R documentation

of all in '.'

August 17, 2011

R topics documented:

`RChurch-package` *R interface to Church*

Description

Provides an interface to the Church set of Scheme commands. Allows models to be specified as R scripts and translates the scripts to Church behind the cenes.

Details

Package:	RChurch
Type:	Package
Version:	1.0
Date:	2011-08-11
License:	GPL 3
LazyLoad:	yes

See `church.model` and `church.sample`

Author(s)

Jon Malmaud <malmaud@mit.edu>

References

~~ Literature or other references for background information ~~

See Also

~~ Optional links to other man pages, e.g. ~~ ~~ <pkg> ~~

`church.model` *Creates a Church program object*

Description

Creates an object which encapsules a single Church program.
A church program consists of a model and a predicate.

Usage

```
church.model(model = function() {
}, predicate = function() {
}, context = function() {
})
```

Arguments

`model`	An R function containing the model.
`predicate`	An R function containing the preciate. Must return a logical value.
`context`	An optional R function providing definitions of objects that can be referenced in the model.

Value

A Church object. It is an S3 object.

Author(s)

Jon Malmaud

See Also

`church.samples` for drawing samples from the model.

Examples

```
model = function() {
  x = rnorm(1,0,1)
  y = rnorm(1,0,1)
  z = x+y
}

my.program = church.model(model, predicate=function() {z>0})
```

church.samples *Draw samples from a Church program*

Usage

```
church.samples(church, variable.names = church$vars, n.iter = 100, thin = 100, m
```

Arguments

church	The Church program, created by church.model
variable.names	A list of variable names defined in the model whose sampled values will be returned
n.iter	The number of iterations to run the sampler for.
thin	The thinning interval. An integer giving the frequency of samples to keep.
method	The sampler method. Either "mcmc" for the Metropolis Hastings sampler or "rejection" for rejection sampling.
inputs	

Note

Note that the total number of iterations the sampler will be run for is n.iter * thin.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function (church, variable.names=church$vars,  n.iter=100,
                            thin=100, method='mcmc', inputs=list()) {
  vars = variable.names
  n.samples= n.iter
  vars.for.church = gsub('\.', vars, '-')
  list_line = sprintf('(list %s)', paste(vars.for.church,collapse=' '))
  church$obs.vars= list_line
  #tmp_file = '/Users/malmaud/tmp/tmp_church.church'
  #church_path = '.:/Users/malmaud/tmp/scheme-tools:/Users/malmaud/tmp/bher'
  #bher_path = '/Users/malmaud/tmp/bher'
  tmp_file = file.path(tempdir(), 'tmp_church.church')
  church_path = paste('.', system.file('scheme-tools', package='RChurch'), system.file('b
  bher_path = system.file('bher', package='RChurch')
  church$inputs = R.to.church.inits(inputs)
  if(method=='mcmc') {
    church$query.line.prefix = paste('(mh-query', n.samples, thin, sep=' ')
    church$query.line.suffix = ')'
  }
  else if(method=='rejection'){
    church$query.line.prefix = sprintf('(repeat %d (lambda () (rejection-query ', n.iter)
    church$query.line.suffix='')))'
  }
```

```
  #file.remove(tmp_file)
  writeLines(church.program(church), tmp_file)
  old_warn = getOption('warn')
  options(warn=-1)
  env_str = c()
  env_str[1] = paste("PATH=", bher_path,':$PATH', sep='')
  env_str[2] = paste("VICARE_LIBRARY_PATH=", church_path,":$VICARE_LIBRARY_PATH", sep='')
  raw_output = system2('bher', tmp_file, env=paste(env_str,collapse="\n"), stdout=T)
  options(warn=old_warn)
  data_start = which(regexpr('^\(', raw_output)==1)[[1]]
  data_str = raw_output[data_start:length(raw_output)]
  data_str = paste(data_str, collapse='')
  data_str = gsub('\(', '', data_str)
  data_str = gsub('\)', '', data_str)
  data_str = gsub('#f', '0', data_str)
  data_str = gsub('#t', '1', data_str)
  data = strsplit(data_str,' ')[[1]]
  data = as.numeric(data)
  n_data = length(data)
  res = list()
  for(i in 1:length(vars)) {
    res[[i]] = data[seq(i, n_data, length(vars))]
#     if(var_types[i]=='logical') {
#       res[[i]] = as.logical(res[[i]])
#     }
  }
  names(res) = vars
  res.mcmc = mcmc(do.call(cbind, res))

  plot(res.mcmc)
  res.mcmc
  }
```

Index