Hierarchical Modeling for Multivariate Spatial Data in R

Andrew O. Finley and Sudipto Banerjee

June 23, 2009

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.

\[
\begin{align*}
> & \text{dat <- read.table("CostaRica/T4.csv", header = T, sep = ",")} \\
> & \text{coords <- as.matrix(dat[, c("X", "Y")])} \\
> & \text{nut.names <- c("Ca", "K", "Mg")} \\
> & \text{log.nut <- log(dat[, nut.names])} \\
> & \text{par(mfrow = c(2, 2))} \\
> & \text{for (i in 1:length(nut.names)) {} } \\
& \quad \text{surf <- mba.surf(cbind(coords, data = log.nut[,} \\
& \quad \quad \quad \text{+ i]), no.X = 100, no.Y = 100)$xyz.est} \\
& \quad \text{image.plot(surf, main = paste("Log ", nut.names[i],} \\
& \quad \quad \quad \text{+ sep = "")})
\end{align*}
\]

\(^1\)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 between ~1-30 m for each of the nutrients; however, the signal is weak, likely due to the paucity of samples. Adjacent study plots, with denser sample arrays, show stronger spatial dependence with effective ranges of ~25 m for Ca, K, and Mg.

```r
> max <- 0.25 * max(as.matrix(dist(dat[, c("X", "Y")])))
> bins <- c(9, 8, 9)
```

Figure 1: Soil nutrient concentrations and sample array.
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).

\[
\begin{pmatrix}
0.5 & 0.2 \\
0.2 & 0.2 & 0.3
\end{pmatrix}
\] (2)

In the following code block we define the model parameters' starting, tuning, and prior distribution, then call \texttt{spMvLM}. Again, for brevity, we have stored the samples from a previous run which took $\sim$30 minutes to collect 50,000 samples. Recall, the sampler must invert the $80 \times 3 = 240$ inter-site dispersion matrix for each iteration.

```r
> q <- 3
> A.starting <- diag(0.5, q)[lower.tri(diag(1, q), TRUE)]
> L.starting <- diag(0.05, q)[lower.tri(diag(1, q), TRUE)]
> A.tuning <- matrix(0.01, q, q)
> diag(A.tuning) <- 0.1
> A.tuning <- A.tuning[lower.tri(diag(1, q), TRUE)]
> L.tuning <- rep(0.03, length(L.starting))
```
Figure 2: Isotropic semivariograms for log nutrient concentrations.
n.samples <- 10
nut.spMvLM <- spMvLM(list(Ca ~ 1, K ~ 1, Mg ~ 1), coords = coords,
+ data = log.nut, starting = list(beta = rep(1, q),
+ phi = rep(3/20, q), A = A.starting, L = L.starting),
+ sp.tuning = list(phi = rep(0.3, q), A = A.tuning,
+ L = L.tuning), priors = list(phi.Unif = rep(c(3/200,
+ 3/10), q), K.IW = list(q + 1, diag(0.1, q)),
+ Psi.IW = list(q + 1, diag(0.1, q))), cov.model = "exponential",
+ n.samples = n.samples, sub.samples = c(1, n.samples,
+ 1), verbose = FALSE, n.report = 500)

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

load(file = "R-data/nut.spMvLM")
p.samples <- nut.spMvLM$p.samples
p.samples[, paste("phi_", 1:q, sep = "")]
+ <- 3/p.samples[, paste("phi_", 1:q, sep = "")]
colnames(p.samples)[colnames(p.samples) == paste("phi_",
+ 1:q, sep = ")]
<- paste("Eff. range ", 1:q, sep = "")
summary(mcmc(p.samples))$quantiles[, c(1, 3, 5)]

2.5%  50%  97.5%
(Intercept).mod1 4.623087477 4.861365890 5.0798522
(Intercept).mod2 4.604892264 4.736641178 4.8667658
(Intercept).mod3 3.864308597 4.041630676 4.2036569
K[1,1] 0.161299765 0.423327047 0.6564792
K[2,1] 0.045591810 0.188027426 0.2992746
K[3,1] 0.096406134 0.279723053 0.4304829
K[2,2] 0.027218855 0.119423104 0.1991463
K[3,2] 0.034165264 0.137652213 0.2209486
K[3,3] 0.092336863 0.220283605 0.3373494
Psi[1,1] 0.014663672 0.048936067 0.2984469
Psi[2,1] -0.026141349 0.009634405 0.1490238
Psi[3,1] -0.011926932 0.015572044 0.2005735
Psi[2,2] 0.017450927 0.055996850 0.1516136
Psi[3,2] -0.008061224 0.024366594 0.1393828
Psi[3,3] 0.013256463 0.043336751 0.1815596
Eff. range 1 10.324638677 16.926401192 53.4271977
Eff. range 2 10.365116771 17.448765898 112.9154112
Eff. range 3 10.788706517 23.060745935 143.5412852

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))
w <- rowMeans(nut.spMvLM$sp.effects)
w.Ca <- w[seq(1, length(w), q)]
w.K <- w[seq(2, length(w), q)]
> w.Mg <- w[seq(3, length(w), q)]
> expand.range <- function(x, p = 0.05) {
+   x <- range(x, na.rm = TRUE)
+   x[1] <- x[1] - p * abs(x[1])
+   x
+ }
> res <- 100
> par(mfrow = c(3, 2))
> surf <- mba.surf(cbind(coords, Ca.resids), no.X = res,
+                 no.Y = res, extend = FALSE)$xyz.est
> z.lim <- expand.range(surf[["z"]])
> image.plot(surf, zlim = z.lim, main = "Ca lm residuals")
> points(coords)
> surf <- mba.surf(cbind(coords, w.Ca), no.X = res, no.Y = res,
+                 extend = FALSE)$xyz.est
> image.plot(surf, zlim = z.lim, main = "Ca spatial effects")
> points(coords)
> surf <- mba.surf(cbind(coords, K.resids), no.X = res,
+                 no.Y = res, extend = FALSE)$xyz.est
> z.lim <- expand.range(surf[["z"]])
> image.plot(surf, zlim = z.lim, main = "K lm residuals")
> points(coords)
> surf <- mba.surf(cbind(coords, K.resids), no.X = res,
+                 extend = FALSE)$xyz.est
> z.lim <- expand.range(surf[["z"]], 0.2)
> image.plot(surf, zlim = z.lim, main = "Mg lm residuals")
> points(coords)
> surf <- mba.surf(cbind(coords, w.Mg), no.X = res, no.Y = res,
+                 extend = FALSE)$xyz.est
> image.plot(surf, zlim = z.lim, main = "Mg spatial effects")
> points(coords)

In the code block above, we unstack the nutrient concentration random spatial effects and compare them with the residual image plots from a non-spatial regression, Figure 3.

2 Prediction

With a sparse sample array, an estimated mean effective range of ~20, and no predictor variables, we cannot expect our prediction to differ much from a
Figure 3: Interpolated surface of the non-spatial model residuals and the mean of the random spatial effects posterior distribution.
constant mean concentration over the domain. In the code block below, we define our prediction grid, construct the prediction design matrix using \texttt{mkMvX}, and call \texttt{spPredict}.

\begin{verbatim}
> 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(nut.spMvLM, start = 1, end = 2,
+ pred.coords = pred.coords, pred.covars = pred.X)

----------------------------------------------------------
Starting prediction
----------------------------------------------------------

> load(file = "R-data/nut.pred")

The \texttt{nut.pred} list object holds the posterior predictive samples for the spatial effects \texttt{w.pred} and response \texttt{y.pred}. Again, like with the random spatial effect in the \texttt{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 \texttt{sp SpatialGridDataFrame} then subsequently to a format that can be plotted by the \texttt{image} or \texttt{fields image.plot} function.

\begin{verbatim}
> y.pred.mu <- apply(nut.pred$y.pred, 1, mean)
> y.pred.sd <- apply(nut.pred$y.pred, 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)]
> 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,
\end{verbatim}
> 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)

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.

> 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,
Figure 4: Interpolated surface of observed log nutrient concentrations and mean of each pixel’s posterior predictive distribution.
Figure 5: Interpolated surface of observed log nutrient concentrations and standard deviation of each pixel’s posterior predictive distribution.

```r
> points(coords)
> 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


