Package ‘DasGuptR’

July 21, 2025

Type Package

Title Das Gupta Standardisation and Decomposition

Version 2.1.0

Maintainer Josiah King <josiah.king@ed.ac.uk>

Description Implementation of Das Gupta's standardisation and decomposition of population rates, as set out ``Standardization and decomposition of rates: A user’s manual'', Das Gupta (1993) <https://www2.census.gov/library/publications/1993/demographics/p23-186.pdf>. The goal of these methods is to calculate adjusted rates based on compositional 'factors' and quantify the contribution of each factor to the difference in crude rates between populations. The package offers functionality to handle various scenarios for any number of factors and populations, where said factors can be comprised of vectors across sub-populations (including cross-classified population breakdowns), and with the option to specify user-defined rate functions.

License GPL (>= 3)

URL https://github.com/josiahpjking/DasGuptR

BugReports https://github.com/josiahpjking/DasGuptR/issues

Depends R (>= 4.1.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Suggests R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

Author Josiah King [aut, cre],
Ben Matthews [aut]

Repository CRAN

Date/Publication 2025-04-07 16:10:12 UTC

Contents

`ccwrap` *Wrapper for cross-classified data that standardises rates across a pair of populations. Because these are (r+r')/2 * Q(a_i), this requires 1) doing the rate standardisation on each sub-population, 2) performing the standardisation on the cross classified structure variables, 3) multiplying and (optionally) aggregating up*

Description

Wrapper for cross-classified data that standardises rates across a pair of populations. Because these are (r+r')/2 * Q(a_i), this requires 1) doing the rate standardisation on each sub-population, 2) performing the standardisation on the cross classified structure variables, 3) multiplying and (optionally) aggregating up

Usage

```
ccwrap(
  pw,
  pop,
  factors,
  id_vars,
  crossclassified,
  agg,
  ratefunction = NULL,
  quietly = TRUE
)
```

Arguments

- `pw` — dataframe containing two populations worth of factor data, with columns specifying 1) population and 2) each rate-factor to be considered. must have column named "pop" indicating the population ID.
- `pop` — name (character string) of variable indicating population
- `factors` — names (character vector) of variables indicating compositional factors
- `id_vars` — character vector of variables indicating sub-populations
- `crossclassified` — character string of variable indicating size of sub-population. If specified, the proportion of each population in a given sub-population (e.g. each age-sex combination) is re-expressed as a product of symmetrical expressions representing the different variables (age, sex) constituting the sub-populations.
- `agg` — logical indicating whether, when cross-classified data is used, to output should be aggregated up to the population level
- `ratefunction` — user defined character string in R syntax that when evaluated specifies the function defining the rate as a function of factors. if NULL then will assume rate is the product of all factors.
- `quietly` — logical indicating whether interim messages should be outputted indicating progress through the P factors

Value

data.frame that includes K-a standardised rates for each population and each factor a, along with differences between standardised rates

---

`dg2pop` *Standardisation and decomposition of rates over K rate-factors and 2 populations. We suggest using dgnpop, which will internally call this function.*

Description

Standardisation and decomposition of rates over K rate-factors and 2 populations. We suggest using dgnpop, which will internally call this function.

Usage

```
dg2pop(pw, pop, factors, id_vars, ratefunction = NULL, quietly = TRUE)
```

Arguments

`pw`	dataframe containing two populations worth of factor data, with columns specifying 1) population and 2) each rate-factor to be considered. must have column named "pop" indicating the population ID.
`pop`	name (character string) of variable indicating population
`factors`	names (character vector) of variables indicating compositional factors
`id_vars`	character vector of variables indicating sub-populations
`ratefunction`	user defined character string in R syntax that when evaluated specifies the function defining the rate as a function of factors. if NULL then will assume rate is the product of all factors.
`quietly`	logical indicating whether interim messages should be outputted indicating progress through the P factors

Value

named list along set of K factors included in the standardisation. Each list element contains a data.frame that includes K-a standardised rates for each population, along with differences between standardised rates

---

dg354 *Das Gupta equation 3.54. internal function called by dg2pop.*

Description

Das Gupta equation 3.54. internal function called by dg2pop.

Usage

```
dg354(df2, i, pop, factors, id_vars, ratefunction, quietly = TRUE)
```

Arguments

`df2`	list of 2 population dataframes, in which each one contains data for all factors for the relevant population, along with variables indicating population and sub-populations
`i`	the index of the factors vector which is not being adjusted for (the alpha in "P-alpha standardised rates")
`pop`	name (character string) of variable indicating population
`factors`	names (character vector) of variables indicating compositional factors
`id_vars`	character vector of variables indicating sub-populations
`ratefunction`	user defined character string in R syntax that when evaluated specifies the function defining the rate as a function of factors. if NULL then will assume rate is the product of all factors.
`quietly`	logical indicating whether interim messages should be outputted indicating progress

Value

data.frame object including K-a standardised rates for each population for given factor a, along with differences between standardised rates

---

`dg611` *Das Gupta equation 6.11: Standardises rates across populations*

Description

Das Gupta equation 6.11: Standardises rates across populations

Usage

```
dg611(srates, all_p, y, factor)
```

Arguments

`srates`	a dataframe/tibble object of standardised rates from dg2pop
`all_p`	character or numeric vector of all N populations
`y`	character/numeric indicating a single population
`factor`	string indicating rate-factor being standardised.

Value

data.frame object including K-a standardised rates for each population for given factor a, across N populations

---

`dg612` *Das Gupta equation 6.12 for a differences between 2 populations when standardised across N populations.*

Description

Das Gupta equation 6.12 for a differences between 2 populations when standardised across N populations.

Usage

```
dg612(srates, all_p, ps, factor)
```

Arguments

`srates`	a dataframe output from dg2p
`all_p`	character or numeric vector of all N populations
`ps`	vector of length 2 specifying a possible pairwise comparison of populations
`factor`	character string indicating name of factor

Value

data.frame object including K-a standardised-rate-differences for each population for given factor a, across N populations

---

`dgcc` *Das Gupta equation 5.36 across N populations: Decomposes cross-classified population structures into a set of symmetric proportions indicating contribution of individual structural variables.*

Description

Das Gupta equation 5.36 across N populations: Decomposes cross-classified population structures into a set of symmetric proportions indicating contribution of individual structural variables.

Usage

```
dgcc(x, pop, id_vars, crossclassified)
```

Arguments

`x`	dataframe consisting of one population, including variables indicating cross-classified structure, and a variable indicating size of each cell
`pop`	variable name (character string) containing population identifier
`id_vars`	character vector of variables indicating cross-classified structure.
`crossclassified`	variable name (character string) containing cell sizes or proportions

Value

inputted data.frame is returned with the addition of variables for each of the the cross-classified variables representing the contribution to the population size.

`dgnpop` *Prithwis Das Gupta's 1993 standardisation and decomposition of rates over K rate-factors and N populations.*

Description

Prithwis Das Gupta's 1993 standardisation and decomposition of rates over K rate-factors and N populations.

Usage

```
dgnpop(
  x,
  pop,
  factors,
  id_vars = NULL,
  crossclassified = NULL,
  ratefunction = NULL,
  agg = TRUE,
  baseline = NULL,
  quietly = TRUE,
  diffs = FALSE
)
```

Arguments

`x`	dataframe or tibble object, with columns specifying 1) population, 2) each rate-factor to be considered, and (optionally) 3) variables indicating underlying sub-populations
`pop`	name (character string) of variable indicating population
`factors`	names (character vector) of variables indicating compositional factors
`id_vars`	character vector of variables indicating sub-populations
`crossclassified`	character string of variable indicating size of sub-population. If specified, the proportion of each population in a given sub-population (e.g. each age-sex combination) is re-expressed as a product of symmetrical expressions representing the different variables (age, sex) constituting the sub-populations. These expressions are then used as compositional factors in the standardisation. If NULL, then providing a single variable as a compositional factor that represents the proportion of the population in each given sub-population will combine the contribution of all sub-population variables.
`ratefunction`	user defined character string in R syntax that when evaluated specifies the function defining the rate as a function of factors. if NULL then will assume rate is the product of all factors. When sub-populations are provided, this should aggregate to a summary value (e.g., for the simple product rate this should be provided as `"sum(A*B*C*)"`.). User-defined functions can also be provided, as whatever string is given here will be parsed and evaluated as any other R code (see example eg4.4).
`agg`	logical indicating whether, when cross-classified data is used, to output should be aggregated up to the population level
`baseline`	baseline population to standardise against. if NULL then will do Das Gupta's full N-population standardisation.
`quietly`	logical indicating whether interim messages should be outputted indicating progress through the K factors and N populations
`diffs`	logical indicating whether to return list of standardised rates and rate-differences, or just the standardised rates.

Details

Population rates are often composed of various different compositional factors. Standardisation techniques calculate the rate were a set of factors to be held constant (either with a specific population as standard, or at the average of the populations). Decomposition methods quantify the amount of the difference between two population crude rate that is due to differences in population characteristics.

Das Gupta's general solution for the decomposition of two rates can be written as:

$$\Delta\text{crude-}r = \sum_{\vec{\alpha} \in K} Q(\vec{\alpha}^{p}) - Q(\vec{\alpha}^{p'})$$

Where K is the set of factors $\alpha, \beta, ..., \kappa$, which may take the form of vectors over sub-populations i. $Q(\vec{\alpha}^{p})$ denotes the rate in population p holding all factors other than α — $K \setminus \alpha$ — equal (standardised across populations p and p'). The total crude rate difference is the sum of all standardised-rate differences, and the standardisation Q is expressed as:

$$Q(\vec{\alpha}^{p}) = \sum_{j=1}^{\lfloor \frac{|K|}{2} \rfloor} \frac{\sum_{L \in \binom{K \setminus \{\alpha\}}{j-1}} f(\{L^{p}, (K \setminus L)^{p'}, \vec{\alpha}^{p}\}) + f(\{L^{p'}, (K \setminus L)^{p}, \vec{\alpha}^{p}\})}{|K| \binom{|K|-1}{j-1}}$$

Where $f(K)$ is the function that defines the calculation of the rate

Value

data.frame containing K-a standardised rates (or differences) for each population.

- `rate`: standardised rate such that factor a is from population p and all other factors are averaged across populations, f(a^p,...)
- `pop`: population p for which factor a is taken from
- `std.set`: set of N populations (minus p) across which the standardisation has been performed
- `factor`: name of factor a that is being considered, such that for the set of factors K, the {K-a}-standardised rate is returned

Examples

```
## 2 populations, R=ab
eg2.1 <- data.frame(
  pop = c("black", "white"),
  avg_earnings = c(10930, 16591),
  earner_prop = c(.717892, .825974)
)

dgnpop(eg2.1, pop = "pop", factors = c("avg_earnings", "earner_prop")) |>
  dg_table()

## 2 populations, R=abc
eg2.2 <- data.frame(
  pop = c("austria", "chile"),
```

```
  birthsw1549 = c(51.78746, 84.90502),
  propw1549 = c(.45919, .75756),
  propw = c(.52638, .51065)
)

dgnpop(eg2.2, pop = "pop", factors = c("birthsw1549", "propw1549", "propw")) |>
  dg_table()

## 2 populations, R=abcd
eg2.3 <- data.frame(
  pop = c(1971, 1979),
  birth_preg = c(25.3, 32.7),
  preg_actw = c(.214, .290),
  actw_prop = c(.279, .473),
  w_prop = c(.949, .986)
)

dgnpop(eg2.3,
  pop = "pop",
  factors = c("birth_preg", "preg_actw", "actw_prop", "w_prop")
) |>
  dg_table()

## 2 populations, R=abcde
eg2.4 <- data.frame(
  pop = c(1970, 1980),
  prop_m = c(.58, .72),
  noncontr = c(.76, .97),
  abort = c(.84, .97),
  lact = c(.66, .56),
  fecund = c(16.573, 16.158)
)

dgnpop(eg2.4,
  pop = "pop",
  factors = c("prop_m", "noncontr", "abort", "lact", "fecund")
) |>
  dg_table()

## 2 populations, vector factors, R=sum(abc)
eg4.3 <- data.frame(
  agegroup = rep(1:7, 2),
  pop = rep(c(1970, 1960), e = 7),
  bm = c(
    488, 452, 338, 156, 63, 22, 3,
    393, 407, 369, 274, 184, 90, 16
  ),
  mw = c(
    .082, .527, .866, .941, .942, .923, .876,
    .122, .622, .903, .930, .916, .873, .800
  ),
  wp = c(
    .058, .038, .032, .030, .026, .023, .019,
```

```
    .043, .041, .036, .032, .026, .020, .018
  )
)

dgnpop(eg4.3,
  pop = "pop", factors = c("bm", "mw", "wp"),
  ratefunction = "sum(bm*mw*wp)"
) |>
  dg_table()

## 2 populations, R=f(ab)
eg3.1 <- data.frame(
  pop = c(1940, 1960),
  crude_birth = c(19.4, 23.7),
  crude_death = c(10.8, 9.5)
)
dgnpop(eg3.1,
  pop = "pop",
  factors = c("crude_birth", "crude_death"),
  ratefunction = "crude_birth-crude_death"
) |>
  dg_table()

## 2 populations, vector factors, R=f(abcd)
eg4.4 <- data.frame(
  pop = rep(c(1963, 1983), e = 6),
  agegroup = c("15-19", "20-24", "25-29", "30-34", "35-39", "40-44"),
  A = c(
    .200, .163, .146, .154, .168, .169,
    .169, .195, .190, .174, .150, .122
  ),
  B = c(
    .866, .325, .119, .099, .099, .121,
    .931, .563, .311, .216, .199, .191
  ),
  C = c(
    .007, .021, .023, .015, .008, .002,
    .018, .026, .023, .016, .008, .002
  ),
  D = c(
    .454, .326, .195, .107, .051, .015,
    .380, .201, .149, .079, .025, .006
  )
)

dgnpop(eg4.4,
  pop = "pop", factors = c("A", "B", "C", "D"),
  id_vars = "agegroup",
  ratefunction = "sum(A*B*C) / (sum(A*B*C) + sum(A*(1-B)*D))"
) |>
  dg_table()

### alternatively:
```

```
myratef <- function(a, b, c, d) {
  return(sum(a * b * c) / (sum(a * b * c) + sum(a * (1 - b) * d)))
}
dgnpop(eg4.4,
  pop = "pop", factors = c("A", "B", "C", "D"),
  id_vars = "agegroup",
  ratefunction = "myratef(A,B,C,D)"
) |>
  dg_table()

## using crossclassified for relative size:
eg5.1 <- data.frame(
  age_group = rep(c(
    "15-19", "20-24", "25-29", "30-34", "35-39",
    "40-44", "45-49", "50-54", "55-59", "60-64",
    "65-69", "70-74", "75+"
  ), 2),
  pop = rep(c(1970, 1985), e = 13),
  size = c(
    12.9, 10.9, 9.5, 8.0, 7.8, 8.4, 8.6, 7.8, 7.0, 5.9, 4.7, 3.6, 4.9,
    10.1, 11.2, 11.6, 10.9, 9.4, 7.7, 6.3, 6.0, 6.3, 5.9, 5.1, 4.0, 5.5
  ),
  rate = c(
    1.9, 25.8, 45.7, 49.6, 51.2, 51.6, 51.8, 54.9, 58.7, 60.4, 62.8, 66.6, 66.8,
    2.2, 24.3, 45.8, 52.5, 56.1, 55.6, 56.0, 57.4, 57.2, 61.2, 63.9, 68.6, 72.2
  )
)
dgnpop(eg5.1,
  pop = "pop", factors = c("rate"),
  id_vars = "age_group",
  crossclassified = "size"
) |>
  dg_table()

## 2 cross-classified variables, 2 populations, R=sum(w*r)
eg5.3 <- data.frame(
  race = rep(rep(1:2, e = 11), 2),
  age = rep(rep(1:11, 2), 2),
  pop = rep(c(1985, 1970), e = 22),
  size = c(
    3041, 11577, 27450, 32711, 35480, 27411, 19555, 19795, 15254, 8022, 2472,
    707, 2692, 6473, 6841, 6547, 4352, 3034, 2540, 1749, 804, 236,
    2968, 11484, 34614, 30992, 21983, 20314, 20928, 16897, 11339, 5720, 1315,
    535, 2162, 6120, 4781, 3096, 2718, 2363, 1767, 1149, 448, 117
  ),
  rate = c(
    9.163, 0.462, 0.248, 0.929, 1.084, 1.810, 4.715, 12.187, 27.728, 64.068, 157.570,
    17.208, 0.738, 0.328, 1.103, 2.045, 3.724, 8.052, 17.812, 34.128, 68.276, 125.161,
    18.469, 0.751, 0.391, 1.146, 1.287, 2.672, 6.636, 15.691, 34.723, 79.763, 176.837,
    36.993, 1.352, 0.541, 2.040, 3.523, 6.746, 12.967, 24.471, 45.091, 74.902, 123.205
  )
)
```

```
dgnpop(eg5.3,
  pop = "pop", factors = c("rate"),
  id_vars = c("race", "age"),
  crossclassified = "size"
) |>
  dg_table()


## 5 populations, R = f(abcd)
eg6.5 <- data.frame(
  pop = rep(c(1963, 1968, 1973, 1978, 1983), e = 6),
  agegroup = c("15-19", "20-24", "25-29", "30-34", "35-39", "40-44"),
  A = c(
    .200, .163, .146, .154, .168, .169,
    .215, .191, .156, .137, .144, .157,
    .218, .203, .175, .144, .127, .133,
    .205, .200, .181, .162, .134, .118,
    .169, .195, .190, .174, .150, .122
  ),
  B = c(
    .866, .325, .119, .099, .099, .121,
    .891, .373, .124, .100, .107, .127,
    .870, .396, .158, .125, .113, .129,
    .900, .484, .243, .176, .155, .168,
    .931, .563, .311, .216, .199, .191
  ),
  C = c(
    .007, .021, .023, .015, .008, .002,
    .010, .023, .023, .015, .008, .002,
    .011, .016, .017, .011, .006, .002,
    .014, .019, .015, .010, .005, .001,
    .018, .026, .023, .016, .008, .002
  ),
  D = c(
    .454, .326, .195, .107, .051, .015,
    .433, .249, .159, .079, .037, .011,
    .314, .181, .133, .063, .023, .006,
    .313, .191, .143, .069, .021, .004,
    .380, .201, .149, .079, .025, .006
  )
)

dgnpop(eg6.5,
  pop = "pop", factors = c("A", "B", "C", "D"),
  id_vars = "agegroup",
  ratefunction = "1000*sum(A*B*C) / (sum(A*B*C) + sum(A*(1-B)*D))"
) |>
  dg_table()

dgnpop(eg6.5,
  pop = "pop", factors = c("A", "B", "C", "D"),
  id_vars = "agegroup",
  ratefunction = "1000*sum(A*B*C) / (sum(A*B*C) + sum(A*(1-B)*D))"
```

```
) |>
  dg_plot()
```

`dg_plot` *Creates a plot of Das Gupta standardised rates across the set of populations*

Description

Creates a plot of Das Gupta standardised rates across the set of populations

Usage

```
dg_plot(dgo, legend.position = "topright")
```

Arguments

`dgo`	output from dgnpop()
`legend.position`	legend position, passed to `graphics::legend` - choose from "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center", or provide xy.coords

Value

A plot of each of the set of K-a standardised rates across populations

---

`dg_table` *Creates a small table of Das Gupta standardised rates. If no populations are specified, rates will be shown for all available populations. If only two populations (or if two particular populations are specified), then rate-differences and 'decomposition effects' are calculated and presented.*

Description

Creates a small table of Das Gupta standardised rates. If no populations are specified, rates will be shown for all available populations. If only two populations (or if two particular populations are specified), then rate-differences and 'decomposition effects' are calculated and presented.

Usage

```
dg_table(dgo, pop1 = NULL, pop2 = NULL)
```

Arguments

`dgo`	output from `dgnpop()`
`pop1`	optional name of first population for decomposition (character/numeric)
`pop2`	optional name of second population for decomposition (character/numeric)

Value

data.frame object with rows for each of the K-a standardised rates and the crude rates, and columns for each of the N populations. When only two populations are included, or if two populations are explicitly specified, standardised rate differences are provided, and are also expressed as a percentage of the crude rate differences (typically referred to as 'decomposition effects').

`reconv` *Scottish Reconvictions data 2004-2016*

Description

Scottish Reconvictions data 2004-2016

Usage

```
data(reconv)
```

Format

An object of class `data.frame` with 130 rows and 8 columns.

References

Scottish Government Reconviction data: (2016/17)

Examples

```
data(reconv)
```

split_popstr — *Das Gupta equation 5.36 for a single population: Decomposes cross-classified population structures into a set of symmetric proportions indicating contribution of individual structural variables.*

Description

Das Gupta equation 5.36 for a single population: Decomposes cross-classified population structures into a set of symmetric proportions indicating contribution of individual structural variables.

Usage

```
split_popstr(x, id_vars, nvar)
```

Arguments

`x`	dataframe consisting of one population, including variables indicating cross-classified structure, and a variable indicating size of each cell
`id_vars`	character vector of variables indicating cross-classified structure.
`nvar`	variable name (character string) containing cell sizes

Value

inputted data.frame is returned with the addition of variables for each of the the cross-classified variables representing the contribution to the population size.

uspop — *US population data 1940-1990*

Description

US population data 1940-1990

Usage

```
data(uspop)
```

Format

An object of class `data.frame` with 459 rows and 4 columns.

Examples

```
data(uspop)
```

Index