Model comparison, Over-fitting, Information Criteria

Philippe Rast

PSC 204B
UC Davis, Winter 2018

Topics

Single-level Regression:

Week 1 Linear Regression (G&H: 3,4)
Week 2 Multiple Regression
Week 3 Violation of Assumptions
Week 4 Logistic Regression and GLM (G&H: 5, 6)
Week 5 Over-fitting, Information Criteria and Model comparison (McE: 6)
Week 6 Regression inference via simulations (G&H: 7–10)

Multilevel Regression:

Week 7 Multilevel Linear Models (G&H: 11–13)
Week 8 Multilevel Generalized Models (G&H: 14, 15)
Week 9 Bayesian Inference (G&H: 18 / McE: 1, 2, 3)
Week 10 Fitting Models in Stan and brms (G&H: 16, 17 / McE: 11)

Overview

Overview

- What is a good model in terms of fit and number of parameters?
  - "law of parsimony"
- Problem-solving principle attributed to William of Ockham (1287–1347)
  - Among competing hypotheses, the one with the fewest assumptions should be selected.
- Commonly known as "Ockham's razor" (or "Occam's razor")
- Heuristic guide, not a scientific method

English Franciscan friar and scholastic philosopher

Ockham's Razor

- Dilemma:
  - Too simple
  - Too complex
- Razor can be hard to use more generally
- Usually we must choose among models that differ in both
  - accuracy
  - simplicity.
- How are we to trade these different criteria against one another?

Over and Underfitting

- Parsimony is not necessarily the best tool
- Two main statistical issues:
- **Overfitting**, which leads to poor prediction by learning too much from the data
- **Underfitting**, which leads to poor prediction by learning too little from the data

- We need statistical tools to address this dilemma:
- Some notion of simplicity is usually present in all of the tools, and so each is commonly compared to Ockham's razor.
- Each tool is equally about improving predictive accuracy.
- Explicitly trade-off of accuracy and simplicity

Approaches that Deal with Overfitting

Two common families of approaches

- Penalized likelihood to regularize parameters
  - Take into account model complexity when estimating parameters of different models
  - Instead of doing simple maximum likelihood estimation, log-likelihood is maximized minus a penalty term
  - Penalty depends on the model and generally increases with the number of parameters
  - Basic idea: Model should not get too excited about data
- Information Criteria
  - Scoring device: Model the prediction task and estimate predictive accuracy for some purpose
  - Model a number of different models and compare

Examples of Overfitting

- We let data speak too much for itself

E.g. We want to predict if a student will land a job interview based on her resume.

- We train a model (find optimal parameter and values) from a dataset of 10,000 resumes and their outcomes
- Next, we try the model out on the original dataset, and it predicts outcomes with 99% accuracy!

! Bad news:

- When we run the model on a new ("unseen") dataset of resumes, we only get 50% accuracy

▷ Our model doesn't *generalize* well from our training data to unseen data.

Examples of Overfitting

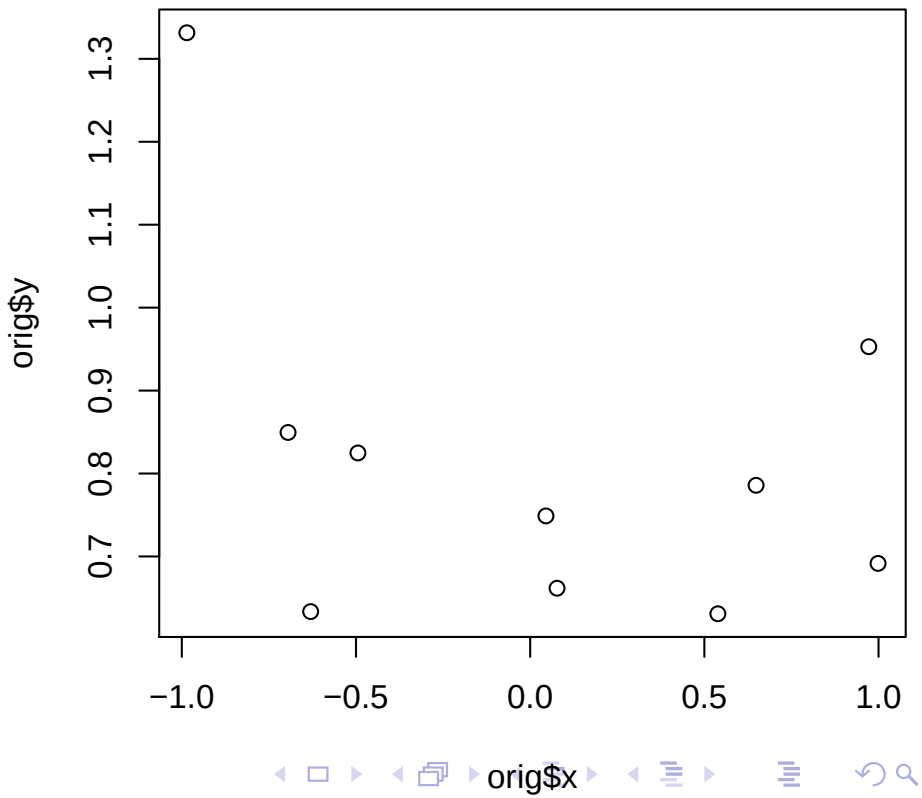
Simulated

```
orig <- data.frame(x=sort(runif(N, -1, 1)))
orig$y = 0.5 - 0.2*orig$x +0.5*orig$x^2 + rnorm(N, 0, 0.1)
```

Model: $y_i \sim N(\mu, \sigma)$

$$\mu = 0.5 - 0.2x_i + 0.5x_i^2$$

$$\sigma = 0.1$$



Examples of Overfitting

Simulated

```
mp0 <- lm(y ~ x, data=orig)
mp7 <- lm(y ~ x + I(x^2)+ I(x^3)+ I(x^4)+ I(x^5) +
     I(x^6)+ I(x^7)+ I(x^8),  data=orig)
```

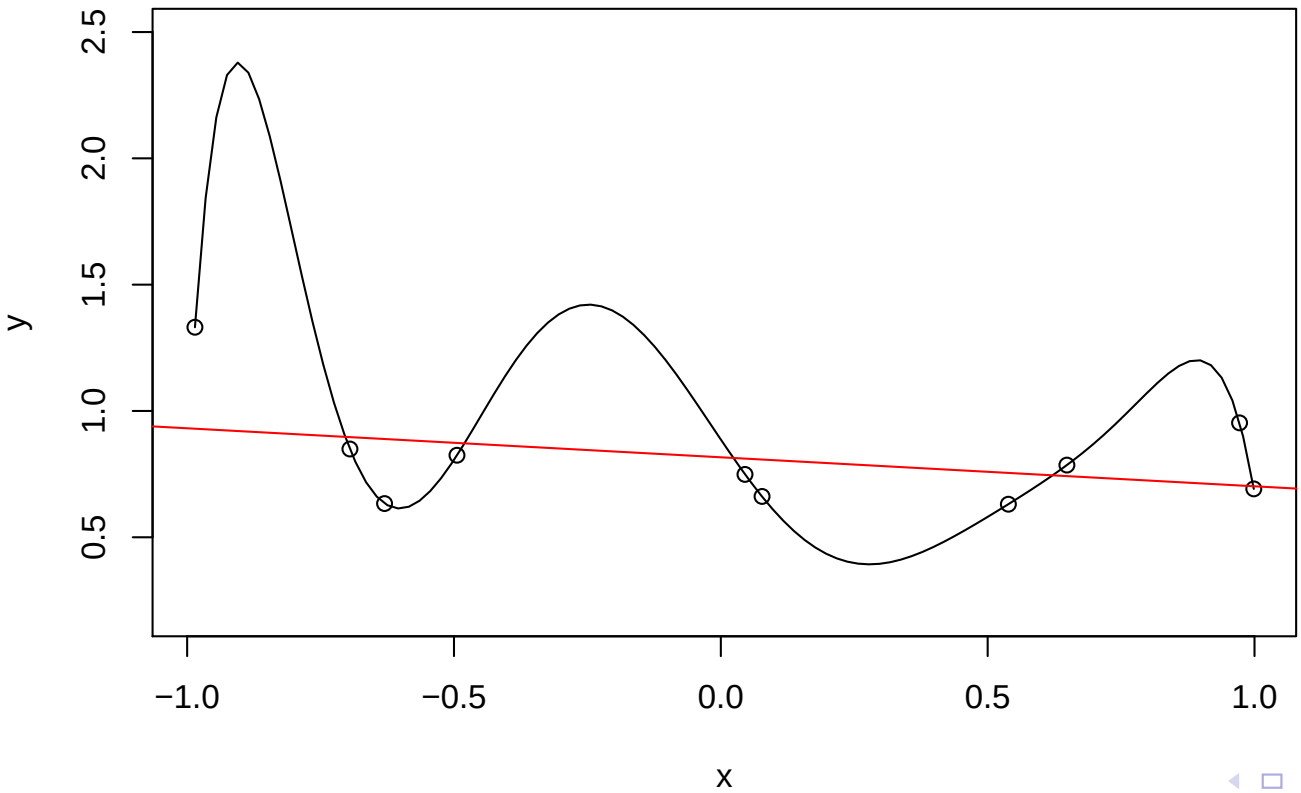
- mp0: Underfit
- mp7: Overfit

```
y_orig <- predict(mp7)
> y_orig
1         2         3         4         5         6         7
1.3314886 0.8493793 0.6337676 0.8246671 0.7495139 0.6610792 0.6310283...
> orig$y
1.3314859 0.8495205 0.6335226 0.8248011 0.7489029 0.6616987 0.6309019...
```

Examples of Overfitting

Simulated

```
mp0 <- lm(y ~ x, data=orig)
mp7 <- lm(y ~ x + I(x^2)+ I(x^3)+ I(x^4)+ I(x^5) +
     I(x^6)+ I(x^7)+ I(x^8),  data=orig)
```



Examples of Overfitting

Simulated

Create new data given the same population model:

```
nd <- data.frame(x=sort(runif(N, -1, 1)))
nd$y <- 0.5 - 0.2*nd$x + 0.5*nd$x^2 + rnorm(N, 0, 0.1)
```

Use fitted values from `mp7` for prediction with new data

```
y_new <- predict(mp7, newdata = nd, type = "response")
> y_new ## new "fitted values"
1         2         3         4         5         6         7
0.8919789 0.8797185 0.8381293 0.8313199 0.8226948 0.7832875 0.7449920...
> y_orig
1         2         3         4         5         6         7
1.3314886 0.8493793 0.6337676 0.8246671 0.7495139 0.6610792 0.6310283...
```

Examples of Overfitting

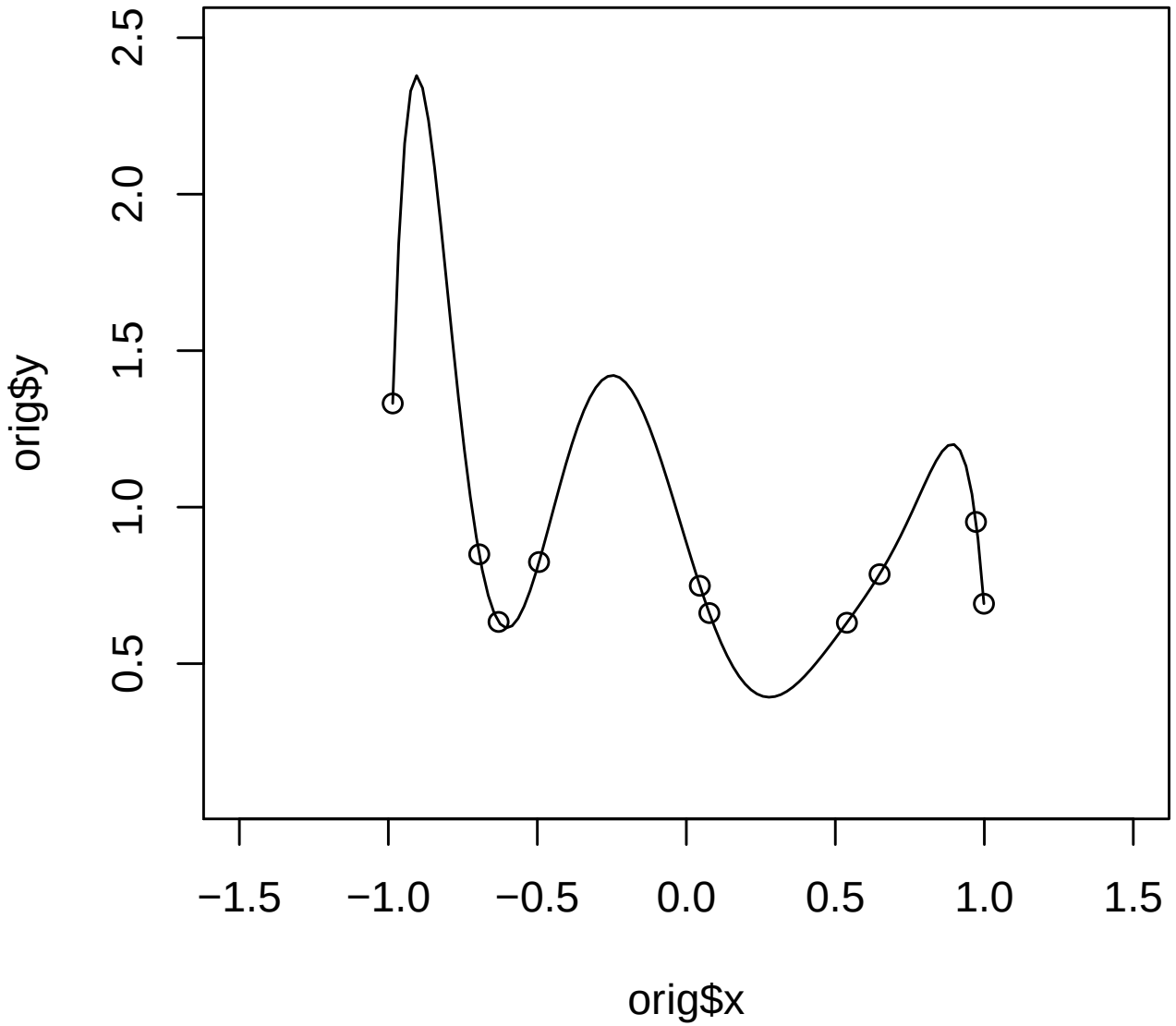
Simulated

- How far away is our prediction from observed on average?
- Root mean squared error (RMSE)

```
> rmse <- function(x1,x2) sqrt(mean((x1-x2)^2))
> mn <- rmse(y_new, nd$y)
> mn
[1] 0.2549604
> mo <- rmse(y_orig, orig$y)
> mo
[1] 0.0002979094
> mn/mo
[1] 855.832
```

Examples of Overfitting

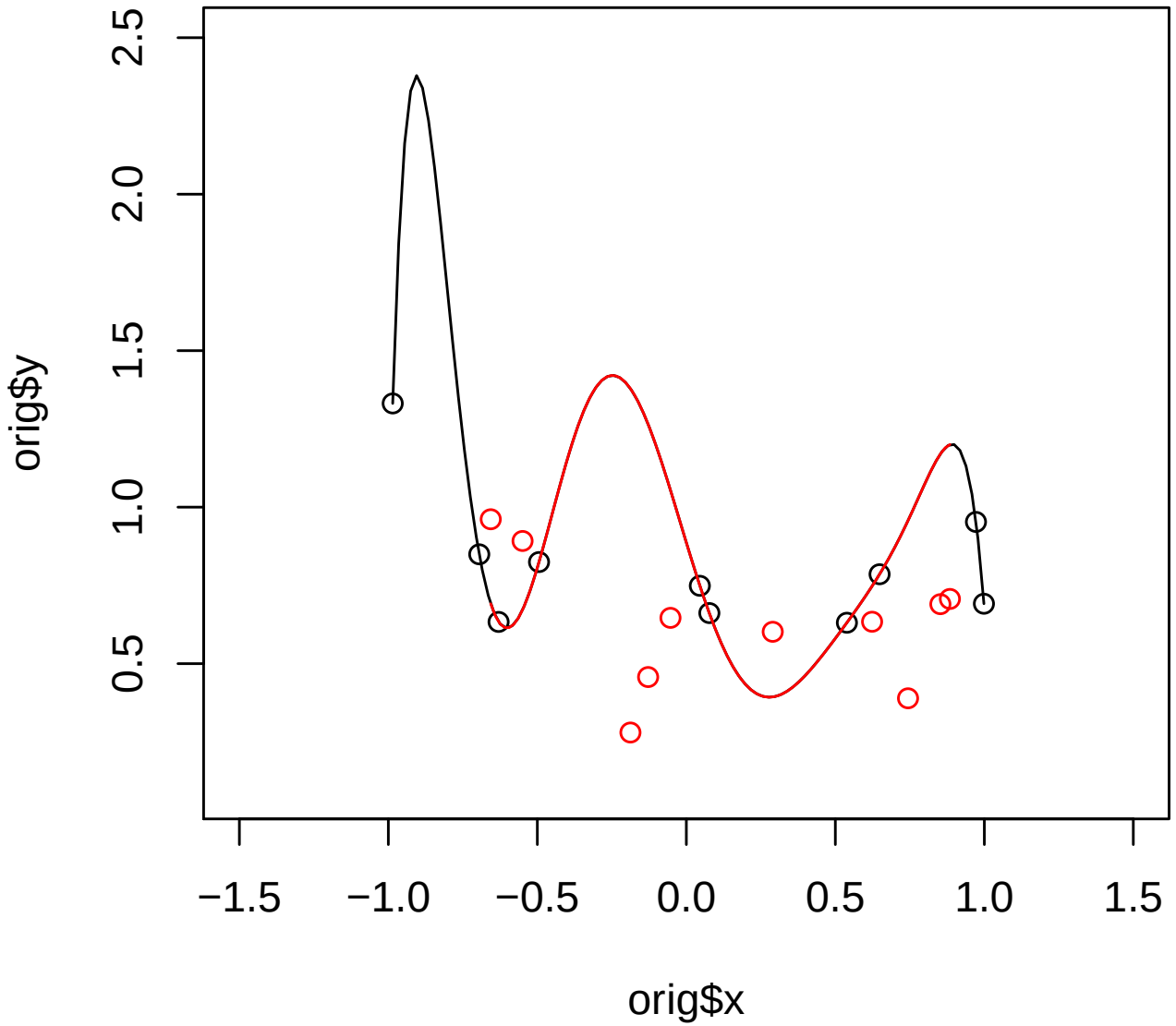
Simulated



- RMSE of residuals:
  - 0.0003

Examples of Overfitting

Simulated



- RMSE of residuals:
  - 0.0003
- RMSE of prediction errors for out-of-sample:
  - 0.2550

Examples of Overfitting

Simulated

Parsimonious (underfitting) model:

```
mp0 <- lm(y ~ x, data=orig)
y_new <- predict(mp0, newdata = nd, type = "response")
```

Compare newly predicted with fitted data from `mp0`

```
> y_new
1         2         3         4         5         6         7
0.9000149 0.8251083 0.7993448 0.7920883 0.7627443 0.7531640 0.7476896...
> y_orig
1         2         3         4         5         6         7
0.9297628 0.8964036 0.8889502 0.8733670 0.8113811 0.8077114 0.7546982...
```

Examples of Overfitting

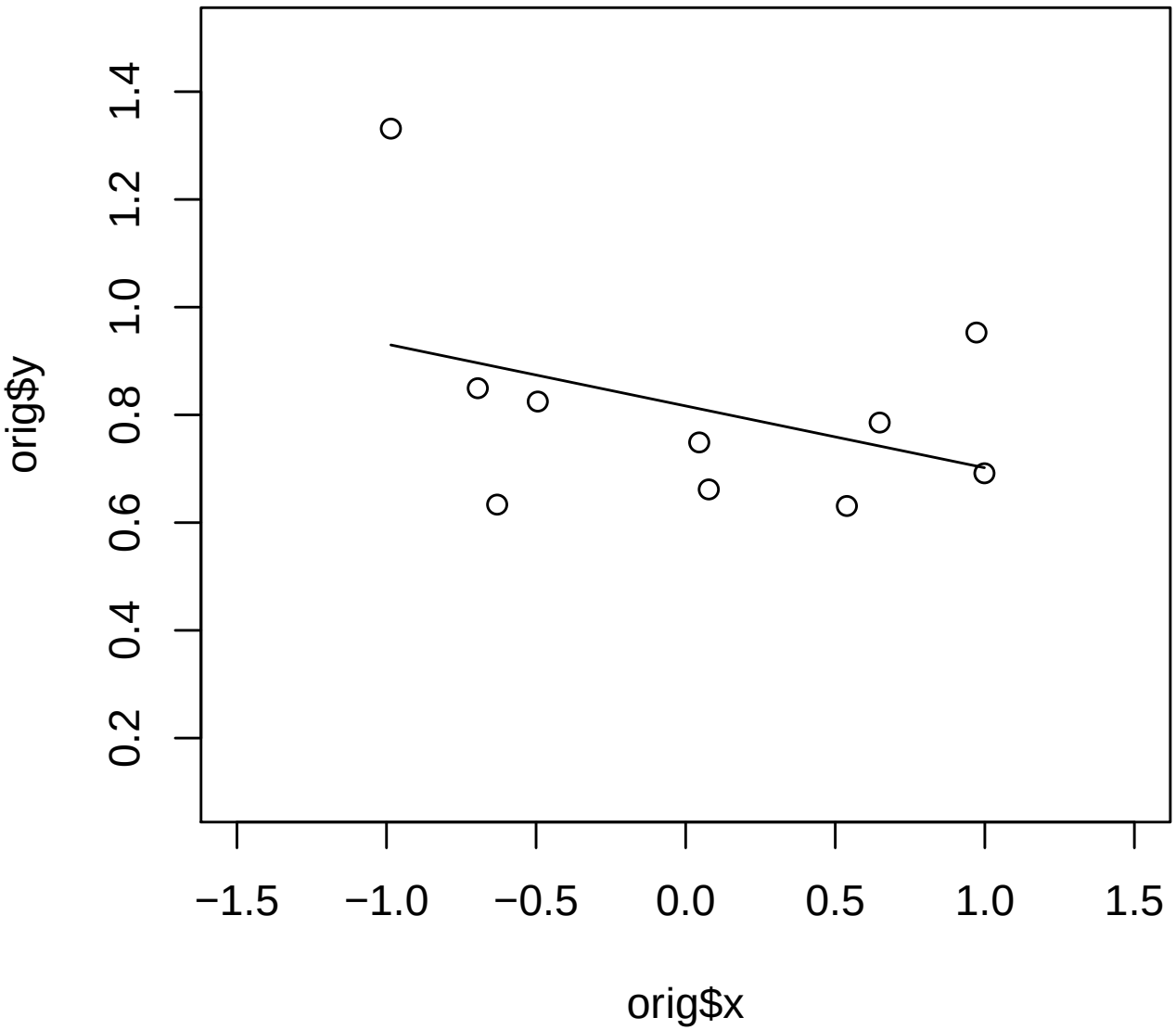
Simulated

- How far away is our prediction from observed on average?
- Root mean squared error (RMSE)

```
> mn <- rmse(y_new, nd$y)
> mn
[1] 0.2429248
> mo <- rmse(y_orig, orig$y)
> mo
[1] 0.1830958
> mn/mo
[1] 1.326763
```

Examples of Overfitting

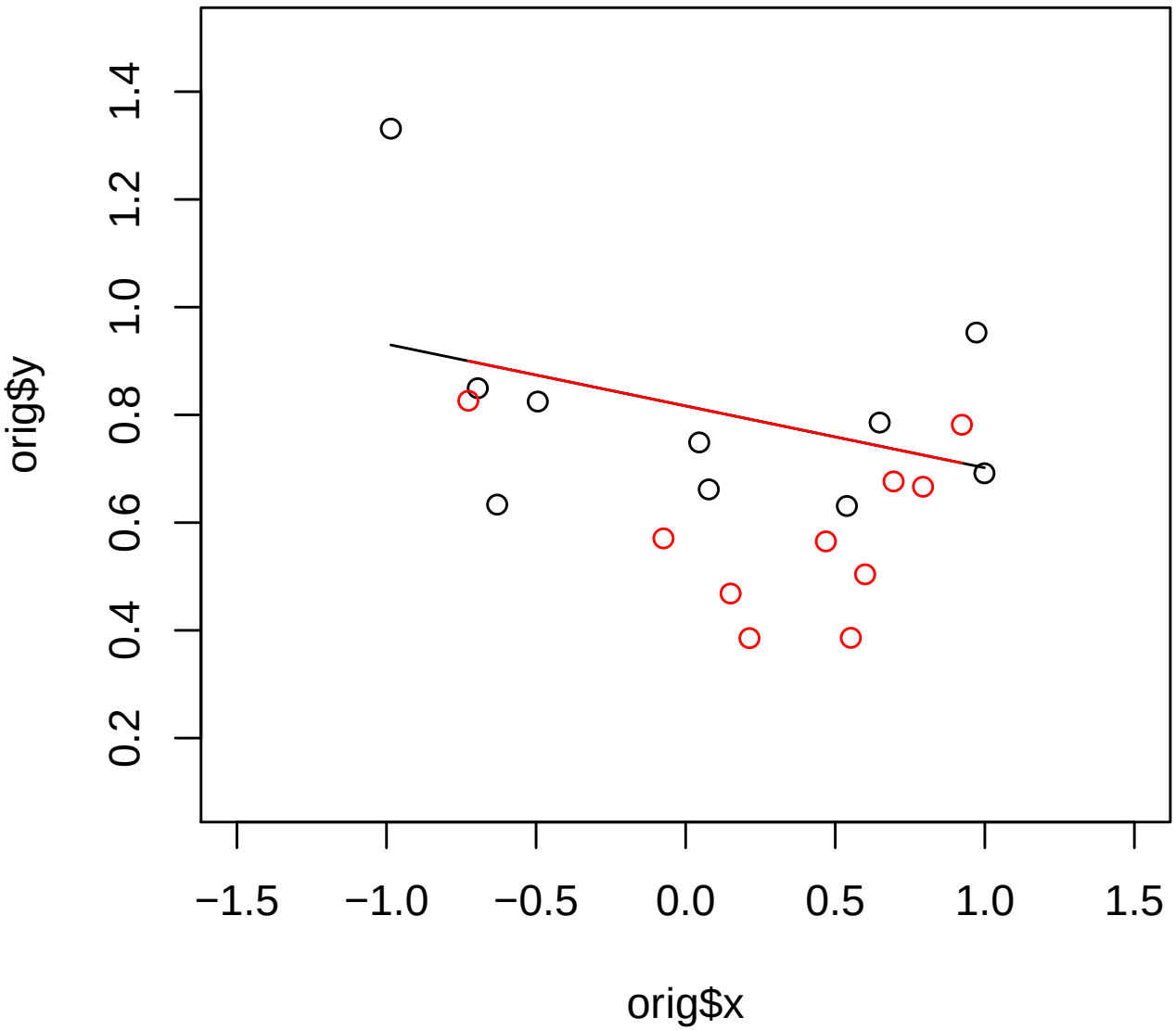
Simulated



- RMSE of residuals:
  - 0.1831

Examples of Overfitting

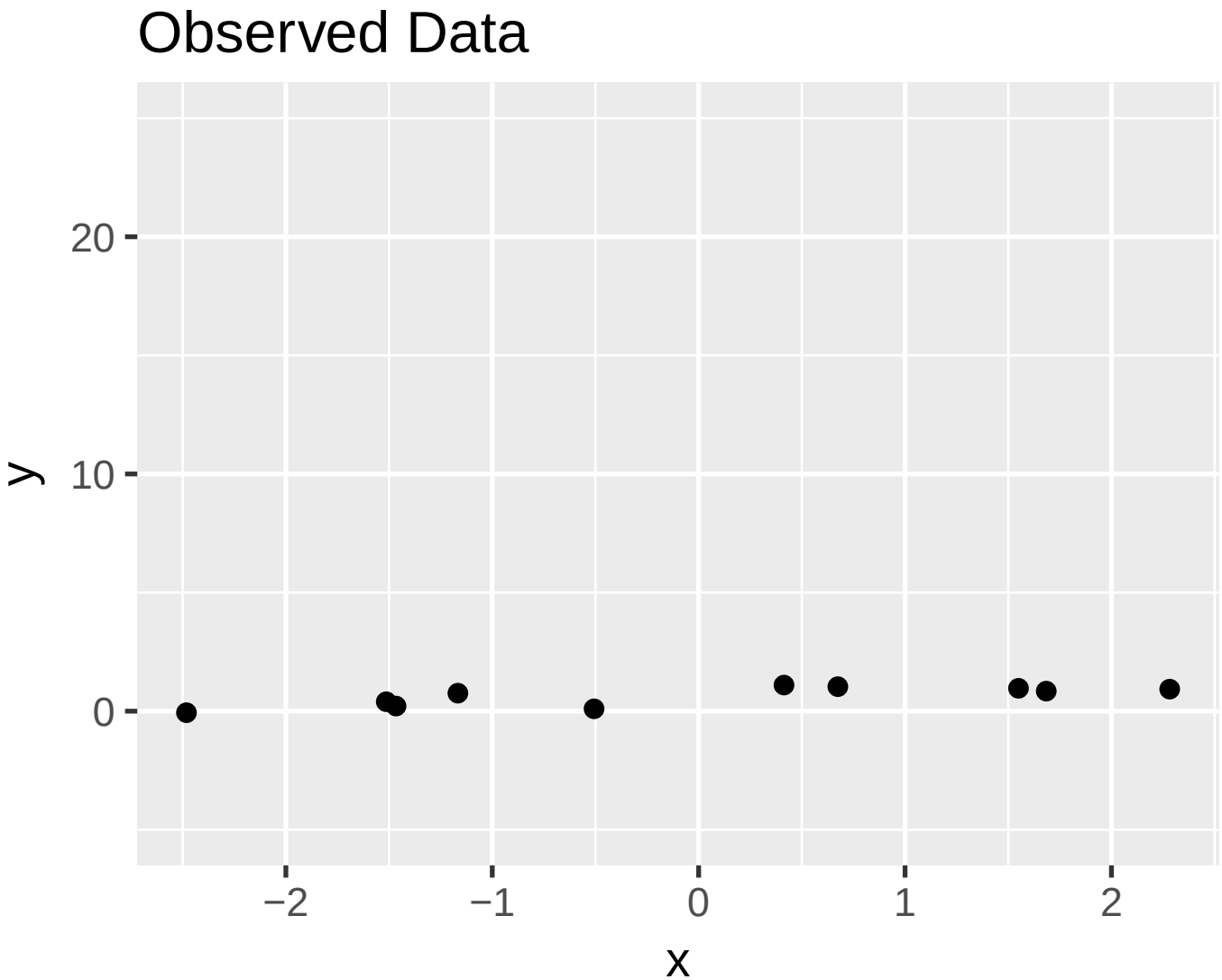
Simulated



- RMSE of residuals:
  - 0.1831
- RMSE of prediction errors for out-of-sample:
  - 0.2430

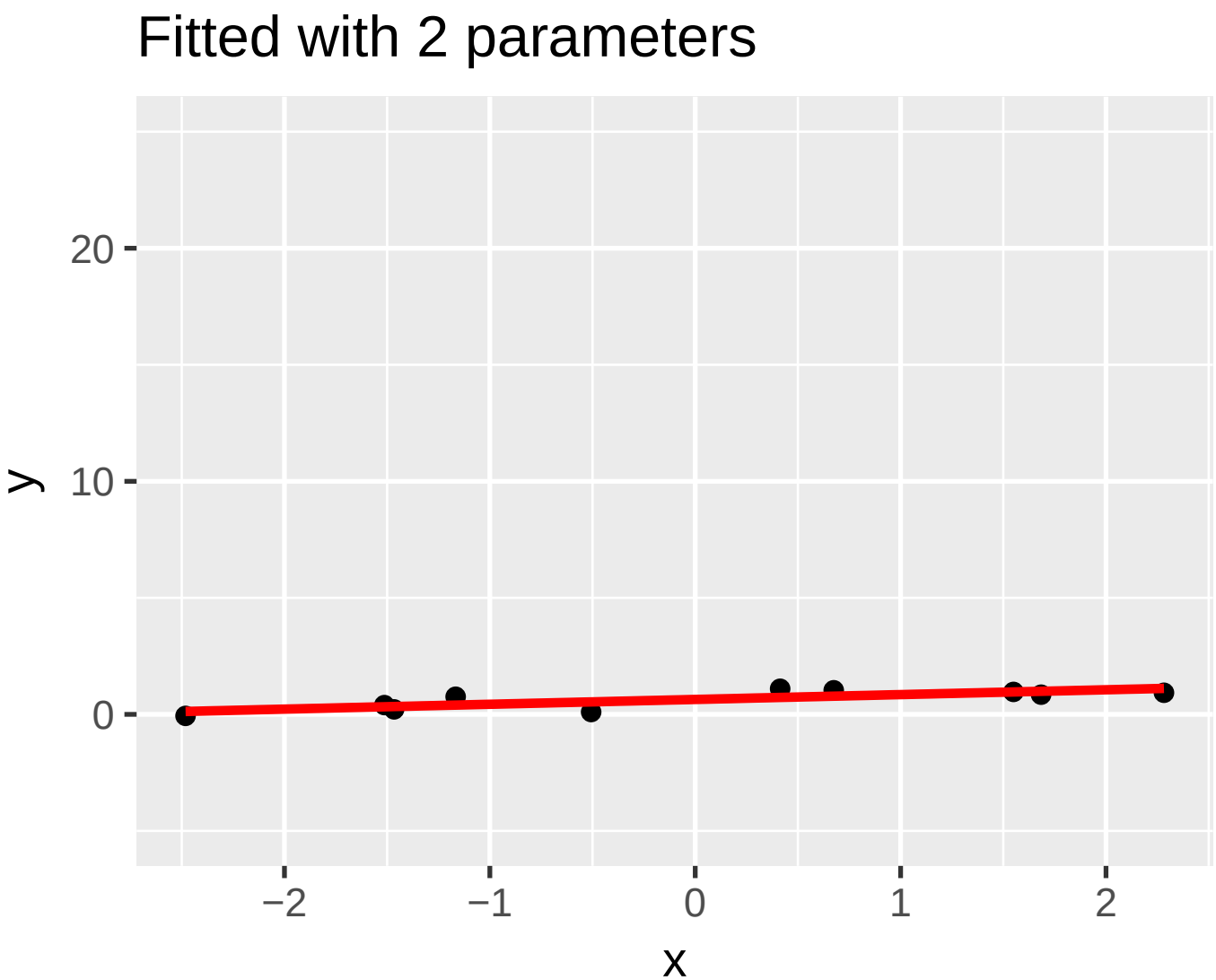
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



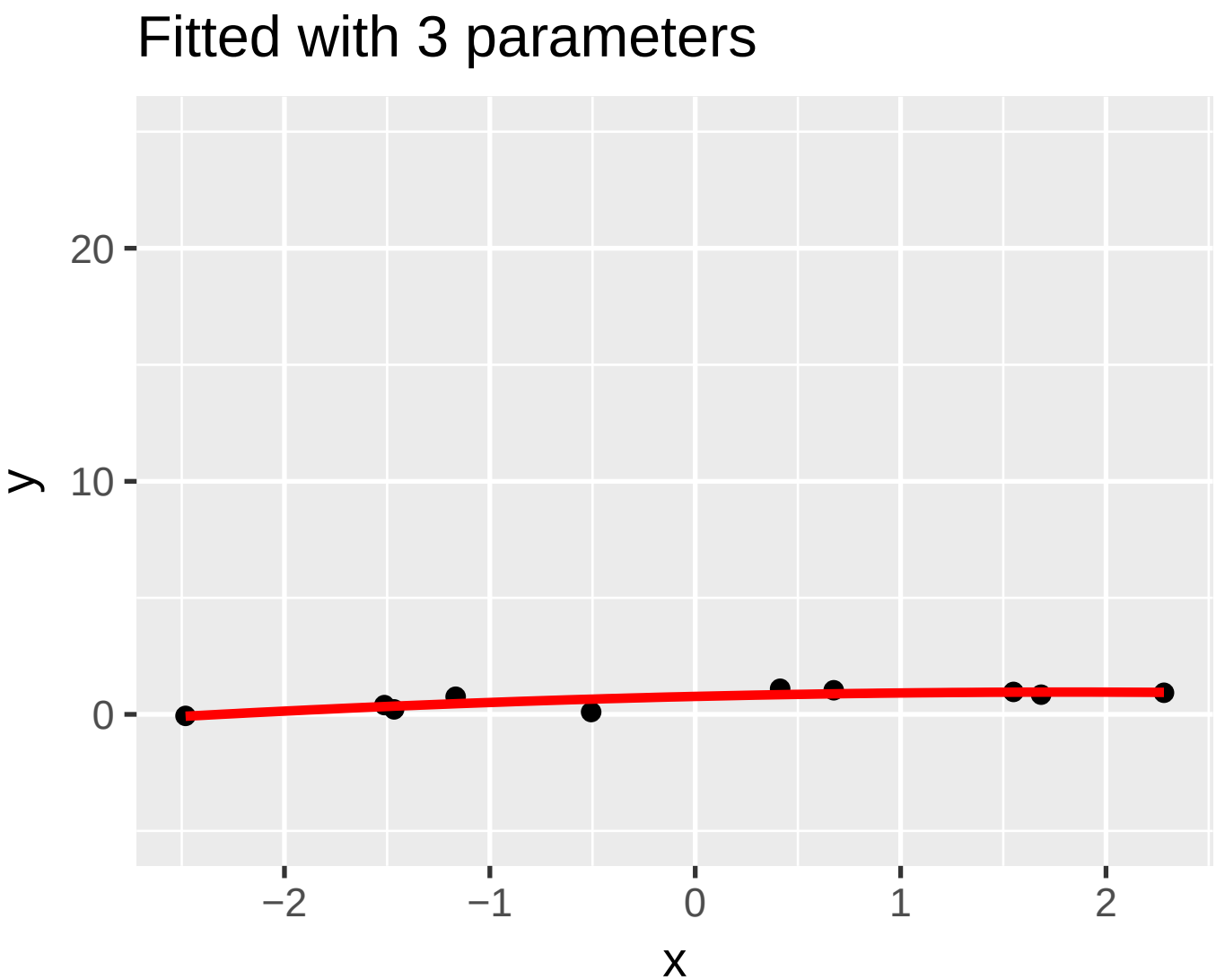
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



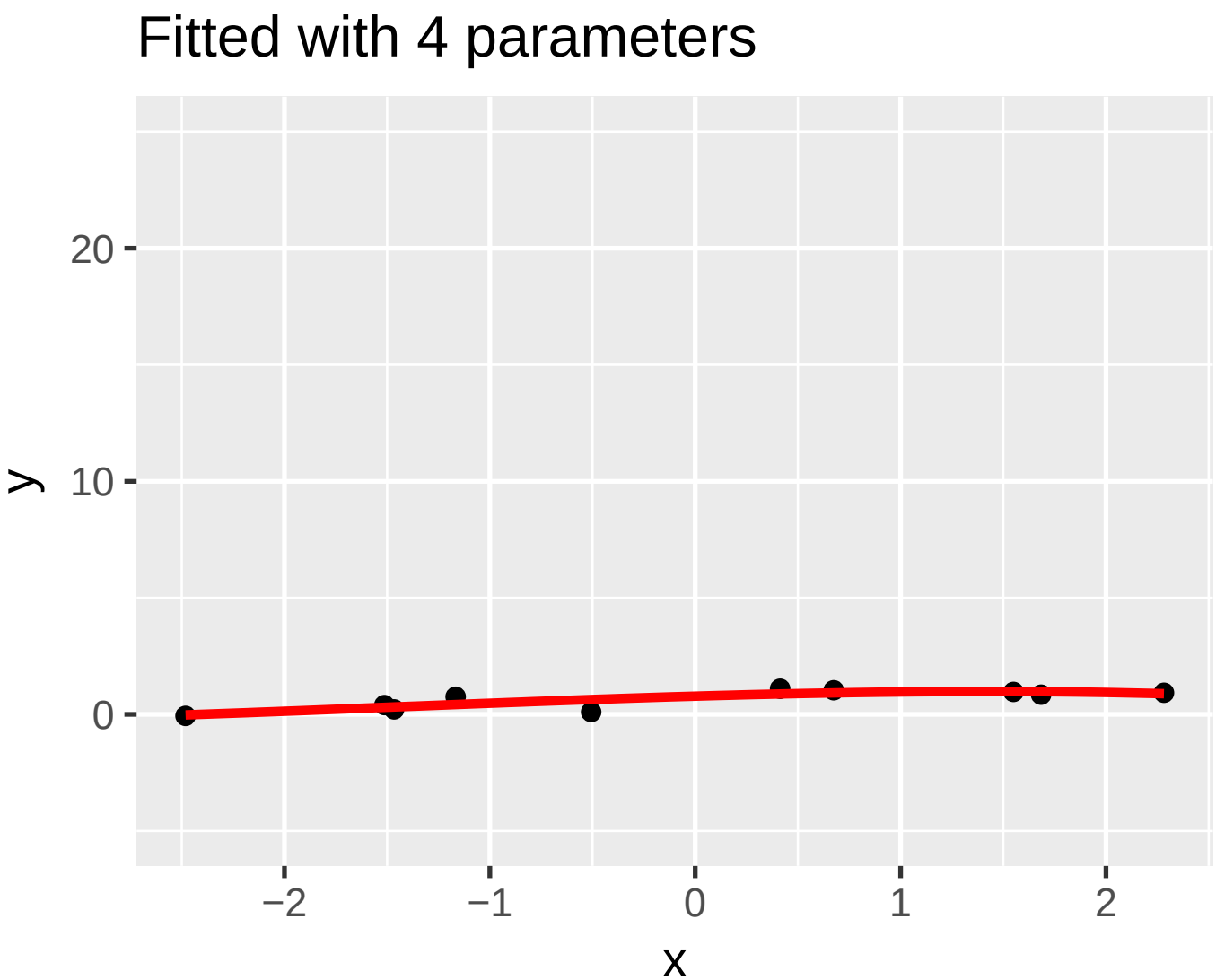
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



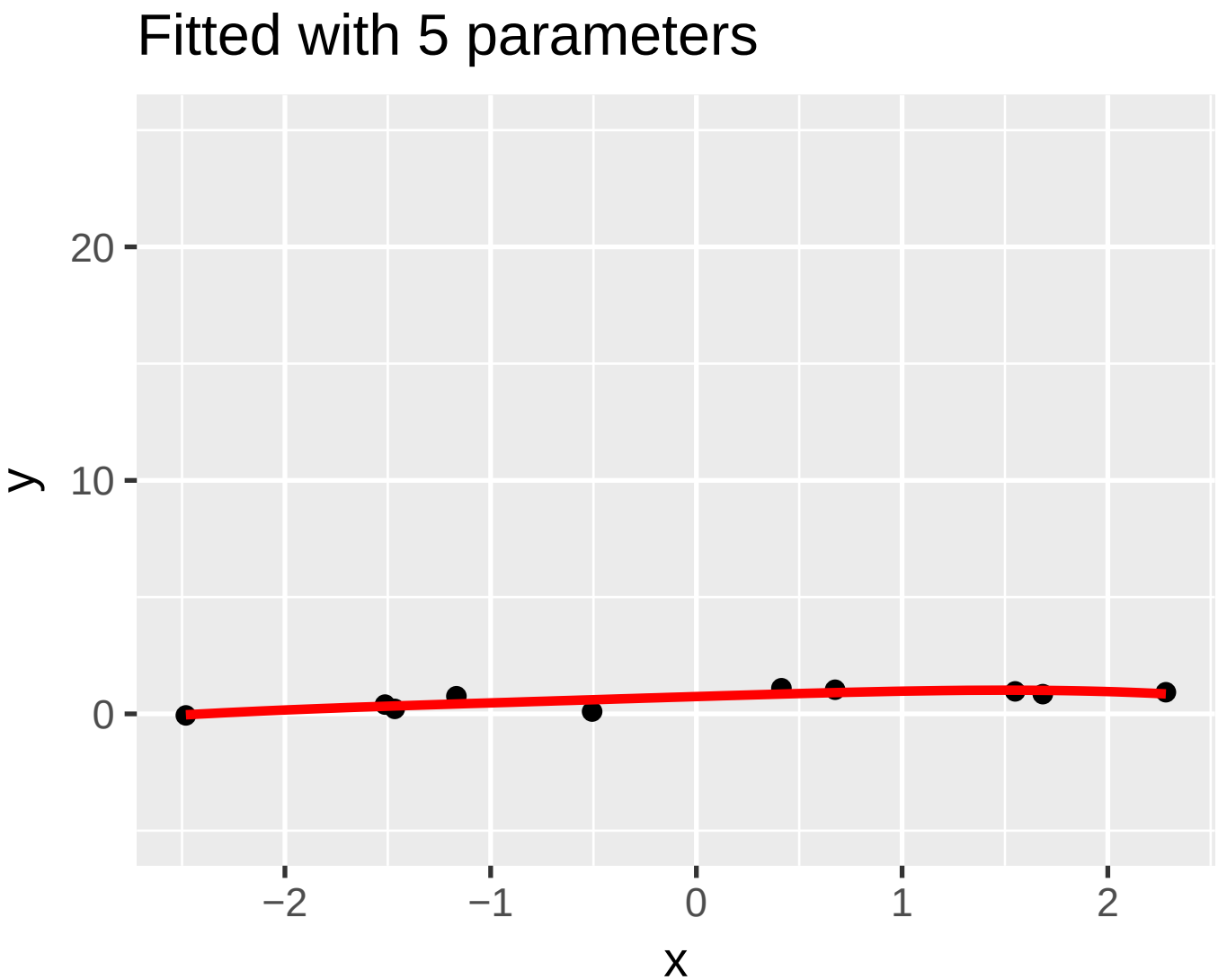
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



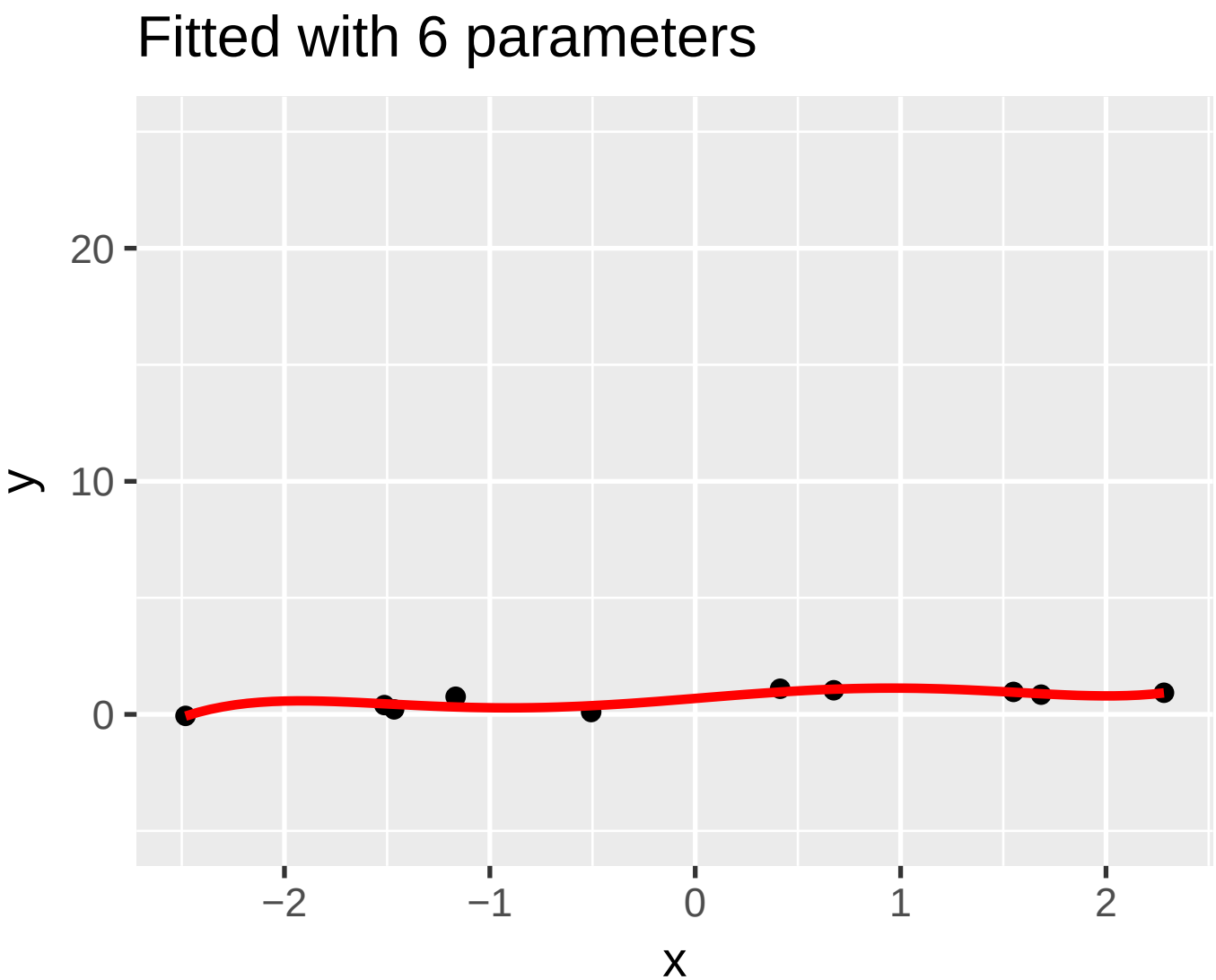
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



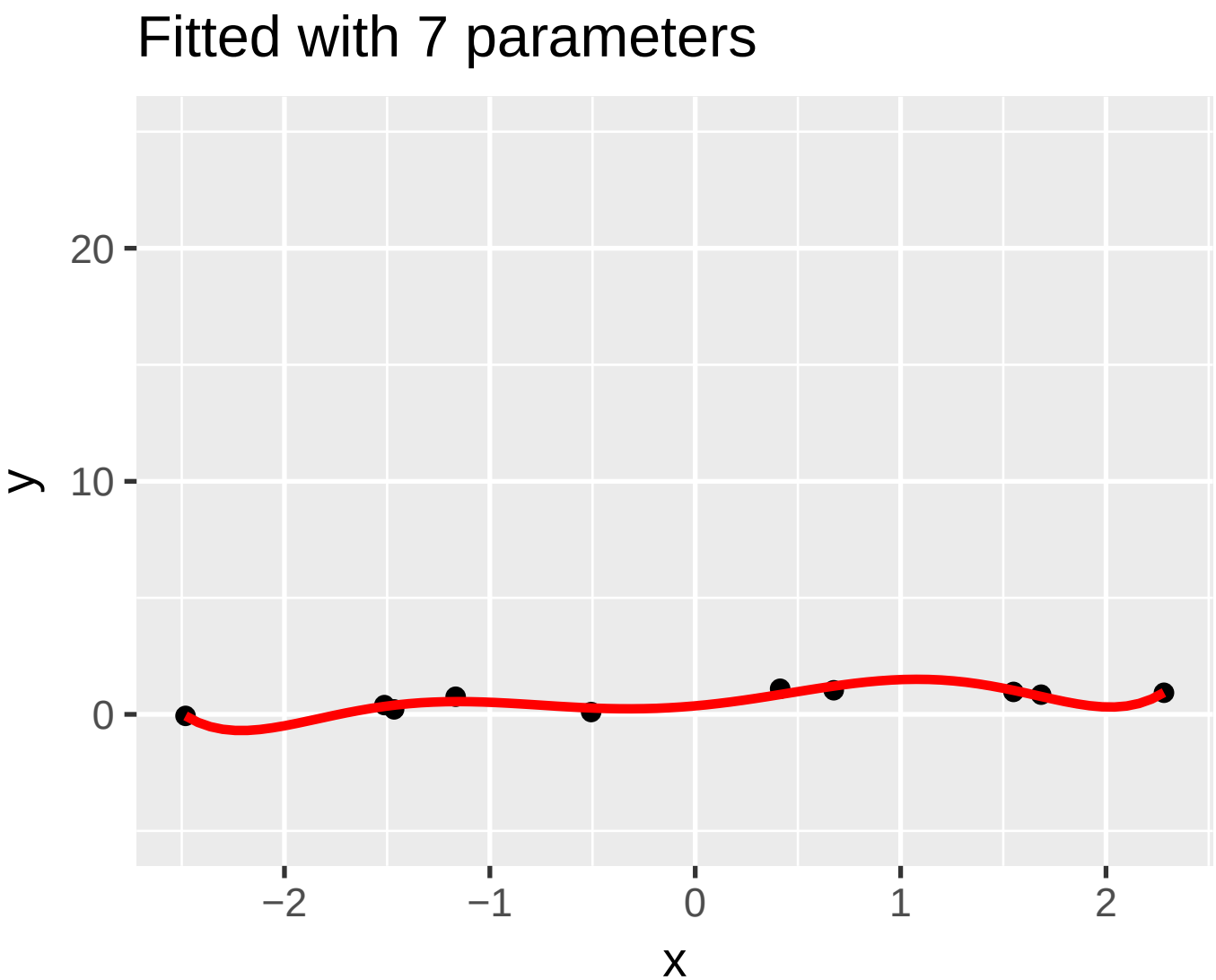
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



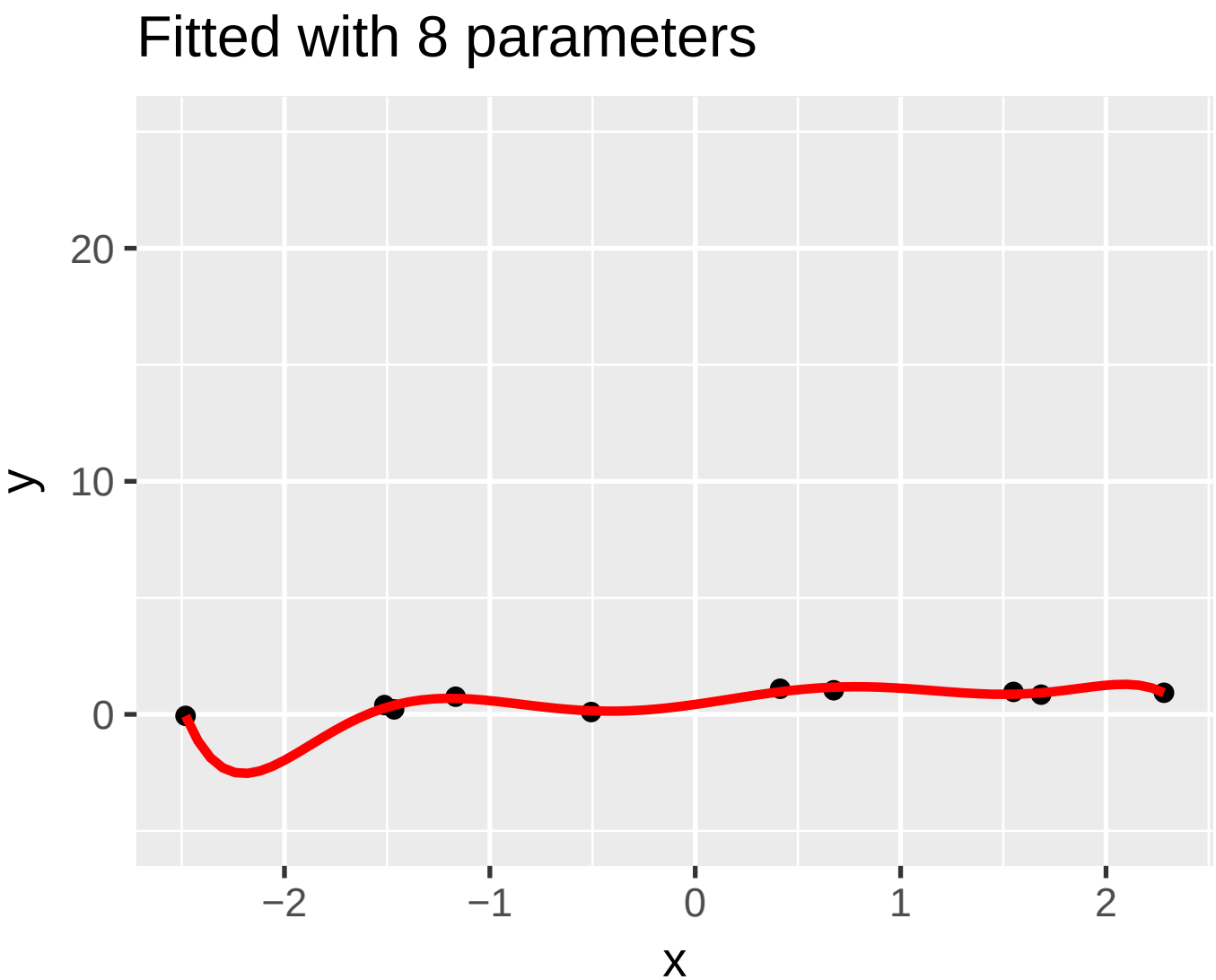
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



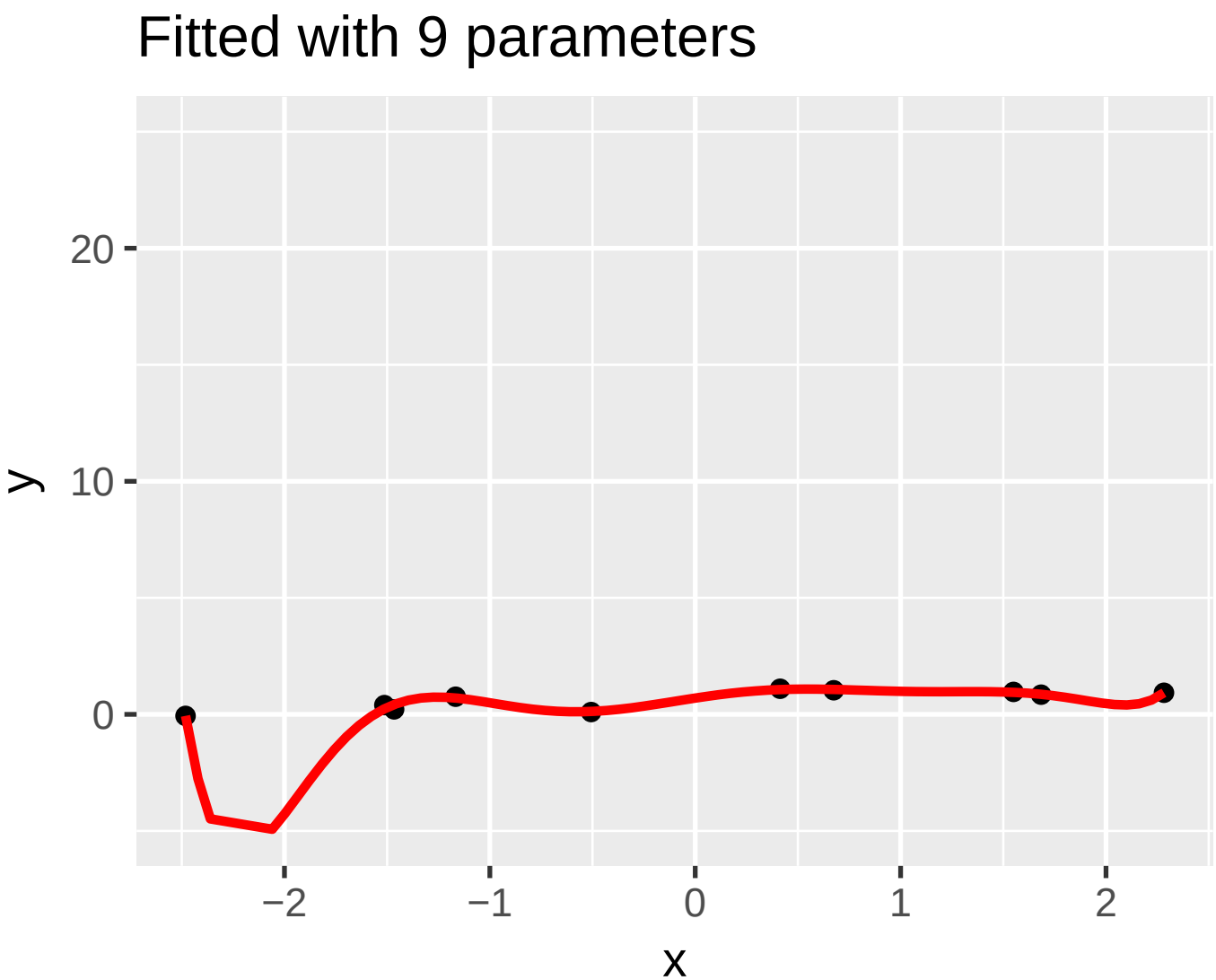
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



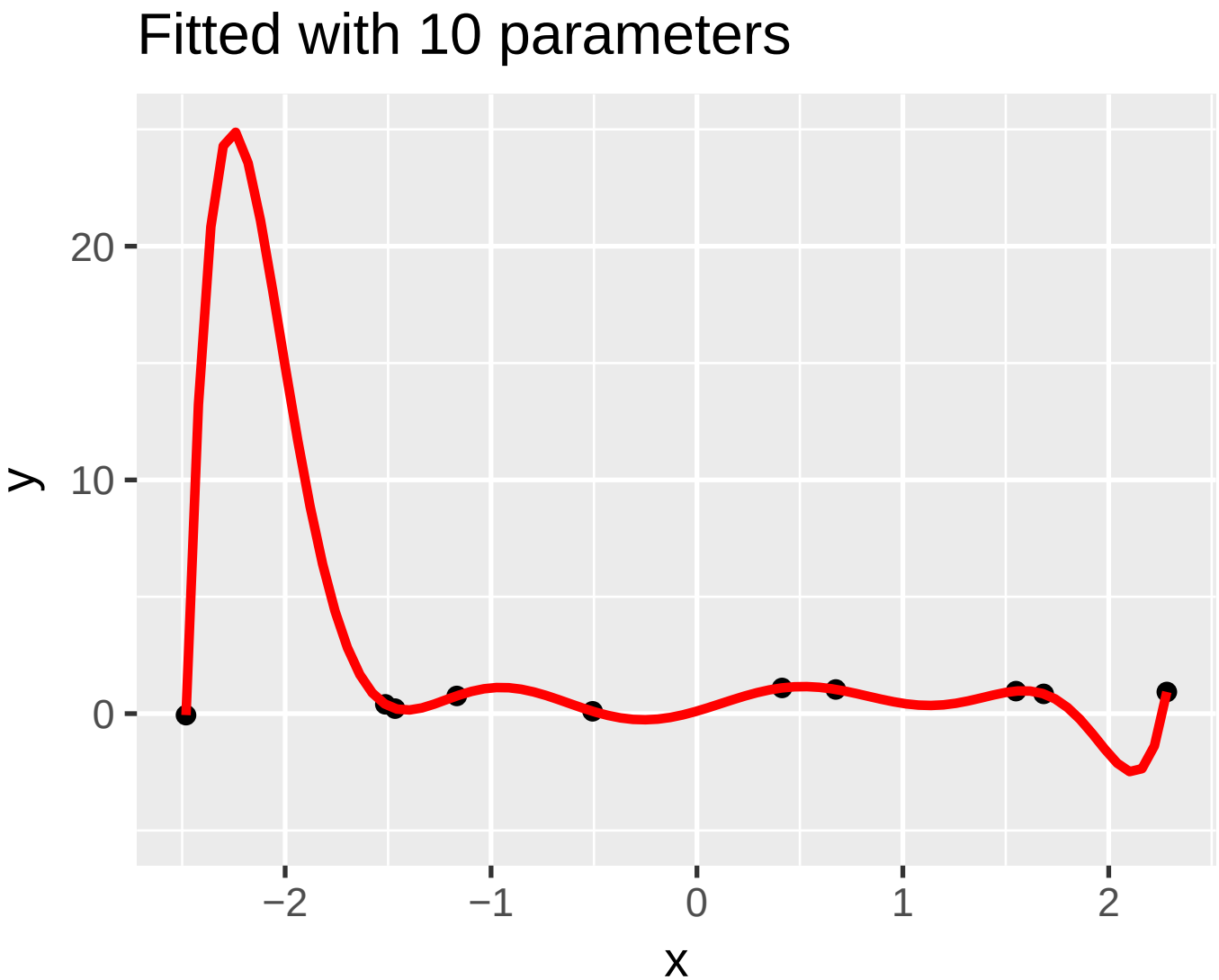
The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?



The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?

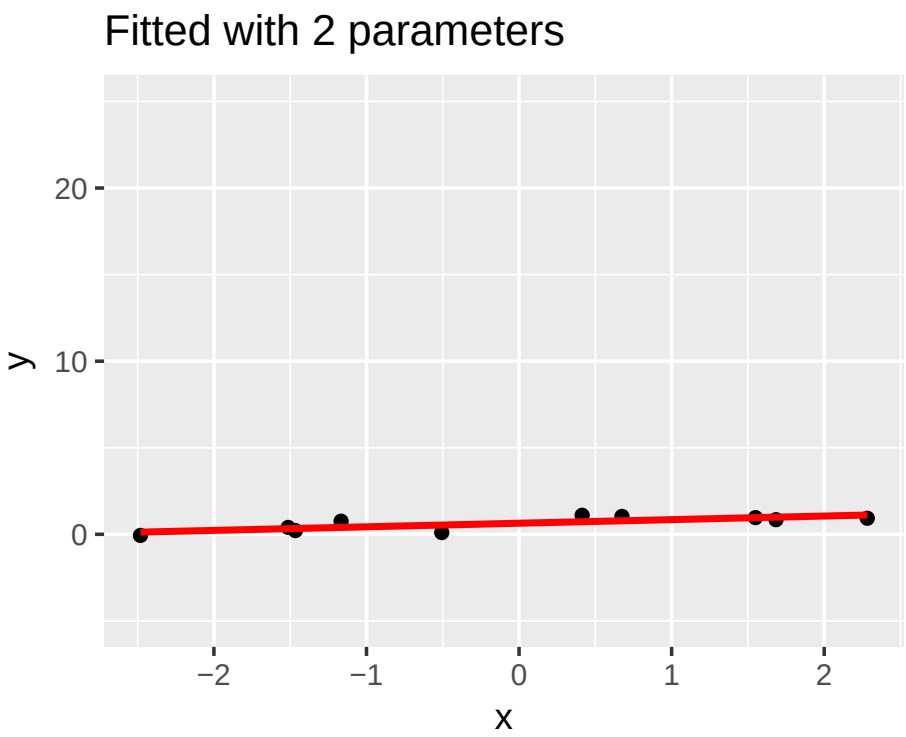


The "right" model

- How do we deal with over- and underfitting?
- Any way to know what the ideal number of parameters are?

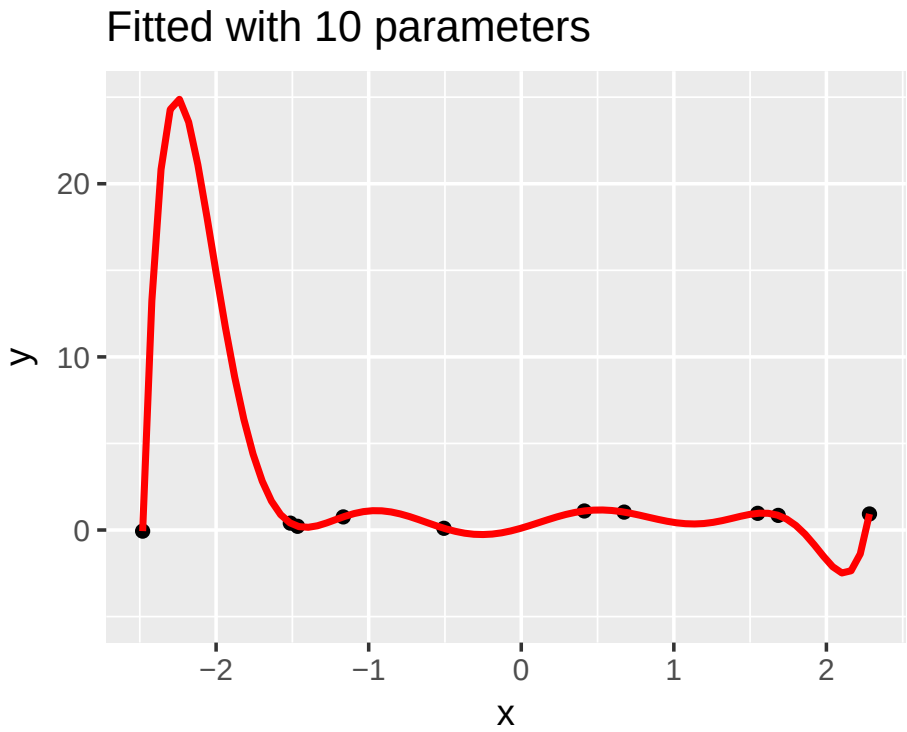
Underfitting

Insensitive to exact data



Overfitting

Very sensitive to exact data



The "right" model

- Looking at p-values, drop whatever is not significant

- *Stargazing*: Using asterisks ($p < 0.05$) to decide which variables improve prediction
- 5% hurdle is arbitrary; doesn't optimize anything

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.886895   0.001218  728.00 0.000874 ***
x            -3.130535   0.017304 -180.92 0.003519 **
I(x^2)        1.282894   0.023005   55.77 0.011415 *
I(x^3)       19.572600   0.116801  167.57 0.003799 **
I(x^4)      -17.391423   0.139300 -124.85 0.005099 **
I(x^5)      -36.914724   0.238120 -155.03 0.004106 **
I(x^6)       43.570763   0.269638  161.59 0.003940 **
I(x^7)       20.402913   0.140698  145.01 0.004390 **
I(x^8)      -27.604677   0.155564 -177.45 0.003588 **
```

The problem with parameters

- Some researchers deal with a huge number of variables
- Eg. take a look here
- We've seen this before
- More complex models always fit better - or equally well
  - very accurate within but very inaccurate out-of-sample

- Too few predictors underfit data
  - inaccurate both within and out-of-sample
- What about stepwise variable selection?

Stepwise Variable Selection

- Commonly employed technique
- Selection of "important" variables
- "it violates every principle of statistical estimation and hypothesis testing." (Harrell, 2015)
  - R^2 values are upward biased
  - Ordinary F and χ^2 test statistics do not have the claimed distribution
  - Standard errors of regression coefficient estimates are biased
  - P-values that are too small
  - Provides regression coefficients that are biased high in absolute value and need shrinkage
  - Rather than solving problems caused by collinearity, variable selection is made arbitrary by collinearity.
  - **It allows us to not think about the problem.**
- ▷ "The choice of the variables to be included depends on estimated regression coefficients rather than their true values, and so X_j is more likely to be included if its regression coefficient is over-estimated than if its regression coefficient is underestimated." (Copas & Long, 1991)

Stepwise Variable Selection

- Derksen and Keselman's (1992), study on backward and forward stepwise selection
  - Variables selected for the final model represented noise 0.20 to 0.74 of the time and the final model usually contained less than half of the actual number of authentic predictors
- No currently available stopping rule has been developed for data-driven variable selection.
- If stepwise variable selection is needed, the step-down or backward method is preferred for the following reasons.
  - Usually performs better than forward stepwise methods, especially when collinearity is present
  - It makes one examine a full model fit, which is the only fit providing accurate standard errors, error mean square, and P-values.
  - Efficient step-down modeling using Wald statistics (method of Lawless and Singhal)

Stepwise Variable Selection

Backward

```
step(lm(y ~ x + I(x^2)+ I(x^3)+ I(x^4)+ I(x^5) + I(x^6) +
      I(x^7)+ I(x^8),  data=orig), direction = 'backward')
Start:  AIC=-144.37
y ~ x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8)

          Df Sum of Sq       RSS      AIC
<none>                 0.0000009 -144.374
- I(x^2)   1  0.002760 0.0027609  -65.948
- I(x^4)   1  0.013834 0.0138345  -49.832
- I(x^7)   1  0.018663 0.0186636  -46.838
- I(x^5)   1  0.021329 0.0213302  -45.502
- I(x^6)   1  0.023174 0.0231746  -44.673
- I(x^3)   1  0.024921 0.0249221  -43.946
- I(x^8)   1  0.027946 0.0279465  -42.800
- x         1  0.029049 0.0290498  -42.413
```

Stepwise Variable Selection

Backward

- Stepwise Variable Selection
- ▷ Failed!
- True model is `I(x^2)`
- AIC missed it - selected full model

- Generally, stepwise selection is not recommended for variable selection

Regularization

- Want the regular features of the sample
- Strategies
  - Cross-validation
  - Penalized likelihood
  - Information criteria
  - Scientific process as iterative learning
- Proper approach depends on purpose

K-Fold Cross Validation

- Widely used approach for estimating error
- Estimates can be used to select best model, and give an idea of the test error of the final chosen model
- Idea is to randomly divide data into K equal-sized parts. We leave out part k, fit the model to other $K-1$ parts (combined), and then obtain prediction for the left out kth part
- This is done in turn for each part $k = 1, 2, ...K$, and then results are combined

K-Fold Cross Validation

Idea

- Divide data in k roughly equal sized parts (K=5)
- Random assignment

- 5 fold cross validation
- Trainings sets are merged: Always Test vs. Validation

k	1	2	3	4	5
1	Train	Train	Train	Train	Validation
2	Train	Train	Train	Validation	Train
...					
5	Validation	Train	Train	Train	Train

K-Fold Cross Validation

Details

- Let the K parts be $C_1, C_2, ...C_K$, where C_k denotes the indices of the observations in par k. There are n_k observations in part k: if N is a multiple of K, then $n_k = n/K$.
- Compute CV error rate

$$\text{CV}_K = \sum_{k=1}^{K} \frac{n_k}{n} MSE_k$$

where $MSE_k = \sum_{i \in C_k} (y_i - \hat{y})^2 / n_k$ (error we obtained from the validation part), and $\hat{y}_i$ is the fit for observation i, obtained from the data with part k removed

- Special case: *Leave-one-out cross validation* (LOOCV) with $K = n$

K-Fold Cross Validation

LOOCV

- With OLS an shortcut makes the ocst of LOOCV the same as that of a single model fit
- $$\text{CV}_n = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2$$

  where $\hat{y}_i$ is teh ith fitted value from the original OLS fit and h_i is the leverage (diagonal of the "hat" matrix). This is like the ordinary MSE, except the ith residual is divided by $1 - h_i$.
- LOOCV can be useful approximation to classic CV.
- Data from each fold are highly correlated

Regularization

Ridge Regression

- Ridge Regression is a regularization method that tries to avoid overfitting
- Penalize large coefficients through the "L2 Norm"
- ▷ L2 Regularization
- L2 quadratic ("ridge") penalty (Hoerl and Kennard, 1970)

- Original OLS estimator
- OLS: $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$

- Penalized Ridge estimator
- Ridge: $\hat{\boldsymbol{\beta}}_{\text{Ridge}} = (\boldsymbol{X}'\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}'\boldsymbol{y}$
- Alternatively: $RSS(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2$
- λ is tuning parameter that needs to be estimated
- if $\lambda = 0$ then $\hat{\boldsymbol{\beta}}_{\text{Ridge}} = \hat{\boldsymbol{\beta}}_{\text{OLS}}$

Regularization

Ridge Regression

- Ridge regression focuses on the $\boldsymbol{X}'\boldsymbol{X}$ predictor covariance matrix
- $\det(\boldsymbol{X}'\boldsymbol{X})$ will not be 0 as it adds $\lambda \boldsymbol{I}$ and ensures that whole term is invertible
- Modifying the matrix in this way effectively eliminates collinearity
  - Leads to more precise, and therefore more interpretable, parameter estimates.
- Recall: Multicollinearity inflates standard errors
- But: Trade-off between variance and bias.
- There is a cost to this decrease in variance: increase in bias.
- However, bias introduced by ridge regression is almost always toward the null.
  - Ridge regression is considered a "shrinkage method" as it typically shrinks the beta coefficients toward 0.

Regularization

Ridge Regression

- Ridge Regression in R

```
library(glmnet)
x <- cbind(1, orig$x, orig$x^2,  orig$x^3,  orig$x^4,
orig$x^5,  orig$x^6,  orig$x^7, orig$x^8)
y <- orig$y

## alpha = 0 for ridge
lambdas <- 10^seq(3, -2, by = -.1)
```

Regularization

Ridge Regression

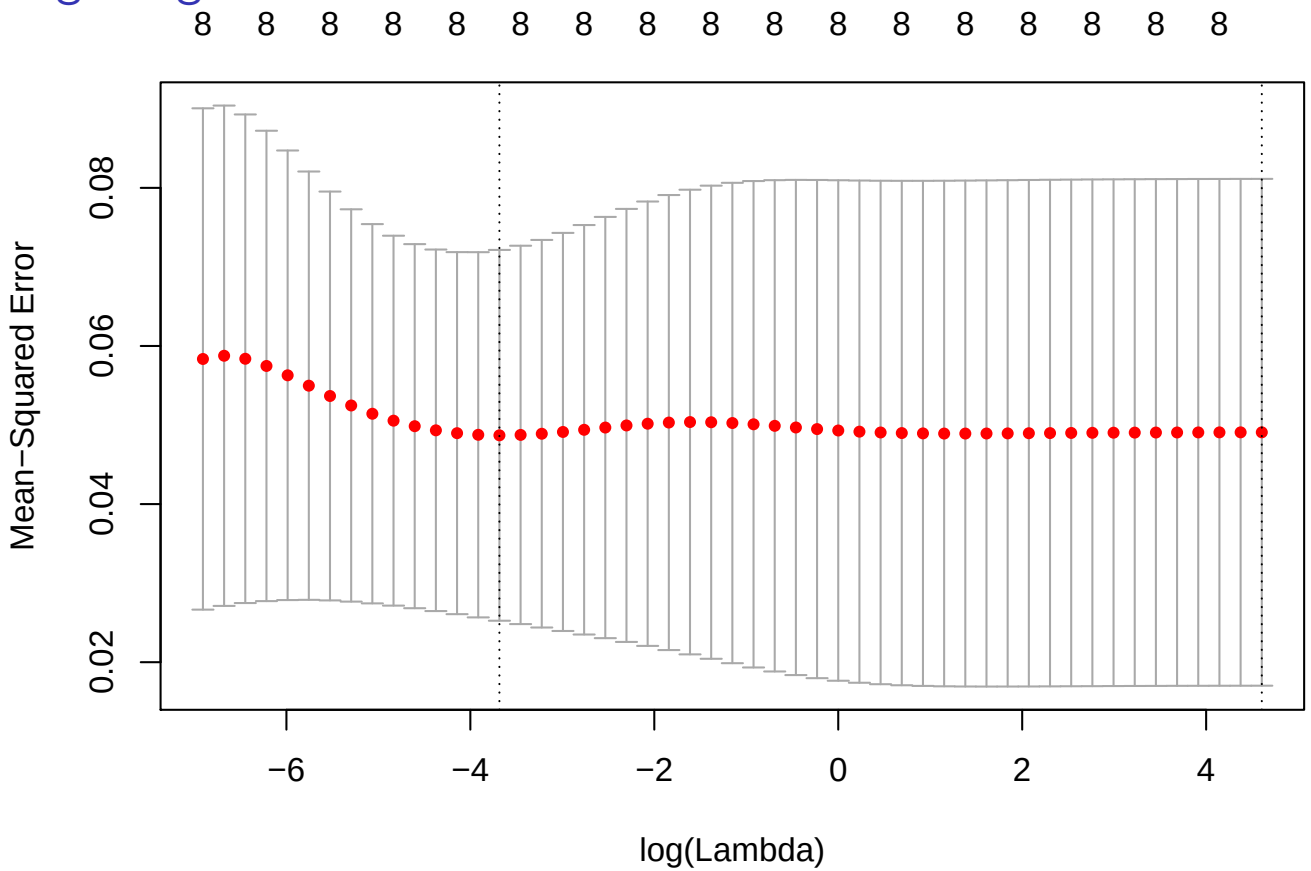
- `glmnet()` runs the model many times for different values of lambda.
- We can automatically find a value for lambda that is optimal by using `cv.glmnet()` as follows:

```
cv.fit <- cv.glmnet(x, y, family='gaussian', alpha=0,
lambda = lambdas)
```

- `cv.glmnet()` uses cross-validation to work out how well each model generalizes
- Cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set).
- Results can be visualized

Regularization

Ridge Regression




```
> opt_lambda <- cv.fit$lambda.min
> opt_lambda
[1] 0.02511886
```

Regularization

Ridge Regression

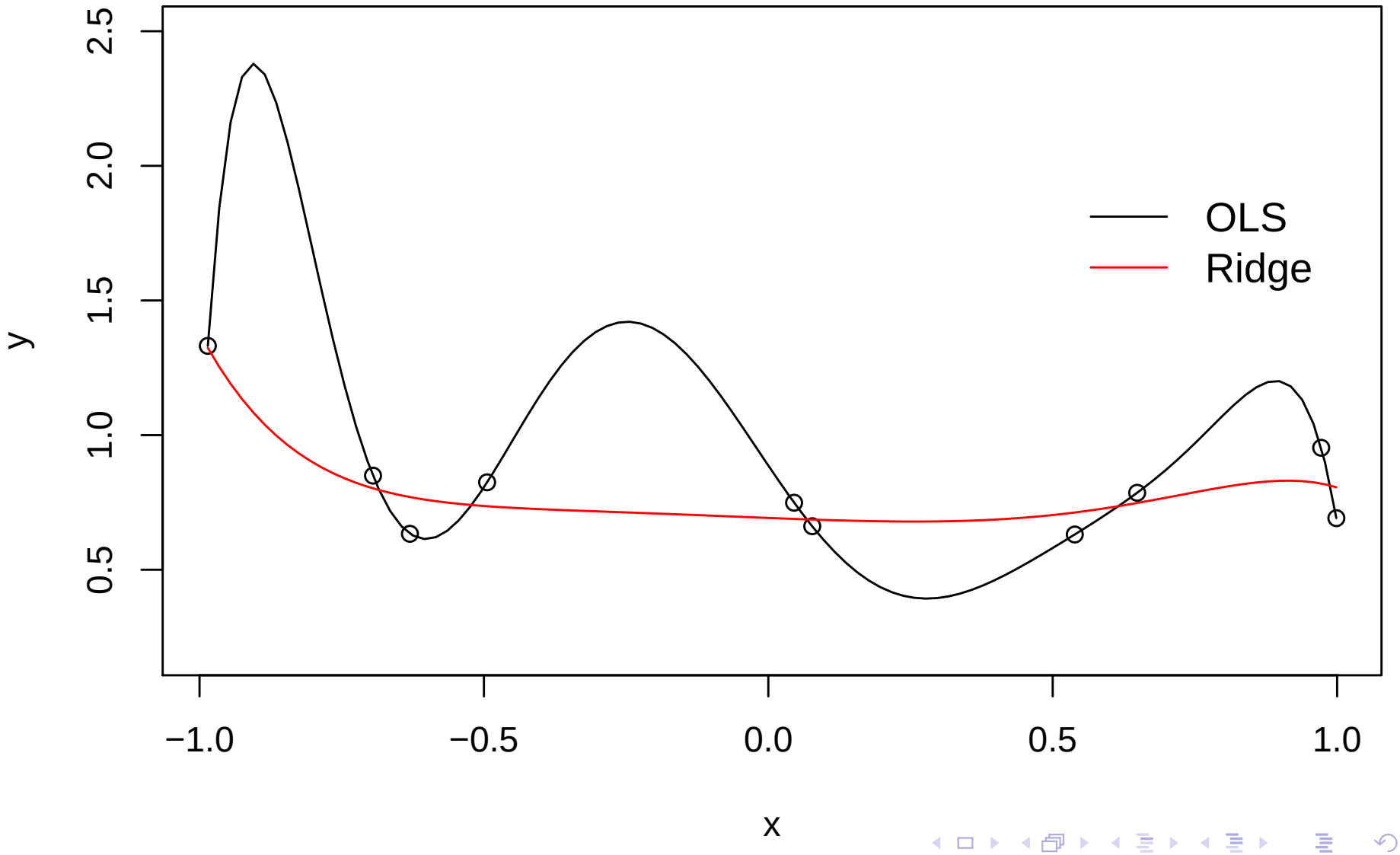
```
> beta_ridge <-
solve(t(x)%*%x + opt_lambda*diag(9)) %*% t(x)%*%y
```

```
> beta_ridge
[,1]
[1,]  0.69235717
[2,] -0.08228291
[3,]  0.03995950
[4,]  0.23359934
[5,]  0.24406832
[6,] -0.06075343
[7,]  0.15559211
[8,] -0.37801645
[9,] -0.03906176
```

```
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.886895   0.00121  728.00 0.0008 ***
x            -3.130535   0.01730 -180.92 0.0035 **
I(x^2)        1.282894   0.02300   55.77 0.0114 *
I(x^3)       19.572600   0.11680  167.57 0.0037 **
I(x^4)      -17.391423   0.13930 -124.85 0.0050 **
I(x^5)      -36.914724   0.23812 -155.03 0.0041 **
I(x^6)       43.570763   0.26963  161.59 0.0039 **
I(x^7)       20.402913   0.14069  145.01 0.0043 **
I(x^8)      -27.604677   0.15556 -177.45 0.0035 **
```

Regularization

Ridge Regression



Regularization

Lasso

- Penalize large coefficients through the "L1 Norm"
- ▷ L1 Regularization
- L1 absolute value ("lasso") penalty Tibshirani (1996, 1997)
- This method does two things:
  - Regularization
  - It can shrinkage some of the coefficients to *exactly* zero
  - The regularization serves to select parameters

- Penalized Lasso estimator
- Lasso: $RSS(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$
- λ is tuning parameter that needs to be estimated

Regularization

Lasso

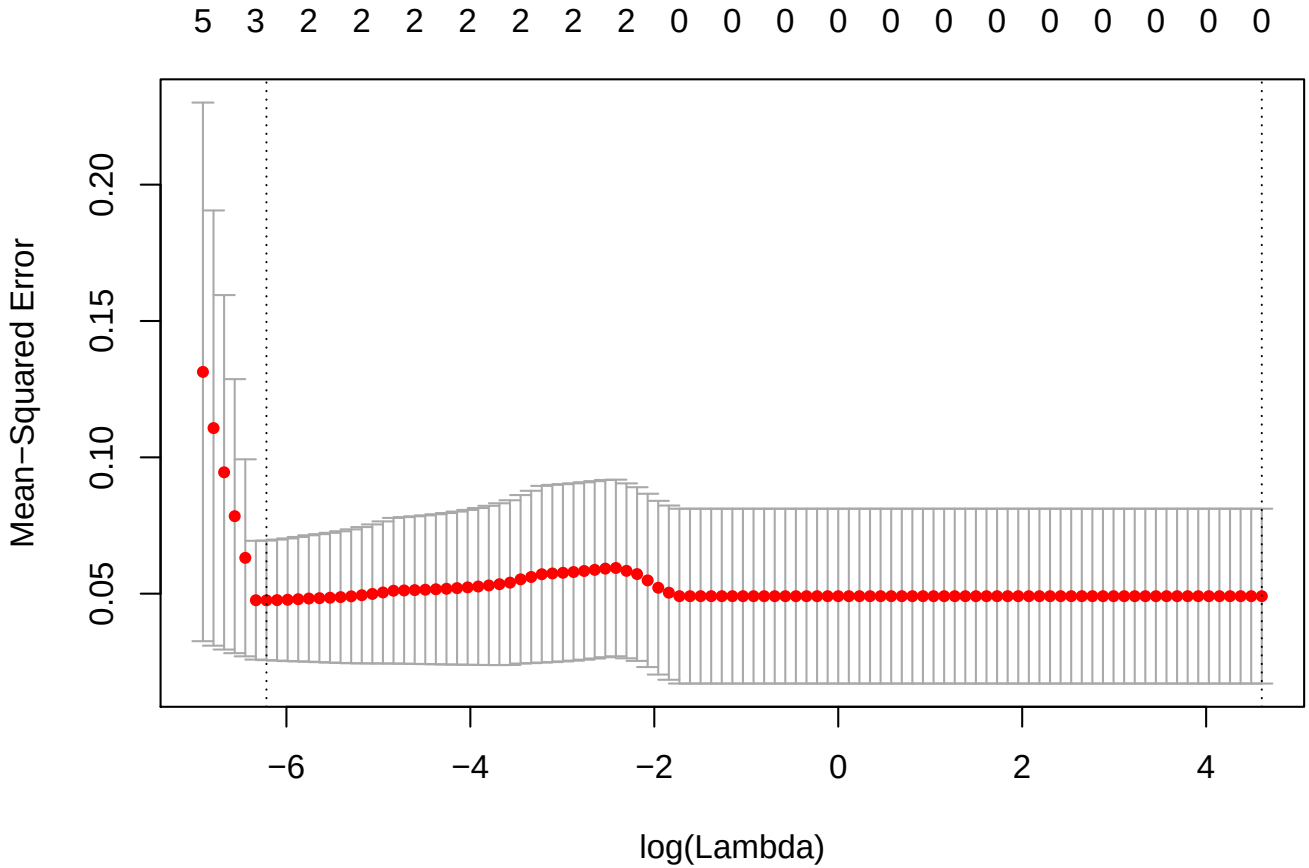
- Lasso Regression in R

```
library(glmnet)
x <- cbind(1, orig$x, orig$x^2,  orig$x^3,  orig$x^4,
orig$x^5,  orig$x^6,  orig$x^7, orig$x^8)
y <- orig$y

## alpha = 1 for Lasso
lambdas <- 10^seq(3, -2, by = -.1)
```

Regularization

Lasso




```
> opt_lambda <- cv.fit$lambda.min
> opt_lambda
[1] 0.001995262
```

Regularization

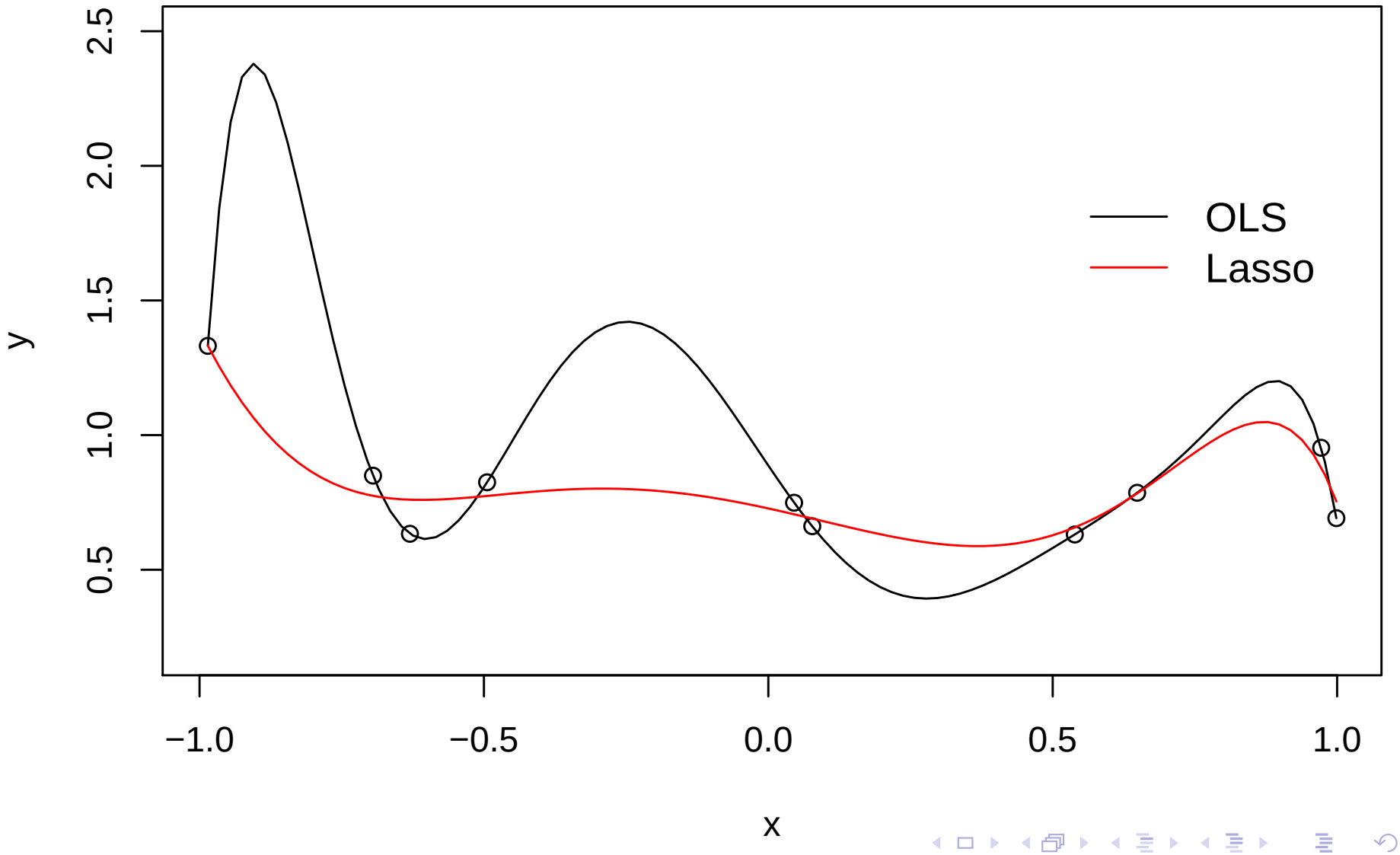
Lasso

```
> beta_lasso
[,1]
[1,]  0.7275693
[2,] -0.4608515
[3,] -0.4397841
[4,]  1.3162448
[5,]  1.2562256
[6,]  0.1128933
[7,]  0.5280974
[8,] -1.2913937
[9,] -1.0031578
```

```
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.886895   0.00121  728.00 0.0008 ***
x            -3.130535   0.01730 -180.92 0.0035 **
I(x^2)        1.282894   0.02300   55.77 0.0114 *
I(x^3)       19.572600   0.11680  167.57 0.0037 **
I(x^4)      -17.391423   0.13930 -124.85 0.0050 **
I(x^5)      -36.914724   0.23812 -155.03 0.0041 **
I(x^6)       43.570763   0.26963  161.59 0.0039 **
I(x^7)       20.402913   0.14069  145.01 0.0043 **
I(x^8)      -27.604677   0.15556 -177.45 0.0035 **
```

Regularization

Lasso



Regularization

- Massive reduction in size of β parameters
- Shrinkage
- In-sample performance is worse
- Much better prediction out-of-sample
- Accept bias but get better prediction in return
- Focus shift from in-sample to out-of-sample
- Popular in data mining

- ▷ Note: Shrinkage methods are generally not invariant to the relative scaling of the covariates
  - Check whether covariates have a natural scaling relative to each other
  - or standardize

Information Theory and Model Performance

- What's a good prediction?

i.e. What do you want the model to do well at?

- This criterion will be called *target*
  - ▷ Information theory provides the target: *out-of-sample deviance*

- Goal is out-of-sample deviance
- Steps toward that goal:
  1. Judge model accuracy by *joint probability*, not average probability
  2. Establish a measurement scale for *distance from target*
  3. *Deviance*, as approximation of relative distance from perfect accuracy

Goal Establish that only *deviance out-of-sample* is of interest

Information Criteria

- Accuracy depends on the definition of the target
- Two dimension when defining a target
  - *Cost-benefit analysis*: How much does it "cost" when we're wrong? How much do we "gain" when we're right?
  - *Accuracy in context*: How much can a model improve in a given context?

- Illustrative example: Pep and no-pep compared to last year

Pep and no-pep

Average hit rate

- Yes $= 1$; No $= 0$
- Probability of answering "yes"
- Model 1:

	1	2	3	4	5	6	7	8	9	10
Predicted	1	1	1	1	1	1	1	1	1	1
Observed	0	0	0	1	1	1	1	1	1	1

- Average chance of correct prediction (hit rate): $(3\times0+7\times1)/10 = 7/10 = 0.7$
- Model 2:

	1	2	3	4	5	6	7	8	9	10
Predicted	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
Observed	0	0	0	1	1	1	1	1	1	1

- Average hit rate: $3\times0.3+7\times0.7/10 = 5.8/10 = 0.58$

Pep and no-pep

Cost and benefit

- On average, it's better to predict "yes" - all the time
  - $.70 > .58$

- Alternative: *Cost and benefit*
- Missing "no" might translate in a bad outcome: -5 points
- Missing "yes" isn't too bad: -1 point

	1	2	3	4	5	6	7	8	9	10
Observed	0	0	0	1	1	1	1	1	1	1
Model 1	-5	-5	-5	0	0	0	0	0	0	0
Model 2	-3.5	-3.5	-3.5	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3	-0.3

- Model 1: -15
- Model 2: -12.6

Measuring Accuracy

Pep and no-pep

How to measure accuracy?

- Cost and benefit
- Average hit rate
  - Average probability does not take into account the individual probabilities

i.e. Probability of getting observation 1 AND ($\cap$) observation 2 AND observation 3... right

- Probability of getting all individual probabilities right
  - Joint probability:

    Model 1: $0^3 \times 1^7 = 0$

    Model 2: $0.3^3 \times 0.7^7 \approx 0.002$

Measuring Accuracy

Pep and no-pep

What is happening here?

- While average probability for Model 1 is higher than for Model 2 joint probability for Model 1 is 0
  - Model 1 *never* expects "no"
  - Model 1 *only* expects "yes"
  - ▷ This is clearly wrong
- Joint probability is a measure that sums up the relative number of ways our observations could happen (**likelihood**).
- ▷ There are many ways that we could end up with a 7 "yes" and 3 "no" observation pool.
- Joint probability can identify the right model – with highest likelihood

How far from truth?

- Truth: The real joint probability of events
- ▷ "The right probabilities given our state of ignorance"
  - Our state of ignorance is described by the model.

  i.e. The probability is in the model, not in the world
  - If we had all of the information relevant for predicting response, "yes" and "no" would be deterministic and the "true" probabilities would be just 0's and 1's

  btw. This leads to a non-identified model in logistic regression

"Absent some relevant information, as in all modeling, outcomes in the small world are uncertain, even though they remain perfectly deterministic in the large world. Because of our ignorance, we can have "true" probabilities between zero and one." (McElrath)

How far from truth?

- Need a way to measure distance of a model from truth
- Distance needs to accommodate:
  - Complexity of prediction task
  - ▷ Some tasks are easier to predict than others (eg. binomial vs. multinomial)
  - By adding more possible outcomes "hit rate" declines
  - Model prediction can be further apart

  i.e. field of possibilities has become larger, which also means that
  - distance among good and bad models increases!

Information theory

- Information Theory: Quantification, storage, and communication of information
  - Signal processing
- Proposed by Claude E. Shannon in 1948

- *Information:* Reduction in uncertainty caused by learning an outcome.
  - Prediction and comparison with actual event.
  - If prediction is far off, from event, we learn a lot
  - If prediction is on target, we learn little from new event
  - The measured decrease in uncertainty is the definition of information in this context

Information Entropy

Information: **Reduction in uncertainty** caused by learning an outcome.

- How to quantify uncertainty? Needs to be:
  - Continuous: If it were not continuous, little changes in probability may result in massive change of uncertainty
  - Increasing with number of possible events: More possibilities are more difficult to predict
  - Additive
- Information Entropy
- ▷ If there are n different possible events and each event i has probability p_i, and we call the list of probabilities p, then the unique measure of uncertainty we seek is:

  $H(p) = -\mathsf{E}\log(p_i) = -\sum_{i=1}^{n} p_i \log(p_i)$
- aka Shannon entropy H
- ▷ *The uncertainty contained in a probability distribution is the average log-probability of an event.*

Information Entropy

Example using pep no-pep

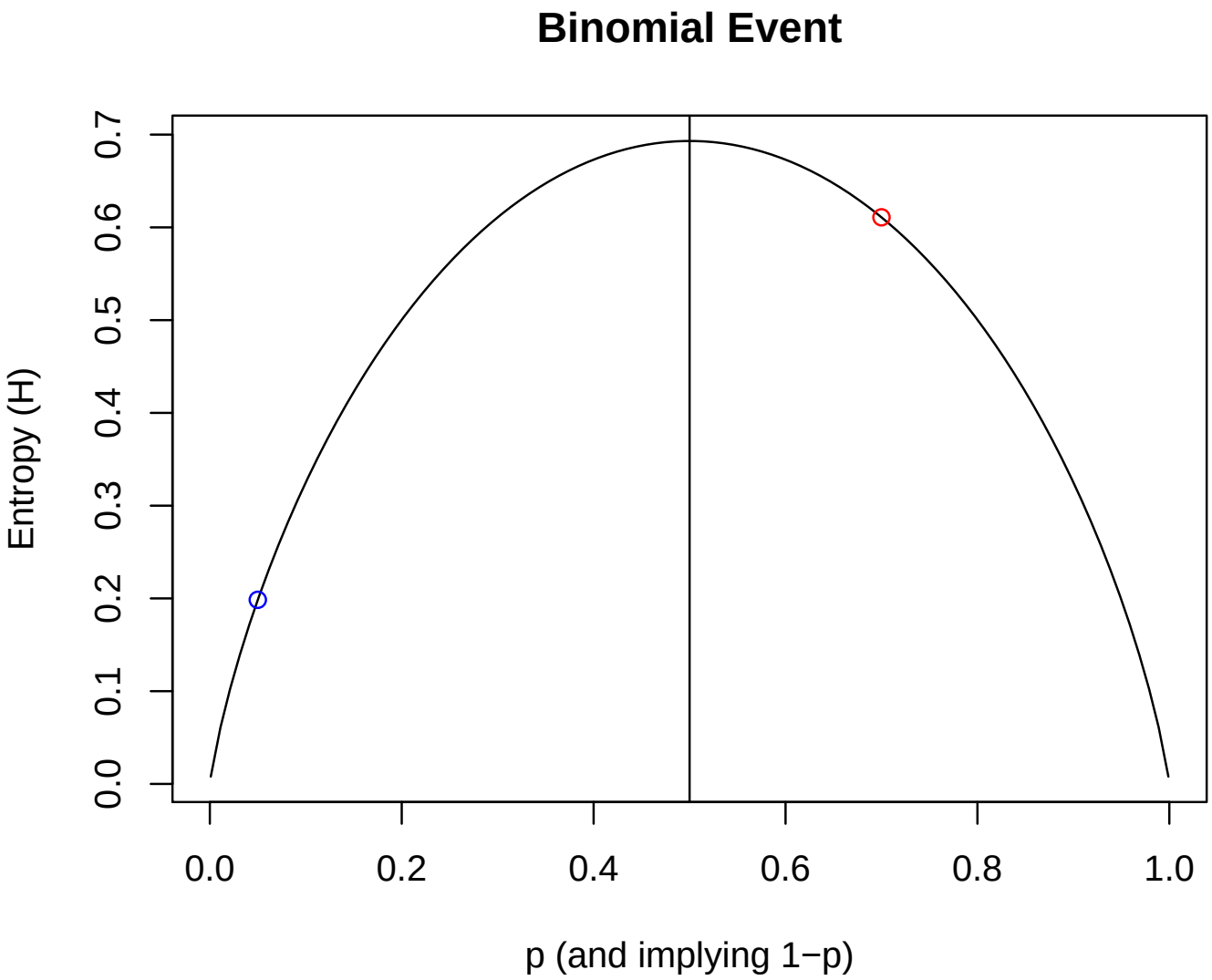
- Probability of answering "yes" is $p_1 = .7$ and "no" $p_2 = .3$
  $H(p) = -\sum_{i=1}^{2} p_i \log(p_i) = -(p_1 \log(p_1) + p_2 \log(p_2)) \approx .61$
- in R:

```
> p <- c(.3, .7)
> -sum(p*log(p))
[1] 0.6108643
```

- More certainty about an event:
- High probability of answering "yes", e.g $p_1 = .95$ with "no" $p_2 = .05$
  $H(p) \approx .20$
- Less certainty about an event:
- For $p_1 = .5$ and $p_2 = .5$
  $H(p) \approx .70$

Information Entropy

Changes in uncertainty/entropy



Information Entropy

Changes in uncertainty/entropy

- Entropy itself has no real meaning for us
- But - it defines the "playfield"
- Also, serves as reference point
- At some point we want to find a good model for an outcome/event
  - A good model is going to reduce uncertainty, i.e., entropy
- ▷ Entropy values can be used to build a measure of accuracy

From Information to Accuracy

- How can we use information entropy to say how far a model is from the target?

- Two probability distributions: p, q
- How accurate is q for describing p?
- Distance from q to p: Divergence
- **Divergence**: The additional uncertainty induced by using probabilities from one distribution to describe another distribution.
- Common measure of divergence among a "true" probability distribution p, and an arbitrary probability distribution q:
  - *Kullback-Leibler divergence* (D_{KL}) or information divergence, information gain, or relative entropy

KL as measure of surprise

- KL divergence is the "unnecessary surprise"
  - Suppose a number X is about to be drawn randomly from a discrete set with probability distribution $p(x)$.

    If Alice knows the true distribution $p(x)$, while Bob believes that the distribution is $q(x)$, then Bob will be more surprised than Alice, on average, upon seeing the value of X.

    The KL divergence is the (objective) expected value of Bob's (subjective) surprisal minus Alice's surprisal. In this way, the extent to which Bob's belief is "wrong" can be quantified in terms of how "unnecessarily surprised" it is expected to make him.

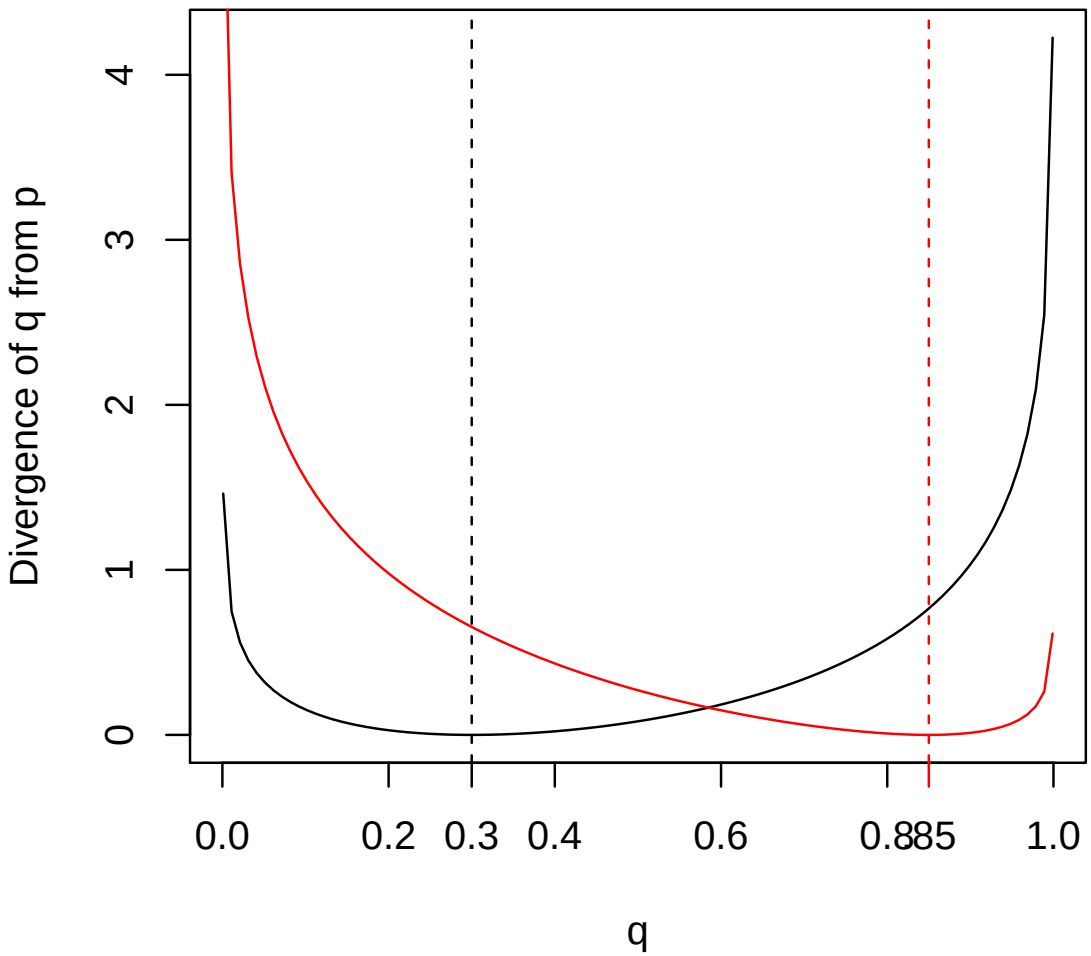
Estimating divergence

- How to estimate D_{KL}?

  $D_{\mathsf{KL}}(p, q) = \sum_i p_i \left[\log(p_i) - \log(q_i)\right]$
- Divergence is the average difference in log probability between the target (p) and model (q)
- Example: True distribution of events is $p_1 = .3$, $p_2 = .7$
- We believe: $q_1 = .25$, $q_2 = 0.75$
- $D_{\mathsf{KL}} = 0.006$
- If we belive: $q_1 = .5$, $q_2 = 0.5$
- $D_{\mathsf{KL}} = 0.082$
- For $p = q$ $D_{\mathsf{KL}} = 0$

Estimating divergence

Asymmetry



- Divergence is 0 for $p = q$
- Distance $H(p, q)$ is not equal to $H(q, p)$

```
DKL <- function(p,q){
sum(p*(log(p) - log(q)))
}

> p <- c(.3, .7)
> q <- c(.85, .15)
> DKL(p,q)
[1] 0.7658754
> DKL(q,p)
[1] 0.654169
```

Estimating divergence

- Problem:
  - We typically don't know p
- Good news – we don't need to know!
  - Focus is on difference between two approximating models:

$$D_{\mathsf{KL}}(p, q) - D_{\mathsf{KL}}(p, r) = -\sum_i p_i(\log(q_i) - \log(p_i)) - \left[-\sum_i p_i(\log(r_i) - \log(p_i))\right]$$
$$= -\sum_i p_i(\log(q_i) - \log(r_i)) = -(\mathsf{E}\log(q_i) - \mathsf{E}\log(r_i))$$

- While we don't know where p is, we can estimate how far apart q and r are, and which is closer to the target!
- All we need to know is a model's average log-probability: $\mathsf{E}\log(q_i)$ for q and $\mathsf{E}\log(r_i)$ for r

From Divergence to Deviance

- Why all the fuzz about information theory and divergence?
- Point was to establish:

1. How to measure the distance of a model from our target.
   - ▷ Information theory gives us the distance measure we need, the K-L divergence.
2. How to estimate the divergence.
   - ▷ Having identified the right measure of distance, we now need a way to estimate it in real statistical modeling tasks.

From Divergence to Deviance

- As noted above, we can use average log-probabilities $\mathrm{E}\log(q_i)$ and $\mathrm{E}\log(r_i)$ to inform us about divergence.
- *Relative* model fit: We don't know how well a single model does, but we know which one does better in comparison

- Let's focus on one part: q
- Deviance, *estimate* of relative information divergence:

  $D(q) = -2\sum_i \log(q_i)$
- Compute it:
  - Compute log probability of each observation
  - Sum all of these log probabilities
  - Multiply by -2

From Divergence to Deviance

Example

```
> N = 50
> x <- runif(N, -5, 5)
> y = 0.5 + 0.2*x + rnorm(N, 0, 0.3)
> m0 <- lm(y ~ x)
> mu <- fitted(m0)
> sample_sd <- sqrt(sum(resid(m0)^2)/N)
> cbind(y, mu, sample_sd)
y            mu sample_sd
1   0.5726698080  0.318857866 0.2821482
2   0.8967246851  1.133804537 0.2821482
3   1.6171938429  1.238156404 0.2821482
...
49   0.3172537458  0.330342058 0.2821482
50   0.3916993868  0.894527560 0.2821482
> deviance <- -2*sum(dnorm(y, mu, sample_sd, log = TRUE))
> deviance
[1] 15.36157
```

From Divergence to Deviance

Example

- R provides the `logLik` function
  - Extracts the sum of all log-likelihoods

```
> deviance
[1] 15.36157
> -2*logLik(m0)
'log Lik.' 15.36157 (df=3)
```

From Deviance to Out-Of-Sample

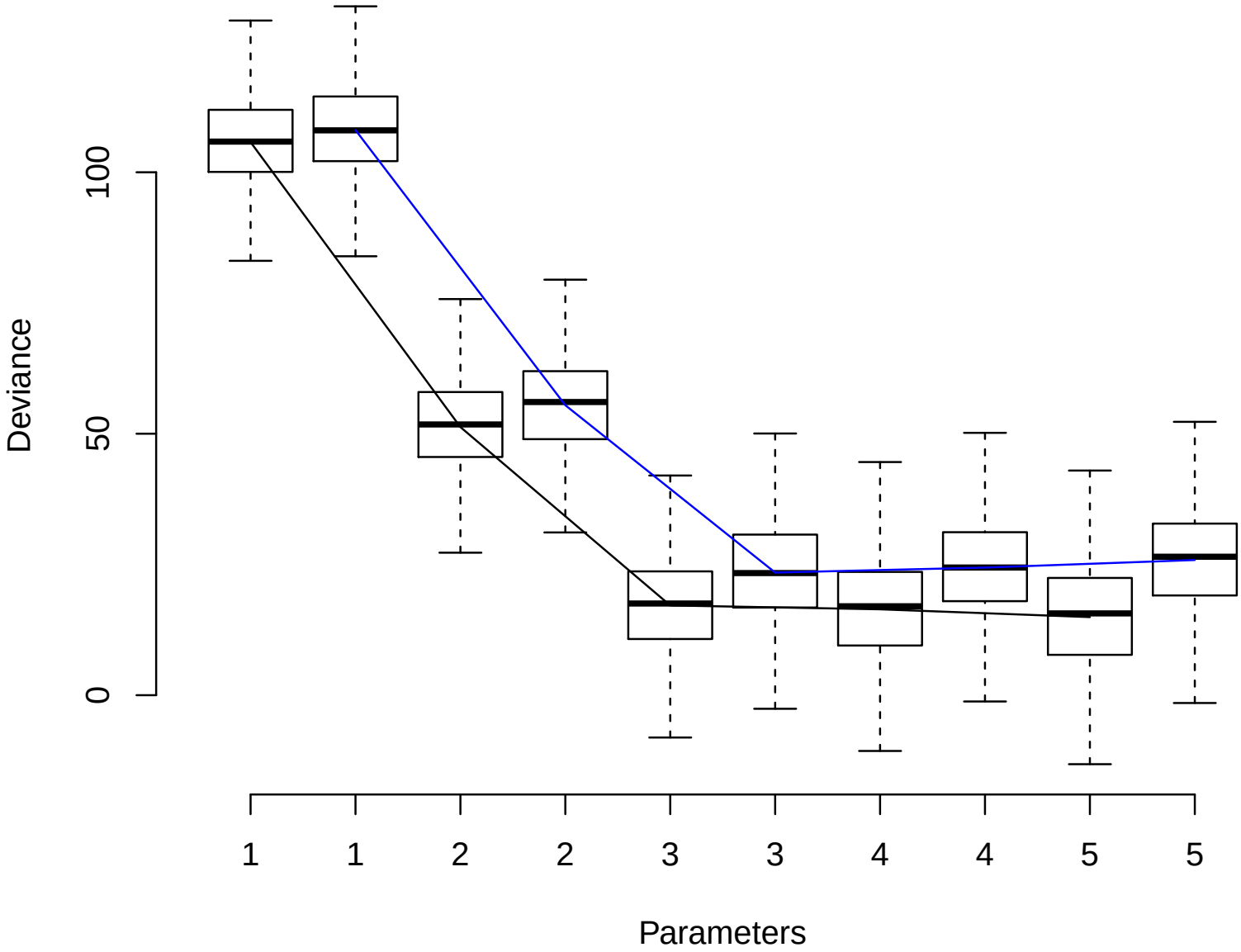
- Deviance is a principled way to measure distance from the target.
- Deviance as computed on previous slides has the same flaw as R^2: It always improves as the model gets more complex
- Deviance in-sample is a measure of retrodictive accuracy, not predictive accuracy

From Deviance to Out-Of-Sample

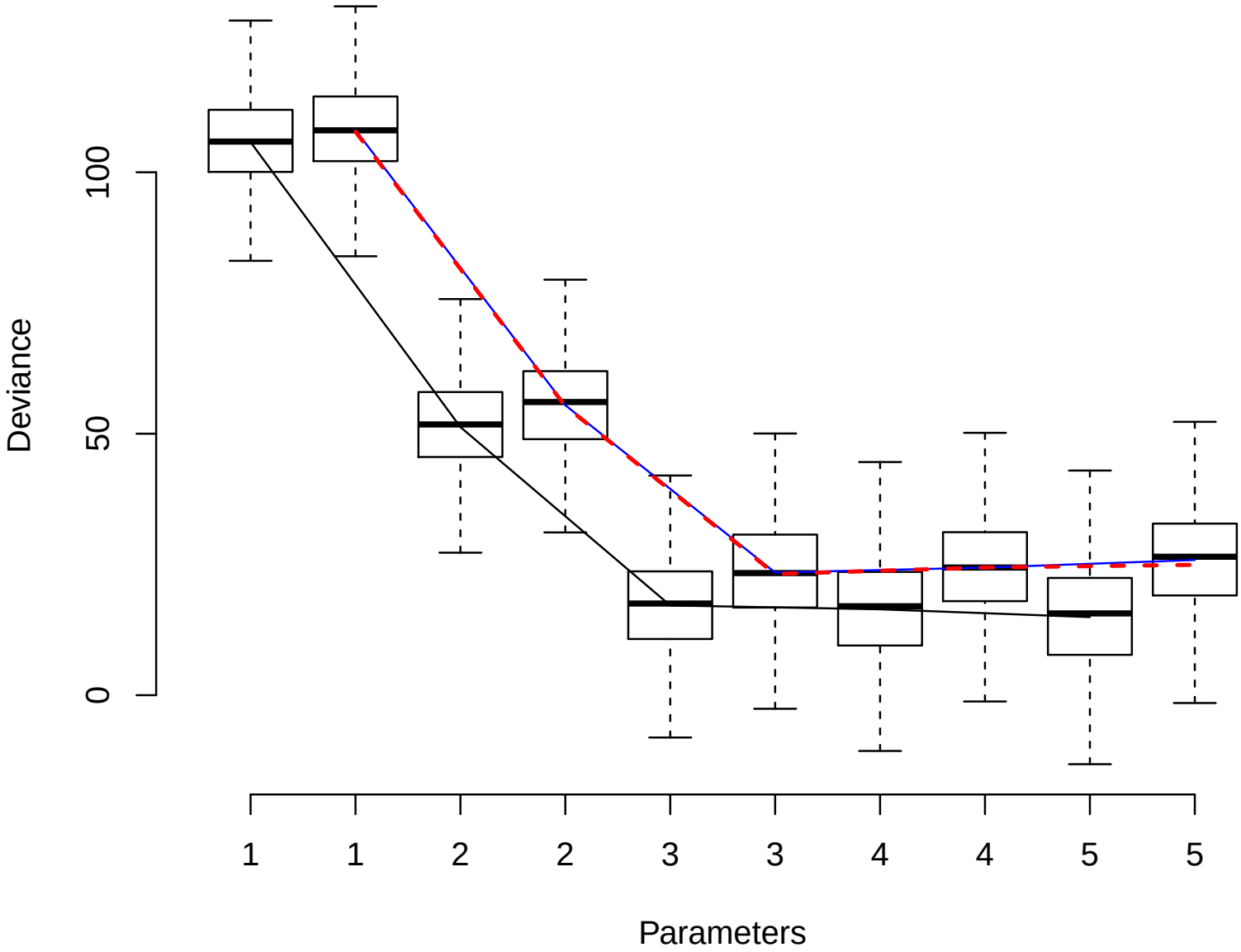
A meta-model of forecasting:

- Cross validation
- Two samples: training and testing, size N
- Fit model to training sample, get D_{train}
- Use fit to training to compute D_{test}
- Difference $D_{\text{test}} - D_{\text{train}}$ is overfitting

From Deviance to Out-Of-Sample



From Deviance to Out-Of-Sample



Akaike information criterion

Under some strict conditions:

$$\text{AIC} = D_{\text{train}} + 2k \approx \text{E}D_{\text{test}} \text{ with } k \text{ number of parameters}$$

- Expected deviance out-of-sample is the deviance in-sample plus 2 $\times$ parameters
- AIC provides an approximation of predictive accuracy, as measured by out-of-sample deviance.
- All information criteria aim at this same target, but are derived under more and less general assumptions
- AIC approximation requires
  - multivariate Gaussian
  - sample size N is much greater than the number of parameters k

Akaike information criterion

$$\text{AIC} = D_{\text{train}} + 2k \approx \text{E}D_{\text{test}}$$

- Conditions:
- No varying/mixed/random effects
- $k << N$ as k approaches N:

$$\text{AICc} = D_{\text{train}} + \frac{2k}{1 - (k+1)/N}$$

Akaike information criterion

Example

- Model Comparison with AIC
- Pep, no-pep
- Comparisons must be based on exactly the same observations
- Fewer observations result in lower deviance (less to sum up), and better AIC

```
mod0 <- glm(y~1, family=binomial, data=ilse)
mod1 <- glm(y~sex, family=binomial, data=ilse)
mod2 <- glm(y~sex+sds.c, family=binomial, data=ilse)
mod3 <- glm(y~sex+sds.c+neo.c, family=binomial, data=ilse)
```

Akaike information criterion

Model Comparison with AIC

```
> summary(mod3)

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.24692    0.15505   8.042 8.82e-16 ***
sex         -0.47353    0.20580  -2.301   0.0214 *
sds.c       -0.04552    0.01889  -2.410   0.0160 *
neo.c       -0.10881    0.01881  -5.786 7.20e-09 ***
---

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 709.32  on 580  degrees of freedom
Residual deviance: 616.45  on 577  degrees of freedom
AIC: 624.45
```

Akaike information criterion

Model Comparison with AIC

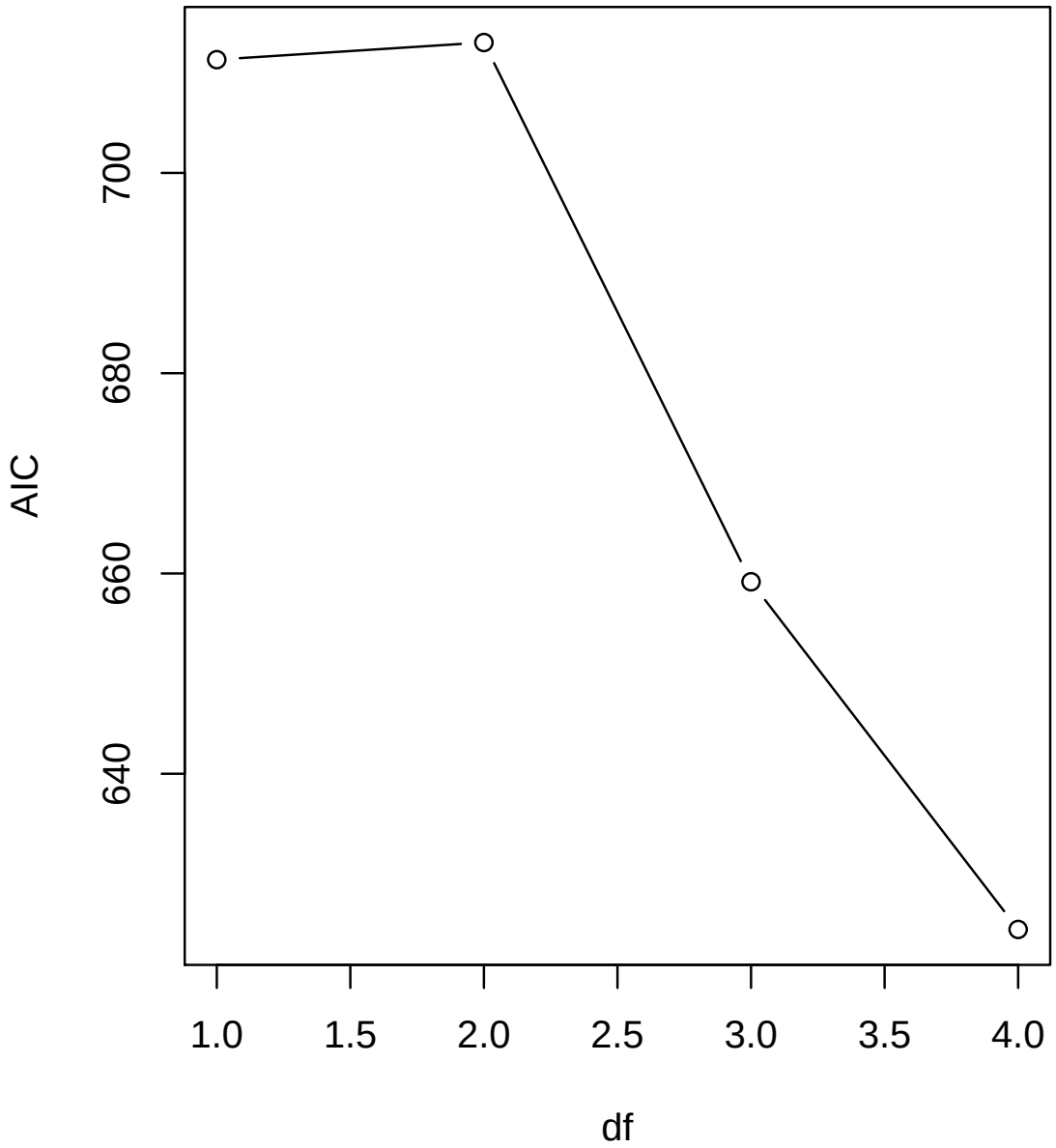
- Extract Deviances for each model and obtain difference

```
> anova(mod0, mod1, mod2, mod3)
Analysis of Deviance Table

Model 1: y ~ 1
Model 2: y ~ sex
Model 3: y ~ sex + sds.c
Model 4: y ~ sex + sds.c + neo.c
Resid. Df Resid. Dev Df Deviance
1       580     709.32
2       579     709.03  1    0.288
3       578     653.17  1   55.853
4       577     616.45  1   36.721
```

Akaike information criterion

Model Comparison with AIC



- Obtain AIC

```
> AIC(mod0, mod1,
      mod2, mod3)
     df       AIC
mod0  1 711.3152
mod1  2 713.0274
mod2  3 659.1748
mod3  4 624.4542
```

Information Criteria

- Based on information theory
- Relative measures
  - Individual deviance has no meaning
  - But they can be compared relative to each other
  - ▷ Smallest deviance indicates best fit
- Deviance is comparable to R^2 in the sense that more parameters decrease deviance
- ▷ Penalized versions approximate out-of-sample accuracy
- Several measures (AIC, BIC, DIC, WAIC...)