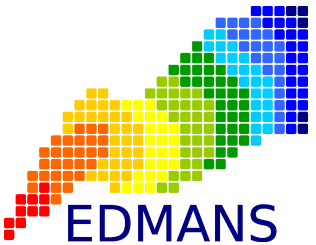
Librería KDSeries (Knowledge Discovery from Time Series)

III Jornadas de Usuarios de R (Noviembre, 2011)

Martínez-de-Pisón, F.J., Fernández, R., Sanz, A. & Antoñanzas, F.

Grupo EDMANS (http://www.mineriadatos.com)



Índice

Tendencias Actuales en la Generación de Información

1. Existen un **aumento significativo de la información que se genera en tiempo real** debido a nuevos avances en: redes sociales, sistema de apoyo a la toma de decisiones, aumento de herramientas de gestión, información debida a sensores remotos (satélites, cámaras, etc.), automatización de procesos industriales, almacenamiento de video y audio, información generada en la «nube», etc.
2. Las empresas demandan, cada vez más, **herramientas que les ayuden a analizar y extraer conocimiento oculto y útil de toda esa información**.
3. Mucha de esa información incluye una «**componente temporal**», es decir, es generada a lo largo del tiempo y en modo continuo.

La Minería de Datos Temporal

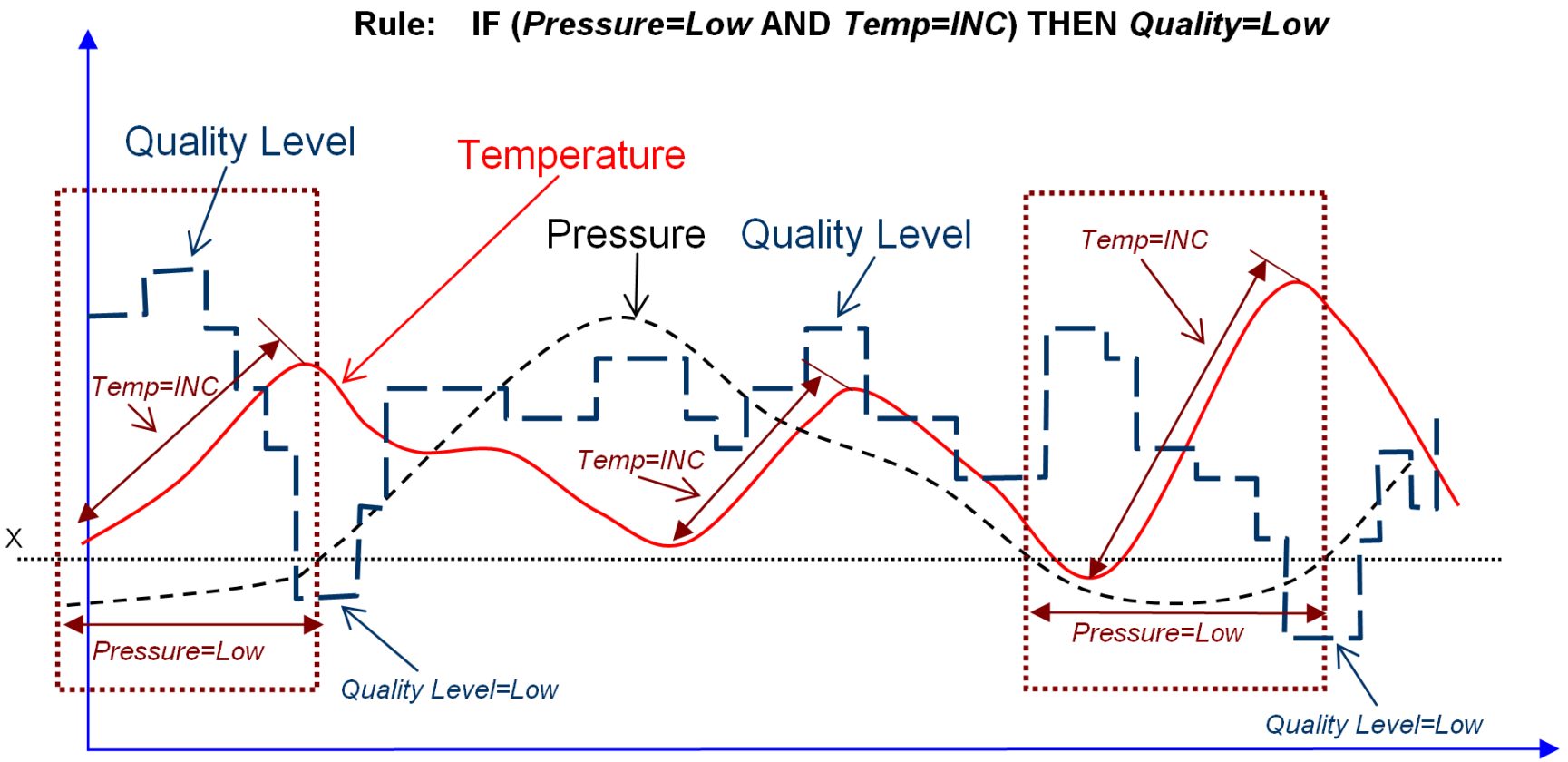
Un dominio de investigación que está muy activo actualmente es el desarrollo de técnicas de Minería de Datos (MD) que permitan trabajar con información que incluya una «componente temporal» (*Temporal Data Mining*, *TDM*).
El objetivo de la TDM consiste fundamentalmente en la búsqueda y extracción de conocimiento oculto y no trivial a partir de este tipo de información.

- **Data stream**: flujo continuo y masivo de datos temporales de duración infinita.
- **Datos secuenciales**: datos temporales ordenados pero sin noción concreta del tiempo.
- **Serie temporal**: datos obtenidos mediante medidas repetitivas en unos tiempos determinados. Muy común en proceso industriales, financieros, empresariales, etc.

Índice

Conocimiento Oculto Inter-transaccional en ST

La idea es poder **encontrar relaciones inter-transaccionales dentro de las series temporales en forma de reglas de conocimiento** del tipo «SI ... ENTONCES ...»

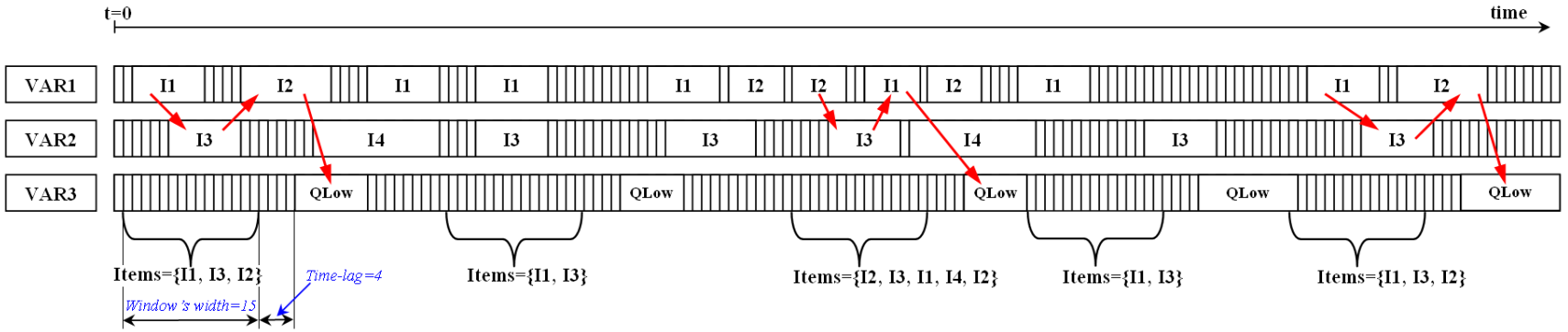


Desarrollo de KDSeries

Objetivo de la librería KDSeries

Desarrollo de funciones útiles para la **búsqueda de reglas de asociación inter-transaccional entre series temporales**: análisis exploratorio, preprocesado y transformación de series temporales, detección de episodios significativos, búsqueda de item frecuentes, extracción de reglas de asociación, etc.

Funciones útiles para **realizar el proceso KDD de un modo iterativo e interactivo con el usuario** pues NO es posible automatizar cada una de las fases o, por lo menos, es muy dificil.



Transactions Database:

1. (t=4) : {I1, I3, I2}
2. (t=41) : {I1, I3}
3. (t=79) : {I2, I3, I1, I4, ~~I2~~}
4. (t=105) : {I1, I3}
5. (t=134) : {I1, I3, I2}

Frequent Itemsets with Support > 50% (>2):

Level 1: {I1=5}, {I2=3}, {I3=5}
Level 2: {I1, I3}=5, {I1, I2}=3, {I2, I3}=3
Level 3: {I1, I2, I3}=3

Association Rules with Support > 50% (>2):

IF {*I1 AND I3*} THEN {*QLow*} (Support=5/5=100%)
IF {*I1*} THEN {*QLow*} (Support=5/5=100%)
IF {*I3*} THEN {*QLow*} (Support=5/5=100%)
IF {*I1 AND I2 AND I3*} THEN {*QLow*} (Support=3/5=60%)
IF {*I1 AND I2*} THEN {*QLow*} (Support=3/5=60%)
IF {*I2 AND I3*} THEN {*QLow*} (Support=3/5=60%)
IF {*I2*} THEN {*QLow*} (Support=3/5=60%)

Índice

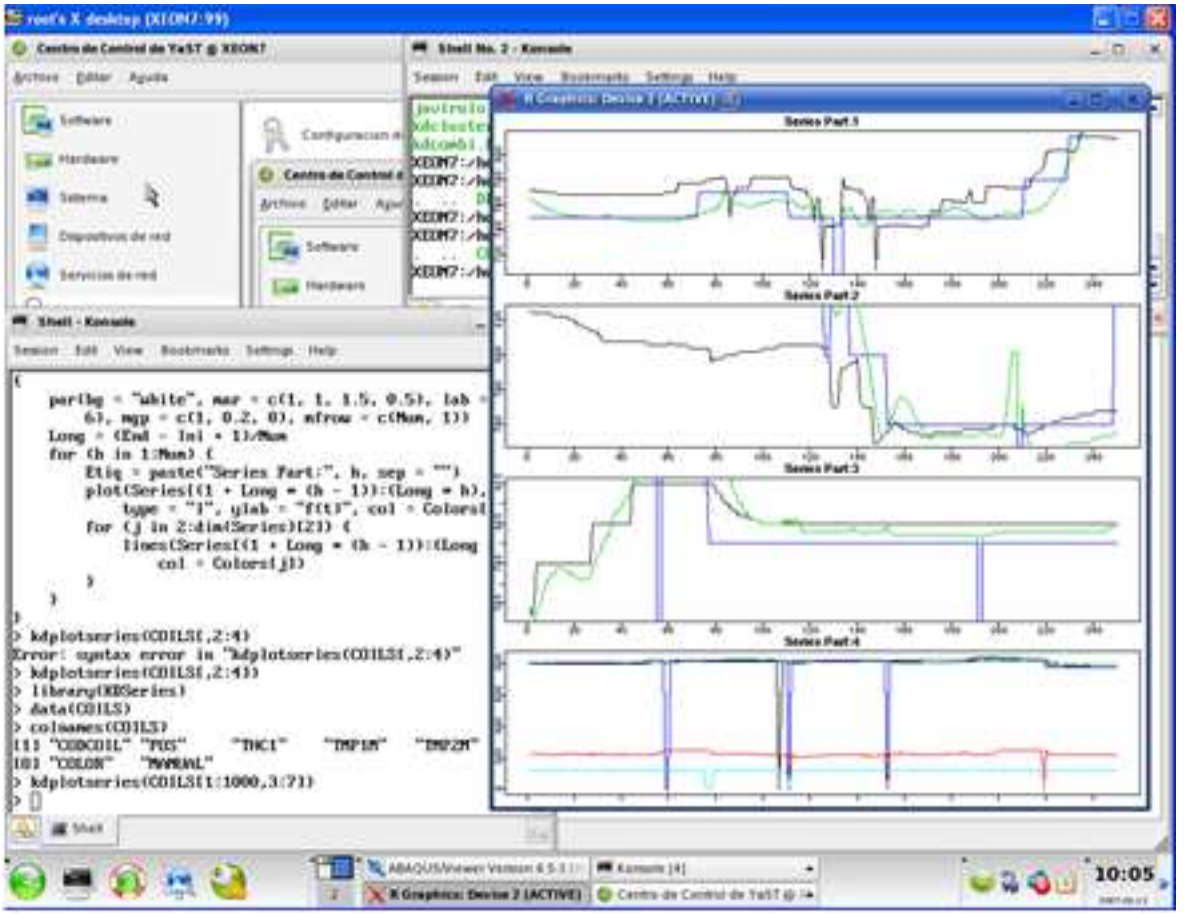
Funciones para la Metodología Propuesta

1. Filtrado de cada serie temporal para eliminar el ruido y obtener la «forma básica» de la misma.
2. Obtención de los máximos y mínimos más importantes.
3. Extracción de los eventos característicos de cada serie temporal.
4. Agrupación de eventos en episodios «significativos» segun criterios definidos por el experto.
5. Fijación del «Consecuente» de la Regla.
6. Búsqueda de items frecuentes según una ventana y desplazamiento definidos por el usuario a partir de un consecuente.
7. Extracción de las reglas de asociación inter-transaccional según unos umbrales de calidad preestablecidos por el analista.

Función: *kdplotseries()*

Función: *kdplotseries(Series, Num=4,Ini=1,End=dim(Series)[1], Colors=1:dim(Series)[2])*
Resultado: Gráfica con las ST en varias divisiones horizontales.

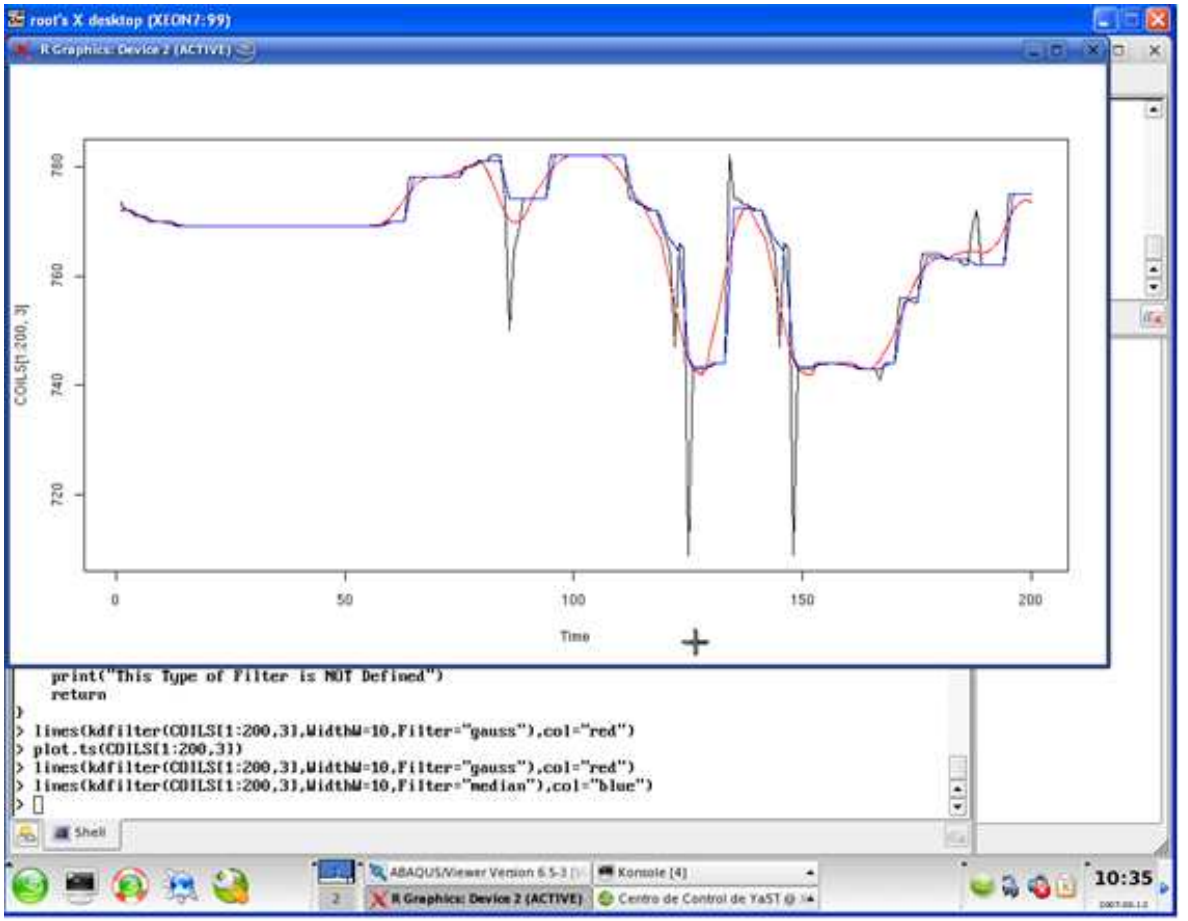
Parámetros de Entrada:
Series=Matriz compuesta por series temporales (cada columna es una ST)
Num=Número de divisiones horizontales de la figura.
Ini=Primer Punto.
End=Ultimo Punto.
Colors=Lista de Colores para cada ST.
Parámetros de Salida:
Ninguno



Función: *kdfilter() y kdmatfilter()*

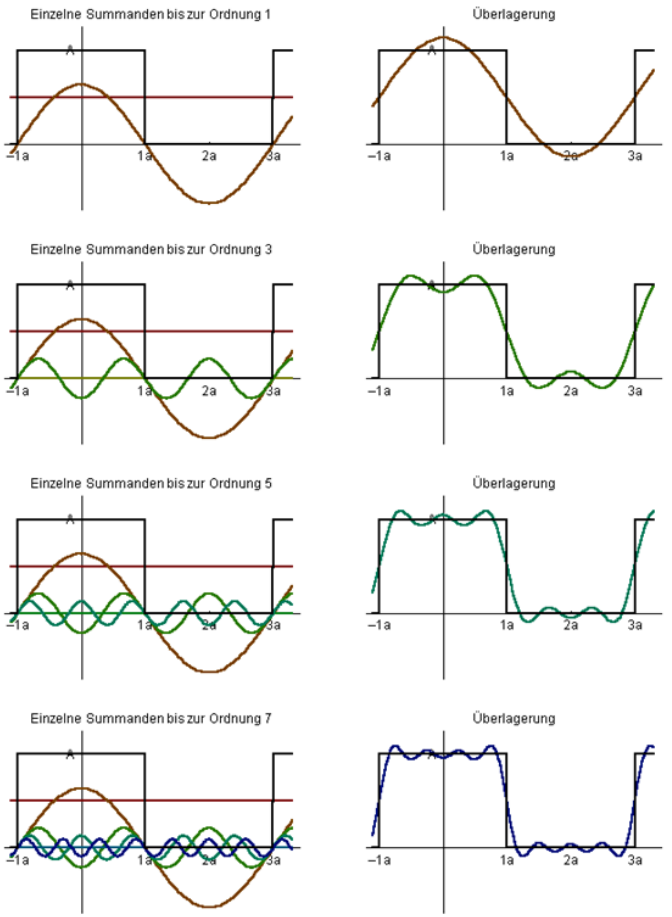
Función: *kdfilter(TSerie, WidthW, Filter="gauss") y kdmatfilter(TSerie, WidthWVect, Filter="gauss")*
Resultado: Filtrado de una serie temporal con uno o varios filtros de ventana deslizante.

Parámetros de Entrada:
TSerie=Serie Temporal a Filtrar.
WidthW o **WidthWVect**=Tamaño de la ventana o ventanas del filtro deslizante.
Filter=Filtro a aplicar: "gauss"=gaussiano, "mean" o "rect"=media, "median"=mediana, "max"=máximo, "min"=mínimo.
Parámetros de Salida:
Vector o Matriz con la Serie o Series Filtradas.



Función: *kdfilterFFT()*

Función: *kdfilterFFT(TSerie, WidthW=0.02*length(TSerie), Slide=WidthW, Range=c(20,70))*
Resultado: Filtrado de una serie temporal con filtro de transformada rápida de Fourier.



Parámetros de Entrada:
TSerie=Serie temporal.
WidthW = Anchura de la ventana.
Slide =Desplazamiento de la ventana.
Range=Rango de frecuencias a mantener (en porcentaje).
Parámetros de Salida:
Serie temporal filtrada.

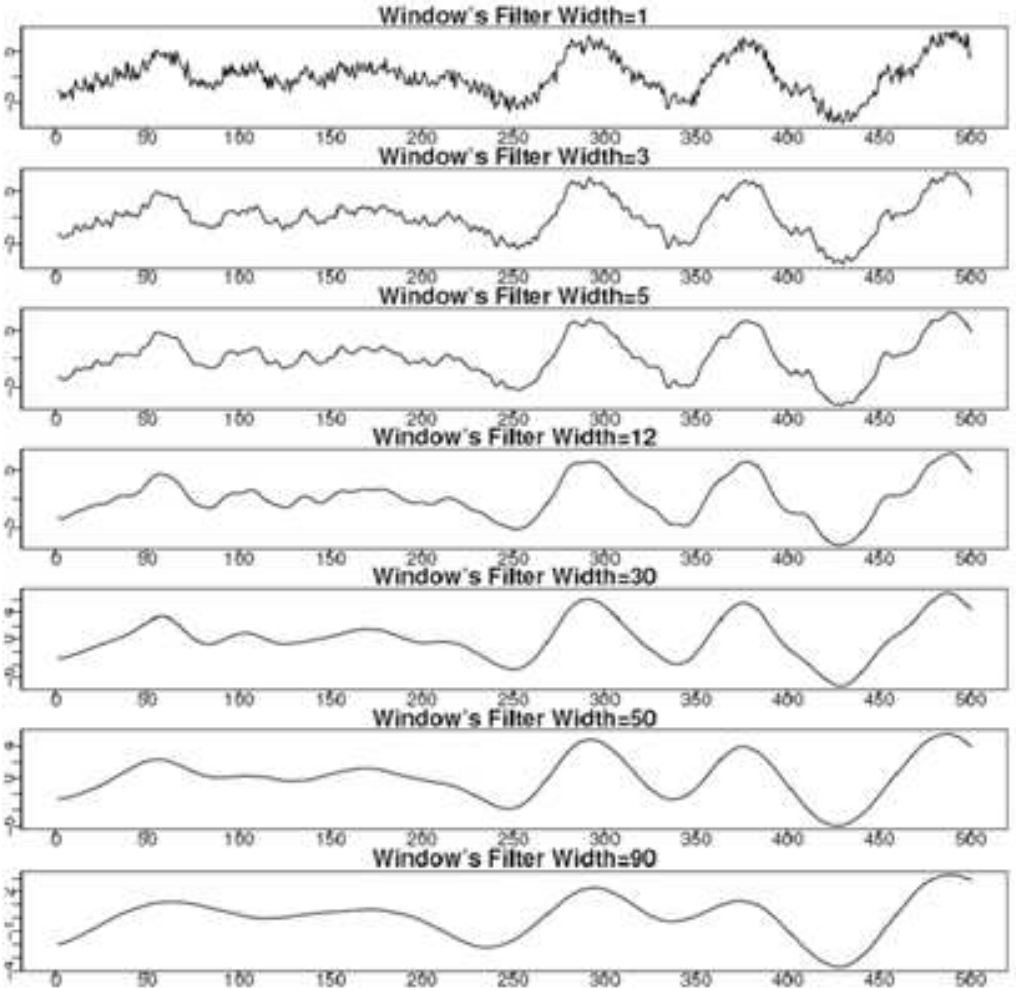
Función: *kdfilterremove()*

Función: *kdfilterremove(TSerie, MinT, Filter="min")*
Resultado: Elimina los valores que están por encima o por debajo de un valor; o dentro de un rango de valores o fuera de él.
Parámetros de Entrada:
TSerie=Serie temporal.
MinT= Valor umbral o vector con los dos valores del Rango.
TFilter= «min» elimina valores que estén por debajo, «max» elimina aquellos que estén por encima, «range» elimina los valores que están dentro del rango, «rangeinv» elimina los valores que estén fuera del rango.
Range=Rango de frecuencias a mantener (en porcentaje).
Parámetros de Salida:
Serie temporal filtrada.

Función: *kdplotmat()*

Función: *kdplotmat (MAT, Positions=1:dim(MAT$MFilt)[1], Ini=1,End=dim(MAT$MFilt)[2])*
Resultado: Gráfica de una serie filtrada con diferentes anchos de ventana.

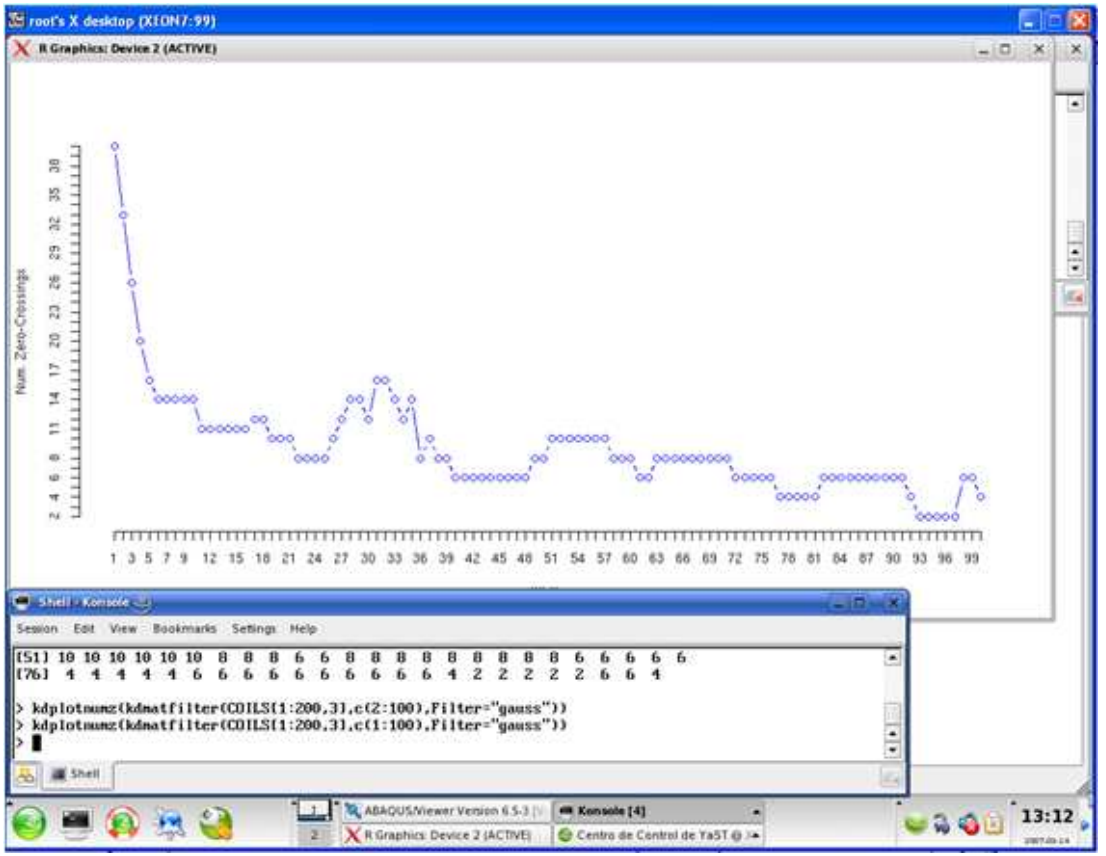
Parámetros de Entrada:
MAT=Matriz con diferentes filtros de una ST (viene de kdmatfilter).
Positions=Indica cuales son las series filtradas que queremos visualizar.
Filter=Filtro a aplicar.
"gauss"=gaussiano, "mean" o "rect"=media,
"median"=mediana,
"max"=máximo,
"min"=mínimo.
Ini=Primer Punto a dibujar.
End=Ultimo Punto a dibujar.
Parámetros de Salida:
Ninguno.



Función: *kdplotnumz()*

Función: *kdplotnumz (MAT, Positions=1:dim(MAT$MFilt)[1])*
Resultado: Muestra en una gráfica la evolución del número de cruces por cero para una matriz con una serie temporal filtrada con diferentes anchos. Esto permite, determinar el mejor ancho de ventana de filtrado.

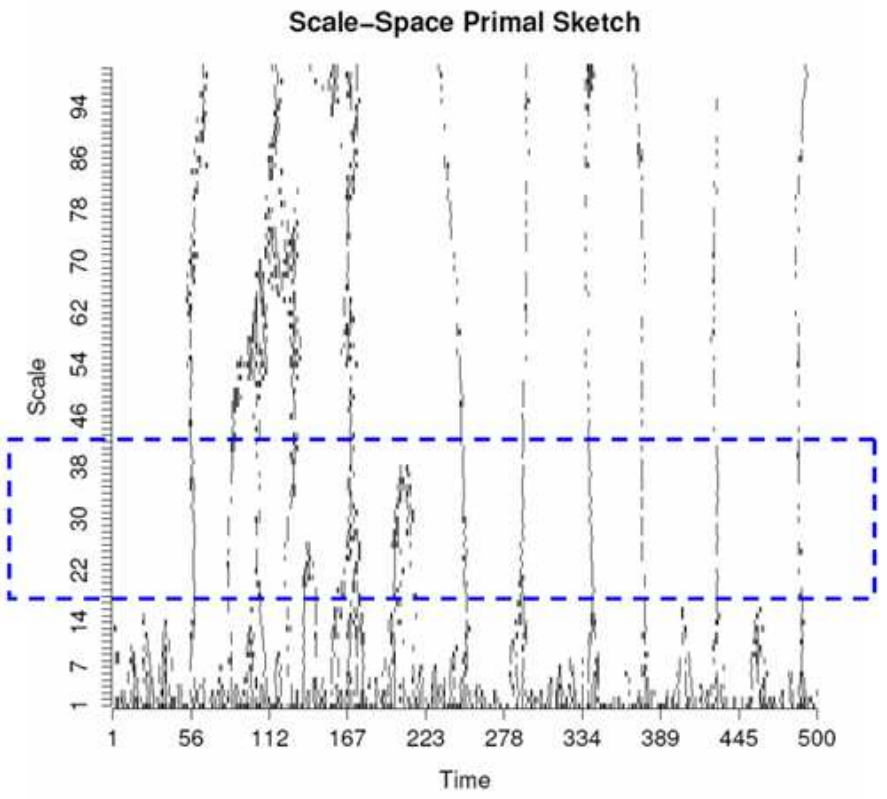
Parámetros de Entrada:
MAT=Matriz con diferentes filtros de una ST (viene de kdmatfilter()).
Positions=Indica cuales son las series filtradas que queremos visualizar.
Parámetros de Salida:
Ninguno.



Función: *kdplotscales()*

Función: *kdplotscales(MAT, Positions=1:dim(MAT$MFilt)[1],Ini=1,End=dim(MAT$MFilt)[2])*
Resultado: Muestra en una gráfica donde están situados los máximos y mínimos de una serie temporal filtrada con diferentes anchos. Esto permite, determinar los rangos mejores de anchos de ventana de filtrado que distorsionen lo menos posible la ST pero que reduzcan el número de máximos y mínimos.

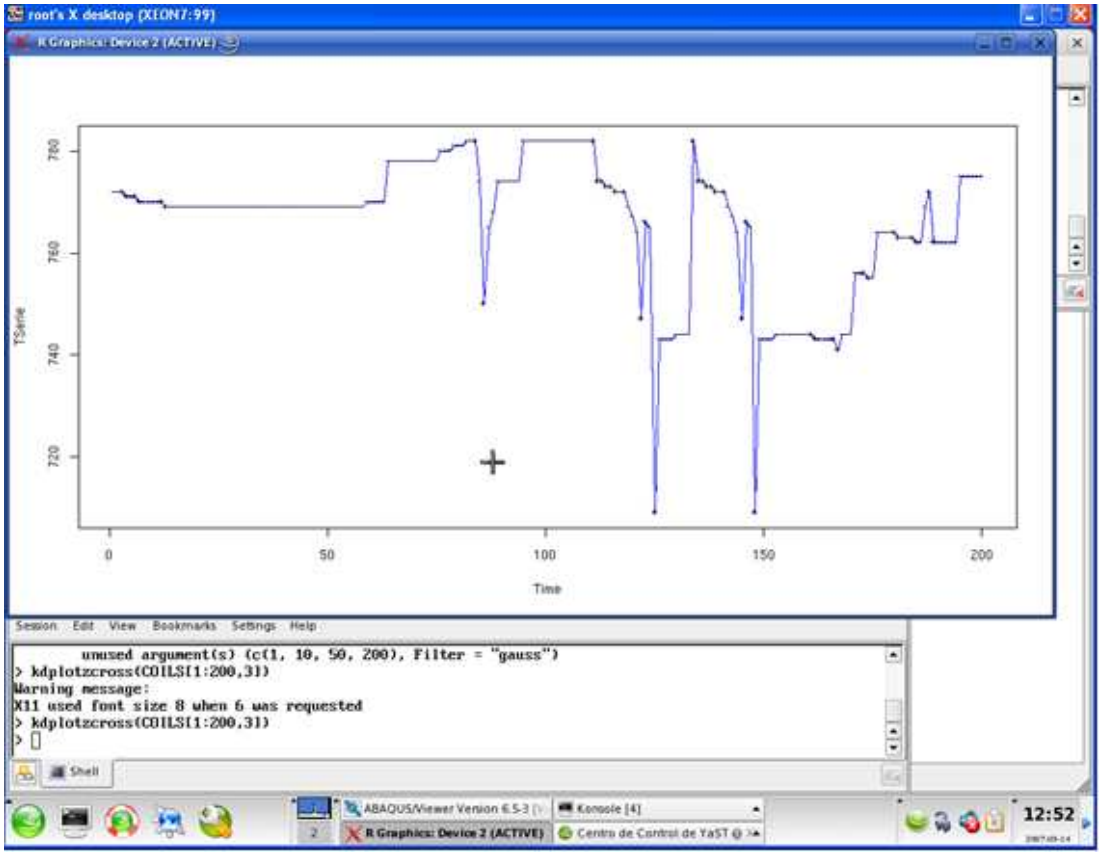
Parámetros de Entrada:
MAT=Matriz con diferentes filtros de una ST (viene de kdmatfilter()).
Positions=Indica cuales son las series filtradas que queremos visualizar.
Ini=Punto inicial.
End=Punto final.
Parámetros de Salida:
Ninguno.



Función: *kdplotzcross()*

Función: *kdplotzcross(TSerie)*
Resultado: Visualiza dónde se producen los cruces por cero de las segunda derivada para identificar los máximos y mínimos de la función.Gráfica de una serie filtrada con diferentes anchos de ventana.

Parámetros de Entrada:
TSerie=Serie temporal.
Parámetros de Salida:
Ninguno.



Función: *kdmaxmin()*

Función: *kdmaxmin(TSerie, R=1.005)*
Resultado: Obtiene los máximos y mínimos caracaterísticos de acuerdo con un Ratio de compresión definido en: (Fink and Pratt, 2004).

Parámetros de Entrada:
TSerie=Serie temporal.
R=Ratio de compresión de la ST de acuerdo a: (Fink and Pratt, 2004).
Parámetros de Salida:
Posición de los máximos y mínimos dentro de la ST.

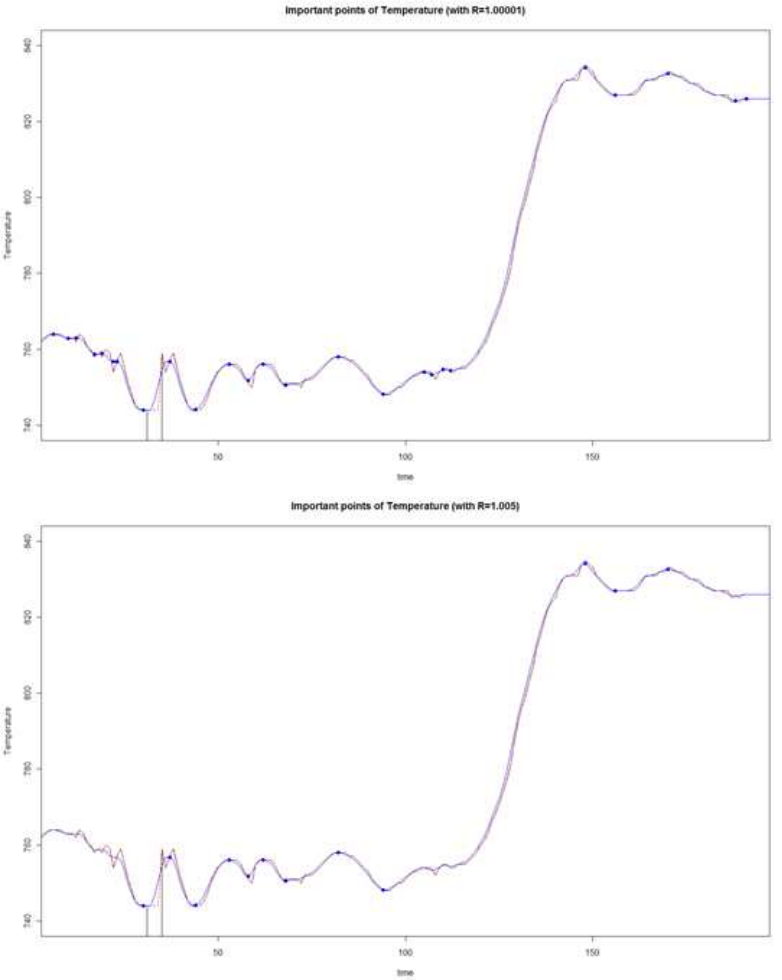
A point a_m of a series $\{a_1,...,a_n\}$ is considered an important minimum if there are indices i and j such that:

$$\begin{cases} a_m \text{ is a mininum among } a_i,\ldots,a_j, \text{ and} \\ \frac{a_i}{a_m} \geq R \quad \text{AND} \quad \frac{a_j}{a_m} \geq R \quad (\text{where, } i \leq m \leq j) \end{cases} \tag{1}$$

and similarly, a_m as an important maximum if:

$$\begin{cases} a_m \text{ is a maximum among } a_i,\ldots,a_j, \text{ and} \\ \frac{a_i}{a_m} \geq R \quad \text{AND} \quad \frac{a_j}{a_m} \geq R \quad (\text{where, } i \leq m \leq j) \end{cases} \tag{2}$$

where R is a (compression) rate which is always greater than one.



Función (I): *kdextract()*

Función: *kdextract(TSerie, Param=c(rep(TRUE,13)), Threshold=c(0.05,0.05,0.05,0.40,0.10,0.40,0.10,0.8,0.2,0.05), LongMin=rep(0,13))*
Resultado: Permite la extracción automática de TODOS los eventos según diversos criterios establecidos y una serie de umbrales configurables por el usuario.
Parámetros de Entrada:

TSerie=Serie temporal (se recomienda filtrarla previamente).
Param (TRUE o FALSE)=
[1]=Obtiene eventos incrementales: **INCREMENTAL**
[2]=Obtiene eventos decrementales: **DECREMENTAL**
[3]=Obtiene eventos horizontales: **HORIZONTAL**
[4]=Obtiene eventos incrementales por encima de un umbral positivo: **INCHIGH**
[5]=(Obtiene eventos incrementales por debajo de un umbral positivo: **INCLOW**
[6]=Obtiene eventos decrementales por encima de un umbral positivo: **DECHIGH**
[7]=Obtiene eventos decrementales por debajo de un umbral positivo: **DECLOW**
[8]=Obtiene eventos por encima de un umbral positivo: **VALHIGH**
[9]=Obtiene eventos por debajo de un umbral positivo: **VALLOW**
[10]=Obtiene eventos por cerca de cero: **VALZERO**
[11]=Obtiene eventos entre VALHIGH y VALLOW: **VALMED**
[12]=Obtiene eventos entre INCHIGH y INCLOW: **INCMED**
[13]=Obtiene eventos entre DECHIGH y DECLOW: **DECMED**

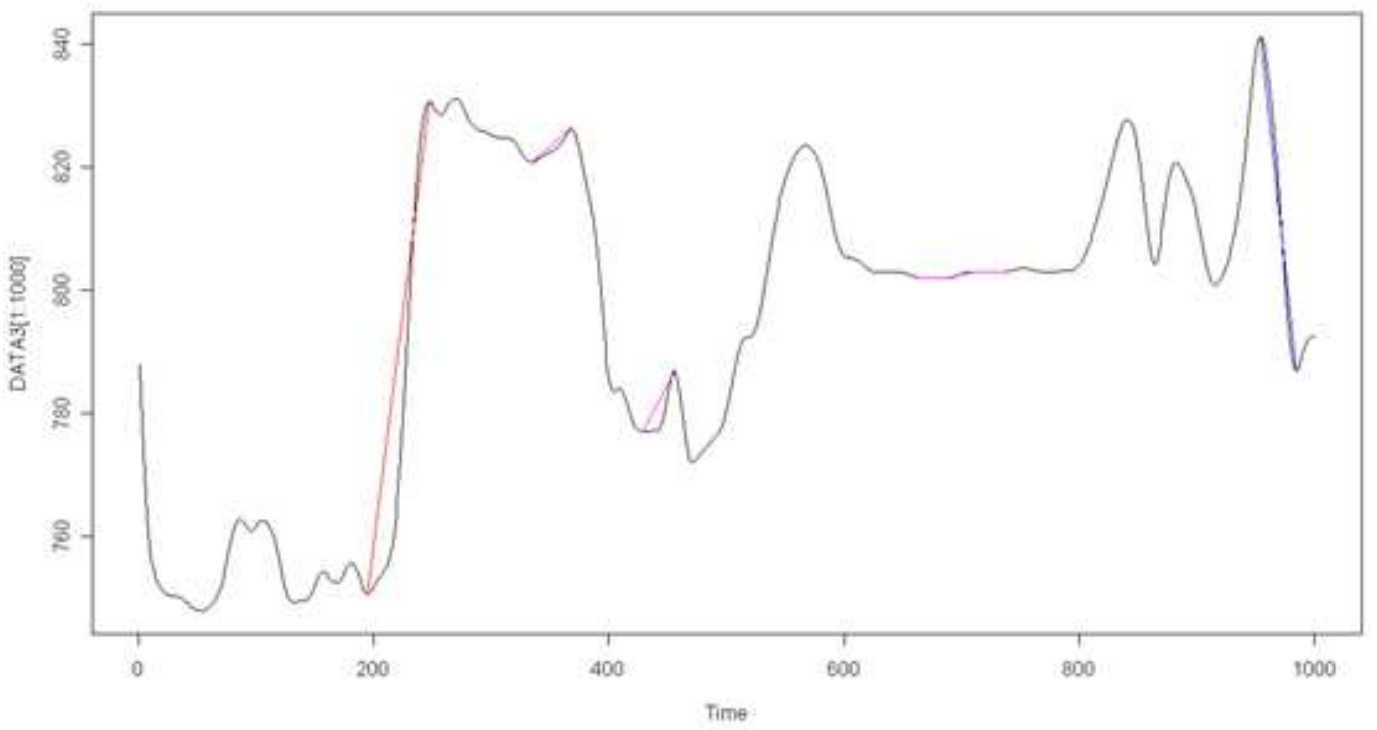
Threshold=Umbrales de corte
[1]= % del rango(TSerie)=si es INC y ESTÁ POR ENCIMA DE ESE VALOR este evento es INC
[2]= % del rango(TSerie)=si DEC y ESTÁ POR ENCIMA DE ESE VALOR este evento es DECREMENTAL
[3]= % del rango(TSerie)=si INC o DEC sino ESTÁ POR DEBAJO DE ESE VALOR este evento es HORIZONAL
[4]= % del rango(TSerie)=si INC y ESTÁ POR ENCIMA DE ESE VALOR este evento es INCHIGH
[5]= % del rango(TSerie)=si INC y ESTÁ POR DEBAJO DE ESE VALOR este evento es INCLOW
[6]= % del rango(TSerie)=si DEC y ESTÁ POR ENCIMA DE ESE VALOR este evento es DECHIGH
[7]= % del rango(TSerie)=si DEC y ESTÁ POR DEBAJO DE ESE VALOR este evento es DECLOW
[8]= % del rango(TSerie)=si TSerie ESTÁ POR ENCIMA DE ESE VALOR este evento es VALHIGH
[9]= % del rango(TSerie)=si TSerie ESTÁ POR DEBAJO DE ESE VALOR este eventoe es VALLOW
[10]= % of max(abs(TSerie))=si abs(TSerie) ESTÁ ENTRE 0 y ESE VALOR este evento es VALZERO

LongMin =Número mínimo de puntos para que un evento pueda ser considerado como tal.

Función (y II): *kdextract()*

Parámetros de Salida:

$MATPatt=Lista con los diferentes segmentos encontrados
$TSerie=Serie Temporal
$DEC=DEC Segments [col1=Posición, col2=Longitud, col3=Altura]
$INC=INC Segments [col1= Posición, col2= Longitud, col3= Altura]
$HOR=HOR Segments [col1= Posición, col2= Longitud, col3= Altura]
Etc....



Función (I): *kdextractSubPatt()*

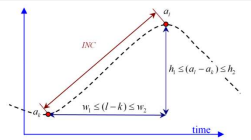
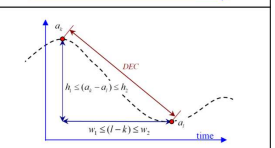
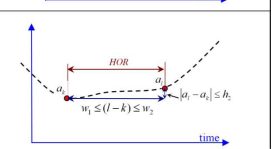
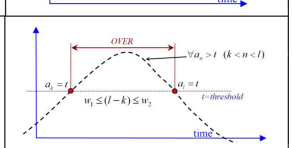
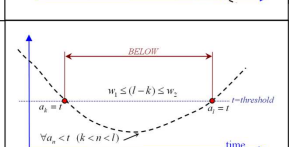
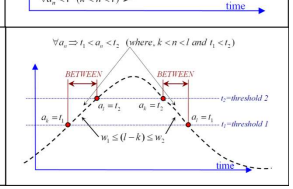
Función: *kdextractsubpatt <- function(TSerie, pattType, pattRangeX=NULL, pattRangeY=NULL, rangeType=c("N","N"), threshold=NULL, level=NULL, namePatt)*
Resultado: Permite extraer eventos configurables por el usuario

Parámetros de Entrada:

TSerie=Serie temporal filtrada.
pattType= Tipo de subpatrón a buscar: Incremental ("I"), decremental ("D"), horizontal ("H") or threshold ("T")
pattRangeX= Rango formado por un vector de dos valores c(mínimo, máximo) de X en los que tiene que estar comprendido el subpatrón.
pattRangeY= Rango formado por un vector de dos valores c(mínimo, máximo) de Y en los que tiene que estar comprendido el subpatrón.
rangeType= Si se consideran los rangos de X e Y por valores fijos (N) o porcentajes (P). Por ejemplo, un vector c("N","P") considera los rangos de X por valores fijos y los de Y por porcentajes.
threshold= Valores de corte en donde buscar los patrones. Por defecto es NULL.
level= Los patrones se buscarán por encima ("+"), debajo ("-") or entre dos valores ("+-") de threshold. Por defecto es NULL.
namePatt= Nombre del patron encontrado.

Parámetros de Salida:

MATPatt=Matriz con los patrones encontrados [col1=Posición, col2=longitud, col3=altura, col4=Nombre del Patron]

Type	*User parameters*	*Conditions*	*Visual interpretation*
Incremental Event (INC)	$\{w_1, w_2\}$ = Range of X within which the curve is to be contained. $\{h_1, h_2\}$ = Range of Y within which the curve is to be contained.	a_k is a minimum a_l is a maximum $w_1 \le (l-k) \le w_2$ $h_1 \le a_l - a_k \le h_2$	
Decremental Event (DEC)	$\{w_1, w_2\}$ = Range of X within which the curve is to be contained. $\{h_1, h_2\}$ = Range of Y within which the curve is to be contained.	a_k is a maximum a_l is a minimum $w_1 \le (l-k) \le w_2$ $h_1 \le a_k - a_l \le h_2$	
Horizontal Event (HOR)	$\{w_1, w_2\}$ = Range of X within which the curve is to be contained. $\{0, h_2\}$ = Range of Y within which the curve is to be contained.	a_k and a_l are important points $w_1 \le (l-k) \le w_2$ $\lvert a_l - a_k \rvert \le h_2$	
Event over a threshold (OVER)	$\{w_1, w_2\}$ = Range of X within which the curve is to be contained. t = Threshold value.	$a_k = t$ $a_l = t$ $w_1 \le (l-k) \le w_2$ $\forall a_n > t \ \ (k < n < l)$	
Event below a threshold (BELOW)	$\{w_1, w_2\}$ = Range of X within which the curve is to be contained. t = Threshold value.	$a_k = t$ $a_l = t$ $w_1 \le (l-k) \le w_2$ $\forall a_n < t \ \ (k < n < l)$	
Event between two thresholds (BETWEEN)	$\{w_1, w_2\}$ = Range of X within which the curve is to be contained. t_1 = Min. threshold value. t_2 = Max. threshold value.	$a_k = t$ $a_l = t$ $w_1 \le (l-k) \le w_2$ $\forall a_n \Rightarrow t_1 < a_n < t_2$ *(where, $k < n < l$)*	

Función (II): *kdextractSubPatt()*

Example

```
> library(KDSeries)
> data(COILS)
> # Preprocessing Temp 2
> DATA <- COILS$TMP2M
> plot.ts(DATA)
> DATA2 <- kdfilterremove(DATA,200)
> plot.ts(DATA2[1:1000])
> DATA3 <- kdfilter(DATA2,20)
> plot.ts(DATA3[1:1000])
> #I <- kdextractsubpatt(DATA3,"I",pattRangeY=c(50,100),namePatt="INC1")
> I <- kdextractsubpatt(DATA3,"I",pattRangeY=c(50,100), namePatt="INC1",threshold=c(780,800),level="+-")
> I$PATT[1:10,]
PosP LongP PosY AltP namePatt
1 195 52 750.7536 79.88099 INC1
2 1497 43 759.9018 70.23312 INC1
3 1798 52 725.4275 90.68528 INC1
4 4236 33 799.3201 55.68223 INC1
5 4324 50 793.9364 60.25899 INC1
6 4514 21 747.6423 52.64799 INC1
7 8099 35 734.4793 88.29717 INC1
8 13202 38 773.7443 54.69858 INC1
9 13352 22 773.8036 73.10013 INC1
10 16472 32 751.0924 80.25870 INC1
> segments(x0=I$PATT[,1],y0=I$PATT[,3], x1=I$PATT[,1]+I$PATT[,2],y1=I$PATT[,3]+I$PATT[,4],col="red")
```

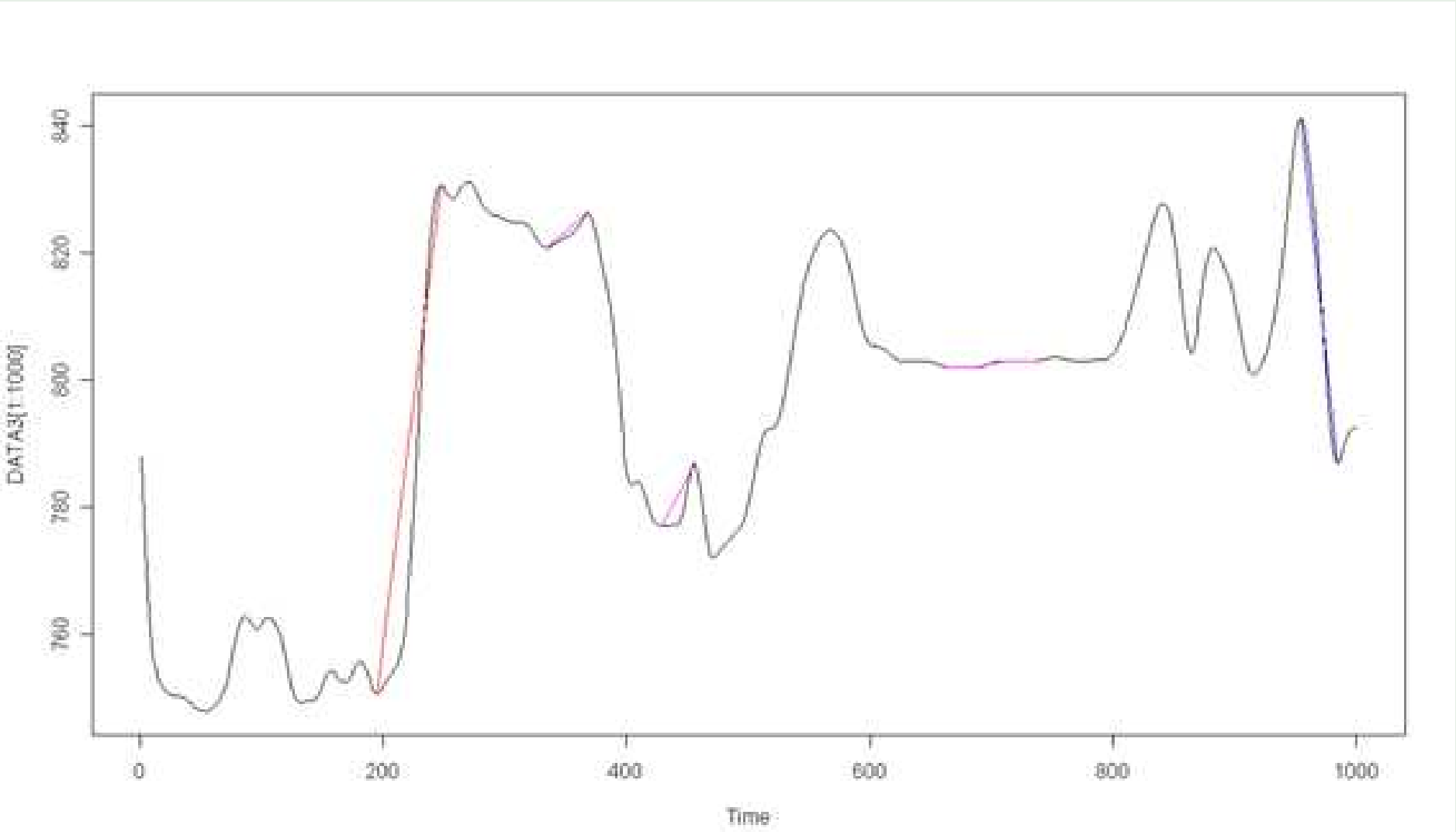
Función (III): *kdextractSubPatt()*

Example

```
> D <- kdextractsubpatt(DATA3,"D",pattRangeY=c(50,100),namePatt="DEC1")
> D$PATT[1:10,]
PosP LongP PosY AltP namePatt
1 954 30 841.1657 -54.25534 DEC1
2 1574 24 826.4428 -58.58808 DEC1
3 1729 26 774.1969 -57.12766 DEC1
4 2038 35 824.5869 -50.62410 DEC1
5 3941 29 827.9818 -68.61295 DEC1
6 4269 34 855.0023 -58.39322 DEC1
7 4422 36 826.8562 -76.86638 DEC1
8 8578 47 834.3368 -63.98063 DEC1
9 13240 37 828.4429 -65.27953 DEC1
10 14101 22 850.4298 -62.70058 DEC1
> segments(x0=D$PATT[,1],y0=D$PATT[,3],x1=D$PATT[,1]+D$PATT[,2],y1=D$PATT[,3]+D$PATT[,4],col="blue")
> H <- kdextractsubpatt(DATA3,"H",pattRangeX=c(20,1000),pattRangeY=c(0,10),namePatt="HOR1")
> H$PATT[1:10,]
PosP LongP PosY AltP namePatt
1 334 34 820.7827 5.547854 HOR1
2 429 27 776.9736 9.991085 HOR1
3 664 23 802.0000 0.000000 HOR1
4 687 23 802.0000 1.000000 HOR1
5 710 26 803.0000 0.000000 HOR1
6 1041 21 789.4338 3.473113 HOR1
7 1182 23 788.0421 2.435395 HOR1
8 1438 24 784.2028 3.690147 HOR1
9 2178 20 773.2475 5.233391 HOR1
10 2240 41 772.3590 2.641050 HOR1
> segments(x0=H$PATT[,1],y0=H$PATT[,3],x1=H$PATT[,1]+H$PATT[,2],y1=H$PATT[,3]+H$PATT[,4],col="magenta")
> MATTOTAL <- rbind(D$PATT,I$PATT, H$PATT)
```

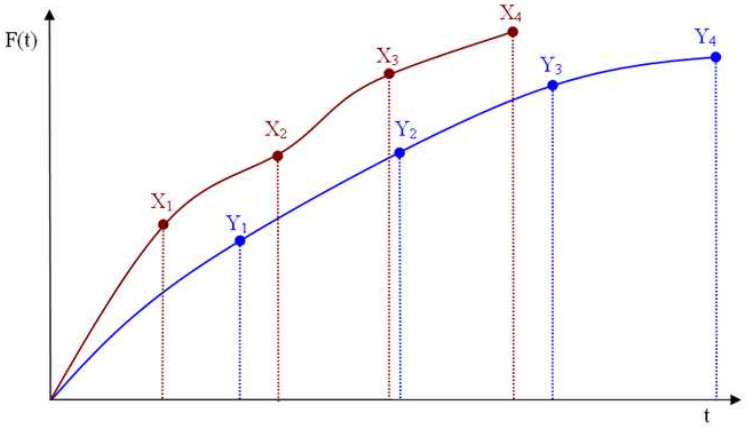
Función (y IV): *kdextractSubPatt()*

Example



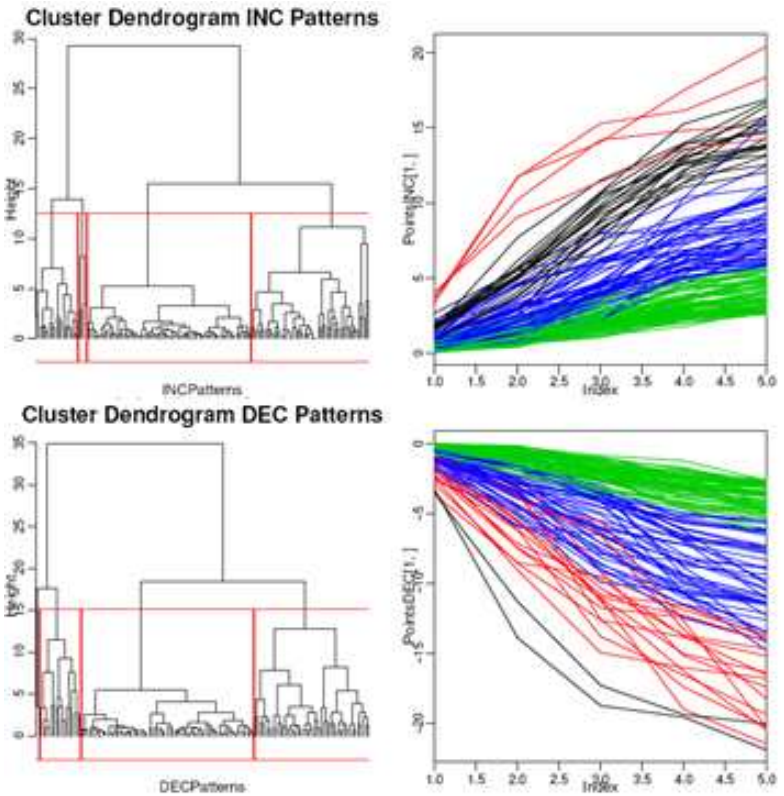
Función (I): *kdclusterpatt()*

Función: *kdclusterpatt(MATPatt, INCParam=c(10,"g",4,"g",4), DECParam=c(10,"g","4","g",4), HORParam=c("g","4"))*
Resultado: Agrupa segmentos similares a partir de la distancia euclídea. Solamente es válido para segmentos del tipo INC, DEC y HOR. El proceso de segmentación, se realiza mediante el uso de dendrogramas que obtienen grupos a partir de dos datos obtenidos de los segmentos: la longitud y la forma. El segundo parámetro corresponde con la distancia euclídea de valores equidistantes entre segmentos.



$$DIS_{Shape} = \frac{1}{N} \cdot \sqrt{\sum_{i=1}^{N} \left(X(i \cdot \frac{L_X}{N}) - Y(i \cdot \frac{L_Y}{N}) \right)^2}$$

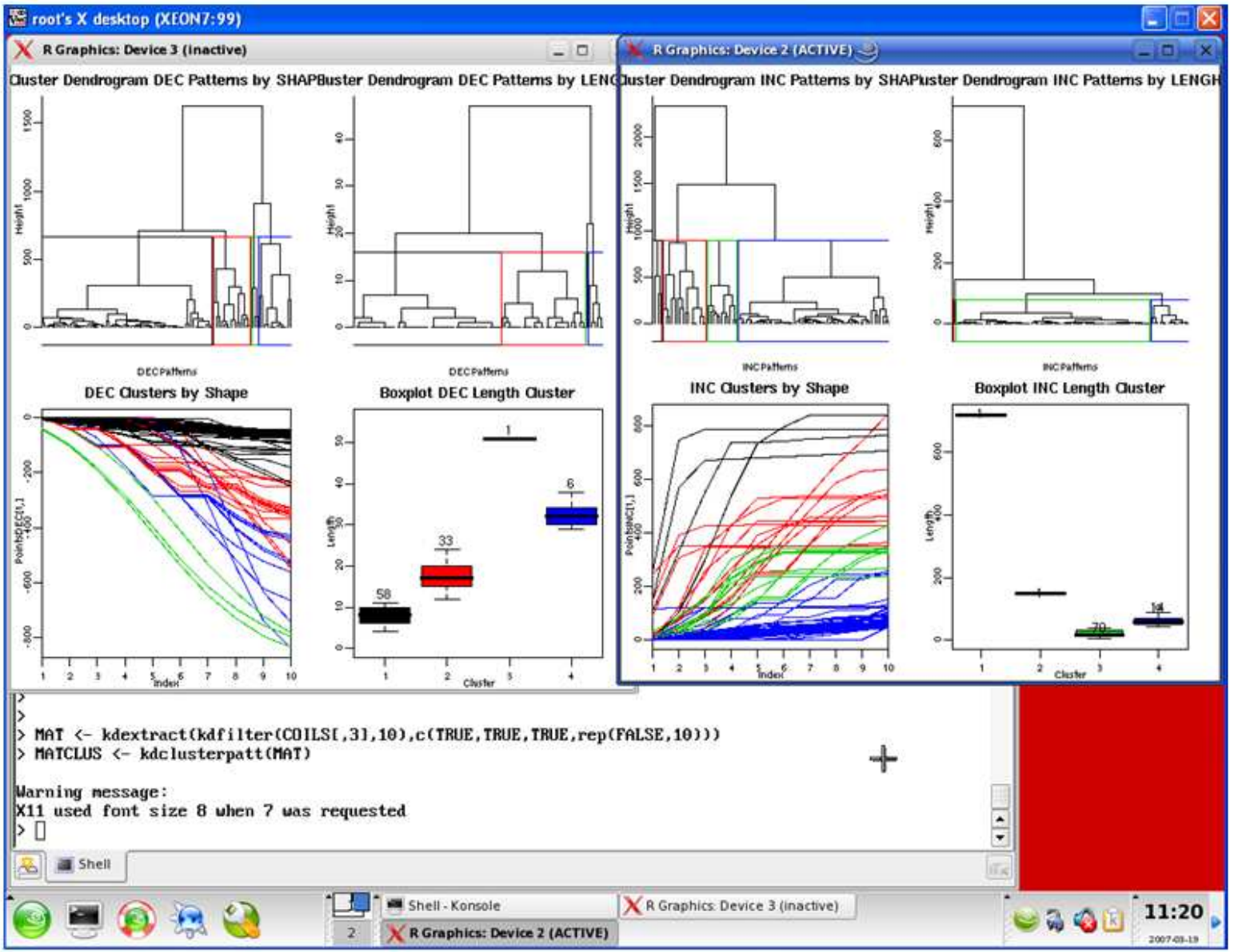
Calculo de la disimilitud por forma.



Función (II): *kdclusterpatt()*

Parámetros de Entrada:
MATPatt =Matriz con diferentes segmentos encontrados (proviene de kdextract() o kdmergepatt()).
INCParam= [1]. Número de subsegmentos a dividir.
[2]. Uso de dendrograma por grupos (g) o altura (h) para clasificar la forma.
[3]. Valor de corte del dendrograma para la forma.
[4]. Uso de dendrograma por grupos (g) o altura (h) para la longitud.
[5]. Valor de corte del dendrograma para la longitud.
DECParam=Igual que INCParam.
HORParam= [1]. Uso de dendrograma por grupos (g) o altura (h) para la longitud.
[2]. Valor de corte del dendrograma para la longitud.
Parámetros de Salida:
$MATSort: Patrones ordenados.
[col1=Posición, col2=Código del patrón.]
$FamilyINC: Patrones de la familia INC si hay.
$FamilyDEC: Patrones de la familia DEC si hay.
$FamilyHOR: Patrones de la familia HOT si hay.
$PointsINC: Puntos usados para agrupar los INC.
$GroupsINCShape[[X]]: Familia de patrones INC por forma.
$GroupsINCLength[[X]]: Familia de patrones INC por longitud.
$LengthPattINC[[X]]: Longitud de los patrones INC.
$PointsDEC: Puntos usados para agrupar los DEC.
$GroupsDECShape[[X]]: Familia de patrones DEC por forma.
$GroupsDECLength[[X]]: Familia de patrones DEC por longitud.
$LengthPattDEC[[X]]: Longitud de los patrones DEC.
$GroupsHORLength[[X]]: Familia de patrones HOR por longitud.
$LengthPattHOR[[X]]: Longitud de los patrones HOR.

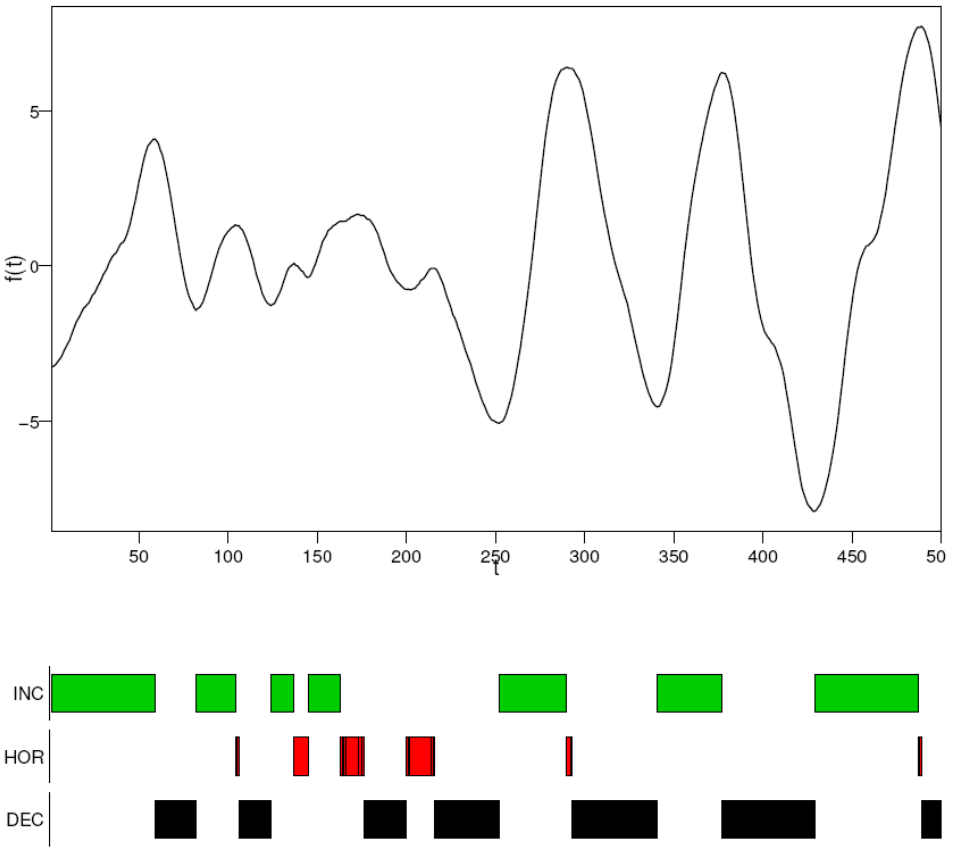
Función (y III): *kdclusterpatt()*



Función: *kdplotpatt()*

Función: *kdplotpatt(Pat, Ini=1, End=length(Pat$Serie))*
Resultado: Dibuja una serie temporal y visualiza los patrones encontrados. Válido solamente para patrones INC, DEC, y HOR.unde eventos parecidos y elimina los segmentos que no son importantes.

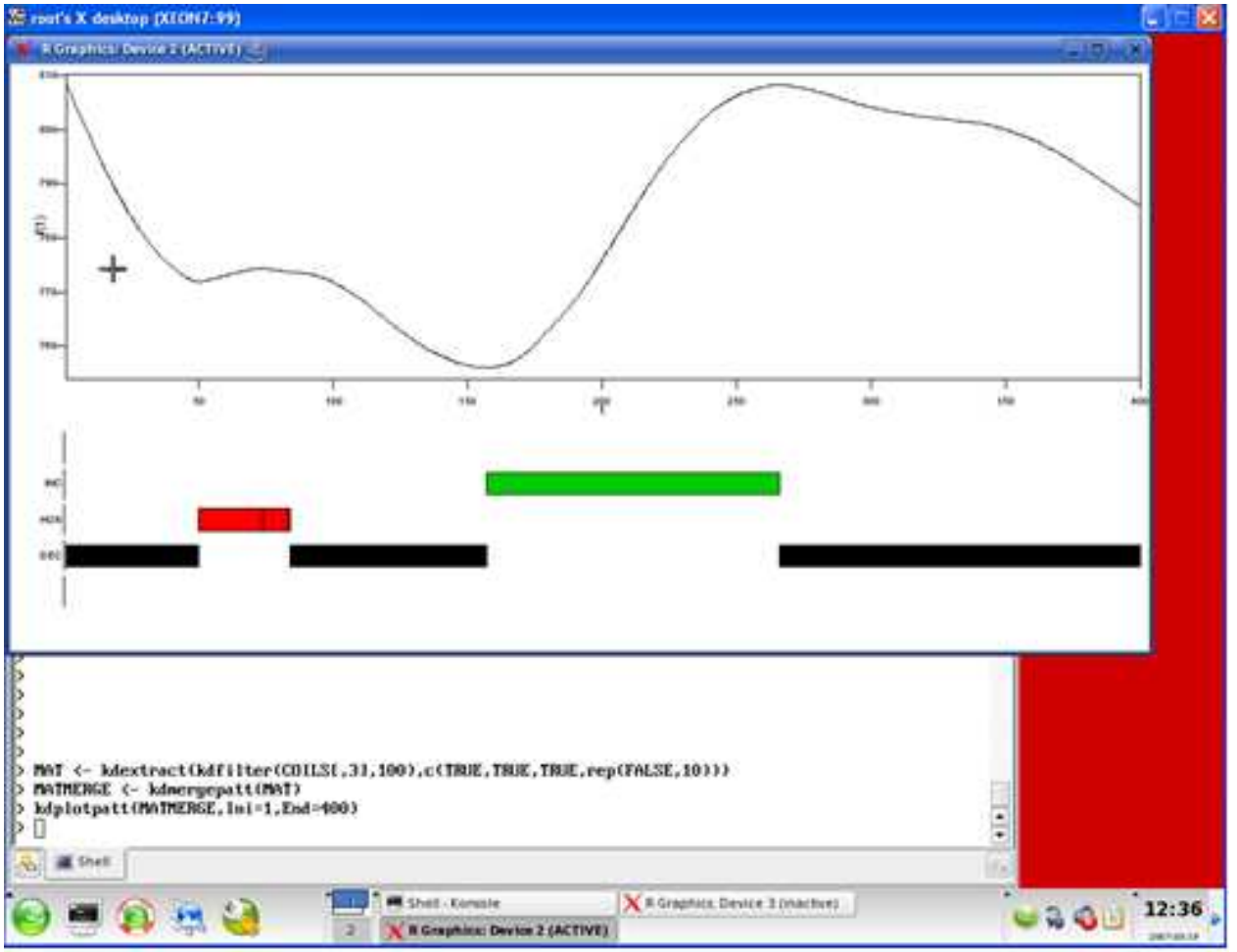
Parámetros de Entrada:
Pat=Matriz proveniente de kdextract()).
Ini= Punto Inicial.
Fin=Punto Final.
Parámetros de Salida:
Ninguno.



Función: *kdmergepatt()*

Función: *kkdmergepatt(MAT, Delete=c(0,0,1), MinHeight=c(0.05,-0.05))*
Resultado: Funde eventos parecidos y elimina los segmentos que no son importantes.

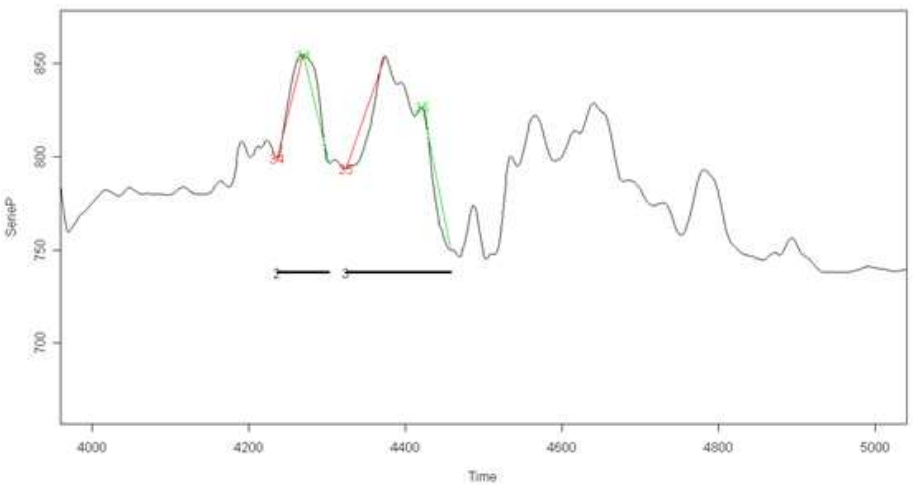
Parámetros de Entrada:
MAT=Matriz proveniente de kdextract()).
Delete= Borra eventos con longitud menos o igual a: x={IND, DEC, HOR}.
MinHeight= Mínima altura para ser considerado. {IND,DEC}.
Parámetros de Salida:
Matriz con los eventos fundidos o eliminados.



Función (I): *kdsearchpatt()*

Función: *kdsearchpatt(MAT, SubPatterns, WinW, namePatt, Plot=FALSE, SerieP=0,Xlim=c(1,length(SerieP)))*
Resultado: Permite extraer episodios complejos a partir de una matriz de eventos buscando secuencias de ellos dentro de una ventana de ancho configurable por el usuario.

Parámetros de Entrada:
MAT= Matriz obtenida de la unión de varias matrices de patrones de la función kdextractsubpatt.
SubPatterns =Secuencia de eventos. Permite el uso de sintaxis del comando “grep" en formato Perl.
WinW= Ancho de la ventana de búsqueda.
Plot= Si es TRUE dibuja la serie temporal y los patrones encontrados.
SerieP= Serie para dibujar si Plot=TRUE.
Xlim= Límites mínimo y máximo para dibujar.
namePatt= Nombre del patron encontrado.
Parámetros de Salida:
PATTERN=Matriz de patrones encontrados.
MAT2=Matriz de combinación de subpatrones encontrados.



Función (y II): *kdsearchpatt()*

Example

```
> library(KDSeries)
> library(KDSeries)
> data(COILS)
> # Preprocessing Temp 2
> DATA <- COILS$TMP2M
> plot.ts(DATA)
> DATA2 <- kdfilterremove(DATA,200)
> plot.ts(DATA2[1:1000])
> DATA3 <- kdfilter(DATA2,20)
> plot.ts(DATA3[1:1000])
> I <- kdextractsubpatt(DATA3,"I",pattRangeY=c(50,100),namePatt="INC1", threshold=c(780,800),level="+-")
> I$PATT[1:10,]
> segments(x0=I$PATT[,1],y0=I$PATT[,3],x1=I$PATT[,1]+I$PATT[,2], y1=I$PATT[,3]+I$PATT[,4],col="red")
> D <- kdextractsubpatt(DATA3,"D",pattRangeY=c(50,100), namePatt="DEC1")
> D$PATT[1:10,]
> segments(x0=D$PATT[,1],y0=D$PATT[,3],x1=D$PATT[,1]+D$PATT[,2],y1=D$PATT[,3]+D$PATT[,4],col="blue")
> H <- kdextractsubpatt(DATA3,"H",pattRangeX=c(20,1000),pattRangeY=c(0,10),namePatt="HOR1")
> H$PATT[1:10,]
> segments(x0=H$PATT[,1],y0=H$PATT[,3],x1=H$PATT[,1]+H$PATT[,2],y1=H$PATT[,3]+H$PATT[,4],col="magenta")
> MATTOTAL <- rbind(D$PATT,I$PATT, H$PATT)
> PATRONES <- kdsearchpatt(MATTOTAL,SubPatterns=c("INC1","DEC1"),WinW=150, namePatt="INC_DEC",
Plot=TRUE, SerieP=H$SerieP,Xlim=c(4000,5000))
```

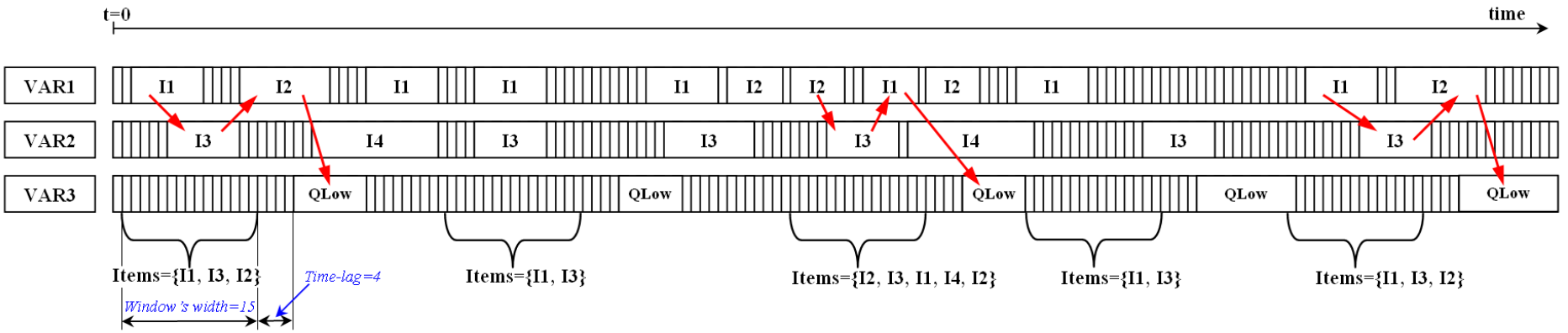
Función (y III): *kdsearchpatt()*

Example

```
> PATRONES
$PATTERN
PosP LongP namePatt
1 1497 101 INC_DEC
2 4236 67 INC_DEC
3 4324 134 INC_DEC
4 13202 75 INC_DEC
5 35407 127 INC_DEC
6 36786 136 INC_DEC
7 51662 76 INC_DEC
$MAT2
PosP LongP PosY AltP namePatt PosP.1 LongP.1 PosY.1 AltP.1
32 1497 43 759.9018 70.23312 INC1 1574 24 826.4428 -58.58808
34 4236 33 799.3201 55.68223 INC1 4269 34 855.0023 -58.39322
35 4324 50 793.9364 60.25899 INC1 4422 36 826.8562 -76.86638
38 13202 38 773.7443 54.69858 INC1 13240 37 828.4429 -65.27953
50 35407 79 762.2538 50.93327 INC1 35500 34 814.2895 -61.41658
53 36786 54 757.8621 71.04530 INC1 36872 50 824.0203 -62.84630
60 51662 24 784.2694 51.49803 INC1 51717 21 819.5083 -63.26126
namePatt.1 TOTALW
32 DEC1 101
34 DEC1 67
35 DEC1 134
38 DEC1 75
50 DEC1 127
53 DEC1 136
60 DEC1 76
```

Función (I): *kditems()*

Función: *kditems(MATEpisodes, Consequent, WinW, TimeLag=0, MinLen=1, MaxLen=Inf)*
Resultado: Realiza la búsqueda de los items (episodios o eventos) a partir de la base de datos de episodios más importantes y usando una ventana deslizante. El proceso consiste en situar la ventana un TimeLag previo a cada momento en que aparece el consecuente predefinido por el usuario y extraer los episodios o eventos que «caen» dentro de la ventana. El resultado es una BD transaccional como la que se muestra en el ejemplo.



Transactions Database:

1. (t=4) : {I1, I3, I2}
2. (t=41) : {I1, I3}
3. (t=79) : {I2, I3, I1, I4, ~~I2~~}
4. (t=105) :{I1, I3}
5. (t=134) :{I1, I3, I2}

Frequent Itemsets with Support > 50% (>2):

Level 1: {I1=5}, {I2=3}, {I3=5}
Level 2: {I1, I3}=5, {I1, I2}=3, {I2, I3}=3
Level 3: {I1, I2, I3}=3

Association Rules with Support > 50% (>2):

IF {*I1 AND I3*} THEN {*QLow*} (Support=5/5=100%)
IF {*I1*} THEN {*QLow*} (Support=5/5=100%)
IF {*I3*} THEN {*QLow*} (Support=5/5=100%)
IF {*I1 AND I2 AND I3*} THEN {*QLow*} (Support=3/5=60%)
IF {*I1 AND I2*} THEN {*QLow*} (Support=3/5=60%)
IF {*I2 AND I3*} THEN {*QLow*} (Support=3/5=60%)
IF {*I2*} THEN {*QLow*} (Support=3/5=60%)

Función (y II): *kditems()*

Función: *kditems(MATEpisodes, Consequent, WinW, TimeLag=0, MinLen=1, MaxLen=Inf)*
Resultado: Realiza la búsqueda de los items (episodios o eventos) a partir de la base de datos de episodios más importantes y usando una ventana deslizante. El proceso consiste en situar la ventana un TimeLag previo a cada momento en que aparece el consecuente predefinido por el usuario y extraer los episodios o eventos que «caen» dentro de la ventana.

Parámetros de Entrada:

MATEpisodes=Matriz con los episodios o eventos importantes.
Consequent=Consecuente de las reglas de asociación a buscar.
WinW=Ancho de la ventana deslizante.
TimeLag=Desplazamiento previo de la ventana con respecto a cada momento en que aparece el consecuente.
MinLen=*Número mínimo de items posible.*
MaxLen=*Máximo número de items posible.*

Parámetros de Salida:

KDITEMS:

$MATEPisodes: Matriz de entrada con los episodios o eventos importantes.
$Consequent: Consecuente definido en la entrada.
$WinW: Ancho de la ventana definido en la entrada.
$TimeLag: Desplazamiento definido en la entrada.
$MATTrans: Base de datos transaccional con los items que han aparecido previos al consecuente.
$NumConseq: Número de veces que aparece el consecuente.

Función: *kdrules()*

Función: *kdrules(KDITEMS, CutBy= «RelSupportWinRule», CutThreshold, OrderBy= «RelConfidenceWinRule», MinLen=1, MaxLen=Inf)*
Resultado: Extraer las reglas de asociación según el soporte.

Parámetros de Entrada:

KDITEMS=Data Frame con los resultados obtenidos de *kditems*().
CutBy=Extrae las reglas según: *«RelSupportWinRule»=Soporte relativo, «RelConfidenceWinRule»=Confianza relativa o «RelLiftWinRule»=Lift Relativo.*
CutThreshold=Umbral de corte para la extracción de las reglas.
OrderBy=Ordenar las reglas según:
«RelSupportWinRule»=Soporte relativo, «RelConfidenceWinRule»=Confianza relativa o «RelLiftWinRule»=Lift Relativo.
MinLen=*Número mínimo de items permitido en el antecedente de las reglas.*
MaxLen=*Máximo número de items permitido en el antecedente de las reglas.*

Parámetros de Salida:

MATRules=Data Frame con las reglas obtenidas.

An association rule according to time constraints is defined as:

$$X \Rightarrow Y \quad [WinW, TimeLag] \tag{5}$$

which corresponds to an antecedent *X* which occurs at a *TimeLag* before consequent *Y* within a time window of width *WinW*, and where:

$$RelSupportWinRule = |X| \; / \; |Y| \tag{6}$$

$$RelConfidenceWinRule = |X| \; / \; |X_{tot}| \tag{7}$$

with $|Y|$ being the number of times the consequent appears, which is equal to the number of transactions of database *T* and $|X|$ being the number of times the frequent itemset *X* appears in T and $|X_{tot}|$ being the number of times *X* appears in the time and within a time window of width *WinW*. Therefore, *RelSupportWinRule* shows the percentage of times *X* has occurred when *Y* has occurred, while *RelConfidenceWinRule* shows the percentage of times the rule has occurred when *X* has occurred.

Índice

Objetivo

Objetivo: identificar las causas que producen defectos en la capa de zinc de las bobinas tratadas en una línea de galvanizado en caliente.

Sections within the HDGL.

Area	*Description*
Pre-heating (PRE)	Strip preheating and cleaning area.
Heating: sub-areas 1 - 8 (01-08)	Sub-areas 1 - 8 of the heating area of the furnace, where the strip temperature is raised to around 850 ºC.
Holding: sub-areas 9 & 10 (09-10)	Area where the strip temperature is held around 850 ºC: Sub-areas 9 and 10.
Slow cooling (ENL)	Area where the strip is cooled slowly to 600-650 ºC.
Rapid cooling (ENR)	Area where the strip is cooled rapidly to 400-450 ºC.
Ageing (IGU)	Area where the strip temperature is equalized before being dipped into the zinc pot.
Zinc pot (BATH)	Molten zinc bath dipping area.
Equalizer (TRM)	Equalizer area.
General (-)	Parameters which are constant throughout the HDGL.

Base de Datos

- 723 bobinas de un nuevo tipo de acero, 39 con defectos (5.4 %).
- Selección de las variables más importantes en cada una de las zonas de la planta:
  - Temperaturas del horno.
  - % de O2 y H2.
  - Temperaturas de la banda en diversos puntos.
  - Temperatura baño de zinc.
  - Velocidad de la banda.
  - etc.
- Mediciones del proceso cada 100 metros de banda.
- Más de 50.000 registros.

Main groups of variables.

Name	*Type*	*Description*
CODE	nominal	Coil code.
CON_H2	numeric	H_2 concentration in the air in area xx.
CON_O2	numeric	O_2 concentration in the air in area xx.
TMP_PR	numeric	Dew point temperature in that area.
Zxx_TMP	numeric	Temperature of area xx.
TMP_Pxx	numeric	Strip temperature measured with pyrometer xx in each process area.
THICK, WIDTH	numeric	Steel strip thickness & width.
SPD	numeric	Strip feed rate (m/min).
BATH_TMP	numeric	Zinc bath temperature.
CMP_BATH	numeric	Chemical composition of zinc bath. Percentages of main chemical elements in bath.
CMP_STEEL	numeric	Chemical composition of the steel strip. Percentages of main chemical elements.
ADHERENCE	binary	Whether zinc coating adherence is within tolerances (0) or not (1).

Proceso de Extracción de Conocimiento

1. Filtrado de las Series Temporales.
2. Búsqueda de eventos importantes: incrementos o decrementos bruscos de temperatura, velocidad, etc.; valores por encima o debajo de un umbral definido por el experto, etc.
3. Búsqueda de reglas.

First 20 rules extracted (arranged by support).

Num.	*Items*	*RelSupportWinRule*	*RelConfidenceWinRule*
1	{Z06_TMP_BELOW}	0.77	0.53
2	{TMP_P02_BELOW}	0.69	0.45
3	**{BATH_TMP_BELOW SPD_DEC Z06_TMP_BELOW}**	**0.67**	**0.76**
4	{BATH_TMP_BELOW Z06_TMP_BELOW}	0.67	0.60
5	{BATH_TMP_BELOW SPD_DEC}	0.67	0.62
6	{SPD_DEC Z06_TMP_BELOW}	0.67	0.62
7	{Z06_TMP_BELOW TMP_P02_BELOW }	0.67	0.45
8	{SPD_DEC}	0.67	0.60
9	{BATH_TMP_BELOW}	0.67	0.72
10	{BATH_TMP_BELOW SPD_DEC Z06_TMP_BELOW TMP_P02_BELOW}	0.62	0.55
11	{BATH_TMP_BELOW SPD_DEC Z06_TMP_BELOW SPD_NOT_HOR}	0.62	0.55
12	{BATH_TMP_BELOW SPD_NOT_HOR MP_P02_BELOW		
13	{BATH_TMP_BELOW Z06_TMP_BELOW SPD_NOT_HOR TMP_P02_BELOW}	0.62	0.72
14	{BATH_TMP_BELOW Z06_TMP_BELOW SPD_NOT_HOR}	0.62	0.73
15	**{BATH_TMP_BELOW SPD_NOT_HOR TMP_P02_BELOW}**	**0.62**	**0.80**
16	{BATH_TMP_BELOW SPD_DEC SPD_NOT_HOR}	0.62	0.58
17	{SPD_DEC Z06_TMP_BELOW SPD_NOT_HOR TMP_P02_BELOW}	0.62	0.55
18	{SPD_DEC Z06_TMP_BELOW SPD_NOT_HOR}	0.62	
19	{SPD_DEC SPD_NOT_HOR TMP_P02_BELOW}	0.62	0.65
20	{Z06_TMP_BELOW SPD_NOT_HOR TMP_P02_BELOW}	0.62	0.70

Índice

Conclusiones

- La librería KDSeries permite realizar de una forma sencilla todas las fases necesarias para el proceso KDD propuesto: preprocesado, búsqueda de máximos y mínimos, búsqueda de eventos significativos, combinación de eventos simples en episodios más complejos, búsqueda de items frecuentes fijado el consecuente y selección de reglas de asociación.

Conclusiones

- La librería KDSeries permite realizar de una forma sencilla todas las fases necesarias para el proceso KDD propuesto: preprocesado, búsqueda de máximos y mínimos, búsqueda de eventos significativos, combinación de eventos simples en episodios más complejos, búsqueda de items frecuentes fijado el consecuente y selección de reglas de asociación.
- Se está aplicando a procesos industriales, relaciones entre valores bursátiles, identificación de causas que producen enfermedades en la vid, etc.

Conclusiones

- La librería KDSeries permite realizar de una forma sencilla todas las fases necesarias para el proceso KDD propuesto: preprocesado, búsqueda de máximos y mínimos, búsqueda de eventos significativos, combinación de eventos simples en episodios más complejos, búsqueda de items frecuentes fijado el consecuente y selección de reglas de asociación.
- Se está aplicando a procesos industriales, relaciones entre valores bursátiles, identificación de causas que producen enfermedades en la vid, etc.
- Las posibilidades futuras son muy prometedoras. Existen muchas posibilidades de mejora y ampliación de KDSeries.

Conclusiones

- La librería KDSeries permite realizar de una forma sencilla todas las fases necesarias para el proceso KDD propuesto: preprocesado, búsqueda de máximos y mínimos, búsqueda de eventos significativos, combinación de eventos simples en episodios más complejos, búsqueda de items frecuentes fijado el consecuente y selección de reglas de asociación.
- Se está aplicando a procesos industriales, relaciones entre valores bursátiles, identificación de causas que producen enfermedades en la vid, etc.
- Las posibilidades futuras son muy prometedoras. Existen muchas posibilidades de mejora y ampliación de KDSeries.
- Actualmente KDSeries está en fase de desarrollo.

Índice

Lecturas Recomendadas (I)

Conti, Dante; Martinez de Pisón, F.J. and Pernía, A.
Finding temporal associative rules in financial time-series: A case of study in Madrid Stock Exchange (IGBM)
Advances in Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS 10). 9th WSEAS International Conference, pp. 60-68, December, 2010

Conti, Dante.
Obtención de conocimiento oculto mediante reglas de asociación en series temporales: análisis y mejora de procesos.
Tesis Doctoral, Universidad de La Rioja, Servicio de Publicaciones, 2010

Fink, E. and Pratt, K.B.
Indexing of compressed time series, in: Last, M., Kandel, A., Bunke, H. (Eds.), Data Mining In Time Series Databases
World Scientific, New York, 51-78, 2004

Lecturas Recomendadas (y II)

Martinez de Pisón, F.J.; Conti, Dante and Pernía, A.
Temporal association rules mining: a heuristic methodology applied to Time Series Databases (TSDBs)
Advances in Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS 10). 9th WSEAS International Conference, pp. 69-74, December, 2010

Martinez de Pisón, F. J.; Sanz, A.; Mtnez de Pisón, E.; Jiménez, E. and Conti, D.
Mining association rules from time series to explain failures in a hot-dip galvanizing steel line
Computer and Industrial Engineering, 2011 (en revisión)

Martinez de Pisón, F.J.; Ordieres, J.; Pernía, A. and Alba, F.
Minería de Datos en Series Temporales para la Búsqueda de Conocimiento Oculto en Históricos de Procesos Industriales
III Taller de Minería de Datos y Aprendizaje (TAMIDA, CEDI 2005), Granada, 2005