

# Package ‘OpasnetUtils’

June 3, 2015

**Type** Package

**Title** Opasnet Modelling Environment Utility Functions

**Version** 1.2.0

**Author** Teemu Rintala, Einari Happonen, Jouni Tuomisto

**Maintainer** Teemu Rintala <teemu.rintala.a@gmail.com>

**Description** Contains tools for open assessment and modelling in Opasnet, a wiki-based web site and workspace for societal decision making (see <[http://en.opasnet.org/w/Main\\_Page](http://en.opasnet.org/w/Main_Page)> for more information). The core principle of the workspace is maximal openness and modularity. Variables are defined on public wiki pages using wiki inputs/tables, databases and R code. This package provides the functionality to download and use these variables. It also contains health impact assessment tools such as spatial methods for exposure modelling.

**Imports** methods, rjson, RCurl, reshape2, triangle, httpRequest, digest, xtable, ggplot2, rgdal, sp, plyr

**License** GPL-3

**LazyLoad** yes

**URL** [http://en.opasnet.org/w/Main\\_Page](http://en.opasnet.org/w/Main_Page) ,  
[http://en.opasnet.org/w/Modelling\\_in\\_Opasnet](http://en.opasnet.org/w/Modelling_in_Opasnet) ,  
<http://en.opasnet.org/w/OpasnetUtils>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2015-06-03 14:30:22

## R topics documented:

Check . . . . .	2
CollapseMarginal . . . . .	4
CollapseTableParser . . . . .	5
ComputeDependencies . . . . .	6
continuousOps . . . . .	7

convert.units . . . . .	8
ddata_apply . . . . .	9
DecisionTableParser . . . . .	10
dropall . . . . .	11
EvalOutput . . . . .	12
Fetch . . . . .	13
fillna . . . . .	14
GIS . . . . .	15
interpret . . . . .	17
oapply . . . . .	18
objects . . . . .	19
odebug . . . . .	21
odecision-class . . . . .	22
odist . . . . .	23
opasnet . . . . .	24
opbase . . . . .	26
openv . . . . .	28
oprint . . . . .	29
Optimize . . . . .	30
op_base . . . . .	31
orbind . . . . .	32
Ovariable . . . . .	32
ovvariable-class . . . . .	34
result . . . . .	35
tidy . . . . .	36
unkeep . . . . .	37
<b>Index</b>	<b>38</b>

---

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 `ovvariable` 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 `ovvariable` with possible adjustments.

**Author(s)**

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

**Examples**

```
#library(OpasnetUtils)
#
Deca <- new(
  "odecision",
  dectable = data.frame(
    Decision = "Dec1",
```

```

Option = c("do it", "BAU"),
Cell = c("", ""),
Change = c("Multiply", "Identity"),
Result = c("2", "1")
)
)
openv.setN(10)
a <- Ovariable("a", data = data.frame(A = c("A", "B"), Result = c("1-2", "2-4")))

b <- Ovariable("b", formula = function(...){2*a}, dependencies = data.frame(Name = "a"))

b <- EvalOutput(b)

b
oapply(a, a@output[c("A", "Dec1")], mean)

```

---

CollapseMarginal

*Collapse marginals*


---

### Description

Apply functions (only sample at the moment) over ovariabile indices

### Usage

```
CollapseMarginal(variable, cols, fun = "mean", probs = NULL, ...)
```

### Arguments

variable	an ovariable
cols	vector of column names or indices to collapse
fun	character name of function to be used to summarize
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), marginal = c(TRUE, TRUE, FALSE))
CollapseMarginal(test, "City", "mean")
```

---

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 <a href="#">Fetch</a> 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>

---

`continuousOps`*Simple continuous index merge and Ops*

---

**Description**

`continuousOps` merges two ovariables by continuous indices and performs an operation.

**Usage**

```
continuousOps(O1, O2, fun, cols = NULL)
```

**Arguments**

<code>O1</code>	ovvariable
<code>O2</code>	ovvariable
<code>fun</code>	function or function name (character) that is to be performed. Typically '*', '+', or some other Ops.
<code>cols</code>	character vector of column names that contain continuous indices. Only needed if there are > 1 indices and the order is critical. Otherwise, shared continuous indices are identified automatically.

**Details**

`O1` is of main interest, while `O2` has information that links to `O1` via continuous index or indices. All locations in these continuous indices of `O1` are created for `O2` assuming that the value in the previous location of `cols` applies. Note that this is asymmetric. Locations in `O2` that are missing from `O1` are omitted. `continuousOps` assumes that all continuous indices are in the same dimension, the first one being the main index. Additional indices affect the outcome only if there are (approximate) ties. Therefore, avoid using this with several continuous indices. However, if continuous indices are NOT shared by both `O1` and `O2`, they cause no trouble.

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

**Value**

Returns an `Ovariable`

**Author(s)**

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

---

`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

<code>x</code>	numeric vector with values to be converted
<code>tounit</code>	character vector of the new units to be used
<code>fromunit</code>	character vector or factor with the current units

## Details

Uses the table in [en.opasnet.org/w/Unit\\_conversions](http://en.opasnet.org/w/Unit_conversions) for the conversions, which can be edited to contain relevant units.

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

## Value

Returns a `data.frame`

## Author(s)

J. Tuomisto <[jouni.tuomisto@thl.fi](mailto:jouni.tuomisto@thl.fi)>

## Examples

```
## Not run:  
convert.units(1, tounit = c("pg", "l"), fromunit = "ug /m3")  
  
## End(Not run)
```



---

ddata_apply	<i>Dynamic data link activation</i>
-------------	-------------------------------------

---

### Description

Fetches the latest data associated with an [ovvariable](#) from the OpasnetBase if available

### Usage

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

### Arguments

<code>ovvariable</code>	an <a href="#">ovvariable</a> with the <code>ddata</code> slot defined as <code>page_id</code> i.e. "Op_en1000"
<code>ddata_tidy</code>	TRUE to run <a href="#">tidy</a> on downloaded data
<code>force_ddata</code>	if TRUE, dynamic data links are used even if the <code>data</code> slot of an <a href="#">ovvariable</a> is already defined
<code>...</code>	excess parameters are ignored

### Details

This function is mostly used internally

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

### Value

Returns the input [ovvariable](#). (Re)defines the `data` slot if it is not already defined and `ddata` is available.

### Author(s)

T. Rintala <[teemu.rintala.a@gmail.com](mailto:teemu.rintala.a@gmail.com)>

### See Also

[ovvariable](#)

---

DecisionTableParser    *Parse data.frame for decisions*

---

## Description

Parses a `data.frame` into `odecisions`.

## 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 [ovvariable](#), which usually means recursively evaluating any dependent variables as well.

### Usage

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

### Arguments

variable	an <a href="#">ovvariable</a>
fillna	if TRUE, <a href="#">fillna</a> 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 <a href="#">ovvariable</a> 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 [ovvariable](#) handling. It runs [ComputeDependencies](#) first, produces a data.frame by combining the return values from [interpreting](#) 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 Checks besides [CheckMarginals](#).

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

### Value

Returns the input [ovvariable](#) 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 <a href="#">EvalOutput</a> 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 <a href="#">EvalOutput</a> 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")
## Not run:
Fetch(deps)
exposure

## End(Not run)

# If variable exists
exposure <- 1
Fetch(deps)
exposure # by default existing variables are not changed
```

---

fillna

*Interpreting empty locations in indices*

---

### Description

Copies result rows that have NAs 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)
```

**Description**

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

**Usage**

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

**Arguments**

Emission	numeric, emission in any units; can be an ovariable as well
Concentration.matrix	<a href="#">ovvariable</a> concentration matrix, usually produced by <code>GIS.Concentration.matrix</code>
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
debug	use TRUE to turn debug prints on
...	excess arguments are ignored or passed to <a href="#">tidy</a> on data download

**Details**

The concentration matrix is computed using PILTTI source-receiver-matrices ([http://en.opasnet.org/w/Piltti\\_source-receptor\\_matrix](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 ([http://en.opasnet.org/w/Special:Opasnet\\_Base?id=op\\_en2949.2012](http://en.opasnet.org/w/Special:Opasnet_Base?id=op_en2949.2012)).

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. All population data that matches cells defined by the concentration matrix is included in the output.

**Author(s)**

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

**Examples**

```
## Not run:
# Excerpt from http://en.opasnet.org/w/Health\_impacts\_of\_fine\_particles\_in\_Rauma
# (not evaluated)

# Paasto Emissions

Paasto <- new(
  "ovvariable",
  name = "Paasto",
  dependencies = data.frame(Name = "tieliikennepaastot", Key = "0194s0uuucjq8Wi"),
  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(
  "ovvariable",
  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(
  "ovvariable",
  name = "Altistuminen",
  dependencies = data.frame(
    Name = c("Pitoisuus", "LO", "LA")
  ),
  formula = function(dependencies, ...) {
    ComputeDependencies(dependencies, ...)

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

    return(out)
  }
)

## End(Not run)

```

---

interpret

*Parse human readable distribution definitions*


---

## Description

Interpret textual data into probability distributions using regular expressions.

## Usage

```
interpret(idata, N = NULL, rescol = "Result", debug = FALSE, ...)
```

## Arguments

idata	input, character or data.frame
N	number of iterations
rescol	name of result column, defaults to "Result"
debug	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.

Numeric values are left as is.

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 NAs.

**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, drop_na = TRUE,
       use_aggregate = TRUE, ..., simplify = TRUE)
ooapply(X, cols, FUN = "sum", ...)
```

**Arguments**

X	an <a href="#">ovvariable</a>
INDEX	list of factors, like <a href="#">tapply</a>
FUN	function to apply, <code>ooapply</code> takes character input (Only "sum", "mean", "min", "max" and "prod" allowed.)
cols	names of columns to be removed (reverse INDEX)
drop_na	if TRUE removes NA from the result automatically
use_aggregate	if TRUE uses <a href="#">aggregate</a> to perform apply, which is considerably faster with sparse variables
...	optional arguments to FUN
simplify	like <a href="#">tapply</a>

**Details**

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

**Value**

Returns an `ovvariable`, with output slot `tapplyed` and marginal adjusted accordingly. `ooapply` is a memory-saving variant of `oapply` when there is exactly one row for each unique combination. `oapply` with `use_aggregate` is fastest in most cases. Any extraneous columns are lost, as with [tapply](#) and [aggregate](#).

**Author(s)**

T. Rintala <[teemu.rintala.a@gmail.com](mailto:teemu.rintala.a@gmail.com)>

**Examples**

```
a <- new("ovvariable", 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)
```

**Description**

Library for using R objects (like [ovvariables](#).) 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**

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

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

**Author(s)**

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

**See Also**

[load](#)  
[save](#)  
[serialize](#)  
[AES](#)

**Examples**

```
# 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 = "Yeah baby yeah!")
## Not run:
objects.store(x, y)

## End(Not run)
# Fetching can be done also from local R installation.
## Not run:
objects.latest("Op_en1390", "objs_save_test")

## End(Not run)
print(x)
print(y)

# Object encrypt and decrypt

key <- "1234567890abcdef"

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

---

odebug

*Debugging ovariables*

---

**Description**

Check shared indices, dropped locations and some statistical metrics

**Usage**

```
odebug(x, variance = FALSE)
```

**Arguments**

x                    ovariable to be debugged  
variance            If TRUE runs variance analysis between indices of the given ovariable.

**Details**

Checks output lengths, number of NAs, common and matching marginals, missing locations in common marginals and performs variance analysis if required. Variance analysis is only performed up to second order, the intended use is to quantify overlap between indices.

**Value**

List of outputs.

**Author(s)**

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

**Examples**

```
test <- data.frame(
  A = c("x", "y", "z"),
  B = rep(c("a", "b", "c"), each = 3),
  C = rep(c("1", "2", "3"), each = 3*3),
  testResult = runif(1*3*3*3)
)
test <- Ovariable(
  name = "test",
  output = test,
  marginal = c(TRUE, TRUE, TRUE, FALSE)
)
odebug(test, variance = TRUE)
```

---

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 [ovvariable](#)

`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>

---

odist

*Random sampling using ovariables as parameters*

---

### 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

```
odirichlet(a, n = 0, ...)
```

### Arguments

<code>a</code>	ovvariable containing distribution parameters
<code>n</code>	numeric, number of samples. If 0, <code>openv\$N</code> is used instead.
<code>...</code>	arguments passed to <a href="#">oapply</a>

### Details

`Odirichlet` is based on functions `ddirichlet` and `rdirichlet` from `gtools` `dirichlet.R` (originally contributed by Ian Wilson). The "Dirichlet function" is the multidimensional generalization of the beta distribution: it's the Bayesian canonical distribution for the parameter estimates of a multinomial distribution. `Odirichlet` samples from the dirichlet distribution given parameter vectors that are processed by `oapply`.

**Value**

ovvariable

**Author(s)**

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

**Examples**

```
openv$N <- 5

test <- Ovariable(
  output=data.frame(
    a = 1:4,
    b = rep(letters[1:4], each = 4),
    c = rep(toupper(letters[1:4]), each = 4^2),
    Result = 1:(4^3)),
  marginal=c(rep(TRUE, 3), FALSE))
out <- odirichlet(test, cols = "a")
oapply(out, FUN = sum, cols = "a")

test <- Ovariable(
  output=data.frame(
    a = 1:4,
    b = rep(letters[1:4], each = 4),
    Iter = rep(toupper(letters[1:4]), each = 4^2),
    Result = 1:(4^3)),
  marginal=c(rep(TRUE, 3),FALSE))
out <- odirichlet(test, cols = "a")
oapply(out, FUN = sum, cols = "a")
```

---

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 = "", ...)
opasnet.page(pagename, wiki = "")
```



## Arguments

filename	Path to file in Opasnet after the "images/" part.
pagename	Name of a page in Opasnet.
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 <code>read.table</code> 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. `opasnet.page` downloads wiki pages using `getURL` from `RCurl`.

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

## Value

<code>opasnet.data</code>	Returns the file data as is.
<code>opasnet.csv</code>	Returns <code>data.frame</code> parsed from csv file.
<code>opasnet.page</code>	Returns the web page as a single string.

## Author(s)

E. Happonen <einari.happonen@thl.fi>, T. Rintala <teemu.rintala.a@gmail.com>

## See Also

[read.table](#)

## Examples

```
## Not run:
opasnet.csv("4/49/Test.zip", wiki = "opasnet_fi", unzip = 'ejpop.csv', sep=';')
opasnet.data("c/cc/Test_bugs_model.txt", wiki = "opasnet_en")

## End(Not run)
```

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.indices(ident, act = NULL, 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.
act	Act 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 <code>data.frame</code> .

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>', ...), <'index name 2>' = c(<'location value 1>', <'location value 2>', ...), ...)`

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

See also: [http://www.loc.gov/standards/iso639-2/php/code\\_list.php](http://www.loc.gov/standards/iso639-2/php/code_list.php)  
<http://en.opasnet.org/>

### Value

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

### Author(s)

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

## Examples

```
## Not run:
# 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')

## End(Not run)
```

---

openv

*OpasnetUtils environment for mutable function parameters*

---

## Description

Container for model-wide parameters such as number of iterations used in random sampling.

## Usage

```
openv.setN(x)
```

## Arguments

x                    numeric, new default number of iterations

## Details

The environment is also automatically populated by CheckDecisions to communicate evaluation to other branches of the recursion tree. The value N in this environment is used by [interpret](#) when N is not given in an EvalOutput(...) call.

## Value

No return value.

## Author(s)

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

**Examples**

```

openv$N
openv.setN(1)
openv$N
openv$N <- 2
openv$N

```

---

oprint	<i>Print ovariables or data frames in html format.</i>
--------	--

---

**Description**

Oprint uses `xtable` to output `ovvariables` or `data.frames` or `matrix` as html formatted tables. In other cases uses regular print but adds formatting.

**Usage**

```

oprint(x, pre = TRUE, ...)
oprint.table(x, show_all = FALSE, sortable = TRUE, ...)

```

**Arguments**

<code>x</code>	Any object with a <code>print</code> or <code>oprint</code> method.
<code>pre</code>	TRUE for <code>&lt;pre&gt;</code> formatting.
<code>show_all</code>	if TRUE all data rows are printed, else only first thousand rows get printed (default).
<code>sortable</code>	if TRUE output table is made sortable.
<code>...</code>	extra arguments are passed to <code>oprint.table</code> and/or <code>xtable</code>

**Details**

If argument `x` is an `ovvariable`, 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. `oprint.table` is not exported. Use `oprint` instead and `...` to pass arguments `show_all` and `sortable`.

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**

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

---

Optimize

*Simple optimization between discrete index locations*

---

**Description**

Finds the minimal value of an `ovvariable` along given indices. Returns sought rows and drops the rest.

**Usage**

```
Optimize(...)
Minimize(data, indices)
```

**Arguments**

... arguments passed to `Minimize`  
data [ovvariable](#) or `data.frame` of suitable format (must contain a "Result" column)  
indices character vector of index names meant to be retained (apply `INDEX`)

**Details**

Uses [tapply](#) and [which.min](#).

**Value**

Returns a `data.frame` which is a subset of the original `data.frame` or `ovvariable@output`.

**Author(s)**

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

**Examples**

```
a <- Ovariable(output = data.frame(A = letters[c(1,1,2,2)], B = c(1,2,1,2), Result = 1:4))
Minimize(a, "B")
Minimize(a, "A")
```

---

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](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 [ovvariable](#) 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 [ovvariables](#) more conveniently

**Usage**

```
Ovariable(name = character(), data = data.frame(),  
  formula = function(...) {}, dependencies = data.frame(),  
  ddata = character(), output = data.frame(), marginal = logical(),  
  subset = character(), 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
marginal	logical for the marginal slot
subset	character string specifying an Opasnet Base subset (See <a href="#">opbase.data</a> 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 <a href="#">objects.store</a> is used instead of <code>objects.put</code> (the former stores the run key in a public database)
...	more arguments can be passed onto <code>objects.store</code> and <code>objects.put</code> in case <code>save == TRUE</code> .

**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**

[ovvariable-class](#)

**Examples**

```
## Not run: Ovariable("A", ddata = "Op_en5674", getddata = TRUE)
k <- Ovariable("k", output = data.frame(B = "a", Result = 1))
o <- Ovariable("o", output = data.frame(B = "a", Result = 2))
k*o
```

---

ovvariable-class	Class "ovvariable"
------------------	--------------------

---

### Description

Standard modelling variables for the Opasnet modelling framework

### Objects from the Class

Objects can be created by calls of the form `new("ovvariable", ...)`. 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 = "ovvariable"): [Math](#) will be applied on Result column of output

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

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

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

**merge** signature(x = "ovvariable", y = "numeric"): same as above

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

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

**Ops** signature(e1 = "ovvariable", e2 = "numeric"): same as above

**Ops** signature(e1 = "ovvariable", e2 = "ovvariable"): the ovvariables 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 = "ovvariable"): plots a simple comparison between sources (data vs formula)

**summary** signature(object = "ovvariable"): returns a data.frame. Takes function\_names and marginals as extra arguments. The former matches character vector elements into functions which will be applied with. The latter matches character vector elements to output data.frame columns which define INDEX. The default is to apply 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	<i>Access result vector of an ovariable</i>
--------	---

---

**Description**

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

**Usage**

```
result(e1)
```

**Arguments**

e1                    an [ovvariable](#)

**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

<code>data</code>	data.frame to be formatted
<code>objname</code>	character prefix to be added to variable specific rows like "Result" and "Unit"
<code>idvar</code>	<a href="#">reshape</a> idvar
<code>direction</code>	<a href="#">reshape</a> direction
<code>widecol</code>	<a href="#">reshape</a> timevar, specific column to be expanded
<code>base1</code>	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

```
## Not run:  
var1 <- opbase.data("Op_en5674")  
var1 <- tidy(var1, "var1")  
var1  
  
## End(Not run)
```

---

unkeep	<i>Remove unnecessary columns</i>
--------	-----------------------------------

---

**Description**

Removes columns in ovariable outputs

**Usage**

```
unkeep(X, cols = NA, sources = FALSE, prevresults = FALSE)
```

**Arguments**

X	an ovariable to be worked on
cols	character vector of columns to be removed
sources	if TRUE removes "...Source"-columns
prevresults	if TRUE removes Result-columns of other variables

**Value**

ovvariable

**Author(s)**

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

# Index

## \*Topic **classes**

odecision-class, [22](#)

ovvariable-class, [34](#)

## \*Topic **database**

op\_base, [31](#)

opbase, [26](#)

AES, [21](#)

aggregate, [19](#)

Check, [2, 6](#)

CheckCollapse, [5](#)

CheckCollapse (Check), [2](#)

CheckDecisions, [10, 22](#)

CheckDecisions (Check), [2](#)

CheckInput (Check), [2](#)

CheckMarginals, [12](#)

CheckMarginals (Check), [2](#)

CollapseMarginal, [3, 4](#)

CollapseTableParser, [5](#)

combine (orbind), [32](#)

ComputeDependencies, [6, 12, 13](#)

continuousOps, [7](#)

convert.units, [8](#)

convert.units, data.frame-method  
(convert.units), [8](#)

convert.units, ovvariable-method  
(convert.units), [8](#)

ddata\_apply, [9](#)

DecisionTableParser, [3, 10, 22](#)

dropall, [11, 14](#)

EvalOutput, [3, 6, 12, 13, 29](#)

Fetch, [6, 13, 34](#)

Fetch2 (Fetch), [13](#)

fill.na.merge (fillna), [14](#)

fillna, [12, 14](#)

GIS, [15](#)

interpret, [12, 17, 28](#)

interpret, character-method (interpret),  
[17](#)

interpret, numeric-method (interpret), [17](#)

load, [21](#)

Math, [34](#)

Math, ovvariable-method  
(ovvariable-class), [34](#)

merge, data.frame, ovvariable-method  
(ovvariable-class), [34](#)

merge, numeric, ovvariable-method  
(ovvariable-class), [34](#)

merge, ovvariable, data.frame-method  
(ovvariable-class), [34](#)

merge, ovvariable, numeric-method  
(ovvariable-class), [34](#)

merge, ovvariable, ovvariable-method  
(ovvariable-class), [34](#)

Minimize (Optimize), [30](#)

oapply, [18, 23](#)

objects, [19](#)

objects.store, [33](#)

odebug, [21](#)

odecision, [10](#)

odecision (odecision-class), [22](#)

odecision-class, [22](#)

odirichlet (odist), [23](#)

odist, [23](#)

oapply (oapply), [18](#)

op\_base, [31](#)

op\_baseGetData (op\_base), [31](#)

op\_baseGetLocs (op\_base), [31](#)

opasnet, [24](#)

opbase, [26, 31](#)

opbase.data, [33](#)

openv, [28](#)

oprint, [29](#)

oprint,data.frame-method (oprint), 29  
oprint,matrix-method (oprint), 29  
oprint,ovvariable-method (oprint), 29  
oprint.table (oprint), 29  
Ops,numeric,ovvariable-method  
    (ovvariable-class), 34  
Ops,ovvariable,numeric-method  
    (ovvariable-class), 34  
Ops,ovvariable,ovvariable-method  
    (ovvariable-class), 34  
Optimize, 30  
orbind, 32  
Ovariable, 32, 34, 35  
ovvariable, 3, 6, 9, 12, 15, 19, 20, 23, 29, 30,  
    32, 35  
ovvariable (ovvariable-class), 34  
ovvariable-class, 34  
  
plot,ovvariable,ANY-method  
    (ovvariable-class), 34  
plot,ovvariable-method  
    (ovvariable-class), 34  
  
read.table, 25  
reshape, 36  
result, 35  
result<- (result), 35  
  
save, 21  
serialize, 21  
summary,ovvariable-method  
    (ovvariable-class), 34  
  
tapply, 19, 30  
tidy, 9, 15, 36  
  
unkeep, 37  
  
which.min, 30  
  
xtable, 29, 30