Package ‘codadiags’

February 19, 2015

Type Package

Title Markov chain Monte Carlo burn-in based on ``bridge'' statistics

Version 1.0

Date 2013-09-17

Author Yann Richet, Xavier Bay, Olivier Jacquet, Alexis Jinaphanh

Maintainer Yann Richet <yann.richet@irsn.fr>

Description Markov chain Monte Carlo burn-in based on ``bridge'' statistics, in the way of coda::heidel.diag, but including non asymptotic tabulated statistics.

License GPL-3

Depends R (>= 3.0.0), coda

LazyData no

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-11-18 10:28:00

R topics documented:

`codadiags-package` *Markov chain Monte Carlo burn-in based on "bridge" statistics.*

Description

This package implements many burn-in procedures based on iterative transient statistic tests. The tests are calibrated on simple AR1 MCMC process, tuned on particle simulation Monte Carlo codes. It should be viewed as a non asymptotic declination of heidel.diag burnin from coda package.

Details

Package:	codadiags
Type:	Package
Version:	1.0
Date:	2013-09-17
License:	GPL-3
Depends:	coda

Author(s)

Yann Richet, Xavier Bay, Olivier Jacquet, Alexis Jinaphanh Maintainer: Yann Richet <yann.richet@irsn.fr>

References

Y. Richet, PhD: "Suppression of the initial transient in Monte Carlo criticality simulations", University of St Etienne, France, 2009.

Examples

```
require(codadiags)
set.seed(123)
x = AR1()
print(bridgestat.diag(x))
y = add.transient(x)
print(bridgestat.diag(y,trunc=10))
```

`ad.cdf` *Anderson-Darling cumulative density function, copy from ADGofTest package.*

Description

Anderson-Darling cumulative density function, copy from ADGofTest package.

Usage

```
ad.cdf(x, n = 1000)
```

Arguments

x	value to test
n	sample size for Anderson-Darling statistic

Author(s)

Carlos J. Gil Bellosta

References

G. and J. Marsaglia, "Evaluating the Anderson-Darling Distribution", Journal of Statistical Software, 2004

Examples

```
require(codadiags)
plot(null.lim.cdf("loglik_mean.brownianbridge",forceUseECDF=TRUE),col='blue',xlim=c(-4,0))
plot(Vectorize(function(x)1-ad.cdf(-x)),col='green',add=TRUE,xlim=c(-4,0))
```

---

add.transient *Add a transient to a given mcmc sequence*

Description

Add a transient to a given mcmc sequence

Usage

```
add.transient(X, a = 100, b = a + 100, step = -1)
```

Arguments

X	sequence to add the transient on
a	last iteration of the constant transient part
b	last iteratio of the transient
step	transient step

Examples

```
require(codadiags)
x = AR1()
plot(x,type='l',col=rgb(.5,0,0,.5))
y = add.transient(x)
lines(y,col=rgb(0,0,0.5,.5))
transient.test(x)
transient.test(y)
```

AR1 *Generate auto-regressive order 1 sequence*

Description

Generate auto-regressive order 1 sequence

Usage

```
AR1(length = 1000, mean = 0, sd = 1, rho = 0.1,
  rand = function() {     rnorm(1) })
```

Arguments

`length`	size of the sequence
`mean`	mean value of the sequence $$\mu$$
`sd`	standard deviation of the sequence $$\sigma$$
`rho`	auto-correlation factor
`rand`	function to generate one random step

Value

Auto Regressive ("AR") sequence

$$X = \mu * \{x_n\}_{1 \leq n \leq N}, x_1 = \sigma * u_1 / \sqrt{1 - \rho^2}, x_n = \rho * x_{n-1} + \sigma * u_n$$

Examples

```
x = AR1()
plot(x,type='l',col=rgb(.5,0,0,.5))
```

autocorr1 — *Basic auto-correlation estimation of a given sequence*

Description

Basic auto-correlation estimation of a given sequence

Usage

```
autocorr1(X)
```

Arguments

`X`	sequence

Value

first auto-correlation coefficient

bay.cdf — *Bay cumulative density function, corresponding to -B(t+)/B(t-), where B(t+) (resp. B(t-)) is the maximum (resp.minimum) of B(t)/(t*(1-t)).*

Description

Bay cumulative density function, corresponding to -B(t+)/B(t-), where B(t+) (resp. B(t-)) is the maximum (resp.minimum) of B(t)/(t*(1-t)).

Usage

```
bay.cdf(x)
```

Arguments

`x`	value to test

Author(s)

X. Bay

`bridgestat.diag` *Iterative truncation procedure based on a bridge statistic.*

Description

Iterative truncation procedure based on a bridge statistic.

Usage

```
bridgestat.diag(x, bridge = "student", stat = "E",
  param = "asymptotic", trunc = 1, eps = 0.1,
  pvalue = 0.3)
```

Arguments

`x`	coda::mcmc sequence (will be cast to if necessary) to truncate transient
`bridge`	bridge type to use: "brownian", student" or "loglik"
`stat`	statistic to use for testing bridge: - if student bridge, "E","var","autocov","loglik_mean","loglik_extremum - if brownian bridge, "E","var","autocov","loglik_mean","loglik_extremum","extremum","ratio_extremu - if loglik bridge, "E","var","autocov","extremum","ratio_extremum"
`param`	if "asymptotic" use asymptotic statistics, else if a list of 'N' and 'rho' use these parameters, if NULL estimate N and rho
`trunc`	number of mcmc iterations to delete: if >=1, it is a constant number, if <1, a percentage of remaining batches
`eps`	Target value for ratio of halfwidth to sample mean (for compatibility with heidel.diag)
`pvalue`	significance level to use in iterative test

References

Heidelberger P and Welch PD. Simulation run length control in the presence of an initial transient. Opns Res., 31, 1109-44 (1983)

See Also

coda::heidel.diag

Examples

```
require(codadiags)
set.seed(123)
x = AR1()
print(bridgestat.diag(x))
y = add.transient(x)
print(bridgestat.diag(y,trunc=10))
```

brownianbridge — *Compute the so called (abusively) "Brownian bridge" process.*

Description

Compute the so called (abusively) "Brownian bridge" process.

Usage

```
brownianbridge(X)
```

Arguments

X	MCMC sampling sequence of length N

Value

cumulative normalized sum sequence:

$$B = \{b_n\}_{0 \le n \le N}, b_n = \frac{n * (\hat{\mu}_{1,n} - \hat{\mu}_{1,N})}{\hat{\sigma}\sqrt{(N)}}$$

Examples

```
x = AR1(rho=0)
bb = brownianbridge(x)
plot(bb,type='l',col='red')
```

cvm.cdf — *Cramer von Mises cumulative density function, import from coda package.*

Description

Cramer von Mises cumulative density function, import from coda package.

Usage

```
cvm.cdf(x)
```

Arguments

x	value to test

References

Csorgo S. and Faraway, JJ. The exact and asymptotic distributions of the Cramer-von Mises statistic. J. Roy. Stat. Soc. (B), 58, 221-234 (1996).

See Also

coda::pcramer

Examples

```
require(codadiags)
plot(null.lim.cdf("var.brownianbridge",forceUseECDF=TRUE),col='blue')
plot(Vectorize(cvm.cdf),col='green',add=TRUE)
```

---

ks.cdf *Kolmogorov-Smirnov cumulative density function, copy from stats::ks.test.*

Description

Kolmogorov-Smirnov cumulative density function, copy from stats::ks.test.

Usage

```
ks.cdf(x, n = 100)
```

Arguments

x	value to test
n	sample size for Kolmogorov-Smirnov statistic

References

George Marsaglia, Wai Wan Tsang and Jingbo Wang (2003), Evaluating Kolmogorov's distribution. Journal of Statistical Software, 8/18. http://www.jstatsoft.org/v08/i18/.

See Also

package stats, ks.c

Examples

```
require(codadiags)
plot(null.lim.cdf("extremum.brownianbridge",forceUseECDF=TRUE),col='blue',xlim=c(0.01,4))
plot(Vectorize(ks.cdf),col='green',add=TRUE,xlim=c(0.01,4))
```

loglikbridge — *Compute the so called "Log-likelihood bridge" process.*

Description

Compute the so called "Log-likelihood bridge" process.

Usage

```
loglikbridge(X)
```

Arguments

X	MCMC sampling sequence of length N

Value

log-likelihood sequence

$$LL = \{ll_n\}_{2\leq n\leq N-2}, ll_n = N * ln(\hat{\sigma}^2_{1,N}) - n * ln(\hat{\sigma}^2_{1,n}) - (N-n)ln(\hat{\sigma}^2_{n+1,N})$$

Examples

```
x = AR1(rho=0)
llb = loglikbridge(x)
plot(llb,type='l',col='red')
```

maxinv.bay.cdf — *CDF of max(x,1/x) (=cdf(x)-cdf(1)+cdf(1)-cdf(1/x)) where x is 'Bay' distributed*

Description

CDF of max(x,1/x) (=cdf(x)-cdf(1)+cdf(1)-cdf(1/x)) where x is 'Bay' distributed

Usage

```
maxinv.bay.cdf(x)
```

Arguments

x	value to test

Examples

```
require(codadiags)
plot(null.lim.cdf("ratio_extremum.brownianbridge",forceUseECDF=TRUE),col='blue',xlim=c(0,10))
plot(Vectorize(maxinv.bay.cdf),col='green',add=TRUE,xlim=c(0,10))
```

null.lim.cdf *Asymptotic CDF for a given statistic*

Description

Asymptotic CDF for a given statistic

Usage

```
null.lim.cdf(stat, forceUseECDF = FALSE)
```

Arguments

stat	statistic
forceUseECDF	if true, - if stat is loglik_mean.brownianbridge, use the Anderson-Darling CDF #NO - if stat is var.brownianbridge, use the Cramer von Mises CDF - if stat is extremum.brownianbridge, use the Kolmogorov-Smirnov CDF #NO - if stat is ratio_loglik_extremum.brownianbridge, use the chi-square (3 freedom degrees) CDF - if stat is ratio_extremum.brownianbridge, use the Bay CDF else, use tabulated empirical CDF built on white noise process (length 10000)

null.param.cdf *Build the null CDF (cumulative density function) for a given statistic, for arbitrary length and autocorrelation sequence.*

Description

Build the null CDF (cumulative density function) for a given statistic, for arbitrary length and autocorrelation sequence.

Usage

```
null.param.cdf(stat, N, rho)
```

Arguments

stat	statistic used
N	length of the target sequence to be tested
rho	autocorrelation (1st coeff) of the target sequence to test

studentbridge *Compute the so called "Student bridge" process.*

Description

Compute the so called "Student bridge" process.

Usage

```
studentbridge(X)
```

Arguments

X	MCMC sampling sequence of length N

Value

Student bridge sequence:

$$S = \{s_n\}_{1 \leq n \leq N-1}, s_n = \sqrt{N-2} \frac{n * (\hat{\mu}_{1,n} - \hat{\mu}_{n+1,N})}{\sqrt{\left(\frac{1}{n} + \frac{1}{N-n}\right) * ((n-1)\hat{\sigma_{1,n}}^2 + (N-n-1)\hat{\sigma_{n+1,N}}^2}}$$

Examples

```
x = AR1(rho=0)
sb = studentbridge(x)
plot(sb,type='l',col='blue')
```

transient.test *Perform a stationary test to check for an initial burn-in in a sequence*

Description

Perform a stationary test to check for an initial burn-in in a sequence

Usage

```
transient.test(x, bridge = studentbridge,
  stat = E.studentbridge, param = NULL, plot = FALSE)
```

Arguments

- `x` — sequence
- `bridge` — bridge builder function
- `stat` — statistic of the bridge to use in the test
- `param` — sequence parameters: length 'N' and first auto-correlation coefficient 'rho', or "asymptotic" for default asymptotic parameters, or NULL for auto estimated parameters
- `plot` — boolean to ask for test plots

Value

A list with class "htest" containing the following components: statistic: the value of the test statistic. p.value: the p-value of the test. method: a character string indicating what type of test was performed. data.name: character string giving the name(s) of the data.

Index