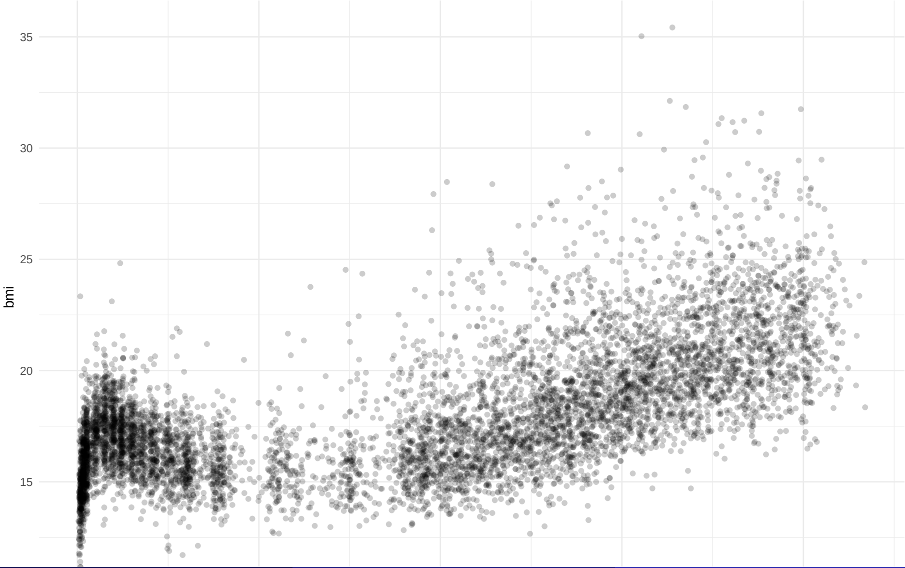
Statistical Learning and Visualization

Non-linear Regression

Maarten Cruyff

BMI Dutch boys

How to predict Body Mass Index from age?



Content

1. Linearity
2. Polynomials
3. Splines
4. Regression trees

Section 1

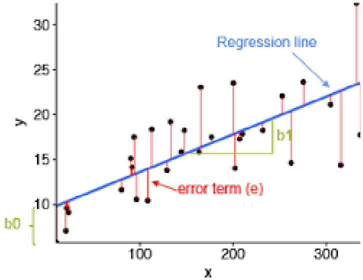
Linearity

Linearity assumption

Assumption of the linear regression model

$$y = \beta_0 + \beta x + \varepsilon, \qquad \varepsilon \sim N(0, \sigma^2)$$

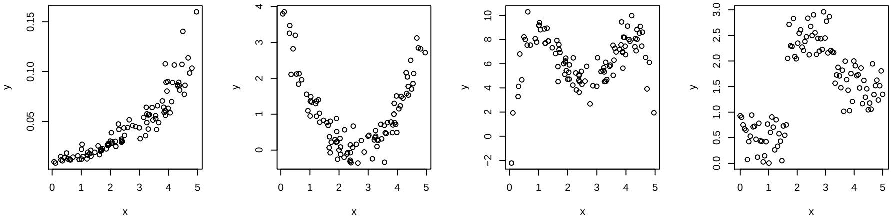
- predictions on straight regression line
- residuals normally distributed and homoscedastic



Non-linearity

Different shapes and forms

- model choice depends on shape and form



Accommodating non-linearity

Different models:

- polynomials

$$y = \beta_0 x^0 + \beta_1 x^1 + \beta_2 x^2 + \beta_3 x^3 + \ldots$$

- splines
  - fit polynomials to non-overlapping regions of X
- tree-based models
  - compute the mean in non-overlapping regions of X

Section 2

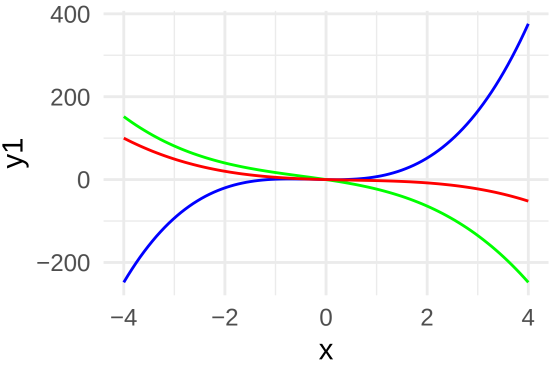
Polynomials

Basis expansion

Expand the feature space with polynomials of X, e.g.

- the cubic polynomial

$$\hat{y} = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$



Making polynomials in R

The straightforward way

- use the function `I()` in the model formula
- `model.matrix()` creates the basis expansion

```
(M <- model.matrix(~ I(x^1) + I(x^2) + I(x^3), data.frame(x = 1:4)))
```

```
  (Intercept) I(x^1) I(x^2) I(x^3)
1           1      1      1      1
2           1      2      4      8
3           1      3      9     27
4           1      4     16     64
attr(,"assign")
[1] 0 1 2 3
```

Multicollinearity

Potential problem with `I()`

- multicollinearity, i.e. high correlation between x, x^2, x^3, etc.

Correlations between polynomials:

```
round(cor(M[, -1]), 3)
```

```
       I(x^1) I(x^2) I(x^3)
I(x^1)  1.000  0.984  0.951
I(x^2)  0.984  1.000  0.991
I(x^3)  0.951  0.991  1.000
```

Orthogonal expansion

The function `poly(x, degree = 3)` creates on orthogonal basis

```
(P <- model.matrix( ~ poly(x, 3), data = data.frame(x = 1:4)))
```

```
  (Intercept) poly(x, 3)1 poly(x, 3)2 poly(x, 3)3
1           1  -0.6708204         0.5  -0.2236068
2           1  -0.2236068        -0.5   0.6708204
3           1   0.2236068        -0.5  -0.6708204
4           1   0.6708204         0.5   0.2236068
attr(,"assign")
[1] 0 1 1 1
```

Correlations

```
round(cor(P[, -1]), 3)
```

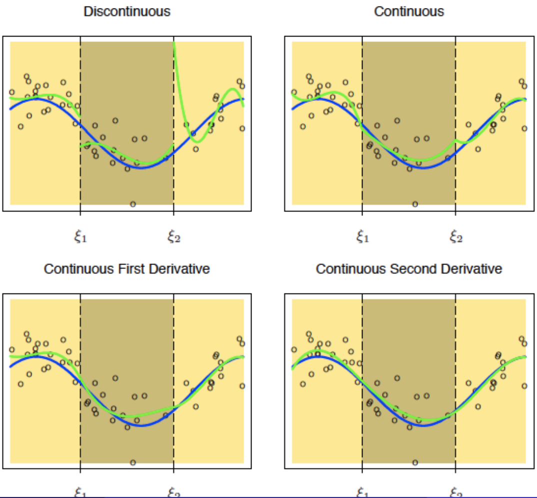
```
            poly(x, 3)1 poly(x, 3)2 poly(x, 3)3
poly(x, 3)1           1           0           0
poly(x, 3)2           0           1           0
poly(x, 3)3           0           0           1
```

Section 3

Splines

B-splines

- Place a number of knots ξ that divide X in non-overlapping regions
- fit cubic polynomial to each region and connect lines by equating 1st and 2nd derivative



Fitting cubis splines in R

Formula for generating B-spline basis matrix in R (package `splines`)

```
bs(x, df = NULL, knots = NULL, degree = 3) # cubic spline

ns(x, df = NULL, knots = NULL, degree = 3) # natural cubic spline
```

- `degree = 3` for cubic polynomial (default)
- `df` number of knots (`df = degree + number of knots`)
- `knots` position of knots in percentiles
- natural cubic spline is linear beyond the boundary knots

Basis matrix cubic spline with df = 4

```r
bs(1:4, df = 4)
```

```
              1         2          3          4
[1,] 0.00000000 0.0000000 0.00000000 0.00000000
[2,] 0.51851852 0.3703704 0.07407407 0.00000000
[3,] 0.07407407 0.3703704 0.51851852 0.03703704
[4,] 0.00000000 0.0000000 0.00000000 1.00000000
attr(,"degree")
[1] 3
attr(,"knots")
50%
2.5
attr(,"Boundary.knots")
[1] 1 4
attr(,"intercept")
[1] FALSE
attr(,"class")
[1] "bs"     "basis"  "matrix"
```

Smoothing splines

Highly flexible spline

1. A *knot* ξ_i for each unique value x_i
2. `df` controls wiggliness (value between 1 and $\# \; x_i$)

Fitting smooth splines in R

```
smooth.spline(y ~ x, df = <nr>)
smooth.spline(y ~ x)
```

- 1st: user-specified `df`
- 2nd: optimal `df` determined with cross-validation

Section 4

Regression trees

Binary recursive partioning algorithm

1. Partition the feature space in distinct, non-overlapping regions
2. Compute the mean of all observations within a region
3. Select the partition that minimizes the MSE
4. Continue partitioning until a stopping criterion is reached

Tree function `rpart()` from package `rpart`

```
reg_tree <- rpart(y ~ x, method = "anova")
plot(reg_tree)
text(reg_tree)
```

Warning: trees tend to overfit, more on this in classification

Example with one feature

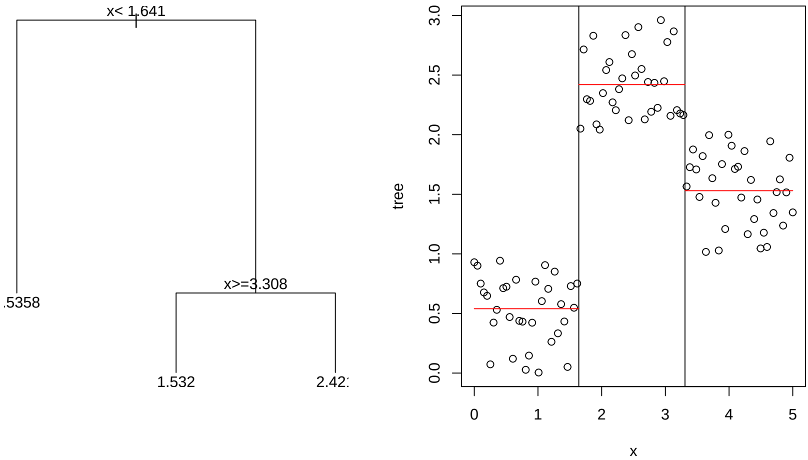


Figure 1: Tree representation (left) and its predictions (right)

Example with two features

Different way of looking at interactions

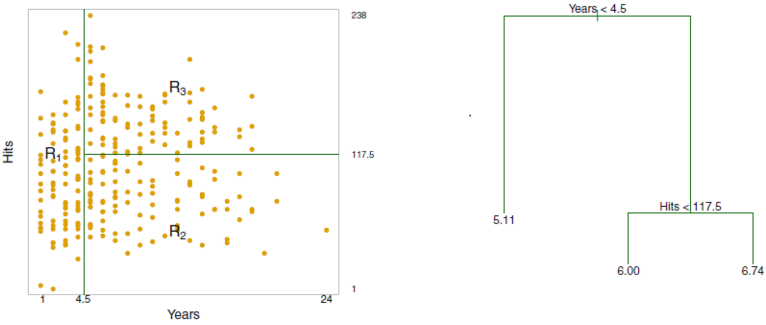


Figure 2: Salaries of baseball players (ISLR)

Preview Lab 2A

Topics

- polynomials
- splines
- trees

Next lab (Feature selection) features interactions