Examples of Cokriging with the Function `cok`

Hao Zhang

April 2014

In this note, I will present two examples of using the function `cok` for ordinary cokriging.

Example 1. Suppose we have two processes observed at two sets of locations, `loc1` and `loc2`, and predict the first process at some new locations, `newloc`. Figure 1 shows all the locations.

```
loc1 = cbind(rep(1:10, 10), rep(1:10, each = 10))/10
loc2 = cbind(rep(2 * (1:4) + 0.2, 4), rep(2 * (1:4) +
    0.2, each = 4))/10
plot(loc1)
points(loc2, pch = 19)
newloc = cbind(rep(1:15, 15), rep(1:15, each = 15))/15
plot(loc1)
points(loc2, pch = 19)
newloc = cbind(rep(1:15, 15), rep(1:15, each = 15))/15
points(newloc, pch = 20)
```

Suppose we will use the following bivariate covariogram for the cokriging,

$$C(h) = \begin{pmatrix} 2 & 0.98 \\ 0.98 & 1 \end{pmatrix} \exp(-h/0.2) + \begin{pmatrix} 1 & 1.04 \\ 1.04 & 3 \end{pmatrix} \exp(-h/0.3).$$

We define the following covariogram in R

```
cov.lcm = function(distance, para) {
    para[1] * exp(-distance/para[2]) + para[3] *
        exp(-distance/para[4])
}
```

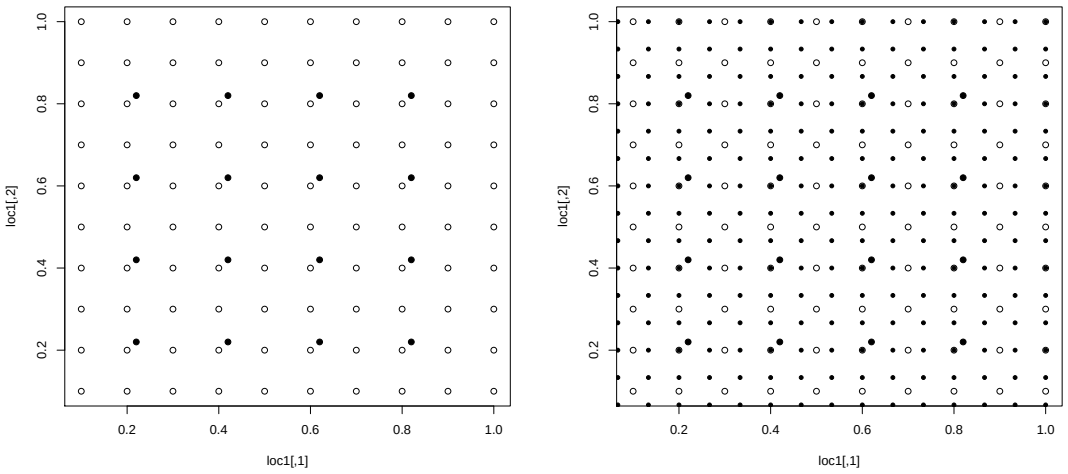


Figure 1: Locations

The function `cok` is applied as follows.

```
y = rnorm(nrow(loc1) + nrow(loc2))
cok(data = list(y[1:100], y[101:116]), sampleLoc = list(loc1,
    loc2), predLoc = newloc[1:10, ], cov.fun = list("cov.lcm",
    "cov.lcm", "cov.lcm"), cov.pars = list(c(2, 0.2,
    1, 0.3), c(1, 0.2, 3, 0.3), c(0.98, 0.2, 1.04,
    0.3)))

##  [1] -0.52962  0.20465  1.37431  0.86329
##  [5] -0.01036 -0.66280 -0.41161  0.01664
##  [9]  0.18117 -0.45962
```

Note for illustration only, I generated i.i.d observations for y. Ideally, we would want to generate from the underlying model. Whatever the observation vector y is, you apply the function `cok` in the same way.

Example 2. The primary spatial variable is the soil temperature and the auxiliary variable is the air temperature in the data set `paws`. Sample data revealed that the two variables are correlated, as shown in Figure 2.

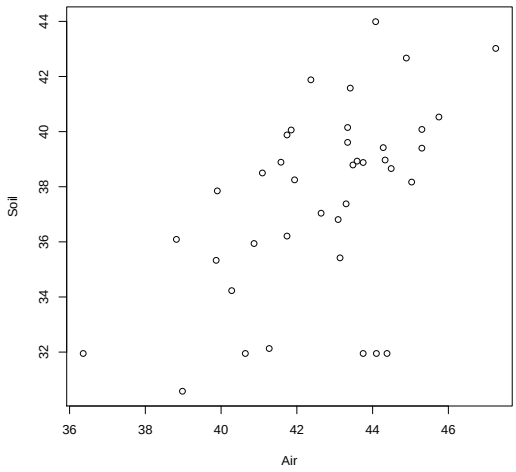


Figure 2: Locations in PAWS data

Let us apply the following model

$$Y_1(\boldsymbol{s}) = a + bY_2(\boldsymbol{s}) + e(\boldsymbol{s}), \tag{1}$$

where $Y_1(\boldsymbol{s})$ and $Y_2(\boldsymbol{s})$ represent the soil and temperature at location $\boldsymbol{s}$, respectively, $e(\boldsymbol{s})$ is a stationary process and independent of the process $Y_2(\boldsymbol{s})$. The predictor for $Y_1(\boldsymbol{s}_0)$ is

$$\hat{Y}_1(\boldsymbol{s}_0) = a + bY_2(\boldsymbol{s}_0) + \hat{e}(\boldsymbol{s}_0), \tag{2}$$

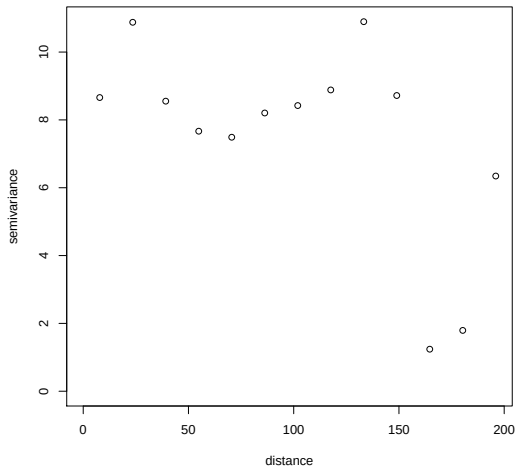
where $\hat{e}(\boldsymbol{s}_0)$ is the ordinary kriging predictor for $e(\boldsymbol{s}_0)$ based on the residuals $e(\boldsymbol{s}_i) = Y_1(\boldsymbol{s}_i) - a - bY_2(\boldsymbol{s}_i)$.

1. Let $C_{11}(h)$ and $C_{22}(h)$ denote the covariogram of $Y_1(\boldsymbol{s})$ and $Y_2(\boldsymbol{s})$, respectively, and $C(\boldsymbol{s})$ denote the covariogram of the error $e(\boldsymbol{s})$. Also let $C_{12}(h)$ denote the cross-covariogram of $Y_1(\boldsymbol{s})$ and $Y_2(\boldsymbol{s})$. Show that

$$C_{11}(h) = b^2 C_{22}(h) + C(h), \quad C_{12}(h) = bC_{22}(h). \tag{3}$$

2. Use $a = 1.3389, b = 0.8458$ to calculate the residuals $e(\boldsymbol{s}_i) = Y_1(\boldsymbol{s}_i) - a - bY_2(\boldsymbol{s}_i)$. Fit an exponential covariogram model to the residuals by the maximum likelihood method.

```r
library(geoR)
attach(paws)

## The following object is masked _by_ .GlobalEnv:
##
##     X

a = 1.3389
b = 0.8458
residuals = Soil8 - a - b * Air
plot(variog(coords = paws[, c("X", "Y")], data = residuals))

## variog: computing omnidirectional variogram

residual.fit = likfit(coords = paws[, c("X", "Y")],
    data = residuals, ini.cov.pars = c(2, 60), nugget = 8,
    method = "ML")

## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.

estim.resid = unlist(residual.fit[c(2, 4, 5)])
```



3. Fit an exponential covariogram model to the air temperature $Y_2(\boldsymbol{s})$ through the maximum likelihood method. You then have an estimated $C_{22}(h)$.

```
air.fit = likfit(coords = paws[, c("X", "Y")], data = paws$Air,
    ini.cov.pars = c(6, 60), nugget = 1, method = "ML")


## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.

(estim.air = unlist(air.likfit[c(2, 4, 5)]))


##  nugget sigmasq     phi
##  0.7216  6.8878 85.2964
```

4. We now use model (2) to find the cokriging prediction for the soil temperature at the last or the 39th location using the air temperature of all 39 locations and the soil temperature at the first 38 locations.

```
cov.exp = function(x, para) {
    para[1] * (x == 0) + para[2] * exp(-x/para[3])
}
cov.soil = function(x, para) {
    beta = 0.8458
    cov.exp(x, para) + beta^2 * cov.exp(x, estim.air)
}
cok(data = list(paws[-39, ]$Soil8, paws$Air), sampleLoc = list(paws[-39,
    c("X", "Y")], paws[, c("X", "Y")]), predLoc = c(paws[39,
    ]$X, paws[39, ]$Y), cov.fun = list("cov.soil",
    "cov.exp", "cov.exp"), cov.pars = list(estim.resid,
    estim.air, c(b * estim.air[1:2], estim.air[3])))

## [1] 38.93

# answer=38.93365
```

5. As shown in class, the cokriging prediction should equal the the following prediction

$$\hat{y}_{39} = 1.3389 + 0.8458 Y_{air,39} + \hat{e}_{39},$$

where $\hat{e}_{39}$ is the ordinary kriging for $e(\boldsymbol{s}_{39})$ based on the residuals at the first 38 locations. You should find that the two predictions are equal.

```
e.pred = krige.conv(coords = paws[-39, c("X", "Y")],
    data = residuals[-39], locations = paws[39, c("X",
        "Y")], krige = krige.control(cov.model = "exponential",
        cov.pars = c(sigmasq = estim.resid[2], phi = estim.resid[3]),
        nugget = estim.resid[1]))$predict

## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

```
a + b * Air[39] + e.pred

##  data
## 38.93
```