or you can specify an exact vector of ages.

Usage

```
ggb(X, minA = 15, maxA = 75, minAges = 8, exact.ages = NULL,
  deaths.summed = FALSE)
```

Arguments

`X`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, `$age`, and `$cod` (if there are more than 1 region/sex/intercensal period).
`minA`	the lowest age to be included in search
`maxA`	the highest age to be included in search (the lower bound thereof)
`minAges`	the minimum number of adjacent ages to be used in estimating
`exact.ages`	optional. A user-specified vector of exact ages to use for coverage estimation
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Details

Census dates can be given in a variety of ways: 1) using Date classes, and column names `$date1` and `$date2` (or an unambiguous character string of the date, like, `"1981-05-13"`) or 2) by giving column names `"day1","month1","year1","day2","month2","year2"` containing integers. If only `year1` and `year2` are given, then we assume January 1 dates. If year and month are given, then we assume dates on the first of the month. If you want coverage estimates for a variety of intercensal periods/regions/by sex, then stack them, and use a variable called `$cod` with unique values for each data chunk. Different values of `$cod` could indicate sexes, regions, intercensal periods, etc. The `$deaths` column should refer to the average annual deaths for each age class in the intercensal period. Sometimes one uses the arithmetic average of recorded deaths in each age, or simply the average of the deaths around the time of census 1 and census 2. To identify an age-range in the traditional visual way, see `ggbChooseAges()`, when working with a single year/sex/region of data. The automatic age-range determination feature of this function tries to implement an intuitive way of picking ages that follows the advice typically given for doing so visually. We minimize the square of the average squared residual between the fitted line and right term.

Value

a `data.frame` with columns for the coverage coefficient `$coverage`, the minimum `$lower` and maximum `$upper` of the age range on which it is based. `$a` and `$b` give the intercept and slope of the line on which the coverage estimate is based. `$delta`, `$k1`, and `$k2` are further derived quantities that may be interesting for advanced users. Rows indicate data partitions, as indicated by the optional `$cod` variable.

References

Hill K. Estimating census and death registration completeness. Asian and Pacific Population Forum. 1987; 1:1-13.

Brass, William, 1975. Methods for Estimating Fertility and Mortality from Limited and Defective Data, Carolina Population Center, Laboratory for Population Studies, University of North Carolina, Chapel Hill.

Examples

```
# The Mozambique data
res <- ggb(Moz)
```

```
res
# The Brasil data
BM <- ggb(BrasilMales)
BF <- ggb(BrasilFemales)
head(BM)
head(BF)
```

`ggbChooseAges` *interactively determine ages to use for estimating coverage*

Description

In a spreadsheet one would typically set up the GGB method to produce a plot that updates as the user changes the age range. This function implements that kind of work flow. This will be intuitive for spreadsheet users, but it does not scale well. Imagine you have 200 territorial units, then you would not want to repeat this task. `ggb()` does the same thing automatically. You can compare the age range you select manually with the one given back by `ggb()` as a diagnostic, for instance. To set up the plot device, just give a single year/region/sex of data. By default it will give the RMSE-optimized age range to start with, but you can specify a vector of exact ages to use as well. All points are plotted, with a fitted line that has been set to a subset of the points, which is plotted in a different color. You can click any point to change the age range, and the plot updates accordingly, up to a maximum of 15 clicks so you don't waste your time. You can stop the plot by either clicking on the graphics device outside the plot area or clicking out the 15 tries (or more if you increase `maxit`).

Usage

```
ggbChooseAges(codi, minA = 15, maxA = 75, minAges = 8,
  exact.ages = NULL, maxit = 15, deaths.summed = FALSE)
```

Arguments

`codi`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, and `$age`.
`minA`	the lowest age to be included in search
`maxA`	the highest age to be included in search (the lower bound thereof)
`minAges`	the minimum number of adjacent ages to be used in estimating
`exact.ages`	optional. A user-specified vector of exact ages to use for coverage estimation.
`maxit`	the maximum number of clicks you can take. Default 15.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Details

If you want to send the results of this into `ggb()`, you can do so by setting `Exact.ages` to `seq(lower,upper,by=5)`, where `$lower`, and `$upper` are the results returned from `ggbChooseAges()` after you're done manually determining the age range.

Value

`data.frame` containing elements `$coverage`, `$lower`, `$upper`, and `ages`.

Examples

```
## Not run:
# for interactive sessions only
# *click points to adjus age range used (yellow)
# *click in margin to stop and return coverage results
ggbChooseAges(Moz)

## End(Not run)
```

`ggbcoverageFromAges` *given a set of ages, what is the implied death registration coverage?*

Description

For a single year/sex/region of data (formatted as required by `ggb()`), what is the registration coverage implied by a given age range? Called by `ggbcoverageFromYear()` and `ggbChooseAges()`.

Usage

```
ggbcoverageFromAges(codi, agesfit, deaths.summed = FALSE)
```

Arguments

`codi`	a chunk of data (single sex, year, region, etc) with all columns required by `ggb()`
`agesfit`	an integer vector of ages, either returned from `ggbgetAgesFit` or user-supplied.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Value

numeric. the estimated level of coverage.

ggbcoverageFromYear — *estimate death registration coverage for a single year/sex/region using the GGB method*

Description

Given two censuses and an average annual number of deaths in each age class between censuses, we can use stable population assumptions to estimate the degree of underregistration of deaths. The method is based on finding a best-fitting linear relationship between two modeled parameters (right term and left term), but the fit, and resulting coverage estimate, depend on exactly which age range is taken. This function either finds a nice age range for you automatically, or you can specify an exact vector of ages. Called by `ggb()`. Users probably don't need to call this directly. Just use `ggb()` instead.

Census dates can be given in a variety of ways: 1) using Date classes, and column names `$date1` and `$date2` (or an unambiguous character string of the date, like, `"1981-05-13"`) or 2) by giving column names `"day1","month1","year1","day2","month2","year2"` containing integers. If only `year1` and `year2` are given, then we assume January 1 dates. If year and month are given, then we assume dates on the first of the month.

Usage

```
ggbcoverageFromYear(codi, exact.ages = NULL, minA = 15, maxA = 75,
  minAges = 8, deaths.summed = FALSE)
```

Arguments

`codi`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, and `$age`.
`exact.ages`	optional. use an exact set of ages to estimate coverage.
`minA`	the minimum of the age range searched. Default 15
`maxA`	the maximum of the age range searched. Default 75
`minAges`	the minimum number of adjacent ages needed as points for fitting. Default 8
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Value

a `data.frame` with columns for the coverage coefficient, and the min and max of the age range on which it is based.

`ggbFittedFromAges` *make the growth-adjusted quasi life table columns required by GGB method*

Description

Called by `ggbChooseAges()` and `ggbcoverageFromYear()`. This simply modulates some code that would otherwise be repeated. Users probably don't need to call this function directly. If columns produced by `ggbMakeColumns()` are not present, then we call it here just to keep things from breaking.

Usage

```
ggbFittedFromAges(codi, agesfit, deaths.summed = FALSE)
```

Arguments

`codi`	a chunk of data (single sex, year, region, etc) with all columns required by `ggb()`
`agesfit`	an a priori set of ages for which to calculate the fit
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Value

codi, with many columns added, most importantly `$rightterm`, `$leftterm`, and `$exclude`.

`ggbgetAgesFit` *determine the age range that minimizes the mean squared error*

Description

Called by `ggbcoverageFromYear()` whenever `exact.ages` are not given. This automates what one typically does visually.

Usage

```
ggbgetAgesFit(codi, minA = 15, maxA = 75, minAges = 8,
  deaths.summed = FALSE)
```

Arguments

`codi`	a chunk of data (single sex, year, region, etc) with all columns required by `ggb()`
`minA`	the lowest age to be included in search
`maxA`	the highest age to be included in search (the lower bound thereof)
`minAges`	the minimum number of adjacent ages to be used in estimating
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Value

a vector of ages that minimizes the RMSE

See Also

codeggbChooseAges

`ggbgetRMS` *calculate the root means square of the error to help find optimal age range*

Description

Called by `ggbgetAgesFit()` whenever the user does not want to manually determine the age range used to determine registration coverage. Probably no need to be called by top-level users. If a user would rather determine the optimal age range some other way, then look to `ggbcoverageFromYear()` where `ggbgetRMS` is called and add another condition or make it call something else.

Usage

```
ggbgetRMS(agesi, codi)
```

Arguments

`agesi`	the set of ages used for this iteration
`codi`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, and `$age`.

Value

the RMSE

`ggbMakeColumns` *make the growth-adjusted quasi life table columns required by GGB method*

Description

Called by `ggbChooseAges()` and `ggbcoverageFromYear()`. This simply modulates some code that would otherwise be repeated. Users probably don't need to call this function directly.

Usage

```
ggbMakeColumns(codi, minA = 15, maxA = 75, deaths.summed = FALSE)
```

Arguments

`codi`	a chunk of data (single sex, year, region, etc) with all columns required by `ggb()`
`minA`	the minimum of the age range searched. Default 15
`maxA`	the maximum of the age range searched. Default 75
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Value

codi, with many columns added, most importantly `$rightterm`, `$leftterm`, and `$exclude`.

`ggbseg` *estimate death registration coverage using the hybrid generalized growth balance and synthetic extinct generation*

Description

Given two censuses and an average annual number of deaths in each age class between censuses, we can use stable population assumptions to estimate the degree of underregistration of deaths. The method estimates age-specific degrees of coverage. The age pattern of these is assumed to be noisy, so we take the arithmetic mean over some range of ages. One may either specify a particular age-range, or let the age range be determined automatically. If the age-range is found automatically, this is done using the method developed for the generalized growth-balance method. Part of this method relies on a prior value for remaining life expectancy in the open age group. By default, this is estimated using a standard reference to the Coale-Demeny West model life table, although the user may also supply a value. The difference between this method and `seg()` is that here we adjust census 1 part way through processing, based on some calculations similar to GGB.

Usage

```
ggbseg(X, minA = 15, maxA = 75, minAges = 8, exact.ages = NULL,
  eOpen = NULL, deaths.summed = FALSE)
```

Arguments

`X`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, `$age`, `$sex`, and `$cod` (if there are more than 1 region/sex/intercensal period).
`minA`	the lowest age to be included in search
`maxA`	the highest age to be included in search (the lower bound thereof)
`minAges`	the minimum number of adjacent ages to be used in estimating
`exact.ages`	optional. A user-specified vector of exact ages to use for coverage estimation
`eOpen`	optional. A user-specified value for remaining life-expectancy in the open age group.
`deaths.summed`	logical. Is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Details

Census dates can be given in a variety of ways: 1) using Date classes, and column names `$date1` and `$date2` (or an unambiguous character string of the date, like, `"1981-05-13"`) or 2) by giving column names `"day1","month1","year1","day2","month2","year2"` containing integers. If only `year1` and `year2` columns are given, then we assume January 1 dates. If year and month are given, then we assume dates on the first of the month. If you want coverage estimates for a variety of intercensal periods/regions/by sex, then stack them, and use a variable called `$cod` with a unique values for each data chunk. Different values of `$cod` could indicate sexes, regions, intercensal periods, etc. The `$deaths` column should refer to the average annual deaths in each age class in the intercensal period. Sometimes one uses the arithmetic average of recorded deaths in each age, or simply the average of the deaths around the time of census 1 and census 2. To identify an age-range in the traditional visual way, see `plot.ggb()`, when working with a single year/sex/region of data. The automatic age-range determination feature of this function tries to implement an intuitive way of picking ages that follows the advice typically given for doing so visually. We minimize the square of the average squared residual between the fitted line and right term. Finally, only specify `eOpen` when working with a single region/sex/period of data, otherwise the same value will be passed in irrespective of mortality and sex.

Value

a `data.frame` with columns for the coverage coefficient `$coverage`, and the minimum `$lower` and maximum `$upper` of the age range on which it is based. Rows indicate data partitions, as indicated by the optional `$cod` variable.

References

Hill K. Methods for measuring adult mortality in developing countries: a comparative review. The global burden of disease 2000 in aging populations. Research paper; No. 01.13; 2001.

Hill K, You D, Choi Y. Death distribution methods for estimating adult mortality: sensitivity analysis with simulated data errors. Demographic Research. 2009; 21:235-254.

Preston, S. H., Coale, A. J., Trussel, J. & Maxine, W. Estimating the completeness of reporting of adult deaths in populations that are approximately stable. Population Studies, 1980; v.4: 179-202

Examples

```
# The Mozambique data
res <- ggbseg(Moz)
res
# The Brasil data
BM <- ggbseg(BrasilMales)
BF <- ggbseg(BrasilFemales)
head(BM)
head(BF)
```

ggbsegCoverageFromYear *estimate death registration coverage for a single year/sex/region using the modified Bennett-Horiuchi method*

Description

Given two censuses and an average annual number of deaths in each age class between censuses, we can use stable population assumptions to estimate the degree of underregistration of deaths. The method estimates age-specific degrees of coverage. The age pattern of these is assumed to be noisy, so we take the arithmetic mean over some range of ages. One may either specify a particular age-range, or let the age range be determined automatically. If the age-range is found automatically, this is done using the method developed for the generalized growth-balance method. Part of this method relies on a prior value for remaining life expectancy in the open age group. By default, this is estimated using a standard reference to the Coale-Demeny West model life table, although the user may also supply a value. The difference between this method and `seg()` is that here we adjust census 1 part way through processing, based on some calculations similar to GGB. Called by `ggbseg()`. Users probably do not need to use this function directly.

Usage

```
ggbsegCoverageFromYear(codi, minA = 15, maxA = 75, minAges = 8,
  exact.ages = NULL, eOpen = NULL, deaths.summed = FALSE)
```

Arguments

`codi`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, `$sex`, `$age`, and `$cod` (to indicate regions, periods, sexes).
`minA`	the minimum of the age range searched. Default 15
`maxA`	the maximum of the age range searched. Default 75
`minAges`	the minimum number of adjacent ages needed as points for fitting. Default 8
`exact.ages`	optional. use an exact set of ages to estimate coverage.
`eOpen`	optional. A user-specified value for remaining life-expectancy in the open age group.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Details

Census dates can be given in a variety of ways: 1) using Date classes, and column names `$date1` and `$date2` (or an unambiguous character string of the date, like, `"1981-05-13"`) or 2) by giving column names `"day1","month1","year1","day2","month2","year2"` containing integers. If only `year1` and `year2` are given, then we assume January 1 dates. If year and month are given, then we assume dates on the first of the month.

Value

a `data.frame` with columns for the coverage coefficient, and the min and max of the age range on which it is based.

`ggbsegMakeColumns` *make the Bennett-Horiuchi quasi life table columns required by the second estimation method*

Description

Called by `ggbsegCoverageFromYear()`. This simply modulates some code that would otherwise be repeated. Users probably don't need to call this function directly.

Usage

```
ggbsegMakeColumns(codi, minA = 15, maxA = 75, agesFit, eOpen = NULL,
  deaths.summed = FALSE)
```

Arguments

`codi`	a chunk of data (single sex, year, region, etc) with all columns required by `ggbseg()`
`minA`	the minimum of the age range searched. Default 15
`maxA`	the maximum of the age range searched. Default 75
`agesFit`	vector of ages as passed in by `ggbsegCoverageFromYear)`
`eOpen`	optional. A value for remaining life expectancy in the open age group.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Details

`agesFit` is a vector passed in from `ggbsegMakeColumns()`, and it was either estimated using the GGB automatic method there, or simply came from the argument `exact.ages` specified in `ggbseg()`. By default we just automatically estimate these. `eOpen` can be either user-specified, or it will be estimated automatically using `eOpenCD()`.

Value

codi, with many columns added, most importantly `$Cx`.

`group01` *group down standard abridged data in child mortality group*

Description

We want 5-year age groups starting from 0. Standard abridged data has 0i,1,5. So we need to group together 0 and 1. Just for the sake of getting comparable results.

Usage

```
group01(X)
```

Arguments

`X`	standard input as required by `ddm()`, `ggb()`, `bh1()`, or `bh2()`

Value

X, with child ages grouped as necessary (or not)

`guessage` *which age is closest to the point clicked?*

Description

a utility function called by `ggbChooseAges()`.

Usage

```
guessage(xvec, yvec, click, age)
```

Arguments

`xvec`	`$rightterm`, as given by `ggbMakeColumns()`
`yvec`	`$lefttterm`, as given by `ggbMakeColumns()`
`click`	a point given by `locator(1)`
`age`	ages present in dataset

Value

the age corresponding to the x,y pair of `$rightterm`, `$lefttterm` closest to the point clicked.

`guessDeathsColumn` *Figure out which column is the deaths column*

Description

This function will pick up `"death"`, `"deaths"`, `"Death"`, or `"Deaths"` (and maybe some others?) and rename it `"deaths"` for easier internal usage.

Usage

```
guessDeathsColumn(X)
```

Arguments

`X`	`data.frame` that we want to check for a deaths column

Value

The same `data.frame`, returned, with the deaths column renamed as `"deaths"`

`headerPrep` *a utility function to prep the header*

Description

This is an internal utility function, to save on redundant lines of code. Not so useful for hand-processing.

Usage

```
headerPrep(X)
```

Arguments

`X`	this is any codi-style `data.frame`

Value

a list of codi chunks (by intercensal period, region, etc), with standardized names, dates, etc.

HMDlogic *Logical utility functions*

Description

These logical functions are like the usual ones, but NA values are treated as FALSE by default. This is not an exhaustive list, but these are the ones that speed our coding, and reduce code clutter. Functions copied from HMD collection directly as-is.

Usage

```
x %==% y

x %!=% y

x %>% y

x %<% y

x %>=% y

x %<=% y
```

Arguments

x, y	any two vectors that can be logically compared.

Details

Note that one of these, %>% makes this package incompatible with the magrittr package.

Examples

```
## Not run:
c(1,2,NA,4,5) == c(1,NA,3,4,NA)
# compare
c(1,2,NA,4,5) %==% c(1,NA,3,4,NA)

## End(Not run)
```

`inUSR` *does a given pairlist of x and y coordinates fall within the plot region?*

Description

Check to see if a point clicked falls in the plot or outside it. This function is used by `ggbChooseAges()`.

Usage

```
inUSR(USR, click)
```

Arguments

`USR,`	as given by `par("usr")`
`click`	a pairlist with elements `$x` and `$y`, as returned by `locator(1)`

Value

logical. `TRUE` if in the plot region.

`isLeapYear` *determine whether a year is a leap year.*

Description

In order to remove `lubridate` dependency, we self-detect leap years and adjust February accordingly.

Usage

```
isLeapYear(Year)
```

Arguments

`Year`	integer of year to query

Value

logical is the Year a leap year or not

Author(s)

Carl Boe

`Moz` *Example data for Mozambique females 1997-2007*

Description

A dataset containing 17 rows and 8 variables: Population counts for 1997 and 2007 in quinquennial age groups 0, 5, ... 75, with an open age of 80. Deaths are given as the average of the age-specific deaths in 1997 and 2007.

Usage

```
Moz
```

Format

A data frame with 17 rows and 8 variables:

cod integer a column of 1s

pop1 integer the census population count in 1997

pop2 integer the census population count in 2007

deaths integer average of 1997 and 2007 deaths

age integer lower age bound for each age group

sex character “f” for female

year1 integer 1997

year2 integer 2007

Source

Data courtesy of Bernardo Queiroz.

`reduceOpen` *chop down or group down the open age*

Description

These methods are not intended to be applied to ages greater than, say 90 or 95. Usually, we’d top out in the range 75 to 85. In any case, the Coale-Demeny life table implementation that we have only goes up to age 95, so there is a practical limitation to deriving a remaining life expectancy for the open age group. If a user tries to apply the Bennett-Horiuchi methods to data with higher open ages, stuff breaks for the time being. So this function chops the data off at `min(maxA,95)`, after having (optionally) grouped data down. This function needs to work with a single partition of data (intercensal period, sex, region, etc).

Usage

```
reduceOpen(X, maxA = 75, group = TRUE)
```

Arguments

`X`	data formatted per the requirements of `bh1()`, `bh2()`
`maxA`	integer ignore ages above this age.
`group`	logical. If `TRUE` we sum down to `min(maxA,95)`. If `FALSE`, we just chop off data above that age.

Value

X, with the open age having been reduced either with or without aggregation.

`seg` *estimate death registration coverage using the synthetic extinct generation method*

Description

Given two censuses and an average annual number of deaths in each age class between censuses, we can use stable population assumptions to estimate the degree of underregistration of deaths. The method estimates age-specific degrees of coverage. The age pattern of these is assumed to be noisy, so we take the arithmetic mean over some range of ages. One may either specify a particular age-range, or let the age range be determined automatically. If the age-range is found automatically, this is done using the method developed for the generalized growth-balance method. Part of this method relies on a prior value for remaining life expectancy in the open age group. By default, this is estimated using a standard reference to the Coale-Demeny West model life table, although the user may also supply a value.

Usage

```
seg(X, minA = 15, maxA = 75, minAges = 8, exact.ages = NULL,
  eOpen = NULL, deaths.summed = FALSE)
```

Arguments

`X`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, `$age`, `$sex`, and `$cod` (if there are more than 1 region/sex/intercensal period).
`minA`	the lowest age to be included in search
`maxA`	the highest age to be included in search (the lower bound thereof)
`minAges`	the minimum number of adjacent ages to be used in estimating
`exact.ages`	optional. A user-specified vector of exact ages to use for coverage estimation
`eOpen`	optional. A user-specified value for remaining life-expectancy in the open age group.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Details

Census dates can be given in a variety of ways: 1) using Date classes, and column names `$date1` and `$date2` (or an unambiguous character string of the date, like, `"1981-05-13"`) or 2) by giving column names `"day1","month1","year1","day2","month2","year2"` containing integers. If only `year1` and `year2` columns are given, then we assume January 1 dates. If year and month are given, then we assume dates on the first of the month. If you want coverage estimates for a variety of intercensal periods/regions/by sex, then stack them, and use a variable called `$cod` with a unique values for each data chunk. Different values of `$cod` could indicate sexes, regions, intercensal periods, etc. The `$deaths` column should refer to the average annual deaths in each age class in the intercensal period. Sometimes one uses the arithmetic average of recorded deaths in each age, or simply the average of the deaths around the time of census 1 and census 2. To identify an age-range in the traditional visual way, see `plot.ggb()`, when working with a single year/sex/region of data. The automatic age-range determination feature of this function tries to implement an intuitive way of picking ages that follows the advice typically given for doing so visually. We minimize the square of the average squared residual between the fitted line and right term. Finally, only specify `eOpen` when working with a single region/sex/period of data, otherwise the same value will be passed in irrespective of mortality and sex.

Value

a `data.frame` with columns for the coverage coefficient `$coverage`, and the minimum `$lower` and maximum `$upper` of the age range on which it is based. Rows indicate data partitions, as indicated by the optional `$cod` variable. `$l25` (`$u25`) give the mean of the lower (upper) quartile of the distribution of age-specific coverage estimates.

References

Bennett Neil G, Shiro Horiuchi. Estimating the completeness of death registration in a closed population. Population Index. 1981; 1:207-221.

Preston, S. H., Coale, A. J., Trussel, J. & Maxine, W. Estimating the completeness of reporting of adult deaths in populations that are approximately stable. Population Studies, 1980; v.4: 179-202

Examples

```
# The Mozambique data
res <- seg(Moz)
res
# The Brasil data
BM <- seg(BrasilMales)
BF <- seg(BrasilFemales)
head(BM)
head(BF)
```

`segCoverageFromAges` *given a set of ages, what is the implied death registration coverage?*

Description

For a single year/sex/region of data (formatted as required by `seg()`, `ggbseg()`), what is the registration coverage implied by a given age range? Called by `segCoverageFromYear()` and `ggbsegCoverageFromYear()`. Here, the function simply takes the arithmetic mean of a given age range of `$Cx`, as returned by `segMakeColumns()` or `ggbsegMakeColumns()`. Not intended for top-level use.

Usage

```
segCoverageFromAges(codi, agesFit)
```

Arguments

`codi`	a chunk of data (single sex, year, region, etc) with all columns required by `ggb()`
`agesFit`	an integer vector of ages, either returned from `ggbgetAgesFit` or user-supplied.

Value

numeric. the estimated level of coverage.

`segCoverageFromYear` *estimate death registration coverage for a single year/sex/region using the Bennett-Horiuchi method*

Description

Given two censuses and an average annual number of deaths in each age class between censuses, we can use stable population assumptions to estimate the degree of underregistration of deaths. The method estimates age-specific degrees of coverage. The age pattern of these is assumed to be noisy, so we take the arithmetic mean over some range of ages. One may either specify a particular age-range, or let the age range be determined automatically. If the age-range is found automatically, this is done using the method developed for the generalized growth-balance method. Part of this method relies on a prior value for remaining life expectancy in the open age group. By default, this is estimated using a standard reference to the Coale-Demeny West model life table, although the user may also supply a value. Called by `seg()`. Users probably do not need to use this function directly.

Usage

```
segCoverageFromYear(codi, minA = 15, maxA = 75, minAges = 8,
  exact.ages = NULL, eOpen = NULL, deaths.summed = FALSE)
```

Arguments

`codi`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, `$sex`, `$age`, and `$cod` (to indicate regions, periods, sexes).
`minA`	the minimum of the age range searched. Default 15
`maxA`	the maximum of the age range searched. Default 75
`minAges`	the minimum number of adjacent ages needed as points for fitting. Default 8
`exact.ages`	optional. use an exact set of ages to estimate coverage.
`eOpen`	optional. A user-specified value for remaining life-expectancy in the open age group.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Details

Census dates can be given in a variety of ways: 1) using Date classes, and column names `$date1` and `$date2` (or an unambiguous character string of the date, like, `"1981-05-13"`) or 2) by giving column names `"day1","month1","year1","day2","month2","year2"` containing integers. If only `year1` and `year2` are given, then we assume January 1 dates. If year and month are given, then we assume dates on the first of the month.

Value

a `data.frame` with columns for the coverage coefficient, and the min and max of the age range on which it is based.

`segMakeColumns` *make the Bennett-Horiuchi quasi life table columns required by the estimation method*

Description

Called by `segCoverageFromYear()`. This simply modulates some code that would otherwise be repeated. Users probably don't need to call this function directly.

Usage

```
segMakeColumns(codi, minA = 15, maxA = 75, eOpen = NULL,
  deaths.summed = FALSE)
```

Arguments

`codi`	a chunk of data (single sex, year, region, etc) with all columns required by `seg()`
`minA`	the minimum of the age range searched. Default 15
`maxA`	the maximum of the age range searched. Default 75
`eOpen`	optional. A value for remaining life expectancy in the open age group.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Value

codi, with many columns added, most importantly `$Cx`.

`segplot` *plot the age-pattern of coverage estimates*

Description

the SEG method works by averaging the coverage estimates over a range of ages. Users may wish to see the age pattern for diagnostic purposes.

Usage

```
segplot(X, minA = 15, maxA = 75, minAges = 8, exact.ages = NULL,
  eOpen = NULL, deaths.summed = FALSE, log = FALSE)
```

Arguments

`X`	`data.frame` with columns, `$pop1`, `$pop2`, `$deaths`, `$date1`, `$date2`, `$age`, `$sex`, and `$cod` (if there are more than 1 region/sex/intercensal period).
`minA`	the lowest age to be included in search
`maxA`	the highest age to be included in search (the lower bound thereof)
`minAges`	the minimum number of adjacent ages to be used in estimating
`exact.ages`	optional. A user-specified vector of exact ages to use for coverage estimation
`eOpen`	optional. A user-specified value for remaining life-expectancy in the open age group.
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.
`log`	logical. should we log the y axis?

Details

All arguments are essentially the same as those given to `seg()`

Value

Function called for its graphical side effects

Examples

```
## Not run:
segplot(Moz)

## End(Not run)
```

`single2abr` *single ages to standard abridged ages*

Description

convert ages of the form 0,1,2,3,4,5,... into 0,1,1,1,1,5,...

Usage

```
single2abr(x)
```

Arguments

`x`	vector of single ages (lower bound) a.k.a. completed age.

Value

vector of the same length indicating which abridged age group each single age belongs to (lower bound)

`slopeint` *get the slope the slope and intercept implied by a set of ages*

Description

Called by `ggbFittedFromAges()` and `ggbChooseAges()`

Usage

```
slopeint(codi, agesfit, deaths.summed = FALSE)
```

Arguments

`codi`	`data.frame` as produced by `ggbMakeColumns()`
`agesfit`	a set of ages to estimate coverage from
`deaths.summed`	logical. is the deaths column given as the total per age in the intercensal period (`TRUE`). By default we assume `FALSE`, i.e. that the average annual was given.

Value

a pairlist with elements `$a` for the intercept and `$b` for the slope

`yint` *get interval as fraction of full years*

Description

Either assume 365 days in the year, or get the precise duration.

Usage

```
yint(Day1, Month1, Year1, Day2, Month2, Year2, reproduce.matlab = FALSE,
  detect.mid.year = TRUE, detect.start.end = TRUE)
```

Arguments

`Day1`	Day of first date
`Month1`	Month of first date
`Year1`	Year of first date
`Day2`	Day of second date
`Month2`	Month of second date
`Year2`	Year of second date
`reproduce.matlab`	logical. default `FALSE`. Assume 365 days in all years?
`detect.mid.year`	logical. default `TRUE`. Should June 30 and July 1 be considered .5?
`detect.start.end`	logical. default `TRUE`. Should Jan 1 always be 0 and Dec 31 always be 1?

Value

decimal value of year fraction (can be greater than 1)

`yint2` *get the time interval without having to specify so many arguments*

Description

We accept dates, and fake them otherwise. Dates must be unique. Iterate over data if necessary for multiple intervals.

Usage

```
yint2(X)
```

Arguments

`X`	`data.frame` with at least columns `$date1` and `$date2`, or `$year1` and `$year2`.

Value

an decimal year value of the time between two dates.

`ypart` *determine the proportion of a year passed as of a particular date*

Description

The fraction returned by this is used e.g. for intercensal estimates. Function uses 'lubridate' package to handle dates elegantly.

Usage

```
ypart(Year, Month, Day, reproduce.matlab = TRUE, detect.mid.year = FALSE,
  detect.start.end = TRUE)
```

Arguments

`Year`	4-digit year (string or integer)
`Month`	month digits (string or integer, 1 or 2 characters)
`Day`	Day of month digits (string or integer, 1 or 2 characters)
`reproduce.matlab`	logical. Default TRUE. Assume 365 days in a year.
`detect.mid.year`	logical. if `TRUE`, June 30 or July 1 will always return .5.
`detect.start.end`	logical. default `TRUE`. Should Jan 1 always be 0 and Dec 31 always be 1?

Index