Package ‘OpasnetUtils’

July 17, 2013

Type Package

Title Utility functions for dealing with data in Opasnet (www.opasnet.org) environment.

Version 1.0.0

Author Teemu Rintala, Einari Happonen, Jouni Tuomisto

Maintainer Einari Happonen <`einari.happonen@thl.fi`>

Description This package is primarily developed for executing open assessments and modeling in Opasnet environment. See http://www.opasnet.org for more.

Imports methods, rjson, RCurl, reshape2, triangle, httpRequest,digest, xtable

License GPL-3

LazyLoad yes

R topics documented:

`OpasnetUtils-package` *Open Assessors Network Modelling Utilities*

Description

This package contains tools made for building stochastic models within Opasnet (`http://www.opasnet.org`).

Details

Package:	OpasnetUtils
Type:	Package
Version:	1.0.0
Imports:	methods, rjson, RCurl, xtable, reshape2, triangle, httpRequest, digest
License:	GPL-3
LazyLoad:	yes
Date:	2013-07-09

Index:

```
Check                   Ovariable Checks
CollapseMarginal        Collapse marginals
CollapseTableParser     Parsing Collapse orders from a table
ComputeDependencies     Evaluate ovariable dependencies
DecisionTableParser     Parse data.frame for decisions
EvalOutput              Evaluate ovariable output
Fetch                   Fetch R objects described in a data.frame
GIS                     Handling spatially distributed variables
OpasnetUtils-package    Open Assessors Network Modeling Utilities
Ovariable               Ovariable constructor
convert.units           Converting units
ddata_apply             Dynamic data link activation
dropall                 Dropall
fillna                  Interpreting empty locations in indices
interpret               Parse human readable distribution definitions
oapply                  Apply for ovariables
objects                 Server side shared R objects
odecision-class         Class '"odecision"'
op_base                 Functions for Interaction with the Opasnet Base
                        (obsolete)
opasnet                 Importing files from Opasnet
opbase                  OpasnetBase Access
```

`oprint`	`Print ovariables or data frames in html format.`
`orbind`	`Rowbinding ovariables`
`ovariable-class`	`Class '"ovariable"'`
`result`	`Access result vector of an ovariable`
`tidy`	`Format database output for use in ovariables`

The operating principle of this package is maximum modularity. Variables are defined publicly on wiki pages using wiki inputs/tables, our database and R code. Using any predefined variable is very easy: fetch the variable from our servers (or your own) and evaluate it. Actual evaluation of variables is done lazily by default: when the evaluation of a variable is explicitly called, all variables it is dependent on are evaluated recursively. There are also a few impact assessment tools like a few GIS functions in the package. To learn more go to `http://en.opasnet.org/w/Modelling_in_Opasnet`.

Author(s)

T. Rintala, <`teemu.rintala.a@gmail.com`>
E. Happonen, <`einari.happonen@thl.fi`>
J. Tuomisto, <`jouni.tuomisto@thl.fi`>

Maintainer: E. Happonen <einari.happonen@thl.fi>

`Check` *Ovariable Checks*

Description

The Check functions are used to introduce common model specific alterations to variables without changing their definition directly.

Usage

```
CheckCollapse(variable, indent = 0, verbose = TRUE, ...)
CheckDecisions(variable, indent = 0, verbose = TRUE, ...)
CheckInput(variable, substitute = FALSE, indent = 0, verbose = TRUE, ...)
CheckMarginals(variable, deps = list(), priormarg = TRUE, indent = 0,
    verbose = TRUE, ...)
```

Arguments

`variable`	an ovariable to run Check on.
`deps`	the dependency list of a latent ovariable, used by CheckMarginals to keep track of index columns.
`priormarg`	flag determining whether columns are assumed to be indices by default when checking marginals.
`substitute`	flag determining whether model inputs should replace or append to current variable values.
`indent`	used by verbose to structure status messages by using indentation.
`verbose`	flag status message printing.
`...`	excess arguments are ignored.

Details

The Check functions are mainly used internally. They check for external instructions (model specific changes); specifically objects in `.GlobalEnv` with prefixes ("Col", "Dec", "Inp"). they are automated in the normal `ovariable` evaluation routine (`EvalOutput`).

`CheckCollapse` uses `CollapseMarginal` which collapses marginals by applying sums, means or samples. Also loses all non-marginal columns except the relevant "Result". It is mainly used to streamline models by reducing rows in data.

`CheckDecisions` checks for and applies decisions on variables. The function makes use of the `odecision-class`, which specifies the target cells as well as the effect. Odecisions are most often produced by `DecisionTableParser`.

`CheckInput` checks and uses outside input (run specific user inputs in models). Input should be in `ovariable` format.

`CheckMarginals` fills the marginal slot of an `ovariable` using information from variable data and upstream variable marginals. Assumes that all depended upon variables are loaded, as should be the case.

See also: `http://en.opasnet.org/`

Value

Original `ovariable` with possible adjustments.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

`CollapseMarginal` *Collapse marginals*

Description

Apply functions (only `sample` at the moment) over ovaribale indices

Usage

```
CollapseMarginal(variable, cols, probs = NULL, ...)
```

Arguments

`variable`	an ovariable
`cols`	vector of column names or indices to collapse
`probs`	list of vectors defining the distribution of values in a column index
`...`	excess arguments are ignored

Details

Samples over a fully defined column index (marginal) treating it as a nuisance parameter. Increases the joint distribution uncertainty (loses information). Weighted sampling is also possible.

Used to streamline heavy models (output has fewer rows of data).

See also: `http://en.opasnet.org/`

Value

Input `ovariable` with possibly lighter output.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

Examples

```
test <- Ovariable("test", output = data.frame(City = c("Helsinki", "Espoo"),
    Iter = 1, testResult = 1:2))
CollapseMarginal(test, "City", NA)
```

`CollapseTableParser` *Parsing Collapse orders from a table*

Description

Parses `data.frames` of specific format to produce "Col" prefixed lists for `CheckCollapse`

Usage

```
CollapseTableParser(CTable, env = .GlobalEnv)
```

Arguments

`CTable`	a data.frame with columns "Variable" (variable names), "Index" (column names) and "Probs" (propabilities of column levels in marginal distribution, comma separated)
`env`	target `environment`, default is `.GlobalEnv`

Details

Used in the Opasnet assessments/analyses to produce multiple model specific Collapse instructions. Using other distribution values than 1 requires knowledge about the amount and order of unique index values. `Probs` values `1` and `NA` are considered equal weighting.

See also: `http://en.opasnet.org/`

Value

No return value, "Col" prefixed variables are written straight into specified `environment`.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

Examples

```
a <- data.frame(Variable = "test", Index = c("City"), Probs = 1)
CollapseTableParser(a)
Coltest
```

ComputeDependencies *Evaluate* ovariable *dependencies*

Description

Fetches, evaluates and Checks ovariable dependencies given in data.frame format

Usage

```
ComputeDependencies(dependencies, forceEval = FALSE, indent = 0,
    verbose = FALSE, new_code = FALSE, ...)
```

Arguments

dependencies	data.frame that defines Fetch targets, usually taken from an ovariable's dependencies slot
forceEval	if TRUE, forcibly re-evaluates existing instances of listed dependencies
indent	verbose print assist for the Check family, used internally
verbose	TRUE to enable status message printing assists (line breaks) and status messages for other function calls
new_code	a flag for compatibility with older code, default FALSE nullifies ComputeDependencies usage in ovariable formulas
...	arguments to pass on to the various recursive checks and evaluations

Details

ComputeDependencies uses Fetch, EvalOutput, CheckDecisions, CheckCollapse and CheckInput to load and pre-process upstream variables. It is automatically called by EvalOutput, but can be seen on the first lines of old ovariable formula code, to avoid applying decisions, inputs and optimizations twice in old code the function does nothing by default. This is no problem since users should not be calling this function at all. ComputeDependencies also does most of the exception handling in the recursive ovariable model.

See also: http://en.opasnet.org/

Value

No return value

Author(s)

T. Rintala <teemu.rintala.a@gmail.com>

`convert.units` *Converting units*

Description

Converts units (to SI equivalents by default)

Usage

```
convert.units(x, tounit = c("kg", "s", "m", "m3", "J", "W", "A",
    "V", "C", "N", "Pa", "Hz", "mol"), fromunit = NULL)
```

Arguments

`x`	`numeric` vector with values to be converted
`tounit`	`character` vector of the new units to be used
`fromunit`	`character` vector or factor with the current units

Details

Uses the table in `en.opasnet.org/w/Unit_conversions` for the conversions, so the units used have to be specified there.

See also: `http://en.opasnet.org/`

Value

Returns a `data.frame`

Author(s)

J. Tuomisto <`jouni.tuomisto@thl.fi`>

Examples

```
convert.units(1, tounit = c("pg", "l"), fromunit = "ug /m3")
```

`ddata_apply` *Dynamic data link activation*

Description

Fetches the latest data associated with an `ovariable` from the OpasnetBase if available

Usage

```
ddata_apply(ovariable, ddata_tidy = TRUE, force_ddata = FALSE, ...)
```

Arguments

`ovariable`	an `ovariable` with the ddata slot defined as page_id i.e. "Op_en1000"
`ddata_tidy`	`TRUE` to run `tidy` on downloaded data
`force_ddata`	if `TRUE`, dynamic data links are used even if the data slot of an `ovariable` is already defined
`...`	excess parameters are ignored

Details

This function is mostly used internally

See also: `http://en.opasnet.org/`

Value

Returns the input `ovariable`. (Re)defines the data slot if it is not already defined ands ddata is available.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

See Also

`ovariable`

`DecisionTableParser` *Parse data.frame for decisions*

Description

Parses a `data.frame` into `odecision`s.

Usage

```
DecisionTableParser(DTable, env = .GlobalEnv)
```

Arguments

`DTable`	`data.frame`
`env`	target `environment`, default is `.GlobalEnv`

Details

Decisions consist of conditions and effects, target a certain variable and may have multiple options.

Input format is described on `http://en.opasnet.org/w/Decision`. Currently usable decision effects are Add, Multiply, Replace, Remove and Identity.

See also: `http://en.opasnet.org/`

Value

No return value. Saves `odecision` class objects into specified `environment`.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

See Also

`CheckDecisions`

Examples

```
modeldecisions <- data.frame(Stakeholder = "Group A", Decision = "More wind power",
  Option = "A few more turbines", Variable = "PowerGeneration", Cell = "Type:Wind",
  Change = "Add", Result = "5")
DecisionTableParser(modeldecisions)
ls()
```

`dropall` *Dropall*

Description

Drops unused factor levels in data.frames

Usage

```
dropall(x)
```

Arguments

x	a `data.frame`

Details

This function makes sure that the `factor levels` in a `data.frame` do not contain entries that have already been removed from the `factor` itself.

See also: `http://en.opasnet.org/`

Value

`data.frame`

Author(s)

J. Tuomisto <`jouni.tuomisto@thl.fi`>

Examples

```
a <- data.frame(A = c("a", "b"), B = c(1,1))
levels(a[[1]])

a <- a[-2 ,]
levels(a[[1]])
a[[1]]

a <- dropall(a)
levels(a[[1]])
a[[1]]
```

`EvalOutput` *Evaluate ovariable output*

Description

Evaluate the output slot of an `ovariable`, which usually means recursively evaluating any dependent variables as well.

Usage

```
EvalOutput(variable, fillna = FALSE, indent = 0, verbose = FALSE, ...)
```

Arguments

`variable`	an `ovariable`
`fillna`	if `TRUE`, `fillna` is attempted at the end
`indent`	internal `integer` argument used in verbose printing
`verbose`	use `TRUE` to enable status messages while processing outputs and various checks
`...`	arguments are passed on to `ovariable` formulas and to dependent `EvalOutput` calls (recursivity), number of iterations (`N`) is commonly set here

Details

`EvalOutput` automates most of the other features related to `ovariable` handling. It runs `ComputeDependencies` first, produces a `data.frame` by combining the return values from `interpret`ing the data slot and running the formula slot function, makes a "Source" -column to distinguish between the two "Results" and lastly `CheckMarginals` is run on the variable (optionally also `fillna`).

Since `EvalOutput` is usually run on the end node of a model, there should not be inputs or decisions hence they are not checked for. In contrast `ComputeDependencies` runs all `Check`s besides `CheckMarginals`.

See also: `http://en.opasnet.org/`

Value

Returns the input `ovariable` with the output slot (re)defined.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

Examples

```
a <- Ovariable("a", data.frame(A = c("a", "b"), Result = c("1-2", "1-4")))
a <- EvalOutput(a, N = 10)
a@output
```

`Fetch` *Fetch R objects described in a data.frame*

Description

Download a batch of R objects from Opasnet servers.

Usage

```
Fetch(dependencies, evaluate = FALSE, indent = 0, verbose = TRUE, ...)
Fetch2(...)
```

Arguments

`dependencies`	`data.frame` which defines variable names and "locations"
`evaluate`	`TRUE` to run `EvalOutput` on each variable (non-`ovariables` are ignored)
`indent`	`integer` internal argument for verbose printing
`verbose`	use `TRUE` to enable status messages in between fetches
`...`	excess arguments are ignored or passed to `EvalOutput` if evaluate is `TRUE`

Details

The input `data.frame` should have columns "Name" and at least one of "Key" and "Ident".

Key is the R-tools session identifier (shown at the end of the url). Ident should be in format <page_id>/<code_name>.

Fetch first checks if the variable (or something with the same name) is already available, if it is nothing will be done. If Key is defined (not NA or "") for a variable it takes precedence over Ident.

Fetch is run as first part of `ComputeDependencies`.

See also: `http://en.opasnet.org/`

Value

No return value. Fetched variables are written in `.GlobalEnv`.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

Examples

```
deps <- data.frame(Name = "exposure", Key = "6WYTFxiZUIxiY8tw")
#Fetch(deps)
#exposure

# If variable exists
exposure <- 1
Fetch(deps)
exposure # by default nothing is changed
```

`fillna` *Interpreting empty locations in indices*

Description

Copies result rows that have `NA`s as index values and replaces the index value with all available values of that index.

Usage

```
fillna(object, marginals)
```

Arguments

`object`	a `data.frame` to be filled
`marginals`	`integer`, positions of columns whose locations contain NAs that should be duplicated

Details

Runs `dropall` before duplication to avoid unnecessary levels.

See also: `http://en.opasnet.org/`

Value

Returns a `data.frame`

Author(s)

J. Tuomisto `<jouni.tuomisto@thl.fi>`

Examples

```
a <- data.frame(A = c("a", "b", NA), B = c(1, 2, 3))
fillna(a, 1)
```

GIS — *Handling spatially distributed variables*

Description

Currently there are only GIS functions for producing spatial concentration maps (`GIS.Concentration.matrix`) and using (closed) spatial population data to calculate exposure (`GIS.Exposure`).

Usage

```
GIS.Concentration.matrix(Emission, LO, LA, distx = 10.5, disty = 10.5,
    resolution = 1, N = 1000, dbug = FALSE, ...)
GIS.Exposure(Concentration.matrix, LO = NULL, LA = NULL, distx = 10.5,
    disty = 10.5, resolution = 1, dbug = FALSE, ...)
```

Arguments

`Emission`	`numeric`, emission in any units; can be an `ovariable` as well
`Concentration.matrix`	`ovariable` concentration matrix, usually produced by `GIS.Concentration.matrix`
`LO`	`numeric`, longitude corresponding to the center of the considered area (emission source)
`LA`	`numeric`, latitude corresponding to the center of the considered area (emission source)
`distx`	`numeric`, maximum distance from center on the x axis of the area, 10.5 corresponds to the source-receiver-matrices used
`disty`	`numeric`, maximum distance from center on the y axis of the area, 10.5 corresponds to the source-receiver-matrices used
`resolution`	`numeric`, size of the grid, default 1 is 1km x 1km grid
`N`	`integer`, number of iterations to run
`dbug`	use `TRUE` to turn debug prints on
`...`	excess arguments are ignored or passed to `tidy` on data download

Details

The concentration matrix is computed using PILTTI source-receiver-matrices (`http://en.opasnet.org/w/Piltti_source-receptor_matrix`). They are originally for modeling PM2.5 distributions in a few Finnish cities between the years 2000 and 2003. To produce a rudimentary probability distribution these matrices are randomized between iterations.

Exposure is calculated by matching a concentration matrix to Finnish population data. Currently used data is closed and its usage hard coded, but open data exists (`http://en.opasnet.org/w/Special:Opasnet_Base?id=op_en2949.2012`) and this function should be updated to be more adaptable.

LA and LO are not required arguments for exposure, but speed the computation significantly.

See also: `http://en.opasnet.org/`

Value

GIS.Concentration.matrix returns an ovariable whose output is a grid defined as bins for coordinates.

GIS.Exposure returns an ovariable whose output is concentration * population, with spatial information stripped to keep the data closed.

Author(s)

T. Rintala <teemu.rintala.a@gmail.com>

Examples

```
# Excerpt from http://en.opasnet.org/w/Health_impacts_of_fine_particles_in_Rauma
# (not evaluated)

# Paasto Emissions

Paasto <- new(
"ovariable",
name = "Paasto",
dependencies = data.frame(Name = "tieliikennepaastot", Key = "0194sOuuucjxq8Wi"),
formula = function(dependencies, ...) {
ComputeDependencies(dependencies, ...)

# Muutetaan paivapaasto vuosipaastoksi ja grammat tonneiksi
out <- tieliikennepaastot * 365 * 1E-6

return(out)
}
)

# Muita tarpeellisia arvoja Other important values

bg.mort <- 45182 / 5203826 # same values as used in PILTTI

## J. T. Tuomisto, A. Wilson, et al. Uncertainty in mortality response to
## airborne fine particulate matter... 2008
erf <- 0.0097
# unit: m^3 /ug

# Ovariablet

## Pitoisuudet Concentrations

Pitoisuus <- new(
"ovariable",
name = "Pitoisuus",
dependencies = data.frame(
Name = c("Paasto", "LO", "LA")
),
formula =  function(dependencies, ...) {
ComputeDependencies(dependencies, ...)

temp <- GIS.Concentration.matrix(Paasto, LO, LA, ...)

return(temp)
}
)

## Altistuminen Exposure

Altistuminen <- new(
"ovariable",
name = "Altistuminen",
dependencies = data.frame(
Name = c("Pitoisuus", "LO", "LA")
),
formula = function(dependencies, ...) {
ComputeDependencies(dependencies, ...)

out <- GIS.Exposure(Pitoisuus, LO, LA, ...)

return(out)
}
)
```

`interpret` *Parse human readable distribution definitions*

Description

Interpret textual data into probability distributions using regular expressions.

Usage

```
interpret(idata, N = 1000, rescol = "Result", dbug = FALSE, ...)
```

Arguments

`idata`	input, `character` or `data.frame`
`N`	number of iterations
`rescol`	name of result column, defaults to "Result"
`dbug`	use `TRUE` to turn on debug prints
`...`	excess arguments are ignored

Details

Interpretation rules are as follows: Empty space is stripped away. "X-Y" defines a uniform distribution between X and Y, if Y/X is greater than 100 then logarithmic uniform distribution is assumed. Negative X and Y are determined by the number of "-": if 2, X is negative; if 3, both are.
"<X" defines a uniform distribution between 0 and X.
"X+-Y" defines a normal distribution with mean X and sd Y.
"X(Y-Z)" defines a normal distribution where Y-Z is assumed the 95-percent confidence interval, from which sd is determined.
If distance from mean to the higher boundary is 50-percent higher than to lower boundary log normality is assumed.
"X:Y:Z" defines a triangular distribution with min, mode and max (can be given in any order).

"X1;X2;...;Xn" defines a random unbiased sample (with replacement) between the given elements. See `http://en.opasnet.org/w/Interpret` for a table.

See also: `http://en.opasnet.org/`

Value

Returns a `data.frame` with an "Iter" column added. Uninterpretable values are converted to `NA`s.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

Examples

```
interpret(c("1-4", "1-1000"), N = 5)
```

`oapply` *Apply for ovariables*

Description

Use `tapply` on the output slot

Usage

```
oapply(X, INDEX = NULL, FUN = NULL, cols = NULL, ..., simplify = TRUE)
```

Arguments

`X`	an `ovariable`
`INDEX`	list of factors, like `tapply`
`FUN`	function to apply
`cols`	names of columns to be removed (reverse INDEX)
`...`	optional arguments to FUN
`simplify`	like `tapply`

Details

See also: `http://en.opasnet.org/`

Value

Returns an `ovariable`, with output slot tapplied and marginal adjusted accordingly.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

Examples

```
a <- new("ovariable", name = "a", output = data.frame(A = c("a", "a", "b", "b"),
    B = c("1", "2", "1", "2"), aResult = 1:4), marginal = c(TRUE, TRUE, FALSE))
oapply(a, FUN = sum, cols = "A")
oapply(a, a@output[c("A")], sum)
```

---

objects   *Server side shared R objects*

Description

Library for using R objects (like `ovariable`s.) stored in Opasnet R server. Also includes basic encryption and decryption functionality for R objects.

Usage

```
objects.encode(obj, key)
objects.decode(eobj, key)
objects.get(token)
objects.latest(page_ident, code_name, verbose = FALSE)
objects.put(..., list = character())
objects.store(..., list = character(), verbose = FALSE)
```

Arguments

`obj`	Any R object.
`eobj`	An encoded object returned by `objects.encode` function.
`key`	Key string to encode or decode objects. Must be 16,32 or 64 characters in length.
`token`	R-tools run token string to identify a stored object on Opasnet R server.
`page_ident`	Opasnet Media Wiki page identifier (e.g. op_en1390).
`code_name`	Name of the R code block in Opasnet Media Wiki (the `name` argument in `rcode` tag).
`verbose`	Flag to set more verbose output (for debug purposes).
`...`	Objects will be passed straight to R core `save` function.
`list`	List will be passed straight to R core `save` function.

Details

The main purpose of this library is to provide means to store R objects to Opasnet R server for later use. This is specifically useful and embraced in Opasnet R ecosystem where this library is mainly used for storing and fetching `ovariable`s. Storing objects is currently only possible within Opasnet Media Wiki environments (using R code inside `rcode` tags), but reading objects stored by running code within public wikis is also possible from local R-installation.

Besides object storing, this library provides basic functionality to encode and decode R objects. This is done by using R base serialization functions and `digest` library. Objects given to `objects.encode` will be encoded by using `AES` function in "ECB" mode. Longer key (16, 32 or 64 characters) obviously means more secured encryption too. Same key must be used for both encryption and decryption.

See also: `http://en.opasnet.org/`

Value

`objects.encode`	Returns encoded object to be decoded with `objects.decode` and given key.
`objects.decode`	Returns decoded object, as it was before encoding with `objects.encode`.
`objects.get`	Returns object or objects stored to Opasnet R server.
`objects.latest`	Returns object or objects stored to Opasnet R server.
`objects.put`	No return value.
`objects.store`	Returns token to identify stored objects on R server.

Author(s)

E. Happonen `<einari.happonen@thl.fi>`

See Also

`load`
`save`
`serialize`
`AES`

Examples

```
library(OpasnetUtils)

# Within Opasnet only! Let's assume that the (en.opasnet.org) page identifier -
# where to code is - would be "Op_en1390" and code name "objs_save_test".
# x <- stats::runif(20)
# y <- list(a = 1, b = TRUE, c = "Jeah baby jeah!")
# objects.store(x, y)

# Fetching can be done also from local R installation.
objects.latest("Op_en1390","objs_save_test")
print(x)
print(y)

# Object encrypt and decrypt

key <- "1234567890abcdef"

eobj <- objects.encode(y, key)
print(eobj)
obj <- objects.decode(eobj, key)
print(obj)
```

`odecision-class` *Class* `"odecision"`

Description

Definition container for `CheckDecisions`

Details

Usually odecisions are created by DecisionTableParser using a full decision table that includes condition and effect descriptions in standard form. Odecisions created by DecisionTableParser do not have condition or effect defined. Instead CheckDecisions does the final parsing into preset effects and conditions. For non-standard conditions and effects decisions can be defined using the `new("odecision", ...)` call.

See also: http://en.opasnet.org/

Objects from the Class

Objects can be created by calls of the form `new("odecision", ...)`.

Slots

`dectable`: Object of class `"data.frame"` describes the decisions and their relevant options. It is merged with the output slot `data.frame` of an `ovariable`

`condition`: Object of class `"list"` contains functions which return a logical vector that should indicate the relevant rows to be affected by a decision-option combination.

`effect`: Object of class `"list"` contains functions which describe the effects of the decision on relevant rows of the output.

Methods

No methods defined with class "odecision" in the signature.

Author(s)

T. Rintala <teemu.rintala.a@gmail.com>

---

opasnet *Importing files from Opasnet*

Description

Functions for downloading files from Opasnet Media Wiki environments.

Usage

```
opasnet.data(filename, wiki = "", unzip = "")
opasnet.csv(filename, wiki = "", unzip = "", ...)
```

Arguments

`filename`	Path to file in Opasnet after the "images/" part.
`wiki`	Name of the Opasnet wiki: "opasnet_en" for en.opasnet.org, "opasnet_fi" for fi.opasnet.org or "heande" for heande.opasnet.org (accessible only within Heande wiki).
`unzip`	Name of the file in the package (if compressed using zip).
`...`	Excess arguments will be passed to `read.table` function when downloading csv-file.

Details

These functions make it easy to download Opasnet files for being used in R. Required path (filename) for file must be resolved using the corresponding Media Wiki. Big data files should always be compressed before uploading to Opasnet. Using the `unzip`-argument makes it easy to download and directly use any zip-compressed files.

See also: `http://en.opasnet.org/`

Value

`opasnet.data`	Returns the file data as is.
`opasnet.csv`	Returns `data.frame` parsed from csv file.

Author(s)

E. Happonen `<einari.happonen@thl.fi>`

See Also

`read.table`

Examples

```
library(OpasnetUtils)
opasnet.csv("4/49/Test.zip", wiki = "opasnet_fi", unzip = 'ejpop.csv', sep=';')
opasnet.data("c/cc/Test_bugs_model.txt", wiki = "opasnet_en")
```

`opbase` *OpasnetBase Access*

Description

Function family for interacting with the Opasnet database.

Usage

```
opbase.data(ident, series_id = NULL, subset = NULL, verbose = FALSE,
    username = NULL, password = NULL, samples = NULL, exclude = NULL,
    include = NULL,  range = NULL, optim_test = TRUE, ...)
opbase.locations(ident, index_name, series_id = NULL, username = NULL,
    password = NULL)
opbase.obj.exists(ident, username = NULL, password = NULL)
opbase.series(ident, username = NULL, password = NULL, verbose = FALSE)
opbase.upload(input, ident = NULL, name = NULL, subset = NULL,
    obj_type = "variable", act_type = "replace", language = "eng",
    unit = "", who = NULL, rescol = NULL, chunk_size = NULL, verbose = FALSE,
    username = NULL, password = NULL, index_units = NULL, index_types = NULL)
```

Arguments

`ident`	Object ident as string (e.g. "op_en1390"). Optional when uploading within Opasnet; page ident will be taken from the page where the code is.
`series_id`	Series identifier as integer.
`index_name`	Column name (index) whose locations should be returned.
`subset`	Subset data name. Objects can have subsets of data, identified by subset names.
`verbose`	Flag to view detailed debug output.
`username`	Opasnet Base username.
`password`	Opasnet Base password.
`samples`	Limit the number of samples in result. Default is to get them all.
`exclude`	Filter result by excluding rows that contain locations defined here as list. Works only with entity type indices!
`include`	Filter result by only including rows that contain locations defined here as list. Works only with entity type indices!
`range`	Filter result by setting ranges for index location values. Works only with number and time type indices!
`optim_test`	Generally faster download, slower only when downloading large probability distributions from the database.
`input`	Input data as `data.frame`.
`name`	Object name for upload.
`obj_type`	Object type string: ’variable’, ’study’, ’method’, ’assessment’, ’class’, ’nugget’ or ’encyclopedia’.
`act_type`	Act type string: ’replace’ or ’append’. Replace type uploads data to new series. Append adds new act to latest series.
`language`	Data language identifier string in ISO 639.2 standard.
`unit`	String identifying the result unit(s).
`who`	Name or alias of the data uploading person.
`rescol`	Name of the result column index.
`chunk_size`	Size of upload data chunk in rows.
`index_units`	Units for indices in vector as strings. E.g. `c(’cm2’,’m2’,’ug/m3’)`.
`index_types`	Types for indices in vector as strings. Possible types are: ’entity’ for limited set of locations, ’number’ for real numbers and ’time’ for date time strings. E.g. `c(’entity’,’entity’,’number’)`.
`...`	Excess arguments are ignored.

Details

This family of functions provide access to Opasnet Base -database. Opasnet Base is the database used for storing Opasnet data. Use the `opbase.data` function to read data from the database and the `opbase.upload` function to upload data to the database. Note that uploading data from local R-installation requires Opasnet Base username and password. These can be obtained only by trusted people.

Exclude and include syntax: `list = (’<index name 1>’ = c(’<location value 1>’, ’<location value 2>’,..`
`’<location value 2>’,...), ...)`

Range syntax: `list = (’<index name 1>’ = c(<range from>|NA, <range to>|NA), ’<index name 2>’ = c(<r`

See also: http://www.loc.gov/standards/iso639-2/php/code_list.php
http://en.opasnet.org/

Value

`opbase.data`	Returns `data.frame` containing the query result data.
`opbase.locations`	Returns list of locations and their ids (as keys).
`opbase.obj.exists`	Returns TRUE if object exists, FALSE if not.
`opbase.series`	Returns vector of series ids.
`opbase.upload`	Returns total number of data rows uploaded.

Author(s)

E. Happonen <`einari.happonen@thl.fi`>

Examples

```
library(OpasnetUtils)

# Read
opbase.data('op_en1390')
opbase.data('op_en2949', subset='2012', include = list('KUNTA' = 322),
    range = list('ID_NRO' = c(20000, 30000), 'XKOORD' = c(NA,244000)))

# Write (works only within Opasnet when username nor password is given)
input <- matrix(c('male', 12334435.123, 22, 'female', 234345.23423, 33),
    ncol=3, byrow=TRUE)
colnames(input) <- c("Sex","Some number","result")
input <- as.data.frame(input)
#res <- opbase.upload(input, ident="op_en1390", name = "Sandbox TEST",
#    index_types = c('entity','number'), unit = "Age", who='Tester person')
```

---

oprint *Print ovariables or data frames in html format.*

Description

This function uses `xtable` to output `ovariable`s or `data.frame`s as html formatted tables.

Usage

```
oprint(x, show_all = FALSE, sortable = TRUE, ...)
```

Arguments

`x`	`ovariable` or `data frame`.
`show_all`	if `TRUE` all data rows are printed, else only first thousand rows get printed (default).
`sortable`	if `TRUE` output table is made sortable.
`...`	arguments can be passed to `xtable`

Details

If argument x is an `ovariable`, its output-slot gets printed. If output-slot is empty, `EvalOutput` will be automatically executed to generate output. This function is aimed for being used within Opasnet only! R console will print out html markup.

See also: `http://en.opasnet.org/`

Value

Input data as html formatted table string.

Author(s)

E. Happonen `<einari.happonen@thl.fi>`

See Also

`xtable`

Examples

```
library(OpasnetUtils)
x <- data.frame(c(1,2),c(2,4))
oprint(x)
```

`op_base` *Functions for Interaction with the Opasnet Base (obsolete)*

Description

A collection of functions used in Opasnet for database interaction. Includes functions for fetching datasets, exploring the dimensions of Opasnet variables and writing objects into the database.

This function family has been replaced by the `opbase` family

Usage

```
op_baseGetData(dsn, ident, ...)
op_baseGetLocs(dsn, ident, ...)
```

Arguments

`dsn`	a defined Data Service Name (in ODBC) to use
`ident`	object identifier in Opasnet (or other)
`...`	arguments for opbase

Details

Obsolete.

See also: `http://en.opasnet.org/w/Opasnet_Base_Connection_for_R`

Value

`op_baseGetData`	Returns data as a data.frame.
`op_baseGetLocs`	Returns dimension information as a data.frame.
`op_baseWrite`	Returns 0 if successful.

Author(s)

Teemu Rintala, <`teemu.rintala@thl.fi`>

Examples

```
## Not run: op_baseGetLocs("opasnet_base", "Op_en4723")
## Not run: asthma <- op_baseGetData("opasnet_base", "Op_en4723", exclude = 48823)
```

`orbind` *Rowbinding ovariables*

Description

Combine two `ovariables` or `data.frames` using rbind even if columns differ

Usage

```
orbind(x, y)
```

Arguments

`x`	first object
`y`	second object

Details

Missing columns from each `ovariable` are added to the other and filled with `NA`.

See also: `http://en.opasnet.org/`

Value

Returns a `data.frame`

Author(s)

J. Tuomisto <`jouni.tuomisto@thl.fi`>

`Ovariable` *Ovariable constructor*

Description

Create `ovariable`s more conveniently

Usage

```
Ovariable(name = character(), data = data.frame(),
    formula = function(...) {0}, dependencies = data.frame(),
    ddata = character(), output = data.frame(), subset = NULL,
    getddata = TRUE, save = FALSE, public = TRUE, ...)
```

Arguments

`name`	`character` string for the name slot, should match object name
`data`	`data.frame` for the data slot
`formula`	`function` for the formula slot
`dependencies`	`data.frame` for the dependencies slot
`ddata`	`character` string specifying an Opasnet page identifier (Op_enXXXX) for the ddata slot
`output`	`data.frame` for the output slot
`subset`	`character` string specifying an Opasnet Base subset (See `opbase.data` for details)
`getddata`	if TRUE dynamic data link will be activated immediately, which means that by default data will not be refreshed at model runtime
`save`	if TRUE resulting `ovariable` will be saved on the server
`public`	if TRUE `objects.store` is used instead of `objects.put` (the former stores the run key in a public database)
`...`	more arguments can be passed onto `objects.store` and `objects.put` in case `save == TRUE`.

Details

Just a regular constructor with integrated dynamic data link activation and storing options.

See also: `http://en.opasnet.org/`

Value

Returns an `ovariable`.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

See Also

`ovariable-class`

Examples

```
## Not run: Ovariable("A", ddata = "Op_en5674", getddata = TRUE)
## Not run: k <- Ovariable("k", output = data.frame(B = "a", Result = 1))
```

`ovariable-class` *Class* `"ovariable"`

Description

Standard modelling variables for the Opasnet modelling framework

Objects from the Class

Objects can be created by calls of the form `new("ovariable", ...)`. Or by using the `Ovariable`-constructor.

Slots

`name`**:** Object of class `"character"` name of variable, should match object name

`output`**:** Object of class `"data.frame"` output from formula and/or data operations

`data`**:** Object of class `"data.frame"` data describing the variable, should have a "Result" column

`marginal`**:** Object of class `"logical"` identifies output columns which are considered indices

`formula`**:** Object of class `"function"` a function that produces a `data.frame` that describes this variable

`dependencies`**:** Object of class `"data.frame"` list of variables that are used in `formula`, format is described in details for `Fetch`

`ddata`**:** Object of class `"character"` specifies an Opasnet page identifier (Op_enXXXX) which will be used to download most recent data on this variable in the Opasnet database

Methods

Math `signature(x = "ovariable")`: `Math` will be applied on Result column of output

merge `signature(x = "data.frame", y = "ovariable")`: `data.frame` will be converted to `ovariable` (with only output slot defined) and then merged

merge `signature(x = "numeric", y = "ovariable")`: `numeric` is converted to `data.frame` and then to `ovariable` and then merged

merge `signature(x = "ovariable", y = "data.frame")`: same as above

merge `signature(x = "ovariable", y = "numeric")`: same as above

merge `signature(x = "ovariable", y = "ovariable")`: output slots will be merged with all = TRUE, a blank ovariable with only output defined is returned

Ops `signature(e1 = "numeric", e2 = "ovariable")`: `numeric` is converted to `data.frame` and then to `ovariable` and then operated

Ops `signature(e1 = "ovariable", e2 = "numeric")`: same as above

Ops `signature(e1 = "ovariable", e2 = "ovariable")`: the `ovariables` are merged and then the two Result columns are operated unto, the result is saved in another Result column (or the same if they are not named: "Var1Result" vs "Result")

plot `signature(x = "ovariable")`: plots a simple comparison between sources (data vs formula)

summary `signature(object = "ovariable")`: returns a `data.frame`. Takes `function_names` and `marginals` as extra arguments. The former matches `character vector` elements into functions which will be `tapplied` with. The latter matches `character vector` elements to output `data.frame` columns which define INDEX. The default is to `tapply` over iterations using `mean`, `sd`, `min`, `quantile(probs=0.025)`, `median`, `quantile(probs=0.975)` and `max`.

Author(s)

T. Rintala <`teemu.rintala.a@gmail.com`>

See Also

`Ovariable`

`result` *Access result vector of an* `ovariable`

Description

A shortcut to the Result column of the `data.frame` in the output slot of an `ovariable`.

Usage

```
result(e1)
```

Arguments

`e1`	an `ovariable`

Details

See also: `http://en.opasnet.org/`

Value

Returns a `numeric` vector

Author(s)

J. Tuomisto <`jouni.tuomisto@thl.fi`>

Examples

```
a <- Ovariable("a", output = data.frame(Result = 1))
result(a)
result(a) <- 10 * result(a)
a@output
```

tidy *Format database output for use in ovariables*

Description

Wrapper for various standard operations applied on ovariable data from the OpasnetBase.

Usage

```
tidy(data, objname = "", idvar = "Obs", direction = "wide",
    widecol = NULL, base1 = FALSE)
```

Arguments

data	data.frame to be formatted
objname	character prefix to be added to variable specific rows like "Result" and "Unit"
idvar	reshape idvar
direction	reshape direction
widecol	reshape timevar, specifiec column to be expanded
base1	compatibility with obsolete database

Details

Uses reshape, renames "Result" and "Unit" columns and gets rid of unwanted columns from old database merged data.

See also: http://en.opasnet.org/

Value

Returns a data.frame

Author(s)

T. Rintala <teemu.rintala.a@gmail.com>

Examples

```
var1 <- opbase.data("Op_en5674")
var1 <- tidy(var1, "var1")
var1
```

Index