



# Welcome to FES 781b

## Introduction to Spatial Statistics

### Syllabus Overview

- On CANVAS as web page. Updated periodically.

### Software and Books

- See Syllabus for recommended books. Both main books are electronically available for free. Several other inexpensive texts are also excellent aids.
- Optional intro to R **Times TBA shortly. You will absolutely learn (and need to learn) R in this class.**

### Projects

- Need a dataset – soon!
- Work in a group if you like – you may learn more with someone to work with!

## **DATA**

- If you have data that interests you, let us know and we'll try to use it in class.

## **Your Questions . . .**



## What kind of data will this class help me analyze?

Here are some projects from students in the last few years :

- Location of UBER pickups over a one month period in the Bay Area
- Location of 2300 incidents (enemy action, detainee operation, friendly action) in Afghanistan from 2007 to 2009. Includes number of people killed or injured.
- Location of U.S. negative tweets about Russian during 2014 Winter Games
- Location of murders in Chicago over a two year period
- Cougar sightings in Maine
- Location of US airports and crashes
- Location of babesiosis and Lyme disease in CT
- Location of firing neurons in a rat brain



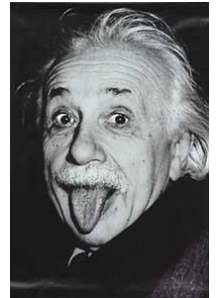
# What are Spatial Statistics?

*(Also known as geostatistics, spatial epidemiology, medical geography, spatial patterns, etc.)*

In order to discuss Spatial statistics we first have to discuss

## Spatial Data

Spatial Data are **proximate observations** measured in **Space** or **Time** (or both!) *(space and time are related in spatial analysis and in other fields . . .)*



- In **Space**, observations are usually in **1 to 3 dimensions** (two being the most common). However, the methods we discuss can be applied to higher dimensional spaces *(although plots are challenging!)* Higher dimensional point process data is used in analysis of electron microscope data, for example.



- **Time** observations are basically a one-dimensional equivalent to Space measurements
- Several recent books on the combination of space and time (<http://www.wiley.com/WileyCDA/WileyTitle/productCd-EHEP002348.html>)

Both kinds of spatial data are assumed to follow the

**First Law of Geography (Time)** : Everything is related to everything else, but near things are more related than far things" (*Tobler, 1970*)

Statistically, this is called (positive)

## SPATIAL AUTOCORRELATION

*Basically, pairs of observations in near space/time are more similar than those that are further apart.*

**NOTE** : this means that spatial data violates one the key pillars assumed in most statistical inference :

**Independence of Observations!**



**SO : Spatial statistics consists of statistical methods designed to describe / test / model / predict relationships among spatially patterned data.**

The spatial trends that result may be due to one of two processes (or both) :

1. If observations are actually independent, then observed patterns are the result of an **underlying spatial gradient** or trend – this ‘means’ we’re talking about the **MEAN!**

*This trend may be linear, quadratic, erratic, or otherwise*

2. If observations are NOT independent, we may observe
  - **Clustering of observations** (sometimes called patches) or
  - **Correlation of proximate measurements** (*i.e. pollution levels, soil moisture, etc*)

*Of course the opposite is also possible – observations pushing each other apart, or observations that are negatively correlated . . .*

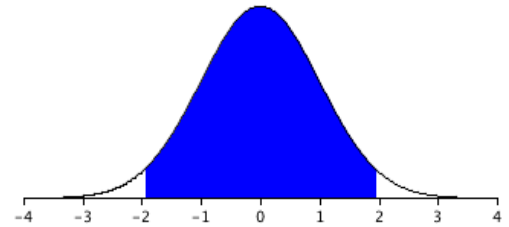
*So why all the fuss about independence?!?*



- Autocorrelation between spatial observations leads to an effective **reduction in sample size**.
- Magnitude of the autocorrelation is affected by the **sampling unit**
- Information about autocorrelation can **aid prediction**

## Example : Your Basic Confidence Interval

Recall the :



## Central Limit Theorem

If  $X_1, X_2, \dots, X_n$  are a sample of  $n$  independent and identically distributed trials from **any** distribution with mean  $\mu$  and standard deviation  $\sigma$ , then for  $n$  large enough,

$$\bar{X}_n \sim N\left(\mu, \frac{\sigma}{\sqrt{n}}\right)$$

Suppose that the  $X$ 's happen to have a normal distribution with

- mean  $\mu$  **unknown**
- variance  $\sigma^2$  **known**

*(say the  $X$ 's are soil pH samples from adjacent plots)*

The CLT tells us that a 95% Confidence interval for the mean

$\mu$  is 
$$\bar{X}_n \pm 1.96 \frac{\sigma}{\sqrt{n}}$$

This is because

$$\text{Var}(\bar{X}_n) = \frac{\sigma^2}{n}$$

A little aside : Recall that if  $X_1$  and  $X_2$  are independent then  
$$\text{Var}(X_1 + X_2) = \text{Var}(X_1) + \text{Var}(X_2)$$

**HOWEVER** : if  $X_1$  and  $X_2$  are **NOT INDEPENDENT** then

$$\text{Var}(X_1 + X_2) = \text{Var}(X_1) + \text{Var}(X_2) + 2\text{Cov}(X_1, X_2)$$

What is the **Covariance**?!?

The Covariance is just unscaled Correlation :

$$\text{Cov}(X_1, X_2) = \text{Correlation}(X_1, X_2)\sigma_{X_1}\sigma_{X_2}$$

so if  $X_1$  and  $X_2$  are independent then  $\text{Cov}(X_1, X_2) = 0$

Incidentally, 
$$\text{Cov}(X_1, X_1) = \text{Var}(X_1)$$

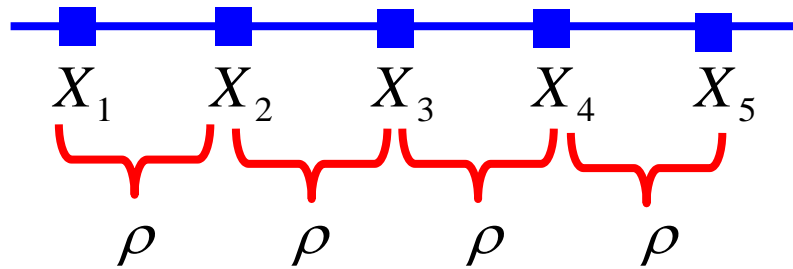
*Back to our Confidence Interval . . . . .*

**NOW** : Suppose that the  $X$ 's are a set of observations in space (time) such that they are **NOT INDEPENDENT**. In particular, we'll say that

$$\text{Cov}(X_i, X_j) = \sigma^2 \rho^{|i-j|} \quad \text{where } i, j = 1, 2, \dots, n$$

*Basically, this is saying that all the  $X_i$  have variance  $\sigma^2$  neighboring observations (distance 1) have a correlation of  $\rho$  and observations further away (distance 2, 3, 4, etc.) have correlation  $\rho^2, \rho^3, \text{ etc.}$  :*





Think about our confidence interval : if you like algebra, you can show that

$$\begin{aligned} \text{Var}(\bar{X}_n) &= \frac{1}{n} \left\{ \sum_{i=1}^n \sum_{j=1}^n \text{Cov}(X_i, X_j) \right\} \\ &= \frac{\sigma^2}{n} \left[ 1 + 2 \left( \frac{\rho}{1-\rho} \right) \left( 1 - \frac{1}{n} \right) - 2 \left( \frac{\rho}{1-\rho} \right)^2 \left( \frac{1-\rho^{n-1}}{n} \right) \right] \end{aligned}$$

Usually,  $\rho=0$  and this is all zero!!!

# Who Cares!?!?!

*Let's add numbers : suppose you measure the pH of soil in 10 adjacent plots on a line, each one unit apart, and find that  $\bar{X}_{10} = 6$ .*



*You happen to know that  $\sigma = 1$ .*

*The CLT states that  $Var(\bar{X}_{10}) = \frac{1}{10}$  and we would make the following 95% Confidence Interval :*

$$6 \pm 1.96 \frac{1}{\sqrt{10}}$$

*HOWEVER* : suppose that plots are spatially **correlated**, say  $\rho = 0.26$ . In this case, our algebra shows that

$$\text{Var}(\bar{X}_{10}) = \frac{1}{10} (1.6)$$

And our confidence interval **REALLY** should be

$$6 \pm 1.96 \frac{1}{\sqrt{10}} * \sqrt{1.6}$$

*Another way of thinking about this. In this example, taking 10 **dependent** observations is equivalent to taking*

$$10 / 1.6 = 6.25 \text{ **independent** observations!}$$



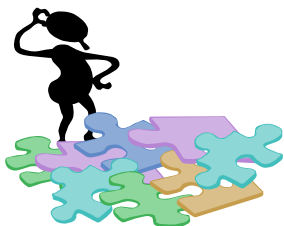
**Note** : while space and time are basically equivalent in spatial analyses, time series analysis has developed as its own area of study. This isn't that class and we'll focus on **Space**.

# Types of Spatial Data

Cressie (1993) identifies three main branches of spatial statistics :

- **Spatial Point Processes**
- **Geostatistics (spatially continuous data)**
- **Discrete Spatial Variation (lattice data)**

However, first, a bit more about how we measure and quantify spatial data . . .

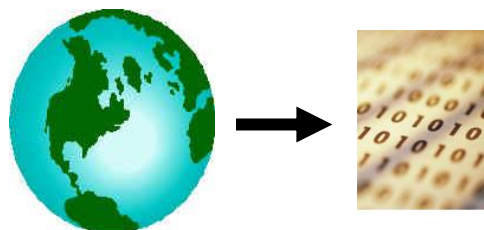


## More on Spatial Data

Lots of divisions/names – mildly tricky to sort out

### Quantifying spatial data

Requires the transformation of continuous, complex world into discrete, finite representations.



Haining (Spatial Data Analysis, 2003, p.44) distinguishes between **Objects** and **Fields**

**Objects** : **discrete** things with unique locations

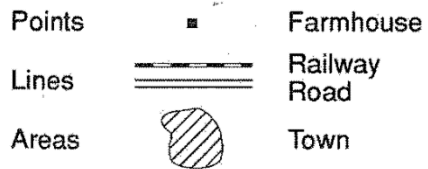
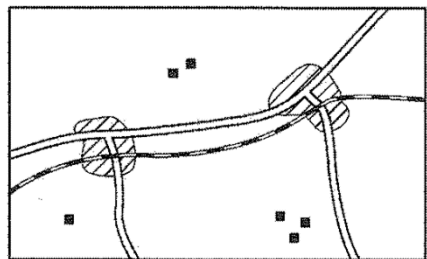
***Examples** : houses, cities (with defined boundaries), a road, a census block, a disease instance location.*

**Fields** : characteristics which exist **continuously** across a region (or a 'field!')

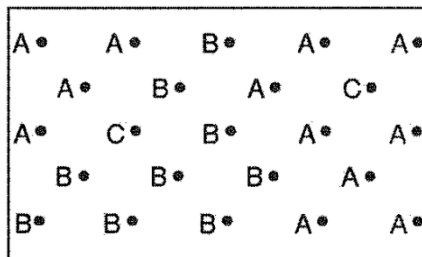
***Examples** : snow depth, soil moisture, land use, soil type, elevation, air pollution level, population density, average rent*

# Example : Haining, p.45

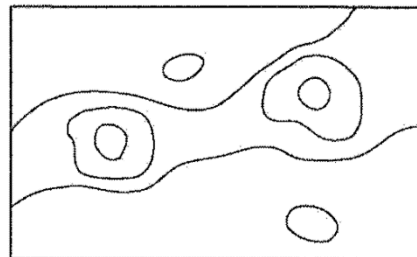
DISCRETE SPACE REPRESENTATION



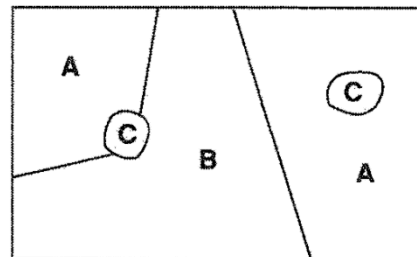
A, B Agricultural  
C Urban



CONTINUOUS SPACE REPRESENTATION



— Population density contours



Land-use surface:

A, B Agricultural  
C Urban



Waller and Gotway (p. 39) use a slightly different description :

*“Spatial data consists of **Features** indexed by spatial locations and with specified **Supports**, and **Attributes** associated with those features.”*

**Features** (objects): an **object** (or unit of measurement) with a specific spatial location.

**Support** : the properties necessary to describe a feature, namely **Size, Shape, Location, and Orientation**

Here are four types of features and supports :

<b>Feature</b>	<b>Feature Description</b>	<b>Support</b>
<b>Point</b>	A precise location in space, a dot on a map. <i>Location of a tree or a disease case</i>	Location (usually (x,y)), (no shape, orientation or size)
<b>Line</b>	A sequence of connected points (linear or curved). <i>Roads, streams, park boundaries, fault lines.</i>	Location (x,y), length, and direction/orientation. (no size)
<b>Area</b>	Region enclosed by lines. <i>Parks, states, lakes, etc.</i>	Location (x,y), area, shape (rectangle, circle, wacky), orientation.
<b>Volume</b>	An area with height/depth. <i>Aquifers</i>	Location, volume, shape, orientation

**Attributes** : measured variables associated with a particular feature

***A little problem*** : Fields consist of uncountably infinitely many features with associated attributes (*i.e in a plot, there are infinitely many locations where you could measure snow depth*).

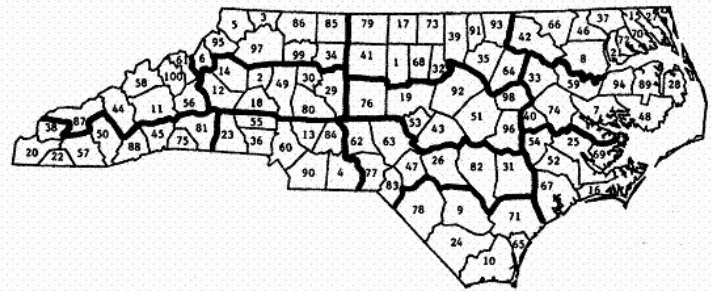


**Storing data about a field requires a finite representation. There are several ways to do this:**

- **Contour Lines / Density Estimation** : a regular curve is defined over the field
- **Pixels** : a field is divided into small regular spatial units (for example, a GRID). The size of the pixel determines the **Spatial Resolution**.

- **Regions** : If groups of pixels are relatively homogenous, a field can be partitioned into **regions** or **irregular polygons**. This requires denoting boundaries between regions. Sometimes regions are predefined and rates are then calculated within a region (i.e. states, counties, etc).

**Example** : Figure 1 of Cressie and Chan, 1989, *Spatial Modeling of Regional Variables*. *J. Amer. Stat. Assoc.*, 84:393-401. The number shown in each county is the alphabetical ordering in Table 1 of the same article, which deals with spatial modeling of counts of SIDS data for counties in North Carolina, USA.



# Spatial Scale : Point Patterns

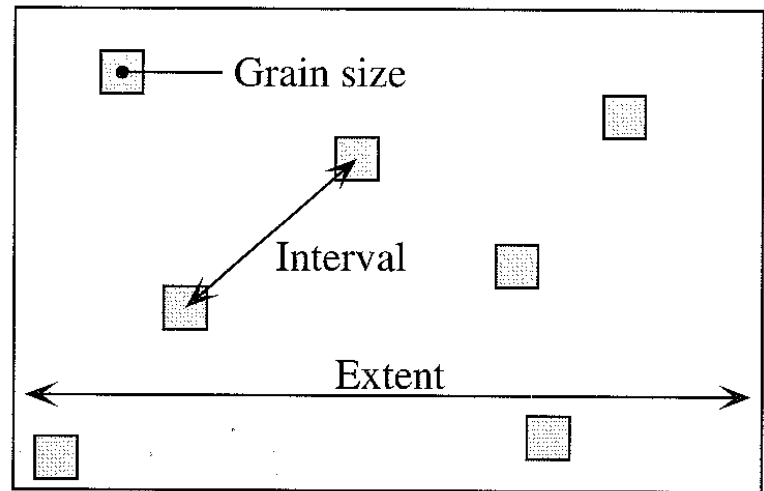


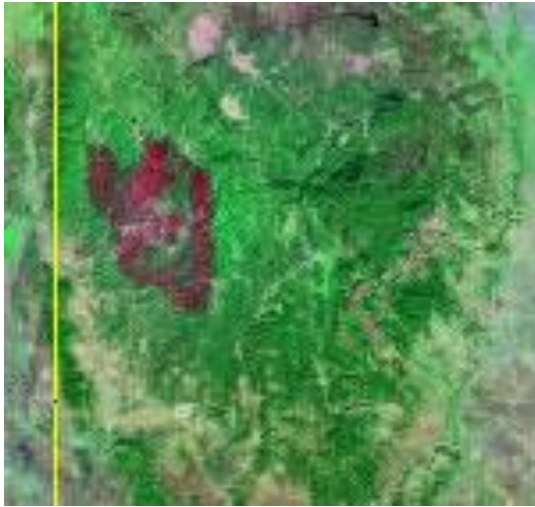
Legendre (1998, p.708) describes three elements of the scale on which spatial sampling is done :

**Grain Size** : size of elementary sampling units. Can be length, diameter, volume, or time interval. This is sometimes also called the **resolution**.

**Sampling interval** : average distance between sampling units. In time series, this the **lag**.

**Extent** : Total length, area, volume, time included in the study. Also called **range**.





**Example** : for satellite imagery, grain size = pixel size and sampling interval = grain size.

**The question of spatial scale is crucial to spatial statistics**



**Example** : Think back to pH in a field. Suppose you want to estimate the average pH in a field. How many observations should you take? (a bit of discussion here)

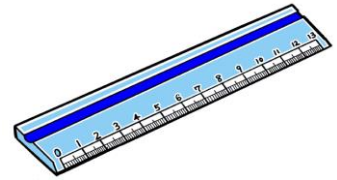
# Geodesy (where in the world . . . ?)



- Spatial data is usually measured somewhere on the earth.
- This requires a **coordinate system** (like latitude, longitude), and a **projection** (from a mostly-sphere to a plane).
- See Waller and Gotway, pages 40-49 for a nice discussion of this, or take intro to GIS, or talk Dana Tomlin (resident GIS expert).
- If you're new to this, you can lose **LOTS** of time trying to work this issue out and merge datasets (like measurement points with an outline of an island . . .)

Assuming you've solved this, we can talk about

# Distance or Metrics between Spatial Observations



## Continuous/Interval Distance Measures (a few of dozens possible)

1. **Euclidean**: root sum-of-squares of differences on each variable (most common – the straight line ‘as the crow flies’ distance between points  $a$  and  $b$ ) :



$$D_{ab} = \left( \sum_{i=1}^k (X_{ai} - X_{bi})^2 \right)^{1/2}$$

For most spatial data,  $k=2$



2. **Manhattan** : sum of absolute differences over variables  
(i.e. have to go on city blocks to get there – not as  
the crow flies)

$$D_{ab} = \sum_{i=1}^k |X_{ai} - X_{bi}|$$

*These are both specific examples of the more  
general*



3. **Minkowski** :  $D_{ab} = \left( \sum_{i=1}^k (|X_{ai} - X_{bi}|)^m \right)^{1/m}$

4. **Squared Euclidean** :  $D_{ab} = \sum_{i=1}^k (X_{ai} - X_{bi})^2$

5. **Great Arc Length** : This applies to two points on a globe :  $a = (\lambda_1, \phi_1)$ ,  $b = (\lambda_2, \phi_2)$  where  $(\lambda, \phi)$  are the longitude, latitude values. For these points



$$D_{ab} = 6378 \arccos[\sin \phi_1 \sin \phi_2 + \cos \phi_1 \cos \phi_2 \cos(\lambda_1 - \lambda_2)]$$

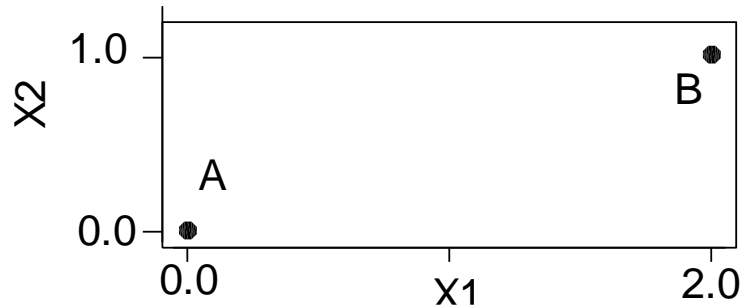
(where 6378 is the radius of the spherical earth in km)

6. **Time** : any ordinal metric that seems appropriate

**Example :** Two points, two variables ( $n=2, k=2$ )

A : (0,0)

B : (2,1)



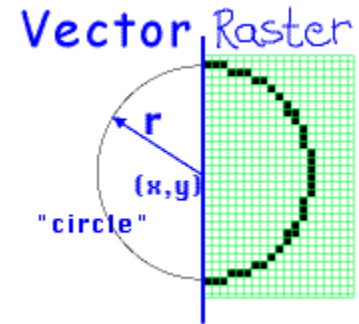
*Euclidean distance* :  $\sqrt{(1-0)^2 + (2-0)^2} = 2.24$

*Manhattan distance* :  $|1-0| + |2-0| = 3$

# Recording Spatial Data : Vector vs . Rastor

## Vector

- Individual locations are recorded using vector notation (direction, length).
- Regions are defined by connected vectors which make a polygon.
- Usually requires less storage space than rastor.



## Rastor

- Data is recorded for every pixel in a grid over a defined region.

- Usually requires more storage space (but compression can reduce this)

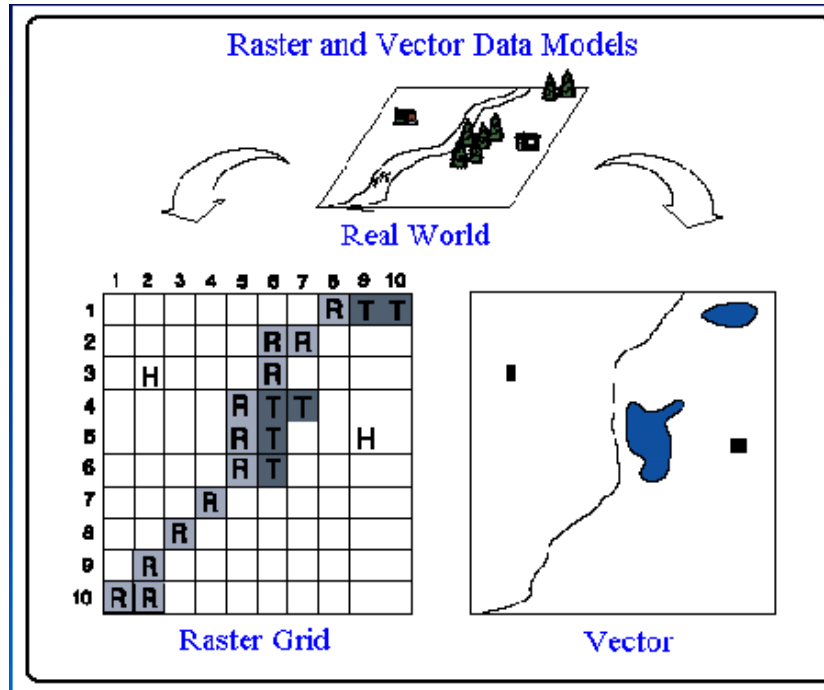


Figure by Brian Klinkenberg, Department of Geography, University of British Columbia

# Topics We'll Cover : A BRIEF Overview

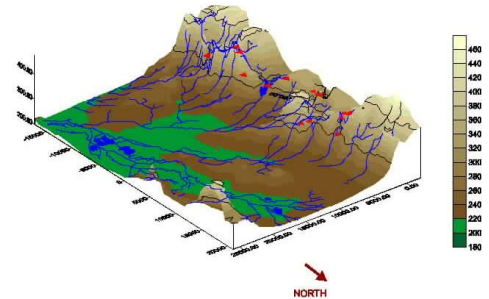
## Visualizing Spatial Data

(a graph is worth 1000 data points . . .)

We won't really cover this as a separate topic, but visualizations of spatial phenomenon are integral to the field.

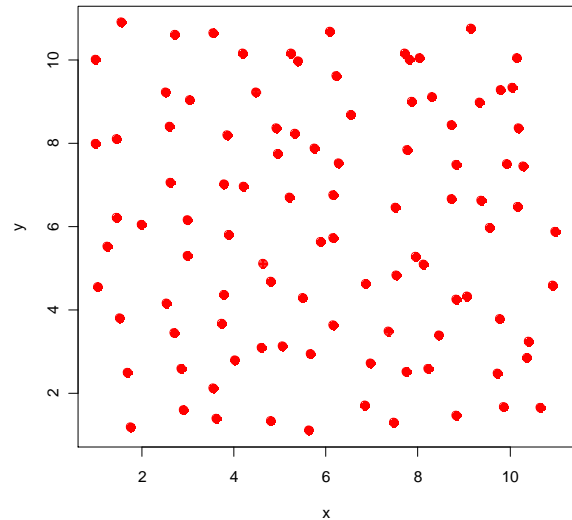
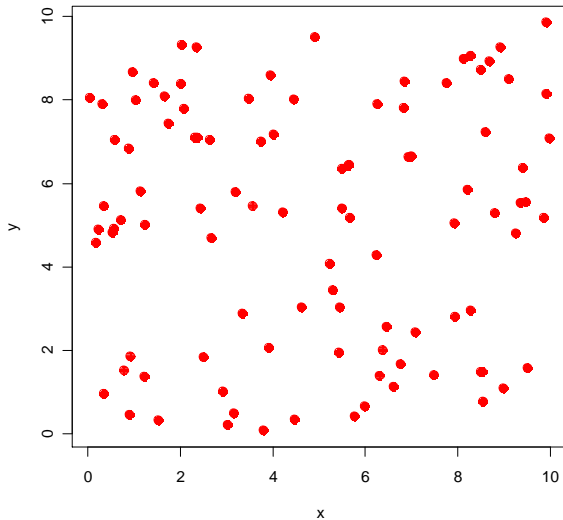
**Maps** : Point maps, contour maps, image maps, surface maps, symbol maps, density maps, etc.

**Smoothing** : weighted averages, kernel smoothing, non-parametric regression, splines, edge effects.



# Point Processes

- This deals primarily with data that occurs only at a **discrete point** (*i.e. tree location, disease location, house location, etc*).
- The detection of patterns requires knowing what we mean by 'no pattern'.
- This requires a discussion of **complete spatial randomness**.
- This allows us to then discuss other phenomenon : **clustering, regularity, stationary processes**.



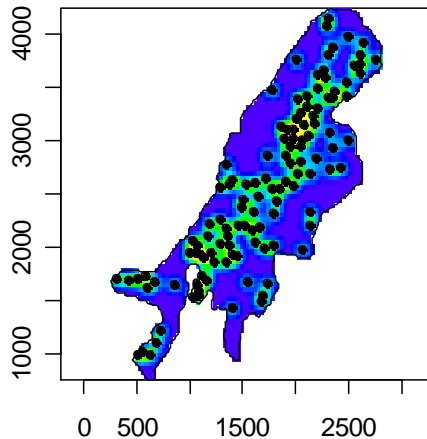
- This leads to **hypothesis tests** for spatial point patterns – random vs. not random
- Then we move to **kernel intensity/density estimation** and **K-functions** (*this means we estimate an underlying generating function for the location of points!*)



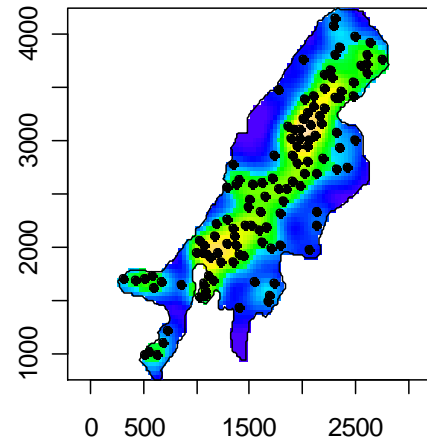
**Example : Uganda Volcano Locations.** This dataset has the locations of 120 volcanoes in the Bunyaruguru volcanic field in west Uganda. Below are kernel smoothed estimates



**Bandwidth = 100 , Grid Size = 80**

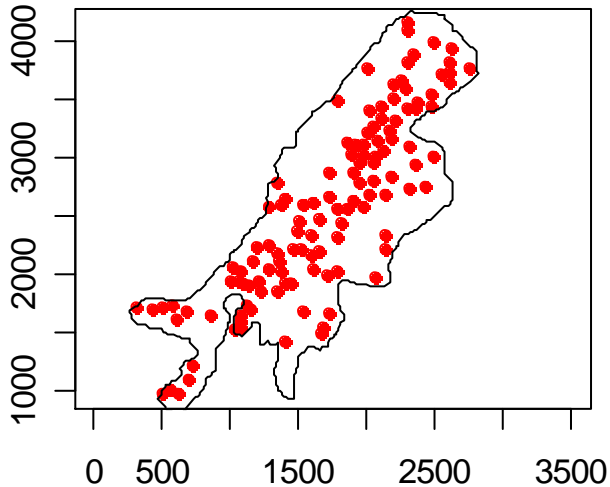


**Bandwidth = 200 , Grid Size = 80**

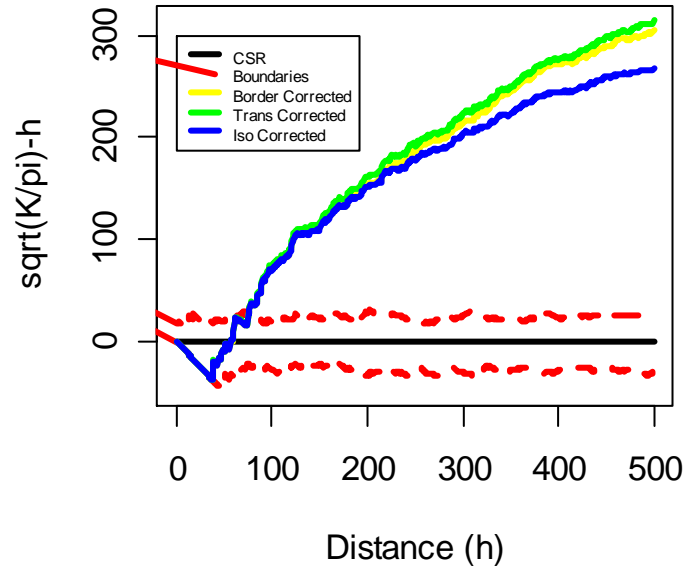


*Without going into too many details, below is a plot used to evaluate clustering/exclusion at various distances in the Uganda volcano data*

**Uganda Volcanos**



**L-plot for Uganda Volcanos**



- **Next** we compare the distribution of **two** simultaneous point processes (i.e. disease cases and controls, or two species of trees, etc.) to see if the processes are attracting/repulsing each other.
- **FINALLY** we'll talk about how to model spatial point patterns under certain assumptions.

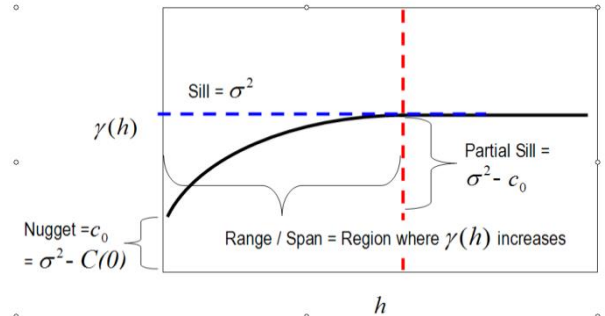
## Measures of Spatial Autocorrelation

- To see if spatially measured attributes are correlated, we have to come up with **hypothesis tests**.
- We'll discuss measures of autocorrelation : **Moran's I**, **Geary's c**.
- We'll also discuss **goodness of fit statistics** for various models.

# Geostatistical Modeling

- How do we model data where we allow the  $X$ 's to be intentionally correlated?
- This is a **BIG** topic that requires review about non-correlated linear models and random vs. fixed effects.
- Another problem that arises is how to model data where near observations are more similar – is the similarity due to an underlying common **MEAN** function, or is it because near things are **CORRELATED?**

- **Variograms, Semivariograms, Correlograms.** Essentially, this is where we quantify the first law of geography (what is a mathematical representation of how near things are related to each other, and at what distance are characteristics independent?)



## Spatial Prediction/Kriging

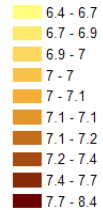
- **Spatial prediction : kriging, co-kriging.** This is really a sub-category of Generalized Linear Models for Correlated Data. It allows us to make a model using spatial data to make predictions of values at new locations using information about spatial correlation.

**Examples : Smoky Mountains pH data.** Data collected on pH levels in Great Smolky Mountain National Park. Goal is to predict pH at unmeasured locations **AND** estimate the errors at each point as well.

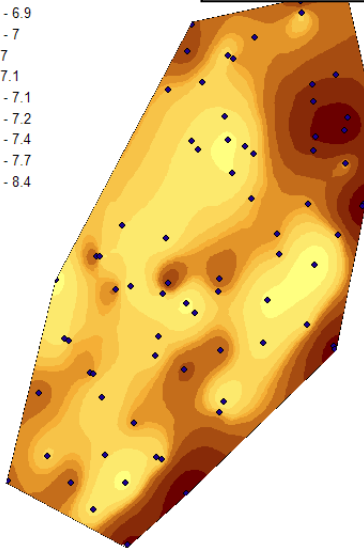


Legend  
Simple Kriging  
Prediction Map  
PH

Filled Contours



Simple Kriging Prediction



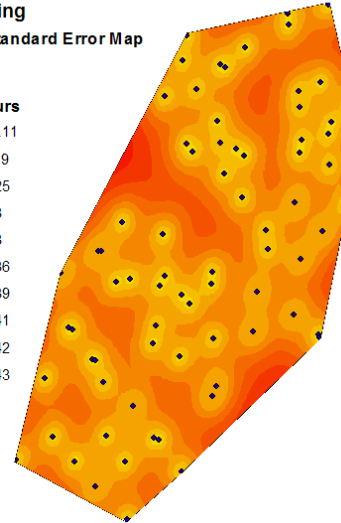
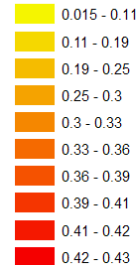
Error Surface

Legend

Simple Kriging  
Prediction Standard Error Map

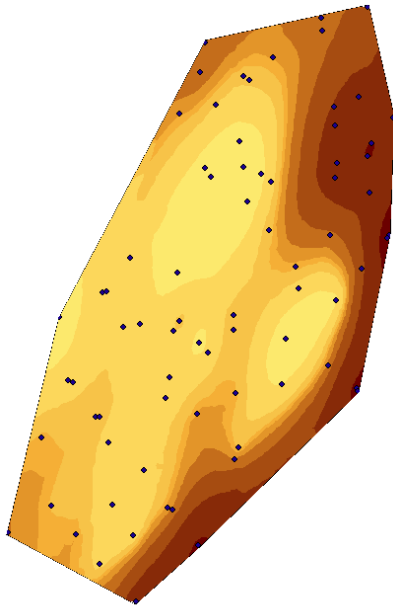
PH

Filled Contours

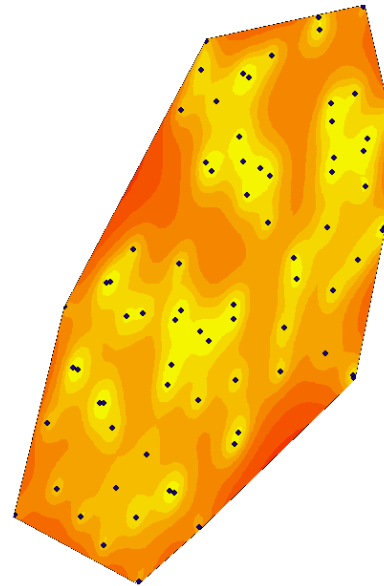


*Ordinary Kriging using anisotropic model (i.e. spatial correlation is directional!!!)*

Ordinary Kriging  
Prediction

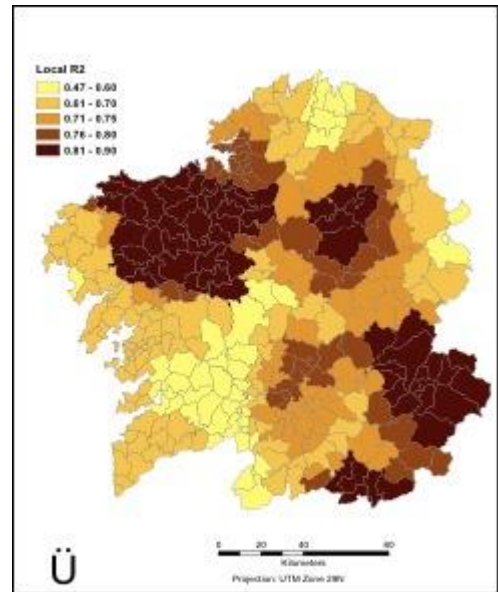


Error  
Surface



# Geographically Weighted Regression

- Exploratory technique designed to indicate where non-stationarity exists on a map
- Essentially, it identifies where local regression coefficients depart from the global estimates.
- It uses a moving weighted localized window to create localized estimates of coefficients at any chosen point.





# Matrices and Vectors



Free Linear Algebra Book online : <http://joshua.smcvt.edu/linearalgebra/>

Here's another one : <http://linear.ups.edu/download.html>

Here's a free MIT course : <http://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/>

Also, Chapter 4 of Dan Zelterman's book is PERFECT for what you need to know for this class

- Multivariate analysis involves 'many variables' (represented as  $X_1, X_2, X_3, \dots$ )
- Each continuous variable can be represented along a line in 'one dimension'

***Data Examples : COLUMNS = VARIABLES!***

**World Bank Data, 2013.** This data set examines 17 measures for 217 countries from around the world. LOTS of missing data. Here's a sample for a few countries:



Country Name	Deforestation	Rural	CO2 per Capita	GNI per capita	Energy Use 2011	Life Expectancy
Albania	1.898606	44.617	1.499316	4520	768.0615635	77.35046
Algeria	11.55369	30.49	3.331512	4970	1108.281628	70.88217
Angola	4.502755	57.51	1.555966	4510	672.739215	51.464
Antigua and Barbuda	4.854369	75.357	5.885158	12720		75.66532

**Music Attitudes.** Attitudes on 5 point scale from a 1993 survey of 1000+ people (1=strong like, 5=strong dislike)

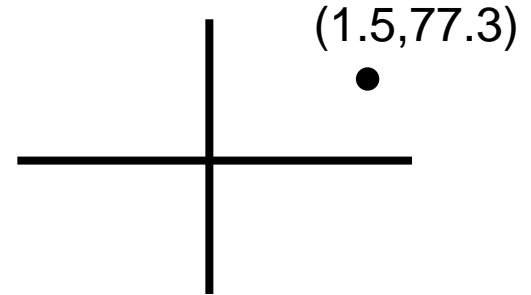


ID	country	classical	jazz	latin	rap	reggae	oldies
1	3	1	2	4	5	5	4
2	3	1	1	1	4	3	1
3	3	1	3	2	4	4	1
4	3	1	2	5	5	5	2

- The measurement of two continuous variables for a given observation can be represented by a point in two-dimensional space

*Ex : for the World Bank Data, we represent CO<sub>2</sub> per capita and Live Expectancy for Albania :*

$$X_1 = 1.5, X_2 = 77.3$$



- In general, the measurement of  $p$  continuous variables for a given object/person can be represented by a point in  $p$ -dimensional space

- **Ex:** 4-dimensional space : values for Albania:

$$X_1 = 1.5, X_2 = 4520, X_3 = 768, X_4 = 77.3$$

$X_1$  = CO2 per capita,  $X_2$  = GNI per capita,

$X_3$  = Energy Use per capita,  $X_4$  = Life Expectancy (years)

(1.5, 4520, 768, 77.3)

*(can't draw this point - but, we can represent as a list of numbers!)*

**Vectors and Matrices are how we deal  
with points in  $p$ -dimensional space!**

# Matrices

## A Matrix is :

- A rectangular group of numbers
- A useful and natural way of organizing data/information
- Notation : an  $r \times c$  matrix has  $r$  rows and  $c$  columns.
- $r$  and  $c$  constitute the **dimension** of the matrix.
- **In General, COLUMNS are variables and ROWS are observations for DATA matrices.**



**Examples** (World Bank Data, 2013). For five countries, measure 3 variables.

### Data Matrix (5x3)

Variables 

Observations  
or  
Subjects



	Deforest. Rate	% Rural	Life Expectancy
Albania	1.9	44.6	77.4
Algeria	11.6	30.5	70.9
Angola	4.5	57.5	51.5
Antigua	4.9	75.4	75.7
Argentina	16.9	8.5	76

### Correlation Matrix (3x3)

	Deforest	% Rural	Life Exp
Deforest	1	-0.82	0.42
% Rural	-0.82	1	0.08
Life Exp	0.42	0.08	1

Matrices are denoted by UPPER CASE **bold** letters

**A**, or **X** for a data matrix

A general matrix of data has  $n$  **observations (rows)** and  $p$  **variables (columns)**

$$\mathbf{X} = \begin{matrix} & \text{Var 1} & \text{Var 2} & \dots & \text{Var } p \\ \text{Obs 1} & x_{11} & x_{12} & \dots & x_{1p} \\ \text{Obs 2} & x_{21} & x_{22} & \dots & x_{2p} \\ & \vdots & \vdots & \ddots & \vdots \\ \text{Obs } n & x_{n1} & x_{n2} & \dots & x_{np} \end{matrix}$$

$n \times p$

*Often list the dimensions of the matrix at the bottom*

For the Music data, the entire dataset is a matrix with 15 columns and 1134 rows.

A matrix with only one column (or row) is a **VECTOR**.



- Vectors are denoted by a **bold** lower case letter (sometimes with a squiggle on top) like  **$\mathbf{x}$**  or  **$\tilde{\mathbf{x}}$**

**Examples** : Deforestation Rate Column Vector  **$\mathbf{x}$**  :

Row Vector for Albania :

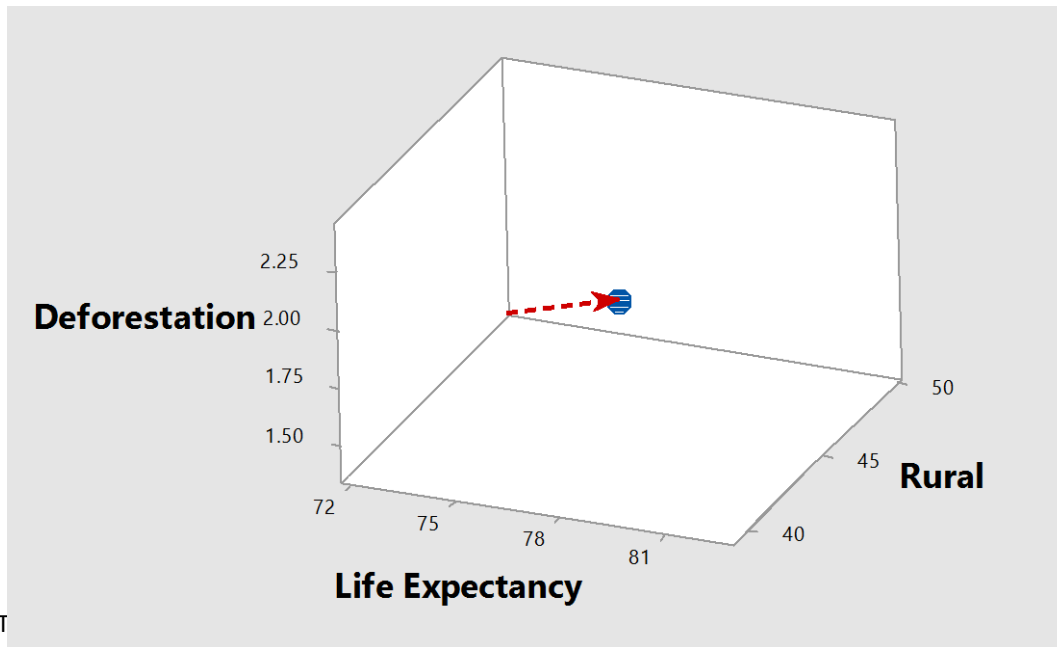
$$\underset{1 \times 3}{\mathbf{x}} = [1.9 \quad 44.6 \quad 77.4]$$

$$\underset{5 \times 1}{\mathbf{x}} = \begin{bmatrix} 1.9 \\ 11.6 \\ 4.5 \\ 4.9 \\ 16.9 \end{bmatrix}$$



Thus, a row vector for a dataset represents all the data collected for that observation and is just ***a point in p-dimensional space:***

$$\mathbf{x}_{1 \times 3} = [1.9 \quad 44.6 \quad 77.4]$$



**Vectors are also useful for giving summary statistics for a whole set of variables :**

***Example : Music Attitudes Data*** - 7x1 vectors gives the mean and standard deviation of the average attitude for each type of music simultaneously :



$\bar{\mathbf{x}}$ 7x1	country	2.39	$\mathbf{S}$ 7x1	country	1.10
	classicl	2.65		classicl	1.22
	jazz	2.54		jazz	1.08
	latin	3.15		latin	1.08
	rap	3.88		rap	1.13
	reggae	3.15		reggae	1.18
	oldies	2.15		oldies	1.06

**Square Matrix** : has equal numbers of rows and columns  
(i.e.  $r = c$ )

*A couple of square 3x3 matrices :*

$$\mathbf{A}_{3 \times 3} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 1 & 0 & 8 \end{bmatrix}$$

$$\mathbf{B}_{3 \times 3} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} 6 & 1 & 0 \\ 2 & 8 & 7 \\ 3 & 4 & 5 \end{bmatrix}$$

*Read subscripts as 'row, column' (i.e. this is the number in the third row and the second column)*

**Trace of a Matrix** : the sum of the elements on the diagonal of a SQUARE matrix (can't get the trace of a non-square matrix)



**Examples :**

$$tr(\mathbf{A}) = tr \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 1 & 0 & 8 \end{bmatrix} = 16$$

$$tr(\mathbf{I}) = tr \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 3$$

## Transpose of a Matrix

Just reverse the rows and the columns (first row becomes first column, second row is second column, etc.). Written as  $\mathbf{A}'$  (' $\mathbf{A}$  transpose')



$$\text{If } \mathbf{A}_{2 \times 3} = \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \end{bmatrix} \text{ then } \mathbf{A}'_{3 \times 2} = \begin{bmatrix} 3 & 7 \\ 2 & 5 \\ 4 & 0 \end{bmatrix}$$

**Definition** : if  $\mathbf{A} = \mathbf{A}'$  then  $\mathbf{A}$  is said to be **Symmetric** (opposite is also true : if  $\mathbf{A}$  is symmetric then  $\mathbf{A} = \mathbf{A}'$ )

**Example : Correlation matrices are symmetric**

	Deforest	Fem. illit	Male illit.
Deforest	1	0.6	0.64
Fem. illit	0.6	1	0.99
Male illit	0.64	0.99	1



**Note** - for a matrix to be **symmetric**, it has to be a **square matrix!**

*A related idea is :*

### **Transpose of a vector**

- A vector **a** with only one column (an  $r \times 1$  matrix) becomes a vector **a'** (a  $1 \times r$  matrix)

**Example :**  $\mathbf{a} = \begin{bmatrix} 3 \\ 7 \\ -2 \end{bmatrix}$ ,  $\mathbf{a}' = [3 \quad 7 \quad -2]$ ,

## Manipulating Matrices

- Most of the usual algebraic manipulations of numbers have a counterpart in matrices.

### Addition/Subtraction of a constant

Add constant to each element of matrix. If  $k = -3$ ,

$$\mathbf{A} + k = \begin{bmatrix} a_{11} + k & a_{12} + k & a_{13} + k \\ a_{21} + k & a_{22} + k & a_{23} + k \\ a_{31} + k & a_{32} + k & a_{33} + k \end{bmatrix}$$

$$= \begin{bmatrix} 3-3 & 2-3 & 4-3 \\ 7-3 & 5-3 & 0-3 \\ 1-3 & 0-3 & 8-3 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 \\ 4 & 2 & -3 \\ -2 & -3 & 5 \end{bmatrix}$$



## Multiplication/Division by a constant

Multiply each element of matrix by the constant. If  $k = 2$ ,

$$\mathbf{A}k = \begin{bmatrix} a_{11}k & a_{12}k & a_{13}k \\ a_{21}k & a_{22}k & a_{23}k \\ a_{31}k & a_{32}k & a_{33}k \end{bmatrix}$$
$$== \begin{bmatrix} 3*2 & 2*2 & 4*2 \\ 7*2 & 5*2 & 0*2 \\ 1*2 & 0*2 & 8*2 \end{bmatrix} = \begin{bmatrix} 6 & 4 & 8 \\ 14 & 10 & 0 \\ 2 & 0 & 16 \end{bmatrix}$$

**Addition/Subtraction of matrices.** Just add element by element.

- **ONLY possible if the dimensions of the matrices are the same.**

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & a_{13} + b_{13} \\ a_{21} + b_{21} & a_{22} + b_{22} & a_{23} + b_{23} \\ a_{31} + b_{31} & a_{32} + b_{32} & a_{33} + b_{33} \end{bmatrix}$$

*For example :*

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 1 & 0 & 8 \end{bmatrix} + \begin{bmatrix} 6 & 1 & 0 \\ 2 & 8 & 7 \\ 3 & 4 & 5 \end{bmatrix} = \begin{bmatrix} 9 & 3 & 4 \\ 9 & 13 & 7 \\ 4 & 4 & 13 \end{bmatrix}$$

*(Note: The original image contains a typo in the second matrix of the addition, where the second row is [2, 8, 7] instead of [2, 8, 0].)*

But . . .

$$\mathbf{C} + \mathbf{B} = \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \end{bmatrix} + \begin{bmatrix} 6 & 1 & 0 \\ 2 & 8 & 7 \\ 3 & 4 & 5 \end{bmatrix} = ??@#\$ @\#$$

*(Note: The original image contains a typo in the first matrix of the addition, where the second row is [7, 5, 0] instead of [7, 5, 0], and the result is a placeholder for an invalid operation.)*

## Multiplication of Matrices (not what you'd think)

$$\mathbf{AB} = \begin{bmatrix} \overrightarrow{a_{11} \quad a_{12} \quad a_{13}} \\ a_{21} \quad a_{22} \quad a_{23} \\ a_{31} \quad a_{32} \quad a_{33} \end{bmatrix} \begin{bmatrix} \downarrow b_{11} \quad b_{12} \quad b_{13} \\ b_{21} \quad b_{22} \quad b_{23} \\ b_{31} \quad b_{32} \quad b_{33} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{bmatrix}$$

Because of the definition,  $\mathbf{AB} \neq \mathbf{BA}$  (in general)

For our example,

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 1 & 0 & 8 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 6 & 1 & 0 \\ 2 & 8 & 7 \\ 3 & 4 & 5 \end{bmatrix}$$

$$\mathbf{AB} = \begin{bmatrix} 3*6+2*2+4*3 & \text{etc.} & \text{etc.} \\ 7*6+5*2+0*3 & \text{etc.} & \text{etc.} \\ 1*6+0*2+8*3 & \text{etc.} & \text{etc.} \end{bmatrix} = \begin{bmatrix} 34 & 35 & 34 \\ 52 & 47 & 35 \\ 30 & 33 & 40 \end{bmatrix}$$

$$\mathbf{BA} = \begin{bmatrix} 3*6+1*7+0*1 & \text{etc.} & \text{etc.} \\ 2*3+8*7+7*1 & \text{etc.} & \text{etc.} \\ 3*3+4*7+5*1 & \text{etc.} & \text{etc.} \end{bmatrix} = \begin{bmatrix} 25 & 17 & 24 \\ 69 & 44 & 64 \\ 42 & 26 & 52 \end{bmatrix}$$

**NOTE : TO MULTIPLY MATRICES, THE NUMBER OF COLUMNS IN THE FIRST MATRIX **MUST** EQUAL THE NUMBER OF ROWS IN THE SECOND MATRIX.**

*Examples of Matrix/Vector multiplication :*

$$\mathbf{a}' \mathbf{a} = \begin{matrix} 1 \times 3 & 3 \times 1 \\ \mathbf{[3} & 7 & -2] \end{matrix} \begin{matrix} \mathbf{[} \\ 3 \\ 7 \\ -2 \\ \mathbf{]} \end{matrix} = 3^2 + 7^2 + (-2)^2 = 62$$

This is a 1x1 matrix, i.e. just a number, called a **SCALER**.

$$\mathbf{a} \mathbf{a}' = \begin{matrix} 3 \times 1 \\ 1 \times 3 \end{matrix} \begin{bmatrix} 3 \\ 7 \\ -2 \end{bmatrix} \begin{bmatrix} 3 & 7 & -2 \end{bmatrix} = \begin{bmatrix} 9 & 21 & -6 \\ 21 & 49 & -14 \\ -6 & -14 & 4 \end{bmatrix}$$

Here, get a 3x3 symmetric matrix.

$$\mathbf{a}' \mathbf{A} = \begin{matrix} 1 \times 3 \\ 3 \times 3 \end{matrix} \begin{bmatrix} 3 & 7 & -2 \end{bmatrix} \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 1 & 0 & 8 \end{bmatrix} = \begin{bmatrix} 56 & 41 & -4 \end{bmatrix}$$

This produces a 1x3 vector (matrix)

$$\mathbf{a}' \mathbf{A} \mathbf{a} = \underset{1 \times 3}{[3 \quad 7 \quad -2]} \underset{3 \times 3}{\begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 1 & 0 & 8 \end{bmatrix}} \underset{3 \times 1}{\begin{bmatrix} 3 \\ 7 \\ -2 \end{bmatrix}} = 463$$

Here, we get a scalar (a number)!

*Examples :*

- *2x3 times 3x5 matrix gives a 2x5 matrix*
- *1x6 times a 6x1 matrix gives a 1x1 matrix = a number!*
- *1x6 matrix times a 2x4 matrix can't be done!*





*You try it : For the matrices below find*

$$\mathbf{A}+3, \mathbf{A}+\mathbf{C}, \mathbf{B}+\mathbf{C}$$

$$\mathbf{AB}, \mathbf{BA}, \mathbf{B}'$$

First column vector of  $\mathbf{C}$  times the transpose of the same column.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 8 \\ 1 & 0 & 8 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 1 & 1 \\ 2 & 3 \\ 3 & -1 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 2 & 0 & 1 \\ 2 & 0 & 3 \\ 4 & 0 & 1 \end{bmatrix}$$

# Matrices in

R was *made* for doing matrices – it thinks in matrices! You create ‘objects’ that are matrices, and then it’s easy to manipulate them.

*There are lots of ways to get data into R, but here I’m doing it manually*

```
matrixA<-matrix(c(3,2,4,7,5,0,1,0,8),ncol=3,byrow=T)
matrixB<-matrix(c(6,1,0,2,8,7,3,4,5),ncol=3,byrow=T)
```

*Here is the first matrix*

```
matrixA
      [,1] [,2] [,3]
[1,]    3    2    4
[2,]    7    5    0
[3,]    1    0    8
```

*The A matrix + 5*

```
matrixA+5
      [,1] [,2] [,3]
[1,]    8    7    9
[2,]   12   10    5
[3,]    6    5   13
```

## *Add the A and B matrices*

**matrixA+matrixB**

	[,1]	[,2]	[,3]
[1,]	9	3	4
[2,]	9	13	7
[3,]	4	4	13

## *Multiply the A and B matrices*

**matrixA%\*%matrixB**

	[,1]	[,2]	[,3]
[1,]	34	35	34
[2,]	52	47	35
[3,]	30	33	40

## *Transpose of a matrix*

**t(matrixA)**

	[,1]	[,2]	[,3]
[1,]	3	7	1
[2,]	2	5	0
[3,]	4	0	8

## Identity Matrix : $\mathbf{I}$ (the multivariate version of $\mathbf{1}$ )

- Square, symmetric matrix
- Diagonal elements are all 1
- All other elements are zero
- Matrix product of  $\mathbf{I}$  with any other matrix  $\mathbf{A}$  is just the other matrix  $\mathbf{A}$  :  
 $\mathbf{AI} = \mathbf{IA} = \mathbf{A}$

$$\mathbf{I}_{3 \times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

*This lets us talk about . .*

## Inverse of a Matrix : $\mathbf{A}^{-1}$

- Only exists for square matrices
- The matrix such that  $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$
- Doesn't always exist



- Only exists if the rows and columns of  $\mathbf{A}$  are all **linearly independent** (i.e. no row or column is simply the linear combination of other rows or columns).

**Example** : The columns of this matrix are independent because the only constants  $c_1, c_2$  that will make the equation true are  $c_1 = c_2 = 0$

$$\mathbf{A} = \begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}, \quad c_1 \begin{bmatrix} 3 \\ 4 \end{bmatrix} + c_2 \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

However, this matrix does not have independent columns since the constants  $c_1 = 1, c_2 = -2$  are a solution to the equation.

$$\mathbf{A} = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}, \quad 1 * \begin{bmatrix} 4 \\ 2 \end{bmatrix} - 2 * \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- If the inverse exists,  $\mathbf{A}$  is called **non-singular** or **invertible**.
- If the inverse does not exist,  $\mathbf{A}$  is called **singular** or **non-invertible**.

*Example : The inverse of the matrix :*

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 1 & 0 & 8 \end{bmatrix}, \quad \mathbf{A}^{-1} = \begin{bmatrix} -3.33 & 1.33 & 1.67 \\ 4.67 & -1.67 & -2.33 \\ 0.42 & -0.17 & -0.08 \end{bmatrix}$$

However, the inverse of this matrix does not exist since the last row is equal to the sum of the first two rows. This matrix is **singular** (non-invertible).

$$\mathbf{A}_{3 \times 3} = \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 10 & 7 & 4 \end{bmatrix}$$

**You figure it out :** what is the inverse of  $\mathbf{I}$ ?  
What is the inverse of the matrix below?

$$\mathbf{A}_{3 \times 3} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



*Getting the inverse – can be done by hand, but it's messy.*



**Matrix Inverse in R: Use the `solve()` function**

`solve(matrixA)`

```
          [,1]      [,2]      [,3]
[1,] -3.3333333  1.3333333  1.6666667
[2,]  4.6666667 -1.6666667 -2.3333333
[3,]  0.4166667 -0.1666667 -0.0833333
```



## Determinant of a matrix

- Denoted by  $|\mathbf{A}|$
- Calculated only for **SQUARE** matrices.
- It's a **single value** (number) or a **scaler**
- Calculation is somewhat complicated.



For a 2x2 matrix :

$$\mathbf{A}_{2 \times 2} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\text{Det}(\mathbf{A}) = |\mathbf{A}| = ad - bc$$

For a 3x3 matrix, put the matrix next to itself. Multiply along arrows and then add/subtract as indicated

$$\begin{array}{c}
 \begin{array}{cc}
 \mathbf{+} & \mathbf{+} \\
 \left[ \begin{array}{ccc} a & b & c \\ d & e & f \\ g & h & i \end{array} \right] & \left[ \begin{array}{ccc} a & b & c \\ d & e & f \\ g & h & i \end{array} \right]
 \end{array}
 &
 \begin{array}{c}
 \begin{array}{ccc}
 \mathbf{-} & \mathbf{-} & \mathbf{-} \\
 \left[ \begin{array}{ccc} a & b & c \\ d & e & f \\ g & h & i \end{array} \right] & \left[ \begin{array}{ccc} a & b & c \\ d & e & f \\ g & h & i \end{array} \right]
 \end{array}
 \end{array}
 \end{array}$$

$$aei + bfg + cdh - ceg - afh - bdi$$

**Example :**  $|\mathbf{A}| = \begin{bmatrix} 3 & 2 & 4 \\ 7 & 5 & 0 \\ 1 & 0 & 8 \end{bmatrix} = -12$



**Matrix Determinant in R:** Use the `det()` function

`det(matrixA)`

[1] -12

## A Few Matrix Algebra Rules

**Addition :**  $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$

**Multiplication :**

In general,  $\mathbf{AB} \neq \mathbf{BA}$

However,  $\mathbf{A(BC)} = (\mathbf{AB})\mathbf{C}$  (order doesn't matter)

**Inverse :**  $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$


**Transpose :**  $(\mathbf{AB})' = \mathbf{B}'\mathbf{A}'$

# Variances and Covariances

The sample Variance / Standard Deviation of  $X_1$  is

$$Var(X_1) = \frac{\sum_{i=1}^n (x_{1i} - \bar{x}_1)^2}{(n-1)} \quad s_{X_1} = \sqrt{Var(X_1)}$$

and the sample correlation  $r$  of  $X_1$  and  $X_2$  is defined as

$$r(X_1, X_2) = \frac{\sum (x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2)}{\sqrt{\sum (x_{1i} - \bar{x}_1)^2} \sqrt{\sum (x_{2i} - \bar{x}_2)^2}}$$


The denominator of this equation is the product of the standard deviations of  $X_1$  and  $X_2$  (and  $1/(n-1)$  )

The sample **Covariance** is simply defined as the **numerator of the correlation** (multiplied by  $1/(n-1)$  ):

$$Cov(X_1, X_2) = \frac{\sum (x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2)}{(n-1)}$$

This means the the correlation can be re-written as

$$r(X_1, X_2) = \frac{Cov(X_1, X_2)}{s_{X_1} s_{X_2}}$$

***I.e. : Correlation is Standardized Covariance!***

Note that if  $X_1 = X_2$ , the covariance is simply the variance of  $X_1$

$$\text{Var}(X_1) = \text{Cov}(X_1, X_1)$$

## Correlation / Covariance matrices

**Example** : Let's suppose we observe the following data (made up from Stevens, renamed by me) :

	Male Illiteracy Rate	Female Illiteracy Rate
Canada	1	1
Russia	3	4
Albania	2	7
<b>MEAN</b>	<b>2</b>	<b>4</b>

Let's look at the mean-centered data, or the deviations (i.e. how much does each data point for each variable vary around the mean)

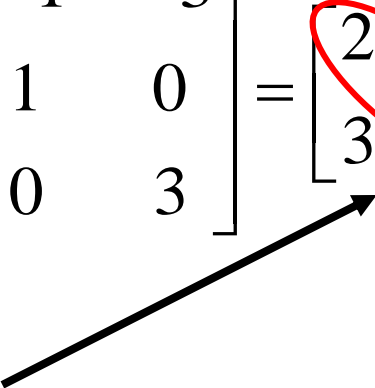
$$\mathbf{X}_d = \begin{matrix} & \mathbf{X} & & \\ & & & \mathbf{\bar{X}} \\ & & & & \\ \mathbf{X}_d = & \begin{bmatrix} 1 & 1 \\ 3 & 4 \\ 2 & 7 \end{bmatrix} & - & \begin{bmatrix} 2 & 4 \\ 2 & 4 \\ 2 & 4 \end{bmatrix} & = & \begin{bmatrix} -1 & -3 \\ 1 & 0 \\ 0 & 3 \end{bmatrix} & = & \begin{bmatrix} x_{11} - \bar{x}_{.1} & x_{12} - \bar{x}_{.2} \\ x_{21} - \bar{x}_{.1} & x_{22} - \bar{x}_{.2} \\ x_{31} - \bar{x}_{.1} & x_{32} - \bar{x}_{.2} \end{bmatrix} \end{matrix}$$

The transpose of this matrix is  $\mathbf{X}'_d = \begin{bmatrix} -1 & 1 & 0 \\ -3 & 0 & 3 \end{bmatrix}$





Define the matrix of **Sums of Squares and Cross-Products**:

$$\mathbf{SSCP} = \mathbf{X}'_d \mathbf{X}_d = \begin{bmatrix} -1 & 1 & 0 \\ -3 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1 & -3 \\ 1 & 0 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 2 & 3 \\ 3 & 18 \end{bmatrix}$$


Note that

- This is a square, symmetric matrix
- The main diagonal entries are the numerators of the

variance :  $Var(X_1) = \frac{\sum_{i=1}^n (x_{1i} - \bar{x}_1)^2}{(n-1)}$  and the off-diagonal entries are the numerator of the covariance :

$$\begin{aligned}
\mathbf{X}'_d \mathbf{X}_d &= \begin{bmatrix} x_{11} - \bar{x}_{.1} & x_{21} - \bar{x}_{.1} & x_{31} - \bar{x}_{.1} \\ x_{12} - \bar{x}_{.2} & x_{22} - \bar{x}_{.2} & x_{32} - \bar{x}_{.2} \end{bmatrix} \begin{bmatrix} x_{11} - \bar{x}_{.1} & x_{12} - \bar{x}_{.2} \\ x_{21} - \bar{x}_{.1} & x_{22} - \bar{x}_{.2} \\ x_{31} - \bar{x}_{.1} & x_{32} - \bar{x}_{.2} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^3 (x_{i1} - \bar{x}_{.1})^2 & \sum_{i=1}^3 (x_{i1} - \bar{x}_{.1})(x_{i2} - \bar{x}_{.2}) \\ \sum_{i=1}^3 (x_{i1} - \bar{x}_{.1})(x_{i2} - \bar{x}_{.2}) & \sum_{i=1}^3 (x_{i2} - \bar{x}_{.2})^2 \end{bmatrix}
\end{aligned}$$

**SO** : dividing **SSCP** by  $(n-1)$  gives the matrix of **VARIANCES AND COVARIANCES**

$$\mathbf{S} = \frac{\mathbf{SSCP}}{(n-1)} = \begin{bmatrix} 1 & 1.5 \\ 1.5 & 9 \end{bmatrix}$$

*Variance of Male Illiteracy* (points to 1)

*Covariance of Male and Female Illiteracy* (points to 1.5)

*Variance of Female Illiteracy* (points to 9)

Incidentally, the **TRACE** of **S** gives the **TOTAL VARIANCE** of all variables (just add up the diagonal values – this turns out to be useful in PCA . . .)

# Matrices meet Regression

**Usual multiple regression model in scalar notation** : if you have  $p$  predictors, the value of  $Y_i$  for a single observation  $i$  is given by

$$Y_i = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p + \varepsilon_i , \quad i = 1, 2, \dots, n$$

$$\varepsilon_i \sim N(0, \sigma^2)$$

**NOW** : if  $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$  is a  $p \times 1$  column vector and

$\mathbf{x}'_i = [1 \quad X_1 \quad X_2 \quad \cdots \quad X_p]$  is a  $1 \times p$  row vector then

$$Y_i = \mathbf{x}'_i \boldsymbol{\beta} + \varepsilon_i \quad (\text{very compact})$$



**NEXT** – stack up all the these equations on top of each other, one for each observation in the dataset :

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_p \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ 1 & x_{31} & x_{32} & \dots & x_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_p \end{bmatrix}$$

**THAT IS :**

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

# What is handy is that we can use matrices to talk about other aspects of regression

## Mean

- For the  $i$ th observation,  $\mu_i = E[Y_i/\mathbf{x}_i]$ : that is, on average, what is the expected or average value of  $Y_i$  given a particular vector  $\mathbf{x}_i$  .
- Average value of all  $n$  observations :

$$\bar{Y} = \text{Mean of } \mathbf{Y} = \boldsymbol{\mu} = E[\mathbf{Y} | \mathbf{X}] = \mathbf{X}\boldsymbol{\beta}$$

## Variance – assuming homoscedasticity (constant variance!)



- For the  $i$ th observation,  $V[Y_i/\mathbf{x}_i] = \sigma^2$  (same for every observation)
- Covariance matrix of all  $n$  observations :

$$V[\mathbf{Y} | \mathbf{X}] = \sigma^2 \mathbf{I} \quad (\text{where } \mathbf{I} \text{ is the identity matrix})$$

## Estimated value of $\beta$

Algebra plus partial derivatives shows that

<https://isites.harvard.edu/fs/docs/icb.topic515975.files/OLSDerivation.pdf>

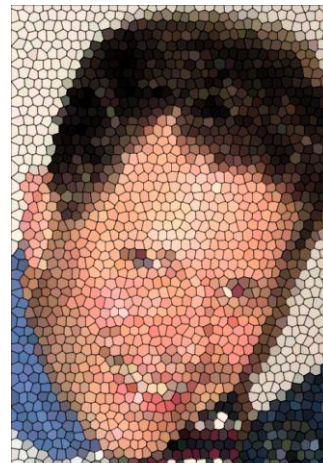
$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{Y} \quad \text{so that} \quad \hat{\mathbf{Y}} = \mathbf{X}\hat{\beta} + \mathbf{e}$$



# Spatial Point Patterns

## Resources :

- [Diggle 2014](#) Also, check out book website : <http://www.lancaster.ac.uk/staff/diggle/pointpatternbook/datasets/> )
- *B&G, Chapter 3*
- *Waller and Gotway, Chapter 5*
- [Baddeley Short Course – AWESOME!](#)
- [Baddeley et al book \(2015\)](#)
- *Cressie Chapter 8*
- *B.D. Ripley, Modelling Spatial Patterns, JRSS, 1977 (39(2), 172-212. A foundational article. (Ripley has several books on Point Processes)*
- *Online : Kate Beard of University of Maine has a nice set of notes for which I am grateful : See Lecture 6-11 : <http://reuningscherer.net/fes781/beardnotes/>*



A Spatial Pointillism Process



## What is a Point Process?

Diggle (2013) puts it nicely : *"A spatial point pattern is a set of locations, irregularly distributed within a designated region and presumed to have been generated by some form of stochastic mechanism (p. xxix)"*

- A point process consist of  $N$  observations over a region  $R$  (or a domain  $D$  or a window  $W$ ) where each observation consists of a single, discrete location.
- $R$  (or  $D$  or  $W$ ) is a subset of  $\mathcal{R}^d$ , and we'll deal with  $d = 2$  dimensions : that is  $R$  is a Region of a plane. However, becoming more common to have  $d = 3$  either as volume, or as space-time (i.e. disease case at a particular time).



## How do we get a Point Process?

- A Point Process is a special kind of **Stochastic Process**.
- A Stochastic Process is a probabilistic model defined by a set of **random variables**  $\{X_1, X_2, \dots, X_N\}$

*Recall that a random variable is a **numerical outcome of a random experiment***

- Usually the  $\{X_1, X_2, \dots, X_N\}$  represent the same measurement at the  $i$ th time or place

***Example** :  $X_i$  might be soil temperature at location  $i$*

**A Spatial Point Process is a stochastic process where each  $X_i$  is the numeric representation of a point location in space (usually  $\mathcal{R}^2$ , i.e. a planar process)**

Some authors distinguish between an

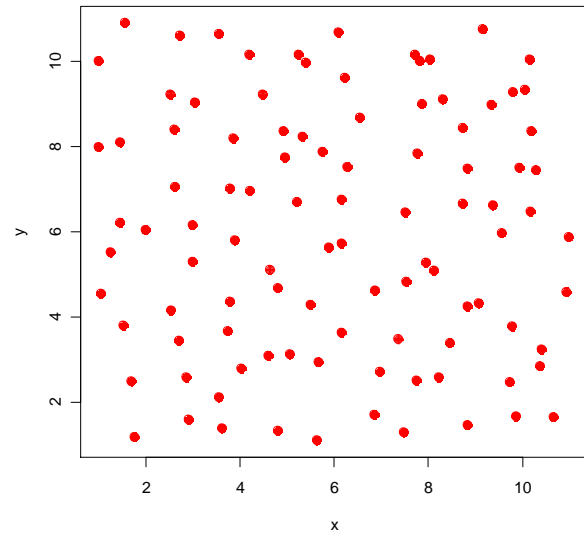
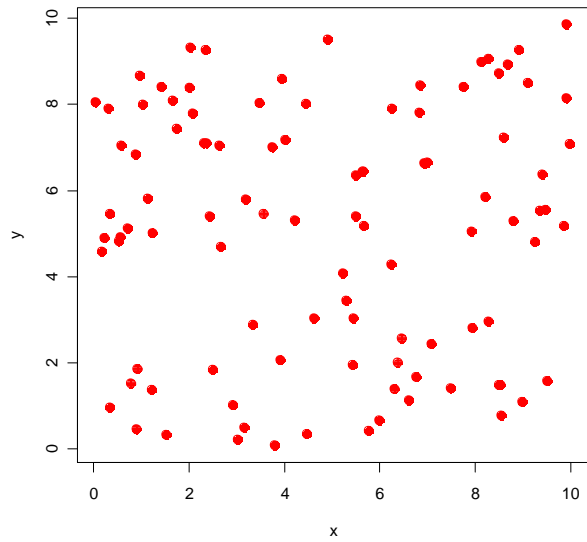
**Event** : an actual observation in region  $R$

**Point** : any other location of interest in region  $R$



## What will we do with a Point Process?

- 1) Determine if the events we observe follow a ‘**random**’ pattern or if they exhibit either **clustering** or **systematic** patterns



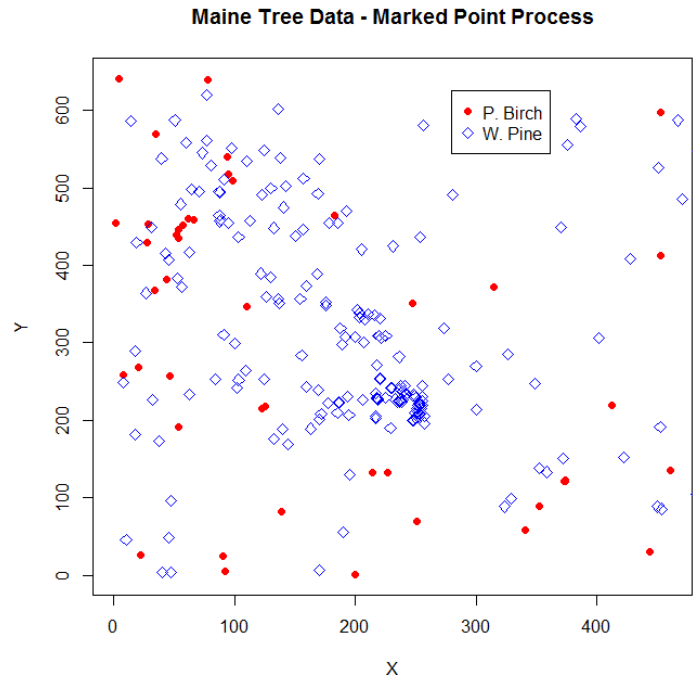
- 2) Estimate how the **intensity** of the point process varies over an area (i.e. the **MEAN** of the process)
- 3) Estimate the presence of **spatial dependence** among events (a.k.a the **VARIANCE** of the process)
- 4) Develop **models** to describe what we observe
- 5) Develop **testing procedures** to 'prove' departures from randomness
- 6) Examine how **multiple point processes** interact with each other
- 7) Examine how other continuous variables (**covariates**) interact with a spatial point process

# Definitions

**MARKS** – each point has an associated attribute which could be categorical (species, disease status), or continuous (diameter).

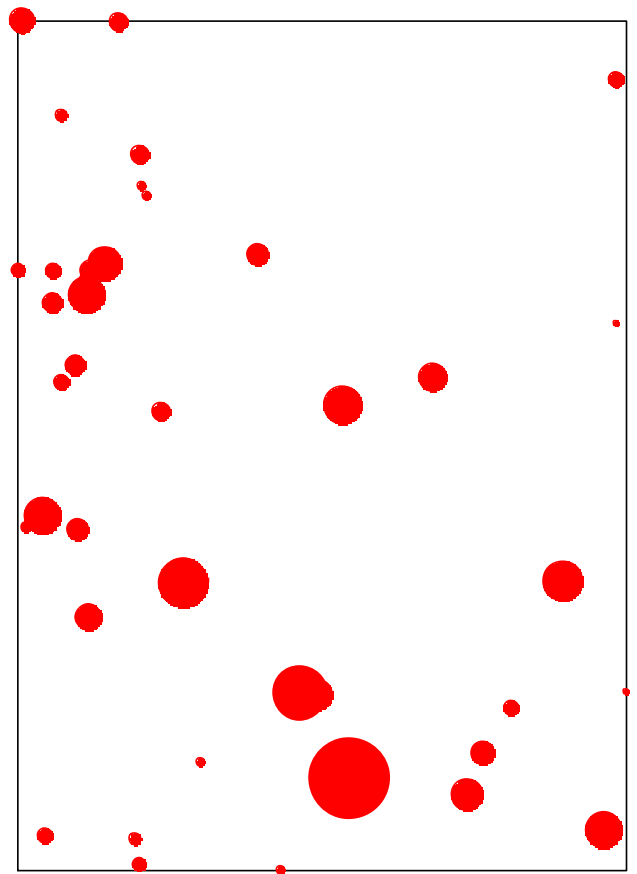
## ***Example : Maine Tree Data.***

*This is data Tim collected on different species at a site in Maine – the location of each instance of 7 tree species was noted over a rectangular region. Here is a marked version for two species : paper birch and white pine. We can look at the relative distribution of each species, but marks allow to examine whether these two processes interact with each other.*



*Just paper birch, marked by diameter at breast height (DBH) :*

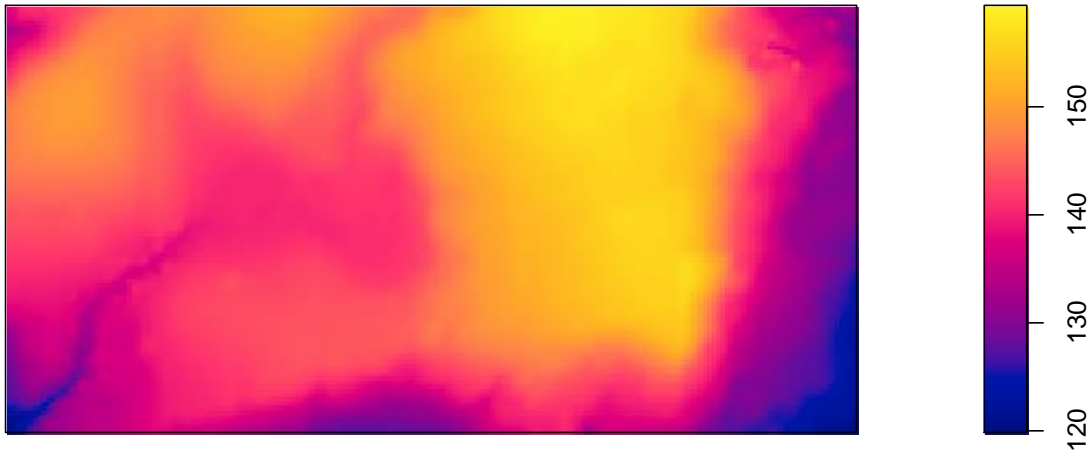
**Paper Birch - Diameter=DBH**





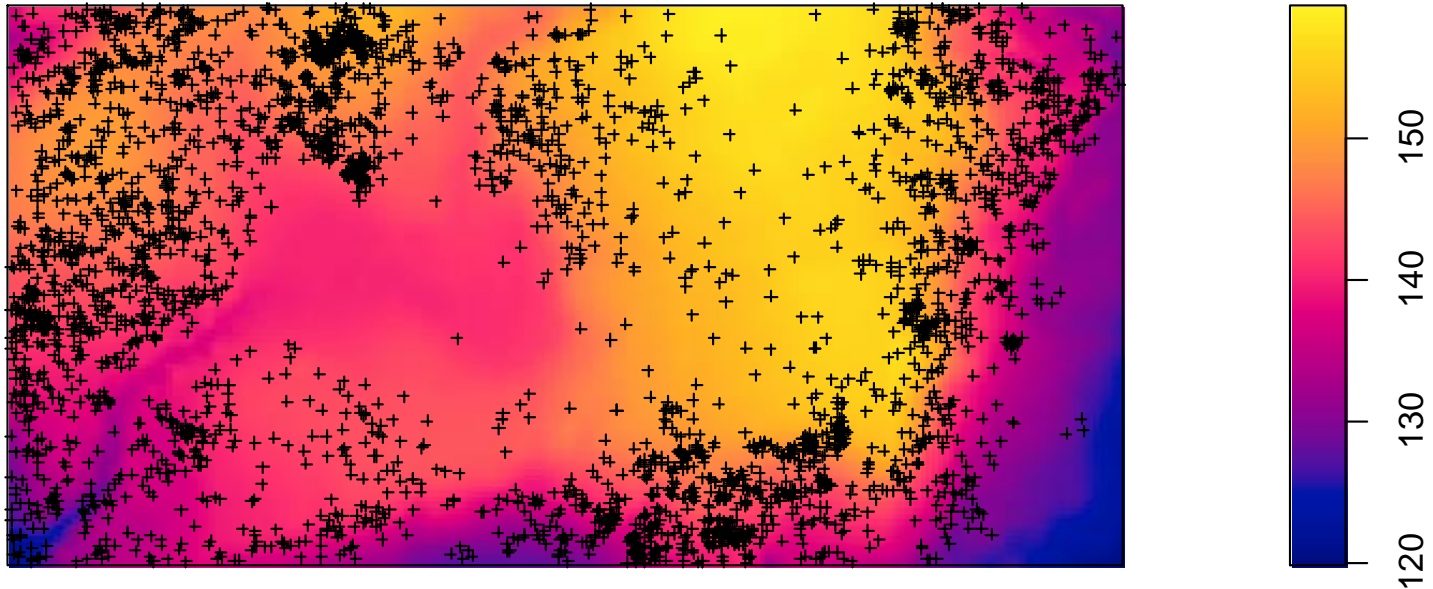
**COVARIATE** – explanatory continuous data measured across  $R$ . Can be displayed as an image or a contour plot. This information can be used to predict probability of

**Example** : (from spatstat : <http://mapas.mma.gov.br/i3geo/pacotes/rlib/win/spatstat/html/bei.html> )  
: *Elevation over tropical rain forest plot* :



By adding the location of trees, we can investigate

- Is relative density of trees related to slope (i.e. changing elevation)?
- After accounting for slope, is there evidence of clustering of trees?





## Spatial Point Processes and R

There are several dozen packages that analyze various kinds of spatial point processes. You can read about these here :

<https://cran.r-project.org/web/views/Spatial.html>

You will want to load the package `spatstat` which deals specifically with point processes and is the **main package we will use**. For information on this package, go to [www.spatstat.org](http://www.spatstat.org). There are several documents that give an overview of the functions of this package. You'll also want the `splancs` package (an older point process package – PPP!)

Functions in `spatstat` work on `ppp` objects. Use the `ppp()` function to convert matrices, dataframes, polygons, etc. to this data type.

Plots on previous pages created using this script :

<http://reuningscherer.net/fes781/rscripts/SpatialPPEXamples.r.txt>

## What Qualifies as a SPP?

*(nice discussion in section 2.2 of Baddeley notes)*



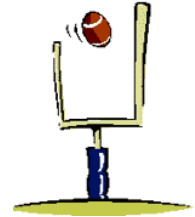
### YES :

- Locations of disease cases
- Location of trees
- Locations of volcanos in Uganda
- Locations of mineral deposits

### MAYBE :

- Locations of accidents on city streets – tricky since can't have an accident that's not on a road – it's really a 'bent' linear process
- Locations of a GPS tracker on a tiger hourly for a month – maybe not since this is really a continuous process with infrequent observations
- Locations of nuclei in densely packed cells – centers can't be next to each other because of biological restrictions – of course the same might be said of trees . . .

## Goal 1 : Random PP vs. Clustered / Systematic PP



*To have this discussion, we first have to decide what we mean by 'random'!*

### **Complete Spatial Randomness (CSR)**

- Events are equally likely to occur at any location within the defined region  $R$
- Knowledge of the boundaries of  $R$  is critical since we need to know where points did **not** occur!

- For a fixed number of events, this means that
  - 1) Events follow a **UNIFORM distribution** over the study region  $R$  :

$$f(x, y) = \frac{1}{|R|}$$

for all  $(x, y) \in R$ , where  $|R|$  is the area of  $R$

**NOTE** : Since the area of  $R$  may vary widely, it's sometimes more helpful to think that the average number of points per unit area is some fixed, constant  $\lambda$  (lambda) : this number is called the **intensity** (also called the **mean!**)

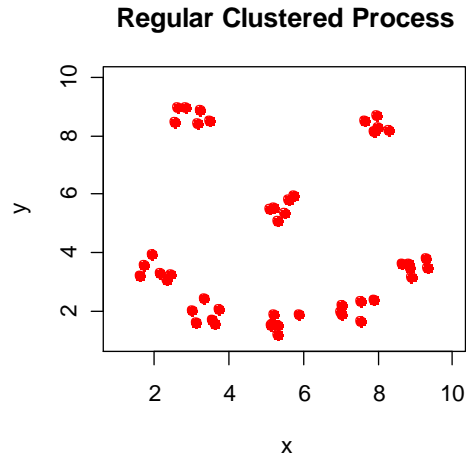
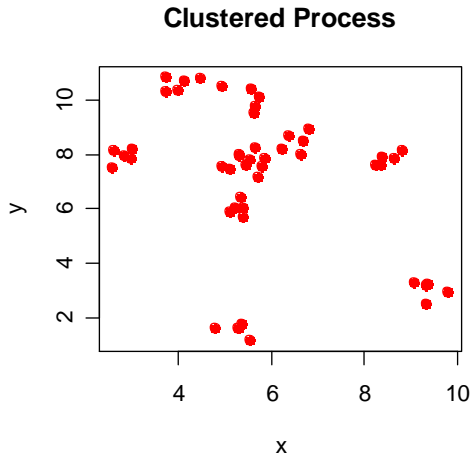
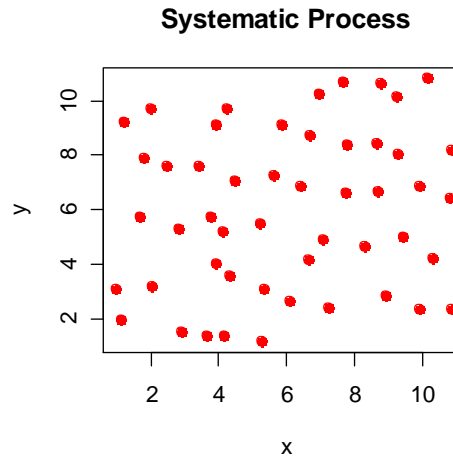
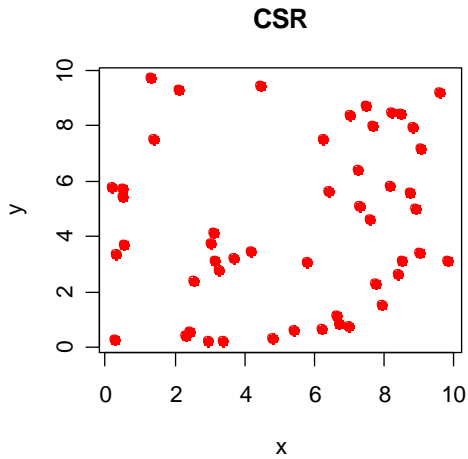


- 2) Events are **independent**

**CSR** serves as the 'baseline' for comparison of other point processes.

- **Clustered Process** : events fall into clumped groupings
- **Regular Process** : there are regular spatial intervals between events (i.e. if species are 'pushing' against each other by inhibiting growth at too close a distance)

Here are some examples :



(R code is online as <http://reuningscherer.net/fes781/rscripts/Simulated%20Spatial%20PP.r.txt>)



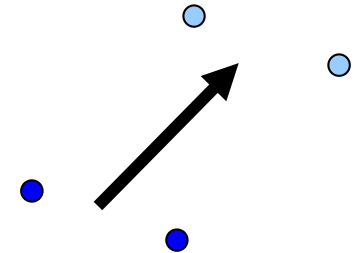
**OK, so how do we model a CSR? We need to discuss**



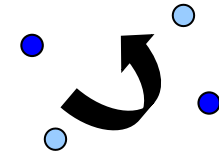
## **Isotropy and Stationarity**

**Stationary Process** : A process that is invariant to translation in space.

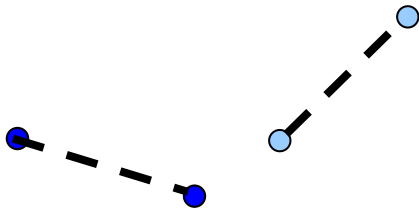
*Relationships between pairs of points depend only on their relative positions to each other.*



**Isotropic Process** : A process that is invariant to rotation in space.



**Stationary and Isotropic Process** :  
*Relationships between pairs of points depend only on the relative distance between points!*



# CSR and the Homogenous Spatial Poisson Process



Recall that if a random variable  $X$  has a **Poisson distribution** then

$$\Pr[X = x] = \frac{\lambda^x}{x!} e^{-\lambda} \quad \text{for } x = 0, 1, 2, 3, \dots$$

Recall that  $x! = (x)(x-1)(x-2)\dots(2)(1)$  and  $e = 2.718$

**Poisson distributions are used to model the number of events per unit of space or time.**

If you do some infinite series expansions, you will find



**Mean of a Poisson** =  $\lambda$  (no surprise here)

**Variance of a Poisson** =  $\lambda$  (here's the surprise!)

**NOW** : A **Homogenous Poisson Point Process** is defined as follows :

- 1) The random number of events  $X$  in a finite region  $A$  has a Poisson distribution with mean  $\lambda|A|$  where  $|A|$  is the area of  $A$  and  $\lambda$  is the **intensity**.

**Intensity** : the number of expected events per unit area. For a **homogenous** process, this is **constant** over the area.

- 2) Given that a total of  $X = N$  events occur in  $A$ , the locations of the events are uniformly distributed across  $A$

# Homogeneous Poisson Point Process = CSR

This means that for a CSR, while the observed number of points over a region  $R$  will vary from one realization to the next, the average (expected) number of points is the fixed quantity  $\lambda|R|$ .

This definition shows how to generate a CSR :

- 1) Generate a Poisson number of points  $X = N$  with mean  $\lambda|R|$ .
- 2) Randomly assign locations for the  $N$  according to a uniform distribution

*See Gotway p.124 or Cressie p.634 for an equivalent definition of a homogeneous Poisson Point Process*

The definition of a homogenous Poisson PP means that is both **stationary** and **isotropic**!

---



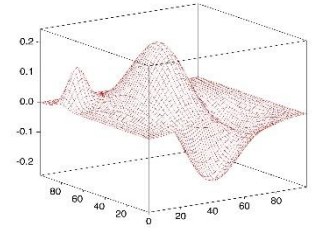
**Spatial Point Processes in R.** Functions in spatstat work on `ppp` objects. Use the `ppp()` function to convert matrices, dataframes, polygons, etc. to this data type.

To generate a CSR (Homogenous Poisson PP) in R, use the `rpoispp()` function:

```
#generate a CSR over the range (0,10) x (0,10) with mean  
1 per unit square  
pois1<-rpoispp(1,win=owin(xrange=c(0,10),yrange=c(0,10)))  
plot(pois1$x,pois1$y)
```

# Heterogenous Poisson Point Process

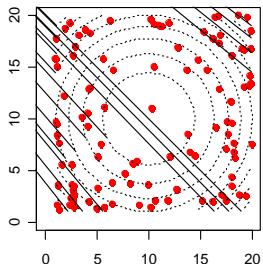
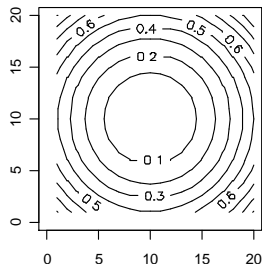
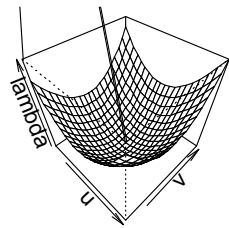
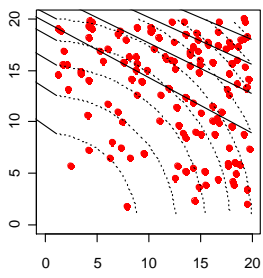
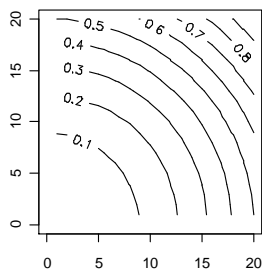
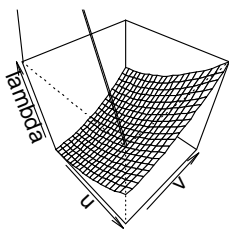
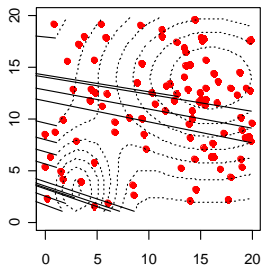
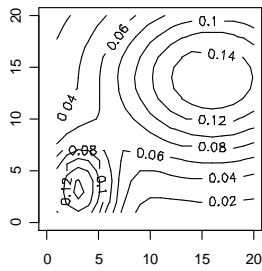
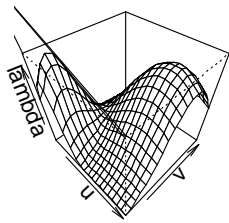
- Under the homogeneous Poisson model, the intensity function  $\lambda$  is **constant**.
- For many processes, a constant risk hypothesis is too restrictive – **the intensity varies over a region** according to population, gradients, etc.
- We define a **heterogenous Poisson point process** with **intensity function  $\lambda(s)$  (think mean)**, where  $s$  is any point in our study region  $R$ , using the criteria
  - 1) The number of events in any region  $A \subset R$  has a Poisson distribution with mean  $\int_A \lambda(s) ds$



- 2) Given that a total of  $X = N$  events occur in  $A$ , the locations of the events are a random sample of  $N$  events sampled proportional to  $\lambda(s)$

A heterogenous Poisson point process is **non-stationary** and **anisotropic** (not the same if you move and/or rotate in space!)

*The following page shows realizations of three heterogenous Poisson point processes with different intensity functions  $\lambda(s)$  :*





# First and Second Order Properties of a SPP

We usually summarize a probability distribution by talking about the **mean** and **variance**. There are analogues for a spatial point process

## First order Properties of a SPP : Intensity $\lambda(s)$

### Aside : Intensity vs. Density

- Statisticians usually talk about a **density** function  $f(s)$  over a region  $R$ .
- Density functions always have total area (volume) = 1  $\int_R f(s)ds = 1$
- Intensity functions do not integrate to 1 : they represent the mean number (expected number) of events per unit area. However, the following is true :

“Count per area”  $\dashrightarrow$   $\frac{\lambda(s)}{\int_R \lambda(s) ds} = f(s)$   $\dashleftarrow$  “Percentage per area”

“Total Count”  $\dashrightarrow$   $\int_R \lambda(s) ds$

I.e. you can convert between intensity functions and density functions!

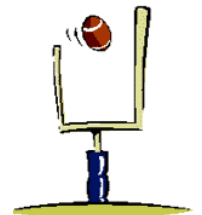
**Second order Properties:** measures relationships between pairs of points

.....

***SO : Let’s say you observe a point process. How do we estimate the intensity function  $\lambda(s)$  ?***

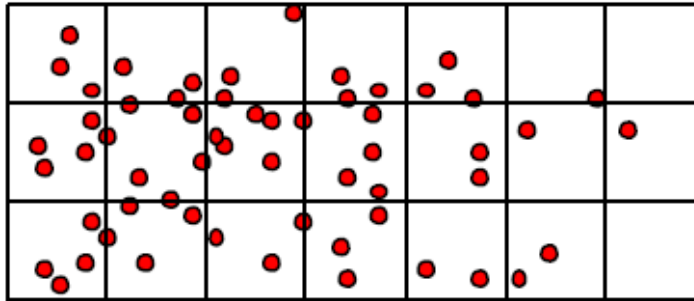


**Goal 2 : Estimating Intensity Functions**



## Quadrat Method : *(example from Kate Beard)*

- 1) Place a grid over the region  $R$  of interest



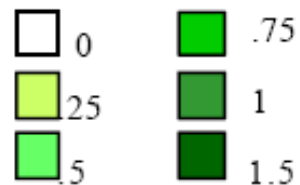
- 2) Count the number of points in each cell

3	3	3	3	3	1	0
4	6	6	4	2	1	1
4	4	3	3	2	2	0

- 3) Convert to estimated intensity  $\hat{\lambda}(s)$   
(assuming each cell is 4 square units)

<b>.75</b>	<b>.75</b>	<b>.75</b>	<b>.75</b>	<b>.75</b>	<b>.25</b>	<b>0</b>
<b>1</b>	<b>1.5</b>	<b>1.5</b>	<b>1</b>	<b>.5</b>	<b>.25</b>	<b>.25</b>
<b>1</b>	<b>1</b>	<b>.75</b>	<b>.75</b>	<b>.5</b>	<b>.5</b>	<b>0</b>

4) (*optional*) Make a density picture to see where points tend to lie



## Issues with quadrat estimation :

- Converts real data to an area value : there is a loss of spatial detail
- Quadrat size :
  - If quadrat is too small, we'll get mostly empty cells and cells with spikes
  - If quadrat is too large, all quadrats will likely have same value and we'll lose any features
  - ESRI Guide (p.82) suggests that a good size is



$$I = \sqrt{2 \frac{|R|}{N}} \quad \text{where}$$

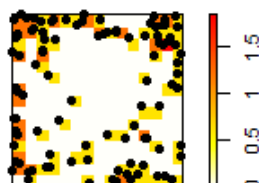
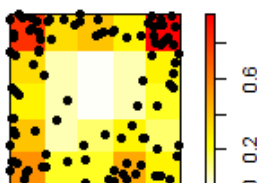
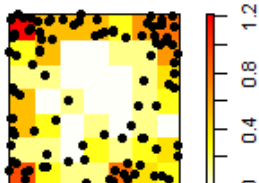
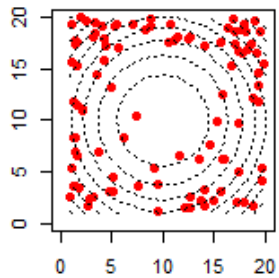
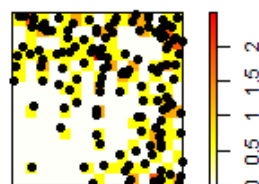
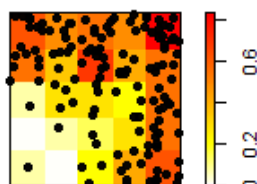
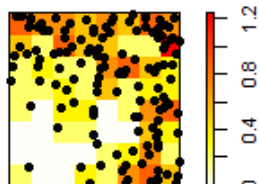
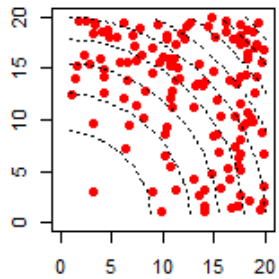
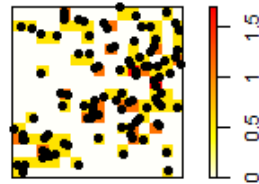
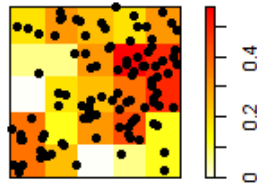
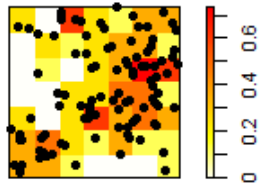
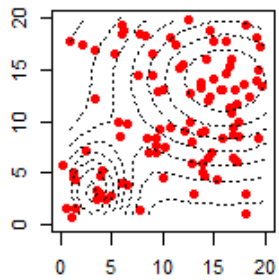
$I$  is the length of the side of the quadrat

$|R|$  is the size of the study area

$N$  is the number of events observed in  $R$

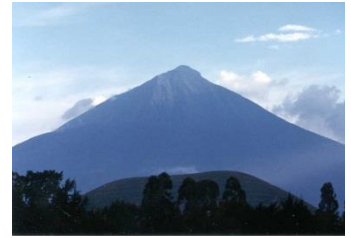


**Quadrat Estimation in R.** Use the function `quadratcount()` which gives the raw counts in each quadrat of the size you specify. This requires that your data be a `ppp` object. The function `image()` can be used to visually plot the result of these quadrats. However, you may prefer to simply add the numeric results of the quadrat counts to an existing point process plot. The first quadrat is the 'optimal' quadrat size, the other use a 5x5 and 15x15 grid. Link : <http://reuningscherer.net/fes781/rscripsts/quadrat.r.txt>

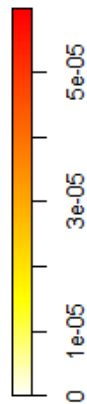
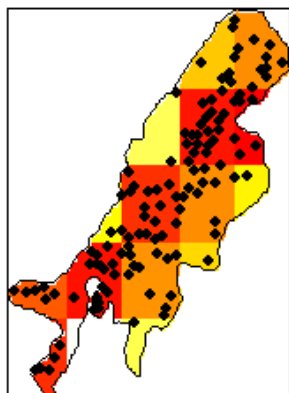




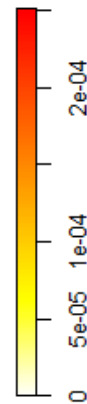
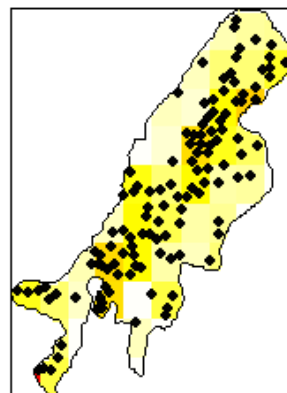
**Example : Uganda Volcano Locations.** This dataset has the locations of 120 volcanoes in the Bunyaruguru volcanic field in west Uganda. Below are the results of quadrat estimation for various grid sizes. R code is also in the file [quadrat.r.txt](#), data files are in `ugdat.dat` (crater locations) and `ugpoly.dat` (polygon of Uganda)



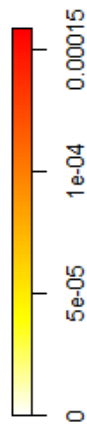
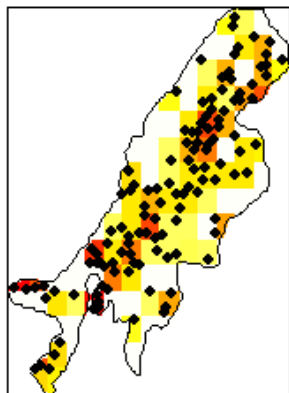
Grid is 5 x 5



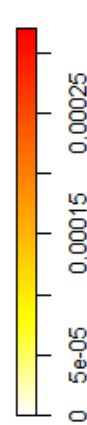
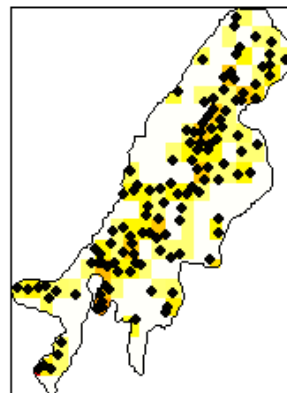
Grid is 10 x 10



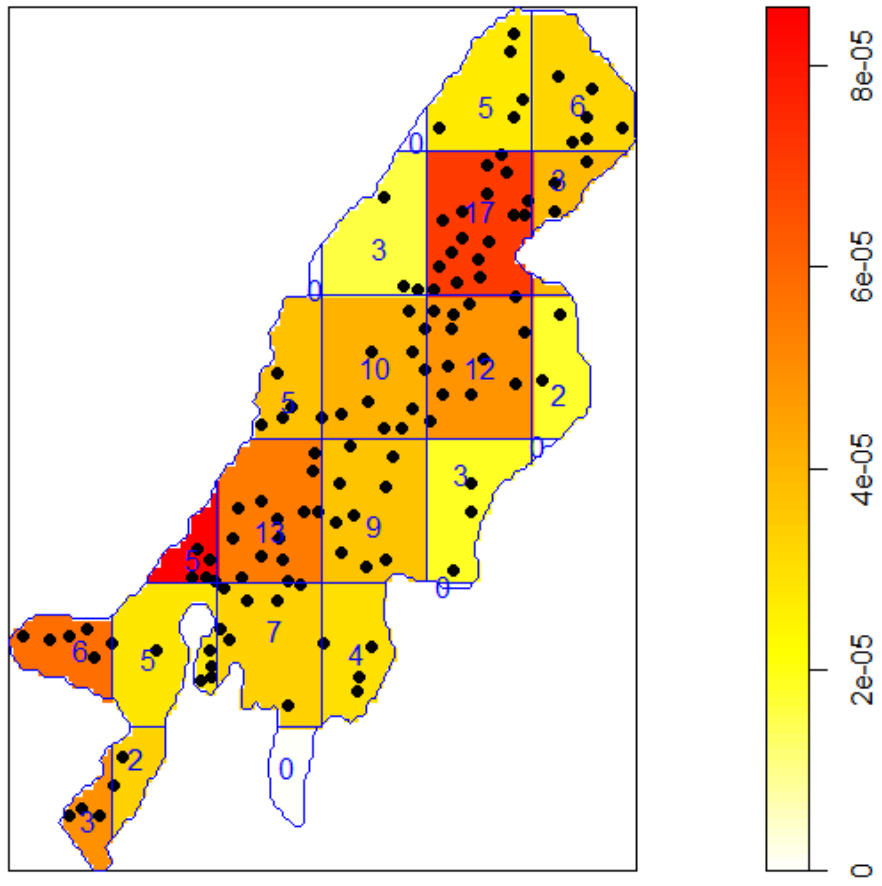
Grid is 15 x 15



Grid is 20 x 20



### Quadrat Count for Uganda Volcanos with Grid Counts

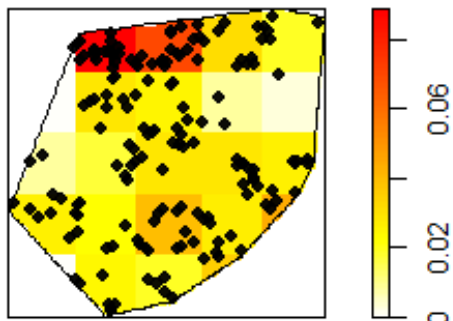


**Example : Juvenile Offenders in Cardiff.** This dataset has the locations of homes of juvenile offenders living in a housing area in Cardiff, Wales in 1971. R code is also in the file

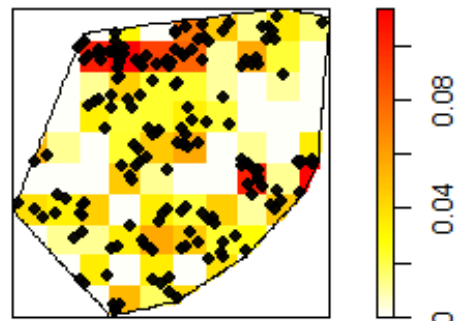


[quadrat.r.txt](#), data files are in `carddat.dat` (house locations) and `cardpoly.dat` (polygon of area). Data from Melvin Hooten at Utah State U.

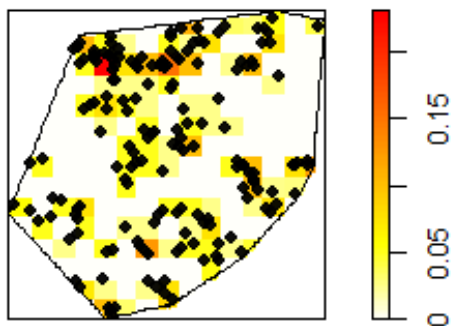
Grid is 5 x 5



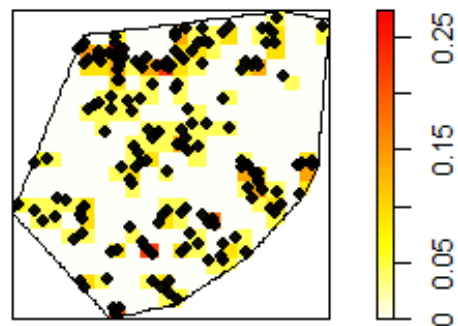
Grid is 10 x 10



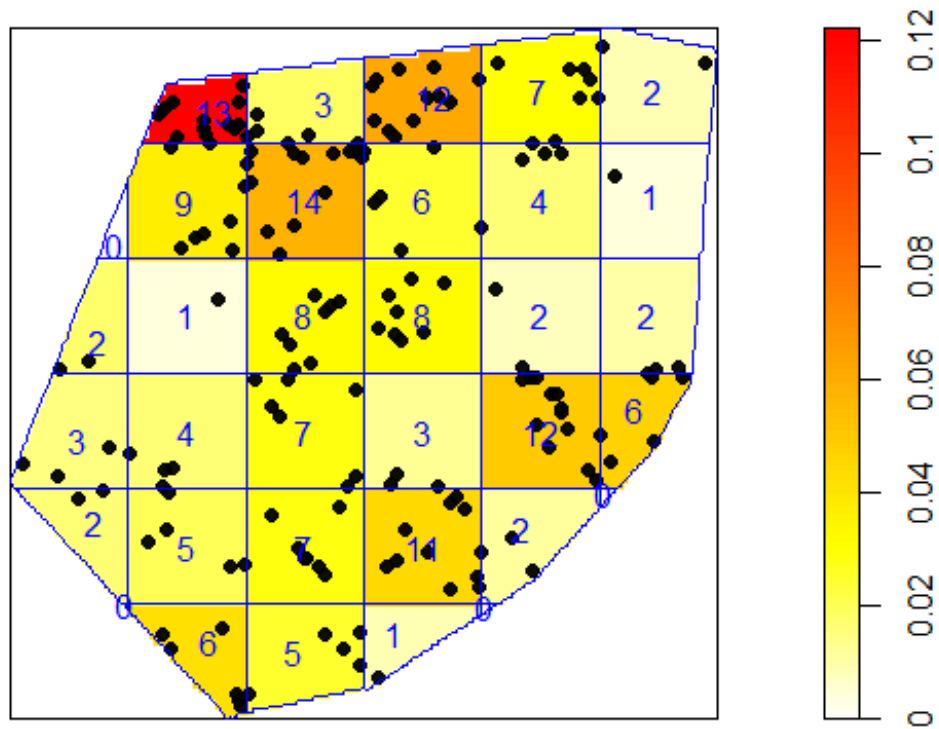
Grid is 15 x 15



Grid is 20 x 20



## Quadrat Count for Cardiff Juv. Off. with Grid Counts

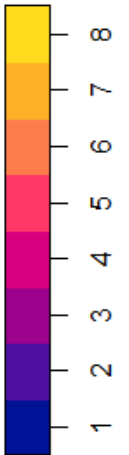
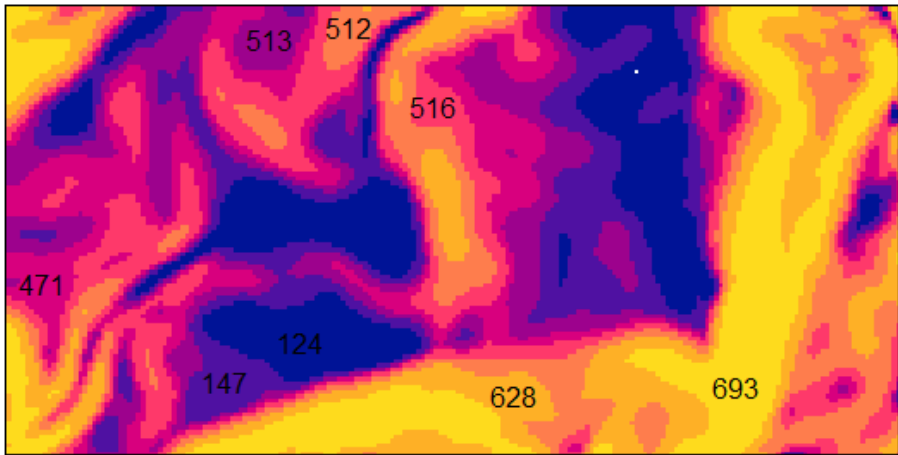
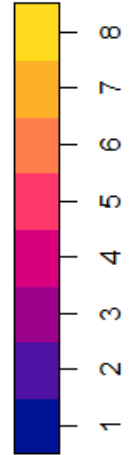
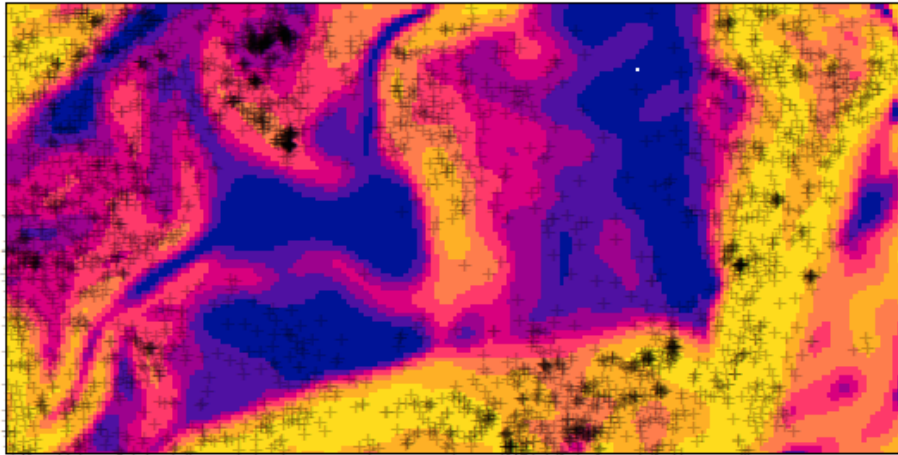


## Non-square Quadrats (Baddeley, section 13.2)

There is no particular reason quadrats have to be rectangles/squares. The only real requirement is that the regions have equal areas.

Baddeley suggests using a potentially related covariate to create quadrats of equal area by dividing the region according to quantiles of the covariate. Then, perform quadrat counts in each quantile

***Example : Tropical Rainforest Data (bei).*** We divide slope into 8 quantiles and use the resulting tessellated plot to define 8 quadrats. A plot with tree locations suggests that tree location and slope may be related. Quadrat counts show that counts clearly differ with slope quantiles (otherwise, counts would all be approximately the same)





# Kernel Smoothing



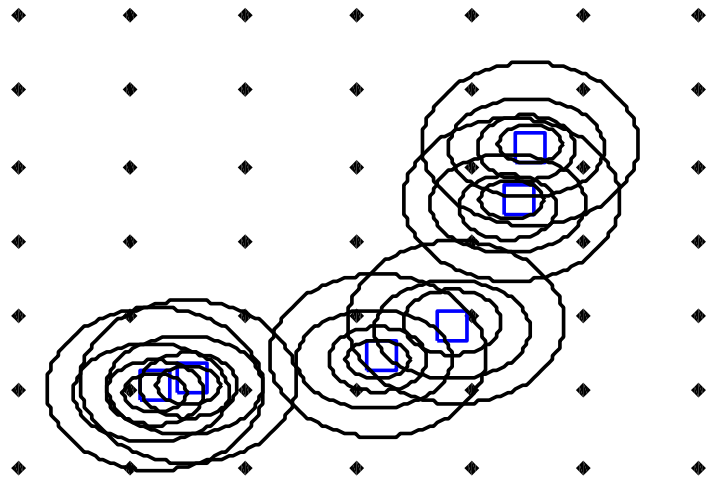
Kernel Smoothing is a quick way to estimate an intensity (or density) function based on a point process.

Here is the basic idea :

- Choose a **kernel** (a small probability blob, a bit of putty)
  - Often a bivariate normal distribution
  - Also, Uniform (constant height), and Triangle, Tri-weight, etc
- Choose a **radius** (basically, how much to spread around the influence of each observation – small radius equals local influence only, large equals wide influence). This is often called the **bandwidth**.



- Put kernel on each observed **event**.
- Get the averaged probability on all desired grid points based on probability blobs
- This gives a non-parametric estimate of the density/intensity function.



**Math** : Mathematically, a **kernel intensity estimate** at a particular point  $s$  is

$$\hat{\lambda}(s) = \frac{1}{b^2} \sum_{i=1}^N \text{kernel} \left( \frac{s - s_i}{b} \right)$$

where

$b$  is the bandwidth

$s_i$  are the observed event locations

$\text{kernel}(\ )$  is the kernel function being used.

Often, a kernel **density** estimate is used instead :

$$\hat{f}(s) = \frac{1}{Nb^2} \sum_{i=1}^N \text{kernel} \left( \frac{s - s_i}{b} \right)$$

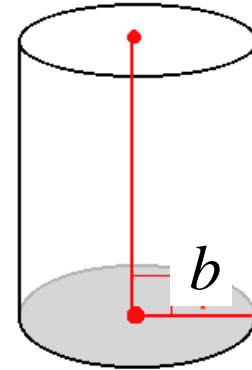
**SO** : Kernel smoothing requires two choices : **kernel** and **bandwidth**

# Kernels

Lots of possibilities – here are some common ones

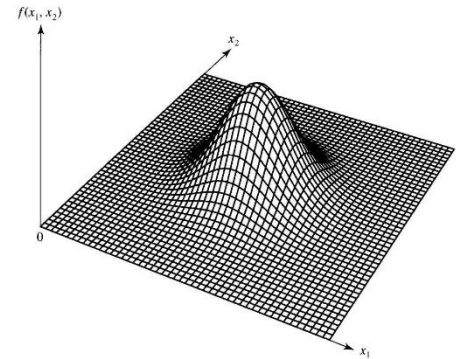
## Uniform

$$\text{kernel}(s) = \begin{cases} \frac{1}{\pi b^2} & |s - s_i| \leq b \\ 0 & \text{otherwise} \end{cases}$$



## Gaussian / Normal

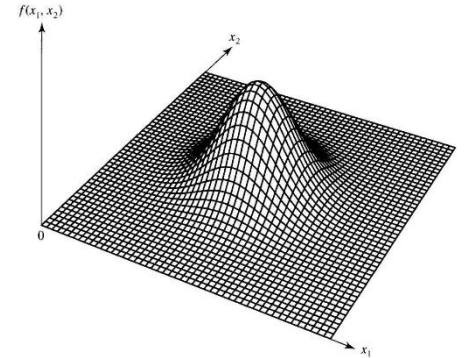
$$\text{kernel}(s) = \frac{1}{2\pi|\Sigma|^{1/2}} e^{-\frac{(s-s_i)' \Sigma^{-1} (s-s_i)}{2}}$$



*(the covariance matrix incorporates the bandwidth  $b$ )*

### **Quartic (bi-weight)**

$$\text{kernel}(s) = \begin{cases} \frac{3}{\pi} (b - (s_i - s))^2 & |s - s_i| \leq b \\ 0 & \text{otherwise} \end{cases}$$



*(basically, a Gaussian-like kernel that gets truncated to make computation easier)*

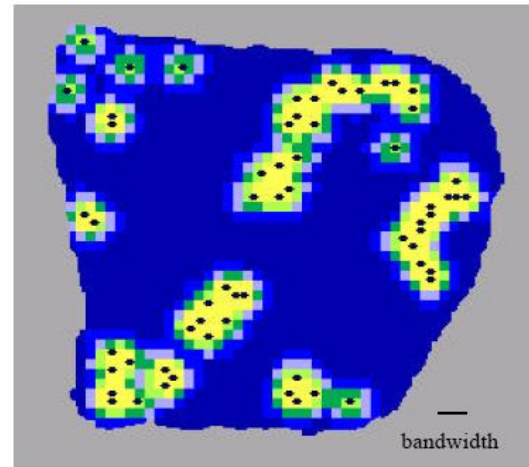
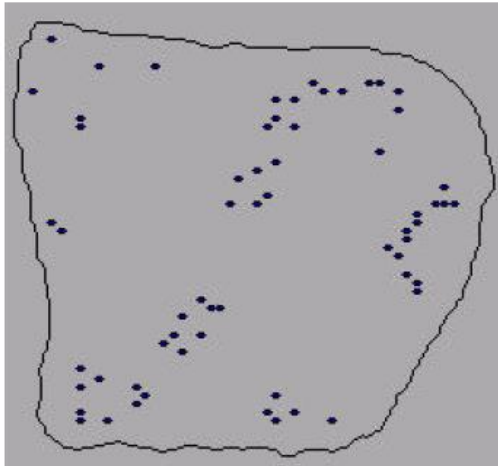
**In practice : the choice of kernel has little effect on the final result. Much more important is the**

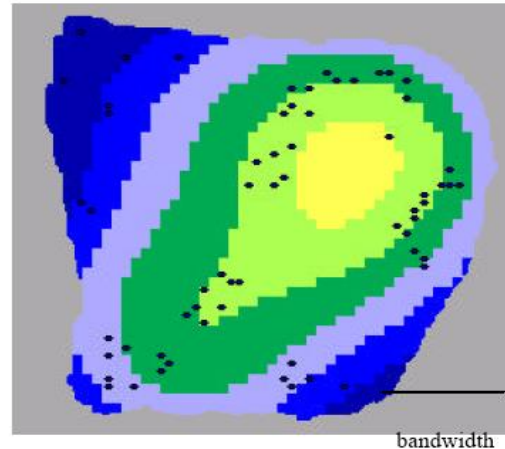
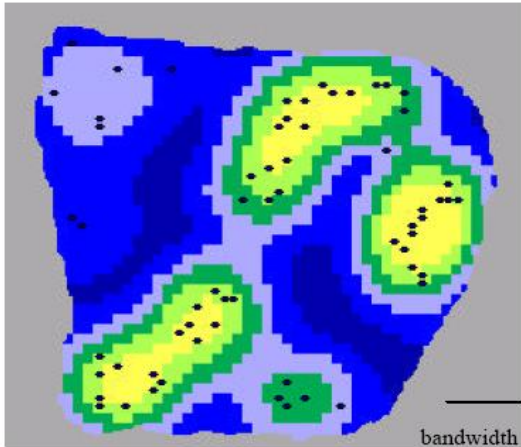
# Bandwidth

**Small bandwidths** make spiky estimates of  $\hat{\lambda}(s)$  (with spikes on the observations)

**Large bandwidths** tend to make everything look uniform and can smooth out interesting features

***Example from Kate Beard's Notes :***





**Note** – there are several recent papers (*two online in materials folder*) discussing **Adaptive Kernel Smoothing**, where the bandwidth changes depending on the density of observations (i.e. wider bandwidth if there are few points in a region)

$$\hat{\lambda}(s) = \sum_{i=1}^N \frac{1}{b(s_i)^2} \text{kernel} \left( \frac{s - s_i}{b(s_i)} \right)$$

where  $b(s_i)$  the bandwidth is now a function that depends on the presence of other events in the neighborhood of  $s_i$ .

---

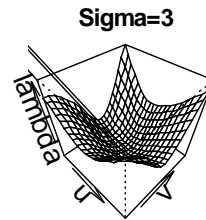
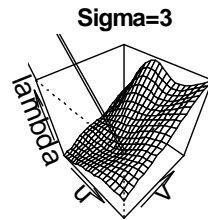
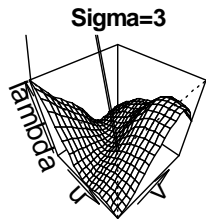
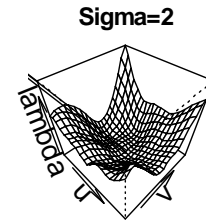
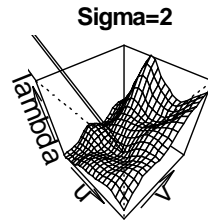
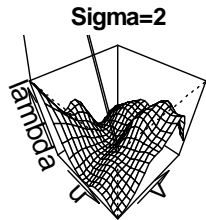
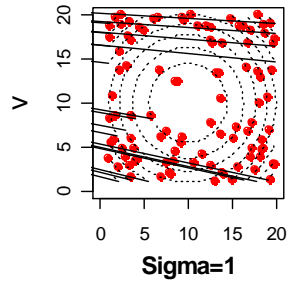
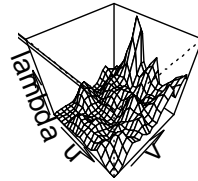
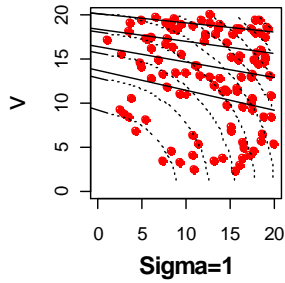
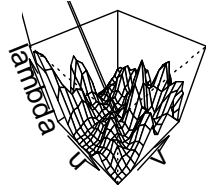
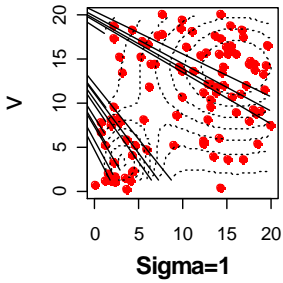


**Kernel Smoothing in R.** There are many functions for kernel smoothing in R. An older function is `ksmooth()` which was replaced by `density()`. In the `spatstat` package, there is a function `density.ppp()`. `density()` allows for several types of kernels while `density.ppp()` uses only Gaussian kernels. The function `kernel2d()` in the `splancs` package also works nicely. Also, there is a new package called `spatialkernel` (about a year old), **AND** another package simply called `ks`.

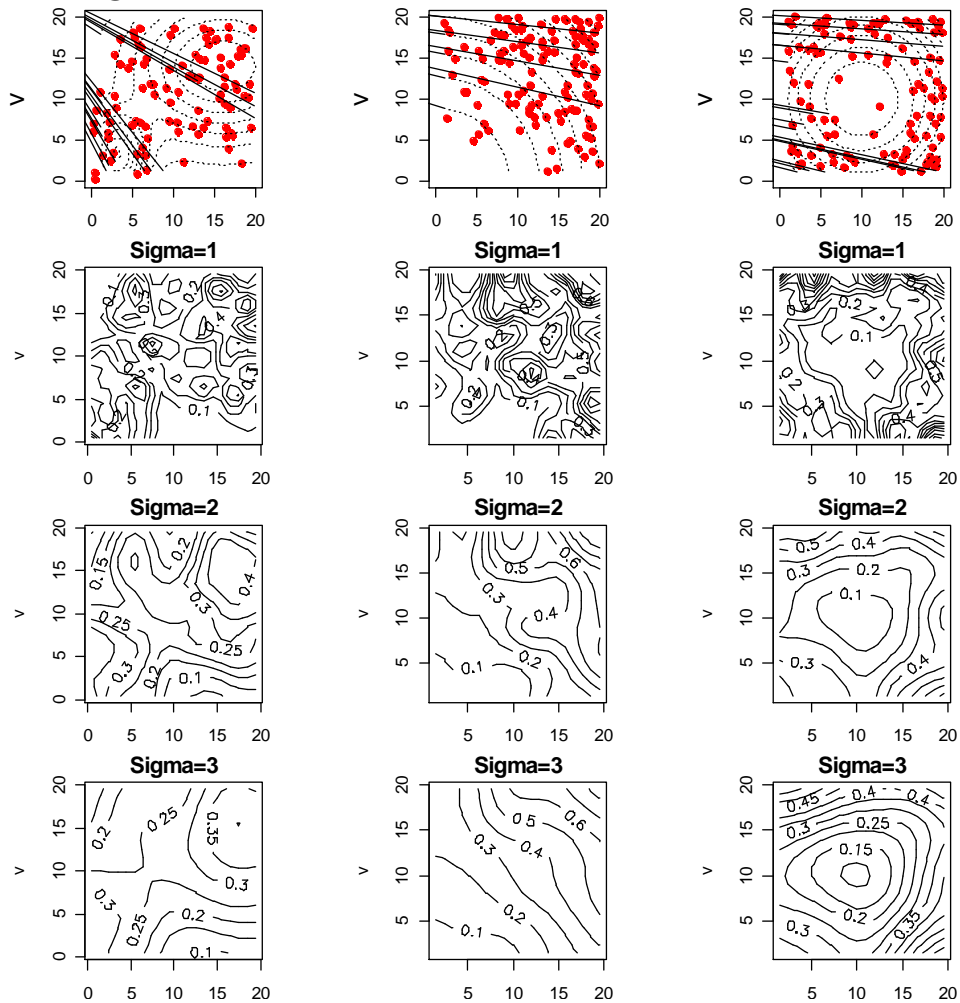
To see the results of your smoothing, you can use `persp()`, `contour()`, or `image()`. I've put the code for all the plots on the following pages online as [kernelsmooth.r.txt](http://reuningscherer.net/fes781/rscripts/kernelsmooth.r.txt). Link : <http://reuningscherer.net/fes781/rscripts/kernelsmooth.r.txt>

The next page contains kernel smoothed images of our three heterogenous Poisson point processes for several bandwidths (all use Gaussian kernel)





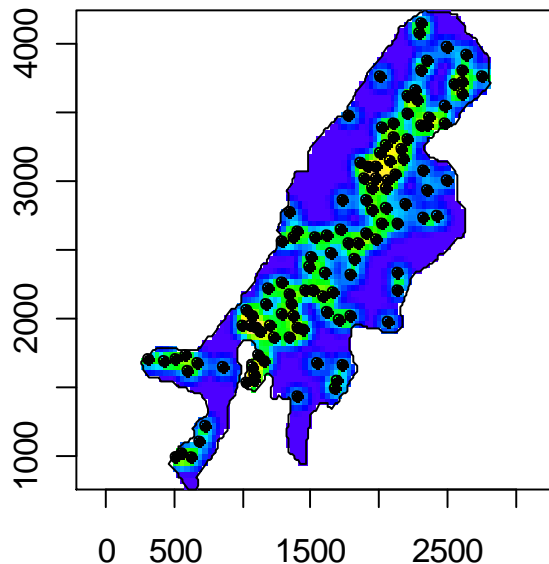
Same thing but with contour plots :



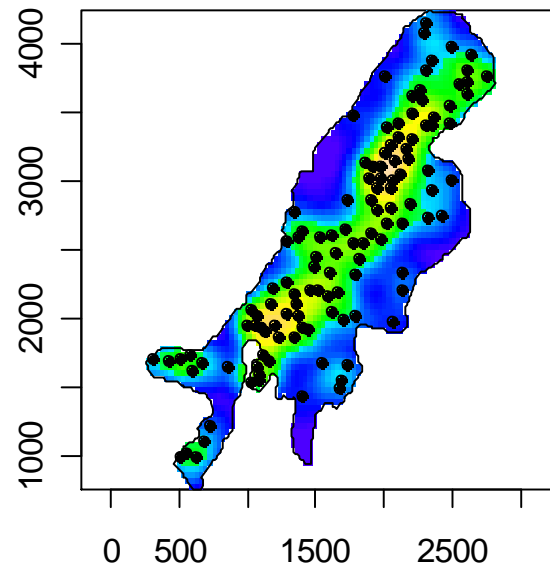
**Example : Uganda Volcano Locations.** *R* code is also in the file [kernelsmooth.r.txt](#)  
Notice that as the bandwidth increases, the plot gets smoother (eventually too smooth!)



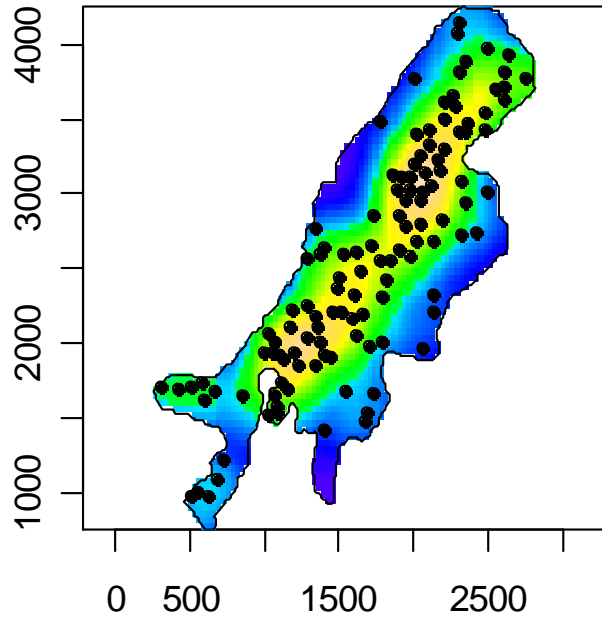
**Bandwidth = 100 , Grid Size = 80**



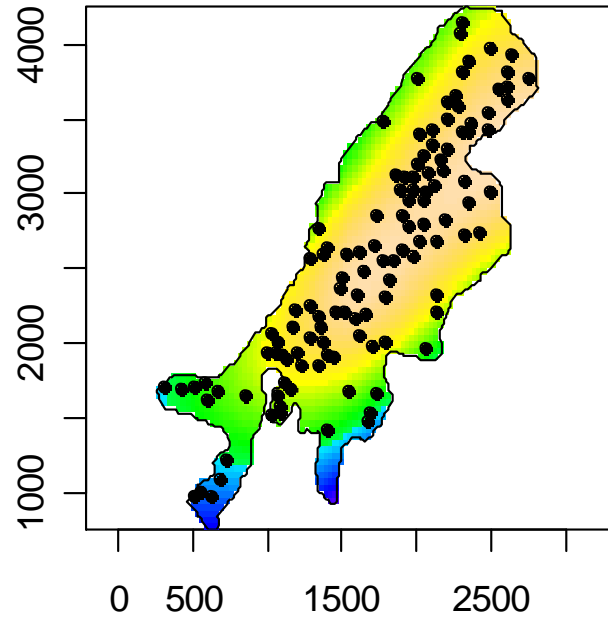
**Bandwidth = 200 , Grid Size = 80**



**Bandwidth = 300 , Grid Size = 80**

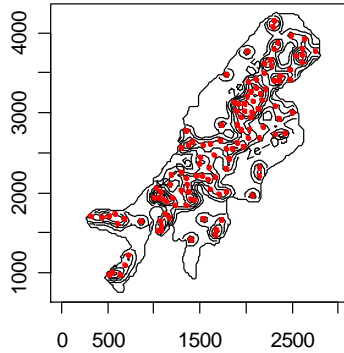


**Bandwidth = 1000 , Grid Size = 80**

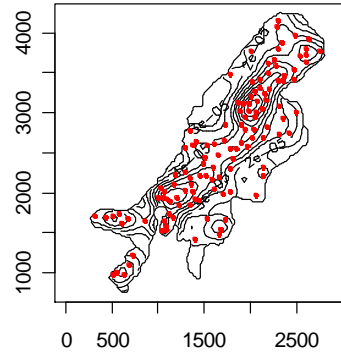


*Same idea, but with contour lines:*

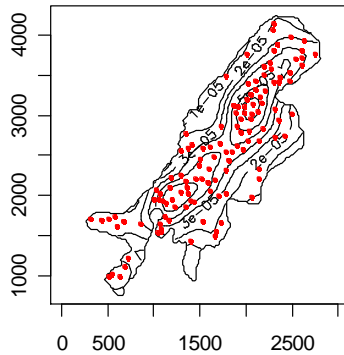
**Bandwidth = 100 , Grid Size = 80**



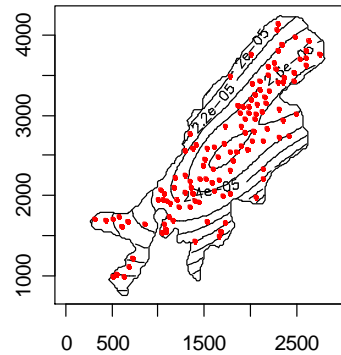
**Bandwidth = 200 , Grid Size = 80**



**Bandwidth = 300 , Grid Size = 80**



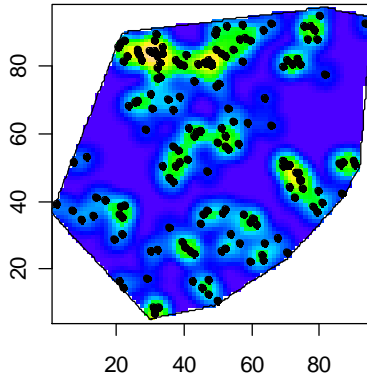
**Bandwidth = 1000 , Grid Size = 80**



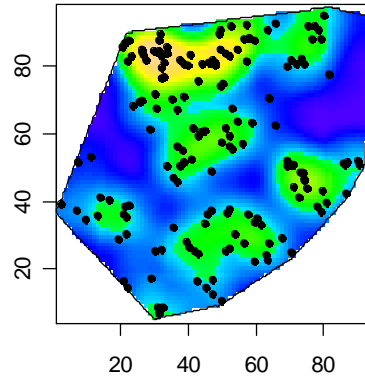
**Example : Juvenile Offenders in Cardiff.** *R* code is also in the file [kernelsmooth.r.txt](#)



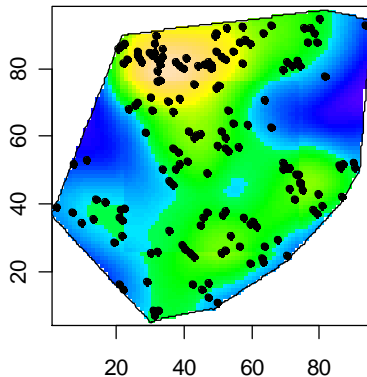
**Bandwidth = 5 , Grid Size = 80**



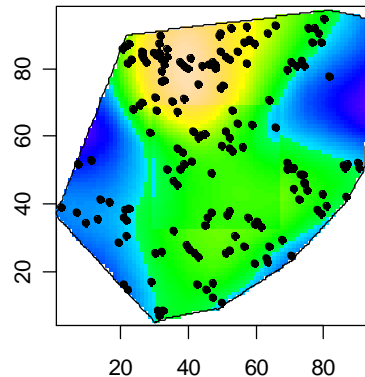
**Bandwidth = 10 , Grid Size = 80**



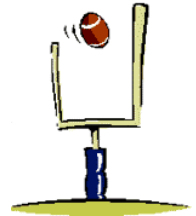
**Bandwidth = 15 , Grid Size = 80**



**Bandwidth = 20 , Grid Size = 80**



## Goal 3 : Second Order Properties of a SPP : Relative Position / Relationships between events



Several measures exist : Nearest neighbor methods (G, F).  
However, most common / popular is

### Ripley's K-function

- Known as the K-function or reduced second moment measure (Ripley 1977, Diggle 1983),
- Defined as

$E( )$  means Expected Value (i.e. average)

$$K(h) = \frac{E(\# \text{events within } h \text{ of a random event})}{\lambda}$$

for all positive distance (**spatial lag**)  $h$

Ripley (1977) showed that  $K(h)$  is equivalent to calculating the **Variance of the number of points in any subregion  $A$**

***So, what should  $K(h)$  look like?***



**CSR :**  $K(h) = \pi h^2$  (Area within  $h$  of any point is just  $\pi h^2$ , intensity is a constant  $\lambda$ , so expected number of events is  $\lambda|A| = \lambda\pi h^2$ )

**Regular :**  $K(h) < \pi h^2$  for  $h$  less than regularity interval

**Clustered :**  $K(h) > \pi h^2$  for  $h$  less than cluster size



## Estimating $K(h)$ (i.e. how to calculate $\hat{K}(h)$ )

Basically, replace expectation with **AVERAGE** :

$$\hat{K}(h) = \frac{1}{\hat{\lambda}N} \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N I(d(i, j) < h)$$

where  $I(\ )$  is the indicator function :  $I(\ ) = \begin{cases} 1 & d(i, j) < h \\ 0 & \text{otherwise} \end{cases}$

and  $\hat{\lambda} = \frac{N}{|R|}$  (number of points divided by size of region)

**Basically,  $\hat{K}(h)$  is just an average of the number of points within  $h$  of each observed event.**

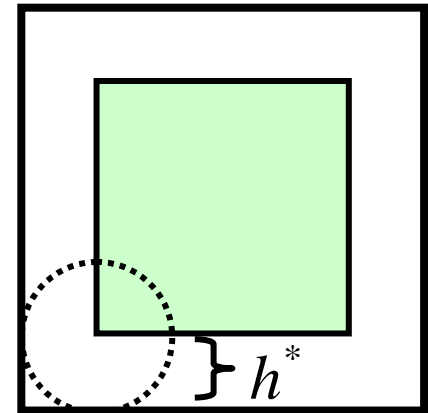
**Problem** : what to do when we hit the edge?

When  $h$  contains area outside our study region, we will underestimate  $K(h)$  .



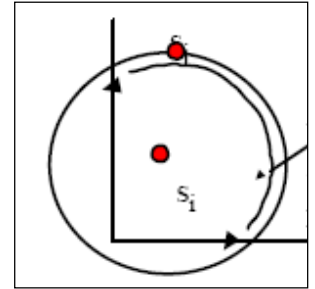
**Common corrections** :

**1) Boundary Correction** : only compute  $K(h)$  using events that are completely within  $h^*$  of the edge, and only compute  $K(h)$  for  $h < h^*$



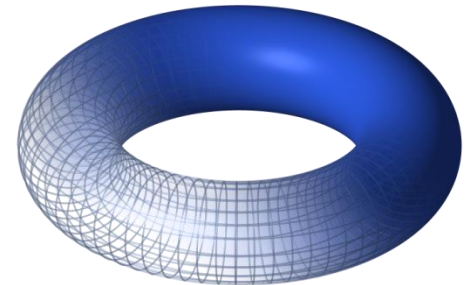
## 2) Edge Correction :

$$\hat{K}(h) = \frac{1}{\hat{\lambda}} \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N \omega_{ij} I(d(i, j) < h)$$



$\omega_{ij}$  is amount of circle at point  $i$  with radius  $d(i, j)$  that is within the study region  $R$

**1) Taurus Correction :** I won't give equation here, but I'll describe the idea (it's a wrapping thing)



$R$  calculates several others . . . .

## Plots using $K(h)$

- Under CSR,  $K(h) = \pi h^2$  **which is a parabola!**
- It would be easier to note departures from CSR if what was 'normal' was a **straight line** (not a parabola)
- **SO** : a plot of  $h$  versus  $\sqrt{\frac{\hat{K}(h)}{\pi}} - h$  **should be a flat line at zero!**
- Sometimes defined as the  $\hat{L}(h) = \sqrt{\frac{\hat{K}(h)}{\pi}} - h$  function (Ripley, 1979)

**SO** : If  $\hat{L}(h)$  looks like a line, our process shows CSR. If not . . . .

## Simulated Bounds for $L(h)$

It is possible to use a Monte Carlo simulation to estimate bounds for  $L(h)$

- 1) Generate a CSR of  $N$  events over the study region  $R$ .
- 2) Calculate  $L(h)$
- 3) Repeat 1) and 2) a 'bunch of' times : maybe 100, maybe 1000. Note however that  $L(h)$  is computationally expensive.
- 4) Use the randomly generated bounds to compare  $L(h)$  for the observed spatial point process.



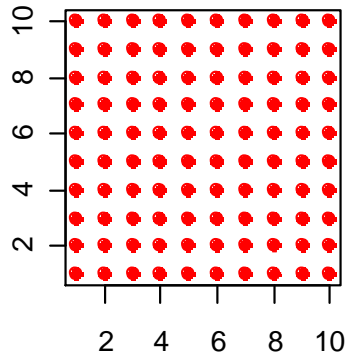
**Ripley's K in R.** The `spatstat` package uses the functions `Kest()` to calculate Ripley's K-function with various corrections. The function `envelope()` does the Monte Carlo simulation of the bounds for  $K(h)$ . There is a function called `Lest()`, but it doesn't subtract  $h$ , so the result is a line at 45 degrees. SO : I've written a modest (and improvable) function in `Lfunction.R` called `L()`. It requires as inputs the outputs from `envelope()` and `Kest()`. Also, I put in a similar function called `Kplot()`. All functions are in the file :

<http://reuningscherer.net/fes781/rscripsts/SpatialPPFuncs.R.txt>

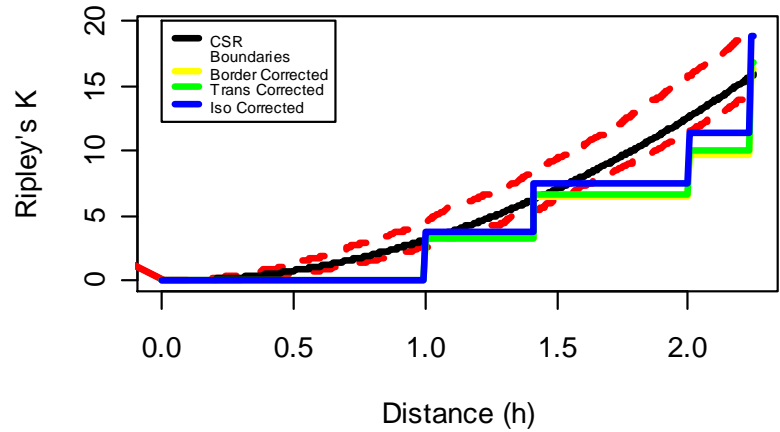
*The plots on the next page show the functions  $K(h)$  and  $L(h)$  for several simulated point processes - code is here :*

<http://reuningscherer.net/fes781/rscripsts/lfunction.r.txt>

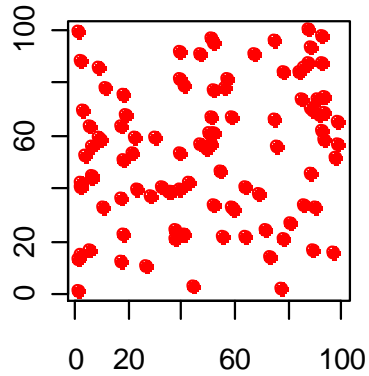
### Regular Grid



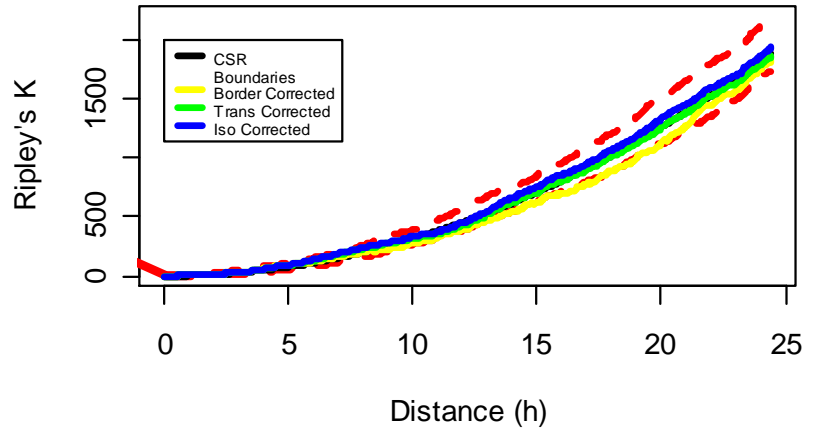
### Ripley's K-plot for Regular Grid



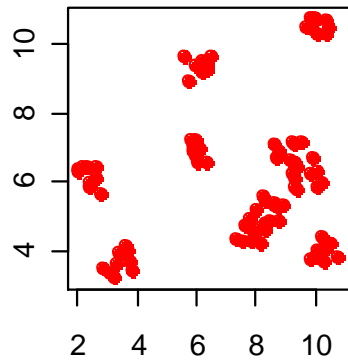
### CSR



### Ripley's K-plot for Randomly Generated CSR

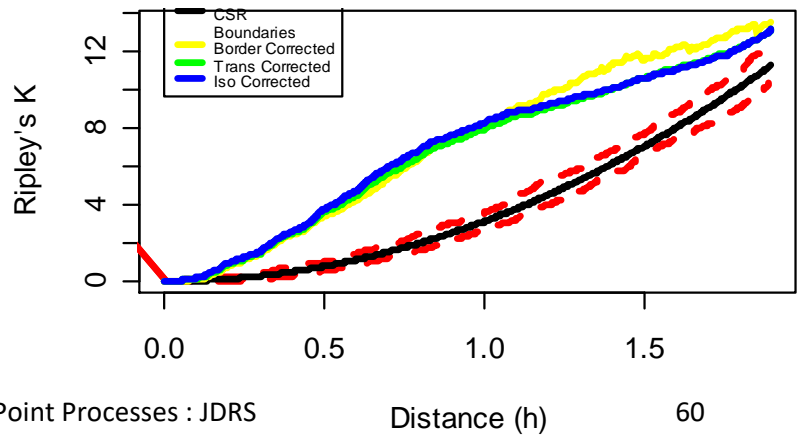


### Clustered Process



FES781b

### Ripley's K-plot for Clustered Process



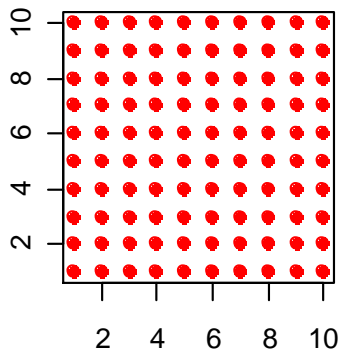
Spatial Point Processes : JDRS

Distance (h)

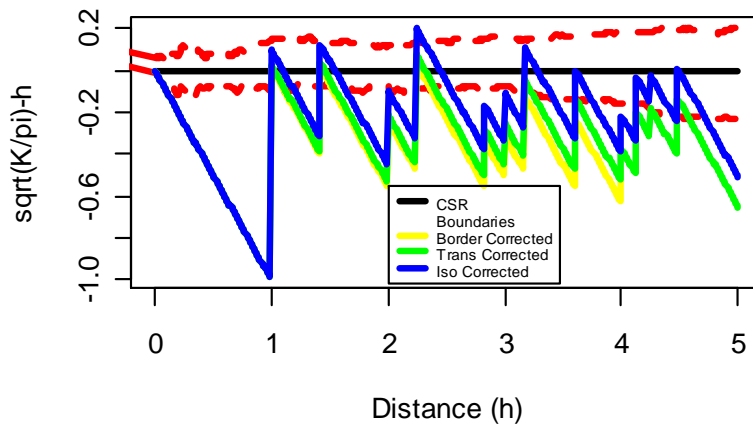
60



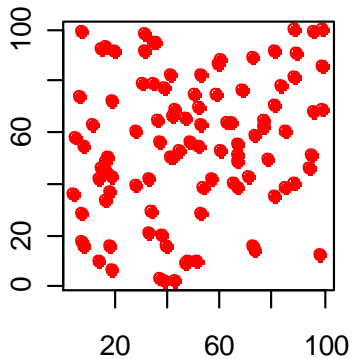
### Regular Grid



### L-plot for Regular Grid

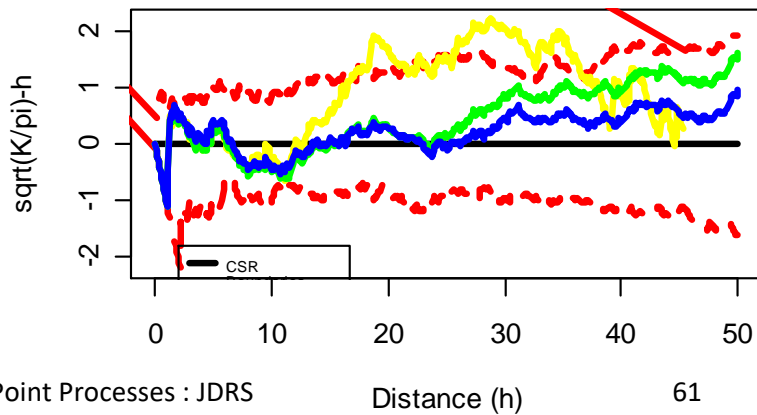


### CSR



FES781b

### L-plot for Randomly Generated CSR



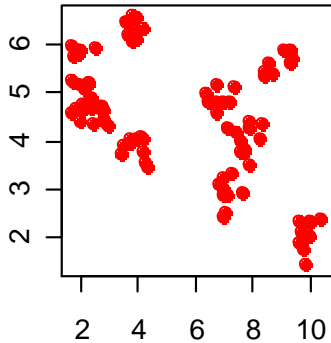
Spatial Point Processes : JDRS

Distance (h)

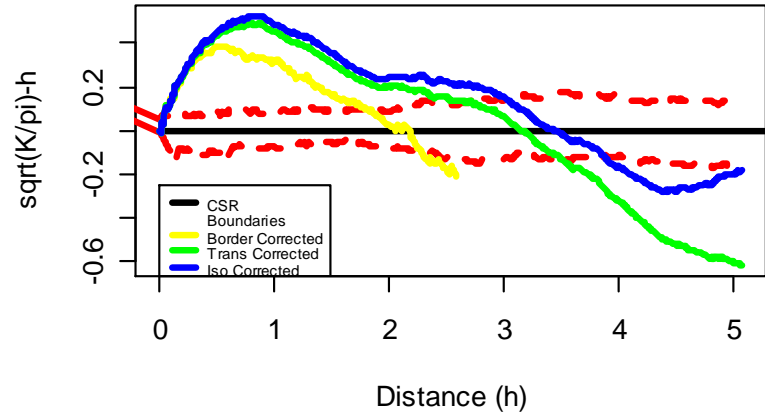
61

Distance (h)

**Clustered Process**

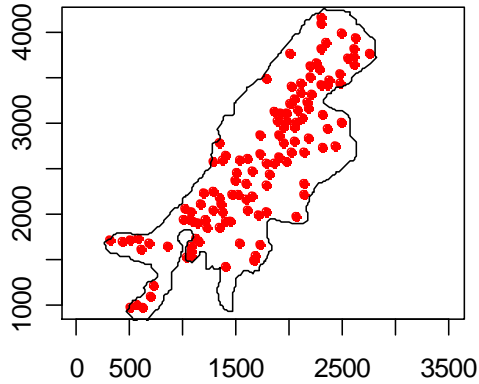


**L-plot for Clustered Process**

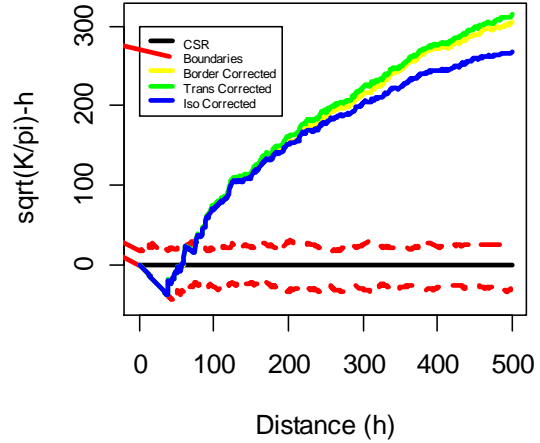


*Finally, here is  $L(h)$  for the volcano and delinquency data! In both cases, it appears that these are clustered processes, although for the volcano data, at distances less than 50 meters, it appears to be a regular process (no volcanos on top of each other)*

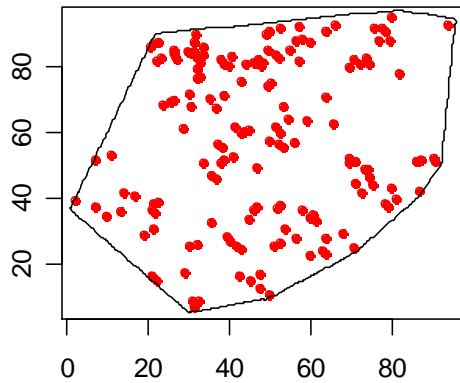
### Uganda Volcanos



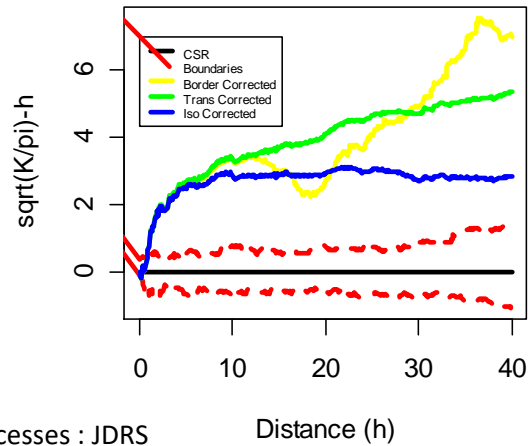
### L-plot for Uganda Volcanos



### Cardiff Delinquents

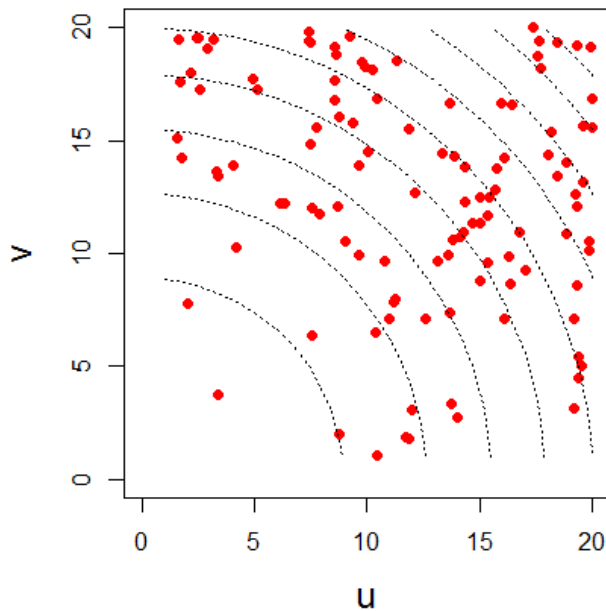


### L-plot for Cardiff Delinquents

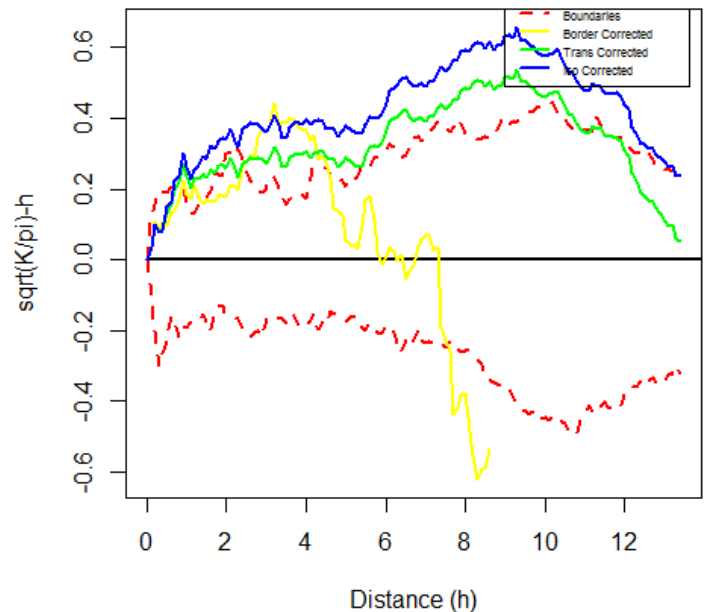


# Clustering – Mean or Variance?

A challenge throughout spatial stats is deciding what patterns are due to spatial correlation, and what patterns are due to changing mean. Here is the L-plot (which shows clustering) when applied to the example where the mean increased from lower left to upper right.



L-plot for Example of Int. Mean as Variance



## So Far :

- 1) Investigated **first order properties** of spatial point processes (i.e. mean/average)
- 2) Compared means to a **completely spatially random** process (CSR)
- 3) Began to investigate second order properties of SPP's (**Ripley's  $K(h)$  /  $L(h)$  functions**)

Note that Ripley's  $K(h)$  function is an **expected value** of the number of points within a distance  $h$  of an observed point.

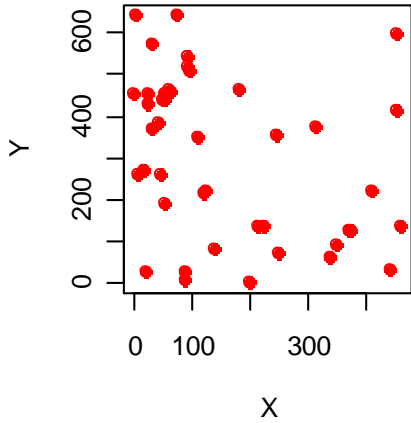
*Before we move on, another . . .*

**Example : Maine Tree Data.** This is data Tim collected on different species at a site in Maine – the location of each instance of 7 tree species was noted over a rectangular region.

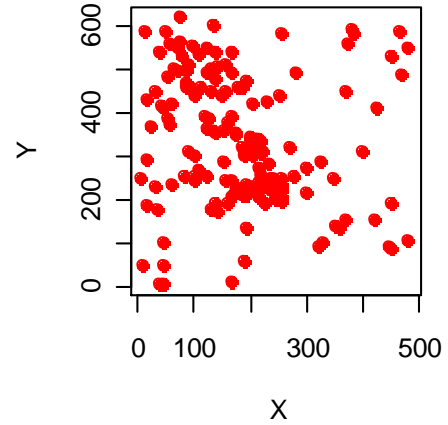


Here are a couple of examples of species, the  $L(h)$  function, and the result of a kernel smoothing. Clustering is evident, especially for White Pine. Code is online as first part of [GandFFunctions.R.txt](http://reuningscherer.net/fes781/rscripsts/gandffunctions.r.txt) Link : <http://reuningscherer.net/fes781/rscripsts/gandffunctions.r.txt>

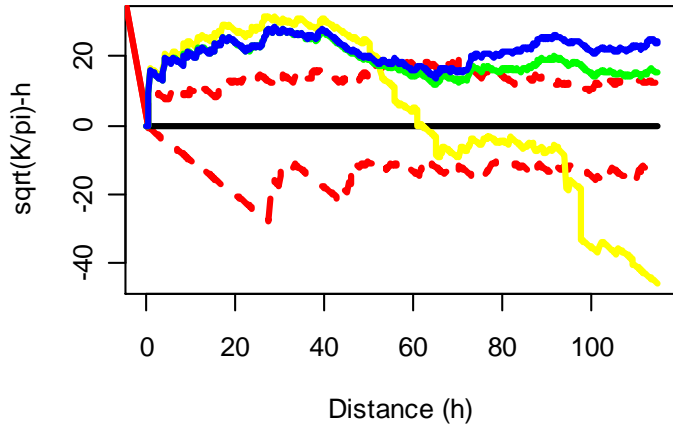
### Paper Birch



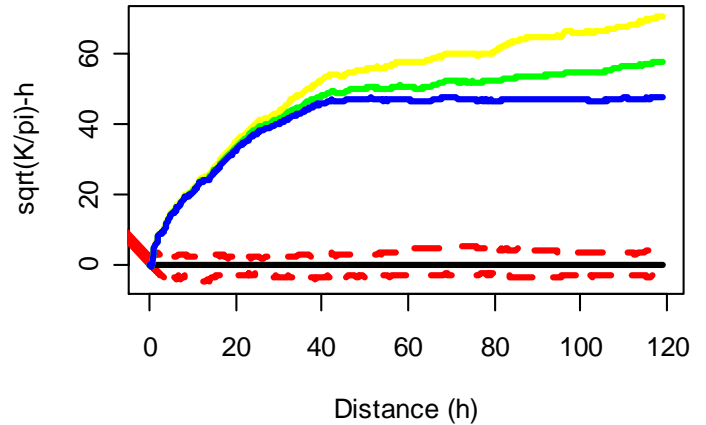
### White Pine



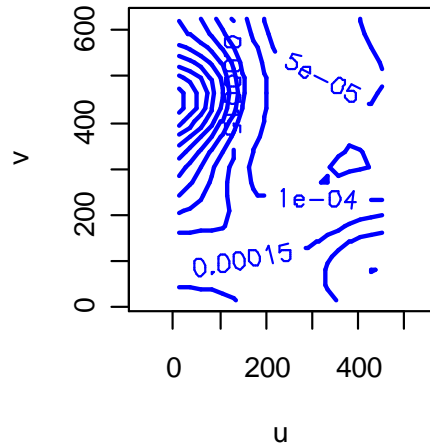
### L-plot for Paper Birch



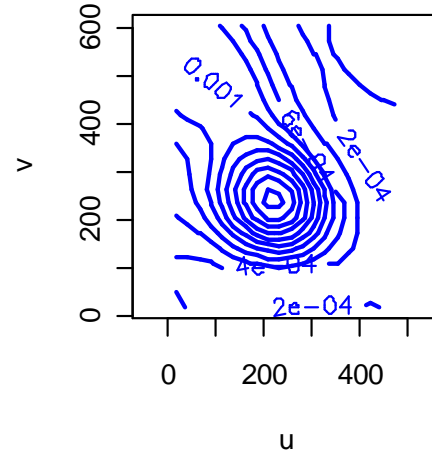
### L-plot for White Pine



**Paper Birch Sigma=70**



**White Pine Sigma=70**





Ripley's  $K(h) / L(h)$  functions examine the estimated number of points expected at fixed distances around an observed location. However, many other methods were originally created to investigate second order properties of spatial point patterns using . . . .

## Nearest Neighbor Methods

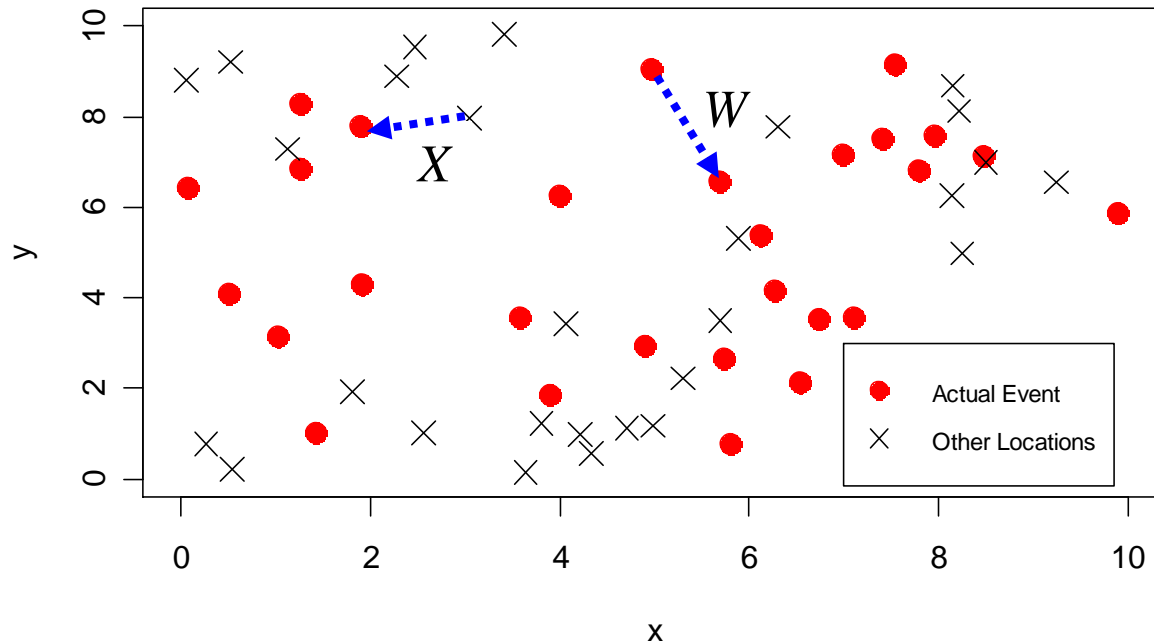
*Used to investigate second order properties of a PP.  
However, first we need to discuss*

**Nearest Neighbor Distances : two types**



**W-event** : distance from an **event** to the **nearest event**. For a set of  $N$  events, these are the values  $\{w_1, w_2, \dots, w_N\}$ .

**X-event** : distance from a randomly-chosen **non-event** location to the **nearest event**. For a randomly selected set of  $M$  locations, these are the values  $\{x_1, x_2, \dots, x_M\}$ .



We can use these events to define functions which estimate the **distribution of nearest neighbor distances**.



## **G-function**

$G(r)$  is defined as

$$G(r) = \Pr(\text{Dist}(\text{random event to nearest event}) \leq r)$$

This is the **cumulative probability distribution of nearest neighbor distances between events**.

We estimate this function with

$$\hat{G}(r) = \frac{\#(w_i \leq r)}{N} = \frac{\sum_{i=1}^N I(w_i \leq r)}{N}$$

where  $I(\cdot)$  is the indicator function = 1 if true, 0 otherwise.  
*Basically, this is the percent of observed nearest neighbor distances that are less than  $r$*

**Note** that  $G(r) = 0$  when  $r = 0$ , and  $G(r) = 1$  as  $r \rightarrow \infty$

**NOW** : if we have CSR, what should  $G(r)$  be? Matern (1971) showed that under CSR (*in 2-dimensions*),

$$G(r) = 1 - e^{-\lambda \pi r^2}, \quad r \geq 0$$

## Reason :

- 1) Assume we have CSR which means that the number of points in some area  $A$  has a Poisson distribution with some mean  $\lambda$  per unit area – or mean  $\lambda A$  in an area of size  $A$ .
- 2) Recall that for a Poisson distribution, the probability of having  $x$  events in area  $A$  is

$$\Pr[X = x] = \frac{\lambda A^x}{x!} e^{-\lambda A} \quad \text{for } x = 0, 1, 2, 3, \dots$$

- 3) This means that the probability of having no events within radius  $r$  of any location is

Pr(N. Neighb. is more than  $r$  units away)

$$= \Pr[X = 0] = \frac{(\lambda \pi r^2)^0}{0!} e^{-\lambda \pi r^2} = e^{-\lambda \pi r^2}$$

- 4) **SO** : This means that

Pr(N. Neighb. is **LESS** than  $r$  units away)

$$= G(r) = 1 - e^{-\lambda \pi r^2}$$

This can be used for comparison of  $\hat{G}(r)$  which is typically plotted vs.  $r$ .

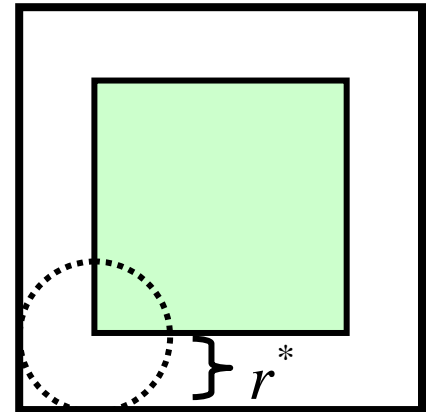
**Problem** : what to do when we hit the edge?



When  $r$  contains area outside our study region, we **overestimate**  $G(r)$  (nearest neighbor might be outside study region)

**Common corrections** :

- 1) Boundary Correction** : only compute  $\hat{G}(r)$  using events that are completely within  $r^*$  of the edge, and only compute  $\hat{G}(r)$  for  $r < r^*$



- 2) Edge Correction** : **R** uses Kaplan-Meier correction for the edge effect.

*Baddeley, A.J. and Gill, R.D. Kaplan-Meier estimators of interpoint distance distributions for spatial point processes. Annals of Statistics , 25, (1997) 263-292.*

Cressie (p.614) lists other corrections. **R** provides 5 corrections by default.

## Simulated Bounds for $G(r)$

It is possible to use a Monte Carlo simulation to estimate bounds for  $G(r)$

- 1) Generate a CSR of  $N$  events over the study region  $R$ .
- 2) Calculate  $\hat{G}(r)$
- 3) Repeat 1) and 2) a 'bunch of' times : maybe 100, maybe 1000. Note however that  $\hat{G}(r)$  is computationally expensive.
- 4) Use the randomly generated bounds to compare to  $\hat{G}(r)$  for the observed spatial point process.





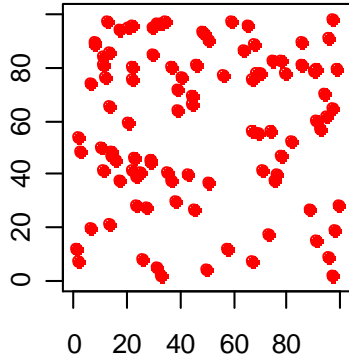
**G in R (Grrr...).** The `spatstat` package uses the function `Gest()` to calculate  $G(r)$  with various corrections. The function `envelope()` with option `fun=Gest` does the Monte Carlo simulation of the bounds for  $G(r)$ . You can use `plot()` to plot the result of `Gest()`. However, this doesn't make the plot with the envelopes. SO : I've written a small function to do this called `G()`. Link : <http://reuningscherer.net/fes781/rscripits/SpatialPPFuncs.R.txt> It requires as inputs the outputs from `envelope()` and `Gest()`.

Examples below :

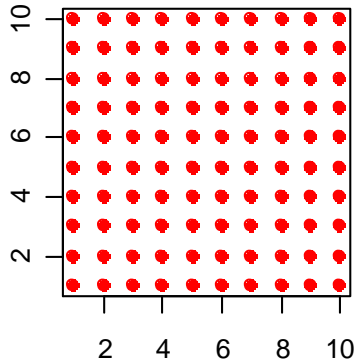
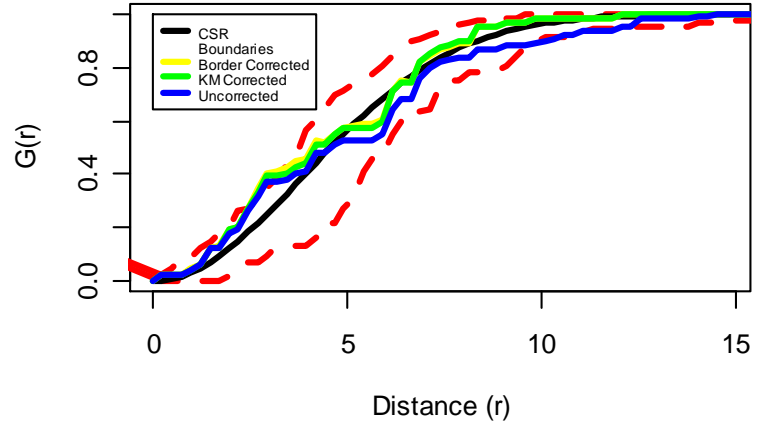
<http://reuningscherer.net/fes781/rscripits/GandFFunctions.R.txt>

.....

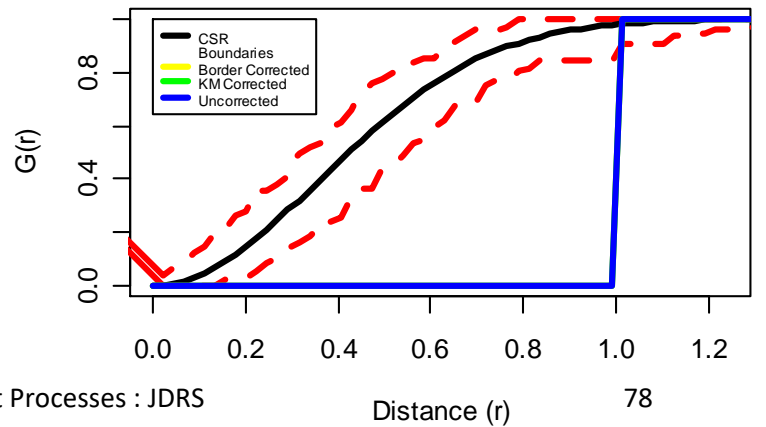
*The plots on the next page show the function  $\hat{G}(r)$  for several simulated point processes. Note that as we would expect,  $\hat{G}(r)$  is unusually high for clustered processes and unusually low for regular processes.*

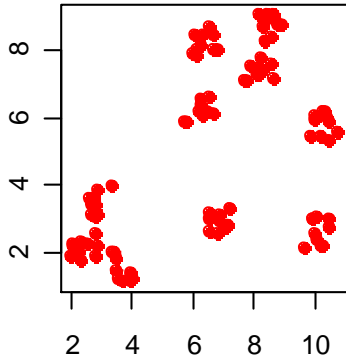


**G-plot for Randomly Generated CSR**

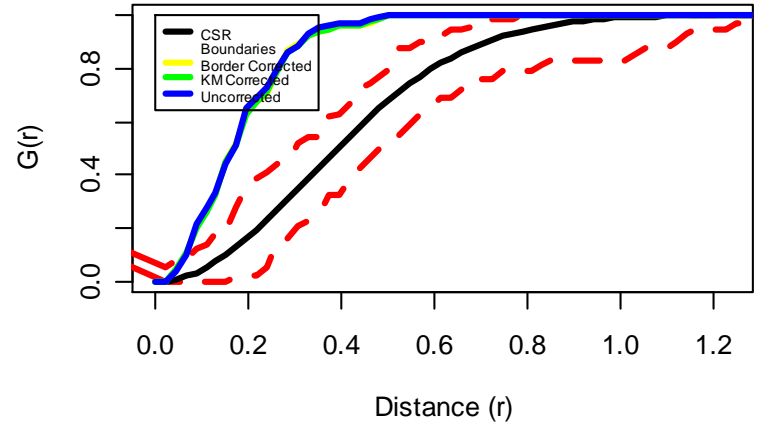


**G-plot for Regular Grid**





**G-plot for Clustered Process**

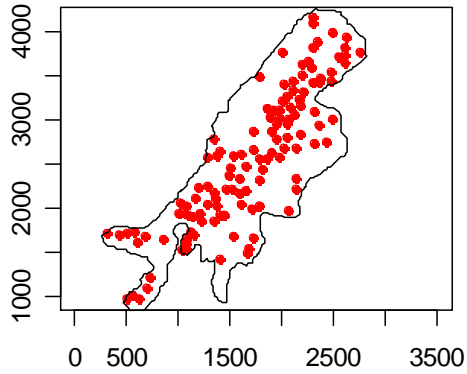


**Example : Uganda Volcano Locations and Cardiff Delinquency Data.**

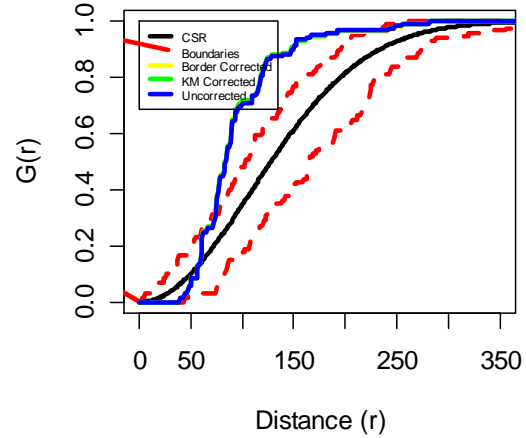
*Below are the results of the G-function with simulation envelopes. Both indicate some clustering at smaller distances, although there appears to be greater separation at very short distances in the volcano data (less than 50 km) – similar to what we saw using the L-function*



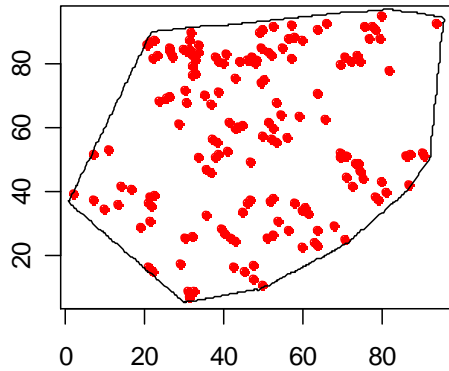
### Uganda Volcanos



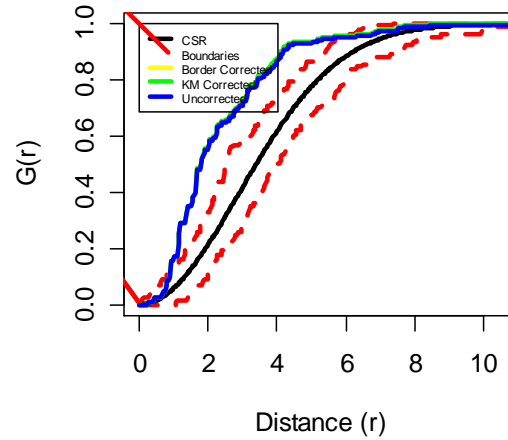
### G-plot for Uganda Volcano Data



### Cardiff Delinquents

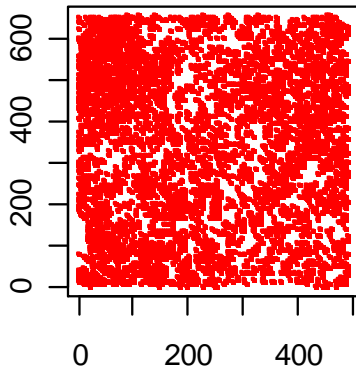


### G-plot for Cardiff Delinquents

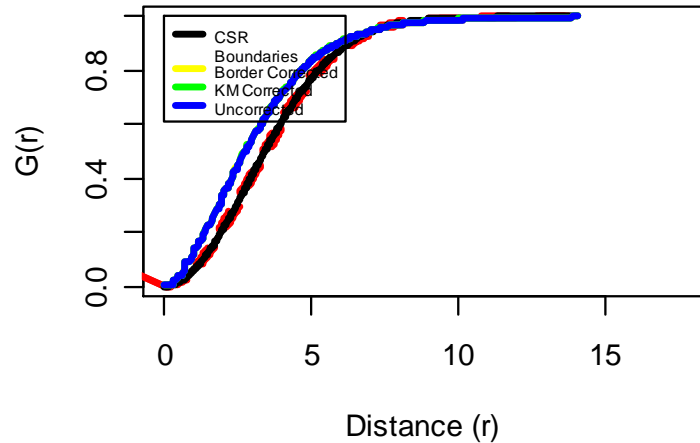


**Example : Maine Tree Data.** This is data Tim collected on different species at a site in Maine. For All trees and for Hemlock and Red Maple, clustering is evident. (most prominent for the red maple).

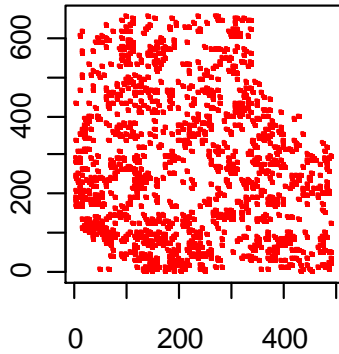
**All Trees**



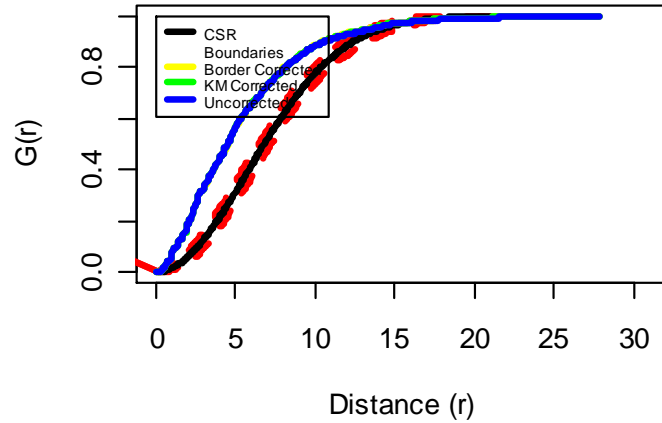
**G-plot for All Trees**



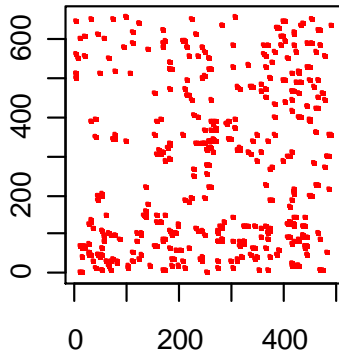
### Hemlock



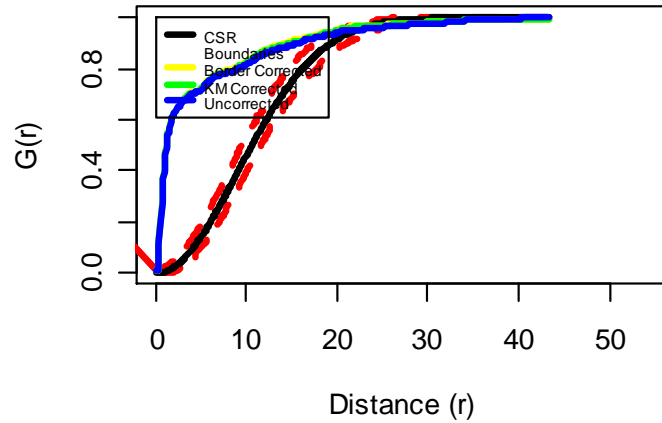
### G-plot for Hemlock



### Red Maple



### G-plot for Red Maple





## F and J functions

$F(r)$  is defined as



$$F(r) = \Pr(\text{Dist}(\text{random location to nearest event}) \leq r)$$

This is the **cumulative probability distribution of nearest neighbor distances from a random point to an event.**

We estimate this function with

$$\hat{F}(r) = \frac{\#(x_i \leq r)}{M} = \frac{\sum_{i=1}^M I(x_i \leq r)}{M}$$



where  $M$  is a random sample of non-event locations and  $I(\cdot)$  is the indicator function = 1 if true, 0 otherwise.

**NOW** : we\* can define the function  $J(r)$  :

$$\hat{J}(r) = \frac{1 - \hat{G}(r)}{1 - \hat{F}(r)}$$

**\*Van Lieshout, M.N.M. and Baddeley, A.J.** A nonparametric measure of spatial interaction in point patterns. *Statistica Neerlandica* **50** (1996) 344-361.

**SO** : if we have CSR, what should  $F(r)$  be? Turns out that under CSR (*in 2-dimensions*),

$$F(r) = G(r) = 1 - e^{-\lambda\pi r^2}, \quad r \geq 0$$

which means that for a CSR process

$$J(r) = 1$$

**Edge corrections and simulation boundaries** – same methods apply to  $F(r)$  (and  $J(r)$ ) as used for  $G(r)$



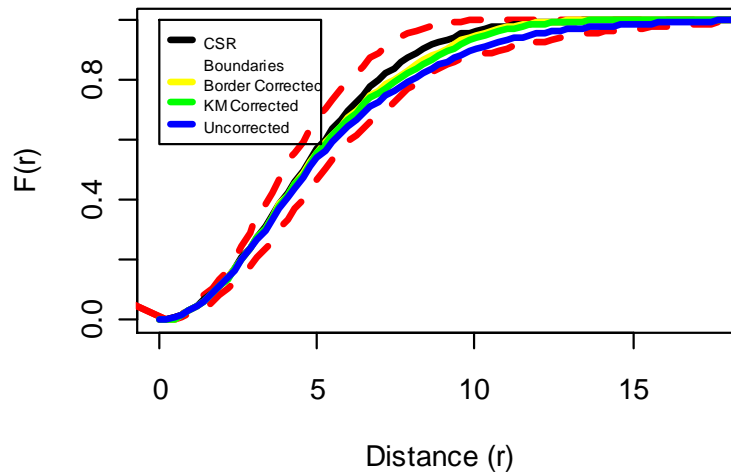
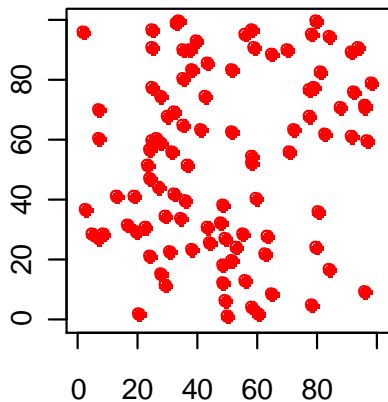
**F and J in R (where's H?).** The `spatstat` package uses the function `Fest()` and `Jest()` to calculate  $F(r)$  and  $J(r)$  with various corrections. The function `envelope()` with option `fun=Fest` or `Gest` does the Monte Carlo simulation of the bounds for  $F(r)$  and  $J(r)$ . You can use `plot()` to plot the result of `Fest()` and `Jest()`. I wasn't so crazy about the plots, so I wrote an `Ffunc()` function. For `Jest`, see code in same file which produces the graphs below. The help function in R gives references of papers exploring the properties of  $J(r)$ . Link :

<http://reuningscherer.net/fes781/rscripts/SpatialPPFuncs.R.txt>

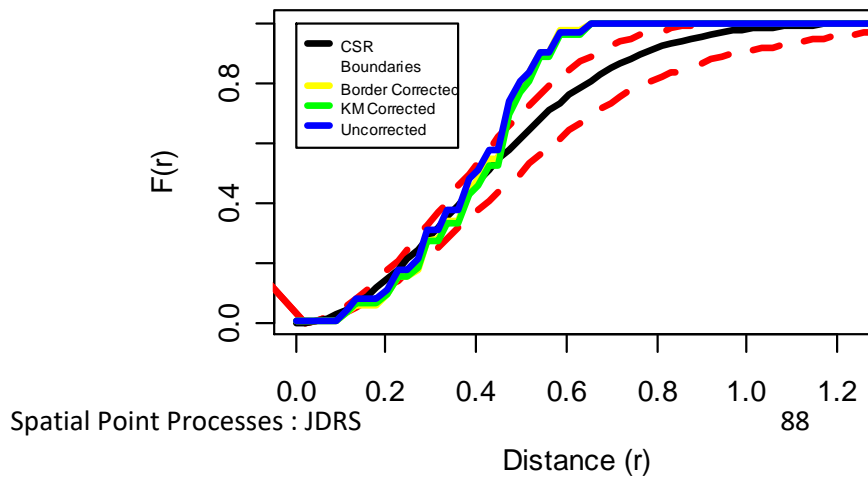
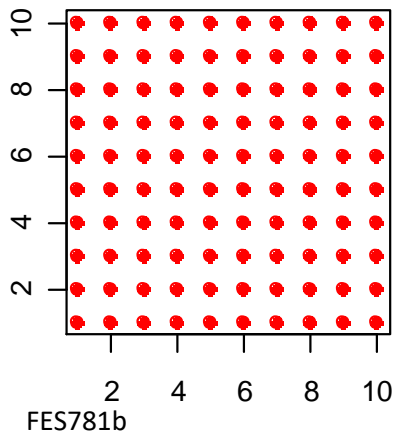
Examples below :

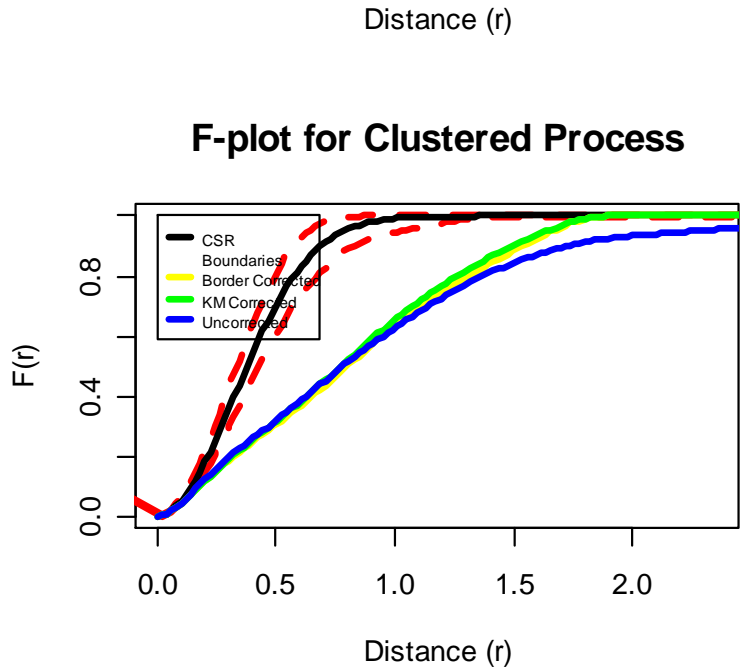
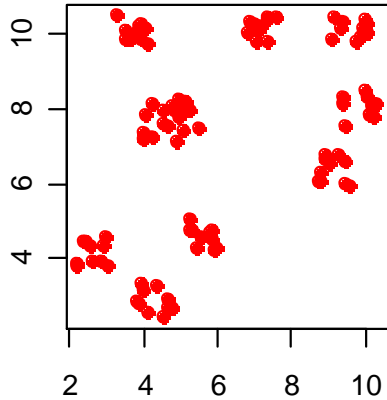
<http://reuningscherer.net/fes781/rscripts/GandFFunctions.R.txt>

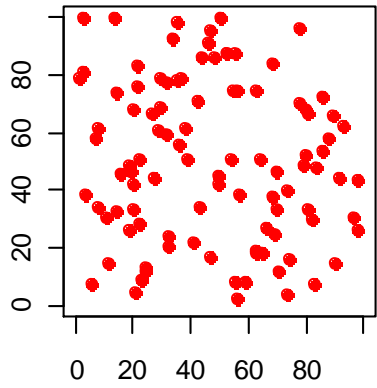
### F-plot for Randomly Generated CSR



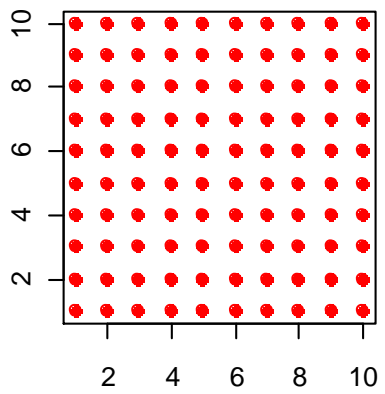
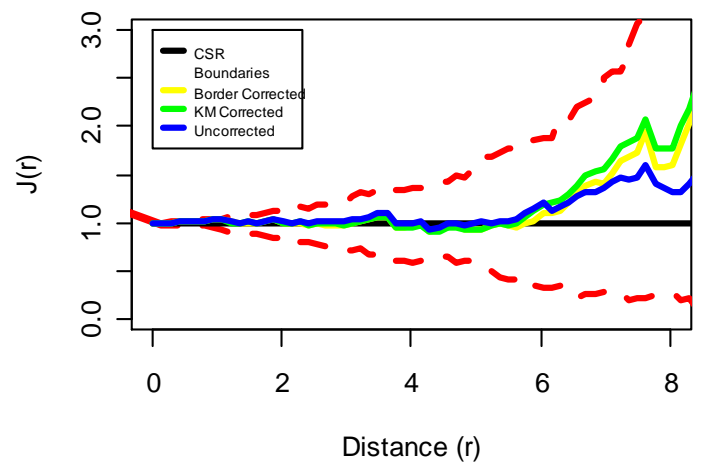
### F-plot for Regular Grid



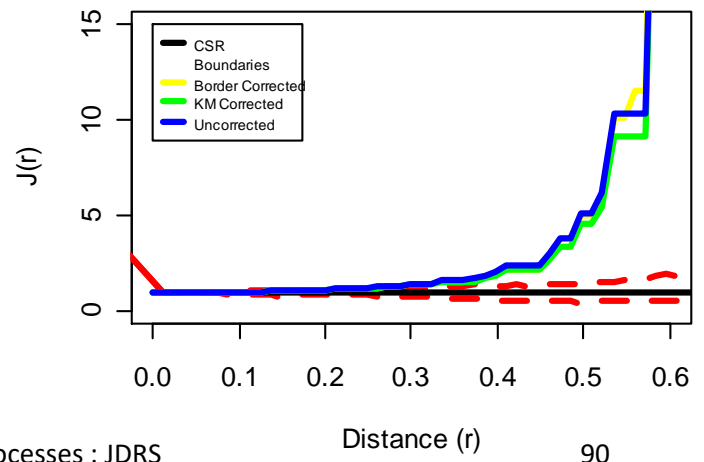


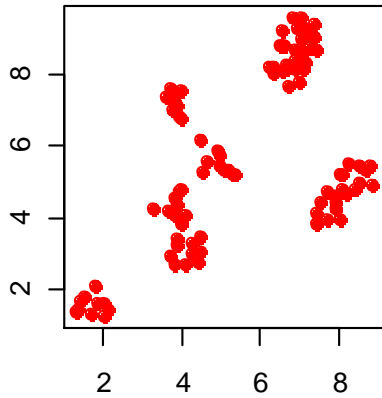


**J-plot for Randomly Generated CSR**

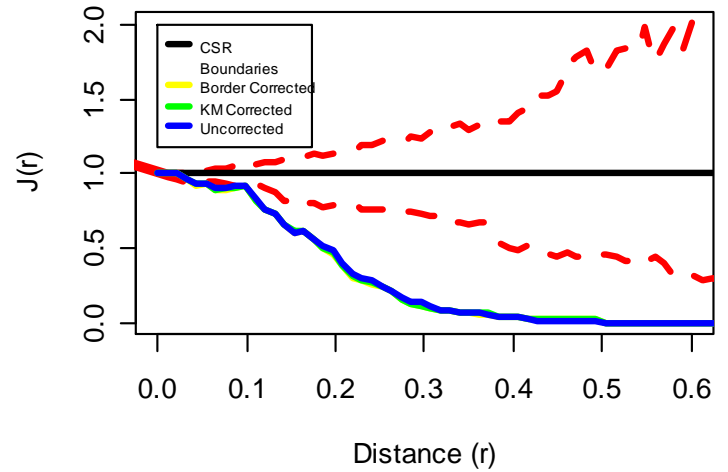


**J-plot for Regular Grid**





**J-plot for Clustered Process**

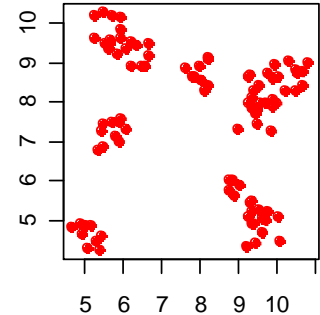


*What does all of this suggest?!?!?*

## Clustered Processes :

- $\hat{G}(r)$  tends to increase to 1 **more rapidly** than the expected value of  $G(r)$  under CSR.
- $\hat{F}(r)$  tends to increase to 1 **more slowly** than the expected value of  $F(r)$  under CSR.

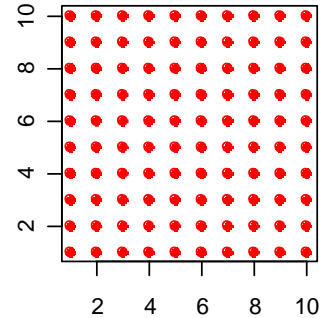
- $$\hat{J}(r) = \frac{1 - \hat{G}(r)}{1 - \hat{F}(r)} = \left( \frac{1 - \text{big}}{1 - \text{small}} \right) \approx 0$$





## Regular Processes :

- $\hat{G}(r)$  tends to increase to 1 **more slowly** than the expected value of  $G(r)$  under CSR.
- $\hat{F}(r)$  tends to increase to 1 **more rapidly** than the expected value of  $F(r)$  under CSR.
- $\hat{J}(r) = \frac{1 - \hat{G}(r)}{1 - \hat{F}(r)} = \left( \frac{1 - \textit{small}}{1 - \textit{big}} \right) \Rightarrow \textit{big}$



# Inference for Spatial Point Processes

*We start by talking about*



## Hypothesis Testing



- **Goal** : perform hypothesis tests relating to the distribution of an observed spatial point process.

***Example** : you'd like to see if a stand of trees or a set of leukemia cases is more clustered than you 'would expect at random (?!?)'*

- **Challenge** is to form **reasonable null and alternative hypotheses**

# CSR as a null hypothesis

- A variety of statistics have been developed to calculate the probability of observing a particular spatial process **if the underlying process is CSR.**
- Most statistics involve **Nearest Neighbor calculations** based on either  $W = \{w_1, w_2, \dots, w_N\}$  or  $X = \{x_1, x_2, \dots, x_M\}$ .

**Aside** : (see *Cressie* p. 602-605 if you want to see the proofs)



**Theoretical distribution of  $W$  and  $X$  :**

- Under CSR,  $2\pi\lambda X^2 \sim \chi_2^2$
- This 'means' that

$$\text{Mean}(X) = \frac{1}{2\sqrt{\lambda}}$$

$$\text{Var}(X) = \frac{(4 - \pi)}{4\lambda\pi}$$

(true for both  $W$  and  $X$ )

## Clark and Evans (1954)

Test statistic is

$$z = \frac{\left( \frac{2\sqrt{\hat{\lambda}}}{n} \sum_{i=1}^n w_i \right) - 1}{\sqrt{(4 - \pi)/n\pi}}$$

where  $\hat{\lambda} = \frac{N}{|R|} = \frac{\{\# \text{ Events}\}}{\{\text{Size of Region}\}}$  is the usual estimate of the intensity.

Under CSR, it can be shown that  $z \sim N(0,1)$  as long as the number of events  $N$  is larger than 10.

For a regular process,  $z$  will be large positive; for a clustered process, it will be large negative.

Cressie describes several dozen other statistics developed in the past 50 years based on the distribution of  $W$  and  $X$ . The table below (Cressie, p. 604) lists a number of these statistics :

**Table 8.6 Nearest-Neighbor Statistics and Their Asymptotic Distributions under csr**

Basic Measurement	Test Statistic	Asymptotic or Exact Distribution	Reference
$W$	A $2(\lambda)^{1/2}\Sigma W_i/n$	$N(1, (4 - \pi)/n\pi)$	Clark and Evans (1954)
	B $2\pi\lambda\Sigma W_i^2$	$\chi_{2n}^2$	Skellam (1952)
$X$	C $\pi\lambda\Sigma X_i^2/n$	$N(1, 1/n)$	Pielou (1959)
	D $n(\Sigma X_i^2)/(\Sigma X_i)^2$	By simulation	Eberhardt (1967)
	E $12n\{n \log(\Sigma X_i^2/n) - \Sigma \log X_i^2\}/(7n + 1)$	$\chi_{n-1}^2$	Pollard (1971)
$X, X_2$	F $\{\Sigma(X_i^2/X_{2,i}^2)\}/n$	$N(1/2, 1/12n)$	Holgate (1965a)
	G $(\Sigma X_i^2)/(\Sigma X_{2,i}^2)$	beta( $n, n$ )	Holgate (1965a)
$X, W$	H $[\Sigma\{X_i^2/(X_i^2 + W_i^2)\}]/n$	$N(1/2, 1/12n)$	Byth and Ripley (1980)
	I $(\Sigma X_i^2)/(\Sigma W_i^2)$	$F_{2n, 2n}$	Hopkins (1954)
$X, Z$	J $2n\Sigma(2X_i^2 + Z_i^2)/\{\Sigma(\sqrt{2}X_i + Z_i)\}^2$	By simulation	Hines and Hines (1979)
	K $48n[n \log\{\Sigma(2X_i^2 + Z_i^2)/n\} - \Sigma \log(2X_i^2 + Z_i^2)]/(13n + 1)$	$\chi_{n-1}^2$	Diggle (1977)
	L $\Sigma\{2X_i^2/(2X_i^2 + Z_i^2)\}/n$	$N(1/2, 1/12n)$	Besag and Gleaves (1973)
	M $2\Sigma\{\min(2X_i^2, Z_i^2)/(2X_i^2 + Z_i^2)\}/n$	$N(1/2, 1/12n)$	Diggle et al. (1976)
$X, Y$	N $2(\Sigma X_i^2)/(\Sigma Z_i^2)$	$F_{2n, 2n}$	Besag and Gleaves (1973)
	O $-2\Sigma[\log\{2X_i^2/(2X_i^2 + Z_i^2)\}]$	$\chi_{2n}^2$	Cormack (1979)
	P $\Sigma R_i/n'$	$N(1/2, 1/12n')$	Cox and Lewis (1976)
	Q $\bar{A}_X - \bar{A}_Y$	$N(0, (\bar{A}_X^2 + \bar{A}_Y^2)/n')$	Satyamurthi (1979)



**Nearest Neighbor Statistics in R.** In the `spatstat` package use the function `clarkevans.test` to calculate the Clarke Evans statistic. I haven't found any other of the statistics listed above so far. SO – project for you if you're interested!

---

***Example*** : See

[http://reuningscherer.net/fes781/rscripts/CSR\\_Tests.r.txt](http://reuningscherer.net/fes781/rscripts/CSR_Tests.r.txt)

# Quadrat Count Test for CSR

- When we create quadrat counts, we create equal sized regions that should all have about the same number of counts
- We can compare the number of counts in each 'cell' to the expected count in each cell under CSR using a **Chi-Square Test!**
- **Problem** – the number of cells is again somewhat arbitrary (i.e. grid size).
- **Problem** – this method may fail to detect departures from CSR (i.e. clustering or regularity) if mean still stays constant

**Example** : See

[http://reuningscherer.net/fes781/rscripts/CSR\\_Tests.r.txt](http://reuningscherer.net/fes781/rscripts/CSR_Tests.r.txt)



## Examples : CSR simulated data

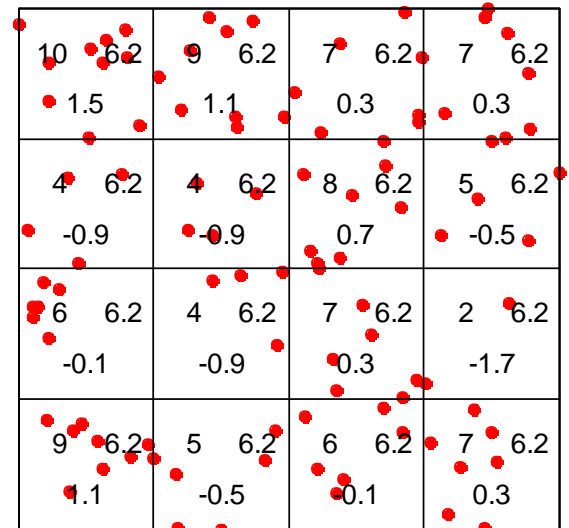
Clark-Evans test No edge correction  
Z-test

```
data: CSR.ppp
R = 0.97321, p-value = 0.6084
alternative hypothesis: two-sided
(z-stat : -0.1631077)
```

Chi-squared test of CSR using  
quadrat counts Pearson X2  
statistic

```
data: CSR.ppp
X2 = 11.36, df = 15, p-value =
0.5466
alternative hypothesis: two.sided
```

Quadrats: 4 by 4 grid of tiles



**Examples : Grid simulated data. Fails to reject null hypothesis (whereas Clark Evans did reject)**

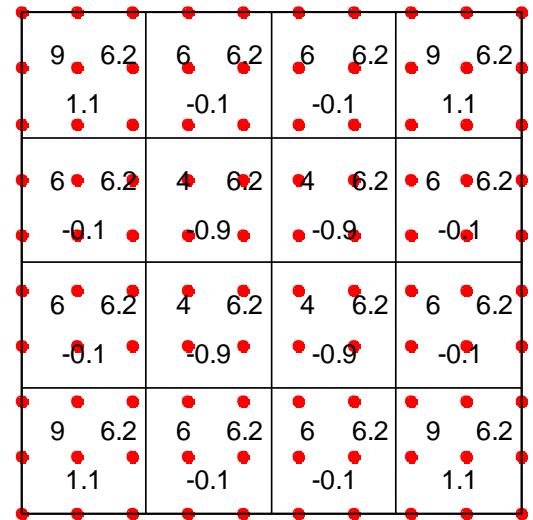
Clark-Evans test No edge correction  
Z-test

```
data: grid.ppp
R = 2.2222, p-value < 2.2e-16
alternative hypothesis: two-sided
(z-stat : 7.442666)
```

Chi-squared test of CSR using  
quadrat counts Pearson X2  
statistic

```
data: grid.ppp
X2 = 8.16, df = 15, p-value =
0.1656
alternative hypothesis: two.sided
```

Quadrats: 4 by 4 grid of tiles



**Examples : Clustered simulated data. Does reject null hypothesis.**

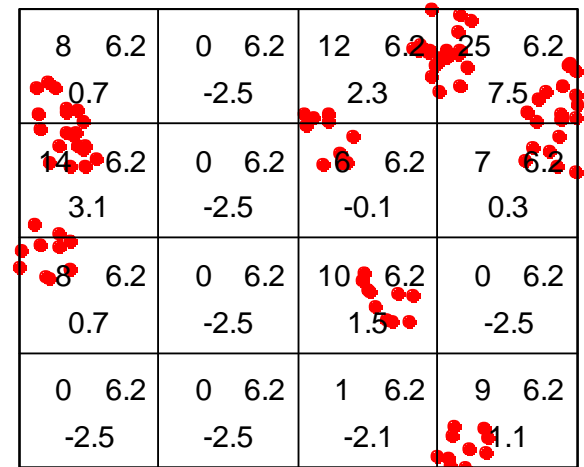
Clark-Evans test No edge correction  
Z-test

```
data: Clust.ppp
R = 0.46289, p-value < 2.2e-16
alternative hypothesis: two-sided
(z-stat : -3.270713)
```

Chi-squared test of CSR using  
quadrat counts Pearson X2  
statistic

```
data: Clust.ppp
X2 = 117.6, df = 15, p-value <
2.2e-16
alternative hypothesis: two.sided
```

Quadrats: 4 by 4 grid of tiles



# Examples : Uganda Volcano Data. Does reject null hypothesis.

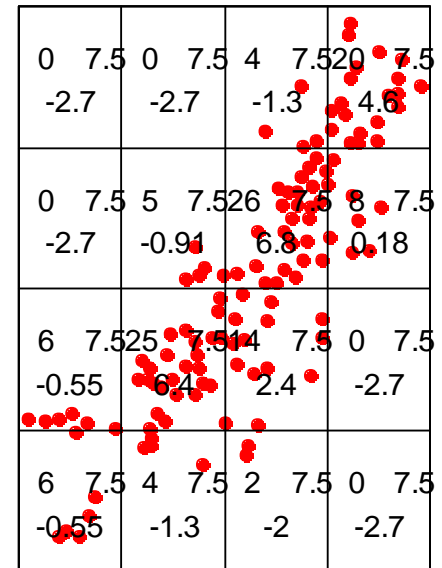
Clark-Evans test No edge correction  
Z-test

```
data: ugppp
R = 0.69198, p-value = 1.082e-10
alternative hypothesis: two-sided
(z-stat : -2.054719)
```

Chi-squared test of CSR using quadrat counts Pearson X2 statistic

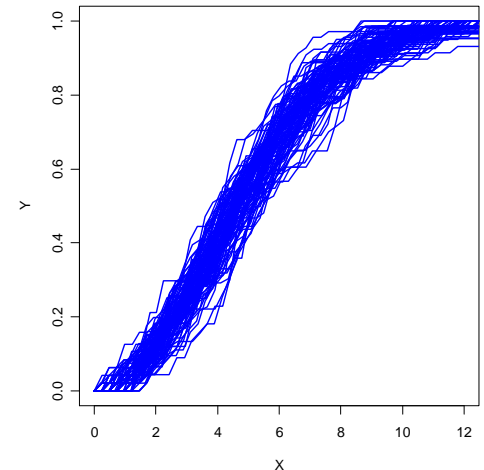
```
data: ugppp
X2 = 159.2, df = 15, p-value < 2.2e-16
alternative hypothesis: two.sided
```

Quadrats: 4 by 4 grid of tiles



## Simulation Envelopes for $K(r)$ , $L(r)$ , $G(r)$ , etc.

- The plots we've produced for  $K(r)$ ,  $L(r)$ ,  $G(r)$ , and  $F(r)$  have all had **simulation envelopes**
- These envelopes are created by generating say, 99 CSR processes with the same number of events as the observed process.
- The various functions are calculated for the simulations and the functions are plotted vs.  $r$
- The **convex hull** (the outside upper and lower boundaries) provide the **simulation envelope boundaries** (example at right is 99 simulations of  $G(r)$  for a CSR with 100 events)



- This means the envelopes are in a sense a (*in example case 99%*) **Confidence Region**.
- **Major deviation of the observed function from the simulated function indicates a significant departure from CSR**

**Example : Tropical Rainforest Data (bei in spatstat package).** We divide slope into 8 quantiles and use the resulting tessellated plot to define 8 quadrats. A plot with tree locations suggests that tree location and slope may be related. Quadrat counts show that counts clearly differ with slope quantiles (otherwise, counts would all be approximately the same). We can perform same chi-square test based on these 8 divisions.

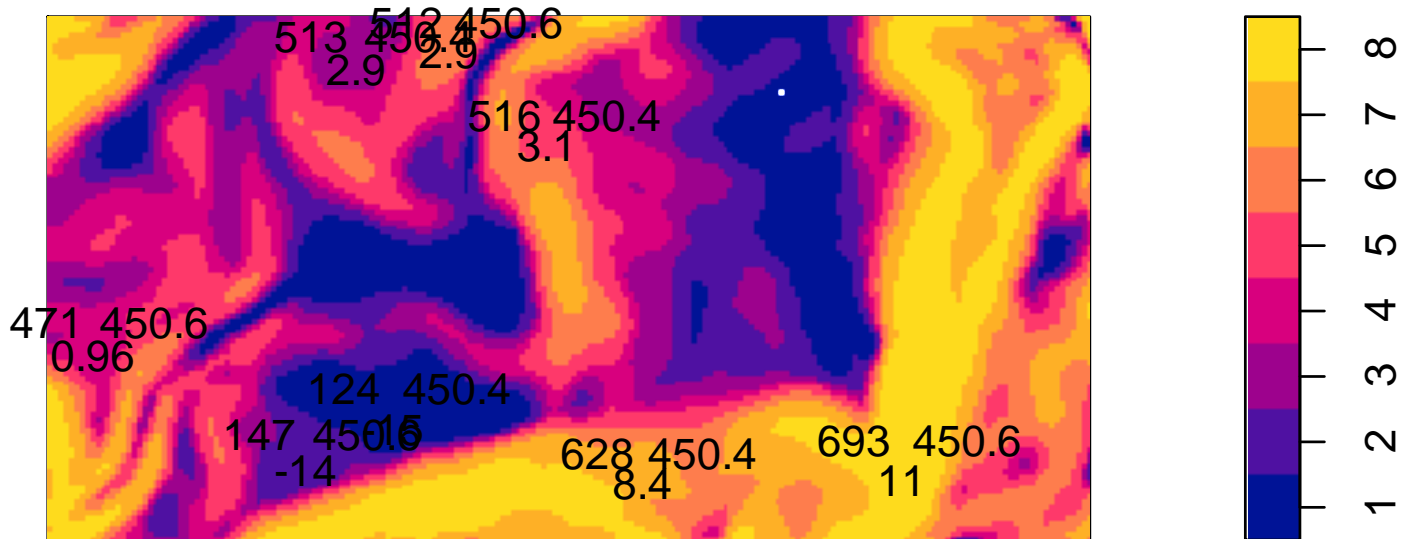
```
Chi-squared test of CSR using quadrat counts
      Pearson X2 statistic
```

```
data:
```

```
X2 = 669.07, df = 7, p-value < 2.2e-16
```

```
alternative hypothesis: two.sided
```

```
Quadrats: 8 tiles (levels of a pixel image)
```



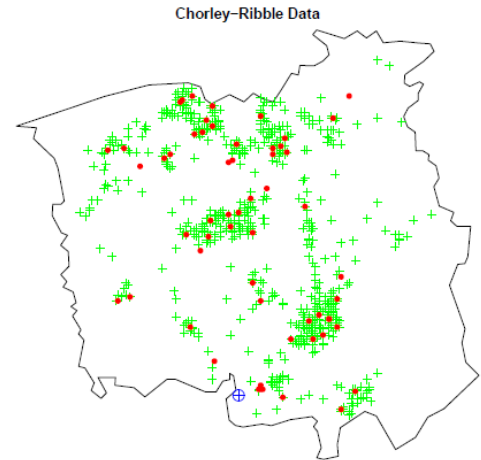
[http://reuningscherer.net/fes781/rscripts/CSR\\_Tests.r.txt](http://reuningscherer.net/fes781/rscripts/CSR_Tests.r.txt)



# Multiple Groups – Marked Point Processes

For an EXCELLENT overview of spatial point processes through the SPATSTAT package see Adrian Baddeley's notes <http://www.csiro.au/resources/pf16h>. See Part VII, p. 177ff for Marked PP.

For a good overview of ALL functions available in SPATSTAT see the Quick Reference Card here : <http://spatstat.org/spatstat/>



- Sometimes point processes are actually combinations of individuals in several groups

**Example** : Cases and Controls, or the Maine Tree Data where multiple species are present

- Each point is said to be **MARKED** – i.e. we have a marked point process

**Aside** - As we mentioned in lecture 1, a process can be marked by a categorical **OR** a continuous variable (for example, tree locations could be 'marked' by their corresponding diameter). For the moment, we'll assume a process is marked by a categorical variable (i.e. group membership)

- We'll discuss both two point processes and multiple point processes, and we'll discuss comparing density estimates as well as multi-group versions of the F, G, K, J, L functions.
- We start with two-groups, sometimes called

# Case-Control Processes



- For many point processes, CSR is NOT the appropriate null hypothesis

***Example** : events will appear to be not CSR simply because the events follow a heterogenous Poisson PP that mirrors, say, population. For example disease or crime events will simply follow the population distribution; or multiple tree species simply group by more favorable soil/climate conditions.*

- In this case we need to ‘correct’ for the underlying intensity function
- Another way to think about this : we have a set of **CASES** and **CONTROLS** and we want to see if they **follow the same underlying intensity function**

# Ratio of Kernel Intensity Estimates



- The basic idea is to compare the intensity function of 'cases' to that of 'controls'
- This comparison is done by computing a kernel intensity estimate for each group over the **SAME region  $R$** .
- A ratio of estimates is then calculated and plotted.

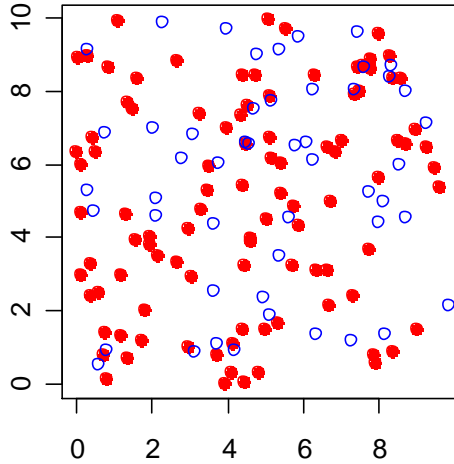
**NOTE** : this method requires that the intensity functions be **normalized** (i.e. standardized) to have total area under the curve = 1. That is, we use **density functions** rather than intensity functions in the ratio.

- The ratio of density functions is the local **Relative Risk** at each point  $s$  in  $R$ .
- **Locations where the Relative Risk is much larger than 1 or much less than 1 indicate areas where the density functions disagree.**
- In practice, the success of this method is dependent on the bandwidth size of the kernel chosen.
- Waller and Gotway also suggest looking at the log(relative risk) surface as a means of comparison of density functions (since this means that ratios become subtracted differences – log of ratio = difference of logs . . . .)

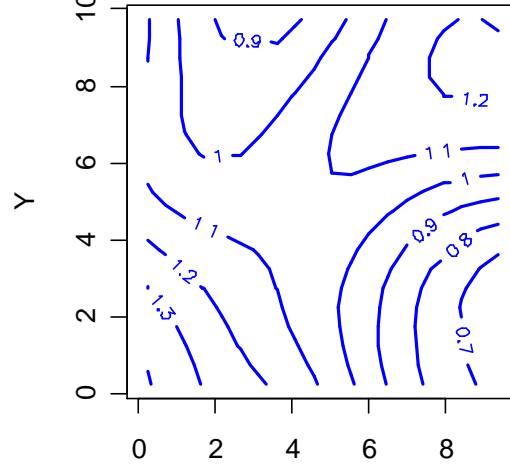
**Example** : *Two random CSR processes, one with 100 observations, the other with 50 observations. I tried several kernel bandwidths and decided on a bandwidth of 2. The intensity functions of each group show no real patterns and the ratio of intensity functions is never more (or less) than 1.5 : 1 suggesting that the distributions are relatively similar.*

<http://reuningscherer.net/fes781/rscripts/kernelratios.r.txt>

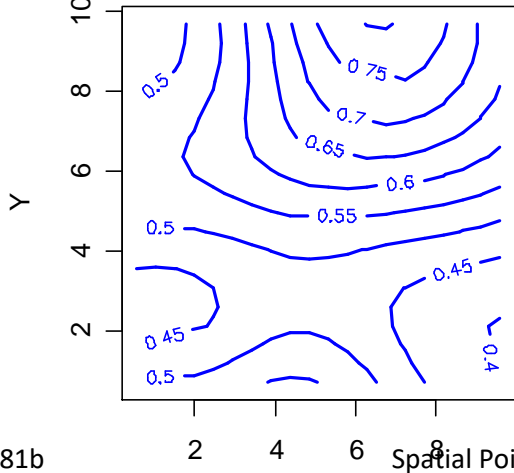
**Cases (Red) and Controls (Blue)**



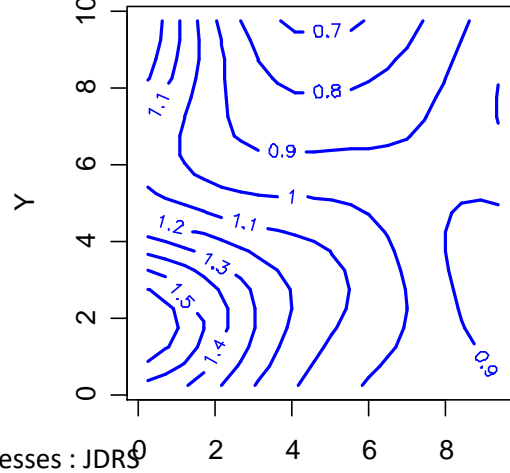
**Cases, Sigma=2**



**Controls, Sigma=2**

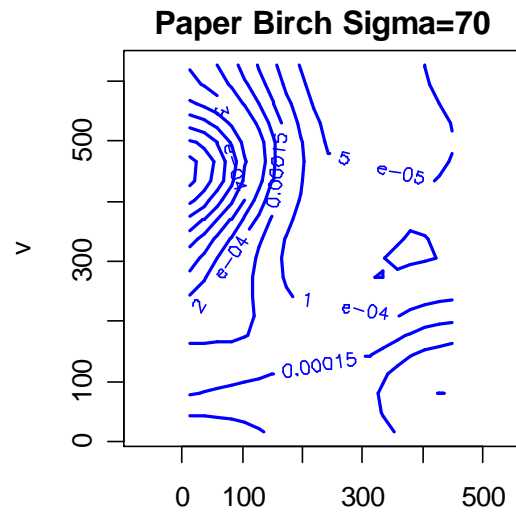
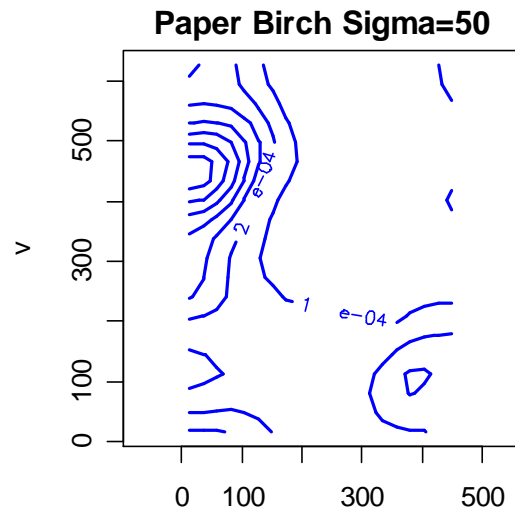
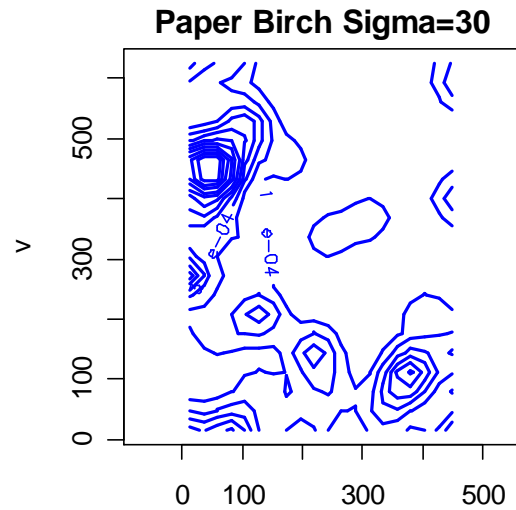
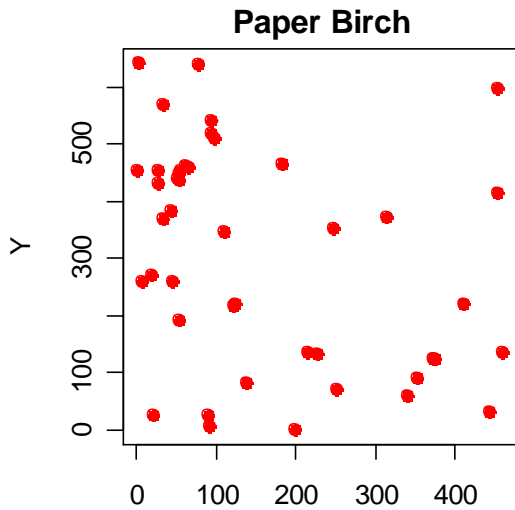


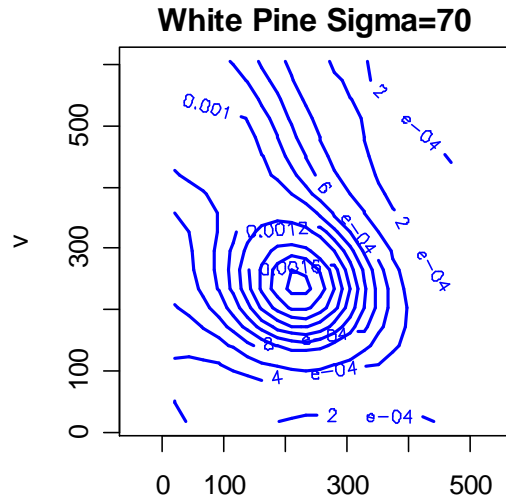
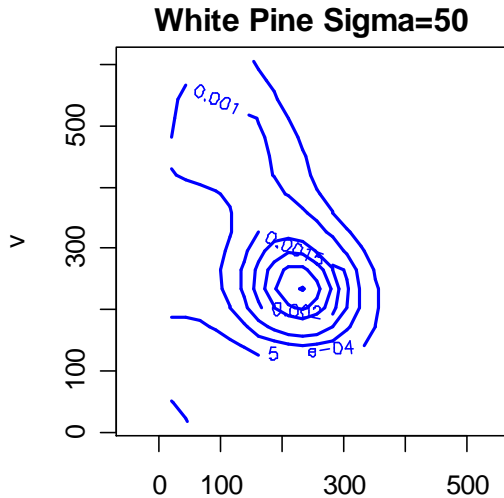
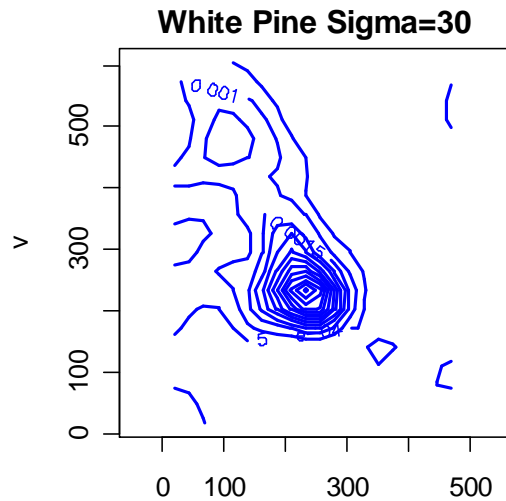
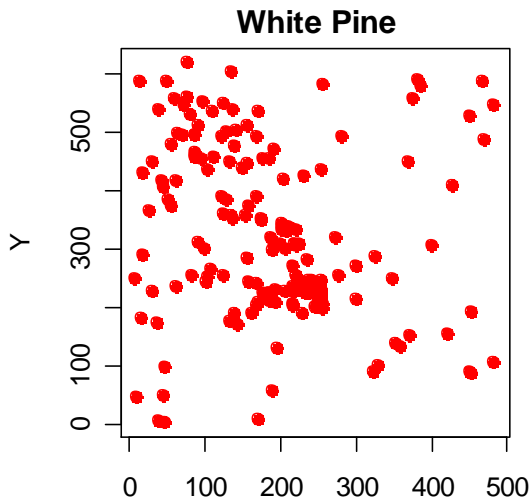
**Ratio of Cases to Controls, Sigma=2**



**Example : Maine Stand data.** Lets examine the relative distribution of two species in Tim's Maine data : paper birch and white pine. It appears that these species are clustered within groups and away from each other. Several kernel bandwidths were tried : I choose  $\sigma=70$ .

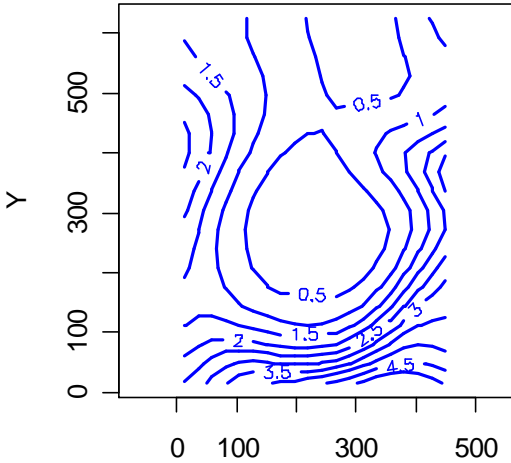




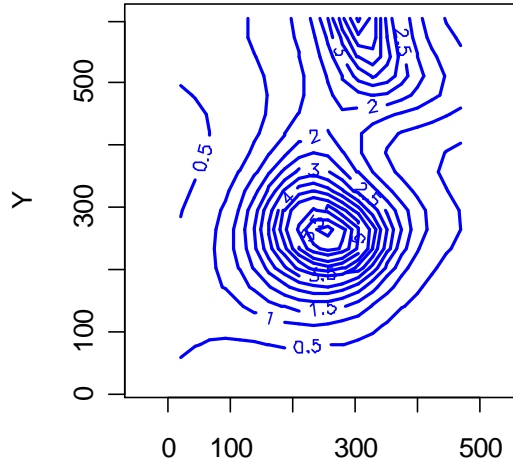


*The plot below shows the ratios of the intensity functions (I did both paperbirch vs. white pine and the reverse). There are regions in both cases where the relative risk is on the order of 4 to 5 times that of the other species. This suggests that these species are 'moving around' each other.*

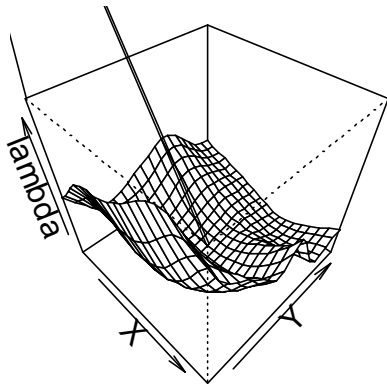
Ratio of PB to WP, Sigma=70



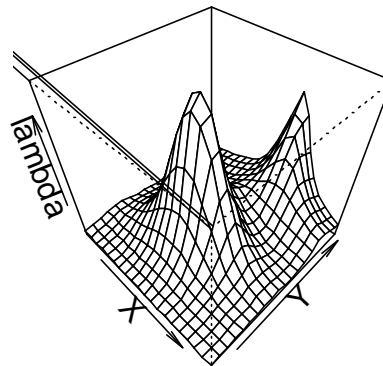
Ratio of WP to PB, Sigma=70



Sigma=70



Sigma=70





**Ratios of Kernel Densities in R.** The plots above were created with the code online as `KernelRatios.R.txt`. This uses the function `density.ppp()`, but you could easily modify to use the `density()` function if you like.

<http://reuningscherer.net/fes781/rscripts/kernelratios.r.txt>

The `spatstat` package actually has some nice ways of doing this now –Script is `MarkedSPP.R.txt`

<http://reuningscherer.net/fes781/rscripts/markedsp.r.txt>

The ratio of density estimates in this method is expressed as relative probabilities rather than relative risk. However, the interpretation is similar

$$\text{Relative Risk} = \frac{\Pr(\text{Group } i)}{\Pr(\text{Group } j)} = \frac{\textit{Intensity}(\text{Group } i)}{\textit{Intensity}(\text{Group } j)}$$

$$\Pr(\text{Group } i \text{ rather than Group } j) = \frac{\Pr(\text{Group } i)}{\Pr(\text{Group } i) + \Pr(\text{Group } j)}$$

*For example, suppose that at a particular location (x,y), there is a 10% chance of seeing a white pine, and a 30% chance of seeing a cedar.*

*The relative risk of cedar to white pine is  $\frac{.3}{.1} = 3$*

*The probability of seeing cedar rather than white pine is  $\frac{.3}{.4} = 0.75$  (i.e. if a tree does occur at location (x,y), it is 3 times as likely to be cedar than white pine).*

---

***Examples*** : *Maine Tree Data, Two CSR's, a CSR and a clustered process, etc.*

# RETURN OF THE ALPHABET SOUP

*(for two groups)*



Let's suppose we have two spatial point processes, each with separate (constant) intensity functions  $\lambda_1$  and  $\lambda_2$ . Our various alphabet functions become a bit more complicated with two groups :



With one group, we defined  $G$  as

$$G(r) = \Pr(\text{Dist}(\text{random event to nearest event}) \leq r)$$

We can now define two separate  $G$  functions :

$$G_{12}(r) = \Pr(\text{Dist}(\text{rnd group 1 event to nearest group 2 event}) \leq r)$$

$$G_{21}(r) = \Pr(\text{Dist}(\text{rnd group 2 event to nearest group 1 event}) \leq r)$$

These functions differ in which group serves as the 'reference' group.

If each group is a CSR process, then as before we have

$$G_{12}(r) = G_{21}(r) = 1 - e^{-\lambda_{(1 \text{ or } 2)}\pi r^2}, \quad r \geq 0$$



We estimate this function with

$$\hat{G}_{12}(r) = \frac{\#(w_{i2} \leq r)}{N_1} = \frac{\sum_{i=1}^{N_1} I(w_{i2} \leq r)}{N_1}$$

where  $N_1$  is the number of observations in group 1,  $w_{i2}$  is the distance from the  $i$ th point in group 1 to the nearest point in group 2, and  $I(\cdot)$  is the indicator function.

*Basically, this is the percent of observed nearest cross-group neighbor distances that are less than  $r$*



$F(r)$  was defined as

$$F(r) = \Pr(\text{Dist}(\text{random location to nearest event}) \leq r)$$

For two groups, this is not particularly interesting (since it basically means we're fitting two separate  $F(r)$  functions, one for each group (i.e. groups aren't affecting each other):

$$F_1(r) = \Pr(\text{Dist}(\text{rnd location to nearest group 1 event}) \leq r)$$

$$F_2(r) = \Pr(\text{Dist}(\text{rnd location to nearest group 2 event}) \leq r)$$

If each group is a CSR process, then as before we have

$$F_1(r) = F_2(r) = 1 - e^{-\lambda(1 \text{ or } 2)\pi r^2}, \quad r \geq 0$$

I'll leave details of how to estimate this function. R can calculate this function (the `Fcross()` function in the `spatstat` package)



Two different  $J(r)$  functions can also be calculated :

$$J_{12}(r) = \frac{1 - G_{12}(r)}{1 - F_2(r)} \quad \text{and} \quad J_{21}(r) = \frac{1 - G_{21}(r)}{1 - F_1(r)}$$

Again, I'll leave details of how to estimate this function. R can calculate this function (the `Jcross()` function in the `spatstat` package). Like F, I'm not sure this is so useful .

..



# Ripley's Bivariate K-function

Before, we defined K as

$$K(h) = \frac{E(\# \text{events within } h \text{ of a random event})}{\lambda}$$

For two groups, we now define

$$K_{12}(h) = \frac{E(\# \text{events in group 2 within } h \text{ of a random group 1 event})}{\lambda_2}$$

$$K_{21}(h) = \frac{E(\# \text{events in group 1 within } h \text{ of a random group 2 event})}{\lambda_1}$$

**If both groups are CSR :**

$$K_{12}(h) = K_{21}(h) = \pi h^2 \text{ as before.}$$

K is estimated as follows :

$$\hat{K}_{12}(h) = \frac{1}{\hat{\lambda}_2 N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} I(d(i, j) < h)$$

where  $I(\ )$  is the indicator function and  $\hat{\lambda}_2 = \frac{N_2}{|R|}$  (number of points in Group 2 divided by size of region)

Basically,  $\hat{K}_{12}(h)$  is just an average of the number of points in group 2 within  $h$  of each observed event in group 1.



$$L_{12}(h) = \sqrt{\frac{K_{12}(h)}{\pi}} \quad \text{and} \quad L_{21}(h) = \sqrt{\frac{K_{21}(h)}{\pi}}$$

(Note that before, I subtracted  $h$  so that this would be a flat line at zero. The spatstat package doesn't do this, and I haven't written a new function yet – SO, just be aware that this looks like a line with slope 1 under CSR)



**Bivariate Comparison Functions in R.** In the script MarkedSPP.R.txt are a few examples of bivariate functions  
<http://reuningscherer.net/fes781/rscripts/marked spp.r.txt>

The R bivariate functions are `Gcross`, `Kcross`, `Lcross`, and `Jcross`

Each of these functions can be used in the envelope function to get randomized 'confidence bands' as a function of distance.



# Ratio of Kernel Intensity Estimates for MORE than two groups



- The basic idea is to compare the intensity function of 'cases' to **ALL OTHER GROUPS COMBINED**.
- This comparison is done by computing a kernel intensity estimate for each group over the **SAME region  $R$** .
- A ratio of estimates is then calculated between a particular group and **ALL OTHER COMBINED GROUPS** and is then plotted.





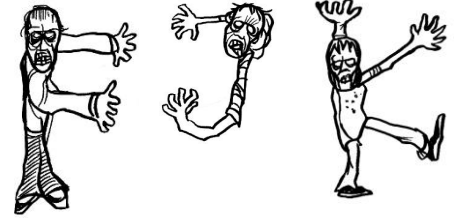
**Ratios of Kernel Densities in R.** In the script `MarkedSPP.R.txt` are a few examples of multiple group kernel density estimates.

<http://reuningscherer.net/fes781/rscripts/markedsp.r.txt>

discussed in class . . . . .



# STILL MORE ALPHABET SOUP (for more than two groups)



*Example : more than two observed tree species*

**Notation** - OK, we're in a bit of trouble cause almost every letter is used now, but we'll say we have lower case  $g$  groups, and let  $a$  and  $b$  be any two particular groups amongst the  $g$  groups.

**FIRST COMMENT** : If you want to examine bivariate relationships between two particular groups (say  $a$  and  $b$ ), then just see previous 4 pages and replace 1 and 2 with say  $a$  and  $b$ , and you're good to go . . .

**SECOND COMMENT** : Suppose instead you want to examine relationships between a particular group  $a$  and **All Other Groups!**



We define  $G$  of one to any other type as

$$G_{a\bullet}(r) = \Pr(\text{Dist}(\text{rnd group } a \text{ event to nearest event in any other group}) \leq r)$$

If each group is a CSR process, then as before we have

$$G_{a\bullet}(r) = 1 - e^{-\lambda_{\bullet}\pi r^2}, \quad r \geq 0$$

where  $\lambda_{\bullet} = \sum_{a=1}^g \lambda_a$

*Basically, this is the percent of observed nearest neighbor other group distances (i.e. from  $a$  to any other group) that are less than  $r$*



$F_{\bullet}(r) = \Pr(\text{Dist}(\text{rnd location to any group event}) \leq r)$   
(*basically a useless function since it means we're just putting all points in one group – so just use regular  $F$  function*)



$$J_{a\bullet}(r) = \frac{1 - G_{a\bullet}(r)}{1 - F(r)}$$



$$K_{a\bullet}(h) = \frac{E(\# \text{events in any other group within } h \text{ of a random group 1 event})}{\lambda_{\bullet}}$$



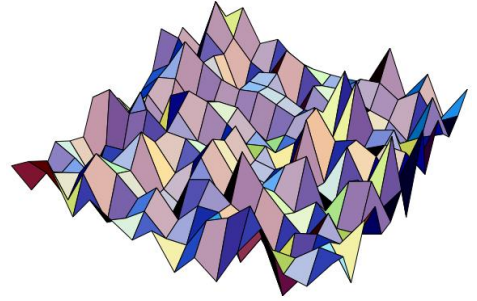
$$L_{a\bullet}(h) = \sqrt{\frac{K_{a\bullet}(h)}{\pi}}$$



**Multi Group Comparison Functions in R.** In the script `MarkedSPP.R.txt` are a few examples of multi group functions. <http://reuningscherer.net/fes781/rscripts/markedsprr.txt>  
The R multi group functions are `Gdot`, `Kdot`, `Ldot`, and `Jdot`

# Point Process Models

**Cressie (1993)** – I have to say, this is dense, but in some ways still the best overview of what is available. Section 8.5.



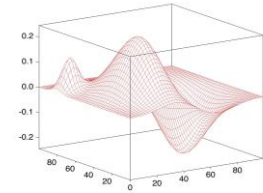
**Diggle (2014)** – See Chapter 6. A good overview

**Baddeley (2015)** – See Chapters 9 and 10. Awesome.

**Spatstat package** – there is actually a wealth of information in the 900+ page spatstat users manual : <http://www.spatstat.org/spatstat/>

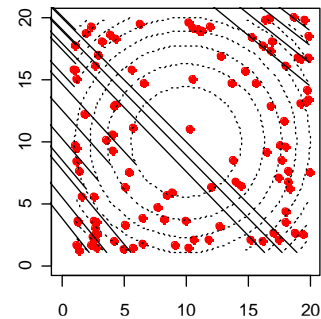
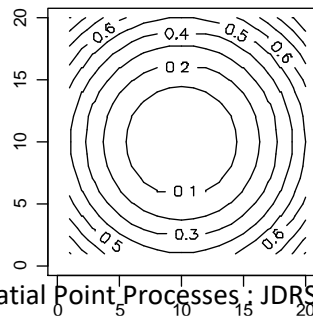
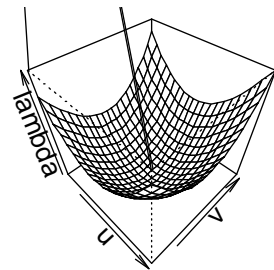
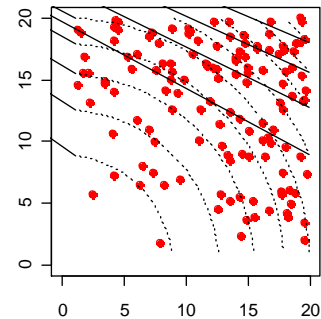
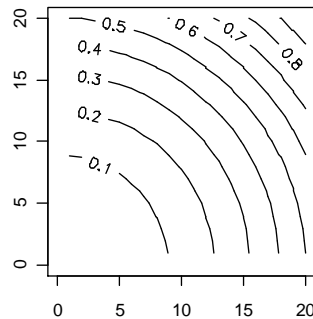
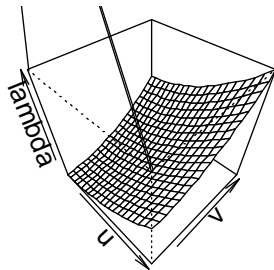
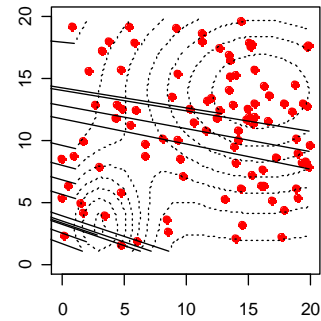
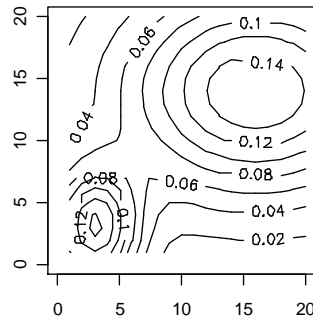
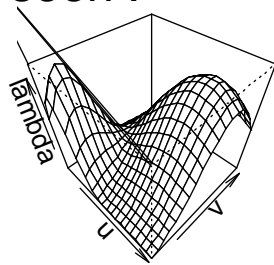
CSR is a rather limited 'null hypothesis', and many of the procedures/functions we've seen are mostly a rejection of CSR. What about some other models, or other hypotheses?

# Heterogenous Poisson PP



- Perhaps the simplest alternative to the CSR, constant intensity  $\lambda$  Poisson PP.
- We already defined a **heterogenous Poisson point process** with **intensity function**  $\lambda(s)$  (think mean), where  $s$  is any point in our study region  $R$ , using the following criteria :
  - 1) The number of events in any region  $A \subset R$  has a Poisson distribution with mean  $\int_A \lambda(s) ds$
  - 2) Given that a total of  $X = N$  events occur in  $A$ , the locations of the events are a random sample of  $N$  events sampled proportional to  $\lambda(s)$

Let's revisit three heterogenous point processes we've already seen :





In these instances, we **KNOW** the underlying intensity functions  $\lambda(s)$  : however, how do we generate a point process with this intensity function? Here's the method proposed by Lewis and Shedler (1979) :

- 1) Determine  $\lambda_{\max}$ , the maximum intensity over the entire study region  $R$ .
- 2) Generate a homogenous point process with intensity  $\lambda_{\max}$  over  $R$
- 3) For a generated point at spatial location  $s$ , keep the point with probability  $\frac{\lambda(s)}{\lambda_{\max}}$ . Determine this independently for each point.



**Inhomogenous Poisson PP in SPATSTAT.** Examples of two of the three heterogenous poisson pp above are available here :

[http://reuningscherer.net/fes781/rscripts/heterogenous\\_poisson\\_pp.txt](http://reuningscherer.net/fes781/rscripts/heterogenous_poisson_pp.txt)

The first example (two multivariate normal distributions) uses the method described above. However, if the underlying intensity function  $\lambda(s)$  is known, you can use the `rpoispp( )` function. This function also allows you to input a grid image where image height equals density function height (i.e. results from kernel smoothing, etc).

## NOW :

- Usually, all we get to see is the data locations, and we want to estimate  $\lambda(s)$ .
- We've already discussed how to use quadrat counts of kernel smoothing to get a non-parametric estimate  $\hat{\lambda}(s)$
- However, if we specify some family of models, we could try to **fit a parametric model for  $\lambda(s)$ !!**

# Parametric Models for $\lambda(s)$

- Use **maximum likelihood** to fit a model  $\lambda_\theta(s)$  with parameter vector  $\Theta$ .
- Need to make sure  $\lambda_\theta(s)$  is non-negative, so need some constraints (see Cressie p. 655)
- Some typical models (from Baddeley) :



**Homogenous Intensity** :  $\lambda_\theta(s) = \lambda$  (CSR)

**Homogenous by Region** :  $\lambda_\theta(s) = \lambda_{R_j}$  where

$R_1, R_2, \dots, R_m$  are distinct (and non-overlapping) regions (like counties or states)





**Intensity by Baseline** :  $\lambda_{\theta}(s) = \theta b(s)$  where  $b(s)$  is a 'baseline' function defined at every point in space (such as underlying population density). This might be appropriate to model density of disease cases ( $\theta$  is the probability a particular person gets a disease).

**Exponential Function of Covariate** :

$\lambda_{\theta}(s) = \exp(\beta_0 + \beta_1 Z(s))$  where  $Z(s)$  is a spatial covariate (such as slope or distance to pollution source, etc).

**Raised Incidence Model** (just combine previous two

models) :  $\lambda_{\theta}(s) = b(s) \exp(\beta_0 + \beta_1 Z(s))$

**General Loglinear Model** :

$\lambda_{\theta}(s) = \exp(b(s) + \beta_1 Z_1(s) + \beta_2 Z_2(s) + \dots + \beta_p Z_p(s))$

# Residuals

There has been work on creating '**residual**' plots for these models to evaluate model fit. Four types of residuals are discussed. The paper to read is :

[Baddeley, A., Turner, R., Moller, J. and Hazelton, M. \(2005\) Residual analysis for spatial point processes. Journal of the Royal Statistical Society, Series B67, 617–666.](#)

## Raw Residuals

- Choose a set of unobserved locations in the study region  $R$ . This can be random, or can be a grid (this is the default in spatstat). The number of points is the area of  $R$  times the estimated overall intensity  $\hat{\lambda}$  times a constant  $K$  (usually  $K=4$ ).

- Create the **quadrature** : the combination of the original locations and the new unobserved locations.
- Calculate the 'raw residual' at every point  $s$  in the quadrature :

$$r(s) = z(s) - w(s)\hat{\lambda}_\theta(s)$$

$z(s) = 1$  if an observed point, 0 if an unobserved point

$\hat{\lambda}_\theta(s)$  = the model predicted intensity at point  $s$

$w(s)$  = weight (based on the number of quadrature points per grid area, or based on area of dirichlet tessellations)

**Example** : suppose  $\hat{\lambda}_\theta(s) = 1$ , we have 100 observed points, and 400 unobserved points over a 10x10 region (so 500 quadrature points).

$$w(s) = \frac{1}{5}, \text{ so}$$

$$r(s) = 1 - \frac{1}{5} * 1 = \frac{4}{5} \text{ for 100 observed points and}$$

$$r(s) = 0 - \frac{1}{5} * 1 = \frac{-1}{5} \text{ for 400 unobserved points}$$

**On average, the residuals will sum to ZERO!!!!**

.....

***For other types of residuals, see paper above . . . .***





## Parametric Inhomogenous Poisson Models in SPATSTAT.

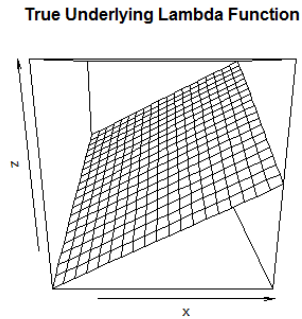
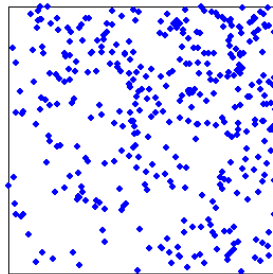
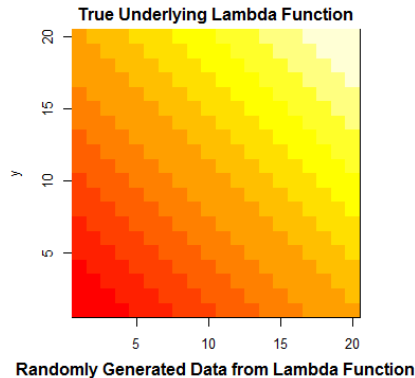
Most general is the `ppm()` function (poisson process model). This fits polynomials of various orders based on spatial location. Can also include covariates. Includes various border correction methods (as we've discussed). Can use maximization methods besides MLE. For residuals, use `plot.msr()` and `diagnose.ppp()`

Examples of two of the three heterogenous poisson pp above are available here, with models and residuals, comparison to kernel smoothing :

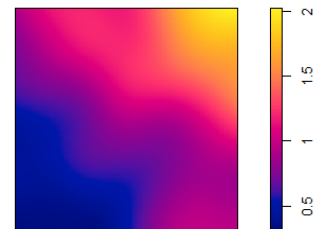
[http://reuningscherer.net/fes781/rscripts/heterogenous\\_poisson\\_pp.txt](http://reuningscherer.net/fes781/rscripts/heterogenous_poisson_pp.txt)

**Example in Depth :** To see how this process of model fitting works, and to examine the tools available in `spatstat`, we first look at a created example :

$$\lambda_{\theta}(s) = \theta b(s) = \frac{(x + y)}{20}$$



Kernel Smoothed Estimate of Lambda Function



## ***Model 1 : Fit a CSR model to this data (i.e. we fit a flat surface that we know is incorrect)***

Poisson process

Intensity: 0.9325

	Estimate	S.E.	CI95.lo	CI95.hi	Ztest	Zval
log(lambda)	-0.06988613	0.05177804	-0.1713692	0.03159696		-1.349725

*This estimates that  $\log(\hat{\lambda}_\theta(s)) = -0.07$  (i.e. a constant)*

*so  $\hat{\lambda}_\theta(s) = 0.93$ . Note that this is a log-linear model (i.e. it's fit on the log-scale)*

## ***Residuals :***

Scalar-valued measure

Approximated by 1976 quadrature points

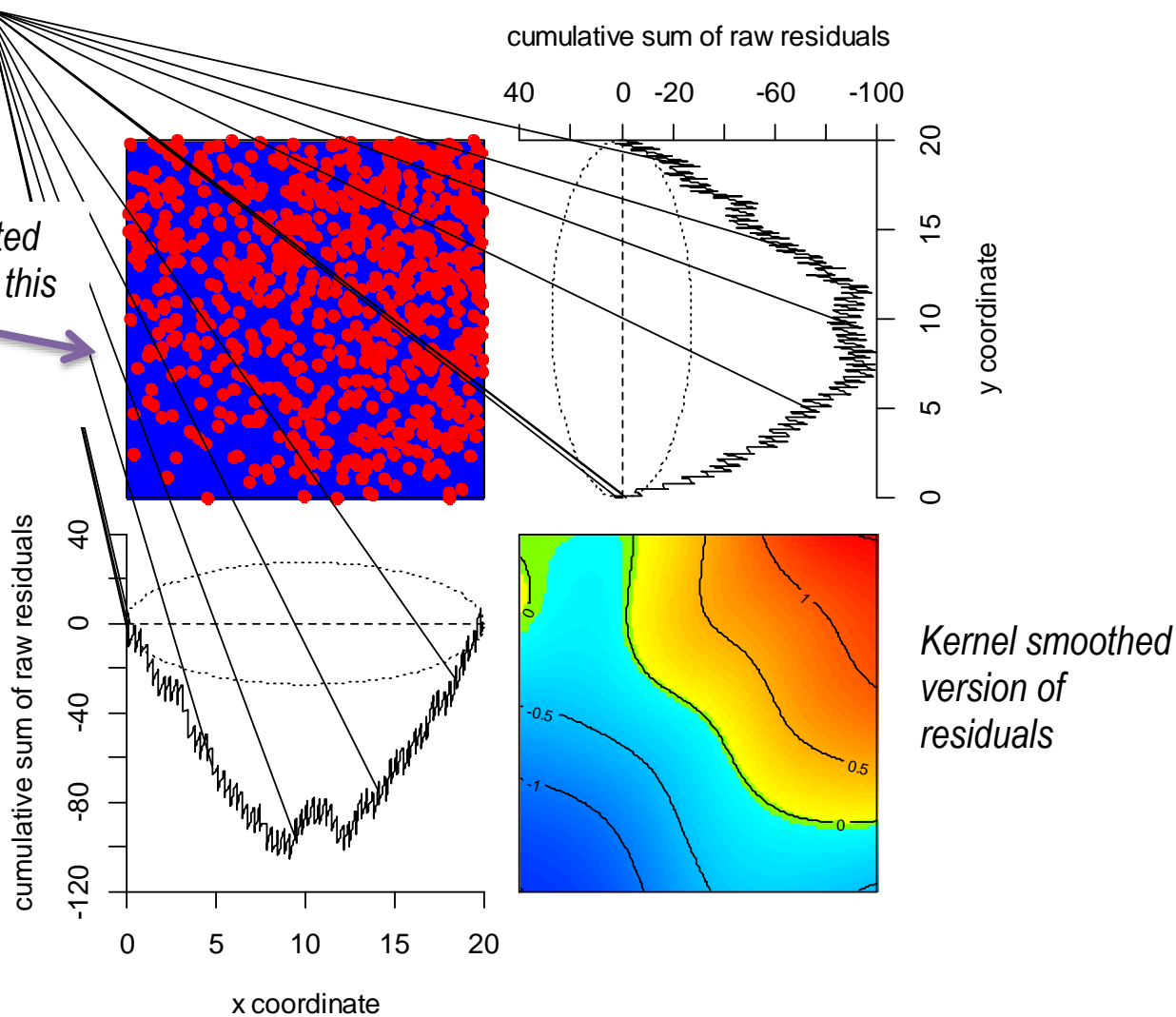
window: rectangle = [0, 20] x [0, 20] units

372 atoms

Total mass:

discrete = 372      continuous = -372      total = -8.8402e-15

*Observed data plotted above predicted (in this case flat/constant) model*



# Evaluating Model Fit

- 1) **Chi-Square Test using Quadrat Counts** – as we discussed previously, we can subdivide the region  $R$  into quadrats (or quantiles) of equal size. We can then compare observed count vs. the model predicted count in each region and perform a Chi-square Test!

*Aside – I figured out that the values reported in the plot below are the Pearson Residuals :*

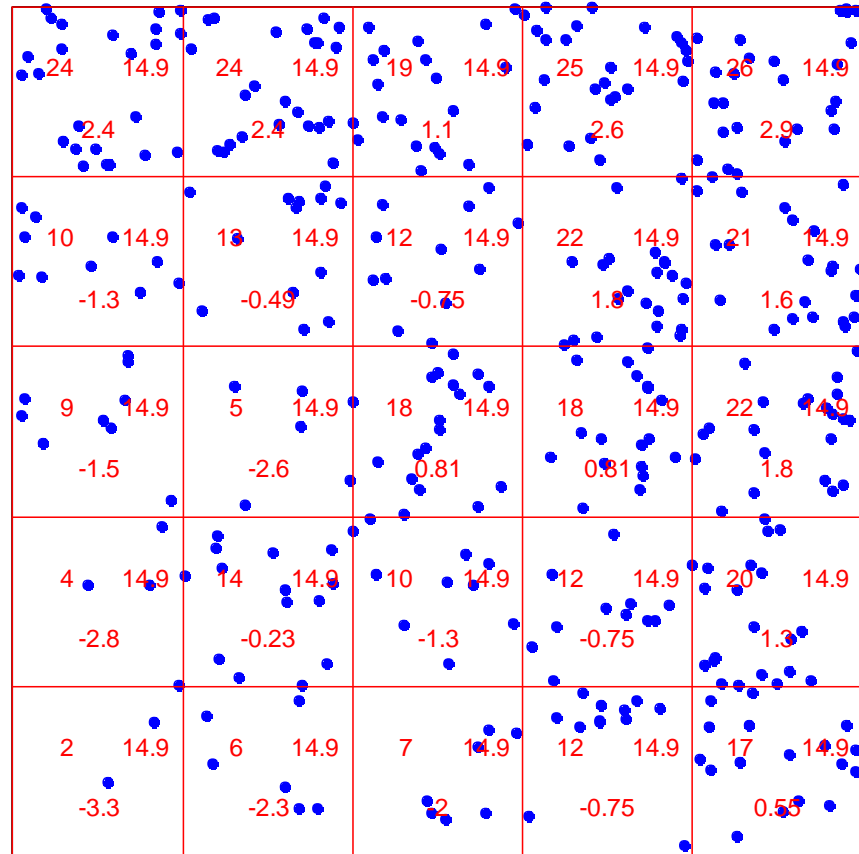
$$\text{Pearson Residual} = \frac{\text{Observed N} - \text{Expected N}}{\sqrt{\text{Expected N}}}$$

```
Chi-squared test of fitted Poisson model 'modell' using
quadrat counts          Pearson X2 statistic
data:  data from modell
X2 = 82.839, df = 24, p-value = 4.268e-08
alternative hypothesis: two.sided
```

Quadrats: 5 by 5 grid of tiles

*Model doesn't fit  
very well . . .*

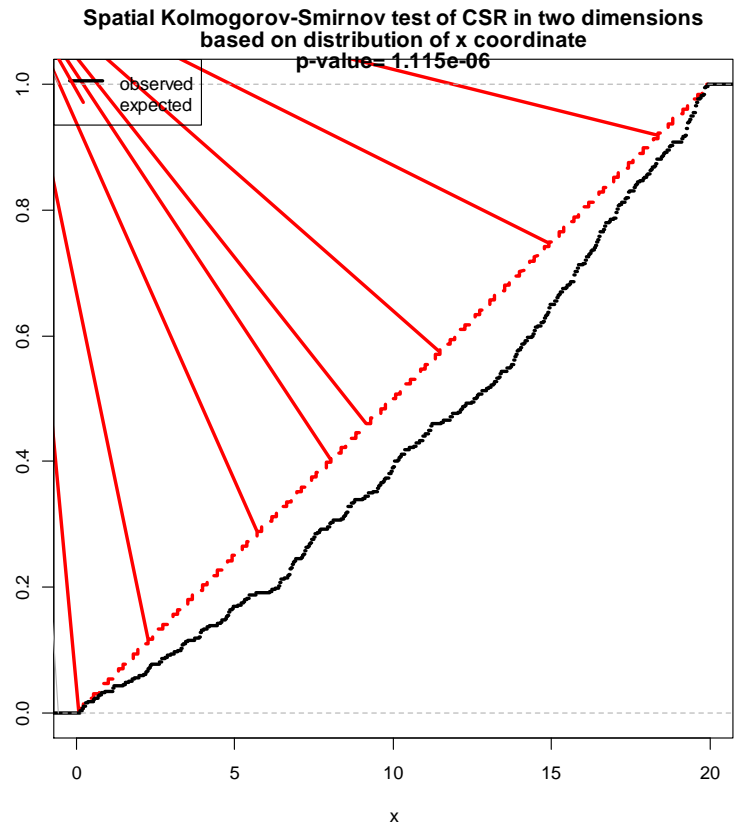
### Chi-Square Test Assuming CSR



- 2) **Kolmogorov-Smirnov Test** – this is a widely used test to compare the empirical CDF to the modeled CDF. Essentially, the test is based on the maximum distance between the two curves. This can be evaluated in the x direction or the y direction, OR based on a covariate (more on this later . . . .)

Spatial Kolmogorov-Smirnov test of CSR in two dimensions

```
data: covariate 'x' evaluated at
points of 'linearpp'
      and transformed to uniform
distribution under CSR
D = 0.13912, p-value = 1.115e-06
alternative hypothesis: two-sided
```



## ***Model 2 : Fit a model increasing linearly in X and Y (which is the true model in this case)***

Nonstationary Poisson process

Log intensity:  $\sim x + y$

Fitted trend coefficients:

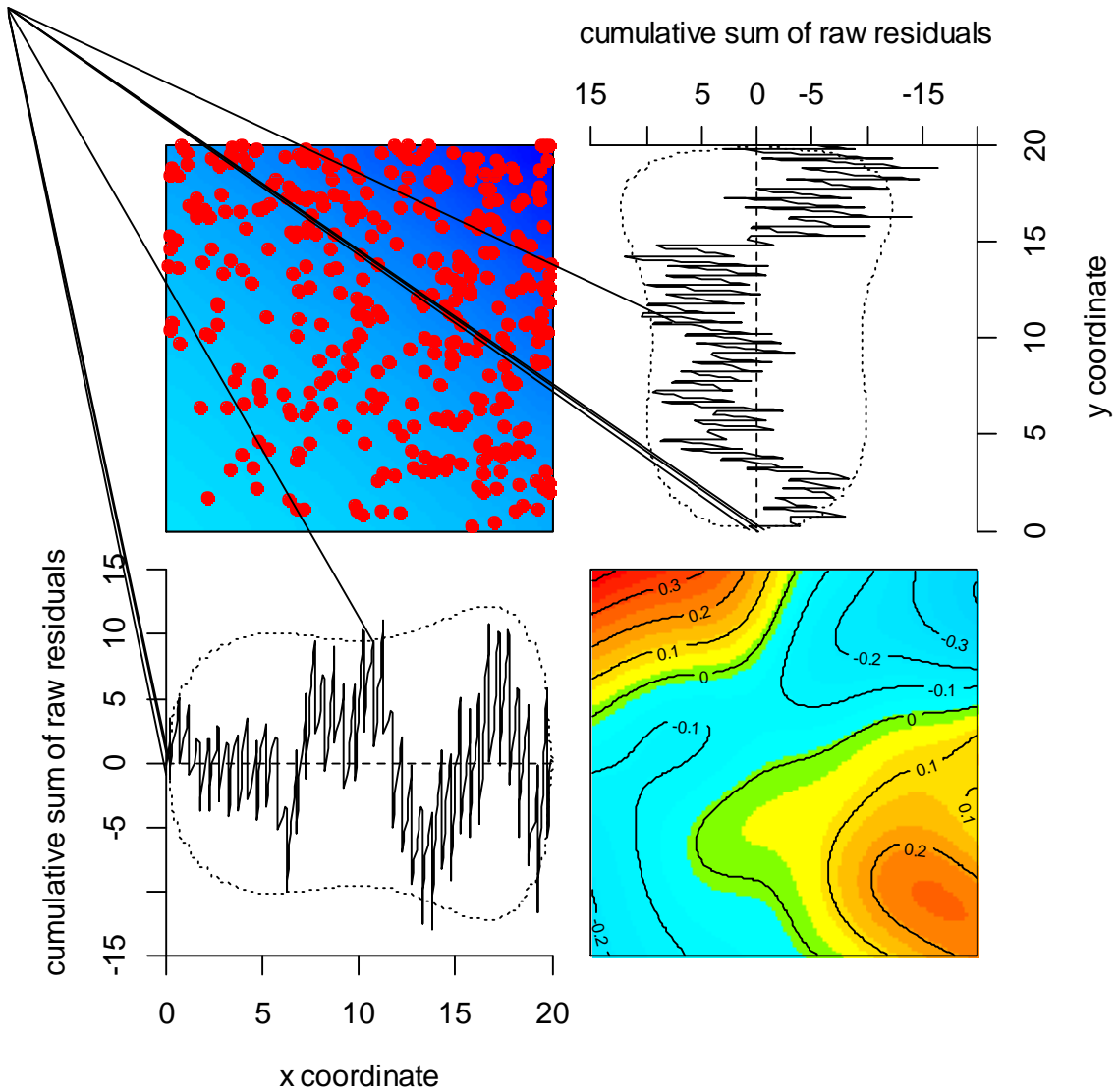
(Intercept)	x	y
-1.23745315	0.04945971	0.05753418

	Estimate	S.E.	CI95.lo	CI95.hi	Ztest	Zval
(Intercept)	-1.23745315	0.162068478	-1.55510153	-0.91980477	***	-7.635372
x	0.04945971	0.009202413	0.03142332	0.06749611	***	5.374646
y	0.05753418	0.009277379	0.03935085	0.07571751	***	6.201556

***Notice that X and Y are both significant predictors***

***Residual on next page – notice that magnitude of errors is much smaller, cumulative residuals in X and Y are both within expected bounds***





Chi-squared test of fitted Poisson model 'model2' using  
quadrat counts

Pearson X2 statistic

data: data from model2

X2 = 27.887, df = 22, p-value = 0.3589

alternative hypothesis: two.sided

Quadrats: 5 by 5 grid of tiles

***Chi-Square Test is non-significant. Note in plot below that  
predicted values are NOT all the same for each cell!***

Spatial Kolmogorov-Smirnov test of inhomogeneous Poisson  
process in two dimensions

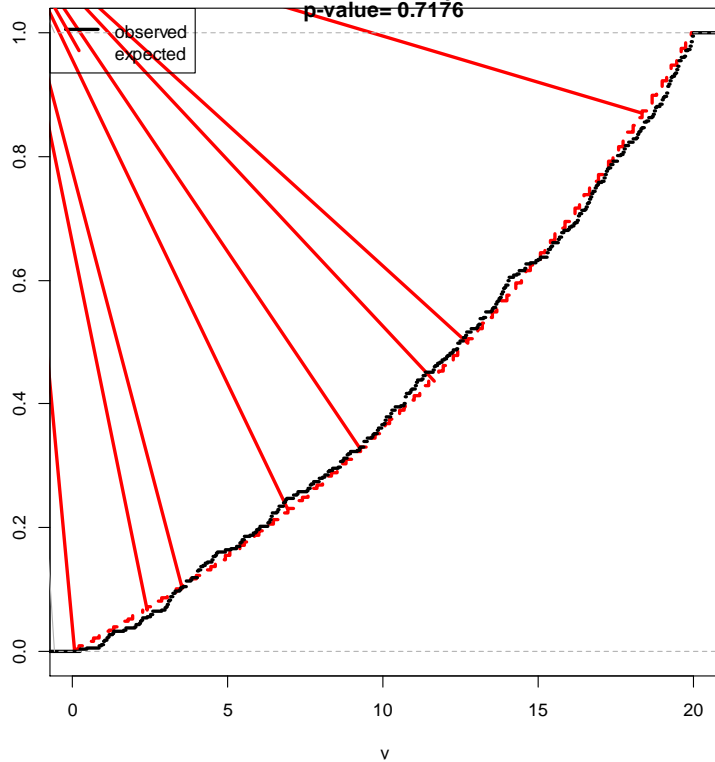
data: covariate 'y' evaluated at points of 'linearpp'  
and transformed to uniform distribution under  
'model2'

D = 0.036096, p-value = 0.7176

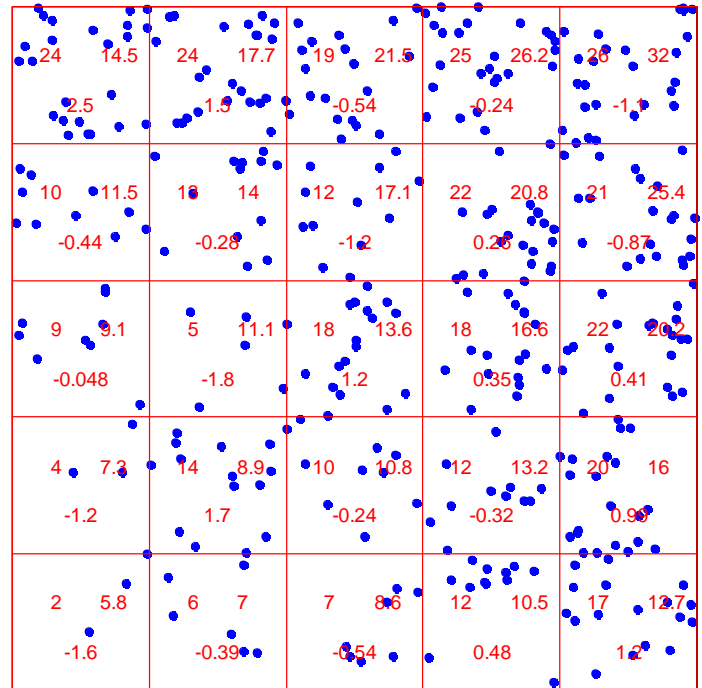
alternative hypothesis: two-sided

***Chi-K-S Test is also non-significant – models seems to FIT!***

Colmogorov-Smirnov test of inhomogeneous Poisson process in two dim based on distribution of y coordinate



Chi-Square Test for Linearly Increasing Model



# Comparing Models

Since we're using maximum likelihood, we can calculate the Akaike Information Criteria (AIC) ( $p$  is the number of estimated parameters in the model) :

$$AIC = 2p - 2\ln(L_{\theta})$$

**Bottom Line : SMALLER IS BETTER**

***Example :** We have a flat CSR model and a linearly increasing in  $X$  and  $Y$  model. Second model is better!*

AIC Model 1 : 799.9926

AIC Model 2 : 734.6801

# Comparing Nested Models



If we compare two NESTED models (i.e. all the terms in model 1 are also in the more complicated model 2), we can actually do a **DEVIANCE Likelihood Ratio Test** –

$$D = 2(\ln(L_{\theta_2}) - \ln(L_{\theta_1}))$$

If the models fit equally well, then  $D$  should have a Chi-Square distribution with degrees of freedom equal to the difference in parameters between model 1 and 2.

**If the  $D$  is too large, then model 2 is better than model 1!**

**Example** : Deviance is 69.3 on 2 degrees of freedom.  
Reject hypothesis that models are equally good - second model is better!

Analysis of Deviance Table

```
Model 1: ~1                      Poisson
Model 2: ~x + y                  Poisson
  Npar Df Deviance Pr(>Chi)
1     1
2     3  2    69.313 8.892e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
' ' 1
```

.....

*In online code, example of overfitting same model*

# Poisson Model with Covariate

***Example*** : BEI data (in spatstat package). See Results in Program

# K and L functions for Inhomogenous Poisson Point Processes



[A. Baddeley, J. Møller and R. Waagepetersen \(2000\). Non- and semi-parametric estimation of interaction in inhomogeneous point patterns. Statistica Neerlandica, 54:329-350.](#)

- Article above proposes  $K_I(h)$  based on an inhomogenous process.
- Assumed is an inhomogenous Poisson point process with mean function  $\lambda(s)$  defined on a region  $R$ .
- Process is assumed to be 'reweighted second-order stationary' – this means simply that after you subtract the underlying mean function, points are neither clustering nor repelling (regular).



- $K_I(h)$  is still

$$K_I(h) = \frac{E(\# \text{events within } h \text{ of a random event})}{\lambda(s)}$$

- Non-homogenous Poisson PP still has  $K_I(h) = \pi h^2$  as before

- **New Estimator for  $K_I(h)$ :**

$$\hat{K}_I(h) = \sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N \frac{I(d(i, j) < h)}{\hat{\lambda}(s_i)\hat{\lambda}(s_j)}$$

Essentially, this means add up  $\frac{1}{\hat{\lambda}(s_i)\hat{\lambda}(s_j)}$  for all points less than  $h$  apart.

**This means we need an estimated intensity function  $\hat{\lambda}(s)$  : use either kernel smoothing or parametric model discussed above**

(spatstat uses kernel smoothing by default)

As before, we can modify  $K_I(h)$  to get a nicer reference function  $L_I(h)$  :

$$\hat{L}_I(h) = \sqrt{\frac{\hat{K}_I(h)}{\pi}} - h$$

and  $L_I(h) = 0$  (flat line) for an inhomogenous Poisson PP.



**Inhomogenous K function in R.** First, you can use functions I wrote previously for making nice K and L plots : just include the following line in your R code

```
source("http://reuningscherer.net/fes781/rscripts/kandlfunctions.r.txt")
```

I've put up examples for two non-homogenous point processes discussed above and for several species in Tim's tree data :

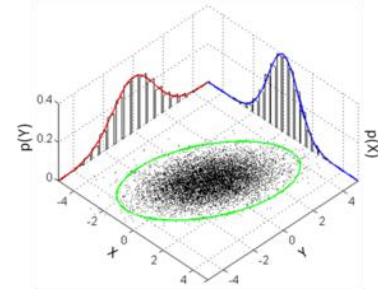
[http://reuningscherer.net/fes781/rscripts/heterogenous\\_poisson\\_pp.txt](http://reuningscherer.net/fes781/rscripts/heterogenous_poisson_pp.txt)

The R function in the spatstat package you need is `kinhom( )`

.....

Results in R – Maine tree data, made up processes. What was a clustered process, now starts to look like a regular or inhibited process!

# Other Models for Point Processes



## 1) Poisson Cluster Processes

Originally started as model for location of insect larvae; also used in bacteriology. Relevant when modelling tree seedling data as well. Sometimes called a **Neyman-Scott process** (1958 paper)

### Three part process :

- 1) Choose location of cluster centers according to a Poisson process with intensity  $\lambda(s)$  (centers could be egg mass locations, pine cone locations, tree fall

locations, etc.). **Note : this could be either a homogenous or inhomogenous process.**

- 2) At each cluster center, choose a random number of offspring  $S$  according to some distribution  $p_S$  ( $p_S$  could be Poisson, uniform, anything)
- 3) Distribute the  $S$  offspring points according to some bivariate distribution  $f(\cdot)$  (often a multivariate normal or Cauchy)

The original points are often included in the final count so that there is at least one point at each cluster location (in case there are no children)

## 2) Cox Process

- Another way to model a clustered process
- **Idea** – clusters form because of random environmental heterogeneity.
- Assume the intensity function  $\lambda(s)$  is a random stochastic process (i.e. not some nice polynomial, but a process with random high and low points across space)
- Under certain assumptions, a Cox process is equivalent to a Poisson Cluster Process (see Cressie, p. 663)



**Cluster Processes R.** The main function is `kppm` – this fits Poisson cluster models, Cox models, and several others. I tried this out in

[http://reuningscherer.net/fes781/rscripts/heterogenous\\_poisson\\_pp.txt](http://reuningscherer.net/fes781/rscripts/heterogenous_poisson_pp.txt)

.....

The models below are discussed in Cressie in some detail. Still figuring out which ones can be done in SPATSTAT.

**3) Simple Inhibition Point Process**

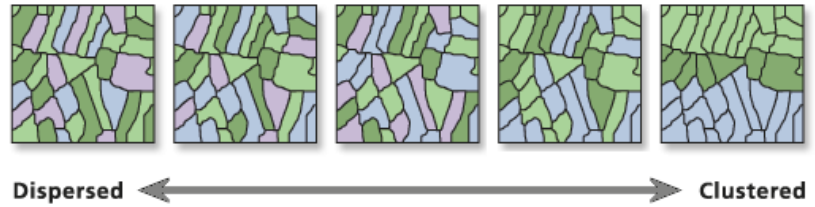
**4) Gaussian Cox Processes**

**5) Markov Point Processes**

**6) Thinned Processes**



# Spatial Association, Clustering, and Autocorrelation



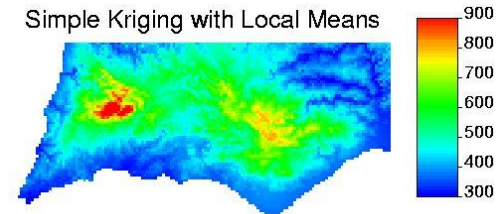
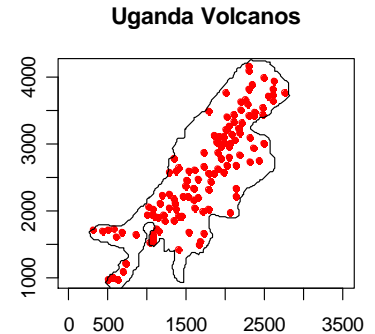
**TIM GREGOIRE** gets credit for much of what is in these notes!

Other places to read :

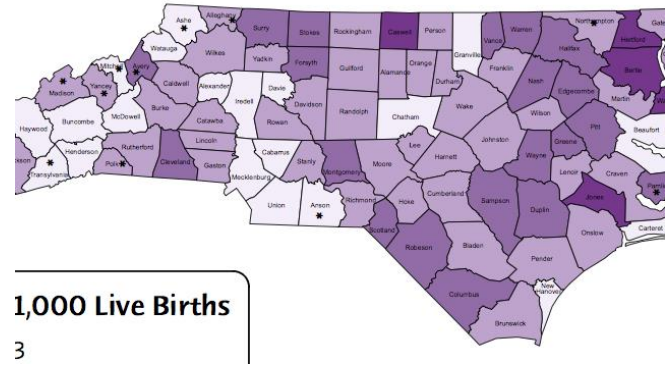
- *Bailey & Gatrell, Chapter 7*
- *Haining (2003), Chapter 8*
- *Waller and Gotway, Chapter 7 (esp. 7.4)*
- *ESRI Guide to GIS vol 2. (2005) – Chapter 3*
- [Griffith 2009 article ‘Spatial Autocorrelation’](#) – nice overview of issues
- Great set of notes from Ron Briggs at U Texas Dallas <http://www.utdallas.edu/~briggs/henan/>

**FIRST** – a reminder of three types of spatial data we consider

- **Spatial Point Patterns** (what we just finished) – fixed location of events (trees, volcanoes, diseases)
- **Spatially Continuous Data** (soil moisture, pollution concentration, income, etc) : Data distributed on a fixed, continuous domain, often termed geostatistical data. Attribute can be measured at an infinite number of locations – this is the topic after this one!



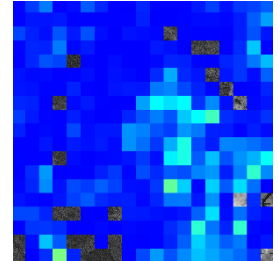
- **Lattice Data** (*area data, regional data*) : Arises from measurements on a fixed, discrete domain, such as counties, countries, census tract regions, pixels. The domain comprises a countable set of locations (subregions) on each of which a measurement may be made of an attribute which pertains to the entire subregion, and not to any particular point within it – diseases cases per county, average income per census block, etc.



This data is our current focus!

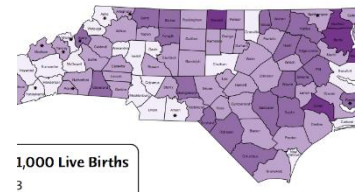
*“Measures of spatial autocorrelation originated from the consideration of data on lattices.”* (Schabenberger & Gotway, 2005, p. 19) “

**Our Goal** : examine associations among the measured values in adjacent elements of the lattice (including how to quantify such a measure)



## Lattice Shapes

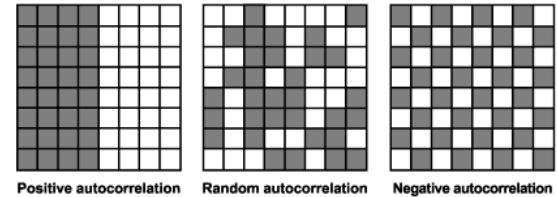
- **Regular** : all areas same size and shape. Includes pixel images.
- **Irregular** : counties, census tracts, property boundaries.



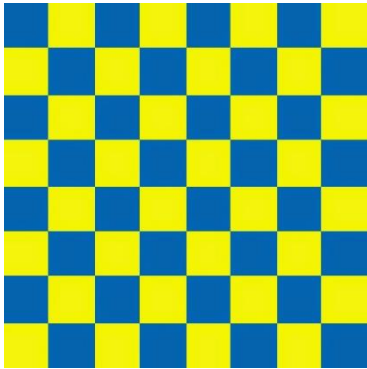
**Distance** : a bit ambiguous with lattice data, especially if irregular. However, some measures of spatial association don't require distance.

## So what is 'Spatial AutoCorrelation'?

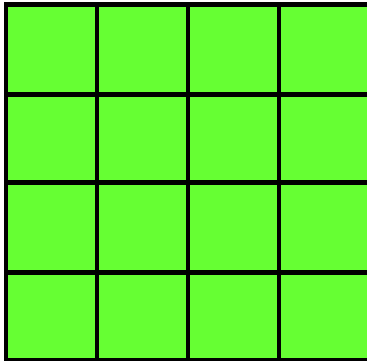
[Griffeth \(2009\)](#) (page 2-2) describes some common definitions :



- Self-correlation attributable to the geographical ordering of data
- A conspicuous map pattern/trend (extreme is all values the same – perfect spatial autocorrelation)
- A diagnostic tool for spatial model misspecification (or missing variables)
- A spatial process mechanism (i.e. a weather front moving across a landscape, the spread of a disease from a central starting location)
- A nuisance parameter when applying traditional stat techniques to spatial data
- An 'outcome of areal unit demarcation' (**Modifiable Area Unit Problem** – more on this later). However, here's an example :



**FIRST** – define a ‘unit’ as a 1x1 square. Here, we have perfect **negative** spatial auto-correlation (think of the blues as +1, and the yellows as -1). A yellow is always next to a blue on all sides – neighbors are perfectly ‘negatively correlated’



**NOW** : Change our definition of a ‘unit’ – make this a 2x2 square. The ‘average’ color is now green for all adjacent units (2x2 squares) – this is perfect **positive** spatial auto correlation!

## Math Definitions

$R$  The entire domain under consideration (eg an entire state)

$A_i$  A sub-region of  $R$  (like a county)

$N$  The total number of sub-regions in  $R$

$i, j$  Indices keeping track of the sub-regions ( $1 \leq i, j \leq N$ )

Note that we assume that

$$R = \bigcup_{i=1}^N A_i \text{ (the union of all sub-regions makes } R \text{)}$$
$$A_i \cap_{i \neq j} A_j = \emptyset \text{ (no two subregions overlap)}$$

$w_{ij}$  Spatial Connectivity Weights – a measure of spatial proximity (for example, might be Euclidean distance from centers, or might be 1 if  $A_i$  and  $A_j$  are ‘connected’ or ‘adjacent’, 0 otherwise)

$sim_{ij}$  Some measure of Similarity between  $A_i$  and  $A_j$

$Y_i$  Any continuous variable measured in each sub -region  $A_i$  (*average age, total disease cases, average income, etc*). **THIS IS THE DATA!**



# Measuring Spatial Association / Clustering / Autocorrelation

- Not easy to define what this means
- **Main goal – come up with a spatial analogue to the usual ‘bivariate’ non-spatial correlation  $r$ .**
- Waller and Gotway, 7.4 :

A ***global index of spatial autocorrelation*** provides a summary over the entire study area of the level of spatial similarity observed among neighboring observations.

- This (naturally) leads to the idea of a local index of spatial autocorrelation which can vary over  $R$

## Global Index of Spatial Autocorrelation – General Form = Global Cross Product Statistic!

$$\Gamma = \frac{\sum_{i=1}^N \sum_{j=1}^N w_{ij} sim_{ij}}{w_{..}}$$

Where  $w_{..} = \sum_{i=1}^N \sum_{j=1}^N w_{ij}$ , i.e. the total sum of ALL pairs of weights.

(Usually assume  $w_{ii} = 0$ )

**Local Index of Spatial Autocorrelation – General Form = Local Cross Product Statistic!** (*i.e. calculated separately for each subregion  $A_i$  .*)

$$\Gamma_i = \frac{\sum_{j=1}^N w_{ij} sim_{ij}}{w_{..}}$$

**Different choices for  $w_{ij}$  and  $sim_{ij}$  distinguish the various spatial autocorrelation measures**

**Commonly used measures of spatial weighting functions**  
 (see p. 261, B&G; pp. 224-225, W&G) *(not an exhaustive list)*

$$\text{i) } w_{ij} = \begin{cases} 1, & \text{if } \mathcal{A}_i \text{ and } \mathcal{A}_j \text{ share a boundary} \\ 0, & \text{otherwise} \end{cases} ;$$

$$\text{ii) } w_{ij} = \begin{cases} 1, & \text{if } \mathcal{A}_j \text{ is one of } k \text{ nearest neighbors to } \mathcal{A}_i \\ 0, & \text{otherwise} \end{cases} ;$$

$$\text{iii) } w_{ij} = \begin{cases} 1, & \text{if } d_{ij} < \delta \text{ for some prescribed distance } \delta \\ 0, & \text{otherwise} \end{cases} ;$$

$$\text{iv) } w_{ij} = \begin{cases} 1/d_{ij}^\alpha, & \text{if } d_{ij} < \delta \text{ for some prescribed distance } \delta \\ 0, & \text{otherwise} \end{cases} \quad [\text{note: } \alpha = 1 \text{ or } 2, \text{ typically}] ;$$

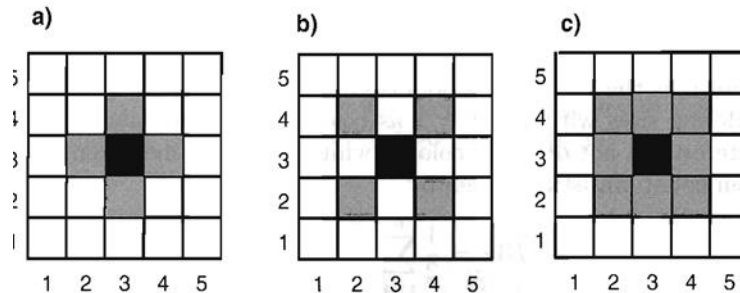
$$\text{v) } w_{ij} = \begin{cases} l_{ij}/l_i, & \text{where } l_{ij} \text{ is the length of common} \\ & \text{boundary, } l_i \text{ is perimeter of } \mathcal{A}_i \\ 0, & \text{otherwise} \end{cases} .$$

## Comments

- All but the last two are binary measures of proximity.
- Some authors discuss an  $N \times N$  proximity matrix of weights :  $\mathbf{W}_{N \times N} = [w_{ij}]$ .
- If all  $w_{ij}$  are 0 or 1,  $\mathbf{W}$  is called a **Binary Connectivity Matrix**, or an **Adjacency Matrix**
- $\mathbf{W}$  is symmetric in cases i), iii), and iv)
- Distance-based measures of proximity iii) and iv) presume some point measure of location (e.g., a centroid) when dealing with area-based (lattice) data.

## Assessing adjacency – Who's Touching!

There are various ways to define a shared boundary, illustrated most simply for regular lattices, as in the following figure (Schabenberger & Gotway (2005).) *Note : in this regular lattice case, each square represents a subregion  $A_i$*



These simple contiguity schemes are named after chess moves:

- a) adjacency is determined by the rook definition;
- b) the bishop definition is used;
- c) the queen definition is used.

In all three cases :

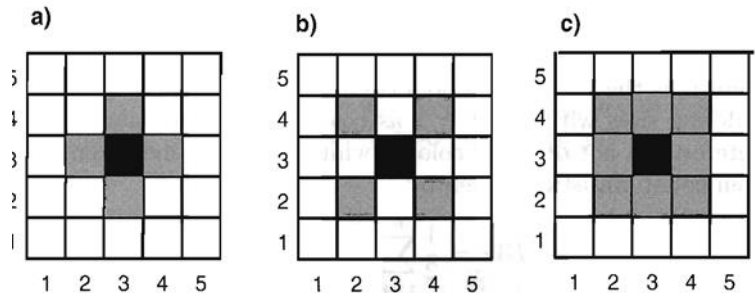
$$w_{ij} = \begin{cases} 1, & \text{if } \mathcal{A}_i \text{ and } \mathcal{A}_j \text{ share a boundary} \\ 0, & \text{otherwise} \end{cases}$$

**Example : 3x3 Grid** (that is assume  $R$  is the 3x3 grid in the center; ignore the other pixels)

$w_{\bullet\bullet} = 24$  with rook adjacency

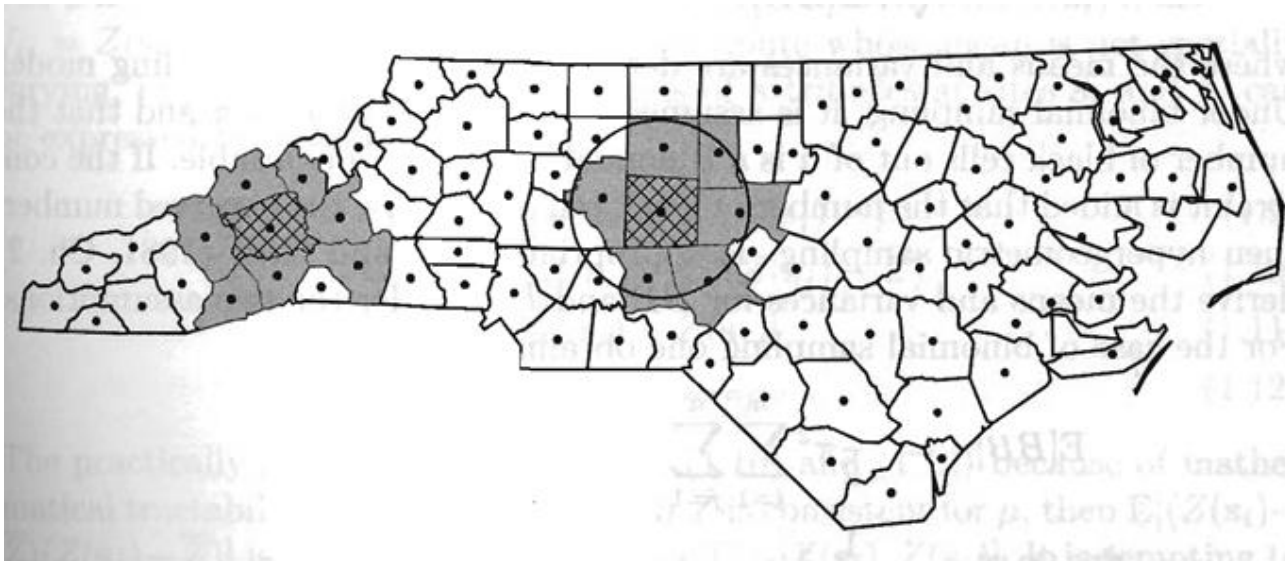
$w_{\bullet\bullet} = 16$  with bishop adjacency

$w_{\bullet\bullet} = 40$  with queen adjacency



**For irregular lattices**, such as the counties of the state of North Carolina Schabenberger & Gotway (2005), chess rules can still apply.

- Rook = neighbors have to have a common boundary of non-zero length
- Bishop = neighbors have a common point, but that's it
- Queen = Rook and/or Bishop





Towards the left is an illustrative example of weighting case *i*) definition for the proximity weight, namely

$$w_{ij} = \begin{cases} 1, & \text{if } \mathcal{A}_i \text{ and } \mathcal{A}_j \text{ share a boundary} \\ 0, & \text{otherwise} \end{cases}$$

Towards the right is an example of weighting case *iv*) based on distance separating centroids of  $A_i$  and  $A_j$ : any county within the radius of the circular area is a neighbor.

In any application, only one proximity rule would be applied!

# Global Indices of Spatial Autocorrelation



## A Global Index :

- Is a number which attempts to capture the strength of similarity of observations among the sub-regions,  $A_i$ , over all of  $R$
- Provides a summary measure of the degree to which similar observations tend to occur near each other (W&G, p. 223).
- Aims to mimic the familiar linear correlation coefficient which measures the strength of similarity between two variables.
- Only deals with a single variable, rather than a pair (hence, the “auto”) and its similarity over space.

# Moran's I

A measure of spatial clustering when  $Y$  is continuous (think avg income, soil moisture, etc).



Definition :

$$I = \frac{\sum_{i=1}^N \sum_{j=1}^N w_{ij} sim_{ij}}{w_{..}}$$

Where

$$sim_{ij} = \frac{(Y_i - \bar{Y})(Y_j - \bar{Y})}{Var(Y)} \text{ and } \bar{Y} = \frac{\sum_{i=1}^N Y_i}{N}$$

Or, substituting

$$I = \frac{\sum_{i=1}^N \sum_{j=1}^N w_{ij} (Y_i - \bar{Y})(Y_j - \bar{Y})}{w_{\bullet\bullet} \text{Var}(Y)}$$

**This looks a lot like a ‘spatial’ version of the Pearson correlation coefficient between two variables,  $X$  and  $Y$**

$$r = \frac{\frac{1}{N} \sum_{i=1}^N (X_i - \bar{X})(Y_i - \bar{Y})}{SD(X)SD(Y)} = \frac{\sum_{i=1}^N \sum_{i=1}^N (X_i - \bar{X})(Y_i - \bar{Y})}{NSD(X)SD(Y)}$$

where the weights are  $w_{ij} = \frac{1}{N}$

W&G (p.228) characterize  $I$  as 'a spatially weighted' form of  $r$ . Alas,  $|I|$  can exceed 1, so it's not a perfect analogue to  $r$

**Note :** Moran's  $I$  will take different values depending on how you define your weights  $w_{ij}$ . This will depend on two things:

- How adjacency is determined (who are neighbors – queen, rook, bishop, etc.)
- What weighting definition is used (binary, distance, etc).

### Properties of Moran's $I$

$I > 0$  (ish)      Neighboring regions tend to have similar values, and hence be clustered.

$I < 0$  (ish)      Neighboring regions tend to have different values.

## Mean and Variance of $I$

**Mean :**  $E[I] = -\frac{1}{N-1}$  (unfortunately, not zero)

**Variance :** this is harder than it looks

Case 1 : if we assume the  $Y_i$  are identically, independently, normally distributed (?!?!), then

$$\text{Var}[I] = \frac{N^2 S_1 - N S_2 + 3\omega_{\bullet\bullet}^2}{(N-1)(N+1)\omega_{\bullet\bullet}^2} - \left(\frac{1}{N-1}\right)^2$$

where

$$S_1 = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\omega_{ij} + \omega_{ji})^2 \quad \text{and} \quad S_2 = \frac{1}{2} \sum_{i=1}^N (\omega_{i\bullet} + \omega_{\bullet i})^2$$

This result was derived by Cliff and Ord (1973, 1981).

Case 2 : Assume that the  $Y_i$  are completely exchangeable (random permutation approach – which is how we get randomized tests.).  $Var[I]$  is similar – and messy!

---

## More on Weights

- $\mathbf{W}_{N \times N} = [w_{ij}]$  the weight matrix
- The  $i$ th row of  $\mathbf{W}$  is the total weights contributed by subregion  $A_i$ .
- Some  $A_i$  will naturally have more neighbors, and hence potentially total overall 'weight'
- May want to row standardize  $\mathbf{W}$  so that each subregion  $A_i$  has the same influence in the calculation of global indices (like Moran's I)
- However, can also leave  $\mathbf{W}$  alone and not worry about some subregions having more influence than others.



## **Moran Scatter Plots** (term coined by Luke Anselin : *Spatial Analytical Perspectives on GIS, 1995*)

If we row-standardize our weight matrix  $\mathbf{W}$  (make them all sum to one), we can write Moran's  $I$  in matrix form

$$I = \frac{\mathbf{Y}'_c \mathbf{W} \mathbf{Y}_c}{\mathbf{Y}'_c \mathbf{Y}_c} = (\mathbf{Y}'_c \mathbf{Y}_c)^{-1} \mathbf{Y}'_c \mathbf{W} \mathbf{Y}_c$$

Where  $\mathbf{Y}_c$  is the  $N \times 1$  vector of centered but not scaled  $Y_i$  values, i.e.  $(Y_i - \bar{Y})$ .

Anselin noted (regression lovers!) that this is slope of the line when we predict  $\mathbf{W} \mathbf{Y}_c$  based on  $\mathbf{Y}_c$ , i.e.  **$I$  is a slope!**

What is  $\mathbf{WY}_c$ ?

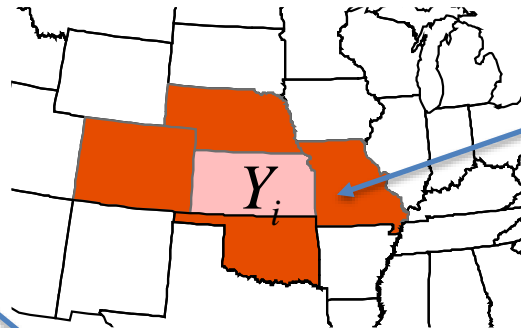
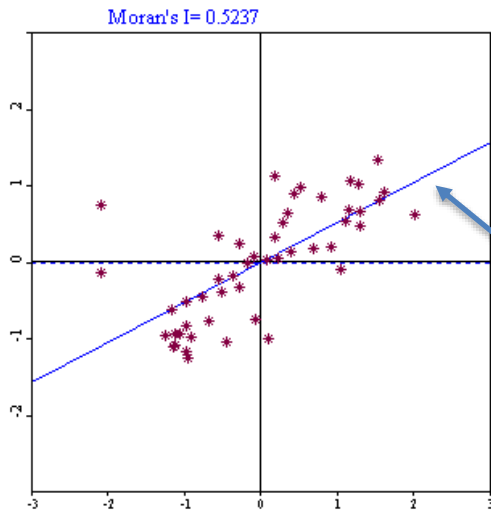
**These called the spatial lags – it's the weighted averages of the  $Y$  values for the neighbors of each subregion  $A_i$ .**

We'll call this  $lag(Y_i)$  (so  $\mathbf{WY}_c$  is the vector of all the lags)

**SO** : we can graphically plot  $\mathbf{WY}_c$  (the lags) vs.  $\mathbf{Y}_c$  in a **MORAN SCATTER PLOT.**

*Aside – dotted lines (horizontal and vertical) represent the mean values of  $\mathbf{WY}_c$  and  $\mathbf{Y}_c$ . The upper right and lower left quadrants defined by these lines are points with positive association between a location and it's spatial lag*

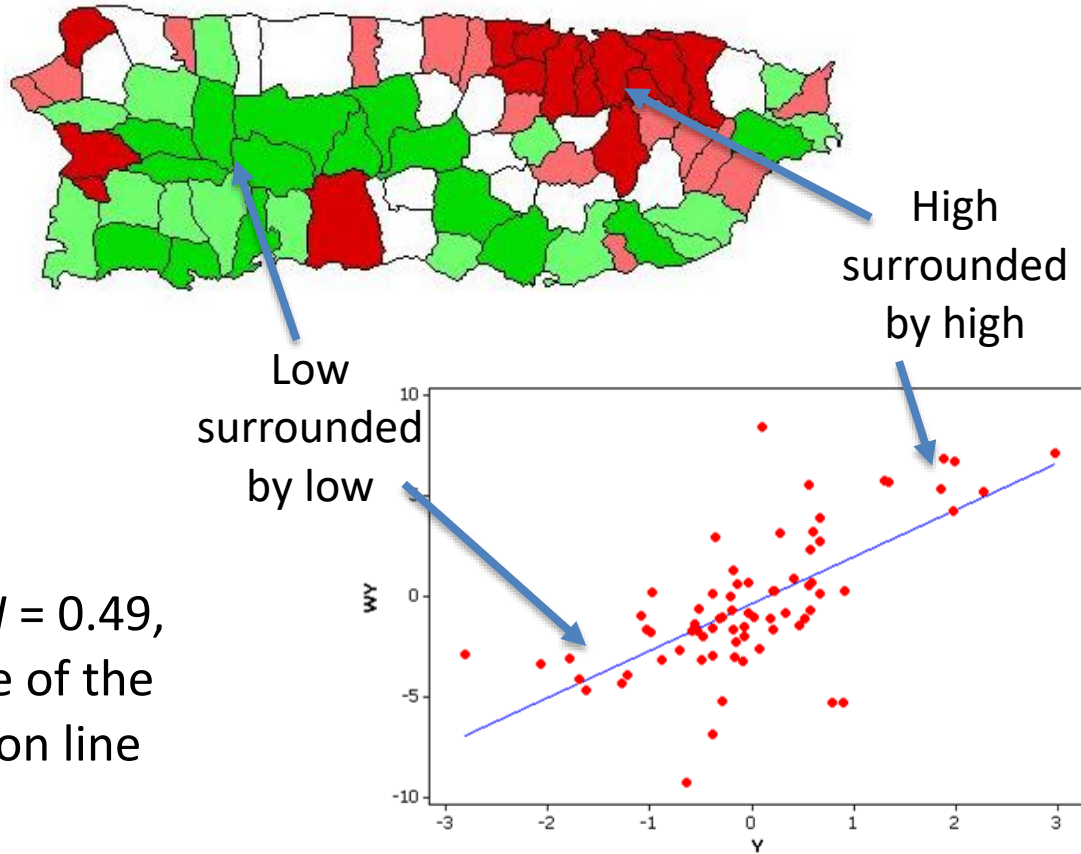
Moran's I can be interpreted as the correlation between variable,  $Y$ , and the **spatial lag** of  $Y$  formed by averaging all the values of  $Y$  for the neighboring subregions (polygons). We can then draw a scatterplot between these two variables (in standardized form):  $Y$  and  $lag(Y_i)$  (i.e.  $\mathbf{WY}_c$ )



$lag(Y_i)$  is the average of  $Y$  in surrounding subregions

Least squares "best fit" line to the points. The slope of this regression line is Moran's I!

## Example: Population Density in Puerto Rico



Moran's  $I = 0.49$ ,  
the slope of the  
regression line

## Geary's $c$ (Geary's contiguity ratio, index, etc).



Here's what Geary wrote in 1954 about the motivation for his statistic :

The problem discussed in this paper is to determine whether statistics given for each county in a country are distributed at random or whether they form a pattern."

Geary's index is 
$$c = \frac{\sum_{i=1}^N \sum_{j=1}^N w_{ij} sim_{ij}}{w_{..}}$$

Where

$$sim_{ij} = \frac{(Y_i - Y_j)^2}{2Var(Y)}$$

Or, substituting

$$c = \frac{\sum_{i=1}^N \sum_{j=1}^N w_{ij} (Y_i - Y_j)^2}{2w_{..}Var(Y)}$$

Handy facts :

$c$  ranges from 0 to 2

- 0 indicates perfect **positive** spatial correlation,
- 2 indicates perfect **negative** spatial correlation
- 1 indicates **no spatial autocorrelation** (i.e.  $E[c] = 1$ )

## Note :

- For Moran, the cross-product is based on the deviations from the mean for the two location values
- For Geary, the cross-product uses the actual values themselves at each location

$Var(c)$  depends on the distribution of the  $Y_i$ , and has been solved for  $Y_i$  normal and under the assumption of exchangeability.

Formulae are messy and not really informative. Computer takes care of this! (or see *Lee and Wong, 1<sup>st</sup> edition, p. 81 and p. 162*)

## $I$ vs $c$ (I c U!)

- $I$  is regarded as a ‘global’ measure across the entire region
- $c$  is considered to be better at quantifying local behavior, but is still a ‘global’ measure.

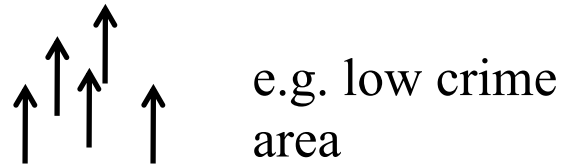
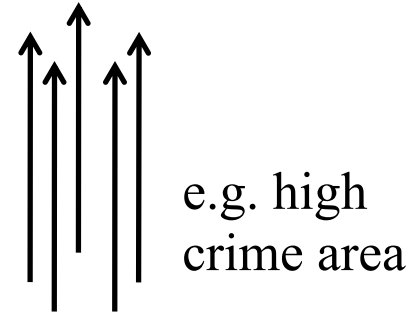
Both assume constant mean and variance (i.e. stationary spatial process).

If this is **NOT** true, it’s hard to rely on the interpretation of  $I$  or  $C$ .



# Hot Spots and Cold Spots

- **Hot Spot** : A place where high values cluster together
- **Cold Spot** : A place where low values cluster together



Moran's I and Geary's c cannot distinguish between them

- They only indicate clustering
- Cannot tell if these are hot spots, cold spots, or both

## Getis & Ord G statistic

- First, have to identify a distance band,  $d$ , within which clustering occurs.

$$G(d) = \frac{\sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N w_{ij} Y_i Y_j}{\sum_{i=1}^N \sum_{\substack{j=1 \\ j \neq i}}^N Y_i Y_j} \quad \text{where } w_{ij} = \begin{cases} 1, & \text{if } d_{ij} < d \\ 0 & \text{otherwise} \end{cases}$$

Note that for subregions  $A_i$  and  $A_j$  that are more than apart,  $d$ , then  $w_{ij} = 0$ , i.e. no contribution to the statistic.

Here, we talk about concentration, rather than correlation :

- If nearby  $Y_i$  values are large ('hot spot'), the numerator will be large.
- If nearby  $Y_i$  values are small ('cold spot'), the numerator will be small.

The authors showed that  $E[G(d)] = \frac{w_{\bullet\bullet}}{N(N-1)}$  and the variance is just messy (see original paper if you care).

The authors write of their statistic (and the [paper](#) is a nice read)

$G(d)$  measures overall concentration or lack of concentration of all pairs of [locations] such that  $i$  and  $j$  are within distance  $d$  or each other.

They emphasize that  $G(d)$  and  $I$  measure different aspects (concentration versus correlation) of spatial pattern, and therefore ought to be used simultaneously.

# Statistical Significance Tests for I and c and G

Hypothesis Test:

$H_0$  : No spatial autocorrelation

$H_a$  : There is spatial autocorrelation

Based on the normal frequency distribution with

$$Z = \frac{I - E[I]}{\sqrt{\text{Var}(I)}} \sim N(0,1) \quad Z = \frac{c - E[c]}{\sqrt{\text{Var}(c)}} \sim N(0,1)$$

$$Z = \frac{G(d) - E[G](d)}{\sqrt{\text{Var}(G(d))}} \sim N(0,1)$$

As mentioned above, there are different formula for calculating  $Var[I]$  or  $Var[c]$  or  $Var[G(d)]$ :

- The free sampling or normality method
- The non-free sampling or randomization method
- These formulae are messy and not really informative

In either case, the statistical test is carried out in the same way.

## Monte Carlo Test – (for $I$ and $c$ )

1. Randomly reassign  $Y_i$  across the various subregions  $A_i$ .
2. Calculate  $I$  (or  $c$ )
3. Repeat many (1000) times
4. Calculate the percent of times  $I$  (or  $c$ ) for the original data is more extreme than the randomized  $I$  (or  $c$ ).  
This is the p-value.



These tests can be done in the `spdep` package using `moran.mc` and `geary.mc`.

## A few thoughts (from TGG):



- The calculations of Variance assume that the Y's are normally distributed. Should you worry if they're not?  
*No – the CLT guarantees that our global stats will still be approximately normally distributed. A bigger worry is non-stationarity (i.e., what if mean is NOT constant across the entire region).*
- Would Moran's I and Geary's c ever give opposite results?  
*Probably not in most 'actual data' situations*
- Would I ever indicate significance and c not indicate significance?  
*Probably in borderline situations, but life isn't all about p-values*



## What if overall mean is NOT constant (not stationary)?

If the overall mean is not constant, what about fitting a regression model to remove mean, and then calculate Moran's I on the residuals?

Great idea! This is function `lm.morantest` (see online example code)

Of course, if our model is misspecified, this could result in spatial autocorrelation among the residuals. (McMillen, "Spatial autocorrelation or model misspecification?" *International Regional Science Review* 26(20) 208-217; available online, Yale SML).



**Spatial Autocorrelation in R.** The `spdep` package is the main package for spatial autocorrelation for subregions. Lots of cool functions; however, I find help files rather esoteric and hard to understand. Here are some important functions :

`moran.test()`      Calculated Moran's I and parametric test for significant departures from spatial independence.

`moran.mc()`      Calculated Moran's I and non-parametric (Monte Carlo) test for significant departures from spatial independence.

`poly2nb()`      Calculate which subregions are neighbors.

`coordinates()`      Gets location of subregion centroids.

`knearneigh()`      Calculate k nearest neighbors for each subregion.

`knn2nb()`      Calculate k nearest neighbors for each subregion.

<code>dnearest()</code>	Calculates number of neighbors within fixed radius of the subregion centroid.
<code>nb2listw()</code>	Calculates actual weights based on input list of neighbors calculated using some scheme mentioned above.
<code>geary.test</code>	Calculate Geary's c and perform parametric test of significance.
<code>geary.mc</code>	Calculate Geary's c and perform non-parametric (Monte Carlo) test of significance.
<code>globalG.test</code>	Calculate Getis-Ord G and perform parametric test of significance.
<code>lm.moran.test</code>	Calculates Moran's I for residuals of a linear regression model (i.e. look for spatial autocorrelation after taking out non-spatial effects)

I've put up examples of how the weights are calculated, and there is detailed analysis of county by county leukemia data from New York State (W&G, page 98, and code from various other authors). Also examples of Crime Rates from neighborhoods in Columbus, OH (regression on residuals).

[http://reuningscherer.net/fes781/rscripts/global\\_indices.r.txt](http://reuningscherer.net/fes781/rscripts/global_indices.r.txt)

## Correlegram – (for $I$ and $c$ and $G$ and, well, correlation)

Think back to weight/neighbor definition iii)

$$\text{iii) } w_{ij} = \begin{cases} 1, & \text{if } d_{ij} < \delta \text{ for some prescribed distance } \delta \\ 0, & \text{otherwise} \end{cases} ;$$

In this case, we can do the following :

- Compute a particular global spatial auto-correlation statistics for various distances  $\delta$ .
- Plot the results on a graph (stat on vertical axis,  $\delta$  on horizontal axis)
- Create confidence intervals for the statistic for each distance

- Look for a distance beyond which spatial auto-correlation seems to have tapered off : This is often called the **patch size**

The relevant function in the `spdep` package is `sp.correlogram`. It is mind-bendingly frustrating to get this function to work (not the best help file ever created).

However, I have an example for the NY census tract data.

[http://reuningscherer.net/fes781/rscripts/global\\_indices.r.txt](http://reuningscherer.net/fes781/rscripts/global_indices.r.txt)

## Bivariate Moran's $I$

We mentioned earlier that our spatial autocorrelation statistics are really univariate statistics. However, there's no reason we couldn't define a bi-variate spatial version of Moran's  $I$ .

Suppose we have two measured variates,  $Y$  and  $Z$

$$I_{YZ} = \frac{\sum_{i=1}^N \sum_{j=1}^N w_{ij} (Y_i - \bar{Y})(Z_j - \bar{Z})}{w_{\bullet\bullet} SD(Y)SD(Z)}$$

You can think of this as a measure of how two variables are correlated in neighboring subregions.

$$E[I_{YZ}] = -\frac{\rho_{YZ}}{N-1} \text{ if there is no spatial autocorrelation}$$

where  $\rho_{YZ}$  is the correlation of  $Y$  and  $Z$  over the entire region. The variance of  $I_{YZ}$  is messy and described in (Czaplewski, R.L et al, 1993, USDA Forest Service Research Paper RM-309).

The `EcoGenetics` package in R will apparently compute bivariate Moran's I. I haven't tried this yet.



# Local Measures of Spatial Auto-Correlation

Global indices of spatial association may suggest the presence of clustering in the population but fail to indicate where clusters are located.

“Recognition of the potential limitations of global measures has led to the development of local measures of spatial autocorrelation” (*Boots, 2002, Ecospice nce*)

They are also called **LISAs** for **Local Indicators of Spatial Association** (*Anselin (1995, Geographic Analysis, 27(2) 93-115)* ).



## Local Moran's $I$

$$I_i = \frac{(Y_i - \bar{Y}) \sum_{j=1}^N w_{ij} (Y_j - \bar{Y})}{\text{Var}(Y)}$$

$$E[I_i] = -\frac{w_{i\bullet}}{N-1}, \text{ where } w_{i\bullet} = \sum_{j=1}^N w_{ij}$$

## Local Geary's $c$

$$c_i = \frac{\sum_{j=1}^N w_{ij} (Y_i - Y_j)^2}{\text{Var}(Y)} \quad E[c_i] = \frac{2Nw_{i\bullet}}{N-1}$$

## Local Getis-Ord $G$

$$G_i(d) = \frac{\sum_{j=1}^N w_{ij}(d) Y_i Y_j}{\sum_{j=1}^N Y_j}$$

$$E[c_i] = \frac{w_{i\bullet}}{N-1}$$

For each of these, the variance is messy and available in Anselin, 1995.

## Characteristics of a LISA:

- Provides a measure of spatial correlation around each sub-region (*census block/county/etc.*)
- Can be used to identify hot and cold spots
- The sum of all LISAs provides a statistic that is proportional to the corresponding global measure of spatial association.
- Are a good exploratory tool to detect subregions within which spatial association is markedly strong, even though the global measure may not indicate spatial autocorrelation.
- Drawback - no theoretical basis to know how these measures are distributed so harder to do hypothesis tests (still can get p-values based on randomization).

The use of LISAs as Exploratory Spatial Data Analysis (ESDA) tools seem to be on much firmer ground than their use in a confirmatory or test setting.

The relevant function in the `spdep` package is `localmoran`. Again, the help files are frustrating.

I have an example for the NY census tract data.

[http://reuningscherer.net/fes781/rscripts/global\\_indices.r.txt](http://reuningscherer.net/fes781/rscripts/global_indices.r.txt)

There is also a `localG` function in `spdep`

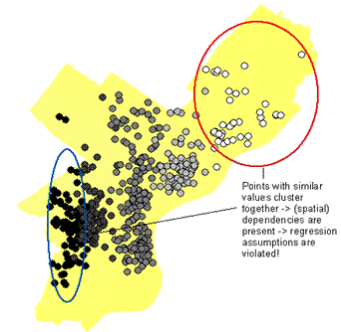
# Spatial Regression

**Usual multiple regression model in scalar notation** : if you have  $p$  predictors, the **linear model** for the value of  $Y_i$  for a single observation  $i$  is given by

$$Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \varepsilon_i , \quad i = 1, 2, \dots, n$$

$$\varepsilon_i \sim N(0, \sigma^2)$$

*Note – other more complex models are possible (non-linear): for now, we'll stick to the linear case*



**NOW** : if  $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$  is a  $p \times 1$  column vector and

$\mathbf{x}'_i = [1 \quad X_{i1} \quad X_{i2} \quad \dots \quad X_{ip}]$  is a  $1 \times p$  row vector then

$$Y_i = \mathbf{x}'_i \boldsymbol{\beta} + \varepsilon_i \quad (\text{very compact})$$



**NEXT** – stack up all these equations on top of each other, one for each observation in the dataset :

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ 1 & x_{31} & x_{32} & \dots & x_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

**THAT IS :**

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$



**Usually we assume homoscedasticity** (*constant variance!*)



- For the  $i$ th observation,  $V[Y_i/\mathbf{x}_i] = \sigma^2$  (same for every observation – this is variance of errors)
- Covariance matrix of all  $n$  observations :

$$V[\mathbf{Y} | \mathbf{X}] = \Sigma = \sigma^2 \mathbf{I} \quad (\text{where } \mathbf{I} \text{ is the identity matrix})$$

**IDEA** : heteroscedasticity, spatial correlation, temporal correlation, are all modeled by modifying  $\Sigma$ , the covariance of  $\mathbf{Y}$  given  $\mathbf{X}$ , i.e. the covariance of the errors!

If we've specified the model 'correctly' (maybe better to say 'adequately'), then

$$\text{Mean of } \mathbf{Y} = \boldsymbol{\mu} = E[\mathbf{Y} | \mathbf{X}] = \mathbf{X}\boldsymbol{\beta}$$

In this case, the mean of the errors (the residuals) should be zero –

$$E|\boldsymbol{\varepsilon}| = \begin{bmatrix} E|\varepsilon_1| \\ E|\varepsilon_2| \\ E|\varepsilon_3| \\ \vdots \\ E|\varepsilon_n| \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

*Note: this is one main reasons we look at residual plots – to see if we've missed any 'mean' trends*

## Back to $\Sigma$

In the usual homoscedastic, independent error case,  $\Sigma = \sigma^2 \mathbf{I}$

Let's think about the more general case :

By definition, the variance of  $\varepsilon_i$  is  $Var(\varepsilon_i) = E[\varepsilon_i^2] - (E[\varepsilon_i])^2$

But, we assume that  $E[\varepsilon_i] = 0$

SO : 
$$Var(\varepsilon_i) = E[\varepsilon_i^2]$$

Similarly the covariance between the pair  $\varepsilon_i$  and  $\varepsilon_j$  is, by definition,  $E[\varepsilon_i \varepsilon_j] - E[\varepsilon_i]E[\varepsilon_j]$ .

This simplifies to just  $E[\varepsilon_i \varepsilon_j]$  (because  $E[\varepsilon_i] = E[\varepsilon_j] = 0$ )

Therefore

$$V(\varepsilon) = \Sigma = \begin{pmatrix} E[\varepsilon_1^2] & E[\varepsilon_1 \varepsilon_2] & \cdots & E[\varepsilon_1 \varepsilon_n] \\ E[\varepsilon_2 \varepsilon_1] & E[\varepsilon_2^2] & \cdots & E[\varepsilon_2 \varepsilon_n] \\ & \vdots & & \vdots \\ E[\varepsilon_n \varepsilon_1] & E[\varepsilon_n \varepsilon_2] & \cdots & E[\varepsilon_n^2] \end{pmatrix}$$

NEXT – typical notation convention is that  $E[\varepsilon_i^2] = \sigma_i^2$  and  $E[\varepsilon_i \varepsilon_j] = \sigma_{ij}$

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n} \\ & \vdots & & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{pmatrix}$$

**Heteroskedasticity alone** :  $\sigma_i^2 \neq \sigma_j^2$  for at least some  $i$  and  $j$ ; however,  $\sigma_{ij} = 0$

**Spatial (other) correlation alone** :  $\sigma_{ij} \neq 0$  for at least some pairs of observations, but  $\sigma_i^2 = \sigma_j^2 = \sigma^2$  (constant variance around mean =  $\mathbf{X}\boldsymbol{\beta}$ )

**Spatial (other) correlation AND heteroskedasticity** :  $\sigma_{ij} \neq 0$  for at least some pairs of observations, **and**  $\sigma_i^2 \neq \sigma_j^2$  for at least some  $i$  and  $j$

**SO : our model**       $\mathbf{Y} = \mu(\mathbf{X}, \boldsymbol{\beta}) + \boldsymbol{\varepsilon} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$

$$\text{Var}(\boldsymbol{\varepsilon}) = \boldsymbol{\Sigma}$$

$\boldsymbol{\Sigma}$  specifies the variation of  $\mathbf{Y}$  around  $\mathbf{X}$

Usually have to put some structure on the elements of  $\boldsymbol{\Sigma}$  to reduce the number of parameters to be estimated (otherwise it's  $(n + 1) * n$  )

**Can also specify the DISTRIBUTION of the errors  $\boldsymbol{\varepsilon}$**

- For binary data, might be a Bernoulli
- For proportions, use a Beta
- For counts, use Poisson or negative binomial
- For continuous data, use **NORMAL!**

$$\mathbf{Y} = \mu(\mathbf{X}, \boldsymbol{\beta}) + \boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \boldsymbol{\Sigma})$$

*Note: the model above indicates that the ERRORS will have a normal distribution, NOT that the data will have a normal distribution. This is why we make quantile plots of the errors, NOT of the data*

*The errors are what is left after subtracting out the mean. This is the **CONDITIONAL** distribution of the Y's*

.....

**Generalized Least Squares** (not to be confused with Generalized Linear Models).

The **ordinary least-squares** estimate of our coefficients  $\boldsymbol{\beta}$  is given by (<https://sites.harvard.edu/fs/docs/icb.topic515975.files/OLSDerivation.pdf>)

$$\tilde{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{Y}$$

The **generalized least-squares** estimate of our coefficients  $\boldsymbol{\beta}$  is given by ([https://en.wikipedia.org/wiki/Generalized\\_least\\_squares](https://en.wikipedia.org/wiki/Generalized_least_squares)) (*think Mahalanobis Distance*)

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\boldsymbol{\Sigma}^{-1}\mathbf{X})^{-1} \mathbf{X}'\boldsymbol{\Sigma}^{-1}\mathbf{Y}$$

In usual regression case, where  $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$  (in this case,  $\boldsymbol{\Sigma}$  is said to have a *spherical* structure), this reduces to the OLS case above.

.....

*At this point, we'll work through some examples to evaluate models of increasing error complexity, and see how this works in R . . .*

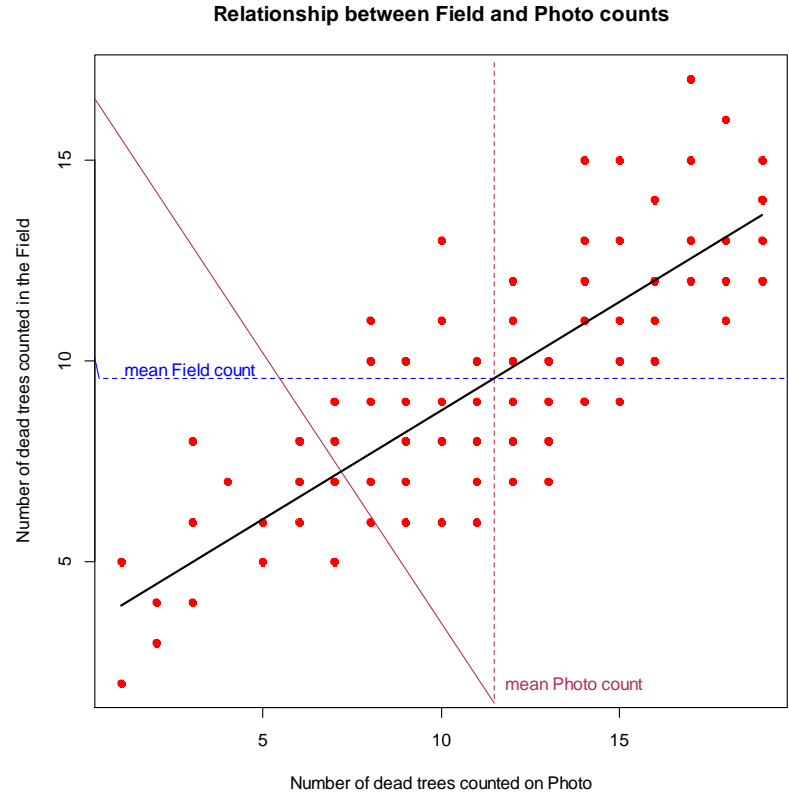


# Case 1 – homoscedasticity and uncorrelated errors

**Example** : Comparison of number of dead trees between aerial photos (X) and field counts (Y) (page 175 of Barrett and Nutt (1979))

Rcode :

<http://www.reuningscherer.net/fes781/Rscripts/Regressionexamples.txt>



It seems reasonable for these data to use the model

$$\mu(\mathbf{X}, \boldsymbol{\beta}) = \beta_0 + \beta_1 X_1$$

In this case, plots were all from different owners, so reasonable to assume errors are pairwise uncorrelated :  $\sigma_{ij} = 0$

Graph suggests that error variance is constant. SO :

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2 & 0 & \dots & 0 \\ 0 & \sigma^2 & \dots & 0 \\ & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma^2 \end{pmatrix} = \sigma^2 \mathbf{I}$$

(*spherical error structure*). Residual standard error is  $\sqrt{\hat{\sigma}^2}$

## GLS Results :

Generalized least squares fit by REML

Model: Field ~ Photo

Data: NULL

	AIC	BIC	logLik
	334.3131	341.3832	-164.1565

Coefficients:

	Value	Std. Error	t-value	p-value
(Intercept)	3.385318	0.5295595	6.392705	0
Photo	0.538317	0.0425284	12.657817	0

Residual standard error: 1.839052

Degrees of freedom: 80 total; 78 residual

## LM results (identical)

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	3.38532	0.52956	6.393	1.1e-08	***
Photo	0.53832	0.04253	12.658	< 2e-16	***

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.839 on 78 degrees of freedom

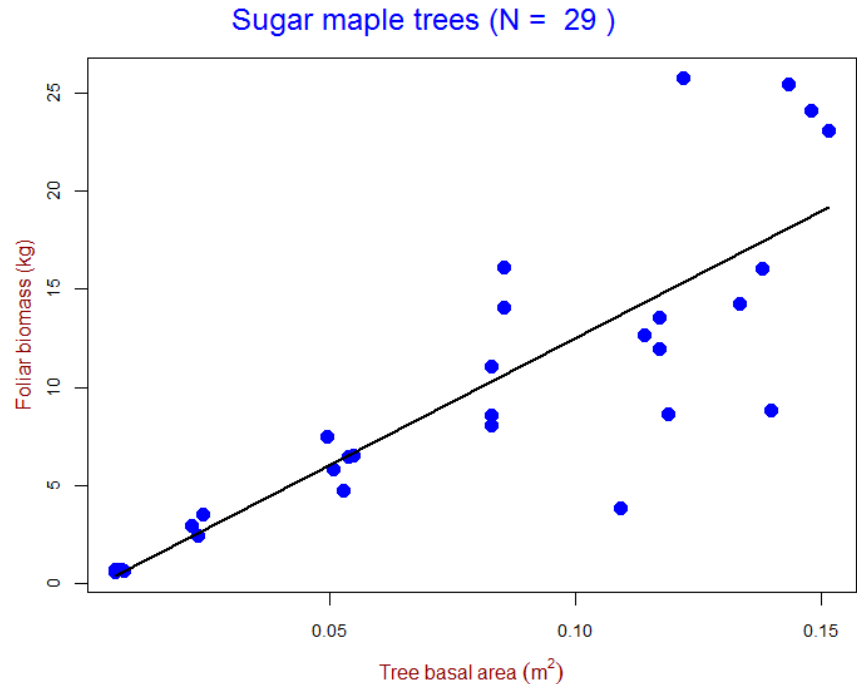
Multiple R-squared: 0.6726, Adjusted R-squared: 0.6684

F-statistic: 160.2 on 1 and 78 DF, p-value: < 2.2e-16

## Case 2 – heteroskedasticity and uncorrelated errors

**Example :** Tree basal area ( $X$ ) as predictive of Foliar biomass ( $Y$ ) (Cunia & Briggs, 1984).

Heteroskedasticity clearly evident – variance of  $Y$  increases as  $X$  increases



Rcode : <http://www.reuningscherer.net/fes781/Rscripts/Regressionexamples.txt>

**Idea** : make covariance matrix include X

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma^2 X_{i1} & 0 & \dots & 0 \\ 0 & \sigma^2 X_{i1} & \dots & 0 \\ & \vdots & & \vdots \\ 0 & 0 & \dots & \sigma^2 X_{in} \end{pmatrix} = \sigma^2 \mathbf{diag}(X_i)$$

That is, variance for each Y value is a scaled function of the corresponding X value.

*Note: output lists residual standard error; however, this is NOT correct (i.e. this doesn't provide an estimate of  $\hat{\sigma}$ )*

Here are five models we'll consider in class (or see R code)

**Model 1** – ignore heteroskedasticity (for comparison only) :

$$\Sigma = \sigma^2 \mathbf{I}$$

**Model 2** – fit error variance as a linear function of X :

$$\Sigma = \sigma^2 \mathbf{diag}(X_i)$$

**Model 3** – same as above, but fit with weighted least squares (usual linear model, but with weights = 1/X). SAME results as Model 2

**Model 4** – fit error variance as a power function of X :

$$\Sigma = \sigma^2 \mathbf{diag}(X_i^{2\delta})$$

For each model, slope relating basal area to foliar biomass is almost IDENTICAL. However, AIC is best for model 4.



**Generalized Least Squares in R.** The main package you want is `nlme`, and the main function for generalized least squares is `gls()`. There are two relevant functions/options for specifying the form of  $\Sigma$  are

`varFunc` – this specifies the form of the diagonal entries of  $\Sigma$   
includes `varFixed`, `varPower`, etc. See Pinheiro and Bates, section 5.2 (on CANVAS under Files → Misc Resources `varFunc.pdf`)

`corStruct` – this specifies the form of the diagonal entries of  $\Sigma$

Tim has some nice examples of the both of these in the online R-code `varFunc_corStruct_examples.r` (and `varFunc_corStruct_examples.r.results.pdf`)

## Case 3 – heteroskedasticity and nonlinearity and uncorrelated errors

**Example :** *Eucalyptus Leaf Area*). Measurements were taken on 744 leaves (area, length and width) and a model was fit to predict leaf area ( $Y$ ) based on the rectangular region length\*width ( $X$ ). In addition, splines were used to account for the slight curvilinearity in the trend. Heteroskedasticity is also evident

<http://www.reuningscherer.net/fes781/Rscripts/Regressionexamples.txt>



## Case 3 – homoscedasticity and longitudinal autocorrelation

**Example** : *LiDAR biomass (no data, just example)* (Bollandsas et al (including Tim Gregoire), 2013, Detection of biomass change in a Norwegian mountain forest area using small footprint airborne laser scanner data. *Statistical Methods and Applications*, 22(2) 113-129)

*LiDAR was flown on two occasions over a 10 km<sup>2</sup> region of southeastern Norway to estimate above ground forest biomass for the region. The same locations were measured 3 years apart. Goal – predict change in biomass on the 52 measured plots based on LiDAR metric covariates.*

**Notation** : 52 plots x 2 measurements

$$Y_{it}, i = 1, 2, \dots, 52 \text{ and } t = 1, 2$$

## Group observations by plot, then by occasion

Assumptions – this is called **compound symmetry**

- Plots are uncorrelated
- Covariance between the paired measurements on each plot was the same for all plots
- Variance was the same for all plots

$$\Sigma = \begin{pmatrix} \sigma^2 & \sigma_{12} & 0 & 0 & \dots & 0 & 0 \\ \sigma_{21} & \sigma^2 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \sigma^2 & \sigma_{12} & \dots & 0 & 0 \\ 0 & 0 & \sigma_{21} & \sigma^2 & \dots & 0 & 0 \\ \vdots & & \vdots & & \ddots & \vdots & \\ 0 & 0 & 0 & 0 & \dots & \sigma^2 & \sigma_{12} \\ 0 & 0 & 0 & 0 & \dots & \sigma_{21} & \sigma^2 \end{pmatrix}$$

Here is the R-code that was used :

```
gls(Y ~ X1 + X2 , method="REML",  
    correlation = corCompSymm(value = 0.5,  
form = ~1 | subject), na.action=na.omit)
```

## Case 4 – homoscedasticity and longitudinal autocorrelation

**Example** : *Heights of boys from Oxford, measured at various ages.*

Rcode :

<http://www.reuningscherer.net/fes781/Rscripts/Regressionexamples.txt>

*In this model, we specify a continuous autoregressive error correlation structure with the corCAR1 correlation function.*

*Let  $\phi$  be the correlation between observations that are a unit (lag 1) distance apart (i.e. one year apart).*

*Then : observations that are a distance  $d$  apart, the CAR autocorrelation is  $\rho(d) = \phi^d$*

*As long as  $0 < \phi < 1$ ,  $\phi^d$  will decrease smoothly and continuously as the distance  $d$  increases.*

# Spatial Regression (cont.)

**Main Text : Model Based Geo-Statistics (MBG)** (*Diggle and Ribeiro*) [Yale online resource](#)

We'll cover concepts in chapters 1, 2, 3, 5 (much of this we've already seen in kriging)

Diggle and Ribeiro are developers of `geoR` package, which is what we'll discuss today (as well as `gls` in `nlme`).

## Resources

- 1) Tim has created programs to reproduce all of the figures in chapters 1, 2, 3, and 5. These are on CANVAS under Files → R Scripts.
- 2) We'll look at two different examples today to get a sense of how spatial regression works in geoR (and nlme).

Main Code for Today :

<http://www.reuningscherer.net/fes781/Rscripts/SpatialRegression.txt>

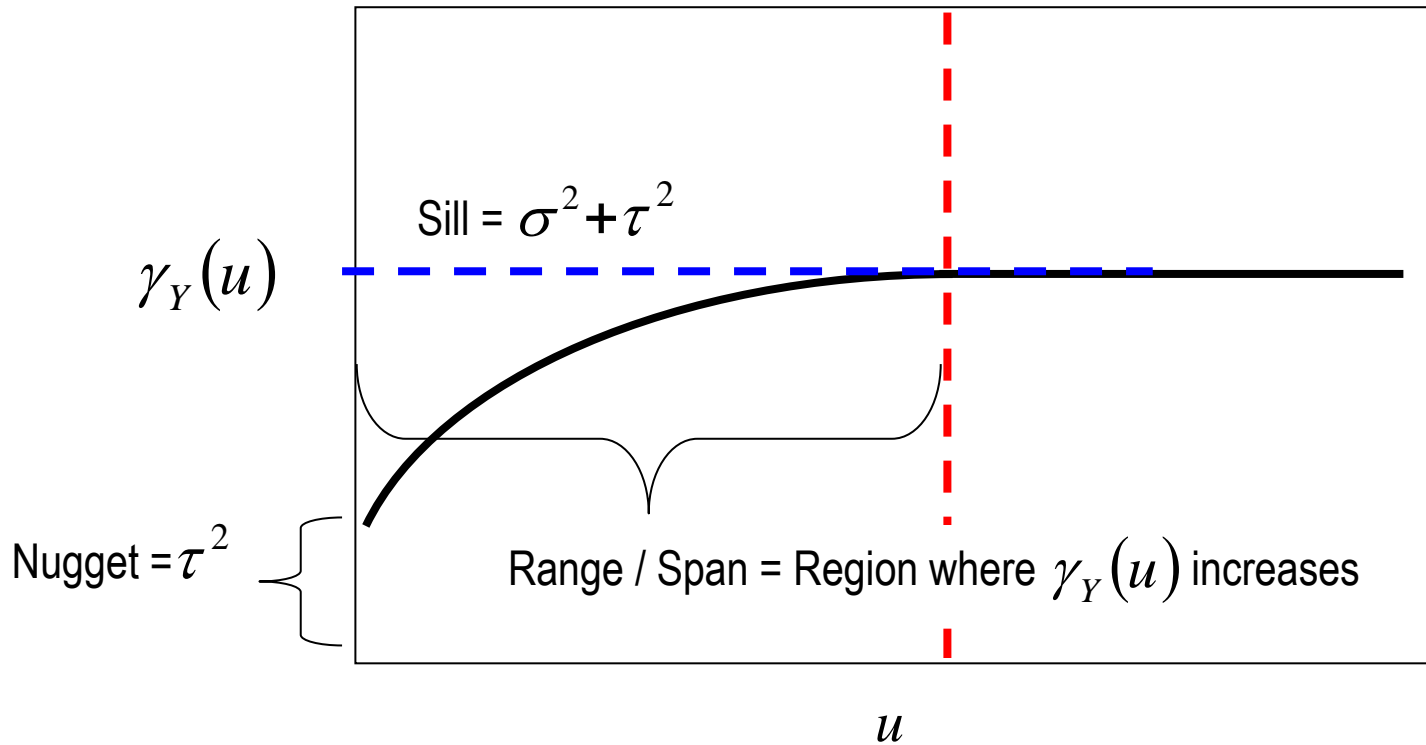
## Let's review the MBG spatial regression model

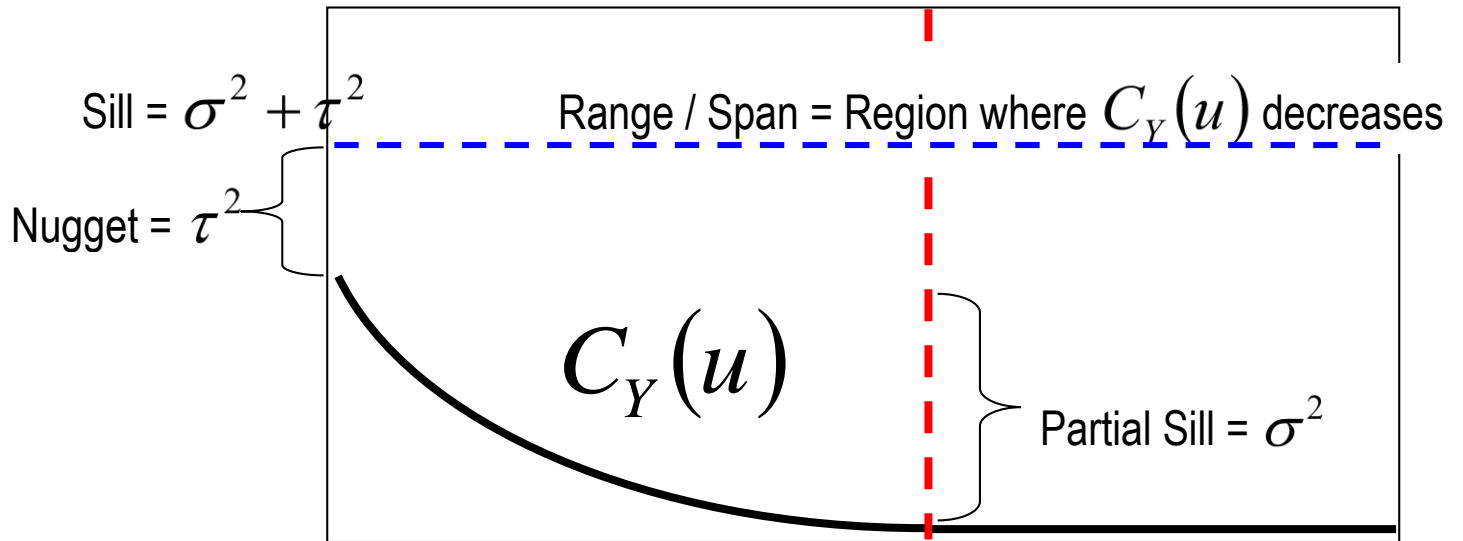
$$Y(\mathbf{x}_i) = \mu(d : \mathbf{x}_i) + S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2) + Z_i$$

- $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  : our observed spatial locations
- $\mu(d : \mathbf{x}_i)$  - the mean which depends on  $\mathbf{x}_i$  (location) and  $d$  (covariates beside location). Could be constant, sloped, quadratic, dependent on covariates like slope/aspect, etc.
- $S(\cdot)$  - the signal **AFTER** we remove the mean trend . The realization of a stochastic process (some continuous quantity). This contains the information on spatial autocorrelation (if any). **Essentially this is our matrix  $\Sigma$** 
  - $\sigma^2$  - the variation of  $S(\cdot)$



- $\rho(u)$  = the **correlation** of observations in  $S(\cdot)$  that are a distance  $u$  apart. If we assume stationarity (and perhaps also isotropy), we can express this through the variogram  $\gamma_Y(u)$  (or covariogram)  $C_Y(u) = \sigma^2 + \tau^2 - \gamma_Y(u)$
- $Z_i$  - random measurement error. We assume  $Z_i \sim N(0, \tau^2)$  **and** the  $Z_i$  are **uncorrelated** with each other **and** with  $S(\cdot)$ .





## Next : a bit more about some common covariogram /variogram models – the MATERN function.

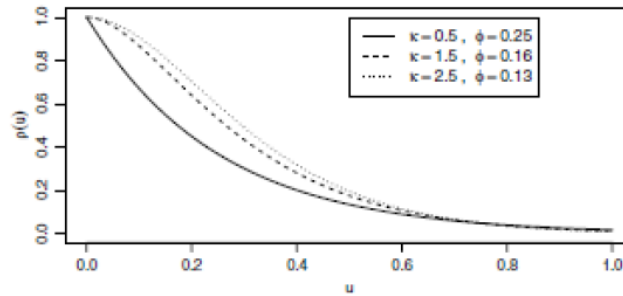
“The Matérn family of correlation functions ... is a two-parameter family,

$$\rho(u) = \{2^{\kappa-1}\Gamma(\kappa)\}^{-1}(u/\phi)^{\kappa}K_{\kappa}(u/\phi)$$

in which  $K_{\kappa}(\bullet)$  denotes a modified Bessel function of order  $\kappa$ ,

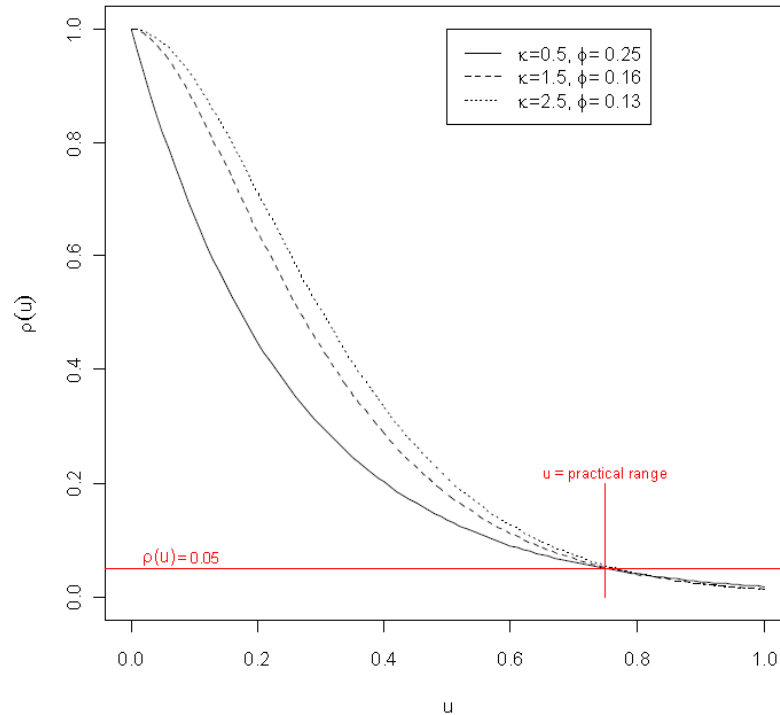
$\phi > 0$  is a scale parameter with the same dimensions as  $u$ ,  
and

$\kappa > 0$ , called the *order*, is a shape parameter  
which determines the analytic smoothness of the underlying  
process  $S(x)$ .”



If  $\kappa = 0.5$ , we get the exponential variogram

The *practical range* is commonly defined as that distance  $u$  at which  $\rho(u) = 0.05$ , as shown in red in the modification of Figure 3.2, below.



**MBG** favors using the Matern family because of its flexibility. They recommend fixing  $\kappa$ , or trying a few fixed  $\kappa$  (.5., 1, 1.5, for example). As we discussed before, this doesn't really have much effect on the fit of the model.

*Note – geoR has a cov.spatial() function with cov.model argument that allows for*

- *matern*
- *exponential*
- *gaussian*
- *spherical*
- *circular*
- *cubic*
- *wave*
- *power*
- *powered.exponential*
- *cauchy*
- *gencauchy*
- *gneiting*
- *gneiting.matern*
- *pure.nugget*

**SO : here is how we incorporate spatial autocorrelation into our model!**

**Idea :** make covariance matrix include the covariogram

$$\Sigma = \begin{pmatrix} \sigma^2 & C(\|x_1 - x_2\|) & \cdots & C(\|x_1 - x_n\|) \\ C(\|x_2 - x_1\|) & \sigma^2 & \cdots & C(\|x_2 - x_n\|) \\ \vdots & \vdots & \ddots & \vdots \\ C(\|x_n - x_1\|) & C(\|x_n - x_2\|) & \cdots & \sigma^2 \end{pmatrix}$$

## **Example 1 : Elevation Data**

<http://www.reuningscherer.net/fes781/Rscripts/SpatialRegression.txt>

Note about trend surface models:

first order  $\equiv$  linear trend:  $\mu(x; \beta) = \beta_0 + \beta_1 x + \beta_2 y;$

second order  $\equiv$  quadratic trend:

$$\mu(x; \beta) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 y^2 + \beta_5 xy.$$



## Monte Carlo Variograms – a visual ‘test’ of spatial autocorrelation (or where spatial auto-correlation ends)

- 1) Do usual empirical variogram using lags
- 2) Next – make empirical variogram using lags, BUT randomly assign Y values to each location
- 3) Repeat 2) 99 (or other) times to create boundary (use convex hull of randomized variograms)
- 4) Plot boundaries to see where spatial auto-correlation ends

Function in `geoR` package is `variog.mc.env ()`

## Example 2 : Swiss Rainfall Data

<http://www.reuningscherer.net/fes781/Rscripts/SpatialRegression.txt>

Note about trend surface models:

first order  $\equiv$  linear trend:  $\mu(x; \beta) = \beta_0 + \beta_1 x + \beta_2 y;$

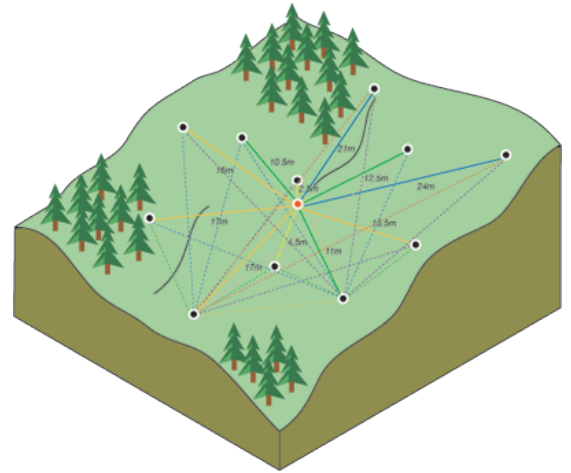
second order  $\equiv$  quadratic trend:

$$\mu(x; \beta) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 y^2 + \beta_5 xy.$$

# Spatial Prediction

## Resources :

- *Diggle and Ribeiro, Chapter 3, 5, 6 – our main source.*
- *Bailey and Gatrell, Interactive Spatial Data Analysis, Part C. Easy to read.*
- *Waller and Gotway, Chapter 8. Online book!*
- *Srivastava and Isaaks, multiple chapters. This is a good book for this topic!*
- *Read Kate Beard online notes as well (see link previously)*
  - *ArcMap Site : [http://desktop.arcgis.com/en/arcmap/10.3/tools/3d-analyst-toolbox/how-kriging-works.htm#ESRI\\_SECTION1\\_7245621C6C2D4B4A8B01E64C88BDF9B6](http://desktop.arcgis.com/en/arcmap/10.3/tools/3d-analyst-toolbox/how-kriging-works.htm#ESRI_SECTION1_7245621C6C2D4B4A8B01E64C88BDF9B6)*



So – where are we . .

# Continuous Data and Stochastic Processes

- We have measurements of a continuous attribute  $Y(\cdot)$  at spatial locations  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ : that is, we have the measurements  $Y(\mathbf{x}_1), Y(\mathbf{x}_2), \dots, Y(\mathbf{x}_N)$ .

**Examples** :  $Y(\mathbf{x}_i)$  might be soil moisture, soil zinc concentration, average basal area, radon levels, rent, income, asthma rates, elevation, etc.

- The locations  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  occur over some domain or region  $R$  that might be a grid, or might be an arbitrary set of points.

We now move toward learning how to **make spatial predictions** for a continuous variable  $Y(\cdot)$  at unmeasured locations in the domain  $R$ .

**In other words, given that measurements of a continuous spatial process at a few locations, how can we use spatial information to make predictions at new locations?**

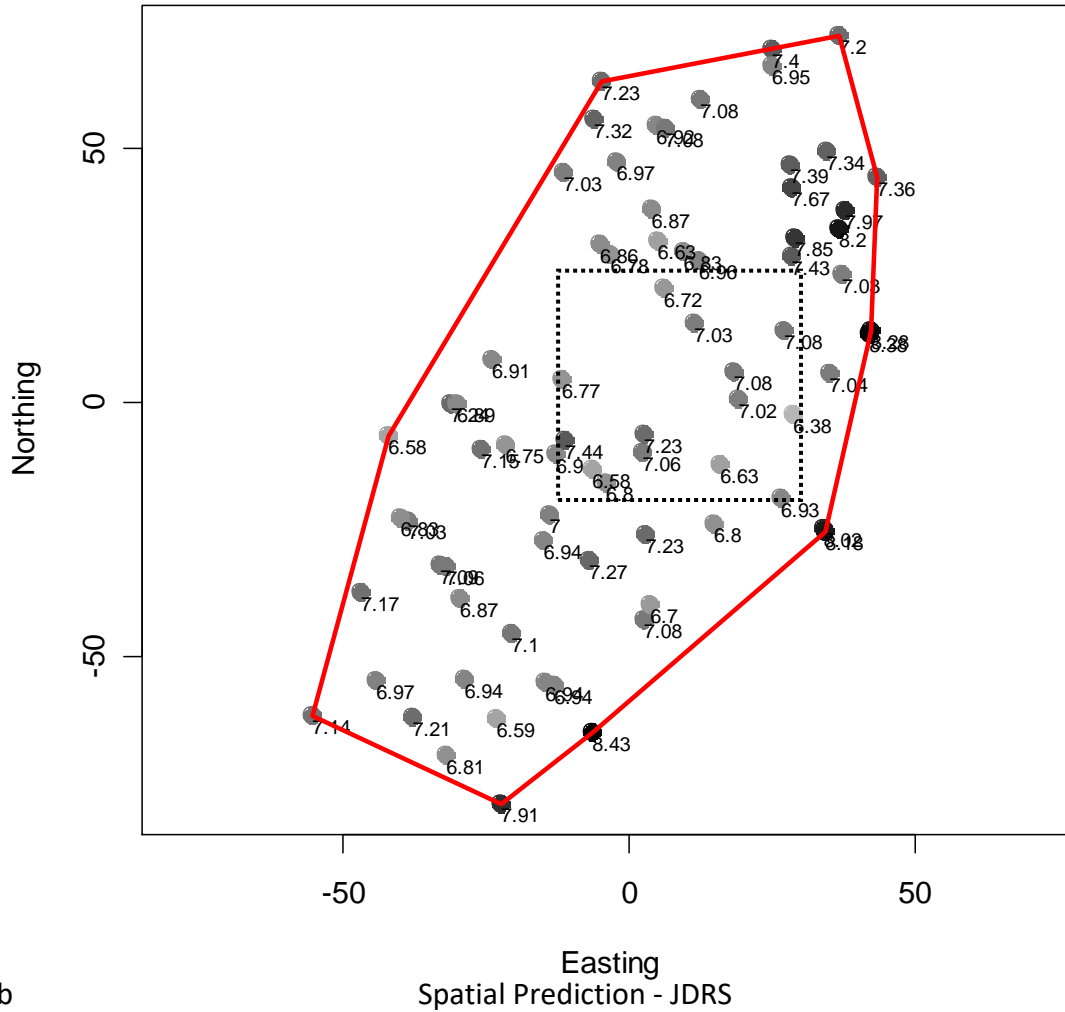
*To help our discussion, here are some motivating examples.*

**Example :** (from Waller and Gotway) : **Smoky Mountain pH Data.** Kaufman et al. (1988) measured water pH (these are the  $Y(\ )$  values) as well as elevation at a number of locations within the Smoky Mountains. A map of pH values is given below : (R code is online as `Variograms.R.txt`)

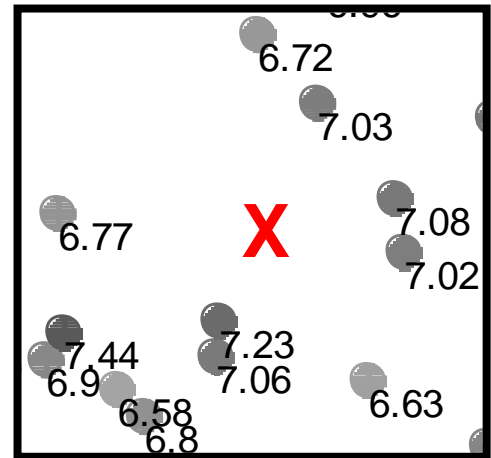


<http://reuningscherer.net/fes781/rscripts/variograms.r.txt>

# Smoky Mountain pH Data

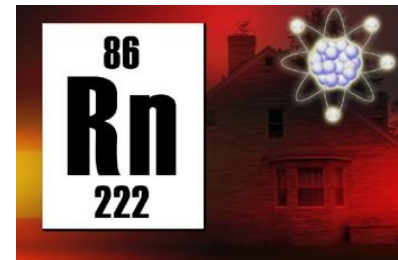


Now – let's look at a subset of the dataset - how should we estimate the pH at a new, unmeasured location denoted by **X**? (pause here for discussion).



Hmm.... how about another

**Example** : (from B&G and other sources) :  
**Radon Data from Lancashire, UK.** Radon levels were measured at 339 (about) houses in the UK in 1989. I've modified this data and added a boundary file. Code (and link to data) is

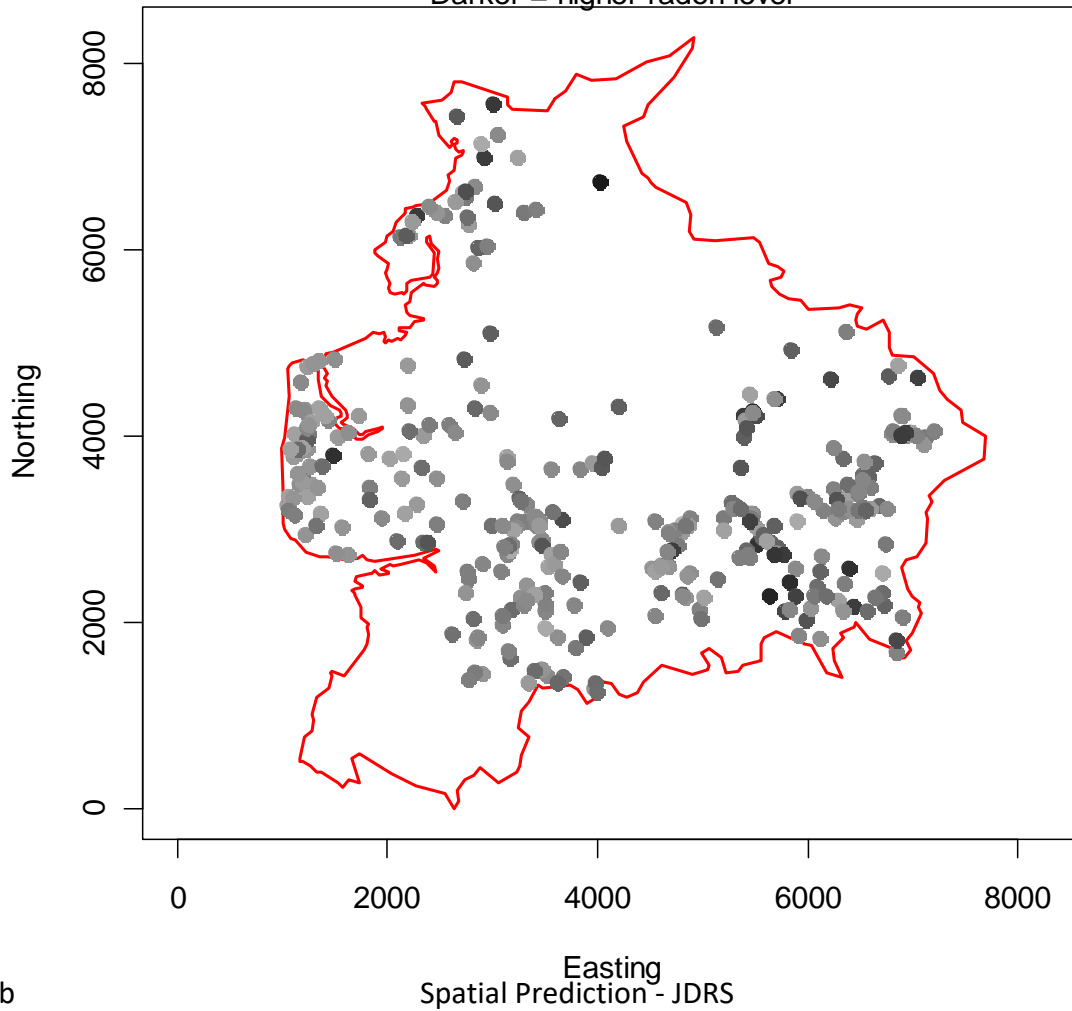


<http://reuningscherer.net/fes781/rscripts/variograms.r.txt>

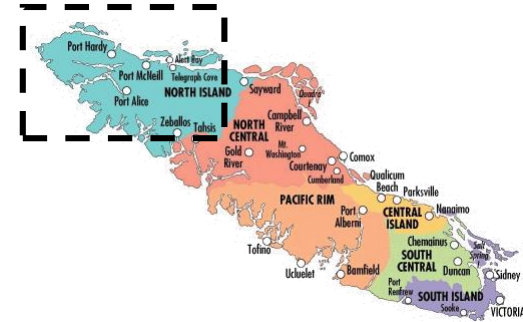


# Lancashire Radon Data

Darker = higher radon level

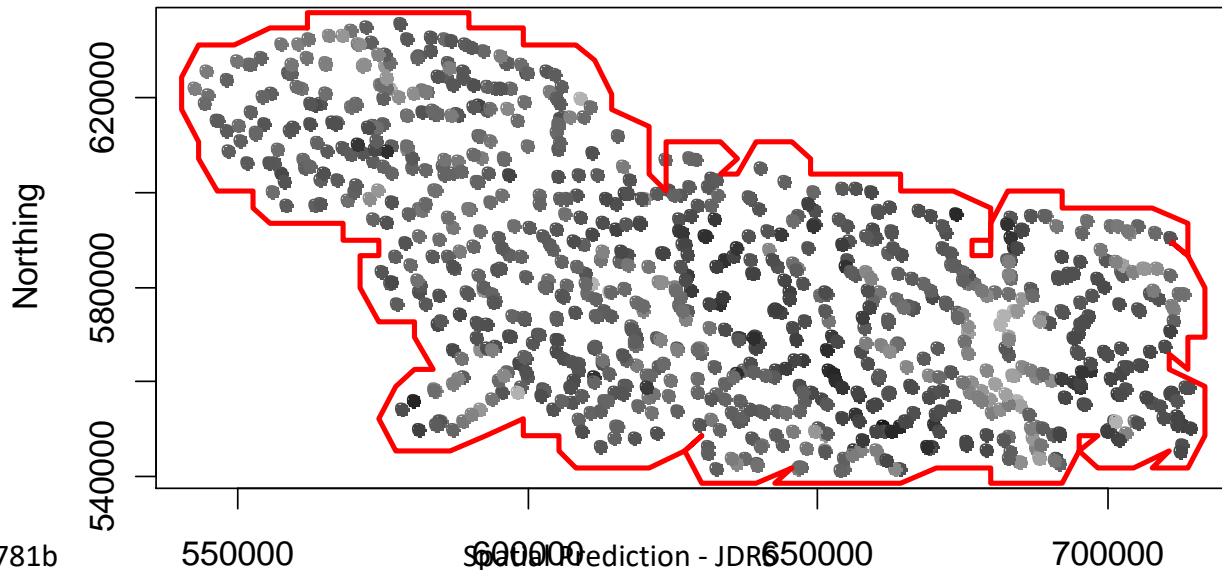


**Example : Metal concentrations on Vancouver Island** (from B&G). This data is from the north-western part of Vancouver Island in BC, Canada. There are 900+ observations of the concentrations of various metals. For now, we'll look at  $\log(\text{nickel})$  values. File to make plot below is here :



[http://reuningscherer.net/fes781/rscripts/Vancouver\\_outline.R.txt](http://reuningscherer.net/fes781/rscripts/Vancouver_outline.R.txt)

### Vancouver Island Nickel Data

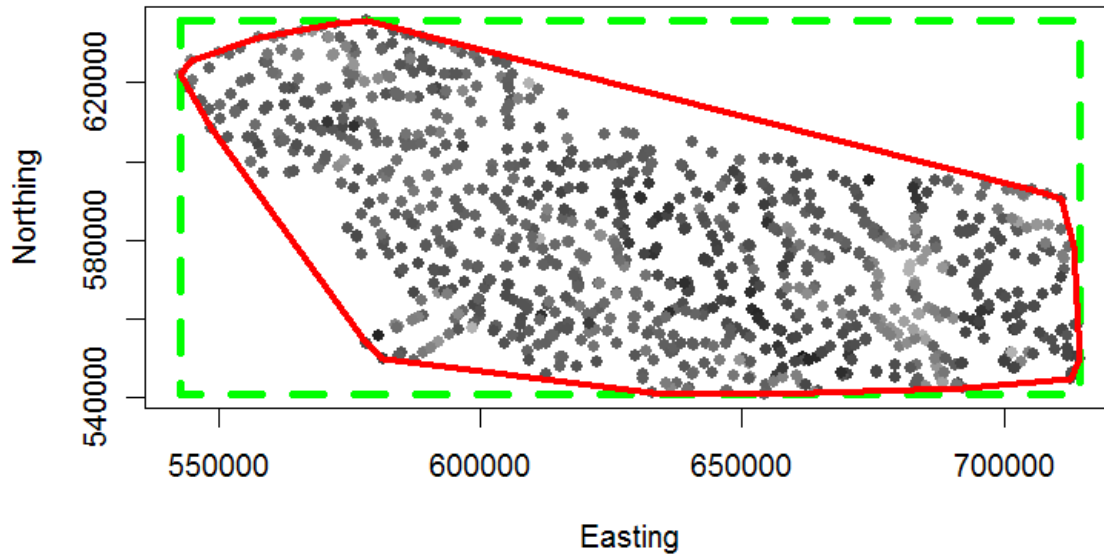


*Um, what's with the goofy red line around the outside? At this point, we pause for a brief detour*



- I have found that spatially continuous data frequently does not come with information on the domain  $R$  over which the data is collected – i.e. it often comes without a boundary!
- Estimation procedures **require** that we have a region/domain  $R$  over which to make predictions.

## Vancouver Island Nickel Data



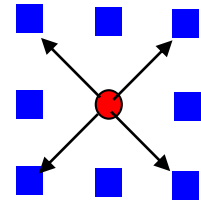
We could just use either a rectangle which includes all observed points or use a **convex hull**, but either process is less than satisfying (especially when we know, for example, that the domain is an island – we don't want to estimate values out in the ocean . . .)

*this pictures uses the R-functions `chull()`, `rect()`, and `polygon()` – see example in `Vancouver_outline.R.txt`.*

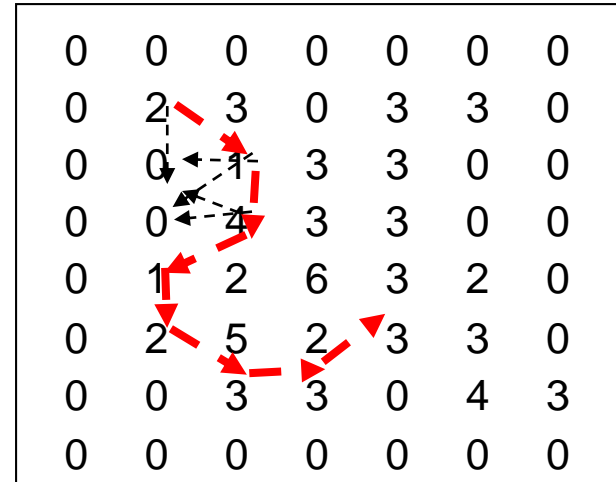
For humans, drawing an outline is a natural process. However, it's quite difficult to get a computer to do the same thing – it follows 'crevices' where we would naturally continue with a smooth boundary.

**SO** : I've written a crude, first-pass R-function called `outline()` that does the following :

- 1) Duplicate the data set 8 times at a specified 'smudge' distance at  $45^\circ$  intervals around the original points (equivalent to duplicating the dataset at all surrounding grid points a a fixed 'smudge' distance). This 'fills in' the dataset and creates a boundary around the original points. Bigger 'smudge' equals bigger boundary.
- 2) Perform quadrat counts on the data with a specified grid size. Fewer grid points makes for a smoother, but less detailed plot.



- 3) Look at the matrix of quadrat counts. The idea is to move around the outside and avoid zero points. The algorithm is as follows : start at one point that is furthest to the left. Move to the first non-zero grid point checking in a counter clockwise fashion but start checking at the point that is **45° counter-clockwise past whichever direction we just came from.**





**Outline in R.** The R script is online at <http://reuningscherer.net/fes781/rscripts/outline.r.txt> Just includes this link in your code to define the function. See example here : [http://reuningscherer.net/fes781/rscripts/Vancouver\\_outline.R.txt](http://reuningscherer.net/fes781/rscripts/Vancouver_outline.R.txt)

*(Pause for R simulation to see how this function works)*

**NEXT** : if we have an outline, wouldn't it be nice to get this into ArcGIS as a polygon so we could use this for kriging?

The R package `shapefiles` does just this! An example for the Vancouver island data is in the file above.

**Final comment** : to get data into ArcGIS, save the file as a .DB4 file. This can be read by ArcGIS.

*Back to our regularly scheduled lecture topic . . .*





We begin our discussion of how to **model spatially continuous data** – that is how to **make predictions** using models that may account for **spatial correlations between measurements**.



**Notation** – *I find MBG's notation a bit confusing at this point (see page 134, Section 6.1), so let me elaborate on our cast of characters.*

**Say hello to our hypothesized model!**

$$Y(\mathbf{x}_i) = \mu(d : \mathbf{x}_i) + S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2) + Z_i$$

**What is this stuff?**

$$Y(\mathbf{x}_i) = \mu(d : \mathbf{x}_i) + S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2) + Z_i$$

- $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  : our observed spatial locations
- $\mu(d : \mathbf{x}_i)$  - the mean which depends on  $\mathbf{x}_i$  (location) and  $d$  (covariates beside location). Could be constant, sloped, quadratic, dependent on covariates like slope/aspect, etc.
- $S(\cdot)$  - the signal **AFTER** we remove the mean trend . The realization of a stochastic process (some continuous quantity). Specifically, we try to observe at locations  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ . **HOWEVER, we don't get to actually observe  $S(\mathbf{x}_1), S(\mathbf{x}_2), \dots, S(\mathbf{x}_N)$ .**
  - $\sigma^2$  - the variation of  $S(\cdot)$
  - $\rho(u)$  = the **correlation** of observations in  $S(\cdot)$  that are a distance  $u$  apart

$$Y(\mathbf{x}_i) = \mu(d : \mathbf{x}_i) + S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2) + Z_i$$

- $Z_i$  - random measurement error. We assume  $Z_i \sim N(0, \tau^2)$  and the  $Z_i$  are **uncorrelated** with each other and with  $S(\cdot)$ .
- $Y(\cdot)$  - the response variable we actually get to observe – the noisy version of  $S(\cdot)$  after we remove the mean

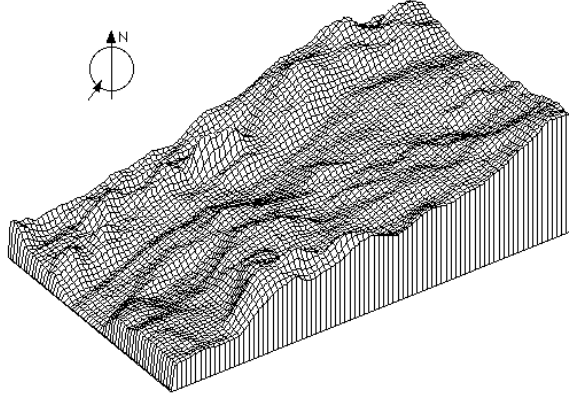
### **Example : Kansas Geological Survey**

(see <http://www.kgs.ku.edu/Tis/surf3/s3trend2.html> )

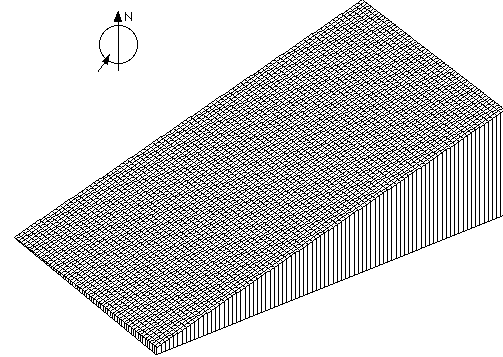
**Surface with trend**  $Y(x_i)$

**Modeled Trend**  $\mu(d : x_i)$

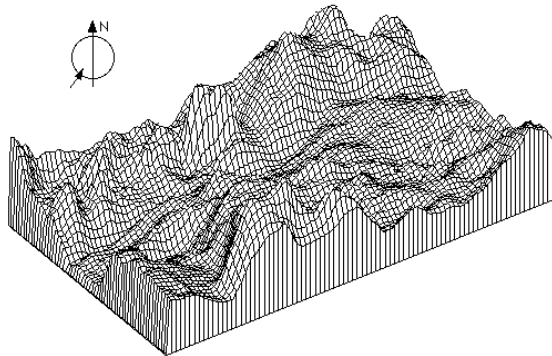
Bouguer gravity, northwest Kansas  
Azimuth -35° (0° is South), elevation 35°



Bouguer gravity, first-order trend, northwest Kansas  
Azimuth -35° (0° is South), elevation 35°



Bouguer gravity, first-order residual, northwest Kansas  
Azimuth -35° (0° is South), elevation 35°



**Stationary Mean Zero Stochastic  
process + noisy errors**

$$S(\mu = 0, x_i, \rho(u), \sigma^2) + Z_i$$

***Our Goal now is to make predictions at some NEW locations. So, more notation . . .***

- $R$  : the domain – the region over which we observed locations  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$
- $\mathbf{x}$  (not  $\mathbf{x}_i$ ) : a new location in the domain  $A$  where we would like to predict  $S(\cdot)$
- $T(\mathbf{x})$  - (that's  $T$  for target). Our estimate of  $S(\cdot)$  at  $\mathbf{x}$  - that is  $T = S(\mathbf{x})$ . I'll note here I'm not clear why MBG defines both  $S(\cdot)$  and  $T(\cdot)$
- $\hat{T}(\mathbf{x})$  - our actual estimate of  $S(\cdot)$  at the point  $\mathbf{x}$ . This is based on the observed values  $Y(\mathbf{x}_1), Y(\mathbf{x}_2), \dots, Y(\mathbf{x}_N)$  (*This is like  $\hat{Y}(\mathbf{x})$  without the mean*)

**SO : GOAL** is to get  $\hat{T}(\mathbf{x})$  (i.e.  $\hat{Y}(\mathbf{x})$ ) -  
our estimate at a **NEW, UNOBSERVED**  
**LOCATION** in  $R$



**Let's look further at our model :**

$$Y(\mathbf{x}_i) = \underbrace{\mu(d : \mathbf{x}_i)}_{\text{First order characteristics}} + \underbrace{S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2)}_{\text{Second order characteristics}} + \underbrace{Z_i}_{\text{Errors}}$$

### First order characteristics

i.e. the **MEAN** of a process – is the mean going up, down across a region in some **large-scale, predictable** manner?

### Second order characteristics

i.e. spatial **CORRELATION / VARIANCE** accounts for smaller scale variability.

Expressed as  $Var[S(\mathbf{x}_i)]$ . If there is no **spatial correlation**, then

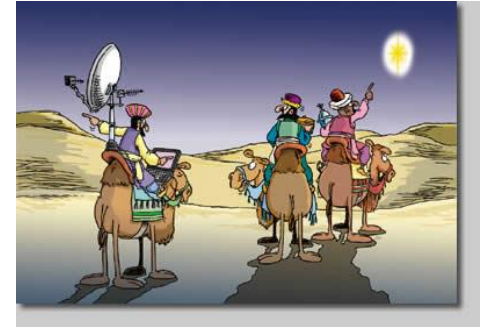
$Var[S(\mathbf{x})] = \mathbf{I}\sigma^2$  (that is, for a single observation,  $Var[S(\mathbf{x}_i)] = \sigma^2$

(variance is constant), and covariance between any two points is  $Cov[S(\mathbf{x}_i), S(\mathbf{x}_j)] = 0$

### Errors

*Note that  $\mu(d : \mathbf{x}_i)$  and  $Var[S(\mathbf{x}_i)]$  are functions of  $\mathbf{x}_i$  - they may change at each location  $\mathbf{x}_i$*

**Where we're headed** : if we make certain assumptions about the structure of  $\mu(d : \mathbf{x}_i)$  and  $Var[S(\mathbf{x}_i)]$  (which may or may not be reasonable), we can achieve



## OPTIMAL SPATIAL PREDICTION

(called **kriging** – optimal means smallest possible errors)

*Of course, this kind of Nirvana is not achieved easily – there are several steps along the way . .*

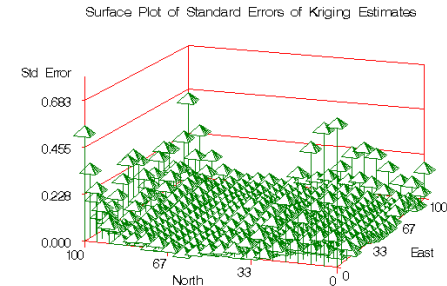


For starters, let's suppose that  $Var[S(\mathbf{x})] = \mathbf{I}\sigma^2$  (that is, there is **NO spatial autocorrelation** – only first order trends)  
**How to estimate  $E[Y(\mathbf{x}_i)] = \mu(d : \mathbf{x}_i)$ ?**

## Spatial Prediction Means only

Here are some methods :

- 1) **Trend Surface Analysis** (or Global Polynomial Interpolation). *Fit a first or second order polynomial using a linear model or using Maximum Likelihood to estimate a surface). We'll see more examples of this later . . .*
- 2) **Inverse Distance Weighting or Interpolation**



# Inverse-Distance Interpolation

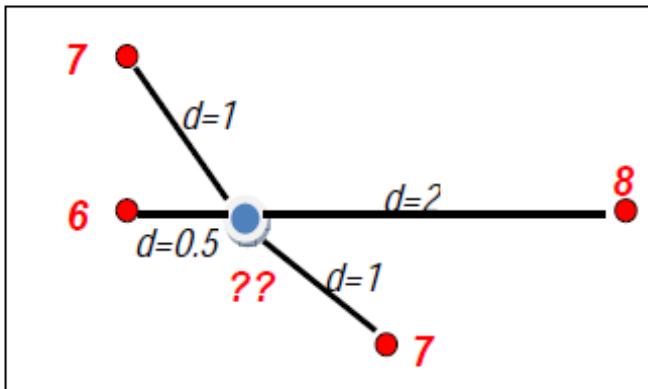
- Suppose we want to make predict the value of  $Y()$  at some new location  $\mathbf{x}_0$ .
- Inverse Distance Interpolation takes a weighted average of the values of  $Y()$  at points around  $\mathbf{x}_0$  such that near values have more influence than far values :

$$T(\mathbf{x}_0) = \frac{\sum_{i=1}^N Y(\mathbf{x}_i) d_{0,i}^{-p}}{\sum_{i=1}^N d_{0,i}^{-p}}$$

$d_{0,i}$  is the distance from  $\mathbf{x}_0$  to  $\mathbf{x}_i$  and  $p$  is the weighting power.

- Usually powers of  $p$  between 1 and 3 are used, and higher powers give further points less weight in the calculation.
  - $p=1$  is a linear decreasing weight (Euclidean distance)
  - $p=2$  is a quadratically decreasing weight (squared Euclidean distance)

**Example** : use IDW to get estimated pH at center point with  $p=1$  :



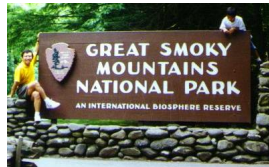
$$\hat{Y}_{???} = \frac{7 * \frac{1}{1} + 7 * \frac{1}{1} + 6 * \frac{1}{0.5} + 8 * \frac{1}{2}}{\frac{1}{1} + \frac{1}{1} + \frac{1}{0.5} + \frac{1}{2}} = 6.667$$

- In practice, most programs use only points within some neighborhood of  $\mathbf{x}_0$  for making predictions (just for ease of computation)

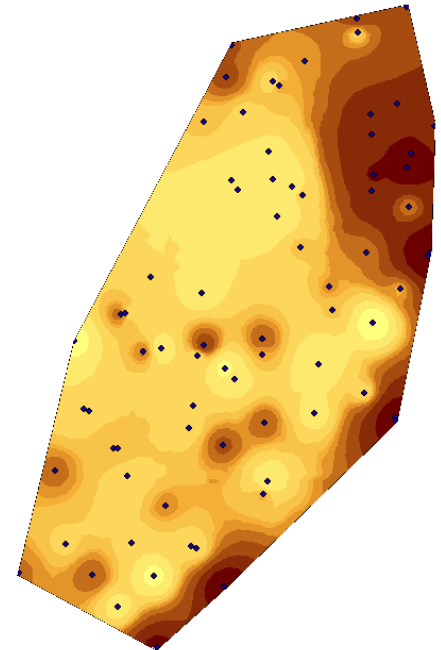
This method is simple, **but** it

- Ignores spatial correlation
- Tends to produce a “Bull’s Eye” pattern around observed points

***Example : Smoky Mountain pH Data.***

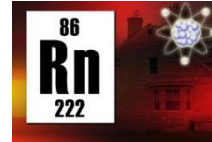


*At right is an example of Inverse Distance Weighting for the Smoky Mountain Data using ArcGIS (use the Geostatistical Analyst - example in class, including Trend Surface Analysis, called ‘Global Polynomial Interpolation’).*

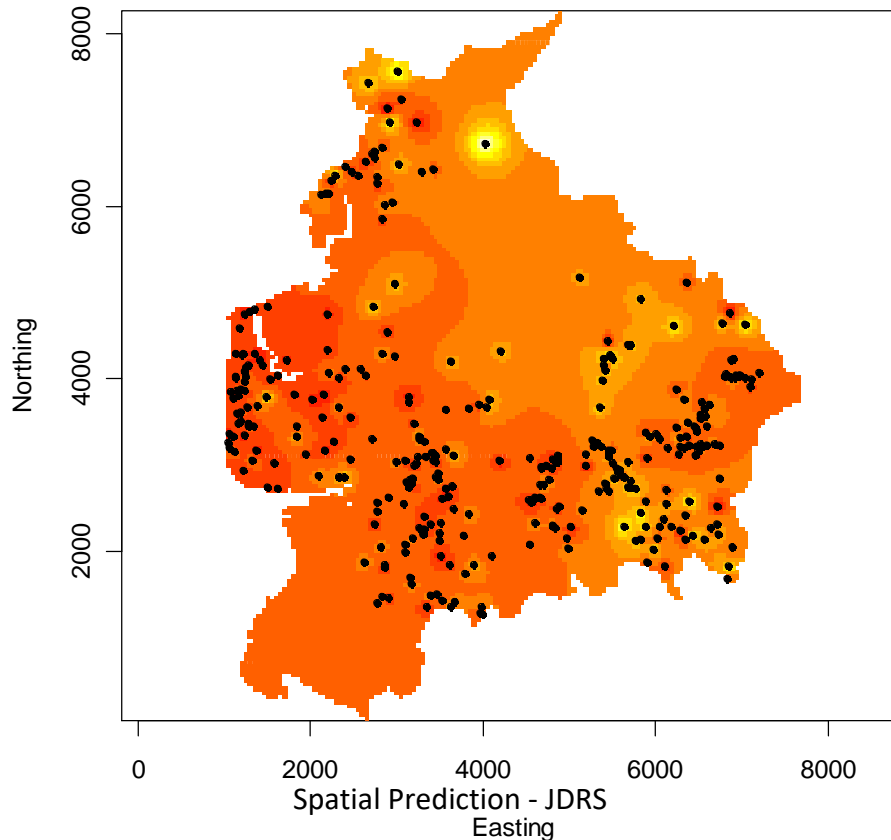




**Example : Lancashire Radon Data.** No evidence of large scale variability in radon levels. - a few high 'spots' – definite tendency toward 'bull's eye' effect.



Inverse Distance Weighting for Radon Data



The next step on our path to Optimal Spatial Prediction Nirvana is to begin to consider data that may contain **spatial autocorrelation** –



That is  $Var[S(\mathbf{x})]$  has some non-zero structure on the diagonal entries. To make this a feasible goal, we assume we have a

# STATIONARY PROCESS



**This means that**

$Var[S(\mathbf{x}_i)] = \sigma^2$  (i.e. for any particular location, the variance is a fixed, constant value, not a function of  $\mathbf{x}_i$ )

*I.e. the variance of the process are **constant** over the entire domain  $A$*

**AND :**

$$Cov[S(\mathbf{x}_i), S(\mathbf{x}_j)] = \sigma^2 \rho(\mathbf{x}_i - \mathbf{x}_j) = \sigma^2 \rho(\mathbf{u})$$

*I.e., the Covariance is dependent only on the difference vector  $\mathbf{u}$  between locations, not on the locations themselves.*



If the Covariance only depends on the length of  $\mathbf{u}$ , (i.e. not the **DIRECTION** of  $\mathbf{u}$ )  $|\mathbf{u}| = u$ , the process is **isotropic**.

*NOTE : when authors say stationary, they usually mean stationary **and** isotropic.*

**NOW** : we want to see if there are patterns of **spatial autocorrelation** among our observations

$S(\mathbf{x}_1), S(\mathbf{x}_2), \dots, S(\mathbf{x}_N)$  (after having accounted for any mean trends – see page 10) : That is, **are observations near each other more similar than observations further apart?**



Let's think about this. A natural way to examine the **similarities** between values of  $S(\ )$  at two locations  $\mathbf{x}_i, \mathbf{x}_j$  is to take the **differences** :  $S(\mathbf{x}_i) - S(\mathbf{x}_j)$ .

This difference is just another **random variable** (i.e. differences of random variables are themselves random variables).

**SO** : What are the mean and variance of  $S(\mathbf{x}_i) - S(\mathbf{x}_j)$ ?

In general, we couldn't say much more about this, since our answer would depend on the locations  $\mathbf{x}_i, \mathbf{x}_j$ .

**HOWEVER** : we're assuming

**STATIONARITY!**

**SO** :



$$Y(\mathbf{x}_i) = \mu(d : \mathbf{x}_i) + S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2) + Z_i$$

$$E(S(\mathbf{x}_i) - S(\mathbf{x}_j)) = 0 - 0 = 0$$

Already took out mean trend so mean of  $S()$  is zero!

$$\text{Var}(S(\mathbf{x}_i) - S(\mathbf{x}_j))$$

$$= \text{Var}(S(\mathbf{x}_i)) + \text{Var}(S(\mathbf{x}_j)) - 2\text{Cov}((S(\mathbf{x}_i), S(\mathbf{x}_j)))$$

$$= 2\sigma^2 - 2\text{Cov}(S(\mathbf{x}_i), S(\mathbf{x}_j))$$

Stationarity means this is only a function of the distance  $u$ , called the **spatial lag**

$$= 2\sigma^2 - 2\sigma^2 \rho(u)$$

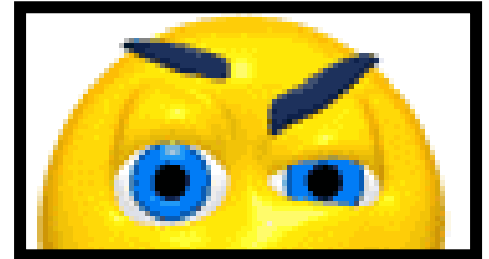
$$= 2[\sigma^2(1 - \rho(u))]$$

$$= 2\gamma(u)$$

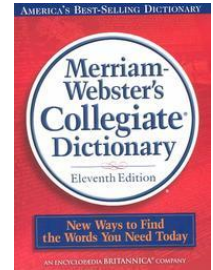
**= THEORETICAL VARIOGRAM!**

$\gamma(u)$  (gamma) is thus half 'o variogram or a **semi-variogram**.  
(however, MBG call a semi-variogram just a variogram, and  
this is what we'll call  $\gamma(u)$  from here on).

**So**, in a sense, the whole stationarity thing  
is to make it possible to evaluate the  
quantities above. Does this make  
you skeptical?



**SO : A Theoretical Variogram is a function which  
describes the nature of spatial dependence between  
locations. Stationarity and Isotropy mean this depends  
only on the distance  $u$  between locations!**



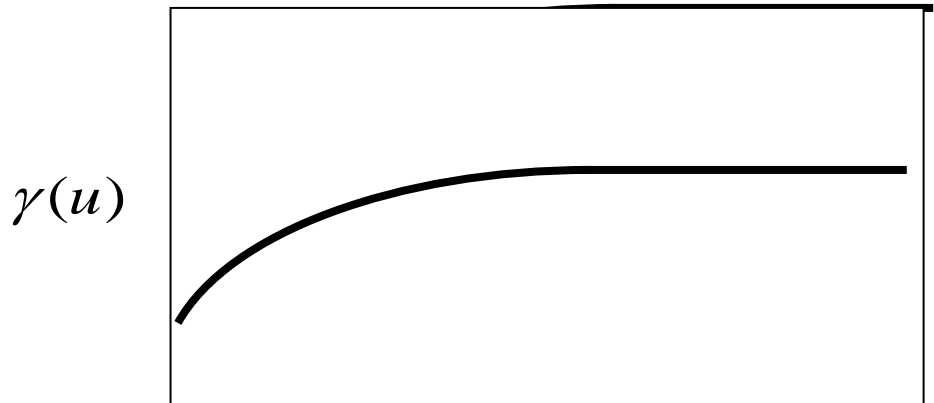
**Note** : the semi-variogram  $\gamma(u)$  is a **function** of  $u$ .  
According to Merriam Webster :

**-gram** : *noun combining form.* Etymology: Latin -*gramma*, from Greek, from *gramma*. drawing : writing : record **Examples** : telegram, histogram

Shouldn't  $\gamma(u)$  be a drawing or picture?

Well . . . semi-variograms are almost always plotted.

**NOW** : Why might a sample variogram look like this?



# Properties of the Theoretical Variogram

Let's suppose that  $S(\cdot)$  is a process such that 'near things are more related than far things', i.e we suppose that

$$\text{Correlation}[S(\mathbf{x}_i), S(\mathbf{x}_j)] = \rho(\mathbf{x}_i - \mathbf{x}_j) = \rho(u) \xrightarrow{u \rightarrow \infty} 0$$

In fact, we suppose that for  $u >$  some fixed **range**, the covariance is effectively zero. This means that our variogram will approach a fixed **sill**  $\sigma^2$  :

$$\gamma(u) = [\sigma^2(1 - \rho(u))] \xrightarrow{u \rightarrow \infty} \sigma^2$$

**NOW** : for points that are right next to each other (i.e. the same location) in our stochastic process, the correlation is

$$\text{Correlation}[S(\mathbf{x}_i), S(\mathbf{x}_i)] = \rho(0) = 1$$

so that  $\gamma(0) = [\sigma^2(1 - 1)] = 0$

# Properties of the OBSERVED Variogram

Remember our proposed model :

$$Y(\mathbf{x}_i) = \mu(d : \mathbf{x}_i) + S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2) + Z_i$$
$$Z_i \sim N(0, \tau^2)$$

$$\text{Var}(Y(\mathbf{x}_i) - Y(\mathbf{x}_j))$$

$$= \text{Var}(S(\mathbf{x}_i) + Z_i - S(\mathbf{x}_j) - Z_j)$$

$$= \text{Var}(S(\mathbf{x}_i) - S(\mathbf{x}_j)) + \text{Var}(Z_i - Z_j)$$

$$= 2[\sigma^2(1 - \rho(u))] + 2\tau^2$$

$$= 2\gamma(u) + 2\tau^2$$

$$= 2\gamma_Y(u)$$

$$= \mathbf{VARIogram\ of\ Y!}$$

We assume that  $S()$  and  $Z()$  are independent of each other, **AND** that the  $Z()$ 's are independent of each other



*Who cares!?!*



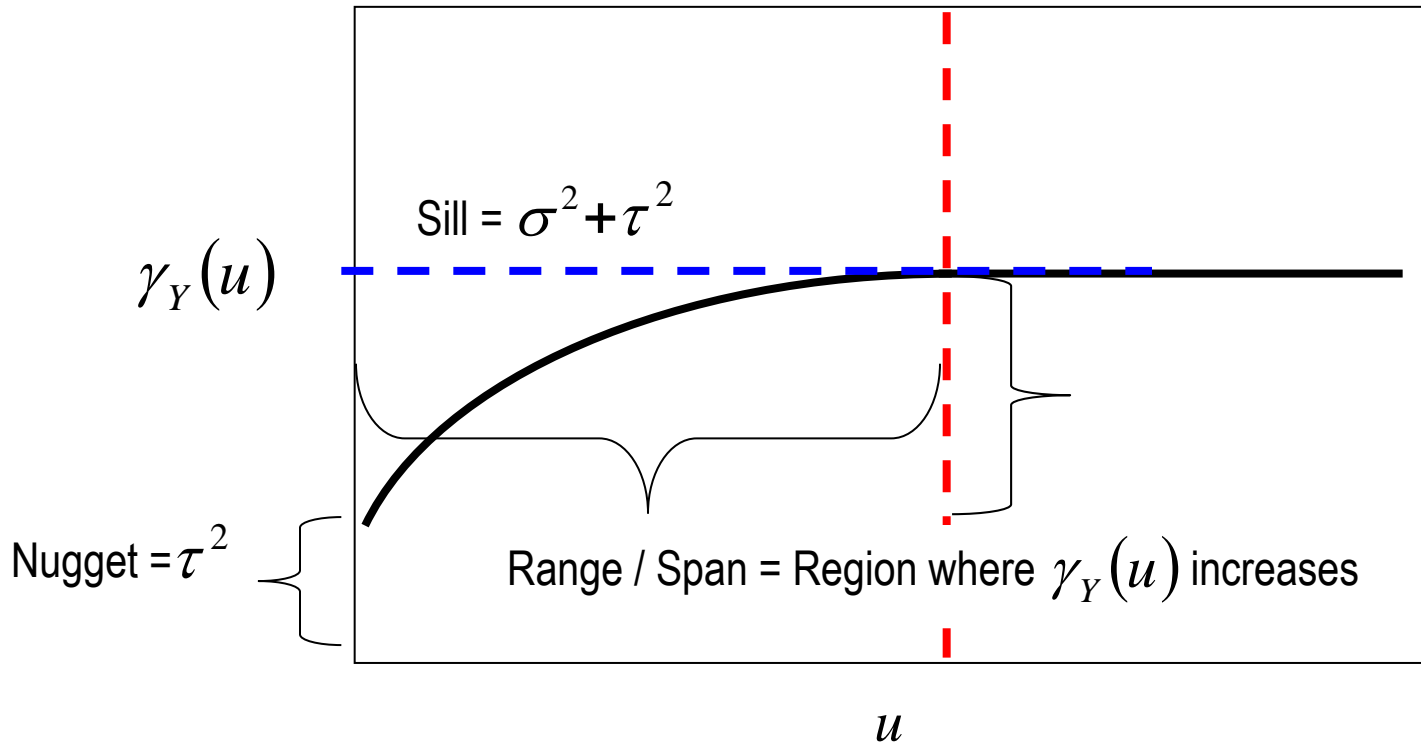
This means that while the variability of  $S(\cdot)$  at a single point is  $\gamma(0) = 0$ , the variability of  $Y(\cdot)$  at a single point is  $\tau^2$ .

**The variance of  $Y(\cdot)$  at a fixed location  $\tau^2$  is called the nugget effect**



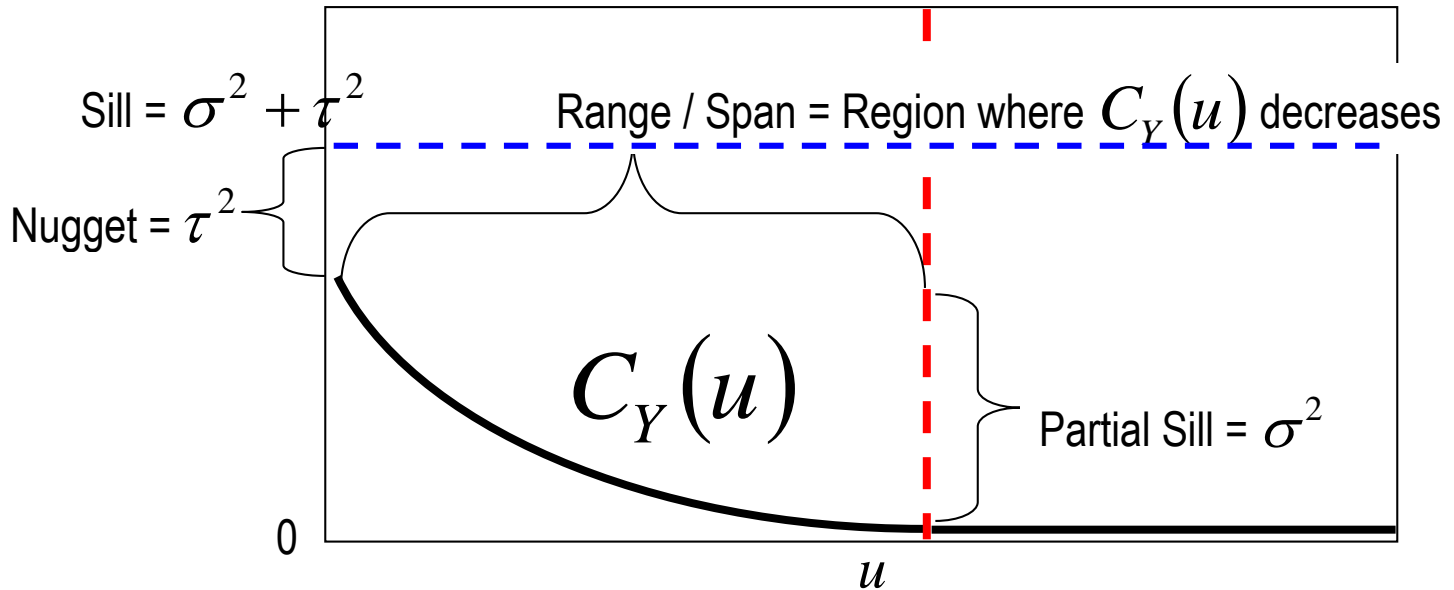
*This can be interpreted as measurement error at a single point, or as variability below the threshold of the measurement scale.*

Sometimes, there is discussion of the **partial sill**, the difference between the sill and the nugget effect (i.e.  $\sigma^2$ )



The corresponding **covariogram**  $C_Y(\mathbf{u})$  is simply

$$C_Y(\mathbf{u}) = \sigma^2 + \tau^2 - \gamma_Y(\mathbf{u})$$



# Estimating a Variogram

Since we don't actually have the true variogram  $\gamma_Y(u)$  (it's Greek – god's speak Greek), we have to estimate it.

**Method of Moments estimator** : The variogram can be thought of as an expected value or average :

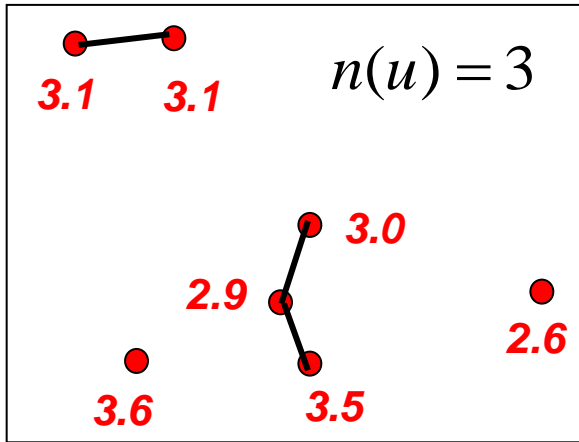
$$\gamma_Y(u) = \frac{1}{2} \text{Var}(Y(\mathbf{x}_i) - Y(\mathbf{x}_j)) = \frac{1}{2} E[(Y(\mathbf{x}_i) - Y(\mathbf{x}_j))^2]$$

In this case, we can take averages of the squared differences in  $Y(\mathbf{x}_i)$  values between pairs of points that are within some fixed **lag** apart for a series of lags :

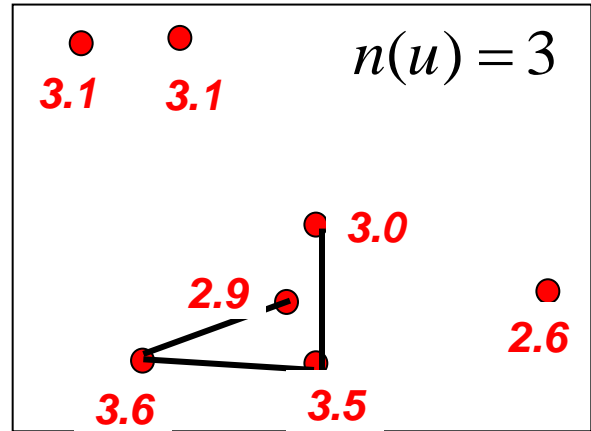
$$\hat{\gamma}_Y(u) = \frac{1}{2n(u)} \sum_{|x_i - x_j| \in [u, u+lag]} [(Y(\mathbf{x}_i) - Y(\mathbf{x}_j))^2]$$

for  $u = [0, lag, 2lag, 3lag, \dots]$  and where  $n(u)$  is the number of pairs within a particular distance range.

The next pages shows an example of how this averaging would work for 7 locations ( $42/2=21$  total pairs of points) : values in ***red italics*** are the values of  $Y(\mathbf{x}_i)$ .



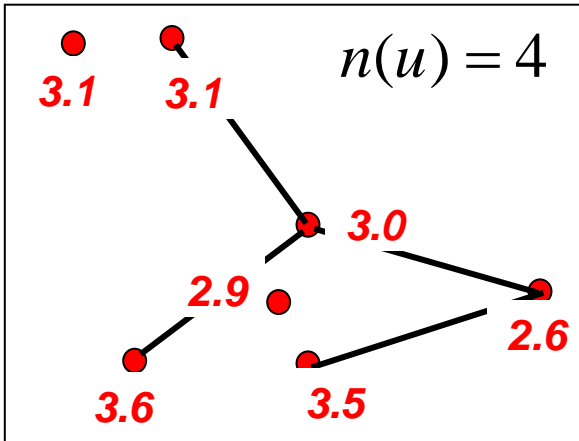
Distances between 0 and lag



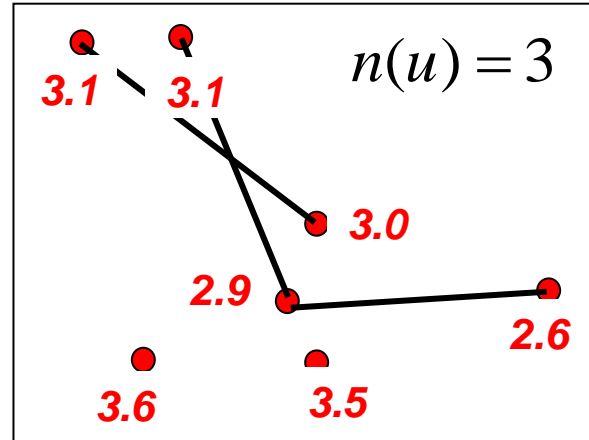
Distances between lag and 2\*lag

$$\hat{\gamma}(0) = \frac{1}{2*3} [(3.1-3.1)^2 + (2.9-3.0)^2 + (2.9-3.5)^2]$$

$$\hat{\gamma}(lag) = \frac{1}{2*3} [(3.0-3.5)^2 + (2.9-3.6)^2 + (3.6-3.5)^2]$$



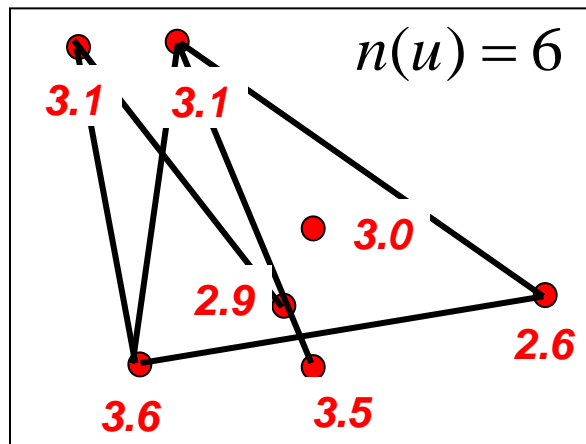
Distances between 2\*lag and 3\*lag



Distances between 3\*lag and 4\*lag

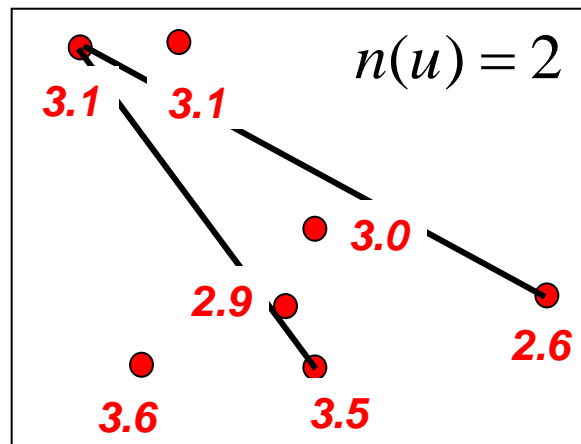
$$\hat{\gamma}(2lag) = \frac{1}{2*4} \left[ \begin{aligned} & (3.1-3.0)^2 + (3.6-3.0)^2 + (2.6-3.0)^2 \\ & + (2.6-3.5)^2 \end{aligned} \right]$$

$$\hat{\gamma}(3lag) = \frac{1}{2*3} \left[ (3.0-3.1)^2 + (2.9-2.6)^2 + (3.1-2.9)^2 \right]$$



Distances between 4\*lag and 5\*lag

$$\hat{\gamma}(4lag) = \frac{1}{2*6} \left[ \begin{aligned} & (3.1-3.6)^2 + (3.6-2.6)^2 + (2.6-3.1)^2 \\ & + (3.1-3.6)^2 + (3.1-3.5)^2 + (3.1-2.9)^2 \end{aligned} \right]$$



Distances between 5\*lag and 6\*lag

$$\hat{\gamma}(5lag) = \frac{1}{2*2} \left[ (3.5-3.1)^2 + (3.1-2.6)^2 \right]$$





**Variograms in R.** There are numerous R-packages that make variograms, all of which I find somewhat less than satisfying. I've finally settled on `geoR` as the best package. This uses the

function `variog()` which was used to produce the plots below.

Choosing `option="cloud"` gives a variogram cloud, essentially setting the lag to zero. The file online that contains the R-code is

`Example_Variogram.R.txt`

[http://reuningscherer.net/fes781/rscripts/example\\_variograms.r.txt](http://reuningscherer.net/fes781/rscripts/example_variograms.r.txt)

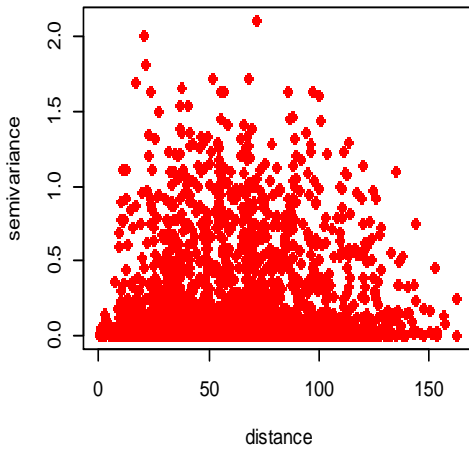
.....

***Example : Smoky Mountain pH Data.*** *The estimated variograms below use a lag distance of 4.5 and a lag distance of 10. The lag of 10 produces a somewhat smoother variogram.*

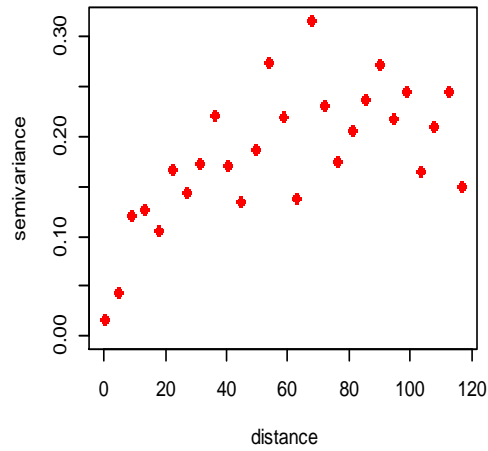
*Neither shows strong evidence of a nugget effect. However, both suggest that pH values at near locations are more correlated than values further apart.*



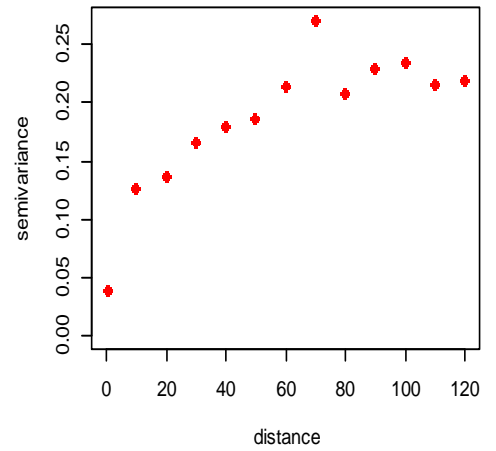
Variogram for pH Data, Cloud



Variogram for pH Data, Lag=4.5



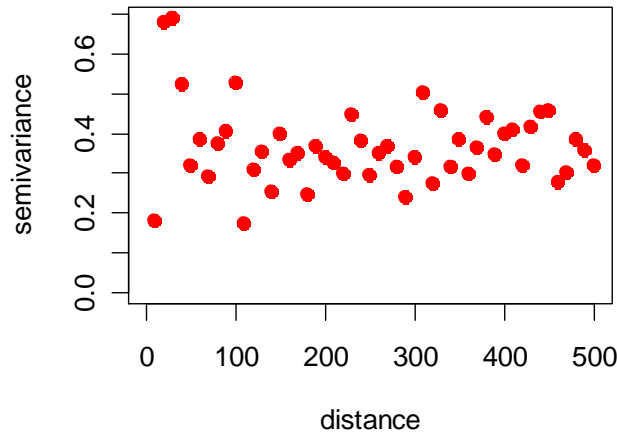
Variogram for pH Data, Lag=10



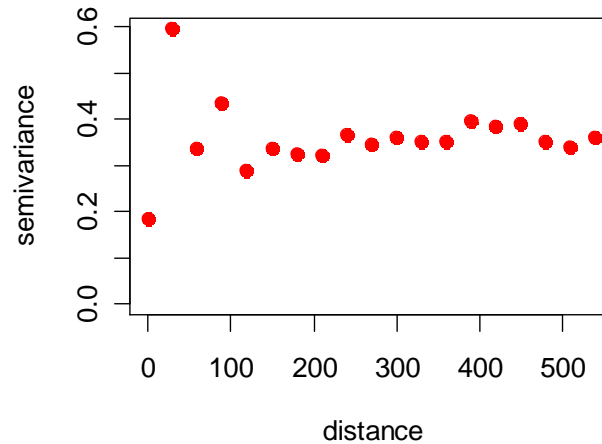
**Example : Lancashire Radon Data.** For this data, larger lags are appropriate since the scale is larger. Here, a lag of 10 is about 100 meters (I think!). The semi-variograms are basically flat – i.e. there is no evidence of any spatial correlation.



**Semi-Variogram for Radon Data, Lag=10**



**Semi-Variogram for Radon Data, Lag=30**



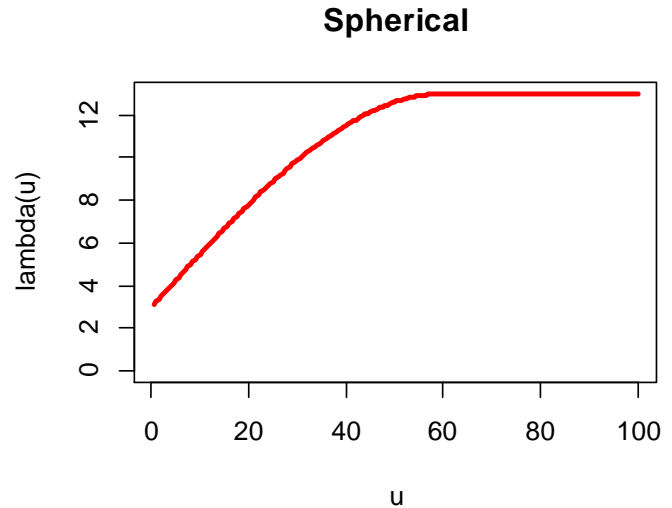
# Models for Stationary (Isotropic) Variograms

- We've already talked about properties of variograms.
- Our estimated variogram will not necessarily (i.e. probably won't) satisfy these properties (i.e. it won't be monotonic increasing, for example!)
- As in other estimation procedures, we can use a **parametric model** to try to 'fit' the estimated variogram and evaluate the model fit.
- A parametric model will also allow us to use information on spatial correlation to make **spatial prediction** (ah, now that would be useful . . .)
- Here are some models commonly used :

*Formulas for these (and a few other) variogram models are on p. 51-56 of MBG, p. 278-280 of Waller and Gotway, p. 440-442 of the SAS Mixed Model manual, etc. However, here are some properties of each :*

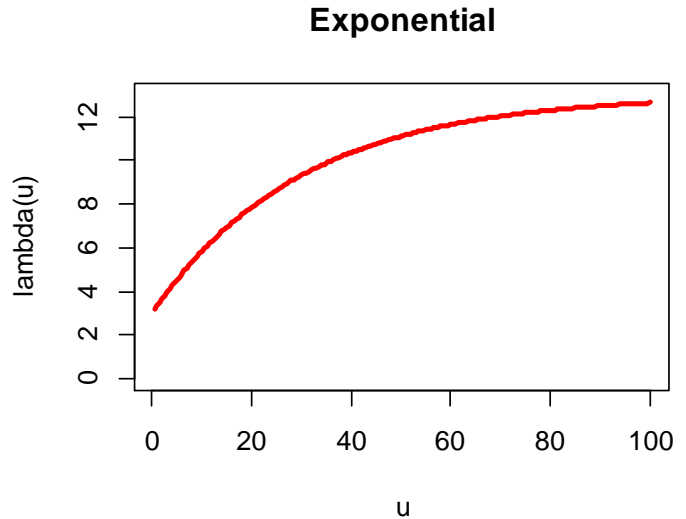
# Spherical

Linear near the origin, reaches a constant sill for  $u > a_s$  (the range)



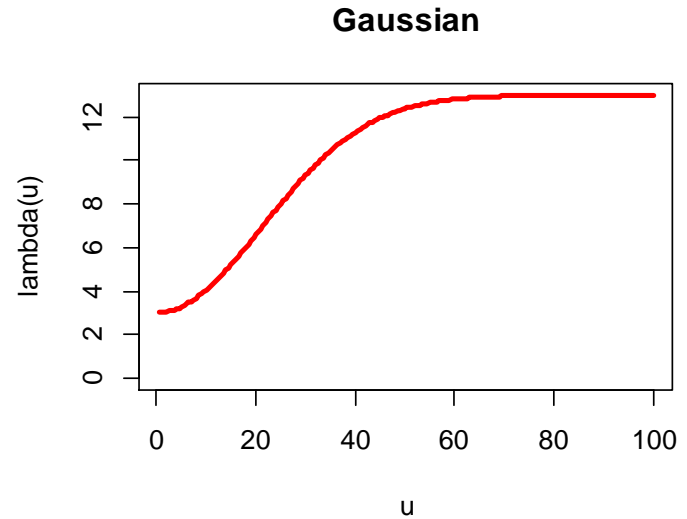
# Exponential

Rises more slowly from the origin than spherical and approaches the sill asymptotically (never quite gets there)



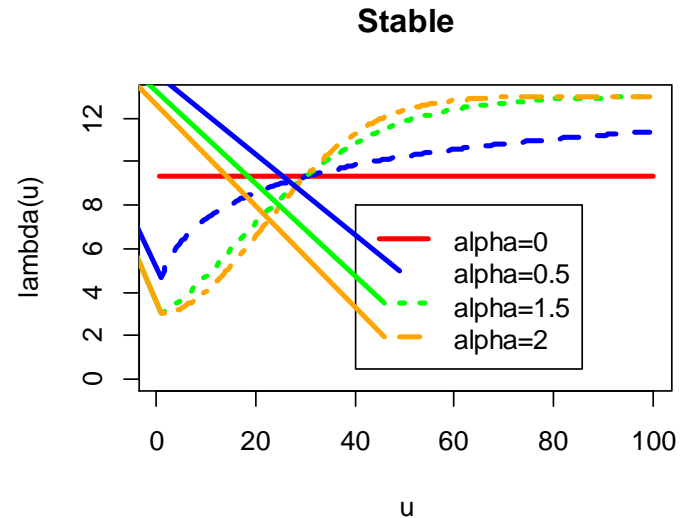
# Gaussian

Parabolic near the origin, rises to sill asymptotically, but more rapidly than the exponential. There are some 'issues' with this model discussed on p. 278 of Waller and Gotway



## Stable

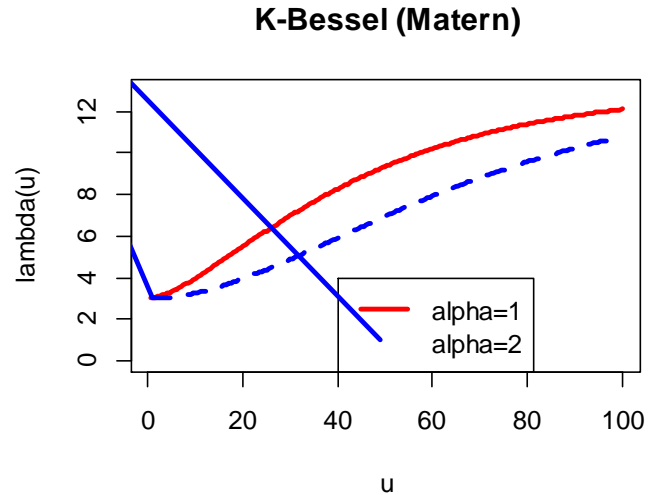
This is a family of models which includes exponential ( $\alpha=1$ ) and Gaussian ( $\alpha=2$ ). Approach sill asymptotically, behavior near origin determined by  $\alpha$ .





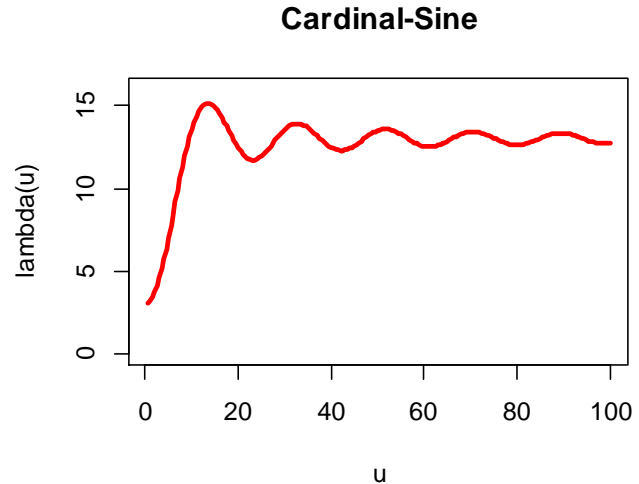
## K-Bessel (Matern)

Another family of models that approach sill asymptotically, behavior near origin determined by alpha. Exponential and Gaussian are members of this family as well.



## Cardinal-Sine

A member of a family of **hole-effect** models. Good for processes with negative spatial autocorrelation of periodic variability.



***ASIDE*** : The choice of variogram model has little effect on the final spatial prediction (except maybe *Cardinal-Sine*)– lag size has a much more profound effect (like radius in kernel smoothing . . .)

# Fitting Models to Variograms

- The most common method for fitting a model to an estimated variogram is to use Non-linear Least Squares Regression (NLSR).
- This fits a model to the data by minimizing

$$\sum_{j=1}^K \left[ (\hat{\gamma}_Y(u(j)) - \gamma_Y(u(j)))^2; \theta \right]$$

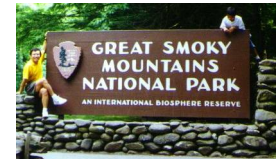
Here,  $K$  is the number of lags and  $\theta$  represents the shape parameters for each model (such as sill, nugget, range, etc).

This estimate has two problems :

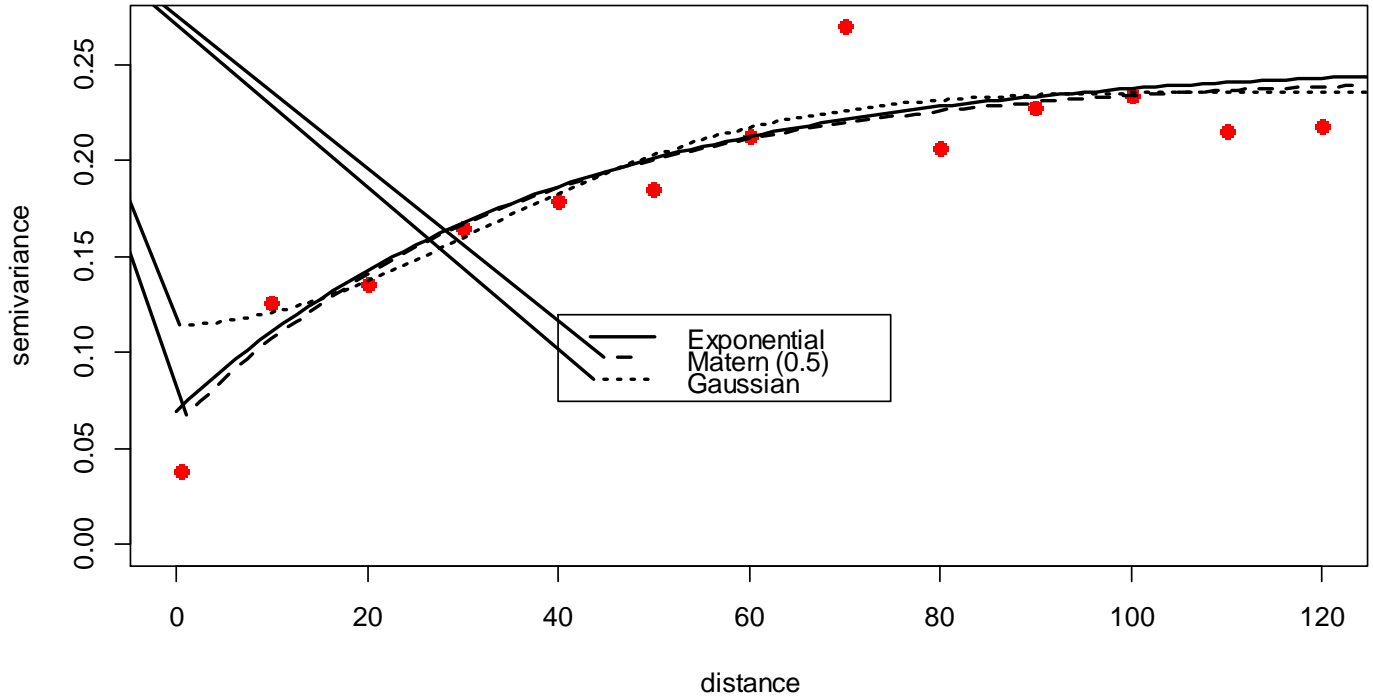
- Variance is not constant from one lag to the next (since each lag contains different numbers of pairs of values)
- Values are likely correlated.

Cressie (1985 – see W&G p.285) suggested using a minimization that accounts for these problems by making use of the variance-covariance matrix of the data.

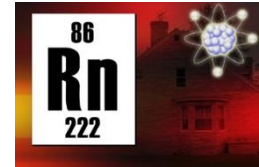
**Examples : Smoky Mountain pH Data.** *This uses the estimated variogram with lags of size 10. Three models were tried : exponential, Gaussian, and Matern ( $Kappa=0.5$ ). The exponential and Matern give the best (and basically identical) results. This uses Cressie's correction.*



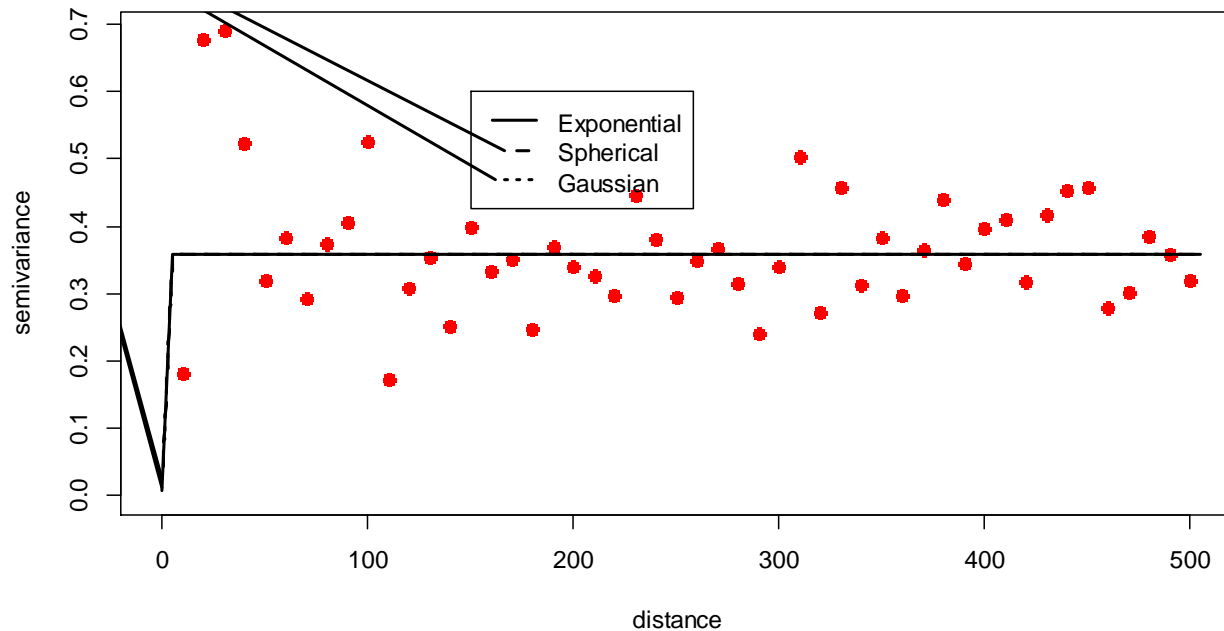
Variogram for pH Data, Lag=10



**Examples : Radon Data.** This uses the estimated variogram with lags of size 10. All models fit equally poorly!



**Semi-Variogram for Radon Data, Lag=30**





**Modeling Variograms in R.** The function `variofit()` fits parametric models to pre-calculated variograms. The function `lines.variogram` can be used to add these fitted values to empirical variograms. The calculations I used for these example datasets are given in `Example_Variogram.R`

[http://reuningscherer.net/fes781/rscripsts/example\\_variograms.r.txt](http://reuningscherer.net/fes781/rscripsts/example_variograms.r.txt)

.....

**Now** : let's go look at the Smoky Mountain's pH data in ArcGIS (10.1).

**Make sure you have the GeoStatistical Analyst Toolbar displayed** (`Customize` → `Toolbars` → `Geostatistical Analyst`)

First, we look at a  
**(semi)v**ariance cloud :  
this plots the squared  
differences

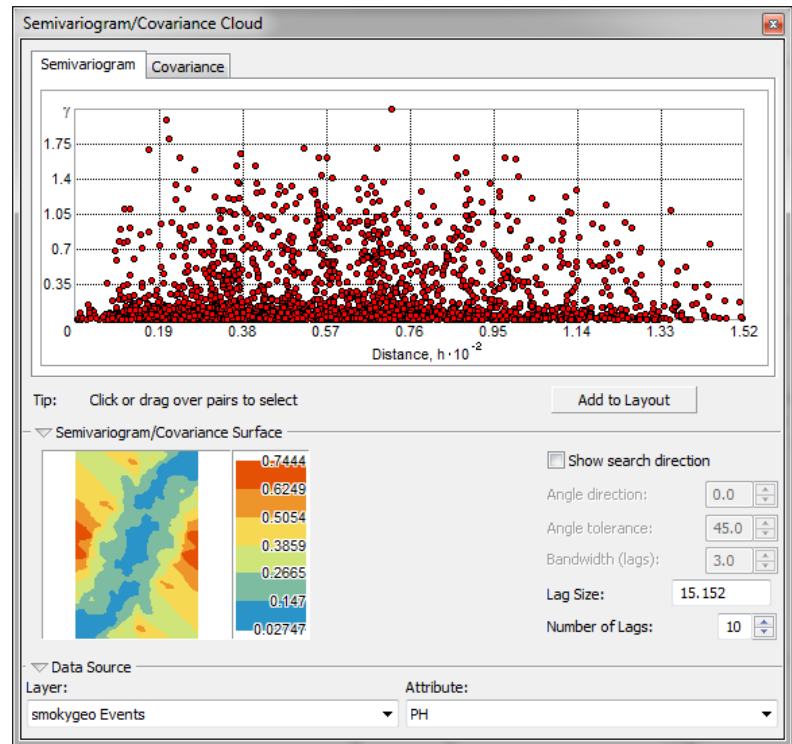
$(Y(\mathbf{x}_i) - Y(\mathbf{x}_j))^2$  for  
every pair of points  
 $[\mathbf{x}_i, \mathbf{x}_j]$ . You can get

this plot using

GeoStatistical  
Analyst → Explore  
Data →

Semivariogram /

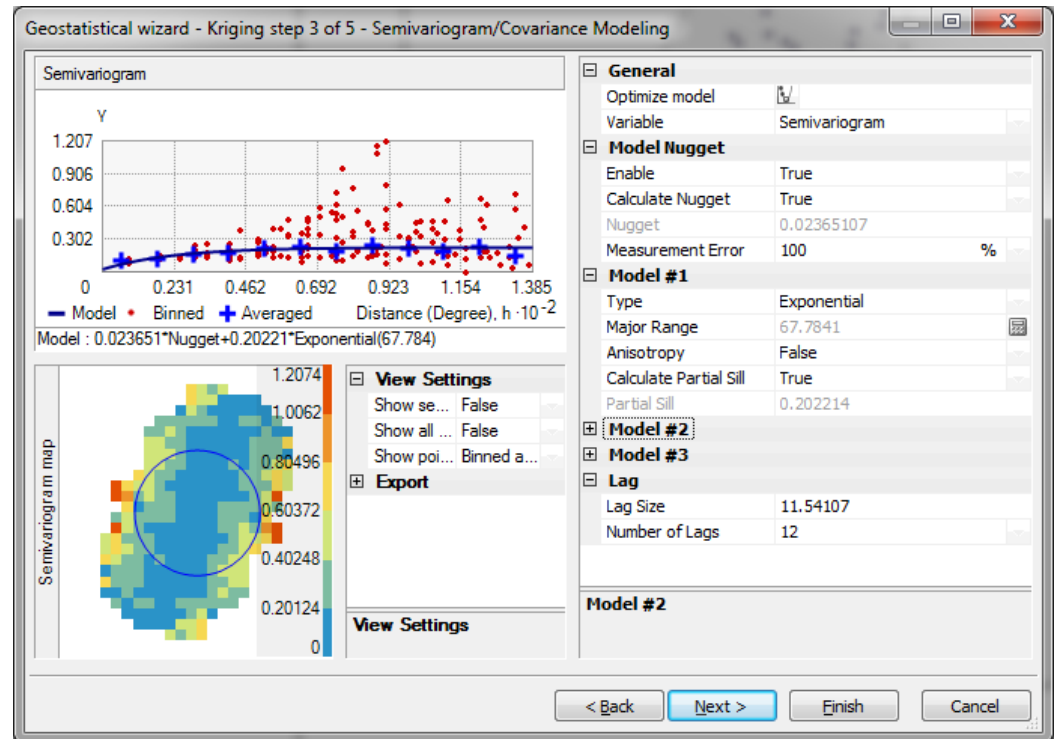
Covariance Cloud. Select the correct layer and Attribute  
(pH in this case). This plot isn't overly helpful really.





To get an estimated and fitted variogram, use GeoStatistical Analyst → GeoStatistical Wizard... Then choose kriging and select PH as the Data Field and ask for an ordinary kriging predictive map. This gives the following window :

*This has many, many more points than the estimated variograms we made in R. What's up with that?*



# Anisotropic Semivariogram Modeling

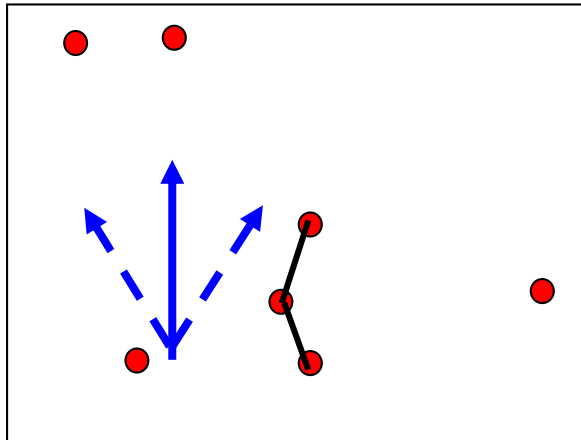
- It may be that correlations observed among the values of  $Y(\cdot)$  are **not** the same in every direction (think about water flow, airflow, surface gradients, etc.)
- In this case, variograms calculated in different directions may not yield the same results.



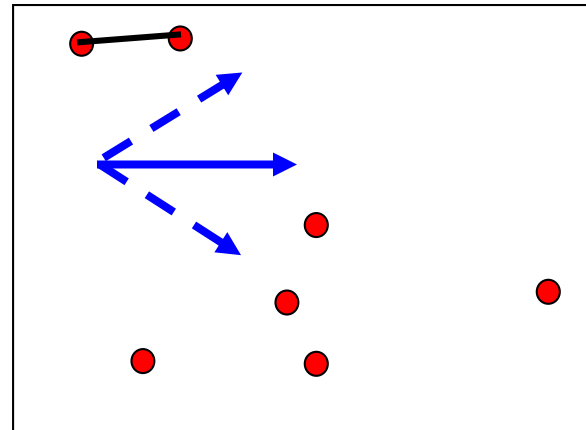
**Estimating Anisotropic Variograms** : The same formula is used

$$\hat{\gamma}_Y(\mathbf{u}) = \frac{1}{2n(\mathbf{u})} \sum_{|x_i - x_j| \in [\mathbf{u} - \frac{1}{2}lag, \mathbf{u} + \frac{1}{2}lag]} [(Y(\mathbf{x}_i) - Y(\mathbf{x}_j))^2]$$

However, now  $\mathbf{u}$  is a vector, not a fixed distance. The estimated variogram only considers differences between pairs of points  $[\mathbf{x}_i, \mathbf{x}_j]$  that are within some tolerance of the direction of  $\mathbf{u}$  AND such that the points are within the lag interval :



Pairs within 1 lag of 0 degrees  
(tolerance = 45 deg)



Pairs within 1 lag of 45 degrees  
(tolerance = 45 deg)

GIS simultaneously plots every estimate

$$\hat{\gamma}_Y(\mathbf{u}) = \frac{1}{2n(\mathbf{u})} \sum_{|x_i - x_j| \in [\mathbf{u} - \frac{1}{2}lag, \mathbf{u} + \frac{1}{2}lag]} [(Y(\mathbf{x}_i) - Y(\mathbf{x}_j))^2]$$

recalculated every 6 degrees or so between 0 and 180 degrees. This is why GIS shows so many points! However, there is a way to change the direction of the variogram in ArcGIS (set Show Search Direction to TRUE and then free-rotate using your mouse . . .)

**Examples : Smoky Mountains.** *There is some evidence that variograms change with direction (look at the y-axes to note changes for 120 degrees). Note that 0 degrees is the same as 180 degrees. At 25 degrees, we see correlation effective over a longer distance (i.e. takes longer to get to the sill). There is also a suggestions that we might need to take out a mean FIRST!*





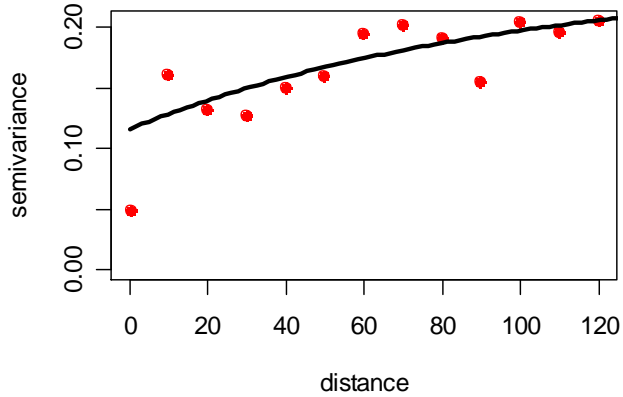
**Anisotropic Variograms in R.** The function `variog()` which we used for general variograms takes three options

`unit.angle`, `direction`, and `tolerance` which specify that a variogram should be fit only in a particular direction. The calculations I used for the plots above are in `Example_Variogram.R`

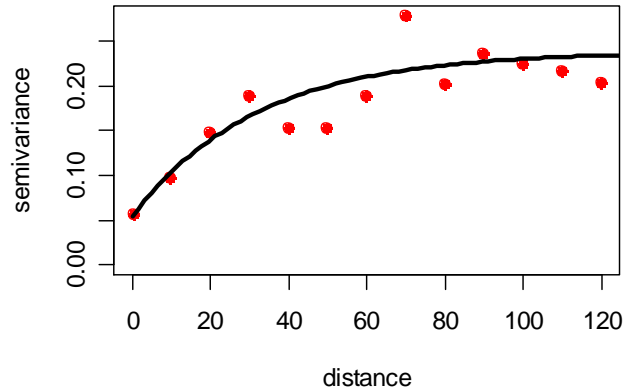
[http://reuningscherer.net/fes781/rscripsts/example\\_variograms.r.txt](http://reuningscherer.net/fes781/rscripsts/example_variograms.r.txt)

.....

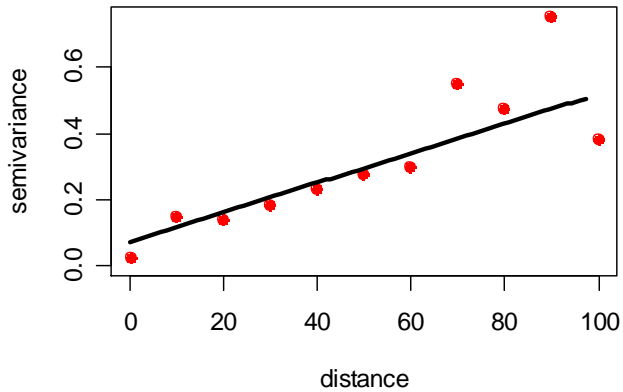
**Semi-Variogram for pH, Lag=10, 0 Deg.**



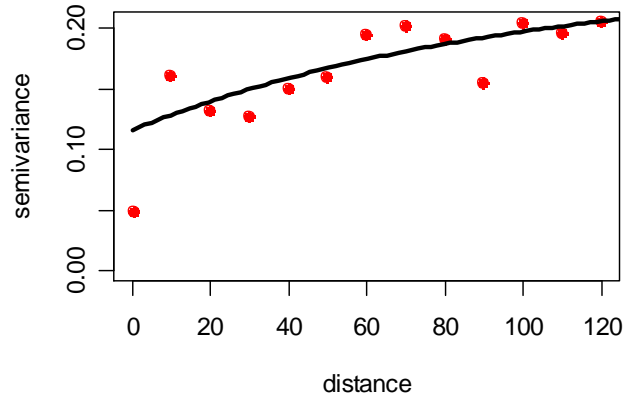
**Semi-Variogram for pH, Lag=10, 60 Deg.**



**Semi-Variogram for pH, Lag=10, 120 Deg.**



**Semi-Variogram for pH, Lag=10, 180 Deg.**



**WELL – It's finally time to start talking about making spatial prediction using spatial correlation!**



*Our Goal : we seek the Nirvana of*



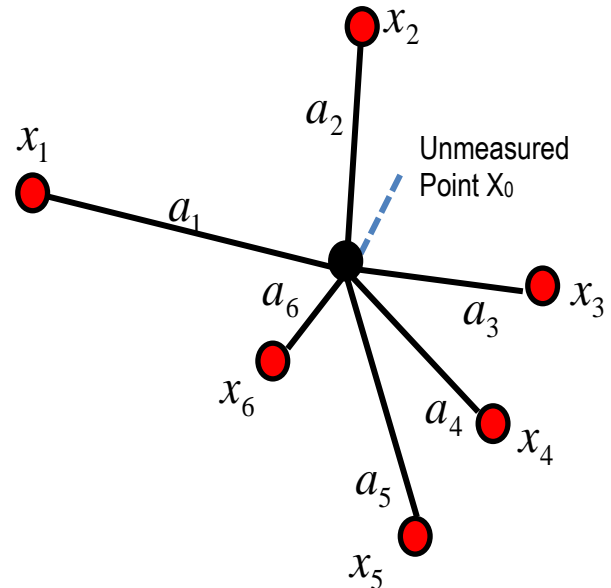
## **OPTIMAL SPATIAL PREDICTION** (called kriging)

- Named after South African mining geologist D.G. Krige who came up with basic concept (can pronounce with hard or soft 'g' . . .)

- Basic idea: prediction at a point  $\mathbf{x}_0$  involves taking a **weighted average** of the observed values (just like IDW or kernel smoothing, etc). :

$$\hat{Y}_{Krig}(\mathbf{x}_0) = \sum_{i=1}^N a_i Y(\mathbf{x}_i)$$

$a_i$  are the **weights**.



- **However**: in kriging, the  $a_i$  weights account for correlations among locations by using the **(co)variogram** (*ahh . . .*)



- Many types :
  - **Simple** – process is stationary, mean=0
  - **Ordinary** – process is stationary, mean=constant
  - **Universal** – process is stationary after removing the mean=some predictable trend surface

Here's how MBH writes the same thing : (p.140, 6.14)

$$\hat{Y}(\mathbf{x}_0) = \mu(\mathbf{x}_0) + \hat{S}(\mathbf{x}_0)$$

$$= \mu(\mathbf{x}_0) + \sum_{i=1}^N a_i (Y(\mathbf{x}_i) - \mu(\mathbf{x}_i))$$

*I'll just comment here that I think MBG is a bit confusing in that they simultaneously use  $S(\cdot)$  to be a mean zero stochastic process, and a non-mean zero process. In any case, think of  $S(\cdot)$  as the **SIGNAL** (the main stochastic process with or without a mean surface)*



## Simple Kriging (MBG 6.2)

Our model which assumes **stationarity** and **isotropy** :

$$Y(\mathbf{x}_i) = \mu(d : \mathbf{x}_i) + S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2) + Z_i$$

Furthermore, we're assuming that  $S(\cdot)$  and  $Z_i$  are both Gaussian (normal).

This means that our dataset  $Y(\mathbf{x}_1), Y(\mathbf{x}_2), \dots, Y(\mathbf{x}_N)$  represent observations from a multivariate normal distribution with mean vector  $\boldsymbol{\mu}(d, \mathbf{x})$  and variance

$$\sigma^2 \mathbf{V} = \sigma^2 \mathbf{R} + \tau^2 \mathbf{I}$$

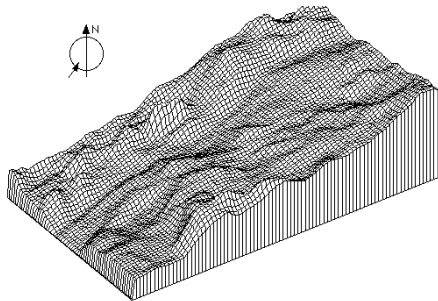
where the elements of  $\mathbf{R}$  are  $r_{ij} = \rho(\|\mathbf{x}_i - \mathbf{x}_j\|)$ , the correlation between points  $i$  and  $j$ !

**NOW** : let's suppose that we **KNOW** the mean  $\mu(d, x)$ .

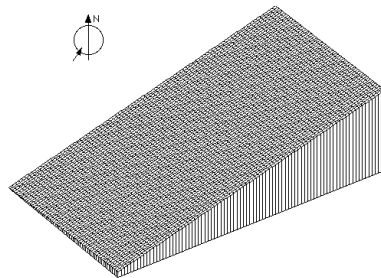
**How could that be possible??**



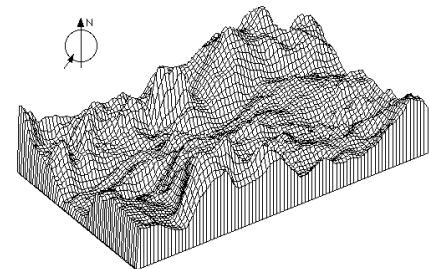
As we discussed, we could fit a trend surface and **remove the trend**. **AFTER** we've removed the trend, the residuals will have **mean ZERO** and common variance, say,  $\sigma^2$ .



*With Trend*



*Modeled Trend*



*Stationary, mean 0*

*We'll fix this to include non-zero means a bit later . . .*

In this case, our model is now a mean zero process where  $Y = \text{Signal} + \text{Noise}$

$$Y(\mathbf{x}_i) = S(\rho(u), \sigma^2) + Z_i$$

And our estimate of  $Y(\cdot)$  becomes the same as  $S(\cdot)$

$$\hat{Y}(\mathbf{x}_0) = \hat{S}(\mathbf{x}_0) = \sum_{i=1}^N a_i (Y(\mathbf{x}_i))$$

**Now** : what is “**optimal**” spatial prediction?

We want to choose weights  $a_i$  so that  $\hat{Y}_{Krig}(\cdot)$


- 1) Is Unbiased**
- 2) Has Minimum Mean-Squared Prediction Error**



**Unbiased** : this means that on average, our guess is correct:

$$E[\hat{Y}_{Krig}(\mathbf{x}_0)] = E[Y(\mathbf{x}_0)] = 0 \quad \text{Since we assume } \mu, Z \text{ have mean } 0$$

This means that

$$0 = E[\hat{Y}_{Krig}(\mathbf{x}_0)] = E\left[\sum_{i=1}^N a_i Y(\mathbf{x}_i)\right] = \sum_{i=1}^N a_i E[Y(\mathbf{x}_i)] = \sum_{i=1}^N a_i * 0$$


**So : no matter what weights we choose, our estimate will be unbiased.**

*So far, so good . . .*

**Minimum Mean-Squared Prediction Error : i.e. MINIMIZE THE VARIANCE!**

**Goal** : minimize

$$\text{Var}[\hat{S}_{Krig}(\mathbf{x}_0) - S(\mathbf{x}_0)]$$

*A few helpful reminders :*

$$\text{Var}[X - Y] = \text{Var}(X) + \text{Var}(Y) - 2\text{Cov}(X, Y)$$

*And the **Covariance** is*

$$\text{Cov}[X, Y] = E[XY - E(X)E(Y)]$$

---

**Now** : *I don't usually do math, but I think in this case it may help : so hang on (and if you don't like my version, see Isaaks and Srivastava Chapter 12 – other books skip too many details in my opinion) . . . .*



$$\begin{aligned} & \text{Var}[\hat{S}_{Krig}(\mathbf{x}_0) - S(\mathbf{x}_0)] \\ &= \text{Var}[\hat{S}_{Krig}(\mathbf{x}_0)] + \text{Var}[S(\mathbf{x}_0)] - 2\text{Cov}[\hat{S}_{Krig}(\mathbf{x}_0), S(\mathbf{x}_0)] \end{aligned}$$

1
2
3

This is just the usual rule for the Variance of a sum of random variables.

NOW : Let's solve each piece :

$$1 \quad \text{Var}[\hat{S}_{Krig}(\mathbf{x}_0)] = \text{Var}\left[\sum_{i=1}^N a_i Y(\mathbf{x}_i)\right]$$

*This is just an expanded rule for the Variance of sums*

$$= \sum_{i=1}^N \sum_{j=1}^N a_i a_j \text{Cov}(Y(\mathbf{x}_i), Y(\mathbf{x}_j))$$

*This is because we have a **stationary process** – we can express as the **covariogram!***

$$= \sum_{i=1}^N \sum_{j=1}^N a_i a_j C(\mathbf{x}_i - \mathbf{x}_j)$$

2  $Var[S(\mathbf{x}_0)] = \sigma^2$     *Since we have a **stationary process!***



$$\begin{aligned}
& \textcircled{3} \quad 2\text{Cov}\left[\hat{S}_{Krig}(x_0), S(x_0)\right] \\
& = 2\text{Cov}\left(\sum_{i=1}^N a_i Y(x_i), S(x_0)\right) \\
& = 2E\left(\left[\sum_{i=1}^N a_i Y(x_i)\right] * S(x_0)\right) - 2E\left(\sum_{i=1}^N a_i Y(x_i)\right) * E(S(x_0)) \\
& = 2E\left(\left[\sum_{i=1}^N a_i Y(x_i)\right] * S(x_0)\right) - 2\sum_{i=1}^N a_i E(Y(x_i)) * E(S(x_0)) \\
& = 2\sum_{i=1}^N a_i \text{Cov}(Y(x_i), Y(x_0)) \\
& = 2\sum_{i=1}^N a_i C(x_i - x_0)
\end{aligned}$$

Expand using formula for kriged estimate

Next, use definition of covariance

We're assuming  $Y$  has mean **ZERO!**

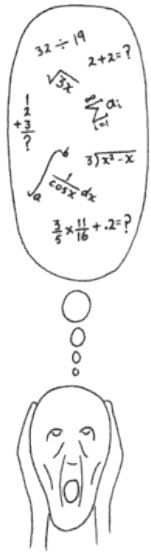
Regroup and use definition of Covariance again

Use **STATIONARITY!**

Put it all back together!

$$\text{Var}\left[\hat{S}_{Krig}(\mathbf{x}_0) - S(\mathbf{x}_0)\right]$$

$$= \sum_{i=1}^N \sum_{j=1}^N \underbrace{a_i a_j}_{1} \underbrace{C(\mathbf{x}_i - \mathbf{x}_j)}_{2} + \sigma^2 - 2 \sum_{i=1}^N \underbrace{a_i}_{3} \underbrace{C(\mathbf{x}_i - \mathbf{x}_0)}$$



This is just a function of the **covariogram** (or, equivalently, the (semi) **variogram** (we could substitute using  $C_Y(u) = \sigma^2 + \tau^2 - \gamma_Y(u)$  if we were so inclined . . .)

This seems a bit messy – how about writing this **squared prediction error** in **MATRIX FORM!**

$$\text{Var}\left[\hat{S}_{Krig}(\mathbf{x}_0) - S(\mathbf{x}_0)\right] = \mathbf{a}'\mathbf{C}\mathbf{a} + \sigma^2 - 2\mathbf{a}'\mathbf{c}_0$$

$$\begin{bmatrix} a_1 & a_2 & \dots & a_N \end{bmatrix} \begin{bmatrix} C(\mathbf{x}_1 - \mathbf{x}_1) & \dots & C(\mathbf{x}_1 - \mathbf{x}_N) \\ C(\mathbf{x}_2 - \mathbf{x}_1) & \dots & C(\mathbf{x}_2 - \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ C(\mathbf{x}_N - \mathbf{x}_1) & \dots & C(\mathbf{x}_N - \mathbf{x}_N) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} + \sigma^2 - 2 \begin{bmatrix} a_1 & a_2 & \dots & a_N \end{bmatrix} \begin{bmatrix} C(\mathbf{x}_0 - \mathbf{x}_1) \\ C(\mathbf{x}_0 - \mathbf{x}_2) \\ \vdots \\ C(\mathbf{x}_0 - \mathbf{x}_N) \end{bmatrix}$$

- C** gives covariances (i.e. **covariogram**) between **all pairs of observed points**. **THIS IS EQUIVALENT TO  $\sigma^2\mathbf{V} = \sigma^2\mathbf{R} + \tau^2\mathbf{I}$  in MBG**
- a** is the vector of **weights** (that's what we're trying to optimize)
- c<sub>0</sub>** is the vector of covariances (**covariogram**) between all **observed points and the new data point  $\mathbf{x}_0$**

**SO** : To minimize this, we differentiate with respect to all the weights  $a_1, a_2, \dots, a_N$ , one at a time. This gives the **simple kriging equations** (there are  $N$  of these equations) :

$$\sum_{j=1}^N a_j C(\mathbf{x}_i - \mathbf{x}_j) = C(\mathbf{x}_0 - \mathbf{x}_i) \quad , \quad \text{for } i = 1, 2, \dots, N$$

OR : in **MATRIX** form

$$\mathbf{C}\mathbf{a}_{opt} = \mathbf{c}_0 \quad \begin{bmatrix} C(\mathbf{x}_1 - \mathbf{x}_1) & \cdots & C(\mathbf{x}_1 - \mathbf{x}_N) \\ C(\mathbf{x}_2 - \mathbf{x}_1) & \cdots & C(\mathbf{x}_2 - \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ C(\mathbf{x}_N - \mathbf{x}_1) & \cdots & C(\mathbf{x}_N - \mathbf{x}_N) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} C(\mathbf{x}_0 - \mathbf{x}_1) \\ C(\mathbf{x}_0 - \mathbf{x}_2) \\ \vdots \\ C(\mathbf{x}_0 - \mathbf{x}_N) \end{bmatrix}$$

**FINALLY** : Solving for the weights  $\mathbf{a}_{opt}$  (just multiply by the inverse of the covariance matrix  $\mathbf{C}^{-1}$ ), we finally have the **simple kriging weights** :

$$\mathbf{a}_{opt} = \mathbf{C}^{-1} \mathbf{c}_0$$

WHAT DOES THIS MEAN?!?!?



The optimal weights  $\mathbf{a}_{opt}$  depend on

- $\mathbf{C}$ , the covariances between all observed locations
- $\mathbf{c}_0$ , the covariances between our new point and all other locations

Because of **stationarity**, these covariances depend only on the **distance vectors** between locations, i.e. on the



**(CO)VARIOGRAM!**



Of course, for **REAL** data, we don't know the (co) variogram (remember, it's a function).

**However** : we know how to

- **Calculate an estimated variogram**
- **Fit a parametric model to the estimated variogram!**

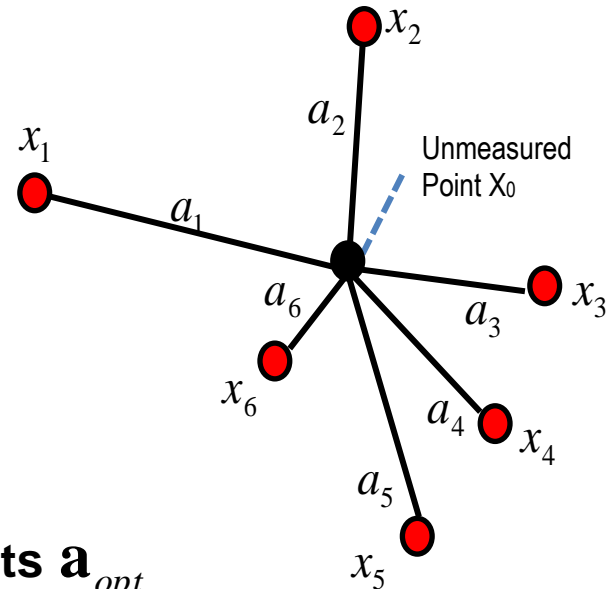
*This is why we went to all that trouble . . . to do kriging!*



**SO** : why are we doing this? (*back to page 70*)

We're trying to estimate the value of  $S()$  at a new location  $\mathbf{x}_0$  as a weighted combination of the values of  $Y()$  at observed points :

$$\hat{S}(\mathbf{x}_0) = \sum_{i=1}^N a_i (Y(\mathbf{x}_i))$$



**But** : We now have our **optimal weights**  $\mathbf{a}_{opt}$

SO : our estimate of the value of  $S()$  at one single point  $\mathbf{x}_0$  :

$$\begin{aligned} \hat{S}_{Krig}(\mathbf{x}_0) &= \mathbf{a}'_{opt} \mathbf{Y} (\mathbf{C}^{-1} \mathbf{c}'_0)' \mathbf{Y} = \mathbf{c}'_0 \mathbf{C}^{-1} \mathbf{Y} \\ &= \mathbf{r}' \mathbf{V}^{-1} \mathbf{Y} = \mathbf{r}' (\mathbf{R} + \tau^2 \mathbf{I})^{-1} \mathbf{Y} \end{aligned}$$

MBG Notation



$$\hat{S}_{Krig}(\mathbf{x}_0) = [C(\mathbf{x}_0 - \mathbf{x}_1) \quad C(\mathbf{x}_0 - \mathbf{x}_2) \quad \dots \quad C(\mathbf{x}_0 - \mathbf{x}_N)] \begin{bmatrix} C(\mathbf{x}_1 - \mathbf{x}_1) & \dots & C(\mathbf{x}_1 - \mathbf{x}_N) \\ C(\mathbf{x}_2 - \mathbf{x}_1) & \dots & C(\mathbf{x}_2 - \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ C(\mathbf{x}_N - \mathbf{x}_1) & \dots & C(\mathbf{x}_N - \mathbf{x}_N) \end{bmatrix}^{-1} \begin{bmatrix} Y(\mathbf{x}_1) \\ Y(\mathbf{x}_2) \\ \vdots \\ Y(\mathbf{x}_N) \end{bmatrix}$$

*Now : You might think this seems like a lot of computation for just one point  $\mathbf{x}_0$  . . .*

However, notice that when kriging over some domain  $A$

- $\mathbf{C}^{-1}\mathbf{Y}$  only has to be calculated **once** : **it doesn't depend on  $\mathbf{x}_0$** .
- Only  $\mathbf{C}_0$  has to be calculated for each point  $\mathbf{x}_0$  : this is the **covariogram relating  $\mathbf{x}_0$  to the set of observed locations  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$** .





**Now** : how about the **uncertainty** in our kriging estimate? I.e., what is the actual expected error of our estimate?

Back to the equation at the top of page 80, but now plug in the values for the weights that give **Minimum Expected Unbiased Error** to get the **KRIGING VARIANCE!**

$$\begin{aligned} & \text{Var}[\hat{S}_{Krig}(\mathbf{x}_0)] \\ &= \mathbf{a}'_{opt} \mathbf{C} \mathbf{a}_{opt} + \sigma^2 - 2\mathbf{a}'_{opt} \mathbf{c}_0 \\ &= (\mathbf{C}^{-1} \mathbf{c}_0)' \mathbf{C} \mathbf{C}^{-1} \mathbf{c}_0 + \sigma^2 - 2(\mathbf{C}^{-1} \mathbf{c}_0)' \mathbf{c}_0 \\ &= \sigma^2 - \mathbf{c}_0' \mathbf{C}^{-1} \mathbf{c}_0 \\ &= \sigma^2 - \mathbf{r}'(\mathbf{R} + \tau^2 \mathbf{I})^{-1} \end{aligned} \quad \text{This is equation 6.15 in MBG on page 141}$$

Notice that if our data actually has **NO** spatial correlation, our kriging variance is just  $\sigma^2$  : basically, we can achieve a small reduction in estimated variance by accounting for covariance!

**If instead we want uncertainty in  $Y(\cdot)$** , then we simply add the variance of  $Z(\cdot)$  :

$$\hat{\sigma}_{Krig}^2 = \sigma^2 - \mathbf{r}'(\mathbf{R} + \tau^2 \mathbf{I})^{-1} + \tau^2$$

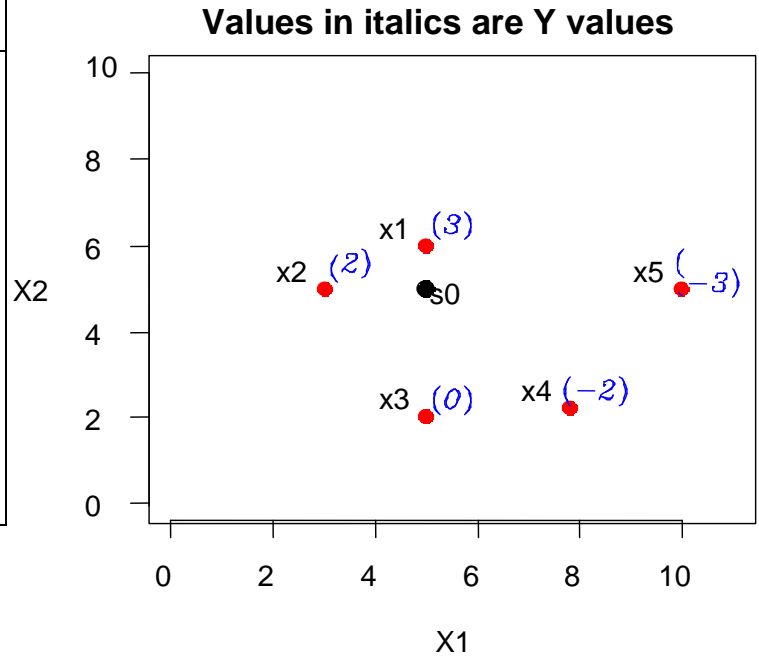
**SO!!!!** If our prediction errors are approximately **normally distributed**, a **95% Prediction Interval for  $Y(\mathbf{x}_0)$**  is

$$\hat{Y}_{Krig}(\mathbf{x}_0) \pm 1.96 \hat{\sigma}_{Krig}(\mathbf{x}_0)$$

*Maybe it's time for a bit of data . . .*

***Example*** (made up). Consider five observed points and one point where we'd like to make an estimate :

Points	$X_1$	$X_2$	$Y$
$x_1$	5	6	3
$x_2$	3	5	2
$x_3$	5	2	0
$x_4$	7.8	2.2	-2
$x_5$	10	5	-3
$x_0$	5	5	?



Notice that the  $Y(\ )$  sum to zero (as if they are the residuals after removing a surface trend). Also, the sample variance is 6.5 (our estimate of  $\sigma^2 + \tau^2$ )

Suppose that we happen to know that points have an exponential, isotropic variogram with **partial sill** = 6 (i.e.  $\sigma^2$ ) and range = 8 :

$$\gamma_s(u) = 6(1 - e^{-3u/8})$$

(typically, we'd estimate and fit a variogram model)

**Now** : we need distances between observed points and distances between  $x_0$  and the observed points

**Distances among observed points**

	$x_1$	$x_2$	$x_3$	$x_4$
$x_2$	2.2			
$x_3$	4.0	3.6		
$x_4$	4.7	5.6	2.8	
$x_5$	5.1	7.0	5.8	3.6

**Distances from  $x_0$  to each point**

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$
$x_0$	1	2	3	4	5


*Since we have isotropy and stationarity, this is all we need to calculate the variogram values for each pair of points :*

**Matrix  $\mathbf{C}$  of Variogram values  
between observed points**

$$\begin{bmatrix} 0 & 3.36 & 4.68 & 4.98 & 5.1 \\ 3.36 & 0 & 4.44 & 5.28 & 5.58 \\ 4.68 & 4.44 & 0 & 3.9 & 5.34 \\ 4.98 & 5.28 & 3.9 & 0 & 4.44 \\ 5.1 & 5.58 & 5.34 & 4.44 & 0 \end{bmatrix}$$

**Vector  $\mathbf{C}_0$  of Variogram values  
between observed points and  $\mathbf{x}_0$**

$$\begin{bmatrix} 0.31 \\ 0.53 \\ 0.68 \\ 0.78 \\ 0.85 \end{bmatrix}$$


$$C(\mathbf{x}_5 - \mathbf{x}_1) = 6(1 - e^{-3*5.1/8})$$

**SO** : our kriging estimate at the point  $\mathbf{x}_0=(5,5)$  is

$$\hat{S}_{Krig}(\mathbf{x}_0) = \mathbf{c}_0 \mathbf{C}^{-1} \mathbf{Y} = 1.9$$

**And** : our estimated kriging variance at this point is

$$\hat{\sigma}_{Krig}^2(\mathbf{x}_0) = \sigma^2 + \tau^2 - \mathbf{c}_0' \mathbf{C}^{-1} \mathbf{c}_0 = 6.5 - 0.5 = 6$$

**SO** : a 95% prediction interval for  $\mathbf{x}_0$  is

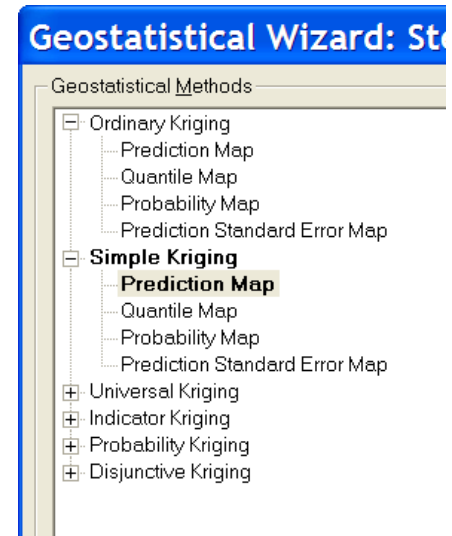
$$1.9 \pm 1.96 * \sqrt{6} = (-2.9, 6.7)$$

Now we just have to do this for every other point in our domain  $A$  . . . Which is why we have computers.



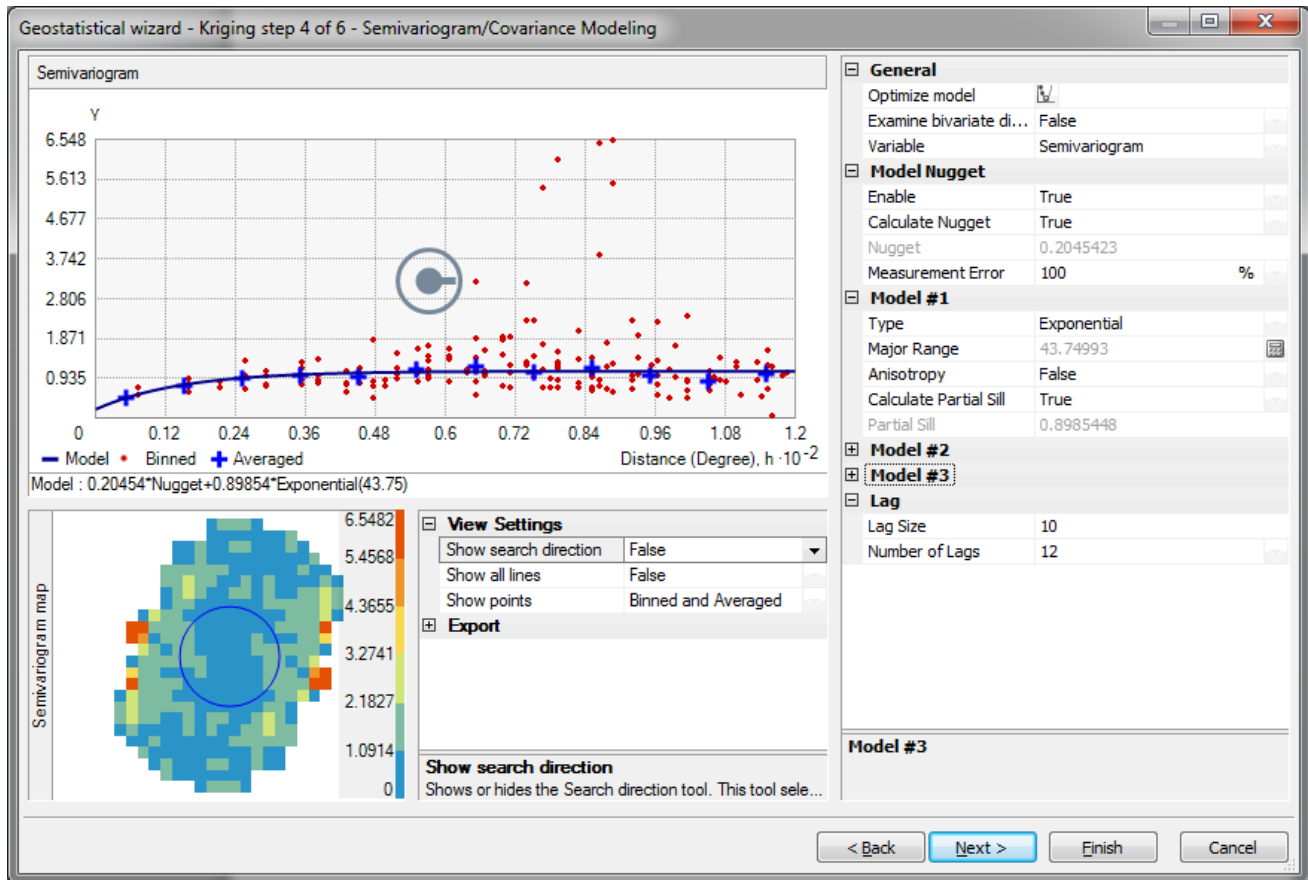


**Kriging in ArcGIS** : GeoStatistical Analyst → Geostatistical Wizard. Then choose Kriging (and make sure you choose the correct attribute). For now, we'll do simple kriging. Prediction Map will give the estimates  $\hat{Y}_{Krig}(\cdot)$  over the entire domain  $A$ ; Prediction Standard Error Map will give  $\hat{\sigma}_{Krig}(\cdot)$  over  $A$ .

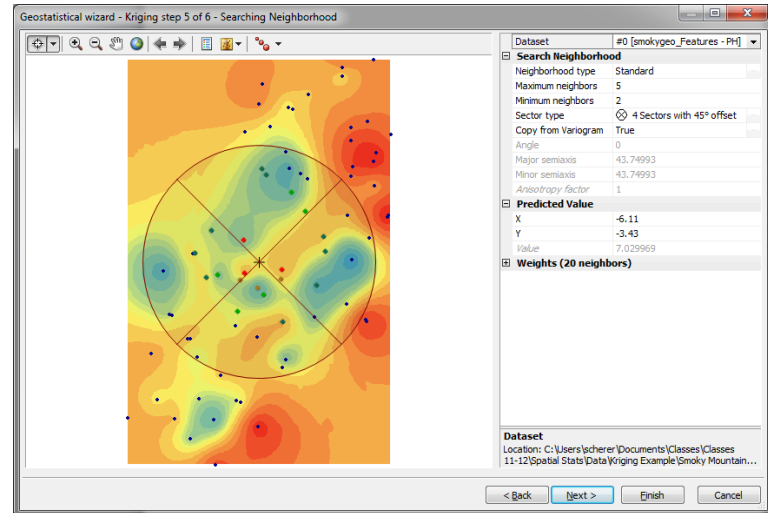


**Examples : Smoky Mountains pH data.** Last time we fit several variograms to fit this data. Seems like an exponential variogram with a lag of about 10 worked reasonably well. For the moment, we just fit a model that is isotropic (same in every direction)



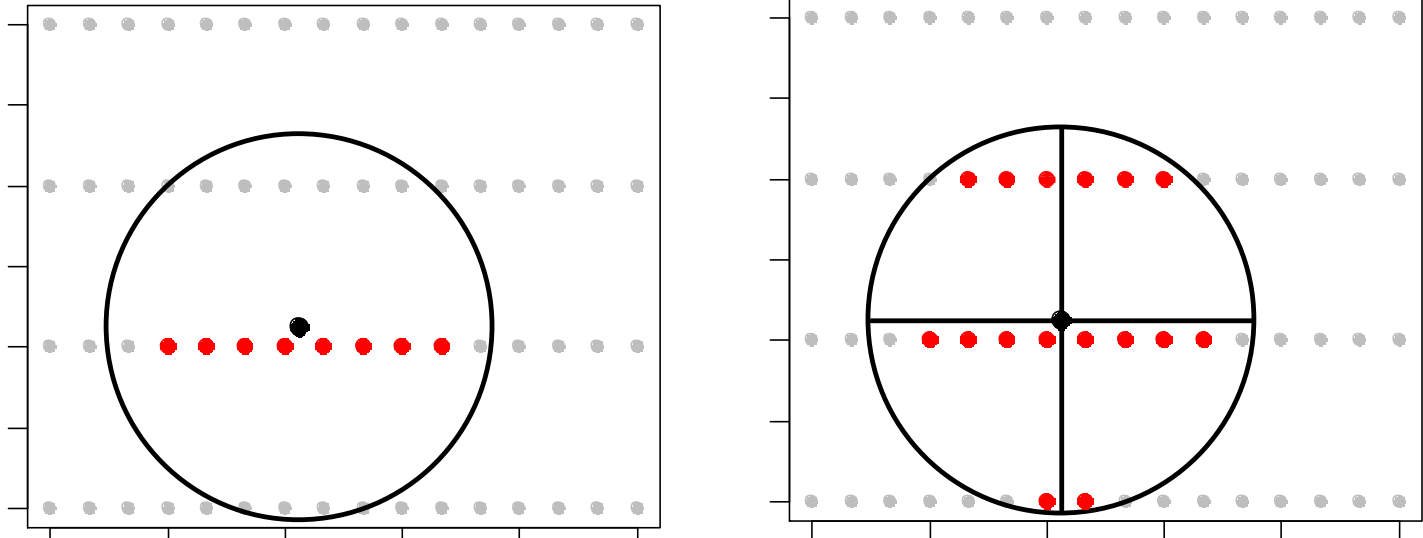


*Now : in practice, GIS will cut-off the number of neighbors to use in making predictions at a particular point (to make computation easier – this acknowledges that far away points don't have much influence). This is specified in the Number of Neighbors to include.*



*Furthermore, GIS will ensure that a minimum number of points are included in each of several directions : this is set by the Shape Type.*

*The idea is to protect against the effect of the original data point sampling scheme (transects, grids, etc). The pictures below (p.182-183 from the ArcGIS Geostat manual) show an example :*

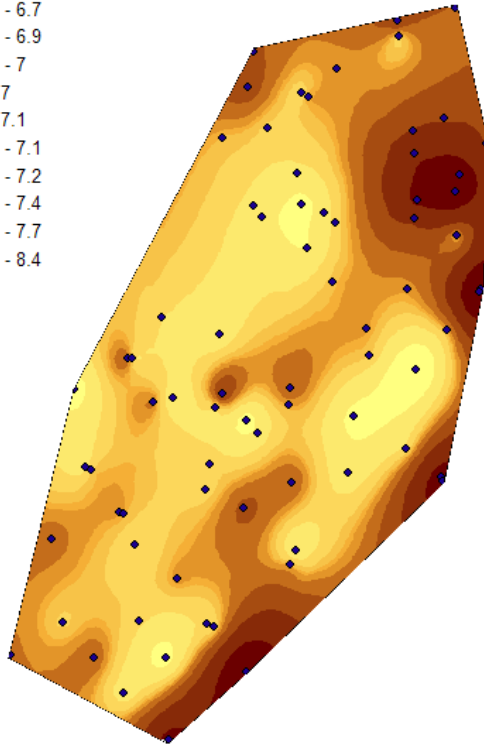


**SO :** *Here are the results! Much smoother than the Inverse Distance Map, and now we have estimates of the errors! Note that the errors increase to about 0.43, and errors are largest furthest from observed values. Incidentally, the standard deviation of all pH values is 0.44!*

Legend  
Simple Kriging  
Prediction Map  
PH  
Filled Contours

- 6.4 - 6.7
- 6.7 - 6.9
- 6.9 - 7
- 7 - 7
- 7 - 7.1
- 7.1 - 7.1
- 7.1 - 7.2
- 7.2 - 7.4
- 7.4 - 7.7
- 7.7 - 8.4

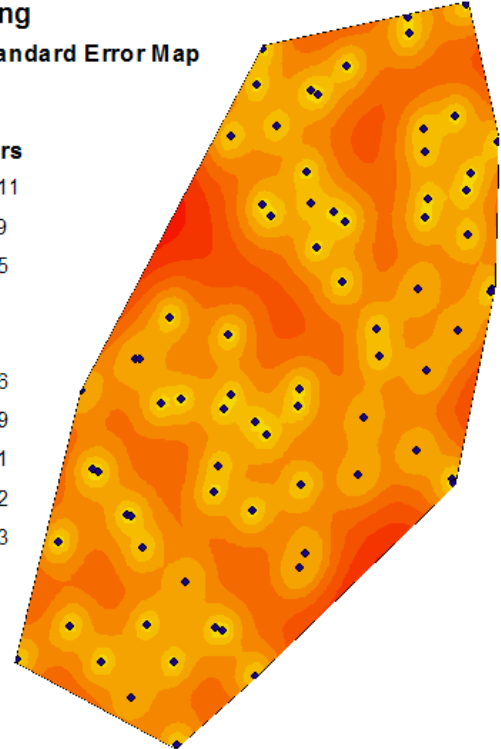
Predicted  
Surface



Legend  
Simple Kriging  
Prediction Standard Error Map  
PH

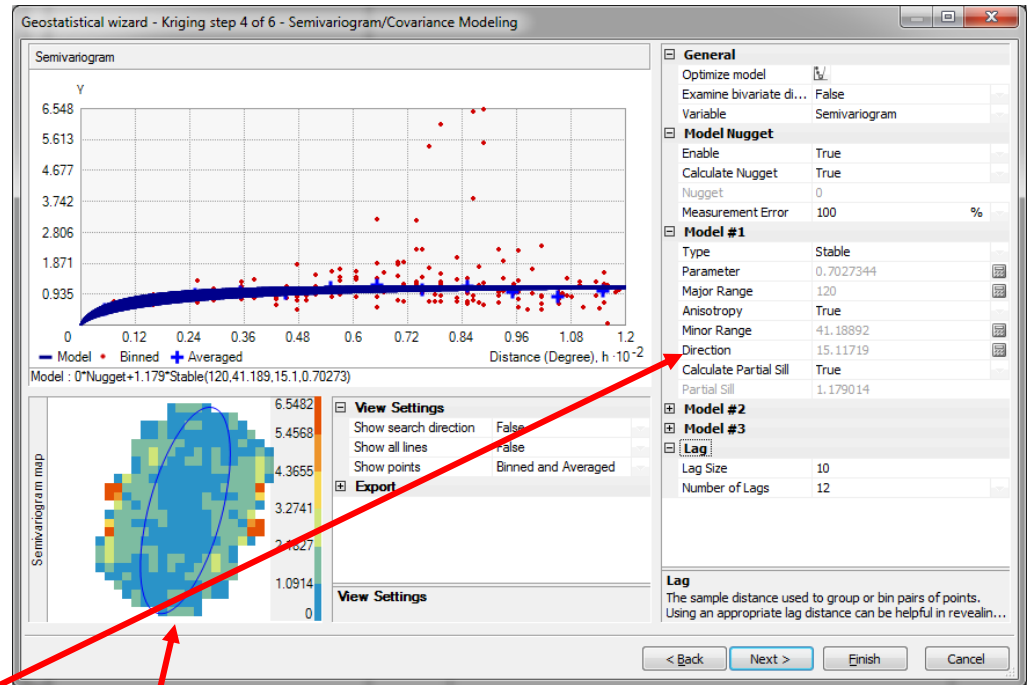
- Filled Contours
- 0.015 - 0.11
  - 0.11 - 0.19
  - 0.19 - 0.25
  - 0.25 - 0.3
  - 0.3 - 0.33
  - 0.33 - 0.36
  - 0.36 - 0.39
  - 0.39 - 0.41
  - 0.41 - 0.42
  - 0.42 - 0.43

Error  
Surface



**NOW** : *Let's revisit the question of **anisotropy**. We saw last time that there was evidence that covariance was not the same in every direction : it seems that covariance is higher in the direction of about  $30^0$  (this means that in the direction of  $30^0$ , the variogram rises to the sill more slowly, while at about  $120^0$ , the variogram rises to the sill more rapidly.*

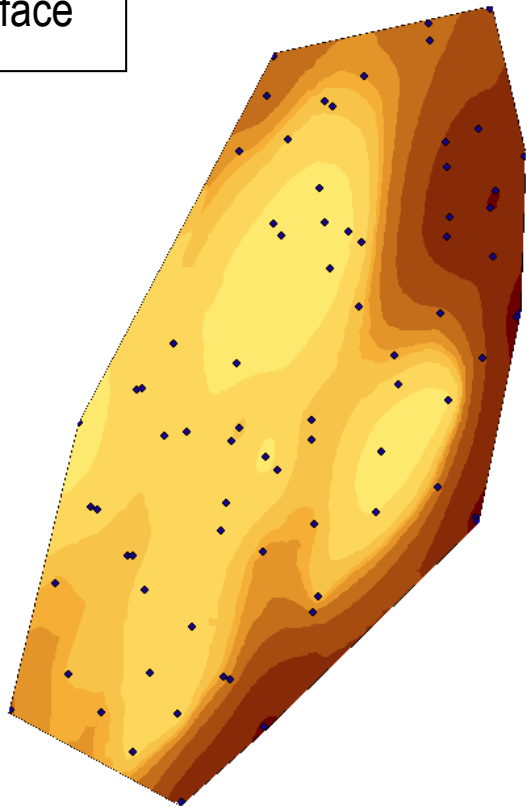
*We can account for this by using a search neighborhood with an elliptical shape that has a major axis parallel to the direction of highest covariance. Practically, this means clicking the anisotropy button in ArcGIS :*



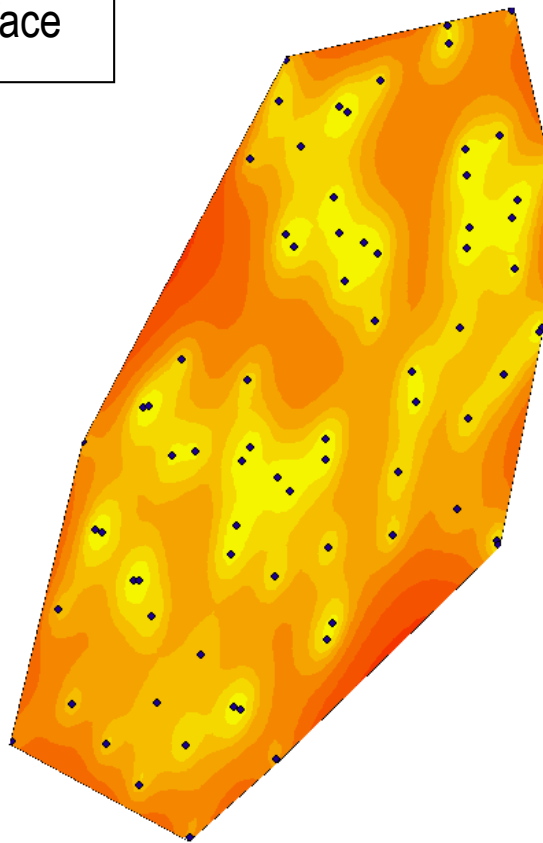
*This shows the direction in which there seems to be greater correlation*

*Notice that the resulting maps have contours that are stretched in the direction of the major elliptical axis.*

Predicted  
Surface



Error  
Surface







## Ordinary Kriging (MBG 6.2.2)

**ONCE AGAIN** : Our model which assumes **stationarity** and **isotropy** :

$$Y(\mathbf{x}_i) = \mu(d : \mathbf{x}_i) + S(\mu = 0, \mathbf{x}_i, \rho(u), \sigma^2) + Z_i$$

In Simple Kriging, we assumed that  $\mu(d, \mathbf{x}) = 0$ .

**NOW** : we move slightly further and assume that the mean is some **UNKNOWN** constant  $\mu(d, \mathbf{x}) = \mu$ . In fact, the mean is not required to be constant over the entire surface – it is assumed to be constant over all points that contribute non-zero weights to estimation at a new point.

We still want  $\hat{Y}_{Krig} ( )$

1) **Unbiased**

2) **Minimum Mean-Squared Prediction Error**

**Unbiased** : this means that on average, our guess is correct:

$$E[\hat{Y}_{Krig}(\mathbf{x}_0)] = E[Y(\mathbf{x}_0)] = \mu$$

This means that

$$\mu = E[\hat{Y}_{Krig}(\mathbf{x}_0)] = E\left[\sum_{i=1}^N a_i Y(s_i)\right] = \sum_{i=1}^N a_i E[Y(\mathbf{x}_i)] = \mu \sum_{i=1}^N a_i$$

**SO** : our kriging prediction must have  $\sum_{i=1}^N a_i = 1$

*i.e. weights must sum to 1.*

**Minimum Mean-Squared Prediction Error** : we still want to minimize

$$\text{Var}[\hat{S}_{Krig}(\mathbf{x}_0) - S(\mathbf{x}_0)] \text{ such that } \sum_{i=1}^N a_i = 1$$

Minimizing with a constraint requires the use of a **Lagrange multiplier**  $m$  :

$$\text{Var}[\hat{S}_{Krig}(\mathbf{x}_0) - S(\mathbf{x}_0)] - 2m \left( \sum_{i=1}^N a_i - 1 \right)$$

*This is just adding zero!*

*This is the same calculation as on page 44.*

*The 2 is just to make things nicer later when we take derivatives . .*

At this point, the calculations are the same as on page 77-80, with one additional term.

$$\begin{aligned} & \text{Var}\left[\hat{S}_{Krig}(\mathbf{x}_0) - S(\mathbf{x}_0)\right] \\ &= \sum_{i=1}^N \sum_{j=1}^N a_i a_j C(\mathbf{x}_i - \mathbf{x}_j) + \sigma^2 - 2 \sum_{i=1}^N a_i C(\mathbf{x}_0 - \mathbf{x}_i) + 2m \left( \sum_{i=1}^N a_i - 1 \right) \end{aligned}$$

This is again a function of the **COVARIOGRAM** between all observed pairs of points and between our new point and all observed points.

SO : To minimize this, we differentiate with respect to all the weights  $a_1, a_2, \dots, a_N$ , and with respect to  $m$ . This gives the **ordinary kriging equations** :

$$\sum_{j=1}^N a_j C(\mathbf{x}_i - \mathbf{x}_j) + m = C(\mathbf{x}_0 - \mathbf{x}_i) , \text{ for } i = 1, 2, \dots, N$$

$$\sum_{i=1}^N a_i = 1$$

In matrix form :

$$\begin{bmatrix} C(\mathbf{x}_1 - \mathbf{x}_1) & \cdots & C(\mathbf{x}_1 - \mathbf{x}_N) & 1 \\ C(\mathbf{x}_2 - \mathbf{x}_1) & \cdots & C(\mathbf{x}_2 - \mathbf{x}_N) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ C(\mathbf{x}_N - \mathbf{x}_1) & \cdots & C(\mathbf{x}_N - \mathbf{x}_N) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \\ m \end{bmatrix} = \begin{bmatrix} C(\mathbf{x}_0 - \mathbf{x}_1) \\ C(\mathbf{x}_0 - \mathbf{x}_2) \\ \vdots \\ C(\mathbf{x}_0 - \mathbf{x}_N) \\ 1 \end{bmatrix}$$

or more concisely :  $\mathbf{C}_+ \mathbf{a}_0 = \mathbf{c}_{+0}$

*The + 's indicate that these matrices are 'expanded' by a row/column to account for the unbiasedness constraint*

Solving for  $\mathbf{a}_{opt}$  and  $m$  gives the **ordinary kriging weights** :

$$\mathbf{a}_{opt} = \mathbf{C}_+^{-1} \mathbf{c}_{+0}$$

*Basically, the same results as for simple kriging, but with an expanded matrix  $\mathbf{C}_+$  which contains an unbiasedness constraint that the weights sum to 1!*

As before, we have

$$\begin{aligned} \hat{S}_{Krig}(\mathbf{x}_0) &= \mathbf{a}'_{opt} \mathbf{Y} (\mathbf{C}^{-1} \mathbf{c}'_0)' \mathbf{Y} = \mathbf{c}'_{+0} \mathbf{C}_+^{-1} \mathbf{Y} \\ &= \mu + \mathbf{r}' \mathbf{V}^{-1} (\mathbf{Y} - \mu \mathbf{I}) \end{aligned}$$

**FINALLY** : estimate of errors includes expanded weight vector.

$$\hat{\sigma}_{Krig}^2(\mathbf{x}_0) = \sigma^2 - \mathbf{c}_{+0}' \mathbf{C}_+^{-1} \mathbf{c}_{+0} + \tau^2$$



## Simple Kriging vs. Ordinary Kriging



What's really going on here?? (MBG, p. 137)

### Simple Kriging

- We “simply” estimate an overall CONSTANT mean as  $\hat{\mu} = \bar{Y}$  (we estimate the mean as the sample mean of all the Y values)
- We subtract this constant mean and krig the residuals  $Y - \hat{\mu}$
- Weights are not constrained to sum to one.

## Ordinary Kriging

- We estimate an overall CONSTANT mean using the generalized least squares estimator :

$$\hat{\mu} = (\mathbf{1}'\mathbf{V}^{-1}\mathbf{1})^{-1} \mathbf{1}'\mathbf{V}^{-1}\mathbf{Y}$$

- This is equivalent to requiring the weights to sum to one.

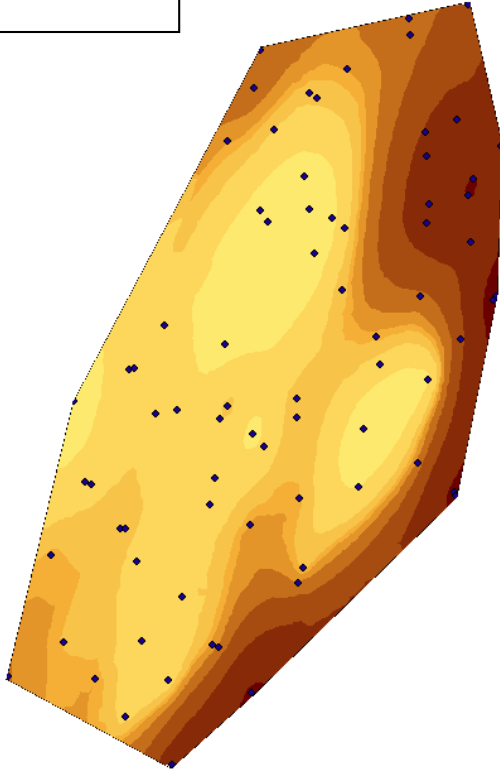
**Ordinary Kriging in ArcGIS** : GeoStatistical Analyst  
→ Geostatistical Wizard. Then choose Kriging  
and Ordinary Kriging.

***Examples : Smoky Mountains pH data. Here are results of ordinary kriging (still using anisotropic exponential variogram with lag of 10). Results are almost identical to that of simple kriging.***

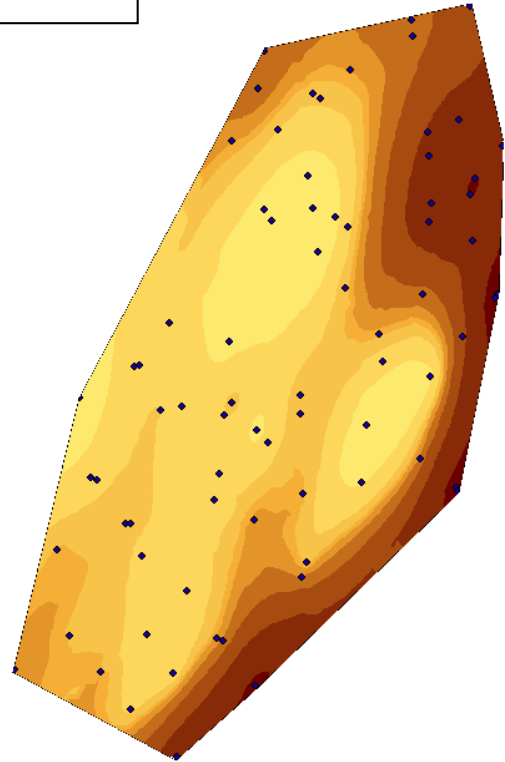




Simple  
Kriging



Ordinary  
Kriging



**Examples : Lancashire Radon Data.** Here are results of ordinary kriging (using isotropic exponential variogram). This is a pretty flat surface (i.e. no real evidence of trends.)

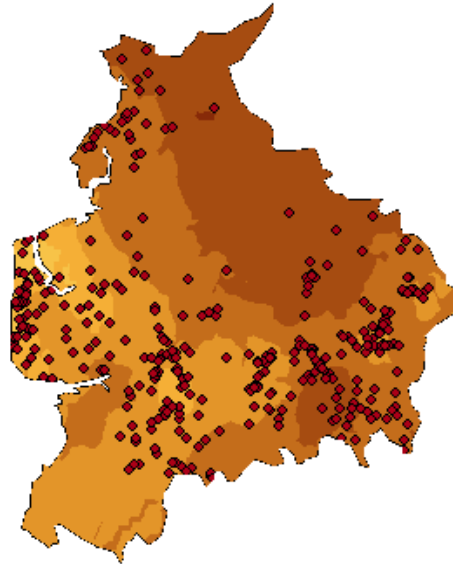
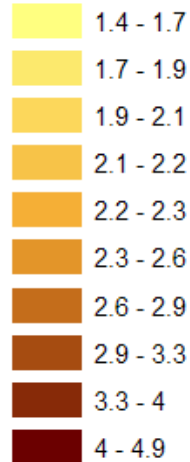
**Legend**

**Ordinary Kriging**

**Prediction Map**

**log(radon)**

**Filled Contours**



# Kriging Weights (MBG, 6.4)



**MBG** has a very nice discussion of the effect of changing various parameters on the kriging weights through examples. I've endeavored to reproduce several of their figures (approximately) and have also created some visual 'simulations' so we can see what's going on. The file is online

<http://reuningscherer.net/fes781/rscripts/KrigingWeightsExample.R.txt>

## 1) Effect of Location

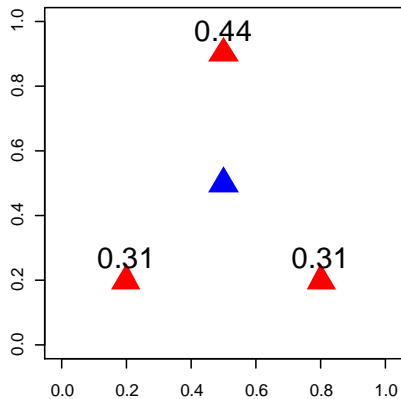
- a. Our variogram model generally means that near observed data points have more effect than far data points on the estimate of  $Y$  at a new location (see *first column below*).

- b. Proximity Effect – we would expect that if two observed data points are near each other, they themselves are correlated. Consequently, kriging **down weights multiple observations near each other** (since we aren't really getting additional independent pieces of information) (*see second column below*).
- c. Collinear Masking Effect – Under many Matern covariogram models, if two observed points lie along line from a new location, the nearest observed point will have a large positive weight, while the further collinear point may have a negative weight (*see third column below*).

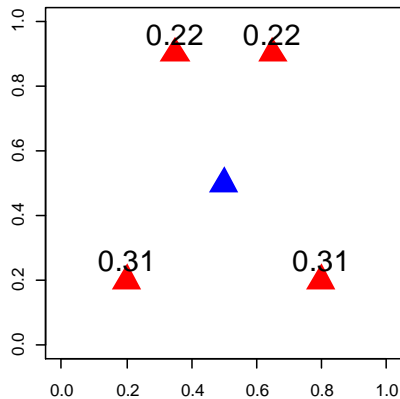
*The plots below were created using both simple kriging and ordinary kriging. In each case*

$$\mu = 0, \sigma^2 = 1, \tau^2 = 0, \phi = 0.8, \kappa = 1.5$$

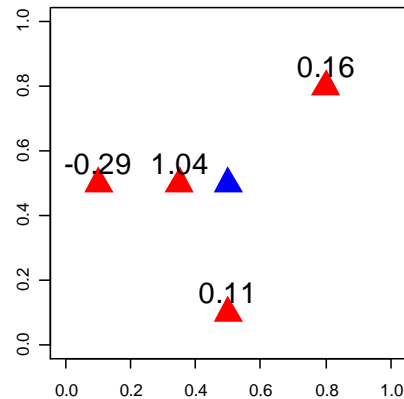
Simple Kriging Weights



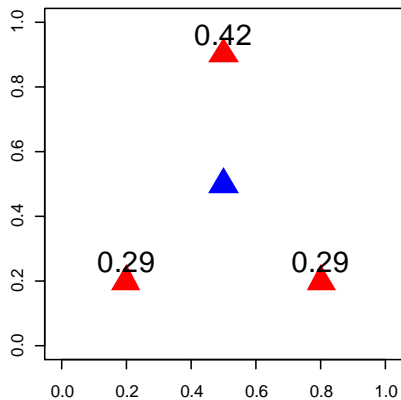
Simple Kriging Weights



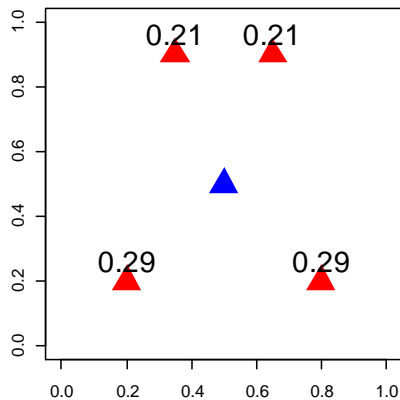
Simple Kriging Weights



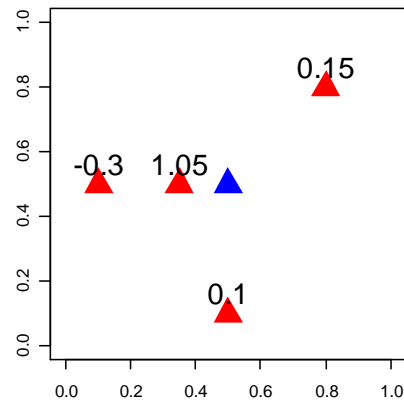
Ordinary Kriging Weights



Ordinary Kriging Weights



Ordinary Kriging Weights



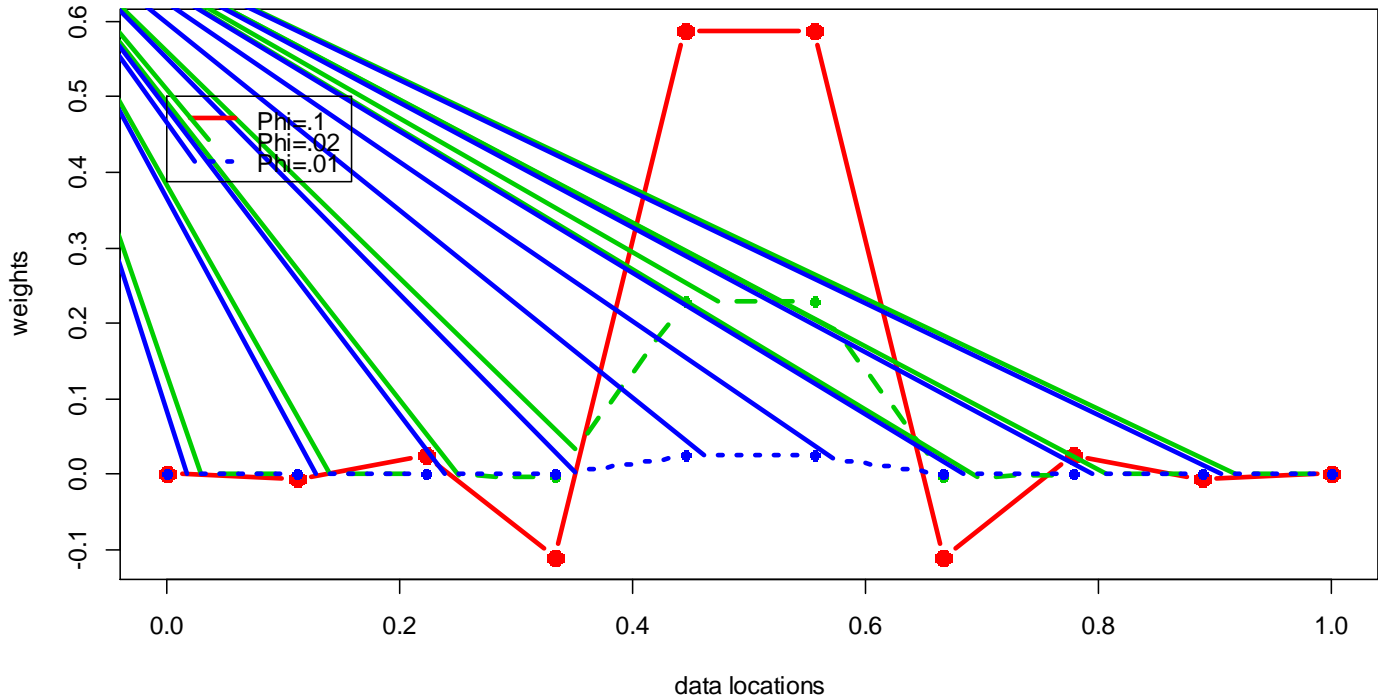
## 2) Effect of Location

- a. As the range/span increases, the weights spread their influence over a longer range (this corresponds to large values for  $\phi$  in the Matern function)
- b. As the range/span decreases, the weights of nearby observed data points decrease to zero (for simple kriging) or all become equal  $1/n$  for ordinary kriging.

*The plot below was created using simple kriging for **10 points on a line in one dimensional space**. Prediction is made at the point  $x_0 = 0.5$ . See R script for a simulation. In each case*

$$\mu = 0, \sigma^2 = 1, \tau^2 = 0, \phi = \text{varied}, \kappa = 1.5$$

**MBG 6.2 : Effect of Sill on Kriging Weights in One Dimension**



### 3) Effect of the Nugget

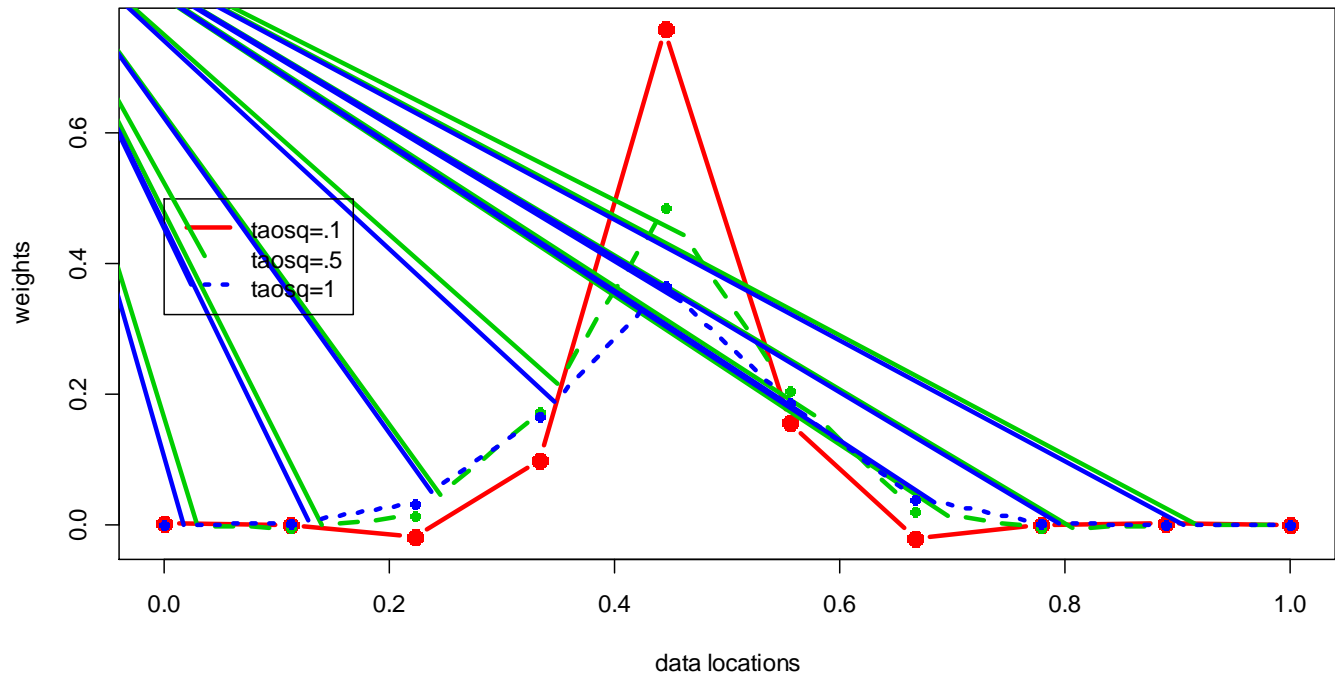
- a. As the nugget increases, all the weights have less effect (since the signal is masked by noise).

*The plots below were created using simple kriging for 10 points in one dimensional space. Prediction is made at the point  $x_0 = 0.45$ . See R script for a simulation. In each case*

$$\mu = 0, \sigma^2 = 1, \tau^2 = \text{varied}, \phi = 0.1, \kappa = 1.5$$



MBG 6.2 : Effect of Nugget on Kriging Weights in One Dimension



# Probability/Indicator Kriging

Suppose that we want to model the probability of an **event** (*like the probability that pH is > 7*). In this case we are modeling an **exceedance probability** :



$$\Pr(Y(\mathbf{x}_0) > y \mid Y_1, Y_2, \dots, Y_N)$$

We can create a kriging surface for this exceedance probability by 'kriging' the **indicator variable**

$I(Y(\mathbf{x}_0) > y)$  based on  $I(Y(\mathbf{x}_i) > y)$  (here  $I(Y(\mathbf{x}_i) > y) = 1$  if  $Y(\mathbf{x}_i) > y$ , 0 otherwise) :

$$\hat{I}_{Krig}(\mathbf{x}_0) = \sum_{i=1}^N a_i I(Y(\mathbf{x}_i) > y)$$

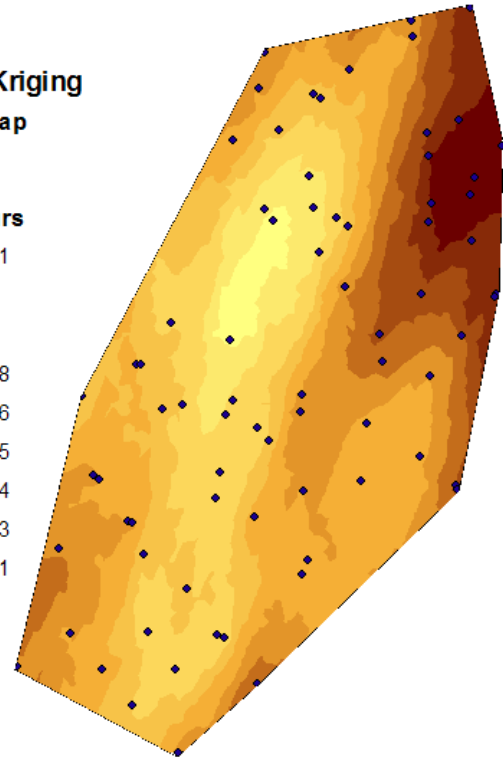
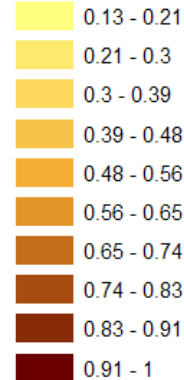
Proceed with ordinary kriging as before, but using the values of  $I(Yx(s_i) > y)$  rather than  $Y(x_i)$ .

*Results for pH Data from ArcGIS:*

**Legend**  
**Probability Kriging**  
**Probability Map**

Prob(pH>7)

**Filled Contours**



# Universal Kriging

Suppose that we believe that there is an **underlying trend in our surface** – say linear or quadratic (see last time). Universal kriging fits a model which **simultaneously removes a trend and krigs the desired surface**.

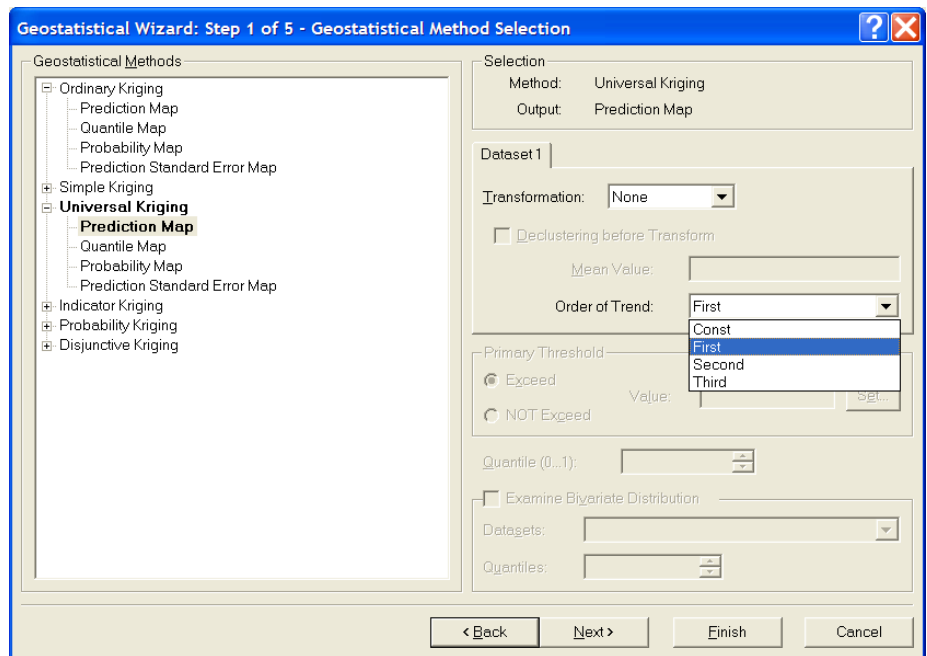


Without doing all the details, think about ordinary kriging.

- In ordinary kriging, we used **Lagrange multipliers** to add a constraint that the weights had to sum to 1:  $\sum_{i=1}^N a_i = 1$ .
- This constraint uses an expanded covariance matrix matrix  $\mathbf{C}_+$  to calculate the weights.

- In Universal Kriging, this matrix is further expanded to add **constraints which will remove linear or quadratic trends (or cubic or . ..)**
- See p. 196 of W&G for matrix details.

*In GIS, using the GeoStatistical Wizard, click on Universal Kriging, then choose **order of trend***



*Here are results for the pH data removing a linear trend : This is somewhat different than the results from simple kriging. Also notice that the errors are not necessarily smaller than those for simple kriging (legend is a bit whacky – 12-29???, given that max should be about .43)*

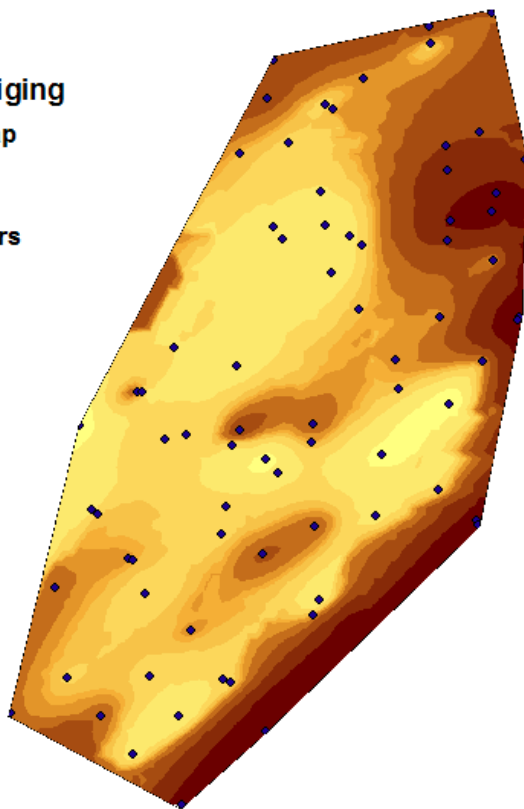
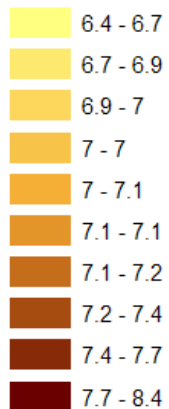
## Legend

### Universal Kriging

#### Prediction Map

pH

#### Filled Contours



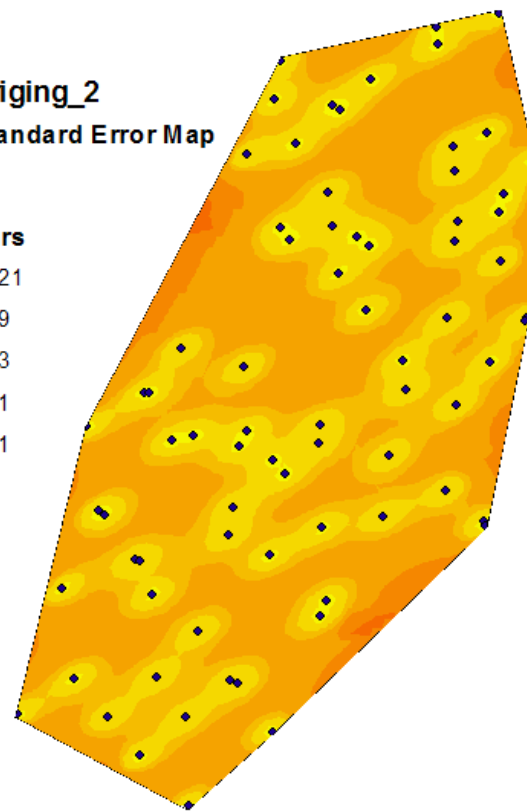
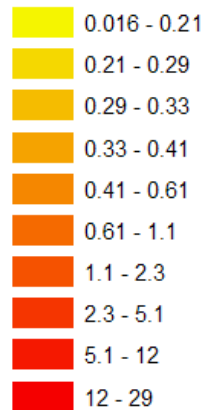
## Legend

### Universal Kriging\_2

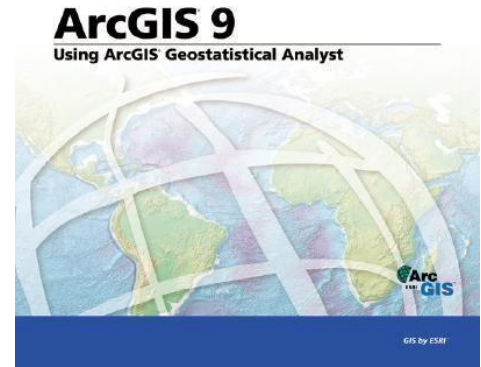
#### Prediction Standard Error Map

pH

#### Filled Contours



If you're using GIS to do variograms, kriging, etc., you should check out Using ArcGIS Geostatistical Analyst for ArcGIS 9. I haven't found an updated version for ArcGIS 10 – however, the details are very similar.



I've put this online as a pdf file

[http://reuningscherer.net/fes781/Resources/geostat\\_analyst.pdf](http://reuningscherer.net/fes781/Resources/geostat_analyst.pdf)

Here is a shorter tutorial (similar information)

<http://reuningscherer.net/fes781/Resources/geostatistical-analyst-tutorial.pdf>



## Kriging (Part 2)

**Kriging in R :** *A word of warning : R is like the wild west. Sometimes you strike it rich. Other times, you get shot in the gut.*



**Kriging in R.** There are two packages which will perform kriging in R. We've so far used the package `geoR` for variograms, and it also works quite well for kriging. However, it is also worth discussing the package `gstat` which has several options not available in `geoR`. Of course, this requires that we learn about variograms in `gstat` (since they're not compatible with `geoR`). We'll look at examples in both packages.

Two files we'll discuss today (both online) :

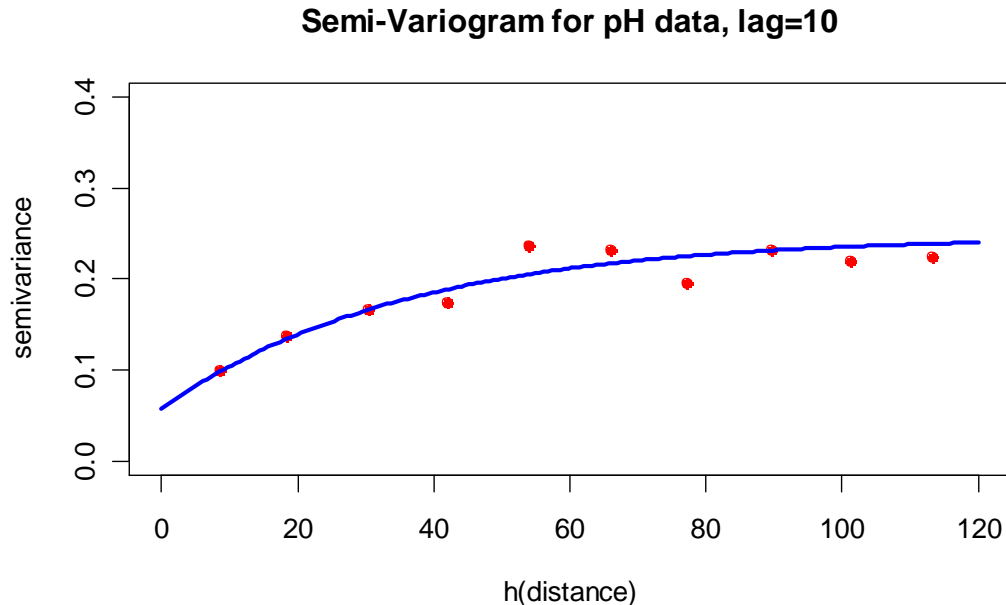
KrigingSmokyMountaingeoR.R.txt. GeoR is nice in that it allows use of maximum likelihood to fit a trend surface and a variogram model simultaneously. However, we'll see this can have mixed results in our example today . . .

<http://reuningscherer.net/fes781/RScripts/KrigingSmokyMountaingeoR.R.txt>

KrigingSmokyMountaingstat.R.txt – gstat has nice option for cross-validation (coming up), and also has nice graphics options.

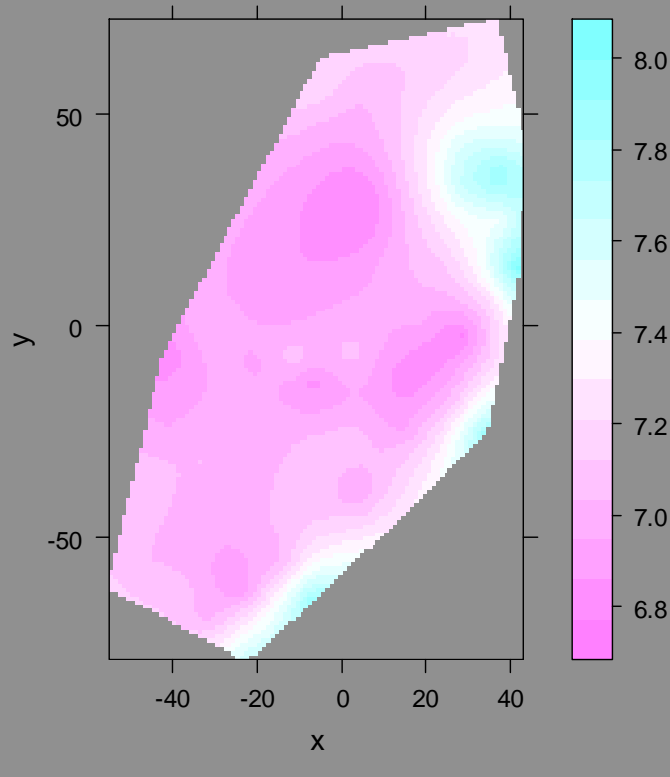
<http://reuningscherer.net/fes781/RScripts/KrigingSmokyMountaingstat.R.txt>

**Example : Smoky Mountain pH data in R using gstat.** This is a sample variogram with a lag of 10, fit with an isotropic exponential variogram model. Similar to what we've already seen . . .

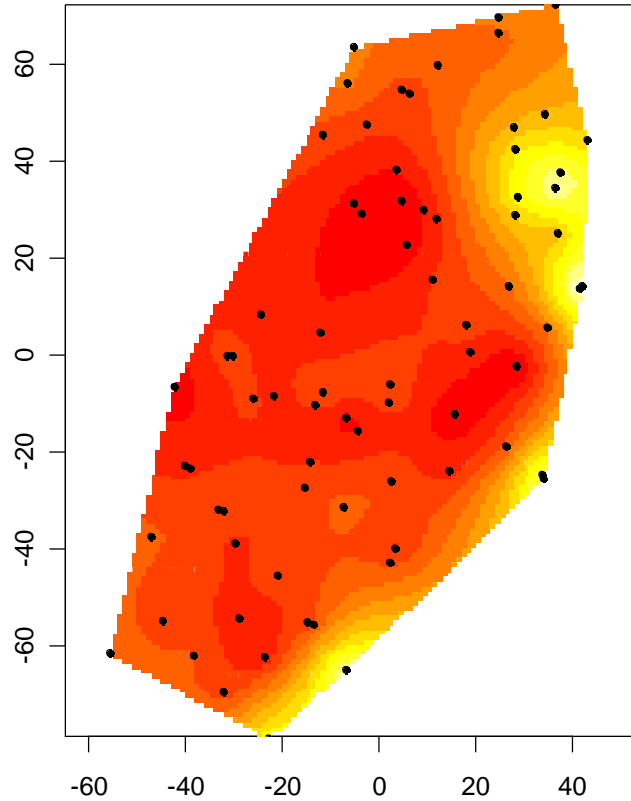


*Here are the results of ordinary kriging : again, we've pretty much seen this before. Shown here are two different graphing functions in R.*

Ordinary Kriging Predictions for pH Data

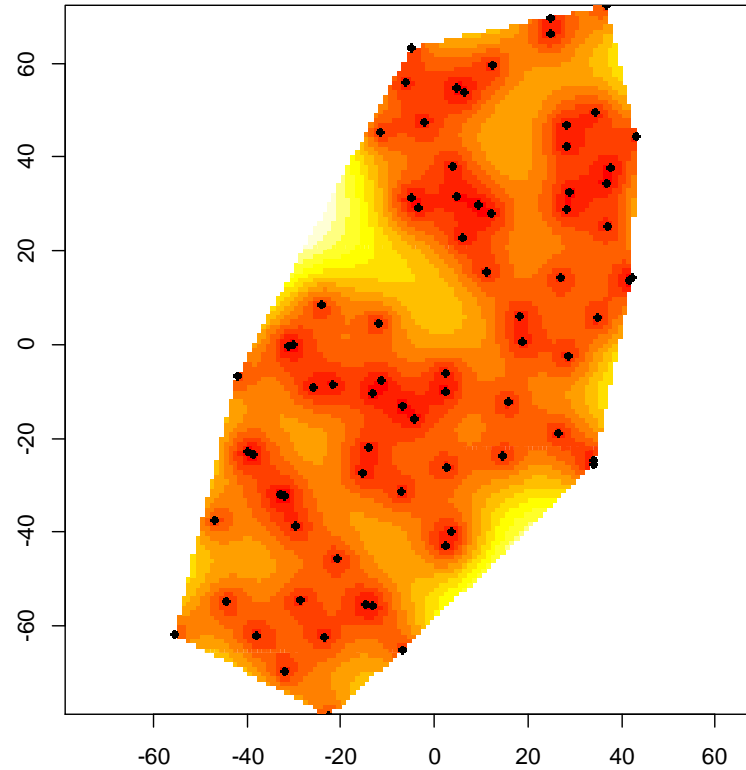


Ordinary Kriging Predictions for pH Data



*Here is an error map.*

**Ordinary Kriging Errors for pH Data, gstat**



*At this point, look at examples on Smoky Mountain pH data in geoR . . . .*

Why go to the trouble of learning R?!?! Well, there are some types of kriging available in R that are not available in ArcGIS. For example . . .

.....

**Suppose** we want to get kriged estimates not at particular points but over regions : i.e. we don't want to estimate  $Y()$  at a point; we want to get an average value of  $Y()$  over a region  $B$  (say a county or a census tract or a mining region or a stream). We could just krig over ever point in  $B$  and then take the average. Or we could use . .

# Block Kriging

(see B&G, Chapter 6 or W&G p. 311-312)



## An Example :

John W Kern, Kenneth O Coyle *Global block kriging to estimate biomass from acoustic surveys for zooplankton in the western Aleutian Islands*, Canadian Journal of Fisheries and Aquatic Sciences 57 pp 2112-2121

Suppose we have a stationary, constant (but unknown) mean process  $Y()$  measured at the points  $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_N$  (basically the same setup as ordinary kriging). We want to estimate  $Y()$  over a region  $B$  :

$$\hat{Y}_{Krig}(B) = \sum_{i=1}^N a_i Y(\mathbf{x}_i)$$

As in other types of kriging, we want  $\hat{Y}_{Krig}(\cdot)$

1) **Unbiased**

2) **Minimum Mean-Squared Prediction Error**

The math is similar to ordinary kriging : using a Lagrange multiplier, rearranging and taking partial derivatives, we get the **ordinary block kriging equations**:

$$\sum_{j=1}^N a_j C(\mathbf{x}_i - \mathbf{x}_j) + m = C(B - \mathbf{x}_i) \quad , \quad \text{for } i = 1, 2, \dots, N$$

$$\sum_{i=1}^N a_i = 1$$



In matrix form :

$$\mathbf{C}_+ \mathbf{a}_0 = \mathbf{c}_+ \mathbf{0}$$

$$\begin{bmatrix} C(\mathbf{x}_1 - \mathbf{x}_1) & \cdots & C(\mathbf{x}_1 - \mathbf{x}_N) & 1 \\ C(\mathbf{x}_2 - \mathbf{x}_1) & \cdots & C(\mathbf{x}_2 - \mathbf{x}_N) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ C(\mathbf{x}_N - \mathbf{x}_1) & \cdots & C(\mathbf{x}_N - \mathbf{x}_N) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \\ m \end{bmatrix} = \begin{bmatrix} C(B - \mathbf{x}_1) \\ C(B - \mathbf{x}_2) \\ \vdots \\ C(B - \mathbf{x}_N) \\ 1 \end{bmatrix}$$

$C(B - \mathbf{x}_i)$  is the **point-to-block covariogram** (basically an average value of covariogram from a point  $\mathbf{x}_i$  to every point in a block  $B$ ) :

$$C(B - \mathbf{x}) \equiv \frac{\int_B C(\mathbf{u} - \mathbf{x}) d\mathbf{u}}{|B|}$$

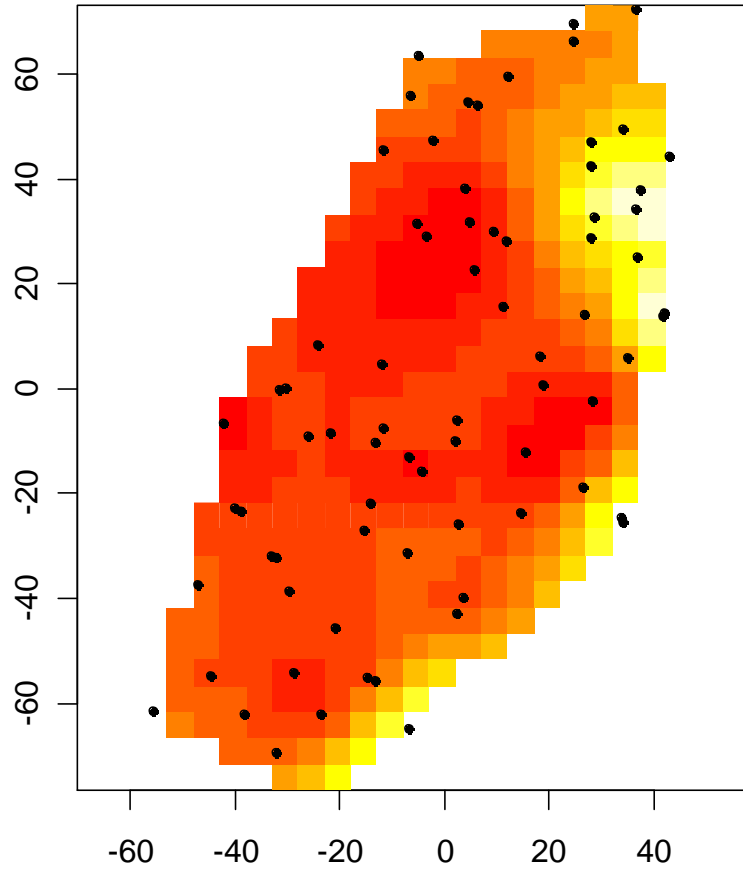
In practice,  $C(\cdot)$  is unknown, so it's estimated and modeled, and instead of calculating an integral,  $B$  is discretized into points :  $C(\cdot)$  is then averaged over these points.

By modifying the process above, one can also do **universal block kriging, indicator/probability block kriging, etc.**

**Example : pH data.** R Code is contained in the file <http://reuningscherer.net/fes781/RScripts/KrigingSmokyMountaingstat.R.txt>  
. Here are results for block of size 5x5.



## Ordinary Block Kriging Predictions for pH Data



## NOW :

One of the features of kriging is that it is an **exact interpolator**: that is, the kriged surface always fits the **data points** exactly (i.e. there are no errors at the actual observed locations).

This makes it difficult to evaluate how well our model worked (and to compare models). One way around this is to use

# Cross Validation

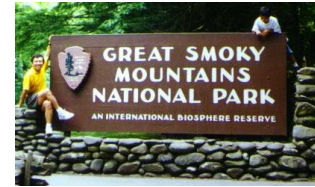


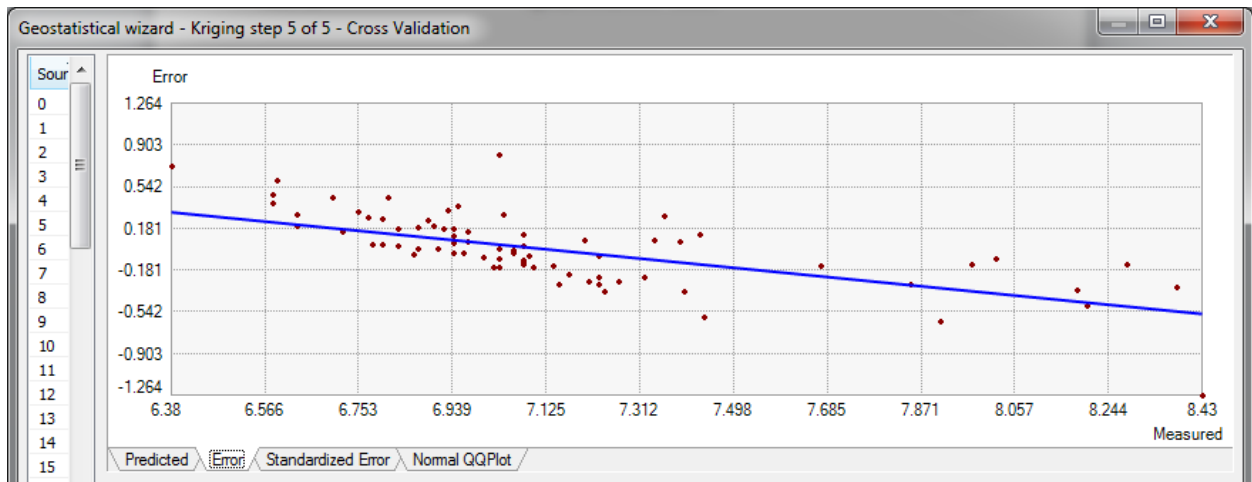
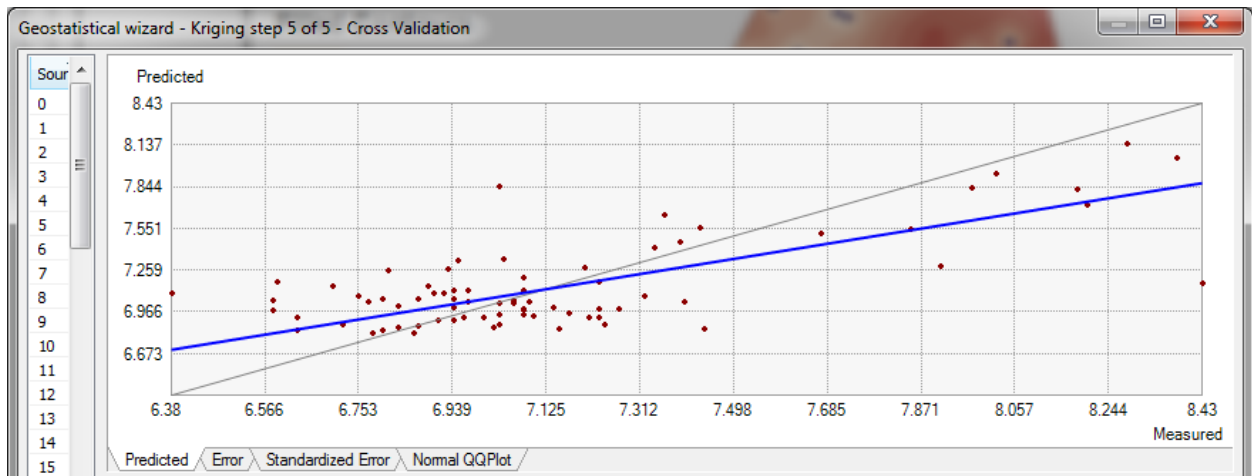
## Simplest form :

- 1) Leave one of the observed points  $\mathbf{x}_i$  out of the dataset  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ .
- 2) Use the rest of the data to perform kriging
- 3) Calculate  $\hat{Y}_{Krig}(\mathbf{x}_i)$  and calculate the **residual** :  
$$r_i = Y_{Krig}(\mathbf{x}_i) - \hat{Y}_{Krig}(\mathbf{x}_i)$$
- 4) Repeat the above for all points in the dataset

Cross validation as described above (leave one out) is **done automatically in ArcGIS** as the last step in the kriging process (i.e. last dialog box). It provides various summaries of the residuals to evaluate the model fit (check for normal residuals, plot observed vs. fitted, etc).

**Example : pH data in ArcGIS, ordinary kriging.** A plot of observed vs. fitted values shows that kriging tends to overestimate low values and underestimate high values (notice that trend line in blue has smaller slope than dotted line of unity if there were no trend). **This is a feature of processes based on averages** (it smooths out high points and fills in low points). This is more pronounced in the plot of Errors vs. Observed values (second tab). Remember “regression toward the mean . . . .”?





## n-fold Cross Validation

It is in fact possible to perform cross-validation by leaving out not just one point at a time but **a fraction of the data set** at a time. This is called **n-fold cross validation**. For example, five-fold cross validation leaves out one-fifth of the dataset at a time.

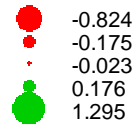
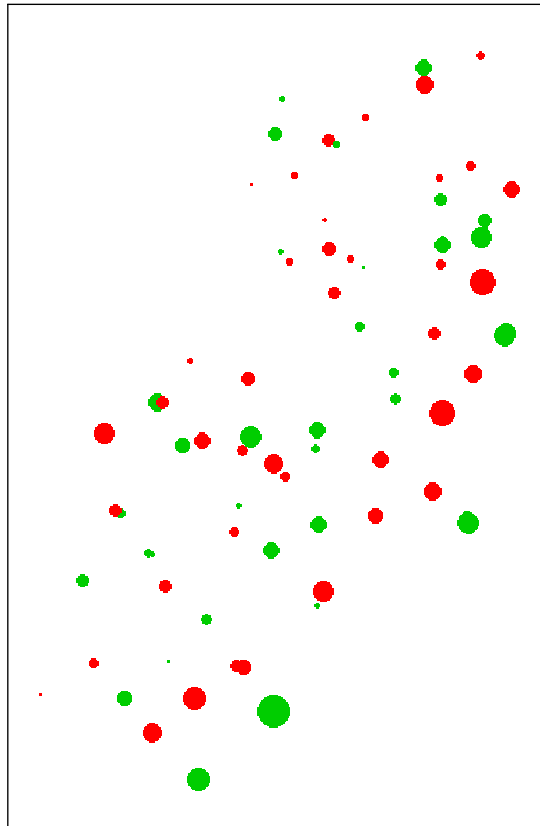


**Cross-Validation in R.** Use the `gstat` function `krige.cv()`. The file above provides the code that produced the plots below.

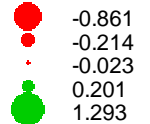
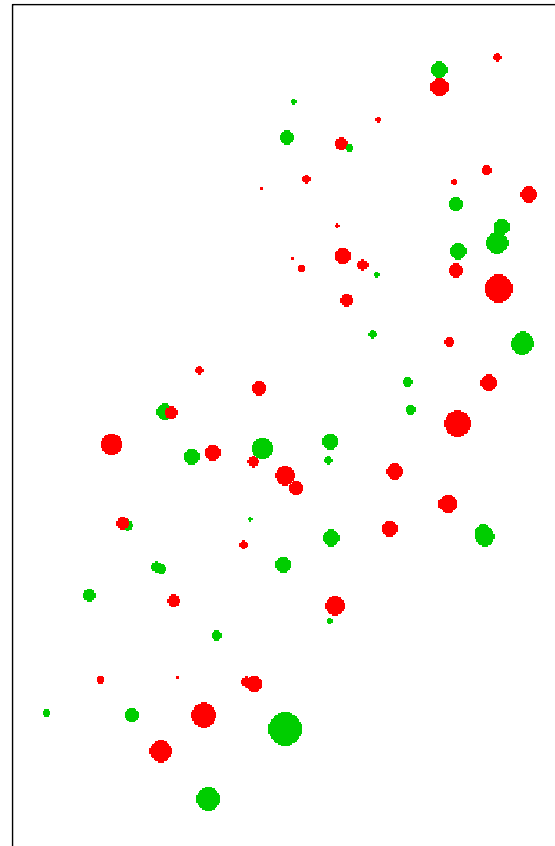
*Results are relatively similar for 5-fold cross validation and one-at-a-time cross validation.*



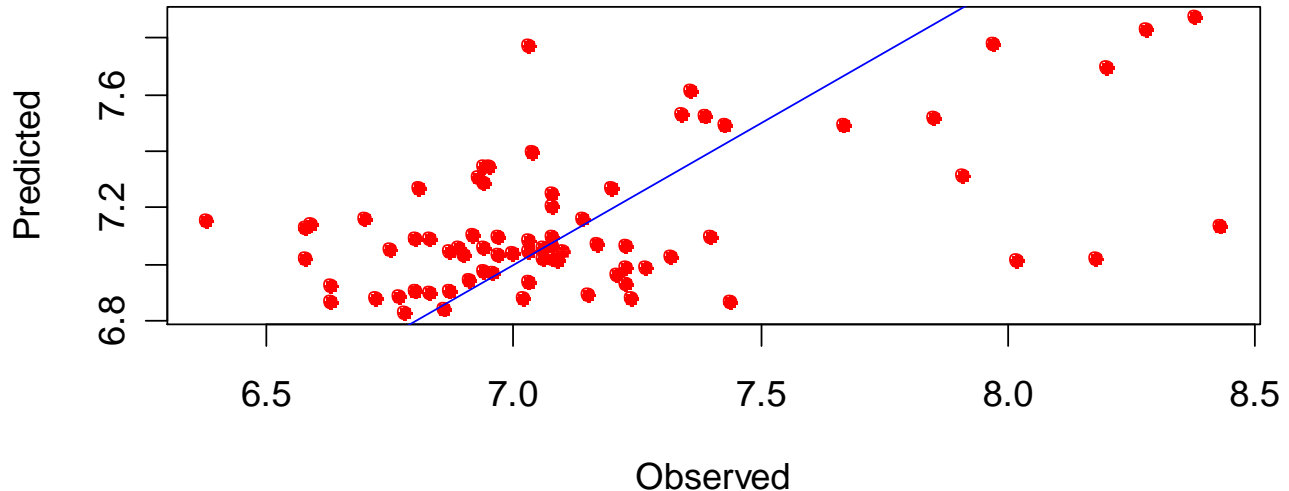
Smoky pH : Leave one out Residuals



Smoky pH : 5 fold CV Residuals



## Obs vs. Pred for 5-fold CV for pH

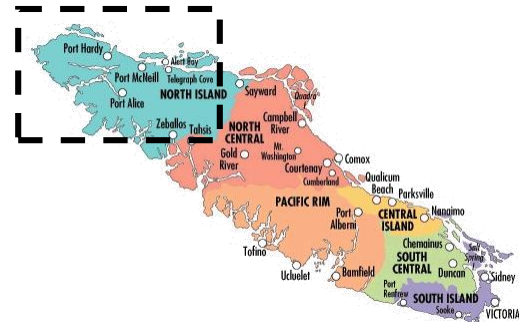


***Last comment – if you have ‘other data’ that you didn’t use for kriging, you can compare the observed data to the kriged data as an estimate of your errors.***

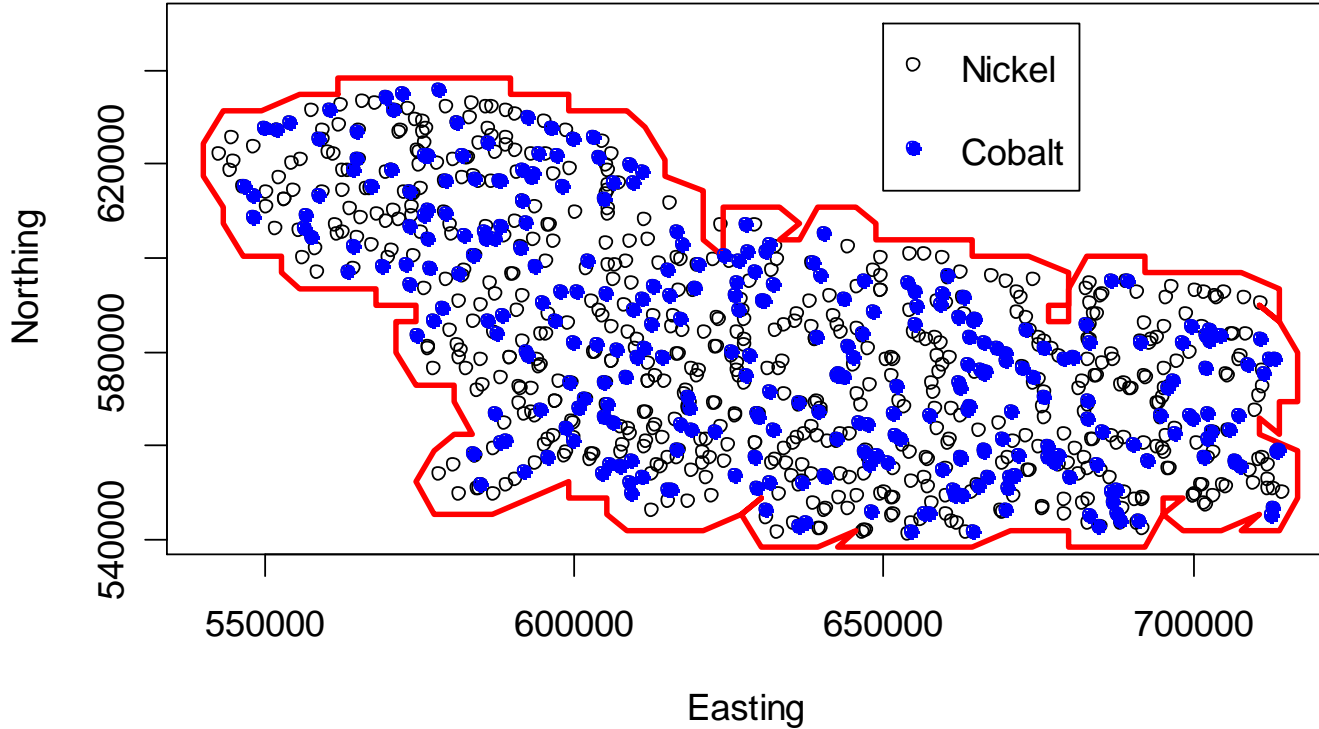
**NOW** : *Let's suppose that*

- At the points  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ , I have data on the process  $Y_1(\cdot)$  (say total biomass levels)
- At the points  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  I also have information on the process  $Y_2(\cdot)$  (say satellite data)
- I **also** have data on the process  $Y_2(\cdot)$  at **additional locations**, say  $\mathbf{x}_{N+1}, \mathbf{x}_{N+2}, \dots, \mathbf{x}_{N+M}$

**Example** (not Smoky Mountain) **Metal concentrations on Vancouver Island** (from B&G). This data is from the north-western part of Vancouver Island in BC, Canada. There are 900+ observations. For the sake of argument, let's suppose that we had all 900+ observations of  $\log(\text{nickel})$ , but only 300 observations of  $\log(\text{cobalt})$ .



# Vancouver Island Metals Data



- Suppose that the values of  $Y_1(\cdot)$  and  $Y_2(\cdot)$  are **related to each other** : that is, they are **correlated**.

***Example** : Vancouver Island has sample correlation of 0.73 at points where cobalt and nickel are both measured (log scale).*

- Maybe we could use the correlation of  $Y_1(\cdot)$  and  $Y_2(\cdot)$  and the fact that we have  $Y_2(\cdot)$  measured at more locations to improve my predictions of  $Y_1(\cdot)$  over the domain  $A$ ! This is the idea behind . . .

# Co-Kriging

- Exists in several forms : Simple, Ordinary, Universal, Block, etc.

- Widely used in environmental situations :

Zimmerman, D. AND D. M. Holland. COMPLEMENTARY CO-KRIGING: SPATIAL PREDICTION USING DATA COMBINED FROM, SEVERAL POLLUTION MONITORING NETWORKS. ENVIRONMETRICS 16(3):219-234, (2005).

Wu, Norvell, and Hopkins : Improved Prediction and Mapping of Soil Copper by Kriging with Auxiliary Data for Cation-Exchange Capacity <http://soil.scijournals.org/cgi/content/full/67/3/919>

Bolstad, Paul V.; Swank, Wayne; Vose, James. Predicting Southern Appalachian overstory vegetation with digital terrain data, Landscape Ecology. 13: 271-283.

- Before we can talk about cokriging, we have to first discuss the



# Cross-variogram



Let's suppose that the processes  $S_1(\cdot)$  and  $S_2(\cdot)$  are both **stationary** (constant means  $\mu_1$  and  $\mu_2$  and constant variances  $\sigma_1^2$  and  $\sigma_2^2$ ). We define the **cross variogram** as

$$2\gamma_{S_1S_2}(\mathbf{h}) = E[(S_1(\mathbf{x} + \mathbf{u}) - S_1(\mathbf{x}))(S_2(\mathbf{x} + \mathbf{u}) - S_2(\mathbf{x}))]$$

*(in words : on average, if I compare two points separated by a vector  $h$ , how do the values of  $Y_1(\cdot)$  change as compared to how the values of  $Y_2(\cdot)$  change?) This is analogous to the idea of the Covariance!*

As with variograms, we won't know this function directly, so we estimate it with

$$\hat{\gamma}_{S_1S_2}(\mathbf{u}) = \frac{1}{2n(\mathbf{u})} \sum_{|x_i - x_j| \in [u - \frac{1}{2}lag, u + \frac{1}{2}lag]} (Y_1(\mathbf{x}_i) - Y_1(\mathbf{x}_j))(Y_2(\mathbf{x}_i) - Y_2(\mathbf{x}_j))$$

Again, as with kriging, we fit a model to  $\hat{\gamma}_{s_1 s_2}$ . The same models discussed for variograms apply to cross-variograms (*exponential, spherical, etc*).

***SO : back to co-kriging . . . .***



# Ordinary Co-kriging

The basic idea of cokriging is to **incorporate information about  $Y_1(\cdot)$  into our predictions of  $Y_2(\cdot)$** : This means that our estimate is now a weighted average of both the values of  $Y_1(\cdot)$  and  $Y_2(\cdot)$ .

$$\hat{Y}_{1Krig}(\mathbf{x}_0) = \sum_{i=1}^N a_{Y_1 i} Y_1(\mathbf{x}_i) + \sum_{j=1}^{N+M} a_{Y_2 j} Y_2(\mathbf{x}_j)$$

As with other forms of kriging, we still want  $\hat{Y}_{1Krig}(\cdot)$

- 1) **Unbiased**
- 2) **Minimum Mean-Squared Prediction Error**

Remember that unbiased means that

$$E[\hat{Y}_{1Krig}(\mathbf{x}_0)] = E[Y_1(\mathbf{x}_0)] = \mu_1$$

There are several ways to ensure that  $\hat{Y}_{1Krig}(\ )$  is unbiased. However, a simple (and most common) way is to require that

$$\sum_{i=1}^N a_{Y_1i} = 1 \quad \text{and} \quad \sum_{j=1}^{N+M} a_{Y_2j} = 1$$

At this point, do lots of math similar to that for other kinds of kriging, have **two** Lagrange multipliers (for constraints above). Rearrange and solve for weights and constraints :

$$\begin{bmatrix} a_{Y_11} \\ \vdots \\ a_{Y_1N} \\ a_{Y_21} \\ \vdots \\ a_{Y_2N+M} \\ m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} \gamma_{Y_1}(\mathbf{x}_1 - \mathbf{x}_1) & \cdots & \gamma_{Y_1}(\mathbf{x}_1 - \mathbf{x}_N) & \gamma_{Y_1Y_2}(\mathbf{x}_1 - \mathbf{x}_1) & \cdots & \gamma_{Y_1Y_2}(\mathbf{x}_1 - \mathbf{x}_{N+M}) & 1 & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \gamma_{Y_1}(\mathbf{x}_N - \mathbf{x}_1) & \cdots & \gamma_{Y_1}(\mathbf{x}_N - \mathbf{x}_N) & \gamma_{Y_1Y_2}(\mathbf{x}_N - \mathbf{x}_1) & \cdots & \gamma_{Y_1Y_2}(\mathbf{x}_N - \mathbf{x}_{N+M}) & 1 & 0 \\ \gamma_{Y_1Y_2}(\mathbf{x}_1 - \mathbf{x}_1) & \cdots & \gamma_{Y_1Y_2}(\mathbf{x}_1 - \mathbf{x}_N) & \gamma_{Y_2}(\mathbf{x}_1 - \mathbf{x}_1) & \cdots & \gamma_{Y_2}(\mathbf{x}_1 - \mathbf{x}_{N+M}) & 0 & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \gamma_{Y_1Y_2}(\mathbf{x}_{N+M} - \mathbf{x}_1) & \cdots & \gamma_{Y_1Y_2}(\mathbf{x}_{N+M} - \mathbf{x}_N) & \gamma_{Y_2}(\mathbf{x}_{N+M} - \mathbf{x}_1) & \cdots & \gamma_{Y_2}(\mathbf{x}_{N+M} - \mathbf{x}_{N+M}) & 0 & 1 \\ m_1 & 1 & \cdots & 1 & \cdots & 0 & 0 & 0 \\ m_2 & 0 & \cdots & 0 & \cdots & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \gamma_{Y_1}(\mathbf{x}_0 - \mathbf{x}_1) \\ \vdots \\ \gamma_{Y_1}(\mathbf{x}_0 - \mathbf{x}_N) \\ \gamma_{Y_1Y_2}(\mathbf{x}_0 - \mathbf{x}_1) \\ \vdots \\ \gamma_{Y_1Y_2}(\mathbf{x}_0 - \mathbf{x}_{N+M}) \\ 1 \\ 0 \end{bmatrix}$$

**Finally :** Rearrange and solve for weights and constraints:

or more concisely :

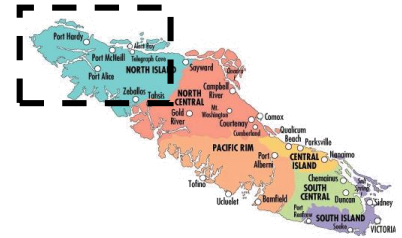
$$\mathbf{a}_0 = \mathbf{C}_+^{-1} \mathbf{c}_+$$

*The + 's indicate that these matrices are 'expanded' by rows/columns to account for the unbiasedness constraints and additional variables*

Note that there are **THREE distinct variograms**:

- $\gamma_{Y_1}(\cdot)$  is the variogram for the  $Y_1(\cdot)$  process (*Cobalt*)
- $\gamma_{Y_2}(\cdot)$  is the variogram for the  $Y_2(\cdot)$  process (*Nickel*)
- $\gamma_{Y_1Y_2}(\cdot)$  is the cross-variogram for relating the  $Y_2(\cdot)$  and  $Y_1(\cdot)$  processes (*how to Nickel and Cobalt co-vary*)

**Example : Vancouver Island.** We use cokriging to try to estimate  $\log(\text{Cobalt})$  levels using  $\log(\text{Nickel})$  values (example in class using ArcGIS).



A few things to note :

- You need TWO datasets – one with nickel values, another with cobalt values
- If you have the same number of observations in the two datasets, co-kriging **will not help**. Just use regular kriging in that case.

## Step 1 : Remove Trends (if any)

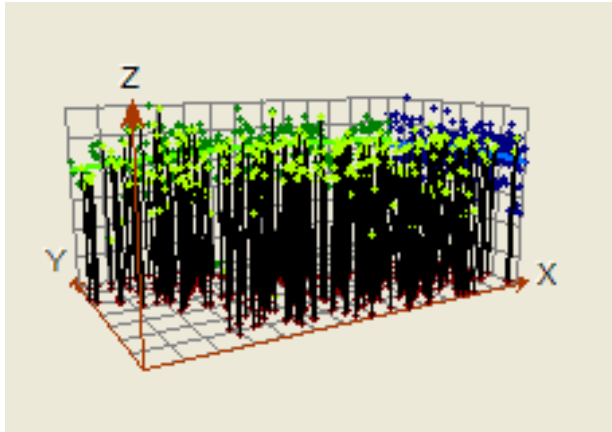
Are there any linear / quadratic / other trends in cobalt presence? How about for nickel? Let's make a few pictures . . .

In ArcGIS, there is a TREND ANALYSIS tool in the Geostatistical Analyst

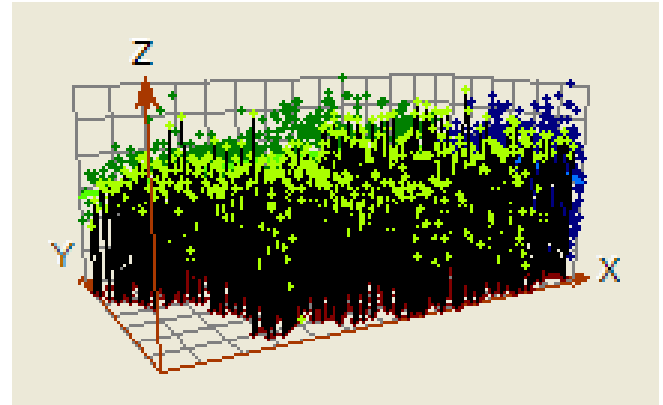
- Click on the Cobalt point layer
- Choose Geostat Analyst → Explore Data → Trend Analysis
- Choose the correct attribute at the bottom of the window

*Here are pictures for cobalt and nickel : I didn't see any trends in Cobalt, perhaps a linear trend in Nickel.*

## *Cobalt*

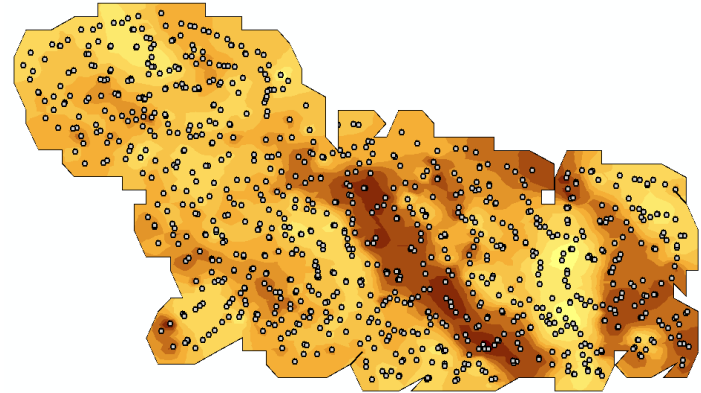
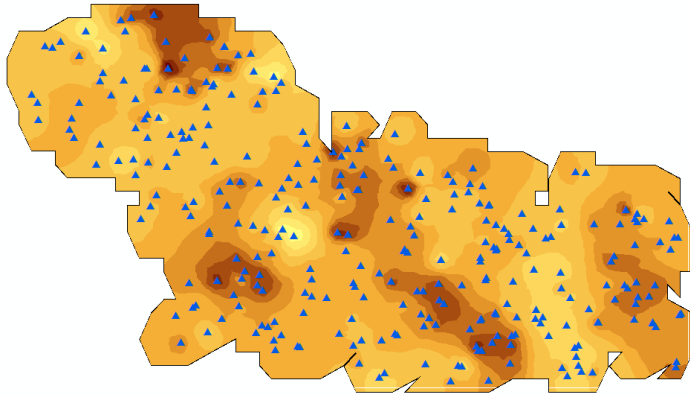


## *Nickel*



It's a bit difficult to tell. **SO** : let's try **Inverse Distance Weighting** (basically smoothing which ignores covariance)

Here are the results for cobalt and nickel



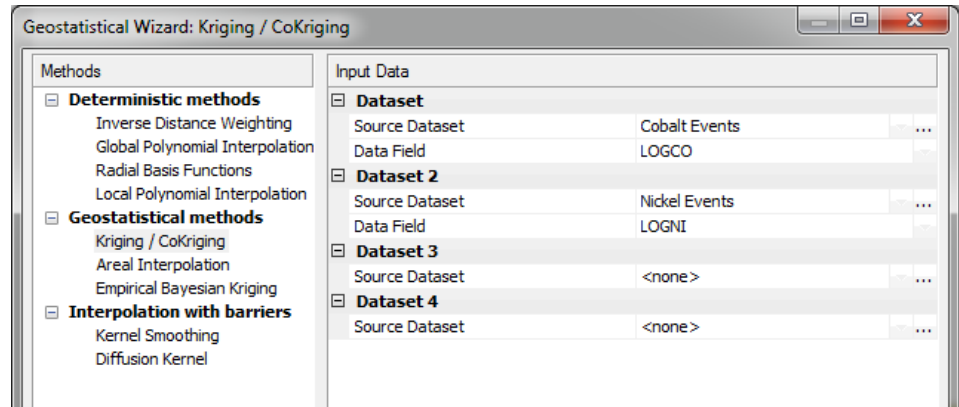
There is perhaps a bit of an uphill trend in the nickel model from right to left, so let's fit a linear trend to nickel.



## Step 2 : Perform Co-kriging

To do cokriging in ArcGIS, use the GeoStatistical Analysts, choose Wizard, and then

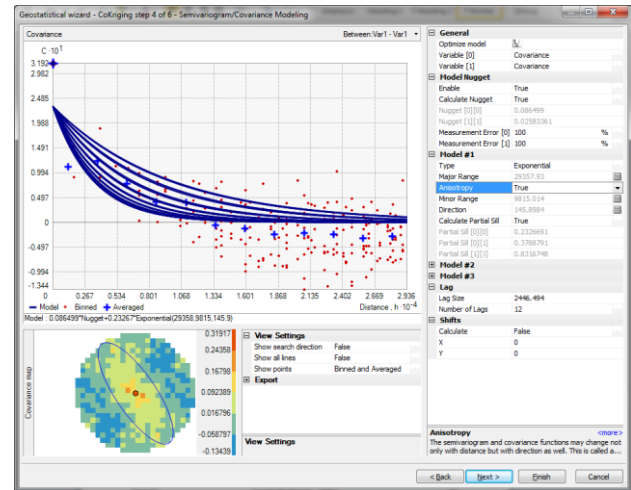
choose co-kriging. You'll need to specify **both datasets**. Note that the **FIRST** dataset should be the smaller dataset that you are trying to cokrig.



The next screen allows you to remove trends and choose the type of co-kriging you'd like (ordinary, simple, universal, probability, etc). We start with ordinary, and remove a linear trend from nickel and no trend from cobalt.

Next, you choose the model for the semivariograms (there are three here, remember, but you can only choose one FAMILY of variograms to use in GIS).

I found suggestions of anisotropy, so I fit an anisotropic model as well for the first variogram.

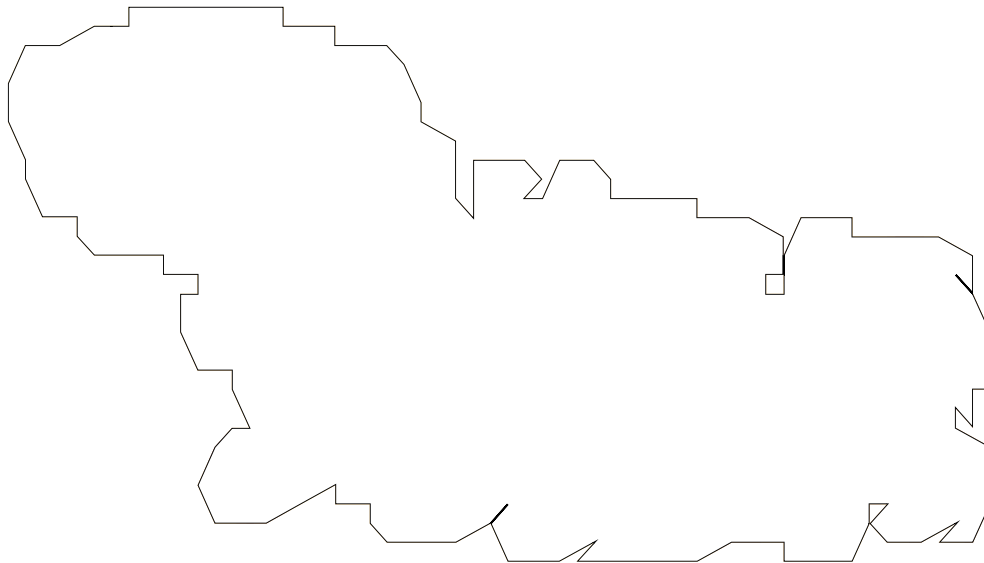


Finally, results are given as far as predicted/observed values using cross-classification (errors are positive for low Cobalt values and negative for large Cobalt values, no surprise). ALSO, we get the fitted parameters for our exponential variogram models :

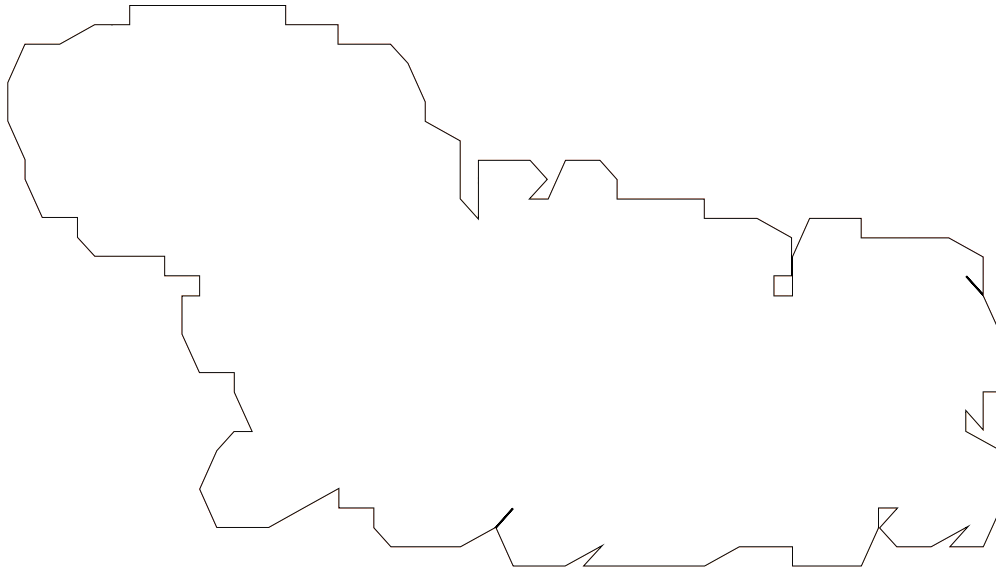
*Note : You may need to extend the size of the layer used for cokriging – right-click on the kriged layer, choose Properties, choose Extant, and extend as necessary.*

*You may also want to trim the kriged surface to your domain polygon – right click on the top LAYER bar, choose properties, choose data-frame, click on Enable Clip to Shape.*

*Here are the results for ordinary co-kriging for cobalt :*



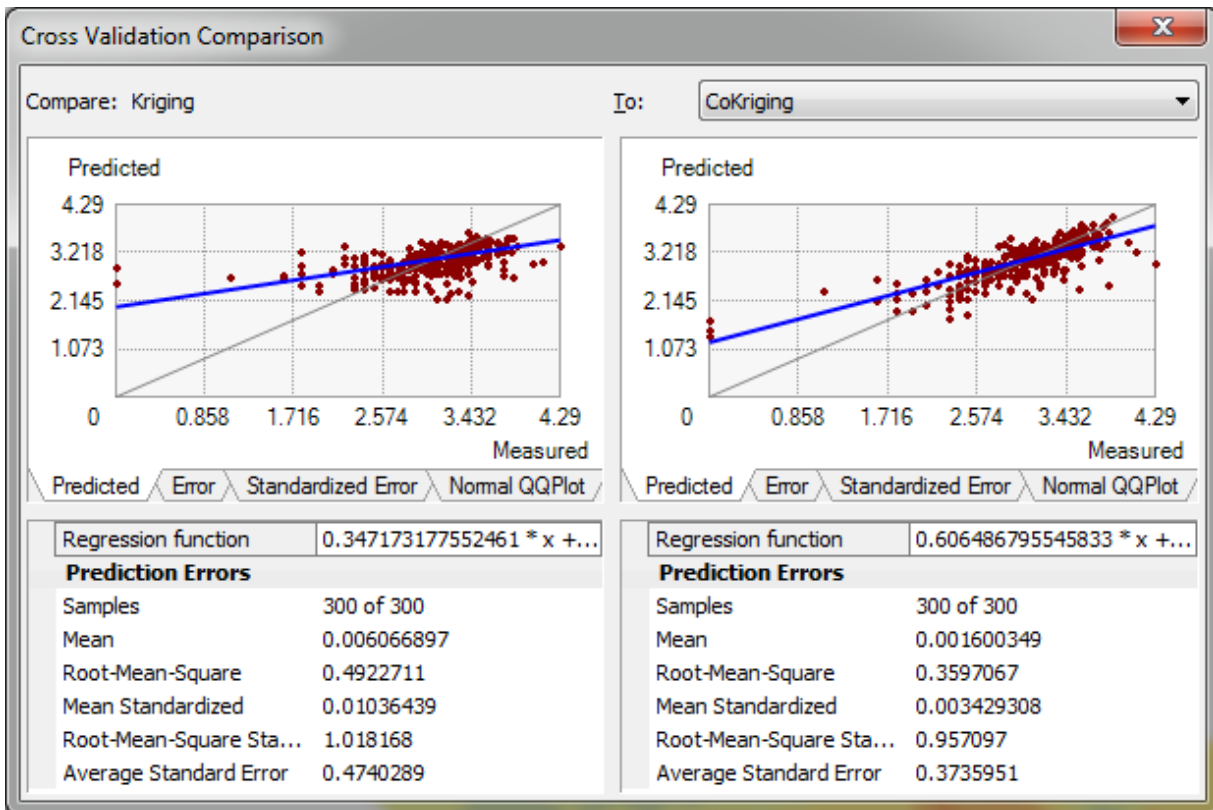
*Here are the errors :*



### **Step 3 : Compare to Ordinary Kriging**

***NOW*** : as a comparison, lets try ordinary kriging, ignoring the nickel data! The output is shown in class . . .

*After fitting these two models, you can right click on either prediction layer and choose 'compare models'. Here are the results :*



*The average residual error is lower for cokriging for each statistic calculated (by the way, the sample standard deviation for the cobalt data is about 0.64, vs. about 0.45-0.47 for the kriged surface).*

## Other kinds of Co-kriging

- **Universal Co-kriging** : Expand the  $C_+$  matrix to include constraints to remove linear, quadratic trends
- **Indicator Co-kriging** : use indicator variables
- **Block Co-kriging** : average over Blocks

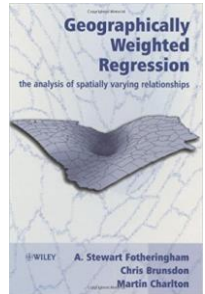
## Other Comments

Note that you could easily add other variables for additional information to help your cokriging (i.e. just expand the matrix to include more covariograms and cross variograms between other variables). GIS allows for up to three additional datasets.

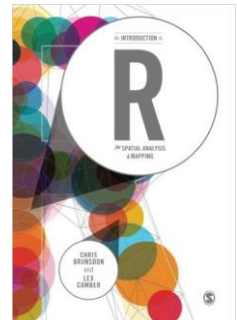
# Geographically Weighted Regression (GWR)

## Resources :

*Geographically Weighted Regression: The Analysis of Spatially Varying Relationships* by A. Stewart Fotheringham et al.



*An Introduction to R for Spatial Analysis and Mapping* by Chris Brunsdon



## R packages

gwr  
spgwr  
GWmodel



**Usual multiple regression model in scalar notation** : if you have  $p$  predictors, the **linear model** for the value of  $Y_i$  for a single observation  $i$  is given by

$$Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \varepsilon_i, \quad i = 1, 2, \dots, n$$
$$\varepsilon_i \sim N(0, \sigma^2) \text{ and } \text{Var}(\boldsymbol{\varepsilon}) = \Sigma$$

- The  $\beta$ 's are global parameters : whatever their value, they apply everywhere in the spatial region being modeled
- $\Sigma$  models the spatial correlation and possibly heteroscedasticity.
- This is true regardless whether the  $\beta$ 's are covariates or just locations as in a trend surface model.

**Idea of GWR** : it may be that the  $\beta$ 's are **NOT constant** over the entire region

They may exhibit a drift or trend in value depending on spatial location

Random coefficients (random effects) in mixed-effects models is another way to let their value vary randomly.

## A bit of Spatial Regression History : the Expansion Method (EM)

Consider the usual simple linear regression model :

$$Y_i = \beta_o + \beta_1 X_{i1} + \varepsilon_i$$

We could let the  $\beta'$  s also be a linear function of the coordinate pair  $(u_i, v_i)$  (say, easting, northing) :

$$\begin{aligned}\beta_o &= \beta_{00} + \beta_{01}u_i + \beta_{02}v_i \\ \beta_1 &= \beta_{10} + \beta_{11}u_i + \beta_{12}v_i\end{aligned}$$

Substituting, we get

$$Y_i = \underbrace{(\beta_{00} + \beta_{01}u_i + \beta_{02}v_i)}_{\text{Intercept}} + \underbrace{(\beta_{10} + \beta_{11}u_i + \beta_{12}v_i)}_{\text{Slope}} X_{i1} + \varepsilon_i$$

- We can fit this using ordinary least squares (OLS)
- Also, the usual regression tools (significance, residual plots, qqplots, etc) all apply.
- Notice this is somewhat different that Trend Surface Analysis (TSA) in that the relationship between the covariate  $X$  and the response  $Y$  can vary over space.
- EM was developed in the early 1970's as a first attempt to deal with spatial autocorrelation (the hope was that this would 'take care' of spatial autocorrelation in the mean part of the model.

# Slight Detour : Locally Weighted Scatterplot Smoothing (LOWESS) and Local Polynomial Regression Fitting (LOESS)

<https://www.youtube.com/watch?v=ncF7ArjJFqM>

<https://www.youtube.com/watch?v=Zn3jw0-CfBc>

[https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug\\_transreg\\_sect016.htm](https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_transreg_sect016.htm)

*(quick picture here)*

*(quick R-code example)*

```
x=c(1:200)
true_y=10*sin(x/25)+x/10
y=true_y+rnorm(200,mean=0, sd=4)

for (i in 1:20){
  plot(x,true_y, type='l', lwd=3,col='red', lty=2)
  points(x,y,pch=19)
  loess1=loess(y~x, span=i/20)
  lines(loess1$x,loess1$fitted, col='green',lwd=3)
  Sys.sleep(1)
}
```

**GWR (developed in late 1990's) : it's like spatial LOESS!**

**GWR Model :**

The  $\beta$ 's become **functions of location**  $(u_i, v_i)$

$$Y_i = \beta_0(u_i, v_i) + \beta_1(u_i, v_i)X_{i1} + \dots + \beta_p(u_i, v_i)X_{ip} + \varepsilon_i$$

Now, we can't actually estimate  $n * (n - 1) * p$  different  $\beta$ 's : instead, perform what is essentially weighted regression, but the weights are spatially determined by some function of distance.

The **ordinary least-squares** estimate of our coefficients  $\beta$  is given by (<https://sites.harvard.edu/fs/docs/icb.topic515975.files/OLSDerivation.pdf> )

$$\tilde{\beta} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{Y}$$

The **generalized least-squares** estimate of our coefficients  $\beta$  is given by ([https://en.wikipedia.org/wiki/Generalized\\_least\\_squares](https://en.wikipedia.org/wiki/Generalized_least_squares) ) (*think Mahalanobis Distance*)

$$\hat{\beta} = (\mathbf{X}'\Sigma^{-1}\mathbf{X})^{-1} \mathbf{X}'\Sigma^{-1}\mathbf{Y}$$

The **geographically weighted least-squares** estimate of our coefficients  $\boldsymbol{\beta}(u_i, v_i)$  is given by

$$\hat{\boldsymbol{\beta}}(u_i, v_i) = \hat{\boldsymbol{\beta}}(i) = (\mathbf{X}'\mathbf{W}(u_i, v_i)\mathbf{X})^{-1} \mathbf{X}'\mathbf{W}(u_i, v_i)\mathbf{Y}$$

Where

$$\mathbf{W}(u_i, v_i) = \mathbf{W}(i) = \begin{pmatrix} w_{i1} & 0 & \cdots & 0 \\ 0 & w_{i2} & \cdots & 0 \\ & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_{in} \end{pmatrix}$$

And  $w_{ij}$  is determined by the distance from the  $j$ th observation to the  $i$ th observation, and  $w_{ij} = 1$ .



In general, each observation  $i$  will have a different weight matrix  $\mathbf{W}(i)$ : hence, each location will have different values for  $\hat{\beta}(u_i, v_i)$

$\hat{\beta}(u_i, v_i)$  estimates for observations close to the  $i$ th observation will be similar in magnitude, but we should get a spatially yet smoothly varying surface of coefficient estimates. The hope is that there will be no residual spatial autocorrelation to deal with.

**SO** : how to determine the weights  $W(i)$  ? **Think Kernel Smoothing!**

## Common Kernels

Exponential weighting kernel:  $w_{ij} = \exp(-d_{ij}/b)$

where  $b$  is the bandwidth and  $d_{ij}$  is the Euclidean distance separating the  $i$ th from the  $j$ th observation.

Gaussian weighting kernel:  $w_{ij} = \exp(-0.5(d_{ij}/b)^2)$

Evidently, the weights attenuate more quickly with Gaussian kernel compared to exponential kernel.

The bi-square, accessed via **gwr.bisquare()**. The bi-square weighting kernel:

$$w_{ij} = \begin{cases} (1 - (d_{ij}/b)^2)^2 & \text{when } d_{ij} < b \\ 0 & \text{else} \end{cases}$$

The tri-cube, accessed via **gwr.tricube()**. The tri-cube weighting kernel:

$$w_{ij} = \begin{cases} (1 - (d_{ij}/b)^3)^3 & \text{when } d_{ij} < b \\ 0 & \text{else} \end{cases}$$

It is also possible to use spatially varying kernels where the bandwidth changes depending on the density of local observations. See `gw.adapt()` in the `spgwr` package.

## NOW – how to determine the ‘optimal’ bandwidth.

All three R packages have a function to determine optimal bandwidth, where the optimum bandwidth  $b$  minimizes the cross-validation sum of squares :

$$CV = \sum_{i=1}^n (y_i - \hat{y}_{-i}(b))^2$$

**Where**  $\hat{y}_{-i}(b)$  is the fitted value of  $y_i$  with the observations for point  $i$  omitted during the calibration process.

In `spgwr`, the optimal bandwidth function is the `gwr.sel( )`. With kernel options of `gwr.Gauss` or `gwr.bisquare` only. The kernel is specified with its `gweight` argument of the `gwr.sel( )` function.

In `gwrr` it is the `gwr.bw.est( )` function with the exponential or gauss kernel options.

In `GWmodel` it is the `bw.gwr( )` function with five kernel options: gaussian, exponential, bisquare, tricube, and boxcar.

## EXAMPLE of GWR : Meuse River Data

[http://reuningscherer.net/fes781/Rscripts/GWR\\_meuse.txt](http://reuningscherer.net/fes781/Rscripts/GWR_meuse.txt)