Package 'IGP'

April 26, 2021

Type Package

Title Interchangeable Gaussian Process Models

Version 0.2.0

Description Creates a Gaussian process model using the specified package. Makes it easy to try different packages in same code, only the package argument needs to be changed. It is essentially a wrapper for the other Gaussian process software packages.

License GPL-3

Imports R6, reticulate

Suggests GPfit, laGP, mlegp, tgp, DiceKriging, CGP, GauPro, hetGP, testthat, lhs, ggplot2, reshape, ContourFunctions, numDeriv

RoxygenNote 7.1.1

URL https://github.com/CollinErickson/IGP

BugReports https://github.com/CollinErickson/IGP/issues

Encoding UTF-8

NeedsCompilation no

Author Collin Erickson [aut, cre]

Maintainer Collin Erickson <collinberickson@gmail.com>

Repository CRAN

Date/Publication 2021-04-26 10:00:19 UTC

R topics documented:

`IGP` *IGP general function*

Description

IGP general function

Usage

```
IGP(X = NULL, Z = NULL, package = NULL, ...)
```

Arguments

`X`	Design matrix
`Z`	Response matrix or vector
`package`	Package to use
`...`	Arguments passed on to IGP_<package>

Value

IGP model

Examples

```
x <- seq(0,1,l=10)
y <- abs(sin(2*pi*x))
IGP(x,y,'DiceKriging')
```

IGP_base *UGP Class providing object with methods for fitting a GP model*

Description

UGP Class providing object with methods for fitting a GP model

UGP Class providing object with methods for fitting a GP model

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/UGP/

`new(X=NULL, Z=NULL, package=NULL, corr="gauss", estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...` This method updates the model, adding new data if given, then running optimization again.

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

`N` Number of data points

`D` Dimension of data

Methods

Public methods:

- `IGP_base$.init()`
- `IGP_base$.update()`
- `IGP_base$.predict()`
- `IGP_base$.predict.se()`
- `IGP_base$.predict.var()`

- IGP_base$.delete()
- IGP_base$new()
- IGP_base$init()
- IGP_base$update()
- IGP_base$predict()
- IGP_base$predict.se()
- IGP_base$predict.var()
- IGP_base$grad()
- IGP_base$grad_num()
- IGP_base$grad_from_theta()
- IGP_base$grad_norm()
- IGP_base$sample()
- IGP_base$theta()
- IGP_base$nugget()
- IGP_base$s2()
- IGP_base$mean()
- IGP_base$max.var()
- IGP_base$at.max.var()
- IGP_base$prop.at.max.var()
- IGP_base$plot()
- IGP_base$delete()
- IGP_base$finalize()
- IGP_base$clone()

Method `.init()`**:**

Usage:

```
IGP_base$.init(...)
```

Method `.update()`**:**

Usage:

```
IGP_base$.update(...)
```

Method `.predict()`**:**

Usage:

```
IGP_base$.predict(...)
```

Method `.predict.se()`**:**

Usage:

```
IGP_base$.predict.se(...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_base$.predict.var(...)
```

Method `.delete()`**:**

Usage:

```
IGP_base$.delete(...)
```

Method `new()`**:**

Usage:

```
IGP_base$new(
  X = NULL,
  Z = NULL,
  package = NULL,
  corr = "gauss",
  estimate.nugget = TRUE,
  nugget0 = 1e-08,
  ...
)
```

Method `init()`**:**

Usage:

```
IGP_base$init(X = NULL, Z = NULL, ...)
```

Method `update()`**:**

Usage:

```
IGP_base$update(Xall = NULL, Zall = NULL, Xnew = NULL, Znew = NULL, ...)
```

Method `predict()`**:**

Usage:

```
IGP_base$predict(XX, se.fit = FALSE, ...)
```

Method `predict.se()`**:**

Usage:

```
IGP_base$predict.se(XX, ...)
```

Method `predict.var()`**:**

Usage:

```
IGP_base$predict.var(XX, ...)
```

Method `grad()`**:**

Usage:

```
IGP_base$grad(XX, num = FALSE)
```

Method `grad_num()`**:**

Usage:

```
IGP_base$grad_num(XX)
```

Method `grad_from_theta()`**:**

Usage:

```
IGP_base$grad_from_theta(XX, theta)
```

Method `grad_norm()`**:**

Usage:

```
IGP_base$grad_norm(XX)
```

Method `sample()`**:**

Usage:

```
IGP_base$sample(XX, n = 1)
```

Method `theta()`**:**

Usage:

```
IGP_base$theta()
```

Method `nugget()`**:**

Usage:

```
IGP_base$nugget()
```

Method `s2()`**:**

Usage:

```
IGP_base$s2()
```

Method `mean()`**:**

Usage:

```
IGP_base$mean()
```

Method `max.var()`**:**

Usage:

```
IGP_base$max.var()
```

Method `at.max.var()`**:**

Usage:

```
IGP_base$at.max.var(X, val = 0.9)
```

Method `prop.at.max.var()`**:**

Usage:

```
IGP_base$prop.at.max.var(
  Xlims = matrix(c(0, 1), nrow = ncol(self$X), ncol = 2, byrow = T),
  n = 200,
  val = 0.9
)
```

Method `plot()`**:**

Usage:

```
IGP_base$plot()
```

Method `delete()`**:**

Usage:

```
IGP_base$delete(...)
```

Method `finalize()`**:**

Usage:

```
IGP_base$finalize(...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_base$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP(package='laGP',X=X1,Z=Z1, corr="gauss")
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()
```

IGP_CGP *IGP R6 object for fitting CGP model*

Description

Class providing object with methods for fitting a GP model

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

`IGP::IGP` -> `IGP_CGP`

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Methods

Public methods:

- `IGP_CGP$.init()`
- `IGP_CGP$.update()`
- `IGP_CGP$.predict()`
- `IGP_CGP$.predict.se()`
- `IGP_CGP$.predict.var()`
- `IGP_CGP$.delete()`
- `IGP_CGP$.theta()`
- `IGP_CGP$.nugget()`
- `IGP_CGP$.s2()`
- `IGP_CGP$.mean()`
- `IGP_CGP$clone()`

Method `.init()`:

Usage:

```
IGP_CGP$.init(...)
```

Method `.update()`**:**

Usage:

```
IGP_CGP$.update(...)
```

Method `.predict()`**:**

Usage:

```
IGP_CGP$.predict(XX, se.fit, ...)
```

Method `.predict.se()`**:**

Usage:

```
IGP_CGP$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_CGP$.predict.var(XX, ...)
```

Method `.delete()`**:**

Usage:

```
IGP_CGP$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_CGP$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_CGP$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_CGP$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_CGP$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_CGP$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

References

Ba, S. and V. Roshan Joseph (2012) "Composite Gaussian Process Models for Emulating Expensive Functions". Annals of Applied Statistics, 6, 1838-1860.

Examples

```
# Takes 17 seconds
n <- 20
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_CGP$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
cbind(u$predict(XX1), ZZ1)
u$delete()
```

IGP_DiceKriging *IGP R6 object for fitting DiceKriging model*

Description

Class providing object with methods for fitting a GP model

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

`IGP::IGP` -> `IGP_DiceKriging`

Public fields

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Active bindings

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Methods

Public methods:

- `IGP_DiceKriging$.init()`
- `IGP_DiceKriging$.update()`
- `IGP_DiceKriging$.predict()`
- `IGP_DiceKriging$.predict.se()`
- `IGP_DiceKriging$.predict.var()`
- `IGP_DiceKriging$.delete()`
- `IGP_DiceKriging$.theta()`
- `IGP_DiceKriging$.nugget()`
- `IGP_DiceKriging$.s2()`
- `IGP_DiceKriging$.mean()`
- `IGP_DiceKriging$clone()`

Method `.init()`:

Usage:

```
IGP_DiceKriging$.init(...)
```

Method `.update()`:

Usage:

```
IGP_DiceKriging$.update(...)
```

Method `.predict()`:

Usage:

```
IGP_DiceKriging$.predict(XX, se.fit, ...)
```

Method `.predict.se()`:

Usage:

```
IGP_DiceKriging$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_DiceKriging$.predict.var(XX, ...)
```

Method `.delete()`**:**

Usage:

```
IGP_DiceKriging$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_DiceKriging$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_DiceKriging$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_DiceKriging$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_DiceKriging$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_DiceKriging$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_DiceKriging$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()
```

IGP_GauPro *IGP R6 object for fitting GauPro model*

Description

Class providing object with methods for fitting a GP model

Format

R6Class object.

Value

Object of R6Class with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

IGP::IGP -> IGP_GauPro

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Methods

Public methods:

- IGP_GauPro$.init()
- IGP_GauPro$.update()
- IGP_GauPro$.predict()
- IGP_GauPro$.predict.se()
- IGP_GauPro$.predict.var()
- IGP_GauPro$.grad()
- IGP_GauPro$.delete()
- IGP_GauPro$.theta()
- IGP_GauPro$.nugget()
- IGP_GauPro$.s2()
- IGP_GauPro$.mean()
- IGP_GauPro$clone()

Method .init():

Usage:

```
IGP_GauPro$.init(...)
```

Method .update():

Usage:

```
IGP_GauPro$.update(...)
```

Method .predict():

Usage:

```
IGP_GauPro$.predict(XX, se.fit, ...)
```

Method .predict.se():

Usage:

```
IGP_GauPro$.predict.se(XX, ...)
```

Method .predict.var():

Usage:

```
IGP_GauPro$.predict.var(XX, ...)
```

Method .grad():

Usage:

```
IGP_GauPro$.grad(XX)
```

Method .delete():

Usage:

```
IGP_GauPro$.delete(...)
```

Method .theta():

Usage:

```
IGP_GauPro$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_GauPro$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_GauPro$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_GauPro$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_GauPro$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_GauPro$new(X=X1,Z=Z1, parallel=FALSE)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()
```

IGP_GauPro_kernel *IGP R6 object for fitting GauPro model*

Description

Class providing object with methods for fitting a GP model

Format

R6Class object.

Value

Object of R6Class with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

IGP::IGP -> IGP_GauPro_kernel

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Methods

Public methods:

- IGP_GauPro_kernel$.init()
- IGP_GauPro_kernel$.update()
- IGP_GauPro_kernel$.predict()
- IGP_GauPro_kernel$.predict.se()
- IGP_GauPro_kernel$.predict.var()
- IGP_GauPro_kernel$.grad()
- IGP_GauPro_kernel$.delete()
- IGP_GauPro_kernel$.theta()
- IGP_GauPro_kernel$.nugget()
- IGP_GauPro_kernel$.s2()
- IGP_GauPro_kernel$.mean()
- IGP_GauPro_kernel$clone()

Method .init():

Usage:

```
IGP_GauPro_kernel$.init(..., kernel = NULL, theta = NULL)
```

Method .update():

Usage:

```
IGP_GauPro_kernel$.update(...)
```

Method .predict():

Usage:

```
IGP_GauPro_kernel$.predict(XX, se.fit, ...)
```

Method .predict.se():

Usage:

```
IGP_GauPro_kernel$.predict.se(XX, ...)
```

Method .predict.var():

Usage:

```
IGP_GauPro_kernel$.predict.var(XX, ...)
```

Method .grad():

Usage:

```
IGP_GauPro_kernel$.grad(XX)
```

Method .delete():

Usage:

```
IGP_GauPro_kernel$.delete(...)
```

Method .theta():

Usage:

```
IGP_GauPro_kernel$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_GauPro_kernel$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_GauPro_kernel$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_GauPro_kernel$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_GauPro_kernel$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_GauPro_kernel$new(X=X1,Z=Z1, parallel=FALSE)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()
```

IGP_GPfit *IGP R6 object for fitting GPfit model*

Description

Class providing object with methods for fitting a GP model

Format

R6Class object.

Value

Object of R6Class with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

IGP::IGP -> IGP_GPfit

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Methods

Public methods:

- IGP_GPfit$.init()
- IGP_GPfit$.update()
- IGP_GPfit$.predict()
- IGP_GPfit$.predict.se()
- IGP_GPfit$.predict.var()
- IGP_GPfit$.delete()
- IGP_GPfit$.theta()
- IGP_GPfit$.nugget()
- IGP_GPfit$.s2()
- IGP_GPfit$.mean()
- IGP_GPfit$clone()

Method `.init()`**:**

Usage:

```
IGP_GPfit$.init(...)
```

Method `.update()`**:**

Usage:

```
IGP_GPfit$.update(...)
```

Method `.predict()`**:**

Usage:

```
IGP_GPfit$.predict(XX, se.fit, ...)
```

Method `.predict.se()`**:**

Usage:

```
IGP_GPfit$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_GPfit$.predict.var(XX, ...)
```

Method `.delete()`**:**

Usage:

```
IGP_GPfit$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_GPfit$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_GPfit$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_GPfit$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_GPfit$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_GPfit$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
n <- 20
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_GPfit$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)

u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)

u$delete()
```

`IGP_GPy` *IGP R6 object for fitting GPy model*

Description

Class providing object with methods for fitting a GP model

Format

`R6Class` object.

Value

Object of R6Class with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

`IGP::IGP` -> `IGP_GPy`

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Methods

Public methods:

- `IGP_GPy$.init()`
- `IGP_GPy$.update()`
- `IGP_GPy$.predict()`
- `IGP_GPy$.predict.se()`
- `IGP_GPy$.predict.var()`
- `IGP_GPy$.delete()`
- `IGP_GPy$.theta()`
- `IGP_GPy$.nugget()`
- `IGP_GPy$clone()`

Method `.init()`:

Usage:

```
IGP_GPy$.init(...)
```

Method `.update()`**:**

Usage:

```
IGP_GPy$.update(...)
```

Method `.predict()`**:**

Usage:

```
IGP_GPy$.predict(XX, se.fit, ...)
```

Method `.predict.se()`**:**

Usage:

```
IGP_GPy$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_GPy$.predict.var(XX, ...)
```

Method `.delete()`**:**

Usage:

```
IGP_GPy$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_GPy$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_GPy$.nugget()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_GPy$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
# Require numpy and GPy in Python, called with R package reticulate
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
```

```
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_GPy$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$predict.var(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()

## End(Not run)
```

IGP_hetGP *IGP R6 object for fitting hetGP model*

Description

Class providing object with methods for fitting a GP model

Format

R6Class object.

Value

Object of R6Class with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

IGP::IGP -> IGP_hetGP

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Methods

Public methods:

- `IGP_hetGP$.init()`
- `IGP_hetGP$.update()`
- `IGP_hetGP$.predict()`
- `IGP_hetGP$.predict.se()`
- `IGP_hetGP$.predict.var()`
- `IGP_hetGP$.grad()`
- `IGP_hetGP$.delete()`
- `IGP_hetGP$.theta()`
- `IGP_hetGP$.nugget()`
- `IGP_hetGP$.s2()`
- `IGP_hetGP$.mean()`
- `IGP_hetGP$clone()`

Method `.init()`:

Usage:

```
IGP_hetGP$.init(
  ...,
  kernel = "Gaussian",
  nug.est = self$estimate.nugget,
  nug = self$nugget0,
 noiseControl = list(k_theta_g_bounds = c(1, 100), g_max = 100, g_bounds = c(1e-06,
    1)),
  lower = rep(1e-06, ncol(self$X)),
  upper = rep(50, ncol(self$X))
)
```

Method `.update()`:

Usage:

```
IGP_hetGP$.update(...)
```

Method `.predict()`:

Usage:

```
IGP_hetGP$.predict(XX, se.fit, ...)
```

Method `.predict.se()`:

Usage:

```
IGP_hetGP$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_hetGP$.predict.var(XX, ...)
```

Method `.grad()`**:**

Usage:

```
IGP_hetGP$.grad(XX)
```

Method `.delete()`**:**

Usage:

```
IGP_hetGP$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_hetGP$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_hetGP$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_hetGP$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_hetGP$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_hetGP$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {x[1]*sin(2*pi*x[1]) + sqrt(x[1])*exp(x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-2)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
```

```
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_hetGP$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)

ContourFunctions::cf(function(x) u$predict(x), pts=X1)
ContourFunctions::cf(function(x) u$predict(x, se.fit=TRUE)$se, pts=X1)

u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
ContourFunctions::cf(function(x) u$predict(x), pts=rbind(X1, X2))
u$delete()

n <- 10
d <- 1
X1 <- runif(n)
X1 <- c(X1, X1)
f1 <- function(x) {abs(sin(pi*x))+sqrt(x)+rnorm(1,0,.1)}
Z1 <- sapply(X1, f1)
plot(X1, Z1)
h1 <- hetGP::mleHetGP(X=matrix(X1,ncol=1),Z=matrix(Z1, ncol=1), lower=c(.1), upper=c(50))
curve(predict(h1, matrix(x, ncol=1))$mean, add=TRUE, col=3)
u <- IGP_hetGP$new(X=X1,Z=Z1)
plot(X1, Z1, col=2); curve(u$predict(matrix(x,ncol=1)), add=TRUE)
```

`IGP_laGP` *IGP R6 object for fitting laGP model*

Description

Class providing object with methods for fitting a GP model

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

IGP::IGP -> IGP_laGP

Public fields

- X Design matrix
- Z Responses
- N Number of data points
- D Dimension of data

Active bindings

- X Design matrix
- Z Responses
- N Number of data points
- D Dimension of data

Methods

Public methods:

- IGP_laGP$.init()
- IGP_laGP$.update()
- IGP_laGP$.predict()
- IGP_laGP$.predict.se()
- IGP_laGP$.predict.var()
- IGP_laGP$.delete()
- IGP_laGP$.theta()
- IGP_laGP$.nugget()
- IGP_laGP$.s2()
- IGP_laGP$.mean()
- IGP_laGP$clone()

Method .init():

Usage:

```
IGP_laGP$.init(
  ...,
  d = NULL,
  g = NULL,
  theta = NULL,
  nugget = NULL,
  no_update = FALSE
)
```

Method .update():

Usage:

```
IGP_laGP$.update(..., no_update = FALSE)
```

Method `.predict()`**:**

Usage:

```
IGP_laGP$.predict(XX, se.fit, ...)
```

Method `.predict.se()`**:**

Usage:

```
IGP_laGP$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_laGP$.predict.var(XX, ...)
```

Method `.delete()`**:**

Usage:

```
IGP_laGP$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_laGP$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_laGP$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_laGP$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_laGP$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_laGP$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_laGP$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()
```

IGP_laGP_GauPro *IGP R6 object for fitting laGP_GauPro model*

Description

Class providing object with methods for fitting a GP model. This mixes laGP and GauPro. It fits the model using laGP, then copies the parameters to a GauPro model for prediction.

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

`IGP::IGP` -> `IGP_laGP_GauPro`

Public fields

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Active bindings

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Methods

Public methods:

- `IGP_laGP_GauPro$.init()`
- `IGP_laGP_GauPro$.update()`
- `IGP_laGP_GauPro$.predict()`
- `IGP_laGP_GauPro$.predict.se()`
- `IGP_laGP_GauPro$.predict.var()`
- `IGP_laGP_GauPro$.grad()`
- `IGP_laGP_GauPro$.delete()`
- `IGP_laGP_GauPro$.theta()`
- `IGP_laGP_GauPro$.nugget()`
- `IGP_laGP_GauPro$.s2()`
- `IGP_laGP_GauPro$.mean()`
- `IGP_laGP_GauPro$clone()`

Method `.init()`:

Usage:

```
IGP_laGP_GauPro$.init(...)
```

Method `.update()`:

Usage:

```
IGP_laGP_GauPro$.update(..., no_update = FALSE)
```

Method `.predict()`:

Usage:

```
IGP_laGP_GauPro$.predict(XX, se.fit, ...)
```

Method `.predict.se()`:

Usage:

```
IGP_laGP_GauPro$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_laGP_GauPro$.predict.var(XX, ...)
```

Method `.grad()`**:**

Usage:

```
IGP_laGP_GauPro$.grad(XX)
```

Method `.delete()`**:**

Usage:

```
IGP_laGP_GauPro$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_laGP_GauPro$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_laGP_GauPro$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_laGP_GauPro$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_laGP_GauPro$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_laGP_GauPro$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_laGP_GauPro$new(X=X1,Z=Z1)
```

```
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()
```

IGP_laGP_GauPro_kernel

IGP R6 object for fitting laGP_GauPro_kernel model

Description

Class providing object with methods for fitting a GP model. This mixes laGP and GauPro with a kernel. It fits the model using laGP, then copies the parameters to a GauPro model for prediction.

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

`IGP::IGP` -> `IGP_laGP_GauPro_kernel`

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Methods

Public methods:

- `IGP_laGP_GauPro_kernel$.init()`
- `IGP_laGP_GauPro_kernel$.update()`
- `IGP_laGP_GauPro_kernel$.predict()`
- `IGP_laGP_GauPro_kernel$.predict.se()`
- `IGP_laGP_GauPro_kernel$.predict.var()`
- `IGP_laGP_GauPro_kernel$.grad()`
- `IGP_laGP_GauPro_kernel$.delete()`
- `IGP_laGP_GauPro_kernel$.theta()`
- `IGP_laGP_GauPro_kernel$.nugget()`
- `IGP_laGP_GauPro_kernel$.s2()`
- `IGP_laGP_GauPro_kernel$.mean()`
- `IGP_laGP_GauPro_kernel$clone()`

Method `.init()`:

Usage:

```
IGP_laGP_GauPro_kernel$.init(...)
```

Method `.update()`:

Usage:

```
IGP_laGP_GauPro_kernel$.update(..., no_update = FALSE)
```

Method `.predict()`:

Usage:

```
IGP_laGP_GauPro_kernel$.predict(XX, se.fit, ...)
```

Method `.predict.se()`:

Usage:

```
IGP_laGP_GauPro_kernel$.predict.se(XX, ...)
```

Method `.predict.var()`:

Usage:

```
IGP_laGP_GauPro_kernel$.predict.var(XX, ...)
```

Method `.grad()`:

Usage:

```
IGP_laGP_GauPro_kernel$.grad(XX)
```

Method `.delete()`**:**

Usage:

```
IGP_laGP_GauPro_kernel$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_laGP_GauPro_kernel$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_laGP_GauPro_kernel$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_laGP_GauPro_kernel$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_laGP_GauPro_kernel$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_laGP_GauPro_kernel$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_laGP_GauPro_kernel$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
c(u$mod.extra$laGP$theta(), u$mod.extra$laGP$nugget())
c(u$mod.extra$GauPro$theta(), u$mod.extra$GauPro$nugget())
u$delete()
```

IGP_LOOEC_GauPro_kernel *IGP R6 object for fitting GauPro model*

Description

Class providing object with methods for fitting a GP model

Format

R6Class object.

Value

Object of R6Class with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

IGP::IGP -> IGP_LOOEC_GauPro_kernel

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Methods

Public methods:

- IGP_LOOEC_GauPro_kernel$.init()
- IGP_LOOEC_GauPro_kernel$.update()
- IGP_LOOEC_GauPro_kernel$.predict()
- IGP_LOOEC_GauPro_kernel$.predict.se()
- IGP_LOOEC_GauPro_kernel$.predict.var()
- IGP_LOOEC_GauPro_kernel$.grad()
- IGP_LOOEC_GauPro_kernel$.delete()
- IGP_LOOEC_GauPro_kernel$.theta()
- IGP_LOOEC_GauPro_kernel$.nugget()
- IGP_LOOEC_GauPro_kernel$.s2()
- IGP_LOOEC_GauPro_kernel$.mean()
- IGP_LOOEC_GauPro_kernel$clone()

Method .init():

Usage:

```
IGP_LOOEC_GauPro_kernel$.init(..., kernel = NULL, theta = NULL)
```

Method .update():

Usage:

```
IGP_LOOEC_GauPro_kernel$.update(...)
```

Method .predict():

Usage:

```
IGP_LOOEC_GauPro_kernel$.predict(XX, se.fit, ...)
```

Method .predict.se():

Usage:

```
IGP_LOOEC_GauPro_kernel$.predict.se(XX, ...)
```

Method .predict.var():

Usage:

```
IGP_LOOEC_GauPro_kernel$.predict.var(XX, ...)
```

Method .grad():

Usage:

```
IGP_LOOEC_GauPro_kernel$.grad(XX)
```

Method .delete():

Usage:

```
IGP_LOOEC_GauPro_kernel$.delete(...)
```

Method .theta():

Usage:

```
IGP_LOOEC_GauPro_kernel$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_LOOEC_GauPro_kernel$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_LOOEC_GauPro_kernel$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_LOOEC_GauPro_kernel$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_LOOEC_GauPro_kernel$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
n <- 30
d <- 2
n2 <- 10
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_LOOEC_GauPro_kernel$new(X=X1,Z=Z1, parallel=FALSE)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)

u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)

u$delete()


# 1D example to see difference
n <- 9
d <- 1
n2 <- 20
f1 <- function(x) {x^2 * sin(2*pi*x)}
X1 <- matrix(seq(0,1,l=n),n,d)
```

```
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-1)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_LOOEC_GauPro_kernel$new(X=X1,Z=Z1, parallel=FALSE)
u$plot()
u$mod$tmod$plot1D()
u$update(Xnew=X2,Znew=Z2)
u$plot()
u$mod$tmod$plot1D()
u$delete()
```

`IGP_LOOEC_laGP_GauPro` *IGP R6 object for fitting laGP_GauPro model with leave-one-out error correction*

Description

Class providing object with methods for fitting a GP model. This mixes laGP and GauPro. It fits the model using laGP, then copies the parameters to a GauPro model for prediction. The predicted errors are adjusted by fitting a third GP model to the leave-one-out absolute t-values.

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

`IGP::IGP` -> `IGP_LOOEC_laGP_GauPro`

Public fields

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Active bindings

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Methods

Public methods:

- `IGP_LOOEC_laGP_GauPro$new()`
- `IGP_LOOEC_laGP_GauPro$.init()`
- `IGP_LOOEC_laGP_GauPro$.update()`
- `IGP_LOOEC_laGP_GauPro$.predict()`
- `IGP_LOOEC_laGP_GauPro$.predict.se()`
- `IGP_LOOEC_laGP_GauPro$.predict.var()`
- `IGP_LOOEC_laGP_GauPro$.grad()`
- `IGP_LOOEC_laGP_GauPro$.delete()`
- `IGP_LOOEC_laGP_GauPro$.theta()`
- `IGP_LOOEC_laGP_GauPro$.nugget()`
- `IGP_LOOEC_laGP_GauPro$clone()`

Method `new()`:

Usage:

```
IGP_LOOEC_laGP_GauPro$new(
  X = NULL,
  Z = NULL,
  package = NULL,
  corr = "gauss",
  estimate.nugget = TRUE,
  nugget0 = 1e-08,
  package2 = NULL,
  ...
)
```

Method `.init()`:

Usage:

```
IGP_LOOEC_laGP_GauPro$.init(..., package2)
```

Method `.update()`**:**

Usage:

```
IGP_LOOEC_laGP_GauPro$.update(..., no_update = FALSE)
```

Method `.predict()`**:**

Usage:

```
IGP_LOOEC_laGP_GauPro$.predict(XX, se.fit, ...)
```

Method `.predict.se()`**:**

Usage:

```
IGP_LOOEC_laGP_GauPro$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_LOOEC_laGP_GauPro$.predict.var(XX, ...)
```

Method `.grad()`**:**

Usage:

```
IGP_LOOEC_laGP_GauPro$.grad(XX)
```

Method `.delete()`**:**

Usage:

```
IGP_LOOEC_laGP_GauPro$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_LOOEC_laGP_GauPro$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_LOOEC_laGP_GauPro$.nugget()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_LOOEC_laGP_GauPro$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_LOOEC_laGP_GauPro$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()
```

IGP_mlegp *IGP R6 object for fitting mlegp model*

Description

Class providing object with methods for fitting a GP model

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

`IGP::IGP` -> `IGP_mlegp`

Public fields

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Active bindings

- `X` Design matrix
- `Z` Responses
- `N` Number of data points
- `D` Dimension of data

Methods

Public methods:

- `IGP_mlegp$.init()`
- `IGP_mlegp$.update()`
- `IGP_mlegp$.predict()`
- `IGP_mlegp$.predict.se()`
- `IGP_mlegp$.predict.var()`
- `IGP_mlegp$.delete()`
- `IGP_mlegp$.theta()`
- `IGP_mlegp$.nugget()`
- `IGP_mlegp$.s2()`
- `IGP_mlegp$.mean()`
- `IGP_mlegp$clone()`

Method `.init()`:

Usage:

```
IGP_mlegp$.init(...)
```

Method `.update()`:

Usage:

```
IGP_mlegp$.update(...)
```

Method `.predict()`:

Usage:

```
IGP_mlegp$.predict(XX, se.fit, ...)
```

Method `.predict.se()`:

Usage:

```
IGP_mlegp$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_mlegp$.predict.var(XX, ...)
```

Method `.delete()`**:**

Usage:

```
IGP_mlegp$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_mlegp$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_mlegp$.nugget()
```

Method `.s2()`**:**

Usage:

```
IGP_mlegp$.s2()
```

Method `.mean()`**:**

Usage:

```
IGP_mlegp$.mean()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_mlegp$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_mlegp$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()
```

IGP_sklearn *IGP R6 object for fitting sklearn model*

Description

Class providing object with methods for fitting a GP model

Format

R6Class object.

Value

Object of R6Class with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

IGP::IGP -> IGP_sklearn

Public fields

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Active bindings

`X` Design matrix

`Z` Responses

`N` Number of data points

`D` Dimension of data

Methods

Public methods:

- IGP_sklearn$.init()
- IGP_sklearn$.update()
- IGP_sklearn$.predict()
- IGP_sklearn$.predict.se()
- IGP_sklearn$.predict.var()
- IGP_sklearn$.delete()
- IGP_sklearn$.theta()
- IGP_sklearn$.nugget()
- IGP_sklearn$clone()

Method `.init()`**:**

Usage:

```
IGP_sklearn$.init(...)
```

Method `.update()`**:**

Usage:

```
IGP_sklearn$.update(...)
```

Method `.predict()`**:**

Usage:

```
IGP_sklearn$.predict(XX, se.fit, ...)
```

Method `.predict.se()`**:**

Usage:

```
IGP_sklearn$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_sklearn$.predict.var(XX, ...)
```

Method `.delete()`**:**

Usage:

```
IGP_sklearn$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_sklearn$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_sklearn$.nugget()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_sklearn$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
# Require sklearn in Python, called with R package reticulate
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_sklearn$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)
u$predict.var(XX1)
u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)
u$delete()

## End(Not run)
```

IGP_tgp *IGP R6 object for fitting tgp model*

Description

Class providing object with methods for fitting a GP model

Format

`R6Class` object.

Value

Object of `R6Class` with methods for fitting GP model.

Methods

Documentation For full documentation of each method go to https://github.com/CollinErickson/IGP/

`new(X=NULL, Z=NULL, package=NULL, estimate.nugget=T, nugget0=F, ...)` This method is used to create object of this class with `X` and `Z` as the data. The package tells it which package to fit the GP model.

`update(Xall=NULL, Zall=NULL, Xnew=NULL, Znew=NULL, ...)` This method updates the model, adding new data if given, then running optimization again.

Super class

IGP::IGP -> IGP_tgp

Public fields

- X Design matrix
- Z Responses
- N Number of data points
- D Dimension of data

Active bindings

- X Design matrix
- Z Responses
- N Number of data points
- D Dimension of data

Methods

Public methods:

- IGP_tgp$.init()
- IGP_tgp$.update()
- IGP_tgp$.predict()
- IGP_tgp$.predict.se()
- IGP_tgp$.predict.var()
- IGP_tgp$.delete()
- IGP_tgp$.theta()
- IGP_tgp$.nugget()
- IGP_tgp$clone()

Method .init():

Usage:

```
IGP_tgp$.init(package = self$package, ...)
```

Method .update():

Usage:

```
IGP_tgp$.update(...)
```

Method .predict():

Usage:

```
IGP_tgp$.predict(XX, se.fit, ...)
```

Method .predict.se():

Usage:

```
IGP_tgp$.predict.se(XX, ...)
```

Method `.predict.var()`**:**

Usage:

```
IGP_tgp$.predict.var(XX, ...)
```

Method `.delete()`**:**

Usage:

```
IGP_tgp$.delete(...)
```

Method `.theta()`**:**

Usage:

```
IGP_tgp$.theta()
```

Method `.nugget()`**:**

Usage:

```
IGP_tgp$.nugget()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
IGP_tgp$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
n <- 40
d <- 2
n2 <- 20
f1 <- function(x) {sin(2*pi*x[1]) + sin(2*pi*x[2])}
X1 <- matrix(runif(n*d),n,d)
Z1 <- apply(X1,1,f1) + rnorm(n, 0, 1e-3)
X2 <- matrix(runif(n2*d),n2,d)
Z2 <- apply(X2,1,f1)
XX1 <- matrix(runif(10),5,2)
ZZ1 <- apply(XX1, 1, f1)
u <- IGP_tgp$new(X=X1,Z=Z1)
cbind(u$predict(XX1), ZZ1)
u$predict.se(XX1)

u$update(Xnew=X2,Znew=Z2)
u$predict(XX1)

u$delete()
```

predict.IGP *Predict for class IGP*

Description

Predict for class IGP

Usage

```
## S3 method for class 'IGP'
predict(object, XX, se.fit = FALSE, ...)
```

Arguments

object	Object of class IGP
XX	Points to predict at
se.fit	Whether the standard error prediction should be returned with the mean prediction
...	Additional parameters

Value

Prediction from object at XX

Examples

```
n <- 12
x <- matrix(seq(0,1,length.out = n), ncol=1)
y <- sin(2*pi*x) + rnorm(n,0,1e-1)
gp <- IGP(package='laGP', X=x, Z=y, parallel=FALSE)
predict(gp, .448)
```

Index