Using the R package **kergp**: Ordinal kernels

Olivier Roustant and Yves Deville

17th March, 2021

This report[1] illustrates the use of ordinal kernels with **kergp**. For that purpose, we choose an analytical function, coming from the beam bending problem, presented in [Roustant et al., 2020]. That function will be viewed as a black-box function, that we aim at approximating with a Gaussian process model. For completeness, we first briefly recall the context.

The beam bending problem

A force is applied to an horizontal beam, fixed in a wall, which creates a vertical deviation, called deflection. Under standard assumptions, the beam deflection is simply expressed as:

$$y(L, S, \tilde{I}) = \frac{PL^3}{3ES^2\tilde{I}} \tag{1}$$

where:

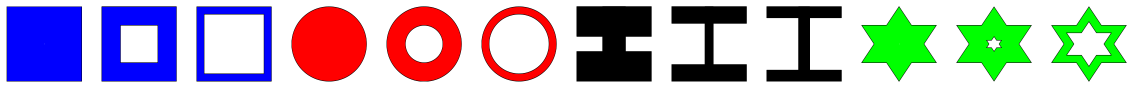
- L is the length of the beam,
- E is the Young modulus characterizing the properties of the beam,
- P is the amplitude of the vertical loading,
- S is the area of the cross-section,
- $\tilde{I} = I/S^2$ is the moment of inertia I normalized by the cross section.

Here, E, P are supposed to be constant: $E = 600\text{GPa}$, $P = 600\text{N}$. On the contrary, L, S are viewed as continuous variables varying in the intervals $[10, 20]$ and $[1, 2]$ respectively. We consider 12 kinds of beams, characterized by 4 different shapes (square, circle, I and star) and 3 filling levels (solid, medium, hollow), as shown in Figure 1. For each beam, the normalized inertia is a fixed values, which can be computed from physics equations. They are given by:

$$\begin{aligned}(\tilde{I}_1, \ldots, \tilde{I}_{12}) = (&0.0833, 0.139, 0.380, 0.0796, 0.133, 0.363,\\ &0.0859, 0.136, 0.360, 0.0922, 0.138, 0.369).\end{aligned} \tag{2}$$

Here, we assume that only the order corresponding to the inertia values is given. Hence, we consider that $\tilde{I}$ is an ordinal discrete variable with 12 levels corresponding to the beams.

[1] Compiled with **R** 4.0.4 using **kergp** 0.5.5.



(a) $\tilde{I}_1$ (b) $\tilde{I}_2$ (c) $\tilde{I}_3$ (d) $\tilde{I}_4$ (e) $\tilde{I}_5$ (f) $\tilde{I}_6$ (g) $\tilde{I}_7$ (h) $\tilde{I}_8$ (i) $\tilde{I}_9$ (j) $\tilde{I}_{10}$ (k) $\tilde{I}_{11}$ (l) $\tilde{I}_{12}$

Figure 1: Representation of the shapes considered for the cross-sections.

Construction of a GP model

Let us now start the illustration with `kergp`. We first prepare the data. We provide the 12 values of inertia. Recall that they are only used to define the order of the levels. The variable `iniLevels` contains the beam initial configurations, and `ordLevels` contains the sorted configurations (with respect to inertia values).

```
myGrid <- expand.grid(Fill = c("solid", "medium", "hollow"),
                      Shape = c("square", "circle", "I", "star"))
iniLevels <- with(myGrid, paste(Shape, Fill, sep = "."))

inertia <- c(0.08333333,  0.13888889, 0.37962963,
             0.07957747,  0.13262912, 0.36251959,
             0.08588435,  0.13554232, 0.35992989,
             0.09221567,  0.13832350, 0.36886267)

ordLevels <- iniLevels[sort(inertia, index = TRUE, decreasing = TRUE)$ix]
names(inertia) <- iniLevels
L <- length(inertia) # number of levels

plot(inertia[ordLevels], main = "Inertia", xlab = "Ordered levels")
```

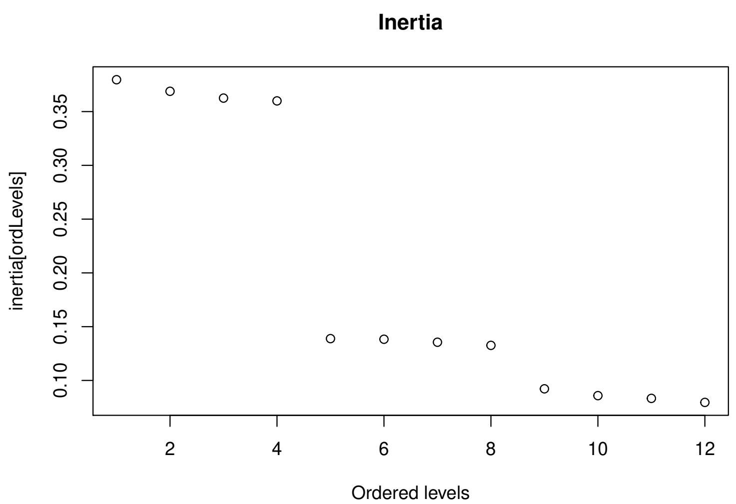


Figure 2: Inertia values, in function of the ordered levels.

The deflection is coded as a function of `x`, whose column `ShapeFill` contains an ordered factor. Mind the conversion to character when computing the inertia corresponding to the level `x[["ShapeFill"]]` (a conversion to integer would give a wrong result here). We have rescale the output by a factor of $1e5$, in order to avoid too small numbers.

```r
deflection <- function(x) {
  I <- inertia[as.character(x[["ShapeFill"]])]
  # caution: as.character must be used
  1e5 * x[["Length"]]^3 / (3e9 * x[["Section"]]^2 * I)
}
```

We now create a design of experiments (learning set), from a maximin Latin hypercube design in three dimensions, corresponding to the three input variables. The levels of `ShapeFill` are obtained by converting the continuous values to an ordered factor. We proceed to a first check, by drawing the output value `Y` versus `ShapeFill`, which should be roughly increasing.

```r
library(DiceDesign)
seed <- 0
m <- 3  # number of design points per level
X0 <- lhsDesign(n = m * L, dimension = 3, seed = seed)$design
set.seed(seed)
X <- maximinSA_LHS(X0)$design

df <- data.frame(Length = (X[ , 1] + 1) * 10,
                 Section = X[ , 2] + 1,
                 ShapeFill = as.integer(floor(X[ , 3] * L) + 1))

df <- within(df, ShapeFill <- ordered(ordLevels[ShapeFill],
                                      levels = ordLevels))
df <- data.frame(df, Y = deflection(df))
plot(Y ~ ShapeFill, data = df)
```

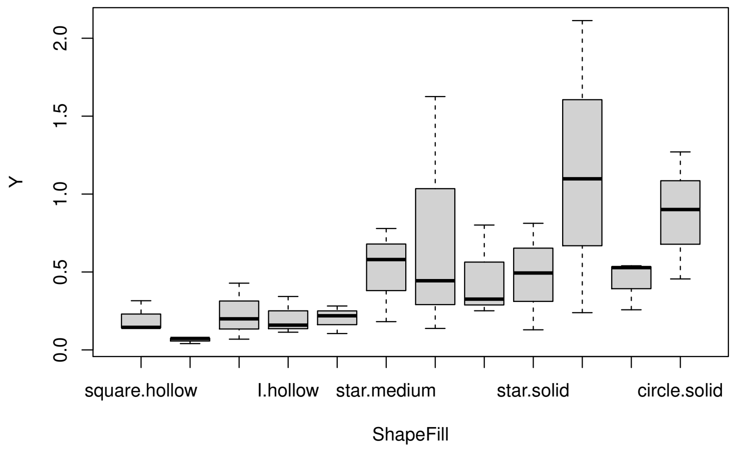


Figure 3: Y versus ShapeFill, showing a roughly monotonic pattern.

Similarly, a test set is constructed.

```r
nNew <- 1000
XNew <- lhsDesign(nNew, 2, seed = seed)$design
dfNew <- data.frame(Length = (XNew[ , 1] + 1) * 10,
                    Section = XNew[ , 2] + 1,
                    ShapeFill = sample(L, nNew, replace = TRUE))
dfNew <- within(dfNew, ShapeFill <- ordered(ordLevels[ShapeFill],
                                            levels = ordLevels))
dfNew <- data.frame(dfNew, Y = deflection(dfNew))
```

Then we consider a Gaussian process for modelling Y. The kernel is built as a product of a Matérn 5/2 kernel for each of the continuous variables `Length` and `Section`, and an ordinal kernel for the ordinal variable `ShapeFill`. The latter is built with a warping function F as:

$$k(x, x') = k_0(F(x), F(x')).$$

Hereafter, we choose the default value for k_0, i.e. a Matérn 5/2 kernel, and a spline of degree 2 for F. The help page of `covOrd` describes the available modelling options, both for setting F (e.g. Normal, spline) or k_0 (e.g. Matérn, cos).

```
library(kergp)
kCont1 <- covRadial(k1Fun1 = k1Fun1Matern5_2,
                    d = 1, cov = "homo", input = "Length")
coefUpper(kCont1) <- c(10, 10)

kCont2 <- covRadial(k1Fun1 = k1Fun1Matern5_2,
                    d = 1, cov = "corr", input = "Section")
coefUpper(kCont2) <- 10

u <- ordered(iniLevels, labels = ordLevels)
kOrd <- covOrd(ordered = u, warpFun = "spline2", inputs = "ShapeFill")

kMix <- covComp(formula = ~ kCont1() * kCont2() * kOrd() )
```

We now fit the model with `kergp`. A minimal number of multistart is recommended.

```
library(doFuture)
registerDoFuture()
plan(multisession, workers = max(parallel::detectCores() - 1, 1))
multistart <- 50

fit <-  gp(formula = Y ~ 1 ,
           cov = kMix, data = df,
           multistart = multistart)
```

We can plot the estimated warping. It shows here 2 jumps, delimiting 3 groups of levels, corresponding to the filling degree of the shape (solid, medium, hollow).

```
plot(as.list(fit$covariance)$kOrd, type = "warp")
```

Finally, we compute the predictions and assess the prediction accuracy with Q^2 criterion (R^2 on a test set).

```
pred <- predict(fit, newdata = dfNew)
Q2 <- function(obs, pred){
    1 - sum((pred - obs)^2) / sum((mean(obs) - obs)^2)
}
print(Q2(obs = dfNew$Y, pred = pred$mean))

## [1] 0.974
```

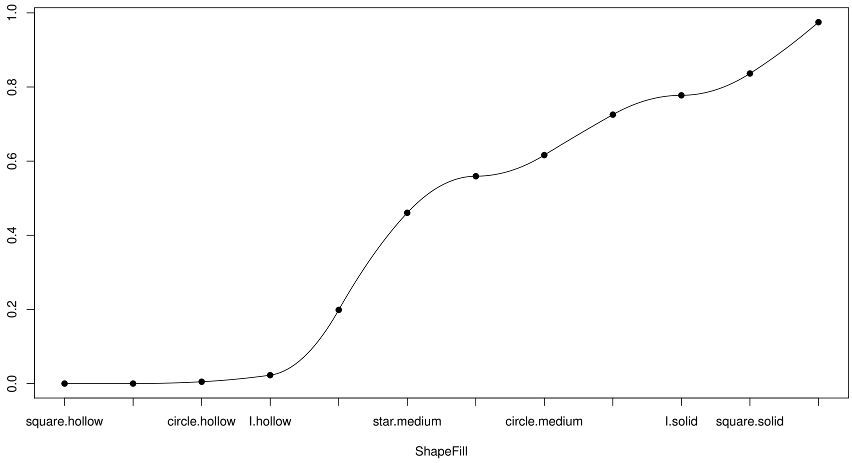


Figure 4: Estimated warping for ShapeFill.

References

[Roustant et al., 2020] Roustant, O., Padonou, E., Deville, Y., Clément, A., Perrin, G., Giorla, J., and Wynn, H. (2020). Group kernels for Gaussian process metamodels with categorical inputs. *SIAM/ASA Journal on Uncertainty Quantification*, 8(2):775–806.