Applications of Nonlinear Regression Methods in Insurance

Douglas McLean, Actuarial Teachers and Researchers Conference, Edinburgh

Dec 2014

Agenda

1. Features of a Good Proxy
2. Multiple Polynomial Regression
3. Artificial Neural Networks
4. Motivating Example
5. Neural Network Analysis in more detail
6. Conclusion

Motivation

» Proxy Generator uses multiple polynomial regression in LSMC which
  - is a well known and robust statistical method
  - has great intuitive appeal
  - has straight-forward formulae
  - uses a simple forward stepwise approach to find a "best" model

» Many proxy generation problems can successfully rely upon polynomials

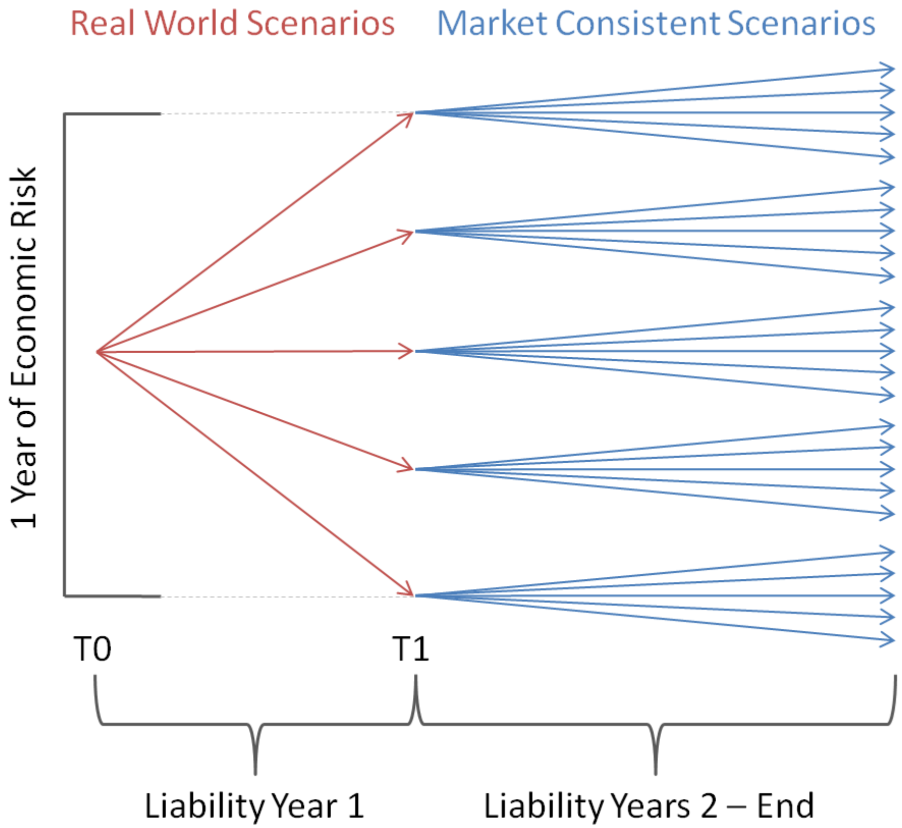
» In our experience, we do see a small number of problems which are more challenging

» To avoid too much analyst intervention for the more challenging fits when hundreds of proxies are needed, is there an alternative regression technique we can rely on?

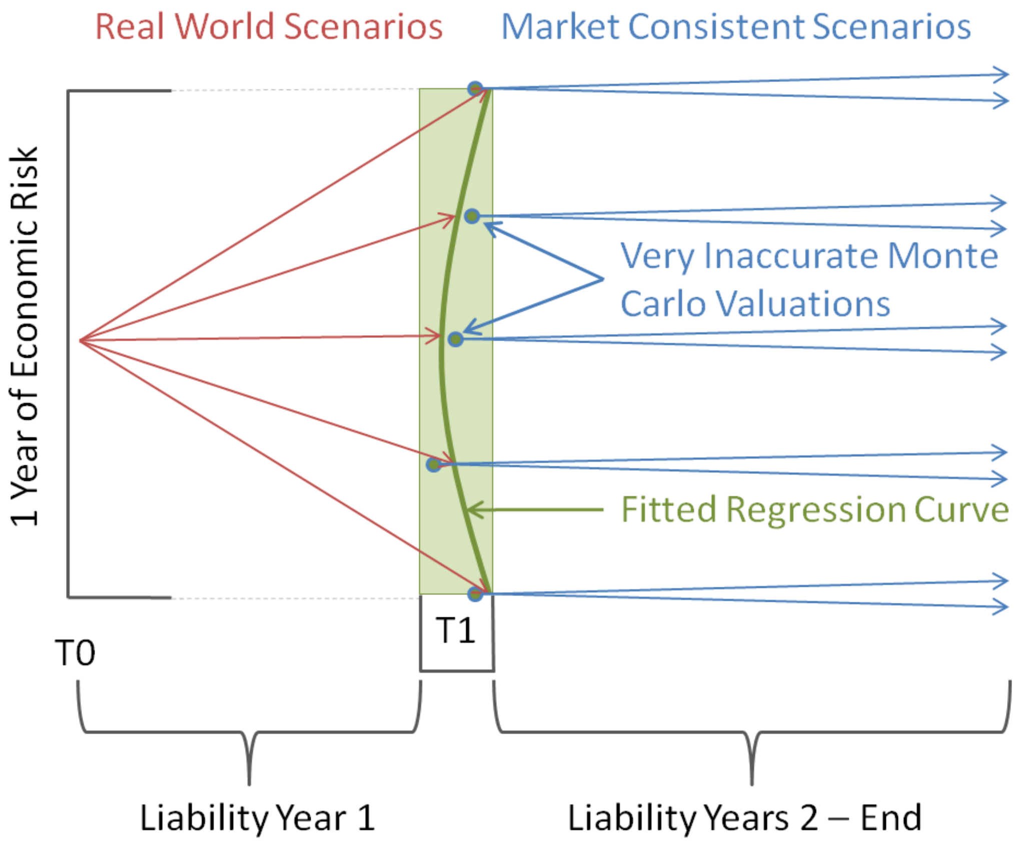
» In this presentation we ask "what other techniques are out there?"

Nested-stochastic simulations

Solvency 2 Regulations Require a “downside risk” measurement



Least Squares Monte-Carlo Solution



Features of a Good Proxy

Features of a Good Proxy: I

- *Parsimony*
  - It should use a minimally sufficient set of risk drivers (including powers and cross terms)

- *Compatibility with downstream software*
  - Ease of communication with downstream software
  - It should use a relatively small number of parameters in a succinct representation

- *Good validation on “accurate” Validation Scenarios*

- *High goodness-of-fit measure without over-fit*
  - The in-sample R-squared should be as high as possible
  - The out-of-sample R-squared should be as close as possible to the in sample R-squared

Features of a Good Proxy: II

- *Unbiased predictions of minimum variance*
  - Any evidence of systematic over- or under-estimation in the model predictions is evidence of bias
  - This often involves trading bias against variance in finding an optimal estimator

- *Scalability to high dimensions*
  - For large numbers of risk drivers and fitting scenarios, the memory requirements and the time taken can become considerable
  - When a large number of parameters are being estimated, their standard errors are large and our ability to recover a meaningful model is reduced

- *Short model fitting time*

- *Good model specification*
  - Proxy models which are well specified will be able to approximate arbitrarily closely the underlying data generation process, given enough fitting scenarios

Alternative Regression Methods for LSMC

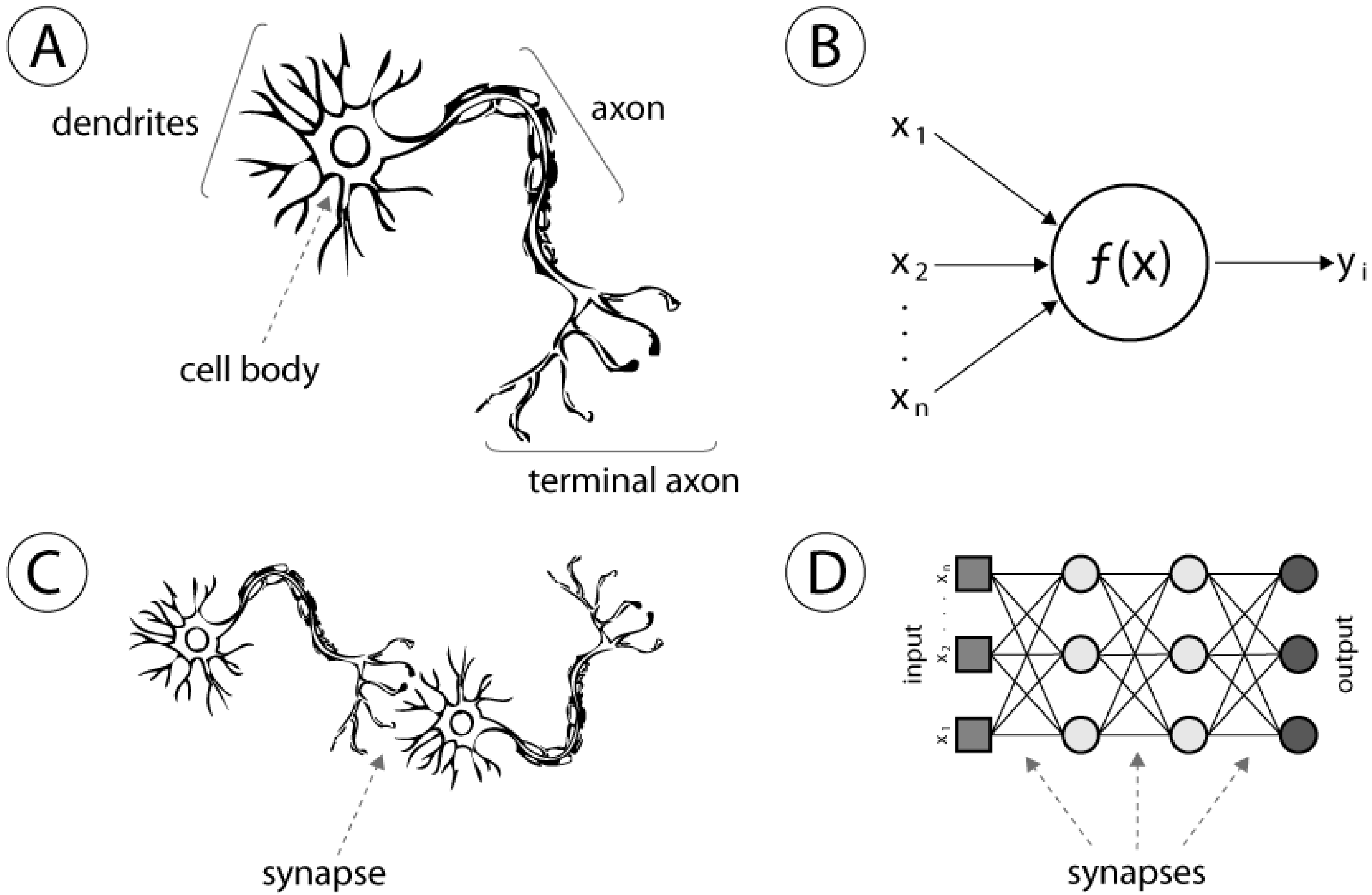
- Examples of linear and nonlinear regression methods:
  - Mixed Effects Multiple Polynomial Regression
  - Generalized Additive Models
  - Artificial Neural Networks
  - Regression Trees
  - Finite Element Methods
- In other work we have considered local regression methods such as
  - kernel smoothing and
  - loess / lowess
- In this presentation we consider the merits of artificial neural networks

Artificial Neural Networks

Artificial Neural Networks

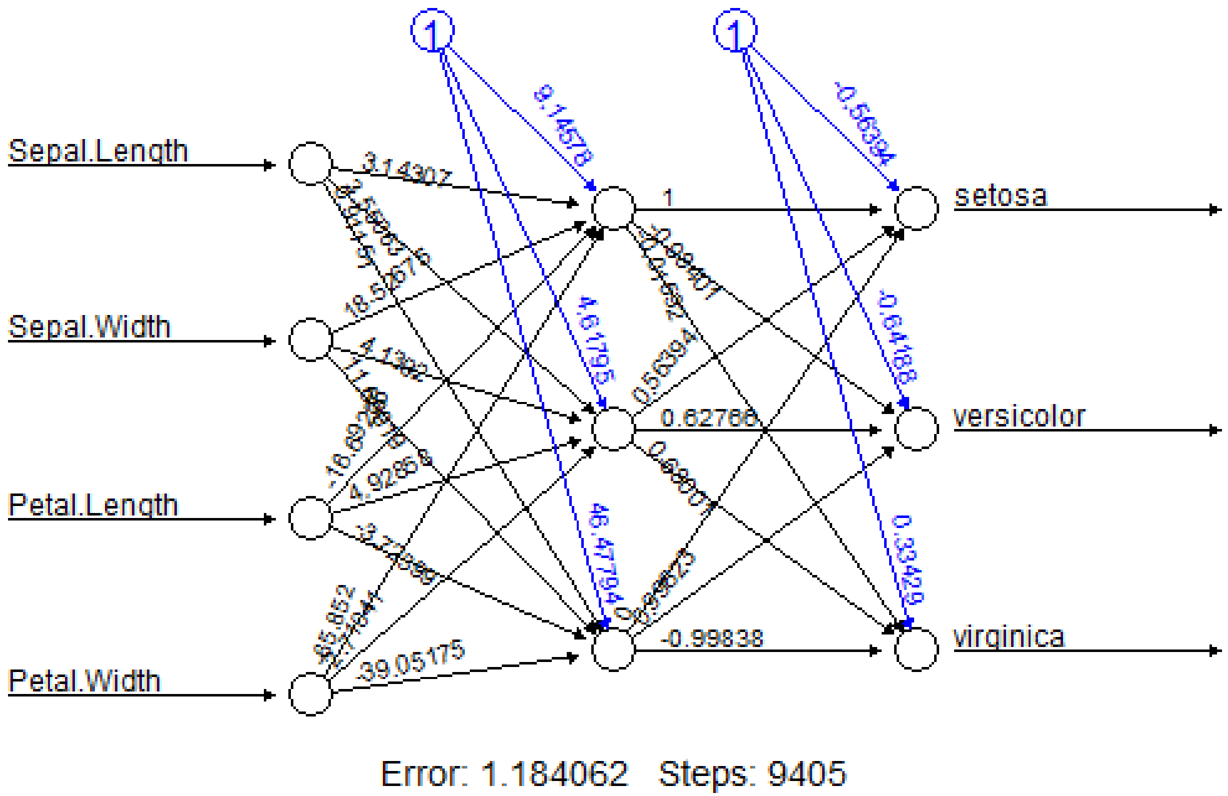
- These were simultaneously invented by the computer science and statistics communities
- They have a heritage of being used in
  - Classification problems such as in spam filters or shopping preferences, learning as they “see” more and more data
  - They are a natural alternative to logistic regression problems
  - They can also be used as nonlinear regression tools
- They also have the unfortunate heritage of being known as “black-box” techniques with little intuitive appeal – they *just work*
- They are often quoted as being *accurate* but subject to *over-fitting* at the same time
- However, if we think of them as nonlinear regression tools then they are simple statistical constructs with parameters to be found by minimizing the mean squared prediction error

But what *is* a neural network?



Neural Network Structure

Input layer / hidden layer / output layer



Formulae

Both multiple polynomials and neural networks have similar functional forms

Polynomial Regression

$$y_k = \beta_0 + \sum_{i=1}^{p} \beta_i^{(1)} x_i + \sum_{i=1}^{p} \sum_{j=1}^{i} \beta_{ij}^{(2)} x_i x_j + \text{higher order terms}$$

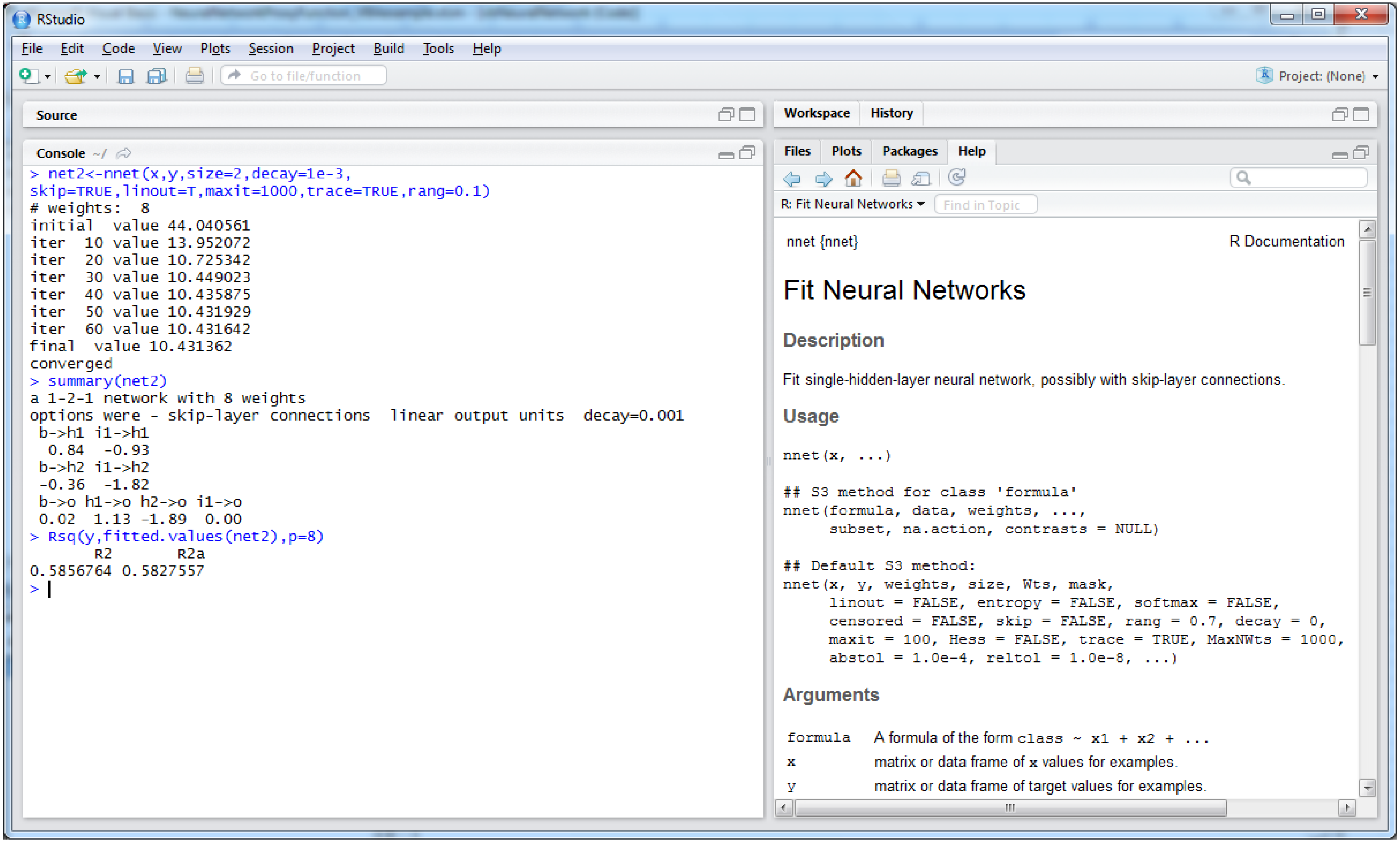
Neural Network and Activation Function

$$y_k = \alpha_k + \sum_{i \to k} w_{ik} x_i + \sum_{j \to k} w_{jk} \varphi\left(\alpha_j + \sum_{i \to j} w_{ij} x_i \right), \quad \varphi(x) = \frac{1}{1 + \exp(-x)}$$

VBA Implementation

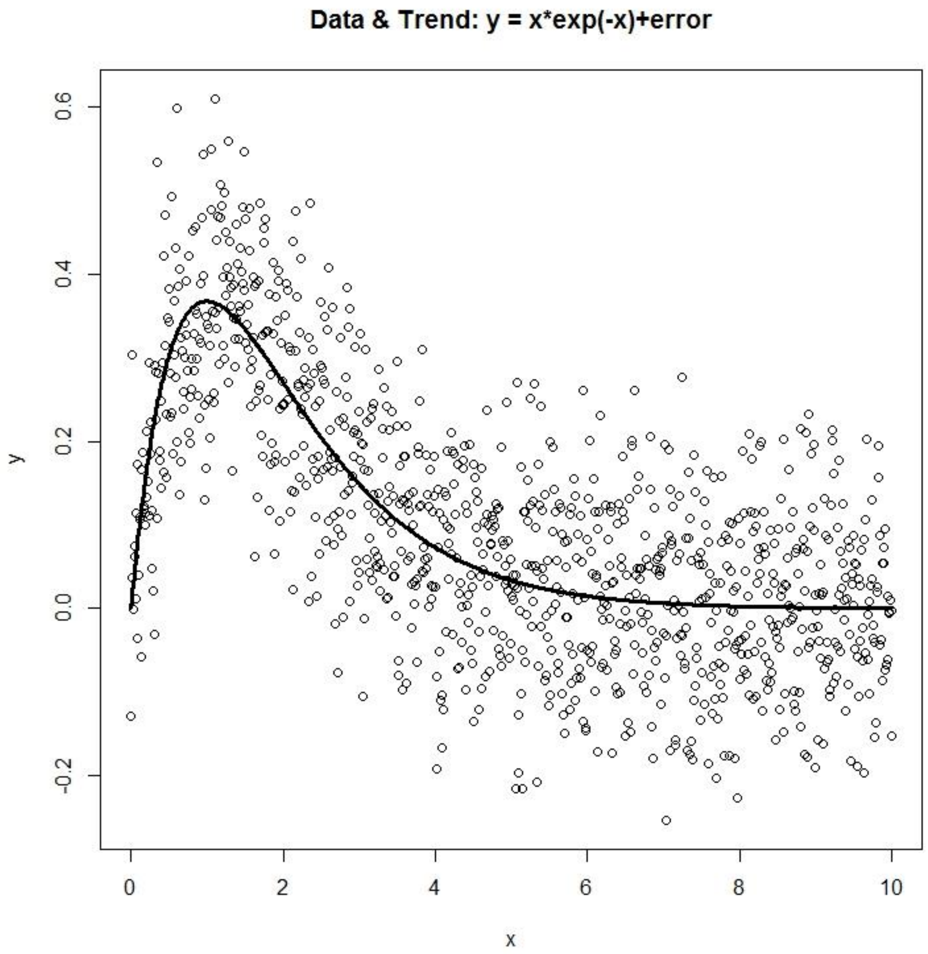
```vba
'Step 1: Scale inputs
For i = 1 To nI
    x(1, i) = x(1, i) / s(i, 1)
Next i

'Step 2: Predict scaled neural network outputs
For k = 1 To nO
    y = 0
    For j = nI + 2 To nI + nH + 1
        'Outer loop to calculate weighted sums from each hidden node at k-th output node
        sum = 0
        For i = 2 To nI + 1
            'Inner loop to calculate weighted sums of inputs at each hidden node
            'and then pass through the logit activation function
            sum = sum + w(i, j - nI - 1) * x(1, i - 1)
        Next i
        'Contribution to each hidden node from the first bias node
        sum = sum + w(1, j - nI - 1)
        'Hidden layer contribution to k-th output node
        y = y + w(j, nH + k) * logit(sum)
    Next j
    For i = 2 To nI + 1
        'Skip layer contributions at k-th output node
        y = y + w(i, nH + k) * x(1, i - 1)
    Next i
    'Final contribution to k-th output from second bias node
    y = y + w(1, nH + k)
    'Reintroduce scale and store results
    output(k) = y * s(nI + k, 1)
Next k
vbNeuralNetworkProxyFunction = output

End Function

'Simple implementation of the logistic function to be used as the neural network's activation f
Function logit(ByRef x As Double) As Double
logit = 1 / (1 + Exp(-x))
End Function
```

Fitting a Neural Network

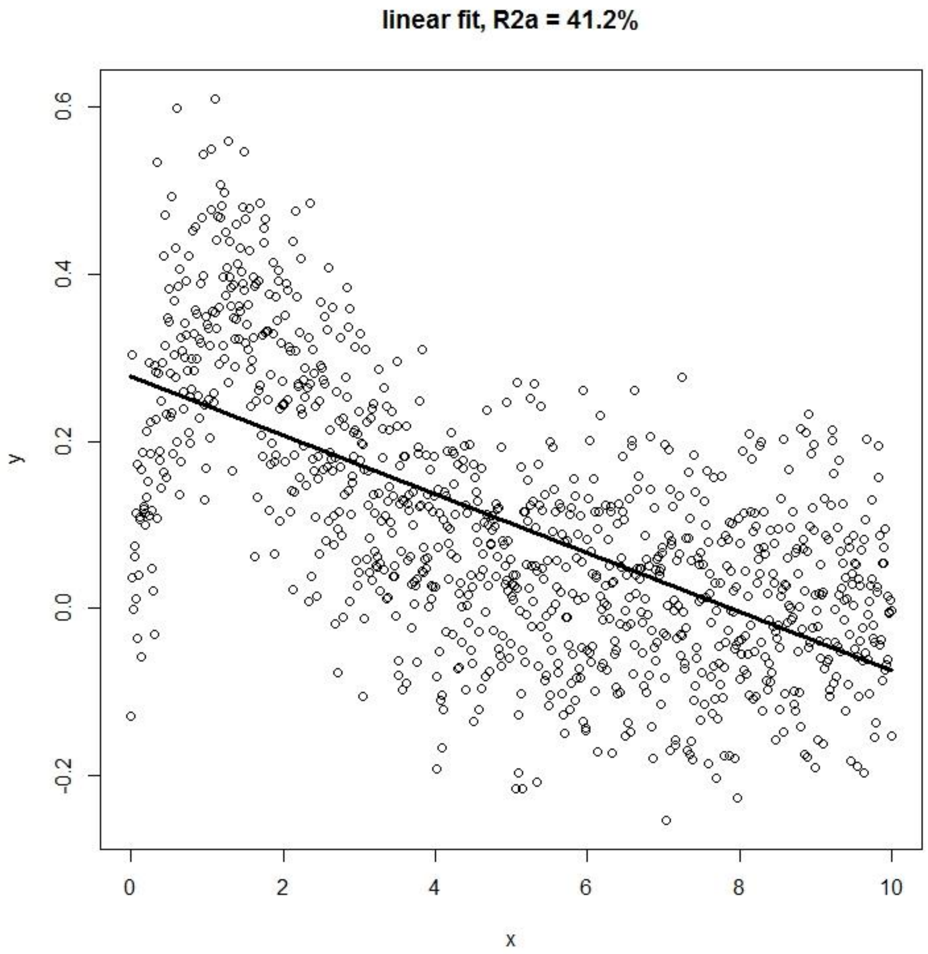


Nonlinear edge case example

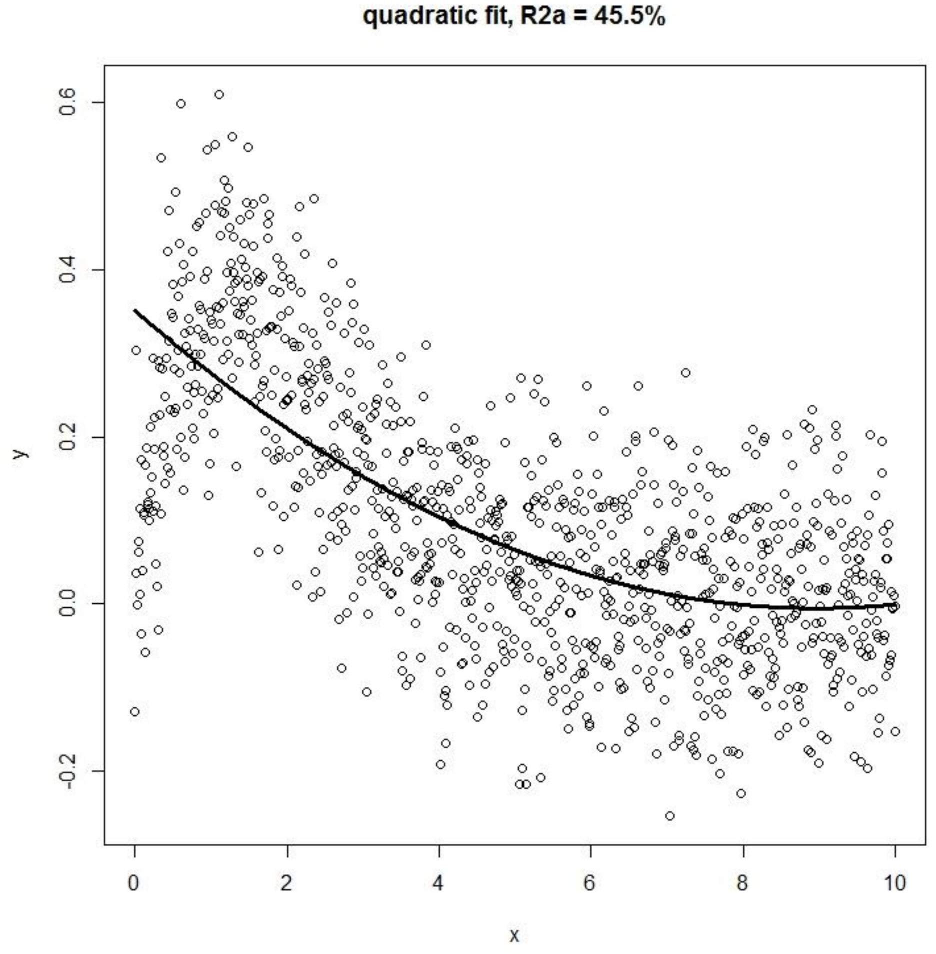
Example 1000 pairs x, y with normal errors (sd 0.1)



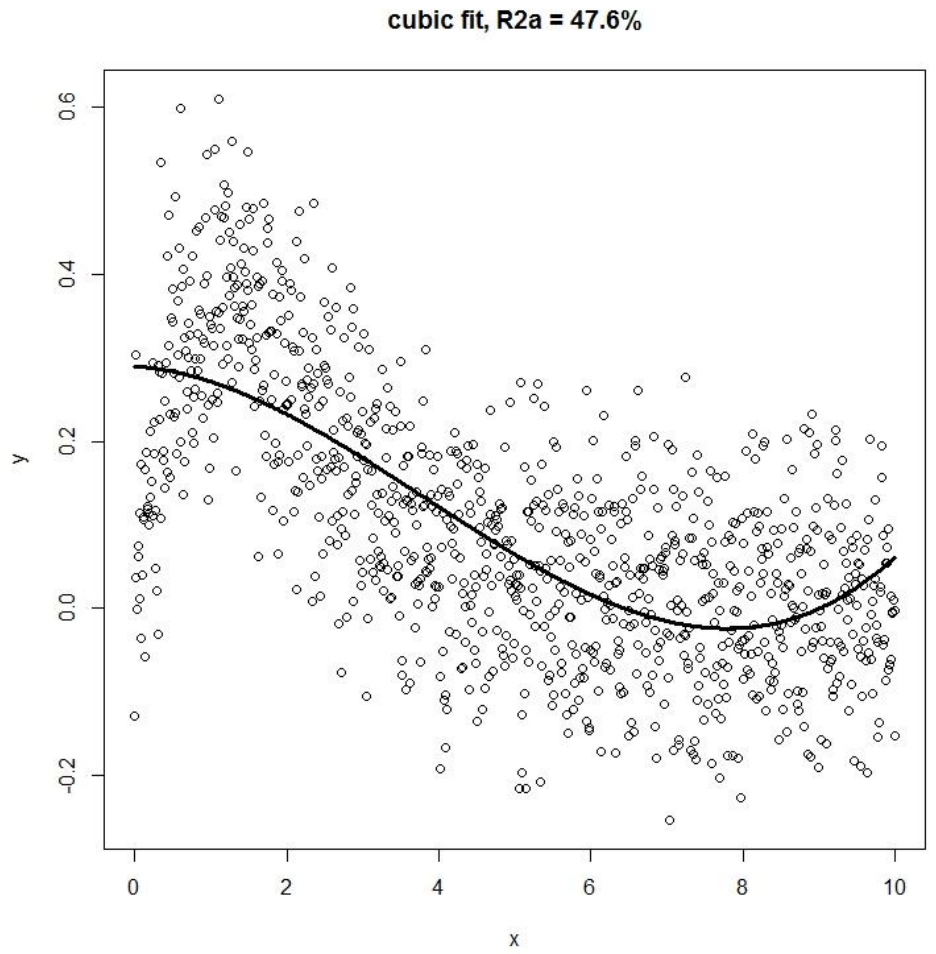
Degree 1 polynomial fit



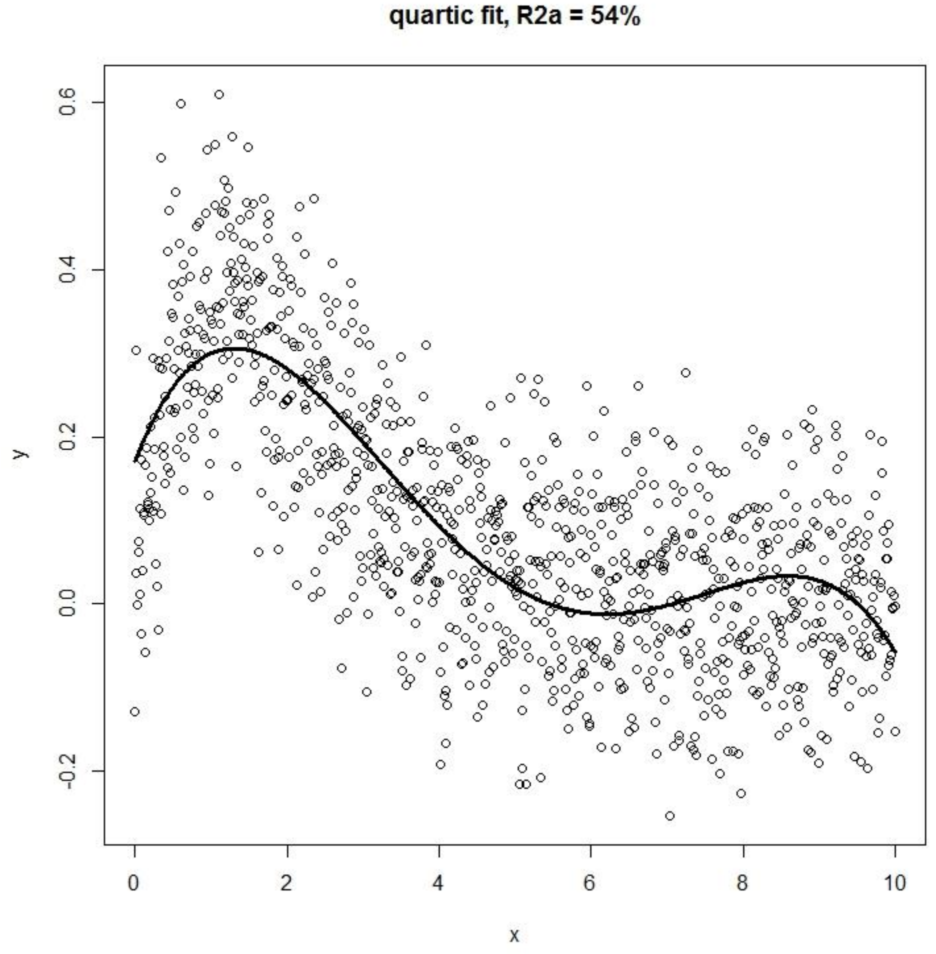
Degree 2 polynomial fit



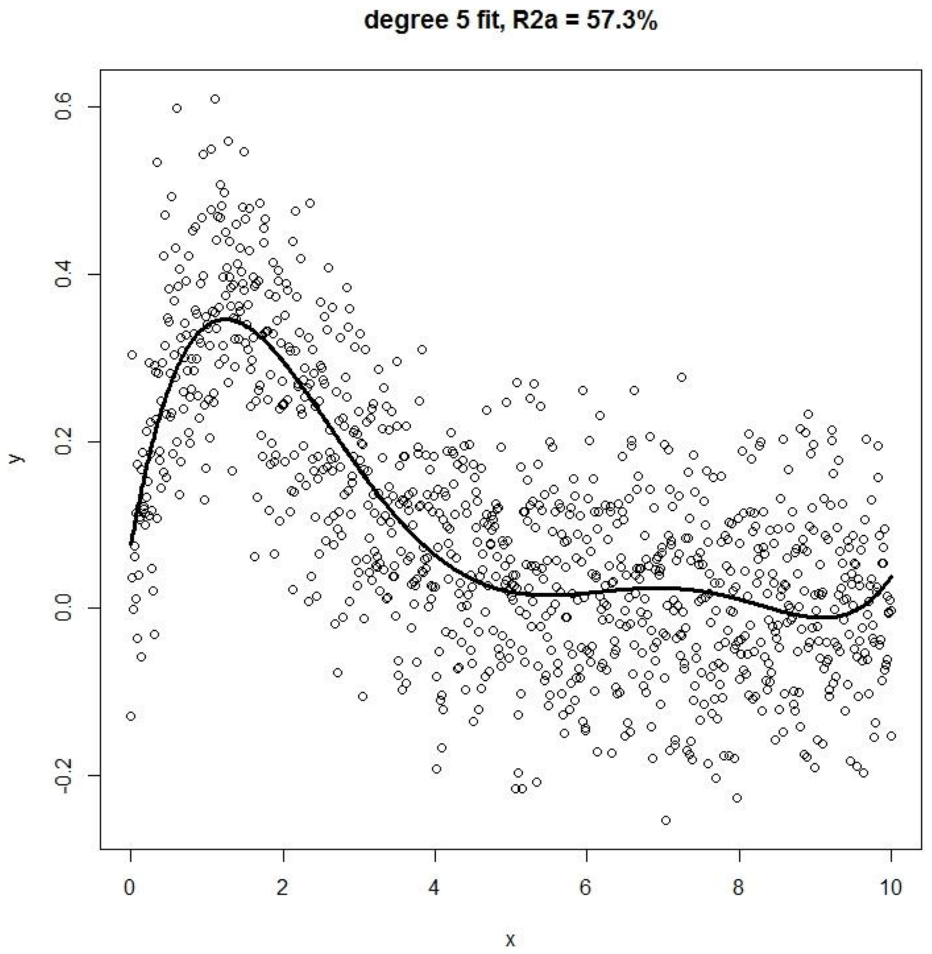
Degree 3 polynomial fit



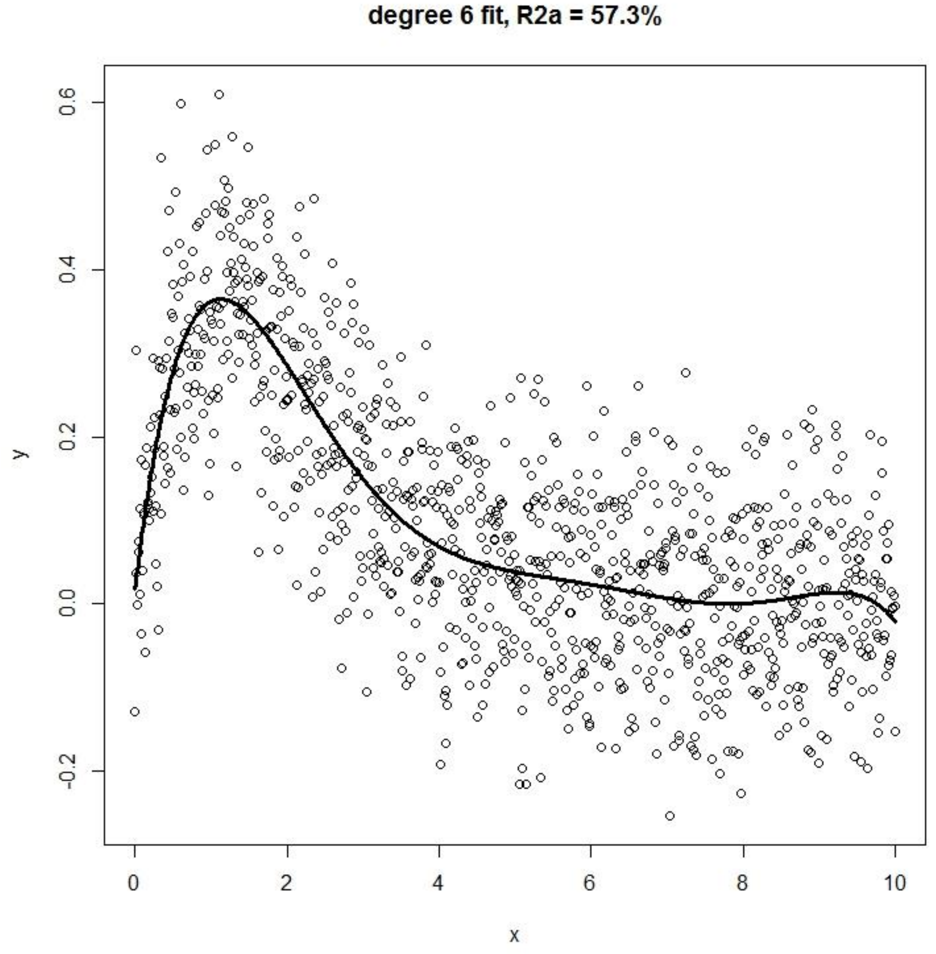
Degree 4 polynomial fit



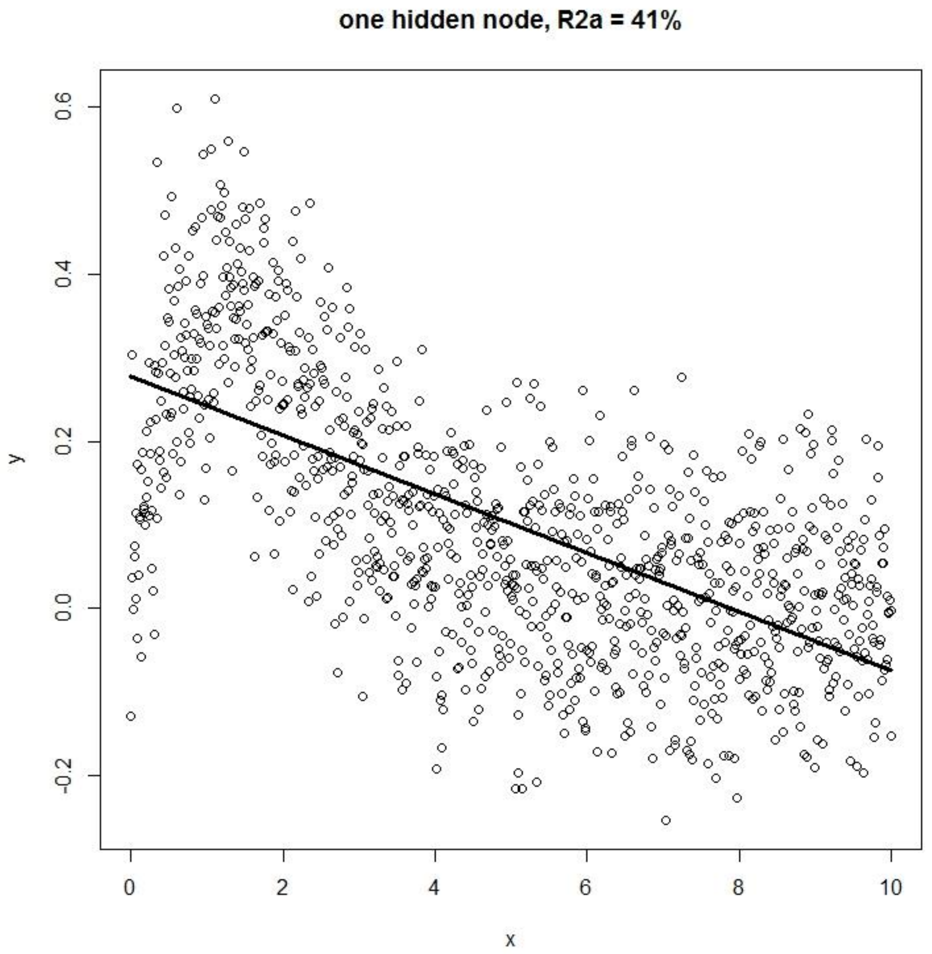
Degree 5 polynomial fit



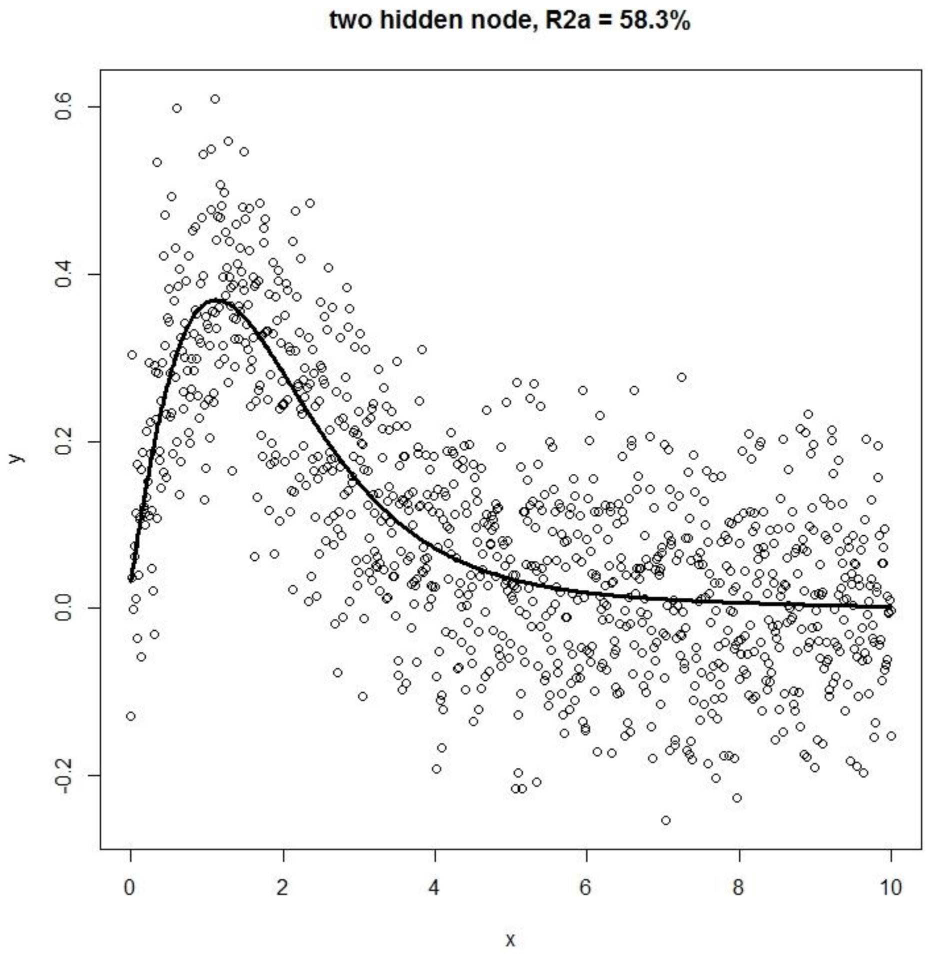
Degree 6 polynomial fit



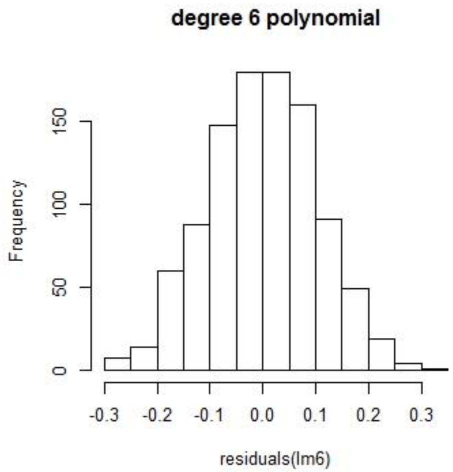
Neural network one hidden node fit

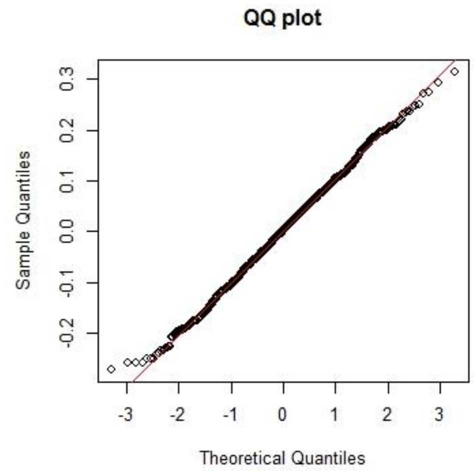


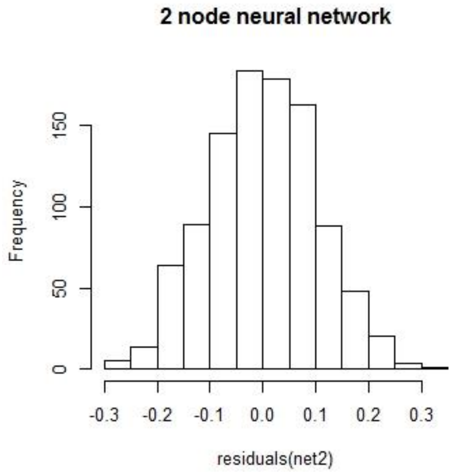
Neural network two hidden node fit

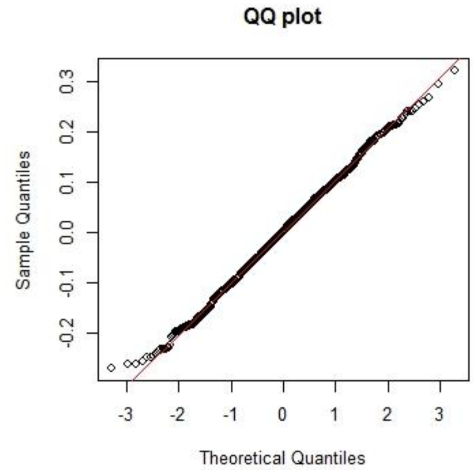


Residuals Analysis

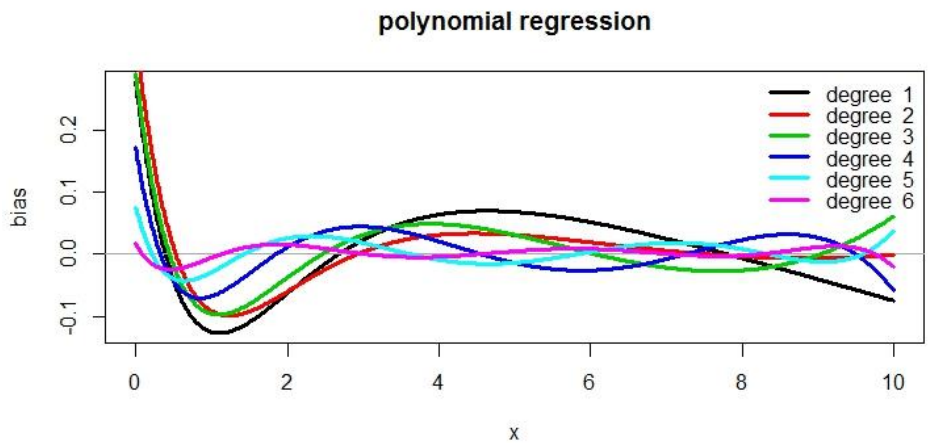


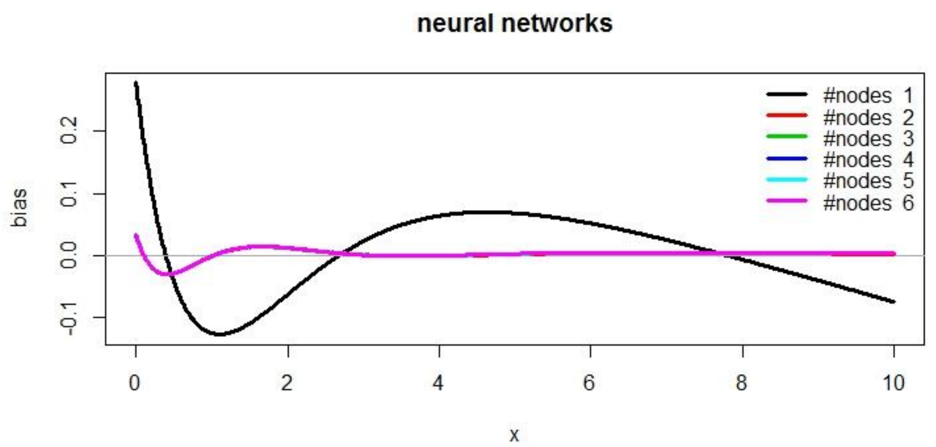






Actual trend subtract the fit





Motivating example

Variable Liability Value

- Life policy has an embedded guarantee of 3.25%
- Involved 9 risk-drivers including equity level and volatility, real and nominal yield curve factors and credit in addition to some non-market risks.
- The exercise was to model the liability in a single time-step / static regression problem.
- Firstly, a multiple polynomial regression was performed
  - up to cubic degree in each risk-driver
  - using a layered forward stepwise approach
  - without term removal
- Secondly, a neural network in 9 input nodes, a bias node, 2 hidden nodes and a skip layer connection was fitted to the same data.

Variable Liability Value (continued)

N=25,000	Regression	Neural Network
Time Taken (seconds)	3797 (1 hr. approx.)	75
Number of terms/weights	52	44
In sample R-squared	72.30%	69.38%
Out of sample R-squared	72.23%	69.28%

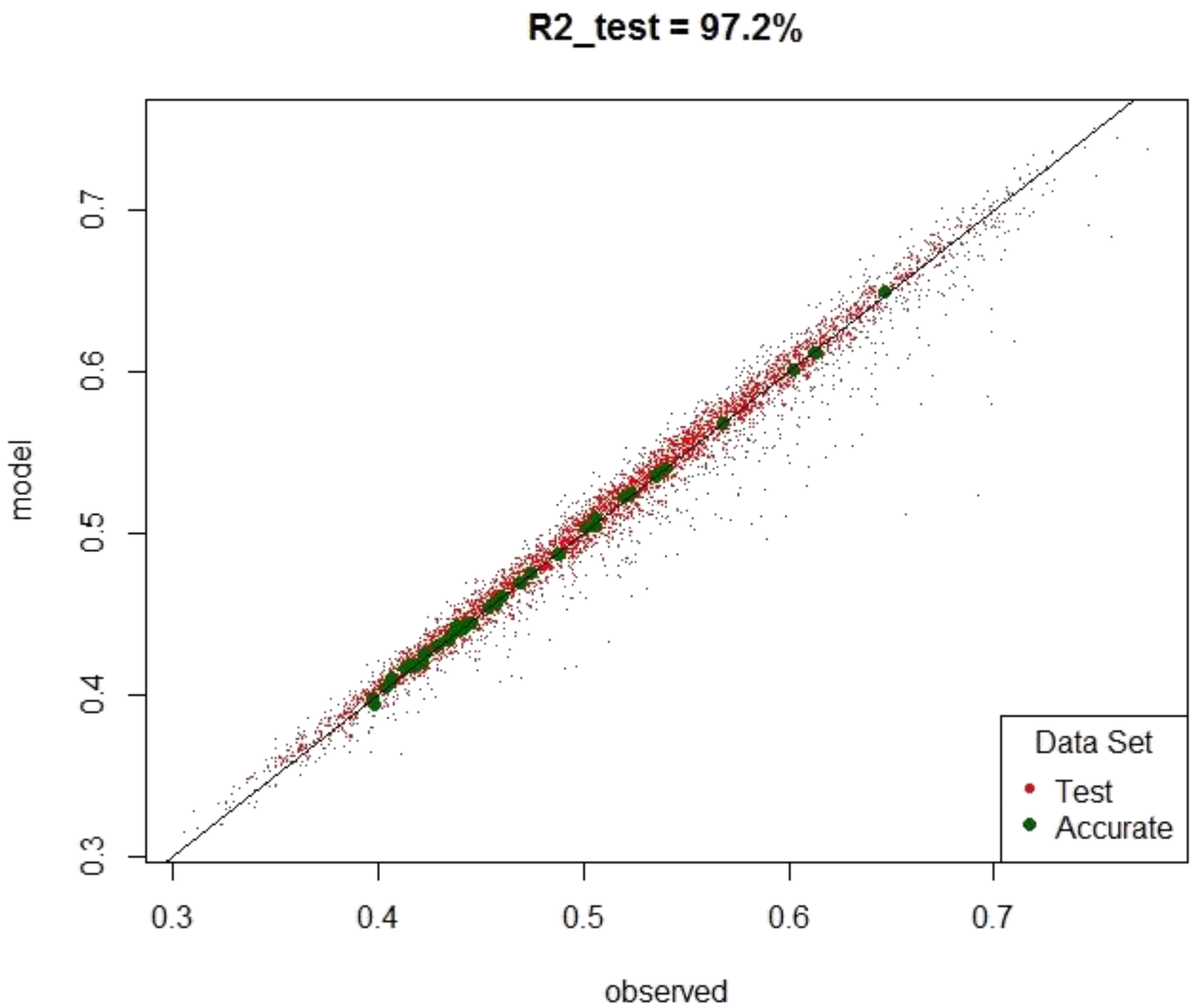
The out-of-sample R-squared is calculated by 10-fold cross validation

Network Analysis in more detail

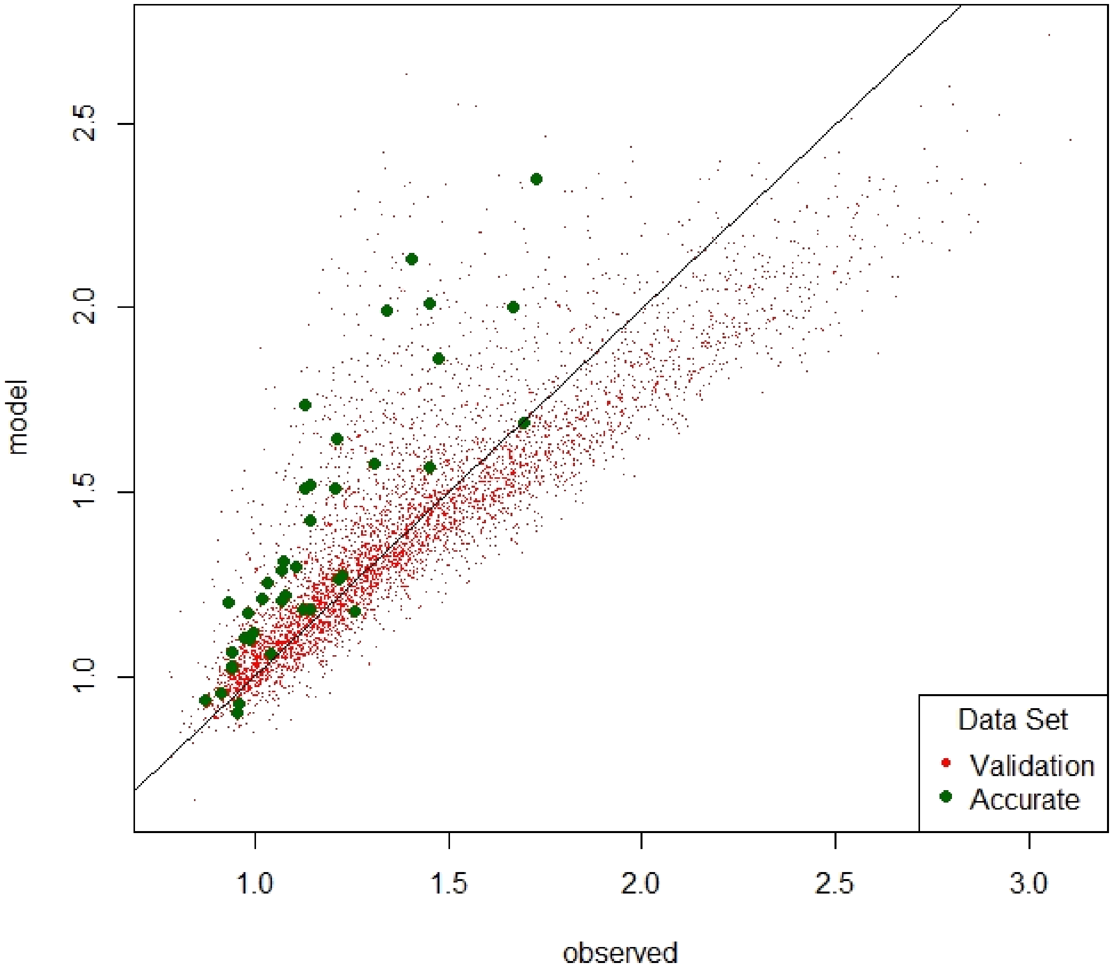
Neural Network Analysis

- Fitting a network involves determining the network weights over a selection of hidden layer sizes and regularisation parameter values
- 25,000 fitting scenarios are split into:
  - 15,000 training scenarios to determine the network weights
  - 5,000 validation scenarios to determine the hidden layer size and weight decay
  - 5,000 test scenarios to assess the network on new / unseen scenarios
- We use the validation set to determine how many scenarios we need
- Illustrate the bias / variance trade-off with hidden layer size and weight decay
- Describe how to deal with heteroscedastic effects

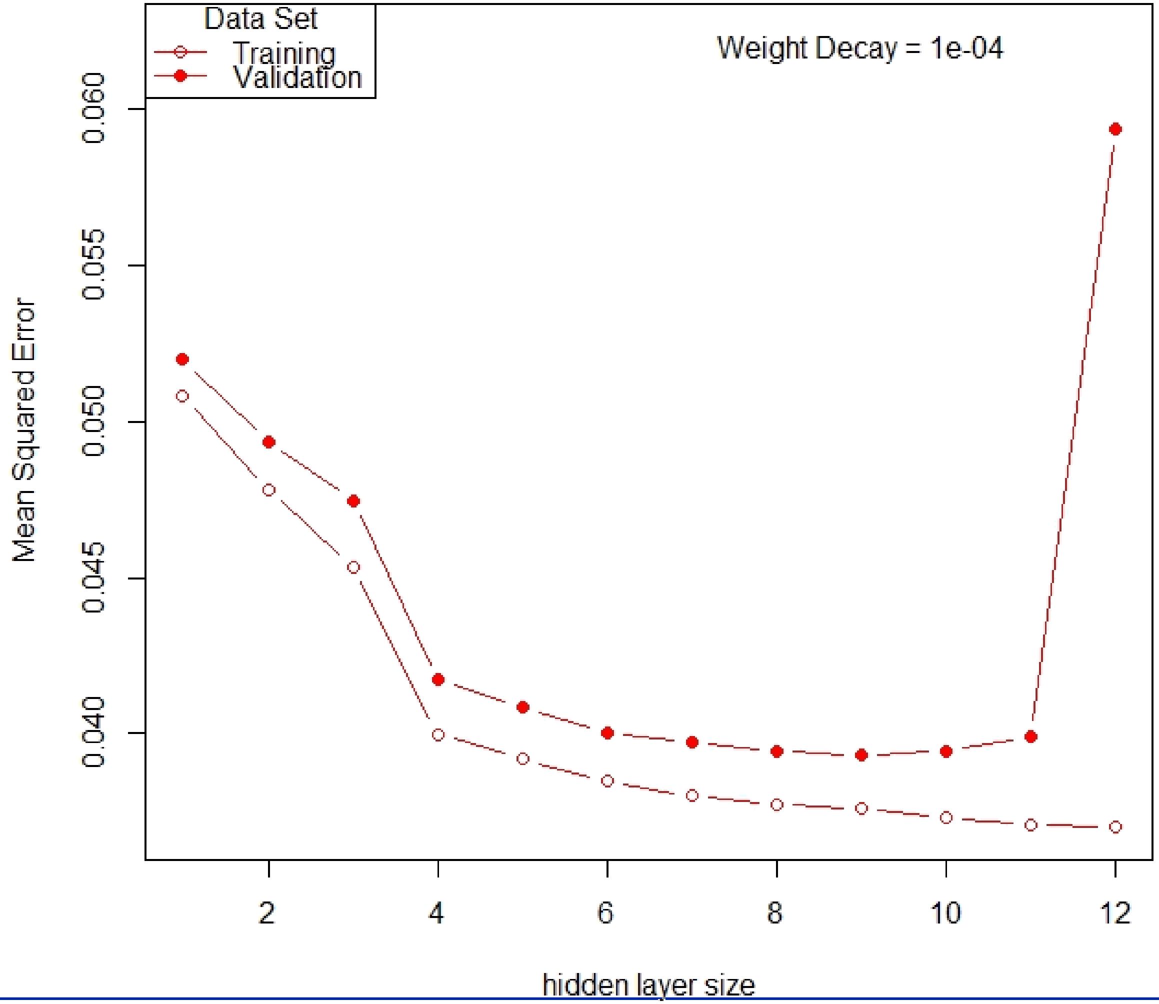
Good model output…



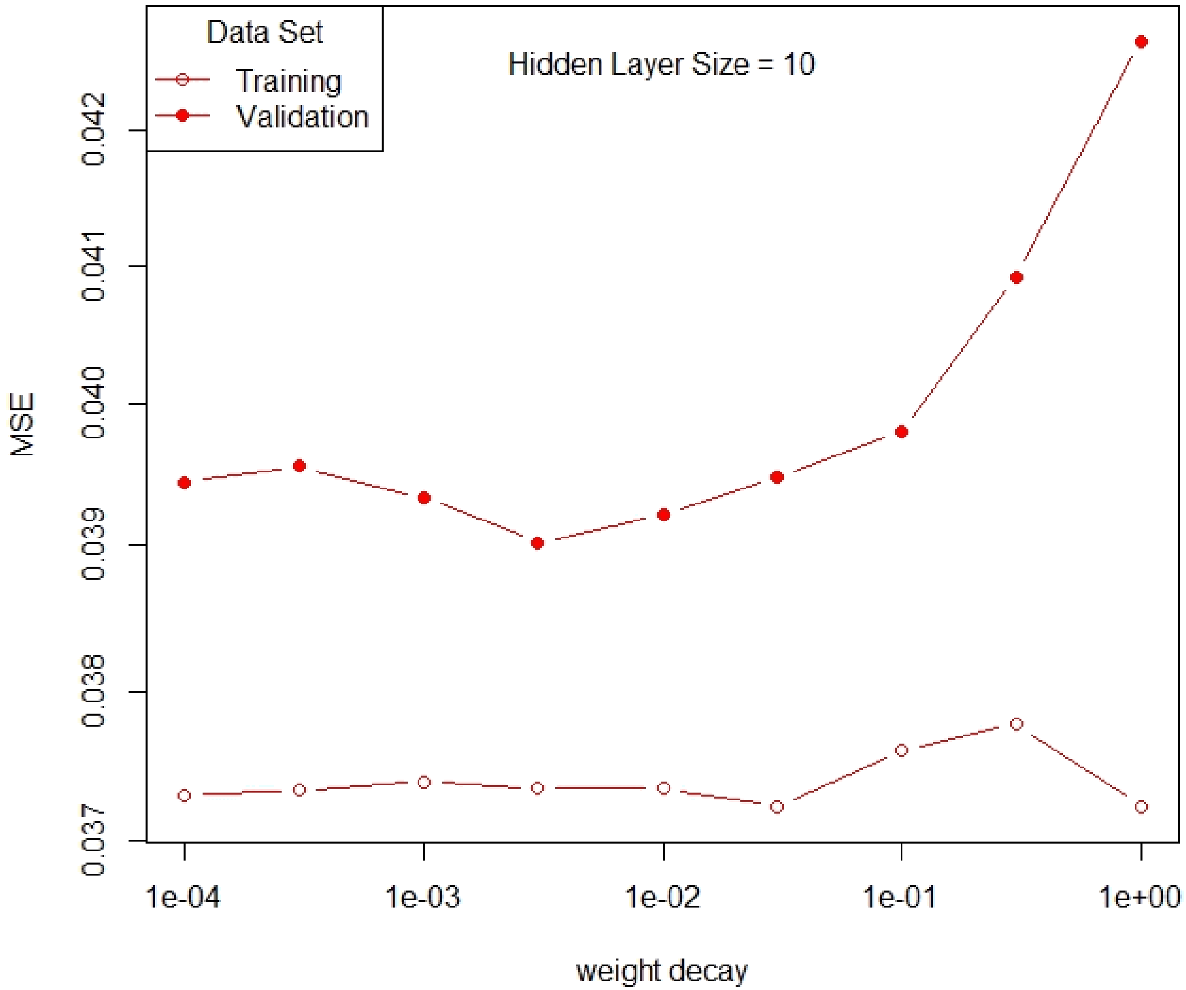
A Challenging Fit!



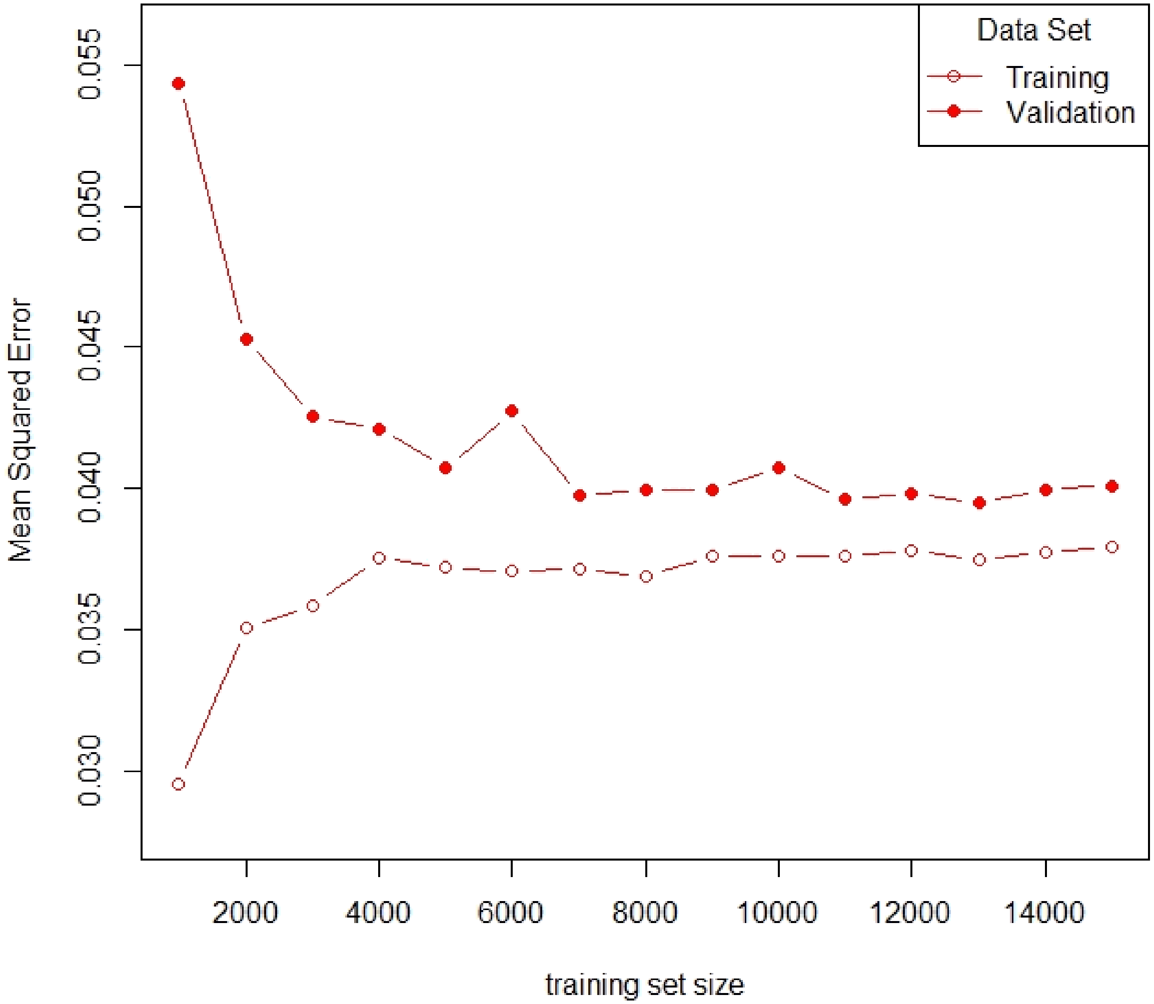
Bias-Variance Trade-off I: for fixed weight decay



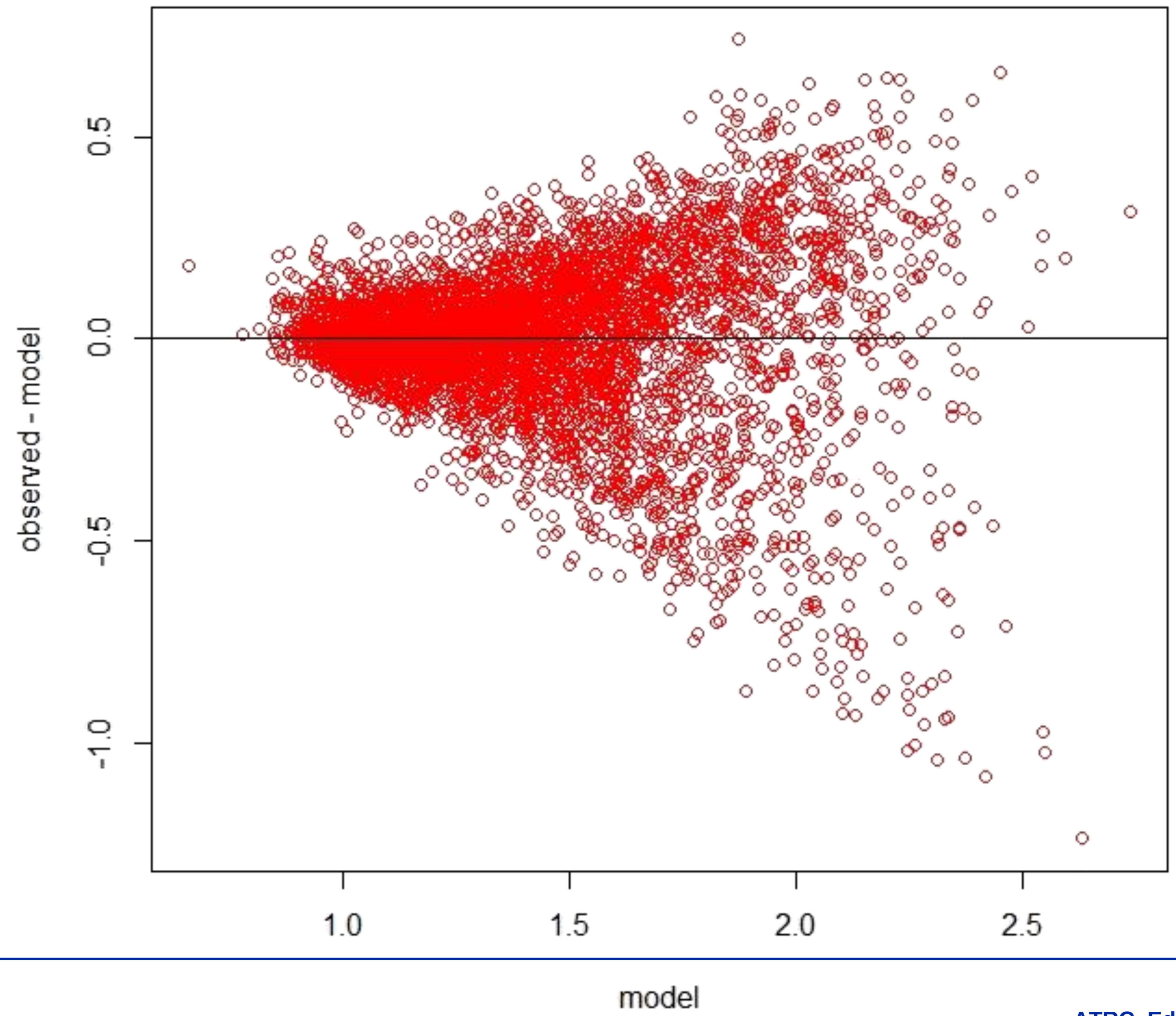
Bias-Variance Trade-off II: for fixed hidden layer size



Variation with Model Size and Fitting Scenario Budget



Heteroscedasticity



Conclusion

- Multiple polynomial regression is a robust and practical solution to the proxy generation problem working in the majority of cases
- Some proxy problems can be more challenging
- Alternative methodologies exist including generalised additive models, local regression methods and artificial neural networks
- We investigated one of these alternative approaches, neural networks, with a view to perhaps including it as an option within ProxyGenerator in the future
- Neural networks work at least as well as multiple polynomial regression
- Bias-variance trade-off and optimal scenario counting was discussed alongwith methods to counteract heteroscedastic effects