Package 'R2OpenBUGS'

February 23, 2026

Title Running OpenBUGS from R

Date 2026-02-23

Version 3.2-5

Description Using this package,
it is possible to call a BUGS model, summarize inferences and
convergence in a table and graph, and save the simulations in arrays for easy access
in R.

Depends R (>= 2.13.0)

Imports coda (>= 0.11-0), boot

SystemRequirements OpenBUGS (>= 3.2.2)

License GPL-2

NeedsCompilation no

Author Andrew Gelman [aut],
Sibylle Sturtz [aut],
Uwe Ligges [cre, aut],
Gregor Gorjanc [ctb],
Jouni Kerman [ctb],
Neal Thomas [aut] (adapted from R2WinBUGS)

Maintainer Uwe Ligges <ligges@statistik.tu-dortmund.de>

Repository CRAN

Date/Publication 2026-02-23 17:00:29 UTC

Contents

R2OpenBUGS-package   *Running OpenBUGS from R*

Description

R2OpenBUGS Call a **BUGS** model, summarize inferences and convergence in a table and graph, and save the simulations in arrays for easy access in R. The main command is [bugs](#).

Details

The following are sources of information on **R2OpenBUGS** package:

DESCRIPTION file	`library(help="R2OpenBUGS")`
This file	`package?R2OpenBUGS`
Vignette	`vignette("R2OpenBUGS")`
Some help files	bugs write.model print.bugs plot.bugs
News	`file.show(system.file("NEWS", package="R2OpenBUGS"))`

---

as.bugs.array   *Convert to bugs object*

Description

Function converting results from Markov chain simulations, that might not be from BUGS, to bugs object. Used mainly to display results with [plot.bugs](#).

Usage

```
as.bugs.array(sims.array, model.file=NULL, program=NULL,
    DIC=FALSE, DICOutput=NULL, n.iter=NULL, n.burnin=0, n.thin=1)
```

Arguments

`sims.array`	3-way array of simulation output, with dimensions n.keep, n.chains, and length of combined parameter vector.
`model.file`	file containing the model written in **OpenBUGS** code
`program`	the program used
`DIC`	logical; whether DIC should be calculated, see also argument `DICOutput` and details
`DICOutput`	DIC value
`n.iter`	number of total iterations per chain used for generating `sims.array`
`n.burnin`	length of burn in, i.e. number of iterations to discarded at the beginning for generating `sims.array`
`n.thin`	thinning rate, a positive integer, used for generating `sims.array`

Details

This function takes a 3-way array of simulations and makes it into a [bugs](#) object that can be conveniently displayed using `print` and `plot` and accessed using `attach.bugs`. If the third dimension of sims() has names, the resulting bugs object will respect that naming convention. For example, if the parameter names are "alpha[1]", "alpha[2]", ..., "alpha[8]", "mu", "tau", then `as.bugs.array` will know that alpha is a vector of length 8, and mu and tau are scalar parameters. These will all be plotted appropriately by `plot` and attached appropriately by `attach.bugs`.

If `DIC=TRUE` then DIC can be either already passed to argument `DICOutput` or calculated from deviance values in `sims.array`.

Value

A [bugs](#) object is returned

Author(s)

Jouni Kerman, <kerman@stat.columbia.edu> with modification by Andrew Gelman, <gelman@stat.columbia.edu>, packaged by Uwe Ligges, <ligges@statistik.tu-dortmund.de>.

See Also

[bugs](#)

---

attach.all   *Attach / detach elements of (bugs) objects to search path*

Description

The database is attached/detached to the search path. See [attach](#) for details.

Usage

```
attach.all(x, overwrite = NA, name = "attach.all")
attach.bugs(x, overwrite = NA)
detach.all(name = "attach.all")
detach.bugs()
```

Arguments

x   An object, which must be of class bugs for attach.bugs.

overwrite   If TRUE, objects with identical names in the Workspace (.GlobalEnv) that are
             masking objects in the database to be attached will be deleted. If NA (the default)
             and an interactive session is running, a dialog box asks the user whether masking
             objects should be deleted. In non-interactive mode, behaviour is identical to
             overwrite=FALSE, i.e. nothing will be deleted.

name   The name of the environment where x will be attached / which will be detached.

Details

While attach.all attaches all elements of an object x to a database called name, attach.bugs
attaches all elements of x$sims.list to the database bugs.sims itself making use of attach.all.

detach.all and detach.bugs are removing the databases mentioned above.
attach.all also attaches n.sims (the number of simulations saved from the MCMC runs) to the
database.

Each scalar parameter in the model is attached as vectors of length n.sims, each vector is attached
as a 2-way array (with first dimension equal to n.sims), each matrix is attached as a 3-way array,
and so forth.

Value

attach.all and attach.bugs invisibly return the [environment](#)(s).

detach.all and detach.bugs detach the environment(s) named name created by attach.all.

Note

Without detaching, do not use `attach.all` or `attach.bugs` on another (bugs) object, because instead of the given name, an object called `name` is attached. Therefore strange things may happen …

See Also

bugs, attach, detach

Examples

```
# An example model file is given in:
model.file <- system.file("model", "schools.txt", package="R2OpenBUGS")
# Some example data (see ?schools for details):
data(schools)
J <- nrow(schools)
y <- schools$estimate
sigma.y <- schools$sd
data <- list ("J", "y", "sigma.y")
inits <- function(){
    list(theta = rnorm(J, 0, 100), mu.theta = rnorm(1, 0, 100),
        sigma.theta = runif(1, 0, 100))
}
parameters <- c("theta", "mu.theta", "sigma.theta")
## Not run:
## See ?bugs if the following fails:
schools.sim <- bugs(data, inits, parameters, model.file,
    n.chains = 3, n.iter = 1000,
    working.directory = NULL)

# Do some inferential summaries
attach.bugs(schools.sim)
# posterior probability that the coaching program in school A
# is better than in school C:
print(mean(theta[,1] > theta[,3]))
# 50
# and school C's program:
print(quantile(theta[,1] - theta[,3], c(.25, .75)))
plot(theta[,1], theta[,3])
detach.bugs()

## End(Not run)
```

| bugs | *Run OpenBUGS from R* |

Description

The bugs function takes data and starting values as input. It automatically writes a **OpenBUGS** script, calls the model, and saves the simulations for easy access in R.

Usage

```
bugs(data, inits, parameters.to.save, n.iter, model.file="model.txt",
    n.chains=3, n.burnin=floor(n.iter / 2), n.thin=1,
    saveExec=FALSE,restart=FALSE,
    debug=FALSE, DIC=TRUE, digits=5, codaPkg=FALSE,
    OpenBUGS.pgm=NULL,
    working.directory=NULL,
    clearWD=FALSE, useWINE=FALSE, WINE=NULL,
    newWINE=TRUE, WINEPATH=NULL, bugs.seed=1, summary.only=FALSE,
    save.history=(.Platform$OS.type == "windows" | useWINE==TRUE),
    over.relax = FALSE)
```

Arguments

data
: either a named list (names corresponding to variable names in the model.file) of the data for the **OpenBUGS** model, *or* a vector or list of the names of the data objects used by the model. If data is a one element character vector (such as "data.txt"), it is assumed that data have already been written to the working directory into that file, e.g. by the function bugs.data.

inits
: a list with n.chains elements; each element of the list is itself a list of starting values for the **OpenBUGS** model, *or* a function creating (possibly random) initial values. Alternatively, if inits=NULL, initial values are generated by **OpenBUGS**. If inits is a character vector with n.chains elements, it is assumed that inits have already been written to the working directory into those files, e.g. by the function bugs.inits.

parameters.to.save
: character vector of the names of the parameters to save which should be monitored

model.file
: File containing the model written in **OpenBUGS** code. The extension must be '.txt'. The default location is given by working.directory. The old convention allowing model.file to be named '.bug' has been eliminated because the new **OpenBUGS** feature that allows the program image to be saved and later restarted uses the .bug extension for the saved images. Alternatively, model.file can be an R function that contains a BUGS model that is written to a temporary model file (see tempfile) using write.model.

n.chains
: number of Markov chains (default: 3)

n.iter
: number of total iterations per chain (including burn in; default: 2000)

n.burnin
: length of burn in, i.e. number of iterations to discard at the beginning. Default is n.iter/2, that is, discarding the first half of the simulations.

n.thin
: Thinning rate. Must be a positive integer. The default is n.thin = 1. The thinning is implemented in the OpenBUGS update phase, so thinned samples are never stored, and they are not counted in n.burnin or n.iter. Setting n.thin=2, doubles the number of iterations OpenBUGS performs, but does not change n.iter or n.burnin. Thinning implemented in this manner is not captured in summaries created by packages such as **coda**.

saveExec   If TRUE, a re-startable image of the OpenBUGS execution is saved with basename
                (model.file) and extension .bug in the working directory, which must be spec-
                ified. The .bug files can be large, so users should monitor them carefully and
                remove them when not needed.

restart   If TRUE, execution resumes with the final status from the previous execution
                stored in the .bug file in the working directory. If n.burnin=0,additional iter-
                ations are performed and all iterations since the previous burnin are used (in-
                cluding those from past executions). If n.burnin>0, a new burnin is performed,
                and the previous iterations are discarded, but execution continues from the sta-
                tus at the end of the previous execution. When restart=TRUE, only n.burnin,
                n.iter, and saveExec inputs should be changed from the call creating the .bug
                file, otherwise failed or erratic results may be produced. Note the default has
                n.burnin>0.

debug   if FALSE (default), **OpenBUGS** is closed automatically when the script has fin-
                ished running, otherwise **OpenBUGS** remains open for further investigation.
                The debug option is not available for linux execution.

DIC   logical; if TRUE (default), compute deviance, pD, and DIC. The results are ex-
                tracted directly from the **OpenBUGS** log, which uses the rule pD = Dbar - Dhat.
                If extraction fails or if there are less iterations than required for the adaptive
                phase, the rule pD=var(deviance) / 2 is computed in R. See [bugs.log](#) for
                more information on extracting results from the log file.

digits   number of significant digits used for **OpenBUGS** input, see [formatC](#)

codaPkg   logical; if FALSE (default) a bugs object is returned, if TRUE file names of **Open-
                BUGS** output are returned for easy access by the **coda** package through function
                [read.bugs](#). A bugs object can be converted to an mcmc.list object as used by
                the **coda** package with the method [as.mcmc.list](#) (for which a method is pro-
                vided by R2OpenBUGS).

OpenBUGS.pgm   For Windows or WINE execution, the full path to the OpenBUGS executable.
                For linux execution, the full path to the OpenBUGS executable or shell script
                (the path to the shell script is not required if the OpenBUGS shell script is
                in the user's PATH variable). If NULL (unset) and the environment variable
                OpenBUGS_PATH is set the latter will be used as the default. If NULL (unset), the
                environment variable OpenBUGS_PATH is unset and the global option R2OpenBUGS.pgm
                is not NULL the latter will be used as the default. If none of the former are set and
                OS is Windows, the most recent OpenBUGS version registered in the Windows
                registry will be used as the default. For other operating systems, the location is
                guessed by Sys.which("OpenBUGS").

working.directory
                sets working directory during execution of this function; **OpenBUGS**' input and
                output will be stored in this directory; if NULL, a temporary working directory
                via [tempdir](#) is used.

clearWD   logical; indicating whether the files 'data.txt', 'inits[1:n.chains].txt',
                'log.odc', 'codaIndex.txt', and 'coda[1:nchains].txt' should be removed
                after **OpenBUGS** has finished. If set to TRUE, this argument is only respected if
                codaPkg=FALSE.

useWINE	logical; attempt to use the Wine emulator to run **OpenBUGS**. Default is FALSE. If WINE is used, the arguments OpenBUGS.pgm and working.directory must be given in form of Linux paths rather than Windows paths (if not NULL).
WINE	Character, path to 'wine' binary file, it is tried hard (by a guess and the utilities which and locate) to get the information automatically if not given.
newWINE	Use new versions of Wine that have 'winepath' utility
WINEPATH	Character, path to 'winepath' binary file, it is tried hard (by a guess and the utilities which and locate) to get the information automatically if not given.
bugs.seed	Random seed for **OpenBUGS**. Must be an integer between 1-14. Seed specification changed between WinBUGS and OpenBUGS; see the OpenBUGS documentation for details.
summary.only	If TRUE, only a parameter summary for very quick analyses is given, temporary created files are not removed in that case.
save.history	If TRUE (the default), trace plots are generated at the end.
over.relax	If TRUE, over-relaxed form of MCMC is used if available from OpenBUGS.

Details

To run:

1. Write a **BUGS** model in an ASCII file (hint: use write.model).

2. Go into R.

3. Prepare the inputs for the bugs function and run it (see Example section).

4. An **OpenBUGS** window will pop up and R will freeze up. The model will now run in **OpenBUGS**. It might take awhile. You will see things happening in the Log window within **OpenBUGS**. When **OpenBUGS** is done, its window will close and R will work again.

5. If an error message appears, re-run with debug=TRUE.

BUGS version support:

OpenBUGS   >=3.2.1

Operation system support:

MS Windows no problem

Linux, intel processors GUI display and graphics not available.

Mac OS X and Unix in general possible with Wine emulation via useWINE=TRUE

If useWINE=TRUE is used, all paths (such as working.directory and model.file, must be given in native (Unix) style, but OpenBUGS.pgm can be given in Windows path style (e.g. "c:/Program Files/OpenBUGS/") or native (Unix) style
(e.g. "/path/to/wine/folder/dosdevices/c:/Program Files/OpenBUGS/OpenBUGS321/OpenBUGS.exe").

Value

If codaPkg=TRUE the returned values are the names of coda output files written by **OpenBUGS** containing the Markov Chain Monte Carlo output in the CODA format. This is useful for direct access with read.bugs.

If codaPkg=FALSE, the following values are returned:

n.chains	see Section 'Arguments'
n.iter	see Section 'Arguments'
n.burnin	see Section 'Arguments'
n.thin	see Section 'Arguments'
n.keep	number of iterations kept per chain (equal to (n.iter-n.burnin) / n.thin)
n.sims	number of posterior simulations (equal to n.chains * n.keep)
sims.array	3-way array of simulation output, with dimensions n.keep, n.chains, and length of combined parameter vector
sims.list	list of simulated parameters: for each scalar parameter, a vector of length n.sims for each vector parameter, a 2-way array of simulations, for each matrix parameter, a 3-way array of simulations, etc. (for convenience, the n.keep*n.chains simulations in sims.matrix and sims.list (but NOT sims.array) have been randomly permuted)
sims.matrix	matrix of simulation output, with n.chains*n.keep rows and one column for each element of each saved parameter (for convenience, the n.keep*n.chains simulations in sims.matrix and sims.list (but NOT sims.array) have been randomly permuted)
summary	summary statistics and convergence information for each saved parameter.
mean	a list of the estimated parameter means
sd	a list of the estimated parameter standard deviations
median	a list of the estimated parameter medians
root.short	names of argument parameters.to.save and "deviance"
long.short	indexes; programming stuff
dimension.short	
	dimension of indexes.short
indexes.short	indexes of root.short
last.values	list of simulations from the most recent iteration; they can be used as starting points if you wish to run **OpenBUGS** for further iterations
pD	an estimate of the effective number of parameters, for calculations see the section "Arguments".
DIC	mean(deviance) + pD

Author(s)

Andrew Gelman, <gelman@stat.columbia.edu>; modifications and packaged by Sibylle Sturtz, <sturtz@statistik.tu-dortmund.de>, Uwe Ligges, and Neal Thomas

References

Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B. (2003): *Bayesian Data Analysis*, 2nd edition, CRC Press.

Sturtz, S., Ligges, U., Gelman, A. (2005): R2WinBUGS: A Package for Running WinBUGS from R. *Journal of Statistical Software* 12(3), 1-16.

See Also

[print.bugs](), [plot.bugs](), as well as **coda** and **BRugs** packages

Examples

```
# An example model file is given in:
model.file <- system.file(package="R2OpenBUGS", "model", "schools.txt")
# Let's take a look:
# file.show(model.file)

# Some example data (see ?schools for details):
data(schools)
schools

J <- nrow(schools)
y <- schools$estimate
sigma.y <- schools$sd
data <- list ("J", "y", "sigma.y")
inits <- function(){
    list(theta=rnorm(J, 0, 100), mu.theta=rnorm(1, 0, 100),
        sigma.theta=runif(1, 0, 100))
}
## or alternatively something like:
# inits <- list(
#   list(theta=rnorm(J, 0, 90), mu.theta=rnorm(1, 0, 90),
#        sigma.theta=runif(1, 0, 90)),
#   list(theta=rnorm(J, 0, 100), mu.theta=rnorm(1, 0, 100),
#        sigma.theta=runif(1, 0, 100))
#   list(theta=rnorm(J, 0, 110), mu.theta=rnorm(1, 0, 110),
#        sigma.theta=runif(1, 0, 110)))

parameters <- c("theta", "mu.theta", "sigma.theta")

## Not run:
## You may need to specify "OpenBUGS.pgm"
## also you need write access in the working directory:
schools.sim <- bugs(data, inits, parameters, model.file,
    n.chains=3, n.iter=5000)
print(schools.sim)
plot(schools.sim)

## End(Not run)
```

---

bugs.data *Writing input for OpenBUGS*

Description

Write file for **OpenBUGS** to read.

Usage

```
bugs.data(data, dir = getwd(), digits = 5, data.file = "data.txt")
```

Arguments

data	either a named list (names corresponding to variable names in the `model.file`) of the data for the **OpenBUGS** model, *or* a vector or list of the names of the data objects used by the model
dir	the directory to write the file 'data.txt' to
digits	number of significant digits used for **OpenBUGS** input, see [formatC](#)
data.file	name for the file R writes the data into.

Value

The name of `data.file` is returned and as a side effect, the data file is written

See Also

The main function to be called by the user is [bugs](#).

---

bugs.inits *Writing input for OpenBUGS*

Description

Write files 'inits1.txt', 'inits2.txt', etc., in the working directory for **OpenBUGS** to read

Usage

```
bugs.inits(inits, n.chains, digits,
    inits.files = paste("inits", 1:n.chains, ".txt", sep = ""))
```

Arguments

`inits`	a list with `n.chains` elements; each element of the list is itself a list of starting values for the **OpenBUGS** model, *or* a function creating (possibly random) initial values
`n.chains`	number of Markov chains
`digits`	number of significant digits used for **OpenBUGS** input, see formatC
`inits.files`	name for the inits files R write the inits into.

Value

Vector of names of `inits.files`; as a side effect, the inits files 'inits*.txt' are written

See Also

The main function to be called by the user is bugs.

`bugs.log`	*Read data from OpenBUGS logfile*

Description

Read data such as summary statistics and DIC information from the **OpenBUGS** logfile

Usage

```
bugs.log(file)
```

Arguments

`file`	Location of the **OpenBUGS** logfile

Details

Returns the OpenBUGS summary statistics and DIC extracted directly from the log file.

Value

A list with components:

`stats`	A matrix containing summary statistics for each saved parameter. Comparable to the information in the element `summary` of a bugs object as returned by bugs.
`DIC`	A matrix containing the DIC statistics as returned from **OpenBUGS**.

Author(s)

Jouni Kerman

See Also

The main function that generates the log file is [bugs](#).

---

plot.bugs	*Plotting a bugs object*

Description

Plotting a bugs object

Usage

```
## S3 method for class 'bugs'
plot(x, display.parallel = FALSE, ...)
```

Arguments

x   an object of class 'bugs', see [bugs](#) for details

display.parallel

                 display parallel intervals in both halves of the summary plots; this is a convergence-monitoring tool and is not necessary once you have approximate convergence (default is FALSE)

...   further arguments to [plot](#)

See Also

[bugs](#)

---

print.bugs	*Printing a bugs object*

Description

Printing a bugs object

Usage

```
## S3 method for class 'bugs'
print(x, digits.summary = 1, ...)
```

Arguments

x   an object of class 'bugs', see [bugs](#) for details

digits.summary   rounding for tabular output on the console (default is to round to 1 decimal place)

...   further arguments to [print](#)

See Also

bugs

---

read.bugs   *Read output files in CODA format*

Description

This function reads Markov Chain Monte Carlo output in the CODA format produced by **Open-BUGS** and returns an object of class mcmc.list for further output analysis using the **coda** package.

Usage

```
read.bugs(codafiles, ...)
```

Arguments

codafiles   character vector of filenames (e.g. returned from bugs in call such as bugs(.....,
                 codaPkg=TRUE, ...)). Each of the files contains coda output for one chain
                 produced by **OpenBUGS**, the *directory* name of the corresponding file 'CODAindex.txt'
                 is extracted from the first element of codafiles.

...   further arguments to be passed to read.coda

See Also

bugs, read.coda, mcmc.list

---

schools   *8 schools analysis*

Description

8 schools analysis

Usage

```
data(schools)
```

Format

A data frame with 8 observations on the following 3 variables.

school See Source.

estimate See Source.

sd See Source.

Source

Rubin, D.B. (1981): Estimation in Parallel Randomized Experiments. *Journal of Educational Statistics* 6(4), 377-400.

Section 5.5 of Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B. (2003): *Bayesian Data Analysis*, 2nd edition, CRC Press.

---

validateInstallOpenBUGS

Compare OpenBUGS/R2OpenBUGS execution to results supplied with OpenBUGS

Description

A selected subset of the examples from the OpenBUGS manual is executed and compared to results supplied with the package to validate installation.

Usage

```
validateInstallOpenBUGS(
    OpenBUGS.pgm=NULL,
    useWINE=FALSE, WINE=NULL,
    newWINE=TRUE, WINEPATH=NULL
    )
```

Arguments

OpenBUGS.pgm	See bugs.
useWINE	logical; attempt to use the Wine emulator to run **OpenBUGS**. Default is FALSE. If WINE is used, the arguments OpenBUGS.pgm and working.directory must be given in form of Linux paths rather than Windows paths (if not NULL).
WINE	Character, path to 'wine' binary file, it is tried hard (by a guess and the utilities which and locate) to get the information automatically if not given.
newWINE	Use new versions of Wine that have 'winepath' utility
WINEPATH	Character, path to 'winepath' binary file, it is tried hard (by a guess and the utilities which and locate) to get the information automatically if not given.

Details

Operation system support:

- **MS Windows** Yes
- **Linux, intel processors** Yes, but GUI display and graphics not available.
- **Mac OS X** and **Unix** Wine emulation via useWINE=TRUE

If useWINE=TRUE is used, all paths (such as working.directory and model.file, must be given in native (Unix) style, but OpenBUGS.pgm can be given in Windows path style (e.g. "c:/Program Files/OpenBUGS/") or native (Unix) style
(e.g. "/path/to/wine/folder/dosdevices/c:/Program Files/OpenBUGS/OpenBUGS321/OpenBUGS.exe").

Value

No data returned. Prints match/no match result to console for each example.

Author(s)

Neal Thomas based on BRugs examples created by Chris Jackaon.

References

Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B. (2003): *Bayesian Data Analysis*, 2nd edition, CRC Press.

Sturtz, S., Ligges, U., Gelman, A. (2005): R2WinBUGS: A Package for Running WinBUGS from R. *Journal of Statistical Software* 12(3), 1-16.

See Also

[bugs](#)

---

write.model   *Creating a OpenBUGS model file*

Description

Convert R function to a **OpenBUGS** model file

Usage

```
write.model(model, con = "model.bug", digits = 5)
```

Arguments

model	R function containing the BUGS model in the BUGS model language, for minor differences see Section Details.
con	passed to [writeLines](#) which actually writes the model file
digits	number of significant digits used for **OpenBUGS** input, see [formatC](#)

Details

BUGS models follow closely S syntax. It is therefore possible to write most BUGS models as R functions.

As a difference, BUGS syntax allows truncation specification like this: dnorm(...) I(...) but this is illegal in R. To overcome this incompatibility, use dummy operator %_% before I(...): dnorm(...) %_% I(...). The dummy operator %_% will be removed before the BUGS code is saved.

Value

Nothing, but as a side effect, the model file is written

Author(s)

original idea by Jouni Kerman, modified by Uwe Ligges

See Also

bugs

Examples

```
## Same "schoolsmodel" that is used in the examples in ?bugs:
schoolsmodel <- function(){
    for (j in 1:J){
        y[j] ~ dnorm (theta[j], tau.y[j])
        theta[j] ~ dnorm (mu.theta, tau.theta)
        tau.y[j] <- pow(sigma.y[j], -2)
    }
    mu.theta ~ dnorm (0.0, 1.0E-6)
    tau.theta <- pow(sigma.theta, -2)
    sigma.theta ~ dunif (0, 1000)
}

    ## some temporary filename:
    filename <- file.path(tempdir(), "schoolsmodel.bug")

## write model file:
write.model(schoolsmodel, filename)
## and let's take a look:
file.show(filename)
```

Index