

Areal Data and Proximity

HES 505 Fall 2023: Session 20

Matt Williamson

Objectives

By the end of today you should be able to:

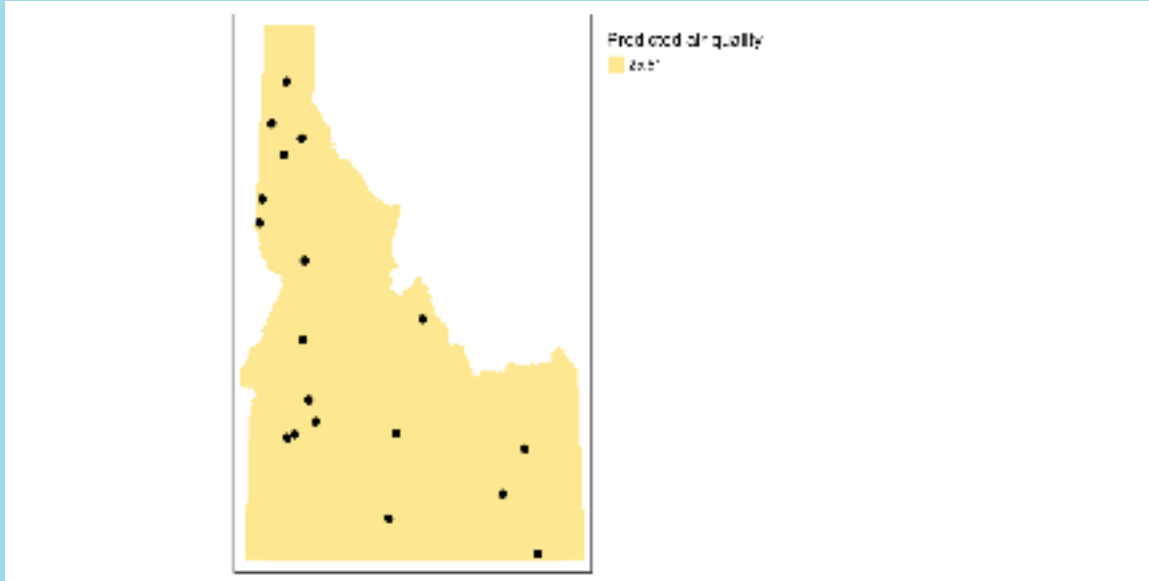
Statistical Interpolation

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



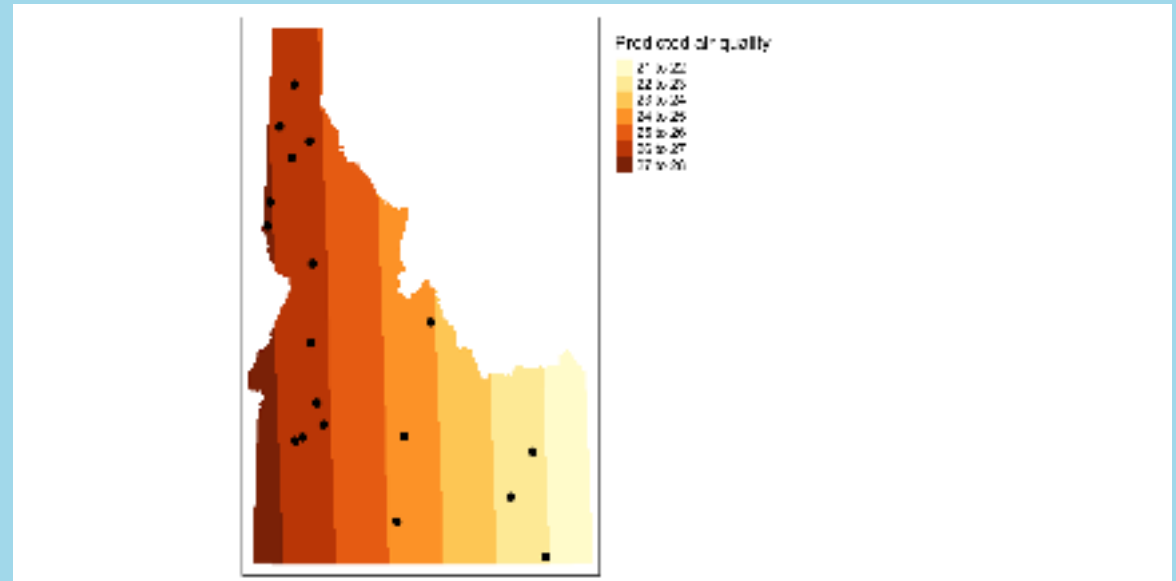
- Simplest form of trend surface
- $Z = a$ where a is the mean value of air quality
- Result is a simple horizontal surface where all values are the same.

0th order trend surface

```
1 #set up interpolation grid
2 # Create an empty grid where n is the total number of cells
3 grd <- as.data.frame(spsample(as(id.cty, "Spatial"), "regular", n=20000))
4 names(grd) <- c("X", "Y")
5 coordinates(grd) <- c("X", "Y")
6 gridded(grd) <- TRUE # Create SpatialPixel object
7 fullgrid(grd) <- TRUE # Create SpatialGrid object
8 proj4string(grd) <- proj4string(as(aq.sum, "Spatial"))
9 # Define the polynomial equation
10 f.0 <- as.formula(meanpm25 ~ 1)
11
12 # Run the regression model
13 lm.0 <- lm( f.0 , data=aq.sum)
14
15 # Use the regression model output to interpolate the surface
16 dat.0th <- SpatialGridDataFrame(grd, data.frame(var1.pred = predict(lm.0, n
17
18 # Convert to raster object to take advantage of rasterVis' imaging
```

1st Order Trend Surface

- Creates a slanted surface
- $Z = a + bX + cY$
- X and Y are the coordinate pairs

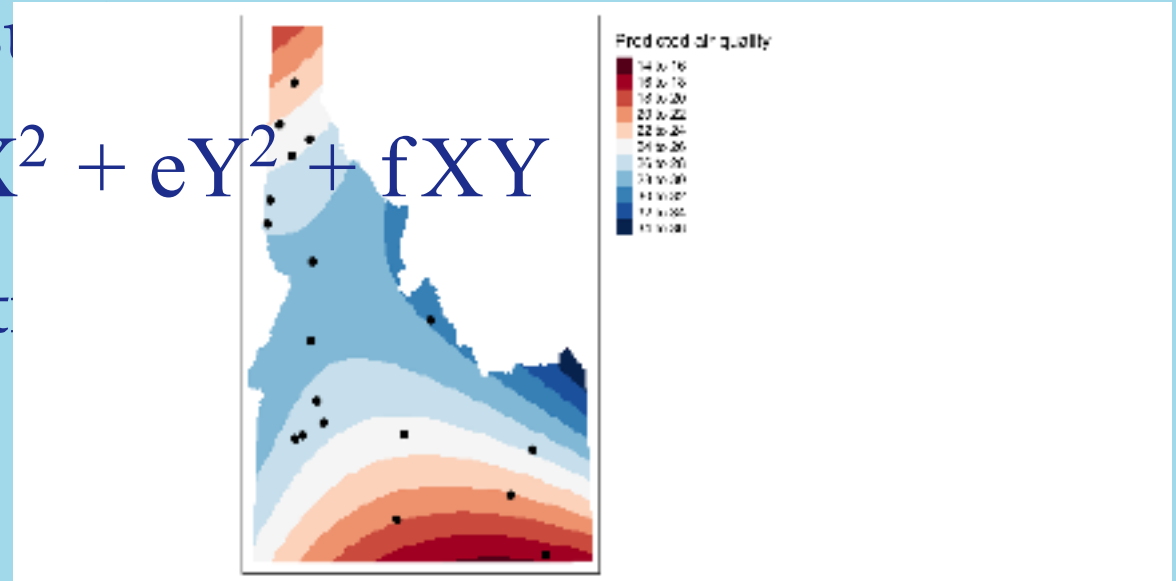


1st Order Trend Surface

```
1 # Define the polynomial equation
2 f.1 <- as.formula(meanpm25 ~ X + Y)
3
4 aq.sum$X <- st_coordinates(aq.sum)[,1]
5 aq.sum$Y <- st_coordinates(aq.sum)[,2]
6
7 # Run the regression model
8 lm.1 <- lm( f.1 , data=aq.sum)
9
10 # Use the regression model output to interpolate the surface
11 dat.1st <- SpatialGridDataFrame(grd, data.frame(var1.pred = predict(lm.1, n
12
13 # Convert to raster object to take advantage of rasterVis' imaging
14 # environment
15 r <- rast(dat.1st)
16 r.m <- mask(r, st_as_sf(id.cty))
```

2nd Order Trend Surfaces

- Produces a parabolic surface
- $Z = a + bX + cY + dX^2 + eY^2 + fXY$
- Highlights the interaction directions



2nd Order Trend Surfaces

```
1 # Define the 1st order polynomial equation
2 f.2 <- as.formula(meanpm25 ~ X + Y + I(X*X)+I(Y*Y) + I(X*Y))
3
4 # Run the regression model
5 lm.2 <- lm( f.2, data=aq.sum)
6
7 # Use the regression model output to interpolate the surface
8 dat.2nd <- SpatialGridDataFrame(grd, data.frame(var1.pred = predict(lm.2, n
9
10 r <- rast(dat.2nd)
11 r.m <- mask(r, st_as_sf(id.cty))
12
13 tm_shape(r.m) + tm_raster(n=10, palette="RdBu", title="Predicted air qualit
14   tm_legend(legend.outside=TRUE)
```

Kriging

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

$$z(\mathbf{x}) = \mu(\mathbf{x}) + \epsilon(\mathbf{x})$$

then any observed value of z is some function of the process ($\mu(\mathbf{x})$) and some error ($\epsilon(\mathbf{x})$)

- Kriging exploits autocorrelation in $\epsilon(\mathbf{x})$ 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 ϵ with the same sign
- **Negative autocorrelation** neighboring observations have ϵ with a different sign (rare in geography)

Ordinary Kriging

- Assumes that the deterministic part of the process ($\mu(\mathbf{x})$) is an unknown constant (μ)

$$z(\mathbf{x}) = \mu + \epsilon(\mathbf{x})$$

Steps for Ordinary Kriging

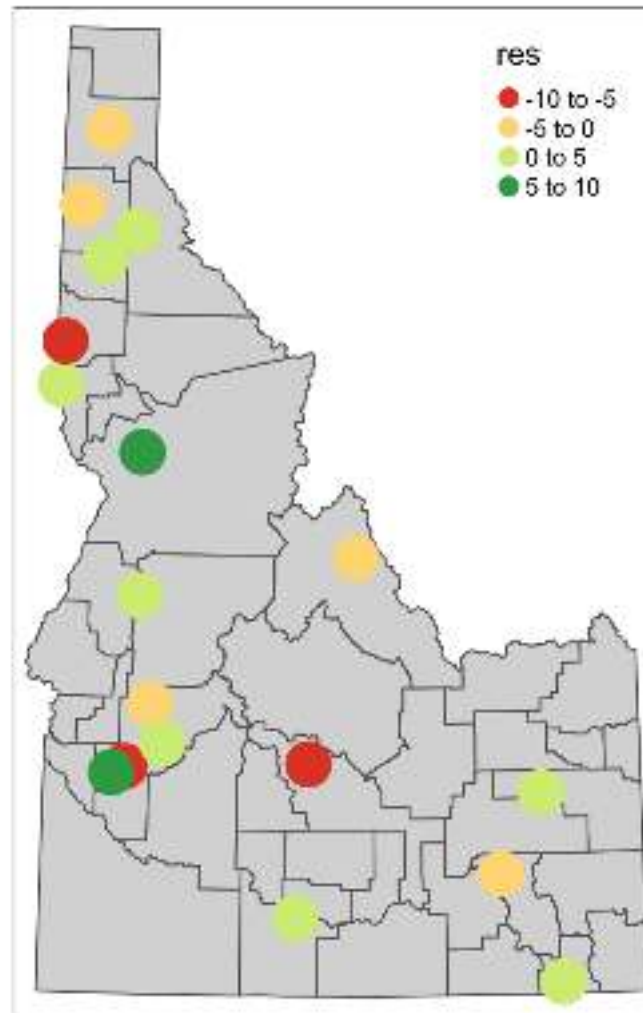
- Removing any **spatial trend** in the data (if present).
- Computing the **experimental variogram**, γ , which is a measure of spatial autocorrelation.
- Defining an **experimental variogram model** that best characterizes the spatial autocorrelation in the data.
- Interpolating the surface using the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

```
1 f.2 <- as.formula(meanpm25 ~ X + Y + I(X*X)+I(Y*Y) + I(X*Y))
2
3 # Run the regression model
4 lm.2 <- lm( f.2, data=aq.sum)
5
6 # Copy the residuals to the point object
7 aq.sum$res <- lm.2$residuals
```

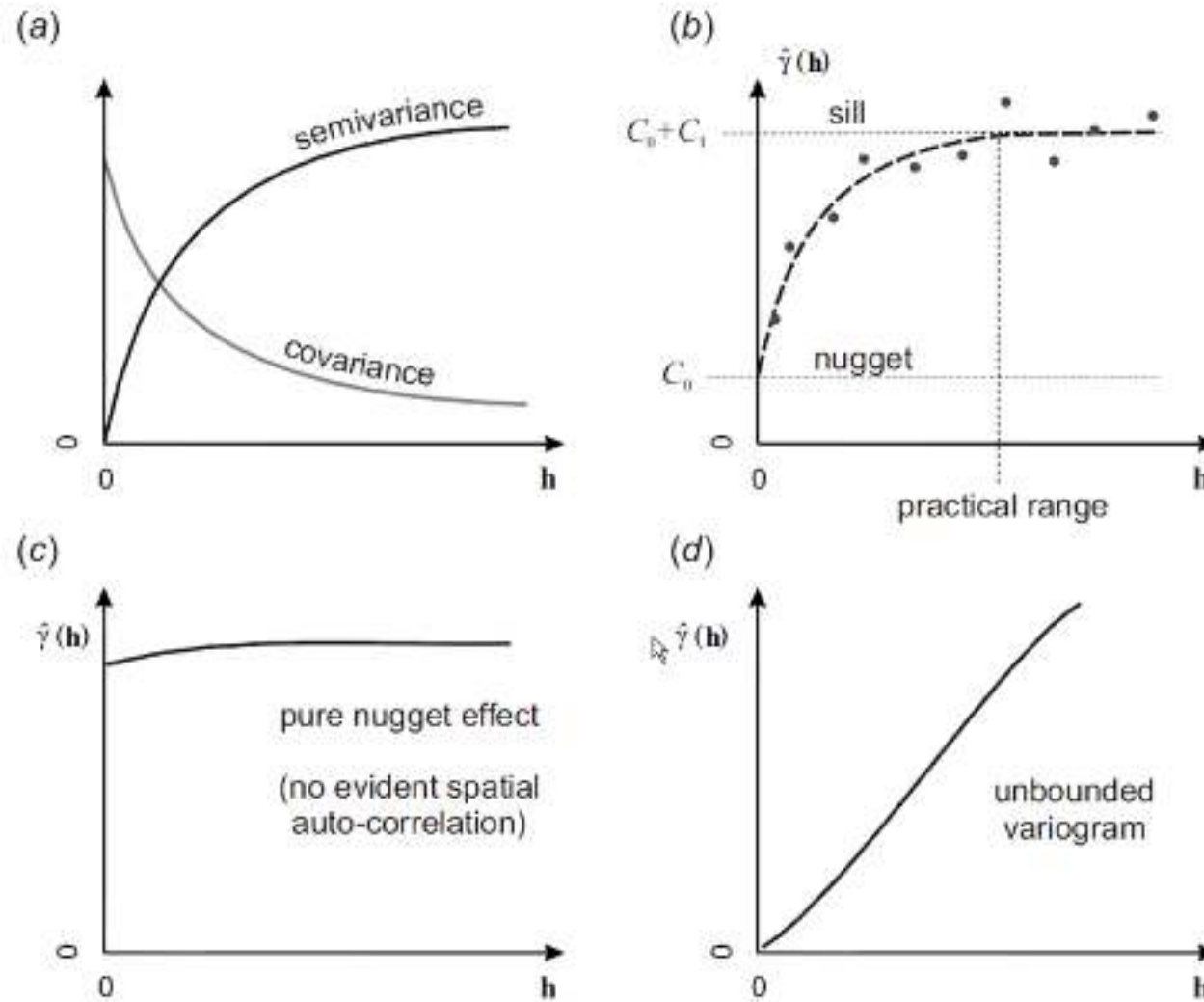

Removing the trend



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

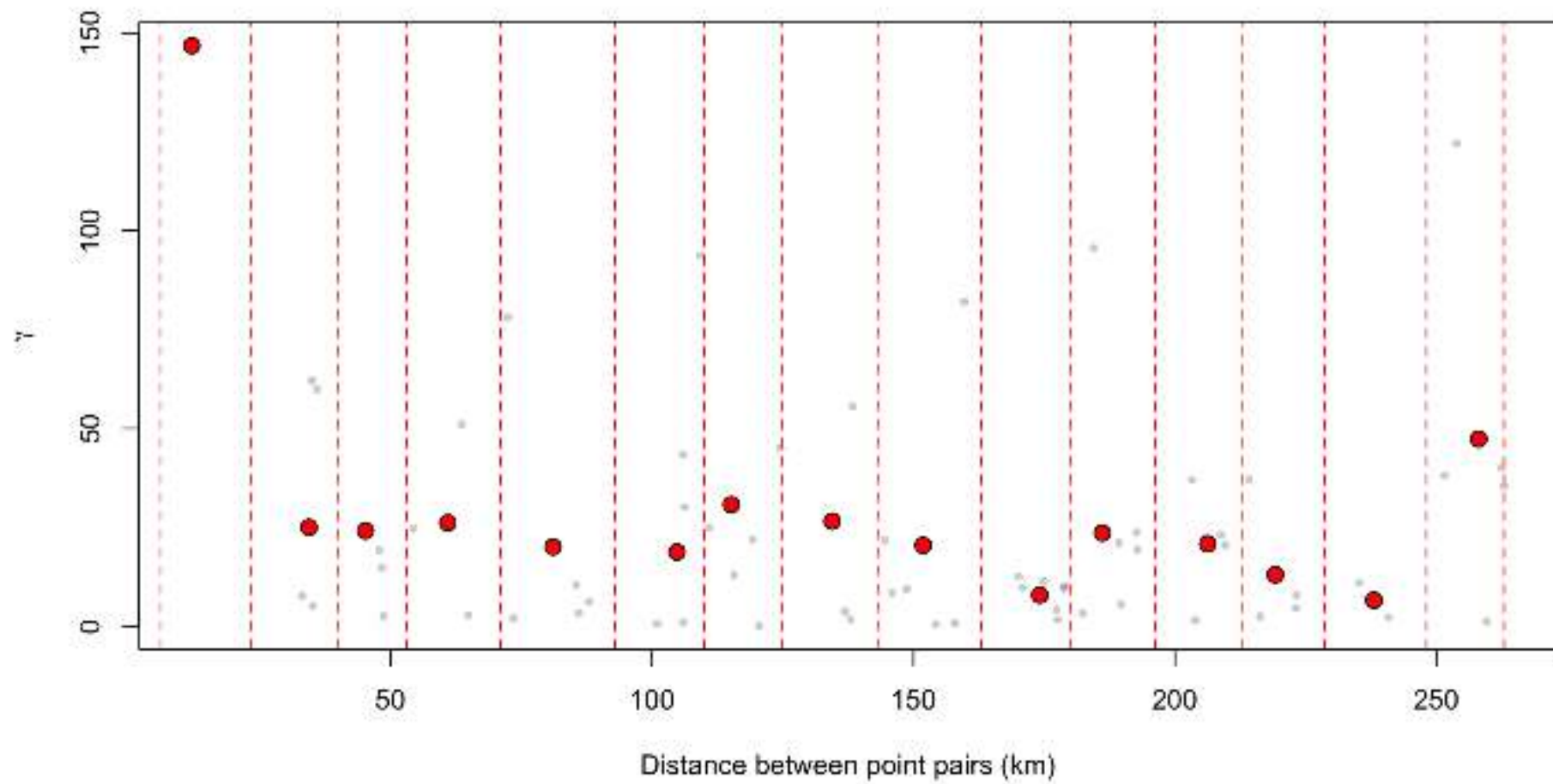
Calculate the experimental variogram

```
1 var.cld <- gstat::variogram(res ~ 1, aq.sum, cloud = TRUE)
2 var.df <- as.data.frame(var.cld)
3 index1 <- which(with(var.df, left==21 & right==2))
4
5 OP <- par( mar=c(4,6,1,1))
6 plot(var.cld$dist/1000 , var.cld$gamma, col="grey",
7       xlab = "Distance between point pairs (km)",
8       ylab = expression( frac((res[2] - res[1])^2 , 2)) )
```

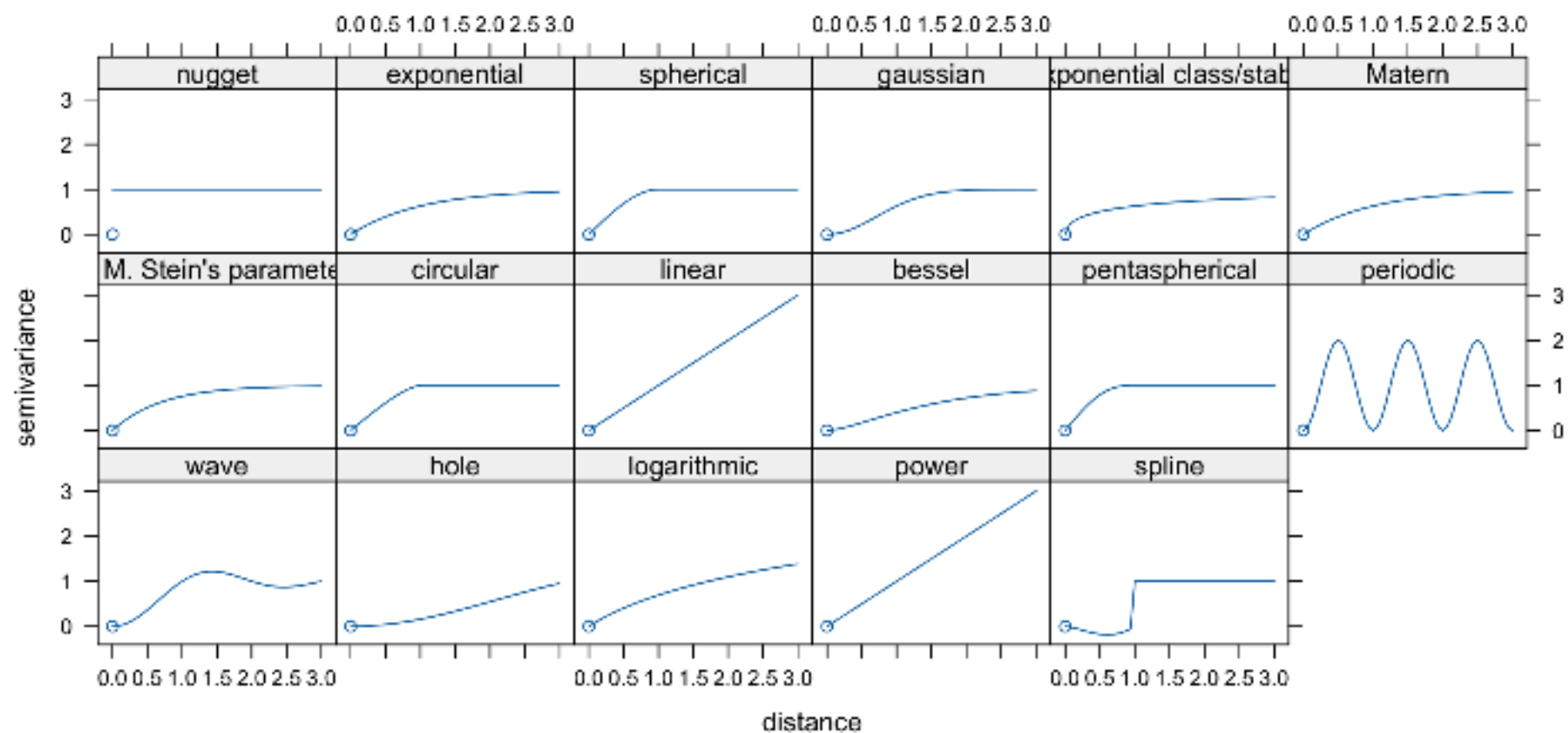
```
1 par(OP)
```

Simplifying the cloud plot

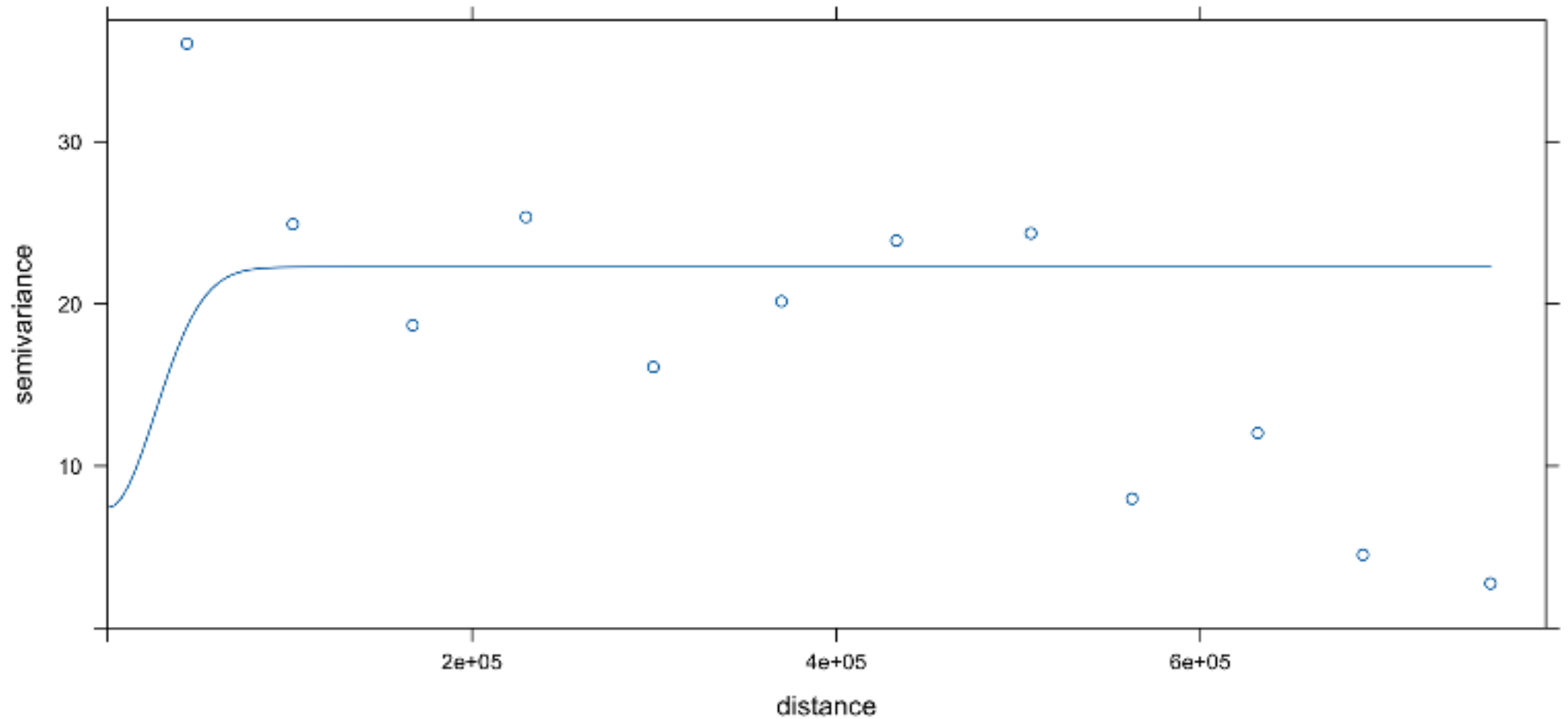
```
1 # Compute the sample experimental variogram
2 var.smpl <- gstat::variogram(f.2, aq.sum, cloud = FALSE)
3
4 bins.ct <- c(0, var.smpl$dist , max(var.cld$dist) )
5 bins <- vector()
6 for (i in 1: (length(bins.ct) - 1) ){
7   bins[i] <- mean(bins.ct[ seq(i,i+1, length.out=2)] )
8 }
9 bins[length(bins)] <- max(var.cld$dist)
10 var.bins <- findInterval(var.cld$dist, bins)
11
12 # Point data cloud with bin boundaries
13 OP <- par( mar = c(5,6,1,1))
14 plot(var.cld$gamma ~ eval(var.cld$dist/1000), col=rgb(0,0,0,0.2), pch=16, c
15       xlab = "Distance between point pairs (km)",
16       ylab = expression( gamma ) )
17 points( var.smpl$dist/1000, var.smpl$gamma, pch=21, col="black", bg="red",
18 abline(v=bins/1000, col="red", lty=2)
```



```
1 par(OP)
```



Looking at the sample Variogram

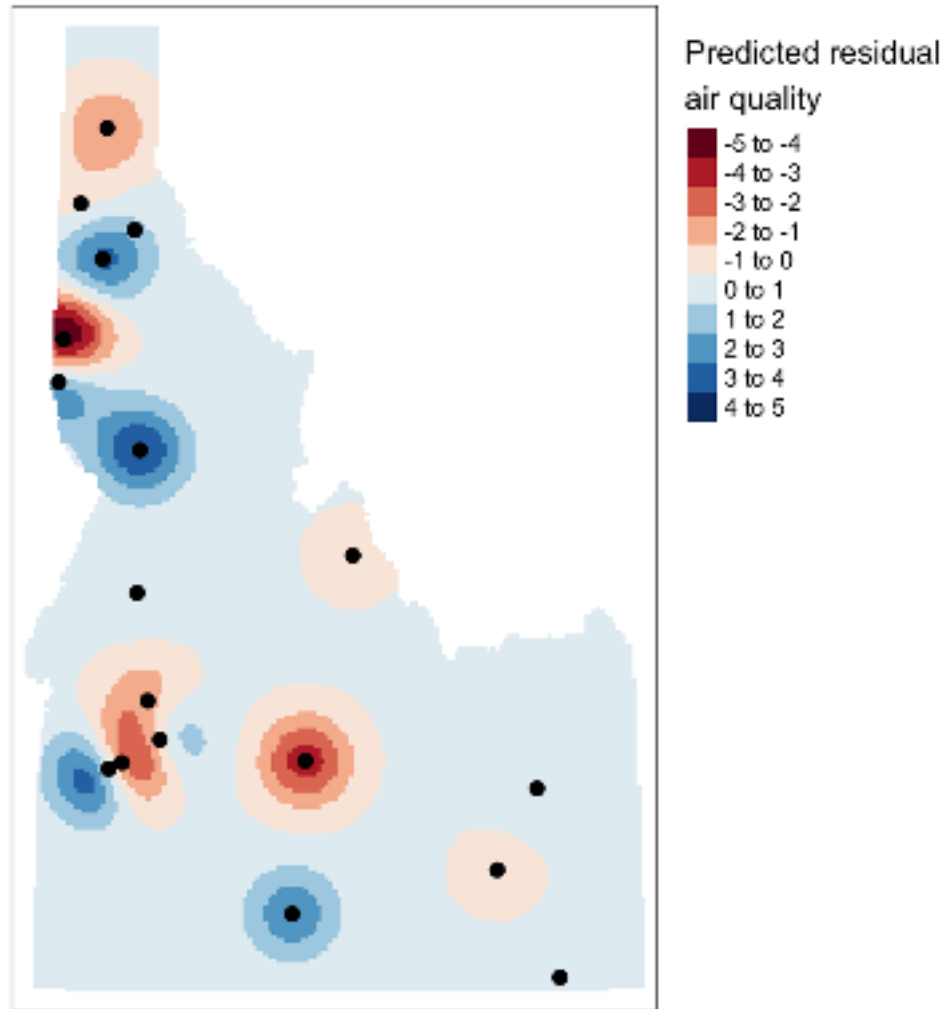


Estimating the sample variogram

```
1 var.smpl <- gstat::variogram(f.2, aq.sum, cloud = FALSE, cutoff = 1000000)
2
3
4 # Compute the variogram model by passing the nugget, sill and range values
5 # to fit.variogram() via the vgm() function.
6 dat.fit <- gstat::fit.variogram(var.smpl, gstat::vgm(nugget = 12, range= 6
```

Ordinary Kriging

[using ordinary kriging]

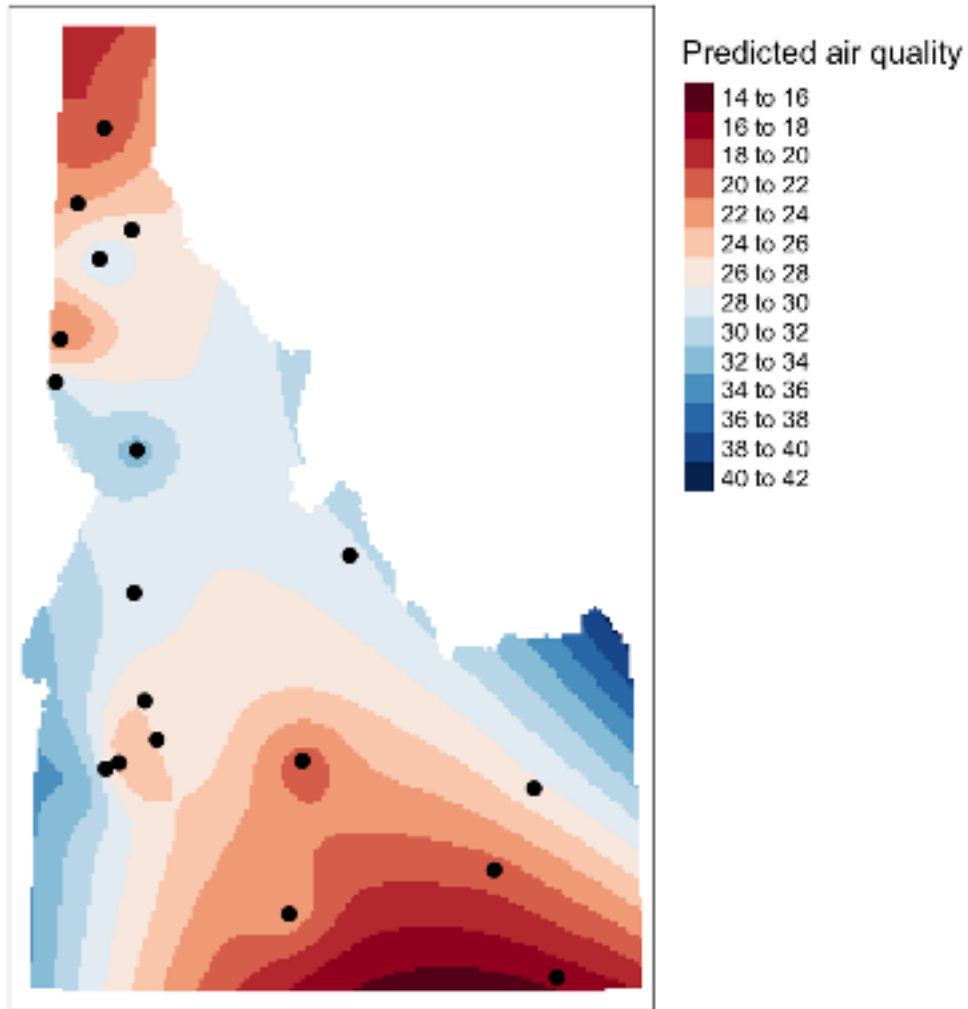


Ordinary Kriging

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

Combining with the trend data

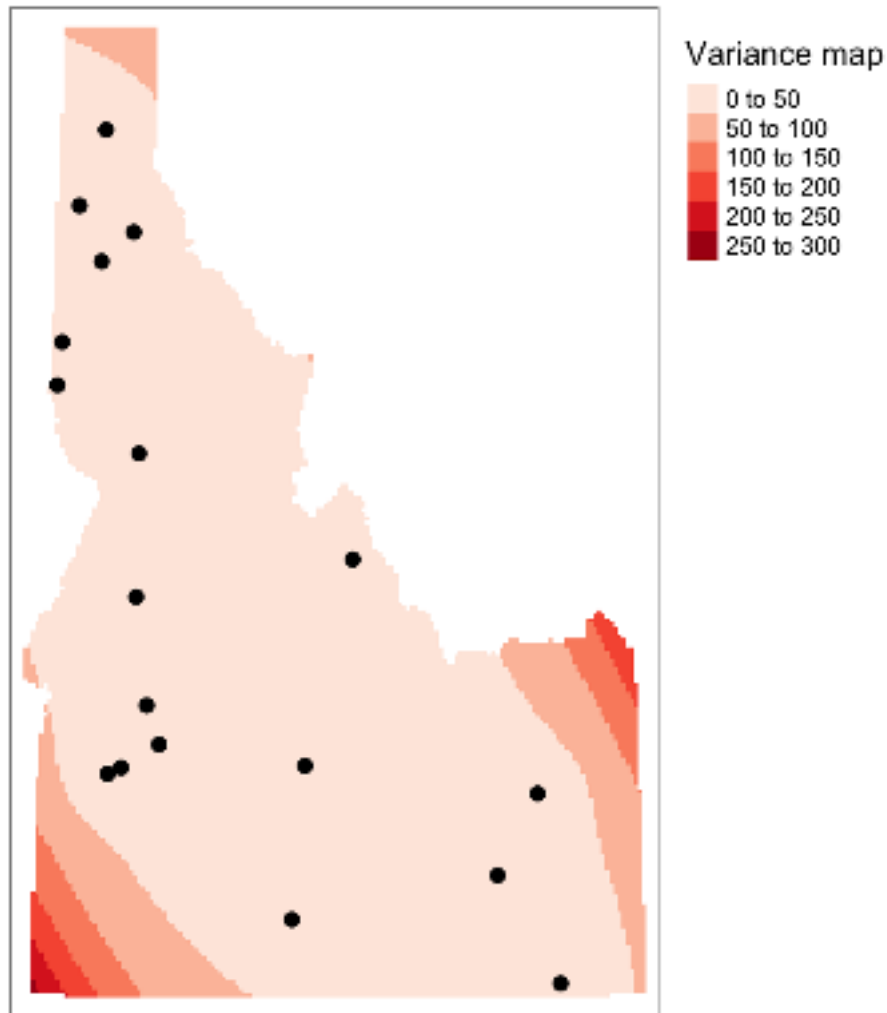
[using universal kriging]



Combining with the trend data

```
1 dat.krg <- gstat::krige( f.2, as(aq.sum, "Spatial"), grd, dat.fit)
2
3 r <- rast(dat.krg)$var1.pred
4 r.m <- mask(r, st_as_sf(id.cty))
5
6 # Plot the raster and the sampled points
7 tm_shape(r.m) + tm_raster(n=10, palette="RdBu", title="Predicted air qualit
8   tm_legend(legend.outside=TRUE)
```

Visualizing Uncertainty



Universal Kriging

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

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


Universal Kriging

Universal Kriging

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

Universal Kriging

Co-Kriging

- relies on autocorrelation in $\epsilon_1(\mathbf{x})$ for z_1 AND cross correlation with other variables ($z_2 \dots z_j$)
- Extending the ordinary kriging model gives:

$$z_1(\mathbf{x}) = \mu_1 + \epsilon_1(\mathbf{x})$$

$$z_2(\mathbf{x}) = \mu_2 + \epsilon_2(\mathbf{x})$$

* Note that there is autocorrelation within both z_1 and z_2 (because of the ϵ) and cross-correlation (because of the location, \mathbf{x})

Co-Kriging

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

```
1 gCoK <- gstat(NULL, 'log.zinc', log(zinc)~1, meuse, locations=~x+y)
2 gCoK <- gstat(gCoK, 'elev', elev~1, meuse, locations=~x+y)
3 gCoK <- gstat(gCoK, 'cadmium', cadmium~1, meuse, locations=~x+y)
4 coV <- variogram(gCoK)
5 coV.fit <- fit.lmc(coV, gCoK, vgm(model='Sph', range=1000))
6
7 coK <- interpolate(r, coV.fit, debug.level=0)
```

Co-Kriging

Co-Kriging

A Note about Semivariograms

