Micro Data Infrastructure: Documentation[*]

Eric Bartelsman [†], Ida Dorn [‡], Mirja Haelbig [§] Alessandro Zona Mattioli [¶]

January 15, 2023

[*]

[†]Tinbergen Institute, VU
[‡]Halle Institute for Economic Research (IWH)
[§]Halle Institute for Economic Research (IWH)
[¶]Tinbergen Institute, VU

1 Introduction

This document provides a description of setup, components and usage of the Micro Data Infrastructure (MDI).

The code infrastructure of the MDI consists of general tools (cf. section 3, i.e. functions that do not refer to particularities of the data, and project specific modules (section 2.2.2) which apply these tools to the data. In addition, the MDI consists of country specific metadata that allows users to harmonize the data and help them assess the feasibility of research projects.

In general, the MDI tools and metadata serve two purposes: (i) the harmonization and preparation of the data and (ii) data analyses.

2 Setup

2.1 Data Preparation

Comprehensive metadata, that are machine- and human readable, are the key to data harmonization. Data harmonization pertains to (i) using consistent nomenclature of the variables included in the panel dataset, and (ii) consistent content and format of the variables in the dataset and iii) common classifications for categorical identifiers such as activity, product, or region. Metadata files are available to allow linking of the appropriate datasets to common firm-level panels, concording statistical classifications (e.g. activity or region) to common definitions, and mapping of variables in each country to a common nomenclature and format.

Nomenclature Table 1 shows a machine- and human-readable mapping of the variables from the underlying datasets to a common name to be used by program code of infrastructure users. The tool remapping_var replaces the naming of variables in use at each NSI (NSIname) with a common variable name (MPname). The NSIs, in conjunction with the MDI team, will maintain required metadata.

Codebooks The data harmonization tools map categorical values to a common scheme to which the user refers. For example, table 3 shows possible variable formats (NSIvtype) and native response categories (NSIcatval) in the original dataset shown in table 2. In the process of data harmonization within the MDI, these response categories are mapped to a unified codebook (MPcatval). The tool remapping_var merges the dataset (table 2) and the machine-readable codebook (table 3) on the respective variable values and native variable format (NSIcatval), and replaces the coding in use at each NSI with the common variable format (MPcatval).

Table 1: Remapping of Variable Names

MPname	DataSource	NSIname	Year	Description
firmid	br	ENT_ID	2009	Unique enterprise identification
birthyr	br	start_ent	2009	Start year for the enterprise ID
nace	br	NACE_M	2009	Main activity of the enterprise (NACE 4-digit)
...	...	...	...	
firmid	br	ent_id	2018	Unique enterprise identification
birthyr	br	start_ent	2018	Start year for the enterprise ID
nace	br	nace_m	2018	Main activity of the enterprise (NACE 4-digit)

Notes: Concordance table for mapping NSI specific variable names (column 'NSIname') to a common naming scheme (column 'MPname').

Table 2: Dataset.

firmid	inpssu	rrdin
13769	'0'	1
13879	'1'	0
17640	'9'	1
18000	' '	9
20129	'1'	0
28370	'0'	0
30497	'0'	.
30987	'0'	9

Table 3: Industry Hierarchy.

Source	NSIvtype	MPvtype	NSIcatval	MPcatval
cis	numeric	bool	.	.
cis	numeric	bool	9	.
cis	numeric	bool	0	0
cis	numeric	bool	1	1
cis	character	bool	' '	.
cis	character	bool	'9'	.
cis	character	bool	'0'	0
cis	character	bool	'1'	1

Concordances Program code can be used to remap classifications (e.g. industry, product, region) in use at each NSI into a common classification. The tool remappingClass translates the classifications in use at each NSI into a common classification based on standard concordance tables.

2.2 Data Analyses

2.2.1 Launcher

The launcher is the main file that launches the respective modules stored in project specific folders. The launcher file is the only file adjusted by the statistical institutes (ie define directories and adjust disclosure parameter) before running the codes.

2.2.2 Projects

Projects are user-specific analyses, consisting of modules (scripts/syntax) and additional files that are needed to carry out the analyses. As these files are project specific, they need to be stored in a dedicated project folder.

Modules Modules are user-written codes that apply the tools from the toolbox on the data to carry out specific analyses.

Aggregation Hierarchies The setup allows the user to generate their own custom classification hierarchies. Table 5 shows an example of a non-standard aggregation hierarchy for business activity from the lowest (h_0) to the highest level of aggregation (h_N). h_1 is the parent node of h_0, h_2 the parent node of h_1, etc. Table 4 shows two columns of the original dataset, the firm identifier (firmid) and the firms' industry classiciation (nace - 4-digit NACE code). Table 4 and 5 can be merged on 'nace' and 'h_0'.

Table 4: Dataset.

firmid	nace
13769	5914
13879	2053
17640	2711
...	...
98586	8220

Table 5: Industry Hierarchy.

h_0	h_1	h_2	h_3	h_4
2053	C20-C21	C19-C23	C	TOT
2711	C26-C27	C24-C30	C	TOT
5914	J58-J60	JN	J-S	TOT
8220	M-N	J-N	J-S	TOT
...	...	...	...	...

2.3 Exporting Results

In order to export results from the remote environment, they need to be properly documented and satisfy the respective country's disclosure rules. Therefore, obligatory tools for aggregation, disclosure check and output documentation need to be applied by users.

Disclosure Routine The disclosure routine consists of several steps. First, when aggregating the data to a certain level of aggregation, the aggregation function (fagg) adds two columns to the data: one containing the number of observation underlying the aggregated cell (numObs) and one containing the share of the top X firms in the total of the cell (domPerc).

Once results are exported (export_db), the second part of the disclosure routine is called.
Primary Disclosure: Based on the previously added columns, the disclosure routine replaces values for which the minimum number of underlying observations is less than required by the country's disclosure rules with '-999'. In addition in case of totals, cells for which the

dominance criterion is not fulfilled are replaced by '-999'. *Secondary Disclosure:* Secondary disclosure is applied for totals if only one child node from a parent node is suppressed due to primary disclosure. In that case, another randomly chosen child node is suppressed, too.

Output Documentation In the main file of the user written modules, users need to initiate a text file that features a brief description of the project:

```
write("Here comes a descirption of the project.",
        file = paste0(dirOUTPUT, "OutputDescription.txt"), sep = "\n")
```

The function export_db then adds an entry for each exported data file to the output description file.

3 Tool Library

---

export_db function to export a given data file in a specific format and add an entry to the output description file.

Description
Function to export a given data file in a specific format and add an entry to the output description file. If output_type == 'sum_stat', export_db calls the disclosure function and replaces all cells that do not satisfy the disclosure rule with '-999'. export_db applies primary and - for totals - secondary disclosure.

Usage
```
export_db(output,format,output_name,output_path,desc_file,
               output_type='sum_stat',description=NULL, hhfile)
```

Arguments

output	dataset	the table you want to export
format	character	csv, RDS, txt, dta, xlsx, sas
output_name	character	name of the output file
output_path	character	output directory
desc_file	character	name of the file that describes the output
output_type	character	either 'sum_stat' for summary statistics, 'reg_tab' for regression table or 'other'
description	string	optional; allows further explanation of the output file; if output_type == 'other', please provide a description
hhfile	dataset	hierarchy file required for secondary disclosure

Value

Objects saved in specified directories.

Author

Mirja Haelbig

Examples

```
# export summary statistic in csv format to specified
# output directory dirOUTPUT
export_db(output=sumstat,
              format='csv',
              output_name = 'SummaryStatistic',
              output_path = dirOUTPUT,
              desc_file = "OutputDescription",
              output_type = "sum_stat",
              hhfile = indHier
```

---

fagg function for generic aggregation, from a sub-aggregate level to a aggregate level.

Description

Function for generic aggregation, from a sub-aggregate level to a aggregate level.

Usage

```
fagg(DT, vlist, bygroups, aggtype=c('sum'), weight=NULL,
      mrgflag=FALSE, disclosure=TRUE)
```

Arguments

DT	dataset	name of original data.table
vlist	varlist	vector of variables for aggregation
bygroups	varlist	vector of level(s) of aggregation
aggtype	string	the type of aggregation (options: sum, sd, mean, median, count, HHI), default as sum
weight	varlist	variable to be used for calculating weighted aggregates, default as NULL
mrgflag	boolean	TRUE if the aggtype should be merged as a new variable to the input dataset
disclosure	boolean	if TRUE, dominance criteria and number of observations are merged to the output dataset (only if mrgflag == FALSE)

Value

Returns a data.table as output.

Author

Cindy Jing Chen

Examples

```
# construct summary statistics (mean and total) of variables emp,
# rev, va by nace and year
sumstat <- fagg(DT=BR_SBS,
                vlist=c('emp','rev','va'),
                bygroups=c('nace','year'),
                aggtype=c('mean','sum'),
                disclosure=T)
```

---

fagg_h generic aggregation function which aggregates the variables in vlist to unique values of the dimensions in a hierarchy file.

Description

generic aggregation function which aggregates the variables in vlist to unique values of the hier dimensions in hhfile by groups; wrapper around fagg

Usage

```
fagg_h(DT, vlist, bygroups, hhfile, hier, aggtype="sum",
       aggweight=NULL, mrg=FALSE, disclosure=TRUE)
```

Arguments

DT	dataset	The name of the input dataset
vlist	varlist	List of numeric variables whose values are aggregated; to the nodes of dimensional variables given in aggdims.;
bygroups	varlist	vector of level(s) of aggregation; bygroups must be in DT;
hhfile	dataset	Dataset with columns of hierarchy from h_0 to h_n. hhfile will me merged to DT on h_0 (hhfile) and bygroups[1] (DT);
hier	string	Name of agg hierarchy: single or multiple nodes. If it is character 'h_x' for a single node , aggregate h_0 to h_x through hhfile; if ALL then aggregate h_0 to h_1 to h_n;
aggtype	string	the type of aggregation required (options: sum, sd, mean, median, agg-weighted.mean, count, count.nna), default as sum.
aggweight	varlist	List of variables used to 'weight' the summary operation; as defined by aggtype;
mrg	boolean	Controls merging results of operation into DT.; Default: mrg=F: DTout contains result of operation; mrg=T: DTout merged into DT, by dims.; DTout is deleted.;

Value
Returns a data.table as output.

Author
Cindy Jing Chen

Examples
```
# construct summary statistics (mean and total) of variables emp,
# rev, va by nace (hierarchy level 'h_2') and year
sumstat <- fagg_h(DT=BR_SBS,
                  vlist=c('emp','rev','va'),
                  bygroups=c('nace','year'),
                  hhfile=indHier,
                  hier='h_2',
                  aggtype=c('mean','sum'),
                  aggweight=NULL, mrg=F, disclosure=T)
```

---

import_data function to read a given data file into an R datatable

---

Description
The function import_data reads data into an R datatable. This function is a wrapper around package 'Haven' functions and BASE R functions read.csv,read.table and read.delim. Valid data types are csv, dta, xlsx, sas7bdat, sav and txt.

Usage
```
import_data(dir, file, typeoffile)
```

Arguments
dir	character	directory where input data is stored
file	character	name of the input data file
typeoffile	character	native file format

Value
Returns a R datatable object.

Author
Eric Bartelsman

Examples
```
# import data in csv format from specified input directory dirINPUTDATA
import_data(dirINPUTDATA, 'SBS_2014', 'csv')
```

---

intensity tool for dimension reduction of boolean variables.

Description

The function intensity reduces the dimensionality of boolean variables by calculating the geometric mean of the predicted probabilities.

Usage

```
intensity(dbin,uniqdim,boollist,contlist,fe)
```

Arguments

dbin	dataset	input database
boollist	character	set of boolean indicators
contlist	character	continuous firm-level indicators used as predictors
fe	character	(optional) set of fixed effects, e.g. industry & time (as categorical variables/in factor notation)

Value

Returns original dataset including a column with the intensity indicator 'intens_probit' .

Author

Mirja Haelbig

Examples

```
# calculate 'innovation intensity', using employment (persons_br) and
# industry (MPnace) as predictors
CISintens <- intensity(
  dbin = br_sbs_cis,
  uniqdim = c('firmid','year'),
  boollist = c('inpd','inps','rrdin','mrkin','orgin'),
  contlist = c('persons_br'),
  fe = c('MPnace')
)
```

---

joint_distribution function to calculate joint distributions.

Description

function that calculates joint distributions; calls fagg

Usage

```
joint_distribution(DT, qnames, vnames, moment, bygroups, hhfile, hier,
                   aggtype, prefix=aggtype, aggweight=NULL,
                   mrg=FALSE, disclosure=TRUE)
```

Arguments

DT	dataset	The name of the input dataset
qnames	varlist	vector of variables names for which to calculate distributional moments; is used as additional element in bygroup
vnames	varlist	vector of numeric variables whose values are aggregated bygroup and moments of qnames
moment	string	distributional moment for qnames. can either be 'decile', 'quintile' or 'quartile'
bygroups	varlist	vector of level(s) of aggregation; bygroups must be in DT;
hhfile	dataset	Dataset with columns of hierarchy from h_0 to h_n. hhfile will me merged to DT on h_0 (hhfile) and bygroups[1] (DT);
hier	string	Name of agg hierarchy: single or multiple nodes. If it is character 'h_x' for a single node , aggregate h_0 to h_x through hhfile; if ALL then aggregate h_0 to h_1 to h_n;
aggtype	string	the type of aggregation required (options: sum, sd, mean, median, agg-weighted.mean, count, count.nna), default as sum.
aggweight	varlist	List of variables used to 'weight' the summary operation; as defined by aggtype;
mrg	boolean	Controls merging results of operation into DT.; Default: mrg=F: DTout contains result of operation; mrg=T: DTout merged into DT, by dims.; DTout is deleted.;

Value

Returns a data.table as output.

Author

Mirja Haelbig

Examples

```
# construct summary statistics (mean and total) of variables emp,
# rev, va by nace (hierarchy level 'h_1'), year and quintiles of emp
jd <- joint_distribution(DT=BR_SBS,
                         qnames = c('emp'),
                         vnames = c('emp','rev','va'),
                         moment = 'quintile',
                         bygroups = c('nace','year'),
                         hhfile = indHier,
                         hier = 'h_1',
                         aggtype = c('mean','sum'),
                         disclosure=T)
```

outlier_routine This function runs a specified outlier routine (trimming or winsorizing) and returns the cleaned data

---

Description

This function runs a specified outlier routine (trimming or winsorizing) and returns the cleaned data.

Usage

```
outlier_routine(dbin, varlist, routine, fraction, both_tails=FALSE, group=NULL)
```

Arguments

dbin	dataset	input data
varlist	character	set of continuous variables for the outlier routine
routine	string	can either be "trim" or "winsorize"
fraction	numeric	fraction to be trimmed or winsorized with 0<x<1
both$_{tails}$	boolean	if TRUE, winsorization/trimming/flag are applied on both tails
flag	boolean	if TRUE, observations above the target pctile are flagged
group	character	option to remove outlier within specified group

Value

returns the cleaned data

Author

Alessandro Zona Mattioli

Examples

```
br_sbs <- outlier_routine(dbin=br_sbs,
                          varlist=c('emp','nq'),
                          routine='winsorize',
                          fraction=.01,
                          group=c('h_2','year'))
```

---

remapping_var creates a remapping between NSI variable names and harmonized variable names, NSI specific coding of variables and harmonized coding.

Description

creates a remapping between NSI variable names and harmonized variable names, NSI specific coding of variables and harmonized coding.

Usage

```
remapping_var(DT, MPnames_remap, MPnames_select , MPcodebook, ds, year)
```

Arguments

DT	dataset	original data.table with only NSIvarname
MPnames_remap	dataset	MetaData file that saves information on MPvarname, NISvarname, datasource, ...etc.
MPnames_select	dataset	MetaData file that includes the selected variables
MPcodebook	dataset	Metadata file that includes concordances
ds	string	a string variable and includes the file prefix, i.e. br or ofats, ...
year	numeric	the year to which the data pertain

Value

returns dataset after harmonizing variable names, coding and type

Author

Cindy Jing Chen, Mirja Haelbig, Alessandro Zona Mattioli

Examples

```
remapping_var(
        DT = SBS,
        MPnames_remap = MPnames_remap,
        MPnames_select = MPnames_select ,
        MPcodebook = MPcodebook,
        ds = sbs ,
        year = 2013
      )
```

---

remappingClass tool to harmonize classifications.

Description

creates a remapping between NSI classifications and a harmonized classification scheme.

Usage

```
remappingClass(DT, conc, nativeClassDT, nativeClassConc, targetClass)
```

Arguments

DT	dataset	Data.table including the nativeClassDT
conc	dataset	Concordance table mapping nativeClassDT to targetClass
nativeClassDT	character	name of native classification in DT
nativeClassConc	character	name of native classification in concordance table
targetClass	character	name of target classification in concordance table

Value

returns dataset with the target classification

Author

Mirja Haelbig

Examples

```
DT <- remappingClass(DT=BR,
                     conc=indRemap,
                     nativeClassDT = 'nace',
                     nativeClassConc = 'NSIind',
                     targetClass = 'MPind')
```