CIROQUO

libKriging - Install & basics

Y. Richet, P. Havé, Y. Deville

04/2022

References

Main **entry point**:

https://github.com/libKriging/libKriging

For **OS** specific issues:

https://github.com/libKriging/libKriging/tree/master/docs/dev/envs

Comaptibility matrix

	Linux Ubuntu:20	macOS 10 & 11 (x86-64)	macOS 12 (ARM)**	Windows 10
Python	✔ 3.6-3.10	✔ 3.6-3.10	✔ 3.9	✔ 3.6-3.9
R	✔ 3.6-4.1	✔ 3.6-4.1		✔ 3.6-4.1
Octave	✔ 5.2.0	✔ 6.2	✔ 6.4	✔ 5.2, ✘6.2
Matlab	✔ R2022	✔ R2022**	✘ R2022	✔ R2022**

'*': requires extra DLL'**': no pre-built packages

Get it

1. Go to: https://github.com/libKriging/libKriging/releases
2. Get latest package:

- R:
  - Windows: **r**libkriging_0.5.0.**zip**
  - Linux: **r**libkriging_0.5.0_Linux-x86_64.**tgz**
  - macOS: **r**libkriging_0.5.0_macOS10.15.7-x86_64.**tgz**
- Matlab/Octave:
  - Windows: **m**libKriging-for-**matlab**_0.5.0_**Windows10**.zip / **m**LibKriging_0.5.0_**MINGW64**_NT10.0-x86_64.tgz
  - Linux: **m**LibKriging_0.5.0_**Linux**-x86_64.tgz
  - macOS: **m**LibKriging_0.5.0_**macOS**10.15.7-x86_64.tgz
- Python:
  - Windows: **py**libkriging-0.5.0-cp**XX**-**win**_amd64.whl
  - Linux: **py**libkriging-0.5.0-cp**XX**-**manylinux**_*.whl
  - macOS: **py**libkriging-0.5.0-cp**XX**-**macosx**_10_15_x86_64.whl
- Otherwise: http://hub.irsn.cloud/ciroquo (Linux/R, Python, Matlab)

Install

```
install.packages("Rcpp")
install.packages(list.files(pattern = "rlibkriging_0.5.0(.*)"), repos=NULL)


# decompress dowloaded file, then
addpath("path/to/mlibkriging")


#pip3 install pylibkriging-0.5.0*.whl
import subprocess
subprocess.check_call([sys.executable, "-m", "pip", "install", "pylibkriging-0.5.0*.whl"])
```

… survey!

https://bit.ly/libK-install

Test: build model

```R
X <- as.matrix(c(0.0, 0.25, 0.5, 0.8, 1.0))
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
y <- f(X)

library(rlibkriging)
k_R <- Kriging(y, X, "gauss")
print(k_R)
```

```Octave
X = [0.0; 0.25; 0.5; 0.8; 1.0];
f = @(x) 1-1/2.*(sin(12*x)./(1+x)+2*cos(7.*x).*x.^5+0.7)
y = f(X);

k_m = Kriging(y, X, "gauss");
disp(k_m.summary());
```

```Python
import numpy as np
X = [0.0, 0.25, 0.5, 0.8, 1.0]
f = lambda x: (1 - 1 / 2 * (np.sin(12 * x) / (1 + x) + 2 * np.cos(7 * x) * x ** 5 + 0.7))
y = [f(xi) for xi in X]

import pylibkriging as lk
k_py = lk.Kriging(y, X, "gauss")
print(k_py.summary())
```

Test: build model

```
## Kriging model:
##
## * data: 5 x 1 -> 5 x 1
## * trend constant (est.): 0.461162
## * variance (est.): 0.102312
## * covariance:
##   * kernel: gauss
##   * range (est.) 0.272046
##   * fit:
##     * objective: LL
##     * optim: BFGS
```

Test: model predict

```r
#...
x <- as.matrix(seq(0, 1, , 100))
p <- predict(k_R, x, TRUE, FALSE)

plot(f); points(X, y)
lines(x, p$mean, col = 'blue')
polygon(c(x, rev(x)), c(p$mean - 2 * p$stdev, rev(p$mean + 2 * p$stdev)),
        border = NA, col = rgb(0, 0, 1, 0.2))
```

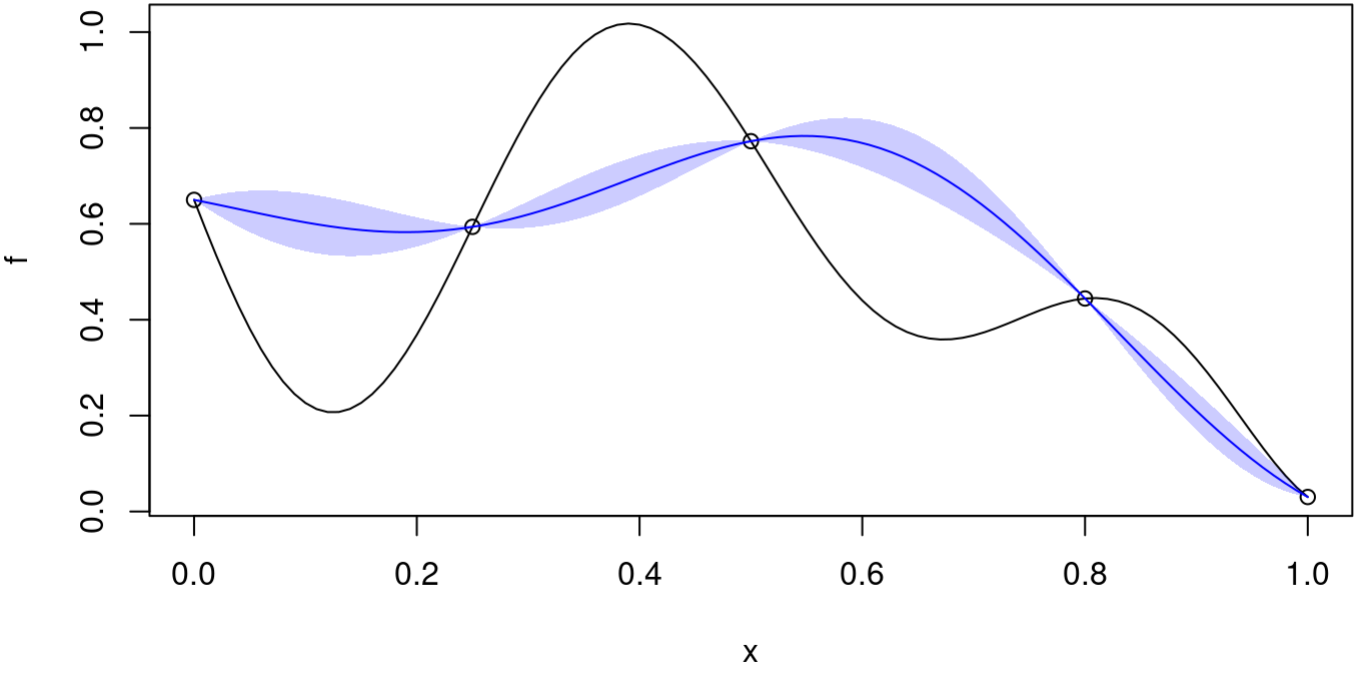
```m
%...
x = reshape(0:(1/99):1,100,1);
[p_mean, p_stdev] = k_m.predict(x, true, false);

%h = figure(1); hold on;
plot(x,f(x)); scatter(X,f(X));
plot(x,p_mean,'b')
poly = fill([x; flip(x)], [(p_mean-2*p_stdev); flip(p_mean+2*p_stdev)],
            'b');
set( poly, 'facealpha', 0.2); %hold off;
```

```python
#...
x = np.arange(0, 1, 1 / 99)
p = k_py.predict(x, True, False)
p = {"mean": p[0], "stdev": p[1], "cov": p[2]}

#import matplotlib.pyplot as pyplot; pyplot.figure(1)
pyplot.plot(x, [f(xi) for xi in x]); pyplot.scatter(X, [f(xi) for xi in X])
pyplot.plot(x, p['mean'], color='blue')
pyplot.fill(np.concatenate((x, np.flip(x))),
            np.concatenate((p['mean'] - 2 * p['stdev'], np.flip(p['mean'] + 2 * p['stdev']))),
```

Test: model predict



Test: model simulate

```r
#...
s <- simulate(k_R,nsim = 10, seed = 123, x=x)

plot(f); points(X,y)
matplot(x,s,col=rgb(0,0,1,0.2),type='l',lty=1,add=T)
```

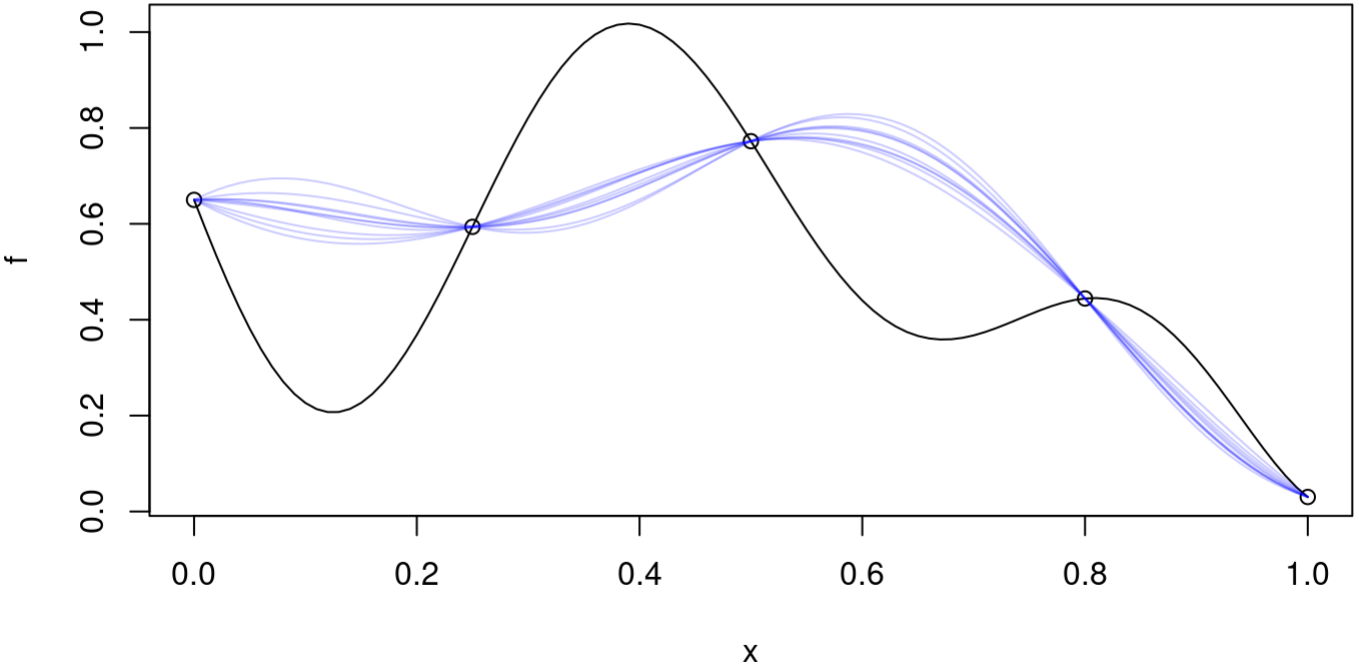
```
%...
s = k_m.simulate(int32(10),int32(123), x);

%h = figure(2); hold on;
plot(x,f(x)); scatter(X,f(X));
for i=1:10
   plot(x,s(:,i),'b');
end
%hold off;
```

```python
#...
s = k_py.simulate(10, 123, x)

#pyplot.figure(2)
pyplot.plot(x, [f(xi) for xi in x]); pyplot.scatter(X, [f(xi) for xi in X])
for i in range(10):
    pyplot.plot(x, s[:, i], color='blue', alpha=0.2)
#pyplot.show()
```

Test: model simulate



Test: fit objective: LL

```
#...
ll = function(t) logLikelihood(k_R,t)$logLikelihood

plot( ll )
```

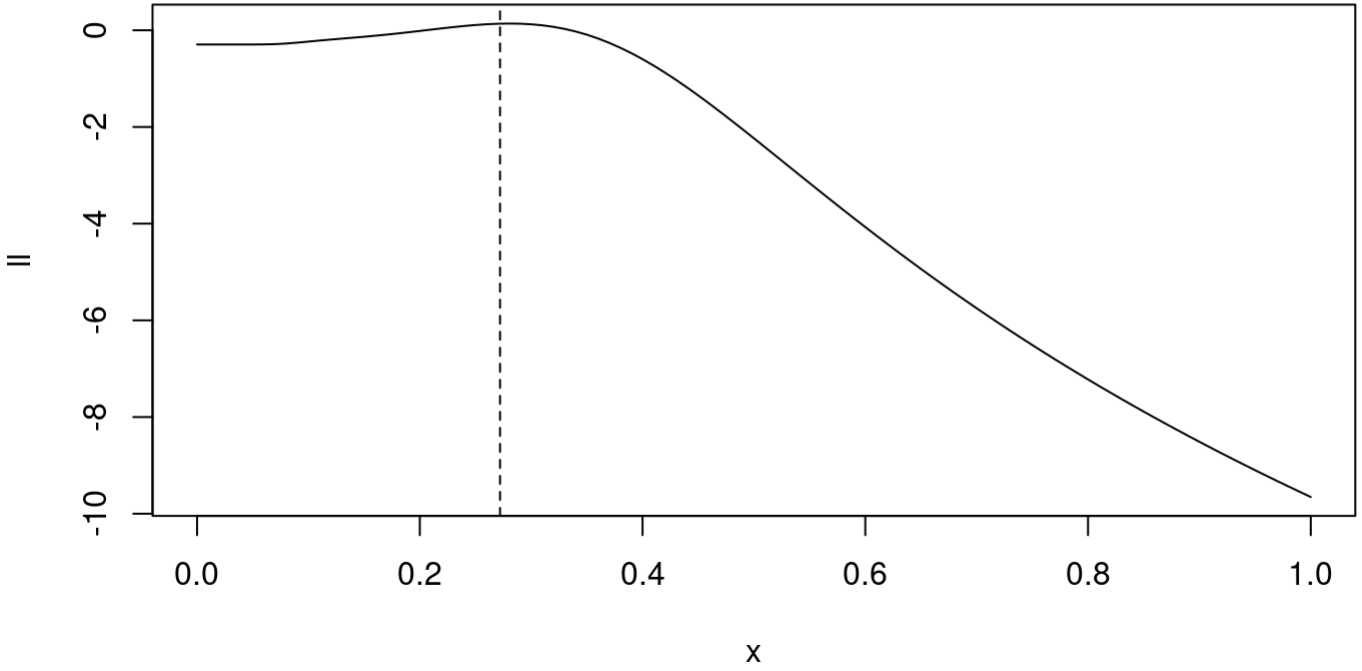
```
%...
function llt = ll (tt)
  global k_m;
  llt = k_m.logLikelihood(tt)
endfunction

t=0:(1/99):1
plot(t,arrayfun(@ll,t))
```

```
#...
def ll(t): return k_py.logLikelihood(t,False,False)[0]

t = np.arange(0,1,1/99)
pyplot.plot(t, [ ll(ti) for ti in t])
```

Test: fit objective: LL



Test: fit objective: LOO

```r
#...
k_R <- Kriging(y, X, "gauss", objective="LOO")
loo = function(t) leaveOneOut(k_R,t)$leaveOneOut

plot( loo )
```

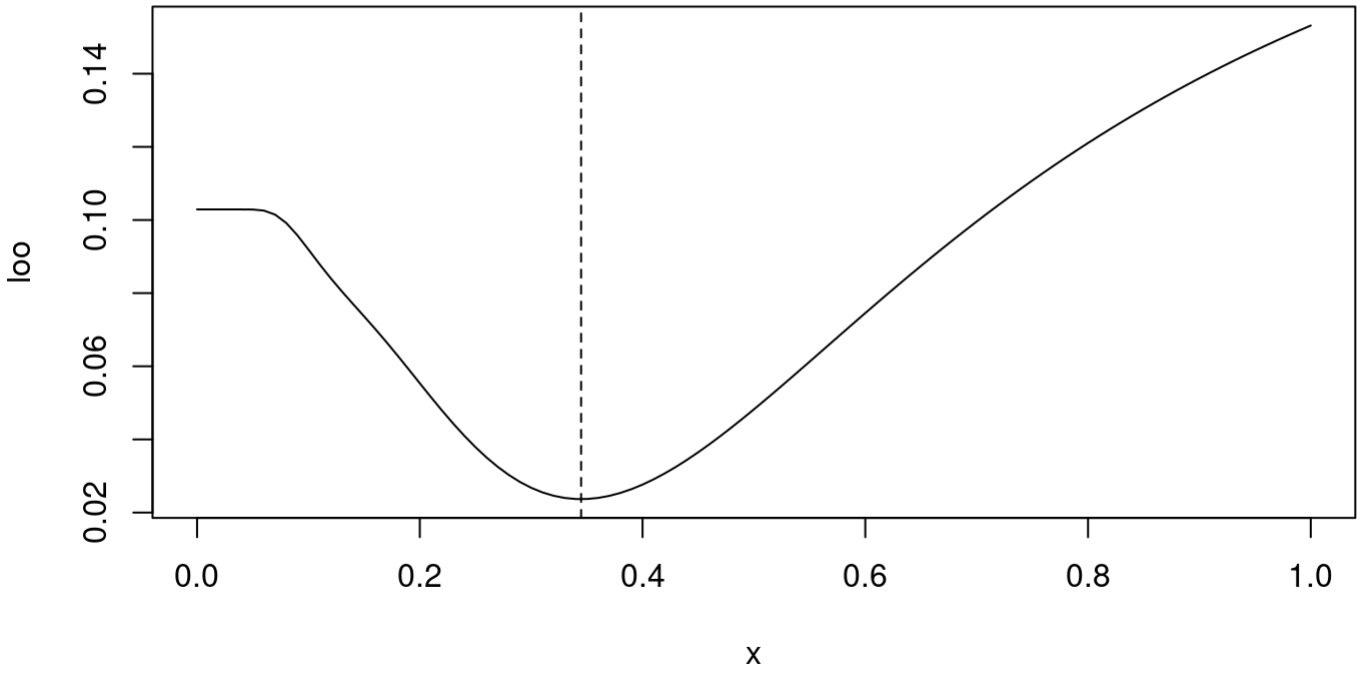
```
%...
k_m = Kriging(y, X, "gauss", objective="LOO")
function loot = loot (tt)
  global k_m;
  loot = k_m.leaveOneOut(tt)
endfunction

t=0:(1/99):1
plot(t,arrayfun(@loot,t))
```

```python
#...
k_py = lk.Kriging(y, X, "gauss", objective="LOO")
def loo(t): return k_py.leaveOneOut(t,False,False)[0]

t = np.arange(0,1,1/99)
pyplot.plot(t, [ loo(ti) for ti in t])
```

Test: fit objective: LOO



Test: fit objective: LMP

```r
#...
k_R <- Kriging(y, X, "gauss", objective="LMP")
lmp = function(t) logMargPost(k_R,t)$logMargPost

plot( loo )
```

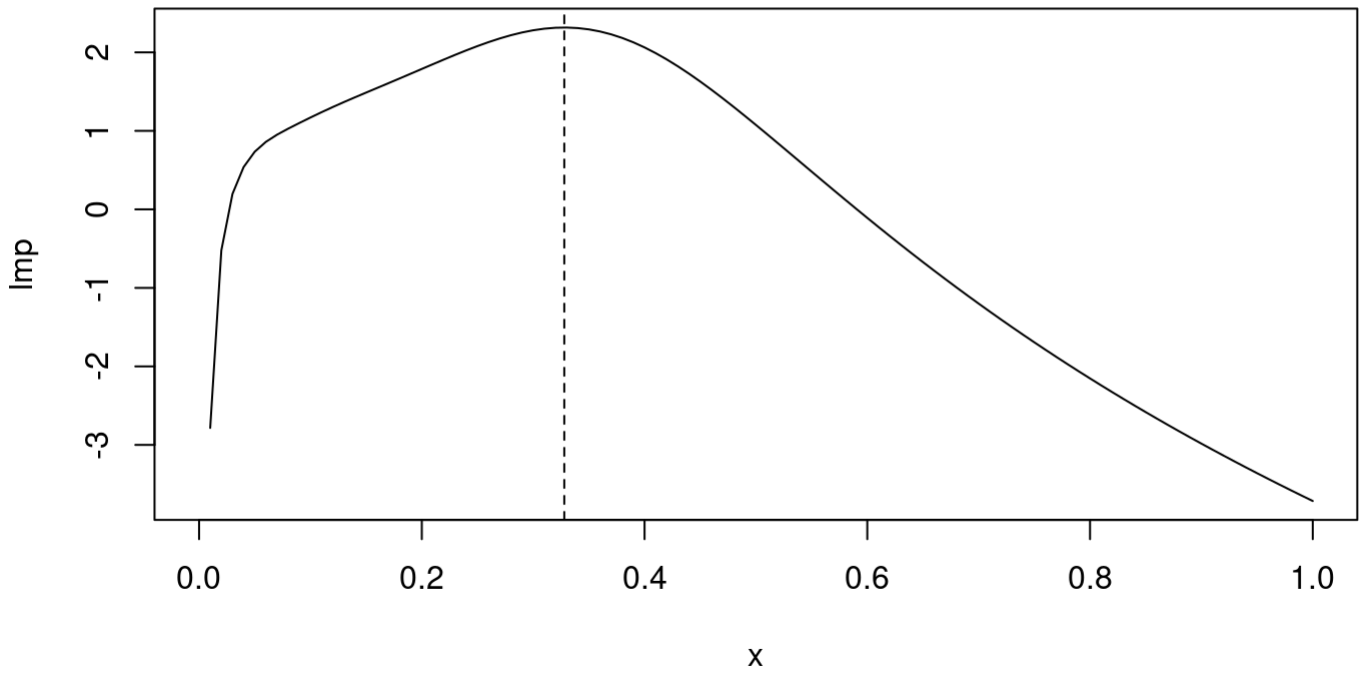
```
%...
k_py = lk.Kriging(y, X, "gauss", objective="LMP")
function lmpt = lmpt (tt)
  global k_m;
  lmpt = k_m.logMargPost(tt)
endfunction

t=0:(1/99):1
plot(t,arrayfun(@lmpt,t))
```

```python
#...
k_py = lk.Kriging(y, X, "gauss", objective="LMP")
def lmp(t): return k_py.logMargPost(t,False,False)[0]

t = np.arange(0,1,1/99)
pyplot.plot(t, [ lmp(ti) for ti in t])
```

Test: fit objective: LMP



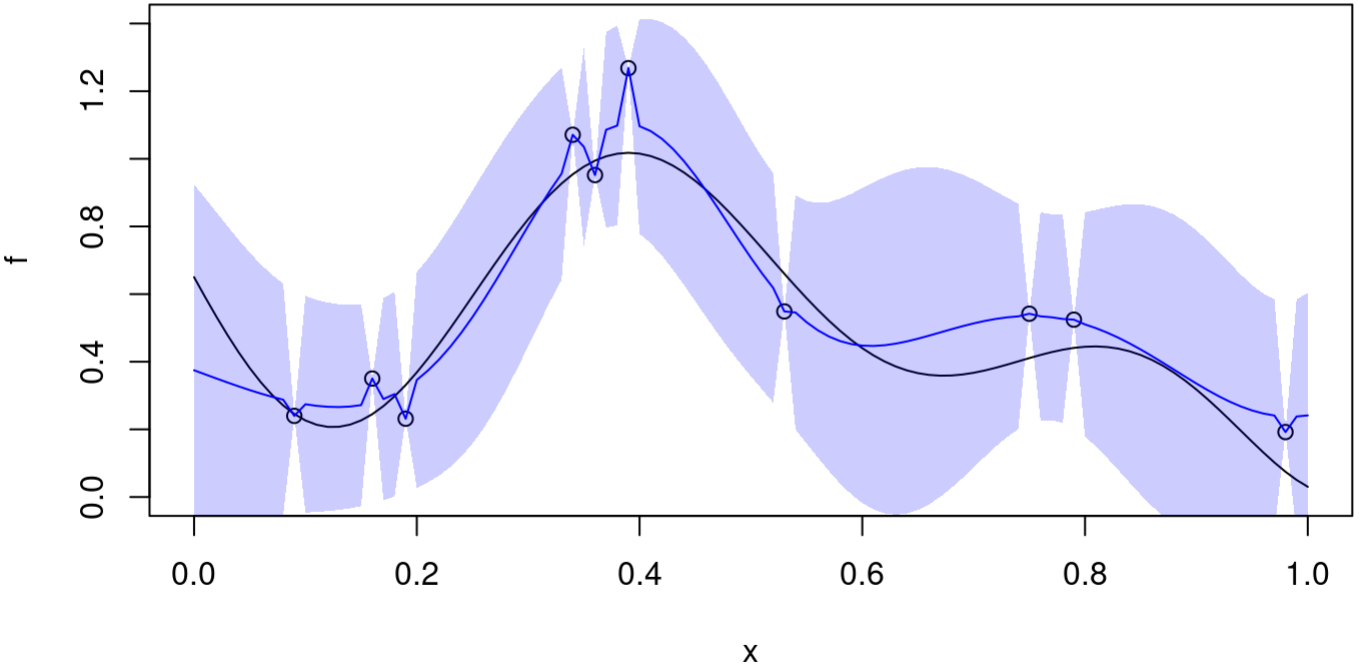
Kriging() -> NuggetKriging()

```
set.seed(123456)
X <- as.matrix(floor(100*runif(10))/100) #as.matrix(c(0.0, 0.25, 0.5, 0.8, 1.0))
f <- function(x) 1 - 1 / 2 * (sin(12 * x) / (1 + x) + 2 * cos(7 * x) * x^5 + 0.7)
y <- f(X) + 0.1 * rnorm(nrow(X))

library(rlibkriging)
k_R <- NuggetKriging(y, X, "gauss")
print(k_R)
```

```
## NuggetKriging model:
##
## * data: 10 x 1 -> 10 x 1
## * trend constant (est.): 0.497633
## * variance (est.): 0.089971
## * covariance:
##   * kernel: gauss
##   * range (est.): 0.0964639
##   * nugget (est.): 0.0155414
##   * fit:
##     * objective: LL
##     * optim: BFGS
```

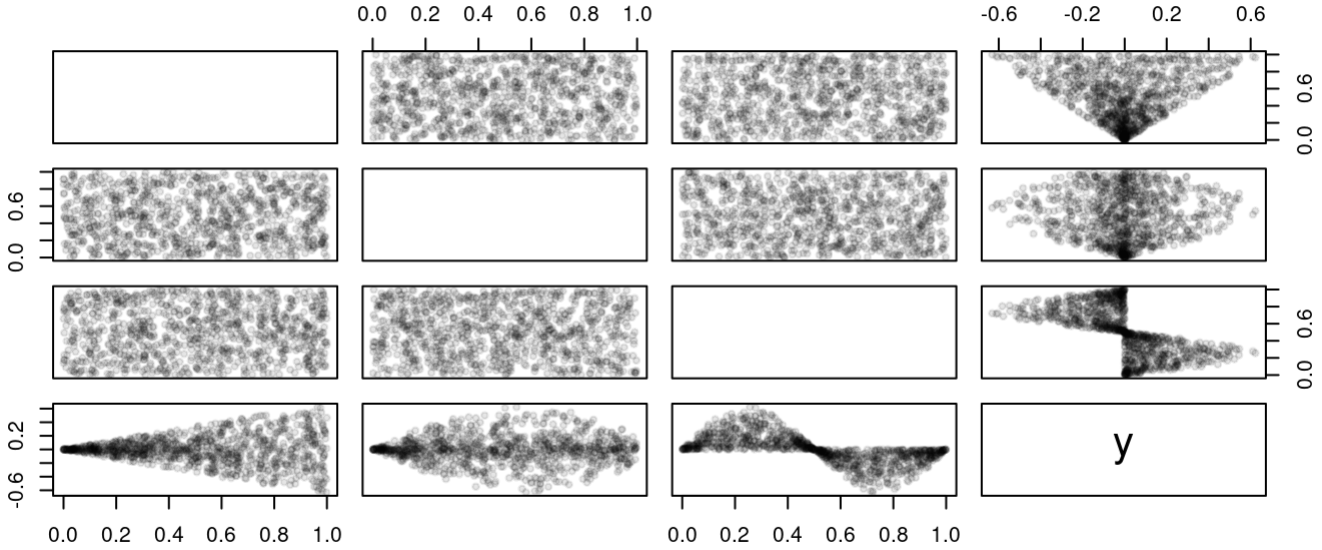
Kriging() -> NuggetKriging()



Cholesky…

```
f <- function(X) apply(X, 1, function(x)
                        prod( sin(2*pi*( x * (seq(0,1,l=1+length(x))[-1])^2 ))))
n <- 1000; d <- 3
set.seed(1234)
X <- matrix(runif(n*d),ncol=d)
y <- f(X)

pairs(cbind(X,y), pch=20,col=rgb(0,0,0,0.1))
```



Cholesky…

DiceKriging:

```
library(DiceKriging)
k = NULL
try( k <- km(response = y, design = X, covtype = "gauss") )
```

```
## Error in chol.default(R) :
##   le mineur dominant d'ordre 500 n'est pas défini positif
```

libKriging, add 1E-10 on R matrix diagonal:

```
r = NULL
try( r <- Kriging(y, X, "gauss") )
print(r)
```

```
## Kriging model:
##
## * data: 1000 x 3 -> 1000 x 1
## * trend constant (est.): 0.00315709
## * variance (est.): 0.0386684
## * covariance:
##   * kernel: gauss
##   * range (est.) 0.597647, 0.693743, 0.290895
##   * fit:
##     * objective: LL
##     * optim: BFGS
```

… survey!

https://bit.ly/libK-examples