

Exploring spatial data in R

Andrew O. Finley and Sudipto Banerjee

March 9, 2010

1 Data preparation and initial exploration

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

• <code>library(classInt)</code>	• <code>library(MBA)</code>
• <code>library(fields)</code>	• <code>library(maptools)</code>
• <code>library(geoR)</code>	• <code>library(rgdal)</code>
• <code>library(gstat)</code>	• <code>library(rgl)</code>
• <code>library(lattice)</code>	• <code>library(sp)</code>

We will use forest inventory data from a long-term ecological research site in western Oregon (WEF). These data consist of a census of all trees in a 10 ha stand. Diameter at breast height (DBH) and tree height (HT) have been measured for all trees in the stand. For a subset of these trees, the distance from the center of the stem to the edge of the crown was measured at each of the cardinal directions.

Rarely do we have the luxury of a complete census, but rather a subset of trees are sampled using a series of inventory plots. This sample is then used to draw inferences about parameters of interest (e.g., mean stand DBH or correlation between DBH and HT). We defer these analyses to subsequent example sessions. Here, we simply use these data to demonstrate some basics of spatial data manipulation, visualization, and exploratory analysis.

We begin by removing rows that have NA for the variables of interest, creating a vector of DBH and HT, and plotting the tree coordinates.

2 Spatial data visualization

```
> data(WEF.dat)
> WEF.dat <- WEF.dat[!apply(WEF.dat[, c("East_m", "North_m",
+   "DBH_cm", "Tree_height_m", "ELEV_m")], 1, function(x) any(is.na(x))),
+   ]
> DBH <- WEF.dat$DBH_cm
> HT <- WEF.dat$Tree_height_m
```

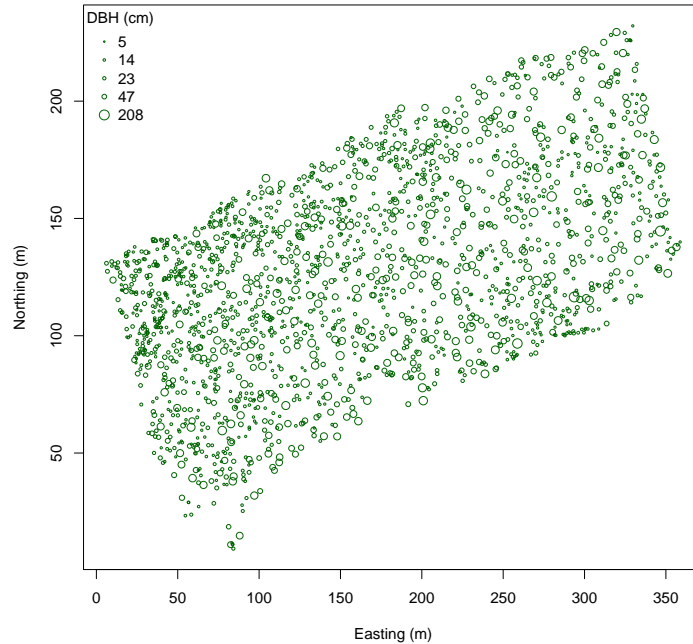


Figure 1: Tree locations on the WEF.

```
> coords <- as.matrix(WEF.dat[, c("East_m", "North_m")])
> plot(coords, pch = 1, cex = sqrt(DBH)/10, col = "darkgreen",
+       xlab = "Easting (m)", ylab = "Northing (m)")
> leg.vals <- round(quantile(DBH), 0)
> legend("topleft", pch = 1, legend = leg.vals, col = "darkgreen",
+       pt.cex = sqrt(leg.vals)/10, bty = "n", title = "DBH (cm)")
```

Let's first take a look at the distribution of DBH across the stand and see if there are any spatial patterns. To do this it is often useful to use a color gradient or ramp to construct the pallet. These can be created using `colorRampPalette` which creates a function that interpolates over a given set of colors. Here we make a color ramp function then create a color palette with five shades.

```
> col.br <- colorRampPalette(c("blue", "cyan", "yellow",
+   "red"))
> col.pal <- col.br(5)
```

We might choose to depict our data with a color pallet consisting of many shades (e.g., 100) or discretize the variable of interest and map each interval to

a color. It is often easier to identify spatial patterns using few shades. Given a pallet, we need to decide how to map the colors to the values of the variable of interest. The **classInt** package offers several useful algorithms for choosing these intervals.

```
> quant <- classIntervals(DBH, n = 5, style = "quantile")
> fisher <- classIntervals(DBH, n = 5, style = "fisher")
> kmeans <- classIntervals(DBH, n = 5, style = "kmeans")
> bclust <- classIntervals(DBH, n = 5, style = "bclust")

Committee Member: 1(1) 2(1) 3(1) 4(1) 5(1) 6(1) 7(1) 8(1) 9(1) 10(1)
Computing Hierarchical Clustering

> par(mfrow = c(2, 2))
> plot(quant, pal = col.pal, xlab = "DBH", main = "Quantile")
> plot(fisher, pal = col.pal, xlab = "DBH", main = "Fisher")
> plot(kmeans, pal = col.pal, xlab = "DBH", main = "Kmeans")
> plot(bclust, pal = col.pal, xlab = "DBH", main = "Bclust")
```

Figure 2 suggests that the algorithm used for mapping colors to the variable could substantially influence our perception of spatial patterns. Indeed, mapping the data using quantile and Bclust based intervals suggest quite different spatial patterns, Figure 3.

```
> quant.col <- findColours(quant, col.pal)
> fisher.col <- findColours(fisher, col.pal)
> par(mfrow = c(1, 2))
> plot(coords, col = quant.col, pch = 19, cex = 0.5,
+       main = "Quantile", xlab = "Easting (m)", ylab = "Northing (m)")
> legend("topleft", fill = attr(quant.col, "palette"),
+       legend = names(attr(quant.col, "table")), bty = "n")
> plot(coords, col = fisher.col, pch = 19, cex = 0.5,
+       main = "Bclust", xlab = "Easting (m)", ylab = "Northing (m)")
> legend("topleft", fill = attr(fisher.col, "palette"),
+       legend = names(attr(fisher.col, "table")), bty = "n")
```

Alternatively, we can choose intervals that are meaningful to the analysis. For instance, in a forest inventory, trees are often identified as sapling, poletimber, sawtimber, and large sawtimber classes, Figure 4.

```
> fixed <- classIntervals(DBH, n = 4, style = "fixed",
+   fixedBreaks = c(0, 12.7, 30.48, 60, max(DBH) +
+   1))
> fixed.col <- findColours(fixed, col.pal)
> plot(coords, col = fixed.col, pch = 19, cex = 0.5,
+       main = "Forestry tree size classes", xlab = "Easting (m)",
+       ylab = "Northing (m)")
```

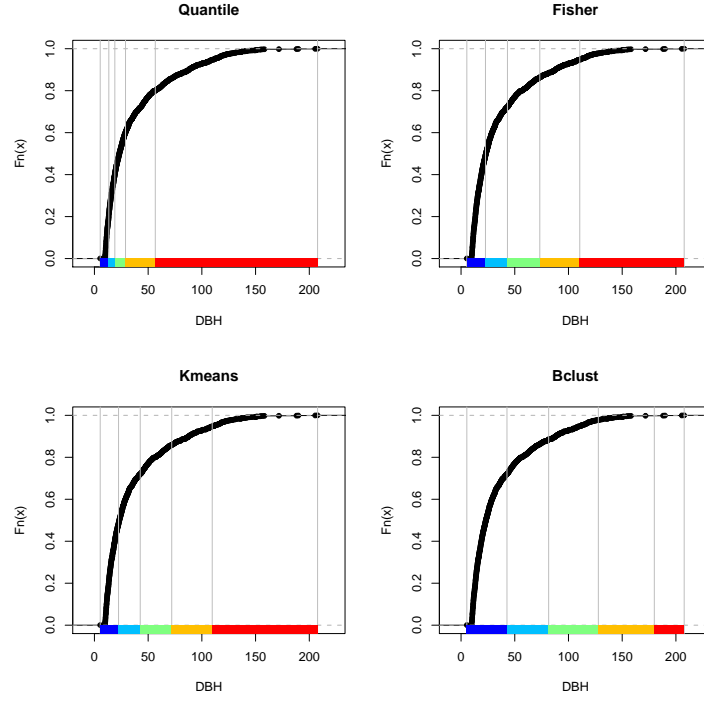


Figure 2: Comparison among several methods for choosing intervals.

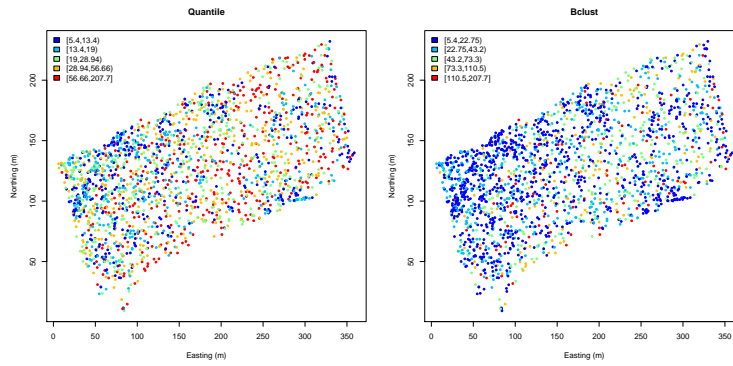


Figure 3: Comparison between maps produced using quantile and bagged clustering interval algorithms.

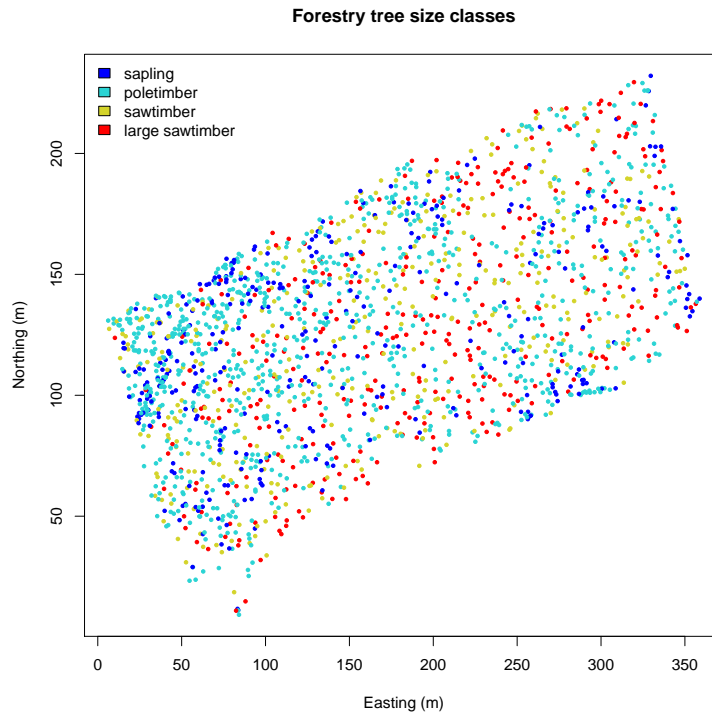


Figure 4: Intervals based on previously defined tree size classes.

```
> legend("topleft", fill = attr(fixed.col, "palette"),
+       legend = c("sapling", "poletimber", "sawtimber",
+       "large sawtimber"), bty = "n")
```

If the data observations are well distributed over the domain, spatial patterns can often be detected by estimating a continuous surface using an interpolation function. Several packages provide suitable interpolators, include the **akima** for linear or cubic spline interpolation and the **MBA** which provides efficient interpolation of large data sets with multilevel B-splines. Functions within both packages produce grids of interpolated values which can be passed to **image** or **image.plot** to produce \mathbb{R}^2 depictions, Figure 5, or to **persp** or similar functions in **rgl** to produce \mathbb{R}^3 depictions, Figure 6.

```
> x.res <- 100
> y.res <- 100
> surf <- mba.surf(cbind(coords, DBH), no.X = x.res,
+   no.Y = y.res, h = 5, m = 2, extend = FALSE)$xyz.est
> image.plot(surf, xaxis = "r", yaxis = "r", xlab = "Easting (m)",
```

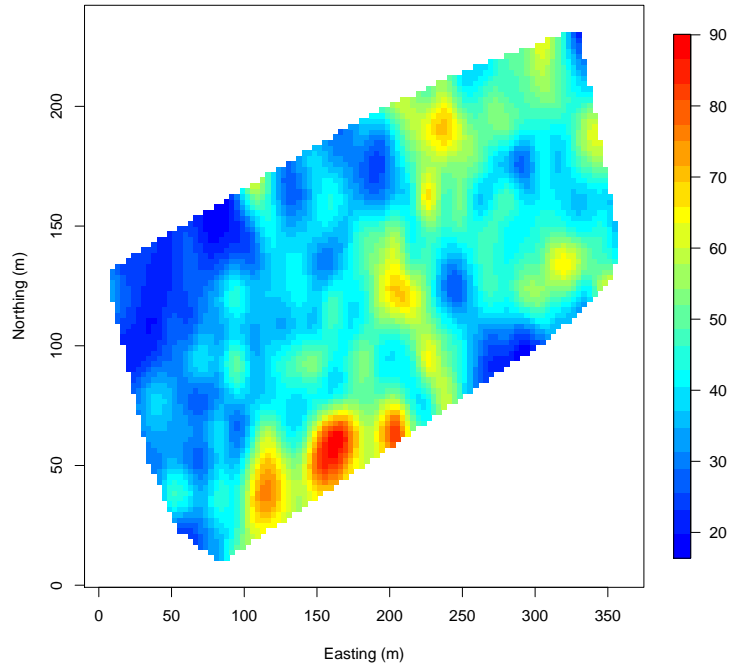


Figure 5: Interpolation of DBH using Multilevel B-spline.

```
+      ylab = "Northing (m)", col = col.br(25))

> col <- rbind(0, cbind(matrix(drape.color(surf[[3]],
+      col = col.br(25)), x.res - 1, y.res - 1), 0))
> surface3d(surf[[1]], surf[[2]], surf[[3]], col = col)
> axes3d()
> title3d(main = "DBH", xlab = "Easting (m)", ylab = "Northing (m)",
+      zlab = "DBH (cm)")

> drape.plot(surf[[1]], surf[[2]], surf[[3]], col = col.br(150),
+      theta = 225, phi = 50, border = FALSE, add.legend = FALSE,
+      xlab = "Easting (m)", ylab = "Northing (m)",
+      zlab = "DBH (cm)")
> image.plot(zlim = range(surf[[3]], na.rm = TRUE),
+      legend.only = TRUE, horizontal = FALSE)
```

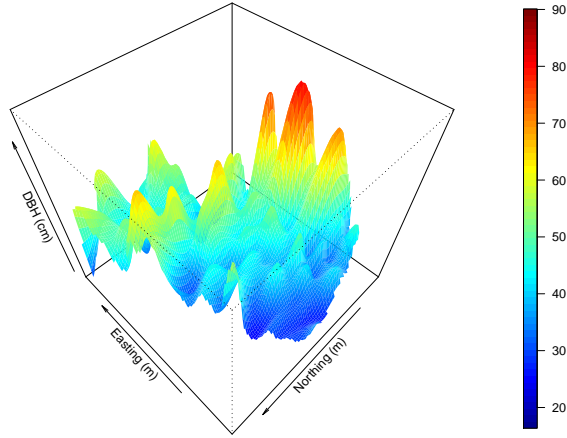


Figure 6: Perspective plot of Multilevel B-spline interpolation of DBH.

3 Variogram Analysis

Our visual inspection of these data suggest that there is some degree of spatial dependence in the distribution of DBH across the WEF. This encourages further exploration using a variogram analysis to quantify the range of spatial dependence and the apportioning of variance into spatial and non-spatial components. Also, within a regression context, are there predictor variables that might explain a portion of the spatial dependence in DBH?

We begin by fitting an isotropic empirical semivariogram using functions within the **geoR** package. In the code block below, we first fit an exponential variogram model to DBH then fit a second variogram model to the residuals of a linear regression of DBH onto tree species. The resulting variograms are offered in Figure 7. 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).

```
> max.dist <- 0.25 * max(iDist(coords))
> bins <- 50
> vario.DBH <- variog(coords = coords, data = DBH,
+   uvec = (seq(0, max.dist, length = bins)))
```

variog: computing omnidirectional variogram

```

> fit.DBH <- variofit(vario.DBH, ini.cov.pars = c(600,
+ 200/-log(0.05)), cov.model = "exponential", minimisation.function = "nls",
+ weights = "equal")

variofit: weights used: equal
variofit: minimisation function used: nls

> lm.DBH <- lm(DBH ~ Species, data = WEF.dat)
> summary(lm.DBH)

Call:
lm(formula = DBH ~ Species, data = WEF.dat)

Residuals:
    Min       1Q   Median       3Q      Max
-78.423  -9.969  -3.561   10.924  118.277

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    89.423      1.303   68.629  <2e-16 ***
SpeciesGF     -51.598      4.133  -12.483  <2e-16 ***
SpeciesNF      -5.873     15.744   -0.373    0.709
SpeciesSF     -68.347      1.461  -46.784  <2e-16 ***
SpeciesWH     -48.062      1.636  -29.377  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 22.19 on 1950 degrees of freedom
Multiple R-squared:  0.5332,    Adjusted R-squared:  0.5323
F-statistic: 556.9 on 4 and 1950 DF,  p-value: < 2.2e-16

> DBH.resid <- resid(lm.DBH)
> vario.DBH.resid <- variog(coords = coords, data = DBH.resid,
+ uvec = (seq(0, max.dist, length = bins)))

variog: computing omnidirectional variogram

> fit.DBH.resid <- variofit(vario.DBH.resid, ini.cov.pars = c(300,
+ 200/-log(0.05)), cov.model = "exponential", minimisation.function = "nls",
+ weights = "equal")

variofit: weights used: equal
variofit: minimisation function used: nls

> par(mfrow = c(1, 2))
> plot(vario.DBH, ylim = c(200, 1200), main = "DBH")
> lines(fit.DBH)

```

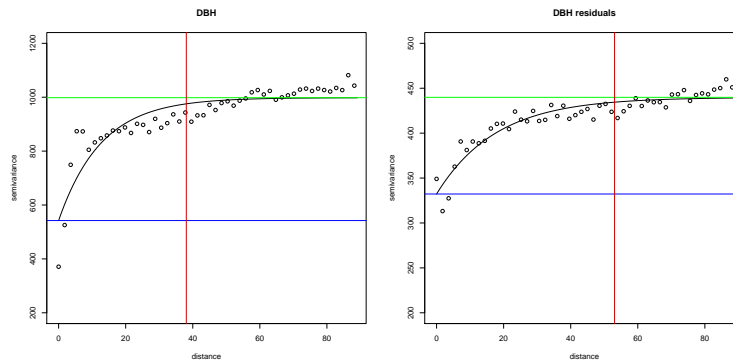



Figure 7: Isotropic semivariograms for DBH and residuals of a linear regression of DBH onto tree species.

```
> abline(h = fit.DBH$nugget, col = "blue")
> abline(h = fit.DBH$cov.pars[1] + fit.DBH$nugget,
+       col = "green")
> abline(v = -log(0.05) * fit.DBH$cov.pars[2], col = "red3")
> plot(vario.DBH.resid, ylim = c(200, 500), main = "DBH residuals")
> lines(fit.DBH.resid)
> abline(h = fit.DBH.resid$nugget, col = "blue")
> abline(h = fit.DBH.resid$cov.pars[1] + fit.DBH.resid$nugget,
+       col = "green")
> abline(v = -log(0.05) * fit.DBH.resid$cov.pars[2],
+       col = "red3")
```

Now let's check for possible anisotropic patterns of spatial dependence using the **geoR** function `variog4` which calculates directional semivariogram.

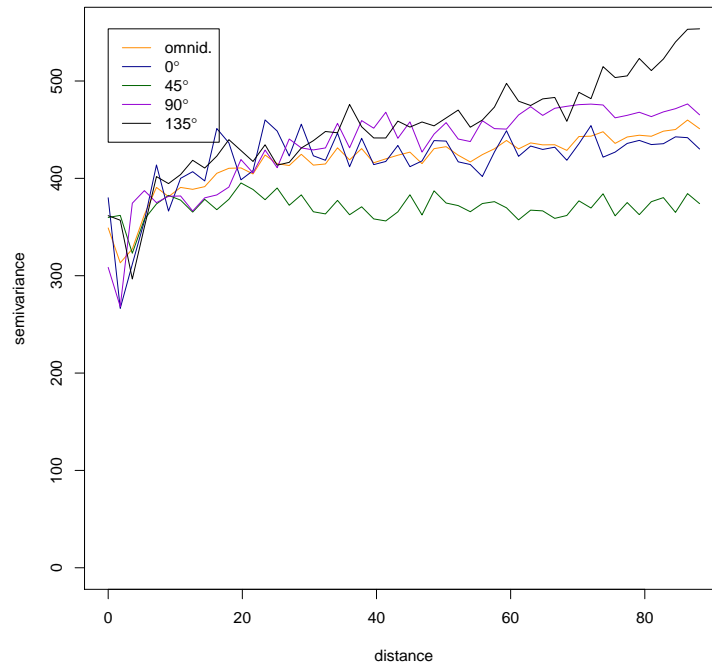


Figure 8: Directional semivariograms for DBH.

```
> vario.DBH.resid <- variog4(coords = coords, data = DBH.resid,
+   uvec = (seq(0, max.dist, length = bins)))

variog: computing variogram for direction = 0 degrees (0 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 45 degrees (0.785 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 90 degrees (1.571 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing variogram for direction = 135 degrees (2.356 radians)
        tolerance angle = 22.5 degrees (0.393 radians)
variog: computing omnidirectional variogram

> plot(vario.DBH.resid, omni = TRUE, main = "DBH residuals",
+   lty = 1, col = c("darkorange", "darkblue", "darkgreen",
+   "darkviolet", "black"))
```

4 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.