2023 554 R Notes on Mapping for Point Data

Jon Wakefield
Departments of Biostatistics and Statistics
University of Washington

2023-03-07

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 modeling a spatially-indexed continuous response via:

- Conventional Kriging via MLE and variants
- A generalized additive model (GAM)
- A Bayesian approach using stochastic partial differential equations (SPDE)

Continuous Response: Motivating 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 $\times$ 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.

Meuse analysis using `geostat` functions

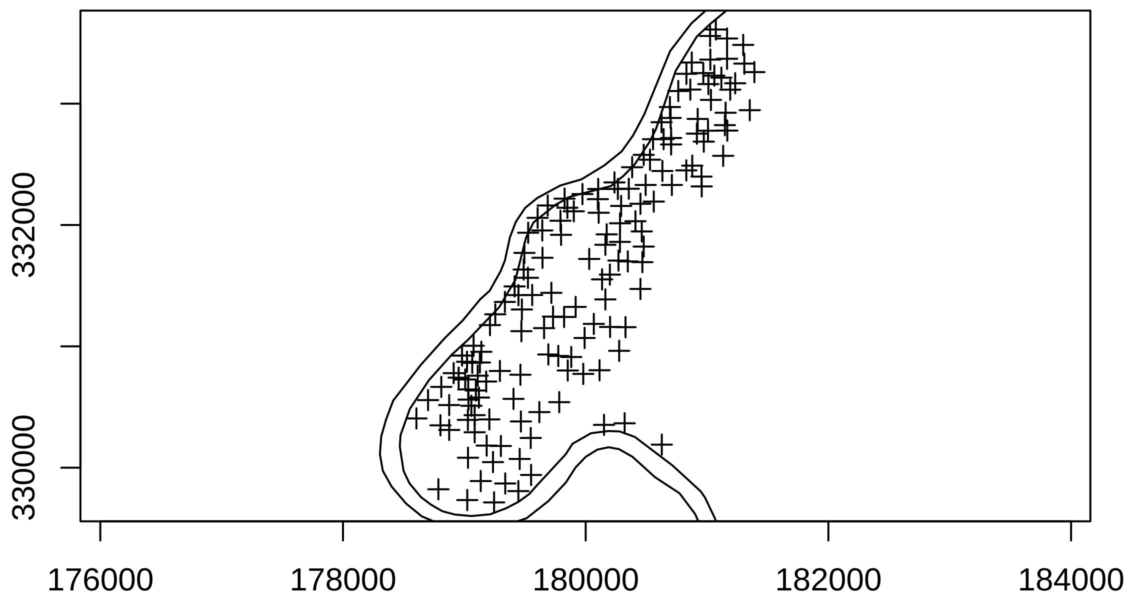
We look at the sampling locations and then examine variograms.

```
library(tidyverse)
library(ggpubr)
library(viridis)
library(geoR)
data("meuse")
library(sp)
library(INLA)

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)
```



Exploratory analysis

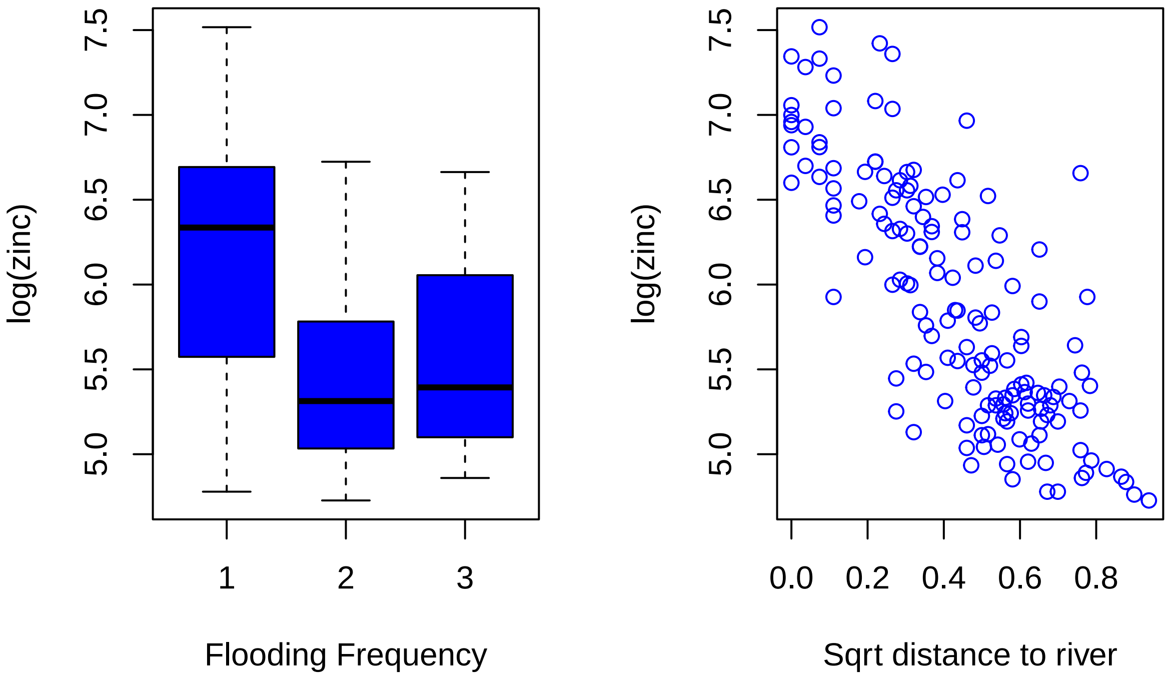
We work with log(zinc) as the distribution is more symmetric than on the original scale, and the variance more constant across levels of covariates.

It's often a good idea to do some exploratory data analysis (EDA), so let's see how log(zinc) varies by two possible covariates:

- Flooding frequency (`ffreq`); 1 = once in two years; 2 = once in ten years; 3 = one in 50 years
- Distance to the Meuse river (`dist`); normalized to $[0, 1]$

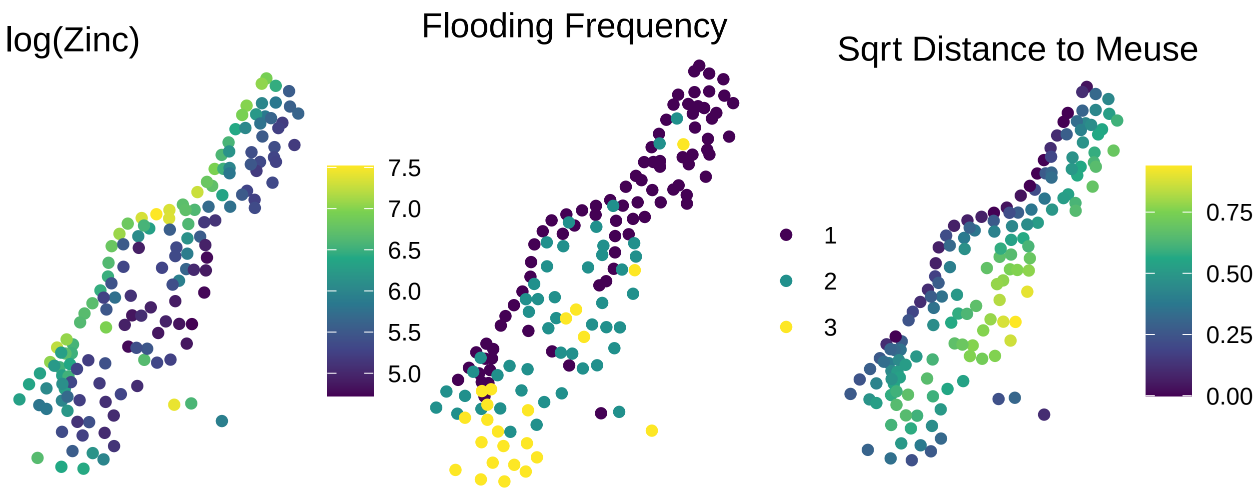
We focus on these, since they are available across the study region and so can be used for prediction. Following previous authors, we take `sqrt(dist)` which is closer to linearity.

```
par(mfrow = c(1, 2))
plot(log(meuse$zinc) ~ meuse$ffreq, ylab = "log(zinc)",
    xlab = "Flooding Frequency", col = "blue")
plot(log(meuse$zinc) ~ sqrt(meuse$dist),
    ylab = "log(zinc)", xlab = "Sqrt distance to river",
    col = "blue")
```



Also map log(zinc) and these covariates.

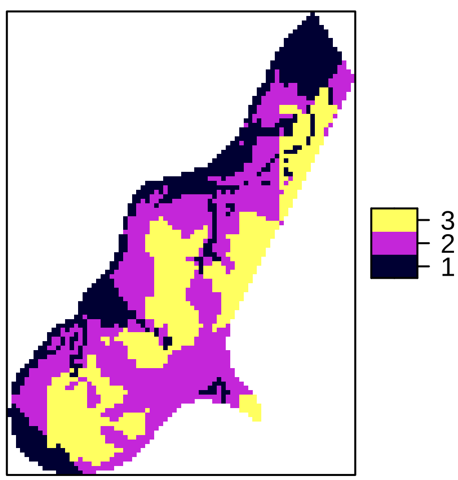
```
m.sf <- sf::st_as_sf(meuse, coords = c("x",
    "y"))
m.sf$logzinc <- log(m.sf$zinc)
m.sf$sdist <- sqrt(m.sf$dist)

a <- ggplot() + geom_sf(data = m.sf[, "logzinc"],
    aes(color = logzinc)) + theme_void() +
    scale_color_viridis_c() + labs(title = "log(Zinc)",
    color = NULL)
b <- ggplot() + geom_sf(data = m.sf[, "ffreq"],
    aes(color = ffreq)) + theme_void() +
    scale_color_viridis(discrete = T) + labs(title = "Flooding Frequency",
    color = NULL)
c <- ggplot() + geom_sf(data = m.sf[, "sdist"],
    aes(color = sdist)) + theme_void() +
    scale_color_viridis_c() + labs(title = "Sqrt Distance to Meuse",
    color = NULL)
ggpubr::ggarrange(a, b, c, nrow = 1, ncol = 3)
```



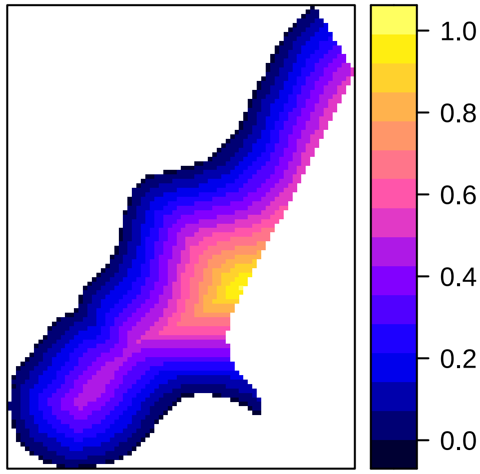
These two covariates are also available for a grid, in the `meuse.grid` data object. We will make use of this when we get to making predictions. Load in `meuse.grid` and take a look at the grid covariates.

```
library(sp)
data(meuse.grid)
coordinates(meuse.grid) = ~x + y
proj4string(meuse.grid) <- CRS("+init=epsg:28992")
gridded(meuse.grid) = TRUE
```

```
spplot(meuse.grid["ffreq"])
```

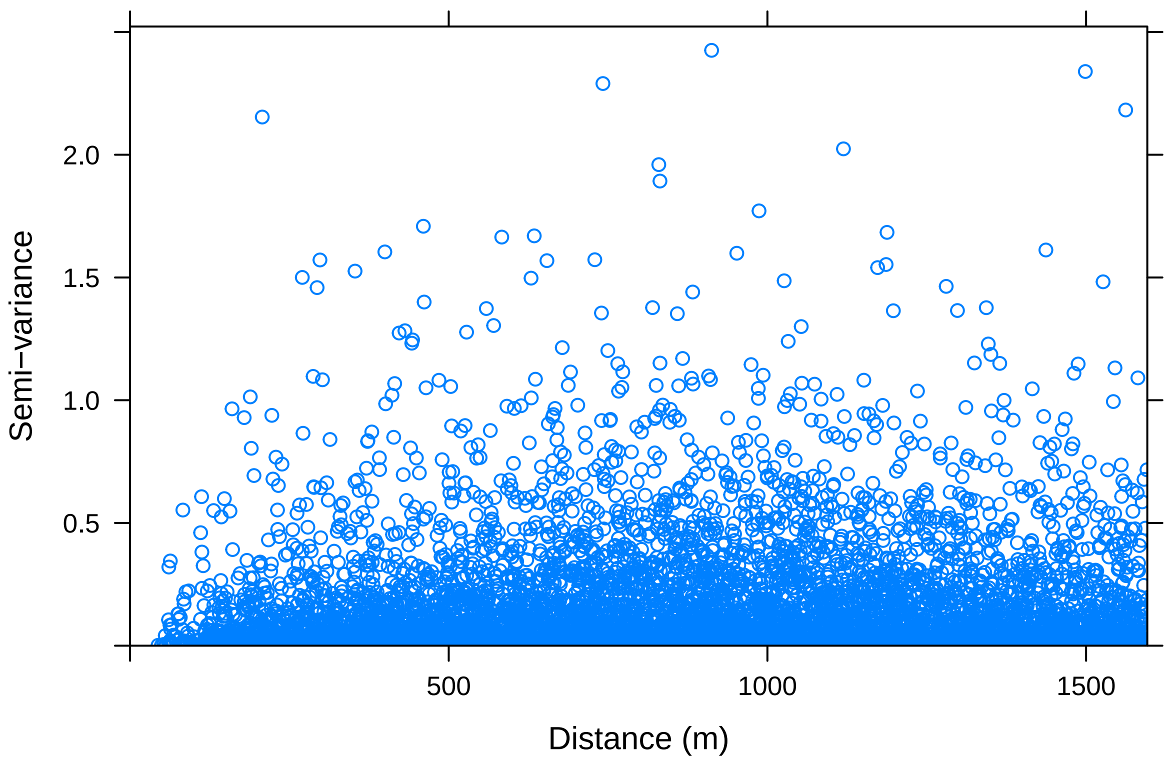



```
spplot(meuse.grid["dist"])
```



log(zinc): Variogram cloud, trend removed

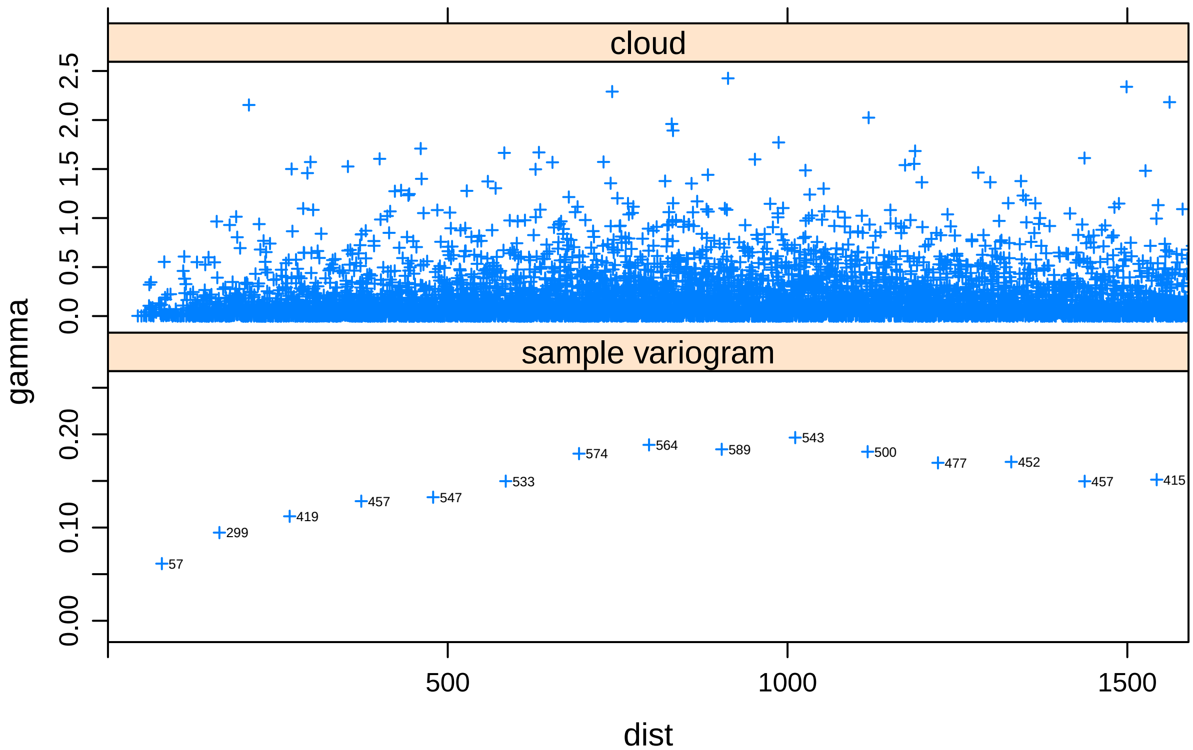
```
library(gstat)
cld <- variogram(log(zinc) ~ sqrt(meuse$dist) +
    as.factor(meuse$ffreq), meuse, cloud = TRUE)
plot(cld, ylab = "Semi-variance", xlab = "Distance (m)")
```



More variograms, with sample sizes

```
library(lattice)
cld <- variogram(log(zinc) ~ sqrt(meuse$dist)+as.factor(meuse$ffreq), meuse, cloud = TRUE)
svgm <- variogram(log(zinc) ~ sqrt(meuse$dist)+as.factor(meuse$ffreq), 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(-.005,0.25)))),
      limits = list(NULL, c(-.005,0.25)))),
    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
)
```

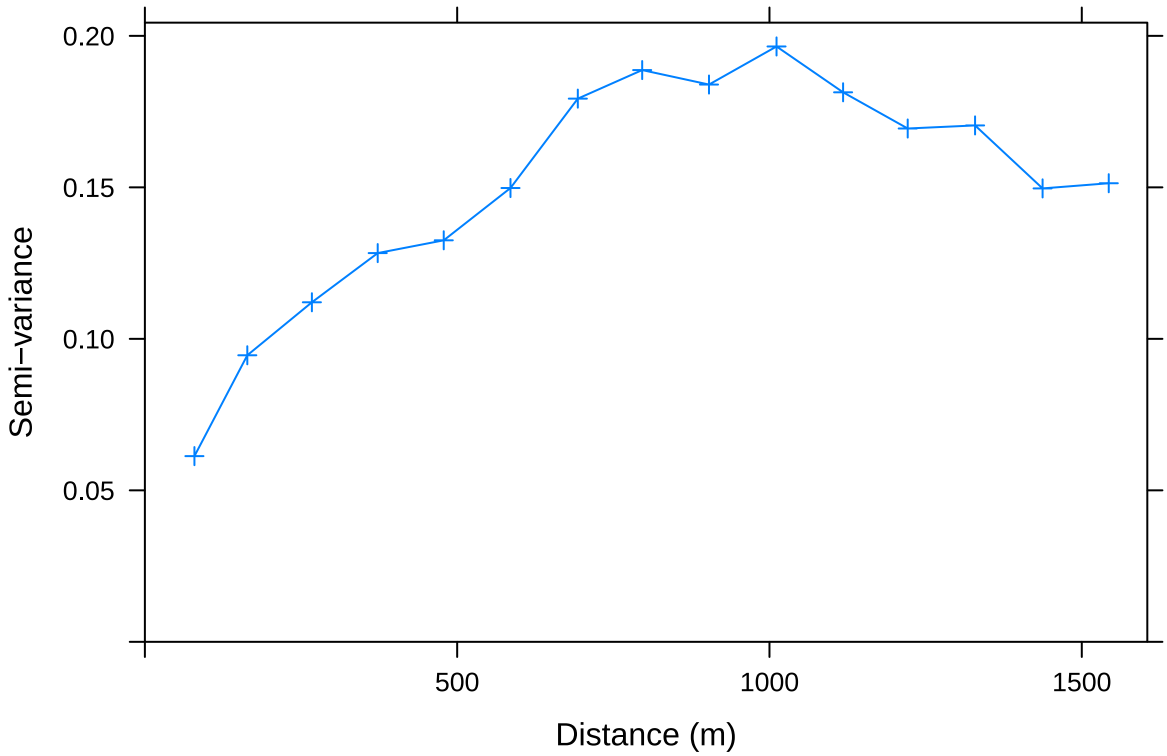


Monte Carlo simulations of semi-variogram

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

```
v <- variogram(log(zinc) ~ sqrt(meuse$dist) +
    as.factor(meuse$ffreq), 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 envelopes under no spatial dependence - it is clear there is dependence here.

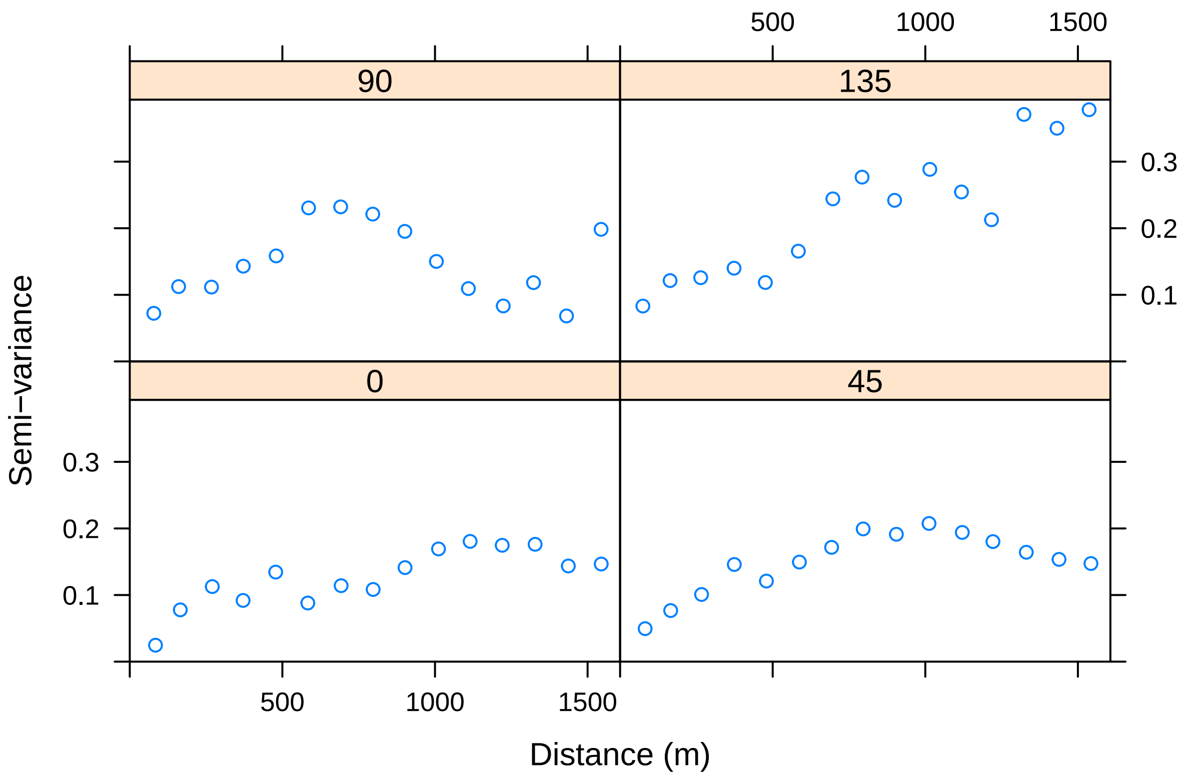
Directional variogram with linear trend removed

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.

```
dircld <- variogram(log(zinc) ~ sqrt(meuse$dist) +
    as.factor(ffreq), meuse, alpha = c(0,
    45, 90, 135))
```

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



Other capabilities in gstat

See

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

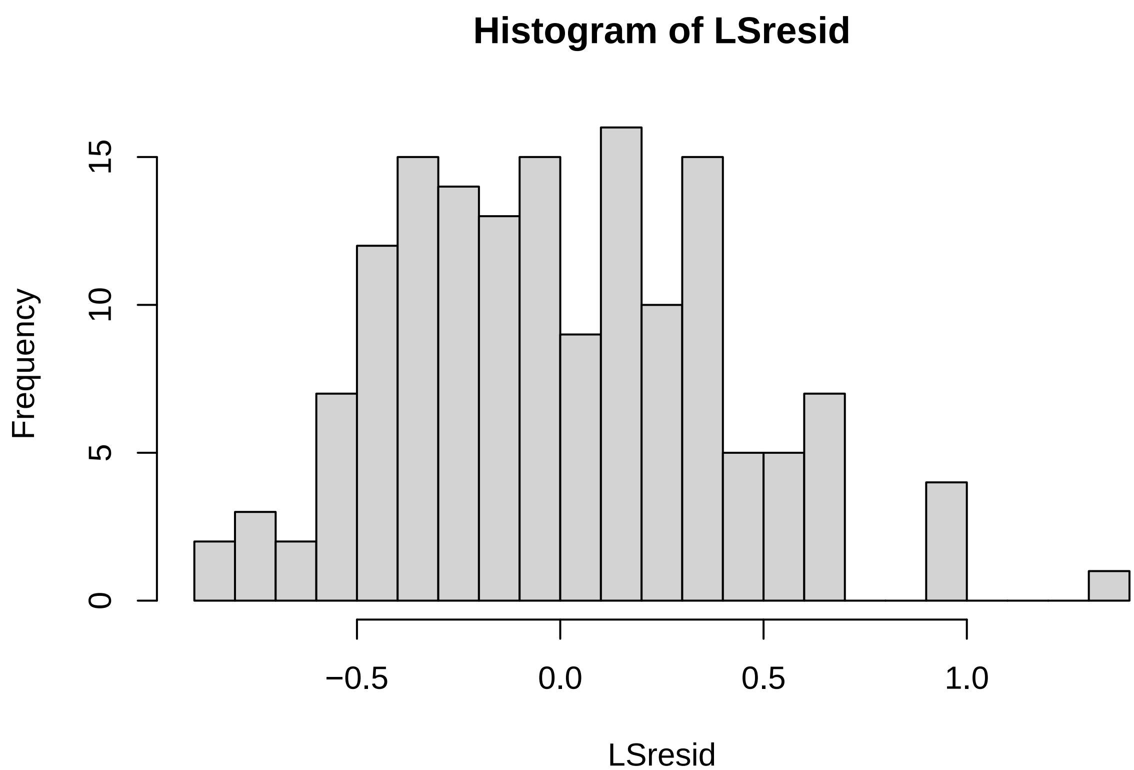
`geoR` for geostatistics

We continue the analysis using functions from the `geoR` library, for which a `geodata` data type is required. There are 155 observations (sampling locations)

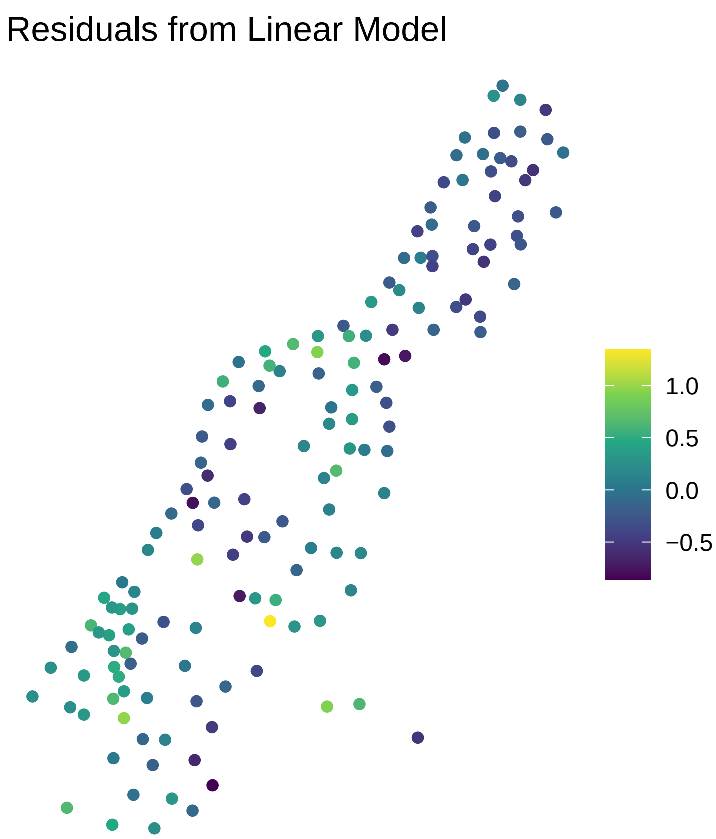
```
zmat <- matrix(cbind(meuse$x, meuse$y, log(meuse$zinc)),
    ncol = 3, nrow = 155, byrow = F)
geozinc <- as.geodata(zmat, coords.col = c(1,
    2), data.col = c(3))
```

First we will be assuming a spatial model on the residuals with `ffreq` and `sqrt(dist)` in the model. Fit this initial linear model, and view the residuals by histogram and map.

```
LSmod <- lm(log(meuse$zinc) ~ sqrt(meuse$dist) +
    as.factor(meuse$ffreq))
LSresid <- residuals(LSmod, type = "pearson")
hist(LSresid, nclass = 25)
```




```
m.sf$resid <- LSresid
ggplot() + geom_sf(data = m.sf[, "resid"],
    aes(color = resid)) + theme_void() +
    scale_color_viridis_c() + labs(title = "Residuals from Linear Model",
    color = NULL)
```



Moran's I

We can calculate Moran's I to see the strength of spatial dependence in the residuals. Our results show that we can reject the null hypothesis that there is zero spatial autocorrelation present in `LSresid` at the $\alpha = 0.05$ level.

```
library(ape)
dists <- as.matrix(dist(cbind(geozinc[1]$coords[,
    1], geozinc[1]$coords[, 2])))
dists.inv <- 1/dists
diag(dists.inv) <- 0
dists.inv[1:5, 1:5]
##             1           2           3           4           5
## 1 0.000000000 0.014116748 0.008414063 0.003857440 0.002729898
## 2 0.014116748 0.000000000 0.007063831 0.003535423 0.002757553
## 3 0.008414063 0.007063831 0.000000000 0.006984644 0.003983684
## 4 0.003857440 0.003535423 0.006984644 0.000000000 0.006482446
## 5 0.002729898 0.002757553 0.003983684 0.006482446 0.000000000

Moran.I(LSresid, dists.inv)
## $observed
## [1] 0.1210698
##
## $expected
## [1] -0.006493506
##
## $sd
## [1] 0.01060145
```

```
##
## $p.value
## [1] 0
```

Maximum likelihood for log(zinc)

We fit with the Matern covariance model with

$$\rho(h) = \frac{1}{2^{\kappa-1}\Gamma(\kappa)} \left(\frac{h}{\phi}\right)^{\kappa} K_{\kappa}\left(\frac{h}{\phi}\right),$$

where h is the distance between 2 points and we take $\kappa = 0.5$ so that ϕ is the only estimated parameter. The other parameters estimated are τ^2, the nugget, and σ^2, the spatial variance. The practical range reported is the distance at which the correlations drop to 0.05, and is a function of ϕ.

We suppress the output from the call.

```
mlfit <- likfit(geozinc, cov.model = "matern",
    ini = c(0.2, 224), trend = ~sqrt(meuse$dist) +
        as.factor(meuse$ffreq))
```

We examine the results, specifically point estimates and standard errors.

```
mlfit$parameters.summary
##            status   values
## beta0   estimated   7.0712
## beta1   estimated  -2.1208
## beta2   estimated  -0.5154
## beta3   estimated  -0.5266
## tausq   estimated   0.0372
## sigmasq estimated   0.1316
## phi     estimated 300.5381
## kappa       fixed   0.5000
## psiA        fixed   0.0000
## psiR        fixed   1.0000
## lambda      fixed   1.0000
for (i in 1:3) {
    cat(cbind(mlfit$beta[i], sqrt(mlfit$beta.var[i,
        i])), "\n")
}
## 7.071249 0.1326244
## -2.120796 0.2365602
## -0.5153624 0.06780136
mlfit$practicalRange
## [1] 900.3318
```

Note that the standard errors change from the least squares fit.

Restricted maximum likelihood for log(zinc)

Restricted MLE is preferable in general for estimation when there are variance parameters.

```
remlfit <- likfit(geozinc, cov.model = "matern",
    ini = c(0.55, 224), lik.method = "RML",
    trend = ~sqrt(meuse$dist) + as.factor(meuse$ffreq))
```

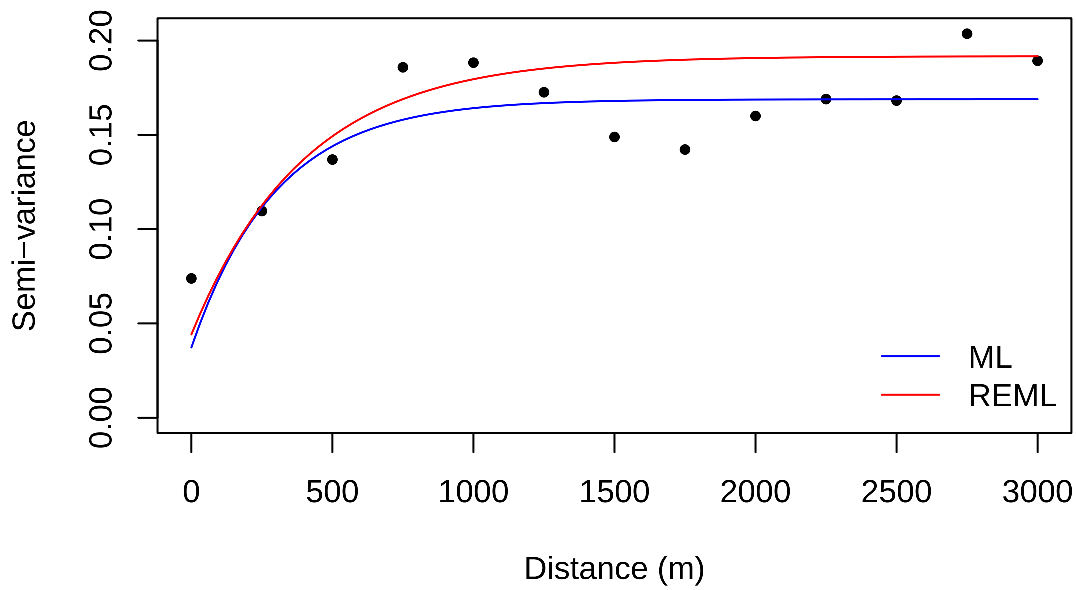
The results show slight differences from ML.

```r
remlfit$parameters.summary
##            status   values
## beta0   estimated   7.0823
## beta1   estimated  -2.1109
## beta2   estimated  -0.5280
## beta3   estimated  -0.5455
## tausq   estimated   0.0442
## sigmasq estimated   0.1476
## phi     estimated 401.4313
## kappa       fixed   0.5000
## psiA        fixed   0.0000
## psiR        fixed   1.0000
## lambda      fixed   1.0000
remlfit$practicalRange
## [1] 1202.581
```

```r
for (i in 1:3) {
    cat(cbind(mlfit$beta[i], sqrt(mlfit$beta.var[i,
        i]), remlfit$beta[i], sqrt(remlfit$beta.var[i,
        i])), "\n")
}
## 7.071249 0.1326244 7.082279 0.1527493
## -2.120796 0.2365602 -2.110917 0.2515499
## -0.5153624 0.06780136 -0.5280202 0.06913553
```

Comparison of estimates

```r
binzinc <- variog(geozinc, uvec = seq(0,
    5000, 250), trend = ~sqrt(meuse$dist) +
    as.factor(meuse$ffreq))
## variog: computing omnidirectional variogram

plot(binzinc, max.dist = 3000, xlab = "Distance (m)",
    ylab = "Semi-variance", pch = 19, cex = 0.6)
lines(mlfit, max.dist = 3000, lty = 1, col = "blue")
lines(remlfit, max.dist = 3000, lty = 1,
    col = "red")
legend("bottomright", legend = c("ML", "REML"),
    lty = c(1, 1), bty = "n", col = c("blue",
        "red"), cex = 1)
```



Prediction for log(zinc) by Kriging

We now fit a linear model in distance and elevation to log(zinc). We then form a `geodata` object with the residuals as the response.

```
lmdata <- data.frame(logzinc = geozinc$data,
    sqrtdist = sqrt(meuse$dist), ffreq = as.factor(meuse$ffreq))
lmfit <- lm(logzinc ~ sqrtdist + ffreq, data = lmdata)
lmfit
##
## Call:
## lm(formula = logzinc ~ sqrtdist + ffreq, data = lmdata)
##
## Coefficients:
## (Intercept)     sqrtdist       ffreq2       ffreq3
##      7.0299      -2.2660      -0.3605      -0.3167
detrend <- as.geodata(cbind(geozinc$coords,
    lmfit$residuals))

likfit(geozinc, ini = c(1000, 50))
## ---------------------------------------------------------------
## 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.
## likfit: estimated model parameters:
##        beta       tausq     sigmasq         phi
## "   6.6360" "   0.0347" "   1.8478" "2142.6170"
## Practical Range with cor=0.05 for asymptotic range: 6418.707
```

```
##
## likfit: maximised log-likelihood = -99.13
```

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

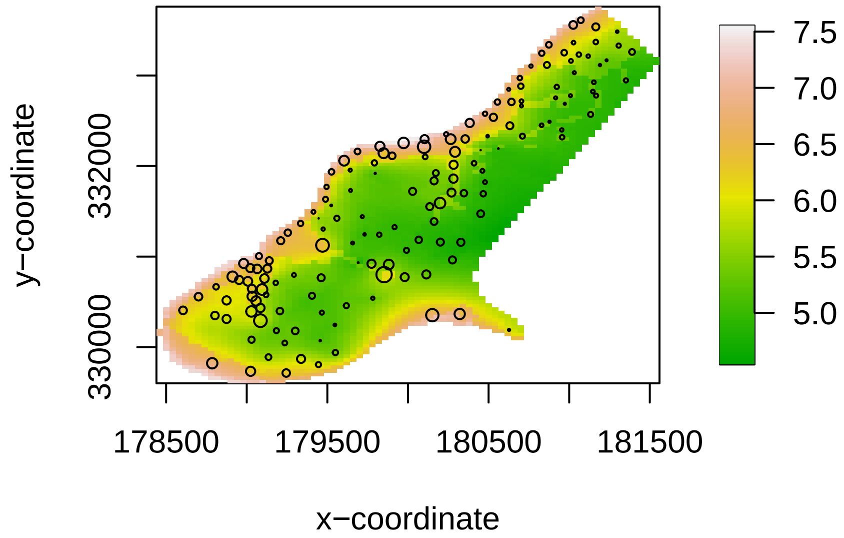
```
# get meuse.grid and expand to all
# combinations
data(meuse.grid)
pred.grid <- expand.grid(sort(unique(meuse.grid$x)),
    sort(unique(meuse.grid$y)))

# kriging
kc <- krige.conv(detrend, loc = pred.grid,
    krige = krige.control(obj.m = mlfit))
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

However, since we're predicting the residuals, to get back to the original task of predicting log(zinc), we can add this value back onto our predictions from `lmfit`, which yields the following:

```
# merge linear predictions onto grid
pred_lm <- meuse.grid[, c("x", "y", "dist",
    "ffreq")]
pred_lm$sqrtdist <- sqrt(pred_lm$dist)
pred_lm$pred_lm <- predict(lmfit, newdata = pred_lm)
pred_resid <- data.frame(pred_resid = kc$predict,
    x = pred.grid$Var1, y = pred.grid$Var2)
pred_logzinc <- merge(pred_resid, pred_lm,
    by = c("x", "y"), all = T)
pred_logzinc$pred_logzinc <- pred_logzinc$pred_lm +
    pred_logzinc$pred_resid
pred_logzinc <- pred_logzinc %>%
    arrange(y, x)
```

To view the results produce an image plot of the predictions, with the data superimposed.

```
library(fields)
image.plot(x = unique(pred.grid[["Var1"]]), y = unique(pred.grid[["Var2"]]),
    z = matrix(pred_logzinc$pred_logzinc, nrow = 78,
        ncol = 104), col = terrain.colors(100), xlab = "x-coordinate",
    ylab = "y-coordinate")
symbols(detrend$coords[, 1], detrend$coords[, 2], circles = (detrend$data -
    min(detrend$data))/1, add = T, inches = 0.04)
```

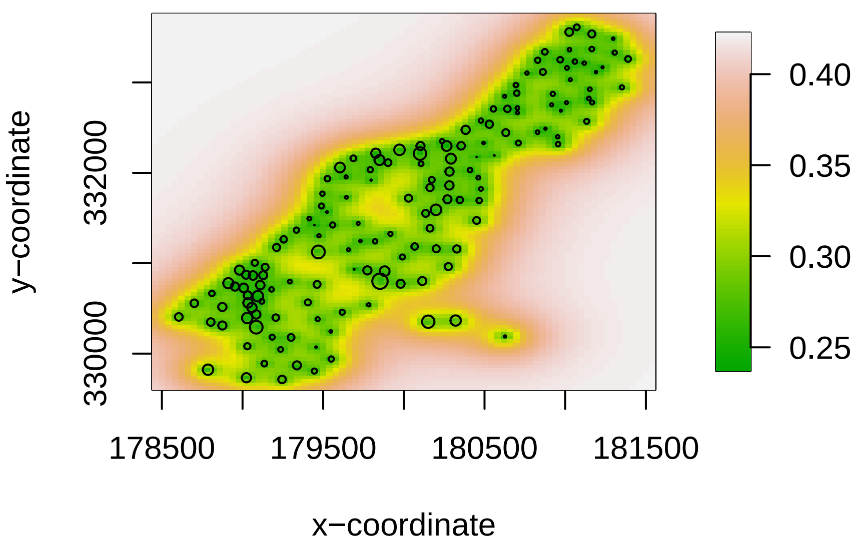



```
# library(sf) df <- st_as_sf(zinc.dat,
# coords=c('x', 'y')) ggplot() + geom_sf(data =
# df, col = 'black', fill = NA) + theme_bw() +
# coord_sf() + xlab('') + ylab('') +
# theme(legend.title = element_blank(),
# panel.grid = element_blank(), panel.border =
# element_blank(), axis.ticks = element_blank(),
# axis.text = element_blank())
```

Standard deviations of prediction for log(zinc)

We now plot the Kriging standard deviations of the predictions.

```
image.plot(x = unique(pred.grid[["Var1"]]), y = unique(pred.grid[["Var2"]]),
    z = matrix(sqrt(kc$krige.var), nrow = 78, ncol = 104),
    col = terrain.colors(100), xlab = "x-coordinate",
    ylab = "y-coordinate")
symbols(detrend$coords[, 1], detrend$coords[, 2], circles = (detrend$data -
    min(detrend$data))/1, add = T, inches = 0.04)
```



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

GAM model for log(zinc)

We now model the log(zinc) surface as linear in the square root of distance to the Meuse and flooding frequency, 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$zinc),
    sqrtdist = sqrt(meuse$dist), ffreq = as.factor(meuse$ffreq))
gam.mod <- gam(lzinc ~ s(x, y, bs = "tp") + sqrtdist +
    ffreq, data = zinc.dat, method = "REML")
```

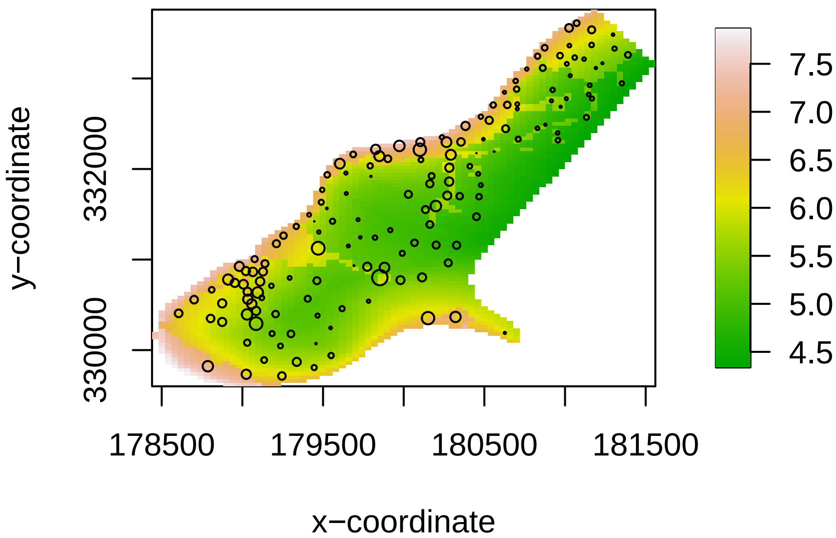
```
summary(gam.mod)
##
## Family: gaussian
## Link function: identity
##
## Formula:
## lzinc ~ s(x, y, bs = "tp") + sqrtdist + ffreq
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.9831     0.1522  45.869  < 2e-16 ***
## sqrtdist     -1.8917     0.3496  -5.411 2.87e-07 ***
## ffreq2       -0.5879     0.0721  -8.154 2.43e-13 ***
## ffreq3       -0.6241     0.1129  -5.530 1.66e-07 ***
## ---
## 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) 19.27  24.06 5.803  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.835   Deviance explained = 85.9%
## -REML = 58.733  Scale est. = 0.085927  n = 155
```

GAM prediction

```
pred.grid.gam <- expand.grid(sort(unique(meuse.grid$x)),
    sort(unique(meuse.grid$y)))
meuse.grid$sqrtdist <- sqrt(meuse.grid$dist)
meuse.grid$zinc.pred.gam <- predict.gam(gam.mod, meuse.grid)
meuse.grid$zinc.pred.gam.sd <- predict.gam(gam.mod,
    se.fit = T, meuse.grid)$se.fit
```

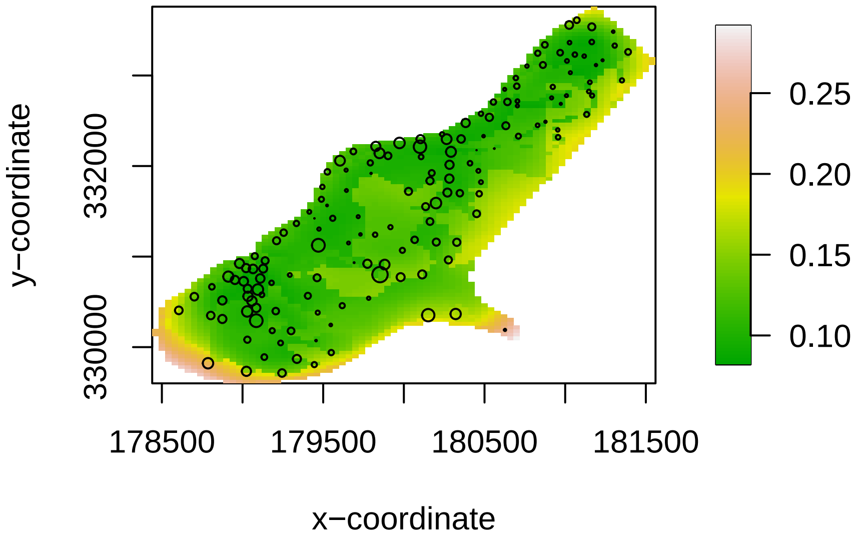
```
zinc.pred.gam <- xtabs(zinc.pred.gam ~ x + y, data = meuse.grid)
zinc.pred.gam[zinc.pred.gam == 0] <- NA
zinc.pred.gam.sd <- xtabs(zinc.pred.gam.sd ~ x + y,
    data = meuse.grid)
zinc.pred.gam.sd[zinc.pred.gam.sd == 0] <- NA
```

```
image.plot(x = pred.grid.gam[["Var1"]][1:78], y = unique(pred.grid.gam[["Var2"]]),
    z = matrix(zinc.pred.gam, nrow = 78, ncol = 104),
    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)
```



Standard deviations of prediction from GAM

```
image.plot(x = pred.grid.gam[["Var1"]][1:78], y = unique(pred.grid.gam[["Var2"]]),
    z = matrix(zinc.pred.gam.sd, nrow = 78, ncol = 104),
    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)
```



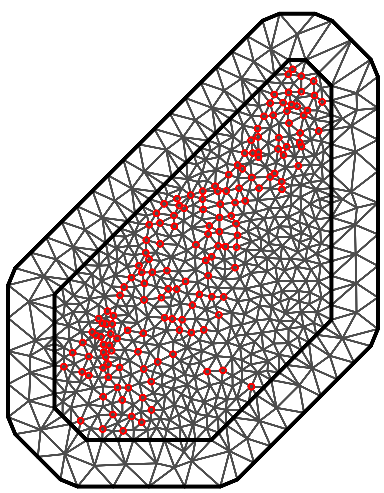
SPDE Model for log(zinc)

We now create a Bayesian SPDE model for log(zinc).

```
zincdf <- zinc.dat %>%
    dplyr::select(y = lzinc, locx = x, locy = y,
        sqrtdist = sqrtdist, ffreq = ffreq)

# Create mesh
max.edge <- 200
mesh <- inla.mesh.2d(loc = zincdf[, c("locx",
    "locy")], offset = c(100, 500), max.edge = c(max.edge,
    max.edge * 3))
```

We can visualize the mesh we created as follows:

```
plot(mesh, asp = 1, main = "")
points(zincdf[, c("locx", "locy")], col = "red",
    cex = 0.4)
```



Next, we create an A matrix and the stack, which we need in order to define our model.

```r
A <- inla.spde.make.A(mesh = mesh, loc = data.matrix(zincdf[, c("locx", "locy")]))

# Define X matrix
Xcov <- data.frame(intercept = 1,
                   sqrtdist = zincdf$sqrtdist)
Xcov <- as.matrix(Xcov)
colnames(Xcov)
## [1] "intercept" "sqrtdist"

# Create the stack
stack <- inla.stack(
  tag = "est",

  # - Name (nametag) of the stack
  # - Here: est for estimating
  data = list(y = zincdf$y),
  effects = list(
    # - The Model Components
    s = 1:mesh$n,
    Xcov = Xcov
  ),
  # - The second is all fixed effects
  A = list(A, 1)
  # - First projector matrix is for 's'
  # - second is for 'fixed effects'
)
```

Next, we can set a prior and define our model.

```r
# Define a prior
prior.median.sd <- 0.07
prior.median.range <- 2000
spde <- inla.spde2.pcmatern(mesh, alpha = 2,
    prior.range = c(prior.median.range, 0.5),
    prior.sigma = c(prior.median.sd, 0.5),
    constr = T)

# Define the model
formula <- y ~ -1 + Xcov + f(s, model = spde)
prior.median.gaus.sd <- 1  # Prior for measurement error
family <- "gaussian"
control.family <- list(hyper = list(prec = list(prior = "pc.prec",
    fixed = FALSE, param = c(prior.median.gaus.sd,
        0.5))))
```

Finally, we can fit our model:

```r
res <- inla(formula,
  data = inla.stack.data(stack, spde = spde),
  control.predictor = list(A = inla.stack.A(stack), compute = T),
  # compute=T to get posterior for fitted values
  family = family,
  control.family = control.family,
  # control.compute = list(config=T, dic=T, cpo=T, waic=T),
  # if Model comparisons wanted
```

```r
  control.inla = list(int.strategy = "eb"),
  # - faster computation
  # control.inla = list(int.strategy='grid'),
  # - More accurate integration over hyper-parameters
  verbose = F
)
summary(res)
##
## Call:
##    c("inla.core(formula = formula, family = family, contrasts = contrasts,
##    ", " data = data, quantiles = quantiles, E = E, offset = offset, ", "
##    scale = scale, weights = weights, Ntrials = Ntrials, strata = strata,
##    ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose =
##    verbose, ", " lincomb = lincomb, selection = selection, control.compute
##    = control.compute, ", " control.predictor = control.predictor,
##    control.family = control.family, ", " control.inla = control.inla,
##    control.fixed = control.fixed, ", " control.mode = control.mode,
##    control.expert = control.expert, ", " control.hazard = control.hazard,
##    control.lincomb = control.lincomb, ", " control.update =
##    control.update, control.lp.scale = control.lp.scale, ", "
##    control.pardiso = control.pardiso, only.hyperparam = only.hyperparam,
##    ", " inla.call = inla.call, inla.arg = inla.arg, num.threads =
##    num.threads, ", " blas.num.threads = blas.num.threads, keep = keep,
##    working.directory = working.directory, ", " silent = silent, inla.mode
##    = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame =
##    .parent.frame)")
## Time used:
##     Pre = 1.56, Running = 0.377, Post = 0.0179, Total = 1.96
## Fixed effects:
##         mean    sd 0.025quant 0.5quant 0.975quant   mode kld
## Xcov1  6.994 0.117      6.764    6.994      7.224  6.994   0
## Xcov2 -2.576 0.242     -3.050   -2.576     -2.102 -2.576   0
##
## Random effects:
##   Name     Model
##     s SPDE2 model
##
## Model hyperparameters:
##                                            mean      sd 0.025quant 0.5quant
## Precision for the Gaussian observations  12.387   3.203      7.314   11.974
## Range for s                             564.083 163.987    306.962  542.425
## Stdev for s                               0.328   0.055      0.233    0.323
##                                         0.975quant    mode
## Precision for the Gaussian observations     19.845  11.178
## Range for s                                947.579 501.801
## Stdev for s                                  0.449   0.314
##
## Marginal log-Likelihood:  -93.88
##  is computed
## Posterior summaries for the linear predictor and the fitted values are computed
## (Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```