Package 'mcgibbsit'

September 25, 2023

Title Warnes and Raftery's 'MCGibbsit' MCMC Run Length and Convergence
Diagnostic

Version 1.2.2

Date 2023-09-23

Author Gregory R. Warnes `<greg@warnes.net>`, Robert Burrows

Depends coda

Description Implementation of Warnes & Raftery's MCGibbsit run-length and
convergence diagnostic for a set of (not-necessarily independent)
Markov Chain Monte Carlo (MCMC) samplers. It combines the quantile
estimate error-bounding approach of the Raftery and Lewis MCMC run
length diagnostic `gibbsit` with the between verses within chain
approach of the Gelman and Rubin MCMC convergence diagnostic.

Maintainer Gregory R. Warnes `<greg@warnes.net>`

License GPL

NeedsCompilation no

RoxygenNote 7.2.3

Encoding UTF-8

URL https://github.com/r-gregmisc/mcgibbsit

BugReports https://github.com/r-gregmisc/mcgibbsit/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository CRAN

Date/Publication 2023-09-25 10:40:02 UTC

R topics documented:

---

mcgibbsit   *Warnes and Raftery's MCGibbsit MCMC diagnostic*

Description

`mcgibbsit` provides an implementation of Warnes & Raftery's MCGibbsit run-length diagnostic for a set of (not-necessarily independent) MCMC samplers. It combines the estimate error-bounding approach of Raftery and Lewis with the between chain variance verses within chain variance approach of Gelman and Rubin.

Usage

```
mcgibbsit(
  data,
  q = 0.025,
  r = 0.0125,
  s = 0.95,
  converge.eps = 0.001,
  correct.cor = TRUE
)

## S3 method for class 'mcgibbsit'
print(x, digits = 3, ...)
```

Arguments

data	an 'mcmc' object.
q	quantile(s) to be estimated.
r	the desired margin of error of the estimate.
s	the probability of obtaining an estimate in the interval
converge.eps	Precision required for estimate of time to convergence.
correct.cor	should the between-chain correlation correction (R) be computed and applied. Set to false for independent MCMC chains.
x	an object used to select a method.
digits	minimal number of *significant* digits, see print.default.
...	further arguments passed to or from other methods.

Details

`mcgibbsit` computes the minimum run length N_{min}, required burn in M, total run length N, run length inflation due to *auto-correlation*, I, and the run length inflation due to *between-chain* correlation, R for a set of exchangeable MCMC simulations which need not be independent.

The normal usage is to perform an initial MCMC run of some pre-determined length (e.g., 300 iterations) for each of a set of k (e.g., $k = 20$) MCMC samplers. The output from these samplers is

then read in to create an `mcmc.list` object and `mcgibbsit` is run specifying the desired accuracy of estimation for quantiles of interest. This will return the minimum number of iterations to achieve the specified error bound. The set of MCMC samplers is now run so that the total number of iterations exceeds this minimum, and `mcgibbsit` is again called. This should continue until the number of iterations already complete is less than the minimum number computed by `mcgibbsit`.

If the initial number of iterations in `data` is too small to perform the calculations, an error message is printed indicating the minimum pilot run length.

The parameters `q`, `r`, `s`, `converge.eps`, and `correct.cor` can be supplied as vectors. This will cause `mcgibbsit` to produce a list of results, with one element produced for each set of values. I.e., setting `q=(0.025,0.975)`, `r=(0.0125,0.005)` will yield a list containing two `mcgibbsit` objects, one computed with parameters `q=0.025`, `r=0.0125`, and the other with `q=0.975,r=0.005`.

Value

An `mcgibbsit` object with components

`call`	parameters used to call 'mcgibbsit'
`params`	values of r, s, and q used
`resmatrix`	a matrix with 6 columns:

> **Nmin** The minimum required sample size for a chain with no correlation between consecutive samples. Positive autocorrelation will increase the required sample size above this minimum value.
>
> **M** The number of `burn in' iterations to be discarded (total over all chains).} \item{N}{T` pendence factor') of the extent to which auto-correlation inflates the required sample size. Values of `I'` larger than 5 indicate strong autocorrelation which may b iness' of the MCMC algorithm.
>
> **R** An estimate of the extent to which between-chain correlation inflates the required sample size. Large values of 'R' indicate that there is significant correlation between the chains and may be indicative of a lack of convergence or a poor multi-chain algorithm.

`nchains`	the number of MCMC chains in the data
`len`	the length of each chain

Author(s)

Gregory R. Warnes <greg@warnes.net> based on the the R function `raftery.diag` which is part of the 'CODA' library. `raftery.diag`, in turn, is based on the FORTRAN program 'gibbsit' written by Steven Lewis which is available from the Statlib archive.

References

Warnes, G.W. (2004). The Normal Kernel Coupler: An adaptive MCMC method for efficiently sampling from multi-modal distributions, https://stat.uw.edu/sites/default/files/files/reports/2001/tr395.pdf

Warnes, G.W. (2000). Multi-Chain and Parallel Algorithms for Markov Chain Monte Carlo. Dissertation, Department of Biostatistics, University of Washington, https://digital.lib.washington.edu/researchworks/handle/1773/9541

Raftery, A.E. and Lewis, S.M. (1992). One long run with diagnostics: Implementation strategies for Markov chain Monte Carlo. Statistical Science, 7, 493-497.

Raftery, A.E. and Lewis, S.M. (1995). The number of iterations, convergence diagnostics and generic Metropolis algorithms. In Practical Markov Chain Monte Carlo (W.R. Gilks, D.J. Spiegelhalter and S. Richardson, eds.). London, U.K.: Chapman and Hall.

See Also

read.mcmc

Examples

```
###
# Create example data files for 20 independent chains
# with serial correlation of 0.25
###

set.seed(42)
tmpdir <- tempdir()

nsamples <- 1000

for(i in 1:20){
  x <- matrix(nrow = nsamples+1, ncol=4)
  colnames(x) <- c("alpha","beta","gamma", "nu")

  x[,"alpha"] <- rnorm (nsamples+1, mean=0.025, sd=0.0025)^2
  x[,"beta"]  <- rnorm (nsamples+1, mean=53,    sd=12)
  x[,"gamma"] <- rbinom(nsamples+1, 20,         p=0.25) + 1
  x[,"nu"]    <- rnorm (nsamples+1, mean=x[,"alpha"] * x[,"beta"], sd=1/x[,"gamma"])

  # induce serial correlation of 0.25
  x <- 0.75 * x[2:(nsamples+1),] + 0.25 * x[1:nsamples,]


  write.table(
    x,
    file = file.path(
      tmpdir,
      paste("mcmc", i, "csv", sep=".")
      ),
    sep = ",",
    row.names = FALSE
  )
}

# Read them back in as an mcmc.list object
data <- read.mcmc(
  20,
  file.path(tmpdir, "mcmc.#.csv"),
  sep=",",
```

```
  col.names=c("alpha","beta","gamma", "nu")
  )

# Summary statistics
summary(data)

# Trace and Density Plots
plot(data)

# And check the necessary run length
mcgibbsit(data)
```

---

read.mcmc   *Read in data from a set of MCMC runs*

Description

Read in data from a set of MCMC runs and create an `mcmc.list` object.

Usage

```
read.mcmc(
  nc,
  sourcepattern,
  ...,
  col.names,
  start = 1,
  end = nrow(tmp)/numComponents * thin,
  thin = 1,
  numComponents = 1
)
```

Arguments

nc	Number of MCMC sampler files to read
sourcepattern	MCMC data file name pattern.
...	Arguments to be passed to `read.table` when loading MCMC sampler data.
col.names	Data file column names (optional)
start, end, thin	
	See documentation for `mcmc`
numComponents	Number of component samplers.

Details

This function reads in the states output from one or more MCMC samplers and creates a single mcmc.list object. sourcepattern will be used as a filename pattern with # replaced by the sampler number. EG, sourcepattern="MCMC.#.csv" will be converted to "MCMC.1.csv", "MCMC.2.csv", etc.

The function read.table is used to read in the data. Options for read.table may be included as part of the call to read.mcmc.

The start, end, and thin arguments can be used to annotate the MCMC samplers with additional information.

Value

An mcmc.list object containing nc component mcmc objects.

Author(s)

Gregory R. Warnes <greg@warnes.net>

See Also

mcmc, mcmc.list, read.table

Examples

```
###
# Create example data files for 20 independent chains
# with serial correlation of 0.
###

set.seed(42)
tmpdir <- tempdir()

for(i in 1:20){
  x <- matrix(rnorm(1000), ncol=4)

  x[,4] <- x[,4] + 1/3 * (x[,1] + x[,2] + x[,3])

  colnames(x) <- c("alpha","beta","gamma", "nu")

  write.table(
    x,
    file = file.path(
      tmpdir,
      paste("mcmc", i, "csv", sep=".")
      ),
    sep = ",",
    row.names=FALSE
  )
}
```

```
# Read them back in as an mcmc.list object
data <- read.mcmc(20, file.path(tmpdir, "mcmc.#.csv"), sep=",")

summary(data)
```

Index