Hierarchical Modeling for Multivariate Spatial Data in R

Andrew O. Finley and Sudipto Banerjee

October 12, 2012

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.

```
> dat <- read.table("CostaRica/T4.csv", header = T, sep = ",")
> coords <- as.matrix(dat[, c("X", "Y")])
> nut.names <- c("Ca", "K", "Mg")
> log.nut <- log(dat[, nut.names])
```

$$\begin{pmatrix} 1 & & \\ 0.7 & 1 & \\ 0.7 & 0.8 & 1 \end{pmatrix} \tag{1}$$

[1] Data provided by Richard Kobe, Ellen Holste, and Tom Baribault with support from NSF DEB 0640904 & 0743609

```
> par(mfrow = c(2, 2))
> for (i in 1:length(nut.names)) {
+     surf <- mba.surf(cbind(coords, data = log.nut[,
+         i]), no.X = 100, no.Y = 100)$xyz.est
+     image.plot(surf, main = paste("Log ", nut.names[i],
+         sep = ""))
+     points(coords)
+ }
```

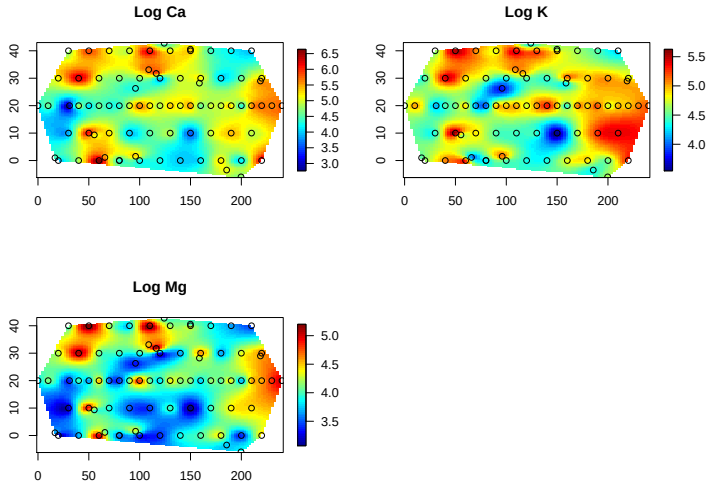

Figure 1: Soil nutrient concentrations and sample array.

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.

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

```
> par(mfrow = c(3, 1))
> for (i in 1:length(nut.names)) {
+     vario <- variog(coords = coords, data = log.nut[,
+         i], uvec = (seq(0, max, length = bins[i])))
+     fit <- variofit(vario, ini.cov.pars = c(0.3, 20/-log(0.05)),
+         cov.model = "exponential", minimisation.function = "nls",
+         weights = "equal")
+     plot(vario, pch = 19, main = paste("Log ", nut.names[i],
+         sep = ""))
+     lines(fit)
+     abline(h = fit$nugget, col = "blue")
+     abline(h = fit$cov.pars[1] + fit$nugget, col = "green")
+     abline(v = -log(0.05) * fit$cov.pars[2], col = "red3")
+ }

variog: computing omnidirectional variogram
variofit: covariance model used is exponential
variofit: weights used: equal
variofit: minimisation function used: nls
variog: computing omnidirectional variogram
variofit: covariance model used is exponential
variofit: weights used: equal
variofit: minimisation function used: nls
variog: computing omnidirectional variogram
variofit: covariance model used is exponential
variofit: weights used: equal
variofit: minimisation function used: nls
```

We continue with fitting a multivariate regression that allows for spatial (K) and non-spatial (Ψ) cross-covariance matrices. We would expect the sum of these matrices to be equal to the aspatial covariance matrix of the observed data (2).

$$\begin{pmatrix} 0.5 & & \\ 0.2 & 0.2 & \\ 0.2 & 0.2 & 0.3 \end{pmatrix} \tag{2}$$

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

```
> 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)
```

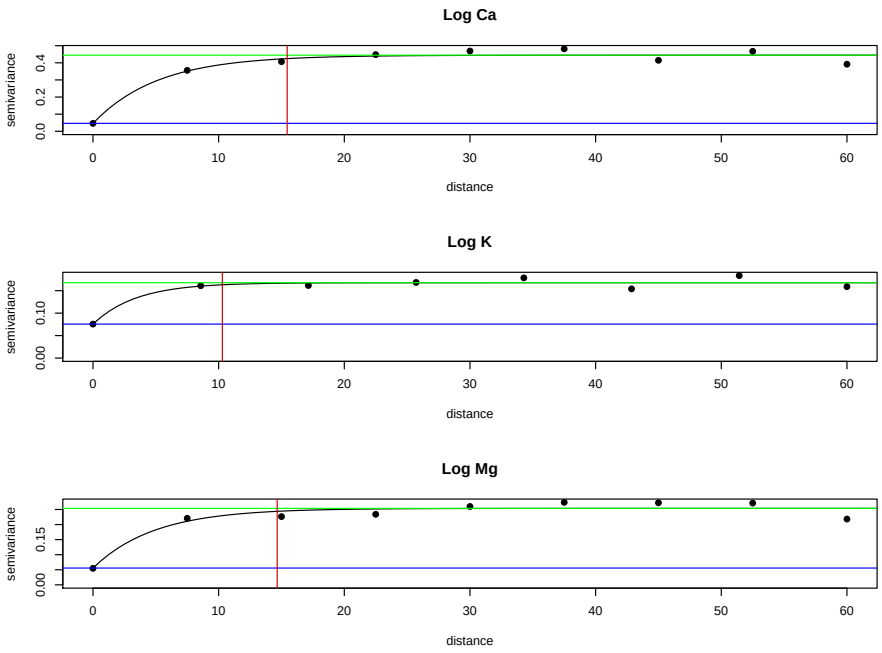

Figure 2: Isotropic semivariograms for log nutrient concentrations.

```
> diag(A.tuning) <- 0.1
> A.tuning <- A.tuning[lower.tri(diag(1, q), TRUE)]
> L.tuning <- rep(0.03, length(L.starting))
> 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 = TRUE, n.report = 5)

----------------------------------------
        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 10.

Priors and hyperpriors:
        beta flat.

        K IW hyperpriors df=4.00000, S=
        0.100        0.000        0.000
        0.000        0.100        0.000
        0.000        0.000        0.100

        Psi IW hyperpriors df=4.00000, S=
        0.100        0.000        0.000
        0.000        0.100        0.000
        0.000        0.000        0.100

        phi Unif hyperpriors
        (0.01500, 0.30000) (0.01500, 0.30000) (0.01500, 0.30000)

Metropolis tuning values:
        A tuning:
                0.100        0.010        0.010        0.100        0.010        0.100

        L tuning:
```

```
                0.030         0.030         0.030         0.030         0.030         0.030

        phi tuning
                0.300         0.300         0.300

Metropolis starting values:
        A starting
        0.500         0.000         0.000
        0.000         0.500         0.000
        0.000         0.000         0.500

        L starting
        0.050         0.000         0.000
        0.000         0.050         0.000
        0.000         0.000         0.050

        phi starting
                0.150         0.150         0.150

-------------------------------------------------
                Sampling
-------------------------------------------------
Sampled: 5 of 10, 50.00%
Report interval Metrop. Acceptance rate: 0.00%
Overall Metrop. Acceptance rate: 0.00%
-------------------------------------------------
Sampled: 10 of 10, 100.00%
-------------------------------------------------
```

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

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 $\mathbf{A}$ 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.

```
> p.samples <- nut.spMvLM$p.samples
> n.samples <- nrow(p.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)
+     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.samples[s, c("K[1,1]", "K[2,1]", "K[3,1]",
+         "K[2,2]", "K[3,2]", "K[3,3]")], q)
+     phi <- p.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])$root
+     }
+ }
> rownames(eff.range) <- paste("Eff. range ", 1:q, sep = "")
> round(summary(mcmc(cbind(p.samples, t(eff.range))))$quantiles[,
+     c(1, 3, 5)], 3)
```

	2.5%	50%	97.5%
(Intercept).mod1	4.642	4.861	5.085
(Intercept).mod2	4.609	4.736	4.873
(Intercept).mod3	3.879	4.042	4.216
K[1,1]	0.343	0.452	0.610
K[2,1]	0.155	0.206	0.288
K[3,1]	0.235	0.308	0.405
K[2,2]	0.082	0.132	0.204
K[3,2]	0.103	0.160	0.221
K[3,3]	0.179	0.244	0.323
Psi[1,1]	0.017	0.028	0.058
Psi[2,1]	-0.021	-0.001	0.028
Psi[3,1]	-0.011	0.001	0.036
Psi[2,2]	0.021	0.042	0.085
Psi[3,2]	-0.012	0.008	0.048
Psi[3,3]	0.011	0.023	0.070
phi_1	0.097	0.185	0.271
phi_2	0.018	0.175	0.294
phi_3	0.019	0.146	0.274
Eff. range 1	11.066	16.223	30.857
Eff. range 2	11.262	16.759	31.669
Eff. range 3	11.886	17.877	49.273

In the code block below, we unstack the nutrient concentration spatial ran-

dom effects and compare them with the residual image plots from a non-spatial regression, Figure 3.

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

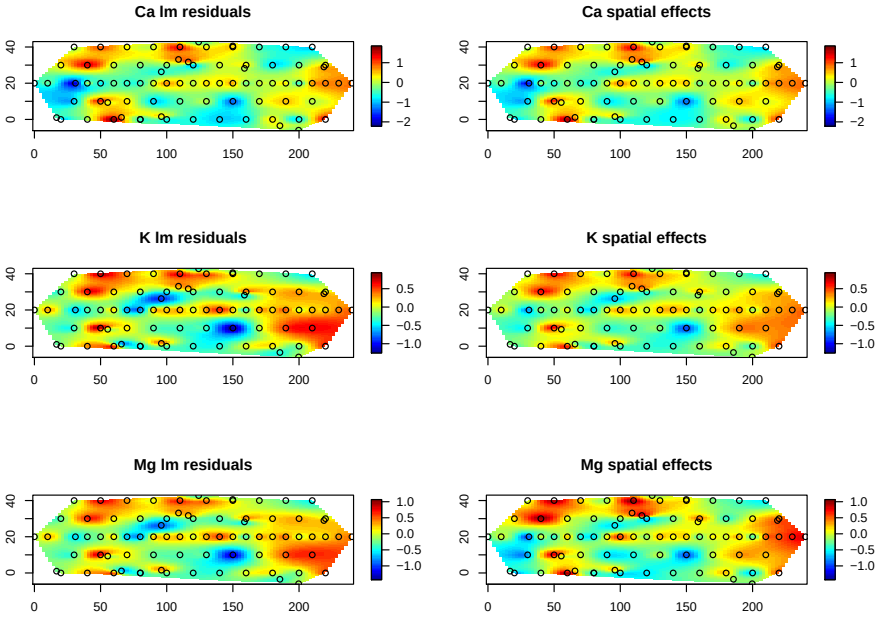

Figure 3: Interpolated surface of the non-spatial model residuals and the mean of the spatial random effects posterior distribution.

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 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`.

```
> 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 `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$y.pred, 1, mean)
> y.pred.sd <- apply(nut.pred$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,
+     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)
```

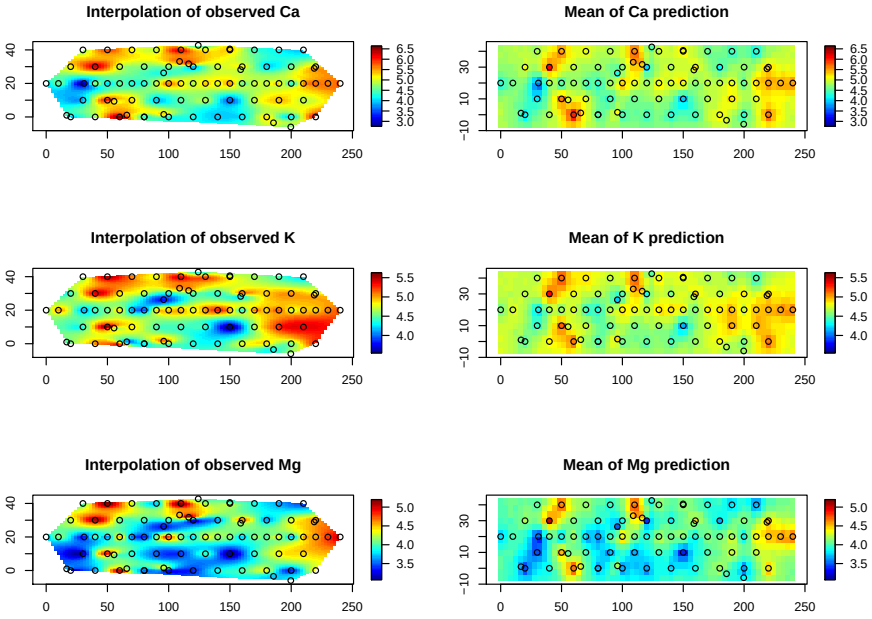

Figure 4: Interpolated surface of observed log nutrient concentrations and mean of each pixel's posterior predictive distribution.

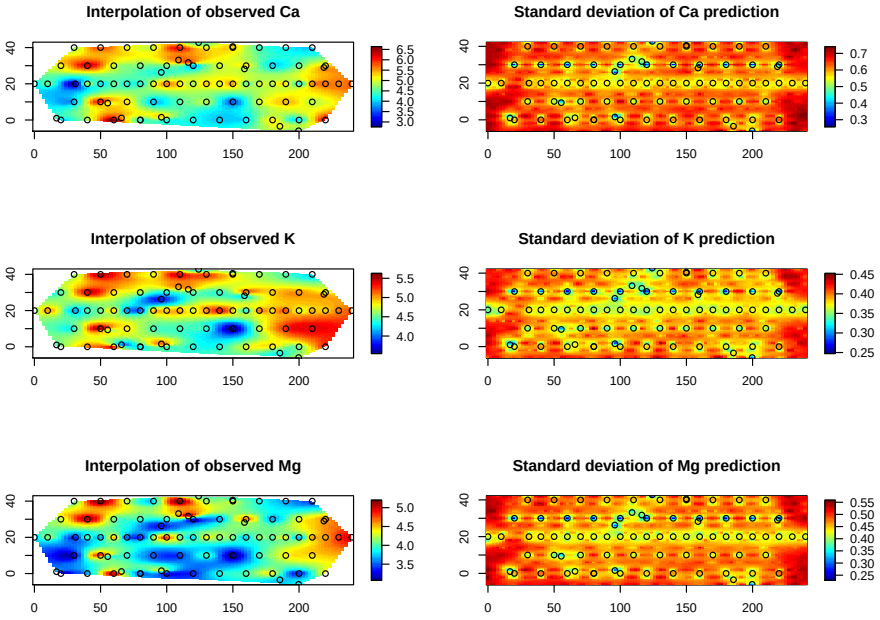

Figure 5: Interpolated surface of observed log nutrient concentrations and standard deviation of each pixel's posterior predictive distribution.

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,
+     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

Banerjee, S., Carlin, B.P., and Gelfand, A.E. (2004). *Hierarchical Modeling and Analysis for Spatial Data*, Boca Raton, FL: Chapman and Hall/CRC Press.

Bivand, R.B., Pebesma, E.J., and Gómez-Rubio, V. (2008). *Applied Spatial Data Analysis with R*, UseR! Series, Springer.

Diggle, P.J. and Riberio, P.J. (2007). *Model-based Geostatistics*, Series in Statistics, Springer.

Gelman, A., Carlin, J.B., Stern, H.S., and Rubin, D.B. (2004). *Bayesian Data Analysis*. Second Edition. Boca Raton, FL: Chapman and Hall/CRC Press.

Gelfand, A.E., Schmidt, A.M. Banerjee, S., and Sirmans, C.F. (2004). Nonstationary multivariate process modelling through spatially varying coregionalization (with discussion), *TEST*, **13**:263–312.