Package 'forecastUNAL'

November 6, 2025

Title Ajuste de Modelos para Análisis de Series de Tiempo

Version 0.0.0.9000

Description Implementación de modelos y análisis de series de tiempo de la asignatura Estadística III (3009137) del Departamento de Estadística de la Universidad Nacional de Colombia, Sede Medellín.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports car,
fANCOVA,
forecast,
ggplot2,
graphics,
gridExtra,
lmtest,
stats,
strucchange

URL https://github.com/cmlopera/forecastUNAL

BugReports https://github.com/cmlopera/forecastUNAL/issues

Depends R (>= 3.5)

LazyData true

Contents

`BPLBtest` *Función* `BPLBtest()`

Description

Realiza pruebas Ljung-Box y Box-Pierce para `m=6, 12, 18,...,[maxlag/6]`, con `maxlag` el máximo orden de rezagos. Usa la función `Box.test()` de la librería `stats`.

Usage

```
BPLBtest(serie, maxlag, type = c("Ljung", "Box"))
```

Arguments

`serie`	Vector de valores del proceso sobre el que se quiere evaluar incorrelación con tests Ljung-Box o Box-Pierce.
`maxlag`	Entero indicando el máximo orden de rezago temporal hasta el cual se prueba la significancia de la función de autocorrelación.
`type`	Cadena de caracteres para indicar el tipo de prueba, `"Ljung"` (por defecto) para las pruebas Ljung-Box y `"Box"` para las pruebas Box-Pierce.

Value

Un objeto `data.frame` con tres variables:

- `Xsquared` los valores de los estadísticos de prueba para `m=6, 12,...,[maxlag/6]`.
- `df` los grados de libertad del estadístico de prueba, corresponden a los valores de `m=6, 12,...,[maxlag/6]`.
- `pvalue` los valores P para las pruebas con `m=6, 12,...,[maxlag/6]`.

Examples

```
#Simulando serie de un proceso estacionario de media cero con variables correlacionadas
zt <- ts(arima.sim(n=300,list(ar=0.7),sd=4000),freq=1,start=1)
#Aplicando pruebas Ljung-Box con máximo orden de correlación de 36
BPLBtest(serie=zt,maxlag=36)
#Aplicando pruebas Box-Pierce con máximo orden de correlación de 36
BPLBtest(serie=zt,maxlag=36,type="Box")
#Simulando serie de un proceso ruido blanco gaussiano
zt <- ts(rnorm(n=300,sd=4000),freq=1,start=1)
#Aplicando pruebas Ljung-Box con máximo orden de correlación de 36
BPLBtest(serie=zt,maxlag=36)
#Aplicando pruebas Box-Pierce con máximo orden de correlación de 36
BPLBtest(serie=zt,maxlag=36,type="Box")
```

CEMENTO *Producción trimestral de cemento Portland.*

Description

Producción trimestral de cemento Portland 1956:Q1 - 1994:Q3, en miles de toneladas.

Usage

```
CEMENTO
```

Format

CEMENTO**:**
Un vector con 155 observaciones.

---

critinfresid *Función* critinfresid()

Description

Calcula criterios de información como en Diebold(2001).

Usage

```
critinfresid(residuales, npar)
```

Arguments

residuales	Un vector de diferencias entre las observaciones y su ajuste, en la escala original de los datos.
npar	Un valor entero igual al número de parámetros del modelo ajustado.

Value

Un vector de longitud tres con p (el número de parámetros) AIC (criterio Akaike) y BIC (criterio Bayesiano)

Examples

```
#Simulando serie aditiva con modelo de tendencia lineal y errores iid normales
t <- 1:300
yt <- 3500+200*t+rnorm(300,sd=4000)
yt <- ts(yt,freq=12,start=c(1980,1))
#Ajuste modelo de tendencia lineal
mrl <- lm(yt~t)
summary(mrl)
ythat <- ts(fitted(mrl),freq=12,start=c(1980,1))
#Gráfico serie observada y su ajuste
plot(yt)
lines(ythat,col=2)
#Calculando AIC y BIC sobre modelo ajustado
p <- length(coef(mrl)) #Número de parámetros
critinfresid(residuales=residuals(mrl),npar=p)

#Simulando serie multiplicativa con modelo log lineal y errores correlacionados de media cero
t <- 1:540
logYt <- 2.5+0.0105*t+arima.sim(n=length(t),list(ar=0.6),sd=0.27)
Yt <- ts(exp(logYt),freq=1,start=1)
#Ajuste modelo log lineal con errores iid normales
modelo <- lm(log(Yt)~t)
summary(modelo)
#Valores estimados de la serie en escala original de los datos
Ythat <- ts(exp(fitted(modelo))*exp(summary(modelo)$sigma^2/2),freq=1,start=start(Yt))
#Gráfico serie observada y su ajuste
plot(Yt)
lines(Ythat,col=2)
legend("topleft",legend=c("Serie","Ajuste"),col=c(1,2),lty=1)
#Calculando AIC y BIC sobre modelo ajustado
seudores <- Yt-Ythat #Seudo residuos para calcular criterios de información
p <- length(coef(modelo)) #Número de parámetros
critinfresid(residuales=seudores,npar=p)
```

DatosChippy *Datos* Chippy *(datos simulados)*

Description

Un conjunto de datos simulados con tendencia `T(t)=sin(cos(t)*exp(-t/2))`.

Usage

```
DatosChippy
```

Format

`DatosChippy`**:**

Un data frame con 210 filas y 2 columnas:

Chippy Valor de la curva de tendencia en cada tiempo.

serie Valor de la serie en cada tiempo.

DescompLoessv02 *Función* DescompLoessv02()

Description

Estima y predice serie estacional aditiva y multiplicativa combinando filtro de la descomposición clásica y la regresión loess.

Usage

```
DescompLoessv02(
  serie.ajuste,
  tipo.descomp,
  grado = 1,
  criterio,
  h = 1,
  level = 0.95
)
```

Arguments

serie.ajuste	Un objeto ts de serie de tiempo univariada con los valores de la serie a ajustar.
tipo.descomp	Tipo de descomposición "additive" ó "multiplicative", en el último caso es bajo modelo parcialmente multiplicativo.
grado	Grado del polinomio local a ser ajustado sobre la serie previamente desestacionalizada con la estacionalidad estimada con el filtro de la descomposición clásica de la función R decompose(), bajo el modelo aditivo o multiplicativo, según valor del argumento tipo.descomp. Su valor por defecto es 1.
criterio	Tipo de criterio de información "aicc" ó "gcv", a usar para la elección del parámetro de suavizamiento en la rutina de loess óptimo mediante la función R loess.as() de la librería fANCOVA. Por defecto es "aicc".
h	Número de predicciones a calcular después del ajuste. Por defecto es 1.
level	Nivel de confianza de los intervalos de predicción. Por defecto 0.95.

Value

Una lista con los siguientes objetos:

- deltasi Vector con las estimaciones de los efectos estacionales.
- alfa.optim Valor del parámetro de suavizamiento loess óptimo.
- nep El número de parámetros equivalentes loess.
- p El total de parámetros aproximados en el ajuste de tendencia y estacionalidad.
- St Objeto ts de la estimación de la componente estacional.
- Tt Objeto ts de la estimación de la tendencia loess.
- ytd La serie desestacionalizada.
- fitted Objeto ts de la serie estimada combinando aditiva o multiplicativamente, según el tipo de descomposición, las estimaciones de la tendencia y de la estacionalidad.

- `forecast` Objeto `ts` multivariado con las predicciones puntuales y por intervalo para los `h` períodos pedidos.
- `residuals` Objeto `ts` con los residuos de ajuste obtenidos como la diferencia entre los valores observados y los ajustados para la serie.
- `MSE` Estimación de la varianza del error de ajuste.

Examples

```
#Serie de tiempo con inicio en enero de 1965
ventascompanyX

m <- 6 #tamaño muestra de validación cruzada
n <- length(ventascompanyX)-m #tamaño muestra de ajuste
t <- 1:n #índice de tiempo en periodos de ajuste

#Serie de tiempo en periodos de ajuste
yt <- ts(ventascompanyX[t],frequency=12,start=c(1965,1))

#Ajuste por descomposición multiplicativa y tendencia loess lineal
#se usa criterio AICC para el parámetro de suavizamiento loess
ajusteDLL1 <- DescompLoessv02(serie.ajuste=yt,tipo.descomp="multiplicative",
                              grado=1,criterio="aicc",h=m,level=0.95)

#Gráfico de la serie y su ajuste
plot(ventascompanyX)
lines(fitted(ajusteDLL1),col=2,lwd=2)
legend("topleft",legend=c("Serie","Ajuste"),col=c(1,2),lty=1)

#Pronósticos puntuales y por IPs en los períodos de validación cruzada
ajusteDLL1$forecast

#Estimación de sigma
sigma <- sqrt(ajusteDLL1$MSE)

#residuos del ajuste
diagresTSModel(modelo=ajusteDLL1,RMSE=sigma,all=FALSE,single=FALSE)
```

`diagresTSModel` *Función* `diagresTSModel()`

Description

Genera gráficos de residuos sobre modelos de series de tiempo. Adicionalmente, puede generar la ACF, PACF muestrales, el gráfico de probabilidad normal, test Shapiro-Wilks y Tests Ljung-Box.

Usage

```
diagresTSModel(modelo, RMSE, all = FALSE, single = TRUE, lagmax = 36)
```

Arguments

modelo	Un objeto R con el ajuste de un modelo de serie de tiempo. Puede ser de clase lm, nls, Arima, HoltWinters, entre otros manejados en la asignatura.
RMSE	Un valor numérico correspondiente a una estimación de la desviación estándar del error de ajuste. Se requiere para las líneas horizontales pasando por -2*RMSE y +2*RMSE.
all	Argumento lógico. Por defecto es FALSE indicando que solo los gráficos de residuos vs. tiempo y vs. valores ajustados serán producidos. Si es TRUE, exhibe además los gráficos de ACF y PACF muestrales, de probabilidad normal, test Shapiro-Wilks y Tests Ljung-Box para m=6, 12, . . . , hasta máximo múltiplo de 6 menor o igual a lagmax.
single	Un valor lógico. Por defecto es TRUE indicando que cada gráfica debe ser presentada en ventana independiente. Si se ajusta a FALSE y all=FALSE, los dos gráficos de residuos son presentados en una sola ventana en un arreglo de 2x1, y si all=TRUE, los cinco gáficos que se generan son presentados en una sola ventana, en un arreglo de 3x2.
lagmax	Valor numérico indicando el máximo orden de rezago a considerar en gráficos de la ACF y PACF muestrales y en las pruebas Ljung-Box. Por defecto su valor es 36.

Value

La función no genera ningún tipo de objeto R sobre el cual luego pueda obtenerse algún valor o resultado, solo las gráficas pedidas y resultados de pruebas exhibidas en la consola R.

Examples

```
library(forecast)
library(lmtest)
library(car)
datos <- ts(CEMENTO,fre=4,start=c(1956,1))
n <- length(datos)-4
t <- 1:n
yt <- ts(datos[t],frequency=4,start=c(1956,1))
poli <- Mipoly(tiempo=t,grado=3)
trimestre <- seasonaldummy(yt)
X <- data.frame(poli,trimestre) #La matriz de los predictores en el ajuste

#Modelo log cúbico estacional con indicadoras (nivel de ref.=Q4)
modelo1 <- lm(log(yt)~.,data=X)
summary(modelo1)
diagresTSModel(modelo=modelo1,RMSE=summary(modelo1)$sigma)
diagresTSModel(modelo=modelo1,RMSE=summary(modelo1)$sigma,all=TRUE,single=TRUE)
diagresTSModel(modelo=modelo1,RMSE=summary(modelo1)$sigma,all=TRUE,single=FALSE)
diagresTSModel(modelo=modelo1,RMSE=summary(modelo1)$sigma,all=FALSE,single=FALSE)

#Modelo exponencial cúbico estacional con indicadoras (nivel de ref.=Q4)
modelo2b <- regexponencialv02(respuesta=yt,data=X)
summary(modelo2b)
diagresTSModel(modelo=modelo2b,RMSE=summary(modelo2b)$sigma,all=TRUE,single=FALSE)

#Modelo log cúbico estacional con indicadoras (nivel de ref.=Q4) y error AR(6)
modelo2 <- Arima(log(yt),order=c(6,0,0),xreg=as.matrix(X),method="ML")
```

```
df2 <- n-length(coef(modelo2)[coef(modelo2)!=0])
coeftest(modelo2,df=df2)
diagresTSModel(modelo=modelo2,RMSE=sqrt(modelo2$sigma2),all=TRUE,single=FALSE)

#Suavizamiento Holt-Winters multiplicativo con predicción de h=4 trimestres
modelo3 <- SuavizamientoEstacional(yt,seasonal="multiplicative",h=4)
diagresTSModel(modelo=modelo3,RMSE=sqrt(modelo3$MSE),all=TRUE,single=FALSE)

#Combinación filtros de descomposición multiplicativa y
#Loess cuadrático óptimo con span según criterio AICC
modelo4 <- DescompLoessv02(serie.ajuste=yt,tipo.descomp="multiplicative",
           grado=2,criterio="aicc",h=4,level=0.95)
diagresTSModel(modelo=modelo4,RMSE=sqrt(modelo4$MSE),all=TRUE,single=FALSE)
```

`estimarecursiva` *Funcion* `estimarecursiva()`

Description

Realiza la estimación recursiva de un modelo de regresión lineal, obteniendo además los residuos recursivos, la gráfica `CUSUMt-Recursivo` y el test `CUSUMt-Recursivo` mediante las funciones `recresid()`, `efp()` y `sctest()`, respectivamente, de la librería `strucchange`.

Usage

```
estimarecursiva(
  respuesta,
  dataX,
  min.n,
  nrow1 = 2,
  ncol1 = 1,
  nrow2 = 2,
  ncol2 = 2,
  plot.recursive = TRUE
)
```

Arguments

`respuesta`	Un vector numérico o serie de tiempo con los valores de la respuesta del modelo de regresión lineal múltiple.
`dataX`	Un `data.frame` cuyas columnas son los predictores del modelo de regresión lineal múltiple, el número de filas debe ser igual a la longitud de la variable `respuesta`.
`min.n`	Un escalar para indicar el mínimo tamaño de muestra con el que debe iniciar la estimación recursiva, el cual no puede ser inferior a `ncol(dataX) + 2`.
`nrow1`, `ncol1`	Valores enteros que definen el número de `filas, columnas` en el diseño de la ventana gráfica para representar las gráficas de los residuales recursivos y del estadístico `CUSUMt-Recursivo`. Si ambos valores son iguales a 1, cada gráfica sale en una ventana por separado.

nrow2, ncol2 Valores enteros que definen el número de filas, columnas en el diseño de la ventana gráfica para representar las gráficas de las estimaciones recursivas de los parámetros cuando el argumento plot.recursive = TRUE. Cuando el número de gráficas a ser representadas supera al número de celdas definido en el diseño de la ventana de salida, entonces se generarán las ventanas gráficas adicionales que sean necesarias con el mismo diseño.

plot.recursive Argumento lógico, sus valores posibles son TRUE (por defecto) y FALSE, para indicar si se desean o no las gráficas de las estimaciones recursivas de los parámetros del modelo de regresión lineal múltiple.

Value

La función muestra en la consola R la estimación del modelo global sobre el total de observaciones leídas en el vector respuesta y los resultados del test CUSUMt-recursivo. Genera también el gráfico de los residuos recursivos y el gráfico del estadístico CUSUMt-recursivo. Además, produce un objeto tipo lista con los siguientes componentes:

- n Una matriz de una sola columna con sus filas siendo los valores de tamaño de muestra desde min.n hasta length(respuesta).
- estimacion_recursiva Un arreglo de matrices, donde cada matriz tiene tres columnas: la estimación de los parámetros y los límites inferior y superior de confianza del 95% de cada parámetro. El número de matrices es igual al total de tamaños de muestras entre min.n y length(respuesta).
- ajusteglobal Un objeto tipo lm() del ajuste de la respuesta vs. los predictores en dataX.
- resid_recursivos Un vector de residuos recursivos.
- test_CUSUM Los resultados (estadístico de prueba y valor-P) del test CUSUMt-recursivo.

Examples

```
library(forecast)
datos <- ts(CEMENTO,frequency=4,start=c(1956,1)) # serie CEMENTO con todos los datos
#Variables para modelo log cuadrático estacional con indicadoras, nivel referencia trimestre 4
t <- 1:length(datos) #Definiendo índice de tiempo para todos los datos
poli <- Mipoly(tiempo=t,grado=2)
trimestre <- seasonaldummy(datos) #Creando variables indicadoras, nivel de  referencia trimestre 4
X <- data.frame(poli,trimestre) #Matriz de predictores
resultados <- estimarecursiva(respuesta=log(datos),dataX=X,min.n=24,
              nrow1=2,ncol1=1,nrow2=2,ncol2=2,plot.recursive=TRUE)
```

Filtrolineal *Función* Filtrolineal()

Description

Ajusta y pronostica series no estacionales a través de filtros lineales basados en medias móviles simples unilaterales o bilaterales, usando la estrategia circular.

Usage

```
Filtrolineal(serie, pesos.wi, tipo, h = 1, level = 0.95)
```

Arguments

serie	Un objeto tipo ts con los valores de la serie a ajustar y pronosticar.
pesos.wi	Vector de pesos de las medias móviles para la estimación del nivel. Para medias móviles simples unilaterales debe ser de longitud m+1, para i=0,1,...,m, y para bilaterales debe ser de longitud 2m+1, para i=-m,...,m.
tipo	El tipo de medias móviles para la estimación del nivel, tipo=1 para medias móviles simples unilaterales, tipo=2 para medias móviles simples bilaterales.
h	Número de períodos futuros a pronosticar, por defecto es 1.
level	El nivel de confianza para los intervalos de pronóstico, por defecto es 0.95.

Value

Una lista con los siguientes componentes:

- nivel Objeto tipo ts de las medias móviles calculadas con la estrategia circular.
- fitted Objeto tipo ts con los valores ajustados de la serie.
- residuals Objeto tipo ts con los valores de la serie de tiempo de los residuos del ajuste.
- forecast Objeto tipo mts con los valores de los pronósticos puntuales y por Intervalos para los h períodos.
- RMSE Estimacion de la raíz cuadrada del MSE del ajuste.

Examples

```
#Serie de la tasa de variación trimestral del PIB in UK 1990:Q1 a 2004:Q1
datos <- ts(GDPchange,freq=4,start=c(1990,1))
#gráfico de la serie
plot(datos,type="b",pch=19,lwd=3)

m <- 3 #tamaño muestra de validación cruzada
n <- length(datos)-m #tamaño muestra de ajuste
t <- 1:n
yt <- ts(datos[t],freq=4,start=c(1990,1))
tnuevo <- (n+1):length(datos)
ytnuevo <- ts(datos[tnuevo],freq=4,start=c(2003,3)) #pronósticos inician en Q3-2003

#Ajuste y pronósticos con Filtro de Henderson de 2m+1=7 términos
w7 <- c(-0.059,0.059,0.294,0.413,0.294,0.059,-0.059) #Vector de pesos wi
FH7 <- Filtrolineal(serie=yt,pesos.wi=w7,tipo=2,h=m) #Obtención de ajustes y pronósticos
ythatFH7 <- fitted(FH7);ythatFH7 #Valores ajustados

#Serie y su ajuste
plot(datos,lwd=2)
lines(ythatFH7,col=2,lwd=2)
legend("bottomright",legend=c("Obs.","FH7"),col=c(1,2),lty=1,cex=2,lwd=2)

#RMSE de ajuste
RMSEFH7 <- FH7$RMSE
RMSEFH7

#pronósticos puntuales y por IPs del 95%
pronFH7 <- FH7$forecast
pronFH7
#Precisión pronósticos puntuales
```

```
forecast::accuracy(FH7$forecast[,1],ytnuevo)
#Precisión pronósticos por IPs
precisintervals(real=ytnuevo,LIP=FH7$forecast[,2],LSP=FH7$forecast[,3])

#Gráficos de residuos de ajuste
diagresTSModel(modelo=FH7,RMSE=RMSEFH7,all=FALSE,single=TRUE)
```

`GDPchange` *Variación porcentual del PIB de UK.*

Description

Variación porcentual trimestral del PIB en el Reino Unido, 1990:1Q - 2004:1Q

Usage

```
GDPchange
```

Format

`GDPchange`**:**
Un vector con 57 observaciones.

`interpdeltas` `Función interpdeltas()`

Description

Extrae las estimaciones y construye gráfico de los efectos estacionales en modelos de regresión global estacionales que usan indicadoras con nivel de referencia el `s` ésimo período calendario. Particularmente en modelos polinomiales estacionales con error RB y con error ARMA, modelos log polinomiales estacionales con error RB y con error ARMA y modelos exponenciales polinomiales estacionales con error RB. En los modelos polinomiales estacionales, bien sea con error RB o ARMA, extrae los valores estimados de los parámetros asociados a las variables indicadoras y crea la gráfica de estos valores vs. período calendario, asignando valor de cero en el último período. En los modelos log polinomiales estacionales con error RB o ARMA y en los modelos exponenciales polinomiales estacionales con error RB, extrae los valores estimados de los parámetros asociados a las variables indicadoras, pero es necesario informar que estos modelos son multiplicativos para que sean exponenciadas estas estimaciones, las multiplique por 100% y las grafique contra el período calendario, asignando el valor de 100% al último periodo del año.

Usage

```
interpdeltas(
  modelo,
  ordenp = 0L,
  ordenq = 0L,
  ordenP = 0L,
  ordenQ = 0L,
  gradopoly,
  aditivo = TRUE,
  plotit = TRUE
)
```

Arguments

modelo	El nombre del objeto R con el modelo ajustado. Objetos tipo lm(), nls() o Arima().
ordenp, ordenq, ordenP, ordenQ	Argumentos de valor entero, para especificar los valores de los ordenes p, q, P, Q de modelos ARMA(p,q)(P,Q)[s]. Por defecto están fijos en 0, es decir, asumiendo error RB. Si el error es un AR(p) el usuario debe especificar el valor de p; si el error es MA(q) debe especificar el valor de q; si es ARMA(p,q) debe especificar p y q, etc.
gradopoly	El grado del polinomio global. Siempre debe ser especificado.
aditivo	Argumento lógico (TRUE, FALSE), por defecto es TRUE, este valor debe usarse en modelos polinomiales estacionales con indicadoras y error RB o error ARMA. En modelos log polinomiales estacionales con error RB o error ARMA, o modelos exponenciales polinomiales estacionales con error RB, este argumento debe especificarse igual a FALSE.
plotit	Argumento lógico (TRUE, FALSE), por defecto es TRUE indicando que se desea la gráfica de los efectos estacionales para su interpretación en la escala de la serie, según si el modelo es aditivo o multiplicativo.

Value

Con plotit=TRUE, despliega la gráfica correspondiente y en la consola R el vector de parámetros estacionales estimados (modelos aditivos) o de los valores exponenciados de estas estimaciones y multiplicados por 100% (modelos multiplicativos). Además, crea un objeto lista con los siguientes componentes:

- periodo Un vector de valores enteros de 1 a s (la longitud del año calendario).
- deltasi Un vector con las estimaciones de los parámetros asociados a las indicadoras y adiciona el valor de cero al final. Este vector aparece cuando aditivo=TRUE.
- expdeltasi100 Un vector con los valores exponenciados de las estimaciones de los parámetros asociados a las indicadoras, multiplicados por 100 y adiciona el valor de 100 al final. Este vector aparece cuando aditivo=FALSE.

Examples

```
library(forecast)
#Ej. 1 Modelo polinomial estacional con indicadoras (nivel ref=12)
datos1 <- salario
n1 <- length(datos1)-12
t1 <- 1:n1
yt1 <- ts(datos1[t1],freq=12,start=start(datos1))
poli1 <- Mipoly(tiempo=t1,grado=4)
mes <- seasonaldummy(yt1)
X1 <- data.frame(poli1,mes)
modelo1 <- lm(yt1~.,data=X1)
summary(modelo1)
#Extrayendo y graficando las estimaciones de los coeficientes de regresión delta_i
interpdeltas(modelo1,gradopoly=4,aditivo=TRUE,plotit=TRUE)

#Ej. 2 log cúbico estacional con indicadoras (nivel ref=Q4), error ARMA(1,2)x(0,1)[4]
datos2 <- ts(CEMENTO,fre=4,start=c(1956,1))
n2 <- length(datos2)-4
t2 <- 1:n2
```

```
yt2 <- ts(datos2[t2],freq=4,start=start(datos2))
poli2 <- Mipoly(tiempo=t2,grado=3)
trimestre <- seasonaldummy(yt2)
X2 <- data.frame(poli2,trimestre)
modelo2 <- Arima(log(yt2),order=c(1,0,2),
seasonal=list(order=c(0,0,1)),xreg=as.matrix(X2),method="ML")
k2 <- length(coef(modelo2)[coef(modelo2)!=0]);k2 #número de parámetros
df2 <- n2-k2 #grados de libertad
lmtest::coeftest(modelo2,df=df2)
interpdeltas(modelo=modelo2,ordenp=1,ordenq=2,ordenP=0,ordenQ=1,
             gradopoly=3,aditivo=FALSE,plotit=TRUE)

#Ej. 3, exponencial cúbico estacional con indicadoras (nivel ref=Q4)
modelo3 <- regexponencialv02(respuesta=yt2,data=X2)
summary(modelo3)
interpdeltas(modelo=modelo3,gradopoly=3,aditivo=FALSE,plotit=TRUE)
```

`LoessOptimo` *Función* `LoessOptimo()`

Description

Para ajuste y pronóstico de una serie no estacional mediante loess óptimo según criterios de información.

Usage

```
LoessOptimo(serie, grado = 1, criterio, h = 1, level = 0.95)
```

Arguments

`serie`	Un objeto tipo `ts` de serie de tiempo univariada con los valores de la serie a ajustar.
`grado`	Grado del polinomio local a ser ajustado sobre la serie. `grado=1` es para loess lineal y `grado=2` es para loess cuadrático. Su valor por defecto es 1.
`criterio`	Tipo de criterio de información `"aicc"` ó `"gcv"`, a usar para la elección del parámetro de suavizamiento en la rutina de loess óptimo mediante la función R `loess.as()` de la librería `fANCOVA`. Por defecto es `"aicc"`.
`h`	Número de predicciones a calcular después del ajuste. Por defecto es 1.
`level`	Nivel de confianza de los intervalos de predicción. Por defecto 0.95.

Value

Una lista con los siguientes objetos:

- `alfa.optim` valor del parámetro de suavizamiento loess óptimo.
- `nep` El número de parámetros equivalentes loess.
- `MSE` Estimación de la varianza del error de ajuste.
- `fitted` Objeto `ts` de la serie estimada.
- `residuals` Objeto `ts` con los residuos de ajuste obtenidos como la diferencia entre los valores observados y los ajustados para la serie.
- `forecast` Objeto `ts` multivariado con las predicciones puntuales y por intervalo para los `h` períodos pedidos.

Examples

```
serie <- ts(DatosChippy$serie,freq=12,start=c(1990,1))
curva <- ts(DatosChippy$Chippy,freq=12,start=c(1990,1))

#Gráfica de la serie y tendencia verdadera
plot(serie,lwd=1)
lines(curva,col=2,lwd=2)
legend("topright",legend=c(expression(sin(cos(t)*exp(-t/2)))),col=2,lty=1,lwd=2)

#Ajuste con validación cruzada con últimos 4 datos
h <- 4
n <- length(serie)-h
t <- 1:n #tiempos de ajuste
#valores a ajustar
yt <- ts(serie[t],freq=frequency(serie),start=start(serie))
tnuevo <- (n+1):length(serie) #tiempos de predicciones
#Fecha inicio predicciones
startpred=end(ts(serie[1:(n+1)],freq=frequency(serie),start=start(serie)))
#Valores observados en tiempos de predicciones
ytnuevo <- ts(serie[tnuevo],freq=frequency(serie),start=startpred)

ajusteLoess <- LoessOptimo(serie=yt,grado=2,criterio="aicc",h=h,level=0.95)
ythat <- fitted(ajusteLoess) #serie ajustada

#Serie y su ajuste
plot(serie)
lines(curva,col=2,lwd=2) #Tendencia verdadera
lines(ythat,col="blue1",lwd=2)
legend("topright",legend=c("Tendencia verdadera","Ajuste"),
col=c(2,"blue1"),lty=1,lwd=2)

#Número aproximado de parámetros
p <- round(ajusteLoess$nep)
p
#AIC y BIC aproximados version exp(C*n(p))
Criterios <- critinfresid(residuales=residuals(ajusteLoess),npar=p)
Criterios

#Pronósticos con IPs del 95% de confianza
pronosticos <- ajusteLoess$forecast
pronosticos

#MSE aproximado del ajuste
MSE <- ajusteLoess$MSE

#Gráficos de residuos de ajuste
diagresTSModel(modelo=ajusteLoess,RMSE=sqrt(MSE),all=FALSE,single=FALSE)
```

`Mipoly` *Función* `Mipoly()`

Description

Genera las variables de los polinomios para modelos de regresión con funciones de tendencia polinomiales. Se basa en la función `poly()` de `stats`.

Usage

```
Mipoly(tiempo, grado)
```

Arguments

tiempo	Un vector con los valores del indice de tiempo t.
grado	El grado p del polinomio deseado, debe ser un valor entero positivo a partir de 1.

Value

Un data.frame cuyas variables son t y sus potencias denominadas t2, t3, . . . , tp, según el grado polinomial p.

Examples

```
t <- 1:20 #índice de tiempo primeros 20 valores
poli2 <- Mipoly(tiempo=t,grado=2) #para predictores con polinomio de grado 2
poli2
poli3 <- Mipoly(tiempo=t,grado=3) #para predictores con polinomio de grado 3
poli3
```

Mytrigon *Función* Mytrigon()

Description

Genera las variables trigonométricas sin(2*pi*Fj*tiempo) y cos(2*pi*Fj*tiempo) en frecuencias Fj.

Usage

```
Mytrigon(tiempo, Frecuencias, indicej)
```

Arguments

tiempo	Vector de valores enteros positivos con el índice de tiempo para los cuales se debe calcular las funciones trigonométricas.
Frecuencias	Vector con las frecuencias Fj correspondientes a las ondas sinusoidales armónicas a considerar.
indicej	Vector de valores enteros indicando el sub índice j de las funciones trigonométricas para cada frecuencia en Frecuencias de manera que a Frecuencias[i] le asigna el indice indicej[i].

Value

Un data.frame con las variables sinFj, cosFj, siendo j el valor asignado en indicej. Para la frecuencia 1/2 solo crea la componente coseno, que corresponde a cos(pi*tiempo). Los pares de variables sinj, cosj aparecen en orden de menor a mayor frecuencia pero conservando el sufijo j asignado en indicej.

Examples

```
t <- 1:20 #índice de tiempo primeros 20 valores
#Trigonométricas en Fj=j/12 para j=1 a 6 y F7=0.35
trigono <- Mytrigon(tiempo=t,Frecuencias=c(c(1:6)/12,0.35),indicej=c(1,2,3,4,5,6,7))
trigono #Variables sinF7, cosF7 corresponden a la frecuencia F7=0.35
```

---

precisintervals *Función* precisintervals()

Description

Evalúa precisión de intervalos de predicción (IPs).

Usage

```
precisintervals(real, LIP, LSP, alpha = 0.05)
```

Arguments

real	Vector numérico con los valores observados en los periodos de pronóstico.
LIP	Vector numérico con los valores de los límites inferiores de los IPs.
LSP	Vector numérico con los valores de los límites superiores de los IPs.
alpha	Escalar entre 0 y 1 correspondiendo al nivel 1-alpha de confianza.

Value

Un vector con AmplitudProm la amplitud promedio, Cobertura(%) la cobertura (porcentaje) y ScoreProm el Score promedio de los IPs.

Examples

```
Real <- c(181958.0,185303.0,183429.0)
Pronost <- data.frame(rbind(c(175546.9,163886.4,188037.2),
c(175938.6,164246.2,188463.5),c(176323.1,164599.1,188882.2)))
names(Pronost) <- c("Pred","LimInf","LimSup")
Pronost
precisintervals(real=Real,LIP=Pronost[,2],LSP=Pronost[,3])
```

---

predictexpo *Función* predictexpo()

Description

Para la predicción puntual y por IPs sobre modelos de regresión exponencial ajustados con la función regexponencialv02(). También calcula I.Cs para la respuesta media en lugar de IPs.

Usage

```
predictexpo(
  modelo,
  new.data,
  level = 0.95,
  interval = c("none", "confidence", "prediction")
)
```

Arguments

modelo	Objeto tipo nls() generado por la función regexponencialv02().
new.data	data.frame con los valores de los predictores en los tiempos de predicción. El nombre de las variables debe ser igual al usado en el ajuste del modelo.
level	Un valor en (0,1) para el nivel de confianza de los intervalos. Por defecto es 0.95.
interval	Una cadena de caracteres correspondiendo a uno de los siguientes casos: "none", "confidence", "prediction". Por defecto es "none", es decir, calcula solo estimaciones o pronósticos puntuales. Para intervalos de confianza en la respuesta media use "confidence" y para intervalos de predicción use "prediction".

Value

Un objeto tipo data.frame con tres variables:

- forecast La estimación o pronóstico puntual de la respuesta.
- lower, upper Los límites inferior y superior de los intervalos pedidos.

Examples

```
#Simulando serie aditiva con modelo de tendencia exponencial lineal y errores correlacionados
n <- 250
t <- 1:n
error <- arima.sim(n=n,list(ar=0.8),sd=108)
Yt <- ts(exp(6.5+0.0105*t)+error,freq=1,start=1)
plot.ts(Yt)
lines(t,exp(6.5+0.0105*t),type="l",col="red2")

#Matriz de predictores
X <- Mipoly(tiempo=t,grado=1)
#Ajuste del modelo exponencial lineal
modelo <- regexponencialv02(respuesta=Yt,data=X)
summary(modelo)
#cinco tiempos de predicción
Xnuevo=Mipoly(tiempo=(n+1):(n+5),grado=1)
#Predicciones puntuales y por IPs del 95% de confianza
predictexpo(modelo=modelo,new.data=Xnuevo,interval="prediction")
```

pruebaDW1 *Función* pruebaDW1()

Description

Realiza test Durbin-Watson para autocorrelación de orden 1 en un MRLM. Usa la función durbinWatsonTest() de la librería car.

Usage

```
pruebaDW1(modelo)
```

Arguments

modelo	Un objeto tipo lm en el que se guardó el ajuste de un MRLM.

Value

un objeto data.frame con las siguientes variables:

- rho(1) estimado estimación de la autocorrelación de orden 1 rho(1) usando los residuales del modelo.
- Estadistico D-W El valor del estadístico de la prueba.
- VP H1: rho(1)>0 El valor P para el test H0: rho(1)=0, vs. H1:rho(1)>0.
- VP H1: rho(1)<0 El valor P para el test H0: rho(1)=0, vs. H1:rho(1)<0.

Examples

```
#Simulando serie con modelo de tendencia lineal y errores RB normales
t <- 1:300
yt <- 3500+200*t+rnorm(300,sd=4000)
yt <- ts(yt,freq=12,start=c(1980,1))
#Gráfico de la serie simulada
plot(yt)
#Ajuste del modelo de tendencia lineal y errores RB normales
mrl <- lm(yt~t)
summary(mrl)
#Aplicando función para test Durbin-Watson
pruebaDW1(modelo=mrl)
#Simulando serie con modelo de tendencia lineal y errores correlacionados
zt <- 3500+200*t+arima.sim(n=300,list(ar=0.7),sd=4000)
zt <- ts(zt,freq=12,start=c(1980,1))
#Gráfico de la serie simulada
plot(zt)
#Ajuste del modelo de tendencia lineal y errores RB normales
mrl2 <- lm(zt~t)
summary(mrl2)
#Aplicando función para test Durbin-Watson
pruebaDW1(modelo=mrl2)
```

regexpoErrorARMAv03 *Función* regexpoErrorARMAv03()

Description

Ajusta y pronostica de forma aproximada modelos de regresión exponenciales como el considerado en la función de usuario regexponencialv02(), pero con error ARMA, así: primero estima los parámetros de la regresión exponencial por mínimos cuadrados no lineales, luego ajusta a los residuos de este modelo un ARMA estacionario de media cero, mediante la funcion Arima() de la libreria forecast. Los valores estimados de la serie son aproximados sumando las estimaciones de la regresión no lineal y del modelo ARMA. El anterior procedimiento también es aplicado en la construcción de pronósticos puntuales. Intervalos de pronóstico son construidos aproximando el error estándar del error de predicción como la raíz cuadrada de la suma de la varianza estimada del error de predicción sobre el modelo ARMA y la varianza estimada (por el método delta) de la estructura exponencial evaluada en los tiempos de predicción.

Usage

```
regexpoErrorARMAv03(
  respuesta,
  data,
  newdata,
  level = 0.95,
  control = stats::nls.control(),
  order = c(0L, 0L, 0L),
  seasonal = list(order = c(0L, 0L, 0L), period = NA),
  fixed = NULL,
  method = c("CSS-ML", "ML", "CSS"),
  optim.method = "BFGS",
  optim.control = list()
)
```

Arguments

respuesta	Un objeto ts univariado con los datos a ser ajustados.
data, newdata	Objetos tipo data.frame, cuyas columnas corresponden a los valores de las variables predictoras del modelo en el ajuste y en el pronóstico, respectivamente.
level	Nivel de confianza para intervalos de predicción, dado como un valor en (0,1), por defecto es 0.95.
control	Una lista opcional de configuraciones de control para la función nls(). Consulte nls.control() para conocer los nombres de los parámetros de control configurables y sus efectos.
order	La especificación de la parte no estacional del modelo ARIMA. Los tres componentes (p,d,q) son los órdenes AR, de diferenciación y el orden MA, respectivamente.
seasonal	La especificación de la parte estacional del modelo ARIMA, más el período (que por defecto es frequency(respuesta)). Este argumento debe ser dado como una lista, con el orden de los componentes y el período, pero una especificación de solo un vector numérico de longitud 3 se convertirá en una lista adecuada con

	las especificaciones de los órdenes estacionales (P,D,Q) de la parte AR, de la diferenciación y del orden MA, de periodo s=frequency(respuesta).
fixed	Un vector numérico opcional de la misma longitud que la suma de los órdenes de la ecuación del ARMA estacionario de media cero a ajustar sobre los residuos del ajuste exponencial. Ver en la ayuda de la función arima() los detalles de este argumento.
method	Método de ajuste, a saber, máxima verosimilitud ("ML"), Máxima verosimilitud combinada con mínimos cuadrados condicionales ("CSS-ML") o mínimos cuadrados condicionales ("CSS"). El valor predeterminado (a menos que existan valores faltantes) es "CSS-ML", en el cual se usan mínimos cuadrados condicionales para encontrar valores iniciales y luego aplica máxima verosimilitud.
optim.method	El valor pasado al argumento 'method' para 'optim' en la funcion Arima(). Por defecto es "BFGS".
optim.control	Una lista de parámetros de control para 'optim', usados en la función Arima().

Value

La función produce una lista con los siguientes componentes:

- coefficients Una matriz con la tabla de parámetros estimados, sus errores estándar, el estadístico T0 y valor P asociado. Tenga en cuenta que no hay una estimación conjunta de los parámetros de regresión de la funcion exponencial y de los parámetros del modelo ARMA del error estructural, de modo que los errores estándar provienen del ajuste separado de las dos estructuras, pero los valores p son calculados bajo una distribución t-student cuyos grados de libertad son df=n - (total de parámetros). Esta tabla puede ser extraida con la función coef() aplicada al objeto R donde se guarde la estimación del modelo.
- fitted Objeto tipo "ts" univariado con los valores ajustados de la respuesta. Estos valores pueden ser extraidos mediante la función fitted() aplicada sobre el objeto R donde se guarde la estimación del modelo.
- residuals Objeto tipo "ts" univariado con los residuos del ajuste de la respuesta. Estos valores pueden ser extraidos mediante la función residuals() aplicada sobre el objeto R donde se guarde la estimación del modelo.
- sigma2 Estimación de la varianza de las innovaciones del modelo ARMA definido para el error estructural del modelo de regresión exponencial. Puede ser extraido como nombre_objeto$sigma2, donde 'nombre_objeto' es el nombre del objeto R donde se guarda la estimación del modelo.
- forecast Objeto tipo "mts", "ts", "matrix", "array", con los pronósticos puntuales y por intervalos de la respuesta para h=nrow(newdata) períodos después del ajuste.

Examples

```
library(forecast)
#Serie estacional trimestral multiplicativa
#de N=155 observaciones, inicio 1956-Q1
datos <- ts(CEMENTO,freq=4,start=c(1956,1))
n <- length(datos)-4
t <- 1:n
#Variables para ajuste y pronóstico modelo exponencial
#cúbico estacional con indicadoras (nivel ref=Q4) y error AR(6)
#serie de primeros n valores
yt <- ts(CEMENTO[t],freq=4,start=start(datos))
poli <- Mipoly(tiempo=t,grado=3)
trimestre <- seasonaldummy(yt)
```

```
#data.frame de los predictores en el ajuste
X <- data.frame(poli,trimestre)
tnuevo <- (n+1):length(datos)
#Fecha inicio predicciones
startpred <- end(ts(datos[1:(n+1)],freq=4,start=c(1956,1)))
#serie de los últimos cuatro valores
ytf <- ts(datos[tnuevo],freq=4,start=startpred)
polinuevo <- Mipoly(tiempo=tnuevo,grado=3)
trimestrenuevo <- seasonaldummy(yt,h=4)
#data.frame de los predictores  en la predicción
Xnuevo <- data.frame(polinuevo,trimestrenuevo)

#Ajuste y pronóstico del modelo
modelo <- regexpoErrorARMAv03(respuesta=yt,data=X,newdata=Xnuevo,
           order=c(6,0,0),method="ML")
coef(modelo)

#Criterios de información
k <- modelo$p;k
critinfresid(residuales=residuals(modelo),npar=k)

#Serie ajustada
ythat <- fitted(modelo)
plot(datos)
lines(ythat,col=2)
legend("topleft",legend=c("Observaciones","ajuste"),lty=1,col=1:2,lwd=2)

#Pronósticos y su precisión
predmod <- modelo$forecast; predmod
ytpron <- predmod[,1]
accuracy(ytpron,ytf)
precisintervals(real=ytf,LIP=predmod[,2],LSP=predmod[,3])

#Gráficos y resultados para validar supuestos sobre errores de ajuste
diagresTSModel(modelo=modelo,RMSE=sqrt(modelo$sigma2),all=TRUE,single=FALSE)
```

`regexponencialv02` *Función* `regexponencialv02()`

Description

Ajusta modelos de regresión de la forma `g(x)=exp(t(x)%*%beta)`, con `t(x)` la transpuesta de un vector de predictores `x` y `beta` el vector de parámetros.

Usage

```
regexponencialv02(respuesta, data, control = stats::nls.control())
```

Arguments

`respuesta`	El vector de valores de la variable respuesta.
`data`	Un objeto `data.frame` con los predictores a usar.
`control`	Una lista opcional de configuraciones de control. Consulte `nls.control()` para los valores de control configurables y su efecto.

Value

Son los mismos predeterminados para los objetos tipo `nls` generados por la función R `nls()`.

Examples

```
#Simulando serie aditiva con modelo de tendencia exponencial lineal y errores iid normales
n <- 250
t <- 1:n
error <- arima.sim(n=n,list(ar=0.8),sd=108)
Yt <- ts(exp(6.5+0.0105*t)+error,freq=1,start=1)
plot.ts(Yt)
lines(t,exp(6.5+0.0105*t),type="l",col="red2")

#Matriz de predictores
X <- Mipoly(tiempo=t,grado=1)
#Ajuste del modelo exponencial lineal
modelo <- regexponencialv02(respuesta=Yt,data=X)
summary(modelo)

#Gráficos de residuos de ajuste
diagresTSModel(modelo=modelo,RMSE=summary(modelo)$sigma,all=FALSE,single=FALSE)
```

RSALES *Ventas mensuales al por menor*

Description

Ventas mensuales del comercio por menor en Estados Unidos, 1954:1-1995:1, en dólares nominales.

Usage

```
RSALES
```

Format

`RSALES`**:**

Un data frame con 493 filas y 1 columna:

rsales Ventas al por menor

Source

Diebold, F. (2001), Elementos de Pronósticos.

salario — *Índice de salario real sector manufactura sin trilla de café, año base 2001.*

Description

Valores mensuales del índice de salario real en Colombia sector de manufactura sin trilla de café, 1990:1 a 2011:5, en puntos del índice

Usage

```
salario
```

Format

`salario`**:**
Un objeto `ts` univariado con 257 observaciones.

Source

DANE (MMM).

SelectModelv02 — *Función* `SelectModelv02()`

Description

Usando criterios de información selecciona el mejor modelo AR(p) completo. Está basado en la Función SelectModel() de la librería FitAR (ya no disponible en Cran-R) de A.I. McLeod and Y. Zhang.

Usage

```
SelectModelv02(
  z,
  lag.max = 15,
  Criterion = "default",
  Best = 3,
  Candidates = 5,
  t = "default"
)
```

Arguments

`z`	Una serie de tiempo.
`lag.max`	Un entero indicando el máximo orden autorregresivo para el proceso `AR(p)`.
`Criterion`	El criterio de información a usar para la selección del orden `p`. Por defecto es `"BIC"` (Bayesian Information Criterion). Otras opciones son `"AIC"` (Akaike Information Criterion), `"UBIC"` (Uncertainty-Penalized Bayesian Information Criterion), `"EBIC"` (Extended BIC), y `"GIC"` (Generalizad Information Criterion).

Best	El número final de modelos a ser seleccionados.
Candidates	El número de inicialmente seleccionados usando el criterio aproximado.
t	Parámetro de ajuste en los criterios"EBIC", "GIC".

Value

La función crean un objeto tipo "matrix", "array" de Best filas por 3 columnas, con los siguientes valores en sus columnas: p que corresponde a los valores del orden autorregresivo para los Best primeros casos de mínimo valor del criterio usado; los valores del criterio de información correspondiente, el exacto: Criterion-Exact y el aproximado: Criterion-Approx, donde Criterion corresponde al acrónimo del criterio de información usado.

Examples

```
#Simulando serie de un proceso AR(1) estacionario de media cero
zt <- ts(arima.sim(n=300,list(ar=0.7),sd=4000),freq=1,start=1)
SelectModelv02(z=zt,Criterion="AIC")
SelectModelv02(z=zt,Criterion="BIC")

#Simulando serie de un proceso AR(6) estacionario de media cero
xt <- arima.sim(n=300,list(ar=c(0.65,0.32,-0.27,0.25,0,-0.23)),sd=0.025)
SelectModelv02(z=xt,Criterion="AIC")
SelectModelv02(z=xt,Criterion="BIC")
SelectModelv02(z=xt,Criterion="EBIC")
SelectModelv02(z=xt,Criterion="UBIC")
SelectModelv02(z=xt,Criterion="GIC")
```

SuavizamientoEstacional

Función SuavizamientoEstacional()

Description

Sobre un objeto ts realiza ajuste y predicción por suavizamiento exponencial Holt-Winters aditivo o multiplicativo. El suavizamiento puede ser óptimo o con valores predefinidos para los parámetros de suavizamiento. Usa la función R HoltWinters() pero presenta en orden de los periodos del año las últimas estimaciones suavizadas de los efectos estacionales.

Usage

```
SuavizamientoEstacional(
  serie,
  alpha = NULL,
  beta = NULL,
  gamma = NULL,
  seasonal = "additive",
  h = 1,
  optim.start = c(alpha = 0.3, beta = 0.1, gamma = 0.1),
  optim.control = list()
)
```

Arguments

serie	Un objeto tipo ts de frecuencia s.
alpha	Parámetro de suavizamiento del nivel, debe ser un valor entre 0 y 1. Por defecto es NULL, en este caso, será hallado minimizando el SSE del ajuste.
beta	Parámetro de suavizamiento de la pendiente, debe ser un valor entre 0 y 1. Por defecto es NULL, en este caso, será hallado minimizando el SSE del ajuste.
gamma	Parámetro de suavizamiento de los efectos estacionales, debe ser un valor entre 0 y 1. Por defecto es NULL, en este caso, será hallado minimizando el SSE del ajuste.
seasonal	Cadena de caracteres para indicar el tipo de descomposición, por defecto es "additive" para el caso aditivo. Para el caso multiplicativo se define como "multiplicative".
h	Número de pronósticos que se realizarán inmediatamente después del ajuste. Por defecto es 1.
optim.start	Vector con componentes de nombres alpha, beta y gamma, de los valores iniciales de los parámetros de suavizamiento pasados al optimizador.
optim.control	Lista con parámetros de control adicionales pasados a optim.

Value

La función produce un objeto tipo lista con los siguientes componentes:

- coefficients Un data.frame con una sola columna de longitud s+2, con los valores finales suavizados del nivel, la pendiente y los efectos estacionales (estos ultimos en el orden i=1, 2, ...,s).
- fitted Objeto ts de los valores ajustados, con la misma frecuencia de serie pero iniciando en el tiempo s+1.
- residuals Objeto ts de los residuos de ajuste con la misma frecuencia de serie.
- forecast Objeto serie de tiempo multivariada (mts), con los pronósticos puntuales (columna 1), y límites de predicción del 95% de confianza (columnas 2 y 3), por defecto, para h=1 períodos después del ajuste.
- MSE El MSE aproximado del ajuste.

Examples

```
#Serie estacional trimestral multiplicativa de N=155 observaciones, inicio 1956-Q1
CEMENTO
datos <- ts(CEMENTO,freq=4,start=c(1956,1))
plot(datos)
t <- 1:151
yt <- ts(CEMENTO[t],freq=4,start=c(1956,1)) #serie de primeros 151 valores
#Fecha inicio predicciones
startpred <- end(ts(CEMENTO[1:152],freq=4,start=c(1956,1)))
#serie últimos cuatro valores
ytf <- ts(CEMENTO[152:length(CEMENTO)],freq=4,start=startpred)

#Ajuste por SEHW multiplicativo óptimo con n=151 y predicción con h=4
modelo <- SuavizamientoEstacional(yt,seasonal="multiplicative",h=4)
ythat <- fitted(modelo) #serie de valores ajustados

#Gráfica del ajuste
```

```
plot(datos)
lines(ythat,col=2)
legend("topleft",legend=c("Original","Ajustada SEHW"),col=c(1,2),lty=1)

#Calculando de forma aproximada AIC y BIC usando exp(C*n(p))
numpar <- frequency(datos)+1 #Aprox. del número de parámetros
Criterios <- critinfresid(residuales=residuals(modelo),npar=numpar)
Criterios

#Gráficos de residuos
MSE <- modelo$MSE; MSE #MSE aproximado del ajuste total del Suavizamiento
diagresTSModel(modelo=modelo,RMSE=sqrt(MSE),all=FALSE,single=FALSE)

#Predicciones puntuales y por IPs del 95% de confianza
predicciones <- modelo$forecast
predicciones
#Precisión de las predicciones
forecast::accuracy(predicciones[,1],ytf)
precisintervals(real=ytf,LIP=predicciones[,2],LSP=predicciones[,3])
```

`ventascompanyX` *Ventas mensuales de la compañía X.*

Description

Serie ficticia de las Ventas mensuales de una compañía X, 1965:1 a 1971:5, en cientos de dólares.

Usage

```
ventascompanyX
```

Format

`ventascompanyX`**:**
Un objeto `ts` univariado con 77 observaciones.

Source

O.D. Anderson, 1976 and O'Donovan, 1983.

Index