Oxford Poverty & Human Development Initiative (OPHI)
Oxford Department of International Development
Queen Elizabeth House (QEH), University of Oxford

OPHI ***Research in Progress Series* 62a**

mpitb: A toolbox for multidimensional poverty indices

Nicolai Suppa[*]

February 2022

Abstract
This paper presents `mpitb` a toolbox for multidimensional poverty indices (MPI). The Stata package `mpitb` comprises several subcommands to facilitate specification, estimation, and analyses of MPIs and supports the popular Alkire-Foster framework to multidimensional poverty measurement. `mpitb` offers several benefits to researchers, analysts and practitioners working on MPIs, including substantial time savings (e.g., due to lower data management and programming requirements) while allowing for a more comprehensive analysis at the same time. Moreover, the toolbox encourages to report standard errors or confidence intervals.

Keywords: multidimensional poverty, MPI, Stata, mpitb, Alkire-Foster method

JEL classification: I32, C88

Acknowledgments
As this toolbox was developed in the context of several releases of the global MPI since 2018, its development benefited from many helpful discussions with the wider OPHI team, ranging from specific aspects of the estimation to user needs. In particular, countless discussions with Usha Kanagaratnam, who is leading the global MPI since 2018, have been essential for both identifying the overall workflow and packaging parts of the code into this toolbox.

Funding Information: The author gratefully acknowledges funding from the la Caixa foundation (grant number LCF/PR/SR20/52550004).

Citation: Suppa, N. (2022). 'mpitb: A toolbox for multidimensional poverty indices.' *OPHI Research in Progress* 62a, Oxford Poverty and Human Development Initiative (OPHI), University of Oxford.

[*] Nicolai Suppa, Centre for Demographic Studies, Autonomous University of Barcelona, Spain and EQUALITAS and OPHI, University of Oxford nsuppa@ced.uab.es.

1 Introduction

By now measures of multidimensional poverty are popular in both academia and practice. Extending previous research on unidimensional poverty measures, several measures of multidimensional poverty have been proposed and axiomatically explored in the literature, for overviews see e.g., Aaberge and Brandolini (2015); Alkire et al. (2015).

Multidimensional measurement approaches to poverty seek to directly capture critical shortfalls in different dimensions of human well-being, such as health, education or social participation. Thereby, multidimensional poverty measures aim to go beyond an exclusive focus on monetary or material metrics in the resource space. The dual-cutoff-counting-approach proposed by Alkire and Foster (2011) is particular popular and, for instance, underlies the global Multidimensional Poverty Index (MPI) which is jointly published by the United Nations Development Programme and the Oxford Poverty and Human Development Initiative (UNDP-OPHI 2021) and has been analyzed in various studies (Jindra and Vaz 2019; Alkire, Nogales, Quinn, and Suppa 2021b). Moreover, official poverty measures in more than a dozen or so countries are based on this method, too.

Ideally, poverty measures involve methods simple enough to be communicable to the wider public. Moreover, from a technical point of view quantities of interest are usually easy to estimate as they are often averages of some sort (e.g., using `mean`). Furthermore, previous Stata packages such as DASP (Abdelkrim and Duclos 2007) and `mpi` (Pacifico and Poege 2017) already support the estimation of selected quantities of multidimensional poverty measurement.

A challenge in applied work, however, emerges on the data management side as usually a huge amount of estimates has to be produced, analyzed, archived, and presented. For preparing a measure of multidimensional poverty for a single country in a single year it is not uncommon to accumulate some 2000 point estimates during the course of a project, let alone a cross-country analysis of poverty changes over time. It is in particular this challenge where `mpitb` seeks to support researchers and practitioners alike.

`mpitb` has been developed to facilitate the production process of the global multidimensional poverty index. Indeed, the present toolbox has been developed in tandem with a new workflow for the global MPI (Suppa 2022). As a consequence, measures and quantities that can be estimated or produced out of the box by `mpitb` are closely aligned with the needs of the global MPI. Due to fundamental commonalities in multidimensional poverty measurement and analyses most features, however, can be expected to support researchers, analysts, and practitioners in their particular projects, too.

More specifically, `mpitb` offers several benefits to analysts of multidimensional poverty, including substantial time savings on the results data management side of the work and a considerable reduction of the programming needs required for a comprehensive analysis. Thereby, the toolbox allows analysts and researchers to focus on indicator construction, measure specification and the very analysis of the results. The use of this toolbox, moreover, encourages (i) to adopt a streamlined worfklow, which may help to spot errors and to improve the replicability of the results, and (ii) to report standard errors for estimated quantities, which is unfortunately not yet common practice in case of poverty estimates (e.g. World Bank 2017).

The remainder of this paper is organized as follows: section 2 introduces key quantities of multidimensional poverty which the toolbox can estimate. Section 3 briefly introduces selected tools and their syntax, whereas section 4 provides some examples.

2 Measuring multidimensional poverty

This section briefly introduces key quantities of the Alkire-Foster approach of multidimensional poverty to facilitate subsequent explanations. For a more comprehensive and formal presentation see Alkire et al. (2015, ch. 5). First, deprivation indicators flag whenever a household fails to achieve an outcome above the critical (deprivation) threshold in a particular dimension of human well-being. In the global MPI, for instance, a household is considered deprived in the nutrition indicator if any person under 70 years of age is undernourished (see Alkire et al. 2021a for further details).

The objective of many multidimensional poverty measures is to identify *multiply* deprived individuals or households. After assigning relative weights to all indicators, one may obtain the so-called deprivation score which is essentially a weighted deprivation count, ranging from zero to one (the maximum possible amount of deprivations). Applying a cross-dimensional poverty cutoff (k) to this deprivation score one can specify how much multiple deprivation a household (or individual) has to experience for being identified as (multidimensionally) poor. According to the global MPI, for instance, a household is considered poor if it experiences 1/3 or more of the maximum possible deprivations. Unlike the deprivation score, the *censored* deprivation score ignores deprivations of the non-poor by replacing their deprivation scores with zeros.

Quantities commonly estimated in this framework include (i) the headcount ratio (H), which is the proportion of people with a weighted deprivation count higher than the threshold k, (ii) the intensity (A), which is the average deprivation among the poor and (iii) the adjusted headcount ratio (M_0 or sometimes MPI), which may be obtained as the mean of censored deprivation score or as the product of H and A.

Further, several indicator-specific measures are of particular interest. First, the uncensored or raw headcount ratios (h_d) simply report the proportion of the population deprived in a particular indicator. Second, the *censored* headcount ratios $h_d(k)$ report the proportion of the population which is deprived in a particular indicator and (multidimensionally) poor at the same time. Third, the absolute contribution of an indicator to M_0, which follows from the dimensional breakdown property satisfied by M_0: all indicator-specific absolute contributions are adding up to M_0 itself. Absolute contributions may be obtained as the product of censored headcount ratios and the indicator's weight. Fourth, the percentage contribution of an indicator, which is the absolute contribution normalized by the value of M_0.

Another feature of the AF-framework is that most quantities may be disaggregated for certain subpopulations such as subnational regions or age groups. Finally, where data permits, changes over time for all of these quantities are frequently reported as absolute or relative changes where each may, moreover, be annualized or not, see Alkire et al. (2015, ch. 9.2).

As indicated above, the full specification of such a measure requires several parametric choices which are normative decisions as they have to balance different needs (data availability, policy relevance, etc.) and, moreover, involve value judgements (see Alkire et al. 2015, ch. 6). Therefore, it is not only important to flag those decisions accordingly, but there is also an interest in documenting the extent to which these choices may affect outcomes. Consequently, a fair amount of alternative specifications is usually estimated and studied as well, including alternative deprivations thresholds, dropping or removing a specific deprivation indicator altogether, alternative (cross-dimensional) poverty cutoffs, and different weighting schemes.

3 Selected tools and their syntax

This section presents the syntax of key tools provided by `mpitb`. Further tools are documented in the help files of the package.

3.1 mpitb set

`mpitb set` specifies the deprivation indicators for a MPI and stores this specification with the currently loaded dataset. One may store several specifications with one data set.

Syntax

`mpitb set` [`,` `name(`*mpiname*`)` `d1(`*varlist*`,` *subopts*`)` `...` `d5(`*varlist*`,` *subopts*`)` `description(`*text*`)` `clear` `replace`]

Options

`name(`*mpiname*`)` specifies the name of a particular MPI or, more precisely, its indicator selection. Internally, *mpiname* also serves as an ID and it is recommended to use short names (at most 10 characters are permitted).

`d1(`*varlist*`,` *subopts*`)` assigns the variables in *varlist* as deprivation indicators to dimension 1. It is recommended to use short variable names (at most 10 characters are permitted). A total of 10 dimensions is permitted. One may set the following *subopts*:

`name(`*dimname*`)` allows to choose a name for a particular dimension (optionally). It is recommended to use short names (at most 8 characters are permitted). If no name is provided dimensions are generically named `d1`, `d2`, etc.

`description(`*text*`)` the description allows to add some extra information of the particular MPI to the data.

`clear` removes all information on MPIs stored with the current data.

`replace` replaces the information for the specified MPI.

3.2 mpitb est

`mpitb est` estimates indices and subindices of multidimensional poverty for one or more parametrizations. Deprivation indicators have to be declared by `mpitb set` beforehand. `mpitb est` estimates levels, levels over time, and changes over time at the aggregate level and for subgroups. `mpitb est` provides standard errors and confidence intervals for most quantities and may take complex survey design into account.

Results may be coherently saved to disk or collected in frames (see [D] **frames**). `mpitb est` may create key variables for the AF-framework and a variable identifying the underlying sample (which takes, e.g., item non-responses into account).

Syntax

`mpitb est , name` [`klist(`*numlist*`)` `weights(`*wgts sopts*`)` `measures(`*mlist*`)` `indmeasures(`*imlist*`)` `indklist(`*numlist*`)` `aux(`*auxlist*`)` `cotmeasures(`*measures*`)` `cotoptions(`*olist*`)` `cotklist(`*numlist*`)` `cotyear(`*varname*`)` `dtasave(`*filename* `[,`*sopts*`])` `lframe(`*name* `[,`*sopts*`])` `lsave(`*filename[* `[,`*sopts*`])` `cotframe(`*name* `[,` *sopts*`])` `cotsave(`*filename* `[,`*sopts*`])` `over(`*varlist*`[,`*sopts*`])` `tvar(`*varname*`)` `svy` `addmeta(`*metalist*`)` `skipgen` `gen` `double` `noestimate`]

Measures and Parameters

`name(`*mpiname*`)` name of the MPI to be estimates (which also serves as ID)

`klist(`*numlist*`)` specifies the (cross-dimensional) poverty cutoff(s) in percentage points. Valid values are integers between 1 and 100. One or more values may be specified.

`measures(`*mlist*`)` the list of permitted measures may include *M0*, *H*, *A*, or *all*.

`weights(`*wgts* `[`*sopts*`])` specifies the weighting scheme(s), where *wgts* may be one of

`equal` this option applies a equal-nested weighting scheme.

`dimw(`*numlist*`)` allows to set arbitrary weighting schemes for dimensions. Weighting schemes can be set using decimal numbers from 0–1. Naturally, the number of weights has to match the number of dimensions and weights must sum up to one.

The order of dimension corresponds to the order used in `mpitb set`. Indicator weight within dimensions receive equal weights.

`indw(`*numlist*`)` allows to set arbitrary weighting schemes for indicators. Weighting schemes can be set using decimal numbers from 0–1. Naturally, the number of weights has to match the number of indicator and weights must sum up to one. The order of indicators corresponds to the order used in `mpitb set`.

Moreover, *sopts* may be `name(`*wgtsname*`)`. This option allows to assign names to particular weighting schemes and is required for use with `indw(`*numlist*`)` while being optional for the others.

`indmeasures(`*imlist*`)` the list of indicator-specific measures may include *hdk* (censored headcount ratios), *actb* (absolute contributions), *pctb* (percentage contribution), or *all*.

`indklist(`*numlist*`)` allows to choose a different set of poverty cutoffs for indicator-specific measures to avoid unnecessary long estimations for numbers of lower priority. Unless explicitly set, `indklist` equals `klist`.

`aux(`*auxlist*`)` the list of auxiliary measures may include *hd*, the uncensored deprivation headcount ratios, *mv*, *N* or *all*. *mv* includes the share of missing values on the indicator level and the retained sample at the aggregate level. The retained sample will be reported with and without sampling weights (if `svy` is set). *N* is the effective sample size, i.e. the number of observations with non-missing information on all indicators.

Changes over time

`cotmeasures(`*mlist*`)` the COT measure list may include *M0* (the adjusted headcount ratio), *H* (the headcount ratio), *A* (the intensity), *hd* (uncensored headcount ratios), *hdk* (censored headcount ratios), or all.

`cotoptions(`*olist*`)` where *olist*, the option list for COT, may include

`total` estimates change over total period of observation, i.e. from the first year of observation to the last year of observation.

`insequence` estimates all consecutive (i.e. year-to-year) changes.

`ann` produces annualized changes over time. This option activated by default. Specify `noann` to skip the estimation of annualized changes.

`raw` produces the raw, i.e. non-annualized changes over time. This option activated by default. Specify `noraw` to skip the estimation of raw changes.

`cotk(numlist)` specifies the poverty cutoffs for the COT estimation

`cotyear(`*varname*`)` specifies the variable which is used for the annualization, which is usually a year variable, where decimal digits are permitted.

Results

`dtasave(`*filename* `[,` *sopts*`])` saves the micro data after creating the variables of the AF-framework. This can be particular useful when mpitb est is run within a loop over countries. Available suboptions *sopts* are

`replace` which replaces any potentially existing dataset.

`lframe(`*name* `[,` *sopts*`])` saves the level estimates into a result frame under the specified name. This option can be useful for adding further custom estimates before saving all results to disk. See `mpitb stores` for adding estimates of custom quantities to the result frame. Available suboptions *sopts* are

`replace` which replaces any potentially existing frame

`lsave(`*filename* `[,` *sopts*`])` saves the levels estimates into the specified dta file. Available suboptions *sopts* are

`replace` which replaces any potentially existing dataset

`cotframe(`*name* `[,` *sopts*`])` saves the change estimates into a result frame under the specified name. Note that a potentially existing frame will be replaced. This option can be useful for adding further custom estimates before saving all results to disk. See `mpitb stores` for adding estimates of custom quantities to the result frame. Available suboptions *sopts* are

`replace` which replaces any potentially existing dataset

`cotsave(`*filename*`[,` *sopts*`])` saves the change estimates into the specified dta file. Available suboptions *sopts* are

`replace` which replaces any potentially existing dataset

Other

`over(varlist [,` *sopts*`])` allows to disaggregate by several variables. By default quantities for the subgroups are estimated for the same measure and parameters as the aggregate quantities. Suboptions allow to avoid unnecessary long estimations for numbers of lower priority. Available suboptions *sopts* are

`klist(`*numlist*`)` allows to choose a different set of poverty cutoffs for disaggregations.

`indklist(`*numlist*`)` allows to choose a different set of poverty cutoffs for disaggregations of indicator-specific measures.

`nooverall` if this option is set aggregate (or national-level) estimates will not be produced. This option may be useful for organizing results across different files.

`svy` estimation accounts for complex survey design of micro data as specified by `svyset`. If `svy` is not set, the data is assumed to be obtained through simple random sampling, which is rarely used in practice.

`addmeta(`*metalist*`)` allows to add meta data every estimate produced. A common application would be to add the ISO country code. *metalist* is used as follows

macname = content [*macname2 = content2 ...*]

`skipgen` this option allows to skip the step of generating all variables of the AF-framework. This can save runtime if variables have already been created by a `mpitb est` run previously. A common application is to save different results into a single file. However, it is up to the user to ensure that all needed variables do exist and that the results files are coherently augmented.

`gen` The default behavior of `mpitb est` is to remove all variables (e.g., deprivation scores or poverty status) generated upon completion of the estimations. Option `gen` allows to keep all variables for cross-checks or additional calculations.

`double` generates non-byte variables as double, which improves the precision with which, e.g., the deprivation score is stored as variable. The default is float.

`noestimate` allows to skip the entire estimation process. This allows to save time if only the generated variables are of interest.

3.3 mpitb refsh

`mpitb refsh` creates or updates the reference sheet, a key feature of the global MPI workflow. The reference sheet contains basic information about the countries covered by the current project. The reference may be used (i) to control estimation and other productions routines efficiently, (ii) for merging external data easily, (iii) reducing the amount of information that estimates are passed through the estimation routine. See the workflow for more details.

Essentially, `mpitb refsh` examines all micro datasets in the specified folder and collects certain information (data characteristics or variables). Afterwards `mpitb refsh` creates the reference sheet comprising this information for each country.

Syntax

`mpitb refsh using` *filename* `, id(`*name*`) path(`*string*`)` [`, clear newfiles update(`*clist*`) sid(`*sid*`) keep(`*namelist*`) char(`*clist*`) depind(`*depind*`) gentvar(`*year*`)`]

Options

`id(`*name*`)` specifies the identifier of a particular dataset and is usually an ISO country code. By default, name is assumed to be a variable name. If, however, option `char(`*clist*`)` is set, name may be a data characteristic, too. The reference sheet will contain at least one observation for each ID (or dataset).

`path(`*path*`)` specifies the path to the cleaned micro datasets. Note that path has to be specified as *folder/subfolder* using slashes "`/`" and not backslashes "\".

`clear` examines every dta file in the specified path for being included into the reference sheet and replaces any potentially already existing reference sheet. Usually, this option is the most convenient.

`newfiles` searches for new files in the specified path and adds them to the reference sheet if encountered. The old entries for this country will be replaced. This option is rarely used.

`update(`*clist*`)` updates the reference for selected countries. Usually *clist* would be country codes and refer to values of the variable specified in `id(`*name*`)` This option is rarely used.

`sid(`*sid*`)` specifies a secondary ID for subgroups within a country (or dataset) which is usually the subnational region variable. Unlike other most other subgroups, coding and labels of regions tend to vary across countries. The reference sheet will contain one observation for each region of a given country.

If `mpitb refsh` encounters are region variable containing only missing values, it only adds a single entry for this country to the reference sheet, whereas a dataset is entirely skipped if the specified variable is not found at all. This convention allows to distinguish countries for which the survey does not allow subnational disaggregation from countries which are not supposed to be included in particular analysis.

`keep(`*namelist*`)` allows to specify variables in the micro dataset, which are kept and stored in the reference sheet. These variables are assumed to be constant across all observations in the micro dataset (and missing values will be ignored). This option allows to pass further information from the micro data files to the reference sheet (and from there to the results file). Use cases may be country codes, survey names or years. Usually, using the option char is preferable, though.

`char(`*clist*`)` specifies a list of data characteristics (see char) of the micro data, which will be retained and added as variables to the reference sheet.

`depind(`*dlist*`)` collects information on deprivation indicators, where *dlist* is a list of all deprivation indicators supposed to be covered. If this option is set the `mpitb refsh` adds the number of indicators found in each dataset and the names of missing indicators.

`gentvar(`*year*`)` generates an integer time variable, which identifies the data rounds for each country based on the variable (or characteristic) year.

3.4 mpitb ctyselect

`mpitb ctyselect` selects one or more countries from the reference sheet and returns their country codes. *varname* is required and contains the name of the variable which holds the country codes. With `mpitb ctyselect` one may conveniently control loops for estimations or other steps in the production process. Called without any option, `mpitb ctyselect` returns all country codes found in the reference sheet.

Syntax

`mpitb ctyselect` *varname* [*if*] [*in*] [, `select(`*ctylist*`)` `rexp(`*regex*`)`]

Options

`select(`*ctylist*`)` allows to manually select country codes. Technically, it simply refers to values of *varname*, which could be string or numeric.

`rexp(`*regex*`)` allows to select country codes based on regular expressions applied to *varname*.

Stored results

`r(ctylist)`	list of countries	`r(Nctylist)`	number of countries

4 Examples

This section provides examples for (i) the basic one year for one country setting, (ii) how to avoid unneeded estimations, (iii) how to add estimates for alternative weighting schemes and indicator selections, (iv) how to estimates both levels and changes over time where data permits, (iv) the basic setup for a cross-country analysis. The underlying datasets are shipped with `mpitb` as ancilliary files.

▷ Example 1: A single year for a single country

For the first examples we use `syn_cdta.dta`, which is ‘cleaned’ synthetic data providing already binary deprivation indicators and, moreover, follows the common household survey structure. For the present example we further restrict this dataset to its first round.

```
. use syn_cdta.dta if t == 1 , clear
. sum
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
      d_nutr |      7,439    .2521844    .4342958          0          1
        d_cm |      7,500    .0629333    .2428592          0          1
      d_satt |      7,484    .3178781    .4656829          0          1
      d_educ |      7,500    .2993333    .4579966          0          1
      d_elct |      7,500       .3976    .4894346          0          1
-------------+---------------------------------------------------------
      d_sani |      7,500       .2384    .4261334          0          1
       d_wtr |      7,500    .2737333    .4459035          0          1
       d_hsg |      7,500    .4177333    .4932186          0          1
      d_ckfl |      7,500       .1484    .3555197          0          1
      d_asst |      7,500    .2829333    .4504543          0          1
-------------+---------------------------------------------------------
        area |      7,500    .5989333    .4901471          0          1
      region |      7,500    10.53347    5.808389          1         20
     stratum |      7,500    1055.853    580.8484        100       2005
         psu |      7,500     1055856    580848.3     100000    2005005
      weight |      7,500           1           0          1          1
-------------+---------------------------------------------------------
        year |      7,500        2010           0       2010       2010
           t |      7,500           1           0          1          1
```

Specifically, this dataset contains ten deprivation indicators, two variables for which we wish to disaggregate our MPI estimates (`area`, `region`), three variables providing information about the underlying survey design (`psu`, `weight`, `stratum`) and two variables providing information for each survey round (`year`, `t`).

MPIs are frequently estimated using household survey data. To account for their complex survey design, it is convenient to rely on Stata's suite of survey data commands, see [SVY] **svy** and [SVY] **svy estimation**. For the present example, first use `svyset` to specify the primary sampling unit, the strata and the sampling weight for each observation; see [SVY] **svyset** for details.

```
. svyset psu [pw=weight], strata(stratum)
Sampling weights: weight
             VCE: linearized
     Single unit: missing
        Strata 1: stratum
 Sampling unit 1: psu
           FPC 1: <zero>
```

For real-world data the documentation of the respective dataset provides the relevant information. Next we specify indicators similar to the global MPI, which are organized in three dimension (health, education and living standards). We use `mpitb set` to assign indicators to dimensions and to provide names for both dimensions (`hl`, `ed`, `ls`) and the entire specification (`trial01`).

```
. mpitb set , na(trial01) d1(d_cm d_nutr, na(hl)) d2(d_satt d_educ, na(ed)) ///
>         d3(d_elct d_wtr d_sani d_hsg d_ckfl d_asst, name(ls)) de(pref. spec)
```

Now assume the task is to estimate for the indicator selection `trial01` all aggregate (M_0, H and A) and indicator-specific measures (h_d, $h_d(k)$ and both absolute and percentage contributions). Further let the preferred weighting scheme be the equal-nested one, i.e. equal weights for all dimensions and equal indicator weights within dimensions and let $k = 20\%, 33\%, 50\%$ be of particular interest. Finally, we wish to obtain disaggregations by subnational regions and by area (i.e. urban versus rural) and to account for the complex survey design. Hence we issue `mpitb est` as follows

```
. mpitb est , name(trial01) meas(all) indmeas(all) aux(hd) klist(20 33 50) ///
>         weight(equal) svy lfr(myresults, replace) over(region area)
```

```
──┘ Specification └──────────────────────────────────────
Name: trial01.
Weighting scheme: equal.
Description: pref. spec
────────────────────────────────────────────────────────────────
Dimension 1: hl         0.3333    (d_cm d_nutr)
Dimension 2: ed         0.3333    (d_satt d_educ)
Dimension 3: ls         0.3333    (d_elct d_wtr d_sani d_hsg d_ckfl d_asst)
────────────────────────────────────────────────────────────────
Indicator 1: d_cm       0.1667
Indicator 2: d_nutr     0.1667
Indicator 3: d_satt     0.1667
Indicator 4: d_educ     0.1667
Indicator 5: d_elct     0.0556
Indicator 6: d_wtr      0.0556
Indicator 7: d_sani     0.0556
Indicator 8: d_hsg      0.0556
Indicator 9: d_ckfl     0.0556
Indicator 10: d_asst    0.0556
────────────────────────────────────────────────────────────────
No missing indicator was found.
──┘ Estimation └─────────────────────────────────────────
# accumulated estimates (levels): 19 (national main completed)
# accumulated estimates (levels): 109 (national indicators completed)
# accumulated estimates (levels): 489 (region completed)
# accumulated estimates (levels): 2289 (region indicators completed)
# accumulated estimates (levels): 2347 (area completed)
# accumulated estimates (levels): 2527 (area indicators completed)
(note: frame myresults not found)
──┘ Result frames & files └──────────────────────────────
  Level frame (myresults): Estimates overview
Number of subgroups:
  area:      2
  region:    20

──────────────────────────────────────
        |       level of analysis
        |   area   nat   region   Total
────────┼─────────────────────────────
measure |
  A     |      6     3       60      69
  H     |      6     3       60      69
  M0    |      6     3       60      69
  actb  |     60    30      600     690
  hd    |     20    10      200     230
  hdk   |     60    30      600     690
  pctb  |     60    30      600     690
  popsh |      2             20      22
  Total |    220   109    2,200   2,529
──────────────────────────────────────

Number of parameters:
  k:         3  (20 33 50)
  wgts:      1  (equal)
  spec:      1  (trial01)
────────────────────────────────────────────────────────────────
```

`mpitb est` reports progress and results along three tabs. The first tab summarizes the underlying specification including the indicators, dimensions, and weights. The second tab shows the progress during the estimation procedures and details the numbers of so far accumulated estimates. The final tab provides a summary of the produced result frames or files, including the number of estimates, the type of measures estimated, or the respective level of analysis).

In the command above we instructed `mpitb est` to store all results into a frame (see [D] **frames**) using the `lframe` option. All produced result files or frames follow a specific structure, which we now briefly explore.

```
. cwf myresults
. d
Contains data
 Observations:         2,529
    Variables:            14
────────────────────────────────────────────────────────────────────────────────
Variable      Storage   Display    Value
    name         type    format    label      Variable label
────────────────────────────────────────────────────────────────────────────────
b               float   %5.4f                 point estimate
se              float   %5.4f                 standard error
ll              float   %5.4f                 CI lower bound
ul              float   %5.4f                 CI upper bound
pval            float   %4.2f                 p-value
tval            float   %4.2f                 t-value
loa             str10   %10s                  level of analysis
measure         str10   %10s                  measure
indicator       str10   %10s                  indicator
spec            str10   %10s                  name of specification
wgts            str10   %10s                  weighting scheme
k               float   %9.0g                 poverty cutoff
ctype           byte    %8.0g      ctype      type of change
subg            int     %8.0g                 subgroup
────────────────────────────────────────────────────────────────────────────────
Sorted by:
     Note: Dataset has changed since last saved.
```

The key feature of this structure is that each observation represents an estimate. The core of each estimate includes the point estimate and its standard error (variables `b` and `se`), whereas the remaining meta variables specify the content for each estimate. See Suppa (2022) for a further discussion and Kanagaratnam and Suppa (2022) for details on the result files of the global MPI.

The result file may be conveniently explored using very basic commands such as `tab`, `list` or `tabdisp`. For instance, to see which measures are available for each level of analysis (`loa`) issue

```
. tab measure loa
           │          level of analysis
   measure │      area        nat     region │     Total
───────────┼─────────────────────────────────┼──────────
         A │         6          3         60 │        69
         H │         6          3         60 │        69
        M0 │         6          3         60 │        69
```

```
         actb │       60         30        600 │       690
           hd │       20         10        200 │       230
          hdk │       60         30        600 │       690
         pctb │       60         30        600 │       690
        popsh │        2          0         20 │        22
──────────────┼──────────────────────────────────┼──────────
        Total │      220        109      2,200 │     2,529
```

To directly inspect particular estimates and their standard errors such as all aggregate measures at the national-level for the preferred poverty cutoff, issue

```
. li measure b se if inlist(measure,"M0","H","A") & loa == "nat" & k == 33 , noo
> b

  ┌──────────────────────────┐
  │ measure        b       se │
  ├──────────────────────────┤
  │       H   0.3352   0.0055 │
  │      M0   0.1424   0.0025 │
  │       A   0.4248   0.0019 │
  └──────────────────────────┘
```

If we were interested in comparing censored and uncensored headcount ratios between urban and rural areas a quick `tabdisp` may provide first insights.

```
. tabdisp indicator measure subg if inlist(measure,"hd","hdk") ///
>         & loa == "area" & inlist(k,33,.) , cell(b)

──────────────────────────────────────────────
          │        subgroup and measure
          │ ────── 0 ──────    ────── 1 ──────
indicator │     hd     hdk         hd     hdk
──────────┼───────────────────────────────────
   d_asst │ 0.2795  0.1258     0.2856  0.1235
   d_ckfl │ 0.1540  0.0732     0.1437  0.0624
     d_cm │ 0.0681  0.0527     0.0595  0.0422
   d_educ │ 0.2980  0.1872     0.3002  0.1895
   d_elct │ 0.4007  0.1688     0.3954  0.1668
    d_hsg │ 0.4044  0.1604     0.4273  0.1805
   d_nutr │ 0.2416  0.1617     0.2595  0.1720
   d_sani │ 0.2584  0.1134     0.2259  0.0943
   d_satt │ 0.3208  0.2057     0.3148  0.1935
    d_wtr │ 0.2728  0.1285     0.2744  0.1251
──────────────────────────────────────────────
```

Note that already `tabdisp` allows to construct more complex tables and even more so the revised `table` command, see [R] **table**. In this example we will not cover proper labeling, though.

◁

▷ Example 2: Avoiding unnecessary estimations

In example 1, we stored the results in a frame. For examples 2 and 3 we will collect all estimates in files in a dedicated folder `results` before we later combine them into a single file. This folder may be created using

```
. mkdir results
```

In the context of multidimensional poverty measures, varying parameters may quickly make the numbers of estimates going through the roof. Having a clear sense of the priorities helps to guide any such analysis. Therefore, `mpitb` allows to reduce the number of estimates as needed.

For example, it is common to explore numbers for M_0, H, and A at the aggregate level for some 10 different values of the poverty cutoff over the entire domain of k, resulting in 30 estimates. Assuming a MPI with 10 deprivation indicators, the estimation of three indicator-specific measures for these values of k would add another 300 estimates at the aggregate level. For a country with 15 regions, another 1500 estimates would have to be added on the subnational level.

To purposefully reduce the number of estimates, `mpitb` allows to specify different ranges of k for different layers of the analysis. By default, the range specified through option `klist()` is applied to all measures and levels of analysis. However, option `indklist()` allows to apply a separate range for the indicator-specific measures, whereas the `over()` option, accepts a suboption `klist()` to restrict the range of alternative parameters for disaggregations. Consider the following example which only differs in the `mpitb est` command.

```
. use syn_cdta.dta if t == 1 , clear
. svyset psu [pw=weight], strata(stratum)
  (output omitted)
. mpitb set , name(trial01) desc(preferred spec) ///
>               d1(d_cm d_nutr, name(hl)) ///
>               d2(d_satt d_educ, name(ed)) ///
>               d3(d_elct d_wtr d_sani d_hsg d_ckfl d_asst, name(ls))
. mpitb est , name(trial01) measures(all) indmeas(all) aux(hd) svy ///
>         k(1 10 20 33 40 (10) 100) over(region area, k(20 33 50) indk(30)) ///
>         indk(20 33 40) weight(equal) lsa(results/trial01, replace)
  (output omitted)
. describe using results/trial01 , s
Contains data
 Observations:         1,232                 5 Feb 2022 09:42
    Variables:            14
Sorted by:
```

First, note that we now use the `lsave` option to store the results immediately to disk. Moreover, `describe` tells us, that the use of `indk()` and the `over` suboptions `klist()` and `indk()` the example above results in some 1200 instead of some 8600 estimates, thereby, saving estimation time and simplifying the subsequent analysis. [1]

◁

1. More precisely, three aggregate and three indicator-specific measures for ten indicators amount to 33 estimates per k (11) and level of analysis (23: national, urban, rural and 20 subnational regions). Moreover, 10 uncensored headcounts (which do not depend on k) may be added for each level of analysis and 22 population shares for all subgroups. All in all, $(33 \times 11 + 10) \times 23 + 22 = 8601$ would have to be estimated.

▷ Example 3: Adding alternative weights and indicator selections

So far our results feature only a single weighting scheme. The toolbox allows to set alternative weights in different ways. First, to set custom *dimensions*-specific weights (with equal weights within dimensions), say, 50% for health and 25% for the other two, use the `dimw(`*numlist*`)` option as follows

```
. mpitb est , n(trial01) m(all) k(33) w(dimw(.5 .25 .25) name(health50)) ///
>         lsa(results/health50, replace) svy
  (output omitted)
. mpitb est , n(trial01) m(all) k(33) w(dimw(.25 .5 .25) name(educ50)) ///
>         lsa(results/educ50, replace) svy
  (output omitted)
. mpitb est , n(trial01) m(all) k(33) w(dimw(.25 .25 .5) name(livst50)) ///
>         lsa(results/livstd50, replace) svy
  (output omitted)
```

Similarly, one may specify the alternative weighting schemes `educ50` and `livst50` . Second, sometimes one may wish assign custom *indicator* weights (e.g., equal indicator weights). Option `indw(`*numlist*`)` allows to do this as follows:

```
. mpitb est , n(trial01) m(all) k(33) lsa(results/ind_equal, replace) ///
>         w(indw(.1 .1 .1 .1 .1 .1 .1 .1 .1 .1) name(ind_equal)) svy
  (output omitted)
```

Finally, in order to consider alternative indicator choices, such as without the deprivation indicator for electricity (`d_elct`), first `mpitb set` this specification which we may call `trial02` and subsequently estimate the desired quantities.

```
. mpitb set , n(trial02) d1(d_cm d_nutr, n(hl)) d2(d_satt d_educ, n(ed)) ///
>         d3(d_wtr d_sani d_hsg d_ckfl d_asst, name(ls)) desc(w/o electricity)
  (output omitted)
. mpitb est , n(trial02) m(all) k(33) w(equal) svy ///
>         lsa(results/trial02, replace)
  (output omitted)
```

Note that this approach allows to conveniently analyze the effects of (i) dropping or adding single indicators, (ii) alternative deprivation thresholds for one or more of the indicators, and (iii) radically different indicator selections.

In this example all results have been saved into a particular file each time `mpitb est` has been run. Often it is convenient to assemble a single comprehensive result file. One way to achieve this is to `append` (see [D] **append**) all files as explicitly specified by a list. For appending all files of a folder, see example 5.

```
. clear
. save results/results  , replace emptyok
. loc flist trial01 trial02 health50 educ50 livstd50 ind_equal
. foreach f in `flist´ {
.         append using results/`f´ , nol
. }
. save results/results , replace
```

We now can explore our comprehensive results file as detailed above using basic Stata commands such as `tab` or `list`.

◁

▷ Example 4: Several years for a single country

This section will illustrate how to estimate both levels and changes over time for all measures. Doing so requires repeated surveys for the same country, i.e. at least repeated cross-sectional data. A convenient way to organize such data is to have an identifier for each survey round and to append the micro datasets. In the sample data `syn_cdta.dta` the time variable `t` may be 1 or 2. As before, we first load and `svyset` the data before we specify the our preferred indicators

```
. use syn_cdta.dta , clear
. svyset psu [pw=weight], strata(stratum)
. mpitb set , name(trial01) desc(preferred spec) ///
>                d1(d_cm d_nutr, name(hl)) ///
>                d2(d_satt d_educ, name(ed)) ///
>                d3(d_elct d_wtr d_sani d_hsg d_ckfl d_asst, name(ls))
```

If we were just interested in the estimation of the levels for the above specified measures in both years we could simply add the `tvar(t)` option to the above `mpit est` commands. The toolbox, however, also offers to directly estimate changes over time. The following command, for example, not only estimates the levels but also changes over time for all specified measures, parameters, and level of analysis.

```
. mpitb est , name(trial01) measures(all) klist(1 33 50) weight(equal) ///
>        lframe(myresults, replace) svy over(region) ///
>        cotmeasures(all) cotframe(mycot, replace) tvar(t) cotyear(year)
  (output omitted)
```

Note that this command again stores frames and not files. Moreover, a dedicated frame for change estimates has to be specified as required data structure for saving change estimates somewhat differs. More specifically, a change estimate is characterized by two points of time (a beginning and an end point), contains an absolute or relative change and may, moreover, be annualized (see also Suppa 2022 on this). The option `ann` instructs the toolbox to provide annualized in addition to raw changes. The required information on the duration of the period of observation is obtained from the option `cotyear(`*year*`)`.

The level estimates are now stored in the frame `myresults` and may be conveniently inspected using `list`. To see, for instance, aggregate estimates of the headcount ratio for all k and t, proceed as follows

```
. frame myresults : sort t k
. frame myresults : li measure wgts t k b se if measure == "H" & loa == "nat" //
> /
>          , noobs sepby(t)

  ┌────────────────────────────────────────────┐
  │ measure    wgts   t    k        b       se │
  ├────────────────────────────────────────────┤
          │       H   equal   1    1   0.9575   0.0024 │
          │       H   equal   1   33   0.3352   0.0055 │
          │       H   equal   1   50   0.0818   0.0032 │
          ├────────────────────────────────────────────┤
          │       H   equal   2    1   0.9205   0.0030 │
          │       H   equal   2   33   0.2308   0.0047 │
          │       H   equal   2   50   0.0411   0.0023 │
          └────────────────────────────────────────────┘
```

The frame containing the results for changes over time may be explored in a similar fashion.

```
. frame mycot : li measure wgts ann t0 t1 k ctype b se if measure == "H" ///
>          & loa == "nat" & ann == 0 , noobs sepby(k)

  ┌──────────────────────────────────────────────────────────────────┐
  │ measure    wgts   ann   t0   t1    k   ctype          b       se │
  ├──────────────────────────────────────────────────────────────────┤
  │       H   equal     0    1    2    1     abs    -0.0370   0.0038 │
  │       H   equal     0    1    2    1     rel    -3.8665   0.3960 │
  ├──────────────────────────────────────────────────────────────────┤
  │       H   equal     0    1    2   33     abs    -0.1044   0.0074 │
  │       H   equal     0    1    2   33     rel   -31.1428   1.8322 │
  ├──────────────────────────────────────────────────────────────────┤
  │       H   equal     0    1    2   50     abs    -0.0407   0.0040 │
  │       H   equal     0    1    2   50     rel   -49.7432   3.4710 │
  └──────────────────────────────────────────────────────────────────┘
```

◁

▷ Example 5: A single year for several countries

A cross-country analysis may benefit even more from a careful folder structure. All cleaned micro datasets to be used in the estimation process are assumed to be stored in a dedicated folder which contains nothing else. The present example features three countries and their datasets are stored in the folder `cdta`. Moreover, all outputs shall be stored in the `results` folder.

```
. dir cdta , wide
syn_ABC_cdta.dta  syn_DEF_cdta.dta  syn_GHI_cdta.dta
```

The first step is to compile the so-called *reference sheet* which will contain survey-constant information, such as the survey name (e.g., “DHS”), the year (e.g., “2015-16”) and the names of subnational regions. The reference sheet may be used (i) to control estimation and other productions routines efficiently, (ii) for easily merging external data and (iii) to reduce the amount of information that estimates are passed through the estimation routine. For more information on the reference sheet and its role in the global MPI workflow, see (Suppa 2022).

The tool `mpitb refsh` first extracts the relevant information from each micro dataset and then compiles this information into a single dta-file. `mpitb refsh` expects the path where to find the micro data and an ID how to distinguish different countries. By default, the `id(name)` option expects the name of a variable, but, if option `char(namelist)` is set, also accepts the name of a data characteristic. Data characteristics are more efficient to store information, which is constant for all observations in a given dataset, see [P] **char** for more details on data characteristics. The datasets of this example carry a data characteristic named `ccty` which contains the ISO country code. Additionally, these datasets also feature data characteristics such `survey` and `year`.

```
. clear all
. mpitb refsh using results/refsh.dta, clear id(ccty) sid(region) p(cdta) ///
>         char(ccty ccnum survey year cty)
  (output omitted)
. li ccty region region_name survey year fname in 1/5, noob sepby(ccty)

  ┌──────────────────────────────────────────────────────────────────────────┐
  │ ccty   region       region_name   survey        year              fname  │
  ├──────────────────────────────────────────────────────────────────────────┤
  │  DEF        3    DEF - region 3     MICS   2018-2019   syn_DEF_cdta.dta  │
  │  DEF        8    DEF - region 8     MICS   2018-2019   syn_DEF_cdta.dta  │
  │  DEF       11   DEF - region 11     MICS   2018-2019   syn_DEF_cdta.dta  │
  │  DEF       15   DEF - region 15     MICS   2018-2019   syn_DEF_cdta.dta  │
  │  DEF        2    DEF - region 2     MICS   2018-2019   syn_DEF_cdta.dta  │
  └──────────────────────────────────────────────────────────────────────────┘
```

If issued without the `sid` option, `mpitb refsh` would simply create a file with one observation per country.

By default, `mpitb refsh` reports region codes (`region`) and names (`region_name`) but also stores the file name (`fname`) and time stamps of the micro dataset. Since survey datasets of some countries may not allow to disaggregate by regions, `mpitb refsh` creates a single entry for countries for which the region variable only contains missing values. Note, however, that this variable has to exist in the micro dataset.

It is convenient to have the reference sheet in a frame immediately at hand.

```
. mkf rs
. frame rs: use results/refsh.dta , clear
```

To perform an estimation across countries, first select the countries from the reference sheet using the tool `mpitb ctyselect`, which only expects the name of the variable containing the country codes. By default `mpitb ctyselect` returns all available countries, but one may choose specific subsets using manually specified country codes, world regions, or regular expressions.

The following loop iterates over all of the selected countries, first loading the microdata according to the file name in the reference sheet, svysets the data, specifies the indicators of the MPI and estimates as desired.

```
. frame rs: mpitb ctyselect ccty
. foreach cty in `r(ctylist)´ {
.         frame rs : qui levelsof fname if ccty == "`cty´" , loc(fname) clean
.         use `"cdta/`fname´"´ , clear
.         svyset psu [pw=weight] , strata(stratum) singleunit(centered)
.
.         mpitb set , n(mympi) d1(d_cm d_nutr, n(hl)) ///
>                 d2(d_satt d_educ, n(ed)) ///
>                 d3(d_elct d_wtr d_sani d_hsg d_ckfl d_asst, name(ls))
.
.         mpitb est , name(mympi) measures(all) klist(33) weight(equal) ///
>                 lsa(results/`cty´_results, replace) over(region) ///
>                 svy addmeta(ccty=`cty´)
. }
```

In the cross-country context it is convenient to store results country-wise in files, using the `lsave(filename)` option. Moreover, the `addmeta(`*metalist*`)` option of `mpitb est` allows to store the country-code for each estimate as a meta variable (`ccty`) into the results file. In order to subsequently combine the country-specific files into a single result file, one may simply append all files stored in a single folder and potentially satisfying a particular file name pattern, as illustrated below

```
. clear
. save results/results , replace emptyok
. loc flist : dir "results/" files "*_results.dta"
. foreach f in `flist´ {
.         append using results/`f´ , nol
. }
```

Finally, one may add region names as provided by the reference sheet to the results file as follows:

```
. gen region = subg if loa == "region"
. frlink m:1 ccty region , frame(rs)
. frget region_name , from(rs)
. save results/results.dta , replace
```

Depending on the scale and the specific features of a particular project, it may be preferable to have a `results_raw.dta` which only contains the appended data and a separate more polished `results.dta`, which additionally contains all the labeling as needed for the analysis or deliverable production.

Having a comprehensive cross-country results file allows to easily explore a wealth of data. For example, how do all three countries perform in key measures for the preferred parametrization?

```
. tabdisp ccty measure if loa == "nat" & inlist(k,33,.), cell(b)

----------------------------------
          |         measure
     ccty |      A        H      M0
----------+-----------------------
      ABC | 0.4248   0.3352  0.1424
      DEF | 0.4070   0.2308  0.0940
      GHI | 0.4070   0.2308  0.0940
----------------------------------
```

Drawing on the labeling information collected by `mpitb refsh` now also provides more informative analyses of subnational regions.

```
. tabdisp region_name measure if loa == "region" & inlist(k,33,.) & ccty == "ABC
> " , c(b) l

--------------------------------------------------
               |             measure
 name in c-data |      A        H       M0   popsh
----------------+---------------------------------
ABC - region 1  | 0.4264  0.3654  0.1558  0.0545
ABC - region 10 | 0.4337  0.3764  0.1632  0.0479
ABC - region 11 | 0.4359  0.3079  0.1342  0.0511
ABC - region 12 | 0.4187  0.3235  0.1355  0.0503
ABC - region 13 | 0.4226  0.3103  0.1312  0.0507
ABC - region 14 | 0.4253  0.3536  0.1504  0.0487
ABC - region 15 | 0.4289  0.3198  0.1372  0.0496
ABC - region 16 | 0.4160  0.3362  0.1398  0.0468
ABC - region 17 | 0.4211  0.3190  0.1343  0.0531
ABC - region 18 | 0.4261  0.2882  0.1228  0.0537
ABC - region 19 | 0.4125  0.3631  0.1498  0.0496
ABC - region 2  | 0.4314  0.3162  0.1364  0.0472
ABC - region 20 | 0.4255  0.3256  0.1385  0.0521
ABC - region 3  | 0.4299  0.3500  0.1505  0.0484
ABC - region 4  | 0.4367  0.3010  0.1315  0.0514
ABC - region 5  | 0.4223  0.3432  0.1449  0.0502
ABC - region 6  | 0.4261  0.3664  0.1561  0.0488
ABC - region 7  | 0.4178  0.3220  0.1345  0.0514
ABC - region 8  | 0.4176  0.3380  0.1411  0.0486
ABC - region 9  | 0.4240  0.3900  0.1654  0.0459
--------------------------------------------------
```

◁

5 Concluding Remarks

`mpitb` seeks to facilitate the work of both academics and practitioners of multidimensional poverty measurement. As the toolbox has been developed in the context of the global MPI it is also tailored to its needs, whether in terms of the underlying data, the quantities produced out of the box, or the related forms of analysis. Multidimensional poverty measurement and analysis is, however, also an active field of research, where new measures, analyses and other methodological innovations are still proposed and discussed. `mpitb` may already be useful for such endeavors and take some load off researchers working these topics. Moreover, the very nature of `mpitb` as a toolbox seeks to allow for further features, novel analyses and additional tools being added in the future. `mpitb` is developed on gitlab (https://gitlab.com/nsuppa/mpitb) and published under the MIT license. Queries from experienced problems to feature requests may be reported using the issue tracker.

6 References

Aaberge, R., and A. Brandolini. 2015. Multidimensional Poverty and Inequality. In *Handbook of Income Distribution*, ed. A. B. Atkinson and F. Bourguignon, chap. 3, 141–216. Vol. 2A. Elsevier.

Abdelkrim, A., and J.-Y. Duclos. 2007. DASP: Distributive Analysis Stata

Package. Technical report, PEP, World Bank, UNDP and Université Laval. http://dasp.ecn.ulaval.ca/.

Alkire, S., and J. Foster. 2011. Counting and Multidimensional Poverty Measurement. *Journal of Public Economics* 95(7-8): 476–487. https://www.sciencedirect.com/science/article/abs/pii/S0047272710001660.

Alkire, S., J. Foster, S. Seth, M. Santos, J. Roche, and P. Ballón. 2015. *Multidimensional Poverty Measurement and Analysis: A Counting Approach*. Oxford: Oxford University Press.

Alkire, S., U. Kanagaratnam, and N. Suppa. 2021a. The Global Multidimensional Poverty Index (MPI): 2020. OPHI MPI Methodological Notes 51, Oxford Poverty and Human Development Initiative, University of Oxford.

Alkire, S., R. Nogales, N. N. Quinn, and N. Suppa. 2021b. Global multidimensional poverty and COVID-19: A decade of progress at risk? *Social Science & Medicine* 291: 114457.

Jindra, C., and A. Vaz. 2019. Good governance and multidimensional poverty: A comparative analysis of 71 countries. *Governance* 32(4): 657–675.

Kanagaratnam, U., and N. Suppa. 2022. The global Multidimensional Poverty Index. Technical report, Oxford Poverty and Human Development Initiative. Mimeo.

Pacifico, D., and F. Poege. 2017. Estimating Measures of Multidimensional Poverty with Stata. *The Stata Journal* 17(3): 687–703. http://journals.sagepub.com/doi/10.1177/1536867X1701700309.

Suppa, N. 2022. The production process of the Global MPI. Technical report, Oxford Poverty and Human Development Initiative (OPHI), University of Oxford. Mimeo.

UNDP-OPHI. 2021. The 2021 Global Multidimensional Poverty Index (MPI): Unmasking disparitiesby ethnicity, caste and gender. Technical report, United Nations Development Programme (UNDP) and the Oxford Poverty and Human Development Initiative, New York.

World Bank. 2017. *Monitoring Global Poverty*. Herndon: World Bank Publications. https://ebookcentral.proquest.com/lib/gbv/detail.action?docID=4733152.