ID5059 L03 - more basis functions

C. Donovan

Today

- OOB
- Variants of basis functions: b-spline, bins
- Multidimensional bases: b-spline, others...
- Hyper-parameters and *tuning*

More generalisation error

- Previously saw k-fold cross validation
- A similar approach arises from bootstrapping: the *Out Of Bag* (OOB) error
- Bootstrapping...

My favourite equation

A model structure that will recur throughout this course (and almost every statistics course) is the apparently simple:

$$\mathbf{y} = f(\mathbf{X}, \boldsymbol{\theta}) + \mathbf{e}$$

If we are able to specify f as additive combination of the variables represented by $\mathbf{X}$ then our problem appears very simple:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$

Note ordinary multiple linear regression, analysis of variance, analysis of covariance, t-tests, polynomial regression and others fall under this model type - it is the specification of the bases in $\mathbf{X}$ that is important.

Basis functions

- functions which are combined to produce more complex functions.
- usually simple themselves, but combine to create complex functions.
- several common methods may be viewed from the basis function perspective: polynomial regression, splines, (as we will see) regression trees, bin-smoothing, wavelet analysis, fourier analysis + many more.

Basis functions: piecewise constants

Divide the x-region(s) into K regions R_j, each having the same number of data points. Basis functions are:

$$b_j(x) = \begin{cases} 0 & x \notin R_j \\ 1 & x \in R_j \end{cases}$$

Now multiply each basis by the average of the outputs in that region.

This gives a discontinuous function, but is easy to derive and interpret.

Basis functions: piecewise constants

```
data(mcycle, package = "MASS")

# going to bin the data into 10 bins
binData <- cut(mcycle$times, breaks = seq(0, 60, length = 10)) %>%
  as.numeric() %>% as.data.frame() %>%
  rename(bin = ".")

# expand out to dummy variable to give a design matrix
binMatrix <- binData %>% fastDummies::dummy_cols(select_columns = "bin") %>%
  select(-bin) %>% mutate(y = mcycle$accel)
```

Basis functions: piecewise constants

```
binMatrix
```

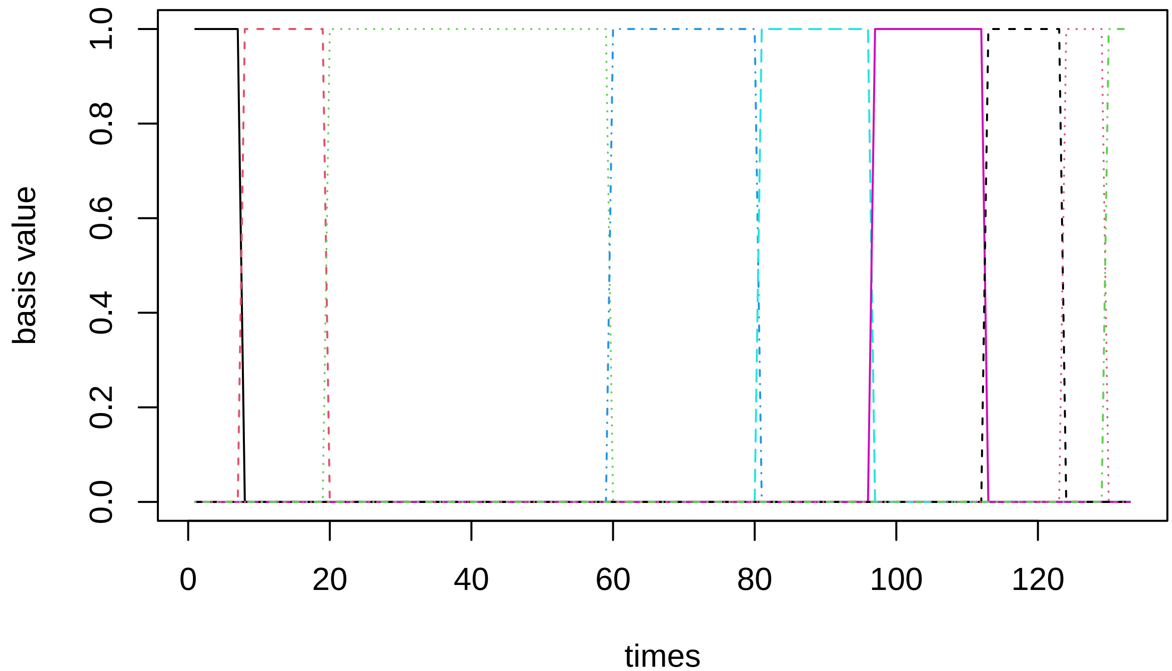
```
##     bin_1 bin_2 bin_3 bin_4 bin_5 bin_6 bin_7 bin_8 bin_9     y
## 1       1     0     0     0     0     0     0     0     0   0.0
## 2       1     0     0     0     0     0     0     0     0  -1.3
## 3       1     0     0     0     0     0     0     0     0  -2.7
## 4       1     0     0     0     0     0     0     0     0   0.0
## 5       1     0     0     0     0     0     0     0     0  -2.7
## 6       1     0     0     0     0     0     0     0     0  -2.7
## 7       1     0     0     0     0     0     0     0     0  -2.7
## 8       0     1     0     0     0     0     0     0     0  -1.3
## 9       0     1     0     0     0     0     0     0     0  -2.7
## 10      0     1     0     0     0     0     0     0     0  -2.7
## 11      0     1     0     0     0     0     0     0     0  -1.3
## 12      0     1     0     0     0     0     0     0     0  -2.7
## 13      0     1     0     0     0     0     0     0     0  -2.7
## 14      0     1     0     0     0     0     0     0     0  -2.7
## 15      0     1     0     0     0     0     0     0     0  -5.4
## 16      0     1     0     0     0     0     0     0     0  -2.7
## 17      0     1     0     0     0     0     0     0     0  -5.4
## 18      0     1     0     0     0     0     0     0     0   0.0
## 19      0     1     0     0     0     0     0     0     0  -2.7
## 20      0     0     1     0     0     0     0     0     0  -2.7
## 21      0     0     1     0     0     0     0     0     0   0.0
## 22      0     0     1     0     0     0     0     0     0 -13.3
## 23      0     0     1     0     0     0     0     0     0  -5.4
## 24      0     0     1     0     0     0     0     0     0  -5.4
## 25      0     0     1     0     0     0     0     0     0  -9.3
## 26      0     0     1     0     0     0     0     0     0 -16.0
## 27      0     0     1     0     0     0     0     0     0 -22.8
## 28      0     0     1     0     0     0     0     0     0  -2.7
```

```
## 29      0    0    1    0    0    0    0    0    0  -22.8
## 30      0    0    1    0    0    0    0    0    0  -32.1
## 31      0    0    1    0    0    0    0    0    0  -53.5
## 32      0    0    1    0    0    0    0    0    0  -54.9
## 33      0    0    1    0    0    0    0    0    0  -40.2
## 34      0    0    1    0    0    0    0    0    0  -21.5
## 35      0    0    1    0    0    0    0    0    0  -21.5
## 36      0    0    1    0    0    0    0    0    0  -50.8
## 37      0    0    1    0    0    0    0    0    0  -42.9
## 38      0    0    1    0    0    0    0    0    0  -26.8
## 39      0    0    1    0    0    0    0    0    0  -21.5
## 40      0    0    1    0    0    0    0    0    0  -50.8
## 41      0    0    1    0    0    0    0    0    0  -61.7
## 42      0    0    1    0    0    0    0    0    0   -5.4
## 43      0    0    1    0    0    0    0    0    0  -80.4
## 44      0    0    1    0    0    0    0    0    0  -59.0
## 45      0    0    1    0    0    0    0    0    0  -71.0
## 46      0    0    1    0    0    0    0    0    0  -91.1
## 47      0    0    1    0    0    0    0    0    0  -77.7
## 48      0    0    1    0    0    0    0    0    0  -37.5
## 49      0    0    1    0    0    0    0    0    0  -85.6
## 50      0    0    1    0    0    0    0    0    0 -123.1
## 51      0    0    1    0    0    0    0    0    0 -101.9
## 52      0    0    1    0    0    0    0    0    0  -99.1
## 53      0    0    1    0    0    0    0    0    0 -104.4
## 54      0    0    1    0    0    0    0    0    0 -112.5
## 55      0    0    1    0    0    0    0    0    0  -50.8
## 56      0    0    1    0    0    0    0    0    0 -123.1
## 57      0    0    1    0    0    0    0    0    0  -85.6
## 58      0    0    1    0    0    0    0    0    0  -72.3
## 59      0    0    1    0    0    0    0    0    0 -127.2
## 60      0    0    0    1    0    0    0    0    0 -123.1
## 61      0    0    0    1    0    0    0    0    0 -117.9
## 62      0    0    0    1    0    0    0    0    0 -134.0
## 63      0    0    0    1    0    0    0    0    0 -101.9
## 64      0    0    0    1    0    0    0    0    0 -108.4
## 65      0    0    0    1    0    0    0    0    0 -123.1
## 66      0    0    0    1    0    0    0    0    0 -123.1
## 67      0    0    0    1    0    0    0    0    0 -128.5
## 68      0    0    0    1    0    0    0    0    0 -112.5
## 69      0    0    0    1    0    0    0    0    0  -95.1
## 70      0    0    0    1    0    0    0    0    0  -81.8
## 71      0    0    0    1    0    0    0    0    0  -53.5
## 72      0    0    0    1    0    0    0    0    0  -64.4
## 73      0    0    0    1    0    0    0    0    0  -57.6
## 74      0    0    0    1    0    0    0    0    0  -72.3
## 75      0    0    0    1    0    0    0    0    0  -44.3
## 76      0    0    0    1    0    0    0    0    0  -26.8
## 77      0    0    0    1    0    0    0    0    0   -5.4
## 78      0    0    0    1    0    0    0    0    0 -107.1
## 79      0    0    0    1    0    0    0    0    0  -21.5
## 80      0    0    0    1    0    0    0    0    0  -65.6
## 81      0    0    0    0    1    0    0    0    0  -16.0
## 82      0    0    0    0    1    0    0    0    0  -45.6
```

```
## 83      0    0    0    0    1    0    0    0    0  -24.2
## 84      0    0    0    0    1    0    0    0    0    9.5
## 85      0    0    0    0    1    0    0    0    0    4.0
## 86      0    0    0    0    1    0    0    0    0   12.0
## 87      0    0    0    0    1    0    0    0    0  -21.5
## 88      0    0    0    0    1    0    0    0    0   37.5
## 89      0    0    0    0    1    0    0    0    0   46.9
## 90      0    0    0    0    1    0    0    0    0  -17.4
## 91      0    0    0    0    1    0    0    0    0   36.2
## 92      0    0    0    0    1    0    0    0    0   75.0
## 93      0    0    0    0    1    0    0    0    0    8.1
## 94      0    0    0    0    1    0    0    0    0   54.9
## 95      0    0    0    0    1    0    0    0    0   48.2
## 96      0    0    0    0    1    0    0    0    0   46.9
## 97      0    0    0    0    0    1    0    0    0   16.0
## 98      0    0    0    0    0    1    0    0    0   45.6
## 99      0    0    0    0    0    1    0    0    0    1.3
## 100     0    0    0    0    0    1    0    0    0   75.0
## 101     0    0    0    0    0    1    0    0    0  -16.0
## 102     0    0    0    0    0    1    0    0    0  -54.9
## 103     0    0    0    0    0    1    0    0    0   69.6
## 104     0    0    0    0    0    1    0    0    0   34.8
## 105     0    0    0    0    0    1    0    0    0   32.1
## 106     0    0    0    0    0    1    0    0    0  -37.5
## 107     0    0    0    0    0    1    0    0    0   22.8
## 108     0    0    0    0    0    1    0    0    0   46.9
## 109     0    0    0    0    0    1    0    0    0   10.7
## 110     0    0    0    0    0    1    0    0    0    5.4
## 111     0    0    0    0    0    1    0    0    0   -1.3
## 112     0    0    0    0    0    1    0    0    0  -21.5
## 113     0    0    0    0    0    0    1    0    0  -13.3
## 114     0    0    0    0    0    0    1    0    0   30.8
## 115     0    0    0    0    0    0    1    0    0  -10.7
## 116     0    0    0    0    0    0    1    0    0   29.4
## 117     0    0    0    0    0    0    1    0    0    0.0
## 118     0    0    0    0    0    0    1    0    0  -10.7
## 119     0    0    0    0    0    0    1    0    0   14.7
## 120     0    0    0    0    0    0    1    0    0   -1.3
## 121     0    0    0    0    0    0    1    0    0    0.0
## 122     0    0    0    0    0    0    1    0    0   10.7
## 123     0    0    0    0    0    0    1    0    0   10.7
## 124     0    0    0    0    0    0    0    1    0  -26.8
## 125     0    0    0    0    0    0    0    1    0  -14.7
## 126     0    0    0    0    0    0    0    1    0  -13.3
## 127     0    0    0    0    0    0    0    1    0    0.0
## 128     0    0    0    0    0    0    0    1    0   10.7
## 129     0    0    0    0    0    0    0    1    0  -14.7
## 130     0    0    0    0    0    0    0    0    1   -2.7
## 131     0    0    0    0    0    0    0    0    1   10.7
## 132     0    0    0    0    0    0    0    0    1   -2.7
## 133     0    0    0    0    0    0    0    0    1   10.7
```

Basis functions: piecewise constants

```
matplot(as.matrix(binMatrix[,-10]), type = "n", ylab = "basis value", xlab = "times")
matlines(as.matrix(binMatrix[,-10]))
```



Basis functions: piecewise constants

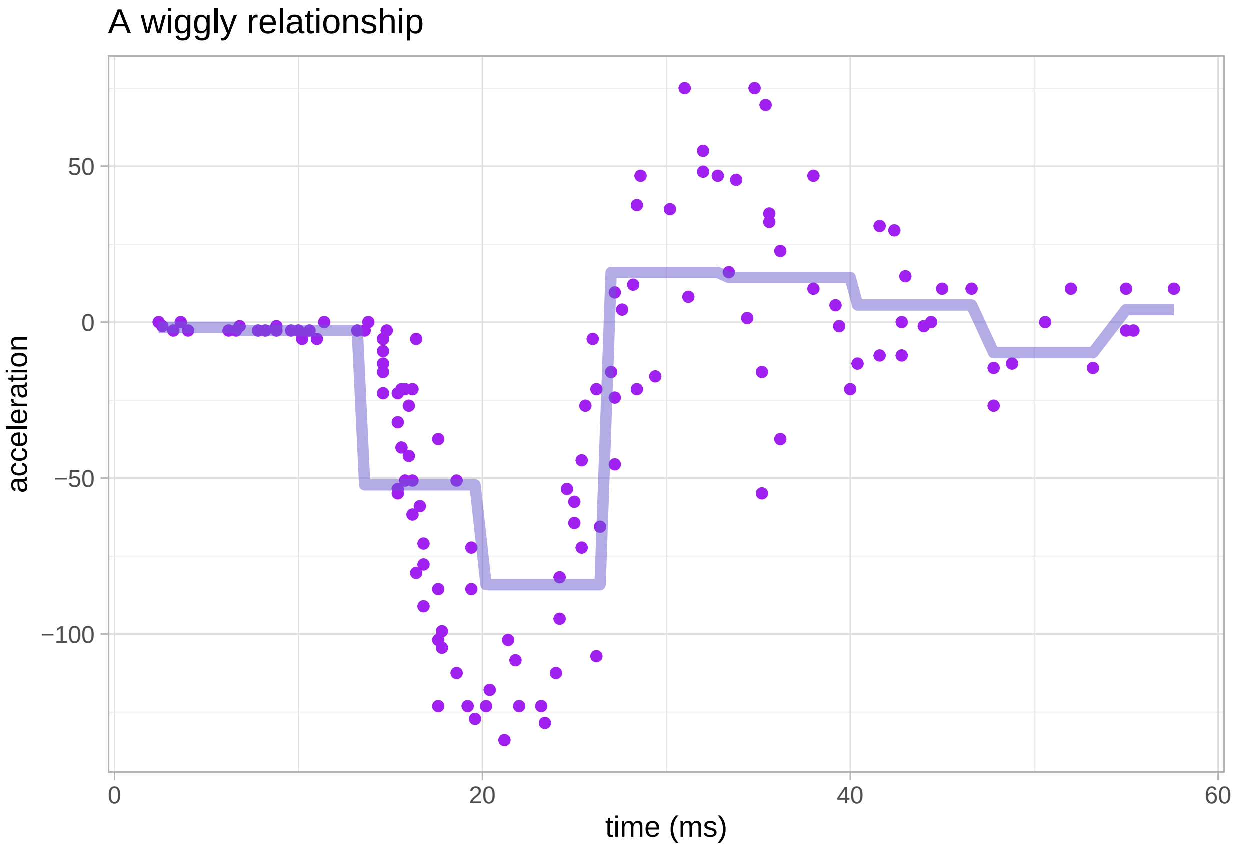
```
# fit this to the data - using OLS aka squared error loss
binFit <- lm(y ~ . -1, data = binMatrix)

# add to the original data for plotting
plottingData <- mcycle %>% mutate(pred = fitted(binFit))

p <- ggplot(data=plottingData) +
    geom_point(aes(x=times, y=accel), col='purple', size=1.5) +
    xlab('time (ms)') + ylab('acceleration') + ggtitle('A wiggly relationship') +
    geom_line(aes(times, pred), col = "slateblue", alpha = 0.5, size = 2) +
    theme_light()

p
```

Basis functions: piecewise constants



A general local univariate basis

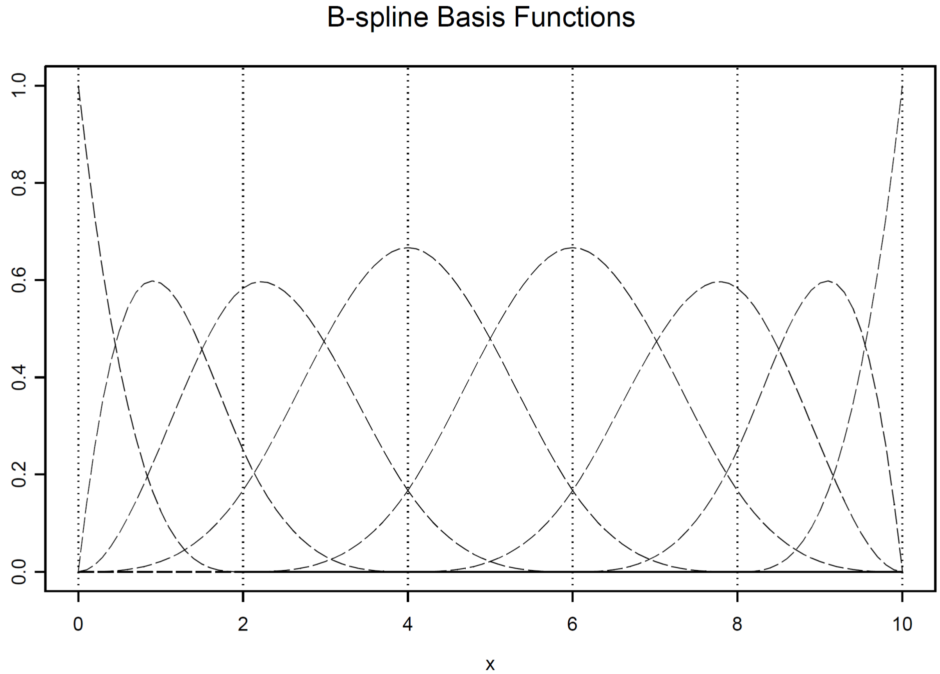
For calculation of a series of b-spline bases (B):

$$B_{i,j}(x) = \frac{x - \delta_i}{\delta_{i+j-1} - \delta_i} B_{i,j-1}(x) + \frac{\delta_{i+j} - x}{\delta_{i+j} - \delta_{i+1}} B_{i+1,j-1}(x)$$

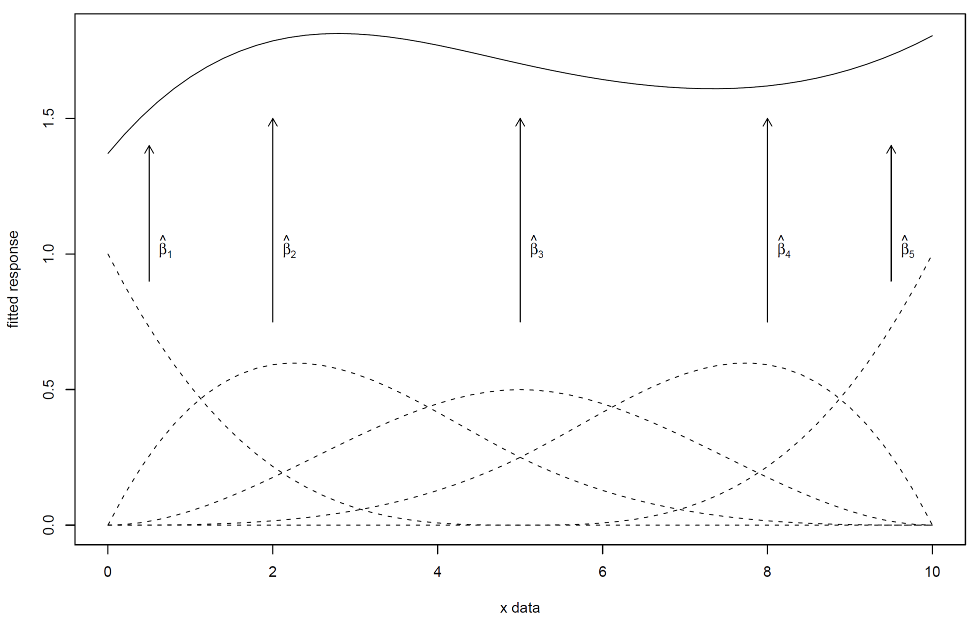
Where we start with:

$$B_{i,1}(x) = \begin{cases} 1 & \delta_i \leq x \leq \delta_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$

Example b-spline basis



Example b-spline basis



B-splines

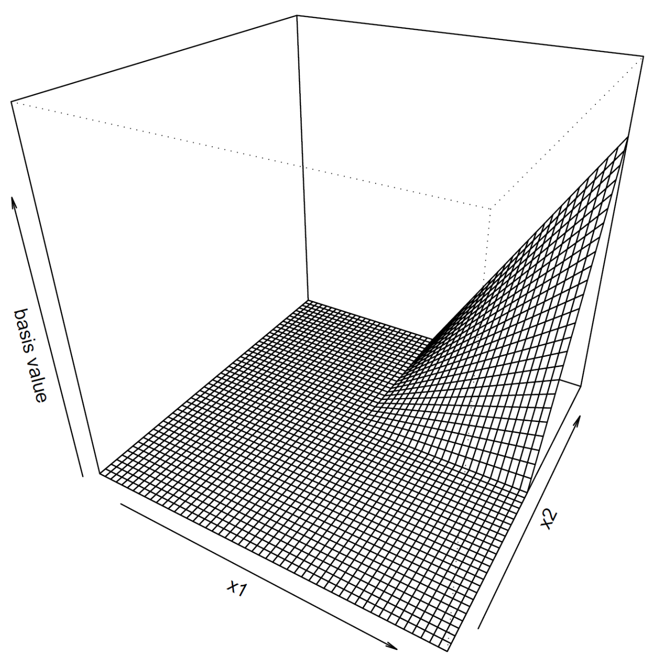
- B-splines are constructed to be zero outside some local region in x
- Calculations are relatively easy

- Curves can be as smooth as desired

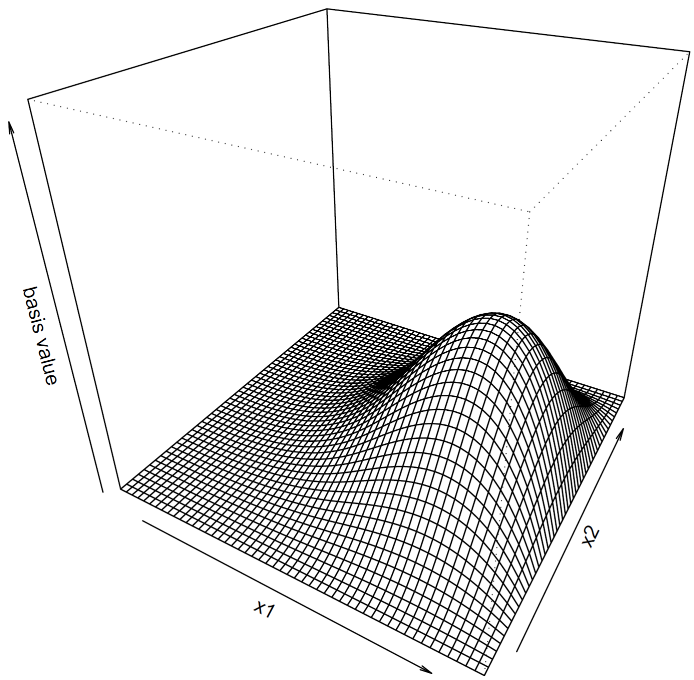
What about more dimensions?

- the result of multiplying a univariate basis function in one dimension against those of another
- results in a shape in higher dimensions
- two b-spline bases of degree 1 (linear) give tensor products that appear to be pyramids or ramps
- more curved bases such as 3rd degree (cubic) b-spline bases will give bump-like tensor products

Example b-spline basis



Example b-spline basis



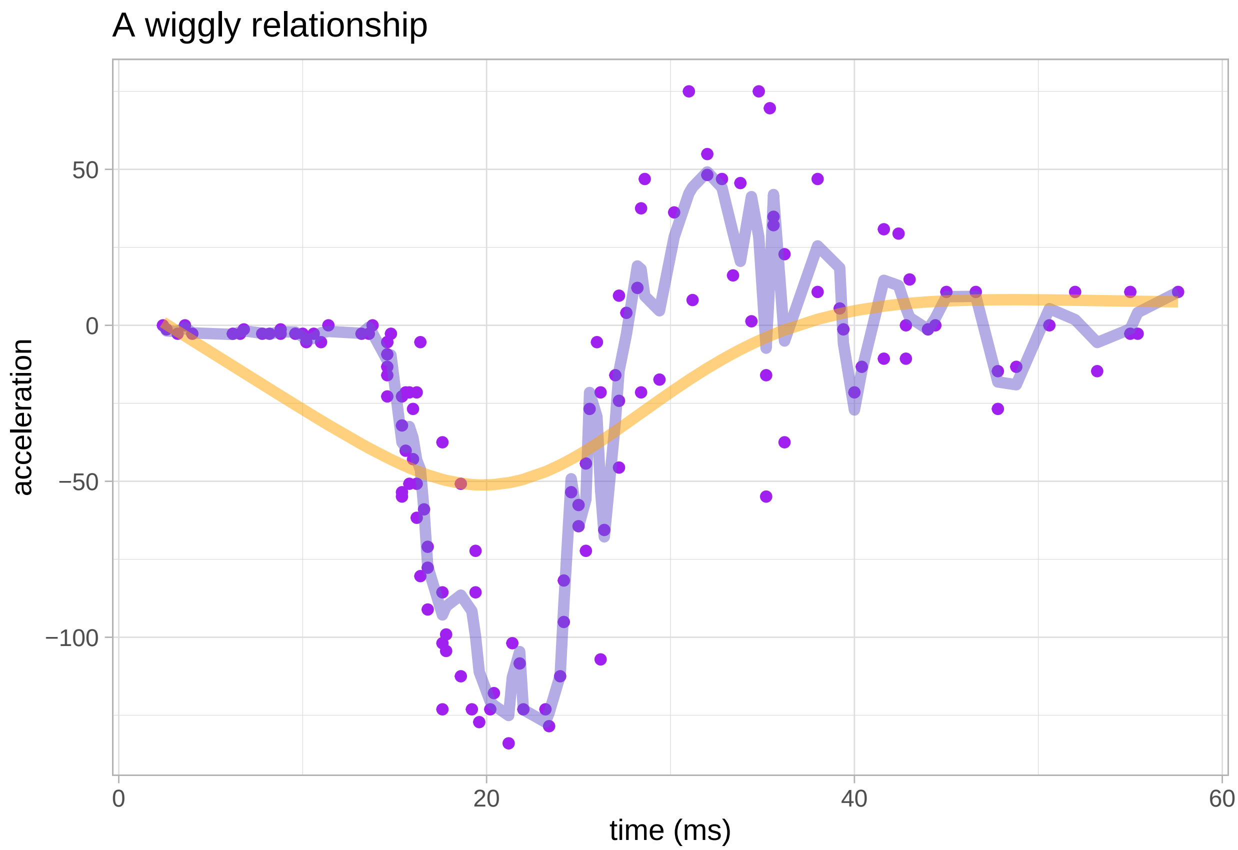
Hyper-parameters

- For a particular f, we estimate its parameters against the loss function
- May be other parameters which govern the complexity/form of f: the *hyper-parameters* e.g. number of bases, band-width of a smoother, size of a neighborhood, learning rate of an NN, etc
- We minimise *generalisation error* to set these i.e. **model selection, rather than model fitting**
- This can be termed hyper-parameter *tuning*

Simple hyper-parameter example

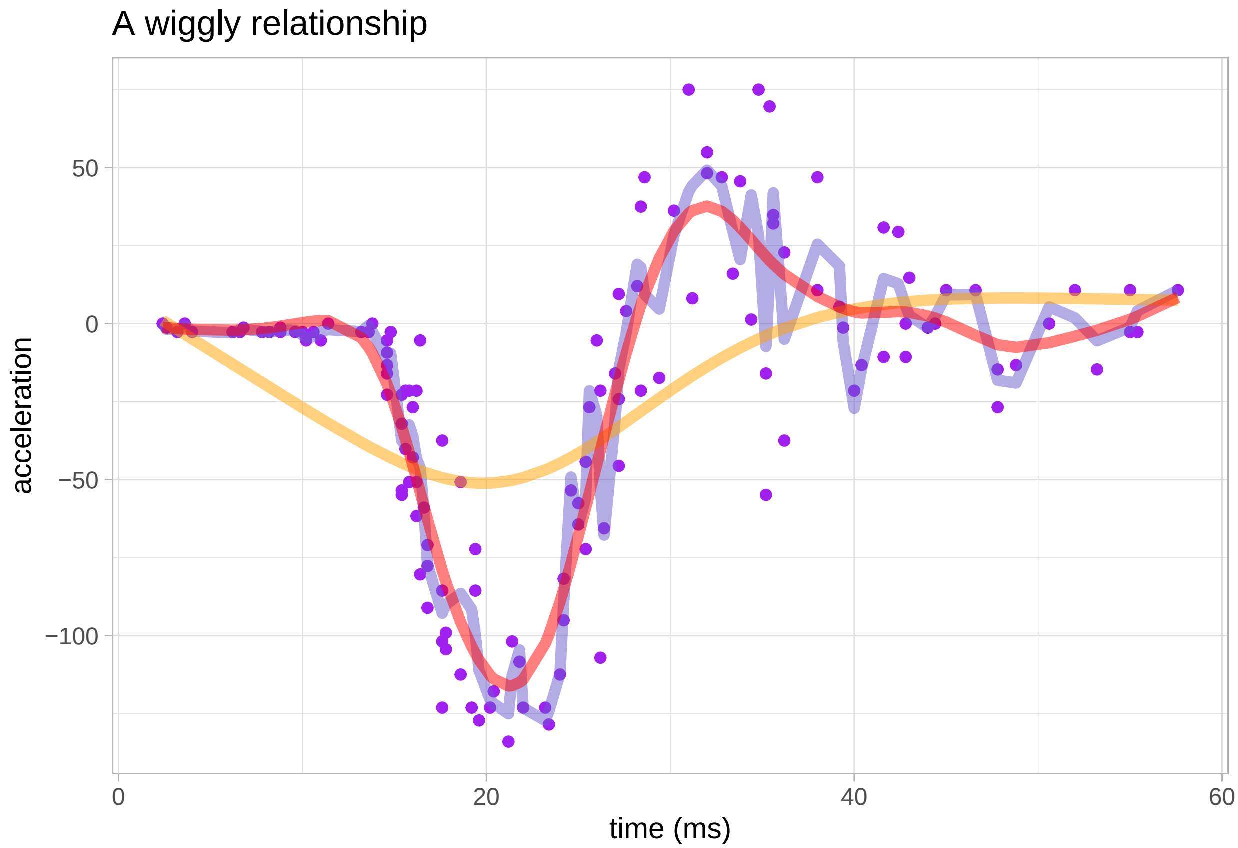
- Splines can have a parameter that governs complexity
- *Smoothing splines* have a smoothing parameter that balances fidelity to data to complexity of curve - a hyper-parameter that needs setting (tuning)

Simpel hyper-parameter example



Simple hyper-parameter example

```
## Warning in smooth.spline(x = mcycle$times, y = mcycle$accel, cv = T): cross-
## validation with non-unique 'x' values seems doubtful
```



Re-iterate

- *loss functions* for estimating model parameters given f
- *generalisation error* for model selection, hyper-parameter tuning

Keywords and Reading

- **bootstrapping and OOB, hyper-parameter, hyper-parameter tuning**
- James *et al*: Section 7.1 - 7.4
- HT&F: Section 5.1, 5.7
- Geron: page 29-31

Work through the associated L03 markdown document