Package 'DemoDecomp'

September 20, 2024

Type Package

Title Decompose Demographic Functions

Version 1.14.1

Date 2024-09-20

Description Three general demographic decomposition methods: Pseudo-continuous decomposition proposed by Horiuchi, Wilmoth, and Pletcher (2008) <doi:10.1353/dem.0.0033>, stepwise replacement decomposition proposed by Andreev, Shkolnikov and Begun (2002) <doi:10.4054/DemRes.2002.7.14>, and lifetable response experiments proposed by Caswell (1989) <doi:10.1016/0304-3800(89)90019-7>.

License GPL-3

LazyLoad yes

RoxygenNote 7.3.1

Suggests covr

RdMacros Rdpack

Imports Rdpack, numDeriv

Depends R (>= 2.10)

BugReports https://github.com/timriffe/DemoDecomp/issues

Encoding UTF-8

LazyData true

NeedsCompilation no

Author Tim Riffe [aut, cre]

Maintainer Tim Riffe <tim.riffe@gmail.com>

Repository CRAN

Date/Publication 2024-09-20 09:20:05 UTC

Contents

`horiuchi` *Numeric Approximation of Continuous Decomposition*

Description

This is an exact R implementation of the decomposition code in Matlab offered by the authors in the supplementary material given here: <http://www.demog.berkeley.edu/~jrw/Papers/decomp.suppl.pdf>. The difference between `DecompContinuous()` and this function is that `DecompContinuousOrig` takes `rates1` and `rates2` as single vectors, rather than as matrices, and output is also returned as a vector. This difference makes the function more flexible, but may add a step when writing the function to be decomposed. See examples.

Usage

```
horiuchi(func, pars1, pars2, N, ...)
```

Arguments

`func`	A function specified by the user. This must be able to take the vectors `rates1` or `rates2` as its argument, and to return the value of the function, `y`, when evaluated for these rates. It may also have additional arguments, not to be decomposed.
`pars1`	vector of covariates to be passed on as arguments to `func()`. Covariates can be in any order, as long as `func()` knows what to do with them. `pars1` is for time 1 (or population 1).
`pars2`	is the same as `pars2` but for time/population 2.
`N`	The number of intervals to integrate over.
`...`	optional parameters to pass on to `func()`. These are not decomposed.

Details

The decomposition works by assuming a linear change in all parameters between `pars1` and `pars2`. At each small step approaching time 2 (the size of which is `1/N`) each parameter is moved forward along its linear trajectory. One at a time, each covariate (of which there are ages*variables of) is switched out twice, once for its value at 1/(2N) forward and once for its value at 1/(2N) backward in time. The difference between `func()` evaluated with these two rate matrices is the change in

yattributable to that particular covariate and that particular time step. Summing over all N time steps, we get the contribution to the difference of each covariate, `effectmat`. The sum of `effectmat` should come very close to `func(rates2)-func(rates1)`. The error decreases with larger `N`, but there is not much point in having an `N` higher than 100, and 20 is usually sufficient. This ought to be able to handle a very wide variety of functions.

If `pars1` are observations from 2005 and `pars2` are observations from 2006 an `N` of 20 would imply a delta of 1/20 of a year for each integration step. Higher `N` provides finer results (a lower total residual), but takes longer to compute. In general, there are decreasing returns to higher `N`. `sum(effectmat)` ought to approximate `func(rates2)-func(rates1)`.

Value

returns `effectmat`, a matrix of the variable effects that is organized in the same way as `pars1` and `pars2`.

References

Andreev EM, Shkolnikov VM, Begun AZ (2002). "Algorithm for decomposition of differences between aggregate demographic measures and its application to life expectancies, healthy life expectancies, parity-progression ratios and total fertility rates." *Demographic Research*, **7**, 499–522.
Andreev EM, Shkolnikov VM, Begun AZ (2002). "Algorithm for decomposition of differences between aggregate demographic measures and its application to life expectancies, healthy life expectancies, parity-progression ratios and total fertility rates." *Demographic Research*, **7**, 499–522.

Examples

```
data(rates1)
data(rates2)

# we need rates1 and rates2 as vectors
rates1 <- c(rates1)
rates2 <- c(rates2)
# look at the function:
R0vec
# 2 things to point out:
# 1) it has an argument pfem, proportion female of births (1/(1+SRB)),
#    that must be specified, but that we don't care about decomposing
# 2) x is a single vector. The the inside of the function needs to
#    either refer to parts of it by indexing, as done here, or else
#    re-assign x to various objects. In this case x[1:l] is Lx and
#    x[(l+1):(2*l)] is Fx...
A <- horiuchi(func = R0vec,
              pars1 = rates1,
              pars2 = rates2,
              N = 10,
              pfem = .4886)
# the output, A, is also a single vector. Each element corresponds
# to the effect of changes in that particular covariate toward the
# overall change in the function value. sum(A) should be close to
# original difference
(check1 <- R0vec(rates2) - R0vec(rates1))
```

```
(check2 <- sum(A))



# This package does not supply default plotting functions, but one
# strategy might be the following:

# reorder A into a matrix (sideways):
A <- t(matrix(A,ncol=2))
# call barplot() twice, once for positive values and again for
# negative values
Apos <- A * .5 * (sign(A) + 1)
Aneg <- A * .5 * abs(sign(A) - 1)
## Not run:
barplot(Apos,
        width = rep(1, length(A) / 2),
        space = 0,
        ylim = range(A),
        main = "A fake decomposition of R0",
        col=c("yellow","green"),
        axisnames = FALSE,
        xlim=c(0, 90),
        ylab = "contrib to change in R0",
        cex.axis = .7)
barplot(Aneg,
        width = rep(1, length(A) / 2),
        add = TRUE,
        space = 0,
        col = c("yellow", "green"),
        axes = FALSE, axisnames = FALSE)
segments(seq(from=0,to=90,by=10),0,seq(from=0,to=90,by=10),-.027,lty=2,col="grey")
text(seq(from=0,to=90,by=10),-.027,seq(from=0,to=90,by=10),pos=1,xpd=T)
legend("bottomright",fill=c("yellow","green"),legend=c("contrib from change in Lx",
"contrib from change in Fx"),title="age specific contrib of changes in Fx and Lx",bg="white")

## End(Not run)
```

LTabr *an abridged lifetable based on M(x)*

Description

Implements the abridged lifetable formulas given in the supplementary material to Andreev et. al. (2012). An entire lifetable is calculated, but only life expectancy at birth is returned.

Usage

```
LTabr(Mx, Age = c(0, 1, cumsum(rep(5, length(Mx) - 2))), radix = 1e+05)
```

Arguments

Mx	numeric vector of abridged mortality rates.
Age	integer, abridged age lower bounds.
radix	numeric. Can be anything positive.

Details

Chiang's a(x) is assumed in the following way: $a(0) = 0.07 + 1.7 * M(0)$, $a(1) = 1.6$, $a(\omega) = \frac{1}{M(\omega)}$, and all others are assumed at mid interval. The last age is assumed open. Everything else is pretty standard.

Value

numeric life expectancy at birth

References

Andreev EM, Shkolnikov VM, Begun AZ (2002). "Algorithm for decomposition of differences between aggregate demographic measures and its application to life expectancies, healthy life expectancies, parity-progression ratios and total fertility rates." *Demographic Research*, **7**, 499–522.
Andreev EM, Shkolnikov VM (2012). "An Excel spreadsheet for the decomposition of a difference between two values of an aggregate demographic measure by stepwise replacement running from young to old ages." *Max Planck Institute for Demographic Research (MPIDR Technical Report TR–2012–002)*.

---

ltre *Caswell's LTRE method of decomposition*

Description

Caswell's Lifetable Response Experiment (LTRE) decomposed a vector-parameterized function by taking derivatives of the objective function with respect to each parameter. The sum-product of the resulting derivative vector and the change in parameter values is a first order approximation of the decomposition. This implementation repeats this operation N times as pars1 warps into pars2 over N steps. This allows for arbitrary precision as N increases, as in the case of the Horiuchi approach.

Usage

```
ltre(func, pars1, pars2, dfunc, N = 20, ...)
```

Arguments

func	A function specified by the user. This must be able to take the vectors rates1 or rates2 as its argument, and to return the value of the function, y, when evaluated for these rates. It may also have additional arguments, not to be decomposed.

pars1	vector of covariates to be passed on as arguments to func(). Covariates can be in any order, as long as func() knows what to do with them. pars1 is for time 1 (or population 1).
pars2	is the same as pars2 but for time/population 2.
dfunc	a derivative function, see details
N	The number of intervals to integrate over.
...	...optional parameters to pass on to func(). These are not decomposed. Also one can use this argument to pass optional arguments to numDeriv::grad().

Details

The case of N=1 differentiates with respect to the arithmetic mean of pars1 and pars2. The ... argument can be used to send extra parameters to func() that do not get decomposed, or to specify other optional arguments to numDeriv::grad() for finer control.

The argument dfunc is optional. If given, it should be a function written to have a first argument func, a second argument x, which consists in the vector of decomposed parameters (same layout at pars1 and pars2), and an option ... argument for undecomposed parameters. Presumably if a derivative function is given then it is analytic or somehow a more parsimonious calculation than numeric derivatives. If left unspecified numDeriv::grad() is used.

As with horiuchi(), the path from pars1 to pars2 is linear, but other paths can be induced by parameterizing func() differently. For example, if you want proportional change from pars1 to pars2 then log them, and write func() to first antilog before continuing. This is not zero-friendly, but in practice power transforms give close results, so you could sqrt() and then square inside func(). If you do this, then dfunc() must be written to account for it too, or you could stick with the default numeric gradient function.

References

Caswell H (1989). "Analysis of life table response experiments I. Decomposition of effects on population growth rate." *Ecological Modelling*, **46**(3-4), 221–237. Caswell H (2006). "Matrix population models." *Encyclopedia of Environmetrics*, **3**.

See Also

grad

Mxc1 *Year 2002 death rates by cause for US males in abridged age classes*

Description

A matrix containing death rates for six causes (one of which is other) for abridged age classes 0-85. Ages are labelled in rows, and causes in column names.

Usage

```
Mxc1
```

Format

A matrix with 19 rows and 6 columns

Source

https://www.demogr.mpg.de/papers/technicalreports/tr-2010-002-files.zip

---

Mxc2 — *Year 2002 death rates by cause for England and Wales males in abridged age classes*

Description

A matrix containing death rates for six causes (one of which is other) for abrdged age classes 0-85. Ages are labelled in rows, and causes in column names.

Usage

```
Mxc2
```

Format

A matrix with 19 rows and 6 columns

Source

https://www.demogr.mpg.de/papers/technicalreports/tr-2010-002-files.zip

---

Mxc2e0abr — *get life expectancy at birth from an (abridged)age-cause matrix*

Description

Given a matrix with abridged ages in rows and causes of death in columns, then calculate life expectancy at birth using `LTabr()`.

Usage

```
Mxc2e0abr(Mxc)
```

Arguments

Mxc	numeric matrix

Details

This assumes that the marginal row sums give all-cause mortality rates. Give an other category if you need to top-up to all-cause mortality. Do not include all-cause mortality itself!

Value

numeric life expectancy at birth

`Mxc2e0abrvec` *get life expectancy at birth from the vec of an age-cause matrix*

Description

Given a vector with abridged ages stacked within causes of death, assign its dimensions, take the age marginal sums using `Mxc2e0abr`, then calculate life expectancy at birth using `LTabr()`.

Usage

```
Mxc2e0abrvec(Mxcvec, dims, trans = FALSE)
```

Arguments

`Mxcvec`	numeric vector, `c(Mxc)`.
`dims`	integer vector of length two, `c(nrow(Mxc),ncol(Mxc))`.
`trans`	do we need to transpose in order to arrive back to an age-cause matrix?

Details

This assumes that the marginal row sums give all-cause mortality rates. Give an other category if you need to top-up to all-cause mortality. Do not include all-cause mortality itself! `length(Mxcvec)` must equal `prod(dim(Mxc))`. This function is meant to be fed to a generic decomposition function, such as `stepwise_replacement()`, or `DecompContinuousOrig()`.

Value

numeric life expectancy at birth

R0vec *R0vec Calculates net reproduction, R0, according to a given set of rates Lx,fx and a fixed proportion female of births,* pfem.

Description

This function is only provided for the examples of horiuchi(). It calculates the sum of the row products of rates multiplied by pfem.

Usage

```
R0vec(x, pfem = 0.4886)
```

Arguments

x	a single vector containing Lx followed by Fx or vice versa. Here, Lx is the survival function integrated within each age interval and with a lifetable radix of 1. Fx is the fertility function, calculated as births/ person years of exposure. Fx should simply contain zeros in ages with no fertility, OR, all vectors should be limited to reproductive ages. Both Lx and Fx should for this function be of the same length.
pfem	the proportion female of births. Something like .49, .48, or (1/(2.05)). This can either be specified as a single number, or it may be allowed to vary by age. For the later case, be sure to specify a value for each age (length(x)/2 values). Default .4886.

Details

The main feature that functions need to have when specified for horiuchi() or stepwise_replacement() is that the rates must all go into a (potentially long) vector, probably consisting in your rate vectors one after the other. Really the decomposition function does not care how things are arranged in the vector- the components of change vector that is returned from horiuchi() will be arranged in exactly the same way as its input rate vectors, so as long as you know how to sort it out, and your function can extract what it needs from the vectors, then it can be specified in any way. For this particular example function, R0vec(), x must be specified with either Lx followed by Fx or vice versa. It would also be possible to redefine the function to place pfem in with the rates vector, x, which would allow this item to be decomposed too. Here it is specified separately in order to demonstrate passing on parameters to the function within horiuchi().

Value

the value of R0 for the given set of rates and proportion female of births.

Examples

```
data(rates1)
# take vec:
x <- c(rates1)
R0vec(x)
```

`rates1` *Fake data generated for horiuchi example.*

Description

These are used to calculate the net reproductive ratio (NRR)

Usage

```
rates1
```

Format

numeric vector of hypothetical fertility and mortality rates

Fx age specific fertility rates at time point 1

Source

Simulated values

Examples

```
## Not run:
data(rates1)
data(rates2)
# nothing fancy
# compare Lx
plot(rates1[,1],type='l',col="blue")
lines(rates2[,1],col="green")
# compare Fx
plot(rates1[,2],type='l',col="blue")
lines(rates2[,2],col="green")

## End(Not run)
```

rates2 *Fake data generated for horiuchi example.*

Description

These are used to calculate the net reproductive ratio (NRR)

Usage

```
rates2
```

Format

numeric vector of hypothetical fertility and mortality rates

Lx a discrete survival function at time point 2

Fx age specific fertility rates at time point 2

Source

Simulated values

Examples

```
## Not run:
data(rates1)
data(rates2)
# nothing fancy
# compare Lx
plot(rates1[,1],type='l',col="blue")
lines(rates2[,1],col="green")
# compare Fx
plot(rates1[,2],type='l',col="blue")
lines(rates2[,2],col="green")

## End(Not run)
```

stepwise_replacement *implementation of the decomposition algorithm of stepwise replacement*

Description

This implements the algorithm described in Andreev et al (2002), with defaults set to approximate their recommendations for replacement ordering and result averaging.

Usage

```
stepwise_replacement(
  func,
  pars1,
  pars2,
  symmetrical = TRUE,
  direction = "up",
  ...
)
```

Arguments

func	A function specified by the user. This must be able to take the vectors pars1 or pars2 as its argument, and to return the value of the function, y, when evaluated for these rates. It may also have additional arguments, not to be decomposed.
pars1	vector of covariates to be passed on as arguments to func(). Covariates can be in any order, as long as func() knows what to do with them. pars1 is for time 1 (or population 1).
pars2	is the same as pars1 but for time/population 2.
symmetrical	logical. default TRUE as recommended by authors. Shall we average the results of replacing 1 with 2 and 2 with 1?
direction	character. One of "up", "down", or "both". Default "up", as recommended by authors.
...	optional parameters to pass on to func().

Details

The symmetrical argument toggles whether or not we replace pars1 with pars2 (FALSE), or take the arithmetic average or replacement in both directions. direction refers to whether we go from the bottom up or top down, or take the arithmetic average of these when replacing vector elements. Although the total difference will always sum correctly, the calculated contribution from individual components can vary greatly depending on the order in general. Defaults are set to symmetrically replace from the bottom up, per the authors' suggestion.

Value

a matrix of the variable effects that is organized in the same way as pars1 and pars2.

References

Horiuchi S, Wilmoth JR, Pletcher SD (2008). "A decomposition method based on a model of continuous change." *Demography*, **45**(4), 785–801. Andreev EM, Shkolnikov VM (2012). "An Excel spreadsheet for the decomposition of a difference between two values of an aggregate demographic measure by stepwise replacement running from young to old ages." *Max Planck Institute for Demographic Research (MPIDR Technical Report TR–2012–002)*.

Examples

```
data(Mxc1)
data(Mxc2)
# we'll want to pass in these dimensions
dims  <- dim(Mxc1)
# we need parameters in vec form
Mxc1v <- c(Mxc1)
Mxc2v <- c(Mxc2)
B     <- stepwise_replacement(func = Mxc2e0abrvec,
pars1 = Mxc1v, pars2 = Mxc2v, dims = dims,
# authors' recommendations:
symmetrical = TRUE, direction = "up")
dim(B) <- dims
# the output, B, is also a single vector. Each element corresponds
# to the effect of changes in that particular covariate toward the
# overall change in the function value. sum(B) should equal the
# original difference
(check1 <- Mxc2e0abr(Mxc2) - Mxc2e0abr(Mxc1))
(check2 <- sum(B))


# This package does not supply default plotting functions, but one
# strategy might be the following:
## Not run:
Age <- c(0, 1, seq(5, 85, by = 5))
matplot(Age, B, type = 'l',
xlab = "Age", ylab = "Contrib to diff in e(0)", col = 1:6)
legend("bottomleft",lty=1:5,col=1:6,
         legend = c("Neoplasms","Circulatory","Respiratory",
     "Digestive","Acc/viol","Other"))

## End(Not run)
```

Index