Package 'fractaldim'

July 22, 2025

Version 0.8-5

Date 2021-10-05

Title Estimation of Fractal Dimensions

Author Hana Sevcikova <hanas@uw.edu>,
Don Percival <dbp@apl.washington.edu>,
Tilmann Gneiting <tilmann@stat.washington.edu>

Maintainer Hana Sevcikova <hanas@uw.edu>

Depends R (>= 2.11.0), abind

Suggests wavelets, pcaPP, RandomFields, snowFT

Description Implements various methods for estimating fractal dimension of time series and 2-dimensional data <doi:10.1214/11-STS370>.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2021-10-07 08:40:02 UTC

Contents

fractaldim-package *Estimating Fractal Dimensions*

Description

Implements various methods for estimating fractal dimension of time series and 2-dimensional data.

Details

The package provides tools for estimating fractal dimension of one- or two-dimensional data, using methods described in Gneiting et al. (2010). The user can take an advantage of the available sliding window technique in which a window of a given size is slided along the data and an estimate is obtained for each position.

The main function is `fd.estimate` which can be used for one dimensional time series, as well as for two dimensional data. It computes one estimate for each method and each sliding window. It is a wrapper for lower level functions for computing just one estimate on the given data, see `fd.estim.method` for details.

Author(s)

Hana Sevcikova <hanas@uw.edu>, Tilmann Gneiting <tilmann@stat.washington.edu>, Don Percival <dbp@apl.washington.edu>

Maintainer: Hana Sevcikova <hanas@uw.edu>

References

Gneiting, T., Sevcikova, H. and Percival, D. B. (2012). Estimators of fractal dimension: Assessing the smoothness of time series and spatial data. Statistical Science, 27(2), 247-277. <doi:10.1214/11-STS370> (Version as technical report available at https://stat.uw.edu/sites/default/files/files/reports/2010/tr577.pdf)

See Also

`fd.estimate`, `fd.estim.method`

fd.estim.method *Estimation of Fractal Dimension via Specific Methods*

Description

The functions estimate a fractal dimension of the given data. Each function uses a different method. Functions for boxcount, hallwood, variogram, madogram, rodogram, variation, incr1, genton, periodogram, wavelet and dctII methods are to be used on one-dimensional time series. The remaining functions (transect, isotropic, squareincr, and filter1) are to be used on two-dimensional data.

Usage

```
fd.estim.boxcount (data, plot.loglog = FALSE, nlags = "auto",
    shift.up=TRUE, plot.allpoints = FALSE, legend.type = 's',
    ..., debuglevel = 0)
fd.estim.hallwood (data, plot.loglog = FALSE, nlags = "auto",
    plot.allpoints = FALSE, legend.type = 's', ..., debuglevel = 0)
fd.estim.variogram (data, ...)
fd.estim.madogram (data, ...)
fd.estim.rodogram (data, ...)
fd.estim.variation (data, p.index = 1, ...)
fd.estim.incr1(data, p.index=2, ...)
fd.estim.genton (data, ...)
fd.estim.periodogram (data, plot.loglog = FALSE, nlags = "auto", ...)
fd.estim.wavelet (data, plot.loglog=FALSE, plot.allpoints = FALSE,
    filter = "haar", J1 = max(1,floor(log2(length(data))/3-1)),
    J0 = floor(log2(length(data))), legend.type = 's',
    ..., debuglevel = 0)
fd.estim.dctII (data, plot.loglog = FALSE, nlags = "auto", ...)

fd.estim.transect.var (data, p.index = 2, ...)
fd.estim.transect.incr1 (data, p.index = 2, ...)
fd.estim.isotropic (data, p.index = 2, direction = 'hvd+d-',
    plot.loglog = FALSE, nlags = "auto", plot.allpoints = FALSE,
    legend.type = 's', ..., debuglevel=0)
fd.estim.squareincr (data, p.index = 2,
    plot.loglog = FALSE, nlags = "auto", plot.allpoints = FALSE,
    legend.type = 's', ..., debuglevel=0)
fd.estim.filter1 (data, p.index = 2, direction = 'hvd+d-',
    plot.loglog = FALSE, nlags = "auto", plot.allpoints = FALSE,
    legend.type = 's', ..., debuglevel=0)
```

Arguments

`data`	For the first eleven functions `data` is a one-dimensional vector. For the last five functions `data` is a matrix.
`p.index`	Parameter p of the variation method (see below).
`direction`	For the 2d estimators, this argument specifies the direction of the estimation (see details below). It can be any combination of the characters 'h' (horizontal), 'v' (vertical), 'd+' (diagonal with positive gradient), and 'd-' (diagonal with negative gradient). These characters should be combined into one string.
`plot.loglog`	Logical value determining if the underlying log-log plots should be plotted.
`nlags`	Number of lags to be used in the estimation. Possible values are `"auto"`, `"all"` or a single number. If `nlags = "auto"`, each method sets the number of lags to the theoretically "best" value for that method. `"all"` means that all lags are included in the estimation.
`shift.up`	For each interval on the horizontal axis, it moves the boxes vertically up to the smallest data point of that interval. If it is `FALSE`, all boxes are on a regular grid.

`plot.allpoints`	Logical. If `FALSE`, only points that were considered in the regression are shown. Otherwise, all points of the log-log plot are shown in the graph and those considered in the regression are marked by filled circles. This argument is only used if `plot.loglog = TRUE`. Note that setting this argument to `TRUE` might (depending on the method) considerably increase the computation run-time.
`filter`	Argument passed to the `modwt` function of the **wavelets** package.
`J0`, `J1`	Parameters of the wavelet method controlling the number of frequencies used in the estimation.
`legend.type`	One of the characters 'f', 's', or 'n'. It controlls the amount of information in the legend of the log-log plot. If it is 'f' (full), values of fd and scale, including the raw values of the corresponding slope and intercept are shown. If it is 's' (short), only fd is shown. Value of 'n' (None) causes no legend being plotted. The argument is only used if `plot.loglog = TRUE`.
`...`	Arguments passed to the plotting function if `plot.loglog = TRUE`. For some functions, ... contain additional arguments, see Details.
`debuglevel`	Controls the amount of debugging messages. The functions produce messages on level 5.

Details

The methodology of these functions is based on the theory described in Gneiting et al (2010). Please refer to this paper for notation. Here we give only a few comments about the implementation.

Box-count estimator: The function `fd.estim.boxcount` determines the smallest possible value of m for which $n \leq 2^m$ is a power of 2. Only data points $x_1, \ldots, x_{n_{eff}}$ are considered for the estimation, where $n_{eff} = 2^{m-1} + 1$. The value of K can be given by the user through the argument `nlags`. If `nlags = "auto"`, box sizes ϵ_k for $k = j, j+1, \ldots, m-2$ are considered, where for all $i < j$ is $N(\epsilon_i) > \frac{n_{eff}}{5}$, i.e. the two largest box sizes and very small boxes are eliminated (corresponds to the Liebovitch and Toth modification).

If `shift.up=TRUE`, the algorithm shifts each vertical column of boxes up to the smallest data value in that column.

$N(\epsilon_k)$ for a particular box is increased if either a data point is contained in the box, or if a line connecting two neighboring data points crosses the box.

Hall-Wood estimator This estimator is a version of box-count that instead of number of boxes considers the area of boxes that cover the underlying curve. Hall and Wood (1993) recommend the use of $L = 2$ which the function `fd.estim.hallwood` uses if the arguments `nlags = "auto"`.

Variation, Variogram, Madogram, Rodogram, and Incr1 estimators: The `p.index` argument of `fd.estim.variation` and `fd.estim.incr1` is the power index p. The madogram, variogram, and rodogram, respectively, correspond to the Variation estimator with p equals 1, 2, and 1/2, respectively. The Incr1 estimator is like Variation but based on second order differences.

Any argument that can be passed to `fd.estim.hallwood` can be passed here as well. In addition, as in the Hall-Wood case, L is set to 2 for these estimators, if `nlags = "auto"`.

Genton robust estimator: This is a highly robust variogram estimator as proposed by Genton (1998). Given $U_i(d) = X_{i/n} - X_{(i-d)/n}$, define

$$\hat{V}(d) = \left[2.2191\{|U_i(d) - U_j(d)|; i < j\}_{(k)}\right]^2, \quad \text{where } k = \binom{\lfloor (n-d)/2 \rfloor + 1}{2}.$$

Thus, the estimator is derived from the k-th quantile of the $U_i(d)$ values. The $\hat{D}_k$ estimator is derived from the log-log plot of $\log(d)$ against $\log(\hat{V}(d))$. The implementation uses the `qn` function of the **pcaPP** package to compute $\hat{V}(d)$.

Here again, the number of lags is set to 2 if `nlags = "auto"` and any arguments of the `fd.estim.hallwood` are accepted here as well.

Periodogram estimator: The method is implemented as proposed by Chan et al. (1995) with notation from Gneiting et al (2010).

As Chan et al. (1995) recommend, we use $L = \lfloor \min(m/2, n^{2/3}) \rfloor$ if `nlags = "auto"`. Any arguments of the `fd.estim.hallwood` are also accepted here.

Wavelet estimator: This method uses J_0 vectors of wavelet coefficients which are obtained using the function `modwt` of the **wavelets** package. The choice of `J0` and `J1` determine the number of frequencies used in the estimation.

DCT-II estimator: If `nlags = "auto"`, we use $L = \lfloor \min(2m, 4n^{2/3}) \rfloor$. Any arguments of the `fd.estim.hallwood` are also accepted here.

The two-dimensional estimators are all based on the Variation method with the power index p (argument `p.index`) with the following alternatives:

Transect For every given direction, a variation estimate (or a variant that uses second differences) is found in each row (for horizontal direction) and/or column (for vertical direction). The resulting estimate is the median over the set of estimates. In the function `fd.estim.transect.var` the line transect estimates are based on first differences; In the function `fd.estim.transect.incr1` they are based on second differences.

This method does not support the feature of creating a log-log plot, since there are many log-log regressions from which the results are derived. The methods also accept arguments `direction`, `nlags` and `debuglevel`.

Isotropic Davies and Hall (1999) on page 12 define the isotropic empirical variogram. This is here implemented more generally using the variation estimator. If `nlags = "auto"`, the number of lags is set to either 3 if diagonal direction is used together with either horizontal or vertical direction or both. If only horizontal or/and vertical direction is used, the number of lags is set to 2.

Square-increment We use the square-increment estimator proposed in eqs. (4.2) through (4.7) of Chan and Wood (2000). Note that this method is equivalent to the Filter 3 approach of Zhu and Stein (2002) which is the way it is implemented in the package. The automatic setting of number of lags is done as for the Isotropic method.

Filter 1 Here, the Filter 1 approach of Zhu and Stein (2002) is implemented. Again, the automatic setting of number of lags is done as for the Isotropic method.

For all methods (but Transect), if the argument `plot.loglog` is `TRUE`, a graph with the log-log plot is shown, including the fitted regression line. Only points included in the regression are plotted, unless the argument `plot.allpoints` is set to `TRUE`. In such a case, points used for fitting the regression line are marked by filled circles.

For using multiple estimation methods via one function see `fd.estimate`.

Value

Each function returns an object of class `FractalDim` with elements:

`dim`	Here it is always 1.
`fd`, `scale`	Single value, namely the estimated fractal dimension and scale, respectively.
`methods`, `methods.coding`	Method name and code used for the estimation.
`window.size`, `step.size`	Size of the data.
`data.dim`	Dimension of the data used for the estimation. It is either one or two.
`loglog`	Object of class `FDloglog` used for the estimation.

Note

Function `fd.estimate` can be used as a wrapper for these functions.

Author(s)

Hana Sevcikova, Don Percival, Tilmann Gneiting

References

Chan, G., Hall, P., Poskitt, D. (1995) Periodogram-Based Estimators of Fractal Properties. Annals of Statistics **23** (5), 1684–1711.

Chan, G., Wood, A. (2000) Increment-based estimators of fractal dimension for two-dimensional surface data. Statistica Sinica **10**, 343–376.

Davies, S., Hall, P. (1999) Fractal analysis of surface roughness by using spatial data. Journal of the Royal Statistical Society Series B **61**, 3–37.

Genton, M. G. (1998) Highly robust variogram estimation. Mathematical Geology **30**, 213–221.

Gneiting, T., Sevcikova, H. and Percival, D. B. (2012). Estimators of fractal dimension: Assessing the smoothness of time series and spatial data. Statistical Science, 27(2), 247-277. (Version as technical report available at https://stat.uw.edu/sites/default/files/files/reports/2010/tr577.pdf)

Hall, P., Wood, A. (1993) On the Performance of Box-Counting Estimators of Fractal Dimension. Biometrika **80** (1), 246–252.

Zhu, Z., Stein, M. (2002) Parameter estimation for fractional Brownian surfaces. Statistica Sinica **12**, 863–883.

See Also

`fd.estimate`

Examples

```
if (requireNamespace("RandomFields", quietly = TRUE)) withAutoprint({
library(RandomFields)
# 1d time series
n <- 256
rf <- GaussRF(x = c(0,1, 1/n), model = "stable",
    grid = TRUE, gridtriple = TRUE,
```

```
    param = c(mean=0, variance=1, nugget=0, scale=1, kappa=1))
par(mfrow=c(4,2))
fd.estim.variogram (rf, nlags = 20, plot.loglog = TRUE)
fd.estim.variation (rf, nlags = 20, plot.loglog = TRUE)
fd.estim.variogram (rf,  nlags = 3, plot.loglog = TRUE,
    plot.allpoints = TRUE)
fd.estim.variation (rf, plot.loglog = TRUE, plot.allpoints = TRUE)
fd.estim.hallwood (rf, nlags = 10, plot.loglog = TRUE)
fd.estim.boxcount (rf, nlags = "all", plot.loglog = TRUE,
    plot.allpoints = TRUE)
fd.estim.periodogram (rf, plot.loglog = TRUE)
fd.estim.dctII (rf, plot.loglog = TRUE)

# 2d random fields
n <- 128
rf2d <- GaussRF(x = c(0,1, 1/n), y = c(0,1, 1/n), model = "stable",
    grid = TRUE, gridtriple = TRUE,
    param = c(mean=0, variance=1, nugget=0, scale=1, kappa=1))
par(mfrow=c(1,3))
fd.estim.isotropic (rf2d, p.index = 1, direction='hv',
                      plot.loglog = TRUE, plot.allpoints = TRUE)
fd.estim.squareincr (rf2d, p.index = 1, plot.loglog = TRUE, plot.allpoints = TRUE)
fd.estim.filter1 (rf2d, p.index = 1, plot.loglog = TRUE, plot.allpoints = TRUE)
})
```

`fd.estimate` *Estimating Fractal Dimensions of Time Series and Two-dimensional Data*

Description

The functions compute a set of fractal dimensions D for time series and two-dimensional data via various methods using a sliding window technique. There is one D computed for each method and for each sliding window of a given size that is moved along the data.

Usage

```
## S3 method for class 'numeric'
 fd.estimate(data, methods = "madogram", window.size = length(data),
    step.size = window.size, trim = TRUE, keep.data = FALSE,
    keep.loglog = FALSE, parallel = FALSE, nr.nodes = NULL,
    debuglevel = 0, ...)

## S3 method for class 'matrix'
 fd.estimate(data, methods = "transect.var", window.size = ncol(data),
    step.size = window.size, trim = TRUE, keep.data = FALSE,
    keep.loglog = FALSE, parallel = FALSE, nr.nodes = NULL,
    debuglevel = 0, ...)
```

Arguments

data	Vector, matrix or data frame.
methods	Vector of character strings specifying methods for which D is estimated. Possible values for one-dimensional data are "variogram", "madogram", "rodogram", "variation", "incr1", "boxcount", "hallwood", "periodogram", "wavelet", "dctII", and "genton". For matrix or data frame the function accepts methods "transect.var", "transect.incr1", "isotropic", "squareincr", and "filter1" (see fd.get.available.methods). Alternatively, it can be a list of lists where each list item contains an entry "name" being the method name and entries corresponding to arguments passed to the specific methods (see Example and fd.estim.method for details.)
window.size	Size (in number of data points) of the sliding window. It should be between 2 and length of data.
step.size	Number of data points by which the sliding window is moved.
trim	Logical. If TRUE, the estimates are trimmed into the theoretically permissible interval, i.e. between 1 and 2 in one-dimensional case and between 2 and 3 in two-dimensional case.
keep.data	Logical. If TRUE, the data are kept in the resulting object.
keep.loglog	Logical. If TRUE, the resulting object contains a list with objects of class FDloglog used for the estimation in each iteration and for each method.
parallel	Logical determining if the process should run in parallel. If TRUE, the package **snowFT** is required. In such a case, all local library paths must be included in the environment variable R_LIBS. In the one-dimensional case, the granularity of the process is given by the number of sliding windows. In the two-dimensional case, the number of spawn processes is equal to the number of sliding windows in the vertical direction.
nr.nodes	Number of nodes on which the computation should be processed if parallel is TRUE.
debuglevel	Controls the amount of debugging messages. The functions produce messages on levels 1 - 4.
...	Arguments passed to lower level functions (defined in fd.estim.method).

Details

In case of one-dimensional time series, the function initiates a sliding window of the given size at the beginning of the time series. The window is moved along the data by the given step size. If parallel is TRUE, computation on each window happens in parallel. In the two-dimensional case, the window is initiated in the top left corner of the data matrix and moved horizontally by the given step size, as well as vertically by the same step size. If the process is running in parallel, processing each row is done in parallel. In both cases, in each iteration estimates of fractal dimension for data within the sliding window are computed using the given estimation methods.

Note that the estimation results are NA for any sliding window that contains NA values.

Arguments that are to be passed to specific methods can be given either directly, if they applies to all given methods. Or, they can be given as a list via the methods argument: There is one list per method that must contain the entry "name" being the method name. Remaining entries in the list corespond to one argument each (see Example below).

Value

An object of class `FractalDim` which consists of the following components:

`dim`	Dimension of the resulting arrays `fd` and `scale` (see below). In the one-dimensional case, possible values are 1 and 2. `dim = 1` means that there has been only one iteration and there is one element in the above arrays per each method used. If `dim = 2`, rows correspond to iterations and columns correspond to methods. Estimation on two-dimensional data can in addition result in `dim = 3`, in which case the first and second dimensions correspond to vertical and horizontal iterations, respectively, and the third dimension corresponds to methods.
`fd`	A `dim`-dimensional array of fractal dimensions.
`scale`	A `dim`-dimensional array of scales, derived from the intercept of the log-log plots on which `fd` were computed. Values are transformed to the scale of the original data.
`methods`	Vector of methods given in the `methods` argument. The order of the elements corresponds to the order of estimates in the "method"-dimension of the above `dim`-dimensional arrays.
`methods.coding`	Vector of internal coding of `methods`. The order of the elements corresponds to the order in `methods`.
`data`	Value of the argument `data`, if `keep.data = TRUE`, otherwise `NULL`.
`data.dim`	Dimension of `data`.
`window.size`	Size of the actual sliding window used in the computation.
`step.size`	Step size by which the sliding window was moved in the computation.
`loglog`	If `keep.loglog=TRUE`, this is a list containing for each iteration a lists of `FDloglog` objects used in the estimation, one per method. The numbering of the methods corresponds to the method order in `methods`.

See Also

`fd.estim.method`, `fd.get.available.methods`, `FDloglog`, `fd.get`

Examples

```
## Not run:
library(RandomFields)
n <- 10000
# generate a time series
rf <- GaussRF(x = c(0, 1, 1/n), model = "stable",
       grid = TRUE, gridtriple = TRUE,
       param = c(mean=0, variance=1, nugget=0, scale=100, kappa=1))

# Plots for two sliding windows of each of the four methods below.
# Argument nlags is common to all methods;
# the 'variation' method has in addition argument p.index
par(mfrow=c(2,4)) # one row per window
fd <- fd.estimate(rf,
       methods = list(list(name="variation", p.index=0.5),
```

```
        "variogram", "hallwood", "boxcount"),
       window.size = 5000, step.size = 5000, plot.loglog = TRUE, nlags = 10)

# 2d random fields
n <- 200
rf2d <- GaussRF(x = c(0,1, 1/n), y = c(0,1, 1/n), model = "stable",
         grid = TRUE, gridtriple = TRUE,
         param = c(mean=0, variance=1, nugget=0, scale=1, kappa=1))
par(mfrow=c(2,2))
# plots for 4 sliding windows (2 horizontal, 2 vertical)
fd2d <- fd.estimate(rf2d, methods="filter1",
         window.size = 100, step.size=100, plot.loglog = TRUE)

## End(Not run)
```

`fd.get` *Access Method for Objects of Class FractalDim*

Description

For given method it returns the corresponding estimates.

Usage

```
fd.get(fractaldim, method)
```

Arguments

`fractaldim`	object of class `FractalDim` as defined in `fd.estimate`.
`method`	character string specifying the method. For 1-d estimators, possible values are "`variogram`", "`madogram`", "`rodogram`", "`variation`", "`incr1`", "`boxcount`", "`hallwood`", "`periodogram`", "`wavelet`", "`dctII`", and "`genton`". For 2-d estimators, possible values are "`transect.var`", "`transect.incr1`", "`isotropic`", "`squareincr`", and "`filter1`" (see `fd.get.available.methods`).

Value

A `FractalDim` object. The original `fractaldim` arrays `fd` and `scale` are reduced in the last dimension into only one method, namely the given `method`.

See Also

`fd.estimate`, `fd.get.available.methods`

Examples

```
## Not run:
library(RandomFields)
x <- seq(0, 10000)
# generate a random field
truealpha <- 1.5
rf <- GaussRF(x = x, model = "stable", grid = TRUE,
    param = c(mean=0, variance=1, nugget=0, scale=100,
        alpha=truealpha))

#compute fractal dimension using various methods
methods <- c("madogram", "variogram", "hallwood", "boxcount",
                        "periodogram","dctII", "wavelet")
fdts <- fd.estimate (rf, methods = methods, window.size = 500,
    step.size = 100, nlags = 10, trim = FALSE, debuglevel = 3)

# plot the variation
cols <- rainbow(length(methods))
plot(ts(fd.get (fdts, methods[1])$fd),ylim=c(min(fdts$fd), max(fdts$fd)),
             ylab="fd", col=cols[1])
for (imeth in 2:length(methods))
    lines(ts(fd.get (fdts, methods[imeth])$fd), col=cols[imeth])
legend('topleft', legend=methods, col=cols, lwd=1)
abline(h=2-truealpha/2)

## End(Not run)
```

`fd.get.available.methods` *Available Estimation Methods*

Description

The function returns a list of estimation methods that can be used in `fd.estimate` and other functions of the package.

Usage

```
fd.get.available.methods(dim = 1)
```

Arguments

`dim`	Dimension of data for which the estimation methods should be obtained.

Value

A list of the available methods. Their order number in the list corresponds to the internal codes of the methods.

See Also

fd.estimate, fd.get

`get.rawFD.from.regression` *Obtaining Regression Object*

Description

Obtaining and summarizing result of a linear regression of the log-log plot, object of class `FDloglog`.

Usage

```
get.rawFD.from.regression(x, y, leaveout = 0)

## S3 method for class 'FDloglog'
summary(object, ...)
```

Arguments

`x`	Values of the x-axis.
`y`	Values of the y-axis.
`leaveout`	Number of points (from the beginning of the arrays) to leave out of the regression.
`object`	Object of class `FDloglog`.
`...`	Not used.

Value

Function `get.rawFD.from.regression` returns an object of class `FDloglog` with the following components:

`alpha, intercept`	Slope and intercept of the regression.
`x, y`	x and y values, including the `leaveout`.
`n`	Length of `x`.
`lsq`	The least squared of the regression.

Author(s)

Hana Sevcikova

See Also

fd.estimate, fd.estim.method

summary.FractalDim *Summary for an Object of Class FractalDim*

Description

The function prints summary of estimates in a FractalDim object.

Usage

```
## S3 method for class 'FractalDim'
 summary(object, ...)
```

Arguments

object	An object of class FractalDim.
...	Not used.

Details

The function prints information about the grid on which the estimates were obtained. For each method it shows the mean and standard deviation of data in each of the two components (fd and scale).

See Also

fd.estimate

Index