

SISMID 2021 R Notes: Mapping for Point Data

Jon Wakefield
University of Washington

2021-07-13

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, and
- generalized additive models.

Continuous Responses

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 $28\text{km} \times 39\text{km}$.

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.

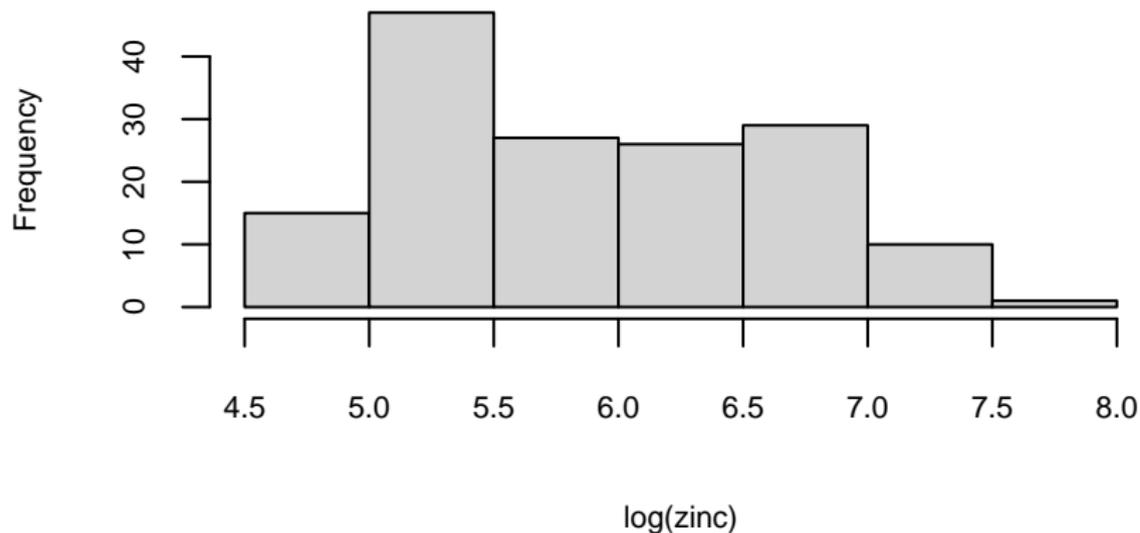
```
library(geoR)
library(sp)
data("meuse")
zmat <- matrix(cbind(meuse$x, meuse$y, log(meuse$zinc)),
              ncol = 3, nrow = 155, byrow = F)
geozinc <- as.geodata(zmat, coords.col = c(1,
              2), data.col = c(3))
```

There are 155 observations (sampling locations).

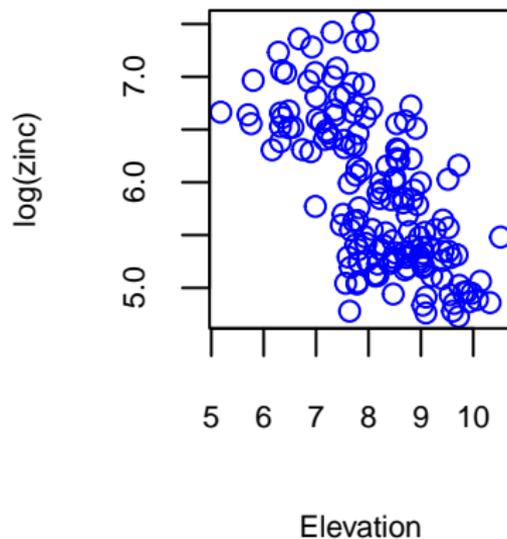
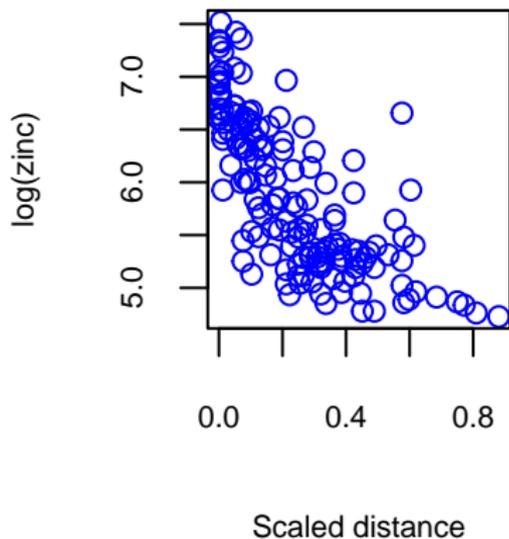
We work with $\log(\text{zinc})$ as the distribution is more symmetric than on the original scale (see next slide), and the variance more constant across levels of covariates.

Zinc data

```
hist(log(meuse$zinc), main = "", xlab = "log(zinc)",  
     cex.lab = 0.7, cex.axis = 0.7)
```

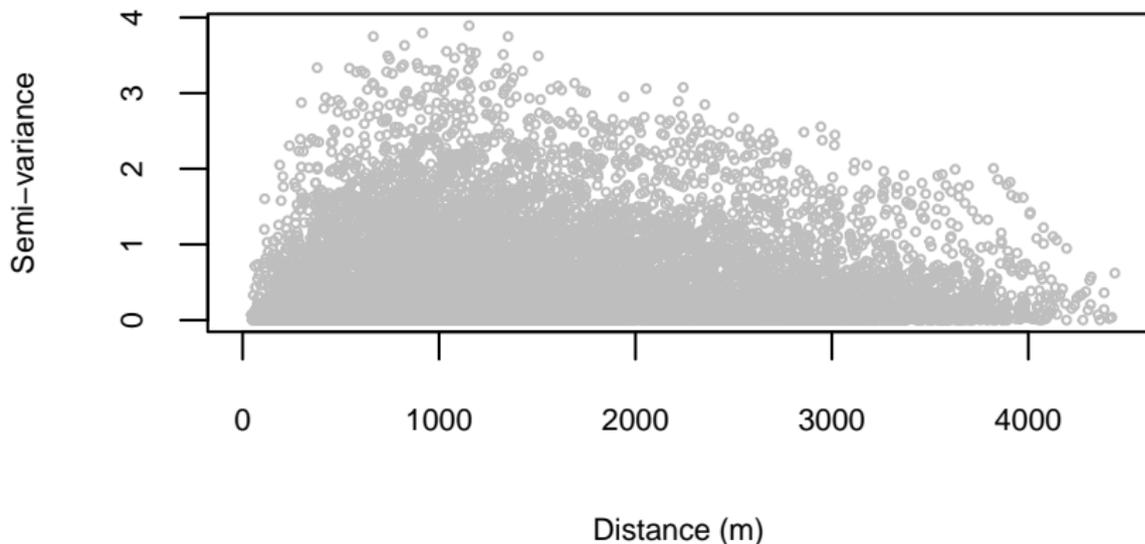


$\log(\text{zinc})$ versus distance from river and elevation



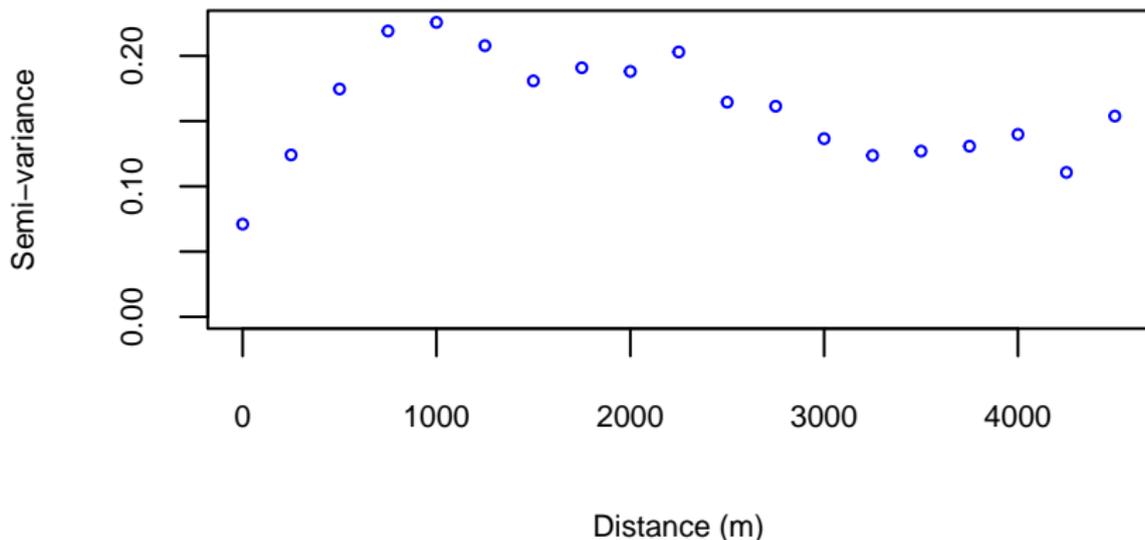
log(zinc): Variogram cloud, constant mean

```
cloudzinc <- variog(geozinc, option = "cloud")  
plot(cloudzinc, ylab = "Semi-variance", xlab = "Distance (m)",  
     col = "grey", cex = 0.4, cex.lab = 0.7,  
     cex.axis = 0.7)
```



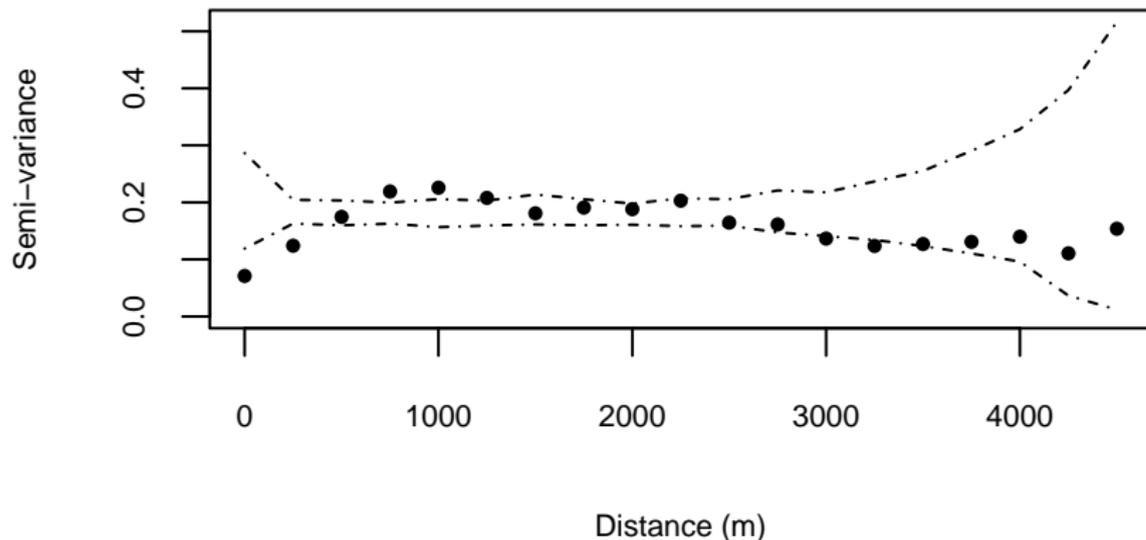
log(zinc): Binned variogram with linear model

```
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", cex.lab = 0.7,
  cex.axis = 0.7)
```



log(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", cex = 0.7, pch = 16,
     cex.lab = 0.7, cex.axis = 0.7)
```



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 $\log(\text{zinc})$

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.

Maximum likelihood for log(zinc)

```

mlfit$parameters.summary
##           status  values
## beta0  estimated  8.6162
## beta1  estimated -2.1072
## beta2  estimated -0.2690
## tausq  estimated  0.0010
## sigmasq estimated  0.2065
## phi    estimated 241.1982
## kappa  fixed     0.5000
## psiA   fixed     0.0000
## psiR   fixed     1.0000
## lambda fixed     1.0000
mlfit$beta.var
##           V1           V2           V3
## V1  0.065431719 -0.006902709 -0.0066575206
## V2 -0.006902709  0.116554531 -0.0024100551
## V3 -0.006657521 -0.002410055  0.0008936568
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.

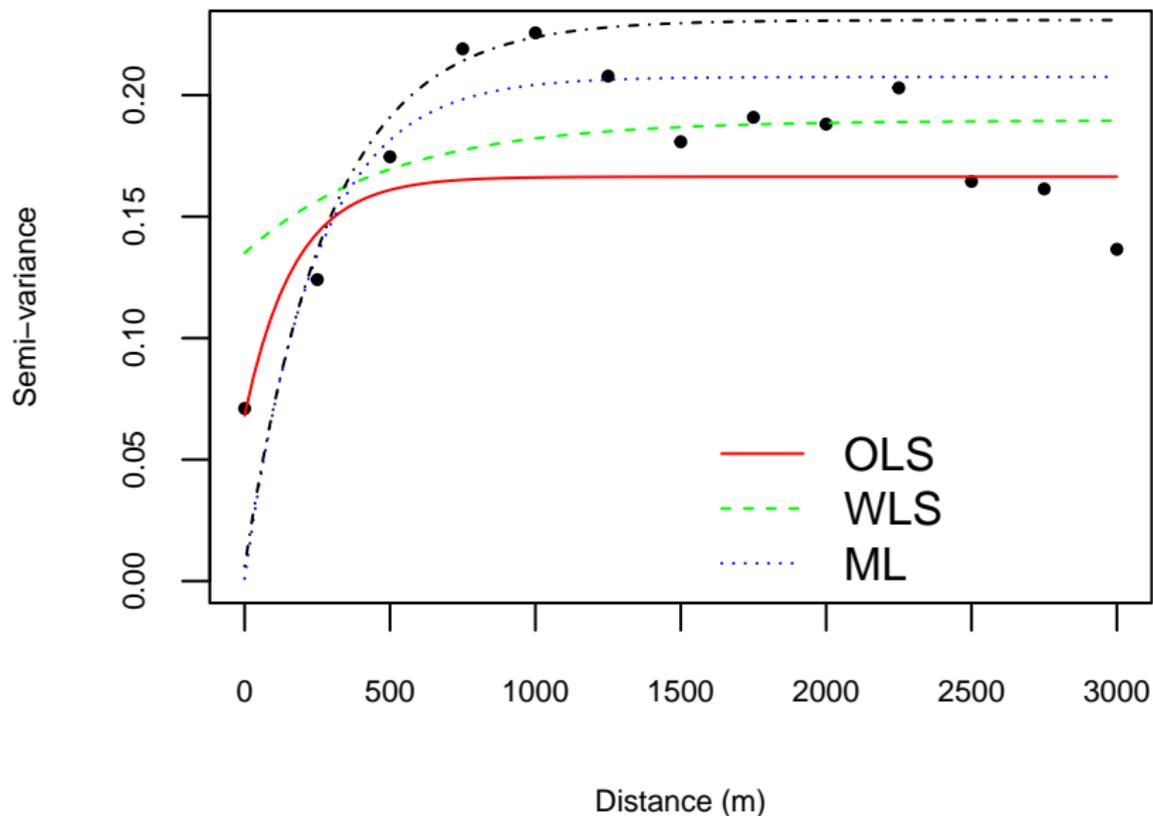
```
remlfit$parameters.summary
##           status  values
## beta0  estimated  8.6396
## beta1  estimated -2.1215
## beta2  estimated -0.2701
## tausq  estimated  0.0061
## sigmasq estimated  0.2248
## phi    estimated 289.1468
## kappa   fixed    0.5000
## psiA    fixed    0.0000
## psiR    fixed 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

Comparison of estimates

Here we compare various fits to the variogram: in general, OLS and WLS are not recommended!

```
plot(binzinc, max.dist = 3000, xlab = "Distance (m)",
     ylab = "Semi-variance", pch = 19, cex = 0.5, cex.axis = 0.7,
     cex.lab = 0.7)
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"))
```

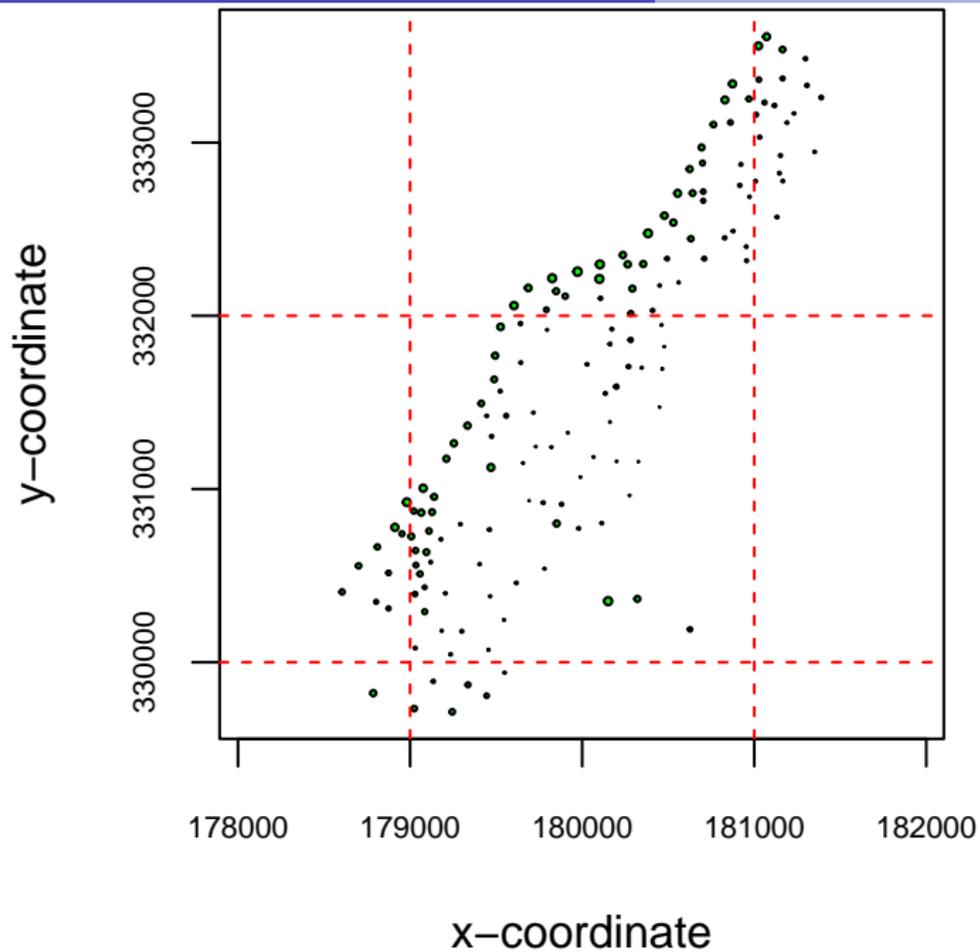
Comparison of estimates



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

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", cex.axis = 0.7)  
# 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 fit a linear model in distance and elevation to $\log(\text{zinc})$.

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
dettrend <- as.geodata(cbind(geozinc$coords, lmfit$residuals))
```

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

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 $\log(\text{zinc})$

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
```

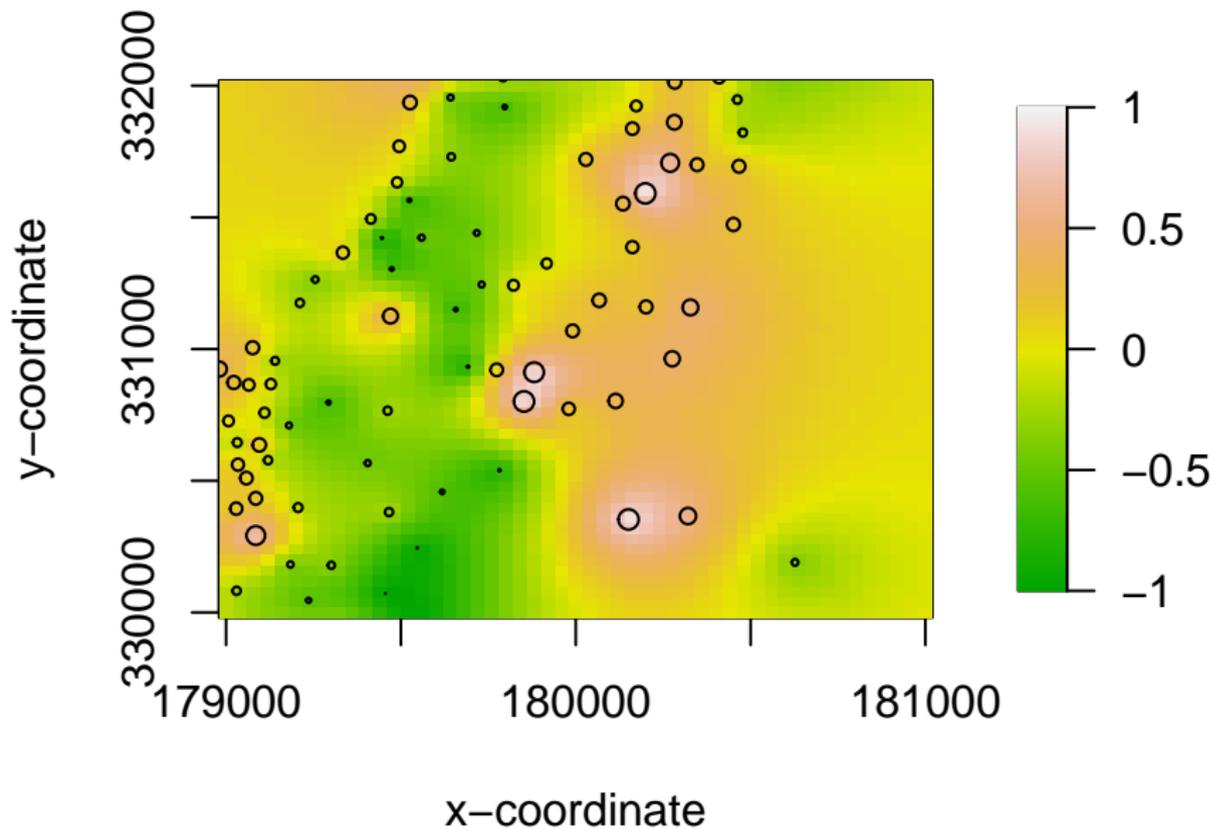
Prediction for log(zinc)

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[["Var2"]]),
  z = matrix(kc$predict, nrow = 51, ncol = 51), col = terrain.colors(100),
  xlab = "x-coordinate", ylab = "y-coordinate", breaks = seq(-1,
    1, , 101), axis.args = list(at = c(-1, -0.5,
    0, 0.5, 1), labels = c("-1", "-0.5", "0", "0.5",
    "1")), legend.cex = 0.5)

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



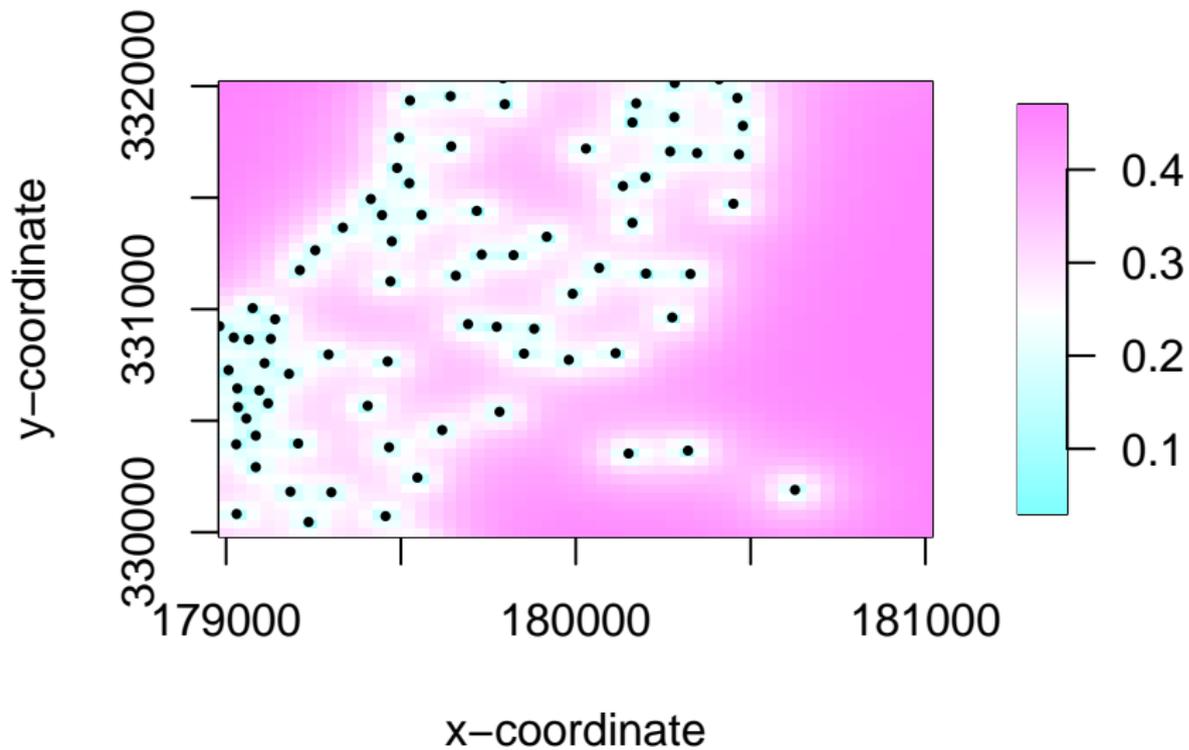
Standard deviations of prediction for log(zinc)

We now plot the Kriging standard deviations of the predictions.

```
image.plot(x = pred.grid[["Var1"]][1:51], y = unique(pred.grid[["Var2"]]),
  z = matrix(sqrt(kc$krige.var), nrow = 51, ncol = 51),
  col = cm.colors(100), xlab = "x-coordinate", ylab = "y-coordinate",
  breaks = seq(0.03, 0.47, , 101), axis.args = list(at = c(0.1,
    0.2, 0.3, 0.4), labels = c("0.1", "0.2", "0.3",
    "0.4"))))

points(detrend$coords[, 1], detrend$coords[, 2], cex = 0.5,
  pch = 16)
```

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



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.

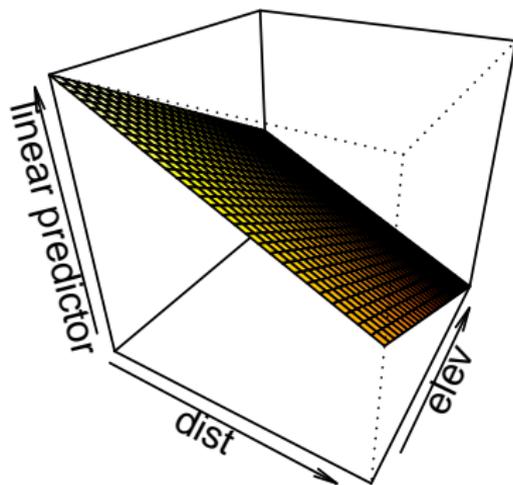
```
library(mgcv)
library(lattice)
library(latticeExtra)
library(RColorBrewer)
zinc.dat <- data.frame(x = meuse$x, y = meuse$y, lzinc = log(meuse$zinc),
  dist = meuse$dist, elev = meuse$elev)
gam.mod <- gam(lzinc ~ s(x, y, bs = "tp") + dist +
  elev, data = zinc.dat, method = "REML")
```

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 8.64e-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. = 0.087094  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))

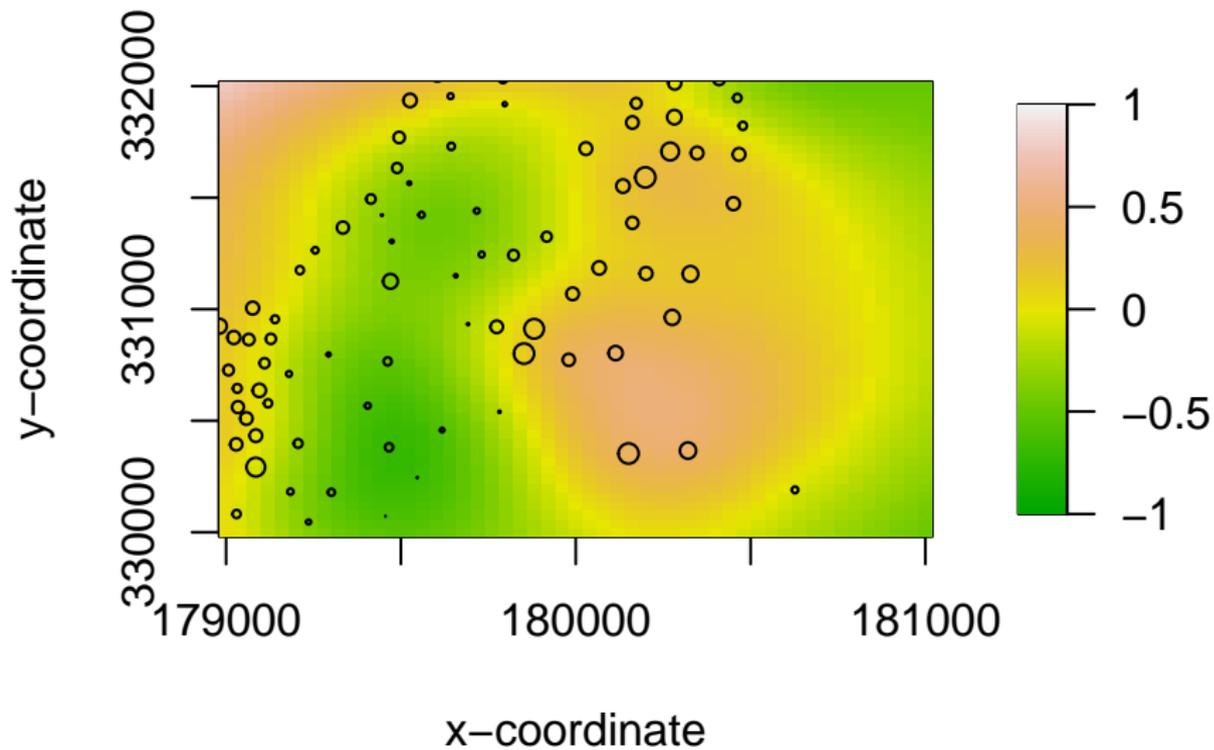
zinc.pred.gam <- predict.gam(gam.mod, pred.dat.gam,
  type = "terms")[, 3]

zinc.pred.gam.sd <- predict.gam(gam.mod, se.fit = T,
  pred.dat.gam, type = "terms")[[2]][, 3]
```

Prediction from GAM

```
library(fields)
image.plot(x = pred.grid.gam[["Var1"]][1:51], y = unique(pred.grid.gam[["Var2"]]),
  z = matrix(zinc.pred.gam, nrow = 51, ncol = 51),
  col = terrain.colors(100), xlab = "x-coordinate",
  ylab = "y-coordinate", breaks = seq(-1, 1, , 101),
  axis.args = list(at = c(-1, -0.5, 0, 0.5, 1), labels = c("-1",
    "-0.5", "0", "0.5", "1"))))

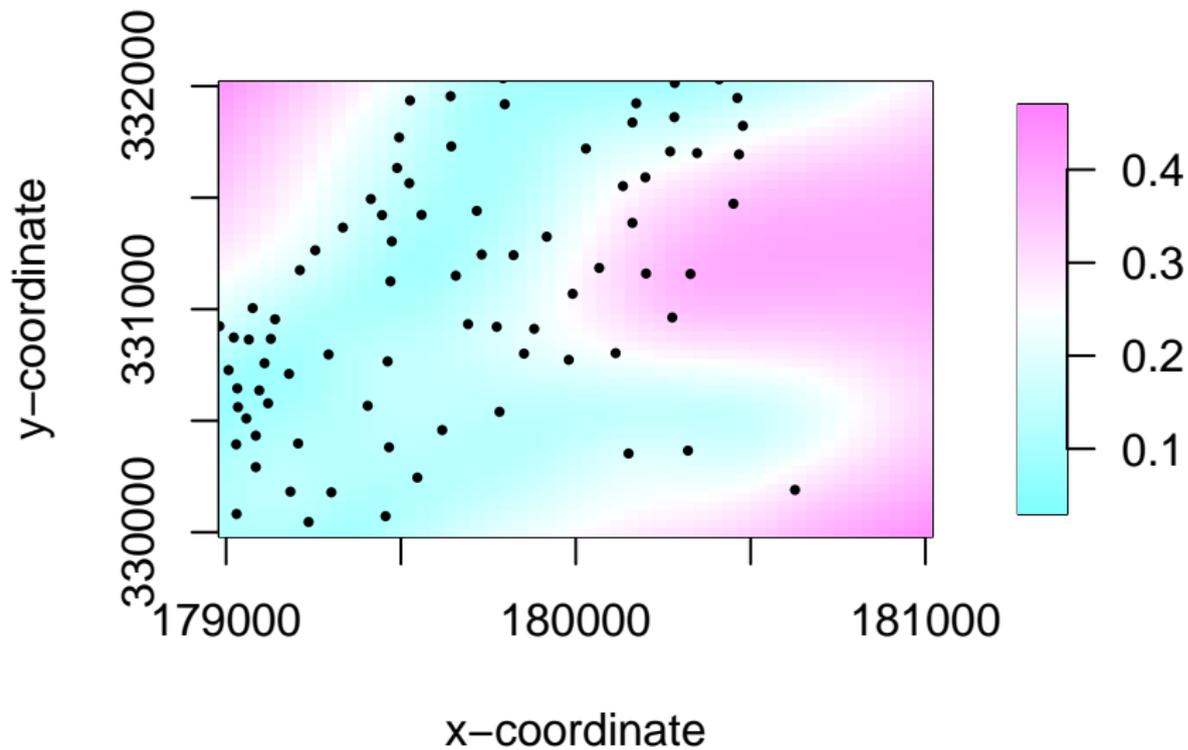
symbols(detrend$coords[, 1], detrend$coords[, 2], circles = (detrend$data -
  min(detrend$data))/1, add = T, inches = 0.04)
```



Standard deviations of prediction from GAM

```
image.plot(x = pred.grid[["Var1"]][1:51], y = unique(pred.grid[["Var2"]]),
  z = matrix(zinc.pred.gam.sd, nrow = 51, ncol = 51),
  col = cm.colors(100), xlab = "x-coordinate", ylab = "y-coordinate",
  breaks = seq(0.03, 0.47, , 101), axis.args = list(at = c(0.1,
    0.2, 0.3, 0.4), labels = c("0.1", "0.2", "0.3",
    "0.4"))))

points(detrend$coords[, 1], detrend$coords[, 2], cex = 0.5,
  pch = 16)
```



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(geostatsp)
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
zincglm <- glm(formula, grid = 30, shape = 1,
  family = "gaussian", buffer = 200, data = data.spdf,
  priorCI = list(c(sd = 0.1, 1), range = c(100,
  1200)))
```

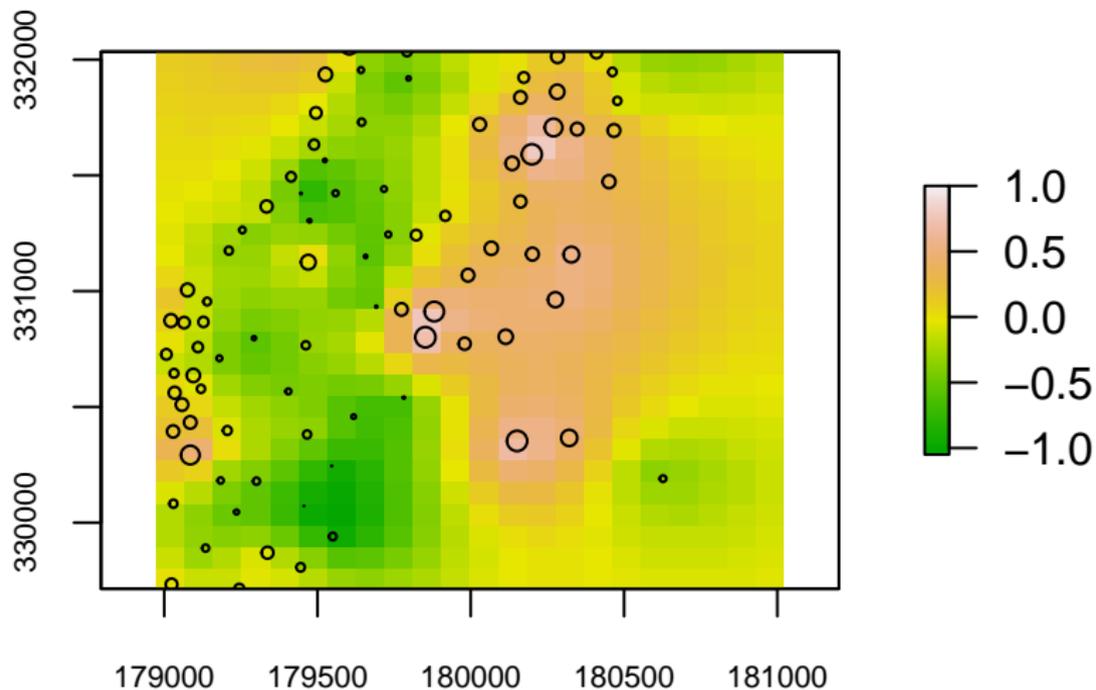
Quick Bayesian analysis (Brown 2014)

```
zincglm$parameters$summary
```

```
##                mean          sd  0.025quant    0.5quant    0.975quant
## (Intercept)  8.7473087 2.655045e-01  8.2294118  8.7457141  9.2738204
## elev        -0.2799555 2.981336e-02 -0.3387179 -0.2799207 -0.2214535
## dist        -2.2056692 3.582847e-01 -2.9244836 -2.2014908 -1.5105945
## range       678.6255708 1.732170e+02 423.4119954 649.1208111 1096.6545165
## sdNugget     0.1837876 1.163287e-03  0.1375532  0.1827611  0.2442996
## sd           0.4594454 6.911672e-03  0.3475434  0.4519217  0.6204072
##                mode          kld      meanExp
## (Intercept)  8.7426958 2.015934e-13 6448.4837657
## elev        -0.2798503 3.903356e-13  0.7576351
## dist        -2.1932857 8.032970e-12  0.1156171
## range       591.4016084          NA          NA
## sdNugget     0.1903354          NA          NA
## sd           0.4645550          NA          NA
```

Bayesian Zinc Prediction

```
zinc.bayes.pred <- zincglm$raster$random.mean
pred.ext <- extent(179000, 181000, 310000,
  332000)
zinc.bayes.pred <- crop(zinc.bayes.pred,
  pred.ext)
plot(zinc.bayes.pred, xlim = c(179000, 181000),
  col = terrain.colors(100), asp = 0.75,
  breaks = seq(-1.05, 1, , 101), axis.args = list(at = c(-1,
  -0.5, 0, 0.5, 1)))
p.xy <- detrend$coords
p.data <- detrend$data
p.index <- which(p.xy[, 1] > 179000)
symbols(detrend$coords[p.index, 1], detrend$coords[p.index,
  2], circles = (detrend$data[p.index] -
  min(detrend$data[p.index]))/1, add = T,
  inches = 0.04)
```



Bayesian Zinc Prediction (SD)

```
zinc.bayes.sd <- zincglm$raster$random.sd
pred.ext <- extent(179000, 181000, 310000,
  332000)
zinc.bayes.sd <- crop(zinc.bayes.sd, pred.ext)
plot(zinc.bayes.sd, xlim = c(179000, 181000),
  col = cm.colors(100), asp = 0.75, breaks = seq(0.03,
  0.47, , 101), axis.args = list(at = c(0.1,
  0.2, 0.3, 0.4)), cex.axis = 0.7,
  cex.lab = 0.7)
points(detrend$coords[p.index, 1], detrend$coords[p.index,
  2], cex = 0.5, pch = 16)
```

