Package 'bspec'

April 17, 2009

Type Package

Title Bayesian spectral inference

Version 1.1

Date 2008-04-29

Author Christian Roever

Maintainer Christian Roever <bspec@web.de>

Description Bayesian inference on the (discrete) power spectrum of time series.

License GPL (>= 2)

Repository CRAN

Date/Publication 2008-04-29 18:15:58

R topics documented:

`acf.bspec` *Posterior autocovariances*

Description

Deriving (posterior) autocovariances or autocorrelations from the spectrum's posterior distribution.

Usage

```
## S3 method for class 'bspec':
acf(x, spec = NULL,
   type = c("covariance", "correlation"),
   two.sided = x$two.sided, ...)
```

Arguments

`x`	a `bspec` object.
`spec`	(optional) a `numeric` vector giving *fixed* values of the spectral parameters (e.g. derived by the `sample` function) for which the autocovariances then are deterministic.
`type`	a `character` string specifying the desired type of output.
`two.sided`	a `logical` flag indicating whether the `spec` values are to be interpreted as *one-sided* or *two-sided*.
`...`	

Details

If `spec` is supplied, the autocovariance (or autocorrelation) function corresponding to that specific spectrum will be returned. As this is a completely deterministic relationship, the "`stderr`" slot of the result will be zero in this case.

If `spec` is *not* supplied, the *(posterior) expected autocovariance* is returned in the "`acf`" element, and its *(posterior) standard deviation* is returned in the "`stderr`" element. The posterior expectation of the autocovariance is only finite if *all (!)* posterior degrees-of-freedom parameters in the `bspec` object are > 2. The posterior variance (and with that the `stderr` element) is only finite if all these are > 4.

Autocorrelations are only returned if `spec` is supplied.

Value

A list of class `bspecACF` containing the following components:

`lag`	a `numeric` vector giving the lags corresponding to the (discrete) autocovariance / autocorrelation values.
`acf`	a `numeric` vector giving the values of the autocovariance / autocorrelation function correponding to the above lags.

stderr	a numeric vector giving the standard errors (posterior standard deviations) of the above autocovariance values.
type	a character string giving the nature of the above acf element: either "covariance" or "correlation".
N	an integer giving the sample size of the original time series.
bspec	a character string giving the name of the bspec object the bspecACF object was generated from.

Note

(Posterior) expectation and standard deviation of the spectrum may in many cases not be finite (see above). Autocorrelations are only returned if spec is supplied.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

References

Roever, C., Meyer, R., Christensen, N. (2008): Modelling coloured noise. Arxiv preprint 0804.3853 [stat.ME], http://arxiv.org/abs/0804.3853, (*submitted for publication*).

See Also

bspec, expectation, sample.bspec, acf

Examples

```
lhspec1 <- bspec(lh)

# without any prior specifications,
# autocovariances are not finite:
print(acf(lhspec1))
str(acf(lhspec1))

# for given values of the spectral parameters,
# the autocovariances are fixed:
str(acf(lhspec1, spec=sample(lhspec1)))

# for all the prior degrees-of-freedom greater than one,
# the expected autocovariance is finite, its variance isn't:
lhspec2 <- bspec(lh, priordf=2, priorscale=0.6, intercept=FALSE)
print(acf(lhspec2))
str(acf(lhspec2))
plot(acf(lhspec2))
```

bspec *Computing the spectrum's posterior distribution*

Description

Derives the posterior distribution of the spectrum of one or several time series, based on data and prior specifications.

Usage

```
bspec(x, ...)
## Default S3 method:
bspec(x, priorscale=1, priordf=0, intercept=TRUE,
  two.sided=FALSE, ...)
```

Arguments

`x`	a time series object of the data to be analysed. May be a univariate (`ts` object) or multivariate (`mts` object) time series.
`priorscale`	*either* a `numeric` vector giving the scale parameters of the spectrum's prior distribution; recycled if of length 1. *Or* a `function` of frequency.
`priordf`	*either* a `numeric` vector giving the degrees-of-freedom parameters of the spectrum's prior distribution; recycled if of length 1. *Or* a `function` of frequency.
`intercept`	a `logical` flag indicating whether to include the 'intercept' (zero frequency) term.
`two.sided`	a `logical` flag indicating whether to refer to a one-sided or a two-sided spectrum. In particular affects the interpretation of the prior scale parameters, and sets the default for some methods applied to the resulting `bspec` object via its `two.sided` element.
`...`	

Details

Based on the assumptions of a zero mean and a finite spectrum, the posterior distribution of the (discrete) spectrum is derived. The data are modeled using the *Maximum Entropy* (Normal) distribution for the above constraints, and based on the prior information about the spectrum specified in terms of the (conjugate) *scaled inverse* χ^2 *distribution*.

For more details, see the references.

Value

A list of class `bspec` containing the following elements:

`freq`	a `numeric` vector giving the (Fourier-) frequencies that the spectral parameters correspond to.
`scale`	a `numeric` vector giving the scale parameters of the posterior distributions of the spectral parameters corresponding to the above frequencies. These - internally- always correspond to the *one-sided* spectrum, regardless of the `two.sided` flag (see below).
`df`	a `numeric` vector giving the degrees-of-freedom parameters of the posterior distributions of the spectral parameters corresponding to the above frequencies.
`priorscale`	a `numeric` vector giving the prior scale parameters.
`priordf`	a `numeric` vector giving the prior degrees-of-freedom parameters.
`datassq`	a `numeric` vector giving the sum-of-squares contributed by the data.
`datadf`	a `numeric` vector giving the degrees-of-freedom contributed by the data.
`N`	the sample size of the original time series.
`deltat`	the sampling interval of the original time series.
`deltaf`	the frequency interval of the Fourier-transformed time series.
`start`	the time of the first observation in the original time series.
`call`	an object of class `call` giving the function call that generated the `bspec` object.
`two.sided`	a `logical` flag indicating whether the spectrum is to be interpreted as one-sided or two-sided.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

References

Roever, C., Meyer, R., Christensen, N. (2008): Modelling coloured noise. Arxiv preprint 0804.3853 [stat.ME], http://arxiv.org/abs/0804.3853, (*submitted for publication*).

See Also

`expectation`, `quantile.bspec`, `sample.bspec`, `ppsample`, `acf.bspec`, `spectrum`

Examples

```
# determine spectrum's posterior distribution
# (for noninformative prior):
lhspec <- bspec(lh)
print(lhspec)

# show some more details:
str(lhspec)
```

```
# plot 95 percent central intervals and medians:
plot(lhspec)

# draw and plot a sample from posterior distribution:
lines(lhspec$freq, sample(lhspec), type="b", pch=20)

########

# compare the default outputs of "bspec()" and "spectrum()":
bspec1    <- bspec(lh)
spectrum1 <- spectrum(lh, plot=FALSE)
plot(bspec1)
lines(spectrum1$freq, spectrum1$spec, col="blue")
# (note -among others- the factor 2 difference)

# match the outputs:
# Need to suppress  tapering, padding and de-trending
# (see help for "spec.pgram()"):
spectrum2 <- spectrum(lh, taper=0, fast=FALSE, detrend=FALSE, plot=FALSE)
# Need to drop intercept (zero frequency) term:
bspec2    <- bspec(lh, intercept=FALSE)
# plot the "spectrum()" output:
plot(spectrum2)
# draw the "bspec()" scale parameters, adjusted
# by the corresponding degrees-of-freedom,
# so they correspond to one-sided spectrum:
lines(bspec2$freq, bspec2$scale/bspec2$datadf,
      type="b", col="green", lty="dashed")

########

# handle several time series at once...
data(sunspots)
# extract three 70-year segments:
spots1 <- window(sunspots, 1750, 1819.99)
spots2 <- window(sunspots, 1830, 1899.99)
spots3 <- window(sunspots, 1910, 1979.99)
# align their time scales:
tsp(spots3) <- tsp(spots2) <- tsp(spots1)
# combine to multivariate time series:
spots <- ts.union(spots1, spots2, spots3)
# infer spectrum:
plot(bspec(spots))
```

`expectation` *Expectations and variances of distributions*

Description

Functions to compute (posterior) expectations or variances of the distributions specified as arguments.

Usage

```
expectation(x, ...)
variance(x, ...)
## S3 method for class 'bspec':
expectation(x, two.sided=x$two.sided, ...)
## S3 method for class 'bspec':
variance(x, two.sided=x$two.sided, ...)
## S3 method for class 'bspecACF':
expectation(x, ...)
## S3 method for class 'bspecACF':
variance(x, ...)
```

Arguments

`x`	A `bspec` or `bspecACF` object.
`two.sided`	A `logical` flag to indicate whether to compute expectation / variance of one- or two-sided spectrum, *if* the argument `x` is a `bspec` object.
`...`	

Value

A `numeric` vector giving the expectations/variances corresponding to the frequencies or lags of the argument.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

References

Roever, C., Meyer, R., Christensen, N. (2008): Modelling coloured noise. Arxiv preprint 0804.3853 [stat.ME], http://arxiv.org/abs/0804.3853, (*submitted for publication*).

See Also

`bspec`, `acf.bspec`

Examples

```
# note the changing expectation
# with increasing prior/posterior degrees-of-freedom:
expectation(bspec(lh))
expectation(bspec(lh, priordf=1, priorscale=0.6))
expectation(bspec(lh, priordf=2, priorscale=0.6))
```

```
# similar for variance:
variance(bspec(lh, priordf=2, priorscale=0.6))
variance(bspec(lh, priordf=3, priorscale=0.6))

# and again similar for autocovariances:
expectation(acf(bspec(lh)))
expectation(acf(bspec(lh, priordf=2, priorscale=0.6)))
variance(acf(bspec(lh)))
variance(acf(bspec(lh, priordf=4, priorscale=0.6)))
```

`likelihood` *Prior, likelihood and posterior*

Description

Prior density, likelihood, and posterior density functions for the posterior distributions specified through a `bspec` object.

Usage

```
dprior(x, ...)
likelihood(x, ...)
dposterior(x, ...)
## S3 method for class 'bspec':
dprior(x, theta, two.sided=x$two.sided, log=FALSE, ...)
## S3 method for class 'bspec':
likelihood(x, theta, two.sided=x$two.sided, log=FALSE, ...)
## S3 method for class 'bspec':
dposterior(x, theta, two.sided=x$two.sided, log=FALSE, ...)
```

Arguments

`x`	a `bspec` object.
`theta`	a `numeric` vector of parameter values, corresponding to the Fourier frequencies in the `x$freq` element.
`two.sided`	a `logical` flag indicating whether the parameters `theta` correspond to the *one-sided* or *two-sided* spectrum.
`log`	a `logical` flag indicating whether to return logarithmic density (or likelihood) values.
`...`	

Details

Prior and posterior are both *scaled inverse* χ^2 *distributions*, and the likelihood is Normal.

Value

A numeric function value.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

References

Roever, C., Meyer, R., Christensen, N. (2008): Modelling coloured noise. Arxiv preprint 0804.3853 [stat.ME], http://arxiv.org/abs/0804.3853, (*submitted for publication*).

See Also

bspec, quantile.bspec, expectation

Examples

```
lhspec <- bspec(lh, priordf=1, priorscale=0.6)

# draw sample from posterior:
posteriorsample <- sample(lhspec)

# plot the sample:
plot(lhspec)
lines(lhspec$freq, posteriorsample, type="b", col="red")

# compute prior, likelihood, posterior:
print(c("prior"     = dprior(lhspec, posteriorsample),
        "likelihood"= likelihood(lhspec, posteriorsample),
        "posterior" = dposterior(lhspec, posteriorsample)))
```

---

one.sided *Conversion between one- and two-sided spectra*

Description

Functions to convert between one- and two-sided bspec objects.

Usage

```
one.sided(x, ...)
two.sided(x, ...)
## S3 method for class 'bspec':
one.sided(x, ...)
## S3 method for class 'bspec':
two.sided(x, ...)
```

Arguments

x	a bspec object.
...	

Details

The conversion only means that the $two.sided element of the returned bspec object is set correspondingly, as internally always the same (one-sided) spectrum is used.

Value

A bspec object (see the help for the bspec function).

Author(s)

Christian Roever, ⟨bspec@web.de⟩

See Also

bspec

Examples

```
lhspec <- bspec(lh)

# compare distributions visually:
par(mfrow=c(2,1))
  plot(lhspec)
  plot(two.sided(lhspec))
par(mfrow=c(1,1))

# ...and numerically:
print(cbind("frequency"=lhspec$freq,
            "median-1sided"=quantile(lhspec,0.5),
            "median-2sided"=quantile(two.sided(lhspec),0.5)))
```

---

ppsample *Posterior predictive sampling*

Description

Draws a sample from the posterior predictive distribution specified by the supplied bspec object.

Usage

```
ppsample(x, ...)
## S3 method for class 'bspec':
ppsample(x, start=x$start, ...)
```

Arguments

x	a bspec object specifying the posterior distribution from which to sample.
start	the start time of the resulting time series.
...	

Value

A time series (ts) object of the same kind (with respect to sampling rate and sample size) as the data the posterior distribution is based on.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

See Also

bspec, sample.bspec

Examples

```
par(mfrow=c(2,1))
plot(lh, main="'lh' data")
plot(ppsample(bspec(lh)), main="posterior predictive sample")
par(mfrow=c(1,1))
```

---

quantile.bspec *Quantiles of the posterior spectrum*

Description

Function to compute quantiles of the spectrum's posterior distribution specified through the supplied bspec object argument.

Usage

```
## S3 method for class 'bspec':
quantile(x, probs = c(0.025, 0.5, 0.975),
  two.sided = x$two.sided, ...)
```

Arguments

x	a bspec object.
probs	a numerical vector of probabilities.
two.sided	a logical flag indicating whether quantiles are supposed to correspond to the *one-sided* or *two-sided* spectrum.
...	

Details

The posterior distribution is a product of independent *scaled inverse* χ^2 *distributions*.

Value

A matrix with columns corresponding to elements of `probs`, and rows corresponding to the Fourier frequencies `x$freq`. If `probs` is of length 1, a vector is returned instead.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

See Also

`bspec`, `quantile`

Examples

```
lhspec <- bspec(lh)

# posterior medians:
print(cbind("frequency"=lhspec$freq,
            "median"=quantile(lhspec, 0.5)))
```

`sample.bspec` *Posterior sampling*

Description

Function to generate samples from the spectrum's posterior distribution specified through the supplied `bspec` object argument.

Usage

```
## S3 method for class 'bspec':
sample(x, size = 1, two.sided = x$two.sided, ...)
```

Arguments

`x`	a `bspec` object.
`size`	the sample size.
`two.sided`	a `logical` flag indicating whether the drawn samples are supposed to correspond to the *one-sided* or *two-sided* spectrum.
`...`	

Details

The posterior distribution is a product of independent *scaled inverse* χ^2 *distributions*.

Value

A (numerical) vector of samples from the posterior distribution of the spectral parameters, of the same length as and corresponding to the $freq element of the supplied bspec object.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

See Also

bspec, ppsample

Examples

```
# determine spectrum's posterior distribution:
lhspec <- bspec(lh)

# plot 95 percent central intervals and medians:
plot(lhspec)

# draw and plot two samples from posterior distribution:
lines(lhspec$freq, sample(lhspec), type="b", pch=20, col="red")
lines(lhspec$freq, sample(lhspec), type="b", pch=20, col="green")
```

---

temper *Tempering of (posterior) distributions*

Description

Setting the tempering parameter of ('tempered') bspec objects.

Usage

```
temper(x, ...)
## S3 method for class 'bspec':
temper(x, temperature = 2, likelihood.only = TRUE, ...)
```

Arguments

x	a bspec object.
temperature	a (positive) 'temperature' value.
likelihood.only	a logical flag indicating whether to apply the tempering to the 'complete' posterior density, or to the likelihood only (default).
...	

Details

In the context of Markov chain Monte Carlo (MCMC) applications it is often desirable to apply *tempering* to the distribution of interest, as it is supposed to make the distribution more easily tractable. Examples where tempering is utilised are *simulated annealing*, *parallel tempering* or *evolutionary MCMC* algorithms. In the context of Bayesian inference, tempering may be done by specifying a 'temperature' T and then manipulating the original posterior distribution $p(\theta|y)$ by applying an exponent $\frac{1}{T}$ either to the complete posterior distribution:

$$p_T(\theta) \propto p(\theta|y)^{\frac{1}{T}} = (p(y|\theta)p(\theta))^{\frac{1}{T}}$$

or to the likelihood part only:

$$p_T(\theta) \propto p(y|\theta)^{\frac{1}{T}} p(\theta).$$

In this context, where the posterior distribution is a product of *scaled inverse* χ^2 *distributions*, the tempered distributions in both cases turn out to be again of the same family, just with different parameters. For more details see also the references.

Value

An object of class `bspec` (see the help for the `bspec` function), but with an additional `temperature` element.

Note

Tempering with the `likelihood.only` flag set to `FALSE` only works as long as the `temperature` is less than `min((x$df+2)/2)`.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

References

Roever, C., Meyer, R., Christensen, N. (2008): Modelling coloured noise. Arxiv preprint 0804.3853 [stat.ME], http://arxiv.org/abs/0804.3853, (*submitted for publication*).

See Also

`temperature`, `bspec`

Examples

```
lhspec <- bspec(lh, priorscale=0.6, priordf=1)

# details of the regular posterior distribution:
str(lhspec)

# details of the tempered distribution
# (note the differing scale and degrees-of-freedom):
str(temper(lhspec, 1.23))
```

`temperature` *Querying the tempering parameter*

Description

Retrieving the "temperature" parameter of ('tempered') `bspec` objects

Usage

```
temperature(x, ...)
## S3 method for class 'bspec':
temperature(x, ...)
```

Arguments

`x`	a `bspec` object.
`...`	

Value

The (`numeric`) value of the `temperature` element of the supplied `bspec` object, if present, and 1 otherwise.

Author(s)

Christian Roever, ⟨bspec@web.de⟩

References

Roever, C., Meyer, R., Christensen, N. (2008): Modelling coloured noise. Arxiv preprint 0804.3853 [stat.ME], http://arxiv.org/abs/0804.3853, (*submitted for publication*).

See Also

`temper`, `bspec`

Examples

```
lhspec1 <- bspec(lh)
lhspec2 <- temper(lhspec1, 1.23)

print(lhspec2$temperature)
print(lhspec1$temperature)

print(temperature(lhspec2))
print(temperature(lhspec1))
```

Index