

# Interpolation

HES 505 Fall 2024: Session 18

Carolyn Koehn

# Assignment Revision

2

# 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

But first...

# 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 produce the same outcome

$$z = 2x + 3y$$

- Results in a spatially continuous field

# Deterministic vs. stochastic processes

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

# Deterministic vs. stochastic processes

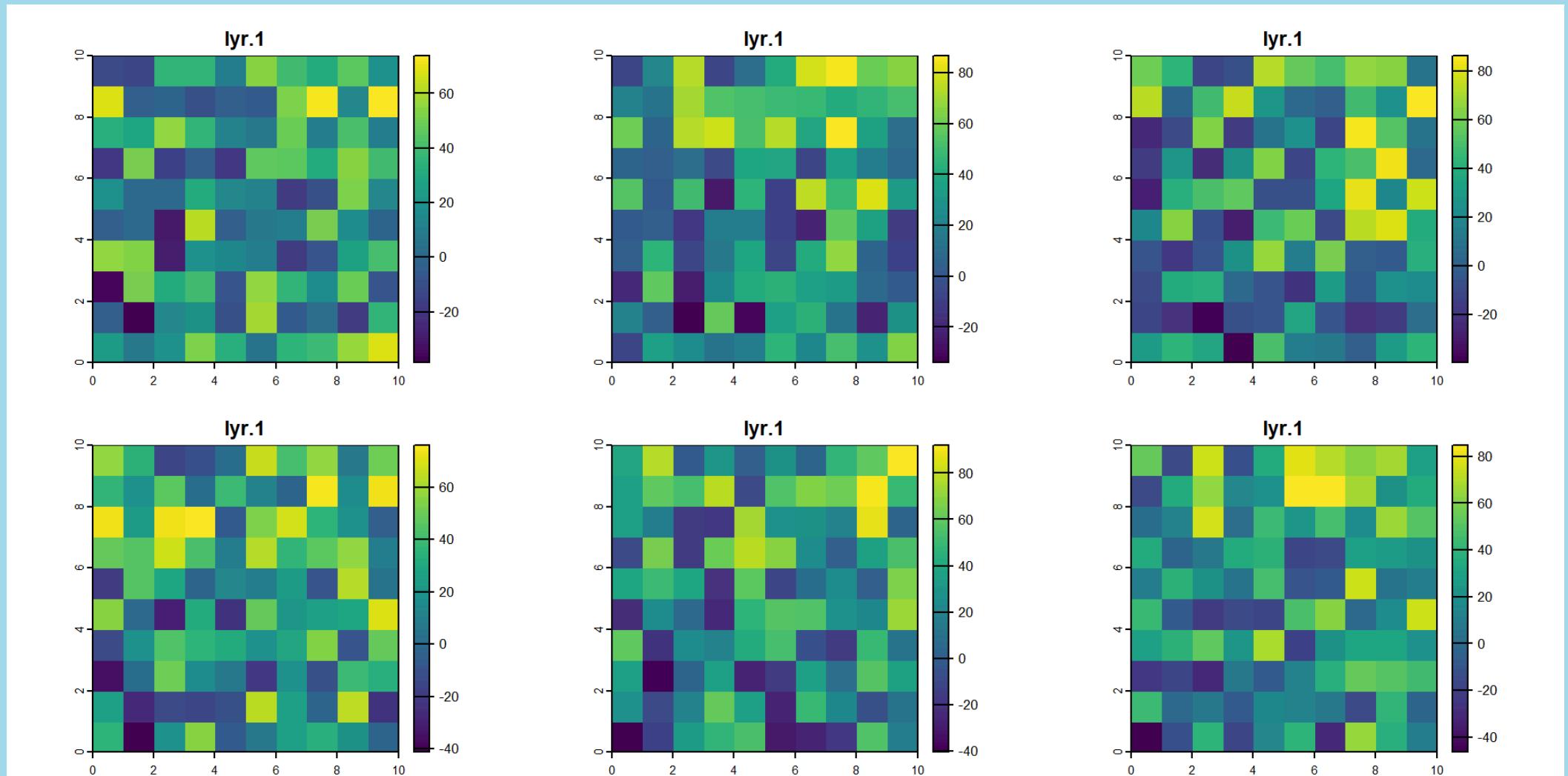
- Stochastic processes: variation makes each realization difficult to predict

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

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

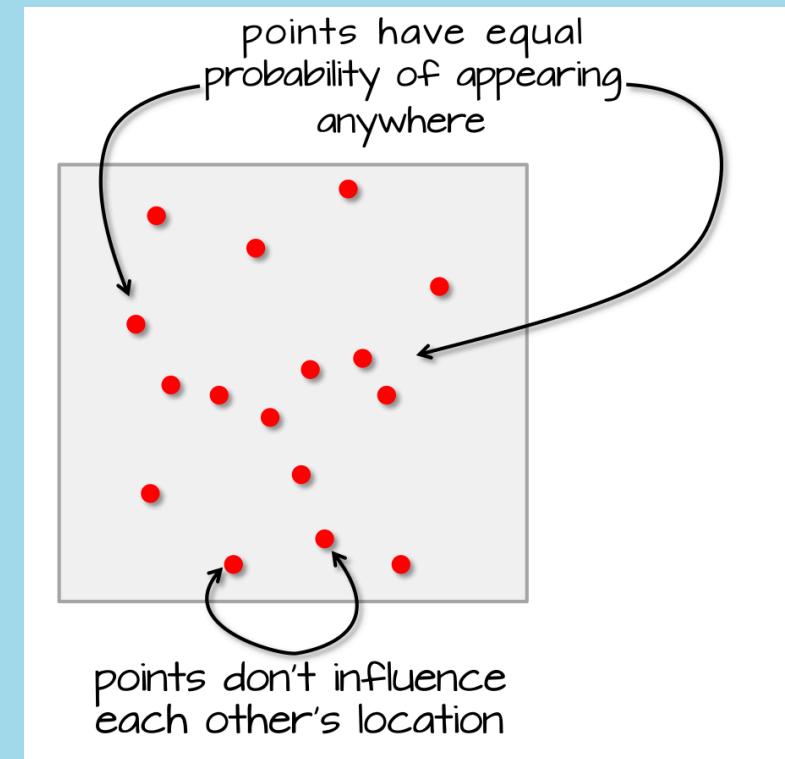
```
1 x <- rast(nrows = 10, ncols=10, xmin = 0,
2 values(x) <- 1
3 fun <- function(z) {
4 a <- z
5 d <- runif(ncell(z), -50,50)
6 values(a) <- 2 * crds(x) [,1] + 3*crds(x) [,2]
7 return(a)
8 }
9
10 b <- replicate(n=6, fun(z=x), simplify=FALSE)
11 d <- do.call(c, b)
```

# 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

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

# Revisiting Ripley's $K$

- Probability is tied to quadrat size – we can deal with this using a moving window
- If points have independent, fixed marginal densities, then they exhibit *complete, spatial randomness* (CSR)
- The  $K$  function is an alternative, based on a series of circles with increasing radius

$$K(d) = \lambda^{-1} E(N_d)$$

- We can test for clustering by comparing to the expectation:

$$K_{CSR}(d) = \pi d^2$$

- if  $k(d) > K_{CSR}(d)$  then there is clustering at the scale defined by  $d$

# Ripley's $K$ Function

- When working with a sample the distribution of  $K$  is unknown
- Estimate with

$$\hat{K}(d) = \hat{\lambda}^{-1} \sum_{i=1}^n \sum_{j=1}^n \frac{I(d_{ij} < d)}{n(n-1)}$$

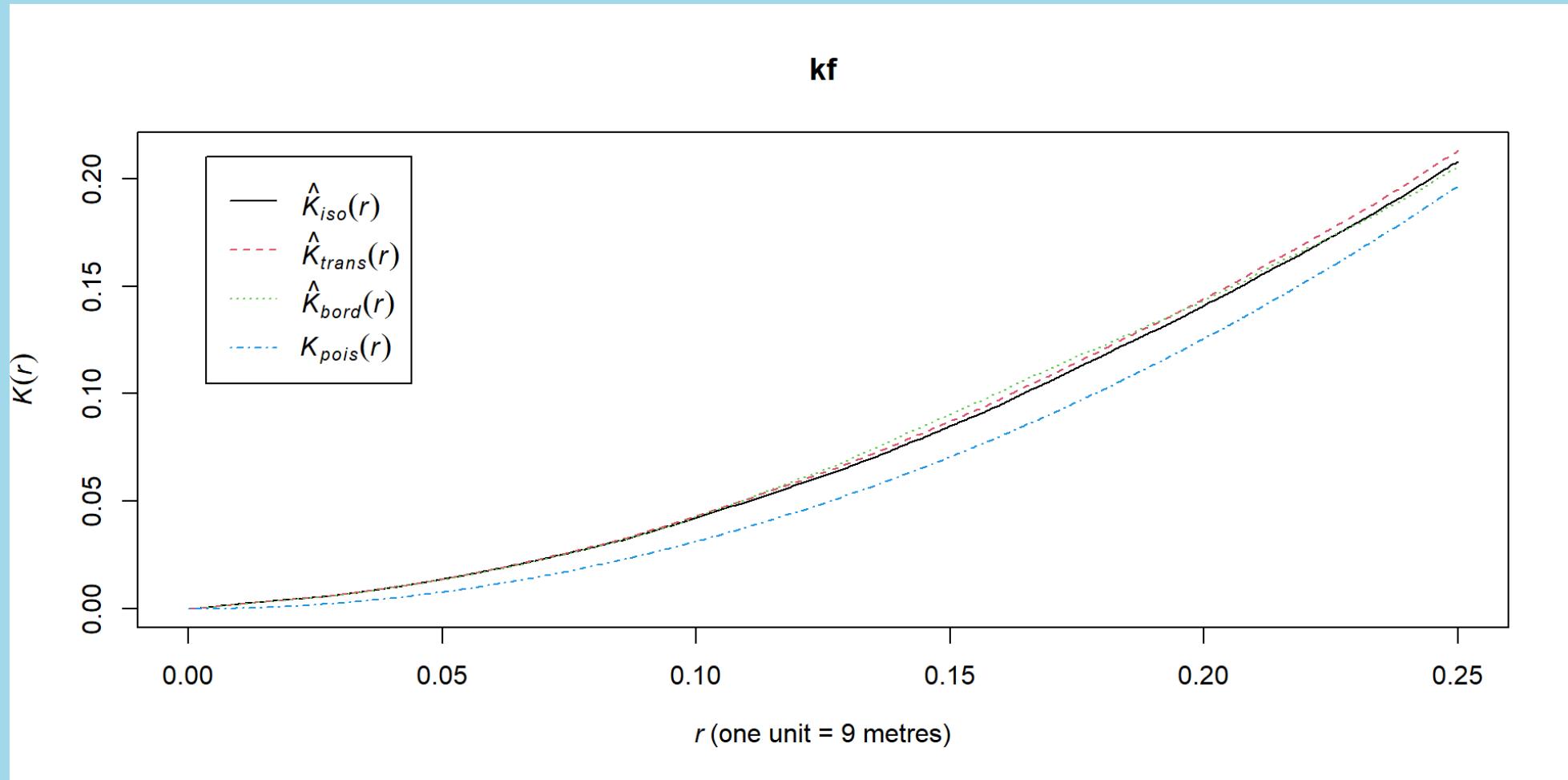
where:

$$\hat{\lambda} = \frac{n}{|A|}$$

# Ripley's $K$ Function

# Ripley's $K$ Function

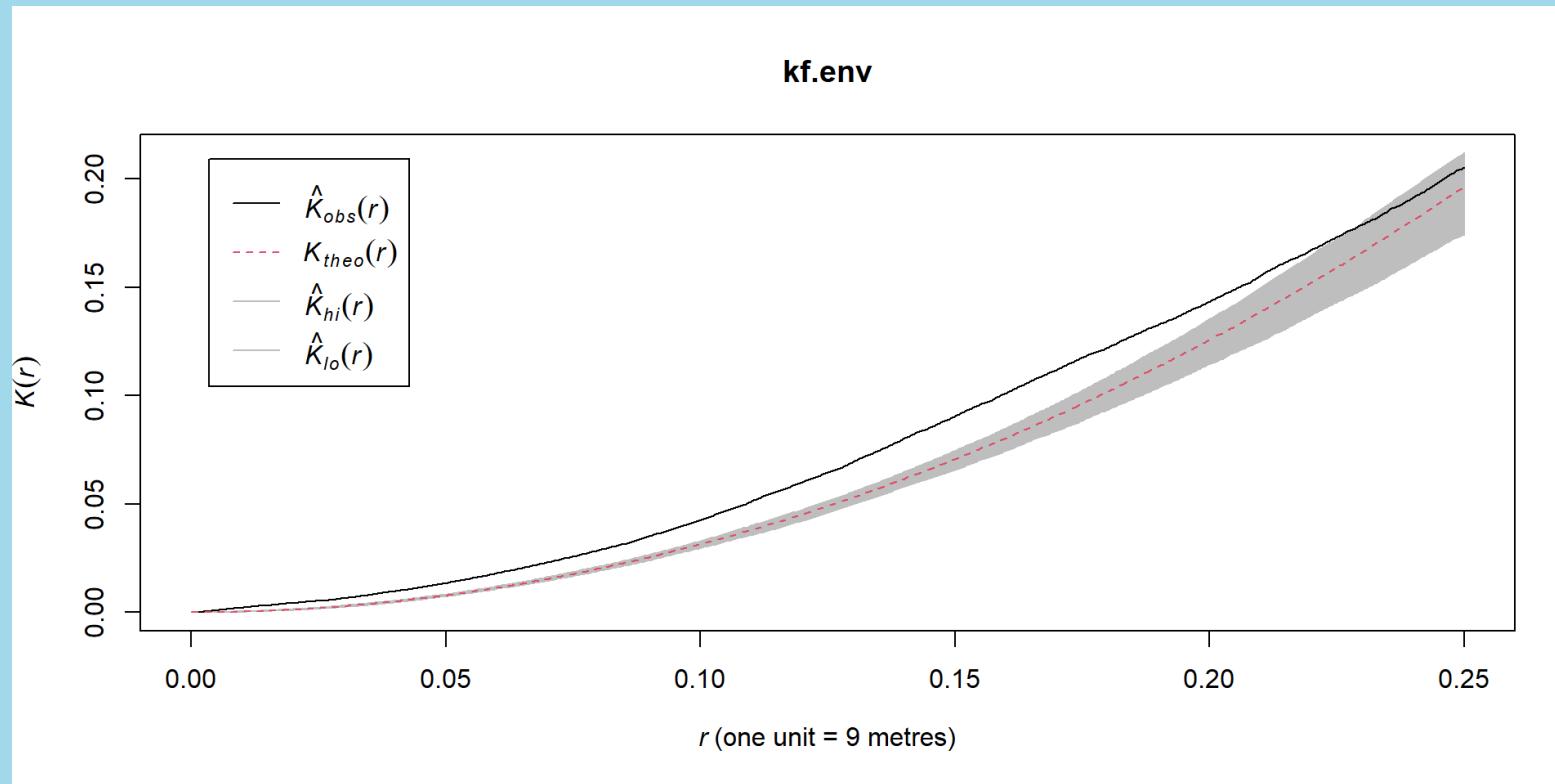
```
1 kf <- Kest(bramblecanes, correction="border")
2 plot(kf)
```



# Ripley's $K$ Function

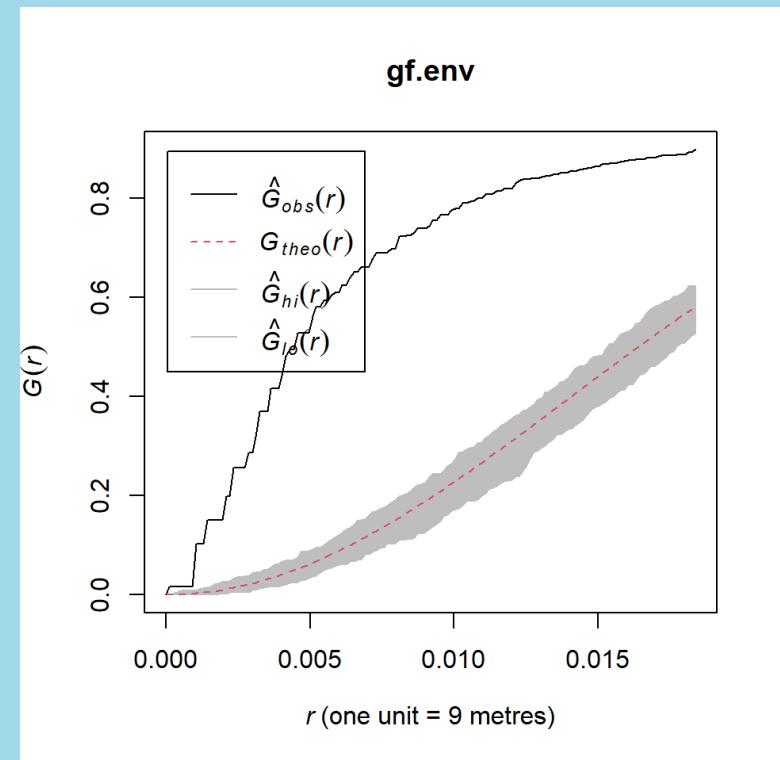
- accounting for variation in  $d$

```
1 kf.env <- envelope(bramblecanes, correction="border", envelope = FALSE, ver  
2 plot(kf.env)
```



# Other functions

- $L$  function: square root transformation of  $K$
- $G$  function: the cumulative frequency distribution of the nearest neighbor distances
- $F$  function: similar to  $G$  but based on randomly located points

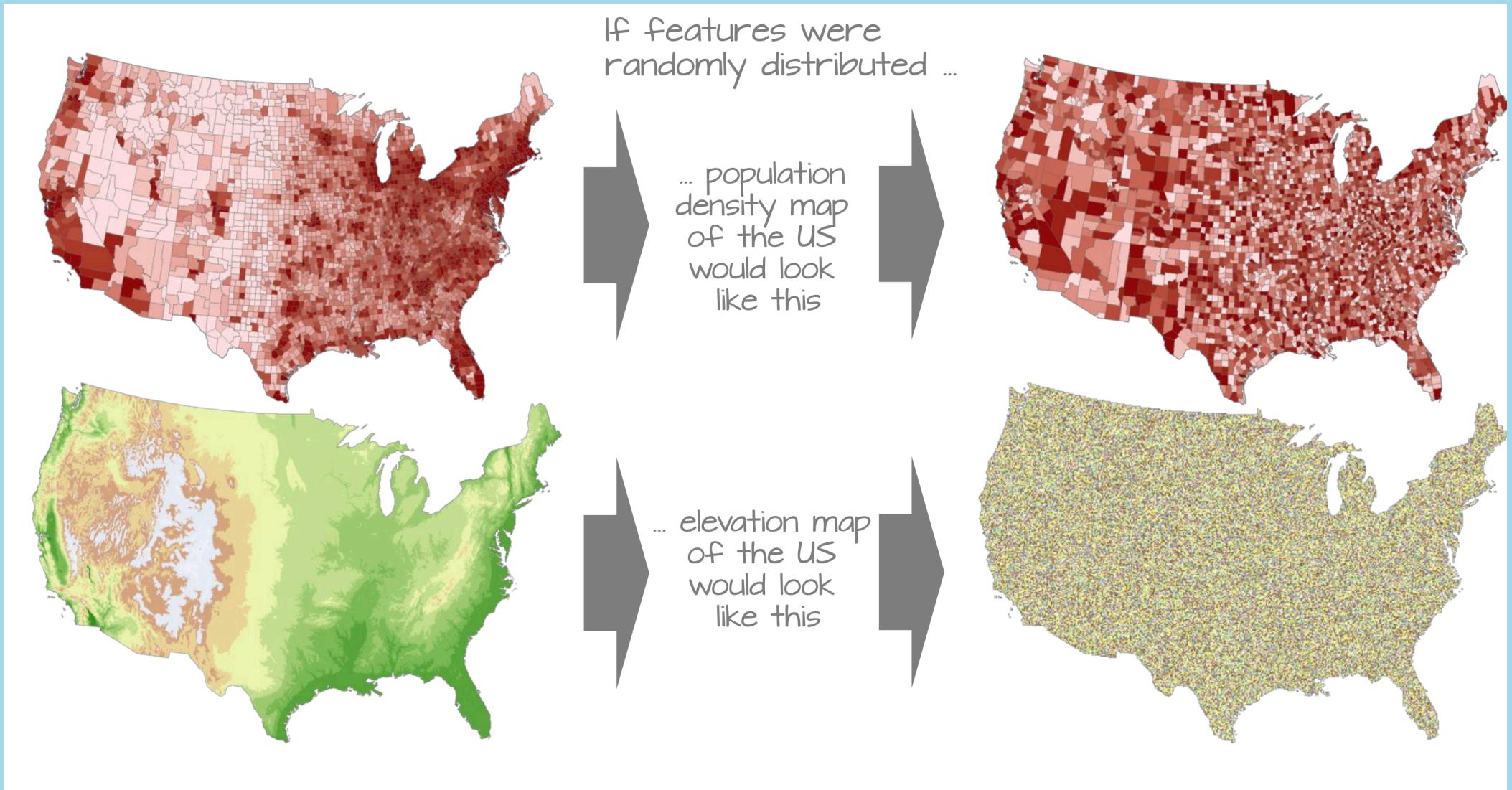


# Tobler's Law

'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

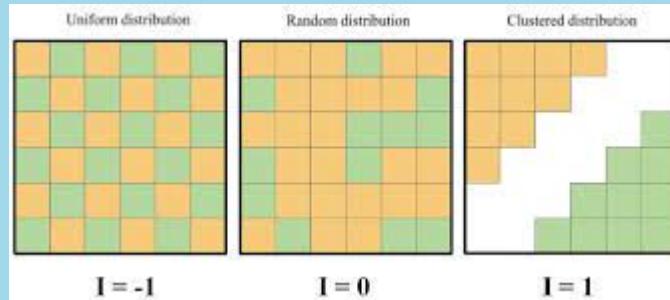


From Manuel Gimond

# (One) Measure of autocorrelation

- Moran's I

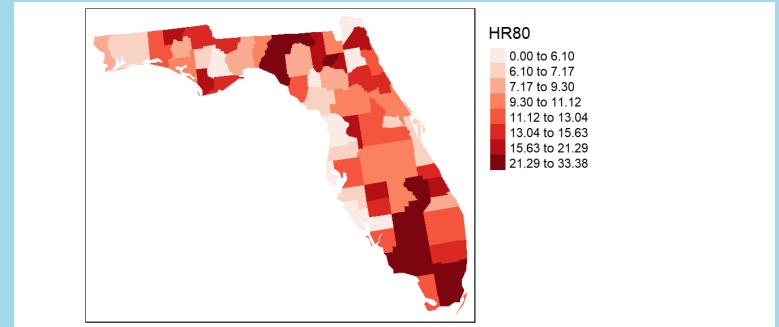
$$I(d) = \frac{\sum_i \sum_{j \neq i} w_{ij} (x_i - \bar{x})(x_j - \bar{x})}{S^2 \sum_i \sum_{j \neq i} w_{ij}}$$



# 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

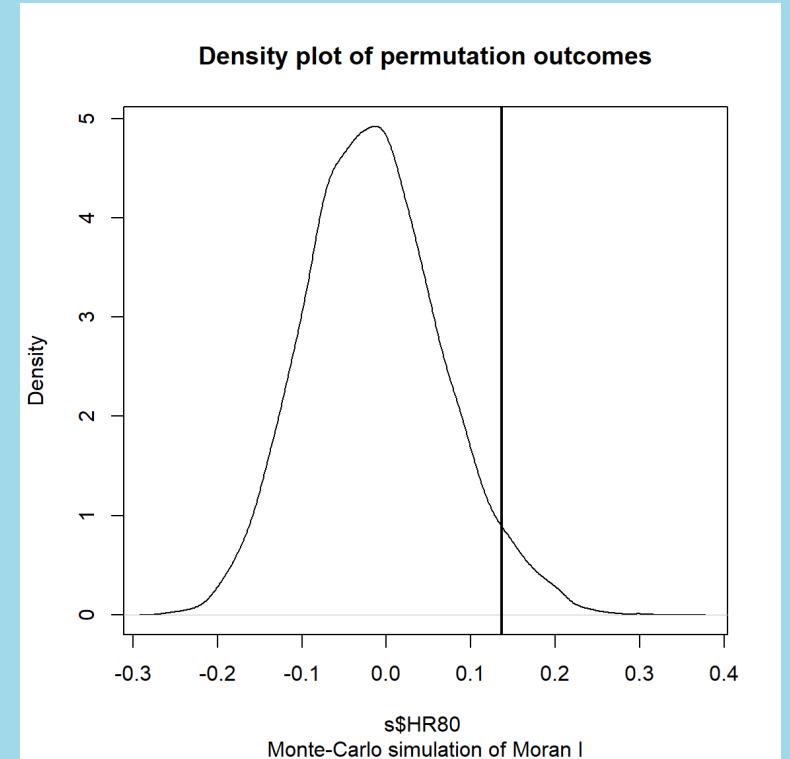
Moran I statistic standard deviate = 1.8891, p-value = 0.02944
alternative hypothesis: greater
sample estimates:
Moran I statistic      Expectation      Variance
0.136277593     -0.015151515     0.006425761
```

# 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,  $\mathbf{W}$ , 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

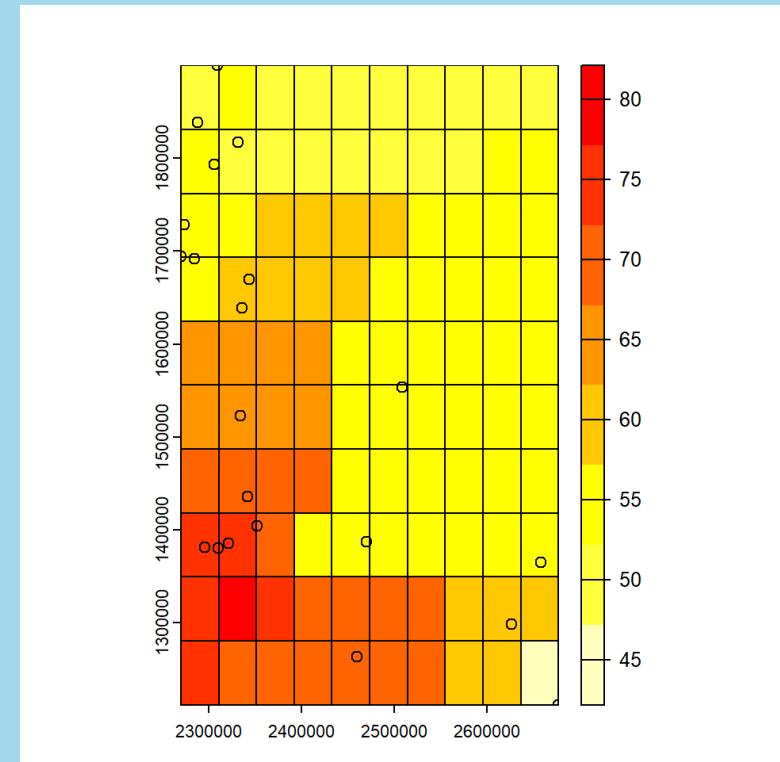
# Nearest neighbor

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

```

1 # data retrieved from https://www.epa.gov/outdoor-air-quality-
2
3 aq <- read_csv("/opt/data/data/classeexamples/ad_viz_plotval_da-
4   st_as_sf(., coords = c("Site Longitude", "Site Latitude"), c-
5   oordinates = c("Site Longitude", "Site Latitude"))
6   st_transform(., crs = "EPSG:8826") %>%
7   mutate(date = as_date(parse_datetime(Date, "%m/%d/%Y")))) %>%
8   filter(., date >= 2024-07-01) %>%
9   filter(., date > "2024-07-01" & date < "2024-07-31")
10 aq.sum <- aq %>%
11   group_by(., `Site ID`) %>%
12   summarise(., meanpm25 = mean(`Daily AQI Value`))
13
14 nodes <- st_make_grid(aq.sum,
15   what = "centers")
16
17 dist <- distance(vect(nodes), vect(aq.sum))
18 nearest <- apply(dist, 1, function(x) which(x == min(x)))
19 aq.nn <- aq.sum$meanpm25[nearest]
20 preds <- st_as_sf(nodes)
21 preds$aq <- aq.nn
22
23 preds <- as(preds, "Spatial")
24 sp::gridded(preds) <- TRUE
25 preds.rast <- rast(preds)

```



# 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

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

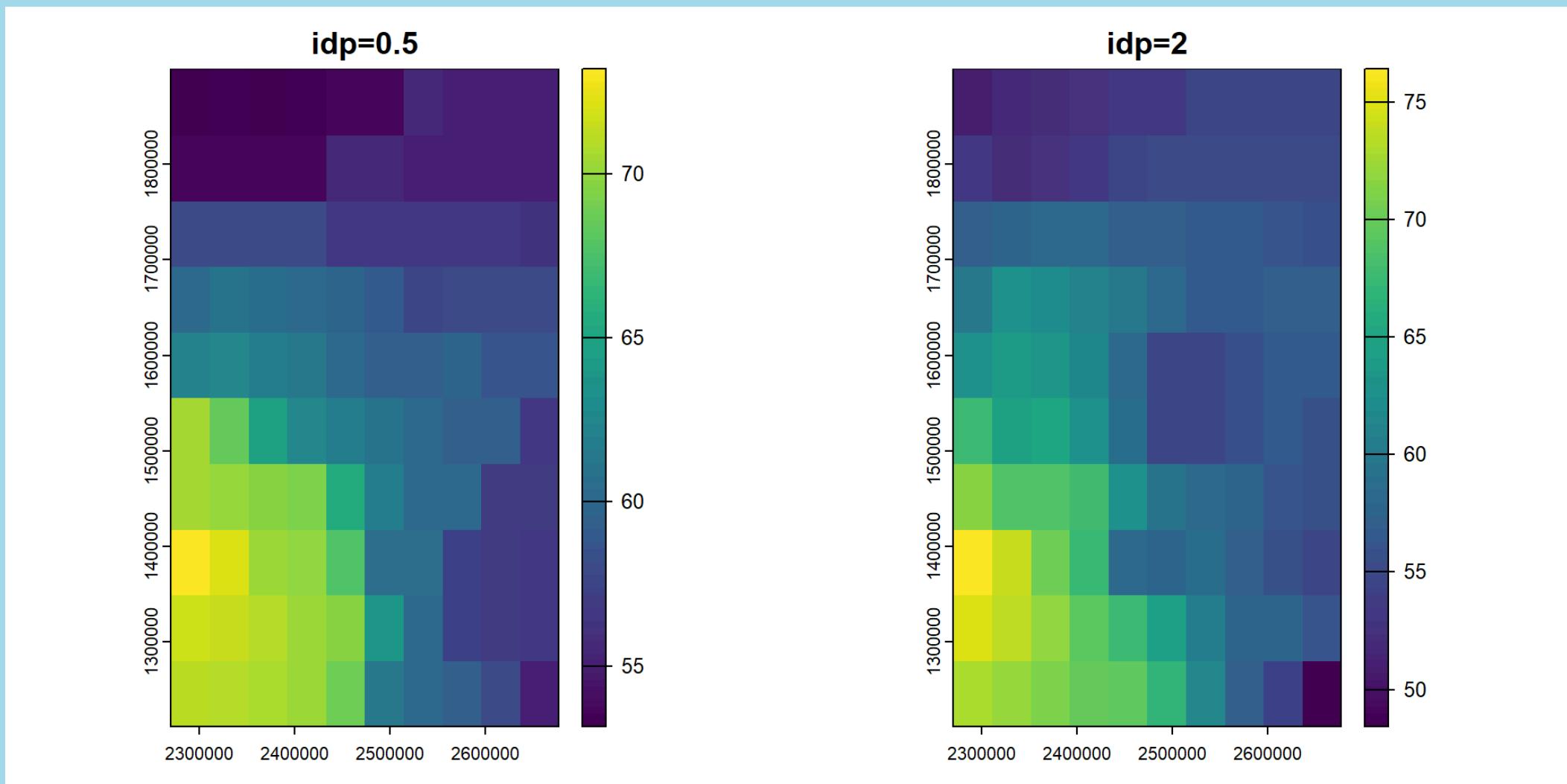
and  $\alpha > 0$  ( $\alpha = 1$  is inverse;  $\alpha = 2$  is inverse square)

# Inverse-Distance Weighting

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

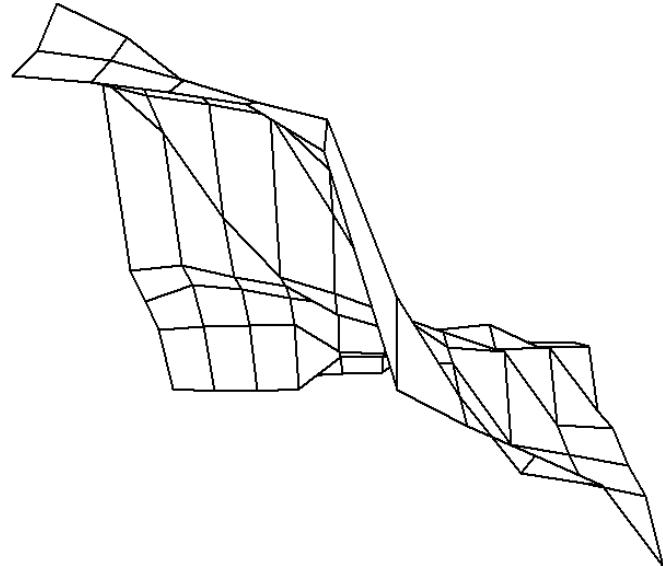
```
1 mgsf05 <- gstat(id = "meanpm25", formula = meanpm25~1, data=aq.sum, nmax=7
2 mgsf2 <- gstat(id = "meanpm25", formula = meanpm25~1, data=aq.sum, nmax=7,
3 interpolate_gstat <- function(model, x, crs, ...) {
4     v <- st_as_sf(x, coords=c("x", "y"), crs=crs)
5     p <- predict(model, v, ...)
6     as.data.frame(p) [,1:2]
7 }
8 zsf05 <- interpolate(preds.rast, mgsf05, debug.level=0, fun=interpolate_gst
9 zsf2 <- interpolate(preds.rast, mgsf2, debug.level=0, fun=interpolate_gstat
```

# Inverse-Distance Weighting



# Inverse-Distance Weighting

**idp=0.5**



**idp=2**

