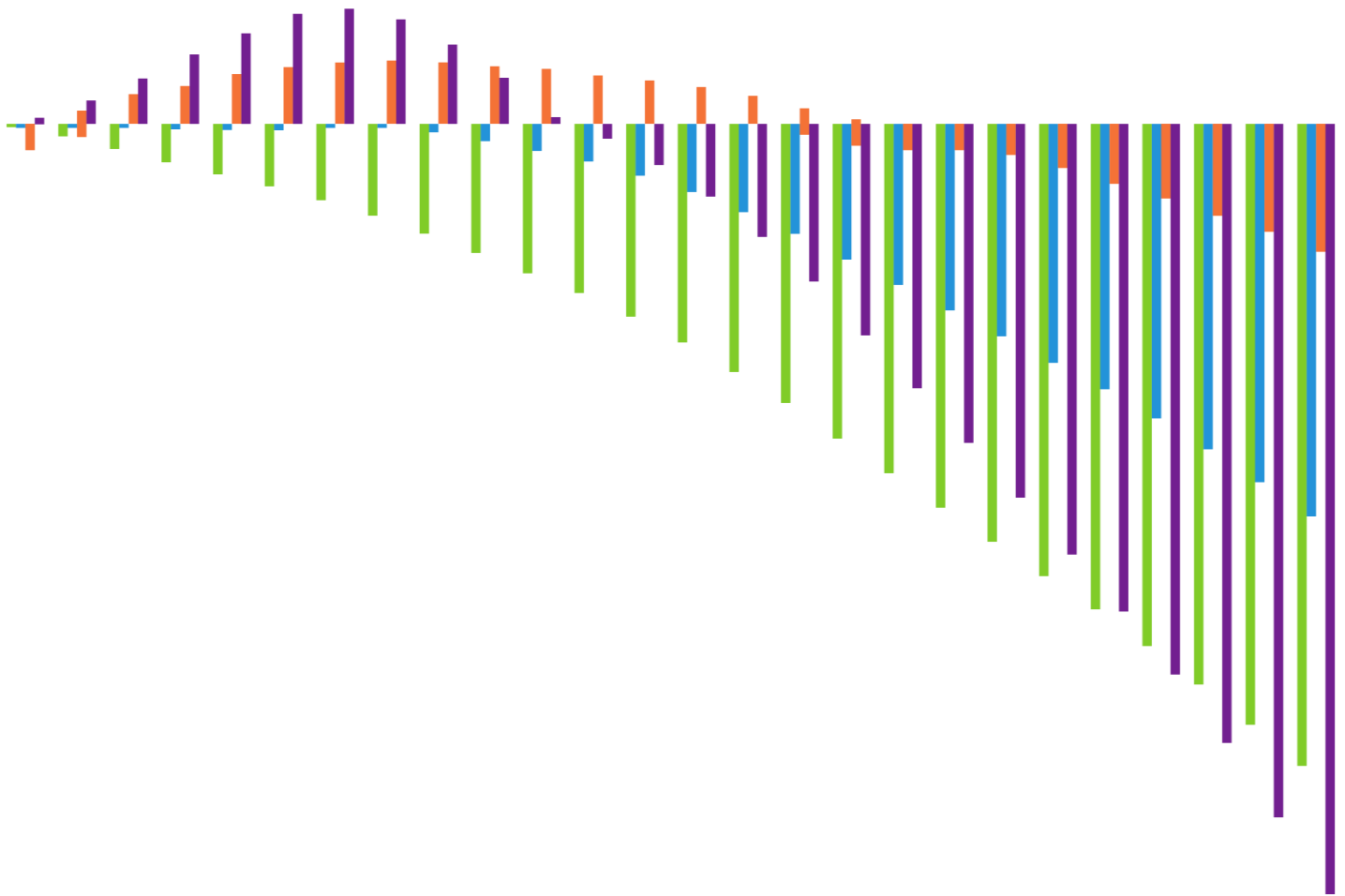


Calculating summary measures of health inequality using Stata: Step-by-step guidance



World Health
Organization

© World Health Organization 2024

Some rights reserved. This work is available under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 IGO licence (CC BY-NC-SA 3.0 IGO; <https://creativecommons.org/licenses/by-nc-sa/3.0/igo>).

Under the terms of this licence, you may copy, redistribute and adapt the work for non-commercial purposes, provided the work is appropriately cited, as indicated below. In any use of this work, there should be no suggestion that WHO endorses any specific organization, products or services. The use of the WHO logo is not permitted. If you adapt the work, then you must license your work under the same or equivalent Creative Commons licence. If you create a translation of this work, you should add the following disclaimer along with the suggested citation: "This translation was not created by the World Health Organization (WHO). WHO is not responsible for the content or accuracy of this translation. The original English edition shall be the binding and authentic edition".

Any mediation relating to disputes arising under the licence shall be conducted in accordance with the mediation rules of the World Intellectual Property Organization (<http://www.wipo.int/amc/en/mediation/rules/>).

Suggested citation. Calculating summary measures of health inequality using Stata: Step-by-step guidance. Geneva: World Health Organization; 2024. Licence: [CC BY-NC-SA 3.0 IGO](https://creativecommons.org/licenses/by-nc-sa/3.0/igo).

Cataloguing-in-Publication (CIP) data. CIP data are available at <http://apps.who.int/iris>.

Sales, rights and licensing. To purchase WHO publications, see <http://apps.who.int/bookorders>. To submit requests for commercial use and queries on rights and licensing, see <https://www.who.int/copyright>.

Third-party materials. If you wish to reuse material from this work that is attributed to a third party, such as tables, figures or images, it is your responsibility to determine whether permission is needed for that reuse and to obtain permission from the copyright holder. The risk of claims resulting from infringement of any third-party-owned component in the work rests solely with the user.

General disclaimers. The designations employed and the presentation of the material in this publication do not imply the expression of any opinion whatsoever on the part of WHO concerning the legal status of any country, territory, city or area or of its authorities, or concerning the delimitation of its frontiers or boundaries. Dotted and dashed lines on maps represent approximate border lines for which there may not yet be full agreement.

The mention of specific companies or of certain manufacturers' products does not imply that they are endorsed or recommended by WHO in preference to others of a similar nature that are not mentioned. Errors and omissions excepted, the names of proprietary products are distinguished by initial capital letters.

All reasonable precautions have been taken by WHO to verify the information contained in this publication. However, the published material is being distributed without warranty of any kind, either expressed or implied. The responsibility for the interpretation and use of the material lies with the reader. In no event shall WHO be liable for damages arising from its use.

Contents

Introduction	4
Data management considerations	6
Difference.....	7
Ratio.....	12
Absolute Concentration Index	17
Relative Concentration Index	25
Slope Index of Inequality.....	33
Relative Index of Inequality.....	42
Between-Group Variance	51
Between-Group Standard Deviation	55
Coefficient of Variation	60
Mean Difference from Mean	66
Mean Difference from the Best-Performing Subgroup.....	72
Mean Difference from a Reference Subgroup	78
Index of Disparity.....	84
Theil Index.....	90
Mean Log Deviation	94
Population Attributable Fraction	98
Population Attributable Risk	103
Further references	108

Introduction

Measuring and monitoring inequalities in health is important for informing policies and programmes that aim to tackle health inequities. Broadly defined, inequalities in health are measurable differences in health across population subgroups defined by dimensions of inequality (demographic, socioeconomic or geographic characteristics). Summary measures of inequality summarise the amount of health inequality across population subgroups in a single number – which facilitates the comparison of inequalities over time and across different settings and indicators.

This document provides step-by-step guidance on how to calculate 21 summary measures of inequality and their 95% confidence intervals using Stata. It is accompanied by Stata do-files containing codes for the calculations and sample datasets. It provides an overview of how each summary is calculated and interpreted, details the data requirements for its calculation, and then goes through the Stata do-file code step-by-step.

Summary measures of health inequality can use either disaggregated data or individual-level data as inputs.

- **Disaggregated data** are health indicator data for different subgroups of a population that are defined according to dimensions of inequality, such as demographic, socioeconomic, or geographic characteristics (such as wealth quintiles or subnational regions). Each record in the dataset contains data for a population subgroup. All of the summary measure codes included in this guidance can be calculated using disaggregated data.
- **Individual-level data** (also referred to as micro data) are data where each record in the dataset contains data pertaining to an individual. The do-files for the summary measures absolute concentration index (ACI), relative concentration index (RCI), slope index of inequality (SII), and relative index of inequality (RII) include codes for their calculation using individual-level and survey data.

Considerations when selecting and calculating summary measures of health inequality

Simple summary measures (difference and ratio) compare the situation in two population subgroups. They can be calculated for all dimensions of inequality (with two subgroups or more). Complex measures are calculated for inequality dimensions with more than two population subgroups and consider the situation in all subgroups. They can only be calculated for dimensions with more than two subgroups.

Selecting appropriate measures for analysing and reporting inequality involves the consideration of several methodological issues. There are considerations relating to the characteristics of the underlying data, which determines the types of measures that can be calculated and how they are calculated:

- ***Whether the dimension of inequality is ordered or non-ordered.*** Ordered dimensions have subgroups with an inherent ordering (such as education level or wealth quintiles), while non-

ordered and binary dimensions have subgroups that cannot logically be ranked (such as subnational regions, ethnicity, or sex).

- ***Whether the dimension of inequality has two or more than two subgroups.*** Simple measures can be calculated for all inequality dimensions, but complex measures can only be calculated for dimensions with more than two subgroups.
- ***The optimum level that is to be achieved for the health indicator.*** For favourable indicators, the goal is to attain a maximum level; for adverse indicators, the goal is to attain a minimum level. Note that for some indicators there is no optimum level. The optimum level impacts the calculation of certain summary measures.

There are also considerations relating to the properties of the different measures and the desired purpose of the analysis:

- ***Whether inequality is measured in absolute or relative terms.*** Absolute measures indicate the magnitude of inequality between population subgroups, while relative measures show proportional inequality between subgroups.
- ***Whether subgroups should be weighted according to their population size or not.*** This involves a value judgement depending on the purpose of the analysis.
- ***The selection of a reference point for non-ordered measures.*** For instance, this may be the mean, the best-performing subgroup, or a user-selected reference group.

The paper [Summary measures of health inequality: a review of existing measures and their application](#) provides further information about these considerations.

Data management considerations

The codes in the Stata do-files are designed to calculate a given summary measure of health inequality for a unique group of population subgroup estimates. If a dataset has multiple settings, dates, data sources, indicators and/or dimensions of inequality, then the code should be run for each unique set of estimates. For example, this could be done using the following steps:

1. Create a unique identifier for each group of estimates using the **group** command (enter as many variables within the brackets as is required to identify a unique group of estimates).
2. Save the dataset (either as a new .dta file or as a temporary file) with the new *group* variable.
3. Then use the **summarize** command to map the distinct values of the *group* variable. This stores the number of distinct values in **r(max)**.
4. Use the **forvalues** command to loop over consecutive values of *group* (from group=1 to group=the maximum value). Each value of *group* is saved in a local macro named *i*. The commands between the braces {} are then executed for each *group*.
5. The saved dataset must be re-opened for each *group* with the **use** command.
6. Only data for the selected group are retained using the **keep** command.
7. The summary measure is then calculated for each group using the relevant code.
8. Once the summary measure has been calculated for the group, use the **keep** command to retain the required variables (e.g., indicator, dimension, and the calculated summary measure estimates) in the first row, and store the results in a temporary file using the **tempfile** command with the group name *i*.
9. After the loop has completed, open the temporary file for the first group '1' and then append the files of all the other groups using the **append** command.

```
*Create unique group identifier
egen group=group(indicator dimension)
tempfile data
save `data'

*Run the calculation for each unique group of estimates
summarize group, meanonly
forvalues i = 1/`r(max)' {
    use `data', clear
    keep if group==`i'

    ... summary measure calculation code ...

    keep indicator dimension summary_measure
    keep in 1
    tempfile `i'
    save ``i'', replace
}

*Append results for all groups of estimates
use `1', clear
append using `2'
```

Difference

Overview

Type of summary measure: Simple; absolute; unweighted.

Definition: The difference (D) is an absolute measure of inequality that shows the difference in a health indicator between two population subgroups.

Interpretation: Greater absolute values indicate higher levels of inequality. D is zero if there is no inequality.

Calculation: D is calculated as:

$$D = y_1 - y_2$$

where y_1 and y_2 indicate the estimates for subgroups 1 and 2. The selection of the two subgroups depends on the characteristics of the inequality dimension and the purpose of the analysis. In addition, the direction of the calculation may depend on the indicator type (favourable or adverse). Below are some examples of how y_1 and y_2 may be identified:

Dimension type			Indicator type	
Number of subgroups	Ordering of subgroups	Reference subgroup	Favourable indicator	Adverse indicator
2 subgroups	N/A	Yes	Reference group – Other group	Other group – Reference group
		No	Highest – Lowest	Highest – Lowest
>2 subgroups	Ordered	N/A	Most-advantaged – most-disadvantaged	Most-disadvantaged – most-advantaged
	Non-ordered	Yes	Reference group – Lowest	Highest – Reference group
		No	Highest – Lowest	Highest – Lowest

The variance of D is calculated as:

$$var(D) = \sigma_1^2 + \sigma_2^2$$

where σ_1 and σ_2 indicate the standard errors of the estimates of subgroups 1 and 2.

Calculating D

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension can be ordered or non-ordered.	String
<i>subgroup</i>	Subgroups of the population within a given dimension of inequality. There can be two or more subgroups within the dimension of inequality.	String
<i>estimate</i>	The subgroup estimate. Estimates must be available for the two subgroups being compared.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If these are missing for the two subgroups being compared, 95% confidence intervals of D cannot be calculated.	Numeric

Optional variables for the identification of y_1 and y_2

Variable	Definition	Type
<i>ordered_dimension</i>	1 for ordered dimensions (which have subgroups with a natural ordered, such as economic status) and 0 for non-ordered or binary dimensions (such as subnational region or sex).	Numeric (0 or 1)
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1, if the dimension is ordered. For example, if measuring economic-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>favourable_indicator</i>	1 for favourable indicators (which measure desirable health events where the ultimate goal is to achieve a maximum level, such as skilled birth attendance) and 0 for non-favourable indicators (which measure undesirable health events where the ultimate goal is to achieve a minimum level, such as under-five mortality rate).	Numeric (0 or 1)
<i>reference_subgroup</i>	A reference subgroup indicated with the value of 1.	Numeric (0 or 1)

Calculation steps

Step 1

Calculate D for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Identify the two population subgroups to subtract. The selection of the two subgroups depends on the characteristics of the inequality dimension and the purpose of the analysis. An example is given here of how subgroups could be identified.

Generate an empty variable, *y*. In the subsequent steps, *y* will take the values of 1 and 2 to identify the two subgroup estimates that should be used to calculate the difference.

- If the dimension is ordered (*ordered_dimension*==1), execute the commands between the braces {}. If the indicator is favourable (*favourable_indicator*=1) then **sort** the estimates from the least to most advantaged. Otherwise, use the **gsort** command to sort the estimates from the most to least advantaged. *y*=1 for the estimate in the bottom row (identified by *_n*==*N*), which is the most advantaged subgroup for favourable indicators and the most disadvantaged subgroup for adverse indicators. *y*=2 for the estimate in the first row (identified by *_n*==1).
- If the dimension is non-ordered (*ordered_dimension*==0), execute the commands between the braces {}. First, use the **summarize** command to identify whether there is a reference subgroup identified or not. If not, **r(max)** will be equal to 0. Store the *r(max)* output in a scalar variable, *useref*.
 - If there is no reference subgroup (*useref*==0) then sort the estimates from the lowest to highest. *y*=1 for the highest estimate, which is in the bottom row (*_n*==*N*) and *y*=2 for the lowest estimate, which is in the first row (*_n*==1).
 - If there is a reference subgroup (*useref*!=0) and the indicator is favourable, *y*=1 for the reference subgroup estimate. *y*=2 for the lowest estimate (or the non-reference estimate, in the case of a binary dimension), identified using the **r(min)** stored result after using the **summarize** command.
 - If there is a reference subgroup (*useref*!=0) and the indicator is adverse, *y*=2 for the reference subgroup estimate. *y*=1 for the highest estimate (or the non-reference estimate, in the case of a binary dimension), identified using the **r(max)** stored result after using the **summarize** command.

```
gen y=.
*Ordered dimension
if ordered_dimension==1 {
    if favourable_indicator==1 sort subgroup_order
    else gsort -subgroup_order
    replace y=1 if _n==N
```

```

    replace y=2 if _n==1
}
*Non-ordered dimension
if ordered_dimension==0 {
    summarize reference_subgroup
    scalar useref=r(max)
    *No reference subgroup
    if useref==0 {
        sort estimate
        replace y=1 if _n==_N
        replace y=2 if _n==1
    }
    *Reference subgroup, favourable indicator
    if useref!=0 & favourable_indicator==1 {
        replace y=1 if reference_subgroup==1
        summarize estimate if reference_subgroup!=1
        replace y=2 if estimate==r(min)
    }
    *Reference subgroup, non-favourable indicator
    if useref!=0 & favourable_indicator==0 {
        replace y=2 if reference_subgroup==1
        summarize estimate if reference_subgroup!=1
        replace y=1 if estimate==r(max)
    }
}
}

```

Step 3 Reshape the dataset.

- 3.1 **Keep** only the indicators required for the next calculation steps – *indicator*, *dimension*, *estimate*, *se*, *favourable_indicator* and *y*.
- 3.2 **Drop** all subgroups that are not required for the calculation of difference ($y=.$).
- 3.3 Use the **reshape** command to reshape the data to a wide format. The variables *estimate* and *se* are reshaped, based on the variable *y*. The variables within **i()** are the unique identifiers of observations in a wide format. This results in the creation of four variables – *estimate1*, *estimate2*, *se1* and *se2*. *estimate1* contains the estimates where $y=1$, *se1* contains the standard errors where $y=1$, and so on.

```

keep indicator dimension estimate se favourable_indicator y
drop if y==.
reshape wide estimate se, i(indicator dimension favourable_indicator) j(y)

```

Step 4 Calculate D.

Generate a variable *d* that is equal to *estimate1* minus *estimate2*. Note that the identification of *estimate1* and *estimate2* and the direction of the calculation (depending on whether the indicator is favourable or not) has already been set using the process described in Step 2.

```
gen d=estimate1-estimate2
```

Step 5 Calculate 95% confidence intervals of D.

- 5.1 The standard error of the difference (d_{se}) is the square root of the variance, which is calculated as the standard error of the first subgroup ($se1$) squared plus the standard error of the second subgroup ($se2$) squared.
- 5.2 The lower 95% confidence interval (d_{ll}) is calculated as the standard error of the difference multiplied by 1.96, subtracted from the difference.
- 5.3 The upper 95% confidence interval (d_{ul}) is calculated as the standard error of the difference multiplied by 1.96, added to the difference.

```
gen d_se=sqrt(se1^2+se2^2)
gen d_ll=d-1.96*d_se
gen d_ul=d+1.96*d_se
```

Ratio

Overview

Type of summary measure: Simple; relative; unweighted.

Definition: The ratio (R) is a relative measure of inequality that shows the ratio of a health indicator between two population subgroups.

Interpretation: R only assumes positive values. The further the value of R from one, the higher the level of inequality. R is one if there is no inequality.

Calculation: R is calculated as:

$$R = \frac{y_1}{y_2}$$

where y_1 and y_2 indicate the estimates for subgroups 1 and 2. The selection of the two subgroups depends on the characteristics of the inequality dimension and the purpose of the analysis. In addition, the direction of the calculation may depend on the indicator type (favourable or adverse). Below are some examples of how y_1 and y_2 may be identified:

Dimension type			Indicator type	
Number of subgroups	Ordering of subgroups	Reference subgroup	Favourable indicator	Adverse indicator
2 subgroups	N/A	Yes	Reference group – Other group	Other group – Reference group
		No	Highest – Lowest	Highest – Lowest
>2 subgroups	Ordered	N/A	Most-advantaged – most-disadvantaged	Most-disadvantaged – most-advantaged
	Non-ordered	Yes	Reference group – Lowest	Highest – Reference group
		No	Highest – Lowest	Highest – Lowest

The variance of R is calculated as:

$$var(R) = \left(\frac{1}{y_2}\right)^2 * \left(\sigma_1^2 + \left(\frac{y_1}{y_2}\right)^2 \sigma_2^2\right)$$

where σ_1 and σ_2 indicate the standard errors of the estimates of subgroups 1 and 2.

Calculating R

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension can be ordered or non-ordered.	String
<i>subgroup</i>	Subgroups of the population within a given dimension of inequality. There can be two or more subgroups within the dimension of inequality.	String
<i>estimate</i>	The subgroup estimate. Estimates must be available for the two subgroups being compared.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If these are missing for the two subgroups being compared, 95% confidence intervals of R cannot be calculated.	Numeric

Optional variables for the identification of y_1 and y_2

Variable	Definition	Type
<i>ordered_dimension</i>	1 for ordered dimensions (which have subgroups with a natural ordered, such as economic status) and 0 for non-ordered or binary dimensions (such as subnational region or sex).	Numeric (0 or 1)
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1, if the dimension is ordered. For example, if measuring economic-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>favourable_indicator</i>	1 for favourable indicators (which measure desirable health events where the ultimate goal is to achieve a maximum level, such as skilled birth attendance) and 0 for non-favourable indicators (which measure undesirable health events where the ultimate goal is to achieve a minimum level, such as under-five mortality rate).	Numeric (0 or 1)
<i>reference_subgroup</i>	A reference subgroup indicated with the value of 1.	Numeric (0 or 1)

Calculation steps

Step 1

Calculate R for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Identify the two population subgroups to divide. The selection of the two subgroups depends on the characteristics of the inequality dimension and the purpose of the analysis. An example is given here of how subgroups could be identified.

Generate an empty variable, *y*. In the subsequent steps, *y* will take the values of 1 and 2 to identify the two subgroup estimates that should be used to calculate the ratio.

- If the dimension is ordered (*ordered_dimension*==1), execute the commands between the braces {}. If the indicator is favourable (*favourable_indicator*=1) then **sort** the estimates from the least to most advantaged. Otherwise, use the **gsort** command to sort the estimates from the most to least advantaged. *y*=1 for the estimate in the bottom row (identified by *_n*==*N*), which is the most advantaged subgroup for favourable indicators and the most disadvantaged subgroup for adverse indicators. *y*=2 for the estimate in the first row (identified by *_n*==1).
- If the dimension is non-ordered (*ordered_dimension*==0), execute the commands between the braces {}. First, use the **summarize** command to identify whether there is a reference subgroup identified or not. If not, **r(max)** will be equal to 0. Store the *r(max)* output in a scalar variable, *useref*.
 - If there is no reference subgroup (*useref*==0) then sort the estimates from the lowest to highest. *y*=1 for the highest estimate, which is in the bottom row (*_n*==*N*) and *y*=2 for the lowest estimate, which is in the first row (*_n*==1).
 - If there is a reference subgroup (*useref*!=0) and the indicator is favourable, *y*=1 for the reference subgroup estimate. *y*=2 for the lowest estimate (or the non-reference estimate, in the case of a binary dimension), identified using the **r(min)** stored result after using the **summarize** command.
 - If there is a reference subgroup (*useref*!=0) and the indicator is adverse, *y*=2 for the reference subgroup estimate. *y*=1 for the highest estimate (or the non-reference estimate, in the case of a binary dimension), identified using the **r(max)** stored result after using the **summarize** command.

```
gen y=.
*Ordered dimension
if ordered_dimension==1 {
    if favourable_indicator==1 sort subgroup_order
    else gsort -subgroup_order
    replace y=1 if _n==N
}
```

```

    replace y=2 if _n==1
  }
  *Non-ordered dimension
  if ordered_dimension==0 {
    summarize reference_subgroup
    scalar useref=r(max)
    *No reference subgroup
    if useref==0 {
      sort estimate
      replace y=1 if _n==_N
      replace y=2 if _n==1
    }
    *Reference subgroup, favourable indicator
    if useref!=0 & favourable_indicator==1 {
      replace y=1 if reference_subgroup==1
      summarize estimate if reference_subgroup!=1
      replace y=2 if estimate==r(min)
    }
    *Reference subgroup, non-favourable indicator
    if useref!=0 & favourable_indicator==0 {
      replace y=2 if reference_subgroup==1
      summarize estimate if reference_subgroup!=1
      replace y=1 if estimate==r(max)
    }
  }
}

```

Step 3

Reshape the dataset.

- 3.1 **Keep** only the indicators required for the next calculation steps – *indicator*, *dimension*, *estimate*, *se*, *favourable_indicator* and *y*.
- 3.2 **Drop** all subgroups that are not required for the calculation of difference ($y=.$).
- 3.3 Use the **reshape** command to reshape the data to a wide format. The variables *estimate* and *se* are reshaped, based on the variable *y*. The variables within **i()** are the unique identifiers of observations in a wide format. This results in the creation of four variables – *estimate1*, *estimate2*, *se1* and *se2*. *estimate1* contains the estimates where $y=1$, *se1* contains the standard errors where $y=1$, and so on.

```

keep indicator dimension estimate se favourable_indicator y
drop if y==.
reshape wide estimate se, i(indicator dimension favourable_indicator) j(y)

```

Step 4

Calculate R.

Generate a variable *r* that is equal to *estimate1* divided by *estimate2*. Note that the identification of *estimate1* and *estimate2* and the direction of the calculation (depending on whether the indicator is favourable or not) has already been set using the process described in Step 2.

```
gen r=estimate1/estimate2
```

Step 5 Calculate 95% confidence intervals of R.

- 5.1 Calculate the variance of the ratio as 1 divided by the *estimate2* squared, multiplied by the standard error of *estimate1* squared, plus the ratio squared multiplied by the standard error of *estimate2* squared. The square root of the result gives the standard error of the ratio (*r_se*).
- 5.2 Calculate the natural log of the ratio, *r_log* (i.e., do a log transformation of the ratio, to transform it to be more-or-less normally distributed).
- 5.3 Calculate the standard error for the log transformed ratio (*r_log_se*) by dividing the standard error of the ratio (*r_se*) by the ratio (*r*).
- 5.4 The lower 95% confidence interval (*r_ll*) is calculated as the exponential of the standard error of the log transformed ratio (*r_log_se*) multiplied by 1.96, subtracted from the log transformed ratio (*r_log*).
- 5.5 The upper 95% confidence interval (*r_ul*) is calculated as the exponential of the standard error of the log transformed ratio (*r_log_se*) multiplied by 1.96, added to the log transformed ratio (*r_log*).

```
gen r_se=sqrt(((1/estimate2^2))*((se1^2)+((r^2)*(se2^2))))
gen r_log=log(r)
gen r_log_se=r_se/r
gen r_ll=exp(r_log-1.96*r_log_se)
gen r_ul=exp(r_log+1.96*r_log_se)
```


Absolute Concentration Index

Overview

Type of summary measure: Complex; absolute; weighted.

Applicability: Ordered; more than two subgroups

Definition: The absolute concentration index (ACI) is an absolute measure of inequality that indicates the extent to which an indicator is concentrated among disadvantaged or advantaged subgroups, on an absolute scale. ACI can be calculated using disaggregated data and individual-level data. Subgroups in disaggregated data are weighted according to their population share, while individuals are weighted by sample weight in the case of data from surveys.

Interpretation: The larger the absolute value of ACI, the higher the level of inequality. Positive values indicate a concentration of the indicator among advantaged subgroups, and negative values indicate a concentration of the indicator among disadvantaged subgroups. ACI is zero if there is no inequality.

Calculation: The calculation of ACI is based on a ranking of the whole population from the most-disadvantaged subgroup (at rank 0) to the most-advantaged subgroup (at rank 1), which is inferred from the ranking and size of the subgroups. The relative rank of each subgroup is calculated as: $X_j = \sum_j p_j - 0.5p_j$. Based on this ranking, ACI can be calculated as:

$$ACI = \sum_j p_j (2X_j - 1) y_j$$

where y_j indicates the estimate for the subgroup j , p_j the population share of subgroup j and X_j the relative rank of subgroup j . The variance of ACI is calculated as:

$$var(ACI) = \sum_j p_j^2 \sigma_j^2 (2X_j - 1)^2$$

where σ_j is the standard error of the estimate for the subgroup j .

For individual-level data, ACI can be calculated as twice the covariance between the health indicator (h) and the relative rank (r):

$$ACI = 2cov(h, r) = \alpha + \beta r_i + \epsilon$$

Given the relationship between covariance and ordinary least squares regression, ACI can also be obtained from a regression of a transformation of the health variable of interest on the relative rank. The variance of ACI can then be calculated from the regression model using the delta method.

Calculating ACI using disaggregated data

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education). The dimension must be ordered, meaning that it has subgroups with an inherent ordering. For example, a dimension of economic status with wealth quintiles that can be ranked from poorest to richest.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a dimension of economic status, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5).	String
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring economic-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>estimate</i>	The subgroup estimate. Estimates must be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of ACI cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric

Calculation steps

Step 1

Calculate ACI for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Rank the population from the most-disadvantaged subgroup (at rank 0) to the most-advantaged subgroup (at rank 1), based on the midpoint of its range in the cumulative population distribution.

- 2.1 Sort the data according to *subgroup_order* using the **sort** command. The subgroups should be sorted from the most-disadvantaged to the most-advantaged for the subsequent steps.
- 2.2 Use the **summarize** command to calculate the total population.
- 2.3 **Generate** a variable *popshare*, which is the share of each subgroup's population out of the total population (stored in **r(sum)** following the previous step).
- 2.4 Calculate the cumulative total of the population share of each subgroup, *cumshare*.
- 2.5 Calculate the midpoint of the population share of each subgroup (**0.5*popshare**) and subtract this from the cumulative total population share (*cumshare*). This produces the cumulative midpoint proportion of each subgroup, saved as *rank*.

```
sort subgroup_order
summarize population
gen popshare=population/r(sum)
gen cumshare=sum(popshare)
gen rank=cumshare-0.5*popshare
```

Step 3

Calculate the ACI.

- 3.1 **Generate** a variable *aci_prep*, which is equal to the population share of each subgroup (*popshare*) multiplied by two times the subgroup's relative *rank* minus one, multiplied by the subgroup *estimate*.
- 3.2 Sum the *aci_prep* values across all subgroups using the **total** command, storing the result in a new variable, *aci*.

```
gen aci_prep=popshare*(2*rank-1)*estimate
egen aci=total(aci_prep)
```

Step 4

Calculate the 95% confidence intervals of the ACI.

- 4.1 Calculate the variance of ACI (*aci_var*) as the population share (*popshare*) squared, multiplied by two times the subgroup's relative *rank* minus one squared, multiplied by the standard error of the subgroup estimate (*se*) squared. Use the **total** command to sum the result across all subgroups.
- 4.2 The standard error of ACI (*aci_se*) is the square root of the variance.
- 4.3 The lower 95% confidence interval (*aci_ll*) is calculated as the standard error of ACI multiplied by 1.96, subtracted from ACI.

- 4.4 The upper 95% confidence interval (*aci_ul*) is calculated as the standard error of ACI estimate multiplied by 1.96, added to ACI.

```
egen aci_var=total((popshare^2)*((2*rank-1)^2)*(se^2))
gen aci_se=sqrt(aciVar)
gen aci_ll=aci-1.96*aci_se
gen aci_ul=aci+1.96*aci_se
```

Calculating ACI using individual-level data

Data requirements

Variable	Definition	Type
<i>id</i>	Individual record identifier	String or numeric
<i>psu</i>	Primary sampling unit (required if data come from a survey)	Numeric
<i>weight</i>	Individual sampling weight (required if data come from a survey)	Numeric
<i>strata</i>	Strata (required if data come from a survey)	Numeric
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. The dimension must be ordered, meaning that it has subgroups with an inherent ordering, and there must be more than two subgroups within the dimension of inequality. For example, within an economic status dimension, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5).	String
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring wealth-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>indicator</i>	Variable containing the indicator that is being measured. Can be categorized as a binary variable (with 0 meaning that the individual has not met the indicator and 1 meaning that the individual has met the indicator) or continuous (e.g., individual height or blood pressure).	Numeric

Calculation steps

Step 1

Declare survey design if the data come from a survey.

The **svyset** command declares the survey design for the dataset. Depending on the nature of the survey, specify **svyset** options. This example uses data from a DHS survey, in which *psu*, *weight* and *strata* must be specified.

```
svyset psu [pweight=weight], strata(strata)
```

Step 2

Rank the population from the most-disadvantaged (at rank 0) to the most-advantaged (at rank 1), taking individual weights into account.

- 2.1 Use the **drop** command to drop any records where there is no indicator estimate.
- 2.2 **Sort** records according to the *subgroup_order* variable.
- 2.3 **Generate** a variable *sumw* which is the sum of all individual weights.
- 2.4 **Generate** a variable *cumw* which is the cumulative sum of each individual weight. The values of *cumw* are moved one row down in the variable *cumw_1*, with the value in the first row being zero.
- 2.5 Calculate the cumulative proportion of individuals within each subgroup. For each value of the *subgroup_order* variable, **generate** a variable *cumwr* containing the maximum value of the cumulative individual weight (*cumw*). Therefore, if individuals are being ranked by a categorical variable (such as wealth quintiles or education level) then there is a single *cumwr* value per subgroup. If individuals are being ranked by a continuous variable (such as wealth index scores) then each individual would have a *cumwr* value (apart from ties, when two individuals have the same ranking value).
- 2.6 Calculate the cumulative proportion of individuals within the previous subgroup. For each value of the *subgroup_order* variable, **generate** a variable *cumwr_1* containing the minimum value of the *cumw* variable.
- 2.7 Calculate the population share of individuals within each subgroup by subtracting *cumwr_1* from *cumwr*. Calculate the midpoint of this population share by multiplying this by 0.5. Add this to *cumwr_1* to get the cumulative midpoint of the population share. Divide by the sum of all individual weights (*sumw*) to produce the relative ranking of each individual (*rank*), which is the proportion of individuals within a given value of the ranking variable.

```
drop if indicator==.
sort subgroup_order
egen sumw=sum(weight)
gen cumw=sum(weight)
gen cumw_1=cumw[_n-1]
    replace cumw_1=0 if cumw_1==.
bysort subgroup_order: egen cumwr=max(cumw)
bysort subgroup_order: egen cumwr_1=min(cumw_1)
gen rank=(cumwr_1+0.5*(cumwr-cumwr_1))/sumw
```

Step 3

Regress the indicator of interest against the rank of each individual using a regression model, accounting for survey design if necessary.

- 3.1 Generate a variable *intercept* that will reflect the individual sampling weights. If weights are declared using **svyset** in Step 1, then they will automatically be accounted for in the regression model using the **svy** prefix in later steps (so *intercept* will take the value 1, having no impact on the regression model). However, if weights are not declared using **svyset** in Step 1, the square root of sampling weights should be used as weights in the least squares regression model.

- 3.3 **Generate** a variable *temp*, which is equal to the share of each individual's weight out of the sum of all individual weights (*sumw*), multiplied by the individual's relative *rank* minus 0.5, squared.
- 3.2 **Generate** a variable *sigma*, which is the sum of *temp* across all records.
- 3.3 Calculate the weighted mean indicator estimate, *meanlhs*. The *temp* variable is multiplied by each individual's indicator estimate, summed across all records, and then divided by the total weight (*sumw*).
- 3.4 **Generate** a variable *lhs* for the weighted indicator estimate, which is equal to two times the variable *sigma*, multiplied by the indicator estimate divided by the mean (*meanlhs*), multiplied by the *intercept* variable (to account for weighting if this is not declared using **svyset**). This is then multiplied by the mean (*meanlhs*), which leads to an assessment of absolute inequality.
- 3.5 **Generate** a variable *rhs* for the weighted rank, which is the individual *rank* multiplied by the *intercept* variable (to account for weighting if this is not declared using **svyset**).
- 3.6 Use the **regress** command to run a linear regression model, with the weighted indicator estimate (*lhs*), the weighted rank (*rhs*) and the *intercept* variable. If the data come from a survey, the **svy** prefix is used to take survey design (defined in Step 1) into account. The **noconstant** option is used to force the constant term (i.e., the intercept in the regression model) to be zero.

```

gen intercept=1                                // if data from a survey
*gen intercept=sqrt(weight)                    // if data not from a survey
gen temp=(weight/sumw)*((rank-0.5)^2)
egen sigma=sum(temp)
    replace temp=weight*indicator
egen meanlhs=sum(temp)
    replace meanlhs=meanlhs/sumw
gen lhs=2*sigma*(indicator/meanlhs)*intercept
    replace lhs=lhs*meanlhs
gen rhs=rank*intercept
svy: regress lhs rhs intercept, noconstant      // if data from a survey
*regress lhs rhs intercept, noconstant        // if data not from a survey

```

Step 4 Calculate the ACI.

Generate a new variable, *aci*, equal to the ACI estimate that was stored at position [1,1] of matrix *e(b)* after running the **regress** command. This is the coefficient of *rhs* in the regression model.

```
gen aci=e(b) [1,1]
```

Step 5 Calculate and store the 95% confidence intervals of the ACI.

- 5.1 Use the **matrix** command to store the variance-covariance matrix (saved in **e(V)** after running the **regress** command) in a matrix called V.
- 5.2 The standard error of ACI (*aci_se*) is the square root of the variance-covariance estimate, stored at position [1,1] of matrix V.
- 5.3 The lower 95% confidence interval (*aci_ll*) is calculated as the standard error of ACI multiplied by 1.96, subtracted from ACI (if the standard error is not equal to zero).
- 5.4 The upper 95% confidence interval (*aci_ul*) is calculated as the standard error of ACI multiplied by 1.96, added to ACI (if the standard error is not equal to zero).

```
matrix V=e(V)
gen aci_se=sqrt(V[1,1])
gen aci_ll=aci-1.96*aci_se if aci_se!=0
gen aci_ul=aci+1.96*aci_se if aci_se!=0
```

Controlling for other factors at the individual-level

An advantage of calculating ACI at the individual level is the ability to control for other factors and characteristics; for example, measuring wealth-related inequality while controlling for the influence of education and place of residence.

To control for other factors, these variables need to be present in the individual-level dataset. Then insert these variables in the regression model in Step 3 above. For example, the model below controls for education (*educ*) and place of residence (*urban*). Steps 4-5 are the same as with univariate analysis.

```
svy: regress lhs rhs intercept educ urban, noconstant
```


Relative Concentration Index

Overview

Type of summary measure: Complex; relative; weighted.

Applicability: Ordered; more than two subgroups

Definition: The relative concentration index (RCI) is a relative measure of inequality that shows the gradient across population subgroups. It indicates the extent to which an indicator is concentrated among disadvantaged or advantaged subgroups, on a relative scale. RCI can be calculated using disaggregated data and individual-level data. Subgroups in disaggregated data are weighted according to their population share, while individuals are weighted by sample weight in the case of data from surveys.

Interpretation: RCI is bounded between -1 and +1. The larger the absolute value of RCI, the higher the level of inequality. Positive values indicate a concentration of the indicator among advantaged subgroups, and negative values indicate a concentration of the indicator among disadvantaged subgroups. RCI is zero if there is no inequality.

Calculation: RCI is calculated by dividing the absolute concentration index (ACI) by the setting average μ . The calculation of RCI is based on a ranking of the whole population from the most-disadvantaged subgroup (at rank 0) to the most-advantaged subgroup (at rank 1), which is inferred from the ranking and size of the subgroups. The relative rank of each subgroup is calculated as: $X_j = \sum_j p_j - 0.5p_j$. RCI can be calculated as:

$$RCI = \frac{\sum_j p_j (2X_j - 1) y_j}{\mu}$$

where y_j indicates the estimate for the subgroup j ; p_j the population share of subgroup j ; X_j the relative rank of subgroup j ; and μ is the indicator mean. The variance of RCI is calculated as:

$$var(RCI) = \frac{\sum_j p_j^2 \sigma_j^2 [(2X_j - 1) - RCI]^2}{\mu^2}$$

where σ_j is the standard error of the estimate for the subgroup j .

For individual-level data, RCI can be calculated as twice the covariance between the health indicator (h) and the relative rank (r), divided by the indicator mean (μ):

$$RCI = \frac{2cov(h, r)}{\mu}$$

Given the relationship between covariance and ordinary least squares regression, RCI can also be obtained from a regression of a transformation of the health variable of interest on the relative rank. The variance of RCI can then be calculated from the regression model using the delta method.

Calculating RCI using disaggregated data

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education). The dimension must be ordered, meaning that it has subgroups with an inherent ordering. For example, a dimension of economic status with wealth quintiles that can be ranked from poorest to richest.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a dimension of economic status, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5).	String
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring economic-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>estimate</i>	The subgroup estimate. Estimates must be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of RCI cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric

Calculation steps

Step 1

Calculate RCI for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Rank the population from the most-disadvantaged subgroup (at rank 0) to the most-advantaged subgroup (at rank 1), based on the midpoint of its range in the cumulative population distribution.

- 2.1 Sort the data according to *subgroup_order* using the **sort** command. The subgroups should be sorted from the most-disadvantaged to the most-advantaged for the subsequent steps.
- 2.2 Use the **summarize** command to calculate the total population.
- 2.3 **Generate** a variable *popshare*, which is the share of each subgroup's population out of the total population (stored in **r(sum)** following the previous step).
- 2.4 Calculate the cumulative total of the population share of each subgroup, *cumshare*.
- 2.5 Calculate the midpoint of the population share of each subgroup (**0.5*popshare**) and subtract this from the cumulative total population share (*cumshare*). This produces the cumulative midpoint proportion of each subgroup, saved as *rank*.

```
sort subgroup_order
summarize population
gen popshare=population/r(sum)
gen cumshare=sum(popshare)
gen rank=cumshare-0.5*popshare
```

Step 3

Calculate the weighted mean of subgroup estimates.

Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```
egen weighted_mean=total(popshare*estimate)
```

Step 4

Calculate RCI as ACI divided by the weighted mean of subgroup estimates.

- 4.1 **Generate** a variable *aci_prep*, which is equal to the population share of each subgroup (*popshare*) multiplied by two times the subgroup's relative *rank* minus one, multiplied by the subgroup *estimate*.

- 4.2 Sum the *aci_prep* values across all subgroups using the **total** command, storing the result in a new variable, *aci*.
- 4.3 Calculate RCI as ACI (*aci*) divided by the weighted mean (*weighted_mean*), multiplied by 100.

```
gen aci_prep=popshare*(2*rank-1)*estimate
egen aci=total(aci_prep)
gen rci=(aci/weighted_mean)*100
```

Step 5 Calculate the 95% confidence intervals of RCI.

- 5.1 Calculate two times the subgroup's relative *rank* minus one, subtract RCI and square the result. Multiply this by the population share (*popshare*) squared and the standard error of the subgroup estimate (*se*) squared. Use the **total** command to sum the result across all subgroups, saved as *rci_prep*.
- 5.2 The standard error of RCI (*rci_se*) is the square root of the variance, which is calculated as *rci_prep* divided by the weighted mean (*weighted_mean*) squared.
- 5.3 The lower 95% confidence interval (*rci_ll*) is calculated as the standard error of RCI multiplied by 1.96, subtracted from RCI.
- 5.4 The upper 95% confidence interval (*rci_ul*) is calculated as the standard error of RCI multiplied by 1.96, added to RCI.

```
egen rci_prep=total((popshare^2)*((2*rank-1)-rci)^2*(se^2))
gen rci_se=sqrt(rci_prep/(weighted_mean^2))
gen rci_ll=rci-1.96*rci_se
gen rci_ul=rci+1.96*rci_se
```

Calculating RCI using individual-level data

Data requirements

Variable	Definition	Type
<i>id</i>	Individual record identifier	String or numeric
<i>psu</i>	Primary sampling unit (required if data come from a survey)	Numeric
<i>weight</i>	Individual sampling weight (required if data come from a survey)	Numeric
<i>strata</i>	Strata (required if data come from a survey)	Numeric
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. The dimension must be ordered, meaning that it has subgroups with an inherent ordering, and there must be more than two subgroups within the dimension of inequality. For example, within an economic status dimension, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5).	String
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring wealth-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>indicator</i>	Variable containing the indicator that is being measured. Can be categorized as a binary variable (with 0 meaning that the individual has not met the indicator and 1 meaning that the individual has met the indicator) or continuous (e.g., individual height or blood pressure).	Numeric

Calculation steps

Step 1

Declare survey design, if the data come from a survey.

The **svyset** command declares the survey design for the dataset. Depending on the nature of the survey, specify **svyset** options. This example uses data from a DHS survey, in which *psu*, *weight* and *strata* must be specified.

```
svyset psu [pweight=weight], strata(strata)
```

Step 2

Rank the population from the most-disadvantaged (at rank 0) to the most-advantaged (at rank 1), taking individual weights into account.

- 2.1 Use the **drop** command to drop any records where there is no indicator estimate.
- 2.2 **Sort** records according to the *subgroup_order* variable.
- 2.3 **Generate** a variable *sumw* which is the sum of all individual weights.
- 2.4 **Generate** a variable *cumw* which is the cumulative sum of each individual weight. The values of *cumw* are moved one row down in the variable *cumw_1*, with the value in the first row being zero.
- 2.5 Calculate the cumulative proportion of individuals within each subgroup. For each value of the *subgroup_order* variable, **generate** a variable *cumwr* containing the maximum value of the cumulative individual weight (*cumw*). Therefore, if individuals are being ranked by a categorical variable (such as wealth quintiles or education level) then there is a single *cumwr* value per subgroup. If individuals are being ranked by a continuous variable (such as wealth index scores) then each individual would have a *cumwr* value (apart from ties, when two individuals have the same ranking value).
- 2.6 Calculate the cumulative proportion of individuals within the previous subgroup. For each value of the *subgroup_order* variable, **generate** a variable *cumwr_1* containing the minimum value of the *cumw* variable.
- 2.7 Calculate the population share of individuals within each subgroup by subtracting *cumwr_1* from *cumwr*. Calculate the midpoint of this population share by multiplying this by 0.5. Add this to *cumwr_1* to get the cumulative midpoint of the population share. Divide by the sum of all individual weights (*sumw*) to produce the relative ranking of each individual (*rank*), which is the proportion of individuals within a given value of the ranking variable.

```
drop if indicator==.
sort subgroup_order
egen sumw=sum(weight)
gen cumw=sum(weight)
gen cumw_1=cumw[_n-1]
    replace cumw_1=0 if cumw_1==.
bysort subgroup_order: egen cumwr=max(cumw)
bysort subgroup_order: egen cumwr_1=min(cumw_1)
gen rank=(cumwr_1+0.5*(cumwr-cumwr_1))/sumw
```

Step 3

Regress the indicator of interest against the rank of each individual using a regression model, accounting for survey design if necessary.

- 3.1 Generate a variable *intercept* that will reflect the individual sampling weights. If weights are declared using **svyset** in Step 1, then they will automatically be accounted for in the regression model using the **svy** prefix in later steps (so *intercept* will take the value 1, having no impact on the regression model). However, if weights are not declared using **svyset** in Step 1, the square root of sampling weights should be used as weights in the least squares regression model.

- 3.2 **Generate** a variable *temp*, which is equal to the share of each individual's weight out of the sum of all individual weights (*sumw*), multiplied by the individual's relative *rank* minus 0.5, squared.
- 3.3 **Generate** a variable *sigma*, which is the sum of *temp* across all records.
- 3.4 Calculate the weighted mean indicator estimate, *meanlhs*. The *temp* variable is multiplied by each individual's indicator estimate, summed across all records, and then divided by the total weight (*sumw*).
- 3.5 **Generate** a variable *lhs* for the weighted indicator estimate, which is equal to two times the variable *sigma*, multiplied by the indicator estimate divided by the mean (*meanlhs*), multiplied by the *intercept* variable (to account for weighting if this is not declared using **svyset**).
- 3.6 **Generate** a variable *rhs* for the weighted rank, which is the individual *rank* multiplied by the *intercept* variable (to account for weighting if this is not declared using **svyset**).
- 3.7 Use the **regress** command to run a linear regression model, with the weighted indicator estimate (*lhs*), the weighted rank (*rhs*) and the *intercept* variable. If the data come from a survey, the **svy** prefix is used to take survey design (defined in Step 1) into account. The **noconstant** option is used to force the constant term (i.e., the intercept in the regression model) to be zero.

```

gen intercept=1                                // if data from a survey
*gen intercept=sqrt(weight)                     // if data not from a survey
gen temp=(weight/sumw)*((rank-0.5)^2)
egen sigma=sum(temp)
    replace temp=weight*indicator
egen meanlhs=sum(temp)
    replace meanlhs=meanlhs/sumw
gen lhs=2*sigma*(indicator/meanlhs)*intercept
gen rhs=rank*intercept
svy: regress lhs rhs intercept, noconstant      // if data from a survey
*regress lhs rhs intercept, noconstant         // if data not from a survey

```

Step 4 Calculate the RCI.

Generate a new variable, *rci*, equal to the RCI estimate, stored at position [1,1] of matrix **e(b)** after running the **regress** command. This is the coefficient of *rhs* in the regression model.

```
gen rci=e(b) [1,1]
```

Step 5 Calculate and store the 95% confidence intervals of the RCI.

- 5.1 Use the **matrix** command to store the variance-covariance matrix (saved in **e(V)** after running the **regress** command) in a matrix called *V*.

- 5.2 The standard error of RCI (*aci_se*) is the square root of the variance-covariance estimate, stored at position [1,1] of matrix V.
- 5.3 The lower 95% confidence interval (*rci_ll*) is calculated as the standard error of RCI multiplied by 1.96, subtracted from RCI (if the standard error is not equal to zero).
- 5.4 The upper 95% confidence interval (*rci_ul*) is calculated as the standard error of RCI multiplied by 1.96, added to RCI (if the standard error is not equal to zero).

```
matrix V=e(V)
gen rci_se=sqrt(V[1,1])
gen rci_ll=rci-1.96*rci_se if rci_se!=0
gen rci_ul=rci+1.96*rci_se if rci_se!=0
```

Controlling for other factors at the individual-level

An advantage of calculating RCI at the individual level is the ability to control for other factors and characteristics; for example, measuring wealth-related inequality while controlling for the influence of education and place of residence.

To control for other factors, these variables need to be present in the individual-level dataset. Then insert these variables in the regression model in Step 3 above. For example, the model below controls for education (*educ*) and place of residence (*urban*). Steps 4-5 are the same as with univariate analysis.

```
svy: regress lhs rhs intercept educ urban, noconstant
```

Measurement scale and bounded health indicators

When health variables are not on a fixed scale (i.e., a measurement scale that has zero corresponding to a situation of complete absence) or when variables are bounded and have a finite upper limit (such as years in school, a health utility index, or any binary indicator), a modified version of RCI may need to be used. See the following resources:

O'Donnell O, Van Doorslaer DE, Wagstaff A, Lindelow M. Analyzing Health Equity Using Household Survey Data A Guide to Techniques and Their Implementation. Washington DC: The World Bank; 2008.

Erreygers G. Correcting the concentration index. Journal of health economics. 2009 Mar 1;28(2):504-15.

Wagstaff A. The bounds of the concentration index when the variable of interest is binary, with an application to immunization inequality. Health economics. 2005 Apr;14(4):429-32.

Erreygers G and Van Ourti T. Measuring socioeconomic inequality in health, health care and health financing by means of rank-dependent indices: A recipe for good practice. Journal of Health Economics. 2011 Jul;30(4):685-694.

Wagstaff A. The concentration index of a binary outcome revisited. Health economics. 2011 Oct;20(10):1155-60.

Kjellsson G and Gerdtham U. On correcting the concentration index for binary variables. Journal of Health Economics. 2013 May;32(3):659-70.

Slope Index of Inequality

Overview

Type of summary measure: Complex; absolute; weighted.

Applicability: Ordered; more than two subgroups

Definition: The slope index of inequality (SII) is an absolute measure of inequality that represents the difference in estimated indicator values between the most-advantaged and most-disadvantaged, while taking into consideration the situation in all other subgroups/individuals – using an appropriate regression model. SII can be calculated using both disaggregated data and individual-level data. Subgroups in disaggregated data are weighted according to their population share, while individuals are weighted by sample weight in the case of data from surveys.

Interpretation: Greater absolute values indicate higher levels of inequality. Positive values indicate that the level of the indicator is higher among advantaged subgroups, while negative values indicate that the level of the indicator is higher among disadvantaged subgroups. SII is zero if there is no inequality.

Calculation: To calculate SII, a weighted sample of the whole population is ranked from the most-disadvantaged subgroup (at rank 0) to the most-advantaged subgroup (at rank 1). This ranking is weighted, accounting for the proportional distribution of the population within each subgroup. The relative rank of each subgroup is calculated as: $X_j = \sum_j p_j - 0.5p_j$.

The indicator of interest is then regressed against this relative rank using an appropriate regression model (e.g., a generalized linear model with logit link), and the predicted values of the indicator are calculated for the two extremes (rank 1 and rank 0). The difference between the predicted values at rank 1 and rank 0 (covering the entire distribution) generates the SII value:

$$SII = \hat{v}_1 - \hat{v}_0$$

The variance of SII is calculated from the regression model using the delta method.

Calculating SII using disaggregated data

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education). The dimension must be ordered, meaning that it has subgroups with an inherent ordering. For example, an economic status dimension with wealth quintiles that can be ranked from poorest to richest.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within an economic status dimension, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5).	String
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring economic-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>estimate</i>	The subgroup estimate. Estimates must be available for all subgroups.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric

Calculation steps

Step 1

Calculate SII for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Rescale estimates to a 0-1 scale.

- 2.1 Use the **summarize** command to identify the maximum estimate, which stored in **r(max)** after running the command.

- 2.2 **Generate** a variable *scale*, which takes the value of 1 if the estimates are all less than 1, 100 if the estimates are between 1 and 100, and so on.
- 2.3 Divide the estimates by *scale* to convert estimates to be on a range between 0 and 1. Save this as *estimate_sc*.

```
summarize estimate
gen scale=1 if r(max)<=1
    replace scale=100 if r(max)>1 & r(max)<=100
    replace scale=1000 if r(max)>100 & r(max)<=1000
    replace scale=10000 if r(max)>1000 & r(max)<=10000
    replace scale=100000 if r(max)>10000 & r(max)<=100000
    replace scale=1000000 if r(max)>100000 & r(max)<=1000000
gen estimate_sc=estimate/scale
```

Step 3

Rank the population from the most-disadvantaged subgroup (at rank 0) to the most-advantaged subgroup (at rank 1), based on the midpoint of its range in the cumulative population distribution.

- 3.1 Sort the data according to *subgroup_order* using the **sort** command. The subgroups should be sorted from the most-disadvantaged to the most-advantaged for the subsequent steps.
- 3.2 Use the **summarize (sum)** command to calculate the total population.
- 3.3 **Generate** a variable *popshare*, which is the share of each subgroup's population out of the total population (stored in **r(sum)** following the previous step).
- 3.4 Calculate the cumulative total of the population share of each subgroup, *cumshare*.
- 3.5 Calculate the midpoint of the population share of each subgroup (**0.5*popshare**) and subtract this from the cumulative total population share (*cumshare*). This produces the cumulative midpoint proportion of each subgroup, saved as *rank*.

```
sort subgroup_order
sum population
gen popshare=population/r(sum)
gen cumshare=sum(popshare)
gen rank=cumshare-0.5*popshare
```

Step 4

Regress the indicator of interest against the midpoint rank of each subgroup using an appropriate regression model, and then calculate predicted values of the indicator for the two extremes (rank 0 and rank 1).

Regress the indicator of interest against the midpoint rank of each subgroup using an appropriate regression model. A generalised linear model is used (**glm**) with the *estimate* as the independent (outcome) variable and *rank* as the dependent variable.

- Each subgroup's population weight is taken into account using probability weights, [**pw=population**].

- The **logit** link function is used to transform the outcome variable on a log scale and therefore bound the predicted values from the model between 0 and 1. This is more suitable for modelling indicators measured as percentages than using simple linear regression (which can theoretically produce predicted values less than 0 and greater than 1 since there is no lower and upper limit). A **binomial** distribution is used because the outcome variable is binary, due to the logit transformation.
- The option **robust** is used to calculate standard errors that are robust to potential issues such as varying levels of uncertainty in the data.
- The option **nolog** omits lines that are of no statistical interest from the output.

```
glm estimate_sc rank [pw=population], link(logit) family(binomial) robust
      nolog
```

Step 5

Calculate SII as the difference between the predicted values of the indicator at rank 0 and at rank 1.

- 5.1 Calculate the predicted values of the indicator for the two extremes (rank 0 and rank 1). Use the **margins** command to calculate the predicted indicator values from the previously fit regression model, at fixed values of the *rank* variable of 0 and 1. The **post** option is specified so that these predicted values are posted as estimation results.
- 5.2 The **nlcom** command calculates estimates and standard errors for nonlinear combinations of coefficients from a regression model. Calculate RII using **nlcom** using the predicted estimate at rank 1 (**_b[2._at]**) minus the predicted estimate at rank 0 (**_b[1._at]**), both produced from the previous **margins** command. The **post** option stores the outputs of the estimation into matrix **e()**.
- 5.3 **Generate** a new variable *sii* equal to the value at position [1,1] of matrix **r(b)**, which is the stored RII result from the previous step.

```
margins, at(rank=(0 1)) post
nlcom (SII: (_b[2._at] - _b[1._at])), post
gen sii=e(b) [1,1]
```

Step 6

Calculate and store the 95% confidence intervals of the SII.

- 6.1 Use the **matrix** command to store the variance-covariance matrix, which was saved in **e(V)** with the **post** command in Step 6, in a matrix called **V**.
- 6.2 The standard error of SII (*sii_se*) is the square root of the variance-covariance estimate (stored at position [1,1] of matrix **V**).
- 6.3 The lower 95% confidence interval (*sii_ll*) is calculated as the standard error of SII multiplied by 1.96, subtracted from SII.
- 6.4 The upper 95% confidence interval (*sii_ul*) is calculated as the standard error of SII multiplied by 1.96, added to SII.

```
matrix V=e(V)
gen sii_se=sqrt(V[1,1])
gen sii_ll=sii-1.96*sii_se
gen sii_ul=sii+1.96*sii_se
```

Step 7

Rescale the SII estimates to the same unit as the indicator.

Multiply the SII and the lower and upper confidence interval estimates by the scale of the indicator (*scale*). For instance, if the indicator is measured as a percentage, multiply by 100. If the indicator is measured as a rate per 1000 population, multiply by 1000.

```
replace sii=sii*scale
replace sii_ll=sii_ll*scale
replace sii_ul=sii_ul*scale
```

Calculating SII using individual-level data

Data requirements

Variable	Definition	Type
<i>id</i>	Individual record identifier	String or numeric
<i>psu</i>	Primary sampling unit (required if data come from a survey)	Numeric
<i>weight</i>	Individual sampling weight (required if data come from a survey)	Numeric
<i>strata</i>	Strata (required if data come from a survey)	Numeric
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. The dimension must be ordered, meaning that it has subgroups with an inherent ordering, and there must be more than two subgroups within the dimension of inequality. For example, within an economic status dimension, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5).	String
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring wealth-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>indicator</i>	Variable containing the indicator that is being measured. Can be categorized as a binary variable (with 0 meaning that the individual has not met the indicator and 1 meaning that the individual has met the indicator) or continuous (e.g., individual height or blood pressure).	Numeric

Calculation steps

Step 1 Declare survey design, if the data come from a survey.

The **svyset** command declares the survey design for the dataset. Depending on the nature of the survey, specify **svyset** options. For example, data from a Demographic and Health Survey (DHS) should specify primary sampling unit (*psu*), individual sampling weight (*weight*) and survey stratification (*strata*).

```
svyset psu [pweight=weight], strata(strata)
```

Step 2

Rank the population from the most-disadvantaged (at rank 0) to the most-advantaged (at rank 1), taking individual weights into account.

- 2.1 Use the **drop** command to drop any records where there is no indicator estimate.
- 2.2 **Sort** records according to the *subgroup_order* variable.
- 2.3 **Generate** a variable *sumw* which is the sum of all individual weights.
- 2.4 **Generate** a variable *cumw* which is the cumulative sum of each individual weight. The values of *cumw* are moved one row down in the variable *cumw_1*, with the value in the first row being zero.
- 2.5 Calculate the cumulative proportion of individuals within each subgroup. For each value of the *subgroup_order* variable, **generate** a variable *cumwr* containing the maximum value of the cumulative individual weight (*cumw*). Therefore, if individuals are being ranked by a categorical variable (such as wealth quintiles or education level) then there is a single *cumwr* value per subgroup. If individuals are being ranked by a continuous variable (such as wealth index scores) then each individual would have a *cumwr* value (apart from ties, when two individuals have the same ranking value).
- 2.6 Calculate the cumulative proportion of individuals within the previous subgroup. For each value of the *subgroup_order* variable, **generate** a variable *cumwr_1* containing the minimum value of the *cumw* variable.
- 2.7 Calculate the population share of individuals within each subgroup by subtracting *cumwr_1* from *cumwr*. Calculate the midpoint of this population share by multiplying this by 0.5. Add this to *cumwr_1* to get the cumulative midpoint of the population share. Divide by the sum of all individual weights (*sumw*) to produce the relative ranking of each individual (*rank*), which is the proportion of individuals within a given value of the ranking variable.

```
drop if indicator==.
sort subgroup_order
egen sumw=sum(weight)
gen cumw=sum(weight)
gen cumw_1=cumw[_n-1]
    replace cumw_1=0 if cumw_1==.
bysort subgroup_order: egen cumwr=max(cumw)
bysort subgroup_order: egen cumwr_1=min(cumw_1)
gen rank=(cumwr_1+0.5*(cumwr-cumwr_1))/sumw
```

Step 3

Regress the indicator of interest against the rank of each individual using a regression model.

Regress the indicator of interest against the midpoint rank of each subgroup using an appropriate regression model. A generalised linear model is used (**glm**) with the indicator estimate as the independent (outcome) variable and *rank* as the dependent variable.

- If the indicator is binary, a logistic regression model should be used by specifying the **logit** link and **binomial** distribution.

- If the indicator is continuous, a linear regression model should be used.
- If the data come from a survey, the **svy** prefix is used to take survey design (defined in Step 1) into account.
- If only individual weights need to be accounted for, use probability weights **[pw=weight]**.

```
*Binary indicator using svyset
svy: glm indicator rank, link(logit) family(binomial) nolog
*Binary indicator using individual weights
glm indicator rank [pw=weight], link(logit) family(binomial) nolog
*Continuous indicator using svyset
svy: glm indicator rank, nolog
*Continuous indicator using individual weights
svy: glm indicator rank [pw=weight], nolog
```

Step 4

Calculate SII as the difference between the predicted values of the indicator at rank 0 and at rank 1.

- 4.1 Use the **margins** command to calculate the predicted indicator values from the previously fit regression model, at fixed values of the *rank* variable of 0 and 1. The **post** option is specified so that these predicted values are posted as estimation results.
- 4.2 The **nlcom** command calculates estimates and standard errors for nonlinear combinations of coefficients from a regression model. Calculate RII using **nlcom** using the predicted estimate at rank 1 (**_b[2._at]**) minus the predicted estimate at rank 0 (**_b[1._at]**), both produced from the previous **margins** command. The **post** option stores the outputs of the estimation into matrix **e()**.
- 4.3 **Generate** a new variable *rii* equal to the value at position [1,1] of matrix **r(b)**, which is the stored RII result from the previous step.

```
margins, at(rank=(0 1)) post
nlcom (SII: (_b[2._at] - _b[1._at])), post
gen sii=e(b) [1,1]
```

Step 5

Calculate and store the 95% confidence intervals of the SII.

- 5.1 Use the **matrix** command to store the variance-covariance matrix (saved in **e(V)** with the **post** command in Step 5) in a matrix called *V*.
- 5.2 The standard error of SII (*sii_se*) is the square root of the variance-covariance estimate (stored at position [1,1] of matrix *V*).
- 5.3 The lower 95% confidence interval (*sii_ll*) is calculated as the standard error of SII multiplied by 1.96, subtracted from SII (if the standard error is not equal to zero).

- 5.4 The upper 95% confidence interval (*sii_ul*) is calculated as the standard error of SII multiplied by 1.96, added to SII (if the standard error is not equal to zero).

```
matrix V=e(V)
gen sii_se=sqrt(V[1,1])
gen sii_ll=sii-1.96*sii_se if sii_se!=0
gen sii_ul=sii+1.96*sii_se if sii_se!=0
```

Step 6

Rescale the SII estimates to the same unit as the indicator.

Multiply the SII and the lower and upper confidence interval estimates by the scale of the indicator. For instance, if the indicator is measured as a percentage, multiply by 100. If the indicator is measured as a rate per 1000 population, multiply by 1000.

Controlling for other factors at the individual-level

An advantage of calculating SII at the individual level is the ability to control for other factors and characteristics; for example, measuring wealth-related inequality while controlling for the influence of education and place of residence.

To control for other factors, these variables need to be present in the individual-level dataset. Insert these variables in the regression model in Step 3 above. For example, the model below controls for education (*educ*) and place of residence (*urban*). Steps 4-6 are the same as with univariate analysis.

```
svy: glm indicator rank educ urban, link(logit) family(binomial) nolog
```

Relative Index of Inequality

Overview

Type of summary measure: Complex; relative; weighted.

Applicability: Ordered; more than two subgroups

Definition: The relative index of inequality (RII) is a relative measure of inequality that represents the ratio of estimated indicator values between the most-advantaged and most-disadvantaged, while taking into consideration the situation in all other subgroups/individuals – using an appropriate regression model. RII can be calculated using disaggregated data and individual-level data. Subgroups in disaggregated data are weighted according to their population share, while individuals are weighted by sample weight in the case of data from surveys.

Interpretation: RII has the value of one if there is no inequality. RII has only positive values. Greater absolute values indicate higher levels of inequality. The further the value of RII from one, the higher the level of inequality. Values larger than 1 indicate the level of the indicator is higher among advantaged subgroups, and values lower than 1 indicate the level of the indicator is higher among disadvantaged subgroups. RII is a multiplicative measure and has to be displayed on a logarithmic scale (values larger than one are equivalent in magnitude to their reciprocal values smaller than one, e.g., a value of 2 is equivalent in magnitude to a value of 0.5).

Calculation: To calculate RII, a weighted sample of the whole population is ranked from the most-disadvantaged subgroup (at rank 0) to the most-advantaged subgroup (at rank 1). This ranking is weighted, accounting for the proportional distribution of the population within each subgroup. The relative rank of each subgroup is calculated as: $X_j = \sum_j p_j - 0.5p_j$.

The indicator of interest is then regressed against this relative rank using an appropriate regression model (e.g., a generalized linear model with logit link), and the predicted values of the indicator are calculated for the two extremes (rank 1 and rank 0). The ratio of the predicted values at rank 1 and rank 0 (covering the entire distribution) generates the RII value:

$$RII = \hat{v}_1 / \hat{v}_0$$

The variance of RII is calculated from the regression model using the delta method.

Calculating RII using disaggregated data

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education). The dimension must be ordered, meaning that it has subgroups with an inherent ordering. For example, a dimension of economic status with wealth quintiles that can be ranked from poorest to richest.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within an economic status dimension, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5).	String
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring economic-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>estimate</i>	The subgroup estimate. Estimates must be available for all subgroups.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric

Calculation steps

Step 1

Calculate RII for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Convert estimates to a 0-1 scale.

- 2.1 Use the **summarize** command to identify the maximum estimate, which stored in **r(max)** after running the command.

- 2.2 **Generate** a variable *scale*, which takes the value of 1 if the estimates are all less than 1, 100 if the estimates are between 1 and 100, and so on.
- 2.3 Divide the estimates by *scale* to convert estimates to be on a range between 0 and 1. Save this as *estimate_sc*.

```
summarize estimate
gen scale=1 if r(max)<=1
    replace scale=100 if r(max)>1 & r(max)<=100
    replace scale=1000 if r(max)>100 & r(max)<=1000
    replace scale=10000 if r(max)>1000 & r(max)<=10000
    replace scale=100000 if r(max)>10000 & r(max)<=100000
    replace scale=1000000 if r(max)>100000 & r(max)<=1000000
gen estimate_sc=estimate/scale
```

Step 3

Rank the population from the most-disadvantaged subgroup (at rank 0) to the most-advantaged subgroup (at rank 1), based on the midpoint of its range in the cumulative population distribution.

- 3.1 Sort the data according to *subgroup_order* using the **sort** command. The subgroups should be sorted from the most-disadvantaged to the most-advantaged for the subsequent steps.
- 3.2 Use the **summarize (sum)** command to calculate the total population.
- 3.3 **Generate** a variable *popshare*, which is the share of each subgroup's population out of the total population (stored in **r(sum)** following the previous step).
- 3.4 Calculate the cumulative total of the population share of each subgroup, *cumshare*.
- 3.5 Calculate the midpoint of the population share of each subgroup (**0.5*popshare**) and subtract this from the cumulative total population share (*cumshare*). This produces the cumulative midpoint proportion of each subgroup, saved as *rank*.

```
sort subgroup_order
sum population
gen popshare=population/r(sum)
gen cumshare=sum(popshare)
gen rank=cumshare-0.5*popshare
```

Step 4

Regress the indicator of interest against the midpoint rank of each subgroup using an appropriate regression model, and then calculate predicted values of the indicator for the two extremes (rank 0 and rank 1).

Regress the indicator of interest against the midpoint rank of each subgroup using an appropriate regression model. A generalised linear model is used (**glm**) with the *estimate* as the independent (outcome) variable and *rank* as the dependent variable.

- Each subgroup's population weight is taken into account using probability weights, [**pw=population**].

- The **logit** link function is used to transform the outcome variable on a log scale and therefore bound the predicted values from the model between 0 and 1. This is more suitable for modelling indicators measured as percentages than using simple linear regression (which can theoretically produce predicted values less than 0 and greater than 1 since there is no lower and upper limit). A **binomial** distribution is used because the outcome variable is binary, due to the logit transformation.
- The option **robust** is used to calculate standard errors that are robust to potential issues such as varying levels of uncertainty in the data.
- The option **nolog** omits lines that are of no statistical interest from the output.

```
glm estimate_sc rank [pw=population], link(logit) family(binomial) robust
nolog
```

Step 5

Calculate RII as the ratio between the predicted values of the indicator at rank 0 and at rank 1.

- 5.1 Calculate the predicted values of the indicator for the two extremes (rank 0 and rank 1). Use the **margins** command to calculate the predicted indicator values from the previously fit regression model, at fixed values of the *rank* variable of 0 and 1. The **post** option is specified so that these predicted values are posted as estimation results.
- 5.2 The **nlcom** command calculates estimates and standard errors for nonlinear combinations of coefficients from a regression model. Calculate RII using **nlcom** using the predicted estimate at rank 1 (**_b[2._at]**) divided by the predicted estimate at rank 0 (**_b[1._at]**), both produced from the previous **margins** command. The **post** option stores the outputs of the estimation into matrix **e()**.
- 5.3 **Generate** a new variable *rii* equal to the value at position [1,1] of matrix **r(b)**, which is the stored RII result from the previous step.

```
margins, at(rank=(0 1)) post
nlcom (RII: (_b[2._at] / _b[1._at])), post
gen rii=r(b) [1,1]
```

Step 6

Calculate and store the 95% confidence intervals of RII.

- 6.1 Note that the confidence intervals calculated by **nlcom** in the previous step are not correct because they are symmetric, while RII is a ratio measured on a logarithmic scale therefore confidence intervals should be non-symmetric. Re-run the **glm** regression model and use the **margins** command to calculate and store the predicted estimates at rank equal to 0 and 1.
- 6.2 Use the **nlcom** command to calculate a logarithmic version of RII. Name this **lnRII**.
- 6.3 Use the **lincom** command to exponentiate the results from the **nlcom** estimation of **lnRII**. The **eform** option reports estimates as exponentials.

6.4 The lower 95% confidence interval (*rii_ll*) is stored in **r(lb)** after running **lincom**.

6.5 The upper 95% confidence interval (*rii_ul*) is stored in **r(ub)** after running **lincom**.

```
glm estimate_sc rank [aw=population], link(logit) family(binomial) robust
    nolog
margins, at(rank=(0 1)) post
nlcom (lnRII: ln(_b[2._at] / _b[1._at])), post
lincom lnRII, eform
gen rii_ll=r(lb)
gen rii_ul=r(ub)
```

Calculating RII using individual-level data

Data requirements

Variable	Definition	Type
<i>id</i>	Individual record identifier	String or numeric
<i>psu</i>	Primary sampling unit (required if data come from a survey)	Numeric
<i>weight</i>	Individual sampling weight (required if data come from a survey)	Numeric
<i>strata</i>	Strata (required if data come from a survey)	Numeric
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. The dimension must be ordered, meaning that it has subgroups with an inherent ordering, and there must be more than two subgroups within the dimension of inequality. For example, within an economic status dimension, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5).	String
<i>subgroup_order</i>	The order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring wealth-related inequality using five wealth quintiles, quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.	Numeric (integer)
<i>indicator</i>	Variable containing the indicator that is being measured. Can be categorized as a binary variable (with 0 meaning that the individual has not met the indicator and 1 meaning that the individual has met the indicator) or continuous (e.g., individual height or blood pressure).	Numeric

Calculation steps

Step 1 Declare survey design, if the data come from a survey.

The **svyset** command declares the survey design for the dataset. Depending on the nature of the survey, specify **svyset** options. For example, data from a Demographic and Health Survey (DHS) should specify primary sampling unit (*psu*), individual sampling weight (*weight*) and survey stratification (*strata*).

```
svyset psu [pweight=weight], strata(strata)
```

Step 2

Rank the population from the most-disadvantaged (at rank 0) to the most-advantaged (at rank 1), taking individual weights into account.

- 2.1 Use the **drop** command to drop any records where there is no indicator estimate.
- 2.2 **Sort** records according to the *subgroup_order* variable.
- 2.3 **Generate** a variable *sumw* which is the sum of all individual weights.
- 2.4 **Generate** a variable *cumw* which is the cumulative sum of each individual weight. The values of *cumw* are moved one row down in the variable *cumw_1*, with the value in the first row being zero.
- 2.5 Calculate the cumulative proportion of individuals within each subgroup. For each value of the *subgroup_order* variable, **generate** a variable *cumwr* containing the maximum value of the cumulative individual weight (*cumw*). Therefore, if individuals are being ranked by a categorical variable (such as wealth quintiles or education level) then there is a single *cumwr* value per subgroup. If individuals are being ranked by a continuous variable (such as wealth index scores) then each individual would have a *cumwr* value (apart from ties, when two individuals have the same ranking value).
- 2.6 Calculate the cumulative proportion of individuals within the previous subgroup. For each value of the *subgroup_order* variable, **generate** a variable *cumwr_1* containing the minimum value of the *cumw* variable.
- 2.7 Calculate the population share of individuals within each subgroup by subtracting *cumwr_1* from *cumwr*. Calculate the midpoint of this population share by multiplying this by 0.5. Add this to *cumwr_1* to get the cumulative midpoint of the population share. Divide by the sum of all individual weights (*sumw*) to produce the relative ranking of each individual (*rank*), which is the proportion of individuals within a given value of the ranking variable.

```
drop if indicator==.
sort subgroup_order
egen sumw=sum(weight)
gen cumw=sum(weight)
gen cumw_1=cumw[_n-1]
    replace cumw_1=0 if cumw_1==.
bysort subgroup_order: egen cumwr=max(cumw)
bysort subgroup_order: egen cumwr_1=min(cumw_1)
gen rank=(cumwr_1+0.5*(cumwr-cumwr_1))/sumw
```

Step 3

Regress the indicator of interest against the rank of each individual using a regression model.

Regress the indicator of interest against the midpoint rank of each subgroup using an appropriate regression model. A generalised linear model is used (**glm**) with the indicator estimate as the independent (outcome) variable and *rank* as the dependent variable.

- If the indicator is binary, a logistic regression model should be used by specifying the **logit** link and **binomial** distribution.

- If the indicator is continuous, a linear regression model should be used.
- If the data come from a survey, the **svy** prefix is used to take survey design (defined in Step 1) into account.
- If only individual weights need to be accounted for, use probability weights **[pw=weight]**.

```
*Binary indicator using svyset
svy: glm indicator rank, link(logit) family(binomial) nolog
*Binary indicator using individual weights
glm indicator rank [pw=weight], link(logit) family(binomial) nolog
*Continuous indicator using svyset
svy: glm indicator rank, nolog
*Continuous indicator using individual weights
svy: glm indicator rank [pw=weight], nolog
```

Step 4

Calculate RII as the ratio between the predicted values of the indicator at rank 0 and at rank 1.

- 4.1 Calculate the predicted values of the indicator for the two extremes (rank 0 and rank 1). Use the **margins** command to calculate the predicted indicator values from the previously fit regression model, at fixed values of the *rank* variable of 0 and 1. The **post** option is specified so that these predicted values are posted as estimation results.
- 4.2 The **nlcom** command calculates estimates and standard errors for nonlinear combinations of coefficients from a regression model. Calculate RII using **nlcom** using the predicted estimate at rank 1 (**_b[2._at]**) divided by the predicted estimate at rank 0 (**_b[1._at]**), both produced from the previous **margins** command. The **post** option stores the outputs of the estimation into matrix **e()**.
- 4.3 **Generate** a new variable *rii* equal to the value at position [1,1] of matrix **r(b)**, which is the stored RII result from the previous step.

```
margins, at(rank=(0 1)) post
nlcom (RII: (_b[2._at] / _b[1._at])), post
gen rii=r(b) [1,1]
```

Step 5

Calculate and store the 95% confidence intervals of the RII.

- 5.1 Note that the confidence intervals calculated by **nlcom** in the previous step are not correct because they are symmetric, while RII is a ratio measured on a logarithmic scale so confidence intervals should be non-symmetric. Re-run the same **glm** regression model (from Step 3) and use the **margins** command to calculate and store the predicted estimates at rank equal to 0 and 1.

- 5.2 Use the **nlcom** command to calculate a logarithmic version of RII, using the natural log (**ln**) of the predicted estimate at rank 1 (**_b[2._at]**) divided by the predicted estimate at rank 0 (**_b[1._at]**). Name this **lnRII**.
- 5.3 Use the **lincom** command to exponentiate the results from the **nlcom** estimation of **lnRII**. The **eform** option reports estimates as exponentials.
- 5.4 The lower 95% confidence interval (**rii_ll**) is stored in **r(lb)** after running **lincom**.
- 5.5 The upper 95% confidence interval (**rii_ul**) is stored in **r(ub)** after running **lincom**.

```
svy: glm indicator rank, link(logit) family(binomial) nolog
      margins, at(rank=(0 1)) post
      nlcom (lnRII: ln(_b[2._at] / _b[1._at])), post
      lincom lnRII, eform
      gen rii_ll=r(lb)
      gen rii_ul=r(ub)
```

Controlling for other factors at the individual-level

An advantage of calculating RII at the individual level is the ability to control for other factors and characteristics; for example, measuring wealth-related inequality while controlling for the influence of education and place of residence.

To control for other factors, these variables need to be present in the individual-level dataset. Insert these variables in the regression model in Step 3 above. For example, the model below controls for education (*educ*) and place of residence (*urban*). Steps 4-5 are the same as with univariate analysis.

```
svy: glm indicator rank educ urban, link(logit) family(binomial) nolog
```

Between-Group Variance

Overview

Type of summary measure: Complex; absolute; weighted.

Applicability: Non-ordered; more than two subgroups

Definition: Between-Group Variance (BGV) is an absolute measure of inequality that considers all population subgroups. Subgroups are weighted according to their population share.

Interpretation: BGV has only positive values, with larger values indicating higher levels of inequality. BGV is zero if there is no inequality. BGV is more sensitive to outlier estimates as it gives more weight to the estimates that are further from the setting average. It is reported as the squared unit of the health indicator.

Calculation: BGV is calculated as the weighted average of squared differences between the subgroup estimates y_j and the setting average μ . Squared differences are weighted by each subgroup's population share p_j :

$$BGV = \sum_j p_j (y_j - \mu)^2$$

The variance of BGV is calculated as:

$$\begin{aligned} var(BGV) = & 4 \sum_j p_j^2 \sigma_j^2 (y_j - \sum_j p_j y_j)^2 \\ & + 2 \left[\left(\sum_j p_j^2 \sigma_j^2 \right)^2 - \left(\sum_j p_j^4 \sigma_j^4 \right) + \left(\sum_j p_j^2 (1 - p_j)^2 \sigma_j^4 \right) \right] \end{aligned}$$

where σ_j is the standard error of the estimate for the subgroup j .

Calculating BGV

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of BGV cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric

Calculation steps

Step 1

Calculate BGV for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Calculate the population share of each subgroup and the weighted mean of subgroup estimates.

- 2.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 2.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 2.3 Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)
```

Step 3

Calculate BGV as the weighted average of squared differences between the subgroup estimates and the weighted mean.

Subtract the weighted mean (*weighted_mean*) from each subgroup *estimate* and square these differences. This is then weighted by each subgroup's population share, *popshare*. Use the **total** command to sum the squared differences across all subgroups.

```
egen bgv=total(popshare*((estimate-weighted_mean)^2))
```

Step 4

Calculate the 95% confidence intervals of the BGV.

- 4.1 Calculate a variable *se2*, which is equal to the square of the population share (*popshare*) multiplied by the square of the standard error (*se*) multiplied by the square of the difference between the subgroup *estimate* and the *weighted_mean*. Use the **total** command to sum this across all subgroups.
- 4.2 Calculate a variable *s2*, which is equal to the square of the population share (*popshare*) multiplied by the square of the standard error (*se*). Use the **total** command to sum this across all subgroups.
- 4.3 Calculate a variable *s4*, which is equal to the population share (*popshare*) to the power of four multiplied by the standard error (*se*) to the power of four. Use the **total** command to sum this across all subgroups.
- 4.4 Calculate a variable *se4*, which is equal to the square of the population share (*popshare*) multiplied by the square of one minus the population share, multiplied by the standard error (*se*) to the power of four. Use the **total** command to sum this across all subgroups.
- 4.5 The standard error of BGV (*bgv_se*) is the square root of the variance, which is calculated as *sum_se2* multiplied by four, plus two times the square of *sum_s2* minus *sum_s4* plus *sum_se4*.
- 4.6 The lower 95% confidence interval (*bgv_ll*) is calculated as the standard error of BGV multiplied by 1.96, subtracted from BGV.
- 4.7 The upper 95% confidence interval (*bgv_ul*) is calculated as the standard error of BGV multiplied by 1.96, added to the BGV.

```
egen se2=total((popshare^2)*(se^2)*((estimate-weighted_mean)^2))
egen s2=total((popshare^2)*(se^2))
egen s4=total((popshare^4)*(se^4))
egen se4=total((popshare^2)*((1-popshare)^2)*(se^4))
gen bgv_se=sqrt(4*se2+2*((s2^2)-s4+se4))
gen bgv_ll=bgv-1.96*bgv_se
```

```
gen bgv_ul=bgv+1.96*bgv_se
```

Between-Group Standard Deviation

Overview

Type of summary measure: Complex; absolute; weighted.

Applicability: Non-ordered; more than two subgroups

Definition: Between-Group Standard Deviation (BGSD) is an absolute measure of inequality that considers all population subgroups. Subgroups are weighted according to their population share.

Interpretation: BGSD has only positive values, with larger values indicating higher levels of inequality. BGSD is zero if there is no inequality. It has the same unit as the health indicator.

Calculation: BGSD is calculated as the square root of the weighted average of squared differences between the subgroup estimates y_j and the setting average μ . Squared differences are weighted by each subgroup's population share p_j :

$$BGSD = \sqrt{\sum_j p_j (y_j - \mu)^2}$$

95% confidence intervals can be calculated using a methodology of simulated estimates. The dataset is simulated a large number of times (e.g., 100) and BGSD is calculated for each of the simulated samples. The 95% confidence intervals are based on the 2.5th and 97.5th centiles of the BGSD results.

Calculating BGSD

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of BGSD cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric
<i>indicator_scale</i>	The scale of the indicator. For example, the scale of an indicator measured as a percentage is 100. The scale of an indicator measured as a rate per 1000 population is 1000. If this is missing, 95% confidence intervals of BGSD cannot be calculated.	Numeric

Calculation steps

Step 1

Calculate BGSD for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Calculate the population share of each subgroup and the weighted mean of subgroup estimates.

- 2.1 Use the **egen** command to calculate the total population, *sum_pop*.

- 2.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 2.3 Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)
```

Step 3

Calculate BGSD as the square root of the weighted average of squared differences between the subgroup estimates and the weighted mean.

- 3.1 Subtract the weighted mean (*weighted_mean*) from each subgroup *estimate* and square these differences. This is then weighted by each subgroup's population share, *popshare*. Use the **total** command to sum the squared differences across all subgroups.
- 3.2 Calculate BGSD as the square root of the result.

```
egen bgds_prep=total(popshare*((estimate-weighted_mean)^2))
egen bgds=sqrt(bgds_prep)
```

Step 4

Calculate the 95% confidence intervals of BGSD using a methodology of simulated estimates.

- 4.1 The dataset will be simulated 100 times and BGSD calculated for each simulation. Use the **forvalues** command to execute the subsequent code contained within braces {} 100 times (represented by 1/100). A local macro, *j*, is set to each simulation; for instance, in the first simulation, *j* is equal to 1.
- 4.2 For each simulated dataset an empty variable is generated, *est_`j'*. For example, for the first simulation, this variable is called *est_1*.
- 4.3 Each estimate will be simulated one at a time. Use the **forvalues** command again to execute the subsequent code contained within the braces {} for each of the rows (or subgroups) in the dataset, starting from 1 to the maximum number of rows (**_N**). A local macro, *n*, is set to identify each row; for instance, for the first row, *n* is equal to 1.
- 4.4 Use the **tempname** command to store an empty local macro, with the name *i`n'*. *`n'* will be replaced by each row number; for instance, for the first row, the local macro will be named *i1*. This will be used to create a temporary scalar variable, set to be blank (=.) to start.

```
forvalues j = 1/100 {
    gen est_`j'=.
    forvalues n = 1/`= _N' {
        tempname i`n'
        gen `i`n'`=.
```

- 4.5 If the indicator is measured as a percentage (*indicator_scale=100*), then the simulated estimates must fall between the values of 0 and 100. To ensure this happens, use the **while** command, which evaluates an expression and executes the code contained within braces until the expression is false. The expression being evaluated is whether the temporary scalar variable *i`n'* is blank.
- Generate a variable simulation containing simulated estimates using the **rnormal** command, which generates normal random samples that have a mean and standard error equal to that in the original dataset.
 - Use the **summarize** command to calculate summary statistics for simulated estimate of the row in question.
 - If the simulated estimate of the row in question, captured using the stored **r(mean)** value from the previous step, is greater than or equal to 0 and less than or equal to 100, then replace the temporary scalar variable *i`n'* with the simulated estimate.
 - **Drop** the simulated estimates, and repeat the process until all rows have a simulated estimate.
- 4.6 If the indicator is not measured as a percentage (*indicator_scale!=100*), then the simulated estimates must be greater than or equal to 0. Repeat the previous steps, with the only difference being that the stored **r(mean)** value must be greater than or equal to 0.

```
*If indicator is a percentage, simulate estimates between 0 and 100
if indicator_scale==100 {
    while `i`n'==. {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 & r(mean)<=100 {
            replace `i`n'=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'' if _n==`n'
}

*If indicator is not a percentage, simulate estimates greater than 0
if indicator_scale!=100 {
    while `i`n'==. {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 {
            replace `i`n'=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'' if _n==`n'
}
```

- 4.7 Calculate BGSD using the simulated estimates and the method outlined in Step 3. The weighted mean must be recalculated using the simulated estimates (*est_`j'*). The simulated estimates (*est_`j'*) and the new weighted mean (*weighted_mean_`j'*) are used in the BGSD formula.

***Calculate BGSD using the simulated estimates**

```
egen weighted_mean_`j`'=total(popshare*est_`j`')
egen bgzd_prep_`j`'=total(popshare*((est_`j`'-weighted_mean_`j`')^2))
gen bgzd_`j`'=sqrt(bgzd_prep_`j`')
drop weighted_mean_`j`' bgzd_prep_`j`'
```

4.8 Calculate the 95% confidence intervals using the BGSD values that were computed using the simulated data.

- Use the **preserve** command to preserve the dataset, so that the original data can be restored later.
- **Keep** only the values in the first row (row=1).
- **Keep** only the BGSD values that were calculated using the simulated data.
- **Generate** a dummy *count* variable, for use in the subsequent step.
- Use the **reshape** command to reshape the data from wide to long format. Now the BGSD values are in a column.
- Use the **centile** command to estimate specified centiles for the BGSD values, namely the 2.5th and the 97.5th centiles, which are equivalent to the lower and upper 95% confidence intervals.
- Use the **matrix** command to store the 2.5th centile value, saved in **r(c_1)**, in a matrix called *bgzd_ll*.
- Use the **matrix** command to store the 97.5th centile value, saved in **r(c_2)**, in a matrix called *bgzd_ul*.
- Use the **restore** command to restore the dataset to the original.
- Use the **drop** command to drop all of the *est_* and *bgzd_* values that were created from the 100 simulations, as these are no longer needed.
- The lower 95% confidence interval (*bgzd_ll*) is equal to the value stored at position [1,1] of matrix *bgzd_ll*.
- The upper 95% confidence interval (*bgzd_ul*) is equal to the value stored at position [1,1] of matrix *bgzd_ul*.

***Calculate the 95% confidence intervals**

```
preserve
  keep in 1
  keep bgzd_*
  gen count=1
  reshape long bgzd_, i(count) j(simulation)
  centile bgzd_, c(2.5 97.5)
  matrix bgzd_ll=(r(c_1))
  matrix bgzd_ul=(r(c_2))
restore
drop est_* bgzd_*
gen bgzd_ll=bgzd_ll[1,1]
gen bgzd_ul=bgzd_ul[1,1]
```

Coefficient of Variation

Overview

Type of summary measure: Complex; relative; weighted.

Applicability: Non-ordered; more than two subgroups

Definition: The Coefficient of Variation (COV) is a relative measure of inequality that considers all population subgroups. Subgroups are weighted according to their population share.

Interpretation: COV has only positive values, with larger values indicating higher levels of inequality. COV is zero if there is no inequality.

Calculation: COV is calculated by dividing the between-group standard deviation (BGSD) by the setting average μ and multiplying the fraction by 100.

BGSD is calculated as the square root of the weighted average of squared differences between the subgroup estimates y_j and the setting average μ . Squared differences are weighted by each subgroup's population share p_j :

$$BGSD = \sqrt{\sum_j p_j (y_j - \mu)^2}$$

Therefore, COV is calculated as:

$$COV = \frac{BGSD}{\mu} * 100$$

95% confidence intervals can be calculated using a methodology of simulated estimates. The dataset is simulated a large number of times (e.g., 100) and COV is calculated for each of the simulated samples. The 95% confidence intervals are based on the 2.5th and 97.5th centiles of the COV results.

Calculating COV

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of COV cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric
<i>indicator_scale</i>	The scale of the indicator. For example, the scale of an indicator measured as a percentage is 100. The scale of an indicator measured as a rate per 1000 population is 1000. If this is missing, 95% confidence intervals of COV cannot be calculated.	Numeric

Calculation steps

Step 1

Calculate COV for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Calculate the population share of each subgroup and the weighted mean of subgroup estimates.

- 2.1 Use the **egen** command to calculate the total population, *sum_pop*.

- 2.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 2.3 Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)
```

Step 3

Calculate COV as the between-group standard deviation (BGSD) divided by the weighted mean.

- 3.1 Subtract the weighted mean (*weighted_mean*) from each subgroup *estimate* and square these differences. This is then weighted by each subgroup's population share, *popshare*. Use the **total** command to sum the squared differences across all subgroups.
- 3.2 Calculate BGSD as the square root of the result.
- 3.3 Calculate COV as BGSD (*bgsd*) divided by the weighted mean (*weighted_mean*), multiplied by 100.

```
egen bgsd_prep=total(popshare*((estimate-weighted_mean)^2))
egen bgsd=sqrt(bgsd_prep)
gen cov=(bgsd/weighted_mean)*100
```

Step 4

Calculate the 95% confidence intervals of COV using a methodology of simulated estimates.

- 4.1 The dataset will be simulated 100 times and COV calculated for each simulation. Use the **forvalues** command to execute the subsequent code contained within braces {} 100 times (represented by 1/100). A local macro, *j*, is set to each simulation; for instance, in the first simulation, *j* is equal to 1.
- 4.2 For each simulated dataset an empty variable is generated, *est_`j'*. For example, for the first simulation, this variable is called *est_1*.
- 4.3 Each estimate will be simulated one at a time. Use the **forvalues** command again to execute the subsequent code contained within the braces {} for each of the rows (or subgroups) in the dataset, starting from 1 to the maximum number of rows (**_N**). A local macro, *n*, is set to identify each row; for instance, for the first row, *n* is equal to 1.
- 4.4 Use the **tempname** command to store an empty local macro, with the name *i`n'*. *`n'* will be replaced by each row number; for instance, for the first row, the local macro will be named *i1*. This will be used to create a temporary scalar variable, set to be blank (=.) to start.

```
forvalues j = 1/100 {
    gen est_`j'=.
    forvalues n = 1/`= _N' {
        tempname i`n'
```

```
gen `i`n'=. .
```

- 4.5 If the indicator is measured as a percentage (***indicator_scale=100***), then the simulated estimates must fall between the values of 0 and 100. To ensure this happens, use the **while** command, which evaluates an expression and executes the code contained within braces until the expression is false. The expression being evaluated is whether the temporary scalar variable *i`n'* is blank.
- Generate a variable simulation containing simulated estimates using the **rnormal** command, which generates normal random samples that have a mean and standard error equal to that in the original dataset.
 - Use the **summarize** command to calculate summary statistics for simulated estimate of the row in question.
 - If the simulated estimate of the row in question, captured using the stored **r(mean)** value from the previous step, is greater than or equal to 0 and less than or equal to 100, then replace the temporary scalar variable *i`n'* with the simulated estimate.
 - **Drop** the simulated estimates, and repeat the process until all rows have a simulated estimate.
- 4.6 If the indicator is not measured as a percentage (***indicator_scale!=100***), then the simulated estimates must be greater than or equal to 0. Repeat the previous steps, with the only difference being that the stored **r(mean)** value must be greater than or equal to 0.

```
*If indicator is a percentage, simulate estimates between 0 and 100
if indicator_scale==100 {
    while `i`n'==. {
        gen simulation=rnormal(est,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 & r(mean)<=100 {
            replace `i`n'=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'' if _n==`n'
}

*If indicator is not a percentage, simulate estimates greater than 0
if indicator_scale!=100 {
    while `i`n'==. {
        gen simulation=rnormal(est,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 {
            replace `i`n'=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'' if _n==`n'
}
```

- 4.7 Calculate COV using the simulated estimates and the method outlined in Step 3. The weighted mean must be recalculated using the simulated estimates (*est_`j'*). The simulated estimates (*est_`j'*) and the new weighted mean (*weighted_mean_`j'*) are used in the COV formula.

***Calculate COV using the simulated estimates**

```
egen weighted_mean_`j'=total(popshare*est_`j')
egen bgstd_prep_`j'=total(popshare*((est_`j'-weighted_mean_`j')^2))
gen bgstd_`j'=sqrt(bgstd_prep_`j')
gen cov_`j'=(bgstd_`j'/weighted_mean_`j')*100
drop weighted_mean_`j' bgstd_prep_`j' bgstd_`j'
```

- 4.8 Calculate the 95% confidence intervals using the COV values that were computed using the simulated data.

- Use the **preserve** command to preserve the dataset, so that the original data can be restored later.
- **Keep** only the values in the first row (row=1).
- **Keep** only the COV values that were calculated using the simulated data.
- **Generate** a dummy *count* variable, for use in the subsequent step.
- Use the **reshape** command to reshape the data from wide to long format. Now the BGSD values are in a column.
- Use the **centile** command to estimate specified centiles for the COV values, namely the 2.5th and the 97.5th centiles, which are equivalent to the lower and upper 95% confidence intervals.
- Use the **matrix** command to store the 2.5th centile value, saved in **r(c_1)**, in a matrix called cov_ll.
- Use the **matrix** command to store the 97.5th centile value, saved in **r(c_2)**, in a matrix called cov_ul.
- Use the **restore** command to restore the dataset to the original.
- Use the **drop** command to drop all of the *est_* and *cov_* values that were created from the 100 simulations, as these are no longer needed.
- The lower 95% confidence interval (*cov_ll*) is equal to the value stored at position [1,1] of matrix cov_ll.
- The upper 95% confidence interval (*cov_ul*) is equal to the value stored at position [1,1] of matrix cov_ul.

***Calculate the 95% confidence intervals**

```
preserve
    keep in 1
    keep cov_*
    gen count=1
    reshape long cov_, i(count) j(simulation)
    centile cov_, c(2.5 97.5)
    matrix cov_ll=(r(c_1))
```



```
matrix cov_ul=(r(c_2))  
restore  
drop est_* cov_*  
gen cov_ll=cov_ll[1,1]  
gen cov_ul=cov_ul[1,1]
```

Mean Difference from Mean

Overview

Type of summary measure: Complex; absolute; weighted or non-weighted.

Applicability: Non-ordered; more than two subgroups.

Definition: The Mean Difference from Mean (MDM) is an absolute measure of inequality that shows the mean difference between each subgroup and the setting average. MDM can be calculated as an unweighted or weighted measure. For the unweighted version, all subgroups are weighted equally. For the weighted version, subgroups are weighted according to their population share.

Interpretation: MDM only has positive values, with larger values indicating higher levels of inequality. MDM is zero if there is no inequality.

Calculation: The unweighted version (MDMU) is calculated as the sum of the absolute differences between the subgroup estimates y_j and the setting average μ , divided by the number of subgroups n :

$$MDMU = \frac{1}{n} * \sum_j |y_j - \mu|$$

The weighted version (MDMW) is calculated as the weighted average of absolute differences between the subgroup estimates y_j and the setting average μ . Absolute differences are weighted by each subgroup's population share p_j :

$$MDMW = \sum_j p_j |y_j - \mu|$$

95% confidence intervals can be calculated using a methodology of simulated estimates. The dataset is simulated a large number of times (e.g., 100) and MDMU/MDMW is calculated for each of the simulated samples. The 95% confidence intervals are based on the 2.5th and 97.5th centiles of the MDMU/MDMW results.

Calculating MDM

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of MDMU/MDMW cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric
<i>indicator_scale</i>	The scale of the indicator. For example, the scale of an indicator measured as a percentage is 100. The scale of an indicator measured as a rate per 1000 population is 1000. If this is missing, 95% confidence intervals of MDMU/MDMW cannot be calculated.	Numeric
<i>setting_average</i>	The indicator average for the setting (e.g., national average). Optional for the calculation of MDMU if <i>population</i> data are not available.	Numeric

Calculation steps

Step 1

Calculate MDM for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Calculate the population share of each subgroup and the weighted mean of subgroup estimates. *Note: alternatively, use the setting average instead of calculating the weighted mean.*

- 2.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 2.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 2.3 Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)
```

Step 3

Calculate unweighted MDM (MDMU):

The average of absolute differences between the subgroup estimates and the weighted mean, divided by the number of subgroups. *Note: alternatively, use the setting average instead of the weighted mean.*

- 3.1 Calculate the absolute difference between each subgroup *estimate* and the weighted mean (*weighted_mean*). Use the **total** command to sum this across all subgroups. Save the result as *mdmu_prep*.
- 3.2 Divide this by the number of subgroups (**_N**) to get the unweighted MDM.
- 3.3 Drop the *mdmu_prep* variable, as it is no longer needed.

```
egen mdmu_prep=total(abs(estimate-weighted_mean))
gen mdmu=mdmu_prep/_N
drop mdmu_prep
```

Step 4

Calculate weighted MDM (MDMW):

The weighted average of absolute differences between the subgroup estimates and the weighted mean.

Calculate the absolute differences between each subgroup *estimate* and the weighted mean (*weighted_mean*), multiplied by the population share of each subgroup (*popshare*). Use the **total** command to sum this all subgroups to get the weighted MDM.

```
egen mdmw=total(popshare*abs(estimate-weighted_mean))
```

Step 5

Calculate the 95% confidence intervals of MDMU and MDMW using a methodology of simulated estimates.

- 5.1 The dataset will be simulated 100 times and MDM calculated for each simulation. Use the **forvalues** command to execute the subsequent code contained within braces {} 100 times (represented by 1/100). A local macro, *j*, is set to each simulation; for instance, in the first simulation, *j* is equal to 1.
- 5.2 For each simulated dataset an empty variable is generated, *est_`j'*. For example, for the first simulation, this variable is called *est_1*.
- 5.3 Each estimate will be simulated one at a time. Use the **forvalues** command again to execute the subsequent code contained within the braces {} for each of the rows (or subgroups) in the dataset, starting from 1 to the maximum number of rows (**_N**). A local macro, *n*, is set to identify each row; for instance, for the first row, *n* is equal to 1.
- 5.4 Use the **tempname** command to store an empty local macro, with the name *i`n'*. *`n'* will be replaced by each row number; for instance, for the first row, the local macro will be named *i1*. This will be used to create a temporary scalar variable, set to be blank (=.) to start.

```
forvalues j = 1/100 {
    gen est_`j'=.
    forvalues n = 1/`= _N' {
        tempname i`n'
        gen `i`n'`=.
```

- 5.5 If the indicator is measured as a percentage (**indicator_scale=100**), then the simulated estimates must fall between the values of 0 and 100. To ensure this happens, use the **while** command, which evaluates an expression and executes the code contained within braces until the expression is false. The expression being evaluated is whether the temporary scalar variable *i`n'* is blank.
 - Generate a variable simulation containing simulated estimates using the **rnormal** command, which generates normal random samples that have a mean and standard error equal to that in the original dataset.
 - Use the **summarize** command to calculate summary statistics for simulated estimate of the row in question.
 - If the simulated estimate of the row in question, captured using the stored **r(mean)** value from the previous step, is greater than or equal to 0 and less than or equal to 100, then replace the temporary scalar variable *i`n'* with the simulated estimate.
 - **Drop** the simulated estimates, and repeat the process until all rows have a simulated estimate.
- 5.6 If the indicator is not measured as a percentage (**indicator_scale!=100**), then the simulated estimates must be greater than or equal to 0. Repeat the previous steps, with the only difference being that the stored **r(mean)** value must be greater than or equal to 0.

```
*If indicator is a percentage, simulate estimates between 0 and 100
if indicator_scale==100 {
```

```

while `i`n'==. {
    gen simulation=rnormal(estimate,se)
    qui summarize simulation if _n==`n'
    if r(mean)>=0 & r(mean)<=100 {
        replace `i`n'=r(mean)
    }
    drop simulation
}
replace est_`j'=`i`n'' if _n==`n'
}

*If indicator is not a percentage, simulate estimates greater than 0
if indicator_scale==100 {
    while `i`n'==. {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 {
            replace `i`n'=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'' if _n==`n'
}

```

- 5.7 Calculate MDMU and MDMW using the simulated estimates and the method outlined in Steps 3 and 4. The weighted mean must be recalculated using the simulated estimates (*est_`j'*). The simulated estimates (*est_`j'*) and the new weighted mean (*weighted_mean_`j'*) are used in the MDMU and MDMW formulas.

```

*Calculate MDMU using the simulated estimates
egen weighted_mean_`j'=total(popshare*est_`j')
egen mdmu_prep=total(abs(est_`j'-weighted_mean_`j'))
gen mdmu_`j'=mdmu_prep/_N
drop weighted_mean_`j' mdmu_prep

*Calculate MDMW using the simulated estimates
egen weighted_mean_`j'=total(popshare*est_`j')
egen mdmw_`j'=total(popshare*abs(est_`j'-weighted_mean_`j'))
drop weighted_mean_`j'

```

- 5.8 Calculate the 95% confidence intervals using the MDMU and MDMW values that were computed using the simulated data.
- Use the **preserve** command to preserve the dataset, so that the original data can be restored later.
 - **Keep** only the values in the first row (row=1).
 - **Keep** only the MDMU and MDMW values that were calculated using the simulated data.
 - **Generate** a dummy *count* variable, for use in the subsequent step.
 - Use the **reshape** command to reshape the data from wide to long format. Now the MDMU and MDMW values are in separate columns.

- Use the **centile** command to estimate specified centiles for the MDMU values, namely the 2.5th and the 97.5th centiles, which are equivalent to the lower and upper 95% confidence intervals.
- Use the **matrix** command to store the 2.5th centile value, saved in **r(c_1)**, in a matrix called **mdmu_ll**.
- Use the **matrix** command to store the 97.5th centile value, saved in **r(c_2)**, in a matrix called **mdmu_ul**.
- Repeat to estimate and store centile values for MDMW.
- Use the **restore** command to restore the dataset to the original.
- Use the **drop** command to drop all of the *est_*, *mdmu_* and *mdmw_* values that were created from the 100 simulations, as these are no longer needed.
- The lower 95% confidence intervals (*mdmu_ll* and *mdmw_ll*) are equal to the value stored at position [1,1] of matrices *mdmu_ll* and *mdmw_ll*, respectively.
- The upper 95% confidence intervals (*mdmu_ul* and *mdmw_ul*) are equal to the value stored at position [1,1] of matrices *mdmu_ul* and *mdmw_ul*, respectively.

***Calculate the 95% confidence intervals**

```

preserve
    keep in 1
    keep mdmu_* mdmw_*
    gen count=1
    reshape long mdmu_ mdmw_, i(count) j(simulation)
    centile mdmu_, c(2.5 97.5)
    matrix mdmu_ll=(r(c_1))
    matrix mdmu_ul=(r(c_2))
    centile mdmw_, c(2.5 97.5)
    matrix mdmw_ll=(r(c_1))
    matrix mdmw_ul=(r(c_2))
restore
drop est_* mdmu_* mdmw_*
gen mdmu_ll=mdmu_ll[1,1]
gen mdmu_ul=mdmu_ul[1,1]
gen mdmw_ll=mdmw_ll[1,1]
gen mdmw_ul=mdmw_ul[1,1]

```

Mean Difference from the Best-Performing Subgroup

Overview

Type of summary measure: Complex; absolute; weighted or non-weighted.

Applicability: Non-ordered; more than two subgroups.

Definition: The Mean Difference from the Best-Performing Subgroup (MDB) is an absolute measure of inequality that shows the mean difference between each population subgroup and the subgroup with the best estimate. MDB can be calculated as an unweighted or weighted measure. For the unweighted version, all subgroups are weighted equally. For the weighted version, subgroups are weighted according to their population share.

Interpretation: MDB only has positive values, with larger values indicating higher levels of inequality. MDB is zero if there is no inequality.

Calculation: The unweighted version (MDBU) is calculated as the average of absolute differences between the subgroup estimates y_j and the estimate for the best-performing subgroup y_{ref} , divided by the number of subgroups n :

$$MDBU = \frac{1}{n} * \sum_j |y_j - y_{ref}|$$

The weighted version (MDBW) is calculated as the weighted average of absolute differences between the subgroup estimates y_j and the estimate for the best-performing subgroup y_{ref} . Absolute differences are weighted by each subgroup's population share p_j :

$$MDBW = \sum_j p_j |y_j - y_{ref}|$$

95% confidence intervals can be calculated using a methodology of simulated estimates. The dataset is simulated a large number of times (e.g., 100) and MDBU/MDBW is calculated for each of the simulated samples. The 95% confidence intervals are based on the 2.5th and 97.5th centiles of the MDBU/MDBW results.

Calculating MDB

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of MDBU/MDBW cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups for the calculation of MDBW (not required for MDBU).	Numeric
<i>indicator_scale</i>	The scale of the indicator. For example, the scale of an indicator measured as a percentage is 100. The scale of an indicator measured as a rate per 1000 population is 1000. If this is missing, 95% confidence intervals of MDBU/MDBW cannot be calculated.	Numeric
<i>favourable_indicator</i>	1 for favourable indicators (which measure desirable health events where the ultimate goal is to achieve a maximum level, such as skilled birth attendance) and 0 for non-favorable indicators (which measure undesirable health events where the ultimate goal is to achieve a minimum level, such as under-five mortality rate).	Numeric (0 or 1)

Calculation steps

Step 1

Calculate MDB for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2 Identify the best-performing subgroup estimate.

- 2.1 Use the **summarize** command to identify the maximum and minimum subgroup estimates, which get saved in **r(max)** and **r(min)** respectively.
- 2.2 **Generate** a variable *ref_estimate* that is equal to the maximum estimate value (**r(max)**) if the indicator is favourable (*favourable_indicator*==1).
- 2.3 **Replace** *ref_estimate* to be equal to the maximum estimate value (**r(max)**) if the indicator is adverse (*favourable_indicator*==0).

```
summarize estimate
gen ref_estimate=r(max) if favourable_indicator==1
replace ref_estimate=r(min) if favourable_indicator==0
```

Step 3 Calculate unweighted MDB (MDBU):

The average of absolute differences between the subgroup estimates and the best-performing subgroup estimate, divided by the number of subgroups.

- 3.1 Calculate the absolute difference between each subgroup *estimate* and the best subgroup estimate (*ref_estimate*). Use the **total** command to sum this across all subgroups.
- 3.2 Divide this by the number of subgroups (**_N**) to get the unweighted MDB.
- 3.3 Drop the *mdbu_prep* variable, as it is no longer needed.

```
egen mdbu_prep=total(abs(estimate-ref_estimate))
gen mdbu=mdbu_prep/_N
drop mdbu_prep
```

Step 4 Calculate weighted MDB (MDBW):

The weighted average of absolute differences between the subgroup estimates and the best-performing subgroup estimate.

- 4.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 4.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 4.3 Calculate the absolute differences between each subgroup *estimate* and the best subgroup estimate (*ref_estimate*), multiplied by the population share of each subgroup (*popshare*). Use the **total** command to sum this all subgroups to get the weighted MDB.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen mdbw=total(popshare*abs(estimate-ref_estimate))
```

Step 5

Calculate the 95% confidence intervals of MDBU and MDBW using a methodology of simulated estimates.

- 5.1 The dataset will be simulated 100 times and MDB calculated for each simulation. Use the **forvalues** command to execute the subsequent code contained within braces {} 100 times (represented by 1/100). A local macro, *j*, is set to each simulation; for instance, in the first simulation, *j* is equal to 1.
- 5.2 For each simulated dataset an empty variable is generated, *est_`j'*. For example, for the first simulation, this variable is called *est_1*.
- 5.3 Each estimate will be simulated one at a time. Use the **forvalues** command again to execute the subsequent code contained within the braces {} for each of the rows (or subgroups) in the dataset, starting from 1 to the maximum number of rows (*_N*). A local macro, *n*, is set to identify each row; for instance, for the first row, *n* is equal to 1.
- 5.4 Use the **tempname** command to store an empty local macro, with the name *i`n'*. *`n'* will be replaced by each row number; for instance, for the first row, the local macro will be named *i1*. This will be used to create a temporary scalar variable, set to be blank (=.) to start.

```
forvalues j = 1/100 {
  gen est_`j'=.
  forvalues n = 1/`= _N' {
    tempname i`n'
    gen `i`n'`=.
```

- 5.5 If the indicator is measured as a percentage (***indicator_scale=100***), then the simulated estimates must fall between the values of 0 and 100. To ensure this happens, use the **while** command, which evaluates an expression and executes the code contained within braces until the expression is false. The expression being evaluated is whether the temporary scalar variable *i`n'* is blank.
 - Generate a variable simulation containing simulated estimates using the **rnormal** command, which generates normal random samples that have a mean and standard error equal to that in the original dataset.
 - Use the **summarize** command to calculate summary statistics for simulated estimate of the row in question.
 - If the simulated estimate of the row in question, captured using the stored **r(mean)** value from the previous step, is greater than or equal to 0 and less than or equal to 100, then replace the temporary scalar variable *i`n'* with the simulated estimate.
 - **Drop** the simulated estimates, and repeat the process until all rows have a simulated estimate.
- 5.6 If the indicator is not measured as a percentage (***indicator_scale!=100***), then the simulated estimates must be greater than or equal to 0. Repeat the previous steps, with the only difference being that the stored **r(mean)** value must be greater than or equal to 0.

```

*If indicator is a percentage, simulate estimates between 0 and 100
if indicator_scale==100 {
    while `i`n'==. {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 & r(mean)<=100 {
            replace `i`n'`=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'`' if _n==`n'
}

*If indicator is not a percentage, simulate estimates greater than 0
if indicator_scale==100 {
    while `i`n'==. {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 {
            replace `i`n'`=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'`' if _n==`n'
}

```

- 5.7 Calculate MDBU and MDBW using the simulated estimates and the method outlined in Steps 3 and 4. The best-performing subgroup estimate must be recalculated using the simulated estimates (*ref_estimate_`j'*). The simulated estimates (*est_`j'*) and the new reference estimate are used in the MDBU and MDBW formulas.

```

*Calculate MDBU using the simulated estimates
summarize est_`j'
gen ref_estimate_`j'=r(max) if favourable_indicator==1
replace ref_estimate_`j'=r(min) if favourable_indicator==0
egen mdbu_prep=total(abs(est_`j'-ref_estimate_`j'))
gen mdbu_`j'=mdbu_prep/_N
drop ref_estimate_`j' mdbu_prep

*Calculate MDBW using the simulated estimates
summarize est_`j'
gen ref_estimate_`j'=r(max) if favourable_indicator==1
replace ref_estimate_`j'=r(min) if favourable_indicator==0
egen mdbw_`j'=total(popshare*abs(est_`j'-ref_estimate_`j'))
drop ref_estimate_`j'

```

- 5.8 Calculate the 95% confidence intervals using the MDBU and MDBW values that were computed using the simulated data.
- Use the **preserve** command to preserve the dataset, so that the original data can be restored later.
 - **Keep** only the values in the first row (row=1).

- **Keep** only the MDBU and MDBW values that were calculated using the simulated data.
- **Generate** a dummy *count* variable, for use in the subsequent step.
- Use the **reshape** command to reshape the data from wide to long format. Now the MDBU and MDBW values are in separate columns.
- Use the **centile** command to estimate specified centiles for the MDBU values, namely the 2.5th and the 97.5th centiles, which are equivalent to the lower and upper 95% confidence intervals.
- Use the **matrix** command to store the 2.5th centile value, saved in **r(c_1)**, in a matrix called *mdbu_ll*.
- Use the **matrix** command to store the 97.5th centile value, saved in **r(c_2)**, in a matrix called *mdbu_ul*.
- Repeat to estimate and store centile values for MDBW.
- Use the **restore** command to restore the dataset to the original.
- Use the **drop** command to drop all of the *est_*, *mdbu_* and *mdbw_* values that were created from the 100 simulations, as these are no longer needed.
- The lower 95% confidence intervals (*mdbu_ll* and *mdbw_ll*) are equal to the value stored at position [1,1] of matrices *mdbu_ll* and *mdbw_ll*, respectively.
- The upper 95% confidence intervals (*mdbu_ul* and *mdbw_ul*) are equal to the value stored at position [1,1] of matrices *mdbu_ul* and *mdbw_ul*, respectively.

***Calculate the 95% confidence intervals**

```

preserve
    keep in 1
    keep mdbu_* mdbw_*
    gen count=1
    reshape long mdbu_ mdbw_, i(count) j(simulation)
    centile mdbu_, c(2.5 97.5)
    matrix mdbu_ll=(r(c_1))
    matrix mdbu_ul=(r(c_2))
    centile mdbw_, c(2.5 97.5)
    matrix mdbw_ll=(r(c_1))
    matrix mdbw_ul=(r(c_2))
restore
drop est_* mdbu_* mdbw_*
gen mdbu_ll=mdbu_ll[1,1]
gen mdbu_ul=mdbu_ul[1,1]
gen mdbw_ll=mdbw_ll[1,1]
gen mdbw_ul=mdbw_ul[1,1]

```

Mean Difference from a Reference Subgroup

Overview

Type of summary measure: Complex; absolute; weighted or non-weighted.

Applicability: Non-ordered; more than two subgroups.

Definition: The Mean Difference from a Reference Subgroup (MDR) is an absolute measure of inequality that shows the mean difference between each population subgroup and a reference subgroup. MDR can be calculated as an unweighted or weighted measure. For the unweighted version, all subgroups are weighted equally. For the weighted version, subgroups are weighted according to their population share.

Interpretation: MDR only has positive values, with larger values indicating higher levels of inequality. MDR is zero if there is no inequality.

Calculation: The unweighted version (MDRU) is calculated as the average of absolute differences between the subgroup estimates y_j and the estimate for the reference subgroup y_{ref} , divided by the number of subgroups n :

$$MDRU = \frac{1}{n} * \sum_j |y_j - y_{ref}|$$

The weighted version (MDRW) is calculated as the weighted average of absolute differences between the subgroup estimates y_j and the estimate for the reference subgroup y_{ref} . Absolute differences are weighted by each subgroup's population share p_j :

$$MDRW = \sum_j p_j |y_j - y_{ref}|$$

95% confidence intervals can be calculated using a methodology of simulated estimates. The dataset is simulated a large number of times (e.g., 100) and MDRU/MDRW is calculated for each of the simulated samples. The 95% confidence intervals are based on the 2.5th and 97.5th centiles of the MDRU/MDRW results.

Calculating MDR

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of MDRU/MDRW cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups for the calculation of MDRW (not required for MDRU).	Numeric
<i>indicator_scale</i>	The scale of the indicator. For example, the scale of an indicator measured as a percentage is 100. The scale of an indicator measured as a rate per 1000 population is 1000. If this is missing, 95% confidence intervals of MDRU/MDRW cannot be calculated.	Numeric
<i>reference_subgroup</i>	A reference subgroup indicated with the value of 1.	Numeric

Calculation steps

Step 1

Calculate MDR for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2 Identify the reference subgroup estimate.

- 2.1 Use the **summarize** command to identify the subgroup estimate when the reference subgroup (indicated in `reference_subgroup`) is equal to 1.
- 2.2 **Generate** a variable `ref_estimate` that is equal to the maximum reference estimate value **r(max)**.

```
summarize estimate if reference_subgroup==1
gen ref_estimate=r(max)
```

Step 3**Calculate unweighted MDR (MDRU):**

The average of absolute differences between the subgroup estimates and the reference subgroup estimate, divided by the number of subgroups.

- 3.1 Calculate the absolute difference between each subgroup *estimate* and the reference subgroup estimate (*ref_estimate*). Use the **total** command to sum this across all subgroups.
- 3.2 Divide this by the number of subgroups (**_N**) to get the unweighted MDR.
- 3.3 Drop the `mdru_prep` variable, as it is no longer needed.

```
egen mdru_prep=total(abs(estimate-ref_estimate))
gen mdru=mdru_prep/_N
drop mdru_prep
```

Step 4**Calculate weighted MDR (MDRW):**

The weighted average of absolute differences between the subgroup estimates and the reference subgroup estimate.

- 4.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 4.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 4.3 Calculate the absolute differences between each subgroup *estimate* and the reference subgroup estimate (*ref_estimate*), multiplied by the population share of each subgroup (*popshare*). Use the **total** command to sum this all subgroups to get the weighted MDR.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen mdRw=total(popshare*abs(estimate-ref_estimate))
```

Step 5

Calculate the 95% confidence intervals of MDRU and MDWR using a methodology of simulated estimates.

- 5.1 The dataset will be simulated 100 times and MDR calculated for each simulation. Use the **forvalues** command to execute the subsequent code contained within braces {} 100 times (represented by 1/100). A local macro, *j*, is set to each simulation; for instance, in the first simulation, *j* is equal to 1.
- 5.2 For each simulated dataset an empty variable is generated, *est_`j'*. For example, for the first simulation, this variable is called *est_1*.
- 5.3 Each estimate will be simulated one at a time. Use the **forvalues** command again to execute the subsequent code contained within the braces {} for each of the rows (or subgroups) in the dataset, starting from 1 to the maximum number of rows (**_N**). A local macro, *n*, is set to identify each row; for instance, for the first row, *n* is equal to 1.
- 5.4 Use the **tempname** command to store an empty local macro, with the name *i`n'*. *`n'* will be replaced by each row number; for instance, for the first row, the local macro will be named *i1*. This will be used to create a temporary scalar variable, set to be blank (=.) to start.

```
forvalues j = 1/100 {
    gen est_`j'=.
    forvalues n = 1/`= _N' {
        tempname i`n'
        gen `i`n'`=.
```

- 5.5 If the indicator is measured as a percentage (**indicator_scale=100**), then the simulated estimates must fall between the values of 0 and 100. To ensure this happens, use the **while** command, which evaluates an expression and executes the code contained within braces until the expression is false. The expression being evaluated is whether the temporary scalar variable *i`n'* is blank.
 - Generate a variable simulation containing simulated estimates using the **rnormal** command, which generates normal random samples that have a mean and standard error equal to that in the original dataset.
 - Use the **summarize** command to calculate summary statistics for simulated estimate of the row in question.
 - If the simulated estimate of the row in question, captured using the stored **r(mean)** value from the previous step, is greater than or equal to 0 and less than or equal to 100, then replace the temporary scalar variable *i`n'* with the simulated estimate.
 - **Drop** the simulated estimates, and repeat the process until all rows have a simulated estimate.
- 5.6 If the indicator is not measured as a percentage (**indicator_scale!=100**), then the simulated estimates must be greater than or equal to 0. Repeat the previous steps, with the only difference being that the stored **r(mean)** value must be greater than or equal to 0.

```
*If indicator is a percentage, simulate estimates between 0 and 100
if indicator_scale==100 {
    while `i`n'`= . {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 & r(mean)<=100 {
```

```

        replace `i`n'`=r(mean)
    }
    drop simulation
}
replace est_`j'`= `i`n'`' if _n==`n'
}

*If indicator is not a percentage, simulate estimates greater than 0
if indicator_scale==100 {
    while `i`n'`= . {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 {
            replace `i`n'`=r(mean)
        }
        drop simulation
    }
    replace est_`j'`= `i`n'`' if _n==`n'
}

```

- 5.7 Calculate MDUR and MDWR using the simulated estimates and the method outlined in Steps 3 and 4. The estimate of the reference subgroup must be identified using the simulated estimates (*ref_estimate_`j'*). The simulated estimates (*est_`j'*) and the new reference estimate are used in the MDUR and MDWR formulas.

```

*Calculate MDUR using the simulated estimates
summarize est_`j'` if reference_subgroup==1
gen ref_estimate_`j'`=r(max)
egen mdru_prep=total(abs(est_`j'`-ref_estimate_`j'`))
gen mdru_`j'`=mdru_prep/_N
drop ref_estimate_`j'` mdru_prep

*Calculate MDWR using the simulated estimates
summarize est_`j'` if reference_subgroup==1
gen ref_estimate_`j'`=r(max)
egen mdwr_`j'`=total(popshare*abs(est_`j'`-ref_estimate_`j'`))
drop ref_estimate_`j'`

```

- 5.8 Calculate the 95% confidence intervals using the MDUR and MDWR values that were computed using the simulated data.
- Use the **preserve** command to preserve the dataset, so that the original data can be restored later.
 - **Keep** only the values in the first row (row=1).
 - **Keep** only the MDUR and MDWR values that were calculated using the simulated data.
 - **Generate** a dummy *count* variable, for use in the subsequent step.
 - Use the **reshape** command to reshape the data from wide to long format. Now the MDUR and MDWR values are in separate columns.

- Use the **centile** command to estimate specified centiles for the MDRU values, namely the 2.5th and the 97.5th centiles, which are equivalent to the lower and upper 95% confidence intervals.
- Use the **matrix** command to store the 2.5th centile value, saved in **r(c_1)**, in a matrix called **mdru_ll**.
- Use the **matrix** command to store the 97.5th centile value, saved in **r(c_2)**, in a matrix called **mdru_ul**.
- Repeat to estimate and store centile values for MDRW.
- Use the **restore** command to restore the dataset to the original.
- Use the **drop** command to drop all of the *est_*, *mdru_* and *mdrw_* values that were created from the 100 simulations, as these are no longer needed.
- The lower 95% confidence intervals (*mdru_ll* and *mdrw_ll*) are equal to the value stored at position [1,1] of matrices *mdru_ll* and *mdrw_ll*, respectively.
- The upper 95% confidence intervals (*mdru_ul* and *mdrw_ul*) are equal to the value stored at position [1,1] of matrices *mdru_ul* and *mdrw_ul*, respectively.

***Calculate the 95% confidence intervals**

```

preserve
  keep in 1
  keep mdru_* mdrw_*
  gen count=1
  reshape long mdru_ mdrw_, i(count) j(simulation)
  centile mdru_, c(2.5 97.5)
  matrix mdru_ll=(r(c_1))
  matrix mdru_ul=(r(c_2))
  centile mdrw_, c(2.5 97.5)
  matrix mdrw_ll=(r(c_1))
  matrix mdrw_ul=(r(c_2))
restore
drop est_* mdru_* mdrw_*
gen mdru_ll=mdru_ll[1,1]
gen mdru_ul=mdru_ul[1,1]
gen mdrw_ll=mdrw_ll[1,1]
gen mdrw_ul=mdrw_ul[1,1]

```

Index of Disparity

Overview

Type of summary measure: Complex; relative; weighted.

Applicability: Non-ordered; more than two subgroups.

Definition: The Index of Disparity (IDIS) is a relative measure of inequality that shows the average difference between each subgroup and the setting average, in relative terms. IDIS can be calculated as an unweighted or weighted measure. For the unweighted version, all subgroups are weighted equally. For the weighted version, subgroups are weighted according to their population share.

Interpretation: IDIS has only positive values, with larger values indicating higher levels of inequality. IDIS is zero if there is no inequality.

Calculation: The unweighted version (IDISU) is calculated as the average of absolute differences between the subgroup estimates y_j and the setting average μ , divided by the number of subgroups n and the setting average μ , and multiplied by 100:

$$IDISU = \frac{\frac{1}{n} * \sum_j |y_j - \mu|}{\mu} * 100$$

The weighted version (IDISW) is calculated as the weighted average of absolute differences between the subgroup estimates y_j and the setting average μ , divided by the setting average μ , and multiplied by 100. Absolute differences are weighted by each subgroup's population share p_j :

$$IDISW = \frac{\sum_j p_j |y_j - \mu|}{\mu} * 100$$

95% confidence intervals can be calculated using a methodology of simulated estimates. The dataset is simulated a large number of times (e.g., 100) and IDISU/IDISW is calculated for each of the simulated samples. The 95% confidence intervals are based on the 2.5th and 97.5th centiles of the IDISU/IDISW results.

Calculating IDIS

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of IDISU/IDISW cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric
<i>indicator_scale</i>	The scale of the indicator. For example, the scale of an indicator measured as a percentage is 100. The scale of an indicator measured as a rate per 1000 population is 1000. If this is missing, 95% confidence intervals of MDRU/MDRW cannot be calculated.	Numeric
<i>setting_average</i>	The indicator average for the setting (e.g., national average). Optional for the calculation of IDISU if population data are not available.	Numeric

Calculation steps

Step 1

Calculate IDIS for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Calculate the population share of each subgroup and the weighted mean of subgroup estimates. *Note: alternatively, use the setting average instead of calculating the weighted mean.*

- 2.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 2.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 2.3 Use the **total** command to sum the subgroup estimates (*estimate*) multiplied by the subgroup population shares (*popshare*) across all subgroups.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)
```

Step 3

Calculate unweighted IDIS (IDISU):

The average of absolute differences between the subgroup estimates and the weighted mean, divided by the number of subgroups and the weighted mean. *Note: alternatively, use the setting average instead of the weighted mean.*

- 3.1 Calculate the absolute difference between each subgroup estimate (*estimate*) and the weighted mean (*weighted_mean*). Use the **total** command to sum this across all subgroups.
- 3.2 Divide this by the number of subgroups (**_N**) and the weighted mean (*weighted_mean*) and multiply the result by 100.
- 3.3 Drop the *idisu_prep* variable, as it is no longer needed.

```
egen idisu_prep=total(abs(estimate-weighted_mean))
gen idisu=(idisu_prep/_N/weighted_mean)*100
drop idisu_prep
```

Step 4

Calculate weighted IDIS (IDISW):

The weighted average of absolute differences between the subgroup estimates and the weighted mean, divided by the weighted mean.

- 4.1 Calculate the absolute difference between each subgroup estimate (*estimate*) and the weighted mean (*weighted_mean*). Multiply this by the population share of each subgroup (*popshare*). Use the **total** command to sum this across all subgroups.
- 4.2 Divide this by the weighted mean (*weighted_mean*) and multiply the result by 100.
- 4.3 Drop the *idisw_prep* variable, as it is no longer needed.

```
egen idisw_prep=total(popshare*abs(estimate-weighted_mean))
gen idisw=(idisw_prep/weighted_mean)*100
drop idisw_prep
```

Step 5

Calculate the 95% confidence intervals of IDISU and IDISW using a methodology of simulated estimates.

- 5.1 The dataset will be simulated 100 times and IDIS calculated for each simulation. Use the **forvalues** command to execute the subsequent code contained within braces {} 100 times (represented by 1/100). A local macro, *j*, is set to each simulation; for instance, in the first simulation, *j* is equal to 1.
- 5.2 For each simulated dataset an empty variable is generated, *est_`j'*. For example, for the first simulation, this variable is called *est_1*.
- 5.3 Each estimate will be simulated one at a time. Use the **forvalues** command again to execute the subsequent code contained within the braces {} for each of the rows (or subgroups) in the dataset, starting from 1 to the maximum number of rows (**_N**). A local macro, *n*, is set to identify each row; for instance, for the first row, *n* is equal to 1.
- 5.4 Use the **tempname** command to store an empty local macro, with the name *i`n'*. *`n'* will be replaced by each row number; for instance, for the first row, the local macro will be named *i1*. This will be used to create a temporary scalar variable, set to be blank (=.) to start.

```
forvalues j = 1/100 {
  gen est_`j'=.
  forvalues n = 1/`= _N' {
    tempname i`n'
    gen `i`n'`=.
```

- 5.5 If the indicator is measured as a percentage (**indicator_scale=100**), then the simulated estimates must fall between the values of 0 and 100. To ensure this happens, use the **while** command, which evaluates an expression and executes the code contained within braces until the expression is false. The expression being evaluated is whether the temporary scalar variable *i`n'* is blank.
 - Generate a variable simulation containing simulated estimates using the **rnormal** command, which generates normal random samples that have a mean and standard error equal to that in the original dataset.
 - Use the **summarize** command to calculate summary statistics for simulated estimate of the row in question.
 - If the simulated estimate of the row in question, captured using the stored **r(mean)** value from the previous step, is greater than or equal to 0 and less than or equal to 100, then replace the temporary scalar variable *i`n'* with the simulated estimate.
 - **Drop** the simulated estimates, and repeat the process until all rows have a simulated estimate.
- 5.6 If the indicator is not measured as a percentage (**indicator_scale!=100**), then the simulated estimates must be greater than or equal to 0. Repeat the previous steps, with the only difference being that the stored **r(mean)** value must be greater than or equal to 0.

```

*If indicator is a percentage, simulate estimates between 0 and 100
if indicator_scale==100 {
    while `i`n'==. {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 & r(mean)<=100 {
            replace `i`n'`=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'`' if _n==`n'
}

*If indicator is not a percentage, simulate estimates greater than 0
if indicator_scale==100 {
    while `i`n'==. {
        gen simulation=rnormal(estimate,se)
        qui summarize simulation if _n==`n'
        if r(mean)>=0 {
            replace `i`n'`=r(mean)
        }
        drop simulation
    }
    replace est_`j'=`i`n'`' if _n==`n'
}

```

- 5.7 Calculate IDISU and IDISW using the simulated estimates and the method outlined in Steps 3 and 4. The weighted mean must be recalculated using the simulated estimates (*est_`j'*). The simulated estimates (*est_`j'*) and the new weighted mean (*weighted_mean_`j'*) are used in the IDISU and IDISW formulas.

```

*Calculate IDISU using the simulated estimates
egen weighted_mean_`j'`=total(popshare*est_`j'`)
egen idisu_prep=total(abs(est_`j'`-weighted_mean_`j'`))
gen idisu_`j'`=(idisu_prep/_N/weighted_mean_`j'`)*100
drop weighted_mean_`j'` idisu_prep

*Calculate IDISW using the simulated estimates
egen weighted_mean_`j'`=total(popshare*est_`j'`)
egen idisw_prep=total(popshare*abs(est_`j'`-weighted_mean_`j'`))
gen idisw_`j'`=(idisw_prep/weighted_mean_`j'`)*100
drop weighted_mean_`j'` idisw_prep

```

- 5.8 Calculate the 95% confidence intervals using the IDISU and IDISW values that were computed using the simulated data.

- Use the **preserve** command to preserve the dataset, so that the original data can be restored later.
- **Keep** only the values in the first row (row=1).
- **Keep** only the IDISU and IDISW values that were calculated using the simulated data.
- **Generate** a dummy *count* variable, for use in the subsequent step.

- Use the **reshape** command to reshape the data from wide to long format. Now the IDISU and IDISW values are in separate columns.
- Use the **centile** command to estimate specified centiles for the IDISU values, namely the 2.5th and the 97.5th centiles, which are equivalent to the upper and lower 95% confidence intervals.
- Use the **matrix** command to store the 2.5th centile value, saved in **r(c_1)**, in a matrix called **idisu_ll**.
- Use the **matrix** command to store the 97.5th centile value, saved in **r(c_2)**, in a matrix called **idisu_ul**.
- Repeat to estimate and store centile values for IDISW.
- Use the **restore** command to restore the dataset to the original.
- Use the **drop** command to drop all of the *est_*, *idisu_* and *idisw_* values that were created from the 100 simulations, as these are no longer needed.
- The lower 95% confidence intervals (*idisu_ll* and *idisw_ll*) are equal to the value stored at position [1,1] of matrices *idisu_ll* and *idisw_ll*, respectively.
- The upper 95% confidence intervals (*idisu_ul* and *idisw_ul*) are equal to the value stored at position [1,1] of matrices *idisu_ul* and *idisw_ul*, respectively.

***Calculate the 95% confidence intervals**

```

preserve
  keep in 1
  keep idisu_* idisw_*
  gen count=1
  reshape long idisu_ idisw_, i(count) j(simulation)
  centile idisu_, c(2.5 97.5)
  matrix idisu_ll=(r(c_1))
  matrix idisu_ul=(r(c_2))
  centile idisw_, c(2.5 97.5)
  matrix idisw_ll=(r(c_1))
  matrix idisw_ul=(r(c_2))
restore
drop est_* idisu_* idisw_*
gen idisu_ll=idisu_ll[1,1]
gen idisu_ul=idisu_ul[1,1]
gen idisw_ll=idisw_ll[1,1]
gen idisw_ul=idisw_ul[1,1]

```

Theil Index

Overview

Type of summary measure: Complex; relative; weighted.

Applicability: Non-ordered; more than two subgroups.

Definition: The Theil Index (TI) is a relative measure of inequality that considers all population subgroups. Subgroups are weighted according to their population share.

Interpretation: Greater absolute values indicate higher levels of inequality. TI is zero if there is no inequality. TI is more sensitive to differences further from the setting average (by the use of the logarithm).

Calculation: TI is calculated as the sum of products of the natural logarithm of the share of the indicator of each subgroup ($\ln \frac{y_j}{\mu}$), the share of the indicator of each subgroup ($\frac{y_j}{\mu}$) and the population share of each subgroup (p_j). TI may be easily interpreted when multiplied by 1000:

$$TI = 1000 * \sum_j p_j \frac{y_j}{\mu} \ln \frac{y_j}{\mu}$$

where y_j indicates the estimate for subgroup j , p_j the population share of subgroup j and μ the setting average.

The variance of TI is calculated as:

$$var(TI) = \frac{\sum_j p_j^2 \sigma_j^2 \left[\left(1 + \ln \left(\frac{y_j}{\mu} \right) \right) - \left(\sum_k p_k \frac{y_k}{\mu} \left(1 + \ln \left(\frac{y_k}{\mu} \right) \right) \right) \right]^2}{\mu^2}$$

where σ_j is the standard error of the estimate for the subgroup j .

Calculating TI

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of TI cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric

Calculation steps

Step 1

Calculate TI for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Calculate the population share of each subgroup and the weighted mean of subgroup estimates.

- 2.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 2.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 2.3 Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)
```

Step 3 Replace any estimates equal to zero with 0.000001 to avoid issues with log.

To avoid a situation where TI cannot be calculated because $\log(0)$ is undefined, replace any estimates that are equal to 0 with a very small number such as 0.000001. **Generate** a variable *estimate_nozeros* that is equal to *estimate*. **Replace** any cases where the estimate equals 0 with 0.000001.

```
gen estimate_nozeros=estimate
replace estimate_nozeros=0.000001 if estimate==0
```

Step 4 Calculate the TI.

- 4.1 Calculate the share of the indicator in each subgroup by dividing the estimate of each subgroup by the weighted mean.
- 4.2 Calculate TI as the sum of the products of the share of the indicator in each subgroup (*rj*), the natural logarithm of the share of the indicator in each subgroup ($\log(rj)$) and the population share of each subgroup (*popshare*).
- 4.3 Multiply this by 1000 (which facilitates interpretation).

```
gen rj=estimate_nozeros/weighted_mean
egen ti_prep=total(rj*log(rj)*popshare)
gen ti=ti_prep*1000
```

Step 5 Calculate the 95% confidence intervals of the TI.

- 5.1 Calculate the sum of the products of the share of the indicator in each subgroup (*rj*), the natural logarithm of the share of the indicator in each subgroup plus one ($1+\log(rj)$) and the population share of each subgroup (*popshare*), saved as *ti_var_prep*.
- 5.2 Calculate the variance of TI (*ti_var*) following the formula provided in the Overview section. Take the natural logarithm of the share of the indicator in each subgroup plus one ($1+\log(rj)$) minus *ti_var_prep*, squared. Multiply this by the population share (*popshare*) squared and the standard error (*se*) of each subgroup squared. Divide the result by the weighted mean (*weighted_mean*) squared. Use the **total** command to sum the result across all subgroups.
- 5.3 The standard error of TI (*ti_se*) is the square root of the variance.
- 5.4 The lower 95% confidence interval (*ti_ll*) is calculated as the standard error of TI multiplied by 1.96, subtracted from TI.

- 5.5 The upper 95% confidence interval (ti_{ul}) is calculated as the standard error of TI multiplied by 1.96, added to TI.

```
egen ti_var_prep=total(rj*(1+log(rj))*popshare)
egen ti_var=total(((1+log(rj)-var_prep)^2)*((popshare^2)*(se^2) /
  (weighted_mean^2)))
gen ti_se=sqrt(ti_var)
gen ti_ll=ti-1.96*ti_se
gen ti_ul=ti+1.96*ti_se
```

Mean Log Deviation

Overview

Type of summary measure: Complex; relative; weighted.

Applicability: Non-ordered; more than two subgroups.

Definition: The Mean Log Deviation (MLD) is a relative measure of inequality that considers all population subgroups. Subgroups are weighted according to their population share.

Interpretation: MLD is zero if there is no inequality. Greater absolute values indicate higher levels of inequality. MLD is more sensitive to differences further from the setting average (by the use of the logarithm).

Calculation: MLD is calculated as the sum of products between the negative natural logarithm of the share of the indicator of each subgroup $[-\ln(\frac{y_j}{\mu})]$ and the population share of each subgroup p_j . MLD may be more easily readable when multiplied by 1000:

$$MLD = 1000 * \sum_j p_j \left[-\ln\left(\frac{y_j}{\mu}\right) \right]$$

where y_j indicates the estimate for subgroup j , p_j the population share of subgroup j and μ the setting average.

The variance of MLD is calculated as:

$$var(MLD) = \sum_j \frac{p_j^2 \sigma_j^2}{\mu^2} \left(1 - \frac{1}{y_j/\mu} \right)^2$$

where σ_j is the standard error of the estimate for the subgroup j .

Calculating MLD

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension should be non-ordered, meaning that its subgroups do not have an inherent ordering – such as subnational region.	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>se</i>	The standard error of the subgroup estimate. If this is missing, 95% confidence intervals of MLD cannot be calculated.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups.	Numeric

Calculation steps

Step 1

Calculate MLD for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Calculate the population share of each subgroup and the weighted mean of subgroup estimates.

- 2.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 2.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 2.3 Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)
```

Step 3 Replace any estimates equal to zero with 0.000001 to avoid issues with log.

To avoid a situation where MLD cannot be calculated because $\log(0)$ is undefined, replace any estimates that are equal to 0 with a very small number such as 0.000001. **Generate** a variable *estimate_nozeros* that is equal to *estimate*. **Replace** any cases where the estimate equals 0 with 0.000001.

```
gen estimate_nozeros=estimate
replace estimate_nozeros=0.000001 if estimate==0
```

Step 4 Calculate the MLD.

Calculate the share of the indicator in each subgroup by dividing each subgroup estimate by the weighted mean and calculate the negative natural logarithm of this. Multiply by the population share of each subgroup (*popshare*). Use the **total** command to sum this across all subgroups. Multiply this by 1000 (which facilitates interpretation).

```
egen mld_prep=total(popshare*-log(estimate_nozeros/weighted_mean))
gen mld=mld_prep*1000
```

Step 5 Calculate the 95% confidence intervals of the MLD.

- 5.1 Calculate the share of the indicator of each subgroup, by dividing the subgroup estimate by the weighted mean (*weighted_mean*).
- 5.2 Calculate the variance of MLD (*mld_var*) following the formula provided in the Overview section. Multiply the standard error (*se*) squared by the population share of each subgroup (*popshare*) squared, divided by the weighted mean (*weighted_mean*) squared. Divide the result by the weighted mean squared, multiplied by one minus the one divided by the share of the indicator of each subgroup (*rj*) squared.
- 5.3 The standard error of MLD (*mld_se*) is the square root of the variance.
- 5.4 The lower 95% confidence interval (*mld_ll*) is calculated as the standard error of MLD multiplied by 1.96, subtracted from MLD.
- 5.5 The upper 95% confidence interval (*mld_ul*) is calculated as the standard error of MLD multiplied by 1.96, added to MLD.

```
gen rj=estimate_nozeros/weighted_mean
egen mld_var=total(((se^2)*(popshare^2))/(weighted_mean^2))*((1-(1/rj))^2))
```



```
gen mld_se=sqrt(mld_var)
gen mld_ll=mld-1.96*mld_se
gen mld_ul=mld+1.96*mld_se
```

Population Attributable Fraction

Overview

Type of summary measure: Complex; relative; weighted.

Applicability: Ordered and non-ordered; more than two subgroups.

Definition: The Population Attributable Fraction (PAF) is a relative measure of inequality that shows the potential improvement in the average of an indicator, in relative terms, that could be achieved if all population subgroups had the same level of the indicator as a reference group.

Interpretation: PAF assumes positive values for favourable indicators and negative values for non-favourable (adverse) indicators. The larger the absolute value of PAF, the higher the level of inequality. PAF is zero if no further improvement can be achieved (i.e., if all subgroups have reached the same level of the indicator as the reference subgroup or surpassed that level).

Calculation: PAF is calculated as the difference between the estimate for the reference subgroup y_{ref} and the setting average μ , divided by the setting average and multiplied by 100:

$$PAF = \frac{y_{ref} - \mu}{\mu} * 100$$

If the indicator is favourable and $PAF < 0$, then PAF is replaced with 0. If the indicator is adverse and $PAF > 0$, then PAF is replaced with 0. The selection of the reference subgroup y_{ref} depends on the characteristics of the inequality dimension and the indicator type. It is the most-advantaged subgroup for ordered dimensions. For non-ordered dimensions, it is the subgroup with the highest estimate for favourable indicators and is the subgroup with the lowest estimate for adverse indicators.

The variance of PAF is calculated as:

$$var(PAF) = \frac{cN[ad(N - c) + bc^2]}{(a + c)^3(c + d)^3}$$

where a , b , c , d , and N are numbers of people based on a 2x2 contingency table:

		Indicator		
		Achieved	Not achieved	Total
Population subgroup	All other subgroups	a	b	$m_1 = a + b$
	Reference subgroup y_{ref}	c	d	$m_2 = c + d$
	Total	$n_1 = a + c$	$n_2 = b + d$	$N = a + b + c + d$

Calculating PAF

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension can be ordered (e.g., economic status) or non-ordered (e.g., subnational region).	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within an economic status dimension, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5). Within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups for the calculation of 95% confidence intervals of PAF.	Numeric
<i>indicator_scale</i>	The scale of the indicator. For example, the scale of an indicator measured as a percentage is 100. The scale of an indicator measured as a rate per 1000 population is 1000. If this is missing, 95% confidence intervals of PAF cannot be calculated.	Numeric
<i>setting_average</i>	The indicator average for the setting (e.g., national average). Optional for the calculation of PAF if population data are not available.	Numeric

Optional variables for the identification of y_{ref}

Variable	Definition	Type
<i>ordered_dimension</i>	1 for ordered dimensions (which have subgroups with a natural ordered, such as economic status) and 0 for non-ordered or binary dimensions (such as subnational region or sex).	Numeric (0 or 1)
<i>subgroup_order</i>	If the dimension is ordered, the order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring economic-related inequality using five wealth quintiles,	Numeric (integer)

quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.

<i>favourable_indicator</i>	1 for favourable indicators (which measure desirable health events where the ultimate goal is to achieve a maximum level, such as skilled birth attendance) and 0 for non-favourable indicators (which measure undesirable health events where the ultimate goal is to achieve a minimum level, such as under-five mortality rate).	Numeric (0 or 1)
-----------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------

Calculation steps

Step 1

Calculate PAF for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Identify the reference subgroup. The choice of reference subgroup depends on the characteristics of the inequality dimension and indicator type for which PAF is being calculated.

Generate an empty variable, *refgroup*. In the subsequent steps, *refgroup* will take the value of 1 to identify the reference subgroup.

- If the dimension is ordered (*ordered_dimension==1*), execute the commands between the braces {}. In this case, the reference subgroup is the most-advantaged, i.e., the subgroup with the largest *subgroup_order*. Use the **summarize** command can be to identify the maximum *subgroup_order*, saved in **r(max)**.
- If the dimension is non-ordered (*ordered_dimension==0*), execute the commands between the braces {}. In these cases, the reference subgroup is the subgroup with the highest indicator value for favourable indicators (*favourable_indicator==1*), or the or the subgroup with the lowest indicator value for adverse indicators (*favourable_indicator==0*). The **summarize** command can be used to identify the maximum and minimum estimates, saved in **r(max)** and **r(min)** respectively.

```
gen refgroup=.
*Ordered dimensions
if ordered_dimension==1 {
    summarize subgroup_order
    replace refgroup=1 if subgroup_order==r(max)
}
*Non-ordered dimensions
if ordered_dimension==0 {
    summarize estimate
```

```

replace refgroup=1 if estimate==r(max) & favourable_indicator==1
replace refgroup=1 if estimate==r(min) & favourable_indicator==0
}

```

Step 3

Calculate the weighted mean of subgroup estimates. *Note: alternatively, use the setting average instead of calculating the weighted mean.*

- 3.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 3.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 3.3 Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```

egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)

```

Step 4

Calculate PAF as the difference between the estimate for the reference subgroup and the weighted mean divided by the weighted mean. *Note: alternatively, use the setting average instead of the weighted mean.*

- 4.1 Identify the estimate of the reference subgroup. Use the **summarize** command to store the estimate value of the subgroup that has been selected as the reference subgroup in **r()**. Then **generate** a variable, *ref_estimate*, equal to this estimate.
- 4.2 Calculate PAF as the difference between the estimate for the reference subgroup (*ref_estimate*) and the weighted mean (*weighted_mean*), divided by the weighted mean, multiplied by 100.
- 4.3 If the indicator is favourable (*favourable_indicator*==1) and PAF has a negative value, use the **replace** command to make PAF equal to zero.
- 4.4 If the indicator is adverse (*favourable_indicator*==0) and PAF has a positive value, use the **replace** command to make PAF equal to zero.

```

summarize estimate if refgroup==1
gen ref_estimate=r(max)
gen paf=((ref_estimate-weighted_mean)/weighted_mean)*100
replace paf=0 if favourable_indicator==1 & paf<0
replace paf=0 if favourable_indicator==0 & paf>0

```

Step 5

Calculate the 95% confidence intervals of PAF.

- 5.1 Identify the population of the reference subgroup. Use the **summarize** command to store the population value of the subgroup that has been selected as the reference subgroup in **r()**. Then **generate** a variable, *ref_population*, equal to this estimate.
- 5.2 Next, construct the estimates for the 2x2 contingency table described in the Overview section.
 - **Generate** a variable *c* containing the number of people who achieved the indicator in the reference subgroup – equal to the reference estimate (*ref_estimate*) divided by the indicator scale (*indicator_scale*) multiplied by the reference population (*ref_population*).
 - **Generate** a variable *d* containing the number of people who did not achieve the indicator in the reference subgroup – equal to the reference population (*ref_population*) minus *c*.
 - **Generate** a variable *a* containing the number of people who achieved the indicator in all the other subgroups – equal to the weighed mean (*weighted_mean*) divided by the indicator scale (*indicator_scale*) multiplied by the total population (*sum_pop*).
 - **Generate** a variable *b* containing the number of people who did not achieve the indicator in all the other subgroups – equal to the total population (*sum_pop*) minus *a* minus the population of the reference group (*ref_population*).
 - **Generate** a variable *N* summing *a*, *b*, *c* and *d*.
- 5.3 Calculate the standard error of PAF (*paf_se*) as the square root of the variance, following the formula in the Overview section.
- 5.4 Calculate lower and upper confidence intervals:
 - The lower 95% confidence interval (*paf_ll*) is calculated as the standard error of PAF multiplied by 1.96, subtracted from PAF.
 - The upper 95% confidence interval (*paf_ul*) is calculated as the standard error of PAF multiplied by 1.96, added to PAF.

```
summarize population if refgroup==1
gen ref_population=r(max)
gen c=(ref_estimate/indicator_scale)*ref_population
gen d=ref_population-c
gen a=(weighted_mean/indicator_scale)*sum_pop-c
gen b=sum_pop-a-ref_population
gen N=a+b+c+d
gen paf_se=sqrt((c*N*(a*d*(N-c)+b*c^2))/((a+c)^3*(c+d)^3))
gen paf_ll=paf-1.96*paf_se
gen paf_ul=paf+1.96*paf_se
```

Population Attributable Risk

Overview

Type of summary measure: Complex; absolute; weighted.

Applicability: Ordered and non-ordered; more than two subgroups.

Definition: Population Attributable Risk (PAR) is an absolute measure of inequality that shows the potential improvement in the average of an indicator, in absolute terms, that could be achieved if all population subgroups had the same level of the indicator as a reference group.

Interpretation: PAR assumes positive values for favourable indicators and negative values for non-favourable (adverse) indicators. The larger the absolute value of PAR, the higher the level of inequality. PAR is zero if no further improvement can be achieved (i.e., if all subgroups have reached the same level of the indicator as the reference subgroup or surpassed that level).

Calculation: PAR is calculated as the difference between the estimate for the reference subgroup y_{ref} and the setting average μ :

$$PAR = y_{ref} - \mu$$

If the indicator is favourable and $PAR < 0$, then PAR is replaced with 0. If the indicator is adverse and $PAR > 0$, then PAR is replaced with 0. The selection of the reference subgroup y_{ref} depends on the characteristics of the inequality dimension and the indicator type. It is the most-advantaged subgroup for ordered dimensions. For non-ordered dimensions, it is the subgroup with the highest estimate for favourable indicators and is the subgroup with the lowest estimate for adverse indicators.

The variance of PAR is constructed from the 95% confidence intervals of PAF ($PAF \pm 1.96PAF_{se}$):

$$var(PAR) = \left(\frac{|\mu(PAF + 1.96PAF_{se}) - (PAF - 1.96PAF_{se})|}{2 * 1.96} \right)^2$$

Calculating PAR

Data requirements

Variable	Definition	Type
<i>indicator</i>	The indicator that is being measured.	String
<i>dimension</i>	A dimension of inequality, which is a demographic, socioeconomic or geographic characteristic (e.g., economic status, education, place of residence, subnational region). The dimension can be ordered (e.g., economic status) or non-ordered (e.g., subnational region).	String
<i>subgroup</i>	A subgroup of the population within a given dimension of inequality. There must be more than two subgroups within the dimension of inequality. For example, within an economic status dimension, the population could be categorized into five subgroups from the poorest (quintile 1) to the richest (quintile 5). Within a subnational region dimension, each subgroup is a different subnational region.	String
<i>estimate</i>	The subgroup estimate. Estimates should be available for all subgroups.	Numeric
<i>population</i>	The number of people within each subgroup (or the weighted population in the case of survey data). Population size must be available for all subgroups for the calculation of 95% confidence intervals of PAR.	Numeric
<i>indicator_scale</i>	The scale of the indicator. For example, the scale of an indicator measured as a percentage is 100. The scale of an indicator measured as a rate per 1000 population is 1000. If this is missing, 95% confidence intervals of PAR cannot be calculated.	Numeric
<i>setting_average</i>	The indicator average for the setting (e.g., national average). Optional for the calculation of PAR if population data are not available.	Numeric

Optional variables for the identification of y_{ref}

Variable	Definition	Type
<i>ordered_dimension</i>	1 for ordered dimensions (which have subgroups with a natural ordered, such as economic status) and 0 for non-ordered or binary dimensions (such as subnational region or sex).	Numeric (0 or 1)
<i>subgroup_order</i>	If the dimension is ordered, the order of subgroups in an increasing sequence starting with the value of 1. For example, if measuring economic-related inequality using five wealth quintiles,	Numeric (integer)

quintile 1 (poorest) is assigned the value of 1, quintile 2 (second poorest) is assigned the value of 2, quintile 3 is assigned the value of 3, and so on.

<i>favourable_indicator</i>	1 for favourable indicators (which measure desirable health events where the ultimate goal is to achieve a maximum level, such as skilled birth attendance) and 0 for non-favourable indicators (which measure undesirable health events where the ultimate goal is to achieve a minimum level, such as under-five mortality rate).	Numeric (0 or 1)
-----------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------

Calculation steps

Step 1

Calculate PAR for each unique group of estimates if the dataset has multiple settings, dates, data sources, indicators and/or dimensions.

If the dataset has multiple settings, dates, data sources, indicators and/or dimensions, then create a unique identifier for each group of subgroup estimates and calculate the measure for each group. See the [Data management considerations](#) section for more information.

Step 2

Identify the reference subgroup. The choice of reference subgroup depends on the characteristics of the inequality dimension and indicator type for which PAF is being calculated.

Generate an empty variable, *refgroup*. In the subsequent steps, *refgroup* will take the value of 1 to identify the reference subgroup.

- If the dimension is ordered (*ordered_dimension*==1), execute the commands between the braces {}. In this case, the reference subgroup is the most-advantaged, i.e., the subgroup with the largest *subgroup_order*. Use the **summarize** command can be to identify the maximum *subgroup_order*, saved in **r(max)**.
- If the dimension is non-ordered (*ordered_dimension*==0), execute the commands between the braces {}. In these cases, the reference subgroup is the subgroup with the highest indicator value for favourable indicators (*favourable_indicator*==1), or the or the subgroup with the lowest indicator value for adverse indicators (*favourable_indicator*==0). The **summarize** command can be used to identify the maximum and minimum estimates, saved in **r(max)** and **r(min)** respectively.

```
gen refgroup=.
*Ordered dimensions
if ordered_dimension==1 {
    summarize subgroup_order
    replace refgroup=1 if subgroup_order==r(max)
}
*Non-ordered dimensions
if ordered_dimension==0 {
    summarize estimate
```

```
replace refgroup=1 if estimate==r(max) & favourable_indicator==1
replace refgroup=1 if estimate==r(min) & favourable_indicator==0
}
```

Step 3

Calculate the weighted mean of subgroup estimates. *Note: alternatively, use the setting average instead of calculating the weighted mean.*

- 3.1 Use the **egen** command to calculate the total population, *sum_pop*.
- 3.2 Calculate the proportion of the total population that is in each subgroup, *popshare*.
- 3.3 Use the **total** command to sum the subgroup *estimate* multiplied by its population share (*popshare*) across all subgroups, which is the weighted mean.

```
egen sum_pop=total(population)
gen popshare=population/sum_pop
egen weighted_mean=total(popshare*estimate)
```

Step 4

Calculate PAR as the difference between the estimate for the reference subgroup and the weighted mean. *Note: alternatively, use the setting average instead of the weighted mean.*

- 4.1 Identify the estimate of the reference subgroup. Use the **summarize** command to store the estimate value of the subgroup that has been selected as the reference subgroup in **r()**. Then **generate** a variable, *ref_estimate*, equal to this estimate.
- 4.2 Calculate PAR as the difference between the estimate for the reference subgroup (*ref_estimate*) and the weighted mean (*weighted_mean*).
- 4.3 If the indicator is favourable (*favourable_indicator*==1) and PAR has a negative value, use the **replace** command to make PAR equal to zero.
- 4.4 If the indicator is adverse (*favourable_indicator*==0) and PAR has a positive value, use the **replace** command to make PAR equal to zero.

```
summarize estimate if refgroup==1
gen ref_estimate=r(max)
gen par=ref_estimate-weighted_mean
replace par=0 if favourable_indicator==1 & par<0
replace par=0 if favourable_indicator==0 & par>0
```

Step 5

Calculate the 95% confidence intervals of PAR.

- 5.1 Identify the population of the reference subgroup. Use the **summarize** command to store the population value of the subgroup that has been selected as the reference subgroup in **r()**. Then **generate** a variable, *ref_population*, equal to this estimate.

5.2 Next, construct the estimates for the 2x2 contingency table described in the Overview section.

- **Generate** a variable *c* containing the number of people who achieved the indicator in the reference subgroup – equal to the reference estimate (*ref_estimate*) divided by the indicator scale (*indicator_scale*) multiplied by the reference population (*ref_population*).
- **Generate** a variable *d* containing the number of people who did not achieve the indicator in the reference subgroup – equal to the reference population (*ref_population*) minus *c*.
- **Generate** a variable *a* containing the number of people who achieved the indicator in all the other subgroups – equal to the weighed mean (*weighted_mean*) divided by the indicator scale (*indicator_scale*) multiplied by the total population (*sum_pop*).
- **Generate** a variable *b* containing the number of people who did not achieve the indicator in all the other subgroups – equal to the total population (*sum_pop*) minus *a* minus the population of the reference group (*ref_population*).
- **Generate** a variable *N* summing *a*, *b*, *c* and *d*.

5.3 Calculate PAF as the difference between the estimate for the reference subgroup (*ref_estimate*) and the weighted mean (*weighted_mean*), divided by the weighted mean, multiplied by 100.

5.4 Calculate the standard error of the population attributable fraction (PAF) (*paf_se*) as the square root of the variance, following the formula in the Overview section of PAF.

5.5 Calculate the standard error of PAR from the 95% confidence intervals of PAF. The PAF confidence intervals are calculated as PAF plus and minus the standard error of PAF multiplied by 1.96. Subtract these confidence intervals, multiply the result by the weighted mean (*weighted_mean*), and divide the absolute result by 2 times 1.96.

5.6 Calculate lower and upper confidence intervals of PAR:

- The lower 95% confidence interval (*par_ll*) is calculated as the standard error of PAF multiplied by 1.96, subtracted from PAF.
- The upper 95% confidence interval (*paf_ul*) is calculated as the standard error of PAF multiplied by 1.96, added to PAF.

```
summarize population if refgroup==1
gen ref_population=r(max)
gen c=(ref_estimate/indicator_scale)*ref_population
gen d=ref_population-c
gen a=(weighted_mean/indicator_scale)*sum_pop-c
gen b=sum_pop-a-ref_population
gen N=a+b+c+d
gen paf=((ref_estimate-weighted_mean)/weighted_mean)*100
gen paf_se=sqrt((c*N*(a*d*(N-c)+b*c^2))/((a+c)^3*(c+d)^3))
gen par_se=abs(weighted_mean*((paf+1.96*paf_se)-(paf-1.96*paf_se)))/(2*1.96)
gen paf_ll=paf-1.96*paf_se
gen paf_ul=paf+1.96*paf_se
```

Further references

- Ahn J, Harper S, Yu M, Feuer EJ, Liu B, Luta G. Variance Estimation and Confidence Intervals for 11 Commonly Used Health Disparity Measures. *JCO Clin Cancer Inform.* 2018 Dec;2:1–19. <https://doi.org/10.1200/CCI.18.00031>
- Harper S, Lynch J, Meersman SC, Breen N, Davis WW, Reichman ME. An overview of methods for monitoring social disparities in cancer with an example using trends in lung cancer incidence by area-socioeconomic position and race-ethnicity, 1992-2004. *Am J Epidemiol.* 2008;167(8):889-899. <https://doi.org/10.1093/aje/kwn016>
- Keppel K, Pamuk E, Lynch J, Carter-Pokras O, Kim I, Mays V, Pearcy J, Schoenbach V, Weissman JS. Methodological issues in measuring health disparities. *Vital Health Stat. Ser. 2.* 2005;141:1–16. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3681823/>
- Moreno-Betancur M, Latouche A; Menvielle G, Kunst A, Rey G. Relative Index of Inequality and Slope Index of Inequality: A Structured Regression Framework for Estimation. *Epidemiology.* 2015;26(4). <https://doi.org/10.1097/EDE.0000000000000311>
- O'Donnell O, van Doorslaer E, Wagstaff A, Lindelow M. Analyzing Health Equity Using Household Survey Data: A Guide to Techniques and Their Implementation. World Bank; Washington, D.C: 2008. <https://hdl.handle.net/10986/6896>
- Schlotheuber A, Hosseinpour AR. Summary Measures of Health Inequality: A Review of Existing Measures and Their Application. *Int J Environ Res Public Health.* 2022 Mar 20;19(6):3697. <https://doi.org/10.3390/ijerph19063697>
- World Health Organization. Handbook on health inequality monitoring: with a special focus on low- and middle-income countries. Geneva: World Health Organization; 2013. <https://www.who.int/publications/i/item/9789241548632>