Areal Data and Proximity

HES 505 Fall 2023: Session 20

Matt Williamson

Objectives

By the end of today you should be able to:

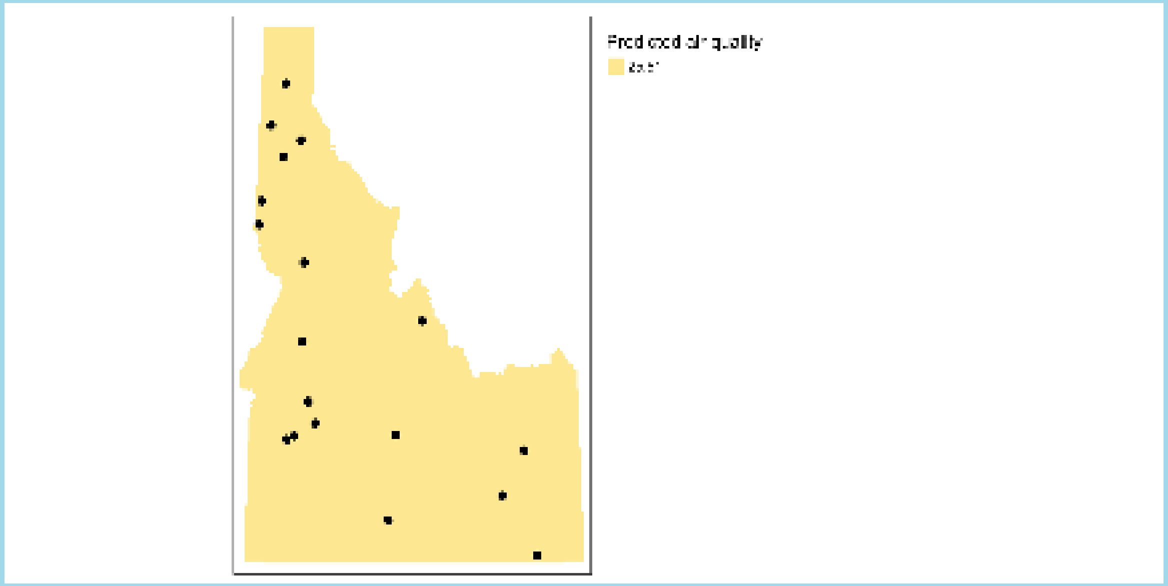
Statistical Interpolation

Statistical Interpolation

Trend Surface Modeling

- Basically a regression on the coordinates of your data points
- Coefficients apply to the coordinates and their interaction
- Relies on different functional forms

0th Order Trend Surface



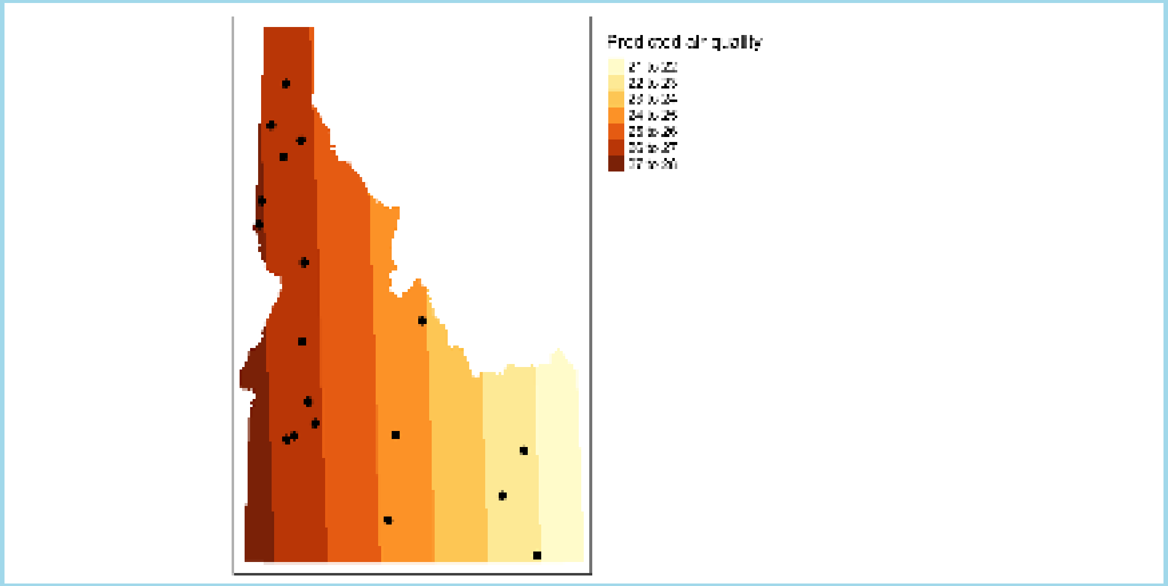
- Simplest form of trend surface
- where is the mean value of air quality
- Result is a simple horizontal surface where all values are the same.

0th order trend surface

```
#set up interpolation grid
# Create an empty grid where n is the total number of cells
grd <- as.data.frame(spsample(as(id.cty, "Spatial"), "regular", n=20000))
names(grd)       <- c("X", "Y")
coordinates(grd) <- c("X", "Y")
gridded(grd)     <- TRUE  # Create SpatialPixel object
fullgrid(grd)    <- TRUE  # Create SpatialGrid object
proj4string(grd) <- proj4string(as(aq.sum, "Spatial"))
# Define the polynomial equation
f.0  <- as.formula(meanpm25 ~ 1)

# Run the regression model
lm.0 <- lm( f.0 , data=aq.sum)

# Use the regression model output to interpolate the surface
dat.0th <- SpatialGridDataFrame(grd, data.frame(var1.pred = predict(lm.0, n

# Convert to raster object to take advantage of rasterVis' imaging
```

1st Order Trend Surface

- Creates a slanted surface
-
- X and Y are the coordinate pairs

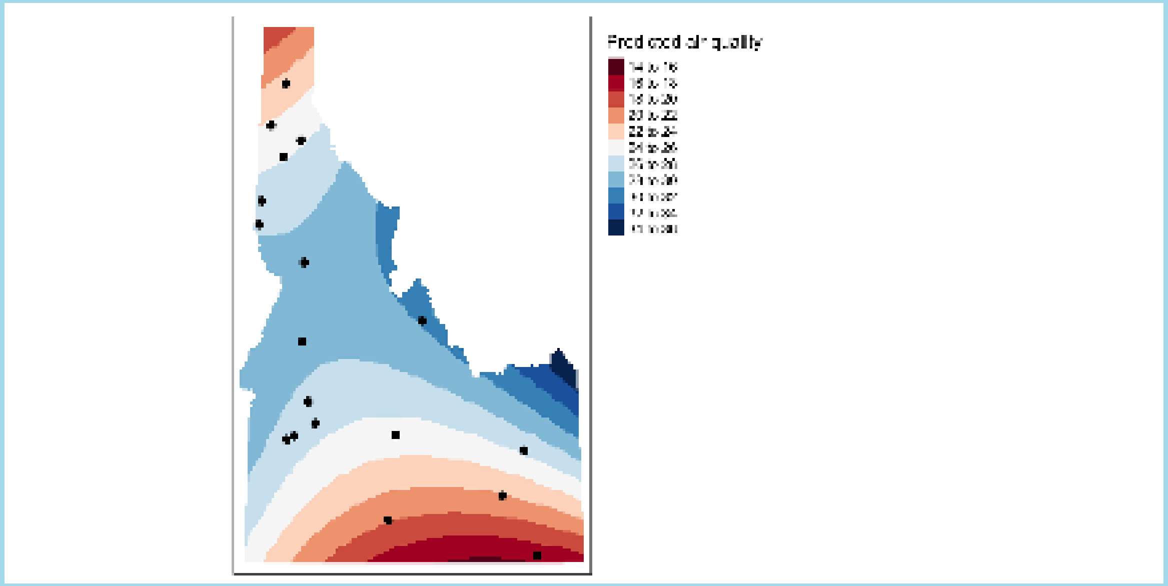


1st Order Trend Surface

```r
# Define the polynomial equation
f.1  <- as.formula(meanpm25 ~ X + Y)

aq.sum$X <- st_coordinates(aq.sum)[,1]
aq.sum$Y <- st_coordinates(aq.sum)[,2]

# Run the regression model
lm.1 <- lm( f.1 , data=aq.sum)

# Use the regression model output to interpolate the surface
dat.1st <- SpatialGridDataFrame(grd, data.frame(var1.pred = predict(lm.1, n

# Convert to raster object to take advantage of rasterVis' imaging
# environment
r   <- rast(dat.1st)
r.m <- mask(r, st_as_sf(id.cty))
```

2nd Order Trend Surfaces

- Produces a parabolic surface
-
- Highlights the interaction of both directions



2nd Order Trend Surfaces

```r
# Define the 1st order polynomial equation
f.2 <- as.formula(meanpm25 ~ X + Y + I(X*X)+I(Y*Y) + I(X*Y))

# Run the regression model
lm.2 <- lm( f.2, data=aq.sum)

# Use the regression model output to interpolate the surface
dat.2nd <- SpatialGridDataFrame(grd, data.frame(var1.pred = predict(lm.2, n

r   <- rast(dat.2nd)
r.m <- mask(r, st_as_sf(id.cty))

tm_shape(r.m) + tm_raster(n=10, palette="RdBu", title="Predicted air qualit
  tm_legend(legend.outside=TRUE)
```

Kriging

- Previous methods predict as a (weighted) function of distance
- Treat the observations as perfect (no error)
- If we imagine that is the outcome of some spatial process such that:

then any observed value of is some function of the process () and some error ()

- Kriging exploits autocorrelation in to identify the trend and interpolate accordingly

Autocorrelation

- **Correlation** the tendency for two variables to be related
- **Autocorrelation** the tendency for observations that are closer (in space or time) to be correlated
- **Positive autocorrelation** neighboring observations have with the same sign
- **Negative autocorrelation** neighboring observations have with a different sign (rare in geography)

Ordinary Kriging

- Assumes that the deterministic part of the process () is an unknown constant ()

Steps for Ordinary Kriging

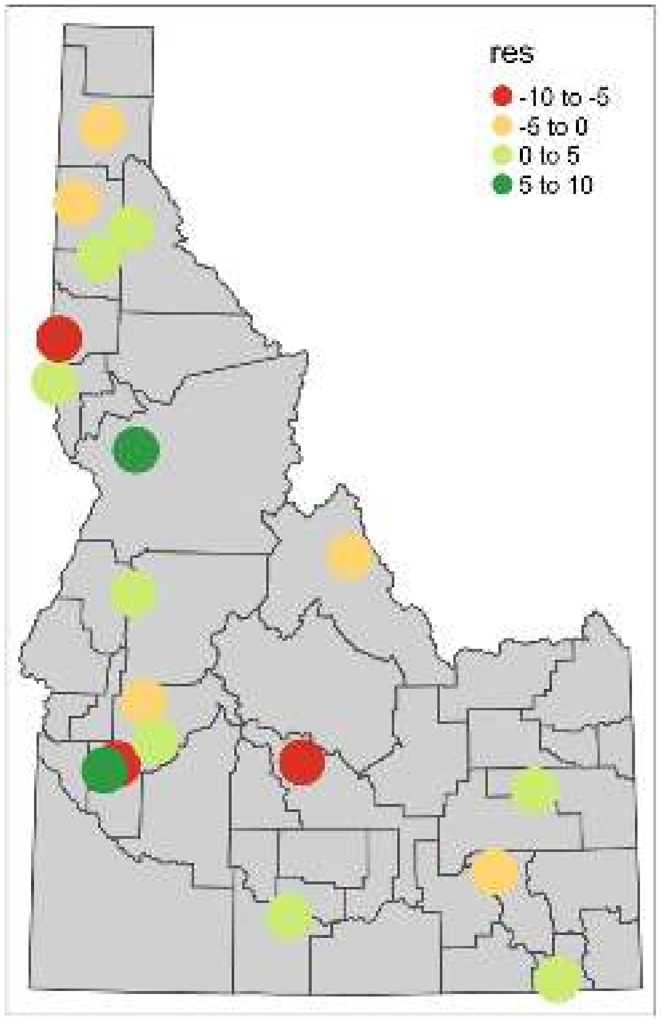
- Removing any **spatial trend** in the data (if present).
- Computing the **experimental variogram**, , which is a measure of spatial autocorrelation.
- Defining an **experimental variogram model** that best characterizes the spatial autocorrelation in the data.
- Interpolating the surface using the experimental variogram.
- Adding the kriged interpolated surface to the trend interpolated surface to produce the final output.

Removing Spatial Trend

- Mean and variance need to be constant across study area
- Trend surfaces indicate that is not the case
- Need to remove that trend

```
f.2 <- as.formula(meanpm25 ~ X + Y + I(X*X)+I(Y*Y) + I(X*Y))

# Run the regression model
lm.2 <- lm( f.2, data=aq.sum)

# Copy the residuals to the point object
aq.sum$res <- lm.2$residuals
```

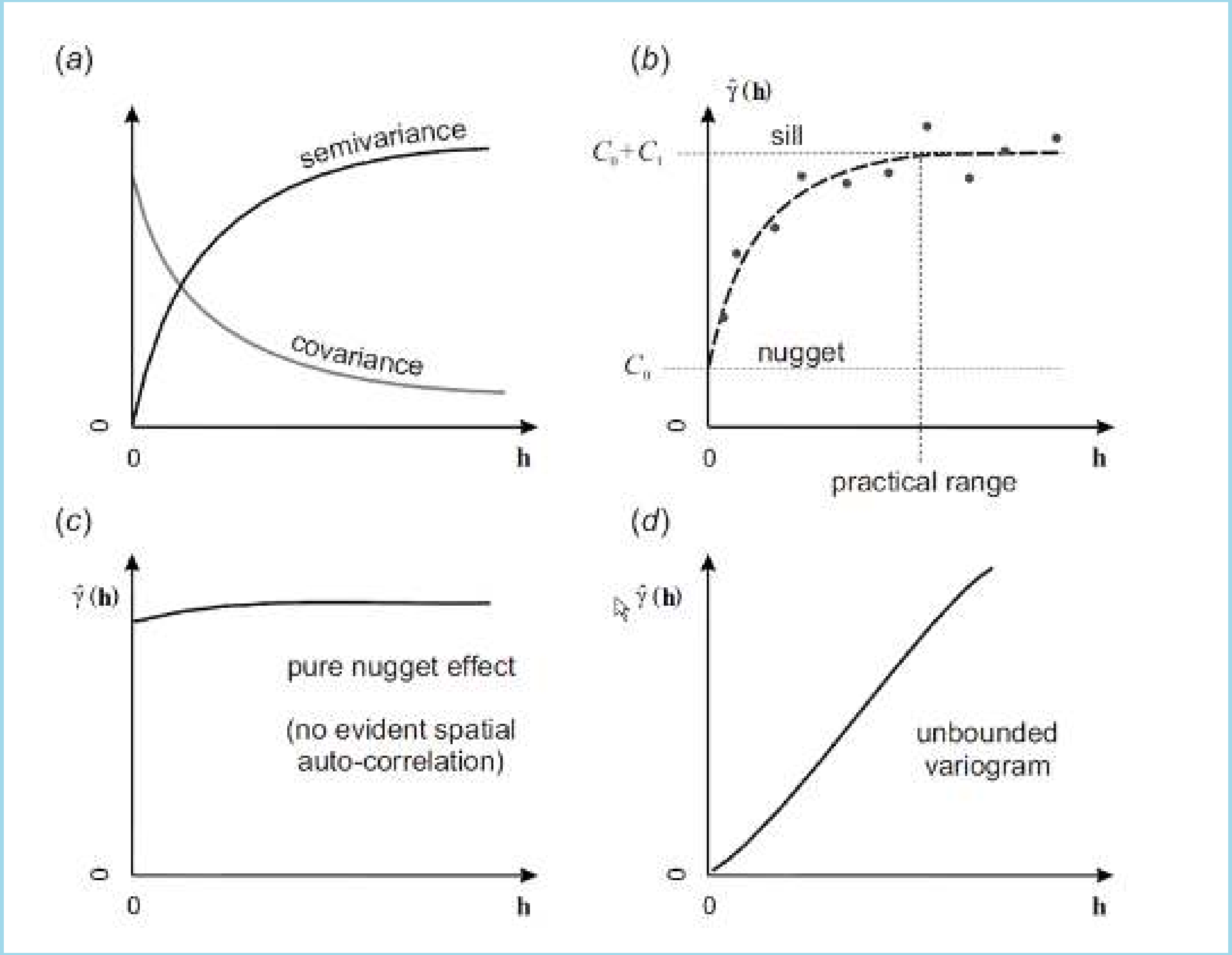
Removing the trend



Calculate the experimental variogram

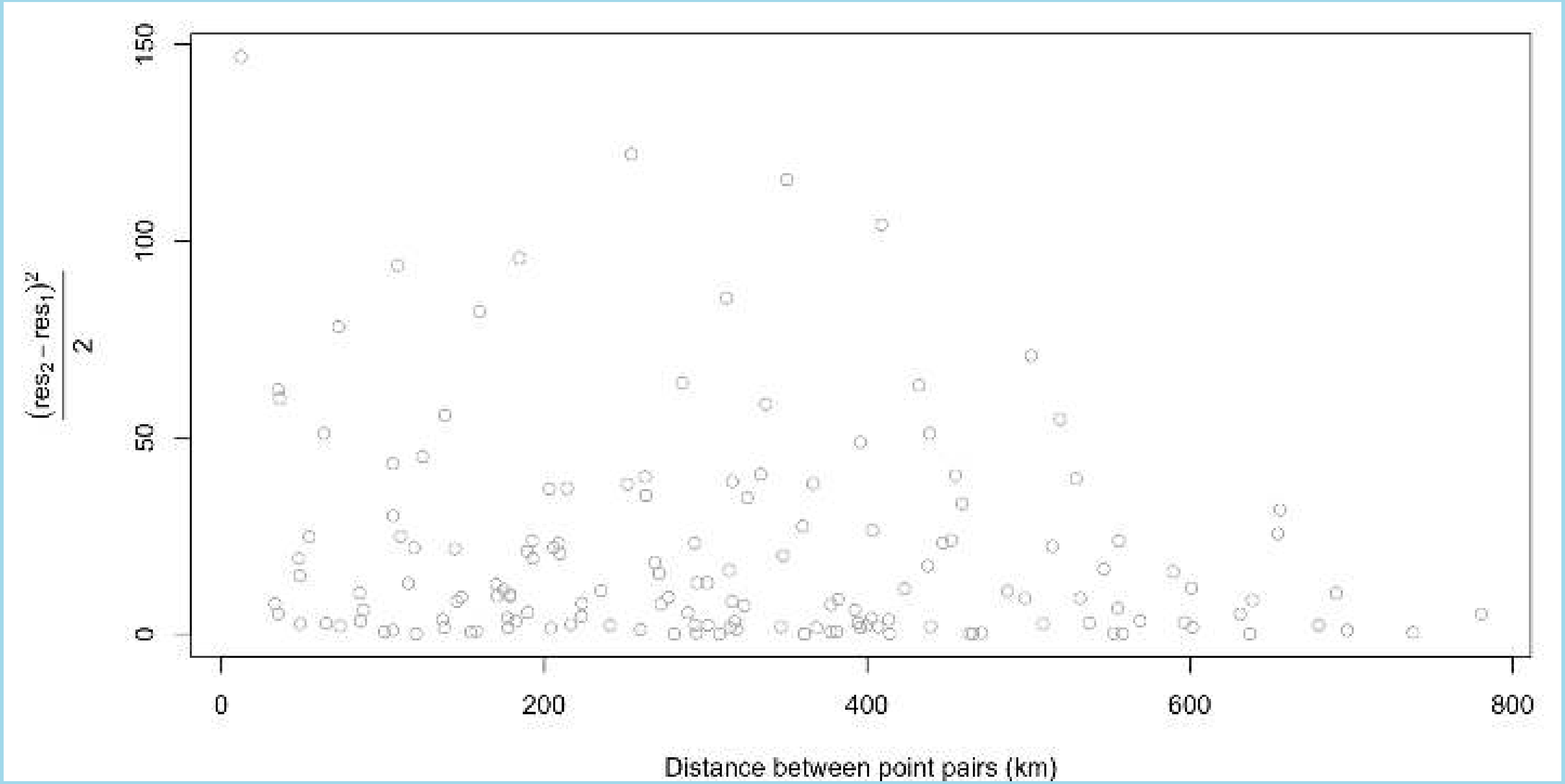
- **nugget** - the proportion of semivariance that occurs at small distances
- **sill** - the maximum semivariance between pairs of observations
- **range** - the distance at which the **sill** occurs
- **experimental** vs. **fitted** variograms

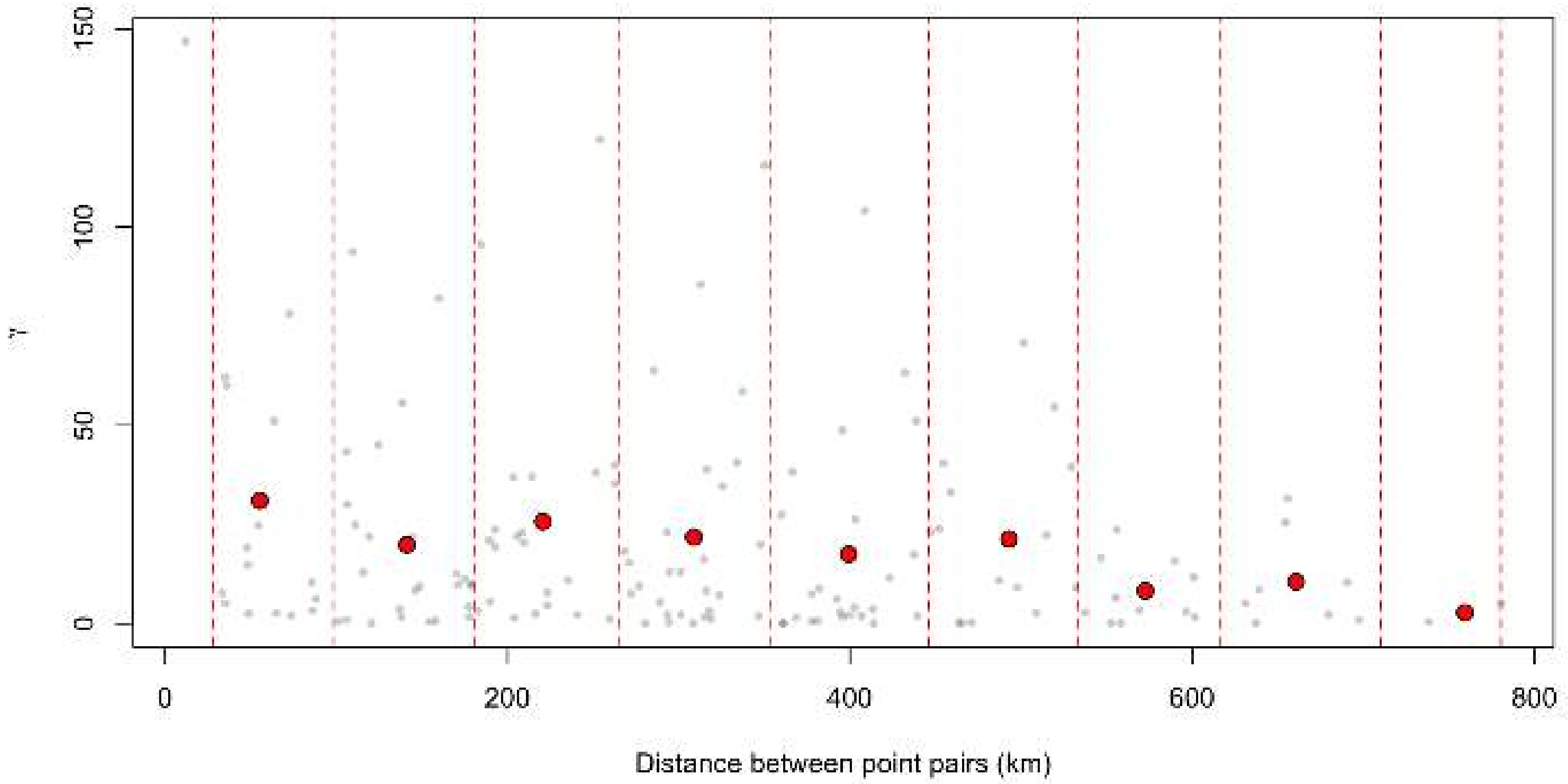
A Note about Semivariograms



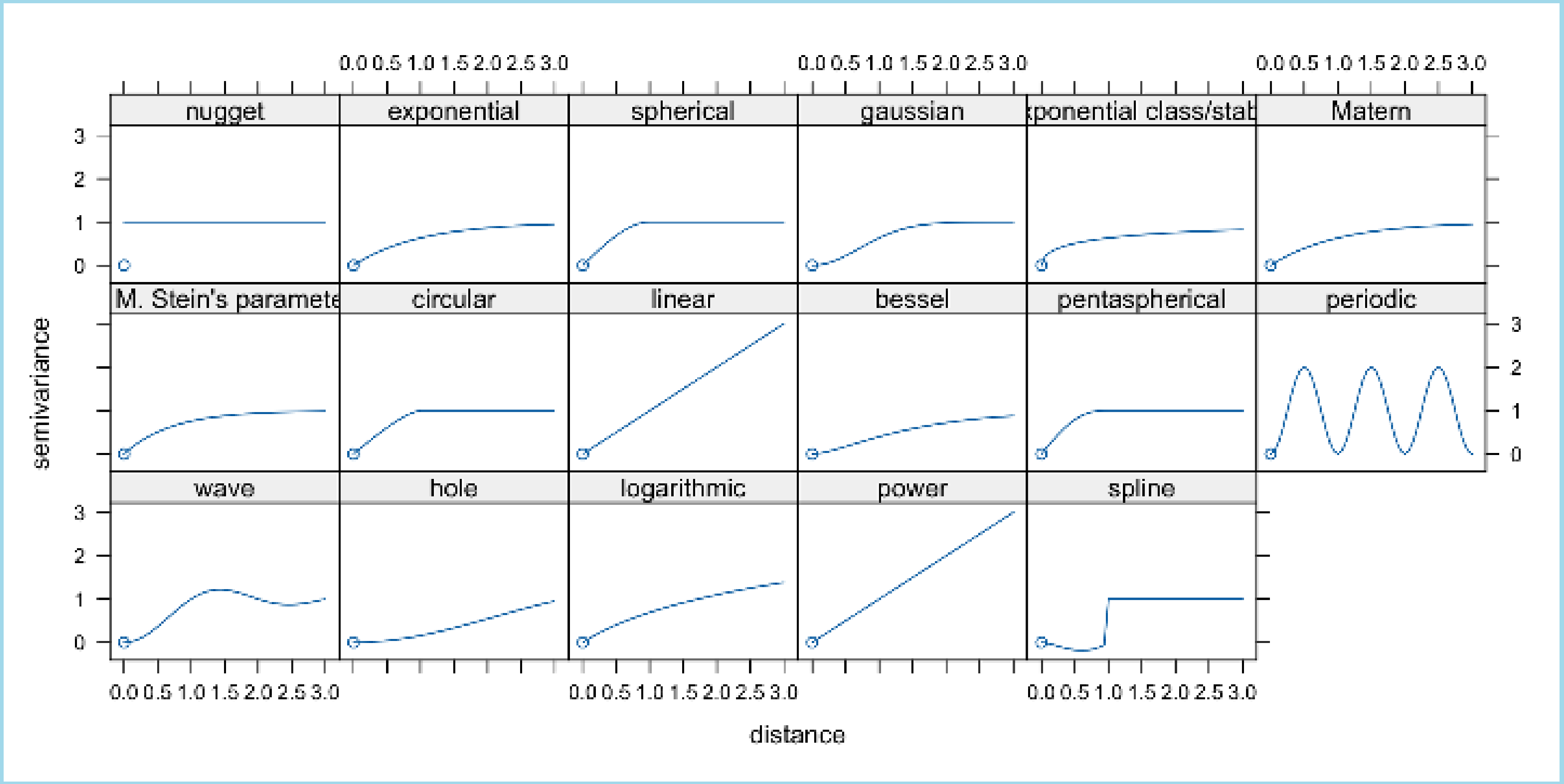
Fitted Semivariograms

Calculate the experimental variogram

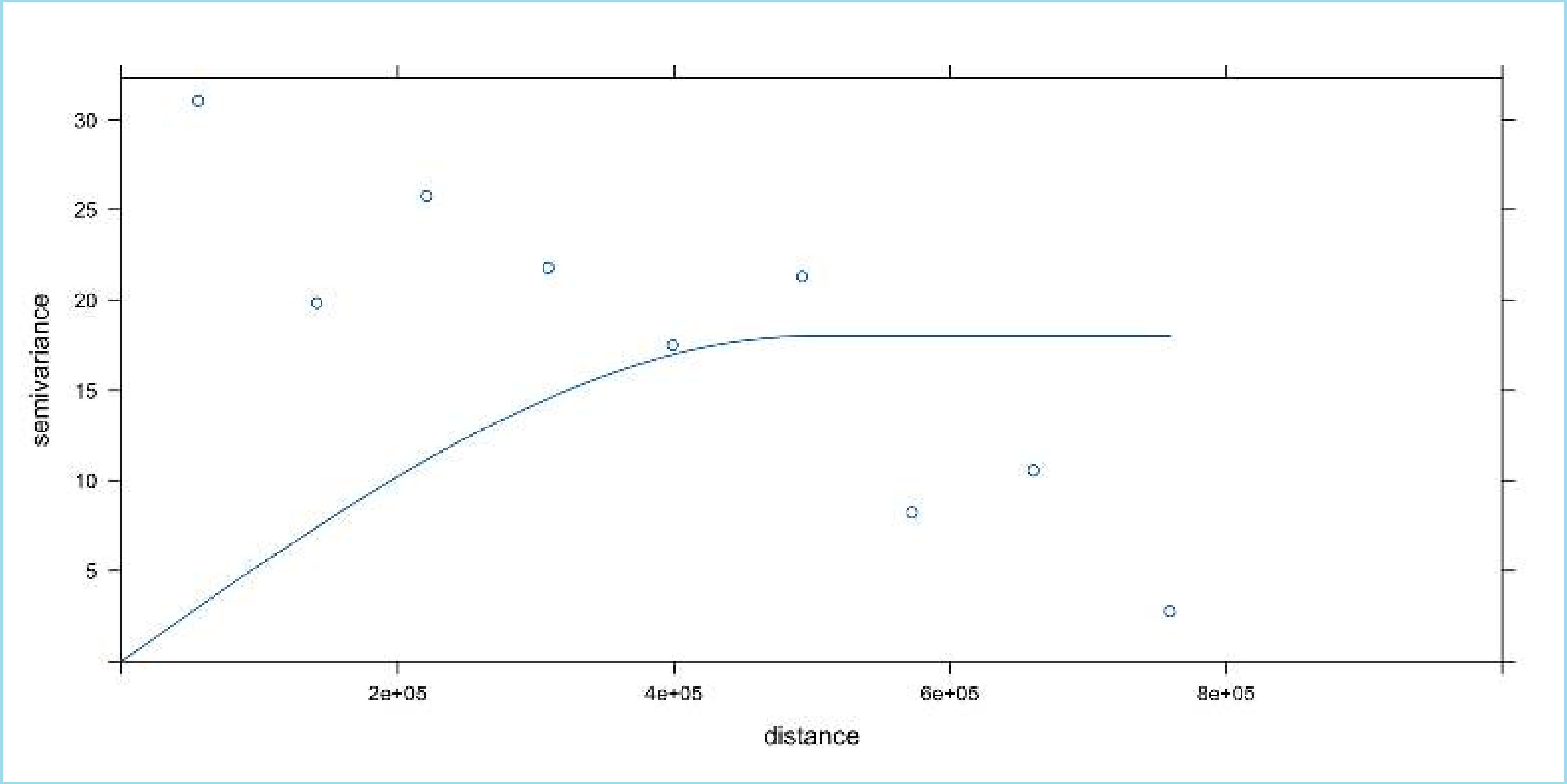




150
100
50
0
γ
0
200
400
600
800
Distance between point pairs (km)


nugget
exponential
spherical
gaussian
xponential class/stat
Matern
M. Stein's paramete
circular
linear
bessel
pentaspherical
periodic
wave
hole
logarithmic
power
spline
semivariance
distance
0.0 0.5 1.0 1.5 2.0 2.5 3.0
0 1 2 3

Looking at the sample Variogram

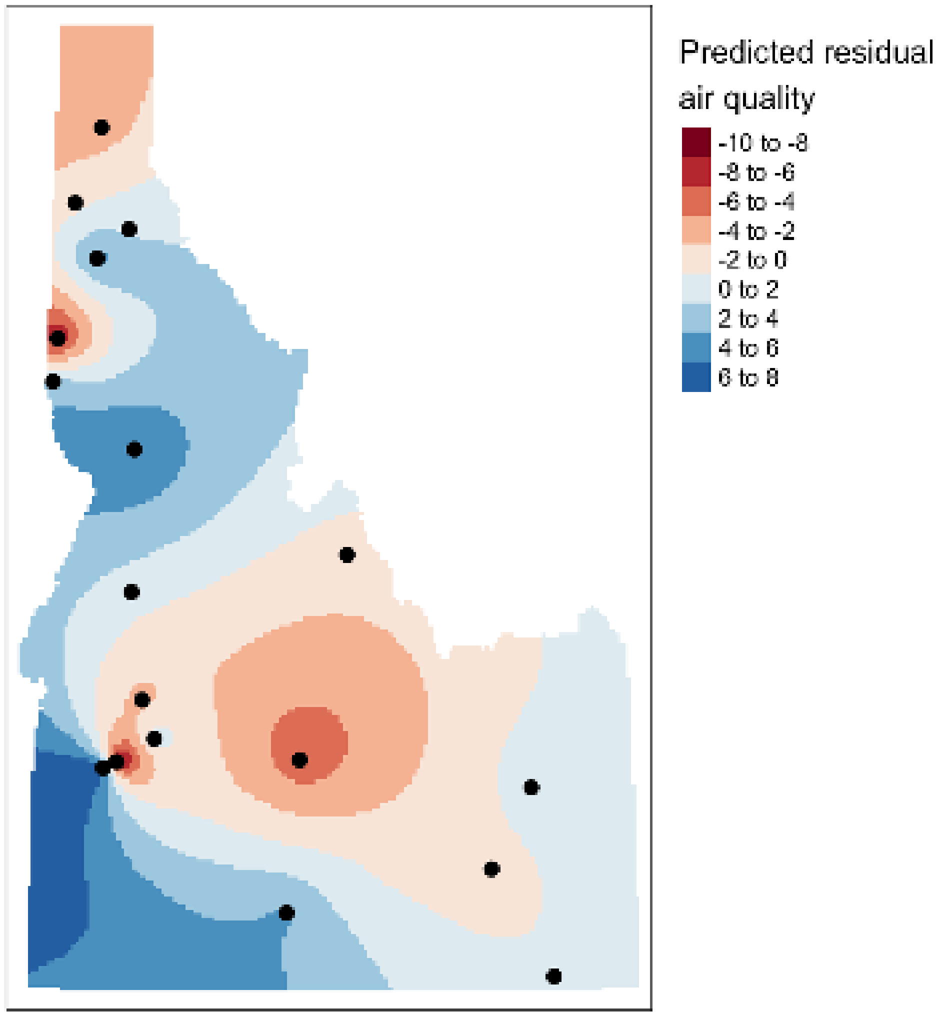


Estimating the sample variogram

```r
var.smpl <- gstat::variogram(f.2, aq.sum, cloud = FALSE, cutoff=1000000, wi


# Compute the variogram model by passing the nugget, sill and range values
# to fit.variogram() via the vgm() function.
dat.fit  <- gstat::fit.variogram(var.smpl, fit.ranges = FALSE, fit.sills =
```

Ordinary Kriging

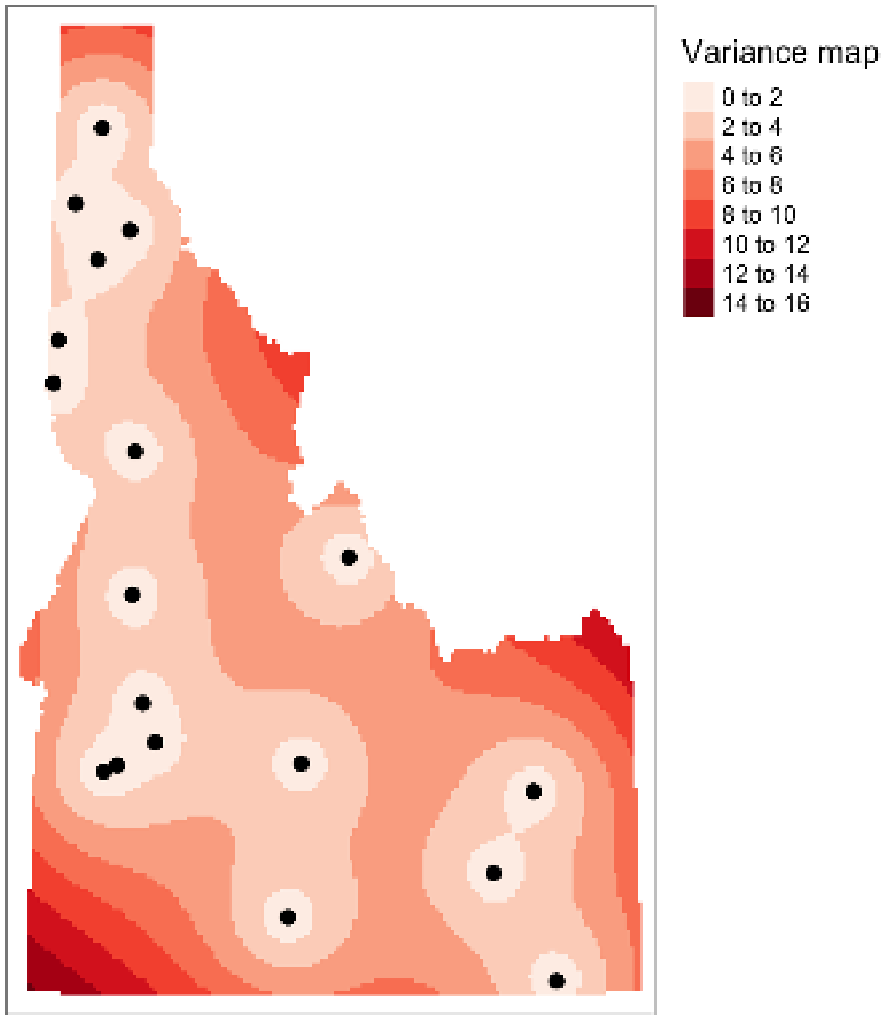
[using ordinary kriging]



Ordinary Kriging

```r
dat.krg <- gstat::krige( res~1, as(aq.sum, "Spatial"), grd, dat.fit)
```

Visualizing Uncertainty



Universal Kriging

- Assumes that the deterministic part of the process () is now a function of the location
- Could be the location or some other attribute
- Now y is a function of some aspect of x

```
vu <- variogram(log(zinc)~elev, ~x+y, data=meuse)
mu <- fit.variogram(vu, vgm(1, "Sph", 300, 1))
gUK <- gstat(NULL, "log.zinc", log(zinc)~elev, meuse, locations=~x+y, model
names(r) <- "elev"
UK <- interpolate(r, gUK, debug.level=0)
```

Universal Kriging

Universal Kriging

```r
vu <- variogram(log(zinc)~x + x^2 + y + y^2, ~x+y, data=meuse)
mu <- fit.variogram(vu, vgm(1, "Sph", 300, 1))
gUK <- gstat(NULL, "log.zinc", log(zinc)~x + x^2 + y + y^2, meuse, location
names(r) <- "elev"
UK <- interpolate(r, gUK, debug.level=0)
```

Universal Kriging

Co-Kriging

- relies on autocorrelation in for AND cross correlation with other variables ()
- Extending the ordinary kriging model gives:

* Note that there is autocorrelation within both and (because of the) and cross-correlation (because of the location,)

- Not required that all variables are measured at exactly the same points

Co-Kriging

- Process is just a linked series of `gstat` calls

```r
gCoK <- gstat(NULL, 'log.zinc', log(zinc)~1, meuse, locations=~x+y)
gCoK <- gstat(gCoK, 'elev', elev~1, meuse, locations=~x+y)
gCoK <- gstat(gCoK, 'cadmium', cadmium~1, meuse, locations=~x+y)
coV <- variogram(gCoK)
coV.fit <- fit.lmc(coV, gCoK, vgm(model='Sph', range=1000))

coK <- interpolate(r, coV.fit, debug.level=0)
```

Co-Kriging

Co-Kriging

A Note about Semivariograms