Spatial Predicton of Arsenic Contaminants in Groundwater

Gaston County, 2012-2022

Eric Delmelle Jiaxin Liu

05 April 2024

Abstract

We use geostastical techniques to map the variations of As contimants measured at different private wells accross Gaston County, North Carolina (2012-2022). Affiliation: University of North Carolina; Charlotte (delmelle@gmail.com)

1. Preambule

Remove any existing datasets

```
rm(list=ls(all=TRUE))
```

Set up working directory

```
setwd("/Users/delmelle/Dropbox (UNC Charlotte)/WRRI2000/maps/")
#setwd("C:\\Users\\edelmel1\\Dropbox (UNC Charlotte)\\WRRI2000\\maps\\")
```

Install and call libraries if needed

Some libraries may already be installed on your computer, so we will only install them if they are required.

```
if (!require('geoR')) install.packages('geoR'); library('geoR')
if (!require('RColorBrewer')) install.packages('RColorBrewer'); library('RColorBrewer')
if (!require('sp')) install.packages('sp'); library('sp')
if (!require('sf')) install.packages('sf'); library('sf')
#if (!require('rgdal')) install.packages('rgdal'); library('rgdal')
if (!require('devtools')) install.packages('devtools'); library('devtools')
if (!require('usethis')) install.packages('usethis'); library('usethis')
if (!require('ggplot2')) install.packages('ggplot2'); library('ggplot2')
if (!require('dplyr')) install.packages('dplyr'); library('dplyr')
if (!require('raster')) install.packages('raster'); library('raster')
if (!require('pacman')) install.packages('pacman'); library('pacman')
if (!require('tmap')) install.packages('tmap'); library('tmap')
if (!require('qpcR')) install.packages('qpcR'); library('qpcR')
if (!require('ggmap')) install.packages('ggmap'); library('ggmap')
if (!require('maps')) install.packages('maps'); library('maps')
if (!require('tidyverse')) install.packages('tidyverse'); library('tidyverse')
if (!require('showtext')) install.packages('showtext'); library('showtext')
if (!require('rvest')) install.packages('rvest'); library('rvest')
```

2. Read data in

Gaston County Border

This is the geographic border of our study area. It is important, since it will allow us to exclude points that fall outside our study region, and only retain estimations inside our study region. Gaston County, North Carolina (942 square kilometers) is a fast-growing county of nearly 225,000 residents (2019) in the South-Central Piedmont section of North Carolina.

```
gaston <- shapefile("/Users/delmelle/Dropbox (UNC Charlotte)/Rstatscripts/gaston_outline_Project.shp")
gaston_sf <- st_as_sf(gaston)  #sf = simple feature
gaston_sf <- st_transform(gaston_sf, 3857) # transform to Spherical Mercator
gaston_geom <- st_geometry(gaston_sf) #st_geometry returns an object of class
#sfc, a list-column with geometries
gaston_border <- gaston_geom[[1]][[1]]
```

Inorganic data

Arsenic data for private wells were obtained from the Gaston County Department of Health and Human Services (GC-DHHS) for 2011 through 2022. The data also contained information on the permit number, owner's name, residential address, collection date, sampling point, pH, and other inorganic compounds.

```
shp <- shapefile("/Users/delmelle/Dropbox (UNC Charlotte)/Rstatscripts/Inorganic_data.shp")
names(shp)
```

```
##   [1] "Loc_name"   "Status"     "Score"      "Match_type" "Match_addr"
##   [6] "LongLabel"  "ShortLabel" "Addr_type"  "Type"       "PlaceName"
##  [11] "Place_addr" "Phone"      "URL"        "Rank"       "AddBldg"
##  [16] "AddNum"     "AddNumFrom" "AddNumTo"   "AddRange"   "Side"
##  [21] "StPreDir"   "StPreType"  "StName"     "StType"     "StDir"
##  [26] "BldgType"   "BldgName"   "LevelType"  "LevelName"  "UnitType"
##  [31] "UnitName"   "SubAddr"    "StAddr"     "Block"      "Sector"
##  [36] "Nbrhd"      "District"   "City"       "MetroArea"  "Subregion"
##  [41] "Region"     "RegionAbbr" "Territory"  "Zone"       "Postal"
##  [46] "PostalExt"  "Country"    "CntryName"  "LangCode"   "Distance"
##  [51] "X"          "Y"          "DisplayX"   "DisplayY"   "Xmin"
##  [56] "Xmax"       "Ymin"       "Ymax"       "ExInfo"     "IN_Address"
##  [61] "IN_Addre_1" "IN_Addre_2" "IN_Neighbo" "IN_City"    "IN_Subregi"
##  [66] "IN_Region"  "IN_Postal"  "IN_PostalE" "IN_Country" "USER_COL_A"
##  [71] "USER_Data_" "USER_Clien" "USER_Addre" "USER_City"  "USER_State"
##  [76] "USER_ZipCo" "USER_Well_" "USER_STARL" "USER_Colle" "USER_Sampl"
##  [81] "USER_Unit"  "USER_Alumi" "USER_Antim" "USER_Arsen" "USER_Bariu"
##  [86] "USER_Beryl" "USER_Boron" "USER_Cadmi" "USER_Calci" "USER_Chlor"
##  [91] "USER_Chrom" "USER_Cobal" "USER_Coppe" "USER_Fluor" "USER_Iron"
##  [96] "USER_Lead"  "USER_Magne" "USER_Manga" "USER_Mercu" "USER_Molyb"
## [101] "USER_Nicke" "USER_Nitra" "USER_Nitri" "USER_pH"    "USER_Potas"
## [106] "USER_Selen" "USER_Silve" "USER_Stron" "USER_Sodiu" "USER_Sulfa"
## [111] "USER_Thall" "USER_Total" "USER_Tot_1" "USER_Tot_2" "USER_Tot_3"
## [116] "USER_Vanad" "USER_Zinc"  "USER_Hexav"
```

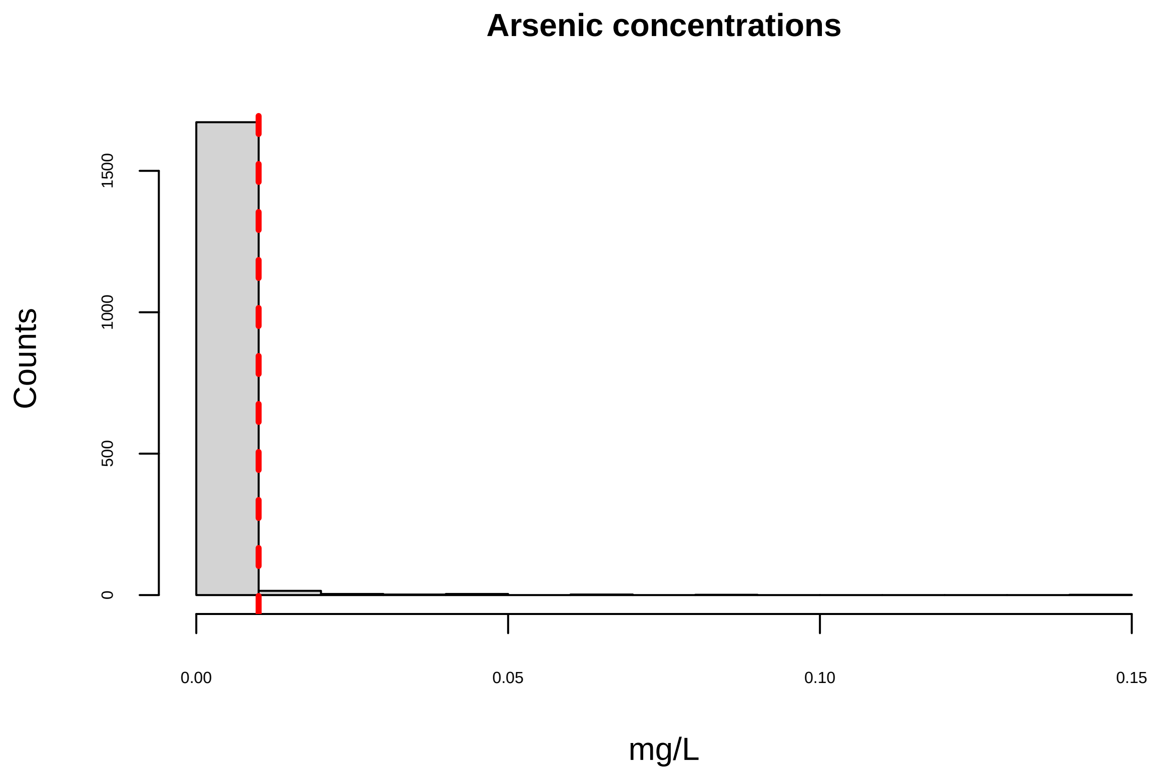
Data cleanup.

We are interested to model the spatial variation of Arsenic in groundwater, as measured in mg/L in water from private wells. The samples are sent to a lab, and results sent back to us in a PDF.

Removing NAs & Converting non-measurables but detectable levels to 0.

Sometimes, there are no-values (NAs); and these need to be removed, and sometimes we know contamination is present, but cannot be accurately measured because its concentration is too small. We recode these values to 0 At the end of this process, we convert all of our values from text to numeric.

```
subsetArsenic<- subset(shp, USER_Arsen!="NA")  ## remove NAs
count<-0
for (i in 1:nrow(subsetArsenic)) {
  if (subsetArsenic$USER_Arsen[i] =="< 0.005"){
    subsetArsenic$USER_Arsen[i] = 0
    count <- count +1
  }
  else if (subsetArsenic$USER_Arsen[i] =="<0.01"){
    subsetArsenic$USER_Arsen[i] = 0
    count <- count +1
    }
  else if (subsetArsenic$USER_Arsen[i] =="<0.001"){
    subsetArsenic$USER_Arsen[i] = 0
    count <- count +1
  }
  else if (subsetArsenic$USER_Arsen[i] =="<0.005"){
    subsetArsenic$USER_Arsen[i] = 0
    count <- count +1
  }
  }
subsetArsenic$USER_ArsB <- as.numeric(subsetArsenic@data[["USER_Arsen"]])
hist(subsetArsenic$USER_ArsB, xlab = 'mg/L', ylab = 'Counts',
     main = 'Arsenic concentrations',  cex.lab=1,
     cex.axis=0.5, cex.main=1, cex.sub=0.5)
mcl<-0.01 # maximum contamination level
abline(v = mcl, col="red", lwd=3, lty=2)
```



```
summary(subsetArsenic$USER_ArsB)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0000000 0.0000000 0.0007407 0.0000000 0.1480000
```

Reprojecting our inorganic data.

Like we did for Gaston County, we are reprojecting our data so they match. Units are in meters.

```
shp_sf <- st_as_sf(subsetArsenic)
shp_sf <- st_transform(shp_sf, 3857)
```

Removing measurements outside study area.

Here, we remove points that have been geocoded outside of the study area.

```
good_points <- st_filter(shp_sf, gaston_sf)
```

Some of the records represent repeated sampling of the same well, e.g., when separate water samples are taken from the kitchen sink and at the well. We therefore retained only the maximum recorded value from the location with the multiple tests to reflect potential groundwater concentration, which will reduce our sample.

```
good_points_new <- good_points %>%
group_by(Match_addr) %>%
summarise(USER_Arsen = max(USER_Arsen))
temp <- st_geometry(good_points_new)
subsetArsenic$USER_ArsB <- as.numeric(subsetArsenic@data[["USER_Arsen"]])
```

3. Preparing our data

Based on the number of points (nrow(good_points_new)), we prepare a vector that will hold the point coordinates, and the Arsenic value to be interpolated. We do not need any other attribute information at this stage, so we only retain the information that we need.

```
pointmatrix <- matrix(data = NA, nrow = nrow(good_points_new), ncol = 3, byrow = FALSE)
for (i in c(1:nrow(good_points_new))){
  pointmatrix[i,1] <- as.numeric(temp[[i]][1])
  pointmatrix[i,2] <- as.numeric(temp[[i]][2])
  pointmatrix[i,3] <- as.numeric(good_points_new$USER_Arsen[i])
}
data_geo <- as.geodata(pointmatrix, coords.col = 1:2, data.col = 3, borders = TRUE)
```

```
## as.geodata: 2 replicated data locations found.
##  Consider using jitterDupCoords() for jittering replicated locations.
## WARNING: there are data at coincident or very closed locations, some of the geoR's functions may not
##  Use function dup.coords() to locate duplicated coordinates.
##  Consider using jitterDupCoords() for jittering replicated locations
```

How many measurements above the MCL limit?

We report the number and percentage of observations above the MCL (here Lead **As** levels larger than 0.01 are deemed unsafe to drink)

```
unsafe <- data_geo[(data_geo$data>=mcl)]
countUnsafe <- length(unsafe)
percUnsafe <- round(100*(length(unsafe) / length(data_geo$data)),2)
cat(countUnsafe, "measurements above", mcl, "mg/L, or", percUnsafe, "% \n")
```

```
## 32 measurements above 0.01 mg/L, or 1.94 %
```

We further search for duplicate coordinates using the **jitterDupCoords** function. The *max* argument is the maximum jittering distance.

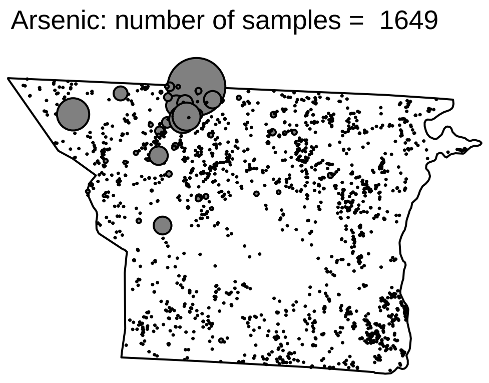
```
dup.coords(data_geo)
```

```
##      dup   Coord1  Coord2 data
## 1387   1 -9024916 4197751    0
## 1391   2 -9024916 4197751    0
## 1624   3 -9025556 4212151    0
## 1633   4 -9025556 4212151    0
```

```
data_geo <- jitterDupCoords(data_geo, max=0.01)
dup.coords(data_geo)
```

```
## NULL
```

```
data_geo$borders <- gaston_border
a <-paste("Arsenic: number of samples = ", length(data_geo$data), "\n")
points(data_geo, xlab = " ",ylab = " ", cex.min=.1, cex.max=4,
               col=gray(seq(1, 0.5, l=100)), yaxt="n", xaxt="n", bty="n")
text(-9040000, 4226500, a, cex=.8)
```



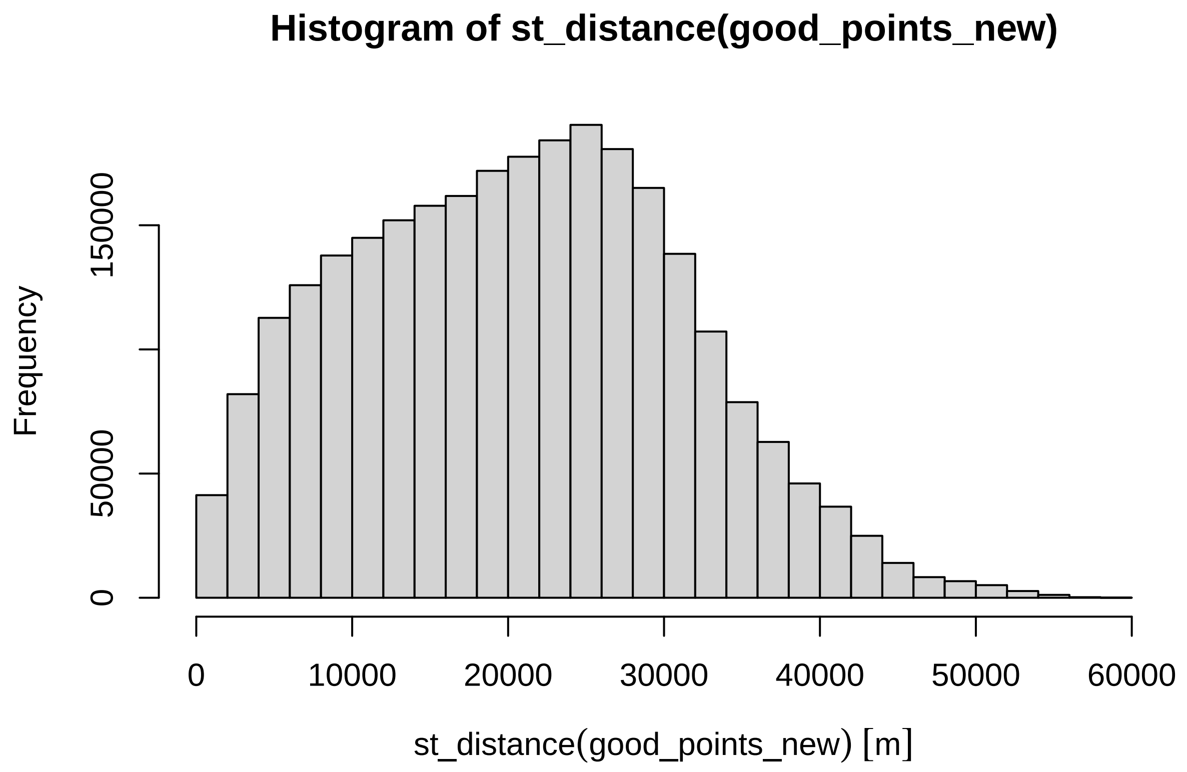
4. Variogram analysis

When we conduct variogram analysis, we want our search to be constrained to a little less than half or our study area. This can easily be obtained by looking at the range of all distances of our observations (using **st_distance**).

```
range(st_distance(good_points_new))
```

```
## Units: [m]
## [1]     0.00 58884.38
```

```
hist(st_distance(good_points_new))
```



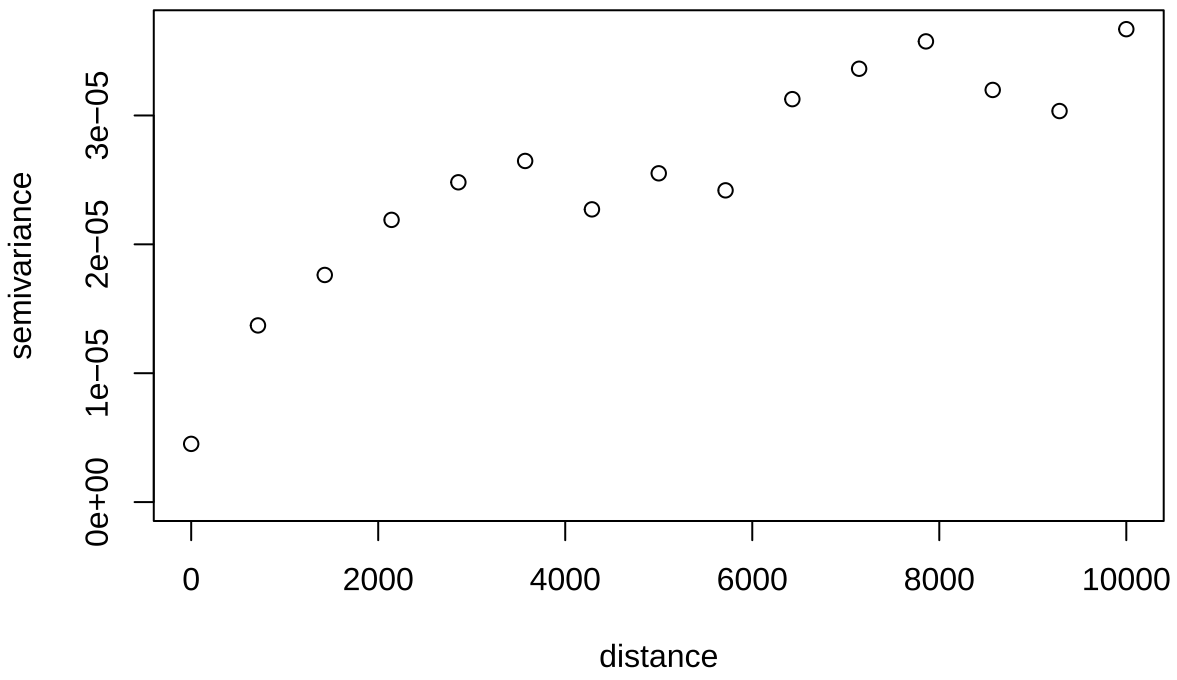
This analysis suggests that 58,884 meters is the maximum distance among the two most distant points. Hence, we will limit our search to about 30,000 meters (half the size of the county, approximately). We see from the histogram that the majority of distance pairs are well represented.

Empirical variogram

Using this information, we can now call the function **variog** from the **geoR** package and calibrate our search. The first argument we use is the input data, the second one is concerned with the search, creating 15 bins (l=15) from 0 to 10,000 meters. The reason to limit to 15km is that prior literature indicated a practical range around 5km For each of these bins, we calculate the average difference in Arsenic among all of the observations separated by this distance h. $gamma(h) = \frac{1}{2N_h} \sum_{i=1}^{N_h} [Y(x_{i+h}) - Y(x_i)]^2$ The semivariance $\gamma(h)$ is then plotted against all these distances. We are looking for a breaking point when the curve will flatten out.

```
plot(vario_model <- variog(data_geo, uvec = seq(0,10000,l=15)))
```

```
## variog: computing omnidirectional variogram
```



Model fitting

Note that we could also compute directional variograms (not shown here). We are now looking to fit a model (curve) to our empirical variogram. Either we can let the computer choose the parameters, or we can provide them to the algorithm. When the nugget is fixed, it means it is equal to 0. Note that we are considering an exponential and a spherical fit, but other options are possible.

```
exp_fit_fix <- variofit(vario_model, cov.model = "exp", fix.nugget = T)
```

```
## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "0"     "3200" "0"   "0.5"
## status        "est"   "est"  "fix" "fix"
## loss value: 3.08113223706879e-06
```

```
exp_fit_nofix <- variofit(vario_model, cov.model = "exp", fix.nugget = F)
```

```
## variofit: covariance model used is exponential
## variofit: weights used: npairs
## variofit: minimisation function used: optim
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "0"     "4800" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 2.16634235266108e-06
```

```
sph_fit_fix <- variofit(vario_model, cov.model = "sph", fix.nugget = T)
```

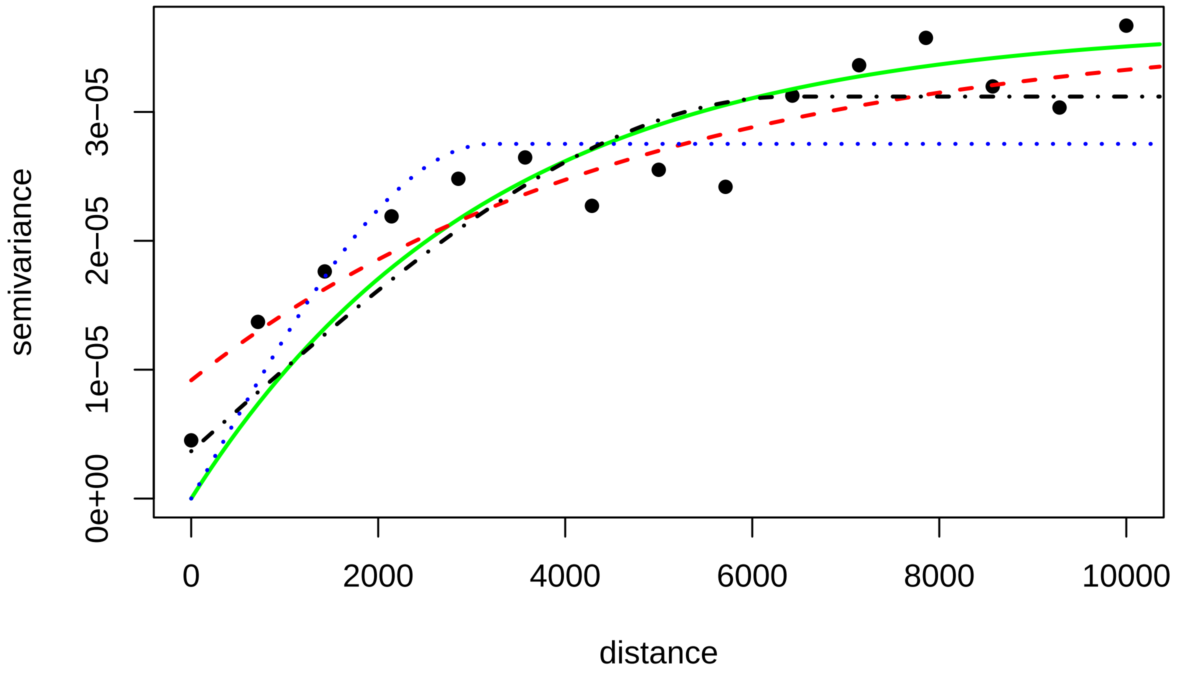
```
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim
## variofit: searching for best initial value ... selected values:
##                 sigmasq phi    tausq kappa
## initial.value "0"     "3200" "0"   "0.5"
## status        "est"   "est"  "fix" "fix"
## loss value: 6.59659105403508e-06
```

```
sph_fit_nofix <- variofit(vario_model, cov.model = "sph", fix.nugget = F)
```

```
## variofit: covariance model used is spherical
## variofit: weights used: npairs
## variofit: minimisation function used: optim
## variofit: searching for best initial value ... selected values:
##                 sigmasq phi    tausq kappa
## initial.value "0"     "6400" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 3.97972654062414e-06
```

We can then plot the fitted model on top of the empirical variogram.

```
plot(vario_model,pch=16)
lines(exp_fit_fix,col="green",lwd=2, lty = 1)
lines(exp_fit_nofix,col="red",lwd=2, lty = 2)
lines(sph_fit_fix,col="blue",lwd=2, lty = 3)
lines(sph_fit_nofix,col="black",lwd=2, lty = 4)
```



Which model is best?

To know which model is better, we look for the one that will minimize the sum of squares, using the **which.min** function.

```
exp_SSQ_fix <- summary(exp_fit_fix)$sum.of.squares
exp_SSQ_nofix <- summary(exp_fit_nofix)$sum.of.squares
sph_SSQ_fix <- summary(sph_fit_fix)$sum.of.squares
sph_SSQ_nofix <- summary(sph_fit_nofix)$sum.of.squares
which.min(list(exp_SSQ_fix, exp_SSQ_nofix, sph_SSQ_fix, sph_SSQ_nofix))
```

```
## [1] 2
```

This indicates that the exponential model where the nugget is not fixed is the best model in terms of the minimization of the sum of squares. However, based on previous literature, we know that the practical range should be between 5 and 10km; and the exponential model with a non-fixed nugget pushes the range too far. We opt instead for a spherical model where the nugget is not fixed. We can also inspect the parameters of the model, suggesting a nugget close to 0, and a range approaching 6500meters.

```
summary(sph_fit_nofix)
```

```
## $pmethod
## [1] "WLS (weighted least squares)"
##
## $cov.model
## [1] "spherical"
##
## $spatial.component
##      sigmasq          phi
## 2.752125e-05 6.400000e+03
##
## $spatial.component.extra
## kappa
##   0.5
##
## $nugget.component
##      tausq
## 3.6695e-06
##
## $fix.nugget
## [1] FALSE
##
## $fix.kappa
## [1] TRUE
##
## $practicalRange
## [1] 6400
##
## $sum.of.squares
##        value
## 3.979727e-06
##
## $estimated.pars
##        tausq      sigmasq          phi
## 3.669500e-06 2.752125e-05 6.400000e+03
##
## $weights
## [1] "npairs"
##
## $call
## variofit(vario = vario_model, cov.model = "sph", fix.nugget = F)
##
## attr(,"class")
## [1] "summary.variomodel"
```

5. Prediction through Kriging

Once our model has been selected, we are ready to estimate Arsenic throughout our region. We first specify the grid against which we want to interpolate Arsenic, using the **expand.grid** function. The *length.out* argument indicates the level of grid division; higher values will result in smaller grid cell. . . .but it will be computationally longer.

```r
prediction_grid <- expand.grid(seq(-9070000, -9005000, length.out = 200),
seq(4182500, 4222500, length.out = 200))
```

Kriging

We now are ready to call krige our data and interpolate at the grid cells, using **krige.conv**. Kriging also computes the prediction error (kriging variance), which is useful to understand the confidence we may have in our estimates. Here we use Ordinary Kriging (unknown mean).

```r
kcb <- krige.conv(data_geo, loc=prediction_grid,
                  krige=krige.control(obj.model=sph_fit_nofix),
                  borders=gaston_border)
```

```
## krige.conv: results will be returned only for prediction locations inside the borders
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

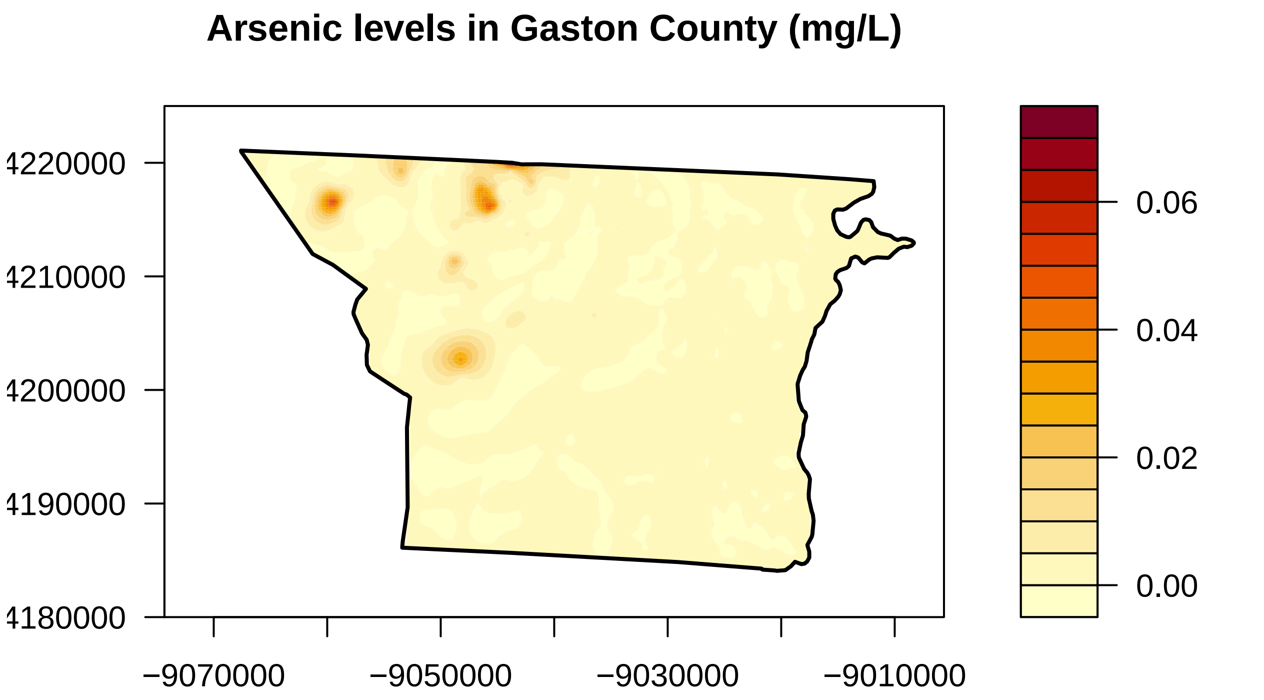
```r
kcb <- krige.conv(data_geo, loc=prediction_grid,
                  krige=krige.control(obj.model=sph_fit_nofix))
```

```
## krige.conv: results will be returned only for prediction locations inside the borders
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

```r
#library(raster)

# Create a blank raster using the extent of your predictions
# Assuming kriging_result$predict has the prediction values
# And kriging_result$coords are the coordinates

# First, find the range for coordinates to set the raster extent
#x.range <- range(prediction_grid$Var1)
#y.range <- range(prediction_grid$Var2)
#raster_obj <- raster(xmn=x.range[1], xmx=x.range[2], ymn=y.range[1], #ymx=y.range[2], nrows=100, ncols
#values(raster_obj) <- kcb$predict
```

We now plot our results using the **contour** function. We note increased elevation of Arsenic in the northwestern part of the County.

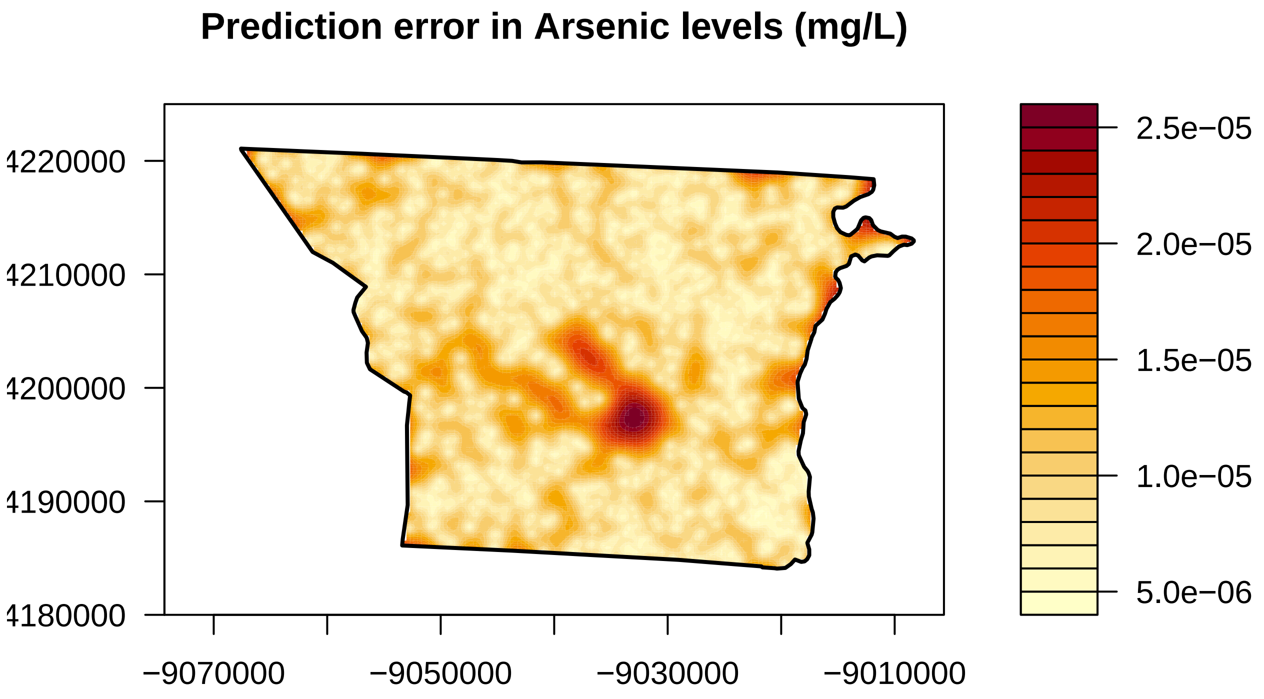
```r
contour(kcb, ylim=c(4180000, 4225000), xlim=c(-9050000,-9030000), filled=TRUE,
        plot.title = title(main = "Arsenic levels in Gaston County (mg/L)"))
```



Kriging error

Likewise, we can also plot the prediction error, suggesting a higher error in the center of the county - this is also where we have less observations of private wells, since most household in the city use municipal water.

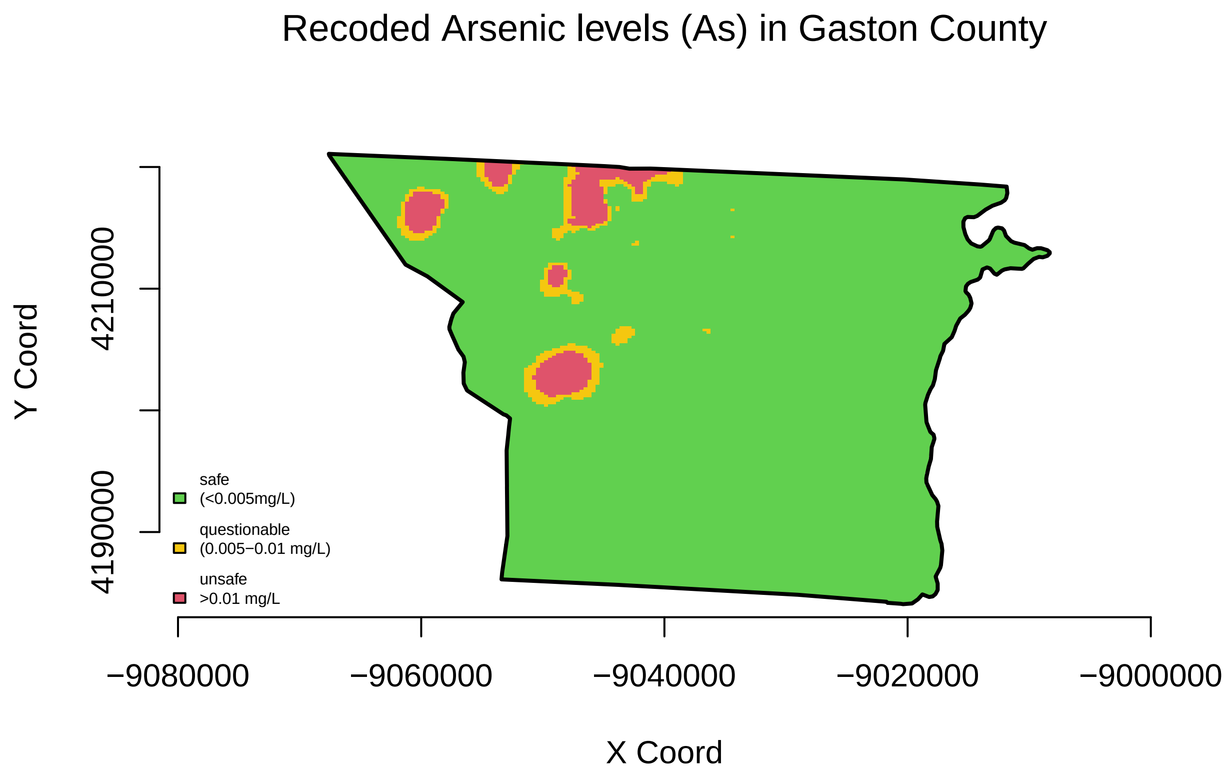
```r
contour(kcb, val=kcb$krige.var ,ylim=c(4180000, 4225000), xlim=c(-9050000,-9030000), filled=TRUE,
        plot.title = title(main = "Prediction error in Arsenic levels (mg/L)"))
```



6. Recode values into safe, questionable, unsafe

Using maximum threshold values set by the environmental protection agency (here Arsenic **As** levels larger than 0.01 are deemed unsafe to drink, so we create three categories: As <0.005mg/L: safe; 0.005mg/L < As < 0.01mg/L: questionable and As >0.01mg/L: unsafe to drink), we can recode our interpolated maps into categories. These maps can be particularly useful for deciding on where to alert citizens, or even conduct additional sampling.

```r
image(x=kcb, ylim=c(4183000, 4225000), xlim=c(-9050000,-9030000), locations=prediction_grid,
      borders=gaston_border, bty = "n",
      breaks=c(-2,mcl/2,mcl,max(kcb[["predict"]])),
      col = c(3,7,2),
      )
##borders=NULL
legend("bottomleft", legend=c("safe \n(<0.005mg/L)\n","questionable \n(0.005-0.01 mg/L)\n","unsafe \n>0
       col=c("green", "yellow","red"), fill = c(3,7,2), cex=.5)
title("Recoded Arsenic levels (As) in Gaston County", font.main = 1.5)
```

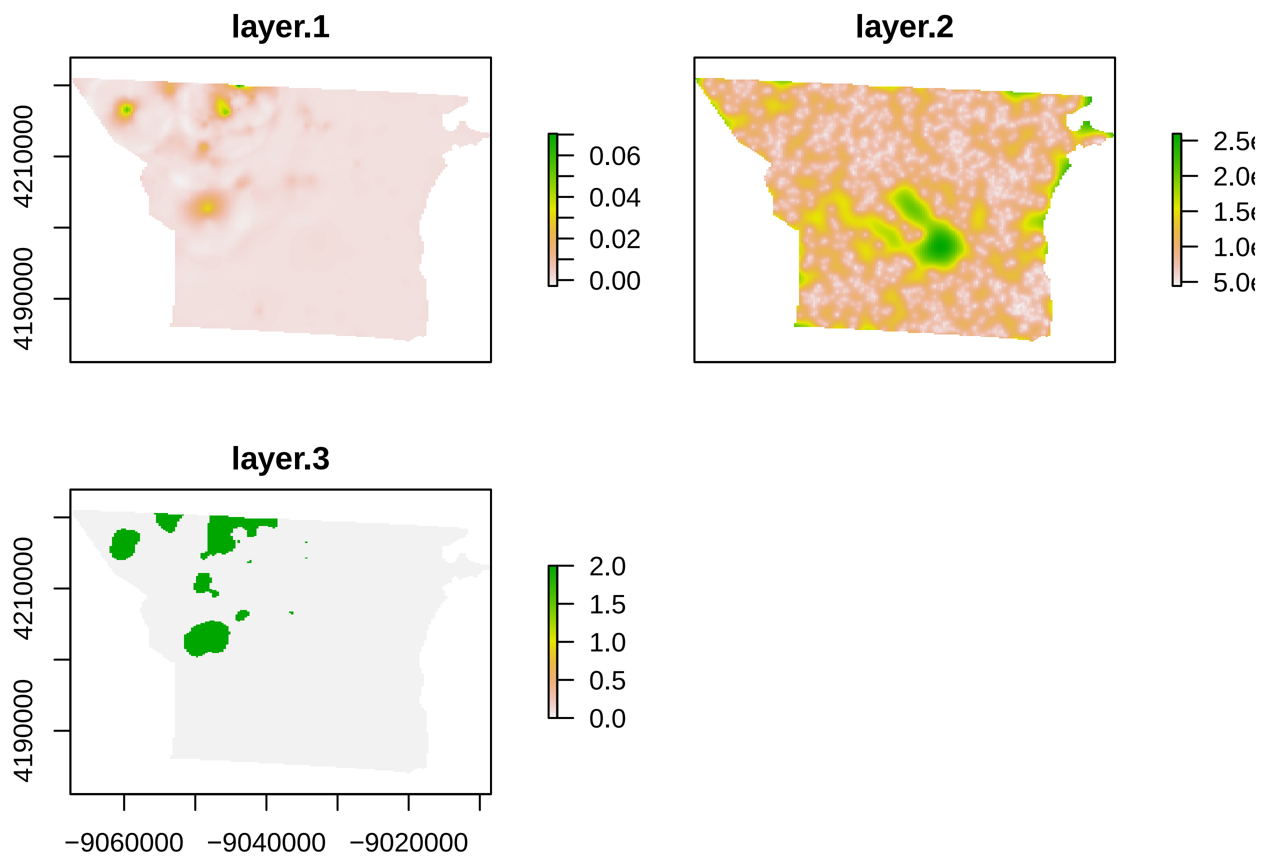


7. Exporting data to CSV

```r
gr0 <- polygrid(prediction_grid, borders = gaston_border, bound = T)

nrow = length(gr0$Var1)
# Create an empty matrix
pointmatrix_1 <- matrix(NA, nrow, ncol = 5, byrow = FALSE)

# Assign values to the first two columns of pointmatrix using the prediction_grid object
for (i in c(1:nrow)){
  pointmatrix_1[i,1] <- as.numeric(gr0[i,1])
  pointmatrix_1[i,2] <- as.numeric(gr0[i,2])
}

# Assign values to the third column of pointmatrix using kcb$predict
# Any missing values are filled with NA
pointmatrix_1[, 3] <- rep(NA, nrow)
pointmatrix_1[1:length(kcb$predict), 3] <- kcb$predict

# Assign values to the third column of pointmatrix using kcb$krige.var
# Any missing values are filled with NA
pointmatrix_1[, 4] <- rep(NA, nrow)
pointmatrix_1[1:length(kcb$krige.var), 4] <- kcb$krige.var

# Assign values to the fifth column of pointmatrix using kcb$predict, after
#they area recoded
# Any missing values are filled with NA
pointmatrix_1[, 5] <- pointmatrix_1[, 3]

pointmatrix_1[, 5][pointmatrix_1[, 5]<mcl/2] <- 0
pointmatrix_1[, 5][pointmatrix_1[, 5]<mcl & pointmatrix_1[, 5]> (mcl/2)] <- 1
pointmatrix_1[, 5][pointmatrix_1[, 5]>mcl] <- 2

#convert to raster
r <- rasterFromXYZ(pointmatrix_1)
class(r)
```

```
## [1] "RasterBrick"
## attr(,"package")
## [1] "raster"
```

```r
plot(r)
```



```
#convert to dataframe
pointmatrix_df <- as.data.frame(pointmatrix_1)

colnames(pointmatrix_df) <- c("x","y","zinc", "error", "safety")

#ggplot(pointmatrix_df, aes(x = pointmatrix_df[,1], y = pointmatrix_df[,2], fill = pointmatrix_df[,4]))
#  geom_tile() +
#  scale_fill_gradientn(colours =Zinc)

write.csv(pointmatrix_df, "/Users/delmelle/Dropbox (UNC Charlotte)/WRRI2000/maps/results/ArsenicB.csv",
```

8. Estimating the RMSE

We can also estimate the root mean square error between our predicted and observed values, and map the location of the errors.

```
prediction_gridRMSE <-cbind(data_geo$coords[,1] , data_geo$coords[,2])
kcb <- krige.conv(data_geo, loc=prediction_gridRMSE,
                  krige=krige.control(obj.model=exp_fit_nofix),
                  borders=gaston_border)
```

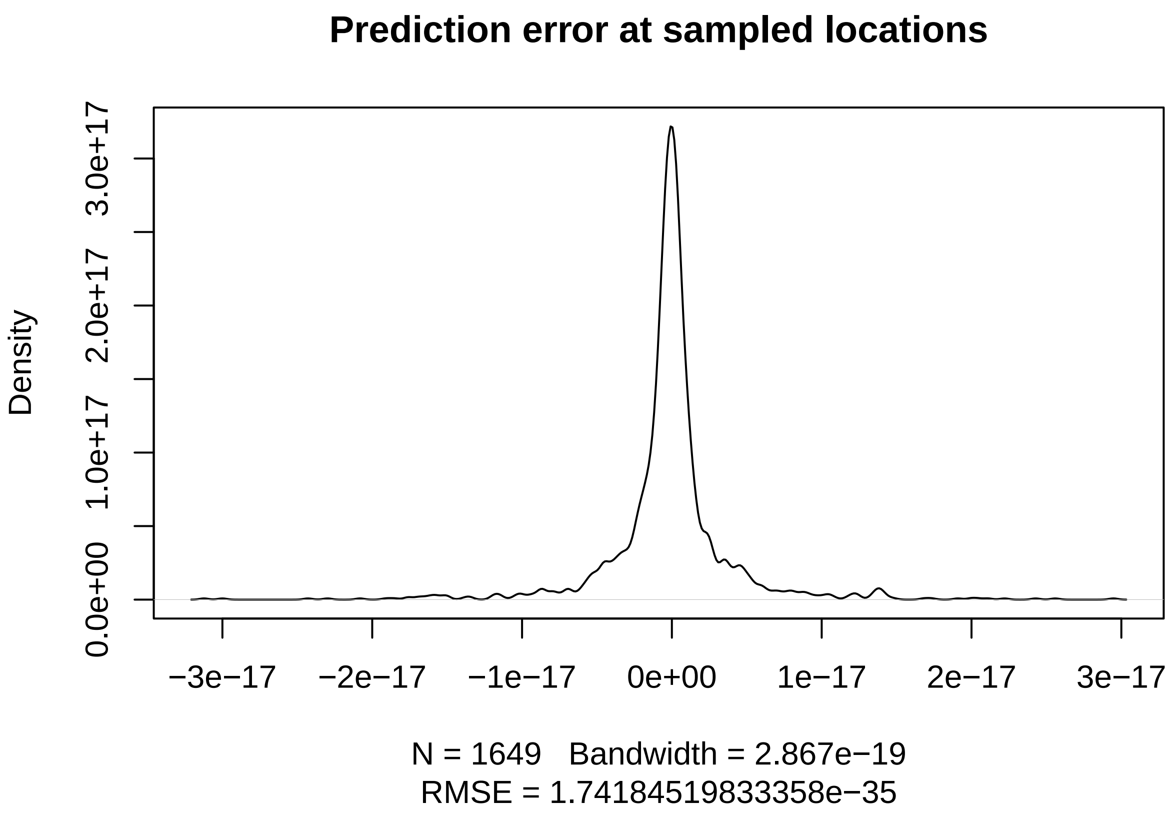
```
## krige.conv: results will be returned only for prediction locations inside the borders
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

```
df <-data.frame(data_geo)
a<-(data_geo$data - kcb$predict)
```

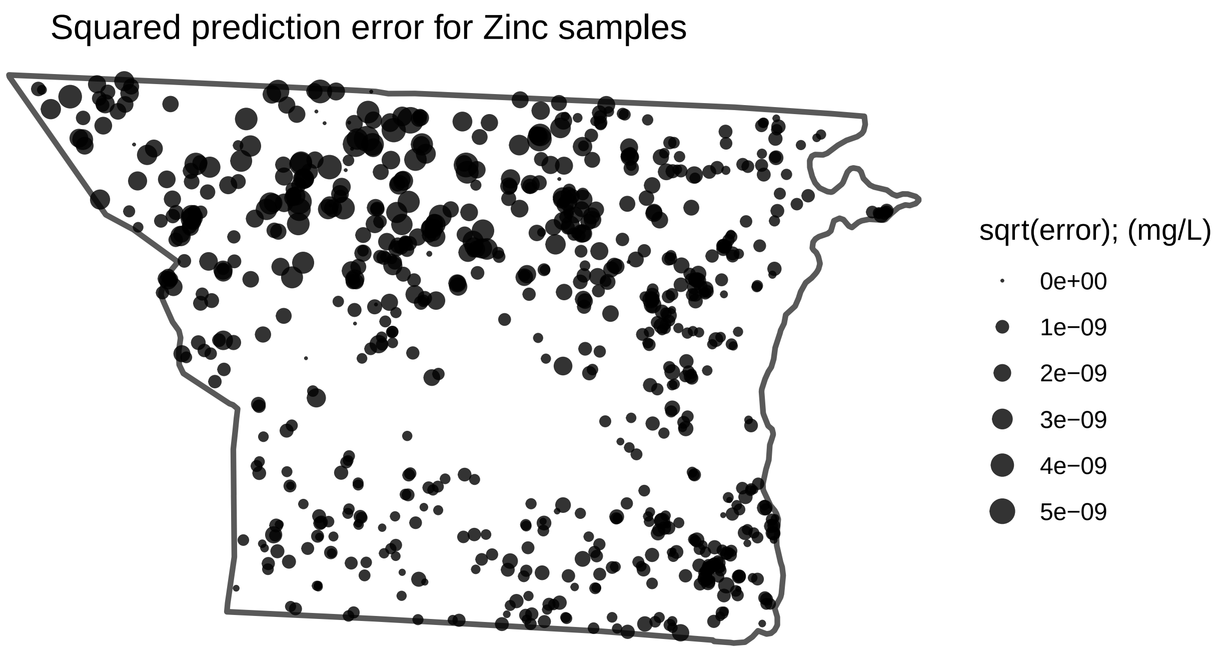
```r
df$a <-a
b<-sum((data_geo$data - kcb$predict)^2)/length(data_geo$data)
c<-((data_geo$data - kcb$predict)/data_geo$data)^2
df$c <-c
titleB <-cat("RMSE = ", b, "\n")
```

```
## RMSE =  1.741845e-35
```

```r
plot(density(a), main="Prediction error at sampled locations",
     sub = paste("RMSE = ", b, sep = ""))
```



```r
map <- ggplot(df) +
  geom_sf(data = gaston_sf, lwd = 1, fill = NA) +
  geom_point(data = df, aes(x = Coord1, y= Coord2, size = sqrt(a)), alpha=0.8)+
  scale_size_continuous(range = c(0.01,4), name="\nsqrt(error); (mg/L)") +
ggtitle("         Squared prediction error for Zinc samples")
map + theme_void()
```


Squared prediction error for Zinc samples
sqrt(error); (mg/L)
0e+00
1e−09
2e−09
3e−09
4e−09
5e−09