

# Interpolation and Autocorrelation

HES 505 Fall 2022: Session 17

Matt Williamson

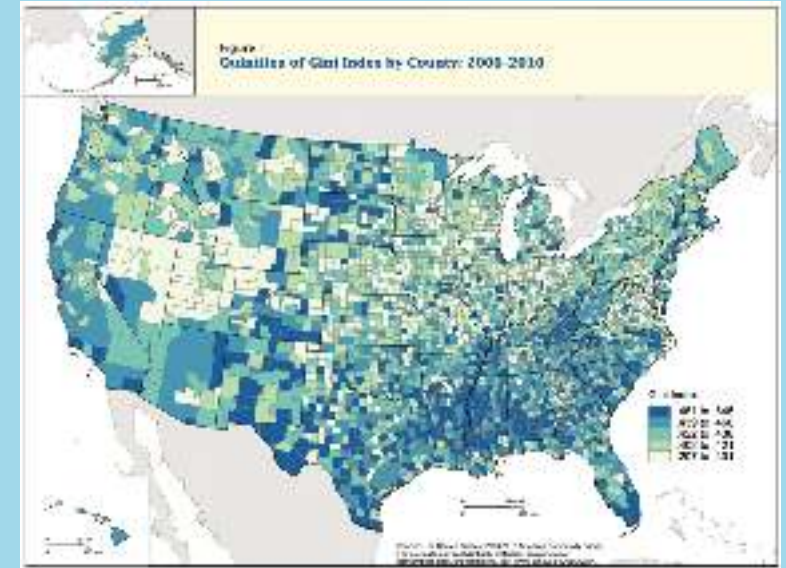
# Objectives

By the end of today you should be able to:

- Distinguish deterministic and stochastic processes
- Define autocorrelation and describe its estimation
- Articulate the benefits and drawbacks of autocorrelation
- Leverage point patterns and autocorrelation to interpolate missing data

# Description vs. process?

- Visualization and the detection of patterns
- The challenge of geographic data
- Implications for analysis



Inequality in the United States: Quintiles of Gini Index by County: 2006–2010. The greater the Gini index, the more unequal a county's income distribution is.

# Patterns as realizations of spatial processes

- A **spatial process** is a description of how a spatial pattern might be *generated*
- **Generative models**
- An observed pattern as a *possible realization* of an hypothesized process

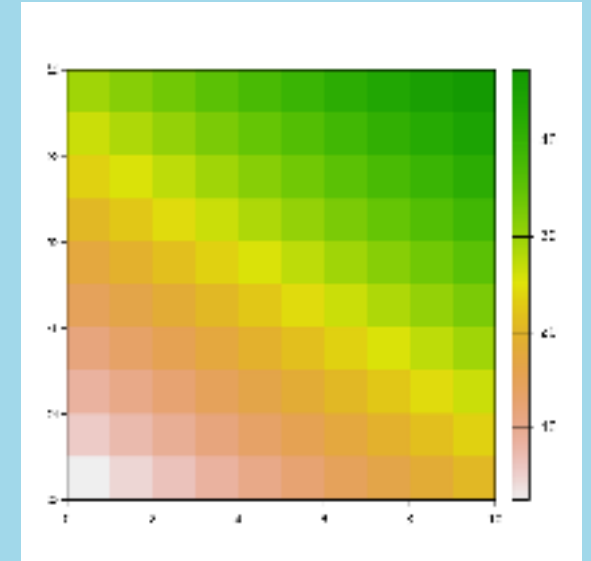
# Deterministic vs. stochastic processes

- Deterministic processes: always produces the same outcome

$$z = 2x + 3y$$

- Results in a spatially continuous field

```
1 x <- rast(nrows = 10, ncols=10, xmin = 0, xmax=10,  
2 values(x) <- 1  
3 z <- x  
4 values(z) <- 2 * crds(x)[,1] + 3*crds(x)[,2]
```



# Deterministic vs. stochastic processes

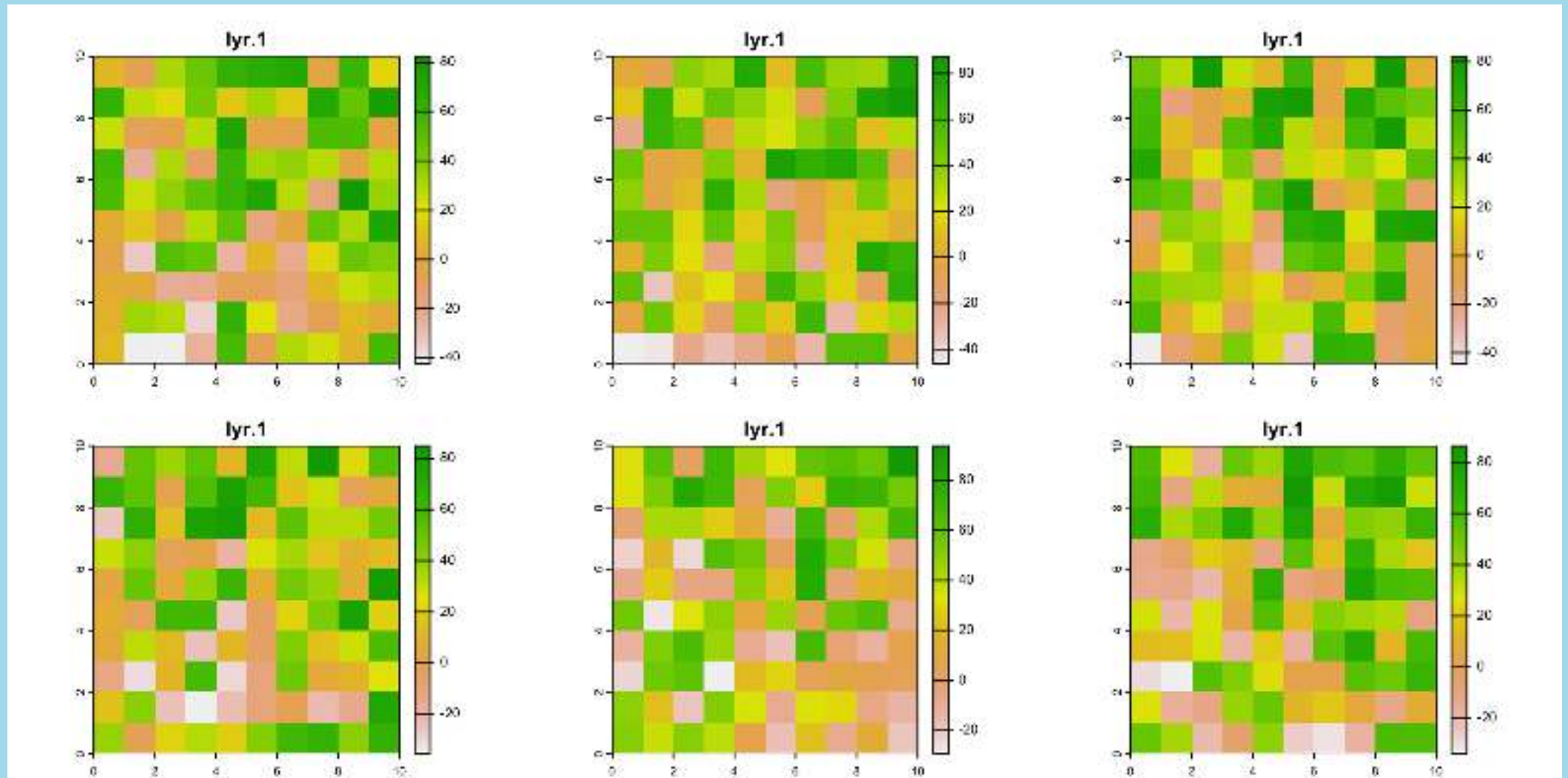
- Stochastic processes: variation makes each realization difficult to predict

$$z = 2x + 3y + d$$

- The *process* is random, not the result (!!)
- Measurement error makes deterministic processes appear stochastic

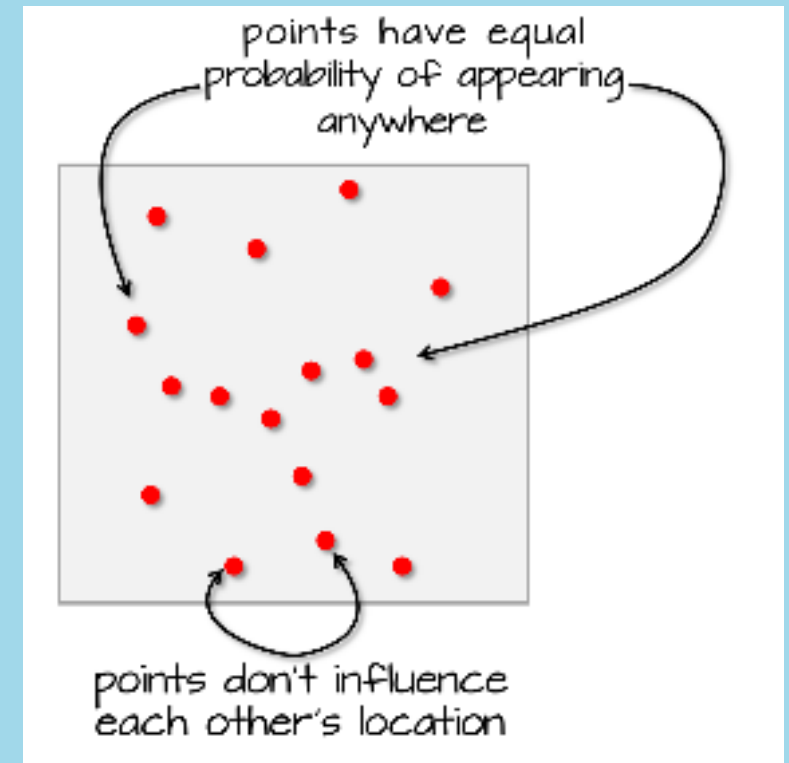
```
1 x <- rast(nrows = 10, ncols=10, xmin = 0,
2 values(x) <- 1
3 fun <- function(z){
4 a <- z
5 d <- runif(ncell(z), -50, 50)
6 values(a) <- 2 * crds(x)[,1] + 3*crds(x)[,
7 return(a)
8 }
9
10 b <- replicate(n=6, fun(z=x), simplify=FAI
11 d <- do.call(c, b)
```

# Deterministic vs. stochastic processes



# Expected values and hypothesis testing

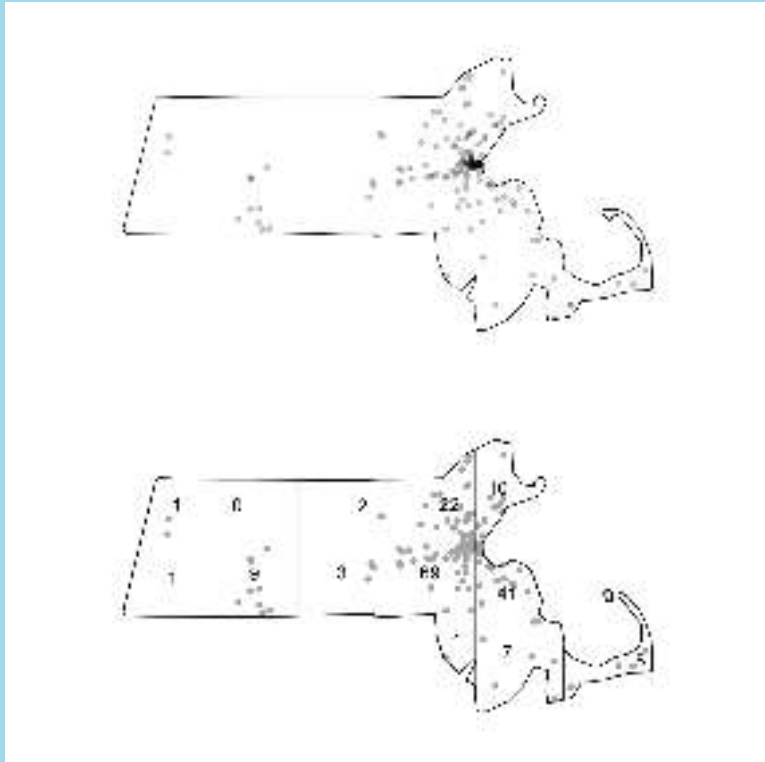
- Considering each outcome as the realization of a process allows us to generate expected values
- The simplest spatial process is Completely Spatial Random (CSR) process
- **First Order** effects: any event has an equal probability of occurring in a location
- **Second Order** effects: the location of one event is independent of the other events



From Manuel Gimond



# Generating expectations for CSR



- We can use quadrat counts to estimate the expected number of events in a given area
- The probability of each possible count is given by:

$$P(n, k) = \binom{n}{x} p^k (1 - p)^{n-k}$$

- Given total coverage of quadrats, then  $p = \frac{\frac{a}{x}}{a}$  and

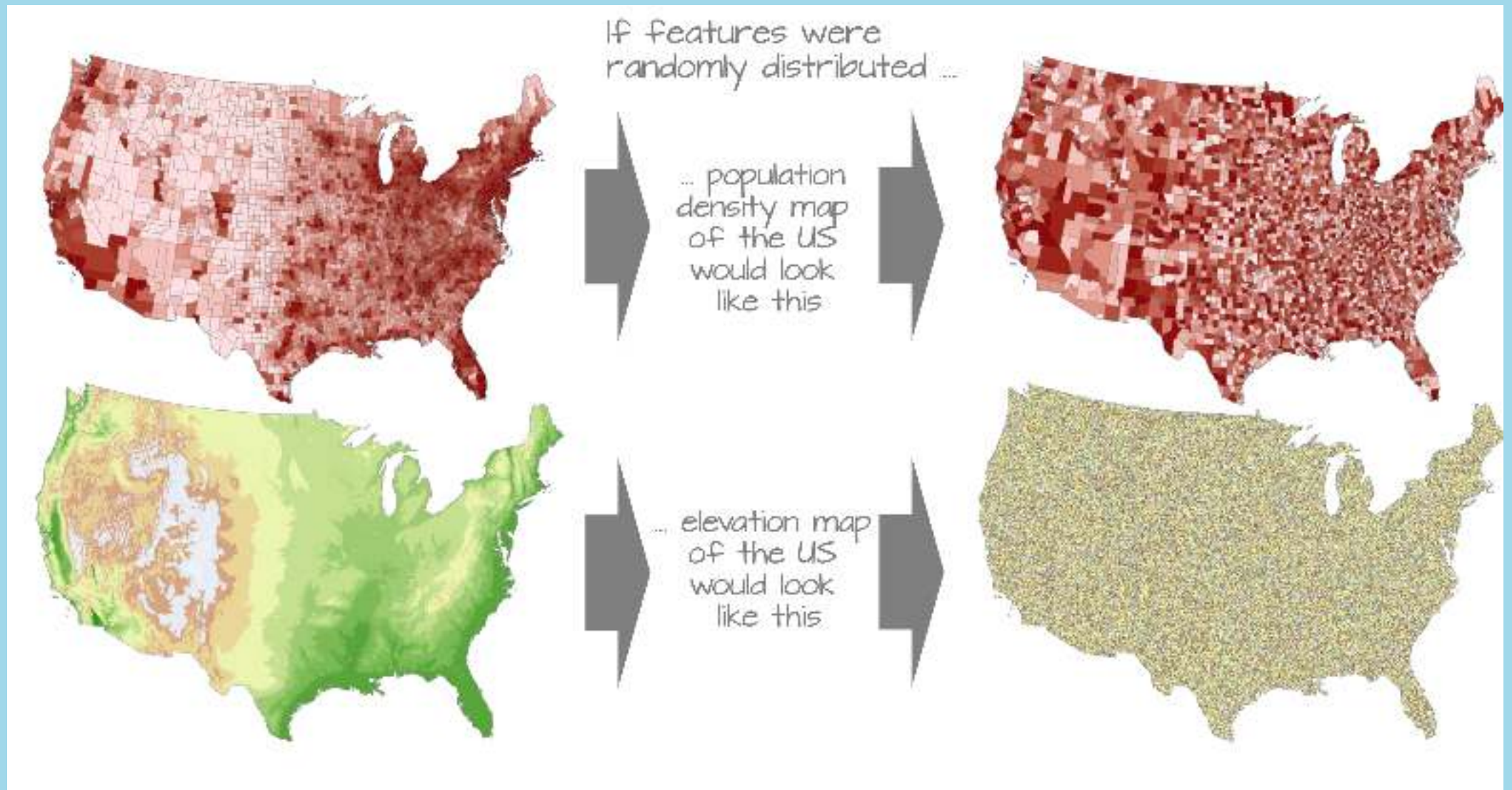
$$P(k, n, x) = \binom{n}{k} \left( \frac{1}{x} \right)^k \left( \frac{x-1}{x} \right)^{n-k}$$

# Tobler's Law

‘everything is usually related to all else but those which are near to each other are more related when compared to those that are further away’.

Waldo Tobler

# Spatial autocorrelation

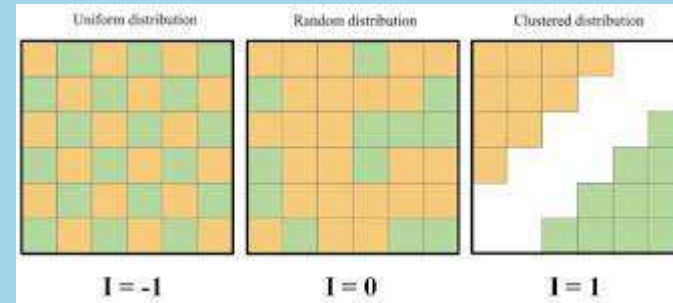


From Manuel Gimond

# (One) Measure of autocorrelation

- Moran's I

$$I(d) = \frac{\sum_i \sum_{j \neq i} w_{ij} (x_i - \bar{x})(x_j - \bar{x})}{S^2 \sum_i \sum_{j \neq i} w_{ij}}$$



# Moran's I: An example

- Use **spdep** package
- Estimate neighbors
- Generate weighted average

```
1 set.seed(2354)
2 # Load the shapefile
3 s <- readRDS(url("https://github.com/mgimond/Data/raw/gh-pages"))
4
5 # Define the neighbors (use queen case)
6 nb <- poly2nb(s, queen=TRUE)
7
8 # Compute the neighboring average homicide rates
9 lw <- nb2listw(nb, style="W", zero.policy=TRUE)
10 #estimate Moran's I
11 moran.test(s$HR80,lw, alternative="greater")
```

Moran I test under randomisation

data: s\$HR80

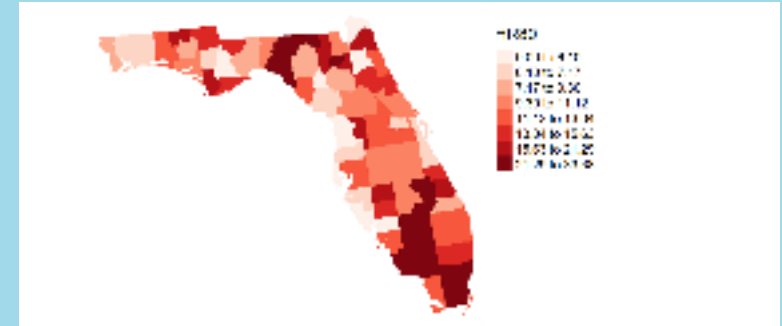
weights: lw

Moran I statistic standard deviate = 1.8891, p-value = 0.02944

alternative hypothesis: greater

sample estimates:

Moran I statistic	Expectation	Variance
0.136277593	-0.015151515	0.006425761



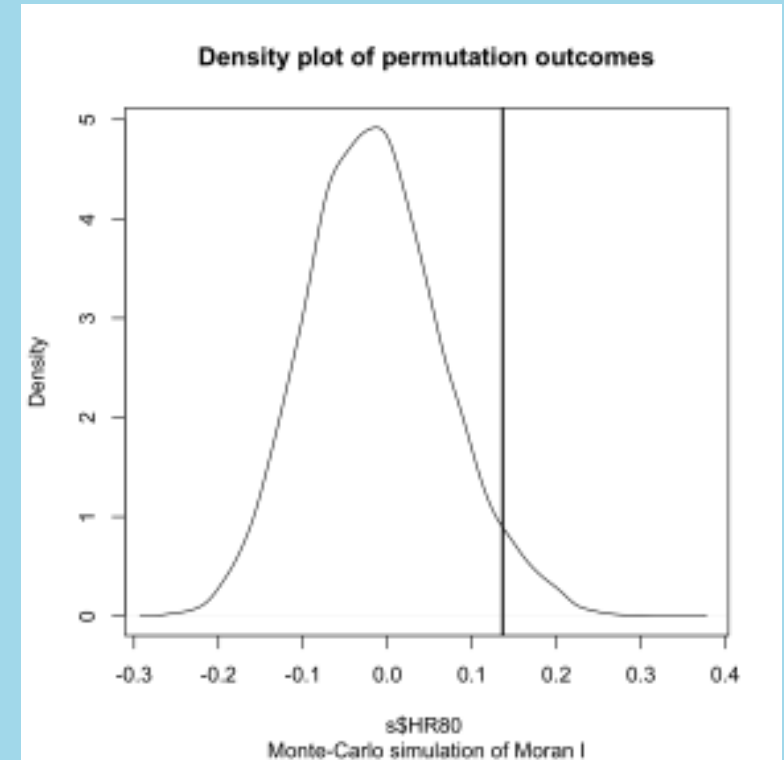
# Moran's I: An example

```
1 M1 <- moran.mc(s$HR80, lw, nsim=9999, alte  
2  
3  
4  
5 # Display the resulting statistics  
6 M1
```

Monte-Carlo simulation of Moran I

```
data: s$HR80  
weights: lw  
number of simulations + 1: 10000
```

```
statistic = 0.13628, observed rank = 9575, p-  
value = 0.0425  
alternative hypothesis: greater
```



# The challenge of areal data

- Spatial autocorrelation threatens *second order* randomness
- Areal data means an infinite number of potential distances
- Neighbor matrices,  $W$ , allow different characterizations

# Interpolation



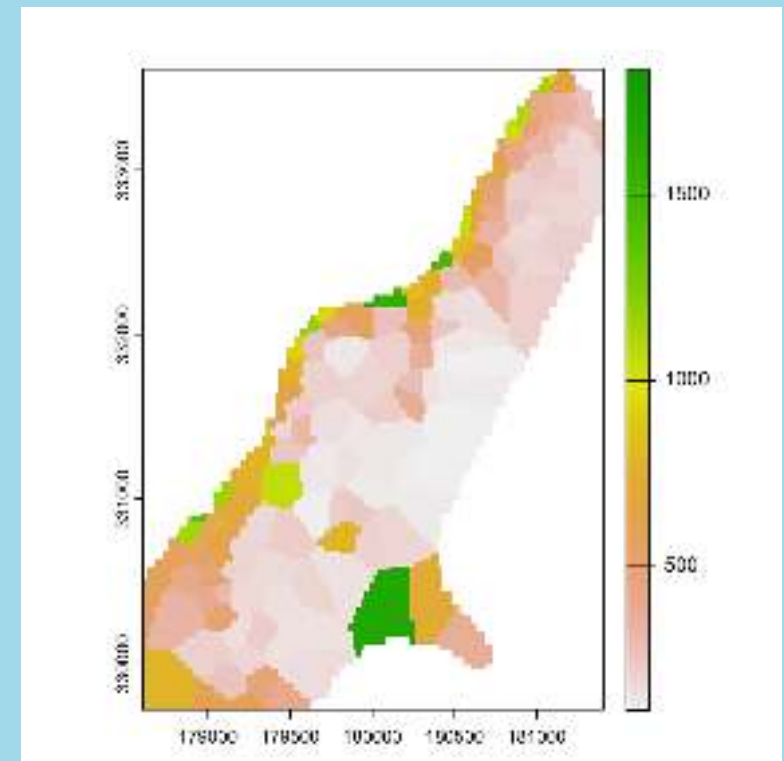
# Interpolation

- Goal: estimate the value of  $z$  at new points in  $\mathbf{x}_i$
- Most useful for continuous values
- Nearest-neighbor, Inverse Distance Weighting, Kriging

# Nearest neighbor

- find  $i$  such that  $|\mathbf{x}_i - \mathbf{x}|$  is minimized
- The estimate of  $z$  is  $z_i$

```
1 data(meuse)
2 r <- rast(system.file("ex/meuse.tif", package="terra"))
3 sfmeuse <- st_as_sf(meuse, coords = c("x", "y"), crs=crs(r))
4 nodes <- st_make_grid(sfmeuse,
5                       cellsize = 25,
6                       what = "centers")
7
8 dist <- distance(vect(nodes), vect(sfmeuse))
9 nearest <- apply(dist, 1, function(x) which(x == min(x)))
10 zinc_nn <- sfmeuse$zinc[nearest]
11 preds <- st_as_sf(nodes)
12 preds$zn <- zinc_nn
13 preds <- as(preds, "Spatial")
14 gridded(preds) <- TRUE
15 preds.rast <- rast(preds)
16 r.resamp <- resample(r, preds.rast)
17 preds.rast <- mask(preds.rast, r.resamp)
```



# Inverse-Distance Weighting

- Weight closer observations more heavily

$$\hat{z}(\mathbf{x}) = \frac{\sum_{i=1} w_i z_i}{\sum_{i=1} w_i}$$

where

$$w_i = |\mathbf{x} - \mathbf{x}_i|^{-\alpha}$$

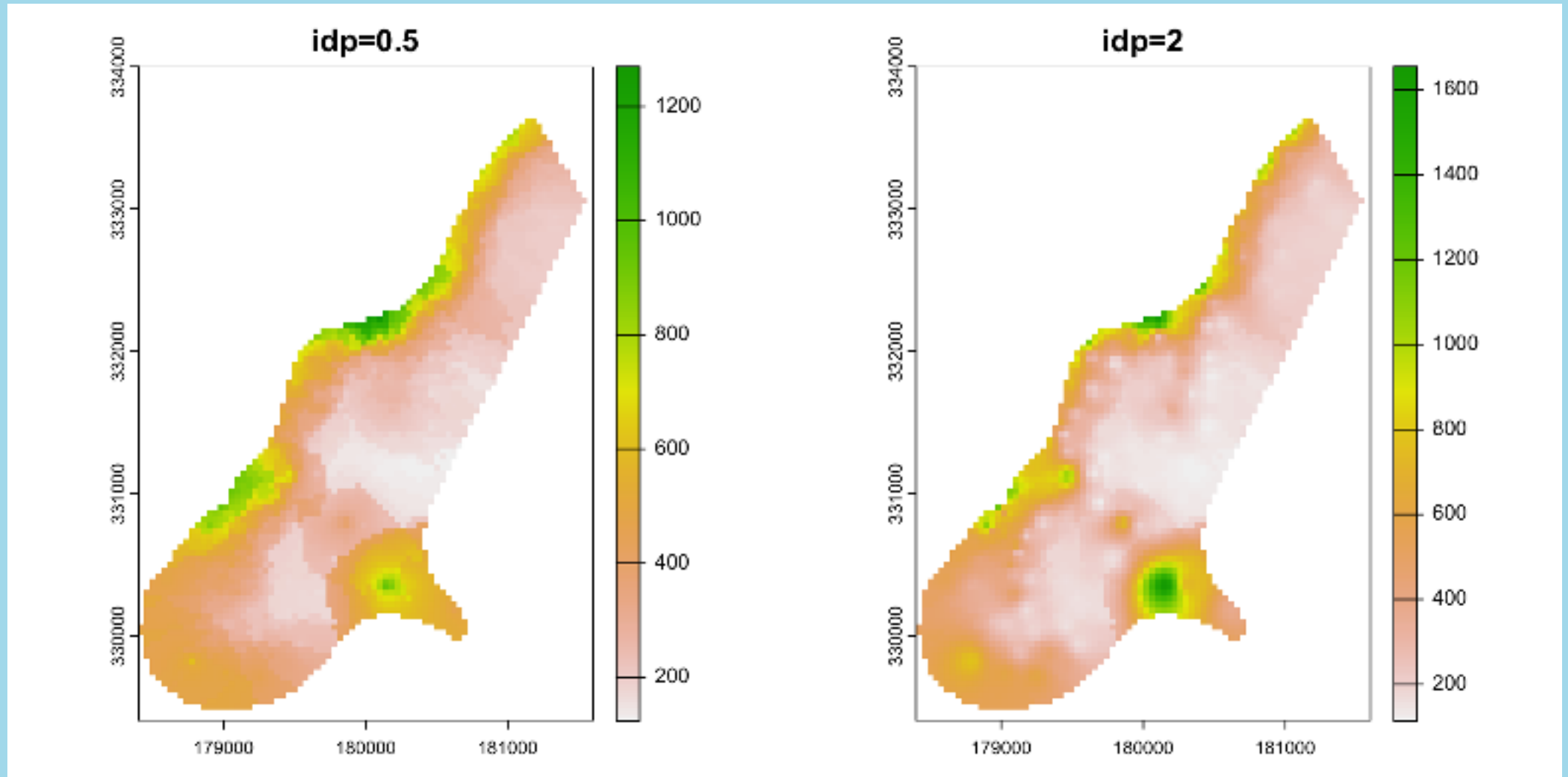
and  $\alpha > 0$  ( $\alpha = 1$  is inverse;  $\alpha = 2$  is inverse square)

# Inverse-Distance Weighting

- `terra::interpolate` provides flexible interpolation methods
- Use the `gstat` package to develop the formula

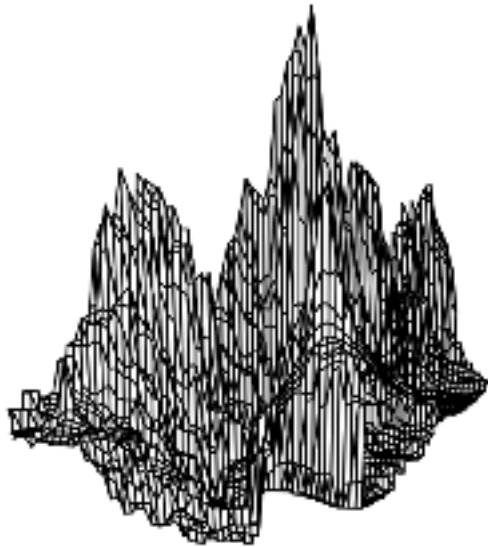
```
1 mgsf05 <- gstat(id = "zinc", formula = zinc~1, data=sfmeuse, nmax=7, set=1
2 mgsf2 <- gstat(id = "zinc", formula = zinc~1, data=sfmeuse, nmax=7, set=li
3 interpolate_gstat <- function(model, x, crs, ...) {
4     v <- st_as_sf(x, coords=c("x", "y"), crs=crs)
5     p <- predict(model, v, ...)
6     as.data.frame(p)[,1:2]
7 }
8 zsf05 <- interpolate(r, mgsf05, debug.level=0, fun=interpolate_gstat, crs=c
9 zsf2 <- interpolate(r, mgsf2, debug.level=0, fun=interpolate_gstat, crs=crs
```

# Inverse-Distance Weighting

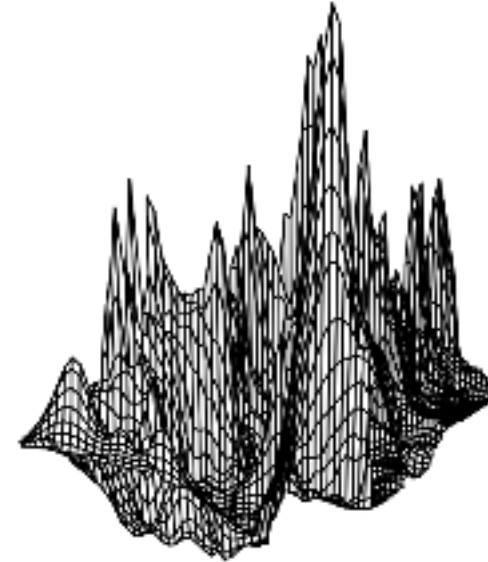


# Inverse-Distance Weighting

idp=0.5



idp=2



# 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  $\epsilon$  with the same sign
- **Negative autocorrelation** neighboring observations have  $\epsilon$  with a different sign (rare in geography)



# Ordinary Kriging

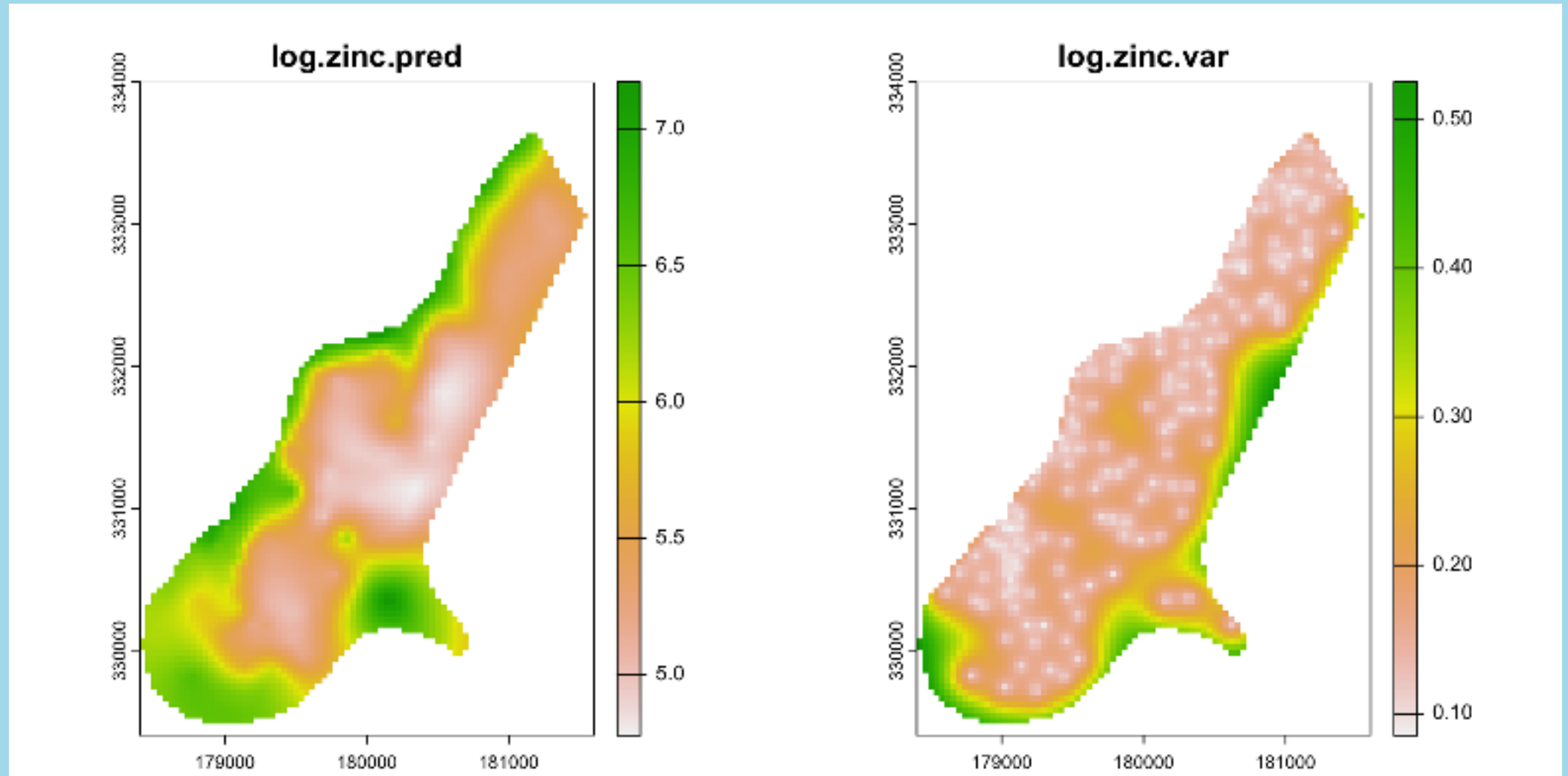
- Assumes that the deterministic part of the process ( $\mu(\mathbf{x})$ ) is an unknown constant ( $\mu$ )

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

\* Specified in call to **variogram** and **gstat** as **y~1** (or some other constant)

```
1 v <- variogram(log(zinc)~1, ~x+y, data=meuse)
2 mv <- fit.variogram(v, vgm(1, "Sph", 300, 1))
3 gOK <- gstat(NULL, "log.zinc", log(zinc)~1, meuse, locations=~x+y, model=mv)
4 OK <- interpolate(r, gOK, debug.level=0)
```

# Ordinary Kriging

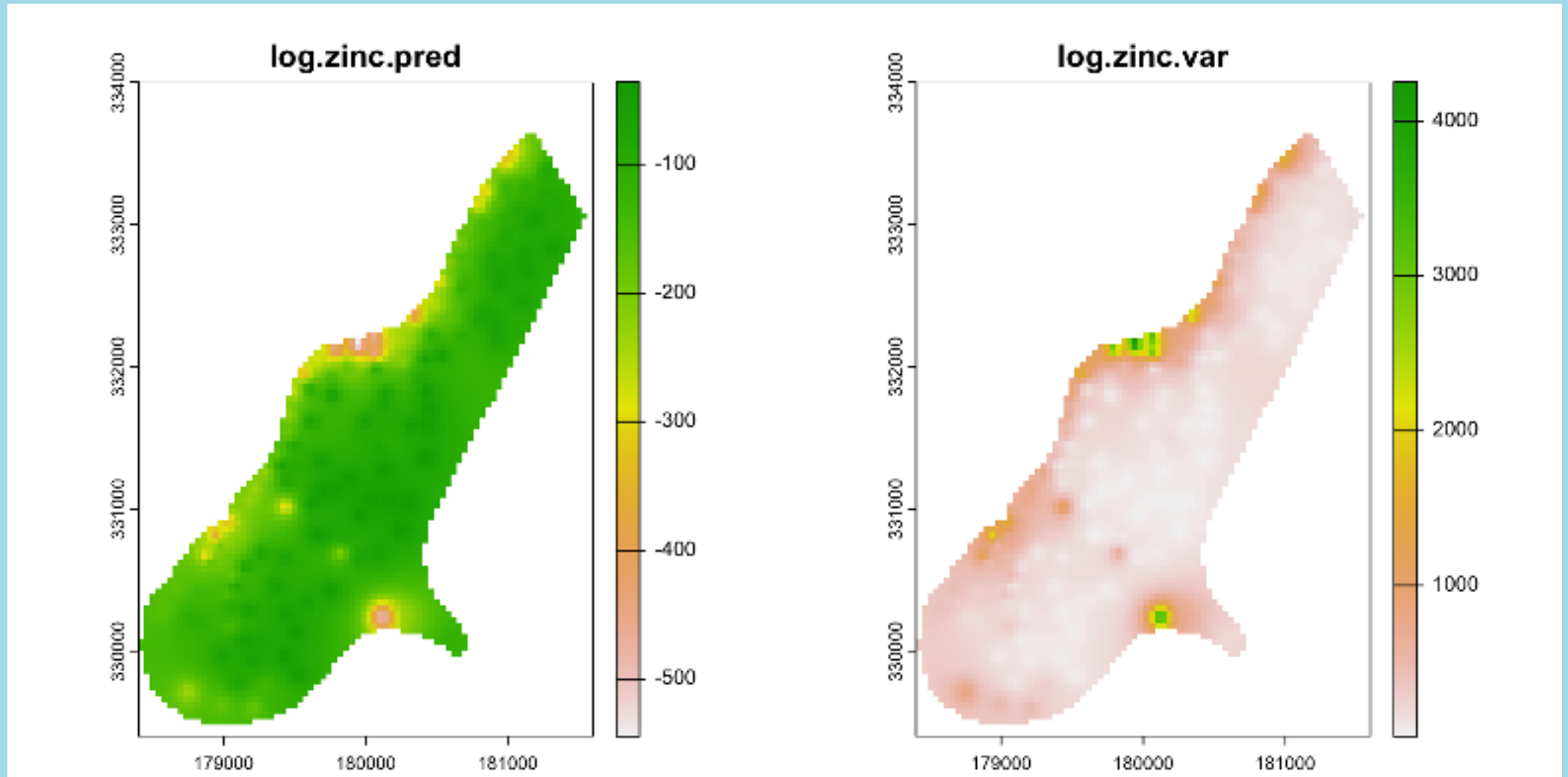


# Universal Kriging

- 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)
```

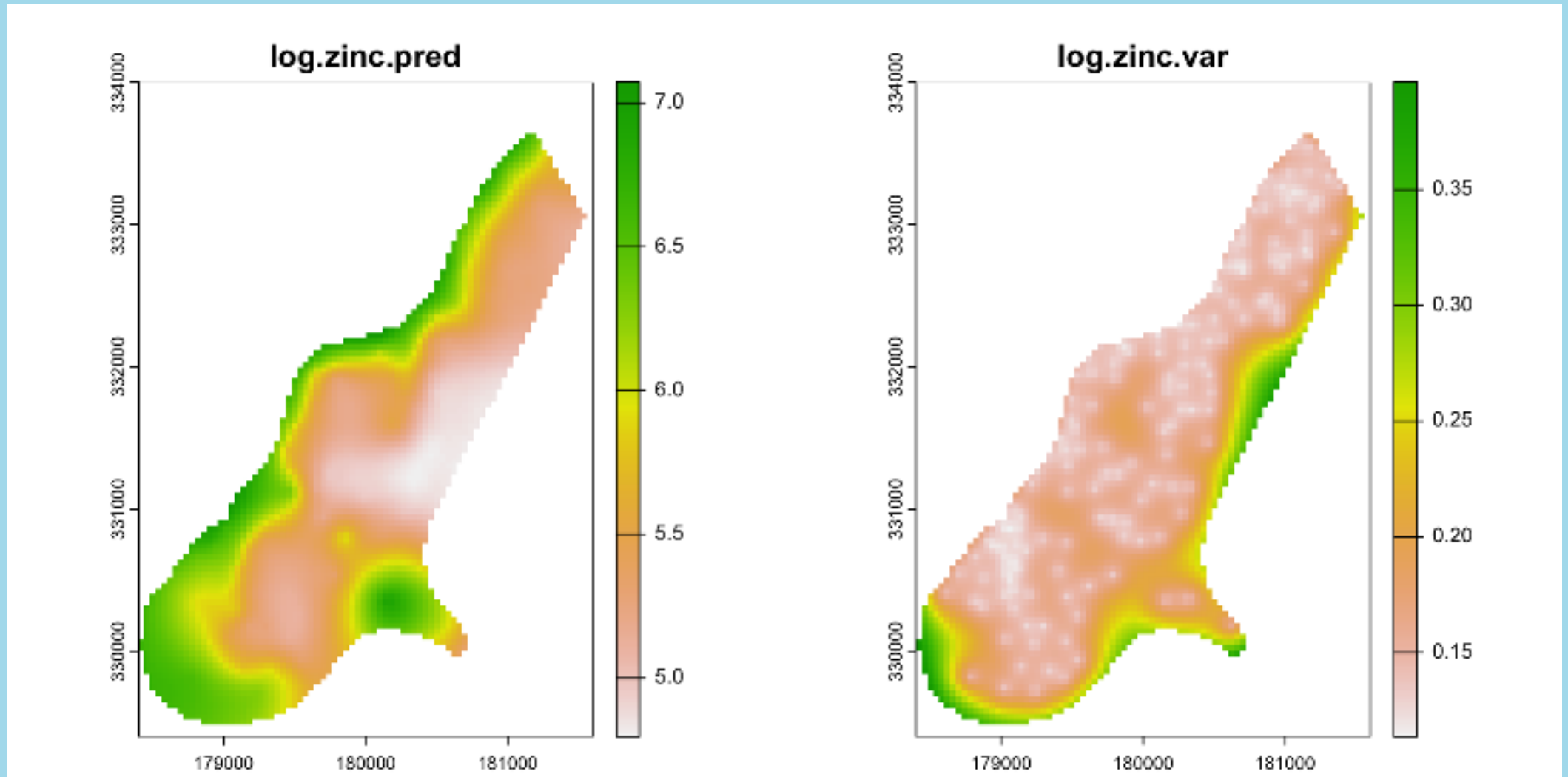
# Universal Kriging



# 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  $\epsilon$ ) and cross-correlation (because of the location,  $\mathbf{x}$ )

# Co-Kriging

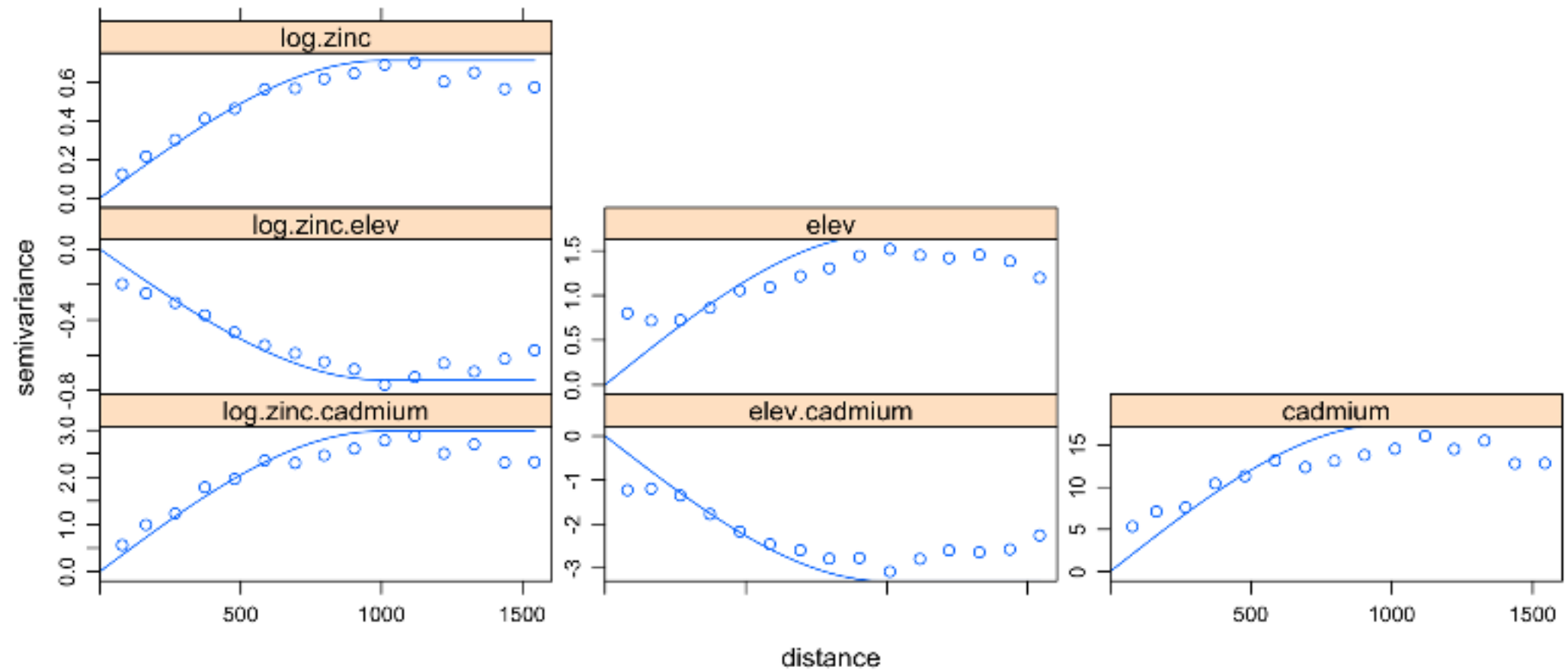
- Process is just a linked series of **gstat** calls

```
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)
```

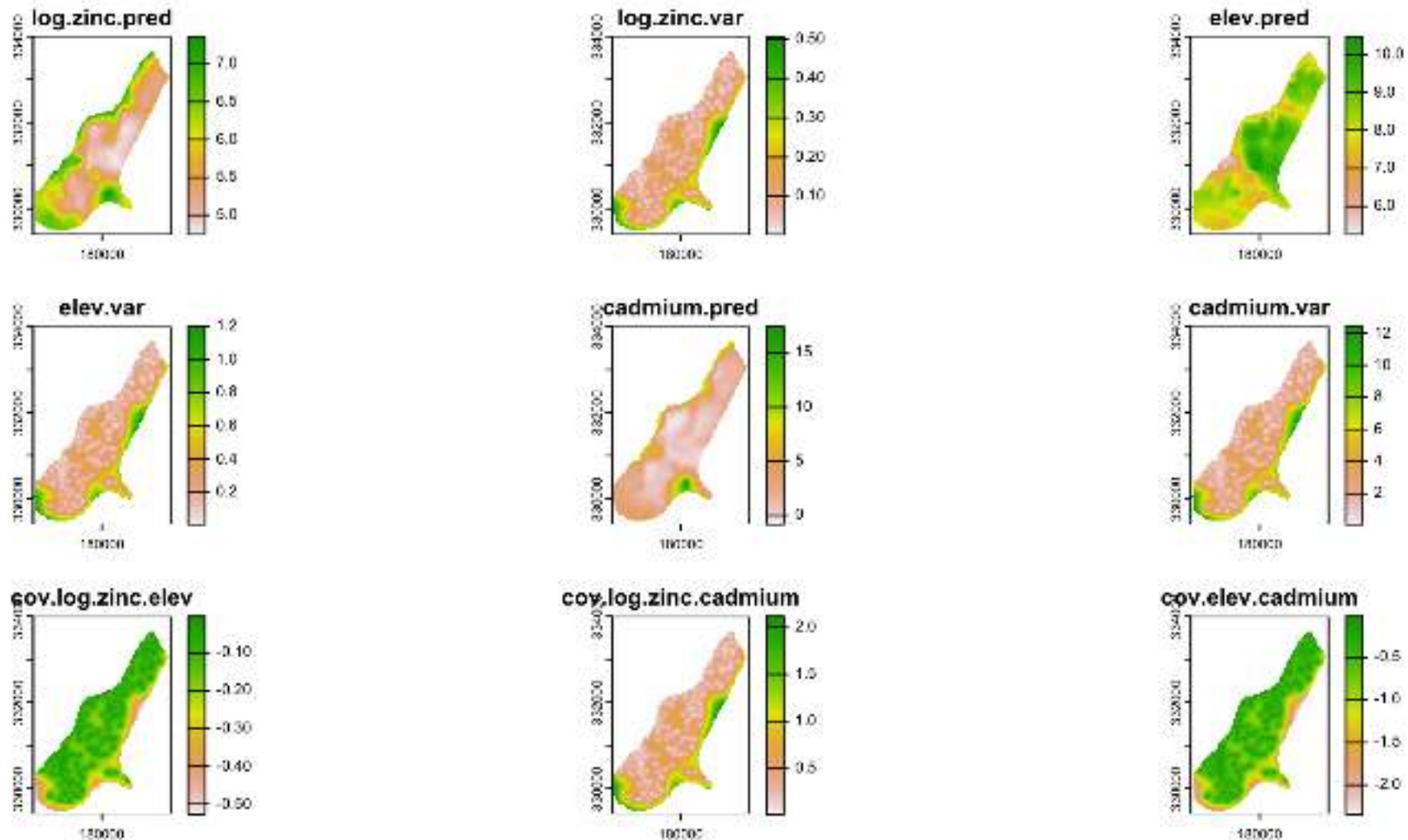


# Co-Kriging

Fitted Co-variogram



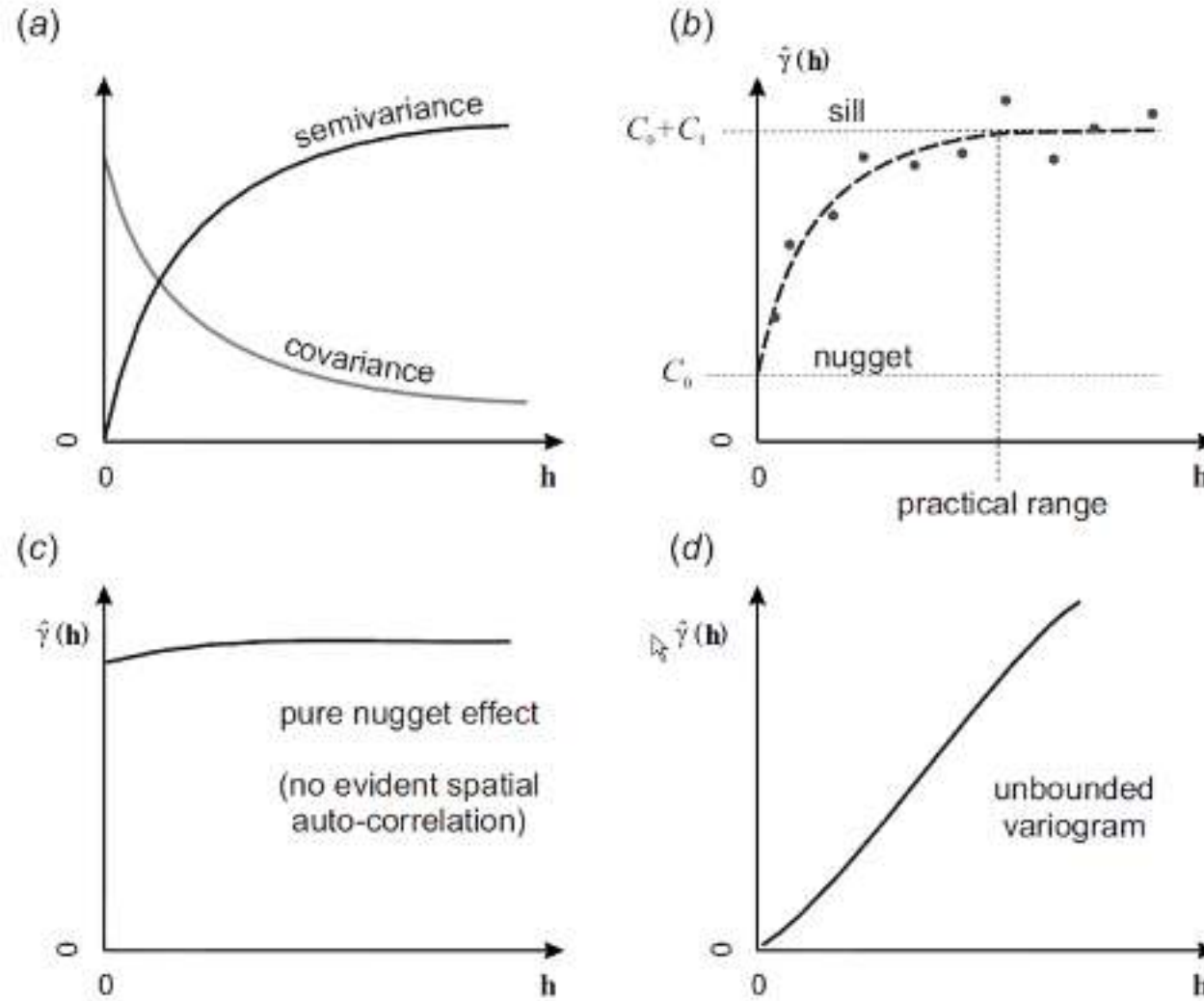
# Co-Kriging



# A Note about Semivariograms

- **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

