# Kriging method

Lecture 10

Joaquin Cavieres

## 1. Meuse data

We will work with the "meuse" data from the "sp" package:

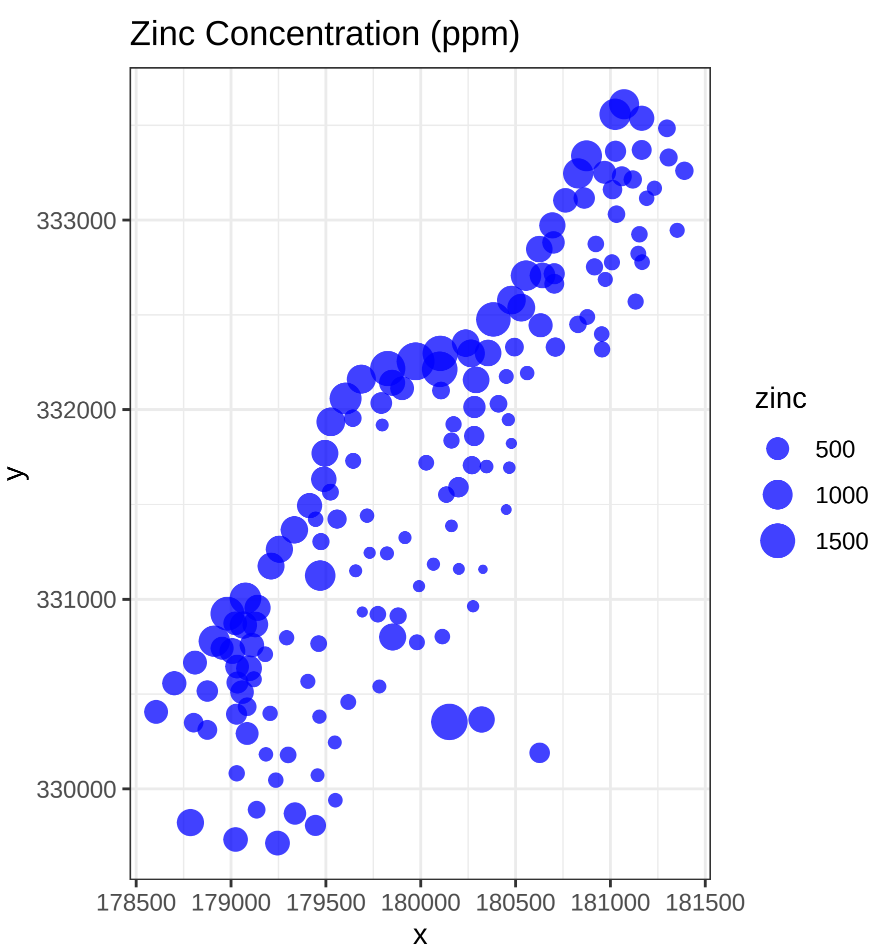
```
# Libraries for spatial data analysis
library(sp)
library(gstat)

# Data for data manipulation and plots
library(dplyr)
library(ggplot2)
library(gridExtra)
library(scales)
library(magrittr)

# Load the meuse data from the sp package
data(meuse)
summary(meuse)
```

```
##        x                y             cadmium           copper      
##  Min.   :178605   Min.   :329714   Min.   : 0.200   Min.   : 14.00  
##  1st Qu.:179371   1st Qu.:330762   1st Qu.: 0.800   1st Qu.: 23.00  
##  Median :179991   Median :331633   Median : 2.100   Median : 31.00  
##  Mean   :180005   Mean   :331635   Mean   : 3.246   Mean   : 40.32  
##  3rd Qu.:180630   3rd Qu.:332463   3rd Qu.: 3.850   3rd Qu.: 49.50  
##  Max.   :181390   Max.   :333611   Max.   :18.100   Max.   :128.00  
##                                                                     
##       lead            zinc             elev             dist        
##  Min.   : 37.0   Min.   : 113.0   Min.   : 5.180   Min.   :0.00000  
##  1st Qu.: 72.5   1st Qu.: 198.0   1st Qu.: 7.546   1st Qu.:0.07569  
##  Median :123.0   Median : 326.0   Median : 8.180   Median :0.21184  
##  Mean   :153.4   Mean   : 469.7   Mean   : 8.165   Mean   :0.24002  
##  3rd Qu.:207.0   3rd Qu.: 674.5   3rd Qu.: 8.955   3rd Qu.:0.36407  
##  Max.   :654.0   Max.   :1839.0   Max.   :10.520   Max.   :0.88039  
```

```
##
##         om           ffreq  soil   lime       landuse       dist.m
##   Min.    : 1.000   1:84   1:97   0:111   W      :50   Min.    :  10.0
##   1st Qu.: 5.300   2:48   2:46   1: 44   Ah     :39   1st Qu.:  80.0
##   Median : 6.900   3:23   3:12           Am     :22   Median : 270.0
##   Mean   : 7.478                         Fw     :10   Mean   : 290.3
##   3rd Qu.: 9.000                         Ab     : 8   3rd Qu.: 450.0
##   Max.   :17.000                         (Other):25   Max.   :1000.0
##   NA's   :2                              NA's   : 1
```

As we know, the meuse dataset contains concentration measurements for a number of chemical elements taken from the Meuse river in the Netherlands. We will visually inspect how zinc varies over the domain of interest where we map concentration to point size:

```
meuse %>% as.data.frame %>%
          ggplot(aes(x, y)) + geom_point(aes(size=zinc), color="blue", alpha=3/4) +
          ggtitle("Zinc Concentration (ppm)") + coord_equal() + theme_bw()
```



Ok, right know we will convert the meuse data (it is a `data.frame`) to a `SpatialPoinstDataFrame` as follow:

```
coordinates(meuse) <- ~ x + y
class(meuse)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

Remember that in a `SpatialPoinstDataFrame` object we have the following features:

- data: contains all the variables associated with different spatial locations
- coords.nrs: contains the column numbers of the spatial coordinates in the dataframe
- coords: a matrix of all spatial locations with corresponding values in the dataframe
- bbox: is the bounding box, that is, four points (or "corners") which denote the spatial extent of the data
- proj4string: contains the projection information, that is, what projection are the coordinates in.

## 2. Fitting a variogram

To calculate the kriging, you must first have a variogram model, from which the data can be interpolated. Thus, you should follow the next steps:

- Calculate the empirical variogram. This is done with the `variogram()` function.
- Fit a model to the empirical variogram.

The `variogram()` function has two arguments: the first being denoting how one or more variables interact spatially, and the second is an `SpatialPoinstDataFrame` where those variables reside.

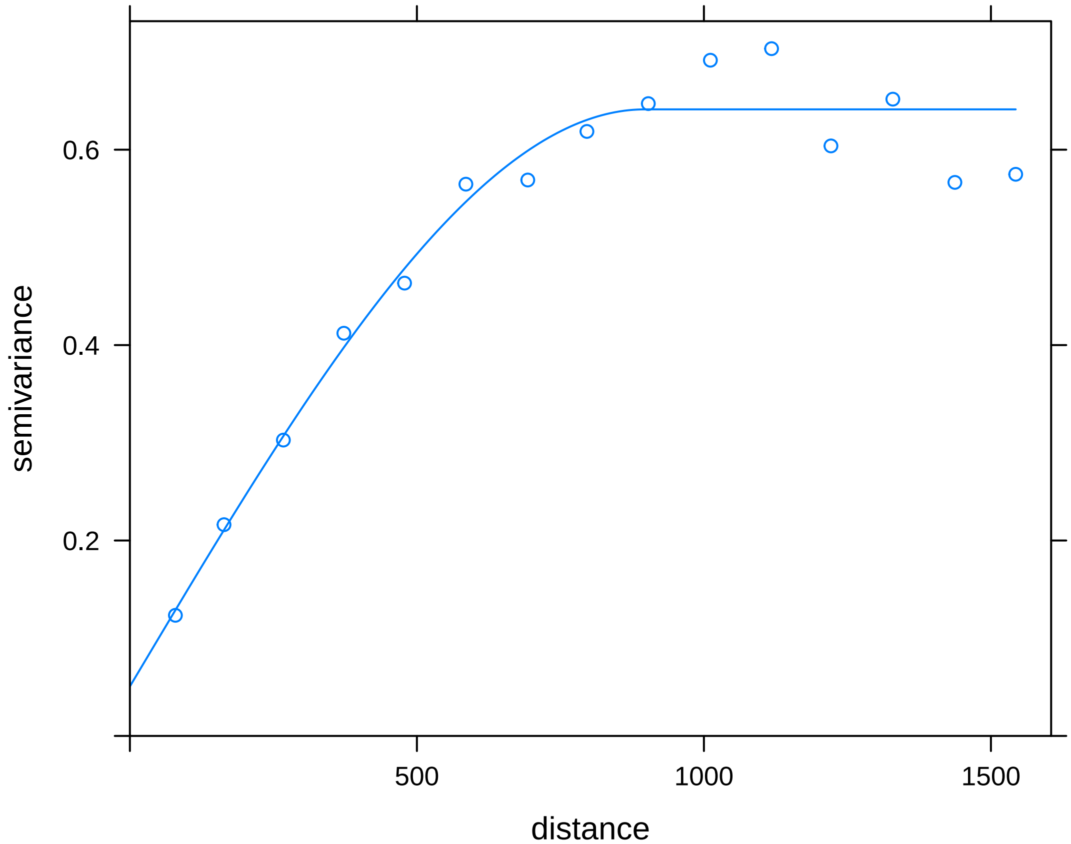
Now, the `fit.variogram()` function, a empirical variogram is the first argument (the object saved previously). The second argument is the model, with parameters, to be fit to the empirical variogram.

So, we will start with the analysis:

```
vario_lzinc      <- variogram(log(zinc)~1, meuse)             # calculates EV values
vario_lzinc_fit_sph <- fit.variogram(vario_lzinc,
                                 model=vgm(1, "Sph", 900, 1)) # fit model to the EV
```

We have a model fitted to the empirical variogram, so we can make a plot to see how well the fit was:

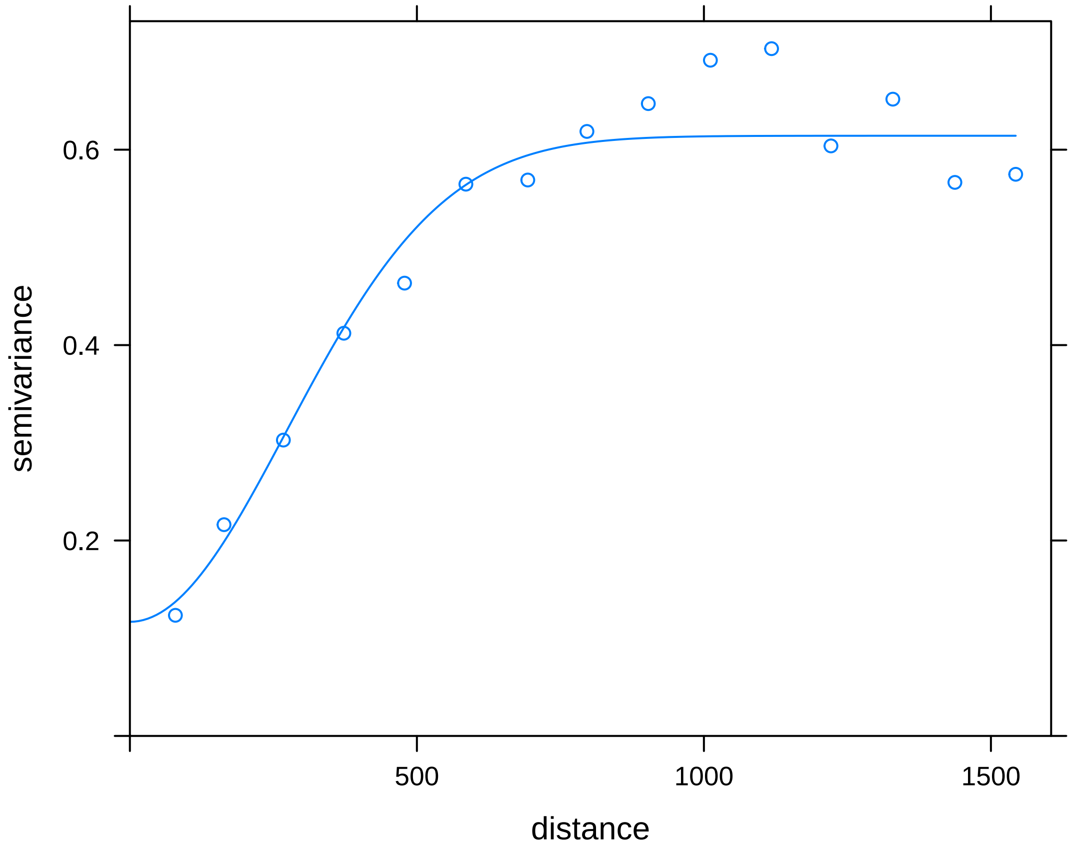
```
plot(vario_lzinc, vario_lzinc_fit_sph) # plot the empirical values, along with the fit model
```




With purposes of evaluation of the fitted model, we will fit another variogram model using a different covariance function. In the first case we used a "spherical" covariance function but now we will use the "gaussian" covariance function as:

```
vario_lzinc_fit_gauss <- fit.variogram(vario_lzinc,
                                       model=vgm(1, "Gau", 900, 1)) # fit model to the EV
```

```
plot(vario_lzinc, vario_lzinc_fit_gauss)
```



# 3. Model evaluation

## 3.1. RMSE

We can assess the predictive power of the model by using the root-mean-square error (RMSE) as model assessment metric.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{1}$$

```
RMSE <- function(residuals){
  sqrt(sum((residuals)^2)/length(residuals))
}
```

## 3.2. Leave-One-Out Cross-Validation

The Leave-One-Out Cross-Validation, or LOOCV, procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. We can perform this analysis using the function `krige.cv()` of the "gstat" package to do kriging cross validation.

```
lzinc_krig_cv_sph <- krige.cv(formula = log(zinc) ~ 1,
                              locations = meuse,
                              model = vario_lzinc_fit_sph)
```

and now for the variogram fitted using the Gaussian covariance function:

```
lzinc_krig_cv_gauss <- krige.cv(formula = log(zinc) ~ 1,
                                locations = meuse,
                                model = vario_lzinc_fit_gauss)
```

and then check the RMSE for both models:

```
RMSE(residuals = lzinc_krig_cv_sph@data$residual)
```

```
## [1] 0.3918035
```

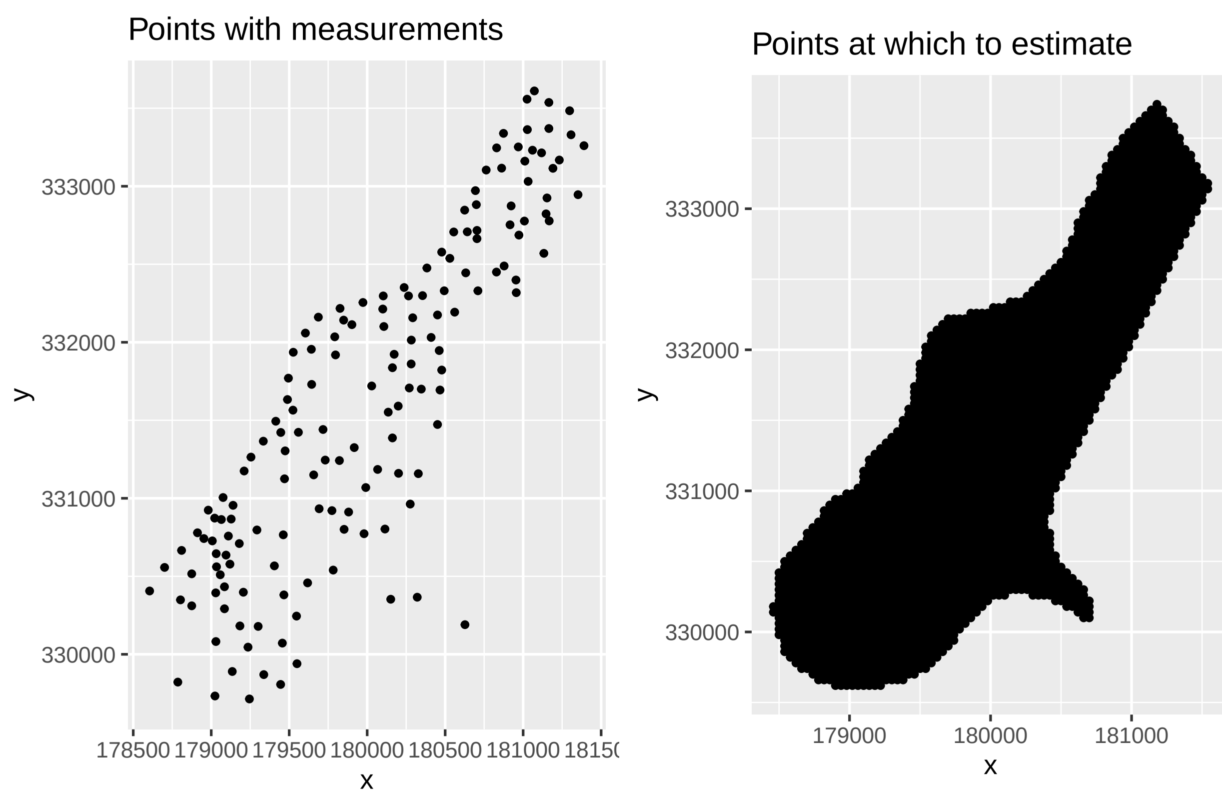
```
RMSE(residuals = lzinc_krig_cv_gauss@data$residual)
```

```
## [1] 0.3980179
```

## 4. Kriging

Considering the definition of "interpolation", it is the predicted values at points we don't have measurements. Thus, we need two spatial domains: one having values associated with the points, and one for which we want estimates. In this example, the spatial domains we use are those of "meuse" and the spatial domain to predict is the "meuse.grid":

```
# Spatial domain to interpolate
data("meuse.grid")

# We can make a comparison between the meuse data and the meuse.grid
p1 <- meuse %>% as.data.frame %>%
  ggplot(aes(x, y)) + geom_point(size=1) + coord_equal() +
  ggtitle("Points with measurements")

# this is clearly gridded over the region of interest
p2 <- meuse.grid %>% as.data.frame %>%
  ggplot(aes(x, y)) + geom_point(size=1) + coord_equal() +
  ggtitle("Points at which to estimate")


grid.arrange(p1, p2, ncol = 2)
```



### 3.1. Interpolation

havind loaded the grid to make the interpolation (`meuse.grid`), we are now ready to sue the Kriging. This can be done with the `krige()` function of the package "gstat", which commonly uses four arguments in the function:

- The model formula.
- An `SpatialPoinstDataFrame` of the spatial domain that has measurements.
- An `SpatialPoinstDataFrame` of the spatial domain to krige over.
- A variogram model fitted to the data.

Considering the previous points, we can peform the Kriging as:

```
coordinates(meuse.grid) <- ~ x + y
lzinc_krig_sph <- krige(log(zinc) ~ 1, meuse, meuse.grid, model = vario_lzinc_fit_sph)
```

```
## [using ordinary kriging]
```

for the "spherical" model. The object `lzinc_krig` contains the following information:

```
summary(lzinc_krig_sph)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##      min    max
## x 178460 181540
## y 329620 333740
## Is projected: NA
## proj4string : [NA]
## Number of points: 3103
## Data attributes:
##    var1.pred        var1.var
##  Min.   :4.777   Min.   :0.08549
##  1st Qu.:5.238   1st Qu.:0.13729
##  Median :5.573   Median :0.16218
##  Mean   :5.707   Mean   :0.18533
##  3rd Qu.:6.172   3rd Qu.:0.21162
##  Max.   :7.440   Max.   :0.50028
```

and the same for the Gaussian model:

```
lzinc_krig_gauss <- krige(log(zinc) ~ 1, meuse, meuse.grid, model = vario_lzinc_fit_gauss)
```

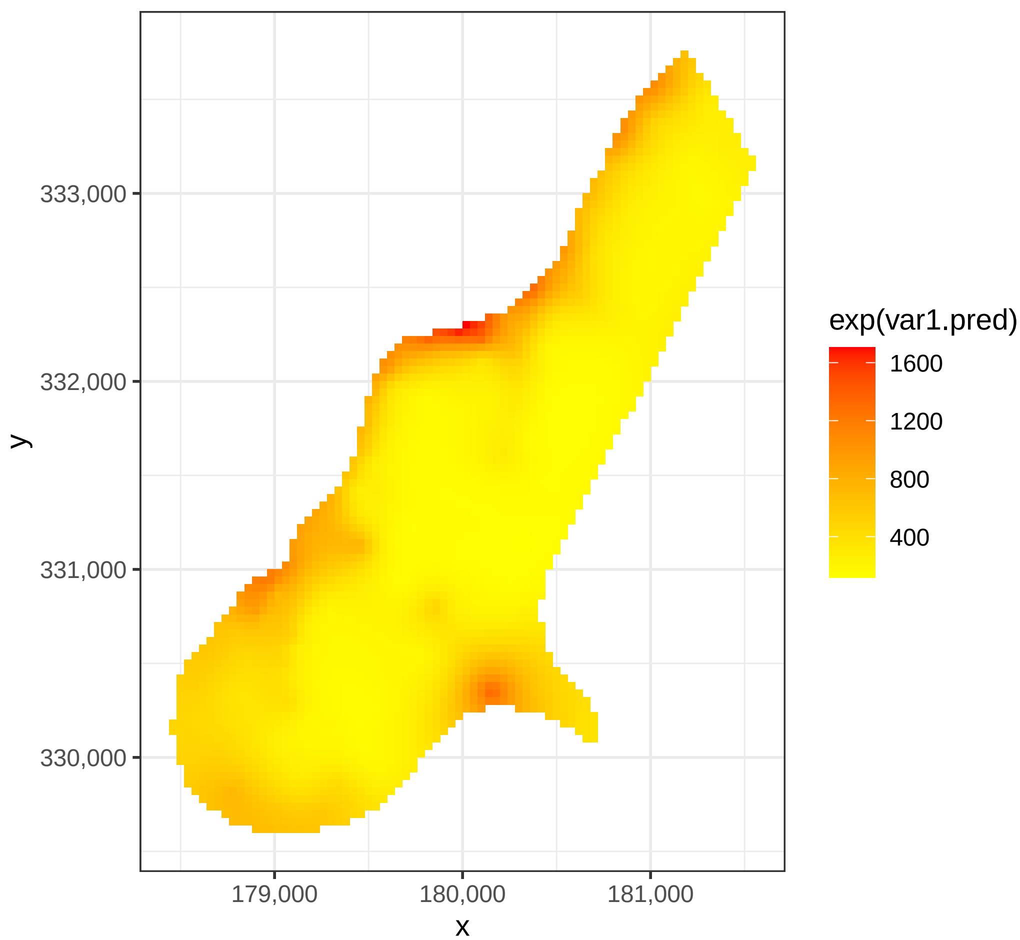
```
## [using ordinary kriging]
```

```r
summary(lzinc_krig_gauss)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##      min    max
## x 178460 181540
## y 329620 333740
## Is projected: NA
## proj4string : [NA]
## Number of points: 3103
## Data attributes:
##    var1.pred        var1.var
##  Min.   :4.777   Min.   :0.1311
##  1st Qu.:5.260   1st Qu.:0.1477
##  Median :5.566   Median :0.1602
##  Mean   :5.706   Mean   :0.1862
##  3rd Qu.:6.164   3rd Qu.:0.1962
##  Max.   :7.386   Max.   :0.5389
```

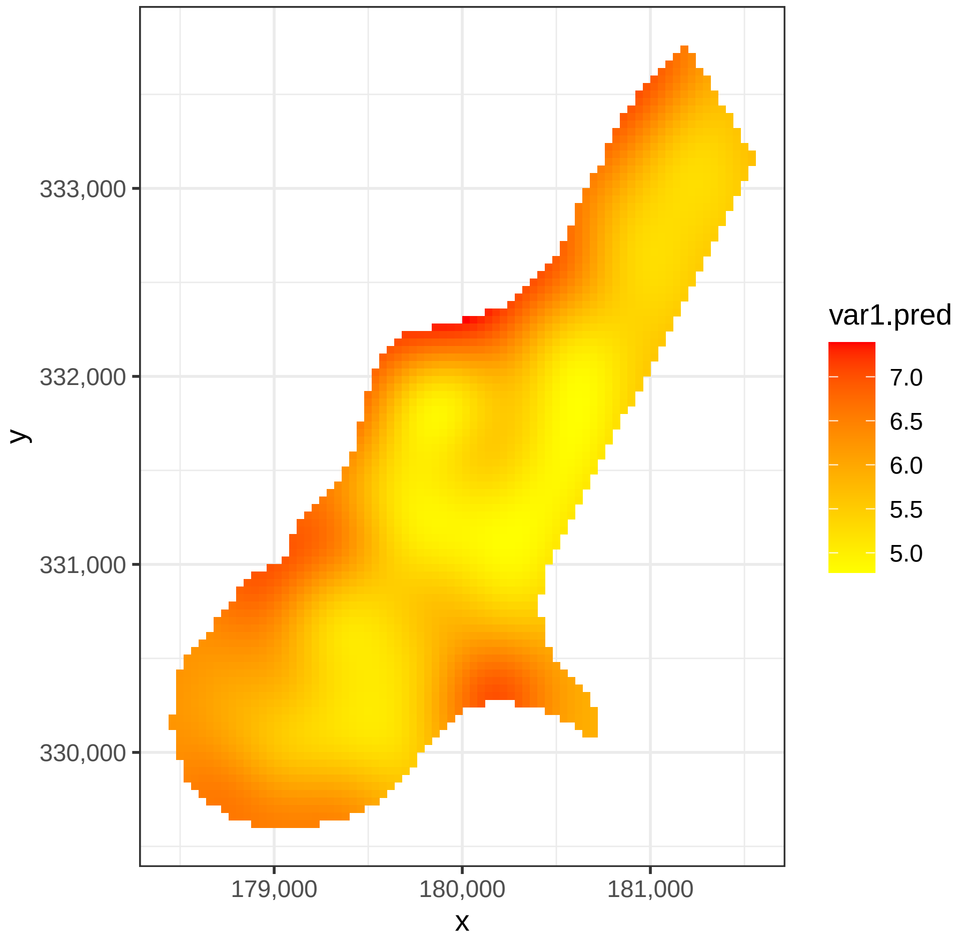
We can make a plot to visualize the interpolation for the spherical model

```
lzinc_krig_sph %>% as.data.frame %>%
  ggplot(aes(x=x, y=y)) + geom_tile(aes(fill=exp(var1.pred))) + coord_equal() +
  scale_fill_gradient(low = "yellow", high="red") +
  scale_x_continuous(labels=comma) + scale_y_continuous(labels=comma) +
  theme_bw()
```



and for the Gaussian model

```
lzinc_krig_gauss %>% as.data.frame %>%
  ggplot(aes(x=x, y=y)) + geom_tile(aes(fill=var1.pred)) + coord_equal() +
  scale_fill_gradient(low = "yellow", high="red") +
  scale_x_continuous(labels=comma) + scale_y_continuous(labels=comma) +
  theme_bw()
```




## References


- Moraga, P., 2023. Spatial Statistics for Data Science: Theory and Practice with R
- Bivand, R. S., Pebesma, E. J., Gomez-Rubio, V., & Pebesma, E. J. (2008). Applied spatial data analysis with R (Vol. 747248717, pp. 237-268). New York: Springer.
- Spatial Data Science with R and "terra". https://rspatial.org/index.html.