Package: Greymodels (via r-universe)

August 22, 2024

Type Package

Title Shiny App for Grey Forecasting Model

Version 2.0.1

Maintainer Jahajeeah Havisha <hjahajeeah@utm.ac.mu>

Description The 'Greymodels' Shiny app is an interactive interface for statistical modelling and forecasting using grey-based models. It covers several state-of-the-art univariate and multivariate grey models. A user friendly interface allows users to easily compare the performance of different models for prediction and among others, visualize graphical plots of predicted values within user chosen confidence intervals. Chang, C. (2019) <doi:10.24818/18423264/53.1.19.11>, Li, K., Zhang, T. (2019) <doi:10.1007/s12667-019-00344-0>, Ou, S. (2012) <doi:10.1016/j.compag.2012.03.007>, Li, S., Zhou, M., Meng, W., Zhou, W. (2019) <doi:10.1080/23307706.2019.1666310>, Xie, N., Liu, S. (2009) <doi:10.1016/j.apm.2008.01.011>, Shao, Y., Su, H. (2012) <doi:10.1016/j.aasri.2012.06.003>, Xie, N., Liu, S., Yang, Y., Yuan, C. (2013) <doi:10.1016/j.apm.2012.10.037>, Li, S., Miao, Y., Li, G., Ikram, M. (2020) <doi:10.1016/j.matcom.2019.12.020>, Che, X., Luo, Y., He, Z. (2013) <doi:10.4028/www.scientific.net/AMM.364.207>, Zhu, J., Xu, Y., Leng, H., Tang, H., Gong, H., Zhang, Z. (2016) <doi:10.1109/appeec.2016.7779929>, Luo, Y., Liao, D. (2012) <doi:10.4028/www.scientific.net/AMR.507.265>, Bilgil, H. (2020) <doi:10.3934/math.2021091>, Li, D., Chang, C., Chen, W., Chen, C. (2011) <doi:10.1016/j.apm.2011.04.006>, Chen, C. (2008) <doi:10.1016/j.chaos.2006.08.024>, Zhou, W., Pei, L. (2020) <doi:10.1007/s00500-019-04248-0>, Xiao, X., Duan, H. (2020) <doi:10.1016/j.engappai.2019.103350>, Xu, N., Dang, Y. (2015) <doi:10.1155/2015/606707>, Chen, P., Yu, H.(2014) <doi:10.1155/2014/242809>, Zeng, B., Li, S., Meng, W., Zhang, D. (2019) <doi:10.1371/journal.pone.0221333>, Liu, L., Wu, L. (2021) <doi:10.1016/j.apm.2020.08.080>, Hu, Y. (2020) <doi:10.1007/s00500-020-04765-3>, Zhou, P., Ang, B., Poh, K. (2006) <doi:10.1016/j.energy.2005.12.002>, Cheng, M., Li, J., Liu, Y., Liu, B. (2020) <doi:10.3390/su12020698>, Wang, H., Wang, P., Senel, M., Li, T. (2019) <doi:10.1155/2019/9049815>, Ding, S., Li, R. (2020) <doi:10.1155/2020/4564653>, Zeng, B., Li, C. (2018) <doi:10.1016/j.cie.2018.02.042>, Xie, N., Liu, S. (2015) <doi:10.1109/JSEE.2015.00013>, Zeng, X., Yan, S., He, F., Shi, Y. (2019) <doi:10.1016/j.apm.2019.11.032>.

License GPL-3

URL `https://github.com/havishaJ/Greymodels`

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0), dplyr, Metrics, cmna, plotly

Imports shiny, shinydashboard, shinyWidgets, ggplot2, readxl, particle.swarm.optimisation, scales, expm

Repository https://havishaj.r-universe.dev

RemoteUrl https://github.com/havishaj/greymodels

RemoteRef HEAD

RemoteSha 3fc19ebe91832d358c1c8f5f4d8027e1c7d6b1eb

Contents

`app_server` *Runs the Shiny app*

Description

Runs the greymodels Shiny app

Usage

```
ui()
server(input, output)
run_app()
```

Arguments

`ui`	Controls the layout and appearance of the greymodels shiny app
`input`	Stores the current values of all of the widgets in the app
`output`	Contains all of the code needed to update the R objects in the app
`server`	Contains the instructions to build the greymodels shiny app

Value

No return value, runs the app

Examples

```
# Only run this example in interactive R sessions

if (interactive()) {

library("shiny")
library("shinydashboard")
library("shinyWidgets")
library("readxl")
library("Metrics")
library("particle.swarm.optimisation")
library("cmna")
library("expm")
library("plotly")
library("ggplot2")
library("scales")
library("dplyr")

run_app <- function(){

  shiny::shinyApp(ui, server, options = list(launch.browser = TRUE))
}

}
```

`BackgroundValues` *Improved background values*

Description

A collection of grey forecasting models with improvements to the underlying background value z.

Usage

```
gm11(x0)
epgm11(x0)
tbgm11(x0)
igm11(x0)
gm114(x0)
```

Arguments

`x0`	Raw data
`gm11`	Basic grey model
`epgm11`	Extrapolation-based grey model
`tbgm11`	Data transformation-based grey model
`igm11`	Improved grey model
`gm114`	Grey model with single variable, one first-order variable, four background values

Value

fitted and predicted values

References

Chang C (2019). Extrapolation-based Grey Model for Small-Dataset Forecasting. Economic Computation and Economic Cybernetics Studies and Research, 53(1), 171-182. DOI:10.24818/18423264/53.1.19.11.

Li K, Zhang T (2019). A Novel Grey Forecasting Model and its Application in Forecasting the Energy Consumption in Shanghai. Energy Systems, pp. 1-16. DOI:10.1007/s12667-019-00344-0.

Ou S (2012). Forecasting Agricultural Output with an Improved Grey Forecasting Model based on the Genetic Algorithm. Computers and Electronics in Agriculture, 85, 33-39. DOI:10.1016/j.compag.2012.03.007.

Li S, Zhou M, Meng W, Zhou W (2019). A new Prediction Model for Forecasting the Automobiles Ownership in China. Journal of Control and Decision, 8(2), 155-164. DOI:10.1080/23307706.2019.1666310.

Examples

```
# GM(1,1) model

# x0 is the original data sequence
x0 <- c(2350,2465,2557,2577,2689,2739,2797,2885,2937,2996)

# Calculate AGO
  x1 <- cumsum(x0)

# Determine length of x0
  n <- length(x0)
```

```
# Generate background value sequence Z
  b <- numeric(n)

  for (i in 1:n){
    b[i] <- -(0.5*x1[i + 1] + 0.5*x1[i])
  }

  b1 <- b[1:n-1]

# Create a matrix B
  B <- matrix(1,nrow=n-1,ncol=2)
  B[,1] <- t(t(b1[1:n-1]))

# Create matrix yn
  yn <- matrix(c(x0),ncol=1)
  yn <- t(t(x0[2:n]))

# Estimate parameters a and b by ordinary least squares method (OLS)
  xcap <- solve (t(B)%*% B)%*% t(B) %*% yn
  a <- xcap[1,1]
  b <- xcap[2,1]

# Calculate fitted values
  scale_with <- function(k)
  {
    (x0[1] - (b/a)) * exp(-a*k) * (1 - exp(a))
  }
  fitted <- scale_with(1:n)
  x0cap <- c(x0[1],fitted[1:n-1])
  x0cap

# A is the number of forecast values
  A <- 4

# Predicted values
  x0cap4 <- scale_with(1 : n+A-1)
  x0cap5 <- tail(x0cap4,A)
  x0cap5

# Fitted and predicted values
  x0cap2 <- c(x0cap,x0cap5)
  x0cap2
```

`CombinedModels` *Combined models*

Description

A collection of hybrid grey forecasting models.

Usage

```
ngbm11(x0)
ggvm11(x0)
tfdgm11(x0)
```

Arguments

`x0`	Raw data
`ngbm11`	Non-linear grey Bernoulli model
`ggvm11`	Grey generalized Verhulst model
`tfdgm11`	Traffic flow mechanics grey model

Value

fitted and predicted values

References

Chen C (2008). Application of the Novel Nonlinear Grey Bernoulli Model for Forecasting Unemployment Rate. Chaos, Solitons and Fractals, 37(1), 278-287. DOI:10.1016/j.chaos.2006.08.024.

Zhou W, Pei L (2020). The Grey Generalized Verhulst model and its Application for Forecasting Chinese Pig Price Index. Soft Computing, 24, 4977-4990. DOI:10.1007/s00500-019-04248-0.

Xiao X, Duan H (2020). A New Grey Model for Traffic Fow Mechanisms. Engineering Applications of Artificial Intelligence, 88(2020), 103350. DOI:10.1016/j.engappai.2019.103350.

Examples

```
#TFDGM (1, 1) model: Traffic flow mechanics grey model

# Input data x0
x0 <- c(129,151,132,144,119,125,127,132)
# AGO
x1 <- cumsum(x0)

n <- length(x0)

z <- numeric(n)
for (i in 1:n){
  z[i] <- 0.5*(x1[i+1] + x1[i])
}
z1 <- z[1:n-1]

mat2 <- matrix(c(z1),ncol=1)

for (i in 1:n){
  z[i] <- (0.5*(x1[i+1] + x1[i]))^2
}
z2 <- z[1:n-1]
```

```
mat1 <- matrix(c(z2),ncol=1)
mat3 <- matrix(1,nrow=n-1,ncol=1)

B <- cbind(mat1, mat2, mat3)

y <- matrix(c(x0),ncol=1)
y <- t(t(x0[2:n]))

pcap <- (solve (t(B) %*% B)) %*% t(B) %*% y
a <- pcap[1,1]
b <- pcap[2,1]
lambda <- pcap[3,1]

p <- b/(2*a)
q <- ((b^2)/(4*(a^2))) - (lambda/a)

forecast <- numeric(n)
for (k in 1:n){
  if (q == 0){
    C2 <- (-1 / (x0[1] + p)) - a
    forecast[k] <-  ( -1 / ((a*k) + C2) ) - p
  } else if (q < 0) {
    c3 <- (1/sqrt(-q)) * atan( (x0[1]+p) / sqrt(-q) ) - a
    forecast[k] <-  sqrt(-q)* tan( sqrt(-q) * ( (a*k) + c3 ) ) - p
  }
}
x1cap <- c(forecast)

x0cap <- numeric(n)
for (i in 1:n){
  x0cap[i] <- x1cap[i+1] - x1cap[i]
}
x0cap1 <- x0cap[1:n-1]
x0cap <- c(x0[1],x0cap1)
# Fitted values
x0cap

A <- 4

forecasta <- numeric(n)
for (k in 1:n+A){
  if (q == 0){
    C2 <- (-1 / (x0[1] + p)) - a
    forecast[k] <-  ( -1 / ((a*k) + C2) ) - p
  } else if (q < 0) {
    c3 <- (1/sqrt(-q)) * atan( (x0[1]+p) / sqrt(-q) ) - a
    forecasta[k] <-  sqrt(-q)* tan( sqrt(-q) * ( (a*k) + c3 ) ) - p
  }
}
x1cap4 <- c(forecasta)

t4 <- length(x1cap4)
```

```
x0cap4 <- numeric(t4)
for (i in 1:t4-1) {
  x0cap4[i] <- x1cap4[i+1] - x1cap4[i]
}
x0cap4 <- c(x0[1],x0cap4[1:t4-1])

# Predicted values
x0cap5 <- tail(x0cap4,A)
x0cap5

# Fitted & Predicted values
x0cap2 <- c(x0cap,x0cap5)
x0cap2
```

ConfidenceInterval *Confidence interval of predicted values*

Description

The CIvalue, CI_rm and CI_mdbgm functions calculate the confidence interval of the predicted values.

Usage

```
CIvalue(fp1,actual1,x,ci)
CI_rm(fp1,actual1,x,ci)
CI_nhmgmp(fp1,x01,x02,x,ci)
CI_igndgm(fp1,actual1,x,ci)
CI_mdbgm(fp1,actual1,x,ci)
```

Arguments

fp1	Fitted and predicted values
actual1	Raw data
x01	Raw data of variable 1
x02	Raw data of variable 2
x	Number of forecasts chosen by the user
ci	The confidence level chosen by the user. Values range between 90%, 95% and 99%.

Value

confidence interval of predicted values

Examples

```
# Confidence interval of predicted values for EPGM (1, 1) model

# fp1 is the sequence of fitted and predicted values

fp1<-c(560,541.4,517.8,495.3,473.7,453.1,433.4,414.5,396.5)

# actual1 is the original data

actual1<-c(560,540,523,500,475)

fp2 <- t(fp1)

w <- length(fp2)

actual2 <- t(actual1)

n <- length(actual2)

fitted1 <- fp2[1:n]

fitted2 <- tail(fp1,4)

# x is the number of values to predict

x <- 4

predicted <- t(fitted2[1:x])

t <- length(predicted)

# Performance error - Root mean square error (rmse)

require("Metrics")

s <- rmse(actual2, fitted1)

sse <- sum((actual2 - fitted1)^2)

mse <- sse / (n - 2)

# ci is the confidence level (90, 95, 99)

ci <- 95

cc <- (ci + 100)/200

t.val <- qt(cc, n - 2)

# Calculate prediction interval

u <- numeric(t)
```

```
l <- numeric(t)

for (i in 1:t) {
  u[i] = predicted[i] + (t.val * (sqrt(mse) * sqrt(i)))
  l[i] = predicted[i] - (t.val * (sqrt(mse) * sqrt(i)))
}

UpperBound <- c(u[1:t])
LowerBound <- c(l[1:t])

CIset <- data.frame(LowerBound,UpperBound)
CIset
```

ExtendedForms *Extended forms of grey models*

Description

A collection of extended grey forecasting models.

Usage

```
dgm11(x0)
dgm21(x0)
odgm21(x0)
ndgm11(x0)
vssgm11(x0)
gom11(x0)
gomia11(x0)
ungom11(x0)
exgm11(x0)
egm11(k,x0,k_A,x0_A)
```

Arguments

x0	Raw data
k	Data index of raw data
x0_A	Raw data (testing set)
k_A	Data index (testing set)
dgm11	Discrete grey model with single variable, first order differential equation
dgm21	Discrete grey model with single variable, second order differential equation model
odgm21	Optimized discrete grey model with single variable, second order differential equation
ndgm11	Non-homogeneous discrete grey model

`vssgm11`	Variable speed and adaptive structure-based grey model
`gom11`	Grey opposite-direction model based on inverse accumulation and traditional interpolation method
`gomia11`	Grey opposite-direction model based on inverse accumulation
`ungom11`	Unbiased grey opposite-direction model based on inverse accumulation
`exgm11`	Exponential grey model
`egm11`	Extended grey model

Value

fitted and predicted values

References

Xie N, Liu S (2009). Discrete Grey Forecasting Model and its Application. Applied Mathematical Modelling, 33(2), 1173-1186. DOI:10.1016/j.apm.2008.01.011.

Shao Y, Su H (2012). On Approximating Grey Model DGM (2, 1). 2012 AASRI Conference on Computational Intelligence and Bioinformatics, 1, 8-13. DOI:10.1016/j.aasri.2012.06.003.

Xie N, Liu S, Yang Y, Yuan C (2013). On Novel Grey Forecasting Model based on Non-homogeneous Index Sequence. Applied Mathematical Modelling, 37, 5059-5068. DOI:10.1016/j.apm.2012.10.037.

Li S, Miao Y, Li G, Ikram M (2020). A Novel Varistructure Grey Forecasting Model with Speed Adaptation and its Application. Mathematical and Computers in Simulation, 172, 45-70. DOI:10.1016/j.matcom.2019.12.02

Che X, Luo Y, He Z (2013). Grey New Information GOM (1, 1) Model based Opposite-Direction Accumulated Generating and its Application. Applied Mechanics and Materials, 364, 207-210. DOI:10.4028/www.scientific.net/AMM.364.207.

Power Load Forecasting based on GOM (1, 1) Model under the Condition of Missing Data. 2016 IEEEPES Asia-Pacific Power and Energy Engineering Conference (APPEEC), pp. 2461-2464. DOI:10.1109/appeec.2016.7779929.

Luo Y, Liao D (2012). Grey New Information Unbiased GOM (1, 1) Model based on Opposite-Direction Accumulated Generating and its Application. Advanced Materials Research, 507, 265-268. DOI:10.4028/www.scientific.net/AMR.507.265.

Bilgil H (2020). New Grey Forecasting Model with its Application and Computer Code. AIMS Mathematics, 6(2), 1497-1514. DOI: 10.3934/math.2021091.

An Extended Grey Forecasting Model for Omnidirectional Forecasting considering Data Gap Difference. Applied Mathematical Modeling, 35, 5051-5058. DOI:10.1016/j.apm.2011.04.006.

Examples

```
# EXGM (1, 1): Exponential grey model
# Input data x0
x0 <- c(2028,2066,2080,2112,2170,2275,2356,2428)

# Calculate accumulated generating operation (AGO)
x1 <- cumsum(x0)

# n is the length of sequence x0
n <- length(x0)

# Create matrix y
y <- matrix(c(x0),ncol=1)
y <- t(t(x0[2:n]))

b <- numeric(n)
for (i in 1:n){
  b[i] <- -0.5*(x1[i+1] + x1[i])
}
b1 <- b[1:n-1]

# Create matrix B2
mat1 <- matrix(c(b1),ncol=1)
mat2 <- matrix(1,nrow=n-1,ncol=1)
f <- numeric(n)
for (i in 1:n){
  f[i] <- ( exp(1) - 1) * exp(-i)
}
f1 <- f[2:n]

mat3 <- matrix(c(f1),ncol=1)
B2 <- cbind(mat1, mat2, mat3)

# Parameters estimation (a, b and c) by ordinary least squares method (OLS)
rcap <- (solve (t(B2) %*% B2)) %*% t(B2) %*% y
a <- rcap[1,1]
b <- rcap[2,1]
c <- rcap[3,1]

scale_with <- function(k)
{
 ( x1[1] - (b/a) - ( ( c/(a-1) )*exp(-1) ) ) * exp(a*(1-k)) + (b/a) + ( c/(a-1) )*exp(-k)
}
forecast1 <- scale_with(1:n)
x1cap <- c(forecast1)
x0cap1 <- numeric(n)
for (i in 1:n){
  x0cap1[i] <- x1cap[i+1] - x1cap[i]
}
x0cap <- c(x0[1],x0cap1[1:n-1])
# Fitted values
x0cap

# A is the number of forecast values
A <- 4
```

```
x1cap4 <- scale_with(1 : n+A )
t4 <- length(x1cap4)
x0cap4 <- numeric(t4-1)
for (i in 1:t4-1) {
  x0cap4[i] <- x1cap4[i+1] - x1cap4[i]
}
x0cap4 <- c(x0[1],x0cap4[1:t4-1])
x0cap5 <- tail(x0cap4,A)
# Predicted values
x0cap5

x0cap2 <- c(x0cap,x0cap5)
# Fitted and predicted values
x0cap2
```

IntervalMultivariable *Multivariate interval sequences*

Description

A collection of multivariate grey forecasting models based on interval number sequences.

Usage

```
igndgm12(LB,UB)
mdbgm12(x01L,x01U,x02L,x02U,x01La,x01Ua,x02La,x02Ua)
```

Arguments

LB, UB	Lower and upper bound of interval sequence
x01L, x01U	Lower and upper bound of first interval sequence (training set)
x02L, x02U	Lower and upper bound of second interval sequence (training set)
x01La, x01Ua	Lower and upper bound of first interval sequence (testing set)
x02La, x02Ua	Lower and upper bound of second interval sequence (testing set)
igndgm12	Interval grey number sequence based on non-homogeneous discrete grey model
mdbgm12	Multivariate grey model based on dynamic background algorithm

Value

fitted and predicted values

References

Xie N, Liu S (2015). Interval Grey Number Sequence Prediction by using Nonhomogeneous Exponential Discrete Grey Forecasting Model. Journal of Systems Engineering and Electronics, 26(1), 96-102. DOI:10.1109/JSEE.2015.00013.

Zeng X, Yan S, He F, Shi Y (2019). Multivariable Grey Model based on Dynamic Background Algorithm for Forecasting the Interval Sequence. Applied Mathematical Modelling, 80(23). DOI:10.1016/j.apm.2019.11.032.

Examples

```
#MDBGM (1, 2) model: Multivariate grey model based on dynamic background algorithm.

# Input data
#x01 Lower and upper bound of sequence 1
#x02 Lower and upper bound of sequence 2

# x01L is the lower bound of sequence 1
x01L <- c(2721,3136,3634,3374,3835,3595,3812,4488)

# x01U is the upper bound of sequence 1
x01U <- c(3975,4349,4556,5103,5097,5124,5631,6072)

# x02L is the lower bound of sequence
x02L <- c(24581,30070,36656,36075,42173,42074,45537,55949)

# x02U is the upper bound of sequence 2
x02U <- c(41731,49700,55567,61684,68295,68342,73989,78194)

x01 <- cbind(x01L,x01U)
x02 <- cbind(x02L,x02U)

# AGO
x11L <- cumsum(x01L)
x11U <- cumsum(x01U)

x11 <- cbind(x11L,x11U)

x12L <- cumsum(x02L)
x12U <- cumsum(x02U)

x12 <- cbind(x12L,x12U)

# Length of sequence
n <- length(x01L)

# Background values
b <- numeric(n)
for (i in 1:n){
  b[i] <- (0.5*x11L[i + 1] + 0.5*x11L[i])
  b1 <- b[1:n-1]
}
z1L <- matrix(c(b1),ncol=1)
```

```
n <- length(x01L)
d <- numeric(n)
for (i in 1:n){
  d[i] <- (0.5*x11U[i + 1] + 0.5*x11U[i])
  d1 <- d[1:n-1]
}
z1U <- matrix(c(d1),ncol=1)

# Create matrix Y
YL <- matrix(c(x01L[2:n]),ncol=1)
YU <- matrix(c(x01U[2:n]),ncol=1)

# Create matrix X
mat1 <- matrix(c(x12L[2:n]),ncol=1)
mat2 <- matrix(c(x12U[2:n]),ncol=1)
mat3 <- matrix(c(x11L[1:n-1]),ncol=1)
mat4 <- matrix(c(x11U[1:n-1]),ncol=1)
mat5 <- matrix(2:n,nrow=n-1,ncol=1)
mat6 <- matrix(1,nrow=n-1,ncol=1)

X <- cbind(mat1,mat2,mat3,mat4,mat5,mat6)

# Parameters estimation by OLS - Lower
A1 <- solve (t(X) %*% X) %*% t(X) %*% YL
miu11 <- A1[1,1]
miu12 <- A1[2,1]
gamma11 <- A1[3,1]
gamma12 <- A1[4,1]
g1 <- A1[5,1]
h1 <- A1[6,1]

# Parameters estimation by OLS - Upper
A2 <- solve (t(X) %*% X) %*% t(X) %*% YU
miu21 <- A2[1,1]
miu22 <- A2[2,1]
gamma21 <- A2[3,1]
gamma22 <- A2[4,1]
g2 <- A2[5,1]
h2 <- A2[6,1]

# Fitted values - Lower
scale_with <- function(k)
{
 (miu11*x12L[k]) + (miu12*x12U[k]) + (gamma11*x11L[k-1]) + (gamma12*x11U[k-1]) + (g1*k) + h1
}
forecast_L <- scale_with(2:n)
x0cap1L <- c(x01L[1],forecast_L)

# Fitted values - Upper
scale_with <- function(k)
{
 (miu21*x12L[k]) + (miu22*x12U[k]) + (gamma21*x11L[k-1]) + (gamma22*x11U[k-1]) + (g2*k) + h2
}
forecast_U <- scale_with(2:n)
x0cap1U <- c(x01U[1],forecast_U)

# Matrix of fitted values (lower and upper)
x0cap <- matrix(c(cbind(x0cap1L,x0cap1U)),ncol=2)
x0cap
```

`Multivariable` *Multivariate sequences*

Description

A collection of grey forecasting models based on multiple variables.

Usage

```
gm13(x1,x2,x3)
igm13(x1,x2,x3)
nhmgm1(x01,x02)
nhmgm2(x01,x02)
gmcg12(x01,x02,dat_a)
gmc12(x01,x02,dat_a)
dbgm12(x01,x02,dat_a)
```

Arguments

`x1, x2, x3`	Raw data of 3 variables (training set)
`x01, x02`	Raw data of 2 variables (training set)
`dat_a`	Raw data of x02 (testing set)
`gm13`	Grey multivariate model with first order differential equation and 3 variables
`igm13`	Improved grey multivariate model with first order differential equation and 3 variables
`nhmgm1`	Non-homogeneous multivariate grey model with first order differential equation and 2 variables with p = 1
`nhmgm2`	Non-homogeneous multivariate grey model with first order differential equation and 2 variables with p = 2
`gmcg12`	Multivariate grey convolution model with first order differential equation and 2 variables using the Gaussian rule
`gmc12`	Multivariate grey convolution model with first order differential equation and 2 variables using the trapezoidal rule
`dbgm12`	Multivariate grey model with dynamic background value, first order differential equation and 2 variables using the Gaussian rule

Value

fitted and predicted values

References

Cheng M, Li J, Liu Y, Liu B (2020). Forecasting Clean Energy Consumption in China by 2025: Using Improved Grey Model GM (1, N). Sustainability, 12(2), 1-20. DOI:10.3390/su12020698.

Wang H, Wang P, Senel M, Li T (2019). On Novel Non-homogeneous Multivariable Grey Forecasting Model NHMGM. Mathematical Problems in Engineering, 2019, 1-13. DOI:10.1155/2019/9049815.

Ding S, Li R (2020). A New Multivariable Grey Convolution model based on Simpson's rule and its Application. Complexity, pp. 1-14. DOI:10.1155/2020/4564653.

Zeng B, Li C (2018). Improved Multivariable Grey Forecasting Model and with a Dynamic Background Value Coefficient and its Application. Computers and Industrial Engineering, 118, 278-290. DOI:10.1016/j.cie.2018.02.042.

Examples

```
# GMC_g (1, 2) model

# Input raw data
x01 <- c(897,897,890,876,848,814)
x02 <- c(514,495,444,401,352,293)
dat_a <- c(514,495,444,401,352,293,269,235,201,187)

# AGO
x11 <- cumsum(x01)
x12 <- cumsum(x02)

n <- length(x01)

b11 <- numeric(n)
b12 <- numeric(n)

for (i in 1:n){
  b11[i] <- -(0.5*x11[i + 1] + 0.5*x11[i])
  b12[i] <- (0.5*x12[i + 1] + 0.5*x12[i])
}
b11a <- b11[1:n-1]
b12a <- b12[1:n-1]

mat1 <- matrix(c(b11a),ncol=1)
mat2 <- matrix(c(b12a),ncol=1)
mat3 <- matrix(1,nrow=n-1,ncol=1)

B <- cbind(mat1, mat2, mat3)

yn <- matrix(c(x01),ncol=1)
yn <- t(t(x01[2:n]))
```

```
xcap <- solve (t(B) %*% B) %*% t(B) %*% yn

beta1 <- xcap[1,1]
beta2 <- xcap[2,1]
u <- xcap[3,1]

fe <- numeric(n)
for (i in 1:n){
  fe[i] <- beta2 * x12[i] + u
}
E <- matrix(c(fe[1:n]),ncol =1)
xrG <- replicate(n,0)
for (t in 2:n){
  sm <- 0
  for (e in 2:t){
    sm <- sm + (  (exp(-beta1*(t - e + 0.5)))) * ( 0.5 * (E[e]+ E[e-1]) )
  }
  xrG[t] <- ( x01[1]*exp(-beta1*(t-1)) ) + sm
}
xcap1G <- c(x01[1],xrG[2:n])
fG <- numeric(n-1)
for (i in 1:n-1){
  fG[i] <- (xcap1G[i+1] - xcap1G[i])
}
f1G <- fG[1:n-1]
x0cap <- matrix(c(x01[1],f1G[1:n-1]),ncol=1)
# Fitted values
x0cap

A <- 4

newx02 <- as.numeric(unlist(dat_a))
m <- length(newx02)
newx12 <- cumsum(newx02)
fe_A <- numeric(m)
for (i in 1:m){
  fe_A[i] <- beta2 * newx12[i] + u
}
E_A <- matrix(c(fe_A[1:m]),ncol =1)
xrG_A <- replicate(m,0)
for (t in 2:m){
  sm <- 0
  for (e in 2:t){
    sm <- sm + (  (exp(-beta1*(t - e + 0.5)))) * ( 0.5 * (E_A[e]+ E_A[e-1]) )
  }
  xrG_A[t] <- ( x01[1]*exp(-beta1*(t-1)) ) + sm
}
xcap1G_A <- c(x01[1],xrG_A[2:m])

fG_A <- numeric(m-1)
for (i in 1:m-1){
  fG_A[i] <- (xcap1G_A[i+1] - xcap1G_A[i])
}
f1G_A <- fG_A[1:m-1]

x0cap4 <- matrix(c(x01[1],f1G_A[1:m-1]),ncol=1)

x0cap5 <- tail(x0cap4,A)
# Predicted values
x0cap5

# Fitted & Predicted values
x0cap2 <- c(x0cap,x0cap5 )
x0cap2
```

Optimization *Optimization-based grey models*

Description

A collection of grey forecasting models using optimization techniques to find optimal parameters of grey models.

Usage

```
optim_psogm(x0)
psogm11(x0)
optim_andgm(x0)
andgm11(x0)
optim_egm11r(x0)
egm11r(x0)
```

Arguments

`x0`	Raw data
`optim_psogm`	Parameters optimization (a and b) by particle swarm optimization(PSO)
`psogm11`	Particle swarm optimization-based grey model
`optim_andgm`	Parameters optimization (r) by PSO
`andgm11`	Adjacent non-homogeneous discrete grey model
`optim_egm11r`	Parameters optimization (r) by PSO
`egm11r`	Even form of grey model with one variable and one first order equation with accumulating generation of order r

Value

fitted and predicted values

References

Zeng B, Li S, Meng W, Zhang D (2019). An Improved Grey Prediction Model for China's Beef Comsumption Forecasting. PLOS ONE, 14(9), 1-18. DOI:10.1371/journal.pone.0221333.

Liu L, Wu L (2021). Forecasting the Renewable Energy Consumption of the European Countries by an Adjacent Non-homogeneous Grey Model. Applied Mathematical Modelling, 89, 1932-1948. DOI:10.1016/j.apm.2020.08.080.

Examples

```
# Input raw data
x0 <- c(2.8,3.8,4.6,5.2,5.7,6.0,6.2,6.92,7.77,8.92,10.06)

# Parameter optimization

library(particle.swarm.optimisation)

fitness_function <- function(value)
  {
    r <- value[1]

    n <- length(x0)

    xr1 <- numeric(n)

    for (i in 1:n){
      xr1[i] <-  ( (r-1)/r ) * sum(x0[1:i]) + (1/r)*x0[i+1]
    }
    xr <-  c(x0[1],xr1[1:n-1])

    mat1 <-matrix(xr[1:n-1], nrow=n-1,ncol=1)
    mat2 <-matrix(2:n-1, nrow=n-1,ncol=1)
    mat3 <- matrix(1,nrow=n-1,ncol=1)

    B <- cbind(mat1, mat2, mat3)

    y <- t(t(xr[2:n]))

    rcap <- (solve (t(B) %*% B)) %*% t(B) %*% y
    beta1 <- rcap[1,1]
    beta2 <- rcap[2,1]
    beta3 <- rcap[3,1]

    scale_with <- function(k)
    {
      ( beta1^k * x0[1] ) + ( ( 1 - beta1^k )/( 1 - beta1 ) ) * (beta2*k + beta3)
    }
    forecast1 <- scale_with(1:n)

    xrcap <- c(x0[1],forecast1)

    matrix2 <- matrix("",1,n)
    matrix2 <- as.numeric(matrix2)
    matrix2[1] <- x0[1]

    for (i in 2:length(matrix2+1)) {
      matrix2[i] <- r*xrcap[i] - (r-1)*sum(matrix2[1:i-1])
    }
    particule_result <- matrix2
    fitness <- -(1/n)*sum(abs((x0-particule_result)/x0)*100, na.rm=TRUE)
    return(fitness)
  }
  values_ranges <- list(c(0.001,5))
  swarm <- ParticleSwarm$new(pop_size = 100,
                             values_names = list("r"),
                             fitness_function = fitness_function,
                             max_it = 100,
                           acceleration_coefficient_range = list(c(0.5,1.5),c(0.5,1.5)),
                             inertia = 0.7,
                             ranges_of_values = values_ranges)
  swarm$run(plot = FALSE,verbose = FALSE,save_file = FALSE)
  swarm$swarm_best_values

  opt_r <- swarm$swarm_best_values[1]
  opt_r

  n <- length(x0)

  xr1r <- numeric(n)
  for (i in 1:n){
    xr1r[i] <-  ( (opt_r-1)/opt_r ) * sum(x0[1:i]) + (1/opt_r)*x0[i+1]
  }
  xoptr <-  c(x0[1],xr1r[1:n-1])

  mat1r <-matrix(xoptr[1:n-1], nrow=n-1,ncol=1)
  mat2r <-matrix(2:n-1, nrow=n-1,ncol=1)
  mat3r <- matrix(1,nrow=n-1,ncol=1)

  Br <- cbind(mat1r, mat2r, mat3r)

  yr <- t(t(xoptr[2:n]))

  rcapr <- (solve (t(Br) %*% Br)) %*% t(Br) %*% yr
  beta1r <- rcapr[1,1]
  beta2r <- rcapr[2,1]
  beta3r <- rcapr[3,1]

  scale_with <- function(k)
  {
    ( beta1r^k * x0[1] ) + ( ( 1 - beta1r^k )/( 1 - beta1r ) ) * (beta2r*k + beta3r)
  }
  forecast1r <- scale_with(1:n)

  xrcapr <- c(x0[1],forecast1r)
```

```
matrix2r <- matrix("",1,n)
matrix2r <- as.numeric(matrix2r)

matrix2r[1] <- x0[1]

for (i in 2:length(matrix2r+1)) {
  matrix2r[i] <- opt_r*xrcapr[i] - (opt_r-1)*sum(matrix2r[1:i-1])
}
x0cap <- c(matrix2r)
# Fitted values
x0cap

A <- 4

# Predicted values
n <- length(x0)
nn <- n + A
scale_with <- function(k)
{
  ( beta1r^k * x0[1] ) + ( ( 1 - beta1r^k )/( 1 - beta1r ) ) * (beta2r*k + beta3r)
}
forecast1ra <- scale_with(1:nn)

xrcapra <- c(x0[1],forecast1ra)
matrix2ra <- matrix("",1,nn)
matrix2ra <- as.numeric(matrix2ra)
matrix2ra[1] <- x0[1]

for (i in 2:length(matrix2ra+1)) {
  matrix2ra[i] <- opt_r*xrcapra[i] - (opt_r-1)*sum(matrix2ra[1:i-1])
}
x0cap4 <- c(matrix2ra)
x0cap5 <- tail(x0cap4,A)
# Predicted values
x0cap5

# Fitted & Predicted values
x0cap2 <- c(x0cap,x0cap5)
x0cap2
```

ParametersEstimation *Parameters estimation*

Description

A collection of grey forecasting models based on parameters estimation.

Usage

```
sogm21(x0)
ngm11k(x0)
ngm11kc(x0)
ongm11kc(x0)
```

Arguments

`x0`	Raw data
`sogm21`	Structured optimized grey model with single variable and second order differential equation
`ngm11k`	Nonlinear grey model
`ngm11kc`	Nonlinear grey model
`ongm11kc`	Optimized nonlinear grey model

Value

fitted and predicted values

References

Xu N, Dang Y (2015). An Optimized Grey GM (2, 1) Model and Forecasting of Highway Subgrade Settlement. Mathematical Problems in Engineering, 2015(1), 1-6. DOI:10.1155/2015/606707.

Chen P, Yu H (2014). Foundation Settlement Prediction based on a Novel NGM Model. Mathematical Problems in Engineering 2014, 242809. DOI:10.1155/2014/242809.

Examples

```
# ONGM (1, 1, k, c) model: Nonlinear grey model

# Input data x0
x0 <- c(23.36,43.19,58.73,70.87,83.71,92.91,99.73,105.08,109.73,112.19,113.45)

# AGO
x1 <- cumsum(x0)

tm <- length(x0)

# Create matrix y
y <- matrix(c(x0),ncol=1)
y <- t(t(x0[2:tm]))

b <- numeric(tm)
for (i in 1:tm){
  b[i] <- -0.5*(x1[i+1] + x1[i])
}
b1 <- b[1:tm-1]
```

```
# Create matrix B2
mat1 <- matrix(c(b1),ncol=1)
mat2 <-matrix(2:tm, nrow=tm-1,ncol=1)
mat3 <- matrix(1,nrow=tm-1,ncol=1)

B2 <- cbind(mat1, mat2, mat3)

# Parameters estimation by OLS
rcap <- (solve (t(B2) %*% B2)) %*% t(B2) %*% y
a <- rcap[1,1]
b <- rcap[2,1]
c <- rcap[3,1]

m <- log ((2+a)/(2-a))
n <- (m*b)/a
p <- (m*c)/a - (n/a) + (n/2) + (n/m)

scale_with <- function(k)
{
  (1-exp(a))*(x1[1]-(n/m)+(n/(m^2))-(p/m))*exp(-m*(k-1))+(n/m)
}
forecast1 <- scale_with(2:tm)

x0cap <- c(x0[1],forecast1)
# Fitted values
x0cap

A <- 4

x0cap4 <- scale_with(1 : tm+A )

x0cap5 <- tail(x0cap4,A)
# Predicted values
x0cap5

# Fitted & Predicted values
x0cap2 <- c(x0cap,x0cap5)
x0cap2
```

`Plots` *plots*

Description

The plots function gives an interactive plot of the model.

Usage

```
plots(x0,x0cap2,ci,model)
```

```
plotrm(x0,x0cap2,ci,model)
plotsmv1(actual1,fp1,ci,model)
plotsmv2(actual1,fitted,ci,model)
plotsigndgm(actual,pred,ci,model)
plots_mdbgm12(actual,pred,ci,model)
```

Arguments

`x0`	Raw data
`x0cap2`	Fitted and predicted data
`actual`	Raw data of interval sequences
`actual1`	Raw data of multi-variate sequences
`fp1`	Fitted and predicted data of first variable
`fitted`	Fitted data of multi-variate sequences
`pred`	Fitted and predicted data of interval sequences
`ci`	The confidence level chosen by the user. Values range between 90%, 95% and 99%.
`model`	The model under considration

Value

plots

Examples

```
# Plots - EPGM (1, 1) model

x0cap2<-c(560,541.4,517.8,495.3,473.7,453.1,433.4,414.5,396.5)

x0<-c(560,540,523,500,475)

# length of x0
n <- length(x0)

fitted2 <- t(x0cap2)

x0cap <- x0cap2[1:n]

# Last 4 values of x0cap2
fitted3 <- tail(x0cap2,4)

x0cap5 <- fitted3

w <- length(x0cap2)
t <- length(x0cap5)

# Performance errors
# Root mean square error
```

```
s <- rmse(x0, x0cap)

# Sum of square error
sse <- sum((x0 - x0cap)^2)

# Mean square error
mse <- sse / (n - 2)

# Calculate confidence interval
ci <- 95
cc <- (ci + 100)/200

t.val <- qt(cc, n - 2)

u <- numeric(t)
l <- numeric(t)
for (i in 1:t) {
  u[i] = x0cap5[i] + (t.val * (sqrt(mse) * sqrt(i)))
  l[i] = x0cap5[i] - (t.val * (sqrt(mse) * sqrt(i)))
}
UB <- c(u[1:t])
LB <- c(l[1:t])

LB1 <- c(x0cap[n],LB)
UB2 <- c(x0cap[n],UB)

l1 <- length(LB1)
d3 <- seq(1, l1, 1)
u1 <- length(UB2)
d4 <- seq(1, u1, 1)

set3 <- data.frame(x=d3, y=LB1)
set4 <- data.frame(x=d4, y=UB2)

d0 <- seq(1, n, 1)
xy1 <- data.frame(x=d0, y=x0)

d1 <- seq(1, w, 1)
xy2 <- data.frame(x=d1, y=x0cap2)

# Create data frame
df <- rbind(xy1, xy2, set3, set4)

# Plots
colors <- c("Raw Data"="red","Fitted&Forecasts"="blue","LowerBound"="green","UpperBound"="yellow")
CI <- c(n:w)

x=y=NULL

p <- ggplot(df) +
  theme_bw() +
 labs(title = 'EPGM (1, 1) model',x = 'Number of observation',y = 'Data Forecast & Prediction') +
  scale_x_continuous(breaks=1:w) +
  scale_y_continuous(labels = scales::comma) +
  geom_point(data = xy1, aes(x = x, y = y), shape = 24, color = "black") +
  geom_point(data = xy2, aes(x = x, y = y), shape = 21, color = "black") +
  geom_point(data = set3, aes(x = CI, y = y), shape = 23, color = "black") +
  geom_point(data = set4, aes(x = CI, y = y), shape = 23, color = "black") +
  geom_line(data = xy1, aes(x = x, y = y,color = "Raw Data")) +
  geom_line(data = xy2, aes(x = x, y = y,color = "Fitted&Forecasts")) +
  geom_line(data = set3, aes(x = CI, y = y,color = "LowerBound"), linetype=2) +
  geom_line(data = set4, aes(x = CI, y = y,color = "UpperBound"), linetype=2) +
  scale_color_manual(name = "Label",values = colors)
r <- ggplotly(p)
r
```

`ResidualModification` *Residual modification*

Description

A collection of grey forecasting models based on residual grey models.

Usage

```
remnantgm11(x0,x0_A)
tgm11(x0,x0_A)
```

Arguments

`x0`	Raw data (training set)
`x0_A`	Raw data (testing set)
`remnantgm11`	Residual-based grey model
`tgm11`	Trigonometric grey model

Value

fitted and predicted values

References

Hu Y (2020). Energy Demand Forecasting using a Novel Remnant GM (1, 1) Model. Soft Computing, pp. 13903-13912. DOI:10.1007/s00500-020-04765-3.

Zhou P, Ang B, Poh K (2006). A Trigonometric Grey Prediction Approach to Forecasting Electricity Demand. Energy, 31(14), 2839-2847. DOI:10.1016/j.energy.2005.12.002.

Examples

```
# TGM (1, 1) model: Trigonometric grey model

x0 <- c(2350,2465,2557,2577,2689,2739,2797,2885,2937,2996)
x0_A <- c(3042,3120,3132,3237)

x1 <- cumsum(x0)

n <- length(x0)

b <- numeric(n)
for (i in 1:n){
  b[i] <- -(0.5*x1[i + 1] + 0.5*x1[i])
}
b1 <- b[1:n-1]

B <- matrix(1,nrow=n-1,ncol=2)
B[,1] <- t(t(b1[1:n-1]))

yn <- matrix(c(x0),ncol=1)
yn <- t(t(x0[2:n]))

xcap <- solve (t(B) %*% B) %*% t(B) %*% yn
a <- xcap[1,1]
b <- xcap[2,1]

scale_with <- function(k)
{
  (x0[1] - (b/a)) * exp(-a*k) * (1 - exp(a))
}
fitted <- scale_with(1:n)

x0cap <- c(x0[1],fitted[1:n-1])

x0cap_GM <- c(x0cap)

n <- length(x0)

r0 <- numeric(n)

for (i in 1:n){
  r0[i] <-x0[i] - x0cap_GM[i]
}
R <- r0[2:n]

rn <- matrix(c(R),ncol=1)

m <- length(rn)

L <- 23

mat1 <- matrix(1,nrow=n-1,ncol=1)
```

```
mat2 <-matrix(1:m,nrow=m,ncol=1)

s <- replicate(n,0)
for (i in 1:n){
  s[i] <- sin( (2*(i-1)*pi)/L )
}
mat3 <- matrix(c(s[2:n]),ncol=1)

c <- replicate(n,0)
for (i in 1:n){
  c[i] <- cos( (2*(i-1)*pi)/L )
}
mat4 <- matrix(c(c[2:n]),ncol=1)

B <- cbind(mat1,mat2,mat3,mat4)

rcap <- (solve (t(B) %*% B)) %*% t(B) %*% rn
b0 <- rcap[1,1]
b1 <- rcap[2,1]
b2 <- rcap[3,1]
b3 <- rcap[4,1]

scale_with <- function(k)
{
  b0 + (b1*k) + (b2*sin( (2*pi*k)/L )) + (b3*cos( (2*pi*k)/L ))
}
forecast <- scale_with(1:m)

r0cap <- c(0,forecast)

xcap_tr <- r0cap + x0cap_GM

A <- 4
scale_with <- function(k)
{
  (x0[1] - (b/a)) * exp(-a*k) * (1 - exp(a))
}
fitted_a <- scale_with(1 : n+A-1)

x0cap_GMa <- c(fitted_a)

predicted_a <- tail(x0cap_GMa,A)

n_a <- length(x0_A)

r0_a <- numeric(n_a)
for (i in 1:n_a){
  r0_a[i] <-x0_A[i] - x0cap_GMa[i]
}
R_a <- r0_a[1:n_a]

rn_a <- matrix(c(R_a),ncol=1)
```

```
scale_with <- function(k)
{
  b0 + (b1*k) + (b2*sin( (2*pi*k)/L )) + (b3*cos( (2*pi*k)/L ))
}
forecast_a <- scale_with(1:m+A)

r0cap_a <- tail(forecast_a,A)

xcap_tra <- r0cap_a +  predicted_a

x0cap5 <- c(xcap_tra)
x0cap2 <- c(xcap_tr,x0cap5 )
# Fitted and predicted values
x0cap2
```

Index