

Package ‘gRaven’

May 8, 2026

Type Package

Title Bayes Nets: 'RHugin' Emulation with 'gRain'

Version 1.1.10

Date 2024-10-10

Maintainer Peter Green <P.J.Green@bristol.ac.uk>

Description Wrappers for functions in the 'gRain' package to emulate some 'RHugin' functionality, allowing the building of Bayesian networks consisting on discrete chance nodes incrementally, through adding nodes, edges and conditional probability tables, the setting of evidence, both 'hard' (boolean) or 'soft' (likelihoods), querying marginal probabilities and normalizing constants, and generating sets of high-probability configurations. Computations will typically not be so fast as they are with 'RHugin', but this package should assist users without access to 'Hugin' to use code written to use 'RHugin'.

License GPL (>= 2)

Depends gRain (>= 1.4.4)

Imports gRbase, methods, rlang

NeedsCompilation no

Author Peter Green [aut, cre],
Therese Graversen [ctb],
Soren Hojsgaard [ctb]

Repository CRAN

Date/Publication 2024-10-11 08:10:05 UTC

Contents

gRaven-package	2
add.edge	3
add.node	4
compile.gRaven	6
compress	7
get.belief	7
get.nodes	8

get.normalization.constant	9
get.parents	10
get.states	11
hugin.domain	12
initialize.domain	13
list.domains	14
map.configurations	15
print.gRaven	16
propagate.gRaven	17
set.finding	18
set.table	19
simulate.gRaven	20
summary.gRaven	21

Index	23
--------------	-----------

gRaven-package	<i>Bayes Nets: 'RHugin' Emulation with 'gRain'</i>
----------------	--

Description

Wrappers for functions in the 'gRain' package to emulate some 'RHugin' functionality, allowing the building of Bayesian networks consisting on discrete chance nodes incrementally, through adding nodes, edges and conditional probability tables, the setting of evidence, both 'hard' (boolean) or 'soft' (likelihoods), querying marginal probabilities and normalizing constants, and generating sets of high-probability configurations. Computations will typically not be so fast as they are with 'RHugin', but this package should assist users without access to 'Hugin' to use code written to use 'RHugin'.

Author(s)

Maintainer: Peter Green <P.J.Green@bristol.ac.uk>

Examples

```
library(gRaven)
rm(list=ls(all=TRUE))

yn <- c("yes", "no")
chest<-hugin.domain()

add.node(chest, "asia", states=yn)
add.node(chest, "tub", states=yn)
add.node(chest, "smoke", states=yn)
add.node(chest, "lung", states=yn)
add.node(chest, "bronc", states=yn)
add.node(chest, "either", states=yn)
add.node(chest, "xray", states=yn)
add.node(chest, "dysp", states=yn)
```

```

add.edge(chest,"tub","asia")
add.edge(chest,"lung","smoke")
add.edge(chest,"bronc","smoke")
add.edge(chest,"either","lung")
add.edge(chest,"either","tub")
add.edge(chest,"xray","either")
add.edge(chest,"dysp","bronc")
add.edge(chest,"dysp","either")

set.table(chest,"asia",c(0.01,0.99))
set.table(chest,"tub",c(0.05,0.95,0.01,0.99))
set.table(chest,"smoke",c(0.5,0.5))
set.table(chest,"lung",c(0.1,0.9,0.01,0.99))
set.table(chest,"bronc",c(0.6,0.4,0.3,0.7))
set.table(chest,"either",c(1,0,1,0,1,0,0,1))
set.table(chest,"xray",c(0.98,0.02,0.05,0.95))
set.table(chest,"dysp",c(0.9,0.1,0.7,0.3,0.8,0.2,0.1,0.9))

chest
get.nodes(chest)
chest$states
sapply(get.nodes(chest),function(x) get.parents(chest,x))
compile(chest)
chest
get.belief(chest,"asia")
get.belief(chest,"tub")
get.belief(chest,"lung")
get.belief(chest,"bronc")
sapply(get.nodes(chest),function(x) get.belief(chest,x))

c2<-clone.domain(chest)

set.finding(chest,"asia","yes")
set.finding(chest,"dysp","yes")
propagate(chest)

get.belief(chest,"asia")
get.belief(chest,"tub")
get.belief(chest,"lung")
get.belief(chest,"bronc")

pEvidence(chest$net)
get.normalization.constant(chest)

pEvidence(c2$net,evidence=list(asia="yes",dysp="yes"))
get.normalization.constant(c2)

```

Description

Add or delete a directed edge from one or more parents.

Usage

```
add.edge(domain, child, parent)
delete.edge(domain, child, parent)
```

Arguments

domain	name of a gRaven domain
child	a character string containing the name of the child node.
parent	a character string specifying by name the parent nodes

Details

Emulates function of the same name in the RHugin package by calls to gRain functions

Value

a NULL value is invisibly returned.

Differences from RHugin

Deleting an edge nullifies the CPT for node n, if any.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo(chest.setup, package="gRaven")
chest
delete.edge(chest, "tub", "asia")
add.edge(chest, "bronc", "asia")
chest
```

add.node

Add or Delete Node in a gRaven domain

Description

Add or delete a node in a gRaven domain.

Usage

```
add.node(domain, name, category = c("chance",
  "decision", "utility", "function"), kind = c("discrete", "continuous", "other"),
  subtype, states)
delete.node(domain, name)
```

Arguments

domain	name of a gRaven domain
name	a character string containing the name of the node to be added or deleted
category	a character string specifying the category of the node
kind	a character string specifying the kind of the node
subtype	a character string, 'labeled', 'numbered' or 'boolean'
states	a character, numeric or logical vector listing the states for this node

Details

Emulates functions of the same name in the RHugin package by calls to gRain functions, `delete.node` deletes the specified node and all incident edges; nullifies all corresponding CPTs.

Value

a NULL value is invisibly returned.

Differences from RHugin

Only discrete chance nodes are currently handled in gRaven.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo(chest.setup, package="gRaven")
chest
add.node(chest, 'dummy', states=1:3)
delete.node(chest, 'either')
chest
```

`compile.gRaven`*Compile a gRaven domain*

Description

Compile a gRaven domain

Usage

```
## S3 method for class 'gRaven'  
compile(object, ...)  
check.compiled(object)
```

Arguments

<code>object</code>	name of gRaven domain
<code>...</code>	additional arguments to <code>compile</code>

Details

`check.compiled` compiles the domain if necessary, having first ensured that necessary information such as `cptables` is in place. If the domain is already compiled, `compile` triggers a warning, but proceeds to re-initialise the grain object `domain$net`, in particular deleting any evidence set.

Value

a NULL value is invisibly returned.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
chest<-hugin.domain()  
add.node(chest,"asia",states=c("yes","no"))  
add.node(chest,"tub",states=c("yes","no"))  
add.edge(chest,"tub","asia")  
chest  
compile(chest)  
chest  
set.finding(chest,"asia","yes")  
set.finding(chest,"dysp","no")  
propagate(chest)  
chest$nodes  
chest$states  
chest$parents
```

compress	<i>Dummy network compression routine</i>
----------	--

Description

Dummy routine.

Usage

```
compress(domain)
```

Arguments

domain name of gRaven domain

Details

compression in the sense used in Hugin is not available in gRain, so this function has no effect.

Value

1

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
compile(chest)
compress(chest)
```

get.belief	<i>Get beliefs in a gRaven domain</i>
------------	---------------------------------------

Description

Get beliefs (marginal joint probabilities) in a gRaven domain

Usage

```
get.belief(domain, nodes)
get.marginal(domain, nodes, class = c("data.frame", "table", "ftable", "numeric"))
```

Arguments

domain	name of gRaven domain
nodes	character vector of names of nodes
class	desired class of output

Details

Emulates functions of the same name in the RHugin package by calls to gRain functions. Unlike with RHugin, gRaven conditions on all entered evidence in reporting probabilities, not only propagated evidence.

Value

For `get.marginal`, a list with one component "table" which is a data frame, table, flat table or numeric vector of marginal joint probabilities, as specified by `class`. For `get.belief`, a vector in the case of a single node (if there is no evidence on that node. a vector of ones), otherwise as `get.marginal`.

Differences from RHugin

`get.belief` handles more than one node at a time.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
compile(chest)
chest
set.finding(chest, "asia", "yes")
set.finding(chest, "dysp", "no")
propagate(chest)
get.belief(chest, "asia")
get.belief(chest, "tub")
get.marginal(chest, c('asia', 'xray', 'tub'))
```

get.nodes

Get the nodes of a gRaven domain

Description

Get the nodes of a gRaven domain

Usage

```
get.nodes(domain)
```

Arguments

domain name of gRaven domain

Details

Emulates function of the same name in the RHugin package by calls to gRain functions

Value

Character vector of node names.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
get.nodes(chest)
chest$states
sapply(get.nodes(chest), function(x) get.parents(chest, x))
```

`get.normalization.constant`

Get the normalisation constant of a gRaven domain

Description

Get the normalisation constant of a gRaven domain

Usage

```
get.normalization.constant(domain, log = FALSE)
```

Arguments

domain name of a gRaven domain
log logical, should constant be returned on log scale?

Value

numeric, the value of the normalisation constant (or its logarithm). If no evidence has been entered, the constant is defined as 1.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```

yn <- c("yes", "no")
chest<-hugin.domain()

add.node(chest, "asia", states=yn)
add.node(chest, "tub", states=yn)
add.node(chest, "smoke", states=yn)
add.node(chest, "lung", states=yn)
add.node(chest, "bronc", states=yn)
add.node(chest, "either", states=yn)
add.node(chest, "xray", states=yn)
add.node(chest, "dysp", states=yn)

add.edge(chest, "tub", "asia")
add.edge(chest, "lung", "smoke")
add.edge(chest, "bronc", "smoke")
add.edge(chest, "either", "lung")
add.edge(chest, "either", "tub")
add.edge(chest, "xray", "either")
add.edge(chest, "dysp", "bronc")
add.edge(chest, "dysp", "either")

set.table(chest, "asia", c(0.01, 0.99))
set.table(chest, "tub", c(0.05, 0.95, 0.01, 0.99))
set.table(chest, "smoke", c(0.5, 0.5))
set.table(chest, "lung", c(0.1, 0.9, 0.01, 0.99))
set.table(chest, "bronc", c(0.6, 0.4, 0.3, 0.7))
set.table(chest, "either", c(1, 0, 1, 0, 1, 0, 0, 1))
set.table(chest, "xray", c(0.98, 0.02, 0.05, 0.95))
set.table(chest, "dysp", c(0.9, 0.1, 0.7, 0.3, 0.8, 0.2, 0.1, 0.9))

compile(chest)
get.normalization.constant(chest)

set.finding(chest, "asia", "yes")
set.finding(chest, "dysp", "yes")
get.normalization.constant(chest)

```

get.parents

Get the parents of a node in a gRaven domain

Description

Get the parents of a node in a gRaven domain

Usage

```
get.parents(domain, n, type = "parents")
```

Arguments

domain	name of gRaven domain
n	name of node
type	only "parents" is currently accepted

Value

A named list of character vectors of names of parents.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
get.nodes(chest)
chest$states
sapply(get.nodes(chest), function(x) get.parents(chest, x))
```

get.states	<i>Get states, edges and children in a gRaven domain</i>
------------	--

Description

Get states, edges and children in a gRaven domain

Usage

```
get.states(domain, nodes=domain$nodes)
get.children(domain, nodes)
get.edges(domain, nodes=domain$nodes)
```

Arguments

domain	name of gRaven domain
nodes	name of node(s)

Details

Emulates functions of the same name in the RHugin package by calls to gRain functions

Value

For `get.states`, vector of state values. For `get.edges`, a list with one element for each node in domain. Each element is in turn a list with a single element `edges` which is a character vector of names of the node's children. An empty vector indicates that the node has no children. For `get.children`, character vector of children's node names in the case of a single node, otherwise list of children of each node, each element being a character vector of names of the node's children.

Differences from RHugin

get.states and get.children allow more than one node; get.edges allows selecting nodes.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
get.states(chest, "asia")
get.children(chest, "asia")
get.edges(chest)
```

hugin.domain

Create or Clone a gRaven Domain

Description

Create or Clone a gRaven domain object.

Usage

```
hugin.domain()
clone.domain(domain)
```

Arguments

domain name of gRaven domain

Details

Emulates functions of the same name in the RHugin package. A gRaven domain is an environment, with additional class attribute 'gRaven'. The environment holds structures such as nodes, states, parents and cptables, populated incrementally by functions in the package, preparing the information needed by the gRain package to create and manipulate the structure net in the domain, which is a grain object.

Value

character string naming a gRaven domain

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```

chest<-hugin.domain()

yn <- c("yes","no")
add.node(chest,"asia",states=yn)
add.node(chest,"smoke",states=yn)
add.node(chest,"tub",states=yn)
add.node(chest,"lung",states=yn)
add.node(chest,"either",states=yn)

add.edge(chest,"tub","asia")
add.edge(chest,"lung","smoke")
add.edge(chest,"either",c("lung","tub"))

set.table(chest,"asia",c(0.01,0.99))
set.table(chest,"tub",c(0.05,0.95,0.01,0.99))
set.table(chest,"smoke",c(0.5,0.5))
set.table(chest,"lung",c(0.1,0.9,0.01,0.99))
set.table(chest,"either",c(1,0,1,0,1,0,0,1))

chest

chest2<-clone.domain(chest)

add.node(chest2,"bronc",states=yn)
add.node(chest2,"dysp",states=yn)
add.node(chest2,"xray",states=yn)

add.edge(chest2,"bronc","smoke")
add.edge(chest2,"dysp",c("bronc","either"))
add.edge(chest2,"xray","either")

set.table(chest2,"bronc",c(0.6,0.4,0.3,0.7))
set.table(chest2,"dysp",c(0.9,0.1,0.7,0.3,0.8,0.2,0.1,0.9))
set.table(chest2,"xray",c(0.98,0.02,0.05,0.95))

chest2

```

initialize.domain *Re-initialise a gRaven domain*

Description

Re-initialise a gRaven domain

Usage

```
initialize.domain(domain)
```

Arguments

domain name of gRaven domain

Details

Emulates function of the same name in the RHugin package. Restores the domain to the state it was in after the call to compile, in particular erasing all entered evidence.

Value

a NULL value is invisibly returned.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
compile(chest)
chest
set.finding(chest, "asia", "yes")
set.finding(chest, "dysp", "no")
propagate(chest)
sapply(get.nodes(chest), function(x) get.belief(chest, x))
get.normalization.constant(chest)

initialize.domain(chest)
sapply(get.nodes(chest), function(x) get.belief(chest, x))
get.normalization.constant(chest)
```

list.domains

List gRaven domain Objects

Description

List gRaven domain Objects

Usage

```
list.domains(print=TRUE)
```

Arguments

print logical, should list of domains be output to console

Details

gRaven domains are R environment objects, with named slots typically including nodes, parents, cptables and net, where the net component is a gRain grain object. The domains that this function finds and lists are all the objects in .GlobalEnv of class "gRaven", and all those objects in this class that are components of a component named 'domains' of a list object in .GlobalEnv.

Value

The function invisibly returns a character vector of the domain names found.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo(chest, package="gRaven")
chest
chest2<-clone.domain(chest)
chest2

set.finding(chest, "asia", "yes")
set.finding(chest, "dysp", "no")
propagate(chest)
get.belief(chest, "asia")
get.belief(chest, "tub")

propagate(chest2)
get.belief(chest2, "asia")
get.belief(chest2, "tub")

list.domains()
```

map.configurations *Get belief in a gRaven domain*

Description

Find the configurations of the specified nodes that occur with probability pmin or greater. These configurations are known as most probable configurations or maximum a posteriori (MAP) configurations.

Usage

```
map.configurations(domain, nodes, pmin)
```

Arguments

domain	name of gRaven domain
nodes	character vector of names of nodes
pmin	a single numeric value between 0 and 1 specifying the minimum probability for the most probable configurations

Details

Emulates function of the same name in the RHugin package by calls to gRain functions

Value

Data.frame with one column for each node in nodes. Each row contains a most probable configuration. The final column of the data.frame (Prob) gives the probability of the configuration.

Author(s)

Therese Graversen, <theg@itu.dk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
set.finding(chest, "asia", "yes")
set.finding(chest, "dysp", "no")
map.configurations(chest, c("lung", "bronc"), 0.0001)
```

print.gRaven	<i>Print a gRaven domain</i>
--------------	------------------------------

Description

Print method for a gRaven domain

Usage

```
## S3 method for class 'gRaven'
print(x, ...)
```

Arguments

x	gRaven domain
...	additional arguments to <code>print</code>

Details

Prints summary information describing the domain.

Value

a NULL value is invisibly returned.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo(chest,package="gRaven")
print(chest)
```

<code>propagate.gRaven</code>	<i>Propagate method for a gRaven domain</i>
-------------------------------	---

Description

Propagate method for a gRaven domain

Usage

```
## S3 method for class 'gRaven'
propagate(object, ...)
```

Arguments

<code>object</code>	character string, name of gRaven domain
<code>...</code>	additional arguments to propagate

Details

Propagates all previously entered evidence through the network. This call is required by functions such as `get.belief` and `map.configurations`, but not before `get.normalization.constant`. In the implementation, findings will have been accumulated in the variable `net$cache` in the domain, by calls to `set.finding`, and are moved to `net$evid` by the `propagate` function.

Value

a NULL value is invisibly returned.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
compile(chest)
chest
set.finding(chest, "asia", "yes")
set.finding(chest, "dysp", "no")
propagate(chest)
get.belief(chest, "asia")
get.belief(chest, "tub")
```

set.finding	<i>Set, retract and inspect findings</i>
-------------	--

Description

Set, retract and inspect findings

Usage

```
set.finding(domain, node, finding)
retract(domain, nodes=domain$nodes)
get.finding(domain, nodes=domain$nodes, type = c("entered", "propagated"),
namestates=FALSE)
```

Arguments

domain	name of gRaven domain
node	name of node
nodes	character vector naming one or more nodes
finding	either a state of node node or a probability distribution over the states, or other non-negative vector of that length.
type	character vector of one or more options to select between evidence entered before or after most recent propagation
namestates	logical, should output have the states labelled

Details

For type, partial matching is used.

Value

For set.finding and retract, a NULL value is invisibly returned. For get.finding, the likelihood vector is invisibly returned for each specified node, in a format governed by namestates. This is a named vector in the case of a single node, otherwise a list of such vectors. For a single node, in the absence of any evidence, a named vector of 1's is returned.

Differences from RHugin

Findings (or evidence) are handled differently in gRaven (and gRain) than in RHugin, and that is reflected in differences in results from these functions, especially `get.finding`. For `set.finding` in gRaven, the case argument is not supported. For `get.finding` in both packages, previously-set evidence is displayed as a non-negative vector indexed by the states, typically a probability distribution; two enhancements in gRaven are that findings on more than one node can be displayed, and that the format of the output can be controlled by `namestates`. In gRaven when new evidence is set by `set.finding`, it replaces any existing evidence on the same node, and all existing evidence on all nodes is "unpropagated". Evidence is held in a data structure cache until propagation, when it is moved to `evid`; these names are used in labelling the output from `get.finding`. Finally if no evidence has been entered on a node, RHugin reports a vector of all ones, while gRaven returns such a vector invisibly.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo(chest, package="gRaven", echo=FALSE)
chest
set.finding(chest, "asia", "yes")
set.finding(chest, "dysp", "no")
propagate(chest)

get.belief(chest, "asia")
get.belief(chest, "tub")
get.finding(chest)

retract(chest, "dysp")
get.belief(chest, "tub")
get.finding(chest)
```

set.table

Set or get CPT in a gRaven domain

Description

Set or get CPT in a gRaven domain

Usage

```
set.table(domain, n, tab = 1, type = c("cpt", "experience",
  "fading"))
get.table(domain, n, type = c("cpt", "experience",
  "fading"), class = c("data.frame", "table",
  "ftable", "numeric"))
```

Arguments

domain	name of gRaven domain
n	name of node
tab	values of conditional probabilities
type	a character string specifying the type of table to set.
class	a character string specifying the class of the returned table

Value

For `set.table`, a NULL value is invisibly returned; for `get.table`, an object of required class.

Differences from RHugin

Only `type="cpt"` and `class="data.frame"` are currently implemented in gRaven.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
chest<-hugin.domain()
add.node(chest,"asia",states=c("yes","no"))
add.node(chest,"tub",states=c("yes","no"))
add.edge(chest,"tub","asia")
compile(chest)
chest
get.table(chest,"asia")
set.table(chest,"asia",c(0.01,0.99))
get.table(chest,"asia")
```

simulate.gRaven	<i>method for a gRaven domain</i>
-----------------	-----------------------------------

Description

Simulate method for a gRaven domain

Usage

```
## S3 method for class 'gRaven'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	character string, name of gRaven domain
nsim	Number of cases to simulate
seed	An optional integer controlling the random number generation
...	additional arguments to simulate

Value

a data frame.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
compile(chest)
chest
set.finding(chest, "asia", "yes")
set.finding(chest, "dysp", "no")
newdata<-simulate(chest, 100)
head(newdata)
```

summary.gRaven

Summary method for a gRaven domain

Description

Summary method for a gRaven domain

Usage

```
## S3 method for class 'gRaven'
summary(object, domain = TRUE, nodes = FALSE, jt = FALSE, print.cliques = FALSE,
        ...)
```

Arguments

object	character string, name of gRaven domain
domain	Logical, should domain be summarised
jt	Logical, should junction tree be summarised
nodes	Logical, should nodes be summarised
print.cliques	Logical, should cliques be printed
...	additional arguments to summary

Value

a data frame.

Differences from RHugin

In gRaven, not all of the options are yet implemented, and generally the information delivered is less detailed than in the RHugin version.

Author(s)

Peter J. Green, <P.J.Green@bristol.ac.uk>

Examples

```
demo("chest", package="gRaven", echo=FALSE)
chest
summary(chest, jt=TRUE)
```

Index

add.edge, 3
add.node, 4

check.compiled (compile.gRaven), 6
clone.domain (hugin.domain), 12
compile, 6
compile.gRaven, 6
compress, 7

delete.edge (add.edge), 3
delete.node (add.node), 4

get.belief, 7
get.children (get.states), 11
get.edges (get.states), 11
get.finding (set.finding), 18
get.marginal (get.belief), 7
get.nodes, 8
get.normalization.constant, 9
get.parents, 10
get.states, 11
get.table (set.table), 19
gRaven (gRaven-package), 2
gRaven-package, 2

hugin.domain, 12

initialize.domain, 13

list.domains, 14

map.configurations, 15

print, 16
print.gRaven, 16
propagate, 17
propagate.gRaven, 17

retract (set.finding), 18

set.finding, 18

set.table, 19
simulate, 21
simulate.gRaven, 20
summary, 21
summary.gRaven, 21