Package 'FAOSTAT'

June 2, 2015

Type Package

Title Download Data from the FAOSTAT Database of the Food and Agricultural Organization (FAO) of the United Nations

Version 2.0

Date 2015-05-28

Author Michael C. J. Kao <michael.kao@fao.org>, Markus Gesmann, Filippo Gheri

Maintainer Filippo Gheri <filippo.gheri@fao.org>

Description A list of functions to download statistics from FAOSTAT (database of the Food and Agricultural Organization of the United Nations) and WDI (database of the World Bank), and to perform some harmonization operations.

URL https://github.com/mkao006/FAOSTATpackage

Imports RJSONIO (>= 0.96-0), plyr (>= 1.7.1), data.table (>= 1.8.2), MASS (>= 7.3-22), classInt (>= 0.1-19), ggplot2 (>= 0.9.3), labeling (>= 0.1), scales (>= 0.2.3)

License GPL (>= 2)

LazyData yes

ZipData no

VignetteBuilder knitr

Suggests knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2015-06-02 00:06:07

R topics documented:

`FAOSTAT-package` *A complementary package to the FAOSTAT database and the Statistical Yearbook of the Food and Agricultural Organization of the United Nations.*

Description

A complementary package to the FAOSTAT database and the Statistical Yearbook of the Food and Agricultural Organization of the United Nations.

Author(s)

Michael. C. J. Kao <michael.kao@fao.org>

Aggregation *Compute Aggregates*

Description

The function takes a relational data frame and computes the aggregation based on the relation specified.

Usage

```
Aggregation(data, aggVar, weightVar = rep(NA, length(aggVar)),
  year = "Year", relationDF = FAOcountryProfile[, c("FAOST_CODE",
  "M49_FAOST_CODE")], aggMethod = rep("sum", length(aggVar)),
  applyRules = TRUE, keepUnspecified = TRUE, unspecifiedCode = 0,
  thresholdProp = rep(0.65, length(aggVar)))
```

Arguments

aggVar	The vector of names of the variables to be aggregated.
weightVar	The vector of names of the variables to be used as weighting when the aggregation method is weighted.
year	The column containing the time information.
data	The data frame containing the country level data.
relationDF	A relational data frame which specifies the territory and the mother country. At least one column must have a corrispondent variable name in the dataset.
aggMethod	Can be a single method for all data or a vector specifying different method for each variable. The method can be "sum", "mean", "weighted.mean".
unspecifiedCode	The output code of the unspecified group.
thresholdProp	The vector of the missing threshold for the aggregation rule to be applied. The default is set to only compute aggregation if there are more than 65 percent of data available (0.65).
applyRules	Logical, specifies whether the thresholdProp rule must be applied or not.
keepUnspecified	Whether countries with unspecified region should be aggregated into an "Unspecified" group or simply drop. Default to create the new group.

Details

The length of aggVar, aggMethod, weightVar, thresholdProp must be the same.

Aggregation should not be computed if insufficient countries have reported data. This corresponds to the argument thresholdProp which specifies the percentage which of country must report data (both for the variable to be aggregated and the weighting variable).

Examples

```
## example.df = data.frame(FAOST_CODE = rep(c(1, 2, 3), 2),
##                         Year = rep(c(2010, 2011), c(3, 3)),
##                         value = rep(c(1, 2, 3), 2),
##                         weight = rep(c(0.3, 0.7, 1), 2))

## Lets aggregate country 1 and 2 into one country and keep country
## 3 seperate.
## relation.df = data.frame(FAOST_CODE = 1:3, NEW_CODE = c(1, 1, 2))
```

`chConstruct` *Construct year to year change*

Description

A function for constructing year to year change

Usage

```
chConstruct(data, origVar, country = "FAOST_CODE", year = "Year",
  newVarName = NA, n = 1)
```

Arguments

`origVar`	The variable in which the year to year change is to be calculated
`country`	The column representing the index of country.
`year`	The column represing the index of year.
`data`	The data frame containing the data
`newVarName`	The name assigned to the new variable, if missing then .CH will be appended.
`n`	The period for the change rate to be calculated.

Value

A data frame containing the computed year to year change rate.

chgr *Absolute change between the year*

Description

Function for generating the n-period absolute change

Usage

```
chgr(x, n = 1)
```

Arguments

x	The time series for the change to be calculated.
n	The period for the growth to be calculated over.

Details

In order to ensure the change calculated is reliable, the following rule are applied.

1. 50% of the data must be present.
2. The length of the time series must be greater than n

Otherwise the growth will not be computed.

Value

The n-period change of the time series.

Examples

```
test.ts = abs(rnorm(100))
chgr(test.ts, 1)
chgr(test.ts, 3)
chgr(test.ts, 10)
```

CHMT *This function avoids double counting of China.*

Description

This function should only be used when performing aggregations.

Usage

```
CHMT(var, data, year = "Year")
```

Arguments

var	The variables that require to be sanitized.
data	The data frame which contains the data
year	The column which correspond to the year.

Details

We decide to use the smaller subsets in the regional level because weighting variable may not exist for other variables for the larger subsets.

The function only work for FAOST_CODE, if the country coding system is not in FAOST_CODE then use the translateCountryCode function to translate it.

constructSYB *Construct/Creat new variable.*

Description

A function used to construct new variables from existing variables.

Usage

```
constructSYB(data, origVar1, origVar2, newVarName = NA,
  constructType = c("share", "growth", "change", "index"), grFreq = 1,
  grType = c("ls", "geo"), baseYear = 2000)
```

Arguments

data	The data frame containing the raw variable
origVar1	The variable name to be used in construction, refer to Details for more information and useage.
origVar2	The variable name to be used in construction, refer to Details for more information and useage.
newVarName	The name assigned to the new variable, if missing then .SC/.SH/.GR/.CH will be appended depending on the type of construction
constructType	The type of construction, refer to Details for more information.
grFreq	The frequency for the growth rate to be computed.
grType	The method for the growth to be calculated, currently supports least squares and geometric.
baseYear	The base year to be used for constructing index.

Details

Currently two types of construction are supported, either share or growth rate computation.

Share can be a share of total or share of another variable depending on whether an additional variable is supplied or not.

Value

A data frame containing both the original data frame and the processed data and also a list indicating whether the construction passed or failed.

---

ebind *A function to bind the different entity level.*

Description

A data frame is chosen over the list is solely for the purpose of transition to ggplot2.

Usage

```
ebind(territory = NULL, subregion = NULL, region = NULL, world = NULL)
```

Arguments

territory	The data frame which contains the territory/country level data
subregion	The sub aggregated region aggregate
region	The macro region aggregate
world	The world aggregate

FAOcheck *This function perform some check on the data*

Description

The function only works for FAOST_CODE. If the country coding system is not in FAOST_CODE then use the translateCountryCode function to translate it.

Usage

```
FAOcheck(var, year = "Year", data, type = c("overlap", "multiChina"),
  take = c("simpleCheck", "takeNew", "takeOld", "complete"))
```

Arguments

var	The variable to be checked.
year	The column which index the time.
data	The data frame.
type	The type of check.
take	The type of check/replacement to be done in case of type equals to overlap.

Examples

```
## test.df =
##    data.frame(FAOST_CODE = rep(c(51,167,199), each = 3),
##      Year = rep(c(1990:1992), 3),
##      Value = c(c(3,4,4), c(2,2,2), c(1,2,NA)))
## FAOcheck(var = "Value", data = test.df, type = "overlap", take = "simpleCheck")
## FAOcheck(var = "Value", data = test.df, type = "overlap", take = "takeNew")
## FAOcheck(var = "Value", data = test.df, type = "overlap", take = "takeOld")
## FAOcheck(var = "Value", data = test.df, type = "overlap", take = "complete")
```

FAOcountryProfile *Country profile*

Description

The country profile containing the codes and names of countries.

FAOmetaTable — *The search tree for FAOSTAT3*

Description

A table containing the relationship between the domain, element, item codes for downloading data from the FAOSTAT API.

FAOregionProfile — *Regional profile*

Description

Region profile containing the codes, names and regional classifications of countries.

FAOsearch — *A function to find the domain, element and item code for a specific FAOSTAT query.*

Description

A function to find the domain, element and item code for a specific FAOSTAT query.

Usage

```
FAOsearch()
```

fillCountryCode — *A function to get country code when not available in data.*

Description

This function can be useful when a dataset provided does not have a country code available.

Usage

```
fillCountryCode(country, data, outCode = "FAOST_CODE")
```

Arguments

country	The column name of the data which contains the country name
data	The data frame to be matched
outCode	The output country code system, defaulted to FAO standard.

`geogr` *Geometric growth rate*

Description

Function for generating the n-period rolling geometric growth rate.

Usage

```
geogr(x, n = 1)
```

Arguments

`x`	The time series for the growth rate to be calculated.
`n`	The period for the growth to be calculated over.

Details

In order to ensure the growth rate calculated is reliable, the following rule are applied.

1. 50% of the data must be present.
2. The length of the time series must be greater than n

Otherwise the growth will not be computed.

Value

The n-period geometric growth rate of the time series.

Examples

```
test.ts = abs(rnorm(100))
geogr(test.ts, 1)
geogr(test.ts, 3)
geogr(test.ts, 10)
```

getFAO *Access to FAO FAOSTAT API.*

Description

A function to access FAOSTAT data through the FAOSTAT API.

Usage

```
getFAO(name = NULL, domainCode = "RL", elementCode = 5110,
  itemCode = 6621, query, printURL = FALSE, useCHMT = TRUE,
  outputFormat = "wide", returnNames = FALSE, returnFlags = FALSE,
  yearRange = NULL, countrySet = NULL)
```

Arguments

name	The name to be given to the variable.
domainCode	The domain of the data.
elementCode	The code of the element.
itemCode	The code of the specific item.
yearRange	A numeric vector containing the years to be downloaded.
countrySet	The FAOSTAT codes of those countries to be downloaded.
query	The object created if using the FAOsearch function.
printURL	Whether the url link for the data should be printed.
useCHMT	logical, whether the CHMT function should be applied to avoid double counting of China.
outputFormat	The format of the data, can be 'long' or 'wide'.
returnNames	Logical, should the area, the element and the item names be reported?.
returnFlags,	Logical, whether the flags should be returned. Only work with outputFormat long.

Details

Need to account for multiple itemCode, currently only support one single variable.

Value

Outputs a data frame containing the specified data.

See Also

getWDI, getWDItoSYB, getFAOtoSYB, FAOsearch

getFAOtoSYB *Access to FAO FAOSTAT API*

Description

A wrapper function using getFAO() to obtain and process multiple data set to obtain data.

Usage

```
getFAOtoSYB(name = NULL, domainCode = "RL", elementCode = 5110,
  itemCode = 6621, query, printURL = FALSE, useCHMT = TRUE,
  yearRange = NULL, countrySet = NULL, outputFormat = c("wide", "long"),
  returnFlags = FALSE)
```

Arguments

name	The name to be given to the variable.
domainCode	The domain code of the variable, see details.
elementCode	The element code of the variable, see details.
itemCode	The item code of the variable, see details.
query	The object created if using the FAOsearch function
printURL	Whether the url link for the data should be printed
useCHMT	logical, whether the CHMT function should be
outputFormat	The format of the data, can be 'long' or 'wide'. appied to avoid double counting of China.
returnFlags,	Logical, whether the flags should be returned. Only work with outputFormat long.
yearRange	A numeric vector containing the years to be downloaded.
countrySet	The FAOSTAT codes of those countries to be downloaded.

Value

A list containing the following elements

entity The entity level data

aggregates The aggregates provided by the FAO

results The status of the download, whether success/failed

See Also

getWDI, getFAO, getWDItoSYB

Examples

```
## The default option is the arable land area
## arlLand.lst = getFAOtoSYB()
```

getWDI *Access to World Bank WDI API*

Description

A function to extract data from the World Bank API

Usage

```
getWDI(indicator = "SP.POP.TOTL", name = NULL, startDate = 1960,
  endDate = format(Sys.Date(), "%Y"), printURL = FALSE,
  outputFormat = "wide")
```

Arguments

indicator	The World Bank official indicator name.
name	The new name to be used in the column.
startDate	The start date for the data to begin
endDate	The end date.
printURL	Whether the url link for the data should be printed
outputFormat	The format of the data, can be 'long' or 'wide'.

Details

Please refer to http://data.worldbank.org/node/18 for any difference between the country code system. Further details on World Bank classification and methodology are available at http://data.worldbank.org/.

Value

A data frame containing the desired World Bank Indicator

See Also

getFAO, getWDItoSYB, getFAOtoSYB

Examples

```
## pop.df = getWDI()
```

getWDImetaData *World Bank Indicator Metadata*

Description

A function to extract the definition and the meta data from the World Bank API

Usage

```
getWDImetaData(indicator, printMetaData = FALSE, saveMetaData = FALSE,
  saveName = "worldBankMetaData")
```

Arguments

indicator	The World Bank official indicator name.
printMetaData	logical, print out the meta data information
saveMetaData	logical, whether meta data should be saved as a local csv file.
saveName	The name of the file for the meta data to save to.

Examples

```
## pop.df = getWDImetaData("SP.POP.TOTL",
##                         printMetaData = TRUE, saveMetaData = TRUE)
```

getWDItoSYB *Access to World Bank WDI API*

Description

The function downloads data from the World Bank API.

Usage

```
getWDItoSYB(indicator = "SP.POP.0014.TO.ZS", name = NULL,
  startDate = 1960, endDate = format(Sys.Date(), "%Y"), printURL = FALSE,
  getMetaData = TRUE, printMetaData = FALSE, saveMetaData = FALSE,
  outputFormat = c("wide", "long"))
```

Arguments

name	The new name to be used in the column.
indicator	The World Bank official indicator name.
startDate	The start date for the data to begin
endDate	The end date.
printURL	Whether the url link for the data should be printed
getMetaData	Whether the data definition and the meta data should be downloaded as well.
printMetaData	logical, print out the meta data information
saveMetaData	logical, whether meta data should be saved as a local csv file
outputFormat	The format of the data, can be 'long' or 'wide'.

Value

A list containing the following elements

data The country level data

aggregates The aggregates provided by the World Bank

metaData The metaData associated with the data

results The status of the download, whether success/failed

See Also

getWDI, getFAO, getFAOtoSYB

Examples

```
## pop.df = getWDItoSYB(name = "total_population",
##                      indicator = "SP.POP.TOTL")
```

---

grConstruct *Construct Growth rate*

Description

A function for constructing growth rate variables.

Usage

```
grConstruct(data, origVar, newVarName = NA, type = c("geo", "ls", "ch"),
  n = 1)
```

Arguments

data	The data frame containing the data
origVar	The variable in which the growth is to be calculated
newVarName	The name assigned to the new variable, if missing then .SC/.SH/.GR will be appended depending on the type of construction.
type	The type of growth rate, can be least squares or geometric
n	The period for the growth rate to be calculated (Refer to the lsgr or the geogr functions.)

Value

A data frame containing the computed growth rate.

Examples

```
test.df2 = data.frame(FAOST_CODE = rep(c(1, 5000), each = 5),
                      Year = rep(1990:1994, 2),
                      a = rep(1:5, 2), b = rep(1:5, 2))
grConstruct(test.df2, origVar = "a", type = "geo", n = 1)
grConstruct(test.df2, origVar = "a", type = "geo", n = 3)
grConstruct(test.df2, origVar = "a", type = "geo", n = 5)
```

---

indConstruct *Construct indices*

Description

A function for constructing indices

Usage

```
indConstruct(data, origVar, newVarName = NA, baseYear = 2000)
```

Arguments

data	The data frame containing the data
origVar	The variable in which the indices is to be computed
newVarName	The name assigned to the new variable, if missing then .SC/.SH/.GR/.CH/.IND will be appended depending on the type of construction.
baseYear	The year which will serve as the base

Value

The indice

Examples

```
test.df = data.frame(FAOST_CODE = rep(1, 100), Year = 1901:2000,
                     test = 1:100)
indConstruct(test.df, origVar = "test", baseYear = 1950)
```

`lsgr` *Least squares growth rate*

Description

Function for generating the n-period rolling least squares growth rate.

Usage

```
lsgr(x, n = 1)
```

Arguments

`x`	The time series for the growth rate to be calculated
`n`	The period for the growth to be calculated over.

Details

Missing values are ommited in the regression. (Will need to check this.)

TODO (Michael): There is still some error associated with this function, will need to investigate further. Will need a rule for this, when the fluctuation is large and data are sufficient then take the lsgr, otherwise the geogr.

In order to ensure the growth rate calculated is reliable, the following rule are applied.

1. 50% of the data must be present.
2. The length of the time series must be greater than n.

Otherwise the growth will not be computed.

Value

The n-period least squares growth rate of the time series

Examples

```
test.ts = abs(rnorm(100))
lsgr(test.ts, 1)
lsgr(test.ts, 3)
lsgr(test.ts, 10)
```

mergeSYB — *Function for merging data from different source.*

Description

This function searches for supported country system and translate the data to allow for join.

Usage

```
mergeSYB(x, y, outCode = "FAOST_CODE", all = TRUE, ...)
```

Arguments

x	data frames, or objects to be coerced to one.
y	data frames, or objects to be coerced to one.
outCode	The country code system to be used to join the different sources.
all	Same as the merge function, defaulted to an outer join.
...	Arguments to be passed on to the merge function.

Details

The names of the data to be merged has to be the same as the FAOcountryProfile code name.

overlap — *This function checks whether there are overlapping between the transitional countries.*

Description

This function checks whether there are overlapping between the transitional countries.

Usage

```
overlap(old, new, var, year = "Year", data, take)
```

Arguments

old	The FAOST_CODE of the old countries
new	The FAOST_CODE of the new countries
var	The variable to be checked
year	The column which index the time.
data	The data frame
take	The type of check/replacement to be done.

printLab *Print labels*

Description

A function to print standardised formatted labels without having messy codes in the functions.

Usage

```
printLab(label, span = FALSE, width = getOption("width"))
```

Arguments

label	The label to be printed
span	Whether the dash should span the whole width of the screen(80 characters)
width	The width of the screen.

Value

The formatted print

scaleUnit *A function to standardize the unit*

Description

The function standardize the data to the desirable unit when the multiplier vector is supplied. For example per 1000 people is scaled to per person by supplying a multiplier of 1000.

Usage

```
scaleUnit(df, multiplier)
```

Arguments

df	The data frame containing the data to be scale
multiplier	The named vector with the multiplier to be scaled. The name is mandatory in order for the function to identify the variable in the data frame. A data.frame can also be supplied with the first column being the name and the second being the numeric multiplier.

Examples

```
## Create the data frame
test.df = data.frame(FAOST_CODE = 1:5, Year = 1995:1999,
  var1 = 1:5, var2 = 5:1)

## Create the named vector for scaling
multiplier = c(1, 10)
names(multiplier) = c("var1", "var2")

## Scale the data
scaleUnit(test.df, multiplier = multiplier)
```

---

shConstruct *Construct share variable*

Description

A function for constructing the share of a variable of an aggregated variable.

Usage

```
shConstruct(data, totVar, shareVar, newVarName = NA)
```

Arguments

data	The data frame containing both the share variable and the aggregated variable
totVar	The aggregated variable.
shareVar	The subset of the aggregated variable which to be divided by.
newVarName	The name assigned to the new variable, if missing then .SC/.SH/.GR will be appended depending on the type of construction

Details

The share of a variable can be share of the World (if additional variable were not supplied) or share of another variable (per Capita if population was supplied).

Value

A data frame with the new constructed variable

Examples

```
## Total variables provided, scale by totVar
test.df = data.frame(FAOST_CODE = 1, Year = 1990:1994, a = 1:5, b = 1:5)
shConstruct(data = test.df, totVar = "a", shareVar = "b")

## Total variables not provided, scale by world aggregate.
test.df2 = data.frame(FAOST_CODE = rep(c(1, 5000), each = 5),
```

```
                      Year = rep(1990:1994, 2),
                      a = rep(1:5, 2), b = rep(1:5, 2))
shConstruct(data = test.df2, totVar = NA, shareVar = "b")
```

`translateCountryCode` *A function to translate between different country coding systems*

Description

The function translate any country code scheme to another if both are in the FAOcountryProfile

Usage

```
translateCountryCode(data, from, to, oldCode)
```

Arguments

`data`	The data frame
`from`	The name of the old coding system
`to`	The name of the new coding system
`oldCode`	The column name of the old country coding scheme

`translateUnit` *Function to translate multipliers*

Description

This function translates number to character name or vice versa

Usage

```
translateUnit(vec)
```

Arguments

`vec`	The vector containing name or number to be translated

Examples

```
## Create numeric vector
myUnit = c(1000, 1e6, 1000, 1e9, 1e9, 1e12)

## Translate numeric to character
myUnit2 = translateUnit(myUnit)
myUnit2

## Now translate back
translateUnit(myUnit2)
```

Index