Spatial Correlation

Peter Claussen

9/5/2017

Libraries

ape : Analyses of Phylogenetics and Evolution

```
library(ape)
```

```
## Warning: package 'ape' was built under R version 3.3.2
```

ncf : Spatial Nonparametric Covariance Functions

```
library(ncf)
```

```
##
## Attaching package: 'ncf'

## The following object is masked from 'package:ape':
##
##     mantel.test
```

spdep : Spatial Dependence: Weighting Schemes, Statistics and Models

```
library(spdep)
```

```
## Loading required package: sp

## Warning: package 'sp' was built under R version 3.3.2

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 3.3.2

##
## Attaching package: 'spdep'

## The following object is masked from 'package:ape':
##
##     plot.mst
```

Data

```
load(file="sample.dat.Rda")
```

In this section, we extend the concept of autocorrelation to two dimensions, but using pointwise covariance (instead of lag classes). The produces single a measure of spatial correlation across a region.

Moran I

We start with the most common statistic for spatial autocorrelation Moran's I, given by

$$I = \frac{N}{\Sigma_i \Sigma_j w_{ij}} \frac{\Sigma_i \Sigma_j w_{ij}(y_i - \bar{y})(y_j - \bar{y})}{\Sigma_i (y_i - \bar{y})^2}$$

where w is a neighbor weight matrix such that $w_{ii} = 0$.

(Alternate notations - note that $N/\Sigma_i (y_i - \bar{y})^2$ is equivalent to the $1/S_i^2$)

The expected value of Moran's I, under no spatial correlation, is $E(I) = \frac{-1}{N-1}$. If data are perfectly dispersed (i.e. checkerboard), then I = -1, while a balanced or stacked arrangement approaches 1.

Neighbors

There are many ways to compute a neighbor matrix, largely depending on the nature of the spatial data. Data that are generated from areal maps, forming grids or lattices, use rules such as "rook" or "queen" neighbors. We'll consider those when we move to grid cell yield maps. For now, our neighbor weights can be compute more simply. Since our points are randomly selected, we can use the inverse of the distance between points; thus, pairs of points that are closer in space will have larger w_{ij} values. We don't need to be concerned about the dimensions, since the coefficient $1/\Sigma_i \Sigma_j w_{ij}$ normalizes w.

Using our subset data from before, we'll first create a simulated data set with a clear spatial gradient from north to south.

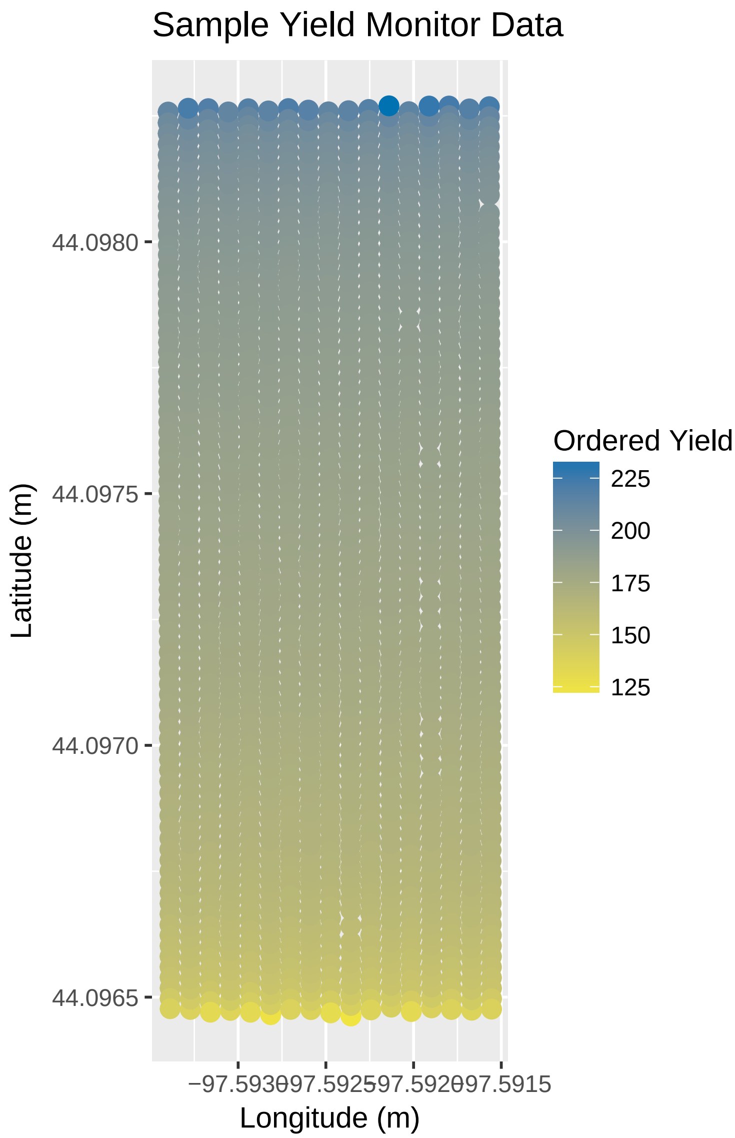
```
YieldOrdered <- sample.dat$Yield[order(sample.dat$Yield)]
LatRank <- rank(sample.dat$Latitude)
sample.dat$YieldOrdered <- YieldOrdered[LatRank]
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#0072B2", "#D55E00", "#F0E442","#CC79A7","#

ggplot(sample.dat, aes(Longitude, Latitude)) +
geom_point(aes(colour = YieldOrdered),size=3) +
scale_colour_gradient(low=cbPalette[7], high=cbPalette[5]) +
labs(colour = "Ordered Yield", x="Longitude (m)", y="Latitude (m)", title = "Sample Yield Monitor Data")
```



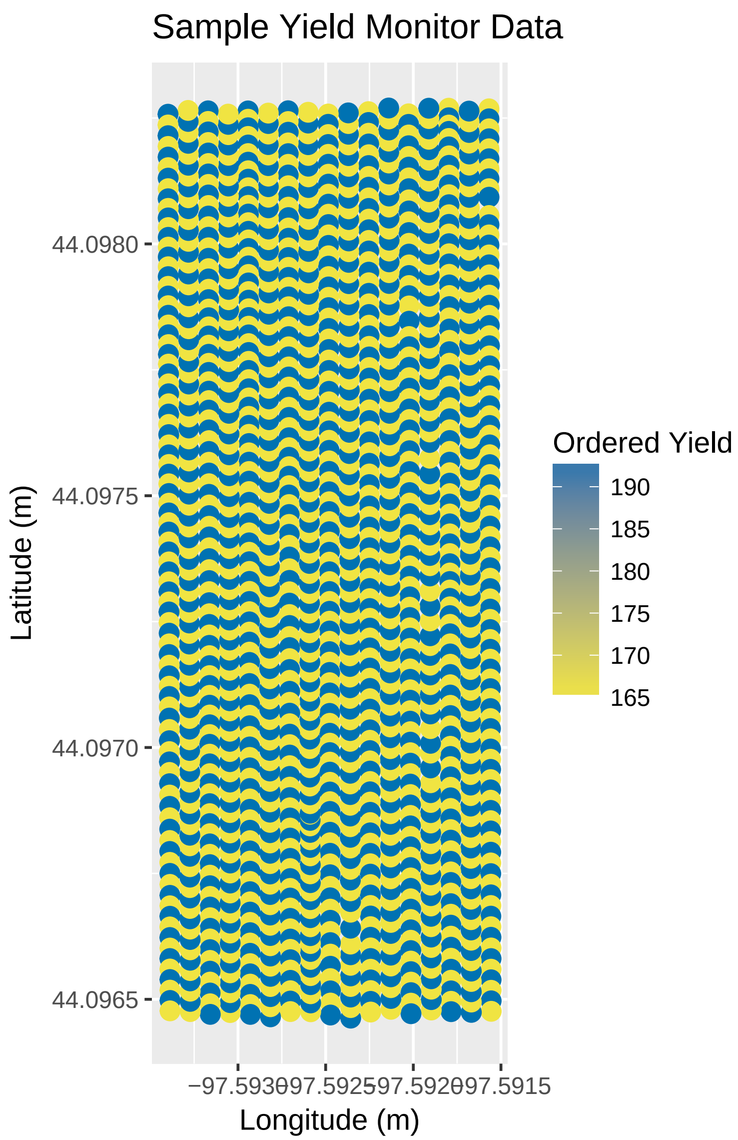
For contrast, we create a checkerboard pattern,

```
mean.yld <- mean(sample.dat$Yield)
sd.yld <- sd(sample.dat$Yield)

is.even <- function(x) {x %% 2 == 0}

for(i in 1:length(sample.dat$Yield)) {
  if(is.even(i)) {
    sample.dat$YieldChecked[i] <- mean.yld + sd.yld
  } else {
    sample.dat$YieldChecked[i] <- mean.yld - sd.yld
  }
}
```

```
ggplot(sample.dat, aes(Longitude, Latitude)) +
geom_point(aes(colour = YieldChecked),size=3) +
scale_colour_gradient(low=cbPalette[7], high=cbPalette[5]) +
labs(colour = "Ordered Yield", x="Longitude (m)", y="Latitude (m)", title = "Sample Yield Monitor Data")
```



Finally, we'll flip alternate points in the ordered gradient to produce a disordered pattern,

```
length(YieldOrdered)
```

```
## [1] 1497
```

```
mid <- floor(length(YieldOrdered)/2)
evens <- (1:mid)*2
odds <- rev(evens)+1
idx <- c()
for(i in 1:mid) {
  idx <- c(idx,evens[i],odds[i])
}
skips <- c(1,idx)
length(skips)
```

```
## [1] 1497
```

```r
head(skips)
```

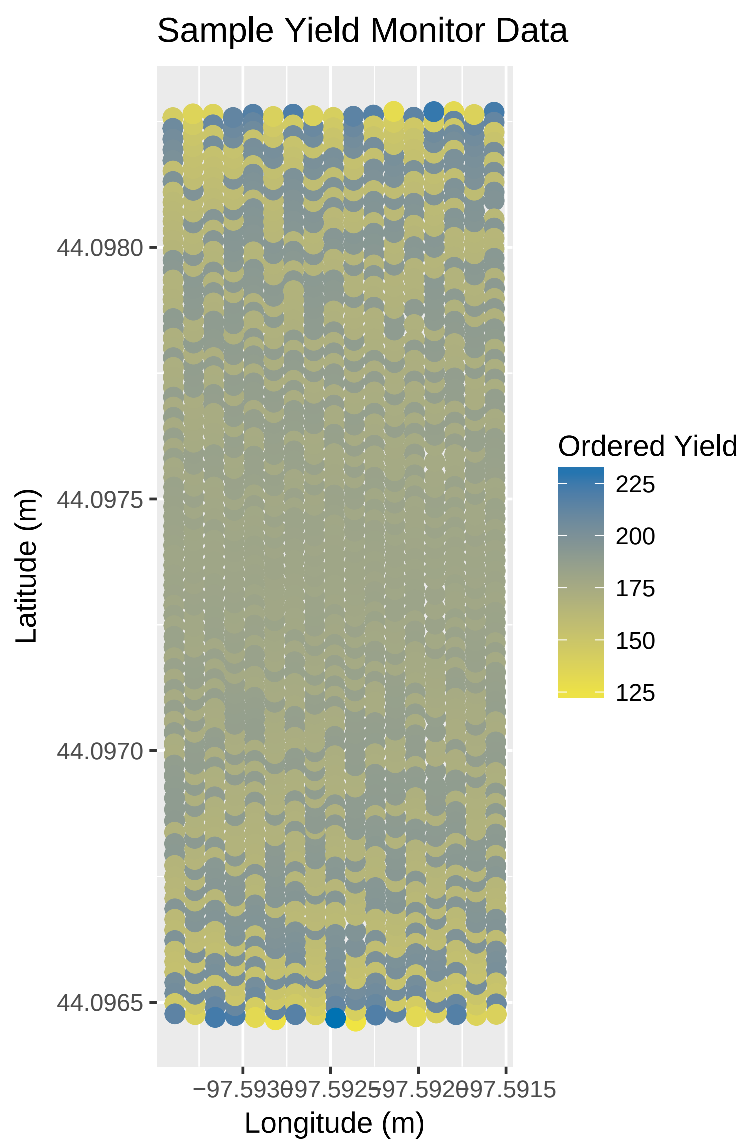
```
## [1]    1    2 1497    4 1495    6
```

```r
tail(skips)
```

```
## [1] 1492    7 1494    5 1496    3
```

```r
YieldDisordered <- YieldOrdered[skips]
sample.dat$YieldDisordered <- YieldDisordered[LatRank]
```

```r
ggplot(sample.dat, aes(Longitude, Latitude)) +
geom_point(aes(colour = YieldDisordered),size=3) +
scale_colour_gradient(low=cbPalette[7], high=cbPalette[5]) +
labs(colour = "Ordered Yield", x="Longitude (m)", y="Latitude (m)", title = "Sample Yield Monitor Data")
```



We create distance matrix using the **dist** function, then invert the distance to define our weight matrix w.

```r
sample.gps.dists <- as.matrix(dist(cbind(sample.dat$Longitude, sample.dat$Latitude)))
sample.gps.dists.inv <- 1/sample.gps.dists
```

```r
diag(sample.gps.dists.inv) <- 0

sample.gps.dists.inv[1:5, 1:5]
```

```
##          1        2        3        4        5
## 1     0.00 50313.97 25180.24 16801.92 12601.07
## 2 50313.97     0.00 50407.08 25225.88 16811.39
## 3 25180.24 50407.08     0.00 50496.47 25223.51
## 4 16801.92 25225.88 50496.47     0.00 50395.20
## 5 12601.07 16811.39 25223.51 50395.20     0.00
```

There are many implementations of Moran's I in R; for simplicity, we'll use the one from `ape`

Ordered

```r
Moran.I(sample.dat$YieldOrdered, sample.gps.dists.inv)
```

```
## $observed
## [1] 0.3659194
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.00146053
##
## $p.value
## [1] 0
```

As we might expect, I for the ordered yields is relatively large. To get a larger value, realisticly, we might consider a field, split in two, and planted with two very different varieties.

Checked

```r
Moran.I(sample.dat$YieldChecked, sample.gps.dists.inv)
```

```
## $observed
## [1] -0.02915056
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.001461746
##
## $p.value
## [1] 1.472015e-84
```

The checked example illustrates what is implied by a Moran's I value more negative than expected.

Disordered and Original

```
Moran.I(sample.dat$YieldDisordered, sample.gps.dists.inv)
```

```
## $observed
## [1] -0.00184221
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.00146053
##
## $p.value
## [1] 0.4215968
```

```
Moran.I(sample.dat$Yield, sample.gps.dists.inv)
```

```
## $observed
## [1] 0.1027251
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.00146053
##
## $p.value
## [1] 0
```

Remember that Moran's I is a global measure of correlation; the large homogeneity in the center of the field masks the checkerboard pattern at the ends. Also note that the actual yield data shows somewhat more homogeneity than this example, but not nearly as much as the uniform trend.

Cartesian coordinates.

Our discussion of local measures of spatial correlation will be easier if we can discuss 'local' in metric terms, instead of GPS coordinates.

```
sample.metric.dists <- as.matrix(dist(cbind(sample.dat$LonM, sample.dat$LatM)))
sample.metric.dists.inv <- 1/sample.metric.dists
diag(sample.metric.dists.inv) <- 0
sample.metric.dists.inv[1:5, 1:5]
```

```
##           1         2         3         4         5
## 1 0.0000000 0.4549639 0.2276923 0.1519313 0.1139451
## 2 0.4549639 0.0000000 0.4558059 0.2281050 0.1520169
## 3 0.2276923 0.4558059 0.0000000 0.4566142 0.2280835
## 4 0.1519313 0.2281050 0.4566142 0.0000000 0.4556985
## 5 0.1139451 0.1520169 0.2280835 0.4556985 0.0000000
```

Does this affect estimates?

```
Moran.I(sample.dat$Yield, sample.metric.dists.inv)
```

```
## $observed
```

```
## [1] 0.1026924
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.00145966
##
## $p.value
## [1] 0
```

Yes, there is a difference, about 0.1%. `dist` computes, by default, a Euclidean distance, $\sqrt{\Sigma(x_i - y_i)^2}$, which doesn't account for curvature found in longitude and latitude. It's small enough that we can continue to work with metric units. This will make correlagrams and neighborhoods easier to interpret.

Local Measures

We can rewrite our formula for Moran's I, by

$$I_i = \frac{N(y_i - \bar{y})}{\Sigma_i(y_i - \bar{y})^2} \Sigma_j w_{ij}(y_j - \bar{y})$$

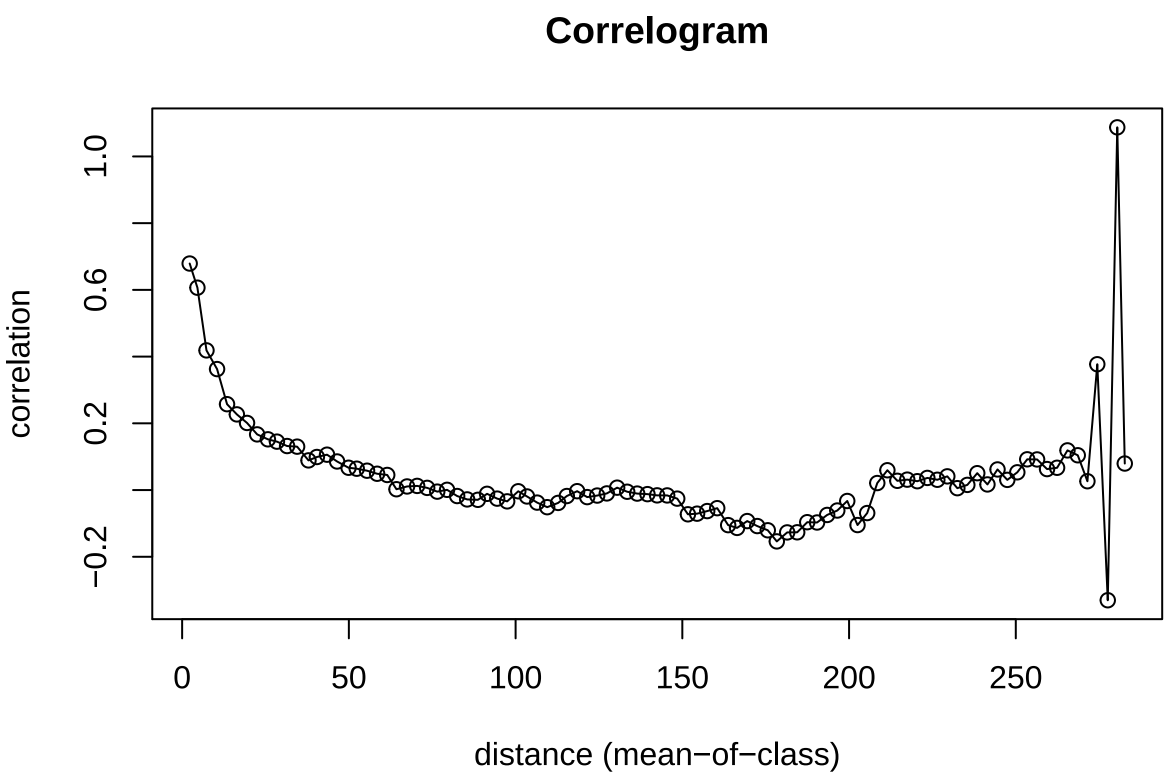
where I_i is a local measure of correlation, and

$$I = \Sigma_i \frac{I_i}{N}$$

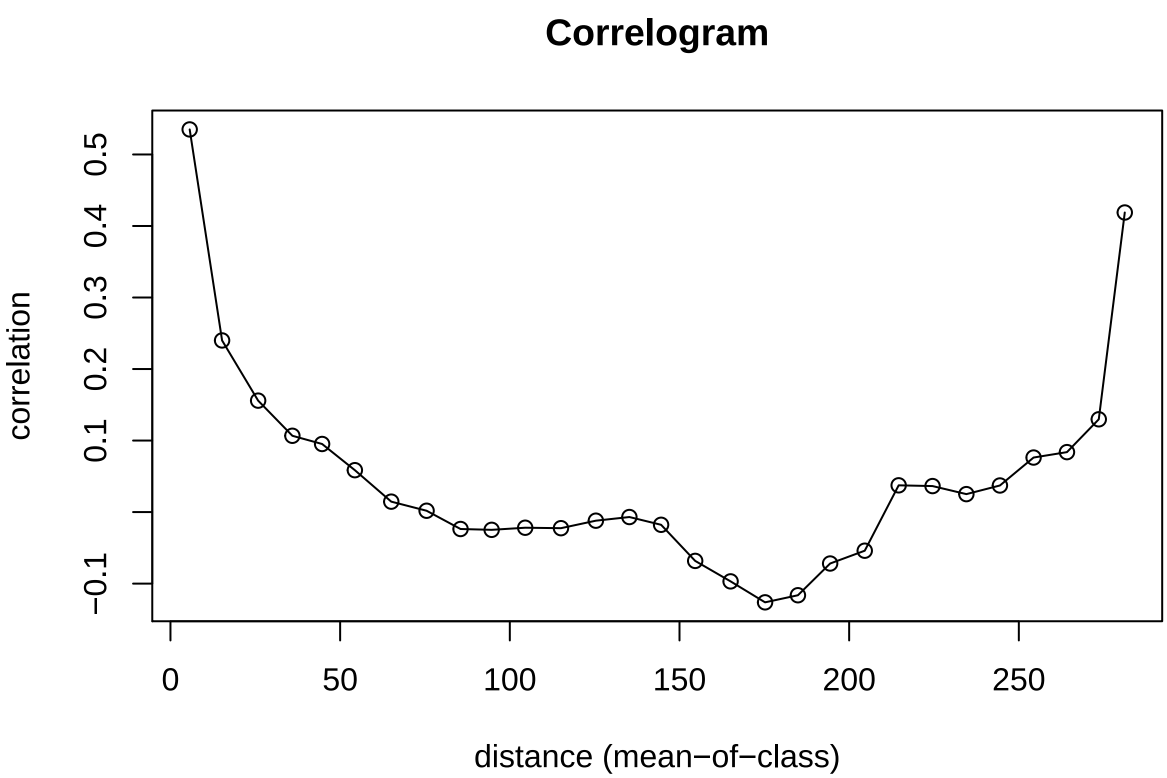
This is the approach used in computing the Local Indicator of Spatial Association (Anselin, 1995), sometimes simply referred to as Local Moran's I. To compute a LISA statistic, we should have a measure of neighborhood. To determine a range, we can compute a correlogram (not a variogram yet) based on Moran's I, using `correlog`.

`resamp` will give us a p-value for each point. We don't need this to determine a range.

```
Yield.3.clg <- correlog(sample.dat$LonM, sample.dat$LatM, sample.dat$Yield,
                        increment=3, resamp=0, quiet=TRUE)
plot(Yield.3.clg)
```




```
Yield.10.clg <- correlog(sample.dat$LonM, sample.dat$LatM, sample.dat$Yield,
                         increment=10, resamp=0, quiet=TRUE)
plot(Yield.10.clg)
```



The `lisa` function requires a explicit value for neighbor distance. By examining the correlogram, we consider

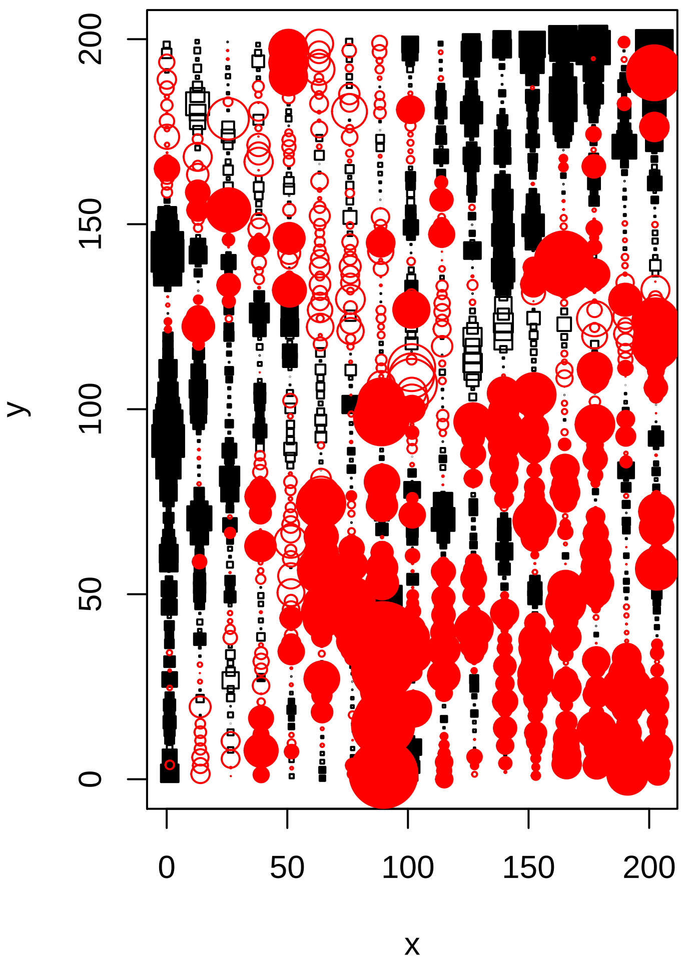
60 meters (correlation approaches 0 at this range).

One word of caution about correlograms (and variograms). There are typically very few points at large distances, so we have little to work with when estimating correlation. It's usually best to ignore the rightmost portion of a correlogram, unless the plots are specifically truncated.

```
Yield.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$Yield,
                   neigh=60, resamp=500, quiet=TRUE)
```

The returned values are `correlation` - Moran's I, and `mean.of.class`, the average of the distances within each distance class. We can visualize these by plotting the results.

```
plot.lisa(Yield.lisa, negh.mean=FALSE)
```

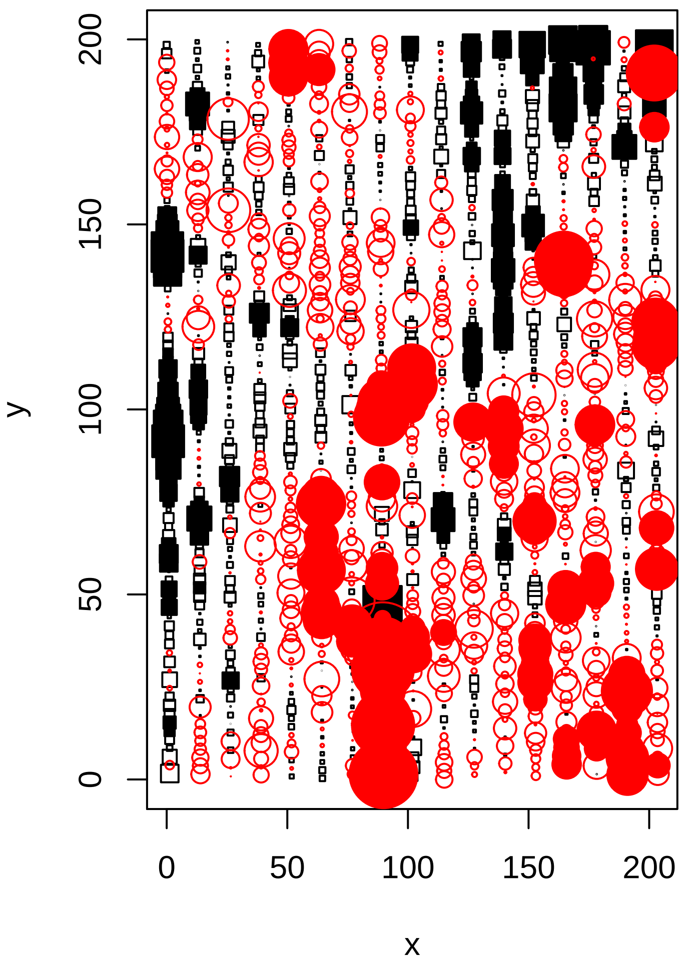


The LISA plot denotes positive deviations from the mean with circles and negative deviations as squares; points with significant local correlation are filled.

Let's recompute LISA, but with a smaller (6 m) neighborhood.

```
Yield.6.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$Yield,
                     neigh=6, resamp=500, quiet=TRUE)
```

```r
plot.lisa(Yield.6.lisa, negh.mean=FALSE)
```



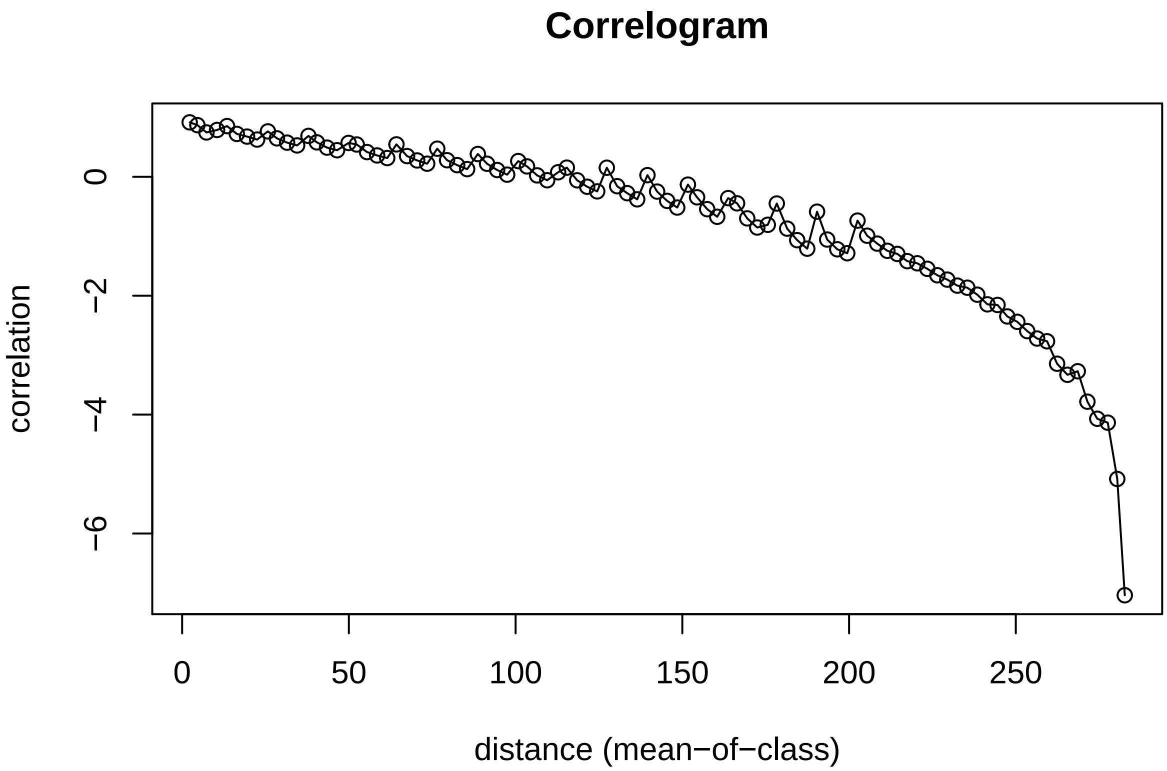
With a smaller neighborhood, we see fewer points identified as significantly correlated.

Simulated Yield Patterns

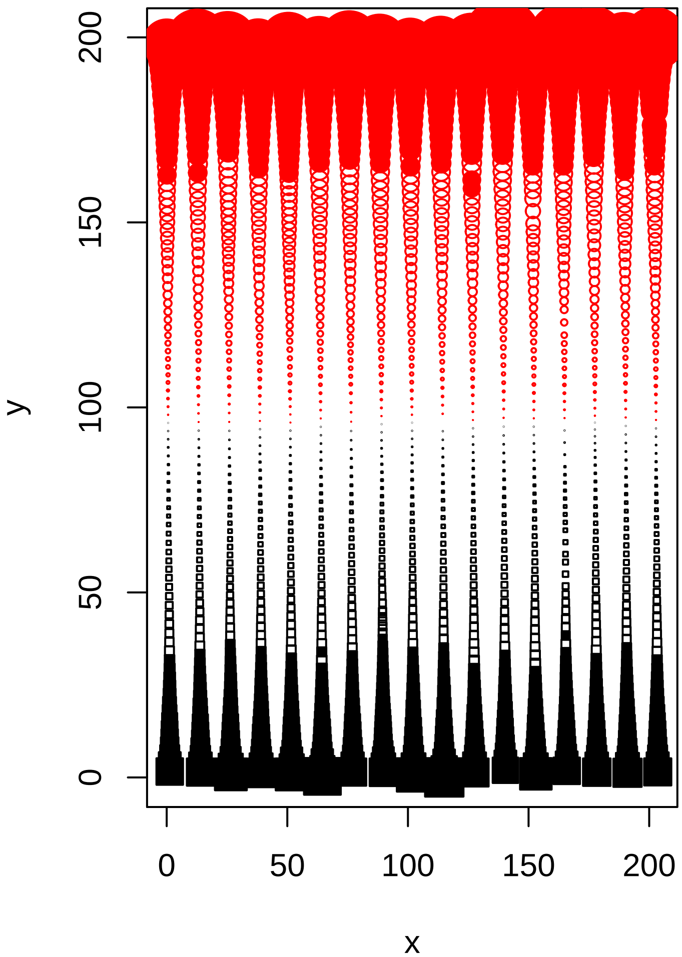
For comparison, our simulated data:

Ordered

```r
YieldOrdered.clg <- correlog(sample.dat$LonM, sample.dat$LatM, sample.dat$YieldOrdered,
                             increment=3, resamp=0, quiet=TRUE)
plot(YieldOrdered.clg)
```

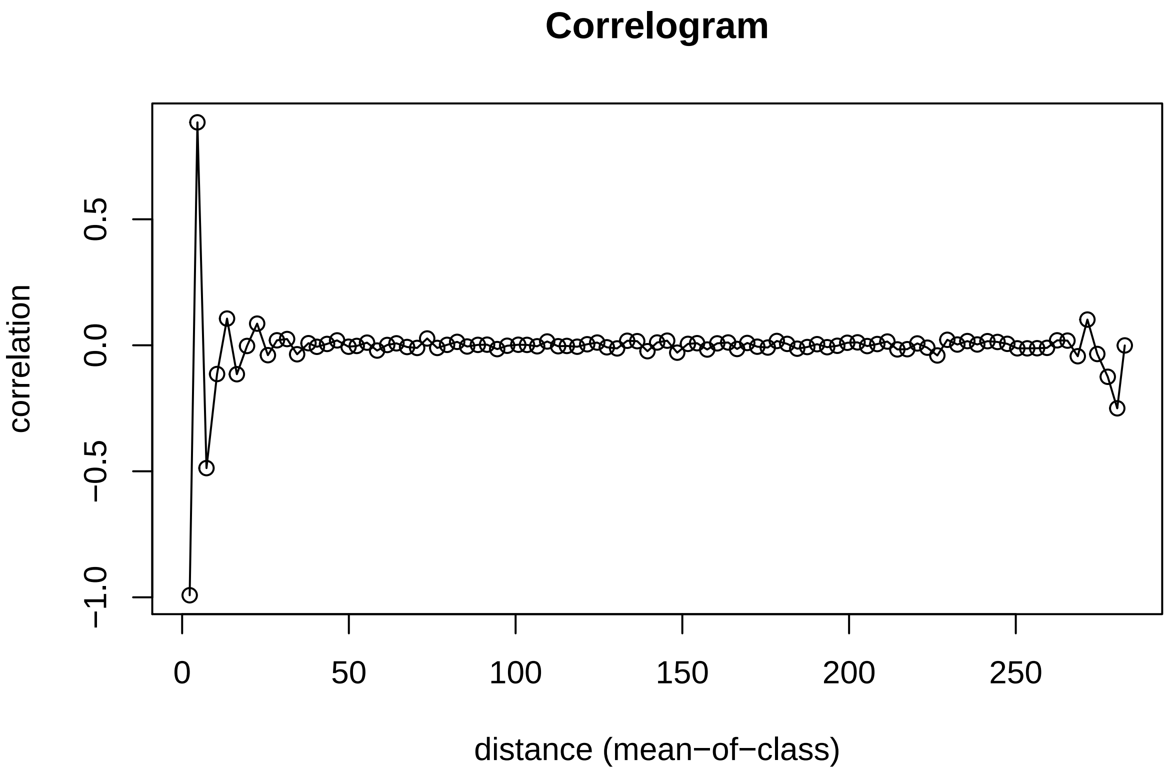



```
YieldOrdered.10.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$YieldOrdered,
                             neigh=10, resamp=500, quiet=TRUE)
plot.lisa(YieldOrdered.10.lisa, negh.mean=FALSE)
```

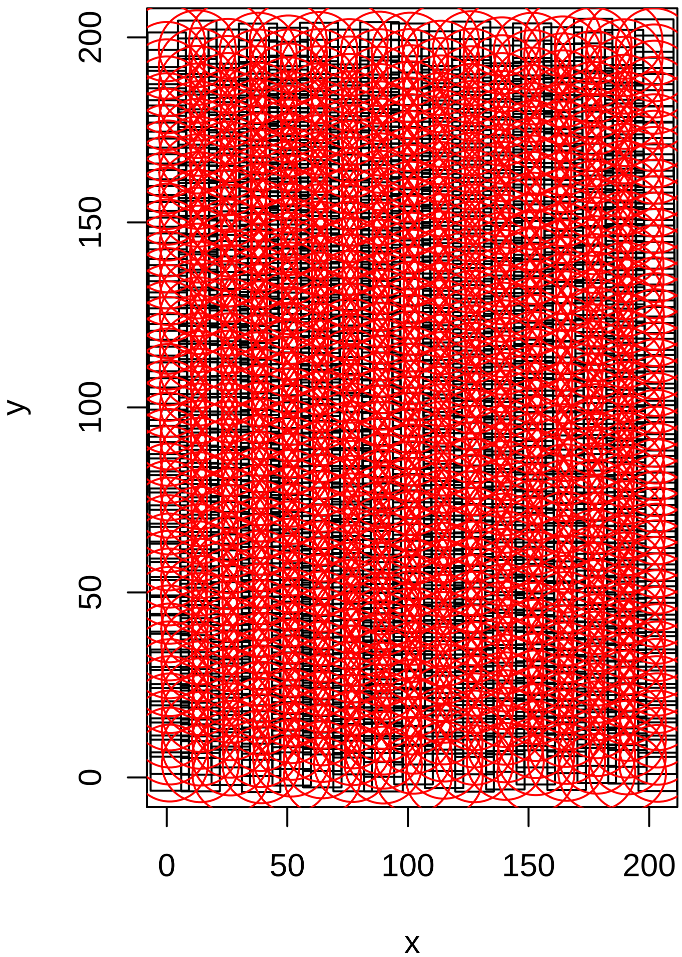


Checked

```
YieldChecked.clg <- correlog(sample.dat$LonM, sample.dat$LatM, sample.dat$YieldChecked,
                             increment=3, resamp=0, quiet=TRUE)
plot(YieldChecked.clg)
```

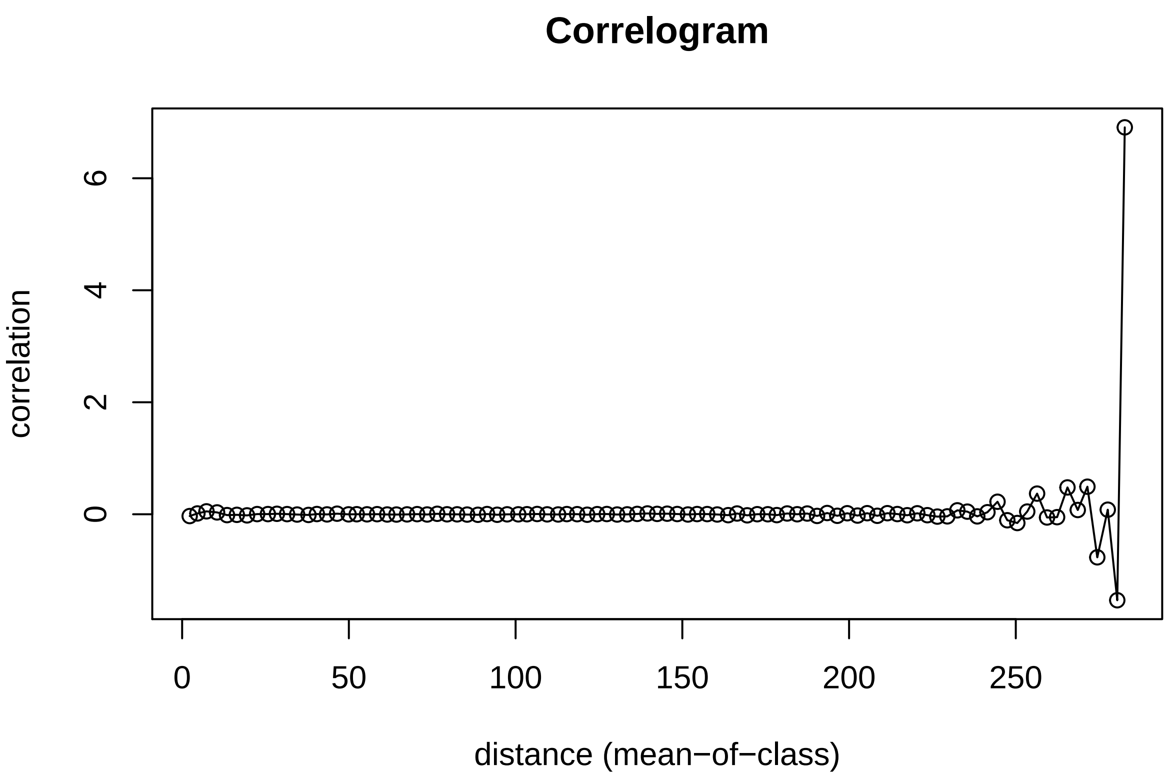



```
YieldChecked.10.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$YieldChecked,
                                  neigh=10, resamp=500, quiet=TRUE)
plot.lisa(YieldChecked.10.lisa, negh.mean=FALSE)
```

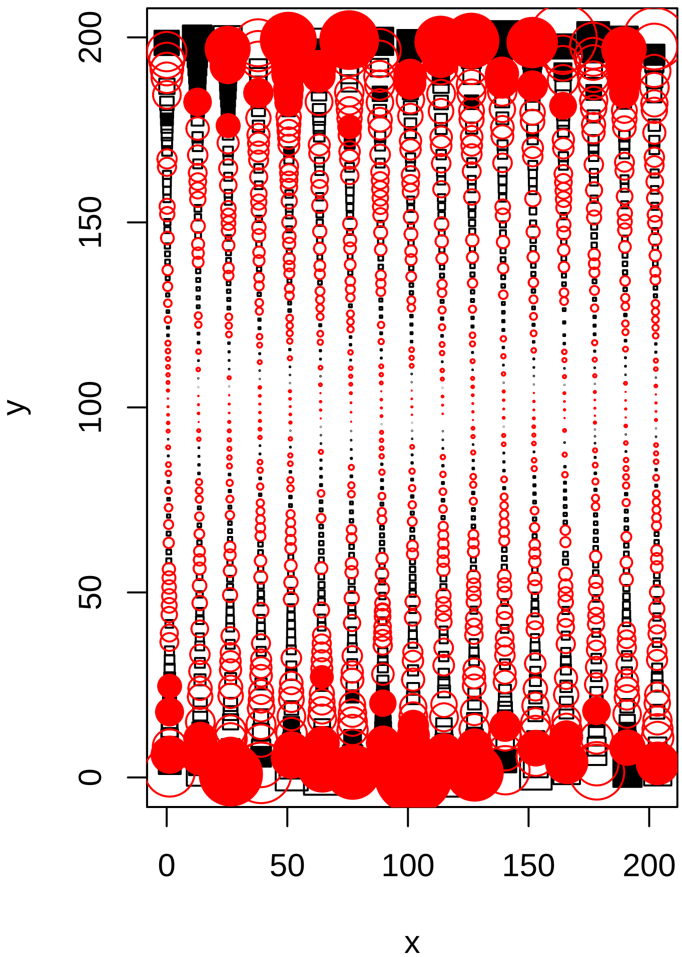


Disordered

```
YieldDisordered.clg <- correlog(sample.dat$LonM, sample.dat$LatM, sample.dat$YieldDisordered,
                                increment=3, resamp=0, quiet=TRUE)
plot(YieldDisordered.clg)
```




```
YieldDisordered.10.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$YieldDisordered,
                                    neigh=10, resamp=500, quiet=TRUE)
plot.lisa(YieldDisordered.10.lisa, negh.mean=FALSE)
```



Other Observations

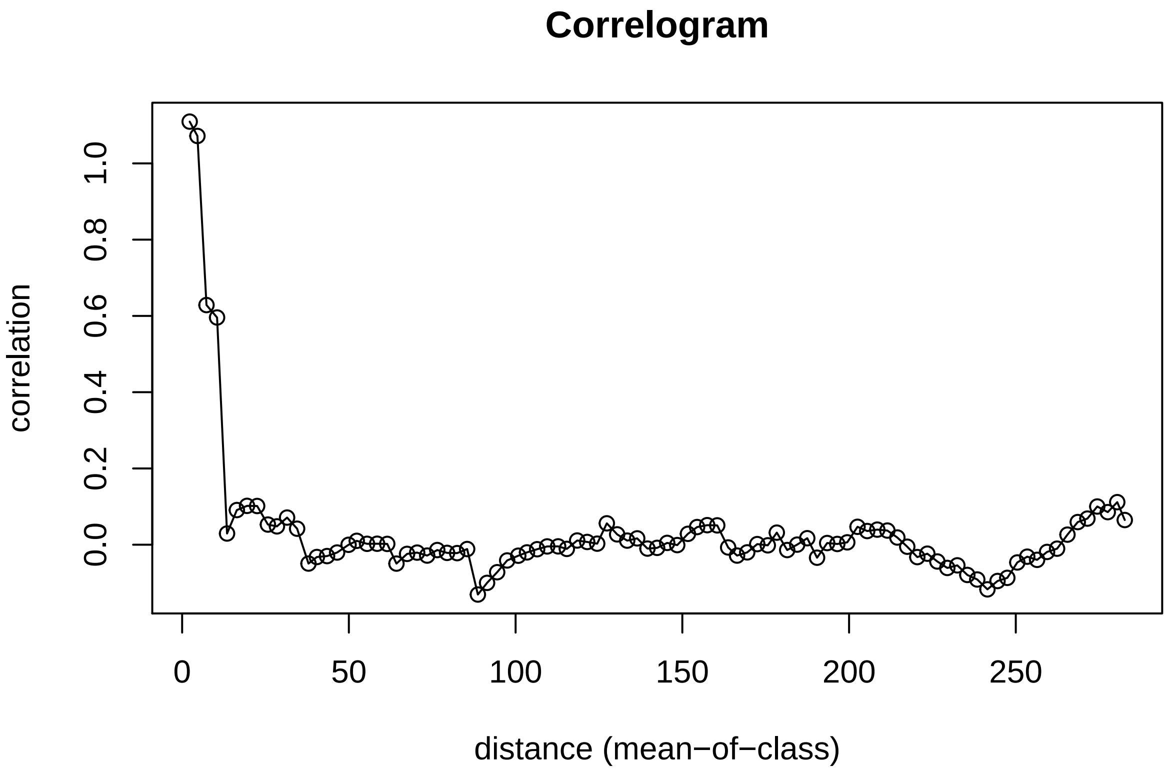
For comparison, we compute global and local measures of spatial correlation for other monitor variables.

Distance

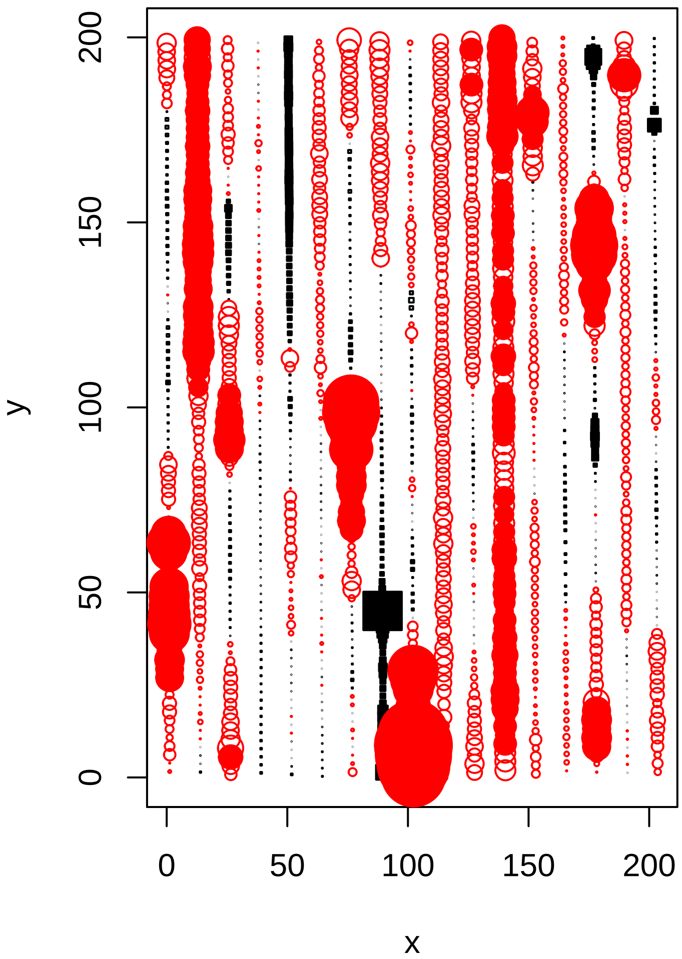
```
Moran.I(sample.dat$Distance, sample.metric.dists.inv)
```

```
## $observed
## [1] 0.1207993
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.001439374
##
## $p.value
## [1] 0
```

```
Distance.metric.clg <- correlog(sample.dat$LonM, sample.dat$LatM, sample.dat$Distance,
                                increment=3, resamp=0, quiet=TRUE)
plot(Distance.metric.clg)
```




```
Distance.10.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$Distance,
                         neigh=10, resamp=500, quiet=TRUE)
plot.lisa(Distance.10.lisa, negh.mean=FALSE)
```

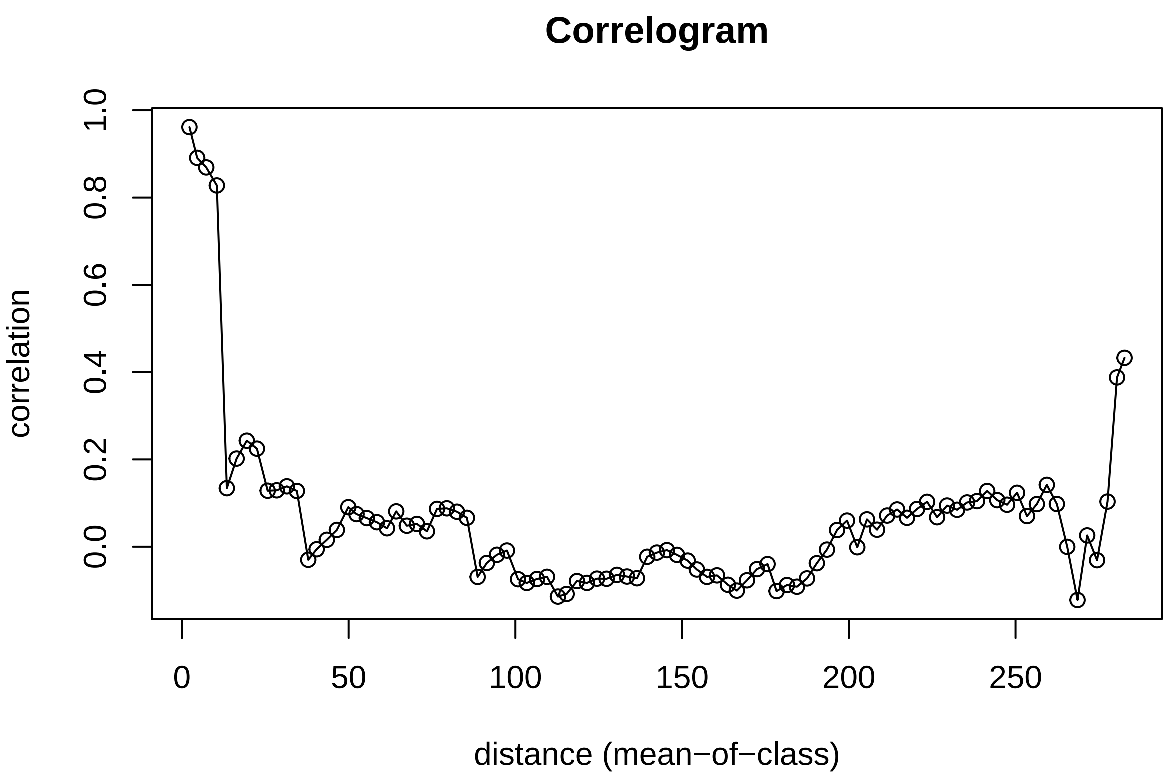


Moisture

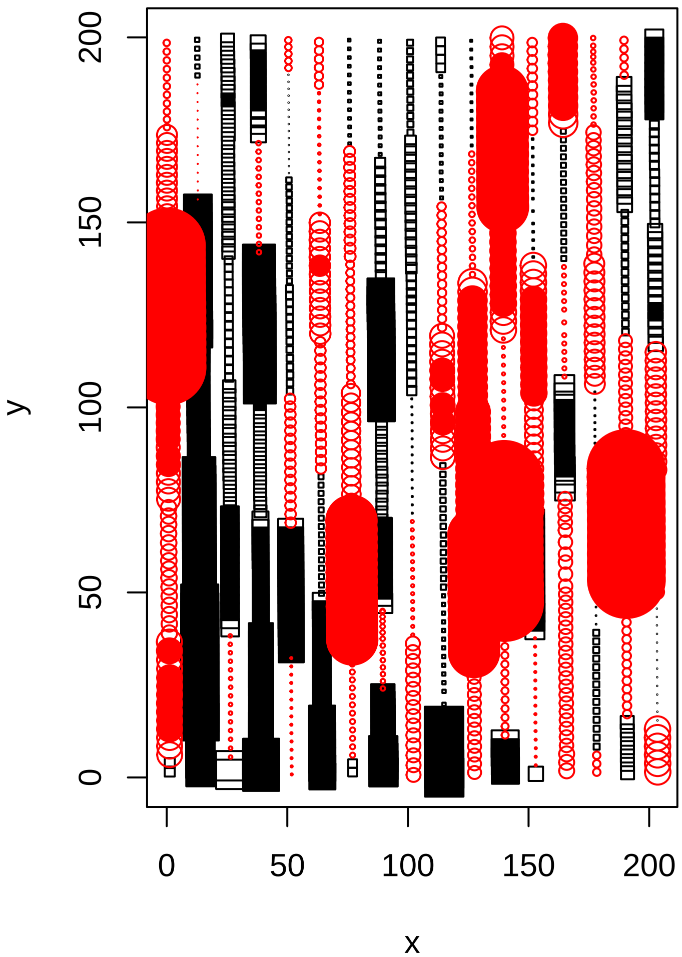
```r
Moran.I(sample.dat$Moisture, sample.metric.dists.inv)
```

```
## $observed
## [1] 0.1169183
##
## $expected
## [1] -0.0006684492
##
## $sd
## [1] 0.001459387
##
## $p.value
## [1] 0
```

```r
Moisture.metric.clg <- correlog(sample.dat$LonM, sample.dat$LatM, sample.dat$Moisture,
                                increment=3, resamp=0, quiet=TRUE)
plot(Moisture.metric.clg)
```




```
Moisture.10.lisa <- lisa(sample.dat$LonM, sample.dat$LatM, sample.dat$Moisture,
                         neigh=10, resamp=500, quiet=TRUE)
plot.lisa(Moisture.10.lisa, negh.mean=FALSE)
```



Other Measures of Spatial Correlation

Geary's C

$$C = \frac{N-1}{\Sigma_i 2\Sigma_j w_{ij}} \frac{\Sigma_i \Sigma_j w_{ij}(y_i - y_j)^2}{\Sigma_i (y_i - \bar{y})^2}$$

Geary's Contiguity Ratio lies between 0 and somewhat larger than 1; values > 1 suggest positiv spatial correlation, values < 1 suggest negative spatial correlation. In general, C is more senstive to local correlation, while I is a better measure of global correlation.

Getis-Ord G.

$$G = \frac{\Sigma_i \Sigma_j w_{ij} \times y_i \times y_j}{\Sigma_i \Sigma_j y_i \times y_j}$$

Getic-Ord G is typically interpret to represent clustering; unusually high or low values grouped together.

spdep

The `spdep` library includes functions for many measures of spatial correlation, including I, C and G. We can use the set of neighbor weights, as before, but we need to convert

```
sample.metric.dists.inv.listw <-mat2listw(sample.metric.dists.inv)
```

The global G test will complain if we use continuous weights; we can create a set of neighborhood weights by

```
neighborhood30 <- dnearneigh(cbind(sample.dat$LonM, sample.dat$LatM), 0, 30)
neighborhood30.listw <- nb2listw(neighborhood30, style="B")
```

One thing to note that these tests default to `alternative="greater"`, unlike `Moran.I`, which defaults to `alternative = "two.sided"`

Ordered

```
moran.test(sample.dat$YieldOrdered,sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
##
##  Moran I test under randomisation
##
## data:  sample.dat$YieldOrdered
## weights: sample.metric.dists.inv.listw
##
## Moran I statistic standard deviate = 230.35, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Moran I statistic       Expectation          Variance
##      3.324262e-01     -6.684492e-04      2.090950e-06
```

```
geary.test(sample.dat$YieldOrdered,sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
##
##  Geary C test under randomisation
##
## data:  sample.dat$YieldOrdered
## weights: sample.metric.dists.inv.listw
##
## Geary C statistic standard deviate = 85.383, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Geary C statistic       Expectation          Variance
##      5.467159e-01      1.000000e+00      2.818356e-05
```

```
globalG.test(sample.dat$YieldOrdered,neighborhood30.listw,alternative = "two.sided")
```

```
##
##  Getis-Ord global G statistic
##
## data:  sample.dat$YieldOrdered
## weights: neighborhood30.listw
##
## standard deviate = 7.0722, p-value = 1.524e-12
## alternative hypothesis: two.sided
## sample estimates:
```

```
## Global G statistic        Expectation          Variance
##       6.023916e-02       5.989162e-02       2.414889e-09
```

Checkerboard

```r
moran.test(sample.dat$YieldChecked,sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
##
##  Moran I test under randomisation
##
## data:  sample.dat$YieldChecked
## weights: sample.metric.dists.inv.listw
##
## Moran I statistic standard deviate = -19.435, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Moran I statistic       Expectation          Variance
##     -2.879501e-02     -6.684492e-04      2.094394e-06
```

```r
geary.test(sample.dat$YieldChecked,sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
##
##  Geary C test under randomisation
##
## data:  sample.dat$YieldChecked
## weights: sample.metric.dists.inv.listw
##
## Geary C statistic standard deviate = -19.435, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Geary C statistic       Expectation          Variance
##      1.028108e+00      1.000000e+00      2.091616e-06
```

```r
globalG.test(sample.dat$YieldChecked,neighborhood30.listw,alternative = "two.sided")
```

```
##
##  Getis-Ord global G statistic
##
## data:  sample.dat$YieldChecked
## weights: neighborhood30.listw
##
## standard deviate = -0.018747, p-value = 0.985
## alternative hypothesis: two.sided
## sample estimates:
## Global G statistic        Expectation          Variance
##       5.989070e-02       5.989162e-02       2.416713e-09
```

Disordered

```r
moran.test(sample.dat$YieldDisordered,sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
##
##  Moran I test under randomisation
```

```
##
## data:  sample.dat$YieldDisordered
## weights: sample.metric.dists.inv.listw
##
## Moran I statistic standard deviate = -0.68816, p-value = 0.4914
## alternative hypothesis: two.sided
## sample estimates:
## Moran I statistic       Expectation          Variance
##     -1.663534e-03     -6.684492e-04      2.090950e-06
```

```r
geary.test(sample.dat$YieldDisordered,sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
##
##  Geary C test under randomisation
##
## data:  sample.dat$YieldDisordered
## weights: sample.metric.dists.inv.listw
##
## Geary C statistic standard deviate = 22.486, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Geary C statistic       Expectation          Variance
##      8.806249e-01      1.000000e+00      2.818356e-05
```

```r
globalG.test(sample.dat$YieldDisordered,neighborhood30.listw,alternative = "two.sided")
```

```
##
##  Getis-Ord global G statistic
##
## data:  sample.dat$YieldDisordered
## weights: neighborhood30.listw
##
## standard deviate = 1.8704, p-value = 0.06143
## alternative hypothesis: two.sided
## sample estimates:
## Global G statistic        Expectation           Variance
##       5.998353e-02       5.989162e-02       2.414889e-09
```

Yield

```r
moran.test(sample.dat$Yield, sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
##
##  Moran I test under randomisation
##
## data:  sample.dat$Yield
## weights: sample.metric.dists.inv.listw
##
## Moran I statistic standard deviate = 68.161, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Moran I statistic       Expectation          Variance
##      9.789362e-02     -6.684492e-04      2.090950e-06
```

```r
geary.test(sample.dat$Yield, sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
##
##  Geary C test under randomisation
##
## data:  sample.dat$Yield
## weights: sample.metric.dists.inv.listw
##
## Geary C statistic standard deviate = 21.301, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Geary C statistic       Expectation          Variance
##      8.869159e-01      1.000000e+00      2.818356e-05
```

```r
globalG.test(sample.dat$Yield, sample.metric.dists.inv.listw,alternative = "two.sided")
```

```
## Warning in globalG.test(sample.dat$Yield, sample.metric.dists.inv.listw, :
## Binary weights recommended (sepecially for distance bands)
```

```
##
##  Getis-Ord global G statistic
##
## data:  sample.dat$Yield
## weights: sample.metric.dists.inv.listw
##
## standard deviate = 5.7141, p-value = 1.103e-08
## alternative hypothesis: two.sided
## sample estimates:
## Global G statistic        Expectation           Variance
##       1.428321e-02       1.424027e-02       5.645810e-11
```

```r
globalG.test(sample.dat$Yield, neighborhood30.listw,alternative = "two.sided")
```

```
##
##  Getis-Ord global G statistic
##
## data:  sample.dat$Yield
## weights: neighborhood30.listw
##
## standard deviate = 7.3097, p-value = 2.677e-13
## alternative hypothesis: two.sided
## sample estimates:
## Global G statistic        Expectation           Variance
##       6.025083e-02       5.989162e-02       2.414889e-09
```