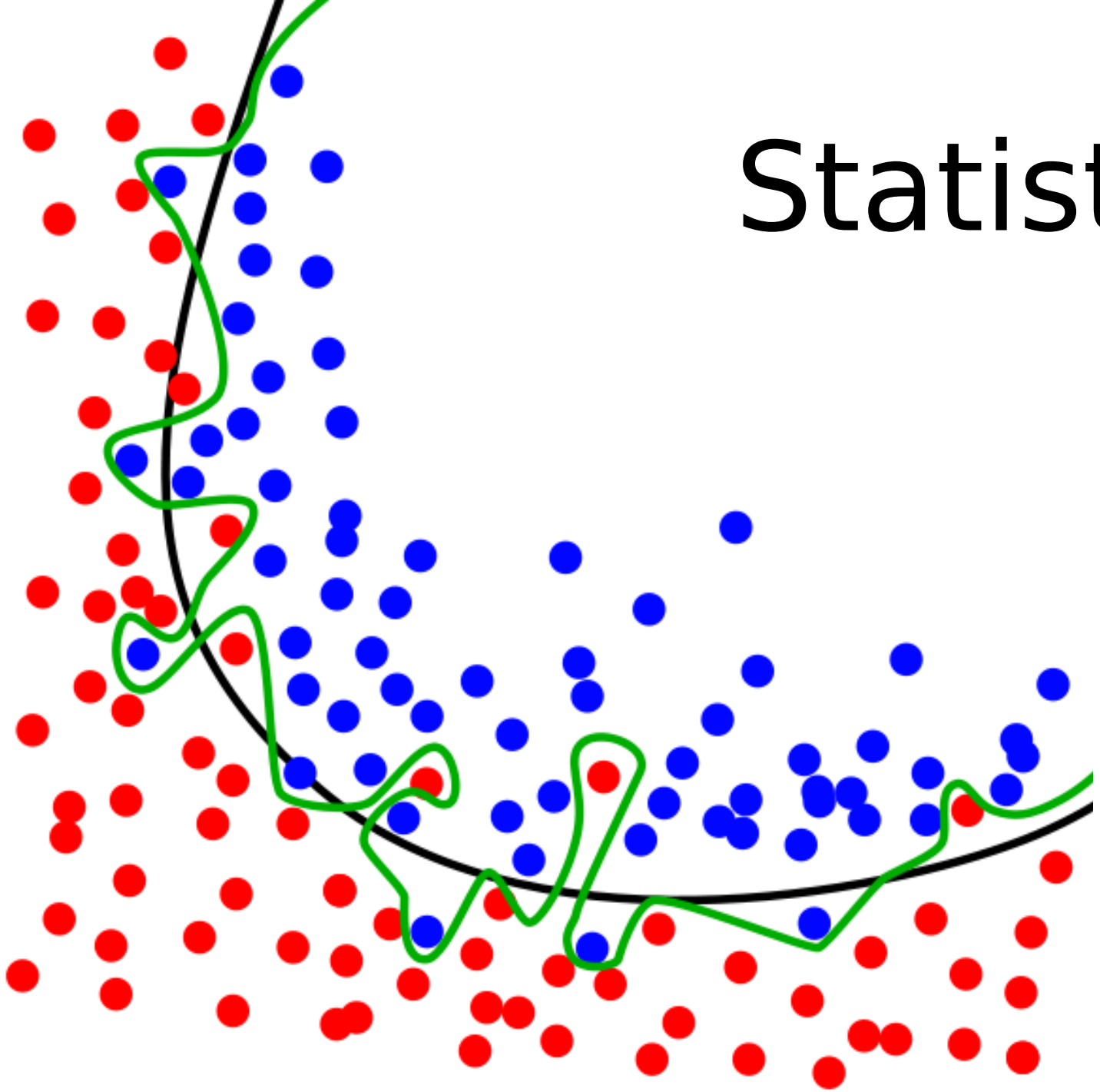
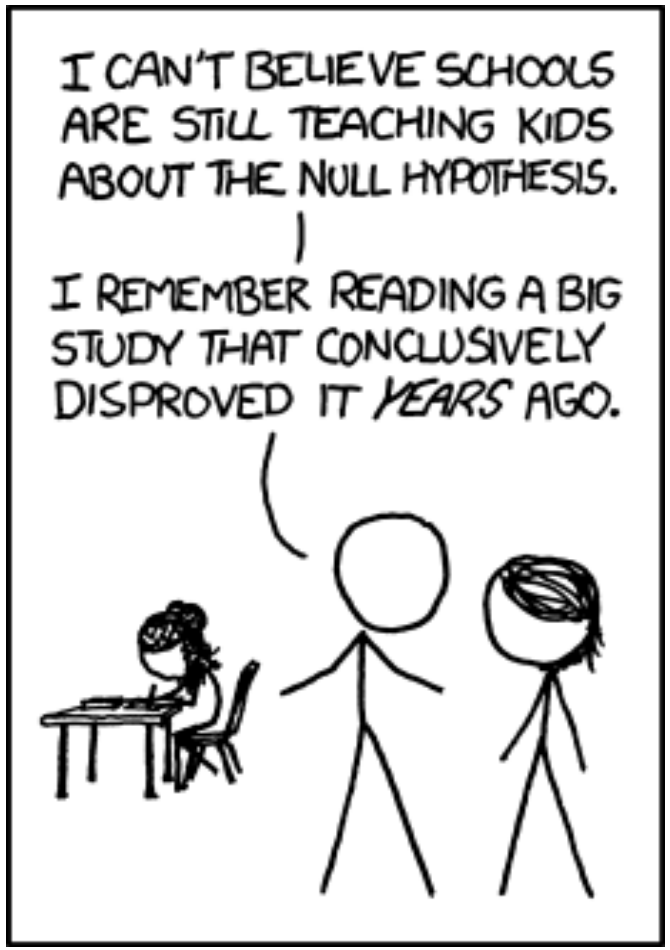


# Statistical Learning



# Supervised learning

*quantity of interest* *function* *predictors* *error*

Assume: $Y = f(X) + \epsilon$

Estimate: $f$ to get: $\hat{f}$

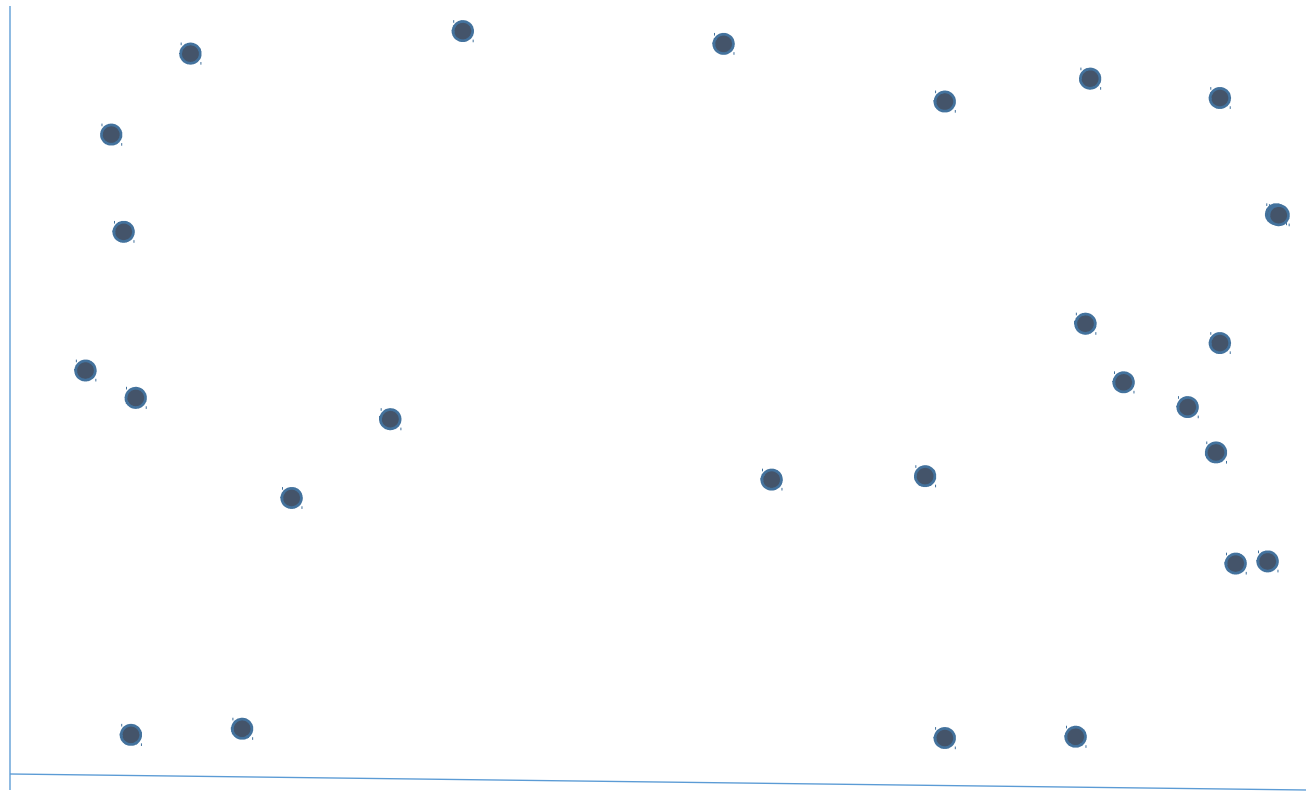
Such that: $\hat{Y} = \hat{f}(X)$

For prediction and/or inference

# Model fit  vs. Model stability

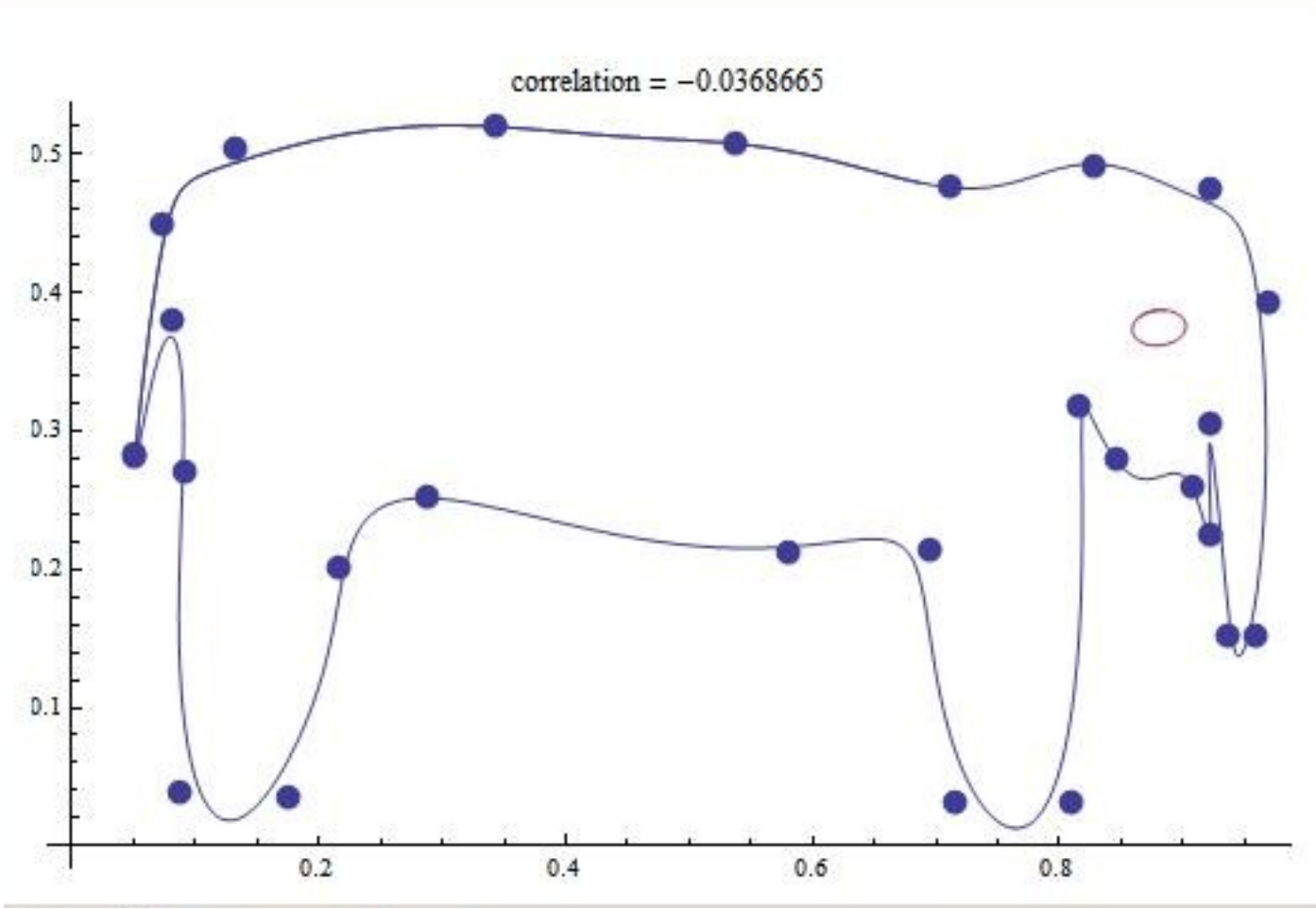
(Bias – variance  trade-off)

(over – under fitting)



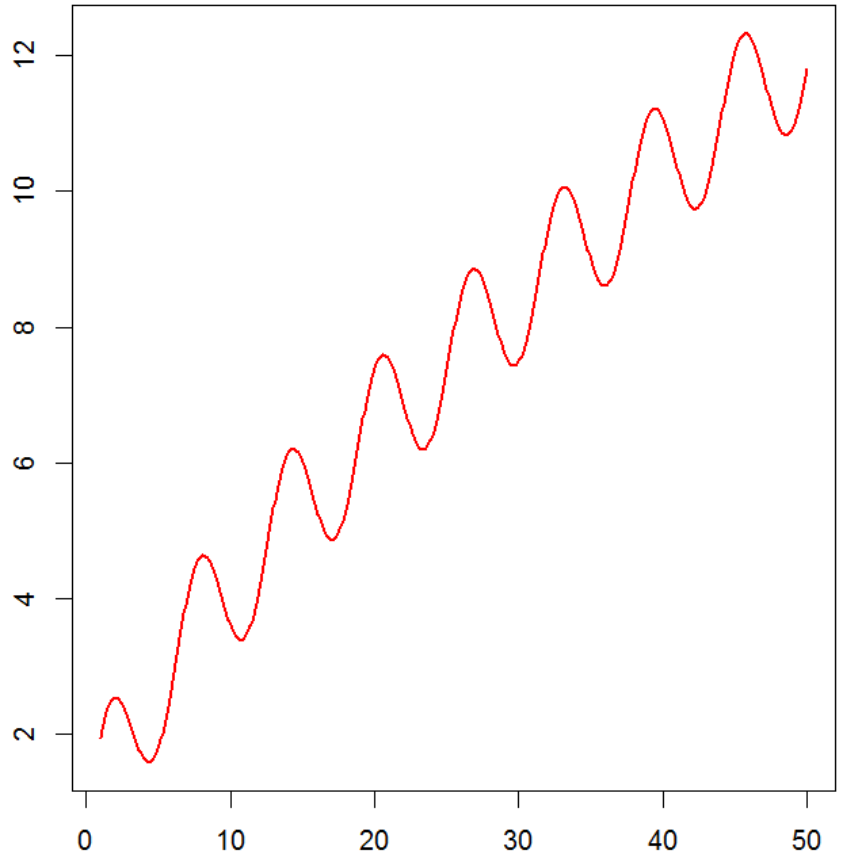
With four parameters I can fit an elephant, and with five I can make him wiggle his trunk

--- John von Neumann



# Assume known function *f*

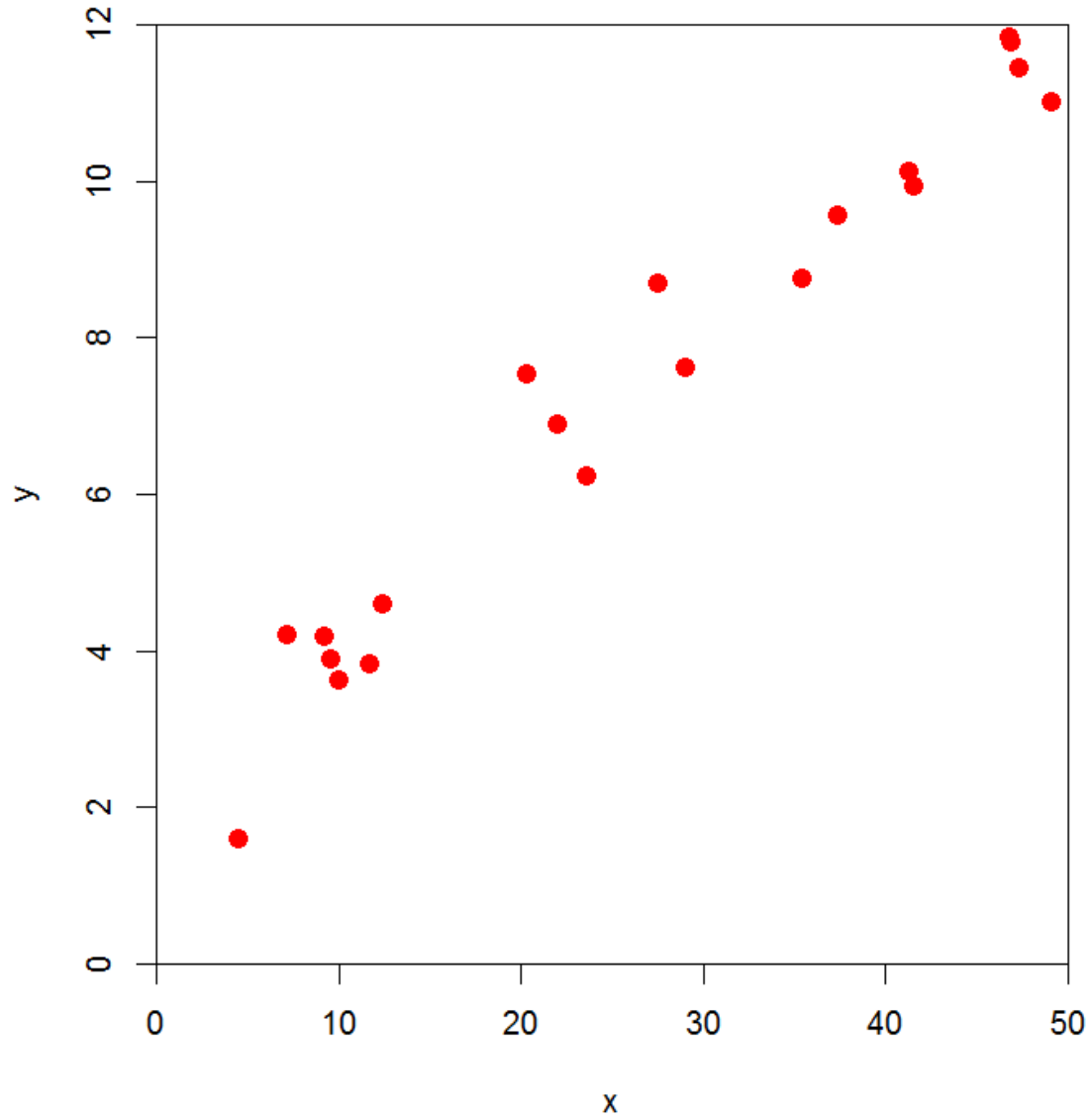
$$Y = f(X) = \frac{X}{10} + \sin(X) + \sqrt{X}$$




```
f <- function(x) x/10 + sin(x) + sqrt(x)
X <- seq(1,50,0.1)
Y <- f(X)
plot(X, Y, type='l')
```

# Collect a sample (here without error)

```
set.seed(2)
x <- sample(X, 20)
y <- f(x)
plot(x, y)
```

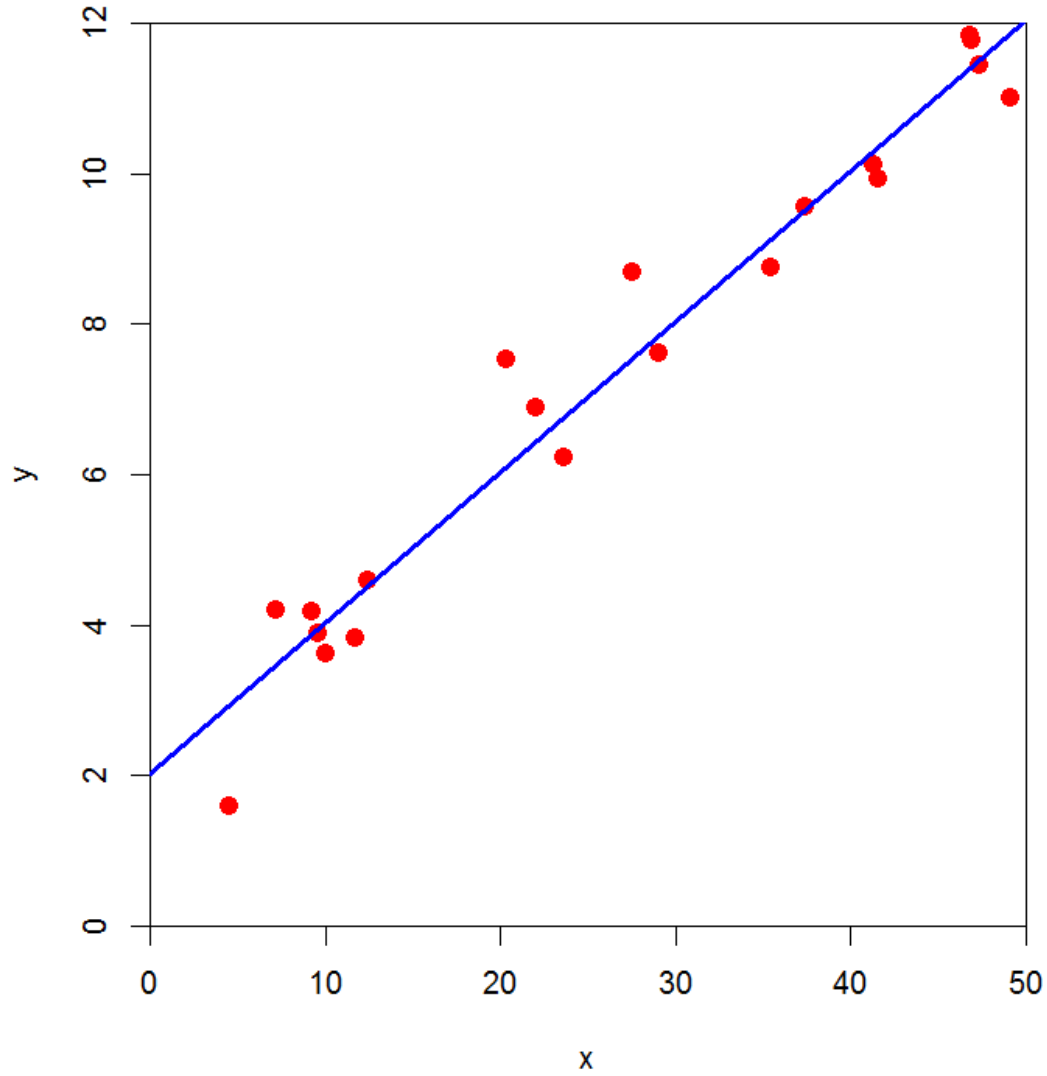


# Estimate *f*

```
> m <- lm(y ~ x)
> abline(m)
> coefficients(m)

(Intercept)                x
   2.032109        0.200007

> summary(m)$r.squared
[1] 0.9573415
```




$$\hat{f}: \quad y = 2.03 + 0.2\,x$$

# Measuring accuracy

$$MSE = \frac{1}{n}\sum_{i=1}^{n}( y_i - \hat{f}(x_i) )^2$$

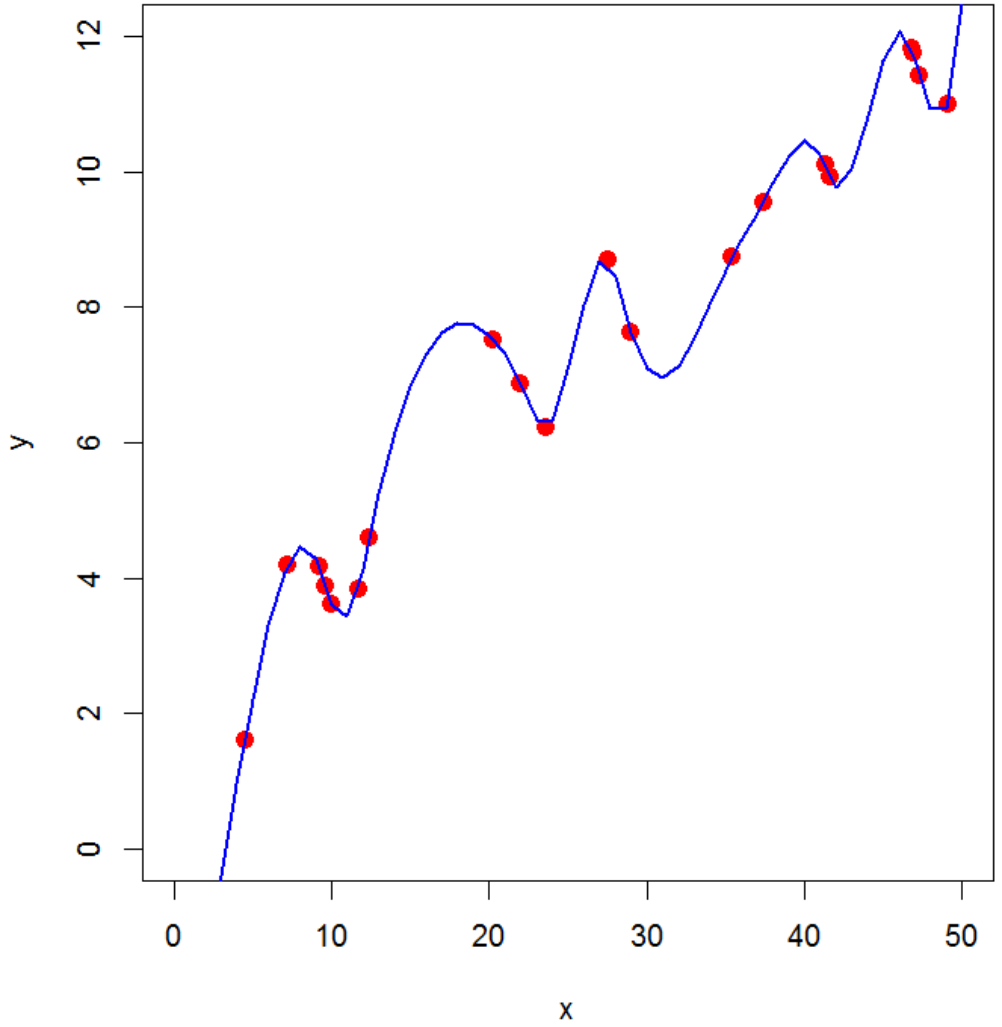
$$= \frac{1}{n}\sum_{i=1}^{n}( y_i - \hat{y}_i )^2$$

```
> p <- predict(m)
> mse <- mean((y - p)^2)
> mse
[1] 0.4119498
```

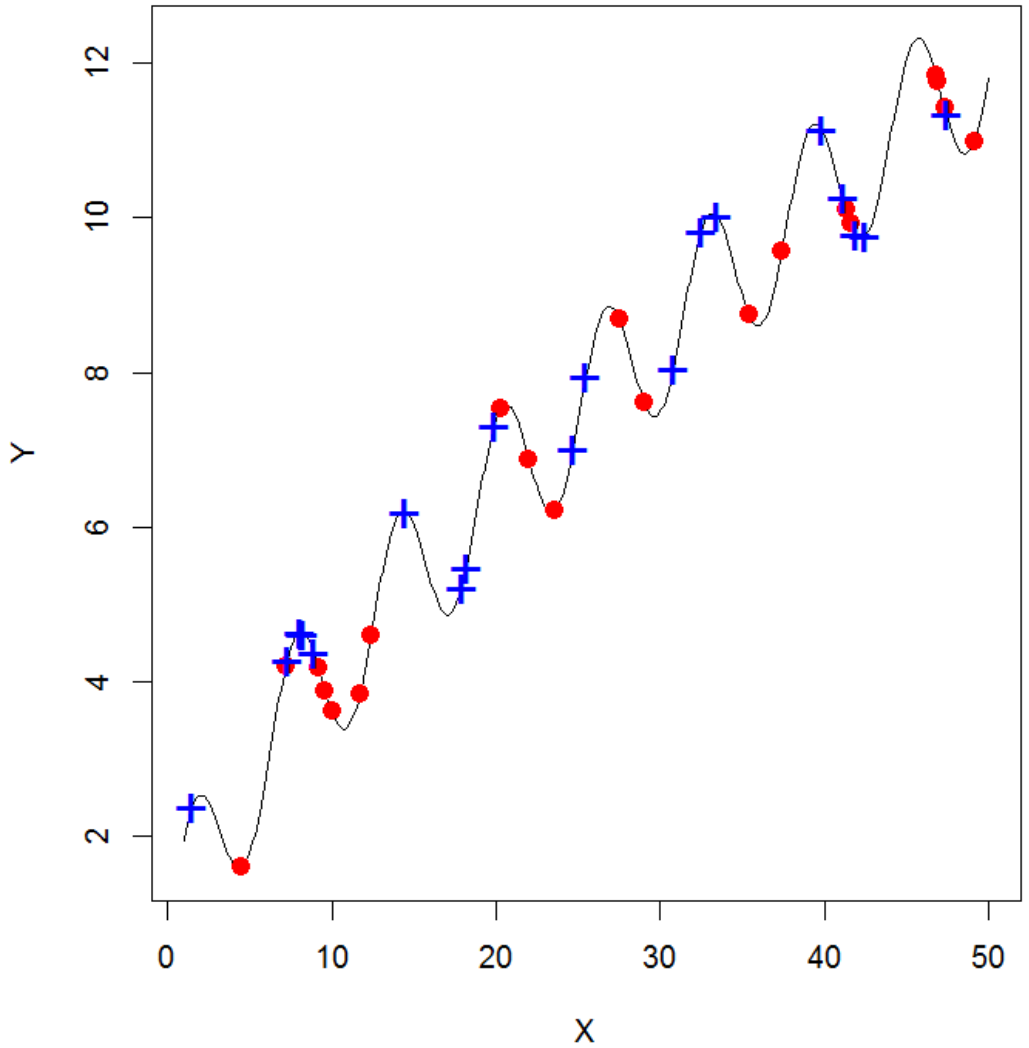
# Non-parametric model

```
> sf <- splinefun(x, y)
> mse <- mean((y - sf(x))^2)
> mse
[1] 0
```

```
> plot(x, y)
> lines(1:50, sf(1:50))
```



# Test and training data




```
> set.seed(3)
> x0 <- sample(X, 20)
> y0 <- f(x0)
> plot(X,Y, type='l')
> points(x, y, col='red', pch=20, cex=2)
> points(x0, y0, col='blue', pch='+', cex=2)
```

# Parametric model

train - train

```
> p <- predict(m)
> mean((y - p)^2)
[1] 0.4119498
```

train - test

```
> pt <- predict(m, data.frame(x=x0))
> mean((y0 - pt)^2)
[1] 0.591895
```

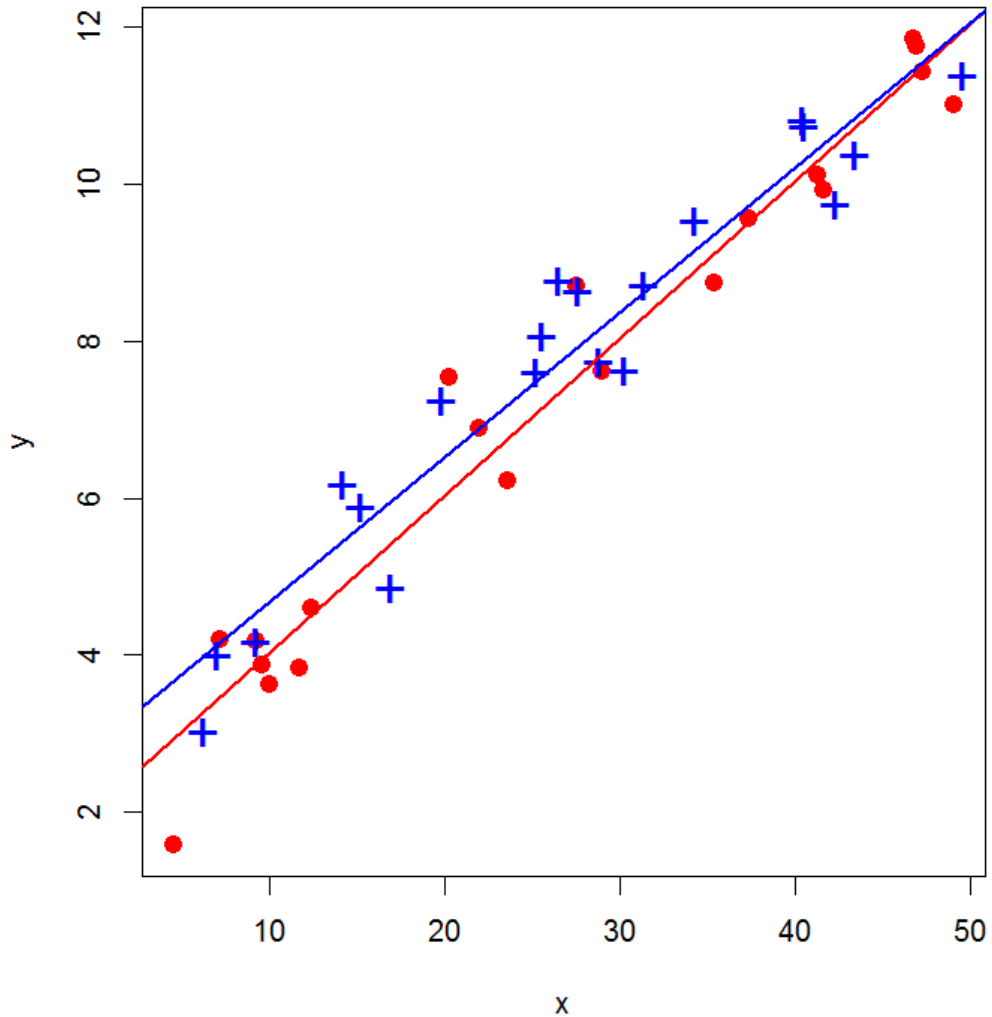
# Nonparametric model

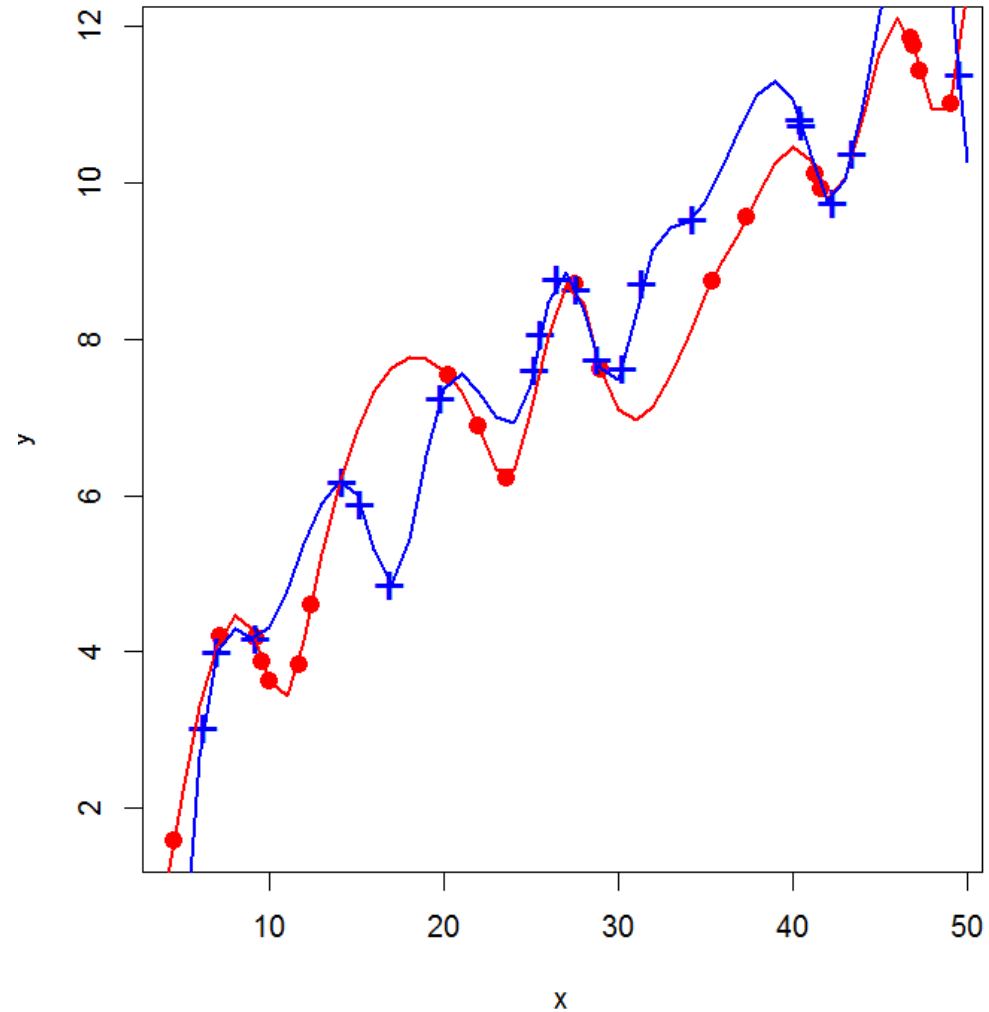
train - train

```
> mean((y - sf(x))^2)
[1] 0
```

train - test

```
> mean((y0 - sf(x0))^2)
[1] 0.7397438
```


y
x
2
4
6
8
10
12
10
20
30
40
50


y
x
2
4
6
8
10
12
10
20
30
40
50

## Parametric models

Easy interpretation & understanding
Not prone to overfitting

## Use model train and test data

Splitting is better than independent data sets (says Robert)

## Beware the variance-bias tradeoff

# Simple linear regression if great. But, you can get in trouble with:

1. **Non-linearity of the data (!!)**
2. **Non-independence of the error terms (perhaps spatially)**
3. Non-constant variance of error terms
4. Outliers
5. High leverage points
6. **Collinearity**

Alternatives to (simple) least squares regression

## For increased accuracy and interpretability

Subset selection --- Stepwise, Lasso

Shrinkage --- Ridge, Lasso

Dimension reduction --- Principal components regression, PLS

## “Machine learning” methods such as Random Forest

# ISLR

## Book website

http://www-bcf.usc.edu/~gareth/ISL/

## Answers to the ISLR questions

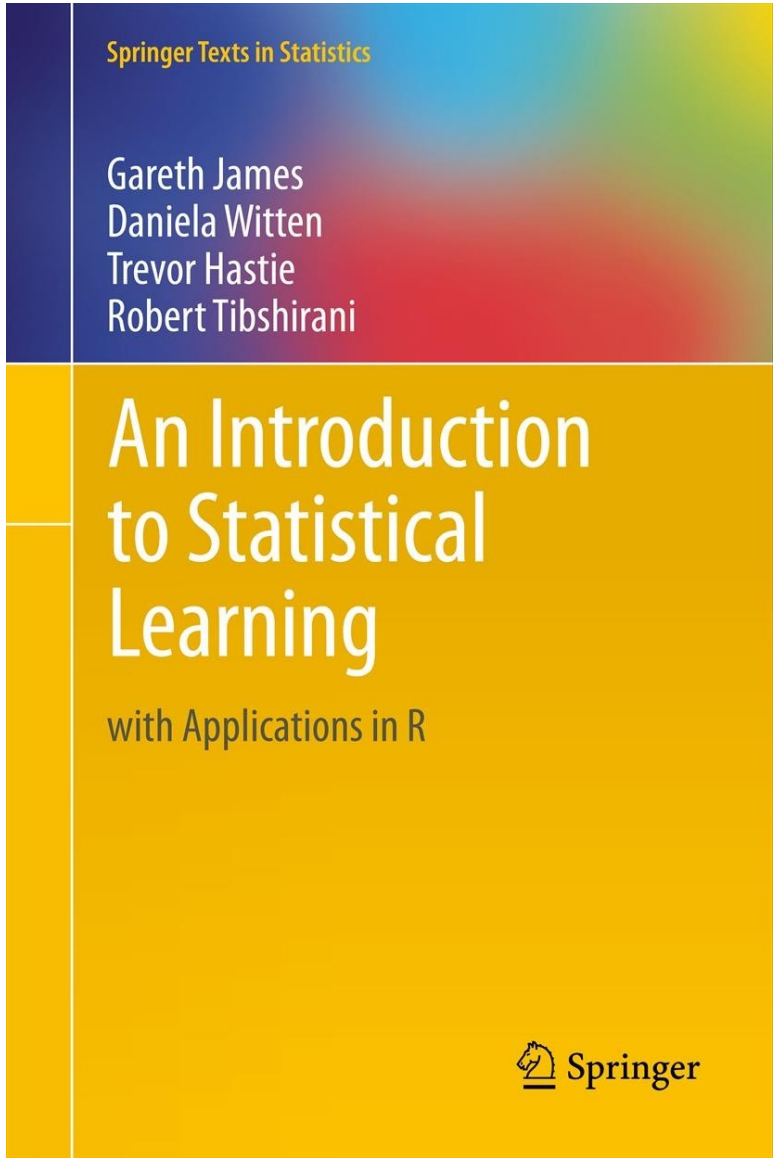
https://github.com/asadoughi/stat-learning

## Lectures by Hastie and Tibshirani

http://www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/

## Slides and R videos by  Abbass Al Sharif

http://www.alsharif.info/#!iom530/c21o7



# Interpolation

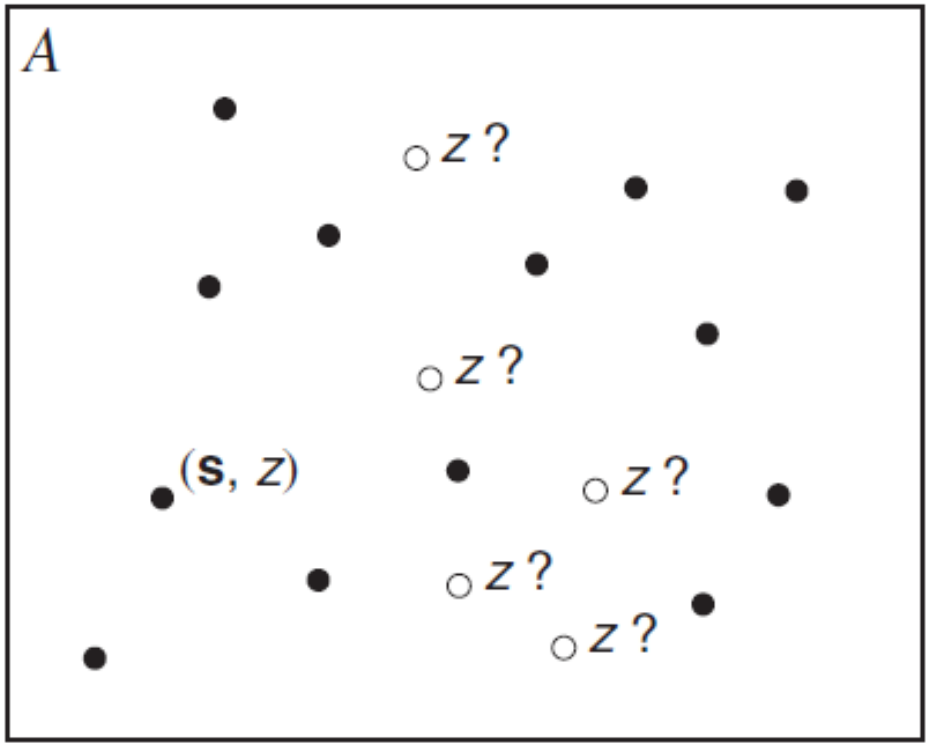



Figure 9.3 The interpolation problem. Control points are black circles, where we know the location, **s**, and the height, $z$, but we require the field height, $z$, anywhere in the region $A$—say, at the unfilled circle locations.

*Source: 'O Sullivan and Unwin*

# Proximity polygons

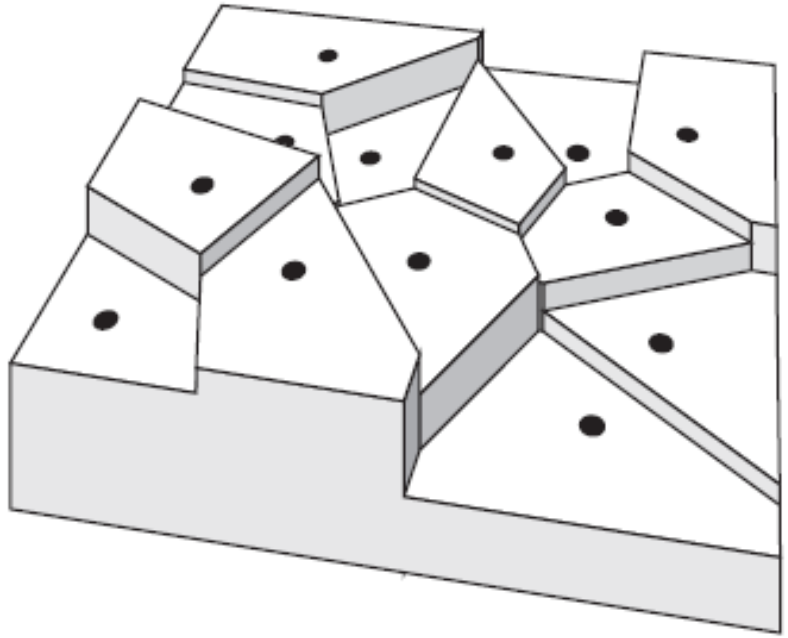
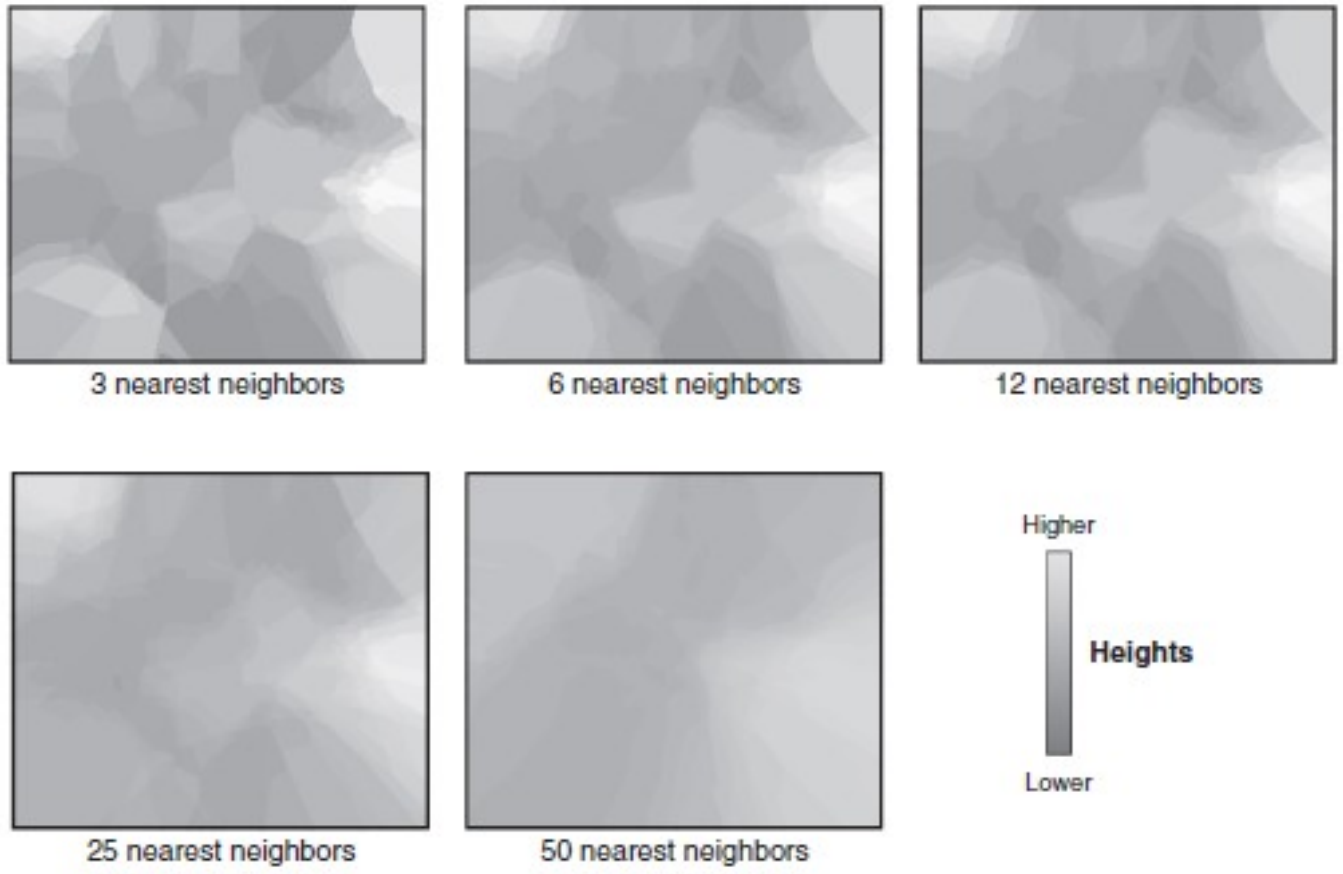


Figure 9.6 The "blocky," discontinuous results of an interpolation using proximal polygons.

(= a two dimensional step function (see ISLR))

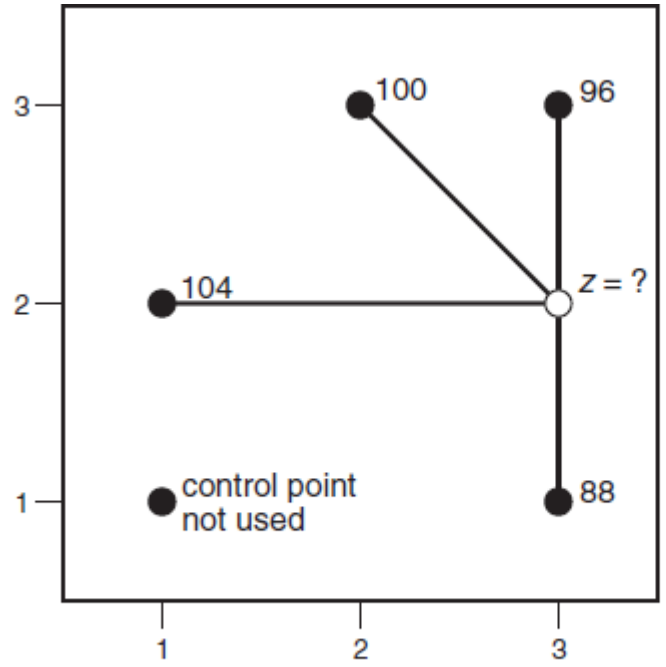
*Source: 'O Sullivan and Unwin*

# Nearest neighbor**s**




*Source: 'O Sullivan and Unwin*

# Inverse distance weighted




Inverse distance weighting in spatial interpolation.

Table 9.1 Illustrating Estimation Using Inverse Distance Weighting

| *Control point* | *Height* $z_i$ | $x_i$ | $y_i$ | *Distance* $d_{ij}$ | *Inverse distance* $1/d_{ij}$ | *Weight* $w_{ij}$ | *Weighted value* $w_{ij}z_i$ |
|---|---|---|---|---|---|---|---|
| 1 | 104 | 1 | 2 | 2.000 | 0.50 | 0.1559 | 16.21 |
| 2 | 100 | 2 | 3 | 1.414 | 0.71 | 0.2205 | 22.05 |
| 3 | 96 | 3 | 3 | 1.000 | 1.00 | 0.3118 | 29.93 |
| 4 | 88 | 3 | 1 | 1.000 | 1.00 | 0.3118 | 27.44 |
| Totals | | | | | 3.21 | 1.0000 | 95.63 |

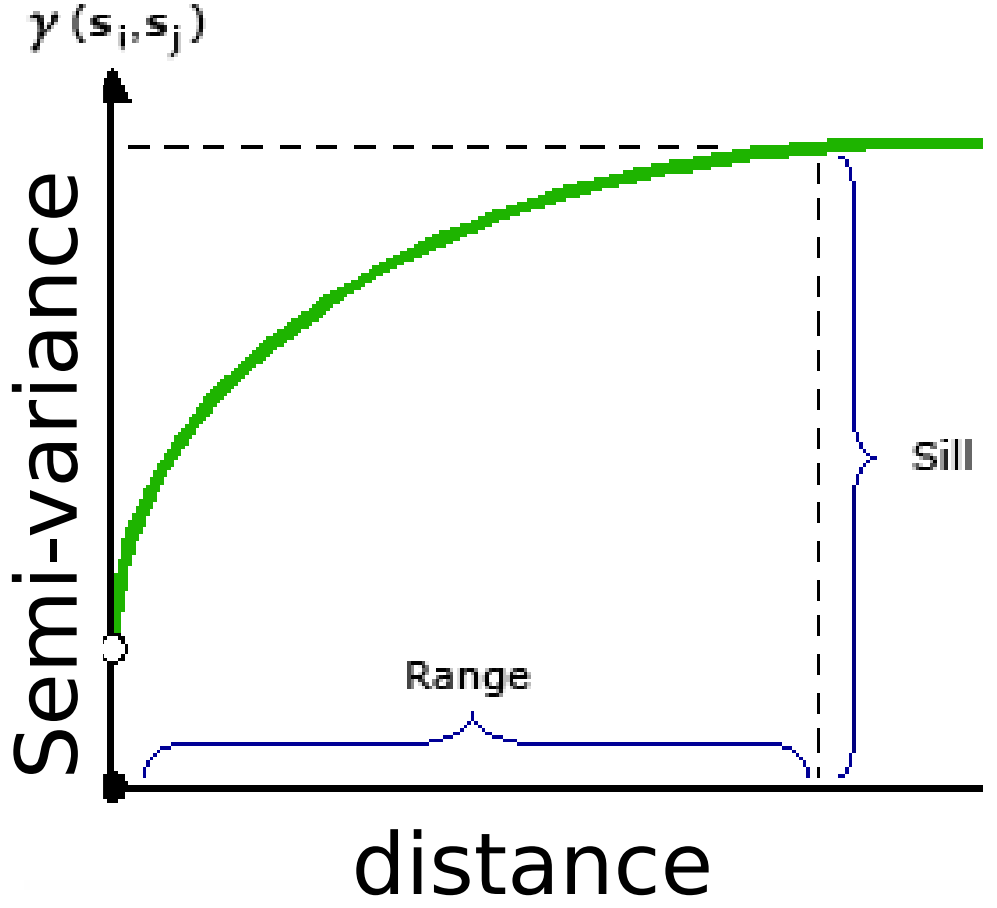
# Kriging

Weighted linear combination of the available samples to predict the response for an unmeasured location.

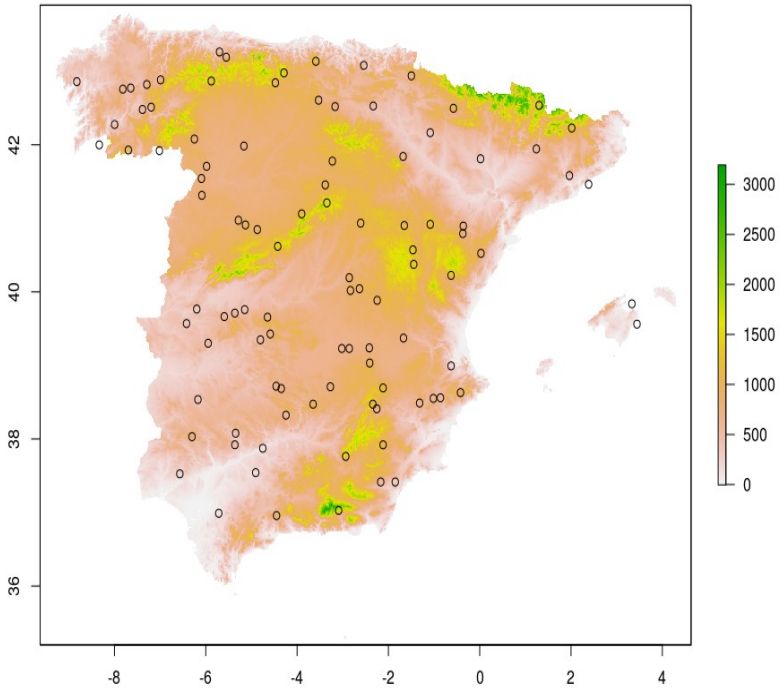
Weighting is based on a function derived from the semi-variogram

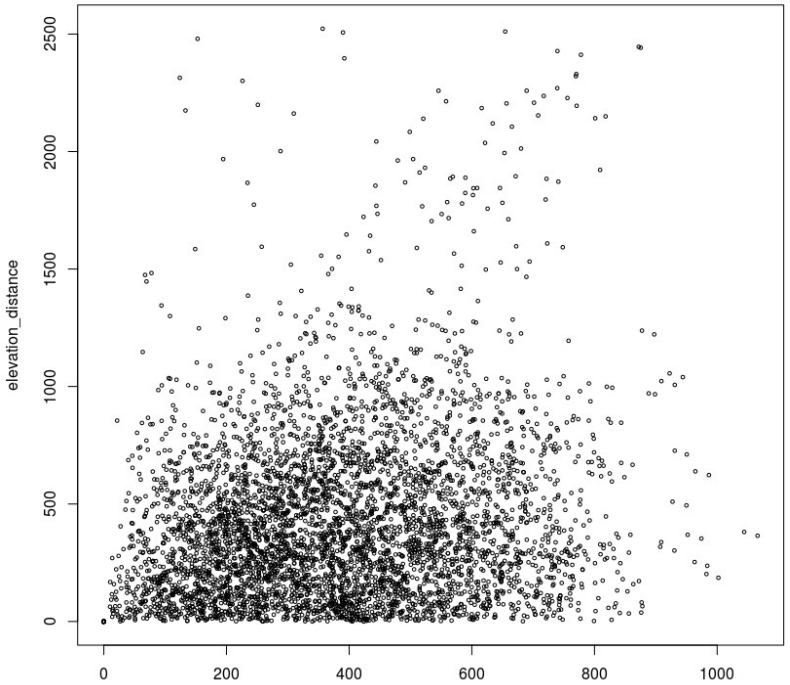
$$\gamma(h) = \frac{1}{2} E\{[f(x) - f(x+h)]^2\}$$

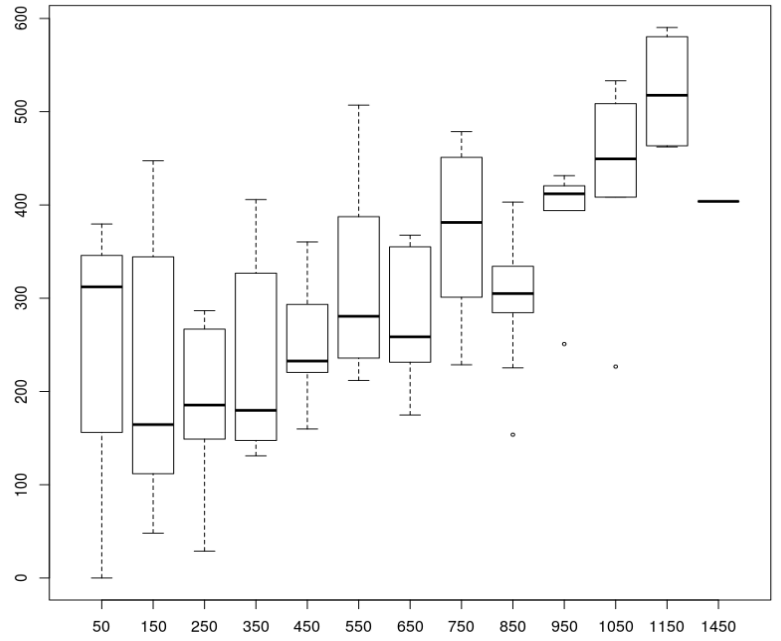
$$\hat{Z}(s_0) = \sum_{i=1}^{N} \lambda_i Z(s_i)$$



```
library(dismo)
alt <- getData('alt', country='TZA')
set.seed(9)
xy <- randomPoints(alt, 100)
elevation  <- extract(alt, xy)
spatial_distance <- pointDistance(xy, lonlat=TRUE)/1000
elevation_distance <- as.matrix(dist(elevation))
plot(spatial_distance, elevation_distance, cex=.5)
bins <- 100 * round(spatial_distance/100)
boxplot(elevation_distance ~ bins)
```



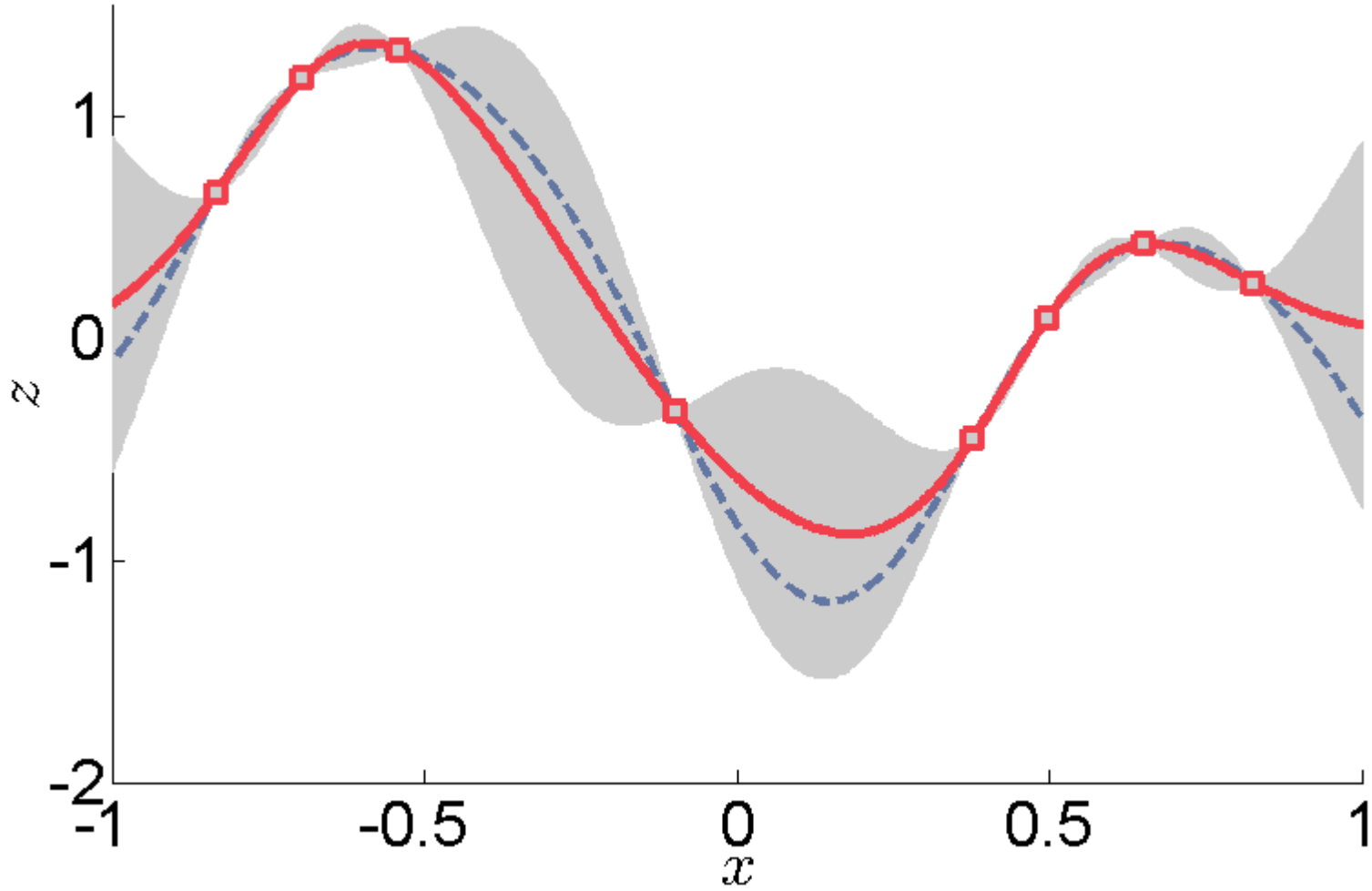



# Kriging

- Explicitly uses observed spatial autocorrelation
  and assumes it is constant (and perhaps isotropic)

- Assumes observations have no error (exact interpolator)
  but allows estimation of uncertainty

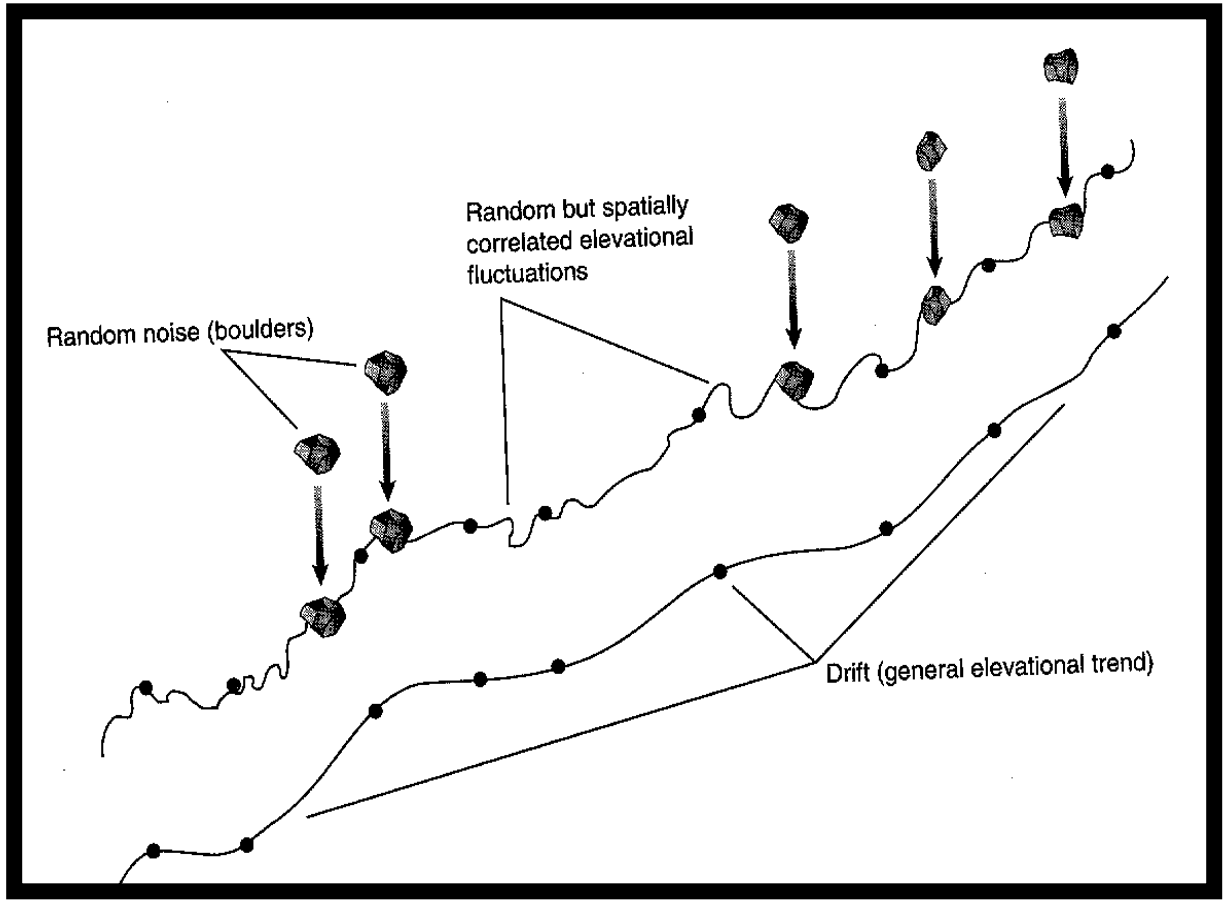
- Is often presented as the *Best Linear Unbiased Estimator*
  but that is theory, it may not apply to real data




Example of one-dimensional data interpolation by Kriging, with confidence intervals. Squares indicate the location of the data. The Kriging interpolation, shown in red, runs along the means of the normally distributed confidence intervals shown in gray. The dashed curve shows a spline that while smooth nevertheless departs significantly from the expected intermediate values given by those means.

# Types of Kriging

- Ordinary kriging
- Universal kriging
- Regression kriging
- Cokriging. …



# Thin plate spline

Polynomial model that minimizes the residual sum of squares subject to a constraint that the function have a certain level of smoothness (or roughness penalty).

In the absence of the rougness penalty it is the polynomial base model

Otherwise it is an *inexact interpolator*