2023 554 R Notes on Mapping for Point Data

Jon Wakefield
Departments of Biostatistics and Statistics
University of Washington

2023-03-09

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).

The model for Kriging and SPDE is

$$y(s_i) = \beta_0 + x_i^{\mathrm{T}}(s_i)\beta_1 + S(s_i) + \epsilon_i,$$

where s_i is the spatial location, $y(s_i)$, $x_i(s_i)$ are spatially indexed response and covariates, $S(s_i)$ is a Gaussian process (GP) random field with some covariance structure (we assume Matern) and $\epsilon_i \sim_{iid} \mathrm{N}(0, \sigma_\epsilon^2)$ is measurement error. The aim is prediction over a grid covering the study region, so that the covariates we use must be available not just at the data points but also at the grid locations. The measurement error is not included in the prediction.

The GAM replaces the GP with a local smoother, and we use a spline.

KP: Add one sentence about SPDE?

Continuous Response: Motivating Example

We illustrate methods for continuous data using Zinc levels close to the Meuse river 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. We will predict the levels of zinc over the study region.

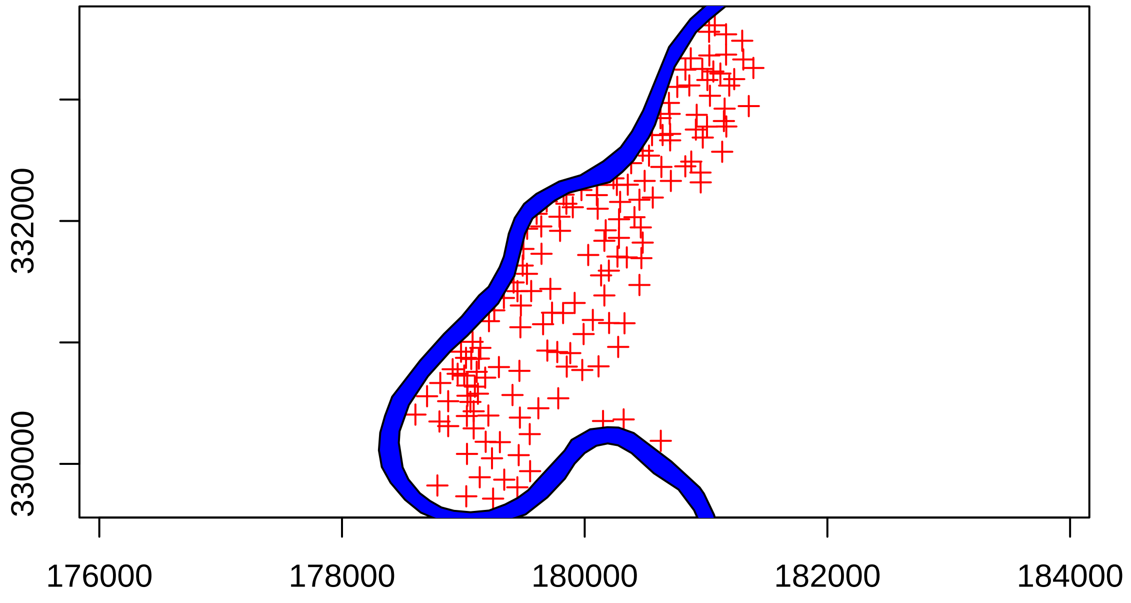
Meuse analysis using `geostat` functions

We look at the sampling locations and then examine variograms.

```r
library(tidyverse)
library(ggpubr)
library(viridis)
library(geoR)
library(sp)
library(sf)
library(INLA)
library(lattice)
library(kableExtra)

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

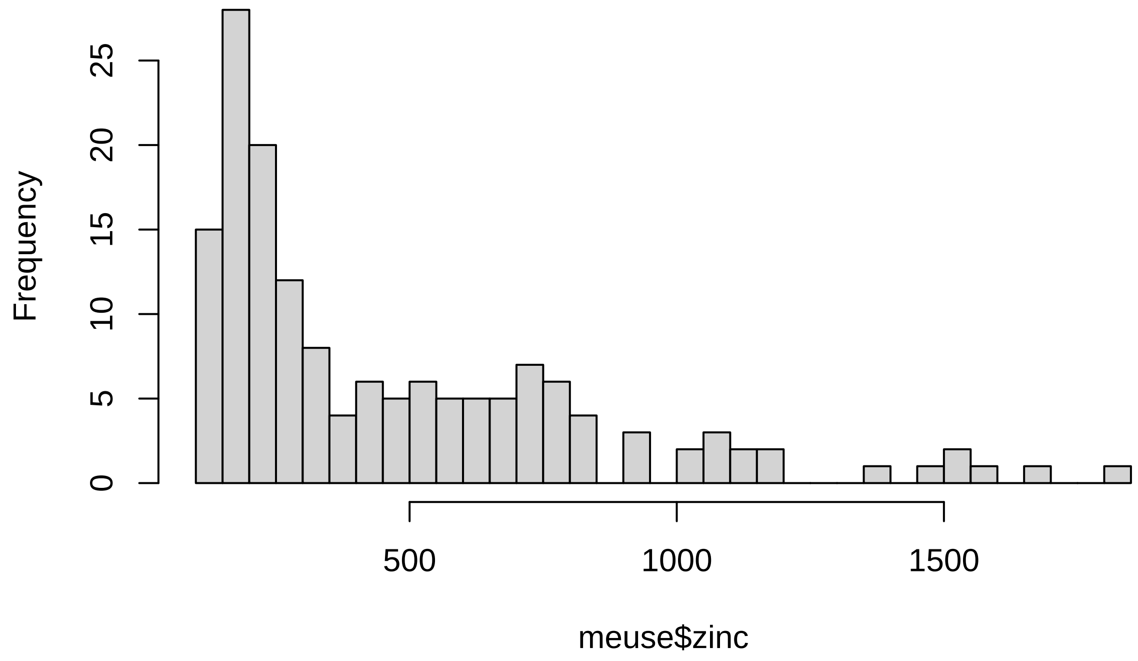
```r
plot(meuse1, axes = T, col = "red")
plot(rivers, add = T, col = "blue")
```



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 (the measurement error standard deviation, σ_ϵ, is assumed constant across the region).

```r
hist(meuse$zinc, nclass = 30, main = "")
```

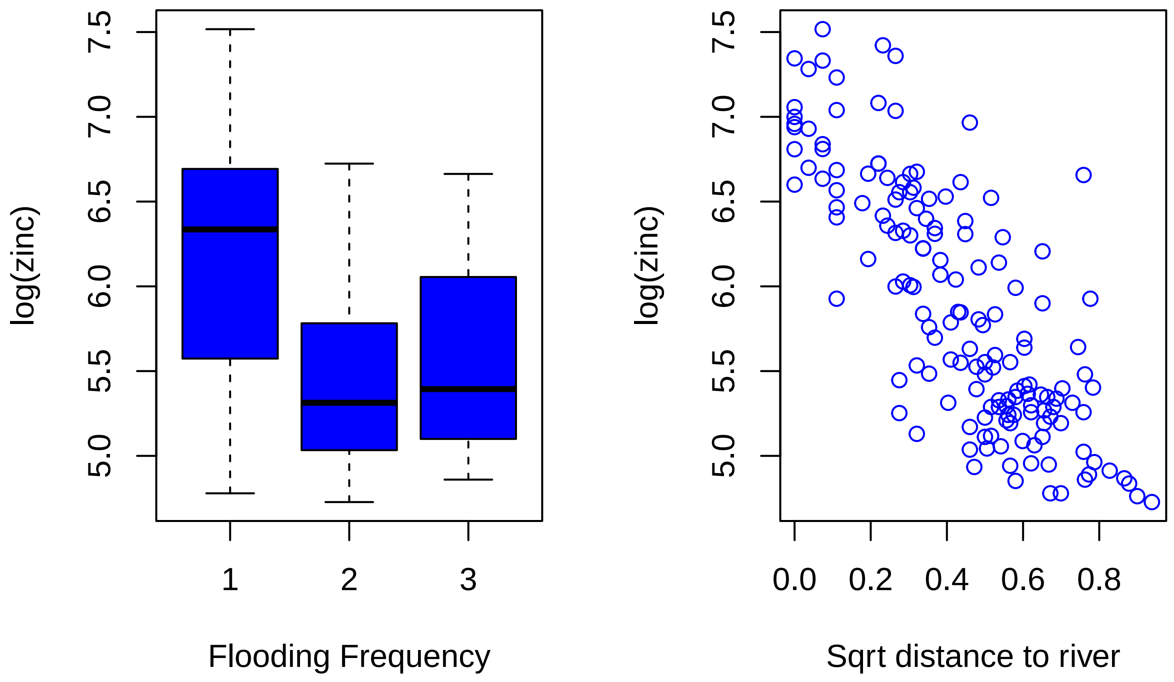


It is a good idea to do some exploratory data analysis (EDA), so let's see how log(zinc) varies by two possible covariates:

- 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 $[0, 1]$

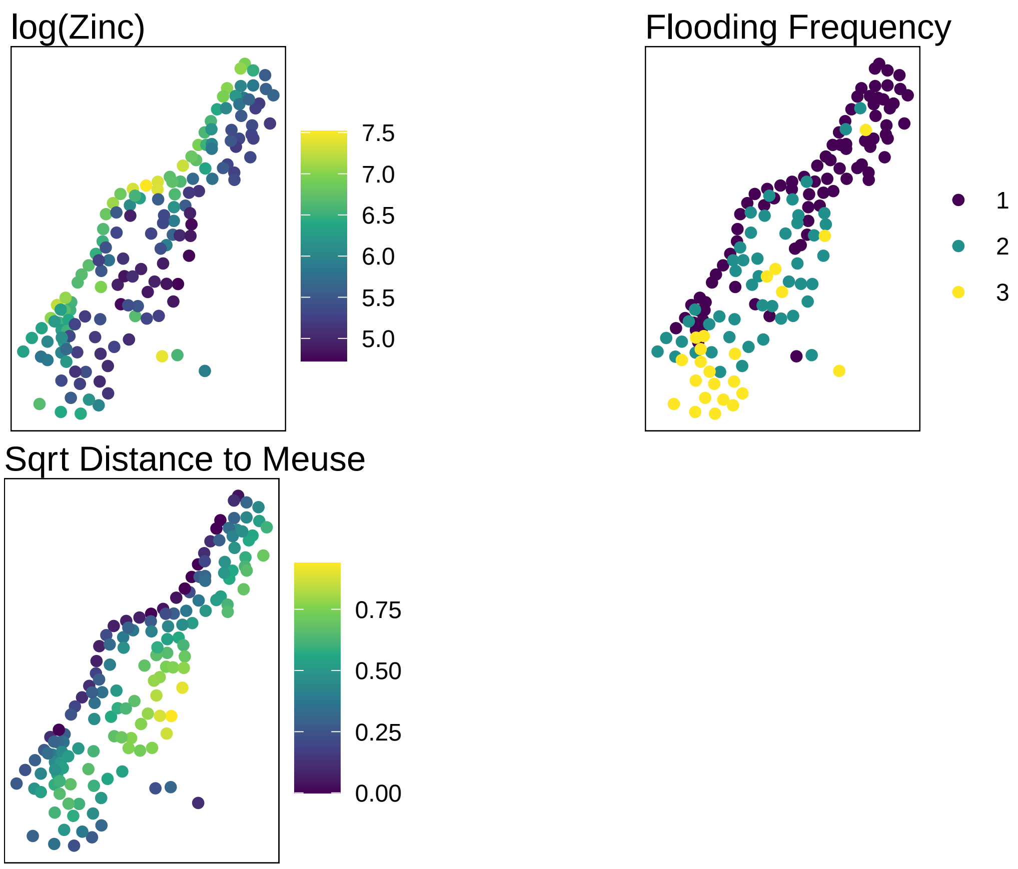
We focus on these, since they are available across the study region and so can be used for prediction. Following previous authors, we take `sqrt(dist)` because log(zinc) is more closely associated linearly with the transformed version.

```r
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.

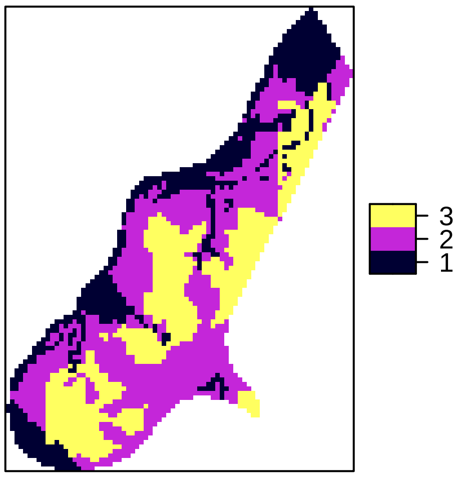
```
m.sf <- sf::st_as_sf(meuse, coords = c("x", "y"))
m.sf$logzinc <- log(m.sf$zinc)
m.sf$sdist <- sqrt(m.sf$dist)

a <- ggplot() + geom_sf(data = m.sf[, "logzinc"], aes(color = logzinc)) +
    theme_void() + scale_color_viridis_c() + labs(title = "log(Zinc)",
    color = NULL) + theme(panel.border = element_rect(fill = NA,
    color = "black"))
b <- ggplot() + geom_sf(data = m.sf[, "ffreq"], aes(color = ffreq)) +
    theme_void() + scale_color_viridis(discrete = T) +
    labs(title = "Flooding Frequency", color = NULL) +
    theme(panel.border = element_rect(fill = NA, color = "black"))
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) + theme(panel.border = element_rect(fill = NA,
    color = "black"))
ggpubr::ggarrange(a, b, c, nrow = 2, ncol = 2)
```



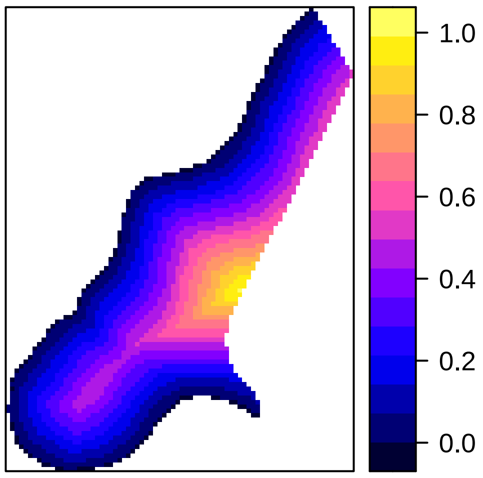
The grid data for these two covariates are available in the `meuse.grid` data object. We will make use of this when we get to making predictions. Load in `meuse.grid` and take a look at the grid covariates.

```
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"])
```



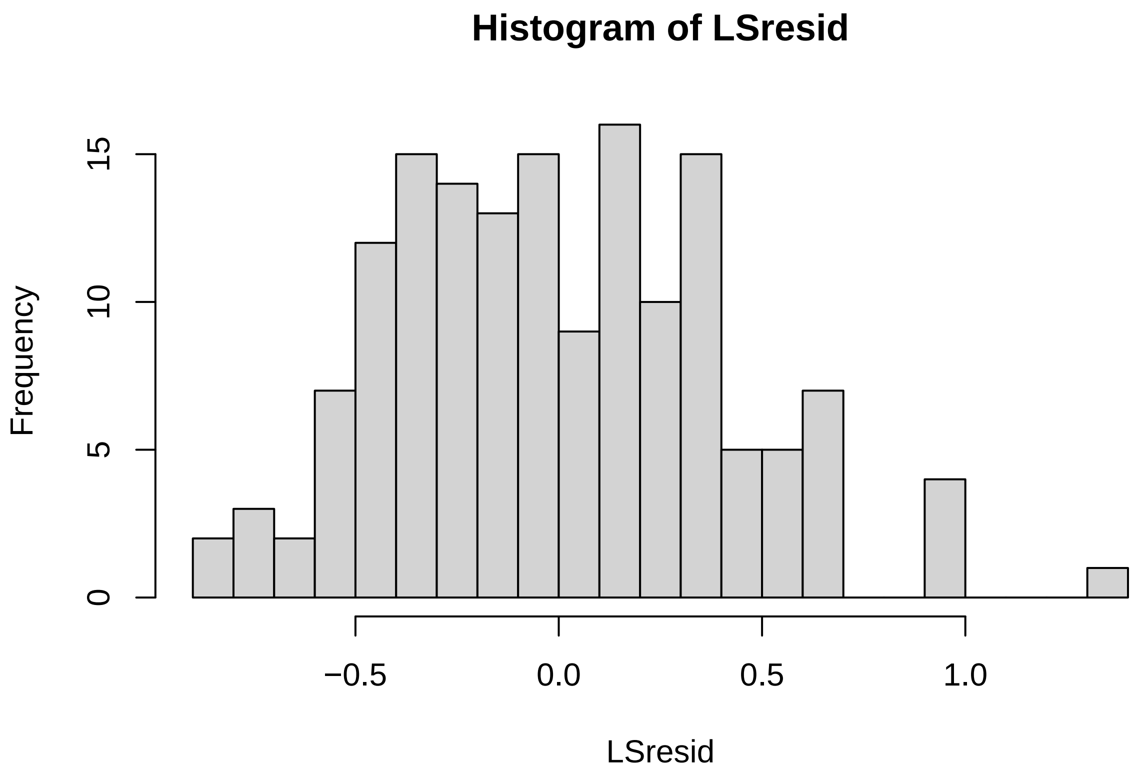
```
spplot(meuse.grid["dist"])
```



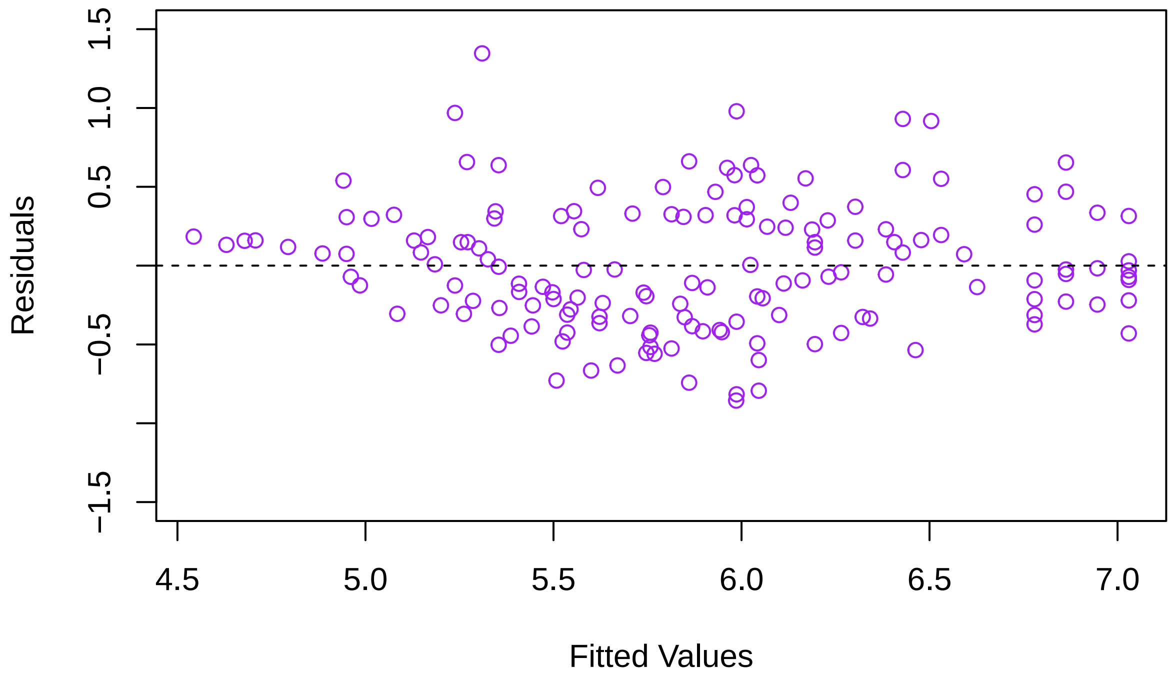
First we will be assuming a spatial model on the residuals with `ffreq` and `sqrt(dist)` in the model. Fit this initial linear model, and view the residuals by histogram and map.

Also plot residuals against fitted values - spread should be constant and looks OK.

```
LSmod <- lm(log(meuse$zinc) ~ sqrt(meuse$dist) + as.factor(meuse$ffreq))
LSresid <- residuals(LSmod, type = "pearson")
hist(LSresid, nclass = 22)
```



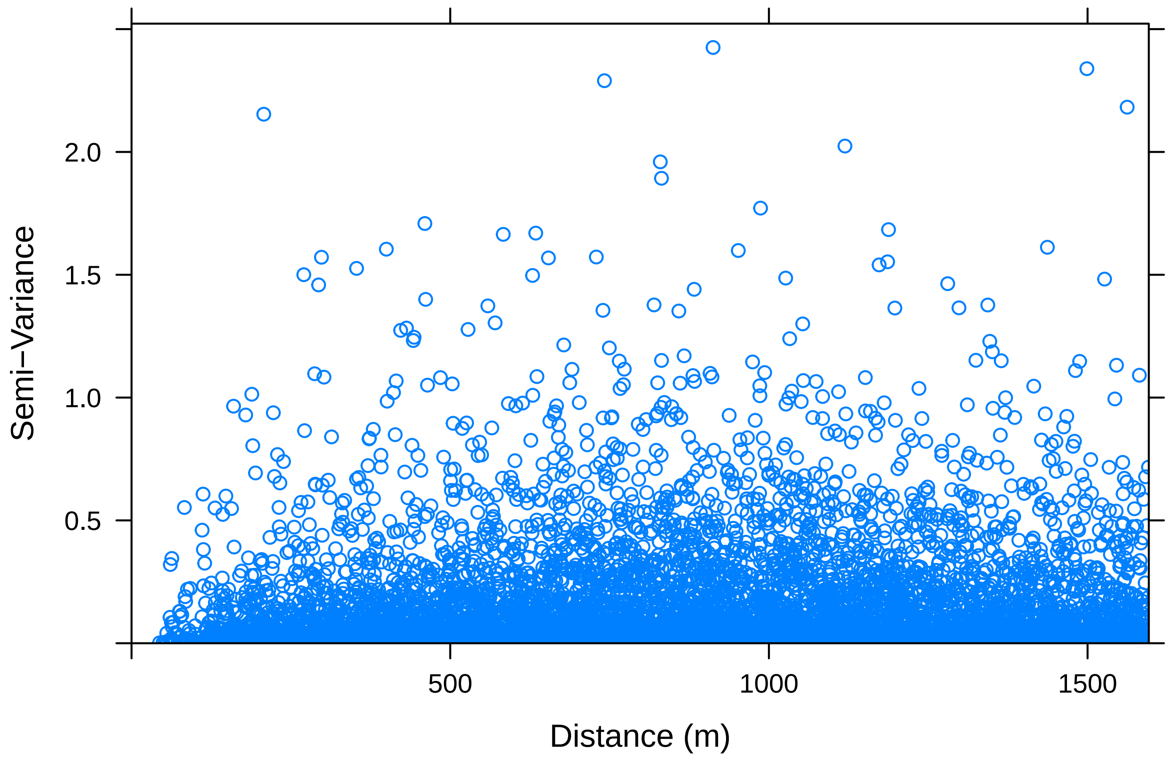
```
plot(LSresid ~ fitted(LSmod), col = "purple", ylim = c(-1.5,
    1.5), xlab = "Fitted Values", ylab = "Residuals")
abline(0, 0, col = "black", lty = 2)
```



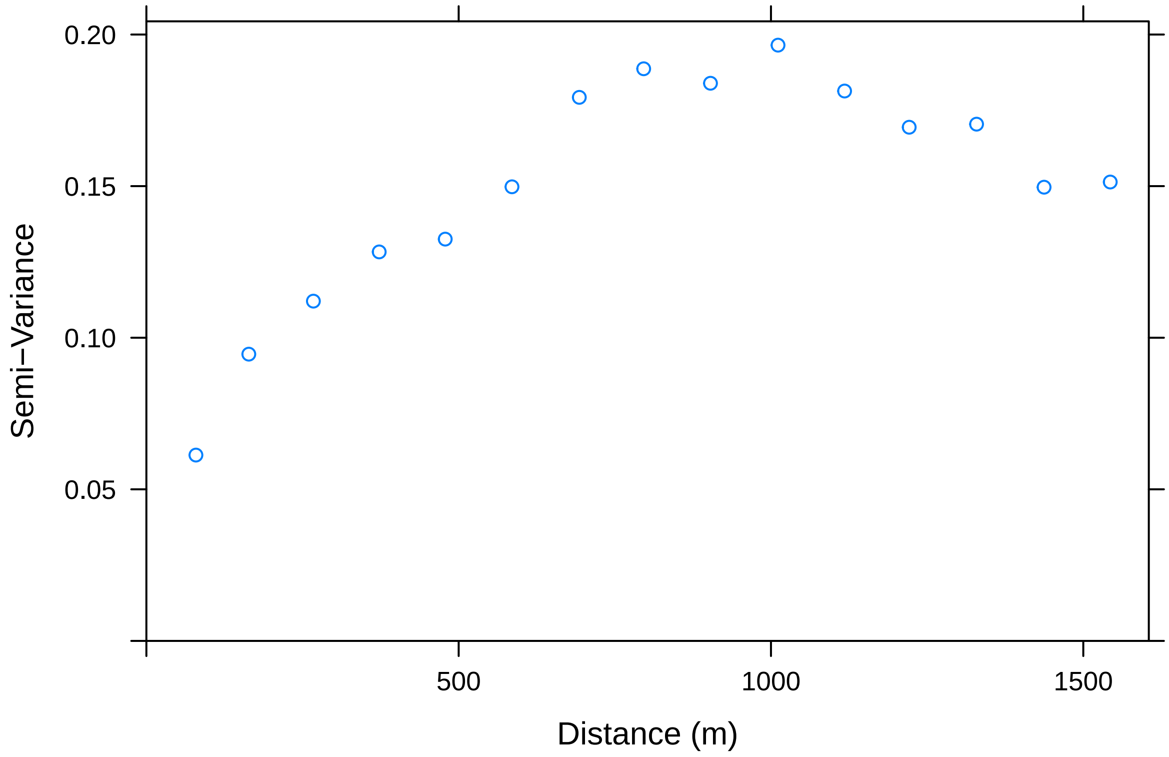
Normality assumption doesn't look terrible (not perfect either).

Variogram cloud for log(zinc) with the covariate trend removed - as usual not too informative.

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



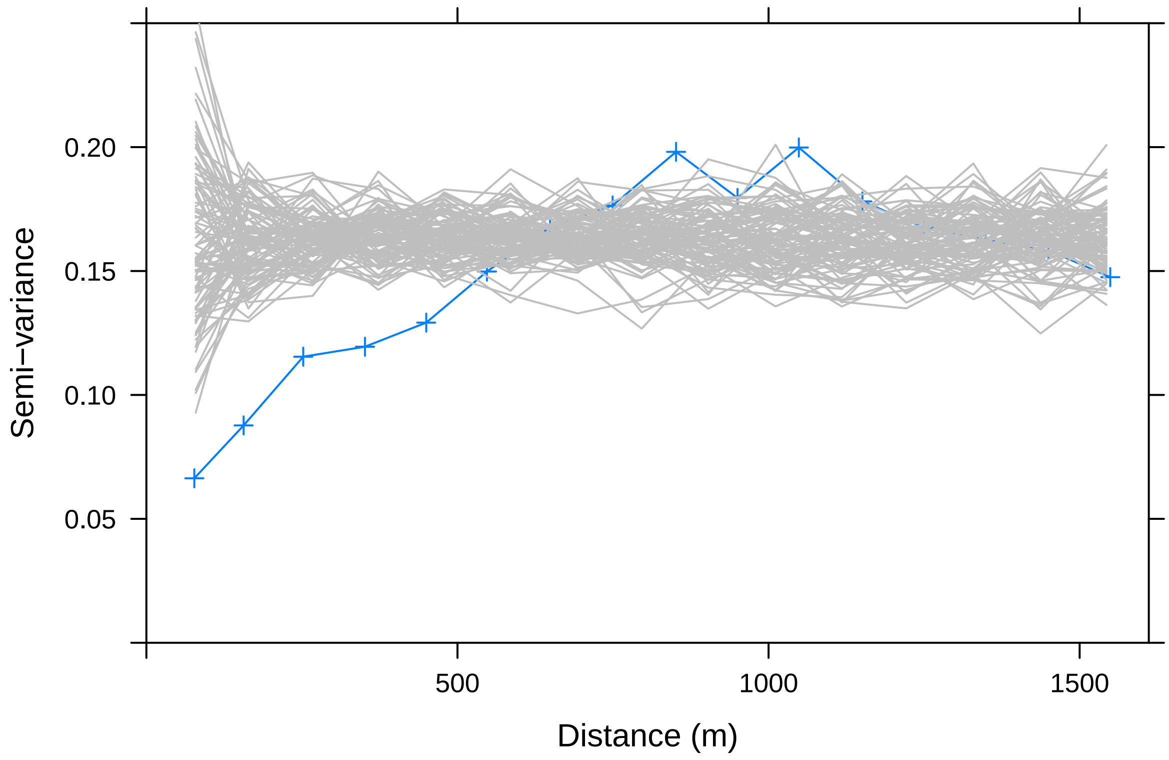
```r
# same thing, with no cloud
bnd <- variogram(log(zinc) ~ sqrt(meuse$dist) + as.factor(meuse$ffreq),
    meuse)
plot(bnd, ylab = "Semi-Variance", xlab = "Distance (m)")
```



Monte Carlo simulations of semi-variogram

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

```r
LSmod <- lm(log(meuse$zinc) ~ sqrt(meuse$dist) + as.factor(meuse$ffreq))
LSresid <- residuals(LSmod, type = "pearson")
meuse$LSresid <- LSresid
v <- variogram(LSresid ~ 1, meuse, width = 100, cutoff = 1600)
plot(v, type = "b", pch = 3, xlab = "Distance (m)",
    ylab = "Semi-variance", ylim = c(0, 0.25))
fn = function(n = 100) {
    for (i in 1:n) {
        meuse$random = sample(meuse$LSresid)
        v = variogram(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 in the residuals here.

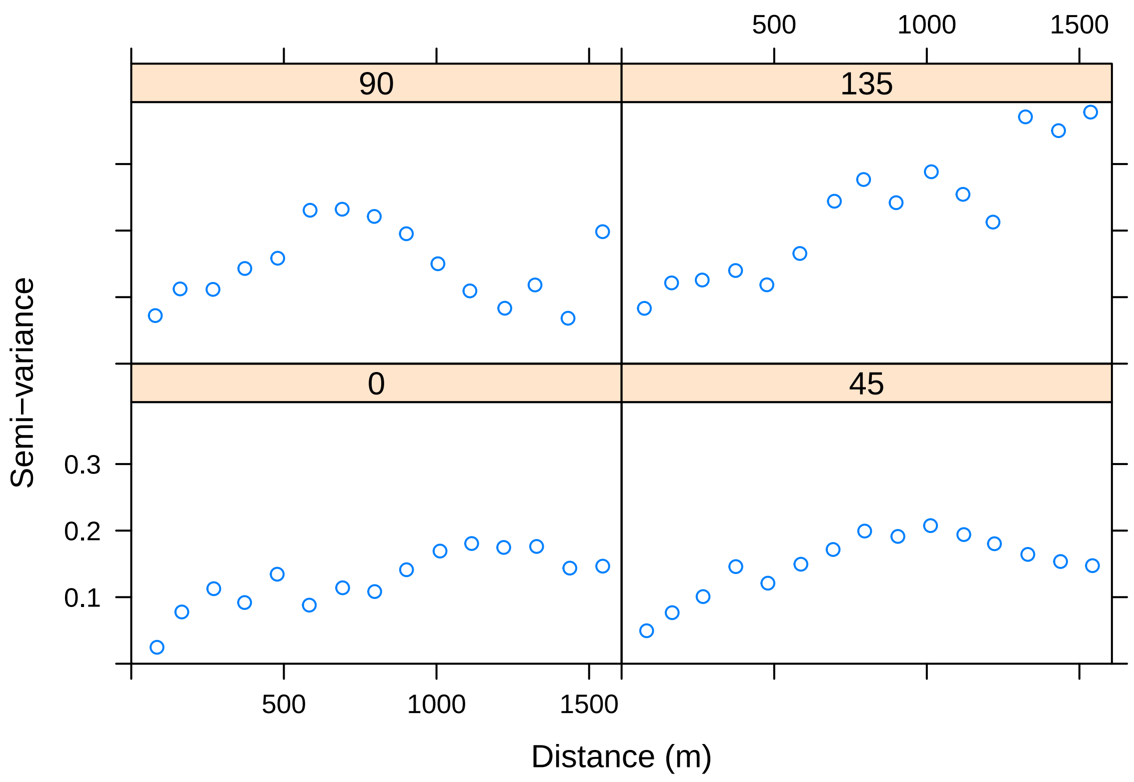
Directional variogram with linear trend removed

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

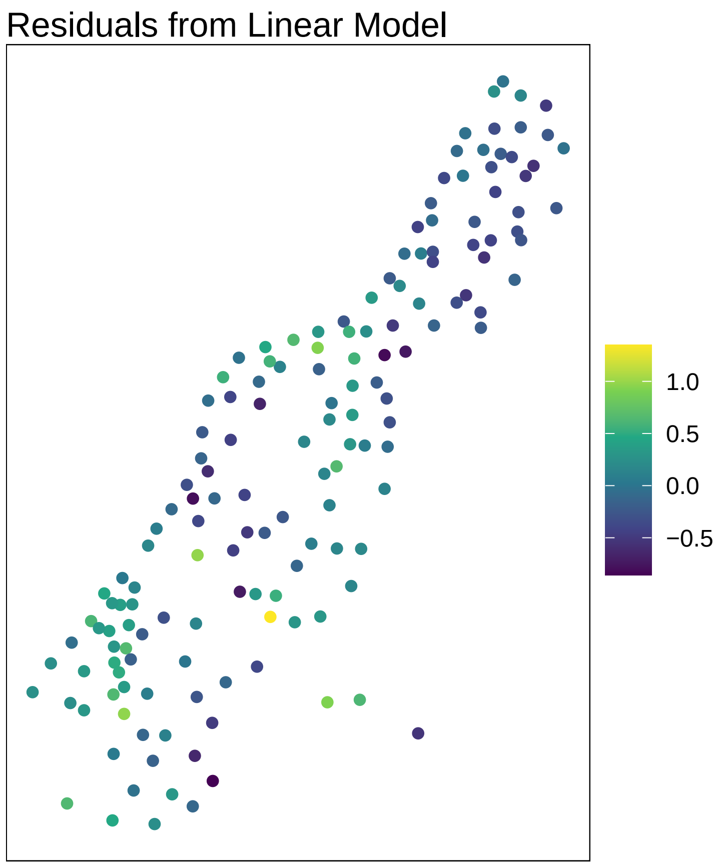
- `fit.variogram` for estimation from the variogram
- `krige` (and associated functions) for Kriging,
- `vgm` generates variogram models

`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))
```

```
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)
LSmod <- lm(log(meuse$zinc) ~ sqrt(meuse$dist) + as.factor(meuse$ffreq))
LSresid <- residuals(LSmod, type = "pearson")
m.sf$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) + theme(panel.border = element_rect(fill = NA,
    color = "black"))
```



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.

This is more evidence (in addition to the variogram) that there is residual spatial dependence in log(zinc).

```
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$parameters.summary
##            status   values
## beta0   estimated   7.0712
## beta1   estimated  -2.1208
## beta2   estimated  -0.5154
## beta3   estimated  -0.5266
## tausq   estimated   0.0372
## sigmasq estimated   0.1316
## phi     estimated 300.5381
## kappa       fixed   0.5000
## psiA        fixed   0.0000
## psiR        fixed   1.0000
## lambda      fixed   1.0000
mlfit$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.

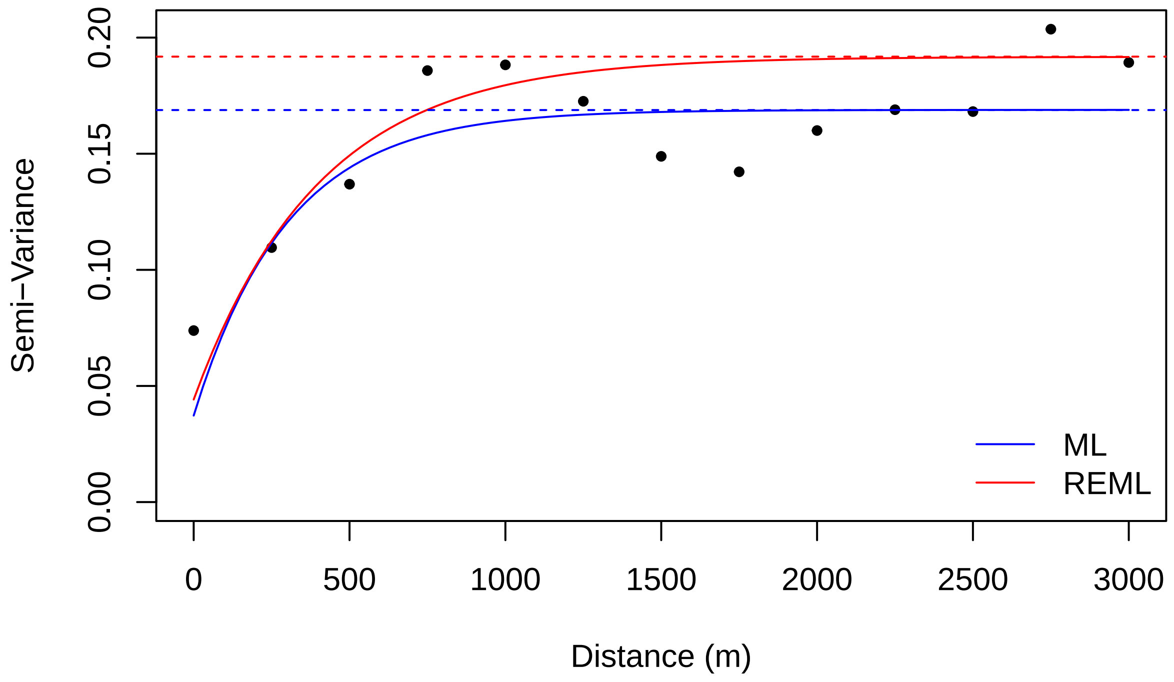
```
remlfit$parameters.summary
##            status   values
## beta0   estimated   7.0823
## beta1   estimated  -2.1109
## beta2   estimated  -0.5280
## beta3   estimated  -0.5455
## tausq   estimated   0.0442
## sigmasq estimated   0.1476
## phi     estimated 401.4313
## kappa       fixed   0.5000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

We examine LS estimates, along with ML and REML and see large increases in the standard errors with the spatial models, which shows that if we were doing spatial regression, the LS model will give us highly inaccurate confidence intervals (much too narrow). There is less information in the data, because of the spatial dependence, and so the standard errors increase.

```
for (i in 1:4) {
    cat(cbind(coef(LSmod)[i], vcov(LSmod)[i, i], mlfit$beta[i],
        sqrt(mlfit$beta.var[i, i]), remlfit$beta[i],
        sqrt(remlfit$beta.var[i, i])), "\n")
}
## 7.029868 0.005099518 7.071249 0.1326244 7.082279 0.1527493
## -2.266017 0.02429007 -2.120796 0.2365602 -2.110917 0.2515499
## -0.3605368 0.006185605 -0.5153624 0.06780136 -0.5280202 0.06913553
## -0.3166713 0.009634544 -0.5265608 0.1015237 -0.545532 0.1046061
```

Fitted variograms under the two models, with the dashed lines reflecting the total variance estimate from the two inferential approaches.

```
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)
mltotvar <- mlfit$parameters.summary[5, 2] + mlfit$parameters.summary[6,
    2]
abline(h = mltotvar, col = "blue", lty = 2)
remltotvar <- remlfit$parameters.summary[5, 2] + remlfit$parameters.summary[6,
    2]
abline(h = remltotvar, col = "red", lty = 2)
```



Prediction for log(zinc) by Kriging

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

```
data(meuse.grid)
kc <- krige.control(trend.d = ~sqrt(meuse$dist) + as.factor(meuse$ffreq),
    trend.l = ~sqrt(meuse.grid$dist) + as.factor(meuse.grid$ffreq),
    obj.m = mlfit)
kc_pred <- krige.conv(geozinc, loc = meuse.grid[, c("x",
    "y")], krige = kc)
## krige.conv: model with mean defined by covariates provided by the user
## krige.conv: Kriging performed using global neighbourhood
```

The predictions and variance are stored in the `predict` and `krige.var` outputs of the `krig.conv` function.

```
meuse.grid$pred <- kc_pred$predict
meuse.grid$sd <- sqrt(kc_pred$krige.var)
```

We can also extract the Gaussian Process predictive surface by subtracting $X\beta$ from the predictions of log(zinc).

```
meuse.grid$pred.S <- meuse.grid$pred - (kc_pred$beta.est[1] +
    kc_pred$beta.est[2] * sqrt(meuse.grid$dist) + kc_pred$beta.est[3] *
    (as.numeric(meuse.grid$ffreq == "2")) + kc_pred$beta.est[4] *
    (as.numeric(meuse.grid$ffreq == "3")))
```

To plot these results, first great an `sf` object from our grid, then use ggplot.

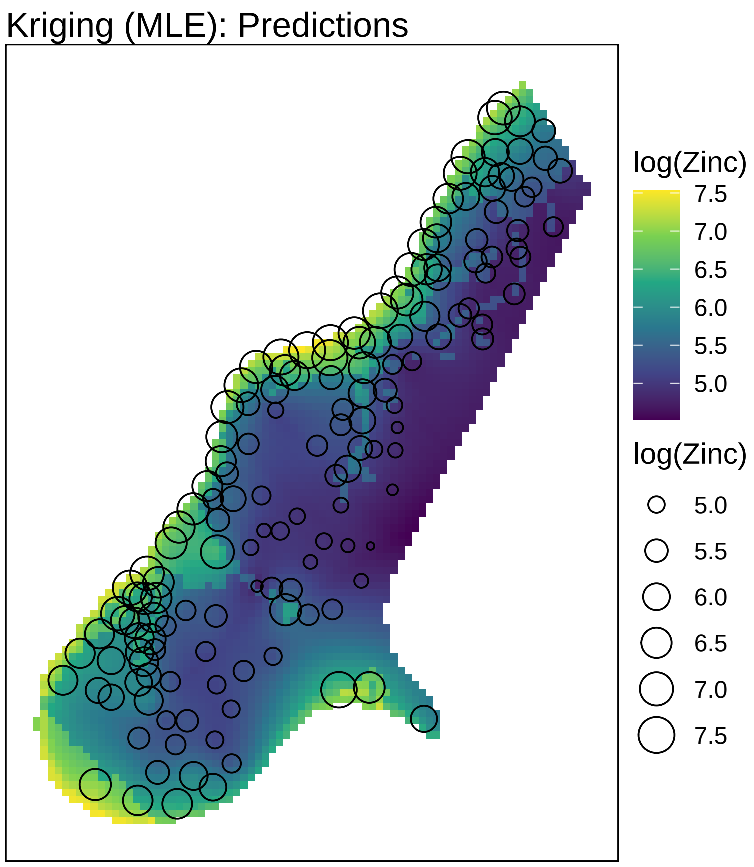
```
meuse.grid.sf <- st_as_sf(meuse.grid, coords = c("x",
    "y"))
meuse.grid.poly <- meuse.grid.sf %>%
    st_make_grid(cellsize = 40, offset = c(min(meuse.grid$x -
        20), min(meuse.grid$y - 20))) %>%
    st_as_sf() %>%
    st_join(meuse.grid.sf)
```

Also get meuse as a sf object.

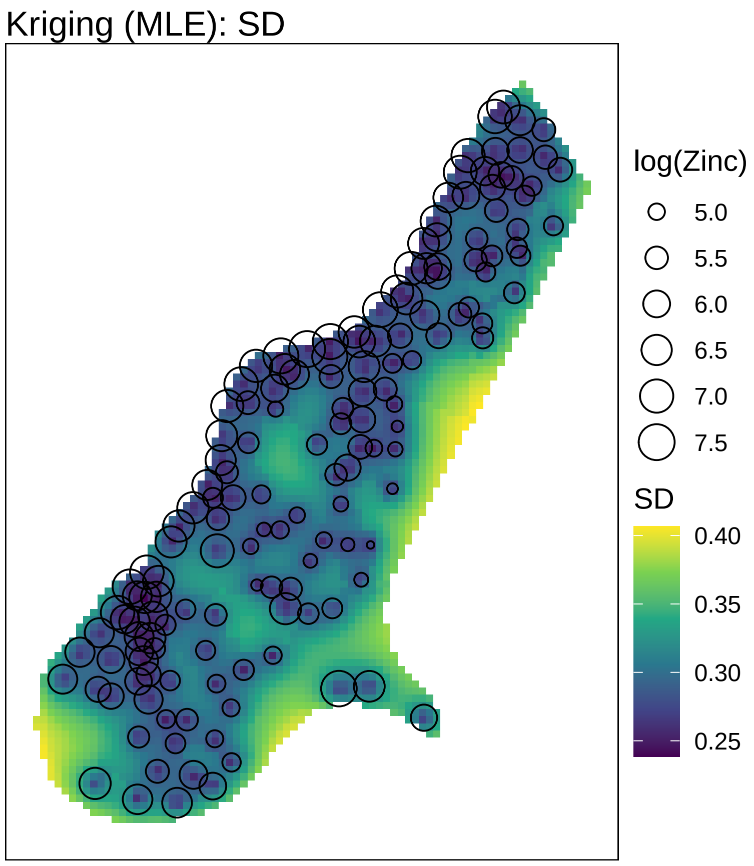
```
meuse.sf <- st_as_sf(meuse, coords = c("x", "y"))
```

Now plot the results.

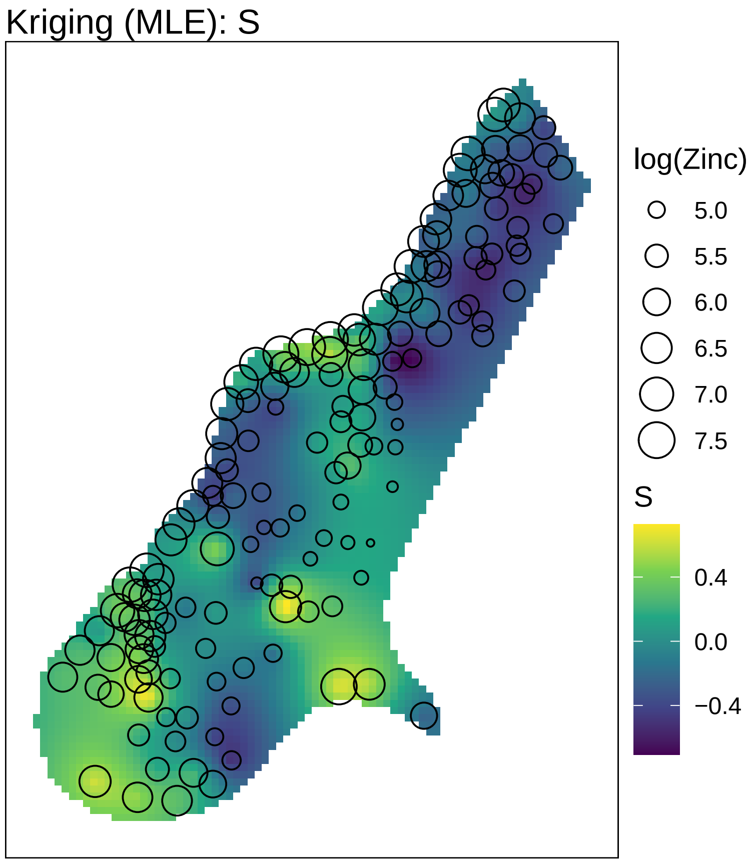
```
ggplot() + geom_sf(data = meuse.grid.poly, aes(fill = pred),
    color = NA) + geom_sf(data = meuse.sf, aes(size = log(zinc)),
    shape = 1) + theme_void() + scale_fill_viridis_c(na.value = "white") +
    labs(title = "Kriging (MLE): Predictions", fill = "log(Zinc)",
        size = "log(Zinc)") + theme(panel.border = element_rect(fill = NA,
    color = "black"))
```



```
ggplot() + geom_sf(data = meuse.grid.poly, aes(fill = sd),
    color = NA) + geom_sf(data = meuse.sf, aes(size = log(zinc)),
    shape = 1) + theme_void() + scale_fill_viridis_c(na.value = "white") +
    labs(title = "Kriging (MLE): SD", fill = "SD",
        size = "log(Zinc)") + theme(panel.border = element_rect(fill = NA,
    color = "black"))
```



```
ggplot() + geom_sf(data = meuse.grid.poly, aes(fill = pred.S),
    color = NA) + geom_sf(data = meuse.sf, aes(size = log(zinc)),
    shape = 1) + theme_void() + scale_fill_viridis_c(na.value = "white") +
    labs(title = "Kriging (MLE): S", fill = "S", size = "log(Zinc)") +
    theme(panel.border = element_rect(fill = NA, color = "black"))
```



Observe that 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)
meuse$ffreq <- as.factor(meuse$ffreq)
gam.mod <- gam(log(zinc) ~ s(x, y, bs = "tp") + sqrt(dist) +
    ffreq, data = meuse, method = "REML")
```

```
summary(gam.mod)
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(zinc) ~ s(x, y, bs = "tp") + sqrt(dist) + ffreq
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.9831     0.1522  45.869  < 2e-16 ***
## sqrt(dist)   -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

Predict log(zinc) using our GAM.

```
meuse.grid$pred.gam <- predict.gam(gam.mod, meuse.grid)
meuse.grid$pred.gam.sd <- predict.gam(gam.mod, se.fit = T,
    meuse.grid)$se.fit
```

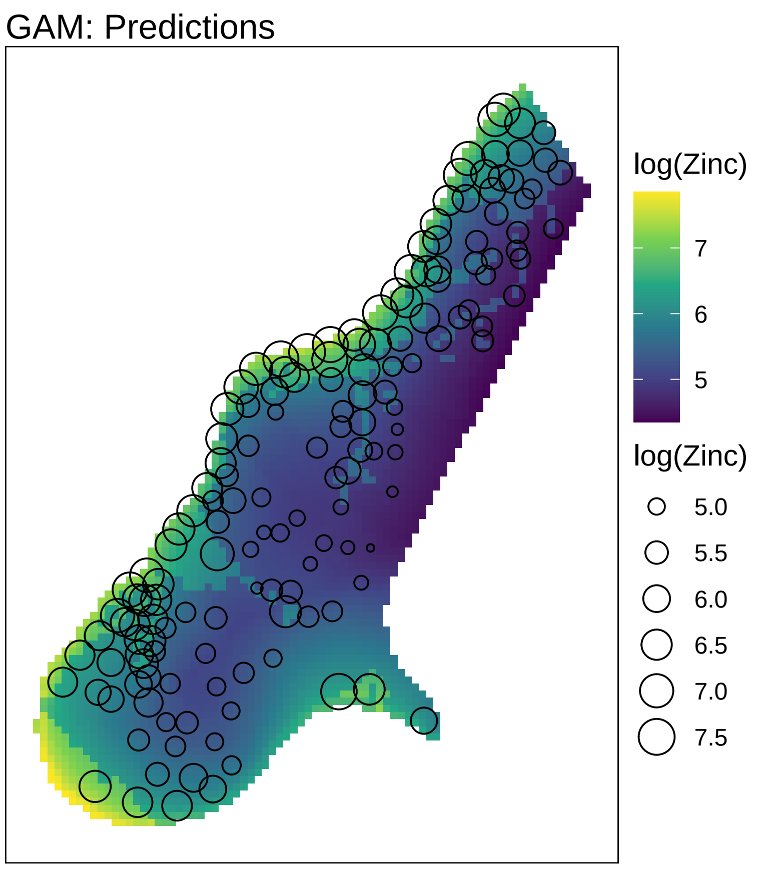
We can also predict just the GAM spline surface using `type="terms"`.

```
meuse.grid$pred.gam.S <- predict.gam(gam.mod, meuse.grid,
    type = "terms")[, 3]
```

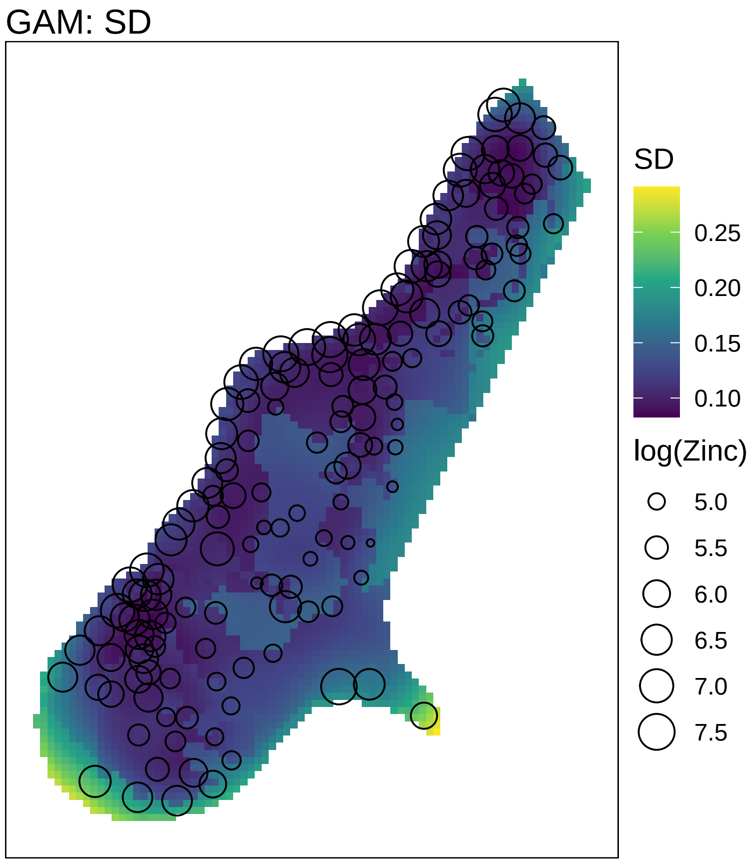
Now prep a `sf` object and plot these results.

```
meuse.grid.sf <- st_as_sf(meuse.grid, coords = c("x",
    "y"))
meuse.grid.poly <- meuse.grid.sf %>%
    st_make_grid(cellsize = 40, offset = c(min(meuse.grid$x -
        20), min(meuse.grid$y - 20))) %>%
    st_as_sf() %>%
    st_join(meuse.grid.sf)
```

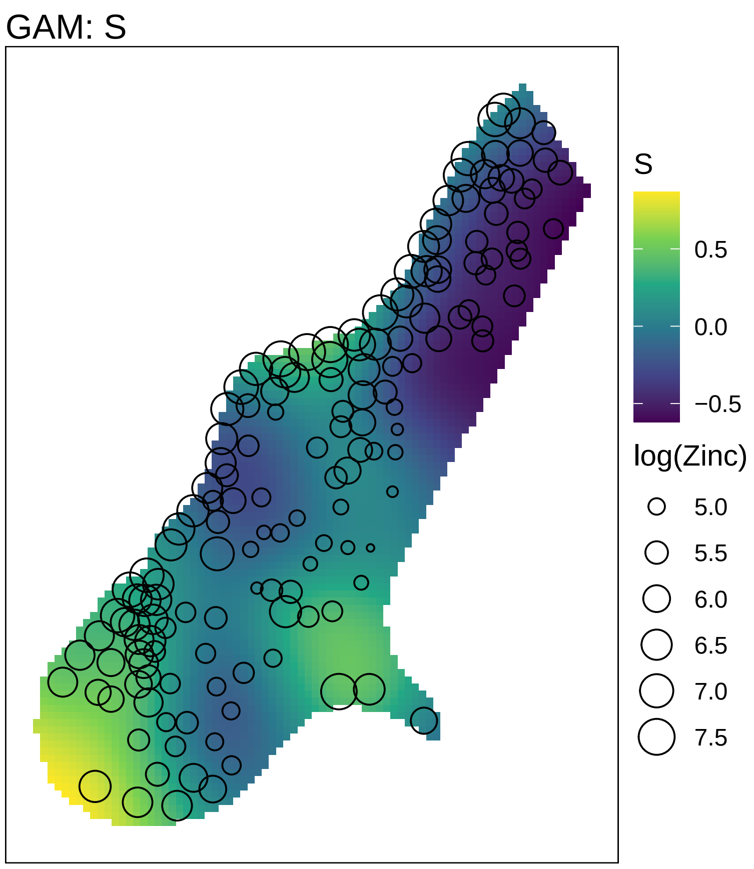
```
# Predictions
ggplot() + geom_sf(data = meuse.grid.poly, aes(fill = pred.gam),
    color = NA) + geom_sf(data = meuse.sf, aes(size = log(zinc)),
    shape = 1) + theme_void() + scale_fill_viridis_c(na.value = "white") +
    labs(title = "GAM: Predictions", fill = "log(Zinc)",
        size = "log(Zinc)") + theme(panel.border = element_rect(fill = NA,
    color = "black"))
```



```
# SD
ggplot() + geom_sf(data = meuse.grid.poly, aes(fill = pred.gam.sd),
    color = NA) + geom_sf(data = meuse.sf, aes(size = log(zinc)),
    shape = 1) + theme_void() + scale_fill_viridis_c(na.value = "white") +
    labs(title = "GAM: SD", fill = "SD", size = "log(Zinc)") +
    theme(panel.border = element_rect(fill = NA, color = "black"))
```



```r
# Spatial surface
ggplot() + geom_sf(data = meuse.grid.poly, aes(fill = pred.gam.S),
    color = NA) + geom_sf(data = meuse.sf, aes(size = log(zinc)),
    shape = 1) + theme_void() + scale_fill_viridis_c(na.value = "white") +
    labs(title = "GAM: S", fill = "S", size = "log(Zinc)") +
    theme(panel.border = element_rect(fill = NA, color = "black"))
```



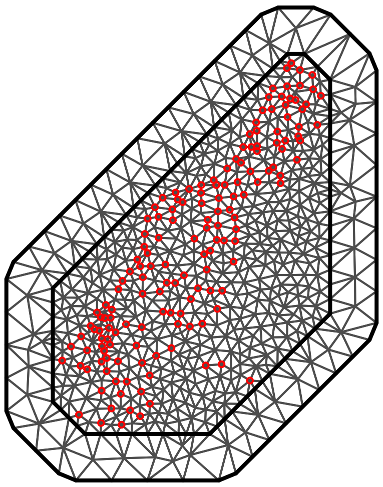
SPDE Model for log(zinc)

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

```
zincdf <- meuse %>%
    as.data.frame() %>%
    mutate(sqrtdist = sqrt(dist), lzinc = log(zinc)) %>%
    select(y = lzinc, locx = x, locy = y, sqrtdist,
        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.

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

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

# 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.

```
# 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
```

```r
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
  control.inla = list(int.strategy = "eb"),
  # - faster computation
  # control.inla = list(int.strategy='grid'),
  # - More accurate integration over hyper-parameters
  verbose = F
)
summary(res)
```

Finally, we can plot the means and standard deviations of our estimates.

```r
local.plot.field <- function(field, mesh, xlim = c(177000,
    183000), ylim = c(329000, 334000), ...) {
    stopifnot(length(field) == mesh$n)
    # - error when using the wrong mesh
    proj <- inla.mesh.projector(mesh, xlim = xlim,
        ylim = ylim, dims = c(300, 300))
    # Project from the mesh onto a 300x300
    # plotting grid using whatever is fed to the
    # function. For example, it could be the
    # posterior mean, or draw from the posterior,
    # or fitted values
    field.proj <- inla.mesh.project(proj, field)
    # Do the projection by taking a convex
    # combination (with up to 3 elements) from
    # the values on the vertices
    image.plot(list(x = proj$x, y = proj$y, z = field.proj),
        xlim = xlim, ylim = ylim, col = plasma(101),
        ...)
}

# Plot mean
local.plot.field(res$summary.random[["s"]][["mean"]],
    mesh, cex.axis = 0.5)
lines(178000 + c(-0.5, 0.5) * (res$summary.hyperpar[2,
    "0.5quant"]), c(333500, 333500), lwd = 3)  # add on the estimated range

# Plot SD
local.plot.field(res$summary.random$s$sd, mesh, cex.axis = 0.5)
```

Summary

Compare the estimates of the fixed effects across methods.

```r
krig_beta <- data.frame(method = "Kriging (MLE)", betas = kc_pred$beta.est,
    beta_num = 1:4)
gam_beta <- data.frame(method = "GAM", betas = coef(gam.mod)[1:4],
    beta_num = 1:4)

all_betas <- rbind(krig_beta, gam_beta)
all_betas %>%
    pivot_wider(id_cols = method, names_from = beta_num,
        values_from = betas) %>%
    kable(format = "latex", booktabs = T, escape = F,
        digits = 2, col.names = c("Method", "$\\beta_0$",
            "$\\beta_1$", "$\\beta_2$", "$\\beta_3$"))
```

Method	β_0	β_1	β_2	β_3
Kriging (MLE)	7.07	-2.12	-0.52	-0.53
GAM	6.98	-1.89	-0.59	-0.62