Package 'regspec'

June 1, 2022

Type Package

Title Non-Parametric Bayesian Spectrum Estimation for Multirate Data

Version 2.6

Date 2022-06-01

Description Computes linear Bayesian spectral estimates from multirate data for second-order stationary time series. Provides credible intervals and methods for plotting various spectral estimates. Please see the paper `Should we sample a time series more frequently?' (doi below) for a full description of and motivation for the methodology.

URL https://doi.org/10.1111/rssa.12210

License GPL-2

LazyData TRUE

NeedsCompilation no

Author Ben Powell [aut, cre],
Guy Nason [aut]

Maintainer Ben Powell <ben.powell@york.ac.uk>

Repository CRAN

Date/Publication 2022-06-01 11:50:02 UTC

R topics documented:

regspec-package *Non-parametric multirate spectral density estimation via linear Bayes.*

Description

Estimate a spectral density function of a stationary time series. Produces a linear Bayes estimate with credible intervals. Can incorporate time series data from multiple realizations with different sampling rate. Can deal with time series data that has been filtered with a known filter (e.g. quarterly totals from monthly values).

Details

Package:	regspec
Type:	Package
Version:	1.0
Date:	2014-04-22
License:	MIT

Currently, the package's centrepiece is the function `regspec`, which implements spectral density estimation from time series data at integer time points. A novel element of the code is it's ability to assimilate subsampled and filter data.

The package's data files, which will be loaded automatically, include synthetic and real examples of time series data that feature in the `Examples` sections of the functions' help files.

Author(s)

Ben Powell

References

Nason, G.P., Powell, B., Elliott, D. and Smith, P. (2016) Should We Sample a Time Series More Frequently? Decision Support via Multirate Spectrum Estimation. Journal of the Royal Statistical Society, Series A., 179, (to appear).

See Also

`regspec`

basis *Basis maker.*

Description

A function for making matrices of sinusoidal basis function values.

Usage

```
basis(x, nb)
```

Arguments

x	The frequencies at which to evaluate the basis functions.
nb	The number of basis functions to include.

Value

A matrix whose rows correspond to input values and whose columns correspond to particular basis functions.

Author(s)

Ben Powell

Examples

```
bas.mat<-basis(seq(0,0.5,length=16),22)
```

Gaussbound *Compute the Gauss bounds for a random variable.*

Description

This is a simple function that computes bounds for a credible interval according to Gauss's inequality. If a random variable has a Lebesgue density with a single mode (mode) and a finite expected squared deviation (tau^2) from this mode, then Gauss's inequality tells us that at least a given proportion (prob) of the distribution's mass lies within a finite symmetric interval centred on the mode.

Usage

```
Gaussbound(mode, tau, prob)
```

Arguments

mode	Numeric. The location of the density's mode.
tau	Numeric. The square root of the expected squared deviation from the mode.
prob	Numeric. A lower bound on the probability mass that is contained within the interval

Value

bounds	An ordered vector containing the lower and upper bounds of the interval.

Author(s)

Ben Powell

References

Pukelsheim, F. (1994) The Three Sigma Rule. *The American Statistician* **48**, 88-91.

Examples

```
Gaussbound(1,1,0.9)
```

---

hindcast *Interpolate process values from a set of data, and an expectation and variance specification for the process's log spectrum.*

Description

This function computes an approximate expectation for a (second-order stationary) process's autocovariance function from the first two moments of its log-spectrum, as encoded in an expectation vector and variance matrix for the coefficients of a basis representation. It then uses this autocovariance to interpolate values of a process and to calculate variances for them.

The function is really here to facilitate the reproduction of an example from Nason, Powell, Elliott and Smith (2016). It may be studied as an example, but is not recommended for general use. Instead, custom Kriging-type estimates ought to be produced by manipulating by hand variance matrices populated with autocovariance function values, which can be computed with the function `logspec2cov`.

Usage

```
hindcast(Dhigh, hightimes, Dlow, lowtimes, predtimes, filter=c(1),
    ebeta, vbeta, SARIMA)
```

Arguments

Dhigh	Vector. The high frequency data.
hightimes	Vector. Integer time points at which the high frequency observations are made.
Dlow	Vector. The low frequency data.
lowtimes	Vector. Integer time points at which the low frequency observations are made.
predtimes	Vector. Integer time points at which hindcasts are required.
filter	Vector. A known vector of filter coefficients arising from the observation process prior to any subsampling. The default is NULL, which corresponds to direct observation and a filter vector of (1,0,0,...). If the data are produced by taking a linear combination of the current and previous process values, for example, one would set this vector to be (w_{t},w_{t-1}).
ebeta	Vector. Expectations for basis coefficients of the log spectrum.
vbeta	Vector. The variance for the basis coefficients of the log spectrum.
SARIMA	List. A list encoding the SARIMA model that acts as an intercept, or base line, for the non-parametric estimate of the log-spectrum. The default is white noise with variance one. The log-spectrum basis coefficients parameterize a deviation away from the SARIMA model's log-spectrum. The contents of the SARIMA list are formatted in line with the format used by the package TSA (see the Examples section for examples).

Value

hindcast	A vector of hindcast expectations
var.hindcast	A covariance matrix for the hindcast values.

Author(s)

Ben Powell

References

Nason, G.P., Powell, B., Elliott, D. and Smith, P. (2016) Should We Sample a Time Series More Frequently? Decision Support via Multirate Spectrum Estimation. Journal of the Royal Statistical Society, Series A., 179, (to appear).

Examples

```
#
# See example in \code{\link{travel}} help file
#
```

`logspec2cov` *Compute autocovariance values from the basis coefficients for the log spectrum.*

Description

This function performs a series of integrals of a spectral density multiplied by sinusoids of increasing frequency in order. The result is a vector of autocovariances for the process values at increasing separation.

The example below shows how this function is useful for informing estimates of missing values in thinned time series data.

Usage

```
logspec2cov(ebeta, vbeta, SARIMA=list(sigma2=1), lags, subdivisions=100)
```

Arguments

`ebeta`	Vector. Expectations for basis coefficients of the log spectrum.
`vbeta`	Vector. The variance for the basis coefficients of the log spectrum.
`SARIMA`	List. A list encoding the SARIMA model that acts as the intercept, or base line, for the non-parametric estimate of the log-spectrum. The default is white noise with variance one. The log-spectrum basis coefficients parameterize a deviation away from the SARIMA model's log-spectrum. The contents of the SARIMA list are formatted in line with the format used by the package `TSA` (see the `Examples` section for examples).
`lags`	Integer. The number of lags to which to calculate the autocovariance.
`subdivisions`	Integer. This number is passed to the numerical integrator that computes the autocovariances from the exponentiated log-spectrum. It is set to `100` by default to reduce CPU demand. For rough spectra is may be advisable to specify a larger value.

Value

`autocovariances`

A vector of approximate expectations for the process's autocovariances for increasing lags, starting with zero lag (the process variance).

Author(s)

Ben Powell

Examples

```
#
# Simulate a complete time series.

set.seed(42)
D <- arima.sim(n=150, model=list(ar=c(-0.5,0.4,0.8), ma=0.2))

# Now define indices for a subsampled historical period,
# a fully sampled historical period, and the missing values.

inda <- seq(1, 100, by=3)
indb <- seq(101, 150, by=1)
indc <- (1:150)[-c(inda, indb)]

#
# Adjust our estimate for the spectrum using the two historical periods.
#

adjustment1 <- regspec(D=D[indb], deltat=1, smthpar=0.85,
plot.spec=FALSE)

adjustment2 <- regspec(D=D[inda], deltat=3, ebeta=adjustment1$ebeta,
vbeta=adjustment1$vbeta, ylim=c(0,60))

lines(spec.true, col=1, lwd=3, lty=2)

#
# Calculate covariances corresponding to the estimate of the spectrum at
# the data locations.
#

k <- logspec2cov(adjustment2$ebeta, adjustment2$vbeta, lag=150)

#
# Compute linear predictors and variances for the missing data conditional
# on the autocovariances.
#

K <- matrix(0, 150, 150)
K <- matrix(k[1+abs(row(K)-col(K))], 150, 150)
d <- solve(K[c(inda, indb),c(inda, indb)],K[c(inda, indb), indc])
hindcast.exp <- crossprod(d, c(D[inda], D[indb]))
hindcast.var <- K[indc, indc]-crossprod(K[c(inda, indb), indc], d)

#
# Plot the observed historical data and the predictions for the missing data.
#

par(cex=0.7)
plot(NA, NA, xlim=c(0,150), ylim=range(D), xlab="time", ylab="x")

#
```

```
#(Observed process values are plotted with black circles.)
#

points(indb, D[indb], type="p", lty=2)
points(c(inda, indb), c(D[inda], D[indb]))

#
# (Hindcasts are plotted with blue circles.)
#
points(indc, hindcast.exp, col=rgb(0.2,0.5,0.7), lwd=2)
for(i in 1:length(indc)) {
lines(rep(indc[i], 2),
hindcast.exp[i]+1*c(-1, 1)*hindcast.var[i, i]^0.5,
col=rgb(0.2,0.5,0.7))
}

#
# Interpolate the plotted data and predictions.
#

x <- c(inda, indb, indc)
y <- c(D[inda], D[indb], hindcast.exp)
lines(sort(x), y[order(x)], lty=2, col=rgb(0.2,0.5,0.7,0.7))

#
# Reveal the true values.
#

# (The union of observed and unobserved data is plotted with red crosses.)
points(D,col=2,pch=4)
```

---

regspec *Non-parametric Multirate Spectral Density Estimation via Linear Bayes.*

Description

This function computes a linear Bayes estimate and approximate credible interval for the spectral density function of a realization from a (second-order stationary) time series. The function also has the ability to update an existing spectral estimate using time series data at a (potentially) different sampling rate, and this can be repeated multiple times. In this way, the routine can be used to estimate the spectrum, and credible intervals, from time series data taken at multiple sampling rates.

Usage

```
regspec(D, deltat=1, nb=100, varmult=2, smthpar=0.8, ebeta=NULL,
vbeta=NULL, filter=NULL, freq.out=seq(0,0.5,length=200),
plot.spec=TRUE, plot.log=FALSE, plot.pgram=FALSE,
plot.intervals=TRUE, ylim=NULL, SARIMA=list(sigma2=1),
centred=FALSE,intname=NULL,...)
```

Arguments

D	Vector. A time series of observations. If no prior information is specified (ie "starting case") then length of series has to be >= 2
deltat	Integer. The number of unit time intervals between observations in the data set D.
nb	Integer. The number of basis functions used to describe the spectral density.
varmult	Scalar. A scaling factor for the variance of the basis coefficients for the log-spectrum.
smthpar	Scalar. A roughness parameter between 0 and 1, controlling the exponential decay of the basis coefficient variances. Smaller values correspond to greater preference for smoothness.
ebeta	Vector. Prior expectations for the basis coefficients of the log spectrum. Specifying prior moments for the coefficients overrides prior information encoded in forvarest.
vbeta	Matrix. Prior covariances for the basis coefficients of the log spectrum.
filter	Vector. A known vector of filter coefficients arising from the observation process prior to any subsampling. The default is NULL, which corresponds to direct observation and a filter vector of (1,0,0,...). If the data are produced by taking a linear combination of the current and previous process values with weights w_t, for example, one would set this vector to be (w_{t},w_{t-1}).
freq.out	Vector. The frequencies at which to evaluate the estimated spectral density.
plot.spec	Logical. If TRUE some kind of spectral plot is produced. If FALSE then no plot is produced.
plot.log	Logical. Should the estimate of the log-spectrum be plotted? Plots the un-logged spectrum if FALSE.
plot.pgram	Logical. Should the periodogram values be plotted on top of the spectrum estimate?
plot.intervals	Logical. Should the pointwise credible intervals be plotted?
ylim	The usual limits that specify the range of values on the y-axis
SARIMA	List. A list encoding the SARIMA model that acts as the intercept, or base line, for the non-parametric estimate of the log-spectrum. The default is white noise with variance one. The log-spectrum basis coefficients parameterize a deviation away from the SARIMA model's log-spectrum. The contents of the SARIMA list are formatted in line with the format used by the package TSA (see the Examples section for examples).
centred	Logical. Has the data been centred? If the data D has exactly zero mean, then the log-periodogram will return infinite values. By setting this option to TRUE, the infinite log-periodogram value is ignored.
intname	Character. A name for the units the time intervals are measured in. This is just used to label the axes.
...	Other arguments for call to plot

Details

Full technical details of the calculations preformed by `regspec` are documented in Nason, Powell, Elliott and Smith (2016) listed in the references.

This function can be used to produce an estimate of the spectrum of a stationary time series realization using linear Bayes methods. The simplest call just requires the user to specify `D` the vector of time series observations.

More specialised uses of this function are as follows. 1. One can additionally specify the value of the argument `deltat` to be the sampling interval of the time series. E.g. `deltat=2` means that the time series observations were sampled every two units of time. With this argument specified the spectrum is calculated/(depicted if plotted) *still* on the frequency scale zero to one half, which is the scale normally associated with unit interval sampling. However, what changes is that the spectral estimate is neutrally extended from the *subsampled* frequency range to the unit interval range. For example, if `deltat=2` then the usual frequency range assocated with data at this sampling interval would be zero to one quarter. However, the premise of `regspec` is that ultimately the series you obtained came from a unit sampled series and so the *real spectrum* you would like to estimate is one zero to one half. Since we have no information on the higher frequencies zero to one quarter the code essentially unfolds the spectrum equally about the line of symmetry at one quarter.

If `deltat=3` or other higher values, similar unfoldings occur. For example, if `deltat=4` then two unfoldings about one quarter and then one-eighth and three-eighths are affected.

Then, subsequent calls to `regspec` at different sampling rates can alter the spectrum depending on the information they contain.

Another key parameter is the `smthpar` which is set at 0.8 by default which usually gives a nice balance between fidelity and smoothness. Increasing this parameter results in a less smooth estimate.

By default aliasing is assumed to be induced by subsampling. For example, when `deltat=2` then it is assumed that the series you have contains the evenly-indexed observations of some putative underlying integer sampled series. However, aliasing can arise in other ways, such as when your unit sampled underlying series has been filtered. For example, if one observes quarterly totals, where each total is the result of summing over consecutive three month periods then the filter is `c(1,1,1)`.

A plot of the estimated spectrum is produced automatically unless the argument `plot.spec` is set to `FALSE`.

Value

The function's output is a list with the following elements:

`freq.out`	Vector. The frequencies at which the estimated spectral density is computed.
`spec`	Vector. Point estimates of the spectrum values. Each of these is computed by exponentiating the sum of the expectation for the log spectrum and half its variance. This is the expectation consistent with the log-spectrum being normally distributed.
`logspec`	Vector. Point estimates for the log-spectrum values. These are the adjusted expectations of the log-spectrum.
`interval`	Matrix. Bounds for the 90 percent credible interval for the the spectral density.

ebeta	Vector. The adjusted expectation for the basis coefficients of the logged spectral density. They contain information on the current estimate of the spectrum and can be supplied to a further call of regspec for adjustment by new data.
vbeta	Matrix. The adjusted variance matrix for the basis coefficients of the logged spectral density. Like ebeta this matrix can be resupplied to a further call to regspec for adjustment.
pgram	List. The periodogram ordinates used in the adjustment.

Author(s)

Ben Powell code. Ben Powell and Guy Nason on help.

References

Nason, G.P., Powell, B., Elliott, D. and Smith, P. (2016) Should We Sample a Time Series More Frequently? Decision Support via Multirate Spectrum Estimation. Journal of the Royal Statistical Society, Series A., 179, (to appear).

Examples

```
## The examples here use datasets Dfexample, Dpexample2, Dpexample3 and
## spec.true, which should be loaded automatically with the package.


# FIRST EXAMPLE
# Estimates a spectrum from a time series observed at integer time points.
#
# Plot a 'point' estimate and intervals around it.
# Also plot the true spectrum afterwards with a dashed line
#
adjustment <- regspec(D=Dfexample[1:24], deltat=1, smthpar=0.8,
ylim=c(0,60), plot.pgram=TRUE)

lines(spec.true, col=1, lwd=3, lty=2)

#
#
# SECOND EXAMPLE
# Does he same except the observations are sampled at every two time units.
#
# Plot a 'point' estimate and intervals around it.

adjustment <- regspec(D=Dpexample2, deltat=2, smthpar=0.8, ylim=c(0,60))

lines(spec.true, col=1, lwd=3, lty=2)

#
# THIRD EXAMPLE
# Now estimate a spectrum from unit sampled data and put answer in the
# object called adjustment1. Then use the estimated quantities in this
# object (notably the ebeta and vbeta components) to update the spectral
```

```
# estimate by a second call to regspec using new, data sampled at even time
# points and put the result into the adjustment2 object
#

adjustment1 <- regspec(D=Dfexample[1:24], deltat=1, smthpar=0.8,
ylim=c(0,60))

lines(spec.true, col=1, lwd=3, lty=2)

adjustment2 <- regspec(D=Dpexample2, deltat=2, ebeta=adjustment1$ebeta,
vbeta=adjustment1$vbeta, ylim=c(0,60))

lines(spec.true, col=1, lwd=3, lty=2)


# FOURTH EXAMPLE
# Estimate spectrum from series observed at each third integer.
# Plot a 'point' estimate and intervals around it.

adjustment <- regspec(D=Dpexample3, deltat=3, smthpar=0.8, ylim=c(0,60))

lines(spec.true, col=1, lwd=3, lty=2)

# FIFTH EXAMPLE
# Estimate a spectrum from one time series of observations at every
# time point and then update with another at every third time point.
#
# Note how information from the first spectral estimate gets passed to
# the second call of regspec via the ebeta and vbeta components/arguments.
#

adjustment1 <- regspec(D=Dfexample[1:24], deltat=1, smthpar=0.8, ylim=c(0,60))

lines(spec.true, col=1, lwd=3, lty=2)

adjustment2 <- regspec(D=Dpexample3, deltat=3, ebeta=adjustment1$ebeta,
vbeta=adjustment1$vbeta, ylim=c(0,60))

lines(spec.true, col=1, lwd=3, lty=2)

# SIXTH EXAMPLE
#
# Estimating a spectrum from a time series of filtered observations.

# Filter the example data.

# Create empty vector
Dfexample.filtered <- c()

# Create filter
filter.vect <- 4*runif(5)

# Now produce filtered data
```

```
m <- length(filter.vect)-1

for(i in 1:(length(Dfexample)-m)){
Dfexample.filtered[i] <- crossprod(Dfexample[i+m-0:m],filter.vect)
}

# Now use filterered data to try and estimate spectrum of original data

adjustment1 <- regspec(D=Dfexample.filtered, smthpar=0.8, filter=filter.vect,
ylim=c(0,80), plot.pgram=TRUE)

lines(spec.true, col=1, lwd=3, lty=2)

# Note here how the periodogram values do not correspond to the estimated
# spectrum because the periodogram of the  filtered data is computed and
# plotted, but then is used to estimate the spectrum of the un-filtered
# process.


# SEVENTH EXAMPLE
# Estimate spectrum according to its deviation from a known SARIMA model.

# Define a SARIMA model like this one

SARIMA0 <- list(ar=0.3,sigma2=1,seasonal=list(sar=0.5,sma=0,period=12))

# or like this one

SARIMA0 <- list(ar=c(-0.5, 0.4, 0.8), ma=0.2, sigma2=1)

# Then perform adjustments as before

adjustment <- regspec(D=Dfexample[1:16], deltat=1, smthpar=0.8, ylim=c(0,60),
SARIMA=SARIMA0, plot.pgram=TRUE)

adjustment <- regspec(D=Dpexample2, deltat=2, smthpar=0.8, ylim=c(0,60),
SARIMA=SARIMA0, plot.pgram=TRUE)

lines(spec.true, col=1, lwd=3, lty=2)

# This is useful for introducing prior beliefs for the structural form of the
# spectrum. Specifically, it is useful for specifying a prior belief in
# seasonality.
```

RSI data *Retail Sales Index (RSI) data*

Description

This data frame is taken from the online data resource of the U.K.'s Office for National Statistics. It contains time-indexed values of a retail sales index, a figure describing the turnover of retail

businesses as a percentage of a 2010 baseline.

Details

The original data files are no longer hosted on the main ONS webpages but, for the foreseeable future, ought to be accessible via the UK's National Archives https://webarchive.nationalarchives.gov.uk/ukgwa/20160105160709/http://www.ons.gov.uk/ons/index.html.

References

Nason, G.P., Powell, B., Elliott, D. and Smith, P. (2016) Should We Sample a Time Series More Frequently? Decision Support via Multirate Spectrum Estimation. Journal of the Royal Statistical Society, Series A., 179, (to appear).

`Synthetic example data` *Synthetic Data for Testing Functions in the regspec Package.*

Description

`spec.true` is a matrix of coordinates for the spectral density of the ARMA(3,1) model with AR and MA coefficients (-0.5, 0.4, 0.8) and (0.2) respectively. This was the model used to simulate the example data sets `Dfexample`, `Dpexample2` and `Dpexample3`, which consist of observations of the process at every time point, at every second time point and every third time point, respectively.

Details

See the package `TSA` for functions to calculate values of the spectral density for different ARMA models, and the function `arima.sim` for simulating time series from them.

References

Nason, G.P., Powell, B., Elliott, D. and Smith, P. (2016) Should We Sample a Time Series More Frequently? Decision Support via Multirate Spectrum Estimation. Journal of the Royal Statistical Society, Series A., 179, (to appear).

`Travel data` *Visits abroad by UK residents*

Description

The Travel data in this package comprises of two data frames `trav.qly` and `trav.mly`. These contain quarterly figures from 2004 to 2010 and monthly figures from 2011 to 2013. They both show the number of UK residents making visits abroad.

Details

The data have been collected by the U.K.'s Office of National Statisitics (ONS) and are contained in the reference tables: *Overseas Travel And Tourism, Q3 2013* and *Monthly Overseas Travel and Tourism, April 2014*. The original data files are no longer hosted on the main ONS webpages but, for the foreseeable future, ought to be accessible via the UK's National Archives https://webarchive.nationalarchives.gov.uk/ukgwa/20160105160709/http://www.ons.gov.uk/ons/index.html.

Examples

```
#
# This example estimates monthly values for UK residents leaving the
# country given historical quarterly values and some more recent
# monthly values. The hindcast function is just a wrapper for code that
# computes an approximate autocovariance function for the process and
# performs some matrix calculations to produce Kriging-type estimators.
#
qt <- 1:nrow(trav.qly)*3
mt <- 84+1:nrow(trav.mly)

#
# Approximately centre data
#

Nhigh <- 14
trav.mly2 <- trav.mly[1:Nhigh,3]-5200
mt <- mt[1:Nhigh]
trav.qly2 <- trav.qly[,3]-3*5200
#
# Construct a likely prior model
#
SARIMA0 <- list(ar=0.6, seasonal=list(period=12,sar=0.6), sigma2=60^2)

#
# Learn about the log-spectrum with regspec
#
adj1 <- regspec(D=trav.mly2, deltat=1, plot.log=TRUE, plot.pgram=TRUE,
varmult=1, smthpar=0.8, SARIMA=SARIMA0, ylim=c(6,20),
intname=" (months)")

adj2 <- regspec(D=trav.qly2, deltat=3, filter=c(1,1,1), ebeta=adj1$ebeta,
vbeta=adj1$vbeta, plot.log=TRUE, ylim=c(6,20), SARIMA=SARIMA0,
plot.pgram=FALSE, intname=" (months)")

#
# Compute a hindcast
#
predtimes <- 1:84

test <- hindcast(Dhigh=trav.mly2, hightimes=mt, Dlow=trav.qly2,
lowtimes=qt, predtimes=predtimes, filter=c(1,1,1), ebeta=adj2$ebeta,
vbeta=adj2$vbeta,SARIMA=SARIMA0)
```

```
test$hindcast <- test$hindcast+5200

#
# Plot hindcast
#

plot(qt, trav.qly[,3], xlim=range(0,qt,mt), type="o",
ylim=range(0,trav.mly,trav.qly), xlab="", xaxt="n", ylab="Trips")

axyrs <- 2004:2012
axlabs <- axyrs

for(i in 1:length(axlabs)) {
axlabs[i]<-paste(axyrs[i])
}

axis(1, line=0, at=(1:41-1)*3, labels=FALSE)

text((1:length(axlabs)-1)*12, -1800, srt = 45, adj = 1,
labels = axlabs, xpd = TRUE)

points(mt, trav.mly2+5200, type="o", cex=0.6)

abline(v=84, lty=2)

points(predtimes, test$hindcast, col=rgb(0.2,0.5,0.7),
type="o", cex=0.6)

for(i in 1:length(predtimes)){
lines(rep(predtimes[i], 2),
test$hindcast[i]+3*c(-1,1)*test$var.hindcast[i,i]^0.5,
col=rgb(0.2,0.5,0.7))
}
```

Index