DasGuptR: An R package for the standardisation and decomposition of rates

Josiah King[1]

Ben Matthews[2]

Abstract

BACKGROUND
Demographic differences between populations are often obscured by ”crude rates” that fail to account for underlying shifts in population composition. This paper introduces a solution for researchers to understand whether and how much a change in a rate (e.g., mortality, fertility, conviction rates) may simply be driven by changes in demographic structure of the groups being compared.

OBJECTIVE
In this paper we introduce DasGuptR, an R package designed to provide a flexible, open-source framework for standardizing and decomposing rates. It specifically implements the method developed by Das Gupta (1993) for aggregate level data, providing researchers with the tools to easily implement these techniques.

METHODS
We provide an introduction to the methods of standardisation and decomposition, and discuss the approaches to these tasks when unit-level data are unavailable. We provide examples of the DasGuptR package to apply these techniques in contexts where: there are any number of component factors (the contributing parts that make up the calculation of a rate); data is cross-classified (sub-populations are defined by the combination of multiple factors, e.g., age and education); standardisation is desired across *multiple* populations (such as across a time-series).

CONTRIBUTION
The DasGuptR R package offers flexible tools for decomposing differences in rates when researchers have access only to aggregate level data, providing a simple the workflow

[1] University of Edinburgh, Email: josiah.king@ed.ac.uk
[2] University of Stirling

for researchers wishing to decompose crude rates into clear, interpretable components of ”rate effects” and ”composition effects” across varied scenarios.

Contents

1. Introduction

Population rates, such as birth rates, imprisonment rates or mortality rates, are frequently in the news, and comparisons of rates — either across time or between populations — are often used to support introduction of policies or as evidence of policy successes (or failures), or to highlight inequalities between groups. The comparison of two or more rates, however, do not always reflect actual differences in the prevalence or incidence of the event in question. Instead, differences in rates between populations of interest may be directly or indirectly influenced by underlying differences in the populations' characteristics. Two related methods — standardisation and decomposition — are frequently employed in order to separate out differences in rates from differences in the compositional factors that make up the populations (or that make up the calculation of the rate itself). Various strategies have been proposed to standardize and decompose population rates, including flexible methods outlined by Das Gupta in a series of articles (1978; 1989; 1991; 1993; 1994) which allow standardization and decomposition across multiple factors and populations. Surprisingly, for researchers wishing to implement Das Gupta's approach there is limited choice of software: here we present the DasGuptR package for R that implements the full functionality described in Das Gupta's 1993 manual for standardising and decomposing rates as a function of a set of K vectors across N populations.

2. Motivation

Standardisation and decomposition are broad terms to capture methods that attempt to answer two questions: What would the rate be if these populations were the same with respect to X? (standardisation); How much of the difference in rates between two populations is due to the populations differing in X? (decomposition). For decomposition based on population proportions, in simple cases the calculation is straightforward enough to be done by hand. However, when more compositional factors are introduced, researchers can either collapse these to allow a simpler decomposition (preventing any conclusions about specific characteristic features of population) (Kitagawa 1955), or treat it as the combinatorics problem developed by Das Gupta (1978) that grows exponentially as the number of factors increases. This latter approach can be neatly extended to multiple populations (as in a time-series), and to situations in which the rate is not a simple product.

In addition to these 'aggregate' approaches to decomposition, a similar approach can be adopted with regression based methods of decomposition using 'unit' level data which regress rates (or binary events) onto a set of explanatory factors, and then estimating a counterfactual outcome had each unit been in the alternative population. One

such regression based decomposition method is widely referred to as Oaxaca-Blinder decomposition (Oaxaca 1973; Blinder 1973). Various packages exist for regression based methods such as the oaxaca (Hlavac 2022) and ddcompose (Gallusser and Meier 2024) R packages and the oaxaca Stata command (Jann 2008). In addition, the process can be conducted with most regression modelling software and some additional work estimating the appropriate counterfactual predictions. For example, a researcher could follow the logic of the marginaleffects package workflow (Arel-Bundock, Greifer, and Heiss 2024) and aggregate unit-level observed and counterfactual outcomes up to the level of the two populations in different ways in order to provide the relevant average predictions (for standardised rates) and comparisons (for decomposition of the crude difference in the two rates). In some cases the regression- and aggregate- approaches to decomposition are equivalent (as is demonstrated in detail in Oaxaca and Sierminska (2025) [3]). The difference here is that the aggregate approach needs only population-level data proportions, and not the individual unit level data.

Various methods for aggregate decomposition have been proposed — see Horiuchi, Wilmoth, and Pletcher (2008) for an overview of five general decomposition methods, as well as recommendations for when different approaches may be preferable — and implemented in the DemoDecomp package in R (Riffe 2024). Surprisingly, however, for researchers wishing to implement Das Gupta's approach there is limited choice of software: Das Gupta provides code in Fortran for examples of his method (Das Gupta 1993); Wang et al. (2000) created an early standalone Windows executable; and, somewhat more recently, Stata commands have been produced (Li 2017). However, the `rdecompose` command requires access to the proprietary Stata software, potentially limiting access to applied researchers. The DasGuptR package for R implements the full functionality described in Das Gupta's 1993 manual for standardising and decomposing rates as a function of a set of K vectors across N populations. Associated package vignettes provide examples of how this can be adapted in order to bootstrap standard errors for decomposition effects (as in Wang et al. (2000)), and how to use standardised cell-specific rates in order to decompose rate differences in to effects for each category of a compositional variable (e.g., allowing us to ask if the difference is driven by change in one specific age-group more than others), as described by Chevan and Sutherland (2009).

[3] The idea is not dissimilar to how regression models can be estimated from the sample moments (i.e. how path tracing rules allow us to estimate regression models from covariance matrices alone).

3. The Das Gupta approach to standardization and decomposition

For a simple motivating example, suppose that we have two populations — a Scottish reconviction cohort[4] from 2006, and a Scottish reconviction cohort from 2016, and we are concerned with the prevalance of reconvictions. Data are available from Scottish Government as part of their Reconvictions Bulletin (Government 2019). In 2006, 32.4% of the reconviction cohort had a reconviction (17272 of 53305), and in 2016 that number was 27.2% (11035 out of 40606).

However, these two populations may differ in their age distributions. To keep things simple, splitting the populations into those above/below 30 years old, we can see that the 2006 cohort has a higher proportion of under 30s (for whom reconvictions are more common - Table 1).

Table 1: Two populations and their age-specific reconviction rates.

Age	Offenders	Age Proportion	Reconvicted	Prevalence
2006				
under 30	31937	0.599	11897	0.373
over 30	21368	0.401	5375	0.252
Total (Crude)	53305	1.000	17272	0.324
2016				
under 30	18159	0.447	5466	0.301
over 30	22447	0.553	5569	0.248
Total (Crude)	40606	1.000	11035	0.272

Standardization tells us what the rate of reconvictions would be in 2016 if the population had the same age distribution as 2006, or vice versa, or even what the rate of reconvictions would be in both years if the populations were the same (at the average) in age distributions. Decomposition tells us how much of the crude difference in rates of $0.324 - 0.272 = 0.052$ is due to differences in the age distributions between the two populations. The decomposition is implicitly linked to standardisation — if *all* of the difference is due to age-distribution differences (illustrated in Table 2), then any "age-standardised rates" would be identical in both populations.

[4] People who were released from a custodial sentence or given a non-custodial sentence in each year.

Table 2: If the rates of reconvictions for each age-group were the same for both populations, differences in age-distributions can still lead to differences in crude rates.

Age	Offenders	Age Proportion	Reconvicted	Prevalence
2006				
under 30	31937	0.599	11897	0.373
over 30	21368	0.401	5375	0.252
Total (Crude)	53305	1.000	17272	0.324
2016				
under 30	18159	0.447	6764	0.373
over 30	22447	0.553	5646	0.252
Total (Crude)	40606	1.000	12410	0.306

The decomposition of the crude rates here is relatively straightforward in that it can be split into two parts: the proportions in each age-group, and the age-group-specific rates. In our example, this is just that the 2016 rate of $0.272 = (0.301 \cdot 0.447) + (0.248 \cdot 0.553)$. We can write this more generally for an arbitrary number of sub-groups as:

$$r = \sum_i a_i b_i$$

where: (1)

$a =$ group specific rate

$b =$ proportion of population in group

In this formulation, it is straightforward to compute "standardised rates" by calculating the rates $\sum_i a_i b_i$ under different counterfactuals such as whether the 2016 population had the same age distribution as 2006, giving $(0.301 \cdot 0.599) + (0.248 \cdot 0.401) = 0.280$, or where the age distribution was held at the average of the two populations (Equation 2).

$$\text{age-standardised-}r^{2006} = \left(0.373 \cdot \frac{0.599 + 0.447}{2}\right) + \left(0.252 \cdot \frac{0.401 + 0.553}{2}\right) = 0.315$$

$$\text{age-standardised-}r^{2016} = \left(0.301 \cdot \frac{0.599 + 0.447}{2}\right) + \left(0.248 \cdot \frac{0.401 + 0.553}{2}\right) = 0.276$$

$$\text{difference} = 0.039 \tag{2}$$

Standardisation holding the compositional factors at their average allows us to consider the decomposition question: If the two populations had the same age distribution, how much would they differ in their rates? The difference between standardised rates (0.039), taken as a proportion of the difference between crude rates (0.052), allows us to attribute 75% of the difference in crude rates as being due to differences in the group-specific rates in the populations (and other stray causes), with 25% being explained by differences in the age distribution. We can similarly see this by substituting in the averages of the group-specific rates instead of the averages of the group proportions, for which we get 0.302 and 0.289, and a difference of 0.013 — 25% of the crude rate difference (Equation 3).

$$\text{rate-standardised-}r^{2006} = \left(\frac{0.373 + 0.301}{2} \cdot 0.599\right) + \left(\frac{0.252 + 0.248}{2} \cdot 0.401\right) = 0.302$$

$$\text{rate-standardised-}r^{2016} = \left(\frac{0.373 + 0.301}{2} \cdot 0.447\right) + \left(\frac{0.252 + 0.248}{2} \cdot 0.553\right) = 0.289$$

$$\text{difference} = 0.013 \tag{3}$$

We can express this method of standardisation and decomposition using letters to differentiate the compositional factors that make up the rate (the age-group proportions and the age-group specific rates) across each sub-population i, with superscripts to distinguish these factors from each of two populations p and p':

$$b\text{-standardised-}r^p = \sum_i \frac{b_i^p + b_i^{p'}}{2} a_i^p \qquad a\text{-standardised-}r^p = \sum_i \frac{a_i^p + a_i^{p'}}{2} b_i^p \tag{4}$$

$$b\text{-standardised-}r^{p'} = \sum_i \frac{b_i^p + b_i^{p'}}{2} a_i^{p'} \qquad a\text{-standardised-}r^{p'} = \sum_i \frac{a_i^p + a_i^{p'}}{2} b_i^{p'} \tag{5}$$

$$\Delta\text{crude-}r = \sum_i \frac{b_i^p + b_i^{p'}}{2} (a_i^p - a_i^{p'}) + \sum_i \frac{a_i^p + a_i^{p'}}{2} (b_i^p - b_i^{p'}) \tag{6}$$

Note that the neatness of decomposing differences in crude rates into the sum of differences in standardised rates cannot be achieved when using one population as the standard because of the interaction between the factors. In Equation 7, the first two terms represent the differences in $b-$ and $a-$standardised rates using population p' as the standard, but to equate this to the difference in crude rates requires the addition of the interaction term.

$$\Delta\text{crude-}r = \sum_i b_i^{p'} (a_i^p - a_i^{p'}) \quad + \quad \sum_i a_i^{p'} (b_i^p - b_i^{p'}) \quad + \quad \sum_i (a_i^p - a_i^{p'})(b_i^p - b_i^{p'}) \tag{7}$$

Das Gupta's solution is to reframe the standardization and decomposition as essentially a combinatorics problem — to compute the rate were only one factor to differ between the two populations, the calculation of the rate is averaged over all possible counterfactuals (different combinations of other factors changing vs being held equal). These are weighted relative to the number of factors varied/held equal, meaning that their importance is lessened the farther the counterfactual is from an actual population. Das Gupta's general solution for the decomposition of two rates can be written as:

$$\Delta\text{crude-}r = \sum_{\vec{\alpha} \in K} Q(\vec{\alpha}^p) - Q(\vec{\alpha}^{p'}) \tag{8}$$

Where K is the set of factors $\alpha, \beta, ..., \kappa$, which may take the form of vectors over sub-populations. $Q(\vec{\alpha}^p)$ denotes the rate in population p holding $K \setminus \{\alpha\}$ (all factors other than α), 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:

$$Q(\vec{\alpha}^p) = \sum_{j=1}^{\lceil \frac{|K|}{2} \rceil} \frac{\sum_{L \in \binom{K \setminus \{\alpha\}}{j-1}} f(\{L^p, (K \setminus L)^{p'}, \vec{\alpha}^p\}) + f(\{L^{p'}, (K \setminus L)^p, \vec{\alpha}^p\})}{|K| \binom{|K|-1}{j-1}} \tag{9}$$

Where $f(K)$ is the function that defines the calculation of the rate (this could be the simple sum of products such as the reconvictions example above, or something arbitrarily complex that returns a single real value). For example, when there are 5 factors, $K = a, b, c, d, e$, and the rate is calculated as the simple product, then this becomes:

$$\begin{aligned} Q(a) = & \frac{abcde + ab'c'd'e'}{5} + \\ & \frac{abcde' + abcd'e + abc'de + ab'cde + ab'c'd'e + ab'c'de' + ab'cd'e' + abc'd'e'}{20} + \\ & \frac{abcd'e' + abc'de' + abc'd'e + ab'c'de + ab'cd'e + ab'cde'}{30} \end{aligned} \tag{10}$$

Das Gupta's method distributes interactions equally across all factors, meaning that standardised rates and decompositions equate to average treatment effects at the level of populations. Where Y is a binary outcome, and P the binary population indicator, using Y^p to denote the random variable of outcomes under population p, and $Y^{p'}$ as the outcomes under population p', we can express crude rates as the expected value of observed outcomes for each population, and the crude rate difference can be written as the straightforward contrast:

$$\Delta\text{crude-}r = \mathbb{E}[Y^{p'}|P = p'] - \mathbb{E}[Y^p|P = p] \tag{11}$$

Das Gupta's methodology estimates, for instance, what the population level rates would be if every factor included in the decomposition was held equal (thereby isolating the differences due to rates rather than population characteristics). The term "held equal"

means *averaged* across the two populations, meaning it is invariant to differences in size between the two populations, and equates to the counterfactual statement in Equation 12. Similarly, we could consider how the rates would differ if the probability of the rate event were equal in the two populations but they differed in their compositional structure (Equation 13).

$$\mathbb{E}[\,\mathbb{E}[Y^{p'}|P=p],\,\mathbb{E}[Y^{p'}|P=p']\,] - \mathbb{E}[\,\mathbb{E}[Y^{p}|P=p],\,\mathbb{E}[Y^{p}|P=p']\,] \tag{12}$$

$$\mathbb{E}[\,\mathbb{E}[Y^{p}|P=p'],\,\mathbb{E}[Y^{p'}|P=p']\,] - \mathbb{E}[\,\mathbb{E}[Y^{p}|P=p],\,\mathbb{E}[Y^{p'}|P=p]\,] \tag{13}$$

4. The DasGuptR package

DasGuptR can be installed either from CRAN using `install.packages("DasGuptR")` or from github using `remotes::install_github("josiahpjking/DasGuptR")`. In this section we describe the main functionality of the DasGuptR package, using a similar exposition structure as Das Gupta (1993), from which examples below are taken[5]. The section is structured as follows:

1. Standardization and decomposition with two factors and two populations
2. Increasing the number of factors
3. Cross-classified population structures (individual sub-populations which are defined by the combination of multiple factors)

4. Increasing the number of populations.
5. Custom rate functions

4.1 Standardization and decomposition with two factors and two populations

The workhorse of the DasGuptR package is `dgnpop()`, which computes the crude rates and the standardised rates for K factors across N populations. `dgnpop()` requires a minimum specification of the following three arguments:

- `x`, the data to be analysed. The DasGuptR workflow requires data to be in tidy format (Wickham 2014), with variables denoting the population and each compositional factor.
- `pop`, a character string indicating which column in the dataframe identifies the

[5] The terminology used throughout reflects the time period in which Das Gupta (1993) and the sources he drew on were writing; for sake of comparison we have adopted the same terminology as Das Gupta (1993).

population/s across which rates are to be standardized and the differences between the rates are to be decomposed

- `factors`, a *vector* of character strings identifying the factors that should be standardized and decomposed across populations. Recommended usage is to create this vector using `c()` and then providing the appropriate character strings separated by commas

It's crucial to be aware that the factors referred to in the outputs of DasGuptR are those that are **not** being held constant in the standardisation. Very often when standardisation is done over a smaller number of factors — say x, y, z — researchers use "xy-standardised rates" to refer to rates were both x and y held constant across the two populations (similar to the names used in Equation 2 to Equation 3). In DasGuptR, these would be presented as the "$K - z$-standardised rates", and is more akin to how one would expect to see the output of regression coefficients. For example, we might have two populations for which a rate r is calculated as $r = ab$

Simple example of a rate as the product of two factors

```
eg.dg <- data.frame(
  pop = c("pop1","pop2"),
  alpha = c(.6,.3),
  beta = c(.5,.45),
  crude = c(.6*.5, .3*.45)
)
```

In this case, `dgnpop()` can be used to perform the relatively straightforward calculation presented in Equation 5.

```
library(DasGuptR)
dgnpop(eg.dg, pop = "pop", factors = c("alpha","beta"))
```

```
    rate  pop std.set factor
1 0.3000 pop1    <NA>  crude
2 0.1350 pop2    <NA>  crude
3 0.2850 pop1    pop2  alpha
4 0.1425 pop2    pop1  alpha
5 0.2250 pop1    pop2   beta
6 0.2025 pop2    pop1   beta
```

The default output from `dgnpop` is a dataframe which contains the output of the rate function (by default, the multiplication of the factors provided) in the `rate` column,

the populations standardized over in the `pop` column, the population used for each standardization in the `std.set` column and the factor omitted from the standardization in the `factor` column. The first N rows in the `dgnpop` output have the factor `crude` which report the results of the rate function on the raw data itself - in other words, the unstandardized values.

The results from `dgnpop` can be quickly turned into a wide table of 'decomposition effects' in the style of Das Gupta using `dg_table()`, which — when working with just two populations — calculates and displays differences in standardised rate and also expresses these as a percentage of the crude rate difference:

```
dgnpop(eg.dg, pop = "pop", factors = c("alpha","beta")) |>
  dg_table()
```

```
       pop1   pop2    diff decomp
alpha 0.285 0.1425 -0.1425  86.36
beta  0.225 0.2025 -0.0225  13.64
crude 0.300 0.1350 -0.1650 100.00
```

4.2 Increasing the number of factors

The addition of more factors into the composition of a population rate is handled in `dgnpop()` by simply adding more terms to the `factors` argument. By default `dgnpop` will take the rate to be the product of all factors specified, although users can provide custom rate functions via the `ratefunction` argument (see Section 4.5 below).

Example 2.3: Percentage having non-marital live births as the product of four factors for White women aged 15 to 19 in the United States, in the years 1971 and 1979.

```
eg2.3 <- data.frame(
  pop = c(1971, 1979),
  # non-marital live births x 100 / non-marital pregnancies
  birth_preg = c(25.3, 32.7),
  # non-marital pregnancies / sexually active single women
  preg_actw = c(.214, .290),
  # sexually active single women / total single women
  actw_prop = c(.279, .473),
  # total single women / total women
  w_prop = c(.949, .986)
)
```

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

```
               1971     1979      diff decomp
birth_preg 2.355434 3.044375 0.6889411  23.05
preg_actw  2.287936 3.100474 0.8125381  27.18
actw_prop  1.988818 3.371723 1.3829055  46.26
w_prop     2.686817 2.791572 0.1047547   3.50
crude      1.433523 4.422663 2.9891394 100.00
```

4.3 Population structures and cross-classified data

It is often the case that we have data on each compositional factor for a set of sub-populations — e.g. conviction rates for people of different age groups — and the crude rates for each population (the overall conviction rates) are the aggregated cell-specific rates. Very often, we will be interested specifically in effects of the sub-population size — i.e., how populations differ with respect to their compositional *structure*. To do this, we require data on the sizes (or relative sizes) of each sub-population.

The simplest case here is one in which there is data on a single set of sub-populations (e.g., age-groups), and we have the group-specific rates and group sizes. The crude rates for the population would be simply the sum of all the group-specific rates weighted by the relative size of the group.

Example 5.1: Household Headship Rates per 100: United States, years 1970 and 1985

```
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),
  # number in age-group / total population * 100
  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
  ),
  # age-group specific rate
  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
  )
)
```

When working with compositional factors that are vectors of sub-population values ('vector factors' in Das Gupta's parlance), and it is the population-level rates that are of interest, the user is either required to either:

1. provide an appropriate rate function that aggregates sub-population rates up to a single value for each population.

2. make use of the `crossclassified` and `id_vars` arguments (which will allow for more complex breakdowns of population structures)

In Example 5.1, where the aggregate rate is defined as a weighted sum $\sum_i w_i r_i$, one easy option is to create a new column of these weights include this in the set of compositional factors passed to `factors` as previously. As the weights are the proportions of the population in each age-group, and in this example we have percentages, we can simply divide by 100. We must then provide a rate function that takes the sum of the products of sub-population weights and sub-population-specific rates.

```
eg5.1$age_str <- eg5.1$size / 100

dgnpop(eg5.1,
  pop = "pop",
  factors = c("age_str", "rate"),
  ratefunction = "sum(age_str*rate)"
)
```

```
     rate  pop std.set  factor
1 44.7268 1970    <NA>   crude
2 47.6940 1985    <NA>   crude
3 45.5876 1970    1985 age_str
4 46.8145 1985    1970 age_str
5 45.3309 1970    1985    rate
6 47.0712 1985    1970    rate
```

Alternatively, we could include the conversion to proportions *inside* the rate function, including the original sub-population sizes variable (`size`) in the vector of factors. Because the inputs to the rate function here are the set of factors, each of which is a vector of sub-population values, internal calls to `sum(size)` will give us the total size for each population, and `size/sum(size)` will give us the sub-population-specific proportions:

```
dgnpop(eg5.1,
  pop = "pop",
  factors = c("size", "rate"),
  ratefunction = "sum( (size/sum(size))*rate )"
)
```

Finally, we can instead provide the variable indicating the size of each sub-population into the `crossclassified` argument of `dgnpop()`, along with also specifying the variable(s) indicating the sub-populations in `id_vars` argument. Note that when this approach is used, any provided rate function should **not** aggregate to the population level.

```
dgnpop(eg5.1,
  pop = "pop",
  factors = c("rate"),
  id_vars = "age_group",
  crossclassified = "size"
)
```

Using `crossclassified` is the most flexible approach as it can be extended to situations in which we have cross-classified data — individual sub-populations which are defined by the *combination* of multiple factors such as age and race/ethnicity, as Example 5.3 below. In cases with cross-classified population structures, defining weights as `size/sum(size)` will collapse the cross-classified factors that make up the structure of the population (e.g., age and race), thereby getting us only part of the way. This allows us to decompose rate differences into "rate" vs "age-and-race" differences, but leaves us unable to separate out the relative contributions of age and race to the aggregate difference in rates. Instead, providing the cell-specific sizes to the `crossclassified` argument will re-express the proportion of the population in a given cell as the product of symmetrical expressions corresponding to each of the cross-classified variables.

Using i to donote a specific sub-population that is uniquely identified through the set of cross-classified variables $K : \{\alpha, \beta, ..., \kappa\}$, and i_κ to denote the specific level of κ for sub-population i, we can indicate the size of sub-population i with $N_{\forall\kappa\in K,\kappa=i_\kappa}$. The expression $N_{\forall\kappa\in K,\kappa=.}$ denotes the total population size, summed across all levels of all variables. The Das Gupta methodology expresses the proportion of the population in sub-population i as a product of $|K|$ variables (Equation 14).

$$\frac{N_{\forall\kappa\in K,\kappa=i_\kappa}}{N_{\forall\kappa\in K,\kappa=.}} = \hat{\alpha}_{\forall\kappa\in K,\kappa=i_\kappa}\hat{\beta}_{\forall\kappa\in K,\kappa=i_\kappa}...\hat{\kappa}_{\forall\kappa\in K,\kappa=i_\kappa} \tag{14}$$

These are defined as the product of ratios that aggregate over different combinations of the cross-classified variables (Equation 15), which are then standardised via the method described earlier (Equation 8, Equation 9), and multiplied by the average cell-specific rates (i.e., $\frac{r_i+r'_i}{2}$) before being aggregated up to the population level to obtain $\alpha\ldots\kappa$-standardised rates.

$$\hat{\alpha}_{\forall \kappa \in K, \kappa = i_\kappa} = \prod_{j=0}^{|K|-1} \left(\prod_{L \in \binom{K \setminus \{\alpha\}}{j}} \frac{N_{\forall \kappa \in K,\ \kappa = \begin{cases} i_\kappa & \text{if } \kappa \in \{\alpha, L\} \\ . & \text{otherwise} \end{cases}}}{N_{\forall \kappa \in K,\ \kappa = \begin{cases} i_\kappa & \text{if } \kappa \in L \\ . & \text{otherwise} \end{cases}}} \right)^{\frac{1}{|K| \binom{|K|-1}{j}}} \tag{15}$$

One such example can be found in Das Gupta's Example 5.3 (provided below), in which the differences in population structures is decomposed into differences due to age-structures and differences due to race-structures.

Example 5.3: Death rates per 1000: United States, years 1970 and 1985

```
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),
  # number of people in age-race-group
  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
  ),
  # death rate in age-race-group
  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()
```

```
                1970     1985       diff  decomp
age_struct   8.385332 9.907067  1.5217347 -221.76
race_struct  9.136312 9.156087  0.0197749   -2.88
rate        10.257368 8.029643 -2.2277254  324.64
crude        9.421833 8.735618 -0.6862158  100.00
```

4.3.1 Sub-population specific rates

Although most often it will be the population-level rates that are of interest to researchers, on occasion it may be that sub-population specific standardised rates are desired. One such occasion would be for the calculation of "category effects" as detailed in Chevan and Sutherland (2009), to examine the amount to which a difference between populations in aggregate rates is attributable to differences in *specific* categories of the variables that identify the sub-populations. To do this, we simply specify the `agg` argument of `dgnpop()` as `FALSE`, indicating we do not want aggregated rates returned. A demonstration of this is available in the package vignette on category effects (`vignette("category_effe package="DasGuptR")`)

4.4 Increasing the number of populations

When standardising across more than two populations, computing the standardisation across all pairs of populations returns $N-1$ sets of standardised rates for each population, and the decomposition between populations are internally inconsistent — i.e. differences in standardised rates between populations 1 and 2, and between 2 and 3, should (but do not) sum to the difference between 1 and 3.

Das Gupta (1993) provided a secondary procedure that takes sets of pairwise standardised rates and resolves these problems. This is presented in Equation 16, $\bar{r}_{x|y}$ denoting a standardised rate in population x that is standardised across populations x and y. When given more than two populations, `dgnpop()` will automatically undertake this procedure. The resulting standardised across all N populations and can be used as previously with `dg_table()` and `dg_plot()`.

$$\bar{r}_{1|23\ldots N} = \frac{\sum_{i=2}^{N} \bar{r}_{1|i}}{N-1} + \frac{\sum_{i=2}^{N} \sum_{j\neq 1,i}^{N} \bar{r}_{i|j} - (N-2)\bar{r}_{i|1}}{N(N-1)} \tag{16}$$

Example 6.5: Illegitimacy Ratio as a function of four vector factors: United States, years 1963, 1968, 1973, 1978, and 1983

```
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"),
  # number of women in age-group / total women
  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
  ),
  # number of unmarried women in age-group /
  #    number of women in age-group
  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
  ),
  # births to unmarried women in age-group /
  #    number of unmarried women in age-group
  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
  ),
  # births to married women in age-group /
  #    married women in age-group
  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()
```

```
          1963     1968     1973     1978      1983
A     72.77031 74.65254 73.83625 71.36001  64.60057
B     53.28255 56.63216 59.52972 79.49763 104.38728
C     62.17750 69.61231 60.48031 68.54219  94.17361
D     54.83361 64.43823 81.24203 79.60288  74.12756
crude 30.94957 53.22084 62.97390 86.88830 125.17462
```

Because using `dg_table()` with multiple populations will return standardised rates for each population, it will not return decomposition effects unless only two populations are specified. Decomposition of the effects between specific populations can be calculated passing two specific populations to the `pop1` and `pop2` arguments of `dg_table`. Note that the standardisation step includes *all* populations in `pop`, not just those passed to `dg_table`. There is no guarantee that these rates will be the same as those which just use two populations for the standardisation - and in most cases they won't be.

```
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(pop1 = 1963, pop2 = 1968)
```

```
          1963     1968      diff decomp
A     72.77031 74.65254  1.882236   8.45
B     53.28255 56.63216  3.349606  15.04
C     62.17750 69.61231  7.434806  33.38
D     54.83361 64.43823  9.604628  43.13
crude 30.94957 53.22084 22.271276 100.00
```

4.5 Custom rate functions

Das Gupta's methodology generalises to rates that are not simply products of the set of factors. The `ratefunction` argument of `dgnpop()` allows the user to define a custom rate function (function f in Equation 9). This may be as simple as a subtraction `a-b` (Example 3.1 below), or something more complicated that aggregates over different combinations of factors, such as `sum(a*b)/sum(a*b*c)` (Example 4.4 below).

Example 3.1: Crude rate of natural increase, US in years 1940 and 1960

```
eg3.1 <- data.frame(
  pop = c(1940, 1960),
  # births x 1000 / total population
  crude_birth = c(19.4, 23.7),
  # deaths x 1000 / total population
  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()
```

```
             1940  1960 diff decomp
crude_birth  9.25 13.55  4.3  76.79
crude_death 10.75 12.05  1.3  23.21
crude        8.60 14.20  5.6 100.00
```

Example 4.4: Illegitimacy Ratio as a function of four vector factors: United States, 1963 and 1983

```
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"),
  # number of women in age-group / total women
  A = c(.200, .163, .146, .154, .168, .169,
        .169, .195, .190, .174, .150, .122),
  # number of unmarried women in age-group /
  #    number of women in age-group
  B = c(.866, .325, .119, .099, .099, .121,
        .931, .563, .311, .216, .199, .191),
  # births to unmarried women in age-group /
  #    number of unmarried women in age-group
  C = c(.007, .021, .023, .015, .008, .002,
        .018, .026, .023, .016, .008, .002),
  # births to married women in age-group /
  #    married women in age-group
  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()
```

```
            1963       1983        diff decomp
A     0.07770985 0.07150487 -0.00620498  -6.59
B     0.04742107 0.09608261  0.04866154  51.64
C     0.05924498 0.08630419  0.02705922  28.72
D     0.05962676 0.08433604  0.02470928  26.22
crude 0.03094957 0.12517462  0.09422505 100.00
```

The `ratefunction` argument can be given any string that when parsed and evaluated will return a single value for a rate. The only criteria is that the function needs refer to the factors included in the `x` argument to `dgnpop` by name. It is possible to

define a custom function in the user's environment, and provide a call to that function to the `ratefunction` argument of `dgnpop()` as a string. Example 4.4 above can also be achieved using the code below. There is no real limit to the complexity of the rate function the user wishes to specify, and Das Gupta provides one such example in which the rate is obtained iteratively via Newton-Raphson.

```
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)"
)
```

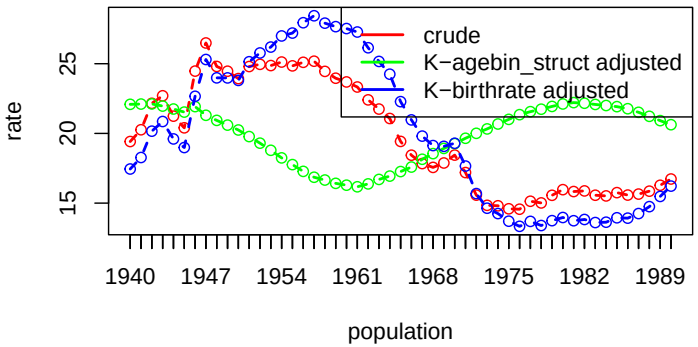
4.6 Visualising standardised rates

When working with multiple populations (typically a time series), we can get plots of the standardised rates using `dg_plot()`, and the output from `dgnpop()` can easily be used in other plotting packages.

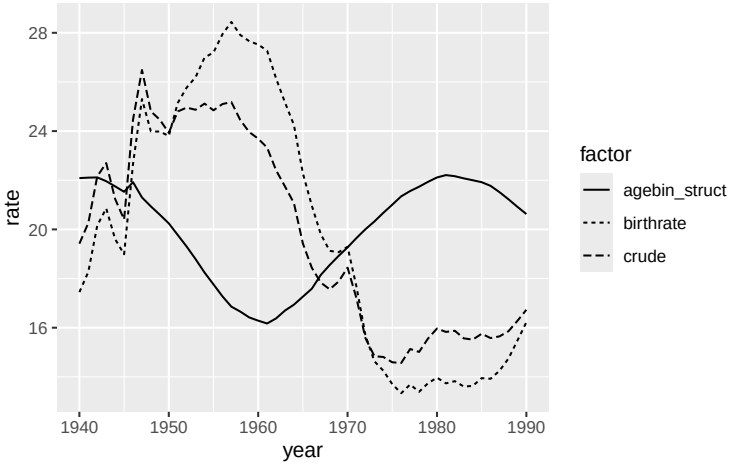
Example 6.6: Birthrates by Nine Age-Sex groups: United States, 1940 to 1990

```
data(uspop)
# birthrate = births per 1000 for age-sex group
# thous = population in thousands of age-sex group

dgo_us <- dgnpop(uspop,
  pop = "year", factors = c("birthrate"),
  id_vars = "agebin", crossclassified = "thous"
)
dg_plot(dgo_us)
```




```
dgo_us |>
  mutate(
    year = as.numeric(as.character(pop))
  ) |>
  ggplot(aes(x = year, y = rate,
             group = factor, lty = factor))+
  geom_line()
```



5. Conclusion

Standardisation and decomposition are common tasks for applied researchers who want to compare rates across populations using aggregate data, but to date there has not been an implementation of Das Gupta's (1993) flexible, general method of standardisation and decomposition available using open-source software. In this paper we have provided a conceptual overview of Das Gupta's approach to standardisation and decomposition and introduced the DasGuptR package which provides such an implementation in the R statistical programming language. The DasGuptR package provides users with the full functionality described in Das Gupta (1993), incorporating standardisation across multiple factors, populations, and with custom rate functions. Resulting standardised rates can be easily visualised, and decomposition effects can be bootstrapped for inference.

References

Arel-Bundock, V., Greifer, N., and Heiss, A. (2024). How to interpret statistical models using marginaleffects for r and python. *Journal of Statistical Software* 111: 1–32.

Blinder, A.S. (1973). Wage discrimination: reduced form and structural estimates. *Journal of Human resources* 436–455.

Chevan, A. and Sutherland, M. (2009). Revisiting das gupta: refinement and extension of standardization and decomposition. *Demography* 46(3): 429–449.

Das Gupta, P. (1978). A general method of decomposing a difference between two rates into several components. *Demography* 15(1): 99–112.

Das Gupta, P. (1989). Methods of decomposing the difference between two rates with applications to race-sex inequality in earnings. *Mathematical Population Studies* 2(1): 15–36.

Das Gupta, P. (1991). Decomposition of the difference between two rates and its consistency when more than two populations are involved. *Mathematical Population Studies* 3(2): 105–125.

Das Gupta, P. (1993). *Standardization and Decomposition of Rates: A User's Manual*.

Das Gupta, P. (1994). Standardization and decomposition of rates from cross-classified data. *Genus* 171–196.

Duxbury, S.W. (2025). Why are state prison populations shrinking? a research note. *Criminology* 63(1): 268–279.

Gallusser, D. and Meier, S. (2024). ddecompose: Detailed distributional decomposition. R package version 1.0.0, https://CRAN.R-project.org/package=ddecompose.

Government, S. (2019). Reconviction rates in scotland: 2016-2017 offender cohort. https://www.gov.scot/publications/reconviction-rates-scotland-2016-17-offender-cohort/.

Hlavac, M. (2022). oaxaca: Blinder-oaxaca decomposition in r. R package version 0.15, https://CRAN.R-project.org/package=oaxaca.

Horiuchi, S., Wilmoth, J.R., and Pletcher, S.D. (2008). A decomposition method based on a model of continuous change. *Demography* 45(4): 785–801.

Jann, B. (2008). The blinder–oaxaca decomposition for linear regression models. *The stata journal* 8(4): 453–479.

Kitagawa, E.M. (1955). Components of a difference between two rates. *Journal of the american statistical association* 50(272): 1168–1194.

Li, J. (2017). Rate decomposition for aggregate data using das gupta's method. *The Stata Journal* 17(2): 490–502.

Oaxaca, R. (1973). Male-female wage differentials in urban labor markets. *International economic review* 693–709.

Oaxaca, R.L. and Sierminska, E. (2025). Oaxaca-blinder meets kitagawa: What is the link? *PLoS One* 20(5): e0321874.

Riffe, T. (2024). Demodecomp: Decompose demographic functions. R package version 1.14.1, https://CRAN.R-project.org/package=DemoDecomp.

Wang, J., Rahman, A., Siegal, H.A., and Fisher, J.H. (2000). Standardization and decomposition of rates: useful analytic techniques for behavior and health studies. *Behavior Research Methods, Instruments, & Computers* 32(2): 357–366.

Wickham, H. (2014). Tidy data. *Journal of statistical software* 59: 1–23.