Hierarchical Modeling for Non-Gaussian Spatial Data in R

Andrew O. Finley and Sudipto Banerjee
June 22, 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(maptools)
- library(rgdal)
- library(sp)

We will use forest inventory data from the U.S. Department of Agriculture Forest Service, Bartlett Experimental Forest (BEF), Bartlett, NH. This dataset holds 1991 and 2002 forest inventory data for 437 plots. Variables include species specific basal area and total tree biomass; inventory plot coordinates; slope; elevation; and tasseled cap brightness (TC1), greenness (TC2), and wetness (TC3) components from spring, summer, and fall 2002 Landsat images.

We use these data to demonstrate some basics of spatial data manipulation, visualization, and univariate spatial logistic regression analysis. The regression model will be used to make prediction of high relative eastern hemlock (EH) basal area\(^1\) for every image pixel across the BEF.

We begin by removing non-forest inventory plots, creating the binary response variable EH which indicates those plots with at least 25% eastern hemlock basal area, and taking a look at plot locations across the forest.

2 Spatial data visualization

```r
> data(BEF.dat)
> BEF.dat <- BEF.dat[BEF.dat$BAREA02_TOT > 0,]
> EH <- as.integer(BEF.dat[, "EH_02BAREA"] > 0.25)
> coords <- as.matrix(BEF.dat[, c("XUTM", "YUTM")])
> plot(coords, typ = "n", xlab = "Easting (m)", ylab = "Northing (m)")
```

\(^1\)Cross section of the tree trunk at 1.3 meter above ground.
Figure 1: Forest inventory plot locations across the BEF.

```
> points(coords[as.logical(EH), ], pch = 19, cex = 1)
> points(coords[!as.logical(EH), ], pch = 1, cex = 1)
> legend("bottomright", pch = c(19, 1), cex = c(1, 1),
+   legend = c("High EH basal area", "Low EH basal area"),
+   bty = "n")
```

Using elevation (ELEV) and slope (SLOPE) as predictor variables, we fit a non-spatial logistic regression to these data.

```
> x.res <- 200
> y.res <- 200
> EH.logit <- glm(EH ~ ELEV + SLOPE, data = BEF.dat,
+   family = binomial("logit"))
> summary(EH.logit)
```

Call:
`glm(formula = EH ~ ELEV + SLOPE, family = binomial("logit"),
     data = BEF.dat)`
Deviance Residuals:

Min 1Q Median 3Q Max
-1.1440 -0.9578 -0.6714 1.3525 1.8916

Coefficients:

    Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.703549 0.395837 1.777 0.0755 .
ELEV -0.003611 0.001346 -2.683 0.0073 **
SLOPE -0.015746 0.022607 -0.696 0.4861

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 514.41 on 414 degrees of freedom
Residual deviance: 491.95 on 412 degrees of freedom
AIC: 497.95

Number of Fisher Scoring iterations: 4

We continue with fitting a logistic regression with spatial random effects. Unlike the Gaussian response spatial regression, we cannot marginalize over the random spatial effects for logistic and Poisson models (and several other GLMs). Rather, the spatial effects must be updated along with $\beta$, $\sigma^2$, and $\phi$. Further, there is no Gibbs form for the $\beta$. Therefore, the spGLM and spMvGLM functions in spBayes use the Metropolis algorithm to update all parameters. The current updating scheme in these functions is relatively inefficient, and therefore, the spatial effects are slow to converge. The rate of convergence can be greatly increased by providing good starting values for the spatial effects, $w$. These starting values can be found by fixing a few other parameters at reasonable values and sampling from $w$'s posterior distributions.

This done in the code block below, where we fix $\sigma^2$ and $\phi$ and sample $\beta$ and $w$. Then the mean of $w$'s posterior distributions are passed to a subsequent call to spGLM where all parameters are free. Again, for brevity, we have stored the samples from a previous run which took $\sim$30 minutes to collect 25,000 samples.

```r
> beta.starting <- coefficients(EH.logit)
> beta.tuning <- t(chol(vcov(EH.logit)))
> n.samples <- 10
> bef.starting <- spGLM(EH ~ ELEV + SLOPE, data = BEF.dat, +   coords = coords, starting = list(beta = beta.starting, +     phi = 3/600, sigma.sq = 10, w = 0), tuning = list(beta = beta.tuning, +     phi = 0, sigma.sq = 0, w = 0.01), priors = list(phi.Unif = c(3/2000, +     3/10), sigma.sq.IG = c(2, 10)), cov.model = "exponential", +     n.samples = n.samples, sub.samples = c(1, n.samples, +     1), verbose = TRUE, n.report = 500)
```
General model description

Model fit with 415 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.
- sigma.sq IG hyperpriors shape=2.00000 and scale=10.00000
- phi Unif hyperpriors a=0.00150 and b=0.30000

Metropolis tuning values:

- beta tuning:
  0.396 0.000 0.000
  -0.001 0.001 0.000
  0.008 -0.019 0.010

- sigma.sq tuning: 0.00000

- phi tuning: 0.00000

Metropolis starting values:

- beta starting:
  0.704 -0.004 -0.016

- sigma.sq starting: 10.00000

- phi starting: 0.00500

Sampling

Sampled: 10 of 10, 100.00%

R-code:

```r
> w.starting <- rowMeans(bef.starting$sp.effects)
> load(file = "R-data/w.starting")
> bef.sp <- spGLM(EH ~ ELEV + SLOPE, data = BEF.dat,
+     coords = coords, starting = list(beta = beta.starting,
```
phi = 3/600, sigma sq = 5, w = w.starting),
+ tuning = list(beta = beta.tuning, phi = 0.05, sigma sq = 0.03,
+ w = 0.01), priors = list(phi.Unif = c(3/1000,
+ 3/10), sigma sq.IG = c(2, 5)), cov.model = "exponential",
+ n.samples = n.samples, sub.samples = c(1, n.samples,
+ 1), verbose = TRUE, n.report = 500)

----------------------------------------
General model description
----------------------------------------
Model fit with 415 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.
  sigma sq IG hyperpriors shape=2.00000 and scale=5.00000
  phi Unif hyperpriors a=0.00300 and b=0.30000

Metropolis tuning values:
  beta tuning:
  0.396   0.000   0.000
  -0.001  0.001   0.000
  0.008  -0.019   0.010

  sigma sq tuning: 0.17321

  phi tuning: 0.22361

Metropolis starting values:
  beta starting:
  0.704  -0.004  -0.016

  sigma sq starting: 5.00000

  phi starting: 0.00500

----------------------------------------
Sampling
----------------------------------------
> load(file = "R-data/bef.sp")
> summary(mcmc(bef.sp$p.samples))$quantiles

<table>
<thead>
<tr>
<th></th>
<th>2.5%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>97.5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>1.478660199</td>
<td>2.3667873563</td>
<td>2.863696390</td>
<td>3.458907845</td>
<td>4.612803772</td>
</tr>
<tr>
<td>ELEV</td>
<td>-0.017227859</td>
<td>-0.0139828258</td>
<td>-0.012222449</td>
<td>-0.010431924</td>
<td>-0.007586838</td>
</tr>
<tr>
<td>SLOPE</td>
<td>-0.041481068</td>
<td>0.0008216242</td>
<td>0.021933429</td>
<td>0.050088514</td>
<td>0.097661564</td>
</tr>
<tr>
<td>sigma.sq</td>
<td>5.249876903</td>
<td>6.0817824268</td>
<td>6.696966338</td>
<td>7.478480674</td>
<td>9.302975206</td>
</tr>
<tr>
<td>phi</td>
<td>0.003430329</td>
<td>0.0041069461</td>
<td>0.004592007</td>
<td>0.005116136</td>
<td>0.006175302</td>
</tr>
</tbody>
</table>

Now let's take a look at surfaces of EH versus the fitted probabilities from the spatial regression, Figure 2. Note, we are interpolating over the binary EH, but this is just meant to give us a feel for how the fitted values compare with patterns seen in the response.

> logit.fitted <- function(eta) {
+ 1/(1 + exp(-eta))
+ }
> fitted <- apply(bef.sp$X %*% t(bef.sp$p.samples[, 1:3]) + 
+ bef.sp$sp.effects, 1, logit.fitted)
> fitted.mu <- apply(fitted, 2, mean)
> par(mfrow = c(1, 2))
> surf <- mba.surf(cbind(coords, EH), no.X = x.res, no.Y = y.res,
+ extend = FALSE)$xyz.est
> image.plot(surf, xaxs = "r", yaxs = "r", main = "Observed EH")
> surf <- mba.surf(cbind(coords, fitted.mu), no.X = x.res,
+ extend = FALSE)$xyz.est
> image.plot(surf, xaxs = "r", yaxs = "r", main = "Fitted probabilities")

3 Prediction

In the code block below, we define our prediction grid within the BEF and call spPredict. Again, we are using samples from a previous spPredict run that was based on post burn-in and thinned posterior samples (i.e., note we specify start=1 and start=2 only to save computing time in this illustration). spPredict took ∼30 minutes to return the posterior predictive samples.
Figure 2: Interpolated surface of EH and the mean of the fitted posterior distribution.
BEF.shp <- readShapePoly("BEF-data/BEF_bound.shp")
BEF.poly <- as.matrix(BEF.shp@polygons[[1]]@Polygons[[1]]@coords)
BEF.grids <- readGDAL("BEF-data/dem_slope_lolosptc_clip_60.img")

BEF-data/dem_slope_lolosptc_clip_60.img has GDAL driver GTiff
and has 81 rows and 81 columns

pred.covars <- cbind(BEF.grids[["band1"]], BEF.grids[["band2"]])
pred.covars <- cbind(rep(1, nrow(pred.covars)), pred.covars)
pred.coords <- SpatialPoints(BEF.grids)@coords
pred.covars <- pred.covars[pointsInPoly(BEF.poly, pred.coords), + ]
pred.coords <- pred.coords[pointsInPoly(BEF.poly, pred.coords), + ]

bef.EH.pred <- spPredict(bef.sp, start = 1, end = 2,
+ pred.coords = pred.coords, pred.covars = pred.covars,
+ verbose = FALSE)

The bef.EH.pred list object holds the posterior predictive samples for the
spatial effects w.pred and response y.pred. 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.

EH.mu <- apply(bef.EH.pred$y.pred, 1, mean)
EH.sd <- apply(bef.EH.pred$y.pred, 1, sd)
EH.w.mu <- apply(bef.EH.pred$w.pred, 1, mean)
EH.w.sd <- apply(bef.EH.pred$w.pred, 1, sd)
EH.pred.grid <- as.data.frame(list(x = pred.coords[, + 1], y = pred.coords[, 2], EH.mu = EH.mu, EH.sd = EH.sd,
+ EH.w.mu = EH.w.mu, EH.w.sd = EH.w.sd))
coordinates(EH.pred.grid) <- c("x", "y")
gridded(EH.pred.grid) <- TRUE
toImage <- function(x) {
+ as.image.SpatialGridDataFrame(x)
+ }
par(mfrow = c(1, 2))
plot(coords, typ = "n", xlab = "Easting (m)", ylab = "Northing (m)",
+ main = "Observed EH inventory plots")
points(coords[as.logical(EH), ], pch = 19, cex = 1)
points(coords[!as.logical(EH), ], pch = 1, cex = 1)
legend("bottomright", pch = c(19, 1), cex = c(1, 1),
+ legend = c("High EH basal area", "Low EH basal area"),
+ bty = "n")
image.plot(toImage(EH.pred.grid["EH.mu"]), xaxs = "r",
+ yaxs = "r", xlab = "Easting (m)", ylab = "Northing (m)",
+ main = "Mean predicted probability of EH")
points(coords)
Figure 3: Observed EH and mean of each pixel’s posterior predictive distribution.

```r
> par(mfrow = c(1, 2))
> image.plot(toImage(EH.pred.grid["EH.sd"]), xaxs = "r",
+   yaxs = "r", xlab = "Easting (m)", ylab = "Northing (m)",
+   main = "SD predicted probability of EH")
> image.plot(toImage(EH.pred.grid["EH.w.mu"]), xaxs = "r",
+   yaxs = "r", xlab = "Easting (m)", ylab = "Northing (m)",
+   main = "Mean predicted spatial effect of EH")
```
Figure 4: Standard deviation and mean of each pixel’s EH and w posterior predictive distribution, respectively.
4 References


