AIML427 Big Data

Week 8-9: Regression 2: Moving Beyond Linearity

Dr Qi Chen

School of Engineering and Computer Science

Victoria University of Wellington

Qi.Chen@ecs.vuw.ac.nz

Outline

- Penalised Methods for Classification
  - Penalised Logistic Regression
- Issues that arise in high dimensions, i.e. p > n
- Going Beyond Linearity
- Regression Splines:
  - natural splines and smoothing splines
- Generalised Additive Models

Penalised Methods For Classification

- The methods we discussed before have analogues for classification:
  - Combine the features X linearly
  - Introduce a penalty term with a tuning parameter λ that controls the bias-variance trade-off
    - λ is typically chosen by cross-validation

- But we have to decide *what error to minimise* based on whether we make probabilistic predictions or definitive predictions for the class labels:
  - Deviance, cross-entropy
  - Mean squared error or mean absolute error
  - Misclassification error – are some misclassifications worse than others?
  - Area-under-the-curve (AUC)

Logistic Regression Model

- Y = Binary response. X = Quantitative predictor.
- π = probability of 1's at any X
- Equivalent forms of the logistic regression model:

Probability form

$$p = \frac{e^{b_0 + b_1 X}}{1 + e^{b_0 + b_1 X}}$$

Logit form

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 X$$

N.B.: This is natural log (aka "ln")

Penalised Logistic Regression

- Logistic regression is a method for binary classification.
- If we use 0 and 1 to code the class labels, the output of logistic regression for test case *i* is

$$\hat{\pi}_i = \mathrm{P}(y_i = 1)$$

- Turn this into a definitive classification via a threshold t:

$$\hat{y}_i = \begin{cases} 1, & \hat{\pi}_i \geq t \\ 0, & \hat{\pi}_i < t \end{cases}$$

- Deviance is $-\sum_i \{y_i \, log\hat{\pi}_i + (1 - y_i) \, log(1 - \hat{\pi}_i)\}$
- Misclassification error is $\sum_i I\{y_i \neq \hat{y}_i\}$

- Default is usually t = 0.5

Penalised Logistic Regression

- The **Credit** dataset: We will consider *whether it is possible to predict which people have a credit card balance greater than 20% of their monthly income*.

```
> y = as.numeric(balance/(income*1000/12)>0.2)
> sum(y) [1] 136
```

- *as.numeric converts boolean TRUE/FALSE into 1/0*
- Note that the number of 0s (400 − 136 = 264) is roughly twice the number of 1s (136).
- Care has to be taken with *unbalanced* datasets like this. Our classifier will need a misclassification error rate much better than the 0.33.
  - Performance measures for unbalanced classification: Precision and Recall, Average Class Accuracy, AUC

Penalised Logistic Regression

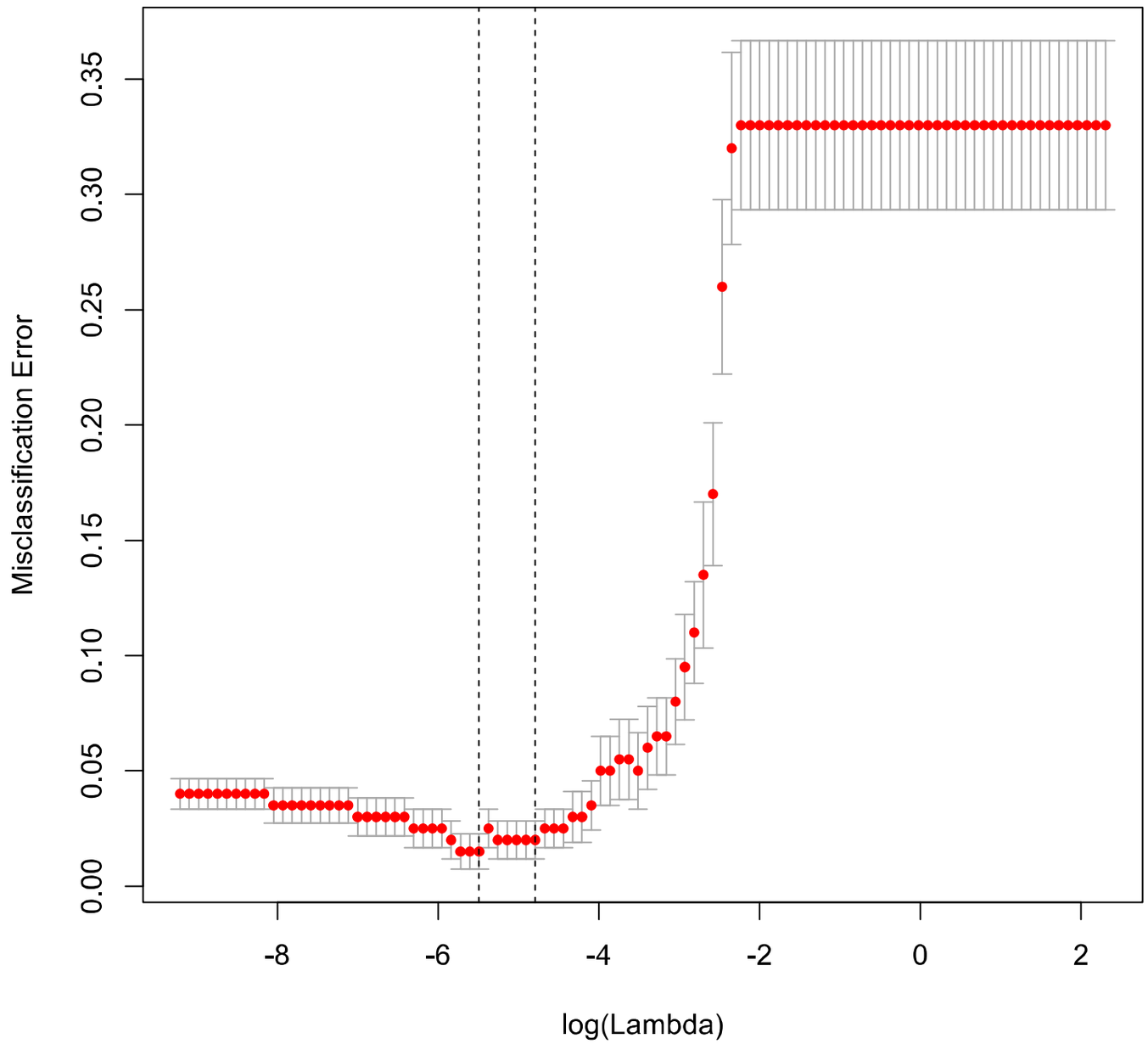
- glmnet allows us to do logistic regression with a ridge regression-type penalty or a lasso penalty.
- Given the matrix X of features and the training and test split, perform the lasso version of logistic regression as follows:

```
> grid = 10^seq(1,-4,100)
> set.seed(987654313)
> cv.out = cv.glmnet(X[train,],y[train],alpha=1,lambda=grid,nfolds=10,thresh=1e-12,
family="binomial",type.measure="class")
```

- family="binomial" specifies to do logistic regression
- type.measure="class" indicates we are using misclassification error

Penalised Logistic Regression

```
> plot(cv.out)
```



Penalised Logistic Regression

- With the CV-selected value for λ, we can make our predictions for the test data:

```
> bestlam = cv.out$lambda.min
>lasso.pred = predict(cv.out,s=bestlam,newx=X[test,],type="class")
>table(lasso.pred,y[test])

lasso.pred  0     1
            0     129   6
            1     1     64
```

- 1 false positive and 6 false negatives
- The misclassification error rate is 3.5%
- We can improve the misclassification error rate to 1% by choosing threshold t = 0.4
- In fact, an AIC-selected logistic regression yields a *perfect* classifier

Penalised Logistic Regression

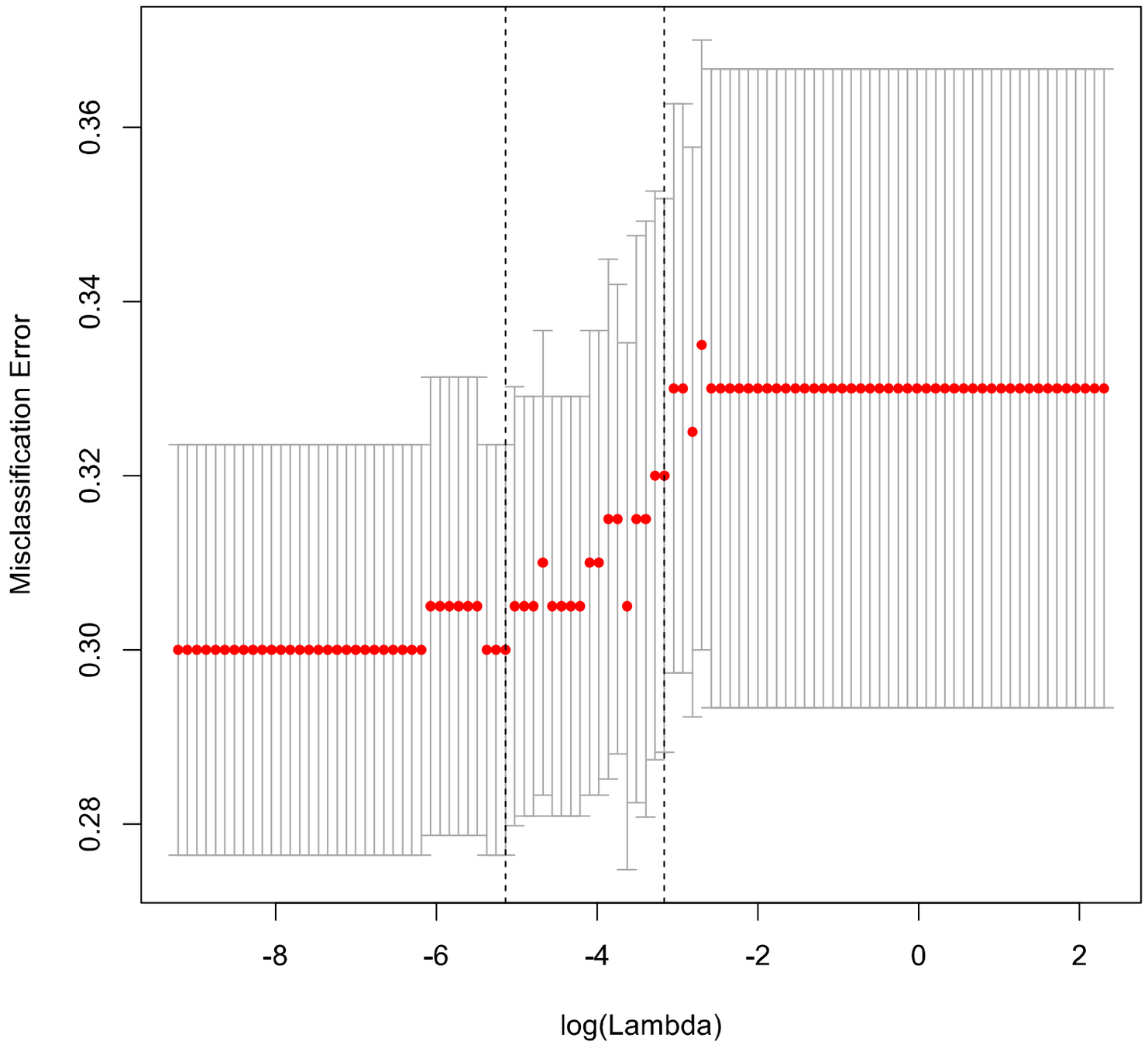
- It turns out limit and rating are highly informative for the class labels.
- The causal mechanism actually runs the other way: *y predicts limit and rating*. *Removing them as features destroys the classifier*:

```
> X = X[,-c(2,3)]
> set.seed(987654313)
> cv.out = cv.glmnet(X[train,],y[train],alpha=1,lambda=grid,nfolds=10,thresh=1e-12,
family="binomial",type.measure="class")
>bestlam = cv.out$lambda.min
> lasso.pred = predict(cv.out,s=bestlam,newx=X[test,],type="class")
> table(lasso.pred,y[test])
lasso.pred  0      1
           0      117   47
           1      13    23
```

- The misclassification error rate is 30%

Penalised Logistic Regression

```
> plot(cv.out)
```



Collinearity and Penalised Methods

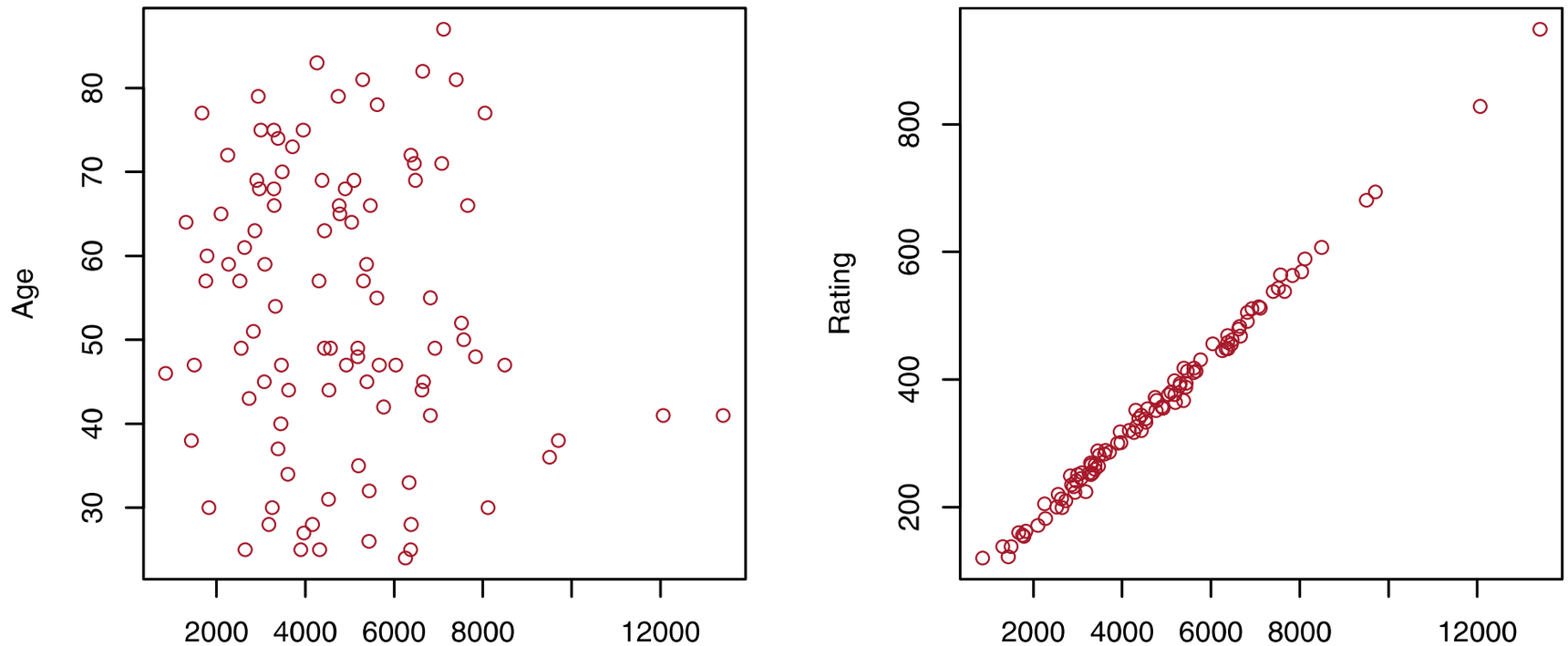
If two or more predictor variables are highly correlated with each other, they are said to be **collinear**.

Collinearity is always a problem for regression, and unfortunately *penalised methods do not fix this*.

- *Limit* and *rating* were obviously correlated in **Credit**.
  - Small changes in the data lead to large changes in the corresponding regression coefficients. This affects interpretability
  - One solution is to drop one of the predictors; alternatively we could combine them into a single predictor
  - *Detecting collinearity* becomes harder when the number of predictors grows
  - *Multicollinearity* can occur between *3 or more variables*, even if the pairwise correlations are small; this is even harder to detect
  - See also ISLR Section 3.3.3 and 6.4.4

Collinearity

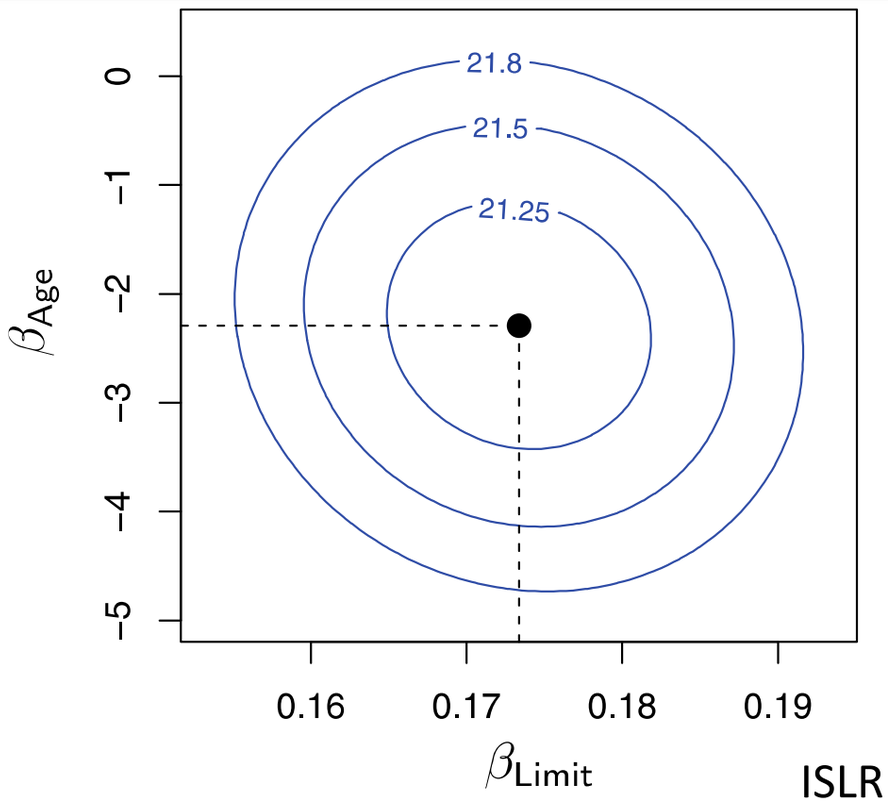
ISLR: FIGURE 3.14.
Table 3.11

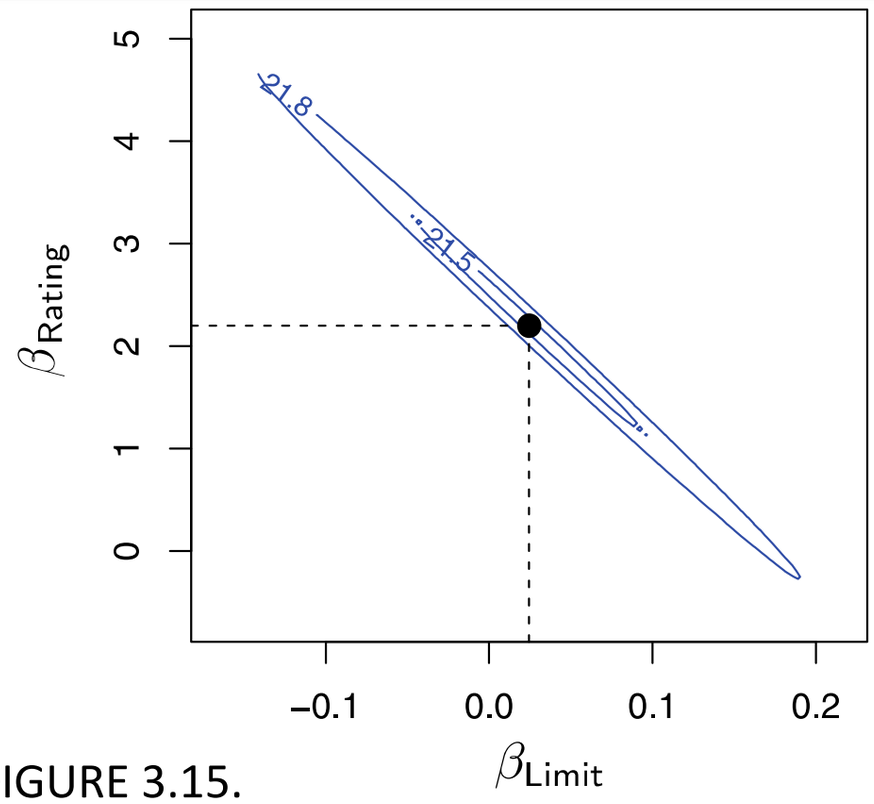


- Collinearity reduces the accuracy of the estimates of the regression coefficients, it causes the standard error for β to grow.

		Coefficient	Std. error	t-statistic	p-value
Model 1	Intercept	−173.411	43.828	−3.957	< 0.0001
	age	−2.292	0.672	−3.407	0.0007
	limit	0.173	0.005	34.496	< 0.0001
Model 2	Intercept	−377.537	45.254	−8.343	< 0.0001
	rating	2.202	0.952	2.312	0.0213
	limit	0.025	0.064	0.384	0.7012

Collinearity





ISLR: FIGURE 3.15.

Left: A contour plot of RSS for the regression of balance onto age and limit. The minimum value is well defined.

Right: A contour plot of RSS for the regression of balance onto rating and limit. Because of the ***collinearity***, many pairs (β_{Limit},β_{Rating}) with a similar value for RSS, **leads to a great deal of *uncertainty*:**

- A broad range of values for the coefficient estimates smallest RSS
- A small change in the data could cause the pair of coefficient values to move anywhere along this valley

Issues In High Dimensions

- *A dataset is said to be high-dimensional if the number of features is greater than the number of observations*, i.e ***p > n***.
- In the last 20 years or so such datasets have become routine, Examples:
  - Images: a single image can correspond to millions of pixel values
  - Genomics: sequence data for an individual, SNPs, gene expression
  - Marketing: search terms, buying behaviour, location information
- The situation when **p ≫ n** is sometimes called the "***large p, small n***" problem
- See also ISLR Section 6.4

Issues In High Dimensions

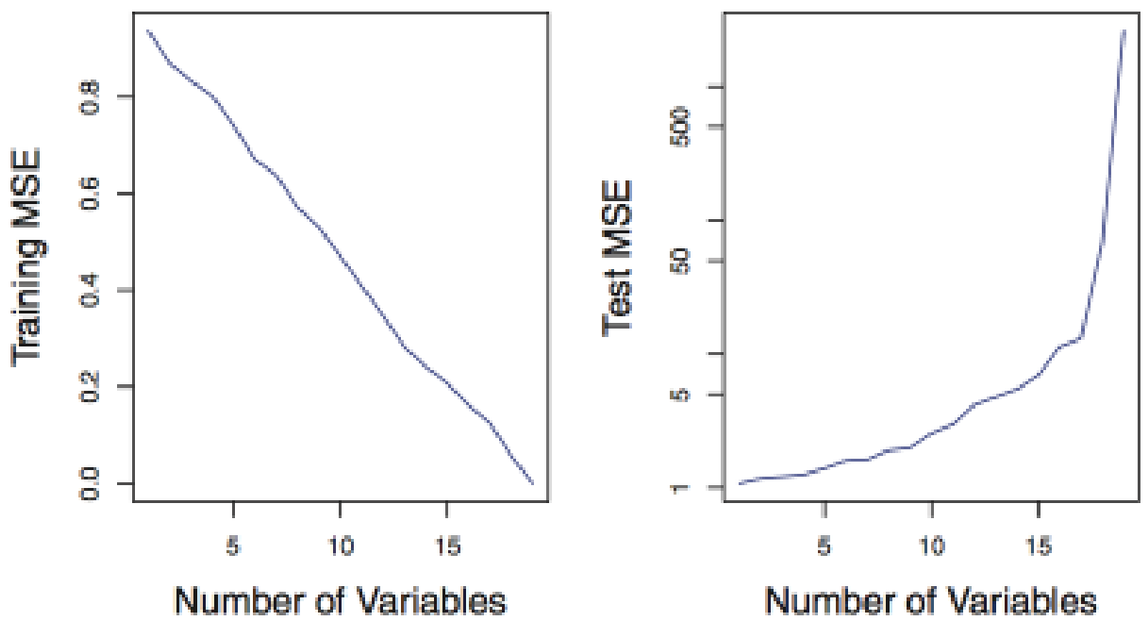
- Unfortunately, most classical statistical methods – such as **least squares** – *do not work in high dimensions*.

- The reason is that the models should be able to fit the observations exactly, this is almost always going to be a case of overfitting. Furthermore, the model fit will not be unique – there will be lots of ways to overfit the data exactly!

- Fortunately, less flexible methods – such as penalised/ regularisation/shrinkage methods – allow us to perform regression and classification in high-dimensional settings,
  - as long as we take due care.

The "Curse Of Dimensionality"

- The fundamental issue is "noise".
  - By adding in many more features – even if some of them are informative – actually adding in more "noise".
  - work very hard to avoid fitting this "noise". This is often referred to as *the curse of dimensionality*.
- An example: n = 20 observations, and regression with between 1 and 20 features, each of which was completely unrelated to the response. Including additional predictors leads to a vast increase in the variance of the coefficient estimates

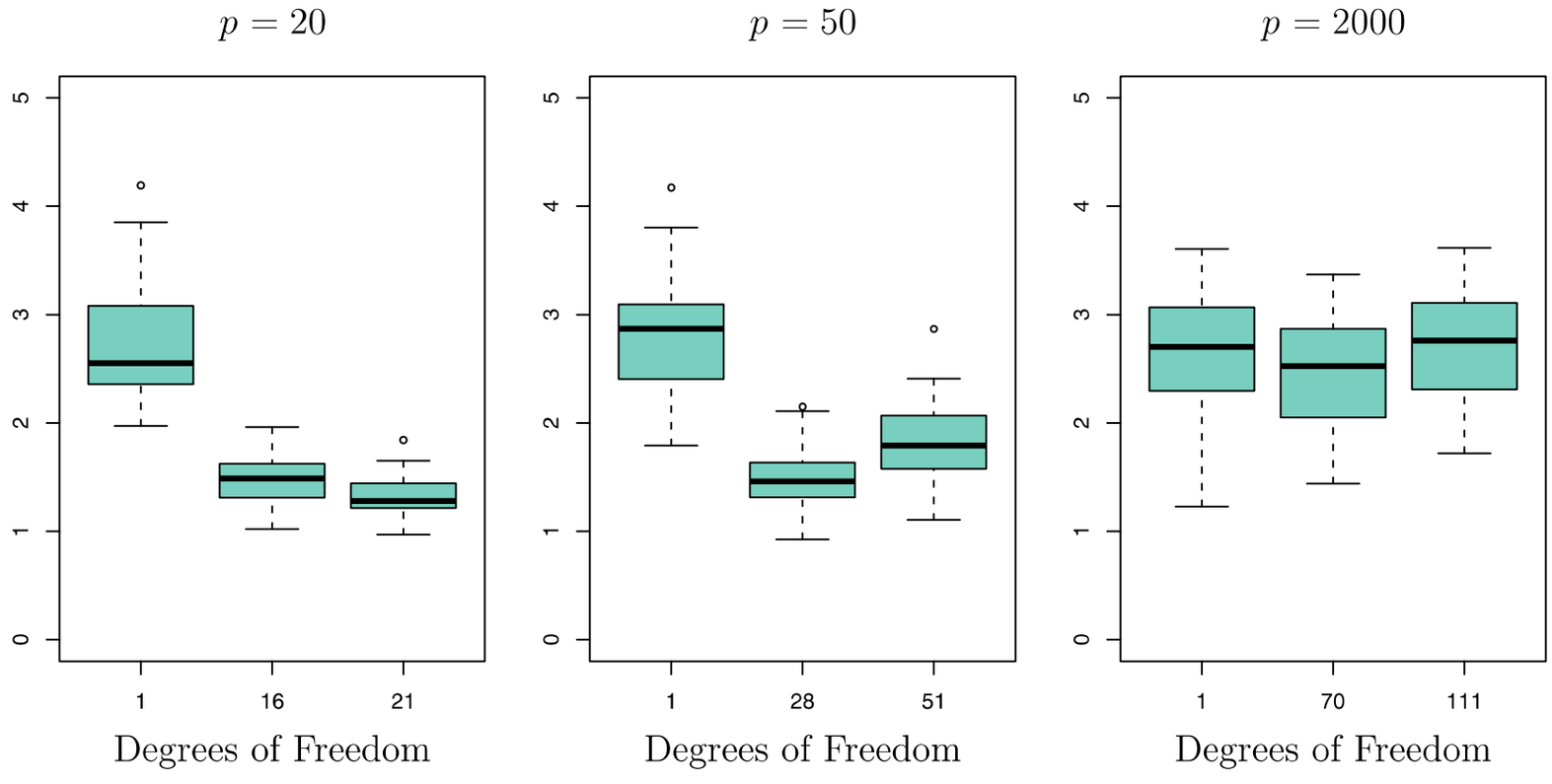


ISLR Figure 6.23

The Lasso in High Dimensions

- The lasso when there are n = 100 observations and p features, of which only 20 are truly informative.
  - The degrees of freedom is the number of non-zero coefficients selected by the lasso as λ changes
  - Lasso continues to work when p > n, but fails in ultra high dimensions.
  - As a rule of thumb, for n = 100, we require p < 1000; for n = 500, we require p < 10, 000.
  - Other methods exist for ultra high dimensions, e.g. elastic net, the smoothly clipped absolute deviation (SCAD), MC+

ISLR Figure 6.24:
Test error (MSE)



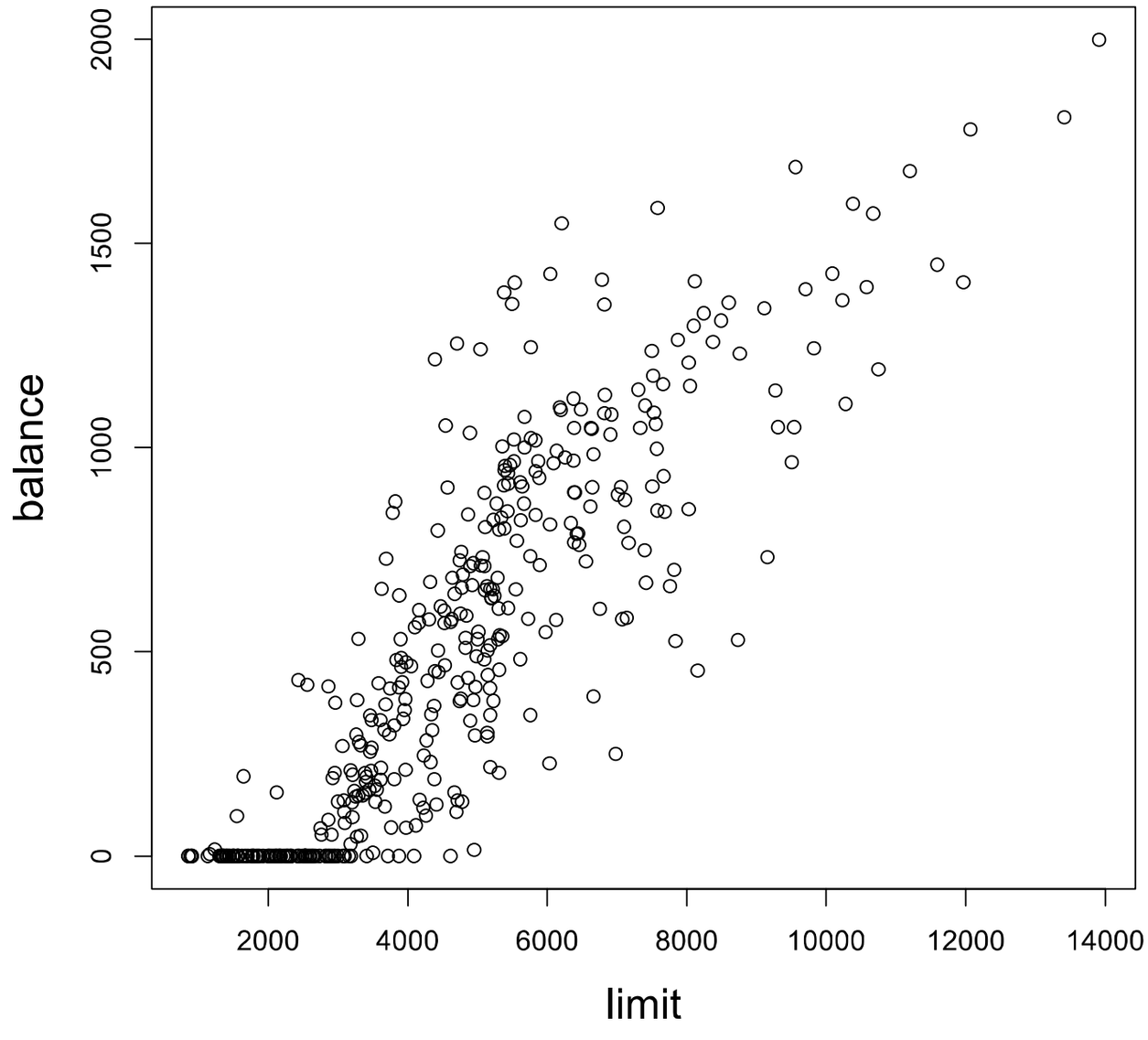
Interpreting Results in High Dimensions

One must take care when reporting results in high-dimensional settings.

- Informative features can easily be overlooked: the additional variance may outweigh the reduction in bias
- Any variable can be written as a linear combination of all the others; which ones are truly informative?
- There may be many (small) subsets of features that have predictive power; a useful model might exist, but it is probably not going to be the only possible one

Going Beyond Linearity

- For the **Credit** dataset, a linear relationship between the response and the features might not be an appropriate assumption:



Simple Extensions of Linear Models

Consider the simplest case when there is only one predictor/variable/feature – call it *x*.

- Ordinary linear regression is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

- Polynomial regression:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i$$

- Step Functions: create cutpoints c_1, c_2, . . . , c_K in the range of x, and then construct K + 1 new variables:

$$\begin{aligned} C_0(X) &= I(X < c_1), \\ C_1(X) &= I(c_1 \leq X < c_2), \\ C_2(X) &= I(c_2 \leq X < c_3), \\ &\vdots \\ C_{K-1}(X) &= I(c_{K-1} \leq X < c_K), \\ C_K(X) &= I(c_K \leq X), \end{aligned}$$

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \ldots + \beta_K C_K(x_i) + \epsilon_i.$$

  - The intervals are non-overlapping and taken together cover the whole range of *x*.

Generalised Additive Models

Extensions of Linear Models

- The ***generalised additive model (GAM)*** for regression is

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

where the f_j are p possibly nonlinear functions of a single variable

- See also ISLR Section 7.7

Generalised Additive Models

- The advantages of GAMs are
  - automatically fit a nonlinear f_j for each feature x_j without manually trying out different transformations
  - Nonlinear fits can potentially lead to more accurate predictions
  - The model is interpretable
    - Still additive mode so we can look at the effect of x_j on y while holding the other variables fixed,
    - The smoothness of the functions f_j can be quantified by degrees of freedom
- The main disadvantage of GAMs is that additivity may still be too restrictive. For example, With many variables, important interactions can be missed.
  - Pairwise interactions of the form $f_{jk}(x_{ij}, x_{ik})$ can be included with a little bit of effort
  - GAMs are useful compromise between linear models and more flexible approaches like **random forests** and **boosting**

Regression Splines: Spline Basis Representation

ISLR Section 7.4

- A regression **spline** models the response y as separate low-degree polynomials defined on different intervals of *x*.
- A cubic spline with K knots:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i$$

  - Basis functions: $b_1, b_2, \ldots, b_{K+3}$
  - More flexible than polynomials and step functions, and in fact are an extension of the two
  - **Importantly**, the polynomials are required to meet smoothly at the interval endpoints (the coefficients change), known as knots
- The modelling questions are:
  - How many knots are there?
  - Where do we put the knots?
  - What is the degree of the polynomials?
- Once we have answered these questions, we simply fit the model using least squares.

Natural Splines

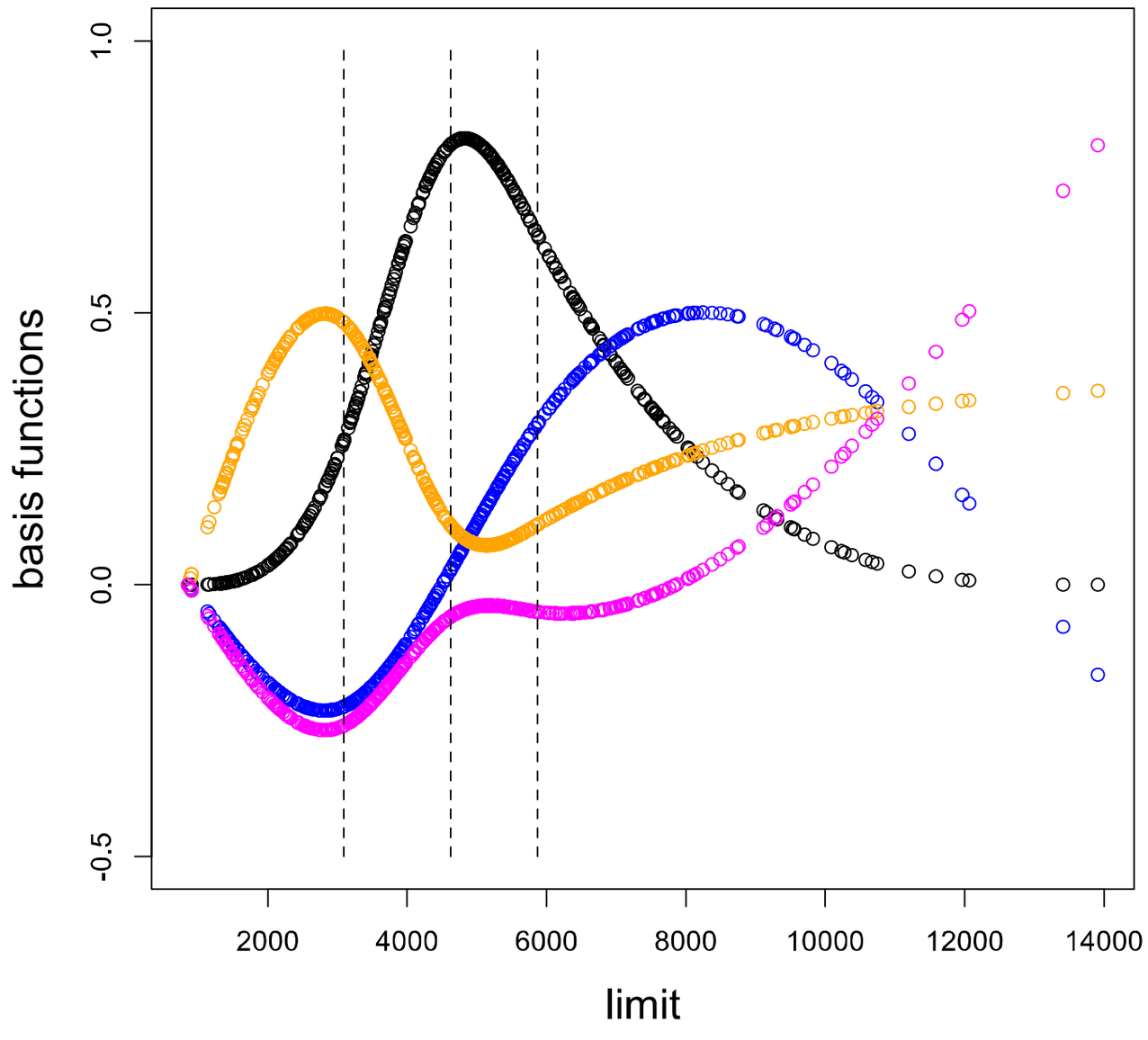
- Natural splines have additional constraints: they are required to be linear at the boundary, i.e. the second derivative is zero there. This makes estimates at the boundary much more stable
- We use the splines package in R to find the natural spline for balance in terms of limit. First, we find the appropriate **basis** functions on the range of limit using ***ns***

```
> attach(Credit)
> library(splines)
> ns.basis = ns(limit,df=4)
> attr(ns.basis,"knots")
       25%        50%        75%
      3088.00    4622.50 5872.75
> attr(ns.basis,"Boundary.knots")
[1]    855        13913
```

- The degrees of freedom df sets the number of basis functions
- The number of (interior) knots is one less than the degrees of freedom
- By default the knots are put at evenly spaced quantiles of x

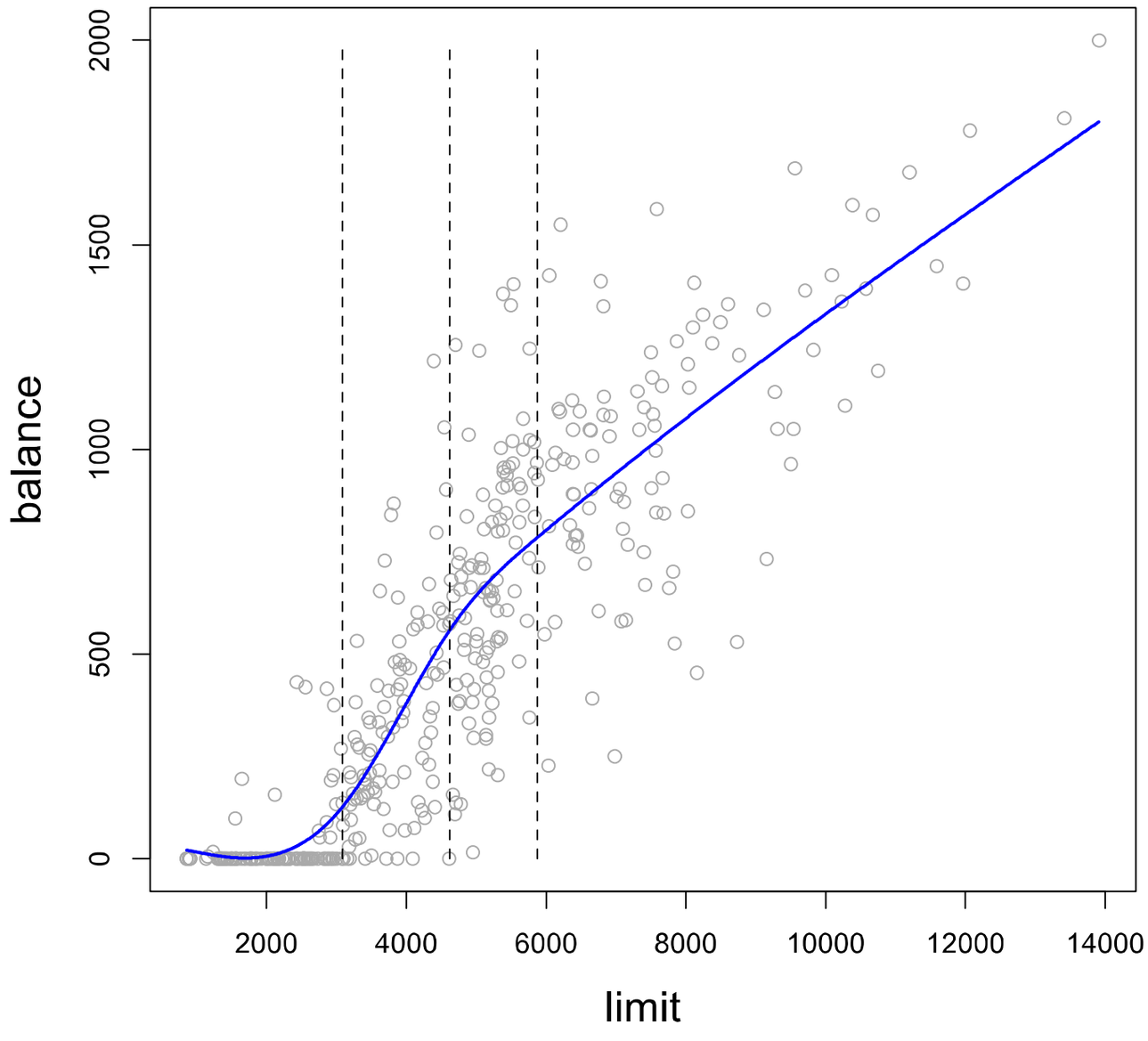
Natural Splines

```
> plot(limit,ns.basis[,1],ylim=c(-0.5,1),ylab="basis functions",cex.lab=1.5)
> points(limit,ns.basis[,2],col="blue")
> points(limit,ns.basis[,3],col="orange")
> points(limit,ns.basis[,4],col="magenta")
```



Natural Splines

```
> ns.fit = lm(balance~ns(limit,df=4))
> lim.grid = seq(min(limit),max(limit),10)
> ns.pred = predict(ns.fit,newdata=list(limit=lim.grid))
> plot(limit,balance,cex.lab=1.5,col="darkgrey")
> lines(lim.grid,ns.pred,col="blue",lwd=2)
```



Natural Splines

- The number of knots can be chosen by cross-validation

- As we will see, when fitting GAMs we will have multiple splines.
  - Then it can be easiest to fix the degrees of freedom for all terms, e.g. to 4

- See also ISLR Section 7.4

Smoothing Splines

- A smoothing spline is the function that minimises

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

  - Loss+Penalty
  - λ>0 is a tuning parameter that controls the bias-variance trade-off.
  - The penalty term $\lambda \int g''(t)^2 dt$ prevents the smoothing spline from being too "wiggly"
- Remarkably, the smoothing spline is a shrunken version of a natural spline with knots at the unique values of $x_1, \ldots . x_n$
  - λ controls the amount of shrinkage:
    - as λ goes from 0 to ∞, the effective degrees of freedom goes down from n to 2
    - Not surprisingly, λ is typically selected by cross-validation
- See also ISLR Section 7.5

Smoothing Splines

- Fitting smoothing splines in R is very straightforward. The effective degrees of freedom can be set manually:

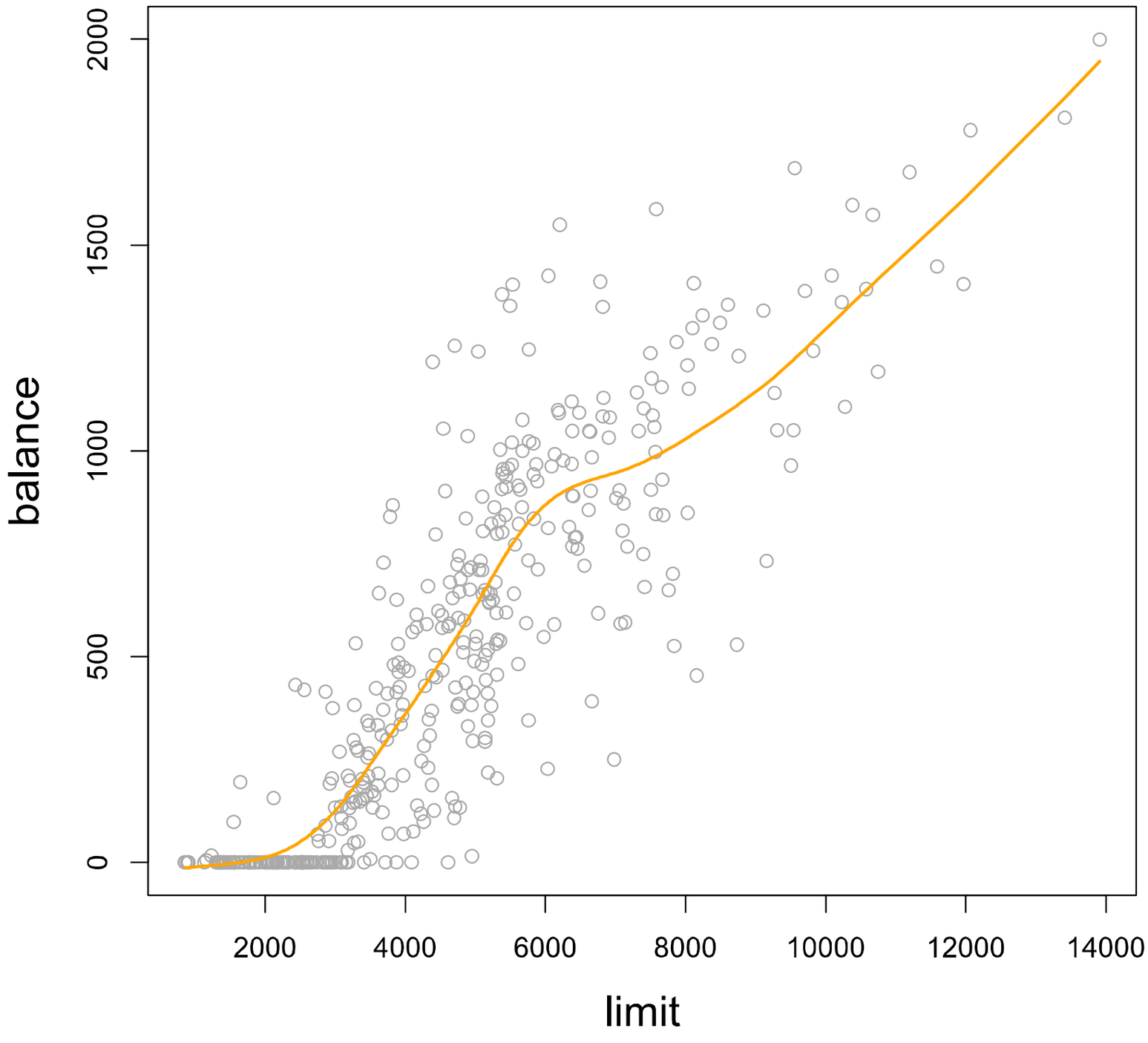
```
> ss.fit = smooth.spline(limit,balance,df=12)
```

- Or we can go straight to cross-validation, which is done by default:

```
> ss.fit = smooth.spline(limit,balance)
> ss.fit Call: smooth.spline(x = limit, y = balance)

Smoothing Parameter spar= 0.9513861 lambda= 0.001114027 (12 iterations)
Equivalent Degrees of Freedom (Df): 8.732188
```

Smoothing Splines

```
> plot(limit,balance,cex.lab=1.5,col="darkgrey")
> lines(ss.fit,col="orange",lwd=2)
```



Back To Generalised Additive Models

- It is very *convenient* in GAMs to use natural spline or smoothing spline functions of the features – though of course we are not restricted to these choices.
  - Fitting a GAM then amounts to **simultaneously** fitting all the splines.
- We return to the **Credit** dataset, restricting our attention to the features *income*, *limit* and *student* as predictors for *balance*.
  - We will use the gam package in R, which includes the gam procedure to fit GAMs
- See also ISLR 7.8.3

Back To Generalised Additive Models

```
> library(gam)
> gam.mod1 = gam(balance~income+limit+student)
> gam.mod2 = gam(balance~income+ns(limit,df=4)+student)
> gam.mod3 = gam(balance~ns(income,df=4)+ns(limit,df=4)+student)
> anova(gam.mod1,gam.mod2,gam.mod3,test="F")
Model 1: balance ~ income + limit + student
Model 2: balance ~ income + ns(limit, df = 4) + student
Model 3: balance ~ ns(income, df = 4) + ns(limit, df = 4) + student
  Resid. Df Resid. Dev Df Deviance F Pr(>F)
1       396    4316997
2       393    2059824  3  2257173 148.5396 < 2.2e-16 ***
3       390    1975449  3    84375   5.5525 0.0009692 ***
```

- gam.mod3 appears to be the best model
- This means there is evidence that *income* and *limit* contribute *nonlinearly* to balance

GAM Test Error

- Now we refit the models to a reduced (training) set and select the model with the best test error.

```
> set.seed(987654312)
> train = sample(1:nrow(Credit),nrow(Credit)/2)
> test = -train
> gam.mod1 = gam(balance~income+limit+student,data=Credit[train,])
> gam.mod2 = gam(balance~income+ns(limit,df=4)+student,data=Credit[train,])
> gam.mod3 = gam(balance~ns(income,df=4)+ns(limit,df=4)+student,data=Credit[train,])
> pred.mod1 = predict(gam.mod1,newdata=Credit[test,])
> pred.mod2 = predict(gam.mod2,newdata=Credit[test,])
> pred.mod3 = predict(gam.mod3,newdata=Credit[test,])
> mse1 = mean((pred.mod1-balance[test])^2)
> mse2 = mean((pred.mod2-balance[test])^2)
> mse3 = mean((pred.mod3-balance[test])^2)
> c(mse1,mse2,mse3)

[1]    11448.209 7181.638          7013.714
```

- gam.mod3 also has the smallest test error

GAM Test Error

- When keep adding more knots to the spline functions:

```
> gam.mod4 = gam(balance~ns(income,df=4)+ns(limit,df=9)+student,data=Credit[train,])
> pred.mod4 = predict(gam.mod4,newdata=Credit[test,])
> mean((pred.mod4-balance[test])^2)

[1]    7644.799
```

GAM With Smoothing Splines

- To use smoothing splines instead of natural splines in a GAM, we use the **s** command – which is part of the gam package – instead of **ns**. Note that s doesn't actually do any smoothing; it just sets up the variable to be used in *gam*
- Often there is not a lot of difference between using natural splines or smoothing splines, but this is not the case in the Credit dataset

```
> gam.mod4 = gam(balance~s(income,df=4)+s(limit,df=4)+student)
> gam.mod5 = gam(balance~s(income,df=4)+s(limit,df=9)+student)
> gam.mod6 = gam(balance~s(income,df=4)+s(limit,df=16)+studen)
> anova(gam.mod4,gam.mod5,gam.mod6,test="F")

Model 1: balance ~ s(income, df = 4) + s(limit, df = 4) + student
Model 2: balance ~ s(income, df = 4) + s(limit, df = 9) + student
Model 3: balance ~ s(income, df = 4) + s(limit, df = 16) + student

Resid. Df Resid. Dev Df Deviance F Pr(>F)
1     390          2124686
2     385          1894589  4.9995   230097   9.5441   1.376e-08 ***
3     378          1822819  7.0006    71770   2.1260    0.04015 *
```

GAM Test Error

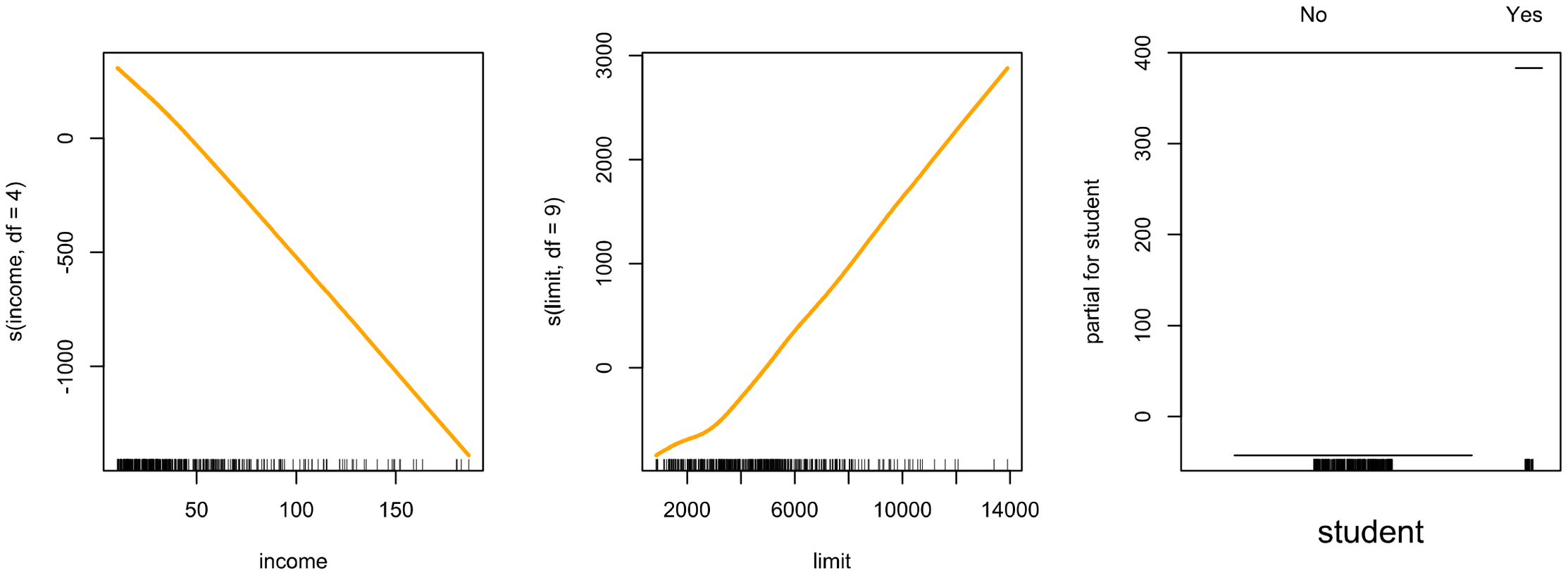
```
> gam.mod4 = gam(balance~s(income,df=4)+s(limit,df=4)+student,data=Credit[train,])
> gam.mod5 = gam(balance~s(income,df=4)+s(limit,df=9)+student,data=Credit[train,])
> gam.mod6 = gam(balance~s(income,df=4)+s(limit,df=16)+student,data=Credit[train,])
> pred.mod4 = predict(gam.mod4,newdata=Credit[test,])
> pred.mod5 = predict(gam.mod5,newdata=Credit[test,])
> pred.mod6 = predict(gam.mod6,newdata=Credit[test,])
> mse4 = mean((pred.mod4-balance[test])^2)
> mse5 = mean((pred.mod5-balance[test])^2)
> mse6 = mean((pred.mod6-balance[test])^2)
> c(mse4,mse5,mse6)

[1] 5417.886     4691.760   4829.545
```

- gam.mod5 has the smallest test error; note that it is also a substantial improvement over the previous smallest error

GAM With Smoothing Splines

- Finally, we again refit the best GAM model to the full training dataset

```
> gam.mod5 = gam(balance~s(income,df=4)+s(limit,df=9)+student)
> par(mfrow=c(1,3))
> plot(gam.mod5,col="orange",lwd=2)
```



GAM Classification

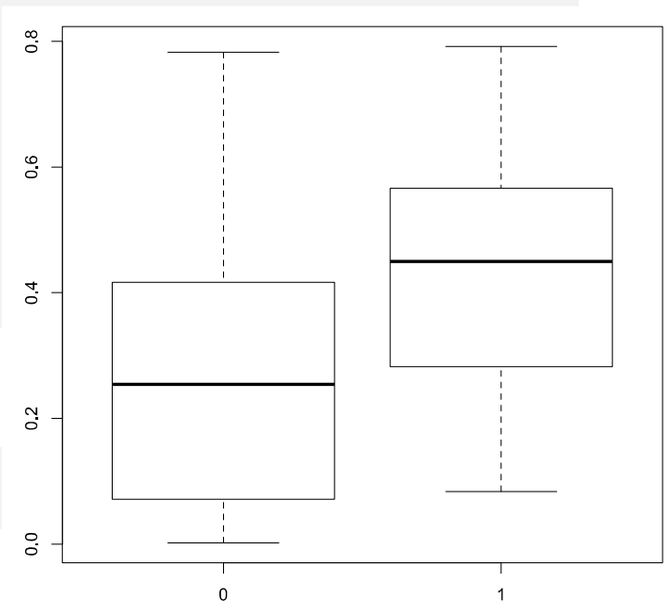
- GAMs can also be straightforwardly applied to classification
- Consider again our Credit classification problem. Within the gam package, logistic regression is achieved by specifying the option family="binomial".
- Here is a worked example that includes

```
> gam.mod5 = gam(balance~s(income,df=4)+s(limit,df=9)+student, family="binomial")
> par(mfrow=c(1,3))
> plot(gam.mod5,col="orange",lwd=2)
```

- Then the predicted probability for class label 1 is

```
> gam.pred = predict(gam.mod,newdata=Credit[test,],type="response")
> table(gam.pred>0.5,y[test])

             0      1
FALSE        118    43
TRUE         12     27
```

```
> boxplot(gam.pred~y[test])
```



Summary

- The use of penalised/shrinkage methods in classification, and the use of the lasso for binary classification via logistic regression
- The pitfalls of high-dimensional datasets, in particular why methods like least squares fail
- Penalised methods tend to work better in high dimensions but there is always a danger of fitting to noise; results must be interpreted carefully
- Generalised additive models (GAMs) allow nonlinear functions of the features
- GAMs are a useful compromise between linear models and even more flexible approaches like random forests and boosting
- We used natural splines and smoothing splines to capture nonlinearity
- Model selection can be carried out by classical statistical methods or by considering test error