

SISMID 2021 R Notes: Mapping for Point Data

Jon Wakefield
University of Washington

2021-07-11

Overview

In these notes we will consider mapping and modeling of point data in which the (nominal) exact locations are known.

We will look at

- continuous responses via mixed (geostatistical) models,

Continuous responses: example

We illustrate methods for continuous data using on Zinc levels in the Netherlands.

This data set gives locations and top soil heavy metal concentrations (in ppm), along with a number of soil and landscape variables, collected in a flood plain of the river Meuse, near the village Stein in the South of the Netherlands.

Heavy metal concentrations are bulk sampled from an area of approximately 28km × 39km.

The Meuse data are in a variety of packages. The version in the geoR library are not a spatial object, but can be used with likelihood and Bayes methods.

geoR for geostatistics

We start the analysis using functions from the geoR library, for which a geodata data type is required.

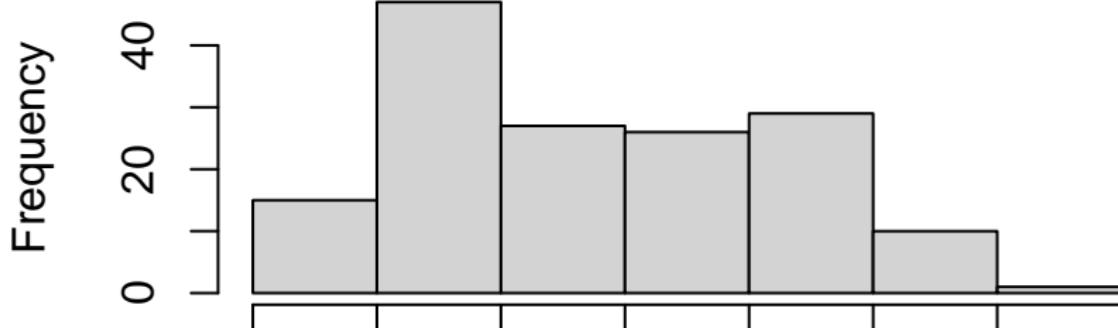
```
library(geoR)
library(sp)
data("meuse")
zmat <- matrix(cbind(meuse$x, meuse$y, log(meuse$zinc)),
               ncol = 3, nrow = 155, byrow = F)
geozinc <- as.geodata(zmat, coords.col = c(1,
                                             2), data.col = c(3))
```

There are 155 observations (sampling locations)

Zinc data

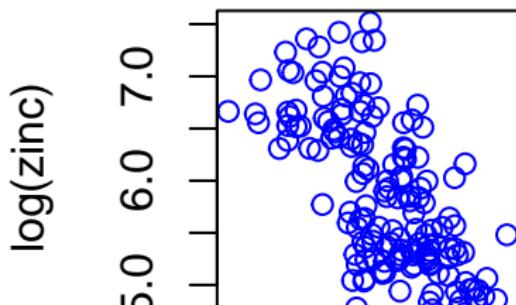
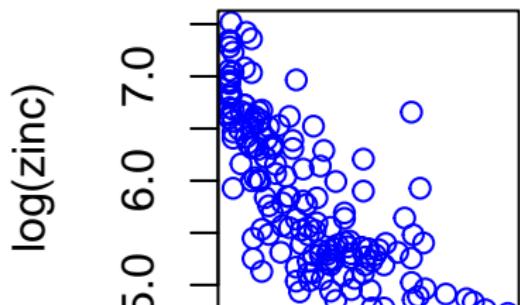
We work with $\log(\text{zinc})$ as the distribution is more symmetric than on the original scale, and the variance more constant across levels of covariates.

```
hist(log(meuse$zinc), main = "", xlab = "log(zinc)")
```



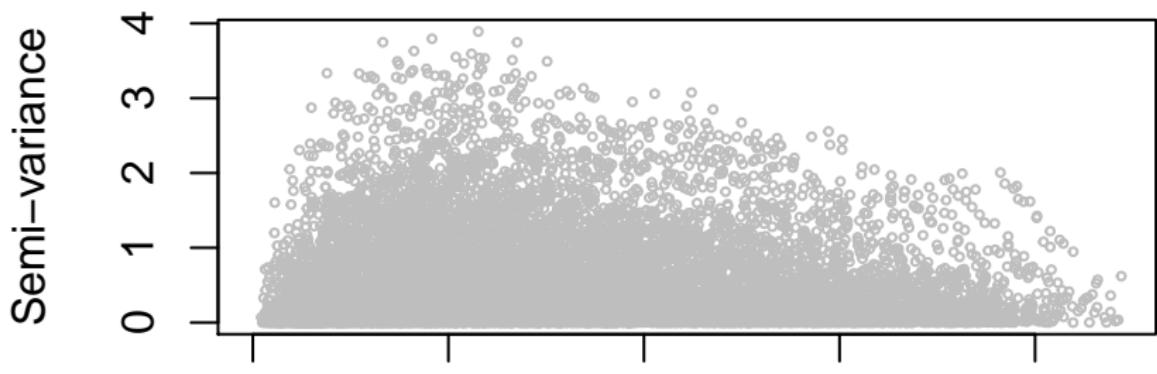
$\log(\text{zinc})$ versus distance from river and elevation

```
par(mfrow = c(1, 2))
plot(log(meuse$zinc) ~ meuse$dist, ylab = "log(zinc)",
     xlab = "Scaled distance", col = "blue")
plot(log(meuse$zinc) ~ meuse$elev, ylab = "log(zinc)",
     xlab = "Elevation", col = "blue")
```



$\log(\text{zinc})$: Variogram cloud, constant mean

```
cloudzinc <- variog(geozinc, option = "cloud")
plot(cloudzinc, ylab = "Semi-variance", xlab = "Distance (m)",
     col = "grey", cex = 0.4)
```

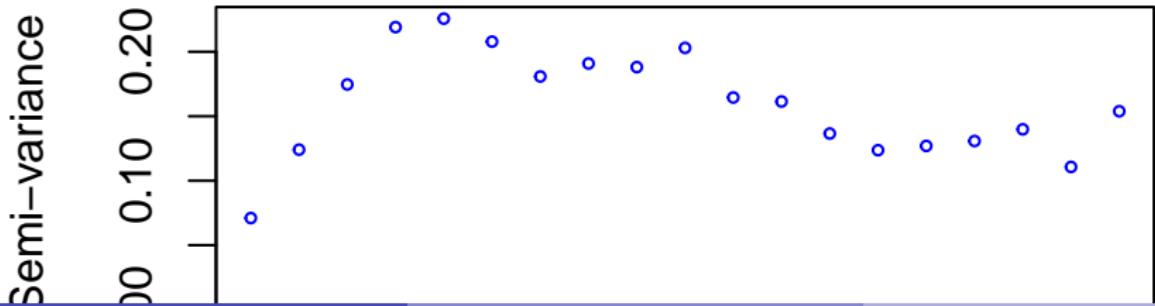


`log(zinc)`: Binned variogram with linear trend in distance and elevation

```

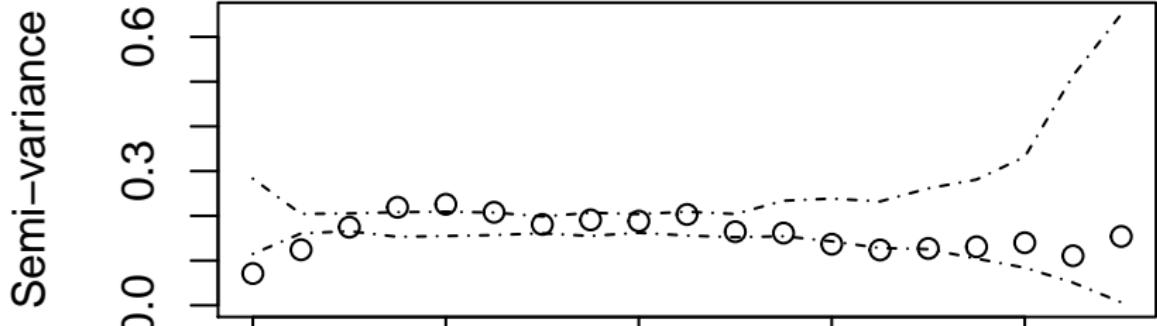
binzinc <- variog(geozinc, uvec = seq(0,
    5000, 250), trend = ~meuse$dist + meuse$elev)
plot(binzinc, ylab = "Semi-variance", xlab = "Distance (m)",
    cex = 0.5, col = "blue")

```



$\log(\text{zinc})$: Monte Carlo envelopes under no spatial dependence: clear there is dependence here

```
geozinc.env <- variog.mc.env(geozinc, obj = binzinc)
plot(binzinc, env = geozinc.env, xlab = "Distance (m)",
     ylab = "Semi-variance")
```



Parameter estimation from the variogram

We now estimate the parameters of the exponential covariance model which in geoR is parameterized as

$$\tau^2 + \sigma^2 \exp(-d/\phi),$$

where d is the distance between the points, σ^2 is the partial sill and τ^2 is the nugget.

The effective range is the distance at which the correlation is 0.05, and if we have a rough estimate of this \tilde{d} (from the binned variogram, for example) then we can solve for an initial estimate $\tilde{\phi} = -\tilde{d}/\log(0.05)$.

Rather than specify a distribution for the data, we can initially estimate the parameters from the sample (binned) variogram.

We illustrate this using OLS and WLS.

Parameter estimation from the binned variogram

We have a non-linear least squares problem, we give initial estimates for σ^2 and ϕ . From the binned variogram, estimate $\tilde{d} = 800$ to give $\tilde{\phi} = 267$.

```
olsfit <- variofit(binzinc, ini = c(0.2, 267), weights = "equal")
## variofit: covariance model used is matern
## variofit: weights used: equal
## variofit: minimisation function used: optim
olsfit
## variofit: model parameters estimated by OLS (ordinary least squares):
## covariance model is: matern with fixed kappa = 0.5 (exponential)
## parameter estimates:
##      tausq    sigmasq      phi
## 0.0682 0.0982 172.7783
## Practical Range with cor=0.05 for asymptotic range: 517.5976
##
## variofit: minimised sum of squares = 0.0209
```

Parameter estimation from the variogram: WLS

The default is to weight by the number of pairs in each bin, though other options are available.

```
wlsfit <- variofit(binzinc, ini = c(0.2, 500))
## variofit: covariance model used is matern
## variofit: weights used: npairs
## variofit: minimisation function used: optim
wlsfit
## variofit: model parameters estimated by WLS (weighted least squares):
## covariance model is: matern with fixed kappa = 0.5 (exponential)
## parameter estimates:
##      tausq    sigmasq      phi
## 0.1350  0.0546 499.9999
## Practical Range with cor=0.05 for asymptotic range: 1497.866
##
## variofit: minimised weighted sum of squares = 12.6632
```

Maximum likelihood for log(zinc)

We suppress the output from the call.

```
mlfit <- likfit(geozinc, ini = c(0.2, 224), trend = ~meuse$dist +
  meuse$elev)
```

The results: estimates of ϕ and τ^2 are quite different from the LS versions.

```
mlfit$parameters.summary
##           status   values
## beta0    estimated  8.6162
## beta1    estimated -2.1072
## beta2    estimated -0.2690
## tausq   estimated  0.0010
## sigmasq estimated  0.2065
## phi      estimated 241.1982
## kappa    fixed     0.5000
## psiA     fixed     0.0000
## psiR     fixed     1.0000
## lambda   fixed     1.0000
mlfit$practicalRange
## [1] 722.5653
```

Restricted maximum likelihood for log(zinc)

```
remlfit <- likfit(geozinc, ini = c(0.55,  
 224), lik.method = "RML", trend = ~meuse$dist +  
 meuse$elev)
```

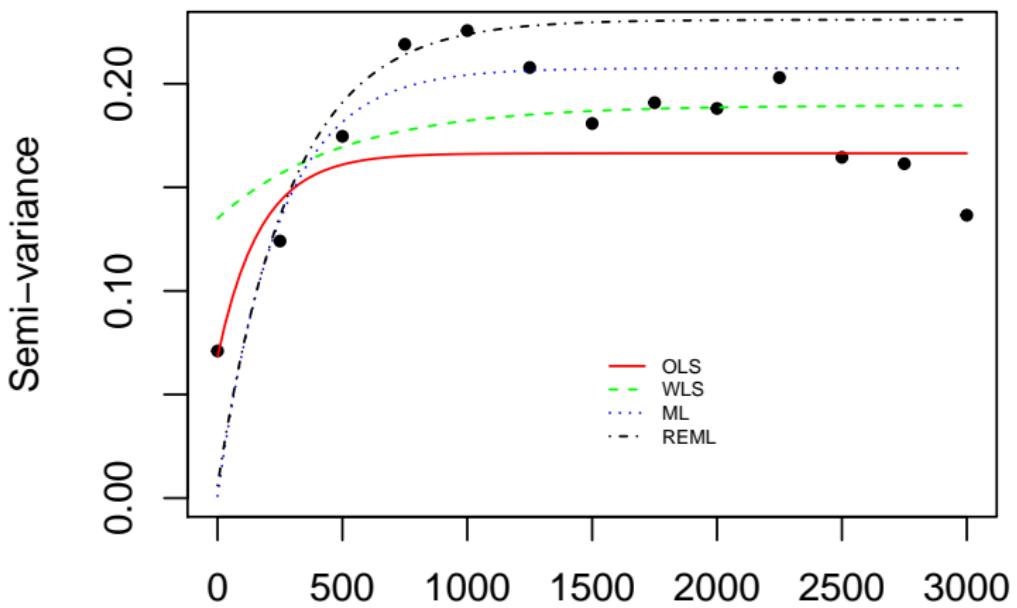
The results: slight differences from ML.

```
remlfit$parameters.summary  
##           status   values  
## beta0    estimated  8.6396  
## beta1    estimated -2.1215  
## beta2    estimated -0.2701  
## tausq    estimated  0.0061  
## sigmasq  estimated  0.2248  
## phi      estimated 289.1468  
## kappa    fixed     0.5000  
## psiA     fixed     0.0000  
## psiR     fixed     1.0000  
## lambda   fixed     1.0000  
remlfit$practicalRange  
## [1] 866.2064
```

Comparison of estimates

```
plot(binzinc, max.dist = 3000, xlab = "Distance (m)",  
      ylab = "Semi-variance", pch = 19, cex = 0.6)  
lines(olsfit, max.dist = 3000, col = "red")  
lines(wlsfit, max.dist = 3000, lty = 2, col = "green")  
lines(mlfit, max.dist = 3000, lty = 3, col = "blue")  
lines(remlfit, max.dist = 3000, lty = 4, col = "black")  
legend("bottomright", legend = c("OLS", "WLS", "ML",  
      "REML"), lty = c(1, 2, 3, 4), bty = "n", col = c("red",  
      "green", "blue", "black"), cex = 0.5)
```

Comparison of estimates



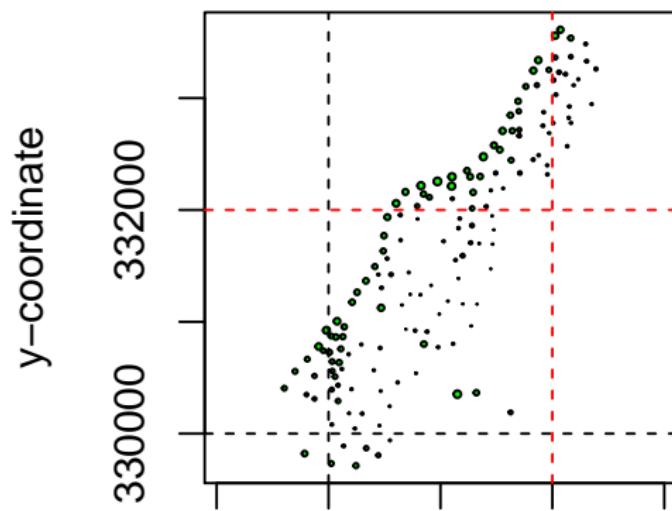
Prediction for log(zinc)

We plot the data along with the region within which we shall carry out prediction.

```
points(geozinc, pt.divide = "data.proportional", cex.min = 0.05,
       cex.max = 0.4, xlab = "x-coordinate", ylab = "y-coordinate",
       col = "green")
# See points.geodata description for explanation
# of this function
abline(h = 330000, lty = 2)
abline(h = 332000, lty = 2, col = "red")
abline(v = 179000, lty = 2)
abline(v = 181000, lty = 2, col = "red")
```

Prediction for $\log(\text{zinc})$

We plot the data along with the region within which we shall carry out prediction.



Prediction for log(zinc)

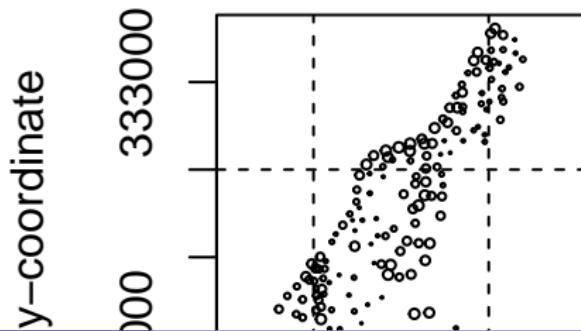
We fit a linear model in distance and elevation to log(zinc).

We then form a geodata object with the residuals as the response.

```
lmfit <- lm(geozinc$data ~ meuse$dist + meuse$elev)
lmfit
##
## Call:
## lm(formula = geozinc$data ~ meuse$dist + meuse$elev)
##
## Coefficients:
## (Intercept)    meuse$dist    meuse$elev
##          8.4845      -1.9600       -0.2607
detrend <- as.geodata(cbind(geozinc$coords, lmfit$residuals))
```

Prediction for log(zinc)

```
points(detrend, pt.divide = "rank.prop", xlab = "x-coordinate",
       ylab = "y-coordinate", cex.min = 0.1, cex.max = 0.5)
abline(h = 330000, lty = 2)
abline(h = 332000, lty = 2)
abline(v = 179000, lty = 2)
abline(v = 181000, lty = 2)
```



Prediction for log(zinc)

Carry out MLE on the detrended data.

```
mlfit2 <- likfit(detrend, ini = c(0.2, 224))
## -----
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##         arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## -----
## likfit: end of numerical maximisation.
mlfit2
## likfit: estimated model parameters:
##       beta      tausq    sigmasq      phi
## " 0.0306" " 0.0000" " 0.2076" "238.0914"
## Practical Range with cor=0.05 for asymptotic range: 713.2582
##
## likfit: maximised log-likelihood = -54.82
```

Prediction for log(zinc)

We now obtain spatial predictions on a grid, using the parameter estimates from the ML fit to the residuals.

Ordinary Kriging is used for prediction.

```
pred.grid <- expand.grid(seq(179000, 181000,  
    l = 51), seq(330000, 332000, l = 51))  
kc <- krige.conv(detrend, loc = pred.grid,  
    krige = krige.control(obj.m = mlfit2))  
## krige.conv: model with constant mean  
## krige.conv: Kriging performed using global neighbourhood
```

Prediction for log(zinc)

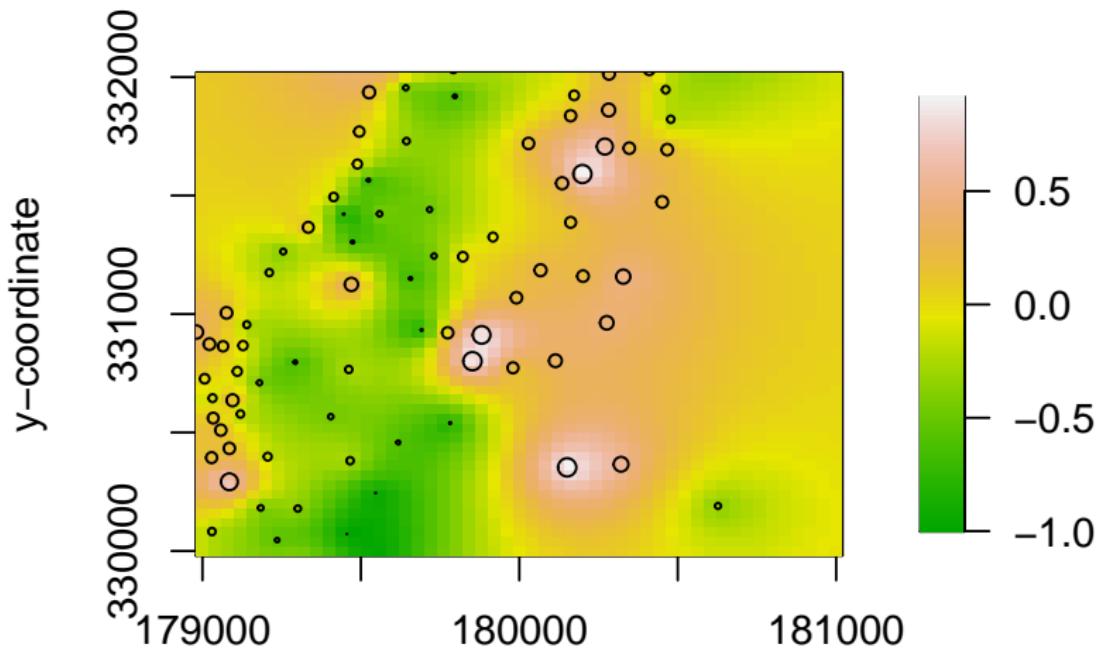
Produce an image plot of the predictions, with the data superimposed.

```
library(fields)

image.plot(x = pred.grid[["Var1"]][1:51], y = unique(pred.grid[["Var1"]])
           z = matrix(kc$predict, nrow = 51, ncol = 51), col = terrain.colors(51),
           xlab = "x-coordinate", ylab = "y-coordinate")

symbols(detrend$coords[, 1], detrend$coords[, 2], circles = (detrend$data -
  min(detrend$data))/1, add = T, inches = 0.04)
```

Prediction for $\log(\text{zinc})$



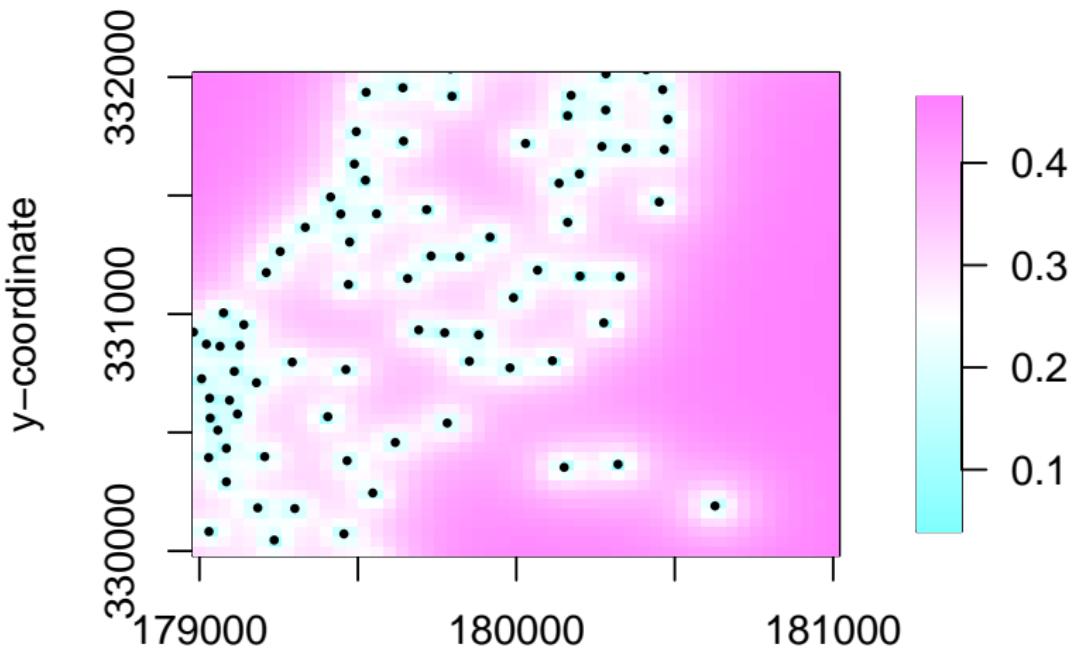
Standard deviations of prediction for log(zinc)

We now plot the Kriging standard deviations of the predictions.

```
image.plot(x = pred.grid[["Var1"]][1:51], y = unique(pred.grid[["Var1"]]),  
           z = matrix(sqrt(kc$krige.var), nrow = 51, ncol = 51),  
           col = cm.colors(100), xlab = "x-coordinate", ylab = "y-coordinate")  
  
points(detrend$coords[, 1], detrend$coords[, 2], pch = 16)
```

The standard deviation is smallest close to the datapoints, as expected.

Standard deviations of prediction for $\log(\text{zinc})$



log(zinc) modeled with a GAM

We now model the log(zinc) surface as linear in distance and elevation, and with the spatial surface modeled with a thin plate regression spline, with the smoothing parameter estimated using REML.

```
library(mgcv)
library(lattice)
library(latticeExtra)
library(RColorBrewer)

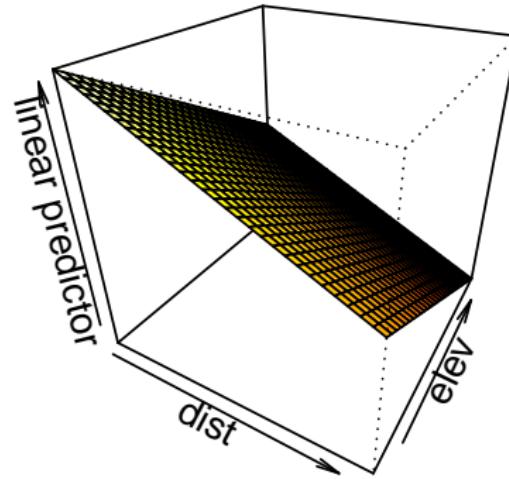
zinc.dat <- data.frame(x = meuse$x, y = meuse$y, lzinc = log(meuse$z))
  dist = meuse$dist, elev = meuse$elev)
gam.mod <- gam(lzinc ~ s(x, y, bs = "tp") + dist +
  elev, data = zinc.dat, method = "REML")
```

log(zinc) modeled with a GAM

```
summary(gam.mod)
##
## Family: gaussian
## Link function: identity
##
## Formula:
## lzinc ~ s(x, y, bs = "tp") + dist + elev
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 8.52282   0.30738 27.727 < 2e-16 ***
## dist        -1.57105   0.62631 -2.508  0.0134 *  
## elev        -0.27677   0.03361 -8.235 1.71e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df      F p-value    
## s(x,y) 22.66  26.52 6.264 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.833  Deviance explained =  86%
## -REML = 67.057  Scale est. = 0.08709/ n = 155
```

GAM output: The fitted distance by elevation surface

```
vis.gam(gam.mod, theta = 30, phi = 30)
```



GAM prediction

```
pred.grid_gam <- expand.grid(seq(179000, 181000, l = 51),  
    seq(330000, 332000, l = 51))  
  
pred.dat_gam <- data.frame(x = pred.grid_gam[, 1],  
    y = pred.grid_gam[, 2], dist = mean(meuse$dist),  
    elev = mean(meuse$elev))  
zinc.pred_gam <- predict.gam(gam.mod, pred.dat_gam,  
    type = "terms")[, 3]
```

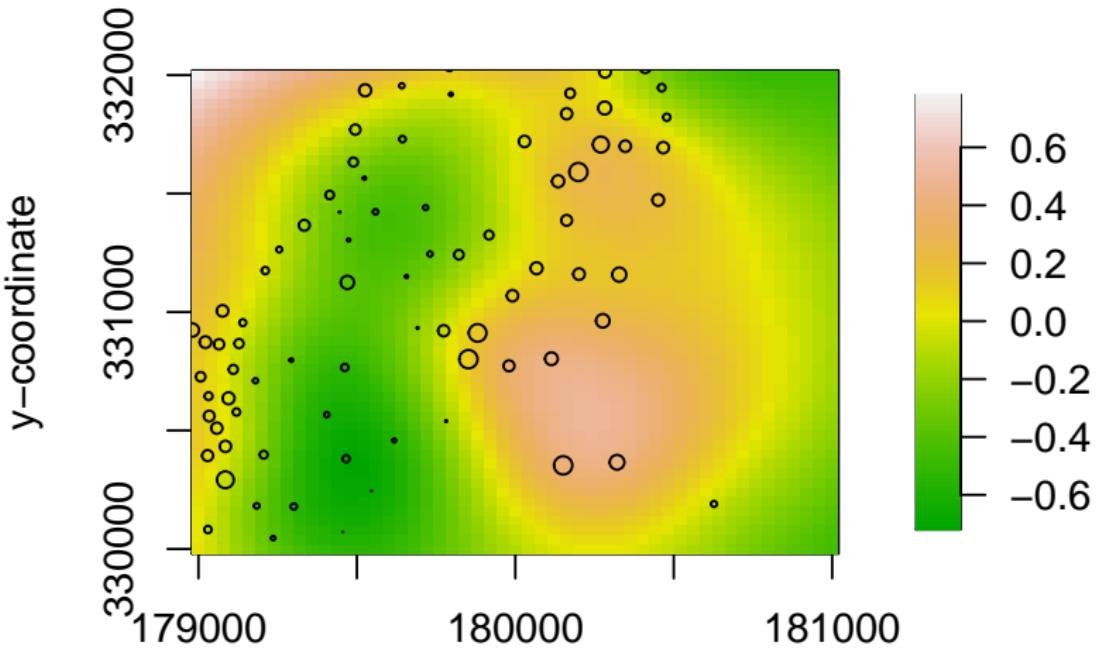
Prediction from GAM

```
library(fields)

image.plot(x = pred.grid_gam[["Var1"]][1:51], y = unique(pred.grid_gam$Var1)[1:51],
           z = matrix(zinc.pred_gam, nrow = 51, ncol = 51),
           col = terrain.colors(100), xlab = "x-coordinate",
           ylab = "y-coordinate")

symbols(detrend$coords[, 1], detrend$coords[, 2], circles = (detrend$data - min(detrend$data))/1, add = T, inches = 0.04)
```

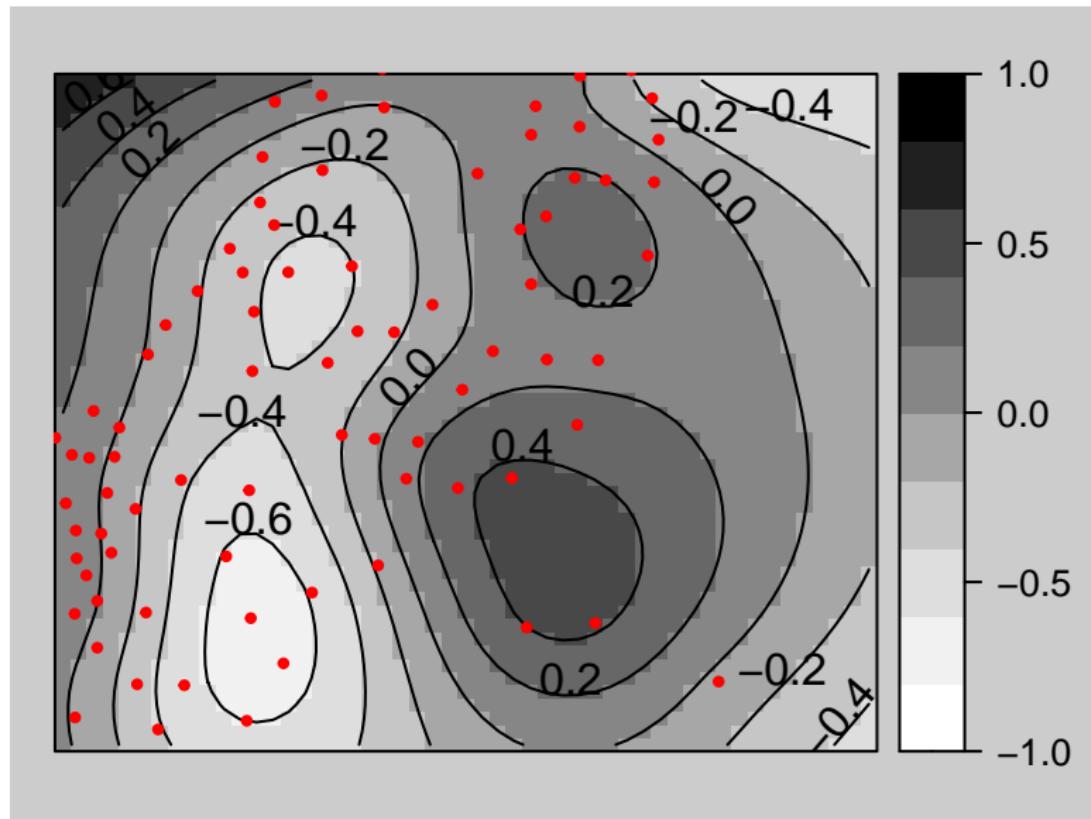
Prediction from GAM



Prediction from GAM

```
# create prediction grid
pred.grid <- expand.grid(seq(179000, 181000, l = 51),
                           seq(330000, 332000, l = 51))
pred.dat <- data.frame(x = pred.grid[, 1], y = pred.grid[, 2],
                        dist = mean(meuse$dist), elev = mean(meuse$elev))
zinc.pred <- predict.gam(gam.mod, pred.dat, type = "terms")[, 3]
# plot the smoother for log(zinc)
print(contourplot(zinc.pred ~ pred.grid[, 1] * pred.grid[, 2],
                   xlab = "", ylab = "", main = "", colorkey = T,
                   scales = list(draw = F), pretty = T, region = T,
                   par.settings = custom.theme(region = brewer.pal(9,
                                                               "Greys"), bg = "grey80")))
# add the observed points
trellis.focus("panel", 1, 1, highlight = FALSE)
lpoints(zinc.dat[, 1], zinc.dat[, 2], pch = 19, col = "red",
        cex = 1.5)
```

Prediction from GAM



Meuse analysis using geostat functions

The `sp` package functions can make full use of the GIS capabilities of R more readily.

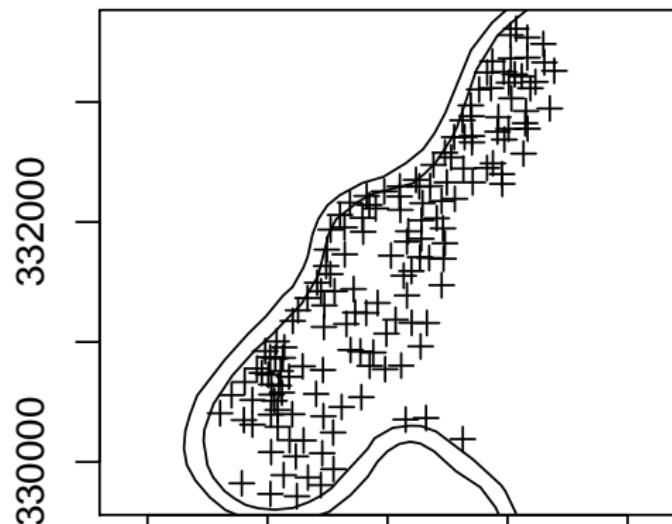
```
rm(list = ls())

pal <- function(n = 9) {
  brewer.pal(n, "Reds")
}

data(meuse)
coords <- SpatialPoints(meuse[, c("x", "y")])
meuse1 <- SpatialPointsDataFrame(coords, meuse)
data(meuse.riv)
river_polygon <- Polygons(list(Polygon(meuse.riv)),
  ID = "meuse")
rivers <- SpatialPolygons(list(river_polygon))
coordinates(meuse) = ~x + y
```

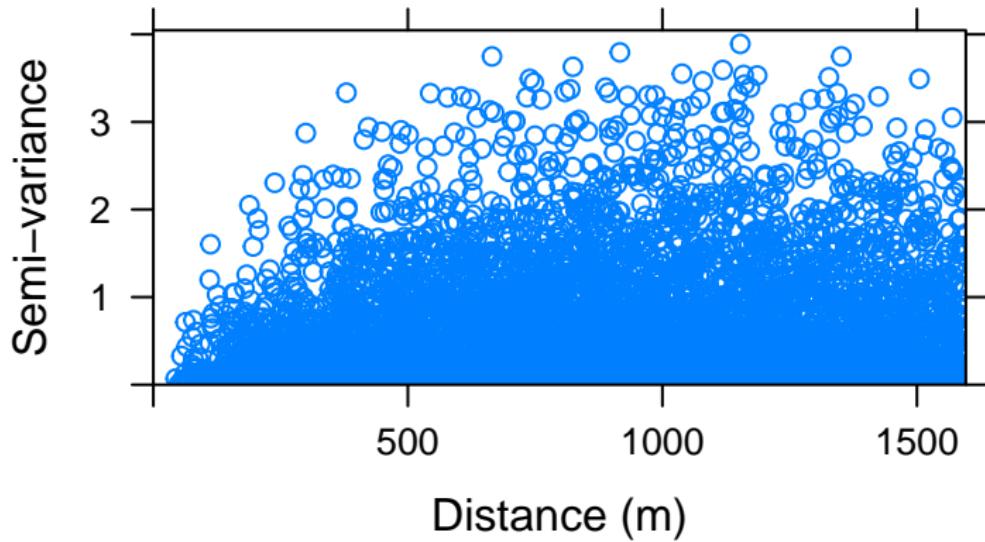
Zinc: Sampling locations

```
plot(meuse1, axes = T)  
plot(rivers, add = T)
```



$\log(\text{zinc})$: Variogram cloud, no trend removed

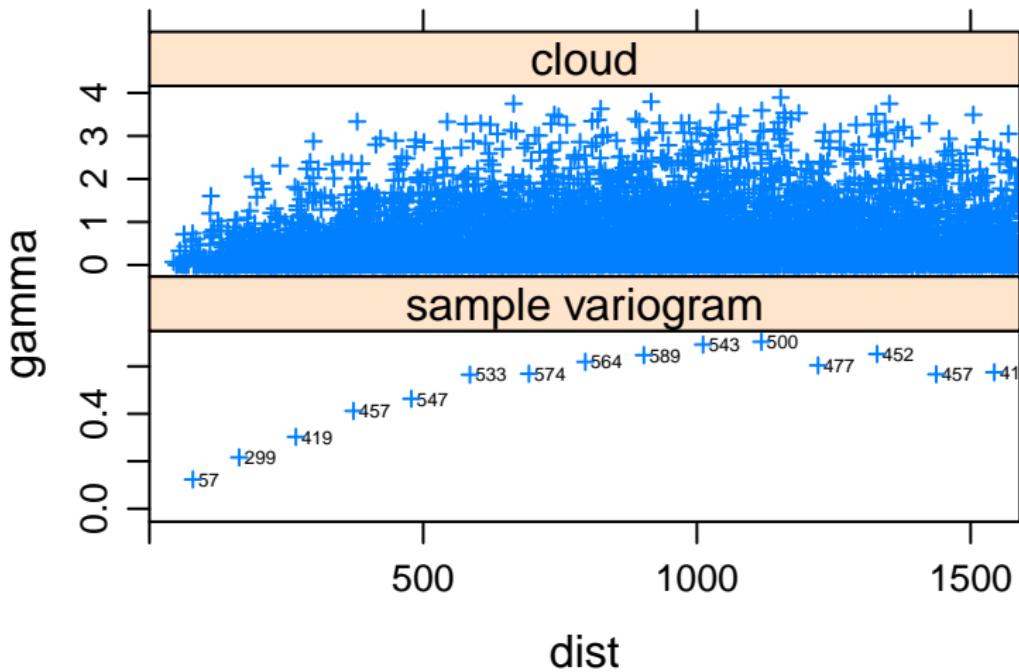
```
library(gstat)
cld <- variogram(log(zinc) ~ 1, meuse, cloud = TRUE)
plot(cld, ylab = "Semi-variance", xlab = "Distance (m)")
```



More variograms, with sample sizes

```
cld <- variogram(log(zinc) ~ 1, meuse, cloud = TRUE)
svgm <- variogram(log(zinc) ~ 1, meuse)
d <- data.frame(gamma = c(cld$gamma, svgm$gamma),
                 dist = c(cld$dist, svgm$dist),
                 id = c(rep("cloud", nrow(cld)), rep("sample variogram", nrow(svgm))))
xyplot(gamma ~ dist | id, d,
        scales = list(y = list(relation = "free",
                               #ylim = list(NULL, c(-.005,0.7))),
                      limits = list(NULL, c(-.005,0.7))),
        layout = c(1, 2), as.table = TRUE,
        panel = function(x,y, ...) {
          if (panel.number() == 2)
            ltext(x+10, y, svgm$np, adj = c(0,0.5),cex=.4) #$
          panel.xyplot(x,y,...)
        },
        xlim = c(0, 1590),
        cex = .5, pch = 3
      )
```

More variograms

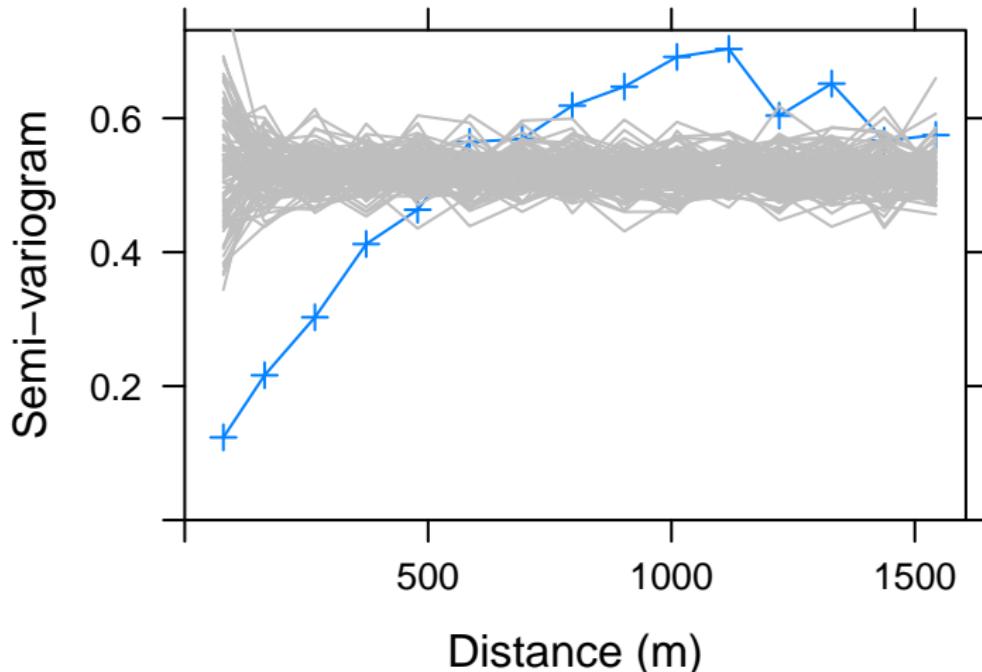


Monte Carlo simulations of semi-variogram

We simulate 100 datasets with random relabeling of points, and then form variograms for each.

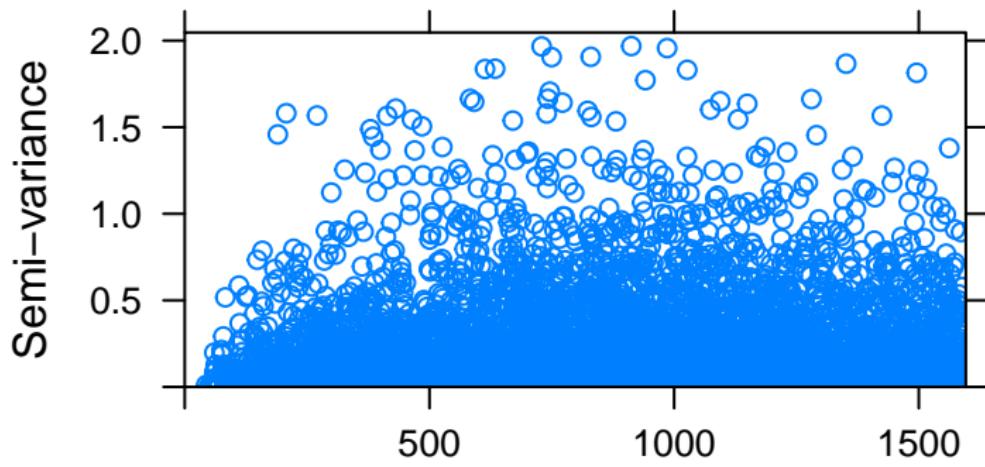
```
v <- variogram(log(zinc) ~ 1, meuse)
plot(v, type = "b", pch = 3, xlab = "Distance (m)",
      ylab = "Semi-variance")
fn = function(n = 100) {
  for (i in 1:n) {
    meuse$random = sample(meuse$zinc)
    v = variogram(log(random) ~ 1, meuse)
    trellis.focus("panel", 1, 1, highlight = FALSE)
    llines(v$dist, v$gamma, col = "grey")
    trellis.unfocus()
  }
}
fn()
```

Monte Carlo simulations of semi-variogram



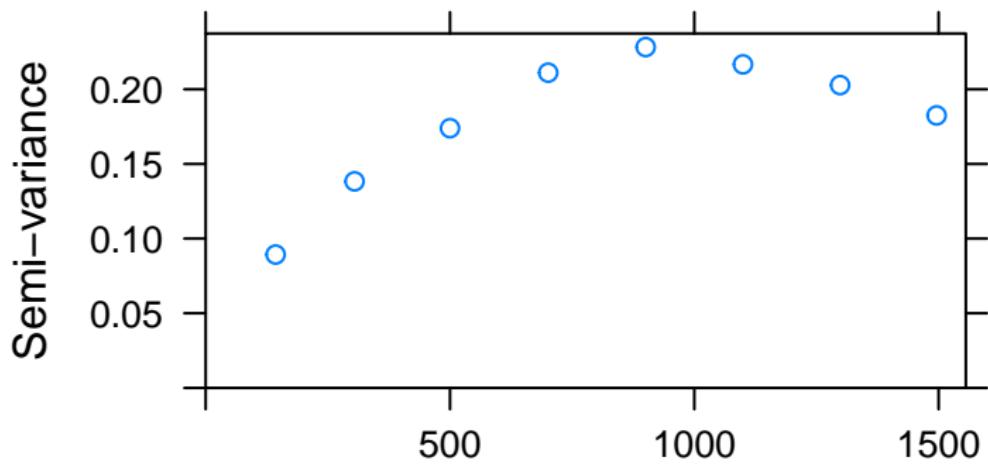
$\log(\text{zinc})$: Variogram cloud, detrended

```
cld2 <- variogram(log(zinc) ~ dist + elev,  
  meuse, cloud = TRUE)  
plot(cld2, ylab = "Semi-variance", xlab = "Distance (m)")
```



$\log(\text{zinc})$: Binned variogram, detrended

```
gstatbin <- variogram(log(zinc) ~ dist +
  elev, meuse, width = 200)
plot(gstatbin, ylab = "Semi-variance", xlab = "Distance (m)")
```



Zinc: Directional variogram with linear trend removed

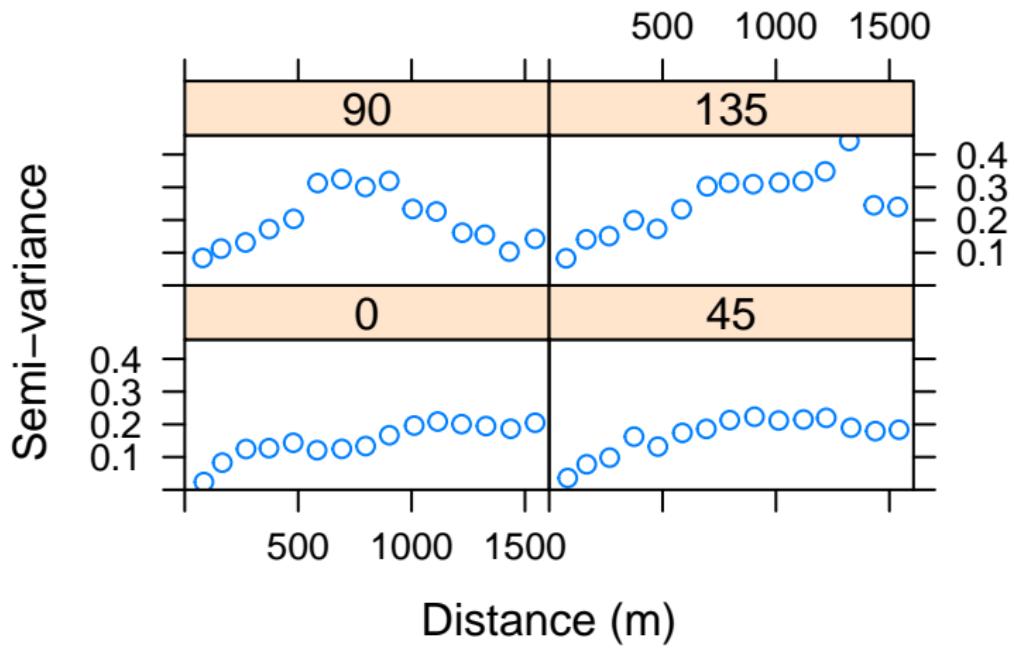
We form 4 variograms with data taken from different directions, with 0 and 90 corresponding to north and east, respectively.

Note that 0 is the same as 180.

```
dirclld <- variogram(log(zinc) ~ dist + elev,  
                      meuse, alpha = c(0, 45, 90, 135))
```

Zinc: Directional variogram with linear trend removed

```
plot(dirclld, xlab = "Distance (m)", ylab = "Semi-variance")
```

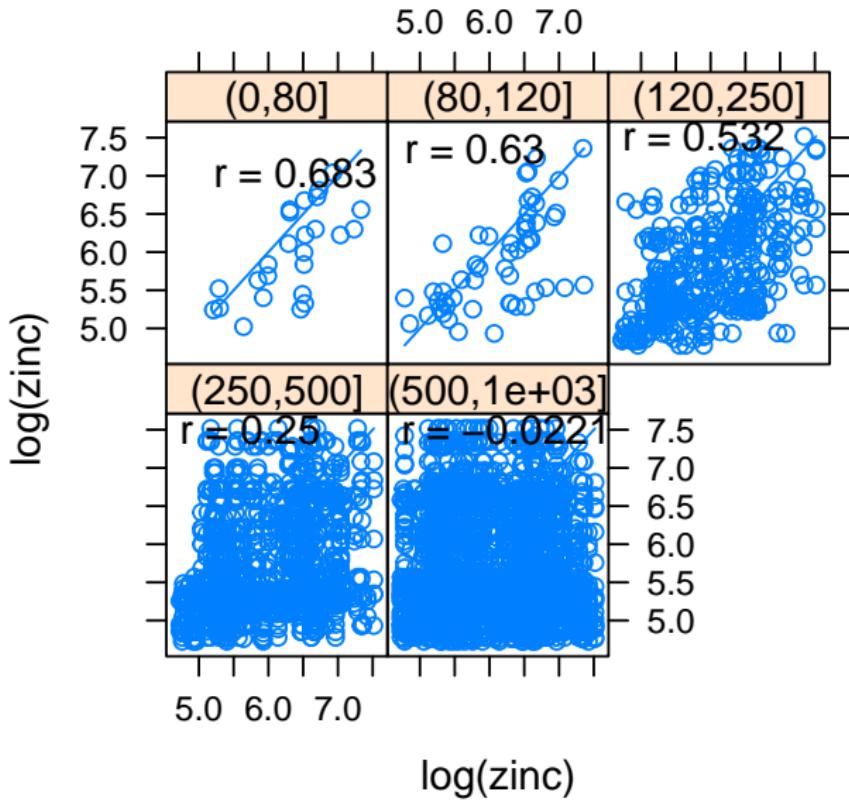


Zinc: Lagged scatterplots

We examine scatterplots of points within different distances of each other. This is another way of assessing whether spatial dependence exists.

```
hscat(log(zinc) ~ 1, meuse, c(0, 80, 120,  
250, 500, 1000), cex = 0.1)
```

lagged scatterplots



Quick Bayesian analysis (Brown 2014)

Fit a multivariate spatial Matern model to the residuals (just to show we can!)

Model is fitted by discretizing space over a grid.

Priors need more thought here

Quick Bayesian analysis (Brown 2014)

```
library(geostatssp)

coord <- matrix(c(meuse$x, meuse$y), ncol = 2)
lzinc <- log(meuse$zinc)
elev <- meuse$elev
dist <- meuse$dist
dfmeuse <- data.frame(lzinc, elev, dist)
data.spdf <- SpatialPointsDataFrame(coords = coord,
  data = dfmeuse)
formula <- lzinc ~ elev + dist
zinclglm <- glgm(formula, grid = 30, shape = 1,
  family = "gaussian", buffer = 200, data = data.spdf,
  priorCI = list(c(sd = 0.1, 1), range = c(100,
    1200)))
zinclglm$parameters$summary
##               mean           sd 0.025quant   0.5quant   0.975quant
## (Intercept) 8.7475484 2.653348e-01 8.2300217 8.7459424 9.2737443
## elev        -0.2799532 2.977606e-02 -0.3386453 -0.2799171 -0.2215290
## dist        -2.2062605 3.585543e-01 -2.9256152 -2.2020628 -1.5106607
## range       676.8708044 1.725023e+02 421.5307817 648.0004171 1092.2676393
## sdNugget     0.1843628 1.171805e-03  0.1382047  0.1829790  0.2461372
## sd          0.4564314 6.877689e-03  0.3457511  0.4503073  0.6139097
```

Other capabilities in gstat

See

- `fit.variogram` for estimation from the variogram
- `krige` (and associated functions) for Kriging,
- `vgm` generates variogram models

Binary point data: Cross-sectional asthma survey

We first consider data that were analyzed by Diggle and Rowlingson (1994). Ten schools were surveyed within the North Derbyshire and parents answered a question, "has the child ever suffered from asthma".

A number of household and child-level variables were also collected.

The following example leans heavily on an analysis carried out in Bivand et al (2013, Chapter 7).

We first load some libraries.

```
library(rgdal)
library(maptools)
library(splancs)
library(spatstat)
```

Binary point data: Cross-sectional asthma survey

The file names in quotes need to be in the current directory: we load data, a boundary, pollution sources and roads.

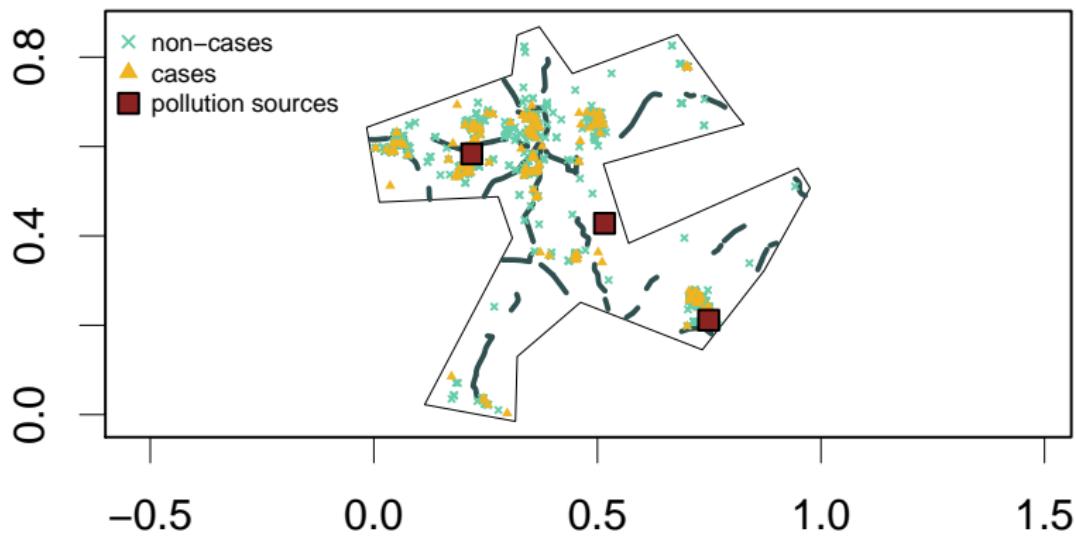
```
spasthma <- readOGR(dsn = "R-examples", layer = "spasthma")
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\wu-th\Dropbox\2019-SISMID\2021-Lectures\2021-SISMID-R-SESSIONS\
## with 1291 features
## It has 9 fields
spbdry <- readOGR(dsn = "R-examples", layer = "spbdry")
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\wu-th\Dropbox\2019-SISMID\2021-Lectures\2021-SISMID-R-SESSIONS\
## with 1 features
## It has 1 fields
spsrc <- readOGR(dsn = "R-examples", layer = "spsrc")
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\wu-th\Dropbox\2019-SISMID\2021-Lectures\2021-SISMID-R-SESSIONS\
## with 3 features
## It has 1 fields
sproads <- readOGR(dsn = "R-examples", layer = "sproads")
## OGR data source with driver: ESRI Shapefile
## Source: "C:\Users\wu-th\Dropbox\2019-SISMID\2021-Lectures\2021-SISMID-R-SESSIONS\
## with 1 features
```

Binary point data

We now provide a plot of the case and non-case locations, along with a boundary, major roads, and pollution sources.

```
plot(spbdry, axes = TRUE, lwd = 0.5)
plot(spreads, add = TRUE, lwd = 2, col = "darkslategrey")
c_c <- (spasthma$Asthma == "case") + 1
plot(spasthma[c_c == 1, ], add = TRUE, pch = 4, cex = 0.6,
     col = "mediumaquamarine")
plot(spasthma[c_c == 2, ], add = TRUE, pch = 17, cex = 0.75,
     col = "goldenrod2")
plot(spsrc, pch = 22, add = TRUE, cex = 1.2, bg = "brown4")
legend("bottomright", legend = c("non-cases", "cases",
    "pollution sources"), pch = c(4, 17, 22), pt.cex = c(0.6,
    0.75, 1.2), pt.bg = c(NA, NA, "brown4"), col = c("mediumaquamarine",
    "goldenrod2", "black"), bty = "n", cex = 0.6)
```

Binary point data



Binary point data: KDE

We now form KDE estimates of the densities of cases $f_1(x)$ and non-cases $f_0(x)$: `kcases` for the cases and `kcontrols` for the non-cases.

These density estimates are evaluated on a grid.

Trial and error leads to a bandwidth choice of 0.06.

We create an object of class `ppp` representing a point pattern dataset in the 2D plane.

Note: we don't have a case-control dataset here, but we use the language to form objects.

Binary point data: KDE

First we set the bandwidth.

```
bwasthma <- 0.06
pppasthma <- as(spasthma, "ppp")
pppasthma>window <- as(spbdry, "owin")
marks(pppasthma) <- relevel(as.factor(pppasthma$marks$Asthma),
  "control")
cases <- unmark(subset(pppasthma, marks(pppasthma) ==
  "case"))
ncases <- npoints(cases)
controls <- unmark(subset(pppasthma, marks(pppasthma) ==
  "control"))
ncontrols <- npoints(controls)
```

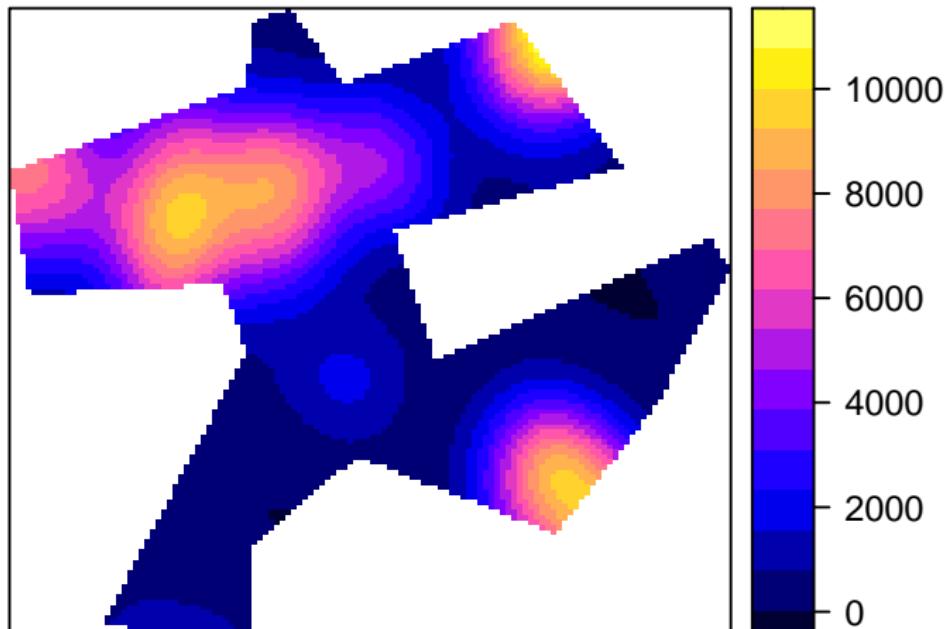
Binary point data: KDE

spkratio object contains both the ratio of density estimates (the odds, kratio) and the log ratio (the log odds, logratio).

```
kcases <- density(cases, bwasthma)
kcontrols <- density(controls, bwasthma)
spkratio0 <- as(kcases, "SpatialGridDataFrame")
names(spkratio0) <- "kcases"
spkratio0$kcontrols <- as(kcontrols, "SpatialGridDataFrame")$v
spkratio <- as(spkratio0, "SpatialPixelsDataFrame")
spkratio$kratio <- spkratio$kcases/spkratio$kcontrols
spkratio$kratiornorm <- spkratio$kratio/(ncases/ncontrols)
spkratio$logratio <- log(spkratio$kratio) - log(ncases/ncontrols)
```

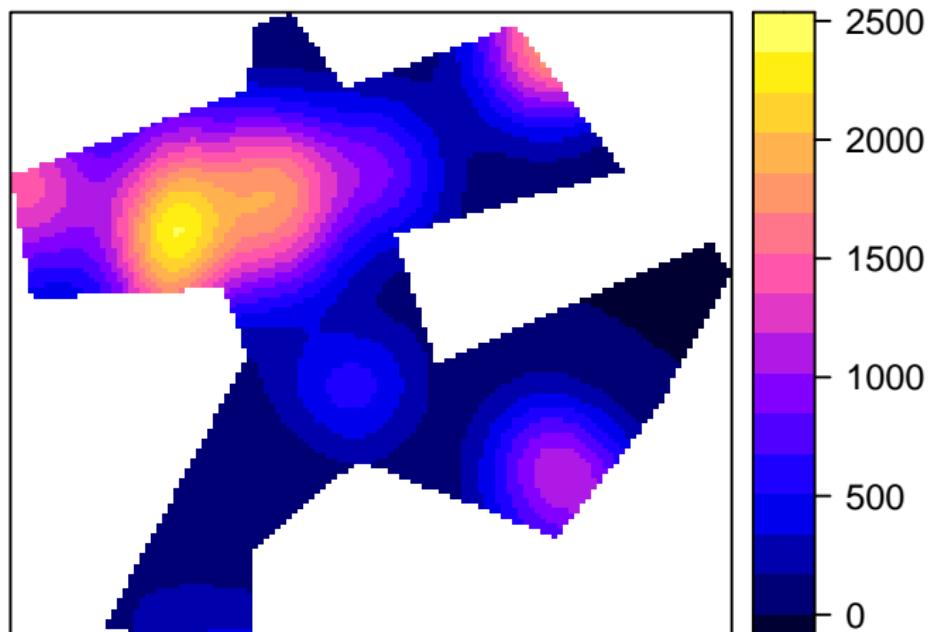
Binary point data: KDEs of control distribution

```
spplot(spkratio, "kcontrols")
```



Binary point data: KDEs of case distribution

```
spplot(spkratio, "kcases")
```



Binary point data

We have

$$\frac{\hat{f}_1(s)}{\hat{f}_0(s)} = \frac{p(s)}{1 - p(s)} \times \frac{c_0}{c_1},$$

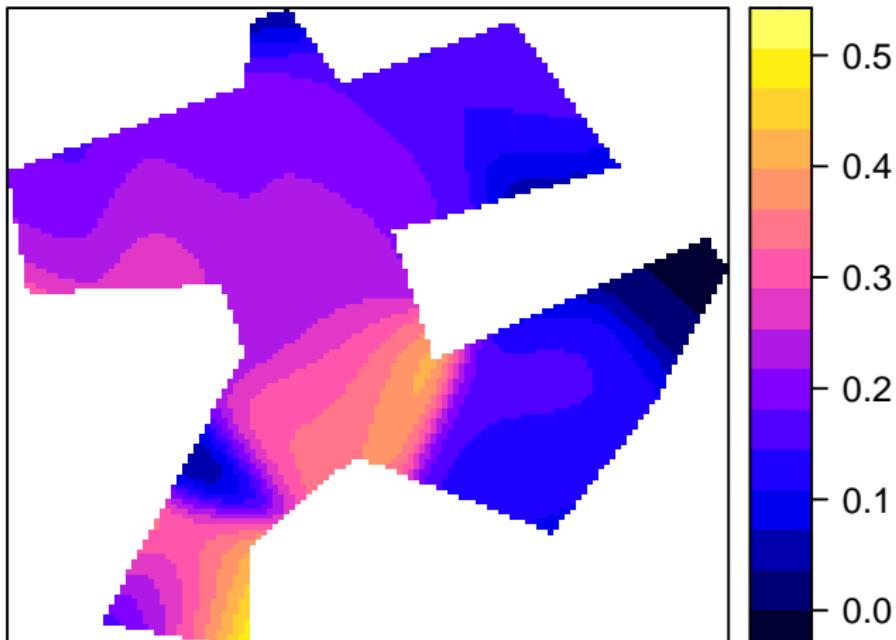
where $c_j = \int_A \lambda_j(s) ds$.

The plot below show the ratio of the densities of cases and controls. We have not included estimates of c_0 , c_1 .

Grey crosses mark the pollution surfaces.

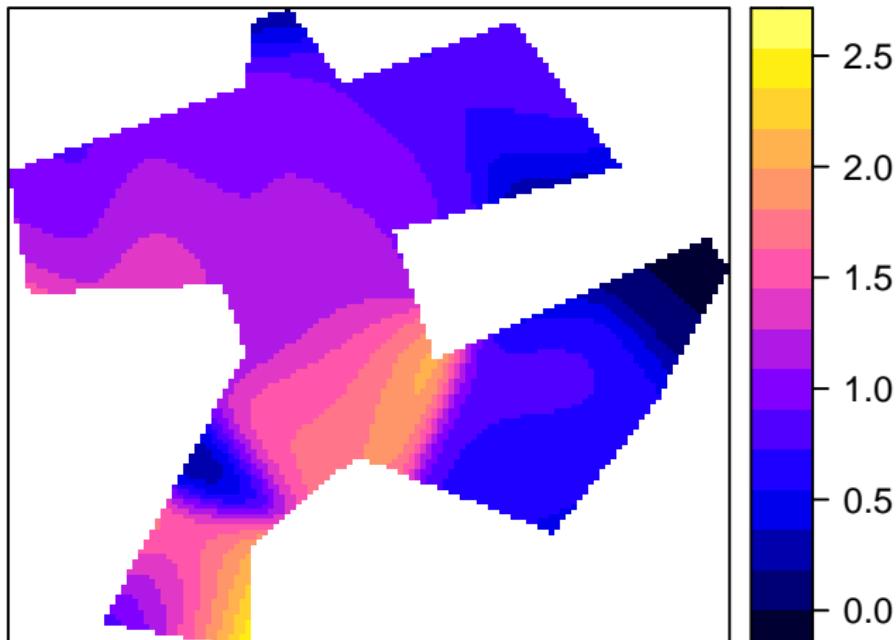
Binary point data: ratio of densities

```
spplot(spkratio, "kratio")
```



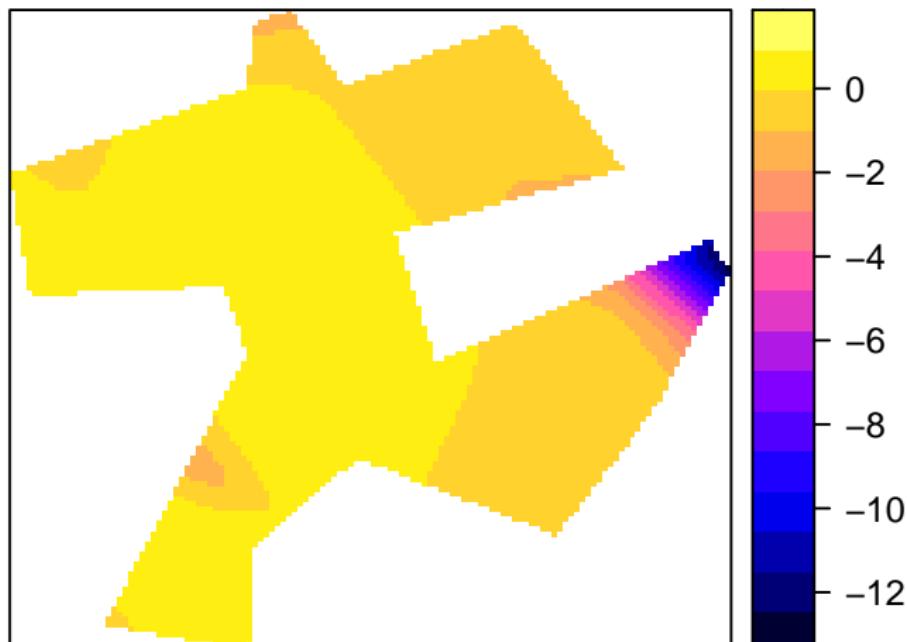
Binary point data: odds surface (ie normalized by n_0/n_1)

```
spplot(spkratio, "kratnorm")
```



Binary point data: log odds ratio surface

```
spplot(spkratio, "logratio")
```



Binary point data

Let

$$\rho(s) = \frac{n_0}{n_1} \times \frac{f_1(s)}{f_0(s)}$$

represent the odds surface.

Let H_0 represent the null of a constant odds surface. Kelsall and Diggle (1995a,b) suggested the use of the statistic

$$T = \int_A [\hat{\rho}(s) - \hat{\rho}_0]^2 ds,$$

where $\hat{\rho}_0 = n_1/n_0$.

This null can be evaluated via a Monte Carlo test in which the points are randomly relabelled and the integral is evaluated on a grid.

Binary point data: p -value map

```
niter <- 99
ratio <- rep(NA, niter)
pvaluemap <- rep(0, nrow(spkratio))
rlabelratio <- matrix(NA, nrow = niter, ncol = nrow(spkratio))
set.seed(1)
for (i in 1:niter) {
  pppasthma0 <- rlabel(pppasthma)
  casesrel <- unmark(subset(pppasthma0, marks(pppasthma0) ==
    "case"))
  controlsrel <- unmark(subset(pppasthma0, marks(pppasthma0) ==
    "control"))
  kcasesrel <- density(casesrel, bwasthma)
  kcontrolsrel <- density(controlsrel, bwasthma)
  kratiorel <- eval.im(kcasesrel/kcontrolsrel)
  rlabelratio[i, ] <- as(as(kratiorel, "SpatialGridDataFrame"),
    "SpatialPixelsDataFrame")$v
  pvaluemap <- pvaluemap + (spkratio$kratio < rlabelratio[i,
    ]))
}
```

Binary point data

We report the p -value that assesses whether the overall surface is flat.

```
cellsize <- kcontrols$xstep * kcontrols$ystep
ratiorho <- cellsize * sum((spkratio$kratio - ncases/ncontrols)^2)
ratio <- cellsize * apply(rlabelratio, 1, function(X,
  rho0) {
  sum((X - rho0)^2)
}, rho0 = ncases/ncontrols)
pvaluerho <- (sum(ratio > ratiorho) + 1)/(niter + 1)
pvaluerho
## [1] 0.62
```

The overall p -value is 0.61 so no evidence of non-constant odds over the map as a whole.

Binary point data

We now start to construct the pointwise p -value map.

```
spkratio$pvaluemap <- (pvaluemap + 1)/(niter + 1)
imgpvalue <- as.image.SpatialGridDataFrame(spkratio["pvaluemap"])
clpvalue <- contourLines(imgpvalue, levels = c(0, 0.05,
    0.95, 1))
cl <- ContourLines2SLDF(clpvalue)
```

Binary point data: set up for contour lines on *p*-value plot

```
library(RColorBrewer)
cl05 <- cl[cl$level == "0.05", ]
xzx <- slot(slot(cl05, "lines")[[1]], "Lines")
cl05a <- SpatialLines(list(Lines(xzx, ID = "0.05")))
lyt05 <- list("sp.lines", cl05a, lwd = 2, lty = 2,
              col = "grey95")
lyt95 <- list("sp.lines", cl[cl$level == "0.95", ],
              lwd = 2, lty = 1)
lytb <- list("sp.polygons", spbdry)
lytp <- list("sp.points", spsrc, cex = 0.9, pch = 4,
              col = "grey95", lwd = 3)
brks <- quantile(spkratio$kratio[spkratio$kratio >
  0], seq(0, 1, 1/10), na.rm = TRUE)
brks[1] <- 0
lbrks <- formatC(brks, 3, 6, "g", " ")
cols <- colorRampPalette(brewer.pal(7, "Reds"))(length(brks) -
  1)
colorkey <- list(labels = lbrks, at = (0:10)/10, height = 0.5)
```

Binary point data

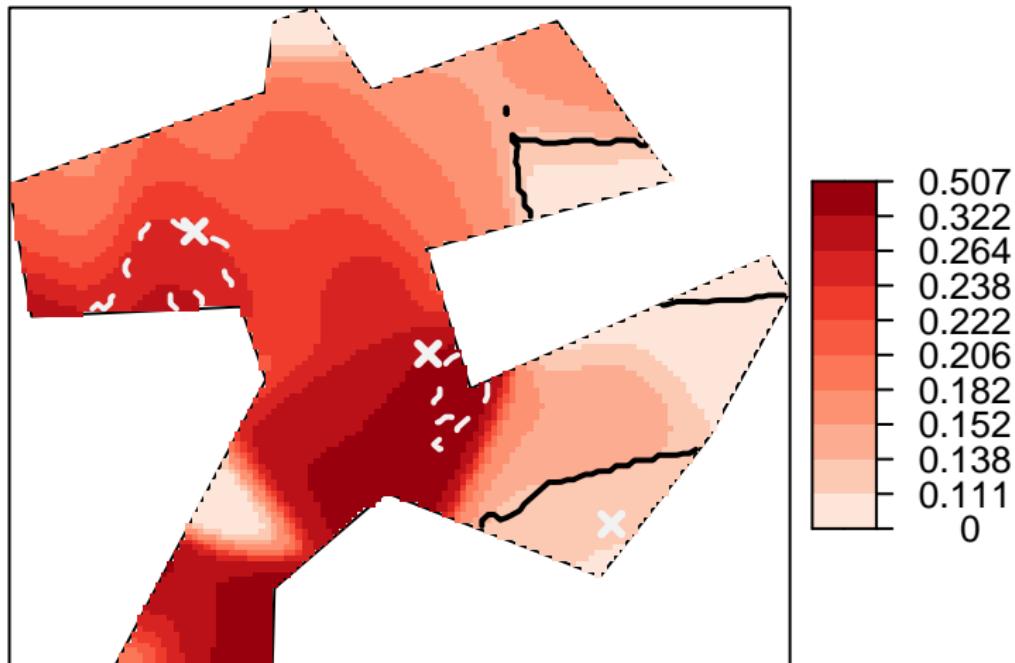
The plot below shows the KDE estimate of the ratio of the intensity of cases and controls, i.e. $\rho(s)$.

The continuous and dashed lines show the surfaces associated with 0.95 and 0.05 p -values, respectively. The estimate of the null value for a constant (flat) surface is $\hat{\rho} = 0.20$.

Grey crosses mark the pollution surfaces.

```
spplot(spkratio, "kratio", col.regions = cols,
  do.log = TRUE, colorkey = colorkey, at = c(0,
    brks[-c(1, 11)], max(spkratio$kratio,
      na.rm = TRUE)), sp.layout = list(lyt05,
    lyt95, lytb, lytp))
```

Binary point data: ratio of densities, with p -values



Binary point data

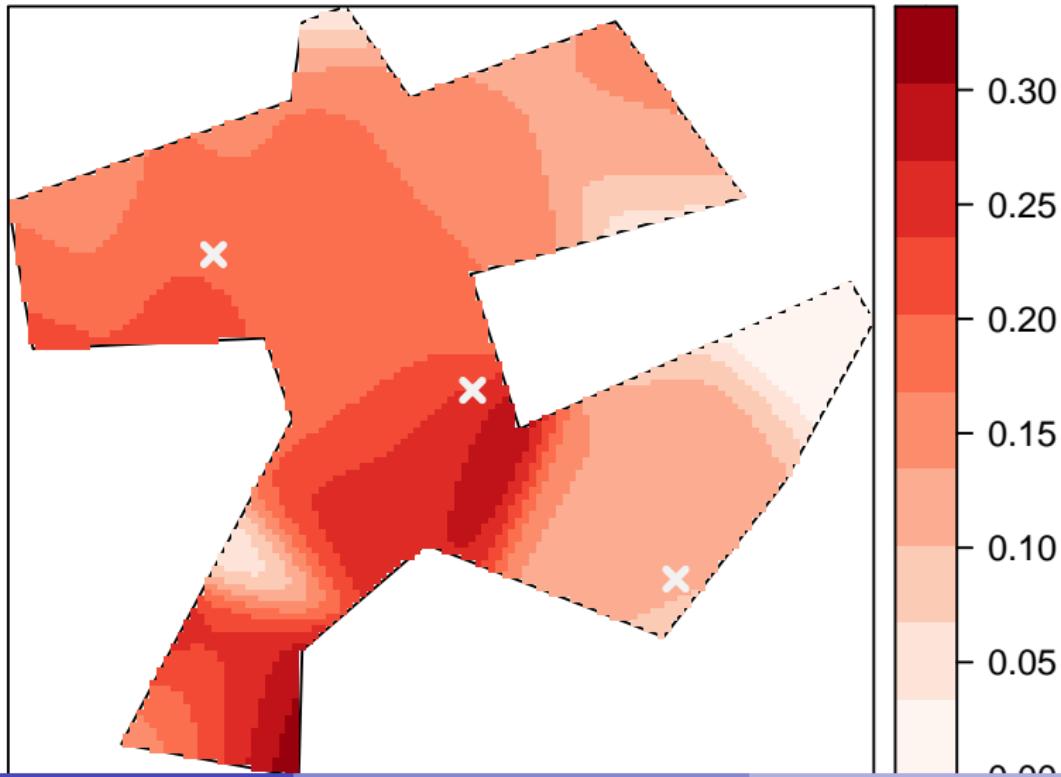
We now turn to a binary regression estimator, as suggested by Kelsall and Diggle (1998).

```
rrbw <- bw.relrisk(pppasthma, hmax = 0.5)
bwasthmap <- 0.06
rr <- relrisk(pppasthma, bwasthmap)
spkratio$prob <- as(as(rr, "SpatialGridDataFrame"),
  "SpatialPixelsDataFrame")$v
ats <- seq(0, max(spkratio$prob), length.out = 11)
cols <- colorRampPalette(brewer.pal(8, "Reds"))(length(ats) - 1)
```

Binary point data: binary regression estimator

```
spplot(spkratio, "prob", col.regions = cols, at = ats,  
      sp.layout = list(lytb, lytp))
```

Binary point data



Binary point data

We now turn to the use of a generalized additive model (GAM).

We use the example of Bivand et al. (2013, Section 7.5.3) in which a penalized regression spline is used, and a variety of covariates are adjusted for.

```
spasthma$y <- as.integer(as.factor(spasthma$Asthma)) -  
  1 #as.integer(!as.integer(spasthma$Asthma)-1)  
ccasthma <- coordinates(spasthma)  
spasthma$x1 <- ccasthma[, 1]  
spasthma$x2 <- ccasthma[, 2]  
spasthma$dist1 <- sqrt(spasthma$d2source1)  
spasthma$dist2 <- sqrt(spasthma$d2source2)  
spasthma$dist3 <- sqrt(spasthma$d2source3)  
spasthma$droads <- sqrt(spasthma$roaddist2)  
spasthma$smoking <- as.factor(as.numeric(spasthma$Nsmokers >  
  0))  
spasthma$Genderf <- as.factor(spasthma$Gender)
```

Binary point data

The gam model has a smoother in space, and a parametric (logistic regression) model for the covariates.

```
library(mgcv)
gasthma <- gam(y ~ 1 + dist1 + dist2 + dist3 + droads +
  Genderf + Age + HayFeverf + smoking + s(x1, x2),
  data = spasthma[spasthma$Gender == 1 | spasthma$Gender ==
    2, ], family = binomial)
```

Binary point data

```
summary(gasthma)
##
## Family: binomial
## Link function: logit
##
## Formula:
## y ~ 1 + dist1 + dist2 + dist3 + droads + Genderf + Age + HayFeverf +
##      smoking + s(x1, x2)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 2.033e+00 9.197e-01 2.210   0.0271 *
## dist1      -9.822e-01 6.072e+00 -0.162   0.8715
## dist2       9.579e+00 5.772e+00  1.660   0.0970 .
## dist3      -1.122e+01 7.874e+00 -1.425   0.1540
## droads     -1.479e-04 1.717e-04 -0.861   0.3890
## Genderf2    3.477e-01 1.562e-01  2.226   0.0260 *
## Age        6.790e-02 3.823e-02  1.776   0.0757 .
## HayFeverf1 -1.188e+00 1.875e-01 -6.335 2.37e-10 ***
## smoking1   -1.651e-01 1.610e-01 -1.025   0.3052
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df Chi.sq p-value
## s(x1,x2) 2.001 2.001 7.004 0.0302 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0403  Deviance explained = 4.94%
## UBRE = -0.12348  Scale est. = 1           n = 1283
```

Binary point data: plot of the residual smoothed surface

