MIND STEP

MODELLING INDIVIDUAL DECISIONS TO SUPPORT THE EUROPEAN POLICIES RELATED TO AGRICULTURE

Deliverable D2.4: Prototype for Interfaces

AUTHORS	Gocht, Alexander (THÜNEN) Neuenfeldt, Sebastian (THÜNEN)
APPROVED BY WP MANAGER:	Gocht, Alexander (THÜNEN)
DATE OF APPROVAL:	30.04.2021
APPROVED BY PROJECT COORDINATOR:	Hans van Meijl (WR)
DATE OF APPROVAL:	10.05.2021
WP NUMBER & TITLE	WP 2 Data requirements for indicators on European policies related to agriculture and data management
PROJECT WEB SITE:	https://mind-step.eu

This document was produced under the terms and conditions of Grant Agreement No. 817566 for the European Commission. It does not necessary reflect the view of the European Union and in no way anticipates the Commission's future policy in this area.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

TABLE OF CONTENTS

LIST OF TABLES

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

ACRONYMS

AgroDataCube	AgroDataCube provides a large collection of both open data and derived data for use in agri-food applications
API	Application Programming Interface
FADN	Farm Accountancy Data Network
fadnUtils	R package to easily load and manipulate FADN data
FSS	Farm Structure Survey
GLOBIOM	Global Biosphere Management Model
globiomvis	R package assists with visualizing GLOBIOM data
IIASA	International Institute for Applied Systems Analysis
Mapsspam2globiom	R package to facilitate the creation of country level crop distribution maps, which can be used as input by the IIASA's Global Biosphere Management Model (GLOBIOM)
URL	Uniform Resource Locator

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

1. INTRODUCTION

This deliverable lists the R packages that are developed to be used as interfaces for different databases of the models of MIND STEP. The packages are work in progress and are will to be updated, improved or new packages are developed at a later stage of this project. The appendix lists the documentations of the fadnUtils and FSS packages. For the remaining packages, i.e. Mapsspam2globiom and globiomvis, a more detailed description of the package and their functions can be found behind the given URLs.

2. OVERVIEW OF R PACKAGES FOR INTERFACES OF DATABASES

In this chapter the developed packages are listed in Table 1. We also provide the URL of the packages, the database characteristic and in which chapter of the deliverable D2.2 a broader description can be found. The package fadnUtils is an interface to work with farm accountancy data, e.g. FADN. The FSS packages contains functions to work with (German) farm structure survey data (FSS). For the bio-physical database AgroDataCube, a R packages was not developed so far. Therefore, a link to the web page is provided in which a documentation of the database itself and the API is given. The packages Mapsspam2globiom and globiomvis provide an interface for the GLOBIOM model.

Table 1: Overview of R packages for interfaces of databases

Name of package	URL link to package	Database characteristic	Link to Document
fadnUtils	https://git-dmz.thuenen.de/mindstep/fadnutilspackages	Economic databases 2.4	D2.2 Chapter 4
FSS	https://git-dmz.thuenen.de/mindstep/fss	Economic databases 2.4	D2.2 Chapter 4
*	https://agrodatacube.wur.nl/	Bio-physical databases 2.5	D2.2 Chapter 4
Mapsspam 2globiom	https://iiasa.github.io/mapspam2globiom/	Current models 2.6	D2.2 Chapter 2 and 4
globiomvis	https://iiasa.github.io/globiomvis	Current models 2.6	D2.2 Chapter 2 and 4

*Note: * - no specific package as interface developed so far.*

Source: Own compilation.

3. ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 817566. Neither the European Commission nor any person acting on behalf of the Commission is responsible for how the following information is used. The views expressed in this publication are the sole responsibility of the author and do not necessarily reflect the views of the European Commission.

APPENDIX

3.1. Package 'fadnUtils'

Package 'fadnUtils'

April 26, 2021

Title An R package to easily load and manipulate FADN data

Type Package

Version 1.0.2

Author Dimitris Kremmydas, Xinxin Yang

Maintainer Dimitris Kremmydas <Dimitrios.KREMMYDAS@ec.europa.eu>

Description Manipulate and perform data analysis with FADN data

License Proprietary software (JRC D.4)

Encoding UTF-8

LazyData TRUE

Depends R (>= 3.4.0)

Imports data.table, jsonlite

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

R topics documented:

1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

`analyzeFormula` *Dissagregates a string formula to a list(add=c("SE610","J830(2)","#289","#267..270"),substract=c("SE626","M632..634(2)"))*

Description

Dissagregates a string formula to a list(add=c("SE610","J830(2)","#289","#267..270"),substract=c("SE626","M632..634

Usage

```
analyzeFormula(formula)
```

Arguments

`formula`	a formula string, see examples

Value

list(add=c(),substract=())

Examples

```
formula="K120..148(7)+K120..148(8)+K120..148(9)+K120..148(10)-K120..148(6)"
formula="#48+#49+#50"
```

`check.column` *Check the variables/column names for calculating the aggregate variables*

Description

The check.column function checks the variables if they exist in a json-file matching the variables in the fadn.raw.rds or fadn.raw.csv (csv-file from FADN-AGRI), returning a list of variables which are not in the raw data file. Then a new json file without unmatched variables can be saved in the extraction_dir. A txt-file (my_logfile.txt) is created in a specific directory (spool.dir) where stores the output messages.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Usage

```
check.column(importfilepath, jsonfile, rewrite_json = TRUE, extraction_dir)
```

Arguments

importfilepath	A fadn.raw.rds or fadn.raw.csv file address.
jsonfile	A json file address.
rewrite_json	Logical, if TRUE (default), a new json file without unmatched variables will be saved. The string "rewrite" will be added in front of the original file name, and they are separated through "_". For example, the name of original json file is A.json, the new json file will be saved as rewrite_A.json. Otherwise, do not rewrite json file.
extraction_dir	Extraction_dir is the folder for extracting the data.

Details

If variables exist in a json-file and not in the fadn.raw.rds file or fadn csv file, then returning all unmatched variables. Json file has 6 objects/categries: "id", "info", "costs", "crops", "subsides", "livstock".

Value

A list of multiple objects. The objects are in the json-file, which have the unmatched variables.

Author(s)

Xinxin Yang <xinxin.yang@thuenen.de>

Examples

```
check.column("./fadn.raw.2009.BEL.rds", "./2014_after.json", TRUE, "./OV")
check.column("BEL2009.csv", "2013_before.json", TRUE, "./OV")
```

`check.data.dir.structure`

Checks if the structure of the fadnUtils.data.dir is ok

Description

Checks if the structure of the fadnUtils.data.dir is ok

Usage

```
check.data.dir.structure(data.dir = NULL, silent = T)
```

Arguments

data.dir	a specific directory to show contents, otherwise it will read the fadnUtils.data.dir
silent	if TRUE, do not print any message

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Value

TRUe if everything is ok; FALSE otherwise

`check.raw_str_map` *Checks if the definitions of a raw_str_map are compatible with a fadn.raw.rds for a certain year and country*

Description

Checks if all values are actual columns of the fadn.raw.rds file

Usage

```
check.raw_str_map(raw_str_map.file, fadn.country = NA, fadn.year = NA)
```

Arguments

`raw_str_map.file`
: The full filepath of the raw_str_map

`check_file_type` *Check the type of load file*

Description

This function checks the type of the load file and read this file. If the file is not a csv or rds file, the execution of the currently running R code will be stopped.

Usage

```
check_file_type(filepath)
```

Arguments

`filepath` A rds or csv file address.

Value

A data frame with cases corresponding to lines and variables to fields in the file.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

`collect.common.id` *Collect Common id*

Description

Load the Fadn.raw.rds data (Data Table) or Fadn.str.rds data (List), then collection the common id from different years on this data.

Usage

```
collect.common.id(my.r.data)
```

Arguments

`my.r.data`	A data object(either a data.table or a list).

Value

A data.table, it includes just one column that named "common_id".

Author(s)

Xinxin Yang

Examples

```
collect.common.id(fadn.raw.rds)
## collection the common "id" from the raw rds data
## for 2009-2012 years and country "BEL".
## Return a DT with one column named "common_id".
```

`convert.to.fadn.raw.rds` *Gets a fadn.raw.csv (csv file from DG-AGRI) and transforms it accordingly to fadn.raw.rds*

Description

It saves two files: - One that contain a wide format of the data, i.e. in tabular format that is identical to the csv data. This is uncompressed data. - One that holds the same information in compressed data. It is a list that contains $data.char and $data.num data.tables in long format. 0 values are removed and only the col.id is the index on both data.tables

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Usage

```
convert.to.fadn.raw.rds(
  file.path = "",
  sepS = ",",
  fadn.year = NA,
  fadn.country = NA,
  keep.csv = F,
  col.id = "ID"
)
```

Arguments

file.path	the full path of the csv file (the filename must be included)
sepS	the separator of the csv files (by default ",")
fadn.year	the year the csv files refers to (e.g. 2001)
fadn.country	the three letter country code the csv files refers to (e.g. "ELL")
keep.csv	if TRUE, copy the csv files to the CSV directory; else do not copy

Value

Saves the fadn.raw.rds file and returns TRUE if everything goes well

---

`convert.to.fadn.str.rds` *Converts an fadn.raw.rds file to fadn.str.rds file using a raw_str_map.json file*

Description

The raw_str_map.json specification is as follows:

Usage

```
convert.to.fadn.str.rds(
  fadn.country = NA,
  fadn.year = NA,
  raw_str_map.file = NULL,
  force_external_raw_str_map = FALSE,
  str.name = NULL,
  DEBUG = F
)
```

Arguments

fadn.country	string with the country to extract the str data
fadn.year	the year to extract the structured data
raw_str_map.file	the full path to the raw_str_map file.
DEBUG	if TRUE, prints more details on the conversion process
str.short_name	the short name of the str data. No spaces and text up to 20 characters

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Details

"id": "COLUMN in every list member in RDS": "COLUMN IN CSV", ..., "info": "COLUMN in info RDS": "COLUMN IN CSV", ..., "livestock": "crops": "CROP NAME 1": "description": "description of crop name", "columns": "VARIABLE NAME": COLUMN IN CSV", ... , "CROP NAME 2": "description": "description of crop name", "columns": "VARIABLE NAME": COLUMN IN CSV", ... , ...

The structure of the str.dir: - A data.dir can hold more than one extractions. - Each extraction has a short name (20 or less characters, whitespace is not allowed) - Each extraction is stored in the data.dir/rds/<extraction_name> - That folder contains the following files: + raw_str_map.json: the raw_str_map + fadn.str.<4-digit YEAR>.<3-letter COUNTRY>.rds: the extracted data

Notes: 1) The computed RDS file contains a list structure with the following keys: info, costs, livestock-animals and crops All are data.tables. For all of them, the first columns are those that are contained in the "id" object "info" and "costs" are in table format, i.e. each farm is one row and data is on columns, as defined in the related raw_str_map.json file. "crops" and "livestock-animals" are in wide data format (https://tidyr.tidyverse.org/), where one farm lies accross many rows, and each row is a farm-crop-variableName-value combination

2) In $id, $info and $costs, "COLUMN IN CSV" can have two forms i) a single column name in the fadn.raw csv file or a combination, e.g. "K120SA+K120FC+K120FU+K120CV-K120BV" ii) the form of an object "source": "the column in the csv", "description": "a description of what this column is about"

3) We attach certain attributes that are useful for identifying informations: i) In $info and $costs, the attribute "column description" provide information of the formula and the description of each column ii) In $crops and $livestock-animals, the attribute "$crops.descriptions" and "$livestock.descriptions", provide the description of each CROP contained there iii) In $crops and $ the attribute "$column.formulas" provide the formulas used in order to derive the VALUE

Value

Saves the rds.str.fadn and returns TRUE if everything goes well

---

`create.data.dir` *Creates a data.dir*

Description

Creates a data.dir

Usage

```
create.data.dir(
  folder.path,

   metadata = "{\n'description': 'No Description Provided',\n'created-by':'',\n'created-at':''\n}"
)
```

Arguments

`metadata`

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Value

TRUE if created succesfully; FALSE otherwise. It return in invisible mode.

`delete.fadn.raw` *Title*

Description

Title

Usage

```
delete.fadn.raw(countries = NULL, years = NULL)
```

Arguments

years

`delete.fadn.str` *Title*

Description

Title

Usage

```
delete.fadn.str(countries = c(), years = c())
```

Arguments

years

`get.available.fadn.raw.rds`
Returns the available YEAR-COUNTRY fadn.raw.rds

Description

Returns the available YEAR-COUNTRY fadn.raw.rds

Usage

```
get.available.fadn.raw.rds(data.dir = NULL)
```

Value

a DT of the available YEAR-COUNTRY fadn.raw.rds

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

`get.available.fadn.str.rds` *Returns the available YEAR-COUNTRY fadn.str.rds, for each str.folder*

Description

Returns the available YEAR-COUNTRY fadn.str.rds, for each str.folder

Usage

```
get.available.fadn.str.rds(data.dir = NULL, extract_dir)
```

Arguments

`extract_dir`	The name of the extraction dir

Value

DT of the available YEAR-COUNTRY fadn.str.rds

`get.data.dir` *Gets the data.dir*

Description

data.dir is the folder where data is stored r package will create two subfolders: csv = location to store the csv files of th DG-AGRI (fadn.raw.csv) rds = location to store rds files (fadn.raw.rds, fadn.str.rds, etc.)

Usage

```
get.data.dir()
```

Value

the value of option("fadnUtils.data.dir")

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

`getFormulaResult` *Aggregates columns for each farms using a formula*

Description

Aggregates columns for each farms using a formula

Usage

```
getFormulaResult(data, SEdata, formulaString, aggregator = sum, onlyValue = T)
```

Arguments

`data`	a fadn.container, containing all tables
`SEdata`	a data.table of already calculated SE
`formulaString`	The formula String to use for aggregation

Value

FID VALUE

Examples

```
#definition of formula SE610+SE615+SE624-SE626
formula=list(add=c("SE610","J830(2)","#289","#267..270"),substract=c("SE626","M632..634(2)"))
list(add=c("#48","#49","#50"),substract=list())
```

`grep.columns.in.raw.rds`
Grep a pattern into a raw.rds column names

Description

Useful for the case where one want to look if there are certain columns present or missing

Usage

```
grep.columns.in.raw.rds(pattern, countries = c("all"), years = c("all"))
```

Arguments

`pattern`	a grep-like character pattern. This parameter is passed as is to the grep function
`countries`	a character vector with all the 3-letter codes of the selected countries, e.g. c("ELL", "ESP"). If "all" is included, all available countries are loaded
`years`	a numeric vector with the years selected. If "all" is included, all available years are loa
`show`	if TRUE, the columsn are printed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Value

Prints the columns and returns them invisibly

`import.fadn.csv` *Imports a DG-AGRI csv into fadnUtils*

Description

It first call the convert.to.fadn.raw.rds and then convert.to.fadn.str.rds

Usage

```
import.fadn.csv(
  file.path,
  raw.f = NULL,
  sepS = ",",
  fadn.year = NA,
  fadn.country = NA,
  keep.csv = F
)
```

Arguments

`file.path`	the full path of the file (the filename must be included)
`raw.f`	the raw_str_map file to use. it must reside inside 'raw_str_maps; folder of the data.dir
`sepS`	the separator of the csv files (by default ",")
`fadn.year`	the year the csv files refers to (e.g. 2001)
`fadn.country`	the three letter country code the csv files refers to (e.g. "ELL")
`keep.csv`	if TRUE, copy the csv files; else do not copy

`load.fadn.raw.rds` *Load all rds.raw.FADN data for selcted years and countries (rbinds them)*

Description

It adds two columns: load.YEAR and load.COUNTRY in each row. This can be used to group per year,country the data

Usage

```
load.fadn.raw.rds(
  countries = c("all"),
  years = c("all"),
  col.filter = NULL,
  row.filter = NULL
)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Arguments

countries	a character vector with all the 3-letter codes of the selected countries, e.g. c("ELL", "ESP"). If "all" is included, all available countries are loaded
years	a numeric vector with the years selected. If "all" is included, all available years are loaded
col.filter	a character vector with the columns to load. If NULL, all columns are loaded. E.g columns=c('ILOTH_VET_V', 'ILVOTH_V','id')
row.filter	a string giving an expression that will be evaluated in order to select rows. If NULL, all rows are returned. E.g. filter='TF8==1'

Value

list("countries"=> c(<RETURNED COUNTRIES), "years"=>c(<AVAILABLE YEARS))

`load.fadn.str.rds` *Load all rds.str.FADN data for seelcted years and countries*

Description

Load all rds.str.FADN data for seelcted years and countries

Usage

```
load.fadn.str.rds(extraction_dir, countries = c("all"), years = c("all"))
```

Arguments

countries	a character vector with all the 3-letter codes of the selected countries, e.g. c("ELL", "ESP"). If "all" is included, all available countries are loaded
years	
str.name	The extractionname to load data from

Value

list("countries"=> c(<RETURNED COUNTRIES), "years"=>c(<AVAILABLE YEARS))

`nested_var` *Check a objest in the json file*

Description

This function checks the node of chosen object/category for the json file and find out the variables which are in json file but not in fadn.raw data file. Returning two lists: unmatched variables/column names and modified json. If unmatched variable exists, this variable will be deleted from the json list.

Usage

```
nested_var(var, rds)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Arguments

var	A object or category of raw json.
rds	All variables/column names in fadn.raw.rds file.

Details

A json file has 6 parent objects/categories: "id", "info", "costs", "crops", "subsides", "livstock". This function checks all objects inside the parent object.

Value

A list of multiple objects. This list combines no machted variables and the modified json for the chosen object/category.

Author(s)

Xinxin Yang

`raw_str_map.merge` *Merges two raw_str_map files and returns either a list or a file*

Description

All entries in the new.raw_str_map file replace those on the source.raw_str_map file

Usage

```
raw_str_map.merge(
  source.raw_str_map.file = NULL,
  new.raw_str_map.file = NULL,
  return.file = F
)
```

Arguments

source.raw_str_map.file	the filename of the source raw_str_map. It must be relative the raw_str_maps of the current data.dir
new.raw_str_map.file	the filename of the mask raw_str_map. It will replace any entries of the source file. It must be relative the raw_str_maps of the current data.dir
return.file	If set to T, a temporary full file path that contains the merge is returned. Otherwise a list with the contents of the merge is returned

Details

Both files must be relative to the current data.dir/raw_str_maps

Value

FALSE in case of problem / if return.file=T, the temporary full path of a file that contains the merged result in json / A list with the contents of the merge if return.file=F

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

`set.data.dir` *Sets the data.dir*

Description

Sets the data.dir

Usage

```
set.data.dir(new.data.dir)
```

Arguments

`new.data.dir`	the full path to the folder where the data.dir will be. Ending slash "/" shall not be present

Value

TRUE if succesfully set the data.dir; FALSE otherwise. Returns in invisible mode.

`show.data.dir.contents` *Show the contents of data.dir*

Description

Show the contents of data.dir

Usage

```
show.data.dir.contents(data.dir = NULL, return.list = F)
```

Arguments

`data.dir`	a specific directory to show contents, otherwise it will read the fadnUtils.data.dir
`return.list`	if T, returns a list, otherwise print the results

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

`take.raw_str_map.columns` *Takes $id, $info, $costs objects of a raw_str_map object and create Source-Description pairs*

Description

Used internally

Usage

```
take.raw_str_map.columns(listcontent)
```

Arguments

`listcontent`

Value

list(COLUMN-NAME = c(SOURCE=csv column name, DESCRIPTION=description of column), ...)

`update_elements.DT` *Updates selected elements of data stored in one DT with new one given in melted format*

Description

The user provides the data.new: id,variable,new value. The function overwrites all existing id-column with the new values

Usage

```
update_elements.DT(data.old, data.new)
```

Arguments

`data.old`	The DT to update
`data.new`	The data to insert. It must have three columns: id,variable,new value. E.g. data.new=data.table("id"=c(810001100105),"variable"=c("AASBIO_CV"),value=c(999999))

Value

a DT with the updated values

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

`write.excel` *Utility to copy data to clipboard for pasting to Excel*

Description

Utility to copy data to clipboard for pasting to Excel

Usage

```
write.excel(d, getRownames = F, ...)
```

Arguments

`d`	the data to copy
`getRownames`	set to T to opy also row.names
`...`	any other parameter for passing to write.table

Value

nothing

Examples

```
write.excel(d);
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Index

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

3.2. Package 'FSS'

Package 'FSS'

April 23, 2021

Type Package

Title A package for preparing (German) Farm Structure Survey (FSS) data for analysis.

Version 0.1.0

Author Sebastian Neuenfeldt

Maintainer Sebastian Neuenfeldt <sebastian.neuenfeldt@thuenen.de>

Description The FSS package is written for the German Farm Structure Survey data as it was provided in the year 2018. This means, that the RDC provides to data sets.
One is the old data set which has variables in the former declination (EF codes) and contains data at maximum data from 1999, 2003 and 2007.
The second data set is declinated in C/C0 codes and has data from 2010, 2013 (only sample), 2016 and 2020 (as of 2021).
How much variables the researcher has requested or how many years depends. This function works fine for all years and all variables up to 2016.

License GPL (>= 3)

Imports data.table

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Collate 'convertCSVtoRdata.R'
'FSS.R'
'generateFakeFSSData_DE.R'

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

`convertCSVtoRdata_DE` *This function converts the given RDC comma seperated value files into Rdata. It is important to verify before if the national RDC provides the data in the form as desired by this function.*

Description

This function converts the given RDC comma seperated value files into Rdata.

Usage

```
convertCSVtoRdata_DE(
  datafiles.dir = NULL,
  intermediate.dir = NULL,
  filename.old = NULL,
  filename.new = NULL
)
```

Arguments

`datafiles.dir`	Directory of the raw data files
`intermediate.dir`	Destination directory of converted Rdata
`filename.old`	Names of the raw data files (without file type)
`filename.new`	Names of the Rdata data files (without file type)

Details

This function is written for the German Farm Structure Survey data as it was provided in the year 2018.This means, that the RDC provides to data sets. One is the 'old' data set which has variables in the former declination (EF codes) and contains data at maximum data from 1999, 2003 and 2007. The second data set is declinated in C/C0 codes and has data from 2010, 2013 (only sample), 2016 and 2020 (as of 2021).

How much variables the researcher has requested or how many years depends. This function works fine for all years and all variables up to 2016.

This function needs of course the data as well as a specific folder structure, at least the RDC data file names and the specific folder names where these files are located and where they should be exported as Rdata files.

Value

Nothing returned, but Rdata exported to destination folder.

Author(s)

Sebastian Neuenfeldt

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Examples

```
## Not run:
convertCSVtoRdata_DE(datafiles.dir="D:/data/in/",intermediate.dir="D:/data/temp/",
   filename.old="Panel_old",filename.new="Panel_new")

## End(Not run)
```

`FSS` *FSS: A package for preparing (German) Farm Structure Survey (FSS) data for analysis.*

Description

The FSS package is written for the German Farm Structure Survey data as it was provided in the year 2018.This means, that the RDC provides to data sets. One is the 'old' data set which has variables in the former declination (EF codes) and contains data at maximum data from 1999, 2003 and 2007. The second data set is declinated in C/C0 codes and has data from 2010, 2013 (only sample), 2016 and 2020 (as of 2021).

Details

How much variables the researcher has requested or how many years depends. This function works fine for all years and all variables up to 2016.

`generateFakeFSSData_DE` *This function provides a fake sample data set which has the form of the German FSS data. It is important to verify before if the national RDC provides the data in the form as desired by this function to have a proper fake data set.*

Description

This function provides a fake sample data set which has the form of the German FSS data.

Usage

```
generateFakeFSSData_DE(
  nobs = 270000,
  years = c(1999, 2003, 2007, 2010, 2013, 2016, 2020),
  C0codes = NULL
)
```

Arguments

`nobs`	Number of observations approximately to be generated
`years`	Years of survey
`C0codes`	Optional variables to be generated, only meaningful for continues variables.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Details

This function is written for the German Farm Structure Survey data as it was provided in the year 2021.This means, that the generated data will be in a form that fits to the variables that are used from 2010 onwards - C0 codes.

In its basic form this function generates data for the years 1999, 2003, 2007, 2010, 2013, 2016 and 2020. For 2013, it is only a sample of the population. The automatically generated variables comprise 4 regional variables, 7 general variables and 7 production based variables.

Regional variables:

- C0010U1: NUTS1
- C0010UG5: NUTS2
- C0010UG4: NUTS3
- AGS: LAU

The regional variables are reasonable, but far away from correct numbers.

General variables:

- C0008U1: year of survey
- nr: farm id
- C0072: weighting factor - generated also for non-sample farms - only relevant for sample farms - weighted sum of a specific variable does not lead to the population sum!
- C0025: "N" population or "S" sample farm
- C0041: legal status - single farm, unincorporate farm (both as private farm) and corporate farm
- C0045: 1 full-time farm, 2 part-time farm, NA neither
- C0060UG1: farm type - aggregated to some relevant farm types in Germany

Production variables:

- C0240: total utilized agricultural area
- C0231, C0232, C0233, C0234: grass land activities
- C0210: arable land

These variables are coherent as grass land and arable land sum up to total land.

Any additional variables provided via C0codes argument are not coherent to these production variables.

Value

Retruns a fake data set based on German FSS data.

Author(s)

Sebastian Neuenfeldt

Examples

```
## Not run:
FSS_data_DE <- generateFakeFSSData_DE()

## End(Not run)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.

Index

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 817566.