Hierarchical Modeling for Multivariate Spatial Data in R

Andrew O. Finley and Sudipto Banerjee

June 9, 2013

1 Data preparation and initial exploration

We make use of several libraries in the following example session, including:

- library(spBayes)
- library(fields)
- library(geoR)
- library(MBA)
- library(sp)

We motivate this session with soil nutrient data which was collected at the La Selva Biological Station, Costa Rica\(^1\). Here, \(n = 80\) soil cores were sampled over a sparse grid centered on a more intensively sampled transect. Soil nutrient concentrations of calcium (Ca), potassium (K) and magnesium (Mg) were measured for each sample. These nutrient concentrations show a high positive correlation (1) suggesting that we might build a richer model by explicitly accounting for spatial association among the \(q = 3\) response variables. Our objective is to predict these nutrients at a fine resolution over the study plot. Ultimately, posterior predictive samples will serve as input to a vegetation competition model. We begin by log transforming the response variables and taking a look at sample location across the study plot.

\[
\begin{pmatrix}
1 \\
0.7 & 1 \\
0.7 & 0.8 & 1
\end{pmatrix}
\]  

(1)

\(^1\)Data provided by Richard Kobe, Ellen Holste, and Tom Baribault with support from NSF DEB 0640904 & 0743609
We can gain a non-statistical estimate of the nutrient concentration surfaces using the MBA package `mba.surf` function, Figure 1. These patterns can be more formally examined using empirical semivariograms. In the code block below, we fit an exponential variogram model to each of the soil nutrients. The resulting variogram estimates are offered in Figure 2. Here the upper and lower horizontal lines are the sill and nugget, respectively, and the vertical line is the effective range (i.e., that distance at which the correlation drops to 0.05). Despite the patterns of spatial dependence seen in Figure 1, the variograms do not show much of a spatial process. Changing the number of bins (`bins`) and maximum distance considered (`max`) will produce effective spatial ranges of less than 20 m for each of the nutrients; however, the signal is weak, likely due to the paucity of samples.

```r
> max <- 0.25 * max(as.matrix(dist(dat[, c("X", "Y")])))
> bins <- c(9, 8, 9)
```
We continue with fitting a multivariate regression that allows for spatial (\(K\)) and non-spatial (\(\Psi\)) cross-covariance matrices. We would expect the sum of these matrices to be equal to the aspatial covariance matrix of the observed data (2).

\[
\begin{pmatrix}
0.5 \\
0.2 & 0.2 \\
0.2 & 0.2 & 0.3
\end{pmatrix}
\]  

(2)

In the following code block we define the model parameters' starting, tuning, and prior distribution, then call \texttt{spMvLM}. Trace plots in Figure 3.
Figure 2: Isotropic semivariograms for log nutrient concentrations.
Psi = Psi.starting)
> tuning <- list(phi = rep(0.1, q), A = A.tuning, Psi = Psi.tuning)
> priors <- list(phi.Unif = list(rep(3/60, q), rep(3/10,
+ q)), K.IW = list(q + 1, diag(0.001, q)), Psi.IG = list(rep(2,
+ q), c(0.05, 0.08, 0.05)))
> m.1 <- spMvLM(list(Ca ~ 1, K ~ 1, Mg ~ 1), coords = coords,
+ data = log.nut, starting = starting, tuning = tuning,
+ priors = priors, cov.model = "exponential", n.samples = n.samples,
+ n.report = 2000)

----------------------------------------
General model description
----------------------------------------
Model fit with 80 observations.

Number of covariates 3 (including intercept if specified).

Using the exponential spatial correlation model.

Number of MCMC samples 10000.

Priors and hyperpriors:
  beta flat.

    K IW hyperpriors df=4.00000, S=
    0.001  0.000  0.000
    0.000  0.001  0.000
    0.000  0.000  0.001

Diag(Psi) IG hyperpriors
parameter shape scale
Psi[1,1]  2.0  0.05
Psi[2,2]  2.0  0.08
Psi[3,3]  2.0  0.05

phi Unif hyperpriors
parameter a    b
phi[1]  0.05000 0.30000
phi[2]  0.05000 0.30000
phi[3]  0.05000 0.30000

----------------------------------------
Sampling
----------------------------------------
Sampled: 2000 of 10000, 20.00%
Report interval Metrop. Acceptance rate: 57.80%
Overall Metrop. Acceptance rate: 57.80%
-------------------------------------------------
Sampled: 4000 of 10000, 40.00%
Report interval Metrop. Acceptance rate: 38.80%
Overall Metrop. Acceptance rate: 48.30%
-------------------------------------------------
Sampled: 6000 of 10000, 60.00%
Report interval Metrop. Acceptance rate: 35.70%
Overall Metrop. Acceptance rate: 44.10%
-------------------------------------------------
Sampled: 8000 of 10000, 80.00%
Report interval Metrop. Acceptance rate: 26.50%
Overall Metrop. Acceptance rate: 39.70%
-------------------------------------------------
Sampled: 10000 of 10000, 100.00%
Report interval Metrop. Acceptance rate: 23.45%
Overall Metrop. Acceptance rate: 36.45%
-------------------------------------------------

> A.starting <- diag(0.05, q)[lower.tri(diag(1, q), TRUE)]
> Psi.starting <- rep(0.1, q)
> m.2 <- spMvLM(list(Ca ~ 1, K ~ 1, Mg ~ 1), coords = coords,
+     data = log.nut, starting = starting, tuning = tuning,
+     priors = priors, cov.model = "exponential", n.samples = n.samples,
+     n.report = 2000)

----------------------------------------
General model description
----------------------------------------
Model fit with 80 observations.
Number of covariates 3 (including intercept if specified).
Using the exponential spatial correlation model.
Number of MCMC samples 10000.

Priors and hyperpriors:
    beta flat.

    K IW hyperpriors df=4.000000, S=
    0.001     0.000
    0.000     0.001
    0.000     0.000

    Diag(Psi) IG hyperpriors
### Parameter Shape Scale

<table>
<thead>
<tr>
<th>parameter</th>
<th>shape</th>
<th>scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Psi}[1,1]$</td>
<td>2.0</td>
<td>0.05</td>
</tr>
<tr>
<td>$\text{Psi}[2,2]$</td>
<td>2.0</td>
<td>0.08</td>
</tr>
<tr>
<td>$\text{Psi}[3,3]$</td>
<td>2.0</td>
<td>0.05</td>
</tr>
</tbody>
</table>

### Phi Unif Hyperpriors

<table>
<thead>
<tr>
<th>parameter</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi[1]$</td>
<td>0.05000</td>
<td>0.30000</td>
</tr>
<tr>
<td>$\phi[2]$</td>
<td>0.05000</td>
<td>0.30000</td>
</tr>
<tr>
<td>$\phi[3]$</td>
<td>0.05000</td>
<td>0.30000</td>
</tr>
</tbody>
</table>

---

### Sampling

Sampled: 2000 of 10000, 20.00%
Report interval Metrop. Acceptance rate: 57.75%
Overall Metrop. Acceptance rate: 57.75%

Sampled: 4000 of 10000, 40.00%
Report interval Metrop. Acceptance rate: 32.30%
Overall Metrop. Acceptance rate: 45.02%

Sampled: 6000 of 10000, 60.00%
Report interval Metrop. Acceptance rate: 28.15%
Overall Metrop. Acceptance rate: 39.40%

Sampled: 8000 of 10000, 80.00%
Report interval Metrop. Acceptance rate: 31.45%
Overall Metrop. Acceptance rate: 37.41%

Sampled: 10000 of 10000, 100.00%
Report interval Metrop. Acceptance rate: 42.10%
Overall Metrop. Acceptance rate: 38.35%

---

```r
> round(summary(m.1$p.theta.samples)$quantiles[, c(1, 3, 5)], 3)

<table>
<thead>
<tr>
<th></th>
<th>2.5%</th>
<th>50%</th>
<th>97.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K[1,1]$</td>
<td>0.009</td>
<td>0.365</td>
<td>0.541</td>
</tr>
<tr>
<td>$K[2,1]$</td>
<td>0.012</td>
<td>0.191</td>
<td>0.291</td>
</tr>
<tr>
<td>$K[3,1]$</td>
<td>0.027</td>
<td>0.295</td>
<td>0.421</td>
</tr>
<tr>
<td>$K[2,2]$</td>
<td>0.019</td>
<td>0.102</td>
<td>0.174</td>
</tr>
<tr>
<td>$K[3,2]$</td>
<td>0.033</td>
<td>0.155</td>
<td>0.231</td>
</tr>
<tr>
<td>$K[3,3]$</td>
<td>0.084</td>
<td>0.240</td>
<td>0.348</td>
</tr>
<tr>
<td>$\text{Psi}[1,1]$</td>
<td>0.070</td>
<td>0.096</td>
<td>0.475</td>
</tr>
</tbody>
</table>
```
Given the assumed exponential correlation function, the effective spatial range associated with the first outcome variable in the multivariate vector, i.e., Ca, is obtained by solving $\rho(d; \phi) = 0.05$ for $d$, i.e., $d = -\ln(0.05)/\phi$. However, because of the linear combination induced by the cross-covariance matrix, the subsequent effective spatial ranges are obtained by solving a system of equations (see Gelfand et al. 2004, p292). For example, the effective spatial range for K is given by solving $(a_{2,1}^2\rho(d; \phi_1) + a_{2,2}^2\rho(d; \phi_2))/(a_{2,1}^2 + a_{2,2}^2) = 0.05$ for $d$, where $a_{2,1}$ and $a_{2,2}$ are the elements of $A$ corresponding to the row and column subscripts. In a similar way, the effective spatial range for Mg is given by solving $(a_{3,1}^2\rho(d; \phi_1) + a_{3,2}^2\rho(d; \phi_2) + a_{3,3}^2\rho(d; \phi_3))/(a_{3,1}^2 + a_{3,2}^2 + a_{3,3}^2) = 0.05$ for $d$. The effective spatial ranges for additional outcomes follow the same pattern.

The effective range along with the other model parameters estimates are offered in the code block below.

```r
> burn.in <- 0.75 * n.samples
> p.theta.samples <- as.matrix(window(mcmc.list(m.1$p.theta.samples, + m.2$p.theta.samples), window = burn.in))
> n.samples <- nrow(p.theta.samples)
> fn <- function(d, a, phi) {
+ 0.05 - sum(a^2 * exp(-phi * d))/sum(a^2)
+ }
> get.A <- function(K, q) {
+ A <- matrix(0, q, q)
+ # Add code for the effective spatial ranges for additional outcomes
+ }
```
+ A[lower.tri(A, TRUE)] <- K
+ A[upper.tri(A, FALSE)] <- t(A)[upper.tri(A, FALSE)]
+ t(chol(A))
+
> eff.range <- matrix(0, 3, n.samples)
> for (s in 1:n.samples) {
+ A <- get.A(p.theta.samples[s, c("K[1,1]", "K[2,1]",
+ "K[3,1]", "K[2,2]", "K[3,2]", "K[3,3]")], q)
+ phi <- p.theta.samples[s, c("phi[1]", "phi[2]",
+ "phi[3]")]
+ for (r in 1:q) {
+ eff.range[r, s] <- uniroot(fn, lower = 0, upper = 1000,
+ tol = 1e-15, a = A[i, 1:r], phi = phi[1:r])$root
+ }
+ }
> rownames(eff.range) <- paste("Eff. range ", 1:q, sep = ",")
> round(summary(mcmc(cbind(p.theta.samples, t(eff.range))))$quantiles[, c(1, 3, 5)], 3)

<table>
<thead>
<tr>
<th></th>
<th>2.5%</th>
<th>50%</th>
<th>97.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>K[1,1]</td>
<td>0.008</td>
<td>0.342</td>
<td>0.567</td>
</tr>
<tr>
<td>K[2,1]</td>
<td>0.016</td>
<td>0.180</td>
<td>0.299</td>
</tr>
<tr>
<td>K[3,1]</td>
<td>0.021</td>
<td>0.282</td>
<td>0.434</td>
</tr>
<tr>
<td>K[2,2]</td>
<td>0.040</td>
<td>0.098</td>
<td>0.168</td>
</tr>
<tr>
<td>K[3,2]</td>
<td>0.044</td>
<td>0.151</td>
<td>0.235</td>
</tr>
<tr>
<td>K[3,3]</td>
<td>0.080</td>
<td>0.235</td>
<td>0.350</td>
</tr>
<tr>
<td>Psi[1,1]</td>
<td>0.070</td>
<td>0.102</td>
<td>0.430</td>
</tr>
<tr>
<td>Psi[2,2]</td>
<td>0.033</td>
<td>0.062</td>
<td>0.208</td>
</tr>
<tr>
<td>Psi[3,3]</td>
<td>0.008</td>
<td>0.018</td>
<td>0.291</td>
</tr>
<tr>
<td>phi[1]</td>
<td>0.115</td>
<td>0.226</td>
<td>0.299</td>
</tr>
<tr>
<td>phi[2]</td>
<td>0.056</td>
<td>0.176</td>
<td>0.295</td>
</tr>
<tr>
<td>phi[3]</td>
<td>0.053</td>
<td>0.142</td>
<td>0.293</td>
</tr>
<tr>
<td>Eff. range 1</td>
<td>10.033</td>
<td>13.284</td>
<td>26.128</td>
</tr>
<tr>
<td>Eff. range 2</td>
<td>10.041</td>
<td>13.343</td>
<td>26.125</td>
</tr>
</tbody>
</table>

In the code block below, we unstack the nutrient concentration spatial random effects and compare them with the residual image plots from a non-spatial regression, Figure 4.

> Ca.resids <- resid(lm(Ca ~ 1, data = log.nut))
> K.resids <- resid(lm(K ~ 1, data = log.nut))
> Mg.resids <- resid(lm(Mg ~ 1, data = log.nut))
> m.1 <- spRecover(m.1, start = burn.in, thin = 10)

-----------------------------------------------

Recovering beta and w
Sampled: 99 of 251, 39.44%
Sampled: 199 of 251, 79.28%

```r
> w <- rowMeans(m.1$p.w.recover.samples)
> w.Ca <- w[seq(1, length(w), q)]
> w.K <- w[seq(2, length(w), q)]
> w.Mg <- w[seq(3, length(w), q)]
> res <- 100
> par(mfrow = c(3, 2))
> surf.r <- mba.surf(cbind(coords, Ca.resids), no.X = res,
+ no.Y = res, extend = FALSE)$xyz.est
> surf.w <- mba.surf(cbind(coords, w.Ca), no.X = res,
+ no.Y = res, extend = FALSE)$xyz.est
> z.lim <- range(c(surf.r[["z"]], surf.w[["z"]]), na.rm = TRUE)
> image.plot(surf.r, zlim = z.lim, main = "Ca lm residuals")
> points(coords)
> image.plot(surf.w, zlim = z.lim, main = "Ca spatial effects")
> points(coords)
> surf.r <- mba.surf(cbind(coords, K.resids), no.X = res,
+ no.Y = res, extend = FALSE)$xyz.est
> surf.w <- mba.surf(cbind(coords, w.K), no.X = res,
+ no.Y = res, extend = FALSE)$xyz.est
> z.lim <- range(c(surf.r[["z"]], surf.w[["z"]]), na.rm = TRUE)
> image.plot(surf.r, zlim = z.lim, main = "K lm residuals")
> points(coords)
> image.plot(surf.w, zlim = z.lim, main = "K spatial effects")
> points(coords)
> surf.r <- mba.surf(cbind(coords, K.resids), no.X = res,
+ no.Y = res, extend = FALSE)$xyz.est
> surf.w <- mba.surf(cbind(coords, w.Mg), no.X = res,
+ no.Y = res, extend = FALSE)$xyz.est
> z.lim <- range(c(surf.r[["z"]], surf.w[["z"]]), na.rm = TRUE)
> image.plot(surf.r, zlim = z.lim, main = "Mg lm residuals")
> points(coords)
> image.plot(surf.w, zlim = z.lim, main = "Mg spatial effects")
> points(coords)
```

2 Prediction

With a sparse sample array, an estimated mean effective range of \( \sim 20 \), and no predictor variables, we cannot expect our prediction to differ much from a constant mean concentration over the domain. In the code block below, we define our prediction grid, construct the prediction design matrix using `mkMvX`, and call `spPredict`. 

---
Figure 4: Interpolated surface of the non-spatial model residuals and the mean of the spatial random effects posterior distribution.
> x.range <- range(coords[, 1])
> y.range <- range(coords[, 2])
> pred.coords <- expand.grid(seq(x.range[1], x.range[2], by = 4), seq(y.range[1], y.range[2], by = 4))
> m <- nrow(pred.coords)
> pred.X <- mkMvX(list(matrix(1, m, 1), matrix(1, m, 1), matrix(1, m, 1)))
> nut.pred <- spPredict(m.1, start = burn.in, thin = 10, pred.coords = pred.coords, pred.covars = pred.X)

-------------------------------------------------
Recovering beta
-------------------------------------------------
Sampled: 99 of 251, 39.44%
Sampled: 199 of 251, 79.28%

General model description
Model fit with 80 observations.
Prediction at 793 locations.
Number of covariates 3 (including intercept if specified).
Using the exponential spatial correlation model.

Sampling
Sampled: 100 of 251, 39.44%
Sampled: 200 of 251, 79.28%

The nut.pred list object holds the posterior predictive samples for the spatial effects $w_{pred}$ and response $y_{pred}$. Again, like with the spatial random effect in the spMvLM object, the posterior samples are stacked by location and therefore need to be unstacked as detailed in the code block below. Here also, we convert our prediction grid into a sp SpatialGridDataFrame then subsequently to a format that can be plotted by the image or fields image.plot function.

> y.pred.mu <- apply(nut.pred$p.y.predictive.samples, + 1, mean)
> y.pred.sd <- apply(nut.pred$p.y.predictive.samples, + 1, sd)
> Ca.pred.mu <- y.pred.mu[seq(1, length(y.pred.mu), q)]
> K.pred.mu <- y.pred.mu[seq(2, length(y.pred.mu), q)]
> Mg.pred.mu <- y.pred.mu[seq(3, length(y.pred.mu), q)]
> Ca.pred.sd <- y.pred.sd[seq(1, length(y.pred.sd), q)]
> K.pred.sd <- y.pred.sd[seq(2, length(y.pred.sd), q)]
> Mg.pred.sd <- y.pred.sd[seq(3, length(y.pred.sd), q)]
K.pred.sd <- y.pred.sd[seq(2, length(y.pred.sd), q)]
Mg.pred.sd <- y.pred.sd[seq(3, length(y.pred.sd), q)]
nut.pred.grid <- as.data.frame(list(x = pred.coords[, + 1], y = pred.coords[, 2], Ca.mu = Ca.pred.mu, K.mu = K.pred.mu, + Mg.mu = Mg.pred.mu, Ca.sd = Ca.pred.sd, K.sd = K.pred.sd, + Mg.sd = Mg.pred.sd))
coordinates(nut.pred.grid) <- c("x", "y")
gridded(nut.pred.grid) <- TRUE
toImage <- function(x) {
  as.image.SpatialGridDataFrame(x)
}
res <- 100
par(mfrow = c(3, 2))
surf <- mba.surf(cbind(coords, log.nut[, "Ca"]), no.X = res, + no.Y = res, extend = FALSE)$xyz.est
z.lim <- range(surf[["z"]], na.rm = TRUE)
image.plot(surf, xaxs = "r", yaxs = "r", main = "Interpolation of observed Ca")
points(coords)
image.plot(toImage(nut.pred.grid["Ca.mu"]), xaxs = "r", + yaxs = "r", zlim = z.lim, main = "Mean of Ca prediction")
points(coords)
surf <- mba.surf(cbind(coords, log.nut[, "K"]), no.X = res, + no.Y = res, extend = FALSE)$xyz.est
z.lim <- range(surf[["z"]], na.rm = TRUE)
image.plot(surf, xaxs = "r", yaxs = "r", main = "Interpolation of observed K")
points(coords)
image.plot(toImage(nut.pred.grid["K.mu"]), xaxs = "r", + yaxs = "r", zlim = z.lim, main = "Mean of K prediction")
points(coords)
surf <- mba.surf(cbind(coords, log.nut[, "Mg"]), no.X = res, + no.Y = res, extend = FALSE)$xyz.est
z.lim <- range(surf[["z"]], na.rm = TRUE)
image.plot(surf, xaxs = "r", yaxs = "r", main = "Interpolation of observed Mg")
points(coords)
image.plot(toImage(nut.pred.grid["Mg.mu"]), xaxs = "r", + yaxs = "r", zlim = z.lim, main = "Mean of Mg prediction")
points(coords)
Figure 5: Interpolated surface of observed log nutrient concentrations and mean of each pixel’s posterior predictive distribution.
Finally, we take a look at the standard deviation of prediction. With such a small spatial range, increased precision does not extend far from the sample locations.

```r
> par(mfrow = c(3, 2))
> surf <- mba.surf(cbind(coords, log.nut[, "Ca"]), no.X = res, +   no.Y = res, extend = FALSE)$xyz.est
> image.plot(surf, main = "Interpolation of observed Ca")
> points(coords)
> surf <- mba.surf(cbind(pred.coords, Ca.pred.sd), no.X = res, +   no.Y = res, extend = FALSE)$xyz.est
> image.plot(surf, main = "Standard deviation of Ca prediction")
> points(coords)
> surf <- mba.surf(cbind(coords, log.nut[, "K"]), no.X = res, +   no.Y = res, extend = FALSE)$xyz.est
> image.plot(surf, main = "Interpolation of observed K")
> points(coords)
> surf <- mba.surf(cbind(pred.coords, K.pred.sd), no.X = res, +   no.Y = res, extend = FALSE)$xyz.est
> image.plot(surf, main = "Standard deviation of K prediction")
> points(coords)
> surf <- mba.surf(cbind(coords, log.nut[, "Mg"]), no.X = res, +   no.Y = res, extend = FALSE)$xyz.est
> image.plot(surf, main = "Interpolation of observed Mg")
> points(coords)
> surf <- mba.surf(cbind(pred.coords, Mg.pred.sd), no.X = res, +   no.Y = res, extend = FALSE)$xyz.est
> image.plot(surf, main = "Standard deviation of Mg prediction")
> points(coords)
```

Figure 6: Interpolated surface of observed log nutrient concentrations and standard deviation of each pixel’s posterior predictive distribution.
> image.plot(surf, main = "Standard deviation of K prediction")
> points(coords)
> surf <- mba.surf(cbind(coords, log.nut[, "Mg"]), no.X = res,
+ no.Y = res, extend = FALSE)$xyz.est
> image.plot(surf, main = "Interpolation of observed Mg")
> points(coords)
> surf <- mba.surf(cbind(pred.coords, Mg.pred.sd), no.X = res,
+ no.Y = res, extend = FALSE)$xyz.est
> image.plot(surf, main = "Standard deviation of Mg prediction")
> points(coords)
3 References


