Spatial Interpolation

Methods for extending information on spatial data fields

John Lewis

Spatial Interpolation

- Data Integration
  - merging spatially incompatible data sets
    - » household survey data with census tract data
- Predicting Values for Locations with no Observations
  - use information on (spatial) pattern at one set of locations/scale to predict values at different locations/scale
- Interpolation and Extrapolation
  - extrapolation: outside range observed values

The type of Spatial Interpolation depends on:

- what type of surface model you want to produce
- what data are available
  - point, line or area
  - continuous or discrete

Interpolation techniques are classed as:

- *Global* - surface model considers all the points
- *Local* - uses only values in a prescribed "nearby" neighborhood

---

- *Exact* - retains the original values of the sample on the resulting surface
- *Inexact* - assigns new values to known data points

- *Deteministic* – provides no indication of the extent of possible error
- *Stochastic* – methods provide probabilistic estimate

Interpolating Spatial Objects

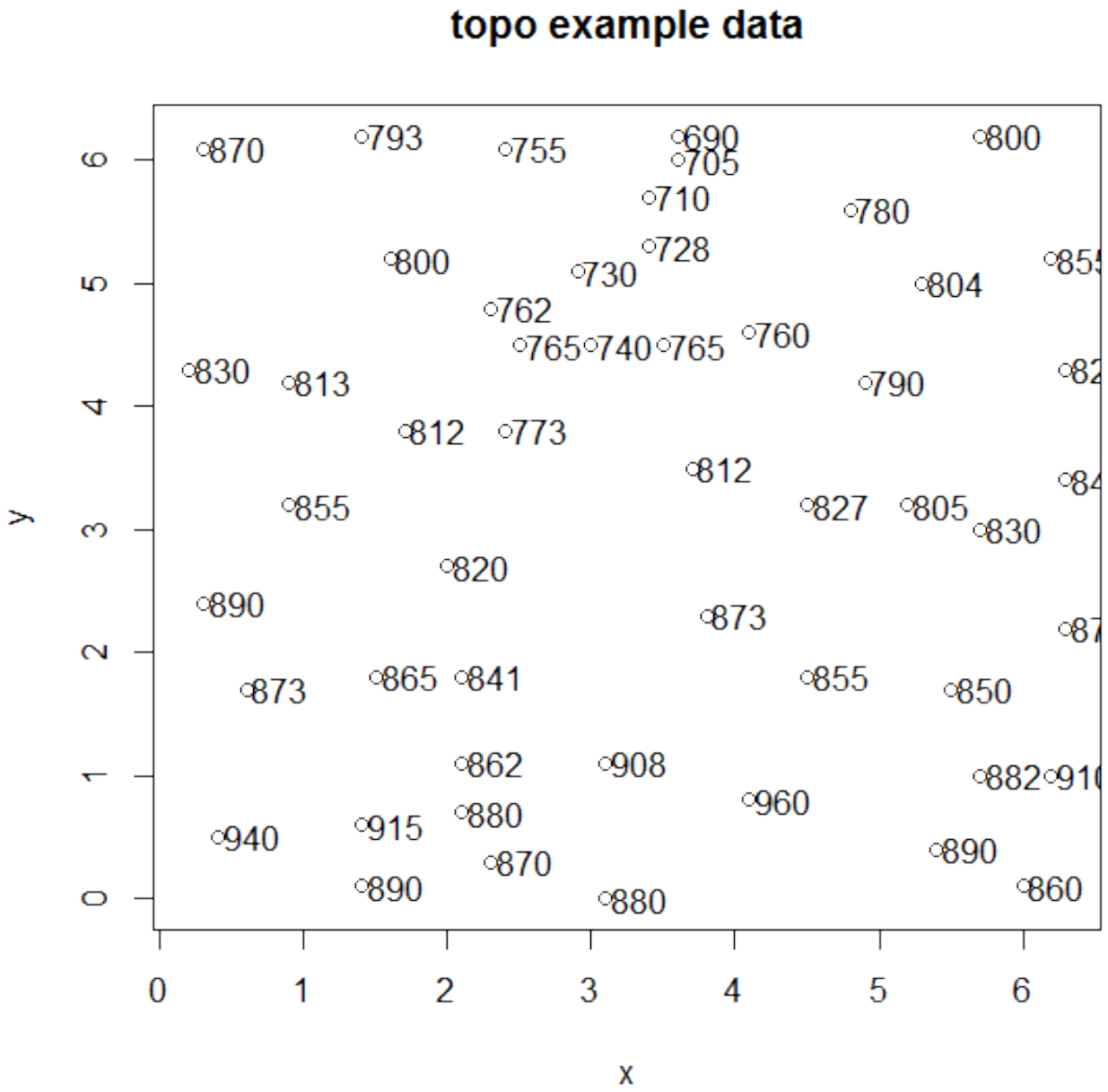
- Point to Point
  - from observed points to unobserved points
    - » e.g. monitoring stations to house locations
- Point to Area
  - from observed points to areal units
    - » e.g. from monitoring stations to census tracts
- Area to Area
  - from one set of observed areas to a different (overlapping) set of areas = areal interpolation
    - » e.g. from counties to water management zones

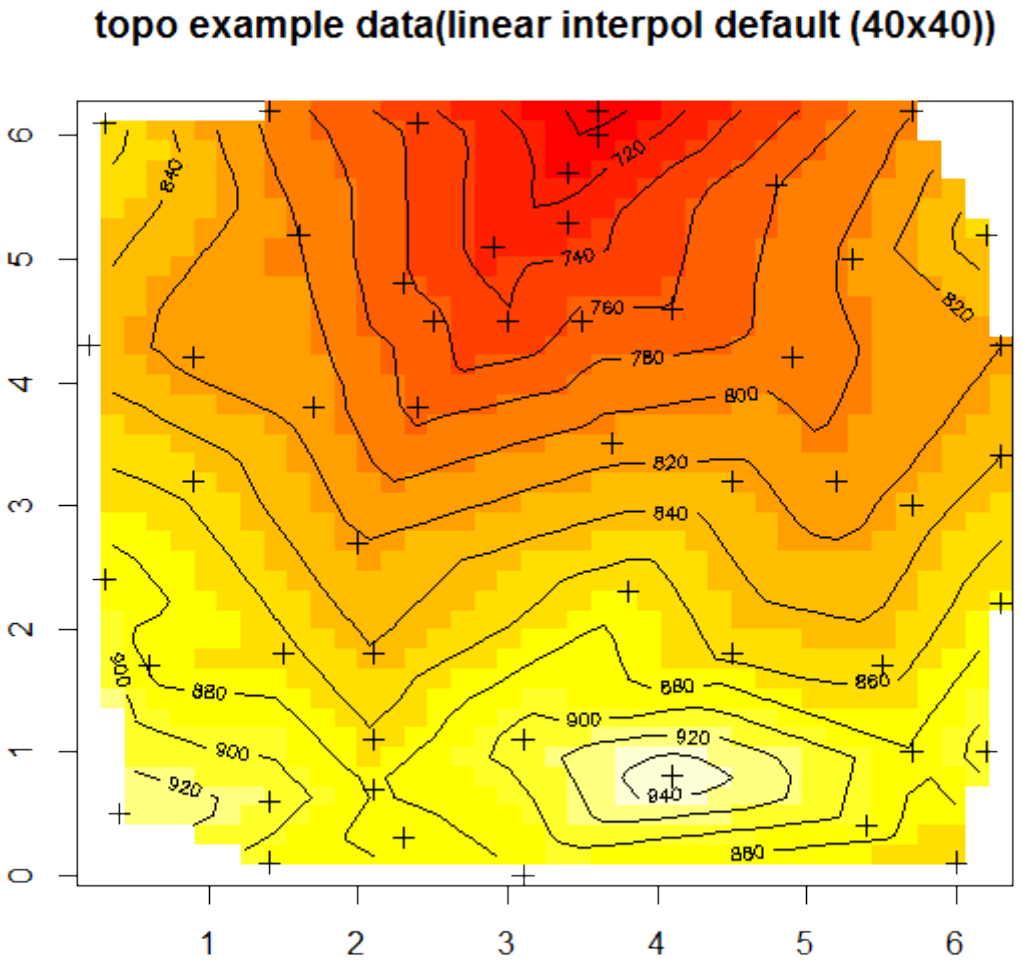
General list of techniques (points)

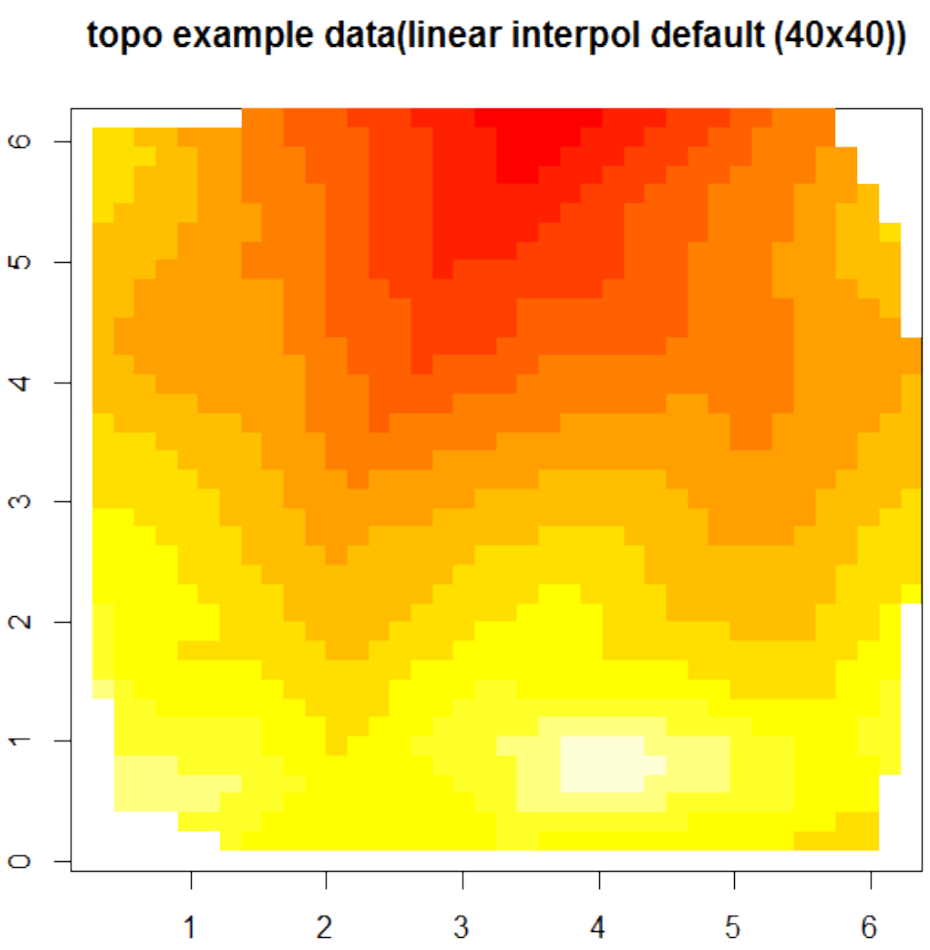
- *trend surface* - a global interpolator which calculates a best overall surface fit
- *IDW* - inverse distance weighting
- *loess* – local weighted smoothing regression (weighted quadratic least squares regression)
- *spline* - mathematical piecewise curve fitting
- *Kriging* - using geostatistical techniques

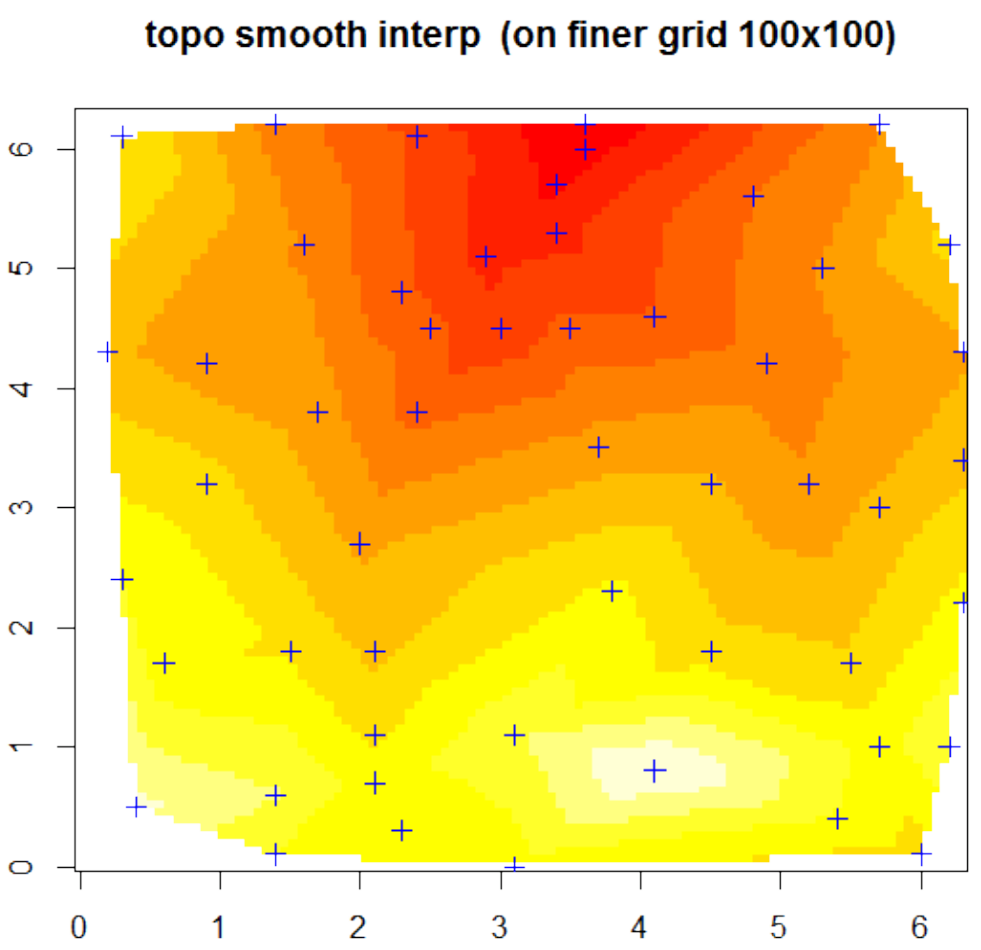
Other procedures

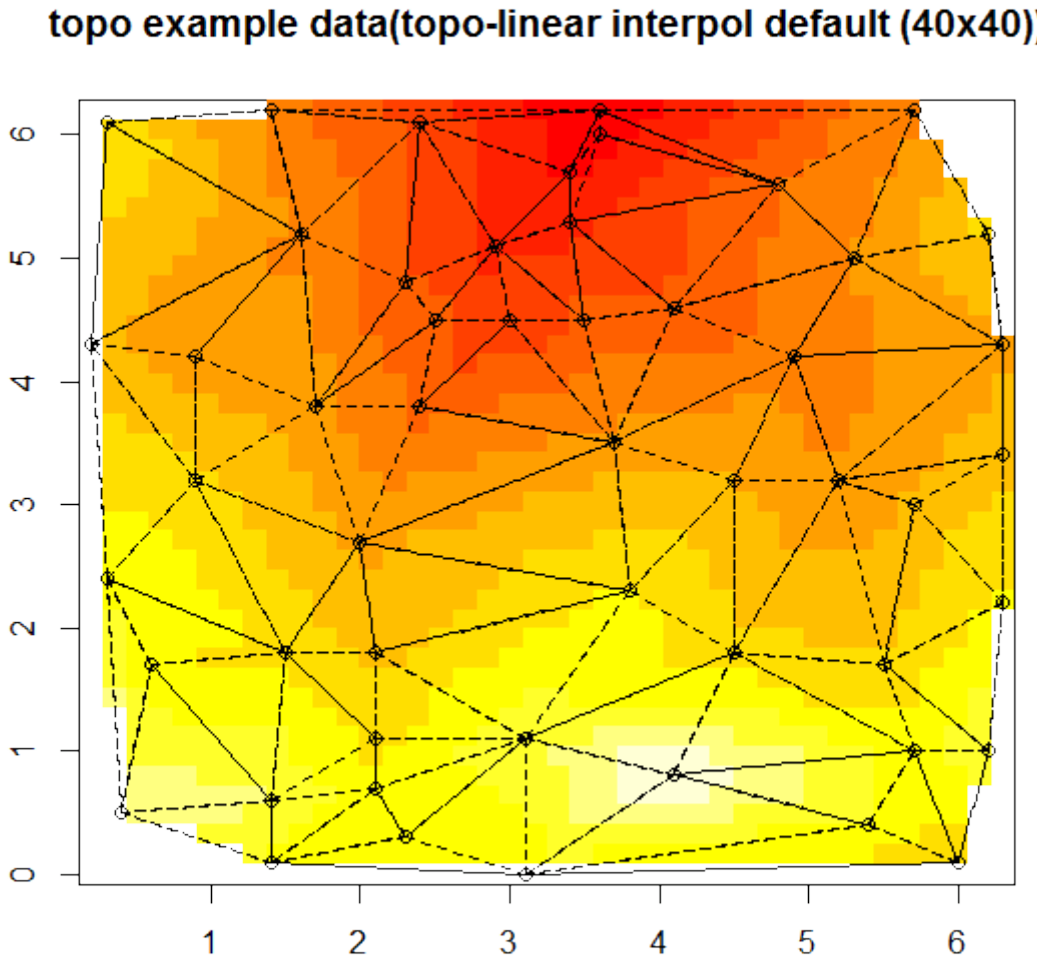
- ***Thiessen or Voronoi tessellation*** **- area divided into regions around the sample data points where every pixel or area in the region is assigned to the data point which is the closest**
- ***TIN*** **- sample data points become vertices of a set of triangular facets covering the study area**


topo example data
870
793
755
690
705
800
710
780
800
728
730
804
762
760
765
740
765
830
813
790
812
773
812
855
827
805
830
820
890
873
873
865
841
855
850
862
908
882
960
940
915
880
870
890
890
880
860
0
1
2
3
4
5
6
y
x

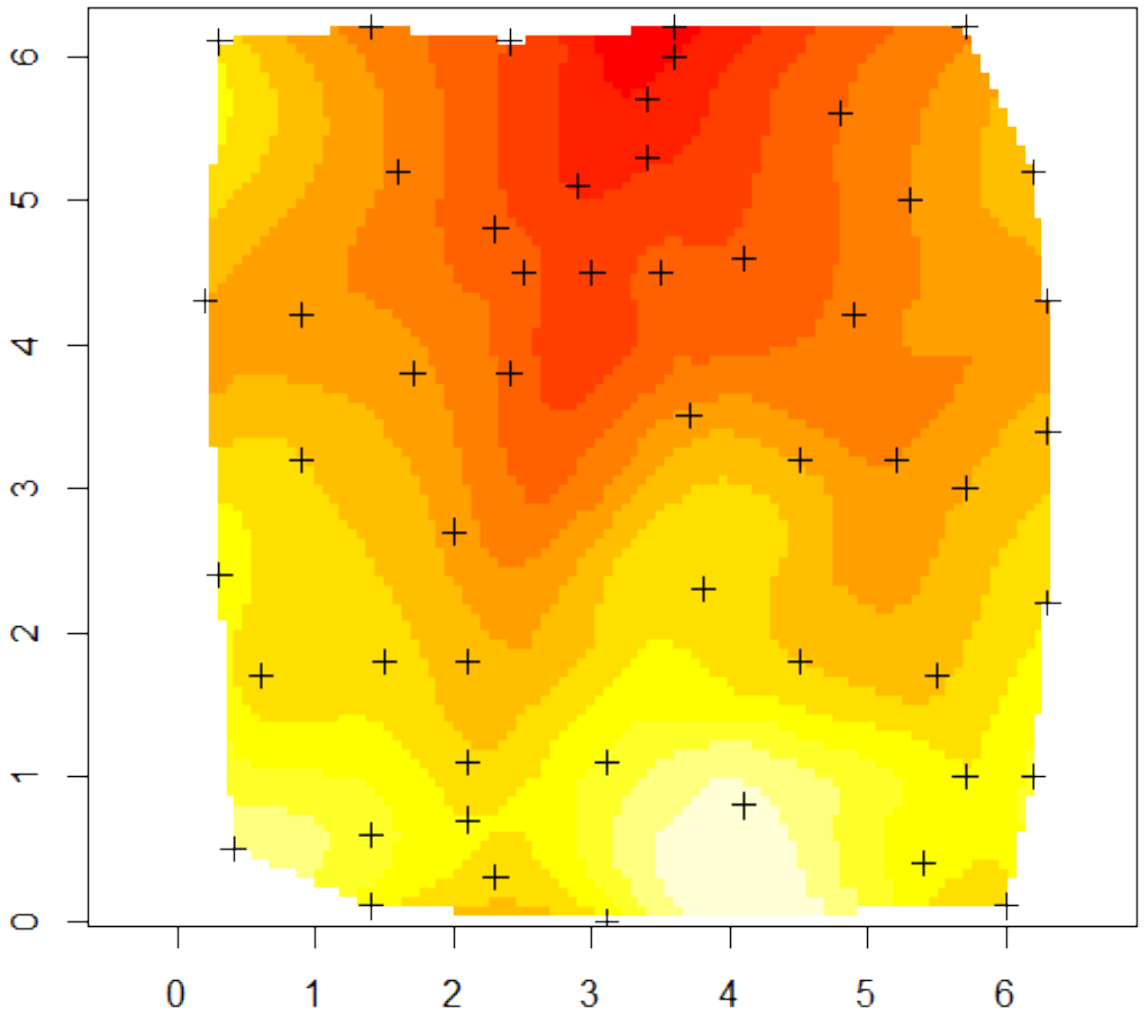

topo example data(linear interpol default (40x40))
720
740
760
780
800
820
840
860
880
900
920
840
820
880
900
920
880
0
1
2
3
4
5
6
1
2
3
4
5
6


topo example data(linear interpol default (40x40))
0
1
2
3
4
5
6


topo smooth interp (on finer grid 100x100)
0
1
2
3
4
5
6



Minimum angle triangles, bounded by data points, are used in the linear interpolation algorithm


topo smooth interp(* bicubic100x100)
0
1
2
3
4
5
6
0
1
2
3
4
5
6

Polynomial trend surfaces

- A global predictor which models a **regional trend**
- The value of a variable at each point depends only on its **coödinates** and parameters of a fitted surface
- This is modelled with a smooth function of position, $z = f(x,y) = f(E,N)$ for grid coördinates; this is called the **trend surface**
- Simple form (plane, 1st order):

$$z = \beta_0 + \beta_x E + \beta_y N$$

- Higher-order surfaces may also be fitted (beware of fitting the noise!)

(from David Rossiter)

General trend surfaces

- General form: a surface of order p:

$$f((x,y)) = \sum_{r+s \leq p} \beta_{r,s} x^r y^s$$

- Order should be suggested by the process
  - One-way trend: 1st order (plane)
  - Dome or depression: 2nd order
  - Folded structure: Higher orders, depending on number of inflection points

Properties: stochastic, inexact, global

(from David Rossiter)

Trend surface interpolation

Polynomials in x, y (or x, y, z):

$$Z(x, y) = \beta_0 + \beta_1 x + \beta_2 y + e(x, y)$$

$$Z(x, y) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 y^2 + \beta_5 xy + e(x, y)$$

coefficient vector β is globally constant:

- allows testing, etc.
- testing assumes... independence

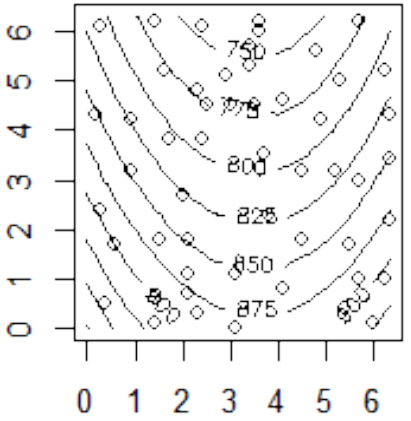
Fitting trend surfaces

- The trend surface is predicted by **linear regression** with **coödinates as the predictor variables** and the response variable to be predicted, using data from all sample points.
- All samples participate equally in the prediction
- We can measure the **goodness of fit** of the trend surface to the sample by the residual sum of squares
- The same **cautions** as in feature-space regression analysis!
- **Ordinary Least Squares** (OLS) is often used but is not really correct, since it ignores possible **correlation among closely-spaced samples**; better is **Generalised Least Squares** (GLS)

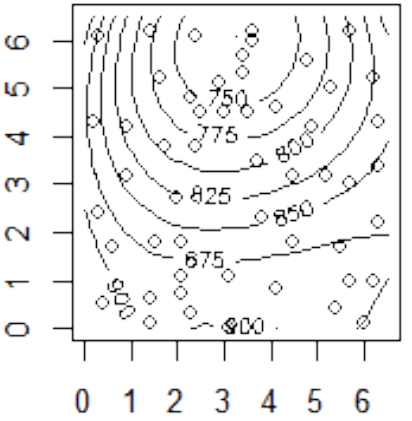
(from David Rossiter)

Trend Surface – Topo

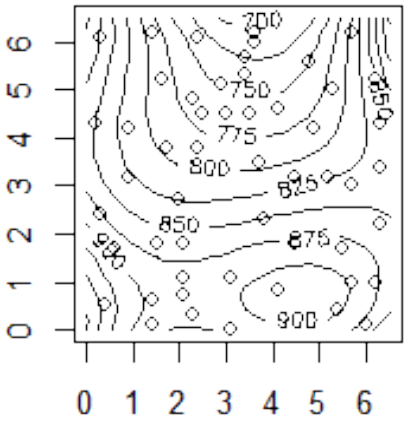
Degree=2



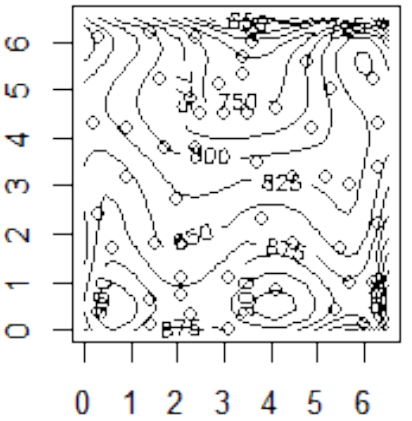
Degree=3

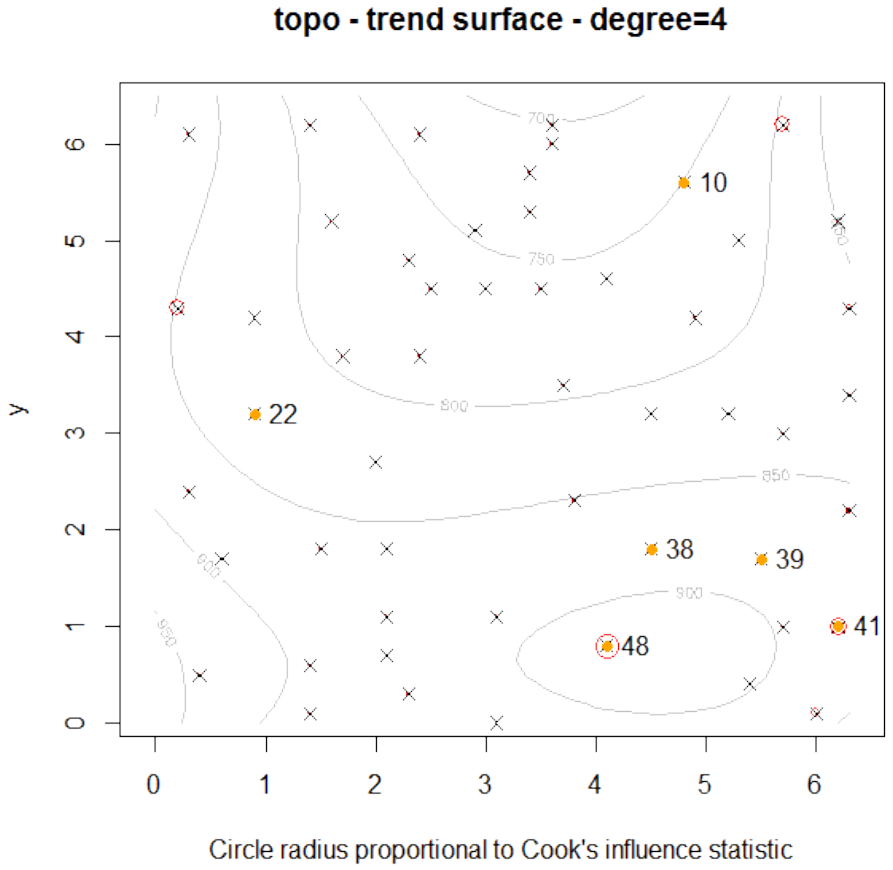


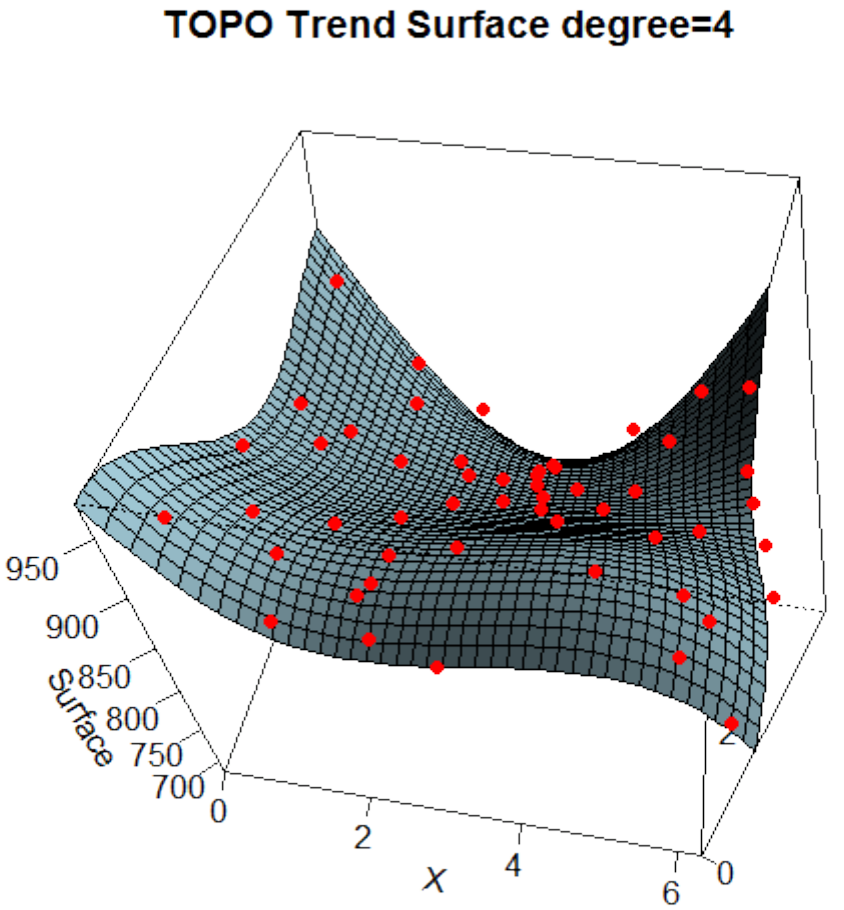
Degree=4



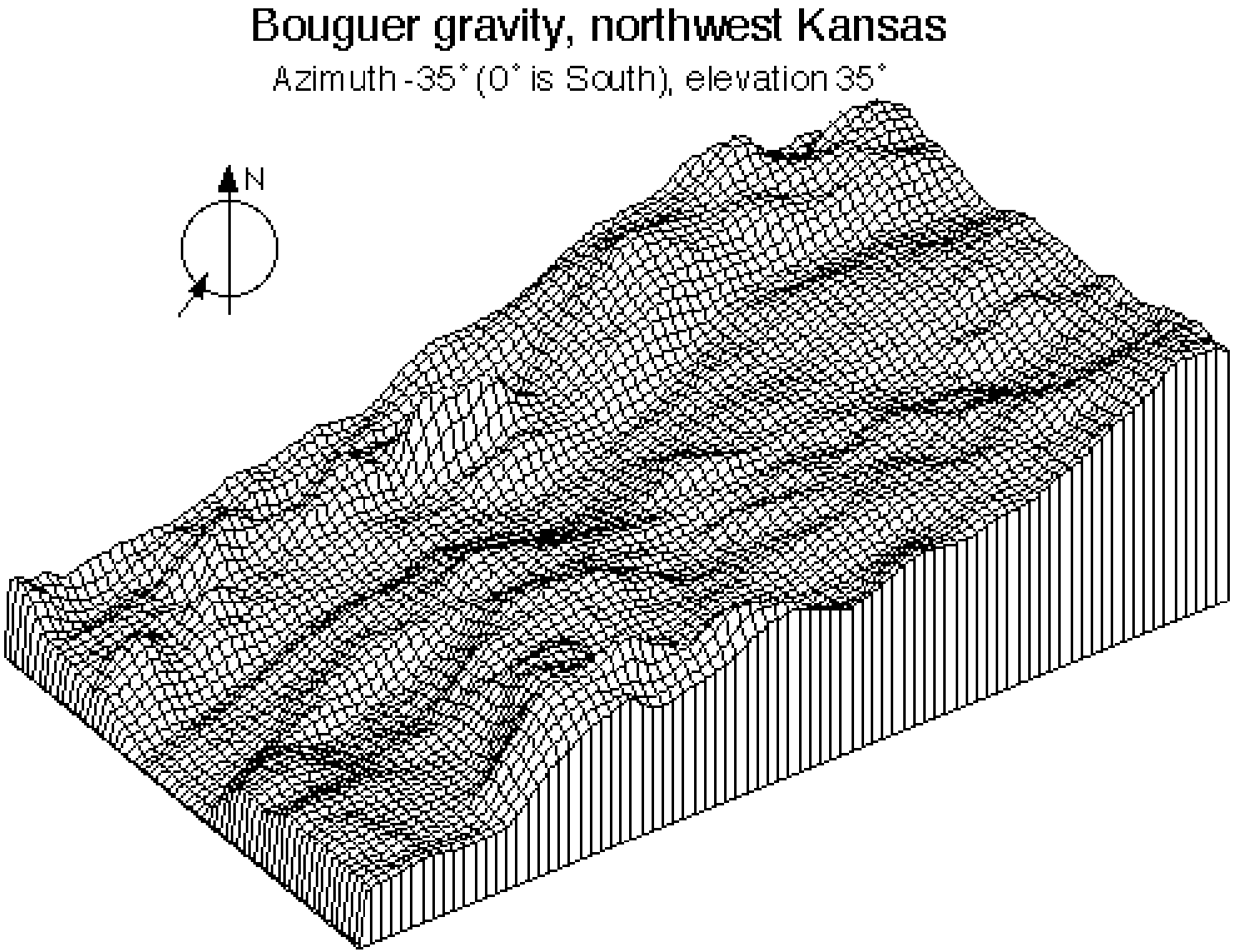
Degree=6

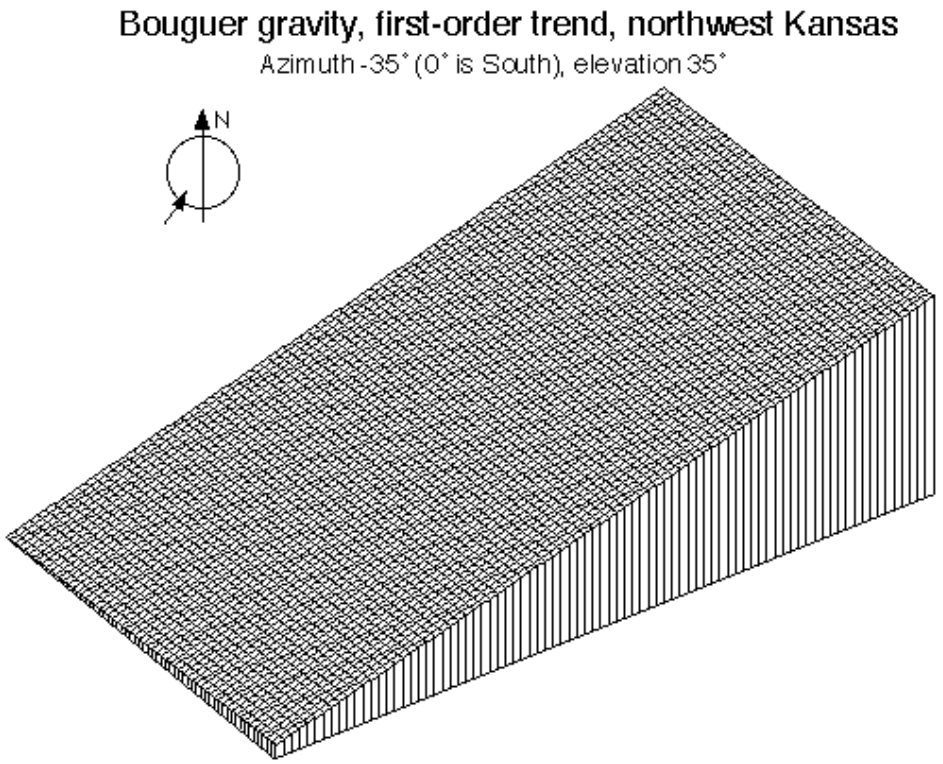



topo - trend surface - degree=4
700
750
800
850
900
950
10
22
38
39
41
48
y
0
1
2
3
4
5
6
Circle radius proportional to Cook's influence statistic

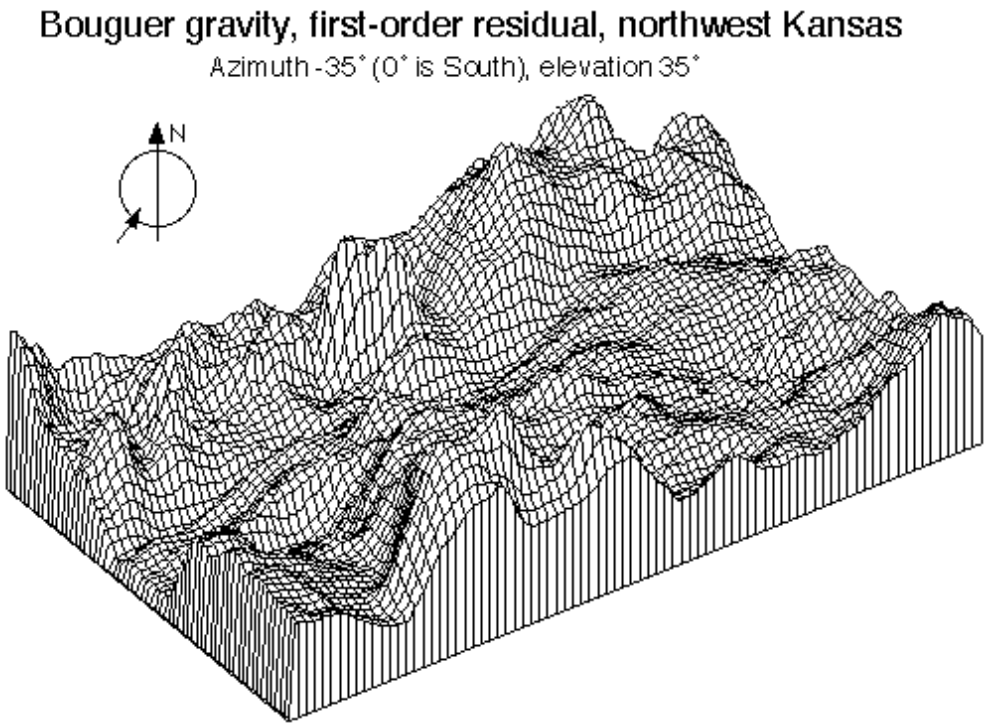

TOPO Trend Surface degree=4
950
900
850
800
750
700
Surface
0
2
4
6
X
0

Trend surface examples





The first-order polynomial of the gravity data set shows the strong east-to-west gradient.

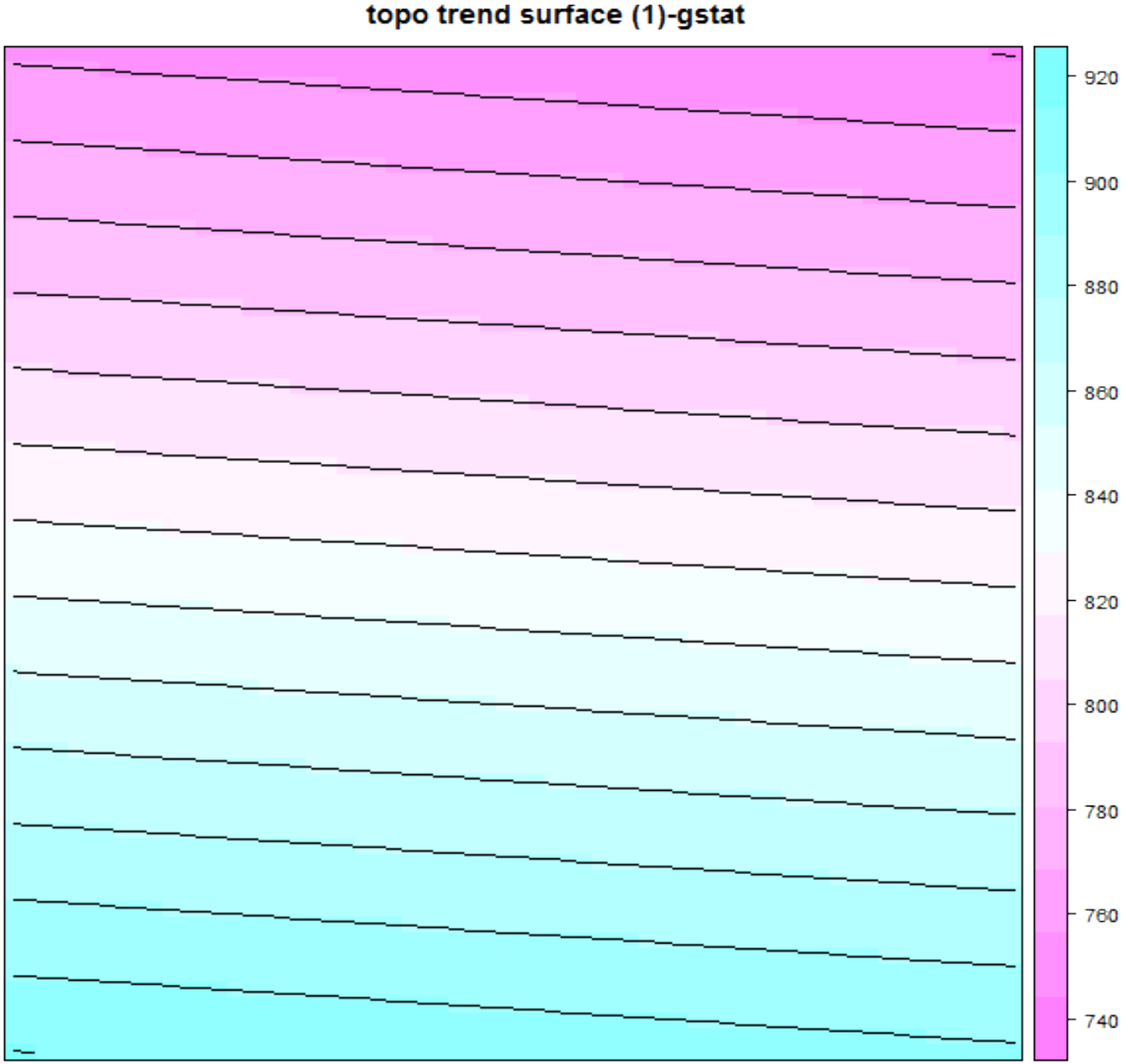


Once the background gradient is removed, the local anomalies are displayed more clearly.

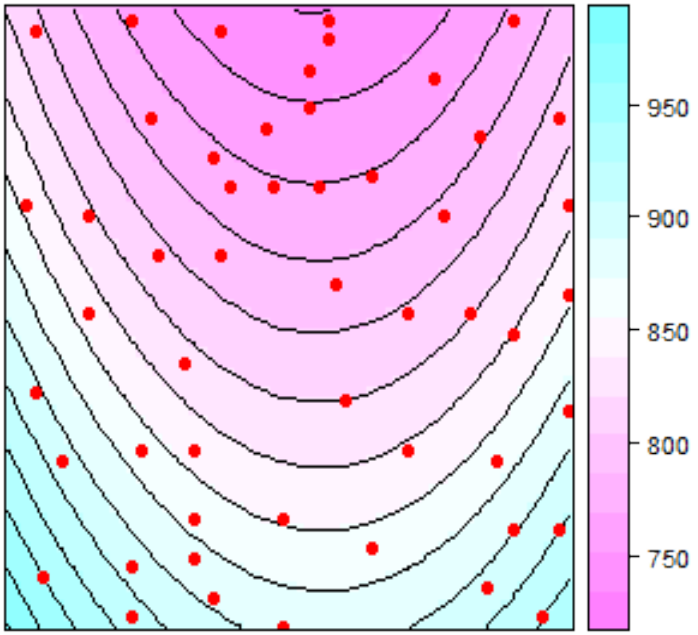
Trend surface in gstat

gstat treats this as a special case of universal kriging with no spatial dependence (i.e. NULL model) ## just to degree=2:this example

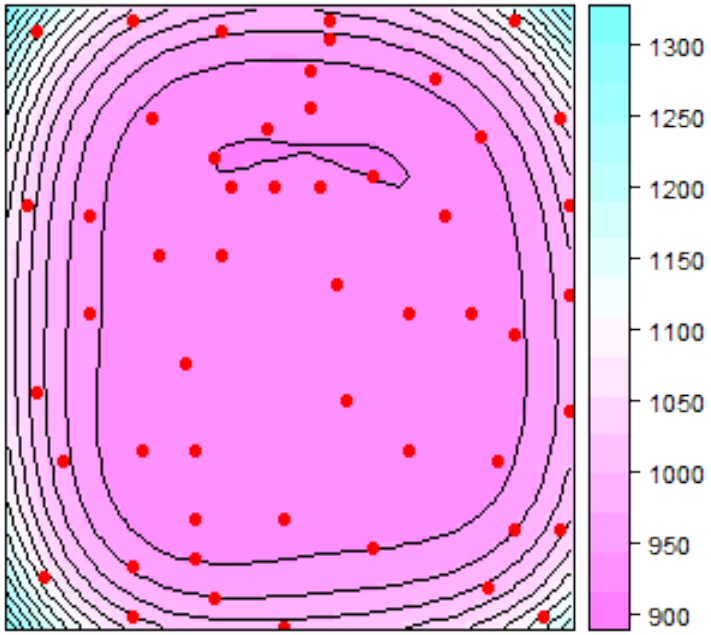
```
data(topo)
coordinates(topo) <- ~ x+y
x=seq(0,6.3, by=0.1);y=seq(0,6.3, by=0.1)
topo.grid <- expand.grid(x=x,y=y)
gridded(topo.grid) <- ~x+y
# 1st-order trend surface
x <- gstat(formula=topo$z ~ 1, data=topo, degree=1)
kt <- predict(x, newdata=topo.grid)
spplot(kt[1], contour=TRUE,main="topo trend surface (1)-gstat")
# 2nd order trend surface
x <- gstat(formula=topo$z ~ 1, data=topo, degree=2)
kt2 <- predict(x, newdata=topo.grid)
p1 <- spplot( kt2[1], contour=TRUE, main="topo trend surface (2)-
    gstat",sp.layout=list(list("sp.points", topo@coords, pch=19,
    col=2)))
# prediction error from OLS regression
p2 <- spplot(kt2[2], contour=TRUE,sp.layout=list(list("sp.points",
      topo@coords, pch=19, col=2)),main="prediction error from OLS
      regression")
print(p1, position = c(0,.5,1,1),more=T)
print(p2, position = c(0,0,1,.5))
```


topo trend surface (1)-gstat
920
900
880
860
840
820
800
780
760
740

topo trend surface (2)-gstat



prediction error from OLS regression



Some Problems with Trend Surfaces

1. Extreme simplicity in form of a polynomial surface as compared to most natural surfaces.

2. Unfortunate tendency to accelerate without limit to higher or lower values in areas where there are no control points, such as along the edges of maps.

3. Computational/statistical difficulties may be encountered if a very high degree polynomial trend surface is fitted.

4. Has problems with outliers

Tessellations

- Thiessen Polygons
  - collection of all points in a plane that are closer to the central point than to any other point
  - notion of minimum transport cost market area
  - all points in the polygon are assigned the same value as the central point
- Properties
  - local, exact, deterministic
  - interpolated values underestimate heterogeneity

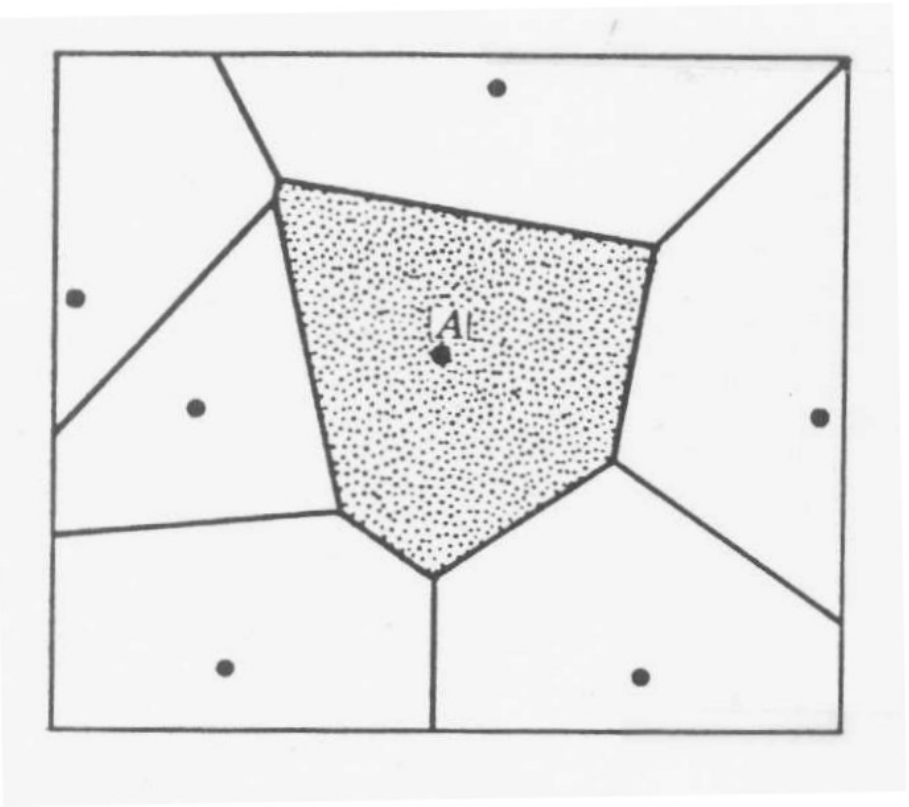


FIGURE 5.45 Thiessen polygon (shaded) around point *A*. All locations within polygon are closer to *A* than to any other point.

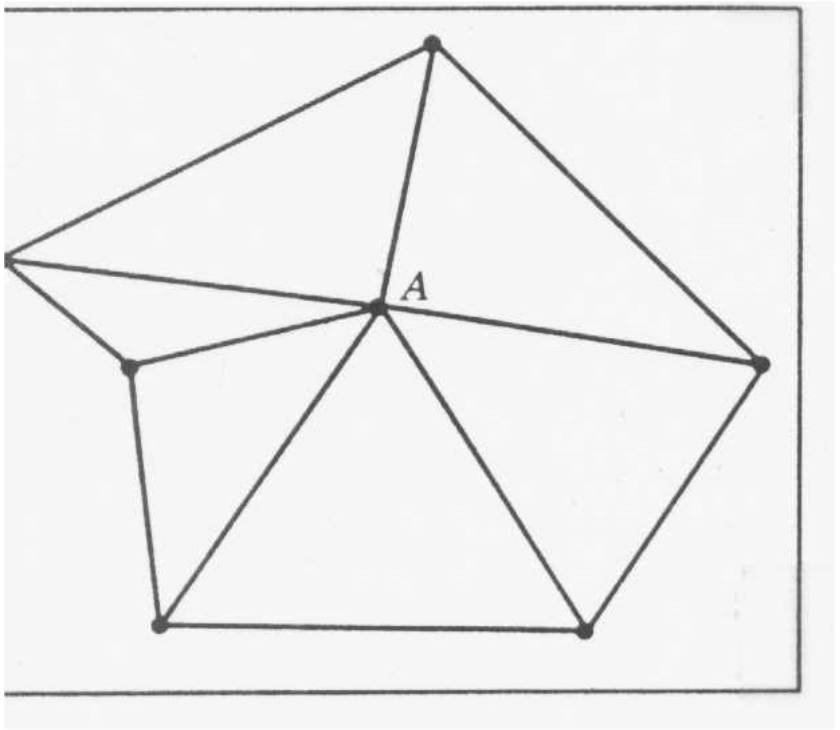
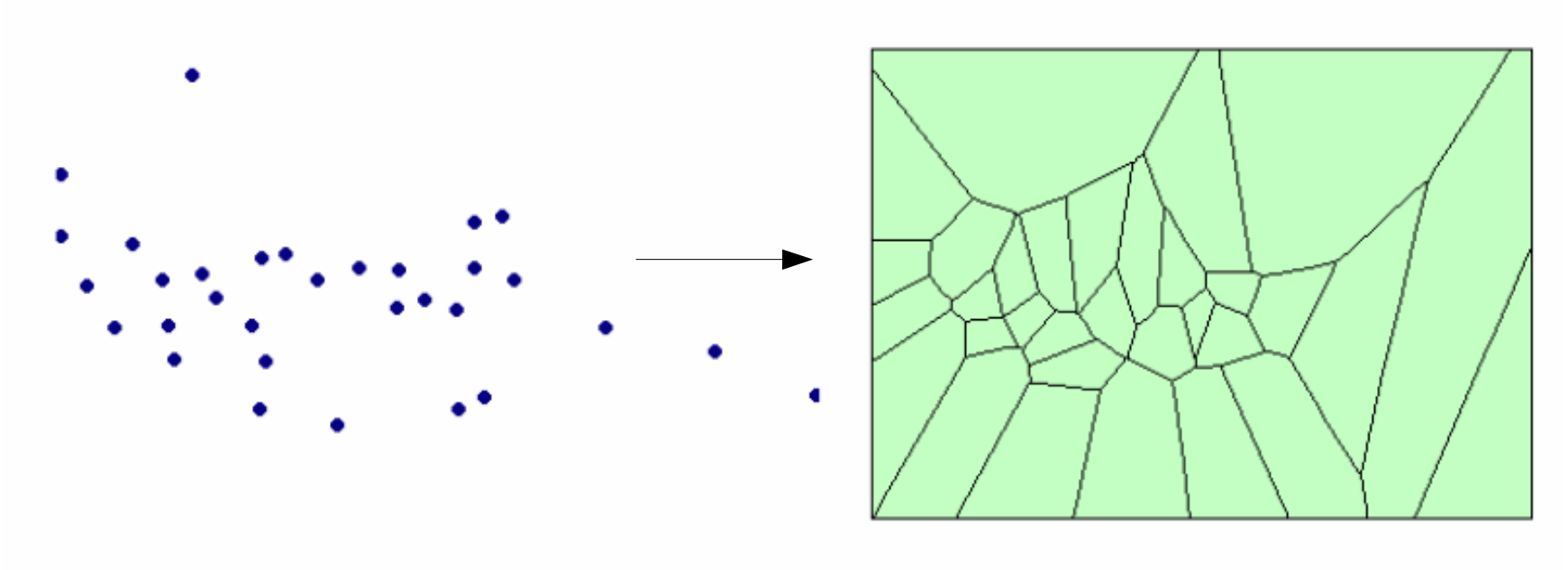
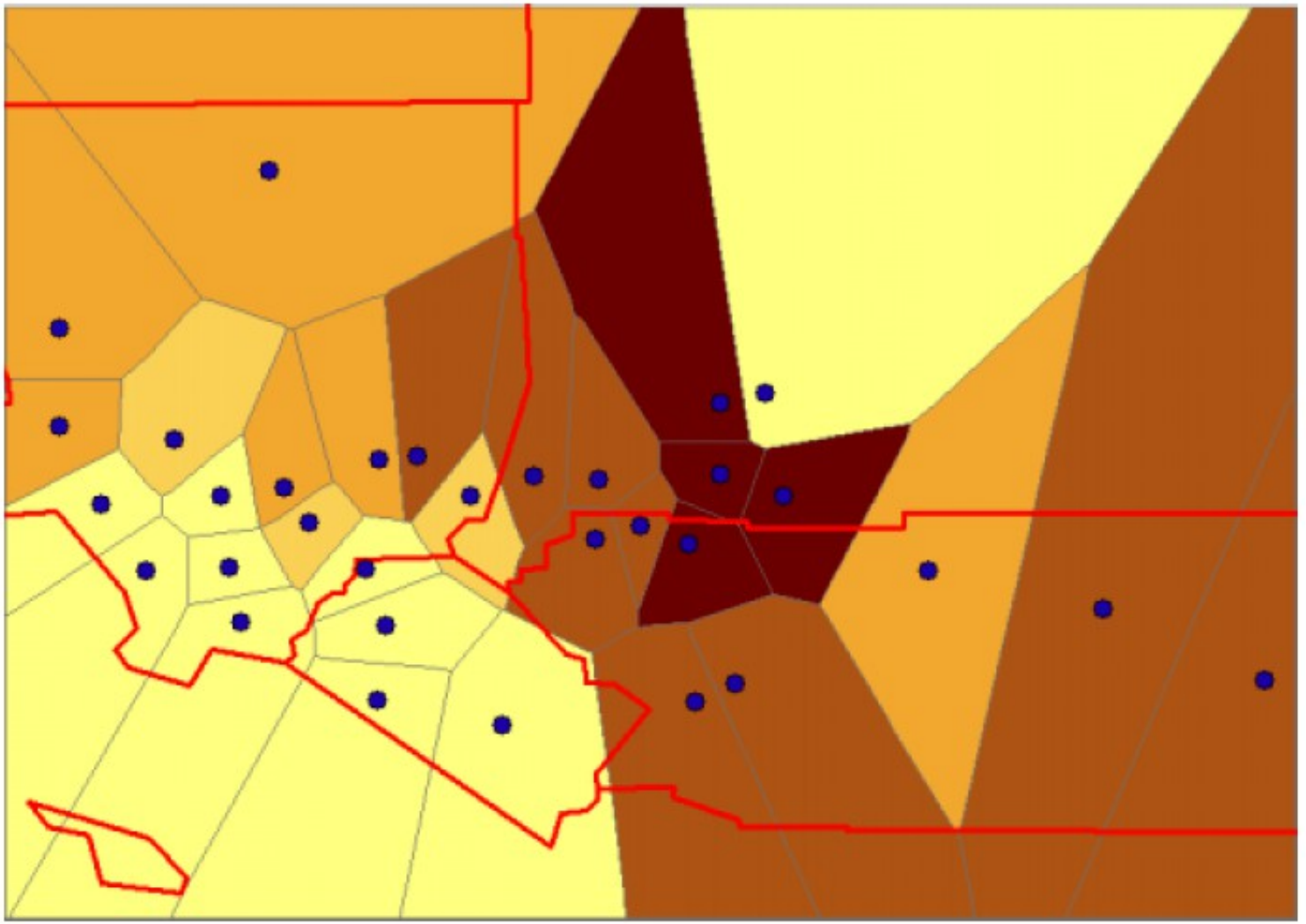


FIGURE 5.46 Delaunay triangle network around point *A*. All connected points are Thiessen neighbors of *A*.

from points to polygons: Thiessen polygons for LA basin





Thiessen Polygon Interpolation - LA Basin Ozone

Inverse distance weighted interpolation (IDW)

This is one of the simplest and most readily available methods. It is based on an assumption that the value at an unsampled point can be approximated as a weighted average of values at points within a certain cut-off distance, or from a given number m of the closest points (typically 10 to 30). Weights are usually inversely proportional to a power of distance

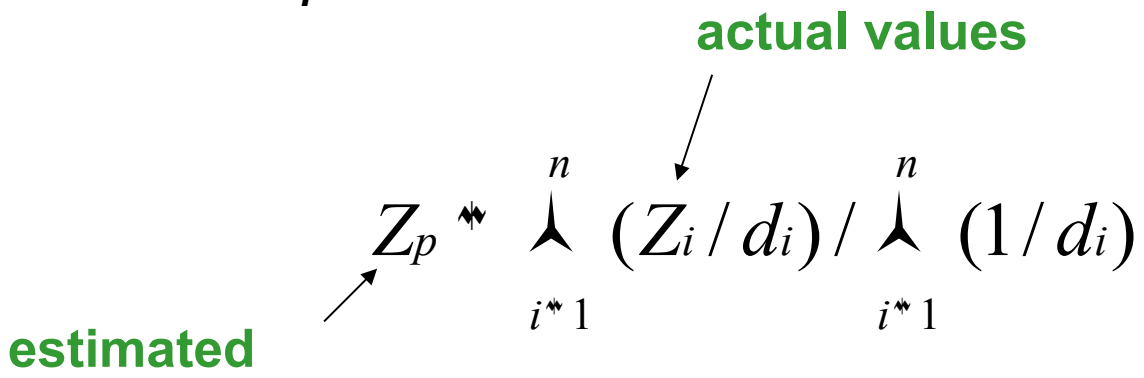
The method often does not reproduce the local shape implied by data and produces local extrema at the data points

Inverse Distance Weighting

- Tobler's Law
  - in space, everything is related to everything else, but closer locations more so
  - use distance decay
- IDW
  - $z_x = \Sigma_i \lambda_i z_i$ with $\Sigma_i \lambda_i = 1$ or $z_x = \Sigma_i w_i z_i / \Sigma_i w_i$
  - w_i = weight
    - » inverse distance function: $1/d_{ix}^2$
- Properties
  - local, exact, deterministic
  - interpolated values are always within range

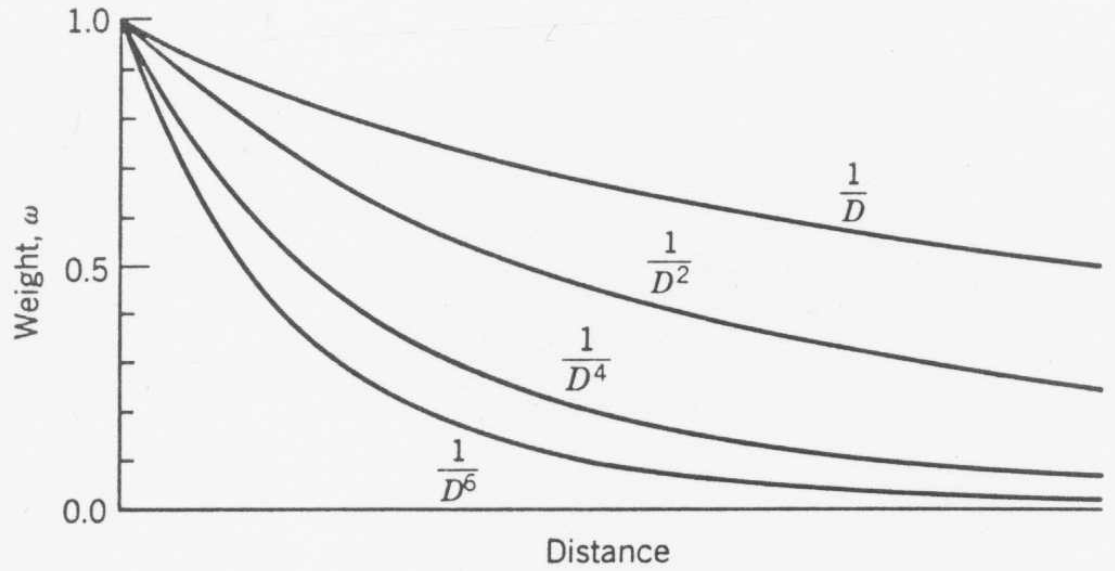
Inverse Distance Weighting

Substitute (d_i)

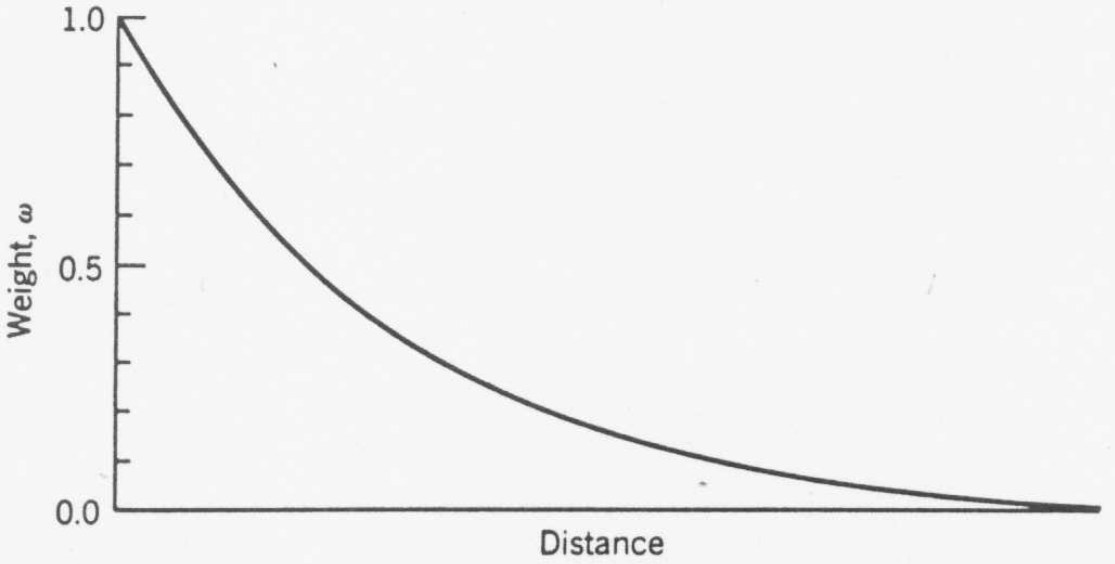


where

$$d_i = \{(x_i - x_p)^2 + (y_i - y_p)^2\}^{0.5}$$



(a)



(b)

FIGURE 5.56 Distance weighting functions used in contouring programs. (*a*) Inverse distance-powered functions. (*b*) Scaled inverse distance-squared function.

Defining sample subsets

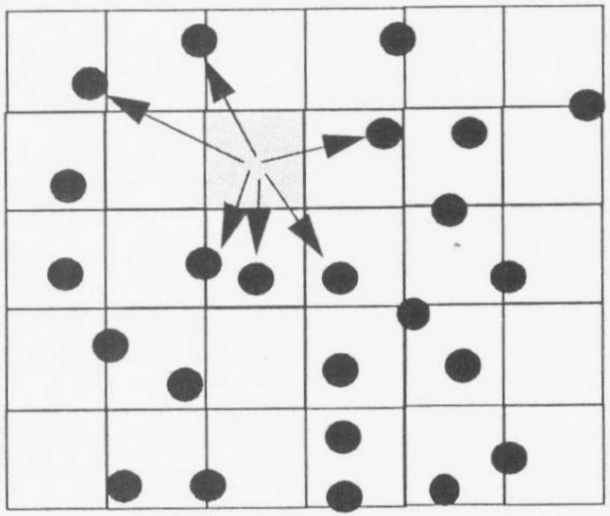
- Which samples are considered during interpolation?

  Nearest number of neighbors

  Samples within a radius

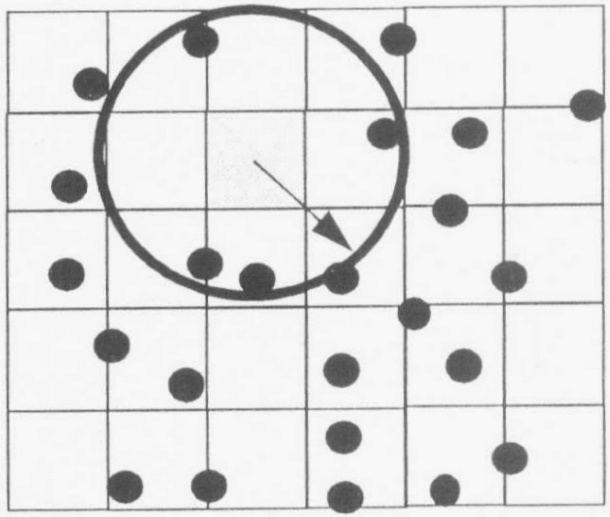
- Only select points are used

Nearest neighbors



Number of nearest neighbors = 6

Fixed radius



Radius = 1280
Samples = 4

A PROBLEM

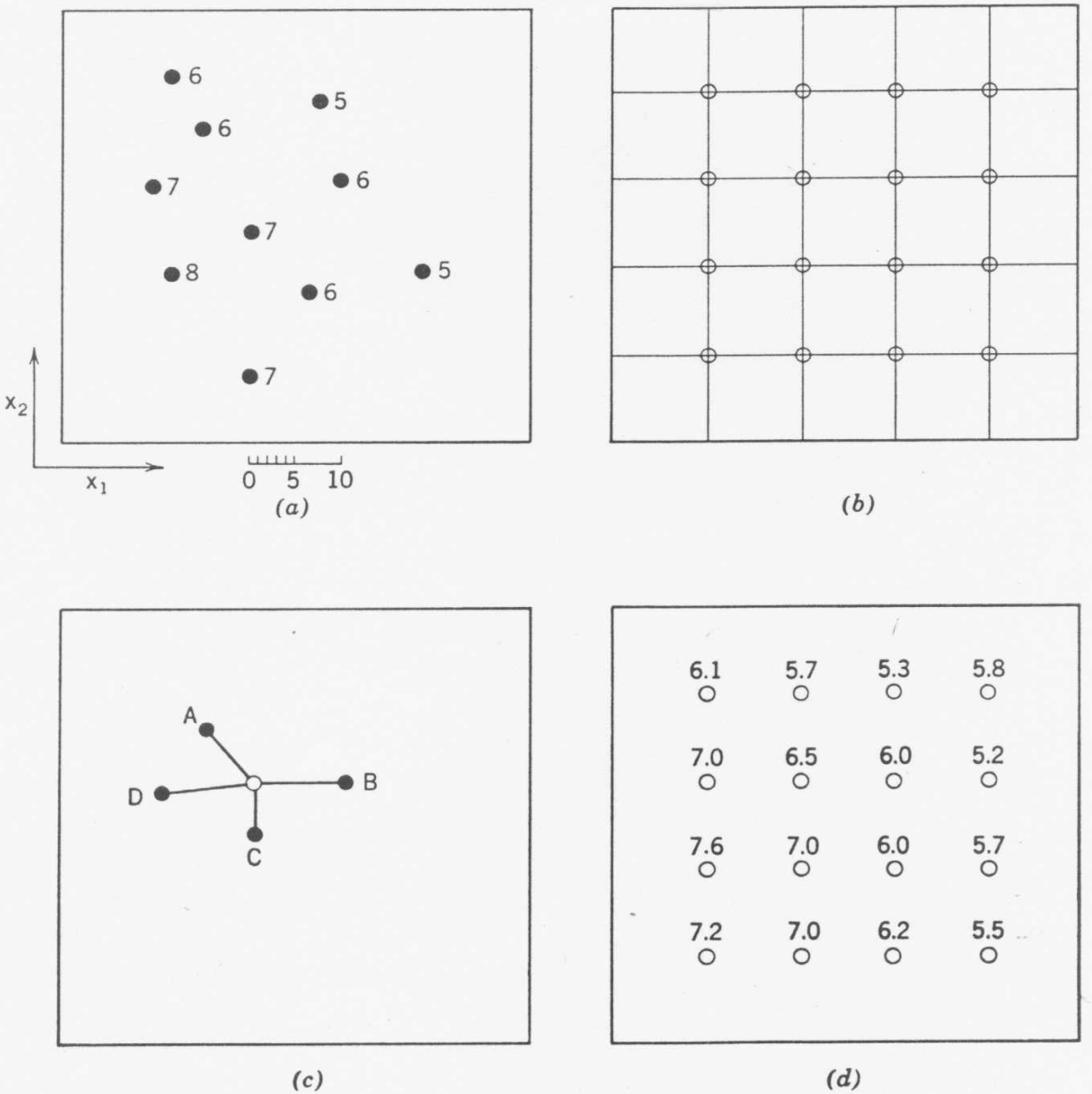
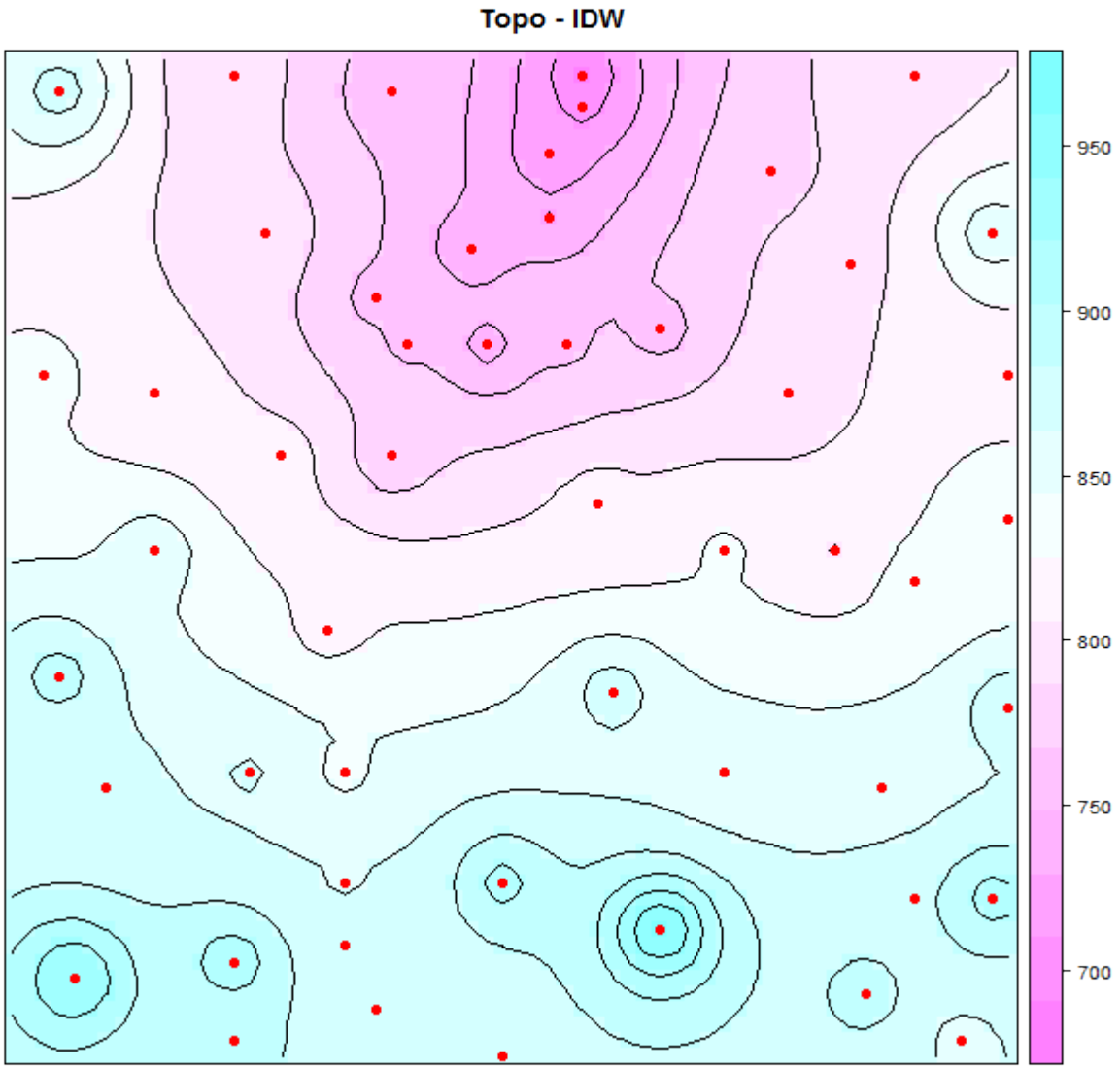


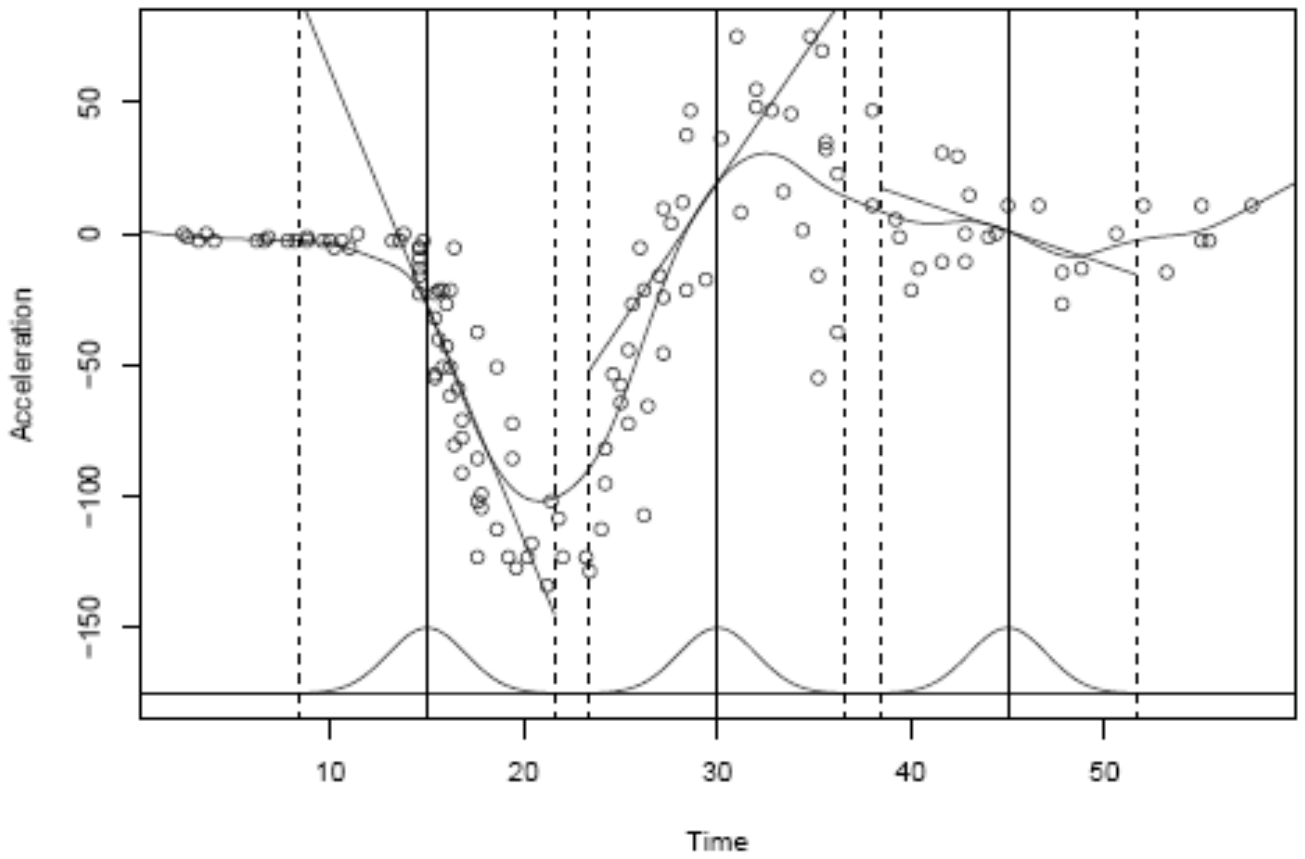
FIGURE 5.53 Steps in computation of grid value for contouring. (*a*) Original set of irregularly spaced control points on a map. Numbers are "elevations." (*b*) Regularly spaced grid network, with nodes whose values are to be calculated. (*c*) Location of four nearest control points to one grid node. These four nearest values will be used to calculate grid value. (*d*) Completed grid with estimated elevations at every node.



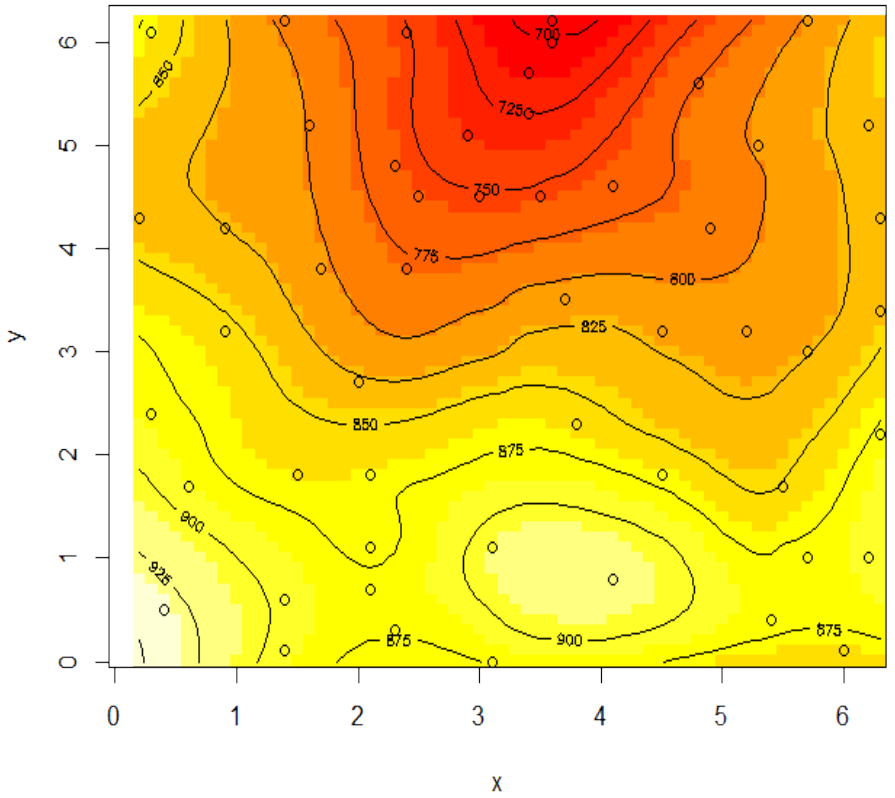

```
#IDW
kid <- krige(topo$z ~ 1, topo, topo.grid, NULL)
spplot(kid["var1.pred"], main = "TopIDW",contour=TRUE,
  sp.layout=list(list("sp.points", topo@coords, pch=19,
  col=2)))
```

Local Polynomial Regression Fitting (LOESS)

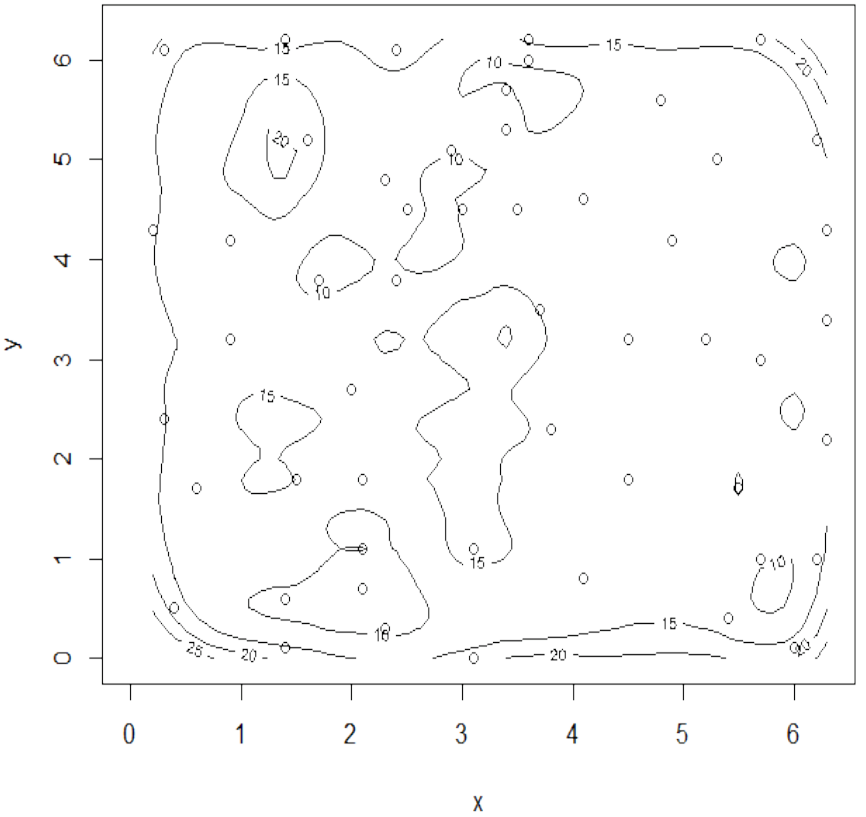
- Fitting is done ***locally***. The fit is made using points in a neighbourhood of x, weighted by their distance from x.
- The size of the neighbourhood is controlled by α (set by span parameter and controls the degree of smoothing).
- The neighbourhood points (α < 1) have tricubic weighting (proportional to (1 – (dist/maxdist)^3)^3).
- For α > 1, all points are used, with the ‘maximum distance’ assumed to be α^(1/p) times the actual maximum distance for p explanatory variables.


Acceleration
50
0
−50
−100
−150
10
20
30
40
50
Time

topo Loess degree = 2



loess -topo standard error degree=2



Spline Regression

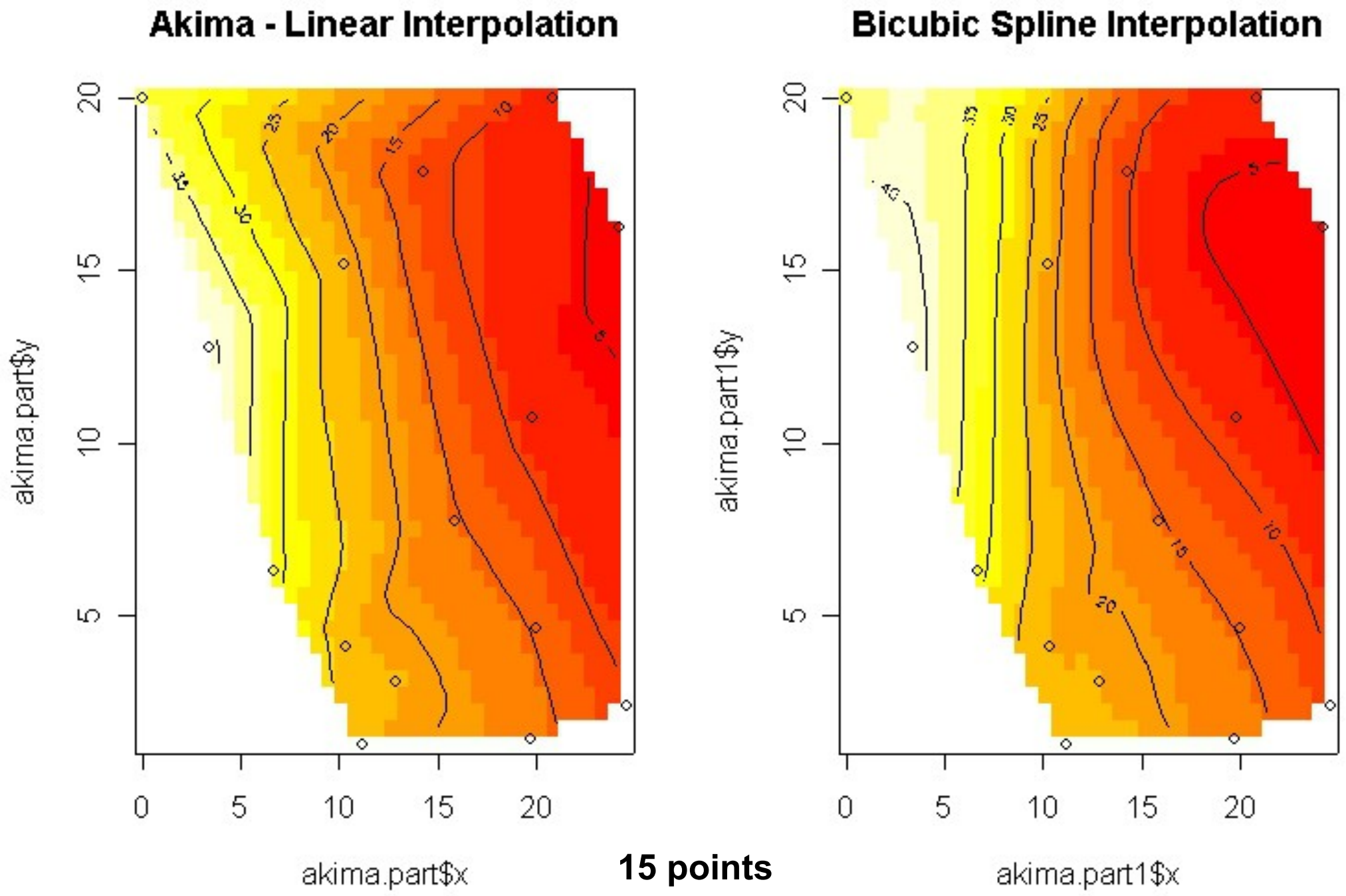
Spline regressions are regression line segments that are joined together at special join points known as spline knots. By definition, there are no abrupt breaks in a spline line. However, at their knots, splines may suddenly change direction (i.e., slope), or, for curved lines, there may be a more subtle, but just as sudden, change in the rate at which the slope is changing. Splines are used instead of quadratic, cubic, or higher-order polynomial regression curves because splines are more flexible and are less prone to multi-collinearity, which is the bane of polynomial regression.

“You can add a spline to whatever model, so your dependent variable, can be continuous, bounded, discrete, categorical, or whatever else you can or cannot think of. A spline is added because you think that the effect of a variable is non-linear, so it makes only sense when the explanatory variable is continuous.” (Lawrence C. Marsh)

B-splines
uses a piecewise polynomial to provide a series of patches resulting in a surface that has continuous first and second derivatives ensures continuity in:

- elevation (zero-order continuity) - surface has no cliffs
- slope (first-order continuity) - slopes do not change abruptly, there are no kinks in contours
- curvature (second order continuity) - minimum curvature is achieved produces a continuous surface with minimum curvature

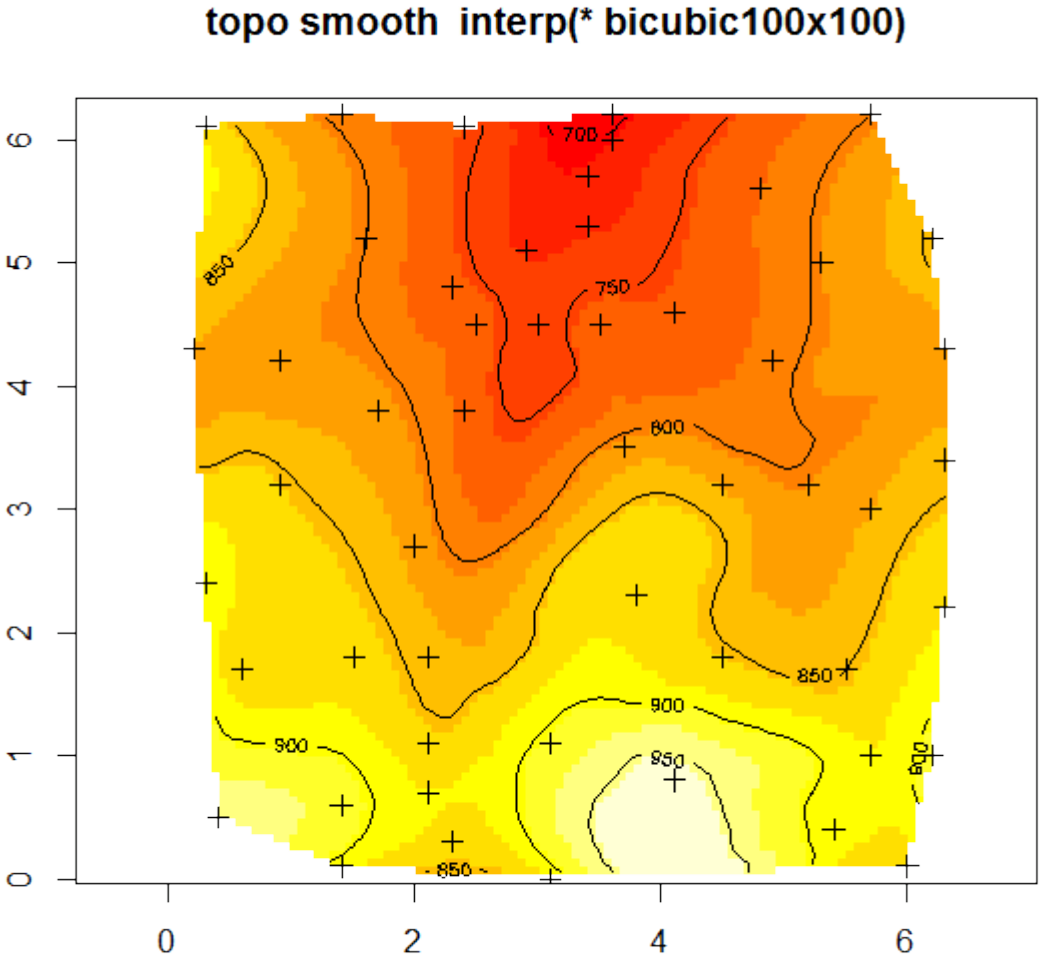
SPLINE



15 points

Bicubic spline

- interpolation of a bivariate function, z(x,y), is made on a rectangular grid in the x-y plane.
- the interpolating function is a piecewise function composed of a set of bicubic (bivariate third-degree) polynomials, each applicable to a rectangle of the input grid in the x-y plane. Each polynomial is determined locally - with the piecewise segments joined at *knots* and with various degrees of smoothness.




```
akima.spl <- with(topo, interp(x,y,z, xo=seq(0,6.3,
   length=100), yo=seq(0,6.3, length=100),linear=FALSE))
image(akima.spl,asp=1,main = "topo smooth  interp(*
   bicubic100x100)")
contour(akima.spl,add=TRUE)
points(topo,pch=3)
```

Thin Plate Spline

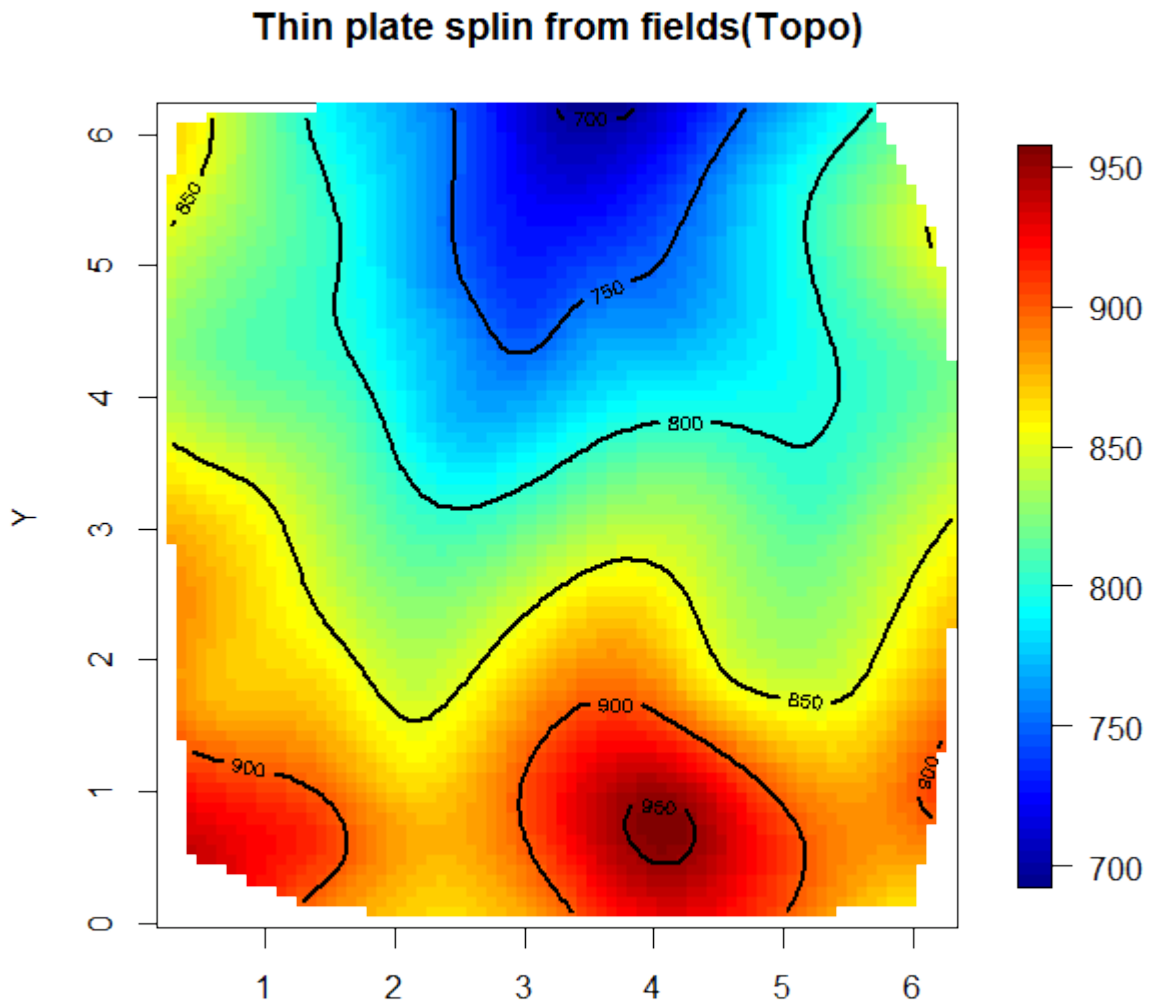
The thin plate spline is a two-dimensional variation of the cubic spline. It is the fundamental solution to the biharmonic equation, and has the kernel form

$$U(r) = r^2 \ln r.$$

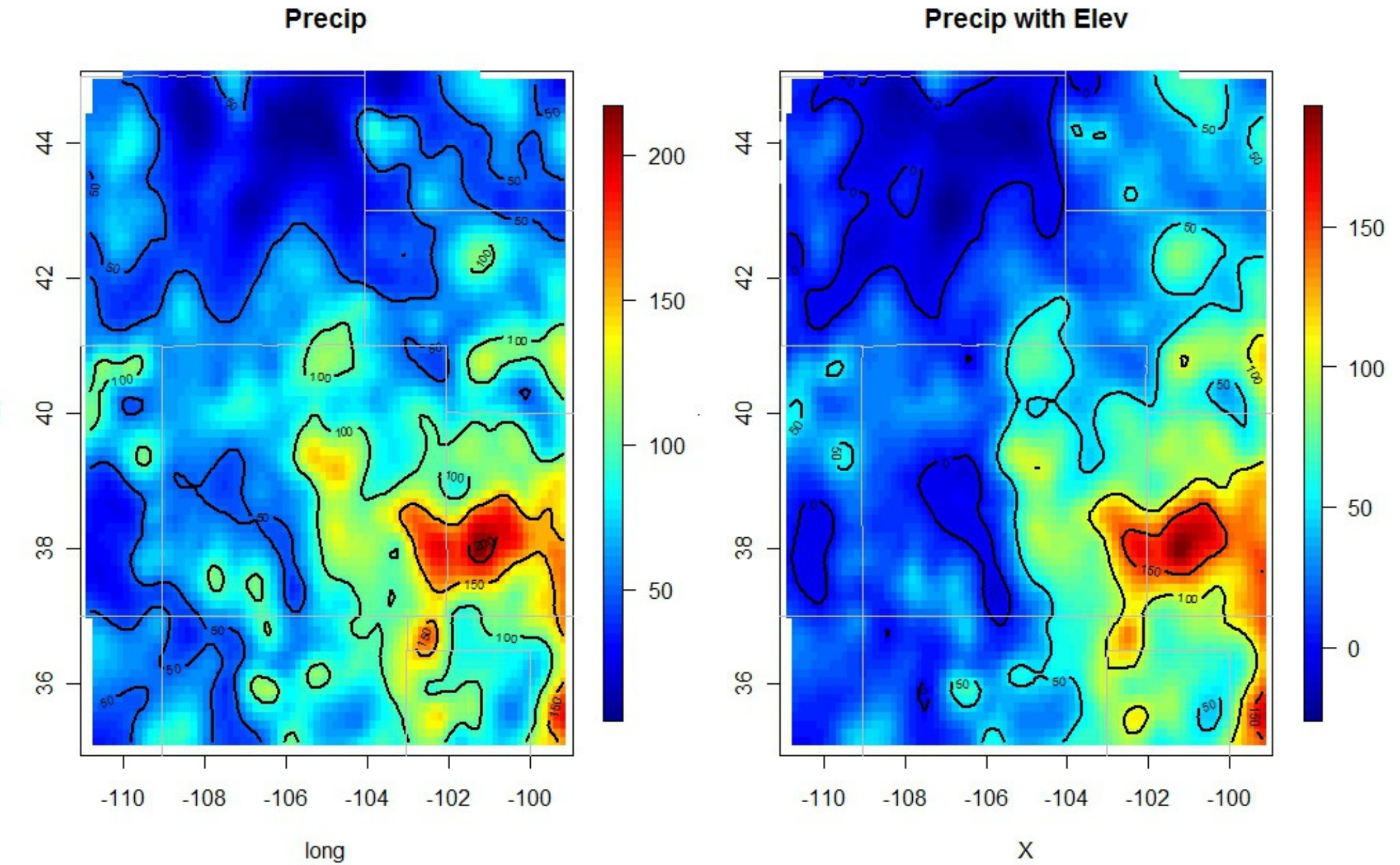
Given a set of data points, a weighted combination of thin plate splines centered about each data point gives the interpolation function that passes through the points exactly while minimizing the so-called "bending energy" or tension. Bending energy is defined here as the integral over domain of the squares of the second derivatives:

$$I[f(x,y)] = \iint_{\mathbb{R}^2} \left(f_{xx}^2 + 2 f_{xy}^2 + f_{yy}^2 \right) dx\, dy.$$

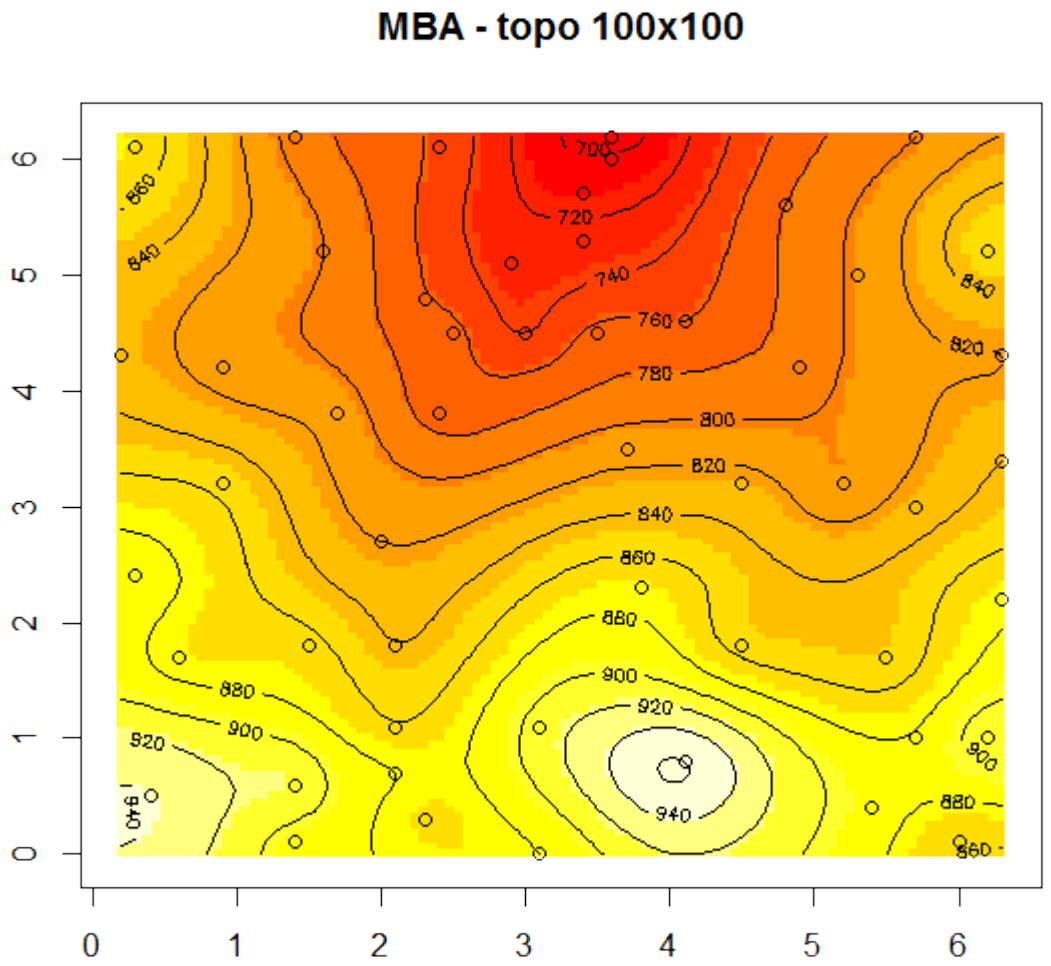
Regularization may be used to relax the requirement that the interpolated point pass through the data points exactly.

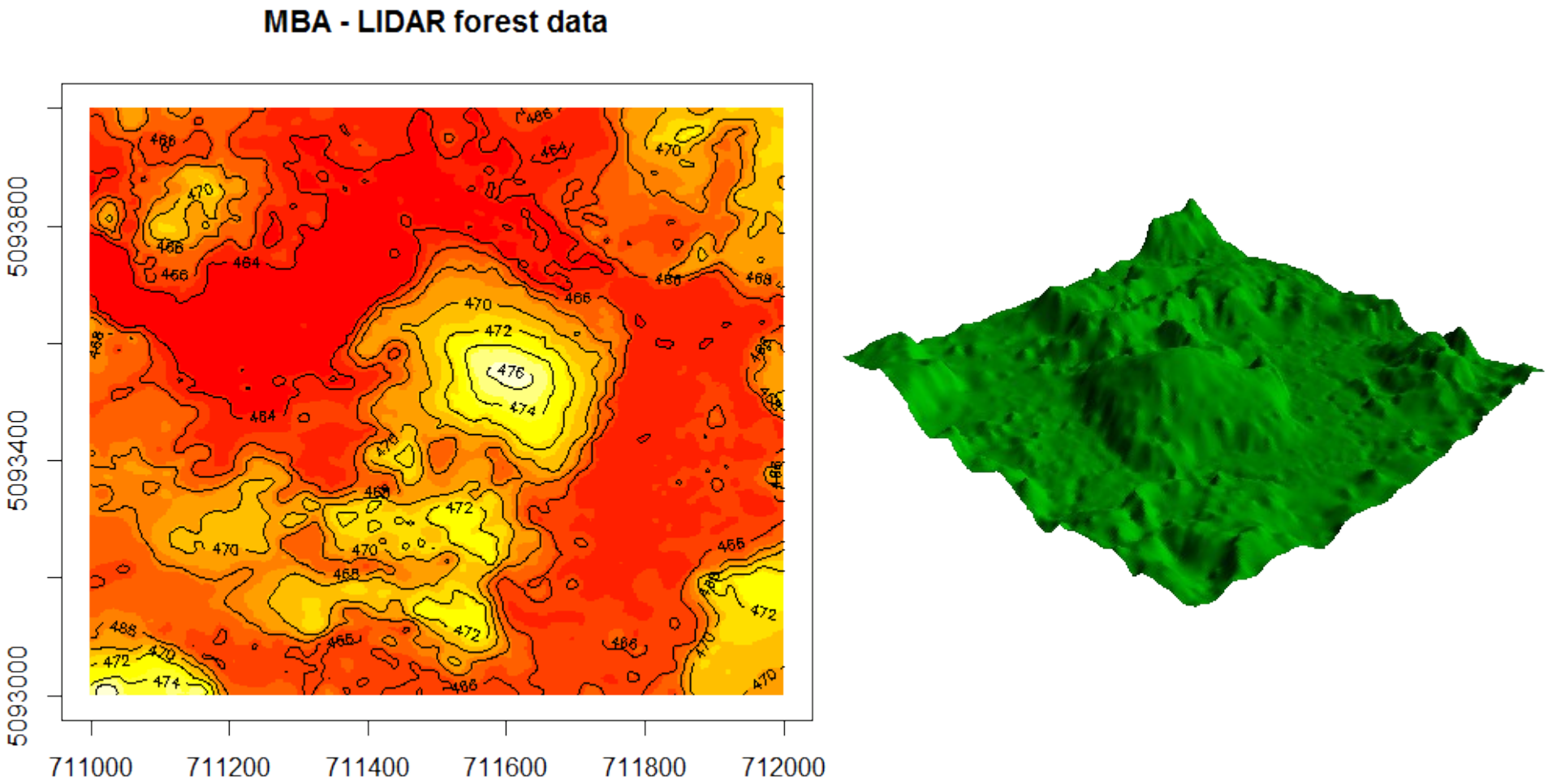



```
#Tps from "fields" for topo  x
data(topo)
setting up data for Tps
x <- topo$x; y <- topo$y; z <- topo$z
tp <- cbind (x,y); tpo <- list(x=tp,y=z)
tout <- Tps(tpo$x,tpo$y) #thin plate spline function
surface(tout)
title("Thin plate spline from fields(Topo)")
```


Precip
Precip with Elev
44
42
40
38
36
-110
-108
-106
-104
-102
-100
long
X
200
150
100
50
0

Mult-level B-splines




MBA - LIDAR forest data
5093800
5093400
5093000
711000
711200
711400
711600
711800
712000

Surface approximation from bivariate scattered data using multilevel B-splines

```
require(MBA)
data(LIDAR)# 10123 points taken over forest area
mba.int <- mba.surf(LIDAR, 300, 300, extend=TRUE)
    $xyz.est
##Image plot
image(mba.int, xaxs="r", yaxs="r",main="MBA - LIDAR
    forest data")
contour(mba.int,add=TRUE)
##Perspective plot
persp(mba.int, theta = 135, phi = 30, col = "green3",
   scale = FALSE, ltheta = -120, shade = 0.75, expand
   = 10, border = NA, box = FALSE)
```

Spline Interpolation

- Fits a minimum-curvature surface through input points
- Like a rubber sheet that is bent around the sample points
- Best for gently varying surfaces such as elevation
- Estimates can result in spurious ridges and valleys

Spatial Prediction-Kriging

- **Model of Spatial Variability – theory of regionalized variable**
  - $Z(x) = \mu(x) + k(x) + \varepsilon$
    - $\mu(x)$ is a deterministic function describing the 'structural' component of Z
    - $k(x)$ denotes the stochastic, locally varying but spatially dependent residuals from $\mu(x)$-the regionalized variable
    - ε is a spatially independent Gaussian noise term with zero mean and variance σ^2

Kriging

- **Kriging is a "collection of generalized linear regression techniques for minimizing an estimation variance defined from a prior model of covariance" (Olea,1991)**

$$\hat{Z}(s_o) = \sum_{i=1}^{n} \lambda_i Z(s_i)$$

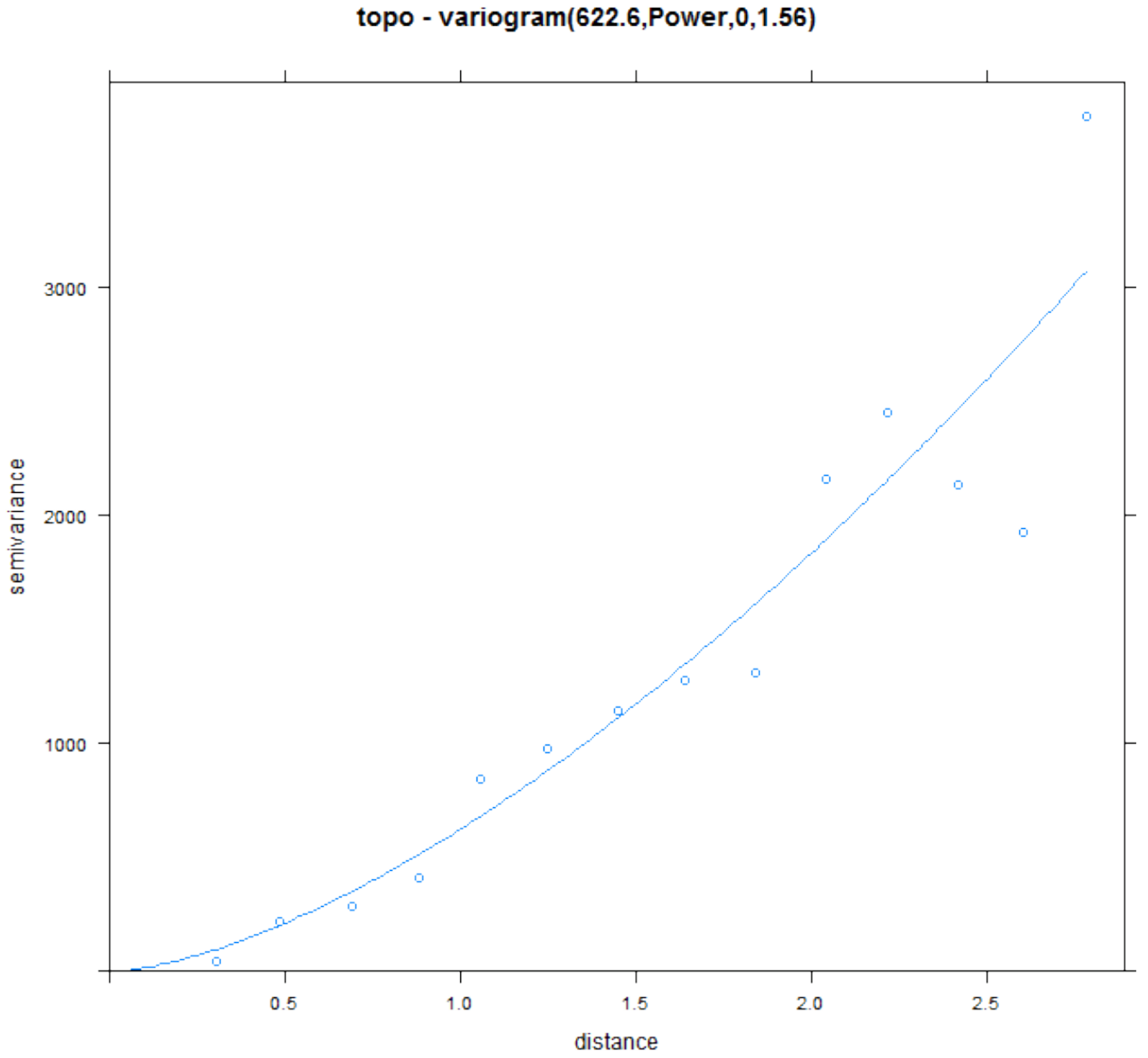
with

$$\sum_{i=1}^{n} \lambda_i = 1$$

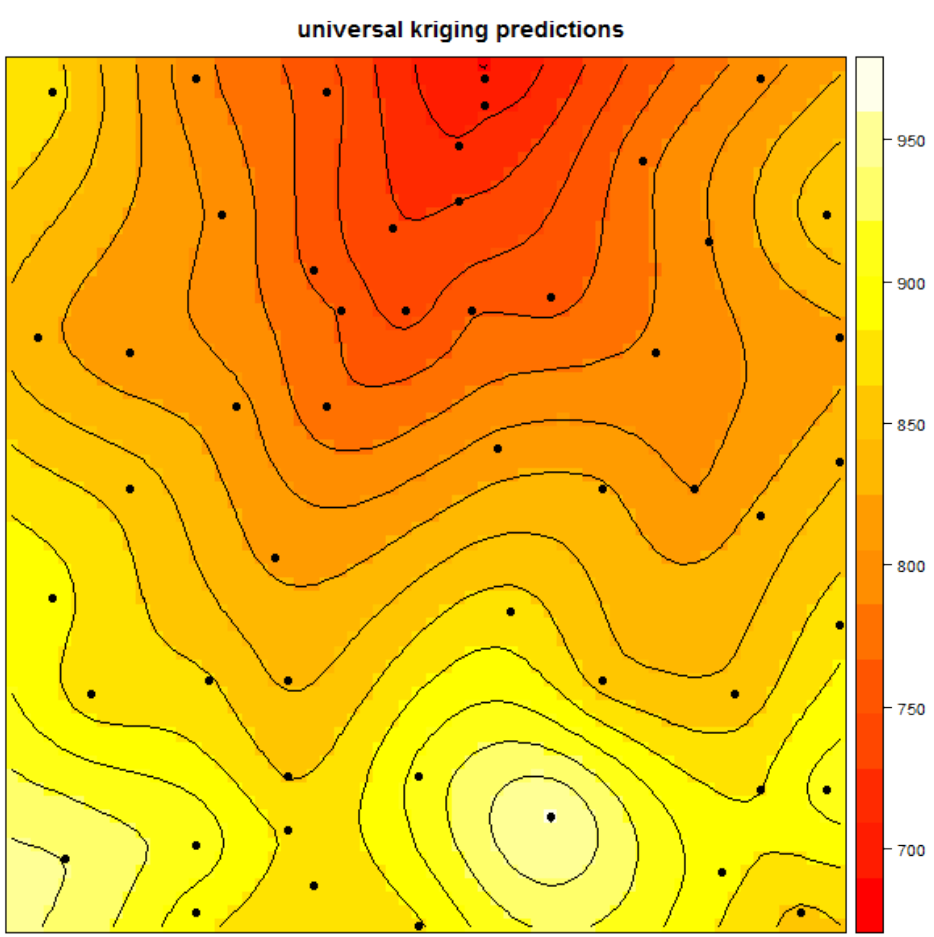
Kriging

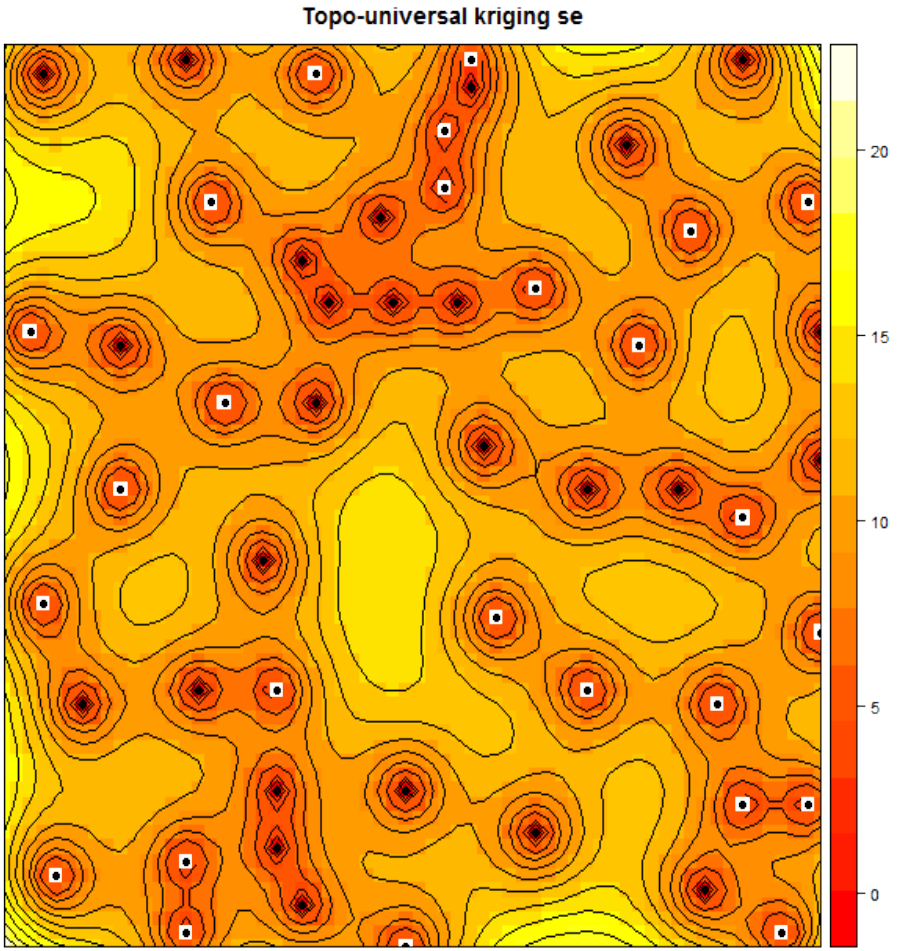
- **Principle**
  - **obtain the best unbiased linear predictor**
  - **takes into account the covariance structure as a function of distance**
- **Best Predictor**
  - **unbiased: $E[z^p - z] = 0$ or no systematic error**
  - **minimum variance of z^p for any other combination of the observed values**
  - **predict for a new locations (s) based on the covariance**

Estimating the variogram as a measure of spatial covariance



Kriging TOPO





Is Kriging Better??

- it provides the no. of points needed to compute the local average
- provides the size, orientation and shape of neighborhood from which the points are drawn
- indicates if there are better ways to estimate the interpolation weights
- its strengths are in the statistical quality of its predictions (e.g.unbiasedness) and in the ability to indicate some measure of the spatial distribution of uncertainty.
- It has been less successful for applications where local geometry and smoothness are the key issues and other methods prove to be competitive or even better

Spatial Simulation

- A stochastic technique of *conditional* simulation is used to generate alternative, equally probable realizations of a surface, reproducing both data and the estimated covariance.
- From such a set of statistical samples one can estimate a spatially-dependent picture of the uncertainty which is inherent in the data.

Simulation is not an instant picture of reality but only a version of it among a myriad of other possibilities. There are two questions that are paramount when doing simulations:

1) How confident can I be of the results?

2) How many simulations should we generate?

Why spatial simulation?

- Recall: the **theory of regionalized variables** assumes that the values we observe come from some **random process**; in the simplest case, with one **expected value** (first-order stationarity) with a **spatially-correlated error** that is the same over the whole area (second-order stationarity).

- So we'd like to see **"alternative realities"**; that is, spatial patterns that, by this theory, *could have* occurred in some "parallel universe" (i.e. another **realization** of the **spatial process**).

- In addition, **kriging maps are unrealistically smooth**, especially in areas with low sampling density.
  - Even if there is a high nugget effect in the variogram, this variability is *not* reflected in adjacent prediction points, since they are estimated from almost the same data.

Goovaerts: "Smooth interpolated maps should not be used for applications sensitive to the presence of **extreme values and their patterns of continuity**." (p. 370)

Example: ground water travel time depends on sequences of large or small values ("critical paths"), not just on individual values.

(from David Rossiter)

Local uncertainty vs. spatial uncertainty

- Recall: kriging prediction also provides a **prediction error**; this is the BLUP and its error **for each prediction location separately**.
- So, at each prediction location we obtain a probability distribution of the prediction, a measure of its **uncertainty**. This is fine for evaluating each prediction individually.
- But, it is *not* valid to evaluate the set of predictions! Errors are *by definition* spatially-correlated (as shown by the fitted variogram model), so we can't simulate the error in a field by simulating the error in each point separately.
- **Spatial uncertainty** is a representation of the error over the **entire field of prediction locations** at the same time.

(from David Rossiter)

Types of Simulation Procedures

- **Simple Monte Carlo method – simulate values for each point which has a unique and an equiprobable value. Since each cell is independent-the results are a stationary noise surface.**

$z(x) = Pr(Z)$ where Z is a spatially independent, normally distributed probability distribution function with mean μ and variance σ^2

Unconditional Simulation – use the variogram to impart spatial information but the actual data points are not used. With the variogram parameters and Monte Carlo procedures the most likely outcome per cell are computed. $z(x) = Pr(Z), \gamma(h)$

Conditional simulation

This simulates the field, while **respecting the sample**.

The simulated maps look more like the best (kriging) prediction, but usually much more spatially-variable (depending on the magnitude of the nugget).

These can be used as inputs into **spatially-explicit models**, e.g. hydrology.

What is preserved in conditional simulation?

1. Mean over field
2. Covariance structure
3. Observed data (points are predicted exactly)

(from David Rossiter)

Conditional simulation

Idea: generate a large set of fields (realizations) that

- honour the data (are conditioned to the data)
- on average, reflect the kriging prediction and variance
- each have a spatial variability, equal to that of the data (in contrast to the kriging prediction map, which is much smoother than the data)
- when to use it? When Z is input to a non-linear model, e.g. some transport or flow model.

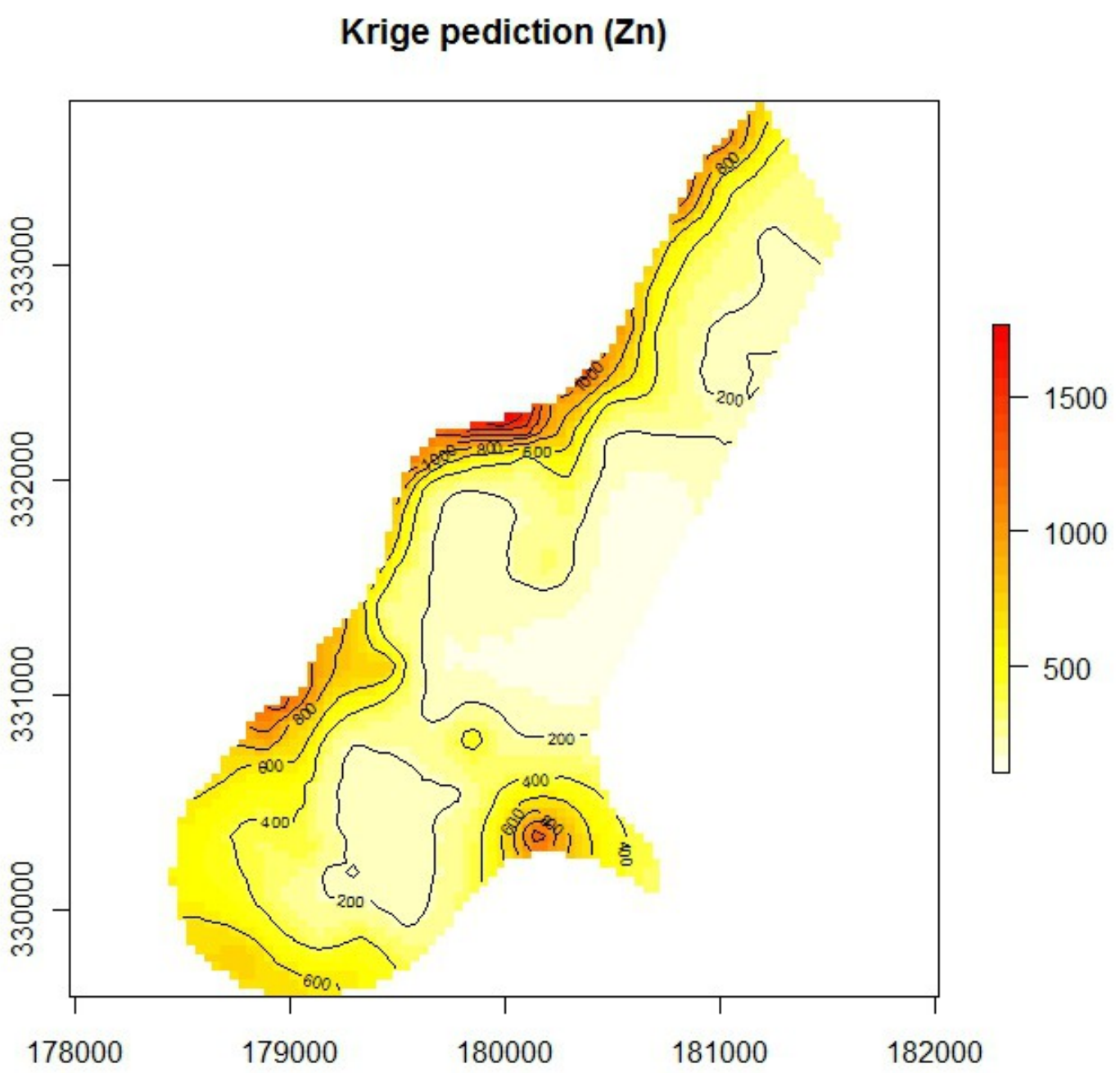
(from Edzer Pebesma)

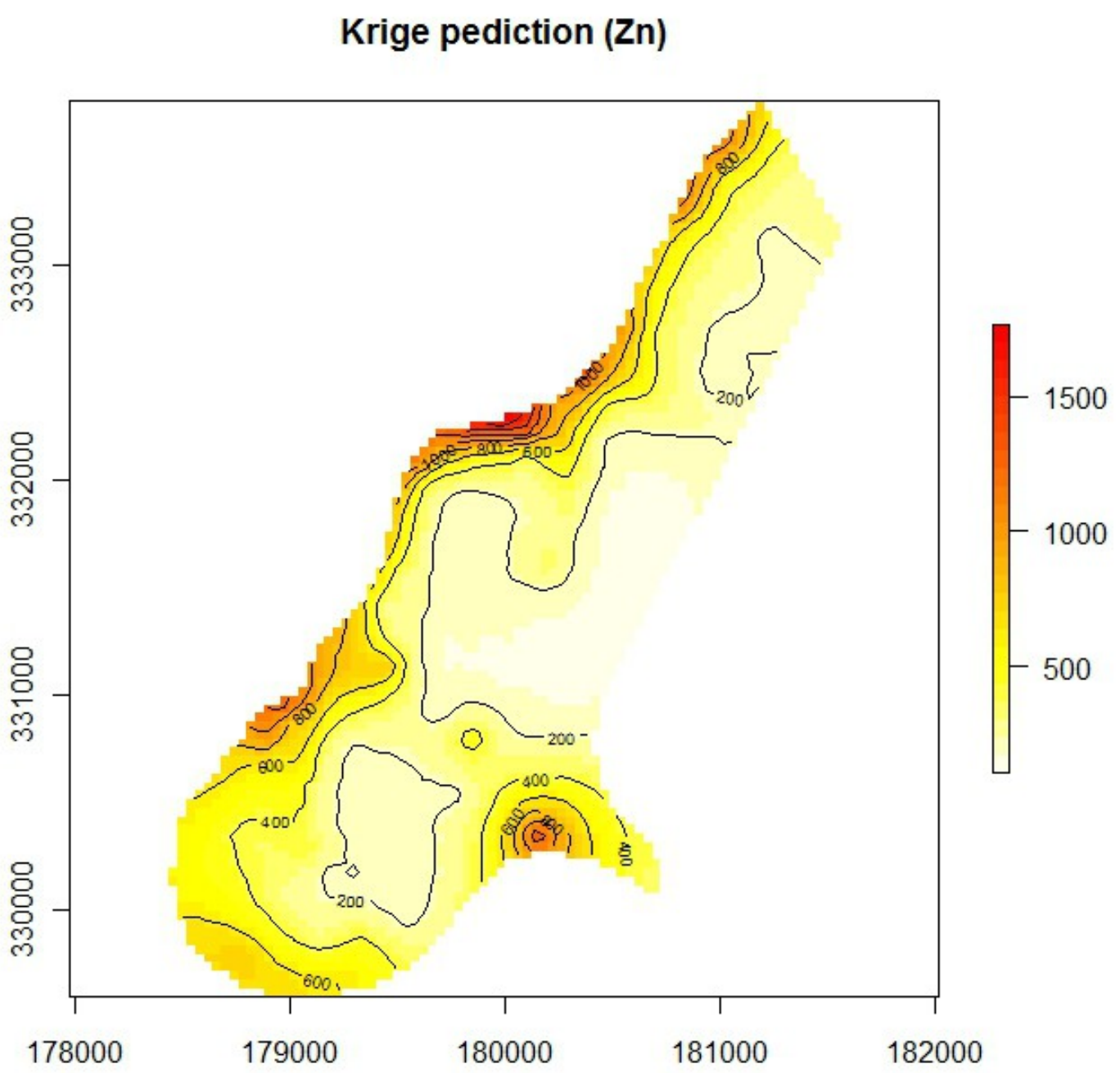
Steps in Simulation Modelling

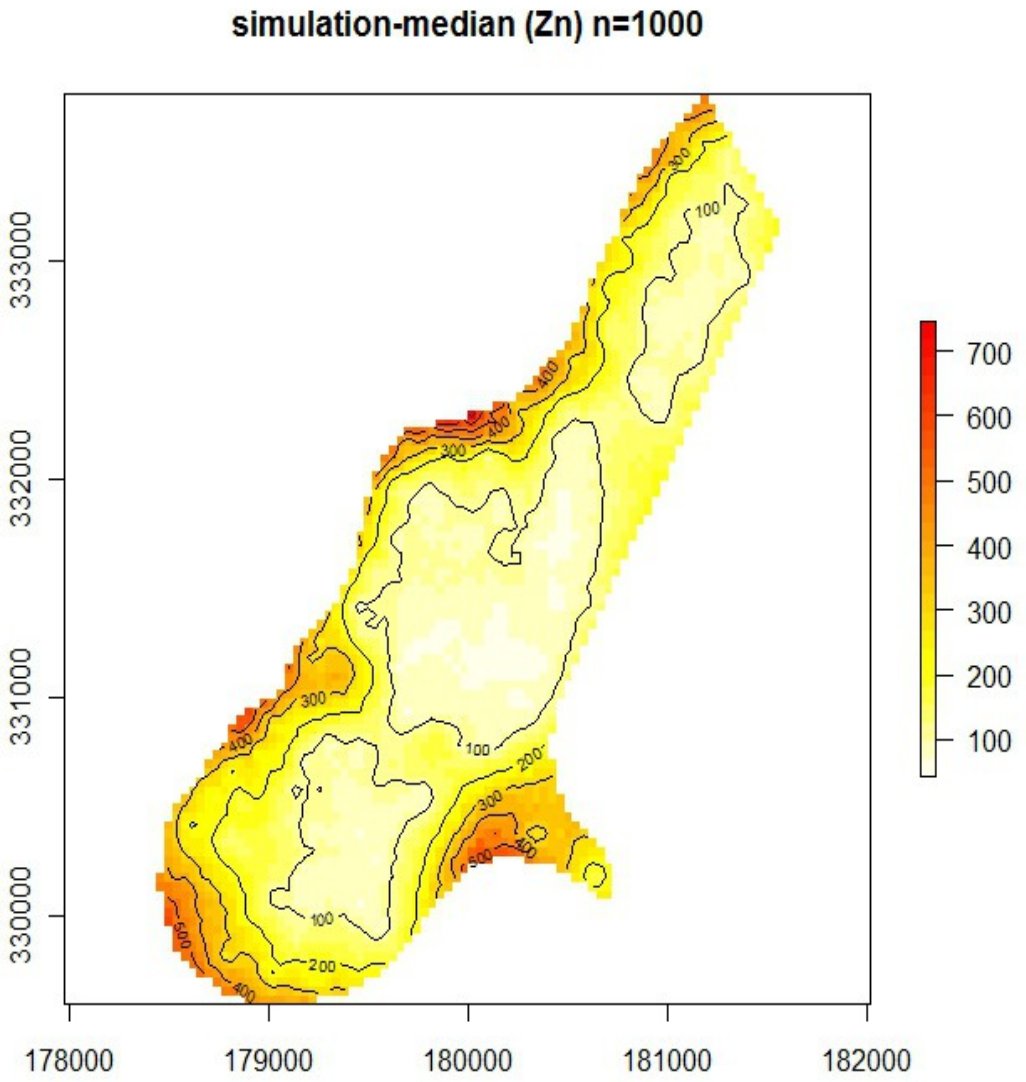
1. Set up the usual equations for simple/ordinary kriging with an overall mean.
2. Select an unsampled data point at random. Compute kriging prediction and standard error using data from the data set in the locality of the point.
3. Draw a random value from the probability distribution defined by the prediction and standard error. Add this to the list of data points.

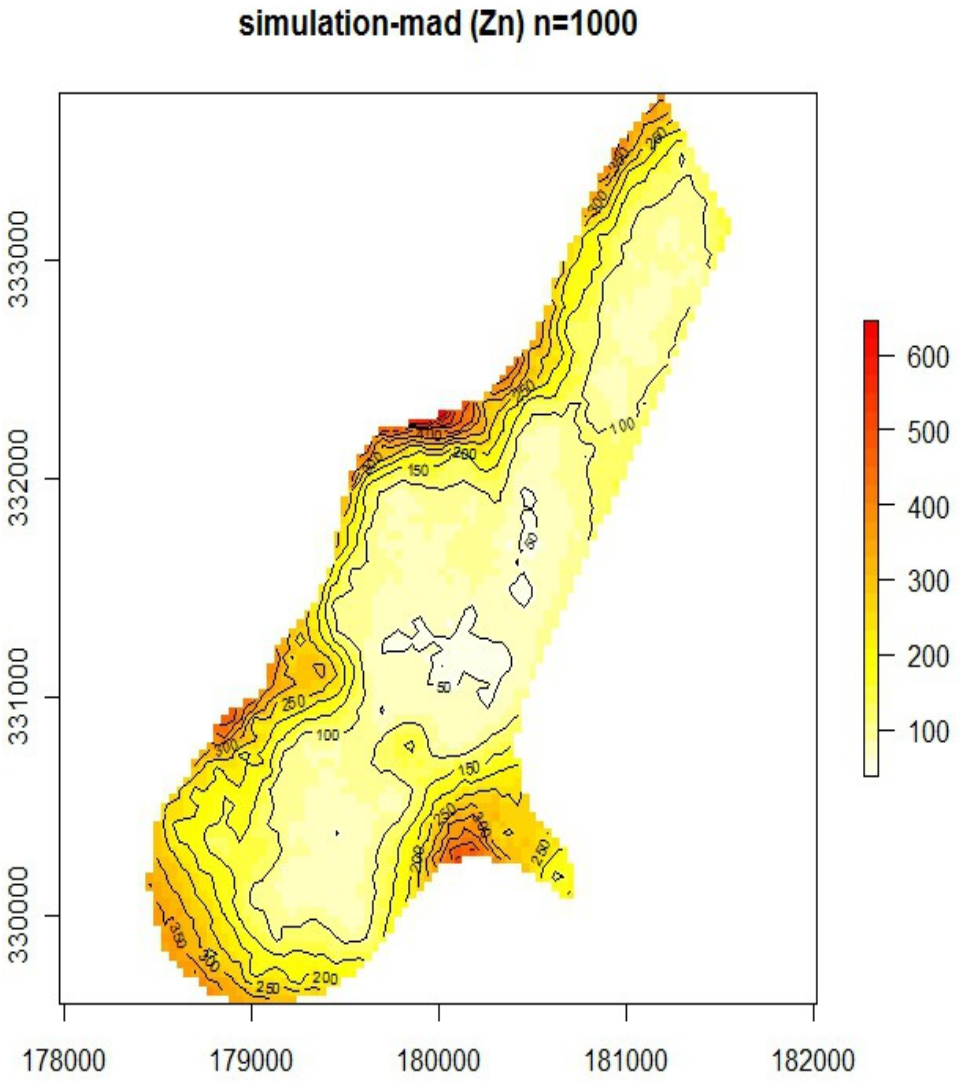
Steps in Simulation (continued)

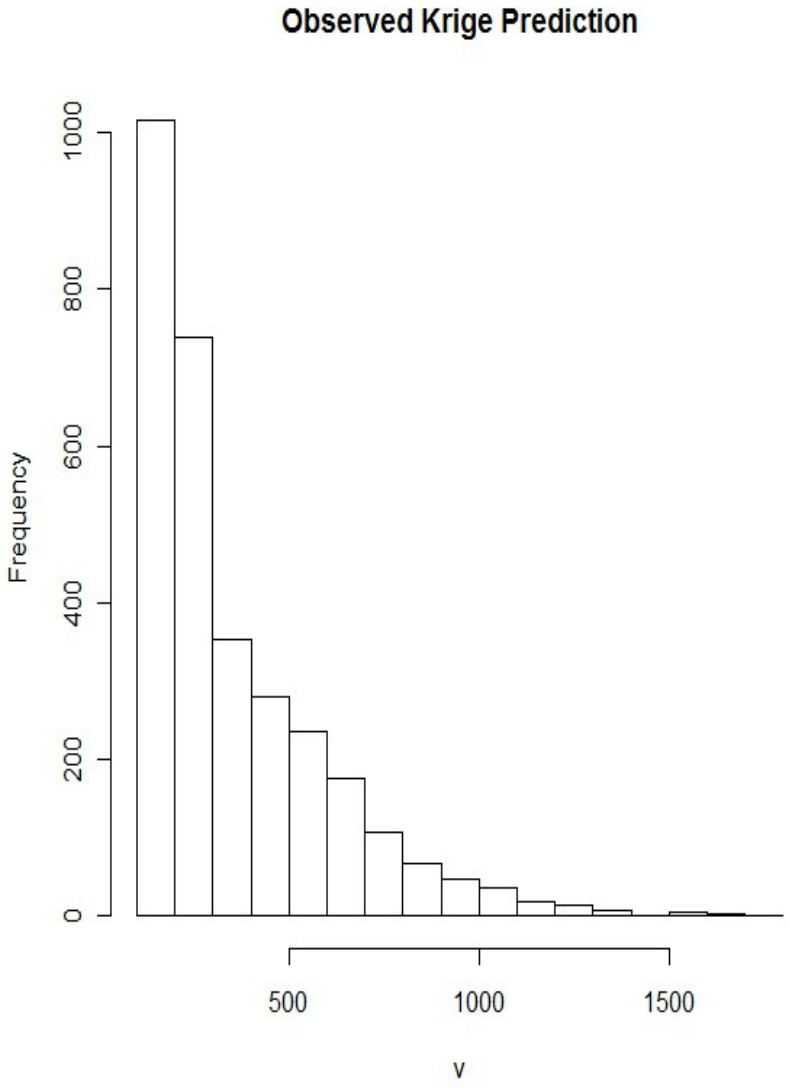
4. Repeat steps 2-3 until all cells/points have been visited and the simulation of one realization is completed.
5. Repeat steps 1-4 until “sufficient” realizations have been created.
6. Compute surfaces for the mean values and standard deviation from all the realizations.

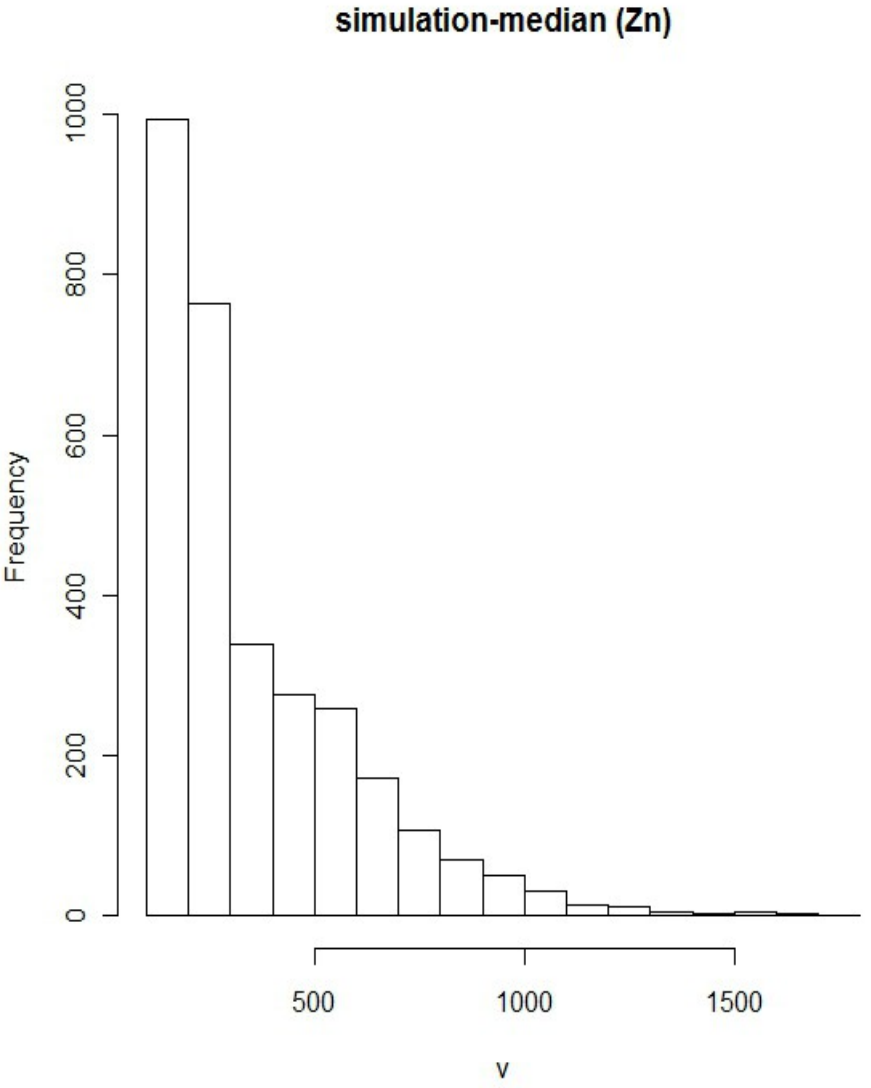

Krige pediction (Zn)
333000
332000
331000
330000
178000
179000
180000
181000
182000
1500
1000
500
200
400
600
800
1000


Krige pediction (Zn)
333000
332000
331000
330000
178000
179000
180000
181000
182000
1500
1000
500
200
400
600
800
1000


simulation-median (Zn) n=1000
333000
332000
331000
330000
178000
179000
180000
181000
182000
700
600
500
400
300
200
100

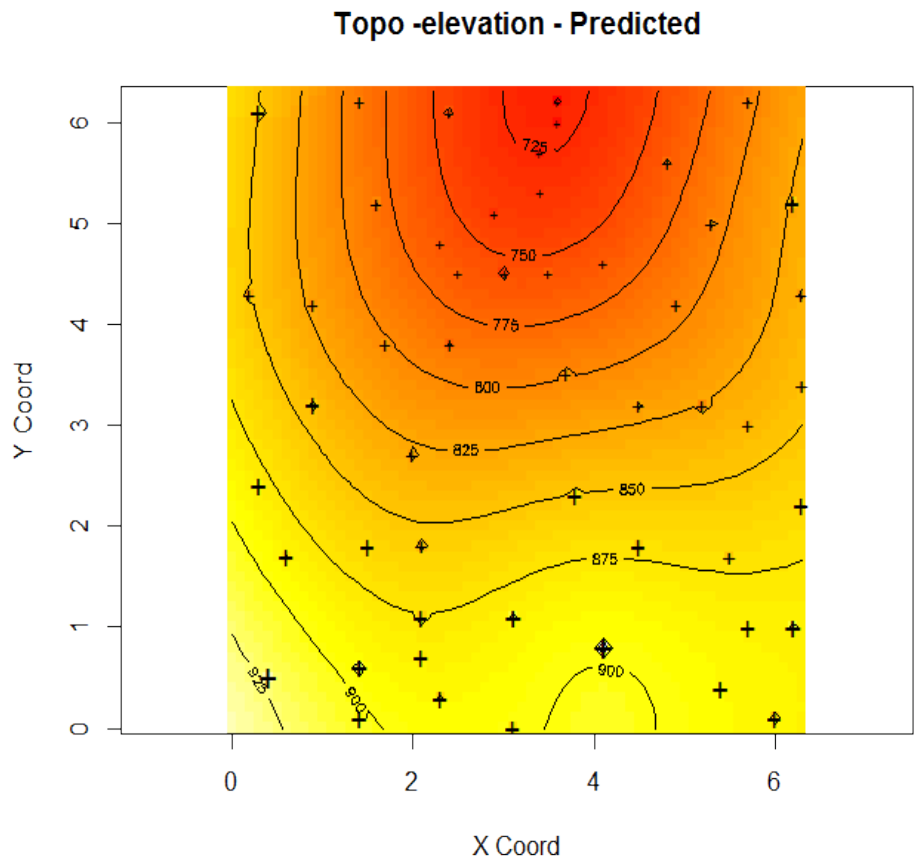

simulation-mad (Zn) n=1000
333000
332000
331000
330000
178000
179000
180000
181000
182000
600
500
400
300
200
100

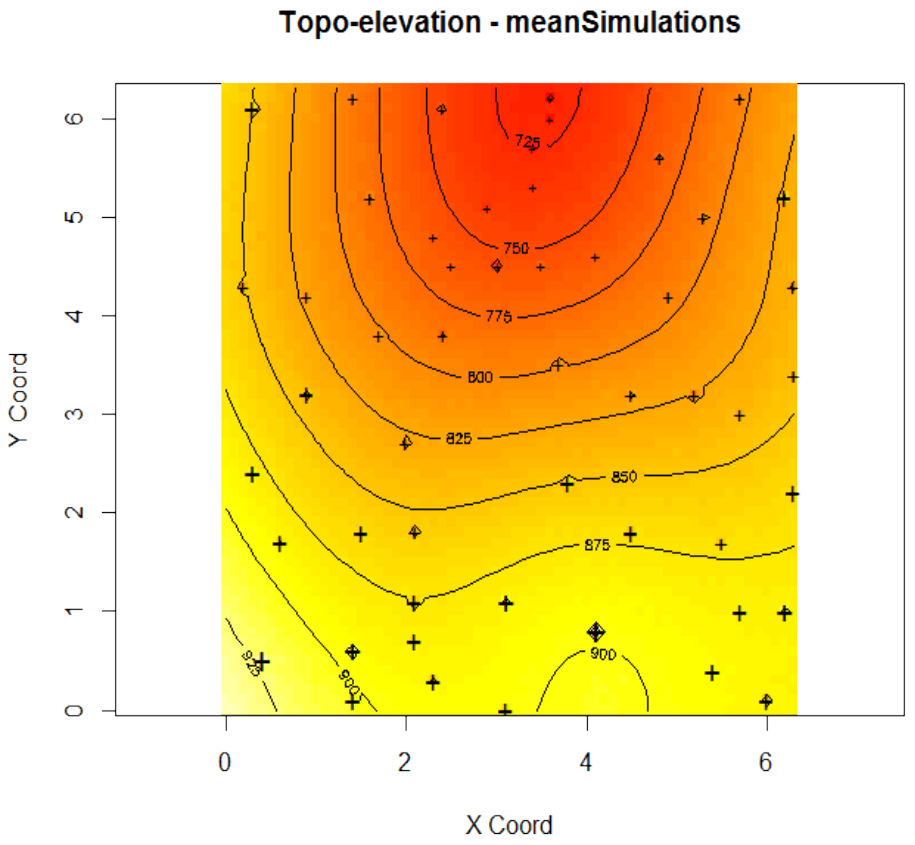

Observed Krige Prediction
Frequency
0
200
400
600
800
1000
500
1000
1500
v

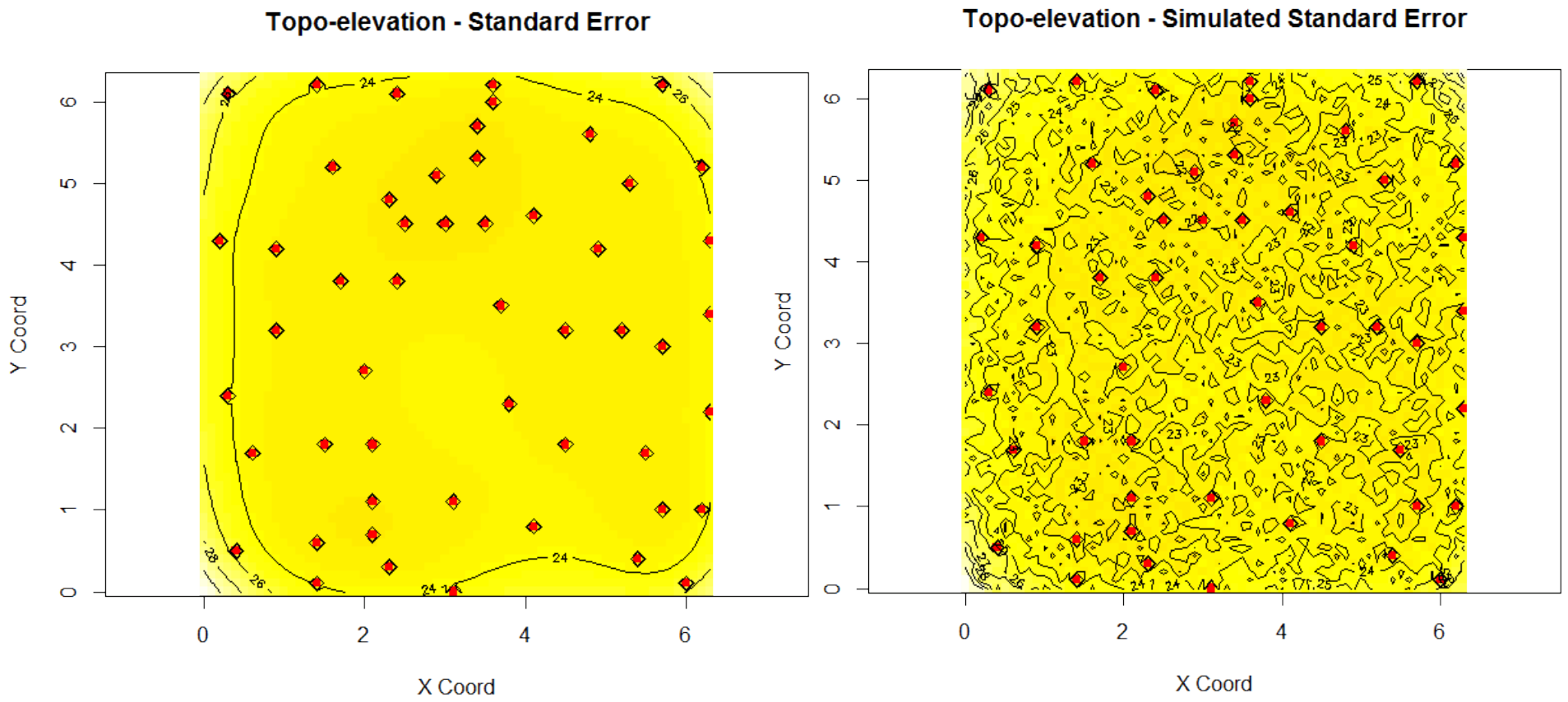

simulation-median (Zn)
Frequency
0
200
400
600
800
1000
500
1000
1500
v

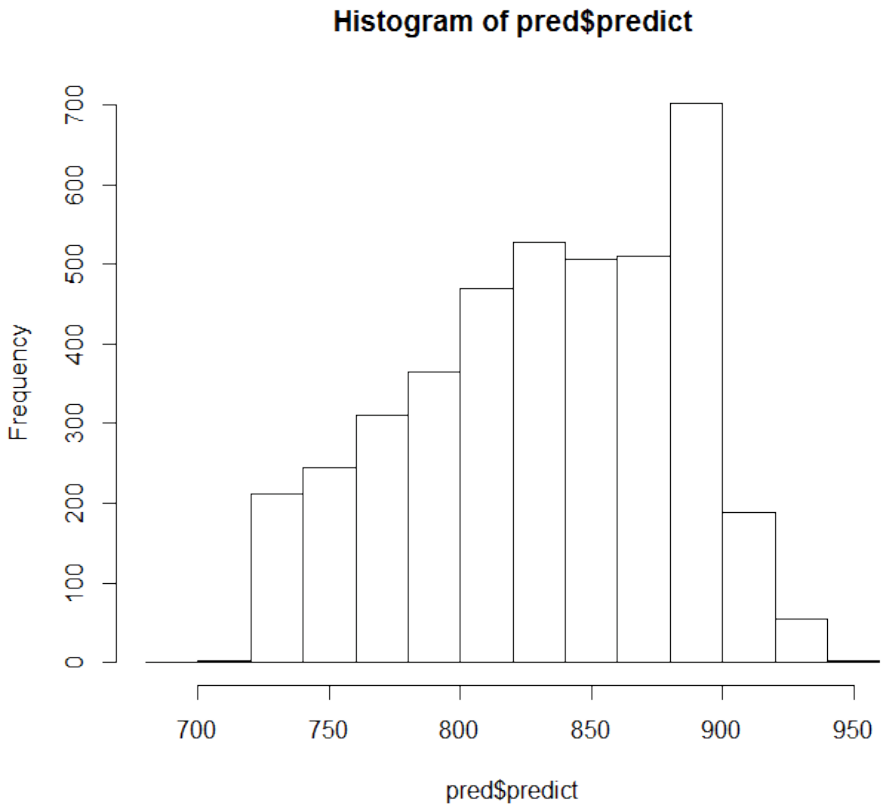
Conditional Simulation in gstat with use of raster

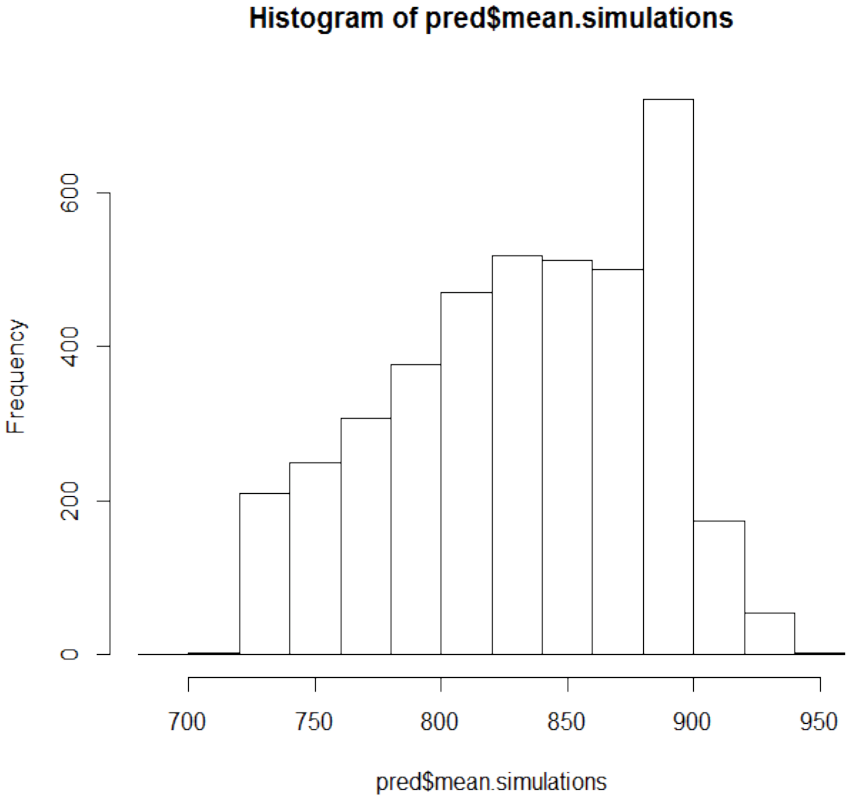
```
require(raster)
set.seed(234)
.
.
v = variogram(log(zinc)~1, meuse)
v.m = fit.variogram(v, vgm(1, "Sph", 900, 1))
tsr <- 1000
sim = krige(log(zinc)~1, meuse, meuse.grid, v.m, nsim = tsr,
      nmax = 20)
s <- brick(sim)
rs2 <- calc(s, fun=function(x){exp(x)}) #backtransform log data
b1 <- stackApply(rs2, indices=c(rep(1,tsr)),
      fun=median, progress="window")
plot(b1,col=rev(heat.colors(25)),main="simulation-median (Zn)
     n=1000")
contour(b1, add=TRUE)
hist(b1,main=("simulation-median (Zn)"))
b2 <- stackApply(rs2, indices=c(rep(1,tsr)),
      fun=mad, progress="window")
plot(b2,col=rev(heat.colors(25)),main="simulation-mad (Zn)
     n=1000")
contour(b2, add=TRUE)
```


Topo -elevation - Predicted
Y Coord
X Coord
0
1
2
3
4
5
6
725
750
775
800
825
850
875
900
925


Topo-elevation - meanSimulations
Y Coord
X Coord
0
1
2
3
4
5
6
725
750
775
800
825
850
875
900
925


Topo-elevation - Standard Error
Y Coord
X Coord
0
1
2
3
4
5
6
24
26
28
Topo-elevation - Simulated Standard Error
Y Coord
X Coord
0
2
4
6
23
24
25
26


Histogram of pred$predict
Frequency
0
100
200
300
400
500
600
700
700
750
800
850
900
950
pred$predict


Histogram of pred$mean.simulations
Frequency
0
200
400
600
700
750
800
850
900
950
pred$mean.simulations

Model Verification

- Regression diagnostics
- Cross Validation
- AIC

AIC

Akaike (1973, 1974) introduced a criterion for model adequacy, first for time-series models and then more generally. He relates how his secretary suggested he call it 'An Information Criterion', AIC.

AIC

AIC has a very appealing simplicity:

$$AIC = -2 \log(\text{maximized likelihood}) + 2p$$

where p is the number of estimated parameters.

Choose the model with the smallest AIC (and perhaps retain all models within 2 of the minimum).

Cross-validation

Leave-one-out CV

The idea is that given a dataset of *N* points, we use our model-building procedure on each subset of size *N*-1, and predict the point we left out. Then the set of predictions can be summarized by some measure of prediction accuracy. Idea goes back at least as far as Mosteller & Wallace (1963).

Leave one out

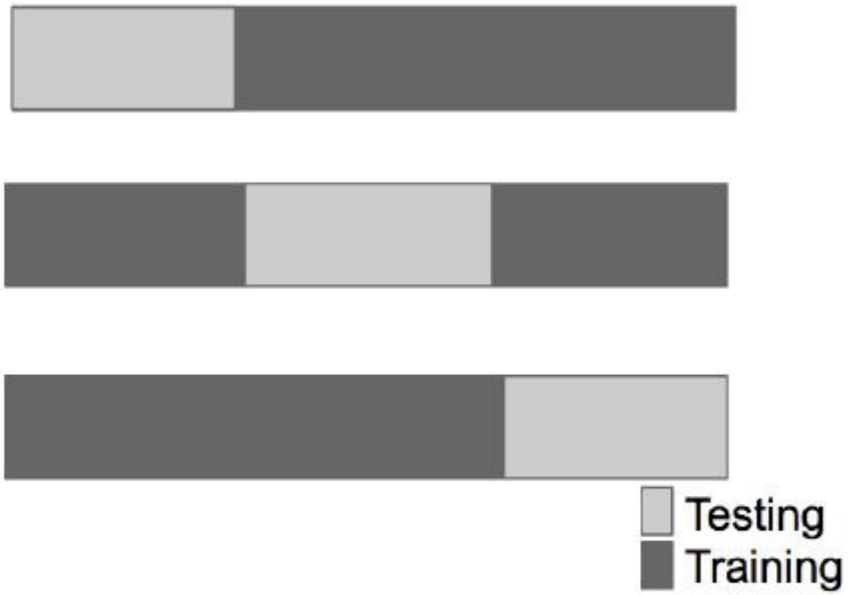
Users/jtleek/Dropbox/Public/courseraPublic/week6/002crossValidation/index.html#1

V-fold cross-validation

- Divide the data into *V* sets, and amalgamate *V* - 1 of them, build a model and predict the result for the remaining set. Do this *V* times leaving a different set out each time.

- How big should *V* be? We want the model-building problem to be realistic, so want to leave out a small proportion. We don't want too much work. So usually *V* is 3–10.

K-fold



file:///Users/jtleek/Dropbox/Public/courseraPublic/week6/002crossValidation/index.html#1

Cross validation statistics

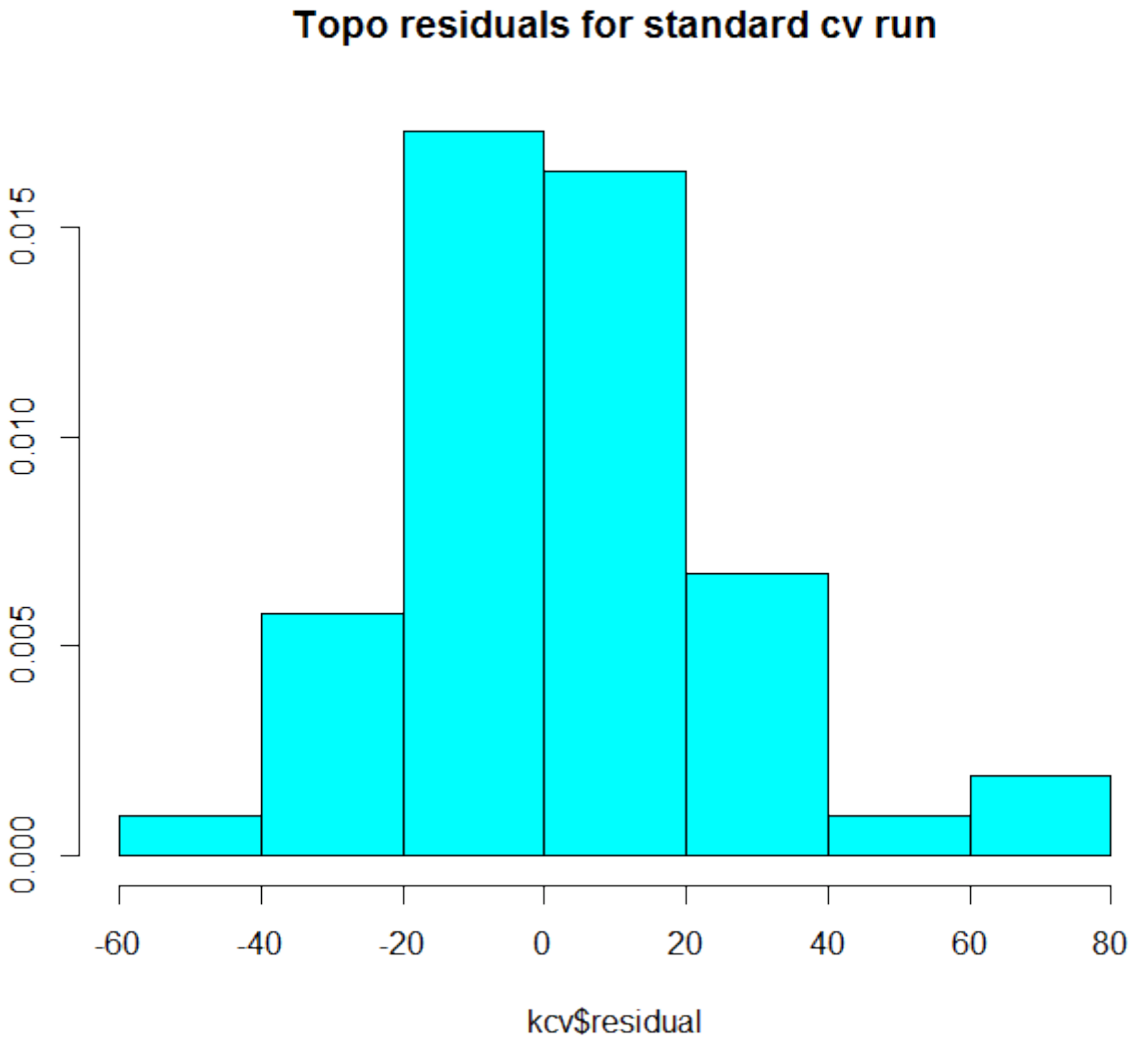
Ideally

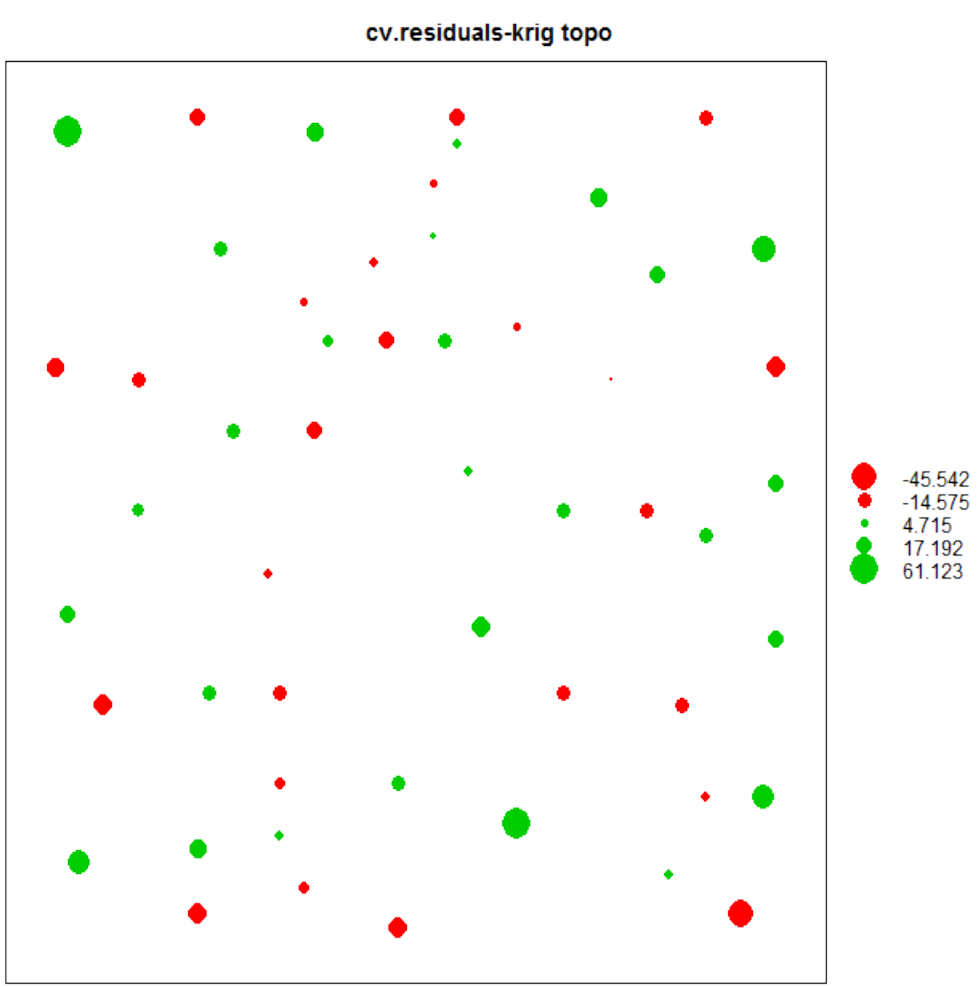
- the correlation between $Z(s_i)$ and $\hat{Z}_{[i]}(s_i)$ should be close to 1
- the variability of $\hat{Z}_{[i]}(s_i)$ should be close to that of $Z(s_i)$ (but will always be smaller: the *smoothing effect*)
- residuals should have zero mean, and small variance (small range, etc.)
- z-scores should contain no outliers (values outside, say $[-3, 3]$)
- z-scores should have unit standard deviation (only this "validates" the prediction error standard deviation).

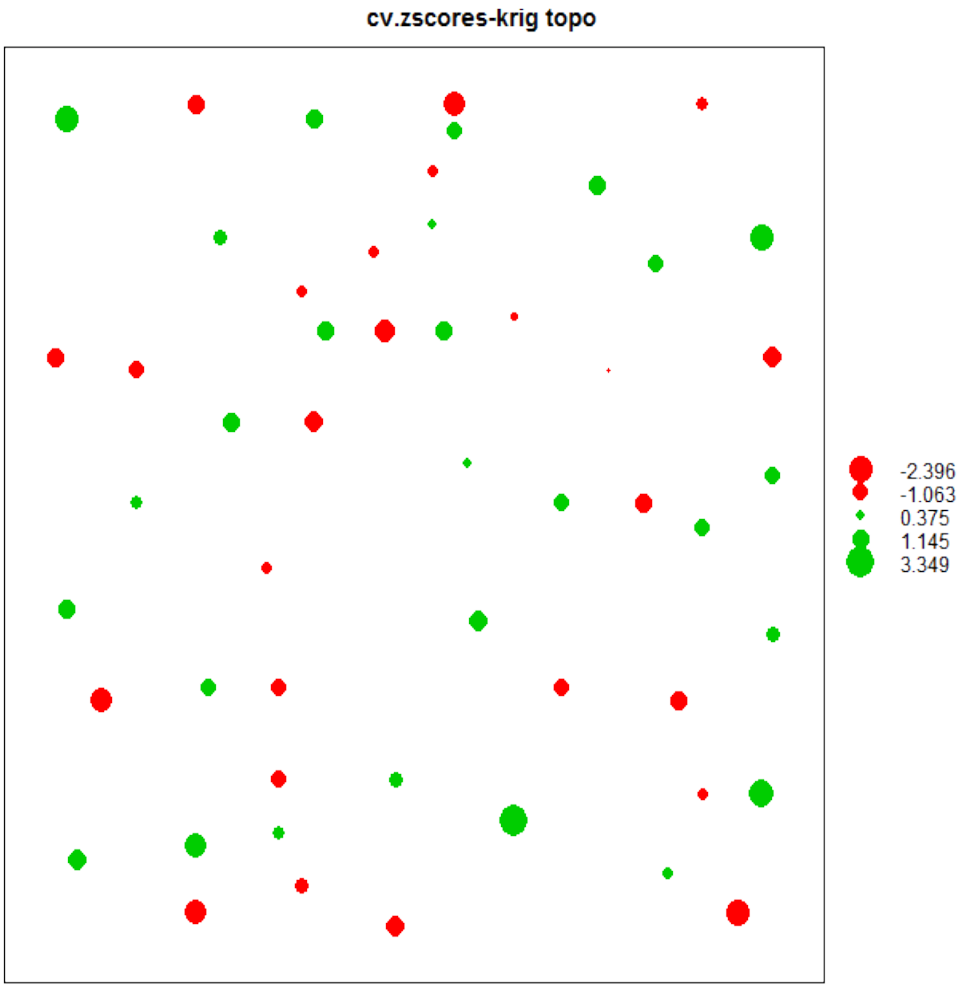
(from Edzer Pebesma)

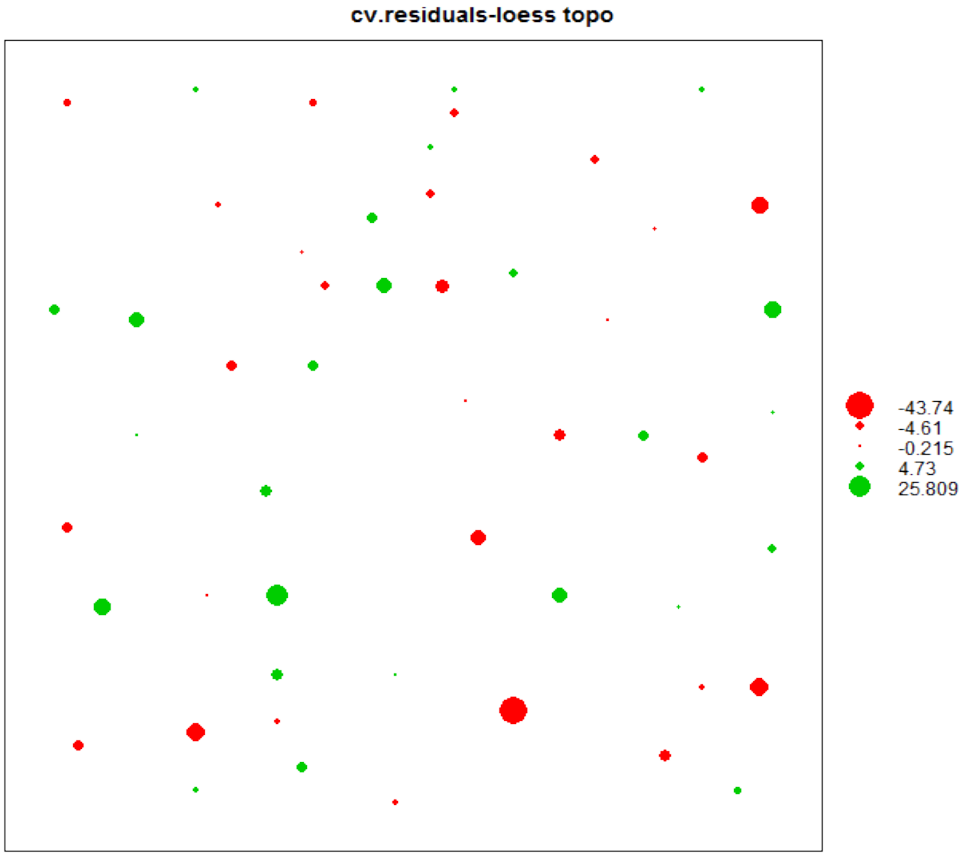
- calculate the residual $Z(s_i) - \hat{Z}_{[i]}(s_i)$ and the normalized residuals, or z-score,

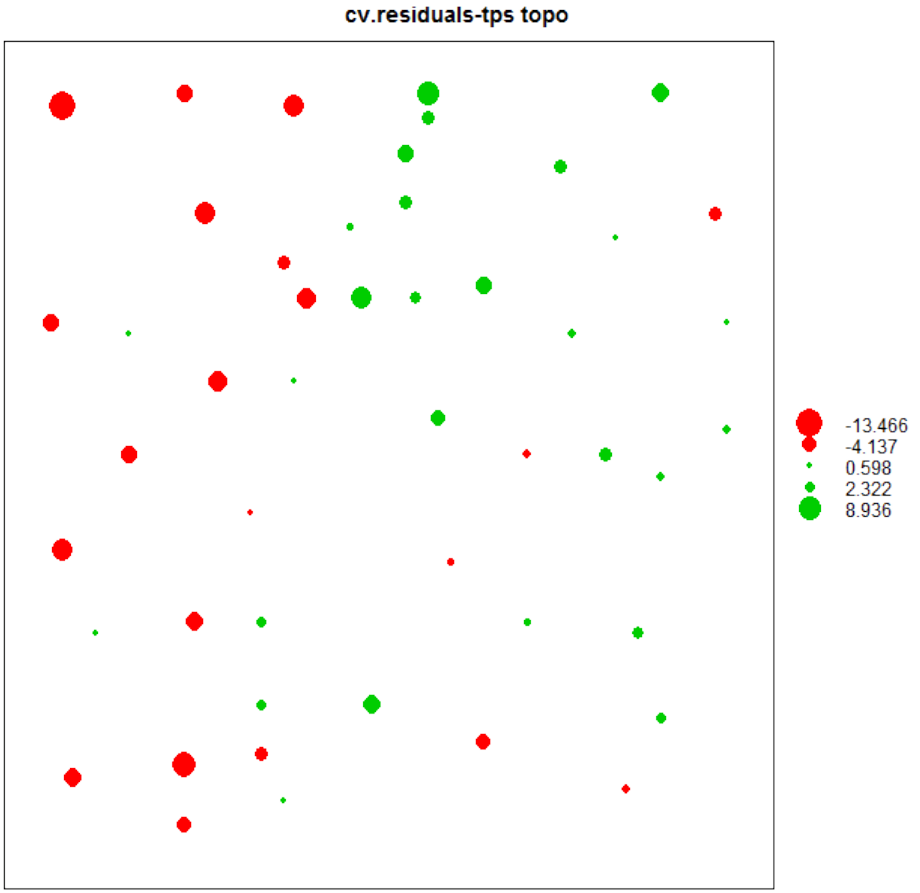
$$\frac{Z(s_i) - \hat{Z}_{[i]}(s_i)}{\sigma(s_i)}$$

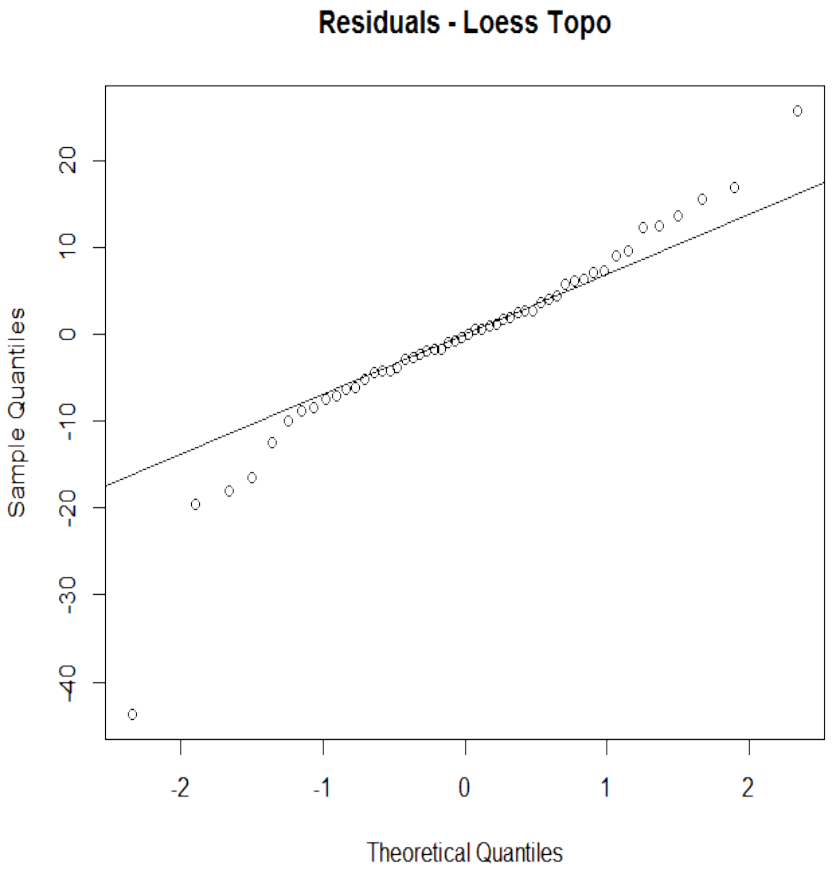

Topo residuals for standard cv run
0.000
0.005
0.010
0.015
-60
-40
-20
0
20
40
60
80
kcv$residual

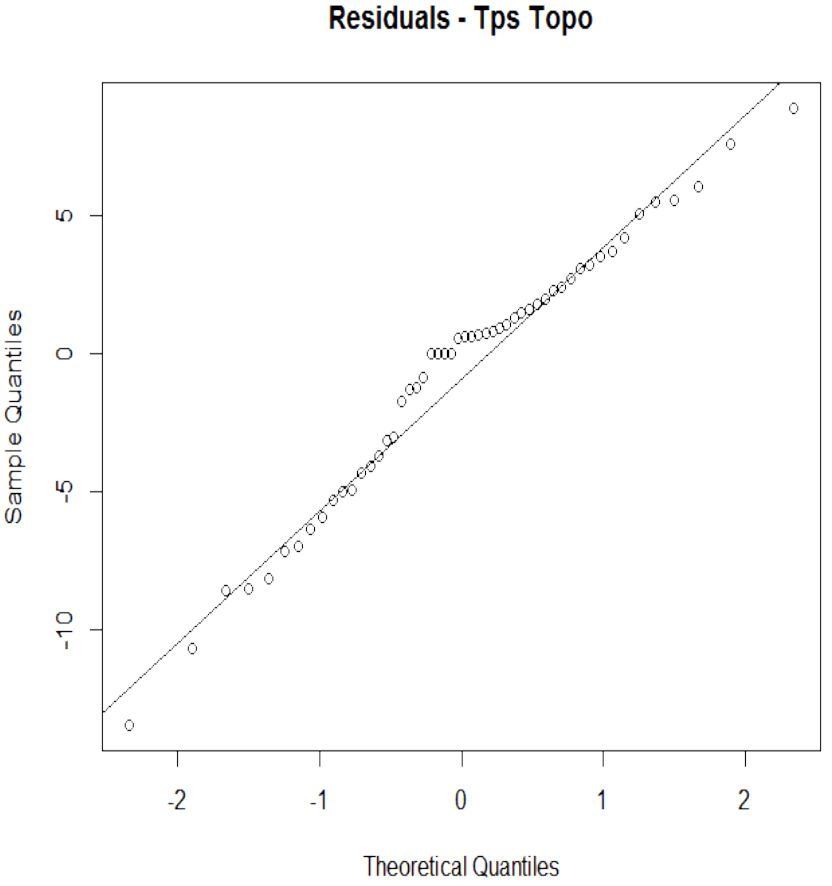

cv.residuals-krig topo
-45.542
-14.575
4.715
17.192
61.123


cv.zscores-krig topo
-2.396
-1.063
0.375
1.145
3.349


cv.residuals-loess topo
-43.74
-4.61
-0.215
4.73
25.809


cv.residuals-tps topo
-13.466
-4.137
0.598
2.322
8.936


Residuals - Loess Topo
Sample Quantiles
Theoretical Quantiles
20
10
0
-10
-20
-30
-40
-2
-1
0
1
2


Residuals - Tps Topo
Sample Quantiles
Theoretical Quantiles
5
0
-5
-10
-2
-1
0
1
2

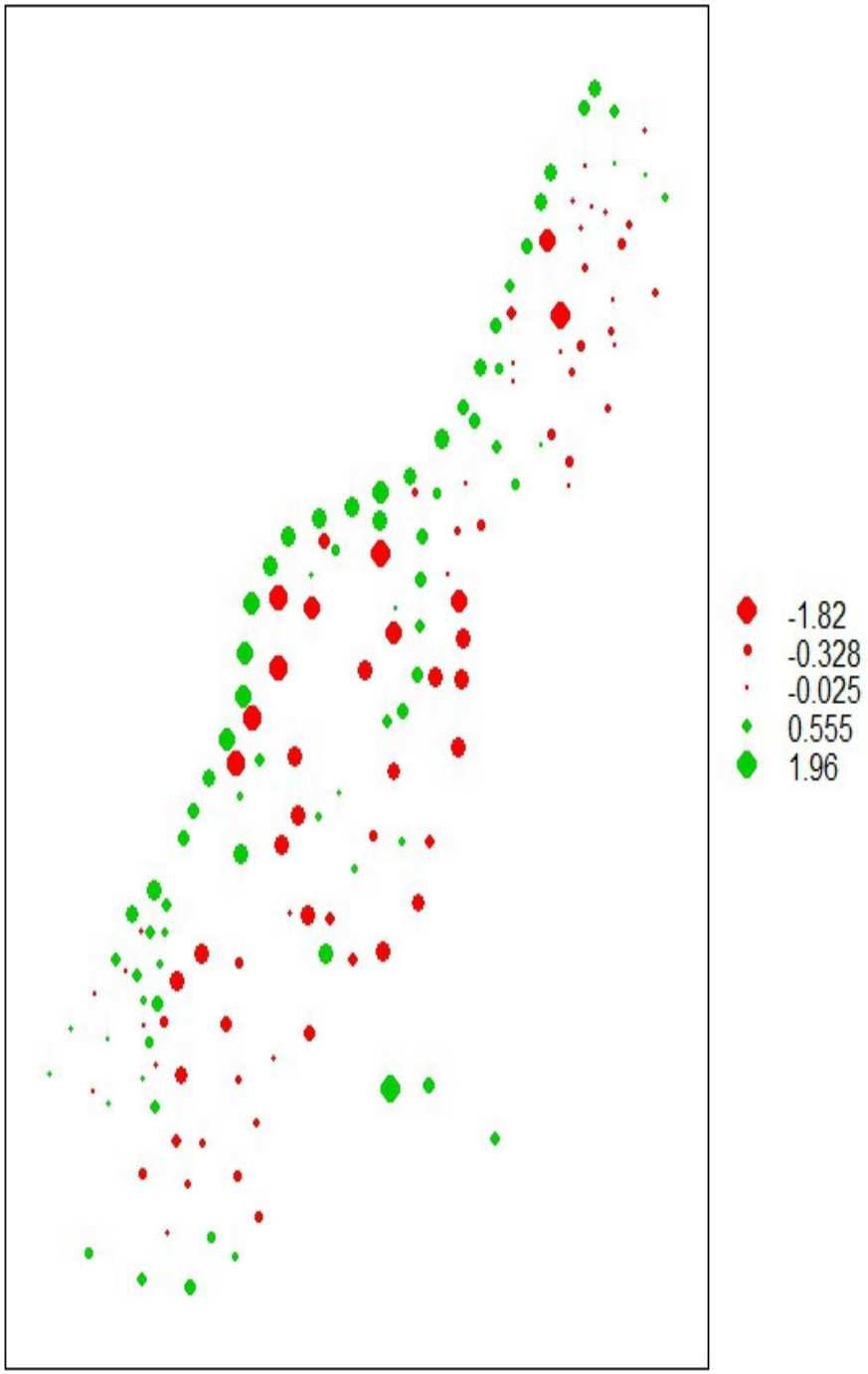
Cross Validation Statistics for TOPO

```
# mean error, ideally 0:
> (mpe <- mean(kcv$residual))
[1] 3.214332
> mpe/mean(topo$z)#standardize to the data mean
[1] 0.003886376
> # MSPE, ideally small
> mean(kcv$residual^2)
[1] 507.5754
> # Mean square normalized error, ideally close to 1
> mean(kcv$zscore^2)
[1] 1.876445
> #RMSE
> (rmse <- sqrt(mean(kcv$residual^2)))
[1] 22.52944
> rmse/sd(topo$z)
[1] 0.3633912
> rmse/IQR(topo$z)
[1] 0.2635022
> # amount of variation (R^2) explained by the models:
> 1-var(kcv$residual, na.rm=T)/var(topo$z)
[1] 0.8680982
> # correlation predicted and residual, ideally 0
> cor(kcv$observed - kcv$residual, kcv$residual)
[1] -0.06767013
```

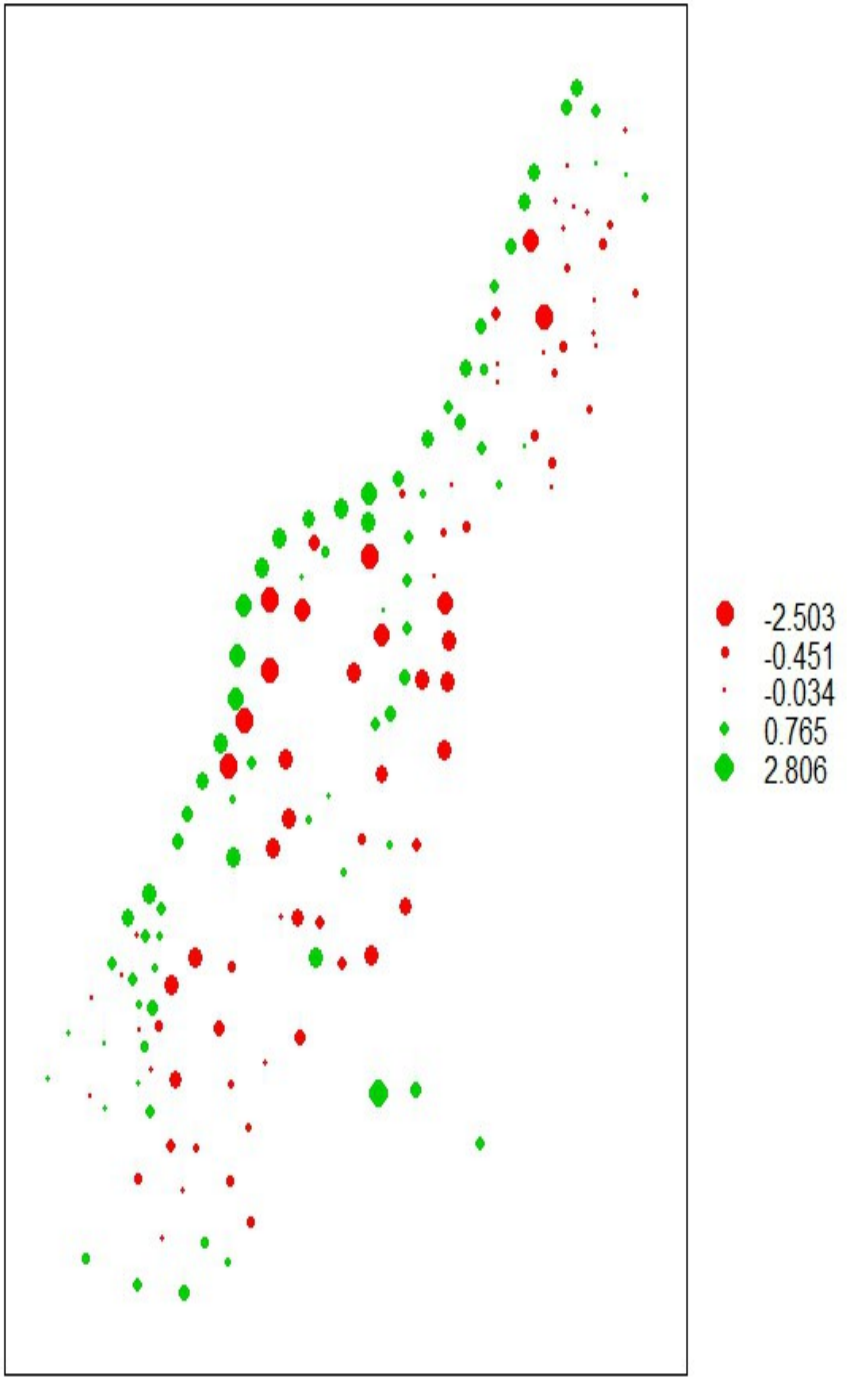
Comparison of TOPO model statistics

	loess	Tps	UK.cv nfold=10
ME	-0.48	-0.69	1.16
RMSE	10.52	4.80	22.0
RMSE/sd	0.17	0.08	0.87
R^2	0.97	0.99	0.87

log(cadmium): 5-fold Residuals Cross Validation-Z-Scores

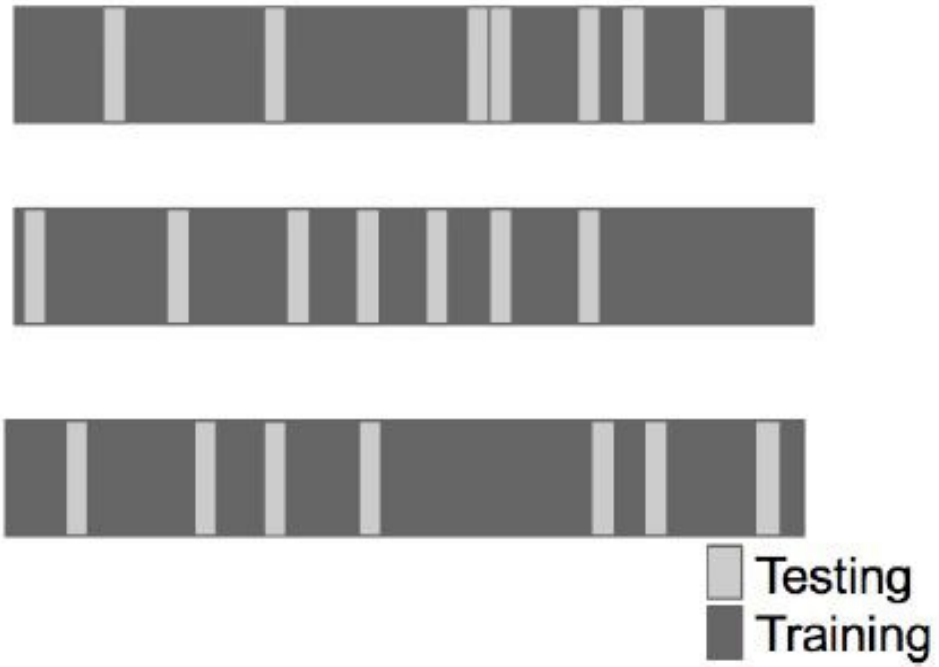


log(cadmium): 5-fold Residuals Cross Validation-Residuals



```
### 5-fold Cross Validation
##################################################
v.fit <- vgm(.5, "Exp", 790, 1.7)
coordinates(meuse) <- ~x+y
cv155 <- krige.cv(log(cadmium)~1, meuse, v.fit,
    nfold=5)
##########################################################
bubble(cv155, "residual", main = "log(cadmium): 5-fold
   Residuals Cross Validation-Residuals",maxsize = 1.5)
bubble(cv155, "zscore", main = "log(cadmium): 5-fold
   Residuals Cross Validation-Z-Scores",maxsize = 1.5)
```

Random subsampling



file:///Users/jtleek/Dropbox/Public/courseraPublic/week6/002crossValidation/index.html#1

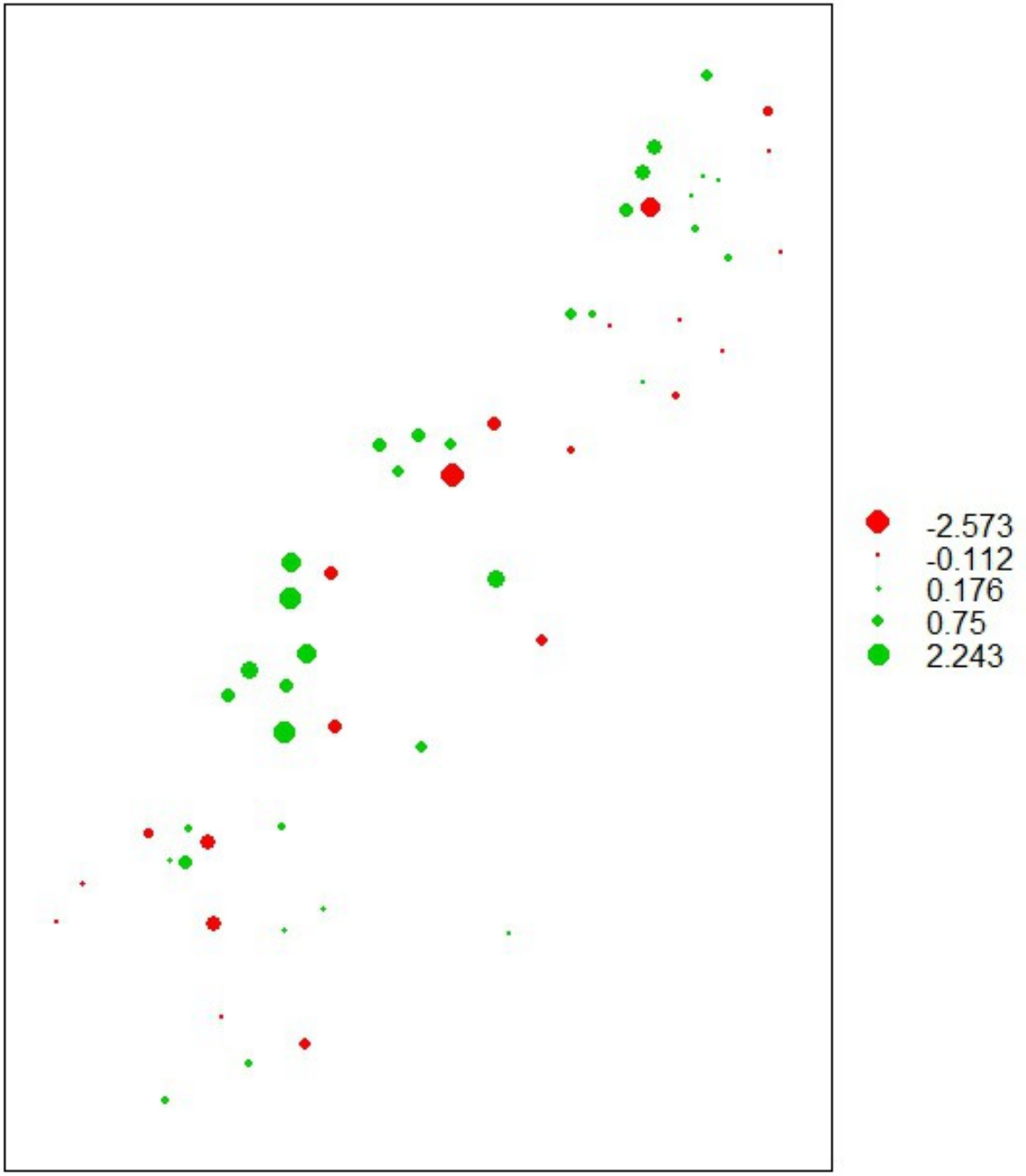
Validating Kriging Model by subsampling

```
set.seed(1357531)
#########################################
sel100 <- sample(1:155, 100)
m.model <- meuse[sel100,]
m.valid <- meuse[-sel100,]
coordinates(m.model) <- ~ x+y
coordinates(m.valid) <- ~ x+y
v100.fit <- fit.variogram(variogram(log(cadmium)~1,
   m.model), vgm(.5, "Exp",800, 1.7))
m.valid.pr <- krige(log(cadmium)~1, m.model, m.valid,
   v100.fit) # predict at #m.valid sites

resid.kr <- log(m.valid$cadmium)- m.valid.pr$var1.pred
# residuals
summary(resid.kr)
resid.mean <- log(m.valid$cadmium) -
mean(log(m.valid$cadmium))
R2 <- 1 - sum(resid.kr^2)/sum(resid.mean^2)
```

R^2 = 0.34

Residuals on 55 sample validation data-Cadmium

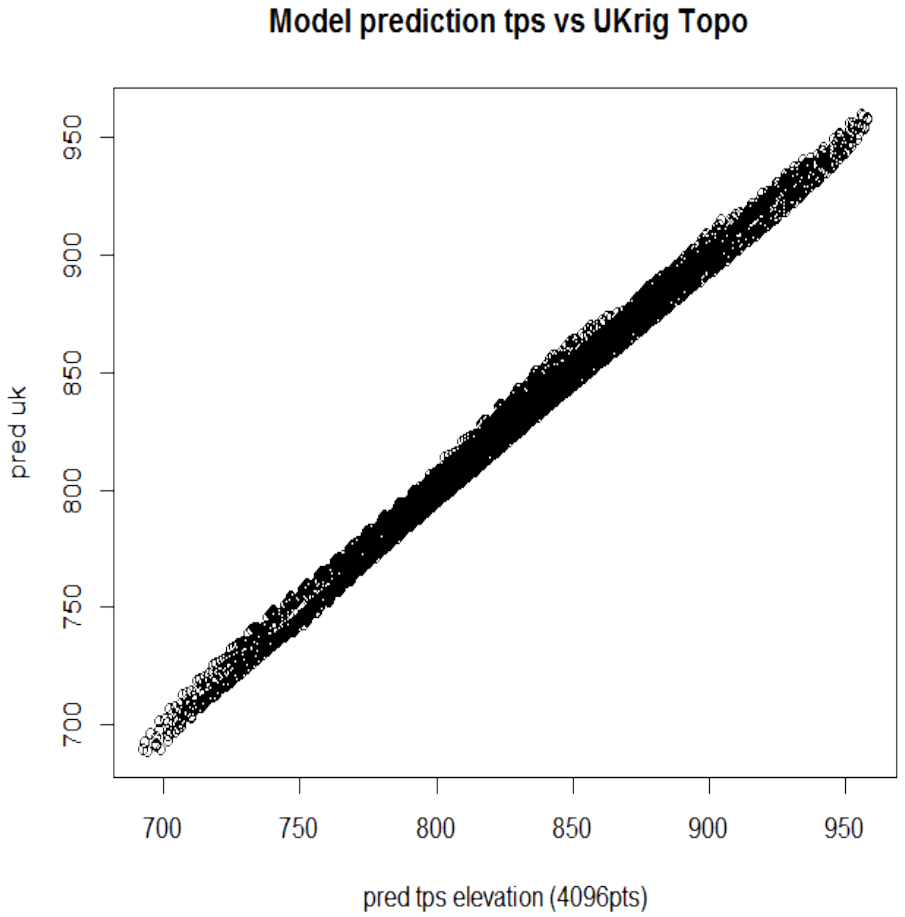


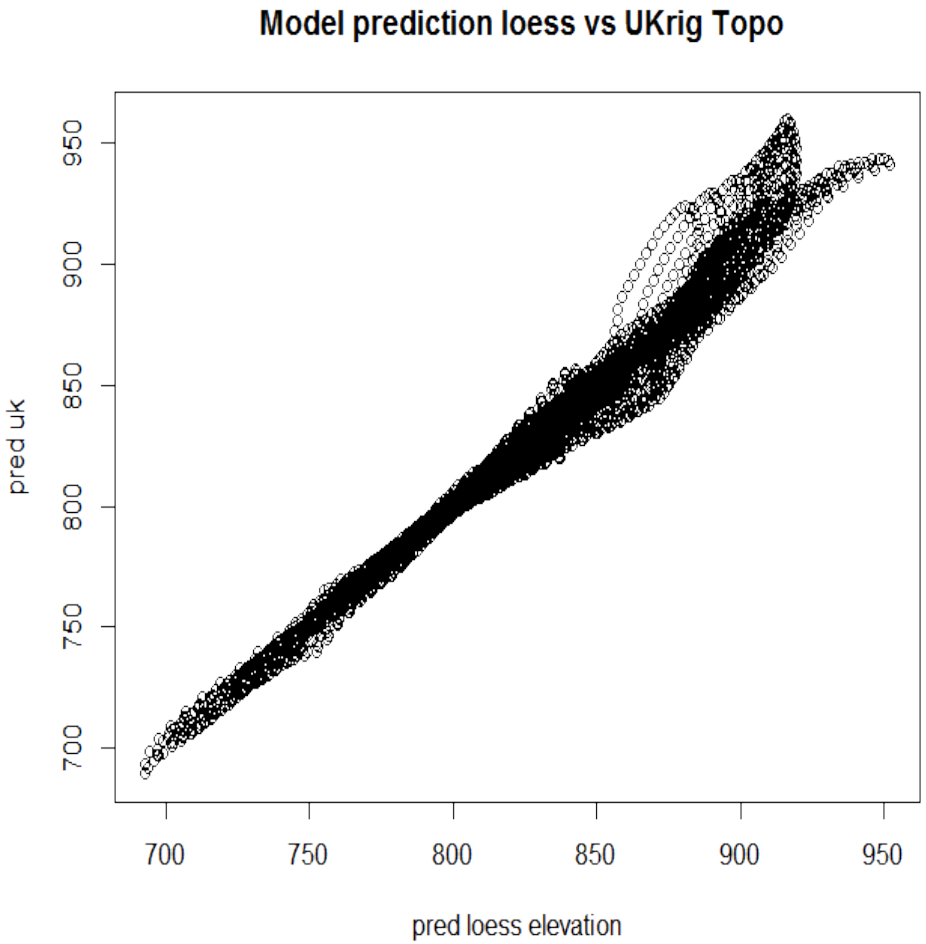
CROSS VALIDATION

- When in prediction mode (i.e., kriging), cross validating the model is one way to diagnose problems occurring with the model's fit.

- The purpose of checking the modeled variogram with cross validation is to expose obvious mistakes, and to view where the model has possible trouble predicting.

- It does not prove the model is right; it only shows where the model fails. Likewise, it is not recommended to confirm the correctness of different model types, e.g. spherical versus power models.


Model prediction tps vs UKrig Topo
pred uk
700
750
800
850
900
950
pred tps elevation (4096pts)


Model prediction loess vs UKrig Topo
pred uk
700
750
800
850
900
950
pred loess elevation

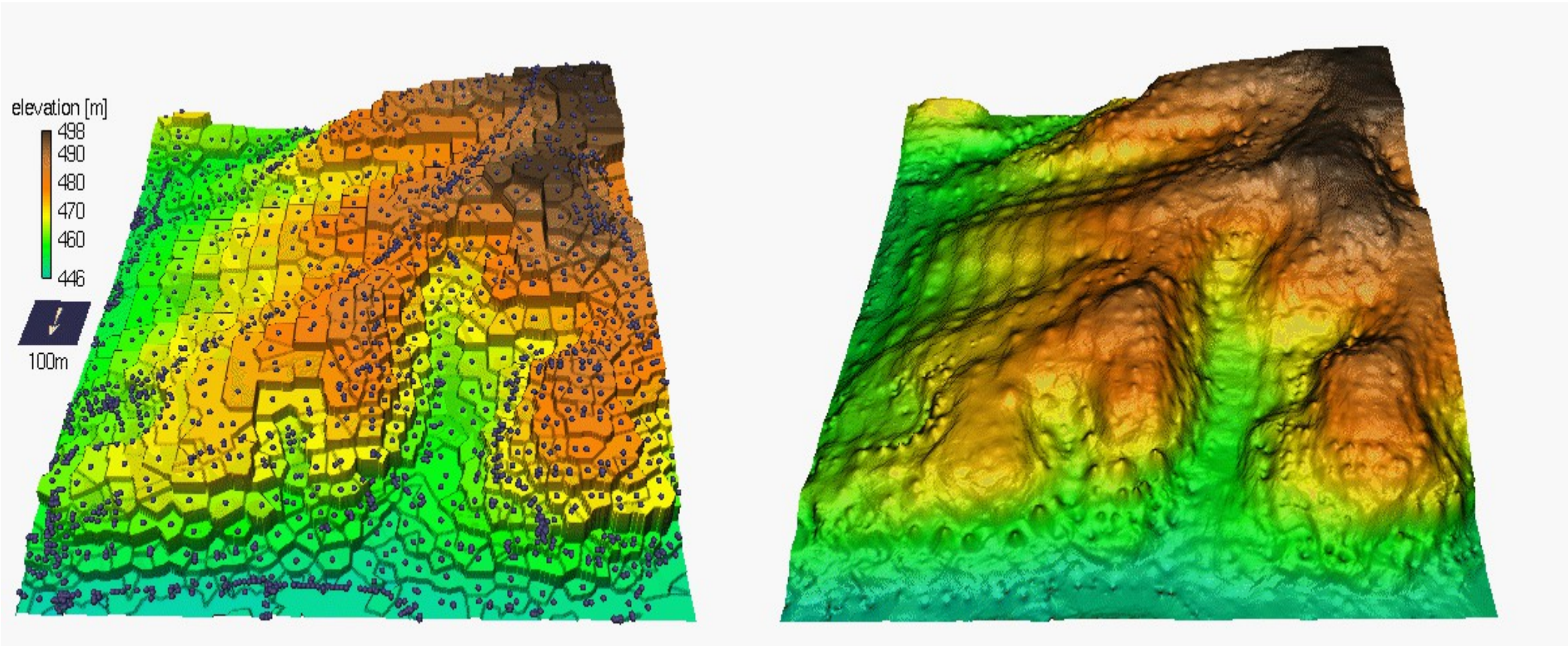
Which approach is "best"?

- No theoretical answer
- Depends on how well the approach models the **'true' spatial structure**, and this is unknown (but we may have prior evidence)
- Should correspond with what we know about the **process** that created the spatial structure

(from David Rossiter)

Voronoi Tesselation

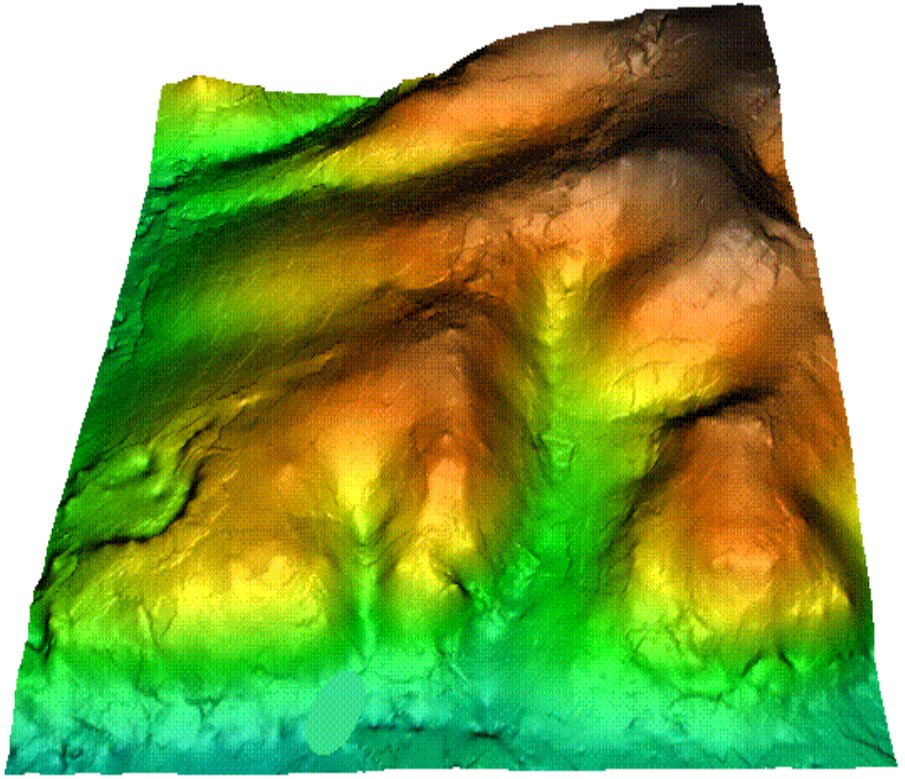
IDW

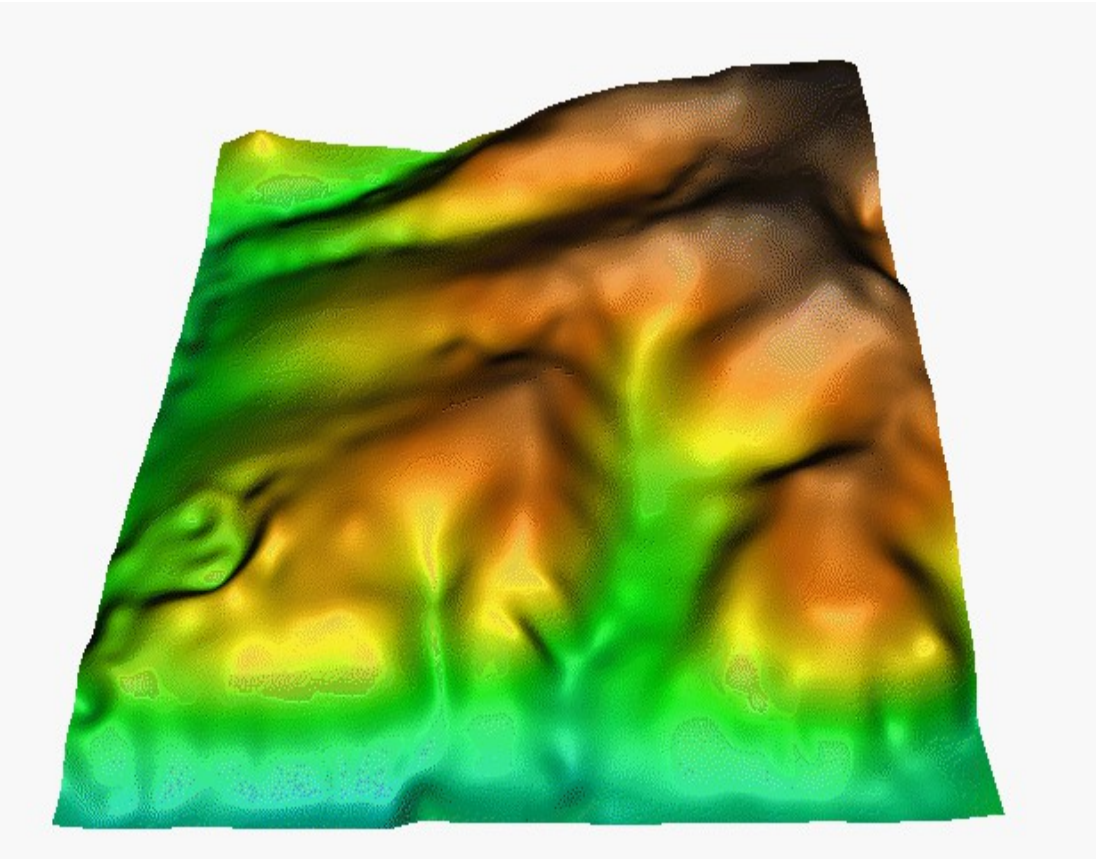


Helena Mitasova: http://skagit.meas.ncsu.edu/~helena/gmslab/interp/

Kriging

Thin plate spline





Helena Mitasova: http://skagit.meas.ncsu.edu/~helena/gmslab/interp/

Approaches to prediction: Global (Regional) Predictors

- Value of the variable depends on **relative geographic position** within a spatial field
  - Example: thickness of a layer of volcanic ash over a buried soil
  - Example: amount of quartz gravels in a soil derived from conglomerate residuum
  - Groundwater depth below a reference level
- Since there is only one process (global), **all** sample points are used to compute the prediction
- (Sometimes the sample points are limited to a region, e.g. in moving trend surfaces or splines)

(from David Rossiter)

Which predictor is “best”? (continued)

- What do we know about the process being modelled?
- Check against an independent **validation** dataset
  - **Mean squared error** (“precision”) of **prediction** vs. **actual** (residuals)
  - **Bias** (“accuracy”) of predicted vs. actual mean
- With large datasets, model with one part and hold out the rest for validation
- **Cross-validation** for small datasets with a modelled structure

(from David Rossiter)

Choosing a Surface Interpolation Model

- Characteristics of the input sample data
- Precision of the sample point measurements
- Frequency and distribution of sample points relative to the scale of variation
- Surface shape and level of local variability
- The context in which the surface model will be used

Interpolate a surface.
Different methods provided by R.

#a) Trend surface fitting polynomial regression surface by least squares (surf.ls, trmat) {spatial package} & (krige) {gstat}
#b) Trend surface fitting by generalized least squares (surf.gls, trmat) {spatial package} & {gstat}
#c) Local polynomial regression fitting with error prediction (loess, predict.loess) {stats})
#d) Bilinear or Bicubic spline interpolation (interp) {akima package}
#e) Multi-level B splines (mba.surf) {MBA package}
#f) Thin plate spline (Tps) {fields package}
#e) Kriging with
    ##variogram calculation (Variogram) {gstat} & (variog) {geoR}
    ##kriging calculation (krige) {gstat} ; (krige.conv) {geoR} ; (Krig) {fields package}
    ##Bayes kriging (krige.bayes) {geoR}
#f) other packages:
    ##variogram calculation (Variogram) {nlme package}
    ##covariance estimation with correlogram (correlogram, variogram) {MASS package}
#g) and many others:
    ## {constrainedKriging} {GEOmap} {geospt} {intamap} {LatticeKrig} {ModelMap} {ramps} {RandomFields} {sgeostat} {spatstat} {spBayes} {SpatialExtremes}

Automatic interpolation package {automap package}
Areal interpolation for GIS data {AIGIS package}

Which method to use, you ask? Why not just explore the **gstat** or **geoR** package or for that matter any of the other packages doing spatial interpolation?

C:\geostat_quebec\spatInter_G13\Demoauto.mappackage&spatial interpolation.htm