

**A Novel Empirical Approach to the  
Defect Content Estimation Problem  
for Software Inspections**

**Frank Padberg  
Universität Karlsruhe  
Germany**

# Motivation

- not all defects are detected during an inspection
- total number of defects is not known exactly
- number of defects is an important management tool  
(cf. prescribed level of defect-freeness)

# Motivation

- not all defects are detected during an inspection
- total number of defects is not known exactly
- number of defects is an important management tool (cf. prescribed level of defect-freeness)
- **reliably estimate** the number of defects in a software document from the outcome of an inspection!

# Inspection Outcome

- list of detected defects
- zero-one matrix: shows which reviewer detected which defect
- classification of the defects

# Existing Estimation Methods

- capture–recapture methods (Eick ea. ICSE 1992)
- curve–fitting methods (Wohlin ea. ICSE 1998)
- studies show that estimates are far too unreliable to be useful in engineering practice (Briand ea. TSE 2000, Biffi ea. ICSE 2001)

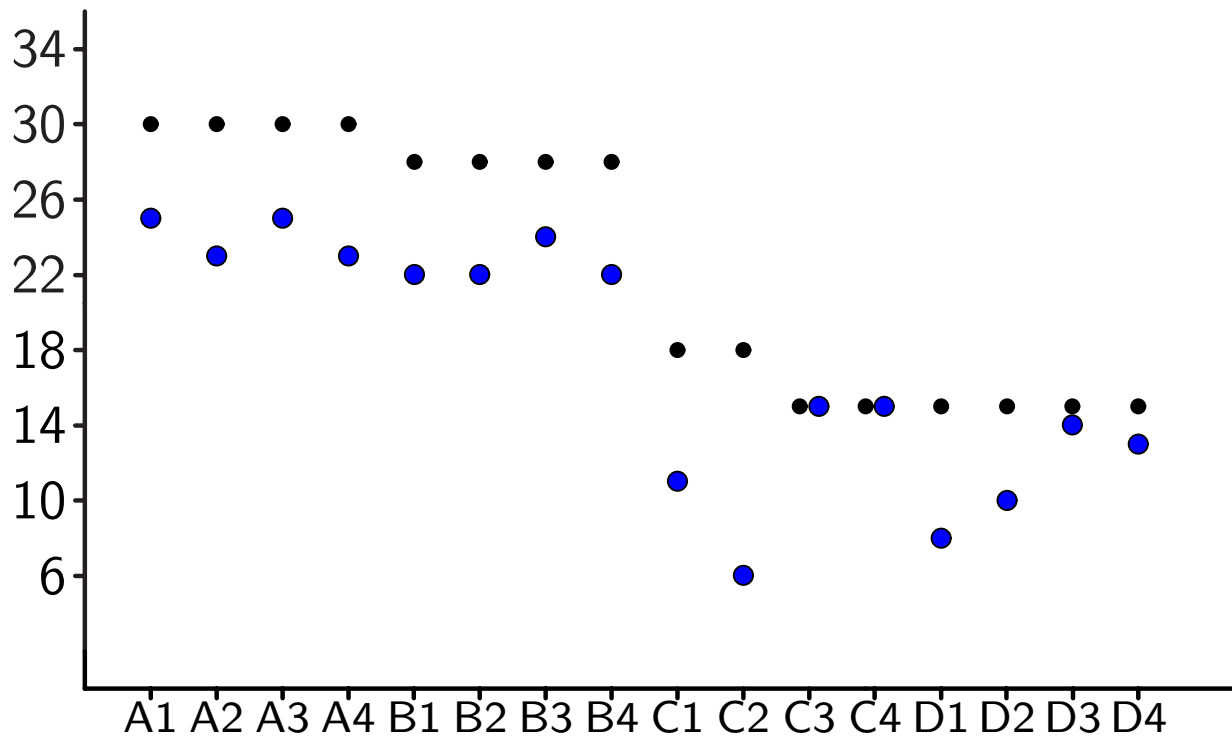
# Sample Database

- 16 inspections from controlled experiments at NASA SEL (Basili ea. 1994/1995)
- specification documents of varying size
- between 6 and 8 reviewers
- true number of defects known exactly
- serves as standard benchmark

# Input Data for Capture–Recapture

- inspection viewed as a short test series
- number  $w_k$  of defects detected by reviewer  $k$
- total number  $d$  of different defects detected
- example:  $(9, 7, 6, 13, 9, 6)$  and  $d = 23$

# Capture–Recapture Estimates

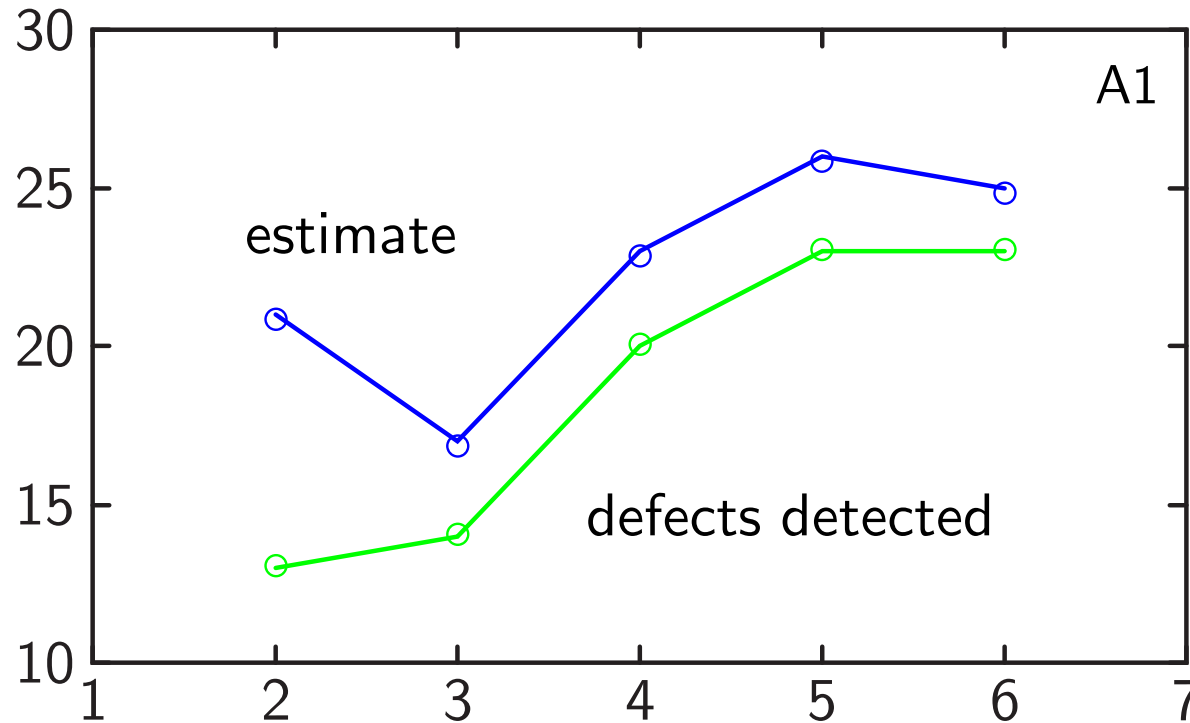


mean abs. error of 24 percent

max error of -67 percent

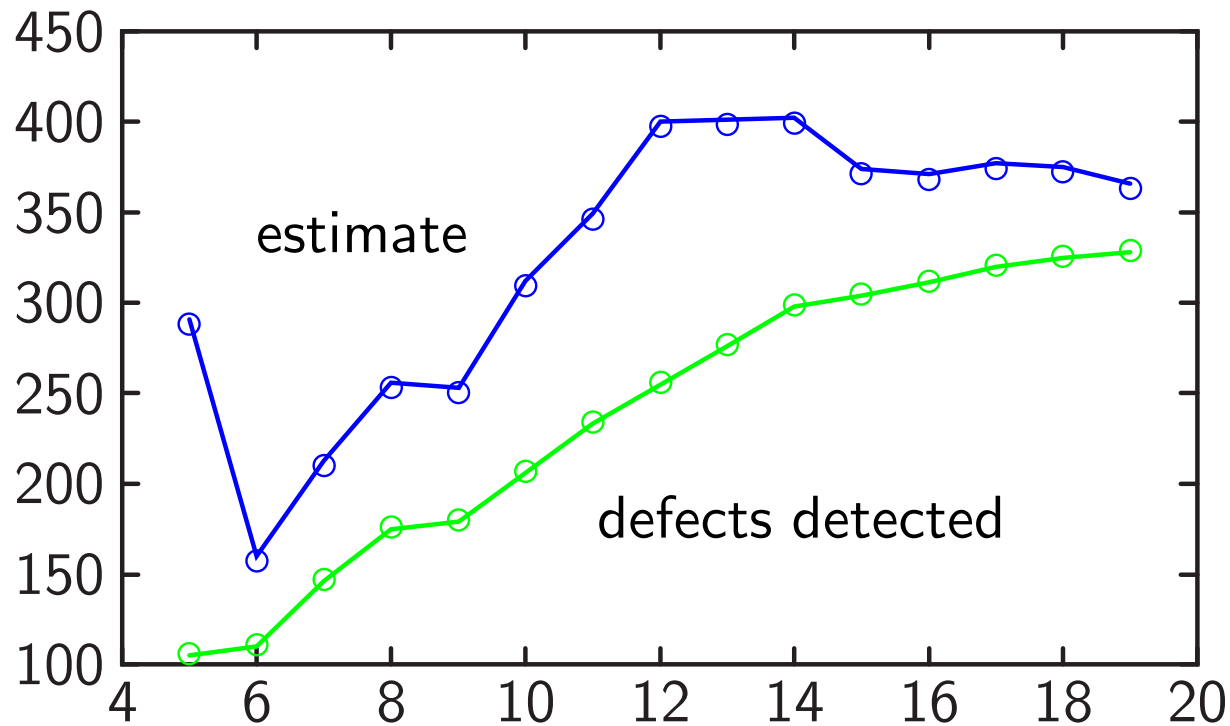


# CR-Estimate versus Number of Reviewers



estimates vary with the number of reviewers;  
final estimate too low (25 instead of 30)

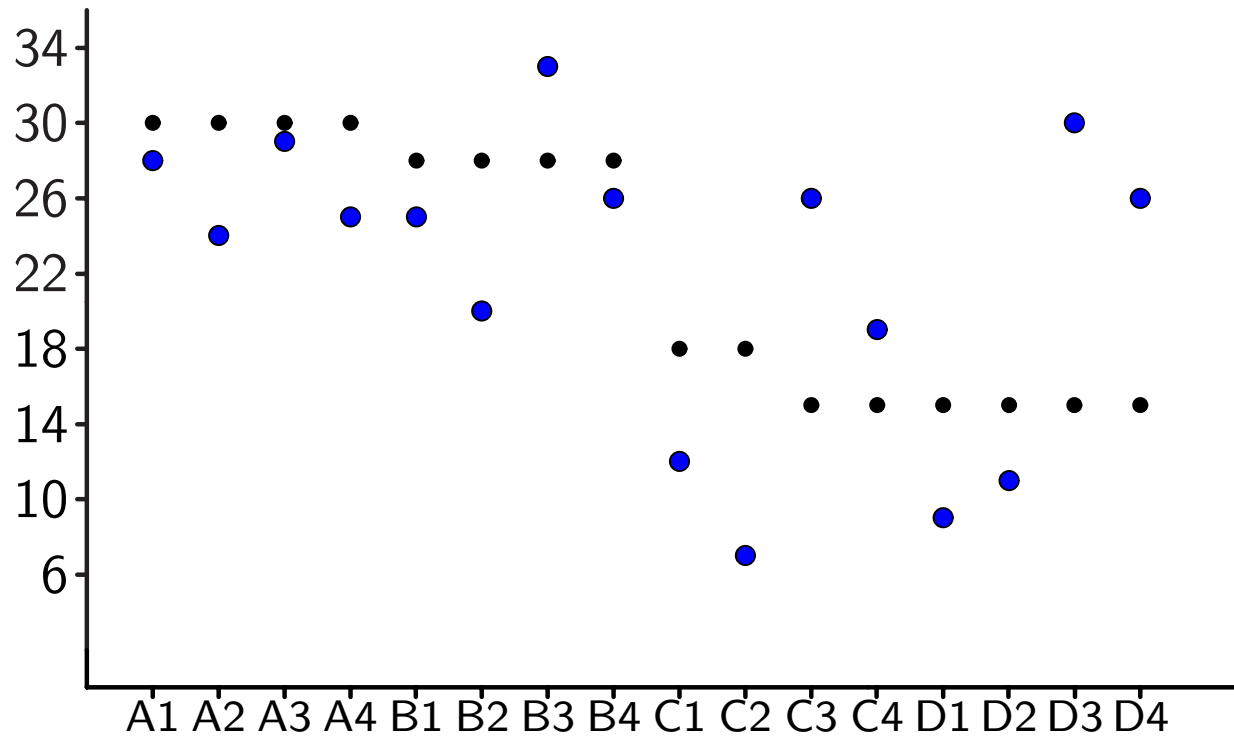
# Cap-Recap Estimate Versus Test Series Length



some test series of length 19

estimate needs some time to stabilize!

# Estimates for Detection Profile Method



mean abs. error of 36 percent

extremely high variation

# Why Capture–Recapture Fails

- mathematics: "test series" is too short

# Why Capture–Recapture Fails

- mathematics: "test series" is too short
- only the outcome of the current inspection enters the estimation

# Why Capture–Recapture Fails

- mathematics: "test series" is too short
- only the outcome of the current inspection enters the estimation
- in other words: no learning from experience

# Machine Learning Approach

- use empirical data about past inspections for estimating
- **learn** relationship between observable **features** of an inspection and **true number of defects** contained in the document

# Machine Learning Approach

- use empirical data about past inspections for estimating
- **learn** relationship between observable **features** of an inspection and **true number of defects** contained in the document
- view defect content estimation as a **regression problem**



# Required Inspection Data

- zero-one matrix
- document meta-data:  
type, size, complexity, ....
- inspection meta-data:  
reading technique, number of reviewers, ....
- true number of defects

# Steps to Take

1. collect empirical inspection data
2. choose features
3. choose regression technique
4. possibly subdivide database (meta-data)
5. do the regression (machine learning)

# Building a Database

- collect data from as many inspections as possible (inspection outcome and meta-data)
- trace defects which are detected in later phases (including maintenance) back to the corresponding document
- compute approximate value for true number of defects for each document in the database

# Candidate Features

- derived from zero–one matrix
- TDD, AVE, MIN, MAX, STD
- example A1:

( 9 , 7 , 6 , 13 , 9 , 6 ) and 23 yields

| TDD | AVE | MIN | MAX | STD |
|-----|-----|-----|-----|-----|
| 23  | 8.3 | 6   | 13  | 2.4 |

# Input Data for Linear Regression

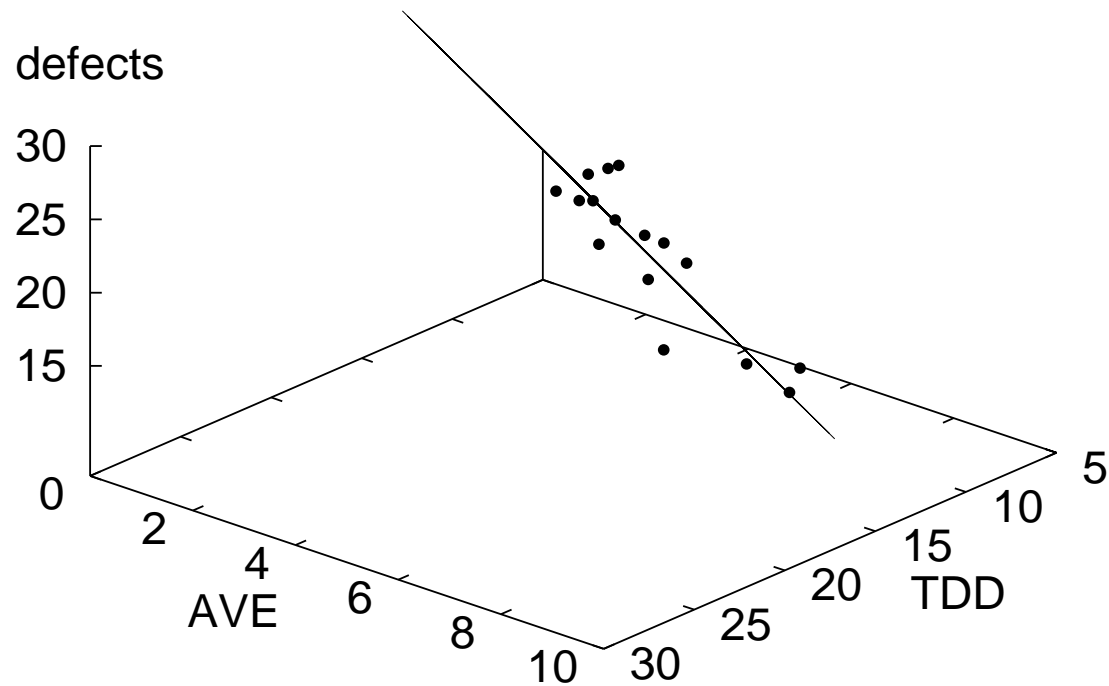
- correlation analysis yields ranking

TDD > AVE > MIN > MAX > STD

- some datapoints:

| inspection | TDD | AVE | target |
|------------|-----|-----|--------|
| A1         | 23  | 8.3 | 30     |
| B1         | 20  | 6.0 | 28     |
| C1         | 10  | 3.2 | 18     |
| D1         | 6   | 1.3 | 15     |

# Regression Hyperplane



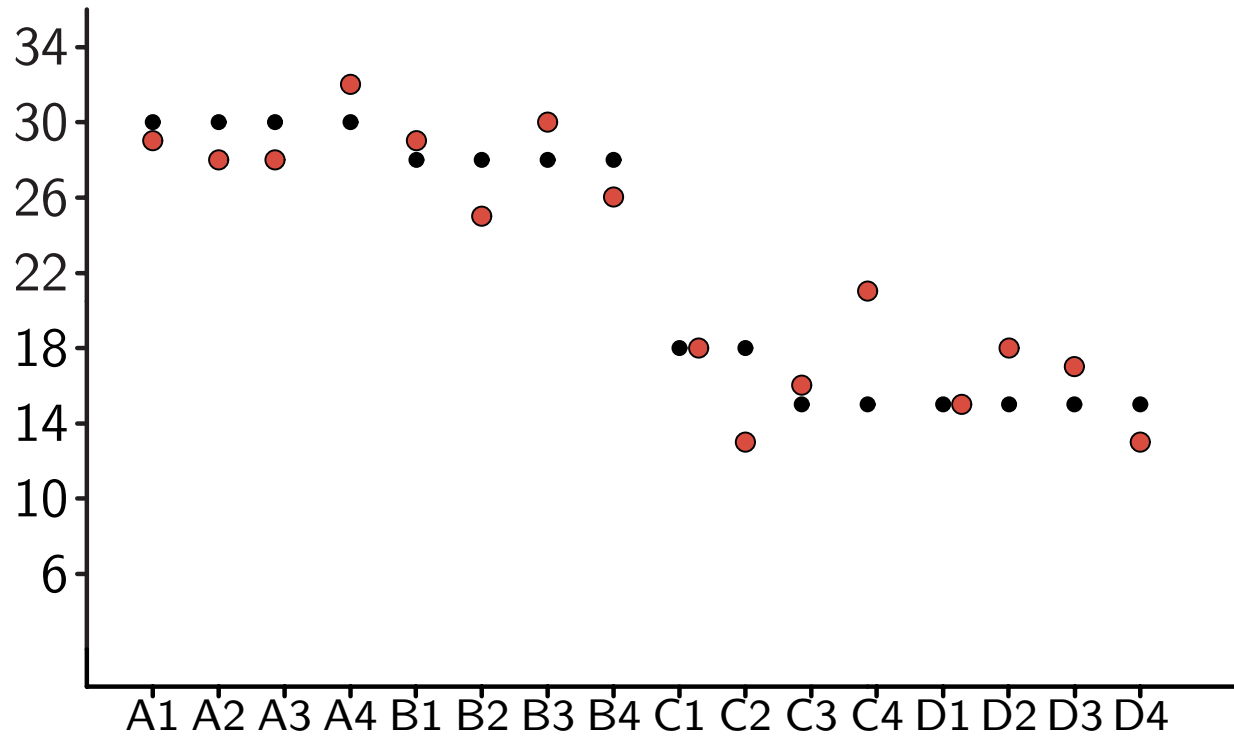
all 16 inspections

some points have large distance to hyperplane

# Jackknife Validation

- leave out an inspection from the database
- compute the regression hyperplane using the remaining 15 inspections
- compute the regression estimate for the one inspection which was left out
- compare the estimate with the true value

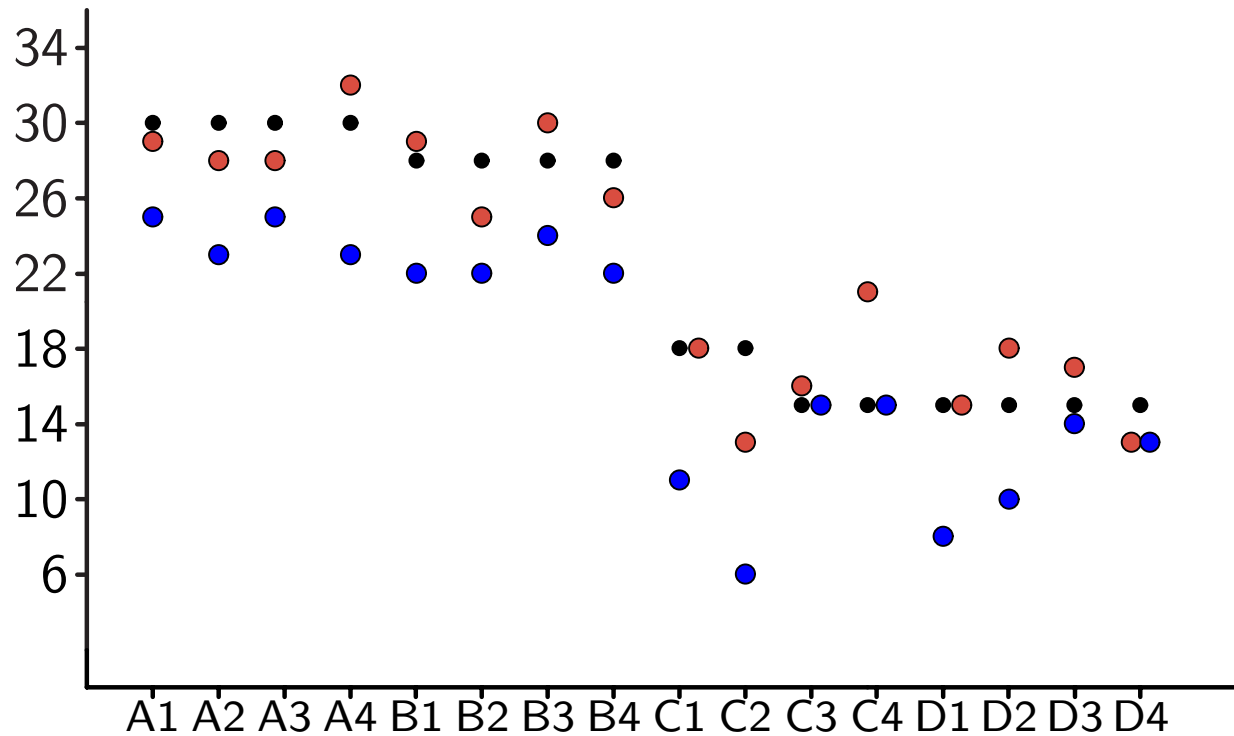
# Linear Regression Estimates



jackknife error of 11 percent  
max error of 40 percent



# Linear Regression versus Capture–Recapture

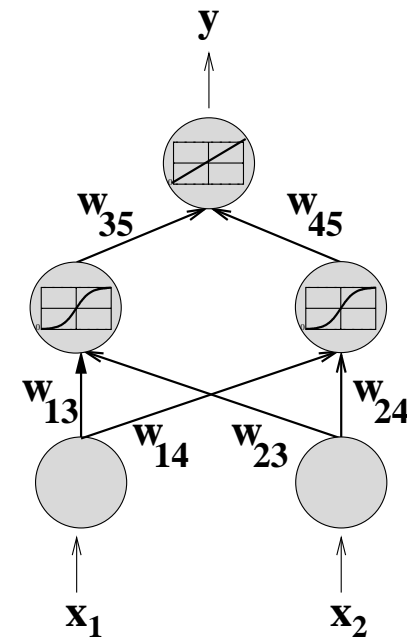
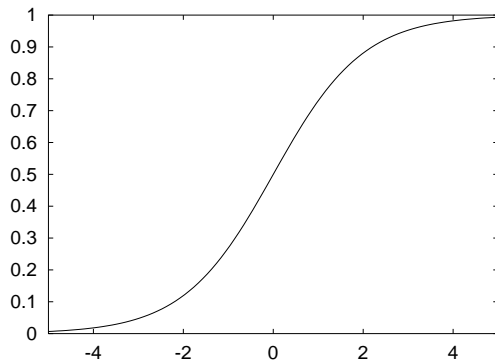


clearly outperforms capture–recapture!  
( 11 percent versus 24 )

# Non-Linear Regression: Neural Networks

$$\text{logist}(x) = \frac{1}{1 + e^{-x}}$$

$$s_i = \text{logist}\left(\sum_j w_{ji} \cdot s_j\right)$$



# Neural Network Topology

- number of inputs
- number of hidden layers
- number of units in hidden layers
- connections between layers

# Training a Neural Network

- fit regression function to training data
- non-linear optimization process (choose weights to minimize error on training data)
- no simple formula
- might get caught in local minimum
- train networks with different initial weights

# Input Data for Non-Linear Regression

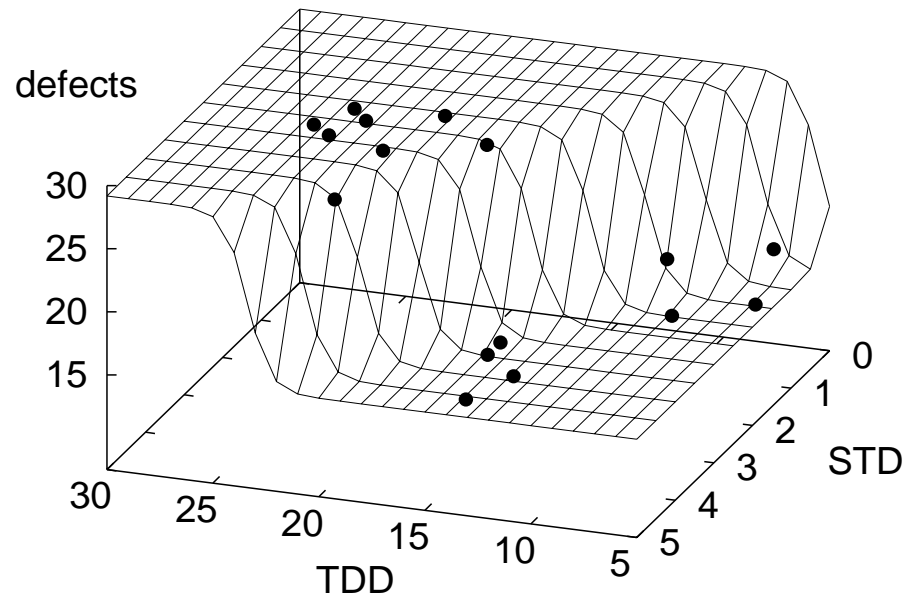
- non-linear feature selection yields ranking

TDD > STD > MAX > MIN > AVE

- STD instead of AVE
- some training patterns:

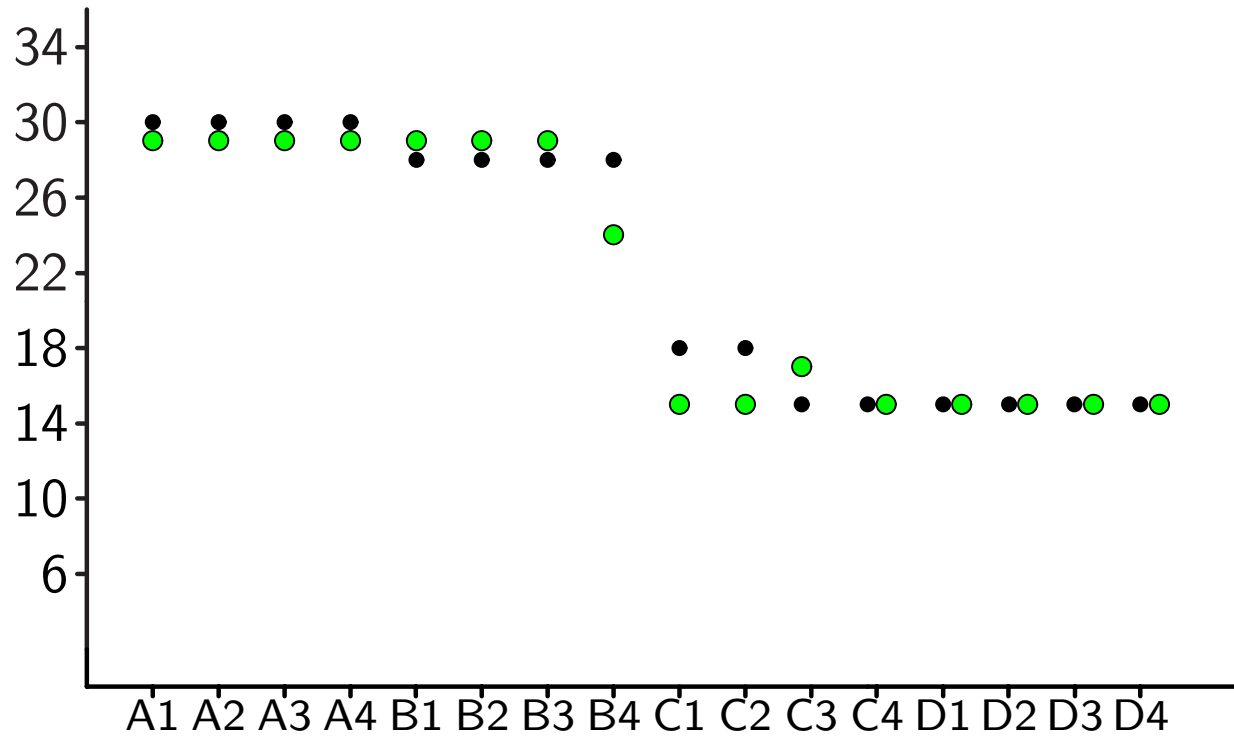
| inspection | TDD | STD | target |
|------------|-----|-----|--------|
| A1         | 23  | 2.4 | 30     |
| B1         | 20  | 1.7 | 28     |
| C1         | 10  | 1.5 | 18     |
| D1         | 6   | 1.4 | 15     |

# Non-Linear Regression Surface



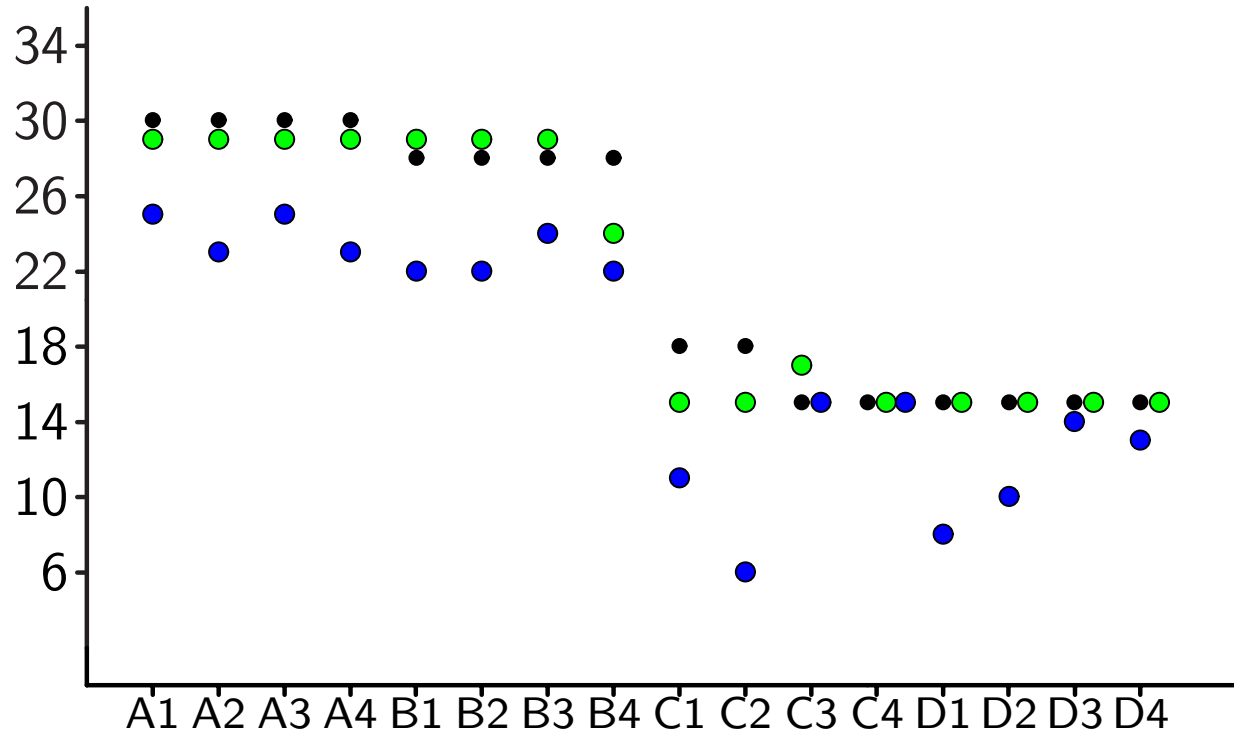
two hidden units in one layer; all 16 inspections  
surface fits data very well

# Neural Network Estimates



jackknife error of 6 percent  
max error of -17 percent

# Neural Networks versus Capture–Recapture

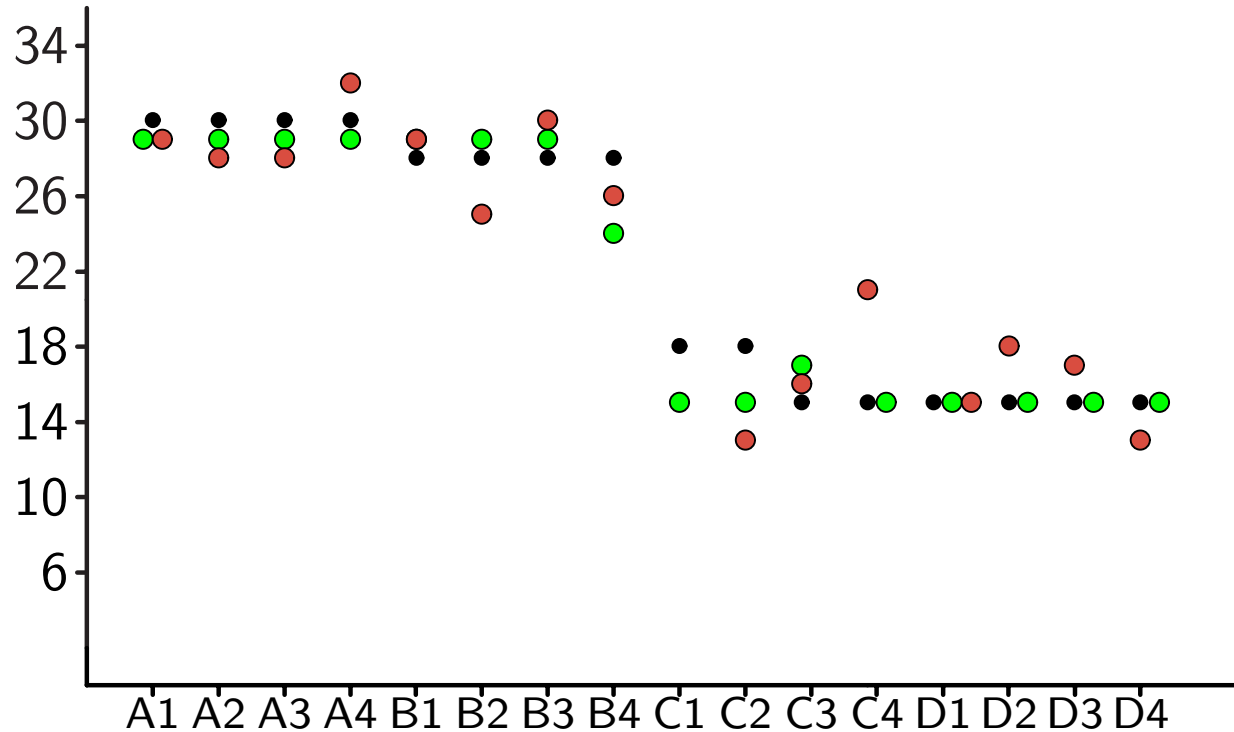


clearly outperforms capture–recapture!

( 6 percent versus 24 )



# Neural Networks versus Linear Regression



outperforms linear regression

( 6 percent versus 11; smaller variance )

# Neural Network Advantages

- much flexibility when fitting to data
- detects non-linearity in the data
- gives guidelines which features to use

# Neural Network Advantages

- much flexibility when fitting to data
- detects non-linearity in the data
- gives guidelines which features to use
- worked well with small benchmark dataset
- automatically adapted to different document types and sizes

# Result Summary

| Method             | mean abs. error | max error |
|--------------------|-----------------|-----------|
| Capture–Recapture  | 24 %            | –67 %     |
| Detection Profile  | 36 %            | 113 %     |
| Linear Regression  | 11 %            | 40 %      |
| Interval Estimates | (7 %)           | (14 %)    |
| Neural Networks    | 6 %             | –17 %     |

novel approaches are promising!  
need more empirical data for validation

# Own Publications About the Defect Content Estimation Problem

- *Empirical Interval Estimates for the Defect Content After an Inspection*  
International Conference on Software Engineering ICSE (2002)
- *Applying Machine Learning to Solve an Estimation Problem in Software Inspections*  
International Conference on Artificial Neural Networks ICANN (2002) (with T. Ragg and R. Schoknecht)  
accepted for Transactions on Software Engineering TSE

## Own Publications (cont.)

- *A Fast Algorithm to Compute Maximum Likelihood Estimates for the Hypergeometric Software Reliability Model*  
Asia-Pacific Conference on Quality Software APAQS (2001)
- *Maximum Likelihood Estimates for the Hypergeometric Software Reliability Model*  
International Journal of Reliability, Quality and Safety Engineering IJRQSE (2003)

**Thank You !**