(GEO)SPATIAL INTERPOLATION
USING POINT INFORMATION TO GENERATE SPATIAL SURFACES

Alejandro Ordonez

Assistant Professor - Department of Bioscience

Section for Ecoinformatics & Biodiversity

Center for Biodiversity Dynamics in a Changing World (BIOCHANGE)

WHAT WE WILL TALK TODAY

1. Spatial Interpolation basics
    1. Using distances: Inverse distance weighting or Bilinear/Cubic interpolation
    2. Using regressions: Linear regressions and locations as predictors
    3. Using Splines: Splines and GAMS

2. Leveraging the Spatial knowledge
    1. Building a variogram: The main ingredient of Geostatistical methods
    2. Kriging: Univariate Geostatistical method
    3. Co-kriging: Multivariate Geostatistical method

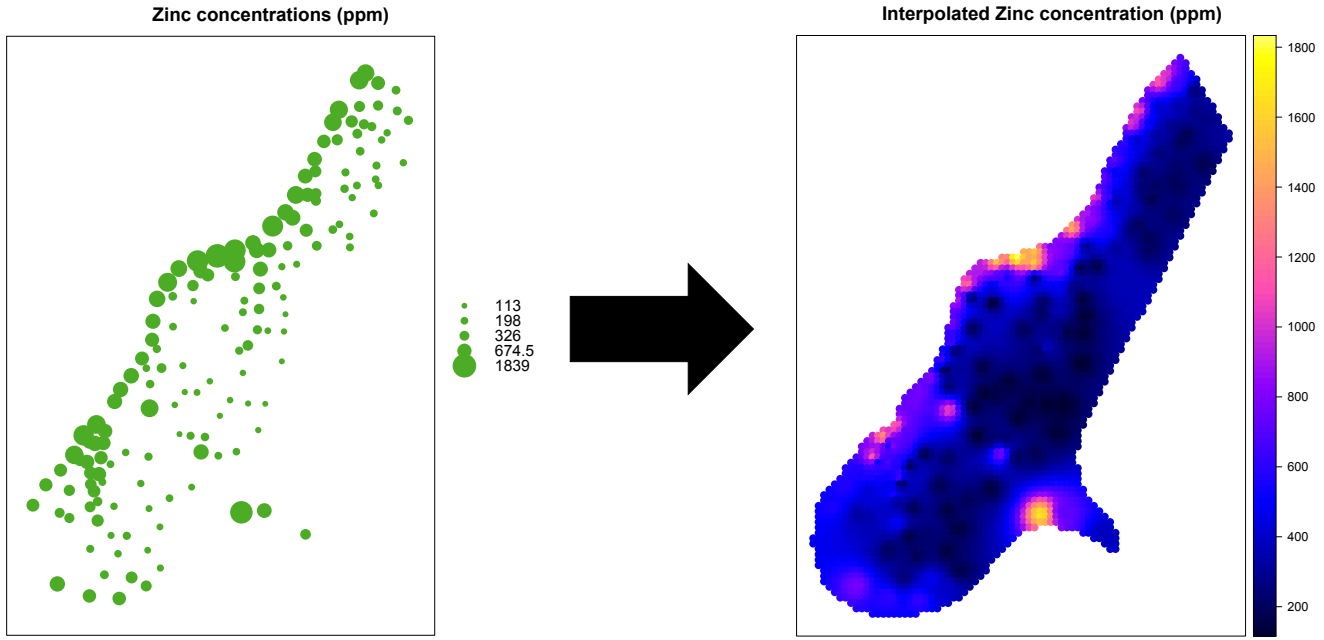
Any questions?

Ready to start?

INTERPOLATION AND GEOSTATISTICS

Typically, a limited number of sometimes arbitrarily chosen sample locations for which measurements/information on the random variable of intertest are available...

but you would like to know how the values of the variable across the study domain.



How can we "fill the gaps" between locations with information?

NON-GEOSTATISTICAL INTERPOLATION METHODS

Basic regression approaches can be used to “interpolate” the value of a variable of interest

- Across the study spatial domain (usually done a regular grid).
- Individual points/polygons defining locations or regions and located in a (i)regular pattern.

These are non-geostatistical “interpolation” approach based on

- Distance as a predictors:
  - Inverse Distance Weighted Interpolation, Bilinear/cubic interpolation
- A variable (locations) as a predictor:
  - Linear regression approaches, splines

GEOSTATISTICAL VS NONGEOSTATISTICAL

Why are some interpolation methods non-geostatistical?

Method such Distance Weighted Interpolation, Bilinear/cubic interpolation are based on the idea that things that are close to one another are more alike than those that are farther apart.

- Proximity defined importance of a known location to predict a ”new” point.

But you do not know how the degree of spatial dependence of a spatial random field (variable)

- Defining a model that describes this dependency (aka the variogram) makes a method Geostatistical.

So far so good?

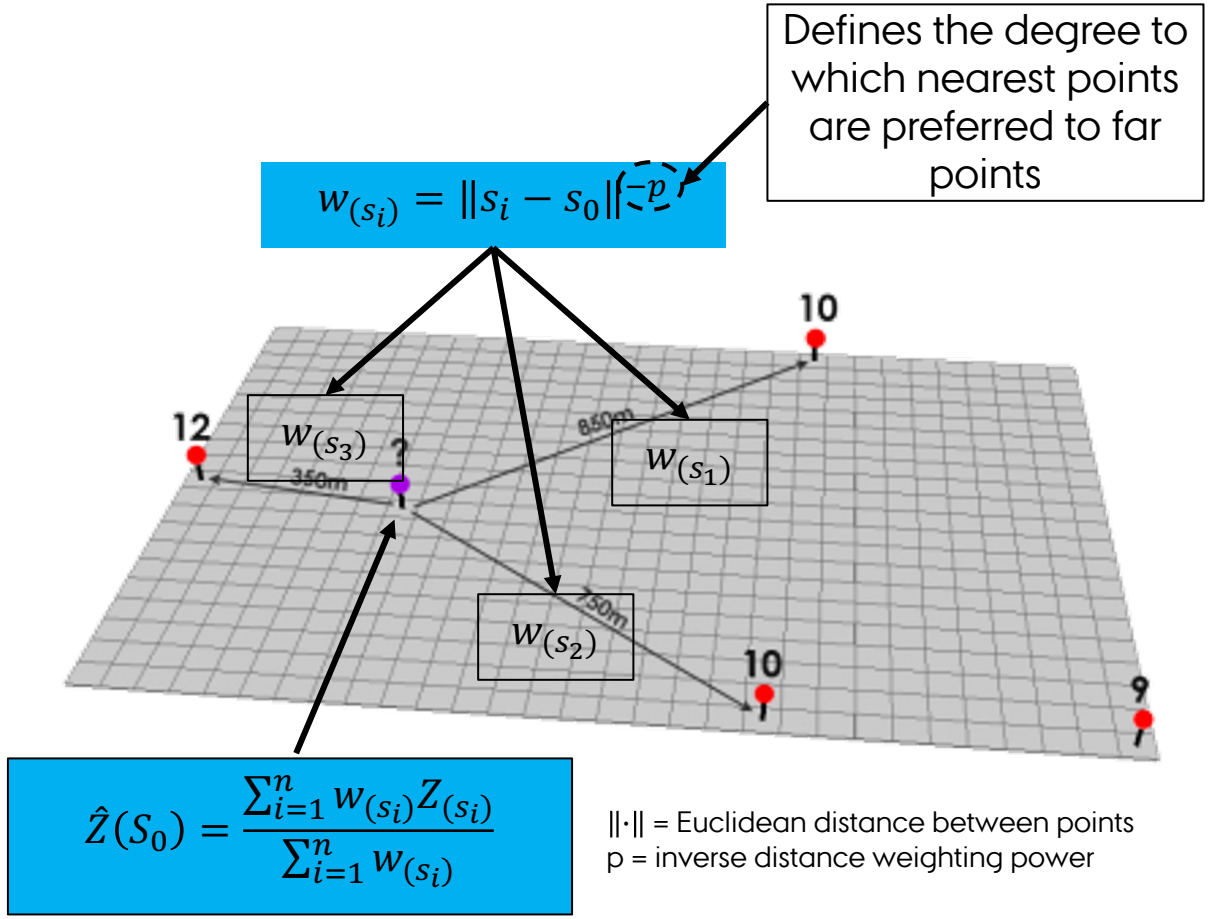
Any questions?

Ready to move on?

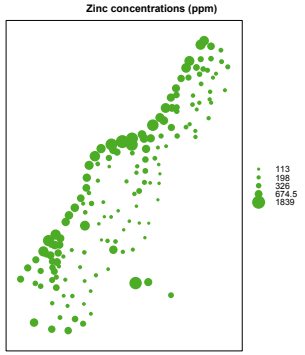
INVERSE DISTANCE-BASED WEIGHTED INTERPOLATION (IDW)

Values are computed using a weighed average approach, where the “weights” are defined by the distance to the “interpolated” location.

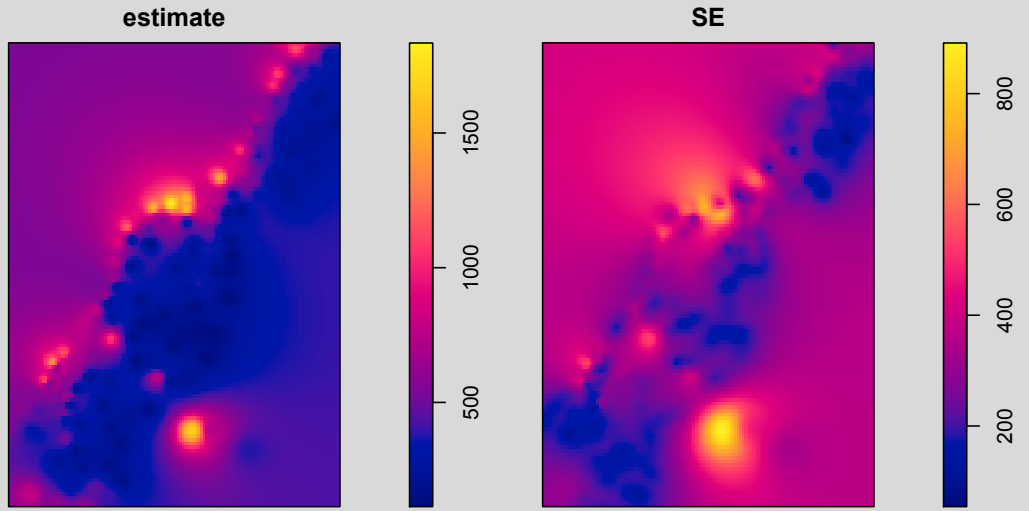
It builds from the idea that near points are more alike than far points…



INVERSE DISTANCE WEIGHTED INTERPOLATION IN R

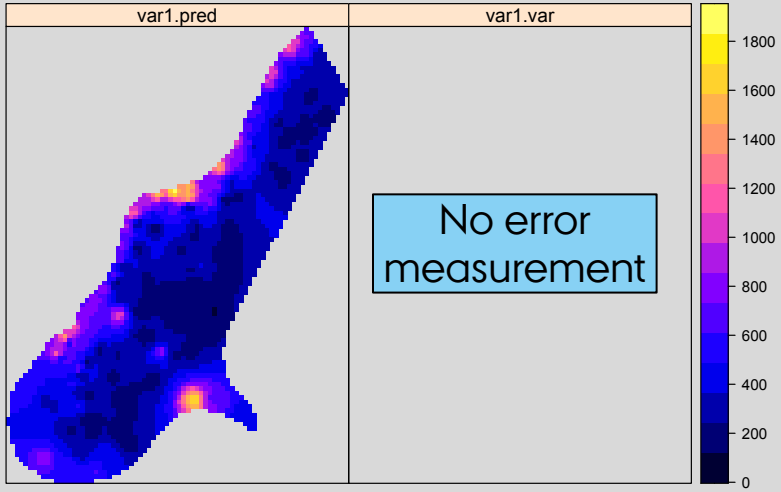


idw function in spatstat




```
plot(idw.spatstat)
```

idw function in gstat




```
spplot(idw.gstat,
       colorkey = TRUE)
```

BILINEAR AND CUBIC INTERPOLATION

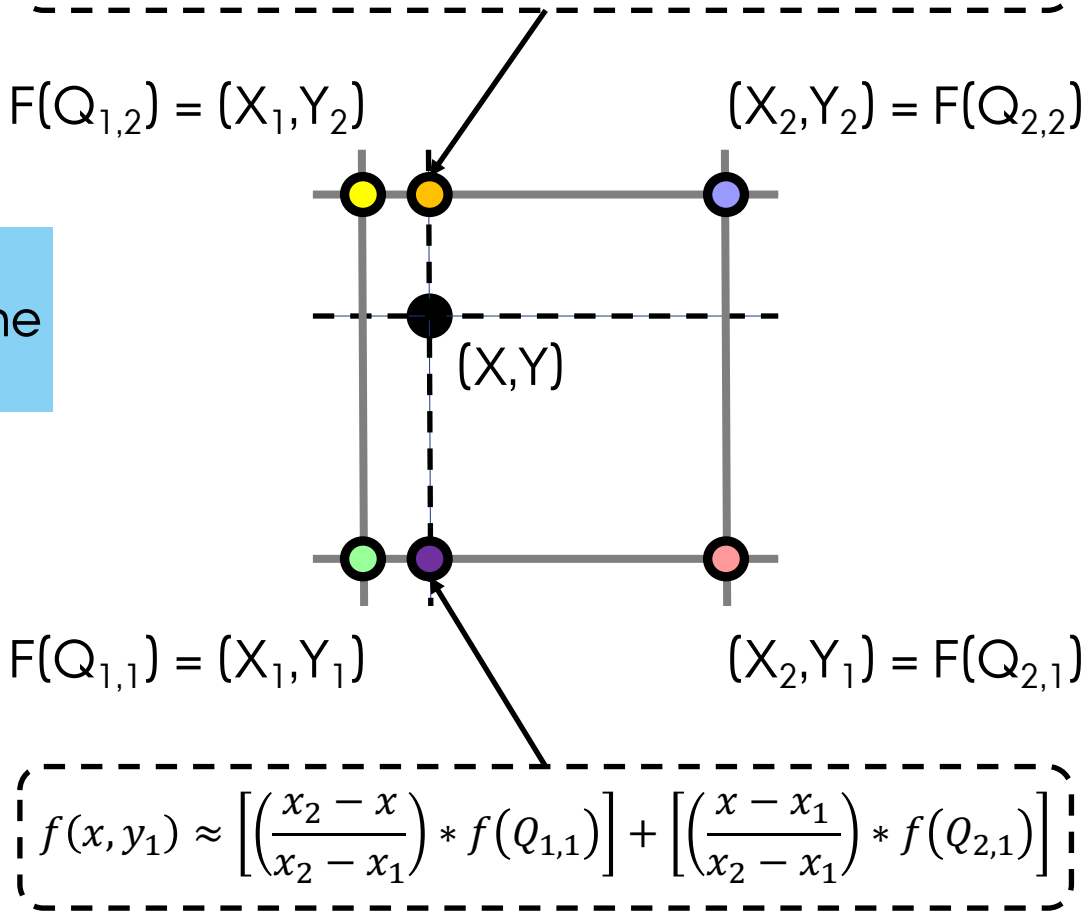
Bilinear and cubic interpolation are techniques for calculating values of a location-based on nearby grid cells using distances as weights

- *Bilinear interpolation* uses the **4** closest cell centers and takes a weighted average of these locations
- *Cubic interpolation* uses the **16** closest cell centers and takes a weighted average of these locations.

Both approaches are implemented in the `interp` function of the `akima` package

BILINEAR INTERPOLATION

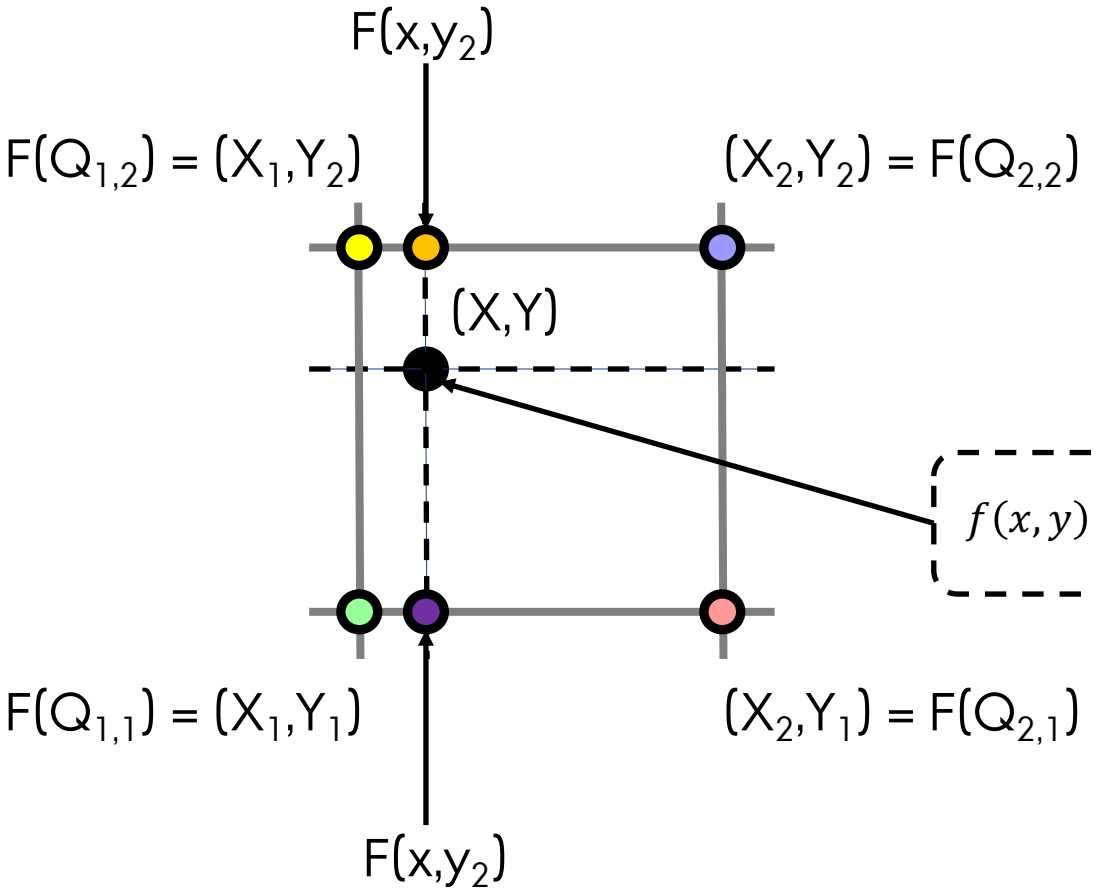
$$f(x, y_2) \approx \left[\left(\frac{x_2 - x}{x_2 - x_1}\right) * f(Q_{1,2})\right] + \left[\left(\frac{x - x_1}{x_2 - x_1}\right) * f(Q_{2,2})\right]$$



Step 1:
Linear interpolation in the X direction

$$f(x, y_1) \approx \left[\left(\frac{x_2 - x}{x_2 - x_1}\right) * f(Q_{1,1})\right] + \left[\left(\frac{x - x_1}{x_2 - x_1}\right) * f(Q_{2,1})\right]$$

BILINEAR INTERPOLATION

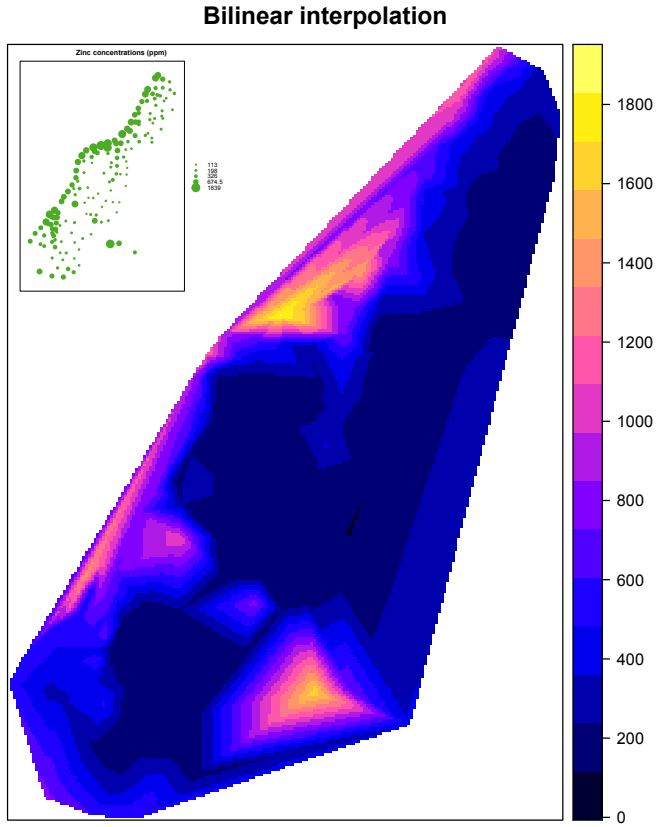


Step 2:
linear interpolation of values generated in step 1 in the Y direction

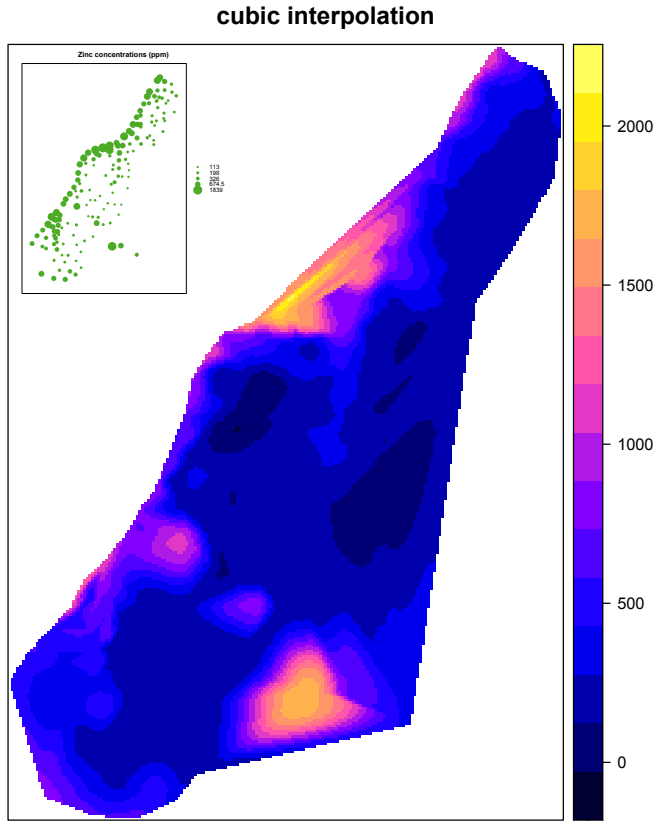
$$f(x,y) \approx \left[\left(\frac{y_2 - y}{y_2 - y_1}\right) * f(x,y_1)\right] + \left[\left(\frac{y - y_1}{y_2 - y_1}\right) * f(x,y_2)\right]$$

BILINEAR AND CUBIC INTERPOLATION IN R

Bilinear interpolation



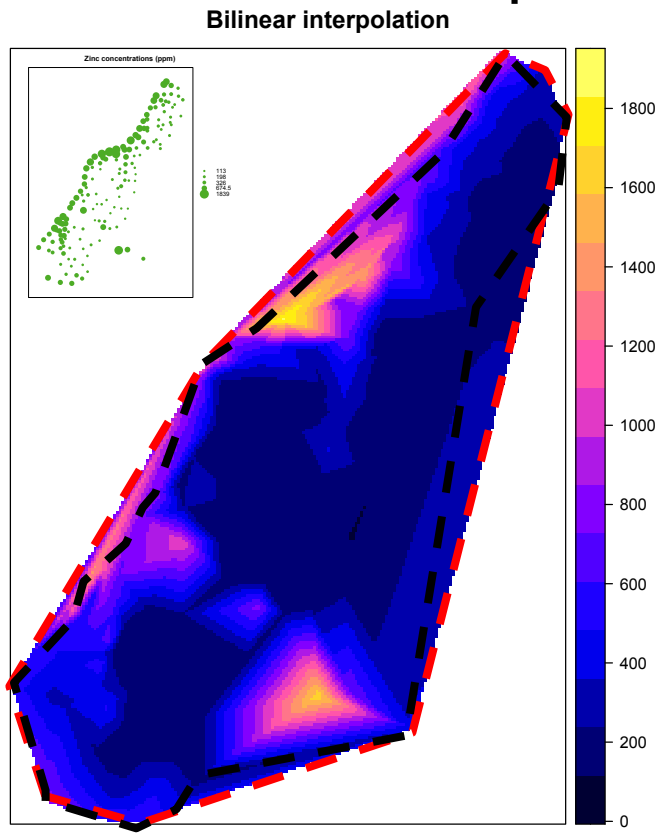
Cubic interpolation



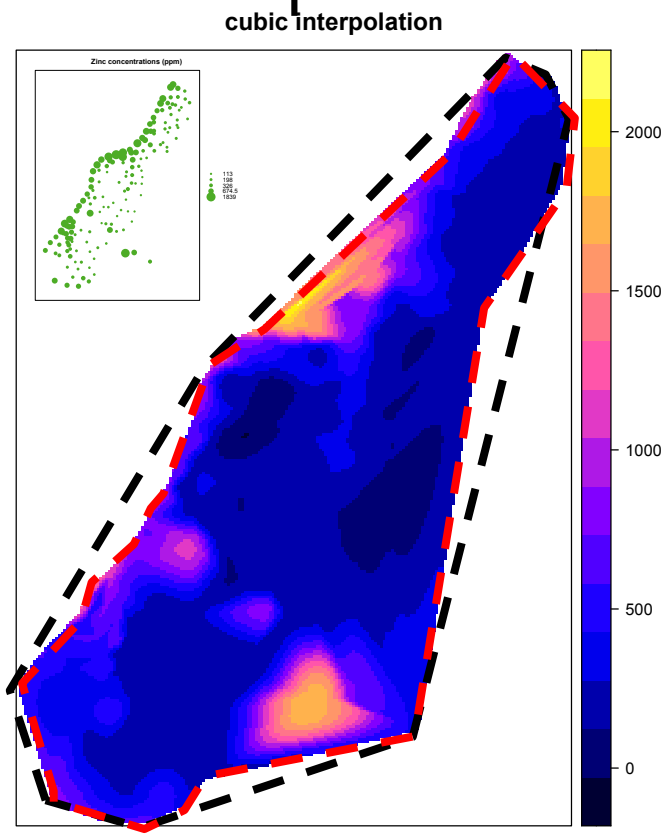
ALEJANDRO ORDONEZ GLORIA
ASSISTANT PROFESSOR

BILINEAR AND CUBIC INTERPOLATION IN R

Bilinear interpolation



Cubic interpolation



The white areas, are the regions outside the convex hull defined by the observations

The difference in shape of convex hulls relates to the difference in “used points

So far so good?

Any questions?

Ready to move on?

SPLINES AS AN INTERPOLATION MODELS

- All approaches are based on the idea of 2-dimensional smoothing splines.
- As IDW and bilinear/cubic interpolation, these methods are not "geostatistical" in the sense that there is no model of the spatial structure of the data.
- The only goal here is to generate a "smooth" surface.

I will present two options

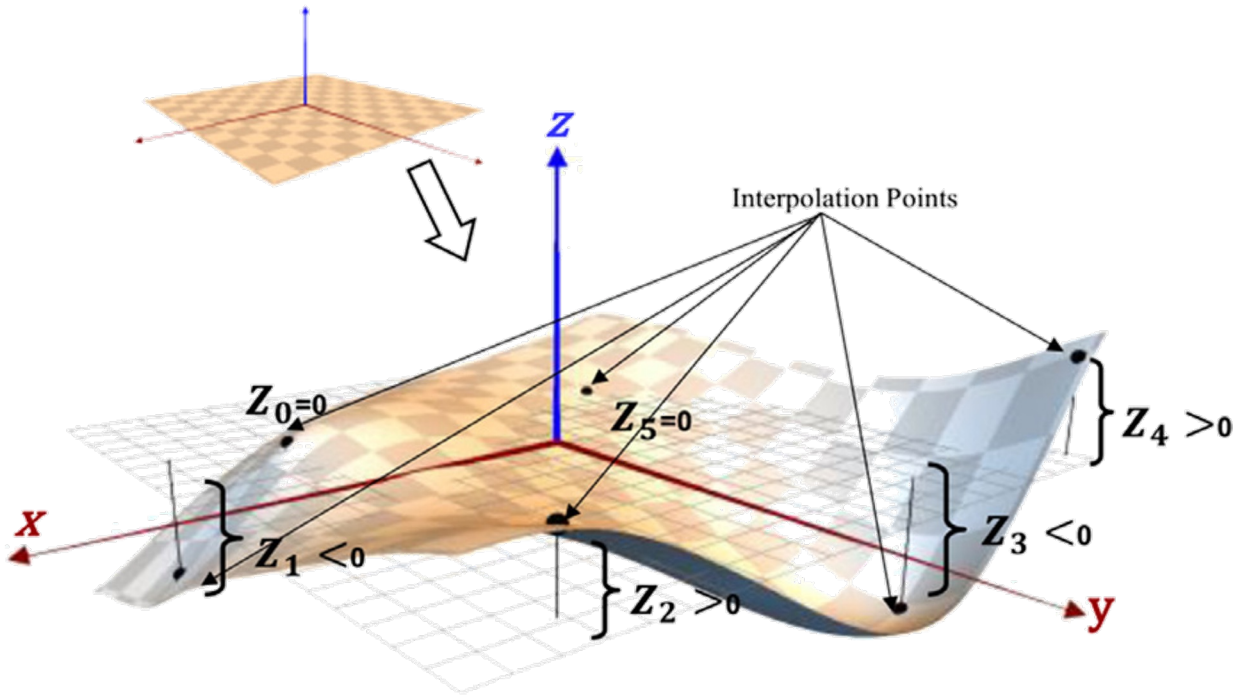
- **Thin-plate splines**: the smoothing, is based on the rigidity of the "surface".
- **GAMs**: Generalised Additive Model where you generate a "smoothing" function that optimized the fit to the used data.

THIN-PLATE SPLINES (TPS) IN R

A good analogy for the method is a a thin (so, flexible) plate that is warped to fit the data.

This “surface” can be:

- Very “rigid”, i.e., just a single surface (the usual least-squares).
- Very “flexible”, (i.e., perfectly fitting every observation).



THIN-PLATE SPLINES (TPS)

This is a 2-dimensional smoothing spline focused on minimize the residual sum of squares (RSS) of the fitted function → a surface in between rigid and flexible.

- This rigidity is defined as a **roughness penalty**: balance between the fit to the observations and smoothness of the function.

- TPS balance between the smoothness and the penalty is defined using a cross-validation approach.

THIN-PLATE SPLINES (TPS)

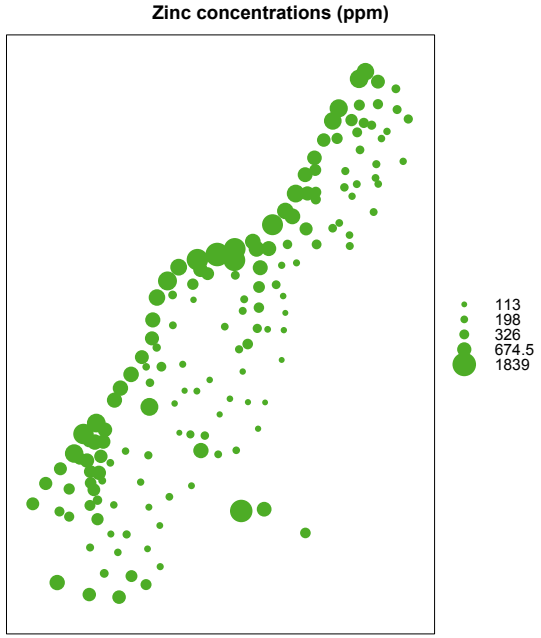
The function to implement a TPS interpolation is `Tps` and is part of the `fields` package (which also includes functions for kriging, sampling design, and visualization)

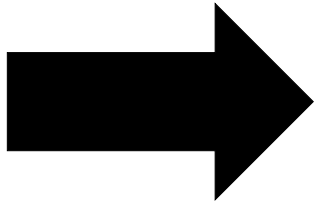
- BTW, The fields package does not use `Spatial*` classes or `data.frames`
- It requires two arguments
  - A vector of dependent variables
  - A matrix of coordinates (rows are observation locations, columns are the two coordinates)

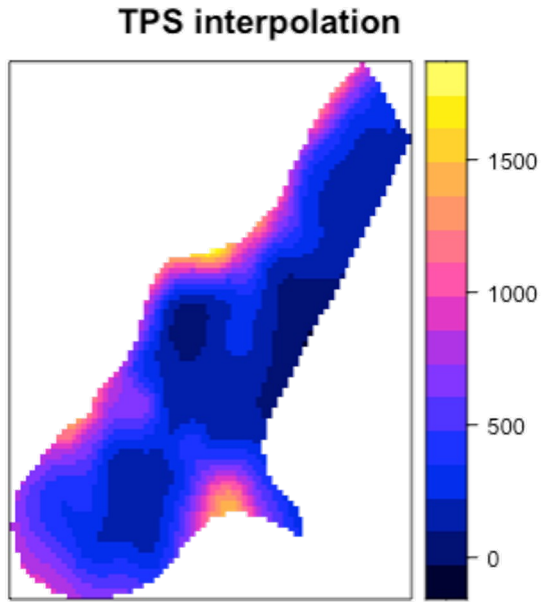
Both of these can be directly extracted form any `Spatial*` Object

THIN-PLATE SPLINES (TPS) IN R

```
require(fields)
data(meuse)
## Build a thin plate spline model
surf.1 <-Tps(x = meuse[,c("x","y")],#Matrix of independent variables (locations)
             Y = meuse$zinc,  # Vector of dependent variables
             lon.lat = T)  # interpreted as longitude and latitude
```







GENERALISED ADDITIVE MODELS (GAMS)

Can handle many distributions of the response variable [normal, binomial, counts]

Terms are simply added together, but each term is a non-parametric smoothed function

$$[g_{(u)} = \beta_0 + f_1(x_{i1}) + \cdots + f_p(x_{ip})]$$

This is a statistical model

Generalised **Additive** **Models**

GENERALISED ADDITIVE MODELS
THE TWO PACKAGES

Using gam

```
gam(Y.Var ~ lo(Xvar, degree=-1, span = 0.5))
```

Degree of the polynomial repression

-1 = linear

Proportion of the observations in the window

Using mgcv

```
gam(Y.Var~s(Xvar, fx=F, k=-1, bs="cr"))
```

fx=F, Spline type fixed (T) penalized (F)

k=-1, Basis function

This default finds the optimal amount of smoothing

bs="cr" Spline family cr=cubic tp=thin-plate

GENERALISED ADDITIVE MODELS FOR TWO DIMENSIONAL DATA

Two approaches:

- Use data measured across the region (just as in a regression approach).
- Use the positional data (Lat/long or Y/X) as predictors.

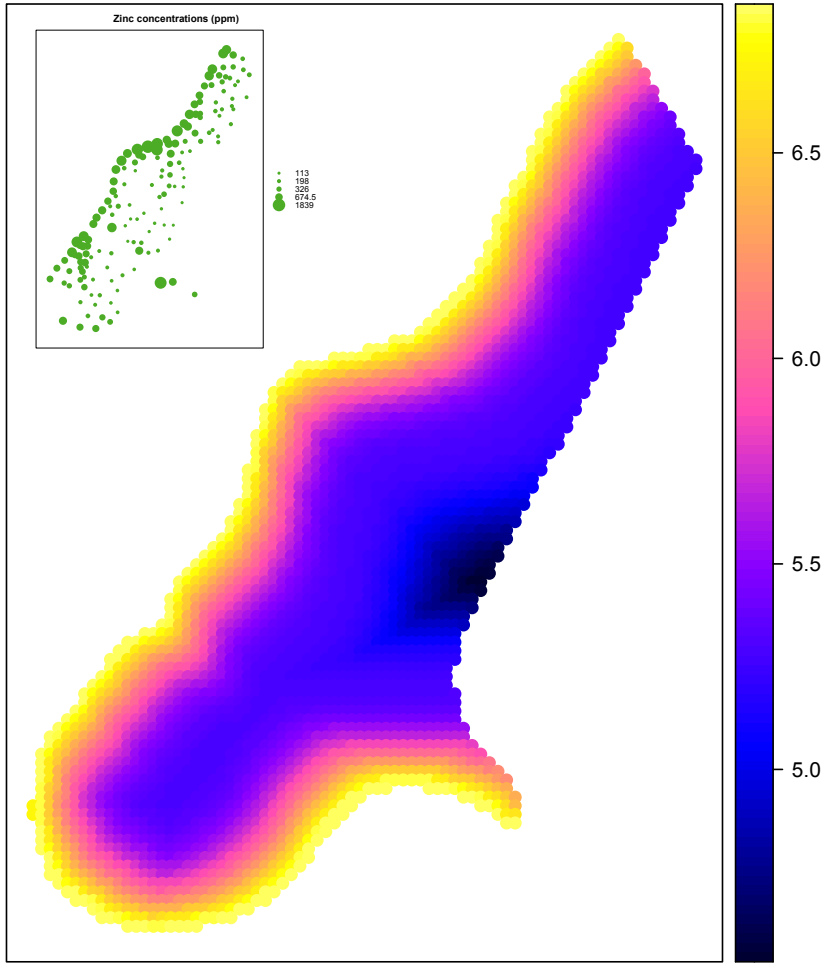
Multiple model specifications depending on the predictors:

- Add each variable individually → as in a regression.
- Use smoothing functions (tensor product smooths) that represent interactions between variables.

GENERALISED ADDITIVE MODELS WITH A PREDICTOR

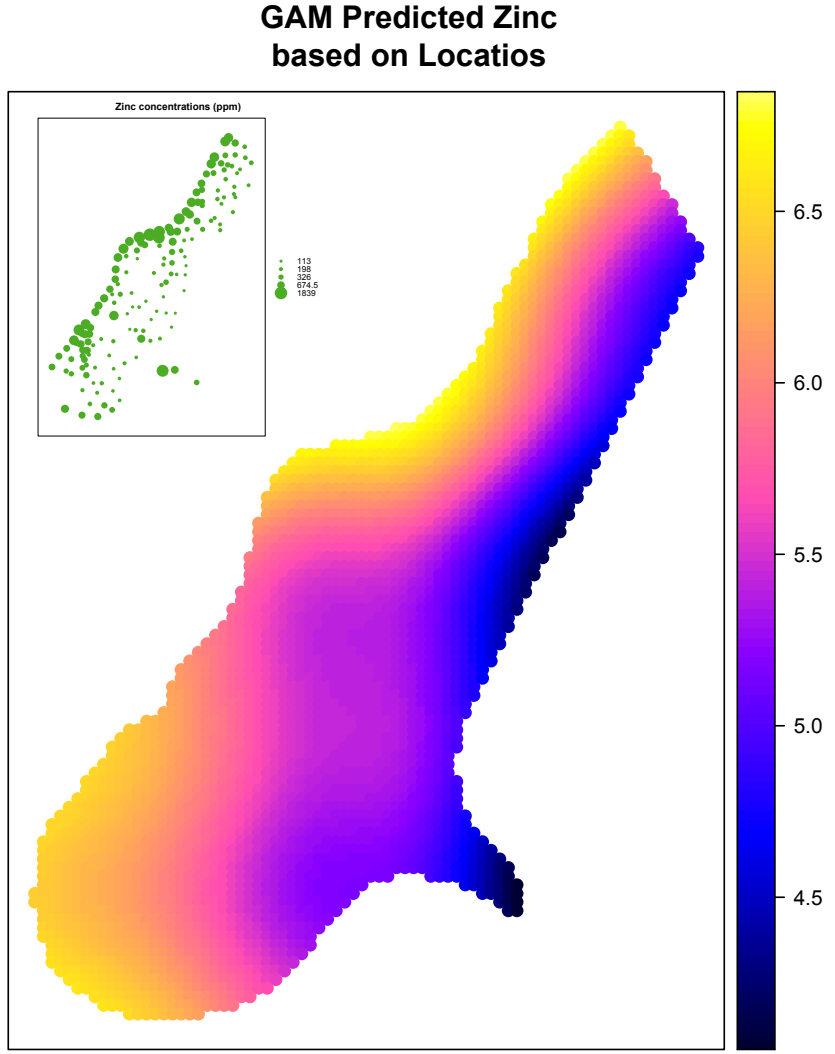
```
## fit model using dist as predictor
surf.1  <- gam(log(zinc) ~s(dist),data= meuse)
## Predict the model
fit1 <- predict.gam(surf.1 ,
                    newdata = meuse.grid)

## Make the predictions a Spatial* Object
meuse.grid $fit1 <- fit1
meuse.SpaGrid <- meuse.grid
coordinates(meuse.SpaGrid) <- ~x + y
spplot(meuse.SpaGrid,"fit1", colorkey = TRUE,
       main="GAM Predicted Zinc")
```

GAM Predicted Zinc based on distance



GENERALISED ADDITIVE MODELS IN TWO DIMENSIONS

```
## fit model using spatial locations as predictors
surf.1  <- gam(log(zinc) ~s(x)+s(y),data= meuse)
## Predict the model
fit1 <- predict.gam(surf.1 ,
                    newdata = meuse.grid)
## Make the predictions a Spatial* Object
meuse.grid $fit1 <- fit1
meuse.SpaGrid <- meuse.grid
coordinates(meuse.SpaGrid) <- ~x + y
spplot(meuse.SpaGrid,"fit1", colorkey = TRUE,
        main="GAM Predicted Zinc\nbased on
Locatios")
```

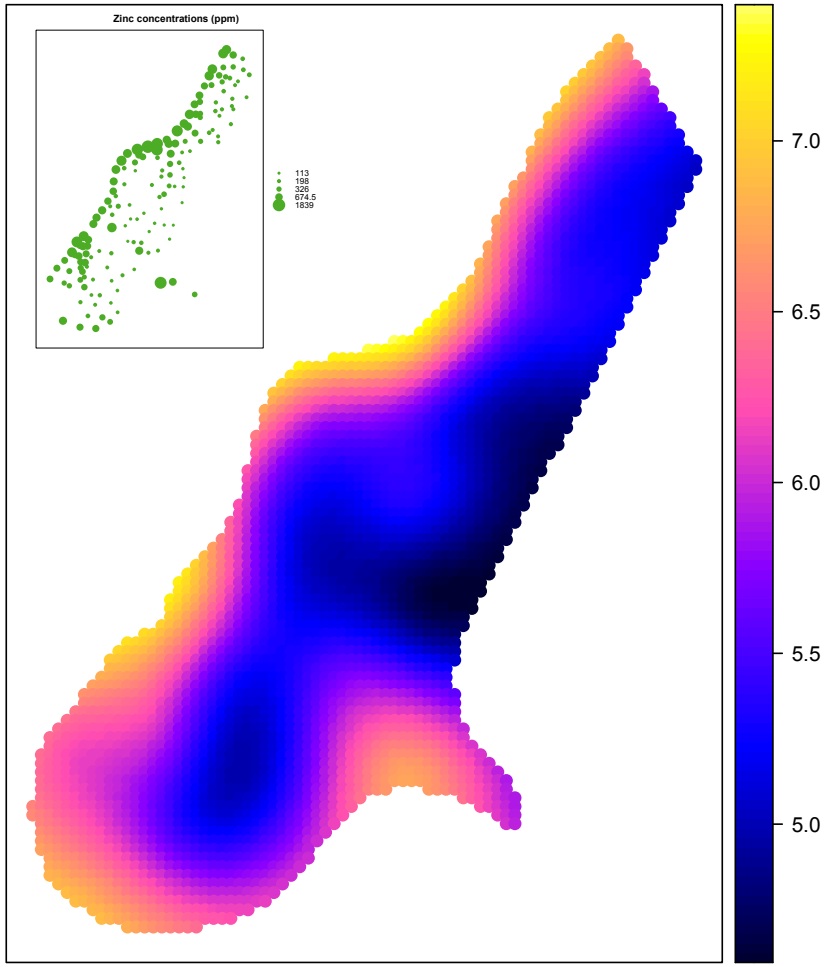


GENERALISED ADDITIVE MODELS IN TWO DIMENSIONS

```
## fit model using Interacting Locations.
surf.1  <- gam(log(zinc) ~s(x,y),data= meuse)
## Predict the model
fit1 <- predict.gam(surf.1 ,
                    newdata = meuse.grid)
## Make the predictions a Spatial* Object
meuse.grid $fit1 <- fit1
meuse.SpaGrid <- meuse.grid
coordinates(meuse.SpaGrid) <- ~x + y
spplot(meuse.SpaGrid,"fit1", colorkey = TRUE,
        main="GAM Predicted Zinc\nbased on
Interacting Locations")
```

Why you can use a two dimensional smoothing and not tensor product smooths

GAM Predicted Zinc based on Interacting Locatios



So far so good?

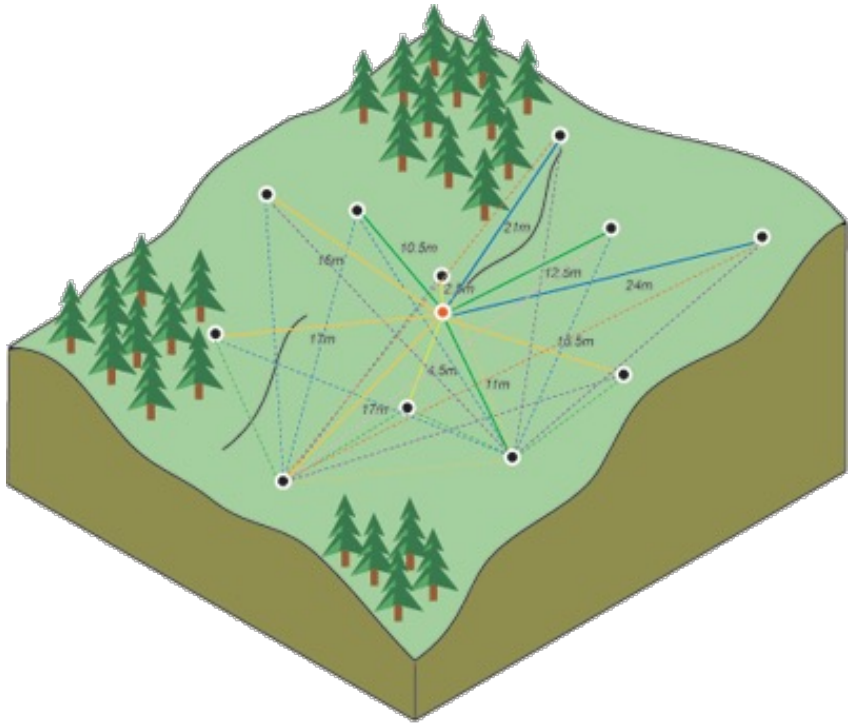
Any questions?

Ready to move on?

WHAT IS REGRESSION KRIGING?

A technique to predict the value of a function at a given point by combining

1. A regression of the dependent variable on auxiliary variables
2. Kriging of the regression residuals.



Kriging uses the variogram to compute **a weighted average** of the known values of the function in the neighborhood of the point.

KRIGING BASICS

Kriging approaches are build on the principle that the (semi)variogram can be used to define a series of weights to the residuals.

$$Z_{(s_o)} = \underbrace{Z_{(s_o)}\hat{\beta}}_{\text{Regression}} + \underbrace{v'V^{-1}}_{\text{Kriging Weights}}\underbrace{(Z_{(s)} - X\hat{\beta})}_{\text{Residuals}}$$

The (semi)variogram defines a distance decay function to specify how important are nearby points for a prediction, and the overall spatial arrangement of the measured points.

UNIVERSAL, ORDINARY, AND SIMPLE KRIGING

Universal Kriging

A series of covariates are used as predictors

```
krige(log(zinc) ~ sqrt(dist),
      data = meuse,
      newdata = meuse.grid,
      model  = vt.fit)
```

Ordinary Kriging

An intercept only model
No Covariates are defined

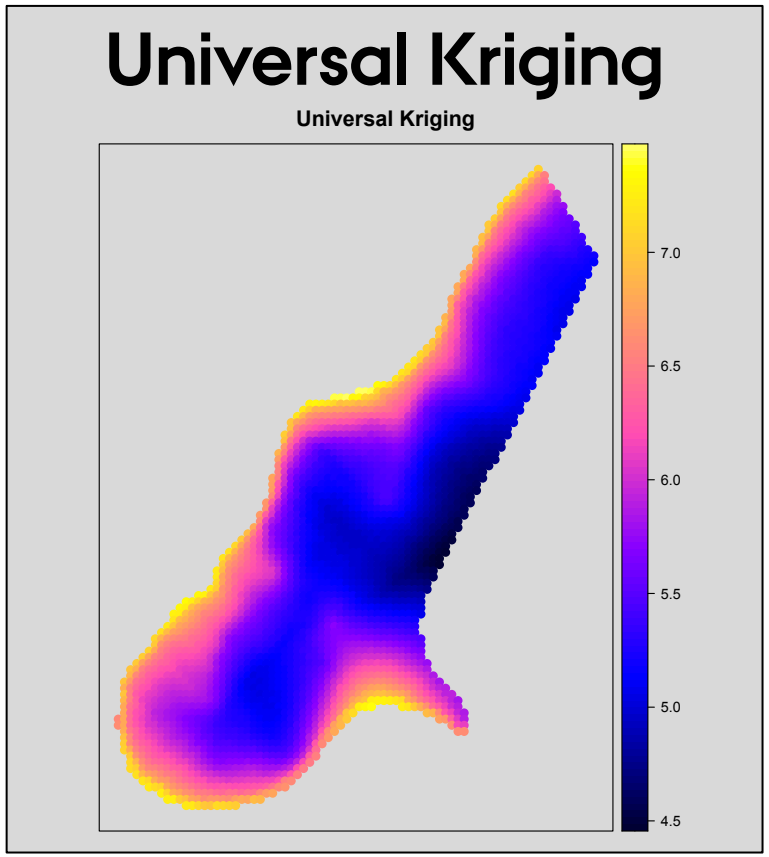
```
krige(log(zinc) ~ 1,
      data = meuse,
      newdata = meuse.grid,
      model  = vt.fit)
```

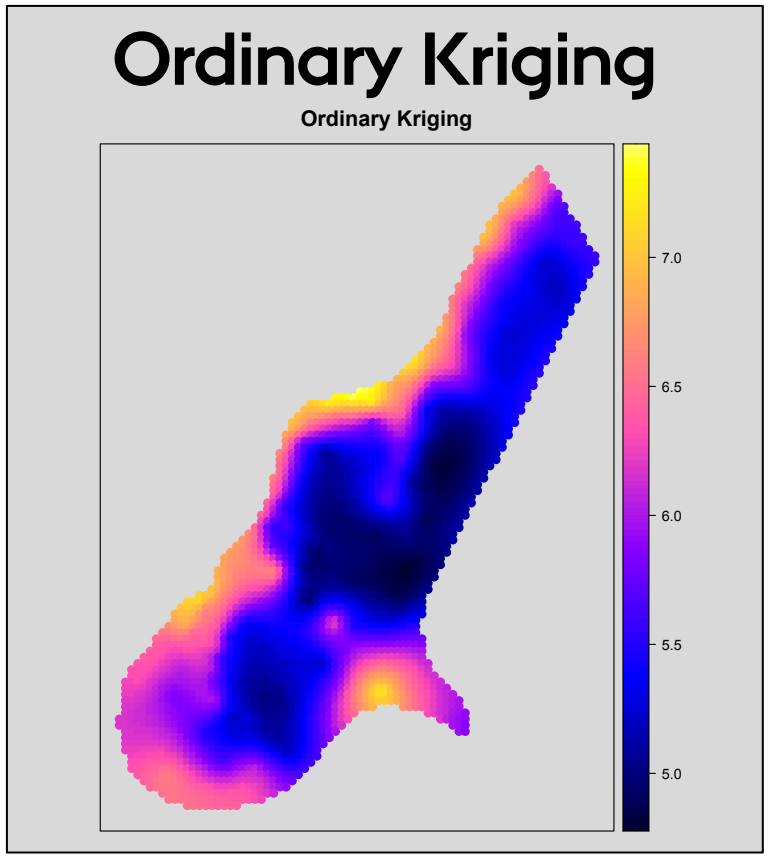
Simple Kriging

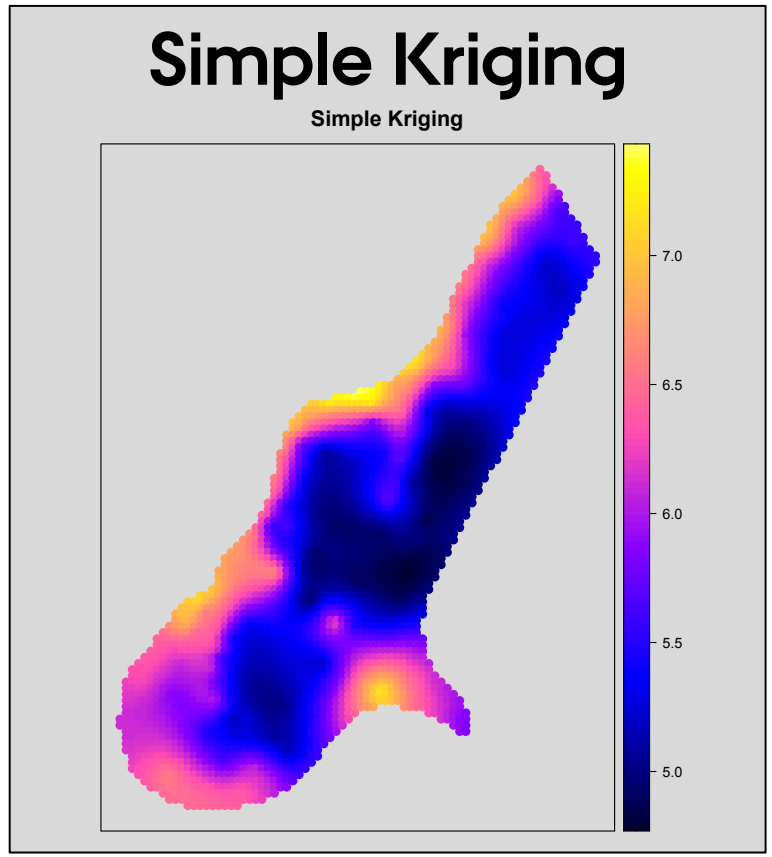
An intercept only model
The intercept is defined

```
krige(log(zinc) ~ 1,
      data = meuse,
      newdata = meuse.grid,
      model  = vt.fit,
      beta = 5.9)
```

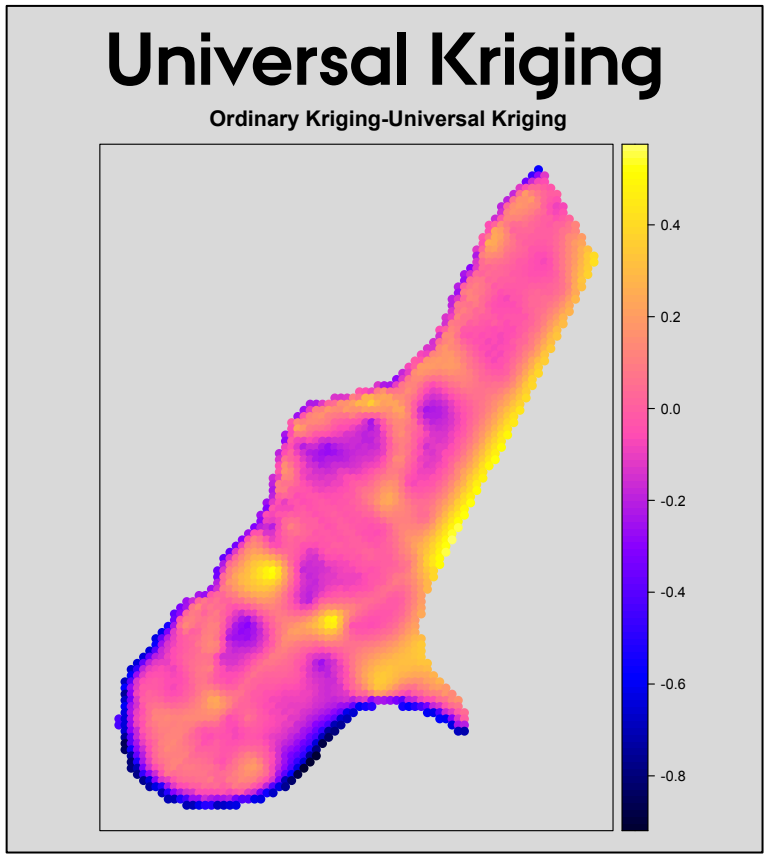
UNIVERSAL, ORDINARY, AND SIMPLE KRIGING

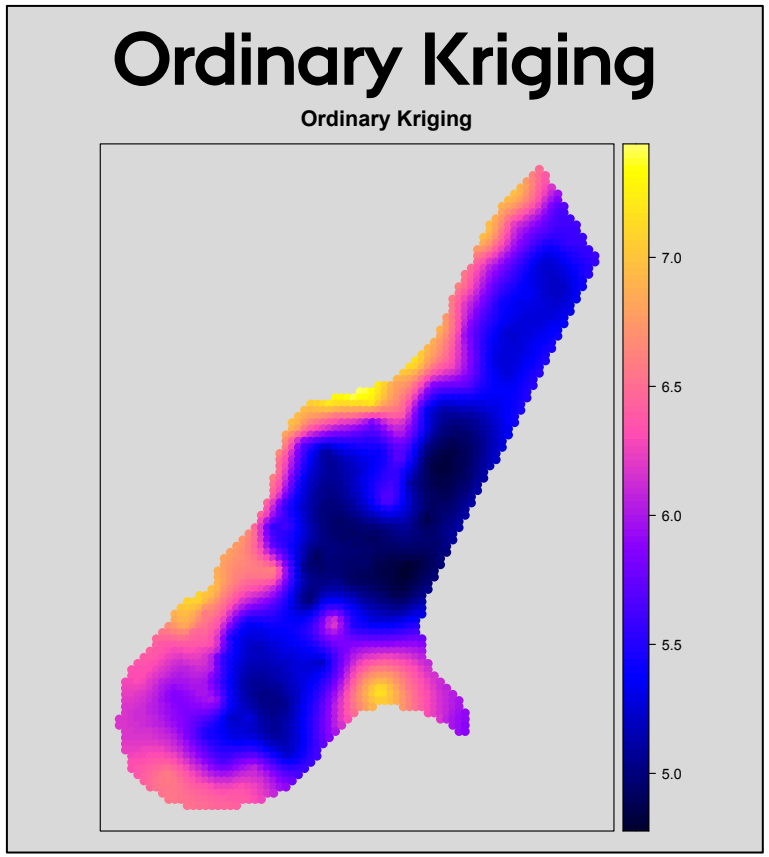


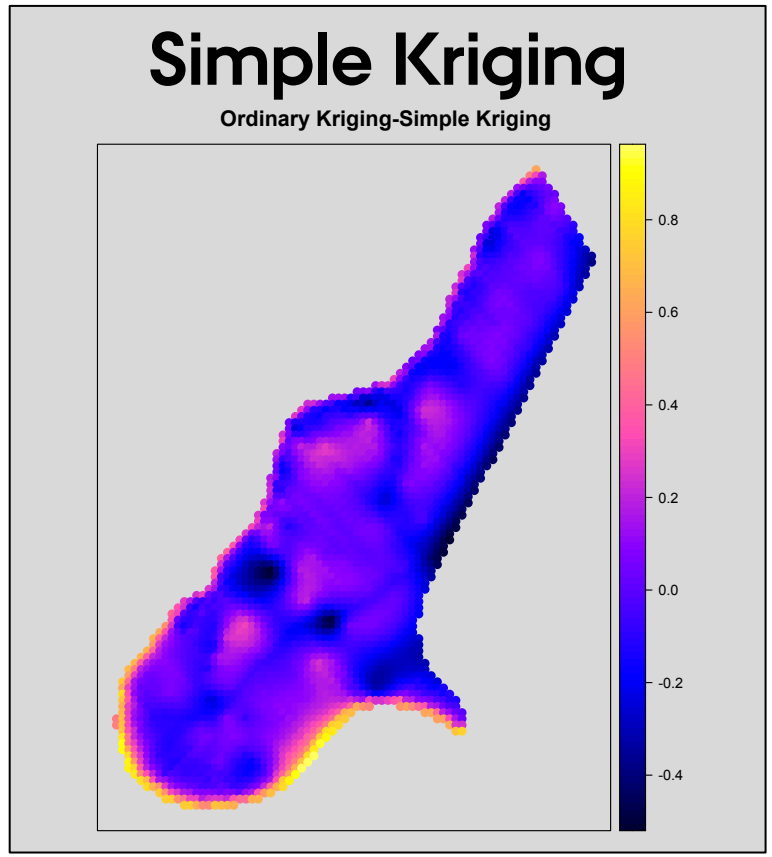




UNIVERSAL, ORDINARY, AND SIMPLE KRIGING DIFFERENCES







So far so good?

Any questions?

Ready to move on?

SPATIAL CORRELATION: THE VARIOGRAM

In geo-statistics the spatial correlation is modelled via the **variogram**

- NOT the **correlogram** → **correlogram** focus on a **ONE** dimensional space (One variable at a time)

To estimate the partial correlation (i.e., the **variogram**) from observational data we assume **stationarity** and **isotropy**.

Stationarity

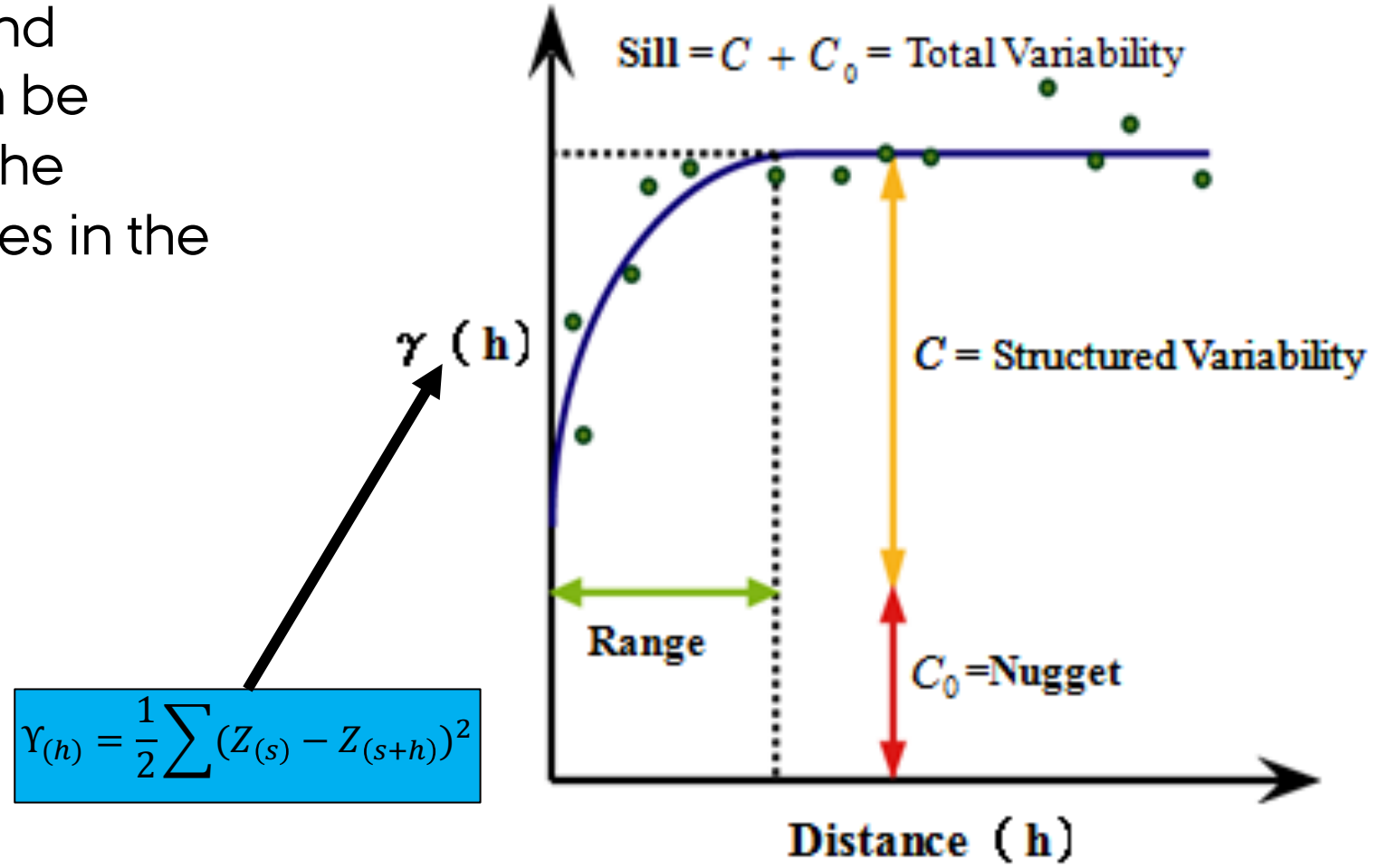
The relation between observations DOES NOT depend on their location but DOES depend on the distance between them.

Isotropy

The relation between observations DOES NOT depend on the direction between two points

SPATIAL CORRELATION: THE VARIOGRAM

By assuming **stationarity** and **isotropy** the variogram can be estimated by establishing the relation between differences in the observations and distance between observations.

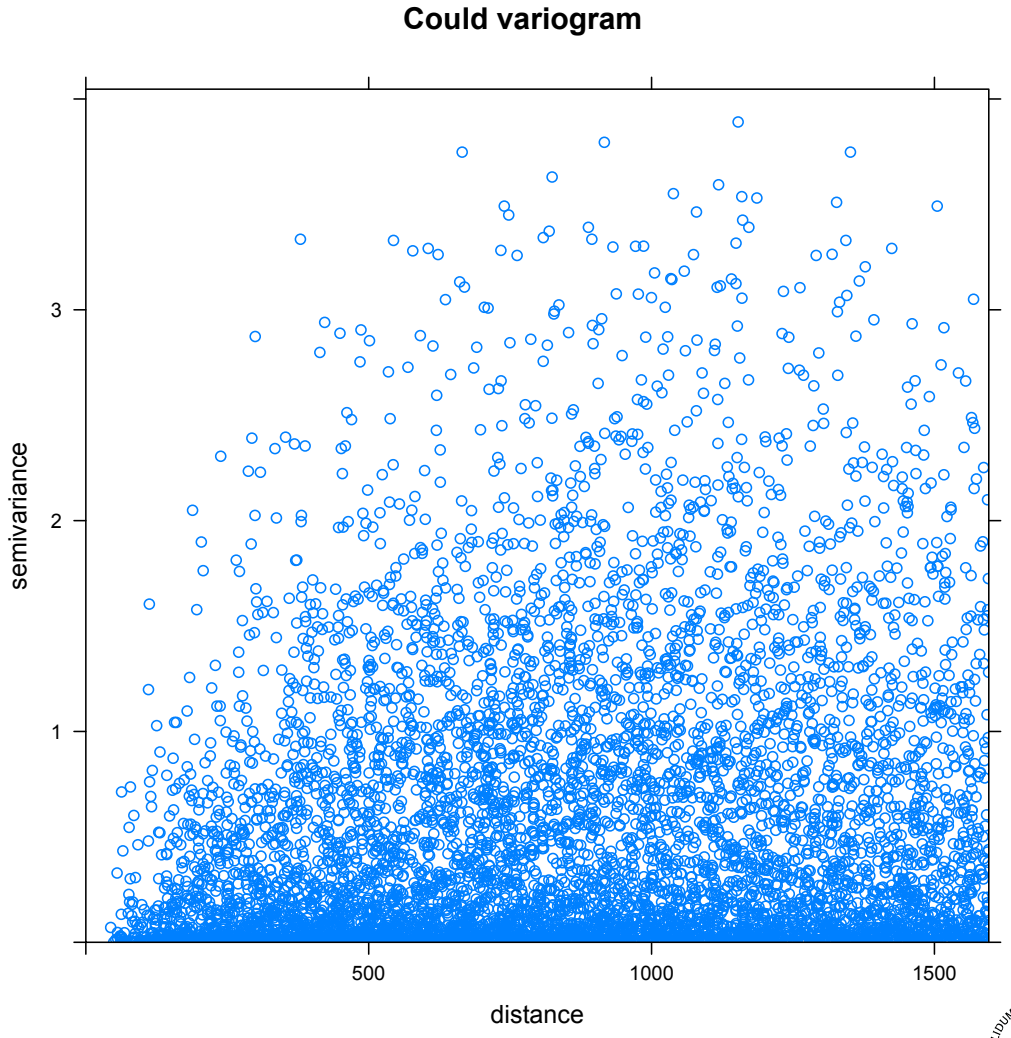


SPATIAL CORRELATION: THE VARIOGRAM

How are variograms put together?

- Plotting all possible squared differences of observation pairs $(Z_{(s_i)} - Z_{(s_j)})^2$ against their separation distance (h_{ij})
- This is wat is called a:

could variogram

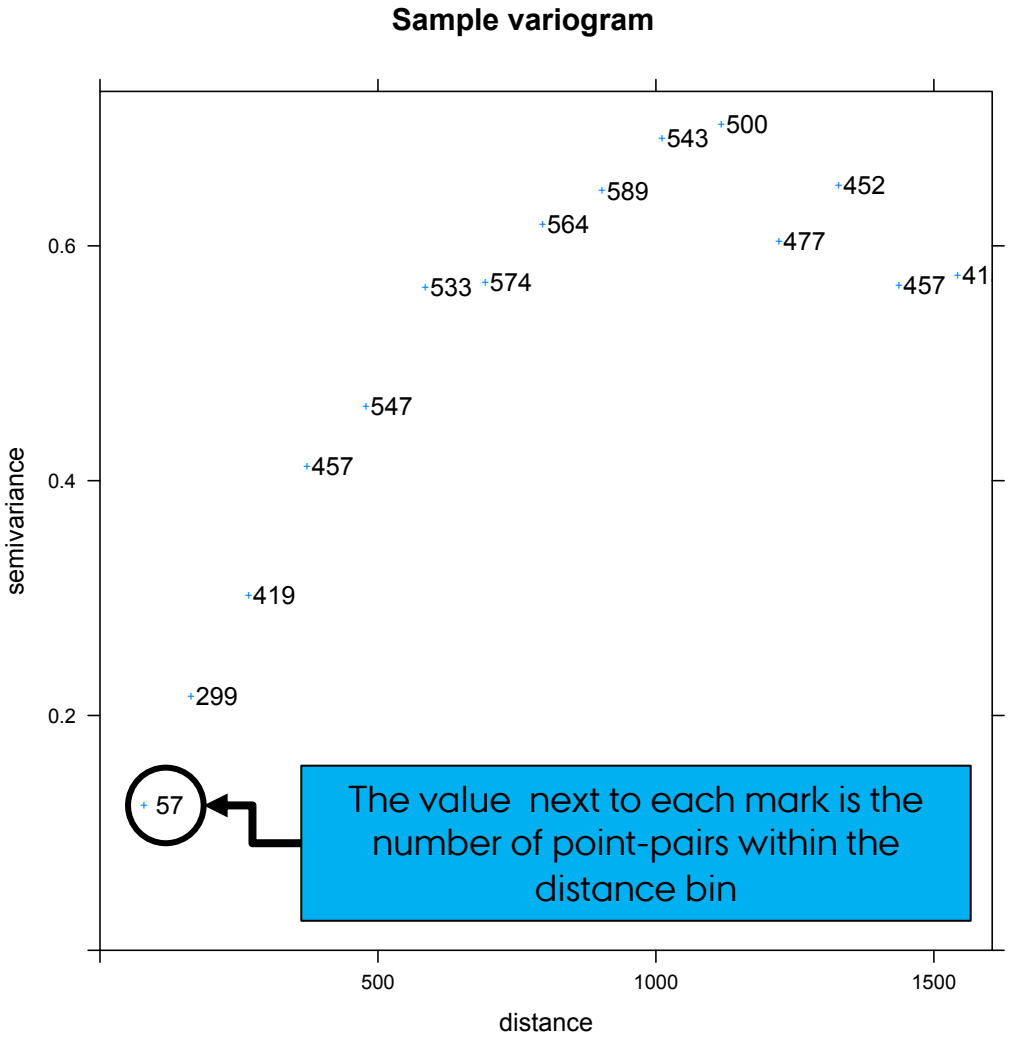


SPATIAL CORRELATION: THE VARIOGRAM

The semivaraince in the **variogram cloud** can be "averaged" over a given distance intervals (*h*) to generate the

sample variogram

Doing this "smooths" the variation in the could variogram and provides a signal of the "full" variogram



SPATIAL CORRELATION: THE VARIOGRAM

The default implementation of the **sample variogram** done by the `variogram` function uses by default:

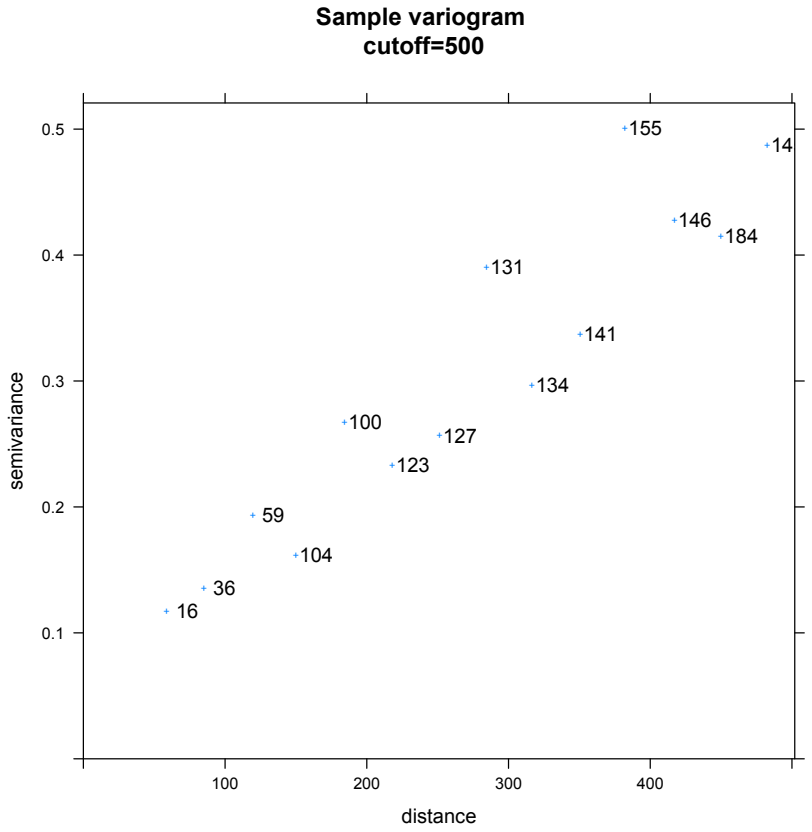
1st – A predefined cut-off distance:

- The maximum distance up to which point pairs are considered is one third of the largest diagonal of the bounding box.

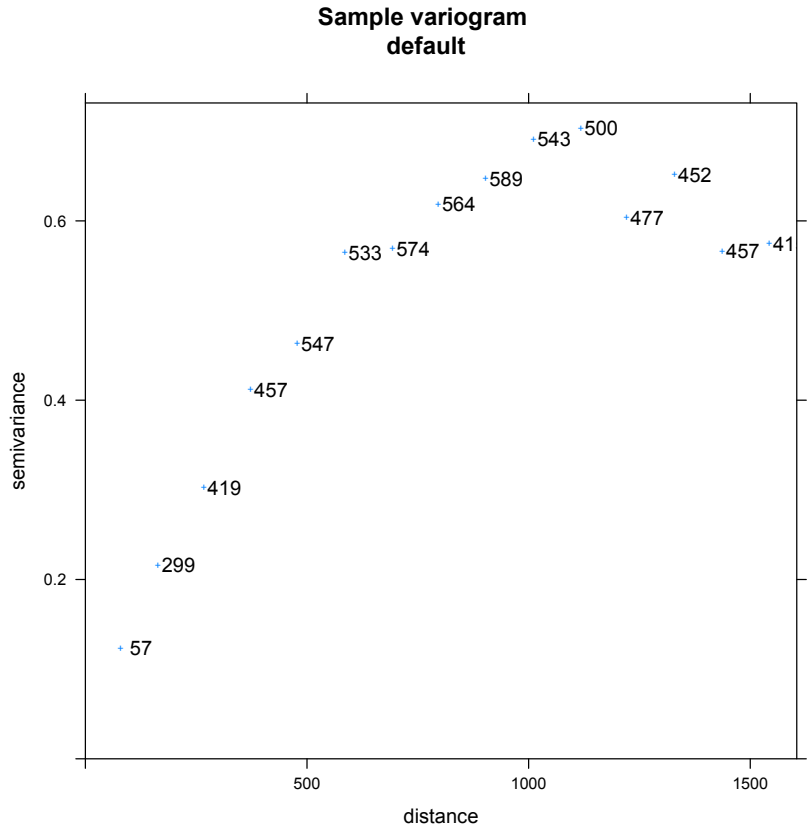
  Why?: autocorrelations are never computed for lags farther than half the series length → makes no sense to compute semi-variances for long distances

- *You would usually want to reduce the cut-off distance* – think local predictions.

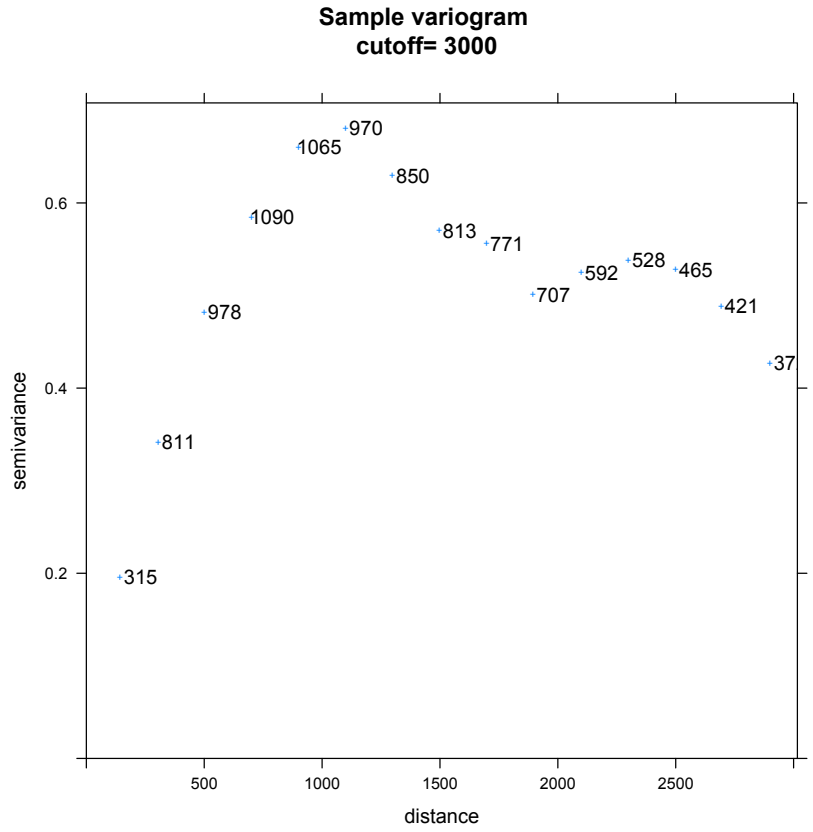
SPATIAL CORRELATION: THE VARIOGRAM



cutoff=500



default



cutoff=3000

SPATIAL CORRELATION: THE VARIOGRAM

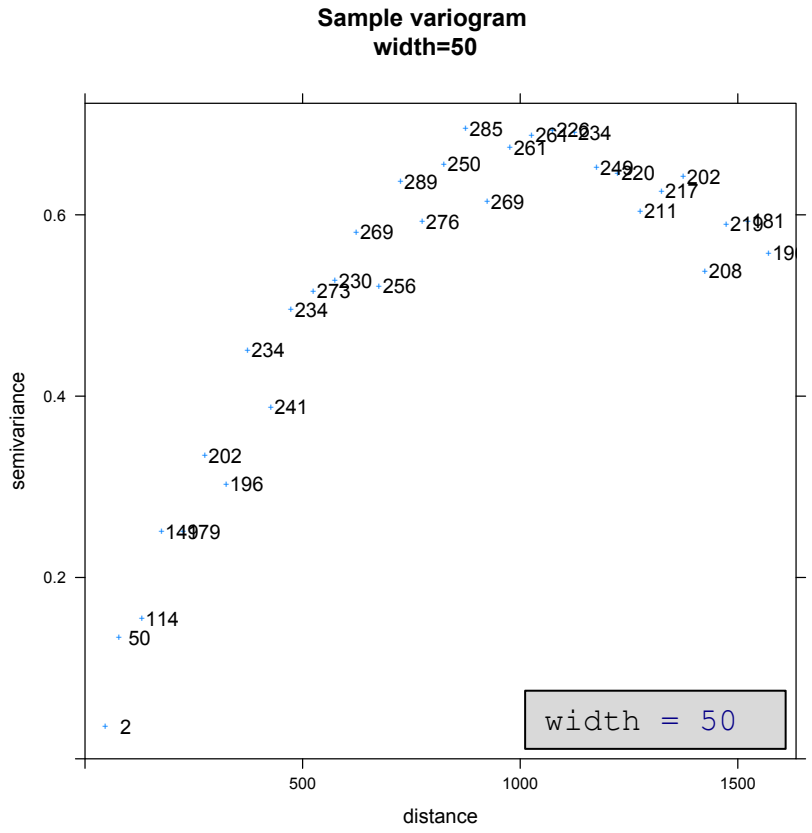
The default definition of the **sample variogram** done by the `variogram` function makes some decisions for you:

2nd – <u>The distance interval width:</u>

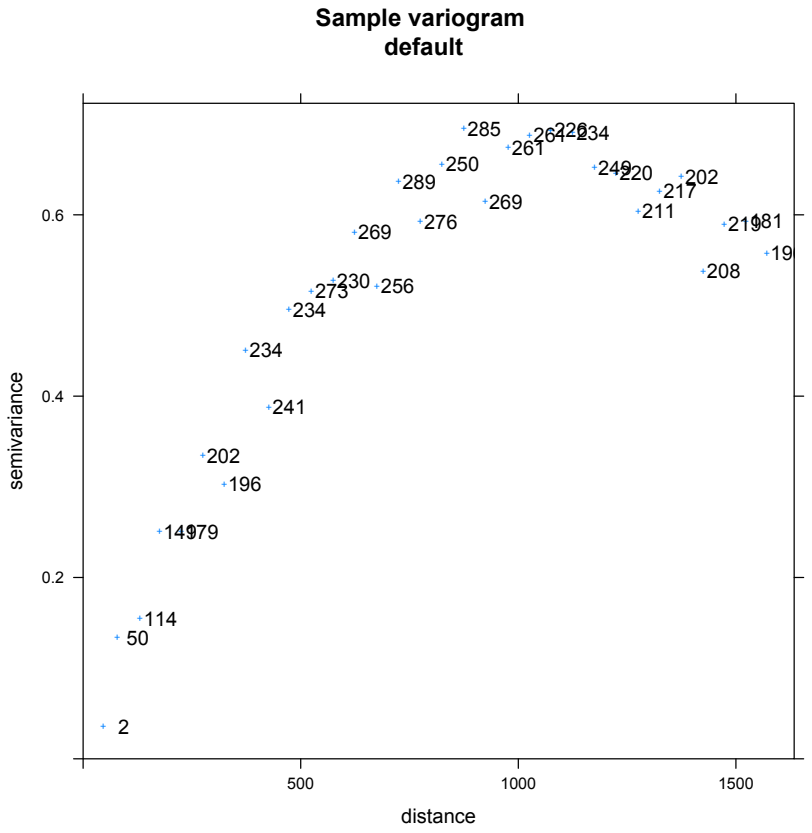
- The distance interval over which the semivariances are averaged is is the cut-off value divided by 15 (to generate 15 points)
- Choosing a smaller interval width will result in more detail

  → but can enhance noise.
- You can specify the intervals in two ways:
  1. Define the `width` argument: size of distance intervals
  2. Define the `boundaries` argument: each interval to be plotted

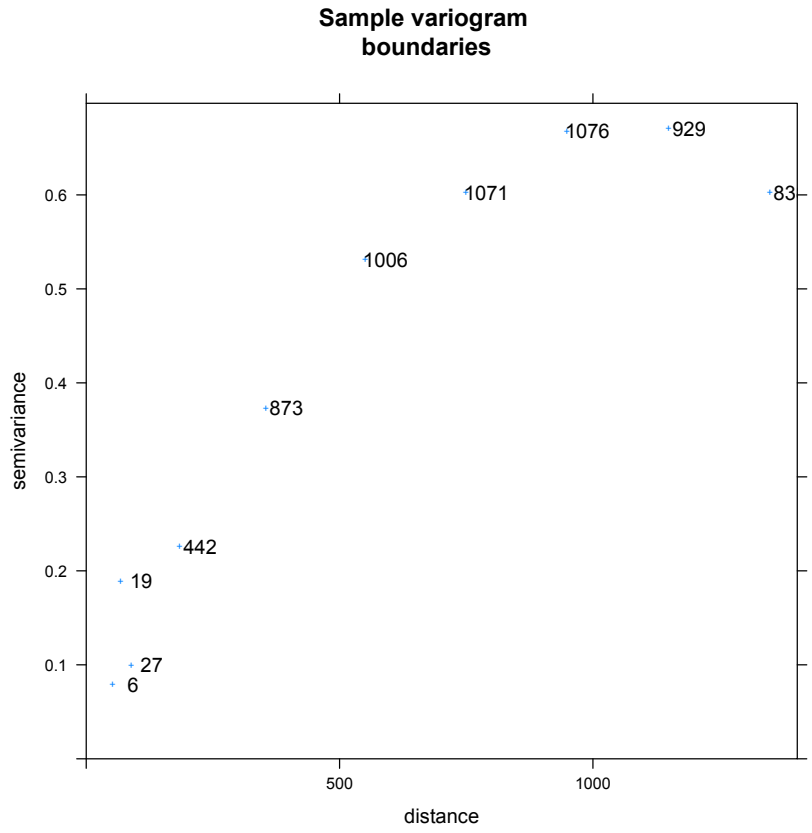
SPATIAL CORRELATION: THE VARIOGRAM



width



default



boundaries

```
boundaries = c(seq(0, 100, 20),
               seq(250, 1500, 200)
```

So far so good?

Any questions?

Ready to move on?

VARIOGRAM BASED MODELLING

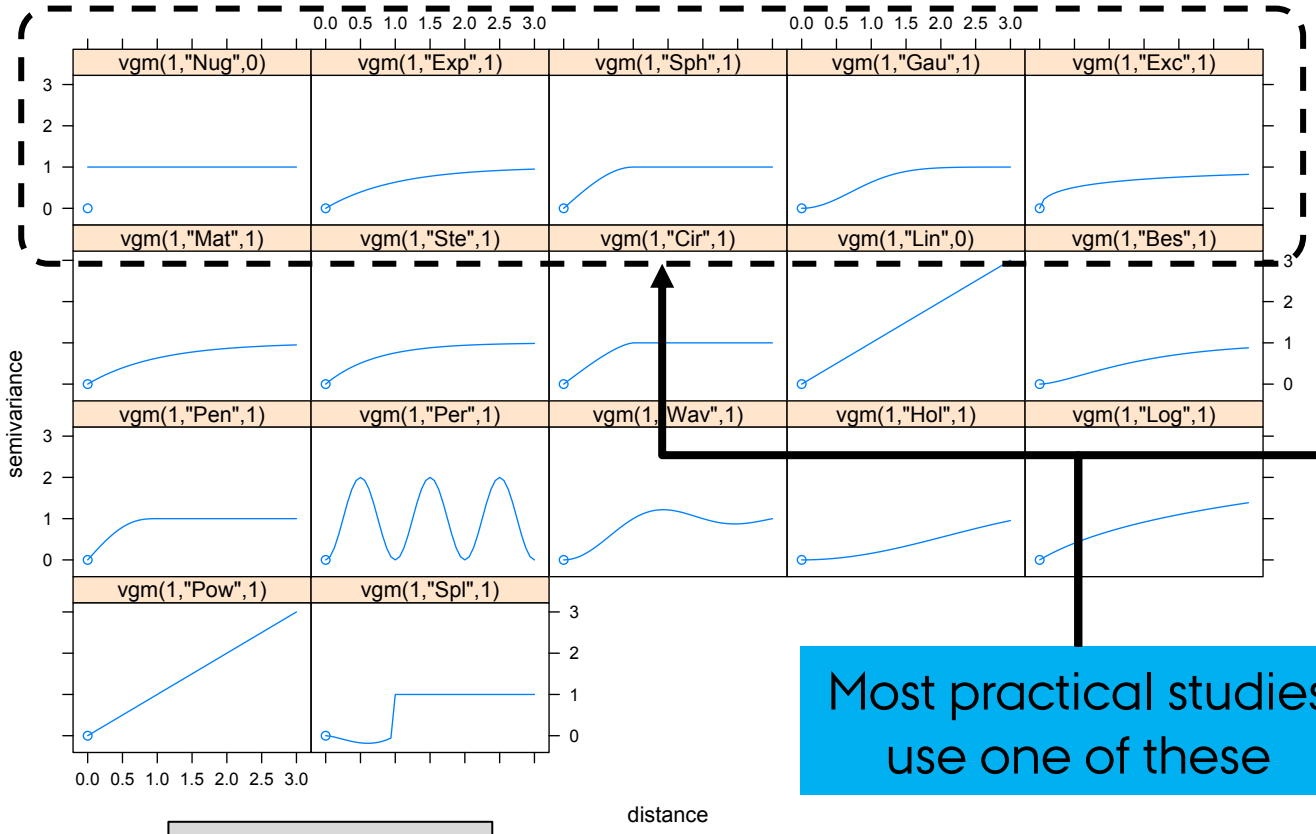
The variogram can be used to define a change in the "weight" of a known location as a function to its distance to the interpolated/predicted point.

- For this, the **could variogram** needs to be translated into a **parametric variogram model** using the data → *the sample variogram is **NOT** needed as an intermediate form BUT IS USEFULL to determine the best model.*
- The selected model is called → **experimental variogram**

How to select the best parametric variogram models:

The traditional way of finding a suitable variogram model is to fit a parametric model to the **variogram cloud** and assess its "fit".

BASIC VARIOGRAM MODELS



`show.vgms()`

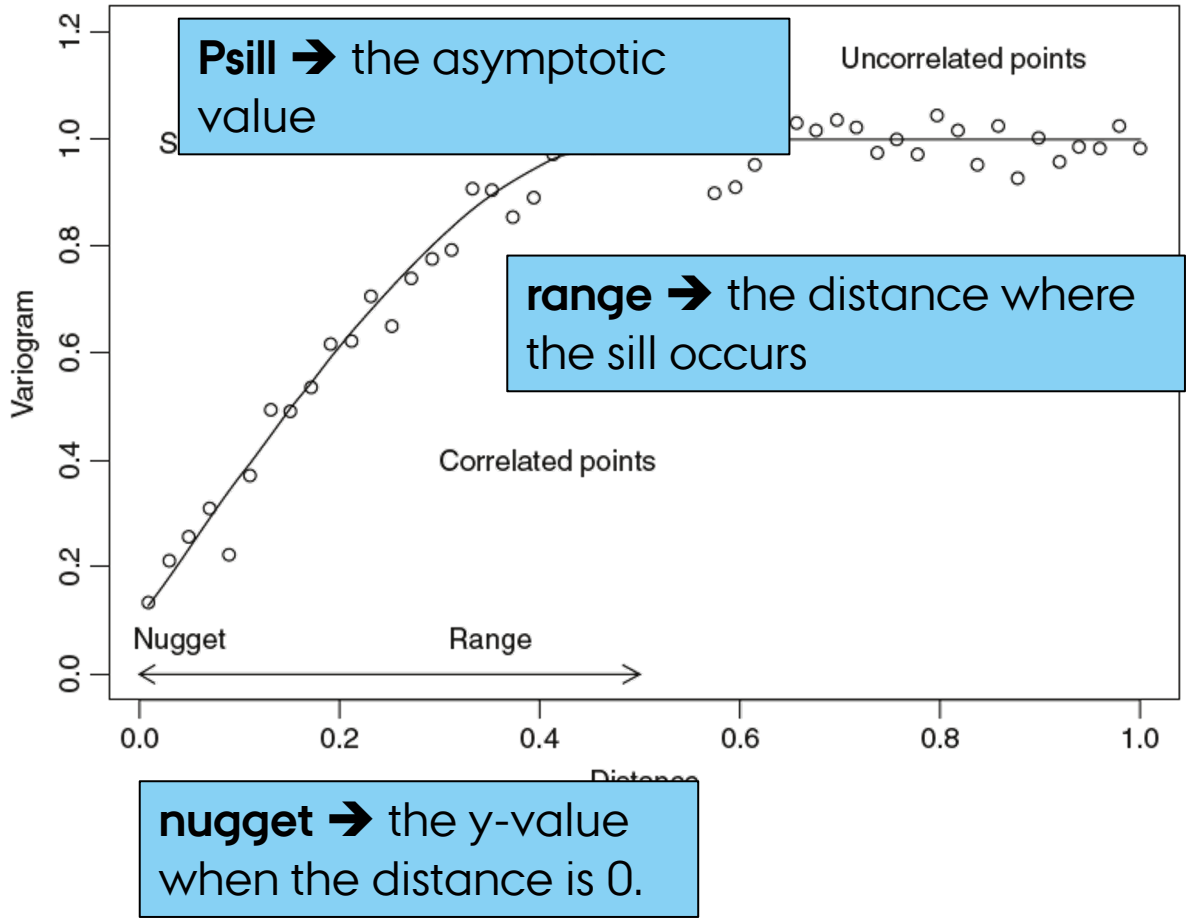
Most practical studies use one of these

	short	long
1	Nug	Nug (nugget)
2	Exp	Exp (exponential)
3	Sph	Sph (spherical)
4	Gau	Gau (gaussian)
5	Exc	Exclass (Exponential class/stable)
6	Mat	Mat (Matern)
7	Ste	Mat (Matern, M. Stein's parameterization)
8	Cir	Cir (circular)
9	Lin	Lin (linear)
10	Bes	Bes (bessel)
11	Pen	Pen (pentaspherical)
12	Per	Per (periodic)
13	Wav	Wav (wave)
14	Hol	Hol (hole)
15	Log	Log (logarithmic)
16	Pow	Pow (power)
17	Spl	Spl (spline)
18	Leg	Leg (Legendre)
19	Err	Err (Measurement error)
20	Int	Int (Intercept)

VARIOGRAM MODELLING

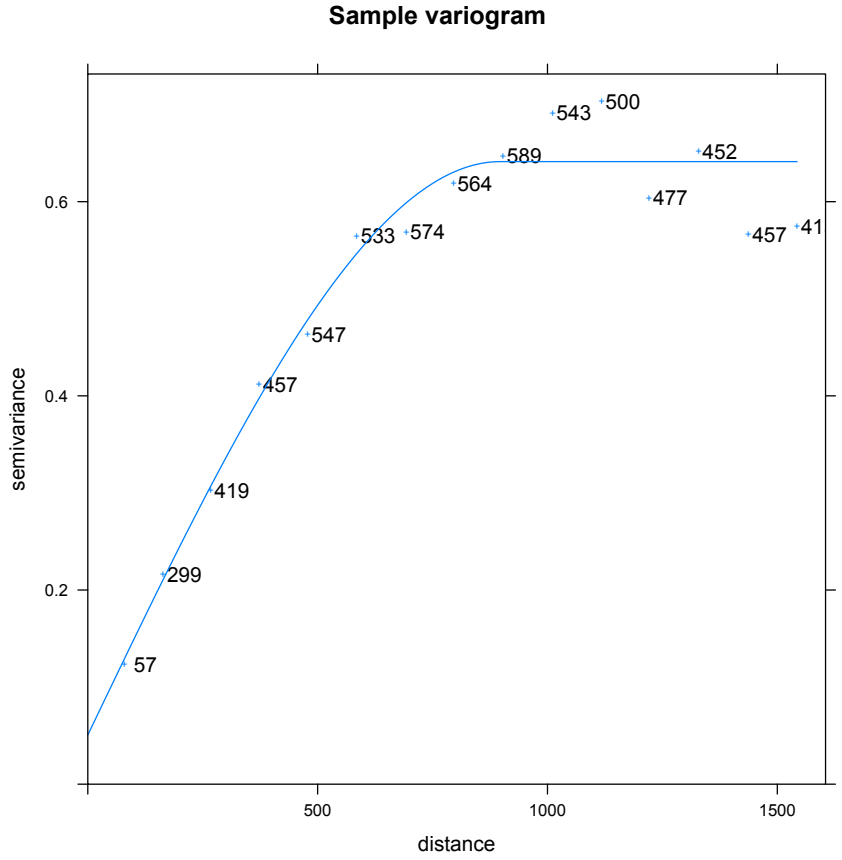
Variogram models are derived using the function `vgm` and defining four main arguments: `psil`, `model`, `range`, and `nugget`

```
vgm(psil = 1,
    model = "Sph",
    range = 300,
    nugget = 0.5)
  model psill range
1   Nug   0.5     0
2   Sph   1.0   300
```



VARIOGRAM MODELLING

Once you have defined a variogram model and its "initial" attributes, it can be fitted to a sample variogram to define the **parametric variogram model** for the data at hand.



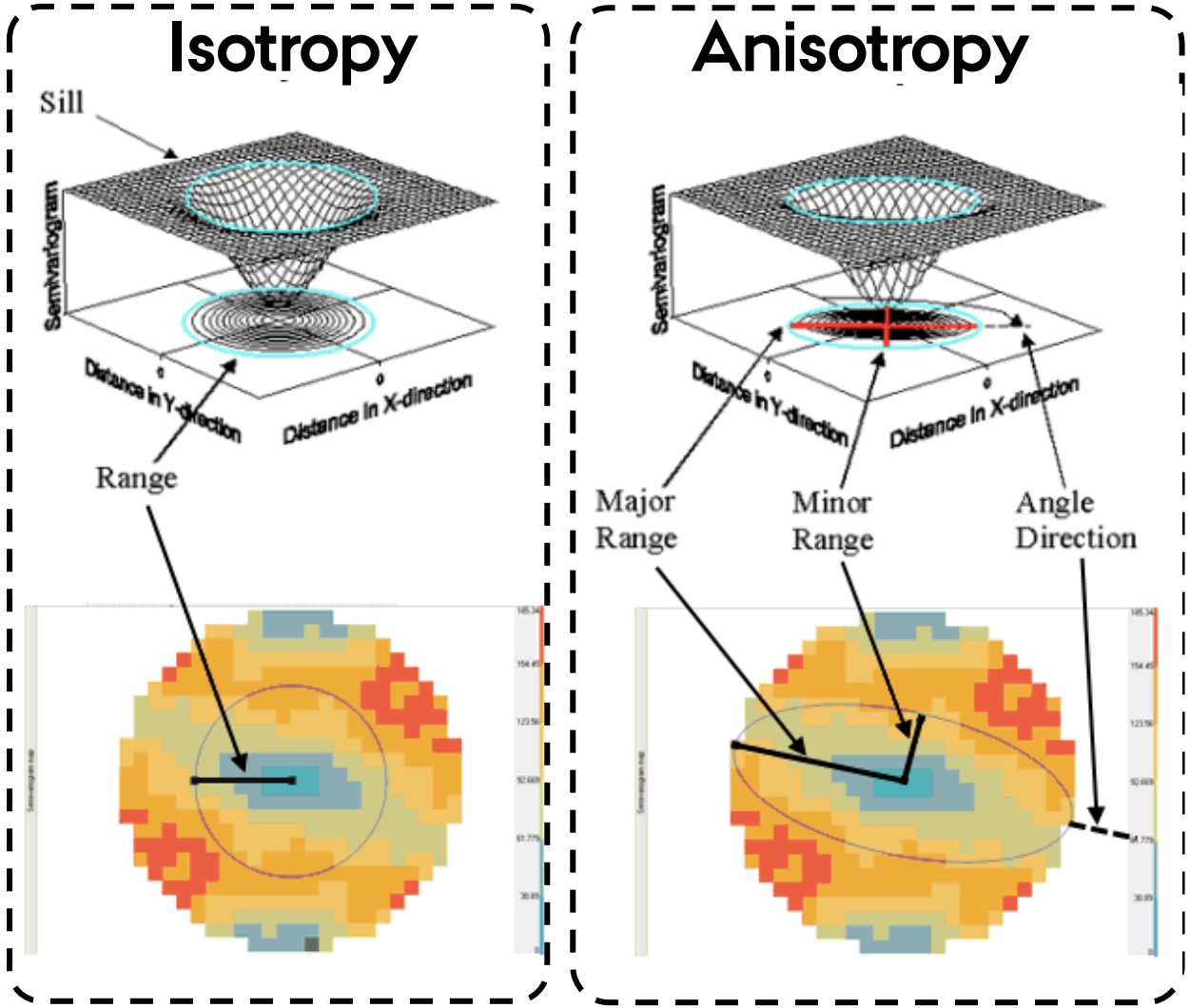

```
v <- variogram(log(zinc) ~ 1, meuse)
v.fit <-fit.variogram(v,
        vgm(1, "Sph", 800, 1))
v.fit
  model      psill    range
1   Nug 0.05065923   0.0000
2   Sph 0.59060463 896.9976
plot(v,v.fit)
```

Note that the variogram attributes that `fit.variogram` generates are different from the original ones.

ANISOTROPY

The `vgm` function default assumption is **isotropy**

The sill is reach at the same time in all directions

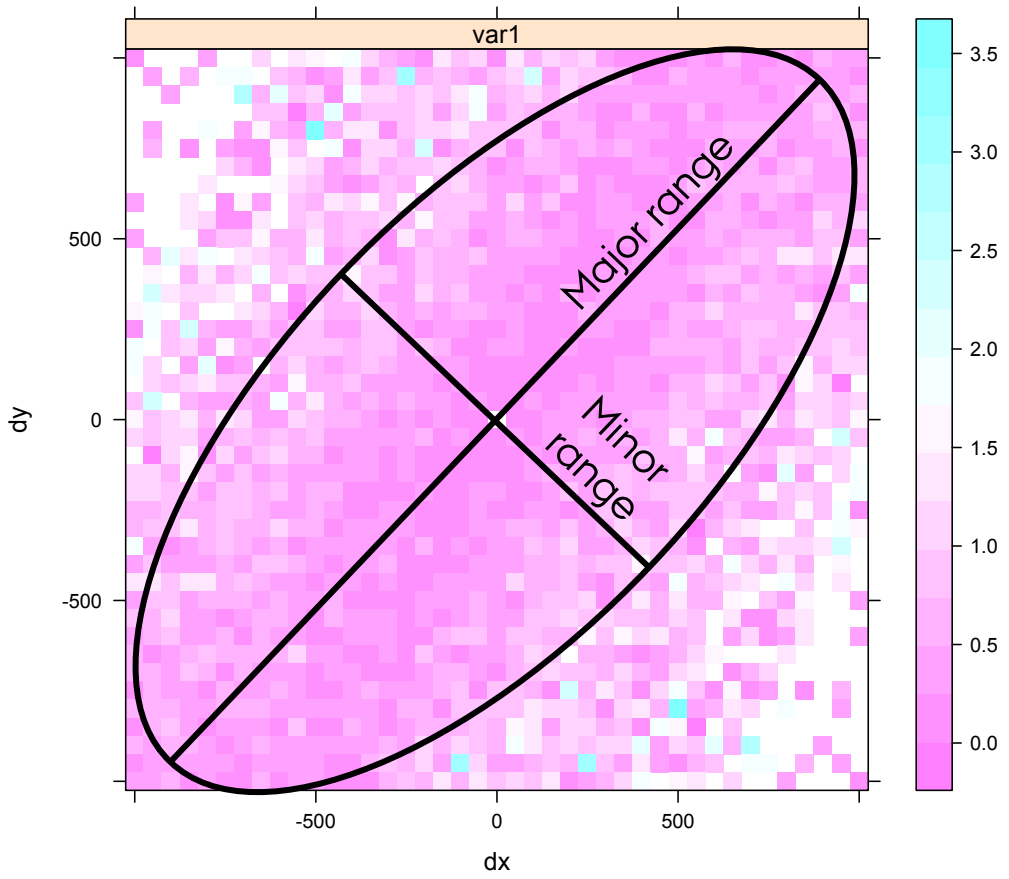


However the variability can change with direction This anisotropy

The sill is reached more rapidly in one direction.

ANISOTROPY

Variogram map



Based on this map is clear that the variance is not isotropic

Max variance is reaches in one but not another direction

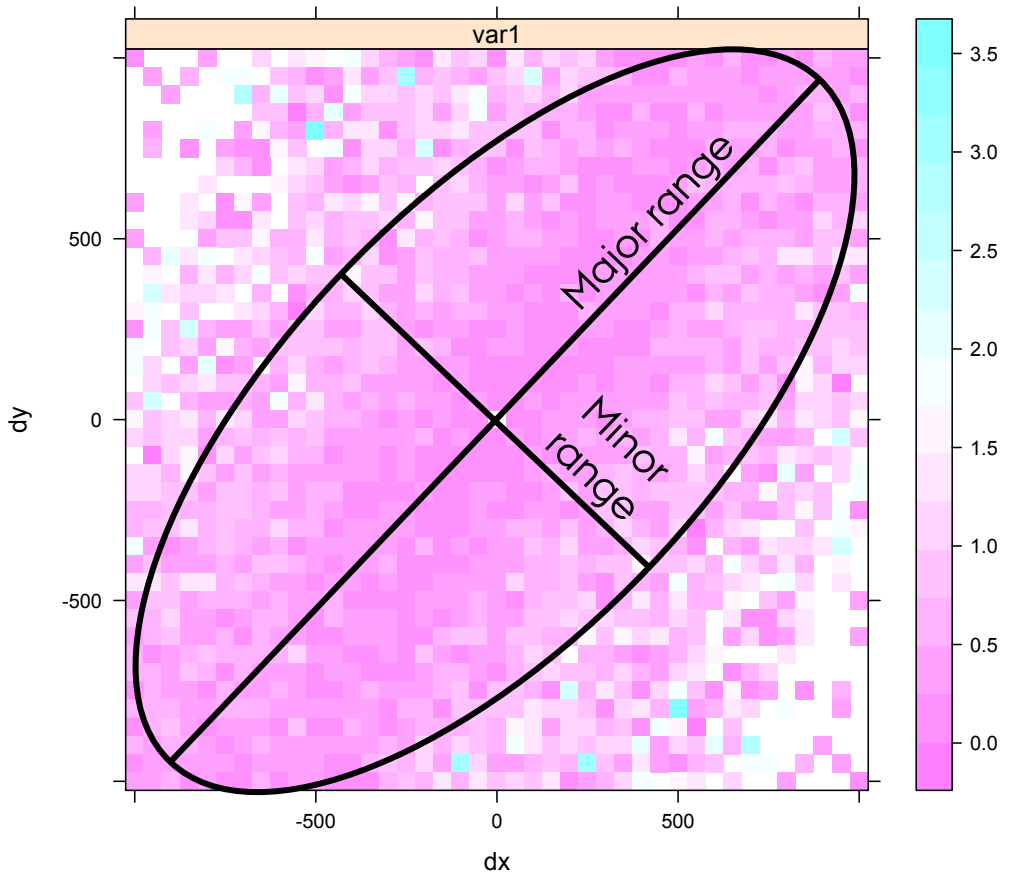
The two factors needed to model the anisotropy are:

1) Angle for the principal direction
2) Ratio of the minor range to the major range

```
plot(variogram(log(zinc) ~ 1,
        data = meuse,
        map = TRUE, ## plot the variance
between points as a function of the
distance in the x and y direction
        cutoff = 1000, width = 50))
```

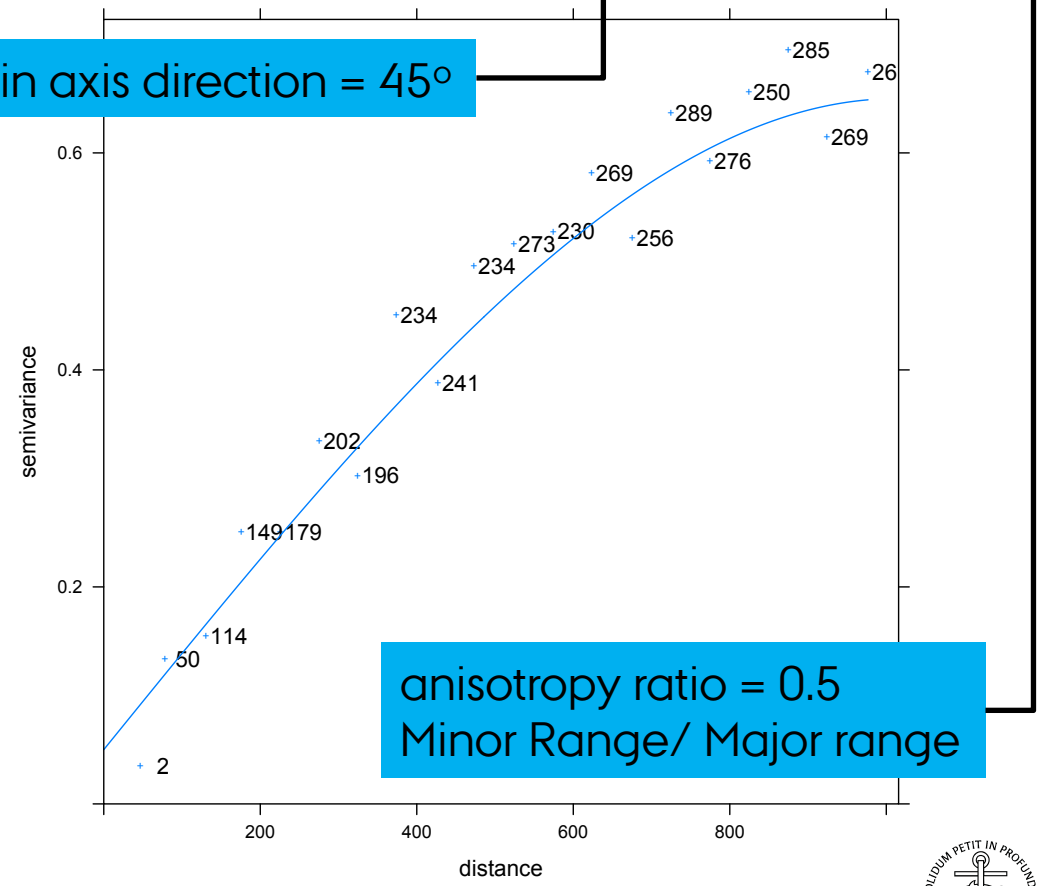
ANISOTROPY

Variogram map




```
v.dir <- variogram(log(zinc) ~ 1, data=meuse,
                   cutoff = 1000, width = 50)
v.anis <- vgm(0.6, "Sph", 1600, 0.05,
# this argument define the "anisotropy" in
the variogram
                anis = c( 45 , 0.5 ))
plot(v.dir, v.anis)
```

main axis direction = 45°



anisotropy ratio = 0.5
Minor Range/ Major range

So far so good?

Any questions?

Ready to move on?

SOME LAST POINTS ON KRIGING

Non-linear Transforms of the Response Variable

- You can transform your response and build a model with it.
- **But…**
  - When the “response” is transformed… the inverse transformation of your kriging predictions is not the expected value of the Original Variable.
  - The same applies to Non-Normal responses (Pres/Abs, Probabilities, counts).

SOME LAST POINTS ON KRIGING

Kriging cannot deal with Singular Matrix Errors – This means a perfect correlation between observations, which are NOT allowed:

These perfect correlations occur when:

1. Observations share the same location:
   - How to find such duplicates – use the `zerodist` function on a `SpatialPoint(DataFrame)` object → **Only keep one!**

2. Variogram models that result in perfect correlations
   - `vgm(0, "Nug", 0)` or `vgm(1, "Gau", 1e20)`

3. Models with perfectly correlated variables.
   - A typical cases are a constant predictors or inverse predictors

So far so good?

Any questions?

Ready to move on?

MODEL DIAGNOSTICS - BASIC

Here the goal is to assess how good is my interpolation, and you can use many ways to assess this:

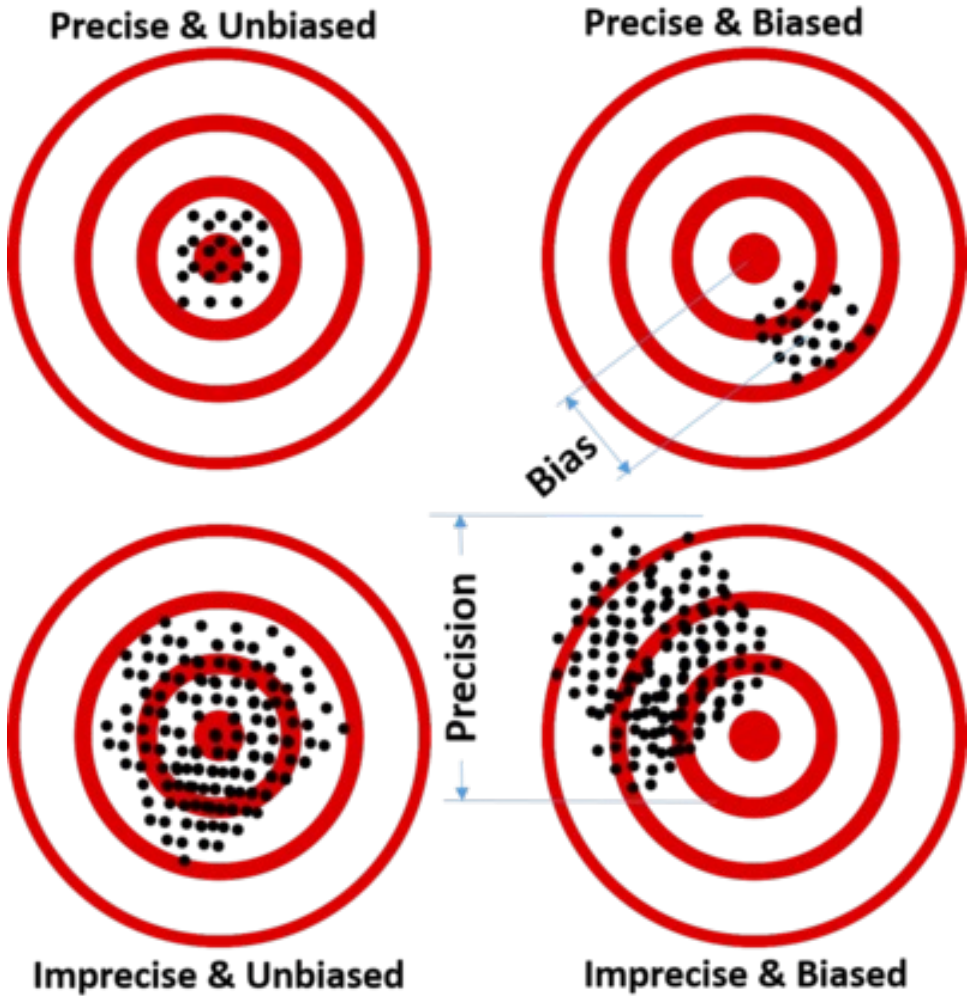
1. Correlation between predicted and observed → just as a pseudo-R^2
   - You could also think about the moments of the "residuals"
2. Assess the **precision** via the root-mean-square error (RMES)

$$RMES = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \hat{x}_1)^2}{N}}$$

3. Assess the **bias** via the mean/median error (ME)

$$ME = \frac{\sum_{i=1}^{n} x_i - \hat{x}_1{}^2}{N}$$

PRECISION AND BIAS



BIAS: expected value of the results differs from the true underlying quantitative parameter being estimated.

PRECISION: variability of model prediction for a given data point or a value which tells us the spread of our data

MODEL DIAGNOSTICS
CROSS-VALIDATION

Cross-validation provide a way to assess how well the model predicts the values at unknown locations.

The logic behind Cross-validation is simple:

Remove one or more data locations to define a training data set used to build a model

Use the created model to predict the data at the rest of the locations (test data).

CROSS-VALIDATION STRATEGIES

There are two ways to think about cross-validation:

- **Leave one out strategy**: One one location is removed at a time:
  - The training dataset used to build this model has N-1 observations.
  - The procedure can be done with all or a given percentage of the observations.
- **Test/train strategy**: A percent of the observations are set apart to build the model (training dataset), and the fit using the remaining information (test dataset)

The accuracy is then assess using either bias or precision metrics on the “testing” dataset/observation

CROSS-VALIDATION STRATEGIES IN R

The `gstat` package has two functions to automate the cross-validation process:

- `krige.cv` function which is used for kriging; and implemented just as the krige function [that is formula, locations, data, newdata, model need to be defined]
- `gstat.cv` function which is used for cokriging; and implemented just by passing the `gstat` object with the linear model of co-regionalisation

In both functions an extra argument (`nfold`) should be passed, to define the training/test dataset sizes.

If `nfold` is not specified, the dataset number of rows it assumes a leave one strategy.

So far so good?

Any questions?

Ready to finish?

IN SUMMARY…

Spatial interpolation is the process of using points with known values to estimate values at other unknown points.

Determining the values in these unknown points can be done:

- Using Non-geostatistical Interpolation methods such:
  - **Inverse distance weighting or Bilinear/Cubic interpolation**: (Based on distances)
  - **Linear regressions or splines**: Using locations and/or widely sampled variables
- Using Geostatistical Interpolation methods such:
  - **Kriging**: Univariate approach
  - **Co-kriging**: Multivariate approach

The semi variogram defines the spatial model that makes these methods Geostatistical

AARHUS
UNIVERSITY