Package: kzs (via r-universe)

August 31, 2024

Type Package

Title Kolmogorov-Zurbenko Spatial Smoothing and Applications

Version 1.4

Date 2008-10-24

Author Derek Cyr <cyr.derek@gmail.com> and Igor Zurbenko <igorg.zurbenko@gmail.com>.

Maintainer Derek Cyr <cyr.derek@gmail.com>

Depends R (>= 2.8.0), graphics, lattice, stats

Description A spatial smoothing algorithm based on convolutions of finite rectangular kernels that provides sharp resolution in the presence of high levels of noise.

License GPL (>= 2)

Contents

kzs 1
kzs.2d 4
kzs.3d_data 6
kzs.md 7
kzs.params 9

Index **11**

`kzs` *Kolmogorov-Zurbenko Spline*

Description

This is a one-dimensional iterative smoothing algorithm based on convolutions of rectangular kernels.

Usage

```
kzs(y, x, smooth, scale, k = 1, edges = TRUE, plot = TRUE)
```

Arguments

y	a one-dimensional vector of real values representing the response variable to be smoothed.
x	a one-dimensional vector of real values representing the input variable.
smooth	a real number defining the width of the smoothing window, i.e., the width of the rectangular kernel.
scale	for an irregularly spaced x, scale is a positive real number that will define a uniform scale along x.
k	an integer specifying the number of iterations kzs will execute; k may also be interpreted as the order of smoothness (as a polynomial of degree k-1). By default, k = 1.
edges	a logical indicating whether or not to display the outcome data beyond the initial range of x. By default, edges = TRUE.
plot	a logical indicating whether or not to produce a plot of the kzs outcome. This is TRUE by default.

Details

The relation between variables Y and X as a function of a current value of X = x [namely, Y(x)] is often desired as a result of practical research. Usually we search for some simple function, Y(x), when given a data set of pairs (Xi, Yi). When plotted, these pairs frequently resemble a noisy plot, and thus Y(x) is desired to be a smooth outcome that captures patterns or long-term trends in the original data, while suppressing the noise. The kzs function is based on convolutions of the rectangular kernel, which is equilvalent to repeated applications of a moving average. According to the Central Limit Theorem, repeated convolutions with rectangular kernels will converge to the Gaussian kernel; the resulting kernel will have finite support equal to smooth*k, which will result in a smooth outcome with diminished noise leakage, which is a feature that the standard Gaussian kernel does not exhibit.

Value

a two-column data frame of paired values (xk, yk):

xk	x values in increments of scale
yk	smoothed response values resulting from k iterations of kzs

Note

Data set (Xi, Yi) must be provided, usually as some observations that occur at certain times; kzs is designed for the general situation, including time series data. In many applications where the input variable, x, can be time, kzs is resolving the problem of missing values in time series or irregularly observed values in longitudinal data analysis.

kzs may take time to completely run depending on the size of the data set used and the number of iterations specified.

For more information on the restrictions imposed on delta and d, consult kzs.params.

Author(s)

Derek Cyr <cyr.derek@gmail.com> and Igor Zurbenko <igorg.zurbenko@gmail.com>

References

Zurbenko, I.G. (1986). *The Spectral Analysis of Time Series*. North Holland Series in Statistics and Probability, Elsevier Science, Amsterdam.

See Also

kzs.params, kzs.2d, kzs.md

Examples

```
# Total time t
t <- seq(from = -round(400*pi), to = round(400*pi), by = .25)

# Construct the signal over time
ts <- 0.5*sin(sqrt((2*pi*abs(t))/200))
signal <- ifelse(t < 0, -ts, ts)

# Bury the signal in noise [randomly, from N(0, 1)]
et <- rnorm(length(t), mean = 0, sd = 1)
yt <- et + signal

# Data frame of (t, yt)
pts <- data.frame(cbind(t, yt))


### EXAMPLE 1 - Apply kzs to the signal buried in noise

# Plot of the true signal
plot(signal ~ t, xlab = "t", ylab = "Signal", main = "True Signal",
type = "l")

# Plot of signal + noise
plot(yt ~ t, ylab = "yt", main = "Signal buried in noise", type = "p")

# Apply 3 iterations of kzs
kzs(y = pts[,2], x = pts[,1], smooth = 80, scale = .2, k = 3, edges = TRUE,
plot = TRUE)
lines(signal ~ t, col = "red")
title(main = "kzs(smooth = 80, scale = .2, k = 3, edges = TRUE)")
legend("topright", c("True signal","kzs estimate"), cex = 0.8,
col = c("red", "black"), lty = 1:1, lwd = 2, bty = "n")
```

```
### EXAMPLE 2 - Irregularly observed data over time

# Cancel a random 20 percent of (t, yt) leaving irregularly observed time points
obs <- seq(1:length(t))
t20 <- sample(obs, size = length(obs)/5)
pts20 <- pts[-t20,]

# Plot of (t,yt) with 20 percent of the data removed
plot(pts20$yt ~ pts20$t, main = "Signal buried in noise\n20 percent of
(t, yt) deleted", xlab = "t", ylab = "yt", type = "p")

# Apply 3 iterations of kzs
kzs(y = pts20[,2], x = pts20[,1], smooth = 80, scale = .2, k = 3, edges = TRUE,
plot = TRUE)
lines(signal ~ t, col = "red")
title(main = "kzs(smooth = 80, scale = .2, k = 3, edges = TRUE)")
legend("topright", c("True signal","kzs estimate"), cex = 0.8,
col = c("red", "black"), lty = 1:1, lwd = 2, bty = "n")
```

---

kzs.2d *Spatial Kolmogorov-Zurbenko Spline*

Description

The kzs.2d function is a spatial extension of the kzs function for two input variables.

Usage

```
kzs.2d(y, x, smooth, scale, k = 1, edges = TRUE, plot = TRUE)
```

Arguments

y	a one-dimensional vector of real values representing the response variable to be smoothed.
x	a two-dimensional matrix of real values containing the input variables X = (X1, X2). Each column represents an input variable.
smooth	a vector of size two that defines the width of the smoothing window along each input variable.
scale	a vector of size two in which each element will define a uniformly spaced scale along its respective input variable.
k	an integer specifying the number of iterations kzs.2d will execute. By default, k = 1.
edges	a logical indicating whether or not to display the outcome data beyond the rectangular range of the two input variables. By default, edges = TRUE.
plot	a logical indicating whether or not to produce a 3-dimensional plot of the kzs.2d outcome. By default, this argument is set to TRUE.

Details

The details for this function are nearly identical to that of kzs, except now extended to three dimensional space. The only difference is that the kzs.2d function averages all y that are contained within a rectangular window made up of sides smooth[1] and smooth[2].

Value

a three column data frame of the form (x1, x2, yk):

x1	the x1 coordinates of a two-dimensional grid.
x2	the x2 coordinates of a two-dimensional grid.
yk	the smoothed response values resulting from k iterations of kzs.2d.

Note

Data set (Y, X1, X2) must be provided, usually as 3-dimensional observations that occur in time or space; kzs.2d is designed for the general situation, including time series data. In many applications where an input variable can be time, kzs.2d can resolve the problem of missing values in time series or or irregularly observed values in Geographical Information Systems (GIS) data analysis. The name of this function, kzs.2d, simply means that there are two input variables required for use.

The graphical output of kzs.2d is a result of the wireframe() function within the **lattice** package.

Author(s)

Derek Cyr <cyr.derek@gmail.com> and Igor Zurbenko <igorg.zurbenko@gmail.com>

See Also

kzs; For more on the parameter restrictions, see kzs.params

Examples

```
# EXAMPLE - Estimating the Sinc function in the interval (-3pi, 3pi)
#           Load the LATTICE package


# Gridded data for X = (x1, x2) input variables
x1 <- seq(-3*pi, 3*pi, length = 60)
x2 <- x1
df <- expand.grid(x1 = x1, x2 = x2)

# Apply the Sinc function to the (x1, x2) coordinates
df$z <- sin(sqrt(df$x1^2 + df$x2^2)) / sqrt(df$x1^2 + df$x2^2)
df$z[is.na(df$z)] <- 1

# Any point outside the circle of radius 3pi is set to 0. This provides
# a better picture of the outcome solely for the purposes of this example.
dst <- sqrt((df$x1 - 0)^2 + (df$x2 - 0)^2)
df$dist <- dst
```

```
df$z[df$dist > 3*pi] <- 0

# Add noise to distort the signal
ez <- rnorm(length(df$z), mean = 0, sd = 1) * 1/4
df$zn <- ez + df$z

### (1) 3D plot of the signal to be estimated by kzs.2d()
wireframe(z ~ x1 * x2, df, main = "Signal to be estimated", drape = TRUE,
colorkey = TRUE, scales = list(arrows = FALSE))

### (2) 3D plot of the signal buried in noise
wireframe(zn ~ x1 * x2, df, main = "Signal buried in noise", drape = TRUE,
colorkey = TRUE, scales = list(arrows = FALSE))

### (3) Execute kzs.2d()
# kzs.2d() may take time to run; k = 1 iteration is used here, but k = 2
# will provide a smoother outcome.
sw <- c(1, 1)
sc <- c(0.2, 0.2)
kzs.2d(y = df[,5], x = df[,1:2], smooth = sw, scale = sc, k = 1, edges = TRUE,
plot = TRUE)
```

`kzs.3d_data` *4-dimensional KZS Output*

Description

This data set contains the output from a KZS operation consisting of 3 input variables, X = (X1, X2, X3), and the single outcome variable Y, which is buried in noise. See the 'Details' for more information.

Usage

```
data(kzs.3d_data)
```

Format

A data frame with 9025 observations on 52 variables. The first two variables are the coordinates of a two-dimensional grid (X1, X2). The remaining 50 variables are KZS output. See more details below.

Details

This data set is based on the example documented in `kzs.2d`. A 2D grid of points, (X1, X2), is constructed over the range [-1.5pi, 1.5pi] and acts as two input variables. The third input variable is time, and has values, 1, 2, ..., 50. For each of the 50 time points, there is a corresponding amplitude, that ranges from 0 to 1. For each unique amplitude, the outcome variable, Y, is calculated by applying the Sinc function to the `(x1, x2)` grid over each of the 50 time points. This process results in 50 columns of data, 1 for each time point and amplitude. As stated in the `kzs.md` documentation, KZS

is a linear operation, and thus is commutative (that is, we can change the order of the operations, without changing the end result). For example, `kzs.3d = kzs.1d + kzs.2d = kzs.2d + kzs.1d`. This property of KZS was used to receive the 4-dimensional result. Using the data described above, `kzs.2d` was first applied to each of the 50 columns of input using `k = 2` iterations, `smooth` = (1.5, 1.5) and `scale` = (0.1, 0.1), which were chosen arbitrarily. Using the resulting data from the `kzs.2d` operation, `kzs.1d` was applied across time. The result of this operation is 4-dimensional data, which can be visualized as a 2-dimensional map with color (blue for low amplitudes near 0 and red for high amplitudes close to 1). Incorporating time, this result can be visualized as a "movie" of the 50 2-dimensional images, where the amplitude (color) is changing from 0 to 1 (blue to red).

This data set has been included in this package due to the significant amount of computer time it took to run. Including this process as an example would not be efficient. Using this data set, the example in the `kzs.md` documentation is constructed.

`kzs.md` *Spatial Kolmogorov-Zurbenko Spline*

Description

The `kzs.md` function is an extension of the `kzs` function to *d* input variables.

Usage

```
kzs.md(y, x, smooth, scale, k = 1, edges = TRUE)
```

Arguments

`y`	a one-dimensional vector of real values representing the response variable to be smoothed.
`x`	a *d*-dimensional matrix of real values containing the input variables X = (X1, X2, ..., X*d*); i.e., each column of the matrix is an input variable.
`smooth`	a real-valued vector of size *d* in which each element defines the range of smoothing for each corresponding variable in `x`.
`scale`	a real-valued vector of size *d* in which each element defines a uniform scale along its corresponding input variable.
`k`	an integer specifying the number of iterations `kzs.md` will execute. By default, `k = 1`.
`edges`	a logical indicating whether or not to display the outcome data beyond the ranges of the *d* input variables in `x`. By default, `edges = TRUE`.

Details

The details for this function are nearly identical to that of `kzs`, except now extended to *d*-dimensional space.

Value

a (d+1)-column data frame of the form `(x1, x2, ...,xd, yk)`. See `kzs.2d` for the general interpretations of these results.

Note

In many applications where input variables can be space, `kzs.md` can resolve the problem of missing values in time series or or irregularly observed valuesin Geographical Information Systems (GIS) data analysis. For these applications, `scale` is especially advantageous because it can create a uniform space over a geographic region to which the algorithm will be applied. Additionally, `kzs.md` can be recommended as a diagnostic tool before applying multiple linear regression analysis due to its capability of displaying nonlinearities of the outcome over the input variables.

There is no graphical output for this function; for two input variables, `kzs.2d` will produce a 3-dimensional plot. For three input variables, a 4-dimensional movie can be constructed over time.

In general, `kzs`, `kzs.2d` and `kzs.md` are all linear operations, and linear operations are commutative. Thus, for example, the outcome of a `kzs.2d` operation is equivalent to `kzs.1d + kzs.1d`; likewise, the outcome of a `kzs.3d` operation is equivalent to a `kzs.2d + kzs.1d`, etc...

Author(s)

Derek Cyr <cyr.derek@gmail.com> and Igor Zurbenko <igorg.zurbenko@gmail.com>

See Also

`kzs`; For more on the parameter restrictions, see `kzs.params`

Examples

```
# This example is an extension of the example documented in kzs.2d. We make
# use of the Sinc function to filter a signal buried in noise over 3-dimensional
# input variables. See the "Details" section of the "kzs.3d_data" data frame
# documentation for specific details.
require(lattice)

# Gridded data for X = (X1, X2) input variables
x1 <- seq(-1.5*pi, 1.5*pi, length = 50)
x2 <- x1
df <- expand.grid(x1 = x1, x2 = x2)

# Time dimension
time <- 1:50

# Change the amplitude of the original function to change from 0 to 1 along time
amplitude <- sort(round(seq(0.02, 1, 0.02), digits = 2))

# Store the time and amplitude together in a data frame
t_amp <- data.frame(cbind(time, amplitude))
```

```
# Create the data set of Sinc function outcomes for each amplitude
sinc <- array(0, dim = c(nrow(df), length(amplitude)))
for (i in 1:length(amplitude)) {
 sinc[,i] <- round(amplitude[i]*sin(sqrt(df$x1^2 + df$x2^2)) /
       sqrt(df$x1^2 + df$x2^2))
 sinc[,i][is.na(sinc[,i])] <- amplitude[i]
}

# Add noise to distort the signal
for (j in 1:ncol(sinc)) {
 ez <- rnorm(nrow(sinc), mean = 0, sd = 1)
 sinc[,j] <- sinc[,j] + ez
}

# Change to a data frame and add the gridded input data
kzs.2d_data <- as.data.frame(cbind(df, sinc))

### Movie of the signal buried in noise
grayscale = colorRampPalette(c("white", "gray", "black"))
for (u in 1:50) {
 plot(levelplot(kzs.2d_data[,u+2] ~ x1*x2, kzs.2d_data,
  col.regions = grayscale, colorkey = FALSE))
}

### Movie of KZS 4-dimensional KZS outcome
data(kzs.3d_data)
bluered = colorRampPalette(c("blue", "cyan2", "green",
        "yellow", "red", "firebrick"), space = "rgb")
for (j in 1:50) {
 plot(levelplot(kzs.3d_data[,j+2] ~ x1*x2, kzs.3d_data,
           at = do.breaks(c(-0.3, 1.0), 100), col.regions = bluered))
}
```

`kzs.params` *Restrictions for KZS Parameters*

Description

For a *d*-dimensional vector of input variables, this function will calculate the values by which the parameters `smooth` and `scale` are bounded by.

Usage

```
kzs.params(x, dimension)
```

Arguments

`x`	a matrix or data frame containing the input variable(s) that is to be used in `kzs`, `kzs.2d`, or `kzs.md`.
`dimension`	an integer specifying the dimensionality of `x`; i.e, the number of columns in `x`.

Details

The compilation of functions within the **kzs** package requires the specification of two parameters: the first is `smooth`, the range of smoothing along each variable in `x`; the second is `scale`, a scale reading of each corresponding input variable in `x`. Each parameter is subject to two restrictions; `smooth[i]` and `scale[i]` must be positive real numbers; `scale[i]` must be less than or equal to the difference of sorted, consecutive `x[,i]` values and `smooth[i]` must be much less than the difference of the maximum and minimum values for its corresponding input variable, `x[,i]`. For each input variable in `x`, there must be a corresponding `smooth` and `scale`. This function was developed to be used prior to any of the functions within **kzs** in order to increase the efficiency of use.

Author(s)

Derek Cyr <cyr.derek@gmail.com> and Igor Zurbenko <igorg.zurbenko@gmail.com>

Examples

```
# Generate 3 random sequences of numbers that would act as the input data set
x1 <- rnorm(100, 3, 6)
x2 <- rnorm(100, 4, 5)
x3 <- runif(100, 0, 1)

# A matrix or a data frame will work
mat <- matrix(c(x1, x2, x3), nrow = 100, ncol = 3)

# Dimensionality is 3 since there are 3 input variables
kzs.params(x = mat, dimension = 3)
```

Index

∗ **datasets**
kzs.3d_data, 6
∗ **nonparametric**
kzs, 1
kzs.2d, 4
kzs.md, 7
kzs.params, 9
∗ **smooth**
kzs, 1
kzs.2d, 4
kzs.md, 7
kzs.params, 9
∗ **ts**
kzs, 1
kzs.2d, 4
kzs.md, 7

kzs, 1, *5*, *7, 8*
kzs.2d, *3*, 4, *8*
kzs.3d_data, 6
kzs.md, *3*, 7
kzs.params, *3*, *5*, *8*, 9