The ordinary linear regression model

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

- library(coda)
- library(rjags)
- library(fields)
- library(MBA)
- library(geoR)

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 summer 2002 Landsat images.

These data are used to demonstrate some basics of spatial data manipulation, visualization, and univariate spatial regression analysis using JAGS and R. In a prior analysis these models were used make prediction of biomass for every image pixel across the BEF.

In the interest of time, we will only fit a few models using 50 plot locations. The code below reads in the BEF data then takes a subset of locations for subsequent model development.

```r
> set.seed(1)
> BEF <- read.table("data/BEF.dat", header = TRUE)
> names(BEF)
[1] "XUTM" "YUTM" "BIO" "ELEV" "SLOPE" "TC1" "TC2"
[8] "TC3"
> n.mod <- 50
> mod <- sample(1:nrow(BEF), size = n.mod)
> BEF.mod <- BEF[mod, ]
```

Our objective is to obtain an estimate, with an associated measure of uncertainty, of biomass as a continuous surface over the domain. We can gain a non-statistical estimate of this surface using the MBA package which provides efficient interpolation of large data sets using multilevel B-splines. The result of the mba.surf function can be passed to image or image.plot to produce \( R^2 \) depictions, Figure 1.

```r
> coords <- as.matrix(BEF[, c("XUTM", "YUTM")])
> bio <- BEF$BIO
> par(mfrow = c(1, 2))
> surf <- mba.surf(cbind(coords, bio), no.X = 100,
+    no.Y = 100, extend = FALSE)$xyz.est
> image.plot(surf, xaxs = "r", yaxs = "r", xlab = "Easting (m)",
+    ylab = "Northing (m)"
```
As noted above, we will use several topographic and remotely sensed predictor variables to help explain the variability in tree biomass. It is often useful to create surfaces of these variables to gain a sense of how they are distributed and covary. As in any regression analysis, we must check and meet all model assumptions. Figure 2 suggests some of the predictor variables are collinear. A simple correlation matrix confirms this and we drop TC1 and SLOPE.

```r
> X.names <- c("ELEV", "SLOPE", "TC1", "TC2", "TC3")
> par(mfrow = c(2, 3))
> for (i in 1:5) {
+   surf <- mba.surf(cbind(coords, BEF[, X.names[i]]),
+                  no.X = 100, no.Y = 100, extend = FALSE)$xyz.est
+   image.plot(surf, xaxs = "r", yaxs = "r", main = X.names[i])
+ }
> round(cor(BEF[, X.names]), 2)

       ELEV SLOPE TC1  TC2  TC3
ELEV  1.00  0.69 -0.11 -0.02 -0.35
SLOPE 0.69  1.00 -0.06  0.03 -0.25
TC1 -0.11 -0.06  1.00  0.90 -0.63
TC2 -0.02  0.03  0.90  1.00 -0.43
TC3 -0.35 -0.25 -0.63 -0.43  1.00

> X.names <- c("ELEV", "TC2", "TC3")
```

Figure 1: (left) Forest inventory plot locations across the BEF. (right) Histogram of the response variable.
Figure 2: Interpolation of predictor variables using a Multilevel B-spline.
Providing initial parameter values almost always helps the MCMC sampling algorithm converge. Initial values can often be obtained by fitting a simpler model and/or from least squares or maximum likelihood estimates of the proposed model parameters. Here, `lm` provides the least squares estimates of $\beta$'s and $\sigma$. These estimates are also helpful for selecting priors and associated hyper-parameters.

```r
> bio.mod <- BEF.mod$BIO
> X.mod <- as.matrix(BEF.mod[, X.names])
> lm.m <- lm(bio.mod ~ X.mod)
> summary(lm.m)

Call:
  lm(formula = bio.mod ~ X.mod)

Residuals:
   Min     1Q   Median     3Q    Max
-199.682 -34.623    4.611  39.833  130.176

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)            -373.18965  334.29742  -1.116   0.2701
X.modELEV              0.08074    0.07028   1.149   0.2565
X.modTC2               1.75764    0.77398   2.271   0.0279 *
X.modTC3               2.90547    2.16804   1.340   0.1868
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 60.74 on 46 degrees of freedom
Multiple R-squared:  0.107 ,  Adjusted R-squared:  0.04881
F-statistic: 1.838 on 3 and 46 DF,  p-value: 0.1535

Given this set of predictors, we can fit an initial model and inspect the chain behavior. As in the previous exercises, we define the data objects needed in the JAGS model along with parameter starting values. Our call to `jags.model` requests three MCMC chains of 50000 iterations.

```r
> X.mod <- cbind(1, X.mod)
> n <- nrow(X.mod)
> p <- ncol(X.mod)
> data <- list(y = bio.mod, X = X.mod, n = n, p = p)
> inits <- list(beta = coefficients(lm.m), sigma.sq = 4000)
> jags.m <- jags.model(file = "ex-2a.jag", data = data,
                      + inits = inits, n.chains = 3, n.adapt = 1000)

Compiling model graph
  Resolving undeclared variables
  Allocating nodes
  Graph Size: 317

Initializing model

The JAGS model file `ex-2a.jag` is provided below. Here, the regression coefficients are assumed to be normally distributed and we place a Gamma prior on the variance parameter $\sigma^2$. Take some time to look at JAGS Gamma distribution definition `dgamma(r,\lambda)`, where $r$ and $\lambda$ are the shape and rate hyper-parameters, respectfully. Following the manual definition, the distribution’s mean is $r/\lambda$ and variance is $r/\lambda^2$. Given
an estimate of $\sigma^2$ of about 4000, and wanting to keep a large variance to reflect our uncertainty in this parameter's value, we use $dgamma(r = 4, \lambda = 0.001)$.

Note, I use the parameter `sigma.sq.prior.check` to check that I have correctly defined the Gamma distribution hyper-parameters. You'll find that everybody has their favorite distribution specification and it is easy to get the hyper-parameters mixed up across definitions.

```r
model{
  for(i in 1:n){
    y[i] ~ dnorm(mu[i], tau.sq)
  }

  mu <- X*b

  for(i in 1:p){
    beta[i] ~ dnorm(0, 0.000001)
  }

  tau.sq <- 1/sigma.sq
  sigma.sq ~ dgamma(4, 0.001)

  sigma.sq.prior.check ~ dgamma(4, 0.001)
}

The trace plot of `sigma.sq.prior.check` in Figure 3 confirms we set the hyper-parameters correctly, i.e., mean of $\sim$4000 and variance of $\sim$4e+06.

```r
> params <- c("beta", "sigma.sq", "sigma.sq.prior.check")
> samps <- coda.samples(jags.m, params, n.iter = 50000)
> plot(samps, density = FALSE)
``` 

We will also check the Gelman-Rubin Diagnostic to assess convergence, Figure 4. Given the potential scale reduction factors are all less then $\sim$1.1 and visual inspection of the chains suggests they are mixing well, we can concluded the chains have converged and we can turn our attention to summarizing the posterior samples.

```r
> gelman.diag(samps)

Potential scale reduction factors:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>beta[1]</td>
<td>1.04</td>
<td>1.13</td>
</tr>
<tr>
<td>beta[2]</td>
<td>1.02</td>
<td>1.05</td>
</tr>
<tr>
<td>beta[3]</td>
<td>1.00</td>
<td>1.02</td>
</tr>
<tr>
<td>beta[4]</td>
<td>1.06</td>
<td>1.17</td>
</tr>
<tr>
<td>sigma.sq</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>sigma.sq.prior.check</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Multivariate psrf

1.04

> gelman.plot(samps)
```
Figure 3: Posterior sample chain and density plots for $\beta$, $\sigma^2$, and $\sigma^2.prior.check$.

Figure 4: Gelman-Rubin diagnostic plots for $\beta_0$ and $\sigma^2$.

With the exception of $\beta_3$ (associated with T2), all of the regression coefficients’ 95% credible intervals include zero, which suggests they are not contributing much to explaining the variability in tree biomass (if you re-run this analysis using all 437 data points you should see a different story).

```r
> burn.in <- 25000
> round(summary(window(samps, start = burn.in))$quantiles[,}
```
Considering the addition of spatial random effects

As mentioned above, we must always check and meet model assumptions. Given these are spatially indexed data and the surface plot of tree biomass suggested that locations near each other had similar values, we should check if our residuals show any signs of spatial dependence. Not only would spatial dependence violate the assumption of this model, but we might be able to use this dependence to help improve the accuracy and precision of predictions. A surface plot of model residuals and a semivariogram (see, e.g., Banerjee et al. 2004) aid in assessing the level of lingering spatial dependence. Let’s begin by generating fitted values from the posterior samples of $\beta$ and $\sigma^2$. I’ll show you an easier way of doing later, but for now it’s instructive to see how to generate, then summarize, the posterior of the fitted values using composition sampling. The code below produces the posterior mean for each observation and store them in `bio.fitted`.

```r
> y.fitted <- function(omega, X) {
+   n <- nrow(X)
+   p <- ncol(X)
+   beta <- omega[1:p]
+   sigma.sq <- omega[p + 1]
+   rnorm(n, X %*% beta, sqrt(sigma.sq))
+ }
> bio.fitted <- apply(as.matrix(window(samps, start = burn.in)),
+   1, y.fitted, X.mod)
> bio.fitted <- apply(bio.fitted, 1, mean)
```

The variogram is created using functions in the `geoR` package and displayed in Figure 5(left).

```r
> resids <- bio.mod - bio.fitted
> coords.mod <- as.matrix(BEF.mod[, c("XUTM", "YUTM")])
> par(mfrow = c(1, 2))
> surf <- mba.surf(cbind(coords.mod, resids), no.X = 100,
+   no.Y = 100, extend = FALSE)$xyz.est
> image.plot(surf, xaxs = "r", yaxs = "r", xlab = "Easting (m)",
+   ylab = "Northing (m)", main = "metric tons per ha")
> max.dist <- 0.5 * max(dist(coords.mod))
> bins <- 10
> v <- variog(coords = coords.mod, data = resids, uvec = (seq(0,
+   max.dist, length = bins)))

variog: computing omnidirectional variogram

> fit.v <- variofit(v, ini.cov.pars = c(2000, 500),
+   fix.nugget = TRUE, nugget = 1000, cov.model = "exponential",
+   minimisation.function = "nls", weights = "equal")
```
The exploratory analysis of the non-spatial model residuals, Figure 5, suggests we need to accommodate spatial dependence among the residuals. As detailed in the lecture slides, this can be achieved by adding a spatial random effect to the mean. Here, we will assume an exponential spatial correlation function with spatial variance and decay parameters $\sigma_0^2$ and $\phi$, respectively. The semivariogram analysis suggests starting values $\sigma_0^2 = 1000$, $\sigma_0^2w = 3000$, and $\phi = 0.0075$. We’ll again assume a non-informative prior for the variance components, i.e., $dgamma(r = 4, \lambda = 0.001)$ and broad spatial support for the spatial decay parameter $\text{dunif}(0.003, 3)$, which corresponds to support between 1 and 1000 distance units.

Fitting this model requires computing the inverse of the $n \times n$ spatial covariance matrix. \texttt{JAGS} does not take advantage of efficient routines for decomposition of symmetric and positive definite matrices (e.g., Cholesky decomposition). Therefore, fitting this and similar models that have structured random effects is computationally challenging (\texttt{BUGS} is no better). See the \texttt{spBayes} R package that is designed to efficiently fit such models.

```r
> D <- as.matrix(dist(coords.mod))
> data <- list(y = bio.mod, X = X.mod, n = n, p = p,
+ D = D)
> inits <- list(beta = coefficients(lm.m), sigma.sq = 2000,
+ sigma.sq.w = 2000, phi = 0.0075)
```
In addition to the spatial covariance parameters, we will also want to monitor the $n$ spatial random effects $w$. As illustrated above, we can obtain fitted values of $y$ in a post-processing step. We can also simply add the to the monitoring list and let JAGS do the work for us.

```r
model{
  for (i in 1:n){
    y[i] ~ dnorm(mu[i] + w[i], tau.sq)
    w.mu[i] ← 0
  }

  mu ← X*b

  for (i in 1:p){
    beta[i] ~ dnorm(0, 0.000001)
  }

  sigma.sq ~ dgamma(2, 0.001)
  tau.sq ← 1/sigma.sq

  sigma.sq.w ~ dgamma(2, 0.001)

  phi ~ dunif(0.003, 3)

  for (i in 1:n){
    for (j in 1:n){
      C.w[i,j] ← sigma.sq.w*exp(-phi*D[i,j])
    }
  }

  P.w ← inverse(C.w)

  w ~ dnmnorm(w.mu, P.w)
}
```

Based on the chain trace plots shown in Figure 6 we clearly need to run this sampler much longer to achieve satisfactory convergence. Also note that with only 50 observations, the variance parameters and spatial range are not well identified by the data. Regardless, the flexibility of the Gaussian process spatial random effects will provide excellent fit to the data and most likely improved prediction at new locations. Figure 7 shows that the random effects have captured nearly all of the spatial patterns in the non-spatial model’s residuals, which suggests we have now met the basic model assumptions of independent and identically distributed residuals.
> params <- c("beta", "sigma.sq", "sigma.sq.w", "phi", "+  "w", "y")
> samps <- coda.samples(jags.sp.m, params, n.iter = 10000)
> plot(samps[, c(paste("beta[", 1:p, "]", sep = ""), 
+  "sigma.sq", "sigma.sq.w", "phi")], density = FALSE)

> burn.in <- 5000
> sub.samps <- as.matrix(window(samps, start = burn.in))
> w.hat <- apply(sub.samps[, paste("w[", 1:n, "]", 
+  sep = "")], 2, mean)
> par(mfrow = c(1, 2))
> surf <- mba.surf(cbind(coords.mod, resids), no.X = 100, 
+  no.Y = 100, extend = TRUE)$xyz.est
> image.plot(surf, main = "Non-spatial model residuals")
> surf <- mba.surf(cbind(coords.mod, w.hat), no.X = 100, 
+  no.Y = 100, extend = TRUE)$xyz.est
> image.plot(surf, main = "Spatial random effects")

> bio.fitted.sp <- apply(sub.samps[, paste("y[", 1:n, 
+  "]", sep = "")], 2, mean)
> par(mfrow = c(1, 2))
> plot(bio.fitted, bio.mod, main = "Non-spatial model fitted values")
> plot(bio.fitted.sp, bio.mod, main = "Spatial model fitted values")

Figure 6: Posterior sample chain and density plots for \(\beta, \sigma^2, \text{sigma.w}\), and \(\phi\).

We can use the deviance information criterion (DIC; Spiegelhalter et al. 2002) to compare the fit of the two models. Like AIC, BIC, and similar criterion, lower DIC values indicate better fit. The \(P_D\) value is the effective number of parameters and is used to penalize models with more parameters. As expected, the
addition of the random effects improves model fit. Again, these functions should be called with more iterations than used below. However, given our initial objective was to predict tree biomass at new locations, our criterion should be based on improvements in predictive performance, which does not necessarily correspond to improvements in fit.
> dic.samples(jags.m, n.iter = 10000, thin = 100)

Mean deviance: 553.7
penalty 5.203
Penalized deviance: 558.9

> dic.samples(jags.sp.m, n.iter = 10000, thin = 100)

Mean deviance: 515.9
penalty 48.59
Penalized deviance: 564.5

1 References
