Package ‘kzft’

February 20, 2015

Version 0.17

Date 2007/09/19

Title Kolmogorov-Zurbenko Fourier Transform and Applications

Author Wei Yang <peterwyang@gmail.com> and Igor Zurbenko
<igorg.zurbenko@gmail.com>

Maintainer Wei Yang <peterwyang@gmail.com>

Depends polynom

Suggests graphics

Description A colletion of functions to implement Kolmogorov-Zurbenko Fourier transform based periodograms and smoothing methods

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-10-29 08:59:04

NeedsCompilation no

R topics documented:

kzft 2
kzp 4
kztp 6
kzw 9

Index **11**

kzft *Kolmogorov-Zurbenko Fourier Transform*

Description

Kolmogorov-Zurbenko Fourier transform is an iterative moving average of the regular Fourier transform.

Usage

```
kzft(x, m, k, p=1, n=1)
coeff.kzft(m,k)
transfun.kzft(m,k,lamda=seq(-0.5,0.5,by=0.01),omega=0)
```

Arguments

`x`	The raw time series
`m`	The length of the window size for a regular Fourier transform
`k`	The number of iterations for the KZFT
`p`	The distance between two successive intervals as a percentage of the total length of the time series
`n`	The sampling frequency rate as a multiplication of the Fourier frequencies
`lamda`	A sequence of frequencies at which the transfer function is calculated
`omega`	A kernal frequency at which the KZFT is applied

Details

Kolmogorov-Zurbenko Fourier transform (KZFT) is a special case of Fourier transform which is used to calculate the periodogram. It was first introduced in Zurbenko (1986). It is an iterative moving average of the regular Fourier transform. A KZFT with parameters m and k (KZFT(m,k)) is defined as k times iteration of a regular Fourier transform over the data with a length of m. Regular Fourier transform has an energy leakage around the Fourier frequencies. KZFT may overcome the leakage by k degrees less where k is the number of iterations in the KZFT. Because of this, KZFT shows strong resistance to noises and can be used to generate high resolution spectrum. From the point view of mean square error, Zurbenko (1986) also showed that KZFT has the closest to the optimal mean square error. For the detailed definition of KZFT, please refer to Zurbenko and Porter (1998) and Neagu and Zurbenko (2002). For information of spectral analysis, please refer to Shumway and Stoffer (2006).

In this package, the frequency unit is cycles per unit time, where the time unit is the time interval of the discrete observations in the raw time series. For example, a signal with a frequency of 0.1 corresponds to a period of 10 time units. If the raw time series is recorded as one obeservation per day, then the period for this signal is 10 days.

References

I. G. Zurbenko, The spectral Analysis of Time Series. North-Holland, 1986.

I. G. Zurbenko, P. S. Porter, Construction of high-resolution wavelets, Signal Processing 65: 315-327, 1998.

R. Neagu, I. G. Zurbenko, Tracking and separating non-stationary multi-component chirp signals, J. Franklin Inst., 499-520, 2002.

R. H. Shumway, D. S. Stoffer, Time Series Analysis and Its Applications: With R Examples, Springer, 2006.

See Also

kzp, kztp, kzw.

Examples

```
# example 1
# coefficient of the kzft(201,5)

aa<-coeff.kzft(201,5);
tt<-seq(1:1001)-501;
zz<-cos(2*pi*0.025*tt);
plot(zz*aa,type="l",xlab="Time unit", ylab="The coefficient",
    main="Coefficients of the kzft");
lines(aa);
lines(-1*aa);

# example 2
# transfer function of the kzft(201,5) at frequency 0.025

lamda<-seq(-0.1,0.1,by=0.001)
tf1<-transfun.kzft(201,1,lamda,0.025)
tf2<-transfun.kzft(201,5,lamda,0.025)
matplot(lamda,cbind(log(tf1),log(tf2)),type="l",ylim=c(-15,0),ylab="Natural log transformation of the coefficie
        main="Transfer function of kzft(201,5) at frequency 0.025")

#example 3
#The example shows the KZFT reproduces nearly perfectly the noise-free pattern of a sin wave with a constant frequ

# raw time series with frequency of 0.01
t<-1:1000
x<-cos(2*pi*(10/1000)*t)

#kzft(200,1) calcualted every one time unit at freuquency 0.01
kzft.x1<-kzft(x,200,1,0.005)
x1<-2*Re(kzft.x1$tf[,2])

#kzft(200,2) calcualted every one time unit at freuquency 0.01
kzft.x2<-kzft(x,200,2,0.005/2)
x2<-2*Re(kzft.x2$tf[,2])
```

```
#kzft(200,3) calcualted every one time unit at freuquency 0.01
kzft.x3<-kzft(x,200,3,0.005/3)
x3<-2*Re(kzft.x3$tf[,2])

par(mfrow=c(2,2))
plot(x,type="l",main="Original time series",xlab="Time unit", ylab="Amplitude")
plot(x1,type="l",main="kzft(200,1)",xlab="Time unit", ylab="Amplitude")
plot(x2,type="l",main="kzft(200,2)",xlab="Time unit", ylab="Amplitude")
plot(x3,type="l",main="kzft(200,3)",xlab="Time unit", ylab="Amplitude")

#example 4
#The example shows the KZFT reproduces nearly perfectly the noise-free pattern of a cos wave with a constant frequ

# raw time series with frequency of 0.01
t<-1:1000
x<-sin(2*pi*(10/1000)*t)

#kzft(200,1) calcualted every one time unit at freuquency 0.01
kzft.x1<-kzft(x,200,1,0.005)
x1<-2*Re(kzft.x1$tf[,2])

#kzft(200,2) calcualted every one time unit at freuquency 0.01
kzft.x2<-kzft(x,200,2,0.005/2)
x2<-2*Re(kzft.x2$tf[,2])

#kzft(200,3) calcualted every one time unit at freuquency 0.01
kzft.x3<-kzft(x,200,3,0.005/3)
x3<-2*Re(kzft.x3$tf[,2])

par(mfrow=c(2,2))
plot(x,type="l",main="Original time series",xlab="Time unit", ylab="Amplitude")
plot(x1,type="l",main="kzft(200,1)",xlab="Time unit", ylab="Amplitude")
plot(x2,type="l",main="kzft(200,2)",xlab="Time unit", ylab="Amplitude")
plot(x3,type="l",main="kzft(200,3)",xlab="Time unit", ylab="Amplitude")
```

kzp *Kolmogorov-Zurbenko Periodogram and Smoothing Methods*

Description

Kolmogorov-Zurbenko periodogram and smoothing using either DiRienzo-Zurbenko (DZ) or Neagu-Zurbenko (NZ) smoothing methods.

Usage

```
kzp(x, m, k, p=1, n=1)
nonlinearity.kzp(pg, K=length(pg))
variation.kzp(pg, K=length(pg))
smooth.kzp(pg,c,K=length(pg),method = "DZ")
```

Arguments

`x`	The raw time series
`m`	The length of the window size for a regular Fourier transform
`k`	The number of iterations for the KZFT
`p`	The distance between two successive intervals as a percentage of the total length of the time series
`n`	The sampling frequency rate as a multiplication of the Fourier frequencies
`pg`	The raw periodogram of a time series
`K`	Half length of the maximum bandwidth of the spectral window being allowed in the algorithms
`c`	A prespecified percentage of total nonlinearity/variation
`method`	Smooth methods: DZ or NZ

Details

Kolmogorov-Zurbenko periodogram is calculated based on the Kolmogorov-Zurbenko Fourier transform. Both DZ and NZ smoothing methods are adaptive which allow the bandwidth of the spectral window vary according to the smoothness of the underlying spectral density. In DZ algorithm, the squared variation of the periodogram is used to evaulate the smoothness of the spectrum. When smoothing the periodogram, the bandwith of the spectral window is extended until the squared variation within the window reaches a prespecified percentage of total variation of the periodogram. In NZ algorithm, the only difference is to use a measure of nonlinearity to evaluate the smoothness of the spectrum. In the situation when the variation is very small at certain frequencies, the bandwidth of the spectral window could be extended very large in order to achieve the prespecified percentage of the total variation. However, this little improvement of the smoothness of the spectrum will cause a considerably increased amount of computation labor. In the algorithms, we set a limit of the maximum bandwidth being allowed to avoid this situation. For details, please refer to DiRienzo and Zurbenko (1998) and Neagu and Zurbenko (2003).

References

I. G. Zurbenko, 1986: The spectral Analysis of Time Series. North-Holland, 248 pp.

I. G. Zurbenko, P. S. Porter, Construction of high-resolution wavelets, Signal Processing 65: 315-327, 1998.

A. G. DiRienzo, I. G. Zurbenko, Semi-adaptive nonparametric spectral estimation, Journal of Computational and Graphical Statistics 8(1): 41-59, 1998.

R. Neagu, I. G. Zurbenko, Algorithm for adaptively smoothing the log-periodgram, Journal of the Franklin Institute 340: 103-123, 2003.

See Also

`kzft`, `kztp`, `kzw`.

Examples

```
#example 1
#show the effect of smoothing methods

N<-480
x<-rep(0,N)

x[1:4]<-rnorm(4)

for ( t in 5:N )
{
x[t]<-2.76*x[t-1]-3.81*x[t-2]+2.65*x[t-3]-0.92*x[t-4]+rnorm(1)
}

for ( t in 1:N )
{
x[t]<-50*cos( 2*pi*(10/N)*t)+40*cos( 2*pi*(20/N)*t )+x[t]
}

kzp.x<-log(kzp(x,480,1))
spg.x1<-smooth.kzp(kzp.x,0.03, method="DZ")$periodogram
spg.x2<-smooth.kzp(kzp.x,0.03, method="NZ")$periodogram

omega<-seq(0,1,length=481)[2:241]

par(mfrow=c(2,2))
plot(omega,kzp.x,main="Raw periodogram",type="l", xlab="Frequency (cycles/time unit)", ylab=" ")
plot(omega,spg.x1,main="Smoothed Periodogram using DZ method",type="l", xlab="Frequency (cycles/time unit)", yla
plot(omega,spg.x2,main="Smoothed Periodogram using NZ method",type="l", xlab="Frequency (cycles/time unit)", yla

#example 2
#show the effect of KZFT
t<-1:2000
y<-1.1*sin(2*pi*0.0339*t)+7*sin(2*pi*0.0366*t)+2*rnorm(1000,0,1)
kzp.y1<-log(kzp(y,1000,1,0.1,1))
kzp.y2<-log(kzp(y,500,2,0.1,2))

spg.y1<-smooth.kzp(kzp.y1,0.01, method="NZ")$periodogram
spg.y2<-smooth.kzp(kzp.y2,0.01, method="NZ")$periodogram

omega<-seq(0,1,length=1001)[21:61]
par(mfrow=c(2,2))
plot(omega,kzp.y1[20:60], main="Periodogram m=1000, k=1", type="l", xlab="Frequency (cycles/time unit)", ylab="
plot(omega,kzp.y2[20:60], main="Periodogram m=500, k=2", type="l", xlab="Frequency (cycles/time unit)", ylab=" "
plot(omega,spg.y1[20:60], main="Smoothed Periodogram m=1000, k=1", type="l", xlab="Frequency (cycles/time unit)"
plot(omega,spg.y2[20:60], main="Smoothed Periodogram m=500, k=2", type="l", xlab="Frequency (cycles/time unit)",
```

kztp — *Kolmogorov-Zurbenko Third-Order Periodogram and Smoothing Method*

Description

Kolmogorov-Zurbenko third-order periodogram and smoothing method

Usage

```
kztp(x,m,k,p=1,n=1,rp1=0,rp2=0.5,cp1=0,cp2=0.5)
variation.kztp(pg, K=dim(pg)[1])
smooth.kztp(pg,c,K=dim(pg)[1])
```

Arguments

`x`	The raw time series
`m`	The length of the window size for a regular Fourier transform
`k`	The number of iterations for the KZFT
`p`	The distance between two successive intervals as a percentage of the total length of the time series
`n`	The sampling frequency rate as a multiplication of the Fourier frequencies
`rp1`	The left bound of the frequencies at which the third-order periodogram will be calculated in one frequency dimension
`rp2`	The right bound of the frequencies at which the third-order periodogram will be calculated in one frequency dimension
`cp1`	The left bound of the frequencies at which the third-order periodogram will be calculated in the other frequency dimension
`cp2`	The right bound of the frequencies at which the third-order periodogram will be calculated in the other frequency dimension
`pg`	The raw third-order periodogram of a time series
`K`	Half of the maximum bandwidth of the spectral window being allowed
`c`	A prespecified percentage of total variation

Details

Kolmogorov-Zurbenko third-order periodogram is calculated based on the Kolmogorov-Zurbenko Fourier transform. The smoothing algorithm is very similar to the DZ algorithm used in smoothing the periodogram. It is also an adaptive algorithm which allows the bandwidth of the spectral window to vary according to the smoothness of the underlying bispectrum density. The spectral window being used here is a square window. In the algorithm, the bandwith of the square window is extended until the squared variation of the third-order periodogram within the window reaches a prespecified percentage of the total variation of the raw third-order periodogram.

References

I. G. Zurbenko, 1986: The spectral Analysis of Time Series. North-Holland, 248 pp.

I. G. Zurbenko, P. S. Porter, Construction of high-resolution wavelets, Signal Processing 65: 315-327, 1998.

W. Yang, I. G. Zurbenko, A semi-adaptive smoothing algorithm in bispectrum estimation, Proceedings of the American Statistical Association, Seattle, 2006.

See Also

kzft, kzp, kzw.

Examples

```
#example 1

t<-1:10000
y<-sin(2*pi*0.1*t)+sin(2*pi*0.2*t)+rnorm(10000,0,3)

pg3.y<-kztp(y,100,1,0.5,1)$mod
pg3.y<-log(pg3.y)
spg3.y<-smooth.kztp(pg3.y,0.01,20)$bispectrum

omega<-seq(0,1,length=101)[2:51]

par(mfrow=c(2,1))
persp(omega, omega, pg3.y, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
     ltheta = 120, shade = 0.75, ticktype = "detailed", xlab = "Frequency (cycles/time unit)",
      ylab = "Frequency (cycles/time unit)", main = "3rd-order Periodogram")
persp(omega, omega, spg3.y, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
     ltheta = 120, shade = 0.75, ticktype = "detailed", xlab = "Frequency (cycles/time unit)",
      ylab = "Frequency (cycles/time unit)", main = "Smoothed 3rd-order Periodogram")

filled.contour(omega,omega,pg3.y,xlab="Frequency (cycles/time unit)",ylab="Frequency (cycles/time unit)",
               main="3rd-order Periodogram")
filled.contour(omega,omega,spg3.y,xlab="Frequency (cycles/time unit)",ylab="Frequency (cycles/time unit)",
               main="Smoothed 3rd-order Periodogram3")

#example 2
#effect of KZFT

t<-1:50000
y<-1.1*sin(2*pi*0.0339*t)+7*sin(2*pi*0.0366*t)+2*rnorm(50000,0,1)

pg3.y1<-kztp(y,1000,1,0.5,1,rp1=0.02,rp2=0.05,cp1=0.02,cp2=0.05)$mod
pg3.y2<-kztp(y,1000,3,0.5,1,rp1=0.02,rp2=0.05,cp1=0.02,cp2=0.05)$mod
pg3.y1<-log(pg3.y1)
pg3.y2<-log(pg3.y2)
spg3.y1<-smooth.kztp(pg3.y1,0.01,10)$bispectrum
spg3.y2<-smooth.kztp(pg3.y2,0.01,10)$bispectrum
```

```
omega<-seq(0,1,length=1001)[21:51]
par(mfrow=c(2,2))
persp(omega, omega, pg3.y1, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
     ltheta = 120, shade = 0.75, ticktype = "detailed", xlab = "Frequency (cycles/time unit)",
      ylab = "Frequency (cycles/time unit)", main = "Original, m=1000, k=1")
persp(omega, omega, pg3.y2, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
     ltheta = 120, shade = 0.75, ticktype = "detailed", xlab = "Frequency (cycles/time unit)",
      ylab = "Frequency (cycles/time unit)", main = "Original, m=1000, k=3")
persp(omega, omega, spg3.y1, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
     ltheta = 120, shade = 0.75, ticktype = "detailed", xlab = "Frequency (cycles/time unit)",
      ylab = "Frequency (cycles/time unit)", main = "Smoothed, m=1000, k=1")
persp(omega, omega, spg3.y2, theta = 30, phi = 30, expand = 0.5, col = "lightblue",
     ltheta = 120, shade = 0.75, ticktype = "detailed", xlab = "Frequency (cycles/time unit)",
      ylab = "Frequency (cycles/time unit)", main = "Smoothed, m=1000, k=3")

filled.contour(omega,omega,pg3.y1,xlab="Frequency (cycles/time unit)",ylab="Frequency (cycles/time unit)",
               main="Original, m=1000, k=1")
filled.contour(omega,omega,pg3.y2,xlab="Frequency (cycles/time unit)",ylab="Frequency (cycles/time unit)",
               main="Original, m=1000, k=3")
filled.contour(omega,omega,spg3.y1,xlab="Frequency (cycles/time unit)",ylab="Frequency (cycles/time unit)",
               main="Smoothed, m=1000, k=1")
filled.contour(omega,omega,spg3.y2,xlab="Frequency (cycles/time unit)",ylab="Frequency (cycles/time unit)",
               main="Smoothed, m=1000, k=3")
```

---

kzw *Kolmogorov-Zurbenko Wavelet*

Description

Kolmogorov-Zurbenko Wavelet is calculated based on Kolmogorov-Zurbenko Fourier transform.

Usage

```
kzw(x,f1=1/length(x),f2=0.5,delta.f=1/length(x),t1=1,t2=length(x),
delta.t=1,n,k=3,method="zero")
kzww(x,f1=1/length(x),f2=0.5,delta.f=1/length(x),t1=1,t2=length(x),
delta.t=1,m,k=3,method="zero")
```

Arguments

`x`	The raw time series
`f1`	The left bound of frequencies at which time-frequency estimation is calculated
`f2`	The right bound of frequencies at which time-freqeuncy estimation is calculated
`delta.f`	Interval of the sampling frequency
`t1`	The left bound of time
`t2`	The right bound of time
`delta.t`	Interval of time

`n`	The sampling frequency rate as a multiplication of the Fourier frequencies
`m`	The length of the window size for a regular Fourier transform
`k`	The number of iterations for the KZFT
`method`	The methods of how to extend time series at either end. The options can be either "zero" (adding zeroes) or "repeat" (symmetrically repeat the time series)

Details

Kolmogorov-Zurbenko Wavelet (KZW) is used to calculte the time-frequency estimation of a time series. It is calculated based on Kolmogorov-Zurbenko Fourier transform. Because of this, it is strongly robust with respect to noise and has a very high resolution property. There are two versions of KZW algorithm, kzw and kzww. The only difference between the two algorithms is the definition of the window size. In kzw, the window size is a multiplication of the period at which the spectrum is calculated. In another word, different length of window is used to calculate the spectrum at different frequencies. In kzww, the window size is fixed for all interested.

References

I. G. Zurbenko, 1986: The spectral Analysis of Time Series. North-Holland, 248 pp.

I. G. Zurbenko, P. S. Porter, Construction of high-resolution wavelets, Signal Processing 65: 315-327, 1998.

R. Neagu, I. G. Zurbenko, Tracking and separating non-stationary multi-component chirp signals, Journal of Franklin Institute 339: 499-520, 2002.

See Also

`kzft`, `kzp`, `kztp`.

Examples

```
# example
t=1:999
p1=rep(10,333)
p2=rep(20,333)
p=c(p1,p2,p1)
f=1/p

x=sin(2*pi*f*t)+rnorm(999,0,1)
plot(t,x,type='l',xlab="Time unit")

kzw.x=kzw(x,30/999,120/999,1/999,1,999,1,7,5)
ff=30:120/999
contour(t,ff,kzw.x$em, xlab="Time unit", ylab="Frequency (cycles/time unit)")
lines(t,f,col="red")
```

Index

∗Topic **ts**
  kzft, 2
  kzp, 4
  kztp, 7
  kzw, 9

coeff.kzft *(*kzft*)*, 2

kzft, 2, *5*, *8*, *10*
kzp, *3*, 4, *8*, *10*
kztp, *3*, *5*, 6, *10*
kzw, *3*, *5*, *8*, 9
kzww *(*kzw*)*, 9

nonlinearity.kzp *(*kzp*)*, 4

smooth.kzp *(*kzp*)*, 4
smooth.kztp *(*kztp*)*, 7

transfun.kzft *(*kzft*)*, 2

variation.kzp *(*kzp*)*, 4
variation.kztp *(*kztp*)*, 7