

# Package ‘TAG’

January 20, 2025

**Type** Package

**Title** Transformed Additive Gaussian Processes

**Version** 0.5.1

**Author** Li-Hsiang Lin and V. Roshan Joseph

**Maintainer** Li-Hsiang Lin <llin79@gatech.edu>

**Description** Implement the transformed additive Gaussian (TAG) process and the transformed approximately additive Gaussian (TAAG) process proposed in Lin and Joseph (2020) <[DOI:10.1080/00401706.2019.1665592](https://doi.org/10.1080/00401706.2019.1665592)>. These functions can be used to model deterministic computer experiments, obtain predictions at new inputs, and quantify the uncertainty of the predictions. This research is supported by a U.S. National Science Foundation grant DMS-1712642 and a U.S. Army Research Office grant W911NF-17-1-0007.

**License** GPL-2

**Depends** R (>= 3.5.0)

**Imports** Rcpp, DiceKriging, Matrix, mgcv, FastGP, mlegp, randtoolbox, foreach

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 6.1.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2021-06-07 18:40:02 UTC

## Contents

check.TAAG	2
cv.TAAG	3
initial.TAG	4
plotTAG	5
pred.TAAG	6
pred.TAG	7
TAAG	8
TAG	10
TAG1step	11

**Index****14**

check.TAAG

*Check TAAG***Description**

This function provides a table for checking whether the TAAG process fits the data well better than an ordinary kriging model.

**Usage**

```
check.TAAG(object)
```

**Arguments**

object            object of class inheriting from "TAAG".

**Value**

A table of the fitted negative likelihood values and cross validation errors obtained from TAAG and the ordinary kriging models from dicekriging and mlegp is returned. Note that, for both criteria, the model with a smaller value are a better model.

**References**

Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.

**See Also**

[TAAG](#) for the estimates of the parameters in the TAG and TAAG, respectively.

**Examples**

```
n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
par.TAG <- TAG(ini.TAG)
N <- 1000
X.test <- sobol(N, dim = p, init = TRUE, scrambling = 2, seed = 5, normal = FALSE)
ytrue <- exp(2*sin(0.5*pi*X.test[,1]) + 0.5*cos(2.5*pi*X.test[,2]))
pre.TAG <- pred.TAG(par.TAG, X.test)
library(DiceKriging)
set.seed(2)
temp.m <- km(formula=~1, design=X, response=par.TAG$ty,
             covtype="gauss", nugget = (10^-15), multistart = 4,
```

```

      control = list(trace = FALSE))
nu.est <- sqrt(2*(coef(temp.m)$range^2))
par.TAAG <- TAAG(par.TAG, nu.est)

check.table <- check.TAAG(par.TAAG)
check.table

```

---

cv.TAAG

*Leave-One-Out Cross Validation Error of a TAG or TAAG Process*


---

### Description

This function evaluates the leave-one-out cross validation errors of a TAG or TAAG process.

### Usage

```
cv.TAAG(object, TAAG.indicator = FALSE)
```

### Arguments

**object**                object of class inheriting from "TAG" or "TAAG".  
**TAAG.indicator**    logical. If TRUE, the object is from TAAG; otherwise, from TAG.

### Value

The values returned from the function is a list containing:

**CV**                    Leave-one-out cross validation errors in the original scale of y.  
**TCV**                  Leave-one-out cross validation errors in the transformed scale.

### References

Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.

### See Also

[TAG](#) and [TAAG](#) for the estimates of the parameters in the TAG and TAAG, respectively.

### Examples

```

n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
par.TAG <- TAG(ini.TAG)
cv.scores <- cv.TAAG(par.TAG)
(CV.TAG <- cv.scores$CV)
(TCV.TAG <- cv.scores$TCV)

```

---

initial.TAG                      *Initial Values of TAG Process*

---

**Description**

This function generates good initial values for the parameters in a TAG process.

**Usage**

```
initial.TAG(y, X, Candi.lambda = seq(from=-2, to=2,by=0.5),
           Adj.omega = TRUE, nug=0.001, nbasis=10, rannum=20, big=FALSE, nsub=31,
           method.1d = "DiceKriging")
```

**Arguments**

y	a response vector of size n, where n is the sample size.
X	an n by p design matrix, where n is the sample size, and p is the number of input variables.
Candi.lambda	a vector containing the candidate values of the Box-Cox transformation parameter. Default is seq(from=-2, to=2,by=0.5).
Adj.omega	logical. If TRUE, the initial estimates for weight parameters are adjusted to avoid 0. Default is TRUE.
nug	a nonnegative value used as the nugget term for fitting the 1-dim GP models. Default is 0.001.
nbasis	a positive integer specifying the basis dimension used in mgcv. Default is 10.
rannum	a positive integer specifying the number of starting values in DiceKriging. Default is 2.
big	logical. If TRUE, the bam function in the mgcv package is used; otherwise, the gam function is used. Default is FALSE.
nsub	a positive integer specifying the number of design points used for obtaining the initial estimates of the length scale parameters. Default is 31.
method.1d	the method used for fitting the 1-dimensional GPs. Currently, the method can be DiceKriging or mlegp.

**Value**

The values returned from the function is a list containing the following components:

omega	The initial estimates of the weight parameters.
s	The initial estimates of the length scale parameters.
lambda	The initial estimate of the Box-Cox transformation parameter.
delta	The initial estimate of the nugget parameter.
nbases	The number of bases used in each dimension.
y	The response vector.
X	The n by p input design matrix.

## References

Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.

Olivier Roustant, David Ginsbourger, Yves Deville (2012). DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. *Journal of Statistical Software*, 51, 1-55.

Dancik, GM and Dorman, KS (2008). mlegp: Statistical analysis for computer models of biological systems using R. *Bioinformatics* 24, 1966-1967

Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R* (2nd edition). New York: CRC press.

## Examples

```
n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
```

```
#An example for some inputs with fewer levels
n <- 18
p <- 2
X1 <- rep(c(0,1,2)/3, 6) # A factor with fewer levels
library(randtoolbox)
X2 <- sobol(n, dim = 1, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
X <- cbind(X1, X2)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
```

---

plotTAG

*Main Effects Plots from a TAG Process*

---

## Description

This function produces the main effects plot from a TAG process.

## Usage

```
plotTAG(object, include.legend = TRUE, legend.position = "bottomright")
```

## Arguments

**object** object of class inheriting from "TAG".

**include.legend** logical value indicating if the legend of the main effects plot is required.

**legend.position** a character indicating the position of the legend.

**Details**

The location of the legend is "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", or "center".

**Value**

A main effects plot in the transformed scale of the response is returned.

**References**

Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.

**See Also**

[TAG](#) for finding estimates for the parameters in a TAG process.

**Examples**

```
n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
par.TAG <- TAG(ini.TAG)
plotTAG(par.TAG)
```

---

pred.TAAG

*Prediction from the TAAG Process*

---

**Description**

This function provides predictions from a TAAG process.

**Usage**

```
pred.TAAG(object, newX, predict.CI = FALSE, zalpha = 1.96)
```

**Arguments**

object	object of class inheriting from "TAAG".
newX	matrix of new values of x at which predictions are needed.
predict.CI	logical value indicating if the confidence interval at each prediction point is required.
zalpha	normal critical value for the confidence interval. Default is 1.96 for 95 % confidence intervals. The zalpha works only when predict.CI is TRUE.

**Value**

The function returns predictions at newX and the confidence intervals (if predict.CI is TRUE). If predict.CI is TRUE, the values returned from the function is a list containing:

Prediction	the prediction at newX.
ConfidenceLB	the lower bound of the prediction confidence interval at newX.(Note that the default is 95 %.)
ConfidenceUB	the upper bound of the prediction confidence interval at newX.

**References**

Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.

**See Also**

[TAAG](#) for the estimates of the parameters in the TAAG.

**Examples**

```
n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
par.TAG <- TAG(ini.TAG)
N <- 1000 # size of testing samples
X.test <- sobol(N, dim = p, init = TRUE, scrambling = 2, seed = 5, normal = FALSE)
ytrue <- exp(2*sin(0.5*pi*X.test[,1]) + 0.5*cos(2.5*pi*X.test[,2]))
pre.TAG <- pred.TAG(par.TAG, X.test)
library(DiceKriging)
set.seed(2)
temp.m <- km(formula=~1, design=X, response=par.TAG$ty,
  covtype="gauss",nugget = (10^-15), multistart = 4,
  control = list(trace = FALSE, verbose = FALSE))
nu.est <- sqrt(2*(coef(temp.m)$range^2))
par.TAAG <- TAAG(par.TAG, nu.est)
pre.TAAG <- pred.TAAG(par.TAAG, X.test)
mean((pre.TAAG$Prediction-ytrue)^2)
```

---

pred.TAG

*Prediction from the TAG process*

---

**Description**

This function provides predictions from a TAG process.

**Usage**

```
pred.TAG(object, newX)
```

**Arguments**

object	object of class inheriting from "TAG".
newX	matrix of new x values at which predictions are needed.

**Value**

The values returned from the function are the predictions at newX.

**References**

Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.

**See Also**

[TAG](#) for the estimates of the parameters in the TAG.

**Examples**

```
n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
par.TAG <- TAG(ini.TAG)
N <- 1000
X.test <- sobol(N, dim = p, init = TRUE, scrambling = 2, seed = 5, normal = FALSE)
ytrue <- exp(2*sin(0.5*pi*X.test[,1]) + 0.5*cos(2.5*pi*X.test[,2]))
pre.TAG <- pred.TAG(par.TAG, X.test)
```

---

 TAAG

---

*Transformed Approximately Additive Gaussian Process*


---

**Description**

This function fits the Transformed Approximately Additive Gaussian (TAAG) process.

**Usage**

```
TAAG(parTAG, nu.est, adj.nu = FALSE)
```



## Arguments

parTAG	object of class inheriting from "TAG".
nu.est	the estimates of the length scale parameters from a standard GP.
adj.nu	logical. If FALSE, the proportional parameter $\eta$ is estimated; otherwise, both $\eta$ and the multiplication factor $\phi$ are estimated. Default is FALSE.

## Details

The details of TAAG process can be found in Lin and Joseph (2019).

When the input dimension is high, set `adj.nu = TRUE` and `nu.est = s0`, where `s0` is the initial values of the length scale parameters from the function `initial.TAG`. Then, the length scale parameter  $\nu$  is set to  $\phi \times s0$ , and  $\phi$  is estimated through function `TAAG`. This is useful especially when the input dimension is high.

## Value

The values returned from the function is a list containing the following components:

omega	The estimates of the weight parameters.
s	The estimates of the length scale parameters.
nu	The estimates of the length scale parameter $\nu$ .
lambda	The estimate of the Box-Cox transformation parameter.
eta	The estimate of the proportion parameter.
phi	The estimate of the multiplication factor for $\nu$ , used for high dimensional data.
obj.fun	The negative of log-unnormalized posterior value (value of the objective function)
ty	The transformed response vector.
X	The n by p input design matrix.

## References

Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.

## See Also

[TAG](#) for the estimates of the TAG parameters, and [pred.TAAG](#) for predictions.

## Examples

```
n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
par.TAG <- TAG(ini.TAG)
```

```

N <- 1000
X.test <- sobol(N, dim = p, init = TRUE, scrambling = 2, seed = 5, normal = FALSE)
y.true <- exp(2*sin(0.5*pi*X.test[,1]) + 0.5*cos(2.5*pi*X.test[,2]))
pre.TAG <- pred.TAG(par.TAG, X.test)
library(DiceKriging)
set.seed(2)
temp.m <- km(formula=~1, design=X, response=par.TAG$ty,
             covtype="gauss",nugget = (10^-15), multistart = 4,
             control = list(trace = FALSE, verbose = FALSE))
nu.est <- sqrt(2*(coef(temp.m)$range^2))
par.TAAG <- TAAG(par.TAG, nu.est)

```

TAG

*Transformed Additive Gaussian Process***Description**

This function fits the Transformed Additive Gaussian (TAG) process.

**Usage**

```
TAG(iniTAG, HighD = FALSE, delta.threshold = -6)
```

**Arguments**

<code>iniTAG</code>	object of class inheriting from "initial.TAG".
<code>HighD</code>	logical. If TRUE, only $\kappa$ and delta will be estimated. This is useful for high dimensional data. Default is False.
<code>delta.threshold</code>	the minimum value of $\log_{10}(\text{delta})$ . Default is -6.

**Details**

The details of the TAG process can be found in Lin and Joseph (2019).

When `HighD = FALSE`, the weight parameters, the length scale parameters, the nugget parameter, and the Box-Cox transformation parameter are estimated. When `HighD = TRUE`, the length scale parameters for TAG is  $\eta*s_0$ , where  $s_0$  is the initial estimate of the length scale parameters. Only  $\eta$  and the nugget parameter are estimated.

**Value**

The values returned from the function is a list containing the following components:

<code>omega</code>	The estimates of the weight parameters.
<code>s</code>	The estimates of the length scale parameters.
<code>lambda</code>	The estimate of the Box-Cox transformation parameter.
<code>delta</code>	The estimate of the nugget parameter in $\log_{10}$ scale. For example, <code>delta = -6</code> means that the estimate of the nugget is $10^{(-6)}$ .

kappa	If HighD is true, an estimate of kappa will be returned, which is a multiplication factor for the initial estimates of the length scale parameters.
ty	The transformed response vector.
X	The n by p input design matrix.

## References

Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.

## See Also

[initial.TAG](#) for finding the initial values for the parameters in a TAG process, and [pred.TAG](#) for prediction.

## Examples

```
n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
ini.TAG <- initial.TAG(y, X)
par.TAG <- TAG(ini.TAG)
```

---

TAG1step

*Conducting All Steps for Building TAG Processes*

---

## Description

This function combines functions `initial.TAG`, `TAG`, and `TAAG` together, so it generates the estimated parameters from a completed TAG process. The returned object from `TAG1step` is in the same format as the one returned from `TAAG`, so the returned object can be passed to function `pred.TAAG` for predictions.

## Usage

```
TAG1step(y, X,
  set.initial = list(Candi.lambda= seq(from=-2, to=2,by=0.5)),
  set.TAG = list(delta.threshold = -6),
  set.TAAG = list(adj.nu = FALSE))
```

**Arguments**

<code>y</code>	a response vector of size $n$ , where $n$ is the sample size.
<code>X</code>	an $n$ by $p$ design matrix, where $n$ is the sample size, and $p$ is the number of input variables.
<code>set.initial</code>	a list for controlling other arguments in function <code>Initial.TAG</code> . See <code>Initial.TAG</code> for more details.
<code>set.TAG</code>	a list for controlling other arguments in function <code>TAG</code> . See <code>TAG</code> for more details.
<code>set.TAAG</code>	a list for controlling other arguments in function <code>TAAG</code> . See <code>TAAG</code> for more details.

**Value**

The values returned from the function is a list containing the following components:

<code>omega</code>	The estimates of the weight parameters.
<code>s</code>	The estimates of the length scale parameters.
<code>nu</code>	The estimates of the length scale parameter $\nu$ .
<code>lambda</code>	The estimate of the Box-Cox transformation parameter.
<code>eta</code>	The estimate of the proportion parameter.
<code>phi</code>	The estimate of the multiplication factor for $\nu$ , used for high dimensional data.
<code>obj.fun</code>	The negative of log-unnormalized posterior value (value of the objective function)
<code>ty</code>	The transformed response vector.
<code>X</code>	The $n$ by $p$ input design matrix.

**References**

- Lin, L.-H. and Joseph, V. R. (2020) "Transformation and Additivity in Gaussian Processes", *Technometrics*, 62, 525-535. DOI:10.1080/00401706.2019.1665592.
- Olivier Roustant, David Ginsbourger, Yves Deville (2012). DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. *Journal of Statistical Software*, 51, 1-55.
- Dancik, GM and Dorman, KS (2008). mlegp: Statistical analysis for computer models of biological systems using R. *Bioinformatics* 24, 1966-1967
- Wood, S.N. (2017) *Generalized Additive Models: An Introduction with R* (2nd edition). New York: CRC press.

**See Also**

[initial.TAG](#) for other arguments that can be listed in `set.initial` for controlling function `Initial.TAG`. [TAG](#) for other arguments that can be listed in `set.TAG` for controlling function `TAG`. [TAAG](#) for other arguments that can be listed in `set.TAAG` for controlling function `TAAG`. [pred.TAAG](#) for predictions.

**Examples**

```
n <- 20
p <- 2
library(randtoolbox)
X <- sobol(n, dim = p, init = TRUE, scrambling = 2, seed = 20, normal = FALSE)
y <- exp(2*sin(0.5*pi*X[,1]) + 0.5*cos(2.5*pi*X[,2]))
par.TAAG <- TAG1step(y, X)

#Predictions From TAG1step
N <- 1000 # size of testing samples
X.test <- sobol(N, dim = p, init = TRUE, scrambling = 2, seed = 5, normal = FALSE)
ytrue <- exp(2*sin(0.5*pi*X.test[,1]) + 0.5*cos(2.5*pi*X.test[,2]))
pre.TAAG <- pred.TAAG(par.TAAG, X.test)
mean((pre.TAAG$Prediction-ytrue)^2)
```

# Index

- \* **Completed TAG Processes**
    - TAG1step, [11](#)
  - \* **Cross Validation**
    - cv. TAAG, [3](#)
  - \* **Estimated Parameters**
    - TAAG, [8](#)
    - TAG, [10](#)
  - \* **Initial Values**
    - initial.TAG, [4](#)
  - \* **Main Effect Plots**
    - plotTAG, [5](#)
  - \* **Model Comparision**
    - check.TAAG, [2](#)
  - \* **Model Validation**
    - cv. TAAG, [3](#)
  - \* **OneStepTAG**
    - TAG1step, [11](#)
  - \* **Predictions**
    - pred. TAAG, [6](#)
    - pred. TAG, [7](#)
  - \* **TAAG**
    - plotTAG, [5](#)
    - pred. TAAG, [6](#)
    - TAAG, [8](#)
  - \* **TAG**
    - initial.TAG, [4](#)
    - plotTAG, [5](#)
    - pred. TAG, [7](#)
    - TAG, [10](#)
- check. TAAG, [2](#)  
cv. TAAG, [3](#)
- initial.TAG, [4](#), [11](#), [12](#)
- plotTAG, [5](#)  
pred. TAAG, [6](#), [9](#), [12](#)  
pred. TAG, [7](#), [11](#)
- TAAG, [2](#), [3](#), [7](#), [8](#), [12](#)
- TAG, [3](#), [6](#), [8](#), [9](#), [10](#), [12](#)  
TAG1step, [11](#)